Sun™ Mainframe Batch Manager ソフトウェア 移行ガイド

Release 10.1.0

Sun Microsystems, Inc.
www.sun.com

Part No. 819-2508-10
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 Migration Guide*
Part No: 817-7443-10
Revision A

目次

図目次

表目次

コード例

はじめに

このマニュアルでは、MVS JCL および VSE JCL バッチジョブを Sun™ Mainframe Batch Manager (Sun MBM) 環境に移行する場合に必要な情報について説明します。

マニュアルの構成

このマニュアルは、次の章から構成されています。

第 1 章では、メインフレームからのジョブの移行プロセスを紹介します。

第 2 章では、`File_Map` について説明します。メインフレームのデータセット名を Sun MBM ジョブからアクセス可能なファイル名に割り当てるために使用します。

第 3 章では、Tool Kit について説明します。Tool Kit は、Sun MBM 環境でタスクを実行するグラフィカルユーザーインタフェース (GUI) です。

第 4 章では、`mvstrans` で MVS JCL メインフレームのジョブと手続きを変換する方法を説明します。

第 5 章では、`dostrans` で VSE JCL メインフレームのジョブと手続きを変換する方法を説明します。

第 6 章では、Job Editor を使用して MVS JCL をインポートする方法を説明します。

第 7 章では、Sun MBM でのさまざまな JCL およびソートのユーティリティーのサポート状況を説明します。

第 8 章では、アプリケーションプログラムを移行する方法を説明します。

第 9 章では、Sun MBM が外部 CICS インタフェース (EXCI) を実装する方法を説明します。

付録 A では、バッチシェルユーティリティーとその機能を説明します。

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』

書体または記号*	意味	例
「 」	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。

- ディレクトリ、または 1 つ以上のディレクトリを指定できる環境変数
- 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 リファレンスマニュアル』	819-2360-10
	『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』	819-2509-10
	『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	*

* マニュアルの注文については、ベンダー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 移行ガイド』、Part No. 819-2508-10

第1章

ジョブの移行の概要

Sun MBM 環境でジョブ制御言語 (JCL) 文から構成されたメインフレームジョブを実行するには、あらかじめジョブを移行する必要があります。この章では、ジョブを移行する方法について説明します。この章の内容は、次のとおりです。

移行プロセス

ジョブの移行は、複数のステップで構成されるプロセスです。次の手順では、関連のあるハイレベルのステップを示します。詳細な手順は、マニュアルの該当する節を参照してください。

▼ ジョブを移行する

1. **Sun MBM 製品をインストールします。**

   詳細は、『Sun Mainframe Batch Manager ソフトウェア インストールガイド』を参照してください。

2. **Sun MBM ノードを作成および構成します。**

   詳細は、『Sun Mainframe Batch Manager ソフトウェア インストールガイド』および『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

3. **ジョブのディレクトリ構造を作成し、そこにソースファイルを格納します。**

   MVS から移行する場合は 44 ページの「ディレクトリ構造の作成」を、VSE から移行する場合は 138 ページの「ディレクトリ構造の作成」を参照してください。

4. **サブシステムを作成します。**

   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。

   BAM を使用してサブシステムを作成すると、少なくとも次の環境変数が設定されます。

EBMSYS	Sun MBM サブシステム名
FILEMAP	File_Map ファイルのパス名
JOBLIB	アプリケーションプログラムのデフォルトディレクトリ
JCLLIB	変換したジョブスクリプトのデフォルトディレクトリ
PROCLIB	変換した手続きのデフォルトディレクトリ
SETUP	Sun MBM サブシステム設定ファイル
SYSOUTDIR	システム出力データセットのディレクトリ
USER_SETUP	サブシステムのユーザー設定ファイル

5. **検査モードでトランスレータを実行して File_Map を作成します。詳細は、図 1-1 を参照してください。**

   必要に応じて File_Map を編集します。詳細は、第 2 章を参照してください。

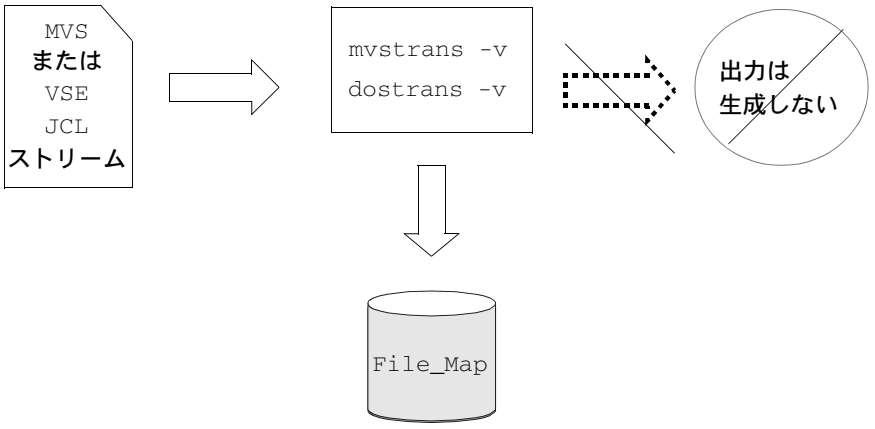


図 1-1 File_Map 検査モードのプロセス

6. **トランスレータを実行してジョブのマクロコードを作成します。詳細は、図 1-2 を参照してください。**

   トランスレータで MVS JCL と VSE JCL からマクロコードを作成する例は、それぞれ第 4 章と第 5 章を参照してください。トランスレータのコマンドの構文は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

7. **アプリケーションプログラムを移行します。**

   詳細は、第 8 章を参照してください。

8. **ジョブを実行します。**

   詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

   `File_Map` の更新後、JCL を変換してマクロジョブスクリプトを生成できます。

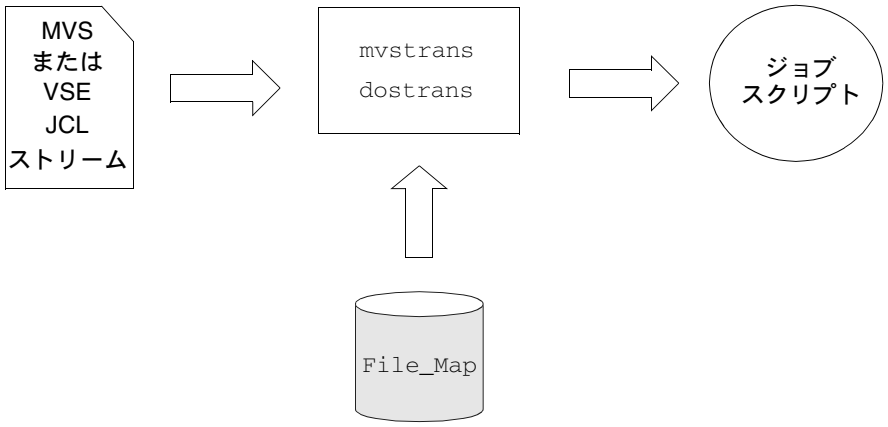


図 1-2　　Sun MBM のジョブ作成フロー

トランスレータの動作

MVS と VSE のトランスレータは、事前定義されたディレクトリのジョブと手続きを読み取って、JCL をマクロジョブスクリプトに変換してから、スクリプトを事前定義されたディレクトリの別のセットに保存します。トランスレータは、検査モードと変換モードの 2 つのモードで動作します。

- 検査モードでは、トランスレータは JCL 内のデータセット名を UNIX 環境に配置可能な名前に割り当てます。
- 変換モードでは、トランスレータは JCL 文からマクロジョブと手続きのスクリプトを作成します。

第2章

ファイルの割り当て

この章では、File_Map とその操作方法について説明します。次のトピックがあります。

Sun MBM でのファイル割り当て方法

File_Map ファイルには、IBM データセット、ライブラリ、および世代データグループ (GDG) を、Sun MBM 環境の対応するファイルに割り当てるエントリのセットが含まれています。

File_Map は、サブシステムを作成するときに作成されます。各サブシステムには独自の File_Map ファイルが含まれています。ファイルの場所は、FILEMAP 環境変数で定義します。File_Map エントリは、検査モード (-v) で JCL トランスレータ mvstrans または dostrans を実行するときに作成されます。

File_Map は、JCL のジョブまたは手続きの変換時に、mvstrans と dostrans で使用されます。また、MVS JCL のインポート時には Job Editor で使用され、変換済みスクリプトの実行時には Sun MBM ランタイムで使用されます。

File_Map へのエントリの作成

File_Map にエントリを作成するには、JCL トランスレータの -v (検査モード) オプションを使用します。File_Map のエントリは、JCL ストリームの内容を基に作成されます。FMROOT 環境変数とトランスレータのコマンドオプションを使用して、エントリの作成方法を制御できます。

- FMROOT 環境変数は、デフォルトのベースディレクトリを File_Map に設定する場合に使用します。詳細は、mvstrans を使用する場合は 46 ページの「FMROOT 環境変数の設定」を、dostrans を使用する場合は 140 ページの「FMROOT 環境変数の設定」を参照してください。
- File_Map を作成する前に、TRANSOPTS 環境変数と、エントリの作成に必要なトランスレータオプションを設定できます。詳細は、mvstrans を使用する場合は 47 ページの「TRANSOPTS 環境変数の設定」を、dostrans を使用する場合は 141 ページの「TRANSOPTS 環境変数の設定」を参照してください。

注意 – File_Map エントリの作成に使用するオプションは、トランスレータでジョブと手続きのスクリプトを作成するときには指定しないでください。

dostrans と mvstrans のオプションについての詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

File_Map の各データセットエントリには、カタログ名が設定されています。デフォルトでは、すべてのデータセットカタログはシステムカタログ MASTERCAT に属しています。MVS JCL の JOBCAT 文と STEPCAT 文、VSE JCL の LIBDEF 文と DLBL 文はデフォルトのカタログ名を上書きします。

File_Map の新規ファイルエントリに作成されるデフォルトのファイルタイプは FS (Sun MTP VSAM ファイル以外のファイル) です。File_Map エントリを編集するには、次のいずれかの方法を使用します。

- 「BAM Application and Subsystems」メニューで、オプション「3 Edit Subsystem's File_Map」を選択します。
- vi などのテキストエディタを使います。
- Sun MBM Tool Kit を使います。このツールキットでは、グラフィカルユーザーインタフェース (GUI) を使って File_Map を編集できます。詳細は、23 ページの「Tool Kit を使った File_Map の更新」を参照してください。
- 複数の File_Map エントリの場合は cfm コマンドを使います。詳細は、24 ページの「cfm コマンドを使った File_Map の更新」を参照してください。
- crtflm コマンドを使います。詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

File_Map の形式

MVS と VSE JCL のどちらを使用しているかによって、JCL 文は異なります。

MVS JCL の場合

- DD 文を使用してデータセットを記述します。
- DD 文の DSN パラメータで、データセット名を指定します。
- JOBCAT 文または STEPCAT DD 文で、データセットを検索するカタログを指定します。
- JOBLIB DD 文で、JCL ストリームに必要なライブラリを指定します。

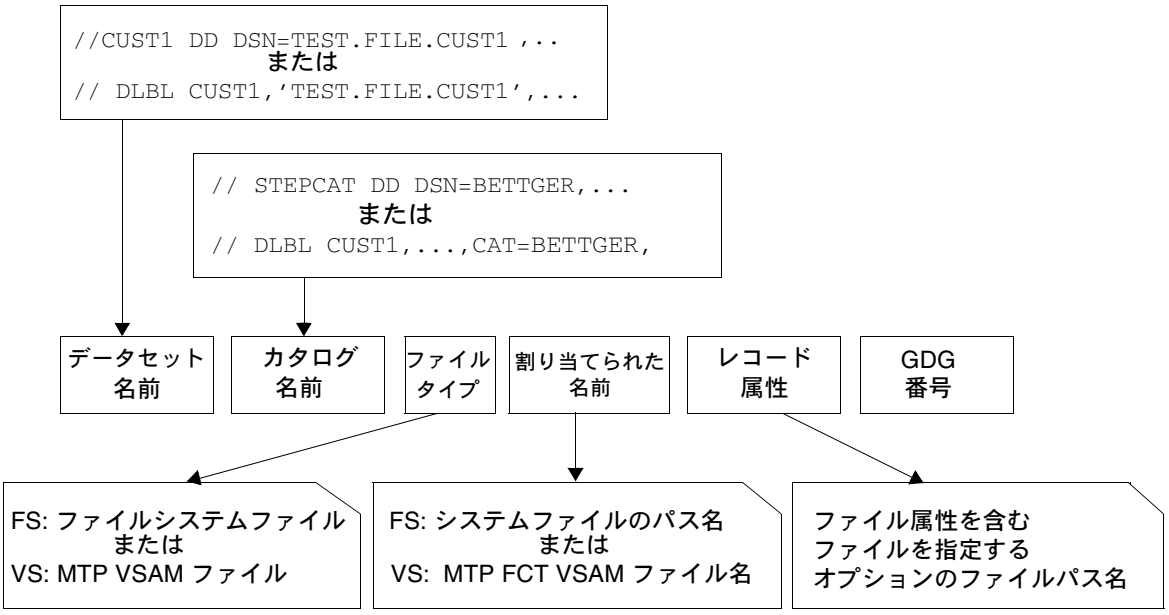
mvstrans では、これらの JCL 文を検証し、データセット、カタログ、およびライブラリ情報を抽出して File_Map に新規エントリを作成します。

VSE JCL の場合

- DLBL 文を使用して、カタログ化されたデータセットを記述します。
- DLBL 文の CATALOG オペランドで、データセットを検索するカタログを指定します。
- DLBL 文の FILE-ID オペランドで、データセット名を指定します。

dostrans では、これらの JCL 文を検証し、データセット、カタログ、およびライブラリ情報を抽出して File_Map に新規エントリを作成します。

図 2-1 に、データセットの File_Map エントリ様式の例を示します。



File_Map のエントリ

```
TEST.FILE.CUST1;MASTERCAT;FS;$SEQFILES/test/file/cust1;;0;
```

図 2-1　　データセットの File_Map エントリ様式

実行可能プログラムの配置に使用するデフォルトシステムライブラリ SYS1.LINKLIB は、Sun MBM サブシステムを作成するときに File_Map に作成されます。図 2-2 に、アプリケーションプログラムを含むライブラリの File_Map エントリ様式を示します。

File_Map が作成されたら、SYS1.LINKLIB エントリを手動で更新し、ディレクトリパスを追加するエントリや、実行可能プログラムを検索するエントリを含めることができます。各ディレクトリパスを区切るには、コロンを使用します。次に例を示します。

```
SYS1.LINKLIB;__LIB;FS;/app1/pgm/cobol1:/app1/pgm/cobol2;;0;
```

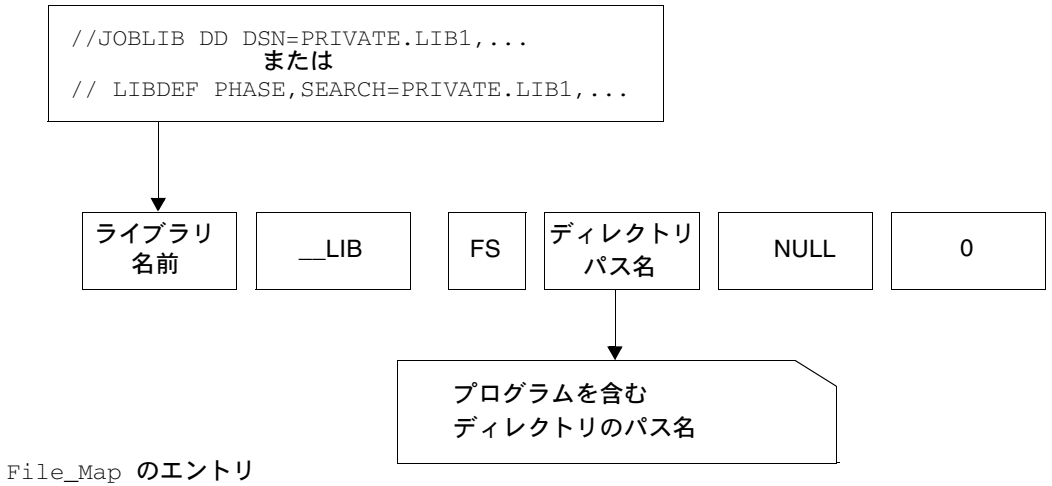


図 2-2　　ライブラリの File_Map エントリ様式

File_Map 内のエントリのタイプは、定義するファイルのタイプによって異なります。たとえば、JCL でパス名の一部に記号パラメータを指定した場合、Sun MBM では環境変数にそのパラメータを含む File_Map エントリが生成されます。バッチジョブでも、その環境変数が参照されます。

サポートされるファイルタイプ

Sun MBM では次のファイルタイプをサポートします。

FS	順編成ファイルシステムファイル
VS	Sun MTP VSAM ファイル
GDG	世代データグループ (GDG)。世代番号を含みます COBOL 環境でのみサポートされます

順編成ファイルシステムファイル (FS)

Sun MBM 環境では、順編成ファイルシステムで管理されるファイルをサポートします。File_Map の「Type」フィールドに FS 値で指定します。FS ファイルは、いずれも Sun MTP 領域 (VSAM ファイル) では管理されません。FS ファイルのレコード形式は、レコード属性 File_Map エントリを使って記述します。詳細は、23 ページの「Tool Kit を使った File_Map の更新」を参照してください。

例

MVS JCL 文の場合

```
//DD1 DD DSN=&DREGN..APPL.CARDS.....
```

VSE JCL 文の場合

```
// DLBL DD1,'&DREGN..APPL.CARDS'.....
```

File_Map のエントリは次のとおりです。

```
&DREGN..APPL.CARDS;MASTERCAT;FS;/tmp/${DREGN}/appl/cards;;0;
```

Sun MTP VSAM ファイル

Sun MTP 領域で管理されるファイルは VSAM ファイルです。Sun MTP では、次の 3 つのタイプをサポートします。このタイプは File_Map の「Type」フィールドに VS 値で指定します。

- キーシーケンスデータセット (KSDS)
- 相対レコードデータセット (RRDS)
- 入力順データセット (ESDS)

VSAM ファイルの場合、「Mapped Name」の File_Map エントリは、Sun MTP 領域のファイル制御テーブル (FCT) と VSAM カタログに定義されている VSAM データセット名です。

例

Sun MTP FCT に CUSTMAST として定義されている VSAM データセットが MVS JCL 文で次のように参照される場合、

```
//CUSTMAST DD DSN=TEST.VSAM.CUSTMAST,DISP=SHR
```

または VSE JCL 文で次のように参照される場合、

```
// DLBL CUSTMAST,'TEST.VSAM.CUSTMAST',VSAM
```

次のように変換されます。

```
TEST.VSAM.CUSTMAST;MASTERCAT;FS;/tmp/test/vsam/custmast;;0;
```

次の例に示すように、File_Map のエントリを修正して、「Type」を VS に変更し、「Mapped Name」を CUSTMAST に変更する必要があります。

```
TEST.VSAM.CUSTMAST;MASTERCAT;VS;CUSTMAST;;0;
```

GDG ファイル

GDG とは、発生順に関連付けられた、同じデータセット名を使用するデータセットのコレクションです。Sun MBM では、順編成ファイル (FS) に割り当てられた GDG のみをサポートします。

注 – GDG は、COBOL アプリケーションプログラムのためにのみサポートされます。

例

会計パッケージでは、週末ごとにその週の実績に関する情報を含む新しいデータセットを作成します。月末には、別のプログラムを使って各週のデータを収集し、月次報告書を印刷します。週次プログラムでは月次と同じファイルを参照します。週末の処理では、手動でファイルをコピーする代わりに GDG を使ってファイルを自動でコピーします。GDG データセットは、接尾辞の数字で参照されます。MVS JCL では、GDG データセットは次のように表示されます。

```
//DD1  DD  DSN=TOTAL(-1)
```

または

```
//DD1  DD  DSN=TOTAL(+1)
```

- 値 0 (括弧内) は、現在の世代を表します
- 負の値は既存の世代を表します
- 正の値はまだカタログ化されていない新しい世代を表します

次の例では、接尾辞のないベース名 GDG としてデータセットを参照しています。

```
//DD1  DD  DSN=TOTAL
```

これは、生成された全データセットの合計、または GDG の全ファイルの連結を意味します。GDG に含むことができるファイルの最大数は、File_Map にある GDG エントリの世代番号フィールドに指定されます。GDG データセットの File_Map エントリタイプの詳細は、17 ページの「File_Map の内容」を参照してください。

ベース名 GDG

GDG はジョブの実行中に一時ファイルを使ってサポートされます。一時ファイルは、ジョブ終了時に GDG の固定メンバーとしてカタログ化されます。一時ファイルは、GDG データセットのベース名と .tmp*n* の接尾辞で生成されます。*n* は、参照される GDG のオカレンスです。

固定 GDG ファイルは、接尾辞 *_nn* 付きで作成されます。*nn* は、GDG のオカレンス数です。

次に例を示します。

GDG がカタログ化される場合、total.tmp1 は名前が次のように変更されます。total_00
total_00 は、名前が total_01 などに変更されます。
そのあと、total.tmp2 は次にコピーされます。total_00
total_00 は、名前が total_01 などに変更されます。

Sun MBM では、次のベース名 GDG (最大 128 メンバー) をサポートします。

- DELETE ディスポジション処理。メインフレーム動作がサポートされます。ベース名が DELETE の DISP で定義されている場合、すべてのオカレンスが削除されます。
- SORT。詳細は、225 ページの「ベース名 GDG ファイルのソート」を参照してください。
- IDCAMS。詳細は、189 ページの「REPRO」を参照してください。
- COBOL プログラムアクセス。詳細は、253 ページの「連結データセットとベース名 GDG へのアクセス」を参照してください。

次の表は、GDG のオカレンスが 3 であるとみなして、次のことを表しています。

- 一時データセットがいつ参照されるか
- 既存のデータセットがいつ参照されるか
- ジョブ終了時にデータセットがどのように変更されるか

JCL	実行	ジョブの終了
(-2)	_02	DD 文の DISP パラメータで指定した DELETE
(-1)	_01	DD 文の DISP パラメータで指定した DELETE
(0)	_00	_02
(+1)	.tmp1	_01
(+2)	.tmp2	_00

説明	
JCL	元の JCL に指定された GDG の参照番号。
実行	バッチシェルスクリプトの実行時に列 1 に関連付けられる GDG オカレンス。既存の GDG オカレンスつまり (-1) または (0) の場合は、実際の固定 GDG オカレンスファイルが参照されます。新規オカレンスつまり (+1) または (+2) の場合は、Sun MBM の実行時に一時 GDG ファイルが作成されます。
ジョブの終了	元の JCL と比較されるジョブ終了時の GDG オカレンスと実行時の GDG オカレンスの関係。たとえば、JCL 列の GDG オカレンス (+1) では、実行時に一時 GDG オカレンス .tmp1 が生成されます。ジョブの終了時に、.tmp1 は、元の _01 ファイルと置き換わって固定 GDG オカレンス _01 にコピーされます。

GDG のすべてのデータセットが 1 つの DD 文で参照される場合、MVS JCL トランスレータでは、GDG に関連するすべての順編成ファイルの連結リストに環境変数 DD_ddname を設定します。COBOL プログラムの実行中に、Sun MBM COBOL 実行時システムは、GDG の複数のファイルを 1 つのファイルと同様に処理します。

新規オカレンスを作成する手順が正常に終了すると、ジョブの終了時に GDG がカタログ化されますが、正常に終了しなかった場合には、GDG はカタログ化されません。COBOL プログラムでファイルがオープンされなくても、GDG はカタログ化されます。Sun MBM 管理者は、サブシステムのユーザー設定ファイル ($USER_SETUP) の EBM_NOCATALOG_EMPTY_GDG 環境変数を設定すると、このアクションを変更できます。

ASSGNDD マクロで GDG を使用する方法は、『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。

GDG ファイルが MVS JCL 文に定義されている場合、

```
//DD2  DD  DSN=TEST.GDG.FILEA(+1),DISP=SHR
```

次のように変換されます。

```
TEST.GDG.FILEA;MASTERCAT;FS;/tmp/test/gdg/filea;;9;
```

File_Map のエントリを変更して、世代番号 (GDG Number) をこの世代グループで使用可能な最大世代番号に変更する必要があります。

```
TEST.GDG.FILEA;MASTERCAT;FS;/tmp/test/gdg/filea;;12;
```

記号パラメータを含む GDG 名

この節では、名前に記号パラメータが含まれる GDG を処理するように File_Map を更新する方法について説明します。

次の例では、ジョブ内の MVS JCL を示します。

```
//JOB1   JOB
//STEP0001 EXEC PROC=TESTPROC,GDGFIL=GDG11
//STEP0002 EXEC PROC=TESTPROC,GDGFIL=GDG22
```

次の例では、ジョブによって呼び出された手続き内の MVS JCL を示します。GDG 名 &GDGFIL には、記号パラメータが含まれています。

```
//TESTPROC PROC GDGFIL=
//PSTP000 EXEC PGM=IEFBR14
//SYSUT1 DD DSN=&GDGFIL(+1),DISP=OLD
//PSTP001 EXEC PGM=IEBCOPY
//SYSUT1 DD DSN=&GDGFIL,DISP=OLD
//SYSUT2 DD DSN=NEWFILE,DISP=(NEW,KEEP,KEEP)
```

ジョブの手順またはジョブによって記号パラメータの値が異なる可能性がある場合、次の手順を実行し、実行時に正しいデータセットが特定されるようにします。

▼ 記号パラメータで参照される GDG の File_Map を構成する

1. **検査モードで mvstrans を実行して、File_Map に GDG のエントリを作成します。**

   例の JCL を使用して、次のコマンドを実行します。

   ```
   $ mvstrans TESTPROC -v
   ```

   File_Map 内のエントリは次のようになります。

   ```
   &GDGFIL;MASTERCAT;FS;/test/data/${GDGFIL};;9;
   ```

2. **変換モードで mvstrans を実行し、マクロジョブおよび手続きのスクリプトを生成します。**

3. **展開可能な &GDGFIL を持つ値を判別します。**

   この例では、値は GDG11 および GDG22 です。

4. **展開されたエントリを File_Map に追加し、最大世代を正しい値に設定します。**

```
GDG11;MASTERCAT;FS;/test/data/GDG11;;52;
GDG22;MASTERCAT;FS;/test/data/GDG22;;7;
```

5. **File_Map の元のエントリを削除します。**

   これは手順 1 で作成されたもので、不要なエントリです。

   ジョブが実行されるときに、&GDGFIL という名前で File_Map エントリが見つからない場合、ASSGNDD マクロは、この例の展開された値 GDG11 または GDG22 を使用して File_Map を検索します。

File_Map の表示

▼ File_Map を表示する

1. **Sun MBM メインメニューを起動し、Sun MBM ノードが実行されていることを確認します。**

   その方法については、『Sun Mainframe Batch Manager ソフトウェア インストールガイド』を参照してください。

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

3. **「System Status」画面が表示されたら、サブシステム名を選択します。**

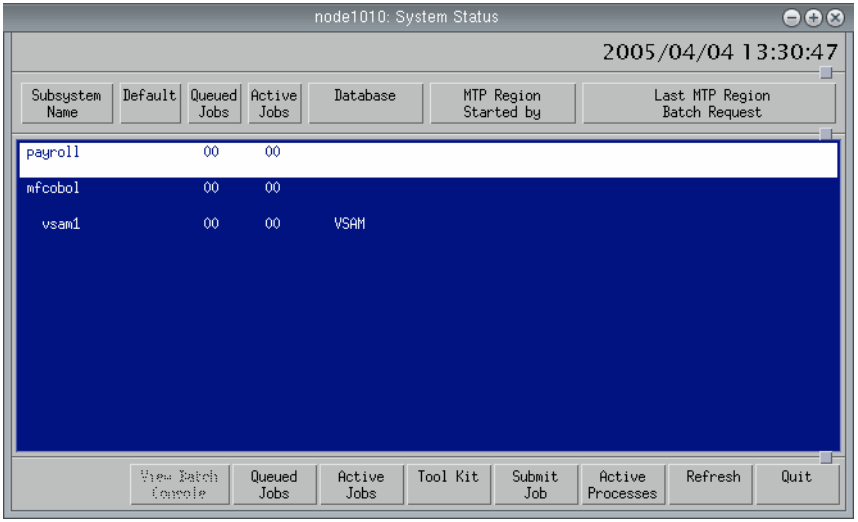


図 2-3　　System Status 画面

4. 「Tool Kit」ボタンをクリックします。

図 2-4 に示された「Tool Kit」画面が表示されます。

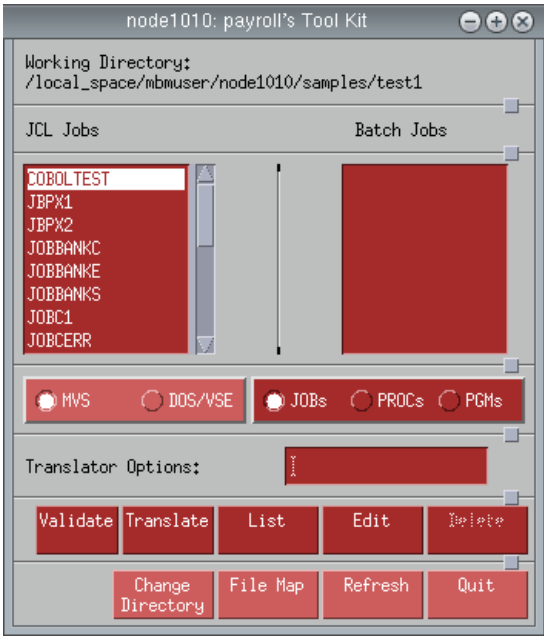


図 2-4　　Tool Kit 画面

5. 「File Map」ボタンをクリックします。

選択したサブシステムの File_Map が表示されます。

File_Map エントリの各フィールドについては、17 ページの「File_Map の内容」を参照してください。

次のボタンが画面に表示されます。

ボタン	機能
View Entry	データセットのエントリを表示します。詳細は、20 ページの「File_Map エントリの表示」を参照してください。
Update Entry	File_Map エントリを更新します。詳細は、23 ページの「Tool Kit を使った File_Map の更新」を参照してください。
Set Query Options	表示する File_Map エントリをフィルタリングするオプションを指定できるウィンドウを表示します。詳細は、21 ページの「File_Map の照会」を参照してください。
Refresh	最新の File_Map を表示します。
Quit	ウィンドウを閉じます。

File_Map の内容

File_Map のエントリ用に以下の情報を表示できます。レコード属性ファイル名および GDG 番号フィールドは、アプリケーション環境で要求されたときに使用されます。

Dataset Name

このフィールドには、データセットの名前が含まれます。

- *dataset-name*

  MVS JCL の場合: DD 文の DSN= パラメータ。

  VSE JCL の場合: DLBL 文のファイル識別子。

- *library-name*

  MVS JCL の場合: JOBLIB、STEPLIB、PROCLIB。

  VSE JCL の場合: LIBDEF 文。

- *generation-data-group-name*

  MVS JCL の場合: DD 文の DSN= パラメータ。

- *jobname-dataset-name*

  一時データセットの場合

Catalog Name

データセットのカタログ名。デフォルトでは、すべてのデータセットは MASTERCAT カタログに所属します。

- *catalog-name*

  MVS JCL の場合: JOBCAT、STEPCAT 文。

  VSE JCL の場合: DLBL 文の CATALOG オペランド。

- __LIB

  dataset-name がライブラリ名の場合のみ。

Type

データセットのファイルタイプ。次のいずれかの値です。

- FS: 順編成ファイル。デフォルト値です。
- VS: Sun MTP VSAM ファイル。
- GDG: 世代別データグループ。GDG 番号が 0 より大きい場合、このエントリは自動的に生成されます。

Mapped Name

トランスレータが生成したパス名。ローカルシステムのデータセット名でファイルまたはライブラリの位置を指定します。

- FS ファイルの場合は、ファイルのパス名です。
- 同じベース名 GDG の GDG データセットでは、File_Map 内にパス名が 1 つだけ定義されています。
- ライブラリの場合は、ライブラリのディレクトリのパス名です。
- VSAM ファイルの場合は、FCT に定義されているデータセット名です。

Record Properties File Name

Sun MBM ユーティリティーの SORT と IDCAMS、および Open PL/I アプリケーションプログラムで使用されるファイル属性を含む順編成ファイルの絶対パス名。

ファイルには位置パラメータが 1 つ以上のスペースで区切られて含まれ、データセット属性を定義します。パラメータは次のとおりです。

- *record-type*

  このパラメータは、Sun MBM の SORT および IDCAMS ユーティリティーからアクセスする FS ファイルで使用されます。有効なエントリは次のとおりです。

  - record

    レコード順。改行文字は含まれません。

- recordv

  可変長レコードファイル。各レコードに、レコードのデータ部分の長さを示す 4 バイトのヘッダー文字が含まれます。4 バイトのヘッダー部分の長さはデータ部分に含まれません。

- line

  行の順編成ファイル。各レコードは改行文字で区切られます。

- mfrcdv

  Micro Focus 可変長レコードファイル。レコードには、最大レコードサイズが 4,095 バイトを超えるかどうかによって 2 バイトまたは 4 バイトのレコードヘッダーが含まれます。ワード境界に揃えてレコード長を調整できます。

- *record-length*

  FS ファイルでのみ使用します。

  ジョブステップが SORT を実行しており、JCL が MVS JCL DD 文または VSE DLBL 文の RECSIZE オペランドの LRECL= オプションを使用して SORTIN や SORTOUT データセットのレコード長を指定していない場合、Sun MBM は、LRECL_SORTIN 環境変数および LRECL_SORTOUT 環境変数をこのパラメータの値に設定します。固定長ファイルの場合はレコード長、可変長ファイルの場合は最大レコード長です。

  レコード属性ファイルが File_Map で定義されていない場合、record はデフォルトの *record-type* になり、SORTOUT の *record-length* はデフォルトの 80 バイトに設定されます。

- *pli-file-type*

  Open PL/I アプリケーションプログラムでアクセスされるファイルの場合、有効な形式は次のとおりです。

  - VSAM: Sun MTP VSAM ファイル。
  - RECORD: レコード順編成ファイル。改行文字は含まれません。
  - STREAM: 行順。各レコードは改行文字で区切られます。

- *pli-record-size*

  Open PL/I アプリケーションプログラムでアクセスされるファイルにのみ適用されます。*pli_file_type* RECORD の場合、このエントリは必須です。数値または環境変数を入力します。

- *pli-record-format*

  Open PL/I アプリケーションプログラムでアクセスされるファイルにのみ適用されます。*pli_file_type* RECORD および STREAM の場合、このエントリは必須です。値またはドル記号 ($) で始まる環境変数です。有効な値は次のとおりです。

  - F: 固定長レコード
  - FB: 固定長ブロック化レコード
  - V: 可変長レコード
  - VB: 可変長ブロック化レコード

コード例 2-1 では、レコード属性ファイルの COBOL および Pl/I のエントリを示します。

コード例 2-1　　レコード属性ファイルのエントリ — 例

```
* COBOL
record 80
recordv 188

* PL/I
record 80 VSAM 80 F
line 133 STREAM 133 F
```

GDG Number

ベース名 GDG データセット用に括弧内に示される世代の最大番号 (128 まで)。

File_Map エントリの表示

▼ File_Map エントリを表示する

- **File_Map 画面で、次のいずれかの方法を使用します。**
  - データセットエントリをダブルクリックします。
  - エントリを選択して、「View Entry」ボタンをクリックします。

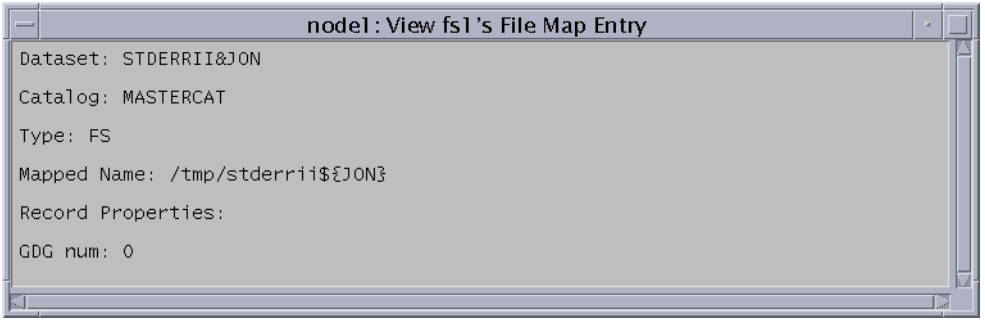


図 2-5　　File_Map エントリの表示

File_Map エントリのソート

ヘッダーにあるボタンの 1 つをクリックすると、昇順または降順で File_Map エントリをソートできます。

たとえば、データセット名でジョブをソートするには、「DataSet Name」ボタンをクリックします。これにより、データセット名が昇順に表示されます。もう一度「DataSet Name」ボタンをクリックすると、データセット名が降順に表示されます。

デフォルトのソート順序は「DataSet Name」フィールドの降順です。

File_Map の照会

▼ File_Map エントリのリストをフィルタする

1. 「File_Map」画面で「Set Query Options」ボタンをクリックします。

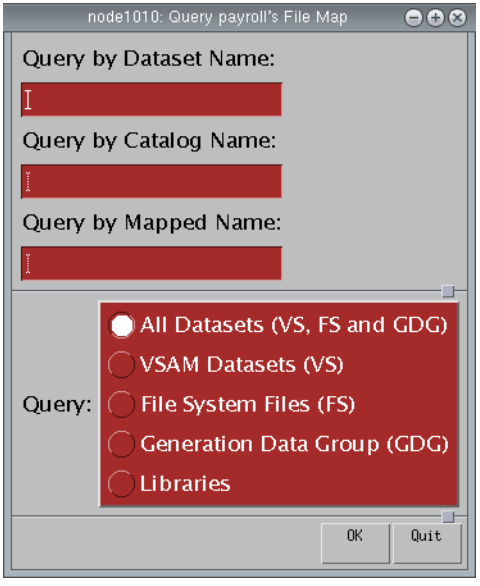


図 2-6　Query File_Map ダイアログボックス

2. 「Set Query Options」ダイアログボックスに条件を入力します。

3. 「OK」をクリックして、照会を実行します。

要求したエントリが「File_Map」画面に表示されます。

「Quit」をクリックするまで照会オプション画面はアクティブです。次に「File_Map」画面が閉じて自動的に再表示され、すべてのデータセットが表示されます。

「Query」ウィンドウで次の情報の一部または全部を入力できます。

Query by Dataset Name	データセット名。大文字・小文字を区別するグローバルオプションです。CCL など、データセット名の一部を指定すると、照会では名前に CCL を含むすべてのジョブを取り込みます。
Query by Catalog Name	指定したカタログ名を含むデータセットだけを取り込みます。
Query by Mapped Name	指定した割り当て名を含むデータセットだけを取り込みます。
Query	All Datasets: すべてのデータセットを取り込みます VSAM Datasets: VSAM データセットだけを取り込みます File System Files: ファイルシステムファイルだけを取り込みます Generation Data Group: GDG だけを取り込みます Libraries: ライブラリだけを取り込みます

次のボタンが画面に表示されます。

OK	照会を実行します。
Quit	この画面を閉じ、「File_Map」画面を再表示して、使用可能なデータセット、VSAM、ファイルシステム、および GDG ファイルの全リストを表示します。

Tool Kit を使った File_Map の更新

File_Map で変更するエントリが少ない場合、変更には Tool Kit を使用すると簡単です。一方、変更が多い場合は、cfm コマンドを使用します。詳細は、24 ページの「cfm コマンドを使った File_Map の更新」を参照してください。

▼ File_Map を更新する

1. 「File_Map」画面で、変更するエントリを選択し、「Update Entry」ボタンをクリックします。
2. 図 2-7 のダイアログボックスが表示されたら、フィールドを変更します。
   各フィールドについては、17 ページの「File_Map の内容」を参照してください。

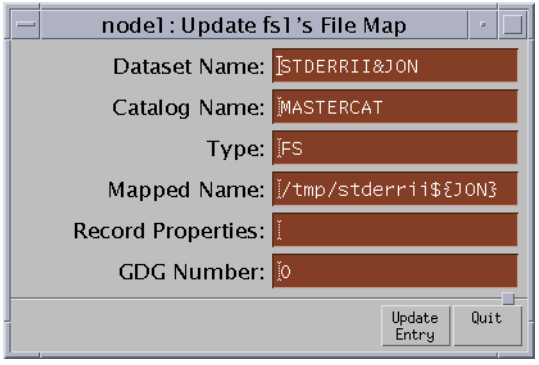


図 2-7　File_Map エントリの更新

3. 「Update Entry」をクリックします。
   更新された「File_Map」画面が表示されます。
4. 「Quit」をクリックして「Update」ダイアログボックスを閉じます。

cfm コマンドを使った File_Map の更新

この節では、cfm コマンドの使用法について説明します。この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
- 割り当てファイル名
- レコード属性
- GDG 番号

cfm コマンドに、変更する File_Map を入力します。次の中から適切なオプションを選択して、変更を指定します。

- File_Map からエントリを選択するには、小文字のオプション (-t、-c など) を指定します。
- 選択したエントリに対する変更を指定するには、大文字のオプション (-T、-C など) を指定します。
- 入力した File_Map でエントリを選択、検索してから、-p オプションを使って標準出力に表示します。-P オプションを使うと、変更したエントリのみが標準出力に表示されます。
- -e オプションを使って、選択したエントリを削除します。削除した項目は標準出力に表示されます。

▼ cfm コマンドの実行を準備する

1. **少なくとも 1 つのサブシステムを作成します。**
2. **サブシステムの環境を設定します。**

   cfm コマンドの完全な構文については、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

cfm コマンドの例

この節では、cfm コマンドのさまざまな使用法を示す例を説明します。

例: 出力データセット名を変更します。

%{Dn} はデータセット名のさまざまなフィールドを抽出します。ここで、*n* は 1 ～ 16 の値です。

```
(%{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。

次のコマンドがあるとします。

```
[node1] # cfm -i input-filemap -o output-filemap -d AA.BB.CC -D %{D1}.%{D2}
```

このコマンドは、次の入力 File_Map エントリを変更します。

```
AA.BB.CC  MASTERCAT  FS  /tmp/files/aa/bb/cc
```

変更の結果は次のとおりです。

```
AA.BB  MASTERCAT  FS  /tmp/files/aa/bb/cc
```

次のコマンドがあるとします。

```
[node1] # cfm -i input-filemap -o output-filemap -d AA.BB.CC -D %{D1}%{D2}
```

このコマンドは、次の入力 File_Map エントリを変更します。

```
AA.BB.CC  MASTERCAT  FS  /tmp/files/aa/bb/cc
```

フィールド D1 と D2 を連結し、エントリを変更した結果は次のとおりです。

```
AABB  MASTERCAT  FS  /tmp/files/aa/bb/cc
```

次のコマンドがあるとします。

```
[node1] # cfm -i input-filemap -o output-filemap -d AA.BB.DD -D XXXX.%{D1}
```

このコマンドは、次の入力 File_Map エントリを変更します。

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

ここで、*XXXX* は、データセット名に含める任意のユーザー指定文字列です。エントリ変更の結果は次のとおりです。

```
XXXX.AA  MASTERCAT  FS  /tmp/files/aa/bb/dd
```

例: 出力先ファイル名を変更します。

環境変数を使用している場合は、単一引用符で環境変数を囲みます。

次のコマンドがあるとします。

```
[node1] # cfm -i input-filemap -o output-filemap -f '$SEQFILES/prod/sort/cards'
-F '$PARMLIB/%{f5}'
```

このコマンドは、次の入力 File_Map エントリを変更します。

```
PROD.SORT.CARDS(CARD1)  MASTERCAT  FS  $SEQFILES/prod/sort/cards/card1
```

変更の結果は次のとおりです。

```
PROD.SORT.CARDS(CARD1)  MASTERCAT  FS  $PARMLIB/card1
```

次のコマンドがあるとします。

```
[node1] # cfm -i input-filemap -o output-filemap -f /prod/lib/idcams -F
/tmp/idcams/%{F4}
```

このコマンドは、次の入力 File_Map エントリを変更します。

```
PROD.LIB.IDCAMS(D118)  MASTERCAT  FS  /tmp/prod/lib/idcams/d118
```

変更の結果は次のとおりです。

```
PROD.LIB.IDCAMS(D118)  MASTERCAT  FS  /tmp/idcams/D118
```

次のコマンドがあるとします。

```
[node1] # cfm -i input-filemap -o output-filemap -f /tmp/files/a/dd1
-F '$SEQFILES/%{F*}'
```

このコマンドは、次の入力 File_Map エントリを変更します。

```
A.DD1  MASTERCAT  FS  /tmp/files/a/dd1
```

変更の結果は次のとおりです。

```
A.DD1  MASTERCAT  FS  $SEQFILES/TMP/FILES/A/DD1
```

次のコマンドがあるとします。

```
[node1] # cfm -i input-filemap -o output-filemap -d A.DD1 -F '$SEQFILES/%{d*}'
```

このコマンドは、次の入力 File_Map エントリを変更します。

```
A.DD1  MASTERCAT  FS  /tmp/files/a/dd1
```

変更の結果は次のとおりです。

```
A.DD1  MASTERCAT  FS  $SEQFILES/a/dd1
```

次のコマンドがあるとします。

```
[node1] # cfm -i input-filemap -o output-filemap -d PROD.PGMLIB
-F /tmp/files/%{d2}/%{D3}
```

このコマンドは、次の入力 File_Map エントリを変更します。

```
PGMLIB(&MBR)  MASTERCAT  FS  /tmp/files/prod/pgmlib/${MBR}
```

変更の結果は次のとおりです。

```
PGMLIB(&MBR)   MASTERCAT   FS   /tmp/files/pgmlib/${MBR}
```

例: 出力先ファイルを作成するための標準出力をリダイレクトします。

```
[node1] # cfm -i $FILEMAP -t GDG -p > fileout.gdg
```

注意 – 出力先ファイル名に `$FILEMAP` を使用しないでください。`File_Map` が破壊されます。

例: ファイルタイプ VS の入力 `File_Map` のすべてのエントリを選択し、標準出力に表示します。

次のコマンドがあるとします。

```
[node1] # cfm -i input-filemap -t VS -p
```

次の形式で、VSAM ファイルを標準出力に表示します。

```
JFV.SSSUREVED.BASACCTK.ACCOUNT.MASTER   MASTERCAT  VS  BASACCT
JFV.SSSUREVD.BASHELPK.ONLINE.HELP       MASTERCAT  VS  BASHELP
JFV.SSUREVD.BASMALIK.MAIL               MASTERCAT  VS  BASMAIL
```

例: `cfm` コマンドで変更されたエントリを表示します。

次のコマンドがあるとします。

```
[node1] # cfm -i input-filemap -t VS -T FS -P
```

`cfm` コマンドによって、`VS` から `FS` に変更されたエントリを表示します。

```
JFV.SSSUREVED.BASACCTK.ACCOUNT.MASTER   MASTERCAT  FS  BASACCT
JFV.SSSUREVD.BASHELPK.ONLINE.HELP       MASTERCAT  FS  BASHELP
JFV.SSUREVD.BASMALIK.MAIL               MASTERCAT  FS  BASMAIL
```

例: レコード属性を変更します。

レコード属性を変更するには、レコード記述子テーブルのフルパス名とディレクトリパス名を指定する必要があります。

```
[node1] # cfm -i input-filemap -o output-filemap -r record-descriptor-table -R directory-pathname
```

record-descriptor-table に次のエントリがあるとします。

```
BSTD10.KSDS            F  80
JF.SSSURESD.DELIVERY   V  132
```

次のファイルを作成します。

```
directory-pathname/F80.cfm
directory-pathname/V132.cfm
```

さらに、指定したデータセットの File_Map エントリにこのディレクトリ文字列を挿入します。

```
BSTD10.KSDS;MASTERCAT;FS;/tmp/bstd010/billtabl/ksds;directory-pathname/F80.cfm;0;
```

固定長レコードの場合、*directory_pathname*/F80.cfm ファイルには次の内容が含まれます。

```
Record          80
```

可変長レコードの場合、*directory_pathname*/V132.cfm ファイルには次の内容が含まれます。

```
Recordv         132
```

レコードタイプがユーザー定義文字列の場合、レコード属性のパス名で指定したファイルは次のように更新されます。

```
string record length
```

例: File_Map で定義されたディレクトリを生成するか、File_Map で定義されたすべてのディレクトリを生成します。

```
cfm -i inputFile_Map -K -x  | grep -v '#' | while read ff
do
       dirname=‘eval echo $ff‘
       echo creating $dirname
       mkdir -p $dirname
done
```

第3章

Tool Kit の使用

この章では、Sun MBM Tool Kit の使用法について説明します。この章の内容は、次のとおりです。

Tool Kit へのアクセス

1. **Sun MBM メインメニューで、「System Status」アイコンをクリックします。**

   図 3-1 に示された画面が表示されます。

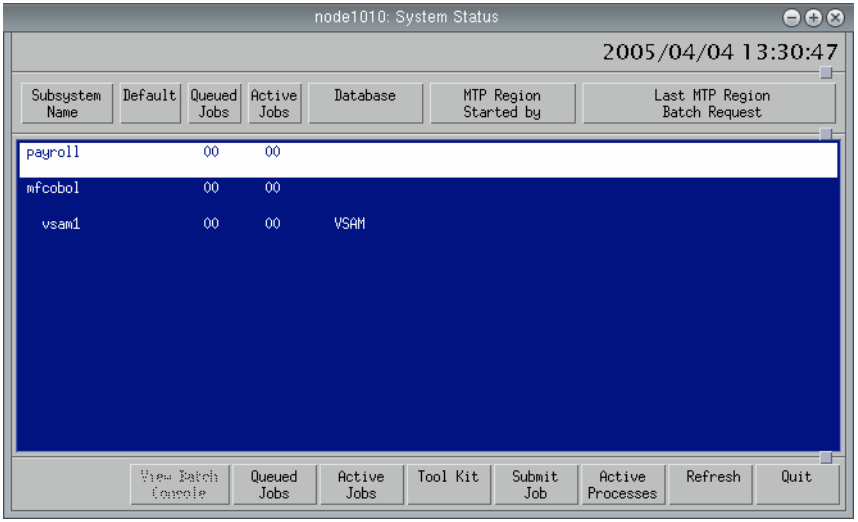


図 3-1　　System Status 画面

2. 「System Status」画面で、アクセスするサブシステムを選択して、「Tool Kit」ボタンをクリックします。

「Tool Kit」ウィンドウが表示されます (図 3-2)。

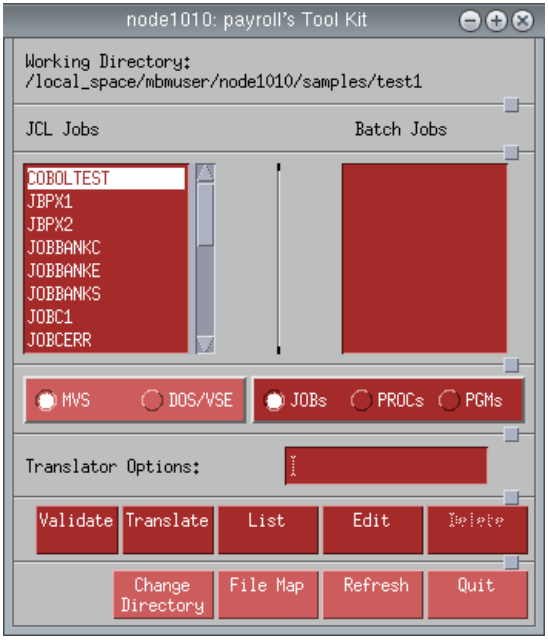


図 3-2　　Tool Kit ウィンドウ

「Tool Kit」ウィンドウでは、次のボタンが常にアクティブです。

ボタン	機能
Change Directory	さまざまな作業用ディレクトリを選択可能にします。
File Map	File_Map を表示し、更新を可能にします。
Refresh	最新の File_Map を表示します。
Quit	ウィンドウを閉じます。

Tool Kit を使用すると、mvstrans または dostrans トランスレータでジョブまたは手続きの検査や変換、COBOL プログラムのコンパイルを実行できます。

詳細は、41 ページの「Tool Kit に影響する環境変数」を参照してください。

JCL ジョブの検査または変換

▼ JCL ジョブを検査または変換する

1. 「Tool Kit」ウィンドウで、次を実行します。

   a. JCL タイプを選択します (この例では「MVS」)。

   b. ファイルタイプを選択します (この例では「JOBs」)。

   c. JCL ジョブを選択します。

   d. 必要に応じてトランスレータオプションを入力します。

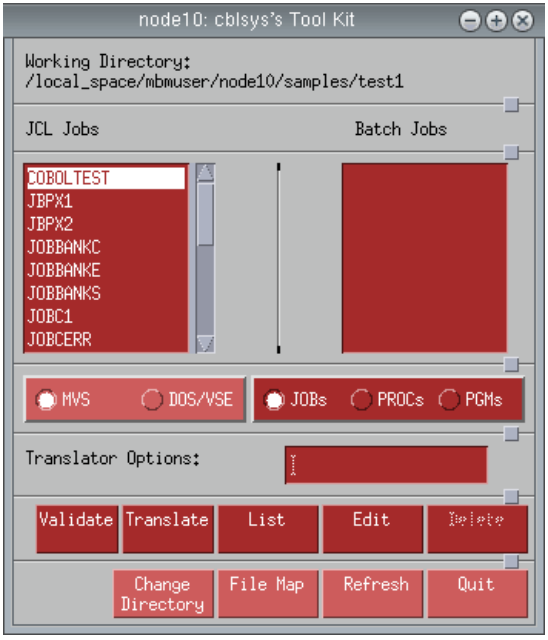


図 3-3　　JCL ジョブの検査または変換

2. 「Validate」または「Translate」ボタンをクリックします。
   - 「Validate」をクリックすると、「Validate Job」画面 (図 3-4) が表示されて、トランスレータが開始されたことが示され、警告またはエラーが表示されます。

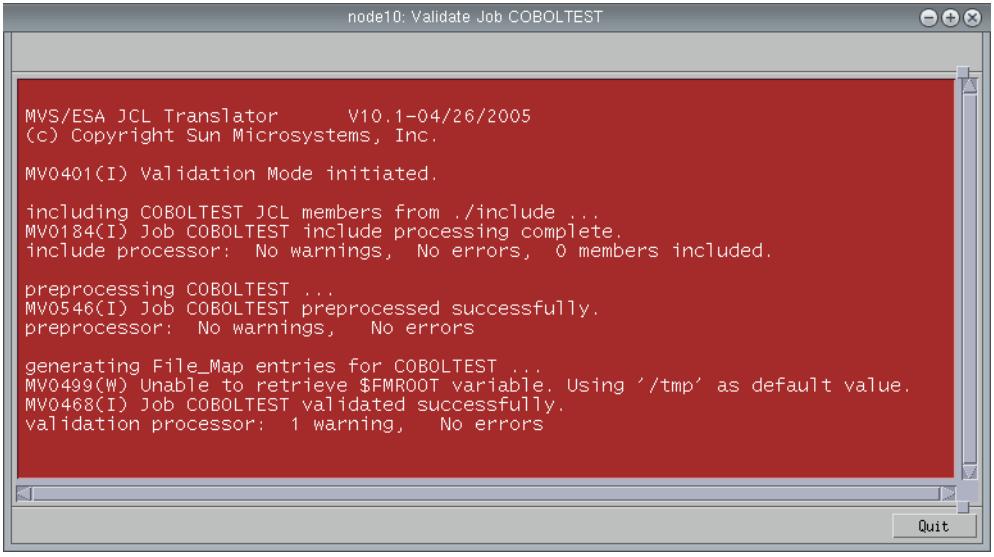


図 3-4　Validate Job 画面

- 「Translate」をクリックすると、「Translate Job」画面が表示されて、トランスレータが起動されたことが示され、警告またはエラーが表示されます。

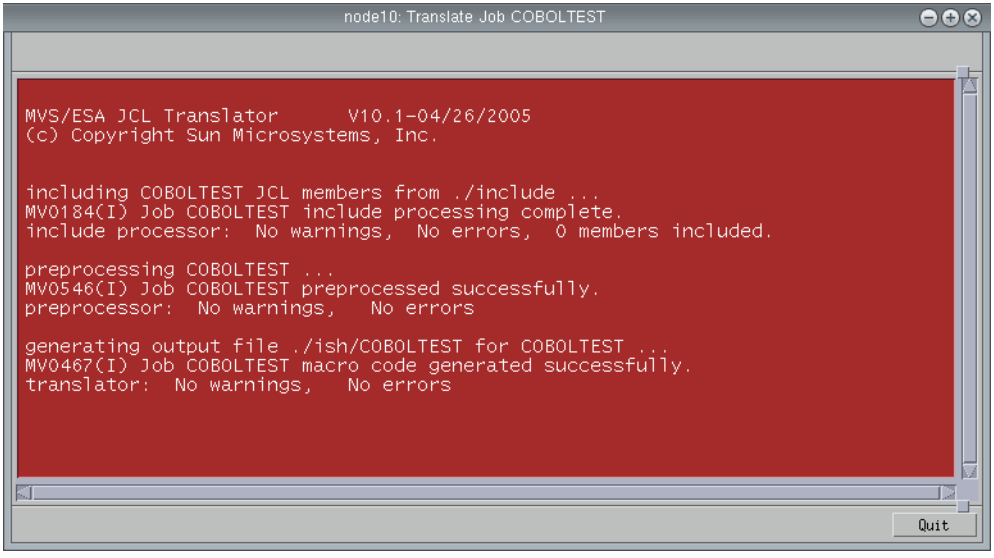


図 3-5　Translate Job 画面

3. **検査または変換が完了すると、「Took Kit」ウィンドウのジョブのリストにジョブが表示されます。**

   ここで、「List」、「Edit」、または「Delete」ボタンをクリックすると、ジョブを表示、編集、または削除できます。

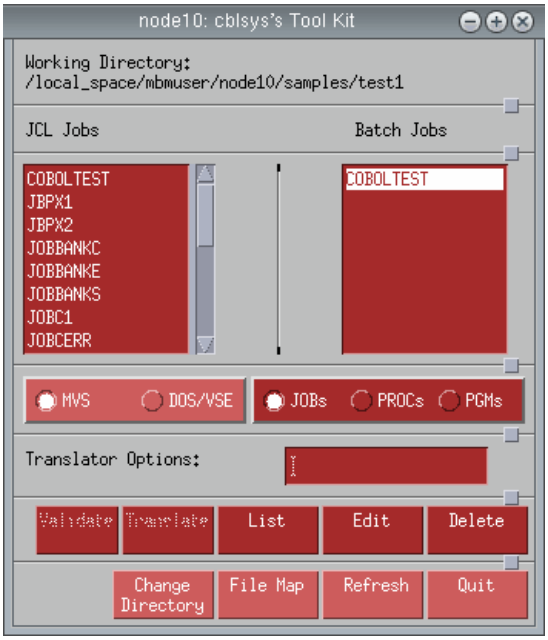


図 3-6　　Tool Kit ウィンドウ

4. **ジョブを一覧表示するには、ジョブを選択して「List」ボタンをクリックします。**

   詳細は、図 3-7 を参照してください。

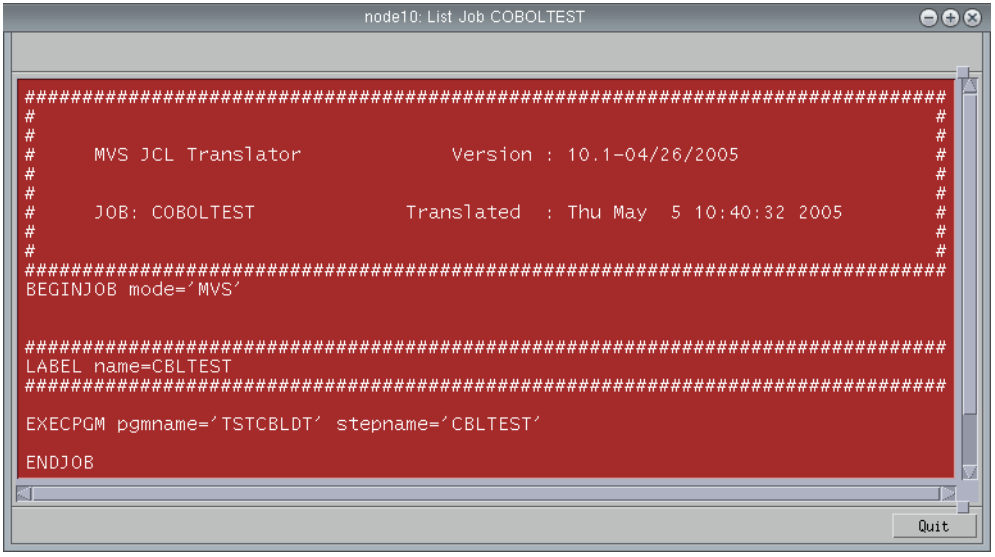


図 3-7　　List Job 画面

JCL 手続きの検査または変換

▼ JCL 手続きを検査または変換する

1. 「Tool Kit」ウィンドウで、次を実行します。
   a. トランスレータタイプを選択します (この例では「MVS」または「VSE」)。
   b. JCL タイプを選択します (この例では「PROCS」)。
   c. JCL 手続きを選択します。
   d. 必要に応じてトランスレータオプションを入力します。
2. 「Validate」または「Translate」ボタンをクリックします。

COBOL プログラムのコンパイル

Tool Kit では、対話型インタフェースを使用して COBOL プログラムをコンパイルします。Sun MBM 環境で COBOL アプリケーションプログラムを使用する方法については、231 ページの「COBOL アプリケーションプログラム」を参照してください。

▼ COBOL プログラムをコンパイルする

1. 「Tool Kit」ウィンドウで、次を実行します。

   a. ファイルタイプを選択します (この例では「PGMs 」)。

   b. プログラムを選択します。

   c. 必要に応じて、コンパイラオプションを変更します。

      詳細は、41 ページの「Tool Kit に影響する環境変数」を参照してください。

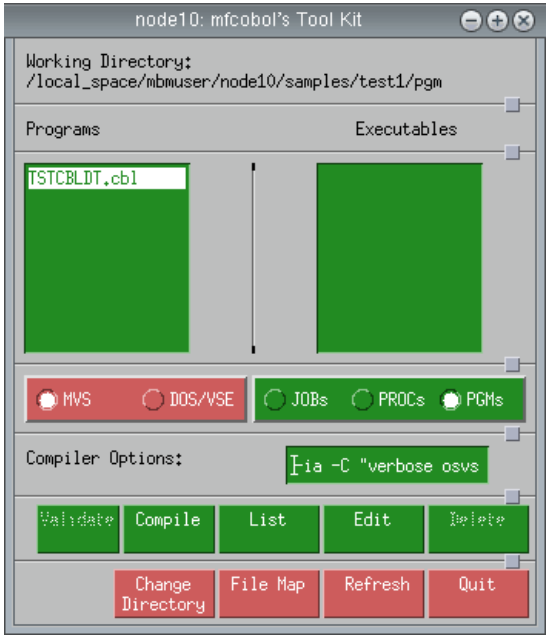


図 3-8　　COBOL プログラムのコンパイル

2. 「Compile」ボタンをクリックします。

   コンパイルの状態を示す「Compile Program」画面が表示されます。詳細は、図 3-9 を参照してください。

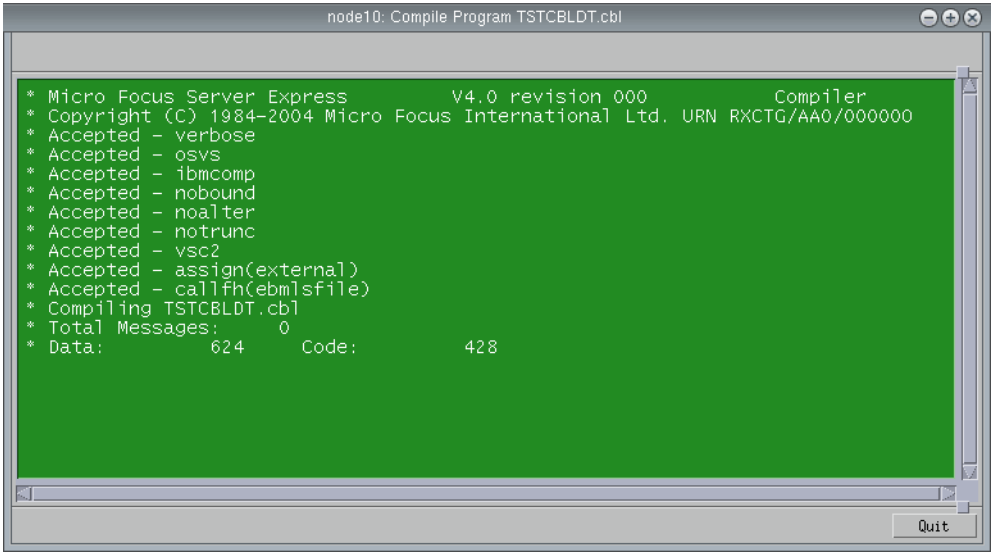


図 3-9　　Compile Program 画面

3. コンパイルが終了すると、「Executables」リストボックスに実行可能プログラムが表示されます。

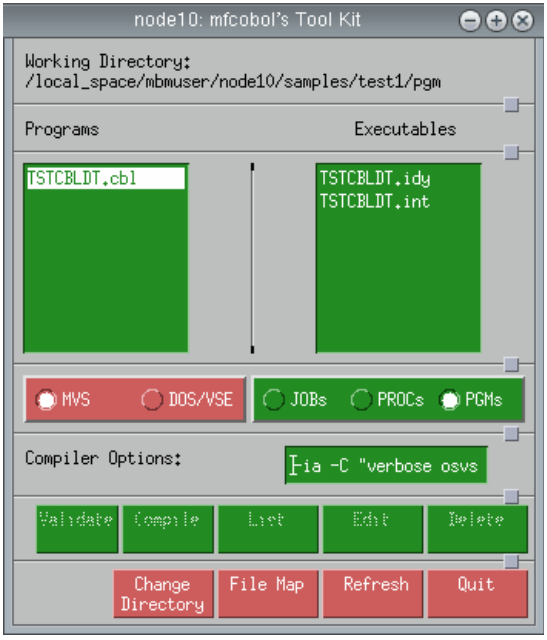


図 3-10　　Tool Kit ウィンドウ

4. COBOL プログラムを表示するには次の手順に従います。
   a. 「Tool Kit」ウィンドウの「Programs」領域で、プログラム名を選択します。
   b. 「List」ボタンをクリックし、図 3-11 の画面を表示します。

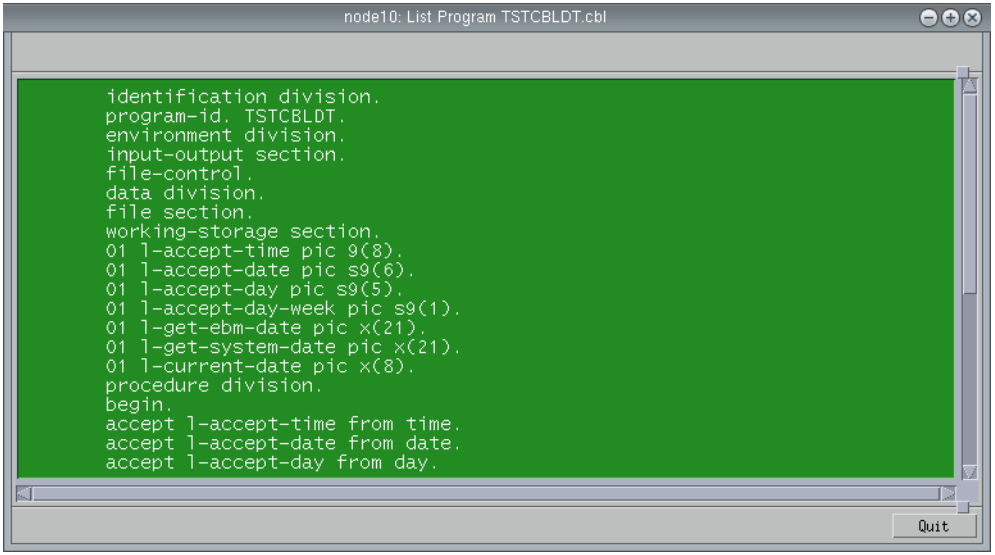


図 3-11　List Program 画面

Tool Kit に影響する環境変数

次の表の環境変数は、Tool Kit の実行に影響します。別のデフォルト値を使用する場合は、Sun MBM メインメニューを起動 (`ebmx`) する前に、`batchenv` ファイル内のこれらの環境変数をリセットする必要があります。`batchenv` 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。

表 3-1　Tool Kit 環境変数

環境変数	説明	デフォルト値
`MVSTRANS_EXEC`	MVS JCL トランスレータ	`$PACK/bin/mvstrans`
`MVSTRANS_OPT`	MVS JCL トランスレータオプション	ジョブ: `-f` 手続き: `-pf`
`DOSTRANS_EXEC`	VSE JCL トランスレータ	`$PACK/bin/dostrans`
`DOSTRANS_OPT`	VSE JCL トランスレータオプション	ジョブ: `-f` 手続き: `-pf`

`COBOL_EXEC`	COBOL コンパイラ	**Server Express** (デフォルト):
		`$COBDIR/bin/cob`
		ACUCOBOL-GT:
		`$ACUCOBOL/bin/ccbl`

表 3-1　Tool Kit 環境変数 (続き)

環境変数	説明	デフォルト値

COBOPT	COBOL コンパイラオプション	**Server Express** (デフォルト):
		`-ia -C "verbose osvs ibmcomp nobound noalter notrunc vsc2 assign=external callfh=ebmlsfile"`
		ACUCOBOL-GT:
		`-Cav "--fileAssign=external"`

PROCOB_EXEC	Oracle COBOL プリコンパイラ	`procob`
PROCOBOPT	Oracle COBOL プリコンパイラオプション	`include='pwd' include= $ORACLE_HOME/procobo/lib irclen=312 maxliteral=161 mode=ANSI hold_cursor=no release=yes maxopencursoe=50 dbms=v7`

第4章

MVS JCL の変換

mvstrans は、MVS JCL トランスレータです。この章では、mvstrans で File_Map にエントリを生成する方法と、mvstrans で MVS JCL ジョブストリームを Sun MBM マクロジョブに変換する方法を説明します。また、MVS JCL 文の変換例も示します。この章の内容は、次のとおりです。

サポートしているそれぞれの MVS JCL 文のリストは、『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。

MVS JCL の変換の準備

この節では、MVS JCL ジョブストリームを変換する前に実行しなければならないタスクを説明します。ノードをインストールし、サブシステムを作成してあることを前提とします。

- 44 ページの「ディレクトリ構造の作成」
- 46 ページの「FMROOT 環境変数の設定」
- 47 ページの「File_Map エントリの作成」。File_Map の詳細は、第 2 章を参照してください。

ディレクトリ構造の作成

mvstrans トランスレータでは、実行するためのディレクトリ構造が必要です。この構造は、最低でも次の表にリストされたサブディレクトリを収めた親ディレクトリから構成されます。

表 4-1　mvstrans ディレクトリ

ディレクトリ	内容
jmvs	JCL ジョブファイル
mvsp	JCL 手続きファイル
ish	変換されたジョブマクロファイル
ishp	変換された手続きマクロファイル

JCL に INCLUDE 文が含まれている場合、include ディレクトリを作成しなければならない場合もあります。

このディレクトリ構造での mvstrans の実行の詳細は、図 4-1 を参照してください。

▼ ディレクトリ構造を作成する

1. ジョブファイルの親ディレクトリを作成します。次に例を示します。

```
$ mkdir -p /user1/batch
```

2. 親ディレクトリに移動します。

```
$ cd /user1/batch
```

3. 入力 JCL ジョブ用に jmvs ディレクトリ、変換されたジョブマクロファイル用に ish ディレクトリを作成します。

```
$ mkdir jmvs ish
```

サブシステムを作成すると、ish ディレクトリを指定するように要求されます。

4. 変換する MVS JCL ジョブファイルのコピーを jmvs ディレクトリに配置します。

5. **入力 JCL 手続きファイル用に** `mvsp` **ディレクトリ、変換された手続きマクロファイル用に** `ishp` **ディレクトリを作成します。**

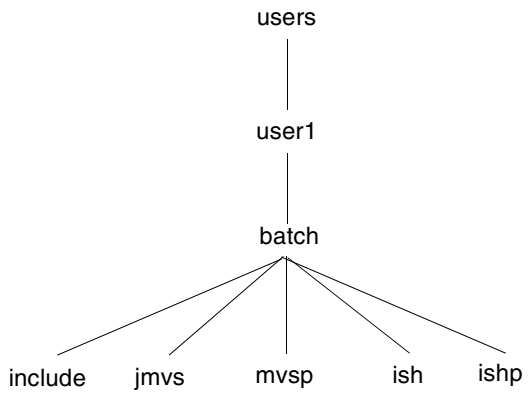
```
$ mkdir mvsp ishp
```

変換された手続きファイルが書き込まれるディレクトリは、サブシステムの作成時に定義されます。このディレクトリは多くの場合、`$PROCLIB` または `ishp` ディレクトリですが、他のディレクトリでもかまいません。

6. **MVS JCL 手続きファイルのコピーを** `mvsp` **ディレクトリに配置します。**

次の図に、`mvstrans` を実行するディレクトリを示します。

```
HOME=/users/user1
PROCLIB=$HOME/batch/ishp
MVSINCLDIR=$HOME/batch/include
```




```
mvstrans を実行するには：
. . $EBMHOME/batchenv subsys1
cd $HOME/batch
mvstrans ...
```

図 4-1 `mvstrans` のディレクトリ構造

変換規則の指定

サブシステムを作成するときには、mvstrans で使用する変換規則を JCL と JES2 の 2 つから指定できます。このオプションの指定方法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。値を指定しない場合は、JCL 変換上書きルールが使用されます。

1 つのジョブまたは一連のジョブに mvstrans を実行するときに -J オプションを使用して、サブシステムの変換規則を上書きできます。-J オプションの詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

注 – このリリースでは JES3 変換規則をサポートしていません。

FMROOT 環境変数の設定

FMROOT 環境変数には、移行環境で File_Map にエントリを作成してライブラリまたはデータセットを割り当てるときに mvstrans で使用するデフォルトの基本ディレクトリ名を記述します。サブシステムのユーザー設定ファイル ($USER_SETUP) に手動で $FMROOT を設定する必要があります。$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。

例

$FMROOT を設定しないと、mvstrans は次のようなエントリを File_Map に生成します。

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

$FMROOT を /user/dir1 に設定すると、mvstrans は次のようなエントリを File_Map に生成します。

```
A.B.C.D;MASTERCAT;FS;/user/dir1/a/b/c/d;;0;
```

実行時に評価される別の環境変数に $FMROOT を設定する場合は、$USER_SETUP ファイルに次の構文を記述します。

```
setenv FMROOT "\\\${ENVVAR}"
```

ここで *ENVVAR* は、$FMROOT に割り当てられた値です。

TRANSOPTS 環境変数の設定

TRANSOPTS 環境変数を使用して、トランスレータが起動されるたびに適用されるトランスレータオプションを設定できます。たとえば、変換ごとにバッチジョブからの日付スタンプとバージョン情報を省略させるには、-n トランスレータオプションを指定します。たとえば、Korn シェルの設定ファイルで TRANSOPTS を設定するには、次のように入力します。

```
TRANSOPTS='-n'; export TRANSOPTS
```

$USER_SETUP ファイルまたはサブシステムに関連したその他の設定ファイルの 1 つで $TRANSOPTS を設定すると、トランスレータオプションが自動的に使用されます。トランスレータを実行する前にコマンドプロンプトで $TRANSOPTS を設定すると、オプションはそのセッションのみで有効になります。

注意 – 設定するオプションを選択するときには注意してください。$TRANSOPTS で -v トランスレータオプションを設定しないでください。設定すると、変換のたびに File_Map エントリが再度作成されます。

File_Map エントリの作成

検査モード (-v オプション) で mvstrans を実行すると、メインフレームデータセット名とライブラリ名は、File_Map というファイルで移行環境のファイル名とディレクトリパス名に割り当てられます。サブシステムを作成すると、File_Map ファイルの位置を指定する FILEMAP 環境変数が設定されます。

検査モード (-v オプション) で mvstrans を実行すると、File_Map にエントリが作成されますが、出力マクロジョブスクリプトは作成されません。次の規則は、検査モードで mvstrans コマンドを使って MVS JCL ストリームから File_Map にエントリを作成する場合に適用されます。

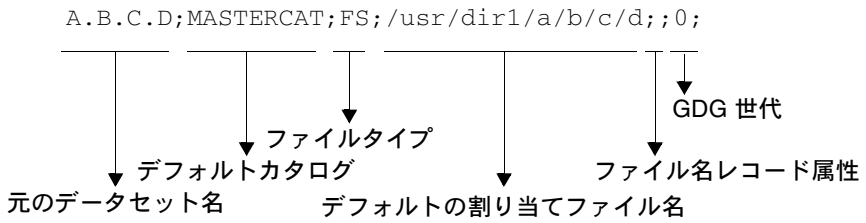
1. 認定されたデータセット。

   割り当てファイル名の前には $FMROOT の値が指定されます。次に例を示します。

   下記 JCL 文で $FMROOT を /usr/dir1 に設定した場合

```
//INFILE DD DSN=A.B.C.D
```

次の File_Map エントリが作成されます。



デフォルトのファイルタイプは FS (Sun MTP VSAM ファイル以外のすべてのファイル) です。データセットが VSAM の場合は、ファイルタイプを VS に変更し、ファイルのフルパス名を Sun MTP 領域のファイル制御テーブル (FCT) で定義したデータセットの VSAM ファイル名 (1 ～ 8 文字) に置換する必要があります。値がある場合、「Filename Record Attributes」フィールドの値は、レコード属性を収めたファイルの名前です。

File_Map を変更するには、BAM ユーティリティー (『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照)、Tool Kit (23 ページの「Tool Kit を使った File_Map の更新」を参照)、または cfm コマンド (24 ページの「cfm コマンドを使った File_Map の更新」を参照) を使用します。次に例を示します。

```
A.B.C.D;MASTERCAT;VS;KSDS1;;0;
```

2. 記号パラメータ参照を含むデータセットは、&A.. 形式で表します。

エントリは $FMROOT 値とデータセット名の変数を基に作成されます。次に例を示します。

下記 JCL 文で $FMROOT の値を /usr/dir1 に設定した場合

```
//INFILE DD DSN=&A..&B..&C..D
```

次の File_Map エントリが作成されます。

```
&A.&B.&C.D;MASTERCAT;FS;/usr/dir1/${A}/${B}/${C}/d;;0;
```

記号パラメータ &A、&B、および &C は、変数 ${A}、${B}、および ${C} にそれぞれ変換されます。

注 – &B.C と定義したデータセットには「.」が含まれていますが、これはデータセット修飾子ではなく、記号変数 &B の一部として処理されます。File_Map に追加するとき、mvstrans ではこのデータセットを /${B}/c ではなく /${B}c と解釈します。

---

3. D&E および D&E.C 形式で指定される変数を含むデータセット。

   生成されたエントリは $FMROOT 値とデータセット名の変数を基に作成されます。次に例を示します。

   下記 JCL 文で $FMROOT を /usr/dir1 に設定した場合

```
//INFILE DD DSN=&D&E.B.C.D&E
```

   次の File_Map エントリが作成されます。

```
&D&E.B.C.D&E;MASTERCAT;FS;/usr/dir1/${D}${E}b/c/d${E};;0;
```

注 – &E.B と定義したデータセットには「.」が含まれていますが、これはデータセット修飾子ではなく、記号変数 &E の一部として処理されます。File_Map に追加するとき、mvstrans ではこのデータセットを /${E}/b ではなく /${E}b と解釈します。

---

4. ライブラリエントリ。

   File_Map のライブラリエントリは、__LIB という文字で File_Map エントリ内の 2 番目のフィールドに指定されます。このエントリは、JOBLIB DD、STEPLIB DD、JOBCAT DD、または STEPCAT DD JCL 文で指定したライブラリを元に作成されます。次に例を示します。

   下記 JCL 文で $FMROOT の値を /usr/dir1 に設定した場合

```
//JOBLIB DD DSN=SUBLIB1
```

   次の File_Map エントリが作成されます。

```
SUBLIB1;__LIB;FS;/usr/dir1/sublib1;;0;
```

mvstrans トランスレータの使用

トランスレータへの入力は、MVS JCL ジョブまたは手続きです。出力は、実行可能ジョブマクロファイルです。このマクロファイルは、バッチサブシステムに対してサブミットされ、実行されます。ジョブマクロファイルを使って、変換された手続きを呼び出します。Sun MBM マクロの詳細は、『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。

mvstrans を使った MVS JCL の変換は、次の 2 段階の処理で行われます。

- mvstrans を検査モード (-v オプション) で実行して、File_Map を作成します。VSAM データセットまたは世代データグループ (GDG) を使用する場合は、File_Map を更新する必要があります。詳細は、第 2 章を参照してください。
- mvstrans を実行して、JCL を変換し、ジョブマクロファイルを作成します。

図 4-1 に、mvstrans を実行するディレクトリを示します。

▼ MVS JCL ジョブと手続きを変換する

1. **サブシステムの環境を設定する方法は、次のとおりです。**
   - ソース batchenv でサブシステム名を指定します。次に例を示します。

```
$ . $EBMHOME/batchenv cblsys
```

   - 次の手順に従います。

   a. Sun MBM メインメニューを表示します。

   b. メインメニューの「Command Prompt」アイコンをクリックし、サブシステム名を入力します。

2. **まだサブシステムのユーザー設定ファイルに $FMROOT を設定していない場合は設定します。**
3. **希望する場合、$TRANSOPTS を設定します。**
4. **元の MVS JCL ジョブと手続きを含む jmvs と mvsp の親ディレクトリに移動します。**

5. **すべてのジョブと手続きファイルに対して検査モード (ジョブの場合は `-v` オプション、手続きの場合は `-v -p` オプション) で `mvstrans` を実行し、サブシステム `File_Map` を作成します。**

   `mvstrans '*'` オプションを使うと、`jmvs` または `mvsp` ディレクトリ内のすべてのファイルを変換できます。`mvstrans` コマンドの構文については、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

6. **`File_Map` を確認し、必要な変更を行います。**

7. **すべてのジョブと手続きファイルに対して `mvstrans -f` を実行し、ジョブおよび手続きスクリプトを作成します。**

   データセットとライブラリに関する情報は、`mvstrans` を使って `File_Map` から取り出します。手続きファイルを変換する場合は `-p` オプションを使用する必要があります。

   元の JCL または作成したバッチジョブを使って、ジョブを管理できます。

   - 元の JCL を維持する場合、変更を行ったらもう一度変換する必要があります。
   - 作成したマクロスクリプトを維持する場合は、トランスレータを再度使用する必要はありません。

   `mvstrans` とジョブのサブミット方法の詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。Sun MBM マクロの詳細は、『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。

MVS JCL 文のサポート

MVS JCL 文の多くはメインフレーム固有であり、Sun MBM 環境には適用されません。これらの文はトランスレータによって変換されず、変換プロセス時に警告が発せられます。そのメッセージについては、『Sun Mainframe Batch Manager ソフトウェア メッセージガイド』を参照してください。

サポートされる文はすべて条件文で、52 ページの「`mvstrans` の制限事項」に説明する制限事項に基づいています。JCL と JES のサポートの詳細は、『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。

mvstrans の制限事項

mvstrans トランスレータにはいくつかの制限事項があるので、JCL ストリームを変更して特定の問題を取り除き、変換処理を継続できるようにする必要があります。

DD 文

逆方向参照はサポートされていません。次に例を示します。

```
//DD1 DD DSN=*.stepname.ddname
```

COBOL プログラムでは、連結データセットは順編成ファイルシステム (FS) ファイル (Sun MTP VSAM ファイル以外のファイル) に対してのみサポートされます。各データセットのレコードサイズは同じである必要があります。

EXEC 文

EXEC PGM パラメータでの逆方向参照はサポートされません。変換出力では、'*.' 接頭辞は省略されます。

例:

```
//STEP1 EXEC PGM=*.stepname.ddname
```

上記の文は次のように変換されます。

```
EXECPGM pgmname='stepname.ddname' stepname='STEP1'
```

JOB 文

1 つの変換につき、1 つの JOB 文だけがサポートされます。JOB 文で特定される追加のジョブストリームを、別の JCL ファイルに抽出します。

JCL 文のパラメータフィールド値では、国別文字 (national characters)、# と @、英文字、数字、または英数字文字列がサポートされます。ただし、国別文字 $、その他の文字 !、*、空白、>、<、文字シーケンス &&、および '' を mvstrans で生成された出力に表示する必要がある場合、これらの文字は ASCII 16 進数表示に変換されます。

例:

```
PARM=$$
```

上記の文は次のように変換されます。

```
parm='/x24/x24'
```

実行時に、16 進数値は、呼び出された COBOL アプリケーションプログラム内で $$ に変換されます。

parm 文字列内の印刷不可能な文字はすべて /x*HH* 形式に変換されます。ここで、*HH* は、文字の 16 進値です。

特殊文字の変換方法の詳細は、『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。

GDG と連結データセット

GDG と連結データセットは、COBOL プログラムだけでサポートされます。

手続き

インストリーム手続きはサポートされません。JCL ストリームからインストリーム手続きを抽出して、別のファイルを作成します。この方法で、手続きがカタログ化されます。

ソート

SORT および ICEMAN の場合、SYSIN で渡された JCL のソート文を手動で変更し、Sun MBM ソートユーティリティー (sortx) 規則 (または使われるソートユーティリティー) に従う必要があります。出力レコードサイズを指定すると、sortx では入力レコードも同じサイズとみなします。

空白

カンマの前の空白文字は JCL 行の継続を示します。ただし、空白のあとに他の文字があるとコメントとみなされます。

記号パラメータ

DD 文で DISP パラメータ値の指定に使われる記号パラメータは、mvstrans ではサポートされません。変換出力では、disp= パラメータが o に設定されます。これはデフォルトの値です。

```
//DDIN1 DD DSN=A.B.C,DISP=(&SYM1,&SYM2)
```

MVS JCL 変換の例

この節では、mvstrans で JCL 文を Sun MBM マクロに変換する例を示します。

コメント (//*) 文

//* コメント文を使用して、ジョブまたは手続きに情報テキストを含めます。通常の MVS JCL コメント (列 1 ～ 3 の //*) は、次の例に示したように Sun MBM シェルコメント行に変換されます。

コード例 4-1　　コメント文

```
...
//*
//*INITIATE MONTHLY REPORT GENERATION
//*
...
```

上記の文は次のように変換されます。

```
...
#  *
#  *INITIATE MONTHLY REPORT GENERATION
#  *
...
```

MVS JCL コメントには特殊な形式があり、指定された *system-command* を実行する EBMSYSCMD マクロ文を生成します。EBMSYSCMD マクロの詳細は、『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。

//*EBMSYSCMD *system-command*

生成されたマクロコマンドは RETURN-CODE と CONDITION-CODE 処理ロジックにより実行が制御されます。次に例を示します。

次の文があります。

```
//*EBMSYSCMD echo "Prepare output bin ABCD for job PREOP1"
```

上記の文は次のように変換されます。

```
EBMSYSCMD << !
echo "Prepare output bin ABCD for job PREOP1"
!
```

区切り記号 (/*) 文

区切り記号 (/*) 文は、インストリームデータの終了を定義します。DD * と DD DATA でサポートされます。

コード例 4-2　　/* 文を使用したインストリームデータの終了の区切り

```
...
//STEP01 EXEC PROC=PROC001
//PSTEP1.INV0001 DD *
AA BB
CC DD
/*
...
```

上記の文は次のように変換されます。

```
...
ASSGNDD ddname='PSTEP1.INV0001' type='INSTREAM' << !
AA BB
CC DD
!
```

ジョブの終了 (//) 文

ジョブの終了 // 文 (または空白文) でジョブストリームの終了を定義します。コメントは使用できません。列 3 ～ 80 は空白にする必要があります。

コード例 4-3　　ジョブの終了 (//) 文で終了するジョブ

```
...
//STEP99 EXEC PGM=PGMAA
//
```

上記の文は次のように変換されます。

```
...
EXECPGM pgmname='PGMAA' stepname='STEP99'
ENDJOB
```

DD 文

DD 文を使用して、ジョブステップ内でアクセスされるデータセットを定義します。データセットに必要なリソースも特定します。

* (アスタリスク) 定位置パラメータ

* (アスタリスク) パラメータは、インストリームデータの開始を定義します。mvstrans は、ASSGNDD マクロを生成して、指定した ddname をインストリームデータセットのマクロ呼び出しの直後の内容に割り当てます。これによって、指定したステップで実行されるアプリケーションプログラムは、ファイルとしてデータにアクセスできます。たとえば、OPEN、READ などです。mvstrans -k オプションは、インストリームデータセットレコードの切り上げを制御します。デフォルトの値は 80 で、インストリームデータセットの 80 列目よりあとのすべての文字が破棄され、変換出力には表示されないことを示しています。

コード例 4-4　　DD * パラメータ

```
...
//STEP01 EXEC PGM=PROGA
//FILEA DD *
00001A Y100B
00002A N000C
/*
...
```

上記の文は次のように変換されます。

```
ASSGNDD ddname='FILEA' type='INSTREAM' << !
00001A Y100B
00002A N000C
!
```

DATA 定位置パラメータ

DATA パラメータは、インストリームデータセットの開始を定義します。mvstrans は、ASSGNDD マクロを生成して、指定した ddname をインストリームデータセットのマクロ呼び出しの直後の内容に割り当てます。mvstrans -k オプションは、インストリームデータセットレコードの切り上げを制御します。デフォルトの値は 80 で、インストリームデータセットの 80 列目よりあとのすべての文字が破棄され、変換出力には表示されないことを示しています。

コード例 4-5 DD DATA パラメータ

```
...
//INV0001 DD DATA
//FILEA DD DSN=TEST.FILEA
//FILEB DD DSN=TEST.FILEB
/*
```

上記の文は次のように変換されます。

```
ASSGNDD ddname='INV0001' type='INSTREAM' << !
//FILEA DD DSN=TEST.FILEA
//FILEB DD DSN=TEST.FILEB
!
```

注 – インストリームデータに JCL 文が含まれる場合、アプリケーションでの使用法に応じて、これらの行を手動で変換する必要があります。

DUMMY 定位置パラメータ

DUMMY 定位置パラメータは、データセットの定義を制限します。mvstrans は、type='DUMMY' の ASSGNDD マクロを生成します。ファイルで読み取りが実行される場合、ファイルの終了条件が設定されます。書き込みが実行される場合、レコードは書き込まれません。

コード例 4-6　　DD DUMMY パラメータ

```
...
//DDINF1 DD DUMMY,DCB=BLKSIZE=6000
...
```

上記の文は次のように変換されます。

```
...
ASSGNDD ddname='DDINF1' type='DUMMY'
...
```

BURST キーワードパラメータ

sysout データセットは、DD 文と SYSOUT パラメータで定義されます。BURST キーワードは、印刷された出力をバーストするかどうかを指定します。DD 文で印刷オプションが指定されている場合、mvstrans は、sysout データセットに SETPRINT マクロを生成します。DD 文では、ASSGNDD マクロの前に、SETPRINT マクロが生成されます。

コード例 4-7　　DD SYSOUT 文の BURST パラメータ

```
...
//PRNT1 DD SYSOUT=*,BURST=Y,COPIES=3,DEST=RMT1
...
```

上記の文は次のように変換されます。

```
...
SETPRINT ddname='PRNT1' burst='Y' copies='3' dest='RMT1'
ASSGNDD ddname='PRNT1' type='SYSOUT' class='JOBCLASS'
...
```

CHARS キーワードパラメータ

sysout データセットは、DD 文と SYSOUT パラメータで定義されます。CHARS キーワードは、使用する文字テーブルを定義します。DD 文で印刷オプションが指定されている場合、mvstrans は、sysout データセットに SETPRINT マクロを生成します。DD 文では、ASSGNDD の前に、SETPRINT マクロが生成されます。

コード例 4-8　　DD SYSOUT 文の CHARS パラメータ

```
...
//PRTFILE1 DD SYSOUT=A,CHARS=CHR1
//PRTFILE2 DD SYSOUT=B,CHARS=(CHR1,CHR2)
...
```

上記の文は次のように変換されます。

```
...
SETPRINT ddname='PRTFILE1' chars='CHR1'
ASSGNDD ddname='PRTFILE1' type='SYSOUT' class='A'
SETPRINT ddname='PRTFILE2' chars='[CHR1, CHR2]'
ASSGNDD ddname='PRTFILE2' type='SYSOUT' class='B'
...
```

印刷オプション値が引用符または括弧で囲まれている場合、mvstrans は引用符または括弧を角括弧に変換します。

COPIES キーワードパラメータ

sysout データセットは、DD 文と SYSOUT パラメータで定義されます。COPIES キーワードは、印刷するコピー部数を指定します。DD 文で印刷オプションが指定されている場合、mvstrans は、sysout データセットに SETPRINT マクロを生成します。DD 文では、ASSGNDD マクロの前に、SETPRINT マクロが生成されます。

コード例 4-9　　DD SYSOUT 文の COPIES パラメータ

```
...
//PRTFIL1 DD SYSOUT=A,COPIES=2
//PRTFIL2 DD SYSOUT=A,COPIES=(1,(2,3))
...
```

上記の文は次のように変換されます。

```
...
SETPRINT ddname='PRTFIL1' copies='2'
ASSGNDD ddname='PRTFIL1' type='SYSOUT' class='A'
SETPRINT ddname='PRTFIL2' copies='[1,[2,3]]'
ASSGNDD ddname='PRTFIL2' type='SYSOUT' class='A'
...
```

印刷オプション値が引用符または括弧で囲まれている場合、mvstrans は引用符または括弧を角括弧に変換します。

SETPRINT マクロで設定された印刷オプションは、Sun MBM スプールユーティリティーで使用されます。Sun 以外のスプールパッケージでも印刷オプション値を取得できます。

DCB キーワードパラメータ

DCB パラメータは、データセット関連の詳細を定義して、アプリケーションプログラムのデータ制御ブロック化 (DCB) マクロが要求する情報を完成します。

コード例 4-10　LRECL および RECFM サブパラメータを使用する DD DCB パラメータ

```
...
//INFILE1 DD DSN=TEST.ORDFILE,DISP=SHR,
//DCB=(RECFM=V,LRECL=100)
...
```

上記の文は次のように変換されます。

```
...
ASSGNDD ddname='INFILE1' recsize='100' recfmt='V' filename='/tmp/test/ordfile'
        dataset='TEST.ORDFILE'
...
```

DEST キーワードパラメータ

sysout データセットは、DD 文と SYSOUT パラメータで定義されます。DEST キーワードは、宛先を指定します。DD 文で印刷オプションが指定されている場合、mvstrans は、sysout データセットに SETPRINT マクロを生成します。DD 文では、ASSGNDD マクロの前に、SETPRINT マクロが生成されます。

コード例 4-11　DD SYSOUT 文の DEST パラメータ

```
...
//PRTFIL1 DD SYSOUT=A,DEST=RMT1
//PRTFIL2 DD SYSOUT=A,DEST=(NODE1,USER1)
...
```

上記の文は次のように変換されます。

```
...
SETPRINT ddname='PRTFIL1' dest='RMT1'
ASSGNDD ddname='PRTFIL1' type='SYSOUT' class='A'
SETPRINT ddname='PRTFIL2' dest='[NODE1,USER1]'
ASSGNDD ddname='PRTFIL2' type='SYSOUT' class='A'
...
```

印刷オプション値が引用符または括弧で囲まれている場合、mvstrans は引用符または括弧を角括弧に変換します。

SETPRINT マクロで設定された印刷オプションは、Sun MBM スプールユーティリティーで使用されます。Sun 以外のスプールパッケージでも印刷オプション値を取得できます。

DISP キーワードパラメータ

mvstrans は DISP パラメータをサポートします。DISP パラメータの各サブパラメータに対して、ASSGNDD マクロに別個のパラメータが生成されます。DISP サブパラメータの値がデフォルトである場合、-F オプションを指定してデフォルトのパラメータ値が生成されない限り、mvstrans は ASSGNDD マクロに対応するパラメータを生成しません。

次の表に、DISP *dataset-status* サブパラメータに対して mvstrans が実行する変換について説明し、DISP *dataset-status* 値を反映して ASSGNDD マクロに disp パラメータが生成されることを示します。

表 4-2 DISP *dataset-status* サブパラメータに対する mvstrans の変換

DISP dataset-status の値	mvstrans が生成する ASSGNDD パラメータ
NEW	disp='o'
OLD	disp='i-o'
SHR	disp='i-o'
MOD	disp='a'
default	disp パラメータのデフォルト値は、定義されるデータセットのファイルタイプによって異なります。 順編成ファイル (File_Map ファイルタイプ FS) = disp='o' VSAM データセット (File_Map ファイルタイプ VS) = disp='i-o' disp パラメータは、mvstrans で -F を使用している場合にのみ生成されます。

次の表に、DISP *disp-normal-termination* サブパラメータに対して mvstrans が実行する変換について説明し、DISP *disp-normal-termination* 値を反映して ASSGNDD マクロに normal パラメータが生成されることを示します。

表 4-3 DISP *disp-normal-termination* サブパラメータに対する mvstrans の変換

DISP disp-normal-termination の値	mvstrans が生成する ASSGNDD パラメータ
DELETE	normal='d'
KEEP	normal='k'

表 4-3　DISP *disp-normal-termination* サブパラメータに対する mvstrans の変換 (続き)

DISP disp-normal-termination の値	mvstrans が生成する ASSGNDD パラメータ
PASS	normal='p'。デフォルトでは、削除に続く処置 (KEEP、CATLG など) が定義されていない場合、Sun MBM は、渡されたデータセットを保持します。この動作を IBM メインフレームのデフォルトの動作 (通常の処置指令を受け取って上書きする後続の手順がない場合、渡されたデータセットを削除) に変更する場合は、サブシステムの $USER_SETUP ファイルに次の環境変数を設定する必要があります。 setenv EBM_DISP_PASS MAINFRAME この環境変数に対する設定内容は、FS タイプのファイルにのみ適用されます。VSAM ファイルには適用されません。
CATLG	normal='k'
UNCATLG	normal='k'
default	normal パラメータのデフォルト値は、どのファイルタイプがデータセットで定義されているか、およびデータセットに NEW の *dataset-status* があるかどうかによって異なります。 順編成ファイル (FS の File_Map ファイルタイプ) = normal='d' VSAM データセット (VS の File_Map ファイルタイプ) = normal='k' *dataset-status* == NEW = normal='d' *dataset-status* != NEW = normal='k' normal パラメータは、mvstrans で -F を使用している場合にのみ生成されます。

表 4-4 に、DISP *disp-abnormal-termination* サブパラメータに対して mvstrans が実行する変換について説明し、DISP *disp-abnormal-termination* 値を反映して ASSGNDD マクロに abend パラメータが生成されることを示します。

表 4-4　DISP *disp-abnormal-termination* サブパラメータに対する mvstrans の変換

DISP disp-abnormal-termination の値	mvstrans が生成する ASSGNDD パラメータ
DELETE	abend='d'
KEEP	abend='k'
CATLG	abend='k'

表 4-4　DISP *disp-abnormal-termination* サブパラメータに対する mvstrans の変換 (続き)

DISP disp-abnormal-termination の値	mvstrans が生成する ASSGNDD パラメータ
UNCATLG	abend='k'
default	abend パラメータのデフォルト値は、どのファイルタイプがデータセットで定義されているか、およびデータセットに NEW の *dataset-status* があるかどうかによって異なります。 順編成ファイル (FS の File_Map ファイルタイプ) = abend='d' VSAM データセット (VS の File_Map ファイルタイプ) = abend='k' *dataset-status* == NEW = abend='d' *dataset-status* != NEW = abend='k' abend パラメータは、mvstrans で -F を使用している場合にのみ生成されます。

コード例 4-12　DISP パラメータを指定する DD 文

```
...
//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 は ASSGNDD *filename* パラメータの生成を省略し、次のように文を変換します。

```
...
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'
...
```

DLM キーワードパラメータ

DLM パラメータは、インストリームデータセットを終了する区切り文字を定義します。mvstrans では、DLM パラメータをサポートし、JES3 と同様に動作します。つまり、DLM 値だけが、インストリームデータセットの終了を示します。DD 文のあとのレコードの 1 列目と 2 列目に DLM 値がない場合、入力の終了でインストリームデータセットを区切ります。

コード例 4-13　　DLM パラメータを使用するインストリームデータセットの定義

```
...
//SYSIN DD *,DLM=XX
ABC
DEF
GHI
XX
...
```

上記の文は次のように変換されます。

```
...
ASSGNDD ddname='SYSIN' type='INSTREAM' << !
ABC
DEF
GHI
!
...
```

DSNAME キーワードパラメータ

DSNAME パラメータは、データセットの名前を指定します。mvstrans は、記号パラメータ参照のある値を含め、DSN 値のすべての形式をサポートしています。検査モード (-v オプション) で実行するときに、mvstrans は DSN 値を使用して File_Map エントリを作成します。アポストロフィ、アスタリスク (*)、空白文字、ドル記号の国別文字 ($) などの多くの文字は、DSN 値が検出されると下線に置換されます。

次の JCL スニペットに、固定的で未修飾のデータセット名を示します。

コード例 4-14　　固定的で未修飾の *dataset-name*

```
...
//DUNQUAL DD DSN=MY$DATA,DISP=(SHR)
...
```

$FMROOT を /sys1/test に設定した場合、JCL 文は次のように変換されます。

```
...
ASSGNDD ddname='DUNQUAL' dataset='MY_DATA' \\\
       filename='/sys1/test/mydata' disp='i-o'
...
```

次の JCL スニペットに、固定的で修飾されたデータセット名を示します。

コード例 4-15　固定的で修飾された *dataset-name*

```
...
//DQUAL DD DSN=PROJA.USER1.MYDATA,DISP=(NEW,CATLG,DEL)
...
```

$FMROOT を /sys1/test に設定した場合、JCL 文は次のように変換されます。

```
...
ASSGNDD ddname='DQUAL' \\\
       dataset='PROJA.USER1.MYDATA' \\\
       filename='/sys1/test/proja/user1/mydata' \\\
       disp='o' normal='k' abend='d'
...
```

次の JCL スニペットに、固定的で修飾されたデータセット名を示します。このデータセットには、記号パラメータ参照が含まれています。

コード例 4-16　固定的で修飾された *dataset-name* (記号パラメータ参照を含む)

```
...
//DQUALSYM DD DSN=&A..B&C.D.&E&F,DISP=SHR
...
```

$FMROOT を /sys1 に設定すると、mvstrans は、DQUALSYM 用に次のエントリを File_Map に生成します。

```
&A.B&C.D.&E&F;MASTERCAT;FS;/sys1/${A}/b${C}d/${E}${F};;0;
```

ただし、mvstrans では & 文字のデータセット名を作成できません。記号パラメータの参照は、ジョブの実行時に、解決される環境変数の参照に置換されます。

mvstrans は、DD 文を次のように変換します。

```
...
ASSGNDD ddname='DQUALSYM' \\\
       dataset='\${A}.B\${C}D.\${E}\${F}' \\\
       filename='/sys1/\${A}/b\${C}d/\${E}\${F}' \\\
       disp='i-o'
...
```

次の JCL スニペットでは、DSN 値が PDS メンバーの DD 文を示します。

コード例 4-17　　DSN 値が PDS メンバーの場合

```
...
//DPDSMEM DD DSN=APPL1.USR1(FLA),DISP=SHR
...
```

$FMROOT を /sys1/test に設定した場合、JCL 文は次のように変換されます。

```
...
ASSGNDD ddname='DPDSMEM' dataset='APPL1.USR1' member='FLA' \\\
       filename='/sys1/test/appl1/usr1/fla' \\\
       disp='i-o' member='FLA'
...
```

この例では、次のようになります。

- DD2.CAF.DATA で指定されたデータセットは、File_Map では次のエントリのある GDG として定義されます。

コード例 4-18　　ベース名 GDG および GDG 世代の特定のオカレンス

```
DD2.CAF.DATA;MASTERCAT;FS;/sys1/test/dd2/caf/data;;9;
```

- USER1.DATA で指定されたデータセットは、File_Map では次のエントリのある GDG として定義されます。

```
USER1.DATA;MASTERCAT;FS;/sys1/test/user/data;;9;
```

次の JCL があるとします。

```
...
//DGDGEN1 DD DSN=USER1.DATA(+1),DISP=(NEW,CATLG,DEL)
//DGDG DD DSN=DD2.CAF.DATA,DISP=(NEW,CATLG,DELETE)
...
```

上記の文は次のように変換されます。

```
...
ASSGNDD ddname='DGDGEN1' dataset='USER1.DATA' \\\
       gdg='+1' filename='/sys1/test/user/data' \\\
       disp='o' normal='k' abend='d'
ASSGNDD ddname='DGDG' dataset='DD2.CAF.DATA' \\\
       gdg='ALL' filename='/sys1/test/dd2/caf/data' \\\
       disp='o' normal='k' abend='d'
...
```

次の JCL スニペットでは、ジョブ名は JOB001 です。

コード例 4-19　　明示的にコーディングされた一時データセット

```
...
//TEMP0001 DD DSN=&&TEMPFILE,DISP=(NEW,PASS,DELETE)
...
```

JCL は次のように変換されます。

```
ASSGNDD ddname='TEMP0001' dataset='JOB001_TEMPFILE' \\\
       type='TEMP' disp='o' normal='k' abend='d'
```

次の JCL スニペットでは、DSN が定義されていません。

コード例 4-20　　暗黙の一時データセット

```
...
//TEMP0002 DD UNIT=SYSSQ,DISP=(NEW,PASS,DELETE)
...
```

JCL は次のように変換されます。

```
ASSGNDD ddname='TEMP0002' type='TEMP' \\\
       disp='o' normal='k' abend='d'
```

次の JCL スニペットは連結データセットを定義します。

コード例 4-21　　連結データセットの定義

```
...
//DCONCAT1 DD DSN=ABE.ONE,DISP=(SHR)
//         DD DSN=ABE.TWO,DISP=SHR
//         DD DSN=ABE.THREE,DISP=SHR
...
```

`$FMROOT` を `/sys1/test` に設定した場合、JCL 文は次のように変換されます。

```
ASSGNDD ddname='DCONCAT1' dataset='ABE.ONE' \\\
       filename='/sys1/test/abe/one' disp='i-o'
ASSGNDD dataset='ABE.TWO' \\\
       filename='/sys1/test/abe/two' disp='i-o'
ASSGNDD dataset='ABE.THREE' \\\
       filename='/sys1/test/abe/three' disp='i-o'
```

FCB キーワードパラメータ

sysout データセットは、DD 文と SYSOUT パラメータで定義されます。FCB キーワードは、用紙制御イメージを定義します。FCB キーワードパラメータなどの印刷オプションが DD 文で指定されている場合、mvstrans は、sysout データセットに SETPRINT マクロを生成します。DD 文では、ASSGNDD マクロの前に、SETPRINT マクロが生成されます。

コード例 4-22　　DD SYSOUT 文の FCB パラメータ

```
...
//PRTREP1 DD SYSOUT=A,FCB=IM01
//PRTREP2 DD SYSOUT=A,FCB=(IN22,VERIFY)
...
```

上記の文は次のように変換されます。

```
...
SETPRINT ddname='PRTREP1' fcb='IM01'
ASSGNDD ddname='PRTREP1' type='SYSOUT' class='A'
SETPRINT ddname='PRTREP2' fcb='[IN22,VERIFY]'
ASSGNDD ddname='PRTREP2' type='SYSOUT' class='A'
...
```

印刷オプション値が引用符または括弧で囲まれている場合、mvstrans は引用符または括弧を角括弧に変換します。

SETPRINT マクロで設定された印刷オプションは、Sun MBM スプールユーティリティーで使用されます。Sun 以外のスプールパッケージでも印刷オプション値を取得できます。

FLASH キーワードパラメータ

sysout データセットは、DD 文と SYSOUT パラメータで定義されます。FLASH キーワードは、sysout データセットの印刷中に使用するフォームオーバーレイを指定します。FLASH キーワードパラメータなどの印刷オプションが DD 文で指定されている場合、mvstrans は、sysout データセットに SETPRINT マクロを生成します。DD 文では、ASSGNDD マクロの前に、SETPRINT マクロが生成されます。

コード例 4-23 DD SYSOUT 文の FLASH パラメータ

```
...
//FRMPRNT DD SYSOUT=*,FLASH=NONE
//REP01 DD SYSOUT=A,FLASH=(OV01,'1')
...
```

上記の文は次のように変換されます。

```
...
SETPRINT ddname='FRMPRNT' flash='NONE'
ASSGNDD ddname='FRMPRNT' type='SYSOUT' class='JOBCLASS'
SETPRINT ddname='REP01' flash='[OV01,[1]]'
ASSGNDD ddname='REP01' type='SYSOUT' class='A'
...
```

印刷オプション値が引用符または括弧で囲まれている場合、mvstrans は引用符または括弧を角括弧に変換します。

SETPRINT マクロで設定された印刷オプションは、Sun MBM スプールユーティリティーで使用されます。Sun 以外のスプールパッケージでも印刷オプション値を取得できます。

HOLD キーワードパラメータ

sysout データセットは、DD 文と SYSOUT パラメータで定義されます。HOLD キーワードは、そのあとのコマンドが解放するまで、印刷のために sysout データセットを保持するようにシステムに指示します。HOLD キーワードパラメータなどの印刷オプションが DD 文で指定されている場合、mvstrans は、sysout データセットに SETPRINT マクロを生成します。DD 文では、ASSGNDD マクロの前に、SETPRINT マクロが生成されます。

コード例 4-24　　DD SYSOUT 文の HOLD パラメータ

```
...
//FRMPRNT DD SYSOUT=*,HOLD=YES
...
```

上記の文は次のように変換されます。

```
...
SETPRINT ddname='FRMPRNT' hold='YES'
ASSGNDD ddname='FRMPRNT' type='SYSOUT' class='JOBCLASS'
...
```

SETPRINT マクロで設定された印刷オプションは、Sun MBM スプールユーティリティーで使用されます。Sun 以外のスプールパッケージでも印刷オプション値を取得できます。

LRECL キーワードパラメータ

sysout データセットは、DD 文と SYSOUT パラメータで定義されます。LRECL キーワードは、データセットのレコード長を定義します。DD 文で印刷オプションが指定されている場合、mvstrans は、sysout データセットに SETPRINT マクロを生成します。DD 文では、ASSGNDD マクロの前に、SETPRINT マクロが生成されます。

コード例 4-25　　LRECL パラメータを指定した DD 文

```
...
//INDF0001 DD DSNAME=USERA.INDF,LRECL=156,DISP=(NEW,KEEP)
...
```

$FMROOT を /sys1/test に設定した場合、JCL 文は次のように変換されます。

```
ASSGNDD ddname='INDF0001' dataset='USERA.INDF' \\\
       filename='/test/usera/indf' recsize='156' \\\
       disp='o' normal='k'
```

MODIFY キーワードパラメータ

sysout データセットは、DD 文と SYSOUT パラメータで定義されます。MODIFY キーワードは、コピー修正モジュールを特定します。DD 文で印刷オプションが指定されている場合、mvstrans は、sysout データセットに SETPRINT マクロを生成します。DD 文では、ASSGNDD マクロの前に、SETPRINT マクロが生成されます。

コード例 4-26 DD SYSOUT 文の MODIFY パラメータ

```
...
//FRMPRNT1 DD SYSOUT=*,MODIFY=AFT1
//REP02 DD SYSOUT=A,MODIFY=(A,'0')
...
```

上記の文は次のように変換されます。

```
...
SETPRINT ddname='FRMPRNT1' modify='AFT1'
ASSGNDD ddname='FRMPRNT1' type='SYSOUT' class='JOBCLASS'
SETPRINT ddname='REP02' modify='[A,[0]]'
ASSGNDD ddname='REP02' type='SYSOUT' class='A'
...
```

印刷オプション値が引用符または括弧で囲まれている場合、mvstrans は引用符または括弧を角括弧に変換します。

印刷出力の処理の詳細は、132 ページの「印刷出力の処理」を参照してください。

OUTLIM キーワードパラメータ

sysout データセットは、DD 文と SYSOUT パラメータで定義されます。OUTLIM キーワードは、現在の DD 文に定義された sysout データセットに書き込みできる論理レコード数を制限するようにシステムに指示します。印刷オプションが DD 文で指定されている場合、mvstrans は、sysout データセットに SETPRINT マクロを生成します。DD 文では、ASSGNDD マクロの前に、SETPRINT マクロが生成されます。

コード例 4-27 DD SYSOUT 文の OUTLIM パラメータ

```
...
//FRMPRNT1 DD SYSOUT=B,OUTLIM=100000
...
```

上記の文は次のように変換されます。

```
...
SETPRINT ddname='FRMPRNT1' outlim='100000'
ASSGNDD ddname='FRMPRNT1' type='SYSOUT' class='B'
...
```

SETPRINT マクロで設定された印刷オプションは、Sun MBM スプールユーティリティーで使用されます。Sun 以外のスプールパッケージでも印刷オプション値を取得できます。

OUTPUT キーワードパラメータ

Sun MBM は、印刷出力の処理の一部として、OUTPUT パラメータをサポートします。印刷出力の処理の詳細は、132 ページの「印刷出力の処理」を参照してください。

コード例 4-28　OUTPUT パラメータを指定した DD 文 DDOUT1 および DDOUT2

```
//STEP1 EXEC PGM=APROG
//OUT1 OUTPUT COPIES=3,DEST=RMT2,OUTLIM=600000
...
//STEP2 EXEC PGM=BPROG
//OUT2 OUTPUT FCB=FFE1,HOLD=Y,BURST=Y,COPIES=5
//DDOUT1 DD SYSOUT=A,OUTPUT=*.OUT2
//DDOUT2 DD SYSOUT=A,OUTPUT=(*.STEP1.OUT1,*.OUT2
...
```

上記の文は次のように変換されます。

```
SETPRINT printid='OUT1' scope='JOB' \\\
       copies='3' dest='RMT2' outlim='600000'
...
SETPRINT printid='OUT2' scope='STEP' \\\
       fcb='FFE1' hold='y' burst='y' copies='5'
...
ASSGNDD ddname='DDOUT1' type='SYSOUT' class='A' \\\
       printid='OUT2'
ASSGNDD ddname='DDOUT2' type='SYSOUT' \\\
       class='JOBCLASS' printid='STEP1.OUT1,OUT2'
...
```

RECFM キーワードパラメータ

RECFM 値が、固定長レコードまたは可変長レコードを示す場合、mvstrans は、DD 文に生成された ASSGNDD マクロに recfmt パラメータを追加します。

表 4-5　ASSGNDD パラメータの RECFM 値

RECFM 値	解釈されるレコードタイプ	mvstrans が生成する ASSGNDD パラメータ
'A'	Fixed (固定)	recfmt='F'
'D'	Variable (可変)	recfmt='V'
'F'	Fixed (固定)	recfmt='F'
'FB'	Fixed (固定)	recfmt='F'

表 4-5　ASSGNDD パラメータの RECFM 値 (続き)

RECFM 値	解釈されるレコードタイプ	mvstrans が生成する ASSGNDD パラメータ
'FBS'	Fixed (固定)	recfmt='F'
'FS'	Fixed (固定)	recfmt='F'
'FT'	Fixed (固定)	recfmt='F'
'M'	Fixed (固定)	recfmt='F'
'U'	Fixed (固定)	recfmt='F'
'V'	Variable (可変)	recfmt='V'
'VB'	Variable (可変)	recfmt='V'
'VBS'	Variable (可変)	recfmt='V'
'VS'	Variable (可変)	recfmt='V'

コード例 4-29　可変長データセットのレコード形式を指定する JCL

```
...
//INDF0001 DD DSNAME=USERA.INDF,
//     DISP=(NEW,KEEP),RECFM=VB
...
```

上記の文は次のように変換されます。

```
ASSGNDD ddname='INDF0001' dataset='USERA.INDF' \\\
       filename='/sys1/tst/usera/indf' \\\
       disp='o' normal='k' recfmt='V'
```

コード例 4-30　固定長データセットのレコード形式を指定する JCL

```
...
//INDF0002 DD DSNAME=USERB.INDF,
// DISP=(NEW,KEEP),RECFM=F,LRECL=82
...
```

上記の文は次のように変換されます。

```
ASSGNDD ddname='INDF0002' dataset='USERB.INDF' \\\
       filename='/tmp/userb/indf' disp='o' normal='k' \\\
       recfmt='f' recsize='82'
...
```

RLS キーワードパラメータ

RLS パラメータは、共有の必要があるレコードを収めた VSAM データセットのレコード共有または共有プロトコルのレベルを指定します。

この機能を有効にするには、BAM で対応する MVS メインフレーム互換オプションを使用してサブシステムを構成しなければいけません。Sun MTP 領域と通信するサブシステムの構成が必要です。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。

コード例 4-31 RLS パラメータ

```
...
//DDIN1 DD DSN=VSAM1,DISP=SHR,RLS=CR
...
```

上記の文は次のように変換されます。

```
ASSGNDD ddname=VSAM1 disp=i/o rls=CR
```

VSAM RC (read consistency) の詳細は、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。

SYSOUT キーワードパラメータ

SYSOUT パラメータは、データセットをシステム出力データセット (sysout データセット) として特定します。mvstrans は、type=SYSOUT を使用して ASSGNDD マクロを生成します。印刷オプションが DD 文にパラメータとして存在するか、SYSOUT パラメータ内のサブパラメータとして存在する場合、SETPRINT マクロも生成され、ASSGNDD の前に出現します。Sun MBM サブシステムを作成すると、ジョブの実行時に、定義したディレクトリの下に sysout ファイルが作成されます。Sun MBM スプーラまたは別のスプールメカニズムを使用して、これらのファイルを印刷できます。

「//DD1 DD SYSOUT=*」のように、アスタリスクパラメータを指定すると、生成された sysout はジョブ履歴ファイルに書き込まれ、この DD の物理 sysout ファイルが sysout ディレクトリから削除されます。

次の JCL スニペットでは、出力クラス、書き込み側モジュール、形式名を指定する DD SYSOUT 文を示します。

コード例 4-32 出力クラス、書き込み側モジュール、および形式名を指定する DD SYSOUT 文

```
...
//PRNTFILE DD SYSOUT=(A,WRIT1,FMAA)
...
```

次のように変換されます。

```
...
SETPRINT ddname='PRNTFILE' writer='WRIT1' forms='FMAA'
ASSGNDD ddname='PRNTFILE' type='SYSOUT' class='A'
...
```

次の JCL スニペットでは、書き込み側モジュール INTRDR、JES2 /*OUTPUT 文の印刷オプションを指定し、出力クラスを指定しない DD SYSOUT 文を示します。

コード例 4-33 書き込み側モジュールおよび印刷オプションを指定する DD SYSOUT 文

```
...
/*OUTPUT CD1 DEST=RMT1,COPIES=2
...
//PRNTFILE DD SYSOUT=(,INTRDR,CD1)
...
```

次のように変換されます。

```
SETPRINT printid='CD1' scope='JOB' overridedd='y' \\\
       dest='RMT1' copies='2'
...
SETPRINT ddname='PRNTFILE' writer='INTRDR'
ASSGNDD ddname='PRNTFILE' type='SYSOUT' \\\
       class='OUTCLASS' printid='CD1'
...
```

UCS キーワードパラメータ

sysout データセットは、DD 文と SYSOUT パラメータで定義されます。UCS キーワードは、汎用文字セットを指定します。DD 文で印刷オプションが指定されている場合、mvstrans は、sysout データセットに SETPRINT マクロを生成します。DD 文では、ASSGNDD マクロの前に、SETPRINT マクロが生成されます。

コード例 4-34 DD SYSOUT 文の UCS パラメータ

```
...
//FRMPRNT1 DD SYSOUT=*,UCS=CS1
//REP02 DD SYSOUT=A,UCS=(YN,,VERIFY)
...
```

上記の文は次のように変換されます。

```
...
SETPRINT ddname='FRMPRNT1' ucs='CS1'
ASSGNDD ddname='FRMPRNT1' type='SYSOUT' class='JOBCLASS'
SETPRINT ddname='REP02' ucs='[YN,,VERIFY]'
ASSGNDD ddname='REP02' type='SYSOUT' class='A'
...
```

印刷オプション値が引用符または括弧で囲まれている場合、mvstrans は引用符または括弧を角括弧に変換します。

SETPRINT マクロで設定された印刷オプションは、Sun MBM スプールユーティリティーで使用されます。Sun 以外のスプールパッケージでも印刷オプション値を取得できます。

特別な DD 名

次の節で説明するように、Sun MBM は、JOBCAT、JOBLIB、STEPCAT、STEPLIB、および SYSIN をサポートします。それ以外はサポートされず、トランスレータによって警告が表示されます。

JOBCAT

JOBCAT は、マスタカタログやプライベートカタログを検索する前に、データセットを検索するカタログを定義します。mvstrans は、JOBCAT 文をサポートしています。mvstrans を検査モード (-v オプション) で実行すると、File_Map に *catalog-name* のライブラリエントリが存在しない場合は作成されます。

File_Map エントリの形式は、次のとおりです。

```
catalog-name;__LIB;FS;mapped-dir;;0;
```

ここで *mapped-dir* は、移行環境での *catalog-name* の割り当てディレクトリです

LIBDEF マクロは、mvstrans 出力での JOBCAT 文を表します。LIBDEF マクロには、JOB の scope パラメータ値があり、ジョブの期間での利用度を示します。

コード例 4-35 JOBCAT DD 文

```
//NOR9999 JOB ASHWORTH
//JOBCAT DD DSN=USRCAT0,DISP=SHR
...
```

mvstrans を検査モード (-v オプション) で実行すると、$FMROOT が /sys1 に設定されている場合、このエントリが File_Map に生成されます。

```
USRCAT0;__LIB;FS;/sys1/usrcat0;;0;
```

変換モードで実行された mvstrans は、次のマクロジョブを作成します。

```
...
BEGINJOB mode='MVS'
LIBDEF scope='JOB' type='CAT' \\\
       dataset='USERCAT0' lib='/sys1/usercat0'
...
```

STEPCAT

STEPCAT は、ジョブカタログを検索する前に、データセットを検索するカタログを定義します。mvstrans は、STEPCAT 文をサポートしています。検査モード (-v オプション) で実行すると、File_Map に *catalog-name* のライブラリエントリが存在しない場合は作成されます。

File_Map エントリの形式は、次のとおりです。

```
catalog-name;__LIB;FS;mapped-dir;;0;
```

ここで *mapped-dir* は、移行環境での *catalog-name* の割り当てディレクトリです

LIBDEF マクロは、mvstrans 出力での STEPCAT 文を表します。LIBDEF マクロには、STEP の scope パラメータ値があり、指定されたジョブステップの期間での利用度を示します。

コード例 4-36　STEPCAT DD 文

```
//STEP1X4 EXEC PGM=TPROG
//STEPCAT DD DSN=VSMCAT3,DISP=SHR
...
```

mvstrans を検査モード (-v オプション) で実行すると、$FMROOT が /sys1 に設定されている場合、次のエントリが File_Map に生成されます。

```
VSMCAT3;__LIB;FS;/sys1/vsmcat3;;0;
```

変換モードで実行された mvstrans は、次のマクロジョブを作成します。

```
...
LIBDEF scope='STEP' type='CAT' \\\
       dataset='VSMCAT3' lib='/sys1/vsmcat3'
...
```

JOBLIB

JOBLIB は、ジョブの EXEC 文で指定されたプログラムを検索するライブラリを定義します。mvstrans は、JOBLIB 文をサポートしています。 mvstrans を検査モード (-v オプション) で実行すると、File_Map に *job-library* のライブラリエントリが存在しない場合は作成されます。

File_Map エントリの形式は、次のとおりです。

job-library;__LIB;FS;*mapped-dir*;;0;

ここで *mapped-dir* は、移行環境での *job-library* の割り当てディレクトリです。

LIBDEF マクロは、mvstrans 出力での JOBLIB 文を表します。LIBDEF マクロには、JOB の scope パラメータ値があり、ジョブの期間での利用度を示します。

コード例 4-37　JOBLIB DD 文

```
...
//JOBX JOB ASHWORTH
//JOBLIB DD DSN=PRIVATE.LIB,DISP=SHR
...
```

mvstrans を検査モード (-v オプション) で実行すると、$FMROOT が /sys1 に設定されている場合、次のエントリが File_Map に生成されます。

```
PRIVATE.LIB;__LIB;FS;/sys1/private/lib;;0;
```

変換モードで実行された mvstrans は、抽出によって次のようにマクロジョブを作成します。

```
...
BEGINJOB mode='MVS'
LIBDEF scope='JOB' type='PGM' \\\
       dataset='PRIVATE.LIB' lib='/sys1/private/lib'
...
```

STEPLIB

STEPLIB は、現在のジョブステップの EXEC 文で指定されたプログラムを検索するためのライブラリを定義します。mvstrans を検査モード (-v オプション) で実行すると、File_Map に *step-library* のライブラリエントリが存在しない場合は作成されます。

File_Map エントリの形式

step-library;__LIB;FS;*mapped-dir*;;0;

ここで *mapped-dir* は、移行環境での *step-library* の割り当てディレクトリです。

LIBDEF マクロは、mvstrans 出力での STEPLIB 文を表します。LIBDEF マクロには、STEP の scope パラメータ値があり、ステップの期間での利用度を示します。

コード例 4-38　STEPLIB DD 文

```
...
//STEP4OF6 EXEC PGM=PABPGM
//STEPLIB DD DSN=PSTEP4.LIB,DISP=SHR
...
```

mvstrans を検査モード (-v オプション) で実行すると、$FMROOT が /sys1 に設定されている場合、次のエントリが File_Map に生成されます。

```
PSTEP4.LIB;__LIB;FS;/sys1/pstep4/lib;;0;
```

変換モードで実行された mvstrans は、次のマクロジョブ文を作成します。

```
...
LIBDEF scope='STEP' type='PGM' \\\
       dataset='PSTEP4.LIB' lib='/sys1/pstep4/lib'
...
```

SYSIN

SYSIN は、インストリームデータの開始を定義します。mvstrans は、SYSIN DD 文の明示的形式と暗黙的形式をサポートしています。1 つのジョブステップに複数の DD SYSIN データセットが見つかった場合、mvstrans は、生成された出力に最初のデータセットだけを定義します。mvstrans -k オプションは、インストリームデータセットレコードの切り上げを制御します。デフォルトの値は 80 で、インストリームデータセットの 80 列目よりあとのすべての文字が破棄され、変換出力には表示されないことを示しています。

コード例 4-39　ジョブステップで明示的にコーディングされた SYSIN DD 文

```
...
//SYSIN DD *
This is sysin data record 1
This is sysin data record 2
This is sysin data record 3
/*
...
```

上記の文は次のように変換されます。

```
...
ASSGNDD ddname='SYSIN' type='INSTREAM' << !
This is sysin data record 1
This is sysin data record 2
This is sysin data record 3
!
...
```

コード例 4-40　データセットレコードに暗黙的な区切りと JCL コメントのある SYSIN DD 文

```
...
//SYSIN DD *
This is sysin data record 1
//* A JCL comment within the instream dataset but not a record
This is sysin data record 2
//DD1 DD DSN=AAA.BB,DISP=SHR Implied end of SYSIN
...
```

上記の文は次のように変換されます。

```
...
ASSGNDD ddname='SYSIN' type='INSTREAM' << !
This is sysin data 1
This is sysin data 2
!
#  * A JCL comment within the instream dataset but not a record
#  * Implied end of SYSIN
ASSGNDD ddname='DD1' dataset='AAA.BB' \\\
       filename='/sys1/aaa/bbb' disp='i-o'
...
```

コード例 4-41　暗黙的な SYSIN DD 文

```
...
//DD1 DD DSN=AAA.BB,DISP=SHR Start of SYSIN follows
This is sysin dataset record 1
This is sysin dataset record 2
/*
...
```

上記の文は次のように変換されます。

```
...
#  * Start of SYSIN follows
ASSGNDD ddname='DD1' dataset='AAA.BB' \\\
       filename='/sys1/aaa/bbb' disp='i-o'
ASSGNDD ddname='SYSIN' type='INSTREAM' << !
This is sysin dataset record 1
This is sysin dataset record 2
!
...
```

連結データセット

複数のデータセットに 1 つの ddname を関連付けられます。アプリケーションプログラムからは、1 つのファイルであるように定義された順序でアクセスされます。連結データセットをコーディングするには、最初の DD 文だけで ddname が指定されます。

Sun MBM は、COBOL プログラム以外では連結データセットをサポートしません。

コード例 4-42　　連結データセットを収めた JCL

```
...
//CIMSUMM DD DSN=CIM01.DATA,DISP=SHR
// DD DSN=CIM02.DATA,DISP=SHR
// DD DSN=CIM03.DATA,DISP=SHR
...
```

上記の文は次のように変換されます。

```
ASSGNDD ddname=CIMSUMM dataset='CIM01.DATA' \\\
       filename='/sys/cim01/data' disp='i-o'
ASSGNDD dataset='CIM02.DATA' \\\
       filename='/sys/cim02/data' disp='i-o'
ASSGNDD dataset='CIM03.DATA' \\\
       filename='/sys/cim03/data' disp='i-o'
...
```

一時データセット

次のいずれかの方法で一時データセットを定義できます。

- 二重アンパサンド &&*datasetname* の使用

```
//TEMPFILA DD DSN=&&TEMP.INV,...
```

- 空の DSN 値の使用

```
//TEMPFILB DD DSN=,...
```

- DD 文からの DSN = パラメータの省略

```
//TEMPFILB DD DISP=(NEW,PASS),...
```

mvstrans は、一時データセットを検出すると、type='TEMP' が指定された ASSGNDD マクロを生成します。

コード例 4-43　　一時データセットの定義を収めたジョブ MYJOB

```
...
//TMPSORT DD DSN=&&SUMMFIL,DISP=(NEW,PASS)
//TMPUPDT DD DISP=(NEW,PASS)
//TMPFILA DD DSN=,DISP=(OLD,PASS)
...
```

上記の文は次のように変換されます。

```
...
ASSGNDD ddname='TMPSORT' dataset='MYJOB_SUMMFIL' \\\
       type='TEMP' disp='o' normal='k'
ASSGNDD ddname='TMPUPDT' type='TEMP' disp='o' normal='k'
ASSGNDD ddname='TMPFILA' type='TEMP' disp='o' normal='k'
...
```

検査モード (-v オプション) で実行すると、mvstrans -g -v を指定しない限り、File_Map に一時データセットのエントリは生成されません。ジョブが実行されて、終了すると常に削除される場合、一意の一時ファイルが動的に作成されます。

EXEC 文

EXEC 文は、現在のジョブステップで実行するプログラムまたは手続きを指定します。呼び出される手続きを含むジョブストリームでの最大ステップは、255 です。

PGM 定位置パラメータ

PGM パラメータは、実行するプログラムの名前を指定します。MVS JCL ジョブステップは、最初の文である EXEC 文から構成され、ステップに関連する他のすべての JCL 文がそのあとに続きます。

コード例 4-44　　EXEC 文の PGM パラメータ

```
...
//STEP01 EXEC PGM=PGM001
//DD1 DD DSN=IN.DATA,DISP=SHR
...
```

上記の文は次のように変換されます。

```
...
#####################################################################
LABEL name=STEP01
#####################################################################
ASSGNDD ddname='DD1' dataset='IN.DATA' \\\
       filename='/sys1/in/data'
EXECPGM pgmname='PGM001' stepname='STEP01'
...
```

mvstrans は、EXEC PGM 文から生成された EXECPGM マクロがジョブステップの最後のマクロ呼び出しになるように、構造を反転します。また、mvstrans は、ジョブステップの開始を示す LABEL マクロを生成します。

PROC 定位置パラメータ

PROC パラメータは、実行する手続きを指定します。次に、記号パラメータが EXEC PROC 文での割り当て値となる例を示します。記号パラメータの詳細は、116 ページの「PROC 文」と117 ページの「SET 文」を参照してください。また、PROC および SET JCL 文のサポートの一覧は、『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。

コード例 4-45　　2 つの EXEC 文からの手続きを実行する文

```
...
//STEP01 EXEC PROC=PROC001,SYM1='XX',SYM2='A$B'
//STEP02 EXEC BPROC2,SYMB1=22,SYMB2=AA
...
```

上記の文は次のように変換されます。

```
...
#####################################################################
#
LABEL name=STEP01
#####################################################################
```

```
EXECPROC procname='PROC001' stepname='STEP01' \\\
       parms='SYM1=XX,SYM2=A_B'

#################################################################
LABEL name=STEP02
#################################################################
EXECPROC procname='BPROC2' stepname='STEP02' \\\
       parms='SYMB1=22,SYMB2=AA'
...
```

COND キーワードパラメータ

COND キーワードは、以前のリターンコードに基づいてテストを定義します。COND パラメータが EXEC 文でコーディングされている場合、ステップの間 JOB 文でコーディングされた COND 文に対して優先されます。

mvstrans は、COND パラメータと ONRETCODE マクロに関連する EVEN または ONLY 条件に対して、COND で指定された残りのすべての条件ごとに、ONCONDCODE マクロを生成します。これらのマクロ呼び出しは、ジョブステップの LABEL マクロのあとに出現します。生成されたマクロ呼び出しで、条件が true のときに Sun MBM が実行する動作が指定されます。EVEN または ONLY 条件が COND 条件リストのどの位置に出現するかにかかわらず、ONCONDCODE マクロは、常に ONRETCODE マクロの前に生成されます。

以前のすべてのステップのリターンコードに対して実行するテストを定義する EXEC 文と COND パラメータ

コード例 4-46　　EXEC 文と COND パラメータ

```
...
//STEP0003 EXEC PGM=PROGA,COND=(4,LT)
...
```

mvstrans によって、条件は、「以前のステップのリターンコードより 4 が小さい場合に現在のステップをバイパスする」から、「以前のステップが返す値が 4 より大きい場合に現在のステップをバイパスする」に変換されます。

```
...
#################################################################
LABEL name=STEP0003
#################################################################
ONRETCODE MAXRC GT 4 BYPASS scope='STEP'
...
EXECPGM pgmname='PROGA' stepname='STEP0003'
...
```

以前のすべてのステップからのリターンコードのテストに加えて、以前のジョブステップからのリターンコードをテストする EXEC 文と COND パラメータ

```
...
//STEP0006 EXEC PGM=ERROR,COND=((8,LT),(4,LE,STEP0001))
...
```

上記の文は次のように変換されます。

```
...
#####################################################################
LABEL name=STEP0006
#####################################################################
ONRETCODE MAXRC GT 8 BYPASS scope='STEP'
ONRETCODE STEP0001 GE 4 BYPASS scope='STEP'
...
EXECPGM pgmname='ERROR' stepname='STEP0006'
...
```

リターンコードは以前のステップからのもので、その 1 つは手続きステップです。

コード例 4-47　　リターンコードの複数のテストを使用する EXEC 文

```
...
//STEP0008 EXEC PGM=PROGA,
// COND=((4,GT,STEP0001),(20,GT,STEP0003.P1))
...
```

上記の文は次のように変換されます。

```
...
#####################################################################
LABEL name=STEP0008
#####################################################################
ONRETCODE STEP0001 LT 4 BYPASS scope='STEP'
ONRETCODE STEP0003.P1 LT 20 BYPASS scope='STEP'
...
EXECPGM pgmname='PROGA' stepname='STEP0008'
...
```

コード例 4-48　　COND パラメータの EVEN および ONLY 条件を使用する EXEC 文

```
...
//STEP0008 EXEC PGM=PROGA,COND=((4,LT),EVEN)
//STEP0009 EXEC PGM=PROGB,COND=ONLY
...
```

上記の文は次のように変換されます。

```
...
################################################################
LABEL name=STEP0008
################################################################
ONCONDCODE NE 0 CONTINUE scope='STEP'
ONRETCODE MAXRC GT 4 BYPASS scope='STEP'
EXECPGM pgmname='PROGA' stepname='STEP0008'

################################################################
LABEL name=STEP0009
################################################################
ONCONDCODE EQ 0 BYPASS scope='STEP'
EXECPGM pgmname='PROGB' stepname='STEP0009'
...
```

次の例に、手続きステップ条件の上書きがどのように変換されるかを示します。

コード例 4-49　　手続きステップ条件の上書きを含む EXEC 文

```
...
//JSTP EXEC PNAME,COND.STEP1=((0,GT),EVEN)
...
```

上記の文は次のように変換されます。

```
...
ONCONDCODE NE 0 CONTINUE scope='STEP' poverride='y' stepname='STEP1'
ONRETCODE MAXRC LT 0 BYPASS scope='STEP' poverride='y' stepname='STEP1'
EXECPROC procname='PNAME' stepname='JSTP'
...
```

PARM キーワードパラメータ

PARM パラメータは、現在のジョブステップで実行されるプログラムに渡されるテキストを指定します。mvstrans は、EXEC PGM 文の PARM パラメータを EXECPGM マクロの parm パラメータに変換します。EXEC PROC 文の PARM パラメータは、部分的にサポートされます。手続きステップで修飾されている場合、mvstrans はそれを無視します。修飾されていない場合、EXECPROC マクロに pgmparm パラメータが生成されます。一部の特殊文字は、次の形式で変換されます。

/x*HH*

この *HH* は、ASCII 文字の 16 進表記です。たとえば、$$ は /x24/x24 に変換されます。

コード例 4-50 PARM パラメータを使用する EXEC PGM 文

```
...
//STEP001 EXEC PGM=PROGA,PARM='With SPACES and $DOLLARS$'
...
```

上記の文は次のように変換されます。

```
...
EXECPGM pgmname='PROGA' stepname='STEP001' \\\
       parm='With_SPACES_and_/x24DOLLARS/x24'
...
```

PARM パラメータ値の空白文字は、下線に変換されます。ただし、COBOL プログラムでは、下線から空白文字に戻された PARM 値を受け取ります。

次に、EXEC PROC 文と 2 つの PARM 値の例を示します。最初の値には、括弧で囲まれた記号パラメータ参照が収められます。2 番目の値は、手続きステップ名で修飾され、無視されます。

```
...
//STEPB EXEC PROC=PROCA,PARM=('Value is quotes',&SYM1),
// PARM.PSTEP2='BB'
...
```

上記の文は次のように変換されます。

```
...
EXECPROC procname='PROCA' stepname='STEPB' \\\
       pgmparm='\'Value_in_quotes\',\${SYM1}'
...
```

IF/THEN/ELSE/ENDIF 文

IF/THEN/ELSE/ENDIF JCL 構造によって、EXEC COND オプションの通常の使用を超えた、JCL ジョブステップの条件付き実行が可能になります。IF 文は、常に条件式と THEN 句の前にあります。

コード例 4-51　　複合条件のある IF/THEN/ELSE/ENDIF 構造

```
//IF1000   IF (STEP0010.RC = 8 AND STEP0020.RC > 0) THEN
...
//EL1000   ELSE
...
//EN1000   ENDIF
```

上記の文は次のように変換されます。

```
LABEL name=IF1000
IF [STEP0010.RC EQ 8 AND STEP0020.RC GT 0] THEN
...

# LABEL name=EL1000
ELSE
...

# LABEL name=EN1000
ENDIF
```

LABEL マクロは、元の JCL IF 文に関連のあるステップ名に基づいて、IF マクロ文の前に生成されることに注意してください。また、入れ子の IF/ELSE/ENDIF 構造があるときに、大規模なマクロジョブの読みやすさを高めて、IF/ELSE/ENDIF の組み分けをより簡単に一致させるために、マクロ LABEL 文の形式でのコメント行が ELSE および ENDIF マクロ用に生成されます。

コード例 4-52　シンプルな条件の IF/THEN/ELSE/ENDIF 構造

```
//IF1013   IF (IF1010.RC > 8) THEN
...
//EL1013   ELSE
...
//EN1013   ENDIF
```

上記の文は次のように変換されます。

```
LABEL name=IF1013
IF [IF1010.RC GT 8] THEN
...

# LABEL name=EL1013
ELSE
...

# LABEL name=EN1013
ENDIF
```

INCLUDE 文

INCLUDE 文は、JCLLIB 文によって定義された PDS メンバーからの JCL 行を現在のジョブストリームにコピーします。

mvstrans は、メンバーからの JCL 文を最終的に変換される JCL ストリームに挿入します。また、メンバーの検索位置の決定に、JCLLIB 文を使用しません。かわりに、$MVSINCLDIR で指定されたディレクトリが検索されます。または、$MVSINCLDIR が設定されていない場合、mvstrans が実行された位置を基準に ./include ディレクトリが検索されます。

この例は、INCLUDE 文を使用して ./include からのファイル INCUST を取り込む場合を示します。

./include/INCUST ファイルの内容が、次のとおりである場合

コード例 4-53 INCLUDE 文

```
//INFILE DD DSN=TEST.CUST,DISP=SHR
//OUTFILE DD DSN=TEST.UPD.CUST,DISP=SHR
```

元の JCL ジョブ

```
//JOB001 JOB
//LIBSRC JCLLIB ORDER=(TEST1.LIB)
//STEP1 EXEC PGM=UPDCUST
//MEMBER1 INCLUDE MEMBER=INCUST
```

mvstrans が JCL を変換する前に上記の文は次のように更新されます。

```
//JOB001 JOB
//LIBSRC JCLLIB ORDER=(TEST1.LIB)
//STEP1 EXEC PGM=UPDCUST
//INFILE DD DSN=TEST.CUST,DISP=SHR
//OUTFILE DD DSN=TEST.UPD.CUST,DISP=SHR
```

JCLLIB 文

JCLLIB 文は、カタログ化された手続きまたはインクルードメンバーを検索するために PDS または PDS のリストを定義します。手続きの検索ライブラリを指定するために使用する場合に限って、mvstrans は、JCLLIB 文をサポートします。検査モード (-v オプション) で実行すると、存在しない場合、File_Map にそれぞれの *dataset-name* のライブラリエントリが作成されます。

File_Map エントリの形式

dataset-name;__LIB;FS;*mapped-dir*;;0;

説明

mapped-dir	移行環境での Sun MBM 手続きスクリプトを収めた *dataset-name* の割り当てディレクトリ。

LIBDEF マクロは、JCLLIB 文で指定した *dataset-name* ごとに生成されます。それぞれの LIBDEF マクロには、JOB の scope パラメータ値があり、ジョブの期間での利用度を示します。

JCLLIB 文によって 2 つの検索ライブラリを指定した JOB

コード例 4-54　　JCLLIB 文

```
//NOR9999 JOB ASHWORTH
//LIBSRCH JCLLIB ORDER=(USER.PROCLIB1,USER.PROCLIB2)
...
```

$FMROOT が /sys1 に設定されている場合、mvstrans -v は、これらのエントリを File_Map に生成します。

```
USER.PROCLIB1;__LIB;FS;/sys1/user/proclib1;;0;
USER.PROCLIB2;__LIB;FS;/sys1/user/proclib2;;0;
```

変換モードで実行された mvstrans は、次のようにマクロジョブを作成します。最初の呼び出しの次の LIBDEF マクロには、定義されるディレクトリのリストが実行時に使用される手続き検索パスに追加されることを示す override='N' パラメータがあります。

```
...
BEGINJOB mode='MVS'
LIBDEF scope='JOB' type='PROC' \\\
       dataset='USER.PROCLIB1' lib='/sys1/user/proclib1'
LIBDEF scope='JOB' type='PROC' override='N' \\\
       dataset='USER.PROCLIB2' lib='/sys1/user/proclib2'
...
```

JOB 文

JOB 文は、ジョブの開始を定義し、ジョブの処理法に関連したパラメータを提供します。

mvstrans は、JOB 文に BEGINJOB マクロを生成します。mode と呼ばれるパラメータは、BEGINJOB で使用されます。パラメータの値は、'MVS' で、ジョブの残りでマクロが MVS JCL と同様に動作することを示しています。

アカウンティング情報

ジョブ固有の情報を提供します。mvstrans は、次のアカウンティングサブパラメータをサポートしています。

sysout-lines	SETPRINT マクロに lines パラメータを設定します。
cards	SETPRINT マクロに cards パラメータを設定します。
forms	SETPRINT マクロに forms パラメータを設定します。
copies	SETPRINT マクロに copies パラメータを設定します。
line-count	SETPRINT マクロに linect パラメータを設定します。

mvstrans では、JOB 文のアカウンティング情報パラメータに出現する場合のある JES2 印刷オプションのサブパラメータ *sysout-lines*、*cards*、*forms*、*copies*、および *line-count* をサポートします。SETPRINT マクロは、JOB 文用に取得した値に設定された適切なパラメータで生成されます。

コード例 4-55　JOB 文と JES2 印刷オプション

```
//JES2JOBA JOB (,,,4000,8888,FFM1,25,,100)
...
```

上記の文は次のように変換されます。

```
...
BEGINJOB mode='MVS'
SETPRINT printid='ALL' scope='JOB' lines='4000' \\\
       cards='8888' forms='FFM1' copies='25' linect='100'
...
```

COND キーワードパラメータ

COND パラメータは、以前のリターンコードに基づいて実行するテストを定義します。mvstrans は、JOB 文の COND パラメータで指定されたそれぞれの条件用に ONRETCODE マクロを生成します。これらのマクロは、BEGINJOB マクロのあとになります。条件が true のときに Sun MBM が実行する動作は、生成されたマクロで指定されます。

次の JOB 文と COND パラメータは、直前に完了したステップからのリターンコードをテストする 1 つの条件を定義します。テストは、ジョブでそれぞれのステップを実行する前に実行されます。

コード例 4-56　　JOB 文と COND パラメータ

```
//JOB001 JOB 678,SMITH,COND=(16,LE)
...
```

mvstrans によって、条件は、「16 が直前に完了したステップのリターンコード以下である場合に、ジョブを終了する」から「直前に完了したステップのリターンコードが 16 以上である場合に、ジョブを終了する」に変換されます。

```
...
BEGINJOB mode='MVS'
ONRETCODE GE 16 GOTO END_JOB
...
```

次の EXEC 文と COND パラメータは、直前に完了したステップからのリターンコードをテストする複数の条件を定義します。テストは、ジョブでそれぞれのステップを実行する前に実行されます。

```
...
//STEP0006 EXEC PGM=ERROR,COND=((8,LT),(4,EQ))
...
```

上記の文は次のように変換されます。

```
...
BEGINJOB mode='MVS'
ONRETCODE GT 8 GOTO END_JOB
ONRETCODE EQ 4 GOTO END_JOB
...
```

LINES キーワードパラメータ

LINES パラメータは、このジョブでの sysout データセットでスプールする最大行数を指定します。値は、1000 行単位で解釈されます。

コード例 4-57　JOB 文と LINES パラメータ

```
...
/TEST1 JOB LINES=30
...
```

上記の文は次のように変換されます。

```
...
BEGINJOB mode='MVS'
SETPRINT printid='ALL' scope='JOB' lines='30'
...
```

MSGCLASS キーワードパラメータ

ジョブログをクラスに割り当てます。mvstrans は、このパラメータを BEGINJOB マクロの jobclass パラメータに変換します。指定しない場合、Sun MBM はクラス a をジョブ実行時のデフォルトとして割り当てます。ジョブの実行時に、CLASS 環境変数は、指定した値に設定され、選択したスプールメカニズムからこの値を使用できます。詳細は、132 ページの「印刷出力の処理」を参照してください。

コード例 4-58　JOB 文と MSGCLASS パラメータ

```
//JOB001 JOB ACT44,MSGCLASS=D
...
```

上記の文は次のように変換されます。

```
...
BEGINJOB mode='MVS' jobclass='D'
...
```

PASSWORD キーワードパラメータ

USER パラメータに関連付けられているパスワードを示します。詳細は、コード例 4-59 を参照してください。

USER キーワードパラメータ

ジョブをサブミットした人を指定します。

コード例 4-59　　JOB 文と USER および PASSWORD パラメータ

```
//TSTJOB JOB ,,USER=johnj,PASSWORD=cx4tn8
...
```

上記の文は次のように変換されます。

```
...
BEGINJOB mode='MVS' username='johnj' password='cx4tn8'
...
```

OUTPUT 文

OUTPUT 文は、sysout データセットの処理オプションを定義します。

OUTPUT キーワードパラメータ

OUTPUT 文のすべてのパラメータはキーワードで、任意の順序でコーディングされます。DEFAULT の例外はありますが、キーワードパラメータは sysout データセットに適用され mvstrans がサポートする印刷オプションを定義します。

sysout データセットは、DD 文と SYSOUT パラメータによって定義されます。mvstrans は、OUTPUT 文で指定された印刷オプションに SETPRINT マクロを生成します。SETPRINT マクロには、OUTPUT 文のタイプを示す *scope* パラメータがあります。

印刷オプション値が引用符または括弧で囲まれている場合、mvstrans は引用符または括弧を角括弧に変換します。

SETPRINT マクロで設定された印刷オプションは、Sun MBM スプールユーティリティーで使用されます。Sun 以外のスプールパッケージでも印刷オプション値を取得できます。

BURST キーワードパラメータ

BURST キーワードは、sysout データセットの印刷方法を定義します。

コード例 4-60　　デフォルト以外のジョブレベルの OUTPUT 文と BURST パラメータ

```
//JOBA JOB
//OUT1 OUTPUT BURST=Y
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUT1' scope='JOB' burst='Y'
...
```

CHARS キーワードパラメータ

CHARS キーワードは、sysout データセットの印刷に使用する文字配列テーブルを定義します。

コード例 4-61　　デフォルト以外のステップレベルの OUTPUT 文と CHARS パラメータ

```
...
//STEP1 EXEC PGM=AAA
//OUT1 OUTPUT CHARS=(GRI1,GRI2)
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUT1' scope='STEP' chars='[GRI1,GRI2]'
...
```

CKPTLINE キーワードパラメータ

CKPTLINE キーワードは、論理ページに収められる最大行数を定義します。

コード例 4-62　デフォルトのジョブレベルの OUTPUT 文と CKPTLINE パラメータ

```
//JOBA JOB
//OUTX OUTPUT DEFAULT=YES,CKPTLINE=4000
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTX' scope='JOB_DEFAULT' ckptline='4000'
...
```

CKPTPAGE キーワードパラメータ

CKPTPAGE キーワードは、チェックポイントの前に印刷したりワークステーションに送信する、最大論理ページ数を定義します。

コード例 4-63　デフォルトのステップレベルの OUTPUT 文と CKPTPAGE パラメータ

```
//STEP1 EXEC PGM=PROGA
//OUTSTEP1 OUTPUT DEFAULT=YES,CKPTPAGE=200
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTSTEP1' scope='STEP_DEFAULT' ckptpage='200'
...
```

CKPTSEC キーワードパラメータ

CKPTSEC キーワードは、sysout データセットの印刷時の、チェックポイント間の経過秒数を定義します。

コード例 4-64　　ステップレベルの OUTPUT 文と CKPTSEC パラメータ

```
...
//STEPXX EXEC PGM=XOX1
//OUTXX OUTPUT CKPTSEC=25
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTXX' scope='STEP' ckptsec='25'
...
```

CLASS キーワードパラメータ

CLASS キーワードは、sysout データセットを特定のクラスに割り当てます。

コード例 4-65　　ステップレベルの OUTPUT 文と CLASS パラメータ

```
...
//STEP4X05 EXEC PGM=GENREP1
//OUT4X5 OUTPUT CLASS='A'
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUT4X5' scope='STEP' class='A'
...
```

COMPACT キーワードパラメータ

COMPACT キーワードは、sysout SNA データセットを SNA 遠隔端末に送信するときに使用する圧縮テーブルを特定します。

コード例 4-66　　ジョブレベルの OUTPUT 文と COMPACT パラメータ

```
...
//JOBWORK JOB
//OUT4X5 OUTPUT COMPACT='TBL278'
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUT4X5' scope='JOB' compact='TBL278'
...
```

CONTROL キーワードパラメータ

CONTROL キーワードは、sysout データセットを印刷する行間隔を定義します。

コード例 4-67　　ジョブレベルの OUTPUT 文と CONTROL パラメータ

```
//JOBWORK JOB
//REPOUT OUTPUT CONTROL=PROGRAM
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='REPOUT' scope='JOB' control='PROGRAM'
...
```

COPIES キーワードパラメータ

COPIES キーワードは、印刷するコピー部数を指定します。

コード例 4-68　　デフォルトのステップレベルの OUTPUT 文と COPIES パラメータ

```
...
//STEPREP1 EXEC PGM=COBREP1
//ROUT OUTPUT COPIES=(,(7,24,17,79)),DEFAULT=Y
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='ROUT' scope='STEP_DEFAULT' \\\
       copies='[,[7,24,17,79]]'
...
```

DATACK キーワードパラメータ

DATACK キーワードは、プリント位置と無効な印刷文字の処理方法を定義します。

コード例 4-69　　ジョブレベルの OUTPUT 文と DATACK パラメータ

```
//JOBWK1 JOB
//OUTALL OUTPUT DATACK=BLKPOS
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTALL' scope='JOB' datack='BLKPOS'
...
```

DEFAULT キーワードパラメータ

OUTPUT 文から生成された SETPRINT マクロの *scope* パラメータ値は、JCL での OUTPUT 文の位置に応じて、DEFAULT パラメータによって決定されます。Y または YES の DEFAULT 値は、SETPRINT マクロの *scope* パラメータ値に _DEFAULT を追加します。これは、*printid* パラメータ値に関連付けられた印刷オプションが、OUTPUT パラメータのない DD SYSOUT 文によって定義された sysout データセットだけに割り当てられることを SETPRINT マクロに示します。

コード例 4-70　デフォルトのジョブレベルの OUTPUT 文と印刷オプション

```
...
//JOB1 BEGINJOB ...
//OJOBDEF OUTPUT DEFAULT=YES,DEST=RMT1,FLASH=STD,BURST=N
...
```

上記の文は次のように変換されます。

```
...
BEGINJOB mode='MVS'
SETPRINT printid='OJOBDEF' scope='JOB_DEFAULT' \\\
       dest='RMT1' flash='STD' burst='n'
...
```

DEST キーワードパラメータ

DEST キーワードは、sysout データセットの宛先を定義します。

コード例 4-71　デフォルトのステップレベルの OUTPUT 文と DEST パラメータ

```
...
//STEPREP1 XEC PGM=COBREP1
//ROUT OUTPUT DEST=SMO45X.DP501,DEFAULT=Y
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='ROUT' scope='STEP_DEFAULT' \\\
       dest='SMO45X.DP501'
...
```

FCB キーワードパラメータ

用紙制御イメージを定義します。

コード例 4-72　デフォルト以外のステップレベルの OUTPUT 文と FCB パラメータ

```
...
//STEPREP1 EXEC PGM=COBREP1
//ROUT OUTPUT FCB=FF21
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='ROUT' scope='STEP' \\\
       fcb='FF21'
...
```

FLASH キーワードパラメータ

FLASH パラメータは、フォームオーバーレイと使用するフォームのコピー部数を指定します。

コード例 4-73　デフォルト以外のステップレベルの OUTPUT 文と FLASH パラメータ

```
...
//STEPREP1 EXEC PGM=PLIREP1
//OSTEP1 OUTPUT FLASH=(LTTD,8)
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OSTEP1' scope='STEP' \\\
       flash='[LTTD,8]'
...
```

FORMDEF キーワードパラメータ

FORMDEF パラメータは、文とオーバーレイフォーム情報を収めたライブラリメンバーを指定します。

コード例 4-74　デフォルトジョブレベルの OUTPUT 文と FORMDEF パラメータ

```
//JOBACCT JOB
//OUTDEF OUTPUT FORMDEF=FMDF1,DEFAULT=Y
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTDEF' scope='JOB_DEFAULT' \\\
       formdef='FMDF1'
...
```

FORMS キーワードパラメータ

FORMS パラメータは、sysout データセットを印刷する書式名を指定します。

コード例 4-75　デフォルトジョブレベルの OUTPUT 文と FORMS パラメータ

```
//JOBACCT JOB
//OUTDEF OUTPUT FORMS=FFFRM2,DEFAULT=YES
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTDEF' scope='JOB_DEFAULT' \\\
       forms='FFFRM2'
...
```

GROUPID キーワードパラメータ

GROUPID パラメータは、sysout データセットが出力グループに関連付けられることを指定します。

コード例 4-76　　ジョブレベルの OUTPUT 文と GROUPID パラメータ

```
//JOBACCT JOB
//OUTDEF OUTPUT GROUPID=X11RRT1
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTDEF' scope='JOB' \\\
       groupid='X11RRT1'
...
```

INDEX キーワードパラメータ

INDEX キーワードは、sysout データセットの左マージンを定義します。

コード例 4-77　　ステップレベルの OUTPUT 文と INDEX パラメータ

```
//JOBSTEP1 EXEC PGM=HRICE1
//STEPOUT OUTPUT INDEX=30
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='STEPOUT' scope='STEP' \\\
       index='30'
...
```

JESDS キーワードパラメータ

この OUTPUT 文で指定されたパラメータに従って、ジョブの一覧表示とログを処理するようにシステムに指示します。

コード例 4-78　　ステップレベルの OUTPUT 文と JESDS パラメータ

```
//JOBSTEPN EXEC PGM=PREP1
//STPOUT OUTPUT JESDS='LOG'
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='STPOUT' scope='STEP' \\\
       jesds='LOG'
...
```

LINDEX キーワードパラメータ

LINDEX キーワードは、sysout データセットの右マージンを定義します。

コード例 4-79　　デフォルトのステップレベルの OUTPUT 文と LINDEX パラメータ

```
...
//STEP004 EXEC PGM=PREP1
//SOUT1 OUTPUT DEFAULT='Y',LINDEX=4
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='SOUT1' scope='STEP_DEFAULT' \\\
       lindex=4
...
```

LINECT キーワードパラメータ

LINECT パラメータは、ページごとの印刷行数を指定します。

コード例 4-80　　ステップレベルの OUTPUT 文と LINECT パラメータ

```
...
//STEP6 EXEC PGM=PREP1
//OUTS6 OUTPUT LINECT=255
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTS6' scope='STEP' \\\
       linect='255'
...
```

MODIFY キーワードパラメータ

MODIFY パラメータは、コピー修正モジュールを特定します。

コード例 4-81　　ステップレベルの OUTPUT 文と MODIFY パラメータ

```
...
//STP06X08 EXEC PGM=PREP1
//OUT6 OUTPUT MODIFY=(MODB,0)
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUT6' scope='STEP' \\\
       modify='[MODB,0]'
...
```

PAGEDEF キーワードパラメータ

PAGEDEF パラメータは、ページモードプリンタでの印刷時に使用される文を収めたコピーライブラリメンバーを特定します。

コード例 4-82　ステップレベルの OUTPUT 文と PAGEDEF パラメータ

```
...
//STP06X08 EXEC PGM=PREP1
//OUT6 OUTPUT PAGEDEF=PSFM1
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUT6' scope='STEP' \\\
       pagedef='PSFM1'
...
```

PIMSG キーワードパラメータ

印刷エラーメッセージを一覧表示する方法を制御します。

コード例 4-83　ステップレベルの OUTPUT 文と PIMSG パラメータ

```
...
//STP7X7 EXEC PGM=PREP1
//OUT7X7 OUTPUT PIMSG=(YES,99)
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUT7X7' scope='STEP' \\\
       pimsg='[YES,99]'
...
```

PRMODE キーワードパラメータ

PRMODE パラメータは、sysout データセットのプロセスモードを指定します。

コード例 4-84　デフォルトジョブレベルの OUTPUT JCL 文と PRMODE パラメータ

```
//JOBACCT JOB
//OUTDEF OUTPUT PRMODE=PAGE,DEFAULT=Y
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTDEF' scope='JOB_DEFAULT' \\\
       prmode='PAGE'
...
```

PRTY キーワードパラメータ

PRTY キーワードは、出力キューでの sysout データセットの優先順位を定義します。

コード例 4-85　デフォルトジョブレベルの OUTPUT 文と PRTY パラメータ

```
//JOBACCT JOB
//OUTDEF OUTPUT PRTY=100,DEFAULT=Y
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTDEF' scope='JOB_DEFAULT' \\\
       prty='100'
...
```

THRESHLD キーワードパラメータ

THRESHLD キーワードは、sysout データセットの最大サイズを定義します。

コード例 4-86　ジョブレベルの OUTPUT 文と THRESHLD パラメータ

```
//JOBACCT JOB
//OUTDEF OUTPUT THRESHLD=86400
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTDEF' scope='JOB' \\\
       threshld='86400'
...
```

TRC キーワードパラメータ

TRC パラメータは、sysout データセットのそれぞれの論理レコードにテーブル参照文字コードが含まれるかどうかを指定します。

コード例 4-87　デフォルトのジョブレベルの OUTPUT 文と TRC パラメータ

```
//JOBA JOB
//OUTDEF OUTPUT TRC=Y,DEFAULT=Y
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTDEF' scope='JOB_DEFAULT' \\\
       trc='Y'
...
```

UCS キーワードパラメータ

UCS パラメータは、使用する汎用文字セットイメージを指定します。

コード例 4-88　　ステップレベルの OUTPUT 文と UCS パラメータ

```
...
//STP7X7 EXEC PGM=PREP1
//OUT7 OUTPUT UCS=B23
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUT7' scope='STEP' \\\
       ucs='B23'
...
```

WRITER キーワードパラメータ

WRITER パラメータは、sysout データセットを処理する外部の書き込み側を特定します。

コード例 4-89　　ステップレベルの OUTPUT 文と WRITER パラメータ

```
...
//STEP002 EXEC PGM=PRNTREP
//OUTRP OUTPUT WRITER=WRTR005
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTRP' scope='STEP' \\\
       writer='WRTR005'
...
```

OUTPUT 文の例

MVS JCL OUTPUT 文は、SETPRINT マクロに変換されます。SETPRINT マクロの printid パラメータは、OUTPUT 文のラベル名に設定されます。これは、OUTPUT 文からの印刷オプションを sysout データセットに適用するために、DD SYSOUT 文で使用される値です。SETPRINT マクロの *scope* パラメータは、OUTPUT 文の位置とタイプに従って設定されます。

OUTPUT 文がジョブでの最初の EXEC 文の前に配置される場合、JOB に設定します。

最初の EXEC 文の前に配置され、Y または YES の DEFAULT パラメータ値を持つ場合、JOB_DEFAULT の値になります。

OUTPUT 文がジョブステップ内に配置される場合、STEP の値になります。

ジョブステップ内にあり、Y または YES の DEFAULT パラメータ値を持つ場合、STEP_DEFAULT の値になります。

OUTPUT 文のその他のパラメータは、すべて直接 SETPRINT マクロに変換されます。

印刷出力の処理と変換の例は、132 ページの「印刷出力の処理」を参照してください。

コード例 4-90　デフォルトのジョブレベルとステップレベルの OUTPUT 文

```
//JOB001 JOB
//OUT1 OUTPUT BURST=YES,CLASS=A,COPIES=2,DEFAULT=Y
//STEP001 EXEC PGM=PRNT0001
//OUT2 OUTPUT COPIES=3
//*
//* The following sysout dataset will have print options
//* applied from step level OUTPUT statement with label OUT2.
//RPRT001 DD SYSOUT=(,REPORT1),CLASS=D,OUTPUT=*.OUT2
//*
//* The sysout dataset below will have print options applied
//* from default job level OUTPUT statement with label OUT1.
//RPRT002 DD SYSOUT=*
```

上記の文は次のように変換されます。

```
BEGINJOB mode='MVS'
SETPRINT printid='OUT1' scope='JOB_DEFAULT' \\\
       burst='YES' class='A' copies='2'
################################################################
LABEL name=STEP001
################################################################
SETPRINT printid='OUT2' scope='STEP' \\\
       copies='3'
#  *
```

```
#  * The following sysout dataset will have print options
#  * applied from step level OUTPUT statement with label OUT2.
SETPRINT ddname='RPRT001' writer='REPORT1'
ASSGNDD ddname='RPRT001' type='SYSOUT' \\\
       class='OUTCLASS' printid='OUT2'
#  *
#  * The sysout dataset below will have print options applied
#  * from default job level OUTPUT statement with label OUT1.
ASSGNDD ddname='RPRT002' type='SYSOUT' class='JOBCLASS'
EXECPGM pgmname='PRNT0001' stepname='STEP001'
ENDJOB
```

PEND 文

PEND 文がカタログ化された手続きに配置される場合、mvstrans は、これを手続きでの JCL 文の終了と解釈します。

PROC 文

PROC 文は、インストリーム手続きまたはカタログ化された手続きの開始を定義し、手続きで使用されるパラメータのデフォルト値を提供します。

mvstrans は、PROC JCL 文から BEGINPROC マクロを生成します。BEGINPROC マクロの parms パラメータで、すべての記号パラメータの割り当てが指定されます。それぞれの記号割り当ては、カンマで区切られます。

次の例では、PROC 文は UPDTMAST という手続きファイル内にあります。

コード例 4-91　PROC 文

```
//PROCA PROC XXX=123,YY=,Z=A&&B,SP='A# B,F=',Q='D&&F'
...
```

上記の文は次のように変換されます。

```
BEGINPROC procname='PROCA' parms='XXX=123,YY=,Z=A_B,SP=A#_B/x2CF=,Q=D/x26/x26F'
...
```

Sun MBM は、インストリーム手続きをサポートしません。BEGINPROC マクロで生成される手続き名は、手続きを収めたファイルの名前と同じです。

記号パラメータ Z での単一 & 文字を表す && は、下線 (_) に変換されます。一方、&& を表す記号パラメータ Q での && は、/x26/x26 に変換されます。

記号パラメータ SP での空白は、下線に変更されます。

mvstrans で特殊文字を変換する方法の詳細は、『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。

SET 文

SET 文を使用して、記号パラメータに値を割り当てます。mvstrans は、SET JCL 文から SETPARM マクロを生成します。SETPARM マクロの parms パラメータで、記号パラメータの割り当てが指定されます。それぞれの記号割り当ては、カンマで区切られます。

コード例 4-92 SET 文

```
...
//SETPRMS SET PARMA=AA&&A,PARMB='B B&&B'
...
```

上記の文は次のように変換されます。

```
...
SETPARM parms='PARMA=AA_A,PARMB=B_B/x26/x26B'
...
```

記号パラメータ PARMA での単一 & を表す && は、下線に変換されます。一方、&& を表す記号パラメータ PARMB での && は、/x26/x26 に変換されます。

記号パラメータ PARMB での空白は、下線に変更されます。特殊文字についての詳細は、『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。

JES2 変換の例

この節では、サポートされる JES2 文とパラメータの変換例を説明します。

/*JOBPARM 文

Sun MBM では、JES2 の /*JOBPARM 文で指定されたすべての印刷オプションパラメータをサポートします。mvstrans は、/*JOBPARM 文に SETPRINT マクロと印刷関連のパラメータを生成します。変換時に、mvstrans では、その他のすべてのパラメータを無視します。

BURST キーワードパラメータ

BURST パラメータは、sysout データセットのデフォルトのバースト特性を指定します。

コード例 4-93　　JES2 の /*JOBPARM 文と BURST パラメータ

```
...
/*JOBPARM BURST=Y
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='ALL' scope='JOB' burst='Y'
...
```

BYTES キーワードパラメータ

BYTES パラメータは、システムがジョブから生成する最大バイト数を指定します。

コード例 4-94　JES2 の /*JOBPARM 文と BURST パラメータ

```
...
/*JOBPARM BYTES=500000
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='ALL' scope='JOB' bytes='500000'
...
```

CARDS キーワードパラメータ

CARDS パラメータは、ジョブの sysout データセットにパンチする出力カードの最大数を指定します。

コード例 4-95　JES2 の /*JOBPARM 文と CARDS パラメータ

```
...
/*JOBPARM CARDS=45545500
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='ALL' scope='JOB' cards='45545500'
...
```

COPIES キーワードパラメータ

COPIES パラメータは、ジョブの sysout データセットにパンチする出力カードの最大数を指定します。

コード例 4-96　　JES2 の /*JOBPARM 文と COPIES パラメータ

```
...
/*JOBPARM COPIES=5
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='ALL' scope='JOB' copies='5'
...
```

FORMS キーワードパラメータ

FORMS パラメータは、sysout データセットに使用する印刷書式を指定します。

コード例 4-97　　JES2 の /*JOBPARM 文と FORMS パラメータ

```
...
/*JOBPARM FORMS=FFXX1
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='ALL' scope='JOB' forms='FFXX1'
...
```

LINECT キーワードパラメータ

LINECT パラメータは、このジョブで sysout データセットのそれぞれの出力ページに印刷する最大行数を指定します。

コード例 4-98　JES2 の /*JOBPARM 文と LINECT パラメータ

```
...
/*JOBPARM LINECT=205
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='ALL' scope='JOB' linect='205'
...
```

LINES キーワードパラメータ

LINES パラメータは、このジョブでの sysout データセットでスプールする最大行数を指定します。値は、1000 行単位で解釈されます。

コード例 4-99　JES2 の /*JOBPARM 文と LINES パラメータ

```
...
/*JOBPARM LINES=12345
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='ALL' scope='JOB' lines='12345'
...
```

PAGES キーワードパラメータ

PAGES パラメータは、ジョブの sysout データセットでの印刷する出力ページの最大部数を指定します。

コード例 4-100　JES2 の /*JOBPARM 文と PAGES パラメータ

```
...
/*JOBPARM PAGES=555000
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='ALL' scope='JOB' pages='555000'
...
```

ROOM キーワードパラメータ

ROOM パラメータは、プログラマのルーム番号を指定します。

コード例 4-101　JES2 の /*JOBPARM 文と ROOM パラメータ

```
...
/*JOBPARM ROOM=RM1
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='ALL' scope='JOB' room='RM1'
...
```

/*OUTPUT 文

Sun MBM では、JES2 の /*OUTPUT 文で指定されるすべての印刷オプションパラメータをサポートしています。mvstrans は、/*OUTPUT 文に SETPRINT マクロと印刷関連のパラメータを生成します。

BURST キーワードパラメータ

印刷する sysout データセットの経路指定方法を決定します。

コード例 4-102 JES2 の /*OUTPUT 文と BURST パラメータ

```
...
/*OUTPUT OUTA BURST=Y
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       burst='Y'
...
```

CHARS キーワードパラメータ

CHARS キーワードは、sysout データセットを印刷するための文字配列テーブルを定義します。

コード例 4-103 JES2 の /*OUTPUT 文と CHARS パラメータ

```
...
/*OUTPUT OUTA CHARS=CHR1
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       chars='CHR1'
...
```

CKPTLNS キーワードパラメータ

CKPTLNS キーワードは、論理ページに収められる最大行数を定義します。

コード例 4-104　JES2 の /*OUTPUT 文と CKPTLNS パラメータ

```
...
/*OUTPUT OUTA CKPTLNS=2000
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       ckptlns='2000'
...
```

CKPTPGS キーワードパラメータ

CKPTPGS キーワードは、チェックポイントの前に印刷したり SNA ワークステーションに送信したりする最大論理ページ数を定義します。

コード例 4-105　JES2 の /*OUTPUT 文と CKPTPGS パラメータ

```
...
/*OUTPUT OUTA CKPTPGS=100
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       ckptpgs='100'
...
```

COMPACT キーワードパラメータ

COMPACT パラメータは、sysout SNA データセットを遠隔 SNA 端末に送信するときに使用する圧縮テーブルを特定します。

コード例 4-106　JES2 の /*OUTPUT 文と COMPACT パラメータ

```
...
/*OUTPUT OUTA COMPACT=CMP002
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y \\\
       compact='CMP002'
...
```

COPIES キーワードパラメータ

COPIES パラメータは、印刷する sysout データセットのコピー部数を指定します。

コード例 4-107　JES2 の /*OUTPUT 文と COPIES パラメータ

```
...
/*OUTPUT OUTA COPIES=8
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       copies='8'
...
```

COPYG キーワードパラメータ

COPYG パラメータは、次のページを印刷する前に印刷される、各ページのコピー部数を指定します。

コード例 4-108　JES2 の /*OUTPUT 文と COPYG パラメータ

```
...
/*OUTPUT OUTA COPYG=2
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       copyg='2'
...
```

DEST キーワードパラメータ

DEST キーワードは、sysout データセットの宛先を定義します。

コード例 4-109　JES2 の /*OUTPUT 文と DEST パラメータ

```
...
/*OUTPUT OUTA DEST=RMT1
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       dest='rmt1'
...
```

FCB キーワードパラメータ

FCB キーワードは、用紙制御イメージを定義します。

コード例 4-110　JES2 の /*OUTPUT 文と FCB パラメータ

```
...
/*OUTPUT OUTA FCB=STD2
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       fcb='STD2'
...
```

FLASH キーワードパラメータ

FLASH パラメータは、フォームオーバーレイと使用するフォームのコピー部数を指定します。

コード例 4-111　JES2 の /*OUTPUT 文と FLASH パラメータ

```
...
/*OUTPUT OUTA FLASH=OVL1
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       flash='OVL1'
...
```

FLASHC キーワードパラメータ

FLASHC キーワードは、最初に印刷されるコピーで始まるオーバーレイで JES2 がフラッシュするコピー部数を定義します。

コード例 4-112　JES2 の /*OUTPUT 文と FLASHC パラメータ

```
...
/*OUTPUT OUTA FLASHC=2
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       flashc='2'
...
```

FORMS キーワードパラメータ

FORMS パラメータは、sysout データセットを印刷するフォームの名前を指定します。

コード例 4-113　JES2 の /*OUTPUT 文と FORMS パラメータ

```
...
/*OUTPUT OUTA FORMS=FRM1
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       forms='FRM1'
...
```

INDEX キーワードパラメータ

INDEX キーワードは、sysout データセットの左マージンを定義します。

コード例 4-114　JES2 の /*OUTPUT 文と INDEX パラメータ

```
...
/*OUTPUT OUTA INDEX=3,...
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       index='3'
...
```

LINDEX キーワードパラメータ

LINDEX キーワードは、sysout データセットの右マージンを定義します。

コード例 4-115　JES2 の /*OUTPUT 文と LINDEX パラメータ

```
...
/*OUTPUT OUTA LINDEX=6
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       lindex='6'
...
```

LINECT キーワードパラメータ

LINEC パラメータは、ページごとの最大印刷行数を指定します。

コード例 4-116　JES2 の /*OUTPUT 文と LINECT パラメータ

```
...
/*OUTPUT OUTA LINECT=40
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       linect='40'
...
```

MODIFY キーワードパラメータ

MODIFY パラメータは、コピー修正モジュールを特定します。

コード例 4-117　JES2 の /*OUTPUT 文と MODIFY パラメータ

```
...
/*OUTPUT OUTA MODIFY=TAB1
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       modify='TAB1'
...
```

MODTRC キーワードパラメータ

MODTRC パラメータは、リストでの位置に基づいて、CHARS パラメータで定義された、使用するテーブル名を指定します。

コード例 4-118　JES2 の /*OUTPUT 文と MODTRC パラメータ

```
...
/*OUTPUT OUTA MODTRC=0,MODTRC=(CHR1,CHR2)
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       modtrc='0' modtrc='[CHR1,CHR2]'
...
```

UCS キーワードパラメータ

UCS パラメータは、使用する汎用文字セットイメージを指定します。

コード例 4-119　JES2 の /*OUTPUT 文と UCS パラメータ

```
...
/*OUTPUT OUTA UCS=UCS1
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUTA' scope='JOB' overridedd='Y' \\\
       ucs='UCS1'
...
```

/*ROUTE 文

/*ROUTE 文を使用して、DEST パラメータによって経路指定されない sysout データセットの宛先を指定したり、ジョブが実行されるネットワークノードを特定したりします。

Sun MBM では、JES2 の /*ROUTE 文の PRINT パラメータをサポートしています。mvstrans は、SETPRINT マクロを生成し、変換中、その他のすべてのパラメータを無視します。

次の変換例に、mvstrans でのさまざまな形式の /*ROUTE PRINT 文の変換を示します。

コード例 4-120 JES2 の /*ROUTE 文

```
...
/*ROUTE PRINT S007
/*ROUTE PRINT S007.OUTR
/*ROUTE PRINT S007:OUTR
/*ROUTE PRINT S007/OUTR
/*ROUTE PRINT S007(OUTR)
...
```

上記の文は次のように変換されます。

```
...
SETPRINT dest='S007' printid='ALL' scope='JOB' overridedd='N'
SETPRINT dest='S007.OUTR' printid='ALL' scope='JOB' overridedd='N'
SETPRINT dest='S007.OUTR' printid='ALL' scope='JOB' overridedd='N'
SETPRINT dest='S007.OUTR' printid='ALL' scope='JOB' overridedd='N'
SETPRINT dest='S007.OUTR' printid='ALL' scope='JOB' overridedd='N'
...
```

印刷出力の処理

この節では、mvstrans を使って、印刷出力を処理する Sun MBM マクロを生成する方法を説明します。

MVS JCL の JOB、DD、および OUTPUT 文と JES2 の /*JOBPARM と /*OUTPUT 文を使って、システム出力 (印刷出力) ファイルとファイルの出力に使用する処理オプション (印刷属性) を指定します。

Sun MBM の現行バージョンでは、JES3 印刷関連の文をサポートしていません。

OUTPUT JCL 文のタイプ

OUTPUT JCL 文には 2 つのタイプがあります。

ジョブレベル	ジョブレベルの OUTPUT 文は、JOB 文のあと、最初の EXEC 文の前にコーディングされます。
ステップレベル	ステップレベルの OUTPUT 文は、EXEC 文のあとにコーディングされます。

OUTPUT JCL 文の参照

DD SYSOUT 文で OUTPUT JCL 文を参照するには、次の 2 とおりの方法があります。

暗黙的	DD SYSOUT 文には、OUTPUT パラメータが含まれません。ステップレベルまたはジョブレベルのデフォルトの OUTPUT 文が存在します。デフォルトの OUTPUT 文は、DEFAULT=Y パラメータで特定されます。
明示的	DD SYSOUT 文には、1 つまたは複数の OUTPUT 文を参照する OUTPUT パラメータが含まれます。

/*OUTPUT JES2 文が DD SYSOUT の 3 番目の定位置パラメータで参照される場合、DD 文の OUTPUT パラメータを使った OUTPUT JCL 文の参照はできません。

/*OUTPUT JES2 文の参照

JES2 /*OUTPUT 文は、DD SYSOUT パラメータの 3 番目の定位置サブパラメータを使って、DD SYSOUT 文で参照されます。

post_exec_pgm シェルスクリプトの使用

SETPRINT および ASSGNDD マクロに指定した印刷オプションで、環境変数が生成されます。ジョブで各プログラムを実行したあと、ユーザー作成のシェルスクリプトでこの環境変数を使用できます。プログラムの実行後、EXECPGM マクロでは、ユーザー作成のシェルスクリプト post_exec_pgm が $PUBLIC/bin ディレクトリにあれば、このスクリプトを実行します。

Sun MBM では、post_exec_pgm.LP などの post_exec_pgm スクリプトのいくつかのスケルトンが $PUBLIC/bin/ に用意されています。このスクリプトを変更し、UNIX lp コマンドを使って UNIX LP サブシステムに印刷要求をサブミットできます。

Sun MBM には、Unibol SI-Spool 製品へのインタフェースも用意されています。詳細は、ご購入先までお問い合わせください。

印刷オプションの処理順序

システム出力用ファイルを定義する DD SYSOUT 文は、次の表に示した順序で処理オプションに適用されます。表 4-6 で、JCL 環境での処理を説明し、表 4-7 で、JES2 を使用する場合の処理を説明します。

表 4-6 JCL SYSOUT DD 印刷処理

DD SYSOUT 文の条件	処理順序
JOB 文から。	JCL OUTPUT および DD SYSOUT 文に対応するオプションが存在する場合、JOB 文のオプションより優先されます。
DD SYSOUT 文と 1 つまたは複数の明示的に参照される OUTPUT JCL 文から。	対応するオプションが両方に存在する場合、DD オプションが優先されます。
DD SYSOUT 文と暗黙的に参照されるステップレベルのデフォルトの OUTPUT JCL 文から。	DD SYSOUT オプションが、対応する OUTPUT オプションより優先されます。
DD SYSOUT 文と暗黙的に参照されるジョブレベルのデフォルトの OUTPUT 文から。	DD SYSOUT オプションが、対応する OUTPUT オプションより優先されます。

表 4-7 JES2 SYSOUT DD 印刷処理

DD SYSOUT 文の条件	処理順序
JES2 /*JOBPARM 文から。	/*OUTPUT および DD SYSOUT 文に対応するオプションが存在する場合、/*JOBPARM 文のオプションより優先されます。
DD 文が、SYSOUT パラメータの 3 番目の定位置サブパラメータを使って JES /*OUTPUT 文を参照する場合、DD SYSOUT 文と JES2 /*OUTPUT 文から。	JES2 /*OUTPUT 文と JCL //OUTPUT 文の組み合わせは、現在のリリースではサポートされていません。

JCL 印刷オプションの変換

JES2 の /*JOBPARM 文、/*OUTPUT 文、および OUTPUT JCL 文は、SETPRINT マクロに変換されます。JOB および DD SYSOUT 文にある印刷オプションも、SETPRINT マクロのパラメータに変換されます。これらの文の処理オプションの有効範囲が異なるので、生成される SETPRINT マクロの有効範囲は次の値の 1 つを使用して指定する必要があります。

表 4-8 SETPRINT 有効範囲指定

SETPRINT 有効範囲指定	説明
scope='STEP_DEFAULT'	変換された OUTPUT 文。DEFAULT=YES パラメータを含むジョブステップにあります。処理オプションは、OUTPUT パラメータを指定していない、同じステップの DD SYSOUT 文でのみ有効です。
scope='JOB_DEFAULT'	変換された OUTPUT 文。ジョブの JOB 文と最初の EXEC 文の間にあって、DEFAULT=YES パラメータを含みます。処理オプションは、OUTPUT パラメータを指定しないで、デフォルトステップレベルの OUTPUT 文を含まないジョブにある DD SYSOUT 文でのみ有効です。
scope='STEP'	変換された OUTPUT 文。DEFAULT=YES パラメータを含まないジョブステップにあります。処理オプションは、名前を参照する OUTPUT パラメータを含む、同じステップ内の DD SYSOUT 文でのみ有効です。
scope='JOB'	変換された OUTPUT 文。ジョブの JOB 文と最初の EXEC 文の間にあり、DEFAULT=YES パラメータを含みません。処理オプションは、OUTPUT 文を参照する OUTPUT パラメータ値を含む、ジョブ内のすべての DD SYSOUT 文で有効です。scope='JOB' は、JES2 の /*JOBPARM 文と /*OUTPUT 文、および JOB JCL 文から印刷オプション用に生成された SETPRINT マクロでも指定されます。

SETPRINT マクロとその構文の詳細は、『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。

SETPRINT マクロで印刷出力先ファイルに指定した印刷属性は、ジョブの実行時に環境変数として表されます。これらの環境変数は、EXECPGM マクロでステップ完了時に呼び出される post_exec_pgm というスクリプトで使用でき、このスクリプトでステップ完了時に印刷出力を処理できます。このスクリプトを変更すると、すべての使用可能な印刷属性の環境変数を確認してから、外部スプールインタフェースに必要なオプションに変換できます。

次の例では、アプリケーションプログラム PRNTREPT で、REPORT1 というレポートを作成しています。このファイルは、印刷出力として定義されます。BURST=YES および CLASS=A の処理オプションを使って、2 回出力する必要があります。印刷オプションは、OUTPUT JCL 文に指定します。このオプションは、OUTPUT パラメータ値 *.OUT1 で REPORT1 を定義する DD SYSOUT 文に適用されます。DD SYSOUT 文とステップレベルの OUTPUT JCL 文は次のようになります。

```
...
//STEP002 EXEC PGM=PRNTREPT
//OUT1 OUTPUT BURST=YES,CLASS=A
//REPORT1 DD SYSOUT=(,,,),COPIES='2',OUTPUT=(*.OUT1)
...
```

上記の文は次のように変換されます。

```
...
SETPRINT printid='OUT1' scope='STEP' burst='YES' class='A'
SETPRINT ddname='REPORT1' copies='2'
ASSGNDD ddname='REPORT1' type='SYSOUT' printid='OUT1'
EXECPGM pgmname='PRNTREPT' stepname='STEP002'
...
```

ASSGNDD マクロの printid パラメータで、OUTPUT 文の名前 'OUT1' を指定します。この値は、printid='OUT1' で定義した SETPRINT マクロの 'class' 値を使用するよう ASSGNDD に指示します。

第5章

VSE JCL の変換

dostrans は、VSE JCL トランスレータです。この章では、dostrans で File_Map にエントリを生成する方法と、dostrans で VSE JCL ジョブストリームを Sun MBM マクロに変換する方法を説明します。また、VSE JCL 文の変換例も示します。この章の内容は、次のとおりです。

サポートしているそれぞれの VSE JCL 文のリストは、『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。

VSE JCL の変換の準備

この節では、VSE JCL ジョブストリームを変換する前に実行しなければならないタスクを説明します。ノードをインストールし、サブシステムを作成してあることを前提とします。

ディレクトリ構造の作成

dostrans トランスレータでは、実行するためのディレクトリ構造が必要です。この構造は、最低でも次の表に一覧表示されたサブディレクトリを収めた、親ディレクトリから構成されます。

表 5-1　dostrans ディレクトリ

ディレクトリ	内容
jdos	JCL ジョブファイル
dosp	JCL 手続きファイル
ish	変換されたジョブマクロファイル
sli	ジョブまたは手続き用のソースライブラリ組み込み (SLI) ファイル

詳細は、図 5-1 を参照してください。

▼ ディレクトリ構造を作成する

1. ジョブファイルの親ディレクトリを作成します。次に例を示します。

```
$ mkdir -p /user1/batch
```

2. 親ディレクトリに移動します。

```
$ cd /user1/batch
```

3. 入力 JCL ジョブとソースライブラリ組み込みファイル用に jdos ディレクトリおよび sli ディレクトリ、変換されたジョブマクロファイル用に ish ディレクトリを作成します。

```
$ mkdir jdos sli ish
```

4. 変換する VSE JCL ジョブファイルのコピーを jdos ディレクトリに配置します。

5. **入力 JCL 手続きファイル用に** dosp **ディレクトリ、変換された手続きマクロファイル用に** ishp **ディレクトリを作成します。**

```
$ mkdir dosp ishp
```

変換された手続きファイルが書き込まれるディレクトリは、サブシステムの作成時に定義されます。このディレクトリは多くの場合、$PROCLIB または ishp ディレクトリですが、他のディレクトリでもかまいません。

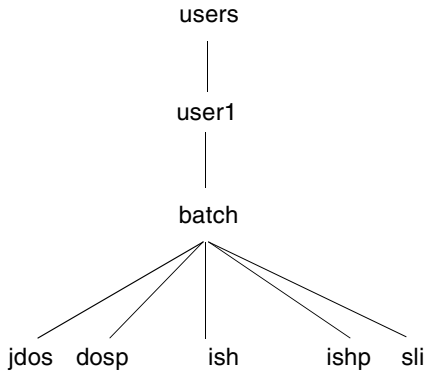
6. **変換する VSE JCL 手続きファイルのコピーを** dosp **ディレクトリに配置します。**

7. **ソースライブラリ組み込み (SLI) ファイルのコピーを** sli **ディレクトリ、または** SLIDIR **環境変数で定義されたディレクトリに配置します。**

これらは、SLI ジョブエントリ文を使って、ジョブと手続きにより取り込まれるファイルです。

次の図に、dostrans を実行するディレクトリを示します。

```
HOME=/users/user1
PROCLIB=$HOME/batch/ishp
SLIDIR=$HOME/batch/sli
```




```
dostrans を実行するには：
. $EBMHOME/batchenv subsys1
cd $HOME/batch
dostrans ...
```

図 5-1 dostrans のディレクトリ構造

FMROOT 環境変数の設定

FMROOT 環境変数には、File_Map にエントリを作成するときに dostrans で使用するデフォルトの基本ディレクトリ名を記述します。サブシステムのユーザー設定ファイル ($USER_SETUP) に手動で $FMROOT を設定する必要があります。$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。

例

$FMROOT を設定しない場合、dostrans では、File_Map に次のようにエントリを生成します。

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

$FMROOT を /user/dir1 に設定した場合、各データセットのエントリは次のようになります。

```
dataset-name;MASTERCAT;FS;/user/dir1/dataset-name;;0;
```

$FMROOT を /user/{P}/dir1 の可変引数に設定した場合、dostrans では各データセットのエントリを次のように生成します。

```
dataset-name;MASTERCAT;FS;/user/${P}/dir1/dataset-name;;0;
```

通常、P は、$HOME/.btshrc ファイルまたはその他の Sun MBM 設定ファイルに設定されているジョブや環境変数から手続きに渡されるパラメータです。実行時に、バッチスクリプトの ${P} が設定値から解釈されます。

実行時に評価される別の環境変数に $FMROOT を設定している場合は、$USER_SETUP ファイルに次の構文を記述します。

```
setenv FMROOT "\\\${ENVVAR}"
```

ここで *ENVVAR* は、$FMROOT に割り当てられた値です。

TRANSOPTS 環境変数の設定

TRANSOPTS 環境変数を使用して、トランスレータが起動されるたびに適用されるトランスレータオプションを設定できます。たとえば、変換ごとにバッチジョブからの日付スタンプとバージョン情報を省略させるには、-n トランスレータオプションを指定します。たとえば、Korn シェルの設定ファイルで TRANSOPTS を設定するには、次のように入力します。

```
TRANSOPTS='-n'; export TRANSOPTS
```

$USER_SETUP ファイルまたはサブシステムに関連したその他の設定ファイルの 1 つで $TRANSOPTS を設定すると、トランスレータオプションが自動的に使用されます。トランスレータを実行する前にコマンドプロンプトで $TRANSOPTS を設定すると、オプションはそのセッションのみで有効になります。

注意 – 設定するオプションを選択するときには注意してください。$TRANSOPTS で -v トランスレータオプションを設定しないでください。設定すると、変換のたびに File_Map エントリが再度作成されます。

File_Map エントリの作成

検査モード (-v オプション) で dostrans を実行するとき、メインフレームデータセット名とライブラリ名は、移行環境の File_Map のファイル名とディレクトリパス名に割り当てられます。サブシステムを作成すると、File_Map ファイルの位置を指定する FILEMAP 環境変数が設定されます。

検査モードで dostrans を実行すると、File_Map にエントリが作成されますが、出力 Sun MBM スクリプトは作成されません。この規則は、VSE JCL ストリームから File_Map にエントリを作成する場合に適用されます。

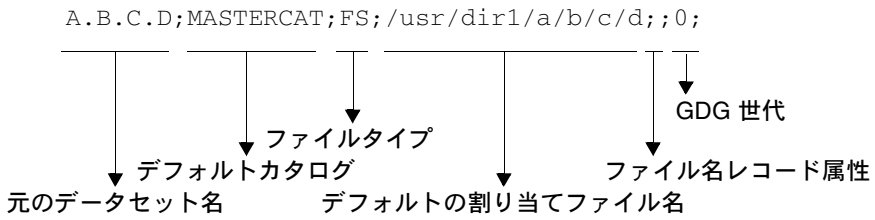
1. 認定されたデータセット

   File_Map に作成されるエントリの前には $FMROOT の値が指定されます。

   下記 JCL 文で $FMROOT を /usr/dir1 に設定した場合

```
// DLBL INFILE,'A.B.C.D'
```

次の File_Map エントリが作成されます。



デフォルトのファイルタイプは FS (Sun MTP VSAM ファイル以外のファイル) です。データセットが VSAM の場合は、ファイルタイプを VS に変更し、ファイルのフルパス名を Sun MTP 領域のファイル制御テーブル (FCT) で定義したデータセットの VSAM 名 (1 ～ 8 文字) に置換する必要があります。値がある場合、「Filename Record Attributes」フィールドの値は、レコード属性を収めたファイルの名前です。

File_Map を変更するには、BAM ユーティリティー (『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照)、Tool Kit (23 ページの「Tool Kit を使った File_Map の更新」を参照)、または cfm コマンド (24 ページの「cfm コマンドを使った File_Map の更新」を参照) を使用します。次に例を示します。

```
A.B.C.D;MASTERCAT;VS;KSDS1;;0;
```

2. &A 形式の記号パラメータを含むデータセット

File_Map のエントリは、$FMROOT の値とデータセット名の変数に基づいて作成されます。次に例を示します。

下記 JCL 文で $FMROOT の値を /usr/dir1 に設定した場合

```
// DLBL INFILE,'&A..B.&C..D'
```

次のデフォルトの File_Map エントリが作成されます。

```
&A..B.&C..D;MASTERCAT;FS;/usr/dir1/${A}/b/${C}/d;;0;
```

記号パラメータは変数に変換されます。

注 – &C.D として定義されるデータセットは、1 つのディレクトリとして処理されます。たとえば、.../${c}d が File_Map エントリに生成されます。

---

3. ...D&E 形式で指定される変数を含むデータセット

   File_Map のエントリは、$FMROOT の値とデータセット名の変数に基づいて作成されます。次に例を示します。

   下記 JCL 文で $FMROOT を /usr/dir1 に設定した場合

```
// DLBL INFILE,'&A.&B.C.D&E'
```

   次の File_Map エントリが作成されます。

```
&A.&B.C.D&E;MASTERCAT;FS;/usr/dir1/${A}/${B}/c/d${E};;0;
```

4. ライブラリエントリ

   File_Map のライブラリエントリは、__LIB という文字で File_Map エントリ内の 2 番目のフィールドに指定されます。このエントリは、LIBDEF JCL 文の SEARCH オペランドの値に基づいて生成されます。次に例を示します。

   下記 JCL 文で $FMROOT の値を /usr/dir1 に設定した場合

```
// LIBDEF PHASE,SEARCH=ALLLIB1.SUBLIB1
```

   次の File_Map エントリが作成されます。

```
ALLLIB1.SUBLIB1;__LIB;FS;/usr/dir1/alllib1/sublib1;;0;
```

dostrans トランスレータの使用

トランスレータへの入力は、VSE JCL ジョブまたは手続きです。そして、実行可能マクロファイルを出力します。ジョブマクロファイルは、サブシステムに対してサブミットされ実行されます。ジョブマクロファイルを使って、変換された手続きを呼び出します。Sun MBM マクロの詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

dostrans を使った VSE JCL の変換は、次の 2 段階の処理で行われます。

- dostrans を検査モード (-v オプション) で実行して、File_Map を作成します。VSAM データセットまたは世代データグループ (GDG) を使用する場合は、dostrans を通常モードで実行する前に File_Map を更新する必要があります。
- dostrans を通常モードで実行して、マクロスクリプトを作成します。

図 5-1 に、dostrans を実行するディレクトリを示します。

▼ VSE JCL ジョブと手続きを変換する

1. **サブシステムの環境を設定する方法は、次のとおりです。**
   - ソース `batchenv` でサブシステム名を指定します。次に例を示します。

```
$ . $EBMHOME/batchenv cblsys
```

   - 次の手順に従います。

   a. Sun MBM メインメニューを表示します。

   b. メインメニューの「Command Prompt」アイコンをクリックし、サブシステム名を入力します。

2. **まだサブシステムのユーザー設定ファイルに `$FMROOT` を設定していない場合は設定します。**

3. **必要に応じて、`$TRANSOPTS` を設定します。**

4. **元の VSE JCL ジョブと手続きを含む `jdos` の親ディレクトリと `dosp` ディレクトリを変更します。**

5. **すべてのジョブと手続きファイルに対して検査モード (ジョブの場合は `-v` オプション、手続きの場合は `-vp` オプション) で `dostrans` を実行し、サブシステム `File_Map` を作成します。**

   詳細は、第 2 章を参照してください。

6. **`File_Map` を確認し、必要な変更を行います。**

7. **すべてのジョブと手続きファイルに `dostrans` をもう一度実行し、ジョブおよび手続きスクリプトを作成します。**

   データセットとライブラリに関する情報は、`dostrans` を使って `File_Map` から取り出します。手続きファイルを変換する場合は `-p` オプションを使用する必要があります。

   元の JCL または作成したバッチジョブを使って、ジョブを管理できます。

   - 元の JCL を維持する場合、変更を行ったらもう一度変換する必要があります。
   - 作成したマクロスクリプトを維持する場合は、トランスレータを再度使用する必要はありません。

   `dostrans`、Sun MBM マクロ、およびジョブのサブミット方法についての詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

VSE JCL 文のサポート

VSE JCL 文の多くはメインフレーム固有であり、Sun MBM 環境には適用されません。これらの文はトランスレータによって変換されず、変換プロセス時に警告が発せられます。そのメッセージについては、『Sun Mainframe Batch Manager ソフトウェア メッセージガイド』を参照してください。

サポートされる文はすべて条件文で、145 ページの「dostrans の制限事項」に説明する制限事項に基づいています。JCL と ジョブエントリ制御言語 (JECL) のサポートについての詳細は、『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。

dostrans の制限事項

継続文字

dostrans では、継続文字によって中断された単一オペランドはサポートされません。次の例の JCL には、複数行が含まれています。

```
//DLBL SORTOUT,'FSSRTOUT',RECCSIZE=50,RECORDS=(100,1000),DISP=*
       (NEW,KEEP,DELETE)
```

上記の文は次のように変換されます。

```
//DLBL SORTOUT,'FSSRTOUT',RECCSIZE=50,RECORDS=(100,1000), *
       DISP=(NEW,KEEP,DELETE)
```

印刷不可能な文字

parm 文字列内の印刷不可能な文字はすべて /x*HH* 形式に変換されます。ここで、*HH* は、文字の 16 進数値です。

VSE ユーティリティー

VSE LIBR および DITTO ユーティリティーは、印刷、ファイルのコピーなどの機能を実行し、メインフレームでのライブラリ管理を提供します。現在のリリースではサポートされません。

VSE JCL 変換の例

この節では、dostrans で JCL 文を Sun MBM マクロに変換する例を示します。

/. (ラベル文)

/. (ラベル) 文は、GOTO または ON 文によって参照されるターゲットラベルを定義します。

コード例 5-1　　JCL ストリームでの /. ラベル文

```
...
/. /. STARTPRG BEGIN PURCHASES REPORT GENERATION
...
```

上記の文は次のように変換されます。

```
...
#######################################################
LABEL name=STARTPRG
#######################################################
...
```

dostrans は、ラベル文でのユーザーコメントを破棄します。

/+ (手続きの終了)

/+ 文は、手続きの終わりをマークします。

```
...
/+ END OF PROCEDURE TESTPROC
```

上記の文は次のように変換されます。

```
...
ENDPROC
```

dostrans は、ユーザーコメントを破棄します。

/* (インストリームデータの終了)

/* 文は、EXEC 文によって実行されるプログラムに入力として提供される、インストリームファイルのあとにくることが期待されます。次の JCL セグメントには、プログラム RUNTRANS への入力として定義されたインストリームデータの行が 1 行あります。

```
...
//EXEC  PGM=RUNTRANS
TRAN01 X1001002
/*
...
```

上記の文は次のように変換されます。

```
...
ASSGNDD ddname='SYSIN' type='INSTREAM' << !
TRAN01 X1001002
!
EXECPGM pgmname='RUNTRANS' stepname='STEP0001'
...
```

現在のステップに、2 つの一時ファイルが作成されます。./SYSIN.${JON} という名前のファイルは、SYSIN に割り当てられます。これによって、プログラム RUNTRANS で入力用に開くことができます。RUNTRANS プログラムが COBOL ACCEPT 文を使用して内容にアクセスできるように、./LN_SYSIN.${JON} ファイルが定義されます。

/& (ジョブの終了)

ジョブの終了を指定します。

コード例 5-2　　/& 文

```
...
/& END OF JOB TESTJOB1
```

上記の文は次のように変換されます。

```
...
ENDJOB
```

ユーザーコメントは、dostrans によって破棄されます。

* (コメント) 文

通常の VSE JCL コメント (列 1 と 2 のアスタリスクと空白) は、VSE JCL の LOG または NOLOG の設定に基づいて、Sun MBM の LOGMSG または DISPLAY マクロ文に変換されます。

コード例 5-3　　コメント文 (*)

```
...
*
* INITIATE MONTHLY REPORT GENERATION
*
...
```

上記の文は次のように変換されます。

```
...
#LOGMSG
#LOGMSG INITIATE MONTHLY REPORT GENERATION
#LOGMSG
```

また、JCL コメントには、EBMSYSCMD マクロ文を生成して、指定された *system command* を実行する特殊な形式もあります。また、『Sun Mainframe Batch Manager ソフトウェア リファレンスマニュアル』も参照してください。

*　EBMSYSCMD *system-command*

生成されたマクロコマンドは RETURN-CODE と CONDITION-CODE 処理ロジックにより実行が制御されます。たとえば、次の文があるとします。

```
* EBMSYSCMD echo "Prepare output bin ABCD for job PREOP1"
```

上記の文は次のように変換されます。

```
EBMSYSCMD << !
echo "Prepare output bin ABCD for job PREOP1"
!
```

ASSGN 文

ほとんどのシステム論理ユニットでは、VSE での機能を提供していますが、これらは Sun MBM 環境には適用されません。したがって、これらの機能は無視され、変換されません。SYSRDR、SYSIPT、および SYSIN は、一時ファイルへの割り当てによってサポートされます。SYSLST、SYSPCH、および SYSOUT と、プリンタデバイス (標準の VSE スプーラ POWER によって制御されるデバイス) に割り当てられるすべてのプログラマ論理ユニットがサポートされます。

Sun MBM 環境でプリンタデバイスに割り当てられるユニットを指定するには、Sun MBM POWER 論理ユニット構成ファイル $PUBLIC/vse/power/lu.cfg にシステム論理ユニットまたはプログラマ論理ユニットを定義する必要があります。また、Sun MBM POWER 物理デバイス構成ファイル $PUBLIC/vse/power/pu.cfg にデフォルト属性のデバイスアドレスを定義する必要があります。

注 – JCL ジョブと手続きを変換する前に、これらのファイルを構成しなければいけません。

ASSGN 操作の機能では、論理名を特定のハードウェアデバイスに割り当てるので、この文のほとんどのオプションは無視されます。これは、Sun MBM 環境では、ディスク装置とテープがファイルに置き換わっているためです。したがって、Sun MBM 環境で適用可能なオペランドは、プリンタデバイスに関連付けられています。オペランドには、SYS*xxx*、*dev-addr*、UA、IGN、SYS*yyy* などがあります。

これらのオペランドは、*dev-addr* が Sun MBM POWER 構成ファイルで定義されたデバイスである場合に限って適用可能です。Sun MBM での POWER 機能のサポートの詳細は、172 ページの「POWER」を参照してください。

次の VSE JCL の例では、INVUPDT という名前の COBOL プログラムの実行時に使用するために、VSE 環境のファイルに論理名を割り当てます。

コード例 5-4 ASSGN 文

```
...
// ASSGN SYS009,190
// DLBL INVMAST,'INVENTORY.MASTER'
// EXTENT SYS009,,,,,45000,5000
// EXTENT SYS010,,,,,40000,8000
// EXEC  INVUPDT
...
```

- アドレス 190 に検出された物理デバイス (ディスクドライブ) に論理ユニット SYS009 を関連付けます
- 論理名 INVMAST をファイル INVENTORY.MASTER に割り当てます
- EXTENT 文で、論理名 SYS009 に関連付けられたデバイス (この例では 190) に INVENTORY.MASTER ファイルがあり、ファイルは 5000 ブロック/トラックを使用してブロック/トラック 45000 で始まり、SYS010 に複数のエクステントがあることを指定します

INVUPDT COBOL プログラムを実行して、次の文を含む INVENTORY.MASTER ファイルにアクセスします。

```
SELECT INFILE ASSIGN TO INVMAST.
...
FD INFILE...
...
OPEN INFILE
...
READ INFILE INTO REC-AREA.
```

JCL は次のように変換されます。

```
...
ASSGNDD ddname='SYS009' type='DUMMY'
ASSGNDD ddname='INVMAST' datasetname='INVENTORY.MASTER'
       type='FS' filename='/tmp/inventory/master'
EXECPGM pgmname='INVUPTD' stepname='STEP00002'
...
```

- 最初の ASSGNDD マクロは、JCL ASSGN 文の変換結果です。論理ユニット名 SYS009 は、Sun MBM POWER 論理ユニットファイル $PUBLIC/vse/power/lu.cfg で出力デバイスとして定義されていない場合、ダミーファイルに割り当てられます。たとえば、INVUPTD プログラムが SYS009 で入出力を試みると、READ に対して EOF 条件が発生し、WRITE に出力は生成されません。

2 番目の ASSGNDD マクロは、JCL DLBL 文の変換結果です。dostrans は、*type* および *file-name* のパラメータ値を File_Map から抽出します。

- EXEC JCL 文は、EXECPGM マクロ呼び出しに変換されます。このマクロ呼び出しは、デフォルトのプログラム検索パスを使って INVUPTD プログラムを検索します。stepname パラメータは、これが元の JCL ジョブストリームでの 2 番目の EXEC 文であることを示します。

EXTENT 文の変換は実行されません。INVENTORY.MASTER ファイルの位置は、ASSGNDD マクロ呼び出しの filename パラメータで指定され、EXTENT 文の変換は不要になります。

デバイスアドレス (この例では 190) が $PUBLIC/vse/power/pu.cfg で Sun MBM POWER 出力デバイスとして構成されている場合、次のようになります。

```
#################################################################
#PU name                                                        #
#################################################################
190
...
```

ASSGN 文は、次のように変換されます。

```
ASSGNDD ddname='SYS009' type='SYSOUT'
```

これで、INVUPDT COBOL プログラムは印刷をスケジューリングされている出力先ファイルに書き込み可能になります。

SELECT 文は、次のようにコーディングされます。

```
SELECT OUTFILE ASSIGN TO SYS009
```

ASSGN 文が論理ユニット名であるデバイスアドレスを参照する場合、次のようになります。

```
// ASSGN SYS009,SYS011
```

また、SYS011 は、$PUBLIC/vse/power/lu.cfg で次のように構成されます。

```
############################################################
#LUNAME CLASS_NAME  PRINTER_ID                             #
############################################################
 SYSLST default  DUMMY1
 SYSPCH default  DUMMY2
 SYSOUT default  DUMMY3
 SYS011 default   ljs7
```

dostrans は、この ASSGNDD マクロ呼び出しを ASSGN 文から生成します。:

```
ASSGNDD ddname='SYS009' alias='SYS011'
```

論理ユニット SYS009 は、ジョブの実行時に SYSOUT タイプのファイルとして処理されます。出力先ファイルは、上記の INVUPDT COBOL プログラムの例と同じ方法でアクセスされます。

制限事項

ASSGN は、ジョブ制御文 (JCS) // ASSGN とジョブ制御コマンド (JCC) ASSGN (// なし) のどちらかです。VSE 環境で、すべての ASSGN JCC コマンドは固定的です。これは、新しい固定割り当てまたはシステムの再初期化によって上書きされるまで、割り当てが永続することを意味しています。

Sun MBM POWER 構成ファイルで POWER に関連するデバイスの固定割り当てをすべて定義しなければいけないので、Sun MBM 環境での ASSGN JCC での割り当ては、一時割り当てとして処理されます。

DATE 文

Sun MBM 環境で、DATE JCS は SETDATE マクロ呼び出しに変換されます。このマクロ呼び出しは、Sun MBM ノード日付を、DATE 文と同じジョブで実行されるアプリケーションプログラムに指定した値で上書きします。

BAM を使用すると、システム日付に影響を与えずに Sun MBM ノード日付を変更できます。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。

COBOL および PL/I アプリケーションでは、テストのためにノードの現在の日付を取得できます。ただし、COBOL (システムで利用可能な場合) では、実行時システムの作成時に Sun MBM 日付/時刻設定を選択する必要があります。詳細は、255 ページの「Sun MBM ノードの現在の日付の取得」を参照してください。

コード例 5-5　　DATE 文

```
...
// DATE 12/31/2004
...
```

上記の文は次のように変換されます。

```
SETDATE value=12/31/2004
```

新しい日付の値が SETDATE マクロ呼び出しのあとのジョブで実行されるアプリケーションプログラムに渡されます。

DLBL 文

Sun MBM 環境では、DLBL の機能によって、現在のジョブのみで使用するためにデータセットに論理名が割り当てられます。そのあとのジョブでデータセットへのアクセスが必要になった場合には、各ジョブにデータセットへの個別の割り当てが必要になります。

OPTION 文の USRLABEL、PARSTD、CLASSTD、および STDLABEL オペランドはサポートされません。これらのオペランドが検出されると、dostrans の変換中に警告が表示されます。

DLBL 文のラベル情報の検索順序

現在の Sun MBM のバージョンでは、ASSGNDD マクロを使用するパーティション一時サブ領域でのラベル定義だけをサポートしています。したがって、アプリケーションプログラムでは、ジョブ環境で定義されたラベルを使用している場合に限って、データセットにアクセスできます。

VSAM ファイルの検索規則

Sun MBM 環境では、DLBL 文の CATALOG オペランドがサポートされ、これによってアプリケーションプログラムは、指定されたカタログから VSAM ファイルにアクセスできます。DLBL 文でカタログが指定されていない場合、デフォルトの MASTERCAT カタログが検索されます。

EXEC 文

次の例に、EXEC 文と PGM および PARM オペランドを示します。

コード例 5-6　EXEC 文と PGM および PARM オペランド

```
...
//EXEC  PGM=PROGA,REAL,PARM='PARM WITH SPACE'
...
```

上記の文は次のように変換されます。

```
...
EXECPGM pgmname='PROGA' stepname='STEP0003' parm='PARM_WITH_SPACE'
...
```

PARM オペランド値での空白文字は、dostrans によって下線に変換されます。ただし、プログラムでは、下線から空白に戻された PARM 値を受け取ります。

コード例 5-7　EXEC 文と PROC および記号パラメータオペランド

```
...
//EXEC PROC=PROCB, A=&zzz, B='xzu', YYY=,QQQ,
       G=",F='&&23',&XYZ&ABC
...
```

上記の文は次のように変換されます。

```
...
EXECPROC procname=PROCB stepname='STEP0004' \\\
       parms='A=\${ZZZ},B=XZU,YYY=,QQQ,G=,F=/x2623,\${XYZ}\${ABC}'
...
```

記号パラメータ ZZZ、XYZ、および ABC の変換に注意してください。

注 – Sun MBM では、& のリテラル値はサポートされません。この例では、形式 F='&&23' のパラメータ文字列が F=/x2623 に変換され、dostrans によって警告が表示されます。

コード例 5-8　　インストリームデータを実行プログラムに渡す EXEC 文

```
...
//EXEC  PGM=PROG1
FIRST LINE
SECOND LINE
THIRD LINE
/*
...
```

上記の文は次のように変換されます。

```
...
ASSGNDD ddname='SYSIN' type='INSTREAM' << !
FIRST LINE
SECOND LINE
THIRD LINE
!
EXECPGM pgmname='PROG1' stepname='STEP0005'
...
```

GOTO 文

GOTO 文とターゲットラベルの間のすべての JCL 文は無視されます。次はその例外です。

- SYSLOG から入力された文 (たとえば CANCEL)
- ターゲットラベルの前に // JOB または /& (ジョブの終了) のいずれかが見つかった場合、呼び出されるすべての手続きを含むジョブは終了します。
- ターゲットラベルの前に /+ (手続きの終了) が見つかった場合、残りのジョブはスキップされます。

コード例 5-9　　GOTO 文

```
...
// GOTO LABEL1
....
```

上記の文は次のように変換されます。

```
...
GOTO LABEL1
...
```

コード例 5-10　　宛先を $EOJ として指定する GOTO 文

```
...
// GOTO $EOJ
...
```

上記の文は次のように変換されます。

```
...
GOTO END_JOB
...
```

$EOJ は、バッチシェルラベル END_JOB に変換されます。

IF 文

dostrans は、IF 文を IF/THEN マクロセットに変換します。ここで、条件は IF 部分で表され、処理は THEN 部分で表されます。

コード例 5-11　　IF 文と GOTO 文

```
...
// IF $RC GT 4 THEN
// GOTO $EOJ
....
```

上記の文は次のように変換されます。

```
...
IF LASTRC GT 4
THEN GOTO END_JOB
...
```

$RC は、バッチシェル変数 LASTRC に変換され、$EOJ は特別なバッチシェルラベル END_JOB に変換されます。$MRC が IF 文でテストされる場合、バッチシェル変数 MAXRC に変換されます。これは、前に実行されたすべてのステップによって設定されたもっとも上位の条件コードを表します。

INCLUDE 文

dostrans は INCLUDE 文を検出すると、指定されたファイルからのデータをジョブにインクルードします。詳細は、168 ページの「* $$ SLI 文」を参照してください。

JOB 文

JOB 文は、ジョブの開始を示します。

変換されるファイルの名前を JOB001A とします。

コード例 5-12　　JOB 文とアカウンティング情報

```
// JOB JOB001 ACCT100
....
....
....
/&
```

上記の文は次のように変換されます。

```
#  JOB:  JOB001A          Translated :Fri May 6 16:56:53 2005 #
#                                                              #
################################################################
BEGINJOB
...
...
...
ENDJOB
```

ファイルの名前によって、変換出力のコメントでのジョブを識別します。アカウンティング情報のオペランドは無視されます。

LIBDEF 文

Sun MBM 環境での検索パスは、EXECPGM マクロの *pgmname* パラメータで指定したアプリケーションプログラムまたはユーティリティーと、EXECPROC マクロの *procname* パラメータで指定した変換後の手続きに適用されます。手続きとアプリケーションプログラムのデフォルトの検索パスは、サブシステム作成時に割り当てられます。

LIBDEF *member-types* PHASE、PROC、および * は、LIBDEF 文の SEARCH オペランド値で使用される場合、LIBDEF マクロ呼び出しに変換されます。実行時に、LIBDEF は 1 つまたはすべての検索パスを更新します。

コード例 5-13　　LIBDEF 文

```
//LIBDEF PHASE,SEARCH=(PHASELIB.SUBLIB1, PHASELIB.SUBLIB2),
       CATALOG=SYSLIB.SUBLIB,PERM
```

dostrans でバッチシェルスクリプトを生成する前に、ジョブファイルに対して検査モード (-v オプション) で実行して、データセットとライブラリをデフォルトの File_Map エントリに抽出する必要があります。これらのエントリを編集して、データセットとライブラリのファイルとディレクトリのパスを割り当てます。この例で、File_Map には次のものが含まれます。

```
...
PHASELIB.SUBLIB1:__LIB;FS;/progs1;;0;
PHASELIB.SUBLIB2;__LIB;FS;/progs2;;0;
...
```

dostrans は LIBDEF の変換時にこれらの File_Map エントリを参照します。

```
...
LIBDEF type=PGM lib=/progs1:/progs2
....
```

(type=PGM パラメータのため) LIBDEF マクロの実行時には、lib パラメータで指定されたディレクトリが最初に検索されます。元の LIBDEF JCL 文の CATALOG および PERM オペランドは無視されます。

LIBDROP 文

LIBDROP 文は、JCL ジョブで発行され、前の LIBDEF 文で指定された手続き検索パスを無視します。

コード例 5-14　LIBDROP 文

```
...
//LIBDROP PROC,SEARCH,CATALOG,PERM
...
```

上記の文は次のように変換されます。

```
...
LIBDROP type=PROC
...
```

(*member-type* が PROC であるため) LIBDROP マクロは、前の LIBDEF マクロ呼び出しによって手続き検索パスに追加されたディレクトリを削除します。

LIBLIST 文

LIBLIST 文は、LIBDEF 文によるライブラリ定義を SYSLOG に表示します。

コード例 5-15　LIBLIST 文

```
...
//LIBLIST *,SYSLOG
...
```

上記の文は次のように変換されます。

```
...
LIBLIST type=ALL
...
```

LIBLIST マクロは、LIBDEF マクロによって定義された手続きおよびプログラムの検索パスを表示します。LIBLIST 文での SOURCE、OBJ、または DUMP の member-type オペランド値は変換されません。この場合、dostrans は文を無視します。この変換によって、LIBLIST マクロは、アプリケーションプログラムおよび手続きの検索パスを、実行時に Sun MBM ジョブの履歴ファイルに書き込みます。

LOG 文

LOG 文は、ジョブ制御文の記録を開始します。

コード例 5-16　LOG JCS 文および LOG JCC 文の形式

```
...
// LOG ACCOUNTING RECORDS UPDATE INITIATED
...
LOG END OF ACCOUNTS PROCESSING
...
```

dostrans によってコメント文に変換されます。

```
LOGMSG '// LOG issued'
LOGMSG 'ACCOUNTING RECORDS UPDATE INITIATED'
...
LOGMSG 'LOG issued'
LOGMSG 'END OF ACCOUNTS PROCESSING'
...
```

LOGMSG マクロ文は、Sun MBM ジョブ履歴ファイルにコメントを書き込みます。DISPLAY マクロ文は、ジョブ履歴ファイルと、オプションとして定義された Sun MBM コンソールにコメントを書き込みます。『Sun Mainframe Batch Manager ソフトウェア リファレンスマニュアル』の LOGMSG および DISPLAY マクロの節を参照してください。

NOLOG 文

NOLOG 文は、ほとんどのジョブ制御文の記録を停止します。

コード例 5-17　NOLOG JCS 文および NOLOG JCC 文の形式

```
...
// NOLOG BACKUPS IN PROGRESS
...
NOLOG BACKUPS COMPLETE
...
```

dostrans によってコメント文に変換されます。

```
...
LOGMSG '// NOLOG issued'
#LOGMSG 'BACKUPS IN PROGRESS'
...
LOGMSG 'NOLOG issued'
#LOGMSG 'BACKUPS COMPLETE'
...
```

NOLOG 操作で入力されたユーザーコメントは、トランスレータの出力に保持されます。

ON 文

$ABEND 形式と $RC 形式の 2 つの条件のある ON 文は、演算子の値が OR の場合、ONCONDCODE と ONRETCODE という 2 つの別個のマクロ呼び出しに変換されます。

ON 文に AND 演算子で接続された 2 つの条件があり、条件の 1 つが $ABEND 形式である場合、ON は ONCONDCODE マクロに変換されます。$ABEND 条件は、他のコード条件に優先されます。

コード例 5-18　　$RC を含む条件のある ON 文の 2 つの形式

```
...
//ON $RC > 0 AND $RC < 8 CONTINUE
...
//ON $RC > 8 OR $ABEND GOTO ABEXIT3
...
/. /. ABEXIT3
...
```

上記の文は次のように変換されます。

```
...
ONRETCODE GT 0 AND LT 8 CONTINUE
...
ONRETCODE GT 8 GOTO ABEXIT3
ONCONDCODE NE 0 GOTO ABEXIT3
...

#########################################################
LABEL name='ABEXIT3'
#########################################################
```

OR 演算子で結合された $ABEND 条件と $RC 条件のある 1 つの ON 文では、ONCONDCODE マクロと ONRETCODE マクロが生成されます。

PAUSE 文

PAUSE 文は、現在のジョブの処理を一時的に停止します。dostrans は、PAUSE 文があると、次の文字を下線 (_) に変換します。

```
' ( ) ! & * $ ^ % @
```

次の例では、PAUSE 文はプリンタの稼動状況を要求するメッセージを SYSLOG に送信します。

コード例 5-19 PAUSE 文

```
//PAUSE ENSURE PRINTER IS ONLINE AND IDLE
...
```

上記の文は次のように変換されます。

```
...
PAUSE msg='    ENSURE PRINTER IS ONLINE AND IDLE'
...
```

PAUSE マクロは、msg パラメータのテキストを、実行時に Sun MBM コンソールと Sun MBM ジョブ履歴ファイルに送信します。そのあと、ジョブは中断されます。ユーザーが中断されたジョブのジョブ番号を指定して rsmjob コマンドを発行した場合に限って、ジョブが再開されます。rsmjob コマンドの詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。PAUSE マクロの詳細は、『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。

PROC 文

PROC 文は、手続きの開始を示し、手続きで使用されるパラメータのデフォルト値を提供します。次の例では、PROC 文は EXECTST1 という手続きファイル内にあります。

コード例 5-20　PROC 文

```
...
// PROC XXX=123,yy=,Z='A&&B',SP='A#  B',
...
```

上記の文は次のように変換されます。

```
BEGINPROC procname='EXECTST1'
parms='XXX=123,YY=,Z=A/x26B,SP=A\\#__B,F='
...
```

JCL で 1 つの '&' を表す '&&' は、/x*HH* に変換されます。ここで *HH* は、文字の 16 進数値です。SP 記号パラメータでの空白は、下線に変換されますが、アプリケーションプログラムがデータにアクセスするときに空白に戻されます。Sun MBMでは、パラメータ値でのアンパサンドをサポートしていません。

SET 文

SET 文は、プログラムを実行するためのプロセス制御を設定します。次に、SET JCC とサポートされているオペランド DATE および UPSI と、サポートされていないオペランド LINECT および RF の例を示します。

コード例 5-21　SET 文

```
...
SET DATE=09/12/2002,LINECT=80,RF=CREATE,UPSI=1011
...
```

上記の文は次のように変換されます。

```
...
SETDATE  value=09/12/2002
SETPGMSW value=1011XXXXX
...
```

RF および LINECT オペランドは、dostrans では無視されます。

SETPARM 文

SETPARM 文を使用して、記号パラメータを定義して値を割り当てます。次の例は、2 つの SETPARM 操作 SETPARM JCS と SETPARM JCC を示しています。

コード例 5-22 SETPARM 文

```
...
// SETPARM XX='',YYY=,ZZ='A A#A&&A',AA=&QQ, *
       LIT=' &SYM1 '
// SETPARM RCODE=$RC,MCODE=$MRC
...
```

上記の文は次のように変換されます。

```
...
SETPARM parms='XX=,YYY=,ZZ=A_A\\#A/x26A,AA=\${QQ},LIT=_\${SYM1}_'
SETPARM parms='RCODE=LASTRC,MCODE=MAXRC'
...
```

変換について次の点に注意してください。

- 記号パラメータは、&QQ から \${QQ} に変換されます
- # と && は、それぞれ対応する 16 進数に変換されます
- リテラル値での空白は、下線に変換されますが、実行時に空白に復元されます
- $RC および $MRC は、それぞれ Sun MBM の変数 LASTRC および MAXRC に変換されます

TLBL 文

TLBL 文には、テープファイルラベル情報が収められます。Sun MBM 環境では、VSE テープファイルはディスクファイルとして表されます。次の JCL セグメントには 2 つの TLBL 文があります。最初の文は、FILEA という新しいファイルのラベル情報を定義します。このファイルは、100 日間保持しなければいけません。2 番目の文は、FILEB という既存のファイルのラベル情報を定義します。ソフトウェアの現在のバージョンでは、テープファイルをサポートしていません。

コード例 5-23 TLBL 文

```
...
//TLBL FILEA,'AA.BB.TEST',100,112233,0191,,,,DISP=NEW
//TLBL FILEB,'AA.BB.PROD',,88919,88,0251,0002, *
       04,DISP=OLD
...
```

dostrans を検査モード (-v オプション) で実行すると、次の File_Map エントリが生成されます。

```
...
AA.BB.TEST;MASTERCAT;FS;/data/aa/bb/test;;0;
AA.BB.PROD;Mastercat;FS;/data/aa/bb/prod;;0;
...
```

TLBL 文に対する最終的な dostrans の変換出力は、次のとおりです。

```
...
ASSGNDD ddname='FILEA' datasetname='AA.BB.TEST' \\\
       type='FS' filename='/data/aa/bb/test' \\\
       disp='i-o' normal='K' abend='K'
ASSGNDD ddname='FILEB' datasetname='AA.BB.PROD' \\\
       type='FS' filename='/data/aa/bb/prod' \\\
       disp='i-o' normal='k' abend='k'
...
```

filename、file-id、および DISP 以外のすべてのオペランドは dostrans では無視されます。

UPSI 文

この文は、あとのプログラムでテストできる 1 バイトのスイッチセットを設定します。次に、UPSI バイトのビット 3 と ビット 4 をオンに、ビット 1 とビット 6 をオフに設定し、残りのビットは既存の値を維持する UPSI 文の例を示します。

コード例 5-24　　UPSI 文

```
...
// UPSI 0x11x0
...
```

上記の文は次のように変換されます。

```
...
SETPGMSW 0X11X0XXX
...
```

SETPGMSW マクロ呼び出しで指定された値で、9 ビットが表されます。これは、Server Express UPSI 値のサイズを格納します。実行時スイッチについての詳細は、Server Express のマニュアルを参照してください。

ジョブエントリ制御言語の変換例

VSE/POWER ジョブのジョブエントリ制御言語 (JECL) 文は、ジョブ入力文の収集およびスケジューリングを可能にし、プログラム実行からの出力のスプール処理を制御します。この文は、接頭辞 '* $$' で始まります。使用可能な JECL 文の一部が Sun MBM によってサポートされます。この節では、サポートする文の変換例を示します。

* $$ DATA 文

dostrans は、変換処理中、ジョブストリームに SLI メンバーをインクルードすることができます。* $$ DATA 文を使用して、ジョブストリームに読み込むときに SLI メンバーの内容を更新することもできます。

SLI メンバーファイルは、./jdos ジョブファイルディレクトリではなく、別のディレクトリに常駐している必要があります。dostrans では、デフォルトで、jdos と同じレベルの ./sli ディレクトリを検索します。SLIDIR 環境変数が設定されていると、この値で指定したディレクトリだけが検索されます。

コード例 5-25 では、SLI メンバーの MEM1 をインクルードするジョブストリーム SIMPLEJ を示しています。MEM1 からインクルードされる JCL は、UPDATEX という名前の * $$ DATA 文を使って更新されます。

ジョブストリーム SIMPLEJ:

コード例 5-25 * $$ DATA 文

```
// JOB SIMPLEJ
// DLBL INFILE1,'CUSTOMER.DAT'
// DLBL INFILE2.'TRANS.DAT'
* $$ SLI  MEM=MEM1
* $$ DATA UPDATEX
// ASSGN SYS020,FEE
// UPSI  0001XXXX
/*$SLI
// EXEC  PGM=SPROG
/&
```

SLI メンバー MEM1:

```
// DLBL OUTFILE,'OUTFILE.DAT'
// ON $RC > 4 GOTO $EOJ
* $$ DATA UPDATEX
* END OF INCLUDED DATA UPDATEX
```

次に示すのは、dostrans で変換用に編成された結果のジョブストリームです。

```
// JOB SIMPLEJ
// DLBL INFILE1,'CUSTOMER.DAT'
// DLBL INFILE2,'TRANS.DAT'
// DLBL OUTFILE,'OUTFILE.DAT'
// ON $RC > 4 GOTO $EOJ
// ASSGN SYS020,FEE
// UPSI  0001XXXX
* END OF INCLUDED DATA UPDATEX
// EXEC  PGM=SPROG
/&
```

INFILE1、INFILE2、と OUTFILE の File_Map エントリは次のようになります。

```
...
CUSTOMER.DAT;MASTERCAT;FS;/test/customer.dat;;0;
TRANS.DAT;MASTERCAT;FS;/test/trans.dat;;0;
OUTFILE.DAT;MASTERCAT;FS;/test/outfile.dat;;0;
...
```

dostrans からの最終的な変換出力は次のとおりです。

```
BEGINJOB
ASSGNDD ddname='INFILE1' datasetname='CUSTOMER.DAT'
       type='FS' filename='/test/customer.dat'
ASSGNDD ddname='INFILE2' datasetname='TRANS.DAT'
       type='FS' filename='/test/trans.dat'
ASSGNDD ddname='OUTFILE' datasetname='OUTFILE.DAT'
       type='FS' filename='/test/outfile.dat'
ONRETCODE GT 4 GOTO END_JOB
ASSGNDD ddname='SYS020' type='SYSOUT'
SETPGMSW value=0001XXXXX
# END OF INCLUDED DATA UPDATEX
EXECPGM pgmname='SPROG' stepname='STEP0001'
ENDJOB
```

* $$ DATA 文は、Sun MBM 環境でのジョブの実行には影響がありません。

* $$ SLI 文

dostrans では、ジョブストリームへの SLI メンバーのインクルードをサポートしています。更新された文は、SLI メンバーから読み取る JCL 文に適用されます。デフォルトでは、dostrans は、SLI メンバーファイルの ./jdos と同じレベルの ./sli ディレクトリを検索します。SLIDIR 環境変数が設定されていると、この値で指定したディレクトリだけが検索されます。

コード例 5-26 * $$ SLI 文と更新文

```
Job Stream SIMPLEJ2
-------------------
// JOB SIMPLEJ2                                00000100
// DLBL INFILE1,'CUSTOMER.DAT'                 00000110
 1 * $$ SLI MEM=MEMSLI2.PRD,S=SYS1.PRODLIB     00000120
 2 // DLBL OUTFILE,'OUTFILE.DAT'               $SLIR200
 3 // DLBL INFILE2,'TRANS.DAT'                 $SLID210
 4 // ASSGN SYS020,FEE                         $SLIA215
 5 // UPSI 0001XXXX                            $SLIA220
// EXEC PGM=SPROG                              00000180
/& 00000190
Member MEMSLI2.PRD
------------------
// DLBL OUTFILE,'TEST.OUT.DAT'                 00000200
// DLBL INFILE2,'TRANS.DAT'                    00000210
// ON $RC > 4 GOTO $EOJ                        00000220
* END OF INCLUDED DATA UPDATEX                00000230
```

イベントのシーケンスは、次のとおりです。

1. SLI メンバーの MEMSLI2.PRD を * $$ SLI 文で取り出します。つまり、VSE/POWER で MEMSLI2.PRD の読み取りを開始します。
2. DLBL 文で、DLBL 文を MEMSLI2.PRD SLI メンバーのシーケンス番号 200 に置換します。
3. シーケンス番号 210 の DLBL 文が、MEMSLI2.PRD SLI メンバーから削除されます。
4. メンバーに 215 のシーケンス番号が存在しないので、MEMSLI2.PRD SLI メンバーのシーケンス番号 220 の前に ASSGN 文が挿入されます。
5. SLI メンバー MEMSLI2.PRD のシーケンス番号 220 のあとに、UPSI 文が挿入されます。

これは、* $$ SLI と更新文の読み取り後のジョブストリームです。

```
// JOB SIMPLEJ2                    00000100
// DLBL INFILE1,'CUSTOMER.DAT'     00000110
// DLBL OUTFILE,'OUTFILE.DAT'      $SLIR200 <-- Replaced
// ASSGN SYS020,FEE                $SLIA215 <-- Added
// ON $RC > 4 GOTO $EOJ            00000220
// UPSI 0001XXXX                   $SLIA220 <-- Added
* END OF INCLUDED DATA UPDATEX    00000230
// EXEC PGM=SPROG                  00000180
/&                                 00000190
```

サンプルの JOB SIMPLEJ2 では、dostrans により、MEMSLI2.PRD というメンバーファイルが検索されます。

結果のジョブストリームは、次の出力ジョブスクリプトに変換されます。

```
BEGINJOB
ASSGNDD ddname='INFILE1' datasetname='CUSTOMER.DAT' \\\
       type='FS' filename='/test/customer.dat'
ASSGNDD ddname='OUTFILE' datasetname='OUTFILE.DAT' \\\
       type='FS filename='/test/outfile.dat'
ASSGNDD ddname='SYS020' type='SYSOUT'
ONRETCODE GT 4 GOTO END_JOB
SETPGMSW value=0001XXXXX
# END OF INCLUDED DATA UPDATEX
EXECPGM pgmname='SPROG' stepname='STEP0001'
ENDJOB
```

* $$ SLI 文は、Sun MBM 環境でのジョブの実行には影響がありません。

* $$ LST 文

dostrans では、* $$ LST 文を認識し、指定した * $$ LST オプションごとに SETPRINT マクロパラメータを作成します。Sun MBM では、* $$ LST オプションを一部だけサポートしますが、* $$ LST 文に指定した各オプションにより環境変数が作成されます。環境変数を使用すると、ユーザーは、post_exec_pgm.vse スクリプトにより独自の統合とロジックを実行できます。

```
// JOB EXAMPLE1
* $$ LST LST=31E,CLASS=A,DISP=K,RBS=2000
// ASSGN SYS017,31E
// EXEC PROGM1,SIZE=(PROGM1,150K)
/&
```

次のコード例は、前の例に対する dostrans の出力を示しています。

コード例 5-27 * $$ LST 変換

```
BEGINJOB
SETPRINT scope='JOB' printid='31E' class='A' disp='K' rbs='2000'
ASSGNDD ddname='SYS017' printid='31E' type='SYSOUT
EXECPGM pgmname='PROGM1' stepname='STEP0001'
ENDJOB
```

* $$ LST オプションの環境変数

* $$ LST 文で使用可能なすべてのオプションには、対応する環境変数があります。形式は次のとおりです。

ddname_option

説明

ddname	システム論理ユニット。前の例では SYS017 です。
option	* $$ LST オプション。JCL ストリームで指定されます。同じ名前のジョブ実行オプションとの混同を避けるため、次の例外があります。 PWD は LSTPWD になります DISPは LSTDISP になります

* $$ LST および SETPRINT マクロオプション

指定したオプションに従って、* $$ LST 文は、SETPRINT scope='JOB' に変換されます。

LST オプション	SETPRINT マクロ	LST オプション	SETPRINT マクロ	LST オプション	SETPRINT マクロ
CLASS='A'	class='A'	CMPACT='A'	compact='A'	LTAB='A'	ltab='A'
COPY='A'	copies='A'	COPYG=(A,B)	copyg='A,B'	PURGE='A'	purge='A'
DEST=(A,B)	dest='A,B'	FCB='A'	fcb='A'	PWD='A'	pwd='A'
DISP='A'	disp='A'	FLASH=(A,B)	flash='A,B'	RBC='A'	rbc='A'
FNO='A'	forms='A'	MODIFY=(A,B)	modify='A,B'	RBM=(A,B)	rbm='A,B'
LST='A'	printid='A'	PRMODE='A'	prmode='A'	REMOTE='A'	remote='A'
PRI='A'	pri='A'	DFLT='A'	dflt='A'	SYSID='A'	sysid='A'

LST オプション	SETPRINT マクロ	LST オプション	SETPRINT マクロ	LST オプション	SETPRINT マクロ
RBS='A'	rbs='A'	DIST='A'	dist='A'	TADDR=(A,B)	taddr='A,B'
BURST='A'	burst='A'	JNM='A'	jnm='A'	UCS='A'	ucs='A'
CHARS=(A,B)	chars='A,B'	JSEP=(A,B)	jsep='A,B'	USER='A'	user='A'

sysout ファイルの管理

VSE sysout ファイルは、ジョブ実行中に累積され、post_exec_pgm.vse、OW、および printspec.xx スクリプトの組み合わせによりジョブ終了時に印刷または操作できます。すべての sysout ファイルは、SYSOUTDIR 環境変数で指定したディレクトリに配置されます。$SYSOUTDIR ディレクトリの下にサブディレクトリが作成され、sysout ファイルを作成したジョブ名とステップ名を特定します。

sysout ファイルに関する情報は、post_exec_pgm.vse スクリプトを使って、各プログラムまたはユーティリティー実行の最後に累積されます。post_exec_pgm.vse スクリプトは、$PUBLIC/bin ディレクトリに保存するか、$PUBLIC/bin の前に PATH 変数で指定したユーザー定義のディレクトリに保存することもできます。サンプルの post_exec_pgm.vse.LP スクリプトは、$PUBLIC/bin ディレクトリにあります。このスクリプトは、hold のディスポジションなしの場合に、$SYSOUTDIR/$JOBNAME/JOB_$JON/SysoutListPrint のマスターリストに 0 以外の長さの sysout ファイルに関する情報を配置します。

hold のディスポジションありの場合、0 以外の長さのファイルに関する情報は、$SYSOUTDIR/$JOBNAME/JOB_$JON/SysoutListHold のファイルに保存されます。ファイルの各行には、sysout ファイルのパス名、ファイル名、および出力属性が含まれます。

OW スクリプトは、ジョブ終了処理時に実行され、sysout ファイルから SysoutFileList ファイルを検索し、処理します。この処理は、印刷またはその他のファイル操作で、2 ステップのアクティビティーです。OW スクリプトでは、ファイル名と属性を取得し、別のスクリプト printspec.xx に渡します。printspec.xx は $PUBLIC/bin またはユーザー定義ディレクトリにもあり、実際の印刷またはその他の実行コマンドが含まれています。サンプルファイル $PUBLIC/bin/printspec.LP を参照してください。

これらのスクリプトでは、ファイルの一般的な印刷を実行し、宛先やコピー部数を指定できます。これらのスクリプトを呼び出すには、OW スクリプトファイルの注釈行にコメントを入れないようにする必要があります。

次の手続きは、ランドスケープモードを印刷するために特定ファイルのプリンタコントロール文字をプリンタに送信する方法を示しています。

▼ プリンタコントロール文字を送信する

1. **ディレクトリ** `$PUBLIC/prtctl` **を作成します。**
2. **必要なエスケープシーケンスを判定します。**
   詳細は、プリンタのマニュアルを参照してください。
3. **エスケープシーケンスを含む論理的宛先名と同じ名前のファイルを作成します。**
   次の例は、PCL タイプのプリンタ用です。

```
* $$ LST LST=ljs5
```

`$PUBLIC/prtctl/ljs5` には圧縮印刷のために次のコマンドが含まれています。

```
Esc&k2S
```

`Esc` は、エスケープ文字を表します。実際の Esc という文字ではありません。

POWER

POWER は、ジョブのスケジューリング、入出力のスプーリング、およびパーティションの開始と終了を管理する、VSE オペレーティング環境のコンポーネントです。

VSE 環境では、実際の物理プリンタに割り当てられている論理ユニットにプログラムを使って書き込む場合、実行中、そのプログラムはデバイスを排他的に制御します。各 I/O が発行されると、プログラムは待機状態に置かれ、プログラムの実行は遅くなります。このプログラムの実行中、他のプログラムはそのデバイスにアクセスできません。POWER は、定義されたデバイスに対するプログラムの I/O 要求を遮断し、プログラムの実行後にキューに入れて処理するように出力をディスクに転送します。

VSE では、`FEE` および `FEF` などの特定のデバイスアドレスは、POWER デバイスとして定義されます。ただし、システムプログラマはこれらのデバイスを再構成できます。一般に、`SYSPCH`、`SYSOUT`、および `SYSLST` は、これらのデバイスに常時割り当てられます。プログラマ論理ユニットも POWER デバイスに割り当てることができます。たとえば、印刷出力を作成するアプリケーションプログラムには、`SELECT` 文が含まれます。この文は、内部ファイル名を `SYS005` などの論理名に割り当てます。この論理名が POWER デバイスに割り当てられます。プログラムでこのファイルに書き込むとき、出力は遮断され、POWER により処理はキューに入れられます。

常時割り当て論理ユニットには、JCL の `ASSGN` 文は必要ありません。割り当てはシステムの初期設定時に行われます。ユニットの割り当て解除、再割り当てが行われるか、無視するオプションを指定した `ASSGN` 文が検出されるまで、JCL の実行中、この割り当ては有効です。

Sun MBM では、SYSOUTDIR ディレクトリの下にディレクトリが作成されます。このディレクトリはジョブ名、ステップ名、論理ユニット名、およびジョブ番号に基づいて一意に作成されます。

POWER に必要な構成ファイル

dostrans を実行する前に、Sun MBM 管理者は次の構成ファイルを更新して、POWER 機能を提供する必要があります。

- $PUBLIC/vse/power/lu.cfg
- $PUBLIC/vse/power/pu.cfg

これらの構成ファイルは、POWER に関連付けられた論理ユニットとデバイスをメインフレームでグローバルに定義します。変換およびジョブの実行中に構成ファイルにアクセスして、POWER デバイスに適用されるファイルを判定します。

lu.cfg ファイル

このファイルには、プリンタデバイスアドレスに常時関連付けられる POWER 論理ユニットのリストが含まれています。このファイルは、$PUBLIC/vse/power ディレクトリにあります。デフォルトでは、次のものが含まれます。

- SYSLST
- SYSPCH
- SYSOUT

次の JCL 例では、POWER 論理ユニットを使用しています。

次の POWER 論理ユニット用の JCL セグメントでは、ジョブの実行時に $PUBLIC/vse/power/lu.cfg 構成ファイルに SYSLST エントリが含まれていると、SYS005 論理ユニットは SYSOUT タイプのファイルとして割り当てられます。

```
// JOB JOB001
// ASSGN SYS005,SYSLST
// ...
// EXEC  PGM=PRINTRPRT
// ...
/&
```

上記の文は次のように変換されます。

```
BEGINJOB
ASSGNDD ddname='SYS005' type='SYSOUT'
...
EXECPGM pgmname='PRINTRPRT' stepname='STEP0001'
ENDJOB
```

次の POWER 論理ユニット用の JCL セグメントでは、ジョブの実行時に $PUBLIC/vse/power/lu.cfg 構成ファイルに SYS005 エントリが含まれていると、SYS006 論理ユニットは SYSOUT タイプのファイルとして割り当てられます。

```
// JOB JOB001
// ASSGN SYS006,SYS005
// ...
// EXEC  PGM=PROG001
// EXEC  PGM=PROG999
/&
```

上記の文は次のように変換されます。

```
BEGINJOB
ASSGNDD ddname='SYS006' alias='SYS005'
EXECPGM pgmname='PROG001' stepname='STEP0001'
EXECPGM pgmname='PROG999' stepname='STEP0002'
ENDJOB
```

pu.cfg ファイル

$PUBLIC/vse/power/pu.cfg ファイルには、POWER に関連付けられるデバイスアドレスのリストが含まれています。デフォルトでは、FEE と FEF が含まれます。変換時に ASSGN 文が検出されると、形式は次のようになります。

```
ASSGN SYSxxx,hhh,...
```

説明

xxx	000 ～ 254 の英数字。
hhh	0 ～9 または A ～ F の 16 進数値。

さらに、$PUBLIC/vse/power/pu.cfg には、値 *hhh* が定義されます。この値は、SYSxxx 論理ユニットを SYSOUT タイプファイルに割り当てます。*hhh* を定義しないと、pu.cfg では SYS*xxx* 論理ユニットをダミーファイルに割り当てます。

次のオプションを指定して実行する場合の dostrans 出力

```
dostrans JCLNAME
```

構成ファイル $PUBLIC/vse/power/lu.cfg の内容は次のとおりです。

```
SYSLST
SYSPCH
SYSOUT
SYS010
```

構成ファイル $PUBLIC/vse/power/pu.cfg の内容は次のとおりです。

```
250
FEE
FEF
```

JCL は次のとおりです。

```
// JOB PRINTCUST
// ASSGN SYS010,250
// ASSGN SYS005,007
// EXEC PROG001
// EXEC PROG002
/&
```

dostrans の出力は次のとおりです。

```
BEGINJOB
ASSGNDD ddname='SYS010' type='SYSOUT'
ASSGNDD ddname='SYS005' type='DUMMY'
EXECPGM pgmname='PROG001' stepname='STEP0001'
EXECPGM pgmname='PROG002' stepname='STEP0002'
ENDJOB
```

この例では、007 は pu.cfg になく、SYS005 は lu.cfg にありません。したがって、SYS005 は DUMMY として割り当てられます。

第6章

MVS JCL のインポート

この章では、Job Editor を使用して MVS JCL をインポートする方法について説明します。この章の内容は、次のとおりです。

この章の次の手順に進む前に、Job Editor のインタフェースと機能を理解するために『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を読んでください。

Sun MBM ソフトウェアによって、バッチジョブと手続きを JCL または Sun MBM マクロジョブとして管理できます。JCL を管理する場合、Job Editor によってグラフィカルユーザーインタフェース (GUI) にインポートして、そこで変更できます。ジョブストリームの変更後、直接 Job Editor からマクロジョブを生成できます。そのあと、ジョブを通常の方法で実行できます。

MVS JCL のインポートの準備

Java アプリケーションである Job Editor を使用する前に、ノードの `batchenv` ファイルに `JDKROOT` 環境変数を追加する必要があります。`$JDKROOT` で、Java のインストール先ディレクトリを指定します。

▼ JCL のインポートを準備する

1. **インポート中のジョブを実行するサブシステムを作成します。**
   すでに作成している場合、次の手順に進みます。
2. **JCL ジョブと手続きを、それぞれ** `/jmvs` **と** `/mvsp` **ディレクトリにコピーします。**
3. **JCL ストリームで定義されたすべてのデータセットが、** `File_Map` **に定義されていることを確認します。**
4. **Job Editor でプロジェクトを作成します。**
   プロジェクトの作成手順については、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

File_Map の選択

JCL ジョブストリームをインポートする前に、`File_Map` を選択する必要があります。`File_Map` には、エントリが収められている場合があります。また、サブシステムを作成した直後は、空の場合もあります。`File_Map` の詳細は、第 2 章を参照してください。

▼ File_Map を指定する

1. **Sun MBM メインメニューを表示します。**
2. **「Job Definitions」アイコンをクリックして、Job Editor を起動します。**
3. **Job Editor のツリーペインからプロジェクトを選択します。**
4. **メニューで、「FileMap」→「Select」を選択します。**
5. **ファイル選択リストが表示されたら、有効な** `File_Map` **を選択します。**
   これは、インポート中のジョブを実行するサブシステムに関連付けられた `File_Map` です。サブシステムの `File_Map` のデフォルトの場所は次のディレクトリです。
   /*node-dir*/bam/subsys/*substem-name*
6. **選択した** `File_Map` **が空の場合、次のように事前定義のエントリを追加できます。**
   a. **メニューで、「FileMap」→「Import Entries」を選択します。**

b. **ファイル選択リストで、インポートする JCL ストリームを選択して「OK」をクリックします。**

これは、JCL で `mvstrans -v` コマンドを実行したことと同じです。

c. **インポート後に、エントリの編集が必要な場合があります。**

次のいずれかのツールを使ってエントリを編集します。

- Tool Kit。23 ページの「Tool Kit を使った `File_Map` の更新」を参照してください。
- `cfm` コマンド。詳細は、24 ページの「`cfm` コマンドを使った `File_Map` の更新」を参照してください。
- BAM の「Update Subsystem」オプション。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。
- `vi` などのテキストエディタ。

`File_Map` が完成すると、JCL をインポートする準備が整います。

JCL のインポート

`File_Map` を選択し、必要に応じて変更すると、JCL のインポートの準備が整います。

▼ JCL をインポートする

1. **Job Editor が起動されていない場合は、起動します。**
2. **ジョブまたは手続きのフォルダを選択するか、左側のツリーペインでジョブまたは手続きのノードを選択します。**

   フォルダとノードの詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。
3. **「File」→「Import」→「MVS JCL」を選択します。**

   `File_Map` を選択していなかった場合、エラーメッセージが表示されます。
4. **ファイル選択リストのダイアログボックスが表示されたら、ジョブまたは手続きを選択します。**

   この時点では、有効なジョブまたは手続きのストリームを選択していることの確認は行われません。

5. 「OK」をクリックして JCL ストリームのインポートを開始します。

   エラーが発生した場合、インポートは中止されます。手続きフォルダ内部のジョブをインポートした場合、または選択したファイルが MVS JCL ストリームではない場合、エラーが発生することがあります。

6. インポート中に、Sun MBM ランタイムまたは Job Editor ツールがサポートしていない JCL 文があると、Job Editor は警告メッセージを表示します。

   処理は続行されますが、インポートされたジョブストリームにこの文のロジックは含まれません。

7. インポートが完了したら、新しいジョブまたは手続きのノードとそのステップ、および必要なデータセットとプログラムがツリーペインに表示されます。

MVS JCL に対する Job Editor のサポート

この節では、MVS JCL に対する Job Editor のサポートについて説明します。入力 MVS JCL ストリームの変更が必要ないくつかの制限事項が存在します。JCL のインポート後に制限事項に気がついた場合、修正して再度 JCL をインポートできます。

表 6-1 Job Editor の制限事項

JCL 文	コメント
DD 文	**.stepname.ddname* 形式での DD DSN パラメータの逆方向参照はサポートされていません。
DD 連結データセット	ファイルシステムファイルのみがインポートされます。
EXEC 文	**.stepname.ddname* 形式での EXEC PGM および PROC パラメータの逆方向参照はサポートされていません。
COND パラメータ	COND.*stepname* 形式の COND パラメータはサポートされていません。
JOB 文	1 つの JCL ストリームにつき、1 つの JOB 文のみがサポートされます。
特殊文字	$、&、および非印刷文字などの特殊文字は、/x*HH* 形式の文字列に変換されます。*HH* は文字に対応する 16 進数値です。特殊文字についての詳細は、『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。
PROC 文	ストリーム内手続きと入れ子の手続きはサポートされていません。 1 つの JCL ストリームにつき、1 つの PROC 文のみがサポートされます。
SORT, ICEMAN, SYCSORT	Job Editor では、SORT SYSIN 指示の自動変換を実行しません。
継続行	最後のカンマ文字に続く空白以外の文字は、すべてインラインコメントとして処理されます。

表 6-1　Job Editor の制限事項 (続き)

JCL 文	コメント

記号パラメータ	次のように補完的なパラメータを表すために使用される記号パラメータです。
	`//SORTIN DD &OP1,DSN=AA.BB.CC`
	または、次のようにパラメータの値を指定するために使用される記号パラメータです。
	`DISP=(&DISP1,&DISP2)`
	は無視されます。

次の表に、MVS JCL 文とその関連パラメータを一覧表示し、Job Editor 環境でのサポートについて説明します。表のリストに文またはパラメータがない場合、グラフィカル環境での制限事項、Sun MBM ランタイムでの制限事項、または文で提供された機能が Sun MBM 環境に適用されないことがその原因です。

注 – ボールド書体で表示された文は、MVS JCL トランスレータではサポートされていますが、Job Editor ではサポートされていません。

MVS JCL トランスレータ (`mvstrans`) についての詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

表 6-2　MVS JCL に対する Job Editor のサポート (1 / 6)

MVS JCL 文	パラメータ	Job Editor の動作
`//*` (コメント)		コメントは、選択したノードの「Description」タブに挿入されます。
`/*` (区切り記号)		サポートされています。この文に関連付けられたグラフィカル表現は存在しません。
`//` (ジョブの終了)		サポートされています。この文に関連付けられたグラフィカル表現は存在しません。
`CNTL`		サポートされていません。
`DD`	Job Editor のツリーペインで、特定のファイルノードがこの文に関連付けられます。	
`DD`	*	入力ファイルノードが、この `DD` 文に関連付けられます。
`DD`	`DATA`	入力ファイルノードが、この `DD` 文に関連付けられます。
`DD`	`DUMMY`	このファイルノードに関連付けられている「Dummy」ラジオボタンを選択します。
`DD`	`DYNAM`	無視されます。

表 6-2　MVS JCL に対する Job Editor のサポート (2 / 6)

MVS JCL 文	パラメータ	Job Editor の動作
DD	ACCODE	無視されます。
DD	AMP	無視されます。
DD	AVGREC	無視されます。
DD	BURST	パラメータ BURST=*value* が、選択した出力ファイルノードに関連付けられます。
DD	CHARS	パラメータ CHARS=*value* が、選択した出力ファイルノードに関連付けられます。
DD	CHKPT	無視されます。
DD	CNTL	無視されます。
DD	COPIES	選択した出力ファイルノードの「Copies」フィールドが、指定した値に設定されます。
DD	DATACLAS	無視されます。
DD	DCB サブパラメータ LRECL	選択したファイルノードの「Record Size」フィールドが、指定した値に設定されます。
DD	DCB サブパラメータ RECFM	選択したファイルノードの「Record Format」フィールドが、指定した値に設定されます。
DD	DCB 他の任意のサブパラメータ	無視されます。
DD	DDNAME	別名ファイルノードが、この DD 文に関連付けられます。
DD	DEST	選択した出力ファイルノードの「Printer Destination」フィールドが、指定した値に設定されます。
DD	DISP	「disposition」、「normal」、および「abend termination」ラジオボタンが、指定した設定パラメータに従って更新されます。
DD	DLM	サポートされています。このパラメータに関連付けられたグラフィカル表現は存在しません。
DD	**DSID**	サポートされていません。
DD	DSNAME	選択したファイルノードの「Path Name」フィールドが、このデータセットの File_Map で指定したファイル名に設定されます。
DD	EXPDT	無視されます。
DD	FCB	パラメータ FCB=*value* が、選択した出力ファイルノードに関連付けられます。

表 6-2　MVS JCL に対する Job Editor のサポート (3 / 6)

MVS JCL 文	パラメータ	Job Editor の動作
DD	FLASH	パラメータ FLASH=*value* が、選択した出力ファイルノードに関連付けられます。
DD	FREE	無視されます。
DD	HOLD	パラメータ HOLD=*value* が、選択した出力ファイルノードに関連付けられます。
DD	KEYLEN	無視されます。
DD	KEYOFF	無視されます。
DD	LABEL	無視されます。
DD	LIKE	無視されます。
DD	LRECL	選択したファイルノードの「Record Size」フィールドが、指定した値に設定されます。
DD	MGMTCLAS	無視されます。
DD	MODIFY	パラメータ MODIFY=*value* が、選択した出力ファイルノードに関連付けられます。
DD	MSVGP	無視されます。
DD	OUTLIM	パラメータ OUTLIM=*value* が、選択した出力ファイルノードに関連付けられます。
DD	**OUTPUT**	サポートされていません。
DD	PROTECT	無視されます。
DD	QNAME	無視されます。
DD	RECFM	選択したファイルノードの「Record Format」フィールドが、指定した値に設定されます。
DD	RECORG	無視されます。
DD	REFDD	無視されます。
DD	RETPD	無視されます。
DD	**RLS**	サポートされていません。
DD	SECMODEL	無視されます。
DD	SPACE	無視されます。
DD	STORCLAS	無視されます。
DD	SUBSYS	無視されます。

表 6-2　MVS JCL に対する Job Editor のサポート (4 / 6)

MVS JCL 文	パラメータ	Job Editor の動作
DD	SYSOUT	出力ファイルノードが、この文に関連付けられます。文は、(JES2 サポートではなく) MVS JCL 専用の文として解釈されます。したがって、最初のパラメータを使用して、選択した出力ファイルの「Class」フィールドが設定されます。2 番目と 3 番目のパラメータを使用して、2 つの出力ファイルのパラメータ WRITER と FORMS が設定されます。
DD	TERM	無視されます。
DD	UCS	パラメータ UCS=*value* が、選択した出力ファイルノードに関連付けられます。
DD	UNIT	無視されます。
DD	VOLUME	無視されます。
DD	**JOBCAT DD** 特殊名	サポートされていません。
DD	**STEPCAT DD** 特殊名	サポートされていません。
DD	JOBLIB DD 特殊名	アプリケーションプログラムディレクトリが、選択したジョブノードに関連付けられます。
DD	STEPLIB DD 特殊名	アプリケーションプログラムディレクトリが、選択したステップノードに関連付けられます。
DD	SYSIN DD	標準または入力ファイルノードが、この文に関連付けられます。
DD	連結データセット	連結ファイルノードが、この文に関連付けられます。
DD	一時データセット	選択したファイルノードの「Temporary」ラジオボタンが選択されます。
ENDCNTL		サポートされていません。
EXEC	Job Editor のツリーペインで、特定のステップノードがこの文に関連付けられます。ステップ名を使用して、選択したステップノードのステップ名フィールドが設定されます。	
EXEC	PGM	「Utility Name」または「Program Name」フィールドが指定した名前で更新されます。
EXEC	PROC	「Procedure Name」フィールドが指定した名前で更新されます。
EXEC	ACCT	無視されます。
EXEC	ADDRSPC	無視されます。
EXEC	COND=EVEN または ONLY	このパラメータを使用して、ステップの「Cond Code」タブが更新されます。

表 6-2　MVS JCL に対する Job Editor のサポート (5 / 6)

MVS JCL 文	パラメータ	Job Editor の動作
EXEC	COND=*condition1, condition2, ...*	最初の条件を使用して、ステップの「Return Code」タブが設定されます。 そのあとのすべての条件はサポートされていません。
EXEC	**DPRTY**	サポートされていません。
EXEC	**DYNAMNBR**	サポートされていません。
EXEC	PARM	このパラメータを使用して、ステップの「Parameters」タブが更新されます。 修飾された PARM パラメータ (PARM.*stepname*) はサポートされていません。
EXEC	PERFORM	無視されます。
EXEC	RD	無視されます。
EXEC	REGION	無視されます。
EXEC	TIME	無視されます。
IF/THEN/ELSE/ENDIF		サポートされていません。
INCLUDE		サポートされています。この文に固有のグラフィカル表現は Job Editor では提供されません。
JCLLIB		ジョブの「Procedures」タブが、対応する File_Map エントリで更新されます。
JOB	Job Editor のツリーペインで、ジョブノードがこの文に関連付けられます。	
JOB	アカウンティング情報	無視されます。
JOB	ジョブの所有者	無視されます。
JOB	ADDRSPC	無視されます。
JOB	CLASS	無視されます。
JOB	**COND**	サポートされていません。
JOB	GROUP	無視されます。
JOB	MSGCLASS	選択したジョブノードの「Printer Class」フィールドが指定した値で更新されます。
JOB	MSGLEVEL	無視されます。
JOB	NOTIFY	選択したジョブノードの「NOTIFY」フィールドが指定した値で更新されます。
JOB	PASSWORD	無視されます。
JOB	PERFORM	無視されます。
JOB	PRTY	無視されます。

表 6-2　MVS JCL に対する Job Editor のサポート (6 / 6)

MVS JCL 文	パラメータ	Job Editor の動作
JOB	RD	無視されます。
JOB	REGION	無視されます。
JOB	RESTART	無視されます。
JOB	TIME	無視されます。
JOB	TYPRUN	無視されます。
JOB	USER	無視されます。
OUTPUT		サポートされていません。
PEND		サポートされていません。
PROC	Job Editor のツリーペインで、手続きノードがこの文に関連付けられます。	
PROC	*parameter*	Job Editor の「*parameters*」タブが、指定した keyword=*value* パラメータで更新されます。
SET		サポートされていません。
XMIT		無視されます。
JES2 文		サポートされていません。

第7章

JCL およびユーザーユーティリティー

mvstrans および dostrans トランスレータは、MVS および VSE の環境で使用可能なもっとも標準的な IBM JCL ユーティリティーを認識します。この章では、Sun MBM がサポートしている JCL ユーティリティーについて説明します。この章の内容は、次のとおりです。

サポートする IDCAMS 関数

Sun MBM では、一部の IDCAMS コマンドが実行時にサポートされています。以下の小節では、SYSIN ファイルに含まれる IDCAMS コマンドを Sun MBM で解釈し処理する方法について説明します。

IDCAMS コマンドを必要としない場合または直接サポートされていない場合は、Sun MBM で何も処理しなかったことを示すリターンコード 8 が発行されます。このリターンコードを変更する場合は、サブシステムのユーザー設定ファイル $USER_SETUP を、次の環境変数を含むように変更する必要があります。

```
setenv EBM_UNSUPPORTED_UTILITY n
```

n は、希望するリターンコードです。

DEFINE ALIAS

DEFINE ALIAS コマンドは、File_Map に存在する順編成システムファイル (FS) の代替データセット名を定義します。このコマンドで、同じファイル名を使った新しいエントリが File_Map に作成されます。

DEFINE CLUSTER

DEFINE CLUSTER コマンドでは、何も処理は行われません。Sun MTP 領域の FCT および VSAM カタログに、VSAM データセットを定義する必要があります。Sun MTP 領域で定義した VSAM データセットを初期化するには、IDCAMS DELETE コマンドを使用します。

IDCAMS DEFINE 関数により、実行中に次の出力メッセージが生成されます。

```
FUNCTION COMPLETED XXXXX is a VSAM dataset:command ignored
```

説明

XXXXX VSAM データセットの名前。

DELETE

DELETE コマンドはカタログを削除します。VSAM データセットと VSAM 以外のデータセットを再初期化します。サポートしているパラメータは次のとおりです。

- entryname
- CLUSTER
- CATALOG
- ALIAS
- GDG
- NONVSAM

注 – VSAM カタログで再利用不可と定義されたレコードを含む VSAM データセットは初期化できません。

IDCAMS DELETE 関数を使用すると、順編成システムファイル (FS) は削除されます。

REPRO

REPRO コマンドでは、次の機能をサポートします。

- ソース連結データセットまたはソースベース名 GDG ファイルと VSAM ターゲットデータセット
- VSAM ターゲットファイルと FS ソースファイル

  AIX プラットフォームでは、FS ソースファイルが 2G バイトを超えることはできません。
- FS ターゲットファイルと VSAM ソースファイル
- FS ファイルにコピーする連結データセットまたはベース名 GDG ファイル
- VSAM ファイルにコピーする連結データセットまたはベース名 GDG ファイル
- 値が 0 の場合の COUNT パラメータ

コード例 7-1　　出力データセットの初期化

```
REPRO INDATASET(TEST.AA) OUTDATASET(TEST.BB) COUNT(0)
```

注 – VSAM データセットは、直接、別の VSAM データセットにコピーできません。

VSAM データセットも含む REPRO コマンドの入力または出力に FS ファイルを指定すると、'filename='、'recsize='、および 'recformat=' オプションをジョブに定義している場合、FS ファイルのレコードタイプの判定は次の順序で行われます。

- 明示的に定義

  REPRO で使用します。
- 明示的に定義しない

  File_Map の 5 番目の位置に定義したファイルのレコード形式やサイズを REPRO で確認します。見つからない場合は、デフォルトの record ファイルタイプが使用されます。

IDCAMS REPRO では、下位互換のために REUSE をデフォルトの動作とします。IDCAMS SYSIN ファイルに指定した NOREUSE、REPLACE、および NOREPLACE パラメータは無視されます。

メインフレームの REPRO のデフォルトアクション NOREUSE および NOREPLACE を使用する場合は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

File_Map に定義するように *n* インスタンスを指定して入力したベース名 GDG の場合、REPRO コマンドでは、連結した *n* 個の最新インスタンス (カタログ化されるかまたはカタログ化されていない) を出力先ファイルにコピーします。

代替索引のある大きい VSAM ファイルの使用法

REPRO を使用して代替索引のある大きい VSAM ファイルを作成またはロードする場合、REPRO コマンドに EBM_SORTMEM パラメータを追加しなければならないことがあります。

構文

```
REPRO EBMSORTMEM(n)
```

n は、代替索引のあるターゲット VSAM ファイルを構築するために、内部で割り当てる主記憶のメガバイト数です。

次に例を示します。

```
REPRO INDATASET(LGE.SEQFILE) OUTDATASET(LGEVS) EBM_SORTMEM(30)
```

この例で OUTDATASET は、代替キーで定義した VSAM ファイルです。

IDCAMS モーダルコマンド

次の表に、IDCAMS モーダルコマンドのサポートを示します。

表 7-1 IDCAMS モーダルコマンド

コマンド	説明	サポート/ サポートされない
IF/THEN/ELSE	条件コードに基づいて、コマンドの実行を制御します。	サポート
DO/END	1 つより多くの機能アクセスメソッドサービスコマンドとそのパラメータを指定します。	サポート
SET	条件コードをリセットします。	サポート
CANCEL	現在のジョブまたはジョブステップの処理を終了します。	サポート
PARM	診断補助と印刷出力オプションを選択します。	サポートされない

サポートされない IDCAMS コマンドを Sun MBM で処理する方法

この節では、Sun MBM 環境で必要ないか、または直接サポートされない IDCAMS コマンドを一覧表示しています。

表 7-2 サポートされない IDCAMS コマンド

コマンド	アクション
ALTER	File_Map エントリ、および Sun MTP 領域の FCT と VSAM (VSAM データセット用) カタログを更新し、ファイル属性の変更を行います。
BLDINDEX	Sun MTP 領域からオンラインで、またはバッチジョブの IDCAMS REPRO コマンドを使用して、unikixbld コマンドを実行します。このコマンドは、入力順編成ファイルからプライマリ VSAM データセットをロードし、代替索引を自動的に構築します。
CHKLIST	適用されません。
CNVTCAT	適用されません。
DEFINE ALTERNATEINDEX	Sun MTP VSAM カタログに、VSAM データセットの代替索引を定義します。プライマリ VSAM データセットを構築するとき、定義した代替索引も構築されます。代替索引のデータセット名を File_Map に入力する必要はありません。
DEFINE GENERATIONDATAGROUP	DEFINE GDG コマンドで指定したデータセット名と、エントリの世代番号フィールドに設定した 0 より大きい値で File_Map に新規エントリを作成し、GDG を定義します。
DEFINE NONVSAM	FS ファイルタイプで File_Map に新規エントリを作成して、VSAM 以外のデータセットをシステムファイルとして定義します。
DEFINE PAGESPACE	適用されません。
DEFINE PATH	適用されません。
DEFINE SPACE	適用されません。
DEFINE USERCATALOG/MASTERCATALOG	File_Map のエントリで各カタログを定義します。
EXAMINE	適用されません。
EXPORT	適用されません。
EXPORTRA	適用されません。
IMPORT	適用されません。
LISTCAT	コマンドは無視され、IDCAMS によって次の警告メッセージが発行されます。 LISTCAT:no action

表 7-2　サポートされない IDCAMS コマンド (続き)

コマンド	アクション
LISTCRA	適用されません。
PRINT	このコマンドでは、データセットの内容を印刷できません。Sun MTP 領域のレコードエディタを使って、VSAM データセットを参照できます。
RESETCAT	適用されません。
VERIFY	VSAM データセットを検証するには、Sun MTP kixverify コマンドを使います。

IDCAMS MAXCC

以前のバージョンの Sun MBM では、0 より大きい MAXCC 値をステップの致命的エラーとして処理し、前の ONCONCODE などの条件に基づいて適切な処理を実行していました。

- BAM でサブシステムを作成する場合、MVS の「Mainframe Compatibility」画面で MAXCC を構成できます。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。
- MAXCC がメインフレーム互換性用に構成された場合、Sun MBM は、現在のステップのリターンコードを MAXCC 値に設定します。以降のステップでは、リターンコードを参照してバイパスロジックを判定できます。

IDCAMS ユーティリティーの制限事項

この節では、Sun MBM 環境で IDCAMS ユーティリティーを使用する際の制限事項を示します。

コメントの継続

制限事項: IDCAMS SYSIN データでは、行継続文字 (-) や文字列継続文字 (+) を使って 1 つのコメントを複数行に渡って継続させることはできません。

回避策: コメントを削除するか、各コメントを 1 行ずつにします。

例 1

```
REPRO /* my comment +
 continued */
```

コメントを次の行に移動します。

```
REPRO -
 /* my comment continued */
```

例 2

```
REPRO -
 /* my comment -
    continued */
```

コメントを次の行に移動します。

```
REPRO -
 /* my comment continued */
```

CATALOG パラメータを含む DELETE 文

制限事項: DELETE 文に CATALOG パラメータが含まれている場合、含まれている空白文字がすべてサポートされるわけではありません。

回避策: 括弧内の空白文字をすべて削除します。

例

```
DELETE (MY.DATASET) CATALOG(MYCATLOG/ MYPASSWD)
```

スラッシュのあとの空白文字を削除します。

```
DELETE (MY.DATASET) CATALOG(MYCATLOG/MYPASSWD)
```

ベース GDG を含む REPRO ステップ

制限事項: INFILE パラメータまたは INDATASET パラメータでベース GDG (すべての GDG インスタンスの連結) を指定している REPRO ステップは、オカレンスがない場合、失敗する可能性があります。

制限事項: REPRO INFILE/INDATASET の ENVIRONMENT(DUMMY) 構文パラメータは、現在のリリースではサポートされていません。パラメータは無視されます。

回避策: ENVIRONMENT サブパラメータを削除します。

例

```
REPRO INFILE(DD1 ENVIRONMENT(DUMMY)) OUTFILE(DD2)
```

ENVIRONMENT サブパラメータを削除します。

```
REPRO INFILE(DD1) OUTFILE(DD2)
```

ソートユーティリティーの使用法

Sun MBM では、Sun MBM sortx ユーティリティーと次の Sun 以外のパッケージのいずれかを使用して、IBM SORT と ICEMAN ソートユーティリティーをサポートしています。

- SyncSort
- CoSORT

ソートユーティリティーの選択

BAM を使ってサブシステムを作成するときにソートユーティリティーを選択できます。このソートユーティリティーは、JCL でソートが必要な場合、常に使用されます。

Sun MBM では、SORT_MODE 環境変数の値に基づいて、使用するソートユーティリティーを判定します。$SORT_MODE を設定していない場合は、sortx ユーティリティーが使用されます。

ソートユーティリティー	SORT_MODE の値	VSAM サポート
SyncSort	syncsort	入力と出力
CoSORT	cosort	入力と出力
sortx	sortx (デフォルト)	入力

注 – sortx ユーティリティーを使用する場合は、JCL SORT ステップの入力ソート指示を適切なユーティリティー構文に変換する必要があります。SyncSort および CoSORT 製品では、トランスレータを使って、IBM SORT 指示を対応する SyncSort/CoSORT 形式に変換します。

mvstrans または dostrans で JCL SORT ステップを変換すると、EXEC PGM 文が Sun MBM SORT シェルスクリプトへの呼び出しに変換されます。このスクリプトは、$PUBLIC/bin に保存され、使用するソートプログラムを判定して、SYSIN データセットの入力ソート指示に基づいてプログラムを実行します。

次の節では、SORT シェルスクリプトで実行するアクションの判定に使用される変数とファイルについて説明します。

SORTIN_TYPE 環境変数

$SORTIN_TYPE には、ソートする入力ファイルのタイプを記述します。ASSGNDD マクロは、次のいずれかの $SORTIN_TYPE を File_Map のファイルタイプフィールドに設定します。

FS　順編成システムファイル (Sun MTP VSAM 以外のファイル)。デフォルトです。

VS　VSAM データセット。

$SORTIN_TYPE を VS に設定すると、sortx、SyncSort、または CoSORT ユーティリティーが呼び出されます。SyncSort および CoSORT ユーティリティーは、通常 sortx よりパフォーマンスにすぐれています。

コマンド行インタフェースを使用する場合は、$SORTIN_TYPE を VS に設定する際に、unikixjob コマンドを使ってジョブをサブミットする必要があります。

SYSIN ファイルによる入力 VSAM データセットソートの制御

ソートステップで指定した SYSIN ファイルを使用して、ソートステップの機能を制御できます。次の条件が考えられます。

- ソート指示を含む SYSIN ファイルが存在する場合

  SORT シェルスクリプトは、サブシステム構成に基づいて、適切なソートユーティリティーを呼び出します。詳細は、194 ページの「ソートユーティリティーの選択」を参照してください。

- 空の SYSIN ファイルが存在する場合

  IBM メインフレームでは、SORT コマンドを使って VSAM データセットをフラットファイルにコピーし (COPYFILE オプション)、パフォーマンスを高めることができます。SYSIN ファイルが存在し、空の場合は、Sun MBM でこの機能を提供します。

  ソートユーティリティーは呼び出されません。代わりに、SORT シェルスクリプトで unikixbld コマンドを実行し、VSAM データセットをフラットファイルに変換します。

- シェルスクリプトを含む SYSIN ファイルが存在する場合

  システムコマンドを実行する方が望ましい場合があります。たとえば、VSAM データセットのバックアップを作成するには、最初に VSAM データセットをフラットファイルに変換して保存するのではなく、kixfile および UNIX cp コマンドを使って、.dta と .idx ファイルをコピーします。

  VSAM データセットをコピーする場合、cp コマンドの前に、kixfile コマンドをまず使用する必要があります。

  UNIX コマンドを含むシェルスクリプトを実行するように Sun MBM に通知するには、SYSIN ファイルの最初の行に次の値を設定します。

  ```
  #!/bin/shell
  ```

  shell は使用しているシェル (sh、csh、ksh) を定義します。

  実行時に、SYSIN ファイルの最初の行にこれらの文字があるかどうか確認されます。ある場合は、SYSIN ファイルをシェルスクリプトとして実行します。

  ユーザーは実行するシェルコマンドを指定して、コマンドのエラー状態を確認し、exit コマンドにより Sun MBM に状態を渡します。

SYSIN ファイルに基づいて実行時に行われるアクション、入力ファイルのタイプ、および出力先ファイルのタイプについては、表 7-3 も参照してください。

SORTOUT_TYPE 環境変数

$SORTOUT_TYPE は、ソートステップでソートした出力先ファイルのタイプを記述します。

FS	順編成システムファイル (Sun MTP VSAM 以外のファイル)。デフォルトです。
VS	VSAM データセット。

ASSGNDD マクロは、$SORTOUT_TYPE をソート出力先ファイルエントリの File_Map の「file_type」フィールドに設定します。VS を $SORTOUT_TYPE に設定した場合、SORT シェルスクリプトでは、次の節で説明するように、ソートの動作を制御します。

SYSIN ファイルによる出力 VSAM データセットソートの制御

ソートステップで指定した SYSIN ファイルを使用して、ソートステップの機能を制御できます。その場合、次の 3 つの条件が考えられます。SYSIN ファイルに基づいて実行時に行われるアクション、入力ファイルのタイプ、および出力先ファイルのタイプについては、表 7-3 も参照してください。

- ソート指示を含む SYSIN ファイルが存在する場合

  Sun MBM では、サブシステム構成に基づいてソートを実行します。詳細は、194 ページの「ソートユーティリティーの選択」を参照してください。次に、unikixbld コマンドを実行し、ソートした FS ファイルを VSAM データセットに変換します。

- 空の SYSIN ファイルが存在する場合

  Sun MBM では、適切な unikixbld コマンドを使って、VSAM データセットファイルを FS ファイルからロードします。

- シェルスクリプトを含む SYSIN ファイルが存在する場合

  システムコマンドを含むシェルスクリプトを実行するように Sun MBM に通知するには、SYSIN ファイルの最初の行に次の値を設定します。

  ```
  #!/bin/shell
  ```

  shell は使用しているシェル (sh、csh、ksh) です。

  実行時に、SYSIN ファイルの最初の行にこれらの文字があるかどうか確認されます。ある場合は、SYSIN ファイルをシェルスクリプトとして実行します。

  ユーザーは実行するシェルコマンドを指定して、コマンドのエラー状態を確認し、exit コマンドにより Sun MBM に状態を渡します。

表 7-3 SYSIN に基づくソートの動作

SYSIN	SORTIN	SORTOUT	アクション
存在しません	FS	FS	ソートユーティリティーのエラー。sortx、SyncSort、または CoSORT によりエラーメッセージが生成されます。
	VS	FS	エラー。ソートユーティリティーによりエラーメッセージが生成されます。
	FS	VS	エラー。ソートユーティリティーによりエラーメッセージが生成されます。
	VS	VS	サポートされていません。

表 7-3　SYSIN に基づくソートの動作 (続き)

SYSIN	SORTIN	SORTOUT	アクション
空ファイル (存在する)	FS	FS	ソートユーティリティーのエラー。sortx、SyncSort、または CoSORT によりエラーメッセージが生成されます。
	VS	FS	REPRO/unikixbld コマンドが呼び出されます。
	FS	VS	REPRO/unikixbld コマンドが呼び出されます。
	VS	VS	サポートされていません。
インストリームが空です	FS	FS	ソートユーティリティーのエラー。sortx、SyncSort、または CoSORT によりエラーメッセージが生成されます。
	VS	FS	REPRO/unikixbld コマンドが呼び出されます。
	FS	VS	REPRO/unikixbld コマンドが呼び出されます。
	VS	VS	サポートされていません。
存在します (ソート指示を含む)	FS	FS	sortx、SyncSort、または CoSORT によりソートが実行されます。
	VS	FS	sortx または SyncSort によりソートが実行されます。
	FS	VS	sortx、SyncSort、または CoSORT によりソートが実行され、unikixbld コマンドが呼び出されます。
	VS	VS	SyncSort サポート。
存在します (シェルスクリプトを含む)	FS	FS	SYSIN シェルスクリプトを実行します。
	VS	FS	SYSIN シェルスクリプトを実行します。
	FS	VS	SYSIN シェルスクリプトを実行します。
	VS	VS	サポートされていません。

sortx ユーティリティーの使用法

固定レコード長の順編成システムファイルまたは VSAM データセットの場合に sortx を使用できます。入力ファイルがこれらのファイルタイプでない場合は、ソートステップの前に IDCAMS ステップを挿入して、ソート入力ファイルを受入可能なファイルタイプの 1 つに変換する必要があります。移行が正しく実行されるように、元の JCL の変更が必要となる場合もあります。

sortx ユーティリティーを使うと、次のことができます。

- レコードクラスの定義
- クラスに属するレコードの取り込みまたは省略
- クラスごとに異なるチェックフィールドの定義
- クラスごとに異なる出力レコードの定義
- 同じキー値のレコードのフィールドの合計
- 省略したレコードの別のファイルへの経路指定
- ソート作業領域の定義
- 一次および二次キーの定義

デフォルトの sortx 実行可能ファイルは $PACK/bin/sortx です。

VS から FS にソートするには、Sun MBM SORT シェルスクリプトで $UNIKIX/bin/sortx を使用します。

sortx コマンドの形式

sortx コマンドの構文は、次のとおりです。

sortx -i {*infile* | stdin} [-m *n*M] -o {*outfile* | stdout} [-r *erropt*] [-s *inrsz*] [-S *source*] [-t *filetype*] [-w *wkdir*] [-x *excfile*]

説明

-i {*infile* | stdin}	*infile* は、VSAM データセットの場合はデータセットの名前で、順編成システムファイルの場合はパス名です。最大文字数は 255 文字です。複数の順編成ファイルを指定するには、$DD_SORTIN にファイルパス名のリストを設定します。次に例を示します。 DD_SORTIN=*pathname1*:*pathname2*.....
-m *n*M	ユーティリティーで使用する主記憶の最大サイズ (M バイト)。デフォルト値は 1M バイトです。次に例を示します。 -m 2M (2M バイトの場合) SORT ユーティリティーを使用する場合、サブシステム設定ファイルに SORT_WK 環境変数を設定して、動的メモリーを増やすことができます。 例: setenv SORT_WK "-m 2M" sortx は、最大使用可能量に対する使用メモリー量を 2 行目に表示します。
-o {*outfile* | stdout}	出力先ファイル *outfile* を作成するか、ソートした出力を標準出力 stdout に表示します。出力先ファイルは 1 つだけ作成されます。

-r *erropt*	データマッピングエラーの場合の処理を指定します。 abt: 処理を中止します (デフォルト) ig: エラーを無視します -s *inrsz*: *infile* 内のレコードの入力レコードサイズ (FS ファイルの場合のみ有効)
-S *source*	使用するソースファイル。たとえば、Sun MBM トランスレータでは $DD_SYSIN を sortx 制御文のソースを含むファイルに設定します。
-t *filetype*	入力ファイルタイプ VS: VSAM ファイル FS: 順編成システムファイル
-w *wkdir*	一時作業ファイル用のディレクトリ。デフォルトディレクトリは現在のディレクトリです。
-x *excfile*	例外ファイルを含むファイルパス名 *excfile*。

sortx の動作を記述する指示は、SYSIN ファイルまたは標準入力 (標準入力からの入力ファイルではない場合) として指定します。

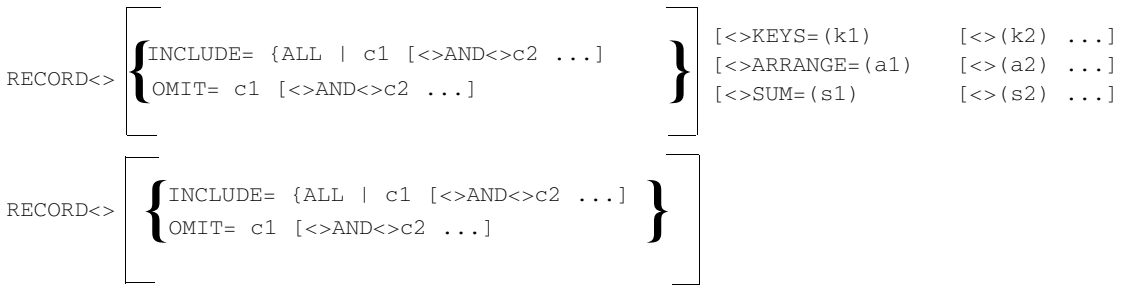
形式の指定で、次の記号を使用します。

<>	空白文字
{ }	オプションを 1 つ選択します
[ ]	オプションのパラメータ
|	OR 演算子

sortx 制御文の形式

sortx 制御文の形式

```
[FUNCTION:  [DESCEND][<>ALTSEQ=hex-string]
            [COLLATE={ ASCII | EBCDIC}]]
ENDSORT:
```



説明

FUNCTION　追加パラメータを指定するオプションの文。指定しない場合は、デフォルト値が使用されます。

DESCEND　指定したキーを降順にソートします。DESCEND を指定し、キーで RV (元に戻す) パラメータを指定すると、キーソートは昇順に戻ります。

ALTSEQ　指定された代替照合手順。値は文字の組を表す 16 進数文字列です。最初の文字は照合手順の 2 番目の文字で置換されます。
このパラメータを指定しない場合は、COLLATE で指定した手順が使用されます。

COLLATE　照合手順
ASCII (デフォルト)
EBCDIC

RECORD　オプションを適用するレコードのクラス。少なくとも 1 つの RECORD 文を指定する必要があります。

INCLUDE　クラスにレコードを取り込む条件。RECORD 文を 1 つ指定した場合のデフォルト値は INCLUDE=ALL です。
複数の RECORD 文を指定する場合は、INCLUDE または OMIT オプションを指定する必要があります。
c1 および c2 で、レコードを取り込む条件を指定します。デフォルトは CHAR です。

OMIT　レコード省略の条件。
c1 および c2 で、レコードに省略を適用する条件を指定します。デフォルトは CHAR です。

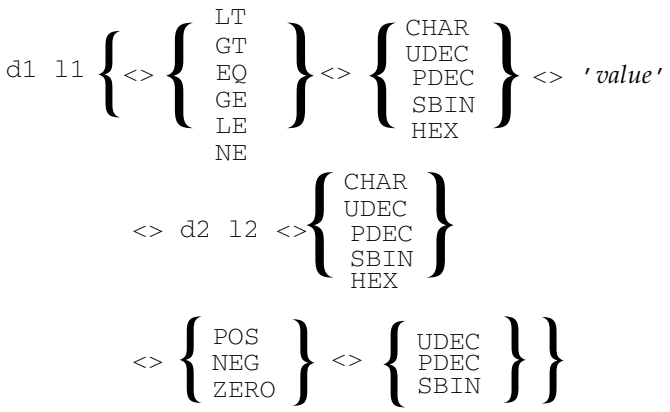
KEYS　指定したクラスのレコードのソートキーフィールド。
k1 および k2 で、入力レコードのキーフィールドを指定します。デフォルトは CHAR です。

ARRANGE　出力レコードのレイアウト。デフォルトはレコード単位です。
a1 および a2 で、出力レコードレイアウトを指定する入力レコードフィールドを指定します。デフォルトは CHAR です。

SUM　　出力レコードの合計フィールド。OMIT を指定した場合は、SUM を指定しないでください。

s1 および s2 で、使用する入力レコードフィールドを指定します。デフォルトは PDEC です。

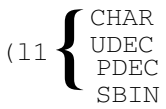
c1 および c2 の形式は、次のとおりです。



a1 および a2 の形式は、次のとおりです。

k1 および k2 の形式は、次のとおりです。

(FORCE f1[<>OR f2...][<>OR FORCE *literal*)



s1 および s2 の形式は、次のとおりです。

(d1 l1 { UDEC | PDEC | SBIN }

説明:

d1 d2	比較するフィールドが始まるレコードの変位。デフォルトは CHAR です。
l1 l2	フィールドの長さ。デフォルトは CHAR です。
f1 f2	確認条件 形式: d1 リテラル TO リテラル
CHAR	文字
EQ	等しい
GE	以上
GT	より大きい
HEX	16 進数
LT	より小さい
LE	以下
NE	等しくない
NEG	負
PDEC	パック 10 進数
POS	正
SBIN	符号付きバイナリ (COBOL では COMP-1/COMP-2)
UDEC	アンパック 10 進数 (COBOL では表示可能な数値)
ZERO	ゼロ

次の例に、RECORD 宣言を示します。

コード例 7-2 RECORD 宣言

```
 RECORD:
        INCLUDE=(1 1 LT '0') AND (2 1 EQ '0')
        INCLUDE=(3 2 NE '01') AND (7 1 EQ '0')
        KEYS=(1 4)
 ENDSORT:
 RECORD:
        KEYS=(1 4)
        ARRANGE=(1 CHAR 'Z') (5 15) (1 PDEC '3') (20 1)
```

コード例 7-2　　RECORD 宣言 (続き)

```
ENDSORT:
RECORD:
       INCLUDE=(4 1 NE ’3’)
       KEYS=(2 3)
       SUM=(6 1 UDEC) (10 1 UDEC)
RECORD:
       INCLUDE=(4 1 EQ ’3’)
       KEYS=(2 1) (3 2)
ENDSORT:
RECORD:
       INCLUDE=(4 1 NE ’3’)
       KEYS=(2 3)
       SUM=(6 1 UDEC) (10 1 UDEC)
RECORD:
       KEYS=(2 3)
       ARRANGE=(5 15) (1 4) (20 1)
ENDSORT:
RECORD:
       OMIT=(4 1 EQ ’3’) and (3 1 NE UDEC ’1’)
RECORD:
       INCLUDE=(4 1 NE ’1’)
       KEYS=(1 4)
ENDSORT:
```

レコード

順編成システムファイル (FS) の場合は、同じ長さの入力レコードを使ってソートする必要があります。レコードは別の入力ファイルに存在していてもかまいません。複数の入力ファイルも可能です。レコードをレコードクラスに分割することもできます。

レコードクラス

レコードクラスには、対応する 1 つまたは複数の特性を持つ 1 つまたは複数のレコードが含まれます。レコードクラスは、RECORD 文と INCLUDE および OMIT 句を使って定義します。

sortin ファイルに次の値が含まれているとします。

```
31A9
72A6
51A6
41B7
71A3
21A8
```

これらのレコードをソートする場合は、次の条件をすべて満たす必要があるとします。

- 2 番目の文字は 1
- 3 番目の文字は A
- 4 番目の文字は最初の文字より大きい

各条件でレコードクラスの特性を定義します。

SYSIN ファイルには、このクラスの定義に使用できる RECORD 宣言が含まれています。

```
RECORD:
       INCLUDE=(2 1 EQ '1') AND (3 1 EQ 'A') AND (4 1 GT 1)
       KEYS (1 1)
ENDSORT:
```

最初の条件は、INCLUDE パラメータ (2 1 EQ '1')で定義され、次のように解釈されます。

レコードの 2 番目の文字、1 文字とリテラル '1' を比較します。等しい場合、条件は一致します。

その他の条件も同様に解釈されます。

sortx コマンド

```
$ sortx -i sortin -s 5 -o sortout -x none -r abt -S sysin -m 1024
```

サイズ (-s 5) は、各レコードの最後にある改行文字も考慮して指定します。

sortout ファイル

```
21A8
31A9
51A6
```

none ファイル

```
72A6
41B7
71A3
```

OMIT 句

OMIT 句は、レコードクラスと同じように機能しますが、条件を満たすレコードはソートに組み込まれないで削除されます。

たとえば、前の節で同じ sortin ファイルを指定した場合、レコードは次の条件を満たしている必要があるとします。

- 2 番目の文字が 1 のレコードを削除
- 3 番目の文字は A
- 4 番目の文字は最初の文字より大きい

SYSIN ファイルには、次の RECORD 宣言が含まれています。

```
RECORD:
       OMIT=(2 1 EQ '1') and (3 1 EQ 'A') and (4 1 GT 1)
RECORD:
       KEYS (1 4)
ENDSORT:
```

sortx コマンド

```
$ sortx -i sortin -s 5 -o sortout -x none -r abt -S sysin -m 1024
```

sortout ファイル

```
41B7
71A3
72A6
```

none ファイル

```
31A9
51A6
21A8
```

各入力レコードの RECORD 宣言が順に確認されます。1 つの RECORD 宣言の条件に合致するレコードは、現在のレコードクラスに割り当てられ、残りの RECORD 宣言の確認は中止されます。

レコードは 1 つのレコードクラスだけに属することができます。レコードクラスごとに出力フィールドと合計フィールドのキーフィールドを指定できます。これにより、異なるクラスに属するレコードを一意に処理できます。

KEYS 句

KEYS 句では、各レコードクラスのキーフィールドを定義します。INCLUDE 句を指定する RECORD 宣言ごとにキーフィールドを定義する必要があります。

次の規則が適用されます。

- 各レコードクラスは独自のキーフィールドを指定する必要がある。
- 各レコードクラスには同じ長さのキーフィールドが含まれている必要がある。
- 各レコードのキーフィールドは同じタイプである必要はない。

sortx コマンドで、各入力レコードのクラスが判定され、キー指定によって、各レコードクラスのチェックフィールドが生成されます。さらに、生成されたチェックフィールドに基づいてすべてのレコードをソートします。

この例では、sortin ファイルに次の値が含まれているとします。

```
0010#100A2XXXXXXXXX
0010#100C2XXXXXXXXX
0010#100A2XXXXXXXXX
0013#100F2XXXXXXXXX
0013#100A2XXXXXXXXX
0013#100A2XXXXXXXXX
0013#100F2XXXXXXXXX
0004#100F2XXXXXXXXX
0004#100A2XXXXXXXXX
0004#100A2XXXXXXXXX
0004#100F2XXXXXXXXX
0004#100C2XXXXXXXXX
0005#10012XXXXXXXXX
```

1. 最初の 4 文字と 9 番目の文字に基づいて、ファイルをソートします。
2. 最初の 4 文字を主キーとしてレコードをソートします。

   重複するキーがあるため、同じキーを持つ複数のレコードが存在します。重複するキーのグループ内では、9 番目の文字が C のレコードは 9 番目が F のレコードより優先され、さらにこの 9 番目が F のレコードは 9 番目が A のレコードより優先されます。

SYSIN ファイルには、次の RECORD 宣言が含まれています。

```
RECORD:
       INCLUDE=(9 1 EQ'F')
       KEYS=(1 1 CHAR) (3 2 UDEC ) (1 PDEC '7')
RECORD:
       INCLUDE=(9 1 EQ 'A')
       KEYS=(1 1 CHAR) (3 2 UDEC ) (1 PDEC '8')
RECORD:
       INCLUDE=(9 1 EQ 'C')
       KEYS=(1 1 CHAR) (3 2 UDEC ) (1 PDEC '6')
ENDSORT:
```

これにより、3 つのレコードクラスが定義されます。キー句は次のように解釈されます。

(1 1 CHAR)	主キーフィールドは入力レコードの文字 1 です。
(3 2 UDEC)	二次キーフィールドには入力レコードの文字 3 と 4 が含まれ、タイプは符号なしの 10 進数です。
(1 PDEC'7')	値 '7' を文字 1 に割り当てます。確認だけに使用されます。

最初のレコードクラスでは、レコードの 9 番目の文字が F に等しい場合、ソート時に最初の文字を 7 に設定することを指定します。ソートは昇順 (デフォルト) であるので、ソートキーフィールドが等しいと、9 番目の文字が C のレコードが先に表示されます。9 番目の文字が F のレコードはそのあとに続き、9 番目の文字が A のレコードは F のレコードのあとに続きます。

sortx コマンド

```
$ sortx -i sortin -s 20 -o sortout -x none -r abt -S sysin -m1024
```

sortout ファイルの内容

```
0004#100C2XXXXXXXXX
0004#100F2XXXXXXXXX
0004#100F2XXXXXXXXX
0004#100A2XXXXXXXXX
0004#100A2XXXXXXXXX
0010#100C2XXXXXXXXX
0010#100A2XXXXXXXXX
0010#100A2XXXXXXXXX
0013#100F2XXXXXXXXX
0013#100F2XXXXXXXXX
0013#100A2XXXXXXXXX
0013#100A2XXXXXXXXX
```

none ファイル

```
0005#10012XXXXXXXXX
```

FORCE 句

FORCE 句を説明するために、次の例を示します。sortin ファイルに次のレコードが含まれているとします。

```
A189#10012XXXXXXXXX
B134#10012XXXXXXXXX
A185#10012XXXXXXXXX
C194#10012XXXXXXXXX
A182#10012XXXXXXXXX
C139#10012XXXXXXXXX
A152#10012XXXXXXXXX
B109#10012XXXXXXXXX
C166#10012XXXXXXXXX
B188#10012XXXXXXXXX
```

レコードの 2 番目の文字を主キーとして使用して、レコードをソートします。キーが等しい場合は、最初の文字に文字 Aを含むレコードが最初です。最初の文字に文字 C を含むレコードが続きます。最初の文字に文字 Bを含むレコードがそのあとに続きます。

SYSIN ファイルには、次の RECORD 宣言が含まれています。

```
RECORD:
       KEYS=(2 1) (FORCE 1'A' TO '1' OR 1'C' TO '2' OR 1'B' TO '3')
ENDSORT:
```

次の sortx コマンドがあるとします。

```
$ sortx -i sortin -s 20 -o sortout -x none -r abt -S sysin -m 1024
```

次の sortout ファイルが生成されます。

```
A185#10012XXXXXXXXX
A182#10012XXXXXXXXX
A152#10012XXXXXXXXX
A189#10012XXXXXXXXX
C139#10012XXXXXXXXX
C166#10012XXXXXXXXX
C194#10012XXXXXXXXX
B188#10012XXXXXXXXX
B109#10012XXXXXXXXX
B134#10012XXXXXXXXX
```

ARRANGE 句

さまざまな入力レコードクラスからの出力レコードに対して、同じサイズを使用する必要があります。ARRANGE 句は、出力レコードのレイアウトを指定します。レコードクラスごとに、出力レコードに対して独自のレイアウトを指定できます。

注 – ARRANGE 句は、出力レコードサイズを上書きする場合があります。

この例では、sortin ファイルに次のレコードが含まれているとします。

```
A089#10012XXXXXXXXX
B034#10012XXXXXXXXX
A085#10012XXXXXXXXX
B094#10012XXXXXXXXX
A082#10012XXXXXXXXX
B039#10012XXXXXXXXX
A052#10012XXXXXXXXX
B009#10012XXXXXXXXX
A066#10012XXXXXXXXX
A088#10012XXXXXXXXX
```

1. 各レコードの最初の文字に基づいて、A と B の 2 クラスのレコードを定義します。
2. 主キーは、2 ～ 5 番目の文字で構成されます。
3. クラス A の出力レコードでは X が * に置換され、クラス B の 出力レコードでは X が - に置換されます。

   SYSIN ファイルには、次の RECORD 宣言が含まれています。

```
RECORD:
    INCLUDE=(1 1 EQ 'A')
    KEYS=(2 4)
    ARRANGE=(1 2) (1 '0') (4 1) (1 '*') (6 2) (12 '012*********') (20 1)
RECORD:
    INCLUDE=(1 1 EQ 'B')
    KEYS=(2 4)
    ARRANGE=(1 2) (1 '0') (4 1) (1 '*') (6 2) (12 '012---------') (20 1)
ENDSORT:
```

レコードクラス A の場合、ARRANGE 句は次のように解釈されます。

入力レコード文字	->	出力レコード文字
(1 2)	1,2	1,2
(1 '0')		3 ('0')
(4 1)	4	4
(1 '*')		5 ('*')
(6 2)	6,7	6,7
(12 '012*********')		8-19 ('012*********')
(20 1)	20	20

次の sortx コマンドがあるとします。

```
$ sortx -i sortin -s 20 -o sortout -x none -r abt -S sysin -m 1024
```

次の sortout ファイルが生成されます。

```
B009*10012---------
B004*10012---------
B009*10012---------
A002*10012*********
A006*10012*********
A002*10012*********
A005*10012*********
A008*10012*********
A009*10012*********
B004*10012---------
```

SUM 句

各レコードクラスに SUM 句を指定できます。SUM 句を使用すると、同じキーフィールド値を持つ同じクラス内で、異なるレコードの指定したフィールドを合計できます。キーフィールド値と合計フィールドの 1 レコードだけが出力先ファイルに書き込まれます。

この概念を説明するため、この例では、sortin ファイルに次のレコードが含まれているとします。

```
A120####01
B133######
A121####02
A120####12
B134######
B001######
A120####02
```

1. 次の 2 つのレコードクラスを定義します。レコードの 1 番目の文字が A の場合はクラス A、レコードの 1 番目の文字が B の場合はクラス B です。
2. 主キーは、レコードの 2、3、4 番目の文字で構成されます。

3. クラス A では、クラスのレコードの主キーが同じ場合は、位置 9 と 10 の文字が合計されます。

   SYSIN ファイルには、次の RECORD 宣言が含まれています。

```
RECORD:
       INCLUDE=(1 1 EQ 'A')
       KEYS=(2 3)
       SUM=(9 2 UDEC)
RECORD:
       INCLUDE=(1 1 EQ 'B')
       KEYS=(2 3)
ENDSORT:
```

次の sortx コマンドがあるとします。

```
$ sortx -i sortin -s 11 -o sortout -x none -r abt -S sysin -m 1024
```

次の sortout ファイルが生成されます。

```
B001######
A120####15
A121####02
B133######
B134######
```

120 の主キーを持つ 1 レコードだけが sortout ファイルに書き込まれます。合計フィールドは、120 の主キーを持つすべての入力レコードの合計です。

FUNCTION 宣言

FUNCTION 宣言では、すべてのレコードクラスに含めるオプションを指定します。次のオプションを指定できます。

- 代替順序 (ALTSEQ 句)
- 照合規則 (COLLATE 句)
- 降順ソート

ALTSEQ 句

ALTSEQ 句を使用すると、ソートのために文字に別の値を割り当てることができます。

この概念を説明するため、この例では、sortin ファイルに次のレコードが含まれているとします。

```
A001
B002
C003
D004
A001
B002
C003
D004
```

1. 主キーは各レコードの最初の文字です。
2. 必要なソート順序は、D、A または B、C、E、F です。

   SYSIN ファイルには、FUNCTION 宣言と RECORD 宣言が含まれています。

```
FUNCTION:
       ALTSEQ=41424441
RECORD:
       INCLUDE=ALL
       KEYS=(1 1)
ENDSORT:
```

ALTSEQ=41424441 は、41 (A) を 42 (B) と置換し、44 (D) を 41 (A) と置換すると解釈されます。この置換は、ソート時に行われますが、出力レコードには反映されません。

次の sortx コマンドがあるとします。

```
$ sortx -i sortin -s 5 -o sortout -x none -r abt -S sysin -m 1024
```

次の sortout ファイルが生成されます。

```
D004
D004
A001
A001
B002
B002
C003
C003
```

COLLATE 句

COLLATE 句を使用すると、レコードの順序付けに使用する文字セットを指定できます。有効な文字セットは、ASCII または EBCDIC です。

sortx の使用規則

sortx には次の規則が適用されます。

1. 空の出力先ファイルと例外ファイルを使用する必要があります。
2. 入力ファイルがファイルシステム (FS) ファイルの場合、出力先ファイルを入力ファイルに指定できます。ただし、終了前にソートが中断された場合、入力ファイルの内容が不整合になることがあります。したがって、出力先ファイルを入力ファイルに指定しないでください。
3. 例外ファイルに書き込まれる各レコードサイズは、対応する入力レコードのサイズと同じです。ARRANGE 句は、例外ファイルには適用されません。
4. 出力レコードサイズは、ARRANGE 句により判定されます。ARRANGE 句を指定しない場合、デフォルトのサイズは入力レコード全体です。
5. レコードで UDEC または PDEC として定義されたフィールドに無効なデータが含まれていると、-i ig オプションを指定した場合以外は、sortx ユーティリティープログラムは異常終了します。レコードはスキップされ、例外ファイルに書き込まれません。

6. 「SUM」フィールドにオーバーフローまたは無効なデータがあると、強制的な中止の原因となります。無視オプションがある場合にはレコードがスキップされないので、データが保証されません。

7. 順編成システムファイル (FS) または標準ファイルからの入力ファイルの場合、すべてのレコードは、inrsz オプションで指定したレコードと同じ長さである必要があります。この長さには、レコードの終了文字 (CR など) も含まれます。

8. 標準構文を使って、スクリプトに標準入力 (stdin) を指定できます。次に例を示します。

```
$ sortx -i cust -o ocust -S stdin -s 200 << !
RECORD:INCLUDE=(33 1 EQ 'A') KEYS=(1 5)
RECORD:INCLUDE=ALL KEYS=(1 4)(8 1)
ENDSORT:
!
```

9. 複数の INCLUDE または OMIT 句を指定している場合、レコードは INCLUDE/OMIT 句の順に分析されます。

10. 1 つの RECORD 文に明示的な INCLUDE または OMIT 句が含まれる場合、条件を満たさないすべてのレコードは省略されるかまたは組み込まれるとみなされます。

11. すべての長さはバイト数で表します。数値フィールドの場合は、次の長さにする必要があります。

    SBIN (COMP-1 または COMP-2) の場合、2 バイトまたは 4 バイト

    UDEC の場合、最大 18 バイト

    PDEC の場合、最大 10 バイト

レコード形式と長さの指定

入力ファイルが VSAM (VS) の場合、すべてのレコード長とレコードタイプ情報は、Sun MTP 領域の FCT および VSAM カタログに維持されます。この節の後半では、順編成システム (FS) ファイルについて説明します。

入力ファイルが順編成システムファイルの場合、JCL ストリームまたはファイルにより、レコードタイプと長さを指定できます。sortx の入力ファイルには、固定長レコードが含まれている必要があります。すべてのレコードは同じサイズです。ARRANGE 句を使用して、出力レコードサイズを上書きできます。

JCL トランスレータでは、ソート入力ファイルのレコード長を次のように評価します。

1. MVS JCL `DD` 文の `LRECL` パラメータまたは VSE JCL `DLBL` 文の `RECSIZE` オペランドでソート入力ファイルを定義している場合、トランスレータではその値が使用されます。

2. レコードサイズパラメータが指定されていない場合、JCL トランスレータでは `File_Map` エントリの 5 番目の位置でファイルのフルパス名を確認します。このエントリにはレコード形式と長さが含まれます。レコード形式と長さが見つかると、トランスレータではその値を使用します。`File_Map` についての詳細は、第 2 章を参照してください。

   ファイルが存在しないか、もしくは情報が含まれていない場合、トランスレータではソート出力先ファイルとして定義されたファイルのレコード長を使用します。デフォルトのレコードタイプが使用されることを示す警告メッセージが表示されます。

トランスレータでは、次のようにソートされた出力先ファイルとして定義されたファイルのレコード長を評価します。

1. ソート出力先ファイルを定義する JCL 文のレコードサイズパラメータに長さが宣言されていると、トランスレータではその長さを使用します。

2. ソート出力先ファイルを定義する JCL 文のレコードサイズパラメータで長さが宣言されていない場合、トランスレータでは `File_Map` エントリの 5 番目の位置でファイルのフルパス名を確認します。このエントリにはレコード形式と長さが含まれます。レコード形式と長さが見つかると、トランスレータではその値を使用します。`File_Map` の詳細は、第 2 章を参照してください。

   ファイルが存在しないか、情報が含まれていない場合、トランスレータではデフォルトのレコード長 80 を使用します。

VSAM 変換に関連する順編成ファイルで使用するレコード形式

Sun MBM `SORT` シェルスクリプトを使用して、VSAM データセットを順編成ファイルにコピーしたり、またはその逆を行うことができます。`SORT` シェルスクリプトで呼び出される `unikixbld` ユーティリティーは、これらの変換に使用される順編成ファイルのレコード形式を必要とします。ユーティリティーは、`File_Map` エントリの順編成ファイルの 5 番目のフィールド (`record_format` と `record_length` を含むファイルを指定するフィールド) でファイルのフルパス名を確認して、この情報を取得します。ファイルが定義されている場合は、これらの値が使用されます。

record_format パラメータには、次の値が使用されます。

line	エディタで操作できるライン形式ファイル。ラインファイルは、行ごとに改行文字で終了します。
record	固定長レコードを含む順編成ファイル。各レコードには、2 進データを含めることができます。レコード終端記号は追加されません。
recordv	可変長レコードを含む順編成ファイル。各レコードには、record 形式などの 2 進データを含めることができます。各レコードには、レコード長を含む 4 バイトの 2 進フィールドが先行します。
mfrcd	固定長レコードを含む Server Express 順編成ファイル。
mfrcdv	可変長レコードを含む Server Express 順編成ファイル。各レコードには、record および recordv 形式などの 2 進データを含めることができます。

詳細は、7 ページの「File_Map の形式」を参照してください。

ソートキー

データタイプソートキーとして、次のように、文字、パック 10 進数、またはアンパック 10 進数 (DISPLAY) を使用できます。

CHAR	文字ソートフィールド
PDEC	パック 10 進数ソートキー
UDEC	アンパック 10 進数ソートキー

ソートの例

次の例では、sortx コマンドにソート指示を指定します。変換する前に、JCL ストリームを変更する必要があります。元の MVS JCL では、SORT FIELDS=(1,2,A), FORMAT=CH を次のように変更する必要があります。

元の JCL

```
//STEP15 EXEC PGM=ICEMAN
//SYSOUT DD SYSOUT=*
//SORTIN DD DSN=XIAVA0.V000.MAC.OS.SEQ.MCHE000,DISP=SHR
//SORTOUT DD DCB=(LRECL=72,BLKSIZE=7200,RECFM=FB),
//     DSN=&&MCHE000,
//     DISP=(,PASS),UNIT=WORKA,
//     SPACE=(CYL,(3,2))
//SORTWK01 DD SPACE=(CYL,(3,1),RLSE),UNIT=SYSDA
//SORTWK02 DD SPACE=(CYL,(3,1),RLSE),UNIT=SYSDA
//SORTWK03 DD SPACE=(CYL,(3,1),RLSE),UNIT=SYSDA
```

```
//SYSIN DD *
SORT FIELDS=(1,2,A),FORMAT=CH
//*
```

上記の文は次のように変更されます。

```
//STEP15 EXEC PGM=ICEMAN
//SYSOUT DD SYSOUT=*
//SORTIN DD DSN=XIAVA0.V000.MAC.OS.SEQ.MCHE000,DISP=SHR
//SORTOUT DD DCB=(LRECL=72,BLKSIZE=7200,RECFM=FB),
//     DSN=&&MCHE000,
//     DISP=(,PASS),UNIT=WORKA,
//     SPACE=(CYL,(3,2))
//SORTWK01 DD SPACE=(CYL,(3,1),RLSE),UNIT=SYSDA
//SORTWK02 DD SPACE=(CYL,(3,1),RLSE),UNIT=SYSDA
//SORTWK03 DD SPACE=(CYL,(3,1),RLSE),UNIT=SYSDA
//*SORT FIELDS=(1,2,A),FORMAT=CH
//SYSIN DD *
RECORD:
       KEYS=(1 2 CHAR)
ENDSORT:
```

次に示すのは、MVS JCL の追加例です。コメントの文は、元の文です。

```
//*SYSIN DD *
//*SORT FIELDS=(1,2,A),FORMAT=CH
//*
//SYSIN DD *
RECORD:
       KEYS=(1 2 CHAR)
ENDSORT:
....
//*SYSIN DD *
//*SORT FIELDS=(1,5,A,11,1,A,6,3,A,122,3,A),FORMAT=CH
//*
//SYSIN DD *
RECORD:
       KEYS=(1 5) (11 1) (6 3) (122 3)
ENDSORT:
....
//*SYSIN DD *
//*SORT FIELDS=(1.21,A),FORMAT=CH
//* SUM FIELDS=(36,8,PD,44,8,PD,52,8,PD,60,8,PD,68,8,PD,76,
//* 8,PD,84,8,PD,92,8,PD,100,8,PD)
//*
//SYSIN DD *
RECORD:
       KEYS=(1 21 CHAR)
```

```
       SUM=(36 8 PDEC) (44 8 PDEC) (52 8 PDEC) (60 8 PDEC)
       (68 8 PDEC) (76 8 PDEC) (84 8 PDEC) (92 8 PDEC)
       (100 8 PDEC)
ENDSORT:
....
//*SYSIN DD *
//*SORT FIELDS=(1.21,A),FORMAT=CH
//* SUM FIELDS=(36,8,PD,44,8,PD,52,8,PD,60,8,PD,68,8,PD,76,
//* 8,PD,84,8,PD,92,8,PD,100,8,PD,108,8,PD,116,8,PD,124,8,
//* PD,132,8,PD,140,8,PD,148,8,PD,156,8,PD,164,8,PD,172,
//* 8,PD,180,8,PD)
//*
//SYSIN DD *
RECORD:
       KEYS=(1 21 CHAR)
       SUM=(36 8 PDEC) (44 8 PDEC) (52 8 PDEC) (60 8 PDEC)
       (68 8 PDEC) (76 8 PDEC) (84 8 PDEC) (92 8 PDEC)
       (100 8 PDEC) (108 8 PDEC) (116 8 PDEC) (124 8 PDEC)
       (132 8 PDEC) (140 8 PDEC) (148 8 PDEC) (156 8 PDEC)
       (164 8 PDEC) (172 8 PDEC) (180 8 PDEC)
ENDSORT:
....
//*SYSIN DD *
//*SORT FIELDS=(1.17,A),FORMAT=CH
//*RECORD TYPE=F,LENGTH=350
//*INCLUDE COND=(1,1,CH,EQ,C'A',AND,2,1,CH,EQ,C'R')
//*END
//*
//SYSIN DD *
RECORD:
       INCLUDE=(1 1 EQ 'A') AND (2 1 EQ 'R')
       KEYS=(1 17 CHAR)
ENDSORT:
```

SyncSort の使用法

この節では、Sun MBM 環境で、SyncSort をソートユーティリティーとして使用するときに、MVS JCL ソートステップで行う必要がある変更について説明します。

注 – サブシステムは、SyncSort を実行するように構成されている必要があります。

SyncSort 変換ユーティリティーを使用して、元のソート指示を含む `SYSIN` ファイルを変換する必要があります。このユーティリティーを使うと、IBM SyncSort 構文の指示が UNIX SyncSort 構文に変換されます。変換された `SYSIN` 出力の `infile` および `outfile` 指定を変更する必要があります。

`SYSIN` ファイルがインストリームデータセットの場合は、`/INFILE` と `/OUTFILE` を次のように変更します。

```
/INFILE  \$DD_SORTIN FIXED \$LRECL_SORTIN
/OUTFILE \$DD_SORTOUT OVERWRITE
```

`SYSIN` ファイルがカタログファイルの場合は、`/INFILE` と `/OUTFILE` を次のように変更します。

```
/INFILE  $DD_SORTIN FIXED $LRECL_SORTIN
/OUTFILE $DD_SORTOUT OVERWRITE
```

たとえば、`SORTIN` のレコード形式が実行時に不明な場合は、SyncSort `SYSIN` がコピーブックに提供され、`SYNCFMT_`*dsname* 環境変数を使用できます。上記の例では、`FIXED` が次の値に置き換えられます。

`$SYNCFMT_SORTIN`

SyncSort を使えるようにサブシステムを構成している場合、`SORT` シェルスクリプトで次の行が実行されます。

```
aa=`cat $DD_SYSIN`
syncsort `eval echo $aa`
```

最初の文では、`$DD_SYSIN` の内容を変数 `aa` に割り当てます。2 番目の文では、`$DD_SYSIN` で指定したファイル内のすべての環境変数を展開し、SyncSort モジュールにデータを引数として渡します。

SyncSort の追加情報については、製品のマニュアルを参照してください。

SyncSort 制御文での特殊文字の使用

ドル符号 ($) は、UNIX シェルおよび Sun MBM エンジンで特殊な意味を持つことがあります。したがって、`/derivedfield  d1 "$"` など、SyncSort のパラメータでこの文字を使用する場合は、次のガイドラインに従ってください。

- SyncSort パラメータで `SYSIN` ファイルがディスクファイルの場合、ドル符号文字を維持するには、ドル符号を二重引用符で囲むか、ドル符号の前にエスケープ文字 (バックスラッシュ) を追加する必要があります。たとえば、次のいずれの形式でも可能です。

  ```
  /derivedfield d1 "$"
  ```

  ```
  /derivedfield d1 \$
  ```

- `SYSIN` がインストリームの場合は、ドル符号の前に 3 つのエスケープ文字を付けるか、エスケープしたドル符号を二重引用符で囲む必要があります。たとえば、次のいずれの形式でも可能です。

  ```
  /derivedfield d1 \\\$
  ```

  ```
  /derivedfield d1 "\$"
  ```

プリプロセス/ポストプロセスファイルの実行

プリプロセスファイルまたはポストプロセスファイルを実行する場合は、SyncSort を実行する前に、サブシステムのユーザー設定ファイルに `$SYNCSORT_EXEC` を設定する必要があります。`$SYNCSORT_EXEC` でユーザースクリプトを指定します。次に例を示します。

```
setenv SYNCSORT_EXEC /home/develop/syncsort.sh
```

次に示すのは、すべてのユーザーに対する読み取りおよび実行権が必要な `syncsort.sh` スクリプトの例です。

```
#!/bin/ksh
if [ "a$RECFOR_SORTIN" = "arecordv" ]
then
 cat $DD_SYSIN | while read line
  do
    line1=`eval echo $line`
    set $line1
    if [ $1 = "/INFILE" ]
    then
      FILE=$2; export FILE
#=>>> add logic here for preprocessing data conversion routine,
#=>>> if required
      echo converting sortin file:$FILE
```

```
      if [ $?-ne 0 ]
      then
        echo ERROR converting input file:$FILE
        exit 255
      fi
    fi
 done
fi

aa=`cat $DD_SYSIN`
syncsort `eval echo $aa`
if [ $?-ne 0 ]
then
  echo ERROR executing:syncsort `eval echo $aa`
  exit 255
fi

if [ "a$RECFOR_SORTOUT" = "arecordv" ]
then
  cat $DD_SYSIN | while read line
  do
    line1=`eval echo $line`
    set $line1
    if [ $1 = "/OUTFILE" ]
    then
      FILE=$2; export FILE
#=>>> add logic here for postprocessing data conversion routine,
if reqd
      echo converting sortout file: $2
      if [ $?-ne 0 ]
      then
        echo ERROR converting output file:$FILE
        exit 255
      fi
    fi
  done
fi
exit 0
# end of syncsort.sh script example
```

SyncSort 統計情報のファイルへの保存

たとえば、SyncSort 統計情報をファイルに保存して、別の処理ステップに渡すには、DSN で、SYSOUT DD 文を定義する必要があります。

例:

```
//SYSOUT DD DSN=TEST.STATS,
//     DISP=(NEW,CATLG,DELETE),
//     UNIT=SYSDA,
//     SPACE=(TRK,(5,5),RLSE)
```

変換後の ASSGNDD マクロは次のとおりです。

```
ASSGNDD ddname='SYSOUT' dataset='TEST.STATS' \\\
       filename='/usr/workdir/data/test.stats' disp='o' \\\
       normal='k' abend='d'
```

CoSORT の使用法

CoSORT を使用するには、ソート制御 SYSIN ファイルを適切な sortcl 構文に変換する必要があります。CoSORT 製品が提供した UNIX 変換ツールにメインフレーム JCL のいずれかを使用します。SORTIN ASSGNDD 文で recsize パラメータを指定する場合は、/LENGTH パラメータの変数を使用できます。次に例を示します。

```
ASSGNDD ddname=SORTIN recsize=143 ...
```

パラメータの値が次の場合

```
/LENGTH 143
```

上記の値を次のように置換できます。

```
/LENGTH $LRECL_SORTIN
```

ソートオプションの標準セットを使用している場合、サブシステムのユーザー設定ファイルの SORTOPTS 環境変数にこれらのオプションを設定できます。sortcl を呼び出すと、$SORTOPTS に定義されたオプションが使用されます。

注 – サブシステムは、CoSORT を実行するように構成されている必要があります。

CoSORT の追加情報については、製品のマニュアルを参照してください。

プリプロセス/ポストプロセスファイルの実行

プリプロセスロジックまたはポストプロセスロジックを実行する必要がある場合は、ユーザーが定義したスクリプト内で sortcl の呼び出しをカプセル化できます。サブシステムのユーザー設定ファイルで $COSORT_EXEC を設定してから、CoSORT を実行する必要があります。$COSORT_EXEC で、sortcl を内部で呼び出すユーザースクリプトを指定します。次の例に、ユーザースクリプトを示します。

```
#!/bin/ksh
rm /prod/copy1
SORT
cp /prod/sortout /prod/sortout_back
```

ベース名 GDG ファイルのソート

Sun MBM では、複数の DD_SORTIN*nn* 環境変数を設定して、ベース名 GDG のソート機能をサポートします。

例

次の SORTIN 文があるとします。

```
//SORTIN  DD  DSN=A.B.C.D,DISP=(OLD,DELETE,KEEP)
```

これに、次の File_Map エントリが対応します。

```
A.B.C.D;MASTERCAT;FS;/tmp/data/abcd;;3;
```

Sun MBM では、$DD_SORTIN00、$DD_SORTIN01、および $DD_SORTIN02 環境変数を設定します。ユーザーは、変換された SYSIN 文を CoSORT または SyncSort 用に変更する必要があります。

SyncSort 用の SYSIN 文は次のようになります。

```
/INFILE \$DD_SORTIN00 ........
/INFILE \$DD_SORTIN01 ........
/INFILE \$DD_SORTIN02 ........
```

この例では、Sun MBM は、これらの環境変数をファイルの 3 つの最新 (カタログ化または非カタログ化された) GDG オカレンスに設定します。

IEBGENER ユーティリティー

Sun MBM では、単一または複数の出力用に IEBGENER ユーティリティーのファイルのコピー機能と再ブロック化機能をサポートします。複数の出力は、コピーされたファイルと再ブロック化されたファイルの組み合わせの場合もあります。再ブロック化とは、入力 (SYSUT1) ファイル用に指定したレコードサイズを基に、異なるレコードサイズを持つ固定長の出力レコードを作成することです。

SYSUT1 入力ファイルは、単一のファイルの場合も、複数の連結ファイルで構成されている場合もあります。データを SYSUT1 から SYSUT*n* に再ブロック化する場合は、次の要件を満たす必要があります。

- SYSUT1 は、固定長レコード形式のファイルでなければなりません。
- SYSUT1 が連結されている場合、各ファイルは同じレコードサイズを指定します。連結された SYSUT1 ファイルのいずれかに対して複数のレコードサイズが指定されている場合は、最後に検出されたものが適用されます。
- SYSUT1 および SYSUT*n* は、それぞれに対応する ASSGNDD マクロ文で指定したレコードサイズとレコード形式を持つ必要があります。
- SYSUT1 ファイルおよび SYSUT2 ファイルが定義されている必要があります。

いずれかの要件を満たさない場合、IEBGENER ユーティリティーは、デフォルトで SYSUT1 ファイルを SYSUT*n* にコピーします。

レコードサイズとレコード形式の情報は、以下のいずれかの方法で提供できます。

- Sun MBM トランスレータを使用して、LRECL パラメータと RECFM=F (または FB) パラメータを含む元のメインフレーム JCL 文を変換する。
- File_Map の 5 番目のファイルエントリ用フィールドに属性ファイル名を指定し、Sun MBM トランスレータを使用して元のメインフレーム JCL を変換する。
- ASSGNDD マクロを手動で更新し、recsize パラメータと recfmt パラメータを含める。

次に、入力レコードサイズと出力レコードサイズに基づくアクションを説明します。

- 入力レコードサイズが出力レコードサイズを下回る場合、入力レコードは、左から右に空白を埋める形で出力レコードにコピーされます。
- 入力レコードサイズが出力レコードサイズを上回る場合、入力レコードは、出力レコードサイズになるまで左から右という形で出力レコードにコピーされます。この結果、入力データは切り捨てられます。
- 入力レコードサイズが出力レコードサイズと同じ場合、入力レコードは出力レコードにコピーされます。

例 1: 次の例では、SYSUT1 が SYSUT2 に再ブロック化されます。その際、各 SYSUT2 レコードの右端に 2 つの空白文字が埋め込まれます。

```
ASSGNDD ddname='SYSUT1' dataset='IN1' filename='\$SEQDATA/input5.record' \\\
    disp='i-o' recsize='5' recfmt='F'
ASSGNDD ddname='SYSUT2' dataset='IN2' filename='\$SEQDATA/gener.out2' \\\
    disp='i-o' recsize='7' recfmt='F'
EXECPGM pgmname='IEBGENER' stepname='STEP01'
```

例 2: 次の例では、SYSUT1 が SYSUT2 にコピーされます。これは、レコードサイズパラメータ (recsize) もレコード形式パラメータ (recfmt) も SYSUT1 に指定されていないためです。

```
ASSGNDD ddname='SYSUT1' dataset='IN1' filename='\$SEQDATA/input5.record' \\\
    disp='i-o' recsize='5'
ASSGNDD ddname='SYSUT2' dataset='IN2' filename='\$SEQDATA/gener.out2' \\\
    disp='i-o' recsize='7' recfmt='F'
EXECPGM pgmname='IEBGENER' stepname='STEP01'
```

例 3: 次の例では、SYSUT1 が SYSUT2 に再ブロック化されます。その際、SYSUT2 出力レコードを基に、各 SYSUT1 レコードの最後の 3 文字が切り捨てられます。

```
ASSGNDD ddname='SYSUT1' dataset='IN1' filename='\$SEQDATA/input5.record' \\\
    disp='i-o' recsize='10' recfmt='F'
ASSGNDD ddname='SYSUT2' dataset='IN2' filename='\$SEQDATA/gener.out2' \\\
    disp='i-o' recsize='7' recfmt='F'
EXECPGM pgmname='IEBGENER' stepname='STEP01'
```

例 4: 次の例では、SYSUT1 が SYSUT2 に再ブロック化され、SYSUT1 が SYSUT3 にコピーされます。これは、レコードサイズ情報もレコード形式情報も SYSUT3 に含まれていないためです。

```
ASSGNDD ddname='SYSUT1' dataset='IN1' filename='\$SEQDATA/input5.record' \\\
    disp='i-o' recsize='10' recfmt='F'
ASSGNDD ddname='SYSUT2' dataset='OUT2' filename='\$SEQDATA/gener.out2' \\\
    disp='i-o' recsize='7' recfmt='F'
ASSGNDD ddname='SYSUT3' dataset='OUT3' filename='\$SEQDATA/gener.out3' \\\
    disp='i-o'
EXECPGM pgmname='IEBGENER' stepname='STEP01'
```

例 5: 次の例では、連結された SYSUT1 が単一の SYSUT2 に再ブロック化されます。その際、各 SYSUT2 レコードの末尾に 2 つの空白文字が埋め込まれます。

```
ASSGNDD ddname='SYSUT1' dataset='IN1' filename='\$FMROOT/input5.record' \\\
    disp='i-o' recsize='5' recfmt='F'
ASSGNDD                 dataset='INX' filename='\$FMROOT/input5.record2' \\\
    disp='i-o' recsize='5' recfmt='F'
ASSGNDD ddname='SYSUT2' dataset='IN2' filename='\$FMROOT/in2' disp='i-o' \\\
    recsize='7' recfmt='F'
EXECPGM pgmname='IEBGENER' stepname='STEP01'
```

IEBUPDTE ユーティリティー

Sun MBM は、IEBUPDTE ユーティリティーの次の機能をサポートします。

```
./[label] ADD {MEMBER=<memname>} {NAME=<name>}
data
.
.
.
```

その他の関数およびこのユーティリティーの詳細な文は、サポートされません。Sun MBM では警告が出力されます。

IEBEDIT ユーティリティー

Sun MBM では、次の 2 つの IEBEDIT 機能をサポートします。

- SYSUT1 DD 文で定義したデータを SYSUT2 DD 文で定義したファイルにコピーします。
- SYSUT2 が SYSOUT データセットで、書き込み側が 'INTRDR' として定義されている場合に実行するように SYSUT1 で定義したジョブストリームをサブミットします。現在のジョブストリームは実行を継続します。

IEFBR14 ユーティリティー

IEFBR14 では、すべてのデータセットの設定を実行します。データセットは、設定で指定したように割り当てられ、初期化されます。

JCL 検査はサポートされません。JCL ストリームを検査するには、「Submit Job to Subsystem」画面で「Validate」オプションを使って、ジョブをサブミットする必要があります。詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

IEBCOPY ユーティリティー

Sun MBM は、IEBCOPY ユーティリティーの次の機能を部分的にサポートします。

- SYSUT1 および SYSUT2 ddnames で指定されたデータセットをコピーします。
- SYSIN で指定された COPY 関数

SELECT および EXCLUDE オプションはサポートされません。

ユーザーユーティリティー

ユーザーユーティリティーを使用すると、ユーザー固有の関数をマクロ実行内でカプセル化できます。

ユーザーユーティリティーは、ファイル status.${JON} に復帰状態の数値を割り当てる必要があります。状態コードは 0 ～ 255 です。

たとえば、myfile というファイルの有無によって、ユーザーユーティリティーがゼロまたはゼロ以外の値を返す場合、UNIX シェルスクリプトによる終了状態の設定は次のようになります。

```
...
if(-f myfile)
then
  echo 0 > status.${JON}
else
  echo 1 > status.${JON}
fi
...
```

ゼロの値は、ユーティリティーが正常に終了したことを示します。ゼロ以外の値は、ユーティリティーが異常終了したため、ジョブステップが中止されることを示します。

EXECPGM マクロは、ユーザーユーティリティーを呼び出す前に COND-CODE を 255 の値で初期化します。status.${JON} ファイルの値が正常終了でゼロに設定されない場合、ステップは初期値 (255) のために中止されます。

status.${JON} ファイルで設定された値は、ステップの COND-CODE 値になります。

第8章

アプリケーションプログラムの移行

この章では、Sun MBM で COBOL、C/C++、Java、および PL/I プログラムを使用する方法について説明します。この章の内容は、次のとおりです。

COBOL アプリケーションプログラム

この節では、実行環境を作成する方法、および COBOL プログラムを準備し、Sun MBM の制御下で実行する方法を説明します。どのベンダーの COBOL を使用するかによって、手順の一部は異なります。現在の環境に適した方法を用いてください。

▼ COBOL 実行時システムを構成する

1. BAM を起動します。
2. 「BAM」メニューで、オプション「3 Applications & Subsystems」を選択します。
3. 「Applications & Subsystems」メニューで、オプション「3 Create a Subsystem」を選択します。

4. **作成するサブシステムの名前を入力してから Return キーを押します。**

5. **COBOL ベンダーを選択します。**

   - デフォルトの COBOL ベンダーである Micro Focus Server Express を使用する場合は手順 7 に進みます。
   - ACUCOBOL-GT® を使用する場合は、「Create」メニューでオプション「1 Application Languages」を選択し、手順 6 に進みます。

6. **図 8-1 の「Application Languages」画面で、ACUCOBOL-GT 用のオプション「2」を選択して、Return キーを押します。**

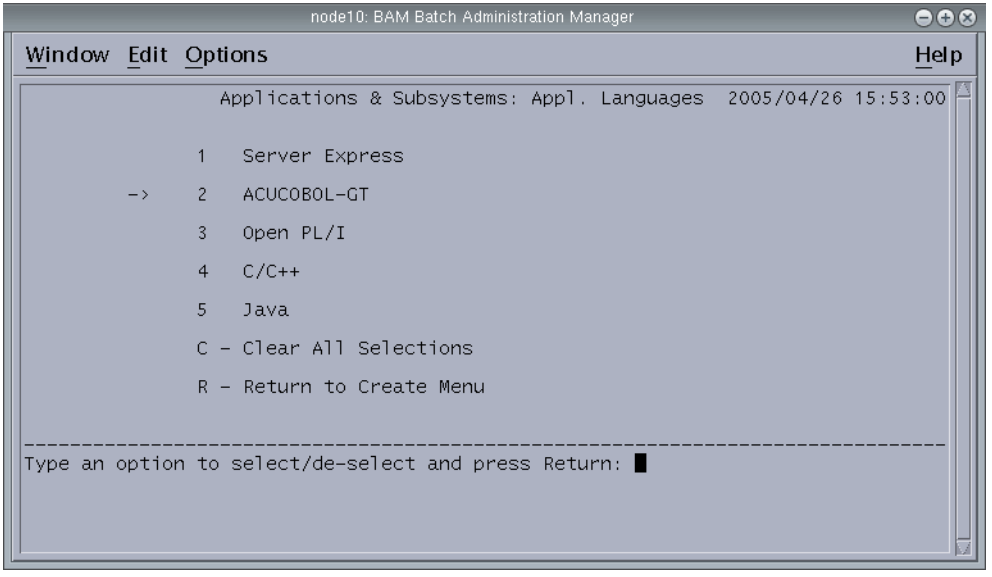


図 8-1　　COBOL の選択

7. **使用するデータファイルのタイプを選択します。**

   - Sun MTP 領域と連動していないサブシステムを構築する場合は、VSAM データ管理がデフォルトではないので、手順 9 に進みます。
   - Sun MTP VSAM ファイルにアクセスするサブシステムを構築する場合は、「Create」メニューでオプション「3 Data Management」を選択し、手順 8 に進みます。

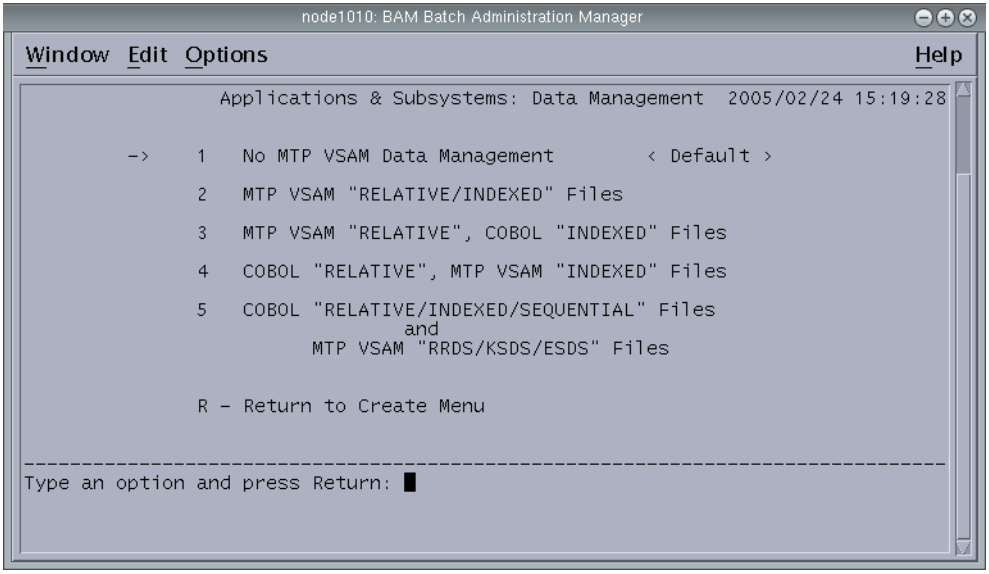


図 8-2　Data Management メニュー

8. **オプションをどれか 1 つ選択して、Return キーを押します。**
9. **続けてその他のサブシステムオプションを選択します。**
10. **サブシステムを構築します。**

    サブシステムと同時に COBOL 実行時システムが作成されます。

    サブシステムの作成についての詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

COBOL アプリケーションプログラムのコンパイル

この節では、Sun MBM の制御下でバッチジョブから実行される COBOL アプリケーションプログラムをコンパイルする方法を説明します。ここでは、VSAM、COBOL、GDG、および連結ファイルタイプのコンパイルオプションについても説明します。

▼ COBOL プログラムをコンパイルする

1. **サブシステムの環境を設定する方法は、次のとおりです。**
   - ソース `batchenv` でサブシステム名を指定します。次に例を示します。

```
$ . $EBMHOME/batchenv cblsys
```

   - 次の手順に従います。

   a. Sun MBM メインメニューを表示します。

   b. メインメニューの「Command Prompt」アイコンをクリックし、サブシステム名を入力します。

2. **COBOL プログラムが RDBMS にアクセスしていて、SQL 文が含まれる場合は、適切な RDBMS プリコンパイラにより元の COBOL プログラムソースを変換して、COBOL コンパイルに必要なソースファイルを生成します。**

   たとえば、Oracle に `procob` プリコンパイラを使用します。

3. **COBOL ソースをコンパイルして実行可能にします。**
   - 32 ビット Server Express (デフォルト) を使用する場合は、`cob` コンパイラを使用します。
   - ACUCOBOL-GT を使用する場合、`ccbl` コンパイラを使用します。

   手順 2 を実行すると、入力ソースファイルがプリコンパイラから出力されます。実行しない場合は、標準プログラムソースが使用されます。

GDG または連結データセットのアクセスに必要なコンパイルオプション

複数のオカレンスを含むベース名 GDG や連結データセットは複数の物理ファイルに割り当てられます。Sun MBM には、`ebmlsfile` というファイルハンドラが用意されています。このファイルハンドラを使うと、ベース名 GDG または連結データセットのすべてのデータにアクセスできます。

Server Express:

次のコンパイルオプションを指定して、ベース名 GDG または連結データセットにアクセスするすべての COBOL プログラムをコンパイルする必要があります。

```
-C "callfh=ebmlsfile"
```

このオプションは、COBOL アプリケーションプログラムで実行される I/O 呼び出しをインターセプトして、制御を `ebmlsfile` ファイルハンドラに引き渡すように COBOL に指示します。

ACUCOBOL-GT:

コンパイル時に特別なオプションは必要ありません。Sun MBM では、ACUCOBOL-GT でサブシステムを構築するときに、`ebmlsfile` で自動的にファイルが処理されます。

制御が渡されると、`ebmlsfile` では、Sun MBM で設定したファイル名環境変数から、現在のファイルをベース名 GDG または連結データセットとして特定します。次に、確実にすべてのデータにアクセスするために適切な動作を行います。詳細は、253 ページの「連結データセットとベース名 GDG へのアクセス」も参照してください。

COBOL オプションの例は、`$PUBLIC/bin/cblmake.sh` スクリプトにあります。

VSAM および COBOL ファイルのアクセスに必要なコンパイルオプション

1 つの COBOL アプリケーションプログラムで、COBOL ファイルと Sun MTP VSAM ファイルにアクセスできます。これらのファイル構成は同じです。`ebmlsfile` ファイルハンドラは、これらのファイルへのアクセス要件を持つプログラムをサポートしています。次にアクセス可能な例を示します。

- Sun MTP VSAM 索引編成ファイルと COBOL 索引編成ファイル
- Sun MTP VSAM 相対編成ファイルと COBOL 相対編成ファイル
- Sun MTP ESDS 順編成ファイルと COBOL 順編成ファイル

Server Express:

次のコンパイラオプションを指定して、VSAM ファイルと COBOL ファイルの両方にアクセスするすべての COBOL アプリケーションプログラムをコンパイルする必要があります。

```
-C "callfh=ebmlsfile"
```

このオプションは、COBOL アプリケーションプログラムで実行される I/O 呼び出しをインターセプトして、制御を `ebmlsfile` ファイルハンドラに引き渡すように COBOL に指示します。

ACUCOBOL-GT:

コンパイル時に特別なオプションは必要ありません。Sun MBM では、ACUCOBOL-GT でサブシステムを構築するときに、`ebmlsfile` で自動的にファイルが処理されます。

制御が渡されると、`ebmlsfile` では、アクセスするファイルタイプに適した I/O ルーチンを呼び出します。

Sun MBM 環境で VSAM ファイルと COBOL ファイルの両方にアクセスする COBOL アプリケーションプログラムには、次の制限事項があります。

- COBOL 内部に含まれる SORT 文を変更する必要があります。この文は、VSAM ファイルのアクセスに USING または GIVING のいずれか、あるいはその両方の句を指定して INPUT または OUTPUT PROCEDURE のいずれか、あるいはその両方の句を使用します。
- 連結データセットまたは GDG として VSAM ファイル (ESDS/KSDS/RRDS) にアクセスできません。
- ファイルの SELECT 文で ORGANIZATION IS RECORD SEQUENTIAL 句を使って、VSAM ESDS ファイルにアクセスする必要があります。VSAM ESDS ファイルに ORGANIZATION LINE SEQUENTIAL 句を指定すると、OPEN 文はエラー 35 により失敗します。

移行した COBOL アプリケーションプログラムからのファイルのアクセスに必要なコンパイルオプション

IBM JCL をジョブマクロスクリプトに変換すると、COBOL アプリケーションプログラムなどのステップ実行可能ファイルでアクセスするために JCL に割り当てられた各ファイルのスクリプトに、環境変数が設定されます。環境変数の値は、Sun MBM 環境でのファイルの場所です。

COBOL プログラムは、SELECT 文の ASSIGN 句を使用して、ファイルの物理位置にアクセスし、ENVIRONMENT DIVISION のファイル識別子を取得します。ジョブマクロスクリプトに設定した環境変数から、COBOL プログラムで確実にファイルの物理位置にアクセスするには、移行されたすべての COBOL アプリケーションプログラムに対して、次のコンパイラオプションのいずれか 1 つを使用する必要があります。

Server Express:

```
-C "assign=EXTERNAL"
```

ACUCOBOL-GT:

```
"--fileAssign=external"
```

このオプションは、すべての SELECT 文の ASSIGN 句に使用したファイル識別子が、デフォルトで、アクセスされる位置を解決する環境変数であることを示します。詳細および例については、277 ページの「アプリケーションプログラムへのファイルの割り当て」を参照してください。

ファイルがジョブから外部的に (通常は ASSGNDD マクロを使用して) 割り当てられておらず、COBOL プログラムで OUTPUT として開かれた場合、ファイルが関連付けられていないという警告が表示され、内部ファイル名が付けられたファイルが現在の作業用ディレクトリに作成されます。

COBOL アプリケーションプログラムの実行

ジョブに代わって実行される COBOL アプリケーションプログラムは、実行時システムの制御下で開始されます。

実行時システムには、cobwrap.cbl というメインの COBOL プログラムがあります。cobwrap.cbl プログラムは $PACK/RTSFS ディレクトリまたは Sun MTP の $UNIKIX/src/RTSVSAM ディレクトリにあります。

メインの COBOL プログラムでは、アプリケーションプログラムの終了ハンドラをインストールします。1 つは正常終了、もう 1 つは異常終了を処理します。そのあと、アプリケーションプログラムを実行して、指定した引数またはプログラムをマクロジョブに渡します。プログラムが終了すると、呼び出された終了ハンドラに基づいて終了状態が設定されます。

COBOL プログラムの検索規則

COBOL プログラムは、EXECPGM マクロを使って実行します。これらのプログラムは、以下の場所のいずれかに配置する必要があります。

- SYS1.LINKLIB: プログラムのデフォルトディレクトリ。サブシステムの File_Map で定義されます。複数のディレクトリをコロンで区切って指定することもできます。
- JOBLIB: ジョブ内の ASSGNDD マクロで定義されます。
- STEPLIB: ジョブステップ内の ASSGNDD マクロで定義されます。

システムでは、次の検索規則を使用してプログラムを検索します。

1. STEPLIB DD 文を使用してジョブ内の単独のジョブステップに対する専用ライブラリを定義する場合、システムは最初にプログラムの STEPLIB ディレクトリを検索します。プログラムが見つからない場合は、SYS1.LINKLIB ディレクトリが検索されます。JOBLIB DD 文がジョブ内にある場合は、無視されます。
2. JOBLIB DD 文があり、ジョブステップに STEPLIB DD 文がない場合、JOBLIB ディレクトリが最初に検索されます。プログラムが見つからない場合は、SYS1.LINKLIB ディレクトリが検索されます。
3. JOBLIB DD 文も STEPLIB DD 文もない場合は、SYS1.LINKLIB ディレクトリが検索されます。

ジョブステップリターンコード

COBOL アプリケーションプログラムを実行するジョブステップには、COBOL プログラムの RETURN-CODE 特殊レジスタの最終値から設定された独自のジョブステップリターンコードセットが含まれます。プログラムが正常に終了すると、Sun MBM ジョブで、後続のジョブステップ処理の条件付きロジックに使用できるジョブステップリターンコードが作成されます。条件付きロジックは、MVS JCL では EXEC 文の COND パラメータから生成され、VSE JCL で $RC パラメータを使用する場合は VSE IF 文から生成されます。

メモリー障害または 0 で除算などでプログラムが中止した場合、0 以外のジョブステップ完了コードを作成してプログラムは終了します。ここで、ジョブステップリターンコード RETURN-CODE は未定義なので、あとのジョブステップ処理の判定に使用されません。

特定の状況では、ジョブステップで正常に終了するプログラムの RETURN-CODE 値が Sun MBM に正しい値を返さないことがあります。このため、条件付き論理でテストされる場合は、あとのジョブステップ処理が中断されます。これは、COBOL プログラムが STOP RUN 文により終了する場合に起こります。この場合、Sun MBM は RETURN-CODE 値を継承しません。この問題を修正するには、STOP RUN 文を GOBACK 文に変更するか、または STOP RUN 文の前にこの呼び出しを追加します。

```
MOVE RETURN-CODE to CCF-RETURN-CODE
CALL "CCF_SET_RETCODE" USING CCF-RETURN-CODE
STOP RUN
```

作業記憶領域に CCF-RETURN-CODE を定義する必要があります。

```
01  CCF-RETURN-CODE  PIC S9(4) COMP.
```

注 – STOP RUN 文の前にこの呼び出しを発行しない場合、リターンコードは 0 にデフォルト設定されます。

省略名を使って CCF_SET_*XXXX* COBOL ルーチンを呼び出すこともできます。

CCF_SET_CONDCODE	CCFSETC
CCF_SET_RETCODE	CCFSETR
CCF_SET_DUMP_AREA	CCFSETD

COBOL ACCEPT 文

COBOL アプリケーションプログラムでオペレータの応答を要求する場合、Sun MBM 関数呼び出し EBM_GET_REPLY をプログラムに含めることができます。この呼び出しで、Sun MBM コンソールにメッセージが表示され、オペレータが応答するまで COBOL の実行は中断されます。オペレータが応答したあと、COBOL プログラムデータ領域にメッセージが返され、実行は再開されます。

例

次の 2 つの引数を指定して、EBM_GET_REPLY 関数を呼び出す必要があります。

1. 最初の引数 MESSAGE-AREA を PIC X(80) として定義します。これには、Sun MBM コンソールに送信するメッセージを含める必要があります。
2. 2 番目の引数 ACCEPT-AREA を PIC X(*nn*) として定義します。これには、オペレータが送信するメッセージを含めるのに十分な領域を指定します。

オペレータは、「Active Jobs」画面または rpljob コマンドを使って、プログラムに応答できます。「Active Jobs」画面を使って待機中のジョブに応答する方法と rpljob コマンドについては、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

コード例 8-1　　コンソールからのオペレータ入力の受け入れ

```
WORKING-STORAGE SECTION.
01 MESSAGE-AREA   PIC X(80) VALUE 'PLEASE PRINT THE TAPE ID'.
01 ACCEPT-AREA    PIC X(80).
PROCEDURE DIVISION.
BEGIN.
CALL "EBM_GET_REPLY" USING MESSAGE-AREA,ACCEPT-AREA.
DISPLAY 'TAPE ID RECEIVED:' ACCEPT-AREA.
```

特定メッセージへの自動応答

COBOL プログラムで Sun MBM コンソールに送信できる有効なメッセージのリストを含む応答テーブルを構成できます。このテーブルに定義したメッセージについては、システムでは COBOL の実行を中断しませんが、テーブルに定義された、対応する応答メッセージで自動的に応答します。

COBOL プログラムが応答テーブルのメッセージを検索できない場合、オペレータがメッセージに応答するまで、実行は中断されます。

応答テーブル内のメッセージの検索は、$USER_SETUP で指定したサブシステム設定ファイルに REPLY_TABLE 環境変数を設定すると有効になります。$REPLY_TABLE には、応答テーブルのフルパスおよびファイル名を指定する必要があります。次に例を示します。

```
setenv REPLY_TABLE $EBMHOME/samples/y2k/reply_table
```

応答テーブルの各行の形式は、次のとおりです。

```
message to the console=>reply message to the COBOL application program
```

例

この応答テーブルには、次の 3 つのメッセージが含まれます。

```
PLEASE PRINT THE CONTROL NUMBERS=>123 456 789
PLEASE PRINT THE TAPE ID=>TD001
PLEASE PRINT THE BALANCE TOTALS=>8888 2222
```

COBOL プログラムで次のメッセージをコンソールに送信するとします。

```
PLEASE PRINT THE TAPE ID
```

Sun MBM では、自動的に次の応答メッセージをプログラムに返します。

```
TD001
```

COBOL アプリケーションプログラム終了ハンドラ

ジョブステップで COBOL プログラムの実行に使用される Sun MBM COBOL 実行時システムでは、プログラムを実行する前に、次の 2 つの終了ハンドラをインストールします。

- CCFexitwrap
- CCFerrorwrap

どちらの終了ハンドラが呼び出されるかは、COBOL プログラムの終了方法によって決まります。終了ハンドラの機能は、COBOL プログラムの終了時に、ジョブステップリターンコードとジョブステップ完了コードを設定することです。次の節では、プログラム終了の例、およびジョブステップリターンコードとジョブステップ完了コードの設定方法について説明します。

GOBACK 文と EXIT PROGRAM 文

GOBACK 文または EXIT PROGRAM 文によって COBOL プログラムが正常に終了した場合は (COBOL RETURN-CODE 変数が 0 以外の場合でも)、インストールされている終了ハンドラは呼び出されません。

終了コードセットは、次のとおりです。

ジョブステップリターンコード	アプリケーションプログラムで、RETURN-CODE の最後の値に設定されます。
ジョブステップ完了コード	0 に設定されます。

データベースは常にコミットされます。

必要に応じて、COBOL プログラムで GOBACK 文を発行する前に、RETURN-CODE 変数を設定する必要があります。次に例を示します。

```
MOVE 20 TO RETURN-CODE
```

STOP RUN 文

STOP RUN 文によって COBOL プログラムが正常に終了した場合 (RETURN-CODE 変数が 0 以外の場合も)、CCFexitwrap 終了ハンドラが呼び出されます。

CCFexitwrap によって、次の終了コードセットが設定されます。

ジョブステップリターンコード	Sun MBM CCF_SET_RETCODE ルーチンが STOP RUN 文が発行される前に呼び出された場合は、RETURN-CODE の値に設定されます。そうでない場合は 0 に設定されます。
ジョブステップ完了コード	0 に設定されます。

データベースは常にコミットされます。Server Express を使用する場合は cobtidy() ルーチン、ACUCOBOL-GT を使用する場合は acu_shutdown() ルーチンで COBOL ファイルがフラッシュされます。詳細は、COBOL のマニュアルを参照してください。

RETURN-CODE 変数の値がジョブステップリターンコードとして必要な場合、STOP RUN 文を GOBACK 文に変更するか、または STOP RUN 文の前に、次の呼び出しを追加します。

```
MOVE RETURN-CODE to CCF-RETURN-CODE
CALL "CCF_SET_RETCODE" USING CCF-RETURN-CODE
STOP RUN
```

作業記憶領域に CCF-RETURN-CODE を定義する必要があります。

```
01     CCF-RETURN-CODE     PIC S9(4) COMP.
```

注 – STOP RUN 文の前にこの呼び出しを発行しない場合、RETURN-CODE は 0 にデフォルト設定されます。

詳細は、238 ページの「ジョブステップリターンコード」も参照してください。

COBOL アプリケーションプログラムの強制的な中止

COBOL アプリケーションプログラムが強制的に中止されると、CCFerrorwrap 終了ハンドラが呼び出され、データベースがロールバックされます。VSAM および RDBMS ファイルのみがロールバックされます。

CCFerrorwrap では、終了コードは次のように設定されます。

ジョブステップリターンコード	未定義。
ジョブステップ完了コード	0 以外の値に設定されます。

COBOL アプリケーションプログラムの強制終了

COBOL アプリケーションプログラムを強制終了するには、2 通りの方法があります。最初の方法は Sun MBM に固有のソリューションではありませんが、2 番目の方法は Sun MBM に固有の機能です。

方法 1

- 0 で 0 を割る COBOL アプリケーションプログラムを作成します。
- 実行中止が必要になるたびにこの COBOL アプリケーションプログラムを呼び出します。
- COBOL CHECKDIV フラグを COBOL アプリケーションプログラムに設定します。

注 – この方法は、Micro Focus Server Express 実行環境でのみ有効です。

```
 $SET CHECKDIV"OSVS"
       IDENTIFICATION DIVISION
       PROGRAM-ID. ABTCBL.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 A          PIC 9(4)   VALUE ZERO.
       01 B          PIC 9(2)   VALUE 1.
       01 REM        PIC 9(2)   VALUE ZERO.
       88 ERROR-COND            VALUE 1.
      *=================================================
       PROCEDURE DIVISION.
       DIVIDE B BY A GIVING REM.
       GOBACK.
```

この方法を使用すると、ジョブステップ完了コードは 255 に設定され、データベースはロールバックされて、あとのジョブステップ処理に影響します。最後のジョブ完了コードは 0 以外の値になり、ジョブが強制的に中止されたことを示します。

方法 2

Sun MBM 関数 CCF_SET_CONDCODE を呼び出します。この関数では、COBOL アプリケーションプログラムの実行を中止する理由によって、特定のジョブステップ完了コード値を設定できます。この関数を使用すると、ジョブステップ完了コードは指定した値に設定され、データベースはロールバックされて、あとのジョブステップ処理に影響します。最後のジョブ完了コードは 0 以外の値になり、ジョブが強制的に中止されたことを示します。

たとえば、作業記憶領域セクションに次のように変数を割り当てます。

```
WORKING-STORAGE SECTION.
.....
01 CCF-COND-CODE    PIC S9(4) comp.
.....
```

実行を中止するには、次のようにします。

- 変数に 1 ～ 255 の値を設定します
- 変数を引き渡す CCF_SET_CONDCODE 関数を呼び出します
- 次のように STOP RUN 文を呼び出します

```
.....
MOVE 100 to CCF-COND-CODE.
CALL "CCF_SET_CONDCODE" USING CCF-COND-CODE.
STOP RUN.
.....
```

注 – CCF_SET_CONDCODE 関数を有効にするには、STOP RUN 文を実行する必要があります。

前の例で指定したように文を実行すると、条件コードが 100 に設定され、データベースはロールバックされて、ジョブは中止されます。

次の例は、lststs コマンドの出力での COBOL プログラムの中止を示しています。

```
starting STEP -- STEP01
IP4104(I) Start CB001 at 14:26:07
rtsfs:RETURN-CODE=0, COND-CODE=100
rtsfs:user exit rollback function executed
IP4105(I) End of CB001 at 14:27:07
STEP1 step was executed cond code 100
step STEP01 start at TIME=02:27:06 PM
step STEP01 stop at TIME=02:27:07 PM
step STEP10 is bypassed
step STEP20 is bypassed
step STEP30 is bypassed
No files to be printed
JOB TEST01 start at TIME=02:26:05 PM
JOB TEST01 stop at TIME=02:27:13 PM
IP4303(S) Job TEST01 aborting at 14:27:13
```

Sun MBM COBOL プログラムダンプ機能の使用

Sun MBM には、COBOL プログラムに障害があった場合に 1 つまたは複数の COBOL データ領域をダンプできる COBOL "CCF_SET_DUMP_AREA" 関数が用意されています。この関数には、次のパラメータが必要です。

- 一意の領域文字列識別子
- データ領域の開始アドレス (*Start-Addr*)
- データ領域の終了アドレス (*End-Addr*)

すべてのサブプログラムも含めて、COBOL プログラムでは、"CCF_SET_DUMP_AREA" 関数を複数回呼び出してダンプするさまざまなデータ領域を定義できます。

メインプログラムで WORKING-STORAGE SECTION を定義します。このプログラムのすべてのサブプログラムにより、他のサブプログラムに固有の領域、つまり WORKING-STORAGE SECTION および LINKAGE SECTION 領域を定義します。

CCF_SET_CONDCODE 関数を使ってプログラムが中止した場合は、ダンプも作成されます。

関数が呼び出されるたびに、Sun MBM では、領域がすでに定義されていないか判定するために検索します。定義されていない場合は、内部テーブルに保存されます。プログラム障害時に、プログラム実行中に定義されたすべての COBOL データ領域がダンプされます。

Start-Addr および *End-Addr* は、どちらも同じ COBOL データ領域内に常駐している必要があります。たとえば、`WORKING-STORAGE SECTION` の開始と終了、または `LINKAGE SECTION` の開始と終了の範囲内です。セクションの範囲を超えることはできません。

注 – `PROCEDURE DIVISION` セクションからアクセスする `LINKAGE SECTION` だけを登録できます。

デフォルトでは、ダンプ出力はジョブ履歴ファイルに生成されます。履歴ファイルの内容の表示についての詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』の `lststs` および `histprt` コマンドの説明を参照してください。

別のファイルにダンプ出力を作成するには、ダンプ出力ファイルを書き込むディレクトリをサブシステム設定ファイルの `EBM_DUMP_DIR` 環境変数に設定します。

例

ジョブ履歴ファイルに、ダンプ出力を経路指定した場所を示すメッセージが表示されます。

```
Creating dump file dump-directory/EBMdump.jobname.stepname.programname.jobnumber
```

COBOL プログラムで使用する呼び出しの形式は、次のとおりです。

```
CALL "CCF_SET_DUMP_AREA" USING AREA-ID, Start-Addr, End-Addr
```

説明

`AREA-ID`	`PIC X(16)`: ダンプする特定のデータ領域に関連する文字列。
Start-Addr	ダンプするデータ領域の開始アドレス。
End-Addr	ダンプするデータ領域の終了アドレス。

例

コード例 8-2 COBOL "CCF_SET_DUMP_AREA" 呼び出しの例

```
WORKING-STORAGE SECTION.
       01 STATUS-OUTFIL   PIC XX VALUE “00”.
       01 CONDCODE      PIC S9(4) COMP.
       01 COUNTREC        PIC S9(4) COMP.
       01 END-WS         PIC X(1).
       01 AREA-ID         PIC X(16) VALUE ‘-CBP5A-WS-’.
PROCEDURE DIVISION.
   CALL “CCF_SET_DUMP_AREA” USING AREA-ID, STATUS-OUTFIL, END-WS.
...
...
```

注 – 実際の値が 16 文字より少ない場合でも、常に AREA-ID を PIC X(16) 文字列として割り当てる必要があります。

次は、COBOL ダンプ出力形式の例です。

コード例 8-3 Sun MBM COBOL ダンプ出力ファイルの例

```
Wednesday May 04 15:44:38 2005 Creating dump file for:
************************
**  JOB  dump5
**  STEP DUMP5
**  PGM  CBDUMP5
**  JON  259
************************
Dumping    Area=-     CBP5A-WS-, Start-Addr=400b9978, End-Addr=400b9990, Size=24
-------------------------------------------------------------------------------
400b9978  00000000  30300000 00000000  00000000
 00000000      *00.............*
400b9988  00000010  03E10000
00000000                          *...............*
-------------------------------------------------------------------------------
```

注 – ダンプする領域のサイズが定義された領域よりも大きい場合は、COBOL マニュアルのデータ整列オプションの説明を参照してください。

コード例 8-4 は、プログラム CBDUMP5 のダンプ出力の例です。ここでは、WORKING-STORAGE SECTION を設定し、各レコードに CBDUMP5A を呼び出して、引数としてレコードを渡します。プログラム CBDUMP5A は、WORKING-STORAGE SECTION と LINKAGE SECTION を設定します。メモリーの境界を越える項目にアクセスを試みた場合、プログラムは中止します。

コード例 8-4 COBOL ダンプの例 (1 / 3)

```
    CBDUMP5 COBOL プログラム

ID              DIVISION.
PROGRAM-ID.      CBDUMP5.
AUTHOR.         Sun MTP.
ENVIRONMENT      DIVISION.
CONFIGURATION    SECTION.
INPUT-OUTPUT     SECTION.
FILE-CONTROL.
SELECT  INREC   ASSIGN TO INPUT1
                   ORGANIZATION IS INDEXED
                   ACCESS IS SEQUENTIAL
                   RECORD KEY IS DOACCT
                   FILE STATUS IS INREC-STATUS.
DATA DIVISION.
FILE SECTION.
FD   INREC.
     ......
     ......
     ......
     ......
WORKING-STORAGE SECTION.
01  INREC-STATUS       PIC  9(2) VALUE ZEROES.
01  CONDCODE           PIC S9(4) COMP.
01  WS-CONS1           PIC X(20).
01  WS-CONS2           PIC X(20).
01  COUNTREC           PIC 9(9) COMP.
01  WS-S               PIC X(16) VALUE 'CBDUMP5-WS'.
01  INREC-S            PIC X(16) VALUE 'INREC'.
01  INREC-S2           PIC X(16) VALUE 'INREC2'.
01  CONDCODE-S         PIC X(16) VALUE 'CONDCODE'.
01  WS-CONS1-S         PIC X(16) VALUE 'WS-CONS1'.
01  WS-CONS2-S         PIC X(16) VALUE 'WS-CONS2'.
01  COUNTREC-S         PIC X(16) VALUE 'COUNTREC'.
01  END-WS             PIC 9(9) COMP.
```

コード例 8-4　　COBOL ダンプの例 (2 / 3)

```
  PROCEDURE DIVISION.
    CALL "CCF_SET_DUMP_AREA" USING WS-S,INREC-STATUS,END-WS.
    DISPLAY ' READING KSDS FILE '.
    OPEN INPUT INREC.
    IF INREC-STATUS NOT = 00 GO TO ABTPRIMER.
    CALL "CBDUMP5A".
    MOVE ZERO TO COUNTREC.
  LOOP1.
    READ INREC AT END GO TO ENDLOOP1.
    IF INREC-STATUS > 02 GO TO PRIMERIOERR.
    CALL "CBDUMP5A-WRITE" USING ACCTREC.
    ADD 1 TO COUNTREC.
    GO TO LOOP1.
    .....
    .....
    .....
  IDENTIFICATION DIVISION.
  PROGRAM-ID. CBDUMP5A.
  ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
  INPUT-OUTPUT SECTION.
  FILE-CONTROL.
    SELECT IFN-CCC ASSIGN TO OUTPUT1
      ORGANIZATION IS SEQUENTIAL
      FILE STATUS IS STATUS-CCC.
  DATA DIVISION.
  FILE SECTION.
    FD IFN-CCC.
    01 FD-CCC.
    ......
    ......
    ......
 WORKING-STORAGE SECTION.
    01 STATUS-CCC PIC XX VALUE "00".
    01 CONDCODE           PIC S9(4) COMP.
    01 COUNTREC           PIC S9(4) COMP.
    01 END-WS             PIC X(1).
    01 CBDUMP5A           PIC X(16) VALUE '--CBDUMP5A-WS---'.
    01 LINK5A             PIC X(16) VALUE '--CBDUMP5A-LNK--'.
  LINKAGE SECTION.
    01  INPUT-REC.
    02  INPUT-DOACCT      PIC X(5).
    02  INPUT-DOSNAME     PIC X(18).
    02  INPUT-DOFNAME     PIC X(12).
    02  INPUT-FILLER      PIC X(347).
    02  END-LNK           PIC X(1).
```

コード例 8-4　　COBOL ダンプの例 (3 / 3)

```
PROCEDURE DIVISION.
  CALL “CCF_SET_DUMP_AREA” USING CBDUMP5A,STATUS-CCC,END-WS.
  DISPLAY‘================================================’.
  MOVE ZERO TO COUNTREC.
  OPEN OUTPUT IFN-CCC.
  IF STATUS-CCC NOT = “00”
  THEN
    DISPLAY “CBDUMP5A:error opening CB01OUT1 file”
    GO TO ABNORMAL-END
  END-IF.
EXIT PROGRAM.
ENTRY “CBDUMP5A-WRITE” USING INPUT-REC.
  CALL “CCF_SET_DUMP_AREA” USING LINK5A,INPUT-REC,END-LNK.
  DISPLAY OUT2-DOACCT ‘ ‘  OUT2-DOSNAME ‘ ‘  OUT2-DOFNAME.
  WRITE FD-CCC FROM INPUT-REC.
  IF STATUS-CCC NOT = “00”
  THEN
    GO TO INVALID-CCC-WRITE
  END-IF.
  ADD 1 TO COUNTREC.

EXIT PROGRAM.
......
......
......
......
```

コード例 8-5 は、lststs ジョブ出力リストファイルの内容です。

コード例 8-5 lststs ジョブ出力リストファイルの例 (1 / 3)

```
IP4104(I) Start CBDUMP5 at Wednesday May 04 15:44:36 2005
READING KSDS FILE
================================================
10000 CAI         QIJUN
10001 GREENFIELD  STEVEN
10002 GREENFIELD  JOYCE
10003 GREENFIELD  JOSH
..............
..............
..............
..............
10004 SIH         ALEX
10005 WOLF        RICHARD
10003 GREENFIELD  JOSH
10004 SIH         ALEX
#################################################################
Wednesday May 04 15:44:38 2005 Creating dump file for:
**********************
**  JOB  dump5
**  STEP DUMP5
**  PGM  CBDUMP5
**  JON  259
**********************
Dumping Area=CBDUMP5-WS, Start-Addr=400b82b8, End-Addr=400b8370, Size=184
--------------------------------------------------------------------------
400b82b8 00000000  31300000 00000000 00000000 00000000  *10..............*
400b82c8 00000010  00000000 00000000 00000000 00000000  *................*
400b82d8 00000020  00000000 00000000 00000000 00000000  *................*
400b82e8 00000030  00000000 00000000 00000000 00000000  *................*
400b82f8 00000040  000003E1 00000000 43424455 4D50352D  *........CBDUMP5-*
400b8308 00000050  57532020 20202020 494E5245 43202020  *WS      INREC   *
400b8318 00000060  20202020 20202020 494E5245 43322020  *        INREC2  *
400b8328 00000070  20202020 20202020 434F4E44 434F4445  *        CONDCODE*
400b8338 00000080  20202020 20202020 57532D43 4F4E5331  *        WS-CONS1*
400b8348 00000090  20202020 20202020 57532D43 4F4E5332  *        WS-CONS2*
400b8358 000000a0  20202020 20202020 434F554E 54524543  *        COUNTREC*
400b8368 000000b0  20202020 20202020                    *                *
--------------------------------------------------------------------------
Dumping Area=--CBDUMP5A-WS---, Start-Addr=400b9978, End-Addr=400b9990, Size=24
--------------------------------------------------------------------------
400b9978 00000000  30300000 00000000 00000000 00000000  *00..............*
400b9988 00000010  03E10000 00000000                    *........        *
--------------------------------------------------------------------------
```

コード例 8-5　　`lststs` ジョブ出力リストファイルの例 (2 / 3)

```
Dumping Area=--CBDUMP5A-LNK--, Start-Addr=400b8130, End-Addr=400b82ae, Size=382
------------------------------------------------------------------------------
400b8130 00000000  31303030 32574F4C 46202020 20202020  *10002WOLF       *
400b8140 00000010  20202020 20202052 49434841 52442020  *       RICHARD  *
400b8150 00000020  20202020 4D522020 31323334 35363738  *    MR  12345678*
400b8160 00000030  39303132 33203454 48205354 20202020  *90123 4TH ST    *
400b8170 00000040  20202020 20202020 20203554 48204349  *          5TH CI*
400b8180 00000050  54592C20 36544820 53544154 45202020  *TY, 6TH STATE   *
400b8190 00000060  20202020 20202020 20202020 20202020  *                *
400b81a0 00000070  20202020 20202020 20202020 20202020  *                *
400b81b0 00000080  20202020 20202020 20202020 20202020  *                *
400b81c0 00000090  20202020 20202020 20202020 20202020  *                *
400b81d0 000000a0  20202020 20202020 20202020 20202020  *                *
400b81e0 000000b0  20202020 20202020 20202020 20202020  *                *
400b81f0 000000c0  20202020 20202020 20202020 20202020  *                *
400b8200 000000d0  20202020 20202020 20202020 20202020  *                *
400b8210 000000e0  20202020 20202020 20202020 20202020  *                *
400b8220 000000f0  20202020 20202020 20203231 31313139  *          211119*
400b8230 00000100  30523131 31312031 314E2020 31303030  *0R1111 11N  1000*
400b8240 00000110  2E303020 20202030 2E303030 30303030  *.00    0.0000000*
400b8250 00000120  30202020 20302E30 30303030 30303020  *0    0.00000000 *
400b8260 00000130  20202030 2E303020 20202030 2E303030  *   0.00    0.000*
400b8270 00000140  30303030 30202020 20302E30 30303030  *00000    0.00000*
400b8280 00000150  30303020 20202030 2E303020 20202030  *000    0.00    0*
400b8290 00000160  2E303030 30303030 30202020 20302E30  *.00000000    0.0*
400b82a0 00000170  30303030 30303020 20202030 2E30      *0000000    0.0  *
------------------------------------------------------------------------------
##############################################################################
Execution error :file '/machine-1/batch/samples/primer/CBDUMP5.int'
error code:114, pc=133, call=2, seg=0
114   Attempt to access item beyond bounds of memory (Signal 11)

...
...
...
```

コード例 8-5 lststs ジョブ出力リストファイルの例 (3 / 3)

```
Thread mode:          No Threads
RTS Error:            COBOL
Sync Signals:         COBOL
ASync Signals:        COBOL
cobtidy on exception: False

rtsvsam:RETURN-CODE=9999, COND-CODE=255
rtsvsam:user exit rollback function executed
IP4105(I) End of CBDUMP5 at Wednesday May 04 15:44:38 2005


/machine-1/batch/acctseq DELETED
DUMP5 step was executed cond code 255
step DUMP5 start at TIME=03.44.35 (PM)
step DUMP5 stop  at TIME=03.44.39 (PM)
```

COBOL エラー終了処理

cobwrap.cbl プログラムには、COBOL ユーザー出口があります。このファイルは、$PACK/RTSFS ディレクトリまたは $UNIKIX/src/RTSVSAM ディレクトリにあります。次の表に、cobwrap.cbl プログラムコンポーネントの一部をリストします。

コンポーネント	説明
CBL_EXIT_PROC	CBLEXIT 終了ハンドラをインストールするために COBOL を呼び出します。STOP RUN が実行されると、Sun MBM COBOL 実行時システムは CCFexitwrap ルーチンを呼び出します。
CBL_ERROR_PROC	CBLERROR 終了ハンドラをインストールするために COBOL を呼び出します。COBOL プログラムが中止すると、Sun MBM COBOL 実行時システムは、CCFerrorwrap ルーチンを呼び出します。
READ_PARM	IBM JCL に指定された PARM パラメータが EXEC 文領域に配置されます。
PROGNAME	Sun MBM では、引数で指定したユーザープログラム名で cobwrap を呼び出します。PROGNAME には、実行するユーザーの COBOL プログラムの名前を指定します。
CCFstopwrap	ユーザーの COBOL プログラムから戻る命令が GOBACK 文により発行されると呼び出されます。このルーチンは、ユーザーの COBOL プログラムの RETURN-CODE に基づいて、データベースをコミットまたはロールバックします。

コンポーネント	説明
CCFexitwrap	ユーザーの COBOL プログラムから戻る命令が STOP RUN 文により発行されると呼び出されます。このルーチンは、ユーザーの COBOL プログラムの RETURN-CODE に基づいて、データベースをコミットまたはロールバックします。Server Express を使用する場合は cobtidy() コマンド、ACUCOBOL-GT を使用する場合は acu_shutdown() コマンドで COBOL ファイルがフラッシュされます。
CCFerrorwrap	ユーザーの COBOL プログラムが失敗すると、呼び出されます。このルーチンは、データベースをロールバックします。

COBOL エラー終了処理をカスタマイズする必要がある場合は、ご購入先に連絡してください。

連結データセットとベース名 GDG へのアクセス

Sun MBM COBOL 実行時システムを使用して、アプリケーションプログラムで、順編成 (FS) 連結データセットおよびベース名 GDG のすべてのファイルにアクセスできます。この実行時システムでは、ファイルにアクセスする COBOLプログラムにファイルを割り当てるのに使ったファイル名環境変数から、ベース名 GDG または連結データセットを特定します。この環境変数の形式は、DD_*filename* です。ここで、*filename* は、COBOL プログラムで SELECT 文の ASSIGN 句に指定した値です。$DD_*filename* は、コロンで区切った連結データセットの物理ファイルのリスト、またはコロンで区切った複数のベース名 GDG のオカレンスに設定します。

次の例では、ファイル名は CONDATA で、DD_CONDATA 環境変数が設定されます。

コード例 8-6 ベース名 GDG または連結データセットへのアクセス

```
DD_CONDATA=file1:file2:file3:....
Basename GDG
DD_INFILE=file1_00:file2_01:file3_02:....
```

COBOL 実行時システムは、COBOL プログラム内の I/O 呼び出しを処理して、複数のファイルが 1 つの論理ファイルに含まれていないかどうかを判定します。たとえば、file1 がファイルの最後に到達すると、リストから file2 を取得して開き、処理します。同じ手順が file3 およびそれ以降にも適用されます。

DD_*filename* 環境変数で指定したファイルは、レコード記述と長さが同じでなければなりません。

ベース名 GDG または連結データセットにアクセスする COBOL プログラムに必要なコンパイルオプションについては、233 ページの「COBOL アプリケーションプログラムのコンパイル」を参照してください。

COBOL プログラムのデバッグ

この節では、デバッグオプションと COBOL デバッガを使った、COBOL プログラムのデバッグ方法について説明します。

Sun MTP VSAM デバッグ機能の使用

BAM を使用してサブシステム用に Sun MTP VSAM Data Management を選択すると、`$UNIKIX/src/RTSVSAM` から生成された COBOL 実行時システムは、VSAM ファイルへのアクセスに使用されます。COBOL プログラムがこの実行時システムで実行されるとき、デバッグオプションを有効にして、デバッグメッセージを `unikixmain.dbg` ファイルにリダイレクトできます。このファイルは、`$KIXSYS` ディレクトリにあります。`unikixjob` コマンドに `-DVSAM` オプションを指定すると、Sun MBM からこの機能を呼び出すことができます。詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

COBOL デバッガの使用

ベンダーの COBOL デバッガを使用して、バッチジョブから COBOL プログラムをデバッグできます。

注 – サブミットしたジョブだけをデバッグできます。

▼ Server Express COBOL プログラムをデバッグする

1. **Server Express コンパイラの Animator オプションを使って、アプリケーションプログラムをコンパイルします。**

   このオプションでは、*prog*`.int` および *prog*`.idy` ファイルが作成されます。*prog* は、COBOL アプリケーションプログラムの名前です。

2. ***prog*`.int` および *prog*`.idy` ファイルを COBOL プログラムの実行ファイルのディレクトリに配置します。**

   このディレクトリに *prog*`.gnt` ファイルがすでに存在する場合は削除します。Sun MBM では *prog*`.int` ファイルの前に *prog*`.gnt` ファイルを検索するため、`.gnt` ファイルが存在すると、Animator は動作しません。

3. **COBOL デバッガオプション (`-a`) を指定する `unikixjob` または `subjob` コマンドを使って、ジョブをサブミットします。**

   実行するコマンドの結果として返されるジョブオカレンス番号 (*jon*) を書き留めます。

4. **デバッグするジョブの *jon* を指定して、端末から `anmjob` コマンドを実行します。**

   COBOL プログラムのデバッグについての詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

▼ ACUCOBOL-GT プログラムをデバッグする

1. **ACUCOBOL-GT コンパイラのデバッグオプションを使って、アプリケーションプログラムをコンパイルします。**

   このオプションでは、*prog*`.acu` ファイルが作成されます。*prog* は、COBOL アプリケーションプログラムの名前です。

2. ***prog*`.acu` ファイルを COBOL プログラムの実行ファイルのディレクトリに配置します。**

3. **COBOL デバッガオプション (`-a`) を指定する `unikixjob` または `subjob` コマンドを使って、ジョブをサブミットします。**

   実行するコマンドの結果として返されるジョブオカレンス番号 (*jon*) を書き留めます。

4. **デバッグするジョブの *jon* を指定して、端末から `anmjob` コマンドを実行します。**

   COBOL プログラムのデバッグについての詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

Sun MBM ノードの現在の日付の取得

ほとんどの COBOL アプリケーションでは、日付を年/月/日の順序で表示します。こうすると、日付フィールド全体を 1 つのユニットとして使用できるので、レコードのソートが簡単です。一般に、ANSI-74 COBOL `ACCEPT` 文を使ってプログラムでシステム日付を取得します。年は 2 桁で返されます。1989 年に改訂された ANSI-85 COBOL には、この目的に使用できる組み込み関数として日付ルーチンのセットが用意されています。

システムで Sun MBM 日付/時刻設定ルーチンをサポートしている場合は、BAM を使ってサブシステムを構成し、標準の COBOL 日付/時刻ルーチンで、現在のシステム日付ではなくバッチシステム日付を返すことができます。詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』の日付/時刻設定の説明を参照してください。

システムで Sun MBM 日付/時刻設定機能をサポートしていない場合は、Sun MBM 関数 `EBM_CURRENT_DATE` を使って、現在のバッチシステム日付を取得する必要があります。

Sun MBM 日付が変更されなかった場合は、`EBM_CURRENT_DATE` 関数により現在のシステム日付が返されます。

注 – Sun MBM 関数の EBM_CURRENT_DATE を使用して、特定の時間間隔でバッチジョブを実行できます。

EBM_CURRENT_DATE 関数は、標準 ANSI-85 COBOL の組み込み関数である CURRENT-DATE で定義した、21 文字の英数字の値を返します。この値は、アプリケーションプログラムを実行するバッチノードの日付、時刻、およびグリニッジ標準時との時差を表します。

ANSI-85 組み込み関数と同様に、EBM_CURRENT_DATE 関数は、実行時に関数が呼び出された時刻に判定された値を返します。したがって、複数の呼び出しで、さまざまな値が返されます。

21 文字の日付値 EBM_CURRENT_DATE の戻り値の形式は、次のとおりです。

yyyymmddhhmmsstt(+/-)*hhmm*

説明

yyyy	年
mm	月: 01 ～ 12
dd	月の日付: 01 ～ 31
hh	時間 (24 時間システム): 00 ～ 23
mm	分: 00 ～ 59
ss	秒: 00 ～ 59
tt	100 分の 1 秒単位: 00 ～ 99
(+/-)	レポートされる時刻は、グリニッジ標準時 (GMT) よりも早い (+) または遅い (-) で表されます。システムにこの機能がない場合、このフィールドはデフォルトで 0 に設定されます。 *hh*: レポートされる時刻が GMT より何時間進んでいるかまたは遅れているかを示します *mm*: レポートされる時刻が GMT より何分進んでいるかまたは遅れているかを示します

コード例 8-7 EBM_CURRENT_DATE 関数の呼び出し

```
WORKING-STORAGE SECTION.
01      GET-CURRENT-DATE  PIC   X(21).
PROCEDURE DIVISION.
CALL  "EBM_CURRENT_DATE"  USING  GET-CURRENT-DATE.
```

C および C++ アプリケーション プログラム

この節では、Sun MBM の制御下で C および C++ プログラムを準備し、実行する方法を説明します。

▼ C または C++ サブシステムを構成する

1. BAM を起動します。
2. 「BAM」メニューで、オプション「3 Applications & Subsystems」を選択します。
3. 「Applications & Subsystems」メニューで、オプション「3 Create a Subsystem」を選択します。
4. 作成するサブシステムの名前を入力してから Return キーを押します。
5. 「Create」メニューで、オプション「1 Application Languages」を選択します。
6. 「Applications & Subsystems」で、「4 C/C++」を選択します。図 8-3 の「Application Languages」画面が表示されます。

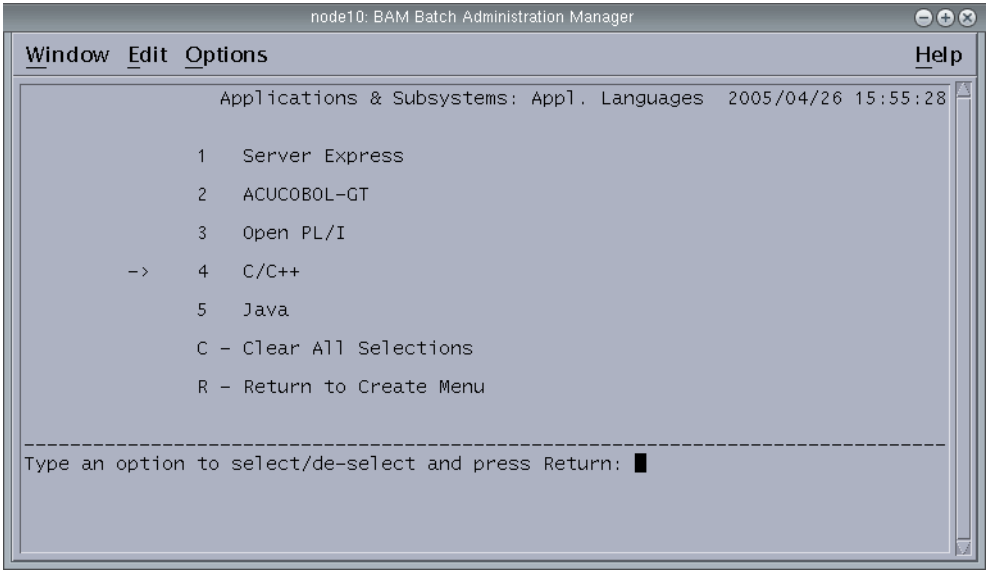


図 8-3　C/C++ の選択

7. 「Create」メニューで、使用しているデータファイルのタイプを選択します。
   - プログラムで Sun MTP VSAM ファイルにアクセスする場合は、「Applications & Subsystems: Data Management」画面で Sun MTP VSAM 相対編成ファイルおよび索引編成ファイルを選択します。
   - プログラムで RDBMS にアクセスする場合は、「Applications & Subsystems: RDBMS」画面でデータベースを選択します。
8. オプションの選択を続行します。
9. サブシステムを構築します。

   サブシステムの作成についての詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

C/C++ プログラムのコンパイル

標準の C/C++ コンパイラおよび `cc`、`CC`、`make`、`lex`、`yacc`、`ld` などの関連ユーティリティーを使ってコマンド行から、または統合開発環境 (IDE) から、プログラムをコンパイルし、リンクできます。

Sun MBM は、特定のコンパイラオプションを必要としません。また、Sun MBM に固有のオブジェクトをアプリケーションプログラムとリンクする必要もありません。

C/C++ コンパイラツールまたは `#pragma` オプションを使って、コンパイラオプションを指定します。カタログ化された手続きの MVS JCL `CPARM` パラメータはサポートしていません。

実行可能ファイルは、サブシステムを作成するときに定義したデフォルトのアプリケーションプログラムディレクトリまたは `File_Map` に定義された JCL ライブラリに配置する必要があります。

Sun MBM では、メインフレームのカタログ式手続きをサポートしません。この手続きは、`EDCCBG`、`CBCBG`、`CBCCBG`、`CBCG` などのプログラムをコンパイル、リンク、または実行するために使用されます。

プログラムの実行

C/C++ プログラムは、次のいずれかの方法で呼び出すことができます。

- `exec()` 系関数を使って別のプログラムから呼び出す方法
- (POSIX.2 で定義されている) `system()` 関数を使用する方法
- `BPXBATCH` ユーティリティーを使用する方法 (259 ページの「`BPXBATCH` プログラムの使用」を参照)
- MVS JCL から `EXEC` 文を使って直接呼び出す方法

他の製品との対話

DB2 は、オープンシステムで使用可能な RDBMS の 1 つを使用してサポートされます。ただし、RDBMS トランスレータを使って、プログラムに組み込まれた SQL 文を前処理する必要があります。C/C++ コンパイラは、SQL トランスレータの出力のコンパイルに使用します。

必要な場合は、アプリケーションプログラムで適切な CONNECT、COMMIT/ROLLBACK 呼び出しを発行します。

コマンド行のリダイレクト

stdin、stdout、または stderr のリダイレクトは、次の機能のいずれかを使用する場合にのみサポートされます。

- exec() 系関数
- system() 関数 (POSIX.2 で定義)
- BPXBATCH ユーティリティー (259 ページの「BPXBATCH プログラムの使用」を参照)

MVS JCL EXEC 文を使用する場合、コマンド行のリダイレクトはサポートされません。

BPXBATCH プログラムの使用

BPXBATCH プログラムを使用して、ユーザーは、MVS JCL ジョブストリームまたはマクロジョブからシステムコマンド、スクリプト、または C/C++ 実行可能ファイルを実行できます。

コード例 8-8 BPXBATCH—インストリームファイルでのシステムコマンドの実行

```
//GO EXEC PGM=BPXBATCH,PARM='SH'
//SYSIN DD *
system commands
/*
```

コード例 8-9 BPXBATCH—シェルコマンドの実行

```
//GO EXEC PGM=BPXBATCH,PARM='SH command-name'
```

コード例 8-10　BPXBATCH—C/C++ プログラムの実行

```
//GO EXEC PGM=BPXBATCH,PARM='PGM program-name'
```

コード例 8-11　BPXBATCH—入力パラメータの指定

```
//GO EXEC PGM=BPXBATCH,PARM='PGM program-name parm1 parm2 parm3'
```

Sun MBM 環境では、デフォルトで stdin だけが DUMMY デバイスに関連付けられます。stdout および stderr は、ジョブリスト出力ファイルにリダイレクトされます。

STDIN、STDOUT、および STDERR DD 文を使って、次のように標準ファイルを割り当てることができます。

```
//GO EXEC PGM=BPXBATCH,PARM='PGM program-name'
//STDIN DD PATH='/stdin-pathname'
//STDOUT DD PATH='/stdout-pathname'
//STDERR DD PATH='/stderr-pathname'
```

PATHOPTS パラメータは、mvstrans では無視されます。

MVS JCL EXEC 文の使用

MVS JCL EXEC 文を使用して、C/C++ プログラムを直接呼び出すことができます。

```
//GO EXEC PGM=program-name
```

実行時オプションは、Sun MBM 環境では適用されません。スラッシュセパレータ (/) を使ってこれらのオプションをプログラムパラメータと区別すると、トランスレータでは、これらのオプションを識別してスキップすることができます。スラッシュの前後には空白文字を挿入します。

有効な文

入力パラメータはプログラムに渡されません。

```
//GO EXEC PGM=PGM1,PARM='RPTOPTS(ON) / '
```

パラメータ parm1、parm2、および parm3 がプログラムに渡されます。

```
//GO EXEC PGM=PGM1,PARM='parm1 parm2 parm3'
```

次の例でも、入力パラメータは parm1、parm2、および parm3 です。mvstrans は、"/ " セパレータまで実行オプションをスキップします。

```
//GO EXEC PGM=PG1,PARM='RPTOPTS(ON) / parm1 parm2 parm3'
```

上記の文は次のように変換されます。

```
EXECPGM PGM='PG1',PARM='parm1 parm2 parm3'
```

無効な文

プログラムに渡される入力パラメータに実行オプションが含まれています。

```
//GO EXEC PGM=PGM1,PARM='RPTOPTS(ON)/parm1 parm2 parm3'
```

プログラムに渡される入力パラメータの区切り記号は、空白文字だけです。

MVS JCL JOBLIB/STEPLIB DD 文の使用

MVS JCL STEPLIB 文と JOBLIB DD 文を使用して、ユーザー指定のディレクトリ内の C/C++ プログラムを実行できます。

次の例では、File_Map の MED.LOAD.INE ライブラリに関連付けられるディレクトリ内のプログラム PGM1 の実行方法を示します。

```
//GO EXEC PGM=PGM1
//STEPLIB DD DSN=MED.LOAD.INE,DISP=SHR
```

さまざまなファイルタイプへのアクセス

順編成ファイルなどの従来のファイルだけが標準の C または C++ 入出力ルーチンを使ってサポートされます。レコード入出力タイプはサポートされません。

以下の小節に、C または C++ プログラムで使用できるようにさまざまなファイルタイプにアクセスする方法を説明します。

連結データセット

連結データセットを使用すると、1 つの ddname で複数のデータセットを指定できます。

```
//MYDD DD DSN=DSN1,DISP=SHR
//      DD DSN=DSN2,DISP=SHR
```

バッチ実行時システムは、次の形式で環境変数を設定します。

```
DD_ddname=pathname1:pathname2:...:pathname3
```

C または C++ プログラムで、この変数を使用して割り当てられた連結パス名を取得します。

たとえば、File_Map にある DSN1 および DSN2 に関連するパス名が /path1 および /path2 の場合、DD_MYDD 環境変数は次のように設定されます。

```
DD_MYDD=/path1:/path2
```

注 – Sun MBM では、fp=fopen("dd:MYDD","flags"); 形式はサポートされません。

区分データセット

区分データセット (PDS) および拡張区分データセット (PDSE) は、分割されたファイルのセットとして実装され、共通ディレクトリに定義されます。

バッチ実行時システムでは、割り当てられたファイルを取得するために C または C++ プログラムで使用できる環境変数を設定します。

```
//DD1 DD DSN=TS012.MY.DATA(MEMBER1),DISP=SHR
```

TS012.MY.DATA(MEMBER1) データセット用の File_Map にある関連ファイルが /data/seqfiles/user1/data/member1 の場合、次の環境変数が関連します。

```
DD_DD1=/data/seqfiles/user1/data/member1
```

注 – Sun MBM では、fp=fopen("dd:DD1","r"); 形式はサポートされません。

区分連結データセット

1 つの ddname で、複数の PDS または PDSE を指定できます。

Sun MBM 実行時システムでは、割り当てられたパス名を取得するために C または C++ プログラムで使用できる環境変数を設定します。

```
//MYDD  DD DSN=TSO12.PDS1(A),DISP=SHR
//      DD DSN=TSO12.PDS2(C),DISP=SHR
```

2 つのメンバーに対して File_Map に含まれる関連パス名が次のような場合、

```
/seqfiles/pds1/a and /seqfiles/pds2/c
```

関連の環境変数に値が設定されます。

```
DD_MYDD=/seqfiles/pds1/a:/seqfiles/pds2/c
```

注 – Sun MBM では、fp=fopen("dd:MYDD(A)","r"); 形式はサポートされません

インストリームデータセット

インストリームデータセットとは、JCL 文の 1 つのセット内に含まれるデータセットです。次に例を示します。

```
//MYDD DD *
record1
record2
record3
/*
```

Sun MBM 実行時システムは、インストリームデータセットに関連する環境変数を設定します。この例では、変数は次のように設定されます。

```
DD_MYDD=/temporary-file-pathname
```

レコード長は常に 80 に設定され、レコード形式は常に固定です。

注 – Sun MBM では、`fp=fopen("DD:MYDD","flags");` 形式はサポートされません。

SYSOUT データセット

`DD` 文に `SYSOUT` パラメータを指定する `SYSOUT` データセットだけを割り当てることができます。

注 – Sun MBM では、`fopen("*","W, lrecl=...");` 形式はサポートされません。

Sun MBM 実行時システムは、`SYSOUT` データセットに関連する環境変数のグループを設定します。次に例を示します。

`DD1_FILENAME=`	`SYSOUT` データセットの名前
`DD1_COPIES=`	印刷するコピー部数
`DD1_DEST=`	この `SYSOUT` データセットの宛先

関連する環境変数の全リストについては、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

テープファイル

テープファイルは、標準の順編成ファイルとして処理されます。テープを割り当てるには、ddname を使用します。

注 – Sun MBM では、fd=fopen("dd:SYSUT2","..."); 形式はサポートされません。

Sun MBM 実行時システムは、割り当てられている順編成ファイル名を C または C++ プログラムから取得するために使用する環境変数を設定します。

たとえば、次の DD 文があるとします。

```
//SYSUT2 DD DSN=TAPE1.OUTPUT,UNIT=xxx,LABEL=.....
```

次の環境変数が関連します。

DD_SYSUT2=/*file-pathname* (File_Map に指定)。

DUMMY データセット

DUMMY データセットは、/dev/null 仮想デバイスを使って処理されます。

DUMMY データセットは、JCL からのみ割り当てることができます。

バッチ実行時システムは、関連する環境変数を /dev/null に設定します。

一時データセット

一時データセットは、JCL からのみ割り当てることができます。

注 – Sun MBM では、fd=fopen("//&&myfile","..."); 形式はサポートされません。

バッチ実行時システムは、割り当てられた一時ファイルを指定する環境変数を設定します。

C および C++ プログラムでは、この環境変数を使用して、関連するパス名を取得します。

VSAM データセット

Sun MBM では、`fopen()`、`fread()`、`fwrite()` などのメインフレーム拡張機能を使った 3 つのタイプの VSAM データ構成はサポートされません。ただし、C/C++ プログラムでは、Sun MTP VSAM/C-ISAM インタフェースを使って VSAM ファイルにアクセスできます。Sun MTP Data Management を使用できるようにサブシステムを構成し、`libcisam.a` Sun MTP ライブラリを使って C/C++ プログラムにリンクする必要があります。詳細は、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。

Sun MBM 実行時システムは、割り当てられている VSAM データセットに対して環境変数を設定します。次に例を示します。

次の文で VSAM データセットが割り当てられているとします。

```
//VS1 DD DSN=VSAM1.CLUSTER,DISP=OLD
```

次の環境変数が関連します。

```
DD_VS1=VSAM1
```

`VSAM1` は、FCT と VSAM カタログ、および `File_Map` に定義されたファイルの名前です。

階層ファイルシステムファイル

データセット名の代わりに `HFS` 名を指定して、MVS JCL `DD` 文の `PATH` オプションで、ファイルを割り当てることができます。次に例を示します。

```
//DD1 DD PATH=/pathname,DISP=SHR
```

端末ファイル

Sun MBMでは、端末ファイルをサポートしません。

メモリーファイル

パフォーマンスを向上するためにメモリーに一時ファイルを常駐させる機能はサポートされません。

世代データグループ (GDG)

GDG は、JCL からのみ割り当てることができます。

注 – Sun MBM では、`fd=fopen("DATA.MYGDG(+1)","...");` 形式はサポートされません。

バッチ実行時システムは、C/C++ プログラムで使用できる特定の環境変数を設定して、割り当てられたパス名を取得します。複数の GDG オカレンスが割り当てられている場合、関連するパス名はコロン (:) で区切られます。次に例を示します。

次の文で GDG が割り当てられているとします。

```
//DD1 DD DSN=MYGDG,DISP=OLD
```

上記の文で、次の関連する環境変数を設定します。

```
DD_DD1=/pathname_MYGDG_00:/pathname_MYGDG_01:/pathname_MYGDG_02
```

入力/出力タイプ

Sun MBM では、レコード I/O タイプの C/MVS 拡張機能はサポートされません。

`LRECL` および `RECFM` などのファイル属性は、環境変数を使って C/C++ プログラムに渡されます。次に例を示します。

次の文でデータセットが割り当てられているとします。

```
//MYDD DD DSN=DSN1,LRECL=120,RECFM=V
```

次の環境変数が関連します。

`File_Map` で指定される `DSN1` データセットの `DD_MYDD=`*/pathname*

`MYDD_LRECL=120`

`MYDD_RECFM=recordv`

次の場所に `RECFM` および `LRECL` を指定できます。

- データセットを割り当てる `DD` 文
- ファイルを割り当てる Sun MBM マクロ `ASSGNDD` 文
- データセットに関連する `File_Map` エントリ

事前定義のストリーム

C プログラムの事前定義のストリームのサポートは、次のとおりです。

- stdin (デフォルトのオープンモードは r) は、MVS JCL DD SYSIN 文から取得されます。
- stdout (デフォルトモードは w) は、ジョブリスト出力ファイルに渡されます。

  Sun MBM では、dd:system または dd:syserr の割り当てに基づく stdout のリダイレクトを行いません。
- stderr (デフォルトモードは w) は、ジョブリスト出力ファイルに渡されます。

  MVS JCL MSGFILE オプションを使って stderr をリダイレクトできません。

C++ プログラムの事前定義のストリームのサポートは、次のとおりです。

- cin: stdin から取得されます
- cout: stdout に渡されます
- cerr: stderr に渡されます

注 – MVS JCL PARM パラメータで、リダイレクト記号を使って stdout をリダイレクトすることはできません。

次に例を示します。

```
//GO EXEC PGM=PGM1,PARM='> LOG.LISTING'
```

ファイルのオープン

C または C++ プログラムを使っている場合、ddname またはデータセット名を使ってファイルを開くことはできません。たとえば、Sun MBM では、次の文はサポートされません。

```
fp=fopen("dd:ddname","...");
fp=fopen("VSAM1.CLUSTER","...");
```

ddname を使用した場合、C/C++ プログラムでは、ddname 関連環境変数 (DD_ddname の形式) を取得し、関連するパス名を入出力関数のパラメータとして使用します。

データセット名を使用した場合、C/C++ プログラムでは、Sun MBM ftval コマンドにより、File_Map から関連するパス名を取得します。

完全修飾されていないパス名でファイルを開くと、BAM で別の作業ディレクトリを指定しない限り、ファイルは現在の作業ディレクトリ (デフォルトではユーザーの $HOME ディレクトリ) で開かれます。

Java アプリケーションプログラム

この節では、Sun MBM の制御下で Java プログラムを準備し、実行する方法を説明します。

注 – Sun MBM 環境では、Java プログラムを使用して Sun MTP VSAM データセットにアクセスすることはできません。

▼ Java サブシステムを構成する

1. BAM を起動します。
2. 「BAM」メニューで、オプション「3 Applications & Subsystems」を選択します。
3. 「Applications & Subsystems」メニューで、オプション「3 Create a Subsystem」を選択します。
4. 作成するサブシステムの名前を入力してから Return キーを押します。
5. 「Create」メニューで、オプション「1 Application Languages」を選択します。
6. 図 8-4 の「Application Languages」画面で、オプション「5 Java」を選択して、Return キーを押します。

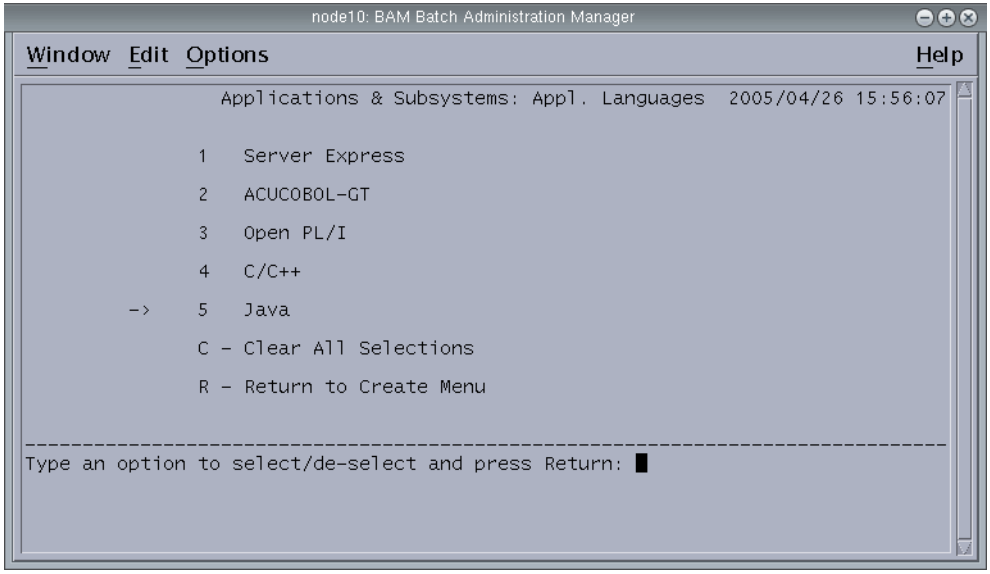


図 8-4　　Java の選択

7. 「Create」メニューで、使用しているデータファイルのタイプを選択します。

   プログラムで RDBMS にアクセスする場合は、「Applications & Subsystems: RDBMS」画面でデータベースを選択します。

8. オプションの選択を続行します。

9. サブシステムを構築します。

   サブシステムの作成についての詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

Java クラスのコンパイル

`javac` および `jar` などの標準 Java SDK ツールを使って、Java クラスをコンパイルし、アーカイブできます。

Sun MBM は、特定の Java コンパイラオプションを必要としません。また、アプリケーションを実行するときに Sun MBM に固有の Java クラスを起動する必要もありません。

Java クラスの起動

次の方法を使って、Java クラスを起動できます。

- Sun MBM EJAVA ユーティリティー
- Sun MBM EJAR ユーティリティー
- MVS JCL JOBLIB および STEPLIB 文

Sun MBM EJAVA ユーティリティー

EJAVA ユーティリティーを使用して、ジョブから Java クラスを起動できます。次に例を示します。

Java クラス MyClass.class を起動するには、次の MVS JCL 文を使用します。

```
//STEP1 EXEC PGM=EJAVA,PARM='MyClass'
```

Sun MBM EJAR ユーティリティー

EJAR ユーティリティーを使用して、Java Archive File (JAR) にアーカイブされている Java メインクラスをジョブから起動できます。次に例を示します。

JAR ファイル MyApp.jar に定義されている Java メインクラスを起動するには、次の MVS JCL 文を使用します。

```
//STEP1 EXEC PGM=EJAR,PARM='MyApp'
```

MVS JCL JOBLIB/STEPLIB DD 文

MVS JCL JOBLIB および STEPLIB DD 文を使用して、ユーザー指定のディレクトリに存在する Java クラスを起動できます。次に例を示します。

次の例では、File_Map の MED.LOAD.INE ライブラリに関連付けられるディレクトリ内に存在する Java クラス、MyClass.class を起動します。

```
//GO EXEC PGM=EJAVA,PARM='MyClass'
//STEPLIB DD DSN=MED.LOAD.INE,DISP=SHR
```

Java クラスへのパラメータの引き渡し

Java クラスに入力パラメータを指定するときには、区切り記号として空白文字を使用する必要があります。次に例を示します。

MVS JCL EXEC 文を使用する場合は、PARM パラメータに空白で区切った入力パラメータを追加します。

```
//GO EXEC PGM=EJAVA,PARM='MyClass parm1 parm2 parm3'
```

Open PL/I アプリケーションプログラム

この節では、Sun MBM の制御下で Open PL/I プログラムを準備し、実行する方法を説明します。

移行の問題: UNIX の Open PL/I 実装では、PL/I コンパイラは、SYSPRINT (暗黙または明示的) と宣言されたすべてのファイルに GLOBALREF 属性を適用します。定義用インスタンス (GLOBALDEF) は、PL/I 実行時ライブラリにあります。PL/I 実行環境は、SYSPRINT というファイルを (SYSPRINT がファイルシステムファイルに割り当てられている場合でも) 標準出力に割り当てます。これにより、SYSPRINT 出力が Sun MBM ジョブ履歴ファイルにリダイレクトされます。

したがって、PL/I を使用するときに、次の操作を行う必要があります。

割り当てられたファイルシステムファイルに SYSPRINT 出力を送信する場合、GLOBALDEF 宣言を 1 つアプリケーションに追加します。

```
DCL SYSPRINT FILE GLOBALDEF EXTERNAL ('SYSPRINT2');
```

同じアプリケーションにリンクされている他のモジュールで、次のように定義します。

```
DCL SYSPRINT FILE GLOBALREF EXTERNAL ('SYSPRINT2');
```

これにより、オブジェクトファイルの外部名 SYSPRINT が SYSPRINT2 に変更され、実行時標準出力のチェックが省略されます。SYSPRINT 出力は、同じ名前でファイルに送られます。

PL/I サブシステムの構成

▼ PL/I サブシステムを構成する

1. BAM を起動します。
2. 「BAM」メニューで、オプション「3 Applications & Subsystems」を選択します。
3. 「Applications & Subsystems」メニューで、オプション「3 Create a Subsystem」を選択します。
4. 作成するサブシステムの名前を入力してから Return キーを押します。
5. 「Create」メニューで、オプション「1 Application Languages」を選択します。
6. 図 8-5 の「Application Languages」画面で、オプション「3 Open PL/I」を選択して、Return キーを押します。

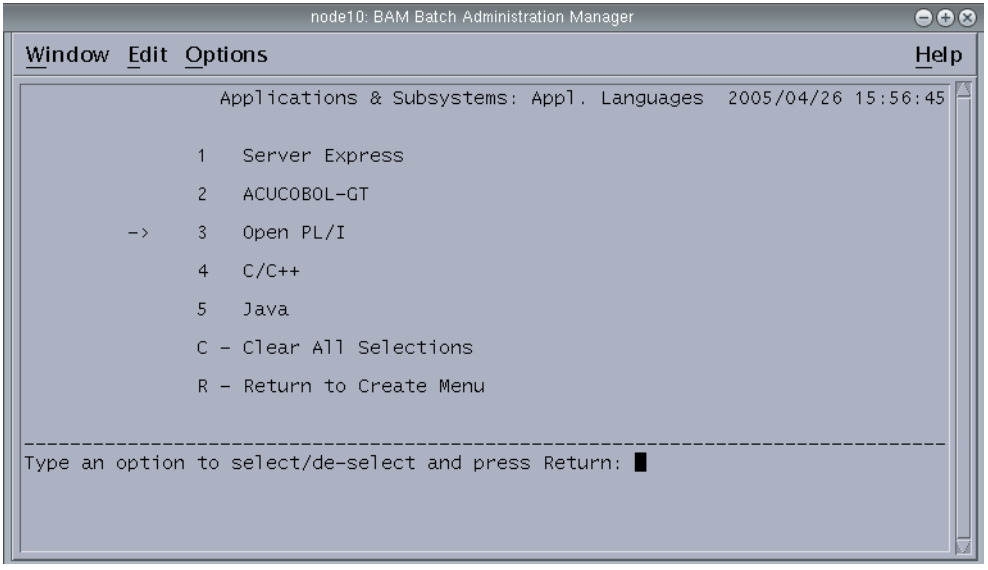


図 8-5　Open PL/I の選択

7. 「Create」メニューで、使用しているデータファイルのタイプを選択します。
   - プログラムで Sun MTP VSAM ファイルにアクセスする場合は、「Applications & Subsystems: Data Management」画面で Sun MTP VSAM 相対編成ファイルおよび索引編成ファイルを選択します。
   - プログラムで RDBMS にアクセスする場合は、「Applications & Subsystems: RDBMS」画面でデータベースを選択します。
8. オプションの選択を続行します。

9. **サブシステムを構築します。**

   サブシステムの作成についての詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

PL/I プログラムのコンパイル

▼ PL/I プログラムをコンパイルする

1. **サブシステムの環境を設定する方法は、次のとおりです。**
   - ソース `batchenv` でサブシステム名を指定します。次に例を示します。

```
$ . $EBMHOME/batchenv plisys
```

   - 次の手順に従います。

   a. Sun MBM メインメニューを表示します。

   b. メインメニューの「Command Prompt」アイコンをクリックし、サブシステム名を入力します。

2. **PL/I プログラムが RDBMS にアクセスし、SQL 文と標準の PL/I 文が含まれる場合は、適切な RDBMS プリコンパイラプログラムにより元の PL/I ソースを変換して、ソースファイルを生成します。**

3. **PL/I プログラムで Sun MTP VSAM ファイルにアクセスする場合は、サブシステムを必ず VSAM Data Management 用に構成します。**

4. **Open PL/I コンパイラでプログラムをコンパイルします。**

   次に例を示します。

```
$ lpiplx prog123.pl1 -ebm -defext
```

   手順 2 を実行すると、入力した PL/I ソースファイルが RDBMS プリコンパイラから出力されます。実行しない場合は、PL/I ソースが出力されます。

5. ldpl1 コマンドを使用し、現在の環境に対応するオプションを指定して、プログラムをリンクします。

たとえば、prog123.o をリンクして prog123.plx を作成します。

```
$ ldpl1 prog123.o -ebm -o prog123.plx
```

リンカーコマンドで使用できるオプションは次のとおりです。

-ebm	Sun MBM で実行し、Open PL/I ファイルハンドラを使用します。
-ebm -ebmvsam	Sun MBM で実行し、Sun MTP VSAM ファイルにアクセスします。
-ebm -oracle	Sun MBM で実行し、Oracle および Open PL/I ファイルハンドラを使用します。
-ebm -ebmvsam -oracle	Sun MBM で実行し、Sun MTP VSAM ファイルと Oracle の両方にアクセスします。
-o *filename*.plx	必要な .plx 拡張子を持つ出力ファイル名を示します。出力ファイル名に -o オプションを使用しない場合、生成後に実行可能ファイルの名前を変更する必要があります。

PL/I プログラムの実行

PL/I プログラムを実行するには、subjob コマンドを使用する必要があります。ジョブに VSAM Data Management が必要な場合は、unikixjob コマンドを使用します。このコマンドについては、『Sun Mainframe Batch Manager ソフトウェアユーザーズガイド』を参照してください。

PL/I プログラムの検索規則

PL/I プログラム実行可能ファイルには、拡張子 .plx を使用する必要があります。PL/I プログラム実行可能ファイルは、EXECPGM マクロを使用して実行します。File_Map の SYS1.LINKLIB エントリで、PL/I 実行可能ファイルを含むディレクトリを定義します。また、実行時に JOBLIB DD および STEPLIB DD 文を使用すると、このディレクトリを上書きできます。検索規則は COBOL と同じなので、詳細は 237 ページの「COBOL プログラムの検索規則」を参照してください。

ジョブステップリターンコード

PL/I アプリケーションプログラムでは、組み込みサブルーチン PLIRETC を使用してジョブステップリターンコードを設定します。プログラムが正常に終了すると、あとのジョブステップ処理を判定するジョブの条件付きロジックに使用できるジョブステップリターンコードが作成されます。条件付きロジックは、MVS JCL では EXEC 文の COND パラメータから生成され、VSE JCL で $RC パラメータを使用する場合は IF 文から生成されます。

メモリー障害または 0 で除算などでプログラムが中止した場合は、0 以外のジョブステップ完了コードを作成してプログラムは終了します。これでは、ジョブステップリターンコードが未定義になるので、あとのジョブステップ処理の判定に使用することはできません。

PL/I プログラムでの File_Map の使用法

File_Map は、プログラム実行可能ファイルと手続きを検索するパスを判定するためにアクセスされます。規則の詳細は、275 ページの「PL/I プログラムの検索規則」を参照してください。

Open PL/I でファイルを開く場合、特定のファイル属性を定義している必要があります。ファイル属性は、PL/I ソースプログラムの OPEN 文か DECLARE 文で定義できます。または、File_Map エントリの 5 番目のフィールドにフルパス名があるレコード属性で定義します。レコード属性ファイルの形式については、第 2 章を参照してください。

ノードの現在の日付の取得

標準の Open PL/I 日付/時刻ルーチンでは、ノードを構成するときに Sun MBM システム日付を返します。このノードを構成しない場合は、システムの現在の日付が返されます。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。

アプリケーションプログラムへのファイルの割り当て

IBM メインフレーム環境から移行したアプリケーションプログラムでは、プログラムを実行する JCL で指定したように、アクセスするファイルの内部ファイル名をデータセット名に割り当てます。

コード例 8-12　COBOL プログラムでの SELECT 文を使ったファイルへのアクセス

```
...
SELECT INPUT-FILE ASSIGN TO INFILE.
...
```

PROG1 の実行に使用される MVS JCL は次のとおりです。

```
...
//STEP01X EXEC PGM=PROG1,REGION=512K
//INFILE DD DSN=INPUT.TRANS,DISP=SHR
...
```

PROG1 の実行に必要な VSE JCL は次のとおりです。

```
..
// DLBL INFILE,'INPUT.TRANS'
//EXEC  PGM=PROG1
..
```

▼ ファイルの適切な割り当てを保証する

1. **プログラムが次のように実行される、サブシステムの File_Map を更新します。**
   - IBM メインフレームプログラムでは、適切なトランスレータを使って検査モードで実行する JCL を処理し、割り当てられたファイルに File_Map エントリを生成します。必要な場合、エントリを変更します。詳細は、第 2 章を参照してください。

- 新しいプログラムでは、アクセスされる各ファイルに、一意のデータセット名 (フィールド 1) とタイプ (フィールド 3) と割り当て名 (フィールド 4) を持つ File_Map エントリが必要です。必要な File_Map ファイルタイプとその予期されるファイル名の値は次のとおりです。

  ネイティブ言語の索引編成ファイル、相対編成ファイル、または順編成ファイルの場合

  - タイプ: FS
  - 割り当て名: ファイルの絶対システムパス名

  VSAM KSDS、RRDS、または ESDS ファイルの場合

  - タイプ: VS
  - 割り当て名: Sun MTP FCT で定義されたファイル名

  詳細は、第 2 章を参照してください。

2. **アプリケーションプログラムを実行するジョブの場合、プログラム内のファイルを割り当てる必要があります。**

   たとえば、COBOL プログラムの SELECT 文の ASSIGN 句に指定するファイル名または PL/I プログラムの OPEN 文の TITLE パラメータが ACCFILE の場合、Sun MBM トランスレータは、IBM JCL で割り当てられたファイル用に生成された出力に、必要な変数を自動的に設定します。

   プログラム PROG1 を実行する MVS JCL では、DD 文は ASSGNDD マクロに変換されます。実行時にこのマクロは、内部で $DD_INFILE を INTRANS に設定します。

```
...
ASSGNDD ddname='INFILE' datasetname='INPUT.TRANS' \\\
       type='VS' filename='INTRANS' disp='i' \\\
       normal='k' abend='k'
...
```

   PROG1 を実行する VSE JCL では、DLBL 文は dostrans により ASSGNDD マクロに変換されます。実行時にこのマクロは、内部で $DD_INFILE を INTRANS に設定します。

```
...
ASSGNDD ddname='INFILE' datasetname='INPUT.TRANS' \\\
       type='VS' filename='INTRANS' disp='i' \\\
       normal='k' abend='k'
...
```

   ファイル INFILE に関連してトランスレータで使用される値は、INFILE 用にカスタマイズされた File_Map エントリから抽出されます。どちらの場合も、$DD_INFILE の値は INTRANS (Sun MTP VSAM ファイル) です。

3. **アプリケーションプログラムが外部の割り当てファイルにアクセスする場合、次のオプションのいずれかでアプリケーションプログラムをコンパイルする必要があります。**

   Server Express: `-C "assign=EXTERNAL"`

   ACUCOBOL-GT: `"--fileAssign=external"`

   プログラムで定義したファイルの物理ロケーションを割り当てるために、適切な変数が設定され、COBOL に渡されます。`PROG1` の `INFILE` の `SELECT` 文は、VSAM データセット `INTRANS` にアクセスするための `$DD_INFILE` を設定します。

   `ASSIGN` 句がハイフンを含むファイル名を参照する場合、COBOL で割り当てられる環境変数名の値は、最後のハイフンのあとのテキストです。たとえば、次の形式の `SELECT` 文があるとします。

   ```
   SELECT XFILE ASSIGN TO AAA-BBB-XXX
   ```

   この文により、COBOL では、`XFILE` のファイルにアクセスする環境変数 `$DD_`*XXX* を解決します。詳細は、236 ページの「移行した COBOL アプリケーションプログラムからのファイルのアクセスに必要なコンパイルオプション」を参照してください。

Sun MBM 日付の設定

COBOL または PL/I アプリケーションプログラムで取得できる Sun MBM 日付を変更するには、次の方法があります。

- BAM を使用して Sun MBM システム日付を変更します。この日付は、Sun MBM ジョブ日付、Sun MBM システムメッセージ、およびジョブアカウンティングに影響します。
- JCL で、Sun MBM ジョブ日付を設定します。特定のジョブまたはステップに影響します。

  MVS JCL の場合、次の形式で `JOBDATE` 環境変数を設定します。

  MM/*DD*/*YYYY*

次に例を示します。

```
//JOBDATE JOB
//STEP01 EXEC PGM=TEST2048
//*EBMSYSCMD setenv JOBDATE 08/30/2002
//STEP02 EXEC PGM=TEST2049
```

このジョブが `mvstrans -s` オプションで変換されると、`setenv` 文がジョブ出力に生成されます。上記の例では、バッチシステム日付を使って `TEST2048` を実行し、バッチジョブ日付 `08/30/2002` を使って `TEST2049` を実行します。

■ VSE JCL の場合、SET 文の DATE オペランドを使用します。次に例を示します。

```
// JOB JOBDATE
// EXEC PGM=TEST2048
// SET DATE=08/30/2002
// EXEC PGM=TEST2049
```

このジョブを実行すると、SET 文により、バッチジョブ日付 08/30/2002 を使って TEST2048 が実行されます。

バッチジョブ日付は、アプリケーションプログラムで使用される内部日付だけに影響します。Sun MBM システム日付、ジョブアカウンティング、または Sun MBM システムメッセージには影響しません。

注 – COBOL プログラムでは、デフォルトで EBM_CURRENT_DATE ルーチンだけがこの日付を返します。標準の COBOL 日付/時刻ルーチンでは、BAM を使ってサブシステムを構成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。

$SYSOUTDIR ディレクトリへの出力の書き込み

BAM を使用してサブシステムを作成する際に、sysout ファイル ($SYSOUTS) の場所を指定すると SYSOUTDIR 環境変数が作成されます。Sun MBM は、サブディレクトリ sysoutdir をそのパスに自動的に追加します。たとえば、sysout の場所として /myhome を BAM プロンプトに入力した場合、生成された sysout ファイルは、ディレクトリ /myhome/sysoutdir ($SYSOUTDIR) に書き込まれます。各ジョブに対して、Sun MBM は、$SYSOUTDIR の下に *jobname* ディレクトリを作成します。各 *jobname* ディレクトリでは、ジョブステップ名を使ってディレクトリを作成します。

- 手続きではなくジョブ内で生成された sysout ファイルの場合、sysout ファイルをジョブステップ名ディレクトリの下に配置します。
- 呼び出し元の手続き内で生成された sysout ファイルの場合、手続きを呼び出したステップ名を使用してジョブステップ名ディレクトリの下にディレクトリを作成します。
- アスタリスクのサブパラメータ (たとえば、//DD1 DD SYSOUT=*) によって生成された sysout ファイルの場合、sysout ファイルは作成されますが、出力がジョブ履歴ファイルに書き込まれると削除されます。

次の例では、$SYSOUTDIR=/XYZ/sysoutdir の場合に、DD 文がジョブ (手続きではない) で参照され、DD 文で SYSOUT クラスを指定します。

```
//ABC JOB
//STEPA EXEC PGM=JOBA
//DD1 DD SYSOUT=(9,...)
```

次のディレクトリが生成されます。

```
/XYZ/ABC/STEPA/DD1_${JON}
```

注 – コード例 8-13 に示す $PUBLIC/bin/OW シェルスクリプトを編集すると、このファイル名を変更できます。

コード例 8-13 OW スクリプトファイル (1 / 4)

```
#!/bin/ksh
ebm_print_sysout()
{
cat $SysoutListPrint | while read linein
do
   unset FILENAME
   unset DEST
   unset PRIORITY
   unset tmpfile
   FILENAME=‘eval echo $linein | cut -f1 -d’ ’’
   export FILENAME
   for args in $linein
   do
      export args
      case $args in
         copies=*)
           COPIES=‘eval echo $args | cut -f2 -d’=’’
           export COPIES
       ;;
           dest=*)
              DEST=‘eval echo $args | cut -f2 -d’=’’
              export DEST
       ;;
```

コード例 8-13　　OW スクリプトファイル (2 / 4)

```
                priority=*)
            PRIORITY='eval echo $args | cut -f2 -d''
            export PRIORITY
       ;;
      esac
   done
#
# see if there are some specific printer control commands
# associated with the destination printer
#
   tmpfilecreate=no; export tmpfilecreate
   if [ -d "$PUBLIC/prtctl" ]
   then
      if [ -f "$PUBLIC/prtctl/$DEST" ]
      then
         tmpfile='ebmsed $EBMTMPDIR/ebm_ow.${EBMSYS}.${JON}.$$'
         tmpfilecreate=yes ; export tmpfilecreate
         cat $PUBLIC/prtctl/$DEST > $tmpfile
         if [ $?-ne 0 ]
         then
            echo "OW:ERROR creating temporary file $tmpfile"
            exit 1
         fi
         cat $FILENAME >> $tmpfile
         if [ [ $?-ne 0 ]
         then
            echo "OW:ERROR creating temporary file $tmpfile"
            exit 1
         fi
         FILENAME=$tmpfile
         export FILENAME
      fi
   fi
   # invoke the print specification file
   #
   . $printspec
   if [ ${tmpfilecreate} = "yes" ]
   then
      rm $tmpfile
   fi
done
return
}
################################################################################
```

コード例 8-13　　OW スクリプトファイル (3 / 4)

```
if [ -f $1 ]
then
    IMPFILE=$1
    cp $IMPFILE $SYSOUTDIR/$IMPFILE
    echo "Report routed to " $SYSOUTDIR/$IMPFILE
    rm -f $IMPFILE
else
    IMPDIR=$1
    if [ -d $IMPDIR ]
        then
      rmdir $IMPDIR >$DEVNULL 2>&1
      if [ $?-eq 0 ]
    then
         echo "No files to be printed"

 else
         echo "Files to be printed:"
         case $BATCH_MODE in
         VSE)
         SysoutListPrint=${SYSOUTDIR}/${JOBNAME}/JOB_${JON}
/SysoutListPrint_${JOB_STIME}
          export SysoutListPrint
            if [ -f "$SysoutListPrint" ]
            then
               if [ -s "$SysoutListPrint" ]
               then
                  cat $SysoutListPrint
       ###########################################################
       ...
       ...
       ###########################################################
       # to actually print files, uncomment the following lines
       #
       # if [ -s $SysoutListPrint ]
       # then
       #    ebm_print_sysout
       # fi
       ###########################################################
               else
                  echo "OW:No sysout files to process"
               fi
            else
               echo "OW:No sysout files to process"
            fi
       ;;
       MVS)
           find $IMPDIR -print | egrep "_$JON$" | while read FILE
```

コード例 8-13　　OW スクリプトファイル (4 / 4)

```
          do
        ###################################################################
        # Uncomment the following two lines if you want to rename your
        # report with a date.time suffix
        ###################################################################
        #  mv $FILE $FILE.$JOB_STIME
        #  FILE=${FILE}.${JOB_STIME} ; export FILE

              if [ $VALIDATE = "y" ]
               then
                  echo $FILE DELETED
                  rm $FILE
               else
                   echo $FILE
               fi
        ###################################################################
        # Uncomment the following lines if you want to delete empty
        # report files.
        ###################################################################
        #  if [ ! -s $FILE ]
        #  then
        #     echo ""
        #     echo "Empty file removed:$FILE "
        #     echo ""
        #     rm -f $FILE
        #  fi
        ###################################################################
               done
          ;;
       esac
       fi # if [ $?-eq 0 ] (files to be printed
   else
        echo "  No files to be printed"
   fi # if [ -d $IMPDIR ]   argument was not file or directory

fi # if [ -f $1 ]
exit 0
```

次は、printspec ファイルの例です。

コード例 8-14　　printspec ファイル

```
#!/bin/ksh
#
# this is a printspec file for using the basic UNIX lp command
#
dest='' ; export dest
priority='' ; export priority
#
if [ "a$DEST" != "a" ]
then
   dest="-d$DEST" ; export dest
else
   if [ "a$DEFAULTDEST" != "a" ]
   then
      dest="-d$DEFAULTDEST" ; export dest
   else
      echo "ERROR printspec.LP:No lp destination specified"
   fi
fi
if [ "a$COPIES" = "a" ]
then
   COPIES=1 ; export COPIES
fi
if [ "a$PRIORITY" != "a" ]
then
   priority="-p$PRIORITY" ; export priority
else
   if [ "a$DEFAULTPRIORITY" != "a" ]
   then
      priority="-p$DEFAULTPRIORITY" ; export priority
   fi
fi

lp -n$COPIES $dest $priority <$FILENAME
```

次の例では、$SYSOUTDIR=/XYZ/sysoutdir の場合に、DD 文がジョブ (手続きではない) で参照され、DD 文で SYSOUT クラスを指定しません。

```
//ABC JOB
//STEPA EXEC PGM=JOBA
/DD1 DD SYSOUT=(,...)
```

次のディレクトリが生成されます。

```
/XYZ/ABC/STEPA/DD1_${JON}
```

ddname はファイル名として使用されます。

ジョブで呼び出される手続きから SYSOUT が生成される場合、追加ディレクトリが生成されます。

```
//ABC JOB
//STEPA EXEC PROC=BBB
              .
              .
//BBB PROC
//STEP01 EXEC PGM=JOBA
//DDA DD SYSOUT=(9,REP1)
```

次のディレクトリが生成されます。

```
/XYZ/ABC/STEPA/STEP01/DDA_${JON}
```

- COPIES および SYSOUT サブパラメータクラスと form_name に対して何も処理は行われません。
- コード例 8-13 に示す $PUBLIC/bin にある OW シェルスクリプトにファイルが渡されます。
- スクリプトへの入力はファイルまたはディレクトリです。ディレクトリは、ジョブ名ディレクトリを示す OW スクリプトに渡されます。次に例を示します。

  ```
  OW ${SYSOUTDIR}
  ```

- 出力は、関連ファイルに書き込まれます。ジョブが中止した場合、出力ファイル等は削除されません。
- ジョブをもう一度実行した場合、ddname sysout ファイルは、同じジョブ名ディレクトリの下に配置されます。ただし、${JON} 接尾辞により、前のレポート名および ddnames とは区別されます。
- DATE.TIME 接尾辞を持つレポートの名前を変更する場合、OW スクリプトの該当する行のコメントを解除します。詳細は、コード例 8-13 を参照してください。

検査モード (-v　オプション) でジョブを実行する場合、レポートは削除されます。

ジョブシーケンス

ジョブシーケンスは、特定の順序でジョブを実行する機能です。たとえば、複数のジョブストリームを並行して実行できます。この場合、各ジョブストリームはジョブの特定のシーケンスです。ジョブのシーケンスは、前のジョブのシーケンスでの状態によって違うことがあります。

すべてのジョブにバッチシェルスクリプトを作成 (変換または手動作成) したあと、ジョブシーケンスを制御するバッチシェルスクリプトを作成できます。

例

ジョブストリーム A には joba1、joba2、joba3、および joba4 が含まれているとします。joba1 が最初に実行され、このジョブの復帰状態が成功の場合は、次に joba2 という順で実行されます。

ジョブストリーム B には jobb1、jobb2、jobb3、および jobb4 が含まれています。jobb1 が最初に実行され、このジョブの復帰状態が成功の場合は、jobb2、jobb3 という順で実行されます。

各ジョブストリームで、前のジョブが異常終了した場合、ジョブストリームは失敗して終了します。A と B のジョブストリームを並行して実行できます。ジョブストリーム A では、バッチシェルスクリプトに次のようなエントリが含まれています。

コード例 8-15　　ジョブシーケンス (1 / 2)

```
cd $HOME                          (jmvs、ish などのディレクトリがホームディレクトリ
                                  の下に作成され、デフォルトの Sun MBM サブシステム
                                  にアクセスできるものとします。)
unikixjob joba1 -w -ca            (joba1 をサブミットし、ジョブが終了するのを待ちま
                                  す。)
setenv ret $status                (ret 変数に joba1 の復帰状態を設定します。)
if ($ret != 0) then               (joba1 の復帰状態を確認します。)
 echo "job joba1 aborted"         (情報メッセージを出力します。)
    exit 1                        (失敗を示すリターンコードでシェルスクリプトを終了し
                                  ます。)
endif
echo "job joba1 terminated successfully"
unikixjob joba2 -w -ca            (次のシーケンスのジョブをサブミットします。)
setenv ret $status
if ($ret != 0) then
 echo "job joba2 aborted"
    exit 1
endif
echo "job joba2 terminated successfully"
```

コード例 8-15　　ジョブシーケンス (2 / 2)

```
unikixjob joba1 -w -ca          (次のシーケンスのジョブをサブミットします。)
 .
 .
```

ジョブストリーム B も同様にコーディングされます。

1 つのジョブシーケンスを実行するには、少なくとも 2 つのアクティビティーが必要です。たとえば、ジョブストリーム A を実行するには、実行を制御するジョブを実行するアクティビティーと、このジョブによりサブミットされたジョブを実行するアクティビティーが必要です。

ジョブストリームの実行を制御するジョブは、ストリームのすべてのジョブが実行されるまで、またはジョブの 1 つが中止し、失敗して終了するまで、アクティビティー内に常駐しています。サブミットしたジョブと同じクラスまたは別のクラスでのアクティビティーに、実行を制御するジョブを割り当てることができます。

たとえば、コード例 8-15 のプログラムでは、joba1、joba2、joba3、および joba4 の実行を制御するジョブが、これらのジョブをサブミットするときにジョブをクラス a に割り当てます。ジョブは、クラス c またはクラス a に割り当てることができます。どちらの場合も、少なくとも 2 つのアクティビティーを使用可能にする必要があります。この場合、クラス a に 1 つのアクティビティー、クラス c に 1 つのアクティビティー、あるいはクラス a に 2 つのアクティビティーが必要です。この方法で、ジョブシーケンスをいくつでも実行できます。

ジョブの同期化

ジョブの同期化は、ジョブの実行順序を制御する機能です。この機能は、ジョブシーケンスとは違います。たとえば、2 つのジョブを並行して実行できますが、どちらも 3 番目のジョブの実行が始まる前に終了する必要があります。ジョブを同期化するには、バッチシェルスクリプトと Sun MBM コマンドを使用します。

たとえば、図 8-6 に示すように、次のシナリオがあるとします。

- ジョブ R4STRT は他のジョブが開始される前に終了する必要があります。
- R4STRT の終了後、ジョブ R4110 と R4111 を並行して実行します。
- R4110 と R4111 の終了後、ジョブ R4112 を実行します。
- R4112 の終了後、ジョブ R4113 を実行します。
- R4113 の終了後、ジョブ R4114 および R4115 を実行します。
- R4114 の終了後、ジョブ R4LST を実行します。

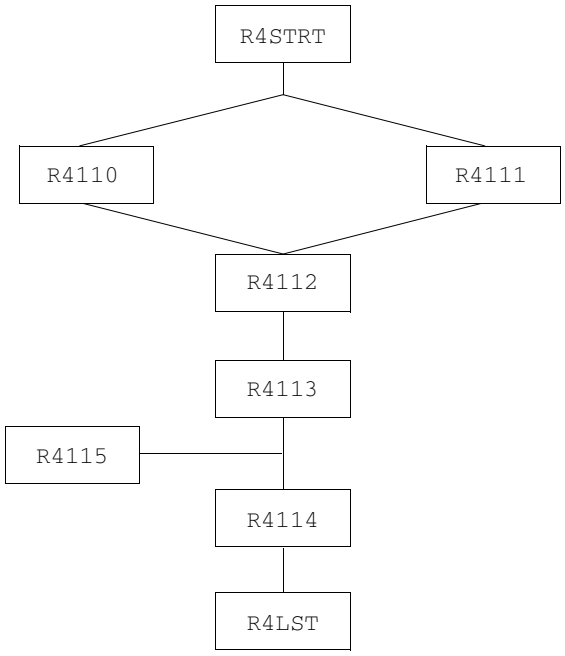


図 8-6　　ジョブの同期化

次の例のバッチシェルスクリプトは、ジョブの同期化を実行します。

コード例 8-16　　ジョブ同期化シェルスクリプトの例 (1 / 2)

```
unikixjob R4STRT -w
   if ( $status == 0 )then
       setenv R4110_NUM `unikixjob R4110 -j‘
       setenv R4111_NUM `unikixjob R4111 -j‘
       insjbl $R4110_NUM $R4111_NUM -w
   if ($status != 0) then
       echo "Error in job R4110 or R4111"
       exit
   endif
unikixjob R4112 -w
   if ( $status == 0) then
       unikixjob R4113 -w
       unikixjob R4115
       unikixjob R4114 -w
       unikixjob R4LST
   else
       echo "Error in job R4112"
       exit
   endif
   else
```

コード例 8-16　　ジョブ同期化シェルスクリプトの例 (2 / 2)

```
        echo "Error in job R4STRT"
        exit
    endif
```

この例では、少なくとも 2 つのアクティビティーを作成する必要があります。ただし、ジョブが並行して実行されるか、または同期しないで実行される場合は、3 つのアクティビティーを作成する必要があります。3 つのアクティビティーのうち 1 つはドライバプログラムで、2 つは R4110 および R4111 など、非同期のジョブを実行するアクティビティーです。

あとのジョブは前のジョブが終了すると始まるので、スクリプトでは、待機オプションである -w を指定して R4STRT ジョブをサブミットします。ジョブが終了するまで、シェルスクリプトに制御は戻りません。これにより、ジョブ R4110 および R4111 がサブミットされる前に、R4STRT は確実に終了します。

R4STRT の状態が成功の場合、ジョブ R4110 および R4111 がサブミットされます。R4110_NUM および R4111_NUM 環境変数には、それぞれのジョブのジョブオカレンス番号が設定されます。これにより待機オプションを指定して insjbl コマンドを発行できます。ジョブ R4110 および R4111 が終了するまで、ドライバプログラムに制御は戻りません。いずれかのジョブが中止すると、insjbl コマンドは、0 以外の状態を返します。

ジョブ R4110 および R4111 が終了したあと、ジョブ R4112 をサブミットできます。このジョブは、待機オプションを指定してサブミットされます。したがって、このジョブは、他のジョブをサブミットする前に終了する必要があります。

ジョブ R4113 が正常に終了すると、ジョブ R4115 をサブミットできます。このジョブの終了で始まるジョブはないので、このジョブには待機オプションは必要ありません。ジョブ R4LST はジョブ R4114 が終了すると始まるので、待機オプションを指定してサブミットされます。

このジョブの終了後に始まるジョブがないので、ジョブ R4114 が終了したあと、待機オプションを指定しないで R4LST をサブミットできます。

この例には、さまざまなバリエーションがあります。適切なシェルスクリプトを作成してジョブを同期化するには、サイトの要件とジョブの依存性を判定する必要があります。

第9章

EXCI インタフェースの使用法

この章では、Sun MBM が外部 CICS インタフェース (EXCI) に対するサポートを実装する方法について説明します。ECI クライアント (Sun MTP Client ソフトウェア) を使用すると、Sun MBM バッチジョブは、分散プログラムリンク (DPL) 要求を Sun MTP 領域に送り、トランザクションを実行できるようになります。次のトピックがあります。

概念

EXCI サポートでは、バッチジョブは Sun MTP 領域でプログラムを実行できます。これは、EXCI サポートなしでバッチジョブが領域リソースにアクセスする方法とは異なります。EXCI を使用しない場合、バッチジョブは、領域の VSAM カタログで定義された VSAM ファイルでのみ機能します。EXCI を使用すると、バッチジョブによって領域でプログラムを実行できます。プログラムは、別のプログラムの実行や VSAM 以外のファイルの更新など、さまざまな作業を実行できます。さらに、EXCI 実装では、バッチサブシステムが複数の領域にアクセスできます。たとえば、1 つのサブシステムを作成するだけで、EXCI をサポートしながら、Sun MTP 領域に通常どおりに接続できます。

図 9-1 では、EXCI 実装が単一のシステムでどのように動作するかを示します。現在の環境では、Sun MTP および Sun MBM は異なるシステムの場合があります。ただし、Sun MTP Client は、Sun MBM と同じシステムにインストールされている必要があります。

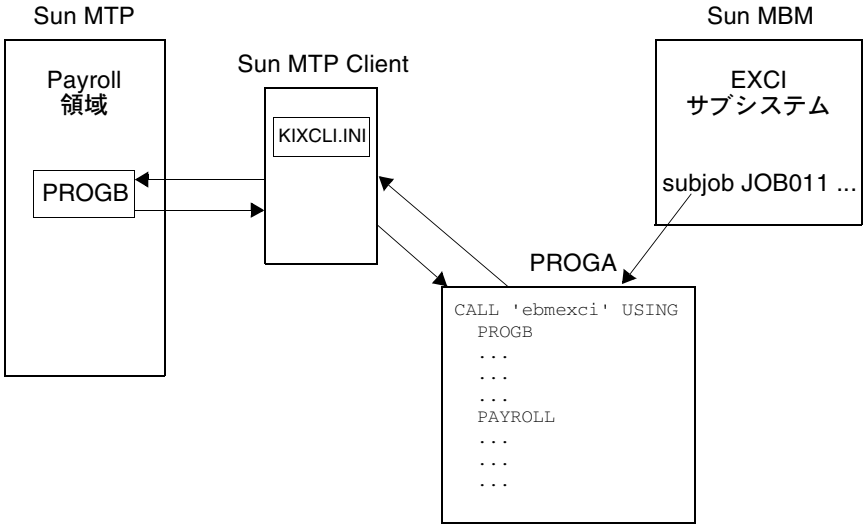


図 9-1　　EXCI 実装

図 9-1 は、次のイベントを示しています。

1. JOB011 は、EXCI サポートが組み込まれた Sun MBM サブシステムにサブミットされます。ジョブは、unikixjob コマンドではなく、subjob コマンドでサブミットされるので注意してください。
2. JOB011 はプログラム PROGA を実行します。
3. PROGA は、Sun MTP 領域で定義されたプログラム PROGB を呼び出します。
4. サブシステムは、Sun MTP Client に要求を送信します。
5. Sun MTP Client は、構成ファイル (KIXCLI.INI) で領域を検出し、要求を転送します。
6. PROGB が領域で実行され、Sun MTP Client によって PROGA に戻り値が返されます。

必要なソフトウェア

現在のアプリケーション環境で EXCI サポートを実装するには、次のソフトウェアをインストールする必要があります。いずれも最低限のリリースです。

- Sun MBM 10.1.0
- Sun MTP Client 7.2 (パッチレベル 1 以上)
- Sun MTP 8.1.0

アプリケーション環境の準備

次の作業を実行して、アプリケーション環境を準備する必要があります。

- TCP/IP 対応にし、ECI/EPI クライアントを受け入れるように Sun MTP 領域を構成します。『Sun MTP Client ユーザーズガイド』で、Sun MTP Client への TCP/IP 接続に対応するように Sun MTP を構成する方法に関する節を参照してください。
- Sun MTP Client の `KIXCLI.INI` ファイルで、対象システム (元の COBOL プログラムの `APPLID` と同じ) を定義します。294 ページの「`KIXCLI.INI` ファイルを更新する」を参照してください。詳細は、『Sun MTP Client ユーザーズガイド』の、Sun MTP Client を構成する方法に関する節を参照してください。
- EXCI インタフェースを使用する COBOL プログラムを変更します。詳細は、294 ページの「COBOL プログラムの変更」を参照してください。
- EXCI サブシステムを作成します。詳細は、297 ページの「EXCI サブシステムの作成」を参照してください。
- Sun MTP、Sun MTP Client、および Sun MBM ノードを起動します。クライアントおよび領域を起動する方法については、『Sun MTP Client ユーザーズガイド』で Sun MTP Client と Sun MTP の起動に関する章を参照してください。ノードを開始および停止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。

▼ KIXCLI.INI ファイルを更新する

このファイルの編集方法については、『Sun MTP Client ユーザーズガイド』を参照してください。

1. **KIXCLI.INI ファイルを開きます。**
2. **COBOL プログラムを実行する領域用のエントリを作成します。**

   次に例を示します。

```
Payroll=TCP,def.mycompany.com,9111,Employee Payroll System
```

   エントリのフィールドは、次のとおりです。

   - 領域の名前: COBOL プログラムで指定した対象システム (APPLID) の名前です。
   - トランスポートプロトコル: TCP
   - ホストアドレス: 領域が実行されているシステムの TCP/IP アドレス。
   - ポート番号: 領域が Sun MTP Client からの TCP/IP 接続を待機しているポート番号。Sun MTP 領域は、unikixmain に -P オプションを付け、リスナーを使用可能にするこのポート番号を指定して起動する必要があります。
   - コメント: オプションのコメントフィールド。

   すべてのフィールドが必須です。コメントを入力しない場合でも、ポート番号フィールドのあとにコンマ区切り記号を入力する必要があります。

3. **編集を終了したら、ファイルを保存して閉じます。**

COBOL プログラムの変更

EXEC CICS LINK コマンドを COBOL CALL 文に置き換えて、EXCI インタフェースで領域にアクセスする COBOL プログラムを変更する必要があります。次の例は、元の COBOL 形式を示します。

```
EXEC CICS LINK
     PROGRAM (TARGET-PROGRAM)
     RETCODE (EXCI-EXEC-RETURN-CODE)
     LENGTH (LINK-COM-LEN)
     DATALENGTH(LINK-DAT-LEN)
     APPLID (TARGET-SYSTEM)
     TRANSID(TARGET-TRANSID)
     COMMAREA (COMMAREA)
     SYNCONRETURN
```

次の例は、COBOL CALL 文の形式を示します。

```
CALL 'ebmexci' USING
     TARGET-PROGRAM,
     EXCI-EXEC-RETURN-CODE,
     LINK-COM-LEN,
     LINK-DATA-LEN,
     TARGET-SYSTEM,
     TARGET-TRANSID,
     COMMAREA,
     SYNCONRETURN.
```

CALL 文で使用する ebmexci API は、位置に依存します。表 9-1 では、簡単な説明とともに位置および変数名を示します。

表 9-1 ebmexci アプリケーションプログラミングインタフェース

位置	変数	必須/任意	説明
1	TARGET-PROGRAM	必須	プログラム名。
2	EXCI-EXEC-RETURN-CODE	必須	リターンコード (20 バイト)。
3	LINK-COM-LEN	COMMAREA が NULL でない場合は必須	COMMAREA の長さ。
4	LINK-DAT-LEN	任意	COMMAREA のデータの長さ。 使用しない場合は、OMITTED に設定されます。
5	TARGET-SYSTEM	任意	使用しない場合は、OMITTED に設定されます。値が指定されていない場合は、Sun MTP Client のデフォルトシステムがデフォルトです。
6	TARGET-TRANSID	任意	使用しない場合は、OMITTED に設定されます。現在、TRANSID はこの実装ではサポートされていません。
7	COMMAREA	任意	使用しない場合は、OMITTED に設定されます。COMMAREA が使用されている場合は、LINK-COM-LEN を設定する必要があります。これは、データを送受信するための通信領域です。
8	SYNCONRETURN	必須	NULL 以外の値にする必要があります。

コード例 9-1 では、EXEC CICS LINK 文と CALL 文を指定した COBOL プログラムの一部を示します。通常は、LINK 文を削除するかコメントアウトします。

コード例 9-1 ebmexci API を含む COBOL プログラム

```
WORKING-STORAGE SECTION.
 01  WS-EXCI-RETURN-CODE-AREA.
     05  WS-EXCI-RC.
         10  WS-EXCI-RC-RESP     PIC S9(08) COMP.
         10  WS-EXCI-RC-RESP2    PIC S9(08) COMP.
         10  WS-EXCI-RC-ABCODE   PIC  X(04).
         10  WS-EXCI-RC-MSGLEN   PIC S9(08) COMP.
         10  WS-EXCI-RC-FILL     PIC  X(08) .
         10  WS-EXCI-RC-MSGPTR   POINTER.
 01  WS-PARM-AREA.
     05  WS-PARM-APPLID          PIC  X(08).
     05  WS-PARM-PROGRAM         PIC  X(08).
     05  WS-PARM-OTHER.
         10  WS-PARM-2ND-ADJ     PIC  X(01).
         10  WS-PARM-BATCH       PIC  X(08).
 01  WS-COMM-AREA.
     05  WS-COMM-DATA            PIC X(2000).
     05  WS-COMM-DATA-LEN        PIC S9(04) COMP.

*-----COBOL------
*     EXEC CICS LINK
*               PROGRAM  (WS-PARM-PROGRAM)
*               COMMAREA (WS-COMM-AREA)
*               LENGTH   (WS-COMM-DATA-LEN)
*+               APPLID   (WS-PARM-APPLID)
*               RETCODE  (WS-EXCI-RC)
*               SYNCONRETURN
*    END-EXEC
*------END COBOL-----

*------BEGIN MBM-EXCI------
           CALL 'ebmexci' USING WS-PARM-PROGRAM,
                                WS-EXCI-RC
                                WS-COMM-DATA-LEN,
                                OMITTED,
                                WS-PARM-APPLID,
                                OMITTED,
                                WS-COMM-AREA,
                                '1'.
*-------END MBM-EXCI-----
```

EXCI サブシステムの作成

作成している Sun MBM EXCI サブシステムは Sun MTP 領域と通信しますが、Sun MTP Client ソフトウェアを介して領域に接続するように構成します。

EXCI と通常の領域接続の両方をサポートするようにサブシステムを構成するには、サブシステムの作成時に「Data Management」画面で Sun MTP オプションのいずれかを選択します。また、Sun MTP Client `KIXCLI.INI` ファイルで複数の領域も定義できます。Sun MBM は、`ebmexci` API を介してそれらの領域にアクセスできます。

ジョブが `JOB` カードで `USER` および `PASSWORD` パラメータを使用している場合、これらのパラメータがどのように変換されるかについては、53 ページの「`JOB` 文」を参照してください。

サブシステムの構成で EXCI のみをサポートしている場合は、`subjob` コマンドでジョブをサブミットします。サブシステムの構成で EXCI と Sun MTP VSAM の両方をサポートしている場合は、`unikixjob` コマンドでジョブをサブミットします。

▼ EXCI サブシステムを作成する

1. **ノードの環境を指定します。**
2. **次のいずれかの方法でバッチ管理マネージャー (BAM) を起動します。**
   - コマンドプロンプトで、`bam` コマンドを実行します。
   - Sun MBM メインメニュー (`ebmx` コマンド) を開いて、「BAM」アイコンをクリックします。
3. **メインメニューで、オプション「`3` Applications & Subsystems」を選択します。**
4. **「Applications & Subsystems」メニューで、オプション「`3` Create a Subsystem」を選択します。**
5. **プロンプトに新しいサブシステム名を入力し、Return キーを押します。**
6. **「Applications & Subsystems」メニューで、オプション「`4` Third Party Packages」を選択します。**
7. **オプション「`3` EXCI」を選択します。**

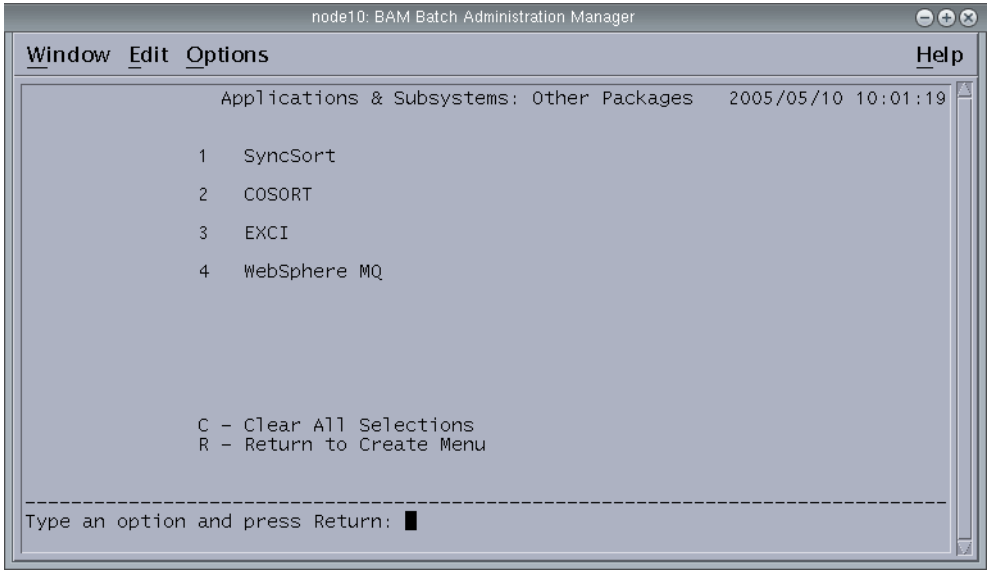


図 9-2　　EXCI オプションの選択

矢印が EXCI オプションを指している場合は、EXCI が選択されています。

8. 必要な Sun 以外の製品を選択し、Return キーを押して選択内容を確定し、「Applications & Subsystems」メニューに戻ります。
9. サブシステムで必要なその他の項目を選択します。
10. 準備が完了したら、オプション「C」を選択し、サブシステムを構築します。
11. Sun MTP Client のインストールディレクトリを指定するように要求されたら、ソフトウェアをインストールしたディレクトリのフルパス名を入力します。
12. 『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』の説明にあるように、構築プロセスを完了します。

付録 A

バッチシェル

Sun MBM ジョブスクリプトは、Sun MBM マクロ、C シェル文、システムコマンド、および組み込み型のバッチシェルコマンドです。これらのスクリプトは、バッチシェルコマンド `btsh` により実行されます。この章では、バッチシェルおよびバッチシェル組み込みコマンドの機能について説明します。この章の内容は、次のとおりです。

Sun MBM マクロの説明は、『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。

C シェル文とシステムコマンドの説明は、ご使用のシステムマニュアルを参照してください。

注 – この付録でバッチシェルに触れている場合はすべて、Sun MBM 製品の一部であるバッチシェルを表します。

バッチシェルの機能

C シェル機能に加えて、バッチシェルには次の機能があります。

- 現在実行しているステップとコマンドを表示します
- 管理コマンドを使用するステップのあとでバッチジョブを停止します
- ステップ内で実行中のコマンドを中止します
- 実行チェーンの各ステップのバッチジョブの状態を表示します
- 検査モードでバッチシェルスクリプトを実行します
- 指定したステップからジョブを再起動します
- 指定したステップで中断します

- ユーザーのコンソールからのデータを受け入れます
- Sun MBM コンソールにメッセージを表示します
- ファイルモード作成マスクを設定または取得します

組み込みコマンド

Sun MBM バッチシェルでは、すべての C シェル構造とシステムコマンドをサポートします。また、バッチシェルスクリプトの実行中に、Sun MBM で実行を制御し、状態を取得できるコマンドもサポートします。

accept

`accept` コマンドは、ジョブを実行するときにユーザーからデータを動的に受け入れる場合に使用します。

形式:

```
accept xxx
```

バッチシェルでは、指定したバッチジョブ番号の `rpljob` コマンドが発行されるのを待ちます。`rpljob` については、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

ユーザーが次のコマンドを実行したあと、

```
[node1/fs1] # rpljob jon string-of-data
```

string-of-data の値が変数 *xxx* に割り当てられます。この値は、制御変数 `&xxx` を参照して取得されます。

display

`display` コマンドは、メッセージを Sun MBM `HistoryFile` ファイルと `$SYSINDIR/ebm_console` ファイルに経路指定し、Sun MBM コンソール端末にメッセージを表示する場合に使用します。

形式:

```
display msg="xxx"
```

次の例では、accept と display 組み込みコマンドを使って表示できるメッセージ、およびオペレータが実行中のジョブと対話する方法を示します。オペレータが、ジョブ用のテープをマウントし、マウントしたテープの番号を確認することを前提としています。

MVS JCL には次の行が含まれています。

```
//JCUSTYTD JOB
//STEP1 EXEC PGM=PROGCYTD
//CUSTYTD DD DSN=CUSTOMER.MASTER.YTD,DISP=(OLD,PASS),
//     UNIT=TAPE,LABEL=(SL),VOL=(,RETAIN,SER=T10038)
```

//*EBMSYSCMD 文を使って、JCL を変更します。この文は、ジョブスクリプトに次の行を挿入するようにトランスレータに送ります。

```
//JCUSTYTD JOB
//*EBMSYSCMD display msg="Mount YTD Customer Master tape :T10038."
//*EBMSYSCMD display msg="Enter tape number mounted."
//*EBMSYSCMD accept tape_number
//*EBMSYSCMD display msg="Tape volume mounted is   :&tape_number"
//*EBMSYSCMD tar xvf &tape_number
//STEP1 EXEC PGM=PROGCYTD
//CUSTYTD DD DSN=CUSTOMER.MASTER.YTD,DISP=(OLD,PASS),
//     UNIT=TAPE,LABEL=(SL),VOL=(,RETAIN,SER=T10038)
```

JCL トランスレータを実行し、ジョブをサブミットしたあと、次のジョブメッセージが Sun MBM コンソール端末または Sun MBM コンソールファイルに表示されます。

```
[node1] :*** IP2034(I) Activity act1 is waiting for new jobs
[node1] :*** BS0336(I) BQM:BQM is now idle
[node1] :*** IP2033(I) Activity act1 starts with job 4
[node1] :*** BS0337(I) BQM:BQM is now running jobs
[node1] :*** IP2110(I) Starting job JCUSTYTD at May 05 14:40:04 2005
[node1] :004 IP4305(I) Start of job JCUSTYTD at May 05 14:40:05 2005
[node1] :004 IP4334(I) History filename:/tmp/JCUSTYTD_4.hst.05May2005.14:40
[node1] :004 Mount YTD Customer Master tape :T10038.
[node1] :004 Enter tape number mounted.
[node1] :004 IP4363(I) The job is waiting for reply
```

オペレータが次のコマンドを入力した場合、

```
[node1/fs1] # rpljob 4 T10038
```

次のメッセージが表示されます。

```
[node1] :004 Tape volume mounted is          :T10038
[node1] :004 IP4304(I) Job JCUSTYTD terminating at May 05 14:47:49 2005
[node1] :*** IP2107(I) Job JCUSTYTD terminated at May 05 14:47:49 2005
[node1] :*** IP2034(I) Activity act1 is waiting for new jobs
[node1] :*** BS0336(I) BQM:BQM is now idle
```

status

バッチシェルスクリプトから返される状態の値は、成功した場合は 0 で、失敗した場合は 0 以外です。状態は、呼び出し元シェルスクリプトのタイプによって、さまざまな方法で確認できます。

呼び出し元のシェルスクリプト	最後のコマンドの状態
C	`$ status`
Bourne	`$ ?`
Korn	`$ status` `$ ?`
batch	`$ status` `$ STATUS` (最終ステップの状態) `$ JOBSTATUS` (ジョブの状態)

umask

umask コマンドを使用すると、ファイルモード作成マスクを設定または取得できます。

形式:

```
umask -S | 000
```

説明

-S	ファイルモード作成マスクを表示します。
000	ファイルモード作成マスクを設定します。ユーザーファイル作成モードマスクは、000 に設定されます。この値は、所有者、グループ、およびその他に対する読み取り/書き込み/実行の権限をそれぞれ表します。指定した 8 進数の各値は、ファイル作成用にシステムで指定した対応する値から差し引かれます。

例:

モード 777 で正常に作成されたファイルは、モード 700 になります。

バッチシェル変数およびファイル

Sun MBM トランスレータで JCL からジョブを変換する場合は、特殊な変数およびファイルが使用されます。ジョブステップの境界内で使用されるものもあれば、手続きの境界内で使用されるものもあります。あるいは、ジョブ全体で使用されるものもあります。独自のバッチジョブを作成している場合、特にプログラムがジョブステップと手続きの実行に関係する場合は、これらのエンティティーが役に立ちます。

注 – 未定義の環境変数をバッチジョブで参照すると、ジョブは中止されます。

バッチシェル環境変数

次の小節では、バッチシェル変数について説明します。これらの変数は、どの構成ファイルにも設定できます。構成ファイルと実行階層の詳細は、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

$DCB_ddname

DCB_ddname 環境変数は、File_Map エントリの 5 番目のフィールドに指定されたファイルを指定します。このファイルには、PL/I アプリケーションに必要な情報が含まれています。

$EBM_REPRO_APPEND

サブシステムの $USER_SETUP ファイルで EBM_REPRO_APPEND 環境変数を Y に設定すると、ジョブ内のメインフレーム以外の動作がそのサブシステムで実行されます。これにより、すべての REPRO が呼び出され、出力ファイルが切り捨てられずに付加されるように FS タイプになります。いずれの場合も、ファイル間の直接付加が実行されます。ただし、mfrcdv ファイルでは、付加の前にヘッダーが削除されます。入力ファイルと出力ファイルは同じファイル形式になっている必要があります。ファイル形式が異なる場合、REPRO 呼び出しは失敗します。レコード長など、それ以外のファイル属性に対するチェックは実行されないので注意してください。したがって、ファイルのレコード長が異なっている場合、ファイルが破壊される可能性があります。

注意 – この動作は、EBM_REPRO_APPEND 変数が設定されているサブシステム内でジョブを実行することで呼び出されるすべての REPRO に当てはまります。

$JOBDATE

この環境変数は、ジョブ実行中に設定され、アプリケーションプログラムで使用される内部日付を上書きします。

形式:

MM/DD/YYYY

たとえば、ジョブに次の文を追加します。

```
setenv JOBDATE 08/30/2001
```

$JOBDATE は、アプリケーションプログラムで使用される内部日付だけに影響します。Sun MBM システム日付、ジョブアカウンティング、または Sun MBM システムメッセージには影響しません。ただし、COBOL プログラムでは、デフォルトで EBM_CURRENT_DATE ルーチンだけがこの日付を返します。システムでこの機能をサポートしている場合、標準の COBOL 日付/時刻ルーチンでは、BAM を使ってサブシステムを構成する必要があります。

$JOBID

$JOBID には、Sun MBM により、subjob または unikixjob コマンドの -J オプションで指定した値が設定されます。lstjob または infjbs コマンドでこの値を表示できます。これらのコマンドの詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

$JOBNAME

この環境変数は、Sun MBM によりジョブの名前に設定されます。$JOBNAME は、Sun MBM ジョブを含むファイルの名前です。

$JOBPARM

$JOBPARM を使用すると、ユーザーは 25 文字以内の文字列を Sun MBM ジョブスクリプトに渡すことができます。

$JOBPARM は、unikixjob または subjob コマンドの -P オプションを使って設定されます。JOBPARM 環境変数は、lstjob または infjbs コマンドで表示できます。これらのコマンドの詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

$JOBSTATUS

JOBSTATUS 環境変数には、ジョブの完了コードが含まれています。この変数は、ジョブの開始時に 0 に初期化されます。

$JOBSTATUS の値は、ジョブステップの完了コードが中止を示す 0 以外の値になった場合、1 に更新されます。ジョブが中止されると、すべての後続のジョブステップはバイパスされます。

ジョブステップ完了コードが 0 の場合は、$JOBSTATUS の初期値が維持されます。

ジョブの最後の $JOBSTATUS の値が 0 であれば、ジョブは正常に終了しています。

$JOB_STIME

この環境変数は、subjob および unikixjob コマンドの -h および -o オプションを使用してすべてのレポート名を生成する場合に、ファイル名の作成に使用されます。$JOB_STIME には、Sun MBM により、バッチジョブの実行開始時間が設定されます。これらのコマンドの詳細は、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

$JON

JON 環境変数は、ジョブをサブミットしたときに Sun MBM によりジョブに割り当てられた一意のジョブオカレンス番号です。

$RUNNING_STEPNAME

この環境変数には、現在実行中のステップの名前が含まれます。変換された MVS JCL ストリームでは、$RUNNING_STEPNAME には、EXEC 文のラベルが設定されます。また、手続きで実行される場合は手続きステップ名が設定されます。

変換された VSE JCL では、$RUNNING_STEPNAME は、ジョブまたは手続き内の EXEC 文のオカレンスによって異なる名前に設定されます。

最初の EXEC 文では $RUNNING_STEPNAME は 'STEP0001' に設定され、3 番目の EXEC 文では $RUNNING_STEPNAME は 'STEP0003' に設定されます。

例:

```
RUNNING_STEPNAME=JSTP0001
RUNNING_STEPNAME=JSTP002_PSTP0010
```

バッチシェルファイル

status.${JON}

status.${JON} ファイルは、SORT、IDCAMS、および変換された $STATUS ジョブの手続きステップで、ステップ完了コードの記録に使用されます。値は、$STATUS に設定されます。

また、変換された手続きでは、SORT および IDCAMS ステップで status.${JON} を使用して、完了コードを記録します。

用語集

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 (名詞) 「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

ebmexci API (名詞) IBM EXCI アプリケーションプログラミングインタフェースの Sun MBM 実装。

ebmmd (名詞) 「メッセージデーモン」を参照。

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ある 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

POWER　(名詞) ジョブのスケジューリング、入出力のスプール処理、パーティションの開始および終了を管理する VSE オペレーティング環境のコンポーネント。

psg_daemon　(名詞) 「プロセスグループデーモン」を参照。

R

RJE　(名詞) 遠隔ジョブエントリプロトコル。

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、および 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。クライアントを除く 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。

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。Sun MTP 領域を Sun MBM ノードに接続するとき、このテーブルはノードのインストールディレクトリを含んでいる必要があります。

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。

アニメートする (動詞) Server Express Animator ソフトウェアを使用してジョブをデバッグすること。

え

エラー表 (名詞) UNIX 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

外部 CICS インタフェース (EXCI)
(名詞) CICS 以外の環境から CICS アプリケーションにより簡単にアクセスできるようにするための CICS アプリケーションプログラミングインタフェース。MVS で動作する CICS 以外のプログラム (クライアントプログラム) が、CICS 領域で動作しているプログラム (サーバープログラム) を呼び出したり、通信領域の手段でデータをやり取りしたりすることが可能になります。CICS プログラムが、別の CICS プログラムに関連付けられているように呼び出されます。[1]

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) ユーザーがノードを開始、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。

1. 「IBM Terminology」より。(名詞句) 2004 年 12 月 21 日に下記から転載
   `http://www-306.ibm.com/ibm/terminology/ef.htm`

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 ジョブや手続きを実行し、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や手続きを実行すると、`File_Map` が 1 つ作成されます (既にファイルが存在している場合は、追加される)。

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、他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。

ジョブエントリ制御言語 (JECL) (名詞) ジョブ入力、文の収集とスケジューリング、および出力スプール処理を管理するための POWER JCL ストリームで使用される VSE JCL 文。

ジョブクラス (名詞) 「クラス」を参照。

ジョブ制御コマンド (JCC) (名詞) 文の最初の 2 つの位置に // なしにコーディングされ、ジョブストリームにありうる VSE JCL 文。

ジョブ制御文 (JCS) (名詞) 文の最初の 2 つの位置が // でコーディングされ、ジョブストリームにある VSE JCL 文。

ジョブの順序付け (名詞) ジョブを特定の順番で実行する機能。

ジョブの同期 (名詞) ジョブ実行の順番を制御する機能。ジョブの同期は、ジョブ実行の依存関係を設定できる点で、ジョブの順序付けとは区別されます。たとえば、JOBC を実行する前に、JOBA および JOBB の両方を完了する必要があります。

す

スレッド (名詞) 「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。

つ

ツールキット (名詞) グラフィカルユーザーインタフェースを持つ Sun MBM アプリケーションで、これによりユーザーは `File_Map` を管理したり、JCL ジョブや手続きを検査および変換したり、COBOL プログラムをコンパイルしたりできます。

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`)。

バッチシェル (名詞) C シェル環境のスーパーセットである Sun MBM バッチ実行環境。

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呼ぶ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 ジョブや手続きを実行し、ジョブや手続きのマクロスクリプトを作成すること。

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) 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。

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。

索引

D

E

F

G

H

I

J

K

L

P

T

お

か

き

く

け

こ

し

す

せ

そ

た

つ

て

と

に

の

は

ひ

ふ

へ

ほ

ま

め

も

ゆ

ら