JobCenter

R13.1

<NQS機能利用の手引き>

■Windows XP, Windows Server 2003, Windows Server 2008, Windows Server 2012 および Excel は、米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。

■UNIX は、The Open Groupが独占的にライセンスしている米国ならびに他の国における登録商標です。

■Solaris は、米国 Sun Microsystems 社の登録商標です。

■SAP, ERP, BI は、SAP AG の商標もしくは登録商標です。

■HP-UX は、米国 Hewlett-Packard 社の商標です。

■AIX は、米国 IBM Corporation の商標です。

■NQSは、NASA Ames Research Center のために Sterling Software 社が開発した Network Queuing System です。

■その他、本書に記載されているソフトウエア製品およびハードウエア製品の名称は、関係各社の登録商標または商標です。

なお、本書内では、R、TM、cの記号は省略しています。

輸出する際の注意事項
本製品(ソフトウエア)は、外国為替令に定める提供を規制される技術に該当いたしますので、日本国外へ持ち出す際には日本国政府の役務取引許可申請等必要な手続きをお取り下さい。許可手続き等にあたり特別な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下さい。

はじめに

本書は、JobCenter の基盤である NQS(Network Queuing System)の機能を JobCenter から利用する方法について説明しています。なお、本書内に記載されている画面例と実際の画面とは異なることがありますので注意してください。

本書の内容は将来、予告なしに変更する場合があります。あらかじめご了承下さい。

1. 読み方

JobCenter を新規にインストール、またはバージョンアップされる場合

→ インストールガイドを参照してください。

JobCenter を初めて利用される場合

→ クイックスタート編を目次に従いお読みください。

JobCenter の基本的な操作方法を理解したい場合

→ 基本操作ガイドを目次に従いお読みください。

環境の構築や各種機能の設定を理解したい場合

→ 環境構築ガイドを参照してください。

JobCenter の操作をコマンドラインから行う場合

→ 本書をお読みください。

その他機能についてお知りになりたい場合

→ 関連マニュアルの内容をお読みいただき、目的のマニュアルを参照してください。

2. コマンドの表記方法

(例)

コマンド中の | は、「または」を意味します。

各オプションは、「-英文字 オプション名($xxx)」または「-数字(-$xxx)」となります。

引数は、「引数名($xxx)」となります。

[ ]付きのオプションは、省略可能です。

・・・ は直前の記述が繰り返し可能なことを意味します。

3. 凡例

本書内での凡例を紹介します。

	気をつけて読んでいただきたい内容です。
	本文中の補足説明
注	本文中につけた注の説明
__	UNIX版のインストール画面の説明では、__部分(下線部分)はキーボードからの入力を示します。

4. 関連マニュアル

JobCenter に関するマニュアルです。JobCenter メディア内に格納されています。

最新のマニュアルは、JobCenter 製品サイトのダウンロードのページを参照してください。

http://www.nec.co.jp/middle/WebSAM/products/JobCenter/download.html

資料名	概要
JobCenter インストールガイド	JobCenterを新規にインストール、またはバージョンアップする場合の方法について説明しています。
JobCenter クイックスタート編	初めてJobCenterをお使いになる方を対象に、JobCenterの基本的な機能と一通りの操作を説明しています。
JobCenter 基本操作ガイド	JobCenterの基本機能、操作方法について説明しています。
JobCenter 環境構築ガイド	JobCenterを利用するために必要な環境の構築、環境の移行や他製品との連携などの各種設定方法について説明しています。
JobCenter NQS機能利用の手引き	JobCenterの基盤であるNQSの機能をJobCenterから利用する方法について説明しています。
JobCenter 操作・実行ログ機能利用の手引き	JobCenter CL/Winからの操作ログ、ジョブネットワーク実行ログ取得機能および設定方法について説明しています。
JobCenter コマンドリファレンス	GUIと同様にジョブネットワークの投入、実行状況の参照などをコマンドラインから行うために、JobCenterで用意されているコマンドについて説明しています。
JobCenter クラスタ機能利用の手引き	クラスタシステムでJobCenterを操作するための連携方法について説明しています。
JobCenter Helper機能利用の手引き	Excelを用いたJobCenterの効率的な運用をサポートするJobCenter Definition Helper (定義情報のメンテナンス)、JobCenter Report Helper (帳票作成)、JobCenter Analysis Helper (性能分析)の3つの機能について説明しています。
JobCenter SAP機能利用の手引き	JobCenterをSAPと連携させるための方法について説明しています。
JobCenter UCXSingleジョブ利用ガイド	JobCenterをUCXSingleと連携させるための方法について説明しています。
JobCenter WebOTX Batch Server連携機能利用の手引き	JobCenterをWebOTX Batch Serverと連携させるための方法について説明しています。
JobCenter Web機能利用の手引き	Webブラウザ上でジョブ監視を行うことができるJobCenter CL/Webについて説明しています。
JobCenter テキスト定義機能の利用手引き	ジョブネットワークやスケジュール、カレンダ、カスタムジョブテンプレートを、テキストファイルを使って定義する方法を説明しています。
JobCenter R13.1 リリースメモ	バージョン固有の情報を記載しています。

5. 改版履歴

版数	変更日付	項目	形式	変更内容
1	2012/07/31	新規作成	－	第1版
2	2012/09/07	修正	－	負荷分散機能の記載を一部修正
3	2012/10/05	版改訂	－	R13.1.1リリースに伴い版改訂
4	2012/12/21	版改訂	－	R13.1.2リリースに伴い版改訂

目次

図目次

表目次

第1章 概要

JobCenterはUNIXやWindows上でバッチ処理を行うためのシステムです。バッチ処理とは、リクエストを受け付けてキューイングして順番に処理する機能です。

JobCenter　の利用により、資源利用のバランスをコントロールしてシステムの効率を上げることができます。JobCenterはNQSというバッチ処理システムを独自に拡張して実装しています。

JobCenterはバッチリクエストというユーザ定義のリクエストと、ネットワークリクエストという実行結果ファイルの転送のみに使用されるリクエストを扱います。

■バッチリクエストは実行すべきジョブをシェルスクリプト(またはバッチスクリプト)として記述し、一括して実行するものです。このリクエストには資源制限値、実行日時、優先順位などの属性を指定することができます。また、複数のリクエスト間の実行順序を定義してジョブネットワークを構成することができます。

■ネットワークリクエストはユーザが投入したリクエストの結果ファイルを転送するためにJobCenterが使用するリクエストで、ユーザが自身で投入することはありません。

JobCenterはリクエストを受け付けると一旦キューに登録し、順番に処理していきます。キューはシステムの中に複数定義することができます。それぞれのキューには同時実行可能数、投入可能ユーザ、資源制限、キュー間実行優先順位などの属性を定義することができます。

パイプキューと呼ばれるリクエスト転送用のキューを定義することにより、リクエストの性質によって投入するキューを自動的に振り分けたり、ネットワークを介したリモートホストにリクエストを投入することができます。さらにネットワーク上の各マシンの負荷に応じてリクエストを割り振り、負荷分散を行うこともできます。

利用者は投入したリクエストに対して状態監視、取消、属性変更、保留、移動などを行うことができます。

JobCenterでは利便性を向上させるために管理者向けや利用者向けのGUIインタフェースを提供しています。GUIはWindows上で利用できます。またクラスタ機能、ERP Option機能やBI Option機能などのオプション機能を提供しています。

■GUI機能やオプション機能についての詳細は「JobCenter　基本操作ガイド」を参照してください。

■キューやリクエストなどの NQS 機能については本マニュアルを参照してください。

1.1. 機能概要

JobCenter は、以下の機能を提供しています。

■SV 機能

ジョブを実行するサーバ機能を提供します。 SV 機能は、 CL 機能を含んでいます。

■CL 機能

ジョブを投入するクライアント機能を提供します。リクエストを投入することはできますが、ジョブ実行は SV 機能の動作するマシンにリクエストを転送して行うことになります。

■API 機能

SV 機能 または CL 機能を利用するための C 言語インタフェースです。この機能を使用するには、 SV 機能 または CL 機能 が必要です。

■EUI 機能

利用者向けの GUI を提供します。環境の構築は UMS 機能 で行う必要があります。この機能を使用するには、 SV 機能 または CL 機能 が必要です。

■UMS 機能

管理者向けの GUI を提供します。ネットワーク上の 1 台のマシンにインストールすることにより、集中的に管理を行うことができます。 UMS 機能は、 EUI 機能およびCL 機能を含んでいます。

■CJC 機能

ジョブマイグレーションなどのクラスタジョブ機能を提供します。この機能を使用するためには障害発生時にデータの移動が可能な共有ディスク装置が必要です。また、本機能を使用するためには、最低2台のサーバと1台のマネージャが必要です。

■ERP 連携機能

SAP ERP との連携機能を提供します。 GUI から ERP ジョブを作成、投入することができます。

■BI 連携機能

SAP BI との連携機能を提供します。 GUI から BI 上のインフォパッケージとプロセスチェーンの起動と状態監視をすることができます。

1.2. 対応製品

JobCenterの製品群および他システムとの接続について説明します。

1.2.1. 製品一覧

以下の製品を提供しています。

表1.1 JobCenterの提供製品

JobCenter UNIX 製品	JobCenter Windows 製品
JobCenter MG	JobCenter MG
JobCenter SV(T1) , SV(T2) , SV(T3)	JobCenter SV(T0) , SV(T1) , SV(T2) , SV(T3)
JobCenter CJC Option	JobCenter CJC Option
JobCenter for ERP Option	JobCenter for ERP Option
JobCenter for BI Option	JobCenter for BI Option
JobCenter Media	JobCenter Media
	JobCenter CL/Win

なお、オペレーティングシステムによってはサポートされていない製品があります。詳しくはリリースメモを参照してください。

1.2.2. 他システムとの接続について

ここではUNIX版JobCenterと、Windows版JobCenterまたはSUPER-UX　NQSとの接続について説明します。

1.2.2.1. SUPER-UX NQS との接続について

SUPER-UX　NQS　は、弊社スーパーコンピュータ「SX　シリーズ」上のUNIXで使用できるNQSです。機能の詳細な説明はSUPER-UX NQSに付属する『NQS 利用の手引』をご覧ください。

JobCenter　からジョブの投入を行うときに、SUPER-UXの機能を使用するための、いくつかのオプションが指定できます。またqstatなどのコマンドを用いてSUPER-UX上のジョブの状態を調べることができます。詳しくは、4章 「JobCenter ユーザコマンド一覧」 をご覧ください。

接続にあたっては、nmapmgr（1M）コマンドで指定するマシンタイプを "nec" のまま使用するようにしてください。

1.2.2.2. Windows版JobCenterとの接続について

以下のような制限がありますのでご注意ください。詳細についてはリリースメモをご覧ください。

■シェルスクリプトの代わりにバッチファイル形式で記述する必要があります。

■日本語処理コード体系が通常のUNIXとは異なります。

■GUI専用のためqstatなど情報表示系のコマンドで情報を表示できません。

■qsubなどで指定する結果ファイルのパス名においてドライブ名 (A: など)を使用することができます。ただし1文字のマシン名は、ドライブ名として解釈されます。

JobCenter　CL/Winの提供するGUIではなく、nmapmgr　(1M)　コマンドでマシン登録を行う場合、UNIX系マシン上のnmapmgrの設定において、Windows版JobCenterのNQS TYPEを"necnt"で登録してください。また、Windowsマシン上のnmapmgrの設定では、UNIX版JobCenterのNQS TYPEを"nec"で登録してください。詳細は「6.5.2 リモートマシン定義」を参照してください。

第2章 JobCenterの構成

JobCenterの処理は、基本処理単位であるリクエストと、リクエストを蓄積するキューの2つの要素から構成されています。

2.1. リクエスト

リクエストはJobCenterにおける基本処理単位で、リクエストを投入することによりはじめてJobCenter　の機能が利用可能となります。リクエストにはバッチ、ネットワークの　2種類があります。以下 2種類のリクエストについて説明します。

2.1.1. バッチリクエスト

バッチリクエストはプログラムの実行を依頼するリクエストです。このリクエストはUNIX版ではシェルスクリプト形式で記述したもの、Windows版ではバッチファイル形式で記述したものをそれぞれ投入することで実現されます。また、このスクリプトのコメント部分にJobCenterに関する操作情報を埋め込むことができ、リクエスト投入時のわずらわしいオプション指定を軽減することができます。

Windows版ではバッチリクエストの埋め込みオプションはサポートしていません。

バッチリクエストはリクエスト実行環境などに関する属性をもっていますが、この属性は利用者が指定することもできますし、指定しなければシステムにより自動的に付加されます。また、投入後に変更できるものもあります。

リクエストの属性としては主に以下のものがあります。

表2.1 リクエストの主な属性とその意味

属　性	意　味
リクエストID	リクエストを特定するために、システムがリクエストにつける固有のIDです。 これはホスト内での通し番号とホスト名を連結したもので、ネットワークを通じて一意なものになります。
リクエスト名	ユーザがリクエストを識別しやすいように、投入時につけるリクエストの名前です。
資源制限値 注1	リクエストが実行時に使用する各資源の制限値です。 代表的なものとしてはファイルサイズ、CPU時間、メモリサイズなどがあります。この属性については後ほど詳しく説明します。
リクエストプライオリティ	リクエストプライオリティは、JobCenter　キュー内で登録されているリクエストの起動順序を決定します。 これは単に起動される順番を決定するもので、リクエスト実行時の優先度とは何の関係もありません。
nice値 注2	バッチリクエストのプロセスの実行優先度を変更できます。
実行シェル 注3	バッチリクエストのシェルスクリプトを実行するシェルプログラムです。
実行結果出力ファイル	バッチリクエストの実行結果が入れられるファイルです。 このファイルは特に指定しない限り、シェルスクリプトの標準出力用と標準エラー出力用の2つのファイルが作成されます。
実行遅延時間	バッチリクエストは通常即座に起動されるようになっていますが、この属性が付加されるとバッチリクエストの起動が指定時間まで遅らされることになります。

注1　Windows版JobCenterでは、バッチリクエストの属性として指定された資源制限値はWindows側で無視されます。

注2　バッチリクエストの属性として指定されたnice値は、Windows上では次のとおり解釈されます。

nice値指定	Windows 上でのプロセスプライオリティクラス
-20	REALTIME
-19～-1	HIGH
0～18	NORMAL
19	IDLE

注3 Windows版の実行シェルはCMD.EXEです。CMD.EXE以外を実行シェルとして指定した場合の動作は保証できません。

2.1.2. バッチリクエストとプロセスグループID (UNIX版)

バッチリクエストが起動されると、そのリクエストを構成するプロセスにプロセスグループID　が設定されます。JobCenterではこのプロセスグループIDで実行リクエストを管理しており、実行リクエストの削除・シグナル送信のときなどに使用します。

リクエストのプロセスグループIDはリクエスト状態表示コマンド(qstat、qstatrコマンドなど)で参照することができます。また、リクエストを構成するプロセスの状況を知りたい場合は　ps(1)コマンドでプロセスグループIDをキーにして調べてください。

バッチリクエスト終了時には、プロセスグループに属するすべてのプロセスは強制終了されます。常駐プロセスをJobCenterのジョブから起動するような使い方をしている場合は、UNIX版ではJobCenterのnqsbgコマンド、Windows版ではstartコマンドを利用するなどでプロセスグループを切り離して起動する必要があります。

2.1.3. バッチリクエストの資源制限

サポートしている資源制限は以下のとおりです。資源制限は setrlimit(2) を使用しています。

■プロセスごとのコアファイルサイズ制限

■プロセスごとのデータセグメントサイズ制限

■プロセスごとの永久ファイルサイズ制限

■プロセスごとのメモリサイズ制限

■プロセスごとのナイス実行値

■プロセスごとのスタックセグメントサイズ制限

■プロセスごとの CPU 時間制限

これらの制限には、リクエストを実行するシェル自身のプロセスの資源も含まれます。

ただし、以上の資源制限がすべて、実行ホストでサポートされているとは限りません。リクエストが実行されるホスト上でサポートされていない資源制限は無視されます。各ホストでサポートされている資源制限はqlimit(1) コマンドで参照することができます。

2.1.4. ネットワークリクエスト

ネットワークリクエストは、バッチリクエストの結果ファイルを指定されたホストに転送するためのリクエストです。ネットワークリクエストはバッチリクエスト終了時に自動的に生成され、

ネットワークキューに投入されます。したがって、ユーザによるネットワークリクエストの作成および投入はできません。

リクエストの属性としては主に以下のものがあります。

■リクエストプライオリティ

バッチリクエストと同様にキュー内のリクエストの起動順番を決定します。

■イベント番号

転送しようとしているファイルの種類を示す番号で、現在は以下の 2 つが用意されています。

30	標準出力ファイル転送
31	標準エラー出力ファイル転送

Windows版JobCenterではネットワークリクエストはサポートしていません。

2.1.5. リクエストの状態

リクエストは投入されてから消滅するまでの間、さまざまな状態を遷移します。リクエストの状態には以下の種類があります。

表2.2 リクエストの状態とその意味

状　態	意　味
RUNNING	リクエストは実行中です。
QUEUED	実行待ちの状態で、スケジュールの対象となります。JobCenter　で定められた順番に従って逐次 RUNNING 状態になります。
WAITING	-aオプションにより、実行開始時刻を待ち合わせています。
HOLD	qhold コマンドなどにより、保留されています。
SUSPEND	qspnd コマンドなどにより、実行一時中断中です。
ROUTING	パイプキューからほかのキューへの送信中です。
ARRIVING	パイプキューからの受信中です。
EXITING	リクエストの実行結果ファイル (STDOUT、 STDERR) を出力中です。

2.2. キュー

JobCenterキューとは、JobCenterが受け付けたリクエストを一時的に溜めておくものであり、JobCenter はこのキューに溜まっているリクエストを順番に実行していきます。

キューにリクエストを登録する場合、リクエストに設定された属性とそのキューに定められた属性の不一致により、リクエストの登録が拒否される場合があります。またキューの状態によってはリクエストの登録や起動ができない場合もあります。

JobCenterキューには、バッチ、パイプ、ネットワークの3種類のキューが用意されています。

2.2.1. バッチキュー

バッチキューとは、バッチリクエスト専用のキューです。このキューには以下に示すような属性がついています。

表2.3 バッチキューの属性

属　性	意　味
資源制限量	キューに登録されるリクエストの資源制限使用量と比較される制限値です。 登録されるリクエストに設定された資源制限値がこの値を超える場合、リクエストの登録は拒否されます。
キュープライオリティ	キュー間での優先度を示す値です。 JobCenter　がリクエストを実行する場合に、どのキューにあるリクエストを最初に実行するかを決めるときに使用されます。この値が大きいキューのリクエストが先に実行されます。優先度が同じキュー同士では、キューへの投入時刻順に従います。
同時実行可能リクエスト数	キュー内で同時に実行できるリクエスト数です。 現在実行しているリクエスト数がこの数に達していた場合、次に実行されるべきリクエストは起動を待たされ、現在実行しているリクエストのどれかが終了するまで起動されません。
キューアクセス状態	キューには投入できるユーザ、グループが設定されている場合があります。この場合、利用者がユーザまたはグループに設定されていないと、そのキューへのリクエストの投入はできません。
pipeonly 属性	この属性がついているキューにはパイプキュー　(「2.2.2　パイプキュー」参照)を経由しなければリクエストの投入ができません。
デマンドデリバリ機能	デマンドデリバリ方式による負荷分散機能を使用するバッチキューに設定されます (「6.7.3 デマンドデリバリ方式」参照)。

2.2.2. パイプキュー

パイプキューとは、リクエスト転送用のキューです。このキューに投入されたリクエストは、他のキューに転送されます。

転送先のキューはローカルホスト上のキューはもちろん、ネットワークを介したリモートホスト上のものでもよく、リモートホストにリクエストを投入する場合はパイプキューを介して投入するという形になります。

このキューには以下に示すような属性がついています。

表2.4 パイプキューの属性

属　性	意　味

転送先キューリスト	転送先のキューのリストです。 キューに投入したリクエストはこの転送先キューリストに設定されたキューに転送されることになります。この転送先キューリストには複数の転送先キューが設定される場合がありますが、設定されている順に現在転送可能なキューがリクエストの転送先のキューとして選ばれます。この転送先キューリストにリモートホスト上のキューが設定されていれば、ネットワーク転送が可能ないわゆるネットワークパイプキューということになります。
キュープライオリティ	キュー間での転送優先度を示す値です。 JobCenter　がリクエストを実行する場合に、どのキューにあるリクエストを最初に転送するかを決定するために使用します。この値が大きいキューのリクエストが先に転送されます。優先度が同じキュー同士では、キューへの投入時刻順に従います。
同時転送可能リクエスト数	キュー内で同時に転送できるリクエスト数です。 現在転送中のリクエスト数がこの数に達していた場合、次に転送されるべきリクエストは転送を待たされ、現在転送中のリクエストのどれかが転送を完了するまで転送を開始されません。
キューアクセス状態	キューには投入できるユーザグループが設定されている場合があります。この場合、利用者がユーザまたはグループに設定されていないと、そのキューへの投入はできません。
pipeonly 属性	この属性がついているキューにはパイプキューを経由しなければリクエストの投入ができません。
事前チェック機能	この属性を指定すると、リクエストをパイプキューに登録する前にそのパイプキューの目的地となっているキューの状態を調べます。そして以下の条件を同時に満たしていればパイプキューに登録されます。 ■目的地のキュー上でリクエストが投入可能かつ実行可能 ■リクエストの資源制限≦目的地のキューの資源制限
ステイウェイト	パイプキューに時間指定　(qsubコマンドの　"-a"　オプション) のリクエストが投入された場合、そのリクエストをパイプキュー上でウェイトさせます。
サーバ	サーバはリクエストをほかのキューに転送するプログラムです。 設定されているプログラムによって目的地の選択方法が異なります。
透過型機能	高速かつ低負荷でローカルのバッチキューにリクエストを転送することができます(「5.5　透過型パイプキューの概要と設定方法」参照)。
デマンドデリバリ機能	負荷分散用のパイプキューであり、デマンドデリバリ方式の機能をもっています(「6.7.3 デマンドデリバリ方式」参照)。

2.2.3. ネットワークキュー

ネットワークキューとは、実行結果出力ファイルの転送用キューです。このキューは実行結果出力ファイルを利用者のもとに転送するためにJobCenterが使用するキューで、利用者がこのキューに直接リクエストを投入することはできません。

Windows版JobCenterではネットワークキューはサポートしていません。

バッチリクエストの実行が終了すると、JobCenterは実行結果出力ファイルを転送するためのネットワークリクエストを自動的に生成し、転送先に対応したネットワークキューに投入します。

このキューには以下に示すような属性がついています。

表2.5 ネットワークキューの属性

属　性	意　味
キュープライオリティ	キュー間での転送優先度を示す値です。 JobCenter　がリクエストを実行する場合に、どのキューにあるリクエストを最初に転送するかを決定するために使用します。この値が大きいキューのリクエストが先に転送されます。優先度が同じキューどうしでは、キューへの投入時刻順に従います。
同時転送可能リクエスト数	キュー内で同時に転送できるリクエスト数です。 現在転送中のリクエスト数がこの数に達していた場合、次に転送されるべきリクエストは転送を待たされ、現在実行しているリクエストのどれかが終了するまで起動されません。
転送先ホスト	ファイルを転送するホスト(サイト)名です。

2.2.4. キューの状態

キューには以下のような状態を取り、その状態によりリクエストの登録・実行が可能かどうかが決まります。

キューの状態は大別して2つの特性があります。第1特性はキューがリクエストの登録を受け付けるかどうかに関するものです。第2特性はリクエストを実行するかどうかに関するものです。

1. 第1特性

   リクエストはキューが投入可能 (enabled) であり, ローカル JobCenter デーモンが稼動中のときに限り、投入可能です。

ENABLED	キューはリクエストの登録を受け付ける状態です。
DISABLED	キューはリクエストの登録を受け付けない状態です。
CLOSED	JobCenter　システム停止中です。したがって、リクエストの登録はできません。

2. 第2特性

INACTIVE	キューはリクエストの実行を行う状態です。ただし、そのキュー上のリクエストで現在実行中のものがない状態です。
RUNNING	キューはリクエストの実行を行う状態です。また、そのキュー上のリクエストで現在実行中のものがある状態です。
STOPPED	キューはリクエストの実行を行わない状態です。また、そのキュー上のリクエストで現在実行中のものもない状態です。
STOPPING	キューはリクエストの実行を行わない状態です。ただし、そのキュー上のリクエストで現在実行中のものがある状態です。
SHUTDOWN	JobCenter システム停止中です。

第3章 JobCenterの操作方法

この章ではJobCenterシステムの利用者の方のために、nqsのコマンドを用いた実際のJobCenterの利用方法について説明します。ここではすでにJobCenter　システムの運用が開始されているものとします。

JobCenterを利用するためには、まずJobCenterへのアクセス権がなければなりません。アクセス権の有無の確認はqstata(1) コマンドを用いて行います。qstata(1) コマンドをオプションなしで実行すると以下のように表示されます。

■アクセス制限を受けていない場合

You are permitted to place requests in NQS.

■ユーザ単位でアクセス制限を受けている場合

You are not permitted to place requests in NQS.

■グループ単位でアクセス制限を受けている場合

Your group is not permitted to place requests in NQS.

アクセス制限を受けているユーザは JobCenter を利用することができませんのでJobCenter管理者に問い合わせてください。

なおリモートホストのアクセス制限状態を参照する場合は、 -h オプションでホスト名を指定してください。

Windows版JobCenterではqstataコマンドはサポートしていません。

3.1. バッチリクエストの作成から終了まで

以下にバッチリクエストの作成から終了までに行う手順を示します。

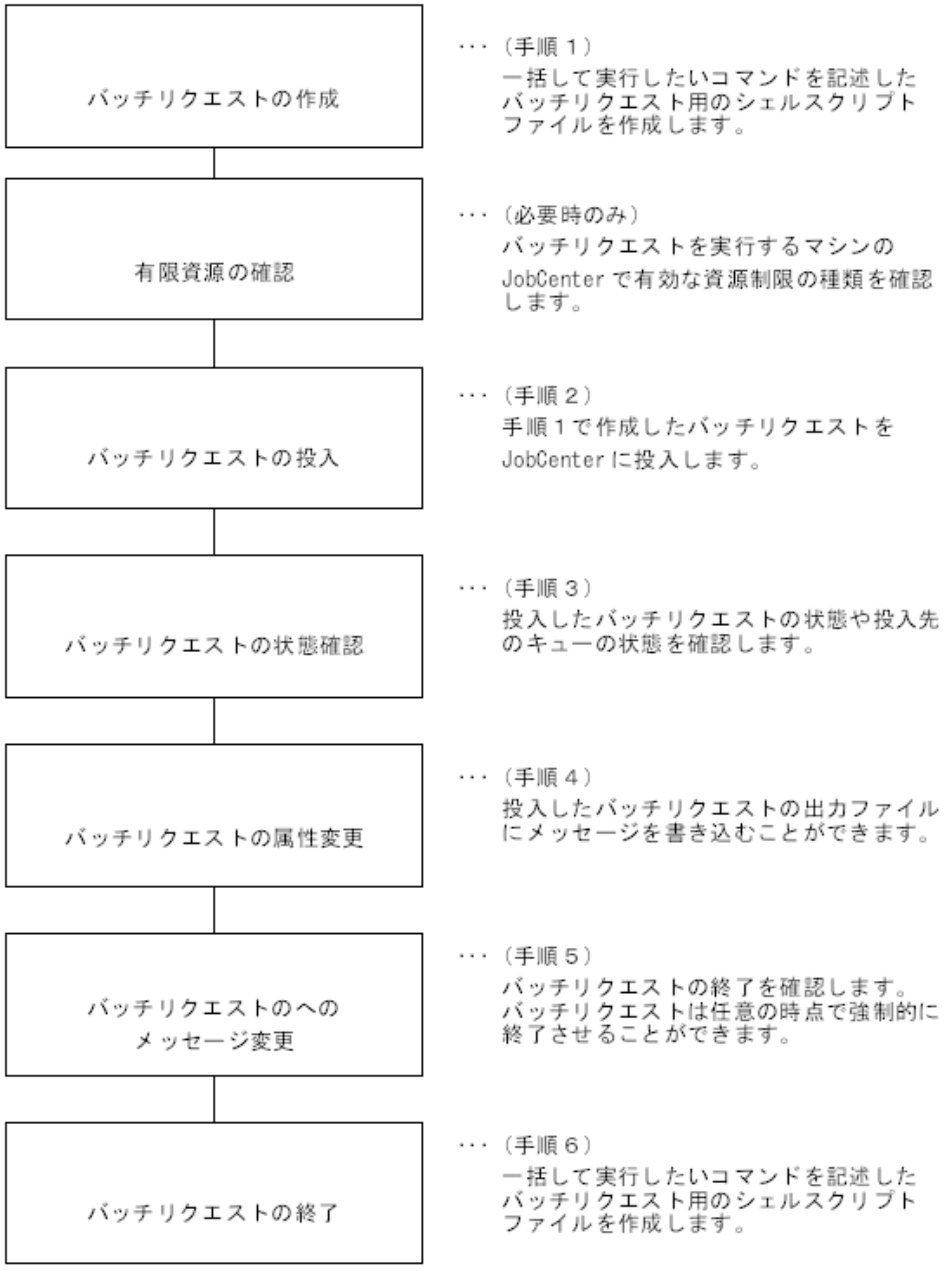


図3.1 バッチリクエストの作成から終了までの手順

3.1.1. バッチリクエストの作成

JobCenter　を使用する最初の手順は、バッチリクエスト用のシェルスクリプトファイルを作成することです。このシェルスクリプトがバッチリクエストとしてJobCenter　に投入され実行されます。このシェルスクリプトの中には通常のシェルスクリプトの場合と同じように任意のコマンドを組み合わせて記述することができます。

ただし、

■シェルスクリプトの実行はJobCenterによってどの端末装置とも無関係に実行されますので、端末装置に対する入出力を要求するようなコマンド(たとえば、 stty(1) コマンドなど)を使用することはできません。

■シェルスクリプトの実行中は、その標準出力、標準エラー出力はそれぞれ任意のファイルに結び付けられているため、処理の途中でインタラクティブな操作を必要とするようなシェルスクリプトはバッチリクエストとして実行することはできません。

上記の制限と後述のJobCenterに対する投入時のオプションをコメント部に記述できる点を除いて、バッチリクエスト用のシェルスクリプトは通常の場合のシェルスクリプトとまったく同じです。また、このシェルスクリプトを解釈するシェルを自由に指定することができるため、sh 用のシェルスクリプトでも csh 用のシェルスクリプトでもその他のシェル用のシェルスクリプトでもかまいません。

以下にバッチリクエスト用シェルスクリプトの例を示します。

```
#
# sample batch request
#
optf77 -o prog1 prog1.f
if [ $? -ne 0 ]
then echo "prog1.f compile error"
exit 1
fi
prog1 < in_data
```

シェルコマンドが入力すべきデータを指定する方法は2つあります。

最初の方法は、次に示すようにコマンドが入力するデータを格納したデータファイルを用意することです。

```
sort < input_data
```

上記の例では、ソートすべきデータは input_data というファイルに格納されています。2番目の方法は、次のようにヒアドキュメントを使用するものです。

```
sort << EOF
Rebert Cohn was
once middeweight boxing
champion of Princeton
EOF
```

上記の例では、sortコマンドの次の行からEOFと書かれた行の直前の行がソートされます。

バッチリクエスト用シェルスクリプトには、そのコメント部にJobCenterの投入時オプションを埋め込むことができます。

オプションの指定方法は、最初のシェルコマンドが現れる前のコメント部分に "@$" という文字列に続けて指定します。ここにはバッチリクエストの投入コマンドであるqsub(1) コマンドのすべてのオプションが記述できます。この埋め込みオプションは通常のシェルにとってはコメント行とみなされるため、シェルスクリプトの実行には影響しません。

以下に埋め込みオプションを用いた例を示します。

```
#
# BATCH request script
#
# @$-a "11:30pm" -lt "21:10"      午後 11 時 30 分に開始、CPU
#                                 使用時間を 21 分 10 秒に制限
```

```
#
# @$-q batch1                       zbatch1 というキューに投入する
# @$                                埋め込みオプション終了宣言
make
```

"@$" のあとに "-" が続いていない行は、埋め込みオプションがこれ以上存在しないという意味になります。また上記の例のように 1 行に複数のオプションを埋め込むことも可能です。

3.1.2. バッチリクエストの投入

1. バッチリクエストの投入

   次の手順は作成したシェルスクリプトをバッチリクエストとして JobCenter に投入することです。バッチリクエストの投入は、コマンドによる場合は qsub(1) コマンドによって行います。

   また、コマンド列にオプションを指定することができます。コマンド列と埋め込みオプションに同じオプションを指定した場合には、コマンド列の方を有効とみなします。コマンド行にシェルスクリプトファイルを指定しなければ、そのまま標準入力からシェルスクリプトを読み込ませることもできます。

   以下にバッチリクエストの投入例を示します。

   ［投入例1］

```
$ qsub -q batch1 script1 ↵
Request 65.host1 submitted to queue: batch1.
$
```

   上記の例は、 batch1 というキューに script1 というシェルスクリプトを実行するバッチリクエストを投入したものです。 qsub コマンドの次の行に示されたメッセージは JobCenter が投入されたバッチリクエストを受理したことを示すメッセージです。このメッセージは次のことを示しています。

```
Request  65.host1  submitted  queue:  batch1.
           ↑                          ↑
           ①                          ②
```

   ①バッチリクエストに対してJobCenterが付けたリクエストIDで、投入したホスト内での連番とそのホスト名から構成され、ネットワーク上で一意に識別されます。

   ②バッチリクエストを投入したキュー名です。

   ［投入例2］

```
$ qsub -q batch1 ↵
make all
CTRL-D(EOF)
Request 66.host1 submitted to queue: batch1.
$
```

   上記の例は、標準入力からシェルスクリプトを読み込む場合の実行例です。

2. qsub投入時のオプション

   qsubによるバッチリクエスト投入時に指定できるオプションとしては、以下の種類のものが用意されています。

   a. 結果ファイル関係のオプション

      バッチリクエストの出力結果 (標準出力および標準エラー出力に出力されるもの) をどこに返すかを指定するオプションです。

通常バッチリクエストの出力結果はいったんJobCenter によって用意されたスプールファイルに出力され、バッチリクエストの実行がすべて終了した時点で所定のディレクトリ配下の所定のファイルにコピーされます。

特に指定しなかった場合は、バッチリクエストの出力結果はバッチリクエストを投入したディレクトリ配下の、「3.1.13 バッチリクエストの出力ファイル」で説明するファイルに返されます。

b. 資源制限用オプション

バッチリクエストは、その実行時に資源制限を行うことができます。資源制限とは、そのバッチリクエストで使用できる CPU 時間やメモリサイズ、ファイルサイズの最大値を設定しておき、バッチリクエストが設定した値を超えて実行しようとしたときにその実行を強制終了させる機能です。

JobCenter ではこの資源制限値を投入時のオプションとして指定することができます。特に指定しなかった場合は、そのバッチリクエストは投入したバッチキューに管理者が設定した資源制限値が適用されます。またオプションとして指定した場合は、投入先のバッチキューに管理者により設定されている値との比較が行われます。

もし、バッチキューの値よりもオプションで指定した値の方が大きい場合は、そのバッチリクエストの投入は拒否されます。なぜなら、そのバッチキューではそのような資源を多く消費するバッチリクエストを実行することができないからです。

設定できる資源制限の種類は各マシンのJobCenter によって異なりますが、各マシンでどのような資源制限が有効かは、 qlimit(1) コマンドで確認することができます。 qlimit コマンドの使用方法については、「3.1.11 有効資源制限の確認」を参照してください。

c. メール関係のオプション

JobCenter では、通常バッチリクエストの実行開始や実行終了が明示的には通知されません。利用者は、「3.1.3 バッチリクエストに関する状態確認」で説明する各コマンドを使用して、投入したバッチリクエストの状態を知ることができます。

一方、バッチリクエストの投入時オプションで、実行開始や実行終了をメールで通知してくるように指定することができます。このオプションが指定されたバッチリクエストの実行が開始されたときまたは終了したときは、 JobCenter からその旨を示すメールが届きます。また、メールの受け取り先を変更することもできます。既定値は投入した本人です。

注意事項

バッチリクエストの実行が正常に終了した場合は、特にオプションで指定したときを除き終了通知のメールは送られませんが、異常終了した場合はオプションの有無にかかわらず必ずその内容を示すメールが投入者に送られます。

d. その他のオプション

その他のオプションとして、バッチリクエストの実行開始時刻を指定するオプション、投入先のキューを指定するオプション、バッチリクエストのプライオリティを指定するオプションなどがあります。

以下に主なオプションの簡単な説明を示します。ただし、詳細な説明および、下記以外のオプションについては、4章 「JobCenter ユーザコマンド一覧」 の qsub(1) の項を参照してください。

3. qsub投入時オプションの説明

a. 結果ファイル関係オプション

-e $filename

バッチリクエストの標準エラー出力を格納する結果ファイルを指定します。

-o $filename

バッチリクエストの標準出力を格納する結果ファイルを指定します。

■ファイル名 $filenameの指定形式-

e、-o オプションに指定するファイル名は以下の形式で指定します。

[$machine:][[/]$path/]$stdout-filename

■完全に指定する場合の例

-e host1:/usr/nqs/result.e

ホスト host1 の /usr/nqs/result.e が結果ファイルになります。

■絶対パスで指定する場合の例

-e /usr/nqs/result.e

リクエストを host1 に投入したとすると、ホスト host1 の /usr/nqs/result が結果ファイルになります。

■相対パスで指定する場合の例

-e result.e

ファイル名が'/' で始まっていない場合は相対パス名と解釈されます。リクエスト投入マシン上に結果ファイルができる場合には、基準のディレクトリは投入時のカレントディレクトリです。投入マシン以外のマシンにファイルができる場合には、基準のディレクトリはそのマシン上のユーザのホームディレクトリです。

-eo

標準エラー出力を標準出力に併合し、結果ファイルを1つにまとめます。

-ke

標準エラー出力結果ファイルを残すホストを、実際にバッチリクエストが実行されたホストにします。

-ko

標準出力結果ファイルを残すホストを、実際にバッチリクエストが実行されたホストにします。

b. 資源制限用オプション

主なものだけをここに挙げます。

-lf $size-limit [,$warn-limit]

プロセスごとのファイルサイズの最大値や警告値を設定します。

-lm $size-limit [,$warn-limit]

プロセスごとのメモリサイズの最大値や警告値を設定します。

-ln $nice-value

プロセスのnice値を設定します。

-lt $time-limit [,$warn-limit]

プロセスごとの CPU 時間制限の最大値や警告値を設定します。

資源制限には最大値と警告値の2つを指定できるものがあります。警告値を指定する場合は最大値以下の値でなければなりません。警告値を省略した場合は、最大値と同じとみなされます。

バッチリクエストの使用資源量が警告値を超えた場合は、それぞれの資源制限で定められたシグナルがバッチリクエストに送信されます。最大値を超えた場合は直ちに実行が中断されます。

資源制限の警告値はバッチリクエストを実行するインプリメンテーションが警告値を実装している必要があります。

■資源制限値指定方式

上記のオプションの引き数の資源制限値は以下の形式で指定します。

■時間制限 (CPU 時間制限)

時間に関する制限値は以下の形式で指定します。

[[$hours:]$minutes:]$seconds[.$milliseconds]

[指定例]

指定値	意　味
1234:58:21.29	1234 時間 58 分 21.29秒
59:01	59 分 1 秒
12345	12345 秒
121.1	121.1 秒

■サイズ制限

サイズに関する制限値は以下の形式で指定します。

[$integer][.$fraction][$units]

$unitsに指定できるものは以下のとおりです。

形　式	意　味
b	バイト
kb	キロバイト

mb	メガバイト
gb	ギガバイト

なお、$unitsを指定しない場合はバイトと解釈されます。

[指定例]

指定値	意　味
1234	1234バイト
1234kb	1234キロバイト
1234.5gb	1234.5ギガバイト

c. メール関係オプション

-mb

リクエストの実行を開始したときにメールの発信を行います。

-me

リクエストの実行が終了したときにメールの発信を行います。

-mu $user-name

メールの送信先を指定します。オプションの引き数$user-nameは以下に示すようにuser (@文字を含まない) か、user@machineのどちらかの形式で指定します。

（例）

-mu user1現ホストのuser1 にメールを送信するようにする

-mu user1@host1ホストhost1 のuser1 にメールを送信するようにする

d. その他のオプション

-a $date-time

指定した時刻までリクエストの実行を待ちあわせます。

以下に時刻の指定例を示します。

指定値	意　味
01-Jan-1990 12am,GMT	世界標準時の 1990 年 1 月 1 日午前 12 時
Tuesday,23:00:00	火曜日の 23 時
11pm tues	火曜日の午後 11 時
tomorrow 23-GMT	世界標準時で明日の 23 時

時刻の詳しい指定方法についてはqsub(1) を参照してください。

以下のように時間指定中に空白をいれる場合は、シェルが 1 つの文字列と解釈できるように、ダブルクォートで囲むか空白をエスケープしなければなりません。

-a "July 4,2000 12:31-GMT"

-nr

リクエストが再実行不可であることを宣言します。

-p $priority

リクエストのキュー内プライオリティを設定します。指定する値は [0...63] の範囲の整数で、値が大きいほど優先度が高くなります。ユーザがリクエストプライオリティを指定しなかった場合はシステムが既定値を割り当てます。

-q $queue-name

バッチリクエストを登録するキューを指定します。このオプションを指定しない場合は、環境変数QSUB_QUEUEの文字列値がリクエストを投入するキューとなります。

QSUB_QUEUEが設定されていない場合は、システム管理者によって定義された既定バッチリクエストキューが投入するキューとなります。

既定バッチリクエストキューも定義されていない場合は、リクエストはキューに投入されることなく、エラーメッセージが出力されます。

-r $request-name

リクエスト名を指定します。リクエスト名を指定しない場合は以下のように自動的にリクエスト名がつけられます。

標準入力からスクリプトを入力した場合リクエスト名は "STDIN" になります。

スクリプトファイルを用いた場合はディレクトリ部分を除いたファイル名になります。例えばスクリプトファイルが /usr/nqs/script の場合は script がリクエスト名になります。

リクエスト名が数字で始まる場合、文字'R' が先頭に付加されます。またリクエスト名が63文字以上になる場合は63文字で打ち切られます。

-s $shell-name

バッチリクエストのスクリプトを実行するシェルの絶対パス名を指定します。このオプションを指定しない場合は、システムに設定された方式でシェルが選択されます。

シェル選択方式はqlimit(1)コマンドで確認することができます。シェル選択方式には以下の 3 とおりがあります。

■fixed

バッチリクエストを実行するシェルとして管理者により指定されたシェルが使用されます。

■free

バッチリクエストを実行する際に、まずリクエストのユーザのログインシェルが起動されます。次にそのログインシェルが、バッチリクエストの内容から適切なシェルを選択し、

そのシェルがバッチリクエストを実行します。つまり、あたかもインタラクティブな処理と同様な形態でバッチリクエストが実行されます。

■login

バッチリクエストを実行するとシェルとして、リクエストのユーザのログインシェルが使用されます。

3.1.3. バッチリクエストに関する状態確認

3.1.3.1. バッチリクエストの状態確認

投入したバッチリクエストの状態を確認するには、 qstatr(1) コマンドを用います。リクエストの指定はリクエスト ID で行いますので、リクエスト ID がわかっていればリクエストの直接指定が可能です。

以下にその例を示します。

```
$ qstatr 72.host1 ↵
============================================================
NQS (R11.10) BATCH REQUEST  HOST: host1
============================================================
REQUEST ID     NAME      OWNER    QUEUE    PRI NICE  STT    PGRP   R
-------------- --------- -------- -------- ---- ---  ----  ------  -
72.host1        STDIN    user1    batch1    20  10   RUN     800  -
------------------------------------------------------------------
$
```

STT　のカラムにリクエストの状態が表示されます。ここで表示される意味はそれぞれ以下のとおりです。

表3.1 バッチリクエストの状態

リクエストの状態	意　味
RUN	実行中
QUE	実行待ち状態
WAT	開始時刻の待ち合わせ中
HLD	保留中
SUS	実行一時中断中
ARI	パイプキューからの受信中
RUT	パイプキューからの送信中
EXT	実行結果ファイルの転送中

STT　以外の項目の内容については、「4.16　qstatr　リクエストの状態表示」を参照してください。

また、qstatr コマンドはリクエスト ID の代わりにリクエスト名で指定することもできます。リクエスト名で指定するときは -r オプションをつけて実行してください。

```
$ qstatr -r request-name ↵
```

このリクエスト名は qsub コマンドでリクエストを投入するときに設定した名前です。

リクエスト ID がわからないときは、現在登録されているすべてのバッチリクエストの情報を参照します。この場合は、 qstatr コマンドでリクエスト ID を指定せずに -b オプションをつけて実行します。

```
$ qstatr -b ↵
=====================================================
NQS (R11.10) BATCH REQUESTS  HOST: host1
=====================================================
REQUEST ID     NAME      OWNER    QUEUE   PRI NICE STT  PGRP R
------------  -------- -------- -------- ---- --- --- ----- -
72.host1        STDIN     user1    batch1    20  10 RUN  800
73.host1        STDIN     user1    batch1    20  10 QUE      -
74.host1        STDIN     user1    batch2    20  10 RUN      -
-----------------------------------------------------------------
$
```

また、リクエストの詳細情報が欲しい場合は、 qstatr コマンドに -f オプションをつけて実行します。

```
$ qstatr -f 72.host1 ↵
================================================
NQS (R11.10) BATCH REQUEST:  72.host1
================================================
        Name: STDIN                         State: running
        Owner: user1
        Group: group1
        Created: Wed Apr 18 1990          Priority: 31
                  11:39:29                 PGRP: 800
Restricted: Already running
QUEUE
        Name: batch1@host1
RESOURCES LIMITS
    Per_process
        Core File Size =         UNLIMITED  <DEFAULT>
        Data Segment =           UNLIMITED  <DEFAULT>
        Permanent File Size =    UNLIMITED  <DEFAULT>
        Memory Size =            UNLIMITED  <DEFAULT>
        Stack Segment =          UNLIMITED  <DEFAULT>
        CPU Time =               UNLIMITED  <DEFAULT>

    Per_request
        Temporary File Space =  UNLIMITED  <DEFAULT>
        CPU Time Limit =        UNLIMITED  <DEFAULT>
        Process Number Limit =  UNLIMITED  <DEFAULT>
        Physical Memory Limit = UNLIMITED  <DEFAULT>

SCHEDULING PARAMETER
        Nice Value                  0

FILES          LEVEL    MODE     NAME
        Stdout:    0   SPOOL    /home/user1/STDIN.o72
        Stderr:    1   SPOOL    /home/user1/STDIN.e72
MAIL
        Address        user1@host1     When: NONE
MISC
        Restartable    Yes              User Mask: 22
        Restartstate   No               Orig.Owner: user1
        Shell:         DEFAULT
$
```

各項目の内容については、 qstatr(1) の項を参照してください。

リクエストがリモートホストに転送されている場合には、そのリクエストを投入したマシンで qstatr の -t オプションを指定して実行するか、そのリクエストの存在するホストで qstatr(1) コマンドを実行してください。

```
$ qstatr -t 2 72.host1 ↵
```

-tオプションをつけることによって、別のホストに転送されたリクエストを表示することができます。また-tの後の数字は探索レベルを示し、リモートホスト上のリクエストを参照する場合は、1か2を指定します。

なお、リクエストの状態を確認するコマンドには qstat(1) コマンドもあります。qstatr とは使用方法や出力形式が異なります。

qstat コマンドの詳細については「4.13 qstat　JobCenterの状態表示」を参照してください。

3.1.3.2. バッチキューの状態確認

バッチキューの状態を確認するときは、 qstatq(1) コマンドを用います。 qstatqコマンドにはシステム上のバッチキューすべてを対象に情報を表示する機能 (-b オプション) があるのでそれを使用します。

```
$ qstatq -b ↵
============================================================
NQS (R11.10) BATCH  QUEUE SUMMARY  HOST: host1
============================================================
QUEUE NAME       ENA STS PRI RLM   TOT QUE RUN WAI HLD SUS ARR EXT
---------------- ---------------- --------------------------------
batch1           ENA RUN  30   2     3   1   1   1   0   0   0   0
batch2           ENA INA  20   3     2   0   0   1   1   0   0   0
---------------- ---------------- --------------------------------
<TOTAL>                       10     5   1   1   2   1   0   0   0
---------------- ---------------- --------------------------------
$
```

ENAのカラムにはキューの第1特性、STSのカラムにはキューの第2特性、その後には各状態のリクエスト数などが表示されます。キューの第1、第2特性については「2.2.4　キューの状態」を参照してください。

また個々にバッチキューを指定してその状態を参照することもできます。この場合はqstatqコマンドの引き数に参照したいバッチキューのキュー名を指定します。

```
$ qstatq batch1 ↵
============================================================
NQS (R11.10) BATCH  QUEUE SUMMARY  HOST: host1
============================================================
QUEUE NAME       ENA STS PRI RLM   TOT QUE RUN WAI HLD SUS ARR EXT
---------------- ---------------- --------------------------------
batch1           ENA RUN  45   2     3   1   1   1   0   0   0   0
---------------- ---------------- --------------------------------
<TOTAL>                        2     3   1   1   1   0   0   0   0
---------------- ---------------- --------------------------------
$
```

■バッチキューの詳細情報を参照する場合

qstatq コマンドに -f オプションをつけて実行します。

```
$ qstatq -f batch1 ↵
```

```
========================================
NQS (R11.10) BATCH  QUEUE:  batch1@host1
========================================
  Priority:  30                  Status:   [ENABLED,INACTIVE]
  Nice Value:            0
  Scheduling Mode: TYPE-0
  Continuous Scheduling Number: Undefined

ENTRIES
  Total:      3
  Queued:     1     Running:    1     Waiting:    1
  Held:       0     Suspending: 1     Arriving:   0

COMPLEX MEMBERSHIP
  cmplex1, cmplex2

RUN LIMITS
  Total run limit:      5
  User run limit =      3     Group run limit : Unlimited

RESOURCE LIMITS
 Per-process
  Core File Size Limit =        UNLIMITED <DEFAULT>
  Data Size Limit =             UNLIMITED <DEFAULT>
  Permanent File Size Limit =  UNLIMITED <DEFAULT>
  Memory Size Limit =           UNLIMITED <DEFAULT>
  Stack Size Limit =            UNLIMITED <DEFAULT>
  CPU Time Limit =              UNLIMITED <DEFAULT>
 Per-request
  CPU Time Limit =              UNLIMITED <DEFAULT>
  Temporary File Space Limit = UNLIMITED <DEFAULT>
  Process Number Limit =        UNLIMITED <DEFAULT>
  Physical Memory Limit =      UNLIMITED <DEFAULT>

ACCESS
  Route:
  User:         root , userl
  Group:

ATTRIBUTE
  LOADBALANCE   ON
  CLUSTER   OFF

LOAD BALANCING PARAMETER
  Keeping request number limit = 1
  Delivery wait time = 30

OTHER PARAMETERS
  Resource retry wait time = 10
  Resource retry time out = UNLIMITED
  Execution priority (relative value) = 0
  Printer client = NONE

CUMULATIVE TIME
  System space time = 127.30 sec
  User space time =    43.38 sec
```

■リモートホスト上のバッチキューの状態を参照する場合

-h オプションで参照したいホストを指定します。

ホスト host1 からホスト nec1 上のバッチキューの状態を参照する場合の例を以下に示します。

```
$ qstatq -h nec1 -b ↵
```

3.1.3.3. パイプキューの状態確認

パイプキューの状態を確認するときは、バッチキューの状態を参照するときと同じようにqstatq(1) コマンドを用います。

qstatq コマンドにはバッチキュー同様システム上のパイプキューすべてを対象に情報を表示する機能 (-p オプション) があるのでそれを使用します。

```
$ qstatq -p ↵
============================================================
NQS (R11.10) PIPE   QUEUE SUMMARY  HOST: host1
============================================================
QUEUE NAME        ENA  STS  PRI  RLM   TOT  QUE  ROU  WAI  HLD  ARR
--------------- -------------------- ------------------------------
pipe1             ENA  INA  20    1     2    2    0    0    0    0
pipe2             DIS  STP  30    2     0    0    0    0    0    0
netpipe1          ENA  ROT  20    1     2    1    1    0    0    0
--------------- -------------------- ------------------------------
<TOTAL>                         10    4    3    1    0    0    0
--------------- -------------------- ------------------------------
$
```

また、個々にパイプキューを指定してその状態を参照することもできます。その場合はqstatqコマンドの引き数に参照したいパイプキューのキュー名を指定します。

```
$ qstatq pipe1 ↵
============================================================
NQS (R11.10) PIPE   QUEUE SUMMARY  HOST: host1
============================================================
QUEUE NAME       ENA  STS  PRI  RLM   TOT  QUE  ROU  WAI  HLD  ARR
--------------- -------------------- ------------------------------
pipe1            ENA  INA  20    1     2    2    0    0    0    0
--------------- -------------------- ------------------------------
<TOTAL>                         1     2    2    0    0    0    0
--------------- -------------------- ------------------------------
$
```

■パイプキューの詳細情報を参照する場合

qstatq コマンドを -f オプションつきで実行します。

```
$ qstatq -f pipe1 ↵
============================================================================
NQS (R011.1) PIPE   QUEUE: pipe1@host1
============================================================================
Priority:  20                 Status:  [ ENABLE , INACTIVE ]
Queue server: /usr/lib/nqs/pipeclient
ENTRIES
Total:         2
Queued:        2        Routing:         0        Waiting:         0
Held:          0        Arriving:        0
```

```
RUN LIMITS
Total run limit:   3
User run limit :   Unlimited      Group run limit : Unlimited
DESTINATIONS
batch1@host1, batch2@host1
ACCESS
Unrestricted access
ATTRIBUTE
BEFORECHECK   OFF
STAYWAIT      OFF
FREEDESTINATION OFF
LOADBALANCE   ON
TRANSPARENT   OFF
LOAD BALANCING PARAMETER
Reserved run limit =     1
Destination retry wait = 3600
CUMULATIVE TIME
System space time= 1.00 sec
User space time=   2.00 sec
$
```

パイプキューでは特に DESTINATIONS の情報が重要です。ここには転送先キューが表示されます。つまりそのキューにリクエストを投入すると、どのキューに転送されるかが示されます。

なお、パイプキューには転送先のキューが複数設定されている場合もあります。たとえば以下のような場合です。

```
DESTINATIONS
batch1@host1, batch2@host1
```

上記の場合は、まず　　batch1@host1　　キューに転送が試みられ、転送不可能であればbatch2@host1に転送されます。

転送不可能の要因としては転送先のキューがリクエストの受け付けをしていないなどがあります。また、この転送先キューにリモートホストのキューが設定されていたら、そのパイプキューはいわゆるネットワークパイプキューの場合があります。

```
DESTINATIONS
batch1@host2
```

上記の場合、リモートホスト host2 のキュー batch1 への転送を表します。

■リモートホスト上のパイプキューの状態を参照する場合

バッチキュー同様 -h オプションで参照したいホストを指定します。

ホスト　host1　からホスト　nec1　上のパイプキューの状態を参照する場合の例を以下に示します。

```
$ qstatq -h nec1 -p ↵
```

キュー名の指定方法にはバッチキューと同様の規則があります。

3.1.4. バッチリクエストの属性変更

バッチリクエストにはさまざまな属性が付与されていますが、それらの属性はリクエストを投入した後に変更することが可能です。

リクエストの属性変更のために　qalter(1)コマンドが用意されています。qalterコマンドの使用法は、変更したいリクエストと属性値をそれぞれリクエスト ID とオプションで指定します。

たとえば、 72.host1 というリクエスト ID をもつリクエストの、プロセスごとのCPU時間制限値を変更したい場合は次のようにします。

```
$ qalter -lt 1000 72.host1 ↵
```

ただし、リクエストがすでに実行中の場合は変更できる属性に限りがあります。

また、登録されているキューに設定されている資源制限値を越えるような値への変更はできません。もし変更不可能だった場合はエラーメッセージが出力されます。

以下にqalterコマンドで変更可能となっている属性に対するオプションのうち、主なものについて説明します。

■主なオプションとその例

-e

標準エラー出力結果ファイルを変更します。

（例）qalter -e host1:/usr/result.e 72.host1

-lm

プロセスごとのメモリサイズ制限値を変更します。

（例）qalter -lm 2kb 72.host1

-o

標準出力結果ファイルを変更します。

（例）qalter -o host1:/usr/result.o 72.host1

-p

リクエストのプライオリティを変更します。

（例）qalter -p 25 72.host1

-ro

標準出力結果ファイル転送モードを変更します。

（例）qalter -ro s 72.host1

-re

標準エラー出力結果ファイル転送モードを変更します。

（例）qalter -re n 72.host1

-nr

リクエスト再実行可不可モードを変更します。

（例）qalter -nr on 72.host1

-mb

リクエスト実行開始時のメール送信モードを変更します。

（例）qalter -mb on 72.host1

-me

リクエスト実行終了時のメール送信モードを変更します。

（例）qalter -me on 72.host1

-mu

メール送信相手を変更します。

（例）qalter -mu user2 72.host1

-s

リクエスト実行シェルを変更します。

（例）qalter -s /bin/csh 72.host1

リクエストがバッチキューに存在する場合、そのシステムがサポートしていない資源制限値に関しては、値を変更することはできません。また、リクエストがパイプキュー上で転送中の場合とバッチキュー上で実行中の場合は、変更できない属性があります。

3.1.5. バッチリクエストの削除

バッチリクエストの削除は qdel(1)コマンドで行います。まだ実行されていないリクエスト、つまり実行待ち (queued)、実行遅延 (waiting)、ホールド (holding)状態のリクエストを削除する場合は、そのリクエスト ID を指定して qdel コマンドを実行します。

```
$ qdel 72.host1 ↵
Request 72.host1 has been deleted.
$
```

削除が正しく行われると、削除された旨を知らせるメッセージが出力されます。このときそのリクエストがすでに実行中であると、以下のように現在実行中である旨を知らせるメッセージが出力されてリクエストは削除されません。実行中のリクエストの削除の仕方については後ほど説明します。

```
$ qdel 73.host1 ↵
Request 73.host1 is running.
$
```

なお、リモートホスト上のリクエストを削除したい場合は、そのリクエストを投入したホスト、またはそのリクエストの存在しているホストでコマンドを実行してください。

qdel　コマンドはリクエストをリクエスト名で指定することもできます。リクエスト名で指定するときは -r オプションを付けてください。

```
$ qdel -r MAKE3 ↵
```

次に、削除したいリクエストが実行中の場合について説明します。

リクエストが実行中の場合は前期のような方法では削除できません。実行中のリクエストを削除するには、 -k オプションを指定して qdel コマンドを実行します。リクエストの指定方法は上記で説明したものと変わりません。

```
$ qdel -k 74.host1 ↵
Request 74.host1 is running, and has been signalled.
```

```
$
```

このような指定をするとリクエストに対して SIGKILL シグナルが送信され、リクエストの実行が強制的に終了させられますが、 SIGKILL シグナル以外のシグナルを送信したいときは、以下のように送信したいシグナル番号を明示的に指定します。

[SIGINTシグナルを送信したい場合]

```
$ qdel -2 72.host1 ↵
```

[SIGHUPシグナルを送信したい場合]

```
$ qdel -1 72.host1 ↵
```

つまり-kと-9は機能的にまったく同じということになります。また-kを指定して実行中ではないリクエストを引き数に指定しても、そのリクエストは正常に削除されます。したがって、実行中のリクエストと実行中ではないリクエストを一度に削除することもできます。

[72.host1 が実行中で 73.host1 が実行中ではない場合]

```
$ qdel -k 72.host1 73.host1 -p ↵
```

なお、結果ファイル転送中のバッチリクエストを指定することにより、そのリクエストを親とするネットワークリクエストを削除することができます。

3.1.6. バッチリクエストの保留／保留解除

バッチリクエストの保留/保留解除はそれぞれqhold(1)、qrls(1)コマンドで行います。リクエストの保留は実行されていないリクエスト、つまり実行待ち（queued)、実行遅延（waiting）状態に限り有効です。

バッチリクエストを保留する(保留状態にする)と、そのリクエストはリクエストの実行スケジューリングの対象から外されます。保留されたリクエストは保留状態を解除されない限り実行されません。

保留状態を解除するとリクエストを保留する前の状態に戻されます。つまり、保留する前の状態が実行待ち（queued）であればその状態に、実行遅延(waiting）状態であればその状態に戻されます。

リクエストを保留する場合は、そのリクエストIDを指定してqholdコマンドを実行します。

```
$ qhold 72.host1 ↵
Request 72.host1 has been held.
$
```

保留が正しく行われると、保留された旨を知らせるメッセージが出力されます。また、何らかの理由でホールドできなかった場合は、その原因を示すエラーメッセージが出力されます。

保留状態を解除する場合は、そのリクエスト ID を指定して qrls コマンドを実行します。

```
$ qrls 72.host1 ↵
Request 72.host1 has been released.
$
```

保留解除が正しく行われると、保留が解除された旨を知らせるメッセージが出力されます。

リクエストが保留中ではなかったなどの理由で保留の解除ができなかった場合は、それに対応するエラーメッセージが出力されます。

```
$ qrls 73.host1 ↵
Request 73.host1 is not holding.
```

```
$
```

qhold、 qrls コマンドは qdel コマンドと同様に、リクエストをリクエスト名で指定したり、リモートマシン上のリクエストを保留、保留解除することができます。

3.1.7. バッチリクエストの一時停止／再開

この項ではバッチリクエストの一時停止 / 再開について説明します。

バッチリクエストの一時停止 / 再開はそれぞれ qspnd(1)、 qrsm(1) コマンドで行います。リクエストの一時停止は実行されているリクエスト、つまり実行中 (running) 状態に限り有効です。

バッチリクエスト一時停止する (suspend 状態にする) と、そのリクエストのプロセスには一時停止シグナル(SIGSTOP) が送信されます。また、 一時停止状態を解除する (running 状態に戻す) と、再開シグナル (SIGCONT) が送信されます。

リクエストを一時停止する場合は、そのリクエスト ID を指定して qspnd コマンドを実行します。

```
$ qspnd 72.host1 ↵
Request 72.host1 has been suspended.
$
```

一時停止が正しく行われると、一時停止された旨を知らせるメッセージが出力されます。

一時停止状態を解除する場合は、そのリクエスト ID を指定して qrsm コマンドを実行します。

```
$ qrsm 72.host1 ↵
Request 72.host1 has been resumed.
$
```

一時停止解除が正しく行われると、その旨を知らせるメッセージが出力されます。リクエストが一時停止中ではなかったなどの理由で一時停止の解除ができなかった場合は、それに対応するエラーメッセージが出力されます。

```
$ qrsm 73.host1 ↵
Request 73.host1 is not suspending.
$
```

qspnd、 qrsm コマンドは qdel コマンドと同様に、リクエストをリクエスト名で指定したり、リモートマシン上のリクエストを一時停止、解除することができます。

3.1.8. バッチリクエストの再登録

バッチリクエストの再登録はqrerun(1) コマンドで行います。リクエストの再登録は実行中 (running) のリクエストに限り有効です。バッチリクエストを再登録すると、そのリクエストの実行が中止され登録されているキューに投入し直されます。ただし、リクエスト ID は以前に付けられていたものが引き継がれます。

この再登録処理の過程で、リクエスト投入時と同様にキューの資源制限値とリクエスト属性の資源制限値の比較が行われます。もし、この比較で投入不可と診断されると再登録は行われず、そのままリクエストの実行が続けられます。

リクエストを再登録する場合は、そのリクエスト ID を指定して qrerun コマンドを実行します。

```
$ qrerun 72.host1 ↵
Request 72.host1 has been rerun.
$
```

再登録が正しく行われると、再登録された旨を知らせるメッセージが出力されます。リクエストが実行中でなかったなどの理由で再登録されなかった場合は、それに対応するエラーメッセージが出力されます。

```
$ qrerun 73.host1 ↵
Request 73.host1 is not running.
$
```

qrerun コマンドは qdel コマンドと同様に、リクエストをリクエスト名で指定したり、リモートマシン上のリクエストを再登録することができます。

3.1.9. バッチリクエストの移動

バッチリクエストの移動はqmove(1)コマンドで行います。リクエストの移動は実行されていないリクエスト、つまり実行待ち(queued) 実行遅延 (waiting)、ホールド (holding) 状態のリクエストにのみ有効です。

リクエストの移動とは、リクエストを現在登録されているバッチキュー以外のバッチキューに登録しなおすことです。したがって移動処理の過程でリクエスト投入時と同様に、移動先のキューの資源制限値と移動されるリクエストの資源制限属性の比較が行われます。もし、この比較で移動先のキューに投入不可と診断されると移動は行われません。

またリクエストの移動形態としては、キュー単位・リクエスト単位の移動が用意されています。キュー単位の移動とは特定のキューに登録されているリクエストすべて　　(実行中のものを除く）を一度に移動する形態で、リクエスト単位の移動とは文字どおり個々のリクエストを移動させることになります。

いずれの場合にも、ほかのユーザのリクエストを移動することはできません。したがって、キュー単位の移動を行う場合は、対象のキューに登録されている自分のリクエストに限り移動されます。

リクエスト単位の移動を行うには、リクエスト ID を指定して qmove コマンドを実行します。

キュー単位の移動を行いたい場合は、 -q オプションで移動元のキューを指定します。

```
$ qmove 72.host1 batch1 ↵
Request 72.host1 has been moved.
$
$ qmove -q batch1 batch2 ↵
Request 72.host1 has been moved.
Request 73.host1 has been moved.
Request 74.host1 has been moved.
$
```

移動が正しく行われると、移動された旨を知らせるメッセージが出力されます。このとき、そのリクエストが移動できない場合は、それに対応する理由を示すメッセージが出力されます。

```
$ qmove 73.host1 batch1 ↵
Request 73.host1 is running.
$
```

qmove コマンドは qdel コマンドと同様に、リクエストをリクエスト名で指定することができます。

3.1.10. バッチリクエストに対するメッセージ送信

実行中バッチリクエストに対してメッセージを送信する場合は、 qmsg（1）コマンドを利用します。送信したメッセージはリクエストの結果ファイルに埋め込まれます。以下にメッセージ送信の手順を示します。

[72.host1 という ID のついたリクエストに対しメッセージを送信する場合]

```
$ qmsg 72.host1 ↵
System shutdown.    （メッセージは標準入力から読み取られます。）
```

```
CTRL-D(EOF)
$
```

qmsgコマンドをオプションの指定なしで実行すると標準出力結果ファイル、標準エラー出力結果ファイルのどちらにもメッセージが送信されます。

なお、標準出力結果ファイルのみ、または標準エラー出力用結果ファイルのみにメッセージを送信したい場合は、それぞれ -o、 -e オプションを指定してください。

```
$ qmsg -o 72.host1 ↵
  (標準出力結果ファイルにメッセージを書き込む)
CTRL-D(EOF)
$ qmsg -e 72.host1 ↵
  (標準エラー出力結果ファイルにメッセージを書き込む)
CTRL-D(EOF)
$
```

qmsg コマンドは qdel コマンドと同様に、リクエストをリクエスト名で指定することができます。

3.1.11. 有効資源制限の確認

有効資源制限とは、OSでサポートされているプロセス実行時の資源制限の機能を利用します。

リクエストの投入時は、さまざまな資源制限値をリクエストの属性としてつけることができますが、リクエストを実行するホストでその資源制限をサポートしていないと、その指定した制限は無視されます。

JobCenterで利用できる有効資源制限の確認は qlimit(1) コマンドで行います。

ローカルホストの有効資源制限を確認する場合は、以下のように引き数をなにも指定しないで qlimit コマンドを実行します。

```
$ qlimit ↵
Data segment size limit (-ld)
Per-process permanent file size limit (-lf)
Per-process memory size limit (-lm)
Stack segment size limit (-ls)
Per-process cpu time limit (-lt)
 :
Nice value (-ln)
Shell strategy = LOGIN
$
```

qlimit コマンドは、そのローカルホストで有効な資源制限の種類と qsub コマンドで指定する場合のオプションを出力し、最後に管理者によって設定されているシェル選択方式を出力します。

またリモートホストの有効資源制限を確認する場合は、以下のように引き数に参照したいホストの名前を指定します (実行結果は、実行したシステムによって異なります)。

```
$ qlimit host2 ↵
Core file size limit (-lc)
Data segment size limit (-ld)
Per-process permanent file size limit (-lf)
Per-process memory size limit (-lm)
Stack segment size limit (-ls)
Per-process cpu time limit (-lt)
 :
Shell strategy = FIXED
$
```

3.1.12. バッチリクエストの終了

バッチリクエストの終了を確認するには、 qstatr コマンドなどで確認します。また、 qwait(1) コマンドで終了を待ち合わせ、終了状態を知ることができます。

```
$ qwait 123.host1 ↵
done 45              (リクエストは終了コード 45 で終了した。)
$
```

```
$ qwait 124.host1 ↵
killed 9             (リクエストは SIGKILL によって終了した。)
$
```

また、リクエスト終了時に送信されてくるメールで判断することもできます。

リクエスト終了時にメールを送信するようにするには、リクエスト投入時に qsub コマンドの-me オプションを指定していなければなりません。ただし何らかの障害が発生してリクエストが破棄された場合は、必ずその障害状況を報告するメールが送信されます。

3.1.13. バッチリクエストの出力ファイル

バッチリクエストが終了すると、そのリクエストの実行結果ファイルが得られます。この結果ファイルは通常 2 つ得られます。

1つはスクリプト実行時に標準出力に出力された内容が格納されたファイルであり、もう1つは標準エラー出力に出力された内容が格納されたファイルです。

これらのファイルはリクエスト投入時に指定することもできますし（「3.1.2 バッチリクエストの投入」参照)、投入後に変更することもできます（「3.1.4 バッチリクエストの属性変更」参照)。

もし結果ファイルの指定を省略した場合は、以下のような規則で結果ファイルが作成されます。

ファイル種別	命名規則
標準出力用結果ファイル	リクエスト名.o<リクエスト連番> 例) リクエスト名が batreq でリクエスト連番が 72 の場合、batreq.o72 が標準出力用結果ファイルになります。
標準エラー出力用結果ファイル	リクエスト名.e<リクエスト連番 例) リクエスト名が batreq でリクエスト連番が 72 の場合、batreq.e72 が標準エラー出力用結果ファイルになります。

なお、リクエスト名はリクエスト投入時に指定できますが、もし指定しなかった場合はスクリプトファイル名がリクエスト名になります。

標準入力からスクリプトを入力した場合は"STDIN" という名前になります。

3.2. ネットワークリクエストの操作方法

ネットワークリクエストは、バッチリクエスト終了時に JobCenter によって自動的に作成・投入されるリクエストです。したがって、ユーザが独自に作成することはできません。

ここでは、ネットワークリクエストが投入されたあとに状態確認・移動・削除する方法について説明します。

本機能はWindows版および現在のバージョンのUNIX版ではサポートしない機能となります。

3.2.1. ネットワークリクエストに関するJobCenterの状態確認

ネットワークリクエストに関するJobCenter の状態としては、ネットワークリクエストの状態およびネットワークキューの状態があります。ここでは、この 2 つの状態の確認方法について説明します。

3.2.1.1. ネットワークリクエストの状態確認

ネットワークリクエストの状態を確認するには、バッチリクエストと同様に qstat(1) や qstatr(1) を使用します。使用方法もまったく同じです。説明は「3.1.3.1 バッチリクエストの状態確認」を参照してください。

ここでは使用例の一部を示します。

■例 1 qstat(1) コマンドによる特定のネットワークキューの状態確認

```
$ qstat net1 ↵
net1@host1;  type=NETWORK;  [ENABLED, STOPPED];  pri=5
  0 run;  2 queued;  0 wait;  0 hold;  0 arrive;
        REQUEST NAME        REQUEST ID        USER   PRI    STATE     PGRP
    1:          STDIN     395.host1(#31)      user1  31     QUEUED
    2:          STDIN     395.host1(#30)      user1  31     QUEUED
$
```

■例 2 qstat(1) コマンドによる特定のネットワークキューの詳細表示

```
$ qstat -l net1 ↵
net1@host1;  type=NETWORK;  [ENABLED, STOPPED];  pri=5
  0 run;  2 queued;  0 wait;  0 hold;  0 arrive;
  Request    1:  Name=STDIN  Id=395.host1  Event#=31
                 Owner=user1    Priority=31  QUEUED
  Created at Mon Mar 28 13:50:55 JST 1994
  Mail address = user1@host1
  Owner user name at originating machine = user1
  Staging-out file name = host1:/home/nqs/STDIN.e395
  Request    2:  Name=STDIN  Id=395.host1  Event#=30
                 Owner=user1     Priority=31  QUEUED
  Created at Mon Mar 28 13:50:55 JST 1994
  Mail address = user1@host1
  Owner user name at originating machine = user1
  Staging-out file name = host1:/home/nqs/STDIN.o395
$
```

■例 3 qstatr(1) コマンドによるネットワークキューの状態確認

```
$ qstatr -N ↵
```

```
================================================
NQS (R11.10) NETWORK REQUEST  HOST: host1
================================================
REQUEST ID        EVENT   NAME     OWNER    QUEUE NAME     PRI STT  PGRP
--------------- ------- -------- -------- --------------- ---- --- ------
396.host1        31(ERR) STDIN    user1    net1             31 QUE
396.host1        30(OUT) STDIN    user1    net1             31 QUE
------------------------------------------------------------------------
$
```

■例 4 qstatr(1) コマンドによる特定のネットワークキューの詳細表示

```
$ qstatr -N ↵
================================================
NQS (R11.10) NETWORK REQUEST: 395.host1
================================================
        Name: STDIN                       State: queued
        Owner: user1                      Priority: 31
        Group: group1                     Event: 30
        Created: Mon Mar 28 1994
                 13:50:55
QUEUE
        Name: net1@host1
STAGING FILE
        Name: /home/nqs/STDIN.o395
MAIL
        Address:         user1@host1
MISC
        Orig.Owner:     user1
$
```

3.2.1.2. ネットワークキューの状態確認

ネットワークキューの状態に関しても、バッチキューと同じく qstat(1) もしくは qstatq(1) を用いて参照します。説明は「3.1.3.2 バッチキューの状態確認」を参照してください。 ここでは使用例の一部を示します。

■例 1 qstat(1) コマンドによる特定のネットワークキューの詳細状態確認

```
$ qstat -x net2 ↵
net2@host1;  type=NETWORK;  [ENABLED, STOPPED];  pri=40
   0 run;  0 queued;  0 wait;  0 hold;  0 arrive;
   Run_limit = 2;
   Cumulative system space time = 0.00 seconds
   Cumulative user space time = 0.00 seconds
   Queue server: /usr/lib/nqs/netclient
   Destination Machine = host2
$
```

■例 2 qstatq(1) コマンドによるネットワークキューの状態確認

```
$ qstatq -N ↵
======================================================
NQS (R11.10) NETWORK QUEUE SUMMARY    HOST: host1
======================================================
QUEUE NAME        DESTINATION MACHINE   ENA STS PRI RLM   TOT QUE RUN WAI
--------------- -------------------- ----------------- -----------------
DefaultNetQue             -             ENA INA  -1  20    0   0   0   0
net1              host1                 ENA INA  20   1    4   0   1   3
```

```
net2              host2                   ENA STP  40   2    0   0   0   0
--------------- -------------------- ---------------- ----------------
<TOTAL>                                            20    4   0   1   3
--------------- -------------------- ---------------- ----------------
$
```

■例 3 qstatq(1) コマンドによる特定のネットワークキューの詳細状態表示

```
$ qstatq -f net1 ↵
============================================================
NQS (R11.10) NETWORK QUEUE: net1@host1
============================================================
   Priority: 20                Status:  [ENABLED  , INACTIVE]
   Queue server: /usr/lib/nqs/netclient
ENTRIES
    Total:      4
    Queued:     0     Running:    1     Waiting:    3
RUN LIMITS
  Total run limit:   1
DESTINATIONS MACHINE (MID)
   host1 (100)
CUMULATIVE TIME
   System space time = 5.20 seconds
   User space time =   0.85 seconds
$
```

3.2.2. ネットワークリクエストの移動

qmove(1)において、リクエスト結果ファイル転送中(exiting)のバッチリクエストとネットワークキューを指定することによって、指定したバッチリクエストの結果ファイル転送のために作成されたネットワークリクエスト全部を、指定したネットワークキューに移動させることができます。

ネットワークリクエストを移動させることにより、結果ファイルを転送するホストを変更することが可能となります。このとき新しく結果ファイルを転送することになったホスト上に、あらかじめバッチリクエストに設定されている結果ファイル転送先のパスと同じパスがなければ、結果ファイル転送に失敗するので注意が必要です。

なお、-qオプションを使ってネットワークキュー内のリクエストをすべて移動させたり、ネットワークリクエストを直接指定して移動させたりすることはできません。

3.2.3. ネットワークリクエストの削除

qdel(1)において、リクエスト結果ファイル転送中(exiting)のバッチリクエストを指定することによって、指定したバッチリクエストの結果ファイル転送のために作成されたネットワークリクエスト全部を削除することができます。

-k オプションや -signo オプションは必要ありません。このとき対象となるネットワークリクエストが結果ファイル転送中(running)だった場合、その結果ファイルはバッチリクエストを実行したマシン上のリクエスト所有者のホームディレクトリに置かれます。

なお、ネットワークリクエストを直接指定することはできません。

3.3. ジョブステップリスタート機能

3.3.1. ジョブステップリスタート機能の概要

ジョブステップリスタート機能は、ジョブのシェルスクリプト内にリクエストの実行状態を保存する記述を追加することで、障害などにより中断したリクエストが再実行される際に、最後に保存した実行状態からリクエストの再実行を行う機能です。

リクエストの情報を採取し、再実行時に実行を開始する箇所を以後チェックポイントと呼びます。

なお本機能を用いることができるシェルは、 Bourne-Shell 系のシェル (/usr/bin/sh や /usr/bin/ksh など) またはC-Shell 系のシェル (/usr/bin/csh など) です。本機能はUNIXのみサポートとなります。

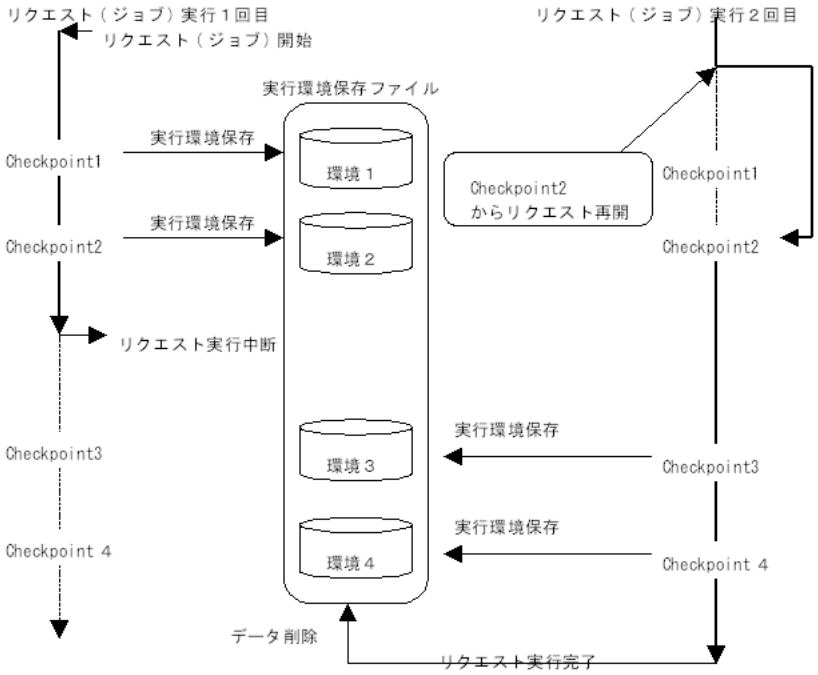


図3.2 ジョブステップリスタート機能の使用イメージ

ジョブステップリスタート機能の動作概要は以下のとおりです。

1. ジョブスクリプト実行前にチェックポイントの記述を検索し、チェックポイント行として記述されたコメント行を、リクエストの状態を採取・保存するコマンドに置換します。

2. ジョブスクリプトが実行を開始すると、チェックポイント行を通過した時点で、その時のリクエストの環境変数、シェル変数および通過したチェックポイント名を内部の実行環境保存ファイルに記録します。

3. リクエストが再実行されると、通過済のチェックポイントを実行環境保存ファイルから検索し、最後に通過したチェックポイントを特定します。

4. 保存した変数を復元し、必要なセットアップを行う関数やスクリプトを呼び出すための命令を追加します。また、最後に通過したチェックポイントから実行を再開するようにシェルを編集します。

   この編集操作は shell の種類によって異なります。

3.3.1.1. 対象シェルが Bourne-Shell 系の場合

■最初のチェックポイントの記述が現れるまで、シェルスクリプトを展開します。展開後、前回実行時に最後に通過したチェックポイントで保存された環境変数、およびシェル変数のデータを復元するスクリプトを、最初のチェックポイントの記述の位置に挿入します。

■前回実行時に最後に通過したチェックポイントから実行を再開するために、最初に出現したチェックポイントから最後に通過したチェックポイントまでのスクリプト記述を削除します。

■動作概要の1.と同様に、残ったチェックポイントの記述をコマンドに置換します。

3.3.1.2. 対象シェルが C-Shell 系の場合

■動作概要1.と同様に、チェックポイントの記述をコマンドに置換します。その時最後に通過したチェックポイントの次の行にチェックポイント名のラベルを挿入します。また、ジョブスクリプトの先頭に以下の環境復元用スクリプトへジャンプする記述を挿入します。

■前回実行時、最後に通過したチェックポイントで保存された環境変数、およびシェル変数のデータを復元するスクリプトを、ジョブスクリプトの後ろに追加します。

■上記で追加したスクリプトの後ろに、ユーザが指定したセットアップ関数、またはファイルを実行する記述を追加し、最後にラベルとして記述されたチェックポイント名にジャンプする命令を追加します。

以上のスクリプトの書き換えは自動的に行われるため、ユーザが意識する必要はありません。チェックポイントからの実行はシェルの編集、または goto 文によって行われるため、各種構文(if や for など) の内部へジャンプするような箇所にチェックポイントを記述することはできません。

本機能を使用することで、リクエストの実行状態によってその実行をユーザが意図的に中断し、ジョブスクリプトの途中から再実行を行うなどのジョブスクリプトの記述が可能となります。リクエスト実行中に採取された実行環境のデータは、リクエストの実行が完了した時点で自動的に削除されます。

ジョブステップリスタート機能を使用する場合、再実行されるリクエストは同一のリクエストでなければなりません。リクエストの実行が完了したものや、異常終了したものについては再実行を行うことはできません。

3.3.2. 保存される実行状態

ジョブステップリスタート機能は、 Bourne-shell(sh, bsh) および、 C-shell(csh) の実行状態を各チェックポイントが通過するタイミングで保存します。チェックポイントではリクエストのプロセス状態そのものは保存しません。チェックポイントで保存されるリクエストの実行状態は以下のとおりです。

■一部の環境変数を除くすべての環境変数

■一部のシェル変数を除くすべてのシェル変数

■-var、-varallオプションによって指定されたシェル変数

■再実行時に実行を開始する位置情報

保存されない環境変数、およびシェル変数については「4.19 # NScheck チェックポイントの設定」を参照してください。

3.3.3. チェックポイントの指定

ジョブステップリスタートの機能を使用するためにはあらかじめジョブスクリプト内に、チェックポイントの記述を行う必要があります。

チェックポイントはコメント行として記述します。このコメント行の記述には2通りの記述方法があり、リクエストの実行時に記述行がそのままチェックポイントとして使用されるものと、リクエスト全体のチェックポイントに関する属性を記述し、その行自体はチェックポイントとして動作しないものとがあります。

チェックポイントとして動作するコメント行の記述には必ずチェックポイント名を記述しなければなりません。チェックポイント名が記述されていない場合、その行はリクエスト全体に対する設定として見なされるか、無効な記述として無視されます。

リクエスト全体に対する設定として見なされるのは実行シェルタイプの指定とリクエスト全体で特別に保存するシェル変数を指定する記述のみです。それ以外の記述がなされた場合、その行は無効なコメント行として無視されます。

チェックポイント名は同一ジョブスクリプト内で一意でなければなりません。同一の名前をもつチェックポイントが複数存在する場合、それらの中で最初に現れた記述のみが有効となります。

チェックポイント関連のコメント行で記述する内容には以下のものがあります。

3.3.3.1. 形式

```
# NScheck [-c] [-varall $save_variables ... ]
# NScheck $checkpoint_name [-f $setup_file | $setup_function] [-var
 $save_variables ...]
```

3.3.3.2. 引数・オプション

-c

スクリプトの実行シェルとして csh 系のシェルを使用する場合、最初のチェックポイントの記述の前に指定します。

本オプションが指定されなかった場合、シェルスクリプトは sh（Bourne-Shell）系のシェルで実行されるものとして扱われます。

$checkpoint_name

チェックポイント名です。リクエストの再実行時のラベルにもなります。リクエスト内で一意となるように指定します。

$setup_function

リクエスト再実行時に環境変数とシェル変数を復元した後、各種設定を行うユーザ指定の関数です。

sh系のシェルではジョブスクリプト内にシェル記述された関数定義を指定することが可能です。関数定義の記述は最初のチェックポイントよりも前方で行ってください。

-f $setup_file

リクエスト再実行時に環境変数とシェル変数を復元した後ジョブスクリプトから読み込まれ、各種設定を行うユーザ指定のスクリプトファイルです。

本オプションを使用する場合、リクエストが再実行される実行マシン上に指定したファイルが存在しなければなりません。ファイルが存在しない場合、リクエストはエラーで終了します。

-var $save_variables

自動的に保存されない特別なシェル変数を保存します。設定はチェックポイントごとに行います。

-varall $save_variables 自動的に保存されない特別なシェル変数を保存します。設定はこの記述以降のすべてのチェックポイントで有効となります。

チェックポイントの記述は、 if、 for、 while、 switch などの構造文の中で使用することはできません。そのような中でチェックポイントを指定した場合には、リクエストの再実行時にそのチェックポイントから再開された時点でエラーとなります。

以下の節では、チェックポイントを記述する場合に必要となる事項を説明していきます。ただし、ジョブスクリプトとして使用するシェルスクリプトの詳しい記述方法については使用するシェルのリファレンスマニュアルを参照してください。

3.3.4. スクリプトの記述例

ジョブステップリスタート機能を使用するジョブスクリプトの作成について例を挙げて説明します。

以下に示すスクリプトは、チェックポイント A、チェックポイント B、チェックポイント C を順番に通過するだけの単純な例です。この例では各チェックポイントで特にチェックポイント名以外の指定をしません。

Bourne shell 系のシェルを使用する場合

```
func() ...
#NScheck A
echo PASS checkpoint A

#NScheck B
echo PASS checkpoint B
sleep 60

#NScheck C
echo PASS checkpoint C
```

C chell 系のシェルを使用する場合

```
#NScheck -c
#NScheck A
echo PASS checkpoint A

#NScheck B
echo PASS checkpoint B
sleep 60

#NScheck C
echo PASS checkpoint C
```

スクリプトの実行シェルとして bsh 系のシェルを使用する場合、スクリプト内で使用する関数の定義は最初に登場するチェックポイント記述よりも以前に行う必要があります。

逆に、ジョブ本体の記述は最初のチェックポイント以降に記述してください。

スクリプトの実行シェルとして csh 系のシェルを使用する場合、最初のチェックポイントの記述の前にシェルタイプの指定を行う # NScheck -c のコメント行を記述する必要があります。このコメント行はリクエスト全体に対する設定として見なされ、この行自体はチェックポイントとしては扱われません。この指定が存在しない場合、シェルスクリプトは bsh 系のシェルで実行されるものと見なされます。この指定が正しく行われなかった場合、リクエストは正常に再実行されません。

3.3.5. スクリプトのテスト

ジョブスクリプトの作成が完了したら、次に作成したジョブスクリプトのチェックポイントの記述が正しいかどうかをテストします。テストには nscpp (1) コマンドを使用します。nscpp コマンドはスクリプト内に記述されたチェックポイント行の記述をテストします。チェックポイントの記述に問題がなければ nscpp は何も表示せずに終了します。nscppの使用方法とテストする内容については「4.20 nscpp チェックポイントの設定のテスト」を参照してください。

3.3.6. スクリプトを実行する

では、この記述したジョブスクリプトを実行してみましょう。

ジョブスクリプトを実行するためには、 qsub コマンドにより、スクリプトファイルを投入します。

以下の例ではスクリプトファイル名を script1 とし、リクエストを実行するバッチキューを batch1 としています。リクエスト投入の際に -sr オプションを付け忘れないようにしてください。 -sr オプションを指定せずにリクエストを投入した場合には、チェックポイントの記述は単なるコメント行として解釈されます。

リクエスト実行中にシステムの停止などが発生しなければ、リクエストは問題なく実行され、終了します。

```
$ qsub -q batch1 -sr script1 ↵
```

3.3.7. リクエストの再実行

リクエストが正常、異常にかかわらず終了した場合、システムの停止などが発生しなければ、チェックポイントで採取されたリクエストの実行情報は自動的に削除されます。

上記で投入したリクエストが実行終了せずに再実行された場合は、チェックポイントからの再実行となります。

リクエストが終了せずに再実行されるのは以下の場合です。

■リクエストの実行中に nqsdaemon がシャットダウンした場合

■リクエストの実行中に qrerun で再実行された場合

投入されたリクエストが上記の状態で再実行された場合、ジョブステップリスタート機能を使用する設定であれば、リクエストは最後に通過したチェックポイントから再実行されます。

前述の例のスクリプトで、チェックポイント B を通過後の sleep 中に、 qrerun を実行して再実行した場合、リクエストはチェックポイントBから再実行されることになります。

チェックポイントから再実行されたリクエスト内では、環境変数 NQS_RESTART_TIME にリクエストが再実行された日時が記録され、最後に通過したチェックポイント名が環境変数 NQS_RESTART_CKPOINT に記録されます。

この環境変数が設定されているかどうかを調べることで、リクエストがチェックポイントから再実行されたかどうかを判定することが可能です。

3.3.8. リクエスト実行の中断

なんらかの原因でリクエストの実行に問題が生じた時、そのリクエストの実行を qrerun(1)コマンドを使用して停止することができます。

リクエストを qrerun で再実行した場合、通常は qrerun の実行後、ただちにスケジューリングが開始されます。

再実行前に障害原因を取り除く必要がある場合は、リクエストのスケジューリングを保留する必要があります。この場合qrerun に -hold オプションを使用することで、リクエストはホールド状態でバッチキュー上にとどまります。ホールド状態になったリクエストは qrls（1）の実行によりスケジューリングを開始します。

また、一定時間を待ちあわせてから自動的に再投入したい場合は、qrerun の -a オプションを使用してリクエストの再実行を行ってください。

3.3.9. ジョブステップリスタート機能の利用例

ここでは、ジョブステップリスタートの機能を使用する場合に必要な記述と、その使用方法を紹介します。

3.3.9.1. リクエスト再実行時設定

リクエストの再実行を行う際に、不要なファイルを削除するなどの特定の処理が必要となる場合があります。それらのセットアップ処理をチェックポイントに記述することで、リクエストが実行を再開する前に、その処理を行うことができます。

セットアップ処理の内容は、シェルスクリプト内に関数定義として記述しておく方法と、外部ファイルとして用意し、実行再開前にジョブスクリプトから読み込んで実行させる方法とが選択できます。

ただし、スクリプトとして準備する場合には、あらかじめリクエストが実行されるマシン上に、それらのファイルを用意しておく必要があります。

負荷分散機能などを使用しているため、リクエストが実行されるマシンを事前に特定できない場合には、リクエストが実行される可能性のあるすべてのマシン上に、同じパスでそれらのファイルが読み込めるように設定しておく必要があります。

以下に設定例を示します。

```
    :
    :
< 処理 1 >
echo Phase1 complete.
# NScheck Complete1 -f /home/private/jobsetup1.sh

< 処理 2 >
# NScheck Complete2 "jobsetup2 $JPSTAT1 $JPSTAT2"
    :
    :
```

まず、リクエストが＜処理 1＞を終了後、＜処理 2＞の実行中に、外部からqrerun を実行して実行を一時的に停止した場合を想定します。

リクエストは Complete1 のチェックポイントからリクエストを再開する前に、 /home/private/jobsetup1.sh で指定されたスクリプトを読み込み、実行した後、リクエストの実行を行います。

-f　オプションを使用して設定ファイルを使用する場合には、設定ファイルの記述は、リクエストを実行するシェルで実行可能なスクリプトでなければなりません。したがって、 C Shell で記述されたスクリプトの再実行時設定ファイルとして Bourne Shell のスクリプトファイルを指定したり、その逆に Bourne Shell のスクリプトの設定ファイルとして C Shell のスクリプトファイルを指定することはできません。

次に＜処理 2＞が完了し、 Complete2 のチェックポイントを通過後再び再実行された場合を想定します。

上記の＜処理　2＞の次の行のチェックポイントでは再実行時のセットアップを外部ファイルでなく、関数呼び出しで行っています。呼び出される関数に引数を与えたい場合には、上記の例のように関数と引数を「"」でくくって記述します。

3.3.9.2. 明示的に保存するシェル変数

通常チェックポイントでは、一部を除いてシェル変数と環境変数は自動的に保存されますが、自動的に保存されないシェル変数を強制的に保存したい場合、あるいは変数の内容に改行、空白文字などを含むものを保存したい場合には、明示的に変数名を指定して保存する必要があります。

指定の方法にはリクエスト全体で共通して保存する形式と、チェックポイントごとに指定する方法があります。

次の例では、シェル変数 IFS(Bourne Shellではフィールド分離文字の指定) を Complete3 のチェックポイントのみで保存する場合の例を示します。

```
    :
    :
< 処理 3 >
echo Phase3 complete.
# NScheck Complete3 -var IFS

< 処理 4 >
# NScheck Complete4
    :
    :
```

シェル変数 IFS をリクエスト内のすべてのチェックポイントで保存したい場合には、最初のチェックポイントの記述が現れる前に、 # NScheck -varall オプションを使用して 、保存するシェル変数名を指定します。

-varall オプションは -c オプションなどと同時に使用できます。詳細については「4.19 # NScheck チェックポイントの設定」を参照してください。

また、保存されない環境変数については明示的に保存するオプションはありません。保存されない環境変数を明示的に保存したい場合には一度シェル変数に保存したい環境変数の値を代入し、シェル変数として復元後、セットアップ関数、またはセットアップファイルなどで環境変数として設定しなおすことで復元することができます。

なお、上記の保存されない環境変数やシェル変数には、シェルの起動時に自動的に設定されるものが含まれており、リクエスト実行中に特に設定しない場合にはデフォルトの値が入るため、通常それらを明示的に保存する必要はありません。

第4章 JobCenter ユーザコマンド一覧

JobCenterユーザコマンドについて説明します。

なおqmgrコマンドとnmapmgrコマンドにつきましては<コマンドリファレンス>の「3.12 nmapmgr ネットワークの構成管理」および<コマンドリファレンス>の「3.13 qmgr 構成管理および運用管理」を参照してください。

4.1. qalter　バッチリクエストの属性変更

```
/usr/bin/qalter [ -h $host-name ] $alt-option $request-id

/usr/bin/qalter [ -h $host-name ] $alt-option -r request-name
```

4.1.1. 機能説明

qalter は、バッチリクエストの資源制限、実行環境などの属性を変更するコマンドです。

4.1.2. オプション

1. リクエスト指定方法

   リクエストはリクエスト ID かリクエスト名で指定します。

   なお、リクエストは 1 つだけ指定できます。

   -h $host-name

   リクエストはコマンド実行ユーザが所有する、ローカルホスト上のリクエストおよびローカルホストに投入してリモートマシンに転送されたリクエストの中から検索します。もしこの範囲外のリクエストを操作する場合は -h オプションでリクエストが存在しているリモートホスト名を指定してください。

   $request-id

   リクエスト ID はリクエストの投入時に表示されるもので、シーケンス番号と投入ホスト名からなります。シーケンス番号部分だけを指定した場合、ホスト名部分はローカルホストになります。

   -r $request-name

   リクエストをリクエスト名で指定したい場合は -r オプションを指定してください。ただし検索の範囲内に同名のリクエストが複数存在していた場合はエラーとなります。

   qalter コマンドで使用する属性変更用オプション（$alt-option）は以下のとおりです。

2. 属性変更オプション（$alt-option）の概要

-a	リクエスト実行時刻を変更します。
-e	標準エラー出力結果ファイルを変更します。
-jm	JOR(ジョブオカレンスレポート) の出力モードを変更します。
-l0	リクエストごとのファイルシステムグループ 0 制限値を変更します。
-l1	リクエストごとのファイルシステムグループ 1 制限値を変更します。
-l2	リクエストごとのファイルシステムグループ 2 制限値を変更します。
-l3	リクエストごとのファイルシステムグループ 3 制限値を変更します。
-lc	プロセスごとのコアファイルサイズ制限を変更します。
-ld	プロセスごとのデータセグメントサイズ制限を変更します。
-lD	リクエストごとのテープ装置台数制限を変更します。
-lf	プロセスごとの永久ファイルサイズ制限を変更します。
-lm	プロセスごとのメモリサイズ制限を変更します。

-lM	リクエストごとのメモリ領域制限を変更します。
-ln	プロセスのナイス実行値を変更します。
-lo	プロセスごとの同時オープンファイル数制限を変更します。
-lO	リクエストごとの同時オープンファイル数制限を変更します。
-lP	リクエストごとのプロセス数制限を変更します。
-ls	プロセスごとのスタックセグメントサイズ制限を変更します。
-lt	プロセスごとの CPU 時間制限を変更します。
-lT	リクエストごとの CPU 時間制限を変更します。
-lu	プロセスごとの CPU 台数制限を変更します。
-lU	リクエストごとの CPU 台数制限を変更します。
-lV	リクエストごとの一時ファイルサイズ制限を変更します。
-lw	プロセスごとのワーキングセット制限を変更します。
-lW	リクエストごとの物理メモリ制限、および事前確保値を変更します。
-lx	プロセスごとの永久ファイル容量制限を変更します。
-lX	リクエストごとの永久ファイル容量制限を変更します。
-mb	リクエスト実行開始時のメール送信モードを変更します。
-me	リクエスト実行終了時のメール送信モードを変更します。
-mu	メール送信先ユーザを変更します。
-nc	チェックポイントモードを変更します。
-nr	リクエスト再実行可不可モードを変更します。
-o	標準出力結果ファイルを変更します。
-p	キュー内でのリクエストプライオリティを変更します。
-re	標準エラー出力結果ファイル転送モードを変更します。
-ro	標準出力結果ファイル転送モードを変更します。
-s	リクエスト実行シェルを変更します。
-v	リクエスト実行レポートを、標準出力と標準エラー出力に出力します。
-ve	リクエスト実行レポートを、標準エラー出力に出力します。
-vo	リクエスト実行レポートを、標準出力に出力します。

3. リクエスト操作権

リクエストの操作はリクエストの所有者でなければできません。

4. 属性変更オプション（$alt-option）の説明

-a $date-time

バッチリクエストの実行時刻を変更します。時刻の指定の仕方については qsub(1) を参照してください。この時刻を変更できるのは、実行される前のリクエストに限ります。

-e [$machine:][[/]$path/]$stderr-filename

バッチリクエストの標準エラー出力先を、指定したファイルに変更します。[] で囲まれた部分は省略可能な部分です。

このファイル名を変更できるのは、実行される前のリクエストに限ります。

-jm $mode

バッチリクエストの JOR(ジョブオカレンスレポート) の出力先を mode に変更します。指定できるモードとして以下のものが用意されています。

■stderr

結果ファイル (stderr) に出力します。

■stdout

結果ファイル (stdout) に出力します。

■file file-name

指定ファイルに出力します。

■mail

メールで通知します。

このオプションは、対象となるリクエストが SUPER-UX 上に存在するときのみ有効です。それ以外の場合はエラーとなります。

-l0 $max-limit [,$warn-limit]

バッチリクエストに現在設定されている全プロセスに対するリクエストごとのファイルシステムグループ 0 (FSG0 = XMU) 制限の値を変更します。

制限値を 2 箇所以上空白で区切る場合、ダブルクォートで囲むか、 qalterとシェルが制限値の記述を 1 つの文字列として解釈できるようにエスケープする必要があります。

登録中のキュー (バッチキュー) の制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については、qsub(1) の制限の項を参照してください。

-l1 $max-limit [,$warn-limit]

バッチリクエストに現在設定されている全プロセスに対するリクエストごとのファイルシステムグループ 1 (FSG1) 制限の値を変更します。

制限値を 2 箇所以上空白で区切る場合、ダブルクォートで囲むか、 qalter とシェルが制限値の記述を 1 つの文字列として解釈できるようにエスケープする必要があります。

登録中のキュー (バッチキュー) の制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については、qsub(1) の制限の項を参照してください。

-l2 $max-limit [,$warn-limit]

バッチリクエストに現在設定されている全プロセスに対するリクエストごとのファイルシステムグループ 2 (FSG2) 制限の値を変更します。

制限値を 2 箇所以上空白で区切る場合、ダブルクォートで囲むか、 qalter とシェルが制限値の記述を 1 つの文字列として解釈できるようにエスケープする必要があります。

登録中のキュー (バッチキュー) の制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については、qsub(1) の制限の項を参照してください。

-l3 $max-limit [,$warn-limit]

バッチリクエストに現在設定されている全プロセスに対するリクエストごとのファイルシステムグループ 3 (FSG3) 制限の値を変更します。

制限値を 2 箇所以上空白で区切る場合、ダブルクォートで囲むか、 qalter とシェルが制限値の記述を 1 つの文字列として解釈できるようにエスケープする必要があります。

登録中のキュー (バッチキュー) の制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については、qsub(1) の制限の項を参照してください。

-lc $max-limit

プロセスごとのコアファイルサイズ制限の値を変更します。

この制限を変更できるのは、実行される前のリクエストに限ります。また登録中のキューの制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-ld $max-limit [,$warn-limit]

プロセスごとのデータセグメントサイズ制限の値を変更します。

この制限を変更できるのは、実行される前のリクエストに限ります。また登録中のキューの制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-lD $max-limit

リクエストごとのテープ装置台数制限の値を変更します。

登録中のキュー (バッチキュー) の制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

-lf $max-limit [,$warn-limit]

プロセスごとの永久ファイルサイズ制限の値を変更します。

現在、UNIXではカーネルにおいて永久ファイルと一時ファイルを区別するようなメカニズムはサポートされていません。つまり、厳密な意味での永久ファイルと一時ファイルの区別はできません。したがって、 UNIX では、この制限をプロセスごとのファイルサイズ制限として使用すべきです。

この制限を変更できるのは、実行される前のリクエストに限ります。また登録中のキューの制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-lm $max-limit [,$warn-limit]

プロセスごとのメモリサイズ制限の値を変更します。

この制限を変更できるのは、実行される前のリクエストに限ります。また登録中のキューの制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-lM $max-limit [,$warn-limit]

リクエストごとのメモリサイズ制限の値を変更します。

この制限を変更できるのは、実行される前のリクエストに限ります。また登録中のキューの制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-ln $value

プロセスごとの nice 値を変更します。実行中のリクエストの nice 値を変更することもできます。登録中のキューの制限値を超える値への変更はできません。このオプションを使って指定された nice 値は、リクエスト実行マシン上で解釈できる値でなければなりません。

-lo $max-limit

プロセスごとの同時オープンファイル数制限の値を変更します。

登録中のキュー (バッチキュー) の制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については、qsub(1) の制限の項を参照してください。

-IO $max-limit

バッチリクエストに現在設定されている全プロセスに対するリクエストごとの同時オープンファイル数制限の値を変更します。

登録中のキュー (バッチキュー) の制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については、qsub(1) の制限の項を参照してください。

-IP $max-limit

リクエストごとのプロセス数制限の値を変更します。

この制限を変更できるのは、実行される前のリクエストに限ります。また登録中のキューの制限値を超える値への変更はできません。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-ls $max-limit [,$warn-limit]

プロセスごとのスタックセグメントサイズ制限の値を変更します。

この制限を変更できるのは、実行される前のリクエストに限ります。また登録中のキューの制限値を超える値への変更はできません。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-lt $max-limit [,$warn-limit]

プロセスごとの CPU 時間制限の値を変更します。

この制限を変更できるのは、実行される前のリクエストに限ります。また登録中のキューの制限値を超える値への変更はできません。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-lT $max-limit

リクエストごとの CPU 時間制限の値を変更します。

この制限を変更できるのは、実行される前のリクエストに限ります。また登録中のキューの制限値を超える値への変更はできません。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-lu $max-limit

プロセスごとの CPU 台数制限の値を変更します。

登録中のキューの制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-lU $max-limit

リクエストごとの CPU 台数制限の値を変更します。

登録中のキューの制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-lV $max-limit

リクエストごとの一時ファイルサイズ制限の値を変更します。

この制限を変更できるのは、実行される前のリクエストに限ります。また登録中のキューの制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-lw $max-limit

プロセスごとのワーキングセットサイズ制限の値を変更します。

この制限を変更できるのは、実行される前のリクエストに限ります。また登録中のキューの制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-lW $max-limit[,$resv]

リクエストごとの物理メモリ領域制限値および事前確保値を変更します。

この制限を変更できるのは、実行される前のリクエストに限ります。また登録中のキューの制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則についてはqsub(1) の資源制限の項を参照してください。

-lx $max-limit[,$warn-limit]

プロセスごとの永久ファイル容量制限の値を変更します。

登録中のキュー (バッチキュー) の制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については、qsub(1) の制限の項を参照してください。

-lX $max-limit[,$warn-limit]

リクエストごとの永久ファイル容量制限の値を変更します。

登録中のキュー (バッチキュー) の制限値を超える値への変更はできません。

この資源制限値は、対象となるリクエストが存在するホストでサポートされている場合に限り有効です。サポートされていない場合は無視されます。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については、qsub(1) の制限の項を参照してください。

-mb $mode

バッチリクエストの実行開始時のメール送信モードを変更します。変更可能なモードは以下のとおりです。

■on

リクエスト実行開始時にメールを送信します。

■off

リクエスト実行開始時にメールを送信しません。

このモードを変更できるのは、実行される前のリクエストに限ります。

-me $mode

バッチリクエストの実行終了時のメール送信モードを変更します。変更可能なモードは以下のとおりです。

■on

リクエスト実行終了時にメールを送信します。

■off

リクエスト実行終了時にメールを送信しません。

このモードを変更できるのは、実行される前のリクエストに限ります。

-mu $user-name

メールを送信するユーザを変更します。

このユーザ名を変更できるのは、実行される前のリクエストに限ります。

-nc $mode

バッチリクエストのチェックポイントモードを指定した mode に変更します。

mode は on か off のどちらかを指定します。 on はチェックポイント採取不可モードで、 off はチェックポイント採取可モードです。

このモードを変更できるのは、実行される前のリクエストに限ります。また、このオプションは、対象となるリクエストがSUPER-UX 上にあるときのみ有効です。それ以外の場合はエラーとなります。

-nr $mode

バッチリクエストの再実行可否モードを指定した mode に変更します。

mode は on か off のどちらかを指定します。 on は再実行可モードで、 off は再実行不可モードです。このモードを変更できるのは、実行される前のリクエストに限ります。

-o [$machine:][[/]$path/]$stdout-filename

バッチリクエストの標準出力先を、指定したファイルに変更します。 [] で囲まれた部分は省略可能な部分です。

このファイル名を変更できるのは、実行される前のリクエストに限ります。

-p $priority

リクエストのキュー内部優先度を変更します。

プライオリティの範囲は、 [0...63] です。詳細は qsub(1) を参照してください。この値を変更できるのは、実行される前のリクエストに限ります。

-re $mode

バッチリクエストの標準エラー出力結果ファイル作成方法を指定した mode に変更します。

mode は s か n のどちらかを指定します。 s はスプールモードで n は非スプールモードです。

このモードを変更できるのは、実行される前のリクエストに限ります。

-ro $mode

バッチリクエストの標準出力結果ファイル作成方法を指定した mode に変更します。

mode は s か n のどちらかを指定します。 s はスプールモードで n は非スプールモードです。

このモードを変更できるのは、実行される前のリクエストに限ります。

-s $shell-name

バッチリクエストシェルスクリプトを実行するシェルをshell-nameで指定したシェルに変更します。shell-nameは絶対パス名で指定します。

このシェルを変更できるのは、実行される前のリクエストに限ります。

-v $level

リクエスト実行レポートを、リクエスト終了時に、リクエストの標準出力と標準エラー出力に出力します。

リクエストのプロセスは、フロックという単位で管理されており、リクエスト実行レポートはフロックの実行レポートとして出力されます。

フロック機能をサポートしていないマシン上のリクエストに対して、本オプションを指定することはできません。

リクエスト実行レポートの出力内容は level によって設定します。

[level]

■1

フロック開始時刻、フロック終了時刻、フロック終了ステータス、フロックタイプ、フロック ID を出力します。

■2

レベル 1 の全出力内容に加えて、テンポラリファイルのアサイン状況、テンポラリファイルのディアサイン状況を出力します。

-ve $level

リクエスト実行レポートを、リクエスト終了時に、リクエストの標準エラー出力に出力します。

リクエストのプロセスは、フロックという単位で管理されており、リクエスト実行レポートはフロックの実行レポートとして出力されます。

フロック機能をサポートしていないマシン上のリクエストに対して、本オプションを指 定することはできません。

リクエスト実行レポートの出力内容は level によって設定します。

[level]

■1

フロック開始時刻、フロック終了時刻、フロック終了ステータス、フロックタイプ、フロック ID を出力します。

■2

レベル 1 の全出力内容に加えて、テンポラリファイルのアサイン状況、テンポラリファイルのディアサイン状況を出力します。

-vo $level

リクエスト実行レポートを、リクエスト終了時に、リクエストの標準出力に出力します。リクエストのプロセスは、フロックという単位で管理されており、リクエスト実行レポートはフロックの実行レポートとして出力されます。

フロック機能をサポートしていないマシン上のリクエストに対して、本オプションを指定することはできません。

リクエスト実行レポートの出力内容は level によって設定します。

[level]

■1

フロック開始時刻、フロック終了時刻、フロック終了ステータス、フロックタイプ、フロック ID を出力します。

■2

レベル 1 の全出力内容に加えて、テンポラリファイルのアサイン状況、テンポラリファイルのディアサイン状況を出力します。

4.1.3. 注意事項

1. 制限事項およびインプリメンテーション留意点

   リクエストがバッチキューに存在する場合、そのシステムがサポートしていない資源制限値に関しては値を変更することはできません。また、リクエストがパイプキュー上で転送中の場合とバッチキュー上で実行中の場合は、変更できない属性があります。

2. その他注意事項

   リクエストがジョブトラッキング機能をもたない NQS が稼働しているホストを経由して転送された場合、リクエストの存在しているリモートホストを発見できない場合があります。

   その場合は -h オプションでリクエストが存在しているホスト名を指定してください。

4.1.4. 関連項目

qlimit(1), qstat(1), qsub(1), qmgr(1M).

4.2. qcat　実行中JobCenter リクエストのエラー/ 入出力ファイルの表示

```
/usr/bin/qcat [ -e ] [ -i ] [ -o ] [ -t $number ] [ -h $host-name ] [ -u $user-name ] $request-id ...

/usr/bin/qcat [ -e ] [ -i ] [ -o ] [ -t $number ] [ -h $host-name ] [ -u $user-name ] -r $request-name ...
```

4.2.1. 機能説明

qcat は実行中のJobCenter リクエストのエラーファイル、入力テキストファイル、出力テキストファイルのいずれかの内容を表示します。

qcat は 指定されたファイルがあれば、そのファイルを順番に読み込み、標準出力に表示します。

4.2.2. オプション

1. リクエスト指定方法

   リクエストはリクエスト ID かリクエスト名で指定します。

   -h $host-name

   リクエストはコマンド実行ユーザが所有する、ローカルホスト上のリクエストおよびローカルホストに投入してリモートマシンに転送されたリクエストの中から検索します。もしこの範囲外のリクエストを操作する場合は -h オプションでリクエストが存在しているリモートホスト名を指定してください。

   $request-id

   リクエスト ID はリクエストの投入時に表示されるもので、シーケンス番号と投入ホスト名からなります。シーケンス番号部分だけを指定した場合、ホスト名部分はローカルホストになります。

   -r $request-name

   リクエストをリクエスト名で指定したい場合は -r オプションを指定してください。ただし検索の範囲内に同名のリクエストが複数存在していた場合はエラーとなります。

2. リクエスト操作権

   リクエストの操作はリクエストの所有者でなければできません。ただしJobCenter管理者特権をもつユーザだけは、 -u $user-name オプションを用いて、指定したユーザのリクエストを操作することができます。

3. 各オプションの説明

   -e

   エラーファイルがあればそれを表示します。

   -i

   入力ファイル (スクリプトファイル) を表示します。

   -o

出力ファイルがあれば、それを表示します。

-t $number

ファイルの最後から number で指定された行数だけさかのぼったところから表示を行います。

-h $host-name

host-name で指定されたマシンから情報を得て表示します。

-u $user-name

user-name で指定したユーザのリクエストについて表示します。

スーパーユーザのみ指定可能です。

-e, -i, -o のいずれのオプションも指定されなければ、入力ファイルの表示を行います。

4.2.3. 注意事項

リクエストがトラッキング機能をもたない　NQS　が稼働しているホストを経由して転送された場合、リクエストの存在しているリモートホストを発見できない場合があります。その場合は-h　オプションでリクエストが存在しているホスト名を指定してください。

4.2.4. 関連項目

qstat(1).

4.3. qchk　バッチリクエストのチェックポイント採取

```
/usr/lib/nqs/qchk [ -f ] [[ -u $user-name ] | [ -h $host-name ]] $request-id ...

/usr/lib/nqs/qchk [ -f ] [[ -u $user-name ] | [ -h $host-name ]] -r $request-name ...
```

4.3.1. 機能説明

qchk は、コマンド行で指定された running および suspending 状態の バッチリクエストのチェックポイントを採取します。リクエストのチェックポイントを採取しても、リクエストの実行は終了しません。

すでにチェックポイントが取られているリクエストからチェックポイントを取る場合は、 -fオプションを使用します。このオプションを指定すると、新たにチェックポイントが採取されます。

チェックポイント採取に成功した場合、古いリスタートファイルは削除されます。このオプションを指定しなかった場合はエラーとなります。

なお、すでにチェックポイントが採取されているリクエストの一覧は、 qstatck(1) で参照することができます。

リクエストのチェックポイントを採取した状態で、システムがストールしたような場合、次回立ち上げ時にそのチェックポイントから自動的にリスタートされます。

4.3.2. オプション

1. リクエスト指定方法

   リクエストはリクエスト ID かリクエスト名で指定します。

   -h $host-name

   リクエストはコマンド実行ユーザが所有する、ローカルホスト上のリクエストおよびローカルホストに投入してリモートマシンに転送されたリクエストの中から検索します。もしこの範囲外のリクエストを操作する場合は -h オプションでリクエストが存在しているリモートホスト名を指定してください。

   -uオプションと併せて使用することはできません。

   $request-id

   リクエスト ID はリクエストの投入時に表示されるもので、シーケンス番号と投入ホスト名からなります。シーケンス番号部分だけを指定した場合、ホスト名部分はローカルホストになります。

   -r $request-name

   リクエストをリクエスト名で指定したい場合は -r オプションを指定してください。ただし検索の範囲内に同名のリクエストが複数存在していた場合はエラーとなります。

2. リクエスト操作権

   リクエストの操作はリクエストの所有者でなければできません。ただしJobCenter管理者特権をもつユーザだけは、 -u $user-name オプションを用いて、指定したユーザのリクエストを操作することができます。

-u $user-name

$user-name で指定したユーザのリクエストについて表示します。

JobCenter管理者のみ指定可能です。

-hオプションと併せて使用することはできません。

4.3.3. 注意事項

リクエストがトラッキング機能をもたない　NQS　が稼働しているホストを経由して転送された場合、リクエストの存在しているリモートホストを発見できない場合があります。その場合は-h　オプションでリクエストが存在しているホスト名を指定してください。

チェックポイントの採取ができるのは、リクエストが　SUPER-UX　上に存在しているときだけです。それ以外のときはエラーになります。

4.3.4. 関連項目

qrst(1), qstat(1), qstatck(1), qmgr(1M).

4.4. qdel　リクエストの削除

```
/usr/bin/qdel [ -k | - $signo  ] [[ -u $user-name ] | [ -h $host-name ]] $request-id ...

/usr/bin/qdel [ -k | - $signo  ] [[ -u $user-name ] | [ -h $host-name ]] -r $request-name ...
```

4.4.1. 機能説明

qdel　は、コマンド行で指定されたリクエストを削除したり、それらに対しシグナルを送信したりします。

-k オプションを指定した場合は、実行中のリクエストに SIGKILL シグナルを送信します。このシグナルを受け取ったリクエストは強制終了し、削除されます。また、-$signo オプション ($signo はシグナル番号) で、 SIGKILL の代わりに任意のシグナルを指定することもできます。

-kおよび　-$signoオプションを指定しなければ、実行中のリクエストを削除することはできません。

指定されたバッチリクエストが結果ファイル出力中（EXITING）であった場合、そのリクエストを親にもつ、すべてのネットワークリクエストが削除されます。削除されるネットワークリクエストが結果ファイル出力中 (RUNNING) だった場合は、 SIGKILL によって強制終了します。また、転送中だった結果ファイルは実行マシン上のリクエストオーナのホームディレクトリに置かれます。

なお、結果ファイル転送中のバッチリクエストを指定した場合は、 -k、 -$signo の各オプションは無視されます。 また、ネットワークリクエストを直接指定することはできません。

4.4.2. オプション

1. リクエスト指定方法

   リクエストはリクエスト ID かリクエスト名で指定します。

   -h $host-name

   リクエストはコマンド実行ユーザが所有する、ローカルホスト上のリクエストおよびローカルホストに投入してリモートマシンに転送されたリクエストの中から検索します。もしこの範囲外のリクエストを操作する場合は -h オプションでリクエストが存在しているリモートホスト名を指定してください。

   -uオプションと併せて使用することはできません。

   $request-id

   リクエスト ID はリクエストの投入時に表示されるもので、シーケンス番号と投入ホスト名からなります。シーケンス番号部分だけを指定した場合、ホスト名部分はローカルホストになります。

   -r $request-name

   リクエストをリクエスト名で指定したい場合は -r オプションを指定してください。ただし検索の範囲内に同名のリクエストが複数存在していた場合はエラーとなります。

2. リクエスト操作権

リクエストの操作はリクエストの所有者でなければできません。ただしJobCenter管理者特権をもつユーザだけは、 -u $user-name オプションを用いて、指定したユーザのリクエストを操作することができます。

-u $user-name

$user-name で指定したユーザのリクエストについて表示します。

JobCenter管理者のみ指定可能です。

-hオプションと併せて使用することはできません。

4.4.3. 注意事項

本システムではリクエストはプロセスグループとして管理されています。そしてシグナルの送信もプロセスグループに対して行われます。したがって、自らプロセスグループを変更するプロセスに対しては、シグナルを送ることはできません。そのようなプロセスには kill(1) コマンドでシグナルを送ってください。

リクエストがジョブトラッキング機能をもたない NQS が稼働しているホストを経由して転送された場合、リクエストの存在しているリモートホストを発見できない場合があります。その場合は -h オプションでリクエストが存在しているホスト名を指定してください。

4.4.4. 関連項目

qstat(1), qsub(1), qmgr(1M).

4.5. qhold　リクエストのホールド

```
/usr/bin/qhold [[ -u $user-name ] | [ -h $host-name ]] $request-id ...

/usr/bin/qhold [[ -u $user-name ] | [ -h $host-name ]] -r $request-name ...
```

4.5.1. 機能説明

qhold は、コマンド行で指定されたバッチリクエストをホールドします。リクエストのホールドとは一時リクエストを実行の対象から外すことであり、この状態の間はリクエストが実行されることはありません。

本コマンドによってホールドされたバッチリクエストは、 qrls(1) コマンドによって、そのホールドを解除することができます。

実行中のリクエストのホールドはできません。

リクエストをホールドした状態でJobCenter　がシャットダウンされた場合は、次回立ち上げ時にその状態が引き継がれます。

4.5.2. オプション

1. リクエスト指定方法

   リクエストはリクエスト ID かリクエスト名で指定します。

   -h $host-name

   リクエストはコマンド実行ユーザが所有する、ローカルホスト上のリクエストおよびローカルホストに投入してリモートマシンに転送されたリクエストの中から検索します。もしこの範囲外のリクエストを操作する場合は -h オプションでリクエストが存在しているリモートホスト名を指定してください。

   -uオプションと併せて使用することはできません。

   $request-id

   リクエスト ID はリクエストの投入時に表示されるもので、シーケンス番号と投入ホスト名からなります。シーケンス番号部分だけを指定した場合、ホスト名部分はローカルホストになります。

   -r $request-name

   リクエストをリクエスト名で指定したい場合は -r オプションを指定してください。ただし検索の範囲内に同名のリクエストが複数存在していた場合はエラーとなります。

2. リクエスト操作権

   リクエストの操作はリクエストの所有者でなければできません。ただしJobCenter管理者特権をもつユーザだけは、 -u $user-name オプションを用いて、指定したユーザのリクエストを操作することができます。

   -u $user-name

   $user-name で指定したユーザのリクエストについて表示します。

JobCenter管理者のみ指定可能です。

-hオプションと併せて使用することはできません。

4.5.3. 注意事項

リクエストがジョブトラッキング機能をもたない NQS が稼働しているホストを経由して転送された場合、リクエストの存在しているリモートホストを発見できない場合があります。その場合は -h オプションでリクエストが存在しているホスト名を指定してください。

4.5.4. 関連項目

qrls(1), qstat(1), qsub(1), qmgr(1M).

4.6. qlimit　システムでサポートされている資源制限とシェル選択方式の表示

```
/usr/bin/qlimit [ $host-name ... ]
```

4.6.1. 機能説明

qlimit は、ローカルホストや指定ホスト上のJobCenter でサポートされているバッチリクエスト資源制限のタイプと、定義されているバッチリクエストのシェル選択方式を表示します。

4.6.2. オプション

$host-nameを省略すると、qlimitはローカルホストの情報を表示します。$host-name　を指定した場合、qlimitは各指定ホストの情報を表示します。

4.6.3. 関連項目

「「6.4 シェル選択方式指定」」

qstat(1), qsub(1), qmgr(1M).

4.7. qmove　リクエストの移動

```
/usr/bin/qmove [[ -u $user-name ] | [ -h $host-name ]] $request-id ... $queue

/usr/bin/qmove [[ -u $user-name ] | [ -h $host-name ]] -r $request-name ... $queue

/usr/bin/qmove [[ -u $user-name ] | [ -h $host-name ]] -q $queue $queue
```

4.7.1. 機能説明

qmove　は、コマンド行で指定されたリクエストを移動します。リクエストの移動とはリクエストを現在登録されているキューから、ほかのキューに移すことです。また -q オプションを指定すると指定キューに登録されているすべてのリクエストを一気に移動させることができます。

リクエストの移動ができるのはリクエストが実行される前のみです。実行中のリクエストを指定した場合はエラーとなります (ネットワークリクエストを除く)。

$request-idで結果ファイル転送中のバッチリクエストを指定し、$queueでネットワークキューを指定すると、指定されたバッチリクエストを親とするネットワークリクエストをすべて指定されたネットワークキューに移動させることができます。このとき、 RUNNING 状態のネットワークリクエストも移動の対象となります。ただし、移動したネットワークキューが転送先としているホストマシン上にリクエストに設定されている結果ファイル出力先のパスとおなじパスがなければ、結果ファイル転送は失敗します。

-q　オプションを使っての、ネットワークキューからネットワークキューへのネットワークリクエストの移動はできません。また、$request-idに直接ネットワークリクエストを指定することもできません。

4.7.2. オプション

1. リクエスト指定方法

   リクエストはリクエスト ID かリクエスト名で指定します。

   -h $host-name

   リクエストはコマンド実行ユーザが所有する、ローカルホスト上のリクエストおよびローカルホストに投入してリモートマシンに転送されたリクエストの中から検索します。もしこの範囲外のリクエストを操作する場合は　-h　オプションでリクエストが存在しているリモートホスト名を指定してください。

   -uオプションと併せて使用することはできません。

   $request-id

   リクエスト　ID　はリクエストの投入時に表示されるもので、シーケンス番号と投入ホスト名からなります。シーケンス番号部分だけを指定した場合、ホスト名部分はローカルホストになります。

   -r $request-name

   リクエストをリクエスト名で指定したい場合は　-r　オプションを指定してください。ただし検索の範囲内に同名のリクエストが複数存在していた場合はエラーとなります。

2. リクエスト操作権

リクエストの操作はリクエストの所有者でなければできません。ただしJobCenter管理者特権をもつユーザだけは、 -u $user-name オプションを用いて、指定したユーザのリクエストを操作することができます。

-u $user-name

$user-name で指定したユーザのリクエストについて表示します。

JobCenter管理者のみ指定可能です。

-hオプションと併せて使用することはできません。

4.7.3. 注意事項

リクエストがジョブトラッキング機能をもたない NQS が稼働しているホストを経由して転送された場合、リクエストの存在しているリモートホストを発見できない場合があります。その場合は -h オプションでリクエストが存在しているホスト名を指定してください。

4.7.4. 関連項目

qstat(1), qsub(1), qmgr(1M).

4.8. qmsg　結果ファイルへのメッセージの送信

```
/usr/bin/qmsg [ -e ] [ -o ] $request-id

/usr/bin/qmsg [ -e ] [ -o ] -r $request-name
```

4.8.1. 機能説明

qmsgコマンドは実行中のバッチリクエストの標準出力および標準エラー出力にメッセージを書き込みます。メッセージは標準入力から読み込まれます。対象とするリクエストはローカルホスト上に存在していなければなりません。

4.8.2. オプション

1. リクエスト指定方法

   リクエストはリクエスト ID かリクエスト名で指定します。

   なお、リクエストは 1 つだけ指定できます。

   $request-id

   リクエスト ID はリクエストの投入時に表示されるもので、シーケンス番号と投入ホスト名からなります。シーケンス番号部分だけを指定した場合、ホスト名部分はローカルホストになります。

   -r $request-name

   リクエストをリクエスト名で指定したい場合は -r オプションを指定してください。ただし検索の範囲内に同名のリクエストが複数存在していた場合はエラーとなります。

2. リクエスト操作権

   メッセージの書き込みはリクエストの所有者かJobCenter 管理者特権をもつユーザだけが行うことができます。

3. 各オプションの説明

   -e

   メッセージをバッチリクエストの標準エラー出力ファイルに書き込みます。

   -o

   メッセージをバッチリクエストの標準出力ファイルに書き込みます。

4.8.3. 関連項目

qstat(1), qsub(1).

4.9. qrerun　バッチリクエストの再登録

```
/usr/bin/qrerun [ -hold ] [ -a $restart-time ] [[ -u $user-name ] | [ -h $host-name ]] $request-id ...

/usr/bin/qrerun [ -hold ] [ -a $restart-time ] [[ -u $user-name ] | [ -h $host-name ]] -r $request-name ...
```

4.9.1. 機能説明

qrerun　は、コマンド行で指定されたバッチリクエストを再登録します。リクエストの再登録とは実行中のリクエストの実行を途中でやめて、あらためて最初から実行しなおすことです。リクエストの実行中断は SIGKILL シグナルを送信することで行います。

リクエストの再登録ができるのはリクエストが実行中 (RUNNING) のときのみです。それ以外の状態のリクエストを指定した場合は警告が発せられます。

4.9.2. オプション

1. リクエスト指定方法

   リクエストはリクエスト ID かリクエスト名で指定します。

   -h $host-name

   リクエストはコマンド実行ユーザが所有する、ローカルホスト上のリクエストおよびローカルホストに投入してリモートマシンに転送されたリクエストの中から検索します。もしこの範囲外のリクエストを操作する場合は　-h　オプションでリクエストが存在しているリモートホスト名を指定してください。

   -uオプションと併せて使用することはできません。

   $request-id

   リクエスト　ID　はリクエストの投入時に表示されるもので、シーケンス番号と投入ホスト名からなります。シーケンス番号部分だけを指定した場合、ホスト名部分はローカルホストになります。

   -r $request-name

   リクエストをリクエスト名で指定したい場合は　-r　オプションを指定してください。ただし検索の範囲内に同名のリクエストが複数存在していた場合はエラーとなります。

2. リクエスト操作権

   リクエストの操作はリクエストの所有者でなければできません。ただしJobCenter管理者特権をもつユーザだけは、-u　$user-nameオプションを用いて、指定したユーザのリクエストを操作することができます。

   -u $user-name

   $user-name で指定したユーザのリクエストについて表示します。

   JobCenter管理者のみ指定可能です。

-hオプションと併せて使用することはできません。

3. 再登録状態の指定

-hold

リクエストを再登録後、スケジューリングを保留する場合に指定します。

このオプションを指定しない場合、リクエストは再登録が完了すると直ちにスケジューリングが開始されます。

-a $restart-time

再登録後、指定時刻に再実行を開始したい場合に指定します。時刻指定の形式は qsub(1) の -a オプション指定と同じです。

4.9.3. 注意事項

本システムではリクエストはプロセスグループとして管理されています。そしてシグナルの送信もプロセスグループに対して行われます。したがって、自らプロセスグループを変更するプロセスに対しては、シグナルを送ることはできません。そのようなプロセスには kill(1) コマンドでシグナルを送ってください。

リクエストがジョブトラッキング機能をもたない NQS が稼働しているホストを経由して転送された場合、リクエストの存在しているリモートホストを発見できない場合があります。その場合は -h オプションでリクエストが存在しているホスト名を指定してください。

4.9.4. 関連項目

qlimit(1), qstat(1), qsub(1), qmgr(1M).

4.10. qrls　バッチリクエストのホールド解除

```
/usr/bin/qrls [[ -u $user-name ] | [ -h $host-name ]] $request-id ...

/usr/bin/qrls [[ -u $user-name ] | [ -h $host-name ]] -r $request-name ...
```

4.10.1. 機能説明

qrls は、 qhold(1) コマンドでホールドされたバッチリクエストのホールドを解除します。リクエストのホールドとは一時リクエストを実行の対象から外すことであり、この状態の間はリクエストが実行されることはありません。したがって本コマンドでホールドを解除すると、リクエストは再び実行の対象となります。

リクエストのホールド解除ができるのはリクエストがホールド (HOLDING) のときに限り、それ以外の状態のリクエストを指定した場合は警告が発せられます。

qmgr(1M) のサブコマンドによってホールドされたリクエストは、このコマンドでは解除できません。

4.10.2. オプション

1. リクエスト指定方法

   リクエストはリクエスト ID かリクエスト名で指定します。

   -h $host-name

   リクエストはコマンド実行ユーザが所有する、ローカルホスト上のリクエストおよびローカルホストに投入してリモートマシンに転送されたリクエストの中から検索します。もしこの範囲外のリクエストを操作する場合は -h オプションでリクエストが存在しているリモートホスト名を指定してください。

   -uオプションと併せて使用することはできません。

   $request-id

   リクエスト ID はリクエストの投入時に表示されるもので、シーケンス番号と投入ホスト名からなります。シーケンス番号部分だけを指定した場合、ホスト名部分はローカルホストになります。

   -r $request-name

   リクエストをリクエスト名で指定したい場合は -r オプションを指定してください。ただし検索の範囲内に同名のリクエストが複数存在していた場合はエラーとなります。

2. リクエスト操作権

   リクエストの操作はリクエストの所有者でなければできません。ただしJobCenter管理者特権をもつユーザだけは、-u　$user-nameオプションを用いて、指定したユーザのリクエストを操作することができます。

   -u $user-name

   $user-name で指定したユーザのリクエストについて表示します。

   JobCenter管理者のみ指定可能です。

-hオプションと併せて使用することはできません。

4.10.3. 注意事項

リクエストがジョブトラッキング機能をもたない NQS が稼働しているホストを経由して転送された場合、リクエストの存在しているリモートホストを発見できない場合があります。その場合は -h オプションでリクエストが存在しているホスト名を指定してください。

4.10.4. 関連項目

qhold(1), qstat(1), qsub(1), qmgr(1M).

4.11. qrsm　バッチリクエストの実行の再開

```
/usr/bin/qrsm [[ -u $user-name ] | [ -h $host-name ]] $request-id ...

/usr/bin/qrsm [[ -u $user-name ] | [ -h $host-name ]] -r $request-name ...
```

4.11.1. 機能説明

qrsm は、コマンド行で指定された実行中断中のバッチリクエストの実行を再開します。実行再開は SIGCONT シグナルを送信することで行います。

リクエストの実行の再開ができるのはリクエストが一時中断 (SUSPEND) のときに限り、それ以外の状態のリクエストを指定した場合は警告が発せられます。

qmgr(1M) のサブコマンドによって中断されたリクエストは、このコマンドでは再開できません。

4.11.2. オプション

1. リクエスト指定方法

   リクエストはリクエスト ID かリクエスト名で指定します。

   -h $host-name

   リクエストはコマンド実行ユーザが所有する、ローカルホスト上のリクエストおよびローカルホストに投入してリモートマシンに転送されたリクエストの中から検索します。もしこの範囲外のリクエストを操作する場合は -h オプションでリクエストが存在しているリモートホスト名を指定してください。

   -uオプションと併せて使用することはできません。

   $request-id

   リクエスト ID はリクエストの投入時に表示されるもので、シーケンス番号と投入ホスト名からなります。シーケンス番号部分だけを指定した場合、ホスト名部分はローカルホストになります。

   -r $request-name

   リクエストをリクエスト名で指定したい場合は -r オプションを指定してください。ただし検索の範囲内に同名のリクエストが複数存在していた場合はエラーとなります。

2. リクエスト操作権

   リクエストの操作はリクエストの所有者でなければできません。ただしJobCenter 管理者特権をもつユーザだけは、 -u $user-name オプションを用いて、指定したユーザのリクエストを操作することができます。

   -u $user-name

   $user-name で指定したユーザのリクエストについて表示します。

   JobCenter管理者のみ指定可能です。

-hオプションと併せて使用することはできません。

4.11.3. 注意事項

本システムではリクエストはプロセスグループとして管理されています。そしてシグナルの送信もプロセスグループに対して行われます。したがって、自らプロセスグループを変更するプロセスに対しては、シグナルを送ることはできません。そのようなプロセスには kill(1) コマンドでシグナルを送ってください。

リクエストがジョブトラッキング機能をもたない NQS が稼働しているホストを経由して転送された場合、リクエストの存在しているリモートホストを発見できない場合があります。その場合は -h オプションでリクエストが存在しているホスト名を指定してください。

4.11.4. 関連項目

qlmit(1), qstat(1), qstatr(1), qsub(1), qmgr(1M).

4.12. qspnd　バッチリクエストの実行の一時中断

```
/usr/bin/qspnd [[ -u $user-name ] | [ -h $host-name ]] $request-id ...

/usr/bin/qspnd [[ -u $user-name ] | [ -h $host-name ]] -r $request-name ...
```

4.12.1. 機能説明

qspndは、コマンド行で指定された実行中のバッチリクエストを一時中断します。実行の中断はSIGSTOPシグナルを送信することで行います。

リクエストの実行の一時中断ができるのはリクエストが実行中のときに限り、それ以外の状態のリクエストを指定した場合は警告が発せられます。

4.12.2. オプション

1. リクエスト指定方法

   リクエストはリクエスト ID かリクエスト名で指定します。

   -h $host-name

   リクエストはコマンド実行ユーザが所有する、ローカルホスト上のリクエストおよびローカルホストに投入してリモートマシンに転送されたリクエストの中から検索します。もしこの範囲外のリクエストを操作する場合は -h オプションでリクエストが存在しているリモートホスト名を指定してください。

   -uオプションと併せて使用することはできません。

   $request-id

   リクエスト ID はリクエストの投入時に表示されるもので、シーケンス番号と投入ホスト名からなります。シーケンス番号部分だけを指定した場合、ホスト名部分はローカルホストになります。

   -r $request-name

   リクエストをリクエスト名で指定したい場合は -r オプションを指定してください。ただし検索の範囲内に同名のリクエストが複数存在していた場合はエラーとなります。

2. リクエスト操作権

   リクエストの操作はリクエストの所有者でなければできません。ただしJobCenter管理者特権をもつユーザだけは、-u　$user-nameオプションを用いて、指定したユーザのリクエストを操作することができます。

   -u $user-name

   $user-name で指定したユーザのリクエストについて表示します。

   JobCenter管理者のみ指定可能です。

   -hオプションと併せて使用することはできません。

4.12.3. 注意事項

qspnd コマンドによって実行中断しているリクエストのプロセスに対して、 kill(1) コマンドなどで、シグナルを送信した場合の動作は保証されません。

本システムではリクエストはプロセスグループとして管理されています。そしてシグナルの送信もプロセスグループに対して行われます。したがって、自らプロセスグループを変更するプロセスに対しては、シグナルを送ることはできません。そのようなプロセスには kill(1) コマンドでシグナルを送ってください。

リクエストがジョブトラッキング機能をもたない NQS が稼働しているホストを経由して転送された場合、リクエストの存在しているリモートホストを発見できない場合があります。その場合は -h オプションでリクエストが存在しているホスト名を指定してください。

4.12.4. 関連項目

qlmit(1), qstat(1), qstatr(1), qsub(1), qmgr(1M).

4.13. qstat　JobCenterの状態表示

```
/usr/bin/qstat [ -a ] [ -l ] [ -m ] [ -u $user-name ] [ -x ] [ $queue-name ...] [ $queue-name@host-name ...]
```

4.13.1. 機能説明

qstat コマンドは、JobCenter キューの状態を表示します。

キューを指定しなければ、ローカルホスト上の全JobCenter キューの状態を表示します。

キューを指定した場合、その指定されたキューの状態のみを表示します。キューは、$queue-name または $queue-name@$host-name のどちらかで指定します。$host-nameの指定を省略した場合は、ローカルホストとみなされます。$queue-name の形式と$queue-name@host-nameの形式を共存させることもできます。

各選択されたキューごとに、 qstat はキューヘッダ (キュー自身の情報) と、キュー内のリクエストに関する情報を表示します。普通、 qstat は要求者の所有するリクエストについてのみ情報を表示します。

4.13.2. オプション

1. 各オプションの説明

-a

すべてのリクエストを示します。

-l

リクエストは長い形式で示されます。

-m

リクエストは中間の長さの形式で示されます。

-u $user-name

$user-name が所有するリクエストについてのみ表示します。

-x

キューヘッダは拡張形式で示されます。

キューヘッダは常にキュー名、キュータイプ、キュー状態 (以下参照)、キューが pipeonly(パイプキューのみからリクエストを受け付ける) かどうかの印、キューのリクエストの数を含んでいます。拡張キューヘッダは優先度の表示とキューの実行制限、加えてアクセス制限、累積使用統計、サーバと目的地 (パイプキューの場合)のキューと資源制限 (バッチキューの場合) が付加されます。

既定で、qstatはリクエストについて、リクエスト名・リクエスト　id・オーナ・相対リクエスト優先度・現在のリクエスト状態 (以下参照) の情報を表示します。実行中のリクエストでは、プロセスグループ ID の情報がローカル JobCenter デーモンで取得できしだい、表示されます。

qstat -m は、リクエストに特定の日時になるまで実行されないオプションが付けて投入されていれば、その日時も表示されます。

qstat -l は、リクエストが作成された時間およびメールが送られるかどうかの印、メールがどこに送られるか、本来のマシンの使用者名を表示します。バッチキューが指定された場合は、資源制限と、予定された stderr、 stdout 結果ファイル、コマンドインタプリタ、 umask 値、再実行されたものかどうかの印、が表示されます。

2. キュー状態

キューの状態は、キューの 2 つの特性により定義されます。詳細については「7.4 キューの運用管理」を参照してください。

■第1の特性は次のように表示されます。

リクエストはキューが投入可能 (enabled) であり, ローカル JobCenter デーモンが稼動中のときに限り、投入可能です。

ENABLED	キューはリクエストの登録を受け付ける状態です。
DISABLED	キューはリクエストの登録を受け付けない状態です。
CLOSED	JobCenterシステム停止中です。従ってリクエストの登録はできません。

■第2の特性は次のように表示されます。

INACTIVE	キューはリクエストの実行を行う状態です。ただし、そのキュー上のリクエストで現在実行中のものがない状態です。
RUNNING	キューはリクエストの実行を行う状態です。また、そのキュー上のリクエストで現在実行中のものがある状態です。
STOPPED	キューはリクエストの実行を行わない状態です。また、そのキュー上のリクエストで現在実行中のものもない状態です。
STOPPING	キューはリクエストの実行を行わない状態です。ただし、そのキュー上のリクエストで現在実行中のものがある状態です。
SHUTDOWN	JobCenterシステム停止中です。

3. リクエスト状態

リクエストの状態は、次のように表示されます。

arriving	リクエストがリモートホストから転送されて登録中です。
holding	リクエストがホールドされていて、ほかの状態(実行中状態を含む)になることを現在妨害されています。
waiting	リクエストが定められた日時になるまで実行されない制約つきで投入されていて、その日時がまだ来ていません。
queued	リクエストが発送中状態や実行中状態を経由済みで、実行するための資格があります。
staging	バッチリクエストはまだ実行を開始していないが、その入力ファイルが実行マシンに運ばれている最中です。
routing。	リクエストがパイプキューの先頭に到着して、そこでサービスを受けようとしている状態です
running	リクエストが最後の目的キューに到達して、実際に実行中の状態です。
suspending	リクエストの実行が中断しています。
departing	リクエストがパイプキューにより転送処理されて、目的地に完全に到着するまで配送中の状態です。
exiting	リクエストが実行を完了しています。リクエストは要求された出力ファイルが投入元に返された後、システムから退去します。

ワークステーションで入力し、大型計算機上のバッチキューに登録され、即時に実行されるバッチリクエストについて考えてみます。

そのリクエストはローカルホスト(ワークステーション)上のパイプキュー内で登録済みおよび発送中、配送中の各状態を遷移し、大型計算機にリクエストが転送されます。リクエストは、その後パイプキューから消えます。

大型計算機上のキューの視点からみれば、リクエストは、バッチキュー内で到着中状態、登録済み、ステージ中(バッチリクエストによって要求されたら)、実行中、終了処理中の状態へと遷移します。実行の終了処理の完了で、バッチリクエストはバッチキューから消えます。

4.13.3. 注意事項

本システムでは、ステージ中 (staging)、配送中 (departing) の各状態をサポートしていません。従ってリクエストがそのような状態になることはありません。

4.13.4. 関連項目

qdel(1), qlimit(1), qsub(1), qmgr(1M).

4.14. qstata　JobCenter アクセス制限の情報表示

```
/usr/bin/qstata [ -a ] [ -ac $acct-code ] [ -n ] [ -h $host-name ]
```

4.14.1. 機能説明

qstataコマンドはJobCenter　アクセス制限の情報を表示します。アクセス制限とは、ユーザに対してJobCenter　　を利用する権利を制限するものです。つまり、qstataコマンドにより、自分がJobCenter を利用できるかどうかを知ることができます。

アクセス制限は、管理者によってユーザあるいはグループ単位に設定されます。アクセス制限を設定されたユーザはJobCenterの機能を利用することはできません。

また、リモートマシン上のユーザはリモートシェル実行権によっても制限を受けます。

また SUPER-UXでは、予算管理によっても制限をうけます。

4.14.2. オプション

1. 各オプションの説明

-a

アクセス制限を受けているユーザグループのリストを表示します。管理者のみが使用できます。

本オプションを指定した場合に表示されるリストの見出しとフィールドの意味は以下のとおりです。

■USER

被アクセス制限ユーザ名リスト

■GROUP

被アクセス制限グループ名リスト

■BUDGET

予算超過によりアクセス制限を受けているユーザ・グループ・アカウントコードのリスト(SUPER-UXでのみ表示)

-hオプションと併せて使用することはできません。

-a　オプションが指定されていない場合、アクセス制限の状態に応じて以下のメッセージを表示します。

■アクセス制限を受けていない場合

You are permitted to place requests in NQS. (返却値 = 0)

■ユーザとしてアクセス制限を受けている場合

You are not permitted to place requests in NQS. (返却値 = 1)

■グループとしてアクセス制限を受けている場合

Your group is not permitted to place requests in NQS. (返却値 = 2)

-ac $acct-code

チェックされるアカウントコードを指定します。 -a オプションと同時に使用することはできません。

このオプションは SUPER-UX 上でのアクセス制限を表示する場合のみ有効です。

-h $host-name

ローカルホストのユーザが、リモートホスト上の JobCenter にアクセスできるかどうかを表示します。

-aオプションと併せて使用することはできません。

-n

メッセージを表示しません。

4.14.3. 注意事項

-h オプションを指定した場合、返却値は正しく返されません。

4.15. qstatq　キューの状態表示

```
/usr/bin/qstatq [ -b ] [ -d ] [ -p ] [ -N ] [ -f ] [ -L ] [ -n ] [ -h $host-name ] [ $queue-name ...]
```

4.15.1. 機能説明

qstatq コマンドはJobCenter キューの状態を表示します。パラメータに$queue-nameを指定すると、そのキューを対象にして情報が出力されます。ただしオプションの種類によっては、キュー名リスト指定が無効になる場合もあります　　(以下に示すオプション一覧参照)。また、$queue-nameを指定しない場合は、対象ホスト上のすべてのキューに関する情報が出力されます。

1. キュー状態

キューの状態は、キューの 2 つの特性により定義されます。詳細については「7.4 キューの運用管理」を参照してください。

■第1の特性は次のように表示されます。

リクエストはキューが投入可能 (enabled) であり, ローカル JobCenter デーモンが稼動中のときに限り、投入可能です。

ENABLED	キューはリクエストの登録を受け付ける状態です。
DISABLED	キューはリクエストの登録を受け付けない状態です。
CLOSED	JobCenterシステム停止中です。従ってリクエストの登録はできません。

■第2の特性は次のように表示されます。

INACTIVE	キューはリクエストの実行を行う状態です。ただし、そのキュー上のリクエストで現在実行中のものがない状態です。
RUNNING	キューはリクエストの実行を行う状態です。また、そのキュー上のリクエストで現在実行中のものがある状態です。
STOPPED	キューはリクエストの実行を行わない状態です。また、そのキュー上のリクエストで現在実行中のものもない状態です。
STOPPING	キューはリクエストの実行を行わない状態です。ただし、そのキュー上のリクエストで現在実行中のものがある状態です。
SHUTDOWN	JobCenterシステム停止中です。

2. リクエスト状態

リクエストの状態は、次のように表示されます。

arriving	リクエストがリモートホストから転送されて登録中です。
holding	リクエストがホールドされていて、ほかの状態(実行中状態を含む)になることを現在妨害されています。
waiting	リクエストが定められた日時になるまで実行されない制約つきで投入されていて、その日時がまだ来ていません。
queued	リクエストが発送中状態や実行中状態を経由済みで、実行するための資格があります。
staging	バッチリクエストはまだ実行を開始していないが、その入力ファイルが実行マシンに運ばれている最中です。
routing。	リクエストがパイプキューの先頭に到着して、そこでサービスを受けようとしている状態です。
running	リクエストが最後の目的キューに到達して、実際に実行中の状態です。

suspending	リクエストの実行が中断しています。
departing	リクエストがパイプキューにより転送処理されて、目的地に完全に到着するまで配送中の状態です。
exiting	リクエストが実行を完了しています。リクエストは要求された出力ファイルが投入元に返された後、システムから退去します。

3. 表示形式(-f、-Lオプションを指定しない場合)

見出しの後ろに(b)と記されているものはバッチキューについて、(p)と記されているものはパイプキューについて、(n)と記されているものはネットワークキューについて示されるものです。また、(SUPER-UX) と記されているものは SUPER-UX でのみ表示されます。いずれのキューの場合もヘッダにはホスト名が表示されます。

表4.1 –f、−Lオプションを指定しない場合のリストの見出しとフィールドの意味

見出し	フィールドの意味
QUEUE NAME (b, d, p, n)	キュー名
DESTINATION MACHINE (n)	結果ファイル転送先ホスト
ENA (b, d, p, n)	状態 (第一要因) ENA (enabled) DIS (disabled) CLS (closed)
STS (b, d, p, n)	状態 (第二要因) STP (stopped/stopping) RUN (running) INA (inactive) SHT (shutdown)
PRI (b, d, p, n)	キュープライオリティ
BPR (b)	ベースプライオリティ (SUPER-UX)
TMS (b)	タイムスライス値 (SUPER-UX)
MPR (b)	メモリスケジューリングプライオリティ (SUPER-UX)
RLM (b, d, p, n)	キューに課せられた同時実行可能リクエスト数制限値
TOT (b, d, p, n)	キューに登録されているリクエスト数
QUE (b, d, p, n)	queued 状態のリクエスト数
RUN (b, d, n)	running 状態のリクエスト数
ROU (p)	routing 状態のリクエスト数
WAI (b, d, p, n)	waiting 状態のリクエスト数
HLD (b, d, p)	holding 状態のリクエスト数
SUS (b)	suspending 状態のリクエスト数
ARR (b, d, p)	arriving 状態のリクエスト数
EXT (b)	exiting 状態のリクエスト数
<TOTAL> (b, d, p, n)	上記の RLM, TOT, QUE, RUN, WAI, HLD, SUS, ARR, ROU, EXT のそれぞれの項の合計。ただしRLMについては バッチキューの場合:　全バッチキューに対して課せられた同時実行可能リクエスト数制限値

	パイプキューの場合:　全パイプキューに対して課せられた同時実行可能リクエスト数制限値 ネットワークキューの場合:　全ネットワークキューに対して課せられた同時転送可能リクエスト数制限値 が表示されます。

4. 表示形式(-Lオプションを指定した場合)

この場合、ヘッダにはホスト名 (最大 15 文字) が表示されます。

意味の後ろに (SUPER-UX) とあるものについては、 SUPER-UX でのみ表示されます。

表4.2 −Lオプションを指定した場合のリストの見出しとフィールドの意味

見出し	フィールドの意味
QUEUE NAME	キュー名 (最大 15 文字)
RLM	キューに課せられた同時実行可能リクエスト数制限値 (Not SUPER-UX)
TOTAL	キューに課せられた同時実行可能リクエスト数制限値 (SUPER-UX)
USER	キューに課せられたユーザごとの同時実行可能リクエスト数制限値 (SUPER-UX)
GROUP	キューに課せられたグループごとの同時実行可能リクエスト数制限値 (SUPER-UX)
XMU	キューに課せられた同時実行可能ファイルシステムグループ 0 (XMU)制限値 (SUPER-UX)
COR/P	プロセスごとのコアファイルサイズ制限値
CPU/P	プロセスごとの CPU 時間制限値
MEM/P	プロセスごとのメモリサイズ制限値
DAT/P	プロセスごとのデータセグメントサイズ制限値
STK/P	プロセスごとのスタックセグメントサイズ制限値
FSZ/P	プロセスごとのファイルサイズ制限値 (Not SUPER-UX)
PFSZ/P	プロセスごとの永久ファイルサイズ制限値 (SUPER-UX)
PFCP/P	プロセスごとの永久ファイル容量制限値 (SUPER-UX)
OFIL/P	プロセスごとの同時オープンファイル数制限値 (SUPER-UX)
NCPU/P	プロセスごとの CPU 台数制限値 (SUPER-UX)
PRC/R	リクエストごとのジョブ構成プロセス数制限値
CPU/R	リクエストごとの CPU 時間制限値
MEM/R	リクエストごとのメモリサイズ制限値 (SUPER-UX)
PFCP/R	リクエストごとの永久ファイル容量制限値 (SUPER-UX)
TPCP/R	リクエストごとのテンポラリファイル容量制限値 (SUPER-UX)
OFIL/R	リクエストごとの同時オープンファイル数制限値 (SUPER-UX)
DRV/R	リクエストごとのテープ装置台数 (SUPER-UX)
FSG0/R	リクエストごとのファイルシステムグループ 0 制限値 (SUPER-UX)
FSG1/R	リクエストごとのファイルシステムグループ 1 制限値 (SUPER-UX)
FSG2/R	リクエストごとのファイルシステムグループ 2 制限値 (SUPER-UX)
FSG3/R	リクエストごとのファイルシステムグループ 3 制限値 (SUPER-UX)
NCPU/R	リクエストごとの CPU 台数制限値 (SUPER-UX)

TMP/R	リクエストごとの一時ファイルサイズ制限 (Not SUPER-UX)
PHY/R	リクエストごとの物理メモリ制限 (Not SUPER-UX)

HP-UX では、プロセスごとのファイルサイズ制限値以外の資源制限は表示されますがサポートはされません。

5. 表示形式(-fオプションを指定した場合)

見出しの後ろに(b)と記されているものはバッチキューについて、(p)と記されているものはパイプキューについて、また(n)と記されているものはネットワークキューについて示されるものです。

いずれのキューの場合もヘッダには"キュー名@ホスト名"が付与されて表示されます。

意味の後ろに (SUPER-UX) とあるものについては、 SUPER-UX でのみ表示されます。

表4.3 −fオプションを指定した場合のリストの見出しとフィールドの意味

見出し	フィールドの意味	
Priority (b, d, p, n)	キュー相対優先度	
Status (b, d, p, n)	状態 (第一要因, 第二要因)	
Base Batch Priority (b)	バッチベースプライオリティ (SUPER-UX)	
Time slice value (b)	タイムスライス値 (SUPER-UX)	
Nice Value (b)	ナイス値	
Memory Priority (b)	メモリスケジューリングプライオリティ (SUPER-UX)	
Mod factor of CPU (b)	再計算時補正値 (SUPER-UX)	
Tick Count (b)	tick カウンタ (SUPER-UX)	
Decay factor (b)	回復係数 (SUPER-UX)	
Decay interval (b)	回復期間 (SUPER-UX)	
Mrt Size Effect (b)	プロセスサイズの MRT(Memory Resident Time) への影響度 (SUPER-UX)	
Mrt Pri Effect (b)	メモリスケジューリングプライオリティの MRT への影響度 (SUPER-UX)	
Aging Range (b)	エージングレンジ (SUPER-UX)	
Mrt Minimum (b)	MRT の下限値 (SUPER-UX)	
Slave Priority (b)	スレーブプライオリティ (SUPER-UX)	
Scheduling Mode (b)	リクエストスケジューリング方式	
Continuous Scheduling Number (b)	一人のユーザがこのキュー内で連続して実行できるリクエスト数	
Resource occupy wait (b)	同時実行可能資源待ち時間 (SUPER-UX)	
Resource Sharing Group (b)	Resource Sharing Group (RSG)(SUPER-UX)	
Queue server (p, n)	リクエスト転送時、または結果ファイル転送時に使用するサーバのパス名	
ENTRIES	キューに登録されているリクエスト数	
	Total (b, d, p, n)	キューに登録されているリクエスト総数
	Queued (b, d, p, n)	queued 状態のリクエスト数
	Running (b, d, n)	running 状態のリクエスト数
	Routing (p)	routing 状態のリクエスト数

	Waiting (b, d, p, n)	waiting 状態のリクエスト数
	Held (b, d, p)	holding 状態のリクエスト数
	Suspending (b)	suspending 状態のリクエスト数
	Arriving (b, d, p)	arriving 状態のリクエスト数
	Exiting (b)	exiting 状態のリクエスト数
COMPLEX MEMBERSHIP (b)	関連キュー複合体リスト	
RUN LIMITS	キューに課せられた同時実行可能リクエスト制限値	
	Total run limit (b, d, p, n)	キューに課せられた同時実行可能リクエスト数制限値
	FSG0 run limit (b)	同時実行可能ファイルシステムグループ 0 (XMU) 資源総量 (SUPER-UX)
	FSG1 run limit (b)	同時実行可能ファイルシステムグループ 1 資源総量 (SUPER-UX)
	FSG2 run limit (b)	同時実行可能ファイルシステムグループ 2 資源総量 (SUPER-UX)
	FSG3 run limit (b)	同時実行可能ファイルシステムグループ 3 資源総量 (SUPER-UX)
	Memory run limit (b)	同時実行可能メモリサイズ資源総量 (SUPER-UX)
	User run limit (b, p)	ユーザ単位に課せられた同時実行可能リクエスト数制限値
	Group run limit (b, p)	グループ単位に課せられた同時実行可能リクエスト数制限値
DESTINATIONS (p)	リクエスト転送先のキューリスト	
DESTINATION MACHINE (MID) (n)	結果ファイル転送先ホスト名 (マシン ID)	
RESOURCE LIMITS	資源制限名 と資源制限値 資源制限名＝資源制限値の形で表される。 資源標準値使用モードが ON のときは "(S: 資源標準値)" の形で資源標準値が表示される。	
	Per-process (b)	プロセスごとの資源制限名と資源制限値
	Per-request (b)	リクエストごとの資源制限名と資源制限値
ACCESS	アクセス制限状況	
	Route (b, d, p)	pipeonly 属性の有無
	User (b, d, p)	ユーザに関するアクセス制限状況
	Group (b, d, p)	グループに関するアクセス制限状況
ATTRIBUTE	キューに設定されている属性 ON もしくは OFF で表示される。	
	BEFORECHECK (p)	チェック機能をもつキューかどうか
	STAYWAIT (p)	ステイウェイト機能をもつキューかどうか

	FREEDESTINATION (p)	自由転送キューかどうか
	LOADBALANCE (b, p)	デマンドデリバリ負荷分散用キューであるかどうか
	TRANSPARENT (p)	透過型パイプキューであるかどうか
LOAD BALANCING PARAMETER	デマンドデリバリ負荷分散用キューの属性 デマンドデリバリ負荷分散用キューの場合のみ表示される。	
	Keeping request number limit (b)	リクエスト保有数制限値
	Delivery wait time (b)	デマンドデリバリ時のジョブ到着待ち時間
	Reserved run limit (p)	デマンドデリバリ時のリクエスト同時転送数
	Destination retry wait (p)	デマンドデリバリ時のリクエスト待ち時間
CUMULATIVE TIME	積算使用時間	
	System space time (b, d, p, n)	システム領域の積算使用時間
	User space time (b, d, p, n)	ユーザ領域の積算使用時間

4.15.2. オプション

1. 各オプションの説明

-b

対象をバッチキューに限定します。

このオプションを指定するとキュー名指定は無効になります。

-p

対象をパイプキューに限定します。

このオプションを指定するとキュー名指定は無効になります。

-N

対象をネットワークキューに限定します。

このオプションを指定するとキュー名指定は無効になります。

-f

対象キューの詳細情報を表示します。

-L

全バッチキューの資源制限を表示します。

このオプションを指定するとキュー名指定は無効になります。

-n

ヘッダを出力しません。

-h $host-name

$host-name で指定したホスト上のキューを対象にします。

本オプションを指定しなければ現ホストを想定します。

4.16. qstatr　リクエストの状態表示

```
/usr/bin/qstatr [ -a ] [ -b ] [ -c ] [ -d ] [ -N ] [ -R ] [ -f ] [ -n ] [ -t $level | -h $host-name ] [ -u $user-name ] [ $request-id ...]

/usr/bin/qstatr [ -a ] [ -b ] [ -c ] [ -d ] [ -N ] [ -R ] [ -f ] [ -n ] [ -t $level | -h $host-name ] [ -u $user-name ] [ -r $request-name ]
```

4.16.1. 機能説明

qstatr　コマンドはJobCenter　リクエストの状態を表示します。パラメータに　request-id　を指定すると、そのリクエストを対象にして情報が出力されます。また、　request-id　を指定しない場合は、対象ホスト上のコマンド実行者の全リクエストに関する情報が出力されます。本コマンドには JobCenter 管理者にしか使用権を与えていないオプションがありますが、一般ユーザがそのオプションを使用すると警告を発し、情報が表示されません。

1. 表示形式(-fオプションを指定しない場合)

   通常示されるリストの見出しとフィールドの意味を説明します。見出しの後ろに(b)とついているものはバッチリクエストについて、また、(n)とついているものはネットワークリクエストについて示されるものです。

   なお、いずれの場合にもヘッダにはホスト名が表示されます。

   意味の後ろに (SUPER-UX) とあるものについては、 SUPER-UX 上の NQS にリクエストが存在する場合にのみ表示されます。

表4.4 －fオプションを指定しない場合のリストの見出しとフィールドの意味

見出し	フィールドの意味	
REQUEST ID (b, d, n)	リクエスト ID	
NAME (b, d)	リクエスト名	
OWNER (b, d, n)	オーナーの意味	
QUEUE (b, d, n)	現在リクエストが存在しているキュー名	
EVENT (n)	ファイル転送イベント番号	
	29	JOR ファイル転送 (SUPER-UX)
	30	標準出力ファイル転送
	31	WAT (waiting)
PRI (b, d, n)	リクエストプライオリティ	
NICE (b)	ナイス値	
MEMORY (b)	実際に使用されているメモリサイズ (単位 kbytes)(SUPER-UX)	
TIME (b)	使用 CPU 時間 (単位 seconds)(SUPER-UX)	
SIZE (d)	出力サイズ	
STT (b, d, n) 状態	ARI (arriving)	
	HLD (holding)	
	WAT (waiting)	
	QUE (queued)	
	RUT (routing)	
	RUN (running)	
	SUS (suspending)(バッチリクエストのみ)	

	EXT (exiting)(バッチリクエストのみ)	
JID (b)	ジョブ ID (SUPER-UX)	
PGRP (b, n)	リクエストのプロセスグループ ID(バッチリクエストの場合) および結果ファイル転送用サーバ (netclient) のプロセス ID(ネットワークリクエストの場合)	
R (b)	再実行の是非	
	*	再実行リクエスト
	-	非再実行

2. 表示形式(-fオプションを指定した場合)

次に、-fオプションを指定した場合に示されるリストの見出しとフィールドの意味を説明します。見出しの後ろに(b)とついているものはバッチリクエストについて、 (p)とついているものはパイプキュー上のリクエストについて、(n)とついているものはネットワークリクエストについて表示されるものです。

なお、いずれの場合にもヘッダにはリクエストIDが表示されます。

意味の後ろに (SUPER-UX) とあるものについては、 SUPER-UX 上の NQS にリクエストが存在する場合にのみ表示されます。

表4.5 −fオプションを指定した場合のリストの見出しとフィールドの意味

見出し	フィールドの意味	
Name (b, d, p, n)	リクエスト名	
Owner (b, d, p, n)	リクエストオーナー	
Group (b, d, p, n)	リクエストグループ	
State (b, d, p, n)	リクエストの状態	
entered at (b, d, p)	スケジューリング開始日付とスケジューリング開始時間 (実行遅延指定がされている場合のみ)	
Created (b, d, p, n)	リクエスト作成日付とリクエスト作成時間	
Priority (b, d, p, n)	リクエストプライオリティ	
JOB ID (b)	ジョブ ID(Running 状態の時のみ)(SUPER-UX)	
Acctcode (b, d, p)	リクエストのアカウントコード (SUPER-UX)	
Restricted (b)	リクエストが同時実行可能資源総量制限をうけて実行待ちになってから経過した時間(SUPER-UX) (制限をうけていない場合は No restricted、すでに実行が始まっている場合は Alreadyrunning と表示されます)	
Event (n)	ファイル転送イベント番号	
	29	JOR ファイル転送 (SUPER-UX)
	30	標準出力ファイル転送
	31	標準エラー出力ファイル転送
QUEUE	キューに関する設定	
	Name (b, d, p, n)	現在リクエストが存在しているキュー名
STAGING FILE	結果出力ファイルに関する設定	
	Name (n)	結果出力ファイル名リスト
RESOURCES LIMITS	資源制限名と資源制限値	

	資源制限名＝資源制限値の形で表示されます。	
	Per_process (b, p)	プロセスごとの資源制限名と資源制限値
	Per_request (b, p)	リクエストごとの資源制限名と資源制限値
SCHEDULING PARAMETER (b, p)	スケジューリングパラメータ名とスケジューリングパラメータ値	
FILES	結果出力ファイルの種類・スプールモードおよびファイル名 ただしSUPER-UX以外の場合は、リクエスト実行レポートレベルも表示されます。	
	Stdout (b, p)	標準出力ファイルのリクエスト実行レポートレベルとスプールモードとファイル名
	Stderr (b, p)	標準エラー出力ファイルのリクエスト実行レポートレベルとスプールモードとファイル名
MAIL	メールに関する設定	
	Address (b, d, p, n)	メールアドレス
	When (b, d, p)	メールを送信するタイミング
MISC	その他	
	Restartable (b, p)	再実行可能の是非
	User Mask (b, p)	umask 値
	Restartstate (b)	リクエストが JobCenter 立ち上げ時に再実行されたものかどうか
	Orig.Owner (b, d, p, n)	オリジナルマシンのオーナ名
	Shell (b, p)	使用シェルのフルパス名
	JOR (b)	JOR 出力先情報値 (SUPER-UX)
	Checkpoint (b)	チェックポイントがとれるかどうか (SUPER-UX)
	Resource Sharing Group (b)	Resource Sharing Group (RSG) (SUPER-UX)
	Size (d)	出力サイズ
	Copies (d)	コピー数

4.16.2. オプション

1. 各オプションの説明

-b

すべてのバッチリクエストの状態を表示します。

-N

すべてのネットワークリクエストの状態を表示します。

-f

対象リクエストの詳細情報を表示します。 -f オプション指定時には無効となります。

-n

ヘッダを出力しません。

-c

キュー名・リクエスト名・リクエスト ID を省略せず表示します。

-h $host-name

ホスト名で指定したホスト上のリクエストを対象にします。本オプションを指定しなければ現ホストを想定します。

-tオプションと併せて使用することはできません。

-t $level

リクエストのサーチレベルを定義します。

■0

qstatr コマンドを実行したホストあるいは -h オプションで指定したホスト上に存在するリクエストのみを対象とします。リクエスト ID を指定しなかった場合は、このレベルがデフォルトとなります。

■1

リモートホストに転送されたリクエストも表示します。出力される情報は一部省略されますが、レベル 2 よりも高速です。

■2

リモートホストに転送されたリクエストも表示します。また、リクエストの完全な情報を表示します。リクエスト ID を指定した場合は、このレベルがデフォルトとなります。

■3

トラッキングファイルの情報が何らかの理由で欠落していたときに、情報を集め直して正しいトラッキングファイルを作成します。

-hオプションと併せて使用することはできません。

-r

対象リクエストをリクエスト名で指定します。

-u $user-name

対象リクエストを指定ユーザグループに限定します。

JobCenter 管理者だけが利用できます。

-a 現ホスト上のすべてのリクエストを対象とします。 本オプションを指定すると、リクエスト ID 指定および -h オプション指定は無効になります。 JobCenter 管理者だけが利用できます。

2. リクエストIDリスト

リクエスト ID リストは以下のいずれかの形式で指定します。

■シーケンス番号

対象ホスト上で採番された ID に対応するリクエストを指定します。

■シーケンス番号.ホスト名

(リモート)ホスト名に対応するホスト上で採番された ID に対応するリクエストを指定します。

リクエストの状態については qstatq コマンドの説明のリクエスト状態を参照してください。

4.17. qsub　バッチリクエストの投入

```
/usr/bin/qsub [ $options ] [ $script-file ]
```

4.17.1. 機能説明

qsub は、 JobCenter にバッチリクエストを投入します。

1. リクエストの実行

   バッチリクエストは以下のような手順で起動されます。

   a. バッチリクエストを構成するプロセスグループのリーダになるプロセスが、JobCenterにより作られます。

   b. プロセスの実グループIDと実効グループIDが、ローカルパスワードファイルに定義されているリクエストのオーナーのグループIDに設定されます。

   c. プロセスの実ユーザIDと実効ユーザIDが、バッチリクエストのオーナーのユーザIDに設定されます。

   d. ファイル作成マスクが、最初に投入された本来のマシン上にユーザが設定している値に設定されます。

   e. ユーザが -s オプション(後述)でシェルを明確に指定した場合、バッチリクエストスクリプトを実行するシェルとして指定されたシェルが選択されます。その他の場合、そのシェルは ローカル JobCenter システムで設定されたシェル選択方式をもとにして選ばれます (-s オプションの説明を参照)。

   f. あたかもユーザが実行マシンのディレクトリ上にログインしたかのように、ユーザのパスワードファイルエントリから環境変数HOME, SHELL, PATH, LOGNAME(一部のシステム),USER(一部のシステム), MAILが設定されます。

   g. シェルスクリプトが、バッチリクエストの実行かどうか認識できるように環境変数 ENVIRONMENT=BATCH が加えられます。この環境変数により、たとえば、入力端末と関係を持たないバッチリクエストの実行時には、端末文字の設定を行わないようにするというようなことができます。

   h. 環境変数 QSUB_WORKDIR, QSUB_HOST, QSUB_REQNAME, QSUB_REQID が環境変数に加えられます。これらの環境変数は、それぞれリクエストが投入されたときのカレントディレクトリ、投入ホストの名前、リクエスト名、リクエストIDの文字列です。

   i. 投入時の環境変数が環境に加えられます。バッチリクエストが投入されたとき、その時点での環境変数 HOME, SHELL, PATH, LOGNAME, MAIL, TZの値は、この目的のために保存されています。そして、それらが再生されるとき、それぞれ次の環境変数、QSUB_HOME, QSUB_SHELL, QSUB_PATH, QSUB_LOGNAME, QSUB_MAIL, QSUB_TZに設定されます。またこのときに、 -x オプションによって投入時の環境からエクスポートされたすべての環境変数が環境に加えられます。

   j. カレントディレクトリが実行マシン上のユーザのホームディレクトリに設定されます。

   k. 選択されたシェルが、バッチリクエストシェルスクリプトを実行するために execされます。

2. リクエストID

   バッチリクエストが正しく投入されると、リクエストのリクエストIDが表示されます。リクエストIDは常に$seqno.$host-nameの形をとっています。ここでいう$seqnoとはJobCenterによりリクエストに割り当てられたシーケンス番号であり、$host-nameはリクエストを投入したマシンの名前です。

この識別文字列は、リクエストが、たとえネットワーク上にあろうとそれを一意に特定するために使用されるものです。

4.17.2. オプション

1. $options

$options には、qsubコマンドの動作やバッチリクエストの特性を指定するさまざまなオプションを指定します。オプションの簡単な説明を以下に示します。

-a	指定時間後にリクエストを実行します。
-ac	リクエストのアカウントコードを指定されたものにします。
-ds	指定キューへリクエストを転送します。
-e	指定目的地へ標準エラー出力を向けます。
-eo	標準出力の出力先に標準エラー出力を向けます。
-hold	投入した後にリクエストを保留状態にします。
-j	JOR を出力するファイルを指定します。
-je	JOR を結果ファイル (STDERR) に出力させます。
-jm	JOR をメールで発信させます。
-jo	JOR を結果ファイル (STDOUT) に出力させます。
-ke	リクエストが実行されたマシン上に標準エラー出力による結果ファイル置きます。
-ko	リクエストが実行されたマシン上に標準出力による結果ファイルを置きます。
-l0	リクエストごとのファイルシステムグループ 0 (FSG0=XMU) 制限値を設定します。
-l1	リクエストごとのファイルシステムグループ 1 (FSG1) 制限値を設定します。
-l2	リクエストごとのファイルシステムグループ 2 (FSG2) 制限値を設定します。
-l3	リクエストごとのファイルシステムグループ 3 (FSG3) 制限値を設定します。
-lc	プロセスごとのコアファイルサイズ制限を設定します。
-ld	プロセスごとのデータセグメントサイズ制限を設定します。
-lD	リクエストごとのテープ装置台数制限を設定します。
-lf	プロセスごとの永久ファイルサイズ制限を設定します。
-lm	プロセスごとのメモリサイズ制限を設定します。
-lM	リクエストごとのメモリサイズ制限を設定します。
-ln	プロセスごとのナイス実行制限値を実行します。
-lo	プロセスごとの同時オープンファイル数制限を設定します。
-lO	リクエストごとの同時オープンファイル数制限を設定します。
-lP	リクエストごとのプロセス数制限を設定します。
-ls	プロセスごとのスタックセグメントサイズ制限を設定します。
-lt	プロセスごとの CPU 時間制限を設定します。
-lT	リクエストごとの CPU 時間制限を設定します。
-lu	プロセスごとの CPU 台数制限を設定します。
-lU	リクエストごとの CPU 台数制限を設定します。
-lV	リクエストごとの一時ファイルサイズ制限を設定します。
-lw	プロセスごとのワーキングセット制限を設定します。

-lW	リクエストごとの物理メモリ制限、および事前確保値を設定します。
-lx	プロセスごとの永久ファイル容量制限を設定します。
-lX	リクエストごとの永久ファイル容量制限を設定します。
-mb	リクエストの実行が始められたときにメールが送られます。
-me	リクエストの実行が終わったときにメールが送られます。
-mu	指定ユーザへリクエストにより送信されるメールが送られます。
-nc	バッチリクエストがチェックポイント採取不可であることを指定します。
-nr	バッチリクエストが再実行不可であることを指定します。
-o	指定目的地へ標準出力を向けます。
-p	キュー内でのリクエストプライオリティを指定します。
-q	指定キューへリクエストを登録します。
-r	リクエスト名を割り当てます。
-re	標準エラー出力をスプールすることなく結果ファイルに出力します。
-ro	標準出力をスプールすることなく結果ファイルに出力します。
-s	シェルスクリプトを解釈するシェルを指定します。
-sr	ジョブステップリスタート機能を使用します。
-v	リクエスト実行レポートを、標準出力と標準エラー出力に出力します。
-ve	リクエスト実行レポートを、標準エラー出力に出力します。
-vo	リクエスト実行レポートを、標準出力に出力します。
-x	リクエストのすべての環境変数を export します。
-z	リクエストを出力なしで投入します。
-Z	リクエスト投入成功時にリクエスト ID のみを出力します。

2. $script-file

   $script-fileにはバッチリクエストとして実行するシェルスクリプトファイルのファイル名を指定します。このスクリプトファイルを指定しなかった場合は、標準入力からスクリプトが読み込まれます。スクリプトはすぐにファイルとしてスプールされるので、スクリプトファイルを後から変更しても、投入されたバッチリクエストには影響しません。

3. 埋め込みオプション

   qsub のコマンドラインで指定するオプションは、埋め込みオプションとしてスクリプトファイル中に記述することもできます。もし、コマンド行に "埋め込みオプション "で指定したものと同じオプションを指定した場合は、コマンド行の方が優先されます。

   埋め込みオプションは "#"で始まる行の中に置きます。 "#"で始まる行はたいていのシェルにとってコメント行となるため、コマンドとして実行されることはありません。埋め込みオプションが置ける範囲は、スクリプトファイルの先頭から、初めてコメント行以外の行が現れるまでです。

   コメント行の "#"に続けて文字列 "@$"を置きます。 "@$"を書かなければ、単なるコメント行と解釈されます。そしてこの "@$"の後に文字 "− "の付いたオプション記述を置きます。オプションはコマンドラインに書くのと同じようにして 1 行に複数個指定できます。また、オプション記述の後に文字 "#"を置くとそこから行末まではコメントになります。

   "#"の前後には空白を入れることができます。 "@$"の後には空白を置くことはできません。文字列 "@$"の後の文字が "-"でない場合、 qsub は埋め込みオプションの解釈をそこで終了します。

埋め込みオプションの引き数が空白文字や文字　　"#"を含む場合には、シェルと同じようにクォートする必要があります。

コメント行以外の行であっても空行および文字 ":"で始まる行は無視するので、これらの後にも埋め込みオプションを置くことができます。

スクリプト内での "埋め込みオプション "の使用例を示します。

```
#
# Batch request script example:
#
# @$-a "11:30pm EDT" -lt "21:10,20:00"
#           # Run request after 11:30 EDT by default,
#           # and set a maximum per-process CPU time
#           # limit of 21 minutes and ten seconds.
#           # Send a warning signal when any process
#           # of the running batch request consumes
#           # more than 20 minutes of CPU time.
# @$-lT 1:45:00
#           # Set a maximum per-request CPU time limit
#           # of one hour, and 45 minutes. (The
#           # implementation of CPU time limits is
#           # completely dependent upon the UNIX
#           # implementation at the execution
#           # machine.)
# @$-mb -me    # Send mail at beginning aand end of
#           # request execution.
# @$-q  batch1 # Queue request to queue: batch1 by
#           # default.
# @$        # No more embedded flags.
#
make all
```

4. オプション詳細説明

-a $date-time

指定された日時までバッチリクエストの実行を遅延します。コマンド行上に空白で区切られた$date-timeを指定する場合は、 -a "July 4,2026 12:31-EDT " のようにダブルクォートで明確に囲むか、qsub　とシェルが$date-time指定を1つの文字列として解釈できるようにエスケープする必要があります。このことは、　"埋め込みオプション"で$date-timeを指定する場合にもいえることです。

$date-timeに指定する構文規則は比較的自由なものです。もし、日付や時間を指定しなかった場合は、既定値として適当なもの (たとえば、日付を指定しなかったら現在の月、日、年) が割り当てられます。

日付は、月と日で指定します。この場合、年は現在のものがとられますが、年も明示的に指定することができます。また、曜日 (たとえば、 Tues) や単語 today や tomorrowのような日付を指定することもできます。曜日や月の名前は 3 文字までに短縮してもかまいませんし、その短縮した文字列にピリオドを付加してもかまいません。

時間の指定は、　24　時間指定か　"am",　"pm"指定のどちらかを使って行います。もし"am", "pm"の指定をしなかった場合は、 24 時間指定と見なされます。

24　時間指定でいう　0:00:00,　12:00:00,　24:00:00　を　"am",　"pm"指定で表すと、"12am","12m ", "12-pm "になります。また "midnight"や "noon"という熟語も時間指定として用いることができます。ここでいう "midnight"は 24:00:00のことです。

また、時間帯を$date-time指定の任意の位置に指定することができます。たとえば"April 1, 1987 13:01-PDT"のように指定します。もし時間帯を指定しなければ、指定日付をもとに夏時間を考慮して、現地方標準時がとられます。

アルファベットの比較は厳密には行われません。つまり、 "WeD "と "weD "は両方とも "水曜日 "と解釈されます。いくつかの有効な$date-time指定例を以下に示します。

01-Jan-1986 12am,PDT

Tuesday,23:00:00

11pm tues.

tomorrow 23-MST

-ac $acct-code

リクエストのアカウントコードを$acct-codeで指定されたものにします。指定するアカウントコードはリクエストの投入者が使用できるものでなければいけません。このオプションが指 定されなかった場合はリクエストを投入したプロセスのアカウントコードをリクエストのアカウントコードとします。

本オプションは、 SUPER-UX 上で実行されるリクエストに対してのみ有効です。

-ds $queue[@$machine]

指定されたキューにリクエストを転送します。

自由転送先パイプキューにリクエストが投入された場合、$queue@$machine で示されたキューにリクエストが転送されます。 @$machineを省略した場合は、実行したマシンのキューを示します。

本オプションは、リクエストを自由転送先パイプキューに直接投入する場合以外には効果がありません。転送先が示された場合、 qstat -xまたは qs- tatr -f のコマンドの Destination の項目に転送先キュー名が表示されます。

このオプションを指定せずに自由転送先パイプキューにリクエストを投入した場合、自由転送先パイプキューは通常のパイプキューと同様にそのリクエストを処理します。

このオプションで指定された転送先にリクエストが転送できなかった場合、パイプキューに設定されている転送先に対して順次転送を試みます。

-e [$machine:][[/]$path/]$stderr-filename

バッチリクエストの標準エラー出力を、指定したファイルに向けます。[] で囲まれた部分は省略可能です。

machine を指定していなければ、 -ke オプションの指定の有無により、バッチリクエストを投入したマシンか、実際に実行が行われるマシンが選ばれます。

path が "/"で始まっていなければ、カレントディレクトリからの相対パス名と解釈します。ただし、目的マシンが qsub マシンと異なる場合は目的マシン上のホームディレクトリからの相対パス名と解釈します。

-eo オプションが指定されたときは、このオプションは指定できません。

-e オプション指定されていない場合の既定値は、リクエスト名の最初の 7 文字、文字列 ".e"そして、リクエスト ID のシーケンス番号の部分をつなげたものとなります。

-eo

バッチリクエストの標準エラー出力を標準出力と同じファイルに向けます。このオプションは、 -e オプションが指定されたときは指定できません。

-hold

リクエストを投入した後に、そのリクエストをホールド状態にします。

-j [$machine:][[/]$path/]$jor-filename

バッチリクエストの JOR(ジョブオカレンスレポート) を指定したファイルに出力させます。-je, -jm, -jo と同時に指定することはできません。

このオプションはSUPER-UX上で実行されるリクエストに対してのみ有効です。それ以外はエラーになります。

-je

バッチリクエストの JOR(ジョブオカレンスレポート) を結果ファイル (STDERR)に出力させます。-j, -jm, -jo と同時に指定することはできません。

このオプションは SUPER-UX 上で実行されるリクエストに対してのみ有効です。それ以外はエラーになります。

-jm

バッチリクエストの JOR(ジョブオカレンスレポート) をメールで送信させます。-j, -je, -jo と同時に指定することはできません。

このオプションは SUPER-UX 上で実行されるリクエストに対してのみ有効です。それ以外はエラーになります。

-jo

バッチリクエストの JOR(ジョブオカレンスレポート) を結果ファイル (STDOUT)に出力させます。-j, -je, -jm と同時に指定することはできません。

このオプションは SUPER-UX 上で実行されるリクエストに対してのみ有効です。それ以外はエラーになります。

-ke

標準エラー結果ファイルを作成するマシンを、実際にバッチリクエストが実行されたマシンにします。このオプションを指定しなかった場合は、バッチリクエストが投入された本来のマシンになります。

このオプションは、 -eo オプションが指定されていたり、 -e オプションにマシン指定が与えられている場合は無効になります。

-ko

標準出力結果ファイルを作成するマシンを、実際にバッチリクエストが実行されたマシンにします。このオプションを指定しなかった場合は、バッチリクエストが投入された本来のマシンになります。

このオプションは、 -o オプションにマシン指定が与えられている場合は無効になります。

-l0 $max-limit[,$warn-limit]

バッチリクエストに現在設定されている全プロセスに対するリクエストごとのファイルシステムグループ 0 (FSG0 = XMU) 制限の最大値や警告値をセットします。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。　もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については制限の項を参照してください。

-l1 $max-limit[,$warn-limit]

バッチリクエストに現在設定されている全プロセスに対するリクエストごとのファイルシステムグループ 1 (FSG1) 制限の最大値や警告値をセットします。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については制限の項を参照してください。

-l2 $max-limit[,$warn-limit]

バッチリクエストに現在設定されている全プロセスに対するリクエストごとのファイルシステムグループ 2 (FSG2) 制限の最大値や警告値をセットします。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については制限の項を参照してください。

-l3 $max-limit[,$warn-limit]

バッチリクエストに現在設定されている全プロセスに対するリクエストごとのファイルシステムグループ 3 (FSG3) 制限の最大値や警告値をセットします。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については制限の項を参照してください。

-lc $max-limit

バッチリクエストの全プロセスに対するプロセスごとのコアファイルサイズ制限の最大値をセットします。もしリクエストのプロセスがこの制限値を超えるファイルを作成しようとして終了した場合、そのコアイメージは、もととなる UNIX インプリメンテーションのアルゴリズムにより必要なサイズに縮小されます。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については資源制限の項を参照してください。

-ld $max-limit[,$warn-limit]

バッチリクエストの全プロセスに対するプロセスごとのデータセグメントサイズ制限の最大値や警告値をセットします。リクエストのプロセスがリクエストに課せられた、この制限の最大値を超えようとした場合、そのプロセスは、もととなる UNIX インプリメンテーションにより選択されたシグナルで終了します。

プロセスごとのデータセグメントサイズ制限の警告をサポートしている UNIX オペレーティングシステムであれば、警告制限の指定をすることができます。警告制限を超えたとき、もととなる UNIX インプリメンテーションによりシグナルが発行され、制限を超えたプロセスに送られます。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

本システムでは最大制限をサポートしています。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については資源制限の項を参照してください。

-ID $max-limit

バッチリクエストの全プロセスに対するリクエストごとの テープ装置台数制限の最大値をセットします。 リクエストのプロセスがリクエストに課せられたこの制限の最大値を超えよう とした場合、 そのプロセスは、もととなる UNIX インプリメンテーションにより選択されたシグナルで終了します。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については制限の項を参照してください。

-lf $max-limit[,$warn-limit]

バッチリクエストの全プロセスに対するプロセスごとの永久ファイルサイズ制限の最大値や警告値をセットします。リクエストのプロセスがリクエストに課せられた制限を超える永久ファイルに書き込もうとした場合、そのプロセスは、もととなる UNIXインプリメンテーションにより選択されたシグナルで終了します。

プロセスごとの永久ファイルサイズ制限の警告をサポートしている UNIXオペレーティングシステムであれば、警告制限の指定をすることができます。警告制限を超えたとき、もととなる UNIX インプリメンテーションにより選択されたシグナルが、制限を破ったプロセスに送られます。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

現在、本システムではカーネルにおいて永久ファイルと一時ファイルを区別するようなメカニズムはサポートされていません。つまり、厳密な意味での永久ファイルと一時ファイルの区別はできないということです。したがって本システムでは、この制限は単なるプロセスごとのファイルサイズ制限として解釈してください。

本システムでは最大制限をサポートしています。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については資源制限の項を参照してください。

-lm $max-limit[,$warn-limit]

バッチリクエストの全プロセスに対するプロセスごとのメモリサイズ制限の最大値や警告値をセットします。リクエストのプロセスがリクエストに課せられた制限を超えようとした場合、そのプロセスは、もととなる UNIX インプリメンテーションにより選択されたシグナルで終了します。

プロセスごとのメモリサイズ制限の警告をサポートしている UNIX オペレーティングシステムであれば、警告制限の指定をすることができます。警告制限を超えたとき、もととなる UNIX インプリメンテーションにより決定されたシグナルが、制限を破ったプロセスに送られます。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については資源制限の項を参照してください。

-lM $max-limit[,$warn-limit]

バッチリクエストの全プロセスに対するリクエストごとのメモリサイズ制限の最大値や警告値をセットします。リクエストのプロセスにより費やされるメモリの合計がリクエストに課せられた制限を超えようとした場合、リクエストのプロセスは、もととなるUNIX インプリメンテーションにより選択されたシグナルで終了します。

プロセスごとのメモリサイズ制限の警告をサポートしている UNIX オペレーティングシステムであれば、警告制限の指定をすることができます。警告制限を超えたとき、もととなる UNIX インプリメンテーションにより決定されたシグナルが、制限を破ったプロセスに送られます。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

本システムではこの制限をサポートしていません。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については資源制限の項を参照してください。

-ln $value

バッチリクエストの全プロセスに関するプロセスごとの nice 値をセットします。

現在本システムでは、nice 値と呼ばれるものの使用がサポートされており、これは、システム内のほかのすべてのプロセスに対するプロセスの実行時間優先度を調節しています。ユーザに、リクエストのプロセスの nice 値を指定させることにより、そのリクエストが実行される優先順位を低く（または高く）することができます。

このオプションで指定する nice 値が負に大きくなれば、プロセスの実行優先度は高くなり、正に大きくなれば、優先度は低くなります。たとえば、同一のベースプライオリティを持ったリクエストどうしでは nice-value "-ln -10 " は "-ln 0 " より CPU 資源を消費することになります。

いろいろな UNIX インプリメンテーションの nice 値の範囲はそれぞれ異なるためJobCenterは実行マシン上でその範囲を超えることになった場合、不正にはならず、指定された nice 値を可能な範囲内の値に束縛します。

このオプションを使って指定された nice 値は、リクエストが実行されるマシンで解釈される値です。

-lo $max-limit

バッチリクエストの全プロセスに対するプロセスごとの同時オープンファイル数制限の最大値をセットします。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報 や limit の厳密な構文規則については制限の項を参照してください。

-lO $max-limit

バッチリクエストの全プロセスに対するリクエストごとの同時オープンファイル数制限の最大値をセットします。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報 や limit の厳密な構文規則については制限の項を参照してください。

-ls $max-limit[,$warn-limit]

バッチリクエストの全プロセスに対するプロセスごとのスタックセグメントサイズ制限の最大値や警告値をセットします。リクエストのプロセスがリクエストに課せられた制限を超えようとした場合、そのプロセスは、もととなる UNIX インプリメンテーションにより選択されたシグナルで終了します。

プロセスごとのスタックセグメントサイズ制限の警告をサポートしている UNIXオペレーティングシステムであれば、警告制限の指定をすることができます。警告制限を超えたとき、もととなる UNIX インプリメンテーションにより選ばれたシグナルが、制限を破ったプロセスに送られます。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については資源制限の項を参照してください。

-lt $max-limit[,$warn-limit]

バッチリクエストの全プロセスに対するプロセスごとの CPU 時間制限の最大値や警告値をセットします。リクエストのプロセスがリクエストに課せられた制限を超えようとした場合、そのプロセスは、もととなる UNIX インプリメンテーションにより選択されたシグナルで終了します。

プロセスごとの CPU 時間制限の警告をサポートしている UNIX オペレーティングシステムであれば、警告制限の指定をすることができます。警告制限を超えたとき、もととなる UNIX インプリメンテーションにより選ばれたシグナルが、制限を破ったプロセスに送られます。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については資源制限の項を参照してください。

-lu $max-limit

バッチリクエストの全プロセスに対するプロセスごとの CPU 台数制限の最大値をセットします。リクエストのプロセスがリクエストに課せられた制限を超えようとした場合、そのプロセスは、もととなる UNIX インプリメンテーションにより選択されたシグナルで終了します。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については資源制限の項を参照してください。

-lU $max-limit

バッチリクエストの全プロセスに対するリクエストごとの CPU 台数制限の最大値をセットします。リクエストのプロセスがリクエストに課せられた制限を超えようとした場合、そのプロセスは、もととなる UNIX インプリメンテーションにより選択されたシグナルで終了します。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については資源制限の項を参照してください。

-lw $max-limit

バッチリクエストの実行に関係するすべてのプロセスに対するプロセスごとのワーキングセットサイズ制限の最大値を設定します。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

本システムではこの制限をサポートしていません。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については資源制限の項を参照してください。

-lx $max-limit[,$warn-limit]

バッチリクエストの全プロセスに対するプロセスごとの永久ファイル容量制限の最大値をセットします。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については制限の項を参照してください。

-lX $max-limit[,$warn-limit]

バッチリクエストの全プロセスに対するリクエストごとの永久ファイル容量制限の最大値をセットします。

すべてのUNIXインプリメンテーションがこの制限をサポートしているとは限りません。もし、バッチリクエストにこの制限を指定して、この制限をサポートしていないマシン上で実行された場合、その制限は単に無視されるだけです。

バッチリクエスト制限の実行に関する情報や limit の厳密な構文規則については制限の項を参照してください。

-mb

リクエストが実行を開始したとき、本来のマシンのユーザへメールを送ります。

-muオプションも指定されていれば、メールはリクエスト投入ユーザの代わりに-mu によって指定されたユーザへ送られます。

-me

リクエストが実行を終了したとき、本来のマシンのユーザへメールを送ります。

-muオプションも指定されていれば、メールはリクエスト投入ユーザの代わりに -mu によって指定されたユーザへ送られます。

-mu $user-name

リクエストに関するメールが届けられるユーザの名前を指定します。

$user-name はuser(@ 文字を含まない) か、 user@machine のどちらかの形式です。このオプションが省略された場合は、リクエストに関係するメールは本来のマシンのリクエスト投入ユーザに送られます。

-nc

リクエストがチェックポイント採取不可であることを宣言します。 このフラグを指定すると、リクエストのチェックポイントを採取できなくなります。

このオプションは、 SUPER-UX 上で実行されるリクエストに対してのみ有効です。

それ以外はエラーとなります。

-nr

リクエストが再起動不可であることを宣言します。このオプションを指定すると、たとえリクエストがシステムシャットダウンやシステムクラッシュのときに実行されていてもシステムブート時に再起動されなくなります。

既定では、すべてのリクエストは再起動可能とされています。この場合、再起動で正常に実行されるかどうかは、適当なプログラミング技術により実行者が保証しなければならないことに注意してください。

ホストクラッシュやシャットダウンの時に動いていなかったリクエストは常に、このオプションの有無にかかわらず後で再びキューに投入されます。

JobCenter がJobCenter コントロールプログラム qmgr(1M) で終了されたとき、SIGTERM シグナルがローカルホスト上で実行しているすべてのJobCenter リクエストのプロセスに送られます。またそれと同時に、登録されているJobCenterリクエストの実行の開始も差し止められます。そして事前に決められた時間が経過した後（通常は 60 秒ですが、この値はオペレータが変更することができます) で、実行中の JobCenter リクエストのプロセスは SIGKILL によって終了させられます。

JobCenter リクエストがJobCenter シャットダウンの後で再起動されるためには、 -nr オプションを指定しないで、しかも起動されたバッチリクエストシェルが SIGTERMシグナルを無視しなければなりません（既定によって行われます)。また、起動されたバッチリクエストシェルは最後の SIGKILL を受信する前に終了してはいけません。

バッチリクエストシェルは単にコマンドおよびプログラムを起動して、その完了を待っているだけです。したがって、バッチリクエストシェルにより実行されるコマンドおよびプログラムもまた、その現状態を保存するために SIGKILL シグナルにより killされるまで、 SIGTERM シグナルを無視しなければなりません。

JobCenter バッチリクエストの再起動に関しては、後述の注意の節を参照してください。

-o [$machine:][[/]$path/]$stdout-fileame

バッチリクエストの標準出力を、指定したファイルに向けます。[ ] で囲まれた部分は省略可能です。

$machineを指定していなければ、 -ko オプションの指定の有無により、バッチリクエストを投入したマシンか、実際に実行が行われるマシンが選ばれます。

$path が "/"で始まっていなければ、カレントディレクトリからの相対パス名と解釈します。ただし、目的マシンがqsubマシンと異なる場合は目的マシン上のホームディレクトリからの相対パス名と解釈します。

-o オプション指定されていない場合の既定値は、リクエスト名の最初の 7 文字、文字列".o"そして、リクエスト ID のシーケンス番号の部分をつなげたものとなります。

-p $priority

リクエストのキュー内部優先度を割り当てます。指定された$priorityは、 [0...63] の範囲の整数でなければなりません。 63 という値はキュー内で最も高いリクエスト優先度を定義し、それに対し 0 という値は最も低い優先度を定義します。この優先度はリクエストの実行優先度を決定しているわけでなく、ただキュー内のリクエストの相対的な順番を決定するのに用いられるだけです。

リクエストがキューに登録されると、新しく投入されたリクエストはその優先度より低い優先度をもつリクエストより前に位置づけられます。同様に、より高い優先度をもつリクエストは、新しいリクエストの前に位置づけられます。新しいリクエストの優先度がすでに存在していたリクエストの優先度と同じ場合は、新しいリクエストより、存在していたリクエストのほうが先行します。

ユーザがキュー内部優先度を指定しなかったら、JobCenter が既定値を割り当てます。

-q $queue

バッチリクエストを登録するキューを指定します。本オプションを指定しなければ、そのユーザの環境変数からQSUB_QUEUEという変数が捜されます。もし、この環境変数が見つかったら、QSUB_QUEUEの文字列値が、リクエストを登録すべきキューの名前と仮定されます。しかし、QSUB_QUEUE環境変数が見つからなかったときは、システム管理者によって定義された既定バッチリクエストキューに登録されます。

既定バッチキューが定義されていなければ、リクエストはキューに登録されることなく、適切なエラーメッセージが出力されます。

-r $request-name

リクエストへ指定された$request-nameをリクエスト名として割り当てます。このオプションを指定しなければ、コマンド行で与えられたスクリプトファイル (先頭のパス名を除く) の名前がとられます。またスクリプトを指定しない場合は、リクエストに割り当てられる既定リクエスト名は、 "STDIN"になります。

もし$request-nameが数字で始まっていると、名前が数字で始まらないように"R"文字が先頭に付加されます。また$request-nameは最大 63 文字で切られます。

-re

既定では、リクエストによって作られるすべての標準エラー出力は、一時的にリクエストが実行されるマシン上の決まったディレクトリにファイルとして保存されます。そして、バッチリクエストが実行を完了したとき、このファイルは、その最終目的地 (リモートマシン上の場合もある) へ転送されます。

この既定 stderr 出力スプーリングは、実行が完了した上で標準エラー出力をリモートマシンに返すといったような手順を望む投入者 (オーナ) にとってネットワークトラフィック費用の削減になります。

しかし、この転送形態を要望しない場合は、この -reオプションを指定します。このオプションを指定するとリクエストにより出される標準エラー出力が生じるのと同時に最終目的ファイルに書き込まれるようになります。

JobCenter インプリメンテーションが、このオプションをサポートしていない場合、それは無視され、標準エラー出力は単にこのオプションがないときのように転送されます。

本システムでは、非スプールモードの結果ファイル転送はローカルマシンに閉じた場合だけ可能です。

-ro

既定では、リクエストによって作られるすべての標準出力は、一時的に、リクエストが実行されるマシン上の決まったディレクトリにファイルとして保存されます。そして、バッチリクエストが実行を完了したとき、このファイルは、その最終目的地 (リモートマシン上の場合もある) へ転送されます。

この既定 stdout 出力スプーリングは、実行が完了した上で標準出力をリモートマシンに返すような手順を望む投入者 (オーナ) にとってネットワークトラフィック費用の削減になります。

しかし、この転送形態を要望しない場合は、この -ro オプションを指定します。このオプションを指定するとリクエストにより出される標準出力が生じると同時に最終目的ファイルに書き込まれるようになります。

JobCenter インプリメンテーションが、このようなオプションをサポートしていない場合、それは無視され、標準出力は単にこのオプションがないときのように転送されます。

本システムでは、非スプールモードの結果ファイル転送はローカルマシンに閉じた場合だけ可能です。

-s $shell-name

バッチリクエストのシェルスクリプトを実行するシェルの絶対パス名を指定します。

このオプションを省略した場合は、システムに設定された方式でシェルが選択されます。シェルの選択方式は qlimit(1) コマンドで確認することができます。シェル選択方式には以下の3通りあります。

■fixed

バッチリクエストを実行するシェルとして管理者により指定されたシェルが使用されます。

■free

バッチリクエストを実行する際に、まずリクエストのユーザのログインシェルが起動されます。次にそのログインシェルが、バッチリクエストの内容から適切なシェルを選択し、そのシェルがバッチリクエストを実行します。あたかも、インタラクティブな処理と同様な形態でバッチリクエストが実行されます。

■login

バッチリクエストを実行するシェルとして、リクエストのユーザのログインシェルが使用されます。

-sr

ジョブステップリスタート機能を使用します。

チェックポイントとしてジョブスクリプト内に記述されたコメント行を解釈します。

本オプションの指定がない場合、チェックポイントのコメント行は単なるコメントとして無視されます。その場合、ジョブステップリスタート機能は動作しません。

ジョブステップリスタート機能が使用するチェックポイントは SUPER-UX で実装されているチェックポイントとは別のものです。したがって -nc オプションの影響を受けません。

本機能は SUPER-UX 版では実装されていません。

-v $level

リクエスト実行レポートを、リクエスト終了時に、リクエストの標準出力と標準エラー出力に出力します。

リクエストのプロセスは、フロックという単位で管理されており、リクエスト実行レポートはフロックの実行レポートとして出力されます。

ただし、 SUPER-UX ではフロックはサポートされていないため、このオプションはSUPER-UX上で実行されるリクエストに対して指定することはできません。

リクエスト実行レポートの出力内容は$levelによって設定します。

[level]

■1

フロック開始時刻、フロック終了時刻、フロック終了ステータス、フロックタイプ、フロック ID を出力します。

■2

レベル 1 の全出力内容に加えて、テンポラリファィルのアサイン状況、テンポラリファイルのディアサイン状況を出力します。

-ve $level

リクエスト実行レポートを、リクエスト終了時に、リクエストの標準エラー出力に出力します。

リクエストのプロセスは、フロックという単位で管理されており、リクエスト実行レポートはフロックの実行レポートとして出力されます。

ただし、 SUPER-UX ではフロックはサポートされていないため、このオプションはSUPER-UX上で実行されるリクエストに対して指定することはできません。

リクエスト実行レポートの出力内容は$levelによって設定します。

[level]

■1

フロック開始時刻、フロック終了時刻、フロック終了ステータス、フロックタイプ、フロック ID を出力します。

■2

レベル 1 の全出力内容に加えて、テンポラリファィルのアサイン状況、テンポラリファイルのディアサイン状況を出力します。

-vo $level

リクエスト実行レポートを、リクエスト終了時に、リクエストの標準出力に出力します。

リクエストのプロセスは、フロックという単位で管理されており、リクエスト実行レポートはフロックの実行レポートとして出力されます。

ただし、 SUPER-UX ではフロックはサポートされていないため、このオプションはSUPER-UX上で実行されるリクエストに対して指定することはできません。

リクエスト実行レポートの出力内容は$levelによって設定します。

[level]

■1

フロック開始時刻、フロック終了時刻、フロック終了ステータス、フロックタイプ、フロック ID を出力します。

■2

レベル 1 の全出力内容に加えて、テンポラリファィルのアサイン状況、テンポラリファイルのディアサイン状況を出力します。

-x

リクエスト投入時のすべての環境変数をリクエスト実行環境にエクスポートします。

バッチリクエストを投入するとき、その時点の環境変数 HOME, SHELL, PATH, LOGNAME(一部のシステム), USER(一部のシステム), MAIL, TZ の値は保存され、バッチリクエ

ストが起動されるときに、それぞれ環境変数　QSUB_HOME,　QSUB_SHELL,　QSUB_PATH, QSUB_LOGNAME, QSUB_USER, QSUB_MAIL, QSUB_TZ として再生されます。

もし -x オプションを指定しなければ、ほかの環境変数はエクスポートされません。しかし、 -x オプションを指定すると、自動的にエクスポートされた変数 (HOME など) の名前と重ならない残りの環境変数もまた、エクスポートされます。

追加した環境変数は、バッチリクエストが起動されたときに同じ名前で再生されます。

-z

メッセージの表示を抑止します。リクエストが正しく投入されても、それを示すメッセージは表示されなくなります。ただしエラーメッセージは常に表示されます。

-Z

バッチリクエストの投入に成功した場合、リクエスト ID のみを出力するようにします。 リクエスト　ID　以外のメッセージは表示されなくなります。ただし、エラーメッセージは表示されます。

5. 資源制限

資源制限オプション(-l)で指定する制限値の形式は以下のとおりです。

a. a) 時間制限

時間に関する制限は次の形式で指定します。

```
[[hours:]minutes:]seconds[.fraction]
```

無制限の場合は“unlimited”という文字列を指定します。以下は指定例です。

単位記述	単　位
1234:58:21.29	1234時間58分21.29秒
59:01	59分1秒
12345	12345秒
121.1	121.1秒

b. サイズ制限

サイズに関する制限は次の形式で指定します。

```
integer[.fraction][units]
```

units に指定できる単位は次のとおりです。

単位記述	単　位
b	バイト
kb	キロバイト(1024 b)
mb	メガバイト(1024 kb)
gb	ギガバイト(1024 mb)

unitsを指定しなかった場合はバイトと解釈されます。無制限の場合は“unlimited”という文字列を指定します。制限値はマシンにとって都合のよい値に変換される場合があります。以下は指定例です。

1234	1234 バイト
1234kb	1234 キロバイト
1234.5gb	1234.5 ギガバイト

c. ナイス値

ナイス値は直接数値を指定します。

リクエストのすべての制限値が、キューの対応する制限値を超えていない場合にのみ、そのリクエストはキューにつながれます。リクエストに無制限値がある場合は、キューの対応する制限値もまた無制限でなければなりません。

リクエストに指定されていない制限値については、キューの対応する制限値がリクエストの制限値となります。

1度リクエストをキューにつないだ後は、キューの制限値が変更されても、リクエストの制限値は変更されません。

本システムではプロセスごとのコアファイルサイズ制限、プロセスごとのデータセグメントサイズ制限、プロセスごとの永久ファイルサイズ制限、プロセスごとのメモリサイズ制限、プロセスごとのナイス実行値、プロセスごとのスタックセグメントサイズ制限、プロセスごとのCPU時間制限は　setrlimit(2)を使用しています。プロセスごとの各種資源制限に対する具体的な動作については setrlimit(2) を参照してください。

4.17.3. 注意事項

1. 制限事項およびインプリメンテーション留意点

■現在のインプリメンテーションでは、 -re と -ro を明示的に指定しない限り、リクエストの実行中に stderr や stdout ファイルを見ることはできません。

■"埋め込みオプション "を埋め込むときに使う "@$"という構文規則は、シェルスクリプトファイル中にあるものとめったに重ならないものとして選ばれたものです。

■リクエスト実行開始時に、.profile(sh および ksh 時) や.login(csh 時) は実行されません。

2. その他注意事項

■JobCenterは、qmgr(1M)のabort queue コマンドや、シャットダウンを実行する場合、実行中のリクエストを強制終了する前に、警告としてバッチリクエストのすべてのプロセスに SIGTERM シグナルを送ります。

■起動されたシェルは、　SIGTERM　シグナルを無視します。もしシェルの現在の直接の子が、SIGTERM シグナルを無視せずにつかまえた場合、この受信によって強制終了され、シェルはスクリプトから次のコマンドを実行します。また、実行中のコマンドが強制終了した場合でも、シェルは SIGTERM シグナルによって強制終了はしません。

■JobCenter から受信された SIGTERM シグナルの受信後、バッチリクエストのプロセスには SIGKILL シグナルを受け取るまでに、その "状態保存 "をするために通常 60 秒の猶予が与えられます (オペレータは、猶予時間を変更することができます)。

■-nr オプションの指定がなく、オペレータの JobCenter シャットダウン要求により終了させられたバッチリクエストは、再起動可能とみなされ、再びキューにつながれます (上述したような SIGKILL シグナルの発行時に、依然バッチリクエストシェルプロセスが存在していたリクエストに限る)。その結果　JobCenter　が再立ち上げされたとき、そのようなリクエストが実行を続けるために再起動されます。ただし、リクエストは再起動可能な構造で記述されていなければなりませんJobCenterは、最初に起動されたとおりに、再び起動するだけです。

■バッチリクエストの完了に伴い、その投入者に mail メッセージを送ることができます(-meオプションの説明を参照)。多くの場合、起動されたボーンシェルや C- シェルの終了コードが示されます。これは、単に exit(2) システムコールを通してシェルが返した値です。

4.17.4. 関連項目

qdel(1), qlimit(1), qstat(1), qmgr(1M).

mail(1), kill(1), setpgrp(2), signal(2).

4.18. qwait　リクエスト終了の待ち合わせ

```
/usr/bin/qwait [ -t $timeout ] [ -f ] $request-id

/usr/bin/qwait [ -t $timeout ] [ -f ] -r $request-name
```

4.18.1. 機能説明

qwait は、リクエスト終了の待ち合わせを行うコマンドです。 qwait コマンドが起動されると、リクエストが終了するまでスリープし、リクエストが終了するとメッセージを表示して終了します。

1. 終了コード

以下の表は、qwaitの終了コードと出力を表したものです。

表4.6 qwaitの終了コードと出力

終了コード	qwaitの出力	意　味
0	done + リクエストの終了コード	リクエストは正常に終了しました
1	killed + シグナル	リクエストはシグナルにより中断しました
2	Deleted	リクエストはqdelコマンドにより削除されました
3	Error	リクエストはJobCenterの内部エラーにより消滅しました
4	time out	タイムアウト時間に達しました
5	not found	リクエストは存在しません
6	qwait error	qwaitコマンド自体のエラーです

2. 使用例

```
REQID1=`qsub -Z JOB1`       # スクリプト JOB1 を実行するリクエストを投入し、
                            # リクエスト ID を REQID1 に格納
EXSTAT1=`qwait $REQID1`     # JOB1 の実行終了を待ち、終了ステータスをEXSTAT1 に格納

if [ "EXSTAT1" = "done 0" ]
then                        # JOB1 が正常に終了していたら、スクリプト JOB2
     qsub JOB2              # を実行するリクエストを投入
else                        # JOB1 が正常に終了しなかった場合は、スクリプト JOB3
     qsub JOB3              # を実行するリクエストを投入
fi
```

4.18.2. オプション

```
-f

まだリクエストが qsub されていない場合も終了を待ち合わせます。
```

```
-r $request-name

リクエスト ID の代わりにリクエスト名を指定します。同じ名前のリクエストが複数ある場合、以下の条件に従ってリクエストを待ち合わせます。

■同一名のリクエストがすべて終了している場合

  qwait は最後に終了したリクエストのステータスを返します。
```

■同一名のリクエストで終了していないものが一つだけ存在する場合

qwait はその終了していないリクエストを待ち合わせます。

■同一名のリクエストで終了していないリクエストが存在する場合

qwait はそれらのリクエストの終了を待ち合わせ、最初に終了したリクエストの結果を返します。

-t $timeout

$timeout で指定した時間をタイムアウト時間とします。タイムアウト指定時の単位は秒です。

4.18.3. 注意事項

リクエストの終了状態はリクエスト終了後も一定期間保存されます。この期間内ではqwait コマンドは即終了し、終了状態を表示します。期間を過ぎたリクエストを qwait した場合、 "not found "となります。

4.18.4. 関連項目

qsub(1).

4.19. # NScheck　チェックポイントの設定

```
# NScheck [ -c ] [-varall $save-variables ... ]

# NScheck $checkpoint-name [ -f $setup-file | $setup-function ] [-var $save-variables ... ]
```

4.19.1. 機能説明

NScheck は、バッチリクエストのシェルスクリプトの中で、ジョブステップリスタート機能を設定します。リクエストが、システムの停止や qrerun(1) によって中断したときに、リクエストの再実行は最後に通過したチェックポイントから行われます。このとき、シェル変数や環境変数は可能な限り自動的に復元されます。

■チェックポイント名

チェックポイント名はリクエスト内で一意でなければなりません。同一のチェックポイント名が複数存在した場合、リクエストは最初に通過した以外の同名のチェックポイントをすべて無視します。

4.19.2. オプション

1. オプションおよび引数

NScheck 行で使用するオプション、および引数は以下のとおりです。

-c

ジョブスクリプトの実行シェルタイプを csh に設定します。本オプションが指定されなかった場合、シェルスクリプトは sh (Bourne-Shell) 系のシェルで実行されるものとして扱われます。本オプションはスクリプトの先頭から、最初のチェックポイントが現れるまでの間で宣言されなければなりません。

-f $setup-file

リクエストの再実行開始時に実行するスクリプトファイルを指定します。本オプションを使用する場合、リクエストが再実行される実行マシン上に指定したファイルが存在しなければなりません。ファイルが存在しない場合、リクエストはエラーで終了します。

-var $save-variables ...

チェックポイントで保存するシェル変数を指定します。通常、後述する特殊なシェル変数を除いて、すべてのシェル変数は自動的に復元されますが、自動的に復元されない、あるいは正しく復元されない可能性のあるシェル変数を $save-variables として指定することにより、それらを復元することが可能となります。ただし、本オプションを使用してもある種の特殊なシェル変数 (直前に実行したコマンドの引数 が入る、などの、シェル内で自動的に設定されるシェル変数など) については復元することはできません。そのような場合には $setup-function、または $setup-file にそれらを復元する記述を行って復元してください。

-varall $save-variables ...

この指定以降のすべてのチェックポイントで共通して保存するシェル変数を指定します。-var オプションによる指定は各チェックポイントにおいてのみ有効となりますが、本オプションで指定されたシェル変数は同一リクエスト内の本オプション指定以降に記述されたすべてのチェックポイントで有効となります。

$checkpoint-name

リクエスト内で一意となるチェックポイント名を指定します。リクエストの再実行はこの名前を参照して再実行を開始する箇所を決定します。同一名のチェックポイントが複数行指定された場合、動作は不定です。

$setup-function

リクエストの再実行時に実行するコマンド、またはシェル定義関数を指定します。　　sh系のシェルではジョブスクリプト内にシェル記述された関数定義を指定することが可能です。関数定義の記述は最初のチェックポイントよりも前方で行ってください。

2. 変数について

■チェックポイントで保存したシェル変数のうち、内容に改行コード、及び空白などを含むものについて、自動では正しく復元されません。これらの内容を復元する場合には、-var、 -varall オプションを使用して、別途シェル変数名を指定する必要があります。

■シェル変数の内容がリスト形式のものについては正しく復元されません。このような場合、シェル変数の内容をいったん環境変数として保存し、復元後セットアップスクリプト内でシェル変数に戻すなどの方法が別途必要となります。

■また、以下に示す環境変数、およびシェル変数はシェルにおいて特殊な機能をもつため、通常リクエストの再実行時に自動的に復元されません。

IFS, PPID, WINDOWID, argv, cwd, loginsh, status, tcsh, terminal, tty, PWD, HOST, OPTARG, _

上記の環境変数、及びシェル変数の再実行前の実行時に使用していた値が必要となる場合には、これらの値を-var、-varall などのオプションを指定して保存することが可能です。

ただし、一部の環境変数については、強制的に再現しても、リクエストの実行環境に合わせて自動的に変更される場合があります。またそのようなシェル変数を強制的に変更した場合、リクエストを実行するシェルの動作が不正となる場合があります。

3. 記述の確認

nscppコマンドを使用することにより、ジョブスクリプト内の #NScheck 行の書式のチェックを行なうことができます。nscppから記述のエラーを指摘された場合には、まずその原因となる記述を修正してください。リクエストの実行時にエラーが検出された場合、エラーとなった記述は単なるコメント行として無視されます。

4.19.3. 注意事項

1. インプリメンテーション留意点

■sh系のシェルを用いる場合、関数定義を除くジョブスクリプトの記述は最初の #NScheck より後方に記述する必要があります。

■#NScheckの記述は if、 for、 whileなどの構造文の途中に記述することはできません。そのような記述がなされた場合にはリクエストの実行が正しく再開されません。また、これらの制限に関してはnscppでは指摘されません。

■本機能を使用する場合、リクエストの実行プログラムは sh 系のシェル、または csh 系のシェルのみが使用可能です。リクエストの実行に、awk、 sed、 perlなどを使用することはできません。ただし、これらの処理系をスクリプト内で呼び出して使用することは可能です。そのような場合、awkなどが解釈する箇所でチェックポイントを採取することはできません。

■ジョブステップリスタート機能を実装していない以前のバージョンのJobCenter、 NQSに対して #NScheck の記述のあるスクリプトを投入した場合、それらの行は単なるコメントとし

て扱われます。また、qsub(1)でリクエストを投入する際に -sr オプションを指定しなかった場合についても同様です。

2. その他注意事項

■-var、 -varall で前述の自動的に復元しない変数を復元した場合、復元した内容が実際の動作環境に一致しないなどの理由により、シェルの動作が不正となる場合があります。

■チェックポイントの通過時に -var、 -varall で指定されたシェル変数が存在しない (変数名が参照できない) 場合、リクエストはエラーで終了します。

4.19.4. 関連項目

qsub(1), nscpp(1).

4.20. nscpp　チェックポイントの設定のテスト

```
/usr/lib/nqs/nscpp $check-script
```

4.20.1. 機能説明

nscpp は、ジョブスクリプトに記述された #NScheck 行の設定のテストを行います。

■テスト内容について

nscpp がテストする内容は以下の項目です。

- #NScheck のオプション

  #NScheck 行に指定されたオプションのチェックを行います。不正なオプションが指定された場合エラーとなります。

- #NScheck の引数として指定されたシェル変数名

  -var、 -varall の引数として指定された変数名に不正な文字が含まれていないかをテストします。

- #NScheck のチェックポイント名

  チェックポイント名に不正な文字が含まれていないかをテストします。また、同一の名前のチェックポイントが存在しないかをテストします。

4.20.2. オプション

```
$check-script

#NScheck の記述をテストするスクリプトファイルです。
```

4.20.3. 注意事項

1. インプリメンテーション留意点

   ■nscpp はシェルスクリプト全体の構文のテストを行いません。したがって、#NScheck の記述ができない for、 while のループ内や、 if などの構造文の中に #NScheck の記述があっても nscpp はエラーを報告しません。

   ■nscpp は指定したシェル変数がチェックポイント通過時に存在するかどうかについてテストしません。

2. その他注意事項

   nscppは　#NScheck行についてのみ構文テストを行います。したがって、ジョブスクリプト内の　#NScheck行以外の箇所にシェルスクリプトとしての構文ミスがあったとしてもそれらについてはエラーを報告しません。

4.20.4. 関連項目

qsub(1), #NScheck(1).

第5章 JobCenter環境の構築

本章では JobCenter システムを起動させるのに最低限必要な項目について説明します。運用形態に応じた JobCenter システムの環境の調節方法については6章 「JobCenter 構成管理」 以降に説明します。

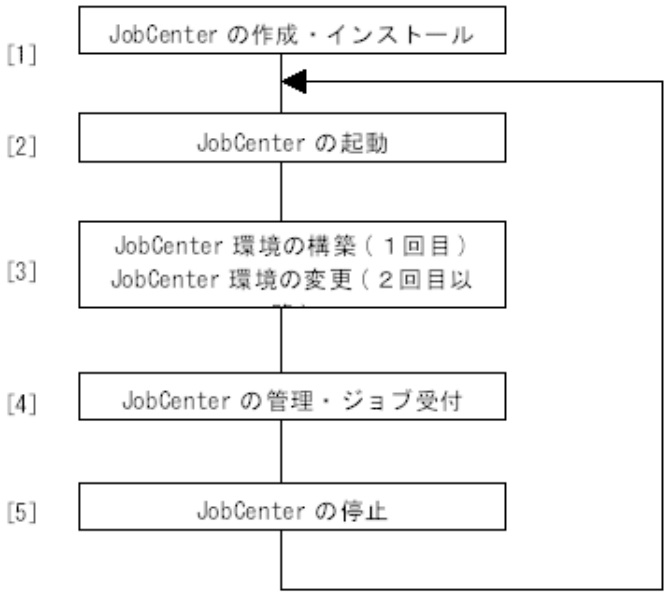


図5.1 JobCenterの環境構築のイメージ

［1］JobCenter システムの作成とインストールを行うステップです。

［2］インストールした JobCenter システムを起動させるステップです。

［3］1 回目の JobCenter の起動時にはJobCenter システムを運用していくうえで必要となる JobCenter キューなどを作成し、 2 回目以降は、それらの追加・変更などをするステップです。

［4］実際にバッチジョブなどを受け付けたり、JobCenter システムの管理を行うステップです。

［5］JobCenter を停止するステップです。

2回目以降のJobCenter 運用は［2］～［5］の手順を繰り返すことになります。

5.1. JobCenterの構成

JobCenter のインストールはリリースメモを参照して行ってください。

JobCenter　のインストールが正しくできたら、次に　JobCenter　環境の構築を行います。JobCenter　環境の構築は、まず各サイトの運用に応じて各キューの構成を設計して、それらを実際に定義します。これらのキューの定義はスーパーユーザもしくはJobCenter管理者が行います。

ここで説明するのは JobCenter の構成を設計する場合に考慮すべき点や参考例の紹介です。それを元に、自分のサイトに合わせて設計するようにしてください。

5.1.1. JobCenterキューの構成

まず JobCenter キューの構成について説明します。JobCenter キューには以下の3つのタイプがあります。

バッチキュー	バッチリクエストを投入するためのキュー
パイプキュー	ほかのキューへリクエストを転送するためのキュー
ネットワークキュー	実行結果ファイルを転送するためのキュー

キューはタイプごとに構成を考える要因が異なります。以下、タイプごとに説明します。

5.1.1.1. バッチキュー

バッチキューは、バッチリクエスト　(ジョブと呼ぶ場合もあります)　を処理するキューで JobCenter の最も基本的なキューです。JobCenter では複数のバッチキューを作成できます。作成できるキューの数に関する制限はありませんが、あまり多く作成するのは運用を複雑にするだけでしょう。

バッチキューの構成を考えるうえでの第一の要因は、資源制限と同時実行可能数による分類です。資源を多量に消費するバッチリクエストが同時に多く流れると、システム全体の効率を落としてしまいます。例えば多量にメモリを消費するバッチリクエストが複数同時に実行されると、カーネルによるスワップ処理が頻繁に行われオーバヘッドに費やされる時間が多くなり、システム全体の効率が低下してしまいます。逆に、あまり資源を消費しないバッチリクエストは複数同時に実行してもオーバヘッドの増加はほとんどないため、複数同時に実行できるようにした方がシステム全体の効率を上げることができるでしょう。

このように、実行するバッチリクエストの所要資源量とその同時実行可能数別に複数のバッチキューを用意するやり方があります。

次の表はこのような観点でバッチキューを設計した例です。優先度はバッチキュー間の優先度であり、 JobCenter の優先度の高いバッチキューから順に次に起動すべきバッチリクエストを探します。

表5.1 バッチキューの設定例1

	優先度	資源制限量	同時実行可能数
小規模ジョブ用キュー	中	小	多（4）
中規模ジョブ用キュー	中	中	中（2）
大規模ジョブ用キュー	中	大	少（1）

表中の同時実行可能数の欄の括弧内の値は、同時実行可能数の例です。

資源制限による分類は、上記の表のように全体として大中小に分ける方法のほかに、資源制限の内容によって分ける方法もあります。たとえば、メモリはあまり使わないが　CPU時間を多く消費するもの、 CPU 時間はあまり使わないが使用するファイルサイズが巨大なもの、といった分類です。ただし、あまり細かくわけると逆にどのバッチキューを使用したらよいかわからないなど、

運用が複雑になってしまいます。最初はある程度大まかにわけ、後で特別なキューを追加していくほうがよいでしょう。

バッチキューの構成を考える上での第二の要因は、各バッチキューを使用できるユーザによる分類があげられます。これはたとえば一般のユーザが使用するバッチキューと特定のプロジェクトのユーザが使用するバッチキューをわけるような方法です。バッチキューは、通常だれでもバッチリクエストを投入できますが、管理者によりバッチリクエストを投入できるユーザを制限できます。

次の表は上記の表5.1「バッチキューの設定例1」にJobCenter　管理者専用の緊急ジョブ用キューと、特定プロジェクト用キューを追加したものです。

表5.2 バッチキューの設定例2

	優先度	資源制限量	ユーザ制限	同時実行可能数
緊急ジョブ用キュー	高	無制限	JobCenter管理者	無制限
特定プロジェクトキュー	中	大	特定プロジェクト	少（1）
小規模ジョブ用キュー	中	小	なし	多（4）
中規模ジョブ用キュー	中	中	なし	中（2）
大規模ジョブ用キュー	中	大	なし	少（1）

実際の資源制限量や同時実行可能数の値は、各サイトにおけるシステム構成によって変わりますので、主記憶はどのくらい実装されているか、ファイルシステムの構成はどうなっているか、どのようなバッチリクエストを実行することが多いかなどを考慮して決定してください。

JobCenter　では、いくつかのバッチキューをひとまとまりにして全体としてバッチリクエストの同時実行可能数を制限するキュー複合体を定義できます。たとえば、上記の例で特定プロジェクト用キューと一般用の大規模ジョブ用キューには、大規模ジョブがそれぞれ 1 つずつ同時に実行できるため、システムとしては同時に 2 個の大規模ジョブが実行できることになります。

しかしシステムとしてはせいぜい 1 個までしか大規模ジョブを同時に実行できない場合、特定プロジェクト用キューと大規模ジョブ用キューからなるキュー複合体を定義し、そのキュー複合体の同時実行可能数を　1　個にすることにより、システムとしての大規模ジョブの同時実行を 1 つに制限できます。この時、特定プロジェクト用キューと大規模ジョブ用キューのどちらかに投入されているバッチリクエスト数が　1　になると、たとえそのキューの同時実行可能数に余裕があっても、それ以上のバッチリクエストは実行されません。

5.1.1.2. パイプキュー

パイプキューは、ほかのキューへリクエストを転送するキューです。パイプキューの用途は大きくわけて 3 つあります。

1. 資源制限などの条件により、リクエストをバッチキューの間で自動的に振り分ける

2. リクエストをリモートホスト上の特定のキューに転送する

3. 複数のホストで負荷を分散する

以下は代表的なパイプキューの使用例です。

表5.3 パイプキューの設定例

	転送先
自動資源分けキュー	小規模ジョブ用バッチキュー

	中規模ジョブ用バッチキュー 大規模ジョブ用バッチキュー
特定目的キュー	ホスト1特定目的キュー
負荷分散キュー	ホスト1自動資源分けキュー ホスト2自動資源分けキュー ホスト3自動資源分けキュー

最初の自動資源分けキューは、転送先が資源制限の異なる 3 つのバッチキューとなっており、このパイプキューに投入すればユーザがいちいち自分でバッチキューを選択しなくても、適当なバッチキューが選択されるようになります。

2 つめの特定目的キューは特定ホストでしか実行できないリクエストをほかのホストから投入できるようにするものです。

最後の負荷分散キューは複数の転送先ホストの中からパイプキューが適当なホストを選ぶことによって、これらのホストの間で負荷分散を行います。

5.1.1.3. ネットワークキュー

ネットワークキューは結果ファイルを転送するためのキューで、その構成は転送先ホストによって決定されます。

Windows版JobCenterではネットワークキューはサポートしていません。

以下はこれらの代表的なネットワークキューの使用例です。

表5.4 ネットワークキューの設定例

	優先度	転送先ホスト	同時転送可能数
ローカルホストLOC用キュー	高	LOC	無制限
リモートホストRMT1用キュー	中	RMT1	8
リモートホストRMT2用キュー	高	RMT2	4

転送先ホストに対応したネットワークキューが用意されていない場合、ネットワークジョブはデフォルト・ネットワークキュー（キュー名: DefaultNetQue）に投入されます。このキューはバッチジョブの結果ファイルをジョブの投入元ホストに出力するものです。

デフォルト・ネットワークキューは、転送先ホストに対応するネットワークキューを作成する前や、ネットワークキューがなんらかの理由で投入を拒否されている場合に使用されます。

基本的にはローカルホストを含む各ホストのネットワークキューを用意していただくことによって、柔軟なスケジューリングが可能となります。

5.2. JobCenterキューの作成

JobCenter　キューはJobCenter　システムの中でもっとも重要なものです。インストール直後は（デフォルトのキューを除くと）キューが全く無い状態になっていますので、まずこれを作成する必要があります。

なおデフォルトキューについては<基本操作ガイド>の「7.2　デフォルトで作成されるキュー」を参照してください。

キューにはその役割に応じて 3種類のキュータイプがあります。

バッチキュー	バッチリクエストを投入するためのキュー
パイプキュー	ほかのキューへリクエストを転送するためのキュー
ネットワークキュー	実行結果ファイル(stdout, stderr, 実行結果レポート)を転送するためのキュー

キューの作成は qmgr(1M) コマンドの create サブコマンドを用いて行います。

create batch_queue $queue_name ...	バッチキューの作成
create pipe_queue $queue_name ...	パイプキューの作成
create network_queue $queue_name ...	ネットワークキューの作成

以下にqmgrサブコマンドによる各キューの作成手順例を示します。実際にはあらかじめ設計しておいたキュー構成案にしたがって作成してください。

1. バッチキューの作成

```
# qmgr ↵
Mgr: create batch_queue batch1 priority=20 ↵
```

以上の手続きでバッチキュー batch1 が作成されます。このとき指定する priorityはキュープライオリティのことですが、キュー構成案にしたがった値を指定してしてください。

その他のキューの属性も指定できますが、後節で詳しく説明します。

2. パイプキューの作成

```
Mgr: create pipe_queue pipe1 priority=20 server=(/usr/lib/nqs/pipeclient) ↵
```

(R12.7以降のWindows版の場合は次のとおり)

```
Mgr: create pipe_queue pipe1 priority=20 ↵
```

以上の手続きでパイプキュー pipe1 が作成されます。このとき指定する priority はキュープライオリティのことですが、キュー構成案にしたがった値を指定してください。ここで指定する server はリクエスト転送処理を行うプログラムのことです(R12.7以降のWindows版では指定不要)。

その他のキューの属性も含めて、後節で詳しく説明します。

3. ネットワークキューの作成

```
Mgr: create network_queue net1 destination=[103] priority=20 ↵
```

以上の手続きでネットワークキュー　net1　が作成されます。このとき指定する　priority　はキュープライオリティのことですが、キュー構成案にしたがった値を指定してください。

destinationにはマシン ID の代わりにマシン名を指定することもできます。その他のキューの属性も指定できますが、後節で詳しく説明します。

5.3. JobCenterキューの属性定義

JobCenterキューには各キューのタイプに応じてそれぞれ特有の属性を定義できます。以下にその各キューの属性について説明します。

5.3.1. バッチキュー

バッチキューの属性としては以下のものがあります。

5.3.1.1. キュープライオリティ

キュー間プライオリティとも呼びます。リクエストのスケジューリングの際にどのキューに登録されているリクエストを優先的に実行するかを決める要因になります。この値が大きい方が優先度が高く、等しい場合はキューへの投入時刻順に従います。この属性はバッチキューに閉じたものであり、ほかのタイプのキューのキュープライオリティとはなんら関係をもちません。

この属性はバッチキューを作成する際に必ず指定しなければなりません。

5.3.1.2. 同時実行可能リクエスト数

そのキューに登録されたリクエストで同時に実行できる数です。この属性はキューを作成するときに指定することもできますが、指定しなくてもかまいませんし、キュー作成後にこの属性を変更することもできます。

ここではqmgrサブコマンドによるキュー作成時の指定方法についてのみ説明しておきます。
キュー作成後の設定・変更については後章で説明します。

```
Mgr: create batch_queue batch1 priority=20 run_limit=3 ↵
```

以上の手続きで、同時実行可能リクエスト数 3 のバッチキュー batch1 が作成されます。なお run_limit を指定しなかった場合は同時実行可能リクエスト数は1となります。

5.3.1.3. 資源制限

資源制限はそのキューに登録されるリクエストが使用する資源を制限するためのものです。詳細については後節で説明します。

5.3.1.4. スケジューリングパラメータ

スケジューリングパラメータは、各リクエストが実際に実行されるときの UNIXカーネルによるCPU 割当などのスケジューリングに関するものです。バッチキューに設定されたパラメータ値は登録されたリクエストに引き継がれ、リクエストが実行される際にそれらのパラメータがセットされます。

スケジューリングパラメータの例としては nice値 が挙げられます。

5.3.1.5. その他の属性

その他の属性として、キュー内リクエストスケジューリング方式、連続スケジュール数があります。詳しくは後節で説明します。

5.3.2. パイプキュー

パイプキューの属性として以下のものがあります。

5.3.2.1. キュープライオリティ

バッチキューで説明したものと同様のものです。

5.3.2.2. 同時実行可能リクエスト数

バッチキューで説明したものと同様のものです。qmgrサブコマンドによる指定例を以下に示します。

```
Mgr: create pipe_queue pipe1 priority=20 run_limit=3 server=(/usr/lib/nqs/pipeclient)↵
```

(R12.7以降のWindows版の場合は以下のとおり)

```
Mgr: create pipe_queue pipe1 priority=20 run_limit=3 ↵
```

以上の手続きで、同時実行可能リクエスト数 3 のパイプキュー pipe1 が作成されます(R12.7以降のWindows版ではserverオプションは指定不要)。

なお run_limit を指定しなかった場合は同時実行可能リクエスト数は1となります。

5.3.2.3. 目的地

目的地とは、そのパイプキューがリクエストを転送する先のキューのことです。この属性はキュー作成時に定義することもできますし、キュー作成後に定義・変更することもできます。ここではキュー作成時にqmgrサブコマンドにより定義する方法について説明します。キュー作成後の定義・変更については、後節で説明します。

```
Mgr: create pipe_queue pipe1 priority=20 server=(/usr/lib/nqs/pipeclient) \
     destination=(batch1@host1,batch2@host1) ↵
```

(R12.7以降のWindows版の場合は以下のとおり)

```
Mgr: create pipe_queue pipe1 priority=20 destination=(batch1@host1,batch2@host1) ↵
```

以上の手続きで、リクエストの転送先が batch1@host1 か batch2@host1 のパイプキューpipe1が作成されます。

上記の例のように目的地は複数定義できます。転送先は設定順に選択されます。したがって、まず batch1@host1 が転送先として選択され、batch1@host1 がリクエスト投入不可能な場合、batch2@host1 が転送先として選択されます。

"batch1@host1"とはホスト host1 上の batch1 というキューであることを示しています。

5.3.2.4. 事前チェック機能

この属性を指定すると、リクエストをパイプキューに登録する前にそのパイプキューの目的地となっているキューの状態を調べて、どこにも転送できない場合はパイプキューへの投入自体、不可能になります。

判定に用いる条件は以下のとおりで、これらを満たしていればパイプキューに登録されます。

■目的地のキュー上でリクエストが投入可能かつ実行可能

■リクエストの資源制限≦目的地のキューの資源制限

ただし、この機能は目的地がローカルのキューの場合にのみ有効です。

この属性を設定していなければ、目的地への転送が不可能であった場合でもパイプキューには登録されます。

ここではキュー作成時のqmgrサブコマンドによる定義方法を示します。

```
Mgr: create pipe_queue pipe1 priority=20 server=(/usr/lib/nqs/pipeclinet) check ↵
```

(R12.7以降のWindows版の場合は以下のとおり)

```
Mgr: create pipe_queue pipe1 priority=20 check ↵
```

このチェック機能をもたせたパイプキューの目的地にはローカルのバッチキューを 1 つ以上設定してください。リモートホストのキュー、パイプキューを指定するとエラーになります。

5.3.2.5. ステイウェイト

この属性が設定してあると、パイプキューに時間指定つき (qsub の -a オプション) のリクエストが投入された場合、そのリクエストをパイプキュー上でウェイトさせます。

したがって、目的地への転送は指定時間になったときに行われます。これはパイプキューを負荷分散の目的で使う場合に使います。なお、負荷分散機能はクラスタシステムでのみ有効です。

5.3.2.6. サーバ

サーバはそのパイプキューでリクエストの転送処理を行うプログラムのことです。

この属性はパイプキューを作成するときに必ず定義しなければなりません。(ただしR12.7以降のWindows版の場合は指定不要です)

■UNIX版

次のいずれかの絶対パスで指定します。

- /usr/lib/nqs/pipeclient (通常型・透過型パイプキューおよびデマンドデリバリ方式で指定)
- /usr/lib/nqs/rrpipeclient (ラウンドロビン方式で指定)
- /usr/lib/nqs/lbpipeclient (負荷情報収集方式で指定、現在のバージョンではサポートしていません)
- /usr/lib/nqs/netclient (ネットワークキューで指定、現在のバージョンではサポートしていません)

■Windows版

- ～R12.6.x

  %InstallDirectory%\lib\NSpipecl.exe (通常型・透過型パイプキューおよびデマンドデリバリ方式で指定)

- R12.7～R12.10.x

  serverの指定は必要有りません。

- R13.1～

  rrpipeclient (ラウンドロビン方式で指定)

  ラウンドロビン方式を使用しない場合、serverの指定は必要有りません。

どのプログラムを設定するかについては、本章の各項目および「6.7 負荷分散環境」を参照してください。また、サーバプログラムの動作については「7.8 ジョブトラッキング」を参照してください。

qmgrサブコマンドによる指定例を以下に示します。

```
Mgr: create pipe_queue pipe1 priority=20 \
     server=(/usr/lib/nqs/lbpipeclinet -n 3 -i 30) \
     destination=(batch1@host1,pipe1@local1,pipe2@local2) staywait ↵
```

5.3.3. ネットワークキュー

本機能はWindows版および現在のバージョンのUNIX版ではサポートしていない機能となります。

ネットワークキューの属性としては以下のものがあります。

5.3.3.1. キュープライオリティ

バッチキューで説明したものと同様のものです。

5.3.3.2. 同時転送可能リクエスト数

バッチキューで説明したものと同様のものです。

5.3.3.3. サーバ

サーバはそのネットワークキューで結果ファイルの転送処理を行うプログラムです。標準サーバプログラムとして /usr/lib/nqs/netclient が用意されています。キュー作成時にサーバを指定しなければ、 NetShepherd パラメータで指定されているサーバプログラムが使用されます。

サーバプログラムとして空文字を指定するか、 NetShepherd パラメータでサーバプログラムが何も指定されていない状態で、サーバプログラムを指定せずにキューを作成した場合、ネットワークキューにはサーバプログラムが設定されません。この場合ネットワークキューはサーバプログラムを使用せずに結果ファイルの転送を行います。

5.3.3.4. 転送先ホスト

転送先ホストのマシン ID です。マシン ID の代わりにマシン名を指定することもできます。qmgr サブコマンドによる指定例を以下に示します。

```
Mgr: create network_queue net1 destination=[103] priority=20 run_limit=8 \
     server=(/usr/lib/nqs/netclient) ↵
```

以上の手続きで、マシン ID が 103 であるホストを転送先とする、同時転送可能リクエスト数が 8 のネットワークキュー net1 が作成されます。

なおrun_limit を指定しなかった場合、同時転送可能リクエスト数は1となります。

5.4. JobCenterキュー複合体の作成

JobCenter　　キュー複合体とは、いくつかのバッチキューをひとまとまりに管理するためのもので、複数のバッチキューで同時に実行できるリクエスト数を制限したいときに作成します。

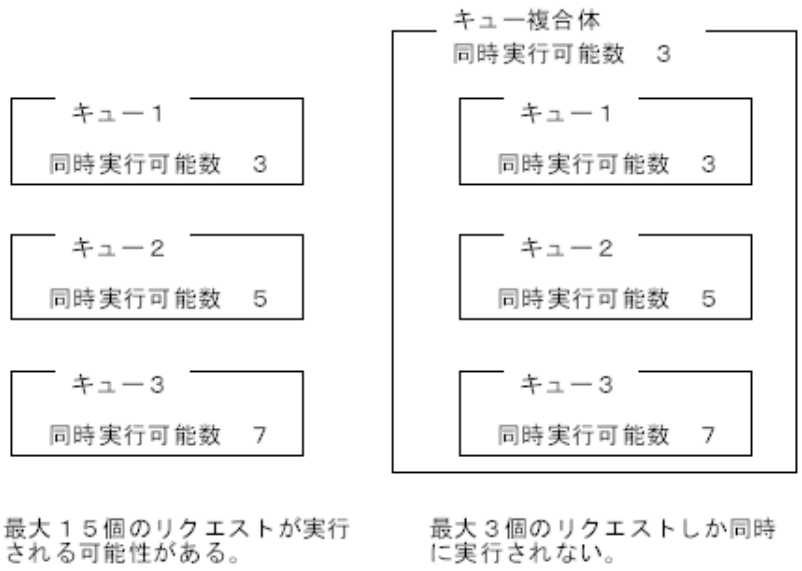


図5.2 キュー複合体の利用イメージ

キュー複合体の作成は qmgr(1M) コマンドの create complex サブコマンドで行います。

```
Mgr: create complex=(batch1,batch2,batch3) complex1 ↵
```

以上の手続きで　batch1、　batch2、　batch3　の　3　つのキューで構成されるキュー複合体 complex1 が作成されます。またキュー複合体に同時実行可能リクエスト数を定義するには、以下のようにします。

```
Mgr: set complex run_limit=3 complex1 ↵
```

以上の手続きで、同時実行可能リクエスト数 3 がキュー複合体 complex1 に設定されます。

5.5. 透過型パイプキューの概要と設定方法

透過型パイプキュー (Transparent pipe queue) は、従来のパイプキューよりも高速かつ低負荷でローカルのバッチキューにリクエストを転送できます。利用目的としては、たとえば負荷分散環境の利用時にたいへん有効です (「6.7 負荷分散環境」参照)。

5.5.1. 動作原理

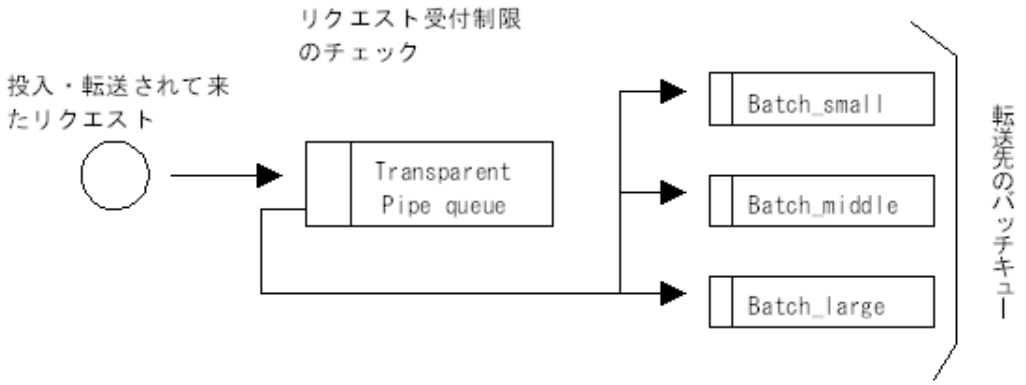


図5.3 透過型パイプキューの処理

透過型パイプキューはリクエストを受け付けるときに、まず自キューの受け付け制限をチェックします。これで問題がない場合には、続けて転送先でこのリクエストの受け付け処理を行うようにします。すべての転送先でリクエストが受け付けられなかった場合、透過型パイプキューもリクエストを受け付けないようにします。リクエストを受け付ける場合は転送先のバッチキュー側で直接受け付けるようにします。

5.5.2. 設定方法

透過型パイプキューの設定手順をqmgrサブコマンドにより説明します。

まず通常の方法でパイプキューを作成します。

```
Mgr: create pipe TransPipe priority=10 server=(/usr/lib/nqs/pipeclient) \
     destination=(BatchSmall, BatchMiddle, BatchLarge) ↵
```

(R12.7以降のWindows版の場合は以下のとおり)

```
Mgr: create pipe TransPipe priority=10 \
     destination=(BatchSmall, BatchMiddle, BatchLarge) ↵
```

ここでは TransPipe という名前のパイプキューを作成しました。このキューの転送先はローカルのマシン上にあるバッチキューです。ここでは、このバッチキューに資源制限がほどこしてあることにします。透過型パイプキューは、このような制限の異なる複数のバッチキューに、リクエストを高速かつ自動的に振り分けるために使用します。

次にこのキューを透過型パイプキューとして機能するように指定します。

```
Mgr: set transparent pipe_queue TransPipe ↵
```

以上で透過型パイプキューの設定が完了しました。設定の確認は qstat -x またはqstatq -f で可能です。

5.5.3. 従来のパイプキューとの違いについて

この透過型パイプキューは従来のパイプキューに対して次の点で異なります。

■転送先がローカルにあるバッチキューに限られます。それ以外のキューを指定しても無効になります。

■キューを STOP 状態にするとリクエストを受け付けることができません。

■転送先のすべてのキューがリクエストを受け付けられない場合、リクエストを受け付けません。

■順番に転送先のバッチキューを試していくときに、その 1 つがデマンドデリバリー用のバッチキュー (LB-BATCH キュー) で状態が MACHINE-BUSY 状態であったならば、それ以降のほかの転送先を試さないようにします。

また透過型パイプキューがリクエストを受け付けられなかった場合に返す（内部的な）エラーコードは、従来のパイプキューと異なり、おおよそ次のようになります。

■通常のパイプキューと同様のコードを返す。

■転送先バッチキューのエラーコードを返す。

■"DISABLE 状態" のエラーコードを返す。

また API(「9 APIライブラリ」参照) を使用する場合には次のことに注意してください。

■リクエストの転入 / 発生イベントは、転送先のキューで発行されます。

5.6. 自由転送先パイプキューの概要と設定方法

自由転送先パイプキューとは、リクエストの転送先をユーザが任意に設定できるパイプキューのことです。自由転送先パイプキューを利用すれば、複数のマシンから構成され、多数のバッチキューが存在するような比較的大規模なシステムにおいても、それぞれのバッチキューに転送するためのパイプキューを用意することなく、一つのパイプキューで集中管理できます。

5.6.1. 設定/ 解除

まず、通常のパイプキューである pipe1 に対して、下記のように qmgr(1M) のサブコマンドを用いてパイプキューに対して自由転送先パイプキューの属性を与えます。

```
Mgr: set free_destination pipe_queue pipe1 ↵
```

これで、 pipe1 が自由転送先パイプキューとして機能します。 qstatq -f pipe1 コマンドを用いて自由転送先パイプキューの属性を確認できます。

```
# qstatq -f pipe1 ↵
    .
    .
ATTRIBUTE
  BEFORECHECK   OFF
  STAYWAIT      OFF
  FREEDESTINATION ON
  LOADBALANCE   OFF
  TRANSPARENT   OFF
    .
    .
```

自由転送先パイプキューの属性の解除は次のように行います。

```
Mgr: set no_free_destination pipe_queue pipe1 ↵
```

5.6.2. 転送先キューの指定方法

qsub コマンドを用いてリクエストを投入する際、 -ds オプションを指定することによって、転送先キューを指定できます。

```
# qsub -q pipe1 -ds queueA@machine1 script1 ↵
```

ここでは、 script1 というシェルスクリプトを実行するリクエストを自由転送先パイプキュー pipe1 に投入し、その際 machine1 の queueA を転送先として指定しています。これによってリクエストは、転送先である machine1 の queueA に転送されて実行されます。

5.7. JobCenterネットワーク環境の構築

JobCenter　でネットワーク機能を使用できるようにするには、ネットワーク環境を構築しなければなりません。ネットワーク環境は nmapmgr(1M) コマンドを用いて構築していきます。

まず、ネットワーク上のマシンのマシンID を定義しなければなりません。自マシンID の設定についてはインストール時にすでに行われていますので、ネットワーク上の他の各マシンのマシンIDを追加設定します。

それぞれのマシンのマシン名とそのマシンIDを、nmapmgrサブコマンドでネットワークデータベースに登録します。

```
# nmapmgr ↵
NMAPMGR>:add mid 101 host2 ↵
```

例えqmgrでNQS管理者権限を付けたユーザであっても、nmapmgrコマンドについてはUNIX版ではroot(スーパーユーザ)、Windows版ではJobCenter管理者アカウントしか設定変更が行えません。それ以外のユーザではshowやgetの表示系サブコマンドのみ実行するようにしてください。

以上の手続きでhost2というマシンはマシンIDが101であるとされ、ネットワーク上のマシンとして認識できるようになります。

5.8. JobCenter 管理者の登録

JobCenter　を運用していくうえで必要な管理を行えるのはJobCenter　管理者と呼ばれるユーザです。スーパユーザはすでにJobCenter　管理者としての権利が認められていますが、ほかの一般ユーザにもその権利を与えることができます。また、　JobCenter　の運用操作をすることを認める、JobCenter 操作員と呼ばれるユーザも設定できます。

JobCenter 管理者は qmgr(1M) コマンドのサブコマンドをすべて使用できますが、 JobCenter 操作員はその一部のサブコマンドしか使用できません。大まかにいえば、JobCenter　操作員はキューの構成やグローバルパラメータなどのNQSの構成変更は行えないようになっています。

NQS設定に関する管理者権限の設定については、 qmgr(1M) の set manager サブコマンドで行います。(なおネットワークの構成管理を行うnmapmgrコマンドについては、追加の管理者権限の設定はできません)

■JobCenter 管理者の設定

```
Mgr: set manager user1:m ↵
```

■JobCenter 操作員の設定

```
Mgr: create network_queue net1 destination=[103] priority=20 ↵
```

以上の手続きで user1 はJobCenter 管理者に、 user2 はJobCenter 操作員にそれぞれ任命されることになります。なお、JobCenter 管理者とJobCenter 操作員の区別は、指定したユーザ名の後ろに、":m"か ":o"のどちらをつけるかで行われます。

■上記はUNIX版のみのサポートとなります。Windows版の場合はset　managerやadd managersサブコマンドでJobCenter管理者以外のユーザにNQSの管理者権限を付けても、そのユーザではqmgrコマンドは起動できませんのでご注意ください。

■例えqmgrでNQS管理者権限を付けたユーザであっても、nmapmgrコマンドについてはUNIX版ではroot(スーパーユーザ)、Windows版ではJobCenter管理者アカウントしか設定変更が行えません。それ以外のユーザではshowやgetの表示系サブコマンドのみ実行するようにしてください。

以上で JobCenter を運用する準備がほぼ整いました。設定した内容はシステムを停止しても保存されています。

第6章 JobCenter 構成管理

本章では JobCenter の構成管理について説明します。

6.1. キュー構成管理

JobCenter　キューの生成方法については前章で説明しましたが、この章では主にキューの構成の変更 (追加、削除、属性変更など) について説明します。

6.1.1. バッチキューの生成

JobCenter　　を運用していく上で新たなバッチキューが必要になったり、運用形態の変更などでバッチキューの追加が必要になることがあります。その場合は、すでにあるバッチキューとの関係に注意して新しいバッチキューを作成してください。

バッチキューの作成方法については、すでに5章　「JobCenter環境の構築」　で説明しましたので、それを参照してください。

6.1.2. バッチキュー属性定義(資源制限)

すでに説明したようにバッチキューの属性として資源制限があります（5章 「JobCenter環境の構築」 参照)。この属性はバッチキューに登録されるリクエストの資源使用量を制限するためのものです。

登録するリクエストに指定された資源使用量がキューに定義した資源制限より多い場合は、キューへの登録を拒否します。この属性を使って、資源を大量に使用することを許すキュー、または少ししか許さないキューなど、キューのクラス分けをすることができます。

資源制限属性の定義は qmgr(1M) の set サブコマンドを用いて行います。 JobCenter では以下の資源制限をサポートしています(HP-UX 上の JobCenter の場合)。

■プロセスごとのコアファイルサイズ制限

■プロセスごとのデータセグメントサイズ制限

■プロセスごとの永久ファイルサイズ制限

■プロセスごとのメモリサイズ制限

■プロセスごとのナイス実行値

■プロセスごとのスタックセグメントサイズ制限

■プロセスごとの CPU 時間制限

以下にプロセスごとの永久ファイルサイズ制限を変更する場合の例を示します。

```
# qmgr ↵
Mgr: set per_process permfile_limit=(100.5kb) batch1 ↵
```

キュー batch1 のプロセスごとの永久ファイルサイズ制限が 100.5 キロバイトになります。

なお、それぞれの資源制限の設定方法については、「JobCenterコマンドリファレンス（R12.6 第3版以降）」の qmgrの項を参照してください。

6.1.3. バッチキュー属性定義(その他)

資源制限以外のバッチキューの属性としては、先にも説明しましたが、キュープライオリティ、同時実行可能リクエスト数、 nice 実行値などがあります。

6.1.3.1. キュープライオリティ

この属性はキュー作成時に必ず設定しなければなりませんが、後から変更することができます。キュープライオリティの変更は qmgr(1M) の set priority サブコマンドで行います。

```
Mgr: set priority=10 batch1 ↵
```

以上の手続きで batch1 のキュープライオリティが 10 に変更されます。

6.1.3.2. 同時実行可能リクエスト数

この属性はキュー作成後に定義・変更することができます。変更はqmgr(1M)のset run_limit サブコマンドで行います。

```
Mgr: set run_limit=5 batch1 ↵
```

以上の手続きで、バッチキュー batch1 の同時実行可能リクエスト数が 5 に変更されます。

6.1.3.3. ナイス実行値

この属性は、キュー作成後に定義・変更します。キュー作成時の既定値は 0 です。変更は qmgr(1M) の set nice_limit サブコマンドで行います。

```
Mgr: set nice_limit=5 batch1 ↵
```

以上の手続きで、バッチキュー batch1 のナイス実行値が 5 に変更されます。

6.1.3.4. キュー内スケジューリング方式

この属性は、キュー作成後に定義・変更します。キュー内スケジューリング方式とは、1つのキュー内にリクエストプライオリティの値が同じであるリクエストが複数あった場合の実行順序の決定方法です。

type0 であれば、投入された順に実行され、 type1 であれば、すべてのユーザのリクエストが公平に実行されるように実行順序を変更します。

キュー作成時の既定値は type0 です。変更は qmgr(1M) の set intra_queue_scheduling_typeサブコマンドで行います。

```
Mgr: set intra_queue_scheduling_type type1 batch1 ↵
```

以上の手続きで、バッチキュー batch1 のキュー内スケジューリング方式が type1 に変更されます。

6.1.3.5. 連続スケジュール数

この属性は、キュー作成後に定義・変更します。連続スケジュール数とは、一人のユーザが連続して実行させることのできるリクエストの数です。

キュー作成時の既定値は 0 (Undefined) です。変更は qmgr(1M) の set continuous_scheduling_number サブコマンドで行います。

```
Mgr: set continuous_scheduling_number 3 batch1 ↵
```

以上の手続きで、バッチキュー batch1 の連続スケジュール数が 3 に変更されます。

6.1.3.6. デマンドデリバリ機能

この属性はキュー作成後に定義、変更することができます。詳しくは「6.7.3 デマンドデリバリ方式」を参照してください。

6.1.3.7. 再起動属性

デーモン再起動時のバッチキューの停止等の機能を、キューの属性として定義します。変更は qmgr(1M) の set queue reboot_mode サブコマンドで行います。

キューの属性 (REBOOT_MODE) に設定できる値と動作は以下の通りです。

表6.1 キューの再起動属性

キュー属性	再起動時のキューの動作
RESTART	従来どおり、自動的にリクエストを再起動します
STOP	キューをSTOP状態にします
PURGE	キュー上のリクエストをエラー削除します
MIGRATION_STOP	CJCによるジョブマイグレーションが発生した場合に、キューを停止します
MIGRATION_PURGE	CJCによるジョブマイグレーションが発生した場合に、キュー上のリクエストをエラー削除します

上記属性のうちCJCに関連するMIGRATION_STOPとMIGRATION_PURGEは、Windows版では意味を持ちません。Windows版ではRESTART、STOP、PURGEのいずれかを指定するようにしてください。

以下に、キュー batch に PURGE属性を与える設定例を示します。

```
Mgr: set queue reboot_mode=PURGE batch ↵
```

上記設定をqstatqコマンドで確認します。

```
Mgr: exit ↵
# qstatq -f batch ↵
```

上記コマンドにより、キューの属性の REBOOT MODE の項目に PURGE が設定された事を確認する事ができます。

ERP・BIジョブはキューを利用しないでジョブの投入を行っているため、キュー起動時の属性（reboot_mode）を設定してもその影響を受けません。なお起動時の設定を"STOP"に設定しジョブの実行を停止させたい場合は、対処策として各ERP・BIジョブの直前にダミーの単位ジョブを配置してください。

6.1.4. パイプキューの生成

JobCenter　を運用していく上で新たなパイプキューが必要になる、あるいは運用形態の変更などでパイプキューの追加が必要になる場合は、既存のパイプキューとの関係に注意して新しいパイプキューを作成してください。

パイプキューの作成方法については、すでに5章　「JobCenter環境の構築」　で説明しましたので、それを参照してください。

6.1.5. パイプキュー属性定義

パイプキューの属性としては、キュープライオリティ、同時転送可能リクエスト数、使用サーバ、目的地があります。

6.1.5.1. キュープライオリティ

バッチキューの属性定義で説明したとおりです。

6.1.5.2. 同時転送可能リクエスト数

この属性は、キュー作成後に定義・変更することができます。変更は qmgr(1M) の set run_limit サブコマンドで行います。

```
Mgr: set run_limit=5 pipe1 ↵
```

以上の手続きで、パイプキュー pipe1 の同時転送可能リクエスト数が 5 になります。

6.1.5.3. チェック機能

この属性は、キュー作成後に設定、解除することができます。チェック機能を設定する場合は、qmgr(1M) の set check サブコマンドで行います。

```
Mgr: set check pipe1 ↵
```

チェック機能を解除する場合は qmgr(1M) の set no_check コマンドで解除します。

```
Mgr: set no_check pipe1 ↵
```

6.1.5.4. 透過型機能

この属性は、キュー作成後に設定、解除することができます。詳しくは「5.5 透過型パイプキューの概要と設定方法」を参照してください。

6.1.5.5. デマンドデリバリ機能

この属性は、キュー作成後に設定、解除することができます。詳しくは「6.7.3 デマンドデリバリ方式」を参照してください。

6.1.5.6. 使用サーバ

この属性は、キュー作成時に必ず設定しなければなりませんが、キュー作成後でも変更が可能です。変更は qmgr(1M) の set pipe_client サブコマンドで行います。

```
Mgr: set pipe_client=(/usr/lib/nqs/lbpipeclient -n 3 -i 30) pipe1 ↵
```

以上の手続きで使用サーバが /usr/lib/nqs/lbpipeclient に変更されます。

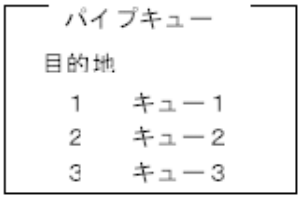
6.1.5.7. 目的地

この属性はキュー作成時にも定義できますが、キュー作成後でも、設定・変更・追加が可能です。

目的地とは、パイプキューに登録されたリクエストを転送する目的キューのことです。この目的地には複数のキューを設定することができます。目的地の選択順はサーバによって違います。

サーバに pipeclient を使用している場合は、選択順は固定です。

たとえば、以下のパイプキューの場合、



次に示すような結果になります。

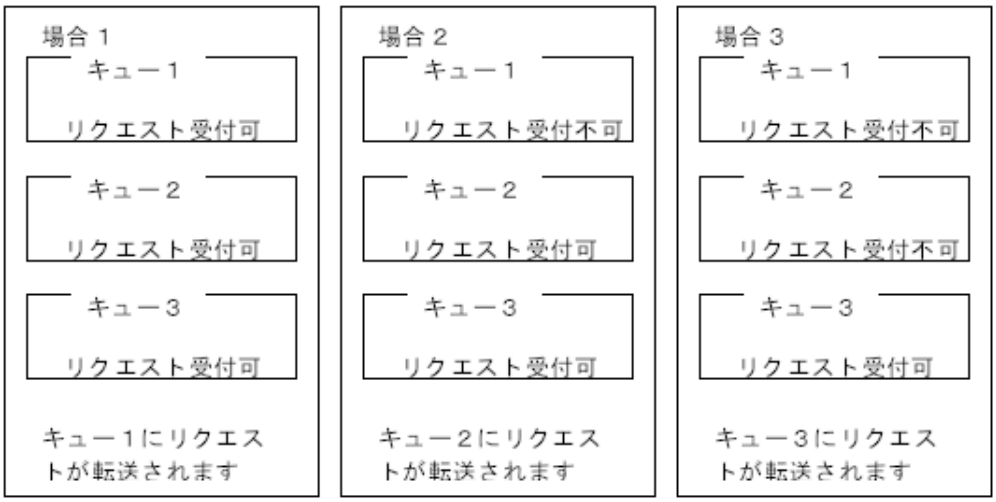


図6.1 パイプキューの転送結果

もし定義されているキューがすべてリクエスト受け付け不可だった場合は、ある一定時間後に再転送を試みます。再転送はある一定期間が経過するまで繰り返されます。

一定期間が経過しても転送が不可能であった場合は、その理由をリクエストを投入したユーザにメールで通知します。

ここで説明した、再転送の間隔、転送が繰り返される期間は、 JobCenter 環境パラメータに設定したものです。詳細については、後節で説明します。

パイプキューの目的地の定義・変更・追加は qmgr(1M) コマンドの add, set サブコマンドを用いて行います。

6.1.5.8. 目的地の定義・変更

```
Mgr: set destination=(batch1@host1) pipe1 ↵
```

以上の手続きでパイプキュー pipe1 の目的地として host1 上の batch1 キューが定義されます。この定義方法を行うと、以前に設定されていた目的地はすべてクリアされます。

6.1.5.9. 目的地の追加

```
Mgr: add destination=(batch2@host1) pipe1 ↵
```

以上の手続きでパイプキュー pipe1 の目的地に host1 上の batch2 キューが追加されます。たとえば、すでに pipe1 の目的地として batch1 が定義されていたら、 pipe1 の目的地はbatch1 と batch2 になります。

また、転送先にリモートホスト上のキューを定義すると、そのパイプキューはいわゆるネットワークパイプキューとなります。 JobCenter のネットワーク機能で最も重要なリモートホストへのリクエストの投入を行えるようにするには、このネットワークパイプキューを作成する必要があります。

目的地のキューの指定は以下の形式で行います。

キュー名 @ ホスト名

したがってネットワークパイプキューを作成する場合は、 "ホスト名 "にリモートホストの名前を指定することになります。

```
Mgr: set destination=(batch2@host2) netpipe1 ↵
```

以上の手続きをホスト host1 上で行うと、 host1 上の netpipe1 パイプキューが、リモートホスト host2 上の batch1 キューにリクエストを投入するためのネットワークパイプキューになります。

6.1.5.10. 再起動属性

デーモン再起動時のパイプキューの停止等の機能を、バッチキューと同様にキューの属性として定義します。変更は qmgr(1M) の set queue reboot_mode サブコマンドで行います。

詳細は、「6.1.3 バッチキュー属性定義(その他)」を参照してください。

6.1.6. ネットワークキューの生成

JobCenter　を運用していく上で、新たなネットワークキューが必要になることや、運用形態の変更などでネットワークキューの追加が必要になることがあります。その場合は、既存のネットワークキューとの関係に注意して新しいネットワークキューを作成してください。

ネットワークキューの作成方法については「5.2 JobCenterキューの作成」、「5.3.3 ネットワークキュー」を参照してください。

本機能はWindows版および現在のバージョンのUNIX版ではサポートしていない機能となります。

6.1.7. ネットワークキュー属性定義

ネットワークキューの属性としては、キュープライオリティ、同時転送可能リクエスト数、使用サーバ、転送先ホストがあります。

本機能はWindows版および現在のバージョンのUNIX版ではサポートしていない機能となります。

6.1.7.1. キュープライオリティ

バッチキューの属性定義で説明したとおりです。

6.1.7.2. 同時転送可能リクエスト数

バッチキューの同時実行可能リクエスト数と同じです。

6.1.7.3. 使用サーバ

この属性は、キュー作成時に指定しなければ環境パラメータに登録されているキューサーバが使用されます。環境パラメータでも設定されていない場合、ネットワークリクエストの処理を、キューサーバを使用せずに行います。

本属性はキュー作成後でも変更が可能です。変更は qmgr(1M) の set network_client サブコマンドで行います。

```
Mgr: set network_client=(/usr/lib/nqs/netclient2) pipe1 ↵
```

以上の手続きで使用サーバが /usr/lib/nqs/netclient2 に変更されます。

6.1.7.4. 転送先ホスト

この属性は、キュー作成後は変更できません。キュー作成時に必ず設定してください。

6.1.8. キューの削除

キューの削除は qmgr(1M) コマンドの delete queue サブコマンドで行います。このサブコマンドはキューのタイプに関係なく使用できます。ただし、削除しようとするキューが enable 状態であったり、キュー内にリクエストがある場合は削除できません。

```
Mgr: delete queue batch1 ↵
```

以上の手続きでバッチキュー batch1 が削除できます。

6.1.9. キュー複合体の生成/ 削除/ 属性定義

キュー複合体の説明および作成の仕方、属性の定義方法についてはすでに第 5 章で説明しましたので、ここではキュー複合体の構成の変更、削除、属性変更について説明します。

6.1.9.1. キュー複合体の構成の変更

キュー複合体を構成しているバッチキューの削除・追加をしたい場合は、qmgr(1M) コマンドの remove queue, add queue サブコマンドを用いて行います。

```
Mgr: remove queue=(batch1) complex1 ↵
```

以上の手続きでキュー複合体 complex1 の構成メンバであったキュー batch1 が、その構成メンバから外されます。

```
Mgr: add queue=(batch4) complex1 ↵
```

以上の手続きでキュー複合体 complex1 の構成メンバにキュー batch4 が加えられます。

6.1.9.2. キュー複合体の削除

キュー複合体の削除は qmgr(1M) コマンドの delete complex サブコマンドを用いて行います。

```
Mgr: delete complex complex1 ↵
```

以上の手続きでキュー複合体 complex1 が削除されます。

ただしキュー複合体を形成していたキューまで削除されることはありません。キュー複合体構成メンバのキューが構成メンバから解放されたと見なされます。

6.1.10. キューアクセス制限の設定/ 解除

キューには各ユーザ、グループに対してのみリクエストの投入を許可する機能があります。この機能を利用して、キューのクラスに応じて、そのキューを使用できるユーザ、グループを制限することができます。

アクセス制限に関する状態としては以下のものがあります。

6.1.10.1. アクセス無制限状態

アクセス制限が設定されていない状態、つまりすべてのユーザ・グループが使用可能な状態のことです。キューを作成した直後はこの状態になっています。

アクセス制限状態からこの状態に変更するには、 qmgr(1M) コマンドの set unrestricted_access サブコマンドで行います。

6.1.10.2. アクセス制限状態

アクセスが制限されている状態です。この状態は qmgr(1M) コマンドの set no_access サブコマンドで設定します。

キューアクセス制限を設定するには、まずアクセス制限状態にする必要があります。アクセス制限状態に移行した直後はスーパーユーザ以外のユーザ・グループはそのキューが使用できない状態になっています。

次に、特定のユーザ・グループにアクセス権を与えます。アクセス権の授与は qmgr(1M) コマンドの add group, add user、アクセス権の剥奪は delete group, delete user サブコマンドで行います。

たとえばキュー batch1 のアクセス権をユーザ user1 とグループ group1 に限定する場合は、まず batch1 をアクセス制限状態に移行します。

```
Mgr: set no_access batch1 ↵
```

user1 に batch1 のアクセス権を授与します。

```
Mgr: add user=user1 batch1 ↵
```

group1 に batch1 のアクセス権を授与します。

```
Mgr: add group=group1 batch1 ↵
```

以上の手順で batch1 の使用権が user1 とgroup1 だけに制限されます。なおスーパーユーザーは、明示的なアクセス権の有無にかかわらずいつでも使用が可能です。

アクセス権の剥奪方法は以下のとおりです。

```
Mgr: delete user=user1 batch1 ↵
```

6.1.11. デフォルトキューの設定/ 解除

デフォルトキューとは、ユーザがリクエストを投入するときにキュー指定をしなかった場合に、選択されるキューのことです。必ず設定しなければならないものではありません。

デフォルトキューはバッチリクエストについて用意できます。それぞれの設定は以下のように qmgr(1M) のサブコマンドで行います。

```
Mgr: set default batch_request queue batch1 ↵
```

以上でバッチリクエスト用のデフォルトキューが batch1 になります。

また、デフォルトキューを解除するには、以下のqmgr(1M) のサブコマンドで行います。

```
Mgr: set no_default batch_request queue ↵
```

以上でバッチリクエスト用のデフォルトキュー設定が解除されます。

6.2. JobCenter管理者の設定/ 解除

JobCenter にはJobCenter の管理を行うJobCenter 管理者と、JobCenter の運用操作を行うJobCenter操作員というユーザを設定する機能があります。

これらはJobCenter 管理者リストに登録されます。スーパーユーザは無条件に JobCenter 管理者として登録されます。

JobCenter 管理者構成管理としては設定・追加・解除という 3 つの機能が用意されています。

6.2.1. JobCenter管理者の設定

JobCenter 管理者の設定方法については、第 5 章で説明しましたので、そちらを参照してください。ただし、この JobCenter 管理者の設定を行うと、設定処理を行う前に設定されていた JobCenter 管理者リストがクリアされ、まったく新しい JobCenter 管理者リストが定義されます。

したがって、すでに設定されている JobCenter 管理者リストを継承し、さらに新たな JobCenter 管理者・操作員を設定したいときは、次に説明するJobCenter 管理者の追加処理を行ってください。

6.2.2. JobCenter管理者の追加

JobCenter 管理者の追加は、 qmgr(1M) コマンドの add managers サブコマンドで行います。

```
Mgr: add manager user2:m ↵
```

以上の手続きで user2 がJobCenter 管理者として追加されます。なお、JobCenter 操作員として追加したい場合は、"user2:o"と指定します。もしすでに user1 がJobCenter 管理者として登録されていたら、JobCenter 管理者は user1 と user2 になります。

6.2.3. JobCenter管理者の解除

JobCenter の管理者の解除は、 qmgr(1M) コマンドの delete managers サブコマンドで行います。

```
Mgr: delete manager user2:m ↵
```

以上の手続きで user2 はJobCenter 管理者から解除されます。JobCenter 管理者の追加と同様にJobCenter 操作員を解除したい場合は、"user2:o"と指定します。

6.3. JobCenter環境パラメータの設定

JobCenter　　では、その運用形態に合わせてサイトに設定できる環境パラメータがあり、インストール時に既定値が設定されています。これを変更することによってさまざまな運用形態をアレンジすることができます。

環境パラメータは qmgr(1M) の show parameter サブコマンドで参照することができます。以下に環境パラメータの表示例を示します。

```
# qmgr ↵
Mgr: show parameter ↵

  Maximum global batch run_limit = 100
  Maximum global network run_limit = 50
  Maximum global pipe run_limit = 50
  Debug level =  0
  Default batch_request priority = 31
  Default batch_request queue = NONE
  Default destination_retry time = 16 seconds
  Default destination_retry wait = 300 seconds
  Default device_request priority = 31
  No default print forms
  Default print queue = NONE
  (Pipe queue request) Lifetime = 168 hours
  Default network_retry time = 16 seconds
  Default network_retry wait = 0 seconds
  Default network_retry time_out = 30 seconds
  Default stage_retry time = 180 seconds
  Default stage_retry wait = 300 seconds
  Default expire time = 259200 seconds
  Log_file = /tmp/nqslog
  Log_file size = 10240 bytes
  Log_file backup = YES
  Mail account = root
  Maximum number of print copies = 2
  Maximum failed device open retry limit = 2
  Maximum print file size = 1000000 bytes
  Netdaemon = /usr/lib/nqs/netdaemon
  Netclient = /usr/lib/nqs/netclient
  Netserver = /usr/lib/nqs/netserver
  (Failed device) Open_wait time = 5 seconds
  NQS daemon is not locked in memory
  Next available sequence number = 397
  Batch request shell choice strategy = FREE
  Mapping mode = TYPE1
  Maximum batch request priority = 0
  Maximum global group submit limit = Unlimited
  Maximum global user submit limit = Unlimited
  Maximum global group run limit = Unlimited
  Maximum global user run limit = Unlimited
  Maximum IDC connection number = 32
  Qwatch event spool size = 65535
  Qwatch event expier time = 3600
  Inter Queue Scheduling mode = TYPE0
  Domain name  = BATCH

Mgr:
```

以下に環境パラメータのそれぞれについて説明します。

■同時実行可能バッチリクエスト数 (Global batch run_limit)

NQS　システムで同時に実行することができるバッチリクエストの数のことです。このパラメータには最高限度値があります。

■同時転送可能リクエスト数 (Global pipe run_limit)

NQS　システムで同時に転送できるリクエスト数のことです。このパラメータには最高限度値があります。

■同時実行可能ネットワークリクエスト数 (Global network run_limit)

NQS　システムで同時に実行することができるネットワークリクエストの数のことです。このパラメータには最高限度値があります。

■デバッグレベル (Debug level)

NQS　ログファイルに出力されるデバッグ情報のレベルのことです。既定では　0　になっています。レベルが 0 のときはデバッグ情報は出力されません。

■既定バッチリクエスト優先度 (Default batch_request priority)

ユーザがバッチリクエスト投入時に優先度の指定をしなかったときに自動的に設定される優先度です。優先度値の範囲は 0 ～ 63 で、値が大きいほど優先度が高くなります。

■既定バッチリクエストキュー (Default batch_request queue)

ユーザがバッチリクエスト投入時にキュー指定をしなかったときに自動的に選択されるキューのことです。

■リクエスト転送リトライ期間 (Default destination_retry time)

リクエストの転送先とのコネクション開設に失敗した場合、パイプクライアントはある一定の期間、リトライの間隔を増やしながらコネクション開設を繰り返し行います。このパラメータはその期間を定義するものです。

■リクエスト転送リトライ間隔 (Default destination_retry wait)

パイプクライアントがネットワーク先のマシンに対してのリクエストの転送に失敗した場合、JobCenter は、一定の時間をおいて再転送を繰り返します。

■既定ネットワーク接続リトライ期間 (Default network_retry time)

リクエスト転送および実行結果のステージアウト以外のコネクション開設に失敗したとき、JobCenter　はある一定の期間、リトライ間隔を増やしながらコネクションの開設を繰り返し行います。このパラメータはその期間を定義するものです。

■既定ネットワーク接続リトライ間隔 (Default network_retry wait)

ネットワークに何らかの障害が発生したとき、　JobCenter　はある一定の時間をおいてネットワーク接続を繰り返します。このパラメータはその間隔を定義するものです。

■既定ネットワークタイムアウト期間 (Default network_retry time_out)

ネットワーク先の応答がない場合に、 socket ストリームの切断と見なすまでの時間です。

■既定結果転送繰り返し期間 (Default stage_retry time)

ネットワークリクエストによるリクエスト結果ファイルの転送に失敗した場合、 JobCenter はこのパラメータで設定した期間、結果ファイル転送を繰り返します。

■既定結果転送間隔 (Default stage_retry wait)

ネットワークリクエストによるリクエスト結果ファイルの転送に失敗した場合、JobCenter はこのパラメータで設定した時間をおいて再転送を繰り返します。

■リクエスト存続時間 (Lifetime)

リクエストの転送に失敗した場合に、リクエストをエラーとして消去するまでの時間です。

■リクエスト終了情報消去時間 (Default expire time)

リクエストの終了情報を保持し続ける期間です。リクエスト終了後、この時間が経過するまでの間は qwait コマンドにより、リクエストの終了情報を取得することができます。あまり長いとディスク容量を余計に消費します。

■ログファイル (Log_file)

ログ情報を出力するファイルを定義します。既定では /dev/null になっています。

■ログファイルサイズ (Log_file size)

ログファイルのサイズの上限値です。ログファイルはこのサイズを超える前に自動的にリセットされます。この項目はインストール時には設定されず、サイズの指定がされるまで表示されません。

■ログファイルバックアップ (Log_file backup)

上記のログファイルサイズの設定に関連して、ログファイルをリセットする前にバックアップファイルを作成するかどうかのフラグです。

ログファイルのサイズが設定されていない (unlimited、または表示されない) 場合、この項目は表示されません。

■メールアカウント (Mail account)

JobCenter システムが送信するメールの発信ユーザを定義します。

■ネットデーモン (network daemon)

JobCenter 内部で使用するプログラムのパス名です。

既定では /usr/lib/nqs/netdaemon になっています。変更しないでください。

■ネットクライアント (network client)

本機能はWindows版および現在のバージョンのUNIX版ではサポートしていない機能となります。

結果ファイル転送用サーバとして使用するプログラムです。既定ではサーバプログラムは設定されていません。

通常サーバプログラムを設定する必要はありません。変更する場合、現在の JobCenter では /usr/lib/nqs/netclient が使用できます。

■ネットサーバ (network server)

JobCenter 内部で使用するプログラムのパス名です。既定では /usr/lib/nqs/netserver になっています。変更しないでください。

■デーモンのロック状況 (NQS daemon is not locked in memory)

JobCenter　デーモンがメモリ内にロックされるかどうかを定義します。lock　local_daemon/unlock local_daemon サブコマンドにより変更します。

■リクエスト連番 (Next available sequence number)

次に投入されるリクエストのリクエスト連番です。このパラメータは変更できません。

■バッチリクエストシェル選択方式 (Shell strategy)

バッチリクエストの実行時に使用されるシェル選択方式を定義します。

■マッピングモード (Mapping mode)

JobCenter マッピングモードを定義します。

■指定可能優先度 (maximum request_priority)

JobCenter　リクエストに一般ユーザが指定できるリクエストプライオリティの最大値を定義するものです。指定範囲は 0 ～ 63 です。

■ユーザごとの同時投入可能リクエスト数 (global user_submit_limit)

システム全体で、 1 人のユーザが同時に投入できるリクエストの数を定義します。

■グループごとの同時投入可能リクエスト数 (global group_submit_limit)

システム全体で、 1 つのグループが同時に投入できるリクエストの数を定義します。

■ユーザごとの同時実行可能リクエスト数 (global user_run_limit)

システム全体で、 1 人のユーザが同時に実行できるリクエストの数を定義します。

■グループごとの同時実行可能リクエスト数 (global group_run_limit)

システム全体で、 1 つのグループが同時に実行できるリクエストの数を定義します。

■最大 IDC コネクション数 (Maximum IDC connection number)

マシングループ内の各マシンとの通信を行う際のオーバヘッドを減らすためにマシン間のコネクションを切断せずに保持しておく最大数です。

■最大イベントスプーリングサイズ (Qwatch event spool size)

本パラメータで指定したサイズを保証するスプールファイルが、受信したイベント(NQS内部で利用される、プロセス間通信用のメッセージイベント)のバッファリングのために作成されます。

実際に作成されるファイルサイズはOSのページサイズに沿って丸められるため、指定したサイズと同じにはならない場合があります。スプールファイルはイベント登録のエントリごとに作成されます。

■最大イベント保持時間 (Qwatch event expier time)

本パラメータによって指定された時間中に、イベント(NQS内部で利用される、プロセス間通信用のメッセージイベント)を受信するプロセスがイベントを読み込まなかった場合、それまでに発生したイベントをすべて破棄し、イベントの登録を解除します。

■キュー間スケジューリング方式 (inter_queue_scheduling_type)

同じキュープライオリティをもつバッチキューが複数存在するときに、それらのキューに投入されているリクエストの実行順序を決定します。

type0 なら投入順に実行され、type1 なら 1 つのキューに投入されたリクエストのみが実行されることのないよう調節します。

以上の環境パラメータを定義・変更するためのコマンドは qmgr(1M) のサブコマンドとしてそれぞれ用意されています。したがって、自分の運用形態に沿わない環境パラメータ値は各サブコマンドで変更します。サブコマンドの詳細については、 qmgr(1M) コマンドの説明を参照してください。

例えば、同時に実行可能なバッチリクエストの数を変更したい場合は、次のようにサブコマンドで変更します。

```
Mgr: set global batch_request_limit 15 ↵
```

以上のようにすれば同時実行可能バッチリクエスト数が 15 になります。

6.4. シェル選択方式指定

シェル選択方式には以下の 3 つのタイプが用意されています。

■FIXED

バッチリクエストを実行するシェルとして、管理者により指定されたシェルが使用されます。指定されたシェルが sh の場合、ユーザの設定ファイル (.profile) は、ジョブ実行時に読み込まれません。

■FREE

バッチリクエストを実行する際に、まずリクエストのユーザのログインシェルが起動されます。次にそのログインシェルが、バッチリクエストを実行するシェルを選択し、そのシェルがバッチリクエストを実行します。つまり、あたかもインタラクティブな処理と同様な形態でバッチリクエストが実行されます。ユーザのログインシェルが sh の場合、ユーザの設定ファイル (.profile) はジョブ実行時に読み込まれません。

■LOGIN

バッチリクエストを実行するシェルとして、そのリクエストのユーザのログインシェルが使用されます。ログインシェルが sh の場合、ユーザの設定ファイル (.profile) は、ジョブ 実行時に読み込まれません。

以上のシェル選択方式はユーザがリクエストを投入する際に、バッチリクエストを実行するシェルを指定した場合は無効になります。

このシェル選択方式は、 JobCenter 環境パラメータにより定義されます。既定値のシェル選択方式は FREE です。

シェル選択方式を FIXED 型に、バッチリクエストを処理するシェルを"/bin/sh" に変更したい場合は次のようにサブコマンドを実行します。

```
Mgr: set shell_strategy fixed=(/bin/sh) ↵
```

シェル選択方式を FREE 型に変更したい場合は次のようにサブコマンドを実行します。

```
Mgr: set shell_strategy free ↵
```

シェル選択方式を LOGIN 型に変更したい場合は次のようにサブコマンドを実行します。

```
Mgr: set shell_strategy login ↵
```

6.5. JobCenterネットワーク環境設定

6.5.1. JobCenterネットワーク環境の概要

JobCenter にはネットワークを介してホスト間でリクエストの送受信を行う機能があります。

この機能を可能にするためには、各ホストで JobCenter ネットワーク環境を整える必要があります。

JobCenter ネットワーク環境には大きく分けて 2 つあります。 1 つは、ホストに関するもので、もう 1 つはユーザに関するものです。

6.5.1.1. ホストに関するネットワーク環境

UNIX　ではそれぞれのマシンにホスト名がつけられています。このホスト名は　UNIXのネットワーク上でマシンを判別するのに用いられています。このホスト名は/etc/hosts　に定義されています。

JobCenter でもこのホスト名をマシン(サイト)の識別に用います。ただしこのホスト名はユーザとのコミュニケーションに用いられるもので、　JobCenter　システムが実際にマシンを認識するのは、そのホスト名と関係づけられたマシンIDを用います。したがって、 JobCenter 管理者はホスト名とマシンID を関係づける必要があります。このマシンID は JobCenter の世界でしか使用されないので、 IP アドレスとは無関係にマシンID を決定することができます。

まずは、JobCenterネットワークに加盟しているマシンの管理者でそれぞれのマシンID　を重複しないように決定しなければなりません。JobCenter　ネットワークに加盟しているマシンのマシンIDをそれぞれ決定したら、まず、自マシンのマシンID　をシステムに設定します。この設定は通常JobCenter　のインストール時に行います。ただし、後に説明する方法によって変更することもできます。

次に、自マシン上の　JobCenter　が他のJobCenter　ネットワークに加盟しているマシンのマシンID　とホスト名が認識できるように、関連付けを行います。この関連付けのことをリモートマシン定義といいます。

6.5.1.2. ユーザに関するネットワーク環境

UNIX ではユーザをユーザ ID、グループ ID で管理しています。 JobCenter でもこれらのユーザ ID、グループ ID を用いて、ユーザの識別などを行っていますので、JobCenter システムで認識できるようにしなければなりません。

ローカルホスト上のユーザのユーザID、グループIDなどの情報は /etc/passwd ファイルを参照して行うので特別な設定を行わなくても構いませんが、リモートホストから利用するユーザについては、リモートユーザのローカルホストの ID を設定しなければなりません。この定義をリモートユーザ定義あるいは、リモートユーザマッピングと呼んでいます。

リモートホストのユーザの ID 設定形態として 3 つ用意されています。これらをマッピングモードと呼びます。

■マッピングモード TYPE1

ユーザ名を用いてユーザのマッピングを行います。つまり、リモートホスト上からJobCenterのアクセスしてきたユーザの名前と一致するローカルホスト上のユーザのID をそのユーザの ID として用いられることになります。

したがって、リモートホストとローカルホストのどちらにも同一ユーザ名で登録されているユーザでないと、そのホスト間でネットワーク機能は使用できないことになります。

■マッピングモード TYPE2

JobCenter　独自のリモートユーザ定義の情報のみでユーザのマッピングを行います。この場合は、たとえ両ホスト間に同一ユーザ名をもつユーザでも、リモートユーザ定義がされていないと、ネットワーク機能が使用できません。

このモードは、ネットワーク機能の使用を特定のユーザに限定したい場合に有効です。

■マッピングモード TYPE3

JobCenter　独自のリモートユーザ定義とユーザ名の両方でユーザのマッピングを行います。この場合はマッピングモード　1　で説明したユーザ名でのマッピングに加えて、リモートユーザ定義で定義した情報でのマッピングも行われます。

したがって、たとえ両ホストに同じ名前でパスワードエントリされていなくても、リモートユーザ定義されていれば、ネットワーク機能の使用ができます。

このモードは、両ホストで異なったユーザ名で登録されているユーザにネットワーク機能の使用を認める場合に有効です。

上記の概念について、次頁の図を参照してください。

ネットワーク上でのセキュリティを守るためには、ユーザマッピングによって対応づけられたユーザが、同一人物であることを保証する必要があります。

JobCenter　独自のリモートユーザマッピング定義によってマッピングされたユーザは、管理者によって同一人物であることが確認されたものと考えられるので、 JobCenter はこれらのユーザによる利用を無条件に許可します。

これに対して、ユーザ名によってマッピングされたユーザが同一人物であることを確認するためには、JobCenterはリモートシェル実行権を参照します。このリモートシェル実行権とはrsh(1)コマンドによるリモートホスト上でのコマンド実行権と同じものです。つまりrshコマンドと同じ仕組みの認証方式をJobCenterでは利用しています。

(ただしrshそのものを実行することはありません。JobCenter内部でrshを利用しているという意味ではありませんので、ご注意ください)

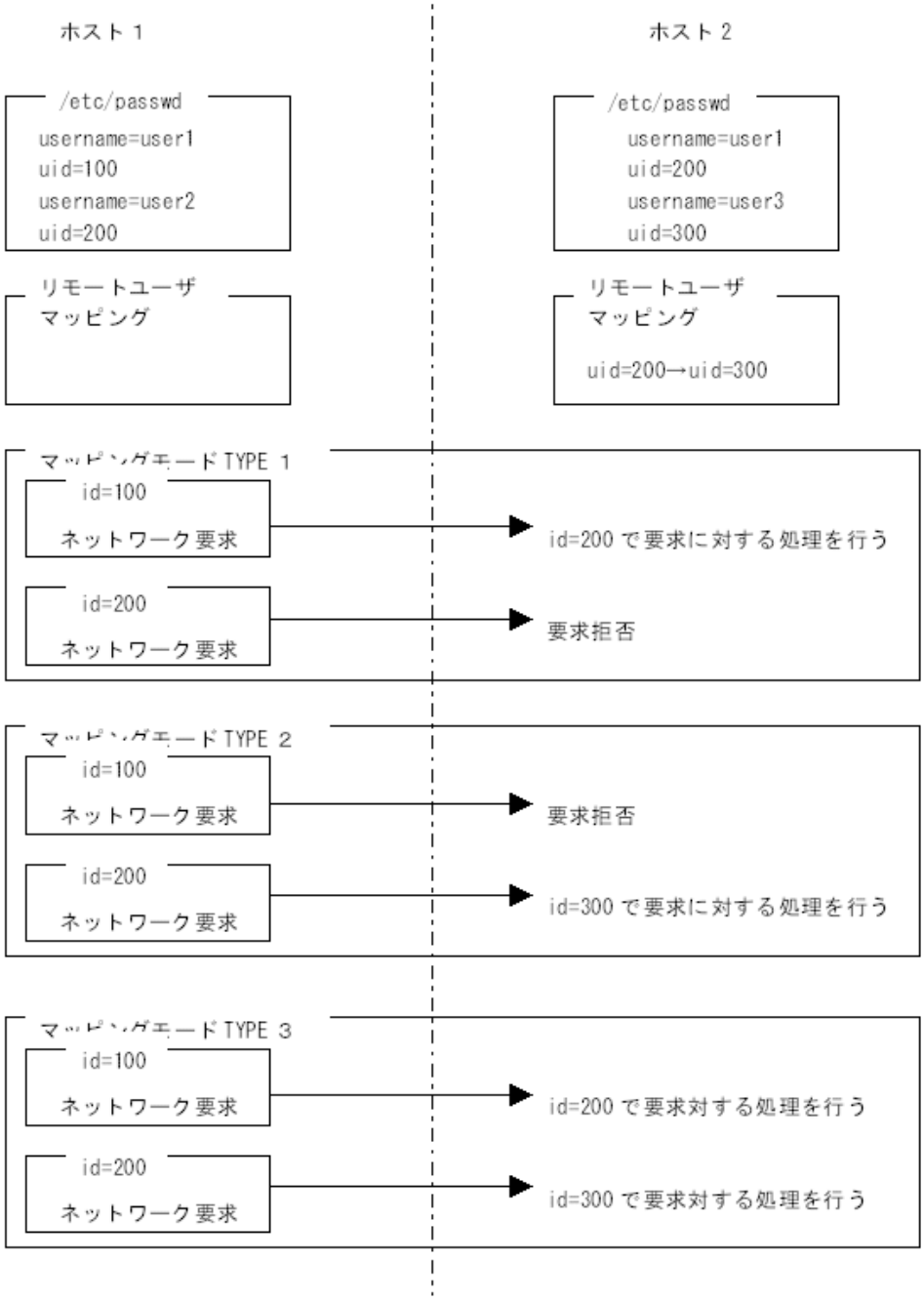


図6.2 マッピングモードとユーザマッピングの例

具体的には、各ユーザがホームディレクトリに .rhosts ファイルを作成する必要があります。

この .rhosts ファイルには、リモートホスト名とそのリモートホストでの対応するユーザ名を空白で区切って 1 行に 1 組ずつ記述します。また、リモートマシンのすべての同じ名前のユーザに、リモートシェル実行権を与える場合は、管理者が /etc/hosts.equivファイルにそのリモートホスト名を記述します。

詳しくはUNIXのユーザコマンドリファレンスのrsh(1) の項を参照してください。

6.5.2. リモートマシン定義

リモートマシン定義とは、マシンに付けられたホスト名と JobCenter ネットワーク上で用いられるマシンID を関連づけることができます。

リモートマシン定義は nmapmgr(1M) コマンドで行います。たとえば以下のようにリモートマシンを定義します。

```
# nmapmgr
NMAPMGR>: add mid 100 host1            ←自分のマシンの定義（インストール時に設定済）
NMAP_SUCCESS: Successful completion.
NMAPMGR>: add mid 110 host2            ←マシン2の定義
NMAP_SUCCESS: Successful completion.
NMAPMGR>: add mid 120 host3            ←マシン3の定義
NMAP_SUCCESS: Successful completion.
NMAPMGR>: exit
# ↵
```

これで、マシン2とマシン3からのネットワーク要求を受け入れる準備ができます。ただし、相手マシン(マシン2とマシン3)上に自分のマシン(マシン1)の定義がされていないと、ネットワーク機能は使用できません。

したがって、ネットワークで接続しようとするマシンの管理者と相談の上、自分のマシンの定義をしてもらってください。

また、一度設定したリモートマシン定義を変更するには以下のように行います。

この例では、 host2 のマシンID を 110 から、 200 に変更しています。

```
# nmapmgr ↵
NMAPMGR>: del mid 110 ↵
NMAP_SUCCESS: Successful completion.
NMAPMGR>: add mid 200 host2 ↵
NMAP_SUCCESS: Successful completion.
NMAPMGR>: exit ↵
```

Windowsマシン～UNIXマシン間でリモートマシン定義の設定を行う際は、それぞれのマシンでNQS TYPEの情報についてもnmapmgr(1M) コマンドで設定する必要があります。

6.5.2.1. UNIXマシン上でWindowsマシン定義のNQS TYPEを設定

UNIXマシンで、add midサブコマンドでリモートマシンを定義した場合、NQS TYPEはデフォルト"nec"で自動的に設定されます。対象のリモートマシンがWindowsである場合、NQS TYPEを"necnt"で再設定してください。

クラスタ環境については、環境変数NQS_SITEにクラスタサイト名をあらかじめ設定してからnmapmgrで再設定してください。

以下の例では、ローカルサイトでリモートマシンのhost2をWindowsマシンとして設定しています。

```
# nmapmgr ↵
NMAPMGR>: add mid 200 host2 ↵
NMAP_SUCCESS: Successful completion.
NMAPMGR>: SET TYPE 200 necnt ↵
```

設定を変更した結果をshow stateサブコマンドで確認してください。NQS TYPEが"EXTENDED TYPE OF NEC-NT"と表示されていることを確認します。

```
NMAPMGR>: show state host2 ↵
HOST NAME:  host2    HOST ID:  200
NQS TYPE:  EXTENDED TYPE OF NEC-NT
MAIL ADDRESS:  not set
USER MAPPING

GROUP MAPPING
```

```
NMAPMGR>: exit ↵
#
```

6.5.2.2. Windowsマシン上でUNIXマシン定義のNQS TYPEを設定

Windowsマシンで、add midサブコマンドでリモートマシンを定義した場合、NQS TYPEはデフォルト"necnt"で設定されます。対象のリモートマシンがUNIXである場合、NQS TYPEを"nec"で再設定してください。

クラスタ環境に対しては、環境変数NQS_SITEにクラスタサイト名をあらかじめ設定してからnmapmgrで再設定してください。

以下の例では、クラスタサイトhost2aでリモートマシンのhost3をUNIXマシンとして設定しています。

```
# NQS_SITE=host2a ↵
# export NQS_SITE ↵
# nmapmgr ↵
NMAPMGR>: add mid 120 host3 ↵
NMAP_SUCCESS: Successful completion. ↵
NMAPMGR>: SET TYPE 120 nec ↵
```

設定を変更した結果をshow stateサブコマンドで確認してください。NQS TYPEが"EXTENDED TYPE OF NEC"と表示されていることを確認します。

```
NMAPMGR>: show state host3 ↵
HOST NAME:  host3    HOST ID:  120
NQS TYPE:  EXTENDED TYPE OF NEC
MAIL ADDRESS:  not set
USER MAPPING

GROUP MAPPING

NMAPMGR>: exit ↵
#
```

6.5.3. リモートユーザ定義

リモートユーザ定義とは、リモートホストのユーザ ID、グループ ID とローカルホスト上でのユーザ ID、グループ ID とを関連づけることです。

リモートユーザ定義は nmapmgr(1M) コマンドで行います。

```
# nmapmgr ↵
NMAPMGR>: add uid 100 200 205 ↵
```

以上の手続きでマシンID が 100 番であるマシンのユーザ ID が 200 番であるユーザは本マシン上ではユーザ ID が 205 番のユーザとして扱われるようになります。

```
NMAPMGR>:add gid 100 300 305 ↵
```

以上の手続きでマシンID が 100 番であるマシンのグループ ID が 300 番であるユーザは本マシン上ではグループ ID が 305 番のユーザとして扱われるようになります。

マッピングモードの設定・変更は qmgr(1M) コマンドで行います。詳細は「JobCenter コマンドリファレンス」を参照してください。

6.5.4. ホスト名の変更

JobCenter がインストールされているマシンのホスト名を変更する場合には、以下の点に注意する必要があります。

■ホスト名を変更したマシンが UMS マシンであった場合

1. UMSマシンで管理しているすべてのマシンをいったんマシングループから削除する
2. すべてのマシン上の nsumsmgr アカウントの.rhosts ファイルの内容を新しいホスト名に変更する
3. 各マシン上の nmapmgr の設定から UMS のマシンID を削除する
4. UMS からすべてのマシンを再登録する
5. ホスト名を変更したマシンに対してジョブを投入するジョブネットワークが存在する場合、同時にジョブネットワークの各ジョブの転送先についても、すべて再設定する

■ホスト名を変更したマシンが UMS の管理下のマシンである場合

1. ホスト名を変更したマシンをいったんマシングループから削除する
2. ホスト名を変更したマシンと連携してJobCenter を使用しているすべてのマシン上の nmapmgr の設定から、変更前のホスト名に対するマシンIDの登録を削除する
3. UMS のメンバから以前のホスト名を削除して、新しいホスト名で再登録する
4. ホスト名を変更したマシンに対してジョブを投入するジョブネットワークをもつマシンが存在する場合、同時にジョブネットワークの各ジョブの転送先についても、すべて再設定する

■ホスト名を変更したマシンを UMS で管理していない場合

1. ホスト名を変更したマシンと連携してJobCenterを使用しているすべてのマシン上で、nmapmgrを使用して変更前のホスト名に対するマシンIDの登録を削除する
2. 新しいホスト名に以前使用していたマシンIDを登録する

なお、上記いずれの場合についてもマシンID は変更しないことを前提としています。マシンID を変更する場合は、変更するマシン上のすべてのキューと、変更するマシン上のキューを転送先に指定しているほかのマシン上のキューの転送先の設定をすべて再登録する必要があります。

6.5.5. 漢字コード変換

リクエストのスクリプトファイルおよび結果ファイルのリモートへの転送時に、漢字コードを変換する機能です。

インストール時に設定した漢字コードが SJIS で、かつリモート側ホストが WindowsNT もしくは EUC のコードを利用するマシンの場合に、以下の設定ファイルを作成して該当するリモートのホスト名を記述してください。なお、次にJobCenterを再起動するまで設定内容は反映されませんのでご注意ください。

ホスト名の記述は、空白、 TAB もしくは改行コードで区切って複数設定可能です。

```
/usr/lib/nqs/codecnv.cnf
```

上記に設定するホスト名は nmapmgr に登録されているマシンのプリンシパルな名前を利用してください。プリンシパルな名前は nmapmgr サブコマンドの get name <mid> で得られます。

■このファイルはMGとSVの「役割」により必要性が決まるものではありません。あくまでも言語環境の異なるJobCenterを混在利用する場合の組み合わせとして判断するようご注意ください。

■自ホストの漢字コードの設定が EUC である場合、およびリモートのホストと自ホストともに SJIS の場合には、本ファイルの設定は行わないでください。

■上記により記述されたホストへの転送では、無条件に SJIS から EUC への変換が行われ、また上記ホストからの結果ファイルの戻りも無条件で EUC から SJIS への変換が行われるようになります。そのため、変換の必要のないホストに対する記述が設定ファイル内に存在した場合、転送したスクリプトファイル、および戻される結果ファイルの内容が不正になる場合があります。

6.6. pipeclient

パイプクライアントプログラムはリクエストの転送用プログラムです。このプログラムはパイプキューの属性として定義するものです。それぞれのパイプキューについて必ず 1 つ選択しなければなりません。パイプクライアントプログラムには

■/usr/lib/nqs/pipeclient

■/usr/lib/nqs/rrpipeclient

■/usr/lib/nqs/lbpipeclient

の 3 つがあります。

これらのプログラムは転送先の選択方法によって使い分けます。詳しくは「6.7 負荷分散環境」を参照してください。

6.7. 負荷分散環境

JobCenter　は、投入されたリクエストをネットワーク上の各ホストに分散して実行することによって効率的な処理を実現することができます。

本節ではまず負荷分散機能について説明し、これらの使い方を具体的な例を挙げて解説します。

6.7.1. 負荷分散機能概要

JobCenterにおいて、リクエストをほかのマシンに転送し実行する場合にはパイプキューを使います。パイプキューの転送先は複数指定でき、リクエストの転送時にはそのうちの 1 つが選択されて転送が行われます。このパイプキュー作成時に負荷分散機能を指定すれば、それが負荷分散パイプキューになります。

負荷分散パイプキューにリクエストを投入すると、いくつかある転送先の中から最も負荷のバランスがとれるマシンを選んでリクエストを転送することで負荷分散が行われるようになります。

負荷分散には次の2種類の方式を用意しています。

表6.2 負荷分散方式とその実現方法

負荷分散方式	実現方法
ラウンドロビン方式	各マシンの負荷を考慮せず、リクエストを目的先マシンにできるだけ均等になるように分散します。
デマンドデリバリ方式	パイプキューとバッチキューが相互に通信し、負荷状況に応じて各リクエストが最も早く実行される最適な実行先を探します。

これらの負荷分散方式は、利用されるシステムの構成や実行されるリクエストの内容などによって使い分けることができます。以下に、負荷分散を利用する際にどの方式を用いればよいかについての指針を述べます。

1. ラウンドロビン方式負荷分散

   投入されるリクエストが比較的短い時間で完了し、使用する資源の量もあまり変わらないような環境での使用に最適です。実行対象ホストとしては、バージョンの異なるJobCenter や NQS を混在して利用することが可能です。

   リクエストの処理時間や使用する資源量に大きくばらつきがあるときには　デマンドデリバリ方式を利用環境に合わせてチューニングして用いるべきです。

2. デマンドデリバリ方式負荷分散

   リクエストの処理時間や使用する資源量に大きくばらつきがある場合に使用します。

   常に最新の負荷状況を反映して、転送先を決定します。ただし本方式は、転送側 (パイプキュー側)　および実行側　(バッチキュー側)　双方ともデマンドデリバリ負荷分散機能をそなえた JobCenter が起動されていて、かつマシングループを構成していなければ使用できません。

6.7.2. ラウンドロビン方式 (rrpipeclient)

ラウンドロビン方式はパイプクライアントがキューにリクエストの投入があるたびに順番に転送先を変えて行きます。

たとえば、リクエスト ID が 147 から 153 の場合は、以下のようにリクエストを分散します。ただし、実際には利用状況によって、転送結果が異なる場合があります。

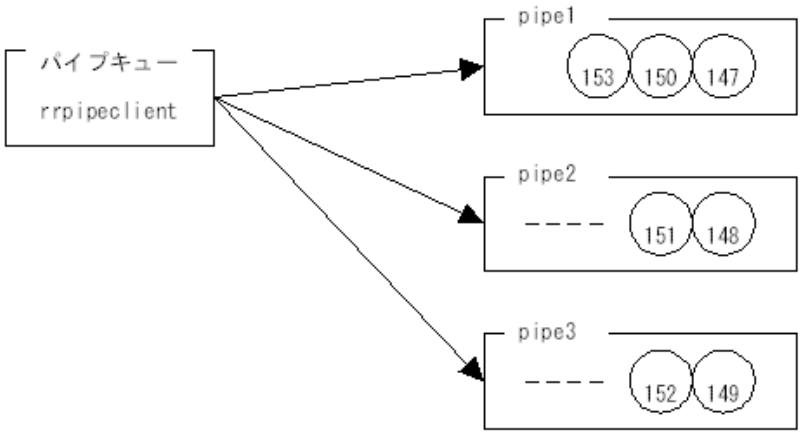


図6.3 ラウンドロビン方式のリクエスト転送イメージ

rrpipeclientを使用したパイプキューを作成する際は、serverとしてUNIX版の場合は/usr/lib/nqs/rrpipeclient、Windows版の場合はrrpipeclientを指定してください。

(UNIX版の場合)

```
Mgr: create pipe_queue pipeR priority=10 run_limit=2 \
     server=(/usr/lib/nqs/rrpipeclient) \
     destination=(rpipe@host1,rpipe@host2,rpipe@host3) ↵
```

(Windows版の場合)

```
Mgr: create pipe_queue pipeR priority=10 run_limit=2 \
     server=(rrpipeclient) \
     destination=(rpipe@host1,rpipe@host2,rpipe@host3) ↵
```

ラウンドロビン方式の転送を解除する場合は、serverとしてUNIX版の場合は/usr/lib/nqs/pipeclient、Windows版の場合は<JobCenterインストールディレクトリ>\bin\NSpipecl.exe を指定してください。

(UNIX版の場合)

```
Mgr: set pipe_client=(/usr/lib/nqs/rrpipeclient) pipeR ↵
```

(Windows版の場合)

```
Mgr: set pipe_client=(C:\JobCenter\SV\bin\NSpipecl.exe) pipeR ↵
```

6.7.3. デマンドデリバリ方式

デマンドデリバリ方式は、パイプキューとバッチキューがお互いに通信することで、リクエストを適切なタイミングで適切なバッチキューに転送し、負荷を分散することができます。

デマンドデリバリ機能は、リクエストを送る側を LOAD-BALANCE パイプキュー (以下 LB-PIPE)、そしてリクエストを受ける側のバッチキューを LOAD-BALANCE バッチキュー (以下LB-BATCH) に指定した場合に使用することができます。

このときのデマンドデリバリ機能の基本的な動作は次のようになります。

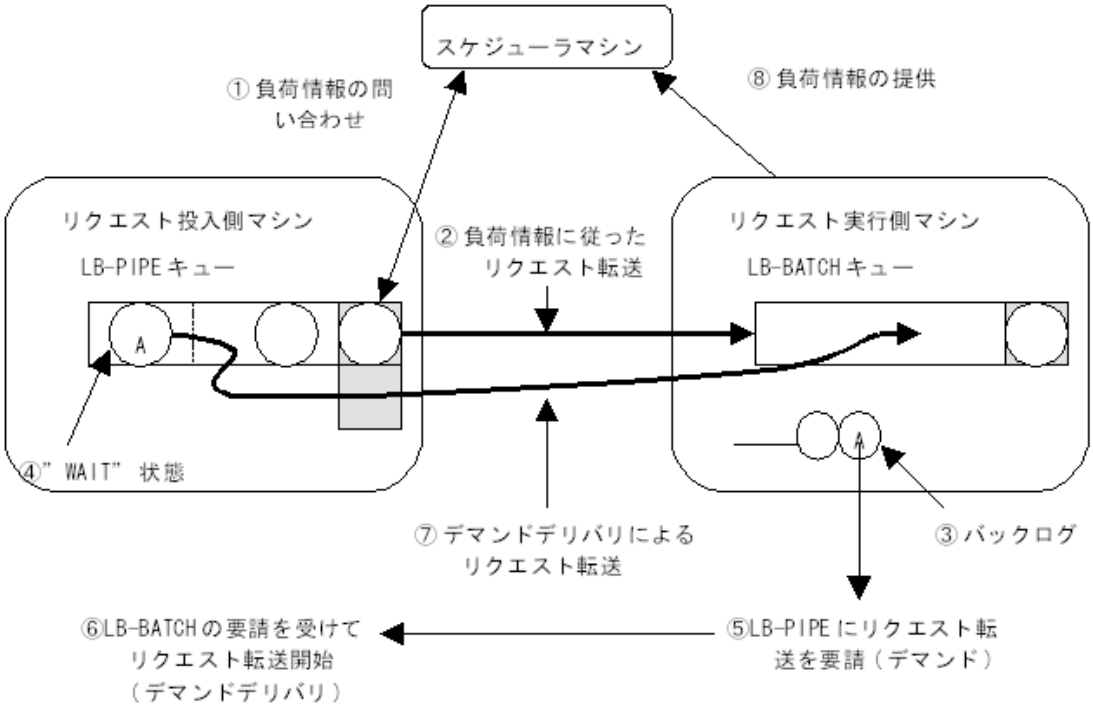


図6.4 デマンドデリバリ機能の構成と動作

LB-PIPE は、通常のリクエスト転送時にはスケジューラマシンに負荷情報を問い合わせ （ 図の ①)、リクエストの転送を試みる順序を決定し転送を行います (「負荷順序転送」と呼びます。図の ②)。

LB-BATCH 側では、リクエストの資源制限、ユーザ制限、キューの状態 (DISABLE 状態など) による制限のほかに、自マシンの状況に応じてリクエストの受け付けを制限します。このうち、最後のケースで制限が行われた状態を MACHINE-BUSY状態といいます。

この転送でリクエストがどこかの LB-BATCH キューに投入されたときは、そこでこのリクエストの転送は終りになります。そうでないときには次の 2 つの場合で動作が異なります。

■MACHINE-BUSY 状態以外の理由でリクエストが受け付けられない場合

通常のパイプキューと同様の動作をします(リクエストの消去、または一定時間後にリトライ)。

■MACHINE-BUSY 状態でリクエストが受け付けられない場合

そのリクエストのIDが LB-BATCH 側にバックログとして記録されます(図の③)。またリクエストは LB-PIPE キュー上に WAIT 状態で止まります (図の ④)。

LB-BATCH キュー側の処理が進んで、バックログに記録された先程のリクエストが受け付け可能な状態になった場合 (図ではリクエスト "A" です)、 LB-BATCH は LB-PIPE にこのリクエストを転送するように要求を出します (図の ⑤)。

要求を受けた LB-PIPE は、そのリクエストをいったん WAIT 状態から QUEUED 状態に戻し、通常よりも優先的なスケジュールを行ってすばやくリクエストをその LB-BATCH に転送します。これがデマンドデリバリです (図の ⑥ および ⑦)。

また、 LB-BATCH キュー上のリクエストやキューの状態に変化があった場合 (たとえば実行中リクエストの完了、キューの Run limit 値変更) はスケジューラマシンにイベントとして通知され、スケジューラマシンが常に最新の負荷状態を把握できるようにします (図の ⑧)。(ただし、この通知が行われるためにはマシングループを組んでいる必要があります)

ここで、LB-BATCH側にそのリクエストに関するバックログが存在しないと「LB-PIPEキュー上にリクエストがあってもデマンドデリバリの対象にはならない」ということに留意してください。

また負荷が比較的低く、LB-PIPEからのリクエスト転送時にすぐに実行受け付け可能なLB-BATCHキューを持つ実行マシンがいくつか存在するような状況では、通常1回目の負荷順序転送でリクエストは受け付けられると考えられます。

これに対し、負荷が高くて転送要求時には実行できるLB-BATCHが見つからないが、どこか1つでも空きしだいすぐに転送して実行したい、というような場合にデマントデリバリ方式は非常に有効です。

次に各モジュールの設定方法を解説します。

6.7.3.1. LB-PIPE の設定

LB-PIPE について下記の設定を行います。

1. マシングループの設定

   スケジューラマシンに負荷の問い合わせを行うときにこの設定が必要になります。もしマシングループが設定されていない、またはスケジューラマシンが存在しない場合には負荷順序ではなくデフォルトの固定された転送先順序で転送を試みます。

   マシングループの設定方法は「6.7.6 マシングループ/ スケジューラマシン」を参照してください。

2. パイプキューに LOAD-BALANCE 特性を与える。

   指定されたパイプキューを LB-PIPE として機能することを宣言します。

3. WAIT 時間の設定 (Destination wait)

   リクエストの転送先が MACHINE-BUSY 状態のときに、そのリクエストを WAIT 状態にして LB-PIPE に再度キュー登録します。このときに WAIT 状態で止まっている時間をキューごとに設定します。デフォルトは 1 時間です。

   この設定時間が経過し、 WAIT 状態の待ち合わせが終わると、リクエストは再びQUEUED状態となり負荷順序転送を試みます。これは通信障害や、マシンダウンなどでLB-BATCH側のバックログが消去されたときに、 LB-PIPE 上でリクエストが留まり続けることを回避する目的があります。

   また、 LB-PIPE のあるJobCenter を再立ち上げした場合、この WAIT 状態は解除され負荷順序転送が行われます。

4. Run limit の確保数 (Reserve run limit) の設定

   Run limit で設定された Running 数の一部をデマンドデリバリ専用に確保しておくことで優先的なスケジュールを実現しています。

   Reserve run limitが設定されない場合、デマンドデリバリのリクエストは通常の転送と同じように、自分の転送順序を待たなければいけません。 (この場合、 LB-BATCH 側がリクエストの待ち合わせをタイムアウトしてしまうかもしれません)。デマンドデリバリの機能を正常に使うには最低でも Run limit を 2 以上、 Reserverun limit を 1 以上に設定する必要があります。

   Reserve run limit には必ず(Runlimit - 1)以下の数を指定するようにしてください。

5. LB-PIPE の転送先キュー

   LB-PIPE で使用可能な転送先のキューは次の 2 つです。

   ■LB-BATCH キュー

■透過型パイプキュー（このパイプキューの転送先にはLB-BATCHキューを指定すること）

これ以外のキューを使用した場合、本機能を使用することはできません。

以下、qmgr のサブコマンドによる LB-PIPE の設定手順を説明します。まず通常の方法でパイプキューを作成します。

```
Mgr: create pipe LBPIPE priority=10 server=(/usr/lib/nqs/pipeclient) \
destination=(LBBATCH1@machine1, LBBATCH2@machine2, LBBATCH3@machine3) run_limit=2 ↵
```

(R12.7以降のWindows版の場合)

```
Mgr: create pipe LBPIPE priority=10 run_limit=2 \
destination=(LBBATCH1@machine1, LBBATCH2@machine2, LBBATCH3@machine3) ↵
```

ここでは LBPIPE という名前の LB-PIPE キューを作成しました(作成直後は[DISABLED, STOPPED]の状態のため、後に適切に状態を変更してください)。このキューの転送先はある 3 つのマシン上にそれぞれある LB-BATCH キューです。また Run limit は省略すると既定値の 1 になりますが、 ここでは2 以上を指定するようにしてください。

次にこのキューを LB-PIPE として機能するように指定します。

```
Mgr: set load_balance pipe_queue LBPIPE reserve_run_limit=1 \
     destination_retry_wait=3600 ↵
```

reserve_run_limit はデマンドデリバリのために確保しておく Run limit の数を指定します。省略時は 0 ですが、必ず 1 以上で、かつパイプキュー作成時に指定したRun limit未満になる数値を指定してください。

destination_waitはMACHINE-BUSY 状態のときにリクエストが WAIT 状態のままキューに止まっている時間です。秒単位で指定します。省略時は 3600 秒 (1 時間) です。

reserve_run_limitとdestination_retry_waitオプションは必ず同時に指定して下さい。片方を省略した場合、省略されたオプションは既定値を指定したとみなしてリセットされますのでご注意ください。

以上で LB-PIPE の設定は完了です。上記の LB-PIPE のパラメータを変更したいときは同じコマンドで設定し直してください。設定の確認は qstat -x コマンドなどで可能です。

また LB-PIPE の設定を解除したいときは次のサブコマンドを指定してください。

```
Mgr: set no_load_balance pipe_queue LBPIPE ↵
```

6.7.3.2. LB-BATCH の設定

LB-BATCH について下記の設定を行います。

1. マシングループの設定

   JobCenter の負荷状況の変化にあわせて、スケジューラマシンに負荷情報を通知します。

   マシングループの設定方法は「6.7.6 マシングループ/ スケジューラマシン」を参照してください。

2. バッチキューに LOAD-BALANCE 特性を与える。

   指定されたバッチキューを LB-BATCH として機能することを宣言します。

3. リクエスト転送の待ち合わせ時間 (Deliver wait)

   LB-PIPE にあるリクエストの転送を指示してから (デマンドデリバリ)、そのリクエストの到着を待ち合わせるときの最大待時間を指定します。デフォルトは 30 秒です。

   このデマンドデリバリのリクエストを待ち合わせている間は、そのリクエストを優先的に受け付けます。つまり、待ち合わせ中にほかのリクエストが転送されてきても受け付けません。この待ち合わせ時間を超えると、そのリクエストの優先的な受付の待ち合わせが解除され、 LB-BATCH のバックログにあるほかのリクエストのデマンドデリバリを行います。

   この最大待時間を超えたリクエストは転送不能状態にあるとみなされて、そのバックログが消去されます。

4. リクエスト保有数制限 (Keep request limit)

   LB-BATCH ではキュー上に QUEUED 状態で保有するリクエスト数の上限を指定することができます。

   0個が指定された場合、またはQUEUED状態のリクエスト数とそのキューに転送中のリクエスト数の合計がリクエスト保有数制限に達した場合は 、キューがRUNNING (INACTIVE) 状態で Run limit に余裕があり、受け付けたリクエストがすぐに実行可能ならばリクエストを受け付けます。

   Run limit に余裕があるケースとしては、同時実行ユーザ数制限に掛かったあるユーザからのリクエストが QUEUED 状態になっており、 Run limit には達していないという場合が考えられます。

   リクエスト保有数制限によってリクエストを受け付けなかった場合は、 MACHINE-BUSY 状態でリクエストを受け付けなかったと LB-PIPE に通知します (現在は、MACHINE-BUSY になるのはこの条件を満たしているときだけです)。

   リクエスト保有数のデフォルトは 0 個です。もしリクエストの転送を待つ間のアイドリング状態 (すなわち、実行中リクエストが完了してからバックログをチェックして、 LB-PIPE にリクエストの転送を要求し、転送されて次の実行が始まるまでの状態)が気になる場合には 1 以上の値を指定するか、 LB-BATCH キューの Run limit を大きくしてください。

   またリクエスト保有数制限は絶対的なものではなく、リクエストを受け付けたタイミングによっては、一時的に制限を超えることもあります。

5. バックログの記録

   LB-BATCH は MACHINE-BUSY 状態でリクエストを受け付けなかった場合にのみ、そのリクエストをバックログに記録します。したがってバックログを記録したくないときにはキューの状態を DISABLE にしてください。

   また バックログは、次の場合に消去されます_。

   ■LB-BATCH 側の nqsdaemon がシャットダウンしたとき

   ■対応するリクエストの転送が完了したとき

   ■デマンドデリバリに失敗したとき

以下、qmgr のサブコマンドによる LB-BATCH の設定手順を説明します。まず通常の方法でバッチキューを作成します。

```
Mgr: create batch LBBATCH priority=10 ↵
```

ここでは LBBATCH という名前の LB-BATCH キューを作成しました(作成直後は[DISABLED, STOPPED]の状態のため、後に適切に状態を変更してください)。次にこのキューを LB-BATCH して機能するように指定します。

```
Mgr: set load_balance batch_queue LBBATCH keep_request_limit=1 deliver_wait=30 ↵
```

keep_request_limit はリクエスト保有数制限です。省略時は 0 です。ここでは 1 を指定し、リクエスト転送中のアイドリングを少なくするようにしています。 deliver_wait はデマンドデリバリでの最大転送待時間です。デフォルトは 30 秒です。

以上で LB-BATCH の作成は完了です。設定の確認は qstat -x コマンドなどで可能です。

また LB-BATCH の設定を解除したいときは次のサブコマンドを指定してください。

```
Mgr: set no_load_balance batch_queue LBBATCH ↵
```

6.7.3.3. 透過型パイプキューの設定

透過型パイプキュー（Transparent Pipe queue, 以下 TPIPE と呼びます）は、 LB-PIPE と資源制限分けされたローカルの複数の LB-BATCH とをつなぐために使用されます。 TPIPE については「5.5 透過型パイプキューの概要と設定方法」を参照してください。

デマンドデリバリの構成の中で TPIPE を使用する場合は次のことに気をつけてください。

■TPIPE の転送先には同一マシン上にある LB-BATCH キューのみを指定してください。

■TPIPE の状態を DISABLE 状態にしても、その転送先にある LB-BATCH へのデマンドデリバリによるリクエストの転送は行われます。これは LB-PIPE と LB-BATCHとの間で直接リクエストの転送を行うためです。

■TPIPE の状態を DISABLE 状態または STOP 状態にした場合、その先にある LB-BATCHにはリクエストが到達しないため、デマンドデリバリ用のバックログが生成されません。

6.7.4. デマンドデリバリ方式による構築例

本節では、デマンドデリバリ方式の負荷分散環境の構築方法を例にそって説明します。

6.7.4.1. 構築例

下の図はデマンドデリバリ機能を用い、リクエストをネットワーク環境下で分散実行する場合のキューの例です。

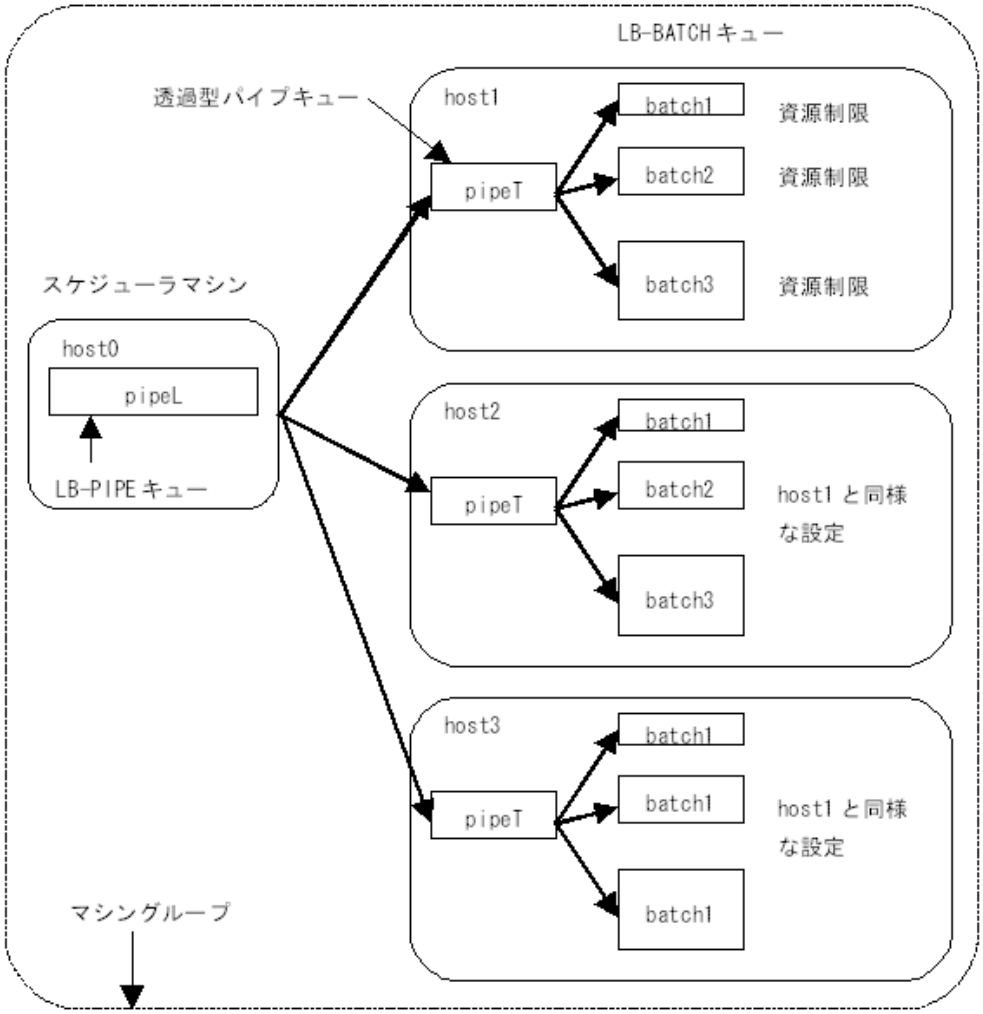


図6.5 デマンドデリバリ方式の利用イメージ

ここでは host0 ～ host3 という 4 台のマシンで負荷分散を行うことにします。リクエストは host0 から投入することにします。また、ここではマシングループを設定しています。host0 ～ host3 は同一グループ内のマシンで host0 がスケジューラマシンとして機能していることにします。

リクエストはまずデマンドデリバリ用のパイプキュー (LB-PIPE キュー) である pipeL に投入されます。このキューに投入されたリクエストは host1～ host3 のいずれかのマシンに分散して転送されます。

リクエストを受けるキューには透過型のパイプキューを用いています。これはリクエストに設定された資源制限によって、自動的にLB-BATCH キューを選択して転送するために使います。

LB-BATCH　はデマンドデリバリ機能を使用するために通常のバッチキューの代わりに使用します。

それぞれのキューの設定手順について説明していきます。

■LB-PIPE キューの作成

デマンドデリバリ機能の転送元のパイプキューとして pipeL を host0 に作成します。

ここでは同時実行数 (run_limit) は 2 を指定しています。そしてデマンドデリバリ転送用にそのうちの 1 つを確保 (reserve_run_limit) しています。デマンドデリバリを使用する場合には必ずこの設定を行ってください。

```
Mgr: create pipe_queue pipeL priority=10 run_limit=2 \
     server=(/usr/lib/nqs/pipeclient) \
     destination=(pipeT@host1,pipeT@host2,pipeT@host3) ↵
Mgr: set load_balance pipe_queue pipeL reserve_run_limit=1 ↵
```

(R12.7以降のWindows版の場合)

```
Mgr: create pipe_queue pipeL priority=10 run_limit=2 \
     destination=(pipeT@host1,pipeT@host2,pipeT@host3) ↵
Mgr: set load_balance pipe_queue pipeL reserve_run_limit=1 ↵
```

destination の部分は転送先のマシンおよびキューを指定しています。各マシンの転送先キュー pipeT には、透過型パイプキューを指定してあります。もし、各マシン上に 1 つのバッチキューしか設定しないならば直接バッチキューに転送してもかまいません。

なお、キューの状態は作成された段階では DISABLE, STOPPED の状態ですので、次のコマンドでキューを稼働状態にしてください。（この作業はこの節で作成する全キューについて行ってください。以後この手順は省略します）

```
Mgr: enable queue pipeL ↵
Mgr: start queue pipeL ↵
```

■LB-BATCH キューの作成

デマンドデリバリ機能を用いるバッチキューとして batch1 ～ batch3 を作成します。それぞれ資源制限の異なるバッチキューにすることにします。このように異なる大きさの資源制限をバッチキューに施すことによって、リクエストの使用する資源特性にあわせたスケジューリングを行うことが可能になります。

この例では簡略化のためにメモリの使用量についてのみ資源制限を施します。またキューにはその資源量に応じて、同時実行数を変えてスケジューリングの優劣をつけることにします。

次のように3 つの LB-BATCH キューを作成します。これは host1 ～ host3 でそれぞれ同じように設定してください。

```
Mgr: create batch_queue batch1 priority=10 run=3 ↵
Mgr: set load_balance batch_queue batch1 ↵
Mgr: set per_process memory_limit=(1mb) batch1 ↵
Mgr: create batch_queue batch2 priority=10 run=2 ↵
Mgr: set load_balance batch_queue batch2 ↵
Mgr: set per_process memory_limit=(2mb) batch2 ↵
Mgr: create batch_queue batch3 priority=10 run=1 ↵
Mgr: set load_balance batch_queue batch3 ↵
Mgr: set per_process memory_limit=(3mb) batch3 ↵
```

■透過型パイプキューの作成

上記の複数のバッチキューの 1 つを自動的に選択してリクエストを投入するために透過型パイプキューを用います。透過型パイプキューの作成は次のように行います。

```
Mgr: create pipe_queue pipeT priority=10 run_limit=1 \
     server=(/usr/lib/nqs/pipeclient) \
     destination=(batch1,batch2,batch3) ↵
Mgr: set transparent pipe_queue pipeT ↵
```

(R12.7以降のWindows版の場合)

```
Mgr: create pipe_queue pipeT priority=10 run_limit=1 \
     destination=(batch1,batch2,batch3) ↵
```

```
Mgr: set transparent pipe_queue pipeT ↵
```

以上の設定で pipeT が、透過型パイプキューとして機能するようになります。

destination は転送先バッチキューを指定しています。ここでは転送先として指定する順序がたいへん重要になってきます。透過型パイプキューはリクエストの転送をこの destination で指定された順序に従って試みていきます。

この例の場合、リクエストに設定された資源制限値とキューに設定された資源制限値を比較して、そのリクエストの制限値の方が小さいか、指定された値がなければそのバッチキューに投入されます。

つまり、投入を試すキューの順序は制限の厳しいバッチキューから行わなければいけないということです。

■マシングループ・スケジューラマシンの作成

最後にマシングループの設定を行います。これは LB-PIPE で負荷分散性能を向上させるために必要な設定です。次のコマンドで host0 ～ host3 に同じように設定してください。

```
Mgr: set machine_group=(host0, host1, host2, host3) ↵
```

上記の設定で、 host0 ～ host3 のマシンがグループを形成していることになります。この場合、最初に指定した host0 がスケジューラマシンとして機能することになります。

以上で、上図に示したようなキュー作成が完了しました。次のように qstat(1) コマンドを用いてキュー構成を確認してください。下記の出力結果は、実際の表示内容を省略してあります。

マシン host0 のキューを表示します。

```
# qstat -x ↵
pipeL@host0;  type=PIPE;  [ENABLED, INACTIVE];  pri=10
  0 depart;  0 route;  0 queued;  0 wait;  0 hold;  0 arrive;
  Run_limit = 2;
  Reserved_run_limit = 1;
  Unrestricted access
  Load_balance
  Queue server: /usr/lib/nqs/lbpipeclient
  Destset = {pipeT@host1, pipeT@host2, pipeT@host3};
```

マシン host1 のキューを表示します。

host2,host3 についても同様な表示が得られます。

```
# qstat -x ↵
pipeT@host1;  type=PIPE;  [ENABLED, INACTIVE];  pri=10
  0 depart;  0 route;  0 queued;  0 wait;  0 hold;  0 arrive;
  Run_limit = 1;
  Unrestricted access
  Transparent
  Queue server: /usr/lib/nqs/pipeclient
  Destset = {batch1@host1, batch2@host1, batch3@host1};

batch1@host1;  type=BATCH;  [ENABLED, INACTIVE];  pri=10
  0 exit;  0 run;   0 stage;   0 queued;  0 wait;  0 hold;  0 arrive;
  Run_limit = 3;
  Unrestricted access
  Load_balance
  Per-process memory size limit = 1 megabytes
```

```
batch2@host1;  type=BATCH;  [ENABLED, INACTIVE];  pri=10
  0 exit;  0 run;   0 stage;   0 queued;  0 wait;  0 hold;  0 arrive;
  Run_limit = 2;
  Unrestricted access
  Load_balance
  Per-process memory size limit = 2 megabytes

batch3@host1;  type=BATCH;  [ENABLED, INACTIVE];  pri=10
  0 exit;  0 run;   0 stage;   0 queued;  0 wait;  0 hold;  0 arrive;
  Run_limit = 1;
  Unrestricted access
  Load_balance
  Per-process memory size limit = 3 megabytes
```

6.7.4.2. リクエストの投入と操作

それでは前節で作成したキューに、リクエストを投入してみましょう。 pipeL に 3 個のリクエストを投入してみます。ただし、 3 つめのリクエストには 2.5MB のメモリサイズ制限を指定します。

```
# qsub -q pipeL job1 ↵
Request 1.host0 submitted to queue: pipeL.

# qsub -q pipeL job2 ↵
Request 2.host0 submitted to queue: pipeL.

# qsub -lm 2.5mb -q pipeL job3 ↵
Request 3.host0 submitted to queue: pipeL.
```

では、投入したリクエストが、どこで実行されているか確認してみます。リクエストの確認には qstatr(1) コマンドを使います。ただし、負荷分散の場合リクエストがどこのマシンに転送されるかわからないので、 qstatr(1) コマンドに -t オプションをつけてください。

そうすると自動的に転送先のマシンを探し出して状態を表示します。ただし、以下の表示では投入したリクエストがまだ実行状態であると仮定しています。

```
# qstatr -t 2 ↵
================================================
NQS (R11.10) BATCH REQUEST  HOST: host1
================================================
REQUEST ID       NAME     OWNER    QUEUE   PRI NICE STT  PGRP  R
-------------- -------- -------- -------- ---- --- --- ------ -
1.host0          job1    root     batch1     31   0 RUN        -
----------------------------------------------------------------
================================================
NQS (R11.10) BATCH REQUEST  HOST: host2
================================================
REQUEST ID       NAME     OWNER    QUEUE   PRI NICE STT  PGRP  R
-------------- -------- -------- -------- ---- --- --- ------ -
2.host0          job2    root     batch1     31   0 RUN        -
----------------------------------------------------------------
================================================
NQS (R11.10) BATCH REQUEST  HOST: host3
================================================
REQUEST ID       NAME     OWNER    QUEUE   PRI NICE STT  PGRP  R
-------------- -------- -------- -------- ---- --- --- ------ -
3.host0          job3    root     batch3     31   0 RUN        -
----------------------------------------------------------------
```

この場合、リクエストはうまく分散され実行されているようです。

負荷分散パイプキューに投入したリクエストは、投入後どこのマシンに行くかわかりません。しかし、どこのマシンに転送されたかは投入元のマシンに記録されています。そしてリクエストを操作する各コマンドはこの記録を読んで、そのマシンに操作要求を転送します。

上記例での 1 つめのリクエストをキャンセルしたい場合には、 host0 で次のように入力します。

```
# qdel -k 1.host0 ↵
```

これで、 1.host0 がどのマシンに転送されていてもキャンセルすることができます。

6.7.5. データファイルの転送について

負荷分散においてリクエストを実行する場合、分散先すべてのマシンにリクエストを実行できる環境が整っている必要があります。つまり、投入したリクエストを実行するために、アプリケーションやデータファイルが必要な場合、転送先すべてのマシンにそれらが存在しなければなりません。

あらかじめすべてのマシンにファイルを置くことができない場合は、以下の方法でファイルを転送します。

1. NFS マウントを利用する

   NFS 機能によって、分散先のすべてのマシンから同じパス名でアクセスできるディレクトリをあらかじめ作っておき、必要なデータはすべてここに置くことにします。

   たとえば、リクエストを実行するために必要なファイルを host0 のディレクトリ /home/nec/apに置くことにします。そして、ほかのマシンの /home/nec/ap に host0 の同じディレクトリをNFS マウントします。そうすれば host0 の /home/nec/ap に存在するファイルを、ほかのマシンからもアクセス可能になります。

2. rcp を利用する

   NFS を利用できない場合には rcp コマンドによって実行時にファイルをコピーします。

   たとえば、 host0 の /home/nec/data というファイルを実行マシンに転送するには、リクエストのシェルスクリプトのはじめに次のように記述します。

   ```
   rcp host0:/home/nec/data .
   ```

   この方法で、結果ファイルを書き戻すこともできます。

   ```
   rcp data host0:/home/nec/data
   ```

6.7.6. マシングループ/ スケジューラマシン

デマンドデリバリ方式の負荷分散機能を実現するために、リクエストの転送元、実行先、およびスケジューリングの各マシンすべてを 1 つの管理単位として扱うのが、マシングループ機能です。

しかしながら本機能は負荷分散のためだけに利用されるわけではなく、複数マシンで構成される JobCenter システムの運用で、 1 台のマシンからシステム全体の動作状況を集中的に管理把握する目的で広く使用することができます。

たとえばマシングループ内で発生する各種イベントを集中管理するのに使用できます。UMS(1.2 章参照) による統合管理はこの機能を使用しています。

6.7.6.1. 動作について

マシングループを定義した場合の動作について説明します。マシングループで定義できるマシンは、デマンドデリバリ機能が利用可能なマシンです。マシングループの定義のなかで、もっとも優先順位の高いマシンがスケジューラマシンとして機能するように選択されます。

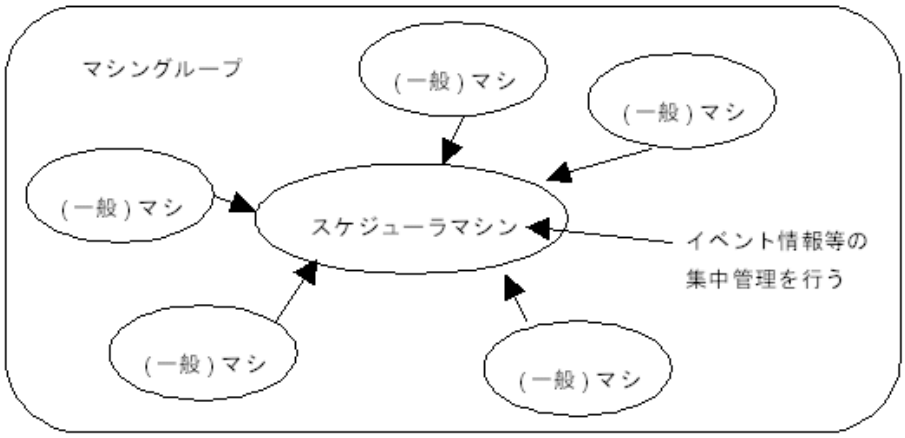


図6.6 マシングループの構成イメージ

マシングループが設定されると、グループ内のマシンはスケジューラマシンにイベントを送ります。これらのイベントにはリクエストの移動、状態変更　　(実行開始や終了、消去など)、およびキューの状態変更 (START/STOP, ENABLE/DISABLE など) などがあります。スケジューラマシン内ではこれらの情報を記録し、ほかのマシンの要求に応じて提供します。

またスケジューラマシンには情報が集中するためほかのマシンよりも負荷がかかるということを留意しておいてください。負荷の増加量はシステムの運用形態に依存します。

6.7.6.2. 操作方法

マシングループおよびスケジューラマシンの設定方法と解除方法について説明します。

qmgr(1M) で次のサブコマンドを実行してください。

```
Mgr: set machine_group=( MAC_A, MAC_B, MAC_C ) ↵
```

上記の設定で3台のマシンからなるマシングループの設定ができました。設定したマシンリストの先頭のマシンが、スケジューラマシンとして設定されます。また、このマシンリスト内には自マシン名が必ず含まれるようにしてください。

ここでは MAC_A がこのグループ内でのスケジューラマシンになります。

グループやスケジューラマシンを変更するには、マシンの記述を変更して再びset machine_groupサブコマンドを実行してください。

マシングループを解除する場合には qmgr(1M) のサブコマンドで次のように指定してください。

```
Mgr: set machine_group=(自マシン/サイト名) ↵
```

これでマシングループが解除されます。また現在のマシングループを確認するには、やはりqmgr(1M) サブコマンドで次のように指定してください。

```
Mgr: show machine_group ↵
```

第7章 JobCenterの運用

本章では、JobCenterの運用手順について説明します。

7.1. JobCenterの起動方法

7.1.1. 通常の起動方法

JobCenter(ローカルサイト)の起動は次のコマンドで実行します。なおJobCenterデーモンの起動はスーパーユーザしか行えません(UNIXの場合)。

```
/usr/lib/nqs/nqsstart [$site-name]
```

本コマンドは、 JobCenter デーモン（/usr/lib/nqs/nqsdaemon）とイベント送信等に関連するデーモンプロセスを起動します。

本コマンドを使用せずにJobCenter デーモンのロードモジュールを直接起動した場合、関連する必要なプロセスの幾つかが起動されないため、一部の機能が使用出来なくなります。

JobCenter(ローカルサイト） はOSシステム立ち上げ時に起動されますが、システムの管理者が上記のコマンドを利用して起動することも可能です。ただし、その場合コマンドの標準出力をファイル等にリダイレクトしておかないと、コンソールまたは端末に各デーモンの出力する情報が表示されるようになります。

本コマンドはデーモンの起動設定ファイル（/usr/lib/nqs/rc/daemon.conf）を読み込み、デーモン起動におけるオプションを読み込みます。(詳細は後述)

```
/usr/lib/nqs/rc/daemon.conf
```

なお、デーモンをサイトモード(クラスタサイト)で起動する場合は、オプションの $site-name を指定します。(ただし、通常はcjcpwコマンドで起動してください)

旧バージョンにおいてデーモン起動時にコマンドラインオプションとして渡していた引数は、上記のdaemon.conf設定ファイル内のNQSDAEMON_OPT エントリにオプションを記述する事により、デーモンに引き継がれます。

例えば、各キューを停止して起動する場合には以下の様なオプションをdaemon.conf設定ファイル内に設定してnqsstart を実行します。

```
NQSDAEMON_OPT=-s
```

本オプションは、複数のオプションを指定可能です。なお、本コマンドはスーパーユーザのみ実行可能です。オプションについての詳細は「7.3 デーモン起動オプション」を参照してください。

7.1.2. 強制起動方法

前回起動時にJobCenterが正常に終了しなかった、または終了方法が正しく無いと、上記のnqsstartコマンドがエラーとなる場合があります。起動しようとするホストまたはサイトでデーモンが起動していない事を確認した後、次のコマンドで起動してください。

```
/usr/lib/nqs/nqsstart -f [$site-name]
```

本コマンドおよびオプションにより、デーモン及び関連プロセスが起動します。

補足（ホスト、またはサイトでデーモンが起動していないことの確認方法）

1. ps –efコマンドでjnwxxx, nqsxxx, NQSxxxなどの名前でgrepしてJobCenterの各デーモンのPIDおよびPPIDを確認します。
2. cjclsコマンドでクラスタサイト上のnqsdaemon のPIDを確認します。（なお、SHUT と表示される場合は、cjclsコマンドの出力結果の同じ行に表示されたサイトにおいてJobCenter が起動していないことを表しています。）

3. ホスト上のデーモンを確認する場合は、2.の結果のnqsdaemonのPIDおよびPPIDが同nqsdaemonのPIDと一致するデーモン群を、1.の結果より除いたものの存在を確認してください。サイト上のデーモンを確認する場合は、②の結果のnqsdaemonのPIDおよびPPIDが同nqsdaemonのPIDと一致するデーモン群の存在を①の結果から確認してください。

7.2. JobCenterの停止方法

JobCenter はOSシステムの shutdown 処理の過程で自動的にサービスが停止されますが、システムの停止を行わずに任意のタイミングで停止することもできます。JobCenter　の停止は次のコマンドで行います。

```
/usr/lib/nqs/nqsstop [$site-name]
```

本コマンドはJobCenter デーモン及び関連するデーモンプロセスを停止します。

本コマンドを使用せずにqmgr の shutdown コマンドやkill 等によりデーモンを停止した場合、一部のプロセスが正常に終了しません。そのためJobCenter　デーモンの再起動時までそれらのプロセスが残っていた場合、正しく動作しなくなる場合があります。

サイトモード(クラスタサイト)で起動したデーモンを停止する場合は $site-name を指定します。(ただし、通常はcjcpw -stopコマンドで停止してください)

7.3. デーモン起動オプション

nqsdaemon の起動設定ファイルの形式は以下のとおりです。

7.3.1. 名称

daemon.conf - デーモンコンフィグレーションファイル

7.3.2. パス

/usr/lib/nqs/rc/daemon.conf (共通)
/usr/spool/nqs/daemon.conf （ローカルサイト）
/usr/spool/nqs/<DB パス>/daemon.conf (クラスタサイト)

7.3.3. 説明

JobCenter で使用するデーモンの動作を設定します。ファイルは

/usr/lib/nqs/rc/daemon.conf が読み込まれた後、

/usr/spool/nqs/daemon.conf (クラスタサイトは /usr/spool/nqs/<DB パス>/daemon.conf) があれば、それが読み込まれます。

同種のパラメータが存在した場合、後に読み込まれたファイルに記述された方を優先して解釈します。

7.3.4. パラメータ

daemon.conf　で定義できるパラメータについて説明します。ここで定義されていないパラメータは一部内部で使用されているキーワードを除き、環境変数としてデーモンプロセスに引き継がれます。

```
exec=NQS|JNWCASTER|SSVM
no_exec=NQS|JNWCASTER|SSVM
```

exec の場合、指定したデーモンを起動します。 no_exec の場合、指定したデーモンを起動しません。各デーモン名とデフォルトの状態は次のとおりです。

NQS	nqsdaemonデーモンを起動します。(デフォルトexec)
JNWCASTER	jnwcasterデーモンを起動します。(デフォルトexec)
SSVM	SSVCenterで使用するデーモンを起動します。(デフォルトno_exec)

■include=<ファイル名>

指定したファイルをデーモンコンフィグレーションファイルの一部として読み込み処理します。

■local_daemon=COMPAT|OFF|SITE

ローカルサイトのデーモンの起動について指定します。

COMPAT	従来どおりにローカルのデーモンを起動します。(デフォルト)
OFF	ローカルのデーモンを起動しません。
SITE	ローカルのデーモンをサイトモードとして起動します。(クラスタサイト利用時は指定必須)

■maintenance=ON|OFF

サイトモードのデーモンの起動について指定します。

ON	サイトモード(クラスタサイト)のデーモンを起動しません。
OFF	サイトモード(クラスタサイト)のデーモンを起動します。(デフォルト)

■NQSDAEMON_OPT=<val>

nqsdaemon 起動時に -x に渡すサブオプションとして次のように指定できます。サブオプションについては getsubopt(3) を参照してください。

オプションの設定例

```
NQSDAEMON_OPT=-x ofauth=JNW,schevt=OFF
```

サブオプション	概　要
-x moerr=OFF	結果ファイル転送失敗をリクエストのエラーとして扱いません。
-x ofauth=LOOSE (デフォルト)	root　は、リモートで実行したリクエストの結果ファイルを/usr/spool/nqs/gui/root/nstrk配下以外で受信することができません。
-x ofauth=JNW	全ユーザは、リモートで実行したリクエストの結果ファイルを　/usr/spool/nqs/gui/ ユーザ名 /nstrk 配下以外で受信することができません。
-x ofauth=RESTRICT	全ユーザは、リモートで実行したリクエストの結果ファイルを受信するために、リモートのジョブを実行するための権限設定が必要です。(逆方向のネットワーク権限が必要)
-x ofauth=COMPAT	LOOSE、JNW、RESRICTいずれの制限も解除し、リモートで実行した全てのジョブリクエストの結果ファイルを任意のユーザのnstrk配下で受信します。(～R3.1の従来どおりの動作)
-x schevt=ON (デフォルト)	従来どおり、スケジューラマシンにリクエストのイベントを送信します。特別な理由がない限り、ONを指定します。
-x schevt=OFF	スケジューラマシンにリクエストのイベントを送信しません。(負荷が軽減します。)
-x errmail=ON (デフォルト)	リクエストにエラーが発生したときメールを送信します。　(従来機能)
-x errmail=OFF	リクエストにエラーが発生したときメールを送信しません。
-x reqsig=ON (デフォルト)	リクエストプロセスは、　SIGTERM　をデフォルトで無視しません。
-x reqsig=OFF	リクエストプロセスは、 SIGTERM をデフォルトで無視します。

■Windows版限定オプション

Windows版JobCenterは、初回ジョブ実行時にユーザトークンをキャッシュし、以後はキャッシュしておいたユーザトークンを再利用してジョブを実行します。そのため、初回ジョブ実行後に、ユーザの設定変更(所属グループの変更等)を行ってもJobCenterを再起動しない限り反映されません。

本オプションを指定することでジョブ実行毎にユーザトークン情報を取得しますので、ユーザ設定が即座に反映されます。ただし、ジョブ実行毎にユーザトークン情報を取得する処理が行われるので、JobCenterの実行性能がその分劣化しますので注意してください。

オプションの設定例

```
NQSDAEMON_OPT=-c
```

-c

ジョブ実行毎にユーザトークン情報を取得する動作を行います。(本オプションはWindows版JobCenterのみサポートされます。)

nqsstart/nqsstop　を利用せずにデーモンの起動、停止を行った場合、daemon.confファイルに記述された起動オプションは無効となります。

7.4. キューの運用管理

JobCenterのキューは、作成しただけではリクエストの登録・実行を行いません。キューには以下のような状態があり、この状態を変更することによりリクエストの登録・実行を可能にしたり、不可能にすることができます。

キューの状態は大別して2つあります。第1特性はキューがリクエストの登録を受け付けるかどうかに関するものです。第2特性はリクエストの実行を行うかどうかに関するものです。

1. 第1特性

   リクエストはキューが投入可能 (enabled) であり, ローカル JobCenter デーモンが稼動中のときに限り、投入可能です。

ENABLED	キューはリクエストの登録を受け付ける状態です。
DISABLED	キューはリクエストの登録を受け付けない状態です。
CLOSED	JobCenter　システム停止中です。したがって、リクエストの登録はできません。

2. 第2特性

INACTIVE	キューはリクエストの実行を行う状態です。ただし、そのキュー上のリクエストで現在実行中のものがない状態です。
RUNNING	キューはリクエストの実行を行う状態です。また、そのキュー上のリクエストで現在実行中のものがある状態です。
STOPPED	キューはリクエストの実行を行わない状態です。また、そのキュー上のリクエストで現在実行中のものもない状態です。
STOPPING	キューはリクエストの実行を行わない状態です。ただし、そのキュー上のリクエストで現在実行中のものがある状態です。
SHUTDOWN	JobCenter システム停止中です。

7.4.1. キューの運用開始/終了

キューの運用は、その状態がリクエスト実行可能状態であれば JobCenter デーモンの立ち上げ、つまりJobCenter システムの稼働と同時に開始されます。それ以外の場合は、 JobCenterシステムを立ち上げ後、その状態をGUIもしくはコマンドで変更することにより開始されます。

またJobCenter　システムの停止と同時、またはキューの状態変更により運用が停止されます。ただし JobCenter システムの停止の場合は実行中のリクエストの実行も強制終了させられますが、キューの状態変更の場合は新しいリクエストの実行がされなくなるだけで、現在実行中のリクエストはそのまま最後まで実行されることが保障されています。

なおJobCenterシステム停止時点でSTOPPEDだったキューが、JobCenterシステム立ち上げ時に自動的にキューがSTOPPEDから他の状態に変更されるようなことはありません)

個々のキューの運用の開始/終了におけるキューの状態変更については次項で説明します。ここでは全キューの運用の開始/終了について説明します。

JobCenter　システムの終了により、全キューの運用を終了することができます。また、JobCenter　システムを停止せずに全キューの運用を終了することもできます。この機能は qmgr(1M) の stop all queue サブコマンドで行います。全キューの運用の開始は、 start all queue サブコマンドで行います。

ただしこのキューの運用とは、リクエストが実行できるかどうかの第2特性に関するもので、リクエスト登録の受付の可否に関してはそのときのキュー状態によります。

7.4.2. キューの状態変更

キューの状態には 2 つの特性があることはすでに述べました。ここでは、キューの状態変更について説明します。

7.4.2.1. 第1特性の変更

キューの状態をリクエスト受付可能状態にするには、 qmgr(1M) コマンドの enable queueサブコマンドで行います。

```
Mgr: enable queue batch1 ↵
```

以上の手続きでbatch1 キューがリクエスト受付可能状態になります。つまり、EN-ABLED 状態になるということです。

キューの状態をリクエスト受付不可状態にするには、 qmgr(1M) コマンドの disable queueサブコマンドで行います。

```
Mgr: disable queue batch1 ↵
```

以上の手続きでbatch1　キューがリクエスト受付不可状態になります。つまりDISABLED状態になるということです。

7.4.2.2. 第2特性の変更

キューの状態をリクエスト実行可能状態にするには、 qmgr(1M) コマンドの start queueサブコマンドで行います。

```
Mgr: start queue batch1 ↵
```

以上の手続きで batch1 キューがリクエスト実行可能状態になります。つまり、実行中リクエストの有無により、 RUNNING か、 INACTIVE 状態のどちらかになるということです。

キューの状態をリクエスト実行不可状態にするには、 qmgr(1M) コマンドの stop queue サブコマンドで行います。

```
Mgr: stop queue batch1 ↵
```

以上の手続きで batch1 キューがリクエスト実行不可状態になります。つまり、実行中リクエストの有無により、 STOPPED か STOPPING 状態のどちらかになるということです。

7.4.3. キューのアボート

キューのアボートとはそのキュー内で実行中のリクエストをすべて強制終了させることです。
キューのアボートは qmgr(1M) コマンドの abort queue サブコマンドを用いて行います。アボートの対象となったリクエストをすべて消滅します。

```
Mgr: abort queue batch1 ↵
```

以上の手続きで batch1 キュー内で現在実行中の全リクエストが削除されます。

7.4.4. キューのパージ

キューのパージとは、キューのアボートとは逆に、そのキュー内で現在キューイング状態の全リクエストを削除することです。キューのパージは qmgr(1M) コマンドの purge queue サブコマンドで行います。

```
Mgr: purge queue batch1 ↵
```

以上の手続きで batch1 キュー内で現在キューイング状態の全リクエストが削除されます。

7.5. リクエストに関する運用管理

JobCenter 管理者には、JobCenter システム上のすべてのリクエストを強制的に管理する特権があります。一般ユーザは他人のリクエストを管理することができません。JobCenter 管理者はシステム上のリクエストを管理し、よりよいJobCenter の運用をしなければなりません。

7.5.1. リクエストの削除

キューのアボート/ パージの方法についてはすでに説明しました。これは、リクエストをキュー単位で削除する方法ですが、個々のリクエストを削除する方法もあります。

リクエストの削除は qmgr(1M) コマンドの delete request サブコマンドで行います。

```
Mgr: delete request req1 ↵
```

以上の手続きでリクエスト req1 は削除されます。

7.5.2. リクエストの保留/ 保留解除

リクエストの保留とは、リクエストを一時的にリクエスト実行のスケジューリングの対象から外すことです。この処理を行うとリクエストはホールド状態になります。

リクエストの保留はqmgr(1M) コマンドの hold request サブコマンドで行い、解除は release request サブコマンドで行います。なお、 qmgr(1M) コマンドで保留を行ったリクエストはユーザが自分で保留解除することはできなくなります。

```
Mgr: hold request req1 ↵
```

以上の手続きでリクエスト req1 はホールド状態になります。

```
Mgr: release request req1 ↵
```

以上の手続きでリクエスト req1 はホールド状態を解除されます。

7.5.3. リクエストの実行中断/ 再開

リクエストの実行中断とは、一時的にリクエストの実行を中断することです。したがって、再開を行えば、中断したところから実行が始まります。この処理をおこなうと、リクエストはサスペンド状態になります。

リクエストの実行中断は qmgr(1M) コマンドの suspend request サブコマンドで行い、再開は resume request サブコマンドで行います。

```
Mgr: suspend request req1 ↵
```

以上の手続きでリクエスト req1 は実行を一時中断され、サスペンド状態になります。

```
Mgr: resume request req1 ↵
```

以上の手続きでリクエスト req1 は実行を再開します。

7.5.4. リクエストの属性変更

リクエストには多くの属性があり、これらはユーザがリクエスト投入時に指定します。 NQS管理者はすべてのリクエストについて属性を変更することができます。したがって、運用規定に沿うようにリクエストの属性を修正することができます。

リクエスト属性の変更は qalter(1) コマンドか qmgr(1M) コマンドの modify サブコマンドを用いて行います。 qmgr(1M) コマンドには、各属性に対応する modify サブコマンドが用意されていますので、変更する属性に合わせてサブコマンドを使いわけてください。

```
Mgr: modify request pppermfile_limit=(50kb) req1 ↵
```

以上の手続きで、リクエスト req1 のプロセスごとの使用ファイルサイズ制限が 50キロバイトになります。

```
Mgr: modify request ppcpu_limit=(120) req1 ↵
```

以上の手続きで、リクエスト req1 のプロセスごとの使用 CPU 時間制限が 120 秒になります。

リクエストがバッチキューに存在する場合、そのシステムがサポートしていない資源制限値に関しては値を変更することはできません。またリクエストがパイプキュー上で転送中の場合とバッチキュー上で実行中の場合はほとんどの属性は変更できません。

7.5.5. リクエストの移動

リクエストの移動とは、リクエストを現在登録されているキューからほかのキューに移すことです。このリクエストの移動方法として2通り用意されています。

1つはキュー単位でそのキューに登録されている全リクエストを移動する方法で、もう1つは個々のリクエストを移動する方法です。ただし、実行中のリクエストは移動することができません。また、移動先のキューに定義された制限に沿わないリクエストは移動することができません。そのような場合、リクエストは元のキューに残されます。

```
Mgr: move queue batch1 batch2 ↵
```

以上の手続きでキュー batch1 に登録されているリクエストがすべてキュー batch2 に移動されます。ただしbatch2 の制限などにかかるものや、現在実行中のものは移動されません。またbatch2 の方法がリクエストの受付不可の場合は、リクエストは移動されません。

```
Mgr: move request req1 batch2 ↵
```

以上の手続きでリクエスト req1 はキュー batch2 に移動されます。

なお結果ファイル転送中のバッチリクエストを指定して、かつ移動先キューにネットワークキューを指定することにより、結果ファイルの転送先ホストを変更することができます。ただしネットワークリクエストを直接指定することはできません。

7.6. JobCenter の状態確認

7.6.1. キュー状態の確認

キュー状態の確認は、 qstat(1), qstatq(1) コマンドか、 qmgr(1M) の show queue サブコマンド、 show long queue サブコマンドで行います。以下に qmgr(1M) のサブコマンドの使用例を示します。

```
# qmgr ↵
Mgr: show queue ↵
batch1@host1;  type=BATCH;  [ENABLED, RUNNING];  pri=20
  0 exit;   1 run;   0 stage;   0 queued;   0 wait;   0 hold;   0 arrive;

       REQUEST NAME      REQUEST ID        USER  PRI    STATE    PGRP
  1:           STDIN      87.host1         user1  31   RUNNING    1598

pipe1@host1 ;  type=PIPE;  [ENABLED, INACTIVE];  pri=20
  0 depart;   0 route;   0 queued;   0 wait;   0 hold;   0 arrive;

device1@host1 ;  type=DEVICE;  [ENABLED, INACTIVE];  pri=20
  0 run;   1 queued;   0 wait;   0 hold;   0 arrive;
```

より詳細な情報が欲しいときは以下のようにします。

```
# qmgr ↵
Mgr: show long queue ↵
batch1@host1;  type=BATCH;  [ENABLED, RUNNING];  pri=10
  0 exit;  1 run;   0 stage;   0 queued;  0 wait;  0 hold;  0 arrive;
  Run_limit = 3;
  User run_limit : Unlimited     Group run_limit : Unlimited
  Cumulative system space time = 23.35 seconds
  Cumulative user space time = 8.86 seconds
  Unrestricted access
  Per-process core file size limit = UNLIMITED <DEFAULT>
  Per-process data size limit = UNLIMITED <DEFAULT>
  Per-process permanent file size limit = UNLIMITED <DEFAULT>
  Per-process memory size limit = UNLIMITED <DEFAULT>
  Per-process stack size limit = UNLIMITED <DEFAULT>
  Per-process CPU time limit = UNLIMITED <DEFAULT>
  Per-request CPU time limit = UNLIMITED <DEFAULT>
  Per-request temporary file space limit = UNLIMITED <DEFAULT>
  Per-request process number limit = UNLIMITED <DEFAULT>
  Per-request physical memory limit = UNLIMITED <DEFAULT>
  Execution nice limit = 0 <DEFAULT>

  Request    1:  Name=STDIN  Id=183.host1
     Owner=user1  Priority=31  RUNNING  Pgrp=766
  Created at Mon Mar 29 15:28:09 JST 1993
  Mail = [NONE]
  Per-proc. core file size limit = UNLIMITED <DEFAULT>
  Per-proc. data size limit = UNLIMITED <DEFAULT>
  Per-proc. permanent file size limit = UNLIMITED <DEFAULT>
  Per-proc. memory size limit = UNLIMITED <DEFAULT>
  Per-proc. stack size limit = UNLIMITED <DEFAULT>
  Per-proc. CPU time limit = UNLIMITED <DEFAULT>
  Per-req.  CPU time limit= UNLIMITED <DEFAULT>
  Per-req.  temporary file space limit= UNLIMITED <DEFAULT>
```

```
  Per-req.  process number limit = UNLIMITED <DEFAULT>
  Per-req.  physical memory limit = UNLIMITED <DEFAULT>

  Execution nice priority = 0 <DEFAULT>
  Standard-error access mode = SPOOL
  Standard-error name = netware:/home/user1/STDIN.e183
  Standard-output access mode = SPOOL
  Standard-output name = netware:/home/user1/STDIN.o183
  Shell = /bin/sh
  Umask =  22

pipe1@host1;  type=PIPE;  [ENABLED, INACTIVE];  pri=10
  0 depart;  0 route;  0 queued;  0 wait;  0 hold;  0 arrive;
  Run_limit = 1;
  Cumulative system space time = 0.28 seconds
  Cumulative user space time = 0.15 seconds
  Unrestricted access
  Queue server: /usr/lib/nqs/pipeclient
  Destset = {batch1@host1, device1@host1};

device1@host1;  type=DEVICE;  [ENABLED, INACTIVE];  pri=10
  0 run;  1 queued;  0 wait;  0 hold;  0 arrive;
  Run_limit = 1;
  Cumulative system space time = 0.00 seconds
  Cumulative user space time = 0.00 seconds
  Unrestricted access
  Devset = {dev1};

Mgr:
```

実行結果は、実行したシステムによって異なります。

7.6.2. JobCenter管理者の確認

JobCenter 管理者の確認は qmgr(1M) コマンドの show managers サブコマンドで行います。

以下に確認例を示します。

```
Mgr: show managers ↵
  root:m
  user1:m
  user2:o
```

表示される情報の意味は以下のとおりです。

■ユーザアカウントの後に ":m" が付加されているユーザはJobCenter 管理者です。

■ユーザアカウントの後に ":o" が付加されているユーザは JobCenter 操作員です。

つまり上記の例では、 root と user1 がJobCenter 管理者で、 user2 がJobCenter 操作員に任命されているということを表します。

7.6.3. JobCenter環境パラメータの確認

JobCenter 環境パラメータの確認は、 qmgr(1M) コマンドの show parameter サブコマンドで行います。以下に確認例を示します。

```
# qmgr ↵
Mgr: show parameter ↵
```

```
  Maximum global batch run_limit = 100
  Maximum global network run_limit = 50
  Maximum global pipe run_limit = 50
  Debug level =             20
  Default batch_request priority = 31
  Default batch_request queue = NONE
  Default destination_retry time = 16 seconds
  Default destination_retry wait = 300 seconds
  Default device_request priority = 31
  No default print forms
  Default print queue = NONE
  (Pipe queue request) Lifetime = 0 hours
  Default network_retry time = 16 seconds
  Default network_retry wait = 0 seconds
  Default network_retry time_out = 300 seconds
  Default stage_retry time = 259200 seconds
  Default stage_retry wait = 300 seconds
  Default expire time = 259200 seconds
  Log_file = /tmp/nqslog
  Log_file size = unlimited
  Mail account = root
  Maximum number of print copies = 2
  Maximum failed device open retry limit = 2
  Maximum print file size = 1000000 bytes
  Netdaemon = /usr/lib/nqs/netdaemon
  Netclient = NONE
  Netserver = /usr/lib/nqs/netserver
  (Failed device) Open_wait time = 5 seconds
  NQS daemon is not locked in memory
  Next available sequence number = 17
  Batch request shell choice strategy = FIXED: /usr/bin/sh
  Mapping mode = TYPE1
  Maximum batch request priority = 0
  Maximum global group submit limit = Unlimited
  Maximum global user submit limit = Unlimited
  Maximum global group run limit = Unlimited
  Maximum global user run limit = Unlimited
  Maximum IDC connection number = 32
  Qwatch event spool size = 65535
  Qwatch event expier time = 3600
  Inter Queue Scheduling mode = TYPE0
Mgr:
```

7.6.4. 有効資源制限の確認

現ホストで有効な資源制限を確認するには、 qlimit(1) コマンドか qmgr(M) コマンドの show limit サブコマンドを用います。以下に確認例を示します。

```
# qmgr ↵
Mgr: show limit ↵
  Core file size limit (-lc)
  Data-segment size limit (-ld)
  Per-process permanent file size limit (-lf)
  Per-process memory size limit (-lm)
  Stack segment limit (-ls)
  Per-process cpu time limit (-lt)
  Per-request cpu time limit = (-lT)
```

```
Per-request temporary file space limit = (-lV)
Per-request process number limit = (-lP)
Per-request physical memory limit = (-lW)
Nice value (-ln)
       .
       .
       .
```

( ) 内は qsub(1) コマンドの対応するオプションを表します。

実行結果は、実行したシステムによって異なります。

7.7. 結果ファイルの保存

JobCenterはユーザのリクエスト実行結果をリクエストに指定されたファイルに出力しますが、なんらかの理由で出力できない場合、リクエストの実行マシン上のユーザホームディレクトリに結果ファイルを作成します。

もしそれも不可能な場合、結果ファイルは　　JobCenterデータベース内に残されたままになります。そして次回JobCenter　立ち上げ時に、その残存結果ファイルは　JobCenter　データベース内の /usr/spool/nqs/private/root/outfai ディレクトリ内に、以下の形式の名前のファイルとして保存されます。

標準結果出力	outシーケンス番号.マシンID
標準エラー結果出力	errシーケンス番号.マシンID

7.8. ジョブトラッキング

7.8.1. トラッキングファイル

パイプキューに投入されたリクエストがどのマシンに転送され、実行し、終了したかという情報は、そのリクエストを投入したマシン上のファイル （トラッキングファイル） に記録されています。

リクエスト投入マシンからリクエストに対して操作コマンドを発行すると、コマンドはこのファイルを参照し、そのリクエストが存在しているマシンを自動的に探しだします。したがって、ユーザは、リクエストがどのマシン上にあるかを意識する必要はありません。

対応コマンド： qalter, qdel, qhold, qmove, qrerun, qrls, qrsm, qspnd, qwait, qstatr

ただし、 qstatr は -t level オプションも指定してください。

7.8.2. トラッキングファイルの情報保持時間

トラッキングファイルは、 qwait、ジョブネットワークといった、リクエスト実行終了の待ち合わせ処理を行う時にも使います。

終了したリクエストに関する情報は一定時間、投入元マシンに保持されます。したがって、トラッキングファイルが削除されるまでの間はリクエストの終了コードを知ることができます。もし、非常に時間のかかるジョブネットワークなどを動かす場合は、この時間をのばす必要があります。保持時間は、この待ち合わせに要する時間よりも長い時間を設定してください。

デフォルトは 3 日間 (259200 秒) です。ただし、あまりにも大きな値を設定すると、ファイルサイズが大きくなるので注意してください。

リクエストの終了情報保持期間の設定は qmgr(1M) のサブコマンド

```
Mgr: set default expire time <expire-time-in-seconds> ↵
```

で行います。

7.8.3. リクエストの存在マシン情報欠落時の復旧法

リクエストがマシンからマシンへ転送されるときに、そのリクエストを投入したマシン上の JobCenter が何らかの原因で稼働していなかった場合、リクエスト投入元のマシン上でのリクエスト存在マシン情報と実際のリクエスト存在マシンが異なる場合があります。

このようなときには、そのリクエストを投入したマシンを再起動した後、 qstatr -t 3 を実行してください。そうすると、転送先を自動的に調べて、リクエストの存在マシン情報が修正されます。

7.8.4. 旧バージョンNQS との接続時の注意

パイプキュー上のリクエストがほかのマシンに転送される際、リクエストが「リクエストの転送処理が旧 NQS から旧 NQS への転送で終了している」という条件の下で、ジョブトラッキング機能をもたないJobCenter(旧 NQS) が稼働しているマシンを経由すると、リクエストを投入されたマシンがリクエストの最終の転送先(つまりそのリクエストが存在しているマシン)を把握できなくなります。

このようなリクエストに対してトラッキング対応のユーザコマンドを実行する場合は -h オプションを用いてホスト名を明記してください。

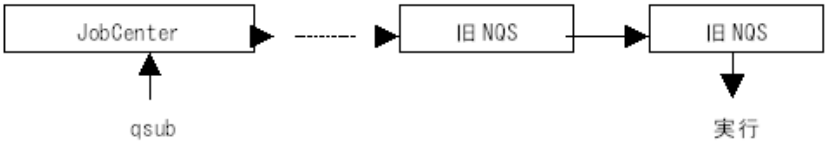


図7.1 旧バージョンのNQSとの接続イメージ

第8章 JobCenter管理者コマンド一覧

JobCenter管理者コマンド(nmapmgr、qmgr)の一覧の記載は、R12.6 第3版以降より「JobCenter コマンドリファレンス」に移行いたしました。

第9章 APIライブラリ

9.1. NQSライブラリの概要説明

9.1.1. 機能説明

NQSライブラリは、 JobCenterのNQSの各機能を C 言語から使うための関数群です。

NQSライブラリはHP-UX(IPF)版のみサポートしております。また、NQSライブラリは互換性のため現在は添付しておりますが、将来のバージョンで製品に添付されなくなる予定です。新規のご使用は特別な理由がない限りお控えください。

9.1.2. 使用方法

NQSライブラリは以下にインストールされています。

```
/usr/lib/nqs/libnqsapi.so
```

NQSライブラリをリンクしてプログラムを作成するには、 コンパイラ(cc コマンド)に以下のオプションを追加してください。

```
-L/usr/lib/nqs/ -lnqsapi
```

NQSライブラリをリンクしたプログラムを実行する前に、lnコマンドで以下のシンボリックリンクを張ってください。

```
# ln -s /usr/lib/nqs/libnqsapi.so /usr/lib/nqs/libnqsapi.so.0 ↵
```

■NQSライブラリは32ビットライブラリです。NQSライブラリとリンクするモジュールは32ビットモジュールとしてコンパイルしてください。

■NQSライブラリをリンクしたプログラムを実行するにはJobCenterがインストールされている必要があります。

9.1.3. 結果領域

9.1.3.1. 領域確保

JobCenter ライブラリ関数は、実行結果を格納するために動的な領域を確保します。この領域はライブラリ中で malloc 関数により確保するので、データを参照した後に NQSfree 関数を用いて開放してください。ライブラリ関数が返すアドレスは必ずしも malloc 領域の先頭アドレスではないので、 free 関数で直接開放することはできません。

9.1.3.2. ブロック構造

結果領域は複数の可変長ブロックにわかれています。それぞれのブロックには結果のデータが入るか、あるいはさらに下位のブロックにわかれています。ライブラリ関数の返すアドレスは最上位の先頭ブロックのアドレスです。

結果領域をアクセスするために、いくつかのマクロが用意されています。

表9.1 結果領域へのアクセス用マクロ

マクロ	動作仕様
NQSblocknext(p)	ポインタpが指すブロックの次のブロックのアドレスを返します。次のブロックがない場合には NULL を返します。

NQSblockchild(p)	ポインタpが指すブロックの下位ブロックの内で先頭のもののアドレスを返します。下位ブロックがない場合には NULL を返します。
NQSblockdata(p)	ポインタpが指すブロックの内容を指すポインタを返します。
NQSblocksize(p)	ポインタpが指すブロックのサイズを返します。
NQSblockcode(p)	ポインタpが指すブロックの参照コード番号とブロックのデータ型を返します。 参照コード番号はこの関数の戻り値とマクロ定数 NQSAPI_CODE との論理積を取ることにより得られます。 データ型についてはブロックデータフォーマットの項を参照してください。

［例］

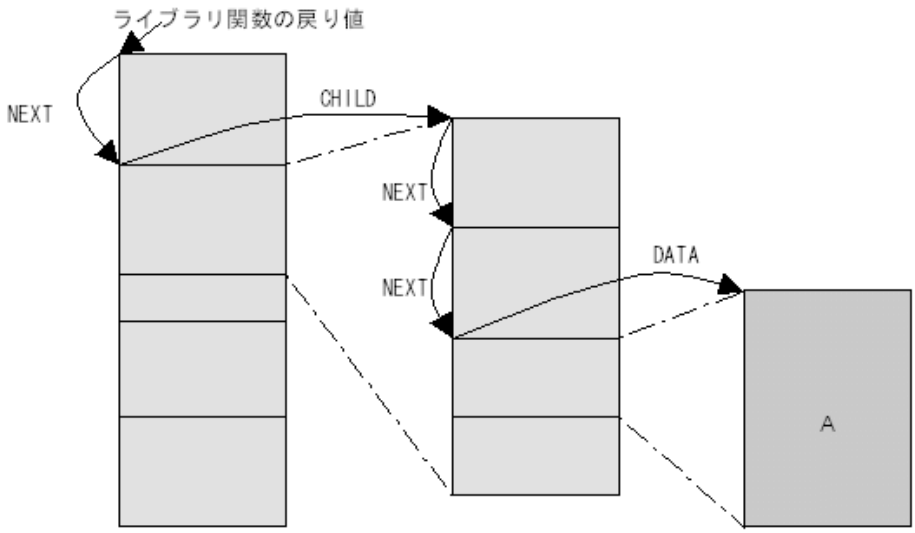


図9.1 データへのアクセスイメージ

たとえばデータ A にアクセスしたい場合には以下のようにします。

```
p = <ライブラリ関数の戻り値>;
p = NQSblocknext( p );
p = NQSblockchild( p );
p = NQSblocknext( p );
p = NQSblocknext( p );
A_ptr = (type_of_A *) NQSblockdata(p);
```

9.1.3.3. ブロックデータフォーマット

各ブロックにはそれぞれのデータ型があります。データ型には以下の種類があります。

■ブロック集合型

内容がさらに複数の下位のブロックにわけられているブロックです。

NQSblockcode() の値とマクロ定数 NQSAPI_COMP との論理積が非ゼロとなります。

■文字列型

'\0' で終る文字列です。

NQSblockcode() の値とマクロ定数 NQSAPI_STRING との論理積が非ゼロとなります。

■long 整数配列型

long　整数の配列です。内容を簡単にアクセスするために参照構造体を用意してあるものもあります。

NQSblockcode() の値とマクロ定数 NQSAPI_INT4 との論理積が非ゼロとなります。

9.1.3.4. 参照コード

各ブロックにはそれぞれのデータの内容を示す参照コードが付けられています。ブロック順序が不定である場合はこの参照コードを使ってブロックをサーチする必要があります。

9.1.4. 共通ブロック

各ライブラリ関数はそれぞれの形式で結果領域を返しますが、最初の 2 つのブロックだけは内容が決まっています。

9.1.4.1. コマンドステータス

関数のエラーコードなどが入ります。

■参照コード 1

■long 配列型

■参照構造体

```
struct NQSapistat {
         long status;
         long tcmcode;
         long total_size;
         long version;
         long reserve[4];
};
```

- status

  関数の終了ステータスです。値の意味は以下のとおりです。

値	意　味
0	正常終了です。
1	タイミング、その他の理由により関数は失敗したが、再度実行すれば正常終了する可能性がある場合です。
2	管理者により設定された制限を超えた場合、またはシステムの利用権利がない場合です。
3	オプション指定のミスなど、ユーザによる設定ミスの場合です。
4	環境設定ミスなどの、管理者による設定ミスの場合です。
5	JobCenter 自体の内部的エラーの場合です。
100	正常終了と異常終了が混在している場合です。

- tcmcode

  予約領域です。

- total_size

  結果領域全体のサイズです。

- version

  JobCenter ライブラリのバージョンです。ビット 0x0ffffL をマスクしてください。戻り値とライブラリのバージョンは以下のように対応しています。

戻り値	バージョン
0	NetShepherd R2.X
3	NetShepherd R3.X
4	NetShepherd R4.X
5	NetShepherd R5.X
6	NetShepherd R6.X
7	NetShepherd R7.X
8	NetShepherd R8.X

- reserve

  予備領域です。

9.1.4.2. メッセージ

エラーの内容を示すメッセージです。

■参照コード 2

■文字列型

9.2. NQSqalter　リクエストの属性変更

```
#include <nqsapi.h>
struct NQSblockhead *NQSqalter(char *argument);
```

9.2.1. 機能説明

引き数 argument は、 qalter コマンドに指定するのと同様の文字列を指すポインタです。この関数は、結果を動的な領域に格納し、その領域のアドレスを返します。

結果を格納する領域は、ライブラリ中で　　malloc　　関数により確保するので、利用者が後で NQSfree 関数を用いて解放する必要があります。

結果領域の各ブロックの内容は以下のとおりです。

1. コマンドステータス (intro(3NQS) 参照)

2. メッセージ (intro(3NQS) 参照)

3. 予備領域

結果領域の格納形式については intro(3NQS) で詳しく述べているので、そちらを参照してください。

9.2.2. 戻り値

それぞれの関数の戻り値は、結果領域の先頭アドレスです。結果領域の確保に失敗した場合は、NULL を返します。

-q　オプション使用時で、リモートホスト上のリクエストに関する操作を行う場合、エラーが発生しても結果コードに反映されない場合があります。

9.3. NQSq***　リクエストの操作

NQSqdel()　　リクエストの削除、シグナル送信
NQSqhold()　　リクエストの保留
NQSqrls()　　リクエストの保留解除
NQSqmove()　リクエストの移動
NQSqspnd()　実行中バッチリクエストの一時中断
NQSqrsm()　　バッチリクエストの実行再開

```
#include <nqsapi.h>
struct NQSblockhead *NQSqdel(char *argument);
struct NQSblockhead *NQSqhold(char *argument);
struct NQSblockhead *NQSqrls(char *argument);
struct NQSblockhead *NQSqmove(char *argument);
struct NQSblockhead *NQSqspnd(char *argument);
struct NQSblockhead *NQSqrsm(char *argument);
```

9.3.1. 機能説明

引き数　argument　は、それぞれのコマンド　(qalter,　qdel,　qhold,　qrls,　qmove,　qspnd,　qrsm)　に指定するのと同様の文字列を指すポインタです。この関数は、結果を動的な領域に格納し、その領域のアドレスを返します。

結果を格納する領域は、ライブラリ中で　malloc　関数により確保するので、利用者が後で

NQSfree 関数を用いて解放する必要があります。

結果領域の各ブロックの内容は以下のとおりです。(3番目以降のブロックは、リクエストごとの結果ブロックが指定されたリクエストの数だけ続きます)

1. コマンドステータス (intro(3NQS) 参照)
2. メッセージ (intro(3NQS) 参照)
3. リクエストごとの結果ブロック (ブロック集合型)
4. リクエストごとの結果ブロック (ブロック集合型)
   :

リクエストごとの結果ブロックに含まれるブロックは以下のとおりです。

1. リクエスト ID (文字列)
2. リクエストのコマンド実行ステータス (内容はコマンドステータスと同様)
3. リクエストのコマンド実行メッセージ (文字列)

結果領域の格納形式については　intro(3NQS)　で詳しく述べているので、そちらを参照してください。

9.3.2. 戻り値

それぞれの関数の戻り値は、結果領域の先頭アドレスです。結果領域の確保に失敗した場合

はNULL を返します。

-h オプション使用時で、リモートホスト上のリクエストに関する操作を行う場合、エラーが

発生しても結果コードに反映されない場合があります。

■使用例

《 NQSqdel の使用例 》

```
#include <nqsapi.h>
main()
{
        struct NQSblockhead *adr,*p;
        struct NQSapistat *sb;
        char *msg;

        adr = NQSqdel( "-k 100.machine1" );

        if( adr == NULL ){
                printf("ERROR");
                exit(1);
        }

        sb = (struct NQSapistat *)NQSblockdata( adr );

        if( sb->status > 0 ){
                p = adr;
                p = NQSblocknext(p);
                msg = (char *)NQSblockdata(p);
                printf( "%s\n", msg );
                exit(1);
        }

        printf("SUCCESS");

        NQSfree( adr );
}
```

9.4. NQSfree　結果領域の開放

```
#include <nqsapi.h>
void NQSfree(struct NQSblockhead *area);
```

9.4.1. 機能説明

NQSfree 関数は、ほかのJobCenter ライブラリ関数が確保した結果領域を開放します。

引き数 area には、各関数の戻り値として返された領域のアドレスを指定します。

9.4.2. 戻り値

なし。

9.5. NQSqstat　JobCenter 情報取得

```
#include <nqsapi.h>
struct NQSblockhead *NQSqstat(char *option);
```

9.5.1. 機能説明

NQSqstat 関数はJobCenter に関する情報を取得します。

結果は動的な領域に格納され、その先頭アドレスを関数値として返します。この結果領域はライブラリ中で malloc 関数により確保するので、利用者が後で NQSfree 関数を用いて解放する必要があります。この領域の説明は intro(3NQS) で詳しく述べているので、そちらを参照してください。

取得する情報は引き数 option で指定します。 option には、オプション説明の項にある各オプション文字列を 1 つの文字列につなげたものを指定します。

9.5.2. オプション

9.5.2.1. 詳細レベル

リクエスト、キュー、ホストについて、どの程度詳細な情報を得るかを指定します。値は 0 が情報なしで大きくなるほど多くの情報が入ります。各詳細レベルとブロックの対応は以下のとおりです。数字はブロック参照コードです。

■ホスト情報

詳細レベル	ブロック
0	101,102(空文字列),103,104,105
1	101,102,103,104,105
2	101,102,103,104,105,106,107

■キュー情報

詳細レベル	ブロック
0	201,202(空文字列),203
1	201,202,203
2	201,202,203,204,205
3	201,202,203,204,205,206,207,208,209,250,251,252,253,260

■リクエスト情報

詳細レベル	ブロック
0	なし
1	301,302,303,304,305
2	301,302,303,304,305,310,311,312,313,314
3	301,302,303,304,305,310,311,312,313,314,320,321,322,323,324,325,326,327,328,329,370,380

以下のレベルオプションを使用する場合は必ずレベル値を指定してください。

```
-hl $n
```

ホストに関する情報の情報量を選択します。 $n には 0 から 2 の数値を指定します。既定値は 1 です。

-ql $n

キューに関する情報の情報量を選択します。 $n には 0 から 3 の数値を指定します。既定値は 1 です。

-rl $n

リクエストに関する情報の情報量を選択します。 $n には 0 から 3 の数値を指定します。既定値は 1 です。

9.5.2.2. ホスト指定オプション

-hn $host-name

$host-name で指定したホストを対象にします。

-ht

ローカルホストから投入されたリクエストが存在しているホストを対象にします。

以上のオプションを指定しなかった場合は、ローカルホストを対象にします。

9.5.2.3. キュー指定オプション

-qn $queue-name

$queue-name で指定したキューを対象にします。

-qb

バッチキューを対象にします。

-qp

パイプキューを対象にします。

以上のオプションを指定しなかった場合は、対象ホスト上のすべてのキューを対象にします。

9.5.2.4. リクエスト指定オプション

-ri $request-id

$request-id で指定したリクエストを対象にします。

-rn $request-name

$request-name で指定したリクエストを対象にします。

-ru $user-name

$user-name で指定したユーザが所有するリクエストに限定します。

-rb

対象リクエストをバッチリクエストに限定します。

-rs

ローカルホストから投入されたリクエストに限定します。

-re

すでに終了しているリクエストを対象に加えます。

以上のオプションを指定しなかった場合は、対象ホストあるいは対象キュー上の終了していないすべてのリクエストを対象にします。

9.5.3. 戻り値

NQSqstat　関数の戻り値は、結果領域の先頭アドレスです。結果領域の確保に失敗した場合は、NULL を返します。

9.5.4. 結果領域

NQSqstat の結果領域には以下のように情報が入ります。

■全体はステータス、メッセージ、ホストリストの 3 つのブロックからなります。

■ホストリストはホスト情報ブロックの集合です。ホスト情報ブロックは 1 つのホストに関する情報の集合です。

■ホスト情報ブロックの中にキューリストがあります。キューリストはキューごとのキュー情報ブロックの集合です。キュー情報ブロックは 1 つのキューに関する情報の集合です。

■キュー情報ブロックの中にリクエストリストがあります。リクエストリストはリクエストごとのリクエスト情報ブロックの集合です。リクエスト情報ブロックは 1 つのリクエストに関する情報の集合です。

■ホスト詳細レベルを 0 に設定すると、複数のホストに存在する各キューは 1 つの匿名ホストの下にすべて集まる形で格納されます。同様に、キュー詳細レベルを 0 に設定すると、複数のキューに存在する各リクエストは 1 つの匿名キューの下にすべて集まる形で格納されます。リクエスト詳細レベルを 0 に設定すると、リクエストリストは空になります。

例)

ホスト host1, host2 の状態を取った場合で、ホスト host2 に、キュー queue1, queue2があり、queue1 にリクエスト 100, 101 がある‥という場合、詳細度指定は

```
-hl 1 -ql 2 -rl 1
```

とする。

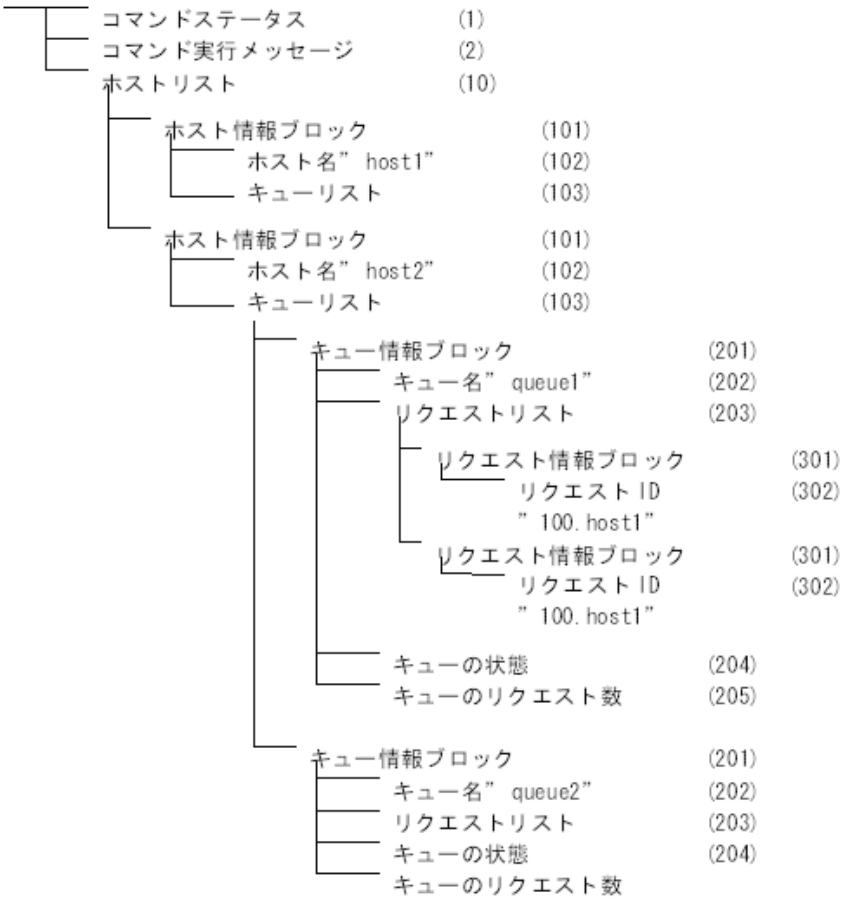


図9.2 結果領域の情報構成

() 内の数字は参照コード番号です。意味はブロックデータの項を参照してください。

9.5.5. ブロックデータ

この項では各ブロックの内容を説明します。各項目名の前にある数字はブロック参照コードです。

10

ホストリスト

ホストの集合です。複数のホスト情報ブロックからなります。

■ブロック集合型

■下位ブロック

■ホスト情報ブロック x ホスト数

101

ホスト情報ブロック

1つのホストに関する情報の集合です。各ブロックが参照コード順に並んでいます。

■ブロック集合型

■下位ブロック

102,103,104,105,106,107

102

ホスト名

文字列型

103

キューリスト

ホストに属するキューの集合です。複数のキュー情報ブロックからなります。

キュー情報レベル 0 の場合は 1 つの匿名キューが入ります。

■ブロック集合型

■下位ブロック

キュー情報ブロック (201) x キュー数

104

予備領域

■long 配列型

105

予備領域

■文字列型

106

ホストの状態

■long 配列型

■参照構造体

```
struct NQShoststat {
        long flag;
        long reserve[15];
};
```

flag	JobCenter の状態を表すフラグです。各ビットが 1 のときの意味は、以下のとおりです。	
	000001	デーモンが起動しています。
	000002	インストールされているプロダクトは NetShepherd/CLです。
	※ビット000001はデーモンが起動していることを表します。	
reserve	予備領域です。	

107

ホスト上のリクエスト数

ホスト上に存在するリクエストの数です。状態ごとにわけられています。

■long 配列型

■参照構造体

```
struct NQSreqnum {
        long total;
        long queued;
        long run;
        long wait;
        long hold;
        long arrive;
        long reserve[10];
};
```

total	リクエストの総数
queued	Queued 状態のリクエスト数
run	Running 状態のリクエスト数
wait	Waiting 状態のリクエスト数
hold	Hold 状態のリクエスト数
arrive	Arriving 状態のリクエスト数
reserve	予備領域

201

キュー情報ブロック

1つのキューに関する情報の集合です。各ブロックが参照コード順に並んでいます。参照コードが欠番になっている部分には将来ブロックが追加される可能性があるので参照コードをサーチする必要があります。

■ブロック集合型

■下位ブロック

202 ～ 260

202

キュー名

キュー情報レベル 0 の場合は匿名キューとなり、空文字列が入ります。

■文字列型

203

リクエストリスト

キューに属するリクエストの集合です。複数のリクエスト情報ブロックからなります。

■ブロック集合型

■下位ブロック

リクエスト情報ブロック (301) x リクエスト数

204

キューの属性

■long 配列型

■参照構造体

```
struct NQSqueuestatus {
        long que_type;
        long priority;
        long runlimit;
        long flags;
        union{
                struct{
                        long keep_req;
                        long dlv_wait;
                        long rsc_wait;
                        long rsc_timeout;
                } batch;
                struct {
                        long rsv_rl;
                        long dest_wait;
                } pipe;
        } v;
        long reserve[8];
};
```

que_type	キューのタイプを表す数字が入ります。1はバッチキュー、4はパイプキューを表します。	
priority	キュープライオリティーです。	
runlimit	同時実行可能リクエスト数制限値です。	
flag	キューの属性を表すビットが入ります。各ビットが1のときの意味は以下のとおりです。	
	000001	START 状態です。
	000002	ENABLED 状態です。
	000010	事前チェック機能がセットされています。
	000020	ステイウェイトがセットされています。
	002000	TRANSPIPE 属性がセットされています。
	004000	LOAD BALANCE属性がセットされています。
	020000	アクセス制限があります。
	040000	パイプオンリー属性がセットされています。
keep_req	LB バッチキューのパラメータのリクエスト保有数制限です。	
dlv_wait	LB バッチキューのパラメータのリクエスト転送待ち時間です。	
rsc_wait	Flock Manager 使用時の資源確保リトライ待ち合わせ時間です。SUPER-UXでは使用されません。	
rsc_timeout	Flock Manager 使用時の資源確保リトライ最大待ち合わせ時間です。SUPER-UX では使用されません。	
rsv_rl	パイプキューのパラメータのデマンドデリバリ時の run limit 確保値です。	
dest_wait	LB パイプキューのパラメータのデマンドデリバリ時のデスティネーションリトライ時間です。	

205

キューにエントリしているリクエストの数

■long 配列型

■参照構造体

参照コード 107 と共通。

206

キューの資源制限値

キューの資源制限値です。資源制限タイプごとの資源制限値ブロックの集合です。

■ブロック集合型

■下位ブロック

1. コアファイルサイズ制限値
2. データ領域サイズ制限値
3. プロセス・パーマネントファイルサイズ制限値
4. リクエスト・パーマネントファイルサイズ制限値
5. プロセス・クイックファイルサイズ制限値
6. リクエスト・クイックファイルサイズ制限値
7. プロセス・テンポラリファイルサイズ制限値
8. リクエスト・テンポラリファイルサイズ制限値
9. プロセス・メモリサイズ制限値
10. リクエスト・メモリサイズ制限値
11. スタック領域サイズ制限値
12. プロセスワーキングセットサイズ制限値
13. プロセス・CPU時間制限値
14. リクエスト・CPU時間制限値
15. ナイス値制限値
16. リクエスト・物理メモリ使用制限値 (SUPER-UX では使用されません)
17. リクエスト・プロセス数制限値
18. 実行優先度値

下位ブロックの総数は不定です。

207

資源制限値

資源制限の値です。資源制限のタイプによって、参照構造体が違います。

■long 配列型

■参照構造体 (サイズ制限値の場合)

```
struct NQSsizelimit {
        long flag;
        unsigned long limit;
```

```
        long unit;
};
```

flag	limit　値の指定、システムの対応などを確認するフラグです。各ビットが1のときの意味は以下のとおりです。	
	0001	limit が指定されています。
	0002	無限大が指定されています。
	0004	システムが対応しています。
limit	サイズ制限値です。	
unit	制限値の単位を表す値です。値の意味する単位は以下のとおりです。	
	0011	byte
	0012	word
	0021	Kbyte
	0022	Kword
	0031	Mbyte
	0032	Mword
	0041	Gbyte
	0042	Gword

■参照構造体 (時間制限値の場合)

```
struct NQStimelimit {
        long flag;
        unsigned long sec;
        long ms;
};
```

flag	limit値の指定、システムの対応などを確認するフラグです。
sec	時間制限値の秒単位部分を表しています。
ms	時間制限値のミリ秒単位部分を表しています。

■参照構造体 (整数制限値の場合)

```
struct NQSvaluelimit {
        long flag;
        long value;
};
```

flag	limit 値の指定、システムの対応などを確認するフラグです。
value	整数型の制限値です。

208

アクセス許可ユーザ

使用が許可されているユーザのユーザ ID が入った配列です。

■long 配列型

209

アクセス許可グループ

使用が許可されているグループのグループ ID が入った配列です。

■long 配列型

250

関係キュー複合体リスト

キュー複合体名のリストです。文字列型のキュー複合体名の集合です。

■ブロック集合型

251

転送先キュー名リスト

リクエスト転送先キュー名のリストです。文字列型のキュー名の集合です。

■ブロック集合型

252

サーバ名

サーバプログラムのパス名です。

■文字列型

260

積算使用時間

■long 配列型

■参照構造体

```
struct NQScumulativetime {
        time_t system_time;
        time_t user_time;
};
```

system_time	システム領域の積算使用時間です。
user_time	ユーザ領域の積算使用時間です。

301

リクエスト情報ブロック

1つのリクエストに関する情報の集合です。各ブロックが参照コード順に並んでいます。参照コードが欠番になっている部分には将来ブロックが追加される可能性があるので参照コードをサーチする必要があります。

■ブロック集合型

■下位ブロック

302 ～ 380

302

リクエスト ID

■文字列型

303

リクエスト名

■文字列型

304

リクエストの属性 (詳細レベル 1)

■long 配列型

■参照構造体

```
struct NQSreqattr1 {
        long   orig_seqno;
        mid_t  orig_mid;
        mid_t  existmid;
        uid_t  owner_qstat;
        long   reserve[4];
};
```

orig_seqno	リクエスト ID の数字部分です。
orig_mid	リクエスト投入マシンのマシン ID です。
existmid	リクエスト存在マシンのマシン ID です。
owner_qstat	qstat 実行マシン上での、リクエストユーザ ID です。不明の場合には -1 が入ります。

305

リクエスト存在マシン名

■文字列型

310

所属キュー名

リクエストの所属しているキュー名です。

■文字列型

311

リクエストの属性 (詳細レベル 2)

■long 配列型

■参照構造体

```
struct NQSreqattr2 {
        long   priority;
        long   state;
        long   process_family;
```

```
        uid_t  owner_exist;
        time_t starttime;
        long   reserve[11];
};
```

priority	リクエストプライオリティです。	
state	リクエストの状態を表す数値です。数値の意味は以下のとおりです。	
	0000002	Running
	0000010	Queued
	0000020	Waiting
	0000040	bytHoldinge
	0000100	Arriving
	0010000	Exited
	0020000	Routed (正確な状態は不明)
	0100002	Suspending
	0200002	Routing
process_family	実行中リクエストのプロセスグループです。	
owner_exist	リクエスト存在マシン上での、リクエストユーザ ID です。不明の場合には -1 が入ります。	
starttime	スケジューリング開始日時です (実行開始前のリクエストのみ)。	

312

リクエストの終了情報

リクエストの終了理由や、終了時刻が入ります。内容はすでに終了したリクエストの場合のみ入ります。

■long 配列型

■参照構造体

```
struct NQSreqexitstate {
        long reason;
        long exitcode;
        mid_t execmid;
        time_t starttime;
        time_t endtime;
        long reserve[3];
};
```

reason	リクエストの終了理由です。 qwait コマンドの終了コードと同じのです。
exitcode	リクエストの終了コード、またはシグナル番号です。
execmid	リクエストを実行したマシン ID です。
starttime	予備領域です。
endtime	リクエストの実行終了時刻です。
reserve	予備領域です。

313

実行ホスト名

リクエストが実行されたホスト名です。内容はすでに終了したリクエストの場合のみ入ります。

■文字列型

314

リクエストエラーメッセージ

リクエストがJobCenter 内部のエラーで終了したときのメッセージの文字列です。内容はすでに終了したリクエストの場合のみ入ります。

■文字列型

320

リクエストの属性 (詳細レベル 3)

■long 配列型

■参照構造体

```
struct NQSreqattr3 {
        long   type;
        uid_t  owner_submit;
        long   flags;
        long   umask;
        time_t createtime;
        long   stdout_mode;
        long   stdout_mid;
        long   stderr_mode;
        long   stderr_mid;
        long   flockflags;
        long   vlevel;
        long   reserve[16];
};
```

type	リクエストのタイプを表す数字が入ります。1はバッチリクエストを表します。	
owner_submit	リクエスト投入マシン上での、リクエストユーザ ID です。不明の場合には -1 が入ります。	
flags	各ビットが 1 のときの意味は以下のとおりです。以下のビット以外のものもセットされることがあります。	
	000010	リクエストの実行が開始したとき、メールを送信します。
	000020	リクエストの実行が終了したとき、メール送信します。
	000040	リクエストが再実行されました。
	000100	再実行可能です。
umask	実行時に設定する umask 値です。	
createtime	リクエスト作成日時です。	
stdout_mode	標準出力の動作モードです。各ビットが 1 のときの意味は以下のとおりです。	
	000001	スプールモードです。
	000010	リクエストが実行されたマシン上に出力ファイルが保持されます。
stdout_mid	stdout 出力を向けるマシン ID です。	

stderr_mode	標準エラー出力の動作モードです。各ビットが 1 のときの意味は以下のとおりです。	
	000001	スプールモードです。
	000002	-eo オプションが指定されました。
	000010	リクエストが実行されたマシン上に出力ファイルが保持されます。
stderr_mid	stderr 出力を向けるマシン ID です。	
flockflags	qsub -v オプションのフラグです。各ビットが 1 のときの意味は以下のとおりです。	
	000010	-v オプションが指定されました。
	000020	-ve オプションが指定されました。
	000040	-vo オプションが指定されました。
vlevel	-v オプションのレベル数値が入ります。	

321

投入マシンでのユーザー名

■文字列型

322

リクエストの資源制限値

リクエストの資源制限値です。資源制限タイプごとの資源制限値ブロックの集合です。

■ブロック集合型

■下位ブロック

キュー資源制限値 (206) と同様です。

323

リクエストの警告制限値

リクエストの資源警告制限値です。資源制限タイプごとの資源制限値ブロックの集合です。

■ブロック集合型

■下位ブロック

キュー資源制限値 (206) と同様です。

324

標準出力ファイル名

■文字列型

325

標準出力転送ホスト名

■文字列型

326

標準エラー出力ファイル名

■文字列型

327

標準エラー出力転送ホスト名

■文字列型

328

メールアドレス

■文字列型

329

使用シェル名

■文字列型

9.5.6. 制限事項

NQSqstat 関数は、 R1.1 マシンでのリクエスト実行結果を得ることができません。

9.6. NQSqsub　バッチリクエストの投入

```
#include <nqsapi.h>
struct NQSblockhead *NQSqsub(char *argument);
```

9.6.1. 機能説明

NQSqsub 関数は、バッチリクエストの投入を行います。

引き数 argument は、 qsub コマンドに指定するのと同様の文字列を指すポインタです。この関数は、結果を動的な領域に格納し、その領域のアドレスを返します。

結果を格納する領域は、ライブラリ中で malloc 関数により確保するので、利用者が後でNQSfree関数を用いて解放する必要があります。

結果領域には以下のブロックが入ります。

1. コマンドステータス (intro(3NQS)参照)
2. メッセージ (intro(3NQS)参照)
3. リクエストID (文字列型)
4. リクエストが投入されたキュー名 (文字列型)

結果領域の格納形式については、 intro(3NQS) で詳しく述べているので、そちらを参照してください。

9.6.2. 戻り値

NQSqsub　関数の戻り値は、結果領域の先頭アドレスです。結果領域の確保に失敗した場合は、NULL を返します。

9.6.3. 使用例

```
#include <nqsapi.h>
main()
{
        struct NQSblockhead *adr,*p;
        struct NQSapistat *sb;
        char *rid;
        char *queue;

        adr = NQSqsub( "-a tomorrow -q queue0 scriptfile1" );

        if( adr == NULL ){
                printf("ERROR");
                exit(1);
        }

        sb = (struct NQSapistat *)NQSblockdata( adr );

        if( sb->status > 0 ){
                printf("ERROR");
                exit(1);
        }

        p = adr;
        p = NQSblocknext(p);
```

```
        p = NQSblocknext(p);
        rid = (char *)NQSblockdata(p);
        p = NQSblocknext(p);
        queue = (char *)NQSblockdata(p);

        printf( "request_id = %s\n", rid );
        printf( "queue_name = %s\n", queue );

        NQSfree( adr );
}
```

スクリプトファイル名は必ず指定してください。

9.7. NQSqwatch　JobCenter イベント通知

```
#include <nqsapi.h>
struct NQSblockhead *NQSqwatch( char *argument );
```

9.7.1. 機能説明

NQSqwatch は、 JobCenter 内でのリクエストの動作やキューの状態変化を各種上位プログラムにリアルタイムに通知します。 NQSqwatch は、 FIFO とデータファイルを使ってJobCenterデーモンからイベントを受け取ります。利用手順は以下のようになります。

1. NQSqwatch 機能の利用開始を宣言して FIFO パス名を得る (-s オプション)。

2. FIFO に通知して欲しいイベントを登録する (-e オプション)。

3. イベントを待つ (-r オプション)。

4. イベント登録を解除する (-c オプション)。

5. NQSqwatch 機能の利用終了を宣言する (-x オプション)。

関数値として返される結果領域のフォーマットおよび利用方法については intro(3NQS) を参照してください。

argument には、オプション説明の項にある各オプション文字列を 1 つの文字列につなげたものを指定します。

9.7.2. オプション

-s

イベント監視機能の利用を開始します。

結果領域の 3 番目のブロックに文字列が入ります。この文字列はイベント通知用ＦＩＦＯのパス名です。以後の NQSqwatch 関数使用時に指定してください。

-x $fifoname

イベント監視機能の利用を終了します。

$fifoname には qwatch -s で得たパス名を指定します。

-e [-f] $fifoname $eventtype $detail [$target...]

監視するイベントを登録します。

$fifoname には qwatch -s で得たパス名を指定します。

$eventtype　以降にはイベント条件を指定します。これらの記述方法についてはイベント条件の項を参照してください。

同時に -f を指定するとイベント 1 回限りで登録解除されるようになります。

結果領域の 3 番目のブロックに文字列が入ります。この文字列はイベント登録 ID で、イベント登録の解除と受け取ったイベントの判別に使います。

-c $fifoname $entryid

イベント登録を解除します。

$fifoname には qwatch -s で得たパス名を指定します。

$entryid には qwatch -e で得た登録 ID を指定します。

-r [-n] [-z] [-t $timeout] $fifoname

イベント発生を待ち、発生したイベントの内容を結果領域に返します。

$fifoname には qwatch -s で得たパス名を指定します。

同時に -n を指定するとイベントが発生するまで待たずに、ただちにコマンドを終了します。

同時に -z を指定するとそれ以前に発生したイベントを消去します。

同時に -t を指定すると指定時間後にタイムアウトします。 $timeout に秒数を指定します。

結果領域の 3 番目以降の各ブロックにはイベントの内容が入ります。イベントフォーマットの項を参照してください。

-p $fifoname $regproc

$regproc で指定したプロセス ID を持つプロセスをイベントを受信するプロセスとしてFIFO に登録します。登録したプロセスが終了した場合、指定した FIFO に対するイベント登録はすべて破棄され、 FIFO は削除されます。イベント受信プロセス登録は、regproc に -1 を指定することで解除されます。

$fifoname には qwatch -s で得たパス名を指定します。

9.7.3. イベント条件

イベント条件は、大分類、小分類、照合条件によって記述します。

■大分類 ($eventtype)

数値を指定します。

■小分類 ($detail)

記号を指定します。記号の意味は大分類によって変わります。記号を複数続けて書くことにより、複数のイベントを同時に待つことができます。

■照合条件 ($target)

オプションと引き数の形式で指定します。指定できる条件は大分類によって変わります。

9.7.3.1. リクエスト系イベントの場合

リクエストの状態変化に伴うイベントです。

大分類には数値 1 を指定します。

小分類の記号は以下のものがあります。

A: リクエストの生成
B: リクエストの終了
C: リクエストの転入
D: リクエストの転出
E: 実行中バッチリクエストの一時中断
F: リクエストのローカル転入

G:　リクエストの状態変化 (HOLD, WAIT, RUN, SUS)

リモートマシンに転送されたリクエストの場合、リクエスト投入元で監視が可能なものは

B, C, G のみです。その他はリクエストが存在するマシン上でのみ使えます。

照合条件には以下のものがあります。

-q $queue キュー名

-i $request-id リクエスト ID

-h $host-name ホスト名

9.7.3.2. キュー操作系イベント

キュー操作に伴うイベントです。

大分類には数値 10 を指定します。

小分類の記号は以下のものがあります。

A:　start
B:　stop
C:　enable
D:　disable

照合条件には以下のものがあります。

-q $queue キュー名

-h $host-name ホスト名

9.7.3.3. 指定例

■リクエストの終了とその他の状態変化を待ち合わせる場合

```
1 BG -i 100.host1
```

■キューのリクエスト数の変化

```
1 ABCDEF -q batch1
```

9.7.3.4. イベントフォーマット

結果領域の第3ブロック以降には次のようなデータが入ります。9番目以降はイベントの種類によって変わります。

■共通

第3ブロック	予備領域
第4ブロック	イベント登録ID(文字列)
第5ブロック	イベント発生時刻(time_t 型 x 1)
第6ブロック	イベント大分類(文字列・イベント条件の項を参照)
第7ブロック	イベント小分類(文字列・イベント条件の項を参照)
第8ブロック	マシン名(文字列)

■リクエスト系イベント

第9ブロック	リクエストID(文字列)
第10ブロック	キュー名(文字列)
第11ブロック	リクエストの状態(整数 x 1: NQSqstat(3NQS) コード311 stateを参照)
第12ブロック	リクエストの終了状態(整数配列: NQSqstat(3NQS) コード312を参照)
第13ブロック	キューのリクエスト数(整数配列: NQSqstat(3NQS) コード107を参照)
第14ブロック	マシンのリクエスト数(整数配列: NQSqstat(3NQS) コード107を参照)

■キュー操作系イベント

第9ブロック	キュー名(文字列)
第10ブロック	キューの状態(整数・NQSqstat(3NQS)を参照)

9.7.3.5. 使用例

■リクエストの終了とその他の状態変化を待ち合わせる場合

```
      .
      .
      .
adr = NQSqwatch("-s");
fifoname = (char *)NQSblockdata(NQSblocknext(NQSblocknext(adr)));
sprintf( buf, "-e %s 1 BG -i %s", fifoname, reqid );
NQSqwatch( buf );
sprintf( buf, "-r %s", fifoname );
NQSqwatch( buf );
      .
      .
      .
```

9.7.3.6. ポーリング

JobCenter内部で発生したイベントはNQSqwatch("-s")で返されるパス名のFIFOを通じて通知されます。これを非同期モードでオープンし、読み込み可能状態をポーリングすることにより、イベント通知を非同期に検出することができます。詳しくはpoll(2),streamio(7)を参照してください。

なお高優先度メッセージは内部で使用するので、ポーリングは通常優先度メッセージに対して行ってください。

9.7.3.7. 注意事項

■タイミングに関する注意

リクエストの終了などの　1回しか起こらないイベントを確実に知るためには、まず終了イベントを登録した後でリクエストが終了していないことを確認し、その後で待ちに入る必要があります。これが逆だと、終了の確認とイベント登録の間でイベントが発生した場合、イベントを取りこぼしてしいます。

■イベントスプーリングファイルの容量に関する注意

イベントスプーリングファイルの容量には限界があるので、大量のイベントを受信しないでおくと保持できなくなったイベントは、古いイベントから順に上書きされて捨てられてしまいます。この場合 qmgr の SEt EVent_spool Size サブコマンドにより、最大スプーリングファイルサイズを拡大してください。

■通知タイムアウトに関する注意

イベントスプーリングファイルに対してイベントの受信プロセスがアクセスしない状態のままで、 JobCenter 環境パラメータの Qwatch event expier time で指定した時間が経過すると、強制的にその FIFO は終了します。したがって、最低でもこのパラメータで指定された時間以内に1度はイベントの待ち合わせ、“-r”の実行を行う必要があります。

ただしNQSqwatch　により受信プロセス登録機能を使用している間は、イベントの通知タイムアウトは無効になります。この後プロセス登録を解除した場合、通知タイムアウトは再び有効となります。

9.7.3.8. 制限事項

NQSqwatch 関数は、 R1.1 マシンでのイベントを監視することができません。

発行年月 December 2012