Sun™ Mainframe Batch Manager ソフトウェア ユーザーズガイド

Release 10.1.0

Sun Microsystems, Inc.
www.sun.com

Part No. 819-2509-10
2005 年 6 月, Revision A

コメントの送付: http://www.sun.com/hwdocs/feedback

原典: *Sun™ Mainframe Batch Manager Software User's Guide*
Part No: 817-7445-10
Revision A

目次

図目次

表目次

はじめに

このマニュアルでは、Sun™ Mainframe Batch Manager (Sun MBM) の次の内容について説明します。

- 環境とサービス
- サブシステム
- ジョブの管理
- コマンド

ご使用のオペレーティング環境とその要件について詳しく理解している必要があります。

マニュアルの構成

第 1 章には、サブシステム、ジョブ、ジョブクラス、ジョブアカウンティング機能など、Sun MBM 環境の概要が記載されています。

第 2 章には、Sun MBM の起動および停止方法、サブシステムの監視方法、ジョブのサブミット方法、有効なジョブの管理と表示方法、ジョブクラスの監視方法、完了したジョブの状態の表示方法、ジョブの統計情報の表示方法が記載されています。また、不正終了やシステムクラッシュからの回復方法についても記載されてます。

第 3 章には、Job Editor の使用方法が記載されています。

第 4 章には、Sun MBM コマンドとそのコマンド例が記載されています。

付録 A には、デーモン、履歴ファイル、コンソール機能、ディレクトリなど、Sun MBM サービスについて記載されています。

UNIX コマンド

このマニュアルには、システムの停止、システムの起動、およびデバイスの構成などに使用する基本的な UNIX® コマンドと操作手順に関する説明は含まれていない可能性があります。これらについては、以下を参照してください。

- 使用しているシステムに付属のソフトウェアマニュアル
- 下記にある Solaris™ オペレーティングシステムのマニュアル

  `http://docs.sun.com`

シェルプロンプトについて

シェル	プロンプト
UNIX の C シェル	*machine_name*%
UNIX の Bourne シェルと Korn シェル	$
スーパーユーザー (シェルの種類を問わない)	#

書体と記号について

書体または記号*	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例。	.login ファイルを編集します。 ls -a を実行します。 % You have mail.
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して表します。	% **su** Password:
AaBbCc123	コマンド行の可変部分。実際の名前や値と置き換えてください。	rm *filename* と入力します。
『 』	参照する書名を示します。	『Solaris ユーザーマニュアル』
「 」	参照する章、節、または、強調する語を示します。	第 6 章「データの管理」を参照。 この操作ができるのは「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	% **grep '^#define ** **XV_VERSION_STRING'**
[ ]	省略可能な項目を示します。	dltjob *jon* [-n *name*]
|	区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。	abtjob *jon* [-s job | cmd]

* 使用しているブラウザにより、これらの設定と異なって表示される場合があります。

このマニュアルでは、次の書式を使用してコマンドを表記します。

```
$ command required-argument [optional-argument]
```

コマンドに省略可能な引数が記述されていない場合は、そのコマンドを入力して Return キーを押します。

ファイル識別子

ファイル識別子は、次の 2 つの部分から構成されます。

- 1 つ以上のディレクトリを指定するディレクトリ、または環境変数
- ファイル識別子の最後の構成要素であるファイル名

ファイル識別子	説明
ディレクトリ	Sun MBM で使用される絶対ディレクトリ名は、60 文字以内でなければなりません。パス名の任意の部分に代えて、先頭にドル符号 ($) を付けた環境変数を使用できます。たとえば、次の 2 行はどちらも有効であり、同じディレクトリを指定します。 • `/local/mbm/pack/bin` • `$PACK/bin` 2 行目の `PACK` 環境変数は `/local/mbm/pack` に設定されています。`$` 指示子を使用することにより、`PACK` 環境変数がそれの完全な意味に展開されます。
環境変数	ディレクトリ名やファイル名を含む名前であり、通常 1 ～ 14 個の大文字です。
ファイル名	ファイル名は、ご使用のプラットフォームの制限事項に従う必要があります。

関連マニュアル

製品	タイトル	Part No.
Sun Mainframe Batch Manager ソフトウェア	『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』	819-2505-10
	『Sun Mainframe Batch Manager ソフトウェア インストールガイド』	819-2506-10
	『Sun Mainframe Batch Manager ソフトウェア メッセージガイド』	819-2507-10
	『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』	819-2508-10
	『Sun Mainframe Batch Manager ソフトウェア リファレンスマニュアル』	819-2360-10
	『Sun Mainframe Batch Manager ソフトウェア ご使用にあたって (Solaris プラットフォーム用)』	819-2510-10
	『Sun Mainframe Batch Manager ソフトウェア 高可用性 (HA) データサービス (Sun Cluster 用)』	819-2511-10
Sun Mainframe Transaction Processing ソフトウェア	『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』	819-2514-10
	『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』	819-2515-10
	『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』	819-2516-10
	『Sun Mainframe Transaction Processing ソフトウェア インストールガイド』	819-2517-10
	『Sun Mainframe Transaction Processing ソフトウェア メッセージガイド』	819-2518-10
	『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』	819-2519-10
	『Sun Mainframe Transaction Processing ソフトウェア 障害追跡とチューニング』	819-2520-10
	『Sun Mainframe Transaction Processing ソフトウェア XA リソースマネージャーの使用』	819-2358-10
	『Sun Mainframe Transaction Processing ソフトウェア ご使用にあたって (Solaris プラットフォーム用)』	819-2521-10
	『Sun Mainframe Transaction Processing ソフトウェア 高可用性 (HA) データサービス (Sun Cluster 用)』	819-2522-10
IBM MVS	『IBM MVS/ESA JCL Reference』	GC28-1479
IBM VSE	『IBM VSE/ESA System Control Statements』	SC33-6713
	『IBM VSE/POWER Administration and Operation』	SC33-6733
Server Express	Server Express のマニュアル	*
ACUCOBOL-GT®	ACUCOBOL-GT のマニュアル	*

製品	タイトル	Part No.
Open PL/I	『Liant Open PL/I User's Guide』	*
	『Liant Open PL/I Language Reference Manual』	*
	『Liant CodeWatch Reference Manual』	*
C	C コンパイラのマニュアル	*

* これらのマニュアルは、ご使用のプラットフォームによって異なります。プラットフォームに該当するマニュアルについては、ご購入先にお問い合わせください。

Sun のマニュアルの注文方法

日本語版を含め、Sun のマニュアルは次のサイトで、表示や印刷、または購入ができます。

`http://www.sun.com/documentation`

Sun の技術サポート

この製品に関して、このマニュアルでも解決しない技術的な質問がある場合は、次のサイトからお問い合わせください。

`http://www.sun.com/service/contacting`

コメントをお寄せください

マニュアルの品質改善のため、お客様からのご意見およびご要望をお待ちしております。コメントは下記よりお送りください。

`http://www.sun.com/hwdocs/feedback`

ご意見をお寄せいただく際には、下記のタイトルと Part No. を記載してください。

『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』、Part No. 819-2509-10

第1章

ジョブ管理の概念

この章では、Sun Mainframe Batch Manager (Sun MBM) に関する次の概念について説明します。

Sun MBM のジョブ管理機能によって、次のことが可能になります。

- 有効なバッチジョブを表示する
- 実行およびスケジューリングを待機しているジョブの状態を表示する
- 実行中のジョブの状態を表示する
- ジョブの実行に関する現行のコマンドと手順を表示する
- ジョブのサブミットをユーザー ID 別およびグループ別に制御する
- ジョブの状態または優先順位を変更する
- ジョブの順序付けと同期化が行われるようにジョブを制御する
- ジョブを取り消す
- ジョブを中断し再開する
- ジョブの監査用にジョブアカウンティングファイルを作成する
- 統計情報を参照する

また、Job Editor を使用してジョブを作成することもできます。詳細は、第 3 章を参照してください。

サブシステム

サブシステムは、ジョブを実行するための環境を定義し、ジョブと Sun Mainframe Transaction Processing (Sun MTP) 領域、またはデータベース管理システム (DBMS) とのインタフェースを実現します。ジョブをサブミットするには、事前にサブシステムを作成する必要があります。サブシステムの作成時に、サブシステムの実行環境を設定する構成ファイルが作成されます。サブシステムの作成方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

同じハードウェアプラットフォーム上に複数の Sun MBM ノードをインストールできます。各ノードには、1 ～ 8 つのサブシステムを作成できます。

図 1-1 は、2 つのジョブがキューで待機し、1 つのジョブが実行中のサブシステム `cblsys` を示します。

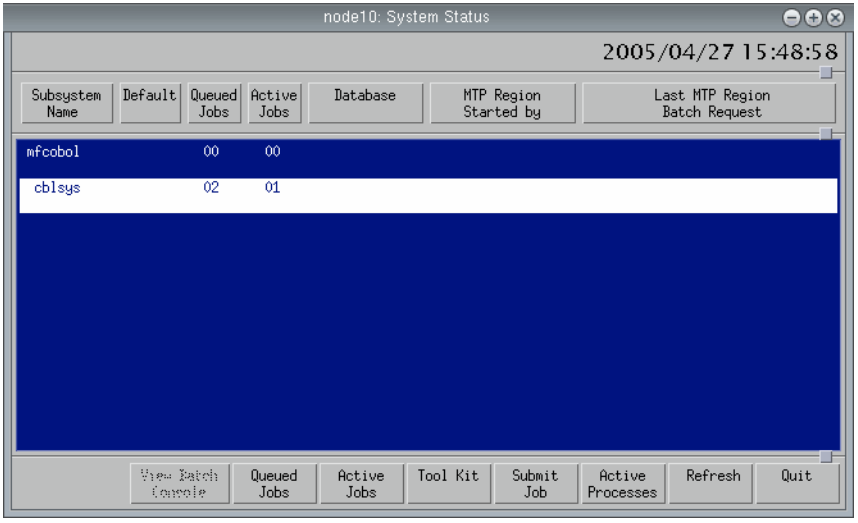

図 1-1 Sun MBM サブシステムの例

ジョブ

サブシステムにサブミットされたジョブは、1 つ以上のバッチアプリケーションプログラムを実行します。

- Sun MBM は、IBM メインフレームから移行されたバッチ環境に対応できるよう設計されています。
- ジョブは、Sun MBM JCL トランスレータにより MVS および VSE JCL ストリームから作成されます。
- ジョブには、次のものを含むことができます。
  - Sun MBM マクロ文 (『Sun Mainframe Batch Manager ソフトウェア リファレンスマニュアル』を参照)
  - C シェル文
  - システムコマンド
  - Sun MBM バッチシェル組み込みコマンド。これらのコマンドの詳細は、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

図 1-2 は、実行中の有効なジョブが表示されている画面を示します。

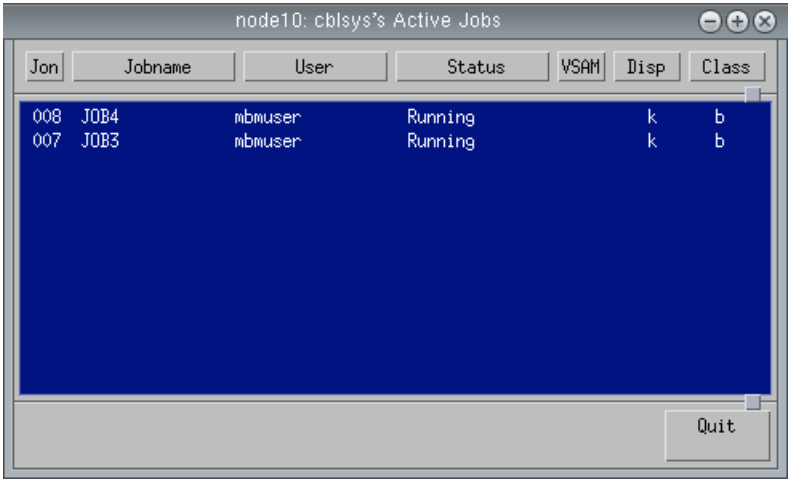

図 1-2　Active Jobs の例

ジョブクラス

Sun MBM では、ジョブクラスを使用して次のようにバッチジョブの実行を制御できます。

- ジョブをサブミットする際には、そのジョブが実行されるクラスを指定します。
- 各ジョブクラスには、アクティビティー数 (図 1-3 の 「Total Threads」) を指定できます。アクティビティーとは、ジョブを実行するために必要な共用メモリーのパーティション (セグメント)です。各ジョブは 1 つのアクティビティー内で実行され、そのアクティビティーはジョブが完了するまで使用されます。
- 1 つのクラスで同時に実行できるジョブの数 (「Active Threads」) は、アクティビティーの数によって決まります。

クラスとアクティビティーの詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

図 1-3 は、「Job Classes」画面の例を示します。

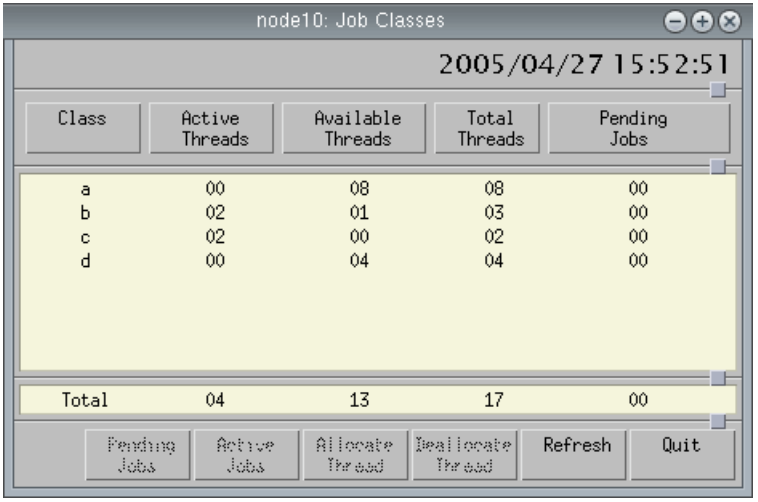

図 1-3 Job Classes の例

ジョブアカウンティング機能

ジョブアカウンティング機能は複数のツールで構成されており、それらのツールを使用することで、ユーザー定義のアプリケーションプログラムやシェルスクリプトを含むアカウンティングシステムを構築できます。

ジョブアカウンティングが有効な場合、ジョブ終了時に、1 つ以上のユーザー定義レコードがジョブアカウンティングファイルに書き込まれます。したがって、監査の目的で、ジョブアカウンティングファイル内のデータをユーザープログラムで集計することができます。

ジョブアカウンティングの設定方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

第2章

サブシステムとジョブの管理

この章の内容は、次のとおりです。

この節で示す例は、『Sun Mainframe Batch Manager ソフトウェア インストールガイド』の説明に従ってノードがインストールされていること、および『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』の説明に従って 1 つ以上のサブシステムが構成されていることを前提としています。

ノードの起動と停止は、Sun MBM メインメニュー、または Batch Administration Manager (BAM) を使用して実行できます。BAM の使用方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

サブシステムの監視

サブシステムのアクティビティーを監視する場合、「System Status」画面からキューで待機中のジョブおよび有効なジョブを表示できます。詳細は、19 ページの「有効なジョブの管理」も参照してください。

▼ 「System Status」画面を表示する

1. Sun MBM メインメニューを表示します。
2. Sun MBM ノードを起動します。

3. メインメニューの「System Status」アイコンをクリックします。

「System Status」画面が表示されます。

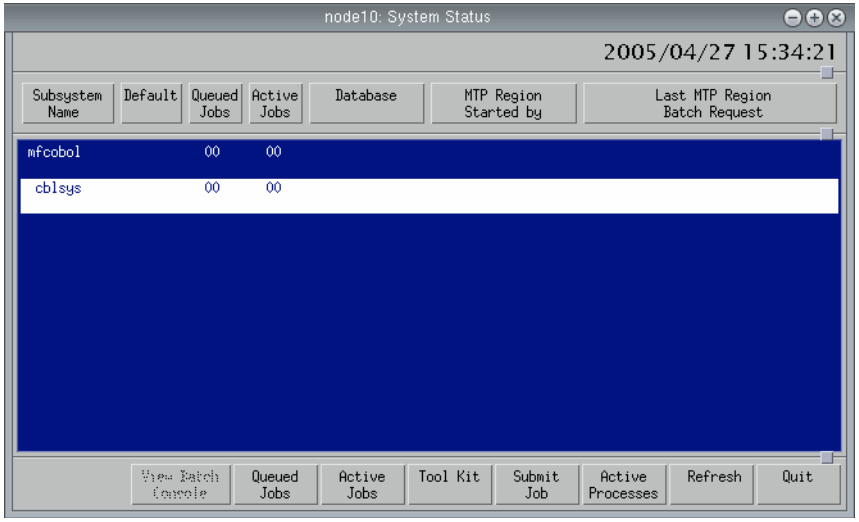

図 2-1　「System Status」画面

「System Status」画面には、次の情報が表示されます。

日付/時刻	その情報が取得された日付/時刻。自動的に再表示されます。
Subsystem Name	ノードに定義されているサブシステム名。
Default	デフォルトのサブシステムの場合は、Y が表示されます。
Queued Jobs	サブシステムのキューに置かれているジョブの合計数。
Active Jobs	サブシステムの有効なジョブの合計数。
Database	サブシステムがアクセスするデータベース。
MTP Region Started by	Sun MTP 領域を起動したユーザー。
Last MTP Region Batch Request	MTP 領域から最後にジョブの実行が要求された日時。

次の表は、「System Status」画面に表示されるボタンとその機能について示します。

ボタン	機能
View Batch Console	コンソールファイルを表示します (『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』の説明に従ってファイルが割り当てられている場合)。
Queued Jobs	選択したサブシステムのキューに置かれているジョブの詳細を表示します。
Active Jobs	選択したサブシステムの有効なジョブの詳細を表示します。
Tool Kit	「Tool Kit」ウィンドウを表示します。ツールキットを使用すると、JCL ジョブまたは手続きの妥当性検査や変換、あるいは COBOL プログラムのコンパイルを行うことができます。ツールキットの詳細は、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。
Submit Job	「Submit Job」画面 (12 ページの「サブシステムへのジョブのサブミット」を参照) を表示し、選択したサブシステムに対してサブミットできるジョブのリストを出力します。 選択したサブシステムをダブルクリックする方法でも、この画面を表示できます。
Active Processes	システムで実行中の有効なプロセスをすべて表示します。
Refresh	画面を再表示します。
Quit	画面を閉じます。

▼ サブシステムのキューで待機中のジョブを表示する

1. **「System Status」画面で、対象サブシステムを選択します。**
2. **「Queued Jobs」ボタンをクリックします。**
   「Queued Jobs」画面が表示されます。

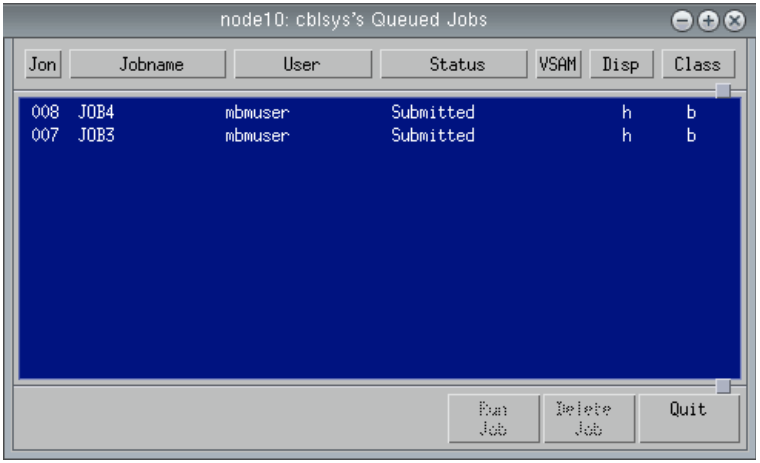

図 2-2 「Queued Jobs」画面

「Queued Jobs」画面には、次の情報が表示されます。

Jon	ジョブに割り当てられているジョブ番号。
Jobname	ジョブの名前。
User	ジョブをサブミットしたユーザーの名前。
Status	ジョブの現在の状態。次のいずれかで示されます。 Submitted (サブミット済み) Aborted (強制的な中止) Finished (完了)
VSAM	Sun MTP VSAM データセットへのアクセスを必要とするジョブかどうかを示します。
Disp	ジョブに対するディスポジションを示します。 d: ジョブを実行するためにディスパッチします。実行後、ジョブは SYSIN ファイルから削除されます。 h: 実行するジョブを保持します。実行後、ジョブは SYSIN ファイルに保持されます。 k: 保存します。ジョブを実行するためにディスパッチします。完了後、ディスポジションは h に変更され、ジョブは SYSIN ファイルに保持されます。
Class	サブミットされたジョブのクラス。

次の表は、「Queued Jobs」画面に表示されるボタンとその機能について示します。

ボタン	機能
Run Job	選択したジョブをサブミットします。ジョブのディスポジションは k に変更され、ジョブは実行後も SYSIN ファイルに保持されます。
Delete Job	選択したジョブを削除します。
Quit	画面を閉じます。

注 – この画面は自動的に再表示されません。キューに置かれているジョブの最新状態を確認するには、画面を選択し直す必要があります。

▼ サブシステムの有効なジョブを表示する

1. 「System Status」画面で、対象サブシステムを選択します。
2. 「Active Jobs」ボタンをクリックします。
   「Active Jobs」画面が表示されます。

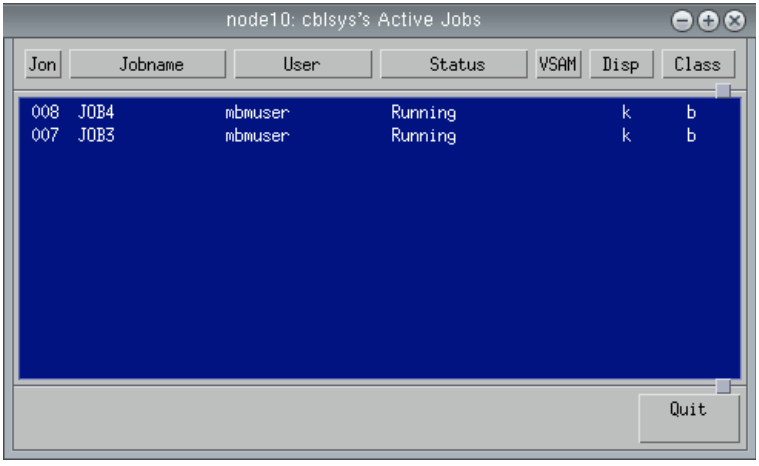

図 2-3 「Active Jobs」画面

「Active Jobs」画面には、次の情報が表示されます。

Jon	ジョブに割り当てられているジョブ番号。
Jobname	ジョブの名前。
User	ジョブをサブミットしたユーザーの名前。
Status	ジョブの現在の状態。次のいずれかで示されます。 Running (実行中)、Suspended (中断)、Aborting (強制的に中止中)
VSAM	Sun MTP VSAM データセットへのアクセスを必要とするジョブかどうかを示します。
Disp	ジョブに対するディスポジションを示します。 d: 削除します。ジョブを実行するためにディスパッチします。実行後、ジョブは SYSIN ファイルから削除されます。 h: 保持します。ジョブを SYSIN ファイル内に保持します。そのジョブを実行するためのスケジューリングは行われません。 k: 保存します。ジョブを実行するためにディスパッチします。完了後、ディスポジションは h に変更され、ジョブは SYSIN ファイルに保持されます。
Class	ジョブに割り当てられているジョブクラス。

この画面を閉じるには、「Quit」ボタンを押します。

注 – この画面は自動的に再表示されません。キューに置かれているジョブの最新状態を確認するには、画面を選択し直す必要があります。

サブシステムへのジョブのサブミット

この節では、実行およびデバッグするためにジョブをサブシステムに対してサブミットする方法について説明します。

▼ ジョブをサブミットする

1. **メインメニューを表示します。**
2. **ノードを起動します。**
3. **「System Status」画面で、ジョブのサブミットを行うサブシステムを選択します。**

4. 「Submit Job」ボタンをクリックし、「Submit Job」画面を表示します。

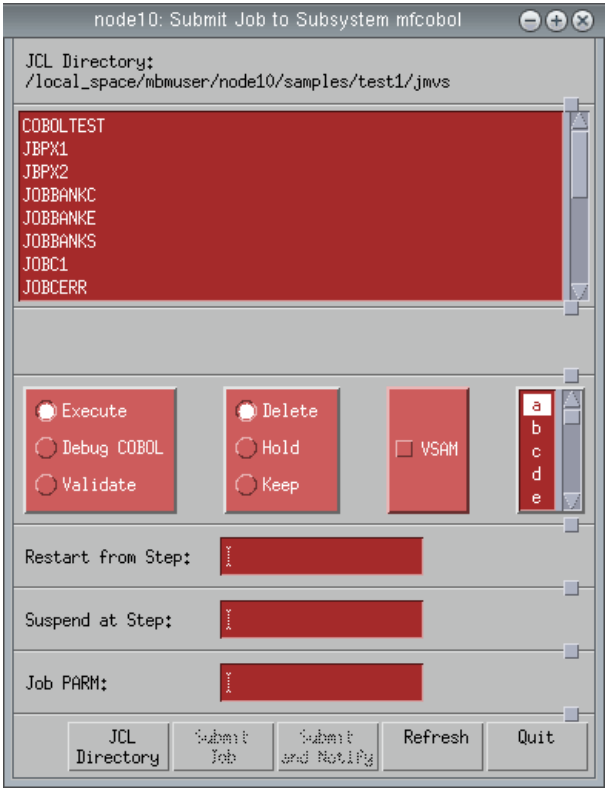

図 2-4　「Submit Job to Subsystem」画面

5. サブミットするジョブを選択し、適切なオプションを選択します。

   詳細は、15 ページの「オプションの追加指定によるジョブのサブミット」を参照してください。

6. 「Submit Job」または「Submit and Notify」をクリックします。

「Submit Job」画面には、次の情報が表示されます。

JCL Directory	ジョブが常駐するディレクトリのパス。
Execution Mode	Execute: ジョブを実行します。 Debug COBOL: COBOL デバッガを使用してジョブをデバッグします。 Validate: ジョブを妥当性検査モードで実行し、ファイルへの書き込みを行わないで、すべてのファイルおよびプログラムが存在するかどうかを検証します。
Disposition Mode	Delete: ジョブを実行するためにディスパッチします。実行後はジョブを SYSIN ファイルから削除します。 Hold: ジョブを SYSIN ファイル内に保持します。そのジョブを実行するためのスケジューリングは行われません。 Keep: ジョブを実行するためにディスパッチします。完了後、ディスポジションは Hold に変更され、ジョブは SYSIN ファイルに保持されます。
Database option	VSAM: Sun MTP VSAM データセットへのアクセスを必要とするジョブかどうかを示します。
Job Class	a 〜 z のクラス。
Restart from Step	選択したジョブを指定のステップで再開します。
Suspend at Step	選択したジョブを指定のステップで中断します。
Job PARM	選択したジョブに指定のパラメータを追加します。

次の表は、「Submit Job」画面に表示されるボタンとその機能について示します。

ボタン	機能
JCL Directory	JCL ディレクトリパスを変更するか、特定のジョブ名をフィルタリングします。
Submit Job	選択したジョブをサブミットします。
Submit and Notify	選択したジョブをサブミットし、ジョブが完了した場合はユーザーに通知します。
Refresh	画面を再表示します。
Quit	画面を閉じます。

サブミットするために選択したジョブをダブルクリックすると、そのジョブの内容がエディタ画面に表示されます (図 2-5 を参照)。表示されたジョブは変更することができます。

図 2-5　　ジョブの内容を示す画面

オプションの追加指定によるジョブのサブミット

「Submit Job」画面のフィールドを使用すると、次の作業を実行できます。

- 指定したステップからジョブを再開します
- 指定したステップでジョブを中断します
- ジョブに追加パラメータを渡します
- 追加の `subjob` または `unikixjob` パラメータを入力します。詳細は、195 ページの「`subjob` - ジョブのサブミット」と 205 ページの「`unikixjob` - Sun MTP へのバッチジョブのサブミット」を参照してください。

図 2-6 は、次のパラメータを付けて `coboltest` ジョブをサブミットする方法を示します。

- ジョブを妥当性検査モードで実行するパラメータ
- ジョブを `SYSIN` ファイル内に保持するパラメータ
- ジョブをクラス `b` で実行するパラメータ
- ジョブステップ `STEP2` から再開するパラメータ
- マクロセットを冗長モード `-DMACRO` で実行するパラメータ

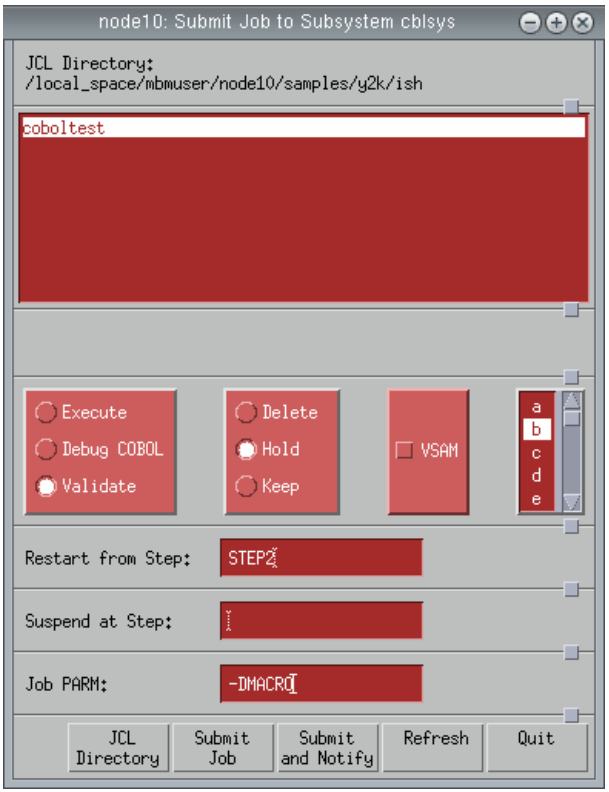

図 2-6　　オプションの追加指定によるジョブのサブミットの例

JCL ディレクトリの変更

「System Status」画面で「Submit Job」ボタンをクリックすると、サブシステム作成時に指定したジョブ用ディレクトリがデフォルトの JCL ディレクトリとなります。別のディレクトリにあるジョブをサブミットすることも可能です。

▼ JCL ディレクトリを変更する

1. 「Submit Job」画面で、「JCL Directory」ボタンをクリックします。
2. ダイアログボックスにパス名を入力します。
   環境変数を使用することもできます。

3. 「OK」をクリックします。

「Submit Job」画面に、選択した JCL ディレクトリにあるジョブのリストが表示されます。

アスタリスク (*) を使用して、ジョブ名が特定のパターンに一致するジョブだけを表示させることもできます。たとえば、ディレクトリを変更し、ACCT で始まる名前のジョブだけを表示するには、次のように実行します。

1. 「JCL Directory」ボタンをクリックします。
2. ディレクトリのパスを入力し、最後の要素を「ACCT*」と入力します。
3. 「OK」をクリックします。

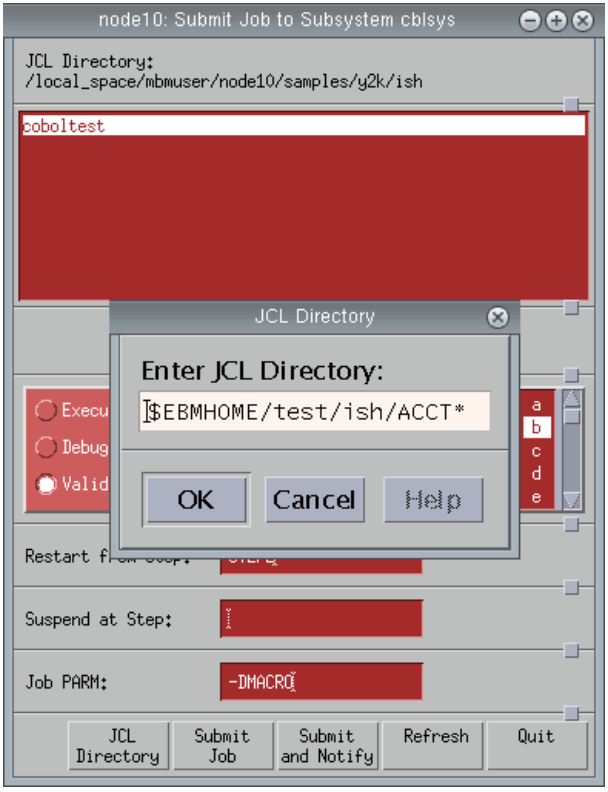

図 2-7　　JCL ディレクトリとフィルタの変更の例

ユーザーへのジョブ完了通知

「Submit and Notify」ボタンを使用してジョブをサブミットすると、ジョブの終了後に「List Completed Job」画面が表示されます。

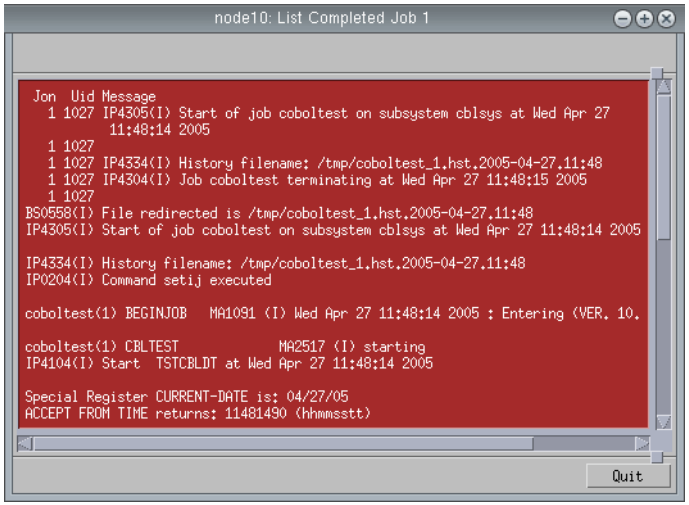

図 2-8　　完了したジョブの一覧表示の例

COBOL デバッガによるデバッグのためのジョブのサブミット

COBOL デバッガを使用するジョブをサブミットするには、事前に COBOL サブシステムを作成しておく必要があります。詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

▼ デバッグするジョブをサブミットする

1. 「Submit Job」画面を表示します。
2. ジョブを選択します。
3. 「Debug COBOL」を選択します。
4. 「Submit Job」または「Submit and Notify」ボタンをクリックします。

5. ジョブのサブミットの確認ダイアログボックスが表示されたら、ジョブ番号を書き留めます。この番号は COBOL デバッガを有効化するときに必要となります。
   詳細は、19 ページの「COBOL デバッガを有効化する」を参照してください。

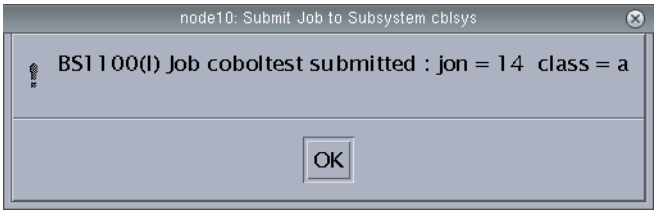

図 2-9　「Submit Job」確認ポップアップウィンドウ

6. 「OK」をクリックして確認ダイアログボックスを閉じます。

▼ COBOL デバッガを有効化する

1. 18 ページの「COBOL デバッガによるデバッグのためのジョブのサブミット」の説明に従ってジョブをサブミットします。
2. Sun MBM メインメニューで、「Command Prompt」アイコンをクリックします。
3. 作業を実行するサブシステムの名前を入力し、Return キーを押します。
4. コマンド `anmjob` *Job #* を入力し、Return キーを押します。
   「Job #」に入力するのは、ジョブのサブミット時に表示されたジョブ番号です。
5. 「COBOL デバッガ」画面が表示され、ジョブのデバッグが実行可能となります。

有効なジョブの管理

▼ ノードの有効なジョブを表示する

1. メインメニューを表示します。
2. ノードが実行中であることを確認します。

3. 「Active Jobs」アイコンをクリックします。

図 2-10 に示す画面が表示されます。

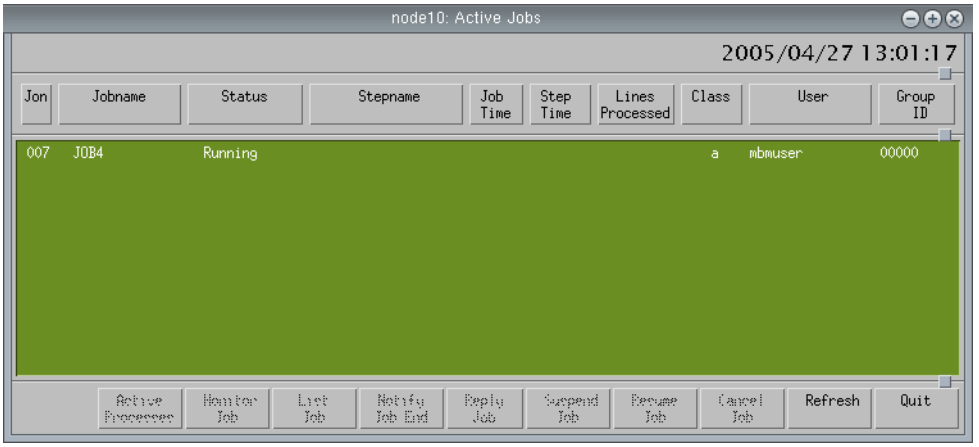

図 2-10 Active Jobs の例

「Active Jobs」画面には、次の情報が表示されます。

日付/時刻	その情報が取得された日付/時刻。自動的に再表示されます。
`Jon`	ジョブに割り当てられているジョブ番号。
`Jobname`	有効なジョブの名前。
`Status`	ジョブの現在の状態。次のいずれかで示されます。 `Running` (実行中) `Suspended` (中断) `Aborting` (強制的に中止中) `Waiting Reply` (応答待ち)
`Stepname`	現在実行中のステップの名前。
`Job Time`	ジョブの実行経過時間。
`Step Time`	現行ステップの実行経過時間。
`Lines Processed`	ジョブストリームのうち処理された行数。
`Class`	ジョブが実行されているクラス。
`User`	ジョブをサブミットしたユーザーの名前。
`Group ID`	制御プロセス `btsh` のプロセス ID。

次の表は、「Active Jobs」画面に表示されるボタンとその機能について示します。

ボタン	機能
Active Processes	選択したジョブの有効なプロセスを表示します。
Monitor Job	ジョブ履歴を監視します。選択したジョブをダブルクリックする方法でも、ジョブを監視できます。
List Job	ジョブ履歴画面を表示します。
Notify Job End	ジョブ終了後に通知を表示します。
Reply Job	ジョブの状態が応答待ちの場合の応答を入力できます。
Suspend Job	選択したジョブを中断します。
Resume Job	中断したジョブを再開します。
Cancel Job	選択したジョブを取り消します。ジョブを取り消すと、確認ポップアップウィンドウが表示されます。
Refresh	画面を再表示します。
Quit	画面を閉じます。

ジョブの監視

ジョブの実行中に、スクロール可能なウィンドウにジョブ履歴を表示できます。この画面へのジョブ出力の書き込みは、そのジョブが終了するまで続きます。

▼ ジョブ履歴を表示する

1. Sun MBM メインメニューで、「Active Jobs」をクリックします。
2. 「Active Jobs」画面で、目的のジョブを選択します。
3. 「Monitor Job」ボタンをクリックするか、選択したジョブをダブルクリックします。

   図 2-11 は、「Monitor Job」画面の例を示します。

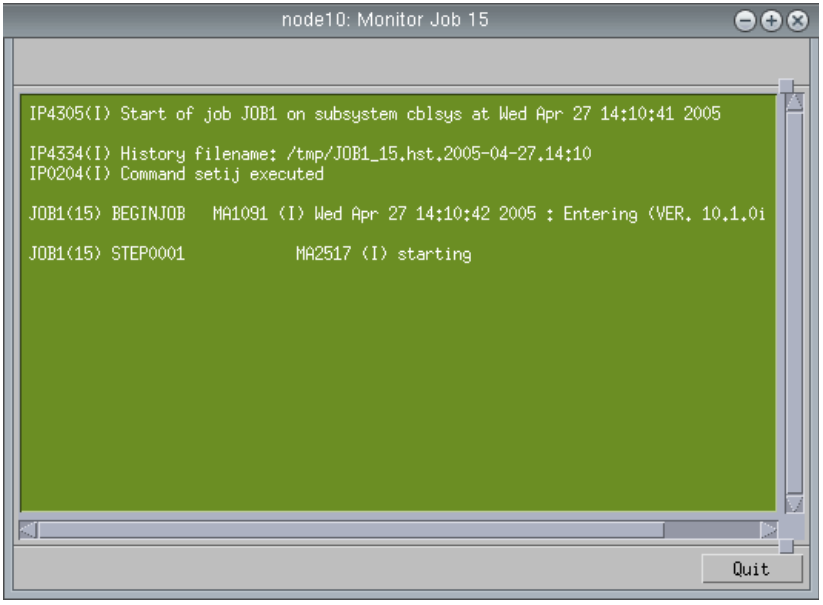

図 2-11　「Monitor Job」画面の例

有効なジョブの履歴の表示

「Active Jobs」画面の「List Job」機能は、21 ページの「ジョブの監視」で説明する「Monitor Job」機能と似ています。ただし「List Job」機能では、ユーザーが「List Job」ボタンをクリックした時点までの出力だけが表示されます。

▼ 有効なジョブの履歴を表示する

1. 「Active Jobs」画面で、目的のジョブを選択します。
2. 「List Job」ボタンをクリックします。

   図 2-12 は、ジョブ履歴画面の例を示します。

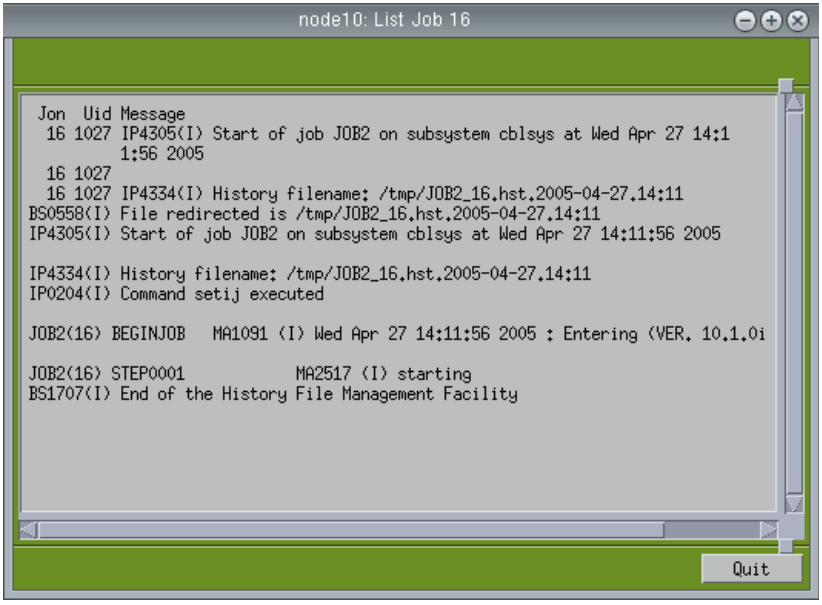

図 2-12　ジョブ履歴の例

▼ 実行中のジョブを中断する

1. 「Active Jobs」画面で、目的のジョブを選択します。
2. 「Suspend Job」ボタンをクリックします。
3. 確認ポップアップウィンドウが表示されたら、「OK」をクリックしてウィンドウを閉じます。

▼ 中断したジョブを再開する

1. 「Active Jobs」画面で、中断したジョブを選択します。
2. 「Resume Job」ボタンをクリックします。
3. 確認のポップアップウィンドウが表示されたら、「OK」をクリックしてウィンドウを閉じます。

▼ 有効なジョブを取り消す

1. 「Active Jobs」画面で、有効なジョブを選択します。
2. 「Cancel Job」ボタンをクリックします。
3. ダイアログボックスが表示されたら、次のいずれかのアクションを行います。
   - 「Cancel Job」をクリックしてジョブを強制的に中止します。
   - 「Cancel With Dump」をクリックします。このボタンをクリックしたときにジョブが Server Express プログラムを実行中の場合、Server Express ダンプが生成されます。ダンプの内容については、Micro Focus FaultFinder のマニュアルを参照してください。
   - ジョブを強制終了しない場合は、「Do Not Cancel」をクリックします。

   取り消しを指示したジョブは、すべての一時ファイルの回復と、すべてのオープンデータセットの復元を行う必要があるため、実際に取り消されるまで数秒間かかります。

ジョブの終了通知の取得

ジョブのサブミット時に「Submit and Notify」をクリックしなかった場合でも、ジョブの完了が通知されるようにすることができます。次の手順に従います。

▼ ジョブの終了通知を取得する

1. 「Active Jobs」画面で、有効なジョブを選択します。
2. 「Notify Job End」ボタンをクリックします。

   ジョブが終了すると、図 2-13 に示す画面が表示されます。

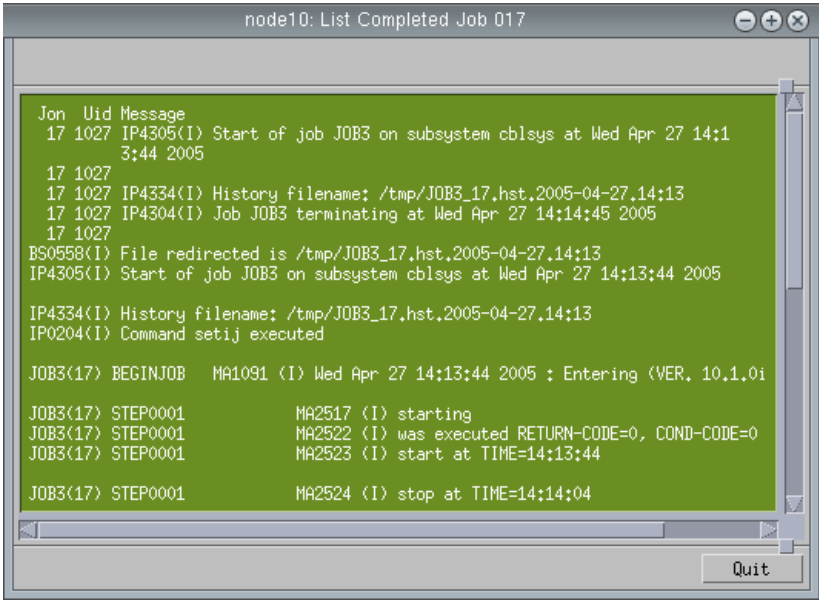

図 2-13　完了したジョブの例

オペレータ入力を必要とするジョブへの応答

「Active Jobs」画面では、オペレータ入力を必要とするジョブへの応答ができます。

▼ ジョブに応答する

1. **ジョブの「Status」列に「`Waiting Reply`」が表示されている場合は、そのジョブを選択して「Reply Job」ボタンをクリックします。**
2. **ポップアップウィンドウに要求された情報を入力し、「OK」をクリックします。**

   応答がジョブに送信されたことを示すメッセージが表示されます。「Status」列が「`Running`」に変わり、ジョブが実行中であることが示されます。
3. **「OK」をクリックし、ポップアップウィンドウを閉じます。**

ジョブクラスの監視

「Job Classes」画面では、ノードに定義されている全サブシステムに対してサブミットされた、有効なジョブと保留状態のジョブのすべてをクラス別に表示できます。クラス別の情報を分析することにより、クラスとスレッドの数がジョブを効率的に処理する上で十分かどうかがわかります。画面上のボタンをクリックすると、スレッドの割り当ておよび割り当て解除を動的に実行できます。

▼ 「Job Classes」画面を表示する

- Sun MBM メインメニューの「Job Classes」アイコンをクリックします。

  「Job Classes」画面が表示されます。

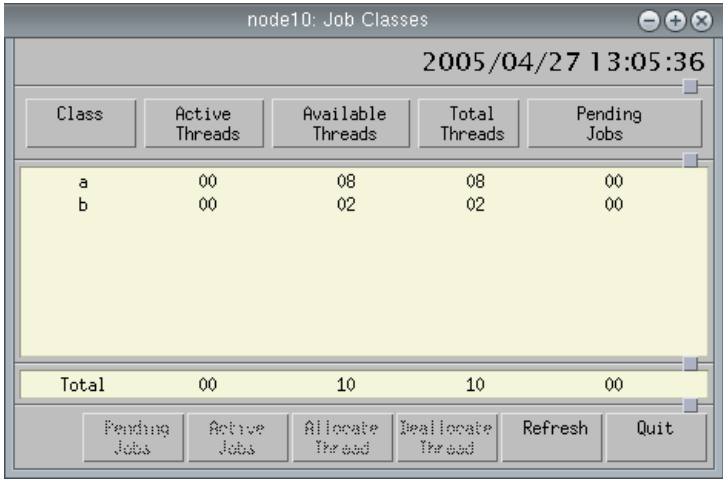

図 2-14　Job Classes 画面

「Job Classes」画面には、次の情報が表示されます。

日付/時刻	その情報が取得された日付/時刻。自動的に再表示されます。
`Class`	a ～ z のジョブクラス。
`Active Threads`	実行中のジョブに割り当てられているスレッド (アクティビティー) の数。
`Available Threads`	使用されていないスレッド (アクティビティー) の数。
`Total Threads`	クラスに割り当てられているスレッド (アクティビティー) の最大数。
`Pending Jobs`	実行を待機しているジョブの数。

次の表は、「Job Classes」画面に表示されるボタンとその機能について示します。

ボタン	機能
Pending Jobs	選択したクラスで実行を待機しているジョブを一覧表示します。ジョブ数が使用可能なスレッド数よりも多いため、ジョブは待機しています。
Active Jobs	選択したクラスで現在実行中のジョブを一覧表示します。
Allocate Thread	選択したクラスにスレッド (アクティビティー) を追加します。
Deallocate Thread	選択したクラスからスレッド (アクティビティー) を削除します。
Refresh	画面を再表示します。
Quit	画面を閉じます。

▼ 特定のクラスのジョブを表示する

特定のクラスについて、有効なジョブまたは保留状態のジョブを一覧表示できます。

1. **「Job Classes」画面で、目的のクラスを選択します。**
2. **「Active Jobs」または「Pending Jobs」ボタンをクリックします。**

図 2-15 は、「Active Jobs」画面と「Pending Jobs」画面の例を示します。

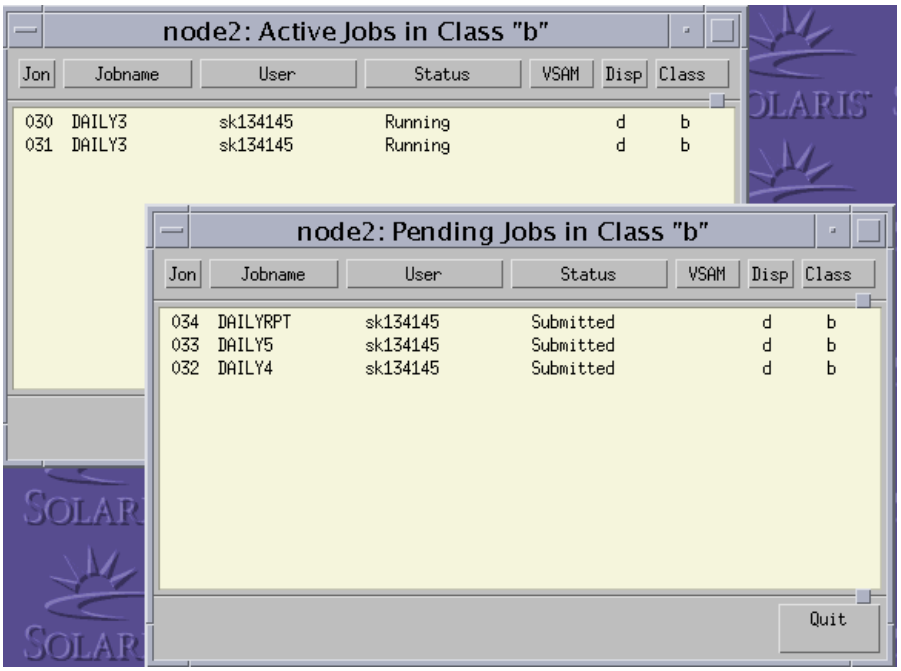

図 2-15　「Active Jobs」画面と「Pending Jobs」画面の例

これらの画面には、次の情報が表示されます。

`Jon`	ジョブに割り当てられているジョブ番号。
`Jobname`	有効なジョブの名前。
`User`	ジョブをサブミットしたユーザーの名前。
`Status`	ジョブの現在の状態。次のいずれかで示されます。 有効なジョブの場合 • `Running` (実行中) • `Suspended` (中断) • `Aborting` (強制的に中止中) • `Waiting Reply` (応答待ち) 保留状態のジョブの場合 • `Submitted` (サブミット済み) • `Aborted` (強制的な中止) • `Finished` (完了)

VSAM	Sun MTP VSAM データセットへのアクセスを必要とするジョブかどうかを示します。
Disp	ジョブに対するディスポジション。 d: 削除します。ジョブを実行するためにディスパッチします。実行後、ジョブは SYSIN ファイルから削除されます。 h: 保持します。そのジョブを実行するためのスケジューリングは行われません。ジョブを SYSIN ファイル内に保持します。 k: 保存します。ジョブを実行するためにディスパッチします。完了後、ディスポジションは h に変更され、ジョブは SYSIN ファイルに保持されます。
Class	ジョブが実行されているクラス。

画面を閉じるには、「Quit」ボタンをクリックします。

注 – この画面は自動的に再表示されません。保留状態のジョブまたは有効なジョブの最新状態を確認するには、そのジョブを選択し直す必要があります。

ジョブクラスへのスレッドの割り当て

ジョブを時間どおり確実に完了させるため、ジョブクラスに追加スレッドを割り当てることが必要となる場合があります。また、存在しないクラスにジョブをサブミットすると、Sun MBM はスレッドが 0 のクラスを作成し、そのジョブを保留状態に置きます。そのジョブは、あとからスレッドを割り当てたうえで実行することができます。

ノードに割り当て可能なスレッドの総数は、そのノードの構成時にユーザーが設定したジョブクラスとアクティビティーの値およびアクティビティーの最大数によって決まります。BAM でクラスとアクティビティーを作成する方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

注 – crtact コマンドを使用してクラスを作成した場合は、この機能が正しく動作しないことがあります。ジョブクラスを作成するには、BAM を使用します。

▼ スレッドをクラスに割り当てる

1. **「Job Classes」画面で、スレッドの追加が必要なクラスを選択します。**

   この例では、クラス b を選択します。図 2-16 は、クラス b の使用可能なスレッド数が 00 を示しているので、2 つのジョブが保留状態になっていることを表しています。

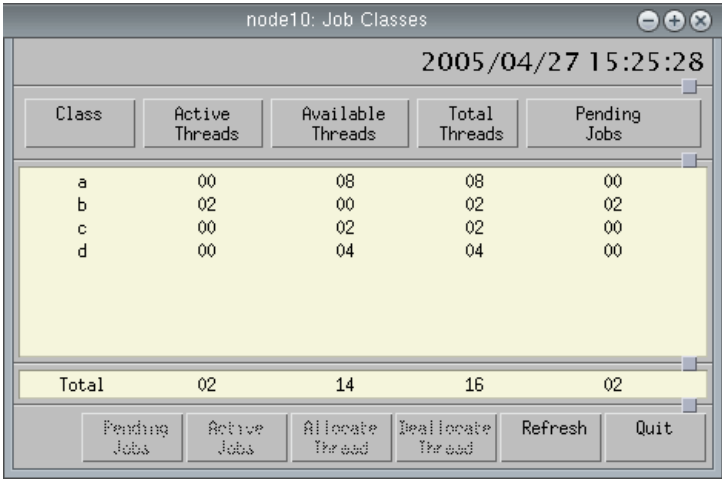

図 2-16　ジョブクラスへのスレッドの割り当て

2. 「Allocate Thread」ボタンをクリックします。

   保留状態のジョブは使用可能なスレッドを使用します。図 2-17 では、クラス `b` に対して有効な 3 つのスレッドを示します。

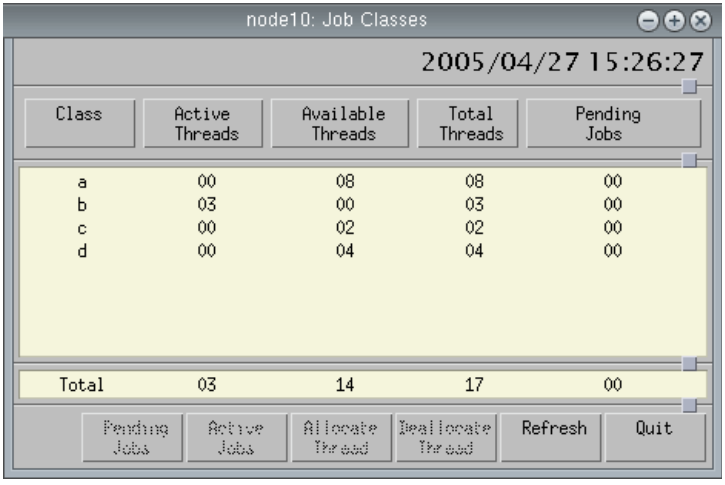

図 2-17　スレッドの割り当ての例

ジョブクラスのスレッドの割り当て解除

保留状態のジョブがすべて実行されたら、追加スレッドの割り当てを解除します。

▼ スレッドを割り当て解除する

1. 「Job Classes」画面で、目的のクラスを選択します。

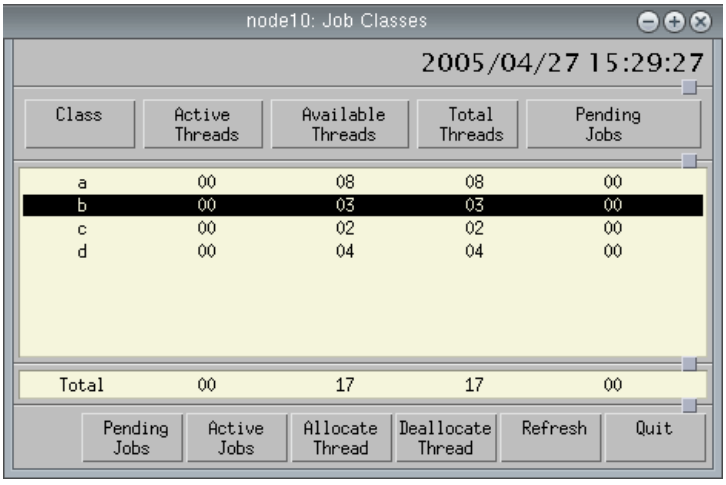

図 2-18　ジョブクラスからのスレッドの割り当て解除

2. 「Deallocate Thread」ボタンをクリックします。

スレッドの総数が 1 つずつ減ります。複数のスレッドの割り当てを解除するには、この手順を繰り返します。

注 – `crtact` コマンドを使用してクラスを作成した場合は、この機能が正しく動作しないことがあります。ジョブクラスを作成するには、BAM を使用します。

完了したジョブの状態と統計情報の表示

「Completed Jobs」画面には、完了したジョブの状態と統計情報が表示されます。これらの情報は、有効なジョブログまたは保存されている照会ファイルに基づいて生成されます。その方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

見出しのボタンをクリックすると、完了したジョブのリストを昇順または降順にソートできます。デフォルトでは、「Ended At」フィールドに表示されるジョブの終了日時によって降順でソートされます。したがって、最後に完了したジョブが最初に表示されます。

例: ジョブ名を基準にしてジョブをソートするには、「Job Name」ボタンをクリックします。これにより、ジョブ名を基準にして昇順にソートされたリストが表示されます。「Job Name」ボタンをもう一度クリックすると、完了したジョブが降順で表示されます。

▼ 完了したジョブのリストを表示する

1. **メインメニューを表示します。**
2. **Sun MBM メインメニューの「Completed Jobs」アイコンをクリックします。**

   図 2-19 は、「Completed Jobs」画面の例を示します。

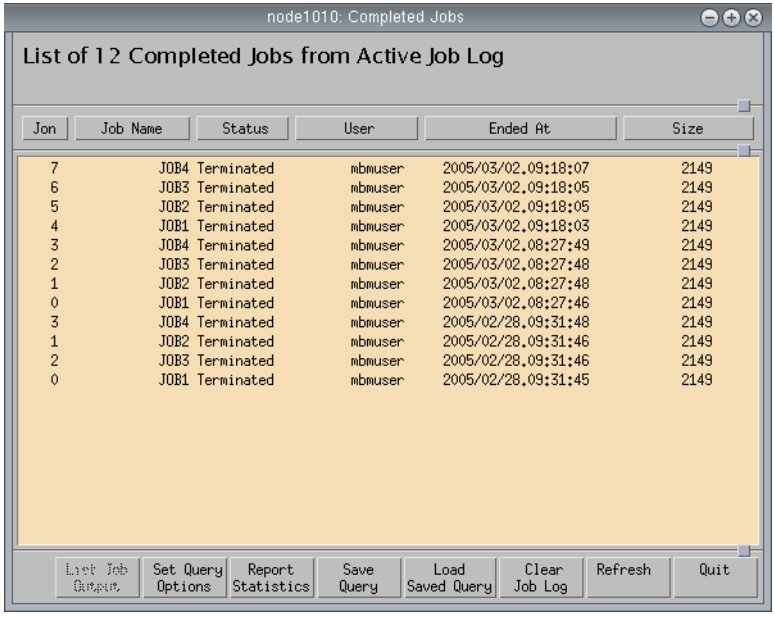

図 2-19 「Completed Jobs」画面の例

「Completed Jobs」画面には、次の情報が表示されます。

`List of` *nnnn* `Completed Jobs from` *file-name*	有効なジョブログまたは保存されている照会ファイルに表示された完了ジョブの数。
`Jon`	ジョブに割り当てられているジョブ番号。
`Job Name`	完了したジョブの名前。
`Status`	完了したジョブの状態。次のいずれかで示されます。 `Terminated` (終了) `Aborted` (強制的な中止) `Cancelled` (取り消し)
`User`	ジョブをサブミットしたユーザーのユーザー ID。
`Ended At`	ジョブが完了した日付と時刻が次の形式で表示されます。 *YYYY/MM/DD.hh:mm:ss*
`Size`	履歴ファイルのサイズ (バイト単位)。

次の表は、「Completed Jobs」画面に表示されるボタンとその機能について示します。

ボタン	機能
List Job Output	ジョブ履歴と統計情報を表示します。
Set Query Options	完了したジョブのリストを特定の値によって照会します。
Report Statistics	完了したジョブに関する統計情報を表示します。
Save Query	照会結果をファイルに保存します。
Load Saved Query	保存されている照会ファイルを開き、完了ジョブのリストを表示します。
Clear Job Log	当日または 1 日以上前に終了したジョブに関するログエントリを削除します。
Refresh	有効なジョブログを再度読み込みます。
Quit	画面を閉じます。

▼ 完了したジョブのジョブ出力を一覧表示する

1. 「Completed Jobs」画面でジョブを選択します。
2. 「List Job Output」ボタンをクリックするか、選択したジョブをダブルクリックします。

   図 2-20 に、この画面の例を示します。

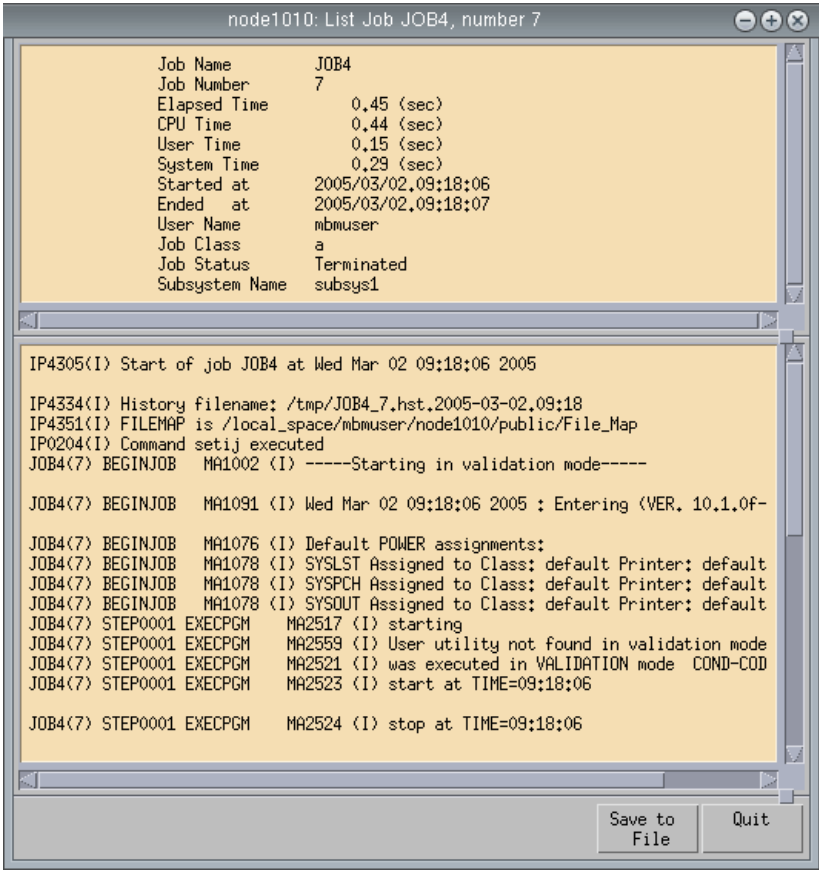

図 2-20　完了したジョブの情報の表示例

この画面には、次の情報が表示されます。

`Job Name`	ジョブの名前。
`Job Number`	ジョブに割り当てられているジョブ番号。
`Elapsed Time`	ジョブの経過クロック時間。次の形式で表示されます。 *seconds.hundredths-of-seconds*
`CPU Time`	ジョブの累計実行時間。次の形式で表示されます。 *seconds.hundredths-of-seconds*
`User Time`	ジョブの累計ユーザー実行時間。次の形式で表示されます。 *seconds.hundredths-of-seconds*
`System Time`	ジョブの累計システム実行時間。次の形式で表示されます。 *seconds.hundredths-of-seconds*
`Started at`	ジョブの開始日付/時刻。次の形式で表示されます。 *YYYY/MM/DD.hh:mm:ss*
`Ended at`	ジョブの終了日付/時刻。次の形式で表示されます。 *YYYY/MM/DD.hh:mm:ss*
`User Name`	ジョブをサブミットしたユーザーの名前。
`Job Class`	ジョブが実行されたジョブクラス。
`Job Status`	ジョブの状態。次のいずれかで示されます。 `Terminated` (終了) `Aborted` (強制的な中止) `Cancelled` (取り消し)
`Subsystem Name`	ジョブが実行されたサブシステムの名前。

次の表はこの画面に表示されるボタンとその機能について示します。

ボタン	機能
Save to File	ジョブ出力の一覧をファイルに保存します。
Quit	画面を閉じます。

完了したジョブ一覧の照会

完了したジョブのリストをフィルタリングするための照会条件を作成し、その条件に一致したジョブだけを表示させるには、次の手順に従います。

▼ 完了したジョブのリストを照会する

1. 「Completed Jobs」画面で、「Set Query Options」ボタンをクリックします。
2. 図 2-21 に示す「Set Query Options」ウィンドウが表示されたら、照会条件を入力します。

   照会オプションについては、表 2-1 を参照してください。

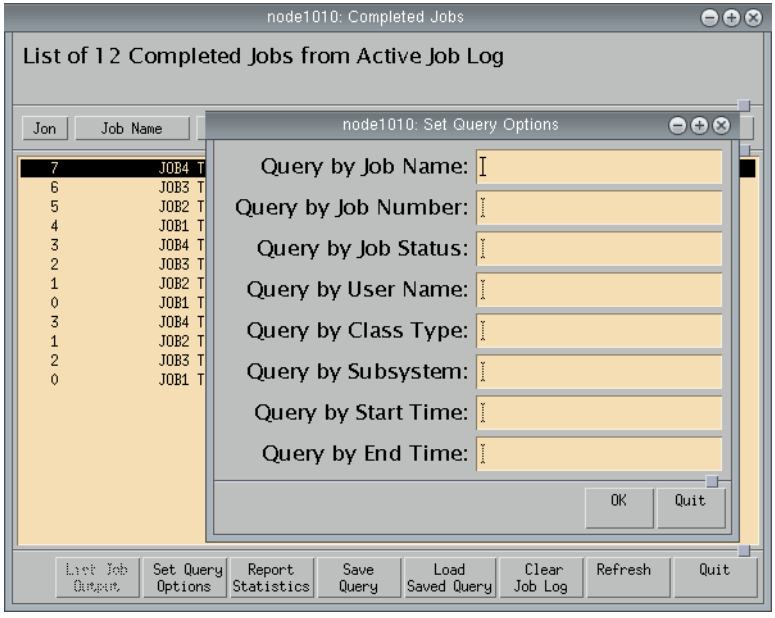

図 2-21　「Set Query Options」画面

3. 「OK」をクリックして、照会を実行します。

   「Completed Jobs」画面に、照会条件に一致したジョブが表示されます。

   「Set Query Options」ウィンドウは、「Quit」ボタンをクリックするまで有効です。

4. **ウィンドウを閉じるには、「Quit」をクリックします。**

ウィンドウを閉じると、使用可能なすべてのジョブが「Completed Jobs」画面に自動的に再表示されます。

「Set Query Options」ウィンドウで、次の情報のいずれか、またはすべてを指定して照会を作成できます。

表 2-1　照会オプション

フィールド	説明
`Query by Job Name`	ジョブ名またはジョブ名の一部を指定します。これは大文字と小文字が区別されるグローバルオプションなので、たとえば `job` と指定した場合は `job*` として認識されます。したがって、`job` で始まるすべてのジョブが照会結果となります。たとえば、`job1`、`job2`、`joba` が表示されます。
`Query by Job Number`	一意のジョブ番号を 0 ～ 999 の数字で指定します。これにより、指定した番号を持つジョブだけが照会結果になります。
`Query by Job Status`	次のいずれかの値のジョブが表示されます。 `T`: Terminated (終了) `C`: Cancelled (取り消し) `A`: Aborted (強制的な中止)
`Query by User Name`	指定したユーザーのジョブが表示されます。
`Query by Class Type`	指定したジョブクラス `a` ～ `z` のジョブが一覧表示されます。
`Query by Subsystem`	指定した Sun MBM サブシステムのジョブが一覧表示されます。
`Query by Start Time`	指定した開始日時のジョブが一覧表示されます。次のいずれかの形式で指定した日付と時刻までに開始されたすべてのジョブが出力されます。 *YYYY/MM* *YYYY/MM/DD* *YYYY/MM/DD.hh* *YYYY/MM/DD.hh:mm* *YYYY/MM/DD.hh:mm.ss*
`Query by End Time`	指定した終了時刻のジョブが一覧表示されます。次のいずれかの形式で指定した日付と時刻までに完了したすべてのジョブが出力されます。 *YYYY/MM* *YYYY/MM/DD* *YYYY/MM/DD.hh* *YYYY/MM/DD.hh:mm* *YYYY/MM/DD.hh:mm.ss*

注 – 「Query by Start Time」と「Query by End Time」の両方を指定すると、その間に実行されたすべてのジョブを照会できます。

この画面では、次のボタンが使用できます。

ボタン	機能
OK	この画面で入力した値に基づく照会が作成されます。
Quit	この画面を閉じ、「Completed Jobs」画面が再表示され、照会可能なジョブのリストが表示されます。

▼ 照会結果を保存する

照会結果のジョブ情報をバックアップファイルに保存するには、次の手順に従います。

1. 「Completed Jobs」画面で、「Save Query」ボタンをクリックします。
2. 「Save Query」ポップアップウィンドウで、ファイルのフルパス名を入力します。

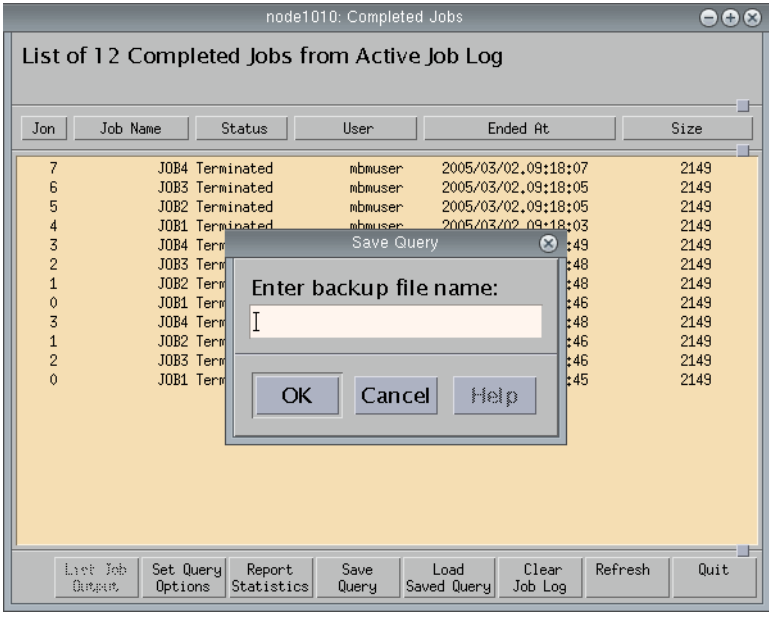

図 2-22 照会結果の保存

3. 「OK」をクリックします。

▼ 保存されている照会結果を読み込む

デフォルトでは、「Completed Jobs」画面には有効なジョブログファイルの完了ジョブが一覧表示されます。また、以前に保存したバックアップファイルも表示できます。

1. 「Completed Jobs」画面で、「Load Saved Query」ボタンをクリックします。
2. 「Load Saved Query」ポップアップウィンドウで、バックアップファイルのフルパス名を入力します。

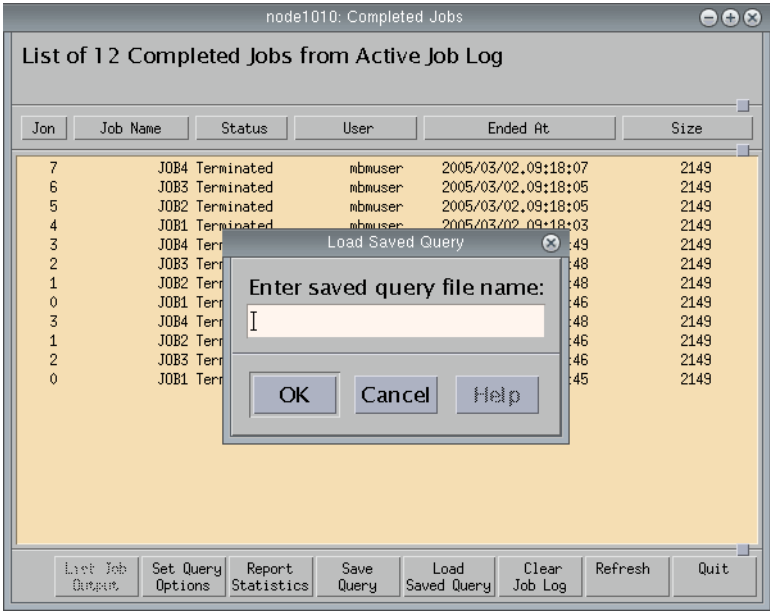

図 2-23　保存されている照会結果の読み込み

3. 「OK」をクリックします。

   「Completed Jobs」画面に、保存されている照会結果が表示されます。
4. 有効なジョブログファイルに含まれる完了ジョブのリストに戻るには、「Refresh」ボタンをクリックします。

▼ 完了したジョブに関する統計情報を表示する

- **「Completed Jobs」画面で、「Report Statistics」ボタンをクリックします。**

  図 2-24 に示すように、このレポートは 2 つのセクションで構成されています。

  - 上のセクションには、現在の「Completed Jobs」画面のすべてのジョブに関する情報が表示されます。
  - 下のセクションには、現在の「Completed Jobs」画面のジョブに関する過去 24 時間以内の情報が表示されます。

  すべての時間情報は、次の形式の秒数で表示されます。

  seconds.hundredths-of-seconds

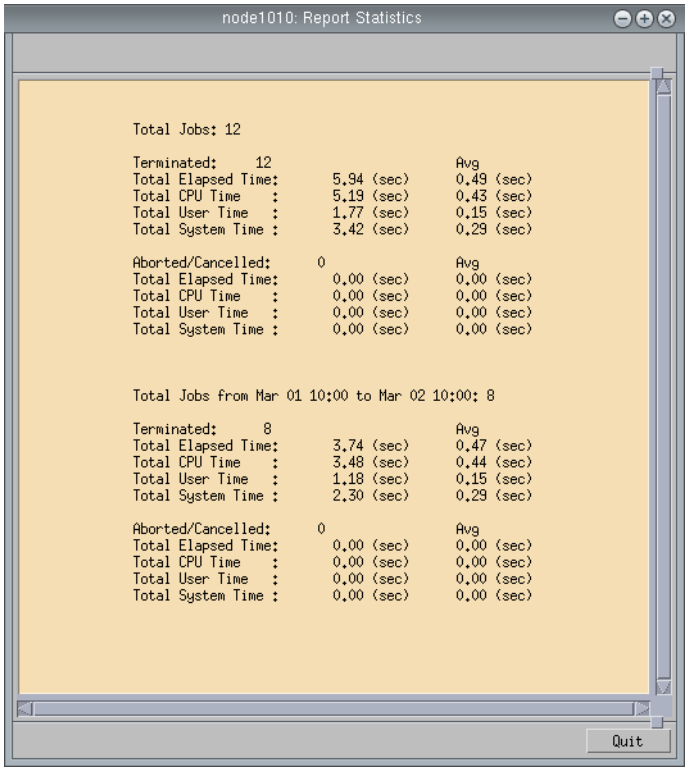

図 2-24 Report Statistics の例

「Report Statistics」画面には、次の情報が表示されます。

`Total Jobs`	完了したジョブの数。
`Total Jobs from` *MM DD hh.mm* `to` *MM DD hh:mm*	選択した照会結果のうち、最後に完了したジョブから 24 時間前の間に完了したジョブの数。
`Terminated`	終了したジョブの数。
`Total Elapsed Time`	ジョブに関する経過クロック時間の合計と平均値。
`Total CPU Time`	ジョブの実行にかかった時間の合計と平均値。
`Total User Time`	ジョブのユーザー実行時間の合計と平均値。
`Total System Time`	ジョブのシステム実行時間の合計と平均値。
`Aborted/Cancelled`	強制的に中止されたジョブまたは取り消されたジョブの数。

システムでは、次の情報を含む使用状況レポートも表示します。

% CPU Time	ジョブの合計実行時間のパーセンテージ。
% User Time	ジョブのユーザー実行時間のパーセンテージ。
% System Time	ジョブのシステム実行時間のパーセンテージ。

ジョブに関する有効なプロセスの表示

▼ 実行中のジョブの有効なプロセスを表示する

1. Sun MBM メインメニューで、「Active Jobs」をクリックします。
2. 実行中のジョブを選択し、「Active Processes」をクリックします。
   「Active Processes for Job」画面が表示されます。

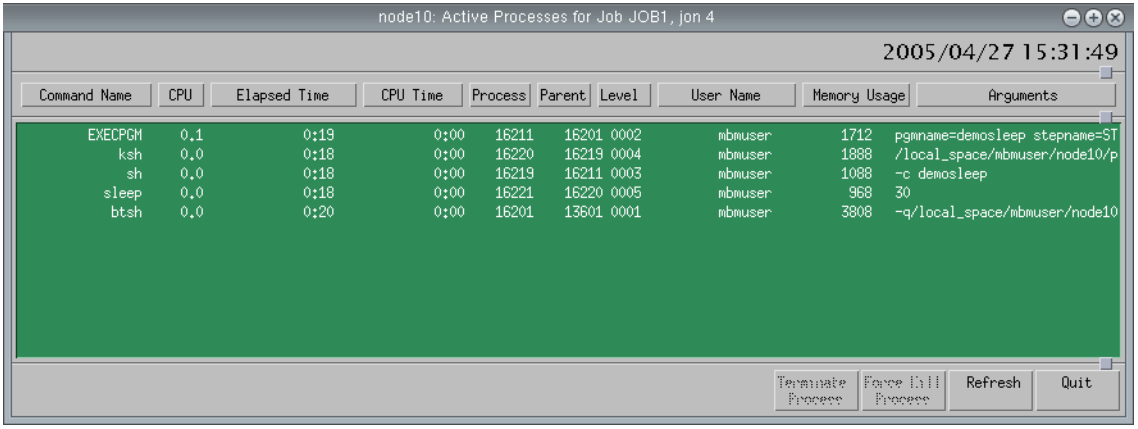

図 2-25　Active Processes for Job

「Active Processes for Job」画面には、次の情報が表示されます。

Command Name	実行されたコマンドの名前。
CPU	スケジューリングに関するプロセッサ使用率。最近の CPU 使用時間の同じ期間の有効 CPU 時間に対する比率が、パーセンテージで表示されます。
Elapsed Time	次の形式によるプロセス開始後の経過時間。 *[[dd-]hh:]mm:ss* 説明 *dd*: 日数 *hh*: 時間数 *mm*: 分数 *ss*: 秒数
CPU Time	次の形式によるプロセスの累積 CPU 時間。 *[[dd-]hh:]mm:ss* 説明 *dd*: 日数 *hh*: 時間数 *mm*: 分数 *ss*: 秒数
Process	プロセスのプロセス ID。
Parent	親プロセスのプロセス ID。
Level	プロセスの親/子連鎖でのレベル番号。通常のジョブでは、メインの `btsh` プロセスはレベル 1、その子プロセスはレベル 2、さらにレベル 2 のプロセスから生成された後続プロセスはレベル 3、というように表示されます。 有効な親プロセスのないプロセスには、9001、9002、9003 ... というレベル番号が付きます。
User Name	プロセスの実効ユーザー ID。
Memory Usage	プロセスが仮想メモリーに占める容量の合計サイズ。
Arguments	コマンド行の引数。

▼ 有効なプロセスの情報をソートする

- **情報をソートする際に基準とするフィールドに対応するボタンをクリックします。**

  同じフィールドのボタンをもう一度クリックすると、リストが逆順にソートされます。デフォルトでは、各フィールドの情報リストは「CPU」フィールドを基準にソートされます。

▼ プロセスを終了する

- **次のいずれかの方法を使用します。**
  - プロセスを選択し、「Terminate Process」をクリックします。
  - プロセスを選択し、「Force Kill」をクリックします。

  プロセスを強制終了すると、データの消失やシステムの信頼性低下などの不都合が生じる恐れがあります。プロセスは、「Terminate Process」を使用して終了します。この方法であれば、プロセスの終了前にシステムの状態および関連するデータが保存されます。「Force Kill」を使用するのは、「Terminate Process」による要求でプロセスが終了できない場合だけにしてください。

注意 – ジョブに関連付けられているメインの `btsh` プロセスは、「Job Active Processes」画面では終了も強制終了もできません。このプロセスを終了または強制終了するには、「Active Jobs」画面でジョブを取り消す必要があります。

ノードまたはサブシステム障害からの回復

ここでは、バッチノードやサブシステムに障害がある場合に回復する方法について説明します。システムがクラッシュした場合、5 つの Sun MBM デーモンのいずれか 1 つが異常終了した場合、またはデーモンのいずれかがハードループ内にある場合などに、バッチノードで障害が発生します。単一のサブシステムで障害が発生した場合、構成されたクラスまたはアクティビティーの数が足りないことなどが原因として考えられます。ただし、すべてのサブシステムが任意のジョブを受け付けているわけではない場合、システムクラッシュやノードの障害が発生する可能性があります。

▼ 障害から回復する

1. **Sun MBM に障害がある場合、`ebmsnap` コマンドを実行し、スナップショットファイルをご購入先に送信します。**

   詳細は、142 ページの「`ebmsnap` - システムのスナップショットの作成」を参照してください。「BAM Problem Determination」メニューからスナップショットを生成することもできます。

   注 – サブシステムが Sun MTP 領域に接続されている場合、`ebmsnap` は、自動的に `kixsnap` ユーティリティーを実行して領域情報を取得します。

   ---

2. **サブシステムが Sun MTP 領域に関連付けられている場合、次の操作を行います。**

   a. **領域が停止していることを確認します。**

   b. **`kixverify` を使用して VSAM ファイルの完全性を検証します。**

   c. **領域を再起動します。**

   クラッシュ後の領域の起動方法については、『Sun Mainframe Transaction Processing ソフトウェア 障害追跡とチューニング』を参照してください。

3. **バッチノードを再起動します。**

   画面に表示された指示に従います。

4. **ジョブの状態を確認します。**

   クラッシュ発生時に、実行中のジョブやキューで待機中のジョブがあった場合、これらのジョブはノードを再起動すると hold 状態になります。

5. **次のいずれかの方法でジョブを再開します。**

   - UNIX プロンプトで、`runjcl` コマンドを実行してジョブを実行します。詳細は、191 ページの「`runjcl` - ジョブクラスの実行」を参照してください。
   - Sun MBM メインメニューで、「Queued Jobs」ボタンをクリックし、対象のジョブを選択して「Run Job」ボタンをクリックします。

第3章

Job Editor の使用方法

この章の内容は、次のとおりです。

注 – Job Editor は、MVS JCL ジョブおよび手続きのインポートをサポートします。ジョブの実行時に、ジョブと手続きは MVS 動作で実行されます。たとえば、ステップの実行中にファイルが割り当てられ、ステップ実行の終了時に解放されます。

Job Editor とは

Job Editor は、Sun MBM ジョブの作成と管理を行うためのツールです。グラフィカルな環境から、作業を論理的なまとまりとして組み立てることができます。このインタフェースにより、必要なリソースを処理するアプリケーションプログラムのシーケンスを定義できます。各アプリケーションプログラムと必要なリソースはグループ化され、ステップとして表されます。

- 一連のステップをグループ化してジョブを構成できます。
- 前のステップが完了しているか失敗しているか、前のいずれかのステップで設定されている論理的な条件に一致しているかどうかなど、異なる基準に基づいてステップを実行できます。

- 一連のステップをグループ化して手続きを形成し、1 つ以上のジョブステップによってあとから呼び出すことができます。

Job Editor では、1 つ以上のプロジェクトを定義できます。プロジェクトとは、ジョブと手続きをグループ化したものです。たとえば、本番のジョブ用とテストジョブ用の 2 つのプロジェクトを作成できます。

Job Editor を使用するには、次のソフトウェアとハードウェアが必要です。

- Java™ virtual machine (JVM™)[1] バージョン 1.4 以上をシステムにインストールする必要があります。バージョンが 1.4.0 より古い場合は、エラーメッセージが表示されます。
- ノードがインストールされているシステムに、グラフィカルな端末を直接接続する必要があります。

ジョブ作成者は Job Editor を使用してジョブを作成または編集できるほか、テストシステム上にマクロジョブスクリプトを生成できます。テスト完了後、マクロジョブは本番システムにダウンロードできます。ここで、これらのジョブは適切なサブシステムにサブミットされて実行されます。

Job Editor の起動方法

この節では、Job Editor を起動する方法について説明します。

▼ Job Editor を起動する

1. **JDKROOT 環境変数を設定します。**

   この環境変数は、JVM の場所を示します。この変数を設定していない場合、Job Editor を起動することはできません。たとえば、$JDKROOT を設定ファイルで設定するには、次のように入力します。

```
JDKROOT=/usr/j2sdk1.4.0_01;export JDKROOT
```

2. **$JDKROOT を設定ファイルで設定した場合、環境を設定するように設定ファイルを取り込みます。**

1. 「Java 仮想マシン」および「JVM」とは、Java プラットフォーム用の仮想マシンを意味します。

3. **次のいずれかの方法を使用して、Job Editor を起動します。**

   - プロンプトで、`jedit` コマンドを実行します。
   - `ebmx` コマンドを入力して Sun MBM メインメニューを表示し、「Job Definitions」アイコンをクリックします。

   Job Editor のメインウィンドウが表示されます。詳細は、図 3-1 を参照してください。

   ウィンドウの左側には、すでに定義されているプロジェクトとジョブがツリー状に表示されます。これらの各要素は「ノード」と呼ばれます。

   右側には、ツリーペインで選択したアイコンの属性が表示されます。たとえば、ツリーペインでジョブステップを選択すると、右側のペインにはそのジョブステップの属性を示すフォームが表示されます。

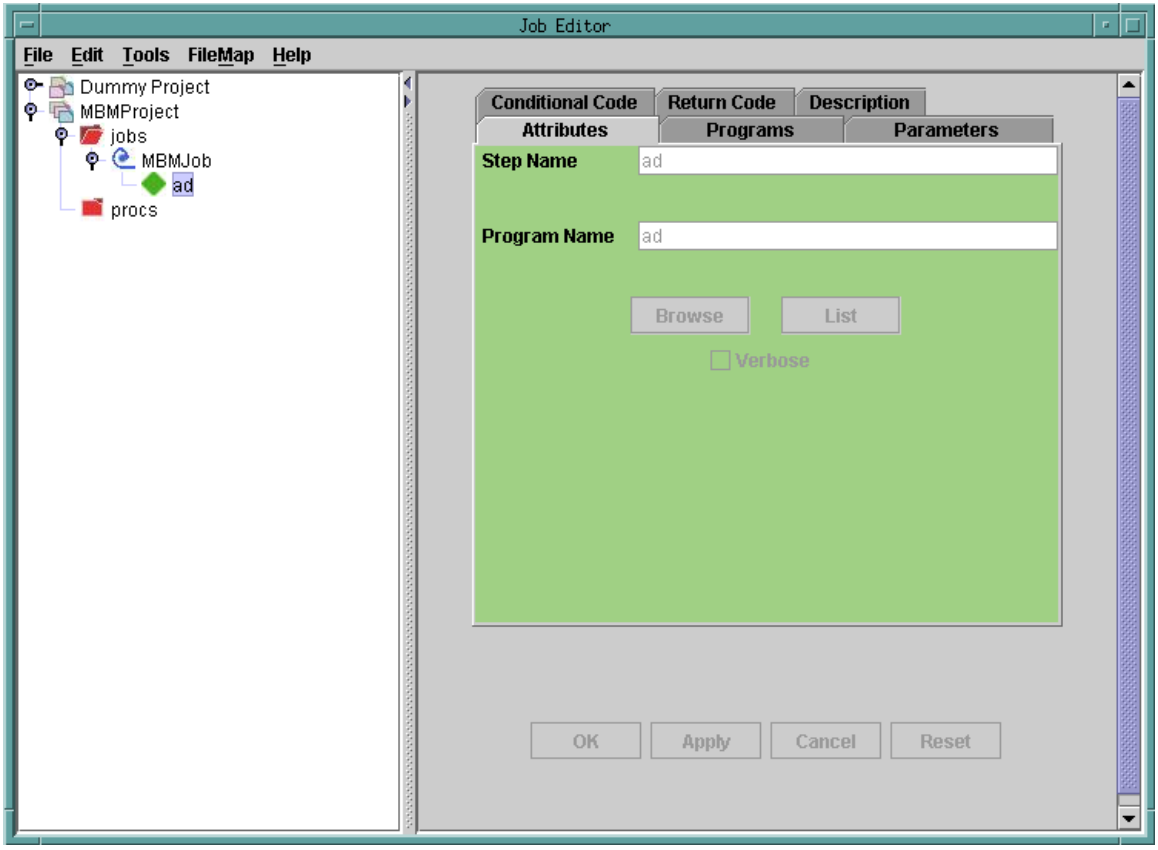

図 3-1　Job Editor ウィンドウ

メニューバーには次の項目があります。

- File: 「File」メニューには、次のオプションがあります。
  - New: 新しいプロジェクト、ジョブ、手続き、ステップ、ファイルタイプを作成できます
  - Export: ジョブおよび手続きをバッチマクロスクリプトにエクスポートします。
  - Import: MVS JCL ジョブおよび手続きをインポートします
  - Save: プロジェクトをディスクに保存します。プロジェクトを保存していない場合、Job Editor は終了時にプロジェクトを保存します
  - Quit: Job Editor を終了し、プロジェクトをディスクに保存します
- Edit: 「Edit」メニューには、次のオプションがあります。
  - Undo: カットなど、直前のアクションを元に戻します
  - Redo: ペーストなど、直前のアクションを再度実行します
  - Cut: 選択したノードをカットし、そのコピーをバッファーに格納します
  - Copy: 選択したノードをコピーします
  - Paste: バッファーの内容を選択したノードの下にペーストします
  - Clear: 変更モードで、編集フィールド内のすべてのテキストを消去します
  - Modify: 選択したノードを編集可能にします
  - Delete: 選択したノードを削除しますが、そのコピーはバッファーに格納されません
  - Copy Special: ノードおよびその下の全サブノードをコピーできます
- Tools: Job Editor のトレースデバッグをオンにするトレースオプションがあります。ご購入先で指示された場合を除き、このオプションを使用しないでください。
- FileMap: 「FileMap」メニューには、次のオプションがあります。
  - Select: プロジェクトに関連付けられている `File_Map` を選択できます
  - Import Entries: JCL ジョブまたは手続きからエントリをインポートします
  - Modify Entries: 現時点では無効のオプションです
- Help: 著作権情報を表示する「About Job Editor」オプションがあります。

プロジェクトの作成

Job Editor プロジェクトは、作業の論理的なまとまりを表す概念です。たとえば、人員刷新プロジェクトであれば、従業員データベースを毎晩更新するジョブおよび手続きがあります。

▼ プロジェクトを作成する

1. **Job Editor のメインウィンドウで、既存のプロジェクトノードを選択します。**

   初めて Job Editor を使用する場合は、「Dummy Project」を選択します。

2. **メニューで、「File」→「New」→「Project」を選択します。**

   プロジェクトの「Attributes」パネルが表示されます。

注 – 新しいプロジェクトを作成する代わりに、「Dummy Project」を選択して「File」→「Modify」を選択し、ダミープロジェクトの名前を変更すると、新しいプロジェクトとして使用できます。

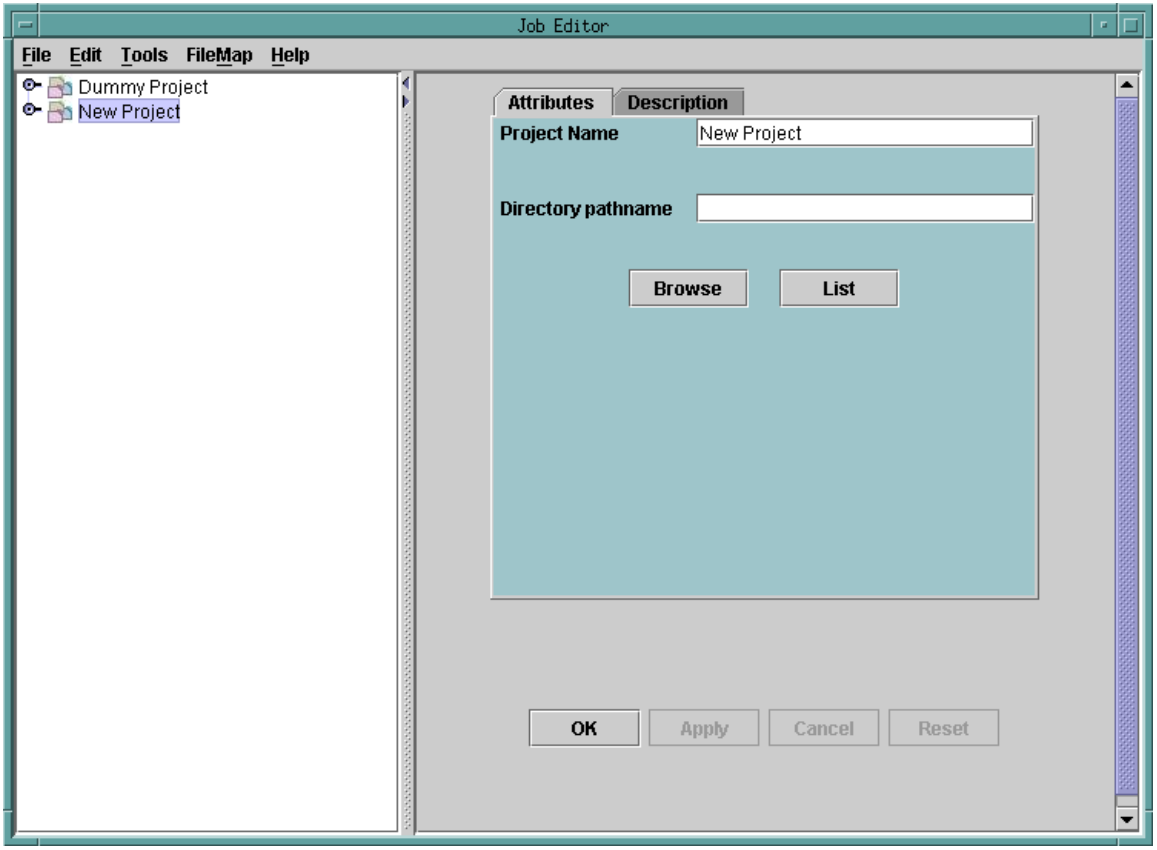

図 3-2　　プロジェクトパネル

3. 「Project Name」テキストボックスに、そのプロジェクトの一意の名前、たとえば、`PROD` と入力します。

   プロジェクトの名前がすでに定義されている場合、プロジェクトを追加しようとすると、メッセージが表示されます。

4. 次のいずれかの方法で、「Directory Pathname」テキストボックスにパス名を入力します。

   - `/home/projects` などのように、プロジェクトの親ディレクトリのフルパス名を入力します。
   - 「Browse」をクリックし、プロジェクトディレクトリを作成するディレクトリを選択します。
   - 「List」をクリックし、定義済みのプロジェクト一覧から親ディレクトリを選択します。

   親ディレクトリがすでに存在する場合、そのディレクトリに対するアクセス権およびサブディレクトリとファイルを作成する権限が必要です。

5. プロジェクトの説明を次のように追加します。

   a. 「Description」タブをクリックします。

   b. 定義するプロジェクトの説明を、「Description」テキストボックスに入力します。

6. 「OK」をクリックし、新しいプロジェクトをツリーペインに追加します。

   2 つのサブフォルダ、jobs と procs が自動的に新しいプロジェクトフォルダに追加されます。これらのフォルダには、新しいプロジェクトのジョブと手続きが含まれています。プロジェクトフォルダの横にある鍵のアイコンをクリックし、ジョブフォルダと手続きフォルダを表示します。

   エラーが発生すると、Job Editor はその理由を表示します。たとえば、指定したディレクトリに対する権限が現在のユーザーにないとエラーが発生します。

   プロジェクト作成後、Job Editor はプロジェクトディレクトリを作成します。たとえば、パス `/home/projects` を選択し、新しいプロジェクト名が PROD の場合、新しいディレクトリは `/home/projects/PROD` というパス名で作成されます。このディレクトリには、すべての内部ファイルと生成された出力ファイルが保存されます。

ジョブの作成

▼ ジョブを作成する

1. ツリーペインからプロジェクトのジョブフォルダを選択します。
2. メニューで、「File」→「New」→「Job」を選択します。

   ジョブパネルが表示されます。
3. ジョブの属性を次のように定義します。

   a. 「Attributes」パネルが表示されていない場合は、「Attributes」タブをクリックします。

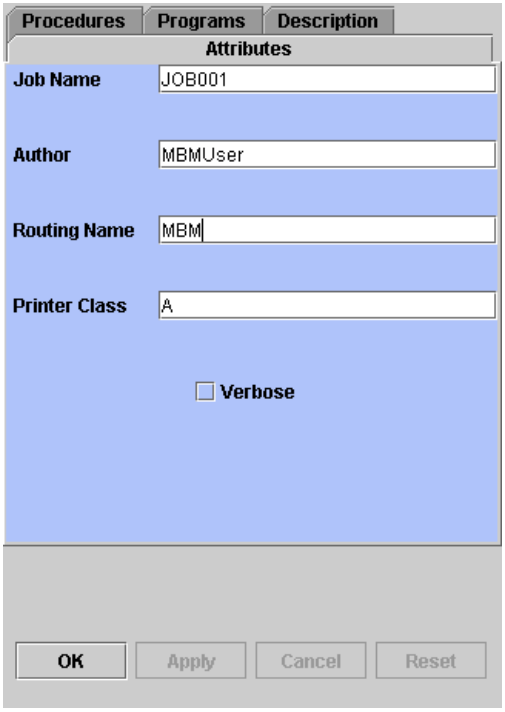

図 3-3　　ジョブの「Attributes」タブ

b. 「Job Name」テキストボックスに、プロジェクトのジョブの一意の名前、たとえば `JOB001` を入力します。

ジョブの名前がすでに定義されている場合は、この新しいジョブを追加しようとするとエラーメッセージが表示されます。

c. 「Author」テキストボックスに作成者の名前を入力します。

d. 「Routing Name」テキストボックスに、通知に使用する目的で実行時にジョブに割り当てる識別子を入力します。

これは、`BEGINJOB` マクロで `programmer` オプションを付けた処理と同じです。

e. デフォルトのジョブプリンタクラス名を「Printer Class」テキストボックスに入力します。

デフォルトクラスは A です。

f. ジョブ履歴ファイルに書き込まれたメッセージなどの情報を確認する場合は、「Verbose」チェックボックスを選択します。

これは、`BEGINJOB` マクロで `verbose` オプションを付けた処理と同じです。

4. 「Procedures」タブをクリックします。

   既存のジョブを編集している場合は、手続きディレクトリの一覧が表示されます。

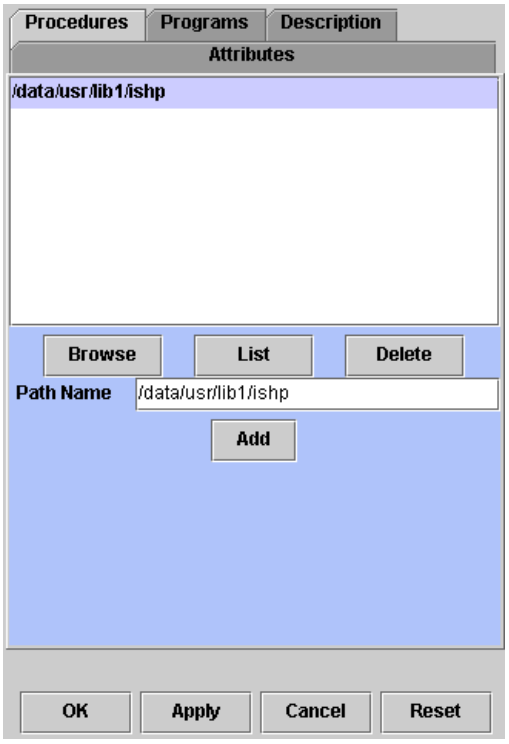

図 3-4　「Procedures」タブ

5. **次のいずれかの方法で、このジョブに関連付けられている変換された手続きを格納しているディレクトリの名前 (通常は `ishp` ディレクトリ) を指定します。**

   - 「Path Name」テキストボックスにディレクトリのパス名を入力します。
   - 「Browse」をクリックし、システム内の特定のディレクトリを検索します。
   - 「List」をクリックし、ほかのジョブにすでに定義されている手続きディレクトリから選択します。

6. **「Add」をクリックします。**

   ディレクトリが手続きディレクトリの一覧に追加されます。

注 – 一覧のエントリを削除するには、削除するディレクトリを選択し、「Delete」をクリックします。

---

7. **手続きディレクトリを追加するには、手順 5 および 6 を繰り返します。**

8. 「Programs」タブをクリックします。

   プログラムディレクトリの一覧が表示されます。

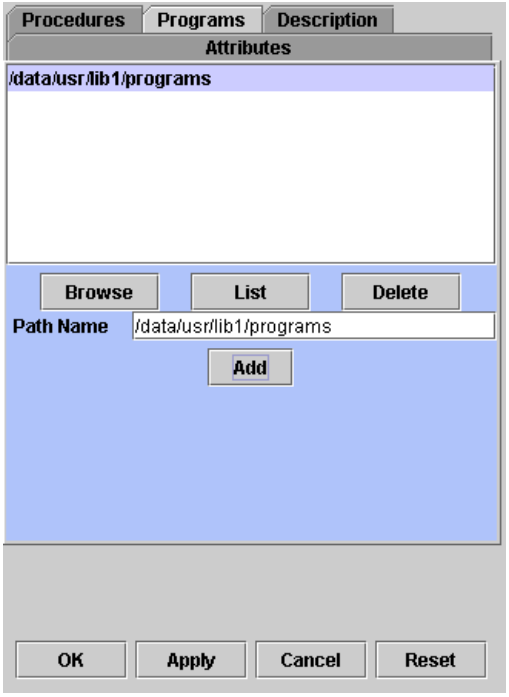

図 3-5　「Programs」タブ

9. 次のいずれかの方法を使用して、このジョブで実行されるアプリケーションプログラムのディレクトリ名を指定します。
   - 「Path Name」テキストボックスに、そのアプリケーションプログラムが保存されているディレクトリのパス名を入力します。
   - 「Browse」をクリックし、特定のディレクトリを検索します。
   - 「List」をクリックし、ほかのジョブにすでに定義されている手続きディレクトリから選択します。
10. 「Add」をクリックします。

    ディレクトリがプログラムディレクトリの一覧に追加されます。

注 – 一覧のエントリを削除するには、削除するディレクトリを選択し、「Delete」をクリックします。

---

11. プログラムディレクトリを追加するには、手順 9 および 10 を繰り返します。

12. ジョブの説明を追加するには、「Description」タブをクリックし、説明テキストを入力します。

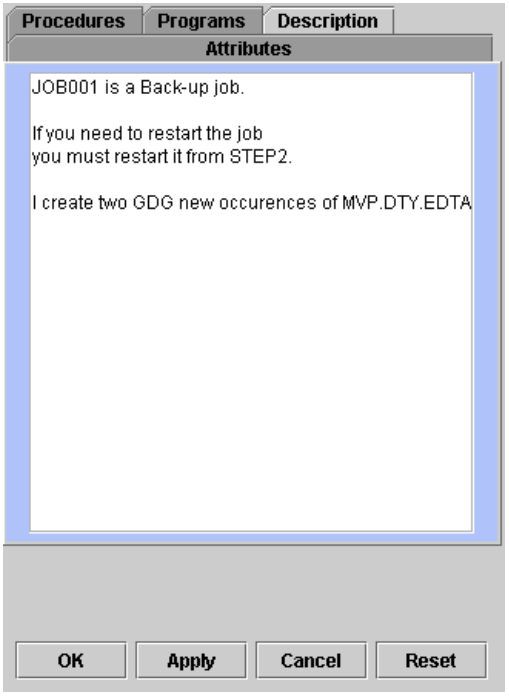

図 3-6　「Description」タブ

13. ジョブの定義が完了したら、「OK」をクリックし、新しいジョブをツリーペインに追加します。

このジョブは、*project*/jobs サブディレクトリにも追加されます。

エラーが発生すると、Job Editor はメッセージを表示します。たとえば、ジョブ名がすでに存在する場合、またはプロジェクトディレクトリにファイルを作成する権限が現在のユーザーにない場合に、エラーが表示されます。

手続きの作成

▼ 手続きを作成する

1. ツリーペインで、新しい手続きを追加するプロジェクトの procs フォルダを選択します。
2. メニューで、「File」→「New」→「Procedure」を選択します。
   手続きパネルが表示されます。

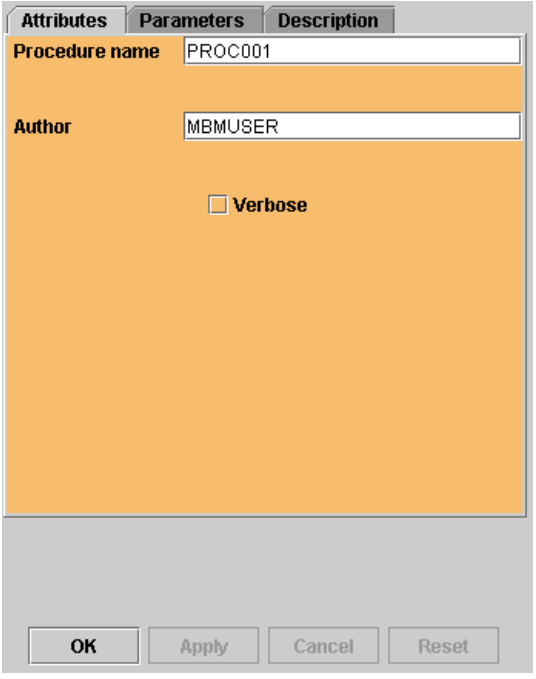

図 3-7　　手続きの「Attributes」タブ

3. 手続きの属性を次のように定義します。
   a. 「Attributes」パネルが表示されていない場合は、「Attributes」タブをクリックします。

b. 「Procedure Name」テキストボックスに、手続きの一意の名前、たとえば PROC001 を入力します。

手続きの名前が同じプロジェクト内ですでに定義されている場合、この手続きを追加しようとするとエラーメッセージが表示されます。

c. 「Author」テキストボックスに作成者の名前を入力します。

d. ジョブ履歴ファイルに書き込まれたメッセージなどの情報を確認する場合は、「Verbose」チェックボックスを選択します。

これは、BEGINPROC マクロで verbose オプションを付けた処理と同じです。

4. 新しいパラメータを追加する方法は次のとおりです。

a. 「Parameters」タブをクリックします。

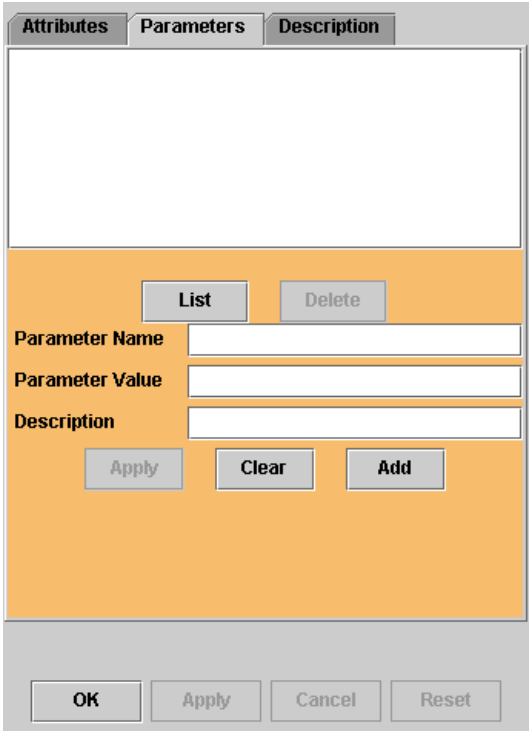

図 3-8　　手続きの「Parameters」タブ

b. 「Parameter Name」テキストボックスに一意の変数名を、また「Parameter Value」テキストボックスにその値を入力します。

c. 「Description」テキストボックスにパラメータの説明を入力します (省略可能)。

d. 「Add」をクリックします。

新しいパラメータが一覧に表示されます。

5. 追加済みのパラメータを変更する手順は次のとおりです。

a. 「Parameters」パネルでパラメータを選択します。

「Parameter Name」、「Parameter Value」、および「Description」テキストボックスに現在の値が表示されます。

b. 値のいずれかを編集し、「Apply」をクリックします。

更新されたパラメータが一覧に表示されます。

注 – エントリを削除するには、削除するパラメータを選択して「Delete」をクリックします。

---

6. プロジェクトですでに定義されているパラメータの 1 つを選択する方法は次のとおりです。

a. 「List」をクリックします。

プロジェクトに定義されているパラメータの一覧が表示されます。

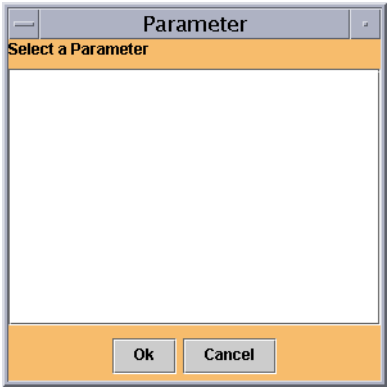

図 3-9　手続きパラメータ一覧のダイアログボックス

b. パラメータを選択します。

c. 「OK」をクリックします。

「Parameter Name」、「Parameter Value」、および「Description」テキストボックスに現在の値が表示されます。

d. 必要に応じて値を編集し、「Add」をクリックします。

パラメータがパラメータの一覧に追加されます。

7. 「Description」タブをクリックします。
8. 「Description」テキストボックスに、作成する手続きの説明を入力します。
9. 「OK」をクリックします。

   新しい手続きが、ツリーペインと *project*/procs サブディレクトリに追加されます。

   エラーが発生すると、Job Editor はメッセージを表示します。たとえば、手続き名がすでに存在する場合、またはプロジェクトディレクトリにファイルを作成する権限が現在のユーザーにない場合に、エラーが表示されます。

ステップの作成

Job Editor を使用すると、次の 3 種類のステップを作成できます。

- 手続き: 手続きを実行します。
- プログラム: アプリケーションプログラムを実行します。
- ユーティリティー: Sun MBM またはユーザーユーティリティーを実行します。Sun MBM とユーザーユーティリティーの詳細は、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

この節では、各種のステップを作成する手順を説明しています。

▼ 手続きを実行するステップを作成する

1. ツリーペインで、新しいステップを追加するジョブまたは手続きを選択します。

   特定の場所にステップを挿入するには、既存のステップを選択します。新しいステップはそのあとに挿入されます。

2. メニューで、「File」→「New」→「Step」→「Procedure」を選択します。

   ステップパネルが表示されます (図 3-10)。

3. ステップの属性を指定する方法は次のとおりです。

a. 「Attributes」パネルが表示されていない場合は、「Attributes」タブをクリックします。

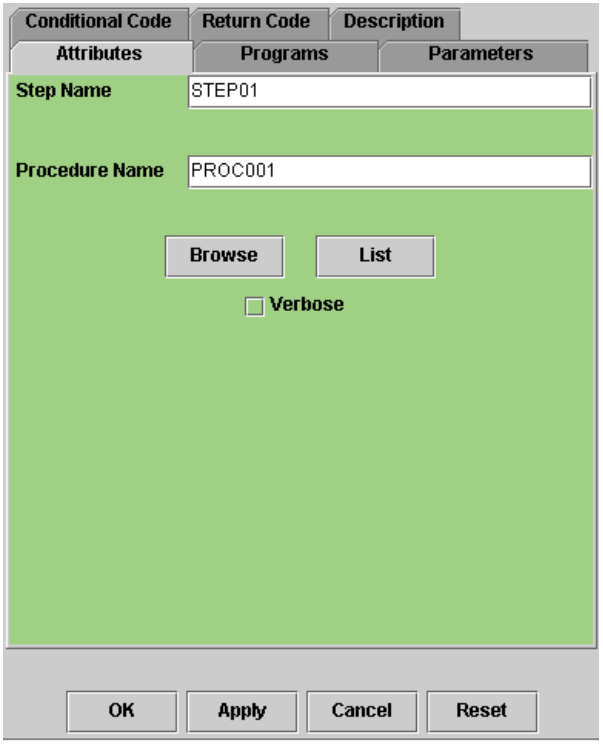

図 3-10　手続きのステップパネル

b. 「Step Name」テキストボックスに、ステップの一意の名前、たとえば `STEP01` を入力します。

入力したステップ名が同じジョブまたは手続き内ですでに定義されている場合、エラーメッセージが表示されます。

c. 次のいずれかの方法を使用して、手続きを指定します。

- 「Procedure Name」テキストボックスに手続き名を入力します。
- 「Browse」をクリックし、ファイルシステムに定義されている手続きを検索します。
- 「List」をクリックし、プロジェクトにすでに定義されている手続きのいずれかを選択します。

d. ジョブ履歴ファイルに書き込まれたメッセージなどの情報を確認する場合は、「Verbose」チェックボックスを選択します。

これは、`EXECPROC` マクロで `verbose` オプションを付けた処理と同じです。

4. 新しいパラメータをステップに追加する方法は次のとおりです。

   a. 「Parameters」タブをクリックします。

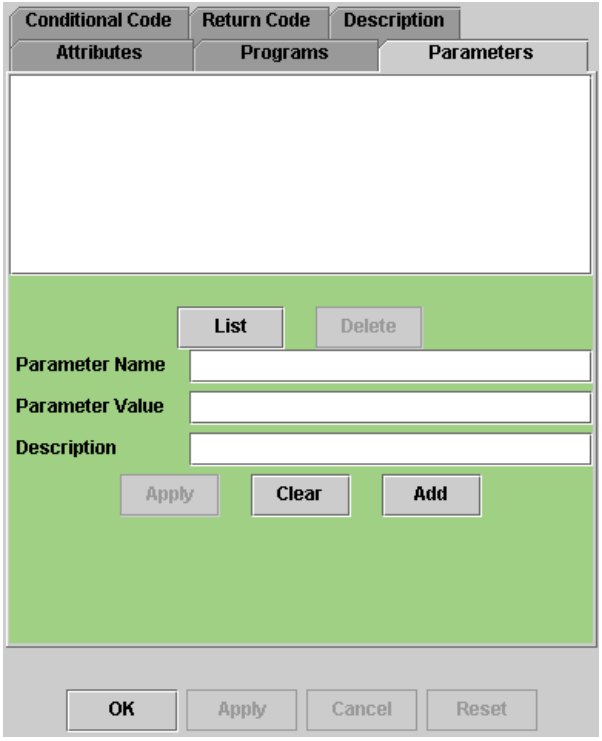

図 3-11　手続きステップ用のパラメータの定義

   b. 「Parameter Name」テキストボックスに一意の名前を、また「Parameter Value」テキストボックスにその値を入力します。

   c. 「Description」テキストボックスに説明を入力します (省略可能)。

   d. 「Add」をクリックします。

      新しいパラメータが一覧に表示されます。

注 – エントリを削除するには、削除するパラメータを一覧から選択して「Delete」をクリックします。

---

5. プロジェクトですでに定義されているパラメータの 1 つを選択する方法は次のとおりです。

   a. 「List」をクリックします。

      プロジェクトに定義されているパラメータの一覧が表示されます。

   b. パラメータを選択します。

   c. 「OK」をクリックします。

      「Parameter Name」、「Parameter Value」、および「Description」テキストボックスに、選択したパラメータが表示されます。

   d. 必要に応じて値を編集し、「Add」をクリックします。

      パラメータが一覧に追加されます。

6. パラメータを変更する方法は次のとおりです。

   a. 一覧でパラメータを選択します。

      「Parameter Name」、「Parameter Value」、および「Description」テキストボックスに現在の値が表示されます。

   b. フィールドのいずれかを編集し、「Apply」をクリックします。

      一覧に、更新後のパラメータが表示されます。

7. 条件およびリターンコードをステップに追加する方法については、69 ページの「条件コードとリターンコードの設定」を参照してください。

8. 説明を追加するには、「Description」タブをクリックし、テキストボックスにステップの説明を入力します。

9. ステップの定義が完了したら、「OK」をクリックします。

   ステップがツリーペインのノードに追加されます。

▼ プログラムを実行するステップを作成する

1. **ツリーペインで、新しいステップを追加するジョブまたは手続きを選択します。**

   特定の場所にステップを挿入するには、既存のステップを選択します。新しいステップはそのあとに挿入されます。

2. **メニューで、「File」→「New」→「Step」→「Program」を選択します。**

   ステップパネルが表示されます (図 3-12)。

図 3-12　プログラムのステップパネル

3. **ステップの属性を指定する方法は次のとおりです。**

   a. **「Attributes」パネルが表示されていない場合は、「Attributes」タブをクリックします。**

   b. **「Step Name」テキストボックスに、ステップの一意の名前、たとえば** `STEP01` **を入力します。**

      入力したステップ名が同じジョブまたは手続き内ですでに定義されている場合、エラーメッセージが表示されます。

c. **次のいずれかの方法を使用して、プログラムを指定します。**

- 「Program Name」テキストボックスにプログラム名を入力します。
- 「Browse」をクリックし、ファイルシステムに定義されているプログラムを検索します。
- 「List」をクリックし、別のジョブにすでに定義されているプログラムから選択します。

d. **ジョブ履歴ファイルに書き込まれたメッセージなどの情報を確認する場合は、「Verbose」チェックボックスを選択します。**

これは、`EXECPGM` マクロで `verbose` オプションを付けた処理と同じです。

4. **「Programs」タブをクリックします。**

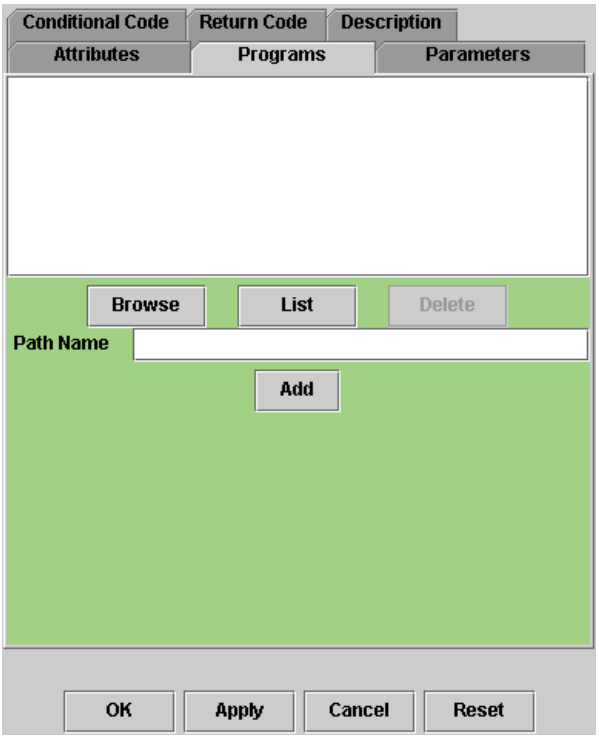

図 3-13　プログラムステップ用のプログラムディレクトリの定義

5. **次のいずれかの方法を使用して、プログラムディレクトリを指定します。**

- 「Path Name」テキストボックスに、そのステップ用のプログラムが保存されているディレクトリのパス名を入力します。
- 「Browse」をクリックし、システム内の特定のディレクトリを検索します。
- 「List」をクリックし、プロジェクト用にすでに定義されているプログラムディレクトリから選択します。

6. 「Add」をクリックします。
   ディレクトリがプログラムディレクトリの一覧に追加されます。

注 – 一覧のエントリを削除するには、削除するディレクトリを選択し、「Delete」をクリックします。

---

7. プログラムディレクトリを追加するには、手順 5 および 6 を繰り返します。
8. パラメータを追加する方法は次のとおりです。
   a. 「Parameters」タブをクリックします。
   b. 「Parameter」テキストボックスに入力パラメータを入力します。
   c. 「Description」テキストボックスに説明を入力します。
9. 条件およびリターンコードをステップに追加する方法については、69 ページの「条件コードとリターンコードの設定」を参照してください。
10. ステップの説明を追加するには、「Description」タブをクリックし、テキストボックスに説明を入力します。
11. ステップの定義が完了したら、「OK」をクリックします。
    ステップがツリーペインのノードに追加されます。

▼ ユーティリティーを実行するステップを作成する

1. ツリーペインで、新しいステップを追加するジョブまたは手続きを選択します。
   特定の場所にステップを挿入するには、既存のステップを選択します。新しいステップはそのあとに挿入されます。
2. メニューで、「File」→「New」→「Step」→「Utility」を選択します。
   ステップパネルが表示されます。

図 3-14　ユーティリティーのステップパネル

3. ステップの属性を指定する方法は次のとおりです。

   a. 「Attributes」パネルが表示されていない場合は、「Attributes」タブをクリックします。

   b. 「Step Name」テキストボックスに、一意のステップ名を入力します。

   c. 次のいずれかの方法でユーティリティー名を入力します。

      - このステップに割り当てるユーティリティー名を入力します。
      - 「Browse」をクリックし、ファイルシステムに定義されているユーティリティーを検索します。
      - 「List」をクリックし、プロジェクト用にすでに定義されているユーティリティー、または Sun MBM が提供するユーティリティーを選択します。

4. パラメータを定義する方法は次のとおりです。

   a. 「Parameters」タブをクリックします。

   b. 「Parameter」テキストボックスに入力パラメータを入力します。

   c. 「Description」テキストボックスに説明を入力します。

5. 条件およびリターンコードをステップに追加する方法については、69 ページの「条件コードとリターンコードの設定」を参照してください。
6. ステップの説明を追加するには、「Description」タブをクリックし、テキストボックスに説明を入力します。
7. ステップの定義が完了したら、「OK」をクリックします。

   ステップがツリーペインのノードに追加されます。

条件コードとリターンコードの設定

ジョブステップまたは手続きステップを作成する際、条件コードとリターンコードを指定できます。EBMSYSCMD マクロと条件コードの詳細は、『Sun Mainframe Batch Manager ソフトウェア リファレンスマニュアル』を参照してください。

▼ 条件コードを設定する

1. Job Editor の左パネルでステップを選択します。
2. メニューで、「Edit」→「Modify」を選択します。
3. ステップパネルで、「Conditional Code」タブをクリックします。
4. ドロップダウンリストから、次の条件演算子のいずれかを選択します。

   EQ、NE、GT、LT、GE、LE
5. 条件コードテキストボックスに、数値を入力します。
6. 「Then」フィールドのドロップダウンリストから、次の条件アクションのいずれかを選択します。

   BYPASS、CONTINUE、GOTO

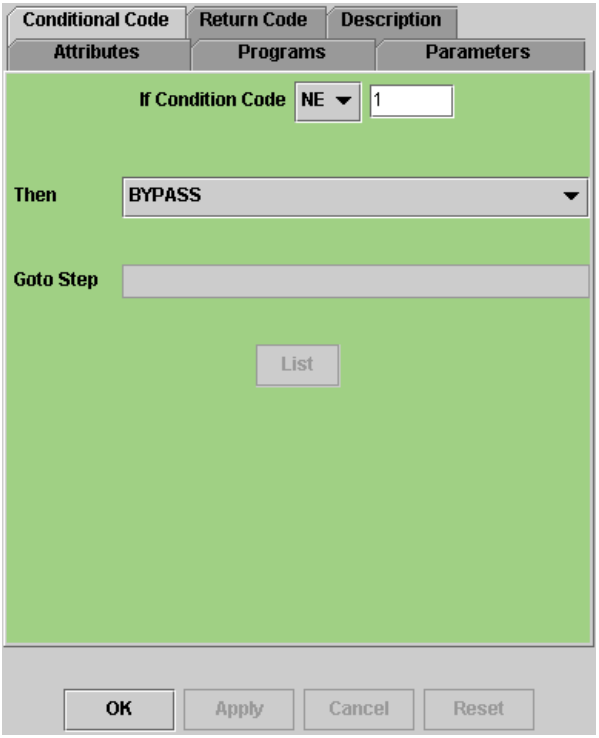

図 3-15　「Conditional Code」タブ

7. 「GOTO」を指定した場合は、ステップ名を選択する必要があります。

   a. 「List」をクリックします。

      ダイアログボックスに、このステップに属するジョブに定義されているすべてのステップが表示されます。

   b. ステップまたは「End of Job」を選択します。

   c. 「OK」をクリックします。

      選択したステップ名が、「Goto Step」テキストボックスに表示されます。このステップ名を編集できます。

8. ステップの変更が完了したら、「OK」をクリックします。

▼ リターンコードを設定する

1. **ステップが編集モードであることを確認します。**
   メニューで、「Edit」→「Modify」を選択します。
2. **「Return Code」タブをクリックします。**

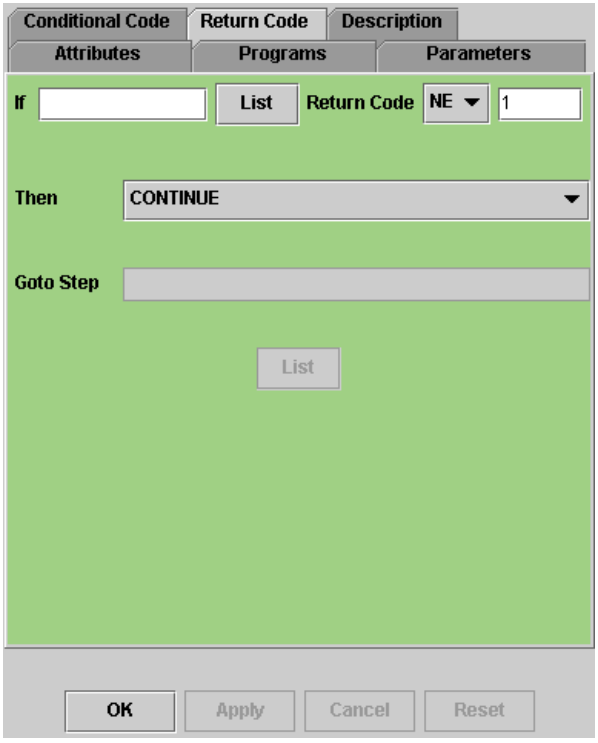

図 3-16　「Return Code」タブ

3. **次のいずれかの方法を使用して、リターンコードをチェックするステップを選択します。**
   - 「If」テキストボックスにステップ名を入力します。
   - 「List」ボタンをクリックしてステップ名を選択するか、「Any」をクリックして任意のステップにチェックを適用します。

     このリストには、ジョブに定義されているステップのうち、選択したステップより上位のものがすべて表示されます。
4. **ドロップダウンリストから、次の条件演算子のいずれかを選択します。**
   EQ、NE、GT、LT、GE、LE
5. **「Return Code」テキストボックスに、数値を入力します。**

6. ドロップダウンリストから、次の条件アクションのいずれかを選択します。

   BYPASS、CONTINUE、GOTO

7. 「GOTO」を指定した場合は、ステップ名を選択する必要があります。

   a. 「List」をクリックします。

      ダイアログボックスに、このステップに属するジョブに定義されているすべてのステップが表示されます。

   b. ステップまたは「End of Job」を選択します。

   c. 「OK」をクリックします。

      選択したステップ名が、「Goto Step」テキストボックスに表示されます。このステップ名を編集できます。

8. ステップの変更が完了したら、「OK」をクリックします。

手続きの上書き

Job Editor は、`COND.`*stepname* 手続きの上書きをサポートしています。いずれも、キーワード `ONCONDCODE` および `ONRETCODE` でパラメータとして処理されます。

たとえば、次の JCL を含むジョブまたは手続きをインポートするとします。

```
//GO EXEC PCOND,UNIT=VTAPE,
//  PARM1=PPPP1,COND.STEP1=(0,EQ)
```

手続きステップパネルの「Parameters」タブには、次の値があります。

Parameter Name: `ONRETCODE`

Parameter Value: `MAXRC EQ 0 BYPASS scope='STEP' poverride='y' stepname='STEP1'`

注 – 新しい手続きステップを作成している場合は、JCL 構文ではなく Sun MBM マクロ構文でパラメータ値を入力してください。

完全な JCL 文の変換されたマクロスクリプトは次のとおりです。

```
ONRETCODE MAXRC EQ 0 BYPASS scope='STEP' poverride='y' stepname='STEP1'
EXECPROC procname='PCOND' stepname='GO' parms='UNIT=VTAPE'
```

ファイル定義の作成

ファイル定義を作成する際に、ファイルをジョブまたは手続きステップに関連付けます。Job Editor では次のファイルタイプをサポートします。

- Standard: ファイルシステムのファイル
- VSAM: VSAM ファイル
- GDG: Sun MBM 世代別データグループファイル
- Input: 入力ストリームファイル (SYSIN を含む)
- Print: プリンタに割り当てられたファイル (SYSOUT)
- Concatenated: Sun MBM 連結ファイル (標準の順編成ファイル、Sun MBM GDG、入力ファイルを含む)
- Alias: 以前の ASSGNDD マクロ文で割り当てられた DDNAME に現在の DDNAME を割り当てます。

▼ ファイル定義を作成する

1. **Job Editor のメインウィンドウのツリーペインで、新しいファイルを追加するジョブ、または手続きのステップを選択します。**

   既存のファイルの間にファイルを挿入するには、新しいファイルの前に来ることになるファイルを選択します。

2. **メニューで、「File」→「New」→「File Type」を選択します。**

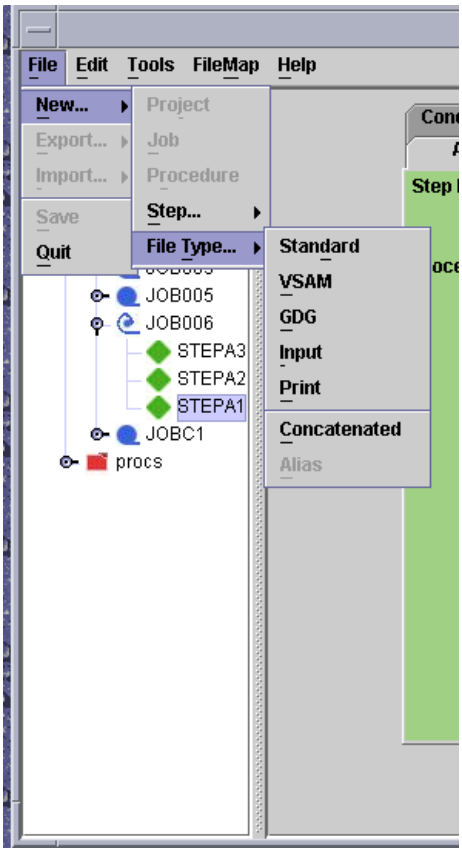

図 3-17　　ファイルタイプの選択

3. **ファイルタイプを選択します。**

   ファイルの「Attributes」パネルが表示されます。

4. **選択したファイルタイプの属性を指定します。**

   ファイルタイプごとに、異なる属性を指定する必要があります。次の節を参照してください。

   - 75 ページの「標準ファイルを定義する」
   - 77 ページの「VSAM ファイルを定義する」
   - 79 ページの「GDG ファイルを定義する」
   - 82 ページの「入力ファイルを定義する」
   - 83 ページの「印刷ファイルを定義する」
   - 87 ページの「連結ファイルの定義」
   - 89 ページの「別名を定義する」

▼ 標準ファイルを定義する

1. 基本的なファイル情報を定義します。

   a. 「Attributes」タブをクリックします。

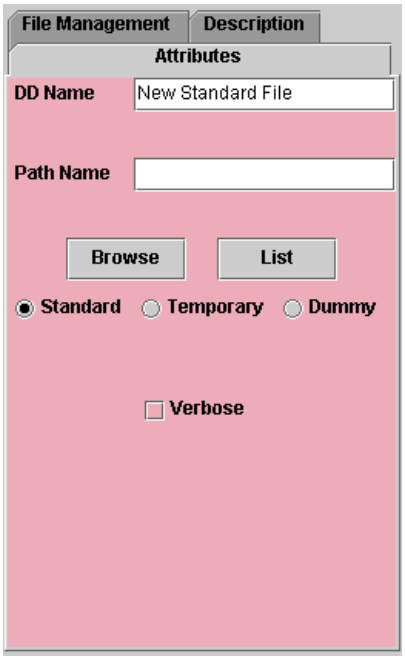

図 3-18　標準ファイルの「Attributes」タブ

   b. 「DD Name」テキストボックスに、ファイルの一意の名前、たとえば `DD1` を入力します。

   入力したファイル名が同じステップ内ですでに定義されている場合、このファイルを追加しようとするとエラーが表示されます。

   c. 割り当てるファイルのパス名を次のいずれかの方法で指定します。

   - 「Path Name」テキストボックスに名前を入力します。`$SEQFILES` などの環境変数を使用できます。
   - 「Browse」をクリックし、システム内のファイルを検索します。
   - 「List」をクリックし、ほかのジョブ用にすでに割り当てられているファイルを選択します。

   選択したパス名が、「Path Name」テキストボックスに表示されます。

d. デフォルトの「Standard」ファイルタイプのままにするか、一時またはダミーファイルタイプに変更します。

Job Editor は、ファイルの現在の設定を保存します。

e. ジョブ履歴ファイルに書き込まれたメッセージなどの情報を確認する場合は、「Verbose」チェックボックスを選択します。

これは、ASSGNDD マクロで verbose オプションを付けた処理と同じです。

2. ファイル管理情報を指定します。

a. 「File Management」タブをクリックします。

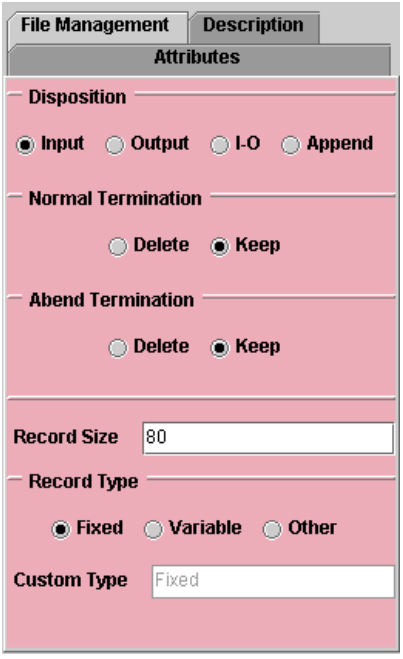

図 3-19　標準ファイルの「File Management」タブ

b. 次の「Disposition」オプションのいずれかを選択します。

- Input
- Output
- I-O (Input/Output)
- Append

c. 次の「Normal Termination」オプションのいずれかを選択します。

- Delete
- Keep

d. 次の「Abend Termination」オプションのいずれかを選択します。

- Delete
- Keep

Normal、Abend、Disposition パラメータの詳細および ASSGNDD マクロについては、『Sun Mainframe Batch Manager ソフトウェア リファレンスマニュアル』を参照してください。

e. 「レコードサイズ」の数値フィールドに値を入力します。

f. 次のレコードタイプから選択します。

- Fixed (固定)
- Variable (可変)
- Other (その他)
- 独自のレコードタイプを指定する場合は、「Custom Type」テキストボックスに情報を入力します。このフィールドはオプションです。

3. 「Description」タブをクリックして、定義するファイルの説明をテキストボックスに入力します。

4. ファイルの定義が完了したら、「OK」をクリックします。

新しいファイル定義が、ツリーペインと選択したステップのサブフォルダに追加されます。

エラーが発生すると、Job Editor はメッセージを表示します。たとえば、必須フィールドのいずれかに情報を入力していない場合、エラーが表示されます。

▼ VSAM ファイルを定義する

1. 基本的なファイル情報を定義します。

a. 「Attributes」タブをクリックします。

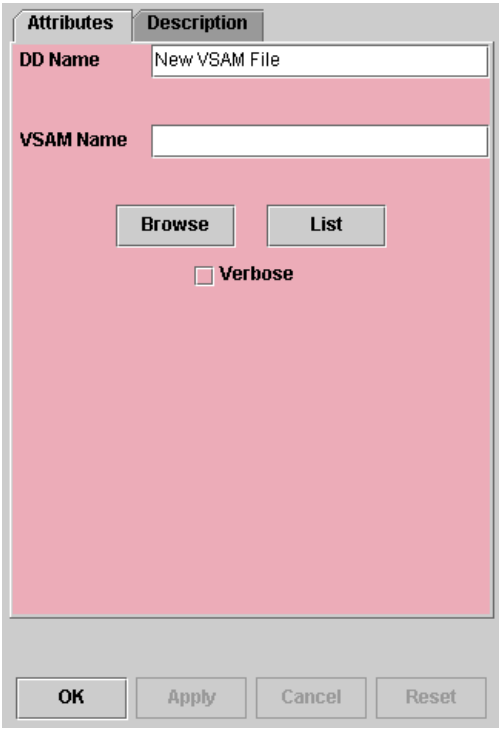

図 3-20　VSAM ファイルの「Attributes」タブ

b. 「DD Name」テキストボックスに、JCL で使用するデータセットの一意の名前、たとえば `DD1` を入力します。

新しいジョブまたは手続き用にこのファイルを定義している場合、この値は、COBOL プログラムでファイルの `SELECT` 文の `ASSIGN` 句にある名前です。

入力したファイル名が同じステップ内ですでに定義されている場合、このファイルを追加しようとするとエラーが表示されます。

c. 「VSAM Name」テキストボックスに、Sun MTP ファイル制御テーブル (FCT) および VSAM カタログに定義されている Sun MTP データセットの名前を入力します。

「List」ボタンを使用すると、以前に定義した VSAM ファイルの一覧を表示できます。「Browse」ボタンを使用しないでください。

d. ジョブ履歴ファイルに書き込まれたメッセージなどの情報を確認する場合は、「Verbose」チェックボックスを選択します。

これは、`ASSGNDD` マクロで `verbose` オプションを付けた処理と同じです。

VSAM ファイルの詳細は、『Sun Mainframe Transaction Processing ソフトウェア管理者ガイド』を参照してください。

2. 「Description」タブをクリックして、定義するファイルの説明をテキストボックスに入力します。

3. ファイルの定義が完了したら、「OK」をクリックします。

   新しいファイル定義が、ツリーペインと選択したステップのサブフォルダに追加されます。

   エラーが発生すると、Job Editor はメッセージを表示します。たとえば、必須フィールドのいずれかに情報を入力していない場合、エラーが表示されます。

▼ GDG ファイルを定義する

1. 基本的なファイル情報を定義します。

   a. 「Attributes」タブをクリックします。

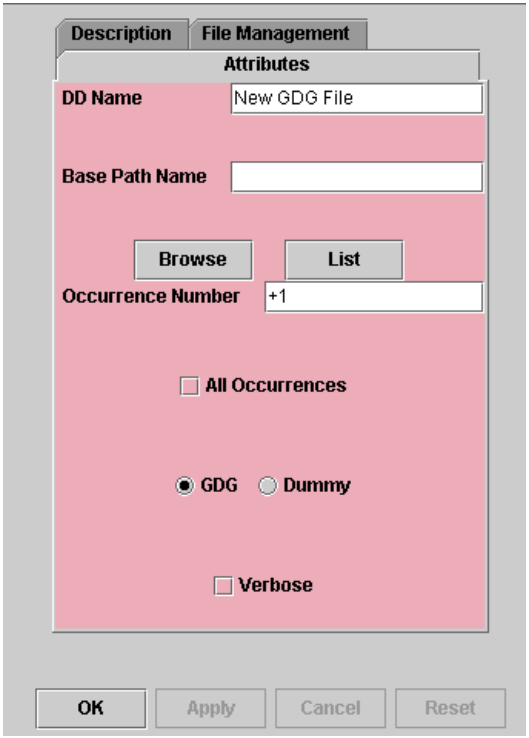

図 3-21　GDG ファイルの「Attributes」タブ

   b. 「DD Name」テキストボックスに、ファイルのベース名、たとえば `DD1` を入力します。

c. 「Base Path Name」テキストボックスに、割り当てる GDG のベースパス名を入力します。これは、GDG のパスおよびファイル名です (_00、_01 など、下線と数字の接尾辞は含まれない)。

次のいずれかの方法を使用します。

- パス名を入力します。
- 「Browse」をクリックし、パス名を検索します。
- 「List」をクリックし、プロジェクト用にすでに定義されているパス名を選択します。

d. GDG オカレンス番号を数値フィールドに入力します。

- たとえば、-3、-2、-1、0、+1、+2 と入力します。
- 「All Occurrences」チェックボックスを選択して、すべての GDG オカレンスを割り当てます。

e. デフォルトの「GDG」ファイルタイプのままにするか、「Dummy」を選択してダミーファイルタイプに変更します。

Job Editor は、ファイルの現在の設定を保存します。

GDG ファイルタイプとオカレンス番号の詳細は、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

f. ジョブ履歴ファイルに書き込まれたメッセージなどの情報を確認する場合は、「Verbose」チェックボックスを選択します。

これは、ASSGNDD マクロで verbose オプションを付けた処理と同じです。

2. ファイル管理情報を指定します。

a. 「File Management」タブをクリックします。

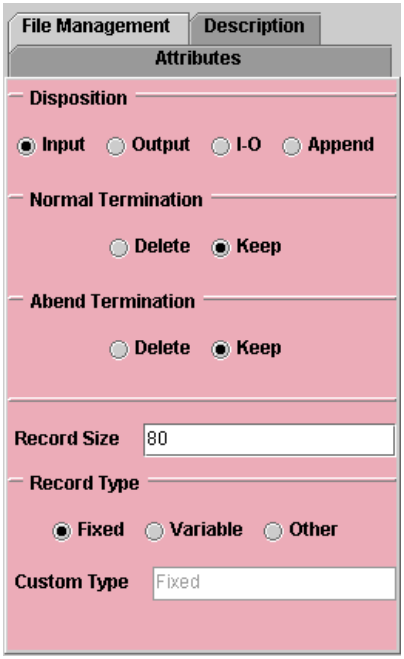

図 3-22　GDG ファイルの「File Management」タブ

b. 次の「Disposition」オプションのいずれかを選択します。

- Input
- Output
- I-O (Input/Output)
- Append

c. 次の「Normal Termination」オプションのいずれかを選択します。

- Delete
- Keep

d. 次の「Abend Termination」オプションのいずれかを選択します。

- Delete
- Keep

e. 「レコードサイズ」の数値フィールドに値を入力します。

f. 次のレコードタイプから選択します。

- Fixed (固定)
- Variable (可変)
- Other (その他)
- 独自のレコードタイプを指定する場合は、「Custom Type」テキストボックスに情報を入力します。このフィールドはオプションです。

3. 「Description」タブをクリックして、定義するファイルの説明をテキストボックスに入力します。

4. ファイルの定義が完了したら、「OK」をクリックします。

   新しいファイル定義が、ツリーペインと選択したステップのサブフォルダに追加されます。

   エラーが発生すると、Job Editor はメッセージを表示します。たとえば、必須フィールドのいずれかに情報を入力していない場合、エラーが表示されます。

▼ 入力ファイルを定義する

1. 基本的なファイル情報を定義します。

   a. 「Attributes」タブをクリックします。

   b. 「DD Name」テキストボックスに、ファイルの一意の名前、たとえば `DD1` を入力します。

   c. 特定のステップに割り当てる 1 行または複数行の入力データを「Input Data text」ボックスに入力します。

   d. ジョブ履歴ファイルに書き込まれたメッセージなどの情報を確認する場合は、「Verbose」チェックボックスを選択します。

      これは、`ASSGNDD` マクロで `verbose` オプションを付けた処理と同じです。

2. 「Description」タブをクリックして、定義するファイルの説明をテキストボックスに入力します。

3. ファイルの定義が完了したら、「OK」をクリックします。

   新しいファイル定義が、ツリーペインと選択したステップのサブフォルダに追加されます。

   エラーが発生すると、Job Editor はその理由を表示します。たとえば、必須フィールドのいずれかに情報を入力していない場合、エラーが発行されます。

▼ 印刷ファイルを定義する

1. 基本的なファイル情報を定義します。

   a. 「Attributes」タブをクリックします。

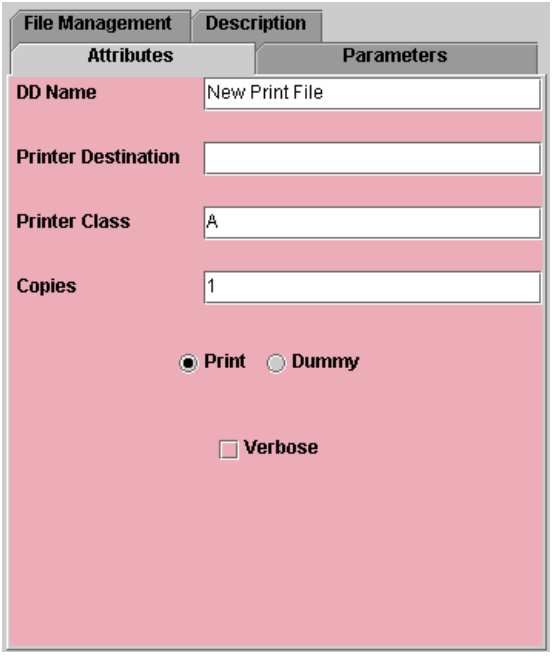

図 3-23　印刷ファイルの「Attributes」タブ

   b. 「DD Name」テキストボックスに、ファイルの一意の名前、たとえば `DD1` を入力します。

   c. プリンタ出力先名を「Printer Destination」テキストボックスに入力します。

   d. クラスタイプを「Printer Class」テキストボックスに入力します。

   e. 印刷する部数を「Copies」テキストボックスに入力します。

   f. デフォルトの「Print」ファイルタイプのままにするか、ダミーファイルタイプに変更します。

      Job Editor は、ファイルの現在の設定を保存します。

   g. ジョブ履歴ファイルに書き込まれたメッセージなどの情報を確認する場合は、「Verbose」チェックボックスを選択します。

      これは、`ASSGNDD` マクロで `verbose` オプションを付けた処理と同じです。

2. ファイル管理情報を指定します。

a. 「File Management」タブをクリックします。

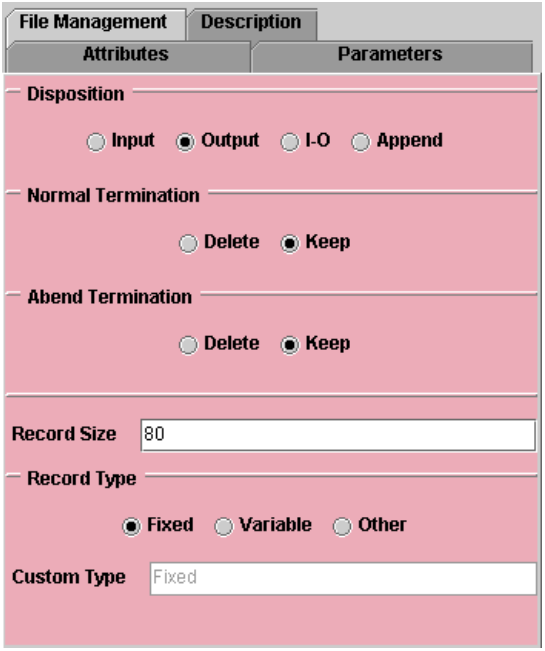

図 3-24　印刷ファイルの「File Management」タブ

b. 次の「Disposition」オプションのいずれかを選択します。

- Input
- Output
- I-O (Input/Output)
- Append

c. 次の「Normal Termination」オプションのいずれかを選択します。

- Delete
- Keep

d. 次の「Abend Termination」オプションのいずれかを選択します。

- Delete
- Keep

e. 「レコードサイズ」の数値フィールドに値を入力します。

f. 次のレコードタイプから選択します。

- Fixed (固定)
- Variable (可変)

- Other (その他)
- 独自のレコードタイプを指定する場合は、「Custom Type」テキストボックスに情報を入力します。このフィールドはオプションです。

3. 印刷ファイルパラメータ (有効な `SYSOUT` パラメータ) を定義する方法は次のとおりです。

   a. 「Parameters」タブをクリックします。

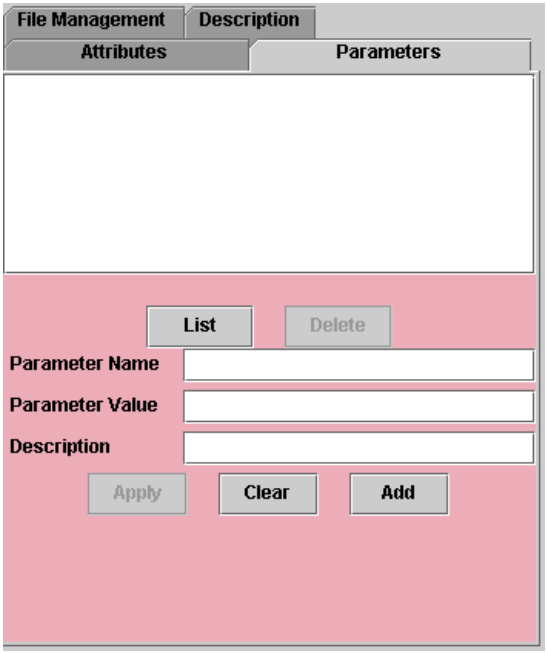

図 3-25　印刷ファイルの「Parameters」タブ

   b. 新しいパラメータを追加する方法は次のとおりです。

      i. 「Parameter Name」テキストボックスに一意の名前を、また「Parameter Value」テキストボックスにその値を入力します。

      ii. 「Description」テキストボックスに説明を入力します (省略可能)。

      iii. 「Add」をクリックします。

      新しいパラメータが一覧に表示されます。

注 – エントリを削除するには、削除するパラメータを一覧から選択して「Delete」をクリックします。

---

c. **プロジェクトですでに定義されているパラメータの 1 つを選択する方法は次のとおりです。**

i. **「List」をクリックします。**

プロジェクトに定義されているパラメータの一覧が表示されます。

ii. **パラメータを選択します。**

iii. **「OK」をクリックします。**

「Parameter Name」、「Parameter Value」、および「Description」テキストボックスに現在の値が表示されます。

iv. **必要に応じて値を編集し、「Add」をクリックします。**

パラメータが一覧に追加されます。

d. **パラメータを変更する方法は次のとおりです。**

i. **一覧でパラメータを選択します。**

「Parameter Name」、「Parameter Value」、および「Description」テキストボックスに現在の値が表示されます。

ii. **フィールドのいずれかを編集し、「Apply」をクリックします。**

一覧に、更新後のパラメータが表示されます。

4. **「Description」タブをクリックして、定義するファイルの説明をテキストボックスに入力します。**

5. **ファイルの定義が完了したら、「OK」をクリックします。**

新しいファイル定義が、ツリーペインと選択したステップのサブフォルダに追加されます。

エラーが発生すると、Job Editor はその理由を表示します。たとえば、必須フィールドのいずれかに情報を入力していない場合、エラーが発行されます。

連結ファイルの定義

連結ファイルには、タイプが GDG、標準、入力のいずれかである 1 つ以上のファイルが含まれます。連結ファイルは、Job Editor のツリーペインで固有のアイコンで表示されます。

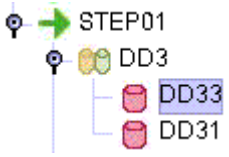

図 3-26　連結ファイルアイコン

▼ 連結ファイルを定義する

1. 基本的なファイル情報を定義します。

   a. 「Attributes」タブをクリックします。

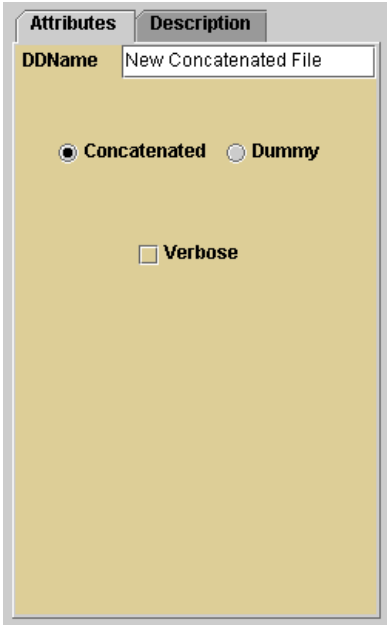

図 3-27　連結ファイルの「Attributes」タブ

b. 「DD Name」テキストボックスに、ファイルの一意の名前、たとえば `DD1` を入力します。

入力したファイル名が同じステップ内ですでに定義されている場合、新しいファイルを追加しようとするとエラーが表示されます。

c. デフォルトの「Concatenated」または「Dummy」を選択します。

d. ジョブ履歴ファイルに書き込まれたメッセージなどの情報を確認する場合は、「Verbose」チェックボックスを選択します。

これは、`ASSGNDD` マクロで `verbose` オプションを付けた処理と同じです。

2. 「Description」タブをクリックして、定義するファイルの説明をテキストボックスに入力します。

3. ファイルの定義が完了したら、「OK」をクリックします。

新しいファイル定義が、ツリーペインと選択したステップのサブフォルダに追加されます。

エラーが発生すると、Job Editor はメッセージを表示します。たとえば、必須フィールドのいずれかに情報を入力していない場合、エラーが表示されます。

▼ 1 つ以上のファイルを連結ファイルに追加する

1. Job Editor のツリーペイン内の連結ファイルアイコンを選択します。

2. 「File」→「New」→「File Type」を選択し、次のいずれかを選択します。
   - Standard
   - GDG
   - Input

3. 選択したファイルタイプに応じて、GDG ファイル、標準ファイル、または入力ファイルを定義する手順を実行します。

連結ファイルに割り当てる各ファイルは、連結ファイルアイコンの下にサブノードとして表示されます。

▼ 別名を定義する

1. 基本的なファイル情報を定義します。

   a. 「Attributes」タブをクリックします。

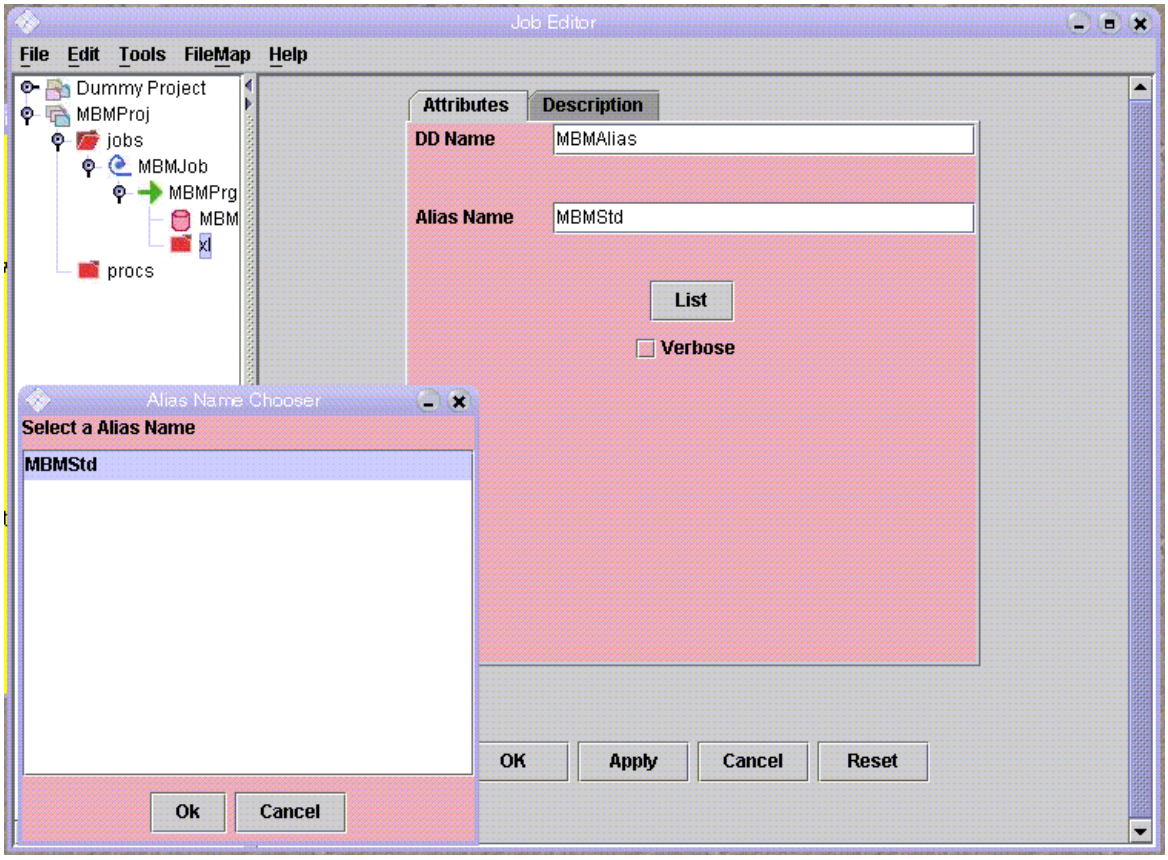

図 3-28　別名ファイルの「Attributes」タブ

   b. 「DD Name」テキストボックスに、ファイルの一意の名前を入力します。

   入力したファイル名が同じステップ内ですでに定義されている場合、新しいファイルを追加しようとするとエラーが表示されます。

   c. 「Alias Name」テキストボックスで、次のいずれかの方法を使用し、DD 名に関連付けるファイルの名前を指定します。

   - 名前がわかっている場合は、その名前を入力します。
   - 「List」をクリックし、前に定義した DD 名の一覧を表示して目的のファイルを選択し、「OK」をクリックします。

d. **ジョブ履歴ファイルに書き込まれたメッセージなどの情報を確認する場合は、「Verbose」チェックボックスを選択します。**

これは、ASSGNDD マクロで verbose オプションを付けた処理と同じです。

2. **「Description」タブをクリックして、定義するファイルの説明をテキストボックスに入力します。**

3. **ファイルの定義が完了したら、「OK」をクリックします。**

新しいファイル定義が、ツリーペインと選択したステップのサブフォルダに追加されます。

エラーが発生すると、Job Editor はメッセージを表示します。たとえば、必須フィールドのいずれかに情報を入力していない場合、エラーが表示されます。

Job Editor ノードの編集

この節では、Job Editor ノードを編集する方法について説明します。

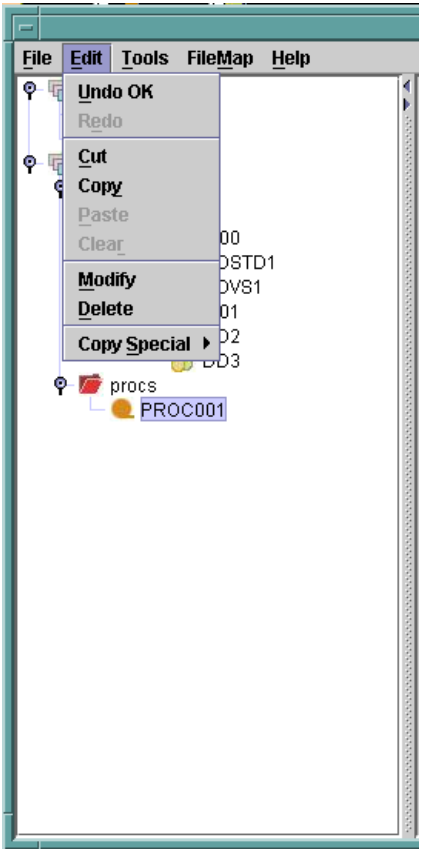

図 3-29　Job Editor ノードの編集

▼ ノードをコピーする

1. コピーするノードを選択します。
2. メニューで、「Edit」→「Copy」を選択します。

   「Edit」→「Copy」メニューオプションは、ノードのみをコピーします。選択したノードに定義されているすべてのサブノードをコピーするわけではありません。

3. 元のノードのコピー先となるノードを選択します。
4. メニューで、「Edit」→「Paste」を選択します。
   元のノードのコピーが、コピー先ノードに追加されます。
5. 右のパネルで、コピーしたノードの新しい名前を入力します。
6. 必要に応じて、その他の変更をノードに加えます。
7. 「OK」をクリックします。

▼ ノードおよびそのサブノードをコピーする

1. コピーするノードを選択します。
2. メニューで、「Edit」→「Copy Special」→「Copy All Subnodes」を選択します。
3. 元のノードのコピー先となるノードを選択します。
4. メニューで、「Edit」→「Paste」を選択します。
   元のノードとそのサブノードのコピーが、コピー先ノードに追加されます。
5. 右のパネルで、コピーしたノードの新しい名前を入力します。
6. 必要に応じて、その他の変更をノードに加えます。
7. 「OK」をクリックします。

注 – コピー先ノードは、元のノードと同じタイプでなければなりません。たとえば、ステップノードからプロジェクト、ジョブ、または手続きノードにファイルノードをコピーすることはできません。

▼ ノードを表示する

- Job Editor のツリーペインで、表示するノードを選択します。

  Job Editor ウィンドウの右半分に、「Attributes」タブが表示されます。このフォームは読み取り専用モードです。ノード定義のほかのタブをクリックして、その情報を表示することもできます。

▼ ノードを変更する

1. **Job Editor のツリーペインで、変更するノードを選択します。**

   Job Editor ウィンドウの右パネルに、「Attributes」タブが読み取り専用モードで表示されます。

2. **メニューで、「Edit」→「Modify」を選択します。**

   これでフォームを編集できるようになります。

3. **このノードのフィールドを変更します。**

   ほかのタブで定義されているフィールドを変更するには、編集するタブをクリックします。

   編集中に間違えた場合は、「Reset」をクリックすると、元の値が表示されます。「Apply」をクリックして変更を適用した場合は、「Reset」ボタンをクリックしても、元の値は復元されません。

4. **変更を適用する方法は、次のとおりです。**

   - 「Apply」をクリックして、現在のパネルに変更を適用します。
   - すべての変更をノードに適用する場合は、「OK」をクリックします。
   - 現在のノードを変更しない場合は、「Cancel」をクリックします。

   「OK」または「Cancel」をクリックすると、フォームには、読み取り専用モードで再度ノード情報が表示されます。

▼ ノードを移動する

1. **移動するノードを選択します。**

   プロジェクト、ジョブ、手続き、ステップ、ファイルノードをカット&ペーストできます。

2. **メニューで、「Edit」→「Cut」を選択します。**

3. **ツリーペインで、コピー先となるノードを選択します。**

4. **メニューで、「Edit」→「Paste」を選択します。**

   カットされたノードとそのすべてのサブノードが、コピー先ノードの下にペーストされます。

5. **右のパネルで、ノードの新しい名前を入力します。**

6. **必要に応じて、その他の変更をノードに加えます。**

7. **「OK」をクリックします。**

注 – ペーストするノードと互換性のないノードを選択すると、「Paste」メニューオプションは使用できません。たとえば、プロジェクトノードまたはファイルノードの下にステップノードをペーストすることはできません。

▼ ノードを削除する

1. **Job Editor のツリーペインで、削除するノードを選択します。**

   プロジェクト、ジョブ、手続き、ステップ、ファイルノードを選択できます。

2. **メニューで、「Edit」→「Delete」を選択します。**

   ノードを削除すると、それに関連付けられているすべてのサブノードとファイルもツリーペインから削除されます。

▼ 削除されたノードを復元する

- **ノードを削除する際にエラーが発生した場合、メニューで「Edit」→「Undo Delete」を選択して元の位置に復元します。**

ジョブと手続きスクリプトの生成

ジョブと手続きの作成後、Job Editor を使用してバッチマクロスクリプトを生成できます。

▼ スクリプトを生成する

1. **Job Editor のツリーペインで、ジョブ、手続き、またはプロジェクトフォルダを選択します。**

   ジョブまたは手続きフォルダを選択する場合、そのフォルダのすべてのジョブまたは手続きは、スクリプト生成の際に選択されます。

2. **メニューで、「File」→「Export」→「Batch Macro File」を選択します。**

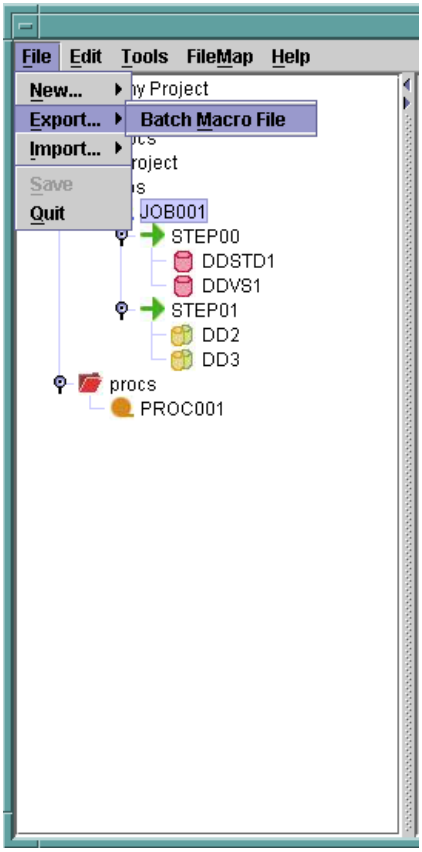

図 3-30　　マクロジョブまたは手続きの生成

3. **ファイル選択リストが表示されたら、プロジェクトディレクトリの名前を入力するか、「Browse」をクリックしてパス名を指定します。**

   このパス名は、プロジェクトの作成時に定義したものです。たとえば、プロジェクト名が PROD でパス名が `/home/projects` の場合、マクロスクリプトのディレクトリパスは `/home/projects/PROD` です。

4. **ファイル選択ウィンドウで「Select」をクリックし、エクスポートを開始します。**

   マクロスクリプトがすでに存在する場合、それを上書きするかどうかを確認するメッセージが表示されます。

   - スクリプトを上書きする場合は、「Yes」または「Yes to All」をクリックします。
   - スクリプト生成を強制的に中止する場合は、「No」または「No to All」をクリックします。

5. **エクスポートが正常に完了したら、「OK」をクリックして確認ダイアログボックスを閉じます。**

第4章

Sun MBM コマンド

GUI によるさまざまなジョブ管理タスクの実行に加え、Sun MBM コマンドでも、ジョブのサブミット、ジョブの状態のチェックなど、さまざまなタスクを実行できます。この章では、これらのコマンドについて説明し、コマンドの使用例を示します。

ソフトウェアをインストールすると、デフォルトで設定されるセキュリティー環境では、すべてのユーザーが Sun MBM コマンドを実行できます。通常、ユーザーは自分自身がサブミットしたバッチジョブだけを参照できます。別のユーザーによってサブミットされたジョブの参照も、ジョブの状態変更も取り消しもできません。ただし、Sun MBM 管理者は任意のジョブの参照、状態変更、および取り消しができます。

システムのセキュリティーを維持するには、システムの本番稼動環境を制御するコマンドへのアクセスを制限する必要があります。各種コマンドおよびその他の機能に対するアクセス制御の詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

次の表に、Sun MBM コマンドを示します。

表 4-1 Sun MBM コマンド (1 / 4)

コマンド	説明
100 ページの「abtjob - バッチジョブの強制的な中止」	現在実行中のジョブまたはコマンドを取り消します。
103 ページの「admlog - ログファイルの管理」	Sun MBM の内部追跡ログファイルを定義するための対話式コマンド。
104 ページの「anmjob - ジョブのデバッグ」	バッチジョブ用に COBOL デバッガを呼び出します。
105 ページの「bam - Batch Administration Manager」	BAM メニューを使用しないで特定のシステム機能を実行します。
106 ページの「batch_shut - ノードの停止」	Sun MBM を停止または終了します。
107 ページの「batch_start - ノードの起動」	Sun MBM ノードを起動します。

表 4-1　Sun MBM コマンド (2 / 4)

コマンド	説明
108 ページの「batchelp - コマンド情報の表示」	すべての Sun MBM コマンドとその記述形式を表示します。
108 ページの「cfm - File_Map の変更」	File_Map を変更します。
114 ページの「chgjcl - 指定したクラスに属しているジョブの変更」	指定したクラスで、実行されていないすべてのジョブのディスポジションを変更します。
116 ページの「chgjob - ジョブの属性の変更」	指定したジョブの属性を変更します。
122 ページの「crtact - アクティビティー (スレッド) の作成」	アクティビティー (スレッド) を作成します。アクティビティーはバッチジョブを実行するために必要です。
124 ページの「crtflm - File_Map エントリの作成」	File_Map 内のエントリを作成、変更、または削除します。
126 ページの「dltact - アクティビティー (スレッド) の削除」	crtact コマンドで作成されたアクティビティー (スレッド) を削除します。
128 ページの「dltjcl - 指定したジョブクラスに属しているジョブの削除」	指定したジョブクラスに属しているジョブを削除します。
130 ページの「dltjob - ジョブの削除」	指定したジョブを SYSIN ファイルから削除します。
131 ページの「dostrans - VSE JCL からマクロジョブスクリプトへの変換」	IBM VSE JCL ストリームを Sun MBM マクロスクリプトに変換します。
138 ページの「ebmdate - Sun MBM の現在日時の取得」	Sun MBM の現在日時を取得します。
139 ページの「ebminfo - 実行環境に関する情報の取得」	現在の Sun MBM ノードまたは遠隔 Sun MBM ノードの実行環境に関する情報を取得します。
142 ページの「ebmsnap - システムのスナップショットの作成」	障害追跡時にシステムのスナップショットを作成します。
143 ページの「ebmsys - サブシステムの作成と管理」	Sun MBM に対して有効なすべてのサブシステムを定義します。
152 ページの「ebmtime - Sun MBM 時刻の取得」	このコマンドは、Sun MBM 時刻を取得するために使用します。
153 ページの「ebmx - Sun MBM メインメニューの表示」	X 端末上で Sun MBM メインメニューを表示します。

表 4-1　Sun MBM コマンド (3 / 4)

コマンド	説明
155 ページの「ftval - File_Map への動的アクセス」	ジョブの実行時、動的に File_Map にアクセスします。
156 ページの「histprt - 履歴ファイルの出力」	batch_start によって最初に Sun MBM を起動した時点で生成されたメッセージを出力します。
158 ページの「infact - アクティビティー情報の取得」	特定のアクティビティーまたはすべてのアクティビティーに関する情報を表示します。
159 ページの「infjbs - 実行中のすべてのジョブに関する情報の取得」	実行中の複数のジョブに関する情報を表示します。
162 ページの「infjob - 実行中のジョブに関する情報の取得」	実行中の 1 つのジョブに関する情報を表示します。
163 ページの「insjbl - ジョブの検査」	1 つまたは複数の待機中のジョブについて、その待機時と完了後の状態を確認できます。
165 ページの「insjob - sysin 内のジョブの状態表示」	特定のジョブについて、サブミット済み、実行中、または終了済みの状態を取得します。
167 ページの「kixdate - ノードの日付の変更」	Sun MTP と Sun MBM システムで使用される日付と時刻を変更できます。
168 ページの「lgprint - ログ情報の取得」	admlog コマンドで定義された追跡ファイルから Sun MBM 内部追跡エントリを抽出し、その情報を $PUBLIC/msg/elog_print ファイルに書き込みます。
171 ページの「lstjcl - ジョブクラス情報の一覧表示」	1 つまたはすべてのジョブクラスについての情報を一覧表示します。
173 ページの「lstjfl - ジョブファイルの一覧表示」	JCL ファイルまたはバッチジョブファイルの内容を一覧表示します。
174 ページの「lstjob - ジョブの属性の一覧表示」	実行待機中の特定のジョブに関する情報を取得します。
178 ページの「lststs - 履歴ファイル内のジョブ状態の一覧表示」	Sun MBM 履歴ファイルからそのユーザーに関連するメッセージを表示し、オプションを指定した場合は、ジョブ履歴ファイルの内容も表示します。
181 ページの「mvstrans - MVS JCL ジョブスクリプトへの変換」	MVS JCL ストリームを Sun MBM マクロスクリプトに変換します。
189 ページの「rpljob - ジョブへのデータの引き渡し」	ジョブスクリプト内のバッチシェル accept 文で一時停止しているジョブに、ユーザーが指定した文字列のデータを渡すことができます。
190 ページの「rsmjob - ジョブの実行の再開」	susjob コマンドによって中断されたバッチジョブの実行を再開します。
191 ページの「runjcl - ジョブクラスの実行」	保持 (h) ディスポジションコードを指定してサブミットしたジョブを、適切なクラスを指定して実行します。

表 4-1　Sun MBM コマンド (4 / 4)

コマンド	説明
193 ページの「runjob - ジョブの実行」	1 つのバッチジョブをディスポジションコード k に設定して実行し、実行後も SYSIN ファイルに保持します。
195 ページの「subjob - ジョブのサブミット」	VSAM 以外のサブシステムにジョブをサブミットします。
203 ページの「susjob - ジョブの中断」	再開コマンド rsmjob が実行されるまでの間、バッチジョブの実行を中断します。
205 ページの「unikixjob - Sun MTP へのバッチジョブのサブミット」	Sun MTP 領域に接続するように構成されたサブシステムにジョブをサブミットします。このコマンドでサブミットするジョブは、通常 VSAM データセットにアクセスする Sun MTP のサービスを利用する必要があります。

abtjob - バッチジョブの強制的な中止

abtjob コマンドは、現在実行中のジョブまたはコマンドを取り消します。

形式

abtjob *jon* [-n *name*] [-N *nodename*] [-s job | step]

説明

jon

ジョブのサブミット時に割り当てられるジョブ番号。

-n *name*

システムコマンドまたはアプリケーションプログラムの名前。-n オプションは cmd パラメータとともに使用できます。このオプションによって、*name* で指定したコマンドを強制的に中止する前に、該当のバッチシェルによって現在実行されているのが確かにそのコマンドかどうかがチェックされます。このチェックを通過しないと、abtjob によってエラーメッセージが表示され、現在実行中のコマンドは強制的に中止されません。

-N *nodename*

abtjob は、*nodename* で指定した遠隔ノード内で *jon* によって識別されるジョブを強制的に中止します。

指定する *nodename* は、ローカルの Sun MBM システムの遠隔 Sun MBM ノードとして定義されている必要があります。また、その遠隔ノードは、ローカルの Sun MBM ノードの 1 つとして構成されている必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

-s job | step

job: ジョブをただちに取り消します (デフォルト)。

step: 現在実行中のステップでコマンドまたはアプリケーションプログラムを強制的に中止します。Server Express 環境では、FaultFinder ダンプも生成されます。

現在実行されているコマンドまたはアプリケーションプログラムを強制的に中止するには、-s オプションを step パラメータとともに指定します。現在実行中のプログラムを確認するには、infjbs コマンドを使用します。詳細は、159 ページの「infjbs - 実行中のすべてのジョブに関する情報の取得」を参照してください。infjbs コマンドを実行すると、現在実行中のバッチジョブの関数が、CmdName というヘッダーの下に表示されます。

-s step パラメータを指定して abtjob を発行すると、該当のステップが強制的に中止されます。

デフォルトの -s job オプションを指定して abtjob を発行すると、指定したジョブがただちに終了します。

このコマンドは、自分のユーザー ID でサブミットしたジョブだけに有効です。別のユーザー ID でサブミットされたジョブを取り消すことはできません。ただし、Sun MBM 管理者は任意のジョブを強制的に中止できます。

例

バッチジョブ test に、次のコマンドが含まれているとします。

```
start
echo before first sleep
sleep 3000
echo after first sleep
sleep 3000
echo before second sleep
sleep 3000
echo after second sleep
exit 0
```

■ このジョブは、subjob コマンドによってサブミットされます。このジョブには、*jon* として 94 が割り当てられ、デフォルトでクラス a が割り当てられています。

```
[node1/fs1] # subjob test
BS1100(I) Job test submitted :jon = 94 class = a
```

■ infjbs コマンドにより、このジョブが現在実行しているステップまたはシェルスクリプトコマンドが表示されます。

```
[node1/fs1] # infjbs
Jon   Jobname Stepname  JobTm     Cmd      CmdTm           User
Act   P C D   LCC       JobStatus PLines   Job_Start       Cmd_Start      Rank
JobIdentifier                     Ebmsys   JobParameter
-------------------------------------------------------------------------------
094   test    BeginJob  00m08s    sleep    00m06s           mbmuser
act1  5 a d   000 000   Running   000003   02May2005 10:08  02May2005 10:08  0
test@02May2005:10:08:42           hazel    null
BS1038(I) infjbs command executed
```

■ abtjob 94 -s step により、このバッチジョブで現在実行中のコマンド (sleep) が強制的に中止されます。この場合、ジョブは強制的に中止されないで、現在のコマンドだけが中止されます。

```
[node1/fs1] # abtjob 94 -s step
BS1046(I) Job aborted (jon 94)
```

■ 次に、後続のコマンド (sleep) が実行されます。

```
[node1/fs1] # infjbs
Jon   Jobname Stepname  JobTm      Cmd       CmdTm           User
Act    P C D   LCC       JobStatus  PLines   Job_Start       Cmd_Start    Rank
JobIdentifier                      Ebmsys    JobParameter
-------------------------------------------------------------------------------
094   test     BeginJob  00m08s     sleep    00m06s           mbmuser
act1  5 a d   000 01    Abt'ing    000007    02May2005 10:08  02May2005 10:11 0
test@02May2005:10:11:28            hazel     null
BS1038(I) infjbs command executed
```

admlog - ログファイルの管理

admlog コマンドは、内部追跡ログファイルを定義するために使用する対話式コマンドです。問題が発生した場合に、サポート担当者が必要な情報や追跡エントリを収集するために admlog を使用します。

admlog コマンドの主な用途は、ログファイルを循環ファイルとシリアルファイルのどちらかに定義することです。循環ファイルとする場合は 1 ファイルだけ定義できます。シリアルファイルとする場合は、複数のファイルを定義できます。複数のシリアルファイルを定義すると、Sun MBM によって最初のファイルの名前に接尾辞 _00 が追加され、そのファイルからログの書き込みが開始されます。次のファイルの名前には接尾辞 _01 が追加され、後続のファイルにも順番に接尾辞が追加されます。すべてのシリアルファイルがログで満杯になると、最初のファイル (接尾辞 _00) が上書きされます。この処理は全ファイルが上書きされるまで続行され、再び最初のファイルから上書きが開始されます。

形式

admlog

次の例は、admlog コマンドを実行した時に表示されるメニューと、使用可能な機能です。

```
[node1/fs1] # admlog
ADMLOG FUNCTIONALITY

1)     Current log environment information
2)     Configure log environment
3)     Set/reset log level for a specific module
4)     Reset all log levels
5)     Print current log levels
6)     Quit admlog
choice > 2

ADMLOG :CONFIGURE LOG ENVIRONMENT
1)     Circular file
2)     Serial files
3)     Set default log file
4)     Quit admlog
choice > 1
Give max file dimension (in Kbytes) [default=32] > 64
Default file pathname:/local_space/mfr/node1/public/msg/elgfile
Give file pathname > /SRO/mbmuser/elgfile
New log configuration activated
Press 'return' key to continue
```

anmjob - ジョブのデバッグ

anmjob は、COBOL デバッガを使用してバッチジョブをデバッグするためのコマンドです。

形式

anmjob *jon*

説明

jon は、ジョブ番号です。

バッチジョブをデバッグするには、unikixjob または subjob コマンドによるジョブのサブミット時に、デバッグオプション -a を指定する必要があります。バッチジョブは、anmjob コマンドによって端末がそのジョブにアクセスするまで待機状態になります。

このコマンドを使用する場合は、次のことを確認する必要があります。

- バッチ COBOL プログラムがデバッグオプション (複数の場合あり) でコンパイルされていること。
- 別のユーザーがサブミットしたジョブをデバッグしていないこと。デバッグできるのは、ユーザー自身がサブミットしたジョブのみです。

anmjob コマンドを実行する場合、対象ジョブに対して unikixjob または subjob コマンドを実行した時と同じウィンドウを使用する必要はありません。

詳細は、19 ページの「COBOL デバッガを有効化する」を参照してください。

例

```
[node1/vs1] # unikixjob testanm -a
BS1100(I) Job testanm submitted:jon 46 class=a
.
.
.
[node1/vs1] # anmjob 46
```

bam - Batch Administration Manager

bam コマンドを使用すると、BAM メニューを使用しないで特定の機能を実行できます。

形式

bam [-u | [*options*]]

説明

-u

使用可能なオプションを表示します。

オプション

startbatch: ノードを起動します

stopbatch: ノードを停止します

systemstatus: 動作中のサブシステムを監視します

activejobs: 動作中のジョブを監視します

completedjobs: 完了したジョブを照会します

jobclasses: 動作中のジョブクラスを一覧表示します

clearjoblog=*nnn*: 当日または 1 日以上前に終了したジョブに関するログエントリを削除します

diskspace: ディスク容量が十分かどうかを確認します

例

ノードを起動します。

```
[node1] # bam startbatch
```

batch_shut - ノードの停止

batch_shut コマンドは、ノードを停止します。ノードが正常に終了しない場合は、-f オプションを指定して batch_shut を実行すると、強制的に終了できます。

注意 – これはユーザーが使用できるコマンドですが、通常は Sun MBM メインメニューのボタンまたは BAM を使用してノードを停止してください。

形式

```
batch_shut [-f|-r]
```

説明

-f

ノード、および有効なすべてのソケットとプロセスを強制終了します。

-r

ノードを終了し、自動的に再起動します。

ノードの停止後、再起動するまでジョブはサブミットできません。

例

```
[node1] # batch_shut
OS1000(I) Shutdown Started
OS1001(I) Shutdown in Progress
OS1002(I) Shutdown of bqm daemon
OS1002(I) Shutdown of psg daemon
OS1002(I) Shutdown of vcf daemon
OS1002(I) Shutdown of lgdem daemon
OS1002(I) Shutdown of ebmmd daemon
OS1003(I) Shutdown Completed

[node1] # batch_shut -f
OS1000(I) Shutdown Started:
Sun MBM daemon "bqm"     pid=8278: ....     shutdown
Sun MBM daemon "ebmmd"   pid=8280: ....     shutdown
Sun MBM daemon "lgdem"   pid=8292: ....     shutdown
Sun MBM daemon "psg_d"   pid=8294:          not running
Sun MBM daemon "vcf"     pid=8275: ....     shutdown
OS1003(I) Shutdown Completed.
[E801] data #
```

batch_start - ノードの起動

batch_start コマンドは、ノードを起動します。-f オプションを指定しないで batch_start を発行すると、ノードはバックグラウンドで実行され、ただちにコマンド行プロンプトが表示されます。

注意 – これはユーザーが使用できるコマンドですが、通常は Sun MBM メインメニューのボタンまたは BAM を使用してノードを起動してください。

形式

```
batch_start [-f]
```

説明

-f

batch_start を実行する時点でノードが完全に起動しているかどうかにかかわらず、すべての Sun MBM デーモンを呼び出すように強制的に batch_start に指示します。システムクラッシュの発生後は、このオプションが必要です。

ノードの起動後、ジョブをサブシステムにサブミットできます。

例

```
[node1] # batch_start
OS1004(I) Starting Sun MBM Batch Node (Version 10.1-06/15/2005)
OS1005(I) Startup in Progress
OS1006(I) Startup of ebmmd daemon executed
OS1006(I) Startup of lgdem daemon executed
OS1006(I) Startup of vcf daemon executed
OS1006(I) Startup of psg daemon executed
OS1006(I) Startup of bqm daemon executed
OS1007(I) Startup Completed
```

batchelp - コマンド情報の表示

batchelp コマンドは、すべての Sun MBM コマンドとその記述形式を表示します。

形式

batchelp [-a] [-v] [*command*] ...

説明

-a

すべてのコマンドとその構文を表示します。

-v

Sun MBM のリリースバージョンを表示します。

1 つまたは複数のコマンドを指定した場合は、そのコマンドの構文規則が表示されます。

例

次の 2 つのコマンドの構文を表示します。

```
[node1] # batchelp lststs infact
BS0017(I) Usage:lststs [-j jon [ -t ] [ -c ]] [-u user] [-N nodename]
BS1993(I) Usage:infact [name] [-N nodename]
```

cfm - File_Map の変更

cfm コマンドを使用すると、File_Map 内にある複数のエントリを変更できます。次のフィールドを変更できます。

- データセット名 (DSN)
- カタログ名
- タイプ
- 割り当て済みファイル名
- レコード属性
- GDG 番号

cfm コマンドの使用例については、『Sun Mainframe Batch Manager ソフトウェア移行ガイド』を参照してください。

形式

```
cfm -i input-filemap [-o output-filemap] [-c catalog] [-C catalog]
[-d DSN] [-D DSN] [-f filename] [-F filename] [-g n] [-G n]
[[-K]|[-K -x]|[-K -y]] [-N GDG-table] [-r record-descriptor-table]
[-R directory-pathname] [-t type] [-T type] [-v VSAM-table]
[-V VSAM-table] [-e] [-p|-P|-n]
```

注 – 文字列または部分文字列に空白が含まれる場合は、単一引用符を使用する必要があります。

説明

`-c` *catalog*

catalog (完全なカタログ名またはその部分文字列) に属する入力 `File_Map` からエントリを選択します。デフォルトの *catalog* は `MASTERCAT` です。

`-C` *catalog*

`-c` *catalog* オプションで指定したエントリの *catalog* 名を変更します。

`-d` *DSN*

DSN (完全なデータセット名またはそのデータセット名の部分文字列) に一致するデータセット名を持つ入力 `File_Map` からエントリを選択します。

`-D` *DSN*

指定した `DSN` の形式を基に選択したエントリのデータセット名を変更します。

`-e`

`-t`、`-c`、`-d`、または `-f` オプションを使って選択したあとで、エントリを削除します。選択したエントリは出力の `File_Map` には書き込まれませんが、入力の `File_Map` に残ります。デフォルトでは、標準出力として削除されるエントリがプログラムで表示されます。

`-f` *filename*

filename (完全な割り当てファイル名またはその部分文字列) に一致する割り当てファイル名を持つ入力 `File_Map` からエントリを選択します。たとえば、パス名に `/tmp/data` を含むすべてのエントリを選択するには、次のように入力します。

```
cfm -f /tmp/data
```

選択した File_Map エントリの割り当てファイル名を変更します。*filename* は、最大 16 フィールドで 1,024 文字までです。

(%{D1},%{D2},%{D3},.....,%{D16}) または (%{d1},%{d2},%{d3},.....,%{d16}) は、大文字または小文字のデータセット名の 16 個のフィールドです。

(%{F1},%{F2},...,%{F16}) または (%{f1},%{f2},...,%{f16}) は、大文字または小文字の割り当てファイル名の 16 個のフィールドです。

新しいパス名の個々のフィールドは、次のうちの 1 つまたはその組み合わせです。

- ユーザー指定の文字列
- 大文字または小文字のデータセット名
- 大文字または小文字の割り当てファイル名

このオプションで (%{D1},%{D2},...%{D16}) または (%{d1},%{d2},...%{d16}) を使用する場合、プログラムにより、これらのフィールドの変数はパス名に合わせて適切な形式に変更されます。

%{F2},%{F3},%{F4}: 大文字のパス名の 2 番目、3 番目、および 4 番目のフィールドを表します。

%{f2},%{f3},%{f4} : 対応するフィールドを小文字で表します。

現在のパス名を大文字または小文字に変換するには、%{F*} または %{f*} を使用します。

パス名に小文字または大文字のデータセット名を含めるには、%{d*} と %{D*} を使用します。データセット名の変数はすべて、パス名に含まれる前に適切な形式に変換されます。

フィールドは MVS JCL のルールを基に決められます。

-g *n*

値が *n* の GDG 番号を持つ入力 File_Map からエントリを選択します。

-G *n*

選択した GDG ファイルのために生成するデータセットの最大数を設定します。1 ～ 99 の数字で *n* を指定します。

-i *input-filemap*

入力する File_Map の名前を定義します。入力 File_Map は変わらないまま残ります。環境変数 FILEMAP も使用できます。次に例を示します。

```
cfm -i $FILEMAP
```

-K

ディレクトリまたはファイルの保全性を調査します。

-n

-t、-c、-d、および -f オプションと一緒に使用すると、データセットのリストおよび現在の世代番号が表示されます。

各データセットの正しい世代番号で *GDG_table* を手動で更新し、-N オプションの入力に使用できます。次に例を示します。

```
cfm -i input-filemap -t GDG -n > GDG-table
```

-N *GDG-table*

出力 File_Map の世代番号を変更します。*GDG_table* は、データセット名と新規世代番号 (1 ～ 99) から構成され、少なくとも 1 つの空白で区切られます。_LIB または VS ファイルの世代番号は変更できません。次に例を示します。

```
BKUP.WSBD011.SCRACRED 7
BKUP.WPUD780.FAXLOG 12
DBKP.WPUD042.MAINTEDI 3
```

-o *output-filemap*

更新する File_Map の一意の出力先ファイル名を定義します。出力 File_Map に使用される名前が存在してはいけません。

-o オプションを指定しない場合は、-p または -P オプションを指定する必要があります。

-p

入力 File_Map からのエントリを選択して表示します。このパラメータを使用する場合は、-t、-c、-d、および -g オプションを使って選択したエントリを標準出力に表示できるため、出力 File_Map を指定する必要はありません。

-P

入力 File_Map で変更したエントリだけが標準出力に表示されます。この表示で、指定した変更の影響を受けるエントリを識別します。

出力 File_Map を指定した場合は、入力 File_Map のコピーが、変更と同時に出力 File_Map に書き込まれます。指定しなかった場合は、変更したエントリだけが標準出力に表示されます。

`-r` *record-descriptor-table*

次の形式の *record-descriptor-table* のフルパス名とエントリです。

DSN type record-length

説明

DSN: データセット名。

type: 次のいずれかの値です。

`F`: 固定長レコード。

`V`: 可変長レコード。

others: ユーザーが指定したその他のレコードタイプ。

record-length: 有効な任意の数字。

`-R` *directory-pathname*

データセットのレコード説明を含むファイルが `cfm` コマンドで作成されるディレクトリのフルパス名。

`-t` *type*

type は次のいずれかです。

`FS`: FS ファイルタイプをすべて選択します。

`VS`: VS ファイルタイプをすべて選択します。

`GDG`: 世代番号が 0 より大きい FS ファイルタイプをすべて選択します。

`LIB`: *catalog* が `_LIB` の FS ファイルタイプをすべて選択します。

デフォルトは、`FS`、`VS`、および `GDG` です。

`-T` *type*

選択したエントリを *type* に変更します。

`FS`: ファイルタイプを `FS` に設定します。

`VS`: ファイルタイプを `VS` に設定します。

`GDG`: 現在の世代番号が 0 以下の場合に世代番号を 9 に設定します。

`LIB`: *catalog* を `_LIB` に設定します。

-v *VSAM-table*

ファイルタイプを VS に変更し、割り当てファイル名を DFHCSD データセット名に変更します。

VSAM-table は、DFHCSD レポートを含む順編成ファイルのフルパス名です。レポートの形式は、次のとおりです。

```
FILE(SSUACCT) GROUP (CLLMAUL)
DESCRIPTION
VSAM-PARAMETERS
DSNAME(SSUVD.ACCT.ACCNT)
...
```

-V *VSAM-table*

ファイルタイプを VS に変更し、割り当て済みファイル名を DFHFCT データセット名に変更します。

VSAM-table は、次の形式の DFHFCT マクロを含むファイルのフルパス名です。

```
BASACCT DFHFCT TYPE=DATASET,
DATASET=BASACCT,
DSNAME=BASACCTK.ACCOUNT.MASTER,
.
.
```

使用される FCT マクロファイルは、メインフレームで、REPRO オプションを指定した IDCAMS ユーティリティーを使って作成されます。

-x

ディレクトリのリストを作成します。このオプションは -K オプションとともに使用します。

-y

FS ファイルおよび GDG ファイルだけをリスト表示します。このオプションは -K オプションとともに使用します。

File_Map の更新に cfm を使用すると、次のリターンコードが生成されます。

- 警告なしで正常に完了した場合、0 を返します。
- 正常に完了したものの警告付きの場合、1 を返します。
- 正常に完了しなかった場合、2 以上の値を返します。

chgjcl - 指定したクラスに属しているジョブの変更

chgjcl コマンドは、指定したクラスに割り当てられているジョブのうち、実行されていないすべてのジョブのディスポジションを変更します。ディスポジションコード (d、h、または k) は、ジョブがどのように処理されるかを示します。

形式

chgjcl *class disp* [-N *nodename*] [-T]

説明

class

ジョブの実行を変更するジョブクラス (a ～ z)。

disp

ジョブのディスポジション。

d: 指定したクラスのすべてのバッチジョブについて、実行のスケジューリングを行います。この値は、ジョブクラスに保持ディスポジションコードが定義されている場合に使用します。デフォルト値です。

h: 指定したクラスに割り当てられているジョブはスケジューリングも実行もされず、あとで処理されるまで SYSIN ファイル内に保持されます。

k: ジョブの実行がスケジューリングされ、実行後も SYSIN ファイル内のジョブまたはシェルスクリプトが保存されます。実行後のジョブのディスポジションは、保持に変更されます。

-N *nodename*

nodename で指定した遠隔ノード上で、指定したクラスに割り当てられているジョブのうち、実行されていないすべてのジョブのディスポジションを変更します。

指定する *nodename* は、ローカルの Sun MBM システム内に、遠隔ノードとして定義されている必要があります。また、その遠隔 Sun MBM システムは、ローカルノードの 1 つとして構成されている必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

-T

指定したクラスに割り当てられているジョブの開始時刻を、当日の時刻に適用します。次に例を示します。

現在の時刻が 14:00、クラスに割り当てられているジョブの開始時刻が 12:00 および 15:00 とします。

開始時刻が 12:00 のジョブはただちに開始されます

開始時刻が 15:00 のジョブは 1 時間後に開始されます

-T オプションを指定しなかった場合、開始時刻が 12:00 のジョブは翌日まで開始されません。

-T オプションは、unikixjob コマンドでサブミットしたジョブだけに有効です。

また、このコマンドが有効とするのは、現在ログインしているユーザー ID でサブミットされたジョブだけです。別のユーザー ID でサブミットされたジョブは影響を受けません。

注 – Sun MBM 管理者は、指定したクラスのすべてのジョブを変更できます。

例

次の場合の実行例を示します。

- ジョブをクラス b に割り当て、保持ディスポジションコード h でサブミットします。
- chgjcl b d コマンドを実行し、クラス b に割り当てられているすべてのジョブのディスポジションコードをディスパッチ (d) に変更します。

```
[node1/fs1] # subjob test -c b -d h
BS1100(I) Job test submitted :jon = 92 class = b

[node1/fs1] # lstjob 92
BS1083(I) Command lstjob
Jon    Jobname P C D  Strtime  Endtime  Start from   Status      Type   M   V
User   Cycle   HistoryDir    Suspend at   SaveHistoryDir         X   A
JobIdentifier                Ebmsys       JobParameter       RescheduledAt
-------------------------------------------------------------------------------
092    JOB001  5 b h   -        -       Begin Job    Submitted   Std    n   n
mbmuser                                                        n   n
test@02May2005:09:58:14      hazel        null
BS1038(I) lstjob command executed
```

```
[node1/fs1] # chgjcl b d
BS1038(I) chgjcl command executed

[node1/fs1] # lstjob 92
BS1083(I) Command lstjob
Jon    Jobname P C D  Strtime  Endtime  Start from   Status      Type   M   V
User   Cycle   HistoryDir    Suspend at   SaveHistoryDir         X   A
JobIdentifier               Ebmsys       JobParameter       RescheduledAt
-------------------------------------------------------------------------------
092    JOB001  5 b d   -        -       Begin Job    Submitted   Std    n   n
mbmuser                                                        n   n
test@02May2005:09:58:14      hazel        null
```

chgjob - ジョブの属性の変更

chgjob コマンドは、指定したジョブの属性を変更します。ただし、実行中のジョブについては chgjob オプションを指定できません。また、このコマンドはユーザー自身がサブミットしたジョブだけに使用できます。別のユーザー ID でサブミットされたジョブの属性変更はできません。

注 – 指定したジョブとそのジョブ ID に関連付けられているサブシステムは変更できません。

形式

chgjob *jon* [-a] [-A | -F] [-c *class*] [-C *cycle-time*] [-d *disp*] [-h *history-dir*] [-m] [-n *name*] [-N *nodename*] [-o *save-hist-dir*] [-P *jon-parameter*] [-p *prior*] [-R *stepname* [.*procstepname*]] [-s *start-time*] [-e *end-time*] [-S *stepname* [.*procstepname*]] [-T] [-x] [-v]

説明

jon

ジョブのオカレンス番号。

-a

ジョブのデバッグをオンに設定します。そのジョブでは、ユーザーが特定の端末から anmjob *jon* を実行するまで、どのプログラムも開始されません。

-A | -F

A: ジョブの終了状態を強制的な中止 (Abt - terminated abnormally) に変更します。

F: ジョブの終了状態を正常終了 (FIN - terminated normally) に変更します。このオプションは、あるジョブの正常終了後に実行されるジョブがある場合に、そのジョブが強制的に中止されたときに使用します。

ジョブを終了状態に設定した場合は、そのジョブに依存するジョブの実行をスケジュールできます。ただし、実行中のジョブの状態は変更できません。

注 – -A、-F のどちらも指定しなかった場合、ジョブはサブミット済みの状態に変更されます。

-c *class*

ジョブのクラス (a 〜 z) を変更します。指定したジョブクラス用のアクティビティーを作成しない限り、ジョブは実行されません。

-C *cycle-time*

ジョブを周期的に実行するように指示します。-C オプションには、サイクル時間を秒数で指定します。これにより、そのサイクル時間とジョブの完了時刻に基づいて、ジョブの再スケジューリングが行われます。たとえば -C 20 と指定した場合、ジョブは完了してから 20 秒後に再びスケジューリングされます。

-C オプションは、-d k オプションとともに使用する必要があります。ディスポジションコードが削除 (d) の場合、ジョブは最初の実行後に SYSIN ファイルから削除されます。したがって、再スケジューリングができません。

注 – このオプションは、unikixjob コマンドでサブミットしたジョブだけに有効です。詳細は、205 ページの「unikixjob - Sun MTP へのバッチジョブのサブミット」を参照してください。

-d *disp*

ジョブにディスポジションを割り当てます。

d: 指定したクラスのすべてのバッチジョブについて、実行のスケジューリングを行います。この値は通常、ジョブクラスに保持ディスポジションコードが定義されている場合に使用します。デフォルト値です。

h: 指定したクラスに割り当てられているジョブはスケジューリングも実行もされず、あとで処理されるまで SYSIN ファイル内に保持されます。

k: ジョブの実行がスケジューリングされ、実行後も SYSIN ファイル内のジョブまたはシェルスクリプトが保存されます。実行後のジョブのディスポジションは、保持に変更されます。

-h *history-dir*

指定した出力先ディレクトリに、ジョブ履歴ファイルを次の形式の名前で作成します。

destination-directory/*jobname-jobnumber*.hst.*date.time*

destination_directory は、BAM を使用してサブシステムを作成する時に定義します。

注 – *date.time* には、LC_TIME 環境変数の定義が適用されます。batchenv ファイルをカスタマイズする方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

-m

ジョブが正常に終了したかどうかについての通知メールを、そのジョブをサブミットしたユーザーに送信するように Sun MBM に指示します。メッセージの形式は、次のいずれかです。

- BQM Job *nnn* (*jobname*) could not be executed between *hh:mm* and *hh:mm*
- BQM Job *nnn* ( ) Ended, Status=000
- BQM Job *nnn* (*jobname*) Aborted, Status=3

説明

BQM: バッチキューマネージャー

nnn: ジョブオカレンス番号

jobname: ジョブのサブミット時に指定したジョブ名

-n *name*

サブミットしたバッチジョブの名前。そのジョブ名が *jon* に関連付けられているかどうかを確認できるようにするため、*name* は必ず指定してください。関連付けられていない場合はエラーメッセージが表示され、属性は変更されません。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上で、指定したクラスで実行されていないジョブ *jon* の属性を変更します。

nodename は、ローカルの Sun MBM システム内に、遠隔ノードとして定義されている必要があります。また、その遠隔 Sun MBM システムは、ローカルノードの 1 つとして構成されている必要があります。

遠隔 Sun MBM ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

-o *save-hist-dir*

ジョブの実行終了時に、実行レポートを指定のディレクトリに書き込みます。*save-hist-dir* は、履歴ファイルを作成する場所を定義する環境変数です。次に例を示します。

この環境変数を `setenv HISTDIR /tmp` と定義した場合は、このオプションを `-o HISTDIR` と入力します。

この環境変数は構成ファイルで指定できます。

注 – `-o` オプションを `-h` オプションと併用した場合、実行レポートは *history_dir* ディレクトリに作成されます。

-P *jon-parameter*

25 文字以内の文字列を、実行時にバッチジョブに渡します。このオプションは、サブミッション時にジョブにパラメータを渡すために使用します。ソースファイルの変更は不要です。

指定した文字列は `$JOBPARM` 変数としてジョブに渡されます。この変数の初期値は NULL 文字列に設定されています。指定する文字列によって、別のファイルをソースファイルとして指定したり、別の変数名を指示したり、または処理を決定したりできます。

-p *prior*

ジョブの優先順位をリセットします。Sun MBM では、この優先順位に基づき、指定したクラスに属しているバッチジョブのスケジューリングが行われます。優先順位の有効な値は `0` 〜 `9` で、`9` が最高優先順位です。

-R *stepname.procstepname*

8 文字以内の名前で指定したジョブステップまたは手続きステップからジョブを再開します。

手続きステップ名を指定するには、*stepname.procstepname* のように、ジョブステップ名と手続きステップ名をピリオド (.) で区切ります。再開するステップが存在しないか、無効な場合、ジョブは開始時に強制的に中止されます。

-s *start-time*

ジョブの実行時刻としてスケジューリング可能な最も早い時刻を、次の形式で指定します。

hh:mm

説明

hh: 24 時間表示による時
mm: 00 〜 59 の分

注 – -s オプションは、unikixjob コマンドでサブミットしたジョブにのみ有効です。詳細は、205 ページの「unikixjob - Sun MTP へのバッチジョブのサブミット」を参照してください。

-e *end-time*

ジョブの実行時刻としてスケジューリング可能な最も遅い時刻を、次の形式で指定します。

hh:mm

説明

hh: 24 時間表示による時
mm: 00 ～ 59 の分

注 – -e オプションは、unikixjob コマンドでサブミットしたジョブにのみ有効です。詳細は、205 ページの「unikixjob - Sun MTP へのバッチジョブのサブミット」を参照してください。

-S *stepname.procstepname*

名前で指定したジョブステップまたは手続きステップで、ジョブを中断します。ステップが存在しない場合、ジョブは終了時まで実行されます。

-T

ジョブに関連付けられている時刻を、当日の時刻に適用します。次に例を示します。

現在の時刻が 14:00 で、サブミット済みのジョブに開始時刻 12:00 および -T オプションが指定されている場合、このジョブの実行は、当日のできるだけ早い時刻でスケジューリングされます。

注 – T オプションは、unikixjob コマンドでサブミットしたジョブにのみ有効です。詳細は、205 ページの「unikixjob - Sun MTP へのバッチジョブのサブミット」を参照してください。

-x

バッチジョブが実際に実行される前にプリプロセスルーチンを呼び出すか、バッチジョブの実行後にポストプロセスルーチンを呼び出します。プリプロセスおよびポストプロセスルーチンは、バッチジョブによって呼び出されます。この -x オプションの機能は、subjob および unikixjob の場合と同じです。

-v

ジョブを妥当性検査モードだけで実行します。ユーティリティーとアプリケーションプログラムは実行されませんが、それらの状態を示す CONDITION-CODE は常にゼロ (正常終了) に設定されます。このオプションを指定すると、プログラムやユーティリティーを実行することなく、ジョブの論理チェックを実行できます。

例

ジョブの実行時刻をスケジューリングします。

- ジョブの実行を当日の特定の時刻でスケジューリングするには、-s および -e オプションを使用します。
- *start-time* が現在時刻と同じか、現在時刻よりもあとの場合に、ジョブが開始されます。
- ジョブが *start-time* と *end-time* の間に実行されるようにスケジューリングされていない場合、そのジョブはスケジューリングされません。
- *end-time* は、*start-time* と相関関係にあります。
  - *start-time* が 16:00、*end-time* が 2:00 の場合、終了時間は次の日になります。この場合、lstjob コマンドは、ジョブの終了時刻を 2:00 tm と指定します (tm は tomorrow の略号)。
  - *start_time* を現在時刻よりもあとの時刻に指定した場合、たとえば *start_time* が 14:00 で現在時刻が 12:00 の場合、そのジョブの実行は当日にスケジューリングされます。
  - *start-time* を現在時刻よりも前の時刻に指定した場合、たとえば *start_time* が 12:00 で現在時刻が 14:00 の場合、そのジョブの実行は次の日にスケジューリングされます。この規則は -T オプションを指定することにより無効にできます。
- クラス b 用のアクティビティーを作成します。
- クラス a と保持 (h) ディスポジションコードを指定したジョブ proacc をサブミットします。
- ジョブの実行には、デフォルトのサブシステムを使用します。
- insjob コマンドにより、ジョブがサブミットされたことが示されます。
- chgjob コマンドにより、ジョブクラスとディスポジションが変更され、ジョブが実行可能になります。
- infact コマンドにより、ジョブがアクティビティー actb 内で実行されていることが示されます。

```
[node1/fs1] # crtact actb -c b

[node1/fs1] # subjob proacc -c a -d h
BS1100(I) Job proacc submitted :jon = 32 class = a
```

```
[node1/fs1] # insjob 32
BS1079(I) user mbmuser:job 32 ---> submitted

[node1/fs1] # infact
BS1038(I) infact command executed
BS1070(I) - Inform on activity names :
actb
BS1038(I) infact command executed

[node1/fs1] # chgjob 32 -c b -d d
BS1038(I) chgjob command executed

[node1/fs1] # infact
BS1065(I) Command infact
BS1070(I) - Inform on activity name :
actb (J032)
BS1038(I) infact command executed

[node1/fs1] # insjob 32
BS1079(I) user mbmuser:job 32 ---> running
```

crtact - アクティビティー (スレッド) の作成

crtact コマンドは、アクティビティー (スレッド) を作成します。バッチジョブを実行するには、アクティビティーが必要です。アクティビティーを作成していない場合はジョブを実行できません。

注 – これはユーザーが使用できるコマンドですが、ジョブクラスを構成するときは必ず BAM を使用してください。crtact を使用してアクティビティーを作成した場合、BAM の GUI の使用時にジョブクラスに関する誤った情報が表示されることがあります。ジョブクラスとアクティビティーの作成と管理については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

形式

crtact *name* [-c *class*] [-N *nodename*]

説明

name

アクティビティー (スレッド) の名前。4 文字の英数字で指定する必要があります。それ以外の形式で指定した場合、「incorrect field length （フィールド長の誤り）」というエラーメッセージが表示されます。

-c *class*

特定のアクティビティーに 1 つ以上のクラスを割り当てます。1 つのアクティビティーに 1 ～ 4 つのジョブクラスを関連付けられます。ジョブクラスは、a ～ z の範囲で事前に定義しておく必要があります。たとえば、あるアクティビティーにジョブクラス a と c を関連付けるには、このクラスパラメータに ac と指定します。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上に、指定した *name* とクラスのアクティビティーを作成します。

nodename は、ローカルの Sun MBM システム内に、遠隔ノードとして定義されている必要があります。また、その遠隔 Sun MBM システムは、ローカルノードの 1 つとして構成されている必要があります。

遠隔 Sun MBM ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

注 – この -N オプションは、ローカルおよび遠隔 Sun MBM システムの両方に使用権限を持つユーザーがコマンドをサブミットした場合にだけ機能します。

例

- 次の例は、クラス a とクラス c からジョブを実行するためのアクティビティーの作成方法を示します。
- ジョブ proacc にクラス c を割り当ててサブミットします。
- このアクティビティーでは、クラス a およびクラス c のジョブを実行できます。
- infact コマンドにより、ジョブが atac アクティビティーに割り当てられたことが示されます。

```
[node1/fs1] # crtact atac -c ac
BS1052 activity atac is waiting for a job
 .
 .
 .

[node1/fs1] # subjob proacc -c c
BS1100(I) Job proacc submitted :jon = 34 class = c
```

```
[node1/fs1] # infact
BS1065(I) Command infact
BS1070(I) - Inform on activity name :
act1 atac (J034)
BS1038(I) infact command executed
```

crtflm - File_Map エントリの作成

crtflm コマンドは、File_Map 内のエントリを作成、変更、または削除します。-C、-M、および -D はそれぞれ、File_Map 内のエントリを作成、変更、および削除するように指示するオプションです。デフォルトは -C です。

crtflm コマンドによる File_Map エントリの変更中は、このファイルがロックされます。したがって、あるユーザーが加えた変更内容を上書きすることなく、複数のユーザーが同一の File_Map にアクセスできます。

形式

crtflm -C *file path* [*catalog*] [-t *type*] [-d *DDS*] [-g *GDN*]

crtflm -D *file* [*catalog*]

crtflm -M *file catalog* [*path*] [-t *type*] [-d *DDS*] [-g *GDN*]

説明

-C

File_Map 内にエントリを作成します (デフォルト)。

catalog

IBM メインフレームのカタログ名。44 文字以内の名前を指定します。カタログ名を指定しなかった場合、デフォルトのカタログ名 MASTERCAT が使用されます。

-d *DDS*

Sun MBM ユーティリティーおよび Open PL/I プログラムに使用されるファイル属性が含まれるファイルを指定する順編成ファイルの完全なパス名。File_Map の詳細は、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

-D

File_Map 内のエントリを削除します。

ファイル

64 文字以内の IBM メインフレームデータセット名。IBM JCL ストリームに指定されているとおりに指定します。

`-g` *GDN*

そのファイルに対して生成されたデータセットの最大数を示す、2 桁以内の GDG (世代別データグループ) 番号。最大値は 99 です。このエントリが `0` の場合、データセットは GDG に属していません。デフォルトは `0` です。

`-M`

`File_Map` 内のエントリを変更します。

path

IBM メインフレームのデータセットの割り当て先になる、最大 256 文字のシステムファイル名または Sun MTP VSAM データセット名。この引数は、ファイルタイプ `VS` (VSAM データセット) および `FS` (システムファイル) の場合には必ず指定します。

`-t` *type*

IBM ファイルが割り当てられるファイルのタイプ。

`FS`: システムファイル (VS 以外のすべてのファイルタイプ)

`VS`: Sun MTP VSAM データセット

例

`File_Map` 内に次のエントリが生成されているものとします。

```
A.B.C.D;MASTERCAT;FS;/users/vsam1/a/b/c/d;;0;
```

このエントリについて、次の `crtflm` コマンドを実行します。

```
[node1/fs1] # crtflm -M A.B.C.D MASTERCAT /users/vsam2/ab/c/d -t VS
```

`File_Map` エントリは次のように変更されます。

```
A.B.C.D;MASTERCAT;VS;/users/vsam2/a/b/c/d;;0;
```

dltact - アクティビティー (スレッド) の削除

dltact コマンドは、crtact コマンドによって作成されたアクティビティー (スレッド) を削除します。対象アクティビティー内で現在実行中のバッチジョブがある場合、そのジョブが完了するまでアクティビティーは削除されません。アクティビティーをただちに削除する必要がある場合は、まず dltact コマンドを実行してアクティビティーの削除を指示してから、abtjob コマンドを実行してジョブを強制的に中止させます。すべてのユーザーが任意のアクティビティーを削除できます。

注 – これはユーザーが使用できるコマンドですが、ジョブクラスを構成するときは必ず BAM を使用してください。dltact を使用してアクティビティーを削除した場合、BAM の GUI の使用時にジョブクラスに関する誤った情報が表示されることがあります。ジョブクラスを削除する方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

形式

dltact *name* [-N *nodename*]

説明

name

アクティビティー (スレッド) の名前。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上で実行中のバッチジョブがないアクティビティーを削除します。

nodename は、ローカルの Sun MBM システム内に、遠隔ノードとして定義されている必要があります。また、その遠隔 Sun MBM システムは、ローカルノードの 1 つとして構成されている必要があります。

遠隔 Sun MBM ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

例

- 次の例は、現在ジョブが実行されているアクティビティーの削除方法を示します。
- アクティビティーを削除します。
- そのアクティビティー内で実行中のジョブを強制的に中止させます。

```
[node1/fs1] # crtact atac -c ab
BS1052(I) activity atac is waiting for a job

[node1/fs1] # subjob proacc -c b
BS1100(I) Job proacc submitted :jon = 36 class = b

[node1/fs1] # infact
BS1065(I) Command infact
BS1070(I) - Inform on activity names :
atac (J036) act1
BS1038(I) infact command executed

[node1/fs1] # dltact atac
BS1055(I) activity atac terminate pending
[node1/fs1] # infact
BS1065(I) Command infact
BS1070(I) - Inform on activity names :
atac (J036) act1
BS1038(I) infact command executed

[node1/fs1] # abtjob 36
BS1046(I) job aborted (jon 36)

[node1/fs1] # infact
BS1065(I) Command infact
BS1070(I) - Inform on activity names :
act1
BS1038(I) infact command executed
```

dltjcl - 指定したジョブクラスに属しているジョブの削除

dltjcl コマンドは、指定したジョブクラスからジョブを削除します。ユーザーが削除できるのは、そのユーザー自身がサブミットしたジョブだけです。ほかのユーザーによってサブミットされたジョブは影響を受けません。Sun MBM 管理者は、どのユーザーがサブミットしたジョブでも削除できます。

形式

dltjcl *class* [-u'*all'|-u *user*] [-N *nodename*]

説明

class

a 〜 z のジョブクラス。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノードで、指定したクラスに割り当てられているジョブのうち、実行中でないジョブを削除します。

nodename は、ローカルの Sun MBM システム内に、遠隔ノードとして定義されている必要があります。また、その遠隔 Sun MBM システムは、ローカルノードの 1 つとして構成されている必要があります。

遠隔 Sun MBM ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

-u'*all'

すべてのユーザーのジョブを削除します。このオプションは Sun MBM 管理者だけが使用できます。

-u *user*

ジョブをサブミットしたユーザーのユーザー ID。Sun MBM 管理者は、-u オプションを使用して、指定したジョブクラスに属しているジョブのうち、特定のユーザー ID のジョブを削除できます。

例

- 次の例では、クラス c 用のアクティビティーをまだ作成していないものとします。
- ジョブ proacc にクラス c を割り当ててサブミットします。
- デフォルトのサブシステムを使用します。
- lstjcl コマンドにより、ジョブがクラス c に割り当てられたことが示されます。
- infact コマンドにより、現在有効なアクティビティーはあるが、クラス c 用のアクティビティーはないことが示されます。
- dltjcl コマンドを実行します。
- lstjcl コマンドにより、クラス c に属していた待機中のジョブが削除されたことが示されます。

```
[node1/fs1] # subjob proacc -c c
BS1100(I) Job proacc submitted :jon = 33 class = a

[node1/fs1] # lstjcl
Jon    Jobname   P C D     Strtime Endtime  Start from Status     Type   M   V
033    prova     5 c d       -       -      BeginJob    Submitted  Std    n  n
BS1038(I) lstjcl command executed

[node1/fs1] # infact
BS1065(I) Command infact
BS1070(I) - Inform on activity names :
actb
BS1038(I) infact command executed

[node1/fs1] # dltjcl c
BS1038 dltjcl command executed

[node1/fs1] # lstjcl c
BS1081(I) command lstjcl
BS1038(I) stjcl command executed
```

dltjob - ジョブの削除

dltjob コマンドは、*jon* で指定したジョブを SYSIN ファイルから削除します。該当のジョブが実行中の場合は、そのことを示すメッセージが出力され、ジョブは削除されません。

ディスポジションコード k を指定してジョブをサブミットすると、そのジョブの状態は SYSIN ファイルに保存されます。dltjob コマンドを実行すると、ジョブおよびその状態が SYSIN ファイルから削除されます。

形式

dltjob *jon* [-n *name*] [-N *nodename*]

説明

jon

ジョブのオカレンス番号。

-n *name*

指定したジョブの *name* と *jon* とが間違いなく関連付けられているかどうかを確認します。関連付けられていない場合はエラーメッセージが表示され、その他の処理は実行されません。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノードで、SYSIN ファイル内の実行中でないジョブのうち、*jon* で指定したジョブを削除します。

nodename は、ローカルの Sun MBM システム内に、遠隔ノードとして定義されている必要があります。また、その遠隔 Sun MBM システムは、ローカルノードの 1 つとして構成されている必要があります。

遠隔 Sun MBM ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

このコマンドは、自分のユーザー ID でサブミットしたジョブだけに有効です。別のユーザー ID に属しているジョブは削除できません。ただし、Sun MBM 管理者は任意のジョブを削除できます。

例

- 保持ディスポジションコード h を指定してジョブ proacc をサブミットします。
- ジョブ proacc を削除します。

```
[node1/fs1] # subjob proacc -d h
BS1100(I) Job proacc submitted :jon = 35 class = a
 .
 .
[node1/fs1] # dltjob 35 -n proacc
BS1047(I) Job proacc deleted
```

dostrans - VSE JCL からマクロジョブスクリプトへの変換

dostrans コマンドは、IBM VSE JCL ストリームを、Sun MBM マクロを含むジョブスクリプトに変換します。dostrans を使用すると、ジョブ制御文、ジョブ制御コマンド、および VSE POWER ジョブ入力制御文を含む VSE JCL を処理できます。

dostrans は、Sun MBM 環境変数 FILEMAP、FMROOT、PROCLIB、SLIDIR、および TRANSOPTS を使用します。これらの環境変数がサブシステムの設定ファイルに設定されていない場合、$USER_SETUP ファイルで設定します。batchenv シェルスクリプトの実行時にそのサブシステム名を引数として指定すると、環境変数が継承されます。

dostrans は、次のサブディレクトリがあるディレクトリから実行する必要があります。

jdos

入力ディレクトリ。変換する VSE JCL ジョブが含まれます。

dosp

入力ディレクトリ。変換する VSE JCL 手続きが含まれます。

sli

入力ディレクトリ。SLIDIR 環境変数が設定されていない場合に、ソースライブラリ組み込み (SLI) ファイルがデフォルトで使用されます。SLI ファイルは、ユーザーが 1 つ以上のジョブに組み込むことのできる、VSE JCL のセグメントです。

ish

出力ディレクトリ。jdos 内の入力ジョブファイルから変換されたジョブスクリプトが含まれます。また、dostrans によって生成され、接尾辞 .lst、.s、および .p の付いた中間ファイルも含まれます。

注 – 変換された手続きの出力用ディレクトリは、サブシステム作成時に設定される PROCLIB 環境変数の値によって決まります。

---

dostrans コマンドには、次の 3 つの処理フェーズがあります。

- SLI ファイルの組み込み

  最終出力ファイルと同じパス名に接尾辞 .s を付けたファイルを作成します。このファイルには、入力ファイルからの JCL 文および参照される SLI メンバーからの JCL 文が含まれます。

- JCL のプリプロセス

  .s ファイル内の継続およびコメント機能を検査します。Sun MBM 環境に適さない、つまりサポートされない JCL はすべて破棄され、警告メッセージが発行されます。このフェーズでは、最終出力ファイルと同じパス名に接尾辞 .p を付けたファイルが生成されます。

- JCL の変換

  .p ファイルの JCL を、マクロとバッチシェルコマンドを含むジョブスクリプトに変換します。

ジョブを生成するために dostrans を使用した場合、このコマンドは File_Map にアクセスし、VSE データセットおよびライブラリに割り当てられているパス名を取得できます。そのためには、出力の生成時にパス名のエントリが存在している必要があります。パス名のエントリは、dostrans に -v オプションを指定することによって生成できます。File_Map については、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

TRANSOPTS 環境変数が TRANSOPTS="-c -v" と設定されている場合、コマンド行に引数を指定しないで dostrans を実行すると、変換処理は妥当性検査モードで行われ、ファイル名は小文字に変換されません。

変換中、出力不能な文字が parm 文字列に含まれている場合、それらの文字はすべて /x*HH* という形式に変換されます。 *HH* はその文字の 16 進数の値です。

形式

dostrans {*filename* | *'*'* | *'xxx*'*} [-b *nnn*] [-c] [-D] [-e] [-E] [-f] [-l] [-L*n*] [-m] [-n] [-o *output-filename*] [-p] [-s] [-S |-P|-I] [-u] [-v] [-V]

説明

filename

変換する jdos サブディレクトリの VSE JCL ジョブファイルまたは dosp サブディレクトリの VSE JCL 手続きファイルのファイル名。

'*'

jdos サブディレクトリのすべての VSE JCL ジョブファイルまたは dosp サブディレクトリのすべての VSE JCL 手続きファイルをジョブスクリプトに変換する場合に指定します。

'*xxx**'

jdos サブディレクトリに含まれ、「*xxx*」に指定した文字で始まるすべての VSE JCL ジョブファイル、または dosp サブディレクトリに含まれ、「*xxx*」で指定した文字で始まるすべての VSE JCL 手続きファイルを変換する場合に指定します。

-b *nnn*

dostrans 出力に生成されるマクロ呼び出し間に、*nnn* で指定した数の空行をセパレータとして挿入します。デフォルトは 1 です。

-c

filename の値を大文字または小文字で File_Map 内に保持します。次に例を示します。

```
//DD1  DD  DSN=AA.BB.cc.DD
```

-c を指定した場合の File_Map エントリ

```
AA.BB.cc.DD;MASTERCAT;FS;/tmp/files/AA/BB/cc/DD;;0;
```

-c を指定しない場合の File_Map エントリ

```
AA.BB.cc.DD;MASTERCAT;FS;/tmp/files/aa/bb/cc/dd;;0;
```

-D

インストリームデータ内のドル記号 ($) を \$ に変換しません。このオプションを使用しない場合、デフォルトの dostrans 動作は、$ を \$ に変更します。

注 – -D オプションを使用すると、非標準メインフレーム動作になります。

-e

ジョブの実行時に、File_Map から *filename* を抽出します。

-E

インストリームデータ内のアンパサンド記号 (&) を \$ に変換します。-E オプションを使用しない場合、デフォルトの dostrans 動作は、& を \& に変更します。

注 – -E オプションを使用すると、非標準メインフレーム動作になります。

-f

ジョブスクリプト出力の再生成を指示します。指定した入力ファイルに対応する出力ファイルがすでに生成されていても、そのファイルは上書きされます。

-l

展開された入力 JCL リストが含まれるファイルを保持します。このファイルは SLI ファイルの組み込みフェーズで生成されます。JCL プリプロセスフェーズと JCL 変換フェーズで表示されるメッセージが示す行番号はすべて、このファイルの内容に関連付けられています。

-L *n*

File_Map 内に生成されるファイル名が適用されるディレクトリの数を制限します。

例

この例では、次の JCL 文で -L オプションを指定した場合と指定しなかった場合に、dostrans がどのように変換を行うかを示します。

```
//TEST DD DSN=AA.BB.CC.DD.EE
&JOBPARM
/*
```

-L 0 オプションを指定した場合

```
filename=${FMROOT}/aa.bb.cc.dd.ee
```

-L 1 オプションを指定した場合

```
filename=${FMROOT}/aa/bb.cc.dd.ee
```

-L オプションを指定しなかった場合

```
filename=${FMROOT}/aa/bb/cc/dd/ee
```

-m

DLBL 文に対する File_Map エントリを、タイプ VS として作成します。デフォルトは FS です。このオプションは -v オプションと併用した場合にだけ有効です。

`-n`

変換の実行日付を示すタイムスタンプ値を、生成された出力ファイルから除外します。このオプションは、異なる日時に生成された出力ファイルを比較する場合に指定します。

`-o` *output-filename*

1 つの入力ファイル名の変換時の出力先である出力ファイルに、*output_filename* を指定します。*output_filename* の値には、パス名でなくファイル名を指定する必要があります。`dostrans` は、次のディレクトリのいずれかの下にファイルを作成します。

- ジョブファイルの場合は、`ish` (`$JCLLIB`) サブディレクトリ
- 手続きファイルの場合は、`ishp` (`$PROCLIB`) ディレクトリ

`-p`

入力ファイル名が VSE JCL 手続きであることを示し、VSE JCL 手続きサブディレクトリ `dosp` を検索するよう `dostrans` に指示します。

`-s`

`dostrans` によって生成されるすべての警告メッセージを抑止します。

`-S | -P | -I`

`-S`: SLI ファイルから JCL を組み込むための SLI ファイル組み込みフェーズだけを実行します。このオプションは、SLI ファイルの組み込み後にジョブストリームの分析が必要な場合に使用します。

`-P`: JCL プリプロセスフェーズだけを実行します。SLI 組み込みフェーズで生成された出力ファイル (接尾辞 `.s`) が、次のいずれかのディレクトリに含まれている必要があります。

入力ファイルがジョブファイルの場合は、`ish` (`$JCLLIB`) サブディレクトリ

入力ファイルが手続きファイルの場合は、`ishp` (`$PROCLIB`) ディレクトリ

`-I`: JCL 変換フェーズだけを実行します。JCL プリプロセスフェーズで生成された出力ファイル (接尾辞 `.p`) が、次のいずれかのディレクトリに含まれている必要があります。

入力ファイルがジョブファイルの場合は、`ish` (`$JCLLIB`) サブディレクトリ

入力ファイルが手続きファイルの場合は、`ishp` (`$PROCLIB`) ディレクトリ

`-u`

`dostrans` オプションの構文を表示します。

`-v`

入力 VSE JCL ストリームファイルの妥当性検査を実行します。変換後の出力は生成されません。

妥当性検査モードでは、dostrans は FILEMAP 環境変数で定義された File_Map ファイルにエントリを自動生成します。ただし、dostrans コマンドを実行する場合は、-v を指定するかどうかにかかわらず、$FILEMAP が既存のファイルを指示するよう事前に設定しておく必要があります。File_Map の詳細および $FILEMAP の設定方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

FMROOT 環境変数は、妥当性検査モードでの変換時にすべてのファイル名の先頭に付けられるパスを定義します。これにより、該当のファイル名に対応する VSE データセット名が生成されます。$FMROOT が設定されていない場合は、デフォルトのパス /tmp が使用されます。$FMROOT の設定方法については、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

-V

変換出力内に生成されているすべてのマクロ呼び出しに対し、冗長モードを有効化します。

例

PRODJOB1 という名前の VSE JCL ジョブを変換します。

```
[node1/fs1] # dostrans PRODJOB1 -f
DOS/VSE JCL Translator      V10.1-04/26/2005
(c) Copyright Sun Microsystems, Inc.

including PRODJOB1 SLI members from /users1/mbmuser/node10/prod/sli ...
DV0110(I) SLI Member MEMBER1 included.
SLI member processor: No warnings,  No errors,  1 members included

preprocessing PRODJOB1 ...
DV2304(W) Line=#015. Job Control Statement unsupported -  Statement=[MTC]
DV2304(W) Line=#016. Job Control Statement unsupported -  Statement=[EXTENT]
DV2102(I) Job PRODJOB1 preprocessed successfully.
preprocessor: 2 warnings,   No errors

generating output for PRODJOB1 ...
DV3316(W) Line=#012. SYSIN logical unit assigned to instream data for potential
program read.
DV3368(W) Line=#017. Space(s) in literal operand of EXEC statement converted to
underscore(s).
DV3368(W) Line=#018. Space(s) in literal operand of EXEC statement converted to
underscore(s).
DV3316(W) Line=#018. SYSIN logical unit assigned to instream data for potential
program read.
DV3368(W) Line=#022. Space(s) in literal operand of EXEC statement converted to
underscore(s).
DV3103(I) Job PRODJOB1 macro code generated successfully.
translator: 5 warnings,   No errors
```

例

次の例では、-D および -E オプションを使用することで JCL からマクロ文への変換がどのように影響を受けるかを示します。

元の JCL

```
//JOB tstinstop
//EXEC PGM=IDCAMS,PARM='TEST'
  $dollar test with -E
  &ampersand test with -D
/&
```

-D または -E オプションを指定せずに dostrans tstintop -f コマンドを実行すると、次のマクロ文になります。

```
BEGINJOB
ASSGNDD ddname='SYSIN' type='INSTREAM' << !
  \$dollar test with -E
  \&ampersand test with -D
!
EXECPGM pgmname='IDCAMS' stepname='STEP0001' parm='TEST'
ENDJOB
```

dostrans tstintop -f -E コマンドを実行すると、次のマクロ文になります。

```
BEGINJOB
ASSGNDD ddname='SYSIN' type='INSTREAM' << !
  \$dollar test with -E
  \${ampersand} test with -D
!
EXECPGM pgmname='IDCAMS' stepname='STEP0001' parm='TEST'
ENDJOB
```

dostrans tstintop -f -D コマンドを実行すると、次のマクロ文になります。

```
BEGINJOB
ASSGNDD ddname='SYSIN' type='INSTREAM' << !
  $dollar test with -E
  \&ampersand test with -D
!
EXECPGM pgmname='IDCAMS' stepname='STEP0001' parm='TEST'
ENDJOB
```

dostrans tstintop -f -D -E コマンドを実行すると、次のマクロ文になります。

```
BEGINJOB
ASSGNDD ddname='SYSIN' type='INSTREAM' << !
  $dollar test with -E
  \${ampersand} test with -D
!
EXECPGM pgmname='IDCAMS' stepname='STEP0001' parm='TEST'
ENDJOB
```

ebmdate - Sun MBM の現在日時の取得

ebmdate コマンドは、Sun MBM の現在日時を返します。デフォルトでは、ebmdate は形式設定指示 %c に基づいて Sun MBM 日付を出力します。このコマンド出力の日付には、EBM_DATE_FORMAT および LC_TIME 環境変数の形式が適用されます。次も参照してください。

- 『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』
- 『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』

形式

```
ebmdate
```

例

曜日名、月名、および 12 時間表示による時刻が表示される日付形式です。

```
export EBM_DATE_FORMAT=" %A %B %d %I:%M:%S %p %Y "
```

次の例は、ebmdate の出力を示します。

```
[node1] # ebmdate
** Monday May 02 02:53:05 AM 2005 **
```

ebminfo - 実行環境に関する情報の取得

ebminfo コマンドは、*install-dir* ディレクトリにインストールされているノードまたは遠隔ノードの実行環境について、コマンドを実行した時点での情報を取得するために使用します。

オプションを指定しないで ebminfo を実行した場合、そのプロセスはノード内で実行されているプロセスのプロセス ID (PID) 値を戻します。ebminfo によって取得された情報から、ノードが実行中かどうかを確認できます。

注 – ebmsys コマンド出力の日付には、EBM_DATE_FORMAT (オプション -D を指定した場合) および EBM_TIME_FORMAT (オプション -T を指定した場合) の形式が適用されます。

形式

ebminfo [[-D] [-N *nodename*] [-p | -r | -s] [-S] [-T]]

-D

ノードの現在日付を表示します (このオプションは、-N オプションで指定した遠隔ノードについても有効です)。

-N *nodename*

指定した遠隔 Sun MBM ノードが実行中であれば、そのノードに関するすべてのプロセス情報が表示されます。このオプションは、遠隔ノードに関する実行情報を取得するために、ほかのすべてのオプションと併用できます。

nodename には、ローカルノードからのコマンドを実行できる遠隔ノードを指定します。

nodename は、ローカルの Sun MBM システム内に、遠隔ノードとして定義されている必要があります。また、その遠隔 Sun MBM システムは、ローカルノードの 1 つとして構成されている必要があります。

遠隔 Sun MBM ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

-p

指定したノードが実行中であれば、そのノードに関する ebmmd デーモンプロセスの PID が表示されます。

-r

指定したノードがクライアントノードからの要求を処理する準備ができていれば、このオプションによってソケットポート番号とホスト名が表示されます。

-s

指定したノードのメッセージキューマネージャー ebmmd が要求を処理する準備ができていれば、そのノードのソケットファイル名が表示されます。

-T

ノードの現在時刻を表示します (このオプションは、-N オプションで指定した遠隔ノードについても有効です)。

-S

指定したノードが最後に起動した日時とともに、それが通常の起動だったか、変更された日付が使用されたかを示す説明が表示されます。

例

ホストマシン (machine-1) 上に 2 つのノードがインストールされ、現在動作中です。ローカルノードは Node1、遠隔ノードは Node2 です。ebminfo により、各ノードに関する数種類の情報が表示されます。

```
[node1] # ebminfo -s
BS0529(I) Local socket of ebmmd daemon for the batch node :/node1/sockets/m62d44bb8

[node1] # ebminfo
BS0532(I) Process list for the batch node Node1:
Fd      Process Name    Pid     Host Name
4       ebmmd           20981   machine-1
5       ebminfo         23335   machine-1
6       lgdem           20996   machine-1
7       vcf             21000   machine-1
8       psg_daemon      21003   machine-1
9       bqm             21011   machine-1

[node1] # ebminfo -p
BS0530(I) Pid of ebmmd daemon for the batch node Node1 : 20981

[node1] # ebminfo -r
BS0518(I) Network socket of ebmmd daemon for the batch node Node1 :
Host Name   Port Number
machine-1     3001
```

```
[node1] # ebminfo -N Node2
BS0532(I) Process list for the batch node Node2:
Fd      Process Name    Pid     Host Name
4       ebmmd           22690   machine-1
5       ebminfo         23605   machine-1
6       lgdem           22704   machine-1
7       vcf             22709   machine-1
8       psg_daemon      22712   machine-1
9       bqm             22716   machine-1
```

次の例は、日時が変更された場合の出力です。

```
[node1] # ebminfo -S
BS0582(I) Batch node startup date:Tue May 03 08:11:32 2005
OS1047(I) Batch node date has been modified by:(+) seconds=19
```

次の例は、絶対日時が設定された場合の出力です。

```
[node1] # ebminfo -S
BS0582(I) Batch node startup date:Mon May 02 23:59:00 2005
OS1056(I) Batch node startup date has been set to:Mon May 02 23:59:00 2005
```

次の例は、通常の起動に関する出力です (日時の変更なし)。

```
[node1] # ebminfo -S
BS0582(I) Batch node startup date:Tue May 03 08:11:32 2005
```

ebmsnap - システムのスナップショットの作成

ebmsnap コマンドは、サポート担当者による問題分析に役立つシステムスナップショットを作成するために使用します。Sun MBM ノードが Sun MTP 領域に関連付けられている場合、ebmsnap は Sun MTP kixsnap ユーティリティーも実行します。

2 つの Sun MBM 環境変数を使用すると、特定のディレクトリの内容をスナップショットファイルに含めるかどうかを指定できます。これらの環境変数は、$USER_SETUP ファイルであらかじめ設定することも、ebmsnap コマンドを実行する直前にコマンドプロンプトで指定することもできます。

環境変数	説明
EBMSNAP_GETISH	サブシステムの ish ($JCLLIB) ディレクトリおよび ishp ($PROCLIB) ディレクトリの内容を収集します。この変数は、任意の値に設定できます。
EBMSNAP_GETHIST	システムで使用可能になっている場合、現在の日付のジョブ履歴ファイルの出力を収集します。この変数は、任意の値に設定できます。

注 – これらのディレクトリの内容には自社の機密事項が含まれていることがあるので、この環境変数のいずれか 1 つまたは両方を設定する際は注意が必要です。これらのディレクトリの内容をスナップショットに含めると、サービスプロバイダが情報を参照することを暗黙的に許可することになります。

ebmsnap ユーティリティーは、一時ディレクトリか指定したディレクトリのどちらかにファイルを生成します。スナップショットファイルを /tmp ディレクトリに書き込む場合、ファイル名の形式は次のようになります。

/tmp/batchsnapdir.*userid*/batchsnapfile.*mmdd_hhmmss*.tar.Z

userid は、ebmsnap ユーティリティーを実行したユーザーのユーザー ID です。日付および時刻のタイムスタンプは、スナップショットを実行した月、日、時、分、秒を表します。

▼ システムのスナップショットを作成する

1. ノードを所有するユーザーとしてログインします。
2. batchenv ファイルを実行します。
3. 次のいずれかの方法で、スナップショットを作成します。
   - コマンドプロンプトから ebmsnap を入力します。
   - 次の手順に従います。

   a. BAM を起動します。

   b. BAM メインメニューで「Problem Determination」→「Create a Snap Shot」を選択します。

形式

ebmsnap [-d *output-directory*] [-t *temp-directory*]

説明

-d *output-directory*

スナップショットファイルが作成される出力用ディレクトリ。デフォルトは、$TMPDIR (設定されている場合) または /tmp です。

-d *temp-directory*

一時スナップショットファイルが作成されるディレクトリ。ebmsnap が正常に終了した場合、作成された一時ファイルはすべて削除されます。デフォルトは、$TMPDIR (設定されている場合) または /tmp です。

ebmsys - サブシステムの作成と管理

ebmsys コマンドは、サブシステムを作成するために使用します。このコマンドでは、次のタイプのサブシステムを作成できます。

- Sun MTP システム
- RDBMS
- COBOL システム

注意 – これはユーザーが使用できるコマンドですが、サブシステムを作成するときは必ず BAM を使用してください。

サブシステムは、アプリケーションプログラムがアクセスするファイルのタイプに基づいて作成します。たとえば、Sun MTP VSAM ファイルにアクセスするアプリケーションプログラムを使用する場合は、Sun MTP 領域を定義して、その領域固有の FCT (ファイル管理テーブル) と VSAM カタログ内に VSAM ファイルを定義する必要があります。リレーショナルファイル (Oracle、Sybase、または DB2) にアクセスするアプリケーションプログラムを使用する場合は、Sun MBM に対して適切な RDBM 実行時システムを定義する必要があります。1 つのノードに対して最大 8 つのサブシステムを定義できます。

作成したサブシステムは、Sun MBM 内部のサブシステムテーブルに登録されます。その *batchsystem-name* を `-k` オプションを指定して `subjob` または `unikixjob` を発行すると、どのサブシステムに対してジョブをサブミットするかを指示できます。

形式

`ebmsys[-a] [-c` *batchsystem-name* `[-u` *user-name* `| -g` *group-name*`]] [-d` *batchsystem-name*`] [-i` *batchsystem-name*`] [-l` *batchsystem-name* `[-a]]` **[**`-N` *nodename*`] [-r` *batchsystem-name*`] [-s` *batchsystem-name*`] [-t]`

説明

オプションを指定しない場合

デフォルトのサブシステムが定義されている場合は、そのサブシステムの状態が表示されます。

`-a`

内部のサブシステムテーブルに定義されているすべてのサブシステムが表示されます。各サブシステムには、作成済み、削除済みのどちらかを示す状態コードが設定されます。表示される情報は、サブシステムテーブル内の最後の 8 つのエントリだけです。それらの情報は `-i` オプションの場合と同じ形式で表示されます。`-a` オプションを指定した `ebmsys` コマンドは、`ebmsys_authority_file` の設定値に関係なく、どのユーザーでも発行できます。

`-c` *batchsystem-name*

ノードの *install-dir*`/.install` ファイルに、新しいサブシステムを作成または定義します。*batchsystem-name* は 1 ～ 8 文字の英数字です。

`-u` *user-name*

指定した名前のユーザーだけが、対象サブシステムにジョブをサブミットできます。

`-g` *group-name*

指定したグループに属しているユーザーだけが、対象サブシステムにジョブをサブミットできます。

batchsystem-name は、有効な設定ファイルに設定する必要があります。たとえば、`.install` ファイルには次のようなエントリが必要です。

```
kixsys1=/batch/mynode/ebmsys/cobsys1.sf
```

説明

`cobsys1.sf` は、対象サブシステムに関する環境変数の定義が含まれる設定ファイルです。

`-d` *batchsystem-name*

サブシステムを削除します。このオプションを指定して削除したサブシステムには、それ以降ジョブをサブミットできません。サブシステムを削除できるのは、そのサブシステムの所有者だけです。

注 – 対象サブシステム用の `SYSIN` ファイル内にジョブが含まれている場合、このコマンドは強制的に中止されます。

`-i` *batchsystem-name*

batchsystem-name の状態を表示します。`-i` オプションを指定した `ebmsys` コマンドは、`ebmsys_authority_file` の設定値に関係なく、どのユーザーでも発行できます。

`-l` *batchsystem-name* [`a`]

このコマンドを発行するユーザーに属しているすべてのジョブの状態が表示されます。このオプションを `-a` オプションとともに指定すると、指定したサブシステムに属しているすべてのジョブの状態が表示されます。

`-l` オプションを指定した `ebmsys` コマンドは、`ebmsys_authority_file` の設定値に関係なく、どのユーザーでも発行できます。ただし、`ebmsys_authority_file` に登録されていないユーザーは、`-a` オプションと `-l` オプションの併用はできません。

`-r` *batchsystem-name*

デフォルトのサブシステムをデフォルト以外の状態にリセットします。デフォルトのサブシステムをリセットできるのは、そのサブシステムの所有者だけです。

別のデフォルトサブシステムを設定するには、まず現在のデフォルトサブシステムをデフォルト以外の状態にリセットしておく必要があります。

`-t`

作成されたすべてのサブシステムの状態を表示します。`-t` オプションを指定した `ebmsys` コマンドは、`ebmsys_authority_file` の設定値に関係なく、どのユーザーでも発行できます。

-s *batchsystem-name*

デフォルトのサブシステム。このオプションを使用した場合は、subjob または unikixjob コマンドの発行時に -k オプションでサブシステムを指定しなくてもかまいません。サブシステムの指定がないジョブは、デフォルトのサブシステムにサブミットされます。該当のサブシステムを所有するユーザーだけが、このデフォルトオプションを指定できます。

注 – 必ず既存のサブシステムの名前を指定してください。

-N *nodename*

指定した遠隔 Sun MBM ノードが実行中であれば、そのノードに関するすべての情報が表示されます。

nodename は、ローカルの Sun MBM システム内に、遠隔ノードとして定義されている必要があります。また、その遠隔 Sun MBM システムは、ローカルノードの 1 つとして構成されている必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

この設定ファイルには、対象サブシステムに関連するすべての環境変数の定義が含まれます。任意の環境変数がバッチジョブに継承されるようにするには、それらの環境変数を設定ファイルに定義しておく必要があります (設定ファイルに読み込み権が必要)。設定ファイルの詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

例

-i オプションを使用すると、次の出力が得られます。

```
[node1] # ebmsys -i fs1
#Batch Subsystem name:fs1
#Time of creation:Wed Apr 27 10:00:41 2005
#User:mbmuser
#Terminal:
#Display:machine-1:8.0
#Restriction type:none
#Setup file:/local_space/mbmuser/node1/bam/subsys/fs1/Setup
#KIXSYS:not set
#Check on KIXSYS is ok
#Std jobs:0 defined, 0 running
#VSAM jobs:0 defined, 0 running
#Processed jobs: 0
#This is the default Batch Subsystem.
```

説明

Time of creation

サブシステムの作成日時 (-c)。

User

サブシステムを所有するユーザーの名前。サブシステムの削除および状態変更ができるのは、そのサブシステムの所有者と Sun MBM 管理者だけです。

Terminal

ebmsys コマンドを発行した端末。

Display

ebmsys コマンドを発行した X 端末の名前。

Restriction type

このサブシステムにジョブをサブミットできるユーザーまたはユーザーグループ。none は、すべてのユーザーがこのサブシステムにジョブをサブミットできることを示します。

Setup file

このサブシステムに関連付けられている設定ファイルのパス名。アクセス上の問題を示すエラーメッセージが表示された場合、このファイルにはアクセスできないことを示します。

KIXSYS

このサブシステムの設定ファイルに定義されている KIXSYS 環境変数の値。

Check on KIXSYS

このサブシステムの設定ファイル内の KIXSYS 値が Sun MBM の管理する値と一致しているかどうかを判定するために実行された、内部チェックの結果。
Sun MBM は、サブシステムが作成されると、その設定ファイルから KIXSYS の値を取得します。

Std jobs

subjob コマンドによってサブミットされたジョブの数と現在実行中のジョブの数。

Sun MTP jobs

unikixjob コマンドによってサブミットされたジョブの数と、現在実行中のジョブの数。

Processed jobs

ノードが前回起動した時点またはこのサブシステムが作成された時点以降に、このサブシステムによって実行されたジョブの数。

動作中の Sun MTP サブシステムについて ebmsys -i コマンドを発行すると、追加の情報が表示されます。

```
[node1] # ebmsys -i vs1

#Batch Subsystem name:vs1
#Time of creation:Apr 27 17:37:05 2005
#User:mbmuser
#Terminal:/dev/pts/0
#Display:machine-1:8.0
#Restriction type:only for user pax
#Setup file:/users2/pkgs/node2/bam/subsys/vs1/Setup
#KIXSYS:/users2/pkgs/finance/sys
#Check on KIXSYS is ok
#Std  jobs:0 defined, 0 running
#VSAM jobs:0 defined, 0 running
#Processed jobs: 0
#This is not the default Batch Subsystem
@unikixstrt pid: 17833
@Sun MTP user:mtpuser
@Sun MTP terminal:/dev/pts/4
@Sun MTP display:machine-1:0.0
@Sun MTP startup time:Apr 27 17:58:47 2005
@Last Sun MTP request:Apr 27 18:01:17 2005
@Requested jobs: 2
@Resolved jobs: 0
@Sun MTP version: 8.1.0 - 2005/06/15
@COM version: 8.0.16-2005/04/26
@Protocol version: 8
```

説明

unikixstrt pid

このサブシステムの Sun MBM に接続されている、Sun MTP unikixstrt プロセッサのプロセス識別子。

Sun MTP user

領域を起動したユーザーの名前。

Sun MTP terminal

領域の起動に使用された端末。

Sun MTP display

領域の起動に使用された X 端末の名前。

Sun MTP startup time

領域の起動日時。

`Last Sun MTP request`

VSAM ジョブの場合、Sun MTP は領域側で実行可能なバッチジョブを Sun MBM から要求します。これは領域がこの種の要求を前回行った日付と時刻です。この日時は、VSAM 構成テーブル (VCT) に定義されているバッチ検索間隔に基づいています。

`Requested jobs`

領域が前回のジョブ要求以降に要求したジョブの数。この数は、VCT の設定パラメータ (最大バッチジョブ数) に応じて異なります。

`Resolved jobs`

Sun MBM が、要求側の領域に対してサブミットできるジョブの数。この数は、Sun MBM 内に設定されているアクティビティーの数に応じて異なります。

`Sun MTP version`

Sun MTP ソフトウェアのバージョン。

`COM version`

Sun MTP パッケージとともに配布された通信ソフトウェアのバージョン。

`Protocol version`

Sun MBM および Sun MTP パッケージのバージョンに関する内部チェックの結果が表示されます。

`ebmsys -l` コマンドを発行すると、そのユーザーが所有するジョブのうち、指定したサブシステムに属しているすべてのジョブが一覧表示されます。

```
[node1] # ebmsys -l ebmsys1

Jon    Jobname  JobIdentifier              User        Type   Status  D  C
671    JOB001   JOB001@02May200518:47:10   mbmuser     Std    Subm    h  a
672    JOB002   JOB002@02May200518:47:11   mbmuser     Std    Subm    h  a
673    JOB003   JOB003@02May200518:47:12   mbmuser     Std    Subm    h  a
674    JOB004   JOB004@02May200518:47:13   mbmuser     Std    Subm    h  a
675    JOB005   JOB005@02May200518:47:13   mbmuser     Std    Subm    h  a
676    JOB006   JOB006@02May200518:47:14   mbmuser     Std    Subm    h  a
```

説明

Jon

ジョブのオカレンス番号。

Jobname

サブミットされたバッチジョブの名前。

JobIdentifier

$JOBID 内に保持されている一意のジョブ ID。この値は、ユーザーが subjob または unikixjob コマンドの発行時に -J オプションで指定しない限り、自動的に生成されます。

User

ジョブをサブミットしたユーザーの名前。

Type

次のいずれかを指定します。

Std: subjob コマンドでサブミット

Sun MTP: unikixjob コマンドでサブミット

Status

次のうちいずれかを指定します。

Subm: Submitted (サブミット済み)

Run: Running (実行中)

Susp: Suspended (中断)

Abrt: Aborted (強制中止)

Fin: Finished (完了)

Rdy: Ready to run (実行可能)

D

ディスポジションコードは次のとおりです。

d: 指定したクラスのすべてのバッチジョブについて、実行のスケジューリングを行います。この値は通常、ジョブクラスに保持ディスポジションコードが定義されている場合に使用します。デフォルト値です。

h: 指定したクラスに割り当てられているジョブはスケジューリングも実行もされず、あとで処理されるまで SYSIN ファイル内に保持されます。

k: ジョブの実行がスケジューリングされ、実行後も SYSIN ファイル内のジョブまたはシェルスクリプトが保存されます。実行後のジョブのディスポジションは、保持に変更されます。

C

ジョブが割り当てられているクラス (a ～ z)。

ebmsys -t コマンドは、作成済みのすべてのサブシステムを一覧表示します。

```
[node1] # ebmsys -t
SubsystemName  Run_Jobs  MTP             MTP User   Last MTP Call
ebmsys1                  not available
ebmsys2                  not available
vssys                    connected       mtpuser     Apr 27 17:58:47
```

説明

Batch subsystem

ebmsys -c コマンドで指定したサブシステム名。

RunJobs

そのシステムで実行中のジョブの数。

MTP

領域の状態には、次のものがあります。

not available: Sun MTP 領域ではありません。

not connected: 領域は Sun MBM に接続されていません。

connected: 領域は Sun MBM に接続されています。

MTP User

領域を起動したユーザーの名前。

Last MTP Call

領域が前回 Sun MBM からジョブを要求した日時。

ebmsys コマンド出力の日付には、EBM_DATE_FORMAT 環境変数の形式が適用されます。ここでは、この環境変数に次の形式が設定されている場合の出力例を示します。

```
export EBM_DATE_FORMAT="** %A %B %d %I:%M:%S %p %Y **"
```

出力は次のようになります。

```
[node1] # ebmsys -i sys1
#Batch Subsystem name:sys1
#Time of creation:** Friday April 29 07:10:53 AM 2005 **
#User:mbmuser
#Processed jobs: 1
#This is the default Batch subsystem.
```

日付の形式設定指示の詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

ebmtime - Sun MBM 時刻の取得

このコマンドは、Sun MBM 時刻を取得します。デフォルトでは、Sun MBM はメインフレームのジョブの時刻の形式設定指示に従って時刻を表示します。このコマンド出力の日付には、EBM_TIME_FORMAT および LC_TIME 環境変数の形式が適用されます。現行の時刻形式を取得または変更する方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』および『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

形式

```
ebmtime
```

例

```
[node1] # ebmtime
09:00:43
```

ebmx - Sun MBM メインメニューの表示

ebmx コマンドは、X 端末上で図 4-1 にある Sun MBM メインメニューを表示するために使用します。

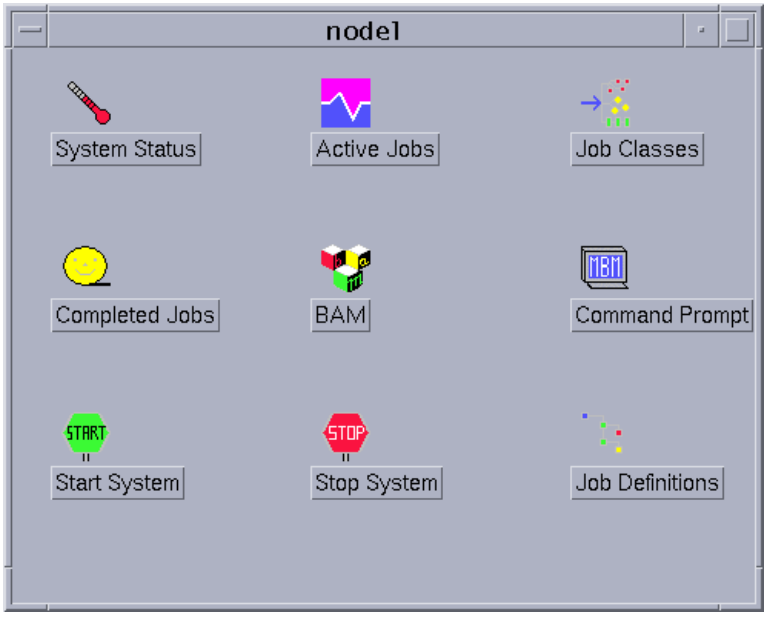

図 4-1　　Sun MBM メインメニュー

次のボタンをクリックすると、さまざまな Sun MBM 機能を使用できます。

- System Status

  動作中のサブシステムの状態を一覧表示します。サブシステムごとに、ユーザーはジョブをサブミットしたり、有効なジョブやキュー内のジョブを一覧表示したりできます。

- Active Jobs

  有効なバッチジョブを一覧表示し、特定のジョブの取り消し、中断、再開、および監視などの制御ができます。

- Job Classes

  ジョブクラスを監視し、クラスごとに有効なジョブと保留中のジョブを一覧表示します。

- Completed Jobs

  完了したジョブの状態と統計情報をレポートします。

- BAM

  Batch Administration Manager (BAM) のセッションを開始します。
- Command Prompt

  特定のサブシステム環境にログインします。
- Start System

  ノードを起動します。
- Stop System

  ノードを停止します。
- Job Definitions

  Job Editor を起動します。詳細は、第 3 章を参照してください。

形式

モノクロ表示の X 端末の場合

```
ebmx monochrome
```

X 端末にフォアグラウンド色を設定する場合

```
ebmx fg=color
```

X 端末にバックグラウンド色を設定する場合

```
ebmx bg=color
```

X 端末上でメインメニューを表示するには、`batchenv` を実行し、`ebmx` コマンドを入力します。

例

色表示の X 端末の例として、フォアグラウンド色を濃い青 (MidnightBlue)、バックグラウンド色を明るい灰色 (LightGrey) に指定してメインメニューを表示します。

```
$ cd node1
$ . ./batchenv
[node1] # ebmx fg="MidnightBlue" bg="LightGrey"
```

先行ドット文字と `./batchenv` との間には空白文字を挿入します。

ftval - File_Map への動的アクセス

ftval ユーティリティーは、ジョブの実行中に File_Map に動的にアクセスします。

形式

ftval *dataset-name catalog-name* [*catalog-name..*] *column*

説明

dataset-name

File_Map のエントリを検索するキー。

catalog-name

File_Map のエントリを検索するキー。

column

File_Map エントリの列。たとえば、*column* が 1 の場合、*dataset-name* が返されます。見つかった列は標準出力に送られます。

キーに一致する値がない場合またはエントリに列がない場合は、空白が返されます。

dataset-name と *catalog-name* を変数として定義することもできます。次に例を示します。

```
ftval $acctdata1 $acctcat1 4
```

データセットに &P などの記号参照が含まれる場合、MVS JCL トランスレータでは、バッチジョブで & の前にエスケープ文字 \ を生成します。このエスケープ文字は、データセットに関する情報を File_Map から取得するときに、ftval が & をバックグラウンド実行用のシステム演算子であると解釈しないようにするために生成されます。

histprt - 履歴ファイルの出力

histprt コマンドは、Sun MBM を最初に起動した時点から生成されたメッセージを出力するために使用します。メッセージ数の上限は循環履歴ファイルのサイズで決まります。ログファイルのサイズを変更する方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

注 – EBM_DATE_FORMAT と LC_TIME 環境変数は、histprt エントリ (histprt 出力の日付カラム) の日付/時間スタンプに影響を与えます。

形式

histprt [-a] [-d *mmddhhmmssyyyy*] [-j *jon*] [-N *nodename*] [-u *user*]

説明

-a

保存されているすべてのメッセージを一覧表示します。

-d *date*

指定した日付のメッセージに対してコマンド histprt を実行します。

形式

mmddhhmmssyyyy

説明

mm	月	01 〜 12
dd	日付	01 〜 31
hh	時刻 (24 時間表示)	00 〜 23
mm	分	00 〜 59
ss	秒	00 〜 59
yyyy	年	1970 〜 2030

-j *jon*

ジョブのオカレンス番号。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上の履歴ファイル内のメッセージを一覧表示します。

nodename は、ローカルの Sun MBM システム内に、遠隔ノードとして定義されている必要があります。また、その遠隔 Sun MBM システムは、ローカルノードの 1 つとして構成されている必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

-u *user*

14 文字以内で指定したユーザーに関するメッセージを一覧表示します。

-u または -j オプションを使用することにより、指定したユーザーまたはジョブ番号に該当するメッセージを出力できます。ユーザーが出力できるのは、自分のユーザー ID でサブミットしたジョブに関するメッセージだけです。Sun MBM 管理者は任意のバッチジョブに関するメッセージを一覧表示できます。

例

ジョブオカレンス番号が 47 のジョブに関する情報を出力します。

```
[node1] # histprt -j 47
BS1746(I) List of the History File.User ken
                                Job
Date                        User Number:
Mon May 02 05:14:16         mbm    47   :IP4305(I) Start of job plan at 02 May 2005:05:14:16
Mon May 02 05:14:16 2002    mbm    47   :IP4334(I) History Filename:/tmp/batch.....
Mon May 02 05:14:20 2002    mbm    47   :IP4303(S) Job plan aborting at 02 May 2005:05:14:20
```

説明

Date

メッセージが生成された日付。

Uid

ジョブをサブミットしたユーザーのユーザー ID 番号。

Job Number

メッセージに関連付けられているジョブオカレンス番号。

histprt コマンド出力には EBM_DATE_FORMAT 環境変数の値が適用され、この環境変数には次の日付形式が設定されているものとします。

```
export EBM_DATE_FORMAT="** %A %B %d %I:%M:%S %p %Y **"
```

日付の形式設定指示のリストについては、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

次に、histprt の出力例を示します。

```
[node1] # histprt
BS1746(I) List of the History File.User mbmuser.
Date   User    Job Number:
** Monday May 02 12:00:22 AM 2005 ** mbmuser 0:IP4305(I) Start of job job1 at Mon May 2
00:00:22 2005
** Monday May 02 12:00:22 AM 2005 **  mbmuser 0:IP4334(I) History
filename:/local_space/mbmuser/test/tmp/histdir/job1_0.hst.02May2005.00:00
** Monday May 02 12:00:25 AM 2005 **  mbmuser 0:IP4303(S) Job job1 aborting at Mon May 2
00:00:25 2005
BS1038(I) histprt command executed
```

注 – histprt 出力の日付には、EBM_DATE_FORMAT 環境変数の形式は適用されません。

infact - アクティビティー情報の取得

infact コマンドは、特定のアクティビティーまたはすべてのアクティビティーに関する情報を取得します。

形式

infact [*name*] [-N *nodename*]

説明

name

アクティビティーの名前。この引数を指定しない場合は、すべてのアクティビティーがレポートされます。アクティビティーの状態は、そのアクティビティーの作成時に使用されたユーザー ID に関係なくレポートされます。

レポートされる情報には、現在そのアクティビティー内で実行中のジョブに関する情報も含まれます。アクティビティー内に実行中のジョブがない場合は、そのアクティビティーの名前だけが出力されます。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上の特定のアクティビティーまたはすべてのアクティビティーに関する情報を表示します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

例

例では、次のことを示します。

- アクティビティー `act1` では、ジョブオカレンス番号 `26` のジョブを実行中です。
- アクティビティー `act6` では、ジョブオカレンス番号 `32` のジョブを実行中です。

```
[node1] # infact
BS1065(I)     Command infact
BS1070(I)     - Inform on activity name:
act1 (J026)   act2 act3 act4 act6 (J032)
BS1038     infact command executed
```

infjbs - 実行中のすべてのジョブに関する情報の取得

`infjbs` コマンドは、現在実行中のすべてのジョブに関する情報を表示します。その情報には、ジョブをサブミットしたユーザーの区別なく、指定した Sun MBM 環境で実行されているすべてのジョブの情報が含まれます。

形式

`infjbs [-N` *nodename*`]`

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上で実行されているジョブに関する情報を表示します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

例

次のジョブは、現在、バッチプログラムの STEP010 で、BAB15 プログラムを実行中です。このジョブの優先順位は 5、ランクは 0 (ゼロ) です。

```
[node1/fs1] # subjob test
BS1100(I) Job test submitted :jon = 98 class = a
[node1/fs1] # infjbs
BS1075(I) Command infjbs
Jon   Jobname  Stepname    JobTm    Cmd     CmdTm          User
Act   P C D  LCC      JobStatus  PLines    Job_Start       Cmd_Start      Rank
JobIdentifier                    Ebmsys   JobParameter
-------------------------------------------------------------------------------
098   test     STEP010     00m08s   BAB    1500m03s        mbmuser
act1  5 a d  000 000  Running    000003    02May2005 10:25 02May2005 10:25   0
test@02May2005:10:25:11           hazel    null
BS1038(I) infjbs command executed
```

説明

Jon

ジョブのオカレンス番号。

Jobname

unikixjob または subjob コマンドで指定されたファイル名。

StepName

現在実行中の *stepname.proc-stepname*。

JobTm

ジョブの実行経過時間。

Cmd

現在実行中のコマンド。

CmdTm

コマンドの実行経過時間。

User

ジョブをサブミットしたユーザーの名前。

Act

ジョブが実行されているアクティビティーの名前。

P

ジョブの優先順位。

C

ジョブが割り当てられているクラス (a 〜 z)。

D

ジョブのディスポジション。

d: 指定したクラスのすべてのバッチジョブについて、実行のスケジューリングを行います。この値は通常、ジョブクラスに保持ディスポジションコードが定義されている場合に使用します。デフォルト値です。

h: 指定したクラスに割り当てられているジョブはスケジューリングも実行もされず、あとで処理されるまで SYSIN ファイル内に保持されます。

k: ジョブの実行がスケジューリングされ、実行後も SYSIN ファイル内のジョブまたはシェルスクリプトが保存されます。実行後のジョブのディスポジションは、保持 (h) に変更されます。

LCC

ジョブに設定された最新の条件コード。STATUS 環境変数の値に対応しています。

JobStatus

2 つの状態 ($JOBSTATUS の値でジョブが実行中か強制中止中か) を示します。ジョブが中断された場合、このフィールドの値は変化しません。

ジョブの状態 (完了、強制中止、中断、サブミット済み、および実行中) を確認するには、lstjob コマンドを使用する必要があります。aborting という文字列が表示された場合、バッチジョブはすべての手順をバイパス中か、COND パラメータの ONLY または EVEN オプションの指定に従って手順を実行中です。

PLines

バッチジョブ中の処理済みの行数。この行数には、ジョブによって呼び出された手続きも含まれます。ただし、ジョブにはループ、分岐、または条件文も含まれるので、シェルスクリプトの行番号としては使用できません。

Job_Start

ジョブが開始された日付と時刻。

JobIdentifier

subjob または unikixjob コマンドの -J オプションによってジョブに割り当てられたジョブ識別子。

Ebmsys

ジョブのサブミット時に subjob または unikixjob コマンドの -K オプションによって指定されたサブシステムの名前。

JobParameter

subjob または unikixjob コマンドの -P オプションによってジョブに渡される文字列。

Cmd_start

現在のコマンドが開始された日付と時刻。

Rank

表示される各行に昇順で割り当てられている、ジョブ番号。

infjob - 実行中のジョブに関する情報の取得

infjob コマンドは、現在実行中の 1 つのジョブに関する情報を表示するために使用します。このコマンドにより、指定したジョブの名前および現在そのジョブで実行中のステップ名が出力されます。該当のジョブが実行中でない場合は、そのことを示すメッセージが出力されます。

形式

infjob [*jon*] [-N *nodename*]

説明

jon

ジョブのシーケンス番号。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上で実行中の指定したジョブに関する情報を表示します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

注 – ユーザーが取得できるのは、自分のユーザー ID でサブミットしたジョブに関する情報だけです。ただし、Sun MBM 管理者は任意のジョブの状態情報を取得できます。

例

`infjob` コマンドの実行ユーザーと同じユーザー ID でサブミットされたジョブの情報を取得します。

```
[node1/fs1] # infjob 32
BS1071(I)  - Command infjob - inform on running jobs
BS1072(I)  - Inform on job 32 :job = proacc      cmd = COB001
BS1038(I)  infjob command executed
.
.
.
```

別のユーザー ID でサブミットされたジョブの情報を取得しようとすると、次のような結果になります。

```
[node1/fs1] # infjob 33
BS1071(I)  Command infjob   - inform on running jobs
BS1003(S)  permission denied - infjob command aborted
```

insjbl - ジョブの検査

`insjbl` コマンドは、1 つまたは複数の待機中のジョブについて、その待機時と完了後の状態を確認します。待機中のジョブは `SYSIN` ファイル内に保持されます。このコマンドに指定したジョブが現在実行中の場合は、そのことを示すメッセージが出力されます。

形式

`insjbl` *jon1* [*jon2* . . . [*jon32*]] [`-N` *nodename*] [`-w`]

説明

jon1 [*jon2* ...[*jon32*]]

ジョブオカレンス番号 (最大 32)。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノードに対して SYSIN ファイル内で実行待機中のジョブに関する状態を表示します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

-w

指定したすべてのジョブが完了するまで監視を続けます。

ジョブが完了すると、そのジョブが正常に完了したか (CONDITION-CODE は 0)、あるいは強制的に中止されたか (CONDITION-CODE は 0 以外) を示すメッセージが表示されます。

注 – ユーザーが取得できるのは、自分のユーザー ID でサブミットしたジョブに関する情報だけです。Sun MBM 管理者は任意のジョブの情報を取得できます。

例

insjbl コマンドを使用して複数のジョブが完了するまで監視を続け、それらの状態を確認する方法を示します。これらのジョブが 1 つでも強制的に中止された場合、状態チェックの結果はゼロ以外の値になります。ジョブの状態コードは次のとおりです。

Run: 実行中

Abrt: Aborted (強制中止)

Susp: Suspended (中断)

Fin: Finished (完了)

Rdy: Ready to run (実行可能)

状態コードの行には、各ジョブの状態の変化がただちに表示されます。

```
[node1/fs1] # insjbl 206,207,208,209,210 -w
206     207     208     209      210
Run     Run     Run     Abrt     Abrt
Susp    Run     Run     Abrt     Abrt
Susp    Abrt    Run     Abrt     Abrt
Run     Abrt    Run     Abrt     Abrt
Fin     Abrt    Fin     Abrt     Abrt
[node1/fs1] # echo $?                                    (状態のチェック)
```

ジョブ数が多すぎるために、ジョブの状態コードをジョブ別のカラムに表示できない場合、次のように縦一列に表示されます。

```
[node1/fs1] # insjbl 206,207,208,209,210,211,212,213,214,215 -w
206 ---> Aborted
207 ---> Aborted
208 ---> Fin
209 ---> Aborted
210 ---> Run
211 ---> Aborted
212 ---> Fin
213 ---> Aborted
214 ---> Run
      .
      .
[node1/fs1] # echo $?
5           （強制中止されたジョブの数）
```

このコマンドでは、実行後 SYSIN ファイルから削除されたジョブについての情報は表示できません。

insjob - sysin 内のジョブの状態表示

insjob コマンドは、特定のジョブについて、サブミット済み、実行中、または終了済みの状態を表示するために使用します。実行待機中のジョブは sysin ファイル内に保持されます。このコマンドに指定したジョブが現在実行中の場合は、そのことを示すメッセージが出力されます。

形式

insjob *jon* [-N *nodename*] [-w]

説明

jon

ジョブのオカレンス番号。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上のジョブについて、サブミット済み、実行中、または終了済みの状態を表示します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

-w

待機中のジョブについて、その待機時と完了後の状態を取得します。ジョブが完了すると、そのジョブが正常に完了したか (CONDITION-CODE は 0)、あるいは強制的に中止したか (CONDITION-CODE は 0 以外) を示すメッセージが表示されます。

注 – ユーザーが取得できるのは、自分のユーザー ID でサブミットしたジョブに関する情報だけです。Sun MBM 管理者は任意のジョブの情報を取得できます。

例

例では、次のことを示します。

- ジョブ prog1 をサブミットします。
- 最初の例では、-w オプションを指定して insjob コマンドを実行し、ジョブが終了するまでの状態の変化を監視します。ジョブが終了するまで制御は戻りません。

■ 2 番目の例は、オプションを指定しないで insjob コマンドを発行した場合を示します。現在のジョブの状態が表示されたあとに制御が戻ります。

```
[node1/fs1] # subjob prog1
BS1100(I)     Job prog1 submitted :jon = 40    class = a
insjob 40 -w
BS1079(I) user dsmith:job 40 ---> running
BS1079(I) user dsmith:job 40 ---> finished
echo $?
1
 .
 .
[node1/fs1] # subjob prog2 -d h
BS1100(I) Job prog2 submitted :jon = 42    class = a
insjob 42
BS1079(I) user :job 42 ---> submitted
echo $?
0
```

insjob コマンドでは、ジョブが SYSIN ファイルに置かれている場合に限り、ジョブの状態を表示できます。

kixdate - ノードの日付の変更

このコマンドで表示される Sun MTP 日付/時刻の構成ユーティリティー (図 4-2) は、Sun MTP および Sun MBM システム用の日付と時刻を変更するための機能をユーザーに提供します。デフォルトでは、コンピュータのシステム時間が Sun MTP と Sun MBM に適用されます。

このユーティリティーを Sun MBM に対して使用するには、Sun MBM 環境を設定する必要があります。詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

このユーティリティーを Sun MTP に対して使用するには、KIXSYS 環境変数を設定する必要があります。このユーティリティーを Sun MTP で使用する方法については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

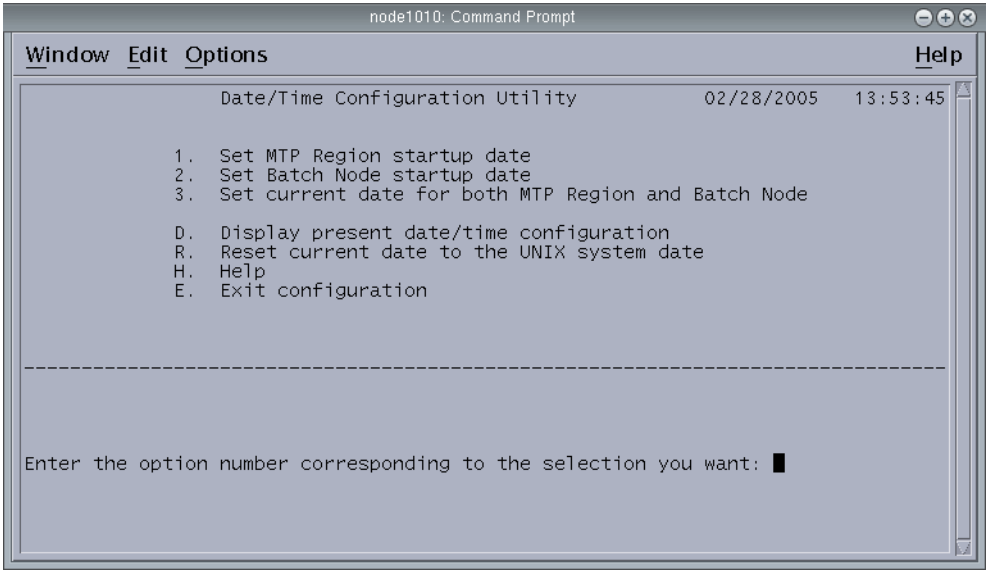

図 4-2　　日付/時刻の構成ユーティリティー (kixdate)

ユーティリティーを実行するには、次のコマンドを入力します。

```
[node1] # kixdate
```

メニューオプションの詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

lgprint - ログ情報の取得

lgprint コマンドは、admlog コマンドで定義された追跡ファイルから Sun MBM 内部追跡エントリを抽出し、その情報を $PUBLIC/msg/elog_print ファイルに書き込むために使用します。詳細は、103 ページの「admlog - ログファイルの管理」を参照してください。lgprint コマンドの実行後は、テキストエディタを使用して $PUBLIC/msg/elog_print ファイルを開き、抽出された追跡エントリを参照できます。

lgprint は、問題が発生した場合に、サポート担当者が内部追跡エントリについての詳細情報を収集するために使用するコマンドです。したがって、正常稼働時にはこのコマンドを使用しないでください。

形式

```
lgprint [-d mmddhhmmyyyy | /mmddhhmmyyyy] [-p pid]
[-r root filename|-f filename] [-s]
```

説明

-d *mmddhhmmyyyy* | */mmddhhmmyyyy*

指定した日付と時刻の範囲で生成された追跡エントリだけが $PUBLIC/msg/elog_print ファイルに書き込まれます。最初の日時には、書き込まれる日付範囲の開始日時を入力し、2 番目の日時には範囲の終了日時を指定します。形式は次のようになります。

mm	月	01 〜 12
dd	日付	01 〜 31
hh	時刻 (24 時間表示)	00 〜 23
mm	分	00 〜 59
yyyy	年	1970 〜 2030

-p *pid*

Sun MBM デーモンの *pid* の追跡エントリが、$PUBLIC/msg/elog_print ファイルに書き込まれます。elog_print ファイルには、指定した PID のデーモンによって生成された追跡エントリだけが書き込まれます。

-r *root filename*|-f *filename*

-r *root-filename*

$PUBLIC/msg/elog_print ファイルに書き込むシリアルファイルの名前。ファイル名の末尾には、_00、_01、という順序番号が接尾辞として追加されます。複数のシリアルファイルを使用できます。

-f *filename*

$PUBLIC/msg/elog_print ファイルに書き込む循環ファイルの名前。

-s

保存されている elgsave ファイルを読み取ります。

例

次の例では、追跡エントリが $PUBLIC/msg/elog_print ファイルに書き込まれます。結果を参照するにはテキストエディタを使用します。

```
[node1] # lgprint -d 121007471996/121008002001
Mon May 02 07:47:44 2005 PID=10290     CPU=0
ID VCFL Pid 10290 (    VCFLOG)
ebm_history_file_msg set to job_only
```

次のコマンドでは、admlog コマンドによって定義された循環ファイル $PUBLIC/msg/elgfile が、$PUBLIC/msg/elog_print ファイルに書き込まれます。ファイルの内容を参照するにはテキストエディタを使用します。

```
[node1] # lgprint -f $PUBLIC/msg/elgfile
```

lgprint コマンド出力の日付には、EBM_DATE_FORMAT 環境変数の形式が適用されます。次の例では、この環境変数に次の形式が設定されている場合の出力例を示します。

```
export  EBM_DATE_FORMAT="** %A %B %d %I:%M:%S %p %Y **"
```

形式設定指示のリストについては、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

lgprint コマンドと出力は次のとおりです。

```
[node1] # lgprint -d 011222351998; pg $PUBLIC/msg/elog_print

*************************************************************
*   L O G     F I L E :elgfile                    *
*************************************************************

** records logged from:** Monday May 02 10:35:00 PM 2005 **

** Friday April 29 11:59:07 PM 2005 **      PID=10290     CPU=0
ID VCFL Pid 10290 (    VCFLOG)
ebm_history_file_msg set to job_only
```

lstjcl - ジョブクラス情報の一覧表示

lstjcl コマンドは、1 つまたはすべてのジョブクラスについての情報を一覧表示するために使用します。

形式

lstjcl [*class*] [-N *nodename*]

説明

class

特定のクラス (a ～ z) の情報を一覧表示します。この引数を指定しなかった場合は、すべてのジョブクラスの情報が表示されます。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上の、指定のジョブクラスに関する情報を表示します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

注 – ユーザーが取得できるのは、自分のユーザー ID でサブミットしたジョブに関する情報だけです。Sun MBM 管理者は、すべてのクラスに割り当てられているすべてのジョブの情報を一覧表示できます。

例

```
[node1/fs1] # lstjcl
BS1081(I) command lstjcl
Jon   Jobname   P C D  Strtime    Endtime   Start from  Status     Type   M   V
002   prova5      c d  10:00tm    2:00tm    BeginJob    Submitted  Std    y   n
005   provb5      b d  -          -         BeginJob    Running    Std    n   n
006   provb5      b d  -          -         BeginJob    Submitted  Std    y   n
BS1038(I) lstjcl command executed
```

説明

Jon

ジョブのシーケンス番号。

Jobname

サブミットしたジョブの名前。

P

ジョブの優先順位。

C

ジョブが割り当てられているクラス (a ～ z)。

D

ディスポジションコード。次のとおりです。

d: 指定したクラスのすべてのバッチジョブについて、実行のスケジューリングを行います。この値は通常、ジョブクラスに保持ディスポジションコードが定義されている場合に使用します。デフォルト値です。

h: 指定したクラスに割り当てられているジョブはスケジューリングも実行もされず、あとで処理されるまで SYSIN ファイル内に保持されます。

k: ジョブの実行がスケジューリングされ、実行後も SYSIN ファイル内のジョブまたはシェルスクリプトが保存されます。実行後のジョブのディスポジションは、保持 (h) に変更されます。

Strtime

ジョブの実行開始時刻 (td は当日、tm は翌日の時刻を示します)。td、tm のどちらも表示されていない場合、その時刻は経過し無効になったことを示します。

Endtime

ジョブの実行終了時刻。

Start from

ジョブの先頭 (BeginJob)、ジョブのステップ名、手続きの先頭、および手続きのステップ名のうち、ジョブの開始ポイントを示します。

Status

ジョブの状態が、Submitted、Running、または Suspended で示されます。

Type

ジョブのタイプ。Sun MTP (Sun MTP システムにサブミットされたバッチジョブ) または Std。

M

ジョブの状態がユーザーにメールで通知されるかどうかが、y または n で示されます。

V

ジョブが妥当性検査モードで実行されるかどうかが、y または n で示されます。

lstjfl - ジョブファイルの一覧表示

lstjfl コマンドは、バッチ JCL ファイルまたはバッチジョブファイルの内容を一覧表示するために使用します。

形式

lstjfl *name* [-e *eln*] [-N *nodename*] [-s *sln*] [-t *type*] [-u *user*]

説明

name

一覧表示するファイルの名前。

-e *eln*

一覧表示するファイルの最終行の番号 (最大 5 桁)。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上のバッチ JCL ファイルまたはバッチジョブファイルの内容を一覧表示します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

-s *sln*

一覧表示するファイルの最初の行の番号 (最大 5 桁)。

-t *type*

一覧表示するファイルのタイプは次のとおりです。

j: jmvs ディレクトリに置かれている JCL ファイル (デフォルト)。

n: ish ディレクトリに置かれている新規のジョブファイルまたはバッチジョブ。

-u *user*

ユーザー ID またはファイル所有者。

注 – ユーザーが一覧表示できるのは、自分のユーザー ID に関連付けられている JCL ファイルまたはジョブファイルだけです。また、それらのファイルが jmvs および ish ディレクトリに置かれている必要があります。Sun MBM 管理者は任意のファイルを一覧表示できます。

例

```
[node1/fs1] # lstjfl plan
BS1080(I) command lstjfl
//PLAN JOB
//JOBLIB DD DSN=TEST.COB.GNT,DISP=SHR
//TPCO EXEC PGM=IDCAMS
//SYSIN DD DSN=TEST.LIBG(SYS001),DISP=SHR
      .
      .
BS1038(I) lstjfl command executed
```

lstjob - ジョブの属性の一覧表示

lstjob コマンドは、実行待機中のジョブに関する情報を取得します。このコマンドに指定したジョブが実行中の場合は、そのことを示すメッセージが出力されます。

形式

lstjob *jon* [-n *name*] [-N *nodename*]

説明

jon

ジョブのオカレンス番号。

-n *name*

jon を確認するためのプログラム名。この名前と *jon* が関連付けられていない場合は、エラーメッセージが表示されます。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上で実行待機中の特定のジョブに関する情報を表示します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

注 – ユーザーが取得できるのは、自分のユーザー ID でサブミットしたジョブに関する情報だけです。Sun MBM 管理者は任意のジョブの情報を取得できます。

例

```
[node1/fs1] # subjob test
BS1100(I) Job test submitted :jon = 107 class = a
[node1/fs1] # lstjob 107
BS1083(I) Command lstjob
Jon    Jobname  P C D   Strtime  Endtime  Start from  Status     Type  M   V
User   Cycle  HistoryDir     Suspend   SaveHistoryDir       X  A
JobIdentifier                   Ebmsys     JobParameter        RescheduledAt
-------------------------------------------------------------------------------
107    JOB001   5 a d  -        -         BeginJob    Submitted  Std   n  n
mbmuser         history1                SAVE_DIR           n  n
JOB001@02May2005:10:41:00       sys1       file1
BS1038(I) lstjob command executed
```

説明

Jon

ジョブのシーケンス番号。

Jobname

サブミットしたジョブの名前。

P

ジョブの優先順位。

C

ジョブが割り当てられているクラス (a 〜 z)。

D

ディスポジションコード。

d: 指定したクラスのすべてのバッチジョブについて、実行のスケジューリングを行います。この値は通常、ジョブクラスに保持ディスポジションコードが定義されている場合に使用します。デフォルト値です。

h: 指定したクラスに割り当てられているジョブはスケジューリングも実行もされず、あとで処理されるまで SYSIN ファイル内に保持されます。

k: ジョブの実行がスケジューリングされ、実行後も SYSIN ファイル内のジョブまたはシェルスクリプトが保存されます。実行後のジョブのディスポジションは、保持 (h) に変更されます。

Strtime

ジョブの実行開始時刻 (td は当日、tm は翌日の時刻を示します)。

Endtime

ジョブの実行終了時刻。

Start from

ジョブの開始ポイントが、ジョブの先頭 (BeginJob)、ステップ名、または手続き名で示されます。

Status

ジョブの状態が、Submitted、Running、Suspended、Aborted、または Finished で示されます。

Type

ジョブのタイプ。Sun MTP (Sun MTP 領域にサブミットされたバッチジョブ) または Std で示されます。

M

ジョブの状態がユーザーにメールで通知されます。

V

ジョブが妥当性検査モードで実行されるかどうかが、y または n で示されます。

User

ジョブをサブミットしたユーザーの名前

Cycle

周期的に実行されるジョブの場合、完了後に再開される間隔が秒数 *nn* で示されます。

HistoryDir

-h オプションを指定した subjob または unikixjob コマンドでサブミットされたジョブの場合の、ジョブ出力の一覧が書き込まれる出力先ディレクトリ。

Suspend at

ジョブが中断するステップ。

SaveHistoryDir

subjob または unikixjob コマンドの -o オプションによってジョブに関連付けられた環境変数の名前。

X

haltrans -x コマンドで解釈されたジョブの下位互換性のためにのみ使用します。値は、y または n です。

A

ジョブがデバッグを実行するためにサブミットされたかどうかを示します。値は、y または n です。

JobIdentifier

subjob または unikixjob コマンドの -J オプションによってジョブに割り当てられたジョブ識別子。

Ebmsys

subjob または unikixjob コマンドの -k オプションによってジョブに関連付けられたサブシステムの名前。

JobParameter

subjob または unikixjob コマンドの -P オプションによってジョブに渡される文字列。

ReschedulledAt

周期的に実行されるジョブの次回実行の開始日時。

lststs - 履歴ファイル内のジョブ状態の一覧表示

lststs コマンドは、Sun MBM 履歴ファイルからそのユーザーに関連するメッセージを表示するために使用します。オプションを指定した場合は、ジョブ履歴ファイルの内容も表示されます。lststs コマンドでは、現在の Sun MBM セッションに関するメッセージだけが表示されます。つまり、lststs コマンドで一覧表示できるのは、そのノードが最後に起動した時点以降に生成されたメッセージだけです。

ジョブ履歴ファイルの名前の形式は次のとおりです。

destination-directory/*jobname-jobnumber*.hst.*ddmonthyyyy*.*hh*:*mm*

注 – 月名については、LC_TIME 環境変数の値に基づいて出力されます。

たとえば、ジョブ TESTVSAM の出力一覧ファイルは次の名前で作成されます。

destination-directory/TESTVSAM.hst.04May2002.13:05

ジョブ履歴ファイルのデフォルトの出力先ディレクトリは /tmp で、ユーザーが BAM ユーティリティーを使用して変更できます。また、BAM では、特定のサブシステムに関するジョブ履歴ファイル用に、別の出力先ディレクトリの定義もできます。その方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。ジョブ履歴ファイルの出力先ディレクトリを変更するには、subjob および unikixjob コマンドの発行時に -h オプションを使用する必要があります。

注 – ファイルシステムのオーバーフローを防ぐため、これらの履歴ファイルディレクトリのすべてを監視し、定期的に削除、圧縮する必要があります。

形式

lststs [-c] [-t] [-j *jon*] [-N *nodename*] [-u *user*]

説明

-c

jon で指定したジョブに関するジョブ履歴ファイルの内容を表示します。

-j *jon*

ジョブオカレンス番号 *jon* に指定したジョブに関するメッセージを一覧表示します。

`-N` *nodename*

nodename で指定した遠隔 Sun MBM ノード上の履歴ファイルから、このコマンドを発行したユーザーに関連するメッセージを一覧表示します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

`-t`

jon で指定したジョブに関するジョブ履歴ファイルの内容を表示します。この結果はシステムコマンド `tail -f` を実行した場合と同じです。画面を切り替えるには、端末割り込みキーを使用します。

`-u` *user*

指定したユーザー ID のメッセージを一覧表示します。指定可能なユーザー ID は 14 文字以内です。このオプションは Sun MBM 管理者だけが使用できます。

注 – ユーザーが参照できるのは、自分のユーザー ID に関連付けられているメッセージだけです。Sun MBM 管理者は任意のユーザー ID を指定してメッセージを参照できます。

例

```
[node1/fs1] # lststs
Jon    Uid    Message
 21    300    IP4305(I) Start of job proacc at May 2 14:43:58 2005
 21    300    IP0991(W) No default destination found
 22    300    IP4305(I) Start of job proacb at May 2 15:10:03 2005
 21    300    proacc started now
     .
     .
```

説明

`Jon`

ジョブのオカレンス番号。

`Uid`

ユーザー ID。

Message

そのジョブに関連付けられているメッセージ。

次の例では、ジョブの実行レポートは、subjob または unikixjob コマンドで -h オプションによって指定された履歴ファイルに書き込まれます。実行中のジョブを監視するには、-t オプションを指定して実行レポートを表示します。

```
[node1/fs1] # lststs -j721 -t
Jon    Uid    Message
721    300    IP4305(I) Start of job test at May 215:42:46 2001
721    300    IP4334(I) History filename:/tmp/test_721.hst.02May2005.15:42
722    300    IP4304(I) Job test terminating at May 2 14:42:55 2005
IP4305(I) Start of job test at May 2 15:42:46 2005
IP4334(I) History filename:/tmp/test_721.hst.02May2005.15.42
IP0991(W) No default destination found
starting JOB -- JOB01
IP4350(S) FILEMAP environment variable is read only
starting STEP -- STEP01
IP4104(I) Start COBOL1 at May 2 15:42:50 2005
PGM=COBOL1 in the validation mode is not executed:
searching in /tmp/sys1/linklib
PGM=COBOL1 not found
IP4105(I) End of COBOL1 at May 2 15:42:50 2005
STEP01 step was executed cond code 0
step STEP01 start    at TIME=03:42:49 PM
step STEP01 stop     at TIME=03:42:60 PM
starting STEP - - STEP02
IP4104(I) Start COBOL2 at May 2 15:42:52 2005
PGM=COBOL2 in the validation mode is not executed:
searching in /tmp/sys1/linklib
PGM=COBOL2 not found
IP4105(I) End of COBOL2 at May 2 15:42:52 2005
STEP02 step was executed cond code 0
step STEP02 start    at TIME=03:42:52 PM
step STEP02 stop     at TIME=03:42:52 PM
No files to be printed
     .
     .
```

mvstrans - MVS JCL ジョブスクリプトへの変換

MVS JCL トランスレータ mvstrans には、MVS JCL ジョブまたは手続きを入力します。トランスレータからは、実行可能なジョブスクリプトが出力されます。このスクリプトには C シェル文、システムコマンド、および Sun MBM マクロ呼び出しを組み込むことができます。ジョブはサブシステムにサブミットされます。

mvstrans では、環境変数 FILEMAP、FMROOT、PROCLIB、および TRANSOPTS が使用されます。これらの環境変数がサブシステムの設定ファイルに設定されていれば、サブシステムに対する batchenv シェルスクリプトの実行時にそのサブシステム名を引数として指定すると、環境変数が継承されます。

mvstrans は、次のサブディレクトリがあるディレクトリから実行する必要があります。

jmvs

入力ディレクトリ。変換する IBM MVS JCL ジョブが含まれます。

mvsp

入力ディレクトリ。変換する IBM MVS JCL 手続きが含まれます。

ish

出力ディレクトリ。jmvs ディレクトリ内の入力ジョブファイルから変換されたマクロジョブスクリプトが含まれます。

注 – 変換された手続きの出力用ディレクトリは、サブシステム作成時に設定される PROCLIB 環境変数の値によって決まります。

MVS JCL の変換プロセスは、次の 2 つの手順で構成されます。

1. まず、JCL の内容の妥当性を検査し File_Map を構築するために、mvstrans を妥当性検査モードで実行します。必要に応じて、File_Map 内の VSAM ファイルまたは世代別データグループ (GDG) を更新します。

2. 通常モードで mvstrans を実行し、ジョブスクリプトを生成します。

TRANSOPTS 環境変数が TRANSOPTS="-c -v" と設定されている場合、コマンド行に引数を指定しないで mvstrans を実行すると、変換処理は妥当性検査モードで行われ、ファイル名は小文字に変換されません。

変換中、出力不能な文字が parm 文字列に含まれている場合、それらの文字はすべて /x*HH* という形式に変換されます。*HH* はその文字の 16 進数の値です。

形式

```
mvstrans filename [-b nnn] [-c] [-C] [-d] [-e] [-E] [-f] [-F] [-g]
[-J type] [-k nn] [-l] [-L n] [-m] [-M|-P|-I] [-n] [-N]
[-o output-filename] [-p] [-r [n|p|w]] [-R]
[-t [all|entry|exit|flow|jcl|lex|nnn[-mmm]]]
[-T [a|d|i|p|s]] [-u] [-U] [-v] [-V]
```

説明

filename

1 つ以上の入力 MVS SP JCL ストリームファイルの名前。それらのジョブファイルは、mvstrans を実行するディレクトリの jmvs サブディレクトリに置かれている必要があります。手続きファイルの場所は $PROCLIB で指定します。

'*' は、ジョブまたは手続きディレクトリ内のすべてのファイルを示します。この引用符は必ず入力してください。* は、1 つ以上のファイル名を指定する場合に、ほかの文字と組み合わせることもできます。

-b *nnn*

mvstrans 出力に生成されるマクロ呼び出し間に、区切り文字として挿入される空行の数 *nnn*。デフォルトは 1 です。

-c

ファイル名の大文字と小文字を File_Map 内に保持します。次に例を示します。

```
//DD1 DD DSN=AA.BB.cc.DD
```

-c を指定した場合の File_Map エントリ

```
AA.BB.cc.DD;MASTERCAT;FS;/tmp/files/AA/BB/cc/DD;;0;
```

-c を指定しない場合の File_Map エントリ

```
AA.BB.cc.DD;MASTERCAT;FS;/tmp/files/aa/bb/cc/dd;;0;
```

-C

JCL COMMENT 文を出力ファイル内に生成しません。

-d

有効なパーサーに応じて、*filename* に対するデバッグ解析プロセスを実行します。mvstrans には 3 つの独立したフェーズがあり、それぞれ固有のパーサーを備えています。この -d オプションは、-I、-M、または -P オプションと併用できます。これらのオプションの詳しい使用法は、ご購入先に問い合わせてください。

-e

データセットを定義するために mvstrans 出力で生成されたすべての ASSGNDD マクロ呼び出しの filename パラメータ、およびライブラリを定義する各 LIBDEF マクロ呼び出しの lib パラメータを省略します。

-E

変換時に、インストリームデータに含まれる記号パラメータを展開します。たとえば、次の JCL を展開します。

```
//PARMIN DD *
&JOBPARM
/*
```

-E オプションを指定しなかった場合は、次のように変換されます。

```
ASSGNDD ddname='PARMIN' type='INSTREAM' <<!
\&JOBPARM
!
```

-E オプションを指定した場合は、次のように変換されます。

```
ASSGNDD ddname='PARMIN' type='INSTREAM' <<!
$JOBPARM
!
```

次の JCL を考えます。

```
//PARMIN DD *
&JOBPARM..FILE1
/*
```

-E オプションを指定しなかった場合は、次のように変換されます。

```
ASSGNDD ddname='PARMIN' type='INSTREAM' <<!
\&JOBPARM..FILE1
!
```

-E オプションを指定した場合は、次のように変換されます。

```
ASSGNDD ddname='PARMIN' type='INSTREAM' <<!
${JOBPARM}.FILE1
!
```

マクロが実行時に展開される場合に $JOBPARM を展開します。そうでない場合は &JOBPARM と同等です。

-f

すでに出力ファイルがある場合でも、出力を再生成するように指示します。

-F

生成されるマクロ呼び出しのデフォルトパラメータ値を書き込みます。通常、デフォルト値と一致するパラメータ値は mvstrans 出力に書き込まれません。

-g

-v オプションと併用することにより、DD 文の DSN パラメータに以下の形式で定義されている一時データセットに関する File_Map エントリを生成させることができます。

DSN=&&..

妥当性検査モードで -g を指定しなかった場合のデフォルト動作では、一時データセットに関する File_Map エントリは生成されません。

-J *type*

指定した *type* の変換規則をすべてのジョブと手続きに適用し、現在 SYSTEM_OUTPUT 環境変数によって定義されているデフォルトの変換規則タイプを無効にします。

JCL: JCL 変換規則を適用します。

JES2: JES2 変換規則を適用します。

-k *nn*

インストリームデータセットのレコードの *nn* 列のあとの文字を切り捨てます。デフォルトは 80 です。

-l

完全な入力 JCL リストが含まれるファイルを保持します。このファイルは、JCL メンバーファイルが含まれる出力ファイルです。プリプロセスフェーズおよび変換フェーズで表示される警告および情報メッセージは、このファイルの内容に関するものです。

-L *n*

File_Map 内に生成されるファイル名が適用されるディレクトリの数を制限します。たとえば、次の JCL を使用します。

```
//TEST DD DSN=AA.BB.CC.DD.EE
&JOBPARM
/*
```

-L 0 オプションを指定した場合は、次のように変換されます。

```
filename=${FMROOT}/aa.bb.cc.dd.ee
```

-L 1 オプションを指定した場合は、次のように変換されます。

```
filename=${FMROOT}/aa/bb.cc.dd.ee
```

-L オプションを指定しない場合は、次のように変換されます。

```
filename=${FMROOT}/aa/bb/cc/dd/ee
```

-m

ASSGN DD 文に recfmt の値として定義されている 'V' (可変長ファイル) を、'mfrcdv' (Micro Focus COBOL 可変長ファイルタイプ) に変更するように指定します。たとえば、次の JCL を使用します。

```
//DD1 DD DSN=YYY,DISP=SHR,RECFM=V
```

-m オプションを指定した場合は、次のように変換されます。

```
ASSGNDD ddname='DD1' dataset='YYY' \\\
       filename='/tmp/files/yyy' disp='i-o' \\\
       recfmt='mfrcdv'
```

-m オプションを指定しない場合は、次のように変換されます。

```
ASSGNDD ddname='DD1' dataset='YYY' \\\
       filename='/tmp/files/yyy' disp='i-o' recfmt='V'
```

[-M|-P|-I]

-M: INCLUDE 文によって指定された JCL メンバーだけを組み込みます。ジョブに関する出力は ./ish ディレクトリ ($JCLLIB) に置かれ、手続きに関する出力は ./ishp ディレクトリ ($PROCLIB) に置かれます。これらの出力ファイルは、入力ファイル名に接尾辞 .s を付けた名前で作成されます。

-P: JCL プリプロセスフェーズだけを実行します。このフェーズでは、入力 JCL を検証し、Sun MBM 環境でサポートされる文とコマンドだけを抽出します。入力されるファイルは、JCL メンバー組み込みフェーズで生成された出力ファイル (接尾辞.s) です。

ジョブに関する出力は ./ish ディレクトリ ($JCLLIB) に置かれ、手続きに関する出力は ./ishp ディレクトリ ($PROCLIB) に置かれます。これらの出力ファイルは、入力ファイル名に接尾辞 .p を付けた名前で作成されます。

-I: JCL 変換フェーズだけを実行します。入力されるファイルは、プリプロセスフェーズで生成された出力ファイル (接尾辞.p) です。この変換フェーズで生成されるファイルは、実行可能なジョブスクリプトファイルです。

このオプションとともに -v オプションを指定した場合、出力は生成されません。ただし、File_Map では必要に応じてデフォルトのデータセットエントリが更新されます。

-n

変換の実行日付を示すタイムスタンプ値を、生成された mvstrans 出力ファイルから除外します。

-N

入力 JCL ファイルのステップごとにステップ名を生成するように指示します。生成されたステップ名は、変換出力内の各ステップの識別子よりも優先され、出力に含まれる各ステップの開始時の LABEL マクロ呼び出し内に組み込まれます。

注 – ステップ名で指定された手続きステップの優先指定または後方参照が JCL 内に含まれている場合は、このオプションを使用しないでください。

-o *output-filename*

1 つの入力ファイルの変換時に使用される出力先ファイル。

-p

入力ファイルが、mvstrans を実行したカレントディレクトリ下の mvsp ディレクトリに含まれている MVS JCL 手続きファイルであることを示します。

-r

警告メッセージの抑止レベル。

n: メッセージは抑止されません。すべての警告メッセージが表示されます。デフォルト値です。

p: JCL 文のパラメータフィールドの処理中に、サポートされないオペランドが検出された場合の警告メッセージを抑止します。

w: mvstrans によって発行されるすべての警告メッセージを抑止します。

-R

この指定により、PROGRAMMER_NAME 環境変数に設定された値が、ジョブをサブミットしたユーザーではなく、ジョブの作成者として使用されます。

-t [all|entry|exit|flow|jcl|lex|*nnn* [-*mmm*]]

追跡機能を有効化します。生成される追跡メッセージのレベルは、次の引数によって制御します。

all: すべての追跡メッセージを表示します。

entry: mvstrans の C 関数の入口に関するメッセージだけを表示します。

exit: mvstrans の C 関数の出口に関するメッセージだけを表示します。

flow: mvstrans の C 関数の実行開始時と終了時のメッセージだけを表示します。

jcl: サポートされる JCL 文の解析および変換時のメッセージを表示します。

lex: 入力 JCL ジョブストリームの字句解析時のメッセージを表示します。

nnn [-*mmm*]: サポートされる JCL 文について、*nnn -mmm* で指定した行範囲で追跡メッセージを生成します。

-T [a|d|i|p|s]

追跡機能を有効化する範囲。

a: JCL 変換プロセス全体を追跡します。デフォルト値です。

d: mvstrans の制御機構を追跡します。

i: JCL 変換フェーズを追跡します。

p: JCL プリプロセスフェーズを追跡します。

s: JCL メンバー組み込みフェーズを追跡します。

-u

mvstrans オプションの構文を表示します。

-U

デバッグおよび追跡オプションを含む mvstrans オプションの構文を表示します。

-v

入力 JCL ストリームファイルの妥当性を検査し、その入力ファイルに定義されているデータセットとライブラリについてデフォルトの File_Map エントリを生成します。

妥当性検査モードでは、mvstrans は FILEMAP 環境変数で定義された File_Map ファイルにエントリを自動生成します。ただし、mvstrans コマンドを実行する場合は、-v を指定するかどうかにかかわらず、$FILEMAP が既存のファイルを指示するよう事前に設定しておく必要があります。FILEMAP 変数については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

FMROOT 環境変数は、妥当性検査モードでの変換時にすべてのファイル名の先頭に付けられるパスを定義します。これにより、該当のファイル名に対応する MVS データセット名が生成されます。$FMROOT が設定されていない場合は、デフォルトのパス /tmp が使用されます。$FMROOT の詳細は、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

-V

mvstrans の出力ファイル内に生成されているすべてのマクロ呼び出しに対し、冗長モードを有効にします。

例

次に、DISP パラメータを指定する DD 文の例を示します。

```
...
//INFILE  DD DSN=TEST.FILE1,DISP=(OLD,KEEP,KEEP)
//OUTFILE DD DSN=TEST.FILE2,DISP=(NEW,CATLG,DELETE)
//MST0001 DD DSN=TEST.MST0001,DISP=SHR
```

mvstrans -e を実行し、ASSGNDD *filename* パラメータの生成を省略するように指示した場合、これらの DD 文は次のように変換されます。

```
...
ASSGNDD ddname='INFILE' dataset='TEST.FILE1' \\\
       disp='i-o' normal='k' abend='k'
ASSGNDD ddname='OUTFILE' dataset='TEST.FILE2' \\\
       disp='o' normal='k' abend='d'
ASSGNDD ddname='MST0001' dataset='TEST.MST0001' \\\
       disp='i-o'
```

JCLLIB 文によって 2 つの検索ライブラリを指定した JOB

```
//NOR9999  JOB JOB1
//LIBSRCH  JCLLIB  ORDER=(USER.PROCLIB1,USER.PROCLIB2)
...
```

$FMROOT が /sys1 に設定されている場合は、mvstrans -v を実行すると、次のようなエントリが File_Map 内に生成されます。

```
USER.PROCLIB1;__LIB;FS;/sys1/user/proclib1;;0;
USER.PROCLIB2;__LIB;FS;/sys1/user/proclib2;;0;
```

rpljob - ジョブへのデータの引き渡し

rpljob コマンドは、ジョブスクリプト内のバッチシェル accept 文で一時停止しているジョブに、ユーザーが指定した文字列のデータを渡すために使用します。ジョブにデータを渡すには、rpljob コマンドを発行することおよび対象ジョブ内に次の形式の accept 文が含まれていることが不可欠な条件です。

形式

accept *xxx*

説明

xxx

rpljob によってジョブに渡される文字列の値を割り当てるバッチシェル変数。

バッチシェルプロセスは、ジョブ内で accept 文を検出すると、そのジョブの *jon* が指定された rpljob コマンドが発行されるまで、ジョブを中断します。コマンドからデータを受け取ったバッチシェルは、変数 *xxx* にそのデータを割り当てます。このタイプの変数は、$*xxx* によって参照されます。

形式

rpljob *jon string* [-N *nodename*]

説明

jon

accept 文で中断しているジョブのジョブ番号。

string

accept 文に渡すデータ。この文字列は二重引用符で囲むこともできます。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上で応答を待機しているジョブに対してこのコマンドを実行します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

rsmjob - ジョブの実行の再開

rsmjob コマンドは、susjob コマンドによって中断されたバッチジョブの実行を再開するために使用します。

形式

rsmjob *jon* [-N *nodename*]

説明

jon

ジョブのオカレンス番号。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上で中断されたバッチジョブの実行を再開します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

ユーザーは、自分のユーザー ID でサブミットしたジョブだけを再開できます。
Sun MBM 管理者は、どのユーザーがサブミットしたジョブでも再開可能です。

例

ジョブ prog1 がサブミットされ、susjob によって中断されたあと、rsmjob コマンドによって再開されるまでを示します。

```
[node1/fs1] # subjob prog1
BS100(I) Job prog1 submitted :jon = 38 class = a
[node1/fs1] # infact
BS1065(I) Command infact
BS1070(I) - Inform on activity names :
act1 (J038)
BS1038(I) infact command executed
 .
 .
[node1/fs1] # susjob 38
BS1048(I) Jon 38 suspended
 .
 .
[node1/fs1] # rsmjob 38
BS1049(I) Jon 38 resumed
```

runjcl - ジョブクラスの実行

runjcl コマンドは、保持 (h) ディスポジションコードを指定してサブミットしてあるジョブを、適切なクラスを指定して実行するために使用します。また、runjcl コマンドでは、1 つのクラス全体に適用するディスポジションコードを k に設定し、そのクラスに割り当てられたジョブを実行して SYSIN ファイル内に保持もできます。

注 – runjcl は、SYSIN ファイル内に置かれているジョブだけに作用するコマンドで、現在実行中のジョブには適用されません。

形式

runjcl *class* [-N *nodename*] [-T]

説明

class

実行対象とする a ～ z のジョブクラス。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上で、保持 (h) ディスポジションコードを指定してサブミット済みのジョブを実行します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

-T

ジョブに関連付けられている時刻を、当日の時刻に適用します。たとえば、現在の時刻が 14:00 の場合、-T オプションを指定しないで開始時刻を 12:00 に指定したジョブをサブミットすると、そのジョブは自動的に翌日の日付でスケジューリングされます。

-T オプションを指定した場合は、Sun MBM アクティビティーと Sun MTP トランザクションサーバーの空き状況に応じ、このジョブの実行が当日のできるだけ早い時刻でスケジューリングされます。

注 – -T オプションは、unikixjob コマンドでサブミットしたジョブだけに有効です。

---

runjcl コマンドは、ユーザー自身のユーザー ID でだけ発行できます。Sun MBM 管理者は、指定したクラスに割り当てられているすべてのジョブのディスポジションを変更できます。

例

例では、次のことを示します。

- クラス b および c に割り当てられ、保持ディスポジションコードとともにサブミットされたジョブの例を示します。
- サブミットされたジョブの状態を、lstjcl コマンドを使用して表示します。
- 定義されているアクティビティー内で実行中のジョブがないことが、infact コマンドによって示されます。

- クラス b に対して runjcl コマンドが実行されたあとに lstjcl コマンドが実行され、クラス b に割り当てられているジョブが実行中になっていることが示されます。

```
[node1/fs1] # subjob prova -c b -d h
BS1100(I) Job prova submitted :jon = 7 class = b
[node1/fs1] # subjob prova -c c -d h
BS1100(I) Job prova submitted :jon = 8 class = c
[node1/fs1] # lstjcl
BS1081(I) command lstjcl
Jon    Jobname PCD   Strtime Endtime  Start from  Status     Type   M  V
002    prova   5cd   -       -        BeginJob    Submitted  Std    n  n
007    prova   5bh   -       -        BeginJob    Submitted  Std    n  n
008    prova   5ch   -       -        BeginJob    Submitted  Std    n  n
BS1038(I) lstjcl command executed
[node1/fs1] # infact
BS1065(I) Command infact
BS1070(I) - Inform on activity names :
acta actb
BS1038(I) infact command executed
[node1/fs1] # runjcl b
BS1038(I) runjcl command executed
[node1/fs1] # infact
BS10v38(I) infact command executed
[node1/fs1] # lstjcl
BS1081(I) command lstjcl
Jon    Jobname PCD   Strtime Endtime  Start from  Status     Type   M  V
002    prova   5cd   -       -        BeginJob    Submitted  Std    n  n
007    prova   5bk   -       -        BeginJob    Running    Std    n  n
008    prova   5ch   -       -        BeginJob    Submitted  Std    n  n
BS1038(I) lstjcl command executed
```

runjob - ジョブの実行

runjob コマンドは、バッチジョブを、ディスポジションコード k に設定して実行し、実行後も SYSIN ファイルに保持するために使用します。このコマンドは、保持 (h) ディスポジションコードが指定されて SYSIN ファイル内に置かれているジョブに対して有効です。

注 – runjob は、SYSIN ファイル内のジョブだけに作用するコマンドで、現在実行中のジョブには適用されません。

形式

`runjob` *jon* `[-N` *nodename*`] [-T]`

説明

jon

ジョブのオカレンス番号。

`-N` *nodename*

nodename で指定した遠隔 Sun MBM ノードの `SYSIN` ファイル内の、指定したバッチジョブにディスポジションコード `k` を設定して実行し、実行後も SYSIN ファイルに保持します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

`-T`

ジョブに関連付けられている時刻を、当日の時刻に適用します。

たとえば、現在の時刻が 14:00 の場合、`-T` オプションを指定しないで開始時刻を 12:00 に指定したジョブをサブミットすると、そのジョブは自動的に翌日の日付でスケジューリングされます。

`-T` オプションを指定した場合は、Sun MBM アクティビティーと Sun MTP トランザクションサーバーの空き状況に応じ、このジョブの実行が当日のできるだけ早い時刻でスケジューリングされます。

注 – `-T` オプションは、`unikixjob` コマンドでサブミットしたジョブだけに有効です。

ユーザーは、自分のユーザー ID でサブミットしたジョブに限り、ディスポジションコードをリセットできます。Sun MBM 管理者は、すべてのジョブのディスポジションコードを変更できます。

例

この例では、保持 (h) ディスポジションコードを指定してジョブをサブミットします。runjob コマンドの発行により、このジョブのディスポジションコードが k に変更され、ジョブのスケジューリングが可能となります。

```
[node1/fs1] # subjob prova -d h -c b
BS1100(I) Job prova submitted :jon = 10 class = b
[node1/fs1] # lstjcl b
BS1081(I) command lstjcl
Jon    Jobname P C D   Strtime Endtime  Start from  Status     Type   M   V
007    prova   5 b h   -       -        BeginJob    Finished   Std    n  n
010    prova   5 b h   -       -        BeginJob    Submitted  Std    n  n
BS1038(I) lstjcl command executed
[node1/fs1] # runjob 10
BS1038(I) runjob command executed
[node1/fs1] # lstjcl b
BS1081(I) command lstjcl
Jon    Jobname P C D   Strtime Endtime  Start from  Status     Type   M   V
007    prova   5 b h   -       -        BeginJob    Finished   Std    n  n
010    prova   5 b k   -       -        BeginJob    Running    Std    n  n
BS1038(I) lstjcl command executed
```

subjob - ジョブのサブミット

subjob コマンドは、Sun MBM に対してジョブをサブミットするために使用します。Sun MTP VSAM ファイルへのアクセスを必要とするジョブは、subjob コマンドを使用してサブミットできません。VSAM ファイルにアクセスするジョブは、unikixjob コマンドを使用してサブミットする必要があります。

形式

subjob *name* [-a] [-c *class*] [-d *disp*]
[-D [COBOL|DEBUG_MACRO|MACRO|VSAM]] [-h *history-dir*] [-j] [-J *jobid*]
[-m] [-N *nodename*] [-o *save-hist-env*] [-p *prior*] [-P *job-parameter*]
[-k *batchsystem-name*] [-R *stepname*[.*procstepname*]]
[-S *stepname*[.*procstepname*]] [-t *type*] [-U *user*] [-v] [-w] [-W *workdir*]
[-x]

説明

name

サブミットするバッチジョブの名前。指定するジョブは、ish または jmvs ディレクトリに置かれている必要があります。subjob コマンドに指定したジョブ名は、その他のコマンドで *jon* とジョブ名とが関連付けられているかどうかを確認するために使用します。

-a

COBOL デバッガを使用して、バッチジョブをデバッグできるように準備します。バッチジョブをデバッグできるのは、ユーザーが anmjob コマンドによってそのバッチプログラムにアクセスした場合だけです。

-c *class*

指定したクラス (a ～ z) に対してジョブをサブミットします。デフォルトは a です。

-d *disp*

ジョブにディスポジションコードを割り当てます。

d: 実行するためにジョブをディスパッチするか、作成します。デフォルト値です。

h: 指定したクラスに割り当てられているジョブはスケジューリングも実行もされず、あとで処理されるまで SYSIN ファイル内に保持されます。

k: ジョブの実行がスケジューリングされ、実行後も SYSIN ファイル内のジョブまたはシェルスクリプトが保存されます。実行後のジョブのディスポジションは、保持 (h) に変更されます。

-D

ジョブの実行中にデバッグを使用可能にします。

COBOL

指定したジョブで実行される COBOL ステップごとに、以下の形式の名前で Sun MBM COBOL 実行時デバッグファイルを生成します。

/tmp/*jobname.stepname.jobnumber*

この形式は、以下の環境変数の設定と同じです。

```
RTSDEBUGFILE=/tmp/$JOBNAME.$STEPNAME.$JON
```

Solaris プラットフォーム: COBOL アプリケーションが不正終了した場合は、pstack ダンプ情報もジョブの履歴ファイルに書き込まれます。

DEBUG_MACRO

デバッグマクロ出力を生成します。

MACRO

マクロを冗長モードで実行します。

VSAM

VSAM 入出力文ごとに、ファイル $KIXSYS/unikixmain.dbg にデバッグ情報を生成します。

複数のオプションを指定する場合は、次のようにコロンで区切る必要があります。

```
[node1/fs1] # subjob testa -D MACRO:COBOL:VSAM ...
```

-h *history-id*

指定した *history_id* に関連付けられている出力先ディレクトリに、ジョブ履歴ファイルを作成します。

history_id 識別子は、ユーザーが BAM ユーティリティーを使用してサブシステムを作成するときに定義します。

ジョブ履歴ファイルの名前の形式は次のとおりです。

destination-directory/*jobname_jobnumber*.*hst*.*date*.*time*

注 – *date.time* には、LC_TIME 環境変数の定義が適用されます。詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

-j

jon を標準出力に出力することを指示します。たとえば、JOB_NUM='subjob job1 -j' と定義した場合は、このジョブの *jon* が $JOB_NUM の値として戻ります。

-J *jobid*

ジョブ識別子 *jobid* をジョブに割り当てます。このオプションは、*jobid* を外部スケジューラから Sun MBM に渡すために使用します。デフォルトのジョブ識別子の形式は *jobname@date.time* です。

注 – *date.time* には、LC_TIME 環境変数の定義が適用されます。詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

-k *batchsystem*

ジョブをサブミットする対象サブシステム。-k オプションの指定がないジョブは、デフォルトのサブシステムにサブミットされます。サブシステムの作成方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

-m

ジョブが正常に終了したかどうかについての通知メールを、そのジョブをサブミットしたユーザーに送信するように Sun MBM に指示します。メッセージの形式は、次のいずれかです。

- BQM Job *nnn* (*jobname*) Ended, Status=000
- BQM Job *nnn* (*jobname*) Aborted, Status=3

説明

BQM: バッチキューマネージャー。

nnn: ジョブのオカレンス番号。

jobname: ジョブのサブミット時に指定したジョブ名。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノードに対してジョブをサブミットします。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

-o *save-hist-env*

ジョブの実行終了時にジョブ履歴ファイルが作成されるディレクトリを示す環境変数。サブシステム構成ファイル内で指定した環境変数を、適切なアクセス権が付与された既存のディレクトリに設定する必要があります。

ジョブ履歴ファイルの名前の形式は次のとおりです。

save-hist-env/jobname-jobnumber.lst.date.time

注 – *date.time* には、LC_TIME 環境変数の定義が適用されます。詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

-o オプションを指定すると、ジョブの開始時刻、実行履歴、終了時刻、状態、および履歴レコードが保存されたファイルについての情報が、lststs コマンド実行時に表示されます。

-p *prior*

指定したバッチジョブに対して、属しているクラス内での優先順位を割り当てます。有効な値は 0 ～ 9 で、9 が最高優先順位です。デフォルトは 5 です。

-P *job-parameter*

JOBPARM 環境変数によってバッチジョブに渡される、25 文字以内の文字列を指定します。

この -P オプションを指定しなかった場合、$JOBPARM には NULL 文字列が設定されます。このオプションによって $JOBPARM が設定された場合は、lstjob またはinfjbs コマンドを使用してその値を参照できます。

-R *stepname.procstepname*

stepname (8 文字以内) で指定したステップで、ジョブまたは手続きを再開します。

あるジョブを、特定の手続き内の特定のステップで再開するには、まずジョブのステップ名 (8 文字以内) を指定し、そのあとに手続き内のステップ名 (*procstepname*) を指定します。

-S *stepname.procstepname*

stepname (8 文字以内) で指定したステップで、ジョブまたは手続きを中断します。中断したジョブを再開するには、rsmjob コマンドを使用します。

-t *type*

ファイルのタイプ。JCL ファイルを指定した場合、subjob コマンドはそのファイルをマクロジョブに変換してからサブミットします。

m: mvstrans を使用して変換される MVS JCL ジョブ。

d: dostrans を使用して変換される VSE JCL ジョブ。

n: バッチシェルスクリプトファイル (デフォルト)。

JCL の機能の中にはサポートされないものもあるので、適切な Sun MBM トランスレータを使用して JCL ファイルを変換する必要があります。変換後のバッチジョブは、妥当性の確認後にサブミットできます。

-U *user*

ジョブを別のユーザー名でサブミットします。

ユーザー 1 が、このコマンド行からサブミットするジョブをユーザー 2 として実行する必要がある場合は、このコマンド行か、「Submit Job」ダイアログの「Job PARM」フィールドでこのオプションを指定する必要があります。

ユーザー 1 がユーザー 2 としてジョブを正常に実行するには、ユーザー 2 のホームディレクトリに定義されているバッチ許可ファイル .batch_auth で、適切な権限がユーザー 1 に付与されている必要があります。また、ユーザー 2 はこのバッチ許可ファイルの所有者である必要があります。したがって、このファイルには、グループおよびその他のユーザーに対して読み取り専用のアクセス権が設定されていることが必須条件です。

.batch_auth ファイルには、以下の形式で記述する必要があります。

job-full-pathname:*username1*, *username2*, *username3*

例

ユーザー Tom は、ディレクトリ /home/batch/jcl/ish 下に定義した JOB1 と JOB2 をユーザー Ken として実行する必要があり、ユーザー Tony は、ディレクトリ /home/batch/jcl/ish 下に定義した JOB2 と JOB3 を同じくユーザー Ken として実行する必要があるとします。

この場合、ユーザー Ken は、そのホームディレクトリに .batch_auth という名前のバッチ許可ファイルを作成する必要があります。このファイルには、グループとその他のユーザーの両方に対する、読み取り専用のアクセス権を設定する必要があります。バッチ許可ファイル内に次の行が必要です。

```
/home/batch/jcl/ish/JOB1:Tom
/home/batch/jcl/ish/JOB2:Tom, Tony
/home/batch/jcl/ish/JOB3:Tony
```

同じジョブのパス名を複数の行に重複して記述はできません。したがって、同じジョブパス名に複数のユーザーを定義する場合は、上の例のようにユーザー名をコンマまたは空白文字で区切る必要があります。

また、1 つの行を 2 行以上に延長する場合は、途中の行の末尾に継続行文字 \ (バックスラッシュ) を必ず入力してください。次に例を示します。

```
/home/batch/jobs/ish/JOB008:Tom, Tony, Frank, Jim, John, \
                            Kate
/home/batch/jobs/ish/JOB009:Tom, Tony, Frank, Jim, John, \
                            Kate, Al, Steve
/home/batch/jcl/ish/JOB1:    Tom
/home/batch/jcl/ish/JOB2:    Tom, Tony
/home/batch/jcl/ish/JOB3:    Tony
```

JOB2 のサブミット時に、Tom と Tony は次のコマンドを使用する必要があります。

```
[node1/fs1] # subjob JOB2 -W/home/batch/jcl/ish -U Ken
```

この /home/batch/jcl/ish/JOB2:エントリが、Ken のホームディレクトリ下の .batch_auth ファイル内に定義され、指定したジョブ JOB2 にユーザー Tom と Tony が定義されていれば、このジョブはユーザー Ken として実行します。

-W

このオプション付きでサブミットされたジョブが完了するまで、そのあとに記述されている Sun MBM コマンドを実行しないで待機するようにシステムに指示します。ジョブが正常に終了した場合は値ゼロを戻し、強制的に中止された場合はゼロ以外の値を戻します。このオプションによってジョブの同期をとる方法については、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』の例を参照してください。

-V

バッチジョブの論理の妥当性を検査するために、アプリケーションプログラムとユーティリティー関数の呼び出しを行わない方式でバッチジョブを実行します。バッチジョブが実行されると、アプリケーションプログラムとユーティリティー関数から実行の正常終了を示すリターンコードのシミュレーションが行われます。詳細は、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

-W *workdir*

サブミットするジョブの親ディレクトリ。たとえば、TESTECHO というジョブが /public/test/ish に置かれている場合、subjob を使用して /opt/batch1/test から次のようにジョブをサブミットできます。

```
[node1/fs1] # pwd
/opt/batch1/test
[node1/fs1] # subjob TESTECHO -W /public/test
```

指定するジョブは、ish ディレクトリに置く必要があります。ただし、ish ディレクトリは暗黙的に定義されているので、-W オプションのディレクトリパスに指定する必要はありません。

遠隔ノードにサブミットするジョブの場所を指示するには、-W オプションを -N オプションとともに指定します。subjob コマンドに-N オプションを指定し、-W は指定しなかった場合のデフォルト動作では、該当の遠隔ノード上の $PUBLIC/ish ディレクトリ内でジョブが検索されます。

-x

バッチジョブが実行される前にプリプロセスルーチンを呼び出すか、バッチジョブの実行後にポストプロセスルーチンを呼び出します。プリプロセスおよびポストプロセスルーチンは、バッチジョブによって呼び出されます。

-x オプションを指定すると、ユーザーのホームディレクトリ内で .pre_process.btsh ファイルが検索されます。ファイルが見つからない場合、$PUBLIC/bin ディレクトリ内で pre_process.btsh ファイルが検索されます。ファイルが見つかった場合、Sun MBM は、ジョブを実行する前にそのファイルを実行します。プリプロセスルーチンによって何らかの環境変数が設定された場合は、その値がバッチジョブに適用されます。

バッチジョブの実行終了時に、そのユーザーのホームディレクトリ内で post_process.btsh ファイルが検索されます。ファイルが見つからない場合、$PUBLIC/bin ディレクトリ内で post_process.btsh ファイルが検索されます。ファイルが見つかった場合、Sun MBM はそのファイルを実行します。

.post_processing.btsh によって環境変数が変更された場合でも、ジョブの終了状態には影響しません。また、ポストプロセスルーチンが強制的に中止した場合も、ジョブの終了状態には影響しません。ジョブが強制的に中止されても、Sun MBM はポストプロセスルーチンを必ず実行します。

次の例はどれも、デフォルトのサブシステムを使用することを前提としています。

例

ジョブのディスポジションオプション -d を使用して、ジョブをサブミットします。このジョブには、保持 (h) ディスポジションコードを割り当てます。つまり、ジョブは SYSIN ファイル内に保存され、実行のスケジューリングは行われません。

```
[node1/fs1] # subjob prog1 -d h
BS1100(I) Job prog1 submitted :jon = 43 class = a
.
.
```

待機オプション -w を使用して、ジョブをサブミットします。このジョブが終了するまで、制御はスクリプトに戻りません。

```
[node1/fs1] # subjob prova -w
JOB 45:Ready
JOB 45:Running
BS1113(I) Job 45 terminated.
```

この例では、次の JCL ストリームを対象として -R (指定したステップで実行を再開) オプションの使用法を示します。

```
//JOB12   JOB
//PROCLIB DD  DSN-PROC.LIB.PROC12
//STEP01  EXEC PROC=PROC12,aaa=BB
//STEP02  EXEC PROC=PROC12A
//STEP03  EXEC PGM=PGM03
//STEP04  EXEC PGM=PGM04
//STEP05  EXEC PROC=PROC12
      .
      .
//PROC12  PROC
//STEP01  EXEC PGM=PGM01
//STEP02  EXEC PGM=PGM02
//STEP03  EXEC PGM=PGM03
```

JOB12 には 5 つのステップがあり、手続き PROC12、および PROC12A が呼び出されます。PROC12 には 3 つのステップが含まれます。このジョブは、変換したあとに、-R オプションを指定してサブミットできます。

```
[node1/fs1] # subjob JOB12 -R STEP01
      (starts the job at STEP01 of JOB12)
[node1/fs1] # subjob JOB12 -R STEP02
      (starts the job at STEP02 of JOB12 - STEP01 is bypassed)
[node1/fs1] # subjob JOB12 -R STEP05.STEP02
      (starts JOB12 at STEP02 of PROC12, which is STEP05 of   JOB12)
```

susjob - ジョブの中断

susjob コマンドは、再開コマンド rsmjob が実行されるまでの間、バッチジョブの実行を中断するために使用します。

形式

susjob *jon* [-N *nodename*]

説明

jon

ジョブのオカレンス番号。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノード上で実行中のバッチジョブを、rsmjob コマンドによって再開されるまで中断する場合に指定します。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

ユーザーは、自分のユーザー ID でサブミットしたジョブだけを中断できます。別のユーザー ID でサブミットされたジョブは影響を受けません。ジョブが中断されると、そのジョブに含まれるすべての子プロセスも中断されます。

例

ジョブのサブミット、中断、再開の流れを示します。

```
[node1/fs1] # subjob prog1
BS100(I) Job prog1 submitted :jon = 38 class = a
[node1/fs1] # infact
BS1065(I) Command infact
BS1070(I) - Inform on activity names :
act1 (J038)
BS1038(I) infact command executed
 .
 .
[node1/fs1] # susjob 38
BS1048(I) Jon 38 suspended
 .
 .
[node1/fs1] # rsmjob 38
BS1049(I) Jon 38 resumed
```

unikixjob - Sun MTP へのバッチジョブのサブミット

unikixjob コマンドは、VSAM データセットへのアクセスなどの Sun MTP サービスを必要とするジョブをサブミットするために使用します。

形式

unikixjob *name* [-a] [-c *class*] [-C *cycle-time*] [-d *disp*]
[-D [COBOL|DEBUG_MACRO|MACRO|VSAM]] [-h *history-dir*] [-j] [-J *jobid*]
[-k *batchsystem*] [-m] [-N *nodename*] [-o *save-hist-env*] [-p *prior*]
[-P *job-parameter*] [-R *stepname*[.*procstepname*]] [-s *start-time*]
[-e *end-time*] [-S *stepname*[.*procstepname*]] [-W *workdir*] [-t *type*] [-T]
[-U *user*] [-v] [-w] [-x] [-X [ALL|VSTIME|WTIME|LIO|PIO]]

説明

name

サブミットするジョブの名前。指定するジョブは、ish または jmvs ディレクトリに置かれている必要があります。このコマンドで指定したジョブ名は、その他のコマンドで *jon* とジョブ名とが関連付けられているかどうかを確認するために使用します。

-a

指定したバッチジョブをデバッグできるように設定します。バッチジョブをデバッグできるのは、ユーザーが anmjob コマンドによってそのバッチプログラムにアクセスした場合だけです。

-c *class*

指定したクラス (a ～ z) に対してジョブをサブミットします。デフォルトは a です。

-C *cycle-time*

ジョブを周期的に実行するように指示します。-C オプションを指定した場合は、-eオプションも指定する必要があります。指定しないと、ジョブサイクルは実行されたまま終了しません。

-C オプションの *cycle_time* は秒数で指定します。これにより、そのサイクル時間とジョブの完了時刻に基づいてジョブが再スケジューリングされるように指示します。次に例を示します。

-C 20　ジョブは完了してから 20 秒後に再びスケジューリングされます。

-C オプションは、-d k オプションとともに使用する必要があります。ディスポジションコードが削除 (d) の場合、ジョブは最初の実行後に SYSIN ファイルから削除されます。したがって、再スケジューリングができません。

このサイクルを終了するには、chgjob コマンドを使用してジョブのディスポジションコードを変更します。次に例を示します。

d: ジョブを実行後に SYSIN ファイルから削除します。

h: 実行を停止します。

-d *disp*

ジョブにディスポジションコードを割り当てます。

d: 実行するためにジョブをディスパッチするか、作成します。デフォルト値です。

h: 指定したクラスに割り当てられているジョブはスケジューリングも実行もされず、あとで処理されるまで SYSIN ファイル内に保持されます。

k: ジョブの実行がスケジューリングされ、実行後も SYSIN ファイル内のジョブまたはシェルスクリプトが保存されます。実行後のジョブのディスポジションは、保持 (h) に変更されます。

-D

ジョブの実行中にデバッグを使用可能にします。複数のオプションを指定する場合は、コロンで区切る必要があります。

COBOL

指定したジョブで実行される COBOL ステップごとに、以下の形式の名前で Sun MBM COBOL 実行時デバッグファイルを生成します。

/tmp/*jobname.stepname.jobnumber*

この形式は、以下の環境変数の設定と同じです。

```
RTSDEBUGFILE=/tmp/$JOBNAME.$STEPNAME.$JON
```

Solaris プラットフォーム: COBOL アプリケーションが不正終了した場合は、pstack ダンプ情報もジョブの履歴ファイルに書き込まれます。

DEBUG_MACRO

デバッグマクロ出力を生成します。

MACRO

マクロを冗長モードで実行します。

VSAM

VSAM 入出力文ごとに、ファイル $KIXSYS/unikixmain.dbg にデバッグ情報を生成します。

-h *history-id*

指定した *history_id* に関連付けられている出力先ディレクトリに、ジョブ履歴ファイルを作成します。

history_id 識別子は、ユーザーが BAM を使用してサブシステムを作成する時に定義します。

ジョブ履歴ファイルの名前の形式は次のとおりです。

destination-directory/*jobname_jobnumber*.*hst*.*date*.*time*

注 – *date.time* には、LC_TIME 環境変数の定義が適用されます。詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

-j

jon を標準出力に出力することを指示します。次に例を示します。

```
JOB_NUM='unikixjob job1 -j'
```

このジョブの *jon* が JOB_NUM 環境変数として戻ります。

-J *jobid*

ジョブ識別子 *jobid* をジョブに割り当てます。このオプションは、*jobid* を外部スケジューラから Sun MBM に渡すために使用します。デフォルトの *jobid* の形式は *jobname@date.time* です。

注 – *date.time* には、LC_TIME 環境変数の定義が適用されます。詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

-k *batchsystem*

ジョブをサブミットする対象サブシステム。-k オプションの指定がないジョブは、デフォルトの Sun MBM サブシステムにサブミットされます。サブシステムの作成方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

-m

ジョブが正常に終了したかどうかについての通知メールを、(次のいずれかの形式で) そのジョブをサブミットしたユーザーに送信します。

- BQM Job *nnn* (*jobname*) could not be executed between 2:00 and 3:00
- BQM Job *nnn* (*jobname*) Ended, Status=000
- BQM Job *nnn* (*jobname*) Aborted, Status=3

説明

BQM: バッチキューマネージャー。

nnn: ジョブのオカレンス番号。

jobname: ジョブのサブミット時に指定したジョブ名。

-N *nodename*

nodename で指定した遠隔 Sun MBM ノードに対してジョブをサブミットします。

nodename をローカル Sun MBM ノード内に遠隔ノードとしてあらかじめ定義し、その遠隔 Sun MBM ノードをローカルノードの 1 つとして構成しておく必要があります。

遠隔ノードの構成方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア構成ガイド』を参照してください。

-o *save-hist-env*

ジョブの実行終了時にジョブ履歴ファイルが作成されるディレクトリを示す環境変数。

サブシステム構成ファイル内で指定された環境変数に、適切なアクセス権が付与された既存のディレクトリを設定する必要があります。

ジョブ履歴ファイルの名前の形式は次のとおりです。

save-hist-env/jobname-jobnumber.lst.date.time

注 – *date.time* には、LC_TIME 環境変数の定義が適用されます。詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

-o オプションを指定すると、ジョブの開始時刻、実行履歴、終了時刻、状態、および履歴レコードが保存されたファイルについての情報が、lststs コマンド実行時に表示されます。

-p *prior*

指定したバッチジョブに対して、属しているクラス内での優先順位を割り当てます。有効な値は 0 ～ 9 で、9 が最高優先順位です。デフォルトは 5 です。

-P *job-parameter*

$JOBPARM によってバッチジョブに渡される、25 文字以内の文字列を指定します。この -P オプションを指定しなかった場合、$JOBPARM には NULL 文字列が設定されます。このオプションによって $JOBPARM が設定された場合は、lstjob または infjbs コマンドを使用してその値を参照できます。

`-R` *stepname.procstepname*

stepname (8 文字以内) で指定したステップで、ジョブまたは手続きを再開します。

あるジョブを、特定の手続き内の特定のステップで再開するには、まずジョブのステップ名を指定し、そのあとに手続き内のステップ名 (*procstepname*) を指定します。

`-s` *start-time* `-e` *end-time*

ジョブの開始時間と終了時間。

ジョブの実行を当日の特定の時刻でスケジューリングするには、`-s` および `-e` オプションを使用します。

形式:

hh:mm

説明

hh: 24 時間表示による時。

mm: 00 〜 59 の分。

start-time が現在時刻と同じか、現在時刻よりもあとの場合、ジョブは開始されます。ジョブが *start-time* と *end-time* の間に実行されるようにスケジューリングされていない場合、そのジョブはスケジューリングされません。

end-time は、*start-time* と相関関係にあります。たとえば、次のように指定したとします。

start-time が 16:00、*end-time* が 2:00 の場合、*end-time* は次の日になります。

start-time を現在時刻よりもあとの時刻に指定した場合、たとえば *start-time* が 14:00 で現在時刻が 12:00 の場合、Sun MBM はそのジョブの実行を当日にスケジューリングします。

start-time を現在時刻よりも前の時刻に指定した場合、たとえば *start_time* が 12:00 で現在時刻が 14:00 の場合、Sun MBM はそのジョブを次の日にスケジューリングします。この規則は `-T` オプションを指定することにより無効にできます。

ただし、ジョブの開始時刻 (たとえば 12:00) をあらかじめ指定しても、その時刻に開始されない場合もあります。これはジョブのスケジューリングにさまざまな要因が関わっているためです。たとえば、次のように指定したとします。

Sun MTP が VCT を検索する間隔が 120 秒に設定されていると、このジョブは 12:02 まで開始されないことがあります。さらに、12:02 の時点で最大バッチジョブ数に達していたり、使用可能なトランザクションサーバーがなかったりした場合、このジョブは開始されません。また、使用可能なアクティビティーがない場合も、いずれかのアクティビティーが使用可能になるまでジョブをスケジューリングできません。したがって、本番稼働のスケジュール要件を確認し、適切なリソースを割り当てる必要があります。

ジョブが当日と翌日のどちらにスケジューリングされたかを確認するには、`lstjcl` または `lstjob` コマンドを使用します。時刻のあとに表示される `td` または `tm` は、それぞれ当日と翌日のどちらにスケジュールされているかを示します。`td`、`tm` のどちらも表示されていない場合、その時刻は経過し無効になったことを示します。

`-S` *stepname.procstepname*

stepname (8 文字以内) で指定した手順で、ジョブまたは手続きを中断します。中断したジョブを再開するには、`rsmjob` コマンドを使用します。

`-t` *type*

ファイルのタイプ。JCL ファイルを指定した場合、`unikixjob` コマンドはそのファイルをマクロジョブに変換してからサブシステムにサブミットします。

`m`: `mvstrans` を使用して変換される MVS JCL ジョブ。

`d`: `dostrans` を使用して変換される VSE JCL ジョブ。

`n`: バッチシェルスクリプトファイル (デフォルト)。

JCL の機能の中にはサポートされないものもあるので、適切な Sun MBM トランスレータを使用して JCL ファイルを変換する必要があります。生成されたジョブは、妥当性の確認後にサブミットできます。

`-T`

ジョブに関連付けられている時刻を、当日の時刻に適用します。たとえば、現在の時刻が 14:00 の場合、`-T` オプションを指定しないで開始時刻を 12:00 に指定したジョブをサブミットすると、そのジョブは自動的に翌日の日付でスケジューリングされます。

`-T` オプションを指定した場合は、当日のできるだけ早い時刻でジョブの実行がスケジューリングされます。

`-U` *user*

ジョブを別のユーザー名でサブミットします。

ユーザー 1 が、このコマンド行からサブミットするジョブをユーザー 2 として実行する必要がある場合は、このコマンド行か、「Submit Job」ダイアログの「`Job PARM`」フィールドで、このオプションを指定する必要があります。

ユーザー 1 がユーザー 2 としてジョブを正常に実行するには、ユーザー 2 のホームディレクトリに定義されているバッチ許可ファイル `.batch_auth` で、適切な権限がユーザー 1 に付与されている必要があります。また、ユーザー 2 はこのバッチ許可ファイルの所有者である必要があります。したがってこのファイルには、グループおよびその他のユーザーに対して読み取り専用のアクセス権が設定されていることが必須条件です。

`.batch_auth` ファイルには、以下の形式で記述する必要があります。

job-full-pathname:*username1*, *username2*, *username3*

例

ユーザー Tom は、ディレクトリ /home/batch/jcl/ish 下に定義した JOB1 と JOB2 をユーザー Ken として実行する必要があり、ユーザー Tony は、ディレクトリ /home/batch/jcl/ish 下に定義した JOB2 と JOB3 を同じくユーザー Ken として実行する必要があるとします。

この場合、ユーザー Ken は、そのホームディレクトリに .batch_auth という名前のバッチ許可ファイルを作成する必要があります。このファイルには、グループとその他のユーザーの両方に対する、読み取り専用のアクセス権を設定する必要があります。バッチ許可ファイル内に次の行が必要です。

```
/home/batch/jcl/ish/JOB1:Tom
/home/batch/jcl/ish/JOB2:Tom, Tony
/home/batch/jcl/ish/JOB3:Tony
```

同じジョブのパス名を複数の行に重複して記述することはできません。したがって、同じジョブパス名に複数のユーザーを定義する場合は、上の例のようにユーザー名をコンマまたは空白文字で区切る必要があります。

また、1 つの行を 2 行以上に延長する場合は、途中の行の末尾に継続行文字 \ (バックスラッシュ) を必ず入力してください。次に例を示します。

```
/home/batch/jobs/ish/JOB008:Tom, Tony, Frank, Jim, John, \
                            Kate
/home/batch/jobs/ish/JOB009:Tom, Tony, Frank, Jim, John, \
                            Kate, Al, Steve
/home/batch/jcl/ish/JOB1:    Tom
/home/batch/jcl/ish/JOB2:    Tom, Tony
/home/batch/jcl/ish/JOB3:    Tony
```

JOB2 のサブミット時に、Tom と Tony は次のコマンドを入力する必要があります。

```
[node1/fs1] # subjob JOB2 -W/home/batch/jcl/ish -U Ken
```

この /home/batch/jcl/ish/JOB2:エントリが、Ken のホームディレクトリ下の .batch_auth ファイル内に定義され、指定したジョブ JOB2 にユーザー Tom と Tony が定義されていれば、Sun MBM はこのジョブをユーザー Ken として実行します。

-v

バッチジョブの論理の妥当性を検査するために、アプリケーションプログラムまたは IDCAMS などのユーティリティー関数の呼び出しを行わない方式でバッチジョブを実行します。バッチジョブが実行されると、アプリケーションプログラムとユーティリティー関数から実行の正常終了を示すリターンコードのシミュレーションが行われます。このオプションにより、バッチジョブの論理の妥当性を検査できます。例については、『Sun Mainframe Batch Manager ソフトウェア移行ガイド』を参照してください。

-w

指定したジョブが完了するまで、そのあとに記述されている Sun MBM コマンドを実行しないで待機するように指示します。このオプションコマンドでは、ジョブが正常に終了した場合は値ゼロが戻り、強制的に中止された場合はゼロ以外の値が戻ります。詳細は、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』のジョブの同期に関する節を参照してください。

-W *workdir*

サブミットするジョブの親ディレクトリ。たとえば、TESTECHO というジョブが /public/test/ish に置かれている場合、ユーザーは unikixjob を使用して、/opt/batch1/test から次のようにジョブをサブミットできます。

```
[node1/fs1] # pwd
/opt/batch1/test
[node1/fs1] # unikixjob TESTECHO -W /public/test
```

指定するジョブは、ish ディレクトリに置く必要があります。ただし、ish ディレクトリは暗黙的に定義されているので、-W オプションのディレクトリパスに指定する必要はありません。

遠隔ノードにサブミットするジョブの場所を指示するには、-W オプションを -N オプションとともに指定します。unikixjob コマンドに -N オプションを指定し、-W は指定しなかった場合のデフォルト動作では、該当の遠隔 Sun MBM ノード上の $PUBLIC/ish ディレクトリ内でジョブが検索されます。

-x

バッチジョブが実行される前にプリプロセスルーチンを呼び出すか、バッチジョブの実行後にポストプロセスルーチンを呼び出します。プリプロセスおよびポストプロセスルーチンは、バッチジョブによって呼び出されます。

-x オプションを指定すると、ユーザーのホームディレクトリ内で .pre_process.btsh ファイルが検索されます。ファイルが見つからない場合、$PUBLIC/bin ディレクトリ内で pre_process.btsh ファイルが検索されます。ファイルが検出されれば、そのファイルを実行します。プリプロセスルーチンによって何らかの環境変数が設定された場合は、その値がバッチジョブに適用されます。

バッチジョブの実行終了時に、そのユーザーのホームディレクトリ内で post_process.btsh ファイルが検索されます。ファイルが見つからない場合、$PUBLIC/bin ディレクトリ内で post_process.btsh ファイルが検索されます。ファイルが検出されれば、そのファイルを実行します。

.post_processing.btsh によって環境変数が変更された場合でも、ジョブの終了状態には影響しません。また、ポストプロセスルーチンが強制的に中止された場合も、ジョブの終了状態には影響しません。バッチジョブが強制的に中止されても、ポストプロセスルーチンは実行されます。

-X

統計情報を表示します。-X オプションを次のいずれかの引数を付けて実行します。

VSTIME	VSAM コードの実行時間を表示します
WTIME	待機時間を表示します
LIO	論理入出力統計情報を表示します
PIO	物理入出力統計情報を表示します
ALL	すべての統計情報を表示します

例

次の例では 2 つのジョブをサブミットしています。

```
[node1/fs1] # unikixjob proacc
BS1100(I) Job proacc submitted :jon = 46 class = a
      .
      .
[node1/fs1] # unikixjob proacb -d h -p 8 -c b
BS1100(I) Job proacb submitted :jon = 47 class = b
```

付録 A

Sun MBM サービス

この章では、サブミットしたジョブのスケジューリング、実行、およびレポート生成のために連係して動作する Sun MBM プロセス (デーモン) について説明します。また、履歴ファイル、コンソール機能、ディレクトリについても説明します。

ノードの起動時、次のデーモンが作成されます。

bqm	バッチキューマネージャー
ebmmd	メッセージデーモン
psg_daemon	プロセスグループデーモン
lgdem	ログデーモン
vcf	仮想コンソール機能

図 A-1 に、ジョブの実行に必要な主要プロセス間の関係を示します。

bqm、psg_daemon、ebmmd、lgdem、vcf デーモンは、ノードが停止するまで有効なままです。

バッチキューマネージャーデーモン

バッチキューマネージャーデーモン (bqm) は、Sun MBM にサブミットされるジョブを管理します。

- ジョブのサブミットの要求を受信すると、そのジョブをシステム入力ファイル (sysin) に書き込みます。Sun MBM は、サブミットされたすべてのバッチジョブを sysin ファイルに保存します。
- サブミットされたジョブに関するすべての特別な要求 (ジョブ状態についてのレポート生成およびジョブ状態の変更など) を扱います。

プロセスグループデーモン

プロセスグループデーモン (psg_daemon) は、実行中のバッチジョブに関する情報を取得するため、または実行が必要なバッチジョブをサブミットするために、bqm デーモンからの要求を受信します。実行が必要なジョブの場合、psg_daemon はバッチシェルを開始するプロセスをフォークし、バッチジョブが実行されます。バッチジョブの状態に関する情報は、共用メモリーに保存されます。

メッセージデーモン

メッセージデーモン (ebmmd) は、bqm および Sun MBM コマンド (たとえば、subjob および unikixjob など) を含む Sun MBM プロセスのメッセージ交換サーバーとして機能します。

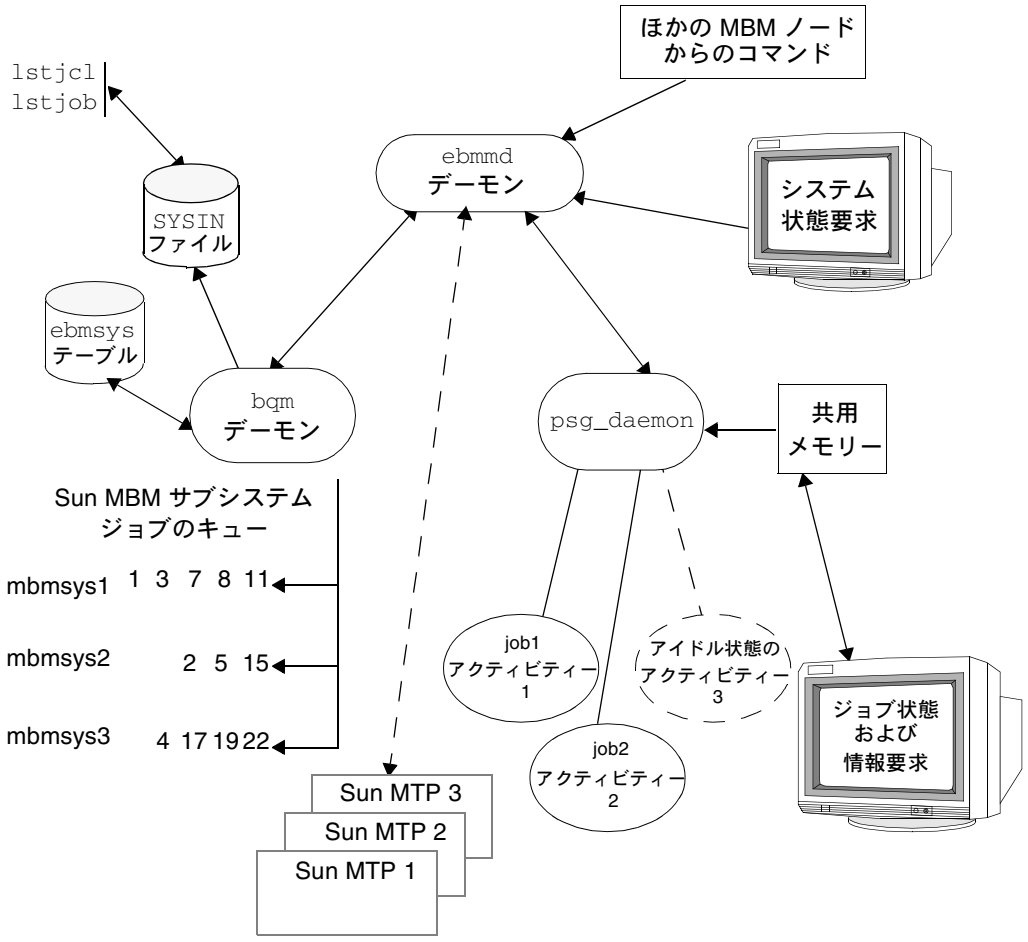

図 A-1 Sun MBM アーキテクチャー

仮想コンソール機能デーモン

次の図に示すように、仮想コンソール機能デーモン (vcf) は、バッチジョブの実行によって出力されるすべてのメッセージを処理します。vcf デーモンが処理した情報は、次の 2 つの機能によって表示または格納されます。

- 履歴ファイル
- コンソール機能

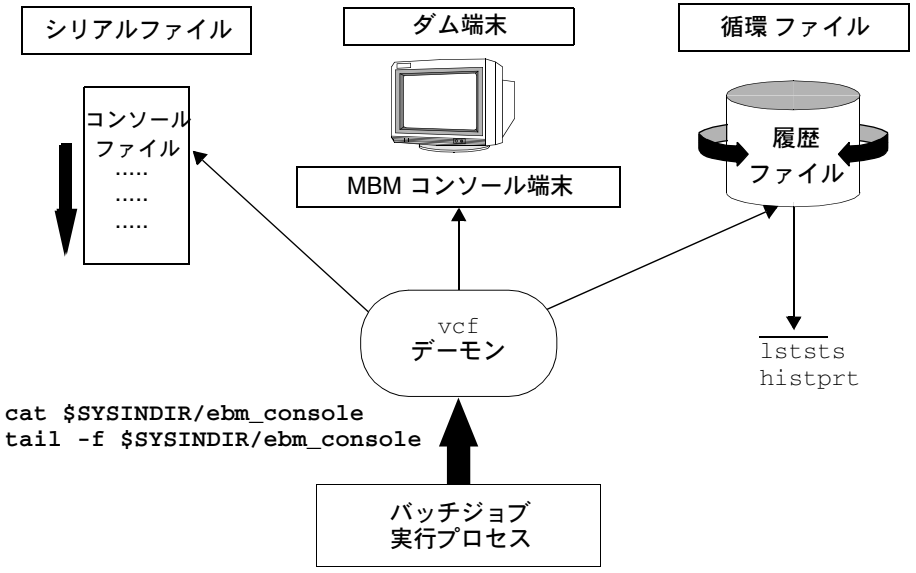

図 A-2　　vcf デーモン

履歴ファイル

履歴ファイル $SYSINDIR/HistoryFile は、ジョブの起動および終了メッセージを格納する循環ファイルです。この履歴ファイルを表示するには、Sun MBM コマンド lststs および histprt を使用します。

- lststs コマンドは、ノードの今回の起動からの履歴情報を表示します。
- histprt コマンドは、履歴ファイル消去後のノードの起動からの履歴情報を取得します。

これらのコマンドの詳細は、第 4 章を参照してください。履歴ファイルの構成方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

次に、これら 2 つのコマンドによって得られる出力の例を示します。

コード例 A-1　　histprt コマンドの出力

```
$ histprt
BS1746(I) List of the History File.User mbmuser.
Date      Uid Jon
05/02/2005 196 1 IP4305(I) Start of job JOB000 at May 02 10:41:33 2005
05/02/2005 196 1 IP4334(I) History filename:/tmp/JOB000 _0.hst.02May2005.10:41
05/02/2005 196 1 I0000 CIM500 :MTD Customer File Update Ended Normally
05/02/2005 196 1 IP4304(I) Job JOB000 terminating at May 02 10:41:33 2005
05/02/2005 196 1 IP4305(I) Start of job JOB001 at May 02 10:42:13 2002
05/02/2005 196 1 IP4334(I) History filename:/tmp/JOB001_1.hst.02May2005.10:42
05/02/2005 196 1 Mount YTD Customer Master tape
05/02/2005 196 1 Enter tape volume number:
05/02/2005 196 1 IP4363(I) The job is waiting for reply
05/02/2005 196 1 Tape volume mounted was: T100745
05/02/2005 196 1 I0000 CIM900 :YTD Customer File Update Ended Normally
05/02/2005 196 1 IP4304(I) Job JOB001 terminating at May 02 10:42:27 2005
BS1038(I)  histprt command executed
```

コード例 A-2　　lststs コマンドの出力

```
$ lststs -j1
Jon Uid Message
1 196 IP4305(I) Start of job JOB001 at May 02 10:42:13 2005
1 196 IP4334(I) History filename:/tmp/JOB001_1.hst.02May2005.10:42
1 196 Mount YTD Customer Master tape.
1 196 Enter tape volume number:
1 196 IP4363(I) The job is waiting for reply
1 196 Tape volume mounted was: T100745
1 196 I0000 CIM900 :YTD Customer File Update Ended Normally
1 196 IP4304(I) Job JOB001 terminating at May 02 10:42:27 2005
BS1707(I) End of the History File Management Facility
```

コンソール機能

コンソール機能には、システムメッセージとジョブの開始および終了のメッセージを表示および格納するために、2 つの機構が備えられています。

- コンソール端末

  MBM ノードに対してオペレータコンソールとして定義された端末デバイス。Sun MBM システムメッセージおよびジョブの開始および終了メッセージは、このコンソール端末宛に出力されます。構成に応じて、コンソール端末では特定のクラスのメッセージまたはすべてのメッセージを表示できます。

  複数のノードで同じコンソール端末を共有する場合は、各ノードに対応するメッセージを判別できるようにコンソール ID を構成する必要があります。その方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

- コンソールファイル

  `$SYSINDIR/ebm_console` ファイルはシリアルファイルです。このファイルには、コンソール端末と同じタイプのメッセージが格納されます。ユーザーは、システムコマンド `cat` または `tail` を使用して、メッセージを表示できます。

コンソールファイルは、起動時に `vcf` デーモンによって追加モードで開かれます。ファイルのサイズは大きくなることがあるので、ユーザーはファイル内容を定期的に消去する必要があります。ノードが稼動中であれば、次のコマンドを発行することによって古いファイル内容の保存および現在のファイル内容の消去ができます。

```
$ cp $SYSINDIR/ebm_console  /backup/ebm_console.date
$ cat/dev/null > $SYSINDIR/ebm_console
```

ディスク容量の確認方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

次に、MBM コンソール機能によって保存されるメッセージの例を示します。Sun MBM システムメッセージには ***、ジョブメッセージにはジョブ番号 (000、001 など) が接頭辞として追加されています。

```
[node1] :***             Starting at 02 May2005 11:17:40
[node1] :*** OS1006(I) Startup of vcf daemon executed
[node1] :*** OS1006(I) Startup of psg daemon executed
[node1] :*** BS0330(I) BQM:beginning of startup procedures
[node1] :*** BS0354(I) Found batch subsystem subsys1, processed jobs = 5
[node1] :*** BS0327(I) BQM:startup procedures successfully completed
[node1] :*** OS1006(I) Startup of bqm daemon executed
[node1] :*** BS0336(I) BQM:BQM is now idle
[node1] :*** OS1007(I) Startup Completed
[node1] :*** IP2104(I) Start of activity act1
[node1] :*** IP2033(I) Activity act1 starts with job 0
[node1] :*** BS0337(I) BQM:BQM is now running jobs
[node1] :*** IP2110(I) Starting job JOB000 at May 02 11:18:50 2005
[node1] :000 IP4305(I) Start of job JOB000 at May 02 11:18:50 2005
[node1] :000 IP4334(I) History filename:/tmp/JOB000_0.hst.02May2005.11:18
[node1] :000 I0000 CIM500 :MTD Customer File Update Ended Normally
[node1] :000 IP4304(I) Job JOB000 terminating at May 02 11:18:50 2005
[node1] :*** IP2107(I) Job JOB000 terminated at May 02 11:18:50 2005
[node1] :*** BS0336(I) BQM:BQM is now idle
[node1] :*** IP2034(I) Activity act1 is waiting for new jobs
[node1] :*** IP2033(I) Activity act1 starts with job 1
[node1] :*** BS0337(I) BQM:BQM is now running jobs
[node1] :*** IP2110(I) Starting job JOB001 at May 02 11:18:59 2005
[node1] :001 IP4305(I) Start of job JOB001 at May 02 11:18:59 2005
[node1] :001 IP4334(I) History filename:/tmp/JOB001_1.hst.02May2005.11:18
[node1] :001 Mount YTD Customer Master tape.
[node1] :001 Enter tape device number:
[node1] :001 IP4363(I) The job is waiting for reply
[node1] :001 Tape Device mounted was: T00127
[node1] :001 I0000 CIM900 :YTD Customer File Update Ended Normally
[node1] :*** IP2107(I) Job JOB001 terminated at May 02 11:19:13 2005
[node1] :001 IP4304(I) Job JOB001 terminating at May 02 11:19:13 2005
[node1] :*** BS0336(I) BQM:BQM is now idle
[node1] :*** IP2034(I) Activity act1 is waiting for new jobs
```

Sun MBM ディレクトリ

Sun MBM をインストールするディレクトリは、ユーザーが INSTEBM スクリプトの実行時に指定します。インストールプロセス中、環境変数の PACK、PUBLIC、SYSINDIR は、それぞれ、pack、public、sysindir ディレクトリを指すように、ノードのインストール (*install-dir*) ディレクトリの下に作成されます。

pack ディレクトリ

ノードの *install-dir*/pack ディレクトリには、複数の構成ディレクトリと Sun MBM オブジェクトに加えて、次の構造体があります。

bin	実行可能ファイル (bam など)。
nlsmsg	ネイティブ言語サポートメッセージファイル。
ish	その他のインストールディレクトリ。
fun	内部的に使用されるバッチユーティリティー定義。
ipsxcat	アプリケーション設定エンティティー
RTSFS	COBOL 実行時システム構築用の構成ディレクトリ。RDBMS へのアクセスをサポートする構造体もあります。
da	内部テーブルファイルを含むディレクトリ。
datefile	日付設定ファイル。BAM を使用して日付を変更する場合にのみ作成されます。

public ディレクトリ

ノードの *install-dir*/public ディレクトリには、ノードディレクトリと次の構造体が含まれます。

bin	バッチシェルスクリプトを処理するためのシステムユーティリティー。
cob	システムの COBOL ライブラリ。MVS の SYS1.LINKLIB または VSE の IJSYSRS.SYSLIB に相当します。
File_Map	メインフレームの JCL データセットを使用しているシステム内のファイルに割り当てるデフォルトのファイルマップ。
jmvs	変換対象の MVS JCL ジョブ。

dos.conf	dostrans により VSE JCL ストリームの変換時に使用される JCL ユーティリティーのリスト。
hal.conf	haltrans により MVS JCL ストリームの変換時に使用される JCL ユーティリティーのリスト。詳細は、haltrans テクニカルノートを参照してください。
ish	変換済みの JCL ジョブ (シェルスクリプト)。
msg	ログディレクトリ。
mvs.conf	mvstrans により MVS JCL ストリームの変換時に使用される JCL ユーティリティーのリスト。
proc	カタログ化されたパブリック手続き。MVS の SYS1.PROCLIB または VSE の IJSYSRS.SYSLIB に相当します。
utilm	マクロやユーティリティーの実行可能ファイル。

sysindir ディレクトリ

ノードの *install-dir*/sysindir ディレクトリには、次の構造体が含まれます。

DefJobAcct	デフォルトのジョブログアカウンティング情報ファイル。
EbmsysTable	サブシステムテーブル。
ebm_console	Sun MBM システムメッセージおよびジョブの開始および終了メッセージが書き込まれるシリアルファイル。このファイルは、BAM でコンソールファイルを構成した場合に作成されます。
HistoryFile	実行されたジョブの履歴が保存される循環ファイル。
jbatch_*string*	サブミットされたジョブファイル。これらのファイルは、対応するジョブが sysin から削除されると削除されます。
jobacct_file	ジョブアカウンティング情報ファイル。
setij.*string*	内部中間ファイル。ジョブごとに 1 つあります。これらのファイルは、対応するジョブが sysin から削除されると削除されます。
sysin	実行用にサブミットされたバッチジョブ。

用語集

A

Animator (名詞) Server Express のデバッグソフトウェア。

B

Batch Administration Manager (BAM) (名詞) Sun MBM ノードやサブシステムを設定したり管理したりするために使用されるツール。

`batchenv` ファイル (名詞) ノードの実行方法を制御する環境変数を含む設定ファイル。各ノードは固有の `batchenv` ファイルを持ち、それらを実行しないとノードが開始されません。

`bqm` 「バッチキューマネージャー」を参照。

D

`dostrans` (名詞) VSE JCL トランスレータ。

E

ebmmd　「メッセージデーモン」を参照。

EbmsysTable　(名詞) 現在定義されているすべてのサブシステム名が含まれるテーブル。*mbm-node-installed-dir*/sysindir ディレクトリにあります。

F

File_Map　(名詞) IBM データセット、ライブラリ、および世代別データグループ (GDG) と、対応する UNIX パス名とを関連付けるエントリを含む特別ファイル。このファイルは、Sun MBM JCL トランスレータおよびサブシステムによって、メインフレーム JCL ストリームの変換時やマクロジョブスクリプトの実行時に使用されます。サブシステムはそれぞれ 1 つの File_Map に関連付けられています。

I

.install ファイル　(名詞) インストールされたノードおよびそれに関連付けられたサブシステムすべての情報を含んだグローバル Sun MBM 構成ファイル。.install ファイルは、ノードをインストールディレクトリにインストールするときに作成されます。.install ファイルは、ノードを開始するたびに読み込まれます。

J

jon　(名詞) ジョブのオカレンス番号。ジョブがサブミットされると、Sun MBM は一意の 3 桁のオカレンス番号を割り当てます。*jon* は、さまざまなコマンドに参照されます。

K

KIXSYS　(名詞) システムテーブルが位置する Sun MTP 領域のディレクトリを示す環境変数。Sun MBM は、この値を使用して領域に接続します。

M

mvstrans　(名詞) MVS JCL トランスレータ。

P

psg_daemon　「プロセスグループデーモン」を参照。

S

Sun Mainframe Batch Manager ソフトウェア (Sun MBM)　(名詞) 制御された環境でバッチジョブを実行するための機能を提供するバッチマネージャー製品。Sun MBM は、バッチ作業負荷を処理し、割り当てられたパラメータ (開始時刻、バッチプロセスの最大数、ジョブの優先順位など) を基にジョブをスケジューリングします。

Sun Mainframe Transaction Processing ソフトウェア (Sun MTP)　(名詞) プロセス間通信サービス、ソケット、COBOL、C、PL/I などの機能を使用してアプリケーションを実行するユーザーアプリケーション。クライアント以外の Sun MTP のすべてのコンポーネントは、メインサーバープロセスである unikixmain によって起動します。

Sun MTP 領域　(名詞) システム上の異なるアプリケーションを定義するプロセス変数、リソース変数、および環境変数のセット。

sysin ファイル　(名詞) システム入力ファイル。このファイルには、サブミットされたジョブのうち実行待ちのものすべてが格納されます。ノードの停止またはシステムの強制的な中止時に sysin ファイル内に置かれているジョブがあった場合は、ノードの再起動時にそのジョブを実行するためのスケジューリングが行われます。

V

VSAM 構成テーブル (VCT)
(名詞) 基本の Sun MTP 構成パラメータを定義する制御テーブル。

あ

アクティビティー
(名詞) ジョブを実行する共用メモリーのセグメント。アクティビティーはジョブクラスに割り当てられます。1 つのクラスに 99 までのアクティビティーを割り当てることができます。

え

エラー表
(名詞) Solaris オペレーティングシステムに付属するインクルードファイルのこと。このファイルへのアクセス方法が分からない場合は、システム管理者にお問い合わせください。

エラーログ
(名詞) Sun MBM によって、ジョブ実行に関わる Sun MBM デーモンおよびバッチプロセスからデバッグメッセージが収集されたファイル。

か

仮想記憶アクセス方式 (VSAM)
(名詞) さまざまなアクセス方式によってレコードにアクセスする方式。

ESDS (入力順データセット) では、レコードは順次に記録され、アクセスされます。

RRDS (相対レコードデータセット) では、レコードはデータセット内で占める位置番号によって検索されます。

KSDS (キーシーケンスデータセット) では、レコードは索引またはキーによって検索されます。

仮想コンソール機能 (vcf) (名詞) バッチジョブ実行によるすべての出力メッセージを処理するデーモン (`vcf`)。

環境変数 (名詞) プログラムファイルおよびアプリケーションの位置を定義する変数。クライアントとサーバーは、どちらも環境変数を使用します。

き

許可ファイル (名詞) ユーザーの、ノードを開始、管理、停止する権限、サブシステムを管理する権限、およびクラスやアクティビティーを作成、変更、削除する権限を制御するファイル。

く

クラス (名詞) 1 つまたは複数のアクティビティーを含む概念エンティティー。ジョブはクラスにサブミットされ、アクティビティーが利用可能な場合に実行されます。1 つのノードで 26 クラスをサポートできます。

け

検査する (動詞) MVS または VSE トランスレータのどちらかを使用して JCL ジョブを実行し、ファイルやプログラムがすべて存在するか検査すること。検査モードでジョブを実行すると、`File_Map` が 1 つ作成されます (すでにファイルが存在している場合は、追加されます)。

こ

コンソール端末 (名詞) ノードにオペレータコンソールとして定義される端末デバイス。さまざまなタイプのエラーメッセージを設定して、コンソール端末に表示できます。

コンソールファイル (名詞) Sun MBM コンソール端末に表示されるものと同じメッセージタイプを格納する連続したファイル。

さ

サブシステム (名詞) 特定のノードに従属する環境であり、ここで特定のタイプのジョブが実行されます。たとえば、Sun MTP 領域の VSAM データセットにアクセスするジョブを実行するためのサブシステムを作成できます。

サブシステム設定ファイル (名詞) サブシステムの作成時、BAM は 2 つの設定ファイルを作成します。1 つは読み取り専用ファイルで、サブシステム作成時に設定される環境変数を含んでいます。もう 1 つはユーザー編集可能ファイルで、ほかの環境変数を追加できます。これらのファイルはそれぞれ、`$SETUP` および `$USER_SETUP` として参照されます。

し

シグナル表 (名詞) Solaris オペレーティングシステムに付属するインクルードファイルのこと。このファイルへのアクセス方法が分からない場合は、システム管理者にお問い合わせください。

ジョブクラス 「クラス」を参照。

ジョブに対するディスポジション (名詞) ジョブの状態。ジョブのディスポジションは次のいずれかになります。

ジョブを実行するためにディスパッチ (スケジューリング) された状態。実行後、ジョブは `sysin` ファイルから削除されます (デフォルト)。

`sysin` ファイルに保持され、実行用にスケジューリングされていない状態。

ディスパッチされるとともに保持されている状態。ジョブは実行用にスケジューリングされます。完了したジョブは、`sysin` ファイルに保持されます。

ジョブの履歴ファイル (名詞) バッチジョブの実行中に収集されたシステムメッセージとアプリケーションメッセージを含むファイル。「ジョブ出力一覧ファイル」とも呼ばれます。

す

スレッド 「アクティビティー」を参照。

せ

世代別データグループ (GDG) (名詞) 複数の物理ファイルで構成される単一の論理ファイル。「世代別データセット」とも呼ばれます。IBM ファイルタイプです。

て

デフォルトのサブシステム (名詞) ジョブのサブミット時にサブシステムが指定されていない場合に、ジョブが実行されるサブシステム。

の

ノード (名詞) Sun MBM ソフトウェアの一意のインストール。

は

バッチキューマネージャー (名詞) Sun MBM サブシステムにサブミットされたジョブを管理するデーモン (`bqm`)。

ふ

ファイルシステム (名詞) 物理ディスクドライブを「パーティション」という小単位の領域に分割する機能。パーティションには、ファイルシステム、スワップ空間、ブートセクタその他の情報を含めることができます。

ファイルのアクセス権 (またはモード) (名詞) オペレーティングシステムの定義に従って、ファイルへのアクセスを制御します。

プロジェクト　(名詞) ジョブと手続きのユーザー定義グループであるジョブエディタ構成概念。

プロセスグループデーモン　(名詞) `bqm` デーモンから要求を受け取り、実行されるバッチジョブについての情報を取得したり、バッチジョブをサブミットしたりするデーモン。

へ

変換する　(動詞) MVS または VSE トランスレータのどちらかを使用して JCL ジョブを実行し、JCL を Sun MBM マクロジョブスクリプトに変換すること。

ま

マクロ文　(名詞) MVS または VSE JCL ジョブおよび手続きの変換によって生成される文。

マニュアルページ　(名詞) `man` コマンドを使用して、コマンドの使用方法を表示できます。たとえば、`grep` コマンドについて表示するときは、プロンプトで `man grep` と入力します。

め

メッセージデーモン　(名詞) Sun MBM プロセスのメッセージ交換サーバーとして機能するデーモン (`ebmmd`)。

ゆ

優先順位　(名詞) ジョブがサブミットされると、明示的または暗黙的に優先順位が割り当てられます。有効な値は `0` 〜 `9` までで、`9` が最優先となります。

り

履歴ファイル　(名詞) ジョブの開始およびジョブの終了メッセージを含む循環ファイル。

る

ルートファイルシステム　(名詞) オペレーティングシステムとその関連ファイルが格納されています。ルートファイルシステムは、完全なファイル名の最初の文字として、スラッシュ (/) で参照されます。

ろ

ログデーモン　(名詞) コンソールメッセージのログ記録を管理するデーモン (`lgdem`)。

索引

E

F

G

H

I

J

K

L

M

か

き

く

け

こ

さ

し

す

せ

そ

た

ろ