NEC Network Queuing System V (NQSV) User's Guide

[API]

Proprietary Notice

The information disclosed in this document is the property of NEC Corporation (NEC) and/or its licensors. NEC and/or its licensors, as appropriate, reserve all patent, copyright and other proprietary rights to this document, including all design, manufacturing, reproduction, use and sales rights thereto, except to the extent said rights are expressly granted to others.

The information in this document is subject to change at any time, without notice.

Preface

This guide explains API of NEC Network Queuing System V (NQSV) job management system.

The manual of NEC Network Queuing System V (NQSV) is composed by following user's guides.

Name	Contents
NEC Network Queuing System V (NQSV) User's Guide [Introduction]	This guide explains the overview of NQSV and configuration of basic system.
NEC Network Queuing System V (NQSV) User's Guide [Management]	This guide explains the various management functions of the system.
NEC Network Queuing System V (NQSV) User's Guide [Operation]	This guide explains the various functions that used by general user.
NEC Network Queuing System V (NQSV) User's Guide [Reference]	The command reference guide.
NEC Network Queuing System V (NQSV) User's Guide [API]	This guide explains the C programming interface (API) to control NQSV.
NEC Network Queuing System V (NQSV) User's Guide [JobManipulator]	This guide explains about the scheduler component : JobManipulator.
NEC Network Queuing System V (NQSV) User's Guide [Accounting & Budget Control]	This guide explains the functions of accounting.

February 2018	1st edition
September 2019	4th edition
January 2020	5th edition
July 2020	6th edition
March 2021	7th edition
January 2023	8th edition

Remarks

(1) This manual conforms to Release 1.00 and subsequent releases of the NEC Network Queuing System V (NQSV).

(2) All the functions described in this manual are program products and conform to the following product names and product series numbers:

Product Name	product series numbers
NEC Network Queuing System V (NQSV) /ResourceManager	UWAF00 UWHAF00 (support pack)
NEC Network Queuing System V (NQSV) /JobServer	UWAG00 UWHAG00 (support pack)
NEC Network Queuing System V (NQSV) /JobManipulator	UWAH00 UWHAH00 (support pack)

(3) UNIX is a registered trademark of The Open Group.

(4) Intel is a trademark of Intel Corporation in the U.S. and/or other countries.

(5) OpenStack is a trademark of OpenStack Foundation in the U.S. and/or other countries.

(6) Red Hat OpenStack Platform is a trademark of Red Hat, Inc. in the U.S. and/or other countries.

(7) Linux is a trademark of Linus Torvalds in the U.S. and/or other countries.

(8) Docker is a trademark of Docker, Inc. in the U.S. and/or other countries.

(9) InfiniBand is a trademark or service mark of InfiniBand Trade Association.

(10) Zabbix is a trademark of Zabbix LLC that is based in Republic of Latvia.

(11) All other product, brand, or trade names used in this publication are the trademarks or registered trademarks of their respective trademark owners.

About This Manual

Notation Conventions

The following notation rules are used in this manual:

Omission Symbol	...	This symbol indicates that the item mentioned previously can be repeated. The user may input similar items in any desired number.
Vertical Bar	|	This symbol divides an option and mandatory selection item.
Brackets	{ }	A pair of brackets indicates a series of parameters or keywords from which one has to be selected.
Braces	[ ]	A pair of braces indicate a series of parameters or keywords that can be omitted.

Glossary

Term	Definition
Vector Engine (VE)	The NEC original PCIe card for vector processing based on SX architecture. It is connected to x86-64 machine. VE consists of more than one core and shared memory.
Vector Host (VH)	The x86-64 architecture machine that VE connected.
Vector Island (VI)	The general component unit of a singe VH and one or more VEs connected to the VH.
Batch Server (BSV)	Resident system process running on a Batch server host to manage entire NQSV system.
Job Server (JSV)	Resident system process running on each execution host to manage the execution of jobs.
JobManipulator (JM)	JobManipulator is the scheduler function of NQSV. JM manages the computing resources and determines the execution time of jobs.
Accounting Server	Acconting server collects and manages account information and manages budgets.
Request	A unit of user jobs in the NQSV system. It consists of one or more jobs. Requests are managed by the Batch Server.
Job	A job is an execution unit of user job. It is managed by Job Server.

Logical Host	A logical host is a set of logical (virtually) divided resources of an execution host.
Queue	It is a mechanism that pools and manages requests submitted to BSV.
BMC	Board Management Controller for short. It performs server management based on the Intelligent Platform Management Interface (IPMI).
HCA	Host Channel Adapter for short. The PCIe card installed in VH to connect to the IB network.
IB	InfiniBand for short.
MPI	Abbreviation for Message Passing Interface. MPI is a standard for parallel computing between nodes.
NIC	Network Interface Card for short. The hardware to communicate with other node.

Contents

1. How to use NQSV/API

1.1. Overview of NQSV/API

NQSV/API is the C language API to refer the information and control with batch server. It is able to create the original client command and scheduler by using this API.

1.2. Install Location

Install location of NQSV/API include file and library files:

- Include file
  /opt/nec/nqsv/include/nqsv.h
- /opt/nec/nqsv/include/nqsv.h Library files

/opt/nec/nqsv/lib64/libnqsv.a	Static Link library
/opt/nec/nqsv/lib64/libnqsv.so	Dynamic Link library

1.3. How to use NQSV/API

1.3.1. How to compile

To use NQSV/API, include the header file nqsv.h on the source code and link the NQSV library file libnqsv.a or libnqsv.so.

Example of compile command line: it compiles apitest.c

```
cc -I /opt/nec/nqsv/include apitest.c /opt/nec/nqsv/lib64/libnqsv.a
```

1.3.2. Connection to batch server

NQSV/API performs reference, control and other processing with batch server, after TPC connection is established with batch server. Therefore when you use the following various NQSV/API functions, it is necessary to call NQSconnect() at first. And when you terminates the NQSV/API processing, call NQSdisconnect() to terminates the connection.

1.3.3. Result Code

Each functions of NQSV/API return the result of its processing by nqs_res structure. nqs_res includes error number and error message.

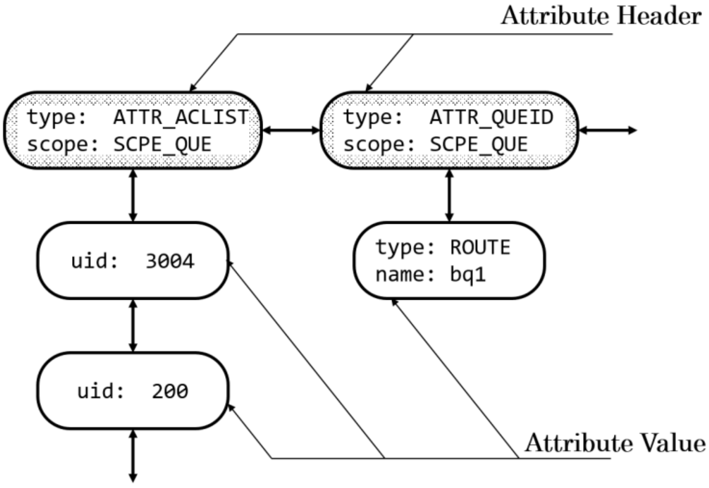
1.3.4. Attributes

NQSV/API can set/get various information of queues, requests, jobs, etc. managed by batch server. The information unit to set/get is called "attribute". The attributes of queue, request and other objects managed by batch server are listed in 4. Attribute Tables.

Each attributes have "scope" which indicates the area that the attribute is applied. The scope identifiers associated with each attributes are as follows:

Scope	Scope Identifiers
Batch Server	SCPE_BSV
Scheduler	SCPE_SCH
Job Server	SCPE_JSV
Execution Host	SCPE_HST
Queue	SCPE_QUE
Request	SCPE_REQ
Job	SCPE_JOB
Process	SCPE_PRC
Node Group	SCPE_NGRP
Workflow	SCPE_WFL

To set/get attributes, make an attribute list of target attributes and pass the attribute list to attribute control API functions. NQSalist() is used to make attribute list. Attribute list has the data structure including chained attribute header data. The attribute header data can contain attribute values. Some attribute list has multiple attribute values, and it is connected by list structure. This attribute value connection is called as "chain".



The attribute list data is dynamically made in the heap area, it is necessary to free by NQSafree() function.

To set/get attributes, use NQSV/API function as following sequence.

- To get attributes

1. Make attribute list (NQSalist())
2. Use attribute control function for each information type with ATTROP_GET mode. (NQSattr*xxx*())
3. Get attribute value from the attribute list data. (NQSaref())

- To set attributes
  1. Make attribute list (NQSalist())
  2. Set attribute values to the attribute list. (NQSaadd())
  3. Use attribute control function for each information type with ATTROP_SET mode. (NQSattr*xxx*())

1.3.5. Event

To get various NQSV events that occur in batch server, use NQSV/API functions as follows using the socket file descriptor returned by NQSconnect().

1. Connect to the batch server by NQSconnect().
2. Set event types to get by NQSevflt().
3. Wait an input to the socket file descriptor by select() or poll().
4. After detecting input to the socket, get the event data using NQSevent().
5. Handle the event data included in nqs_event structure.
6. Return to 3.

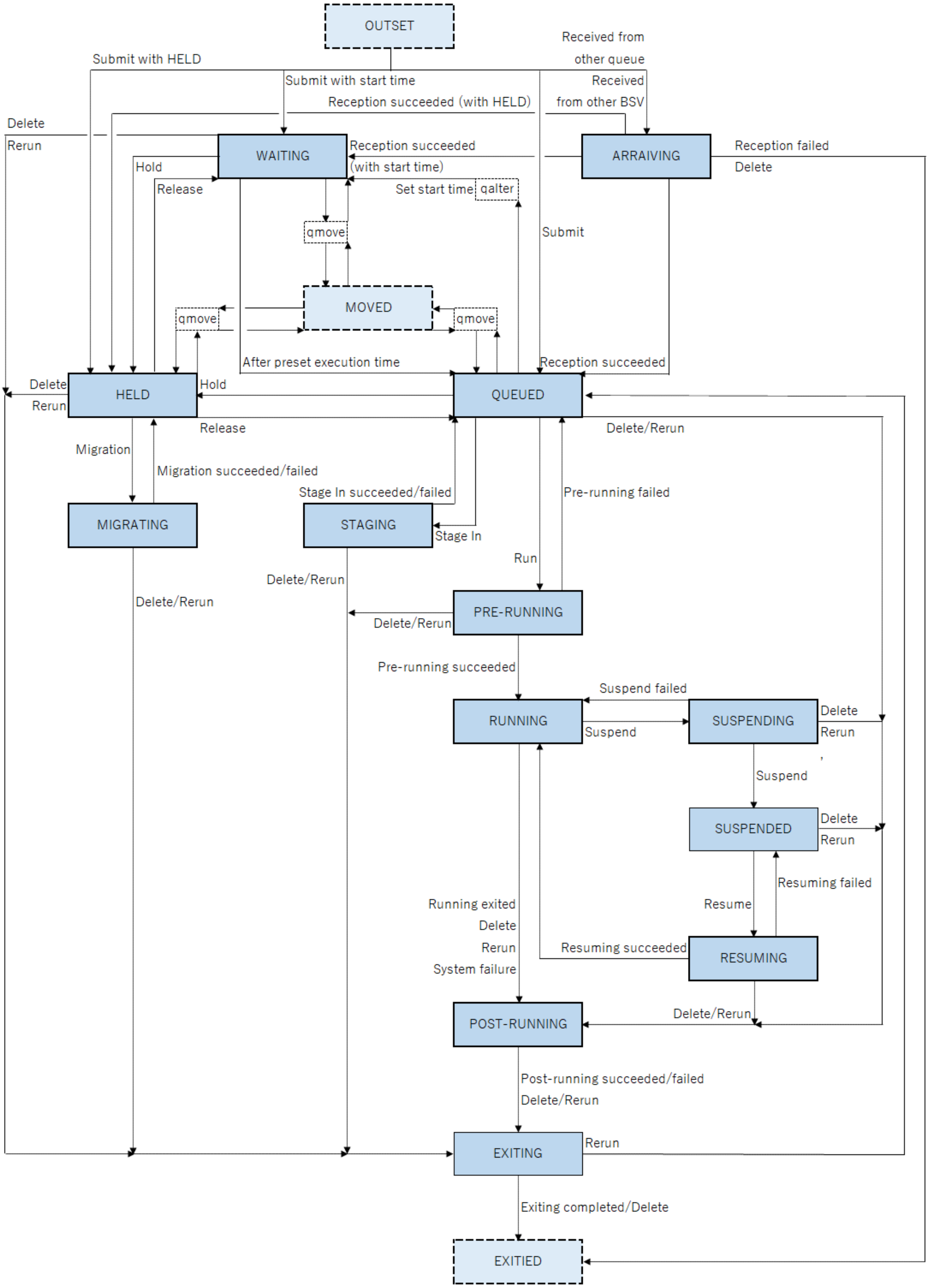
2. State Transition of Request

2.1. State Transition

The following figure shows the states of the request in batch queue and interactive queue, and its state transition. The state enclosed by solid line is the real state of the request and the state enclosed by dotted line is the virtual state. In the figure, the arrows show the direction of state transition and the reasons of the state transitions are described with the arrows.

When a request's state transition occurs, the request state type event will be sent to the NQSV/API client. The request state type event includes the following information and NQSV/API client program can trace requests using the event information.

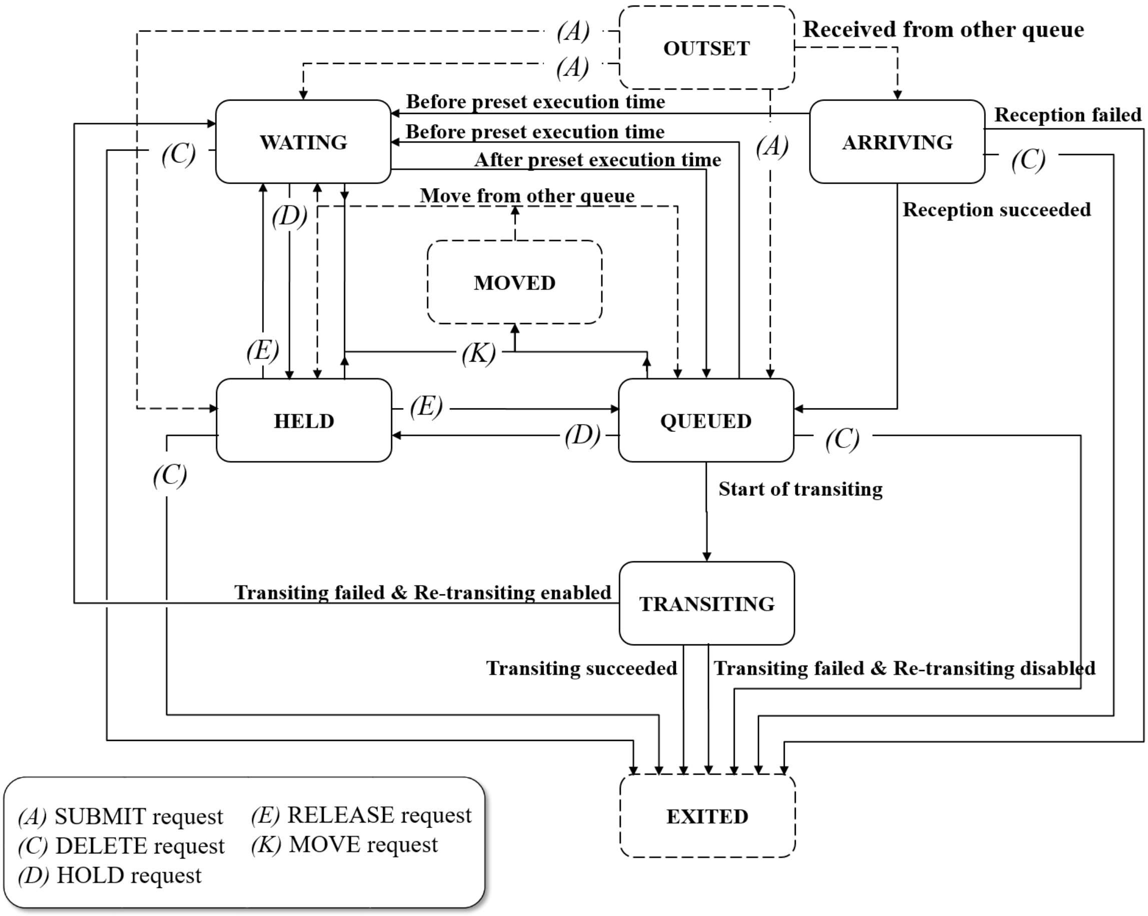
- Current state of the request
- Previous state of the request
- Reason of the state transition


OUTSET
Received from
other queue
Submit with HELD
Submit with start time
Received
Reception succeeded (with HELD)
from other BSV
Delete
Rerun
WAITING
Reception succeeded
(with start time)
ARRAIVING
Reception failed
Delete
Hold
Release
Set start time
qalter
qmove
Submit
MOVED
qmove
qmove
After preset execution time
Reception succeeded
Delete
Rerun
HELD
Hold
QUEUED
Release
Delete/Rerun
Migration
Migration succeeded/failed
Stage In succeeded/failed
Pre-running failed
MIGRATING
STAGING
Stage In
Run
Delete/Rerun
Delete/Rerun
PRE-RUNNING
Delete/Rerun
Pre-running succeeded
Suspend failed
RUNNING
Suspend
SUSPENDING
Delete
Rerun
Suspend
SUSPENDED
Delete
Rerun
Resuming failed
Running exited
Resume
Delete
Rerun
Resuming succeeded
RESUMING
System failure
POST-RUNNING
Delete/Rerun
Post-running succeeded/failed
Delete/Rerun
EXITING
Rerun
Exiting completed/Delete
EXITIED

2.2. State Transition in Routing Queue

The figure to the right illustrates states where a request can accept in the routing queue.
No API event is created pertaining to the request state transition made in the routing queue.

Batch Request Status Transition in Routing Queue



2.3. State of Request

NQSV request can have the states as follows.

- ARRIVING

  The request is being received from a routing queue.
- TRANSITING

  The request is being transferred from a routing queue to another queue.
- WAITING

  The request is waiting until the time when the execution is scheduled to start.
-

- QUEUED

  The request is queued and scheduled for execution. It will transit to RUNNING state when the batch scheduler signals to start execution.

- STAGING

  Batch jobs or network request are generated. The stage-in files are transferred from client hosts to the execution host.

- PRE-RUNNING

  The information required to execute batch jobs is being transferred to each job server. The master job synchronizes with all related slave jobs before execution. Pre-processing is performed and if an error occurs during processing, the request will return to the QUEUED state after cancelling each process up to that point backwards.

- RUNNING

  Batch jobs associated with the batch request is being executed. In case of the MPI job, it transits to POST-RUNNING state as soon as the master job is finished.

  The RUNNING state will be maintained as long as the master job is executed even though all slave jobs are completed to execute.

  The finishing of the slave jobs does not give any influence to the state.

  In case of the distributed jobs (non MPI job), the request transits to the POST-RUNNING state when all batch jobs are finished.

- POST-RUNNING

  Post-processing after completing execution of batch jobs is performed.

- EXITING

  The standard/error output file and stage-out file of the request are transferred from the execution host to the client host.

- CHKPNTING

  A periodic checkpoint for the request has been issued. After storing the checkpoint restart file, the batch jobs continue execution.

  Checkpoint processing is performed and if an error occurs during processing, the request will return to the RUNNING state after cancelling each process up to that point backwards.

- HOLDING

  A checkpoint request has been issued. After storing the checkpoint restart file, the batch jobs will be ended.

  Checkpoint processing is performed and if an error occurs during processing, the request will return to the RUNNING state after cancelling each process up to that point backwards.

- HELD

  The request is not the target of scheduling and does not accept "run" or "restart" request from the scheduler.

  If a checkpoint request has been issued during RUNNING state, the checkpoint restart file is generated in this (HELD) state.

- RESTARTING

  The request is being restarted from the checkpoint file previously stored.

  Restart processing is performed and if an error occurs during processing, the state will return to the QUEUED state after cancelling each process up to that point backwards.

- SUSPENDING

  The request is waiting until all of its batch jobs are stopped.

- SUSPENDED

  All the batch jobs for the request are stopped.

- RESUMING

  The request is waiting until all of its batch jobs are restarted.

- MIGRATING

  The batch jobs associated with the request are being moved to other job servers.

The following states are virtual states.

- EXITED

  This virtual state is defined as the end of a state transition request in which the request is gone.

-

- MOVED

  This virtual state is defined as the end point before movement or the beginning point after movement where a request has already moved from one queue to another. The request state is identical before and after movement. For example, when a HELD request is moved to another queue, the state of the request in the destination queue is also HELD. This is also true to QUEUED and WAITING queues.

2.4. Request Stall

A request with a child job temporarily stops changing its status and waits until its link is established again if the link breaks between the batch server and the job server that controls a job in an execution node. The request in this status is called a "stalled" request.

It is possible to do only deletion or re-running to the stalled request.

3. API Events

3.1. Event Types

Event Types	Event Identifier	Reference	Timing of Occurrence
Batch server related events			
NQSEVT_BSV	NQSEVT_BSV_LINKDOWN	(none)	The API link is broken.
Execution host state related events			
NQSEVT_HST	NQSEVT_HST_ACTIVE	evt_hst	The execution host is activated.
	NQSEVT_HST_INACTIVE		The execution host is inactivated.
Job server related events			
NQSEVT_JSV	NQSEVT_JSV_LINKUP	evt_jsv	A link with the job server is established.
	NQSEVT_JSV_LINKDOWN		A link with the job server is broken.
	NQSEVT_JSV_ATTACH		The job server is registered to the batch server.
	NQSEVT_JSV_DETACH		The job server's registration is removed from the batch server.
	NQSEVT_JSV_BOUND		The job server is bound to a queue (first time only).
	NQSEVT_JSV_UNBOUND		The job server is unbound from all queues.
Queue state related events			
NQSEVT_QST	NQSEVT_QST_ACTIVE	evt_qst	The queue is active (executable).
	NQSEVT_QST_INACTIVE		The queue is inactive (unexcitable).
	NQSEVT_QST_ENABLE		The queue is enabled.
	NQSEVT_QST_DISABLE		The queue is disabled.
	NQSEVT_QST_CREATE		The queue is created.

	NQSEVT_QST_DELETE		The queue is deleted.
	NQSEVT_QST_BINDJSV		A job server is connected to the queue. (first time only)
	NQSEVT_QST_BINDSCH		A scheduler is connected to the queue.
	NQSEVT_QST_BINDNGRP		A node group is connected to the queue
	NQSEVT_QST_UNBINDJSV		A job server is disconnected from the queue.
	NQSEVT_QST_UNBINDSCH		A scheduler is disconnected from the queue.
	NQSEVT_QST_UNBINDNGRP		A node group is disconnected from the queue.
Queue attribute related events			
NQSEVT_QAT	NQSEVT_QAT_RESLIM	evt_qat	The resource limit value of the queue is changed.
	NQSEVT_QAT_PRIORITY		The queue priority is changed.
	NQSEVT_QAT_RSTFDIR		The restart file storage directory is changed
Request state related events			
NQSEVT_RST	NQSEVT_RST_ARRIVING	evt_rst	The request is ARRIVING.
	NQSEVT_RST_WAITING		The request is WAITING.
	NQSEVT_RST_QUEUED		The request is QUEUED.
	NQSEVT_RST_STAGING		The request is STAGING.
	NQSEVT_RST_PRERUNNING		The request is PRERUNNING.
	NQSEVT_RST_RUNNING		The request is RUNNING.
	NQSEVT_RST_POSTRUNNING		The request is POSTRUNNING.
	NQSEVT_RST_EXITING		The request is EXITING.
	NQSEVT_RST_EXITED		The request is EXITED.
	NQSEVT_RST_CHKPNTING		The request is CHKPNTING.
	NQSEVT_RST_HOLDING		The request is HOLDING.
	NQSEVT_RST_HELD		The request is HELD.

	NQSEVT_RST_RESTARTING		The request is RESTARTING.
	NQSEVT_RST_SUSPENDING		The request is SUSPENDING.
	NQSEVT_RST_SUSPENDED		The request is SUSPENDED.
	NQSEVT_RST_RESUMING		The request is RESUMING.
	NQSEVT_RST_MIGRATING		The request is MIGRATING.
	NQSEVT_RST_MOVED		The request is MOVED.
	NQSEVT_RST_STALLED		The request is stalled.
	NQSEVT_RST_REVIVED		The request is revived.
Request attribute related events			
NQSEVT_RAT	NQSEVT_RAT_RESLIM	evt_rat	The resource limit value of the request is changed.
	NQSEVT_RAT_ACCTCODE		The account code of the request is changed.
	NQSEVT_RAT_EXETIME		The execution time of the request is changed.
	NQSEVT_RAT_RSVTIME		The reservation time of the request is changed.
	NQSEVT_RAT_PRIORITY		The request priority of the request is changed.
	NQSEVT_RAT_MIGRATABL		The migration attribute of the request is changed.
	NQSEVT_RAT_HOLDABL		The hold attribute of the request is changed.
	NQSEVT_RAT_MAILADDR		The e-mail address of the request is changed.
	NQSEVT_RAT_KNLPRM		The value of kernel parameter of the request is changed.
Node group related events			
NQSEVT_NGRP	NQSEVT_NGRP_CREATE	evt_ngrp	A node group is created.
	NQSEVT_NGRP_DELETE		A node group is deleted.
	NQSEVT_NGRP_ADDNODE		Job server is added to the node group.
	NQSEVT_NGRP_DELNODE		Job server is removed from the node group.
Template related events			

NQSEVT_TMPL	NQSEVT_TMPL_CREATE	evt_template	A template is created.
	NQSEVT_TMPL_DELETE		A template is deleted.
	NQSEVT_TMPL_MODIFY		The configuration of the template is changed.
Custom resource related events			
NQSEVT_CRS	NQSEVT_CRS_CREATE	evt_crs	A custom resource is created.
	NQSEVT_CRS_DELETE		A custom resource is deleted.
	NQSEVT_CRS_RESCHANGED		The configuration of the custom resource is changed.

* Request state related events and Request attribute related events are sent to the each sub-request of parametric request.

4. Attribute Tables

This section describes attributes that the NQSV handles.

The "Reference" field in the Attribute Table shows an API authority type required to refer to an attribute and the "Alter" field shows the possibility of attribute alteration by NQSattrxxx (3) and an API authority required for attribute alteration.

The API authorities are classified as follows:

Symbol	Description
MGR	API authority of PRIV_MGR or higher
OPE	API authority of PRIV_OPE or higher
USR	API authority of PRIV_USR or higher
USR(R)	API authority of PRIV_USR or higher (The PRIV_USR authority must be the owner of the request or job. The PRIV_GMGR authority must be the group manager of the request or job.)
USR(Q)	API authority of PRIV_USR or higher (The PRIV_USR authority must be allowed to access the queue. The PRIV_GMGR authority must be allowed to access the queue of the managed group.)
X	Reference and alteration of the attribute are prohibited.

Refer the next chapter about the data types.

4.1. Batch server attributes

Attribute name	Attribute type	Type	Reference	Alteration	Chain	Description	Scope
							BSV
Version number	ATTR_VERNO	char *	USR	X	X	Batch server version [Max. length: NQS_LEN_VERSION]	o
Host name	ATTR_HOSTNAME	char *	USR	X	X	Name of batch server host [Max. length: NQS_LEN_HOSTNAME]	o
Machine ID	ATTR_MID	int	USR	X	X	Machine ID of batch server	o
Log file name	ATTR_LOGPATH	char *	USR	OPE	X	Full path name of NQS log file [Max. length: NQS_LEN_FILENAME]	o
Log level	ATTR_LOGLEVEL	nqs_range	USR	OPE	X	Output level of NQS log	o
No. of log files retained	ATTR_LOGFGEN	nqs_range	USR	OPE	X	Number of old NQS log files retained	o
Max. log file length	ATTR_LOGFLEN	int	USR	OPE	X	Max. length of the NQS log file [in bytes, 0 for unlimited length]	o
License information	ATTR_LICINFO	nqs_license	USR	X	o	Use status of each license	o
Heart beat interval	ATTR_HBINTVAL	int	USR	OPE	X	Interval of heart beat transmission between batch server and job server [in seconds, 0 for transmission stop]	o
Load information sampling interval	ATTR_LHINTVAL	int	USR	OPE	X	Interval of sampling load information of the execution host [in seconds, 0 for sampling stop]	o

Job resource size sampling interval	ATTR_JUINTVAL	int	USR	OPE	X	Interval of sampling the size of resource used by the job [in seconds, 0 for sampling stop]	o
Routing queue run limit	ATTR_ROURLIM	nqs_range	USR	OPE	X	Run limit of routing queues in the entire batch server	o
Routing queue retry interval	ATTR_ROURTYIVAL	int	USR	OPE	X	Time allowed before the routing queue tries another transfer (in seconds)	o
Routing queue retry span	ATTR_ROURTYSPAN	nqs_range	USR	OPE	X	Span allowed that routing queue repeats the retrial(in seconds)	o
Network queue run limit	ATTR_NETRLIM	nqs_range	USR	OPE	X	Run limit of network queues in the entire batch server	o
Network queue retry interval	ATTR_NETRTYIVAL	int	USR	OPE	X	Time allowed before the network queue tries another transfer (in seconds)	o
Network queue retry span	ATTR_NETRTYSPAN	nqs_range	USR	OPE	X	Span allowed that the network queue repeats the retrial(in seconds)	o
ACCT server host name	ATTR_ACCTHOST	char *	USR	OPE	X	ACCT server host name.	o
ACCT Server TCP port number	ATTR_ACCTPORT	int	USR	OPE	X	TCP port number which ACCT server communicate with NQSV	o
Job account output directory	ATTR_JACCT_DIR	char *	USR	MGR	X	Output directory for job account file on the BSV. Default is /var/opt/nec/nqsv/acm/jacct	o
Budget check	ATTR_NQS_BUDGETCHECK	int	USR	MGR	X	Setting for enabling budget check by accounting server.	o

						0: Do not check the budget (OFF) (default) 1: Check the budget of request. 2: Check the budget of reservation. 3: Check the budget of request and reservation.	
Request accounting switch	ATTR_RACCTSW	int	USR	MGR	X	Switch for request accounting. 0: OFF (default) 1: ON	o
Request account file	ATTR_RACCTPATH	char *	USR	MGR	X	Request account file name. Default is /var/opt/nec/nqsv/bsv/account/reqacct	o
Reservation accounting switch	ATTR_RESERVATION_ACCT	int	USR	MGR	X	Switch for reservation accounting. 0: OFF (default) 1: ON	o
Reservation account file	ATTR_RESERVATION_ACCT_FILE	char *	USR	MGR	X	Reservation account file name. Default is /var/opt/nec/nqsv/acm/rsvacct/rsvacct	o
Specify Group for Request	ATTR_SPCFYGRP	int	USR	OPE	X	It indicates that Designated Group Execution Function for Request is enable or not. (0:disable 1:enable)	o
Allow Absolute Path	ATTR_ABSLT_EXEPATH	int	USR	MGR	X	It indicates that specifying absolute path for the staging files allowed or not. (0:refuse 1:allow)	o
Items for maximum number of submitted requests							
Submit limit	ATTR_GBLSBLM	nqs_range	USR	OPE	X	Maximum number of requests to submit per a batch server	o
User submit	ATTR_USRSBLM	int	USR	OPE	X	Maximum number of requests to submit per a	o

limit						user of the batch server (0 or positive integer, 0 for unlimited number of request)	
Group submit limit	ATTR_GRPSBLM	int	USR	OPE	X	Maximum number of requests to submit per a group of the batch server (0 or positive integer, 0 for unlimited number of request)	o
Specified user submit limit	ATTR_USRSBLM_N	nqs_int_u	USR	OPE	o	Maximum number of requests to submit of the specified user per the batch server (0 or positive integer, 0 for unlimited number of request)	o
Specified group submit limit	ATTR_GRPSBLM_N	nqs_int_g	USR	OPE	o	Maximum number of requests to submit of the specified group per the batch server (0 or positive integer, 0 for unlimited number of request)	o
Parametric request							
Sub-request number limit	ATTR_GBLSUBREQENT	nqs_range	USR	OPE	X	Maximum number of sub-requests in a parametric request. （1-999999) Default is 100.	o
Template							
OpenStack Template	ATTR_OSTEMPLATE	nqs_ostemplate	USR	OPE	o	Definition of OpenStack Template.	o
Container Template	ATTR_COTEMPLATE	nqs_cotemplate	USR	OPE	o	Definition of Container Template.	o
Number of	ATTR_TEMP_REQS	nqs_temp_r	USR	X	o	Number of requests using template.	o

requests using template		eqs					

4.2. Scheduler attributes

Attribute name	Attribute type	Type	Reference	Alteration	Chain	Description	Scope
							SCH
Scheduler ID	ATTR_SCHID	nqs_schid	USR	X	X	Identifier of the scheduler	o
Version number	ATTR_VERNO	char *	USR	X	X	Scheduler version [Max. length: NQS_LEN_VERSION]	o
Scheduler name	ATTR_SCHNAME	char *	USR	X	X	Name of scheduler [Max. length: NQS_LEN_SCHNAME]	o
Host ID	ATTR_HSTID	nqs_hid	USR	X	X	Identifier of host running the scheduler	o
Scheduler Information	ATTR_SCHDMSG	char *	USR	SCH	X	Messages from the scheduler [Max. length: NQS_LEN_SCHDMSG]	o
Scheduling interval	ATTR_SCHINTERVAL	int	USR	SCH	X	Scheduling interval (JobManipulator)	o
Scheduling status	ATTR_SCHSTAT	int	USR	SCH	X	Scheduling Status SCHST_START: Scheduling is in service. SCHST_STOP: Scheduling is out of service.	o

4.3. Job server attributes

Attribute name	Attribute type	Type	Reference	Alteration	Chain	Description	Scope
							JSV
Job server ID	ATTR_JSVID	nqs_jsvid	USR	X	X	Identifier of job server	o
Version number	ATTR_VERNO	char *	USR	X	X	Job server version [Max. length: NQS_LEN_VERSION]	o
Job server name	ATTR_JSVNAME	char *	USR	X	X	Name of job server [Max. length: NQS_LEN_JSVNAME]	o
Job server status	ATTR_JSVST	nqs_jsvst	USR	X	X	Status of job server	o
Execution host ID	ATTR_HSTID	nqs_hid	USR	X	X	Identifier of host running the job server	o
Queue ID	ATTR_QUEID	nqs_qid	USR	X	o	Identifier of connected queue	o
RSG number	ATTR_RSGNO	nqs_range	USR	X	o	RSG number assigned to a job under control (All RSG numbers related to the bound queues can be get by chain.)	o
Migration parameter	ATTR_MIGPRM	nqs_migprm	USR	OPE	X	parameters for tuning performance of job migration	o
Scheduler Information	ATTR_SCHDMSG	char *	USR	SCH	X	Messages from the scheduler [Max. length: NQS_LEN_SCHDMSG]	o
JM license status	ATTR_JMLICENSE	int	USR	X	X	True: Using a JobManipulator's license False: Not using a JobManipulator's license	o
Node group ID	ATTR_NGRPID	nqs_ngrpid	USR	X	o	Noe group the job server belongs to	o
Execution host type	ATTR_HSTTYPE	int	USR	X	X	Type of execution host. EXECUTION(0) or BAREMETAL(1)	o

Defined GPU Number	ATTR_DEFINED_NGPUS	int	USR	X	X	Defined GPU Number of baremetal server	o
Defined Memory Size	ATTR_DEFINED_MEMORY	nqs_rsgres	USR	X	X	Defined Memory size of baremetal server	o
Defined CPU Number	ATTR_DEFINED_NCPUS	int	USR	X	X	Defined CPU Number of baremetal server	o
OpenStack Template	ATTR_OSTEMPLATE	nqs_ostemplate	USR	X	X	Template which used to boot the baremetal server.	o
RSG information	ATTR_RSGINFO	nqs_rsginfo	USR	X	o	RSG information (memory, swap, number of CPUs, load averages, number of GPUs, and number of VEs) of the JSV host	o
Action of HCA failure detected	ATTR_HCACHK	int	USR	OPE	X	The action for JSV when HCA failure is detected. HCACHK_OFF HCACHK_DOWN HCACHK_UNBIND	o
Items for host load information							
Memory information	ATTR_RBSPMEM	nqs_rsgres	USR	X	X	Information on physical memory size per execution host	o
Swapping information	ATTR_RBSPSWAP	nqs_rsgres	USR	X	X	Information on swapping size per execution host	o
Number-of-CPUs information	ATTR_RBCPUNM	nqs_rsgres	USR	X	X	Information on number of CPUs per execution host	o
Load average information	ATTR_RBLDAVG	nqs_rsgavg	USR	X	X	Information on load average per execution host	o

CPU average information	ATTR_RBCPUAVG	nqs_rsgavg	USR	X	X	Information on CPU average per execution host	o

4.4. Execution host attributes

Attribute name	Attribute type	Type	Reference	Alteration	Chain	Description	Scope
							Execution host
Execution host ID	ATTR_HSTID	nqs_hid	USR	X	X	Identifier of execution host	o
Operating system name	ATTR_SYSNAME	char *	USR	X	X	OS name of execution host [Max. length: NQS_LEN_UTSNAME]	o
OS version number	ATTR_VERNO	char *	USR	X	X	OS version of execution host [Max. length: NQS_LEN_UTSNAME]	o
OS release number	ATTR_RELNO	char *	USR	X	X	OS release name of execution host [Max. length: NQS_LEN_UTSNAME]	o
Hardware name	ATTR_HWNAME	char *	USR	X	X	Hardware name of execution host [Max. length: NQS_LEN_UTSNAME]	o
Job server ID	ATTR_JSVID	nqs_jsvid	USR	X	X	Job server ID of the execution host	o
Node agent	ATTR_NODEAGENT	nqs_hid	USR	X	X	Host in which the Node agent running to watch the execution host.	o
State of the host	ATTR_HST	nqs_hst	USR	X	X	Host state of running. HOSTST_ACTIVE Running HOSTST_INACTIVE Stop	o
Execution host type	ATTR_HSTTYPE	int	USR	X	X	Type of execution host. EXECUTION(0) or BAREMETAL(1)	o
Defined GPU Number	ATTR_DEFINED_NGPUS	int	USR	X	X	Defined GPU Number of baremetal server	o

Defined Memory Size	ATTR_DEFINED_MEMORY	nqs_rsgres	USR	X	X	Defined Memory size of baremetal server	o
Defined CPU Number	ATTR_DEFINED_NCPUS	int	USR	X	X	Defined CPU Number of baremetal server	o
OpenStack Template	ATTR_OSTEMPLATE	nqs_ostemplate	USR	X	X	Template which used to boot the baremetal server.	o
Socket Resource information	ATTR_SOCKETINFO	nqs_socket	USR	X	o	Amount of socket resource and it usage.	o
CPUSET information	ATTR_CPUSETINFO	nqs_cpuset	USR	X	o	Resource information of every CPUSET	o
GPU detailed information	ATTR_GPUINFO	nqs_gpuinfo	USR	X	o	Each GPU detailed information on the execution host.	o
VE node information	ATTR_VEINFO	nqs_veinfo	USR	X	o	Detailed information of VE node on the execution host.	o
RSG information	ATTR_RSGINFO	nqs_rsginfo	USR	X	X	RSG information (Total value of memory, swap, number of CPUs, load averages, number of GPUs, and number of VEs) of the execution host. * Available members of RSG information are different according to the operating system as follows.	o
Items for host load information							
Memory information	ATTR_RBSPMEM	nqs_rsgres	USR	X	X	Information on physical memory size per execution host	o
Swapping	ATTR_RBSPSWAP	nqs_rsgres	USR	X	X	Information on swapping size per execution	o

information						host	
Number-of-CPUs information	ATTR_RBCPUNM	nqs_rsgres	USR	X	X	Information on number of CPUs per execution host	o
Number-of-GPUs information	ATTR_RBGPUNM	nqs_rsgres	USR	X	X	Information on number of GPUs per execution host	o
Load average information	ATTR_RBLDAVG	nqs_rsgavg	USR	X	X	Information on load average per execution host	o
CPU average information	ATTR_RBCPUAVG	nqs_rsgavg	USR	X	X	Information on CPU average per execution host	o
Number of VE node information	ATTR_RBVENM	nqs_rsgres	USR	X	X	Information on number of VE nodes per execution host.	o

4.5. Queue attributes (for batch queue, interactive queue)

Attribute name	Attribute type	Type	Reference	Alteration	Chain	Description	Scope						
							Queue	Request	Execution host	Job server	Job	Process	VE node
Queue ID	ATTR_QUEID	nqs_qid	USR	X	X	Identifier of execution queue	o						
Queue status	ATTR_QUEST	nqs_qst	USR(Q)	OPE	X	Current queue status	o						
Priority	ATTR_PRIORITY	nqs_range	USR(Q)	OPE	X	Queue priority	o						
Scheduler ID	ATTR_SCHID	nqs_schid	USR(Q)	X	X	Identifier of connected scheduler	o						
Scheduler name	ATTR_SCHNAME	char *	USR(Q)	X	X	Name of connected scheduler [Max. length: NQS_LEN_SCHNAME]	o						
Checkpoint attribute	ATTR_CHKPNTABL	nqs_range	USR(Q)	OPE	X	Periodic checkpoint interval of 0 or positive integer (unit: minute). No checkpoint where 0. (Batch queue only)		o					

Hold privilege	ATTR_HOLDPRIV	int	USR(Q)	OPE	X	Authority privilege of client necessary for holding request in the RUNNING state. (batch queue only) PRIV_SCH: Scheduler authority and above PRIV_MGR: Manager authority and above PRIV_OPE: Operator authority and above PRIV_GMGR: Group Manager authority and above PRIV_SPU: Special user authority and above PRIV_USR: General user authority and above PRIV_NON: Any authority is acceptable.		o					

Suspend privilege	ATTR_SUSPRIV	int	USR(Q)	OPE	X	Authority privilege of client necessary for suspending request in the RUNNING state. (For privilege type, please refer to "Hold privilege".)		o					
User Exit information	ATTR_USEREXIT	nqs_uexit	USR(Q)	OPE	o	Information of User EXIT which started by each execution phase of batch job.					o		
Refusing submission by route	ATTR_RFUSSB	int	USR(Q)	OPE	o	Refuse to submit a batch request by route. RFUSSB_QSUB: Refuse to submit via qsub (NQScrereq) RFUSSB_QMOV: Refuse to submit via qmove (NQSmovreq). (exclude interactive queue) RFUSSB_LCRQ: Refuse to routing via local routing queue RFUSSB_RMRQ: Refuse to routing via remote routing queue	o						
Priority Range(MGR)	ATTR_MGRPRRNG	nqs_hilo	USR(Q)	MGR	X	A range which NQSV Manager can set for the request priority. (exclude interactive queue)		o					

Priority Range(OPE)	ATTR_OPEPRRNG	nqs_hilo	USR(Q)	MGR	X	A range which NQSV Operator can set for the request priority. (exclude interactive queue)		o					
Priority Range(SPU)	ATTR_SPUPRRNG	nqs_hilo	USR(Q)	MGR	X	A range which Special User can set for the request priority. (exclude interactive queue)		o					
Priority Range(USR)	ATTR_USRPRRNG	nqs_hilo	USR(Q)	MGR	X	A range which a general user can set for the request priority. (exclude interactive queue)		o					
ACL mode	ATTR_ACLMODE	int	USR(Q)	MGR	X	ACL_ACCESS: Users are allowed to access the queue only when the user name is in the ACL user name list or the group name is in the ACL group name list.	o						
ACL user name list	ATTR_ACLUNAME	char *	USR(Q)	MGR	o								

ACL group name list	ATTR_ACLGNAME	char *	USR(Q)	MGR	o	ACL_NOACCESS: Users are not allowed to access the queue when the user name is in the ACL user name list or the group name is in the ACL group name list. The maximum length of the user name is NQS_LEN_USERNAME. The maximum length of the group name is NQS_LEN_GROUPNAME.							
Submit Limit with Supplementary group name	ATTR_SUPGIDCHK	int	USR(Q)	OPE	X	Set valid/invalid of submit limit of request using supplementary group name. If true, the supplementary group name at request creation time is added to the target for submit limit check with ACL group name list.	o						
Number of requests counted per status	ATTR_NREQST	nqs_nreqst	USR(Q)	X	X	Number of requests counted per status	o						

JSV Auto bind control	ATTR_AUTOBIND	int	USR(Q)	OPE	X	Controls the auto bind at JSV link up True: JSV is bound automatically. (default) False: JSV is not bound automatically.				o			
Restart file's directory	ATTR_RSTFDIR	char *	USR(Q)	OPE	X	Full path in which the restart file is stored on execution host [Max. length: NQS_LEN_RSTFPATH] (exclude interactive queue)					o		
Scheduler Information	ATTR_SCHDMSG	char *	USR(Q)	SCH	X	Messages from the scheduler [Max. length: NQS_LEN_SCHDMSG]	o						
Job Number Range	ATTR_NJOBRNG	nqs_hilo	USR(Q)	OPE	X	A range of batch jobs which can be created for each request		o					
Number of JSV	ATTR_NJSVS	int	USR(Q)	X	X	Total number of the job servers which bind to the queue	o						
Sub-request number limit	ATTR_SUBREQLM	int	USR(Q)	OPE	X	Maximum sub-request number of the parametric request submitted to the queue. (exclude interactive queue)	o						
Attach function enable	ATTR_QATTACH	int	USR(Q)	MGR	X	The request is attachable or not 1: enable 0: disable	o						

IntelMPI process manager	ATTR_INTMPI_PMGR	int	USR(Q)	OPE	X	Process manager setting of IntelMPI request. NQSII_IMPI_HYDRA: hydra (default) NQSII_IMPI_MPD : mpd	o						
Hook script function	ATTR_HOOKFUNC	int	USR(Q)	OPE	X	Controls the hook script function. 0: disabled (default) 1: enabled	o						
Time-out of UserEXIT script	ATTR_UEXIT_TIMEOUT	int	USR(Q)	OPE	X	Time-out time of UserEXIT script (sec). Time-out is disabled when 0 specified. (default: 0)	o						
Time-out of UserPP script	ATTR_UPP_TIMEOUT	int	USR(Q)	OPE	X	Time-out time of UserPP script (sec). Time-out is disabled when 0 specified. (default: 300 sec)	o						
Custom resource limit	ATTR_QUECRINFO	nqs_quecrinfo	USR(Q)	OPE	o	Custom resource information about the queue.	o						
Total VE node number	ATTR_TOTALVENUM	nqs_rrange	USR(Q)	OPE	X	Number of total VE node range that can submit		o					
Range for number of VE node	ATTR_VENUM	nqs_rrange	USR(Q)	OPE	X	Number of VE node range that can run simultaneously					o		

Defined VE node number	ATTR_VENUMDEFINE	int	USR(Q)	OPE	X	Defined VE node number that used to submitting with total VE node number.	o						
Allowance for exclusive execution request	ATTR_EXCLUSIVE	int	USR(Q)	MGR	X	Allowance for exclusive execution request		o					
Stage-out enable	ATTR_DOSTGOUT	int	USR	OPE	X	The request do the stage-out or not 0: skip the stage-out 1: do the stage-out	o						
Delete the urgent request that failed to execute	ATTR_DELURGNTREQ	int	USR(Q)	OPE	X	Whether to delete the urgent request that failed to execute 0: not delete (default) 1: delete	o						
Use partial process swapping	ATTR_USEPPS	int	USR(Q)	OPE	X	Whether to use the partial process swapping 0: not use 1: use (default)	o						
Interactive queue information (only for interactive queue)													

Waiting option	ATTR_REALTIME_SCHEDULING	int	USR(Q)	OPE	X	Actions when execution host is not assigned immediately for an interactive request: MODE_CANCEL: Submission is canceled. MODE_WAIT: Request waits scheduling. MODE_MANUAL: Cancel or wait by qlogin's option.	o						
Forced shell	ATTR_RESTRICT_SHELL	char *	USR(Q)	OPE	X	The shell program specified in this attribute is used to execute interactive session. If this attribute is not set, submitted user's login shell or the shell specified by qlogin's -S option is used.	o						
Idle timer	ATTR_IDLETIMER	int	USR(Q)	OPE	X	The default idle timer (unit: minute) of the request submitted to the queue. (0 means idle timer is OFF.)	o						
Items for resource limits													
Max. elapsed time	ATTR_ELPSTIM	nqs_rlim	USR(Q)	OPE	X	Maximum time elapse allowed since the execution started		o					
Max. CPU time	ATTR_CPUTIM	nqs_rlim	USR(Q)	OPE	X	Maximum CPU activity time					o	o	

Max. number of CPU	ATTR_CPUNUM	nqs_rnum	USR(Q)	OPE	X	Maximum number of CPUs that can run simultaneously					o		
Max. number of files opened	ATTR_FILENUM	nqs_rnum	USR(Q)	OPE	X	Maximum number of files opened simultaneously						o	
Max. memory size	ATTR_MEMSZ	nqs_rlim	USR(Q)	OPE	X	Maximum memory size available					o	o	
Max. data size	ATTR_DATASZ	nqs_rlim	USR(Q)	OPE	X	Maximum data segment size available						o	
Max. stack size	ATTR_STACKSZ	nqs_rlim	USR(Q)	OPE	X	Maximum stack size available						o	
Max. core file size	ATTR_CORESZ	nqs_rlim	USR(Q)	OPE	X	Maximum core file size to be created						o	
Max. file size	ATTR_FILESZ	nqs_rlim	USR(Q)	OPE	X	Maximum file size to be created						o	
Submit limit	ATTR_GBLQSBLM	nqs_range	USR(Q)	OPE	X	Maximum number of requests to submit per a queue (0 or positive integer, 0 for unlimited number of request)	o						
User submit limit	ATTR_USRQSBLM	int	USR(Q)	OPE	X	Maximum number of requests to submit per a user of the queue (0 or positive integer, 0 for unlimited number of request)	o						

Group submit limit	ATTR_GRPQSBLM	int	USR(Q)	OPE	X	Maximum number of requests to submit per a group of the queue (0 or positive integer, 0 for unlimited number of request)	o						
Max. virtual memory size	ATTR_VMEMSZ	nqs_rlim	USR(Q)	OPE	X	Maximum virtual memory size available					o	o	
Max. number of GPU	ATTR_GPUNUM	nqs_rnum	USR(Q)	OPE	X	Maximum number of GPUs that can run simultaneously					o		
Number of total VE node	ATTR_TOTALVENUM	nqs_rrange	USR(Q)	OPE	X	Total VE node number		o					
Limit for number of VEnode	ATTR_VENUM	nqs_rrange	USR(Q)	OPE	X	Range of VE node number to submit that can run simultaneously.					o		
Defined VE node number	ATTR_VENUMDEFINE	int	USR(Q)	OPE	X	Defined VE node number that is used to submit with total VE node number.	o						
Limit for CPU time range.	ATTR_CPUTIMRNG	nqs_rrange	USR(Q)	OPE	X	Range of CPU Time limit.					o		
Limit for CPU number range.	ATTR_CPUNUMRNG	nqs_rrange	USR(Q)	OPE	X	Range of CPU number limit.					o		
Limit for memory size range.	ATTR_MEMSZRNG	nqs_rrange	USR(Q)	OPE	X	Range of Memory size limit.					o		

Limit for virtual memory size range.	ATTR_VMEMSZRNG	nqs_rrange	USR(Q)	OPE	X	Range of Virtual Memory size limit.					o		
Limit for GPU number range.	ATTR_GPUNUMRNG	nqs_rrange	USR(Q)	OPE	X	Range of GPU number limit.					o		
Limit for VE CPU time range.	ATTR_VECPUTIMRNG	nqs_rrange	USR(Q)	OPE	X	Range of VE CPU time limit.					o	o	o
Limit for VE memory size range.	ATTR_VEMEMSZRNG	nqs_range	USR(Q)	OPE	X	Range of VE memory size limit.					o	o	o
HCA port number	ATTR_HCA	nqs_hca	USR(Q)	OPE	X	Number of HCA port number range that can submit					o		
Items for limitations of the specified user													
Job Number Range of specified user	ATTR_NJOBRNG_U	nqs_hilo_u	USR	OPE	o	A range of batch jobs of the specified user which can be created for each request		o					
Max. elapsed time of specified user	ATTR_ELPSTIM_U	nqs_rlim_u	USR	OPE	o	Maximum time elapse of the specified user allowed since the execution started		o					
Submit limit of specified user	ATTR_USRQSBLM_N	nqs_int_u	USR	OPE	o	Maximum number of requests to submit of the specified user of the queue	o						
Items for limitations of the specified group													

Job Number Range of specified group	ATTR_NJOBRNG_G	nqs_hilo_g	USR	OPE	o	A range of batch jobs of the specified group which can be created for each request		o					
Max. elapsed time of specified group	ATTR_ELPSTIM_G	nqs_rlim_g	USR	OPE	o	Maximum time elapse of the specified group allowed since the execution started		o					
Submit limit of specified group	ATTR_GRPQSBLM_N	nqs_int_g	USR	OPE	o	Maximum number of requests to submit of the specified group of the queue	o						
Items for Kernel parameters													
RSG number	ATTR_RSGNO	nqs_range	USR(Q)	OPE	X	Resource sharing group number						o	
Nice value	ATTR_NICE	nqs_range	USR(Q)	OPE	X	Nice value						o	

4.6. Routing queue attributes

Attribute name	Attribute type	Type	Reference	Alteration	Chain	Description	Scope
							Queue
Queue ID	ATTR_QUEID	nqs_qid	USR	X	X	Queue identifier of routing queue	o
Queue status	ATTR_QUEST	nqs_qst	USR(Q)	OPE	X	Current queue status	o
Priority	ATTR_PRIORITY	nqs_range	USR(Q)	OPE	X	Queue priority	o
Run limit	ATTR_ROURLIM	nqs_range	USR(Q)	OPE	X	Maximum number of simultaneously executed items per routing queue	o
Destination queue	ATTR_DESTQUE	nqs_qdesc	USR(Q)	OPE	o	Information on the destination queue	o
Refusing submission by route	ATTR_RFUSSB	int	USR(Q)	OPE	o	Refuse to submit a batch request by route. RFUSSB_QSUB: Refuse to submit via qsub(NQScrereq) RFUSSB_QMOV: Refuse to submit via qmove(NQSmovreq) RFUSSB_LCRQ: Refuse to routing via local routing queue RFUSSB_RMRQ: Refuse to routing via remote routing queue	o
ACL mode	ATTR_ACLMODE	int	USR(Q)	MGR	X	ACL_ACCESS: Users are allowed to access the queue only when the user name is in the ACL user name list or the group name is in the ACL group name list.	o
ACL user name list	ATTR_ACLUNAME	char *	USR(Q)	MGR	o		o

ACL group name list	ATTR_ACLGNAME	char *	USR(Q)	MGR	o	ACL_NOACCESS: Users are not allowed to access the queue when the user name is in the ACL user name list or the group name is in the ACL group name list. The maximum length of the user name is NQS_LEN_USERNAME. The maximum length of the group name is NQS_LEN_GROUPNAME.	o
Submit Limit with Supplementary group name	ATTR_SUPGIDCHK	int	USR(Q)	OPE	X	Set valid/invalid of submit limit of request using supplementary group name. If true, the supplementary group name at request creation time is added to the target for submit limit check with ACL group name list.	o
Number of requests of each status	ATTR_NREQST	nqs_nreqst	USR(Q)	X	X	Number of requests counted of each status	o
Hook script function	ATTR_HOOKFUNC	int	USR(Q)	OPE	X	Controls the hook script function. 0: disabled (default) 1: enabled	o
Submit limit	ATTR_GBLQSBLM	nqs_range	USR(Q)	OPE	X	Maximum number of requests to submit per a queue (0 or positive integer, 0 for unlimited number of request)	o

User submit limit	ATTR_USRQSBLM	int	USR(Q)	OPE	X	Maximum number of requests to submit per a user of the queue (0 or positive integer, 0 for unlimited number of request)	o
Group submit limit	ATTR_GRPQSBLM	int	USR(Q)	OPE	X	Maximum number of requests to submit per a group of the queue (0 or positive integer, 0 for unlimited number of request)	o
Items for limitations of the specified user/group							
Specified user submit limit	ATTR_USRQSBLM_N	nqs_int_u	USR	OPE	o	Maximum number of requests to submit of the specified user of the queue	o
Specified group submit limit	ATTR_GRPQSBLM_N	nqs_int_g	USR	OPE	o	Maximum number of requests to submit of the specified group of the queue	o

4.7. Network queue attributes

Attribute name	Attribute type	Type	Reference	Alteration	Chain	Description	Scope
							Queue
Queue ID	ATTR_QUEID	nqs_qid	USR	X	X	Identifier of network queue	o
Queue status	ATTR_QUEST	nqs_qst	USR(Q)	OPE	X	Current queue status	o
Priority	ATTR_PRIORITY	nqs_range	USR(Q)	OPE	X	Queue priority	o
Run limit	ATTR_NETRLIM	nqs_range	USR(Q)	OPE	X	Maximum number of simultaneously executed items per network queue	o
Run limit per batch request	ATTR_BREQRLIM	nqs_range	USR(Q)	OPE	X	Maximum number of simultaneously executed items per batch request	o
Client host name	ATTR_HOSTNAME	char *	USR(Q)	OPE	X	Name of a client host to be staged [Max. length: NQS_LEN_HOSTNAME]	o
Network request	ATTR_NETREQ	nqs_netreq	USR(R)	X	o	Information of network request	o
Staging Method	ATTR_STGMETHOD	int	USR(Q)	OPE	X	Information of network request STGMTD_INTERNAL: Internal Staging Method of NQSV default function STGMTD_EXTERNAL: External Staging Method	o

Extended buffer size	ATTR_EXSTGBSZ	nqs_range	USR(Q)	OPE	X	Extended buffer size for file staging function. This unit is "byte" and its range is from 0 up to 512x1024. If you specify 0 as a buffer size, NQSV uses the standard buffer for the file staging. The size of the standard buffer is 4000 bytes.	o

4.8. Request attributes

Attribute name	Attribute type	Type	Reference	Alteration	Chain	Description	Scope			
							Request	Job	Process	VE node
Request ID	ATTR_REQID	nqs_rid	USR	X	X	Identifier of request	o			
Request status	ATTR_REQST	nqs_rst	USR	X	X	Current request status	o			
Request type	ATTR_REQTYP	int	USR	X	X	Bit map of request type: REQTYP_FLAG_QLOGIN (0x000001): Interactive request submitted by qlogin REQTYP_FLAG_QRSH (0x000002): Interactive request submitted by qrsh	o			
Queue ID	ATTR_QUEID	nqs_qid	USR(R)	X	X	Identifier of queue where the request is submitted	o			
Request owner	ATTR_REQOWN	nqs_udesc	USR(R)	X	X	Information on request owner (The gid of nqs_udesc is same group of request)	o			
Group information	ATTR_REQOWNGRP	nqs_gdesc	USR(R)	X	X	Information on request group (Request group is the primary group of request owner. Or, if Designated Group Execution Function is enabled, Request group is the specified group when submitting the request.)	o			

Job topology	ATTR_JTPLGY	int	USR(R)	USR(R)	X	Job execution topology JTPLGY_DISTRIB: Distributed job JTPLGY_NECMPI: necmpi job JTPLGY_OPENMPI: openmpi job JTPLGY_INTMPI: intmpi job JTPLGY_MVAPICH: mvapich job JTPLGY_PLTMPI: pltmpi job	o			
Re-run attribute	ATTR_RERUNABL	int	USR(R)	USR(R)	X	"Re-runnable" flag ("true" for re-runnable) (batch request only)	o			
Restart attribute	ATTR_RESTARTABL	int	USR(R)	X	X	"Restartable" flag ("true" for restart) (batch request only)	o			
Checkpoint attribute	ATTR_CHKPNTABL	nqs_range	USR(R)	USR(R)	X	Periodic checkpoint interval of -1, 0 or positive integer (unit: minute). (batch request only) No checkpoint where 0. The ATTR_CHKPNTABL of execution queue is used as interval where -1.	o			
Migration attribute	ATTR_MIGRATABL	int	USR(R)	USR(R)	X	"job migratable" flag ("true" for migratable) (batch request only)	o			
Hold attribute	ATTR_HOLDABL	int	USR(R)	USR(R)	X	"Holdable" flag ("true" for holdable) (batch request only)	o			

Hold type	ATTR_HOLDTYPE	int	USR(R)	X	X	Client authority for holding (batch request only) OPBY_SCHEDULER：Held by the scheduler OPBY_MANAGER ：Held by the manager OPBY_OPERATOR ：Held by the operator OPBY_GMANAGER：Held by the group manager OPBY_SPUSER ：Held by the special user OPBY_USER ：Held by the request owner OPBY_NONE ：Not held	o			
Suspend type	ATTR_SUSPTYPE	int	USR(R)	X	X	Client authority for suspending OPBY_SCHEDULER：Suspended by the scheduler OPBY_MANAGER ：Suspended by the manager OPBY_OPERATOR ：Suspended by the operator OPBY_GMANAGER ：Suspended by the group manager OPBY_SPUSER ：Suspended by the special user OPBY_USER ：Suspended by the request owner OPBY_NONE ：Not Suspended	o			
Rerun count	ATTR_RERUNCNT	int	USR(R)	X	X	Rerun count of request	o			

Account code	ATTR_ACCTCODE	char *	USR(R)	USR(R)	X	Account code [Max. length: NQS_LEN_ACCTCODE]	o			
Priority	ATTR_PRIORITY	nqs_range	USR(R)	USR(R)	X	Request priority	o			
Request name	ATTR_REQNAME	char *	USR(R)	USR(R)	X	Request name [Max. length: NQS_LEN_REQNAME]	o			
Standard output path name	ATTR_STDOUT	nqs_pdesc	USR(R)	USR(R)	X	Destination of standard output file	o			
Standard error output path name	ATTR_STDERR	nqs_pdesc	USR(R)	USR(R)	X	Destination of standard error output file	o			
Request log output path name	ATTR_STDLOG	nqs_pdesc	USR(R)	USR(R)	X	Destination of request log output file	o			
Request log output level	ATTR_LOGLEVEL	nqs_range	USR(R)	USR(R)	X	Request log output level	o			
Staging information	ATTR_STGFILE	nqs_stgfile	USR(R)	X	o	Information on files to be staged	o			
Shell name	ATTR_SHELLPATH	char *	USR(R)	USR(R)	X	Full path name of job execution shell [Max. length: NQS_LEN_PATHNAME]	o			

Mail option	ATTR_MAILOPTS	int	USR(R)	USR(R)	X	Mail posting option (Two or more options can be specified at a time.) MAIL_SENDBGN: Mailed when the request execution starts. MAIL_SENDEND: Mailed when the request execution ends.	o			
Mail address	ATTR_MAILADDR	nqs_mdesc	USR(R)	USR(R)	X	Mail address	o			
Job execution environment condition	ATTR_JOBCOND	nqs_jcond	USR(R)	X	o	Conditions on execution host for each job	o			
Number of JOBCOND	ATTR_NJCONS	int	USR(R)	X	X	Total entry counts of conditions on execution jobs	o			
Request group	ATTR_REQGRP	nqs_rgrp	USR(R)	X	X	Request group for request connection	o			
Creation time	ATTR_CRETIME	time_t	USR(R)	X	X	Time when request was created (seconds elapsed since EPOCH)	o			
Submission time	ATTR_ENTTIME	time_t	USR(R)	X	X	Time when request is submitted to current queue (seconds elapsed since EPOCH)	o			
Execution time	ATTR_EXETIME	time_t	USR(R)	USR(R)	X	Time when request exited WAITING status (seconds elapsed since EPOCH)	o			
Reserved time	ATTR_RSVTIME	time_t	USR(R)	USR(R)	X	Time reserved to start execution of request (seconds elapsed since EPOCH) (exclude interactive request)	o			
Start time	ATTR_BGNTIME	time_t	USR(R)	X	X	Time when execution of request started (seconds elapsed since EPOCH)	o			

Exit time	ATTR_ENDTIME	time_t	USR(R)	X	X	Time at which execution of request ended (seconds elapsed since EPOCH)	o			
UMASK value	ATTR_UMASK	int	USR(R)	X	X	UMASK value when request is submitted			o	
Job environment variable	ATTR_ENVIRON	nqs_keyval	X	X	o	Where there is an environment variable name and variable value			o	
Migration file	ATTR_MIGFILE	nqs_migfile	USR(R)	X	o	The migration files are user files, or checkpoints files which are copied to destination host as the job is moved while job migration (exclude interactive request)	o			
User Custom Attribute	ATTR_USERATTR	nqs_keyval	USR(R)	X	o	The attribute that user can define	o			
Restart file path	ATTR_RSTFDIR	char *	USR(R)	USR(R)	X	The relative path on the execution host to which the restart file is stored [Max. length: NQS_LEN_PATHNAME]	o			
Reservation ID	ATTR_ADVRSVID	int	USR(R)	X	X	Reservation ID for advance reservation ID is an integer value of 0 or greater. A negative number means unset. (exclude interactive request)	o			
Assigned time	ATTR_ASSTIME	time_t	USR(R)	SCH	X	The time that the request is planned to start execution	o			

Attach function enable	ATTR_QATTACH	int	USR(R)	MGR	X	The request is attachable or not 1: enable 0: disable	o			
Request is under attach or not	ATTR_QATTACH_EXEC	int	USR(R)	X	X	Request is under attach or not. 0: Not attached 1: Attached	o			
Job flag	ATTR_JOBEXIST	int	USR(R)	X	X	The flag which indicates the request has jobs or not. 0: No jobs. 1: Any jobs are created.	o			
Preceding requests	ATTR_PRECEDING	nqs_rid	USR(R)	X	o	The request IDs of preceding requests.	o			
Parallel execution requests	ATTR_PARALLEL	nqs_rid	USR(R)	X	o	The request IDs of parallel execution requests.	o			
Following requests cancel flag	ATTR_CANCELAFTER	int	USR(R)	X	X	The flag which indicates following requests are canceled or not when the request terminates abnormally. 1: Canceled. 0: Not canceled.	o			
Workflow ID	ATTR_WFID	nqs_wid	USR(R)	X	X	The workflow ID to which the request belongs.	o			
UserPP script	ATTR_USERPP	nqs_upp	USR(R)	USR(R)	o	UserPP script information.	o			
OpenStack Template	ATTR_OSTEMPLATE	nqs_ostemplate	USR(R)	X	X	Specified OpenStack template.	o			

Container Template	ATTR_COTEMPLATE	nqs_cotemplate	USR(R)	X	X	Specified Container template.	o			
Custom resource information	ATTR_REQCRINFO	nqs_reqcrinfo	USR(R)	USR(R)	o	Specified custom resource information.	o			
Total VE node number	ATTR_TOTALVENUM	nqs_rrange	USR(R)	X	X	Number of total VE node range that can submit	o			
Exclusive execution	ATTR_EXCLUSIVE	int	USR(R)	X	X	Type of exclusive execution 0: Not exclusive execution 1: Host unit exclusive execution	o			
Actual usage value of the custom resource	ATTR_CRUSG	nqs_crusg	USR(R)	X	O	Actual usage value of the custom resources	o			
Whether to capture SIGTERM	ATTR_ACCEPTSIGTERM	int	USR(R)	USR(R)	X	Whether to capture SIGTERM to capture SIGTERM 0: Ignore SIGTERM (Default) 1: Can capture SIGTERM	o			
Parametric request information (for parametric request only)										
Number of sub-requests	ATTR_NSUBREQS	nqs_nsubreq	USR(R)	X	X	Number of sub-requests to execute in the parametric request.	o			
Specified string for sub-request number	ATTR_SUBREQSTR	char *	USR(R)	X	X	The string which indicates sub-request numbers specified by qsub -t.	o			
Interactive request information (for interactive request only)										

Submitted host	ATTR_SUBMIT_HOST	nqs_hid	USR(R)	X	X	Submitted host name of the interactive request.	o			
Session port	ATTR_PORT	int	USR(R)	X	X	The port number to connect the interactive session.	o			
Wait option	ATTR_SCH_WAIT	int	USR(R)	X	X	Action of submitted interactive request when no execution host is assigned immediately. MODE_CANCEL: Cancel the submission. MODE_WAIT: Wait for scheduling.	o			
Forced shell	ATTR_RESTRICT_SHELL	char *	USR(R)	OPE	X	The shell program which is mandatory used to execute interactive session.	o			
Idle timer	ATTR_IDLETIMER	int	USR(R)	USR	X	Idle timer (unit: minute). 0 means the idle timer is OFF.	o			
Items for resource limit values										
Max. elapsed time	ATTR_ELPSTIM	nqs_rlim	USR(R)	USR(R)	X	Maximum elapsed time allowed since the start of execution	o			
Max. CPU time	ATTR_CPUTIM	nqs_rlim	USR(R)	USR(R)	X	Maximum CPU activity time		o	o	
Max. number of CPU	ATTR_CPUNUM	nqs_rnum	USR(R)	USR(R)	X	Maximum number of CPUs running simultaneously		o		
Max. number of files opened	ATTR_FILENUM	nqs_rnum	USR(R)	USR(R)	X	Maximum number of files opened simultaneously			o	
Max. memory size	ATTR_MEMSZ	nqs_rlim	USR(R)	USR(R)	X	Available maximum memory size limit		o	o	
Max. data size	ATTR_DATASZ	nqs_rlim	USR(R)	USR(R)	X	Maximum data segment size available			o	
Max. stack size	ATTR_STACKSZ	nqs_rlim	USR(R)	USR(R)	X	Maximum stack size available			o	

Max. core file size	ATTR_CORESZ	nqs_rlim	USR(R)	USR(R)	X	Maximum core file size that can be created			o	
Max. file size	ATTR_FILESZ	nqs_rlim	USR(R)	USR(R)	X	Maximum file size that can be created			o	
Max. virtual memory size	ATTR_VMEMSZ	nqs_rlim	USR(R)	USR(R)	X	Available maximum virtual memory size limit		o	o	
Max. number of GPU	ATTR_GPUNUM	nqs_rnum	USR(R)	USR(R)	X	Maximum number of GPUs running simultaneously		o		
Range for number of VE node	ATTR_VENUM	nqs_rrange	USR(R)	USR(R)	X	Range for number of VE nodes running simultaneously		o		
Max. CPU time of VE	ATTR_VECPUTIMRNG	nqs_rrange	USR(R)	USR(R)	X	Maximum CPU activity time of VE		o	o	o
Max. memory size of VE	ATTR_VEMEMSZRNG	nqs_rrange	USR(R)	USR(R)	X	Available maximum memory size limit of VE		o	o	o
HCA port number	ATTR_HCA	nqs_hca	USR(R)	USR(R)	X	Range for number of HCA port number		o		
Group of jobs	ATTR_JOBGROUP	nqs_jobgroup	USR(R)	USR(R)	O	Different attribute values for each group of jobs in the hybrid jobs		o		
Items for Kernel parameters										
RSG number	ATTR_RSGNO	nqs_range	USR(R)	X	X	Resource sharing group number			o	
Nice value	ATTR_NICE	nqs_range	USR(R)	OPE	X	Nice value			o	
Items for resource used values										
Elapsed time	ATTR_USEELPSTIM	int	USR(R)	X	X	Elapsed time after request starts execution [unit: second]	o			

CPU time usage	ATTR_USECPUTIM	long long	USR(R)	X	X	CPU time in use (Total CPU time used by the system and CPU time used by the user) The terminated processes are not included. [unit: microseconds]	o		
Amount of CPU time used	ATTR_ACCCPUTIM	long long	USR(R)	X	X	Amount of CPU time from start of execution (Total CPU time used by the system and CPU time used by the user) The terminated processes are included. [unit: microseconds]	o		
Memory usage	ATTR_USEMEMSZ	long long	USR(R)	X	X	Size of memory in use (total of memories for text, data and stack, shared memory, etc.) [unit: byte]	o		
Virtual memory usage	ATTR_USEVMEMSZ	long long	USR(R)	X	X	Size of virtual memory in use [unit: byte]	o		

4.9. Job attributes

Attribute name	Attribute type	Type	Reference	Alteration	Chain	Description	Scope		
							Job	Process	VE node
Job ID	ATTR_JOBID	nqs_jid	USR	X	X	Identifier of batch job	o		
Execution job ID	ATTR_EXEJID	int	USR(R)	X	X	Job identifier assigned by Kernel SID (session ID). Negative integer while job is not in progress	o		
Job owner	ATTR_JOBOWN	nqs_udesc	USR(R)	X	X	Information on job owner	o		
Job server ID	ATTR_JSVID	nqs_jsvid	USR(R)	X	X	Job server controlling the job	o		
Execution host ID	ATTR_HSTID	nqs_hid	USR(R)	X	X	Execution host having the job	o		
VM host ID	ATTR_VMHOSTNAME	char *	USR(R)	X	X	VM host or Container hostname having the job * If the job does not execute on VM or Container, it returns NULL.	o		
Assigned socket	ATTR_SOCKETS	char *	USR(R)	X	X	Assigned socket number. *"(none)" will be returned if not assigned.	o		
Job exit code	ATTR_EXITCODE	int	USR(R)	X	X	Exit code of job (Highest process in the job) Negative integer if execution is not completed	o		
Restart file	ATTR_RSTFINFO	nqs_rstf	USR(R)	X	X	Information of restart file.	o		
Account code	ATTR_ACCTCODE	char *	USR(R)	X	X	This attribute is not given to each job. The attribute value of parent request is	o		

						referred to.			
Items for resource limits									
Max. CPU time	Same as those of batch request attributes		USR(R)	X	X	These attributes are not given to each job. The attribute values of parent request are referred to.	o	o	
Max. number of CPU			USR(R)	X	X		o		
Max. number of files opened			USR(R)	X	X			o	
Max. memory size			USR(R)	X	X		o	o	
Max. data size			USR(R)	X	X			o	
Max. stack size			USR(R)	X	X			o	
Max. core file size			USR(R)	X	X			o	
Max. file size			USR(R)	X	X			o	
Max. virtual memory size			USR(R)	X	X		o	o	
Max. number of GPU			USR(R)	X	X		o		
Max. CPU time of VE			USR(R)	X	X		o	o	o
Max. memory size of VE			USR(R)	X	X		o	o	o
Items for Kernel parameters									
RSG number	Same as those of batch request attributes		USR(R)	X	X	These attributes are not given to each job. The attribute values of parent request are referred to.		o	
Nice value			USR(R)	X	X			o	

Items for resource usage									
CPU time usage	ATTR_USECPUTIM	long long	USR(R)	X	X	CPU time in use (Total CPU time used by the system and CPU time used by the user). The terminated processes are not included. [unit: microseconds]	o		
Amount of CPU time used	ATTR_ACCCPUTIM	long long	USR(R)	X	X	Amount of CPU time from start of execution.(Total CPU time used by the system and CPU time used by the user) The terminated processes are included. [unit: microseconds]	o		
Memory usage	ATTR_USEMEMSZ	long long	USR(R)	X	X	Size of memory in use (total of memories for text, data and stack, shared memory, etc.) [unit: bytes]	o		
Virtual memory usage	ATTR_USEVMEMSZ	long long	USR(R)	X	X	Size of virtual memory in use. [unit: bytes]	o		
Actual usage value of the custom resource	ATTR_CRUSG	nqs_crusg	USR(R)	X	O	Actual usage value of the custom resources	o		

4.10. Node group attributes

Attribute name	Attribute type	Type	Reference	Alteration	Chain	Description	Scope
							Node group
Node group ID	ATTR_NGRPID	nqs_ngrpid	USR	X	X	Identifier of the node group.	o
Comment	ATTR_COMMENT	char *	USR	OPE	X	Comment of the node group.	o
Bind flag	ATTR_BINDABLE	int (boolean)	USR	OPE	X	The flag which indicates the node group can be bound to a queue or not.	o
Job server ID	ATTR_JSVID	nqs_jsvid	USR	X	o	List of the job servers that belong to the node group.	o
Queue ID	ATTR_QUEID	nqs_qid	USR	X	o	List of the queues to which the node group is bound.	o

4.11. Workflow attributes

Attribute name	Attribute type	Type	Reference	Alteration	Chain	Description	Scope
							Workflow
Workflow ID	ATTR_WFID	nqs_wid	USR	X	X	Identifier of the workflow	o
Request ID	ATTR_REQID	nqs_rid	USR	X	o	Request ID of the requests which is included in the workflow	o
Owner of workflow	ATTR_WFLOWN	nqs_udesc	USR	X	X	Owner information of the workflow	o

4.12. Custom resource attributes

Attribute name	Attribute type	Type	Reference	Alteration	Chain	Description	Scope
							Batch server
Custom resource ID	ATTR_CRID	nqs_crid	USR	X	X	Custom resource ID	o
Consumer	ATTR_CONSUMER	int	USR	MGR	X	Consumer of custom resource. ・ CR_JOB: job ・ CR_REQ: request	o
Resource amount control information	ATTR_CRRESOURCE	nqs_crresource	USR	MGR	o	Amount control information of custom resource.	o
Check Mode	ATTR_CR_CHKMD	int	USR	MGR	X	Resource monitoring mode. ・ CR_CM_OFF: Not check the actual value ・ CR_CM_MOMENT: Check as momental value ・ CR_CM_INTEGRATE: Check as integrate value	o
Job Termination	ATTR_CR_TRMJB	int	USR	MGR	X	Terminate the job if it exceed the limit. 0: Disabled 1: Enabled	o
Unit	ATTR_CR_UNIT	char *	USR	MGR	X	The unit of the resource. [Maximum: NQS_LEN_CRUNIT]	o

5. Definition of Data Types and Constants

5.1. Symbol constants

NQSV defines the constants listed below with nqsv.h.

Constant name	Value	Description
NQS_MAX_JSVNO	10239	Maximum job server number
NQS_MAX_JOBNO	10239	Maximum job number
NQS_MAX_SCHNO	15	Maximum batch scheduler number
NQS_LEN_HOSTNAME	255	Maximum host name length
NQS_LEN_FILENAME	255	Maximum file name length
NQS_LEN_PATHNAME	1023	Maximum path name length
NQS_LEN_UTSNAME	63	Maximum UTS name length
NQS_LEN_JSVNAME	15	Maximum job server name length
NQS_LEN_SCHNAME	15	Maximum batch scheduler name length
NQS_LEN_QUENAME	15	Maximum queue name length
NQS_LEN_REQNAME	63	Maximum request name length
NQS_LEN_ACCTCODE	127	Maximum account code length
NQS_LEN_MAILADDR	1023	Maximum mail address length
NQS_LEN_JOBCOND	255	Maximum length of job execution environment condition expression
NQS_LEN_ERRMSG	127	Maximum API error message length
NQS_LEN_USERNAME	47	Maximum user name length
NQS_LEN_GROUPNAME	47	Maximum group name length
NQS_LEN_VERSION	15	Maximum version character string length
NQS_LEN_LICNAME	15	Maximum license feature name length
NQS_LEN_COMMENT	63	Maximum comment character string length
NQS_LEN_RSTFPATH	47	Maximum length of restart file storage path name
NQS_LEN_SCHDMSG	4000	Maximum scheduler message length
NQS_LEN_COMMAND	2047	Maximum length of command
NQS_LEN_JOBNODSC	1535	Maximum length of job number string
NQS_LEN_NGRPNAME	15	Maximum length of node group name
NQS_LEN_TEMPLATENAME	47	Maximum length of template name
NQS_LEN_VMIMGNAME	47	Maximum length of image name
NQS_LEN_FLAVORNAME	47	Maximum length of flavor name
NQS_LEN_TEMPLATECUSTOM	400	Maximum length of custom define
NQS_LEN_TEMPLATECOMMENT	255	Maximum length of comment
NQS_MAX_CRNUM	20	Maximum number of custom resource
NQS_LEN_CRNAME	15	Maximum length of custom resource name
NQS_LIM_UNUSED	0	Unused specification of custom resource
NQS_LEN_CPUSETNAME	255	Maximum length of CPUSET name
NQS_LEN_CPUS	255	Maximum length of CPU number string
NQS_LEN_MEMS	255	Maximum length of memory node number string
NQS_LEN_GPUNAME	255	Maximum length of GPU device name

5.2. Structures

- **nqs_aid (Attribute identifier)**

```
typedef struct nqs_aid {
    int type;            /* attribute type */
    int scope;           /* scope */
} nqs_aid;
```

"type" is the type of the attribute while "scope" indicates a range to which the attribute is applied.

- **nqs_alist (Attribute list identifier)**

```
typedef int nqs_alist;
```

This structure is used to identify an attribute list and is an integer of 0 or above.

- **nqs_cotemplate (Container template)**

```
typedef struct nqs_cotemplate {
    char template_name[NQS_LEN_TEMPLATENAME+1]; /* template name */
    char image_name [NQS_LEN_VMIMGNAME+1];  /* image name */
    int cpunum;                              /* CPU number */
    int memsz;                               /* memory size */
    int memunit;                             /* memory size unit */
    int gpunum;                              /* GPU number */
    char custom[NQS_LEN_TEMPLATECUSTOM+1];   /* custom define */
    char comment[NQS_LEN_TEMPLATECOMMENT+1]; /* comment */
    int lock;                                /* lock status */
    int starttimeout;                        /* start time-out */
    int stoptimeout;                         /* stop time-out */
    int venum;                               /* VE number */
    nqs_usehca usecha;                       /* HCA port number */
} nqs_cotemplate;
```

This structure is information about Container template.

The member lock has a value of TEMPLATE_LOCK or TEMPLATE_UNLOCK.

- **nqs_cpuset (CPUSET information)**

```
typedef struct nqs_cpuset {
    int no;                            /* RSG number*/
    char name[NQS_LEN_CPUSETNAME+1];   /* CPUSET name */
    char cpus[NQS_LEN_CPUS+1];         /* core number  */
    char mems[NQS_LEN_MEMS+1];         /* memory node number */
} nqs_cpuset;
```

This structure has CPUSET information that is matched with RSG number.

- **nqs_crid (Custom resource ID)**

```
typedef struct nqs_crid {
    char cr_name[NQS_LEN_CRNAME+1];    /* Custom resource */
} nqs_crid;
```

cr_name has a custom resource name.

- **nqs_crresource (Custom resource amount control information)**

```
typedef struct nqs_crresource {
    int  cr_type;                       /* type */
```

```
   char cr_target[NQS_LEN_HOSTNAME+1];  /* target */
   int  cr_available;                   /* available */
} nqs_crresource;
```

This structure has amount control information of a custom resource.

The maximum of the simultaneous available resource to a target of the amount control indicated at cr_type and cr_target is stocked in cr_available.

cr_type is target type of amount control, and the set value is following one of them (macro-definition).

- CR_BSV_DEFAULT The target of amount control is BSV (default value).
- CR_HOST_DEFAULT The target of amount control is execution host (default value).
- CR_HOST The target of amount control is execution host that is, individual specified

The target by which cr_target is amount control only when cr_type is CR_HOST, an individual specified execution host name is stocked. Other cases are NULL character.

- **nqs_entry (Entry identifier)**

```
typedef int nqs_entry;
```

This structure is used to identify an entry and is an integer of 0 or above.

- **nqs_event (API event)**

```
typedef struct nqs_event {
   int event_id;             /* event identifier */
   time_t occur_time;        /* time of event occurrence time */
   union {
      struct evt_jsv jsv;    /* job server related event  */
      struct evt_qst qst;    /* queue status related event  */
      struct evt_qat qat;    /* queue attribute related event */
      struct evt_rst rst;    /* request state related event */
      struct evt_rat rat;    /* request attribute related event */
      struct evt_ngrp ngrp;  /* node group related event */
      struct evt_hst hst;    /* Execution host related event */
      struct evt_template tmpl; /* Template related event */
      struct evt_crs;           /* Custom resource related event */
   } cargo;                  /* event content */
} nqs_event;

struct evt_jsv {
   nqs_jsvid jsvid;   /* job server ID */
   nqs_hid hid;       /* host ID */
   int  state_link;   /* Link status */
   int   bind_count;  /* Bind status */
   nqs_vjsvid vjsvid; /* Internal Use Only */
};

struct evt_qst {
   nqs_qid qid;       /* queue ID */
   nqs_qst qst;       /* current queue state */
   int bind_id;       /* bound/unbound object ID */
   nqs_ngrpid ngrpid; /* bound/unbound node group */
};

struct evt_qat {
   nqs_qid qid;      /* queue ID */
   nqs_aid aid;      /* attribute ID */
   nqs_alist ad;     /* attribute list */
};

struct evt_rst {
   nqs_rid rid;      /* request ID */
   nqs_qid qid;      /* queue ID */
   nqs_rst rst;      /* current request state */
};

struct evt_rat {
   nqs_rid rid;      /* request ID */
   nqs_qid qid;      /* queue ID */
   nqs_aid aid;      /* attribute ID */
   nqs_alist ad;     /* attribute list */
};

struct evt_ngrp {
   nqs_ngrpid  ngrpid;   /* node group ID */
   nqs_aid aid;          /* attribute ID */
   nqs_alist ad;         /* attribute list */
};

struct evt_hst {
   nqs_hid hid;       /* host ID */
   nqs_hst hst;       /* host state */
   nqs_jsvid jsvid;   /* job server ID */
};

typedef struct evt_template {
   int type;                       /* template type */
   union {
      struct nqs_vmtemplate vm_tmpl;
      struct nqs_ostemplate os_tmpl;  /* OpenStack template */
      struct nqs_cotemplate co_tmpl;  /* Container template */
   }t1;
} evt_template;

struct evt_crs {
   nqs_crid crid;     /* Custom resource ID */
   int consumer;      /* consumer */
   nqs_aid aid;       /* attribute ID */
   nqs_alist ad;      /* attribute list */
};
```

Judge the "nqs_event.cargo" union member that stores event contents from the event type. Use NQSEVT_TYPE macro to get event types.

"evt_rst.qid" and "evt_rat.qid" set the identifier of a queue to which a request is submitted.

"evt_jsv.hid" sets the identifier of the execution host where the job server exists. "evt_qst.bind_id" sets the job server number of the job server that is connected or disconnected when the event identifier is NQSEVT_QST_BINDJSV or NQSEVT_QST_UNBINDJSV. Similarly, "evt_qst.bind_id" sets the scheduler number of the scheduler when the event identifier is NQSEVT_QST_BINDJSV or NQSEVT_QST_UNBINDJSV. The evt_qst.bind_id value is indeterminate for other queue status related events.

The attribute values in the attribute related event is data in the attribute list format and created in the API as an attribute list for event processing. The old attribute list is discarded each time the NQSevent function is executed.

When node group related event is received, job server ID (ATTR_JSVID) added to the node group (or removed from the node group) can be get using "aid" and "ad" in the "evt_ngrp" if event ID is NQSEVT_NGRP_ADDNODE or NQSEVT_NGRP_ADDNODE.

In case of the template event, when the template type "type" is "NQSII_TEMPLATE_TYPE_OPENSTACK", information on making, change or an eliminated OpenStack template can be acquired by "os_tmpl" in the "evt_template". And when the template type "type" is "NQSII_TEMPLATE_TYPE_CONTAINER", information on making, change or a template for eliminated containers can be acquired by "co_tmpl" in the "evt_template".

When an event identifier is NQSEVT_CRS_RESCHANGED in case of the custom resource system event, an addition or an eliminated amount control information list (ATTR_CRRESOURCE) can acquire it by aid, ad in evt_crs.

- **nqs_gbcset (GBC assign information)**

```
typedef struct nqs_gbcset {
    int  njobs_from;   /* Number od jobs (Start number of range) */
    int  njobs_to;     /* Number of jobs (Last number of range) */
    int  ngbc;         /* number of GBCs */
} nqs_gbcset;
```

Number of GBCs assigned to the request (number of jobs is in tha range of "njobs_from"-"njobs_to") is set to "ngbc".

- **nqs_gdesc (Group descriptor)**

```
typedef struct nqs_gdesc {
    gid_t gid;                            /* Group ID */
    char *gname [NQS_LEN_GROUPNAME+1];   /* Group Name */
} nqs_gdesc;
```

This structure indicates the group has a group name "gname" and a group ID of "gid".

- **nqs_gpuinfo (GPU detailed information)**

```
typedef struct nqs_gpuinfo {
     int device_no;                  /* GPU device number */
     char name [NQS_LEN_GPUNAME];    /* GPU device name */
     int total_global_mem;          /* Global memory (MB) */
} nqs_gpuinfo;
```

This structure has GPU detailed information.

- **nqs_hid (Host identifier)**

```
typedef struct nqs_hid {
    struct in_addr ip;      /* IP address */
} nqs_hid;
```

"ip" is the IP address of a remote host (such as execution host and client host) recognized by the batch server. It is a network byte order.

- **nqs_hilo (Maximum and Minimum value)**

```
typedef struct nqs_hilo {
    int high;      /* Maximum value */
    int low;       /* Minimum value */
} nqs_hilo;
```

"nqs_hilo" contains maximum value and minimum value of an attribute.

- **nqs_hilo_g (Maximum and Minimum value of Group)**

```
typedef struct nqs_hilo_g {
    nqs_gdesc group;     /* Group descriptor */
    nqs_hilo hilo;       /* Minimum and Maximum value */
} nqs_hilo_g;
```

"nqs_hilo_g" contains maximum and minimum value, and the restricted group information of an attribute.

- **nqs_hilo_u (Maximum and Minimum value of User)**

```
typedef struct nqs_hilo_u {
    nqs_udesc user;      /* User descriptor */
    nqs_hilo hilo;       /* Minimum and Maximum value */
} nqs_hilo_u;
```

"nqs_hilo_u" contains maximum and minimum value, and the restricted user information of an attribute.

- **nqs_hst (Host State)**

```
typedef struct nqs_hst {
    int state_curr;     /* Current State */
    int state_prev;     /* Previous State */
    int state_reason;    /* Reason */
    time_t state_time;   /* Occurrence Time */
}  nqs_hst;
```

"nqs_hst" structure contains the information of execution host state as "state_curr"(current state), "state_prev"(previous state), "state_reason"(state transition reason) and "state_time"(state transition time).

The state can take the following values.

HOSTST_ACTIVE	Running
HOSTST_INACTIVE	Stopped

The state transition reason is as follows.

State Transition Reason	Description
HOSTRSN_POWERSAVING_DCOFF	Inactivated by scheduler's Power-saving function.
HOSTRSN_RETURN_POWERSAVING_DCOFF	Activated by scheduler's Power-saving function.
HOSTRSN_JSVLINKUP	Activated by jobserver LINK UP.
HOSTRSN_JSVLINKDOWN	Inactivated by jobserver LINK DOWN.
HOSTRSN_NODE_ABNORMAL_STOP	Inactivated when node was down by failure detection program.
* When node agent was not used.	
HOSTRSN_JSVLINKUP	Activated by jobserver LINK UP.
HOSTRSN_JSVLINKDOWN	Inactivated by jobserver LINK DOWN.

- **nqs_int_g (Limit value of Group)**

```
typedef struct nqs_int_g {
   nqs_gdesc group;                  /* Group descriptor */
   int val;                         /* Limit value */
} nqs_int_g;
```

This structure indicates a group information and the limit value.

- **nqs_int_u (Limit value of User)**

```
typedef struct nqs_int_u {
   nqs_udesc user;                  /* User descriptor */
   int val;                         /* Limit value */
} nqs_int_u;
```

This structure indicates a user information and the limit value.

- **nqs_jcond (Job execution environment condition)**

```
typedef struct nqs_jcond {
   int jobno;                         /* Job number */
   char *condition;                   /* Condition expression */
} nqs_jcond;
```

"condition" specifies a condition that the execution host starts a job "jobno". The format of "condition" is defined for each batch scheduler. The character string from the top to NQS_LEN_JOBCOND bytes is valid in "condition".

- **nqs_jid (Job identifier)**

```
typedef struct nqs_jid {
   nqs_rid rid;           /* Request ID */
   int jobno;             /* Job number */
} nqs_jid;
```

"rid" is a request identifier of a parent request and "jobno" is a serial number (job number) of a job having an identical request as a parent. "jobno" of 0 indicates a master job and "jobno" of 1 or above is a slave job.

- **nqs_jsvid (Job server identifier)**

```
typedef struct nqs_jsvid {
   int jsvno;            /* Job server number */
} nqs_jsvid;
```

"jsvno" is a job server number (integer) in the range of 0 to NQS_MAX_JSVNO

- **nqs_jsvst (Job server status)**

```
typedef struct nqs_jsvst {
   int state_link;     /* Link Status */
   int state_bind;     /* Bind Status */
} nqs_jsvst;
```

"state_link" is the link status (TCP connection) between the job server while the batch server. "state_bind" is the connection status between the job server and the execution queue.

The "state_link" and "state_bind" values are as follows:

- state_link

JSVST_LINKUP	The job server is linked to the batch server.
JSVST_LINKDOWN	The job server is not linked to the batch server.

- state_bind

JSVST_BIND	The job server is connected to the queue.
JSVST_UNBIND	The job server is not connected to the queue.

- **nqs_keyval (Key value pair)**

```
typedef struct nqs_keyval {
   char *key;       /* Key */
   char *val;       /* Value */
} nqs_keyval;
```

This structure indicates a pair of a key character string and a value character string. The total length of "key" and "val" character strings must be up to 4000 bytes (conforming to the API packet restriction).

- **nqs_license (License information)**

```
typedef struct nqs_license {
   feature[NQS_LEN_LICNAME + 1]; /* License function name */
   int max_license;          /* Number of maximum licenses */
   int busy_license;         /* Number of licenses under use */
} nqs_license;
```

The number of licenses of each function names (FEATURE) is stored.

- **nqs_mdesc (Mail address descriptor)**

```
typedef struct nqs_mdesc {
   char mail[NQS_LEN_MAILADDR + 1];  /* Mail address */
} nqs_mdesc;
```

"mail" is a mail address in the "user_name@mail_domain" format. Delimit them with a space or comma character to specify two or more mail addresses.

- **nqs_migfile (information of the migration file)**

```
typedef struct nqs_migfile {
   char path[NQS_LEN_PATHNAME + 1];
                    /* absolute path of migration file  */
#   define MIG_TFL_CHKPF "chkpnt_files"
                    /* marker for checkpoint file */
} nqs_migfile;
```

"path" sets a absolute path of a user file, or a checkpoint file which is copied from the source migration host to the destination migration host. If path is a directory, the all files in the directory are copied. Exceptionally, all checkpoint files (open files at the time of execution of checkpoint) are copied if "path" is set to the string value chkpnt_files. Files are not copied if the specified path already exists on destination host.

- **nqs_migprm (Migration parameters)**

```
typedef struct nqs_migprm {
   char if_hname[NQS_LEN_HOSTNAME + 1];
         /* hostname of network I/F used for file transfer  */
   int sockbuf_sz;      /* size of socket buffer */
   nqs_range iobuf_sz;  /* size of file I/O */
} nqs_migprm;
```

"nqs_migprm" defines the parameters for control copying of files between execution hosts with the job migration. Files are copied by network-I/F specified for "if_hname". The socket buffer size specified for "sockbuf_sz" is set to both ends of socket. "sockbuf_sz" is an integer of 0 or above(unit:byte), and 0 means OS default value. "iobuf_sz" holds the buffer size used for disk-I/O, socket-I/O, and the value must be from 1 to 8388608(8M byte).(unit:byte)

- **nqs_netreq (Network request information)**

```
typedef struct nqs_netreq {
   nqs_rid rid;         /* Parent batch request ID */
   nqs_pdesc file;      /* Transmitting place/agency file name */
   nqs_udesc user;      /* Owner of network request */
   int dir;             /* Direction of staging */
   int stgno;           /* Staging file number */
   int state;           /* Status of network request */
   nqs_res res;         /* Result code */
} nqs_netreq;
```

"nqs_netreq" structure shows information on the network request. "rid" is the batch request ID by which this network request is created. "file" is staging object file name on the client host. If "dir" is STAGE_IN, the file is transmitting agency file name. If "dir" is STAGE_OUT, the file is transmission place file name.

"state" is the present status of the network request. The value is as shown in the table below.

REQST_QUEUED	waiting to start staging.
REQST_RUNNING	staging is being executed.
REQST_WAITING	waiting to retry.

When "state" is REQST_WAITING, "res" shows the cause of failure in staging.

- **nqs_ngrpid (Node group ID)**

```
typedef struct nqs_ngrpid {
   int type;                          /* Type of node group */
   char name[NQS_LEN_NGRPNAME + 1];  /* Node group name */
} nqs_ngrpid;
```

"name" is the node group name and "type" is the type of the node group. "type" can take the following values.

type value	Description
NQS_NGRPTYPE_COMMON	Common node group
NQS_NGRPTYPE_NWTOPOLOGY	Network topology type node group

- **nqs_nreqst (Requests number of each status)**

```
typedef struct nqs_nreqst {
   int outset;          /* REQST_OUTSET */
   int arriving;        /* REQST_ARRIVING */
   int waiting;         /* REQST_WAITING */
   int queued;          /* REQST_QUEUED */
   int prerunning;      /* REQST_PRERUNNING */
   int running;         /* REQST_RUNNING */
   int postrunning;     /* REQST_POSTRUNNING */
   int exiting;         /* REQST_EXITING */
   int exited;          /* REQST_EXITED */
   int held;            /* REQST_HELD */
   int holding;         /* REQST_HOLDING */
   int restarting;      /* REQST_RESTARTING */
   int suspending;      /* REQST_SUSPENDING */
   int suspended;       /* REQST_SUSPENDED */
   int resuming;        /* REQST_RESUMING */
   int migrating;       /* REQST_MIGRATING */
   int moved;           /* REQST_MOVED */
   int transiting;      /* REQST_TRANSITING */
   int staging;         /* REQST_STAGING */
   int chkpnting;       /* REQST_CHKPNTING */
   int g_queued;        /* REQST_GQUEUED */
   int forwarding;      /* REQST_FORWARDING */
} nqs_nreqst;
```

The number of requests counted of each status is stored.

- **nqs_nsubreq (Sub-request information)**

```
typedef struct nqs_nsubreq {
   int total;      /* Total number of sub-requests */
   int active;     /* Number of sub-requests resides in the batch
server */
   int done;       /* Number of exited sub-requests */
   char option[NQS_LEN_COMMAND+1];
                   /* String to describe sub-request numbers */
} nqs_nsubreq;
```

- **nqs_odesc (Object descriptor)**

```
typedef struct nqs_odesc {
   int obj_type;            /* Object type */
   union {
      nqs_schid schid;     /* Scheduler identifier */
      nqs_jsvid jsvid;     /* Job server identifier */
      nqs_hid hid;         /* Host identifier */
      nqs_qid qid;         /* Queue identifier */
      nqs_rid rid;         /* Request identifier */
      nqs_jid jid;         /* Job identifier */
      nqs_ngrpid ngrpid;   /* Node group identifier */
      nqs_wid wid;         /* Workflow identifier */
   } obj;
} nqs_odesc;
```

This structure is a control unit in the batch server. This is used to specify seven kinds of objects. Each identifier in the "obj" union must be the same as that specified by "obj_type". The table below shows available "obj_type" values and identifiers to be referred to.

obj_type value	Reference identifier	Description
NQS_OBJ_BSV	(None)	Indicates the batch server.
NQS_OBJ_SCH	obj.schid	Indicates the scheduler.
NQS_OBJ_JSV	obj.jsvid	Indicates the job server.
NQS_OBJ_HST	obj.hid	Indicates the host.
NQS_OBJ_QUE	obj.qid	Indicates the queue.
NQS_OBJ_REQ	obj.rid	Indicates the request.
NQS_OBJ_PRM	obj.rid	Indicates the parametric request.
NQS_OBJ_JOB	obj.jid	Indicates the job.
NQS_OBJ_NGRP	obj.ngrpid	Indicates the node group.
NQS_OBJ_WFL	obj.wid	Indicates the workflow.

- **nqs_ostemplate (OpenStack template)**

```
typedef struct nqs_ostemplate {
   char template_name[NQS_LEN_TEMPLATENAME+1];/* template name */
   char image_name [NQS_LEN_VMIMGNAME+1];      /* OS image name */
   int cpunum;                                 /* CPU number */
   int memsz;                                  /* memory size */
   int memunit;                               /* memory size unit */
   int gpunum;                               /* GPU number */
   char custom[NQS_LEN_TEMPLATECUSTOM+1];    /* custom define */
   char comment[NQS_LEN_TEMPLATECOMMENT+1]; /* comment */
   int lock;                                  /* lock status */
   char flavor[NQS_LEN_FLAVORNAME+1];         /* flavor name */
   int starttimeout;                          /* start time-out */
```

```
    int stoptimeout;                                /* stop time-out */
} nqs_ostemplate;
```

This structure is information about OpenStack template.

The member lock has a value of TEMPLATE_LOCK or TEMPLATE_UNLOCK.

- **nqs_pdesc (Path descriptor)**

```
typedef struct nqs_pdesc {
    char path[NQS_LEN_PATHNAME + 1];  /* Absolute path name  */
    char host[NQS_LEN_HOSTNAME + 1];  /* Host name */
} nqs_pdesc;
```

This structure indicates the file has an absolute path name of "path" on the host "host".

The path element can contain the meta characters below.

%r Expanded to a request ID (sequence number. host name).

%s Expanded to a sequence number in the request ID.

%m Expanded to a machine ID (integer value) in the request ID.

%j Expanded to a job number.

%% Expanded to a "%".

- **nqs_qdesc (Queue descriptor)**

```
typedef struct nqs_qdesc {
    char name[NQS_LEN_QUENAME + 1];   /* Queue name */
    char host[NQS_LEN_HOSTNAME + 1];  /* Host name */
    int retry_mode;      /* Retry mode flag */
    time_t retry_at;     /* next retry time (elapsed since EPOCH) */
    time_t retry_in;
                  /* retry mode start time (elapsed since EPOCH) */
} nqs_qdesc;
```

This structure indicates a queue with a queue name "name" on the host "host".

The above description indicates that the destination queue of an attribute to be referred to is in the RETRY mode when "retry_mode" is not 0. "retry_at" sets a time point where the next transfer is made while "retry_in" sets a time point where the destination queue first entered the RETRY mode in this event.

In attribute alteration, "retry_xxx" is ignored.

- **nqs_qid (Queue identifier)**

```
typedef struct nqs_qid {
    int type;                         /* Queue type */
    char name[NQS_LEN_QUENAME + 1];  /* Queue name */
} nqs_qid;
```

"type" and "name" are the type and name of a queue respectively. The queue types are as follows:

QUETYP_EXECUTE	Batch queue
QUETYP_INTERACTIVE	Interactive queue
QUETYP_ROUTING	Routing queue

QUETYP_NETWORK	Network queue

- **nqs_qst (Queue status)**

```
typedef struct nqs_qst {
   int state_run;     /* Permission to execute */
   int state_sub;     /* Permission to submit */
   int state_accept;   /* Permission to receive global requests */
} nqs_qst;
```

"state_run" indicates the permission status to execute a request, while "state_sub" indicates the permission status to submit a request.

- state_run

QUEST_ACTIVE	Can execute a request.
QUEST_INACTIVE	Cannot execute a request

- state_sub

QUEST_ENABLE	Can submit a request.
QUEST_DISABLE	Cannot submit a request

The "permission to execute a request" status does not change when "state_run" is QUEST_UNSPEC in attribute alteration. Similarly, the "permission to submit a request" status does not change when "state_sub" is QUEST_UNSPEC.

- **nqs_quecrinfo (Custom resource information about queue)**

```
typedef struct nqs_quecrinfo {
   nqs_crid crid;                  /* Custom resource name */
   int cr_std;                     /* standard value */
   nqs_hilo cr_limit;               /* resource limit range */
   int cr_permit_unused;             /* permit unused or not */
} nqs_quecrinfo;
```

This structure has amount control information on a custom resource of a queue. cr_permit_unused is a flag of whether amount control non-applicable specify of the custom resource which is at the time of request investment is permitted, and the price which can be taken is following one of them (macro-definition).

CR_PERMIT_UNUSED_YES	Permit to specify unused(0).
CR_PERMIT_UNUSED_NO	Not permit to specify unused(0).

- **nqs_range (Integer range)**

```
typedef struct nqs_range {
   unsigned int upper;       /* High limit value */
   unsigned int curr;        /* Current value */
   unsigned int lower;       /* Low limit value */
} nqs_range;
```

"upper" sets a high limit value and "lower" sets a low limit value. "curr" is always a value between "upper" and "lower" values and must not be outside this range. Only "curr" can be changed. ("upper" and "lower" cannot be changed.)

- **nqs_reqcrinfo (Custom resource information about request)**

```
typedef struct nqs_reqcrinfo {
    nqs_crid crid;                        /* Custom resource name */
    int cr_req;         /* Specified amount of the resource */
} nqs_reqcrinfo;
```

This structure has the use amount information of a custom resource of a request. The use amount about the custom resource indicated in crid is stocked in cr_req.

- **nqs_res (API result code)**

```
typedef struct nqs_res {
    int err;                              /* error number */
    char msg[NQS_LEN_ERRMSG + 1];    /* error message */
} nqs_res;
```

This structure is used to inform the content of an error of the API function. An error number representing the type of error is set for "err" while a character string describing the details of the error is set for "msg".

- **nqs_rgrp (Request group specification)**

```
typedef struct nqs_rgrp {
    nqs_rid rid;            /* lead request ID */
    int grpno;              /* request group number */
} nqs_rgrp;
```

"nqs_rgrp" specifies the request group which joins in request connection. "rid" is the request ID of the lead request (first submit request). "grpno" is the request group number and is an integer value over 0.

When lead request ID is the same request, execution is scheduled according to the following rules.

- All requests in the request group which includes the lead request are scheduled first.
- The request group with the same request group number is scheduled according to the same time.
- If the request-group-number is different, a small-number-group is faster than a large one for scheduling. Note that the following group is not scheduled until all requests in a preceding group end.

A negative value is stored in "grpno" if it is not the request connection.

- **nqs_rid (Request identifier)**

```
typedef struct nqs_rid {
    int seqno;             /* Sequence number */
    int mid;               /* Machine ID */
    int subreq_no;          /* Sub-request number */
} nqs_rid;
```

"seqno" is a sequence number made by the batch server when a request is created and "mid" is a machine ID of the batch server host (or machine ID of a submitting host when the request is created on the NQS).

"subreq_no" is the sub-request number of the parametric request. But the "rid" indicates the parametric request, "subreq_no" is -2 and the "rid" indicates the single request, "subreq_no" is -1.

Type of request	subreq_no
Single request (Not a parametric request)	-1
Parametric request	-2
Sub-request in a parametric request	>=0

- **nqs_rlim (Size/time limit value)**

```
typedef struct nqs_rlim {
    int max_limit;       /* High limit value  */
    int max_unit;        /* Unit of high limit value */
    int cur_limit;       /* Warning value */
    int cur_unit;        /* Unit of warning value */
    int std_limit;       /* Standard value */
    int std_unit;        /* Unit of standard value */
} nqs_rlim;
```

"max_limit" and "max_unit" indicate a high limit value. Similarly, "cur_xxx " and "std_xxx " indicate a warning value and a standard value respectively. "xxx_unit" values can be as shown below.

NQS_LIM_BYTE	In bytes
NQS_LIM_KBYTE	In kilobytes
NQS_LIM_MBYTE	In megabytes
NQS_LIM_GBYTE	In gigabytes
NQS_LIM_TBYTE	In terabytes
NQS_LIM_PBYTE	In petabytes
NQS_LIM_EBYTE	In exabytes
NQS_LIM_SEC	In seconds

The size/time is not limited when "xxx_limit" is NQS_LIM_UNLIMITED. In attribute alteration, attributes having NQS_LIM_UNSPECIFIED specified for "xxx_limit" are not altered. Specify NQS_LIM_UNSPECIFIED for "max_limit" and alter the attribute to alter warning and standard values only (without altering the high limit values). This is also true when altering other values (without altering the warning value or standard value).

"std_xxx" is valid only for attributes related to queue resource limits and ignored for other attributes. (These value are indeterminate in attribute reference.)

- **nqs_rlim_g (Size/time limit value of Group)**

```
typedef struct nqs_rlim_g{
   nqs_gdesc group;         /* Group descriptor */
   nqs_rlim rlim;           /* Size/time limit value */
} nqs_rlim_g;
```

This structure indicates a group information and the size/time limit value.

- **nqs_rlim_u (Size/time limit value of User)**

```
typedef struct nqs_rlim_u{
   nqs_udesc user;          /* User  descriptor */
   nqs_rlim rlim;           /* Size/time limit value */
} nqs_rlim_u;
```

This structure indicates a user information and the size/time limit value.

- **nqs_rnum (Number limit value)**

```
typedef struct nqs_rnum {
   int max_limit;      /* High limit value */
   int std_limit;      /* Standard value */
} nqs_rnum;
```

"max_limit" indicates a high limit value while "std_limit" indicates a standard value. The number is not limited when "xxx_limit" is NQS_LIM_UNLIMITED. In attribute alteration, attributes with NQS_LIM_UNSPECIFIED specified for "xxx_limit" are not altered. To alter the standard value only (without altering the high limit). Specify NQS_LIM_UNSPECIFIED for "max_limit" to alter the attribute. This is also true to alter other values (without altering the standard value).

"std_limit" is valid only for attributes related to queue resource limits and ignored for other attributes. (In attribute reference, this value is indeterminate.)

- **nqs_hca (HCA port number Ranges)**

```
typedef struct nqs_hca {

   nqs_rrange for_io; /* HCA port number range for ScaTeFS */

   nqs_rrange for_mpi; /* HCA port number range for MPI */

   nqs_rrange for_all; /* HCA port number range for both of ScaTeFS
and MPI*/

} nqs_hca;
```

The range of HCA port number by the type of HCA

- **nqs_rrange(Resource Ranges)**

```
typedef struct nqs_nqs_rrange{
   int min_limit;     /* Minimum limit */
   int min_unit;      /* Unit of minimum limit */
   int max_limit;     /* Maximum limit */
   int max_unit;      /* Unit of maximum limit */
```

```
    int warn_limit;     /* Warning limit */
    int warn_unit;      /* Unit of Warning limit */
    int std_limit;      /* Standard limit */
    int std_unit;       /* Unit of Standard limit */
} nqs_rrange;
```

This structure indicates resource ranges value.

- **nqs_rsgavg (Average information)**

```
typedef struct nqs_rsgavg {
    double avg01;       /* Average of latest one minute */
    double avg05;       /* Average of latest five minutes */
    double avg15;       /* Average of latest fifteen minutes  */
} nqs_rsgavg;
```

"nqs_rsgavg" sets average information of the execution host. Each member sets the average of loads (average processes waiting to be executed) or the CPU average (multiplying the CPU activity ratio by the number of CPUs) for a unit time period (1, 5 or 15 minutes).

- **nqs_rsgres (Resource information)**

```
typedef struct nqs_rsgres {
    int initial;       /* Quantity of resource allocated */
    int using;         /* Quantity of resource in use */
    int maximum;       /* Maximum quantity of resource available */
    int unitsz;        /* Unit size of resource quantity */
} nqs_rsgres;
```

"nqs_rsgres" sets resource information of the execution host. "nqs_rsgres" sets resource information of each execution host. The "initial" value is always equal to the "maximum" value which set the physical memory size or swapping size (in pages) of the host. "using" sets the physical memory size or swapping size (in pages) currently used. The page size is stored in "unitsz" with byte in case of memory/swap. "unitsz" is always 1 for CPU number. The "initial" and "maximum" values are the number of CPUs recognized by the OS. "using" sets the number of CPUs in use (multiplying the number of CPUs by the ratio of currently used) when the attribute type is ATTR_RBCPUNM.

- **nqs_rsginfo (RSG information)**

```
typedef struct nqs_rsginfo {
        int rsgno;             /* RSG number */
        nqs_rsgres rbspmem;   /* RB: SP memory */
        nqs_rsgres rblpmem;   /* RB: LP memory */
        nqs_rsgres rbspswap;  /* RB: SP swap size */
        nqs_rsgres rblpswap;  /* RB: LP swap size */
        nqs_rsgres rbcpunum;  /* RB: CPU */
        nqs_rsgavg rbldavg;   /* RB: Load average */
        nqs_rsgavg rbcpuavg;  /* RB: CPU average */
        nqs_rsgres rbgpunum;  /* RB: GPU */
        nqs_rsgres rbvenum;   /* RB: VectorEngine */
} nqs_rsginfo;
```

"nqs_rsginfo" is the RSG setting information of execution host. When the execution host is Linux, "rsgno" is 0 only and only "rbspmem", "rbspswap", "rbcpunum", "rbldavg", "rbcpuavg" and "rbgpunum" have values.

- **nqs_rst (Request status)**

```
typedef struct nqs_rst {
    int state_curr;     /* current request state*/
    int state_prev;     /* just previous request state  */
    int state_reason;   /* Reason of state transition  */
    time_t state_time;  /* Time of transition to current request
state */
    int stalled;        /* True when the request is stalled */
    int exit_status;    /* Request exit status  */
    int elaps_time;     /* Real elapse time of request execution */
    int deleted_by; /* User privilege that deleted the request */
    nqs_res res;        /* Result code  */
    nqs_bsv bsv;        /* Forwarded destination BSV */
    int bsv_selected;  /* True if execution BSV is selected.*/
    int vm_ctrl;   /* True if VM/Container is operated in PRR or POR*/
} nqs_rst;
```

"state_curr" sets the current request state while "state_prev" sets the previous request state. "state_reason" sets the cause of the state transition. When the state transition is due to an error, "res" sets the error type. When the result code indicates a normal status transition, "res.err" sets NQS_ESUCCESS. "state_time" sets the time period (seconds elapsed since EPOCH) where the request enters the current state.

When "state_curr" is REQST_POSTRUNNING, "exit_status" sets the exit status of the request (master job).

Request state	
REQST_OUTSET	Initial status
REQST_ARRIVING	Receiving
REQST_WAITING	Waiting for the execution time
REQST_QUEUED	Waiting for the execution starts
REQST_STAGING	Staging
REQST_PRERUNNING	Pre-running
REQST_RUNNING	Running
REQST_POSTRUNNING	Post-running
REQST_EXITING	Exiting
REQST_EXITED	Exited
REQST_CHKPNTING	periodic checkpointing
REQST_HELD	Held
REQST_HOLDING	Checkpointing for hold
REQST_RESTARTING	Restarting
REQST_SUSPENDING	Suspending
REQST_SUSPENDED	Suspended
REQST_RESUMING	Resuming
REQST_MIGRATING	Migrating
REQST_MOVED	Moved

Reason for state transition

REQRSN_DELETE	DELETE request
REQRSN_YET_EXETIME	Before preset execution time
REQRSN_JUST_EXETIME	After preset execution time
REQRSN_ARRIVE	Receiving from other queue
REQRSN_ARRIVE_SUCCESS	Reception succeeded
REQRSN_ARRIVE_FAIL	Reception failed
REQRSN_SUBMIT	SUBMIT request
REQRSN_STAGEIN	Stage-in request
REQRSN_STAGEIN_SUCCESS	Stage-in succeeded
REQRSN_STAGEIN_FAIL	Stage-in failed
REQRSN_PRERUN_SUCCESS	Pre-running succeeded
REQRSN_PRERUN_FAIL	Pre-running failed
REQRSN_RUN	RUN request
REQRSN_EXIT	Exited
REQRSN_RERUN	RE-RUN request
REQRSN_POSTRUN_SUCCESS	Post-running succeeded
REQRSN_POSTRUN_FAIL	Post-running failed
REQRSN_REQUE_SUCCESS	Re-queuing
REQRSN_DONE	Termination
REQRSN_CHKPNT	CHECKPOINT request
REQRSN_CHKPNT_SUCCESS	CHECKPOINT succeeded
REQRSN_CHKPNT_FAIL	CHECKPOINT failed
REQRSN_HOLD	HOLD request
REQRSN_HOLD_SUCCESS	HOLD succeeded
REQRSN_HOLD_FAIL	HOLD failed
REQRSN_RELEASE	RELEASE request
REQRSN_RESTART	RESTART request
REQRSN_RESTART_SUCCESS	RESTART succeeded
REQRSN_RESTART_FAIL	RESTART failed
REQRSN_SUSPEND	SUSPEND request
REQRSN_SUSPEND_SUCCESS	SUSPEND succeeded
REQRSN_SUSPEND_FAIL	SUSPEND failed
REQRSN_RESUME	RESUME request
REQRSN_RESUME_SUCCESS	RESUME succeeded
REQRSN_RESUME_FAIL	RESUME failed
REQRSN_MIGRATE	MIGRATE request
REQRSN_MIGRATE_SUCCESS	MIGRATE succeeded
REQRSN_MIGRATE_FAIL	MIGRATE failed
REQRSN_MOVE	MOVE request
REQRSN_MOVED	Moved from other queue
REQRSN_SYSTEM_FAILURE	System failure on Execution host
REQRSN_ROLLBACK	Rollback request
REQRSN_FORWARD_SUCCESS	Forwarding success
REQRSN_FORWARD_FAIL	Forwarding failed
REQRSN_FORWARD_RFAIL	Forwarding failed (Enable retry)
REQRSN_BSVSELECT	Batch server selected
REQRSN_ALLEXITED	All subrequests terminated
REQRSN_FORCELOCAL	Force execution at local BSV
REQRSN_EXIT_FAIL	Failed to delete sub-request
REQRSN_MOVE_FAIL	Failed to move request

REQRSN_STAGEOUT_FAIL	External staging failed

For details of request states, see 2 State Transition of Request.

"bsv" was for global requests but is no currently in use.

"vm_ctrl" indicates whether a virtual machine by OpenStack or a container by Docker is in the start/stop processing.

- in case of 1 : A virtual machine by OpenStack is in the start/stop process.
- in case of 2 : A container by Docker is in the start/stop process.
- in case of 0 : Other than above.

- **nqs_rstf (Restart file information)**

```
typedef struct nqs_rstf {
    time_t date;               /* Creating date  */
    long long size;           /* File size  */
    int intval;               /* Interval of periodic checkpoint */
    char rstfdir[NQS_LEN_RSTFPATH + 1];
                              /* Restart file stored directory */
} nqs_rstf;
```

Information of the restart file of the job is stored. The periodic checkpoint is OFF mode when "interval" is 0. The restart file does not exist when "time" is 0.

- **nqs_schid (Scheduler identifier)**

```
typedef struct nqs_schid {
    int schno;           /* Scheduler number */
} nqs_schid;
```

"schno" is a batch scheduler number (integer) in the range of 0 to NQS_MAX_SCHNO.

- **nqs_socket (Socket resource information)**

```
typedef struct nqs_socket {
    int no;                              /* Socket number */
    int initial_cpu;                     /* CPU number */
    int using_cpu;                       /* CPU usage */
    long long initial_mem;               /* Memory size */
    long long using_mem;                 /* Memory usage */
    char cpus[NQS_LEN_CPUS+1];           /* core number */
    char mems[NQS_LEN_MEMS+1];           /* memory node number */
} nqs_socket;
```

This structure has socket resource information of every socket.

- **nqs_stgfile (Staging file information)**

```
typedef struct nqs_stgfile {
    int dir;           /* Direction of staging  */
    char cli_host[NQS_LEN_HOSTNAME + 1];
```

```
                    /* Client host name  */
   char cli_path[NQS_LEN_PATHNAME + 1];
                    /* Complete path name on client host  */
   char exe_path[NQS_LEN_PATHNAME + 1];
                    /* Relative path name on execution host */
   char exe_jobno[NQS_LEN_JOBNODSC + 1];
                    /* Job number array */
   int stgno;       /* Staging file number */
   int status;      /* Status of staging */
} nqs_stgfile;
```

This "nqs_stgfile" describes a file to be staged. "dir" sets the direction of staging ("STAGE_IN" for a file copy direction from client host to execution host or "STAGE_OUT" for the opposite direction).

When the staging direction is "STAGE_IN," the file "cli_path" on the client host "cli_host" is copied to a file "exe_path" on each execution host that executes a job specified by "exe_jobno".

The file "exe_path" on each execution host that executes a job specified by "exe_jobno" is added to a single file and the single file is copied to the file "cli_path" on the client host "cli_host" when the staging direction is "STAGE_OUT".

A subscript in the "exe_jobno" arrangement corresponds to a job number, while only job numbers corresponding to an element of a real value is staged. The following strings can be specified for "exe_jobno".

(1) Single number	0
(2) All jobs using "ALL"	ALL
(3) Two or more numbers using delimiters ","	0,2,5
(4) Consecutive numbers using "-"	0-4
(5) Combination of (3) and (4)	0,2,4-6

"cli_path" must be specified absolutely while "exe_path" must be specified relatively. The path must be specified relative to the environment variable "STGDIR" when accessing "exe_path" from a job for I/O.

The specified path is directory if a last character of "cli_path" or "exe_path" is '/'.

The "cli_path" and "exe_path" elements contain the meta characters below.

%r Expanded to a request ID (sequence number. host name).<

%s Expanded to a sequence number in the request ID.

%m Expanded to a machine ID (integer value) in the request ID.

%j Expanded to a job number.

%% Expanded to a "%".

"stgno" is a serial number for the each stage in/out file.

"status" shows the present staging situation. The value is as shown in the table below.

STGFST_NOTYET	staging has not been executed yet.
STGFST_PRGRESS	staging is being executed now.
STGFST_SUCCESS	staging succeeded.
STGFST_FAILURE	staging failed.

- **nqs_temp_reqs (Number of requests using template)**

```
typedef struct nqs_temp_reqs {
    char template_name[NQS_LEN_TEMPLATENAME+1]; /* Template name */
    int requests;            /* Number of requests using template */
} nqs_temp_reqs;
```

This structure indicates the template name has "template_name", and number of requests using this template has "requests".

- **nqs_template (Template information)**

```
typedef struct nqs_template {
    int type;                              /* template type */
    union {
        struct nqs_vmtemplate vm_tmpl;
        struct nqs_ostemplate os_tmpl;  /* OpenStack template */
        struct nqs_ostemplate co_tmpl;  /* Container template */
    }t1;
} nqs_template;
```

This structure indicates the template type has "type", and definition information of the template is below.

Type of template	type	Structure of the template information
OpenStack	NQSII_TEMPLATE_TYPE_ OPENSTACK	struct nqs_ostemplate os_tmpl
Container	NQSII_TEMPLATE_TYPE_ CONTAINER	struct nqs_cotemplate co_tmpl

- **nqs_udesc (User descriptor)**

```
typedef struct nqs_udesc {
    char name[NQS_LEN_USERNAME + 1];  /* User name */
    uid_t uid;                        /* User ID */
    gid_t gid;                        /* Group ID */
} nqs_udesc;
```

This structure indicates the user has a user name "name" and a user ID/group ID of "uid/gid" respectively.

- **nqs_uexit (User Exit information)**

```
#define UEXNUM_SCR  4  /* Maximum number of script per location  */

typedef struct nqs_uexit {
   int location;
           /* Location from which user EXIT script is started */
   struct {
       int order;
           /* Execution sequence number of User EXIT script */
       char name[NQS_LEN_FILENAME+1];
           /* File name of user EXIT script */
   } uexscr[UEXNUM_SCR];
} nqs_uexit;
```

Information on user EXIT executed in the location specified by "location" is stored.

The value which can be specified for location is as follows.

UEXLOC_PRERUN	Just before starting of execution [in the PRE-RUNNING state]
UEXLOC_PSTRUN	Just after termination of execution [in the POST-RUNNING state]
UEXLOC_HLDING	Just after retrieving a hold checkpoint [in the HOLDING state]
UEXLOC_RSTING	Just before restarting from the checkpoint [in the RESTARTING state]

Information on each array element in user EXIT script executed by a specified location is stored up to four scripts in the array of "uexscr" structure which consists of four elements. In the "uexscr" structure, "order" is execution sequence in script and "name" is script file name.

The value which can be specified for "order" is as follows.

UEXODR_NON	Not executed
UEXODR_1ST	First execution in the location
UEXODR_2ND	2nd execution in the location
UEXODR_3RD	3rd execution in the location
UEXODR_4TH	4th execution in the location

"name" stores the file name of User EXIT script. The file name stored in "name" is regarded to exist in "/opt/nec/nqsv/sbin/uex_prog/" on the batch server host. "name" cannot include path element ('/').

When order is set to UEXODR_NON, the value of the corresponding name is undefined.

- **nqs_upp (UserPP script information)**

```
typedef struct nqs_upp {
    int location;                          /* location */
    char path[NQS_LEN_PATHNAME+1];         /* path */
} nqs_upp;
```

This structure has a information about UserPP.

The value of location is one of following.

UEXLOC_PRERUN	Just before starting of execution [in the PRE-RUNNING state]
UEXLOC_PSTRUN	Just after termination of execution [in the POST-RUNNING state]

- **nqs_wid (Workflow ID)**

```
typedef struct nqs_wid {
    int id;              /* Sequence number of the Workflow */
    int mid;             /* Machine ID the workflow created */
} nqs_wid;
```

"id" is the sequential number when a workflow is created in the batch server. "mid" is the batch server's machine ID.

6. API Functions

6.1. Attribute list functions

6.1.1. Create Attribute List

Name

NQSalist -- Create Attribute List

Format

```
#include <nqsv.h>

nqs_alist NQSalist(nqs_alist ad, nqs_aid *aid, nqs_res *res)
```

Function

Creates an attribute list. NQSalist creates a new attribute list having an attribute header "aid" as the top element when the attribute list identifier "ad" is a negative integer or add "aid" to an existing attribute list "ad" when the attribute list identifier "ad" is an integer of 0 or above.

Return value

When executed successfully, NQSalist() returns an integer of 0 or above as the attribute list identifier. When an error occurs, NQSalist() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_EEXIST]

Attribute header "aid" already exists in the attribute list.

[NQS_ENOENT]

No attribute list having "ad".

[NQS_EINVAL]

Invalid argument specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

Related items

NQSafree(3), NQSaadd(3), NQSaref(3), NQSadel(3)

6.1.2. Release Attribute List

Name

NQSafree -- Free Attribute List

Format

```
#include <nqsv.h>

int NQSafree(nqs_alist ad, nqs_aid *aid, nqs_res *res)
```

Function

Releases an attribute list of the specified attribute list identifier "ad" and deletes only attributes having "aid" from the attribute list when "aid" is not a null or deletes all attributes in the list and releases the attribute list itself when "aid" is a null.

Return value

When executed successfully, NQSafree() returns 0. When an error occurs, NQSafree() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOENT]

No attribute list having "ad".

No attribute header "aid" in the attribute list.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSalist(3), NQSaadd(3), NQSaref(3), NQSadel(3)

6.1.3. **Add Values to Attribute List**

Name

NQSaadd -- Add Attribute Value to Attribute List

Format

```
#include <nqsv.h>

int NQSaadd(nqs_alist ad, nqs_aid *aid, void *val, size_t sz, nqs_res *res)
```

Function

Adds an attribute value "val" to under an attribute header having "aid" in the attribute list specified by an attribute list identifier "ad" When the attribute header already has an attribute value, "val" is added to the end of the attribute value chain. Specify the size of "val" (in bytes) for "sz".

"val" types are dependent upon attribute values. For more information, see Attribute List.

Notes

To specify an attribute value of the character type (char *) as "val," specify the size of the character string (to the end character '\0') for "sz".

Return value

When executed successfully, NQSaadd() returns 0. When an error occurs, NQSaadd() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOENT]

No attribute list having "ad".

No attribute header "aid" in the attribute list.

[NQS_EINVAL]

Invalid argument specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

Related items

NQSalist(3), NQSafree(3), NQSaref(3), NQSadel(3)

6.1.4. **Delete Values to Attribute List**

Name

NQSadel -- Delete Attribute Value from Attribute List

Format

```
#include <nqsv.h>

int NQSadel(nqs_alist ad, nqs_aid *aid, void *val, nqs_res *res)
```

Function

Deletes an attribute value having "aid" from the attribute list specified by an attribute list identifier "ad". When "val" is NULL, all attribute values are deleted. If "val" is not NULL then the attribute value equal to "val" is deleted. When attribute values to be deleted are not exist, NQSadel() ends in normal.

Return value

When executed successfully, NQSadel() returns 0. When an error occurs, NQSadel() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOENT]

No attribute list having "ad".

No attribute header "aid" in the attribute list.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSalist(3), NQSafree(3), NQSaref(3), NQSaadd(3)

6.1.5. **Refer to Values in Attribute List**

Name

NQSaref -- Refer to Attribute Value in Attribute List

Format

```
#include <nqsv.h>

void *NQSaref(nqs_alist ad, nqs_aid *aid, void *val, nqs_res *res)
```

Function

Searches an attribute value of an attribute "aid" in an attribute list having an attribute list identifier "ad" and returns its pointer. NQSaref returns a pointer to an attribute under an attribute header when "val" is a null or a pointer to an attribute value just under an attribute value specified by "val" when "val" is not a null.

When two or more attribute values are chained, you can refer to all attribute values in the chain by specifying, the pointer that is returned by the first NQSaref() for "val" in the next NQSaref().

Return value

When executed successfully, NQSaref() returns a pointer to an attribute value in the attribute list. When the attribute "aid" has no attribute value, NQSaref() returns a null and NQS_EEMPTY in the error number. When an attribute value in the end of the attribute value chain is specified for "val," NQSaref() returns a null and sets NQS_EALLOVER in the error number. When an error occurs, NQSaref() returns a null and sets a result code in "res".

As NQSaref() returns a "void" type pointer, cast with a type specific to each attribute value for reference. For details of attribute value types, see Attribute List.

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOENT]

No attribute list having "ad".

No attribute header "aid" in the attribute list.

[NQS_EEMPTY]

The specified attribute has no attribute value.

[NQS_EALLOVER]

Tried to read an attribute value below the end of the attribute value chain.

[NQS_EJSVDOWN]

Some of the execution host attributes cannot be referenced when the job server is down.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSalist(3), NQSafree(3), NQSaadd(3), NQSadel(3)

6.1.6. **Operation of Attribute Entry**

Name

NQSopenattr, NQSreadattr, NQScloseattr -- Operate Attribute Entry

Format

```
#include <nqsv.h>

nqs_entry NQSopenattr(nqs_odesc *odesc, int object, nqs_alist ad, nqs_res *res)
nqs_alist NQSreadattr(nqs_entry entry, nqs_res *res)
int NQScloseattr(nqs_entry entry, nqs_res *res)
```

Function

NQSopenattr() selects objects related to a parent object "odesc" from objects (job server, request, etc.) specified by "object," creates an attribute list "ad" for each of the selected objects, gets the attribute values in bulk, creates a list (attribute entry) having the attribute lists as its elements, initializes its index to 0, and returns an entry identifier to identify it. When an error occurs, NQSopenattr() returns a negative integer and sets an API result code in "res".

The table below lists objects that can be specified for "odesc" and "object" and their available combinations.

odesc.obj_type	obj	OBJECT	Objects
NQS_OBJ_BSV	None	NQS_OBJ_SCH	All schedulers in the system
		NQS_OBJ_HST	All execution hosts in the system
		NQS_OBJ_JSV	All job servers in the system
		NQS_OBJ_QUE	All queues in the system
		NQS_OBJ_REQ	All requests in the system
		NQS_OBJ_BREQ	All batch requests in the system
		NQS_OBJ_IREQ	All interactive requests in the system
		NQS_OBJ_PRM	All parametric requests in the system
		NQS_OBJ_JOB	All jobs in the system
		NQS_OBJ_NGRP	All node groups in the system
NQS_OBJ_SCH	odesc.obj.schid	NQS_OBJ_QUE	All queues bound to the specified scheduler
NQS_OBJ_JSV	odesc.obj.jsvid	NQS_OBJ_JOB	All jobs controlled by the specified job server
		NQS_OBJ_QUE	All queues bind the specified job server
		NQS_OBJ_NGRP	All node groups include the specified job server
NQS_OBJ_HST	odesc.obj.hid	NQS_OBJ_JSV	All job servers on the specified execution host
NQS_OBJ_QUE	odesc.obj.qid	NQS_OBJ_JSV	All job servers bound to the specified queue
		NQS_OBJ_REQ	All requests submitted to the

			specified queue
		NQS_OBJ_PRM	All parametric requests submitted to the specified queue
		NQS_OBJ_NGRP	All node groups bound to the specified queue
NQS_OBJ_REQ	odesc.obj.rid	NQS_OBJ_JOB	All jobs having the specified request as the parent
NQS_OBJ_NGRP	odesc.obj.ngrpid	NQS_OBJ_JSV	All job servers included in the specified node group
		NQS_OBJ_HST	All execution hosts included in the specified node group
		NQS_OBJ_QUE	All queues bind the specified node group
NQS_OBJ_PRM	odesc.obj.rid	NQS_OBJ_REQ	All sub-requests in the specified parametric request

NQSreadattr() returns an attribute entry pointed by the internal index by the "nqs_alist" type and increments the internal index by one. When the last attribute entry comes, NQSreadattr() returns -1 and sets NQS_EALLOVER in the error number of "res". When an error occurs, NQSreadattr() returns -1 and sets an API result code in "res".
As the "nqs_alist" type variable that NQSreadattr() returned is an attribute list identifier, you can refer to respective attribute values by NQSalist().

NQScloseattr() deletes the attribute entry created by NQSopenattr(). When executed successfully, NQScloseattr() returns a 0. When an error occurs, NQScloseattr() returns a negative integer and sets an API result code in "res".

Notes

When you refer to the attribute list identifier returned by NQSreadattr() after NQScloseattr() was executed, the operation is not assured.
The attribute without a reference right is not acquired and simply ignored. (This is not an error.)

The attribute acquisition using this function is functionally almost the same as the attribute acquisition by a set of the NQSopen{jsv,hst,que,req,job} function and the NQSattr{jsv,hst,que,req,job} function, but faster than it.

Error

When an error occurs, one of the result codes below is set as an error number in the result code.
[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot send the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_ENOENT]

Invalid entry identifier specified.

Invalid attribute list identifier specified.

[NQS_EUNKNOWN]

Unknown attribute type specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSaref(3)

6.2. API initialize/exit functions

6.2.1. Open API Link

Name

NQSconnect -- Open API Link

Format

```
#include <nqsv.h>

int NQSconnect(char *hostname, int port, int priv, nqs_res *res)
```

Function

Establishes an API link with a batch server on the batch server host "hostname". When "hostname" is a null, NQSconnect uses the name of a batch server host in "/etc/opt/nec/nqsv/api_client.conf" if it is specified or tries to connect to "localhost" if it is not specified.

When "port" is 1 or above, NQSconnect uses a port number "port" for connection with the batch server. When "port" is 0, NQSconnect uses a port number in "api_client.conf" if it is specified or a default port number if it is not specified for connection with the batch server.

"priv" specifies an API authority. The API authority is used to limit available API functions. There are five API authorities.

PRIV_SCH	scheduler authority
PRIV_MGR	manager authority
PRIV_OPE	operator authority
PRIV_GMGR	group manager authority
PRIV_SPU	special user authority
PRIV_USR	general user authority

The PRIV_SCH authority is the highest and the PRIV_USR authority is the lowest. The PRIV_SCH authority (scheduler authority) can use all API functions, but PRIV_MGR and PRIV_OPE authorities can use only part of API functions. PRIV_GMGR authority is the manager authority which limited functions to the scope of management groups. The lowest PRIV_USR authority (general user authority) cannot use API functions that require NQS operator or higher authority. The PRIV_SPU authority can refer informations of requests, jobs, etc for other users in addition to the PRIV_USR authority.

Notes

The socket descriptor returned by NQSconnect() is closed in the API in the execution of

NQSdisconnect(). If the socket descriptor is closed in the other method, the API operation may not be assured.

Return value

When executed successfully, NQSconnect() returns a socket descriptor to supervise API events. When an error occurs, NQSconnect() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ELICENSE]

Cannot get a license for API link.

[NQS_EISCONN]

The API link is already established.

[NQS_ENETDB]

Cannot find a host of the specified host name.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_ESYSCAL]

Error occurred in the system call

[NQS_EPERM]

The batch server rejected a connection by the specified API authority.

[NQS_EACCTAUTH]

The access by an unknown user name was refused.

[NQS_EINVAL]

Invalid argument specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

File

/etc/opt/nec/nqsv/api_client.conf	NQSV/API client setting file

Related items

NQSdisconnect(3), NQSevent(3)

6.2.2. **Close API Link**

Name

NQSdisconnect -- Close API Link

Format

```
#include <nqsv.h>

int NQSdisconnect(nqs_res *res)
```

Function

Breaks the API link.

Return value

When executed successfully, NQSdisconnect() returns 0. When an error occurs, NQSdisconnect() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, the result code below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

Related items

NQSconnect(3)

6.3. API event related functions

6.3.1. Get API Event

Name

NQSevent -- Get API Event

Format

```
#include <nqsv.h>

int NQSevent(nqs_event *event, nqs_res *res)

NQSEVT_TYPE(event_id)
```

Function

NQSevent() reads one of the received API and stores it in "event". NQSEVT_TYPE() is a macro that returns the event type of the event identifier "event_id".

Notes

The attribute value in the event is stored in the dedicated attribute list in the API. This attribute list is newly created each time NQSevent() reads an event containing attribute values (and the preceding attribute list is discarded at that time).

Return value

When executed successfully, NQSevent() returns 0. When an error occurs, NQSevent() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_ENOEVENT]

No event received.

[NQS_EUNKNOWN]

Unknown attribute in the event.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSevflt(3)

6.3.2. **Set Event Filter**

Name

NQSevflt -- Set Event Filter

Format

```
#include <nqsv.h>

int NQSevflt(int event_type, int op, nqs_res *res)
```

Function

NQSevflt() sets an event filter in the API link to select an event type that the API client receives. The target event type is specified by "event_type" and sets in the filter according to the operation "op". Values below can be specified for "op".

EVFLT_ADD

Adds "event_type" to the event filter.

EVFLT_DEL

Deletes "event_type" from the event filter.

Return value

When executed successfully, NQSevflt() returns 0. When an error occurs, NQSevflt() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EEXIST]

Un-cataloged event type to be added.

[NQS_ENOENT]

Un-cataloged event type to be deleted.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSevent(3)

6.4. Scheduler related functions

6.4.1. Register Scheduler Identifier

Name

NQSregsch -- Catalog Batch Scheduler Identifier and Name

Format

```
#include <nqsv.h>

int NQSregsch(nqs_schid *schid, char *name, char *version, nqs_res *res)
```

Function

NQSregsch() catalogs the identifier and name of a batch scheduler that called NQSregsch(). NQSregsch() cannot change the identifier that is already cataloged. You cannot specify an identifier that is used by the other scheduler.

An integer value in the range of 0 to NQS_MAX_SCHNO can be specified as a scheduler number (nqs_schid.schno) in the batch scheduler identifier.

"name" can be any character string (up to NQS_LEN_SCHNAME) representing a scheduler name. When "name" is a null, the batch server assigns a proper scheduler name.

"version" can be any character string (up to NQS_LEN_VERSION) representing a scheduler version.

Notes

NQSregsch() requires an API authority of PRIV_SCH or higher to run.

Return value

When executed successfully, NQSregsch() returns 0. When an error occurs, NQSregsch() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot send the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to run the API function.

[NQS_ETOOLONG]

Too long scheduler name specified "name".

Too long character string specified "version".

[NQS_EOUTRNG]

The specified scheduler number is outside the valid range.

[NQS_EEXIST]

The specified identifier is now used by the other scheduler.

[NQS_EALREADY]

The specified scheduler already has an identifier cataloged.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.4.2. **Operation of Scheduler Entry**

Name

NQSopensch, NQSreadsch, NQSrewindsch, NQSclosesch -- Operate Scheduler Entry

Format

```
#include <nqsv.h>

nqs_entry NQSopensch(nqs_odesc *odesc, nqs_res *res)
nqs_schid *NQSreadsch(nqs_entry entry, nqs_res *res)
int NQSrewindsch(nqs_entry entry, nqs_res *res)
int NQSclosesch(nqs_entry entry, nqs_res *res)
```

Function

NQSopensch() selects a scheduler related to a specified object "odesc" from batch schedulers linked to the batch server, creates a list (scheduler entry) having the scheduler identifier as the element, initializes the internal index to 0, and returns an entry identifier to identify it. When an error occurs, NQSopensch() returns a negative integer and sets an API result code in "res".

The table below lists objects that can be specified for "odesc".

odesc.obj_type value	member to be referred	Objects
NQS_OBJ_BSV	None	All schedulers in the system

NQSreadsch() returns an entry pointed to by the internal pointer by the "nqs_schid" type pointer and increments the internal index by one. When the last entry comes, NQSreadsch() returns a null and sets NQS_EALLOVER in the error number of "res". When an error occurs, NQSreadsch() returns a null and sets an API result code in "res".

NQSrewindsch() re-initializes the internal index to 0. When an error occurs, NQSrewindsch() returns a negative integer and sets an API result code in "res".

NQSclosesch() deletes a job server entry created by NQSopensch() and returns 0 when executed successfully. When an error occurs, NQSclosesch() returns a negative integer and sets an API result code in "res".

Notes

When you refer to a pointer pointed by a job server identifier returned by NQSreadsch() after NQSclosesch() was executed, the operation is not assured.

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_ENOENT]

Invalid entry identifier specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSattrsch(3)

6.4.3. **Operation of Scheduler Attributes**

Name

NQSattrsch -- Scheduler Attribute Operation Function

Format

```
#include <nqsv.h>

int NQSattrsch(nqs_schid *schid, nqs_alist ad, int op, nqs_res *res)
```

Function

Operates the attribute of a scheduler having the identifier "schid". NQSattrsch executes a specified operation "op" on attributes in the attribute list "ad". A value below can be specified for "op".

ATTROP_GET

Copies the attribute of the scheduler onto the attribute in the attribute list "ad".

Return value

When executed successfully, NQSattrsch() returns 0. When an error occurs, NQSattrsch() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority given to access the attribute.

[NQS_ENOENT]

Cannot find the specified attribute list.

[NQS_ENOSCH]

Cannot find the specified scheduler.

[NQS_EUNKNOWN]

Unknown attribute type specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSalist(3), NQSafree(3), NQSaadd(3), NQSaref(3)

6.5. Batch server related functions

6.5.1. Operate Batch Server Attributes

Name

NQSattrbsv -- Batch Server Attribute Operate Function

Format

```
#include <nqsv.h>

int NQSattrbsv(nqs_alist ad, int op, nqs_res *res)
```

Function

Operates the attribute of a batch server. NQSattrbsv executes an attribute operation "op" on an attribute in the attribute list "ad". Values below can be specified for "op".

ATTROP_GET

Copies a batch server attribute on an attribute "ad".

ATTROP_SET

Substitutes the batch server attribute by attribute "ad".

Return value

When executed successfully, NQSattrbsv() returns 0. When an error occurs, NQSattrbsv() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority given to access the attribute.

[NQS_ENOENT]

Cannot find the specified attribute list.

[NQS_ERANGE]

Attribute value outside a specified range.

[NQS_EUNKNOWN]

Unknown batch server attribute specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSalist(3), NQSafree(3), NQSaadd(3), NQSaref(3)

6.5.2. **Stop Batch Server**

Name

NQSshutbsv -- Stop Batch Server

Format

```
#include <nqsv.h>

int NQSshutbsv(nqs_res *res)
```

Function

Shuts down the batch server.

Notes

NQSshutbsv() requires an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQSshutbsv() returns 0. When an error occurs, NQSshutbsv() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to run the API.

Related items

NQSconnect(3)

6.5.3. Registration of Execution Hosts

Name

NQSattachhst, NQSdetachhst -- Managing the registration of execution hosts

Format

```
#include <nqsv.h>

int NQSattachhst(nqs_hid *hid, nqs_jsvid *jsvid, nqs_res *res )
int NQSdetachhst(nqs_hid *hid, nqs_jsvid *jsvid, int opt, nqs_res *res )
```

Function

NQSattachhst() registers execution host specified as "hid" with the job server ID as "jsvid" to the batch server. The registration can be executed while the job server is down. If "hid" or "jsvid" is already registered, NQSattachhst() returns error.

NQSdetachhst() removes registration of execution host specified by "hid" or "jsvid". Either "hid" or "jsvid" can have value and the other must be set to NULL.

When "ip.s_addr=0" in "hid" and "jsvno=-1" in "jsvid" are specified, all execution hosts are removed.

No execution host specified by "hid" or "jsvid" is registered, NQSdetachhst() returns error.

And the execution host has a job NQSdetachhst() also returns error.

opt value	Action
DETACH_FLAG_DEFAULT	When the execution host has a job, NQSdetachhst() returns error.

Notes

NQSattachhst() and NQSdetachhst() require an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQSattachhst() and NQSdetachhst() return 0. When an error occurs, NQSattachhst() and NQSdetachhst() return a negative integer and set an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to run the API.

Related items

NQSconnect(3)

6.6. Job server related functions

6.6.1. Operation of Job server Entry

Name

NQSopenjsv, NQSreadjsv, NQSrewindjsv, NQSclosejsv -- Operate Job Server Entry

Format

```
#include <nqsv.h>

nqs_entry NQSopenjsv(nqs_odesc *odesc, nqs_res *res)
nqs_jsvid *NQSreadjsv(nqs_entry entry, nqs_res *res)
int NQSrewindjsv(nqs_entry entry, nqs_res *res)
int NQSclosejsv(nqs_entry entry, nqs_res *res)
```

Function

NQSopenjsv() selects a job server related to an object "odesc" from job servers linked to the batch server. Creates a list (job server entry) having the job server identifier as its element, initializes the internal index to 0, and returns an entry identifier to identify it. When an error occurs, NQSopenjsv() returns a negative integer and sets an API result code in "res".
The table below lists objects that can be specified for "odesc".

odesc.obj_type value	obj member to be referred	Objects
NQS_OBJ_BSV	None	All of servers
NQS_OBJ_HST	odesc.obj.hid	All job servers on the specified execution host
NQS_OBJ_QUE	odesc.obj.qid	All job servers bound to a specified execution queue

NQSreadjsv() returns an entry pointed to by the internal index by the "nqs_jsvid" type pointer and increments the internal index by one. When the last entry comes, NQSreadjsv() returns a null and sets NQS_EALLOVER in the error number of "res". When an error occurs, NQSreadjsv() returns a null and sets an API result code in "res".

NQSrewindjsv() re-initializes the internal index to 0. When an error occurs, NQSrewindjsv() returns a negative integer and sets an API result code in "res".

NQSclosejsv() deletes a job server entry created by NQSopenjsv() and returns 0 when the processing is completed. When an error occurs, NQSclosejsv() returns a negative integer and sets an API result code in "res".

Notes

When you refer to a pointer that points to a job identifier returned by NQSreadjsv() after NQSclosejsv() was executed, the operation is not assured.

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_ENOENT]

Invalid entry identifier specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSattrjsv(3)

6.6.2. **Operation of Job server Attributes**

Name

NQSattrjsv -- Job Server Attribute Operation Function

Format

```
#include <nqsv.h>

int NQSattrjsv(nqs_jsvid *jsvid, nqs_alist ad, int op, nqs_res *res)
```

Function

Operates the attribute of a job server having the identifier "jsvid". NQSattrjsv executes a specified operation "op" on attributes in the attribute list "ad". A value below can be specified for "op".

ATTROP_GET

Copies the attribute of the job server onto the attribute "ad".

Return value

When executed successfully, NQSattrjsv() returns 0. When an error occurs, NQSattrjsv() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority given to access the attribute.

[NQS_ENOENT]

No specified attribute list.

[NQS_ENOJSV]

No specified job server.

[NQS_EUNKNOWN]

Unknown job server attribute specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSopenjsv(3), NQSreadjsv(3), NQSclosejsv(3), NQSalist(3), NQSafree(3), NQSaadd(3), NQSaref(3)

6.6.3. **Control Job Server**

Name

NQSctljsv -- Job Server Control Function

Format

#include <nqsv.h>

int NQSctljsv(nqs_jsvid *jsvid, int cmd, void *args, nqs_res *res)

Function

Executes a control command specified for "cmd" to a job server having the identifier "jsvid". If "cmd" needs an argument, set a pointer of the argument to "args".

The list below shows a control command that can be specified for "cmd" and a type for "args" and an explain of the function.

"cmd": JSVCTL_LINKDOWN

"args": NULL (not necessary)

Disconnect the TCP-connect between a job server indicated by jsvid and a batch server forcedly. If it has been already disconnected, NQSctljsv() ends in normal.

"cmd": JSVCTL_BOOTUP

"args": char *host_name

jsvid	args	Action
jsvid.jsvno=-1	host name	Start the job server whose host name is "host_name" with the registered job server ID.
jsvid.jsvno>=0	host name	Start the job server whose host name is "host_name" with the specified job server ID.
jsvid.jsvno=-1	NULL	All job server registered in the batch server is started.

"cmd": JSVCTL_SHUTDOWN

"args": int *force

jsvid	args	Action
jsvid.jsvno>=0	force = 0	Shutdown the job server whose ID is "jsvid". But shutdown of the job server which has jobs is rejected.
jsvid.jsvno>=0	force != 0	Shutdown the job server whose ID is "jsvid". Shutdown is done regardless the job server has batch

		jobs or not.

"cmd": JSVCTL_SHUTJSV

"args": char *host_name

jsvid	args	Action
jsvid.jsvno>=0	NULL	Shutdown the job server whose ID is "jsvid". But shutdown of the job server which has jobs is rejected.
jsvid.jsvno=-1	host name	Shutdown the job server whose host name is "host_name". But shutdown of the job server which has jobs is rejected.
jsvid.jsvno=-1	NULL	Shutdown all job server registered in the batch server. But shutdown of the job server which has jobs is rejected.

"cmd": JSVCTL_FSHUTJSV

"args": char *host_name

jsvid	args	Action
jsvid.jsvno>=0	NULL	Shutdown the job server whose ID is "jsvid". Shutdown is done regardless the job server has batch jobs or not.
jsvid.jsvno=-1	host name	Shutdown the job server whose host name is "host_name". Shutdown is done regardless the job server has batch jobs or not.
jsvid.jsvno=-1	NULL	Shutdown all job server registered in the batch server. Shutdown is done regardless the job server has batch jobs or not.

"cmd": JSVCTL_BOOTNGRP

"args": char *node_group

jsvid	args	Action
jsvid.jsvno=-1	node group	Start the job server on the hosts which belong to node group specified by "node_group".

"cmd": JSVCTL_SHUTNGRP

"args": char *node_group

jsvid	args	Action
jsvid.jsvno=-1	node group	Shutdown the job server on the hosts which belong to node group specified by "node_group". But shutdown of the job server which has jobs is rejected.

"cmd": JSVCTL_FSHUTNGRP

"args": char *node_group

jsvid	args	Action
jsvid.jsvno=-1	node group	Shutdown the job server on the hosts which belong to node group specified by "node_group". Shutdown is done regardless the job server has batch jobs or not.

Notes

The API authority of PRIV_MGR or higher is required to execute NQSctljsv().

To execute JSVCTL_BOOTUP, the launcher daemon needs to reside on the execution host.

Return value

When executed successfully, NQSctljsv() returns 0. When an error occurs, NQSctljsv() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to run the API function.

[NQS_ERANGE]

The JSV ID specified is outside a range.

[NQS_ENOJSV]

No specified job server. (In case except cmd is JSVCTL_BOOTUP.)

[NQS_EALREADY]

The job server specified has already started up.

(In case cmd is JSVCTL_BOOTUP.)

[NQS_EREFUSE]

Rejected because the job server specified had the batch jobs.

(In case cmd is JSVCTL_SHUTDOWN.)

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.7. Execution host related functions

6.7.1. Operation of Execution host Entry

Name

NQSopenhst, NQSreadhst, NQSrewindhst, NQSclosehst -- Operate Host Entry

Format

```
#include <nqsv.h>

nqs_entry NQSopenhst(nqs_odesc *odesc, nqs_res *res)
nqs_hid *NQSreadhst(nqs_entry entry, nqs_res *res)
int NQSrewindhst(nqs_entry entry, nqs_res *res)
int NQSclosehst(nqs_entry entry, nqs_res *res)
```

Function

NQSopenhst() selects an execution host related to a specified object "odesc" from execution hosts (having one or more job servers linked to the job server) that is recognized by the batch server, creates a list (execution host entry) having the host identifier as the element, its index to 0, and returns an entry identifier to identify it. When an error occurs NQSopenhst() returns a negative integer and sets an API result code in "res".
The table below lists objects that can be specified for "odesc".

odesc.obj_type value	obj member to be referred	Objects
NQS_OBJ_BSV	None	All execution hosts

NQSreadhst() returns an entry pointed to by the internal index by the "nqs_hid" type pointer, and increments the internal index by one. When the last entry comes, NQSreadhst() returns a null and sets NQS_EALLOVER in the error number of "res". When an error occurs, NQSreadhst() returns a null and sets an API result code in "res".

NQSrewindhst() re-initializes the internal index to 0. When an error occurs, NQSrewindhst() returns a negative integer and sets an API result code in "res".

NQSclosehst() deletes a job entry created by NQSopenhst(). And returns 0 when executed successfully. When an error occurs, NQSclosehst() returns a negative integer and sets an API result code in "res".

Notes

When you refer to a pointer that points to an execution host identifier returned by

NQSreadhst() after NQSclosehst() was execute, the operation is not assured.

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_ENOENT]

Invalid entry identifier specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSattrhst(3)

6.7.2. **Operation of Execution host Attributes**

Name

NQSattrhst -- Execution Host Attribute Operation Function

Format

```
#include <nqsv.h>

int NQSattrhst(nqs_hid *hid, nqs_alist ad, int op, nqs_res *res)
```

Function

Operates the attribute of an execution host having a host identifier "hid". NQSattrhst() executes a specified operation "op" on attributes in the attribute list "ad". A value below can be specified for "op".

ATTROP_GET

Copies the attribute of the execution host onto the attribute "ad".

Return value

When executed successfully, NQSattrhst() returns 0. When an error occurs, NQSattrhst() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority given to access the attribute.

[NQS_ENOENT]

Cannot find the specified attribute list.

[NQS_ENOHST]

Cannot find the specified execution host.

[NQS_EUNKNOWN]

Unknown attribute type specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSopenhst(3), NQSreadhst(3), NQSclosehst(3), NQSalist(3), NQSafree(3), NQSaadd(3), NQSaref(3)

6.8. Queue related functions

6.8.1. Operation of Queue Entry

Name

NQSopenque, NQSreadque, NQSrewindque, NQScloseque -- Operate Queue Entry

Format

```
#include <nqsv.h>

nqs_entry NQSopenque(nqs_odesc *odesc, nqs_res *res)
nqs_qid *NQSreadque(nqs_entry entry, nqs_res *res)
int NQSrewindque(nqs_entry entry, nqs_res *res)
int NQScloseque(nqs_entry entry, nqs_res *res)
```

Function

NQSopenque() selects a queue related to a specified object "odesc" from queues (batch queue, interactive queue or routing queue) on the batch server, creates a list (queue entry) having the queue identifier as the element, initializes the internal index to 0, and returns an entry identifier to identify it. When an error occurs, NQSopenque() returns a negative integer and sets an API result code in "res".

The table below lists objects that can be specified for "odesc".

odesc.obj_type value	obj member to be referred	Objects
NQS_OBJ_BSV	None	All queues in the system
NQS_OBJ_SCH	odesc.obj.schid	All execution queues bound to the specified scheduler

NQSreadque() returns an entry pointed to by the internal index by the "nqs_qid" type pointer, and increments the internal index by one. When the last entry comes, NQSreadque() returns a null and sets NQS_EALLOVER in the error number of "res". When an error occurs, NQSreadque() returns a null and sets an API result code in "res".

NQSrewindque() re-initializes the internal index to 0. When an error occurs, NQSrewindque() returns a negative integer and sets an API result code in "res".

NQScloseque() deletes a queue entry created by NQSopenque(). When executed successfully, NQScloseque() returns 0. When an error occurs, NQScloseque() returns a negative integer and sets an API result code in "res".

Notes

When you referred to a pointer pointed to the queue identifier returned by NQSreadque() after NQScloseque() was executed, the operation is not assured.

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_ENOENT]

Invalid entry identifier specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSattrque(3)

6.8.2. **Operation of Queue Attributes**

Name

NQSattrque -- Queue Attribute Operation Function

Format

```
#include <nqsv.h>

int NQSattrque(nqs_qid *qid, nqs_alist ad, int op, nqs_res *res)
```

Function

Operates the attribute of an execution queue or routing queue having the identifier "qid". NQSattrque() executes a specified operation "op" on attributes in the attribute list "ad". A value below can be specified for "op".

ATTROP_GET

Copies the queue attribute onto the attribute in the attribute list "ad".

ATTROP_SET

Replaces the queue attribute by the attribute in the attribute list "ad".

ATTROP_ADD

Adds the attribute in the attribute list "ad" to the queue attribute.

ATTROP_DEL

Deletes the attribute in the attribute list "ad" from the queue attribute.

You can specify ATTROP_ADD and ATTROP_DEL only for attributes that can chain attribute values.

Return value

When executed successfully, NQSattrque() returns 0. When an error occurs, NQSattrque() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority given to access the attribute.

[NQS_ENOENT]

Cannot find the specified attribute list.

Cannot delete the destination queue as it has not been registered.

[NQS_ENOQUE]

Cannot find the specified queue.

[NQS_ELOOP]

Destination queue specified in loop.

[NQS_EEXIST]

Cannot add the queue as the destination queue is already cataloged.

[NQS_EFLOOD]

Too many destination queues (more than the maximum number of queues registered).

[NQS_EOPE]

ATTROP_ADD was executed on an attribute that cannot chain attribute values.

ATTROP_DEL was executed on an attribute that cannot chain attribute values.

[NQS_ENAMPDB]

The specified host has not been registered in the NMAP database.

[NQS_EUNKNOWN]

Unknown attribute type specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified

Related items

NQSconnect(3), NQSopenque(3), NQSreadque(3), NQScloseque(3), NQSalist(3), NQSafree(3), NQSaadd(3), NQSaref(3)

6.8.3. **Create Queue**

Name

NQScreque -- Create Queue

Format

```
#include <nqsv.h>

int NQScreque(nqs_qid *qid, nqs_alist ad, nqs_res *res)
```

Function

Creates a new execution queue or routing queue having a queue type and a queue name specified by a queue identifier "qid". The attribute in the attribute list "ad" is set in the created queue.

Notes

NQScreque() requires an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQScreque() returns 0. When an error occurs, NQScreque() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENOENT]

Cannot find the specified attribute list.

[NQS_EEXIST]

Existing queue specified.

[NQS_EUNKNOWN]

Unknown attribute type specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSqttrque(3), NQSalist(3), NQSafree(3), NQSaadd(3), NQSaref(3)

6.8.4. **Delete Queue**

Name

NQSdelque -- Delete Queue

Format

```
#include <nqsv.h>

int NQSdelque( nqs_qid *qid, nqs_res *res )
```

Function

Deletes an execution queue or routing queue having a queue type and a queue name specified by the queue identifier "qid". Only disabled queues without a request can be deleted.

Notes

NQSdelque() requires an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQSdelque() returns 0. When an error occurs, NQSdelque() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENOQUE]

Cannot find the specified queue.

[NQS_EENABLE]

Queue enabled status.

[NQS_EHASREQ]

Request existing in the queue.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSqttrque(3), NQSalist(3), NQSafree(3), NQSaadd(3), NQSaref(3)

6.8.5. **Connect Scheduler**

Name

NQSbindsch -- Connect Execution Queue to Batch Scheduler

Format

```
#include <nqsv.h>

int NQSbindsch(nqs_qid *qid, nqs_schid *schid, nqs_res *res)
```

Function

NQSbindsch() connects (or binds) a batch scheduler having the identifier "schid" to a queue having the identifier "qid".

The queue types that can be specified to "qid" are batch queue and interactive queue.

Notes

The API authority of PRIV_MGR or higher is required to execute NQSbindsch().

Return value

When executed successfully, NQSbindsch() returns 0. When an error occurs, NQSbindsch() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to a batch server.

[NQS_ECONFAIL]

Cannot connect to the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority given to execute the API function.

[NQS_ENOQUE]

Cannot find the specified queue.

[NQS_EWRNGTYP]

The queue type of specified queue is wrong.

[NQS_ENOSCH]

The specified scheduler is not linked to the batch server.

[NQS_EBINDSCH]

The specified queue is already bound to a scheduler.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSunbindsch(3)

6.8.6. **Disconnect Scheduler**

Name

NQSunbindsch -- Disconnect Execution Queue from Batch Scheduler

Format

```
#include <nqsv.h>

int NQSunbindsch(nqs_qid *qid, nqs_schid *schid, nqs_res *res)
```

Function

NQSunbindsch() breaks a connection between a specified batch scheduler "schid" and a specified queue "qid".

The queue types that can be specified to "qid" are batch queue and interactive queue.

Notes

NQSunbindsch() requires an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQSunbindsch() returns 0. When an error occurs, NQSunbindsch() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENOQUE]

Cannot find the specified queue.

[NQS_EWRNGTYP]

Unacceptable the queue type of specified queue.

[NQS_ENOSCH]

The specified scheduler is not linked to the batch server.

[NQS_EBINDSCH]

The specified queue is not bound to the scheduler.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSbindsch(3)

6.8.7. **Connect Job Server**

Name

NQSbindjsv -- Connect Execution Queue to Job Server

Format

```
#include <nqsv.h>

int NQSbindjsv(nqs_qid *qid, nqs_jsvid *jsvid, nqs_res *res)
```

Function

NQSbindjsv() connects (or binds) a job server having the server identifier "jsvid" to a queue having the identifier "qid".

The queue types that can be specified to "qid" are batch queue and interactive queue.

NQSbindjsv() cannot bind a job server that is already connected to the queue.

Notes

The API authority of PRIV_MGR or higher is required to execute NQSbindjsv().

Return value

When executed successfully, NQSbindjsv() returns 0. When an error occurs, NQSbindjsv() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to a batch server.

[NQS_ECONFAIL]

Cannot connect to the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority given to execute the API function.

[NQS_ENOQUE]

Cannot find the specified queue.

[NQS_EWRNGTYP]

Unacceptable the queue type of specified queue.

[NQS_ENOJSV]

The specified job server is not linked to the batch server.

[NQS_EBINDJSV]

The specified job server is already bound to a queue.

[NQS_EQUEDOM]

The specified job server is waiting a job submitted to other queue.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSunbindjsv(3)

6.8.8. **Disconnect Job Server**

Name

NQSunbindjsv -- Disconnect Execution Queue from Job Server

Format

```
#include <nqsv.h>

int NQSunbindjsv(nqs_qid *qid, nqs_jsvid *jsvid, nqs_res *res)
```

Function

NQSunbindjsv() breaks a connection between a specified job server "jsvid" and a specified queue "qid".

The queue types that can be specified to "qid" are batch queue and interactive queue.

Notes

NQSunbindjsv() requires an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQSunbindjsv() returns 0. When an error occurs, NQSunbindjsv() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to run the API function.

[NQS_ENOQUE]

Cannot find the specified queue.

[NQS_EWRNGTYP]

Unacceptable the queue type of specified queue.

[NQS_ENOJSV]

The specified job server is not linked to the batch server.

[NQS_EBINDJSV]

The specified job server is not bound to the queue.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSbindjsv(3)

6.9. Request related functions

6.9.1. Operation of Request Entry

Name

NQSopenreq, NQSopenbreq, NQSopenireq, NQSreadreq, NQSrewindreq, NQSclosereq
-- Operate Request Entry

NQSopenprm, NQSreadprm, NQSrewindprm, NQScloseprm
-- Operate Parametric request Entry

Format

```
#include <nqsv.h>

nqs_entry NQSopenreq(nqs_odesc *odesc, nqs_res *res)
nqs_entry NQSopenbreq(nqs_odesc *odesc, nqs_res *res)
nqs_entry NQSopenireq(nqs_odesc *odesc, nqs_res *res)
nqs_rid *NQSreadreq(nqs_entry entry, nqs_res *res)
int NQSrewindreq(nqs_entry entry, nqs_res *res)
int NQSclosereq(nqs_entry entry, nqs_res *res)

nqs_entry NQSopenprm(nqs_odesc *odesc, nqs_res *res)
nqs_rid *NQSreadprm(nqs_entry entry, nqs_res *res)
int NQSrewindprm(nqs_entry entry, nqs_res *res)
int NQScloseprm(nqs_entry entry, nqs_res *res)
```

Function

NQSopenreq() selects a request (sub-request is selected for the parametric request) related to a specified object "odesc" from requests on the batch server, creates a list (request entry) having the request identifier as the element, initializes the internal index to 0, and returns an entry identifier to identify it. When an error occurs, NQSopenreq() returns a negative integer and sets an API result code in "res".

The table below lists objects that can be specified for "odesc".

odesc.obj_type value	obj member to be referred	Objects
NQS_OBJ_BSV	None	All requests
NQS_OBJ_QUE	odesc.obj.qid	All requests in the specified queue
NQS_OBJ_PRM	odesc.obj.rid	All sub-request in the specified parametric request.

NQSopenbreq() and NQSopenireq can open the request entry for batch request and interactive request separately.

NQSreadreq() returns an entry pointed to by the internal pointer by the "nqs_rid" type pointer and increments the internal index by one. When the last attribute entry comes, NQSreadreq() returns a null and sets NQS_EALLOVER in the error number of "res". When an error occurs, NQSreadreq() returns a null and sets an API result code in "res".

NQSrewindreq() re-initializes the internal index to 0. When an error occurs, NQSrewindreq() returns a negative integer and sets an API result code in "res".

NQSclosereq() deletes a request entry created by NQSopenreq() and returns 0 when executed successfully. When an error occurs, NQSclosereq() returns a negative integer and sets an API result code in "res".

For parametric request, use NQSopenprm(), NQSreadprm(), NQSrewindprm() and NQScloseprm(), instead.

Notes

When you refer to a pointer pointed to by the request identifier returned by NQSreadreq() after NQSclosereq() was executed, the operation is not assured.

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot send the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_ENOENT]

Invalid entry identifier specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSattrreq(3)

6.9.2. Operation of Request Attributes

Name

NQSattrreq -- Request Attribute Operation Function

Format

```
#include <nqsv.h>

int NQSattrreq(nqs_rid *rid, nqs_alist ad, int op, nqs_res *res)
```

Function

Operates the attribute of a request having the identifier "rid". NQSattrreq executes a specified operation "op" on attributes in the attribute list "ad". A value below can be specified for "op".

ATTROP_GET

Copies the attribute of the request onto the attribute in the attribute list "ad".

ATTROP_SET

Replaces the attribute of the request by the attribute in the attribute list "ad".

ATTROP_ADD

Adds attribute values in the attribute list "ad" to the attribute of the request.

ATTROP_DEL

Deletes attribute values in the attribute list "ad" from the attribute of the request.

You can specify ATTROP_ADD and ATTROP_DEL only for attributes that can chain attribute values.
You can not specify ATTROP_SET for sub-request of parametric request specified by "rid" (rid.subreq_no >= 0).

Return value

When executed successfully, NQSattrreq() returns 0. When an error occurs, NQSattrreq() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority given to access the attribute.

[NQS_ENOENT]

Cannot find the specified attribute list.

[NQS_ENOREQ]

Cannot find the specified request.

[NQS_EUNKNOWN]

Unknown attribute type specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSopenreq(3), NQSreadreq(3), NQSclosereq(3), NQSalist(3), NQSafree(3), NQSaadd(3), NQSaref(3)

6.9.3. **Create Request**

Name

NQScrereq, NQSleadreq -- Create Batch Request and Set it to Queue

Format

#include <nqsv.h>

int NQScrereq(nqs_rid *rid, nqs_qid *qid, nqs_alist ad, char *script, nqs_res *res)
int NQSleadreq(nqs_rid *rid,nqs_res *res)

Function

NQScrereq() creates a new request having a specified file "script" as a job script and sets it in a batch queue having a queue identifier "qid". When the attribute list identifier "ad" is an integer of 0 or more, all attributes in the attribute list are set as the initial attribute of the request.

NQSleadreq() notifies the batch server when all connected requests were submitted. Request ID of the lead request (request submitted first in request connection) is set in "rid".

Attributes that can have the initial request attribute are as follows:

Attribute name	Attribute type	Type	Scope		
			Request	Job	Process
Job Topology	ATTR_JTPLGY	int	o		
Re-run Attribute	ATTR_RERUNABL	int	o		
Checkpoint Attribute	ATTR_CHKPNTABL	int	o		
Migration Attribute	ATTR_MIGRATABL	int	o		
Hold Attribute	ATTR_HOLDABL	int	o		
Hold Type	ATTR_HOLDTYPE	int	o		
Account Code	ATTR_ACCTCODE	char *	o		
Priority	ATTR_PRIORITY	nqs_range	o		
Request Name	ATTR_REQNAME	char *	o		
Standard Output Path Name	ATTR_STDOUT	nqs_pdesc	o		
Standard Error Output Path Name	ATTR_STDERR	nqs_pdesc	o		
Request Log Output Path Name	ATTR_STDLOG	nqs_pdesc	o		

Request Log Output Level	ATTR_LOGLEVEL	nqs_range	o		
Staging File Information	ATTR_STGFILE	nqs_stgfile	o		
Shell Name	ATTR_SHELLPATH	char *	o		
Mail Option	ATTR_MAILOPTS	int	o		
Mail Address	ATTR_MAILADDR	nqs_mdesc	o		
Job Execution Environment Condition	ATTR_JOBCOND	nqs_jcond	o		
Request Group	ATTR_REQGRP	nqs_rgrp	o		
Execution Time	ATTR_EXETIME	time_t	o		
Job Environment Variable	ATTR_ENVIRON	nqs_keyval			o
Migration file	ATTR_MIGFILE	nqs_migfile	o		
User Custom Attribute	ATTR_USERATTR	nqs_keyval	o		
Restart file Path	ATTR_RSTFDIR	char *	o		
Reservation ID	ATTR_ADVRSVID	int	o		
Max. Elapsed Time	ATTR_ELPSTIM	nqs_rlim	o		
Max.CPU Time	ATTR_CPUTIM	nqs_rlim		o	o
Max. number of CPU	ATTR_CPUNUM	nqs_rnum		o	
Max. number of Files Opened	ATTR_FILENUM	nqs_rnum			o
Max. Memory Size	ATTR_MEMSZ	nqs_rlim		o	o
Max. Data Size	ATTR_DATASZ	nqs_rlim			o
Max. Stack Size	ATTR_STACKSZ	nqs_rlim			o
Max. Core File Size	ATTR_CORESZ	nqs_rlim			o
Max. File Size	ATTR_FILESZ	nqs_rlim			o
Max. virtual memory size	ATTR_VMEMSZ	nqs_rlim		o	o
Max. number of GPU	ATTR_GPUNUM	nqs_rnum		o	
Range for number of VE node	ATTR_VENUM	nqs_rrange		o	

Notes

It is necessary to set the ATTR_REQGRP attribute in all connected requests (includes lead request) as requesting an initial attribute. However, in the ATTR_REQGRP attribute of the lead request, set only an appropriate value (request group number) in grpno, and always set -1 in seqno and mid. (When NQSleadreq(3) is executed, the batch server set lead request ID's seqno and mid)

Return value

When an error occurs, NQScrereq() sets the request ID of a newly created request in "rid" and returns 0 as the return value. When an error occurs, NQScreque() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_ESYSCAL]

Error in the system call.

[NQS_EWRNGSZ]

Cannot transfer the job script.

[NQS_ENOENT]

Cannot find the specified attribute list "ad".

[NQS_EACCTAUTH]

Un-cataloged request owner account.

[NQS_EACCESSDEN]

Inhibited to access the batch server.

[NQS_EFATAL]

Cannot take a sequence number.

[NQS_ETOOLONG]

Too long result file name.

[NQS_EUNKNOWN]

Unsupported initial attribute in the attribute list "ad".

[NQS_EWRNGTYP]

Submit a connected request to wrong queue.

[NQS_EUNSUPPORT]

Unsupported job type specified.

[NQS_EBADVAL]

The request attribute with an illegal value exists.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSalist(3), NQSafree(3), NQSaadd(3), NQSaref(3)

6.9.4. **Create job and Start stage-in**

Name

NQSstgreq -- Create jobs and start stage-in sequence

Format

```
#include <nqsv.h>

int NQSstgreq(nqs_rid *rid, int num, nqs_jsvid jsvid[], nqs_res *res)
```

Function

NQSstgreq() requires the batch server to create the batch jobs for specified batch request "rid", and requires to start the stage-in sequence for batch request. This function is valid only for QUEUED requests in the execution queue.
The number of batch jobs newly generated is specified for "num". "jsvid" array specifies a job server that runs the job. The subscript of "jsvid" array is corresponding to a job number. For example, "jsvid[3].jsvno=7" indicates that job #3 is allocated to job server #7. A "num" value (number of jobs) can be 1 to (NQS_MAX_JOBNO + 1). If STGOPT_LEAVE is specified to jsvno of "jsvid" array, the execution of this element is not done, and status of the batch job doesn't change.

When a batch job corresponds to job number specified already exists, the old batch job is abandoned and this batch job will be created newly on the job server specified for jsvid.

The execution queue containing the request to be run must be bound to a process (usually a scheduler) that called NQSstgreq().

You cannot specify parametric request (rid.subreq_no=-2) to "rid".

Notes

NQSstgreq() requires an API authority of PRIV_SCH or higher to run.
The stage-in files are transmitted from the client host to the batch server host temporarily(/var/opt/nec/nqsv/bsv/private/root/input), and then they are transmitted to each execution hosts on which the job will be executed. NQSstgreq() returns the NQS_EDISCONTI error if the batch job of job number 0(master job) is not included or the job number is not consecutive when all the batch jobs are created.

When NQSstgreq() is successfully called, batch requests will be STAGING, then stage-in procedure will start. All batch jobs are cancelled if an error happens in STAGING, and requests will be QUEUED.

Return value

When executed successfully, NQSstgreq() returns 0. When an error occurs, NQSstgreq() returns a negative integer and sets an API result code in "res". If NQSstgreq() returns an error, the status of each batch job is returned to the status immediately before NQSstgreq() is called.

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to run the API function.

[NQS_EBINDSCH]

The calling process is not bound to the queue that contains the request.

[NQS_EOUTRNG]

The specified job server number is outside the valid range.

The specified job number is outside the valid range.

[NQS_ENOREQ]

Cannot find the specified request.

[NQS_EWRNGSTS]

Unacceptable request status.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

[NQS_EDISCONTI]

The job numbers of the batch job are not consecutive.

Related items

NQSconnect(3), NQSbindsch(3), NQSrunreq(3)

6.9.5. **Start Request**

Name

NQSrunreq -- Run Request

Format

```
#include <nqsv.h>

int NQSrunreq(nqs_rid *rid, nqs_res *res)
```

Function

NQSrunreq() asks the batch server to run a specified request "rid". It is necessary already to create the job with NQSstgreq(3) as for the request. This function is valid only for QUEUED requests in the execution queue except requests that have been QUEUED from HOLDING (requests having a restart file on the job server).
You need to bind the execution queue the process where NQSrunreq is called like the scheduler. And also need to bind the execution queue all job servers which manage the object job.

You cannot specify parametric request (rid.subreq_no=-2) to "rid".

Notes

NQSrunreq() requires an API authority of PRIV_SCH or higher to run.

Return value

When executed successfully, NQSrunreq() returns 0. When an error occurs, NQSrunreq() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to run the API function.

[NQS_EBINDSCH]

The calling process is not bound to the queue containing the request.

[NQS_EBINDJSV]

The job server is not bound to a queue containing the request.

[NQS_ENOREQ]

Cannot find the specified request.

[NQS_ENOJOB]

There is no job.

[NQS_EWRNGSTS]

Unacceptable request status.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSbindjsv(3), NQSbindsch(3), NQSstgreq(3)

6.9.6. **Delete Request**

Name

NQSdelreq -- Delete Request

Format

```
#include <nqsv.h>

int NQSdelreq(nqs_rid *rid, int grace, nqs_res *res)
```

Function

NQSdelreq() deletes a specified request "rid". When the execution host has a job in execution, NQSdelreq() forcibly terminates the job by a signal. In this case, NQSdelreq() first sends SIGTERM to the job, waits a specified time period "grace" (in seconds) and sends SIGKILL. When "grace" is 0, NQSdelreq() immediately sends SIGKILL only.

Notes

NQSdelreq() requires an API authority of PRIV_MGR or higher, API authority PRIV_GMGR for the group of the request or the owner of the request.

Return value

When executed successfully, NQSdelreq() returns 0. When an error occurs, NQSdelreq() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to delete the request.

[NQS_ENOREQ]

Cannot find the specified request.

[NQS_EREFUSE]

The request is temporarily undeletable.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSrqureq(3)

6.9.7. **Send Signal to Request**

Name

NQSsigreq -- Send Signal to Request

Format

```
#include <nqsv.h>

int NQSsigreq(nqs_rid *rid, char *sig, nqs_res *res)
```

Function

NQSsigreq() sends a specified signal "sig" to all jobs having a specified request "rid" as the parent. NQSsigreq() is valid for jobs whose parent request is RUNNING or SUSPENDED. For other jobs, NQSsigreq() returns an error.
"sig" specifies a signal name having a prefix "SIG" which is one of signal names in the Signal List below. If the specified signal is not supported by the operating system of the execution host, NQSsigreq() does not send the signal and simply ignores it.

When SIGSTOP is specified for a RUNNING request, the request changes its status to SUSPENDING and undergoes suspending process.

When SIGCONT is specified for a SUSPENDED request, the request changes its status to RESUMING and undergoes resuming process.

Signal	Linux
SIGHUP	o
SIGINT	o
SIGQUIT	o
SIGILL	o
SIGTRAP	o
SIGABRT(SIGIOT)	o
SIGBUS	o
SIGFPE	o
SIGKILL	o
SIGSEGV	o
SIGUSR1	o
SIGUSR2	o
SIGPIPE	o
SIGALRM	o
SIGTERM	o
SIGCHLD(SIGCLD)	o
SIGCONT	o
SIGSTOP	o

SIGTSTP	o
SIGTTIN	o
SIGTTOU	o
SIGURG	o
SIGXCPU	o
SIGXFSZ	o
SIGWINCH	o
SIGIO	o
SIGPOLL	o
SIGPWR	o
SIGVTALRM	o
SIGPROF	o
SIGSTKFLT	o
SIGUNUSED	o

Notes

NQSsigreq() requires an API authority of PRIV_MGR or higher, API authority PRIV_GMGR for the group of the request or the owner of the request.

Return value

When executed successfully, NQSsigreq() returns 0. When an error occurs, NQSsigreq() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to send a signal to the request.

[NQS_ENOREQ]

Cannot find the specified request.

[NQS_EWRNGSTS]

The request is not RUNNING or SUSPENDED.

[NQS_ESTALLED]

The request is being stalled.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSsigjob(3)

6.9.8. **Hold Request**

Name

NQShldreq -- Hold Request

Format

```
#include <nqsv.h>

int NQShldreq(nqs_rid *rid, nqs_res *res)
```

Function

NQShldreq() starts holding a request "rid". A request to be held must be QUEUED, WAITING, or RUNNING. When hold prohibition is set to the request, NQShldreq() becomes an error.

When holding a RUNNING request, NQShldreq() automatically gets an exit type checkpoint for each job and creates a restart file on the execution host.

When a request holding succeeds, the authority of the client which carried out holding is set in the hold type attribute (ATTR_HOLDTYPE).

Notes

NQShldreq() requires an API authority of PRIV_MGR or higher, API authority PRIV_GMGR for the group of the request or the owner of the request.

Return value

When executed successfully, NQShldreq() returns 0. When an error occurs, NQShldreq() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to hold the request.

[NQS_ENOREQ]

Cannot find the specified request.

[NQS_EWRNGSTS]

The request is not QUEUED, WAITING, or RUNNING.

[NQS_EDISABLE]

The request is set to DISABLED (Inhibited to Be Held).

[NQS_ESTALLED]

The request is stalled.

[NQS_ESYSCAL]

Error in the system call.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSrlsreq(3), NQSrstreq(3)

6.9.9. Release Request

Name

NQSrlsreq -- Release Request

Format

```
#include <nqsv.h>

int NQSrlsreq(nqs_rid *rid, nqs_res *res)
```

Function

NQSrlsreq() starts to release a specified request "rid". The request to be released must be HELD.

Notes

NQSrlsreq() requires a HOLD type attribute (that stores the authority for HOLD process) set in the request or a higher API authority to run.

Return value

When executed successfully, NQSrlsreq() returns 0. When an error occurs, NQSrlsreq() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to release the request.

[NQS_ENOREQ]

Cannot find the specified request.

[NQS_EWRNGSTS]

The request is not HELD.

[NQS_ESYSCAL]

Error in the execution of a system call.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQShldreq(3), NQSrstreq(3)

6.9.10. **Restart Request**

Name

NQSrstreq -- Restart Request

Format

```
#include <nqsv.h>

int NQSrstreq(nqs_rid *rid, nqs_res *res)
```

Function

NQSrstreq() restarts a specified request "rid". Only QUEUED requests can be started by NQSrstreq(). All jobs to be restarted must exist as restart files on the execution host.

The execution queue containing the request to be restarted must be bound to a process (usually a scheduler) that called NQSrstreq() and the job server managing the jobs to be restarted must be bound to this queue.

Notes

NQSrstreq() requires an API authority of PRIV_SCH or higher to run.

Return value

When executed successfully, NQSrstreq() returns 0. When an error occurs, NQSrstreq() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to restart the request.

[NQS_ENOREQ]

Cannot find the specified request.

[NQS_EWRNGSTS]

The request is not QUEUED.

[NQS_EBINDSCH]

The calling process is not bound to the queue that contains the request.

[NQS_EBINDJSV]

The job server managing the jobs to be restarted is not bound to the queue.

[NQS_ENOJOB]

No job having the specified request as the parent.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQShldreq(3), NQSrlsreq(3), NQSbindjsv(3), NQSbindsch(3)

6.9.11. **Re-run Request**

Name

NQSrqureq -- Stop and Re-queue Request

Format

```
#include <nqsv.h>

int NQSrqureq(nqs_rid *rid, int grace, nqs_res *res)
```

Function

NQSrqureq() stops the processing of specified request "rid" and backs the request to the QUEUED status with no batch jobs. (This is called re-queuing.) NQSrqureq() first sends SIGTERM to jobs executed on the execution host, waits for a specified time period "grace" (in seconds) and then sends SIGKILL. When "grace" is 0 or below, SIGKILL is immediately sent to jobs on the execution host.

NQSrqureq() is valid for requests that has batch jobs regardless of the status. When given to a request with no batch jobs, NQSrqureq() returns an error. Similarly, when given to a request that is inhibited to re-run, NQSrqureq() returns an error.

Notes

NQSrqureq() requires an API authority of PRIV_OPE or higher, API authority PRIV_GMGR for the group of the request or the owner of the request.

Return value

When executed successfully, NQSrqureq() returns 0. When an error occurs, NQSrqureq() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to run the API function.

[NQS_ENOREQ]

Cannot find specified request.

[NQS_EWRNGSTS]

The request is not RUNNING.

[NQS_EDISABLE]

The request is inhibited to re-run.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSdelreq(3)

6.9.12. **Move Request Between Queues**

Name

NQSmovreq -- Move Request Between Queues

Format

```
#include <nqsv.h>

int NQSmovreq(nqs_rid *rid, nqs_qid *qid, nqs_res *res)
```

Function

NQSmovreq() moves a request "rid" to a queue "qid" (batch queue, interactive queue or routing queue). Only QUEUED, WAITING, and HELD requests having no job (having no job restart file on the execution host) can be moved. The request after movement has the same status as that before movement.

You cannot specify parametric request (rid.subreq_no=-2) to "rid".

Notes

NQSmovreq() requires an API authority of PRIV_OPE or higher, API authority PRIV_GMGR for the group of the request or the owner of the request.
Parametric request cannot be specified to "rid".

Return value

When executed successfully, NQSmovreq() returns 0. When an error occurs, NQSmovreq() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to move the request.

[NQS_ENOREQ]

Cannot find the specified request.

[NQS_ENOQUE]

Cannot find the specified queue.

[NQS_EWRNGSTS]

The request is not QUEUED, WAITING, or HELD.

[NQS_EWRNGTYP]

Unacceptable the queue type of specified queue.

[NQS_EHASJOB]

The specified request has a job.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.10. Job related functions

6.10.1. Operation of Job Entry

Name

NQSopenjob, NQSreadjob, NQSrewindjob, NQSclosejob -- Operate Job Entry

Format

```
#include <nqsv.h>

nqs_entry NQSopenjob(nqs_odesc *odesc, nqs_res *res)
nqs_jid *NQSreadjob(nqs_entry entry, nqs_res *res)
int NQSrewindjob(nqs_entry entry, nqs_res *res)
int NQSclosejob(nqs_entry entry, nqs_res *res)
```

Function

NQSopenjob() selects jobs related to an object "odesc" from jobs in the execution host, creates a list (job entry) having their job identifiers as its elements, initializes its index to 0, and returns an entry identifier to identify it. When an error occurs NQSopenjob() returns a negative integer and sets an API result code in "res".
The table below lists objects that can be specified for "odesc".

odesc.obj_type value	obj member to be referred	Objects
NQS_OBJ_BSV	None	All jobs
NQS_OBJ_JSV	odesc.obj.jsvid	All jobs controlled by a specified job server.
NQS_OBJ_REQ	odesc.obj.rid	All jobs having a specified request as the parent

NQSreadjob() returns an entry pointed to by the internal index by the "nqs_jid" type pointer and increments the internal index by one. When the last entry comes, NQSreadjob() returns a null and sets NQS_EALLOVER in the error number of "res". When an error occurs, NQSreadjob() returns a null and sets an API result code in "res".

NQSrewindjob() re-initializes the internal index to 0. When an error occurs, NQSrewindjob() returns a negative integer and sets an API result code in "res".

NQSclosejob() deletes a job entry created by NQSopenjob(). When an error occurs, NQSclosejob() returns a negative integer and sets an API result code in "res".

Notes

When you refer to a pointer that points to a job identifier returned by NQSreadjob() after NQSclosejob() was executed, the operation is not assured.

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_ENOENT]

Invalid entry identifier specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSattrjob(3)

6.10.2. **Operation of Job Attributes**

Name

NQSattrjob -- Job Attribute Operation Function

Format

```
#include <nqsv.h>

int NQSattrjob(nqs_jid *jid, nqs_alist ad, int op, nqs_res *res)
```

Function

Operates the attribute of a job having a job identifier "jid". NQSattrjob executes a specified operation "op" on attributes in the attribute list "ad". Values below can be specified for "op".

ATTROP_GET

Copies the attribute of the job onto the attribute "ad".

ATTROP_SET

Replaces the job attribute by the attribute "ad".

Return value

When executed successfully, NQSattrjob() returns 0. When an error occurs, NQSattrjob() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority given to access the attribute.

[NQS_ENOENT]

Cannot find the specified attribute list.

[NQS_ENOREQ]

Cannot find the parent request of the specified job.

[NQS_ENOJOB]

Cannot find the specified job.

[NQS_EUNKNOWN]

Unknown attribute type.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSalist(3), NQSafree(3), NQSaadd(3), NQSaref(3)

6.10.3. **Send Signal to Job**

Name

NQSsigjob -- Send Signal to Job

Format

```
#include <nqsv.h>

int NQSsigjob(nqs_jid *jid, char *sig, nqs_res *res)
```

Function

NQSsigjob() sends a specified signal "sig" to a specified job "jid" NQSsigjob() is valid for jobs whose parent request is RUNNING or SUSPENDED. For other jobs, NQSsigjob() returns an error.

"sig" specifies a signal name having a prefix "SIG" which is one of signal names in the Signal List of "Function" of NQSsigreq(). If the specified signal is not supported by the operating system of the execution host, NQSsigjob() does not send the signal and simply ignores it.

Notes

NQSsigjob() requires an API authority of PRIV_MGR or higher, API authority PRIV_GMGR for the group of the parent request of the job or the owner of the parent request of the job.

Return value

When executed successfully, NQSsigjob() returns 0. When an error occurs, NQSsigjob() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to send a signal to the job.

[NQS_ENOREQ]

Cannot find the parent request of the specified job.

[NQS_ENOJOB]

Cannot find the specified job.

[NQS_EWRNGSTS]

The parent request of the job is not RUNNING or SUSPENDED.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSsigreq(3)

6.10.4. **Migrate Job**

Name

NQSmigjob -- Migrate Batch Job

Format

```
#include <nqsv.h>

int NQSmigjob(nqs_jid *jid, nqs_jsvid *jsvid, nqs_res *res)
```

Function

NQSmigjob() migrates a job "jid" to under control of a job server "jsvid". Even when the source job serve stops, NQSmigjob() can migrate a job, but if the destination job server stops, an error occurs.

To migrate between two different execution hosts, the job server databases (/var/opt/nec/nqsv/jsv) of the hosts must be shared by a shared file system such as NFS.

Notes

NQSmigjob() requires an API authority of PRIV_OPE or higher to run.

Return value

When executed successfully, NQSmigjob() returns 0. When an error occurs, NQSmigjob() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to migrate a batch job.

No authority to execute the job on the destination host.

[NQS_ENOREQ]

Not exist parent request for the specified job.

[NQS_ENOJOB]

Not exist the specified job.

[NQS_ENOJSV]

Cannot find the specified job server.

[NQS_EWRNGSTS]

The parent request of the job is not HELD.

[NQS_EQUEDOM]

The batch job attempted to move to the queue which is not current.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQShldreq(3)

6.11. Node Group related functions

6.11.1. Operations of Node Group Entry

Name

NQSopenngrp, NQSreadngrp, NQSrewindngrp, NQSclosengrp -- Operate Node Group Entry

Format

```
#include <nqsv.h>

nqs_entry NQSopenngrp (nqs_odesc *odesc, nqs_res *res)
nqs_ngrpid *NQSreadngrp(nqs_entry entry, nqs_res *res)
int NQSrewindngrp(nqs_entry entry, nqs_res *res)
int NQSclosengrp(nqs_entry entry, nqs_res *res)
```

Function

NQSopenngrp() selects a node group related to a specified object "odesc" from nodegroups that is recognized by the batch server, creates a list (node group entry) having the node group identifier as the element, its index to 0, and returns an entry identifier to identify it. When an error occurs NQSopenngrp() returns a negative integer and sets an API result code in "res".

The table below lists objects that can be specified for "odesc".

odesc.obj_type value	obj member to be referred	Objects
NQS_OBJ_BSV	None	All node groups
NQS_OBJ_JSV	odesc.obj.jsvid	Node groups which include specified job server
NQS_OBJ_QUE	odesc.obj.qid	Node groups which is bound to specified queue

NQSreadngrp() returns an entry pointed to by the internal index by the "nqs_ngrpid" type pointer, and increments the internal index by one. When the last entry comes, NQSreadngrp() returns a null and sets NQS_EALLOVER in the error number of "res". When an error occurs, NQSreadngrp() returns a null and sets an API result code in "res".

NQSrewindngrp() re-initializes the internal index to 0. When an error occurs, NQSrewindngrp() returns a negative integer and sets an API result code in "res".

NQSclosengrp() deletes a node group entry created by NQSopenngrp(). And returns 0 when executed successfully. When an error occurs, NQSclosengrp() returns a negative integer and sets an API result code in "res".

Notes

When you refer to a pointer that points to a node group identifier returned by NQSreadngrp() after NQSclosengrp() was execute, the operation is not assured.

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_ENOENT]

Invalid entry identifier specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSattrngrp(3)

6.11.2. **Operation of Node Group Attributes**

Name

NQSattrngrp -- Node group Attribute Operation Function

Format

```
#include <nqsv.h>

int NQSattrngrp(nqs_ngrpid *ngrpid, nqs_alist ad, int op, nqs_res *res)
```

Function

Operates the attribute of a node group having a node group identifier "ngrpid". NQSattrngrp() executes an operation specified by "op" on attributes in the attribute list specified by "ad". A value below can be specified for "op".

ATTROP_GET

Copies the attribute of the node group onto the attribute "ad".

ATTROP_SET

Replace the attribute values of the node group attributes specified by "ad".

Return value

When executed successfully, NQSattngrpt() returns 0. When an error occurs, NQSattrngrp() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority given to access the attribute.

[NQS_ENOENT]

Cannot find the specified attribute list.

[NQS_ENONGRP]

Cannot find the specified node group.

[NQS_EUNKNOWN]

Unknown attribute type specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSalist(3), NQSafree(3), NQSaadd(3), NQSaref(3)

6.11.3. **Create/Delete Node Group**

Name

NQScrengrp, NQSdelngrp -- Create/Delete Node Group

Format

```
#include <nqsv.h>

int NQScrengrp(nqs_ngrpid *ngrpid, nqs_alist ad, nqs_res *res)
int NQSdelngrp(nqs_ngrpid *ngrpid, nqs_res *res)
```

Function

NQScreangrp() creates a new node group with the group identifier specified by "ngrpid". If an attribute list including ATTR_COMMENT attribute is specified to "ad", the comment is set at the creation.

NQSdelngrp() deletes the node group whose identifier is "ngrpid".

Notes

NQScrengrp() and NQSdelngrp() require an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQScrengrp() and NQSdelngrp() return 0. When an error occurs, NQScrengrp() and NQSdelngrp() return a negative integer and set an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENOENT]

Cannot find the specified attribute list.

[NQS_EEXIST]

Existing node group specified.

[NQS_EUNKNOWN]

Unknown attribute type specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.11.4. **Add Job server to Node Group**

Name

NQSincludejsv, NQSincludejsvrange, NQSincludengrp -- Add Jobserver to Node Group

Format

```
#include <nqsv.h>

int NQSincludejsv(nqs_ngrpid *ngrpid, nqs_jsvid jsvid[], nqs_res *res)
int NQSincludejsvrange(nqs_ngrpid *ngrpid, nqs_jsvid jsvid[], nqs_res *res)
int NQSincludengrp(nqs_ngrpid *ngrpid, nqs_ngrpid *source, nqs_res *res)
```

Function

NQSincludejsv(), NQSincludejsvrange() and NQSincludengrp() add job servers to the node group specified by "ngrpid".

NQSincludejsv() is used to specify job servers by list of job server identifiers to "jsvid[]".

NQSincludejsvrange() is used to specify job servers by range of job server identifiers. Specify the start job server number to "jsvid[0]" and the last job server number to "jsvid[1]".

NQSincludengrp() is used to specify job servers by source node group to "source".

The job server to be added by NQSincludejsv(), NQSincludejsvrange() or NQSincludengrp() must be already registered in the batch server.

Notes

NQSincludejsv(), NQSincludejsvrange() and NQSincludengrp() require an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQSincludejsv(), NQSincludejsvrange() and NQSincludengrp() return 0. When an error occurs, they return a negative integer and set an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENONGRP]

Cannot find the specified node group.

[NQS_ENOJSV]

Cannot find the specified job server.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.11.5. Remove Job server from Node Group

Name

NQSexcludejsv, NQSexcludejsvrange, NQSexcludengrp -- Remove Jobserver from Node Group

Format

```
#include <nqsv.h>

int NQSexcludejsv(nqs_ngrpid *ngrpid, nqs_jsvid jsvid[], nqs_res *res)
int NQSexcludejsvrange(nqs_ngrpid *ngrpid, nqs_jsvid jsvid[], nqs_res *res)
int NQSexcludengrp(nqs_ngrpid *ngrpid, nqs_ngrpid *source, nqs_res *res)
```

Function

NQSexcludejsv(), NQSexcludejsvrange() and NQSexcludengrp() remove job servers from the node group specified by "ngrpid".

NQSexcludejsv() is used to specify job servers to remove by list of job server identifiers to "jsvid[]".

NQSexcludejsvrange() is used to specify job servers to remove by range of job server identifiers. Specify the start job server number to "jsvid[0]" and the last job server number to "jsvid[1]".

NQSexcludengrp() is used to specify job servers to remove by source node group to "source".

When the specified job server is not included in the node group of "ngrpid", NQSexcludejsv(), NQSexcludejsvrange() and NQSexcludengrp() only skip to perform for the job server.

Notes

NQSexcludejsv(), NQSexcludejsvrange() and NQSexcludengrp() require an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQSexcludejsv(), NQSexcludejsvrange() and NQSexcludengrp() return 0. When an error occurs, they stop removing job servers and return a negative integer. Also an API result code is set in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENONGRP]

Cannot find the specified node group.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.11.6. Bind/Unbind Node Group to Queue

Name

NQSbindngrp, NQSunbindngrp() -- Bind/Unbind Node group to a queue.

Format

```
#include <nqsv.h>

int NQSbindngrp(nqs_qid *qid, nqs_ngrpid *ngrpid, nqs_res *res)
int NQSunbindngrp(nqs_qid *qid, nqs_ngrpid *ngrpid, nqs_res *res)
```

Function

NQSbindngrp() connects (or binds) the node group specified by "ngrpid" to the queue whose identifier us "qid". It acts like all job servers included in the node group is bound to the specified queue.

NQSunbindngrp() disconnects (or unbinds) the node group specified by "ngrpid" from the queue whose identifier us "qid". It acts like all job servers included in the node group is unbound from the specified queue.

The queue types that can be specified to "qid" are batch queue and interactive queue.

Notes

The API authority of PRIV_MGR or higher is required to execute NQSbindngrp() and NQSunbindngrp().

Return value

When executed successfully, NQSbindngrp() and NQSunbindngrp() return 0. When an error occurs, they return a negative integer and set an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to a batch server.

[NQS_ECONFAIL]

Cannot connect to the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority given to execute the API function.

[NQS_ENONGRP]

Cannot find the specified node group.

[NQS_ENOQUE]

Cannot find the specified queue.

[NQS_EWRNGTYP]

Unacceptable the queue type of specified queue.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.12. Template related functions

6.12.1. Create/Delete Template

Name

NQScretemp, NQSdeltemp -- Create/Delete Template

Format

```
#include <nqsv.h>

int NQScretemp(nqs_template *template, nqs_res *res)
int NQSdeltemp(char *template_name, nqs_res *res)
```

Function

NQScretemp () creates a new template according to the template specified by "template". Definition information (structure in "template") is designated as the type of made templates (template->type) below.

Type of template	template->type	Structure in "template" which refers
OpenStack	NQSII_TEMPLATE_TYPE_OPENSTACK	struct nqs_ostemplate os_tmpl
Container	NQSII_TEMPLATE_TYPE_CONTAINER	struct nqs_cotemplate co_tmpl

NQSdeltemp () deletes the template whose identifier is "template_name".

Notes

NQScretemp(), NQSdeltemp() require an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQScretemp() and NQSdeltemp() return 0. When an error occurs, they return a negative integer and set an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENOENT]

Cannot find the specified template.

[NQS_EEXIST]

Existing template specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.12.2. **Edit Template**

Name

NQSedittemp - Edit Template.

Format

```
#include <nqsv.h>

int NQSedittemp(nqs_template *template, int changeflg, nqs_res *res)
```

Function

NQSedittemp() overwrites a template of specified template->template_name by information in "template". Data of an overwritten target is indicated by a flag of "changeflg".

The flag by which designation is possible in "changeflg" is below. More than one specify is also possible to take OR.

・ TMPL_MEMBER_IMGNAME	image name
・ TMPL_MEMBER_CPU	CPU number
・ TMPL_MEMBER_MEM	Memory size
・ TMPL_MEMBER_GPU	GPU number
・ TMPL_MEMBER_CUSTOM	Custom define
・ TMPL_MEMBER_COMMENT	Comment
・ TMPL_MEMBER_FLAVOR	Flavor name
(Only when template->type is NQSII_TEMPLATE_TYPE_OPENSTACK)	
・ TMPL_MEMBER_STARTTIMEOUT	Start time-out
・ TMPL_MEMBER_STOPTIMEOUT	Stop time-out

Notes

NQSedittemp() require an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQSedittemp() returns 0. When an error occurs, NQSedittemp() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENONENT]

Cannot find the specified template.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.12.3. **Lock and Unlock Template**

Name

NQSlocktemp, NQSunlocktemp - Lock and Unlock the Template.

Format

```
#include <nqsv.h>

int NQSlocktemp(char *template_name, nqs_res *res)
int NQSunlocktemp(char *template_name, nqs_res *res)
```

Function

NQSlocktemp() lock the template specified by "template_name".

NQSunlocktemp() unlock the template specified by "template_name".

Notes

NQSlocktemp() and NQSunlocktemp() require an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQSlocktemp() and NQSunlocktemp() return 0. When an error occurs, they return a negative integer and set an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENONENT]

Cannot find the specified template.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.13. OpenStack Template related functions

6.13.1. Create/Delete OpenStack Template

Name

NQScreostemp, NQSdelostemp -- Create/Delete OpenStack Template

Format

```
#include <nqsv.h>

int NQScreostemp(nqs_ostemplate *template, nqs_res *res)
int NQSdelostemp(char *template_name, nqs_res *res)
```

Function

NQScreostemp () creates a new OpenStack template according to the template specified by "template".

NQSdelostemp () deletes the OpenStack template whose identifier is "template_name".

Notes

NQScreostemp(), NQSdelostemp() require an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQScreostemp() and NQSdelostemp() return 0. When an error occurs, they return a negative integer and set an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENOENT]

Cannot find the specified template.

[NQS_EEXIST]

Existing template specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.13.2. **Edit OpenStack Template**

Name

NQSeditostemp - Edit OpenStack Template.

Format

```
#include <nqsv.h>

int NQSeditostemp(nqs_ostemplate *template, int changeflg, nqs_res *res)
```

Function

NQSeditostemp() overwrites a template of specified template->template_name by information in "template". Data of an overwritten target is indicated by a flag of "changeflg".

The flag by which designation is possible in "changeflg" is below. More than one specify is also possible to take OR.

・ OSTMPL_MEMBER_IMGNAME	OS image name
・ OSTMPL_MEMBER_CPU	CPU number
・ OSTMPL_MEMBER_MEM	Memory size
・ OSTMPL_MEMBER_GPU	GPU number
・ OSTMPL_MEMBER_CUSTOM	Custom define
・ OSTMPL_MEMBER_COMMENT	Comment
・ OSTMPL_MEMBER_FLAVOR	Flavor name
・ OSTMPL_MEMBER_STARTTIMEOUT	Start time-out
・ OSTMPL_MEMBER_STOPTIMEOUT	Stop time-out

Notes

NQSeditostemp() require an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQSeditostemp() returns 0. When an error occurs, NQSeditostemp() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENONENT]

Cannot find the specified template.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.13.3. **Lock and Unlock OpenStack Template**

Name

NQSlockostemp, NQSunlockostemp - Lock and Unlock the OpenStack Template.

Format

```
#include <nqsv.h>

int NQSlockostemp(char *template_name, nqs_res *res)
int NQSunlockostemp(char *template_name, nqs_res *res)
```

Function

NQSlockostemp() lock the template specified by "template_name".

NQSunlockostemp() unlock the template specified by "template_name".

Notes

NQSlockostemp() and NQSunlockostemp() require an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQSlockostemp() and NQSunlockostemp() return 0. When an error occurs, they return a negative integer and set an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENONENT]

Cannot find the specified template.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.14. Baremetal server related functions

6.14.1. Attach and detach baremetal server

Name

NQSattachhst_bm, NQSdetachhst_bm - Attach and detach the baremetal server.

Format

```
#include <nqsv.h>

int NQSattachhst_bm(nqs_hid *hid,   nqs_jsvid *jsvid, int cpu, int memsz, int memunit,
int gpu, nqs_res *res)
int NQSdetachhst_bm(nqs_hid *hid, nqs_jsvid *jsvid, nqs_res *res)
```

Function

NQSattachhst_bm() attaches baremetal server as an execution host to batch server. It is specified the job server ID, cpu, memsz, memunit, and gpu.

To attach baremetal server, the job server must not LINKUP. When there is something about which specified hid or jsvid agrees with the execution host registered already, it'll be an error.

The following one can be specified for memunit (the unit of the memory size).

NQS_LIM_BYTE	Byte
NQS_LIM_KBYTE	KiloByte
NQS_LIM_MBYTE	MegaByte
NQS_LIM_GBYTE	GigaByte
NQS_LIM_TBYTE	TeraByte
NQS_LIM_PBYTE	PetaByte
NQS_LIM_EBYTE	ExaByte

NQSdetachhst_bm() detaches baremetal server. Please set NULL as the person who designates and doesn't designate one of hid and jsvid.

A job server can carry out elimination of a baremetal server only in the state which doesn't stand up. When anything to agree with specified hid or jsvid doesn't exist, it'll be an error.

Notes

NQSattachhst_bm() and NQSdetachhst_bm() require an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQSattachhst_bm() and NQSdetachhst_bm() return 0. When an error occurs, they return a negative integer and set an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

[NQS_EALREADY]

Specified host or JSV ID has already attached.

Related items

NQSconnect(3)

6.15. Custom Resource related functions

6.15.1. Operations of Custom Resource Entry

Name

NQSopencustom, NQSreadcustom, NQSrewindcustom, NQSclosecustom -- Operate Custom resource Entry

Format

```
#include <nqsv.h>

nqs_entry NQSopencustom (nqs_odesc *odesc, nqs_res *res)
nqs_customid *NQSreadcustom(nqs_entry entry, nqs_res *res)
int NQSrewindcustom(nqs_entry entry, nqs_res *res)
int NQSclosecustom(nqs_entry entry, nqs_res *res)
```

Function

NQSopencustom() selects a custom resource related to a specified object "odesc" from custom resources that is recognized by the batch server, creates a list (custom resource entry) having the custom resource identifier as the element, its index to 0, and returns an entry identifier to identify it. When an error occurs NQSopencustom() returns a negative integer and sets an API result code in "res".

The table below lists objects that can be specified for "odesc".

odesc.obj_type value	obj member to be referred	Objects
NQS_OBJ_BSV	None	All custom resources

NQSreadcustom() returns an entry pointed to by the internal index by the "nqs_customid" type pointer, and increments the internal index by one. When the last entry comes, NQSreadcustom() returns a null and sets NQS_EALLOVER in the error number of "res". When an error occurs, NQSreadcustom() returns a null and sets an API result code in "res".

NQSrewindcustom() re-initializes the internal index to 0. When an error occurs, NQSrewindcustom() returns a negative integer and sets an API result code in "res".

NQSclosecustom() deletes a custom resource entry created by NQSopencustom(). And returns 0 when executed successfully. When an error occurs, NQSclosecustom() returns a negative integer and sets an API result code in "res".

Notes

When you refer to a pointer that points to a custom resource identifier returned by NQSreadcustom() after NQSclosecustom() was execute, the operation is not assured.

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_ENOENT]

Invalid entry identifier specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSattrcustom(3)

6.15.2. **Operation of Custom Resource Attributes**

Name

NQSattrcustom -- Custom resource Attribute Operation Function

Format

#include <nqsv.h>

int NQSattrcustom(char *cr_name, nqs_alist ad, int op, nqs_res *res)

Function

Operates the attribute of a custom resource having a custom resource name "cr_name". NQSattrcustom() executes an operation specified by "op" on attributes in the attribute list specified by "ad". A value below can be specified for "op".

ATTROP_GET

Copies the attribute of the custom resource onto the attribute "ad".

ATTROP_SET

Replace the attribute values of the custom resource attributes specified by "ad".

ATTROP_ADD

Add the attribute of the custom resource onto the attribute "ad".

ATTROP_DEL

Delete the attribute values of the custom resource attributes specified by "ad".

Return value

When executed successfully, NQSattcustomt() returns 0. When an error occurs, NQSattrcustom() returns a negative integer and sets an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority given to access the attribute.

[NQS_ENOENT]

Cannot find the specified attribute list.

[NQS_ENOCR]

Cannot find the specified custom resource.

[NQS_EUNKNOWN]

Unknown attribute type specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3), NQSalist(3), NQSafree(3), NQSaadd(3), NQSaref(3)

6.15.3. **Create/Delete Custom Resource**

Name

NQScrecustom, NQSdelcustom -- Create/Delete Custom resource

Format

```
#include <nqsv.h>

int NQScrecustom(nqs_crid *crid, int consumer, nqs_alist ad, nqs_res *res)
int NQSdelcustom(nqs_crid *crid, nqs_res *res)
```

Function

NQScrecustom() creates a new custom resource with the custom resource identifier specified by "crid". The consume unit of the made custom resource is designated in consumer and the following one of prices are designated.

- CR_JOB : job
- CR_REQ : request

It is possible to designate control information to use the custom resource in "ad". Even if this information isn't designated, it is possible to make a custom resource, and -1 is designated in "ad" in that case.

NQSdelcustom() deletes the custom resource whose identifier is "crid".

Notes

NQScrecustom() and NQSdelcustom() require an API authority of PRIV_MGR or higher to run.

Return value

When executed successfully, NQScrecustom() and NQSdelcustom() return 0. When an error occurs, NQScrecustom() and NQSdelcustom() return a negative integer and set an API result code in "res".

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect with the batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_EPERM]

No authority to execute the API function.

[NQS_ENOENT]

Cannot find the specified attribute list.

[NQS_EEXIST]

Existing node group specified.

[NQS_EUNKNOWN]

Unknown attribute type specified.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

6.16. Utility functions

6.16.1. Get API Version

Name

NQSapiver -- Get API Version

Format

```
#include <nqsv.h>

char *NQSapiver(void)
```

Function

NQSapiver() gets a character string of the NQSV/API version in the format below.

xx.yy (zzzz)	xx	...	Major version number (2 digits in decimal)
	yy	...	Minor version number (2 digits in decimal)
	zzzz	...	Platform name (OS identification name)

NQSapiver() can be invocated independently of whether the API link is established.

Return value

NQSapiver() always succeeds and returns a pointer to the version character string.

6.16.2. Convert between Machine ID and Host Name

Name

NQSmid2hname, NQShname2mid

-- Convert Machine ID to Host Name, Convert Host Name to Machine ID

Format

```
#include <nqsv.h>

char *NQSmid2hname(int mid, nqs_res *res)
int NQShname2mid(char *hostname, nqs_res *res)
```

Function

NQSmid2hname() converts a machine ID "mid" to a corresponding host name and returns a pointer to a character string representing a host name. When failing in conversion, NQSmid2hname() sets a result code in "res" and returns a null.

NQShname2mid() converts a host name "hostname" to a corresponding machine ID and returns an integer representing a machine ID. When failing in conversion, NQShname2mid() sets a result code in "res" and returns a negative integer.

Error

When an error occurs, one of the result codes below is set as an error number in the result code.

[NQS_ENOTCONN]

Not connected to the batch server.

[NQS_ECONFAIL]

Cannot connect to a batch server.

[NQS_EPKTSEND]

Cannot send the API packet.

[NQS_EPKTRECV]

Cannot receive the API packet.

[NQS_EDISCONN]

Disconnected from the batch server in transmission.

[NQS_ENETDB]

Cannot find a host of the specified host name.

[NQS_ENAMPDB]

The specified host has not been cataloged in the NMAP database.

[NQS_ENOMEM]

Cannot allocate memory dynamically.

[NQS_EINVAL]

Invalid argument specified.

Related items

NQSconnect(3)

7. Sample Codes

7.1. Display the Load Information of Execution Host

This sample displays the load information of each execution host every five seconds.

```
#include <sys/types.h>
#include <stdio.h>
#include <libgen.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include "nqsv.h"


void usage(char *argv[])
{
    fprintf(stderr, "Usage: %s [-h <bsv host>]\n", basename(argv[0]));
    exit(1);
}


int main(int argc, char *argv[])
{
    int priv = PRIV_USR;
    nqs_odesc obj;
    nqs_entry entry;
    nqs_alist ad;
    nqs_res res;
    nqs_hid *hid;
    int i, sd, c;
    char *bsv;
    void *p;
    nqs_aid aid[] = {
        {ATTR_HSTID,     SCPE_HST},
        {ATTR_RBSPMEM,   SCPE_HST},
        {ATTR_RBSPSWAP,  SCPE_HST},
        {ATTR_RBCPUNM,   SCPE_HST},
        {ATTR_RBLDAVG,   SCPE_HST},
        {ATTR_RBCPUAVG,  SCPE_HST},
        {       -1,           -1}
    };

    bsv = NULL;
    while ((c = getopt(argc, argv, "h:")) != -1) {
        switch (c) {
        case 'h':
            bsv = optarg;
            break;
        default:
            usage(argv);
        }
    }
    if ((sd = NQSconnect(bsv, 0, priv, &res)) < 0) {
        fprintf(stderr, "NQSconnect: %s\n", res.msg);
        exit(1);
    }
    ad = -1;
    i = 0;
    while (aid[i].type != -1) {
        if ((ad = NQSalist(ad, &aid[i], &res)) < 0) {
            fprintf(stderr, "NQSalist#%d: %s\n", i, res.msg);
            exit(1);
        }
        i ++;
    }
    printf("ExecHost         memory        swap       "
          "CPU       Load avg.          CPU avg.\n"
          "-----------------------------------------------"
          "---------------------------------------------\n");
    fflush(stdout);

    while (1) {
        obj.obj_type = NQS_OBJ_BSV;
        if ((entry = NQSopenhst(&obj, &res)) < 0) {
            fprintf(stderr, "NQSopenhst: %s\n", res.msg);
            exit(1);
        }
        while ((hid = NQSreadhst(entry, &res)) != NULL) {
            if (NQSattrhst(hid, ad, ATTROP_GET, &res) < 0) {
                fprintf(stderr, "NQSattrhst: %s\n", res.msg);
                exit(1);
            }
            i = 0;
            while (aid[i].type != -1) {
                if ((p = NQSaref(ad, &aid[i], NULL, &res)) == NULL) {
                    fprintf(stderr, "NQSaref#%d: %s\n", i, res.msg);
                    exit(1);
                }
                switch (aid[i].type) {
                case ATTR_HSTID:
                    printf("%s ", inet_ntoa(((nqs_hid *)p)->ip));
                    break;
                case ATTR_RBSPMEM:
                    printf("%6d %6d ",
                          ((nqs_rsgres *)p)->initial,
                          ((nqs_rsgres *)p)->using);
                    break;
                case ATTR_RBSPSWAP:
                    printf("%6d %6d ",
                          ((nqs_rsgres *)p)->initial,
                          ((nqs_rsgres *)p)->using);
                    break;
                case ATTR_RBCPUNM:
                    printf("%3d %3d ",
                          ((nqs_rsgres *)p)->initial,
                          ((nqs_rsgres *)p)->using);
                    break;
                case ATTR_RBLDAVG:
                    printf("%6.2f %6.2f %6.2f ",
                          ((nqs_rsgavg *)p)->avg01,
                          ((nqs_rsgavg *)p)->avg05,
                          ((nqs_rsgavg *)p)->avg15);
                    break;
                case ATTR_RBCPUAVG:
                    printf("%6.2f %6.2f %6.2f ",
                          ((nqs_rsgavg *)p)->avg01,
                          ((nqs_rsgavg *)p)->avg05,
                          ((nqs_rsgavg *)p)->avg15);
                    break;
                }
                i ++;
            }
            printf("\n");
            fflush(stdout);
        }
        if (NQSclosehst(entry, &res) < 0) {
            fprintf(stderr, "NQSclosehst: %s\n", res.msg);
            exit(1);
        }
        sleep(5);
    }
    /* NOTREACHED */
}
```

7.2. Display the Request Status Event

This sample displays the NQSV event information of the batch request status.

```
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <libgen.h>
#include "nqsv.h"


void usage(char *argv[])
{
    fprintf(stderr, "Usage: %s -h <bsv host>\n", basename(argv[0]));
    exit(1);
}

int main(int argc, char **argv)
{
    fd_set rfds;
    nqs_res res;
    nqs_event event;
    int c, sd, priv = PRIV_USR;
    char *bsv, *date, *p;

    bsv = NULL;
    while ((c = getopt(argc, argv, "h:")) != -1) {
        switch (c) {
        case 'h':
            bsv = optarg;
            break;
        default:
            usage(argv);
        }
    }
    if ((sd = NQSconnect(bsv, 0, priv, &res)) < 0) {
        fprintf(stderr, "NQSconnect: %s\n", res.msg);
        exit(1);
    }
    if (NQSevflt(NQSEVT_RST, EVFLT_ADD, &res) < 0) {
        fprintf(stderr, "NQSevflt: %s\n", res.msg);
        exit(1);
    }
    FD_ZERO(&rfds);
    while (1) {
        FD_SET(sd, &rfds);
        if (select(sd + 1, &rfds, NULL, NULL, NULL) < 0) {
            exit(1);
        } else if (FD_ISSET(sd, &rfds)) {
            if (NQSevent(&event, &res) < 0) {
                fprintf(stderr, "NQSevent: %s\n", res.msg);
                exit(1);
            }
            date = ctime(&event.occur_time);
            if ((p = strchr(date, '\n')) != NULL) {
                *p = '\0';
            }
            printf("[%s] ", date);
            switch (NQSEVT_TYPE(event.event_id)) {
            case NQSEVT_BSV:
                if (event.event_id == NQSEVT_BSV_LINKDOWN) {
                    printf("API link was down.\n");
                    exit(0);
                }
                break;
            case NQSEVT_RST:
                switch (event.event_id) {
                case NQSEVT_RST_ARRIVING:    p = "ARRIVING";
                    break;
                case NQSEVT_RST_WAITING:     p = "WAITING";
                    break;
                case NQSEVT_RST_QUEUED:      p = "QUEUED";
                    break;
                case NQSEVT_RST_PRERUNNING:  p = "PRE-RUNNING";
                    break;
                case NQSEVT_RST_RUNNING:     p = "RUNNING";
                    break;
                case NQSEVT_RST_POSTRUNNING:  p = "POST-RUNNING";
                    break;
                case NQSEVT_RST_EXITING:     p = "EXITING";
                    break;
                case NQSEVT_RST_EXITED:      p = "EXITED";
                    break;
                case NQSEVT_RST_HOLDING:     p = "HOLDING";
                    break;
                case NQSEVT_RST_HELD:        p = "HELD";
                    break;
                case NQSEVT_RST_RESTARTING:  p = "RESTARTING";
                    break;
                case NQSEVT_RST_SUSPENDING:  p = "SUSPENDING";
                    break;
                case NQSEVT_RST_SUSPENDED:   p = "SUSPENDED";
                    break;
                case NQSEVT_RST_RESUMING:    p = "RESUMING";
                    break;
                case NQSEVT_RST_MIGRATING:   p = "MIGRATING";
                    break;
                case NQSEVT_RST_MOVED:       p = "MOVED";
                    break;
                case NQSEVT_RST_STAGING:     p = "STAGING";
                    break;
                case NQSEVT_RST_CHKPNTING:    p = "CHKPNTING";
                    break;
                }
                printf("rid: %d.%d state: %s ",
                       event.cargo.rst.rid.seqno, event.cargo.rst.rid.mid, p);
                if (event.cargo.rst.rst.res.err) {
                    printf("(err: %s)\n", event.cargo.rst.rst.res.msg);
                } else {
                    printf("\n");
                }
                fflush(stdout);
                break;
            }
        }
    }
    /* NOTREACHED */
}
```

Appendix.A Update history

6th edition

- Add the attributes about the urgent request.
- Add the attribute to capture SIGTERM.

7th edition

- Deleted the description about Multi-cluster.
- 4.8 Request attributes
  Add Platform MPI

8th edition

- 2.1 State Transition
  Update the figure of the states of the request

NEC Network Queuing System V (NQSV) User's Guide

[API]

January 2023　8th edition

NEC Corporation