JobCenter

R12.8

〈NonStop Server 機能ガイド〉

- Windows XP, Windows 2003, Windows 2008 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、X/Open カンパニーリミテッド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、®、TM、©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 のインストールやバージョンアップ方法などについて説明することを目的としています。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。

- **読み方**

  NonStop Server 上に JobCenter SV をインストール、またはバージョンアップされる場合
  → NonStop 版 JobCenter SV インストールガイドを参照してください。

  JobCenter MG と NonStop Server 上の JobCenter SV の連携環境を構築したい場合
  → NonStop Server 環境構築ガイドを目次に従いお読みください。

  NonStop Server 上での JobCenter SV の機能を理解したい場合
  → 本書をお読みください。

  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。

  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。

- **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。
  注　：本文中につけた注の説明
  備考　：本文中の補足説明

• 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。

URL：http://h50146.www5.hp.com/doc/manual/openview/jc.html

資料名	概　要
JobCenter インストールガイド	NonStop Server 以外の OS に JobCenter を新規にインストール、またはバージョンアップする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 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 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 インポート・エクスポート機能利用の手引き	ユーザ環境のバックアップや環境の移行の際に必要な、JobCenter 上のジョブネットワーク定義、スケジュール定義およびカレンダ定義のインポート･エクスポート機能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 テンプレートガイド	JobCenter に標準添付されている各種テンプレートの利用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。
NonStop 版 JobCenter SV インストールガイド	NonStop 版 JobCenter SV を新規にインストール、またはバージョンアップする方法について説明しています。
JobCenter NonStop Server 機能ガイド	本書
JobCenter NonStop Server 環境構築ガイド	NonStop Server 版 JobCenter を利用するためのシステムの環境構築方法について説明しています。

• 改版履歴

版　数	変更日付	項　目	形　式	変更内容
1	2009/05/28	新規作成	ー	初版。

目　　次

1. 機能概要

NonStop Server のアーキテクチャを有効に利用するため、次のような機能が用意されています。

- JobCenter のプロセスを監視する機能
- ジョブのプロセスを監視する機能
- JobCenter のメッセージを EMS ログに書き出す機能
- ジョブ・プロセスの起動で CPU を分散する機能
- Guardian 環境のジョブ・プロセスを起動する機能

JobCenter の監視

JobCenter のプロセスを監視し、障害発生時に全てのジョブを停止し、JobCenter の環境を再起動する機能です。この役割はマスタ JCMON が担います。マスタ JCMON は Guardian 環境で動作し、プロセスペアの構成になっています。マスタ JCMON は jcmoncontrol コマンドから起動され、稼働中は常に JobCenter を監視し続けます。

ジョブの監視

JobCenter から実行されたジョブを監視し、障害発生時にそのジョブで実行されているプロセスを停止します。この役割はスレーブ JCMON が担います。スレーブ JCMON は、マスタ JCMON と同じプログラムで、同じように Guardian 環境で動作し、プロセスペアの構成になっています。スレーブ JCMON は、ジョブごとに jcmonsh または jcmontacl から起動され、ジョブの完了と共にスレーブ JCMON は停止します。

EMS ログ出力

JobCenter はそれ自身のログファイルを持ちますが、EMS ログにも同時に出力することができます。この機能は logdaemon に組み込まれていて、EMS ログに対応することで各種運用管理ツールとの連携が可能になります。また EMS ログ出力はカスタマイズ可能で、代替 EMS コレクタの指定やイベント番号などを設定することができます。

プロセス起動の分散

JobCenter は、ラウンドロビンに CPU を割り当て、ジョブの実行を行うことができます。この機能は nqsdaemon に組み込まれ、システムに存在する CPU を有効的に利用することができます。また CPU の使用方法はカスタマイズが可能で､カスタマイズを行うことで使用する CPU を制限することもできます。

Guardian プロセス対応

ジョブの実行用シェルに jcmontacl を使用することで、Guardian 環境のジョブを実行することができます。jcmontacl はスレーブ JCMON を介して TACL を起動しますので、単位ジョブで登録するスクリプトには TACL の OBEY ファイルやマクロを記述することができます。

1.1 NonStop Server 上の JobCenter の構成

NonStop Server 上の JobCenter によるプロセス構成は以下のイメージ図のようになっています。

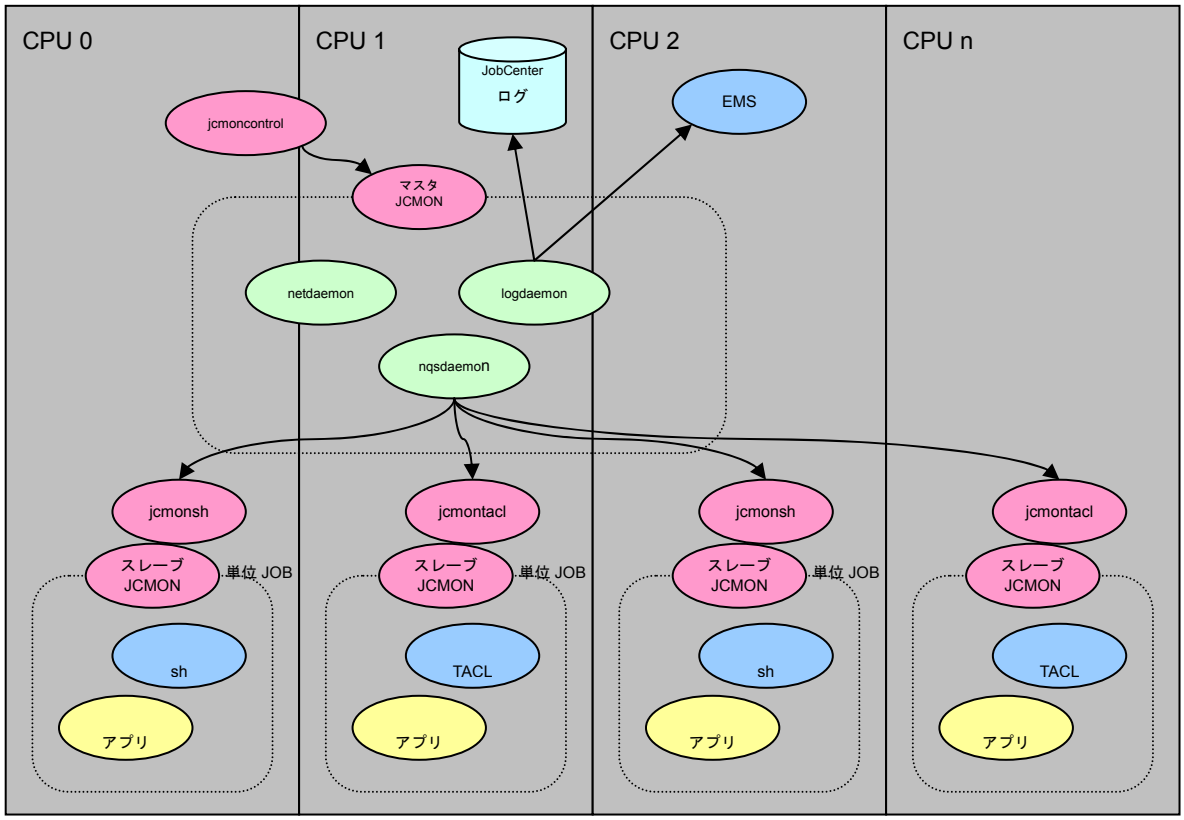

図 1-1　JobCenter の構成イメージ

Jcmoncontrol

マスタ JCMON に対するコマンドインターフェース

- マスタ JCMON の起動や JobCenter のプロセス環境の設定などを行う
- nqsstart や nqsstop から間接的に使用され、直接使用することはない

マスタ JCMON

JobCenter のデーモンを監視するモニタプロセス

- JobCenter のデーモンの起動を行う
- JobCenter のデーモンの監視を行う

スレーブ JCMON

単位ジョブ内で実行中のプロセスの監視を行うモニタプロセス

- sh または TACL を起動する
- sh または TACL から実行されたプロセスの追跡および監視を行う

Jcmonsh

OSS 環境用の JobCenter とモニタプロセスの仲介プロセス

- スレーブ JCMON を起動する
- スレーブ JCMON に sh の起動を要求する
- ジョブの完了を待つ

Jcmontacl

Guardian 環境用の JobCenter とモニタプロセスの仲介プロセス

- スレーブ JCMON を起動する
- スレーブ JCMON に TACL の起動を要求する
- ジョブの完了を待つ

2. JobCenter とジョブの監視

2.1 コンセプト

NonStop 版 JobCenter のジョブの監視および対障害性のコンセプトについて説明します。

2.1.1 JobCenter に障害が発生したとき

JobCenter に障害が発生した場合、障害をマスタ JCMON が検知します。マスタ JCMON は JobCenter の障害を検知すると、JobCenter のデーモンの生き残りのプロセスを全て停止し、全てのスレーブ JCMON にジョブの強制停止を指示します。ジョブの強制停止が完了すると、マスタ JCMON は JobCenter の再起動を行います。

（1） 障害要因

- JobCenter が動作する CPU に障害が発生した
- JobCenter のデーモンが異常終了した
- JobCenter のデーモンがオペレーションにより停止させられた
- JobCenter の親デーモンが子デーモンが存在するのに停止した
- JobCenter のデーモンがシャットダウン待ちの状態ではないのに全て停止した

（2） 結果

- JobCenter のデーモンの生き残りを全て停止する
- 全てのスレーブ JCMON にジョブの強制停止を指示する
- JobCenter を再起動する

備考

上記、JobCenter のデーモンとは、nqsdaemon、logdaemon、netdaemon(親)を意味します。

2.1.2 ジョブに障害が発生したとき

ジョブに障害が発生した場合、障害をそのジョブを担当するスレーブ JCMON が検知します。障害を検知したスレーブ JCMON は、ジョブに属する生き残りのプロセスを全て強制的に停止し、スレーブ JCMON 自身も停止します。

（1） 障害要因

- ジョブのプロセスがエラー完了した
- ジョブが動作する CPU に障害が発生した
- ジョブのプロセスが異常終了した
- ジョブのプロセスがオペレーションにより停止させられた
- ジョブのシェルが子プロセスが存在するのに停止した

（2） 結果

- ジョブのプロセスを全て強制的に停止する
- JobCenter にエラー応答する

2.1.3 アプリのあるリモートノードで障害が発生したとき

（1） リモートノードでのＣＰＵ障害

下記の図のようにノードＡのジョブからノードＢのアプリケーションを起動している状況で、アプリケーションが動作するＢのＣＰＵで障害が発生した場合、障害をそのジョブを担当するスレーブ JCMON が検知します。障害を検知したスレーブ JCMON は、ジョブに属する生き残りのプロセスを全て強制的に停止し、スレーブ JCMON 自身も停止します。

（2） リモートノード障害（含むネットワーク障害）

ノードＡのジョブからノードＢのアプリケーションを起動している状況で、Ｂのノードダウンもしくは完全なネットワークダウンの障害が発生した場合､障害を検知したスレーブ JCMON は､ノードＡ上のジョブに属する生き残りのプロセスを全て強制的に停止し、スレーブ JCMON 自身も停止します。

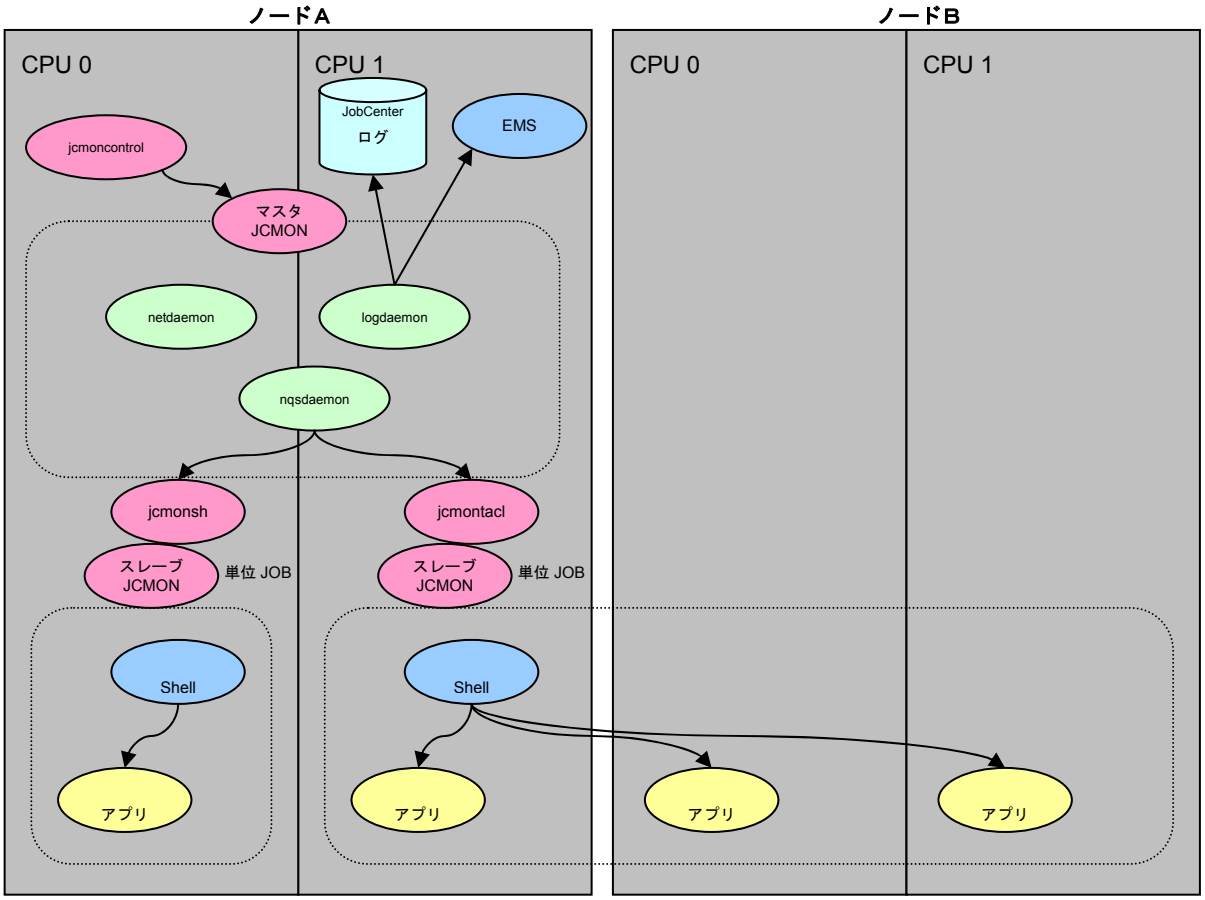

図 2-1 リモートノード実行時のプロセス構成のイメージ

リモートノードのアプリケーションを起動する運用管理での注意事項

リモートノードダウンでジョブが異常終了した場合、運用管理者は原則としてジョブの再起動を行います。ただし、複数ノードにまたがってジョブを実行している場合、ノード障害（含むネットワーク障害）等の発生タイミングによって、リモートノードのアプリケーション自体は正常に終了しているにも関わらず、ジョブ全体として異常終了の表示となるケースがあります。運用管理者は、このようなケースも想定し、無条件に再起動するのではなく、別途、ジョブの実行結果を確認の上、再起動する（場合によっては何もしない）ようにしてください。

2.2 JobCenter の監視

JobCenter の監視はマスタ JCMON が行います。マスタ JCMON は Guardian 環境で動作し、プロセスペアの構成になっています。JobCenter をマスタ JCMON で監視するには、JobCenter の起動をマスタ JCMON から行う必要があります。マスタ JCMON から JobCenter の起動および監視を行うためには、環境変数の定義と設定ファイルの定義が必要になります。

2.2.1 環境変数

JCMON に関連する環境変数には以下に説明するものがあります。

（1） JCMON 環境変数（必須）

JCMON のプログラムファイル名を設定します。この環境変数は必須で、この環境変数を使用して jcmoncontrol がマスタ JCMON の起動を行います。また jcmonsh と jcmontacl もこの環境変数を使用して、スレーブ JCMON の起動を行います。

例 JCMON=¥$DSMSCM.JOBCNTR.JCMON
（”$” はシェルの変数と判断されてしまうので ”¥” でエスケープする）

（2） JCMON_NAME 環境変数（必須）

マスタ JCMON のプロセス名を設定します。この環境変数は必須で、jcmoncontrol がマスタ JCMON を起動するときに、ここで設定したプロセス名で起動します。またスレーブ JCMON がマスタ JCMON と通信する際にも、この環境変数を使用します。

例 JCMON_NAME=¥$JCMON
（”$” はシェルの変数と判断されてしまうので ”¥” でエスケープする）

（3） JCMON_CONFIG 環境変数

JCMON の設定ファイルの名前を設定します。jcmoncontrol が JCMON の設定ファイルを使用しますが、その際、この環境変数に設定されている設定ファイルを読み込みます。この環境変数が設定されていないと、カレントディレクトリの jcmon.conf を使用するようになります。

例 JCMON_CONFIG=/usr/lib/nqs/jcmon/jcmon.conf

（4） JCMON_EMS 環境変数

JCMON が使用する EMS コレクタのプロセス名を設定します。この環境変数が設定されていると、設定された EMS のコレクタにログを出力します。この環境変数が設定されていない場合は、デフォルトの EMS コレクタ（$0）を使用します。

例 JCMON_EMS=¥$JEMS
（”$” はシェルの変数と判断されてしまうので ”¥” でエスケープする）

2.2.2 設定ファイル

マスタ JCMON の動作環境と JobCenter のデーモン・プロセスの環境を設定するためのファイルです。設定ファイルには複数のセクションが存在し、マスタ JCMON の設定を行うためのモニタ・セクションと JobCenter のデーモンの設定を行うためデーモン・セクションから構成されます。モニタ・セクションは１つだけ記述することができ、デーモン・セクションは複数の記述が可能です。

（1） 構文

①. セクション名

１カラム目からセクションの名前を記述する

②. パラメータ

１カラム目をスペースもしくはタブで記述する

③. コメント

１カラム目に#（シャープ）を記述する

設定ファイルのイメージ

```
#   comment

#   comment
Section1
                Parameter1 = Value1
                Parameter2 = Value2
                Parameter3 = Value3

#   comment
Section2
                Parameter1 = Value1
                Parameter2 = Value2
                Parameter3 = Value3
```

（2） モニタ・セクション（*MONITOR*）

セクション名　***MONITOR***

パラメータ　**設定する値**

term　**マスタ JCMON のホーム端末**

ホーム端末には恒久的に存在する端末を設定する必要があり、通常は VHS の端末名を設定します。マスタ JCMON は、ここで設定された端末で起動し、マスタ JCMON から起動された JobCenter のデーモンは、マスタ JCMON のホーム端末を引き継ぐため、同様にここで設定された端末がホーム端末となります。

例　term=$VHS

注 1

VHS をホーム端末に設定した場合、VHS の扱える端末の制限値に注意すること。

注 2

ジョブの実行時に OSS の run コマンドや TACL の RUN コマンドのオプションでホーム端末を設定してプログラムを実行した場合は、そのコマンドで指定した端末がホーム端末となる。

cpus　**マスタ JCMON が使用する CPU**

マスタ JCMON のプライマリ・プロセスの CPU とバックアップ・プロセスの CPU を“:” (Colon)で区切って記述します。

例　cpus=0:1

priority　**マスタ JCMON のプライオリティ**

マスタ JCMON のプライオリティを 1～199 の範囲で設定します。

例　priority=160

(3) デーモンセクション

セクション名　**1～64 文字の任意の文字列**

パラメータ　**設定する値**

path　**デーモンのプログラムパス**

起動するデーモンのプログラム名をフルパスで記述します。

例　path=/usr/lib/nqs/nqsdaemon

arg　**デーモンに対する引数**

起動時にデーモンに渡す引数を記述します。

例　arg=-d

cpus　**デーモンが使用する CPU**

デーモンが使用する CPU を記述する。CPU の記述方法には３種類あります。

- プライマリ CPU とバックアップ CPU

  プライマリの CPU とバックアップの CPU を ":" (Colon) で区切って記述します。使用中の CPU で障害が発生すると、バックアップの CPU でデーモンを再起動するようになります。

  例　cpus=2:3

- 使用可能な複数 CPU

  使用可能な CPU を "," (Comma) で区切って記述します。CPU は記述された順番に使用され、使用中の CPU で障害が発生すると次の CPU を使用して再起動が行われます。

  例　cpus=1,3,5

- １つの CPU

  使用する CPU を１つだけ記述します。使用していた CPU に障害が発生するとデーモンは再起動されず、CPU が復旧したときに再起動されるようになります。

  例 cpus=1

restart　　**デーモンの再起動する制限値**

デーモン・プロセスに障害が発生するとマスタ JCMON が再起動を行いますが、その再起動回数の制限値を記述します。デーモン・プロセスに障害が発生すると、マスタ JCMON は制限値に達するまで再起動を試み、制限値に達すると再起動を行わなくなります。制限値に 0 を設定すると再起動は行われず、-1 が設定されると永久に再起動を試みるようになります。

例　restart=5

process　　**デーモンのプロセス名**

起動するデーモンのプロセス名を記述します。

例　process=$JCSV

priority　　**デーモンのプライオリティ**

デーモン・プロセスのプライオリティを 1～199 の範囲で設定します。デーモンのプロセスはマスタ JCMON に監視されるため、マスタ JCMON よりもプライオリティを下げて設定するようにします。

例　priority=150

group　　**デーモンのグループ**

デーモンのプロセスが所属するグループの名前を記述します。グループ名は任意の名前を使用でき、同一のグループ名のデーモンは同一のグループと見なされるようになります。このグループを設定すると、同一のグループに属するデーモンは、障害時にグループ単位で再起動が行われるようになります。

例　group=JobCenter

stdin　　**デーモンの標準入力**

デーモンに標準入力を割り当てる際に設定します。省略時は/dev/null が使用されます。

stdout　　**デーモンの標準出力**

デーモンに標準出力を割り当てる際に設定します。省略時は/dev/null が使用されます。

stderr　　**デーモンの標準エラー**

デーモンに標準エラーを割り当てる際に設定します。省略時は/dev/null が使用されます。

（4） 設定ファイルの例

```
*MONITOR*
                        term=$VHS
                        cpus=0:1
        priority=160
nqsdaemon
                        path =/usr/lib/nqs/nqsdaemon
                        arg=-d
                        cpus=1:0
                        restart=10
                        process=/G/jcsv
                        priority=150
                        group = JobCenter
                        stdin = /dev/null
                        stdout= /dev/null
                        stderr =/dev/null
```

2.2.3 jcmoncontrol コマンド

マスタ JCMON の起動や停止は、マスタ JCMON の制御コマンドである jcmoncontrol コマンドを使用して行います。jcmoncontrol コマンドは、マスタ JCMON の起動やデーモン・プロセスの起動を行う際に、環境変数や設定ファイルを参照し、その内容に従って起動を行います。

（1） startup サブコマンド

マスタ JCMON を起動します。

```
jcmoncontrol startup
```

メモ マスタ JCMON のプログラム名は環境変数 JCMON から取得し、マスタ JCMON のプロセス名は環境変数 JCMON_NAME で決定されます。使用する CPU は設定ファイルのモニタセクションの cpus できまります。起動されたマスタ JCMON のプライオリティは、設定ファイルのモニタセクションの priority の値になります。

（2） start サブコマンド

JobCenter のデーモンを起動します。

```
jcmoncontrol start daemonname
```

daemonname
設定ファイル中のデーモンのセクションで定義したセクション名

メモ 起動されるデーモンは設定ファイルのデーモン・セクションの名前から決定され、起動するプログラムやプロセス名、プライオリティや CPU などのプロセスの属性は、設定ファイルの該当デーモン・セクションのパラメータに従います。

（3） shutdown サブコマンド

マスタ JCMON をシャットダウン待ち状態にします。

```
jcmoncontrol shutdown
```

メモ マスタ JCMON はシャットダウン待ち状態になるとデーモン･プロセスの終了を待ちますが､デーモン・プロセスの終了の仕方には注意しません。通常状態では再起動が掛かるような終了の仕方でも、シャットダウン待ちの状態に入ると再起動を行うことはありません。シャットダウン待ち状態に入ると、デーモンの終了を待ち、デーモン・プロセスが全て終了するとマスタ JCMON も停止します。

2.2.4 JobCenter の起動と停止

JobCenter をマスタ JCMON で監視するためには、マスタ JCMON から JobCenter のデーモンを起動する必要があります。また JobCenter を停止する際も、jcmoncontrol コマンドを使用したシャットダウンの手順に従う必要があります。

（1） JobCenter の起動

nqsstart コマンドを用いて行います。このコマンドは、内部で以下の動作を行い、正常に JobCenter のデーモンの起動を行います。

- jcmoncontrol コマンドを使用してマスタ JCMON を起動します。
- jcmoncontrol コマンドを使用してマスタ JCMON に JobCenter のデーモンの起動を指示します。

```
/usr/lib/nqs/nqsstart
```

（2） JobCenter の停止

nqsstop コマンドを用いて行います。このコマンドは、内部で以下の処理を行い、正常に JobCenter のデーモンの停止を行います。

- jcmoncontrol コマンドを使用してマスタ JCMON をシャットダウン待ち状態にします。
- JobCenter の通常の停止コマンドを実行します。

例

```
/usr/lib/nqs/nqsstop
```

2.3 ジョブの監視

ジョブの監視は、スレーブ JCMON が行います。スレーブ JCMON にジョブを監視させるためには、シェルとして OSS 環境のジョブには jcmonsh、Guardian 環境のジョブには jcmontacl を使用してジョブを実行します。jcmonsh や jcmontacl はシェルの役割をしますので、JobCenter で使用するジョブのシェルを jcmonsh や jcmontacl に設定することでスレーブ JCMON を使用したジョブの監視を行うことができるようになります。

2.3.1 シェルの種類とジョブの環境

OSS 標準のシェルを使用すると、スレーブ JCMON を使用せずにアプリケーションを実行します。そのため一般的な JobCenter の利用方法と同じになりますが、ジョブの監視は行うことができません。ジョブの監視を行うには、jcmonsh または jcmontacl を使用することになります。

	OSS の標準シェル(/bin/sh)	**jcmonsh**	**jcmontacl**
動作環境	OSS 環境	OSS 環境	Guardian 環境
スクリプト	Bourne Shell のスクリプト	Bourne Shell のスクリプト	TACL の OBEY やマクロ
ジョブの監視	×	○	○
起動シーケンス	1. JobCenter 2. /bin/sh 3. ユーザ・アプリケーション	1. JobCenter 2. jcmonsh 3. スレーブ JCMON 4. /bin/sh 5. ユーザ・アプリケーション	1. JobCenter 2. jcmontacl 3. スレーブ JCMON 4. TACL 5. ユーザ･アプリケーション

2.3.2 OSS 環境のジョブ

OSS の標準シェルを使用するとジョブの監視を行うことができませんが、使用するシェルに jcmonsh を設定するだけで、OSS の標準シェルを使用したの同様のスクリプトと使用方法でジョブの監視を行うことができるようになります。

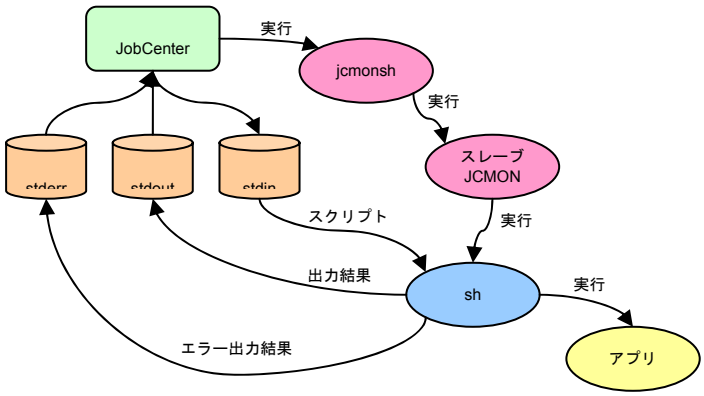

図 2-2 OSS 環境のジョブ実行イメージ

2.3.3 Guardian 環境のジョブ

Guardian 環境のアプリケーションを OSS の標準シェルから実行するためには、gtacl コマンドを使用する Bourne Shell のスクリプトを記述し実行する必要があります。しかし jcmontacl を使用することで、TACL の OBEY をそのまま CL/Win より記述することができ、Guardian 環境のアプリケーションもそのまま実行することができるようになります。

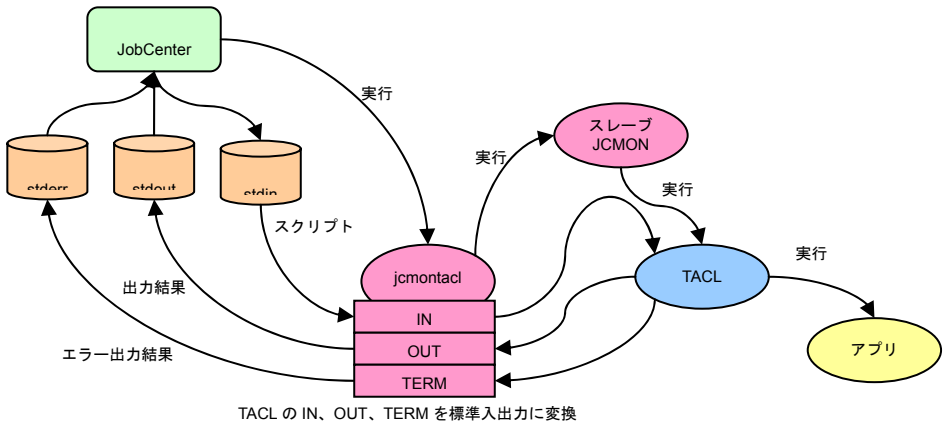

図 2-3 Guardian 環境のジョブ実行イメージ

（1） TACL に渡される文字

jcmontacl によって行中に含まれる書式送り（'¥f'）、改行（'¥n'）、復帰（'¥r'）、水平タブ（'¥t'）および垂直タブ（'¥v'）は全て空白に変換されます。また、行末に存在する 1 文字以上の空白文字は不要なため取り除かれます。また行末に記述された＆は、TACL と同様に次の行への記述の継続を意味します。

注意事項

ベル文字のような特殊文字が記述可能なエディタから JobCenter CL/Win のスクリプトにカット＆ペーストすると、TACL が処理できない特殊文字が混入する可能性があります。したがって、スクリプトへ入力するテキストの作成には、特殊文字が記述可能なエディタは使用しないでください。

（2） 環境変数と PARAM のマッピング

jcmontacl は、自身の持つ環境変数を PARAM に変換します。その際 PARAM 名で許されない “_” (Underscore)を “^” (Circumflex)に変換します。また、PARAM 全体での容量制限が 1024 バイトになりますので、1024 バイトを超える部分の環境変数は除かれます。

変換例

環境変数	PARAM
QSUB_REQID	QSUB^REQID
NSJNW_BASEDATE	NSJNW^BASEDATE
NSJNW_JNWNAME	NSJNW^JNWNAME
NSJNW_UJNAME	NSJNW^UJNAME

（3） PARAM への変換を行わない環境変数の設定

PARAM 全体で容量制限があるため、jcmontacl から PARAM への変換を行わない環境変数を設定することができます。

注意事項

設定ファイルの作成・編集作業は JobCenter が停止している状態で行ってください。

設定方法

- /usr/lib/nqs/rc/jcmontacl_unset.conf という設定ファイルを作成します。
- 設定ファイルに read 権限を与えてください。

例

```
chmod 444     /usr/lib/nqs/rc/jcmontacl_unset.conf
chown 65535 /usr/lib/nqs/rc/jcmontacl_unset.conf
chgrp SUPER /usr/lib/nqs/rc/jcmontacl_unset.conf
```

- 設定ファイルには変換を行わない環境変数を、一行につき一つ記述します。
- 改行コードは、LF、CR+LF どちらでも構いません。
- 行頭や行末にタブやスペースが入っても構いません。
- 行頭に#が入っている場合はコメント行として扱われます。

下記の表にあげる指定可能な環境変数のみ変換を行わない対象として設定できます。

設定される条件	設定ファイルに指定可能な環境変数のリスト			
UNIX 版 JobCenter MG からジョブを投入した場合	QSUB_HOME		QSUB_SHELL	
	QSUB_LOGNAME		QSUB_TZ	
	QSUB_MAIL		QSUB_USER	
	QSUB_PATH			
Windows 版 JobCenter MG からジョブを投入した場合	QSUB_ComSpec		QSUB_SystemRoot	
	QSUB_HOMEDRIVE		QSUB_temp	
	QSUB_HOMEPATH		QSUB_tmp	
	QSUB_Os2LibPath		QSUB_USERNAME	
	QSUB_Path		QSUB_windir	
	QSUB_SystemDrive			
CL/Win からジョブネットワークを投入した場合	NSJNW_BASEDATE		NSJNW_JNWTIME	
	NSJNW_BASETIME		NSJNW_PARAM	
	NSJNW_DIR		NSJNW_UJNAME	
	NSJNW_HOST		NSJNW_UJERREC	
	NSJNW_JNWNAME		NSJNW_USER	
UNIX 版, Windows 版 共通で設定される環境変数	QSUB_HOST		INPUT_SCRIPT	※
	QSUB_REQID		JCMON	
	QSUB_REQNAME		JCMON_NAME	※
	QSUB_WORKDIR		JCMON_EXEC	
	STANDARD_OUTPUT	※	JCMON_EMS	
	STANDARD_ERROR	※	JCMON_CONFIG	

※印は、NonStop 版 JobCenter が内部制御で用いる環境変数ですが、jcmontacl_unset.conf に記載しても動作に問題はありません。

2.3.4 終了コード

NonStop Kernel では完了コードとして-32768～32767 の範囲の値が使用されますが、JobCenter では 0～255 の範囲の終了コードが有効になります。これは JobCenter の制限ではなく、終了コードを 8 ビットで表現するのが一般的で、これは OSS でも同様です。しかし NonStop Kernel の完了コードには、負の値を持つ物が幾つかあり、ユーザ・アプリケーションにも完了コードとして 0～32767 の範囲で使用が許されています。そのためスレーブ JCMON では、完了コードとリターンコードを次のようにマッピングし、JobCenter に報告します。

完了コード	リターンコード
-32768～-1	255
0～255	0～255
256～32767	255

単位ジョブトラッカウィンドウの詳細情報からの確認

単位ジョブトラッカウィンドウの詳細情報では､[終了理由] の行にてリターンコードの確認ができます。単位ジョブトラッカウィンドウについては「JobCenter 基本操作ガイド」を参照してください。

（1） jcmonsh を使用する場合のリターンコード

/bin/sh の完了コードが JobCenter のリターンコードにマッピングされて表示されます。

（2） jcmontacl を使用する場合のリターンコード

TACL の完了コードが JobCenter のリターンコードにマッピングされて表示されます。TACL では 0 は正常終了を意味し、0 以外は異常終了を意味します。

TACL の完了コードは、「Guardian Procedure Calls Reference Manual」の Completion Codes に準じています。（Completion Code が標準化される以前に作成された TACL のコマンドはこの限りではありません。）

注意事項

jcmonsh や jcmontacl 自身が異常終了した場合も、完了コードは「Guardian Procedure Calls Reference Manual」の Completion Codes に準じます。その際のメッセージは EMS ログより確認してください。

2.3.5 ジョブ監視の注意事項

ジョブの監視を行うスレーブ JCMON は、１つの単位ジョブから派生する全てのアプリケーションの完了を待ち合わせて、その単位ジョブが完了したと判断します。

- 派生する全てのアプリケーションの完了を待ち合わせるため、アプリケーションにループ障害が発生した場合、オペレータが対処を行う必要があります。またアプリケーションの実行時間や完了時刻がわかる場合には事前に超過警告等の設定をしておくことも可能です。

- アプリケーションの完了の待ち合わせは、バックグラウンドに投入されるアプリケーションに関しても同様に行われます。

- 完了を待ち合わせたくないアプリケーションや、仕様上、強制停止(完了コード 6)が正常動作として発生するアプリケーションは、jobid を 0 として投入することができます。この場合、投入されたアプリケーションは JCMON の監視対象外となるので注意が必要です。

JCMON の監視対象外のアプリケーションをバックグラウンドに投入する場合

- jcmontacl

例

```
TACL/IN $DSMSCM.JOBCU01.SAMPLE, OUT $S.#JC, CPU 3, NAME $JOBX, TERM $ZHOME, NOWAIT, JOBID 0/
```

何らかの端末を term として設定することを推奨します。

- jcmonsh

例

```
run -name=/G/jobx -jobid=0 sh /temp/sample &
```

3. EMS ログ出力

JobCenter に関する EMS ログの設定方法について説明します。

3.1 JobCenter のメッセージ

JobCenter には次のようなメッセージの種類があり、それぞれデフォルトで EMS ログに出力されるかされないか、イベント番号がいくつか、通常のメッセージなのかクリティカルなメッセージなのかが設定されています。

種類	EMS に出力する/しない	イベント No	通常/クリティカル
ログメッセージ	する	1	通常
情報メッセージ	する	2	通常
警告メッセージ	する	3	クリティカル
エラーメッセージ	する	4	クリティカル
重大メッセージ	する	5	クリティカル
デバッグメッセージ	しない	6	通常
プロセス死活メッセージ	しない	(注)	通常

(注)
情報の内容に応じたイベント No にて出力されます。

3.2 EMS 出力のカスタマイズ

出力される EMS ログのカスタマイズ方法について説明します。

3.2.1 カスタマイズ用環境変数

EMS 出力のカスタマイズは、/usr/lib/nqs/rc/jcmonenv.sh ファイル内で行います。それぞれ次のような環境変数に値を設定することで、カスタマイズを行うことができます。

種類	環境変数名
ログメッセージ	JOBCENTER_EMS_LOG
情報メッセージ	JOBCENTER_EMS_INFO
警告メッセージ	JOBCENTER_EMS_WARN
エラーメッセージ	JOBCENTER_EMS_ERROR
重大メッセージ	JOBCENTER_EMS_FATAL
デバッグメッセージ	JOBCENTER_EMS_DEBUG
プロセス死活メッセージ	JOBCENTER_EMS_PROCESS

3.2.2 構文

環境変数名={ON|OFF}[:イベント番号[:{C|N}]]

（1） EMS に出力する/しない

ON または OFF で設定します。ON に設定した場合、メッセージは EMS ログにも出力されます。OFF に設定した場合、JobCenter のログファイルにだけメッセージが出力されます。

（2） イベント番号

1 から 32767 の範囲で設定します。

（3） 通常/クリティカル

N または C を設定します。N を設定した場合、メッセージは通常のメッセージとして出力されます。C を設定した場合、メッセージはクリティカルなメッセージとして出力されます。

環境変数 JOBCENTER_EMS_PROCESS については、イベント番号、通常/クリティカルの設定はできません。

3.2.3 設定例

以下にそれぞれのカスタマイズ用の環境変数について設定例を示します。

（1） 例１ JOBCENTER_EMS_DEBUG=ON

デバッグメッセージを EMS ログに出力するように設定します。

（2） 例２ JOBCENTER_EMS_INFO=OFF

情報メッセージを EMS ログに出力しないように設定します。

（3） 例３ JOBCENTER_EMS_WARN=ON:1000

警告メッセージのイベント番号を 1000 に設定します。

（4） 例４ JOBCENTER_EMS_ERROR=ON:2000:N

エラーメッセージのイベント番号を 2000 にして通常のメッセージに設定します。

（5） 例５ JOBCENTER_EMS_LOG=ON:3000:C

ログメッセージのイベント番号を 3000 にしてクリティカルなメッセージに設定します。

（6） 例６ JOBCENTER_EMS_PROCESS=ON

プロセス死活メッセージを EMS ログに出力するように設定します。

3.3 EMS コレクタの指定

EMS コレクタが指定されないと、デフォルトの EMS コレクタ($0)が使用されますが、環境変数 JOBCENTER_EMS_COLLECTOR に EMS コレクタを指定すると、指定した EMS コレクタに EMS イベントが出力されます。

例

JOBCENTER_EMS_COLLECTOR=¥$JEMS

（$JEMS の前の¥’バックスラッシュ’は、シェルの変数に解釈されないようにするためのエスケープ文字です）

3.4 カスタマイズ内容の反映

EMS ログ出力のカスタマイズは環境変数で設定することができますが、環境変数の取得は JobCenter の起動時に行われます。そのため、変更内容の反映には JobCenter の再起動が必要になります。

4. プロセス起動の分散

NonStop 版 JobCenter では、起動されるジョブ・プロセスの CPU を分散させることが可能です。

4.1 使用する CPU の指定

4.1.1 ラウンドロビンによる分散

JobCenterはデフォルトで、全てのCPUを使用するようにラウンドロビンでCPUの割り当て、ジョブを実行します。しかし、使用するCPUを制限することも可能です。使用するCPUを制限するためには、環境変数JOBCENTER_CPUSにジョブで使用するCPUを指定します。CPUが指定されると、その指定したCPUだけを使用してジョブの実行を行うようになります。CPUの指定は ”,”(Comma)で区切り、最大16個までのCPUを設定することができ、記述されている順番にCPUを割り当てます。

例

8 CPU のシステムで CPU 指定無し　　CPU は　0 1 2 3 4 5 6 7 0 1 2　..... と割り当てられる

8 CPU のシステムで JOBCENTER_CPUS=1,3,5,7　　CPU は　1 3 5 7 1 3 5 7 1 3..... と割り当てられる

4.1.2 設定ファイル

JobCenter で使用する CPU の設定は、/usr/lib/nqs/rc/jcmonenv.sh ファイル内で行います。

4.1.3 構文

CPU の指定は、“,”(Comma)で区切り、最大 16 個までの CPU を設定することができ、記述されている順に CPU が割り当てられます。

記述例：JOBCENTER_CPUS=1,3,5

4.1.4 CPU 指定が無い場合

JOBCENTER_CPUS の指定が無いと、JobCenter は全ての CPU を使用してジョブを割り当て実行します。例えば 4 CPU のシステムのとき、最初のジョブは CPU 0 で実行され、次のジョブは CPU 1 で実行されます。このように CPU は順番に使用されジョブに対して割り当てられます。そして 4 番目のジョブが CPU 3 で実行されると、次の 5 番目の JOB は CPU 0 に戻り、割り当てが繰り返されます。

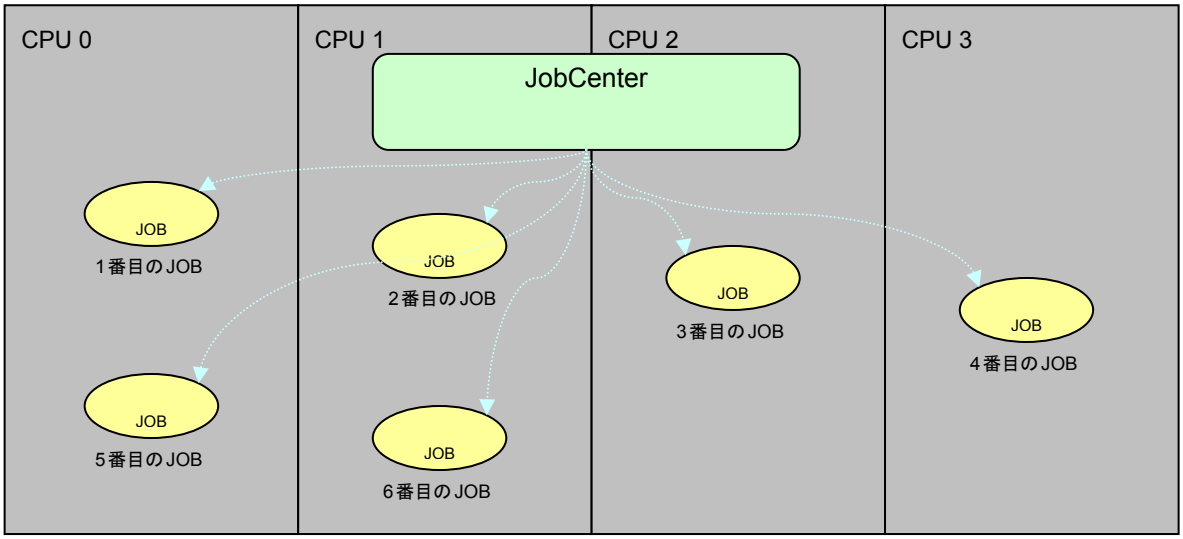

図 4-1 CPU 指定がない場合のジョブの割り当てイメージ

4.1.5 使用する CPU が指定されていた場合

使用する CPU が指定されていた場合、その指定の通りに CPU が割り当てられジョブが実行されます。例えば、4 CPU のシステムで JOBCENTER_CPUS=3,1,2 という指定が行われていた場合、最初のジョブは CPU 3 で実行され、次のジョブは CPU 1 で実行され、そしてその次のジョブは CPU 2 で実行されます。更に次のジョブは CPU 3 に戻り、割り当てが繰り返されます。

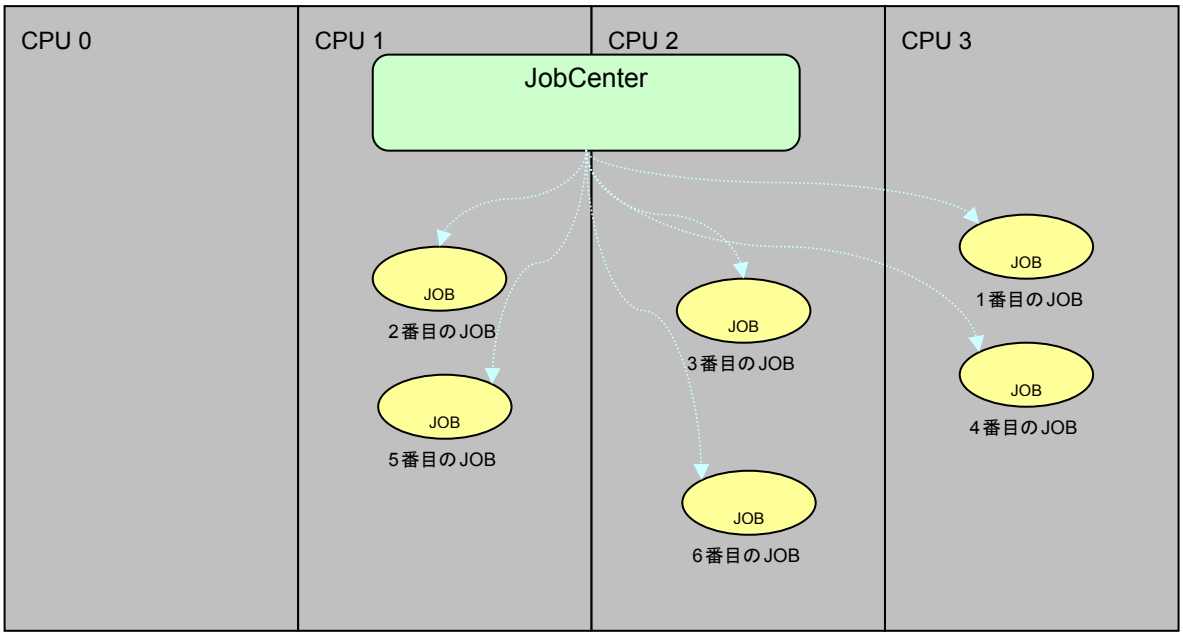

図 4-2 使用する CPU を指定した場合のジョブの割り当てイメージ

4.2 カスタマイズ内容の反映

使用する CPU のカスタマイズは環境変数で設定することができますが、環境変数の取得は JobCenter の起動時に行われます。そのため、変更内容の反映には JobCenter の再起動が必要になります。

5. メッセージ

メッセージには以下の 2 種類のメッセージがあります。本章ではそれぞれのメッセージについて説明しています。

- JobCenter のメッセージ

  JobCenter のデーモンが出力するメッセージです。

- JCMON 関連のメッセージ

  マスタ JCMON、スレーブ JCMON、jcmonsh、jcmontacl が出力するメッセージです。

5.1 メッセージの説明について

メッセージ番号	イベントの番号です。
メッセージ・テキスト	メッセージそのものです。メッセージ・テキストにパラメータが含まれる場合がありますが、パラメータについては斜体で表現し、その説明が欄外に記述されます。
原因	メッセージを生成した状況またはエラー
結果	システムの状況またはエラーの効果
回復処置	レポートされたエラーから回復するために必要な手順
補足情報	メッセージ・テキストに関する補足的な情報がある場合に記述しています。

5.2 JobCenter のメッセージ

サブシステム名：NEC.JOBCNTR.R12

1

NQS(LOG): *message*

message

JobCenter のメッセージ

原　　因： JobCenter がログメッセージを出力した。

結　　果： *message* を確認してください。

回復処置： 通知メッセージです。処置は不要です。

2

NQS(INFO): *message*

message

JobCenter のメッセージ

原　　因： JobCenter が情報メッセージを出力した。

結　　果： *message* を確認してください。

回復処置： 通知メッセージです。処置は不要です。

3

NQS(WARN): *message*

message

JobCenter のメッセージ

原　　因： JobCenter が警告メッセージを出力した。

結　　果： *message* を確認してください。

回復処置： 警告通知メッセージです。基本的には処理は不要です。
補足情報にて情報を開示しているものについては、必要に応じた OS 設定の調査や変更、リソースの追加等の対応を検討してください。また、nmapmgr の設定方法については、JobCenter のマニュアルを参照してください。同じメッセージが大量に出力される場合、情報を採取して製品サポートに問い合わせることが可能です。（保守契約が必要です。）

補足情報： **establishasync: error at nmap_get_nam (mid=*xxx*,errno=*yyy*)**
xxx というマシン ID（NQS ホスト ID）のホスト名が不明です。
ネットワークの名前解決および nmapmgr の設定をご確認ください。
errno.h ファイルにて、errno である yyy を参照することで原因がわかる場合があります。

establishasync: error at localmid (errno=*xxx*)
errno.h ファイルにて、errno である yyy を参照することで原因がわかる場合があります。

establishasync: error at getservbyname (errno=*xxx*)
ライブラリ関数 getservbyname が実行できません。
ネットワークの名前解決および OS の設定を確認してください。
errno.h ファイルにて、errno である xxx を参照することで原因がわかる場合があります。

establishasync: error at gethostbyname (name=*xxx*,errno=*yyy*)
xxx というホスト名に対して、ライブラリ関数 gethostbyname が実行できません。
ネットワークの名前解決および OS の設定を確認してください。
errno.h ファイルにて、errno である yyy を参照することで原因がわかる場合があります。

establishasync: error at socket (errno=*xxx*)
ライブラリ関数 socket が実行できません。
ネットワークおよび OS の設定を確認してください。
errno.h ファイルにて、errno である xxx を参照することで原因がわかる場合があります。

establishasync: error at bind (name=*xxx*,errno=*yyy*)
ライブラリ関数 bind が実行できません。
ネットワークおよび OS の設定を確認してください。
errno.h ファイルにて、errno である yyy を参照することで原因がわかる場合があります。

establishasync: error at bind (no port) (name=*xxx*,errno=*yyy*)
ライブラリ関数 bind が実行できません。
ネットワークおよび OS の設定を確認してください。
errno.h ファイルにて、errno である yyy を参照することで原因がわかる場合があります。

error at fcntl (name=*xxx*,errno=*yyy*)
ライブラリ関数 fcntl が実行できません。
ネットワークおよび OS の設定を確認してください。
errno.h ファイルにて、errno である yyy を参照することで原因がわかる場合があります。

establishasync: error at connect (name=*xxx*,errno=*yyy*)
ライブラリ関数 connect が実行できません。
ネットワークおよび OS の設定を確認してください。
errno.h ファイルにて、errno である yyy を参照することで原因がわかる場合があります。

idc_read:error at read (*n*) (mid=*xxx*,errno=*yyy*)
上記メッセージは JobCenter MG とのコネクションが一時的に切断されたことを
意味します。通信は自動的に再接続いたしますので、処置は不要です。

Netserver: client hostname(xxx) unknown to local host.
xxx というホスト名がローカルホスト上で不明です。
ネットワークの名前解決および nmapmgr の設定をご確認ください。

Netserver: client internet addr(xxx) unknown
xxx という IP アドレスが不明です。
ネットワークの名前解決および nmapmgr の設定をご確認ください。

Netserver: client mid(xxx) unknown to local host.
xxx というマシン ID（NQS ホスト ID）が不明です。
nmapmgr の設定をご確認ください。

Netserver: Client on non-secure port
NQS は、通信元ホストは、セキュアポート(512～1024 番ポート)のうち、未使用のものを使います。しかし、それに反し、1024 番よりも大きいポート番号から送信先に通信を行っているため、警告メッセージが出力されます。ファイアウォールに JobCenter が利用するポートの穴あけが行われているか、また、NAT 機器やロードバランサなどのネットワーク機器がポート変換を行っていないかを確認してください。

Netserver: Invalid batch request packet.
Netserver: Invalid device request packet.
Netserver: non-batch, non-device packet.
Netserver received bad packet.
JobCenter が利用しているポートに対して解釈できないパケットが送られました。処置

は不要です。このメッセージが大量に出力される場合は、送信側のアプリケーションを特定して解決してください。送信側のアプリケーションが JobCenter の場合は、ネットワークの品質の点検を行ってください。

Process *xxx* waiting for system file table overflow
activity to cease in order to return output file.
プロセス ID xxx が必要とするシステムファイルテーブルのリソースが不足しています。
システムファイルテーブルのリソースを確認する必要があります。

Process *xxx* waiting for free i-node to become available
プロセス ID xxx が必要とする i-node のリソースが不足しています。
マシンの負荷を下げるか、i-node のカーネルパラメータの変更を検討してください。

Sorry!! the output file: *xxx* isn't able to be saved.
標準出力やエラー出力をファイルに書き出そうとして失敗したことを表しています。/usr/bin や/usr/spool/nqs/配下のファイルのオーナやモードなどを変更していないか確認してください。

TZ specification is too long
環境変数 TZ の設定が適切かどうかをご確認ください。

Netdaemon: fork failed. errno=yyy
ライブラリ関数 fork が実行できません。
errno.h ファイルにて、errno である yyy を参照することで原因がわかる場合があります。

4

NQS(ERROR): *message*

message

JobCenter のメッセージ

原　　因： JobCenter がエラーメッセージを出力した。

結　　果： *message* を確認してください。

回復処置： 障害の可能性があります。
OS の設定値に関する message の場合は、マシン環境をご確認ください。ハードウェア障害が同時期に発生していないかについても必要に応じてご確認ください。
情報を採取して製品サポートに問い合わせることが可能です。（保守契約が必要です。）

補足情報： **Insufficient processes to fork request server. errno=*yyy*.**
Fork failed(server). errno=*yyy*
Fork failed(netclient). errno=*yyy*
ライブラリ関数 fork が実行できません。
errno.h ファイルにて、errno である yyy を参照することで原因がわかる場合があります。

5

NQS(FATAL): *message*

message

JobCenter のメッセージ

原　　因： JobCenter が重大メッセージを出力した。

結　　果： *message* を確認してください。

回復処置： 障害の可能性があります。
OS の設定値に関する message の場合は、マシン環境をご確認ください。ハードウェア障害が同時期に発生していないかについても必要に応じてご確認ください。
情報を採取して製品サポートに問い合わせることが可能です。（保守契約が必要です。）

補足情報： **Unable to fork() NQS local daemon.(errno=*yyy*)**
Unable to fork() NQS network daemon.(errno=*yyy*)
ライブラリ関数 fork が実行できません。
errno.h ファイルにて、errno である yyy を参照することで原因がわかる場合があります。

6

NQS(DEBUG): *message*

message

JobCenter のメッセージ

原　　因： JobCenter がデバッグメッセージを出力した。

結　　果： *message* を確認してください。

回復処置： Debug 用のメッセージです。処置は不要です。

5.3 JCMON 関連のメッセージ

サブシステム名：NEC.JCMON.V01

1000

JCMON startup.

原　　因： JCMON が起動された。

結　　果： JCMON は初期化を行っている。

回復処置： 通知メッセージです。処置は不要です。

1001

JCMON shutdown complete.

原　　因： JCMON が停止した。

結　　果： JCMON が監視している全てのプロセスが停止された。

回復処置： 通知メッセージです。処置は不要です。

1002

Create backup process on CPU *nn*.

nn

JCMON のバックアップ・プロセスの CPU 番号。

原　　因： JCMON のバックアップ・プロセスが生成された。

結　　果： バックアップ・プロセスが実行され、障害に備えて同期が開始される。

回復処置： 通知メッセージです。処置は不要です。

1003

Shutdown wait.

原　　因：　JCMON が停止待ち状態になった。

結　　果：　JCMON が監視している全てのプロセスが停止するのを待っている。

回復処置：　通知メッセージです。処置は不要です。

1004

Backup process down.

原　　因：　JCMON のバックアップ・プロセスが停止した。

結　　果：　プライマリ・プロセスは処理を継続する。ただし、バックアップ・プロセスが再作成されるまで、CPU 障害に対して耐障害性がない。

回復処置：　通知メッセージです。処置は不要です。

1005

Backup cpu down, CPU *nn*.

nn

JCMON のバックアップ・プロセスが動作していた CPU の番号。

原　　因：　JCMON のバックアップ・プロセスの CPU が障害を起こした。

結　　果：　プライマリ・プロセスは処理を継続する。ただし、CPU 障害に対して耐障害性がない。

回復処置：　ダウンしたプロセッサを再ロードしてください。

1006

Primary process down.

原　　因：　JCMON のプライマリ・プロセスが停止した。

結　　果：　バックアップ・プロセスがテイクオーバし、プライマリ・プロセスとなり処理を続行する。ただし、バックアップ・プロセスが作成されるまで、CPU 障害に対して耐障害性がない。

回復処置：　通知メッセージです。処置は不要です。

1007

Primary cpu down, CPU *nn*.

nn

JCMON のプライマリ・プロセスが動作していた CPU の番号。

原　　因： JCMON のプライマリ・プロセスの CPU が障害を起こした。

結　　果： バックアップ・プロセスがテイクオーバし、プライマリ・プロセスとなり処理を続行する。ただし、バックアップ・プロセスが作成されるまで、CPU 障害に対して耐障害性がない。

回復処置： ダウンしたプロセッサを再ロードしてください。

1008

Failover persistent processes

原　　因： JobCenter を再起動した。

結　　果： 実行中の全てのジョブと JobCenter のデーモン・プロセスが停止し、JobCenter の再起動が行われた。

回復処置： 通知メッセージです。処置は不要です。ただし、JobCenter が正常に起動されているか確認し、可能でならばその原因を修正し、再発を回避して下さい。

1009

Alternate EMS collector error *error#*, Change to default EMS collector($0)

error#

エラーを識別するファイル・システム・エラー番号。

原　　因： 代替 EMS コレクタが使用できない。
（このメッセージはマスタ JCMON が代表して生成します）

結　　果： デフォルトの EMS コレクタ($0)を使用する。

回復処置： 通知メッセージです。処置は不要です。ただし、「Guardian プロシージャ・エラー & メッセージ・マニュアル」のファイル・システム・エラーを参照し、障害の原因を判断して下さい。可能なら再発を回避して下さい。

1010

GMOMNOTIFY *name* (*cpu*,*pin*) jobid=*jobid* Type=*type* Program=*program_name* pid=*pid*

name
プロセス名

cpu
CPU 番号

pin
PIN 番号

jobid
プロセスを JCMON の監視対象とするかどうかのフラグ

type
プロセスタイプ(OSS、または Guardian)

program_name
プログラム・ファイル名

pid
OSS 環境でのプロセス ID(Guardian 環境では 0)

原　　因： プロセスが生成された。

結　　果： プロセスが生成された。

回復処置： 通知メッセージです。処置は不要です。

1011

PROCDEATH *name* cc=*cc* jobid=*jobid*

name
プロセス名

error
プロセスの Completion Code

jobid
プロセスを JCMON の監視対象とするかどうかのフラグ

原　　因： プロセスが終了させられた。

結　　果： プロセスが終了させられた。

回復処置： 通知メッセージです。処置は不要です。ただし、「Guardian プロシージャ・エラー & メッセージ・マニュアル」のファイル・システム・エラーを参照し、障害の原因を判断して下さい。可能なら再発を回避して下さい。

1012

killer (*cpu*,*pin*) cc=*cc* name=*name*

cpu

CPU 番号

pin

PIN 番号

cc

プロセスの Completion Code

name

プロセス名

原　　因： プロセスが異常終了させられた。

結　　果： プロセスが異常終了させられた。

回復処置： Completion Code の値を参考に、「Guardian プロシージャ・エラー & メッセージ・マニュアル」のファイル・システム・エラーを参照し、障害の原因を判断して下さい。可能なら再発を回避して下さい。

1100

Add persistent process, name *name*.

name

JobCenter のデーモン・プロセスの名前（デーモン・セクション名）

原　　因： JobCenter のデーモン・プロセスの定義が追加された。

結　　果： JobCenter のデーモン・プロセスの定義が追加される。

回復処置： 通知メッセージです。処置は不要です。

1101

Delete persistent process, name *name*.

name

JobCenter のデーモン・プロセスの名前（デーモン・セクション名）

原　　因： JobCenter のデーモン・プロセスの定義が削除された。

結　　果： JobCenter のデーモン・プロセスの定義が削除される。

回復処置： 通知メッセージです。処置は不要です。

1102

Start persistent process, name *name*.

name

JobCenter のデーモン・プロセスの名前（デーモン・セクション名）

原　　因： JobCenter のデーモン・プロセスが起動された。

結　　果： JobCenter のデーモン・プロセスが起動される。

回復処置： 通知メッセージです。処置は不要です。

1103

Stop persistent process, name *name*.

name

JobCenter のデーモン・プロセスの名前（デーモン・セクション名）

原　　因： JobCenter のデーモン・プロセスが停止された。

結　　果： JobCenter のデーモン・プロセスが停止される。

回復処置： 通知メッセージです。処置は不要です。

1104

Abort persistent process, name *name*.

name

JobCenter のデーモン・プロセスの名前（デーモン・セクション名）

原　　因： JobCenter のデーモン・プロセスを強制的に停止した。

結　　果： JobCenter のデーモン・プロセスは強制的に停止され、必要があれば再起動される。

回復処置： 通知メッセージです。処置は不要です。

1105

Fail persistent process, name *name*, cc *cc#*.

name

JobCenter のデーモン・プロセスの名前（デーモン・セクション名）

cc#

完了コード

原　　因： JobCenter のデーモン・プロセスの異常終了を検知した。

結　　果： 全てのジョブを強制的に停止し、可能であれば JobCenter を再起動する。

回復処置： 通知メッセージです。処置は不要です。ただし、可能であれば原因を修正し、再発を回避して下さい。

1106

Restart Over persistent process, name *name*.

name

JobCenter のデーモン・プロセスの名前（デーモン・セクション名）

原　　因： JobCenter のデーモン・プロセスの再起動が制限値に達した。

結　　果： 再起動の制限値に達したため、JobCenter の再起動は行われない。

回復処置： 再起動が繰り返された原因を修正し、再発を回避して下さい。

1201

Start process group, group name *name*.

name

デーモン・グループの名前

原　　因： デーモン・グループが起動された。

結　　果： デーモン・グループに所属する全てのプロセスが起動される。

回復処置： 通知メッセージです。処置は不要です。

1202

Stop process group, group name *name*.

name

デーモン・グループの名前

原　　因： デーモン・グループが停止された。

結　　果： デーモン・グループに所属する全てののプロセスが停止される。

回復処置： 通知メッセージです。処置は不要です。

1203

Abort process group, group name *name*.

name

デーモン・グループの名前

原　　因： デーモン・グループが強制的に停止された。

結　　果： デーモン・グループに所属する全てのプロセスが強制停止される。

回復処置： 通知メッセージです。処置は不要です。

1300

Stop all processes.

原　　因：　実行中の全てのジョブ・プロセスを停止した。

結　　果：　実行中のジョブ・プロセスは全て停止する。

回復処置：　通知メッセージです。処置は不要です。

1301

Abort all processes.

原　　因：　実行中の全てのジョブ・プロセスを強制的に停止した。

結　　果：　実行中のジョブ・プロセスは全て強制的に停止する。

回復処置：　通知メッセージです。処置は不要です。

2000

Job submit, id *id#*, name *jnw*:*name*.

id#

JobCenter で採番されたジョブの ID

jnw

ジョブネットワーク名

name

単位ジョブの名前

原　　因： ジョブの実行を開始した。

結　　果： ジョブは開始され、ジョブ・プロセスが起動される。

回復処置： 通知メッセージです。処置は不要です。

2001

Job complete, id *id#*, name *jnw*:*name*.

id#

JobCenter で採番されたジョブの ID

jnw

ジョブネットワーク名

name

単位ジョブの名前

原　　因： ジョブが完了した。

結　　果： ジョブ・プロセスは全て終了し、ジョブが完了する。

回復処置： 通知メッセージです。処置は不要です。

2002

Job aborted, id *id#*, name *jnw*:*name*.

id#

JobCenter で採番されたジョブの ID

jnw

ジョブネットワーク名

name

単位ジョブの名前

原　　因： ジョブが強制的に停止した。

結　　果： ジョブは強制的に停止した。

回復処置： ジョブの状況を確認し、必要であれば原因を取り除いた後、再度ジョブを実行して下さい。

2003

Job failed, id *id#*, name *jnw*:*name*, cc *cc#*[, *text*]

id#

JobCenter で採番されたジョブの ID

jnw

ジョブネットワーク名

name

単位ジョブの名前

cc#

完了コード

text

追加の情報

原　　因： ジョブ・プロセスの異常終了を検知した。

結　　果： ジョブを強制的に終了する。

回復処置： ジョブの状況を確認し、必要であれば原因を取り除いた後、再度ジョブを実行して下さい。

9000

Monitor create error, error *error#*, detail *detail#*.

error#

PROCESS_LAUNCH_プロシージャのエラーコード。

detail#

PROCESS_LAUNCH_プロシージャの詳細エラーコード。

原　　因： JCMON の起動に失敗した。

結　　果： 処理続行不可能なエラー。処理を停止する。

回復処置： 「Guardian プロシージャ・エラー & メッセージ・マニュアル」のファイル・システム・エラーを参照し、障害の原因を判断して下さい。可能なら再発を回避して下さい。

9001

Monitor *operation* error, error *error#* monitor *name*.

operation

行った操作

error#

エラーを識別するファイル・システム・エラー番号。

name

JCMON のプロセス名。

原　　因： JCMON と通信ができない。

結　　果： 処理続行不可能なエラー。処理を停止する。

回復処置： 「Guardian プロシージャ・エラー & メッセージ・マニュアル」のファイル・システム・エラーを参照し、障害の原因を判断して下さい。可能なら再発を回避して下さい。

9002

Not complete initialization, Processing cannot be continued.

原　　因：　初期化が正常に完了していない。

結　　果：　処理続行不可能なエラー。処理を停止する。

回復処置：　通知メッセージです。処置は不要です。このエラーが起こるケースとしては、プライマリ・プロセスがバックアップ・プロセスを生成し、バックアップ・プロセスの初期化が行われるが、それが完了する前にプライマリ・プロセスが停止した場合に発生します。

9003

Process launch error, program *program* error *error#* detail *detail#*.

program

プログラム・ファイル名。

error#

PROCESS_SPAWN_プロシージャのエラーコード

detail#

PROCESS_SPAWN_プロシージャの詳細コード

原　　因：　OSS プロセスの生成に失敗した。

結　　果：　ジョブ・プロセスの実行は行われない。エラー完了する。

回復処置：　「Guardian プロシージャ・エラー & メッセージ・マニュアル」のファイル・システム・エラーを参照し、障害の原因を判断して下さい。可能なら再発を回避して下さい。

9004

Process launch error, program *program* error *error#* detail *detail#*.

program

プログラム・ファイル名。

error#

PROCESS_LAUNCH_プロシージャのエラーコード。

detail#

PROCESS_LAUNCH_プロシージャの詳細コード。

原　　因： Guardian プロセスの生成に失敗した。

結　　果： ジョブ・プロセスの実行は行われない。エラー完了する。

回復処置： 「Guardian プロシージャ・エラー & メッセージ・マニュアル」のファイル・システム・エラーを参照し、障害の原因を判断して下さい。可能なら再発を回避して下さい。

9005

Registration error, error *error#*.

error#

エラーを識別するファイル・システム・エラー番号。

原　　因： マスタの JCMON への登録に失敗しました。

結　　果： ジョブ・プロセスの実行は行われない。エラー完了する。

回復処置： このエラーが発生するケースとして、マスタの JCMON の CPU に障害が発生した場合です。それ以外の場合でこのエラーが発生したときは、「Guardian プロシージャ・エラー & メッセージ・マニュアル」のファイル・システム・エラーを参照し、障害の原因を判断して下さい。可能なら再発を回避して下さい。

9006

Slave monitor communication error, *operation* error *error#*.

operation

行った操作

error#

エラーを識別するファイル・システム・エラー番号。

原　　因： スレーブの JCMON と通信できません。

結　　果： スレーブの JCMON に対して、JobCenter の異常を通知できない。

回復処置： 通知メッセージです。処置は不要です。スレーブの JCMON と通信できないということは、スレーブの JCMON が既に停止している場合です。

9007

Shell process communication error, *operation* error *error#*.

operation

行った操作

error#

エラーを識別するファイル・システム・エラー番号。

原　　因： jcmonsh もしくは jcmontacl と通信できません。

結　　果： ジョブの実行結果を返せない。

回復処置： ジョブの状況を確認して、必要があれば環境を回復し、再度実行して下さい。

9008

Get ancestor process handle, error *error#*.

error#

エラーを識別するファイル・システム・エラー番号。

原　　因： 親(shepherd)プロセスのハンドルが取得できません。

結　　果： ジョブ・プロセスの実行は行われない。エラー完了する。

回復処置： 「Guardian プロシージャ・エラー & メッセージ・マニュアル」のファイル・システム・エラーを参照し、障害の原因を判断して下さい。可能なら再発を回避して下さい。

9009

Parents were lost, error *error#*.

error#

エラーを識別するファイル・システム・エラー番号。

原　　因：　親(shepherd)プロセスが停止している。

結　　果：　ジョブの続行は行われず、エラー完了する。

回復処置：　原因を取り除き、必要があればジョブを再度実行して下さい。

9900

Out of memory, can not allocate *data*.

data

メモリ割り当ての対象。

原　　因：　メモリの割り当てができない。

結　　果：　致命的なエラー。処理を停止する。

回復処置：　JobCenter の全環境を停止し、原因を取り除いた後、再度 JobCenter の環境を起動して下さい。

9901

Out of resource, SIGNALTIMEOUT.

原　　因：　タイマーのリソースが足りない。

結　　果：　致命的なエラー。処理を停止する。

回復処置：　JobCenter の全環境を停止し、原因を取り除いた後、再度 JobCenter の環境を起動して下さい。

9999

Internal Error, *info*

info

内部エラーの情報

原　　因：　内部のエラーが発生した。

結　　果：　致命的なエラー。処理を停止する。

回復処置：　JobCenter の全環境を停止し、サポート担当者に連絡して下さい。

索　　引

発行年月　　May 2009