NEC

NEC Network Queuing System V (NQSV) 利用の手引

[API 編]

輸出する際の注意事項

本製品（ソフトウェアを含む）は、外国為替および外国貿易法で規定される規制貨物（または役務）に該当することがあります。

その場合、日本国外へ輸出する場合には日本国政府の輸出許可が必要です。

なお、輸出許可申請手続きにあたり資料等が必要な場合には、お買い上げの販売店またはお近くの当社営業拠点にご相談ください。

はしがき

本書は、NEC Network Queuing System V（NQSV）によるジョブ管理の API について説明したものです。

備考

(1)　本書はNEC Network Queuing System V (NQSV) R1.00以降に対応しています。

(2)　本書に説明しているすべての機能はプログラムプロダクトであり、以下のプロダクト名およびプロダクト番号に対応しています。

プロダクト名	型番
NEC Network Queuing System V (NQSV) /ResourceManager	UWAF00 UWHAF00（サポートパック）
NEC Network Queuing System V (NQSV) /JobServer	UWAG00 UWHAG00（サポートパック）
NEC Network Queuing System V (NQSV) /JobManipulator	UWAH00 UWHAH00（サポートパック）

(3)　UNIX は The Open Group の登録商標です。

(4)　Intelは、アメリカ合衆国およびその他の国における Intel Corporation の商標です。

(5)　OpenStackは、アメリカ合衆国およびその他の国における OpenStack Foundation の商標です。

(6)　Red Hat OpenStack Platformは、アメリカ合衆国およびその他の国における Red Hat, Inc. の商標です。

(7)　Linux は Linus Torvalds氏の米国およびその他の国における登録商標あるいは商標です。

(8)　Dockerはアメリカ合衆国及びその他の国におけるDocker, Inc.の商標です。

(9)　InfiniBand は、InfiniBand Trade Associationの商標またはサービスマークです。

(10)　Zabbixはラトビア共和国にあるZabbix LLCの商標です。

(11)　その他、記載されている会社名、製品名は、各社の登録商標または商標です。

本書の対象読者

本書は、NQSV の API ライブラリを利用するソフトウェア開発者を対象読者として書かれたものです。

本書は、対象読者が C 言語のプログラミング全般、および、NQSV に関する一般的な知識を知っていることを前提として記述されています。

本書の読み進め方

本書は、次の構成となっています。章ごとに対象読者の範囲は異なっており、表の一番右の列にその範囲を示しています。記載された対象読者の後に(*)がついている章については、該当する読者は必ずお読みください。

章	タイトル	内容	対象読者
1	NQSV/APIの利用方法	APIの概要と利用方法	ソフトウェア開発者(*)
2	リクエストの状態遷移	バッチキュー、会話キューのリクエストが取り得る状態と状態間の遷移関係	ソフトウェア開発者(*)
3	APIイベント	APIイベントの種類	ソフトウェア開発者(*)
4	属性一覧	NQSVで取り扱う属性の一覧	ソフトウェア開発者(*)
5	データ型／定数定義	データ型／定数定義の説明	ソフトウェア開発者(*)
6	API関数	API関数情報	ソフトウェア開発者(*)
7	サンプルコード	サンプルコードの紹介	ソフトウェア開発者

関連説明書

NEC Network Queuing System V (NQSV)のマニュアルは以下で構成されています。

マニュアル名称	内容
NEC Network Queuing System V (NQSV) 利用の手引［導入編］	システムの全体像および基本的なシステムの構築方法について説明します。
NEC Network Queuing System V (NQSV) 利用の手引［管理編］	NQSV の管理者が実施する各種設定について説明しています。
NEC Network Queuing System V (NQSV) 利用の手引［操作編］	NQSV の一般利用者が使用する各種機能について説明しています。
NEC Network Queuing System V (NQSV) 利用の手引［リファレンス編］	コマンドリファレンスに関して説明しています。
NEC Network Queuing System V (NQSV) 利用の手引［API 編］	NQSV を操作する C 言語のプログラミングインタフェース (API) について説明しています。
NEC Network Queuing System V (NQSV) 利用の手引［JobManipulator 編］	NQSV のスケジューラコンポーネント JobManipulator に関して説明しています。
NEC Network Queuing System V (NQSV) 利用の手引［アカウンティング・予算管理編］	NQSV のアカウンティング機能に関して説明しています。

表記上の約束

本書では次の表記規則を使用しています。

省略記号 ...	前述の項目を繰り返すことができることを表しています。ユーザは同様の項目を任意の数だけ入力することができます。
縦棒 |	オプションまたは必須の選択項目を分割します。
中かっこ {}	１つを選択しなければならない一連パラメータまたはキーワードを表しています。
角かっこ []	省略可能な一連パラメータまたはキーワードを表しています。

用語定義・略語

用語・略語	説　明
ベクトルエンジン (VE、Vector Engine)	SX-Aurora TSUBASAの中核であり、ベクトル演算を行う部分です。複数のコアと共用メモリが実装されています。PCI Expressカードであり、x86サーバに搭載して使用します。
ベクトルホスト (VH、Vector Host)	ベクトルエンジンを保持するサーバ、つまり、ホストコンピュータを指します。
ベクトルアイランド (VI)	VH1台にVEを1枚ないし複数枚組み込んだ単位を指します。
バッチサーバ（BSV）	バッチサーバホスト上に常駐し、NQSV全体を管理します。
ジョブサーバ（JSV）	実行ホスト上に常駐し、ジョブの実行を管理します。
JobManipulator（JM）	NQSVの提供するスケジューラ機能です。計算リソースの空き容量を管理し、ジョブの開始時刻を割り当てます。
アカウンティングサーバ	アカウント情報の収集・管理、予算管理を行います。
リクエスト	NQSVにおけるユーザジョブの単位であり、１つまたは複数のジョブから構成されたものです。バッチサーバによって管理されます。
ジョブ	ユーザジョブの実行単位であり、ジョブサーバによって管理されます。
論理ホスト	実行ホストの資源を論理的（仮想的）に分割し、ジョブに割り当てたものです
キュー	受け取ったリクエストをプールし、管理する機構です。
BMC	Board Management controllerの略語です。IPMI(Intelligent Platform Management Interface)と呼ばれる業界標準のサーバーマネジメントインタフェースに準拠したサーバー管理を行います。
HCA	Host Channel Adapterの略語です。IBネットワークに接続するためにサーバ側に取り付けるPCIeカードです。
IB	InfiniBandの略語です。
MPI	Message Passing Interfaceの略語です。主にノード間で並列コンピューティングを行うための標準規格です。
NIC	Network Interface Cardの略語です。他ノードと通信するためのハードウェアです。

目 次

は し が き i
用語定義・略語 vi
目 次 vii
第 1 章 NQSV/API の利用方法 1
1.1 NQSV/API の概要 1
1.2 パッケージインストール 1
1.3 インストールパス 1
1.4 使用方法 2
1.4.1 コンパイル方法 2
1.4.2 バッチサーバとの接続 2
1.4.3 リザルトコード 2
1.4.4 属性操作方法 2
1.4.5 イベント取得方法 4
第 2 章 リクエストの状態遷移 5
2.1 キュー内での状態遷移 5
2.2 転送キュー内での状態遷移 7
2.3 リクエスト状態の詳細 7
2.4 リクエストのストール 10
第 3 章 API イベント 11
3.1 API イベントの種類 11
第 4 章 属性一覧 15
4.1 バッチサーバ属性 16
4.2 スケジューラ属性 20
4.3 ジョブサーバ属性 21
4.4 実行ホスト属性 23
4.5 キュー属性（バッチキュー、会話キュー） 25
4.6 転送キュー属性 38
4.7 ネットワークキュー属性 40
4.8 リクエスト属性 41
4.9 ジョブ属性 52

4.10 ノードグループ属性 55
4.11 ワークフロー属性 56
4.12 カスタムリソース属性 57
第 5 章 データ型／定数定義 58
5.1 記号定数 58
5.2 構造体 60
第 6 章 API 関数 85
6.1 属性リスト関数 85
6.1.1 属性リストの作成 85
6.1.2 属性リストの解放 86
6.1.3 属性リストへの属性値の追加 87
6.1.4 属性リスト内の属性値の削除 88
6.1.5 属性リスト内の属性値の参照 89
6.1.6 属性エントリの操作 90
6.2 API 初期化／終了関数 93
6.2.1 API リンクのオープン 93
6.2.2 API リンクのクローズ 95
6.3 API イベント関連関数 96
6.3.1 API イベントの取得 96
6.3.2 イベントフィルタの設定 97
6.4 スケジューラ関連関数 98
6.4.1 スケジューラ ID の登録 98
6.4.2 スケジューラエントリの操作 100
6.4.3 スケジューラ属性操作関数 102
6.5 バッチサーバ関連関数 103
6.5.1 バッチサーバ属性操作関数 103
6.5.2 バッチサーバの停止 104
6.5.3 実行ホストの登録・削除 105
6.6 ジョブサーバ関連関数 107
6.6.1 ジョブサーバエントリの操作 107
6.6.2 ジョブサーバ属性操作関数 109
6.6.3 ジョブサーバの制御 110
6.7 実行ホスト関連関数 113
6.7.1 実行ホストエントリの操作 113

6.7.2 実行ホスト属性操作関数 115
6.8 キュー関連関数 116
6.8.1 キューエントリの操作 116
6.8.2 キュー属性操作関数 118
6.8.3 キューの作成 120
6.8.4 キューの削除 121
6.8.5 スケジューラの接続 122
6.8.6 スケジューラの切り放し 123
6.8.7 ジョブサーバの接続 124
6.8.8 ジョブサーバの切り放し 125
6.9 リクエスト関連関数 126
6.9.1 リクエストエントリの操作 126
6.9.2 リクエスト属性操作関数 128
6.9.3 リクエストの作成とキューへの投入 129
6.9.4 ジョブの生成とステージインの開始 132
6.9.5 リクエストの実行 134
6.9.6 リクエストの削除 136
6.9.7 リクエストへのシグナル送信 137
6.9.8 リクエストのホールド 139
6.9.9 リクエストのリリース 141
6.9.10 リクエストのリスタート 142
6.9.11 リクエストのリラン 143
6.9.12 リクエストのキュー間移動 144
6.10 ジョブ関連関数 145
6.10.1 ジョブエントリの操作 145
6.10.2 ジョブ属性操作関数 147
6.10.3 ジョブへのシグナル送信 148
6.10.4 ジョブのマイグレーション 149
6.11 ノードグループ関連関数 150
6.11.1 ノードグループエントリの操作 150
6.11.2 ノードグループ属性操作関数 152
6.11.3 ノードグループの作成・削除 153
6.11.4 ノードグループへのジョブサーバ追加 154
6.11.5 ノードグループからのジョブサーバ削除 156

6.11.6 ノードグループのキューへのバインド・アンバインド 158
6.12 テンプレート関連関数 159
6.12.1 テンプレートの作成・削除 159
6.12.2 テンプレートの編集 161
6.12.3 テンプレートのロック・アンロック 162
6.13 OpenStack テンプレート関連関数 163
6.13.1 OpenStack テンプレートの作成・削除 163
6.13.2 OpenStack テンプレートの編集 164
6.13.3 OpenStack テンプレートのロック・アンロック 165
6.14 ベアメタルサーバ関連関数 166
6.14.1 ベアメタルサーバの登録・削除 166
6.15 カスタムリソース関連関数 168
6.15.1 カスタムリソースエントリの操作 168
6.15.2 カスタムリソース属性操作関数 170
6.15.3 カスタムリソースの作成・削除 171
6.16 ユーティリティ関数 172
6.16.1 API バージョンの取得 172
6.16.2 マシン ID ホスト名変換 173
第 7 章 サンプルコード 174
7.1 実行ホストの負荷情報表示 174
7.2 リクエスト状態系イベントの表示 177
付録 A 発行履歴 180
A.1 発行履歴一覧表 180
A.2 追加・変更点詳細 180
索 引 181

第1章　NQSV/API の利用方法

1.1　NQSV/APIの概要

NQSV/API は、バッチサーバに接続して各種情報取得や各種操作を行うための C 言語 API です。本 API を利用することにより、独自のクライアントコマンドやスケジューラを作成することができます。

1.2　パッケージインストール

本 API を利用したプログラムを開発するために、事前に C 言語のアプリケーション開発環境を構築してください。

アプリケーション開発環境が構築されたホストに、NQSV/ResourceManager の中の NQSV-API-X.XX-X.x86_64.rpm パッケージをインストールしてください。

1.3　インストールパス

NQSV/API のヘッダ、ライブラリのインストール先は以下のとおりです。

ヘッダファイル

/opt/nec/nqsv/include/nqsv.h

NQSVライブラリ

/opt/nec/nqsv/lib64/libnqsv.a　... スタティックリンクライブラリ

/opt/nec/nqsv/lib64/libnqsv.so　... ダイナミックリンクライブラリ

1.4　使用方法

1.4.1　コンパイル方法

NQSV/API を利用するには、ソースコード上でヘッダファイル nqsv.h をインクルードの上、NQSV ライブラリ libnqsv.a または libnqsv.so とリンクしてください。

コンパイル時のコマンド例：apitest.c をコンパイルする場合

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

1.4.2　バッチサーバとの接続

NQSV/API は、バッチサーバと TCP 接続(API リンク)を確立した上で、バッチサーバとの間で各種情報取得や各種操作の処理を行います。

このため、以下で説明する NQSV/API 関数の呼び出しを行う際には、最初に NQSconnect()関数を呼び出して、バッチサーバとの接続の確立を行い、NQSV/API での処理を終了する際は、NQSdisconnect()でバッチサーバとの接続の終了処理を行う必要があります。

1.4.3　リザルトコード

NQSV/API の各関数は、nqs_res 構造体により、処理の結果（リザルトコード）を設定して返却するようになっています。

リザルトコードは、エラー番号とエラーメッセージを含み、これにより、エラーの種類と具体的なエラーの内容を判断することができます。

1.4.4　属性操作方法

NQSV/API では、バッチサーバとの間で、キュー、リクエスト、ジョブ等に関する各種の情報の設定、参照を行うことができます。

このとき、バッチサーバとの間で設定、参照を行う情報の単位を属性といいます。

キュー、リクエスト等、バッチサーバが管理している各種の対象物に対する属性の一覧に関しては、第 4 章.属性一覧 を参照してください。

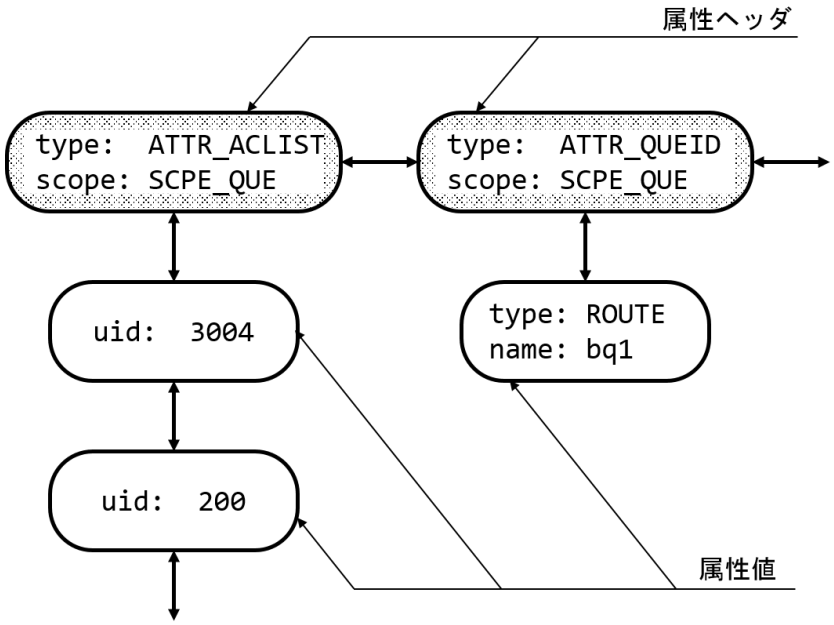
また、各属性にはその属性が適用される範囲を示すスコープが付随しており、スコープの識別子は以下が定義されています。

スコープ	スコープ識別子
バッチサーバ	SCPE_BSV
スケジューラ	SCPE_SCH
ジョブサーバ	SCPE_JSV
実行ホスト	SCPE_HST
キュー	SCPE_QUE

リクエスト	SCPE_REQ
ジョブ	SCPE_JOB
VEノード	SCPE_VENODE
プロセス	SCPE_PRC
ノードグループ	SCPE_NGRP
ワークフロー	SCPE_WFL

各種の属性に関する参照、設定／変更等の操作を行うには、操作を行う対象の属性に関する属性リストを作成し、属性リストを各種の属性操作用関数に渡す必要があります。

属性リストの作成には、NQSalist()関数を使用します。属性リストは、下図のように、属性の種類を表す属性ヘッダがつながったデータ構造を持ち、各属性ヘッダは、属性に対する値（属性値）を保持することができます。属性リストの種類によっては属性値を複数持つ場合があり、これらはリスト構造で連結されます。このような属性値の連結をチェインと呼びます。



NQSalist()で作成した属性リストのデータはヒープ領域に動的に作成されるため、使用後はNQSafree()関数で解放する必要があります。

属性の参照、設定／変更は、属性リストを使用して以下の手順で NQSV/API 関数を使用します。

- 属性の参照

1. 参照したい属性に関する属性リストを作成（NQSalist()）
2. 参照対象に対応する属性操作関数を使用（ATTROP_GET モード）して、バッチサーバから属性リスト上へ属性値を取得(NQSattrXXX())

3. 属性リストから属性値を参照（NQSaref()）

- 属性の設定／変更

1. 変更したい属性に関する属性リストを作成（NQSalist()）
2. 属性リストに設定／変更をする属性値をセット（NQSaadd()）
3. 属性操作関数を使用（ATTROP_SET モード）して、属性リストにセットした属性と属性値をバッチサーバへ送信し設定（NQSattrXXX()）

1.4.5　イベント取得方法

NQSV/API により、バッチサーバから各種のイベントの情報を取得するには、NQSconnect()でバッチサーバと接続した際に返却される、バッチサーバと接続したソケットのファイルディスクリプタを使用して、以下の手順で NQSV/API 関数を使用します。

1. NQSconnect()によりバッチサーバと接続
2. NQSevflt()関数により、バッチサーバから取得したいイベントの種別を指定
3. 1 で返却されるファイルディスクリプタへの通信の受信待ち合わせ（select(2)、poll(2)等を使用してソケットへの入力を待ち合わせます。）
4. ソケットへの入力を検知したら、NQSevent()関数で、受信したイベントの内容の読み込み
5. NQSevent()呼び出し時に指定した nqs_event 構造体に格納されているイベントのデータを処理
6. 3 へ戻ってイベントの待ち合わせ

第2章　　リクエストの状態遷移

2.1　キュー内での状態遷移

次ページの図は、バッチキューと会話キューのリクエストが取り得る状態と状態間の遷移関係を示す図です。

実線四角の状態はリクエストが実際に取り得る状態を表しており、qstat 等で表示されるリクエスト状態と一致します。

点線四角の状態は状態遷移を完結させるために導入した仮想的な状態であり、実際にリクエストがこれらの状態になることはありません。

矢印は状態遷移の方向を示しており、矢印に対して状態遷移理由が記載されています。

キュー内でリクエストが状態遷移を起すと、API クライアントに対してリクエスト状態系イベントが発行されます。

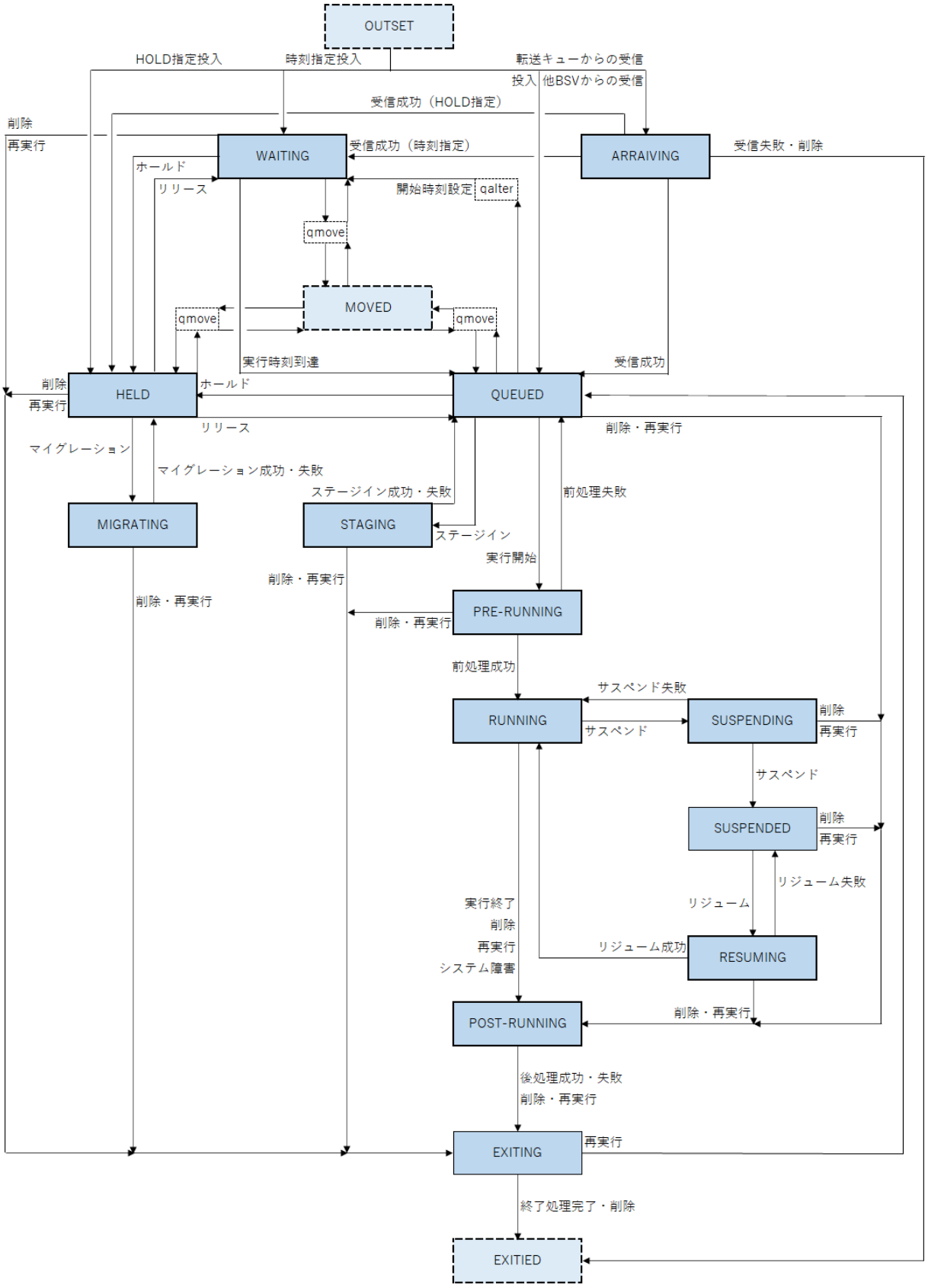
この API イベントには、

現在のリクエスト状態

直前のリクエスト状態

状態遷移理由

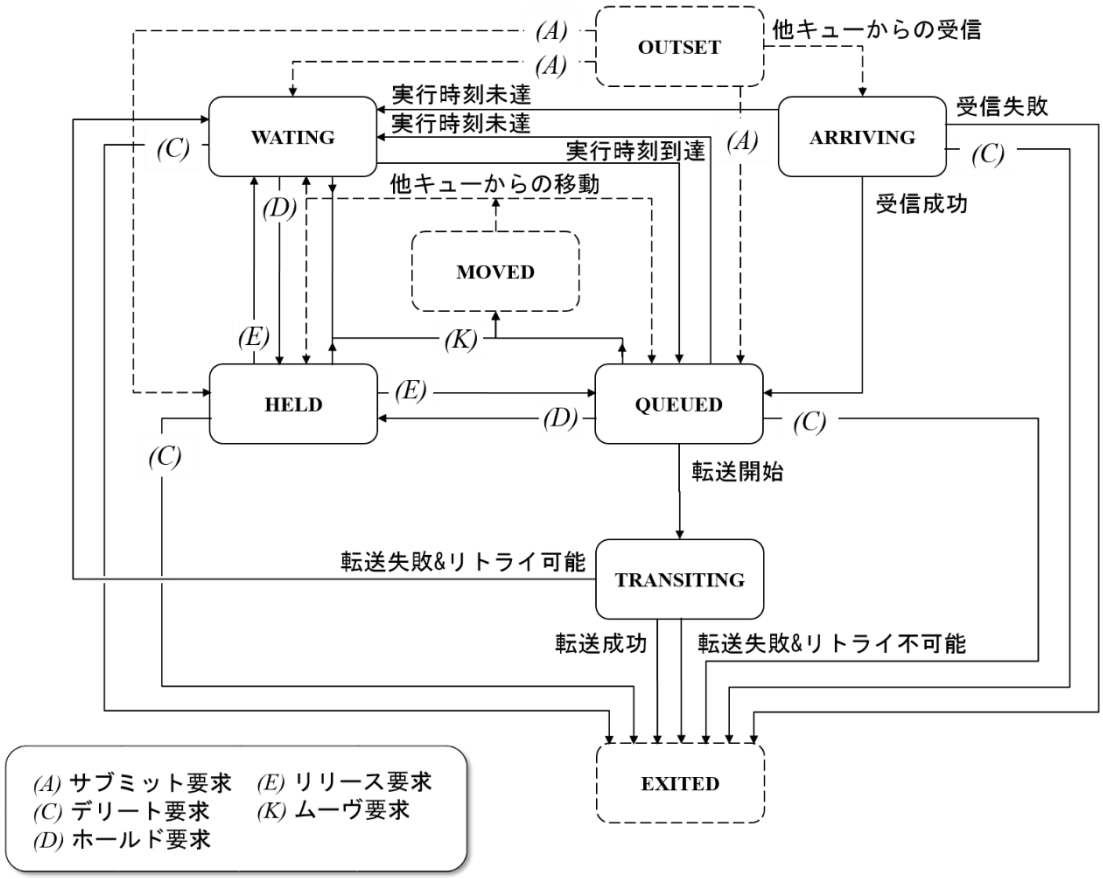
が含まれており、API クライアントはこの情報を元に該当リクエストの状態遷移をトレースすることができます。


OUTSET
HOLD指定投入
時刻指定投入
転送キューからの受信
投入
他BSVからの受信
受信成功（HOLD指定）
削除
再実行
WAITING
受信成功（時刻指定）
ARRAIVING
受信失敗・削除
ホールド
リリース
開始時刻設定
qalter
qmove
MOVED
qmove
qmove
実行時刻到達
受信成功
削除
再実行
HELD
ホールド
QUEUED
リリース
削除・再実行
マイグレーション
マイグレーション成功・失敗
ステージイン成功・失敗
前処理失敗
MIGRATING
STAGING
ステージイン
実行開始
削除・再実行
削除・再実行
PRE-RUNNING
削除・再実行
前処理成功
サスペンド失敗
RUNNING
サスペンド
SUSPENDING
削除
再実行
サスペンド
SUSPENDED
削除
再実行
リジューム失敗
実行終了
削除
再実行
システム障害
リジューム
リジューム成功
RESUMING
削除・再実行
POST-RUNNING
後処理成功・失敗
削除・再実行
EXITING
再実行
終了処理完了・削除
EXITIED

2.2　転送キュー内での状態遷移

下図は、転送キュー内のリクエストが取り得る状態をまとめた図です。転送キュー内で発生したリクエスト状態遷移に関する API イベントは生成されません。

転送キュー内でのバッチリクエスト状態遷移



2.3　リクエスト状態の詳細

リクエストの状態には以下の種類があります。

ARRIVING 状態

転送キュー経由で転送されてきたリクエストを受信しています。

TRANSITING 状態

他のキューへリクエストを転送しています。

WAITING 状態

実行時刻に到達するのを待ち合わせています。

QUEUED 状態

実行待ちの状態です。スケジュールの対象となります。

STAGING 状態

リクエストのジョブの生成、およびクライアントホストから実行ホストへのステージング対象ファイルの転送を行っています。

PRE-RUNNING 状態

リクエスト内のジョブ実行の前処理を行なっています。

ジョブ実行に必要な情報を各ジョブサーバへ転送し、全てのスレーブジョブを起動した後、最後にマスタージョブを起動します。
前処理は段階的に行われ、途中でエラーが発生した場合、その時点までの処理を逆順に辿ってキャンセルした後、QUEUEDへ戻ります。

RUNNING 状態

リクエスト内のジョブが実行中です。

MPIジョブの場合は、マスタージョブの終了を検出した時点でPOST-RUNNINGへ遷移します。スレーブジョブの終了は状態に影響を与えませんので、仮に全スレーブジョブが終了したとしても、マスタージョブが実行中である限りRUNNING状態が維持されます。
分散ジョブの場合は、全てのジョブが終了した時点でPOST-RUNNINGへ遷移します。

POST-RUNNING 状態

リクエスト内のジョブ実行後の後処理を行っています。

EXITING 状態

リクエスト内のジョブの標準出力／エラー出力、ステージング対象ファイルを実行ホストからクライアントホストへ転送しています。

CHKPNTING 状態

継続型チェックポイントの採取を行っています。

リスタートファイル採取後、ジョブは実行を続けます。

チェックポイント処理は段階的に行われ、途中でエラーが発生した場合、その時点までの処理を逆順に辿ってキャンセルした後RUNNINGへ戻ります。

HOLDING 状態

終了型チェックポイントの採取を行っています。

リスタートファイル採取後、ジョブは終了します。

チェックポイント処理は段階的に行われ、途中でエラーが発生した場合、その時点までの処理を逆順に辿ってキャンセルした後RUNNINGへ戻ります。

HELD 状態

リクエストはスケジューリング対象から外されています。

スケジューラからのラン要求またはリスタート要求は受け付けません。

ジョブの実行中（RUNNING状態）にホールドされた(終了型チェックポイントの採取が行われた）場合に、HELD 状態でリスタートファイルが生成されています。

RESTARTING 状態

以前に採取されたチェックポイントからのリスタート処理を行っています。

リスタート処理は段階的に行われ、途中でエラーが発生した場合、その時点までの処理を逆順に辿ってキャンセルした後QUEUEDへ戻ります。

SUSPENDING 状態

対象リクエスト内のジョブの停止処理を行い、全ジョブが停止するのを待ち合わせている状態です。

SUSPENDED 状態

対象リクエスト内の全てのジョブが停止しています。

RESUMING 状態

対象リクエスト内のジョブの実行再開処理を行い、全ジョブが実行を再開するのを待ち合わせている状態です。

MIGRATING 状態

対象リクエスト内に、あるジョブサーバの管理下から別のジョブサーバへ移動中のジョブがある状態です。

以下は、仮想的な状態です。

EXITED 状態

リクエストが消滅した状態で、状態遷移の終点として定義される仮想的な状態です。

MOVED 状態

リクエストのキュー間移動が完了した状態で、移動前の終点／移動後の始点として定義される仮想的な状態です。

MOVEDの前後は常に同一のリクエスト状態になります。例えば、HELD状態のリクエストを別のキューへ移動した場合、移動先での最初の状態もHELDになります。QUEUED、WAITINGに関しても同様です。

2.4　リクエストのストール

ジョブが実行ホスト上に存在している状態で、そのジョブを管理しているジョブサーバとバッチサーバ間のリンクが途切れた場合、バッチサーバ上では、該当ジョブを実行しているリクエストの状態遷移を一時的に停止し、ジョブサーバとのリンクが再開するのを待ちます。この状態をリクエストのストールと呼びます。ストールしているリクエストに対する操作は、削除またはリランのみが実行可能です。

第3章　API イベント

3.1　APIイベントの種類

イベントタイプ	イベント識別子	参照メンバ	発生タイミング
バッチサーバ系イベント			
NQSEVT_BSV	NQSEVT_BSV_LINKDOWN	(none)	APIリンクが切断された
実行ホスト状態系イベント			
NQSEVT_HST	NQSEVT_HST_ACTIVE	evt_hst	実行ホストが稼働状態となった
	NQSEVT_HST_INACTIVE		実行ホストが停止状態となった
ジョブサーバ系イベント			
NQSEVT_JSV	NQSEVT_JSV_LINKUP	evt_jsv	ジョブサーバとのリンクが確立した
	NQSEVT_JSV_LINKDOWN		ジョブサーバとのリンクが切断された
	NQSEVT_JSV_ATTACH		ジョブサーバがバッチサーバに登録された
	NQSEVT_JSV_DETACH		ジョブサーバがバッチサーバから登録削除された
	NQSEVT_JSV_BOUND		ジョブサーバが初めてキューにバインドされた
	NQSEVT_JSV_UNBOUND		ジョブサーバが最後のキューからアンバインドされた
キュー状態系イベント			
NQSEVT_QST	NQSEVT_QST_ACTIVE	evt_qst	キューが実行可能状態になった
	NQSEVT_QST_INACTIVE		キューが実行禁止状態になった
	NQSEVT_QST_ENABLE		キューが投入可能状態になった
	NQSEVT_QST_DISABLE		キューが投入禁止状態になった
	NQSEVT_QST_CREATE		キューが作成された
	NQSEVT_QST_DELETE		キューが削除された
	NQSEVT_QST_BINDJSV		キューにジョブサーバが接続された (最初に接続した場合のみ)

	NQSEVT_QST_BINDSCH		キューにスケジューラが接続された
	NQSEVT_QST_BINDNGRP		キューにノードグループが接続された
	NQSEVT_QST_UNBINDJSV		キューからジョブサーバが切り放された
	NQSEVT_QST_UNBINDSCH		キューからスケジューラが切り放された
	NQSEVT_QST_UNBINDNGRP		キューからノードグループが切り放された
キュー属性系イベント			
NQSEVT_QAT	NQSEVT_QAT_RESLIM	evt_qat	キューの資源制限値が変更された
	NQSEVT_QAT_PRIORITY		キュー・プライオリティが変更された
	NQSEVT_QAT_RSTFDIR		リスタートファイル格納ディレクトリが変更された
リクエスト状態系イベント			
NQSEVT_RST	NQSEVT_RST_ARRIVING	evt_rst	リクエストがARRIVING状態になった
	NQSEVT_RST_WAITING		リクエストがWAITING状態になった
	NQSEVT_RST_QUEUED		リクエストがQUEUED状態になった
	NQSEVT_RST_STAGING		リクエストがSTAGING状態になった
	NQSEVT_RST_PRERUNNING		リクエストがPRERUNNING状態になった
	NQSEVT_RST_RUNNING		リクエストがRUNNING状態になった
	NQSEVT_RST_POSTRUNNING		リクエストがPOSTRUNNING状態になった
	NQSEVT_RST_EXITING		リクエストがEXITING状態になった
	NQSEVT_RST_EXITED		リクエストがEXITED状態になった
	NQSEVT_RST_CHKPNTING		リクエストがCHKPNTING状態になった
	NQSEVT_RST_HOLDING		リクエストがHOLDING状態になった
	NQSEVT_RST_HELD		リクエストがHELD状態になった
	NQSEVT_RST_RESTARTING		リクエストがRESTARTING状態になった
	NQSEVT_RST_SUSPENDING		リクエストがSUSPENDING状態になった
	NQSEVT_RST_SUSPENDED		リクエストがSUSPENDED状態になった
	NQSEVT_RST_RESUMING		リクエストがRESUMING状態になった
	NQSEVT_RST_MIGRATING		リクエストがMIGRATING状態になった
	NQSEVT_RST_MOVED		リクエストがMOVED状態になった

	NQSEVT_RST_STALLED		リクエストがストールした
	NQSEVT_RST_REVIVED		リクエストがストールから復帰した
リクエスト属性系イベント			
NQSEVT_RAT	NQSEVT_RAT_RESLIM	evt_rat	リクエストの資源制限値が変更された
	NQSEVT_RAT_ACCTCODE		リクエストのアカウントコードが変更された
	NQSEVT_RAT_EXETIME		リクエストの実行時刻が変更された
	NQSEVT_RAT_RSVTIME		リクエストの予約時刻が変更された
	NQSEVT_RAT_PRIORITY		リクエスト・プライオリティが変更された
	NQSEVT_RAT_MIGRATABL		リクエストのマイグレーション属性が変更された
	NQSEVT_RAT_HOLDABL		リクエストのホールド属性が変更された
	NQSEVT_RAT_MAILADDR		リクエストのメールアドレスが変更された
	NQSEVT_RAT_KNLPRM		リクエストのカーネルパラメータ値が変更された
ノードグループ系イベント			
NQSEVT_NGRP	NQSEVT_NGRP_CREATE	evt_ngrp	ノードグループが作成された
	NQSEVT_NGRP_DELETE		ノードグループが削除された
	NQSEVT_NGRP_ADDNODE		ノードグループにジョブサーバが追加された
	NQSEVT_NGRP_DELNODE		ノードグループからジョブサーバが削除された
テンプレート系イベント			
NQSEVT_TMPL	NQSEVT_TMPL_CREATE	evt_template	テンプレートが作成された
	NQSEVT_TMPL_DELETE		テンプレートが削除された
	NQSEVT_TMPL_MODIFY		テンプレートの内容が変更された
カスタムリソース系イベント			
NQSEVT_CRS	NQSEVT_CRS_CREATE	evt_crs	カスタムリソースが作成された
	NQSEVT_CRS_DELETE		カスタムリソースが削除された
	NQSEVT_CRS_RESCHANGED		カスタムリソースの設定が変更された

※リクエスト状態系イベント、および、リクエスト属性系イベントに関しては、パラメトリックリクエストの場合、各サブリクエスト単位でイベントが通知されます。

第4章　　属性一覧

本節では、NQSV で取り扱う属性を説明します。一覧の参照欄には属性を参照する際に必要な API 権限が、変更欄には NQSattr*xxx*(3)による属性変更の可否、および変更に必要な API 権限が記載されています。

API 権限は、以下の記号で記されています。

記号	説明
MGR	PRIV_MGR（管理者権限）以上のAPI権限が必要です。
OPE	PRIV_OPE（操作員権限）以上のAPI権限が必要です。
USR	PRIV_USR（一般利用者権限）以上のAPI権限が必要です。
USR(R)	PRIV_USR（一般利用者権限）以上のAPI権限が必要です。 但し、PRIV_USRの場合はリクエスト/ジョブの所有者でなければなりません。 また、PRIV_GMGR の場合はリクエスト/ジョブのグループが管理しているグループでなければなりません。
USR(Q)	PRIV_USR（一般利用者権限）以上のAPI権限が必要です。 但し、PRIV_USRの場合はキューへのアクセスが許可されている必要があります。 また、PRIV_GMGR の場合はキューのアクセス制限で管理しているグループが許可されている必要があります。
×	参照/変更は禁止されています。

APIで取り扱うデータ型については次章を参照してください。

4.1　バッチサーバ属性

属性名	属性種別	型	参照	変更	チェイン	説明	スコープ B S V
バージョン番号	ATTR_VERNO	char *	USR	×	×	バッチサーバのバージョン [最大長: NQS_LEN_VERSION]	○
ホスト名	ATTR_HOSTNAME	char *	USR	×	×	バッチサーバホストのホスト名 [最大長: NQS_LEN_HOSTNAME]	○
マシンID	ATTR_MID	int	USR	×	×	バッチサーバホストのマシンID	○
ログファイル名	ATTR_LOGPATH	char *	USR	OPE	×	NQSログファイルのフルパス [最大長: NQS_LEN_FILENAME]	○
ログレベル	ATTR_LOGLEVEL	nqs_range	USR	OPE	×	NQSログの出力レベル	○
ログファイル保存数	ATTR_LOGFGEN	nqs_range	USR	OPE	×	過去のNQSログファイルの保存数	○
ログファイルの最大長	ATTR_LOGFLEN	int	USR	OPE	×	NQSログファイルの最大長[バイト単位、0の場合は無制限]	○
ライセンス情報	ATTR_LICINFO	nqs_license	USR	×	○	各ライセンスの使用状況	○
ハートビート間隔	ATTR_HBINTVAL	int	USR	OPE	×	バッチサーバ、ジョブサーバ間のハートビート送信間隔[秒単位、0の場合は送信停止]	○
負荷情報採取間隔	ATTR_LHINTVAL	int	USR	OPE	×	実行ホストの負荷情報を採取する間隔[秒単位、0の場合は採取停止]	○
ジョブ資源量採取間隔	ATTR_JUINTVAL	int	USR	OPE	×	ジョブの資源使用量を採取する間隔[秒単位、0の場合は採取停止]	○

転送キュー実行数制限値	ATTR_ROURLIM	nqs_range	USR	OPE	×	バッチサーバ全体での転送キュー同時実行数の上限	○
転送キューリトライ間隔	ATTR_ROURTYIVAL	int	USR	OPE	×	転送キューが次に転送を試みるまでの待ち時間(秒単位)	○
転送キューリトライ可能期間	ATTR_ROURTYSPAN	nqs_range	USR	OPE	×	転送キューがリトライ動作を繰り返すことが可能な期間(秒単位)	○
ネットワークキュー実行数制限値	ATTR_NETRLIM	nqs_range	USR	OPE	×	バッチサーバ全体でのネットワークキュー同時実行数の上限	○
ネットワークキューリトライ間隔	ATTR_NETRTYIVAL	int	USR	OPE	×	ネットワークキューが次に転送を試みるまでの待ち時間(秒単位)	○
ネットワークキューリトライ可能期間	ATTR_NETRTYSPAN	nqs_range	USR	OPE	×	ネットワークキューがリトライ動作を繰り返すことが可能な期間(秒単位)	○
ACCTサーバのホスト名	ATTR_ACCTHOST	char *	USR	OPE	×	ACCTサーバの実行ホスト名	○
ACCTサーバのTCPポート番号	ATTR_ACCTPORT	int	USR	OPE	×	ACCTサーバがNQSVと通信する際に使用するTCPポート番号	○
ジョブアカウントファイルの出力ディレクトリ	ATTR_JACCT_DIR	char *	USR	MGR	×	BSV上のジョブアカウントファイル出力ディレクトリ 既定値：/var/opt/nec/nqsv/acm/jacct	○
予算超過チェック	ATTR_NQS_BUDGETCHECK	int	USR	MGR	×	アカウンティングサーバによる予算超過チェックを行うか否かの設定 0: 予算超過チェックを実施しない(OFF)（既定値） 1: リクエストの予算超過チェックを実施 2: 予約の予算超過チェックを実施 3: リクエストと予約の両方の予算超過チェックを実施	○
リクエストアカウンテイングスイッチ	ATTR_RACCTSW	int	USR	MGR	×	リクエストアカウンティングの出力のON/OFF	○

						0: OFF（既定値） 1: ON	
リクエストアカウント ファイル	ATTR_RACCTPATH	char *	USR	MGR	×	リクエストアカウントファイル名 既定値： /var/opt/nec/nqsv/bsv/account/reqacct	○
予約アカウンティング スイッチ	ATTR_RESERVATION_ACCT	int	USR	MGR	×	予約アカウンティングの出力のON/OFF 0: OFF（既定値） 1: ON	○
予約アカウントファイル	ATTR_RESERVATION_ACCT_FILE	char *	USR	MGR	×	予約アカウントファイル名 既定値： /var/opt/nec/nqsv/acm/rsvacct/rsvacct	○
グループ指定機能	ATTR_SPCFYGRP	int	USR	OPE	×	リクエストのグループ指定実行機能が有効か無効かを示す（0：無効　1：有効）	○
ステージングファイルの 絶対パス指定	ATTR_ABSLT_EXEPATH	int	USR	MGR	×	ステージングファイルの絶対パス指定の可否を示す （0：指定不可　1：指定可能）	○
リクエスト投入数制限値関連							
同時投入数制限値	ATTR_GBLSBLM	nqs_range	USR	OPE	×	バッチサーバ単位のリクエスト投入数の上限値	○
ユーザ投入数制限値	ATTR_USRSBLM	int	USR	OPE	×	バッチサーバ単位の１ユーザ当たりのリクエスト投入数の上限値（0 以上の整数。0 は無制限を表します。）	○
グループ投入数制限値	ATTR_GRPSBLM	int	USR	OPE	×	バッチサーバ単位の１グループ当たりのリクエスト投入数の上限値（0 以上の整数。0 は無制限を表します。）	○
ユーザ名指定の投入数制限値	ATTR_USRSBLM_N	nqs_int_u	USR	OPE	○	バッチサーバ単位のユーザ名指定での投入数制限値 （0以上の整数。0は無制限を表します。）	○

グループ名指定の投入数制限値	ATTR_GRPSBLM_N	nqs_int_g	USR	OPE	○	バッチサーバ単位のグループ名指定での投入数制限値 （0以上の整数。0は無制限を表します。）	○
サブリクエスト生成数制限値関連							
同時生成サブリクエスト数制限値	ATTR_GBLSUBREQENT	nqs_range	USR	OPE	×	バッチサーバ単位の同時生成サブリクエスト数の上限値 (1~999999の範囲で指定。既定値は100。)	○
テンプレート関連							
OpenStack用テンプレート	ATTR_OSTEMPLATE	nqs_ostemplate	USR	OPE	○	OpenStack用のテンプレート定義	○
コンテナ用テンプレート	ATTR_COTEMPLATE	nqs_cotemplate	USR	OPE	○	コンテナ用のテンプレート定義	○
使用中のリクエスト数	ATTR_TEMP_REQS	nqs_temp_reqs	USR	×	○	テンプレートを使用しているリクエスト数	○

4.2　スケジューラ属性

属性名	属性種別	型	参照	変更	チェイン	説明	スコープ
							SCH
スケジューラID	ATTR_SCHID	nqs_schid	USR	×	×	スケジューラID	○
バージョン番号	ATTR_VERNO	char *	USR	×	×	スケジューラのバージョン[最大長: NQS_LEN_VERSION]	○
スケジューラ名	ATTR_SCHNAME	char *	USR	×	×	スケジューラ名[最大長: NQS_LEN_SCHNAME]	○
ホストID	ATTR_HSTID	nqs_hid	USR	×	×	スケジューラを実行しているホスト	○
スケジューラ情報	ATTR_SCHDMSG	char *	USR	SCH	×	スケジューラからのメッセージが格納されます。[最大長: NQS_LEN_SCHDMSG]	○
スケジューリングインターバル	ATTR_SCHINTERVAL	int	USR	SCH	×	スケジューリングインターバル（秒）（JobManipulatorのみ）	○
スケジューリング状態	ATTR_SCHSTAT	int	USR	SCH	×	スケジューリング状態 SCHST_START: スケジューリング実施中 SCHST_STOP:　スケジューリング停止中	○

4.3　ジョブサーバ属性

属性名	属性種別	型	参照	変更	チェイン	説明	スコープ JSV
ジョブサーバID	ATTR_JSVID	nqs_jsvid	USR	×	×	ジョブサーバID	○
バージョン番号	ATTR_VERNO	char *	USR	×	×	ジョブサーバのバージョン [最大長: NQS_LEN_VERSION]	○
ジョブサーバ名	ATTR_JSVNAME	char *	USR	×	×	ジョブサーバ名 [最大長: NQS_LEN_JSVNAME]	○
ジョブサーバ状態	ATTR_JSVST	nqs_jsvst	USR	×	×	ジョブサーバの状態	○
実行ホストID	ATTR_HSTID	nqs_hid	USR	×	×	ジョブサーバを実行している実行ホスト	○
キューID	ATTR_QUEID	nqs_qid	USR	×	○	接続しているキュー	○
RSG番号	ATTR_RSGNO	nqs_range	USR	×	○	管理下のジョブがアサインされるRSG番号 ※接続しているキューのRSGに応じてチェインで取得可能	○
マイグレーションパラメータ	ATTR_MIGPRM	nqs_migprm	USR	OPE	×	ジョブマイグレーション性能チューニング用パラメータ	○
スケジューラ情報	ATTR_SCHDMSG	char *	USR	SCH	×	スケジューラからのメッセージが格納されます。 [最大長: NQS_LEN_SCHDMSG]	○
JMライセンス状態	ATTR_JMLICENSE	int	USR	×	×	JMライセンス使用中（True）、非使用中（False）を示す	○
所属ノードグループID	ATTR_NGRPID	nqs_ngrpid	USR	×	○	所属するノードグループID	○
実行ホストのタイ	ATTR_HSTTYPE	int	USR	×	×	実行ホストのタイプ	○

プ						EXECUTION（0）またはBAREMETAL（1）	
定義済GPU数	ATTR_DEFINED_NGPUS	int	USR	×	×	ベアメタルサーバの場合、搭載GPU数定義値	○
定義済メモリ量	ATTR_DEFINED_MEMORY	nqs_rsgres	USR	×	×	ベアメタルサーバの場合、搭載メモリ量定義値	○
定義済CPU数	ATTR_DEFINED_NCPUS	int	USR	×	×	ベアメタルサーバの場合、搭載CPU数定義値	○
OpenStack用テンプレート	ATTR_OSTEMPLATE	nqs_ostemplate	USR	×	×	ベアメタルサーバの場合、起動時に使用したテンプレート	○
RSG情報	ATTR_RSGINFO	nqs_rsginfo	USR	×	○	JSVホストの各RSGのメモリ情報、スワップ情報、利用可能CPU台数情報、ロードアベレージ情報、GPU台数情報、VEノード台数情報	○
HCA障害検知時の対処	ATTR_HCACHK	int	USR	OPE	×	HCA障害検知時の対処方法を保持する。 HCACHK_OFF HCACHK_DOWN HCACHK_UNBIND	○
ホスト負荷情報関連							
メモリ情報	ATTR_RBSPMEM	nqs_rsgres	USR	×	×	実行ホスト単位の物理メモリサイズに関する情報	○
スワップ情報	ATTR_RBSPSWAP	nqs_rsgres	USR	×	×	実行ホスト単位のスワップサイズに関する情報	○
CPU台数情報	ATTR_RBCPUNM	nqs_rsgres	USR	×	×	実行ホスト単位のCPU実装台数に関する情報	○
ロードアベレージ情報	ATTR_RBLDAVG	nqs_rsgavg	USR	×	×	実行ホスト単位のロードアベレージに関する情報	○
CPUアベレージ情報	ATTR_RBCPUAVG	nqs_rsgavg	USR	×	×	実行ホスト単位のCPUアベレージ(CPU使用率×CPU実装台数)に関する情報	○

4.4　実行ホスト属性

属性名	属性種別	型	参照	変更	チェイン	説明	スコープ
							実行ホスト
実行ホストID	ATTR_HSTID	nqs_hid	USR	×	×	実行ホストID	○
オペレーティングシステム名	ATTR_SYSNAME	char *	USR	×	×	実行ホストのOS名[最大長：NQS_LEN_UTSNAME]	○
OSバージョン番号	ATTR_VERNO	char *	USR	×	×	実行ホストのOSバージョン名[最大長：NQS_LEN_UTSNAME]	○
OSリリース番号	ATTR_RELNO	char *	USR	×	×	実行ホストのOSリリース名[最大長：NQS_LEN_UTSNAME]	○
ハードウェア名	ATTR_HWNAME	char *	USR	×	×	実行ホストのハードウェア名[最大長：NQS_LEN_UTSNAME]	○
ジョブサーバID	ATTR_JSVID	nqs_jsvid	USR	×	×	ジョブサーバID	○
ノード管理エージェント	ATTR_NODEAGENT	nqs_hid	USR	×	×	接続中のノード管理エージェントホスト（ノード管理エージェント接続時のみ）	○
実行ホスト状態	ATTR_HST	nqs_hst	USR	×	×	実行ホストの稼働状態 HOSTST_ACTIVE　稼働状態 HOSTST_INACTIVE　停止状態	○
実行ホストのタイプ	ATTR_HSTTYPE	int	USR	×	×	実行ホストのタイプ EXECUTION（0）またはBAREMETAL（1）	○
定義済GPU数	ATTR_DEFINED_NGPUS	int	USR	×	×	ベアメタルサーバの場合、搭載GPU数定義値	○
定義済メモリ量	ATTR_DEFINED_MEMORY	nqs_rsgres	USR	×	×	ベアメタルサーバの場合、搭載メモリ量定義値	○
定義済CPU数	ATTR_DEFINED_NCPUS	int	USR	×	×	ベアメタルサーバの場合、搭載CPU数定義値	○

OpenStack用テンプレート	ATTR_OSTEMPLATE	nqs_ostemplate	USR	×	×	ベアメタルサーバの場合、起動時に使用したテンプレート	○
ソケットリソース情報	ATTR_SOCKETINFO	nqs_socket	USR	×	○	ソケット毎のリソース量と使用量の情報	○
CPUSET情報	ATTR_CPUSETINFO	nqs_cpuset	USR	×	○	CPUSET毎のリソース情報	○
GPUの詳細情報	ATTR_GPUINFO	nqs_gpuinfo	USR	×	○	実行ホストに搭載されている各GPUの詳細情報	○
VEノードの詳細情報	ATTR_VEINFO	nqs_veinfo	USR	×	○	実行ホストに搭載されている各VEノードの詳細情報	○
RSG情報	ATTR_RSGINFO	nqs_rsginfo	USR	×	×	実行ホストのメモリ情報、スワップ情報、CPU台数情報、ロードアベレージ情報の合計値、GPU台数情報、VEノード台数情報 ※取得可能メンバは、以下の属性と同様	○
ホスト負荷情報関連							
メモリ情報	ATTR_RBSPMEM	nqs_rsgres	USR	×	×	実行ホスト単位の物理メモリサイズに関する情報	○
スワップ情報	ATTR_RBSPSWAP	nqs_rsgres	USR	×	×	実行ホスト単位のスワップサイズに関する情報	○
CPU台数情報	ATTR_RBCPUNM	nqs_rsgres	USR	×	×	実行ホスト単位のCPU実装台数に関する情報	○
GPU台数情報	ATTR_RBGPUNM	nqs_rsgres	USR	×	×	実行ホスト単位のGPU実装台数に関する情報	○
ロードアベレージ情報	ATTR_RBLDAVG	nqs_rsgavg	USR	×	×	実行ホスト単位のロードアベレージに関する情報	○
CPUアベレージ情報	ATTR_RBCPUAVG	nqs_rsgavg	USR	×	×	実行ホスト単位のCPUアベレージ(CPU使用率×CPU実装台数)に関する情報	○
VEノード台数情報	ATTR_RBVENM	nqs_rsgres	USR	×	×	実行ホスト単位のVEノード実装台数に関する情報。maximumにはVI上に搭載されている全VE数がセットされる。initialには、その時点で使用可能な（statusおよびos statusが両方ともにONLINEの）VEノード数がセットされる。	○

4.5　キュー属性（バッチキュー、会話キュー）

属性名	属性種別	型	参照	変更	チェイン	説明	スコープ						
							キュー	リクエスト	実行ホスト	ジョブサーバ	ジョブ	プロセス	VNノード
キューID	ATTR_QUEID	nqs_qid	USR	×	×	キューID	○						
キュー状態	ATTR_QUEST	nqs_qst	USR(Q)	OPE	×	現在のキュー状態	○						
プライオリティ	ATTR_PRIORITY	nqs_range	USR(Q)	OPE	×	キュー・プライオリティ	○						
スケジューラID	ATTR_SCHID	nqs_schid	USR(Q)	×	×	接続しているバッチスケジューラのスケジューラID	○						
スケジューラ名	ATTR_SCHNAME	char *	USR(Q)	×	×	接続しているバッチスケジューラの名前[最大長：NQS_LEN_SCHNAME]	○						
チェックポイント属性	ATTR_CHKPNTABL	nqs_range	USR(Q)	OPE	×	定期的チェックポイント間隔を表す0または正の整数値(分単位)。0の場合は定期的チェックポイントを実行しない。（会話キュー無し）		○					

ホールド特権	ATTR_HOLDPRIV	int	USR(Q)	OPE	×	RUNNING中のリクエストをホールドする際に必要なクライアント権限 （会話キュー無し） PRIV_SCH: スケジューラ権限以上 PRIV_MGR: 管理者権限以上 PRIV_OPE: 操作員権限以上 PRIV_GMGR:グループ管理者権限以上 PRIV_SPU: 特別利用者権限以上 PRIV_USR: 一般ユーザ権限以上 PRIV_NON: どの権限でも可		○					

サスペンド特権	ATTR_SUSPRIV	int	USR(Q)	OPE	×	RUNNING中のリクエストをサスペンドする際に必要なクライアント権限 PRIV_SCH:　スケジューラ権限以上 PRIV_MGR:　管理者権限以上 PRIV_OPE:　操作員権限以上 PRIV_GMGR:グループ管理者権限以上 PRIV_SPU:　特別利用者権限以上 PRIV_USR:　一般ユーザ権限以上 PRIV_NON:　どの権限でも可		○					

ユーザEXIT情報	ATTR_USEREXIT	nqs_uexit	USR(Q)	OPE	○	ジョブの実行時に起動されるユーザEXITに関する情報。 nqs_uexitのlocationで実行タイミングを指定 ・UEXLOC_PRERUN： 実行開始直前 [PRE-RUNNING状態] ・UEXLOC_PSTRUN： 実行終了直後 [POST-RUNNING状態] ・UEXLOC_HLDING： 終了型チェックポイント完了直後 [HOLDING状態] （会話キュー無し） ・UEXLOC_RSTING： リスタート処理実行前 [RESTARTING状態] （会話キュー無し）					○		

投入経路制限	ATTR_RFUSSB	int	USR(Q)	OPE	○	投入経路別のリクエスト投入制限。下記経路のうち、設定されている経路からの投入を拒否します。 ・RFUSSB_QSUB qsub	qlogin (NQScrereq) ・RFUSSB_QMOV qmove (NQSmovreq) （会話キュー無し） ・RFUSSB_LCRQ ローカルからの転送 ・RFUSSB_RMRQ リモートからの転送	○						
プライオリティ範囲(MGR)	ATTR_MGRPRRNG	nqs_hilo	USR(Q)	MGR	×	NQSV管理者が指定可能なリクエスト・プライオリティの範囲 （会話キュー無し）			○				
プライオリティ範囲(OPE)	ATTR_OPEPRRNG	nqs_hilo	USR(Q)	MGR	×	NQSV操作員が指定可能なリクエスト・プライオリティの範囲 （会話キュー無し）			○				
プライオリティ範囲(SPU)	ATTR_SPUPRRNG	nqs_hilo	USR(Q)	MGR	×	特別利用者が指定可能なリクエスト・プライオリティの範囲（会話キュー無し）			○				

プライオリティ範囲(USR)	ATTR_USRPRRNG	nqs_hilo	USR(Q)	MGR	×	一般利用者が指定可能なリクエスト・プライオリティの範囲（会話キュー無し）		○					
ACLモード	ATTR_ACLMODE	int	USR(Q)	MGR	×	【ACLモードがACL_ACCESSの場合】 ユーザ名がACLユーザ名リストに含まれるか、またはグループ名がACLグループ名リストに含まれるユーザのみがキューへのアクセスが許可されます。 【ACLモードがACL_NOACCESSの場合】 ユーザ名がACLユーザ名リストに含まれるか、またはグループ名がACLグループ名リストに含まれるユーザはキューへのアクセスが禁止されます。 ユーザ名の最大長は、NQS_LEN_USERNAMEです。 グループ名の最大長は、NQS_LEN_GROUPNAMEです。	○						
ACLユーザ名リスト	ATTR_ACLUNAME	char *	USR(Q)	MGR	○								
ACLグループ名リスト	ATTR_ACLGNAME	char *	USR(Q)	MGR	○								

補助グループ名投入制限	ATTR_SUPGIDCHK	int	USR(Q)	OPE	×	補助グループ名によるリクエスト投入制限の有効/無効を切り替える。 属性値が真の場合、ACLグループ名リストによる投入制限チェックの際にリクエスト作成時の補助グループ名がチェック対象に加えられる。	○						
状態別リクエスト数	ATTR_NREQST	nqs_nreqst	USR(Q)	×	×	状態別に集計されたリクエスト数	○						
JSV自動バインド制御	ATTR_AUTOBIND	int	USR(Q)	OPE	×	JSVリンクアップ時の自動バインドを制御します。 ・属性値が真の場合: 自動バインドを行う。(既定値) ・属性値が偽の場合: 自動バインドを行わない。				○			
リスタートファイル格納パス	ATTR_RSTFDIR	char *	USR(Q)	OPE	×	リスタートファイルが格納される実行ホスト上のフルパス名 [最大長:NQS_LEN_RSTFPATH] （会話キュー無し）					○		
スケジューラ情報	ATTR_SCHDMSG	char *	USR(Q)	SCH	×	スケジューラからのメッセージが格納されます。 [最大長:NQS_LEN_SCHDMSG]	○						

生成ジョブ数範囲	ATTR_NJOBRNG	nqs_hilo	USR(Q)	OPE	×	リクエスト毎に生成可能なジョブ数の範囲		○					
ジョブサーバ数	ATTR_NJSVS	int	USR(Q)	×	×	キューにbindしているジョブサーバの総数	○						
投入サブリクエスト数制限	ATTR_SUBREQLM	int	USR(Q)	OPE	×	キュー単位の投入サブリクエスト数制限（会話キュー無し）	○						
アタッチ機能利用	ATTR_QATTACH	int	USR(Q)	MGR	×	1の場合アタッチ機能利用可能（既定値） 0の場合アタッチ機能利用不可	○						
IntelMPIプロセスマネージャ	ATTR_INTMPI_PMGR	int	USR(Q)	OPE	×	IntelMPIリクエストのプロセスマネージャの設定 ・NQSII_IMPI_HYDRA： 　hydra（既定値） ・NQSII_IMPI_MPD： 　mpd	○						
フックスクリプト制御	ATTR_HOOKFUNC	int	USR(Q)	OPE	×	フックスクリプト機能の有効/無効を切り替える 0：無効（既定値）　1：有効	○						
ユーザEXITスクリプトのタイムアウト時間	ATTR_UEXIT_TIMEOUT	int	USR(Q)	OPE	×	ユーザExitスクリプトのタイムアウト時間（秒） 0の場合、タイムアウトの監視をしない（既定値は0）	○						

ユーザPPスクリプトのタイムアウト時間	ATTR_UPP_TIMEOUT	int	USR(Q)	OPE	×	ユーザPPスクリプトのタイムアウト時間（秒） 0の場合、タイムアウトの監視をしない（既定値は300秒）	○						
カスタムリソース利用量制限情報	ATTR_QUECRINFO	nqs_quecrinfo	USR(Q)	OPE	○	キューでのカスタムリソース利用量情報（カスタムリソース名、利用量の既定値、利用量の指定範囲、利用量制御対象外指定の可否）	○						
VEノード総数制限	ATTR_TOTALVENUM	nqs_rrange	USR(Q)	OPE	×	VE総数指定投入の範囲制限		○					
同時実行VEノード台数制限	ATTR_VENUM	nqs_rrange	USR(Q)	OPE	×	同時実行可能なVEノード台数の制限					○		
既定VE搭載数	ATTR_VENUMDEFINE	int	USR(Q)	OPE	×	VE総数指定投入時に使用する論理ホスト毎のVE搭載数既定値	○						
排他実行リクエストの投入可否	ATTR_EXCLUSIVE	int	USR(Q)	MGR	×	排他実行リクエストの投入（1ノード1ジョブ）の可否を設定。 ０：投入不可（既定値） １：投入可能（既定値）		○					
ステージアウト実施	ATTR_DOSTGOUT	int	USR(Q)	OPE	×	ステージアウトを実施するかどうかの切り替え 0：実施しない 1：実施する	○						

実行不可緊急リクエストの削除	ATTR_DELURGNTREQ	int	USR(Q)	OPE	×	実行不可緊急リクエストの削除。 1:削除する 0:削除しない（既定値）	○						
VEOS の Partial Process Swapping 利用可否	ATTR_USEPPS	int	USR(Q)	OPE	×	VEOS の Partial Process Swappingの利用可否。 1:利用する（既定値） 0:利用しない	○						
会話キュー関連（会話キューのみ）													
待ち合わせ指定	ATTR_REALTIME_SCHEDULING	int	USR(Q)	OPE	×	投入されたリクエストに即時実行可能なノードが割り当てられなかった場合の動作を規定。 ・MODE_CANCEL： 投入キャンセル ・MODE_WAIT： 待ち合わせ ・MODE_MANUAL： 投入時の指定に従う	○						
強制実行シェル	ATTR_RESTRICT_SHELL	char *	USR(Q)	OPE	×	強制的に実行するシェルを指定 指定がない場合、ユーザのログインシェル、または、投入時の指定シェルを実行する	○						
アイドルタイマー	ATTR_IDLETIMER	int	USR(Q)	OPE	×	会話リクエストに設定するアイドルタイマーの既定値（分） ０の場合OFF	○						

資源制限値関連													
経過時間制限値	ATTR_ELPSTIM	nqs_rlim	USR(Q)	OPE	×	実行開始からの経過時間による制限		○				○	
CPU時間制限値	ATTR_CPUTIM	nqs_rlim	USR(Q)	OPE	×	CPU使用時間による制限					○	○	
同時実行CPU台数制限値	ATTR_CPUNUM	nqs_rnum	USR(Q)	OPE	×	同時実行可能なCPU台数の制限					○		
オープンファイル数制限値	ATTR_FILENUM	nqs_rnum	USR(Q)	OPE	×	同時オープンファイル数の制限						○	
メモリサイズ制限値	ATTR_MEMSZ	nqs_rlim	USR(Q)	OPE	×	使用可能な最大メモリサイズの制限					○	○	
データサイズ制限値	ATTR_DATASZ	nqs_rlim	USR(Q)	OPE	×	使用可能な最大データセグメントサイズの制限						○	
スタックサイズ制限値	ATTR_STACKSZ	nqs_rlim	USR(Q)	OPE	×	使用可能な最大スタックサイズの制限						○	
コアファイルサイズ制限値	ATTR_CORESZ	nqs_rlim	USR(Q)	OPE	×	生成可能な最大コアファイルサイズの制限						○	
ファイルサイズ制限値	ATTR_FILESZ	nqs_rlim	USR(Q)	OPE	×	生成可能な最大ファイルサイズの制限						○	
投入数制限値	ATTR_GBLQSBLM	nqs_range	USR(Q)	OPE	×	キュー単位のリクエスト投入数の上限値（0以上の整数。0は無制限を表します。）	○						
ユーザ投入数制限値	ATTR_USRQSBLM	int	USR(Q)	OPE	×	キュー単位の１ユーザ当たりのリクエスト投入数の上限値（0以上の整数。0は無制限を表します。）	○						

グループ投入数制限値	ATTR_GRPQSBLM	int	USR(Q)	OPE	×	キュー単位の１グループ当たりのリクエスト投入数の上限値（0以上の整数。0は無制限を表します。）	○						
VE CPU時間制限	ATTR_VECPUTIMRNG	nqs_rrange	USR(Q)	OPE	×	VEのCPU時間の制限値					○	○	○
VEメモリサイズ制限	ATTR_VEMEMSZRNG	nqs_rrange	USR(Q)	OPE	×	VEのメモリサイズの最小値、最大値、警告値、既定値					○	○	○
仮想メモリサイズ制限値	ATTR_VMEMSZ	nqs_rlim	USR(Q)	OPE	×	使用可能な最大仮想メモリサイズの制限					○	○	
同時実行GPU台数制限値	ATTR_GPUNUM	nqs_rnum	USR(Q)	OPE	×	同時実行可能なGPU台数の制限					○		
VEノード総数制限	ATTR_TOTALVENUM	nqs_rrange	USR(Q)	OPE	×	VE総数指定投入の範囲制限		○					
同時実行VEノード台数制限	ATTR_VENUM	nqs_rrange	USR(Q)	OPE	×	同時実行可能なVEノード台数の制限					○		
既定VE搭載数	ATTR_VENUMDEFINE	int	USR(Q)	OPE	×	VE総数指定投入時に使用する論理ホスト毎のVE搭載数既定値	○						
CPU 時間制限範囲	ATTR_CPUTIMRNG	nqs_rrange	USR(Q)	OPE	×	CPU 使用時間による制限範囲					○		
同時実行CPU 台数制限範囲	ATTR_CPUNUMRNG	nqs_rrange	USR(Q)	OPE	×	同時実行可能なCPU 台数の制限範囲					○		
メモリサイズ制限範囲	ATTR_MEMSZRNG	nqs_rrange	USR(Q)	OPE	×	使用可能な最大メモリサイズの制限範囲					○		
仮想メモリサイズ制限範囲	ATTR_VMEMSZRNG	nqs_rrange	USR(Q)	OPE	×	使用可能な最大仮想メモリサイズの制限範囲					○		
同時実行GPU 台数制限範囲	ATTR_GPUNUMRNG	nqs_rrange	USR(Q)	OPE	×	同時実行可能なGPU 台数の制限範囲					○		

HCAポート数	ATTR_HCA	nqs_hca	USR(Q)	OPE	×	使用可能なHCAポート数の制限					○		
ユーザ名指定の制限値													
ユーザ名指定の生成ジョブ数範囲	ATTR_NJOBRNG_U	nqs_hilo_u	USR	OPE	○	ユーザ名指定で設定する、リクエストごとに生成可能なジョブ数の範囲		○					
ユーザ名指定の経過時間制限値	ATTR_ELPSTIM_U	nqs_rlim_u	USR	OPE	○	ユーザ名指定で設定する、実行開始からの経過時間による制限		○					
ユーザ名指定の投入数制限値	ATTR_USRQSBLM_N	nqs_int_u	USR	OPE	○	キュー単位のユーザ名指定でのリクエスト投入数の制限値	○						
グループ名指定の制限値													
グループ名指定の生成ジョブ数範囲	ATTR_NJOBRNG_G	nqs_hilo_g	USR	OPE	○	グループ名指定で設定する、リクエストごとに生成可能なジョブ数の範囲		○					
グループ名指定の経過時間制限値	ATTR_ELPSTIM_G	nqs_rlim_g	USR	OPE	○	グループ名指定で設定する、実行開始からの経過時間による制限		○					
グループ名指定の投入数制限値	ATTR_GRPQSBLM_N	nqs_int_g	USR	OPE	○	キュー単位のグループ名指定でのリクエスト投入数の制限値	○						
カーネルパラメータ関連													
RSG番号	ATTR_RSGNO	nqs_range	USR(Q)	OPE	×	リソースシェアリンググループ番号						○	
ナイス値	ATTR_NICE	nqs_range	USR(Q)	OPE	×	ナイス値						○	

4.6　転送キュー属性

属性名	属性種別	型	参照	変更	チェイン	説明	スコープ キュー
キューID	ATTR_QUEID	nqs_qid	USR	×	×	転送キューのキューID	○
キュー状態	ATTR_QUEST	nqs_qst	USR(Q)	OPE	×	現在のキュー状態	○
プライオリティ	ATTR_PRIORITY	nqs_range	USR(Q)	OPE	×	キュー・プライオリティ	○
同時実行数制限値	ATTR_ROURLIM	nqs_range	USR(Q)	OPE	×	転送キュー単位の同時実行数の上限	○
宛先キュー	ATTR_DESTQUE	nqs_qdesc	USR(Q)	OPE	○	宛先キューに関する情報	○
投入経路制限	ATTR_RFUSSB	int	USR(Q)	OPE	○	投入経路別のリクエスト投入制限。下記経路のうち、設定されている経路からの投入を拒否します。 ・RFUSSB_QSUB:　qsub (NQScrereq) ・RFUSSB_QMOV:　qmove (NQSmovreq) ・RFUSSB_LCRQ:　ローカルからの転送 ・RFUSSB_RMRQ:　リモートからの転送	○
ACLモード	ATTR_ACLMODE	int	USR(Q)	MGR	×	【ACLモードがACL_ACCESSの場合】 ユーザ名がACLユーザ名リストに含まれるか、またはグループ名がACLグループ名リストに含まれるユーザのみがキューへのアクセスが許可されます。	○
ACLユーザ名リスト	ATTR_ACLUNAME	char *	USR(Q)	MGR	○		○

ACLグループ名リスト	ATTR_ACLGNAME	char *	USR(Q)	MGR	○	【ACLモードがACL_NOACCESSの場合】 ユーザ名がACLユーザ名リストに含まれるか、またはグループ名がACLグループ名リストに含まれるユーザはキューへのアクセスが禁止されます。 ユーザ名の最大長はNQS_LEN_USERNAMEです。 グループ名の最大長はNQS_LEN_GROUPNAMEです。	○
補助グループ名投入制限	ATTR_SUPGIDCHK	int	USR(Q)	OPE	×	補助グループ名によるリクエスト投入制限の有効/無効を切り替えます。 属性値が真の場合、ACLグループ名リストによる投入制限チェックの際にリクエスト作成時の補助グループ名がチェック対象に加えられます。	○
状態別リクエスト数	ATTR_NREQST	nqs_nreqst	USR(Q)	×	×	状態別に集計されたリクエスト数	○
フックスクリプト制御	ATTR_HOOKFUNC	int	USR(Q)	OPE	×	フックスクリプト機能の有効/無効を切り替える 0：無効（既定値）　1：有効	○
投入数制限値	ATTR_GBLQSBLM	nqs_range	USR(Q)	OPE	×	キュー単位のリクエスト投入数の上限値（0以上の整数。0は無制限を表します。）	○
ユーザ投入数制限値	ATTR_USRQSBLM	int	USR(Q)	OPE	×	キュー単位の１ユーザ当たりのリクエスト投入数の上限値（0 以上の整数。0 は無制限を表します。）	○
グループ投入数制限値	ATTR_GRPQSBLM	int	USR(Q)	OPE	×	キュー単位の１グループ当たりのリクエスト投入数の上限値（0 以上の整数。0 は無制限を表します。）	○
ユーザ名・グループ名指定の制限値							
ユーザ名指定の投入数制限値	ATTR_USRQSBLM_N	nqs_int_u	USR	OPE	○	キュー単位のユーザ名指定でのリクエスト投入数の制限値	○
グループ名指定の投入数制限値	ATTR_GRPQSBLM_N	nqs_int_g	USR	OPE	○	キュー単位のグループ名指定でのリクエスト投入数の制限値	○

4.7　ネットワークキュー属性

属性名	属性種別	型	参照	変更	チェイン	説明	スコープ キュー
キューID	ATTR_QUEID	nqs_qid	USR	×	×	ネットワークキューのキューID	○
キュー状態	ATTR_QUEST	nqs_qst	USR(Q)	OPE	×	現在のキュー状態	○
プライオリティ	ATTR_PRIORITY	nqs_range	USR(Q)	OPE	×	キュー・プライオリティ	○
キュー毎同時実行数制限値	ATTR_NETRLIM	nqs_range	USR(Q)	OPE	×	ネットワークキュー単位の同時実行数の上限	○
リクエスト毎同時実行数制限値	ATTR_BREQRLIM	nqs_range	USR(Q)	OPE	×	リクエスト単位の同時実行数の上限	○
クライアントホスト名	ATTR_HOSTNAME	char *	USR(Q)	OPE	×	ステージング対象クライアントホストのホスト名 [最大長: NQS_LEN_HOSTNAME]	○
ネットワークリクエスト	ATTR_NETREQ	nqs_netreq	USR(R)	×	○	ネットワークリクエストに関する情報	○
ステージング方式	ATTR_STGMETHOD	int	USR(Q)	OPE	×	ネットワークキューのステージング方式 ・STGMTD_INTERNAL： 　バッチサーバ内蔵の標準ステージング方式 ・STGMTD_EXTERNAL： 　ユーザ定義の外部ステージング方式	○
拡張バッファサイズ	ATTR_EXSTGBSZ	nqs_range	USR(Q)	OPE	×	ステージング用バッファとして使用する拡張バッファのサイズをByte単位で指定します。 指定できる範囲は 0～(512×1024)で、0の場合は4000Byteの標準バッファを使用します。	○

4.8　リクエスト属性

属性名	属性種別	型	参照	変更	チェイン	説明	スコープ			
							リクエスト	ジョブ	プロセス	VEノード
リクエストID	ATTR_REQID	nqs_rid	USR	×	×	リクエストID	○			
リクエスト状態	ATTR_REQST	nqs_rst	USR	×	×	現在のリクエスト状態	○			
リクエストタイプ	ATTR_REQTYP	int	USR	×	×	以下のフラグ（ビット）がONかどうかで、リクエストのタイプを識別可能 ・REQTYP_FLAG_QLOGIN（0x000001）：qloginによる会話リクエスト ・REQTYP_FLAG_QRSH　（0x000002）：qrshによる会話リクエスト	○			
キューID	ATTR_QUEID	nqs_qid	USR(R)	×	×	リクエストが投入されているキューのID	○			
リクエスト所有者	ATTR_REQOWN	nqs_udesc	USR(R)	×	×	リクエスト所有者に関する情報 ※nqs_udesc のgidはリクエストのグループのGID。	○			

グループ情報	ATTR_REQOWNGRP	nqs_gdesc	USR(R)	×	×	リクエストのグループに関する情報。 ※リクエストのグループは、リクエスト所有者のプライマリグループ、またはリクエストのグループ指定実行機能ONの場合のリクエスト投入時に指定したグループ。	○			
ジョブ形態	ATTR_JTPLGY	int	USR(R)	USR(R)	×	ジョブの実行形態 ・JTPLGY_DISTRIB　　：分散ジョブ ・JTPLGY_ NECMPI　　：necmpiジョブ ・JTPLGY_OPENMPI　　：openmpiジョブ ・JTPLGY_INTMPI　　：intmpiジョブ ・JTPLGY_MVAPICH　　：mvapichジョブ ・JTPLGY_PLTMPI　　：pltmpiジョブ	○			
リラン属性	ATTR_RERUNABL	int	USR(R)	USR(R)	×	リラン可能か否かを表すフラグ。真ならばリラン可能。（会話リクエスト無し）	○			
リスタート属性	ATTR_RESTARTABL	int	USR(R)	×	×	リスタートが可能か(全ジョブのリスタートファイルが存在するか)否かを表すフラグ。真ならば可能。（会話リクエスト無し）	○			
チェックポイント属性	ATTR_CHKPNTABL	nqs_range	USR(R)	USR(R)	×	0の場合は定期的チェックポイントを実行しません。 -1 の場合は投入キューのATTR_CHKPNTABL属性値が代入されます。	○			
マイグレーション属性	ATTR_MIGRATABL	int	USR(R)	USR(R)	×	ジョブマイグレーション可能か否かを表すフラグ。真ならばマイグレーション可能。(会話リクエスト無し)	○			

ホールド属性	ATTR_HOLDABL	int	USR(R)	USR(R)	×	ホールド可能か否かを表すフラグ。真ならばホールド可能。（会話リクエスト無し）	○			
ホールドタイプ	ATTR_HOLDTYPE	int	USR(R)	×	×	ホールド実行時のクライアント権限（会話リクエスト無し） ・OPBY_SCHEDULER：スケジューラが実行 ・OPBY_MANAGER：管理者が実行 ・OPBY_OPERATOR：操作員が実行 ・OPBY_GMANAGER：グループ管理者が実行 ・OPBY_SPUSER：特別利用者が実行 ・OPBY_USER：リクエスト所有者が実行 ・OPBY_NONE：ホールドされていない	○			
サスペンドタイプ	ATTR_SUSPTYPE	int	USR(R)	×	×	サスペンド実行時のクライアント権限 ・OPBY_SCHEDULER：スケジューラが実行 ・OPBY_MANAGER：管理者が実行 ・OPBY_OPERATOR：操作員が実行 ・OPBY_GMANAGER：グループ管理者が実行 ・OPBY_SPUSER：特別利用者が実行 ・OPBY_USER：リクエスト所有者が実行 ・OPBY_NONE：サスペンドされていない	○			
リラン回数	ATTR_RERUNCNT	int	USR(R)	×	×	リクエストがリランされた回数。（会話リクエスト無し）	○			
アカウントコード	ATTR_ACCTCODE	char *	USR(R)	USR(R)	×	アカウントコード[最大長：NQS_LEN_ACCTCODE]	○			

プライオリティ	ATTR_PRIORITY	nqs_range	USR(R)	USR(R)	×	リクエスト・プライオリティ	○			
リクエスト名	ATTR_REQNAME	char *	USR(R)	USR(R)	×	リクエストの名前[最大長：NQS_LEN_REQNAME]	○			
標準出力パス名	ATTR_STDOUT	nqs_pdesc	USR(R)	USR(R)	×	標準出力ファイルの転送先（会話リクエスト無し）	○			
標準エラー出力パス名	ATTR_STDERR	nqs_pdesc	USR(R)	USR(R)	×	標準エラー出力ファイルの転送先（会話リクエスト無し）	○			
リクエストログ出力パス名	ATTR_STDLOG	nqs_pdesc	USR(R)	USR(R)	×	リクエストログ出力ファイルの転送先（会話リクエスト無し）	○			
リクエストログの出力レベル	ATTR_LOGLEVEL	nqs_range	USR(R)	USR(R)	×	リクエストログの出力レベル（会話リクエスト無し）	○			
ステージングファイル情報	ATTR_STGFILE	nqs_stgfile	USR(R)	×	○	ステージング対象ファイルに関する情報（会話リクエスト無し）	○			
シェル名	ATTR_SHELLPATH	char *	USR(R)	USR(R)	×	ジョブ実行シェルのフルパス名[最大長:NQS_LEN_PATHNAME]	○			

メール オプション	ATTR_MAILOPTS	int	USR(R)	USR(R)	×	メール通知に関するオプション(複数同時指定可) ・MAIL_SENDBGN： 　リクエスト実行開始時にメールする ・MAIL_SENDEND： 　リクエスト実行終了時にメールする	○			
メール アドレス	ATTR_MAILADDR	nqs_mdesc	USR(R)	USR(R)	×	メールアドレス	○			
ジョブ 実行環境条件	ATTR_JOBCOND	nqs_jcond	USR(R)	×	○	ジョブ毎の実行ホストに関する条件	○			
JOBCOND 総数	ATTR_NJCONS	int	USR(R)	×	×	ジョブ実行環境条件の総エントリ数	○			
リクエスト グループ	ATTR_REQGRP	nqs_rgrp	USR(R)	×	×	リクエスト連携時に自身が所属するリクエストグループ	○			
作成時刻	ATTR_CRETIME	time_t	USR(R)	×	×	リクエストが作成された時刻(EPOCHからの経過秒)	○			
投入時刻	ATTR_ENTTIME	time_t	USR(R)	×	×	リクエストが現在のキューに投入された時刻(EPOCHからの経過秒)	○			
実行時刻	ATTR_EXETIME	time_t	USR(R)	USR(R)	×	リクエストがWAITING状態を終える時刻(EPOCHからの経過秒)	○			
予約時刻	ATTR_RSVTIME	time_t	USR(R)	USR(R)	×	リクエストの実行開始を予約した時刻(EPOCHからの経過秒)（会話リクエスト無し）	○			

開始時刻	ATTR_BGNTIME	time_t	USR(R)	×	×	リクエストが実行を開始した時刻(EPOCHからの経過秒)	○			
終了時刻	ATTR_ENDTIME	time_t	USR(R)	×	×	リクエストが実行を終了した時刻(EPOCHからの経過秒)	○			
UMASK値	ATTR_UMASK	int	USR(R)	×	×	リクエスト投入時のUMASK値			○	
ジョブ 環境変数	ATTR_ENVIRON	nqs_keyval	×	×	○	環境変数名と変数値の対			○	
マイグレーションファイル	ATTR_MIGFILE	nqs_migfile	USR(R)	×	○	ジョブマイグレーション時、ジョブと同時にマイグレーション先ホストへコピーされるユーザファイルまたはチェックポイントファイル。（会話リクエスト無し）	○			
ユーザ定義属性	ATTR_USERATTR	nqs_keyval	USR(R)	×	○	ユーザが定義可能な属性	○			
リスタートファイル 格納パス	ATTR_RSTFDIR	char *	USR(R)	USR(R)	×	リスタートファイルが格納される実行ホスト上の相対パス名[最大長：NQS_LEN_PATHNAME]（会話リクエスト無し）	○			
予約区間ID	ATTR_ADVRSVID	int	USR(R)	×	×	事前予約機能用の予約区間ID。　IDは0以上の整数値で表され、負数はIDが未設定であることを表す。（会話リクエスト無し）	○			
実行開始予定時刻	ATTR_ASSTIME	time_t	USR(R)	SCH	×	スケジューラが割り当てた実行開始予定時刻	○			
アタッチ機能利用	ATTR_QATTACH	int	USR(R)	MGR	×	1の場合アタッチ機能利用可能 0の場合アタッチ機能利用不可	○			

アタッチフラグ	ATTR_QATTACH_EXEC	int	USR(R)	×	×	リクエストがアタッチ中か否かを示します。 0: 未アタッチ 1: アタッチ中	○			
ジョブの有無	ATTR_JOBEXIST	int	USR(R)	×	×	リクエストがジョブを持つか否かを示します。 0: ジョブなし 1: ジョブあり	○			
先行 リクエスト	ATTR_PRECEDING	nqs_rid	USR(R)	×	○	先行リクエストとして設定されているリクエストのリクエストID	○			
同時実行開始リクエスト	ATTR_PARALLEL	nqs_rid	USR(R)	×	○	同時実行開始を指定された他のリクエストのリクエストID	○			
エラー時後続リクエストキャンセル	ATTR_CANCELAFTER	int	USR(R)	×	×	リクエストが異常終了した場合に、後続リクエストを削除するか否かを示します。 1: 削除する 0: 削除しない	○			
ワークフローID	ATTR_WFID	nqs_wid	USR(R)	×	×	リクエストが属するワークフローのID	○			
ユーザPPスクリプト情報	ATTR_USERPP	nqs_upp	USR(R)	USR(R)	○	ジョブ実行直前または直後に起動するユーザPPスクリプトの情報	○			
OpenStack 用テンプレート	ATTR_OSTEMPLATE	nqs_ostemplate	USR(R)	×	×	OpenStack用のテンプレート指定値	○			
コンテナ用テンプレート	ATTR_COTEMPLATE	nqs_cotemplate	USR(R)	×	×	コンテナ用のテンプレート指定値	○			

カスタムリソース利用量情報	ATTR_REQCRINFO	nqs_reqcrinfo	USR(R)	USR(R)	○	リクエストでのカスタムリソース利用量情報（カスタムリソース名、利用量）	○			
VEノード総数	ATTR_TOTALVENUM	nqs_rrange	USR(R)	×	×	VEノード総数指定値	○			
排他実行属性	ATTR_EXCLUSIVE	int	USR(R)	×	×	排他実行の種類 0: 排他実行しない 1: host単位で排他実行する	○			
カスタムリソース使用実績値	ATTR_CRUSG	nqs_crusg	USR(R)	×	○	カスタムリソースの使用実績値	○			
SIGTERM 捕捉可否	ATTR_ACCEPTSIGTERM	int	USR(R)	USR(R)	×	SIGTERM捕捉の可否 0: SIGTERMを無視する（既定値） 1: SIGTERMを無視しない	○			
パラメトリックリクエスト関連情報（パラメトリックリクエストのみ）										
サブリクエスト数	ATTR_NSUBREQS	nqs_nsubreq	USR(R)	×	×	サブリクエスト数	○			
サブリクエスト番号指定文字列	ATTR_SUBREQSTR	char *	USR(R)	×	×	qsub -tで指定したサブリクエスト番号指定文字列	○			
会話リクエスト関連情報（会話リクエストのみ）										
投入ホスト	ATTR_SUBMIT_HOST	nqs_hid	USR(R)	×	×	リクエスト投入ホスト（会話リクエストのみ）	○			

会話用接続ポート番号	ATTR_PORT	int	USR(R)	×	×	会話リクエストのセッション接続ポート番号（会話リクエストのみ）	○			
待ち合わせ指定	ATTR_SCH_WAIT	int	USR(R)	×	×	スケジューリングの結果、実行ホストの割り当てができなかった場合の動作指定（会話リクエストのみ） MODE_CANCEL：投入キャンセル MODE_WAIT：　　待ち合わせ	○			
強制実行シェル	ATTR_RESTRICT_SHELL	char *	USR(R)	OPE	×	強制的に実行するシェル（会話リクエストのみ）	○			
アイドルタイマー	ATTR_IDLETIMER	int	USR(R)	USR	×	会話リクエストのアイドルタイマーを分単位で保持 ０の場合、アイドルタイマー＝OFF	○			
資源制限値関連										
経過時間制限値	ATTR_ELPSTIM	nqs_rlim	USR(R)	USR(R)	×	実行開始からの経過時間による制限	○			
CPU時間制限値	ATTR_CPUTIM	nqs_rlim	USR(R)	USR(R)	×	CPU使用時間による制限		○	○	
同時実行CPU台数制限値	ATTR_CPUNUM	nqs_rnum	USR(R)	USR(R)	×	同時実行可能なCPU台数の制限		○		
オープンファイル数制限値	ATTR_FILENUM	nqs_rnum	USR(R)	USR(R)	×	同時オープンファイル数の制限			○	
メモリサイズ制限値	ATTR_MEMSZ	nqs_rlim	USR(R)	USR(R)	×	使用可能な最大メモリサイズの制限		○	○	

データサイズ制限値	ATTR_DATASZ	nqs_rlim	USR(R)	USR(R)	×	使用可能な最大データセグメントサイズの制限			○	
スタックサイズ制限値	ATTR_STACKSZ	nqs_rlim	USR(R)	USR(R)	×	使用可能な最大スタックサイズの制限			○	
コアファイルサイズ制限値	ATTR_CORESZ	nqs_rlim	USR(R)	USR(R)	×	生成可能な最大コアファイルサイズの制限			○	
ファイルサイズ制限値	ATTR_FILESZ	nqs_rlim	USR(R)	USR(R)	×	生成可能な最大ファイルサイズの制限			○	
VE CPU 時間制限	ATTR_VECPUTIMRNG	nqs_rrange	USR(R)	OPE	×	VEのCPU時間による制限		○	○	○
VEメモリサイズ制限	ATTR_VEMEMSZRNG	nqs_rrange	USR(R)	OPE	×	使用可能なVEのメモリサイズの制限		○	○	○
仮想メモリサイズ制限値	ATTR_VMEMSZ	nqs_rlim	USR(R)	USR(R)	×	使用可能な最大仮想メモリサイズの制限		○	○	
同時実行GPU台数制限値	ATTR_GPUNUM	nqs_rnum	USR(R)	USR(R)	×	同時実行可能なGPU台数の制限		○		
同時実行VEノード台数 制限値	ATTR_VENUM	nqs_rrange	USR(R)	USR(R)	×	同時実行可能なVEノード台数の制限		○		
HCAポート数	ATTR_HCA	nqs_hca	USR(R)	USR(R)	×	使用可能なHCAポート数の制限		○		

ジョブのグループ	ATTR_JOBGROUP	nqs_jobgroup	USR(R)	USR(R)	○	ハイブリッドジョブにおけるジョブのグループ毎に異なる属性。		○		
カーネルパラメータ関連										
RSG番号	ATTR_RSGNO	nqs_range	USR(R)	×	×	リソースシェアリンググループ番号			○	
ナイス値	ATTR_NICE	nqs_range	USR(R)	OPE	×	ナイス値			○	
資源使用量関連										
経過時間	ATTR_USEELPSTIM	int	USR(R)	×	×	リクエストが実行を開始してからの経過時間　[単位: 秒]	○			
CPU時間 使用量	ATTR_USECPUTIM	long long	USR(R)	×	×	現時点のCPU時間使用量(システム時間とユーザ時間の合計)。終了したプロセスは含まれない。　[単位: マイクロ秒]	○			
CPU 積算使用量	ATTR_ACCCPUTIM	long long	USR(R)	×	×	実行開始からの積算CPU時間使用量(システム時間とユーザ時間の合計)。終了したプロセスも含まれる。　[単位: マイクロ秒]	○			
メモリ使用量	ATTR_USEMEMSZ	long long	USR(R)	×	×	メモリ使用量(テキスト、データ、スタック、共用メモリ等の総和)　[単位: バイト]	○			
仮想メモリ 使用量	ATTR_USEVMEMSZ	long long	USR(R)	×	×	仮想メモリ使用量。　　[単位: バイト]	○			

4.9　ジョブ属性

属性名	属性種別	型	参照	変更	チェイン	説明	スコープ		
							ジョブ	プロセス	VEノード
ジョブID	ATTR_JOBID	nqs_jid	USR	×	×	ジョブID（リクエストとリクエスト内の番号）	○		
実行ジョブID	ATTR_EXEJID	int	USR(R)	×	×	カーネルによってアサインされる実行ホスト上でのジョブID（SID(セッションID)。） ジョブが実行中でない場合は負の整数。	○		
ジョブ実行者	ATTR_JOBOWN	nqs_udesc	USR(R)	×	×	ジョブ実行者に関する情報	○		
ジョブサーバID	ATTR_JSVID	nqs_jsvid	USR(R)	×	×	ジョブを管理しているジョブサーバ	○		
実行ホストID	ATTR_HSTID	nqs_hid	USR(R)	×	×	ジョブが存在している実行ホスト	○		
仮想マシンホストID	ATTR_VMHOSTNAME	char *	USR(R)	×	×	ジョブが存在している仮想マシンまたはコンテナのホスト名 ※仮想マシンまたはコンテナで実行していない場合はNULL	○		
割り当てソケット情報	ATTR_SOCKETS	char *	USR(R)	×	×	割り当てソケット番号 ※未割り当ての場合は "(none)"	○		
ジョブ終了コード	ATTR_EXITCODE	int	USR(R)	×	×	ジョブ(ジョブ内の最上位プロセス)の終了コード。実行が終了していない場合は負の整数。	○		
リスタートファイル	ATTR_RSTFINFO	nqs_rstf	USR(R)	×	×	リスタートファイルに関する情報。（会話ジョブ無し）	○		
アカウントコード	ATTR_ACCTCODE	char *	USR(R)	×	×	ジョブ独自には持たず、親リクエストの属性値を参照する	○		
資源制限値関連									
CPU時間制限値	リクエスト属性と同じ		USR(R)	×	×	ジョブ独自には持たず、親リクエストの属性値を参照	○	○	

同時実行CPU台数制限値			USR(R)	×	×	する	○		
オープンファイル数制限値			USR(R)	×	×			○	
メモリサイズ制限値			USR(R)	×	×		○	○	
データサイズ制限値			USR(R)	×	×			○	
スタックサイズ制限値			USR(R)	×	×			○	
コアファイルサイズ制限値			USR(R)	×	×			○	
ファイルサイズ制限値			USR(R)	×	×			○	
VE CPU時間制限			USR(R)	×	×		○	○	○
VEメモリサイズ制限			USR(R)	×	×		○	○	○
仮想メモリサイズ制限値			USR(R)	×	×		○	○	
同時実行GPU台数制限値			USR(R)	×	×		○		
カーネルパラメータ関連									
RSG番号	リクエスト属性と同じ		USR(R)	×	×	ジョブ独自には持たず、親リクエストの属性値を参照する		○	
ナイス値			USR(R)	×	×			○	
資源使用量関連									
CPU時間使用量	ATTR_USECPUTIM	long long	USR(R)	×	×	現時点のCPU時間使用量(システム時間とユーザ時間	○		

						の合計)。終了したプロセスは含まれない。 [単位: マイクロ秒]			
CPU積算使用量	ATTR_ACCCPUTIM	long long	USR(R)	×	×	実行開始からの積算CPU時間使用量(システム時間とユーザ時間の合計)。終了したプロセスも含まれる。 [単位: マイクロ秒]	○		
メモリ使用量	ATTR_USEMEMSZ	long long	USR(R)	×	×	メモリ使用量(テキスト、データ、スタック、共用メモリ等の総和) [単位: バイト]	○		
仮想メモリ使用量	ATTR_USEVMEMSZ	long long	USR(R)	×	×	仮想メモリ使用量。 [単位: バイト]	○		
カスタムリソース使用実績値	ATTR_CRUSG	nqs_crusg	USR(R)	×	○	カスタムリソース使用実績値	○		

4.10 ノードグループ属性

属性名	属性種別	型	参照	変更	チェイン	説明	スコープ
							ノードグループ
ノードグループID	ATTR_NGRPID	nqs_ngrpid	USR	×	×	ノードグループID	○
コメント	ATTR_COMMENT	char *	USR	OPE	×	コメント	○
バインド許可フラグ	ATTR_BINDABLE	int(boolean)	USR	OPE	×	バインド可能の場合True、不可の場合False	○
ジョブサーバIDリスト	ATTR_JSVID	nqs_jsvid	USR	×	○	ノードグループに含まれるJSVのリスト	○
キューIDリスト	ATTR_QUEID	nqs_qid	USR	×	○	接続しているキューのリスト	○

4.11 ワークフロー属性

属性名	属性種別	型	参照	変更	チェイン	説明	スコープ
							ワークフロー
ワークフローID	ATTR_WFID	nqs_wid	USR	×	×	ワークフローのID	○
リクエストID	ATTR_REQID	nqs_rid	USR	×	○	ワークフローに属するリクエストのID	○
ワークフローの所有者	ATTR_WFLOWN	nqs_udesc	USR	×	×	ワークフローの所有者情報	○

4.12 カスタムリソース属性

属性名	属性種別	型	参照	変更	チェイン	説明	スコープ
							BSV
カスタムリソースID	ATTR_CRID	nqs_crid	USR	×	×	カスタムリソースID	○
消費単位	ATTR_CONSUMER	int	USR	MGR	×	カスタムリソースを消費する単位 ・CR_JOB：job ・CR_REQ：request	○
利用量制御情報リスト	ATTR_CRRESOURCE	nqs_crresource	USR	MGR	○	カスタムリソースの利用量制御情報 （対象種別、対象、同時に利用可能なリソースの最大値）	○
監視モード	ATTR_CR_CHKMD	int	USR	MGR	×	資源監視モード ・CR_CM_OFF：監視しない。 ・CR_CM_MOMENT：瞬間値として監視する。 ・CR_CM_INTEGRATE：積算値として監視する。	○
ジョブ強制終了	ATTR_CR_TRMJB	int	USR	MGR	×	資源超過時ジョブ強制終了機能 0：無効 1：有効	○
単位	ATTR_CR_UNIT	char *	USR	MGR	×	資源の単位［最大長：NQS_LEN_CRUNIT］	○

第5章　　データ型／定数定義

5.1　記号定数

定数名	定数値	説明
NQS_MAX_JSVNO	10239	ジョブサーバ番号の最大値
NQS_MAX_JOBNO	10239	ジョブ番号の最大値
NQS_MAX_SCHNO	15	バッチスケジューラ番号の最大値
NQS_LEN_HOSTNAME	255	ホスト名の最大長
NQS_LEN_FILENAME	255	ファイル名の最大長
NQS_LEN_PATHNAME	1023	パス名の最大長
NQS_LEN_UTSNAME	63	UTS名の最大長
NQS_LEN_JSVNAME	15	ジョブサーバ名の最大長
NQS_LEN_SCHNAME	15	バッチスケジューラ名の最大長
NQS_LEN_QUENAME	15	キュー名の最大長
NQS_LEN_REQNAME	63	リクエスト名の最大長
NQS_LEN_ACCTCODE	127	アカウントコードの最大長
NQS_LEN_MAILADDR	1023	メールアドレスの最大長
NQS_LEN_JOBCOND	255	ジョブ実行環境条件式の最大長
NQS_LEN_ERRMSG	127	APIエラーメッセージの最大長
NQS_LEN_USERNAME	47	ユーザ名の最大長
NQS_LEN_GROUPNAME	47	グループ名の最大長
NQS_LEN_VERSION	15	バージョン文字列の最大長
NQS_LEN_LICNAME	15	ライセンス機能名の最大長
NQS_LEN_COMMENT	63	コメント文字列の最大長
NQS_LEN_RSTFPATH	47	リスタートファイル格納パス名の最大長
NQS_LEN_SCHDMSG	4000	スケジューラメッセージの最大長
NQS_LEN_COMMAND	2047	コマンド文字列の最大長
NQS_LEN_JOBNODSC	1535	ジョブ番号の文字列の最大長
NQS_LEN_NGRPNAME	15	ノードグループ名の最大長
NQS_LEN_TEMPLATENAME	47	テンプレート名の最大長
NQS_LEN_VMIMGNAME	47	テンプレートのイメージ名の最大長
NQS_LEN_FLAVORNAME	47	テンプレートのフレーバ名の最大長
NQS_LEN_TEMPLATECUSTOM	400	テンプレートの独自定義文字列の最大長
NQS_LEN_TEMPLATECOMMENT	255	テンプレートのコメント文字列の最大長
NQS_MAX_CRNUM	20	カスタムリソースの最大数
NQS_LEN_CRNAME	15	カスタムリソース名の最大長
NQS_LEN_CRUNIT	5	カスタムリソースに設定する単位の最大長
NQS_LIM_UNUSED	0	カスタムリソース利用量のunused指定
NQS_LEN_CPUSETNAME	255	CPUSET名の最大長
NQS_LEN_CPUS	255	CPU番号の文字列の最大長
NQS_LEN_MEMS	255	メモリノード番号の文字列の最大長

NQS_LEN_GPUNAME	255	GPUデバイス名の最大長

5.2　構造体

- nqs_aid (属性識別子)

```
typedef struct nqs_aid {
    int type;               /* 属性種別 */
    int scope;              /* スコープ */
} nqs_aid;
```

type は属性のタイプ、scope は属性が適用される範囲を表します。

- nqs_alist (属性リスト識別子)

```
typedef int nqs_alist;
```

属性リストを識別するための 0 以上の整数です。

- nqs_cotemplate (コンテナ用テンプレート定義)

```
typedef struct nqs_cotemplate {
    char template_name[NQS_LEN_TEMPLATENAME+1];  /* テンプレート名 */
    char image_name [NQS_LEN_VMIMGNAME+1];       /* イメージ名 */
    int cpunum;                                  /* CPU 数 */
    int memsz;                                   /* メモリ量 */
    int memunit;                                 /* メモリ量の単位 */
    int gpunum;                                  /* GPU 数 */
    char custom[NQS_LEN_TEMPLATECUSTOM+1];       /* 独自定義 */
    char comment[NQS_LEN_TEMPLATECOMMENT+1];     /* コメント */
    int lock;                                    /* ロック状態 */
    int starttimeout;                            /* 起動見込時間 */
    int stoptimeout;                             /* 停止見込時間 */
    int venum;                                   /* VE 数 */
    nqs_usehca usecha;                           /* HCA ポート数 */
} nqs_cotemplate;
```

コンテナ用テンプレート定義の情報です。

ロック状態 lock は、TEMPLATE_LOCK または TEMPLATE_UNLOCK のいずれかの値となります。

- nqs_cpuset (CPUSET情報)

```
typedef struct nqs_cpuset {
    int no;                             /* RSG 番号  */
    char name[NQS_LEN_CPUSETNAME+1];    /* CPUSET 名 */
    char cpus[NQS_LEN_CPUS+1];          /* コア番号  */
    char mems[NQS_LEN_MEMS+1];          /* メモリノード番号 */
} nqs_cpuset;
```

nqs_cpuset 構造体は、RSG 番号に対応した CPUSET 情報を保持します。

- nqs_crid (カスタムリソースID)

```
typedef struct nqs_crid {
    char cr_name[NQS_LEN_CRNAME+1];     /* カスタムリソース */
} nqs_crid;
```

cr_name には、カスタムリソースの名前が格納されます。

- **nqs_crresource (カスタムリソースの利用量制御情報)**

```
typedef struct nqs_crresource {
    int  cr_type;                          /* 利用量制御の対象種別 */
    char cr_target[NQS_LEN_HOSTNAME+1]; /* 利用量制御の対象   */
    int  cr_available;           /* 利用量制御で同時利用可能リソースの最大値 */
} nqs_crresource;
```

nqs_ crresource 構造体は、カスタムリソースの利用量制御情報を保持します。
cr_type と cr_target で示す利用量制御の対象に対する、同時利用可能リソースの最大値を cr_available に格納します。
cr_type は利用量制御の対象種別で、設定値は以下のいずれかです（マクロ定義)。

- CR_BSV_DEFAULT　利用量制御の対象を BSV（既定値）とする
- CR_HOST_DEFAULT 利用量制御の対象を実行ホスト（既定値）とする
- CR_HOST　　利用量制御の対象を実行ホスト名で個別指定する

cr_target は cr_type が CR_HOST の場合のみ利用量制御の対象を個別指定する実行ホスト名を格納します。その他の場合は NULL 文字です。

- **nqs_entry (エントリ識別子)**

```
typedef int nqs_entry;
```

エントリを識別するための 0 以上の整数です。

- **nqs_event (APIイベント)**

```
typedef struct nqs_event {
    int event_id;              /* イベント識別子 */
    time_t occur_time;         /* イベント発生時刻 */
    union {
        struct evt_jsv jsv;    /* ジョブサーバ系イベントの内容 */
        struct evt_qst qst;    /* キュー状態系イベントの内容 */
        struct evt_qat qat;    /* キュー属性系イベントの内容 */
        struct evt_rst rst;    /* リクエスト状態系イベントの内容 */
        struct evt_rat rat;    /* リクエスト属性系イベントの内容 */
        struct evt_ngrp ngrp;  /* ノードグループ系イベントの内容 */
        struct evt_hst hst;       /* 実行ホスト状態系イベントの内容 */
        struct evt_template tmpl; /* テンプレート系イベントの内容 */
        struct evt_crs;           /* カスタムリソース系イベントの内容 */
    } cargo;                      /* イベント内容 */
} nqs_event;

struct evt_jsv {
    nqs_jsvid jsvid;   /* ジョブサーバ ID */
    nqs_hid hid;       /* ホスト ID */
    int   state_link;  /* リンク情報 */
    int    bind_count; /* バインドキュー数 */
    nqs_vjsvid vjsvid; /* Internal Use Only */
};

struct evt_qst {
    nqs_qid qid;       /* キューID */
    nqs_qst qst;       /* 現在のキュー状態 */
    int bind_id;       /* 接続/切り放し対象のシーケンス番号 */
    nqs_ngrpid ngrpid; /* 接続/切り放し対象のノードグループ */
};

struct evt_qat {
    nqs_qid qid;       /* キューID */
    nqs_aid aid;       /* 属性識別子 */
    nqs_alist ad;      /* 現在の属性値(属性リスト) */
};

struct evt_rst {
    nqs_rid rid;       /* リクエスト ID */
    nqs_qid qid;       /* キューID */
    nqs_rst rst;       /* 現在のリクエスト状態 */
};

struct evt_rat {
    nqs_rid rid;       /* リクエスト ID */
    nqs_qid qid;       /* キューID */
    nqs_aid aid;       /* 属性識別子 */
    nqs_alist ad;      /* 現在の属性値(属性リスト) */
};

struct evt_ngrp {
    nqs_ngrpid  ngrpid;   /* ノードグループ ID */
    nqs_aid aid;          /* 属性識別子 */
    nqs_alist ad;         /* 属性値(属性リスト) */
};
```

```
struct evt_hst {
    nqs_hid hid;        /* ホスト ID */
    nqs_hst hst;        /* ホスト稼働状態 */
    nqs_jsvid jsvid;    /* 現在の JSVID */
};

typedef struct evt_template {
    int type;                           /* テンプレートタイプ */
    union {
        struct nqs_vmtemplate vm_tmpl;
        struct nqs_ostemplate os_tmpl;  /* OpenStack テンプレート定義 */
        struct nqs_cotemplate co_tmpl;  /* コンテナ用テンプレート定義 */
    }t1;
} evt_template;

struct evt_crs {
    nqs_crid crid;      /* カスタムリソース ID */
    int consumer;       /* 消費単位  */
    nqs_aid aid;        /* 属性識別子 */
    nqs_alist ad;       /* 属性値(属性リスト) */
};
```

イベント内容が格納されている nqs_event.cargo 共用体のメンバはイベントタイプから判断してください。イベントタイプは NQSEVT_TYPE マクロで得ることができます。

evt_rst.qid および evt_rat.qid にはリクエストが投入されているキューのキューID が格納されます。

evt_jsv.hid にはジョブサーバが存在している実行ホストのホスト ID が格納されます。

イベント識別子が NQSEVT_QST_BINDJSV または NQSEVT_QST_UNBINDJSV の場合、evt_qst.bind_id には接続/切り放し対象となったジョブサーバのジョブサーバ番号が設定されます。同様に NQSEVT_QST_BINDJSV または NQSEVT_QST_UNBINDJSV の場合はスケジューラのスケジューラ番号が設定されます。それ以外のキュー状態系イベントでは evt_qst.bind_id の値は不定です。

属性系イベントに含まれる属性値は属性リスト形式のデータで、イベント処理専用の属性リストとして API 内部に生成されます。 また、NQSevent 関数を実行する度に古い属性リストは破棄されます。

ノードグループ系イベントの場合、イベント識別子が NQSEVT_NGRP_ADDNODE、または、NQSEVT_NGRP_DELNODE のときには、追加、または、削除されたジョブサーバのジョブサーバ番号（ATTR_JSVID）が、 evt_ngrp 内の aid, ad によって取得できます。

テンプレート系イベントの場合、テンプレートタイプ type が NQSII_TEMPLATE_TYPE_OPENSTACK の時、作成・変更または削除された OpenStack テンプレートの情報が evt_template 内の os_tmpl で取得できます。また、テンプレートタイプ type が NQSII_TEMPLATE_TYPE_CONTAINER の時、作成・変更または削除されたコンテナ用テンプレートの情報が evt_template 内の co_tmpl で取得できます。

カスタムリソース系イベントの場合、イベント識別子が NQSEVT_CRS_RESCHANGED の時、追加または削除された利用量制御情報リスト（ATTR_CRRESOURCE）が evt_crs 内の aid, ad によって取得できます。

- **nqs_gbcset（GBC割り当て情報）**

```
typedef struct nqs_gbcset {
```

```
    int  njobs_from;    /* ジョブ数(範囲の開始) */
    int  njobs_to;      /* ジョブ数(範囲の終了) */
    int  ngbc;          /* 割り当て GBC 数 */
} nqs_gbcset;
```

ジョブ数が、njobs_from～njobs_to の範囲のリクエストに対して、割り当てる GBC 数を ngbc とするよう設定されていることを表します。

- **nqs_gdesc（グループ記述子）**

```
typedef struct nqs_gdesc {
    gid_t gid;                          /* グループ ID */
    char *gname [NQS_LEN_GROUPNAME+1];  /* グループ名 */
} nqs_gdesc;
```

グループ名が gname、グループ ID が gid であるグループを表します。

- **nqs_gpuinfo（GPU詳細情報）**

```
typedef struct nqs_gpuinfo {
    int device_no;                      /* デバイス番号 */
    char name [NQS_LEN_GPUNAME+1];      /* デバイス名 */
    int total_global_mem;               /* グローバルメモリ量 (MB) */
} nqs_gpuinfo;
```

nqs_ **gpuinfo** 構造体は、GPU の詳細情報を保持します。

- **nqs_hid (ホストID)**

```
typedef struct nqs_hid {
    struct in_addr ip;      /* IP アドレス */
} nqs_hid;
```

ip はバッチサーバが認識しているリモートホスト(実行ホスト、クライアントホスト等)の IP アドレスで、ネットワークバイトオーダです。

- **nqs_hilo (上下限値)**

```
typedef struct nqs_hilo {
    int high;       /* 上限値 */
    int low;        /* 下限値 */
} nqs_hilo;
```

属性値の上限と下限を表します。

- **nqs_hilo_g（グループ別上下限値）**

```
typedef nqs_hilo_g {
  nqs_gdesc group;          /* 制限対象のグループ */
  nqs_hilo hilo;            /* 上下限値 */
} nqs_hilo_g;
```

グループ別に設定する属性値の上限と下限を表します。

- **nqs_hilo_u（ユーザ別上下限値）**

```
typedef nqs_hilo_u {
  nqs_udesc user;           /* 制限対象のユーザ */
  nqs_hilo hilo;            /* 上下限値 */
} nqs_hilo_u;
```

ユーザ別に設定する属性値の上限と下限を表します。

-

- **nqs_hst (ホスト状態)**

```
typedef struct nqs_hst {
    int state_curr;       /* 現在の状態 */
    int state_prev;       /* 直前の状態 */
    int state_reason;     /* 状態遷移理由 */
    time_t state_time;    /* 状態遷移時刻 */
}  nqs_hst;
```

nqs_hst 構造体は実行ホストの状態を表します。
現在の状態 state_curr、直前の状態 state_prev、状態遷移理由 state_reason、状態遷移時刻 state_time を保持します。
状態については、以下の値をとります。

HOSTST_ACTIVE　稼働状態
HOSTST_INACTIVE　停止状態

状態遷移理由は、ノード管理エージェントからの取得情報、およびジョブサーバのリンクアップ／ダウンにより、以下のものを設定します。

状態遷移理由	意味
HOSTRSN_POWERSAVING_DCOFF	スケジューラからの省電力機能によるシャットダウン（停止状態）
HOSTRSN_RETURN_POWERSAVING_DCOFF	スケジューラからの省電力機能による起動（稼働状態）
HOSTRSN_JSVLINKUP	ジョブサーバリンクアップ（稼働状態）
HOSTRSN_JSVLINKDOWN	ジョブサーバリンクダウン（停止状態）
HOSTRSN_NODE_ABNORMAL_STOP	障害通知コマンドによるノードダウン（停止状態）
※ノード管理エージェントが非接続の場合	
HOSTRSN_JSVLINKUP	ジョブサーバリンクアップ（稼働状態）
HOSTRSN_JSVLINKDOWN	ジョブサーバリンクダウン（停止状態）

- **nqs_int_g（グループ別制限数）**

```
typedef nqs_int_g {
  nqs_gdesc group;          /* 制限対象のグループ */
  int val;                  /* 制限数 */
} nqs_int_g;
```

グループ別に設定する属性値の数値を表します。

- **nqs_int_u（ユーザ別制限数）**

```
typedef nqs_int_u {
  nqs_udesc user;          /* 制限対象のユーザ */
  int val;                 /* 制限数 */
} nqs_int_u;
```

ユーザ別に設定する属性値の数値を表します。

- **nqs_jcond（ジョブ実行環境条件）**

```
typedef struct nqs_jcond {
    int jobno;                          /* ジョブ番号 */
    char *condition;                    /* 条件式 */
} nqs_jcond;
```

jobno 番ジョブを起動する実行ホストの条件を condition で指定します。 condition のフォーマットはバッチスケジューラ毎に定義されます。 condition 文字列は先頭から NQS_LEN_JOBCOND バイトまでが有効です。

- **nqs_jid（ジョブID）**

```
typedef struct nqs_jid {
    nqs_rid rid;             /* リクエスト ID */
    int jobno;               /* ジョブ番号 */
} nqs_jid;
```

rid は親リクエストのリクエスト ID、jobno は同一リクエストを親に持つジョブのシリアル番号(ジョブ番号)です。 jobno が 0 のジョブはマスタージョブ、1 以上はスレーブジョブを表します。

- **nqs_jsvid（ジョブサーバID）**

```
typedef struct nqs_jsvid {
    int jsvno;              /* ジョブサーバ番号 */
} nqs_jsvid;
```

jsvno はジョブサーバ番号で範囲 0～NQS_MAX_JSVNO の整数値です。

- **nqs_jsvst（ジョブサーバ状態）**

```
typedef struct nqs_jsvst {
    int state_link;      /* リンク状態 */
    int state_bind;      /* バインド状態 */
} nqs_jsvst;
```

state_link はジョブサーバ、バッチサーバ間のリンク状態(TCP コネクションの有無)を、state_bind はジョブサーバとキューの接続状態をそれぞれ表します。取りうる値は下表のとおりです。

・ state_link

JSVST_LINKUP	リンクが確立している
JSVST_LINKDOWN	リンクが切断されている

・ state_bind

JSVST_BIND	キューに接続している
JSVST_UNBIND	キューから切り放されている

- **nqs_keyval (キー値ペア)**

```
typedef struct nqs_keyval {
    char *key;        /* キー */
    char *val;        /* 値 */
} nqs_keyval;
```

キー文字列と値文字列の組を表します。key と val の文字列長の合計は 4000 バイト以下に制限されます(API パケットの制約)。

- **nqs_license (ライセンス情報)**

```
typedef struct nqs_license {
    feature[NQS_LEN_LICNAME + 1]; /* ライセンス機能名 */
    int max_license;              /* 最大ライセンス数 */
    int busy_license;             /* 使用中のライセンス数 */
} nqs_license;
```

機能名(FEATURE)毎のライセンス数が格納されます。

- **nqs_mdesc (メールアドレス記述子)**

```
typedef struct nqs_mdesc {
    char mail[NQS_LEN_MAILADDR + 1];  /* メールアドレス */
} nqs_mdesc;
```

mail は user_name@mail_domain 形式のメールアドレスです。複数アドレスの場合はスペースもしくはカンマ文字で区切ります。

- **nqs_migfile (マイグレーションファイル情報)**

```
typedef struct nqs_migfile {
    char path[NQS_LEN_PATHNAME + 1];    /* マイグレーションファイルの絶対パス
*/
# define MIG_TFL_CHKPF "chkpnt_files" /* チェックポイントファイル指定マーカー
*/
} nqs_migfile;
```

path はマイグレーション元ホストからマイグレーション先ホストへコピーされるユーザファイルまたはチェックポイントファイルの絶対パスが格納されます。パス名がディレクトリを指す場合は配下の全ファイルがコピー対象となります。例外として path に文字列 chkpnt_files が指定された場合は全てのチェックポイントファイル(チェックポイント実行時点でジョブがオープンしていたファイル群)がコピー対象となります。指定されたパスがマイグレーション先ホスト上に既に存在する場合、対象ファイルのコピーは実行されません。

- **nqs_migprm (マイグレーションパラメータ)**

```
typedef struct nqs_migprm {
    char if_hname[NQS_LEN_HOSTNAME + 1];  /* ファイル転送に用いるネットワーク
I/F のホスト名 */
    int sockbuf_sz;                       /* ソケットバッファサイズ */
    nqs_range iobuf_sz;                   /* ファイル I/O サイズ */
} nqs_migprm;
```

ジョブマイグレーションに伴う、ファイルの実行ホスト間コピーを制御するパラメータ。ファイルコピーは if_hname で指定されるネットワーク I/F を使って行われ、両端のソケットに対して sockbuf_sz で指定されるソケットバッファサイズが設定されま

す。 sockbuf_sz は 0 以上の整数値(バイト単位)で 0 は OS 既定値を意味します。iobuf_sz はディスク I/O、ソケット I/O 時に使用するバッファサイズ(バイト単位)が格納され、指定可能な値は 1 以上、8388608(8M バイト)以下です。

- **nqs_netreq (ネットワークリクエスト情報)**

```
typedef struct nqs_netreq {
    nqs_rid rid;                /* 親リクエスト ID */
    nqs_pdesc file;             /* 転送先/転送元ファイル名 */
    nqs_udesc user;             /* ネットワークリクエストの所有者 */
    int dir;                    /* ステージング方向 */
    int stgno;                  /* ステージングファイル番号 */
    int state;                  /* ネットワークリクエスト状態 */
    nqs_res res;                /* リザルトコード */
} nqs_netreq;
```

nqs_netreq 構造体はネットワークリクエストに関する情報を表し、rid はこのネットワークリクエストを生成したリクエストの ID が格納されます。 file にはクライアントホスト上のステージング対象ファイル名が格納され、dir が STAGE_IN の場合には転送元ファイル名を、STAGE_OUT の場合には転送先ファイル名をそれぞれ表します。state には現在のネットワークリクエスト状態が格納され、以下の値をとります。

REQST_QUEUED　ステージング開始待ち。
REQST_RUNNING ステージング実行中。
REQST_WAITING　リトライ待ち。

state が REQST_WAITING の場合、res にはステージングに失敗した原因が格納されます。

- **nqs_ngrpid (ノードグループID)**

```
typedef struct nqs_ngrpid {
    int type;                                   /* ノードグループタイプ */
    char name[NQS_LEN_NGRPNAME + 1];            /* ノードグループ名 */
} nqs_ngrpid;
```

name にはノードグループの名前が、type にはノードグループのタイプが格納されます。type として以下の値をとります。

type値	説明
NQS_NGRPTYPE_COMMON	一般ノードグループを表します。
NQS_NGRPTYPE_NWTOPOLOGY	ネットワークトポロジーを考慮したスケジューリングのためのノードグループ

- **nqs_nreqst (状態別リクエスト数)**

```
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;
```

リクエスト状態別に集計されたリクエスト総数が格納されます。

- **nqs_nsubreq (サブリクエスト数情報)**

```
typedef struct nqs_nsubreq {
    int total;              /* 総サブリクエスト数 */
    int active;             /* BSV に存在するサブリクエスト数 */
    int done;               /* 実行終了したサブリクエスト数 */
    char option[NQS_LEN_COMMAND+1] /* サブリクエスト番号指定文字列 */
} nqs_nsubreq;
```

- **nqs_odesc (オブジェクト記述子)**

```
typedef struct nqs_odesc {
    int obj_type;              /* オブジェクト種別 */
    union {
        nqs_schid schid;       /* スケジューラ ID */
        nqs_jsvid jsvid;       /* ジョブサーバ ID */
        nqs_hid hid;           /* ホスト ID */
        nqs_qid qid;           /* キューID */
        nqs_rid rid;           /* リクエスト ID */
        nqs_jid jid;           /* ジョブ ID */
        nqs_ngrpid ngrpid;     /* ノードグループ ID */
 nqs_wid wid;           /* ワークフローID */
    } obj;
} nqs_odesc;
```

バッチサーバ内での管理単位である、10 種類のオブジェクトを指定する際に使用されます。obj 共用体内の各 ID は obj_type で指定されたものが有効になります。obj_type に指定可能な値と参照される ID の対応は下表のとおりです。

obj_type値	参照されるID	説明
NQS_OBJ_BSV	(なし)	バッチサーバを表します
NQS_OBJ_SCH	obj.schid	スケジューラを表します
NQS_OBJ_JSV	obj.jsvid	ジョブサーバを表します
NQS_OBJ_HST	obj.hid	ホストを表します
NQS_OBJ_QUE	obj.qid	キューを表します
NQS_OBJ_REQ	obj.rid	リクエストを表します
NQS_OBJ_PRM	obj.rid	パラメトリックリクエストを表します
NQS_OBJ_JOB	obj.jid	ジョブを表します
NQS_OBJ_NGRP	obj.ngrpid	ノードグループを表します
NQS_OBJ_WFL	obj.wid	ワークフローを表します

- **nqs_ostemplate (OpenStack用テンプレート定義)**

```
typedef struct nqs_ostemplate {
    char template_name[NQS_LEN_TEMPLATENAME+1];  /* テンプレート名 */
    char image_name [NQS_LEN_VMIMGNAME+1];       /* OS イメージ名 */
    int cpunum;                                  /* CPU 数 */
    int memsz;                                   /* メモリ量 */
    int memunit;                                 /* メモリ量の単位 */
    int gpunum;                                  /* GPU 数 */
    char custom[NQS_LEN_TEMPLATECUSTOM+1];       /* 独自定義 */
    char comment[NQS_LEN_TEMPLATECOMMENT+1];     /* コメント */
    int lock;                                    /* ロック状態 */
    char flavor[NQS_LEN_FLAVORNAME+1];           /* フレーバ名 */
    int starttimeout;                            /* 起動見込時間 */
    int stoptimeout;                             /* 停止見込時間 */
} nqs_ostemplate;
```

OpenStack 用テンプレート定義の情報です。
ロック状態 lock は、TEMPLATE_LOCK または TEMPLATE_UNLOCK のいずれかの値となります。

- **nqs_pdesc (パス記述子)**

```
typedef struct nqs_pdesc {
    char path[NQS_LEN_PATHNAME + 1];  /* 絶対パス名 */
    char host[NQS_LEN_HOSTNAME + 1];  /* ホスト名 */
} nqs_pdesc;
```

ホスト host 上の絶対パス名が path であるファイルを表します。

path の要素として、以下のメタ文字が使用可能です。

%r	リクエスト ID(シーケンス番号.ホスト名)に展開されます。
%s	リクエスト ID 内のシーケンス番号に展開されます。
%m	リクエスト ID 内のマシン ID(整数値)に展開されます。
%j	ジョブ番号に展開されます。
%%	%に展開されます。

- **nqs_qdesc (キュー記述子)**

```
typedef struct nqs_qdesc {
    char name[NQS_LEN_QUENAME + 1];   /* キュー名 */
    char host[NQS_LEN_HOSTNAME + 1];  /* ホスト名 */
    int retry_mode;                   /* リトライモード・フラグ */
    time_t retry_at;                /* 次回のリトライ時刻 (EPOCH からの経過時間)
*/
    time_t retry_in;              /* リトライモード開始時刻 (EPOCH からの経過時間)
*/
} nqs_qdesc;
```

ホスト host 上のキュー名が name であるキューを表します。
属性を参照する際、対象が宛先キューで、かつ、retry_mode が非 0 の場合は、宛先キューがリトライモードに入っていることを表します。またその場合、retry_at には次に転送が試みられる時刻が、retry_in には最初にリトライモードに入った時刻がそれぞれ格納されます。
属性の変更時には、retry_xxx は無視されます。

- **nqs_qid (キューID)**

```
typedef struct nqs_qid {
    int type;                         /* キュー種別 */
    char name[NQS_LEN_QUENAME + 1];   /* キュー名 */
} nqs_qid;
```

type はキューの種別、name はキュー名です。
type は以下の値をとります。

QUETYP_EXECUTE	バッチキュー
QUETYP_INTERACTIVE	会話キュー
QUETYP_ROUTING	転送キュー
QUETYP_NETWORK	ネットワークキュー

- **nqs_qst (キュー状態)**

```
typedef struct nqs_qst {
    int state_run;      /* 実行許可 */
    int state_sub;      /* 投入許可 */
    int state_accept;   /* ACCEPT/REFUSE 状態 */
} nkqs_qst;
```

state_run はリクエストの実行許可状態を、state_sub は投入許可状態をそれぞれ表し

ます。取り得る値は以下のとおりです。

・　state_run
QUEST_ACTIVE 実行可能
QUEST_INACTIVE　実行禁止
・　state_sub
QUEST_ENABLE　　投入可能
QUEST_DISABLE　　投入禁止

属性の変更操作の際、state_run に QUEST_UNSPEC が指定されていた場合、実行許可状態は変更されません。同様に、state_sub が QUEST_UNSPEC であった場合は、投入許可状態は変更されません。

- **nqs_quecrinfo（キューのカスタムリソースの利用量情報）**

```
typedef struct nqs_quecrinfo {
    nqs_crid crid;                      /* カスタムリソース名 */
    int cr_std;                         /* 利用量の既定値 */
    nqs_hilo cr_limit;                  /* 利用量の指定範囲（上下限値） */
    int cr_permit_unused;               /* 利用量制御対象外指定の可否 */
} nqs_quecrinfo;
```

nqs_quecrinfo 構造体は、キューのカスタムリソースの利用量制御情報を保持します。cr_permit_unused はリクエスト投入時のカスタムリソースの利用量制御対象外指定を許可するか否かのフラグで、とりうる値は以下のいずれかです（マクロ定義）。

・　CR_PERMIT_UNUSED_YES　カスタムリソースの利用量制御対象外指定
を許可
・　CR_PERMIT_UNUSED_NO　カスタムリソースの利用量制御対象外指定
・　は不可

・ハイブリッドジョブにおけるジョブのグループ毎に異なる属性

```
typedef struct nqs_jobgroup {
    int start_jobno;     /*   属するジョブ番号の最小値     */
    int end_jobno;/*   属するジョブ番号の最大値     */
    nqs_rrange venum;      /*   ATTR_VENUM SCPE_JOB の値      */
    nqs_rlim cputim;   /*   ATTR_CPUTIM SCPE_JOB の値     */
    nqs_rnum cpunum;      /*   ATTR_CPUNUM SCPE_JOB の値     */
    nqs_rnum gpunum;      /*   ATTR_GPUNUM SCPE_JOB の値     */
    nqs_rlim memsz;   /*   ATTR_MEMSZ SCPE_JOB の値      */
    nqs_rlim vmemsz; /*   ATTR_VMEMSZ SCPE_JOB の値     */
    nqs_hca usehca;    /*   ATTR_HCA SCPE_JOB の値          */
    nqs_rlim cputimprc;     /*   ATTR_CPUTIM SCPE_PRC の値     */
    nqs_rlim memszprc;     /*   ATTR_MEMSZ SCPE_PRC の値      */
    nqs_rlim vmemszprc;   /*   ATTR_VMEMSZ SCPE_PRC の値     */
    nqs_rlim coresz;    /*   ATTR_CORESZ SCPE_PRC の値     */
    nqs_rlim stacksz;   /*   ATTR_STACKSZ SCPE_PRC の値  */
    nqs_rlim datasz;    /*   ATTR_DATASZ SCPE_PRC の値   */
    nqs_rnum filenum;      /*   ATTR_FILENUM SCPE_PRC の値  */
    nqs_rlim filesz;      /*   ATTR_FILESZ SCPE_PRC の値   */
    int advrsvid;   /*   ATTR_ADVRSVID  の値              */
    int njobs;              /*   属するジョブ数                  */
    nqs_rrange venode;      /*   VE 総数                              */
} nqs_jobgroup;
```

- **nqs_range (整数範囲)**

```
typedef struct nqs_range {
    unsigned int upper;          /* 上限値 */
    unsigned int curr;           /* 現在値 */
    unsigned int lower;          /* 下限値 */
} nqs_range;
```

upper は上限値、lower は下限値を表し、curr は現在値を表します。curr は常に upper 以下、lower 以上の値で、この範囲を越えることはできません。また、変更可能なのは curr のみで、upper、lower の変更はできません。

- **nqs_reqcrinfo (リクエストのカスタムリソースの利用量情報)**

```
typedef struct nqs_reqcrinfo {
    nqs_crid crid;                      /* カスタムリソース名 */
    int cr_req;                         /* リクエスト投入時に指定した利用量 */
    char cr_unit[NQS_LEN_CRUNIT+1];  /* カスタムリソースに設定された単位 */
} nqs_reqcrinfo;
```

nqs_reqcrinfo 構造体は、リクエストのカスタムリソースの利用量情報を保持します。crid で示すカスタムリソースについての利用量を cr_req に格納します。

・nqs_crusg（カスタムリソースの使用実績値)」

```
typedef struct nqs_crusg {
    nqs_crid crid;                      /* カスタムリソース名 */
    int      checkmode;                 /* 資源監視モード */
    double   usage;                     /* 使用実績値 */
    char     *unit[NQS_LEN_CRUNIT+1]; /* 単位 */
} nqs_crusg;
```

nqs_crusg 構造体は、カスタムリソースの使用実績値を保持します。crid はカスタムリソース名、checkmode は資源監視モード、usage は使用実績値、unit は単位が格納されます。

- **nqs_res (APIリザルトコード)**

```
typedef struct nqs_res {
    int err;                        /* エラー番号 */
    char msg[NQS_LEN_ERRMSG + 1];   /* エラーメッセージ */
} nqs_res;
```

API 関数のエラー内容を通知するための構造体であり、err にはエラー種別を表すエラー番号が、msg にはエラーの詳細内容を記述した文字列がセットされます。

- **nqs_rgrp (リクエストグループ指定子)**

```
typedef struct nqs_rgrp {
    nqs_rid rid;                /* リードリクエスト ID */
    int grpno;                  /* リクエストグループ番号 */
} nqs_rgrp;
```

リクエスト連携の際に自身が加わるリクエストグループを指定します。 rid はリードリクエスト(連携対象リクエストのうち最初に投入されるリクエスト)のリクエスト ID です。 grpno はリクエストグループ番号で 0 以上の整数値です。
リードリクエスト ID が同一のリクエスト群は、以下のルールに従って実行をスケジュールされます。

・　リードリクエストが含まれるリクエストグループ内の全リクエストが最初にスケジュール対象になる。
・　同一のリクエストグループ番号を持つリクエスト群は、同じタイミングでスケジュール対象になる。
・　異なるリクエストグループ番号を持つリクエスト間は、グループ番号がより小さいグループから順次スケジュール対象となる。ただし、先行するグループ内の全リクエストが終了するまで、後続のグループはスケジューリング対象にはならない。

リクエスト連携の対象リクエストで無い場合、grpno には負の値が格納されます。

- **nqs_rid (リクエストID)**

```
typedef struct nqs_rid {
    int seqno;              /* シーケンス番号 */
    int mid;                /* マシン ID */
    int subreq_no;          /* サブリクエスト番号 */
} nqs_rid;
```

seqno はリクエスト作成時にバッチサーバによって採番されるシーケンス番号、mid はバッチサーバホストのマシン ID です。
subreq_no には、パラメトリックリクエストの場合に、サブリクエスト番号が格納されます。ただし、パラメトリックリクエスト自身を示す場合は、subreq_no に-2、パラメトリックリクエストでない通常のリクエストの場合には subreq_no に-1 が格納されます。

リクエスト種別	subreq_noの値
通常のリクエスト（パラメトリックではないリクエスト）	-1
パラメトリックリクエスト（サブリクエストではなく全体を示す）	-2
パラメトリックリクエスト内の各サブリクエスト	>=0

- **nqs_rlim (サイズ/時間制限値)**

```
typedef struct nqs_rlim {
    int max_limit;          /* 上限値 */
    int max_unit;           /* 上限値の単位 */
    int cur_limit;          /* 警告値 */
    int cur_unit;           /* 警告値の単位 */
    int std_limit;          /* 標準値 */
    int std_unit;           /* 標準値の単位 */
} nqs_rlim;
```

max_limit と max_unit で制限の上限値を表します。同様に cur_xxx は警告値を、std_xxx は標準値を表します。 xxx_unit には以下の値を指定することが可能です。

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

xxx_limit が NQS_LIM_UNLIMITED の場合は無制限を意味します。属性値を変更する場合、xxx_limit に NQS_LIM_UNSPECIFIED が指定された値は変更対象になり

ません。上限値は変更せず、警告値、標準値のみを変更したい場合は、max_limit に NQS_LIM_UNSPECIFIED を指定して属性の変更操作を行ってください。警告値、標準値に関しても同様です。

std_xxx が意味を持つのはキュー資源制限に関する属性の場合のみです。他の属性では std_xxx は無視されます(参照時の値は不定です)。

- **nqs_rlim_g (グループ別サイズ/時間制限値)**

```
typedef nqs_rlim_g {
  nqs_gdesc group;          /* 制限対象のグループ */
  nqs_rlim rlim;            /* サイズ/時間制限値 */
} nqs_rlim_g;
```

グループ別に設定する属性値のサイズ/時間制限値を表します。

- **nqs_rlim_u (ユーザ別サイズ/時間制限値)**

```
typedef nqs_rlim_u {
  nqs_udesc user;           /* 制限対象のユーザ */
  nqs_rlim rlim;            /* サイズ/時間制限値 */
} nqs_rlim_u;
```

ユーザ別に設定する属性値のサイズ/時間制限値を表します。

- **nqs_rnum (個数制限値)**

```
typedef struct nqs_rnum {
    int max_limit;       /* 上限値 */
    int std_limit;       /* 標準値 */
} nqs_rnum;
```

max_limit は制限の上限値を表し、std_limit は標準値を表します。 xxx_limit が NQS_LIM_UNLIMIT の場合は無制限を意味します。属性値を変更する場合、xxx_limit に NQS_LIM_UNSPECIFIED が指定された値は変更対象になりません。上限値は変更せず、標準値のみを変更したい場合は、max_limit に NQS_LIM_UNSPECIFIED を指定して属性の変更操作を行ってください。標準値に関しても同様です。

std_limitが意味を持つのはキュー資源制限に関する属性の場合のみです。他の属性ではstd_limitは無視されます(参照時の値は不定です)。

- **nqs_hca (HCAポート数制限範囲)**

```
typedef struct nqs_hca {
    nqs_rrange for_io;     /* SCATEFS ダイレクト IO 用 HCA のポート数制限範囲 */
    nqs_rrange for_mpi;    /* MPI 用 HCA のポート数制限範囲 */
    nqs_rrange for_all;    /* SCATEFS ダイレクト IO 用と MPI 用の両方 HCA のポート
数制限範囲 */
} nqs_hca;
```

使用する HCA のポート数の範囲を HCA 種別ごとに表します。

- **nqs_rrange (資源制限範囲)**

```
typedef struct nqs_rrange {
    int min_limit;              /* 最小値 */
```

```
    int min_unit;               /* 最小値の単位 */
    int max_limit;              /* 最大値 */
    int max_unit;               /* 最大値の単位 */
    int warn_limit;             /* 警告値 */
    int warn_unit;              /* 警告値の単位 */
    int std_limit;              /* 標準値 */
    int std_unit;               /* 標準値の単位 */
} nqs_rrange;
```

資源制限範囲を表します。

- **nqs_rsgavg (アベレージ情報)**

```
typedef struct nqs_rsgavg {
    double avg01;           /* 最近 1 分間のアベレージ */
    double avg05;           /* 最近 5 分間のアベレージ */
    double avg15;           /* 最近 15 分間のアベレージ */
} nqs_rsgavg;
```

nqs_rsgavg には実行ホストのアベレージ情報が格納されます。
単位時間あたり(1、5、15 分)のロードアベレージ(実行待ちプロセス総数の平均値)または、CPU アベレージ(非アイドル CPU 時間と全 CPU 時間の比の平均値と実装 CPU 台数の積)を表します。

- **nqs_rsgres (リソース情報)**

```
typedef struct nqs_rsgres {
    int initial;            /* 割り当てられているリソース量 */
    int using;              /* 現在使用中のリソース量 */
    int maximum;            /* 使用可能な最大リソース量 */
    int unitsz;             /* リソース量の単位サイズ */
} nqs_rsgres;
```

nqs_rsgres には実行ホストのリソース情報が格納されます。
実行ホスト単位のリソース情報が格納されます。 initial と maximum は常に同じ値でありホストの物理メモリサイズ/スワップサイズがページ単位で格納されます。
using には現在使用中の物理メモリサイズ/スワップサイズがページ単位で格納されます。
対象リソースがメモリ/スワップの場合、unitsz にはページサイズがバイト単位で格納されます。 対象リソースが CPU 台数の場合、unitsz は常に 1 です。 属性種別が ATTR_RBCPUNM の場合、initial と maximum には OS が認識している CPU の実装台数が格納されます。using には現在使用中の CPU 台数(実装 CPU 台数と CPU 使用率の積。小数点以下切り上げ)が格納されます。

- **nqs_rsginfo (RSG情報)**

```
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;
```

実行ホストの RSG の設定情報。実行ホストが Linux の場合は、rsgno は 0 のみ、値は、rbspmem（メモリ量）、rbspswap（スワップ領域）、rbcpunum（CPU 数）、rbldavg（ロードアベレージ）、rbcpuavg（CPU アベレージ）、rbgpunum（GPU 数）のみ利用可。

- **nqs_rst (リクエスト状態)**

```
typedef struct nqs_rst {
    int state_curr;          /* 現在のリクエスト状態 */
    int state_prev;          /* 直前のリクエスト状態 */
    int state_reason;        /* 状態遷移理由 */
    time_t state_time;       /* 現在のリクエスト状態へ遷移した時刻 */
    int stalled;             /* リクエストがストールしている場合は真 */
    int exit_status;         /* リクエストの終了ステータス */
    int elaps_time;          /* リクエストの経過時間 */
    int deleted_by;          /* 削除実行時の権限 */
    nqs_res res;             /* リザルトコード */
    nqs_bsv bsv;             /* 転送先 BSV */
    int bsv_selected;        /* 実行 BSV 選択済みの場合は真 */
    int vm_ctrl;             /* OpenStack VM/Container の状態 (PRR,POR) */
} nqs_rst;
```

state_curr は現在のリクエスト状態が、state_prev には一つ前のリクエスト状態がセットされます。また、state_reason には状態遷移が発生した理由が格納され、エラーによる状態遷移だった場合には res にエラー種別を表すリザルトコードが、正常な状態遷移であった場合には res.err に NQS_ESUCCESS が代入されます。 state_time にはリクエストが現在の状態へ遷移した時刻(EPOCH からの経過秒)が設定されます。state_curr が REQST_POSTRUNNING の場合は、exit_status にリクエスト(マスタージョブ)の終了ステータスが格納されます。

リクエスト状態	
REQST_OUTSET	初期状態
REQST_ARRIVING	受信中
REQST_WAITING	実行時刻待ち
REQST_QUEUED	実行開始待ち
REQST_STAGING	ステージイン処理実行中
REQST_PRERUNNING	前処理実行中
REQST_RUNNING	実行中
REQST_POSTRUNNING	後処理実行中
REQST_EXITING	終了処理実行中

REQST_EXITED	終了状態
REQST_CHKPNTING	定期的チェックポイント実行中
REQST_HELD	ホールド状態
REQST_HOLDING	ホールド・チェックポイント実行中
REQST_RESTARTING	リスタート処理実行中
REQST_SUSPENDING	サスペンド処理実行中
REQST_SUSPENDED	サスペンド状態
REQST_RESUMING	リジューム処理実行中
REQST_MIGRATING	マイグレーション処理実行中
REQST_MOVED	移動完了

状態遷移理由	
REQRSN_DELETE	デリート要求
REQRSN_YET_EXETIME	実行時間未達
REQRSN_JUST_EXETIME	実行時間到達
REQRSN_ARRIVE	他キューからの受信
REQRSN_ARRIVE_SUCCESS	受信成功
REQRSN_ARRIVE_FAIL	受信失敗
REQRSN_SUBMIT	サブミット要求
REQRSN_STAGEIN	ステージイン要求
REQRSN_STAGEIN_SUCCESS	ステージイン成功
REQRSN_STAGEIN_FAIL	ステージイン失敗
REQRSN_PRERUN_SUCCESS	前処理成功
REQRSN_PRERUN_FAIL	前処理失敗
REQRSN_RUN	ラン要求
REQRSN_EXIT	実行終了
REQRSN_RERUN	リラン要求
REQRSN_POSTRUN_SUCCESS	後処理成功
REQRSN_POSTRUN_FAIL	後処理失敗
REQRSN_REQUE_SUCCESS	再キューイング
REQRSN_DONE	終了処理完了
REQRSN_CHKPNT	チェックポイント要求
REQRSN_CHKPNT_SUCCESS	チェックポイント成功
REQRSN_CHKPNT_FAIL	チェックポイント失敗
REQRSN_HOLD	ホールド要求
REQRSN_HOLD_SUCCESS	ホールド成功
REQRSN_HOLD_FAIL	ホールド失敗
REQRSN_RELEASE	リリース要求
REQRSN_RESTART	リスタート要求
REQRSN_RESTART_SUCCESS	リスタート成功
REQRSN_RESTART_FAIL	リスタート失敗
REQRSN_SUSPEND	サスペンド要求
REQRSN_SUSPEND_SUCCESS	サスペンド成功
REQRSN_SUSPEND_FAIL	サスペンド失敗

REQRSN_RESUME	リジューム要求
REQRSN_RESUME_SUCCESS	リジューム成功
REQRSN_RESUME_FAIL	リジューム失敗
REQRSN_MIGRATE	マイグレーション要求
REQRSN_MIGRATE_SUCCESS	マイグレーション成功
REQRSN_MIGRATE_FAIL	マイグレーション失敗
REQRSN_MOVE	リクエスト移動要求
REQRSN_MOVED	他キューからの移動
REQRSN_SYSTEM_FAILURE	実行ホストのシステム障害
REQRSN_ROLLBACK	ロールバック要求
REQRSN_ALLEXITED	全サブリクエスト終了
REQRSN_FORCELOCAL	強制ローカル実行
REQRSN_EXIT_FAIL	サブリクエスト削除失敗
REQRSN_MOVE_FAIL	リクエスト移動失敗
REQRSN_STAGEOUT_FAIL	外部ステージング失敗

リクエスト状態に関する詳細は、第 2 章.リクエストの状態遷移を参照してください。

bsv は、グローバルリクエストにおける転送先の BSV 用でしたが、現在は使用されていません。

vm_ctrl は、OpenStack による仮想マシンまたは Docker によるコンテナが、起動・停止中か否かを示します。

- 1 の場合：OpenStack による仮想マシンが起動・停止中
- 2 の場合：Docker によるコンテナが起動・停止中
- 0 の場合：上記以外

- **nqs_rstf (リスタートファイル情報)**

```
typedef struct nqs_rstf {
    time_t date;                /* 作成日時 */
    long long size;             /* ファイルサイズ */
    int intval;                 /* 定期的チェックポイントの間隔 */
    char rstfdir[NQS_LEN_RSTFPATH + 1];  /* リスタートファイル格納ディレクトリ
*/
} nqs_rstf;
```

ジョブのリスタートファイルに関する情報が格納されます。 intval が 0 の場合は定期的チェックポイントが OFF になっていることを表します。 time が 0 の場合、リスタートファイルは存在しません。

- **nqs_schid (スケジューラID)**

```
typedef struct nqs_schid {
    int schno;              /* バッチスケジューラ番号 */
} nqs_schid;
```

schno はバッチスケジューラ番号で範囲 0～NQS_MAX_SCHNO の整数値です。

- **nqs_socket (ソケットリソース情報)**

```
typedef struct nqs_socket {
    int no;                             /* ソケット番号 */
    int initial_cpu;                    /* CPU 搭載量 */
    int using_cpu;                      /* CPU 使用量 */
    long long initial_mem;              /* メモリ搭載量 */
    long long using_mem;                /* メモリ使用量 */
    char cpus[NQS_LEN_CPUS+1];          /* コア番号 */
    char mems[NQS_LEN_MEMS+1];          /* メモリノード番号 */
} nqs_socket;
```

nqs_socket 構造体は、ソケット番号毎のソケットリソース情報を保持します。

- **nqs_stgfile (ステージングファイル情報)**

```
typedef struct nqs_stgfile {
    int dir;                                /* ステージング方向 */
    char cli_host[NQS_LEN_HOSTNAME + 1]; /* クライアントホスト名 */
    char cli_path[NQS_LEN_PATHNAME + 1]; /* クライアントホスト上での絶対パス名
*/
    char exe_path[NQS_LEN_PATHNAME + 1]; /* 実行ホスト上での相対パス名 */
    char exe_jobno[NQS_LEN_JOBNODSC + 1];/* 転送対象ジョブのジョブ番号 */
    int stgno;                              /* ステージングファイル番号 */
    int status;                             /* ステージング状況 */
} nqs_stgfile;
```

nqs_stgfile 構造体はステージング対象ファイルに関する情報を表します。 dir にはステージングの方向が格納され、値が STAGE_IN の場合はクライアントホストから実行ホストへの(ステージイン動作)、STAGE_OUT の場合はその逆への（ステージアウト動作)ファイルコピーを表します。
ステージインの場合、クライアントホスト cli_host 上のファイル cli_path は、exe_jobno で指定されたジョブが実行される各実行ホスト上のファイル exe_path へコピーされます。
ステージアウトの場合、exe_jobno で指定されたジョブが実行される各実行ホスト上のファイル exe_path は一つのファイルにアペンドされ、クライアントホスト cli_host

上のファイル cli_path へコピーされます。
exe_jobno はステージング対象となるジョブのジョブ番号です。 exe_jobno には以下の文字列が指定可能です。

(1)　数字の単独指定　　（例：0 ）
(2)　"ALL"による全ジョブ指定　（例：ALL）
(3)　","による複数指定　　（例：0,2,5）
(4)　"-"による連続の番号指定　（例：0-4 ）
(5)　(3),(4)の組合せ指定　　（例：0,2,4-6）

cli_path は絶対パスで、exe_path は相対パスで指定する必要があります。ジョブ中からの exe_path に対する I/O では、環境変数 STGDIR を起点とした相対パスとしてアクセスする必要があります。
cli_path、または exe_path の最後の文字が'/'の場合、指定されたパスはディレクトリを表します。
cli_path の要素として、以下のメタ文字が使用可能です。

%r　　リクエスト ID(シーケンス番号.ホスト名)に展開されます。
%s　　リクエスト ID 内のシーケンス番号に展開されます。
%m　　リクエスト ID 内のマシン ID(整数値)に展開されます。
%j　　ジョブ番号に展開されます。
%%　　%に展開されます。

stgno はステージイン/アウト対象ファイル毎に付加される通し番号です。
status は現在のステージング状況を表し、取りうる値は下表のとおりです。

STGFST_NOTYET	ステージングはまだ実行されていません。
STGFST_PRGRESS	ステージングは現在実行中です。
STGFST_SUCCESS	ステージングは成功しました。
STGFST_FAILURE	ステージングは失敗しました。

- **nqs_temp_reqs（テンプレートを使用中のリクエスト数）**

```
typedef struct nqs_temp_reqs {
    char template_name[NQS_LEN_TEMPLATENAME+1];  /* テンプレート名 */
    int requests;                                /* 使用中のリクエスト数 */
} nqs_temp_reqs;
```

template_name にテンプレート名が、requests にそのテンプレートを使用しているリクエストの数が格納されます。

- **nqs_template (テンプレート情報)**

```
typedef struct nqs_template {
    int type;                              /* テンプレートタイプ */
    union {
        struct nqs_vmtemplate vm_tmpl;
        struct nqs_ostemplate os_tmpl;  /* OpenStack テンプレート定義 */
        struct nqs_cotemplate co_tmpl;  /* コンテナ用テンプレート定義 */
    }t1;
} nqs_template;
```

テンプレートの種類を type で、テンプレートの情報を以下の構造体で指定します。

テンプレートの種類	type	テンプレート情報の構造体
OpenStack用	NQSII_TEMPLATE_TYPE_OPENSTACK	struct nqs_ostemplate os_tmpl
コンテナ用	NQSII_TEMPLATE_TYPE_CONTAINER	struct nqs_cotemplate co_tmpl

- **nqs_udesc (ユーザ記述子)**

```
typedef struct nqs_udesc {
    char name[NQS_LEN_USERNAME + 1];  /* ユーザ名 */
    uid_t uid;                        /* ユーザ ID */
    gid_t gid;                        /* グループ ID */
} nqs_udesc;
```

ユーザ名が name、ユーザ ID/グループ ID がそれぞれ uid/gid であるユーザを表します。

- **nqs_uexit (ユーザEXIT情報)**

```
#define UEXNUM_SCR      4           /* ロケーション当りの最大スクリプト数 */

typedef struct nqs_uexit {
    int location;                        /* ユーザ EXIT が起動されるロケーション
*/
    struct {
        int order;                      /* ユーザ EXIT スクリプトの実行順序 */
        char name[NQS_LEN_FILENAME+1]; /* ユーザ EXIT スクリプトのファイル名 */
    } uexscr[UEXNUM_SCR];
} nqs_uexit;
```

location で指定される状態遷移タイミング（ロケーション）において実行される、ユーザ EXIT に関する情報が格納されます。 location として指定可能な値は以下のとおりです。

UEXLOC_PRERUN	実行開始直前［PRE-RUNNING状態中］
UEXLOC_PSTRUN	実行終了直後［POST-RUNNING状態中］
UEXLOC_HLDING	終了型チェックポイント処理完了直後［HOLDING状態中］
UEXLOC_RSTING	リスタート処理実行前［RESTARTING状態中］

4 つの要素からなる uexscr 構造体の配列には、指定ロケーションで実行されるユーザ EXIT スクリプトの情報が各配列要素毎に格納されます(最大 4 スクリプトまで)。uexscr 構造体では、メンバ order がスクリプトの実行順序を、メンバ name がスクリプトのファイル名を表します。

order に指定可能な値は以下のとおりです。

UEXODR_NON	実行対象にはなりません。
UEXODR_1ST	ロケーション内で1番目に実行されます。
UEXODR_2ND	ロケーション内で2番目に実行されます。
UEXODR_3RD	ロケーション内で3番目に実行されます。
UEXODR_4TH	ロケーション内で4番目に実行されます。

name にはユーザ EXIT スクリプトのファイル名が格納されます。 name に格納されるファイル名はバッチサーバホスト上の/opt/nec/nqsv/sbin/uex_prog 直下に存在するとみなされます。また、name 中にパス要素('/')を含めることはできません。
order が UEXODR_NON に設定された場合、対応する name の値は不定です。

- **nqs_upp (ユーザPPスクリプト情報)**

```
typedef struct nqs_upp {
    int location;                /* ユーザ PP スクリプトを起動するロケーション*/
    char path[NQS_LEN_PATHNAME+1];            /* ユーザ PP スクリプトのパス名 */
} nqs_upp;
```

nqs_upp 構造体は、location の状態遷移タイミング（ロケーション）で実行する、ユーザ PP スクリプトに関する情報を格納します。location がとりうる値は以下のいずれかです。

UEXLOC_PRERUN	実行開始直前［PRE-RUNNING状態中］
UEXLOC_PSTRUN	実行終了直後［POST-RUNNING状態中］

- **nqs_wid (ワークフローID)**

```
typedef struct nqs_wid {
    int id;                  /* ワークフローシーケンス番号 */
    int mid;                 /* マシン ID */
} nqs_wid;
```

id はワークフロー生成時にバッチサーバによって採番されるシーケンス番号、mid はバッチサーバホストのマシン ID です。

第6章　API 関数

6.1　属性リスト関数

6.1.1　属性リストの作成

【名前】NQSalist -- 属性リストの作成
【形式】

```
#include "nqsv.h"
nqs_alist NQSalist(nqs_alist ad, nqs_aid *aid, nqs_res *res)
```

【機能説明】

属性リストを作成します。 属性リスト識別子 *ad* が負の整数の場合、属性ヘッダ *aid* を先頭要素とした新たな属性リストを作成します。 *ad* が 0 以上の整数の場合、*ad* で指定される既存の属性リストへ *aid* を追加します。

【戻り値】

正常に処理が完了すると、NQSalist()は属性リスト識別子である 0 以上の整数値を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_EEXIST	aidで指定された属性ヘッダが属性リスト内に既に存在しています。
NQS_ENOENT	adで指定された属性リストが存在しません。
NQS_EINVAL	不正な引数が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。

【関連項目】

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

6.1.2 属性リストの解放

【名前】NQSafree -- 属性リストの解放
【形式】

```
#include "nqsv.h"
int NQSafree( nqs_alist ad, nqs_aid *aid, nqs_res *res)
```

【機能説明】

属性リスト識別子 *ad* で指定される属性リストを解放します。 *aid* が NULL で無い場合は、*aid* と一致する属性のみを属性リストから削除します。 *aid* が NULL の場合はリスト中の全属性を削除し属性リストそのものを解放します。

【戻り値】

正常に処理が完了すると、NQSafree()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOENT	adに一致する属性リストが存在しません。 aidで指定された属性ヘッダが属性リスト内に存在しません。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.1.3 属性リストへの属性値の追加

【名前】NQSaadd -- 属性リストへの属性値の追加
【形式】

```
#include "nqsv.h"
int NQSaadd(nqs_alist ad, nqs_aid *aid, void *val, size_t sz, nqs_res *res)
```

【機能説明】

属性リスト識別子 *ad* で指定される属性リスト内で、*aid* と一致する属性ヘッダ配下に属性値 *val* を追加します。 属性ヘッダが既に属性値を持っていた場合、*val* を属性値チェインの最後尾に追加します。 *sz* には *val* のサイズをバイト単位で指定します。

val の型は属性値ごとに異なります。詳細は属性一覧を参照してください。

【戻り値】

正常に処理が完了すると、NQSaadd()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOENT	adに一致する属性リストが存在しません。 aidで指定された属性ヘッダが属性リスト内に存在しません。
NQS_EINVAL	不正な引数が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。

【関連項目】

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

6.1.4 属性リスト内の属性値の削除

【名前】NQSadel -- 属性リスト内の属性値の削除
【形式】

```
#include "nqsv.h"
int NQSadel(nqs_alist ad, nqs_aid *aid, void *val, nqs_res *res)
```

【機能説明】

属性リスト識別子 *ad* で指定される属性リスト内で、*aid* に一致する属性の属性値を削除します。*val* が NULL の場合は全ての属性値を、非 NULL の場合は *val* の内容と一致する属性値のみを削除します。 削除の対象となる属性値が存在しない場合は単に正常終了します。

【戻り値】

正常に処理が完了すると、NQSadel()は 0 を返します。エラーが発生した場合は負の整数が返され、API リザルトコードが *res* に設定されます。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOENT	adに一致する属性リストが存在しません。 aidで指定された属性ヘッダが属性リスト内に存在しません。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.1.5　属性リスト内の属性値の参照

【名前】NQSaref -- 属性リスト内の属性値の参照
【形式】

```
#include "nqsv.h"
void *NQSaref( nqs_alist ad, nqs_aid *aid, void *val, nqs_res *res)
```

【機能説明】

属性リスト識別子 *ad* で指定される属性リスト内で、*aid* に一致する属性の属性値を指すポインタを返却します。 *val* が NULL の場合は属性ヘッダ直下の属性値へのポインタを、非 NULL の場合は *val* が指す属性値の直下に位置する属性値へのポインタを返却します。
複数の属性値がチェインしている場合、最初の NQSaref()呼び出しで返されたポインタを次回呼び出し時に *val* に指定することで、チェインしている全ての属性値を参照することができます。

【戻り値】

正常に処理が完了すると、NQSaref()は属性リスト内の属性値を指すポインタを返します。 *aid* で指定された属性が属性値を持っていない場合は NULL を返却し、NQS_EEMPTY をエラー番号にセットします。 属性値チェインの最後尾に位置する属性値を *val* に指定した場合は NULL を返し、エラー番号には NQS_EALLOVER をセットします。 エラーが発生した場合は NULL を返し、API リザルトコードを *res* に設定します。
NQSaref()は void 型ポインタを返しますので、参照の際には属性値ごとに決められた型でキャストしてください。 属性値の型に関する詳細は属性一覧を参照してください。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOENT	adに一致する属性リストが存在しません。 aidで指定された属性ヘッダが属性リスト内に存在しません。
NQS_EEMPTY	指定された属性は属性値を持っていません。
NQS_EALLOVER	属性値チェインの最後尾を越えて読み出そうとしました。
NQS_EJSVDOWN	実行ホスト関連の属性の場合、ジョブサーバがダウンのため属性値の参照ができない場合があります。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.1.6 属性エントリの操作

【名前】NQSopenattr, NQSreadattr, NQScloseattr -- 属性エントリの操作
【形式】

```
#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)
```

【機能説明】

NQSopenattr()は、*object* で指定したオブジェクト(ジョブサーバやリクエスト等)の中から、*odesc* で指定された親オブジェクトに関連するオブジェクトを選別し、各オブジェクトに対して *ad* で指定した属性リストを作成した後、属性値を一括取得します。
そしてその属性リストを要素に持つ一覧表(属性エントリ)を作成して内部インデックスを 0 に初期化した後、それを識別するためのエントリ識別子を返却します。
エラーが発生した場合には負の整数を返し、API リザルトコードを *res* に設定します。
odesc、および *object* で指定可能なオブジェクトとその組み合わせは下表のとおりです。

odesc.obj_typeの値	参照objメンバ	OBJECTの値	対象
NQS_OBJ_BSV	なし	NQS_OBJ_SCH	システム内の全スケジューラ
		NQS_OBJ_HST	システム内の全実行ホスト
		NQS_OBJ_JSV	システム内の全ジョブサーバ
		NQS_OBJ_QUE	システム内の全キュー
		NQS_OBJ_REQ	システム内の全リクエスト
		NQS_OBJ_BREQ	システム内の全バッチリクエスト
		NQS_OBJ_IREQ	システム内の全会話リクエスト
		NQS_OBJ_PRM	システム内の全パラメトリックリクエスト
		NQS_OBJ_JOB	システム内の全ジョブ
		NQS_OBJ_NGRP	システム内の全ノードグループ
NQS_OBJ_SCH	odesc.obj.schid	NQS_OBJ_QUE	指定のスケジューラにバインドしている全キュー
NQS_OBJ_JSV	odesc.obj.jsvid	NQS_OBJ_JOB	指定のジョブサーバが管理する全ジョブ
		NQS_OBJ_QUE	指定のジョブサーバとバインドしている全キュー
		NQS_OBJ_NGRP	指定のジョブサーバ含む全ノードグループ
NQS_OBJ_HST	odesc.obj.hid	NQS_OBJ_JSV	指定の実行ホスト上にある全ジョブサーバ
NQS_OBJ_QUE	odesc.obj.qid	NQS_OBJ_JSV	指定のキューにバインドしている全ジョブサーバ

		NQS_OBJ_REQ	指定のキューに投入されている全リクエスト
		NQS_OBJ_PRM	指定のキューに投入されている全パラメトリックリクエスト
		NQS_OBJ_NGRP	指定のキューにバインドしている全ノードグループ
NQS_OBJ_REQ	odesc.obj.rid	NQS_OBJ_JOB	指定のリクエストを親に持つ全ジョブ
NQS_OBJ_NGRP	odesc.obj.ngrpid	NQS_OBJ_JSV	指定のノードグループに属する全 JSV
		NQS_OBJ_HST	指定のノードグループに属する全実行ホスト
		NQS_OBJ_QUE	指定のノードグループをバインドしている全キュー
NQS_OBJ_PRM	odesc.obj.rid	NQS_OBJ_REQ	パラメトリックリクエスト内の全サブリクエスト

NQSreadattr()は、内部インデックスが指す属性エントリを nqs_alist 型で返し、内部インデックスを一つインクリメントします。 エントリの最後に達した場合は-1 を返し、*res* 内のエラー番号に NQS_EALLOVER を設定します。 エラーが発生した場合には-1 を返し、API リザルトコードを *res* に設定します。
NQSreadattr()が返した nqs_alist 型変数は属性リスト識別子ですので、NQSalist()を使って各属性値を参照することができます。

NQScloseattr()は、NQSopenattr()が作成した属性エントリを削除します。処理に成功した場合は 0 を、エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【注意事項】

NQScloseattr()の実行後に、NQSreadattr()が返した属性リスト識別子を参照した場合の動作は保証されません。
参照権のない属性は取得されず、単に無視されます(エラーにはなりません)。

本関数を使った属性取得は、NQSopen{jsv,hst,que,req,job}関数と NQSattr{jsv,hst,que,req,job}関数を組み合わせた場合とほぼ同じ機能になりますが、動作速度はより高速です

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_ENOENT	無効なエントリ識別子が指定されました。 無効な属性リスト識別子が指定されました。
NQS_EUNKNOWN	未知の属性種別が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3), NQSaref(3)

6.2　API初期化／終了関数

6.2.1　API リンクのオープン

【名前】NQSconnect -- API リンクのオープン
【形式】

```
#include "nqsv.h"
int NQSconnect( char *hostname, int port, int priv, nqs_res *res)
```

【機能説明】

ホスト名が *hostname* であるバッチサーバホスト上のバッチサーバとの API リンクを確立します。*hostname* が NULL の場合、/etc/opt/nec/nqsv/api_client.conf 内にバッチサーバホスト名の指定があればそれを使用し、指定がなければ localhost への接続を試みます。
port が 1 以上の場合、バッチサーバとの接続に *port* で指定されたポート番号を用います。*port* が 0 の場合、api_client.conf 内に指定があれはそれを使用し、なければデフォルトのポート番号を使って接続を試みます。

priv にはユーザ権限を指定します。ユーザ権限により使用可能な API 機能が制限されます。ユーザ権限には、以下の 6 種類があります。

PRIV_SCH	スケジューラ権限
PRIV_MGR	管理者権限
PRIV_OPE	操作員権限
PRIV_GMGR	グループ管理者権限
PRIV_SPU	特別利用者権限
PRIV_USR	一般利用者権限

ユーザ権限は PRIV_SCH がもっとも高く、PRIV_USR が最低です。 PRIV_SCH はスケジューラ用の権限で全ての API 関数を使用できます。PRIV_MGR、PRIV_OPE は一部の機能が使用できません。PRIV_GMGR は管理するグループの範囲に限定した管理者の権限です。またもっとも権限の低い PRIV_USR は一般ユーザレベルの権限であり、NQS 操作員以上の権限が必要な機能は使用できません。PRIV_SPU は、PRIV_USR の権限に加えて他人のリクエスト情報やジョブ情報等が参照可能です。

【注意事項】

NQSconnect()が返したソケット記述子は NQSdisconnect()実行時に API 内部でクローズされます。これ以外の方法でソケット記述子クローズした場合、それ以降の API の動作は保証されません。

【戻り値】

正常に処理が完了すると、NQSconnect()は API イベント監視用のソケット記述子を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ELICENSE	APIリンクに必要なライセンスが取得できませんでした。
NQS_EISCONN	APIリンクは既に確立されています。

NQS_ENETDB	指定されたホスト名を持つホストが見つかりませんでした。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_ESYSCAL	システムコール内でエラーが発生しました。
NQS_EPERM	指定されたユーザ権限による接続がバッチサーバによって拒否されました。
NQS_EACCTAUTH	未知のユーザ名によるアクセスを拒否しました。
NQS_EINVAL	不正な引数が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。

【ファイル】

/etc/opt/nec/nqsv/api_client.conf　　NQSV/API クライアント設定ファイル

【関連項目】

NQSdisconnect(3), NQSevent(3)

6.2.2 API リンクのクローズ

【名前】NQSdisconnect -- API リンクのクローズ
【形式】

```
#include "nqsv.h"
int NQSdisconnect( nqs_res *res)
```

【機能説明】
API リンクを切断します。

【戻り値】
正常に処理が完了すると、NQSdisconnect()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】
エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。

【関連項目】
NQSconnect (3)

6.3 APIイベント関連関数

6.3.1 API イベントの取得

【名前】NQSevent -- API イベントの取得
【形式】

```
#include "nqsv.h"
int NQSevent( nqs_event *event, nqs_res *res)
NQSEVT_TYPE( event_id )
```

【機能説明】

NQSevent()は受信した API イベントを一つ読み出し、*event* に格納します。
NQSEVT_TYPE()は指定されたイベント識別子 *event_id* のイベントタイプを返すマクロです。

【注意事項】

イベントに含まれる属性値は、API 内部に作成される専用の属性リストに格納されます。 この属性リストは属性値を含んだイベントを NQSevent()で読み出す度に新規に作成されます(一つ前の属性リストはその時点で破棄されます)。

【戻り値】

正常に処理が完了すると、NQSevent()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_ENOEVENT	イベントは受信されていません。
NQS_EUNKNOWN	イベント内に未知の属性が含まれていました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSevflt(3)

6.3.2 イベントフィルタの設定

【名前】NQSevflt -- イベントフィルタの設定
【形式】

```
#include "nqsv.h"
int NQSevflt( int event_type, int op, nqs_res *res)
```

【機能説明】

NQSevflt()は API クライアントが受信すべきイベントタイプを取捨選択するためのイベントフィルタを API リンクに設定します。 対象となるイベントタイプを *event_type* で指定し、*op* の動作に従ってフィルタを設定します。*op* に指定できる値は以下のとおりです。

EVFLT_ADD　イベントフィルタへ *event_type* を追加します。
EVFLT_DEL　イベントフィルタから *event_type* を削除します。

【戻り値】

正常に処理が完了すると、NQSevflt()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EEXIST	追加しようとしたイベントタイプは登録済です。
NQS_ENOENT	削除しようとしたイベントタイプは未登録です。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSevent(3)

6.4　スケジューラ関連関数

6.4.1　スケジューラ ID の登録

【名前】NQSregsch -- スケジューラ ID と名前の登録
【形式】

```
#include "nqsv.h"
int NQSregsch( nqs_schid *schid, char *name, char *version nqs_res *res)
```

【機能説明】

NQSregsch()を呼び出したスケジューラ自身の ID と名前をバッチサーバに登録します。 既に登録済の ID は変更できません。 また、他のスケジューラが使用している ID は指定できません。
スケジューラ ID 内のスケジューラ番号(nqs_schid.schno)に指定可能な値は 0～NQS_MAX_SCHNO の整数値です。

name にはスケジューラ名を表す任意の文字列(最大長は NQS_LEN_SCHNAME)を指定できます。*name* が NULL の場合はバッチサーバが適当な名前をつけます。

version にはスケジューラのバージョンを表す任意の文字列(最大長は NQS_LEN_VERSION)を指定する事が可能です。

【注意事項】

NQSregsch()の実行には PRIV_SCH 以上のユーザ権限が必要です。

【戻り値】

正常に処理が完了すると、NQSregsch()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ETOOLONG	*name*で指定されたスケジューラ名が長すぎます。
	*version*で指定されたバージョン文字列が長すぎます。
NQS_EOUTRNG	指定されたスケジューラ番号は有効範囲外です。
NQS_EEXIST	指定されたIDは他のスケジューラが使用中です。
NQS_EALREADY	指定されたスケジューラには既にIDが登録されています。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3)

6.4.2 スケジューラエントリの操作

【名前】NQSopensch, NQSreadsch, NQSrewindsch, NQSclosesch -- スケジューラエントリの操作

【形式】

```
#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 )
```

【機能説明】

NQSopensch()は、バッチサーバにリンクしているスケジューラの中から、*odesc* で指定されたオブジェクトに関連するスケジューラを選別します。 そして、そのスケジューラ ID を要素に持つ一覧表(スケジューラエントリ)を作成して内部インデックスを 0 に初期化した後、それを識別するためのエントリ識別子を返却します。 エラーが発生した場合には負の整数を返し、API リザルトコードを *res* に設定します。

odesc で指定可能なオブジェクトは下表のとおりです。

odesc.obj_typeの値	参照objメンバ	対象
NQS_OBJ_BSV	なし	システム内の全スケジューラ

NQSreadsch()は、内部インデックスが指すエントリを nqs_schid 型ポインタで返し、内部インデックスを一つインクリメントします。エントリの最後に達した場合は NULL を返し、*res* 内のエラー番号に NQS_EALLOVER を設定します。エラーが発生した場合には NULL を返し、API リザルトコードを *res* に設定します。

NQSrewindsch()は内部インデックスを 0 に再初期化します。エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

NQSclosesch()は、NQSopensch()が作成したジョブサーバエントリを削除します。処理に成功した場合は 0 を、エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【注意事項】

NQSclosesch()の実行後に、NQSreadsch()が返したスケジューラ ID を指すポインタを参照した場合の動作は保証されません。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_ENOENT	無効なエントリ識別子が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSattrsch(3)

6.4.3 スケジューラ属性操作関数

【名前】NQSattrsch -- スケジューラ属性操作関数
【形式】

```
#include "nqsv.h"
int NQSattrsch( nqs_schid *schid, nqs_alist ad, int op, nqs_res *res)
```

【機能説明】

スケジューラ ID が *schid* であるスケジューラの属性を操作します。 属性リスト *ad* に含まれる属性について、*op* で指定された属性操作が実行されます。 *op* に指定可能な値は以下のとおりです。

ATTROP_GET　*ad* 内の属性へスケジューラ属性をコピーします。

【戻り値】

正常に処理が完了すると、NQSattrsch()は 0 を返します。エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_ENOENT	指定された属性リストが存在しません。
NQS_ENOSCH	指定されたスケジューラが存在しません。
NQS_EUNKNOWN	未知の属性種別が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.5 バッチサーバ関連関数

6.5.1 バッチサーバ属性操作関数

【名前】NQSattrbsv -- バッチサーバ属性操作関数
【形式】

```
#include "nqsv.h"
int NQSattrbsv( nqs_alist ad,  int op,  nqs_res *res)
```

【機能説明】

バッチサーバの属性を操作します。 属性リスト *ad* に含まれる属性について、*op* で指定された属性操作が実行されます。 *op* に指定可能な値は以下のとおりです。

ATTROP_GET　*ad* 内の属性へバッチサーバ属性をコピーします。
ATTROP_SET　*ad* 内の属性へバッチサーバ属性を置き換えます。

【戻り値】

正常に処理が完了すると、NQSattrbsv()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_ENOENT	指定された属性リストが存在しません。
NQS_ERANGE	属性値が設定可能な範囲を越えています。
NQS_EUNKNOWN	未知のバッチサーバ属性が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.5.2　バッチサーバの停止

【名前】NQSshutbsv -- バッチサーバの停止
【形式】

```
#include "nqsv.h"
int NQSshutbsv( nqs_res *res )
```

【機能説明】
バッチサーバを停止(シャットダウン)します。

【戻り値】
正常に処理が完了すると、NQSshutbsv()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】
エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。

【関連項目】
NQSconnect (3)

6.5.3　実行ホストの登録・削除

【名前】NQSattachhst, NQSdetachhst -- バッチサーバへの実行ホストの登録、削除
【形式】

```
#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 )
```

【機能説明】

NQSattachhst()は、*hid* で指定された実行ホストを *jsvid* に指定したジョブサーバ番号とあわせて、実行ホストとしてバッチサーバに登録します。実行ホストの登録は、ジョブサーバが立ち上がっていない状態でも実行が可能です。既に登録されている実行ホストに、指定された *hid* または *jsvid* が一致するものがある場合は、エラーとなります。

NQSdetachhst()は、*hid* で指定された実行ホストまたは *jsvid* に指定したジョブサーバ番号に対応する実行ホストを、バッチサーバの登録実行ホストから削除する。実行ホストの指定は、*hid*、*jsvid* のいずれかを設定し、指定しない方には NULL を指定設定してください。
hid として ip.s_addr＝0 を指定し、かつ、*jsvid* として JSV 番号＝－1 を指定した場合は、全実行ホストの登録を削除します。
既に登録されている実行ホストに、指定された *hid* または *jsvid* が一致するものがない場合は、エラーとなります。
指定した実行ホスト上にジョブが存在する場合は、エラーとなります。

optの値	動作
DETACH_FLAG_DEFAULT	実行ホスト上にジョブが存在する場合､削除を行わず、エラーとする。

【注意事項】

NQSattachhst(),NQSdetachhst()の実行には PRIV_MGR 以上のユーザ権限が必要です。

【戻り値】

正常に処理が完了すると、NQSattachhst(), NQSdetachhst()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。

【関連項目】

NQSconnect (3)

6.6　ジョブサーバ関連関数

6.6.1　ジョブサーバエントリの操作

【名前】NQSopenjsv, NQSreadjsv, NQSrewindjsv, NQSclosejsv -- ジョブサーバエントリの操作

【形式】

```
#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 )
```

【機能説明】

NQSopenjsv()は、バッチサーバにリンクしているジョブサーバの中から、*odesc* で指定されたオブジェクトに関連するジョブサーバを選別します。 そして、そのジョブサーバ ID を要素に持つ一覧表(ジョブサーバエントリ)を作成して内部インデックスを 0 に初期化した後、それを識別するためのエントリ識別子を返却します。エラーが発生した場合には負の整数を返し、API リザルトコードを *res* に設定します。

odesc で指定可能なオブジェクトは下表のとおりです。

odesc.obj_typeの値	参照objメンバ	対象
NQS_OBJ_BSV	なし	接続BSV内の全ジョブサーバ
NQS_OBJ_HST	odesc.obj.hid	指定の実行ホスト上に存在する全ジョブサーバ
NQS_OBJ_QUE	odesc.obj.qid	指定のキューにバインドしている全ジョブサーバ

NQSreadjsv()は、内部インデックスが指すエントリを nqs_jsvid 型ポインタで返し、内部インデックスを一つインクリメントします。 エントリの最後に達した場合は NULL を返し、*res* 内のエラー番号に NQS_EALLOVER を設定します。 エラーが発生した場合には NULL を返し、API リザルトコードを *res* に設定します。

NQSrewindjsv()は内部インデックスを 0 に再初期化します。 エラーが発生した場合は負の整数を返し、API リザルトコードを res に設定します。

NQSclosejsv()は、NQSopenjsv()が作成したジョブサーバエントリを削除します。処理に成功した場合は 0 を、エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【注意事項】

NQSclosejsv()の実行後に、NQSreadjsv()が返したジョブサーバ ID を指すポインタを参照した場合の動作は保証されません。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_ENOENT	無効なエントリ識別子が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSattrjsv(3)

6.6.2 ジョブサーバ属性操作関数

【名前】NQSattrjsv -- ジョブサーバ属性操作関数
【形式】

```
#include "nqsv.h"
int NQSattrjsv( nqs_jsvid *jsvid,  nqs_alist ad,  int op,  nqs_res *res )
```

【機能説明】

ジョブサーバ ID が *jsvid* であるジョブサーバの属性を操作します。 属性リスト *ad* に含まれる属性について、*op* で指定された属性操作が実行されます。*op* に指定可能な値は以下のとおりです。

ATTROP_GET *ad* 内の属性へジョブサーバ属性をコピーします。

【戻り値】

正常に処理が完了すると、NQSattrjsv()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_ENOENT	指定された属性リストが存在しません。
NQS_ENOJSV	指定されたジョブサーバが存在しません。
NQS_EUNKNOWN	未知のジョブサーバ属性が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.6.3　ジョブサーバの制御

【名前】NQSctljsv -- ジョブサーバの制御
【形式】

```
#include "nqsv.h"
int NQSctljsv( nqs_jsvid *jsvid,  int cmd,  void *args,  nqs_res *res )
```

【機能説明】

jsvid で指定したジョブサーバに対して、*cmd* で示される制御コマンドを実行します。 *cmd* が引数を必要とする場合は、引数へのポインタを *args* に指定します。*cmd* に指定可能な制御コマンドとその引数 *args* の型、およびコマンドの機能説明は以下の通りです。

cmd: **JSVCTL_LINKDOWN**
args: **NULL** (不要)
指定されたジョブサーバとバッチサーバ間の TCP 接続を強制的に切断します。既に切断されている場合は何もせず正常終了します。

cmd: **JSVCTL_BOOTUP**

jsvid	args = char *host_name	動作
jsvid.jsvno = -1	ホスト名指定	host_nameで指定された実行ホスト上で、登録されているジョブサーバIDでジョブサーバを起動します。
JSV番号指定	ホスト名指定	host_nameで指定された実行ホスト上でjsvid をジョブサーバIDとするジョブサーバを起動します。
jsvid.jsvno = -1	NULL	バッチサーバに登録されている全ジョブサーバを起動します。

cmd: **JSVCTL_SHUTDOWN**

jsvid	args = int *force	動作
JSV番号指定	force = 0	jsvid をジョブサーバIDとするジョブサーバをシャットダウンします。 ただし、ジョブを持つジョブサーバのシャットダウンは拒否されます。
JSV番号指定	force != 0	jsvid をジョブサーバIDとするジョブサーバをジョブの有無に関係なくシャットダウンします。

cmd: **JSVCTL_SHUTJSV**

jsvid	args = char *host_name	動作
JSV番号指定	NULL	jsvid をジョブサーバIDとするジョブサーバをシャットダウンします。 ただし、ジョブを持つジョブサーバのシャットダウンは拒否されます。

jsvid.jsvno = -1	ホスト名指定	host_nameで指定された実行ホスト上のジョブサーバをシャットダウンします。ただし、ジョブを持つジョブサーバのシャットダウンは拒否されます。
jsvid.jsvno = -1	NULL	バッチサーバに登録されている全ジョブサーバをシャットダウンします。 ただし、ジョブを持つジョブサーバのシャットダウンは拒否されます。

cmd: JSVCTL_FSHUTJSV

jsvid	args = char *host_name	動作
JSV番号指定	NULL	jsvid をジョブサーバIDとするジョブサーバをジョブの有無に関係なくシャットダウンします。
jsvid.jsvno = -1	ホスト名指定	host_nameで指定された実行ホスト上のジョブサーバをジョブの有無に関係なくシャットダウンします。
jsvid.jsvno = -1	NULL	バッチサーバに登録されている全ジョブサーバをジョブの有無に関係なくシャットダウンします。

cmd: JSVCTL_BOOTNGRP

jsvid	args = char *node_group	動作
jsvid.jsvno = -1	ノードグループ名指定	node_groupで指定されたノードグループに属する実行ホスト上のジョブサーバを起動します。

cmd: JSVCTL_SHUTNGRP

jsvid	args = char *node_group	動作
jsvid.jsvno = -1	ノードグループ名指定	node_groupで指定されたノードグループに属する実行ホスト上のジョブサーバをシャットダウンします。 ただし、ジョブを持つジョブサーバのシャットダウンは拒否されます。

cmd: JSVCTL_FSHUTNGRP

jsvid	args = char *node_group	動作
jsvid.jsvno = -1	ノードグループ名指定	node_groupで指定されたノードグループに属する実行ホスト上のジョブサーバをジョブの有無に関係なくシャットダウンします。

【注意事項】

NQSctljsv()の実行には PRIV_MGR 以上のユーザ権限が必要です。
JSVCTL_BOOTUP を実行するためには、実行ホスト上にランチャーデーモンが常駐している必要があります。

【戻り値】

正常に処理が完了すると、NQSctljsv()は 0 を返します。エラーが発生した場合は負の整数が返され API リザルトコードが *res* に設定されます。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	APIを実行する権限がありません。
NQS_ERANGE	指定されたジョブサーバIDは指定可能な範囲を超えています。
NQS_ENOJSV	指定されたジョブサーバが存在しません(cmd が JSVCTL_BOOTUP以外の場合)。
NQS_EALREADY	指定されたジョブサーバは既に起動しています(cmd が JSVCTL_BOOTUPの場合)。
NQS_EREFUSE	指定されたジョブサーバはジョブを持っているためコマンドは拒否されました。 (cmdがJSVCTL_SHUTDOWNの場合)
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3)

6.7 実行ホスト関連関数

6.7.1 実行ホストエントリの操作

【名前】NQSopenhst, NQSreadhst, NQSrewindhst, NQSclosehst -- 実行ホストエントリの操作

【形式】

```
#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 )
```

【機能説明】

NQSopenhst()は、バッチサーバが認識している実行ホスト(バッチサーバにリンクしているジョブサーバが一つ以上存在している実行ホスト)の中から、*odesc* で指定されたオブジェクトに関連する実行ホストを選別します。 そして、そのホスト ID を要素に持つ一覧表(実行ホストエントリ)を作成して内部インデックスを 0 に初期化した後、それを識別するためのエントリ識別子を返却します。 エラーが発生した場合には負の整数を返し、API リザルトコードを *res* に設定します。
odesc で指定可能なオブジェクトは下表のとおりです。

odesc.obj_typeの値	参照objメンバ	対象
NQS_OBJ_BSV	なし	接続BSV内の全実行ホスト

NQSreadhst()は、内部インデックスが指すエントリを nqs_hid 型ポインタで返し、内部インデックスを一つインクリメントします。 エントリの最後に達した場合は NULL を返し、*res* 内のエラー番号に NQS_EALLOVER を設定します。 エラーが発生した場合には NULL を返し、API リザルトコードを *res* に設定します。

NQSrewindhst()は内部インデックスを 0 に再初期化します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

NQSclosehst()は、NQSopenhst()が作成した実行ホストエントリを削除します。処理に成功した場合は 0 を、エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【注意事項】

NQSclosehst()の実行後に、NQSreadhst()が返したホスト ID を指すポインタを参照した場合の動作は保証されません。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。

NQS_ENOENT	無効なエントリ識別子が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSattrhst(3)

6.7.2 実行ホスト属性操作関数

【名前】NQSattrhst -- 実行ホスト属性操作関数
【形式】

```
#include "nqsv.h"
int NQSattrhst( nqs_hid *hid,  nqs_alist ad,  int op,  nqs_res *res  )
```

【機能説明】

ホスト ID が *hid* である実行ホストの属性を操作します。 属性リスト *ad* に含まれる属性について、*op* で指定された属性操作が実行されます。*op* に指定可能な値は以下のとおりです。

ATTROP_GET　*ad* 内の属性へ実行ホスト属性をコピーします。

【戻り値】

正常に処理が完了すると、NQSattrhst()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_ENOENT	指定された属性リストが存在しません。
NQS_ENOHST	指定された実行ホストが存在しません。
NQS_EUNKNOWN	未知の属性種別が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.8　キュー関連関数

6.8.1　キューエントリの操作

【名前】NQSopenque, NQSreadque, NQSrewindque, NQScloseque -- キューエントリの操作

【形式】

```
#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 )
```

【機能説明】

NQSopenque()は、バッチサーバ上に存在しているキュー(バッチキュー、会話キューおよび転送キュー)の中から、*odesc* で指定されたオブジェクトに関連するキューを選別します。 そして、そのキューID を要素に持つ一覧表(キューエントリ)を作成して内部インデックスを 0 に初期化した後、それを識別するためのエントリ識別子を返却します。 エラーが発生した場合には負の整数を返し、API リザルトコードを *res* に設定します。

odesc で指定可能なオブジェクトは下表のとおりです。

odesc.obj_typeの値	参照objメンバ	対象
NQS_OBJ_BSV	なし	接続BSV内の全キュー
NQS_OBJ_SCH	odesc.obj.schid	指定のスケジューラにバインドしている全キュー

NQSreadque()は、内部インデックスが指すエントリを nqs_qid 型ポインタで返し、内部インデックスを一つインクリメントします。 エントリの最後に達した場合は NULL を返し、*res* 内のエラー番号に NQS_EALLOVER を設定します。 エラーが発生した場合には NULL を返し、API リザルトコードを *res* に設定します。

NQSrewindque()は内部インデックスを 0 に再初期化します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

NQScloseque()は、NQSopenque()が作成したキューエントリを削除します。 処理に成功した場合は 0 を、エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【注意事項】

NQScloseque()の実行後に、NQSreadque()が返したキューID を指すポインタを参照した場合の動作は保証されません。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。

NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_ENOENT	無効なエントリ識別子が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSattrque(3)

6.8.2 キュー属性操作関数

【名前】NQSattrque -- キュー属性操作関数
【形式】

```
#include "nqsv.h"
int NQSattrque( nqs_qid *qid,  nqs_alist ad,  int op,  nqs_res *res  )
```

【機能説明】

キューID が *qid* で指定したキューの属性または転送キューの属性を操作します。属性リスト *ad* に含まれる属性について、*op* で指定された属性操作が実行されます。 *op* に指定可能な値は以下のとおりです。

ATTROP_GET　*ad* 内の属性へキュー属性をコピーします。
ATTROP_SET　*ad* 内の属性でキュー属性を置き換えます。
ATTROP_ADD　*ad* 内の属性値をキュー属性に追加します。
ATTROP_DEL　*ad* 内の属性値をキュー属性から削除します。

ATTROP_ADD、ATTROP_DEL は属性値のチェインが可能な属性に対してのみ指定できます。

【戻り値】

正常に処理が完了すると、NQSattrque()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_ENOENT	指定された属性リストが存在しません。
	削除しようとした宛先キューは登録されていません。
NQS_ENOQUE	指定されたキューが存在しません。
NQS_ELOOP	宛先キューの指定がループしています。
NQS_EEXIST	追加しようとした宛先キューは既に登録されています。
NQS_EFLOOD	宛先キューの最大登録数を越えています。
NQS_EOPE	属性値チェインが許されていない属性に対してATTROP_ADDが実行されました。
	属性値チェインが許されていない属性に対してATTROP_DELが実行されました。
NQS_ENMAPDB	NMAPデータベースに登録されていないホストが指定されました
NQS_EUNKNOWN	未知の属性種別が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.8.3 キューの作成

【名前】NQScreque -- キューの作成
【形式】

```
#include "nqsv.h"
int NQScreque( nqs_qid *qid, nqs_alist ad, nqs_res *res)
```

【機能説明】

キューID *qid* で指定されたキュー種別とキュー名を持つキューを新規に作成します。作成されたキューには属性リスト *ad* に含まれる属性が設定されます。

【注意事項】

NQScreque()の実行には PRIV_MGR 以上のユーザ権限が必要です。

【戻り値】

正常に処理が完了すると、NQScreque()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOENT	指定された属性リストが存在しません。
NQS_EEXIST	指定されたキューは既に存在しています。
NQS_EUNKNOWN	未知の属性種別が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.8.4 キューの削除

【名前】NQSdelque -- キューの削除
【形式】

```
#include "nqsv.h"
int NQSdelque( nqs_qid *qid,  nqs_res *res )
```

【機能説明】

キューID *qid* で指定されたキュー種別とキュー名を持つキューを削除します。 削除の対象となるキューは投入禁止でかつリクエストが投入されていない状態でなければなりません。

【注意事項】

NQSdelque()の実行には PRIV_MGR 以上のユーザ権限が必要です。

【戻り値】

正常に処理が完了すると、NQSdelque()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOQUE	指定されたキューが存在しません。
NQS_EENABLE	キューが投入可能状態です。
NQS_EHASREQ	キューにリクエストが投入されています。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.8.5 スケジューラの接続

【名前】NQSbindsch -- キューとバッチスケジューラの接続
【形式】

```
#include "nqsv.h"
int NQSbindsch( nqs_qid *qid, nqs_schid *schid, nqs_res *res )
```

【機能説明】

NQSbindsch()は、スケジューラ ID が *schid* に一致するスケジューラを *qid* で指定したキューID を持つキューに接続(バインド)します。
qid で指定可能なキューはバッチキュー、会話キューのみです。 また、既にスケジューラに接続しているキューはバインドできません。

【注意事項】

NQSbindsch()の実行には PRIV_MGR 以上のユーザ権限が必要です。

【戻り値】

正常に処理が完了すると、NQSbindsch()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOQUE	指定されたキューが存在しません。
NQS_EWRNGTYP	指定されたキューが指定可能なキューではありません。
NQS_ENOSCH	指定されたスケジューラはバッチサーバにリンクしていません。
NQS_EBINDSCH	指定されたキューは既にスケジューラにバインドされています。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSunbindsch(3)

6.8.6　スケジューラの切り放し

【名前】NQSunbindsch -- キューとスケジューラの切り放し
【形式】

```
#include "nqsv.h"
int NQSunbindsch( nqs_qid *qid, nqs_schid *schid, nqs_res *res )
```

【機能説明】

NQSunbindsch()は、スケジューラ ID が *schid* に一致するスケジューラと *qid* で指定したキューID を持つキュー間の接続を切断します。
qid で指定可能なキューはバッチキュー、会話キューのみです。

【注意事項】

NQSunbindsch()の実行には PRIV_MGR 以上のユーザ権限が必要です。

【戻り値】

正常に処理が完了すると、NQSunbindsch()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOQUE	指定されたキューが存在しません。
NQS_EWRNGTYP	指定されたキューは指定可能なキューではありません。
NQS_ENOSCH	指定されたスケジューラはバッチサーバにリンクしていません。
NQS_EBINDSCH	指定されたキューはスケジューラにバインドしていません。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSbindsch(3)

6.8.7 ジョブサーバの接続

【名前】NQSbindjsv -- キューとジョブサーバの接続
【形式】

```
#include "nqsv.h"
int NQSbindjsv( nqs_qid *qid,  nqs_jsvid *jsvid,  nqs_res *res  )
```

【機能説明】

NQSbindjsv()は、ジョブサーバ ID が *jsvid* に一致するジョブサーバを *qid* で指定したキューID を持つキューに接続(バインド)します。
qid で指定可能なキューはバッチキュー、会話キューのみです。 また、既にキューに接続しているジョブサーバはバインドできません。

【注意事項】

NQSbindjsv()の実行には PRIV_MGR 以上のユーザ権限が必要です。

【戻り値】

正常に処理が完了すると、NQSbindjsv()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOQUE	指定されたキューが存在しません。
NQS_EWRNGTYP	指定されたキューは指定可能なキューではありません。
NQS_EBINDJSV	指定されたジョブサーバは既にキューにバインドされています。
NQS_EQUEDOM	指定されたジョブサーバは他のキューに投入されているジョブを持っています。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSunbindjsv(3)

6.8.8　ジョブサーバの切り放し

【名前】NQSunbindjsv -- キューとジョブサーバの切り放し
【形式】

```
#include "nqsv.h"
int NQSunbindjsv( nqs_qid *qid, nqs_jsvid *jsvid, nqs_res *res )
```

【機能説明】

NQSunbindjsv()は、ジョブサーバ ID が *jsvid* に一致するジョブサーバと *qid* で指定したキューID を持つキュー間の接続を切断します。
qid で指定可能なキューはバッチキュー、会話キューのみです。

【注意事項】

NQSunbindjsv()の実行には PRIV_MGR 以上のユーザ権限が必要です。

【戻り値】

正常に処理が完了すると、NQSunbindjsv()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOQUE	指定されたキューが存在しません。
NQS_EWRNGTYP	指定されたキューは指定可能なキューではありません。
NQS_ENOJSV	指定されたジョブサーバはバッチサーバにリンクしていません。
NQS_EBINDJSV	指定されたジョブサーバはキューにバインドしていません。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSbindjsv(3)

6.9 リクエスト関連関数

6.9.1 リクエストエントリの操作

【名前】NQSopenreq, NQSopenbreq, NQSopenireq, NQSreadreq, NQSrewindreq, NQSclosereq -- リクエストエントリの操作
NQSopenprm, NQSreadprm, NQSrewindprm, NQScloseprm
-- パラメトリックリクエストの操作

【形式】

```
#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 )
```

【機能説明】

NQSopenreq()は、バッチサーバ上に存在しているリクエストの中から、*odesc* で指定されたオブジェクトに関連するリクエストを選別します。 パラメトリックリクエストの場合、サブリクエストが選別されます。そして、そのリクエスト ID を要素に持つ一覧表(リクエストエントリ)を作成して内部インデックスを 0 に初期化した後、それを識別するためのエントリ識別子を返却します。 エラーが発生した場合には負の整数を返し、API リザルトコードを *res* に設定します。
odesc で指定可能なオブジェクトは下表のとおりです。

odesc.obj_typeの値	参照objメンバ	対象
NQS_OBJ_BSV	なし	接続BSV内の全リクエスト
NQS_OBJ_QUE	odesc.obj.qid	指定のキュー内に存在する全リクエスト
NQS_OBJ_PRM	odesc.obj.rid	パラメトリックリクエスト内の全サブリクエスト

NQSopenbreq()、NQSopenireq()により、それぞれバッチリクエスト、会話リクエストを区別してオープンすることができます。

NQSreadreq()は、内部インデックスが指すエントリを nqs_rid 型ポインタで返し、内部インデックスを一つインクリメントします。 エントリの最後に達した場合は NULL を返し、*res* 内のエラー番号に NQS_EALLOVER を設定します。エラーが発生した場合には NULL を返し、API リザルトコードを *res* に設定します。

NQSrewindreq()は内部インデックスを 0 に再初期化します。エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

NQSclosereq()は、NQSopenreq()が作成したリクエストエントリを削除します。処理に成功した場合は 0 を、エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

パラメトリックリクエスト（各サブリクエストではなく、全体を示す）のエントリの取得には、NQSopenprm()、NQSreadprm()、NQSrewindprm()、NQScloseprm()を使用します。

【注意事項】

NQSclosereq()の実行後に、NQSreadreq()が返したリクエスト ID を指すポインタを参照した場合の動作は保証されません。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_ENOENT	無効なエントリ識別子が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSattrreq(3)

6.9.2 リクエスト属性操作関数

【名前】NQSattrreq -- リクエスト属性操作関数
【形式】

```
#include "nqsv.h"
int NQSattrreq( nqs_rid *rid,  nqs_alist ad,  int op,  nqs_res *res  )
```

【機能説明】

リクエスト ID が rid であるリクエストの属性を操作します。 属性リスト ad に含まれる属性について、op で指定された属性操作が実行されます。 op に指定可能な値は以下のとおりです。

ATTROP_GET	*ad* 内の属性へリクエスト属性をコピーします。
ATTROP_SET	*ad* 内の属性でリクエスト属性を置き換えます。
ATTROP_ADD	*ad* 内の属性値をリクエスト属性に追加します。
ATTROP_DEL	*ad* 内の属性値をリクエスト属性から削除します。

ATTROP_ADD、ATTROP_DEL は属性値のチェインが可能な属性に対してのみ指定できます
また、*rid* で指定したリクエストがパラメトリックリクエストのサブリクエストの場合（rid.subreq_no>=0）、ATTROP_SET は指定できません。

【戻り値】

正常に処理が完了すると、NQSattrreq()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_ENOENT	指定された属性リストが存在しません。
NQS_ENOREQ	指定されたリクエストが存在しません。
NQS_EUNKNOWN	未知の属性種別が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.9.3　リクエストの作成とキューへの投入

【名前】NQScrereq, NQSleadreq -- リクエストの作成とキューへの投入
【形式】

```
#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)
```

【機能説明】

NQScrereq()は、*script* で指定されたファイルをジョブスクリプトとする新規リクエストを作成し、キューID*qid* で指定されたバッチキューへ投入します。 属性リスト識別子 *ad* が 0 以上の整数の場合、属性リスト内の全属性がリクエストの初期属性として設定されます。

NQSleadreq()は、リクエスト連携時に全ての連携対象リクエストが投入された事をバッチサーバに通知します。 *rid* にはリードリクエスト(連携対象の内、最初に投入されたリクエスト)のリクエスト ID を設定します。

リクエスト初期属性として設定可能な属性は下表のとおりです。

属性名	属性種別	型	スコープ		
			リクエスト	ジョブ	プロセス
ジョブ形態	ATTR_JTPLGY	int	○		
リラン属性	ATTR_RERUNABL	int	○		
チェックポイント属性	ATTR_CHKPNTABL	int	○		
マイグレーション属性	ATTR_MIGRATABL	int	○		
ホールド属性	ATTR_HOLDABL	int	○		
ホールドタイプ	ATTR_HOLDTYPE	int	○		
アカウントコード	ATTR_ACCTCODE	char *	○		
プライオリティ	ATTR_PRIORITY	nqs_range	○		
リクエスト名	ATTR_REQNAME	char *	○		
標準出力パス名	ATTR_STDOUT	nqs_pdesc	○		
標準エラー出力パス名	ATTR_STDERR	nqs_pdesc	○		
リクエストログ出力パス名	ATTR_STDLOG	nqs_pdesc	○		
リクエストログの出力レベル	ATTR_LOGLEVEL	nqs_range	○		
ステージングファイル情報	ATTR_STGFILE	nqs_stgfile	○		
シェル名	ATTR_SHELLPATH	char *	○		
メールオプション	ATTR_MAILOPTS	int	○		
メールアドレス	ATTR_MAILADDR	nqs_mdesc	○		
ジョブ実行環境条件	ATTR_JOBCOND	nqs_jcond	○		
リクエストグループ	ATTR_REQGRP	nqs_rgrp	○		
実行時刻	ATTR_EXETIME	time_t	○		
ジョブ環境変数	ATTR_ENVIRON	nqs_keyval			○

マイグレーションファイル	ATTR_MIGFILE	nqs_migfile	○		
ユーザ定義属性	ATTR_USERATTR	nqs_keyval	○		
リスタートファイル格納パス	ATTR_RSTFDIR	char *	○		
予約区間ID	ATTR_ADVRSVID	int	○		
経過時間制限値	ATTR_ELPSTIM	nqs_rlim	○		
CPU時間制限値	ATTR_CPUTIM	nqs_rlim		○	○
同時実行CPU台数制限値	ATTR_CPUNUM	nqs_rnum		○	
オープンファイル数制限値	ATTR_FILENUM	nqs_rnum			○
メモリサイズ制限値	ATTR_MEMSZ	nqs_rlim		○	○
データサイズ制限値	ATTR_DATASZ	nqs_rlim			○
スタックサイズ制限値	ATTR_STACKSZ	nqs_rlim			○
コアファイルサイズ制限値	ATTR_CORESZ	nqs_rlim			○
ファイルサイズ制限値	ATTR_FILESZ	nqs_rlim			○
仮想メモリサイズ制限値	ATTR_VMEMSZ	nqs_rlim		○	○
同時実行GPU台数制限値	ATTR_GPUNUM	nqs_rnum		○	
同時実行VEノード数制限値	ATTR_VENUM	nqs_rrange		○	

【注意事項】

連携対象となる全リクエスト(リードリクエストも含みます)には、リクエスト初期属性として ATTR_REQGRP 属性を設定する必要があります。 ただし、リードリクエストの ATTR_REQGRP 属性に関しては grpno(リクエストグループ番号)のみを適切な値に設定し、seqno および mid は常に-1 を設定してください(NQSleadreq(3)実行時にバッチサーバがリードリクエスト ID の seqno、mid に変更します)。

【戻り値】

正常に処理が完了すると、NQScrereq()は新たに作成したリクエストのリクエスト ID を *rid* にセットし返却値として 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_ESYSCAL	システムコール内でエラーが発生しました。
NQS_EWRNGSZ	ジョブスクリプトの転送に失敗しました。
NQS_ENOENT	*ad*で指定された属性リストが存在しません。
NQS_EACCTAUTH	リクエスト所有者のアカウントが登録されていません。
NQS_EACCESSDEN	バッチサーバへのアクセスが禁止されています。
NQS_EFATAL	シーケンス番号の採番に失敗しました。
NQS_ETOOLONG	結果ファイル名が長すぎます。
NQS_EUNKNOWN	未サポートの初期属性が*ad*内に含まれています。
NQS_EWRNGTYP	連携リクエストを投入可能なキュー以外に投入しようと

	しました。
NQS_EUNSUPPORT	未サポートのジョブ形態が指定されました。
NQS_EBADVAL	不正な値を持ったリクエスト属性が存在します。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.9.4　ジョブの生成とステージインの開始

【名前】NQSstgreq -- ジョブの生成とステージインの開始
【形式】

```
#include "nqsv.h"
int NQSstgreq( nqs_rid *rid,  int num,  nqs_jsvid jsvid[],  nqs_res *res )
```

【機能説明】

NQSstgreq()は、*rid* で指定したリクエストに関するジョブを生成した後、ステージイン処理を開始します。 この関数はキューに投入されている QUEUED 状態のリクエストに対してのみ有効です。
num には新たに生成するジョブの個数を、また、*jsvid* 配列にはジョブを実行するジョブサーバを指定します。 *jsvid* 配列の添字はジョブ番号に対応しており、例えば、jsvid[3].jsvno が 7 の場合、ジョブサーバ番号 7 のジョブサーバにはジョブ番号 3 のジョブが配置されます。*num* に指定可能なジョブ数は、1～(NQS_MAX_JOBNO + 1) です。ジョブサーバ番号として STGOPT_LEAVE が指定された *jsvid* 配列要素に対する処理は行われず、該当ジョブの状態は現状のまま保持されます。

指定したジョブ番号に対応するジョブが既に生成されている場合、既存のジョブは破棄され、新たに *jsvid* で指定されたジョブサーバ上に生成されます。

対象リクエストが投入されているキューは、NQSstgreq()を呼び出したプロセス(通常はスケジューラ)にバインドされている必要があります。

rid には、パラメトリックリクエスト（rid.subreq_no=-2）は指定できません。

【注意事項】

NQSstgreq()の実行には PRIV_SCH 以上のユーザ権限が必要です。
全てのジョブが生成された時点で、ジョブ番号 0 番のジョブ(マスタージョブ)が含まれていない、もしくはジョブ番号が連続していない場合、NQSstgreq()は NQS_EDISCONTI エラーを返します。

NQSstgreq()の呼び出しが成功すると、リクエストは STAGING 状態へ遷移しステージイン処理が開始されます。STAGING 状態中にエラーが発生した場合は全てのジョブは破棄され、リクエストは QUEUED 状態へ戻されます。

【戻り値】

正常に処理が完了すると、NQSstgreq()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。NQSstgreq()がエラーを返した場合、各ジョブの状態は本関数を呼び出す直前の状態に戻されます。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。

NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_EBINDSCH	呼び出しプロセスは、リクエストが投入されているキューをバインドしていません。
NQS_EOUTRNG	指定されたジョブサーバ番号は有効範囲外です。
	指定されたジョブ番号は有効範囲外です。
NQS_ENOREQ	指定されたリクエストが存在しません。
NQS_EWRNGSTS	要求を受け付けられないリクエスト状態です。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。
NQS_EDISCONTI	対象ジョブのジョブ番号が連続していません。

【関連項目】

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

6.9.5　リクエストの実行

【名前】NQSrunreq -- リクエストの実行
【形式】

```
#include "nqsv.h"
int NQSrunreq( nqs_rid *rid,  nqs_res *res )
```

【機能説明】

NQSrunreq()は *rid* で指定されたリクエストの実行開始(ジョブの起動)をバッチサーバへ指示します。 この関数はキューに投入されている QUEUED 状態のリクエストに対してのみ有効です。 また、対象リクエストは NQSstgreq(3)によって既にジョブが生成されていなければなりません。 ただし、ジョブサーバ上にリスタートファイルを持つリクエスト(HOLDING 状態を経由して QUEUED 状態へ遷移したリクエスト)は対象外です。
対象リクエストが投入されているキューは、NQSrunreq()を呼び出したプロセス(通常はスケジューラ)にバインドされており、 かつ、対象ジョブを管理する全てのジョブサーバは同キューにバインドされている必要があります。

rid には、パラメトリックリクエスト（rid.subreq_no=-2）は指定できません。

【注意事項】

NQSrunreq()の実行には PRIV_SCH 以上のユーザ権限が必要です。

【戻り値】

正常に処理が完了すると、NQSrunreq()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_EBINDSCH	呼び出しプロセスは、リクエストが投入されているキューをバインドしていません。
NQS_EBINDJSV	ジョブサーバが、リクエストが投入されているキューにバインドしていません。
NQS_ENOREQ	指定されたリクエストが存在しません。
NQS_ENOJOB	ジョブが生成されていません。
NQS_EWRNGSTS	要求を受け付けられないリクエスト状態です。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.9.6　リクエストの削除

【名前】NQSdelreq -- リクエストの削除
【形式】

```
#include "nqsv.h"
int NQSdelreq( nqs_rid *rid, 0 int grace,  nqs_res *res  )
```

【機能説明】

NQSdelreq()は、*rid* で指定されたリクエストを削除します。 実行ホスト上に実行中のジョブが存在する場合は、シグナルによってジョブを強制終了させます。ジョブへのシグナル送信ではまず SIGTERM を送り、*grace* で指定された秒数を待ってから SIGKILL を送ります。*grace* が 0 以下の場合は SIGKILL のみを直ちに送信します。

【注意事項】

NQSdelreq()の実行には PRIV_MGR 以上のユーザ権限を持っているか、対象リクエストのグループのグループ管理者権限 PRIV_GMGR を持っているか、あるいは対象リクエストの所有者でなければなりません。

【戻り値】

正常に処理が完了すると、NQSdelreq()は返却値として 0 を返します。 エラーが発生した場合は負の整数が返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOREQ	指定されたリクエストが存在しません。
NQS_EREFUSE	リクエストが一時的に削除不可な状態にあります。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSrunreq(3)

6.9.7　リクエストへのシグナル送信

【名前】NQSsigreq -- リクエストへのシグナル送信
【形式】

```
#include "nqsv.h"
int NQSsigreq( nqs_rid *rid,  char *sig,  nqs_res *res  )
```

【機能説明】

NQSsigreq()は、*rid* で指定されたリクエストを親に持つ全ジョブへ *sig* で指定されたシグナルを送信します。 対象リクエストが RUNNING、SUSPENDED 状態以外の場合、NQSsigreq()はエラーを返します。
sig は接頭辞 SIG で始まるシグナル名で、下表のうちの一つが指定できます。 ただし、実行ホストのオペレーティングシステムでサポートされていないシグナルであった場合は送信されず、単に無視されます。

RUNNING 状態のリクエストに対して SIGSTOP が指定された場合、リクエスト状態は SUSPENDING へ遷移しリクエストのサスペンド処理が開始されます。

SUSPENDED 状態のリクエストに対して SIGCONT が指定された場合、リクエスト状態は RESUMING へ遷移しリクエストのリジューム処理が開始されます。

シグナル名	実行ホストOS
	Linux
SIGHUP	○
SIGINT	○
SIGQUIT	○
SIGILL	○
SIGTRAP	○
SIGABRT(SIGIOT)	○
SIGBUS	○
SIGFPE	○
SIGKILL	○
SIGSEGV	○
SIGUSR1	○
SIGUSR2	○
SIGPIPE	○
SIGALRM	○
SIGTERM	○
SIGCHLD(SIGCLD)	○
SIGCONT	○
SIGSTOP	○
SIGTSTP	○
SIGTTIN	○
SIGTTOU	○
SIGURG	○
SIGXCPU	○
SIGXFSZ	○

SIGWINCH	○
SIGIO	○
SIGPOLL	○
SIGPWR	○
SIGVTALRM	○
SIGPROF	○
SIGSTKFLT	○
SIGUNUSED	○

【注意事項】

NQSsigreq()の実行には PRIV_MGR 以上のユーザ権限を持っているか、対象リクエストのグループのグループ管理者権限 PRIV_GMGR を持っているか、あるいは対象リクエストの所有者でなければなりません。

【戻り値】

正常に処理が完了すると、NQSsigreq()は返却値として 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOREQ	指定されたリクエストが存在しません。
NQS_EWRNGSTS	リクエスト状態がRUNNINGまたはSUSPENDEDではありません。
NQS_ESTALLED	リクエストはストール中です。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSsigjob(3)

6.9.8　リクエストのホールド

【名前】NQShldreq -- リクエストのホールド
【形式】

```
#include "nqsv.h"
int NQShldreq( nqs_rid *rid,  nqs_res *res )
```

【機能説明】

NQShldreq()は、*rid* で指定されたリクエストのホールド処理を開始します。 ホールド対象リクエストの状態は QUEUED、WAITING、あるいは RUNNING のうちの一つである必要があります。 リクエストがホールド禁止に設定されている場合はエラーになります。
RUNNING 状態のリクエストをホールドする場合には各ジョブに対して終了型チェックポイントが自動的に採取され、実行ホスト上にリスタートファイルを生成します。

ホールド処理が成功すると、ホールドを実行したクライアントの権限をリクエストのホールドタイプ属性(ATTR_HOLDTYPE)に格納します。

【注意事項】

NQShldreq()の実行には PRIV_OPE 以上のユーザ権限を持っているか、対象リクエストのグループのグループ管理者権限 PRIV_GMGR を持っているか、あるいは対象リクエストの所有者でなければなりません。

【戻り値】

正常に処理が完了すると、NQShldreq()は返却値として 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOREQ	指定されたリクエストが存在しません。
NQS_EWRNGSTS	リクエスト状態がQUEUED、WAITING、RUNNINGではありません。
NQS_EDISABLE	リクエストがホールド禁止に設定されています。
NQS_ESTALLED	リクエストはストール中です。
NQS_ESYSCAL	システムコール実行中にエラーが発生しました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.9.9 リクエストのリリース

【名前】NQSrlsreq -- リクエストのリリース
【形式】

```
#include "nqsv.h"
int NQSrlsreq( nqs_rid *rid,  nqs_res *res )
```

【機能説明】

NQSrlsreq()は、*rid* で指定されたリクエストのリリース処理を開始します。 リリース対象リクエストの状態は HELD である必要があります。

【注意事項】

NQSrlsreq()の実行には、リクエストに設定されているホールドタイプ属性(ホールド実行時の権限が格納されている)と同等かそれ以上のユーザ権限が必要です。

【戻り値】

正常に処理が完了すると、NQSrlsreq()は返却値として 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOREQ	指定されたリクエストが存在しません。
NQS_EWRNGSTS	リクエスト状態がHELDではありません。
NQS_ESYSCAL	システムコール実行中にエラーが発生しました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.9.10 リクエストのリスタート

【名前】NQSrstreq -- リクエストのリスタート
【形式】

```
#include "nqsv.h"
int NQSrstreq( nqs_rid *rid,  nqs_res *res )
```

【機能説明】

NQSrstreq()は、*rid* で指定されたリクエストのリスタート処理を開始します。リスタート対象リクエストの状態は QUEUED である必要があります。 また、リスタート対象となる全ジョブは実行ホスト上にリスタートファイルとして存在していなければなりません。
対象リクエストが投入されているキューは、NQSrstreq()を呼び出したプロセス(通常はスケジューラ)にバインドされており、かつ、リスタート対象ジョブを管理しているジョブサーバは同キューにバインドされている必要があります。

【注意事項】

NQSrstreq()の実行には PRIV_SCH 以上のユーザ権限が必要です。

【戻り値】

正常に処理が完了すると、NQSrstreq()は返却値として 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOREQ	指定されたリクエストが存在しません。
NQS_EWRNGSTS	リクエスト状態がQUEUEDではありません。
NQS_EBINDSCH	呼び出しプロセスは、リクエストが投入されているキューをバインドしていません。
NQS_EBINDJSV	リスタート対象ジョブを管理するジョブサーバが、キューにバインドしていません。
NQS_ENOJOB	指定されたリクエストを親に持つジョブが存在しません。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.9.11 リクエストのリラン

【名前】NQSrqureq -- リクエストの実行中止と再キューイング
【形式】

```
#include "nqsv.h"
int NQSrqureq( nqs_rid *rid,  int grace,  nqs_res *res)
```

【機能説明】

NQSrqureq()は、*rid* で指定されたリクエストの処理を中止し、ジョブを持たない QUEUED 状態へ戻します（再キューイング)。 実行ホスト上で実行されているジョブにはまず SIGTERM を送り、*grace* で指定された秒数を待った後 SIGKILL を送ります。*grace* が 0 以下の場合は SIGKILL のみを直ちに送信します。
NQSrqureq()はジョブを持つリクエストであればリクエスト状態に関係なく実行可能です。 ジョブを持たないリクエスト、または、リラン禁止に設定されているリクエストに対して実行した場合はエラーになります。

【注意事項】

NQSrqureq()の実行には PRIV_OPE 以上のユーザ権限を持っているか、対象リクエストのグループのグループ管理者権限 PRIV_GMGR を持っているか、あるいは対象リクエストの所有者でなければなりません。

【戻り値】

正常に処理が完了すると、NQSrqureq()は返却値として 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOREQ	指定されたリクエストが存在しません。
NQS_EWRNGSTS	リクエスト状態がRUNNINGではありません。
NQS_EDISABLE	リクエストがリラン禁止に設定されています。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3), NQSdelreq(3)

6.9.12 リクエストのキュー間移動

【名前】NQSmovreq -- リクエストのバッチキュー間移動
【形式】

```
#include "nqsv.h"
int NQSmovreq( nqs_rid *rid,  nqs_qid *qid,  nqs_res *res )
```

【機能説明】

NQSmovreq()は、*rid* で指定されたリクエストを *qid* で指定されたキュー(バッチキュー、会話キューまたは転送キュー)へ移動させます。 ジョブを持たない(実行ホスト上にジョブのリスタートファイルが存在しない)QUEUED、WAITING、および HELD 状態のリクエストのみが移動可能です。 移動後のリクエスト状態は移動前の状態と同じになります。

rid には、パラメトリックリクエスト（rid.subreq_no=-2）は指定できません。

【注意事項】

NQSmovreq()の実行には PRIV_OPE 以上のユーザ権限を持っているか、対象リクエストのグループのグループ管理者権限 PRIV_GMGR を持っているか、あるいは対象リクエストの所有者でなければなりません。

【戻り値】

正常に処理が完了すると、NQSmovreq()は返却値として 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOREQ	指定されたリクエストが存在しません。
NQS_ENOQUE	指定されたキューが存在しません。
NQS_EWRNGSTS	リクエスト状態がQUEUED、WAITING、またはHELDではありません。
NQS_EWRNGTYP	指定されたキューが指定可能なキューではありません。
NQS_EHASJOB	指定されたリクエストにはジョブが存在しています。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3)

6.10 ジョブ関連関数

6.10.1 ジョブエントリの操作

【名前】NQSopenjob, NQSreadjob, NQSrewindjob, NQSclosejob -- ジョブエントリの操作
【形式】

```
#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 )
```

【機能説明】

NQSopenjob()は、実行ホストに存在しているジョブの中から、*odesc* で指定されたオブジェクトに関連するジョブを選別します。 そして、そのジョブ ID を要素に持つ一覧表(ジョブエントリ)を作成して内部インデックスを 0 に初期化した後、それを識別するためのエントリ識別子を返却します。 エラーが発生した場合には負の整数を返し、API リザルトコードを *res* に設定します。
odesc で指定可能なオブジェクトは下表のとおりです。

odesc.obj_typeの値	参照objメンバ	対象
NQS_OBJ_BSV	なし	接続BSV内の全ジョブ
NQS_OBJ_JSV	odesc.obj.jsvid	指定のジョブサーバによって管理されている全ジョブ
NQS_OBJ_REQ	odesc.obj.rid	指定のリクエストを親に持つ全ジョブ

NQSreadjob()は、内部インデックスが指すエントリを nqs_jid 型ポインタで返し、内部インデックスを一つインクリメントします。 エントリの最後に達した場合は NULL を返し、*res* 内のエラー番号に NQS_EALLOVER を設定します。 エラーが発生した場合には NULL を返し、API リザルトコードを *res* に設定します。

NQSrewindjob()は内部インデックスを 0 に再初期化します。エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

NQSclosejob()は、NQSopenjob()が作成したジョブエントリを削除します。 処理に成功した場合は 0 を、エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【注意事項】

NQSclosejob()の実行後に、NQSreadjob()が返したジョブ ID を指すポインタを参照した場合の動作は保証されません。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。

NQS_ENOENT	無効なエントリ識別子が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSattrjob(3)

6.10.2 ジョブ属性操作関数

【名前】NQSattrjob -- ジョブ属性操作関数
【形式】

```
#include "nqsv.h"
int NQSattrjob( nqs_jid *jid,  nqs_alist ad,  int op,  nqs_res *res )
```

【機能説明】

ジョブ ID が *id* であるジョブの属性を操作します。 属性リスト *ad* に含まれる属性について、*op* で指定された属性操作が実行されます。 *op* に指定可能な値は以下のとおりです。

ATTROP_GET　　*ad* 内の属性へジョブ属性をコピーします。
ATTROP_SET　　*ad* 内の属性でジョブ属性を置き換えます。

【戻り値】

正常に処理が完了すると、NQSattrjob()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_ENOENT	指定された属性リストが存在しません。
NQS_ENOREQ	指定されたジョブの親リクエストが存在しません。
NQS_ENOJOB	指定されたジョブが存在しません。
NQS_EUNKNOWN	未知の属性種別が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.10.3 ジョブへのシグナル送信

【名前】NQSsigjob -- ジョブへのシグナル送信
【形式】

```
#include "nqsv.h"
int NQSsigjob( nqs_jid *jid,  char *sig,  nqs_res *res )
```

【機能説明】

NQSsigjob()は、*jid* で指定されたジョブへ *sig* で指定されたシグナルを送信します。 対象ジョブの親リクエストが RUNNING、SUSPENDED 状態以外の場合、NQSsigjob()はエラーを返します。
sig は接頭辞 SIG で始まるシグナル名で、NQSsigreq()の機能説明にあるシグナル一覧表のうちの一つを指定できます。 ただし、実行ホストのオペレーティングシステムでサポートされていないシグナルであった場合は送信されず、単に無視されます。

【注意事項】

NQSsigjob()の実行には PRIV_MGR 以上のユーザ権限を持っているか、対象ジョブの親リクエストのグループのグループ管理者権限 PRIV_GMGR を持っているか、あるいは対象ジョブの親リクエストの所有者でなければなりません。

【戻り値】

正常に処理が完了すると、NQSsigjob()は返却値として 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOREQ	指定されたジョブの親リクエストが存在しません。
NQS_ENOJOB	指定されたジョブが存在しません。
NQS_EWRNGSTS	ジョブの親リクエストの状態が RUNNING または SUSPENDEDではありません。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSsigreq(3)

6.10.4 ジョブのマイグレーション

【名前】NQSmigjob -- ジョブのマイグレーション
【形式】

```
#include "nqsv.h"
int NQSmigjob( nqs_jid *jid,  nqs_jsvid *jsvid,  nqs_res *res  )
```

【機能説明】

NQSmigjob()は、*jid* で指定されたジョブを *jsvid* で指定されたジョブサーバの管理下へ移動(マイグレーション)します。 移動元のジョブサーバが停止している場合でもジョブの移動は可能ですが、移動先が停止している場合はエラーになります。
異なる実行ホスト間を跨いでマイグレーションを行う場合は、両ホストのジョブサーバデータベース(/var/opt/nec/nqsv/jsv)が NFS 等の共有ファイルシステムによって共有されている必要があります。

【注意事項】

NQSmigjob()の実行には PRIV_OPE 以上のユーザ権限が必要です。

【戻り値】

正常に処理が完了すると、NQSmigjob()は返却値として 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	マイグレーションを実行する権限がありません。
	移動先ホスト上でジョブを実行する権限がありません。
NQS_ENOREQ	指定されたジョブの親リクエストが存在しません。
NQS_ENOJOB	指定されたジョブが存在しません。
NQS_ENOJSV	指定されたジョブサーバが存在しません。
NQS_EWRNGSTS	ジョブの親リクエストの状態がHELDではありません。
NQS_EQUEDOM	現在のキューとは異なるキューへ移動しようとしました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQShldreq(3)

6.11 ノードグループ関連関数

6.11.1 ノードグループエントリの操作

【名前】NQSopenngrp, NQSreadngrp, NQSrewindngrp, NQSclosengrp -- ノードグループエントリの操作

【形式】

```
#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 )
```

【機能説明】

NQSopenngrp()は、バッチサーバに存在するノードグループの中から、*odesc* で指定されたオブジェクトに関連するノードグループを選別します。 そして、そのノードグループ ID を要素に持つ一覧表(ノードグループエントリ)を作成して内部インデックスを 0 に初期化した後、それを識別するためのエントリ識別子を返却します。 エラーが発生した場合には負の整数を返し、API リザルトコードを *res* に設定します。

odesc で指定可能なオブジェクトは下表のとおりです。

odesc.obj_typeの値	参照objメンバ	対象
NQS_OBJ_BSV	なし	接続BSV内の全ノードグループ
NQS_OBJ_JSV	odesc.obj.jsvid	指定のジョブサーバを含む全ノードグループ
NQS_OBJ_QUE	odesc.obj.qid	指定のキューにバインドされている全ノードグループ

NQSreadngrp()は、内部インデックスが指すエントリを nqs_ngrpid 型ポインタで返し、内部インデックスを一つインクリメントします。 エントリの最後に達した場合は NULL を返し、*res* 内のエラー番号に NQS_EALLOVER を設定します。エラーが発生した場合には NULL を返し、API リザルトコードを *res* に設定します。

NQSrewindngrp()は内部インデックスを 0 に再初期化します。エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

NQSclosengrp()は、NQSopenngrp()が作成したノードグループエントリを削除します。 処理に成功した場合は 0 を、エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【注意事項】

NQSclosengrp()の実行後に、NQSreadngrp()が返したノードグループ ID を指すポインタを参照した場合の動作は保証されません。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。

NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_ENOENT	無効なエントリ識別子が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSattrngrp(3)

6.11.2 ノードグループ属性操作関数

【名前】NQSattrngrp -- ノードグループ属性操作関数
【形式】

```
#include "nqsv.h"
int NQSattrngrp( nqs_ngrpid *ngrpid, nqs_alist ad, int op, nqs_res
*res )
```

【機能説明】

ノードグループ ID が *ngrpid* であるノードグループの属性を操作します。 属性リスト *ad* に含まれる属性について、*op* で指定された属性操作が実行されます。 *op* に指定可能な値は以下のとおりです。

ATTROP_GET　　*ad* 内の属性へノードグループ属性をコピーします。
ATTROP_SET　　*ad* 内の属性でノードグループ属性を置き換えます。

【戻り値】

正常に処理が完了すると、NQSattrngrp()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_ENOENT	指定された属性リストが存在しません。
NQS_ENONGRP	指定されたノードグループが存在しません。
NQS_EUNKNOWN	未知の属性種別が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.11.3 ノードグループの作成・削除

【名前】NQScrengrp, NQSdelngrp -- ノードグループの作成・削除
【形式】

```
#include "nqsv.h"
int NQScrengrp( nqs_ngrpid *ngrpid,  nqs_alist ad,  nqs_res *res )
int NQSdelngrp( nqs_ngrpid *ngrpid,  nqs_res *res )
```

【機能説明】

NQScrengrp()は、ノードグループの ID が *ngrpid*（ノードグループ名＋ノードグループタイプ）のノードグループを新規に作成します。*ad* にコメント属性（ATTR_COMMENT）を指定することで、同時にコメントの設定も行うことができます。

NQSdelngrp()は、ノードグループ ID が *ngrpid*(ノードグループ名＋ノードグループタイプ)のノードグループを削除します。

【注意事項】

NQScrengrp(),NQSdelngrp() の実行には PRIV_MGR 以上のユーザ権限を持っていなければなりません。

【戻り値】

正常に処理が完了すると、NQScrengrp(), NQSdelngrp()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_ENOENT	指定された属性リストが存在しません。
NQS_EEXIST	指定されたノードグループが既に存在します。
NQS_EUNKNOWN	未知の属性種別が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3)

6.11.4 ノードグループへのジョブサーバ追加

【名前】NQSIncludejsv, NQSIncludejsvrange, NQSIncludengrp -- ノードグループへのジョブサーバの追加

【形式】

```
#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 )
```

【機能説明】

NQSIncludejsv(), NQSIncludejsvrange(), NQSIncludengrp()は、*ngrpid* に指定されたノードグループに対して JSV の追加を行います。
NQSIncludejsv()では追加するジョブサーバのジョブサーバ番号のリストを *jsvid* にて配列で渡します。配列の終端のジョブサーバ番号には、-1 を設定する必要があります。

NQSIncludejsvrange()では、追加するジョブサーバのジョブサーバ番号の範囲を jsvid 配列で渡します。ジョブサーバ番号の範囲の開始番号と終了番号をそれぞれ 0 番目、1 番目の要素として渡します。

NQSIncludengrp()では、*source* で指定したノードグループに属する全ジョブサーバを追加します。

NQSIncludejsv(), NQSIncludejsvrange(), NQSIncludengrp()によって追加するジョブサーバは、バッチサーバに登録されている必要があります。

【注意事項】

NQSIncludejsv(), NQSIncludejsvrange(), NQSIncludengrp()の実行には PRIV_MGR 以上のユーザ権限を持っていなければなりません。

【戻り値】

正常に処理が完了すると、NQSIncludejsv(), NQSIncludejsvrange(), NQSIncludengrp()は 0 を返します。指定した JSV の追加でエラーが発生した場合は、指定した JSV の追加をその時点で打ち切り、負の整数を返すとともに、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_ENONGRP	指定されたノードグループが存在しません。
NQS_ENOJSV	指定されたジョブサーバはバッチサーバに登録されていません。

NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3)

6.11.5 ノードグループからのジョブサーバ削除

【名前】NQSexcludejsv, NQSexcludejsvrange, NQSexcludengrp -- ノードグループからのジョブサーバの削除

【形式】

```
#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 )
```

【機能説明】

NQSexcludejsv(), NQSexcludejsvrange(), NQSexcludengrp()は、*ngrpid* に指定されたノードグループから JSV の削除を行います。
NQSexcludejsv()では削除するジョブサーバのジョブサーバ番号のリストを *jsvid* にて配列で渡します。配列の終端のジョブサーバ番号には、-1 を設定する必要があります。

NQSexcludejsvrange()では、削除するジョブサーバのジョブサーバ番号の範囲を jsvid 配列で渡します。ジョブサーバ番号の範囲の開始番号と終了番号をそれぞれ 0 番目、1 番目の要素として渡します。

NQSexcludengrp()では、*source* で指定したノードグループに属する全ジョブサーバを削除します。

NQSexcludejsv(), NQSexcludejsvrange(), NQSexcludengrp()によって指定したジョブサーバが、*ngrpid* で指定したノードグループに属していない場合、処理がスキップされるのみで、エラーとはなりません。

【注意事項】

NQSexcludejsv(), NQSexcludejsvrange(), NQSexcludengrp()の実行には PRIV_MGR 以上のユーザ権限を持っていなければなりません。

【戻り値】

正常に処理が完了すると、NQSexcludejsv(), NQSexcludejsvrange(), NQSexcludengrp()は 0 を返します。指定した JSV の追加でエラーが発生した場合は、指定した JSV の追加をその時点で打ち切り、負の整数を返すとともに、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_ENONGRP	指定されたノードグループが存在しません。

NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3)

6.11.6 ノードグループのキューへのバインド・アンバインド

【名前】NQSbindngrp, NQSunbindngrp -- ノードグループのキューへのバインド・アンバインド

【形式】

```
#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)
```

【機能説明】

NQSbindngrp()は、*ngrpid* に指定されたノードグループに属する全ジョブサーバに対して、*qid* に指定したキューへのバインドを行います。

NQSunbindngrp()は、*ngrpid* に指定されたノードグループに属する全ジョブサーバに対して、*qid* に指定したキューからのアンバインドを行います。

qid で指定可能なキューはバッチキュー、会話キューのみです。

【注意事項】

NQSbindngrp ()､NQSunbindngrp()の実行には PRIV_MGR 以上のユーザ権限を持っていなければなりません。

【戻り値】

正常に処理が完了すると、NQSbindngrp ()､NQSunbindngrp()は 0 を返します。エラーが発生した場合は、負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_ENONGRP	指定されたノードグループが存在しません。
NQS_ENOQUE	指定されたキューが存在しません。
NQS_EWRNGTYP	指定されたキューが指定可能なキューではありません。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3)

6.12 テンプレート関連関数

6.12.1 テンプレートの作成・削除

【名前】NQScretemp, NQSdeltemp -- テンプレートの作成・削除
【形式】

```
#include "nqsv.h"
int NQScretemp(nqs_template *template, nqs_res *res)
int NQSdeltemp(char *template_name, nqs_res *res)
```

【機能説明】

NQScretemp()は指定された *template* に従って、新たなテンプレートを作成します。
作成するテンプレートの種類（*template*->type）と定義情報（*template* 内の構造体）は以下で指定します。

テンプレートの種類	*template*->type	参照する*template*内の構造体
OpenStack用	NQSII_TEMPLATE_TYPE_OPENSTACK	struct nqs_ostemplate os_tmpl
コンテナ用	NQSII_TEMPLATE_TYPE_CONTAINER	struct nqs_cotemplate co_tmpl

NQSdeltemp()はテンプレート名が *template_name* のテンプレートを削除します。

【注意事項】

NQScretemp(), NQSdeltemp()の実行には PRIV_MGR 以上のユーザ権限を持っていなければなりません。

【戻り値】

正常に処理が完了すると、NQScretemp(),NQSdeltemp()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOENT	指定されたテンプレートが存在しません。
NQS_EEXIST	指定されたテンプレートが既に存在します。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3)

6.12.2 テンプレートの編集

【名前】NQSedittemp -- テンプレートの編集
【形式】

```
#include "nqsv.h"
int NQSedittemp(nqs_template *template, int changeflg, nqs_res *res)
```

【機能説明】

NQSedittemp ()は *template* で指定されたテンプレート情報を上書きします。上書きする対象のデータは *changeflg* のフラグで示します。
changeflg に指定可能なフラグは以下です。OR をとることで複数指定も可能です。

- TMPL_MEMBER_IMGNAME　イメージ名
- TMPL_MEMBER_CPU　CPU 数
- TMPL_MEMBER_MEM　メモリサイズ
- TMPL_MEMBER_GPU　GPU 数
- TMPL_MEMBER_CUSTOM　独自定義
- TMPL_MEMBER_COMMENT　コメント
- TMPL_MEMBER_FLAVOR　フレーバ名
  （*template*->type が NQSII_TEMPLATE_TYPE_OPENSTACK の時のみ）
- TMPL_MEMBER_STARTTIMEOUT　起動見込時間
- TMPL_MEMBER_STOPTIMEOUT　停止見込時間

【注意事項】

NQSedittemp()の実行には PRIV_MGR 以上のユーザ権限を持っていなければなりません。

【戻り値】

正常に処理が完了すると、NQSedittemp()は 0 を返します。エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOENT	指定されたテンプレートが存在しません。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3)

6.12.3 テンプレートのロック・アンロック

【名前】NQSlocktemp, NQSunlocktemp --テンプレートのロック・アンロック
【形式】

```
#include "nqsv.h"
int NQSlocktemp(char *template_name, nqs_res *res)
int NQSunlocktemp(char *template_name, nqs_res *res)
```

【機能説明】

NQSlocktemp()はテンプレート名が *template_name* のテンプレートをロックします。

NQSunlocktemp()はテンプレート名が *template_nam*e のテンプレートをロック解除します。

【注意事項】

NQSlocktemp(), NQSunlocktemp()の実行には PRIV_MGR 以上のユーザ権限を持っていなければなりません。

【戻り値】

正常に処理が完了すると、NQSlocktemp(), NQSunlocktemp()は 0 を返します。エラーが発生した場合は、負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOENT	指定されたテンプレートが存在しません。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3)

6.13 OpenStackテンプレート関連関数

6.13.1 OpenStack テンプレートの作成・削除

【名前】NQScreostemp, NQSdelostemp -- OpenStack テンプレートの作成・削除
【形式】

```
#include "nqsv.h"
int NQScreostemp(nqs_ostemplate *template, nqs_res *res)
int NQSdelostemp(char *template_name, nqs_res *res)
```

【機能説明】

NQScreostemp()は指定された *template* に従って、新たなテンプレートを作成します。

NQSdelostemp()はテンプレート名が *template_name* のテンプレートを削除します。

【注意事項】

NQScreostemp(), NQSdelostemp()の実行には PRIV_MGR 以上のユーザ権限を持っていなければなりません。

【戻り値】

正常に処理が完了すると、NQScreostemp(),NQSdelostemp()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOENT	指定されたテンプレートが存在しません。
NQS_EEXIST	指定されたテンプレートが既に存在します。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3)

6.13.2 OpenStack テンプレートの編集

【名前】NQSeditostemp -- OpenStack テンプレートの編集
【形式】

```
#include "nqsv.h"
int NQSeditostemp(nqs_ostemplate *template, int changeflg, nqs_res *res)
```

【機能説明】

NQSeditostemp()は指定された *template*->template_name のテンプレートを、*template* 内の情報で上書きします。上書きする対象のデータは *changeflg* のフラグで示します。
changeflg に指定可能なフラグは以下です。OR をとることで複数指定も可能です。

- OSTMPL_MEMBER_IMGNAME　　OS イメージ名
- OSTMPL_MEMBER_CPU　　CPU 数
- OSTMPL_MEMBER_MEM　　メモリサイズ
- OSTMPL_MEMBER_GPU　　GPU 数
- OSTMPL_MEMBER_CUSTOM　　独自定義
- OSTMPL_MEMBER_COMMENT　　コメント
- OSTMPL_MEMBER_FLAVOR　　フレーバ名
- OSTMPL_MEMBER_STARTTIMEOUT　起動見込時間
- OSTMPL_MEMBER_STOPTIMEOUT　停止見込時間

【注意事項】

NQSeditostemp()の実行には PRIV_MGR 以上のユーザ権限を持っていなければなりません。

【戻り値】

正常に処理が完了すると、NQSeditostemp()は 0 を返します。エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOENT	指定されたテンプレートが存在しません。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3)

6.13.3 OpenStack テンプレートのロック・アンロック

【名前】NQSlockostemp, NQSunlockostemp -- OpenStack テンプレートのロック・アンロック

【形式】

```
#include "nqsv.h"
int NQSlockostemp(char *template_name, nqs_res *res)
int NQSunlockostemp(char *template_name, nqs_res *res)
```

【機能説明】

NQSlockostemp()はテンプレート名が *template_name* のテンプレートをロックします。

NQSunlockostemp()はテンプレート名が *template_nam*e のテンプレートをロック解除します。

【注意事項】

NQSlockostemp(), NQSunlockostemp()の実行には PRIV_MGR 以上のユーザ権限を持っていなければなりません。

【戻り値】

正常に処理が完了すると、NQSlockostemp(), NQSunlockostemp()は 0 を返します。エラーが発生した場合は、負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_ENOENT	指定されたテンプレートが存在しません。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3)

6.14 ベアメタルサーバ関連関数

6.14.1 ベアメタルサーバの登録・削除

【名前】NQSattachhst_bm, NQSdetachhst_bm -- バッチサーバへのベアメタルサーバの登録、削除

【形式】

```
#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)
```

【機能説明】

NQSattachhst_bm()は、*hid* で指定されたベアメタルサーバを *jsvid* に指定したジョブサーバ番号、並びに、*cpu*（CPU 数）, *memsz*（メモリ量）, *memunit*（メモリ量の単位）, *gpu*（GPU 数）で指定したリソース情報とあわせて、実行ホストとしてバッチサーバに登録します。
ベアメタルサーバの登録は、ジョブサーバが立ち上がっていない状態でのみ実行が可能です。既に登録されている実行ホストに、指定された *hid* または *jsvid* が一致するものがある場合は、エラーとなります。
memunit（メモリ量の単位）には以下のいずれかの値を指定できます。

指定可能な値	単位
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()は、*hid* で指定されたベアメタルサーバまたは *jsvid* に指定したジョブサーバ番号に対応するベアメタルサーバを、バッチサーバの登録実行ホストから削除します。*hid*、*jsvid* のいずれかを指定し、指定しない方には NULL を設定してください。
ベアメタルサーバの削除は、ジョブサーバが立ち上がっていない状態でのみ実行が可能です。指定された *hid* または *jsvid* に一致するものが存在しない場合は、エラーとなります。

【注意事項】

NQSattachhst_bm(),NQSdetachhst_bm()の実行には PRIV_MGR 以上のユーザ権限が必要です。

【戻り値】

正常に処理が完了すると、NQSattachhst_bm(), NQSdetachhst_bm()は 0 を返します。 エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	API関数の実行に必要な権限がありません。
NQS_EALREADY	指定された実行ホスト、またはジョブサーバ番号が既に登録されています。

【関連項目】

NQSconnect (3)

6.15 カスタムリソース関連関数

6.15.1 カスタムリソースエントリの操作

【名前】NQSopencustom, NQSreadcustom, NQSrewindcustom, NQSclosecustom -- カスタムリソースエントリの操作

【形式】

```
#include "nqsv.h"
nqs_entry NQSopencustom( nqs_odesc *odesc, nqs_res *res )
nqs_crid *NQSreadcustom( nqs_entry entry, nqs_res *res )
int NQSrewindcustom( nqs_entry entry, nqs_res *res )
int NQSclosecustom( nqs_entry entry, nqs_res *res )
```

【機能説明】

NQSopencustom()は、バッチサーバに存在するカスタムリソース情報の中から、*odesc* で指定されたオブジェクトに関連するカスタムリソース情報を選別します。そして、そのカスタムリソース ID を要素に持つ一覧表(カスタムリソースエントリ)を作成して内部インデックスを 0 に初期化した後、それを識別するためのエントリ識別子を返却します。
エラーが発生した場合には負の整数を返し、API リザルトコードを *res* に設定します。
odesc で指定可能なオブジェクトは下表のとおりです。

odesc.obj_typeの値	参照objメンバ	対象
NQS_OBJ_BSV	なし	接続BSV内の全カスタムリソース

NQSreadcustom()は、内部インデックスが指すエントリを nqs_crid 型ポインタで返し、内部インデックスを一つインクリメントします。 エントリの最後に達した場合は NULL を返し、*res* 内のエラー番号に NQS_EALLOVER を設定します。
エラーが発生した場合には NULL を返し、API リザルトコードを *res* に設定します。

NQSrewindcustom()は内部インデックスを 0 に再初期化します。エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

NQSclosecustom()は、NQSopencustom()が作成したカスタムリソースエントリを削除します。 処理に成功した場合は 0 を、エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【注意事項】

NQSclosecustom()の実行後に、NQSreadcustom()が返したカスタムリソース ID を指すポインタを参照した場合の動作は保証されません。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。

NQS_ENOENT	無効なエントリ識別子が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3), NQSattrcustom(3)

6.15.2 カスタムリソース属性操作関数

【名前】NQSattrcustom -- カスタムリソース属性操作関数
【形式】

```
#include "nqsv.h"
int NQSattrcustom( char *cr_name, nqs_alist ad, int op, nqs_res *res )
```

【機能説明】

カスタムリソース名が *cr_name* であるカスタムリソース情報の属性を操作します。属性リスト *ad* に含まれる属性について、*op* で指定された属性操作が実行されます。

op に指定可能な値は以下のとおりです。

ATTROP_GET　*ad* 内の属性へカスタムリソース属性をコピーします。
ATTROP_SET　*ad* 内の属性でカスタムリソース属性を置き換えます。
ATTROP_ ADD　　*ad* 内の属性値をカスタムリソース属性に追加します。
ATTROP_ DEL　*ad* 内の属性値をカスタムリソース属性から削除します。

【戻り値】

正常に処理が完了すると、NQSattrcustom ()は 0 を返します。エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_ENOENT	指定された属性リストが存在しません。
NQS_ENOCR	指定されたカスタムリソースが存在しません。
NQS_EUNKNOWN	未知の属性種別が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

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

6.15.3 カスタムリソースの作成・削除

【名前】NQScrecustom, NQSdelcustom -- カスタムリソースの登録、削除
【形式】

```
#include "nqsv.h"
int NQScrecustom( nqs_crid *crid, int consumer, nqs_alist ad, nqs_res *res )
int NQSdelcustom( nqs_crid *crid, nqs_res *res )
```

【機能説明】

NQScrecustom()は、カスタムリソースの ID が *crid*（カスタムリソース名）のカスタムリソース情報を新規に作成します。*consumer* には作成するカスタムリソースの消費単位を指定し、以下の何れかの値を指定します。

- CR_JOB：job
- CR_REQ：request

ad にはカスタムリソースの利用量制御情報を指定することができます。属性リスト *ad* に含まれる属性について、本情報を指定しなくてもカスタムリソースを作成することができ、その場合は *ad* には-1 を指定します。
カスタムリソース名 *cr_name* であるカスタムリソース情報の属性を設定します。属性リスト *ad* に含まれる属性について、カスタムリソース作成後に設定されます。
NQSdelcustom()は、カスタムリソース ID が *crid*（カスタムリソース名）のカスタムリソース情報を削除します。

【注意事項】

NQScrecustom(), NQSdelcustom() の実行には PRIV_MGR 以上のユーザ権限を持っていなければなりません。

【戻り値】

正常に処理が完了すると、NQScrecustom(), NQSdelcustom() は 0 を返します。
エラーが発生した場合は負の整数を返し、API リザルトコードを *res* に設定します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_EPERM	属性にアクセスする権限がありません。
NQS_EEXIST	指定されたカスタムリソースが既に存在します。
NQS_EUNKNOWN	未知の属性種別が指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect(3)

6.16 ユーティリティ関数

6.16.1 API バージョンの取得

【名前】NQSapiver -- API バージョンの取得
【形式】

```
#include "nqsv.h"
char *NQSapiver( void )
```

【機能説明】

NQSapiver()は NQSV/API のバージョン文字列を以下の形式で取り出します。

xx.yy(zzzz)　xx　メジャー・バージョン番号（10 進 2 桁）
yy マイナー・バージョン番号（10 進 2 桁）
zzzz　プラットフォーム名(OS 識別名)

NQSapiver()は API リンクの有無に関係なく呼び出すことが可能です。

【戻り値】

NQSapiver()は必ず成功し、バージョン文字列へのポインタを返します。

6.16.2 マシン ID ホスト名変換

【名前】NQSmid2hname, NQShname2mid -- マシン ID ― ホスト名変換
【形式】

```
#include "nqsv.h"
char *NQSmid2hname( int mid, nqs_res *res )
int NQShname2mid( char *hostname, nqs_res *res )
```

【機能説明】

NQSmid2hname()は *mid* で指定されたマシン ID を対応するホスト名に変換し、ホスト名を表す文字列へのポインタを返します。 変換に失敗した場合はリザルトコードを *res* に設定し、NULL を返します。
NQShname2mid()は *hostname* で指定されたホスト名を対応するマシン ID に変換し、マシン ID を表す整数値を返します。 変換に失敗した場合はリザルトコードを *res* に設定し、負の整数を返します。

【エラー】

エラーの場合、以下のうちの一つをリザルトコード内のエラー番号に設定します。

NQS_ENOTCONN	バッチサーバに接続していません。
NQS_ECONFAIL	バッチサーバとの接続に失敗しました。
NQS_EPKTSEND	APIパケットの送信に失敗しました。
NQS_EPKTRECV	APIパケットの受信に失敗しました。
NQS_EDISCONN	通信中にバッチサーバとの接続が切断されました。
NQS_ENETDB	指定されたホスト名を持つホストが見つかりませんでした。
NQS_ENAMPDB	NMAPデータベースに登録されていないホストが指定されました。
NQS_ENOMEM	メモリの動的確保に失敗しました。
NQS_EINVAL	不正な引数が指定されました。

【関連項目】

NQSconnect (3)

第7章　　サンプルコード

本節では NQSV/API を利用した簡単なサンプルコードを紹介します。

7.1　実行ホストの負荷情報表示

ジョブサーバを実行している各実行ホストの負荷情報を、5 秒間隔で表示します。

```
#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　リクエスト状態系イベントの表示

リクエスト状態系イベントを受信し、各バッチリクエストの状態遷移を表示します。

```
#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 */
}
```

付録 A　発行履歴

A.1　発行履歴一覧表

2018 年	2 月	初版
2019 年	9 月	第 4 版
2020 年	1 月	第 5 版
2020 年	7 月	第 6 版
2021 年	3 月	第 7 版
2023 年	1 月	第 8 版

A.2　追加・変更点詳細

第 6 版

- 4.6 キュー属性（バッチキュー、会話キュー、グローバルキュー）

  緊急リクエストに関する項目を追加

- 4.9 リクエスト属性

  SIGTERM捕捉可否についての項目を追加

第 7 版

- マルチクラスタに関する記述を削除
- 4.8 リクエスト属性

  PlatformMPIを追加

第 8 版

- 2.1 キュー内での状態遷移

  状態遷移図を更新

索 引

A

API イベント....11
API 関数.... 85
ARRIVING 状態 7

B

BMC.... vi

C

CHKPNTING 状態 8

E

EXITED 状態 9
EXITING 状態 8

H

HCA.... vi
HELD 状態.... 9
HOLDING 状態 8

I

IB vi

M

MIGRATING 状態 9
MOVED 状態 10
MPI.... vi

N

NIC vi
nqs_aid 60
nqs_alist 60
nqs_cotemplate 60
nqs_cpuset.... 60
nqs_crid 60
nqs_crresource 61
nqs_entry.... 61
nqs_event 62
nqs_gbcset 63
nqs_gdesc 64
nqs_gpuinfo 64
nqs_hca.... 76
nqs_hid 64
nqs_hilo 64
nqs_hilo_g.... 65
nqs_hilo_u 65
nqs_hst 65
nqs_int_g 65
nqs_int_u.... 66
nqs_jcond.... 66
nqs_jid 66
nqs_jsvid.... 66
nqs_jsvst.... 66
nqs_keyval.... 67
nqs_license 67
nqs_mdesc 67
nqs_migfile 67
nqs_migprm 67
nqs_netreq.... 68
nqs_ngrpid.... 69
nqs_nreqst.... 69
nqs_nsubreq 69
nqs_odesc.... 70
nqs_ostemplate 70
nqs_pdesc 71
nqs_qdesc 71
nqs_qid 71
nqs_qst 71
nqs_quecrinfo 72
nqs_range.... 74
nqs_reqcrinfo.... 74
nqs_res 74
nqs_rgrp 74
nqs_rid.... 75
nqs_rlim 75
nqs_rlim_g.... 76
nqs_rlim_u 76
nqs_rnum 76
nqs_rrange 76
nqs_rsgavg 77
nqs_rsginfo 78
nqs_rsgres 77
nqs_rst.... 78
nqs_rstf.... 81
nqs_schid.... 81
nqs_socket 81
nqs_stgfile 81
nqs_temp_reqs 82
nqs_template.... 83
nqs_udesc 83
nqs_uexit.... 83
nqs_upp 84
nqs_wid.... 84

NQSaadd 87
NQSadel 88
NQSafree 86
NQSalist 85
NQSapiver 172
NQSaref 89
NQSattachhst 105
NQSattachhst_bm 166
NQSattrbsv 103
NQSattrcustom 170
NQSattrhst 115
NQSattrjob 147
NQSattrjsv 109
NQSattrngrp 152
NQSattrque 118
NQSattrreq 128
NQSattrsch 102
NQSbindjsv 124
NQSbindngrp 158
NQSbindsch 122
NQScloseattr 90
NQSclosecustom 168
NQSclosehst 113
NQSclosejob 145
NQSclosejsv 107
NQSclosengrp 150
NQScloseprm 126
NQScloseque 116
NQSclosereq 126
NQSclosesch 100
NQSconnect 93
NQScrecustom 171
NQScrengrp 153
NQScreostemp 163
NQScreque 120
NQScrereq 129
NQScretemp 159
NQSctljsv 110
NQSdelcustom 171
NQSdelngrp 153
NQSdelostemp 163
NQSdelque 121
NQSdelreq 136
NQSdeltemp 159
NQSdetachhst 105
NQSdetachhst_bm 166
NQSdisconnect 95
NQSeditostemp 164
NQSedittemp 161
NQSevent 96
NQSevflt 97
NQSexcludejsv 156
NQSexcludejsvrange 156
NQSexcludengrp 156
NQShldreq 139
NQShname2mid 173
NQSincludejsv 154
NQSincludejsvrange 154
NQSincludengrp 154
NQSleadreq 129
NQSlockostemp 165
NQSlocktemp 162
NQSmid2hname 173
NQSmigjob 149
NQSmovreq 144
NQSopenattr 90
NQSopenbreq 126
NQSopencustom 168
NQSopenhst 113
NQSopenireq 126
NQSopenjob 145
NQSopenjsv 107
NQSopenngrp 150
NQSopenprm 126
NQSopenque 116
NQSopenreq 126
NQSopensch 100
NQSreadattr 90
NQSreadcustom 168
NQSreadhst 113
NQSreadjob 145
NQSreadjsv 107
NQSreadngrp 150
NQSreadprm 126
NQSreadque 116
NQSreadreq 126
NQSreadsch 100
NQSregsch 98
NQSrewindcustom 168
NQSrewindhst 113
NQSrewindjob 145
NQSrewindjsv 107
NQSrewindngrp 150
NQSrewindprm 126
NQSrewindque 116
NQSrewindreq 126
NQSrewindsch 100
NQSrlsreq 141
NQSrqureq 143
NQSrstreq 142
NQSrunreq 134
NQSshutbsv 104
NQSsigjob 148
NQSsigreq 137
NQSstgreq 132
NQSunbindjsv 125
NQSunbindngrp 158
NQSunbindsch 123
NQSunlockostemp 165
NQSunlocktemp 162

P

POST-RUNNING 状態 8
PRE-RUNNING 状態 8

Q

QUEUED 状態 8

R

RESTARTING 状態 9
RESUMING 状態 9
RUNNING 状態 8

S

STAGING 状態 8
SUSPENDED 状態 9
SUSPENDING 状態 9

T

TRANSITING 状態 7

V

VE vi
VI クラスター vi

W

WAITING 状態 7

か

カスタムリソース属性 57

き

キュー属性 25

し

実行ホスト属性 23
ジョブサーバ属性 21
ジョブ属性 52

す

スケジューラ属性 20

そ

属性一覧 15

て

転送キュー属性 38

ね

ネットワークキュー属性 40

の

ノードグループ属性 55

は

バッチサーバ属性 16

へ

ベクトルエンジン vi
ベクトルホスト vi

り

リクエスト属性 41
リクエストのストール 10

わ

ワークフロー属性 56

NEC Network Queuing System V (NQSV)
利用の手引 [API編]

2023年 1月 第8版

日本電気株式会社

東京都港区芝五丁目７番１号

TEL(03)3454-1111（大代表）