# Sun™ Mainframe Transaction Processing ソフトウェア 開発者ガイド

Release 8.1.0

Sun Microsystems, Inc.
www.sun.com

Part No. 819-2516-10
2005 年 6 月, Revision A

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

**原典**: *Sun™ Mainframe Transaction Processing Software Developer's Guide*
Part No: 817-7432-10
Revision A

# 目次

# 図目次

# 表目次

# コード例

# はじめに

このマニュアルはアプリケーション設計者および開発者を対象としており、IBM CICS の操作および管理の経験、さらに Sun™ Mainframe Transaction Processing ソフトウェア (Sun MTP) の管理、ファイル構造、およびインストールについての実務経験があることを前提としています。

## お読みになる前に

このマニュアルに記載された情報を十分に活用していただくためには、次のマニュアルに記載された内容を理解しておく必要があります。

- 『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』
- 『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』

## マニュアルの構成

第 1 章では、アプリケーションの作成に使用する開発システムメインメニューおよびツールについて説明します。Sun MTPの標準規格についての情報も記載されています。

第 2 章では、Sun MTP ファイルエディタの使用法について説明します。

第 3 章では、開発システムの UNIX® ファイルユーティリティーの使用法について説明します。

第 4 章では、Sun MTP がサポートする CICS API について、および Sun MTP と CICS の相違点について説明します。

第 5 章では、コマンド行インタフェースと開発システムツールを使用してオンライン COBOL アプリケーションプログラムをコンパイルする方法について説明します。

第 6 章では、Liant Open PL/I で記述されたアプリケーションプログラムに対する Sun MTP のサポートについて説明します。また、開発システムツールを使って、オンライン PL/I プログラムをコンパイルする方法についても説明します。

第 7 章では、C プログラミング言語で記述されたアプリケーションプログラムに対する Sun MTP のサポートについて説明します。

第 8 章では、Java™ プログラミング言語で記述されたアプリケーションプログラムに対する Sun MTP のサポートについて説明します。

第 9 章では、画面生成ユーティリティー (SGU) の画面およびファンクションキーを使って、マップを作成および管理する方法について説明します。また、さまざまなツールを使って、マップをアセンブルする方法についても説明します。

第 10 章では、リレーショナルデータベース管理システム (RDBMS) の SQL インタフェースに対する Sun MTP のサポートについて説明します。

第 11 章では、クライアントアプリケーションがソケットを使って Sun MTP 領域と通信する方法について説明します。

第 12 章では、WebSphere MQ クライアントと連携させるための Sun MTP の設定方法について説明します。また、WebSphere MQ メッセージング/キューイング API にアクセスするためのプログラムに影響を及ぼす問題についても説明します。

第 13 章では、MQ-JMS Bridge の使用法について説明します。

第 14 章では、Sun MTP のデバッグ機能、COBOL ソースデバッガ、Liant CodeWatch for PL/I、および C シンボリックデバッガを使って、オンラインアプリケーションをデバッグする方法について説明します。

第 15 章では、Sun MTP を使用したバッチプログラムの起動、コンパイル、設定、および実行方法について説明します。バッチプログラムのデバッグに関する情報も記載されています。

第 16 章では、Sun MTP リソースで定義されていない C プログラムを使用した VSAM ファイルにアクセスする方法について説明します。サポートされている C-ISAM 機能についても説明します。

# 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。<br>ls -a を実行します。<br>% 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**<br>Password: |
| *AaBbCc123* | コマンド行の可変部分。実際の名前や値と置き換えてください。 | rm *filename* と入力します。 |
| 『 』 | 参照する書名を示します。 | 『Solaris ユーザーマニュアル』 |
| 「 」 | 参照する章、節、または、強調する語を示します。 | 第 6 章「データの管理」を参照。<br>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 **<br>**XV_VERSION_STRING'** |
| [ ] | 省略可能な項目を示します。 | unikixmain [-Q] |
| { } | セパレータ ( \| ) で区切られた代替オプション | kixfile [-r{Y\|N}] |
| \| | 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。 | EXEC CICS READ DATASET \|<br>FILE |

* 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。

# 関連マニュアル

| 製品 | タイトル | Part No. |
|---|---|---|
| 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 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 Batch Manager ソフトウェア | 『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』 | 819-2505-10 |
| | 『Sun Mainframe Batch Manager ソフトウェア インストールガイド』 | 819-2506-10 |
| | 『Sun Mainframe Batch Manager ソフトウェア メッセージガイド』 | 819-2507-10 |
| | 『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』 | 819-2508-10 |
| | 『Sun Mainframe Batch Manager ソフトウェア リファレンスマニュアル』 | 819-2360-10 |
| | 『Sun Mainframe Batch Manager ソフトウェア 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 Cluster 用の High Availability エージェント | 『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 |
| | 『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 Security Facility 高可用性 (HA) データサービス (Sun Cluster 用)』 | 819-2512-10 |
| Sun Mainframe Security Facility | 『Sun Mainframe Security Facility 管理者ガイド』 | 819-2359-10 |
| | 『Sun Mainframe Security Facility ご使用にあたって (Solaris プラットフォーム用)』 | 819-2513-10 |

| 製品 | タイトル | Part No. |
|---|---|---|
| IBM CICS | 『CICS アプリケーション・プログラミング・リファレンス』 | SC33-1170 |
| | 『CICS アプリケーション・プログラミング・ガイド』 | SC33-0674 |
| | 『CICS Master Index』 | SC33-1074 |
| | 『CICS Supplied Transactions』 | SC33-1686 |
| | 『CICS System Programming Reference』 | SC33-1689 |
| Server Express | Server Express のマニュアル | * |
| ACUCOBOL-GT | ACUCOBOL-GT のマニュアル | * |
| Liant 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 | * |
| C-ISAM | 『C-ISAM Programmer's Manual』 | * |
| | 『System Performance Tuning』、Mike Loukides 著、砂原秀樹監訳、株式会社アスキー発行、1995 | |

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

# Sun のマニュアルの注文方法

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

http://www.sun.com/documentation

# Sun の技術サポート

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

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

# コメントをお寄せください

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

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

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

『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』、Part No. 819-2516-10

# 第1章

# 開発システム

Sun MTP 開発システムでは、1 つのメニューからテーブルマネージャー、ファイルマネージャー、レコードエディタ、および画面作成ユーティリティー (SGU) を実行できます。アプリケーションプログラムのコンパイル、BMS マクロのアセンブル、ソースファイルの編集、およびファイルコマンドとユーティリティーの実行を行うためのツールも用意されています。

この章の内容は、次のとおりです。



## 開発システムの開始

実行中の Sun MTP 領域にローカルで接続していて、端末にブランクのトランザクション画面が表示されている場合、`CMNU` トランザクションを入力して開発システムを開始します。

3270 端末または TN3270 エミュレータ経由で遠隔接続している場合は、`CMNU` を使用できません。この場合に使用できるのは、開発システムの一部に直接アクセスするトランザクションのいずれか 1 つだけです。たとえば、`CTBL` を使ってテーブルマネージャーを開きます。

# Development System メインメニュー

開発システムメインメニューは、利用可能なツールの中からいずれか 1 つを選択するためのインタフェースです。図 1-1 に、開発システムメインメニューを示します。

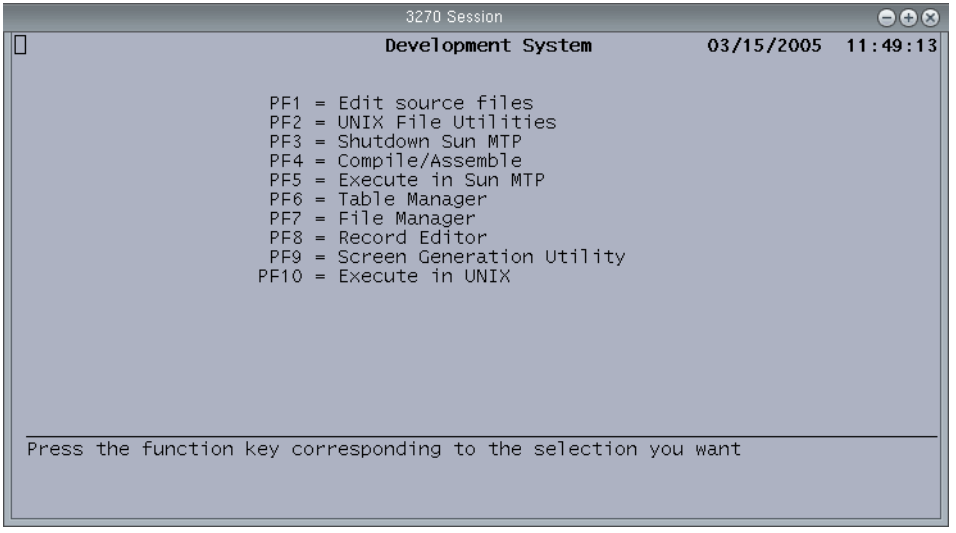


**図 1-1** Development System メインメニュー

使用するツールを開くには、ファンクションキーを押します。

**表 1-1** 開発システムのファンクションキー

| ファンクションキー | 動作 |
|---|---|
| PF1 | ファイルエディタを起動します。詳細は、第 2 章を参照してください。 |
| PF2 | 参照、コピー、連結、削除、印刷、名前の変更、検索の UNIX ファイル機能を実行します。詳細は、第 3 章を参照してください。 |
| PF3 | Sun MTP クライアントおよび Sun MTP サーバーを停止します。確認メッセージが表示されたときに、このファンクションキーをもう一度押すと、その領域が停止します。 |
| PF4 | COBOL プログラム、Liant Open PL/I プログラム、BMS マップをコンパイルおよびアセンブルします。 |
| PF5 | メニューシステムを終了し、端末を使用してトランザクション識別子を入力します。 |

**表 1-1**　開発システムのファンクションキー (続き)

| ファンクションキー | 動作 |
|---|---|
| PF6 | テーブルマネージャーを開きます。テーブルには、アプリケーションのリソースの定義が保存されています。ブランクのトランザクション画面で CTBL トランザクション識別子を入力すると、開発システムメインメニューを省略してテーブルマネージャーが開始されます。テーブルマネージャーについては、『Sun Mainframe Transaction Processing 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。 |
| PF7 | VSAM カタログを表示するファイルマネージャーを開きます。このカタログでは、アプリケーションの VSAM ファイルを定義します。ブランクのトランザクション画面で CFMS トランザクション識別子を入力すると、開発システムメインメニューを省略してファイルマネージャーを開始できます。ファイルマネージャーの使用手順については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |
| PF8 | レコードエディタを起動します。ブランクのトランザクション画面で CRED トランザクション識別子を入力すると、開発システムメインメニューを省略してレコードエディタが起動します。詳細は、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |
| PF9 | 画面作成ユーティリティー (SGU) を開始します。ブランクのトランザクション画面で CSGU トランザクション識別子を入力すると、開発システムメインメニューを省略して SGU が開始します。詳細は、第 9 章を参照してください。 |
| PF10 | コマンドモードに入ります。このコマンドモードでは、プロンプトから unikix 以外のすべての UNIX コマンドを入力できます。unikix を入力すると、Sun MTP クライアントが端末ですでに有効になっていることを示すエラーメッセージが表示されます。開発システムに戻るには、Ctrl+D を押します。コマンドモードでの操作中に現在のディレクトリの変更または新たな環境変数の設定を行った場合に Ctrl+D を押すと、これらの変更や設定が無効になります。したがって、ほかの Sun MTP ユーティリティーの環境変数を設定するときには、このオプションは使用しません。環境変数を設定する必要がある場合は、領域を停止してから環境変数を設定し、領域を再起動します。 |

# 開発システムの制限

Sun MTP の開発機能は、オペレーティングシステムで提供されているプロセスおよび文字ベースのサービスに大きく依存します。これらのサービスは通常、IBM SNA ネットワークで使用される端末デバイスおよびアクセス方法と互換性がないので、3270 デバイスまたは経路指定された端末から Sun MTP 領域で作業するときに一部の開発機能が制限を受けます。

IBM 3270 デバイスと経路指定された端末は、開発者の端末としては使用できません。これらを使用して、開発されるプログラムをテストし、実際のネットワークと端末でプログラムを検証できます。

UNIX システムに直接接続している 3270 デバイスでは、`CMNU` および `CSGU` トランザクションを実行できません。さらに、経路指定された端末では、`CEMT`、`CTBL`、`CFMS`、`CMNU`、`CRED`、および `CSGU` トランザクションを実行できません。システムトランザクションについては、『Sun Mainframe Transaction Processing ソフトウェア管理者ガイド』を参照してください。

# Sun MTP 標準規格

この節では、Sun MTP の標準規格に関する次の事項について説明します。

- 5 ページの「画面の形式」
- 6 ページの「ファンクションキー」
- 6 ページの「データの検査」
- 7 ページの「ファイル識別子」

ご使用のユーザーアプリケーションによって、標準規格が異なる場合があります。

# 画面の形式

すべてのデータエントリ画面は、4 つの領域に分かれています。

**表 1-2**　画面の形式

| 画面の領域 | 説明 |
|---|---|
| ヘッダー領域 (1 行目) | 現在の画面名および現在の日付/時刻が示されます。 |
| 詳細領域<br>説明領域 (2 ～ 20 行目) | メニュー、データエントリ画面、およびファイル選択領域が含まれます。 |
| 応答領域 (21 行目) | 通常は、行全体に下線 (_) が引かれています。データをディスクに書き込む処理などの実行でエラーが発生すると、エラー条件を示すメッセージが強調表示されます。実行中の処理の状態が表示される場合もあります。 |
| ファンクションキー<br>説明領域 (22 ～ 24 行目) | 画面で使用可能なファンクションキーおよびそれを押したときに実行できる操作が表示されます。 |

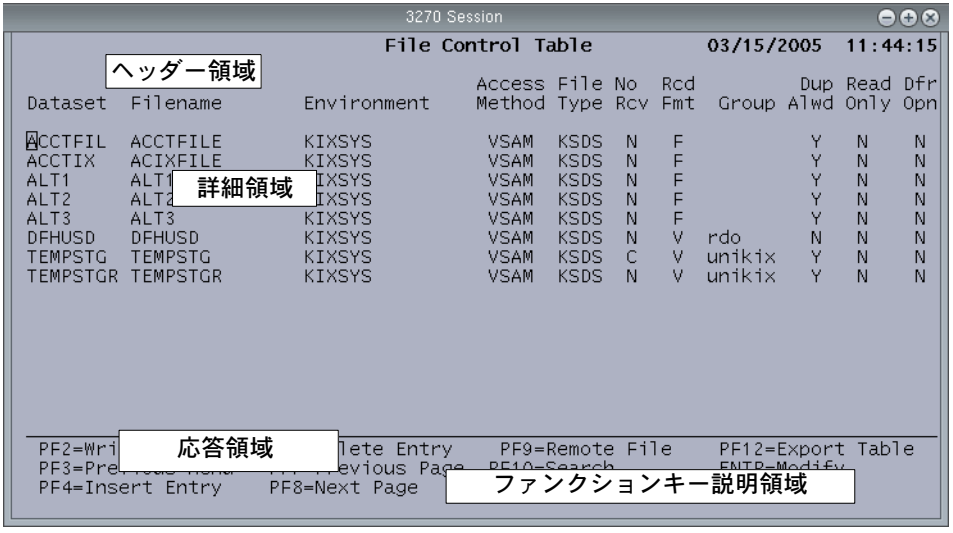


**図 1-2**　Sun MTP 画面形式の例

# ファンクションキー

次のファンクションキーは、Sun MTP 画面で特定の操作を実行します。

表 1-3　標準ファンクションキー

| ファンクションキー | 動作 |
|---|---|
| PF3 | 前の画面に戻ります。画面で変更したデータをディスクに保存する前にこのキーを押すと、次の警告が表示されます。<br>`Table has been modified. Press PF3 if modification is only temporary.`<br>変更を保存しない場合は PF3 を押し、別の操作を実行する場合は他のキーを押します。 |
| Enter | 現在表示されているエントリのデータを変更します。保護されていない値を上書きしてこのキーを押すと、画面の内容が変更されます。すべてのフィールドが関連の検証テストに通った場合だけデータが変更されます。 |
| Clear | 新しいトランザクションを入力できるように画面を消去します。 |
| Reset | 数値フィールドへのテキストの入力など、操作エラーのあとでシステムをリセットします。 |

キーボードの割り当てについては、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

# データの検査

データフィールドには大文字および小文字の両方を入力できます。ほとんどの場合、小文字は大文字に変換されます。一部のデータフィールドでは、小文字が有効なデータとして受け入れられます。通常、CICS コマンド関連のフィールドでは大文字に変換され、その他のフィールドでは入力されたままの状態になります。たとえば、データセットは大文字に変換され、ファイル名は入力されたままの状態になります。

データ入力の検査により、無効なデータが含まれるフィールドは強調表示されます。エラーデータが入力されている最初のフィールドにカーソルが移動し、応答領域に次のようなメッセージが表示されます。

```
Data in field invalid/required
```

# ファイル識別子

ファイル識別子は、次の 2 つの要素で構成されます。

- 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部分を構成するファイル名

ファイル識別子を入力する場合、次の表に記載されている指定事項に従ってください。

**表 1-4**　　ファイル識別子

| ファイル識別子 | 説明 |
|---|---|
| ディレクトリ | Sun MTP で使用される絶対ディレクトリ名は、50 文字以内でなければいけません。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、同一のディレクトリ名を示します。<br>• `mtp/mtp8/finance/sys`<br>• `$KIXSYS`<br>`$` 符号は、`KIXSYS` 環境変数を完全な値に展開します。 |
| 環境変数 | ディレクトリやファイルの名前、または値 (1 ～ 14 文字) です。<br>環境変数はすべて大文字です。 |
| ファイル名 | 拡張子も含む Sun MTP ファイル名 (1 ～ 14 文字) です。 |

第2章

# ファイルエディタ

Sun MTP ファイルエディタでは、編集するファイルを選択したり、編集に使用するデフォルトエディタを開いたりすることができます。デフォルトのエディタは `vi` エディタですが、`kixed` シェルスクリプトをカスタマイズして他のエディタを起動するように設定できます。Sun MTP シェルスクリプトのカスタマイズについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

この章の内容は、次のとおりです。



## ファイルエディタの起動とファイルの選択

### ▼ ファイルエディタを起動し、ファイルを選択する

1. **開発システムメインメニューで PF1 キーを押すと、図 2-1 に示すファイルメニュー画面が表示されます。**

   画面上部のフィールドに表示条件が示されます。これらのフィールドについては、表 2-1 を参照してください。

   画面の下の部分には、選択可能なファイルが指定した条件に基づいて一覧表示されます。このファイル表示領域の各列については、表 2-2 を参照してください。

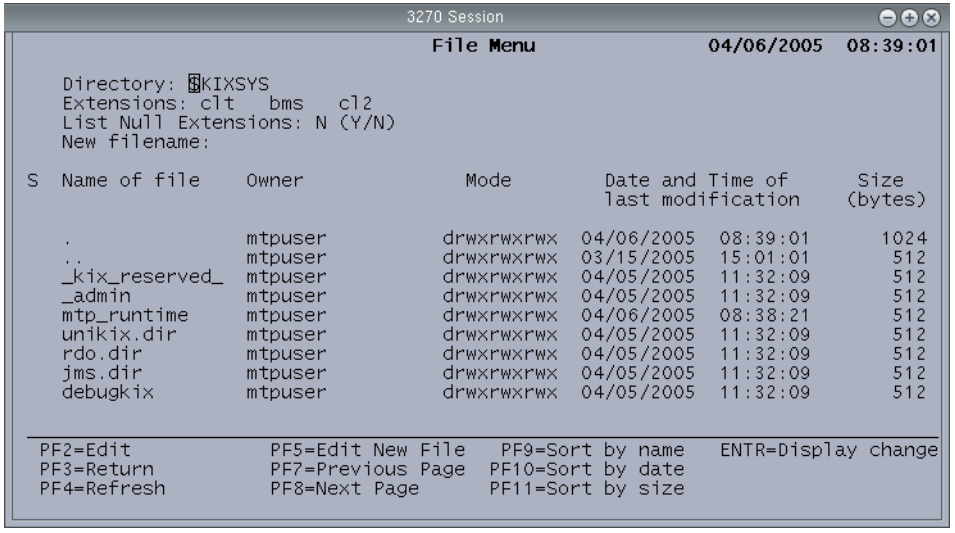


**図 2-1** ファイルエディタ – File Menu 画面

2. **画面上部のファイル条件を指定して PF4 キーを押すと、条件に合致するファイルが表示されます。**

   「New filename」フィールドに名前を入力して PF5 キーを押すと、その名前が付けられた空ファイルがエディタ上で開きます。

3. **Tab キーを押して、表示条件フィールドから画面左側にある S というラベルの付いた選択列にカーソルを移動します。**

4. **ファイル名の横に S または s の文字を入力してファイルを選択します (複数選択可能)。**

   「..」を選択すると、「Directory」フィールドが親ディレクトリに変わり、画面が更新されて親ディレクトリの内容が表示されます。

5. **PF2 キーを押して、選択したファイルをエディタで開きます。**

   複数ファイルを選択すると、すべてのファイルの編集を終えるまでファイルが順次表示されます。

# 表示条件フィールド

次の表に、表示条件フィールド、各フィールドのデフォルト値、および有効値を示します。

**表 2-1**　表示条件フィールド

| フィールド名 | デフォルト値 | 有効値 |
|---|---|---|
| ディレクトリ | `$KIXSYS` | 編集するファイルの位置を指定する絶対パス名です。ディレクトリの最初のパラメータが環境変数の場合、最初の文字をドル記号 ($) にする必要があります。 |
| Extensions | `bms`、`clt`、`cl2`、`cbl`、`pl`、`plt`、`pli`、`pl1` | 異なる拡張子を 8 つまで指定できます。各拡張子は 3 文字で構成され、大文字・小文字が区別されます。拡張子を 1 つ入力すると、次の拡張子の入力位置にカーソルが自動的に移動します。 |
| List Null Extensions | `N` | `Y` と入力すると、ファイルが拡張子なしで表示されます。 |
| New filename | ブランク | ファイル名は、Sun MTP のファイル命名規則に従って指定する必要があります。拡張子を含めることもできます。PF5 キー (新規ファイルの編集) を押すと、新規ファイルが作成されます。 |

# ファイル情報フィールド

次の表に、ファイルメニュー画面のファイル情報フィールドを示します。

**表 2-2** ファイルメニュー画面—ファイル情報フィールド

| フィールド名 | 説明 |
|---|---|
| Name of file | ファイル名の始めの 14 文字です。<br>ファイル表示領域の上の 2 行「.」と「..」は、それぞれ現在のディレクトリ (.) と親ディレクトリ (..) を示します。その後続の行には、そのディレクトリ内のファイル名およびディレクトリ名が一覧表示されます。 |
| Owner | ファイルの所有者です。<br>多くの場合、一覧表示されたファイルの所有者は現在画面を開いているユーザーです。ファイルの所有者が現在のユーザーではない場合、そのファイルは読み取り専用になる場合があります。 |
| Mode | ファイルタイプおよびアクセス権を表す 10 文字のフィールドです。最初の文字はファイルタイプを表します。 もっとも一般的なファイルタイプは次の 2 つです。<br>• `d`: ディレクトリ<br>• `-`: 通常のテキストファイル<br>ファイルタイプの完全なリストについては、オペレーティングシステムのドキュメントを参照してください。<br>残りの 9 文字は、ファイルのアクセス権を表します。アクセス権は、次のように 3 文字ずつの 3 セットに分けられます。<br>• 2 ～ 4 文字目は、ファイルの所有者のアクセス権<br>• 5 ～ 7 文字目は、グループのアクセス権<br>• 8 ～ 10 文字目は、その他のユーザーのアクセス権。表示される値はファイル名のみに適用され、ディレクトリには適用されません<br>次の 3 文字はそれぞれ、各セットで同じ意味を持ちます。<br>• 1 文字目 `r` は読み取りを許可<br>• 2 文字目 `w` は書き込みを許可<br>• 3 文字目 `x` は、ファイルが実行可能ファイルであると認識されていることを示します |
| Date and Time of last modification | 2 つの列があり、最初の列は *MM/DD/YYYY* 形式の日付を示します。2 列目は、*HR:MIN:SEC* 形式の 24 時間表示を示します。 |
| Size (bytes) | ファイルサイズのバイト数です。 |

# ファイルエディタのファンクションキー

ファイルエディタのファイルメニュー画面で、次のキーが使用できます。

**表 2-3**　ファイルエディタのファンクションキー

| ファンクションキー | 動作 |
|---|---|
| PF2 | `kixed` シェルスクリプトで定義されているエディタを起動します。 |
| PF3 | ファイルエディタを閉じて開発システムメインメニューに戻ります。 |
| PF4 | 画面のファイルリスト部分のデータを更新します。編集の終了後に PF4 キーを押すと、ファイル情報に加えた変更が表示に反映されます。 |
| PF5 | 「New filename」フィールドに入力した名前で空のファイルを開きます。ファイル名を指定しないと、エラーメッセージが表示されてファイル名の入力が求められます。 |
| PF7 | ファイルの前のページを表示します。ファイルの最初のページが表示されているときにこのキーを押すと、次のメッセージが表示されます。<br>`Beginning of data reached` |
| PF8 | ファイルの次のページを表示します。ファイルの最後のページが表示されているときにこのキーを押すと、次のメッセージが表示されます。<br>`End of data reached` |
| PF9 | ディレクトリの内容をファイル名順にソートして再表示します。 |
| PF10 | ディレクトリの内容をファイルの最終変更日順にソートして再表示します。 |
| PF11 | ディレクトリの内容をファイルサイズ順にソートして再表示します。 |
| Enter | 表示条件フィールドに対するすべての変更を有効にします。画面が更新され、新しい条件に合致したファイルが表示されます。 |

# 第3章

# UNIX ファイルユーティリティー

UNIX ファイルユーティリティーのメニューを使って、Sun MTP 環境内からファイルを操作できます。ユーティリティーは、デフォルトのエディタとして `vi` を起動します。`vi` エディタの使用法については、UNIX のマニュアルを参照してください。デフォルトのエディタを変更する方法については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

この章の内容は、次のとおりです。

# UNIX File Utilities メニュー

図 3-1 に、メニュー構造を示します。

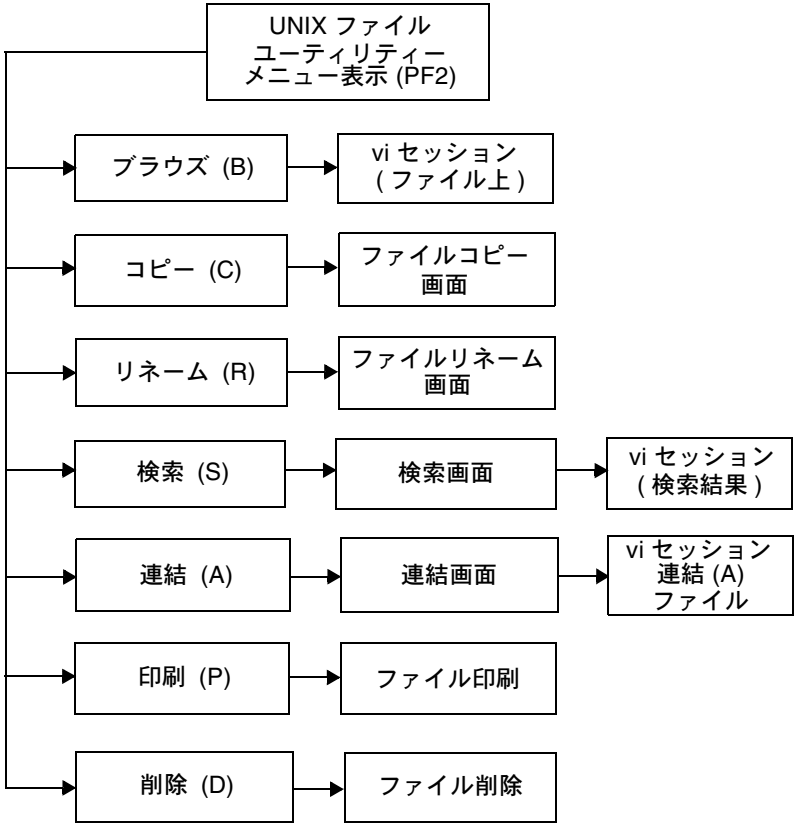


**図 3-1** UNIX ファイルユーティリティーメニューマップ

## ▼ ファイルユーティリティーメニューを表示する

- **開発システムメインメニューで PF4 キーを押します。**

  図 3-2 に示すメニューは、Sun MTP 標準ファイル選択画面の例です。

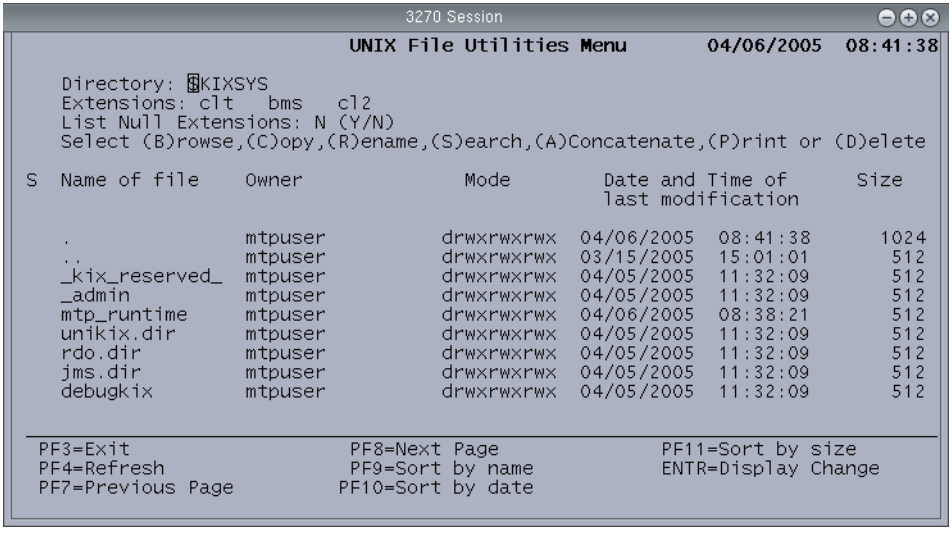


**図 3-2** UNIX File Utilities メニュー

画面上部の表示条件フィールドの 3 つの行は、表 2-1 で説明した最初の 3 行です。「Select」フィールドには、一覧表示されたファイルに適用できるオプションが表示されます。ファイル表示領域には、表 2-2 と同じフィールドがあります。

## ▼ オプションを適用する

- **対象ファイルの横の選択列にオプションの文字 (括弧内の文字) を入力します。**

  入力には大文字を使用します。

  各ファイルに適用できるオプションは 1 つのみです。ただし、同じオプションを複数のファイルに適用できます。また、同じ画面上で異なるオプションを指定できます。Enter を押すと、オプションが順次実行されます。

UNIX ファイルユーティリティーメニューでは、次のファンクションキーが使用できます。

表 3-1　UNIX ファイルユーティリティーのファンクションキー

| ファンクションキー | アクション |
|---|---|
| PF3 | UNIX ファイルユーティリティーメニューを閉じて、開発システムメインメニューに戻ります。 |
| PF4 | 画面のファイルリスト部分のデータを更新します。変更の終了後に PF4 キーを押すと、ファイル情報に加えた変更が表示反映されます。 |
| PF7 | ファイルの前のページを表示します。ファイルの最初のページが表示されているときにこのキーを押すと、次のメッセージが表示されます。`Beginning of data reached.`(**データの先頭です**) |
| PF8 | ファイルの次のページを表示します。ファイルの最後のページが表示されているときにこのキーを押すと、次のメッセージが表示されます。`End of data reached.`(**データの末尾です**) |
| PF9 | ディレクトリの内容をファイル名順にソートして再表示します。 |
| PF10 | ディレクトリの内容をファイルの最終変更日順にソートして再表示します。 |
| PF11 | ディレクトリの内容をファイルサイズ順にソートして再表示します。 |

# ファイルユーティリティーの使用法

この節では、ファイルユーティリティーの使用法について説明します。

## ブラウズオプション (B)

ブラウズオプションを選択すると、`kixbrw` シェルスクリプトを実行し、指定されているエディタで、選択したファイルを読み取り専用ファイルとして開きます。

# コピーオプション (C)

コピーオプションでは、1 ファイルをコピーします。コピーするファイルを選択して Enter キーを押すと、図 3-3 のファイルコピー画面が表示されます。

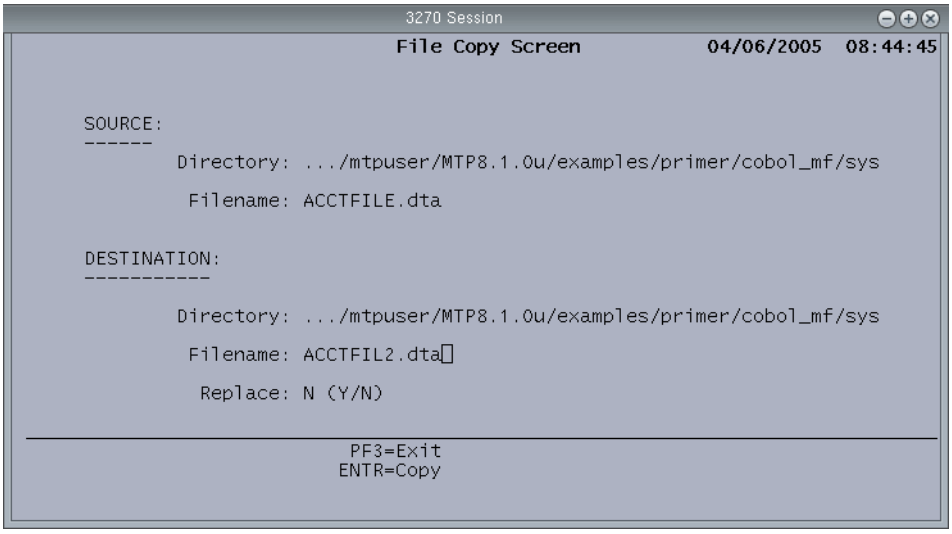


**図 3-3** UNIX File Utilities—File Copy 画面

ファイルコピー画面は、次の 2 つの部分から構成されています。

| フィールド名 | SOURCE (コピー元) | DESTINATION (コピー先) |
|---|---|---|
| Directory | ソースファイルがあるディレクトリ | デフォルトでは source ディレクトリですが、変更可能です。 |
| Filename | ソースファイルの名前 | ファイル名を入力します。入力しない場合、ファイル名の入力を要求されます。 |
| Replace | | 既存のファイルをリネームファイルで置き換えるかどうかを指定します。<br>• N: 既存のファイルを置き換えません。同じ名前のファイルが存在する場合、置換オプションの変更またはファイル名の変更を要求するエラーメッセージが表示されます。<br>• Y: 既存のファイルが自動的に置き換えられます。 |

**注意 –** 置換オプションを `Y` にしてファイルをコピーすると、確認メッセージは表示されません。誤ってファイルを上書きした場合、バックアップから復元する以外には元のファイルを復元できません。

データを入力して Enter キーを押すと、`kixcopy` シェルスクリプトが実行されてファイルがコピーされます。

## リネームオプション (R)

リネームオプションでは、1 つのファイルの名前を変更します。名前を変更するファイルを選択して Enter キーを押すと、次の画面が表示されます。

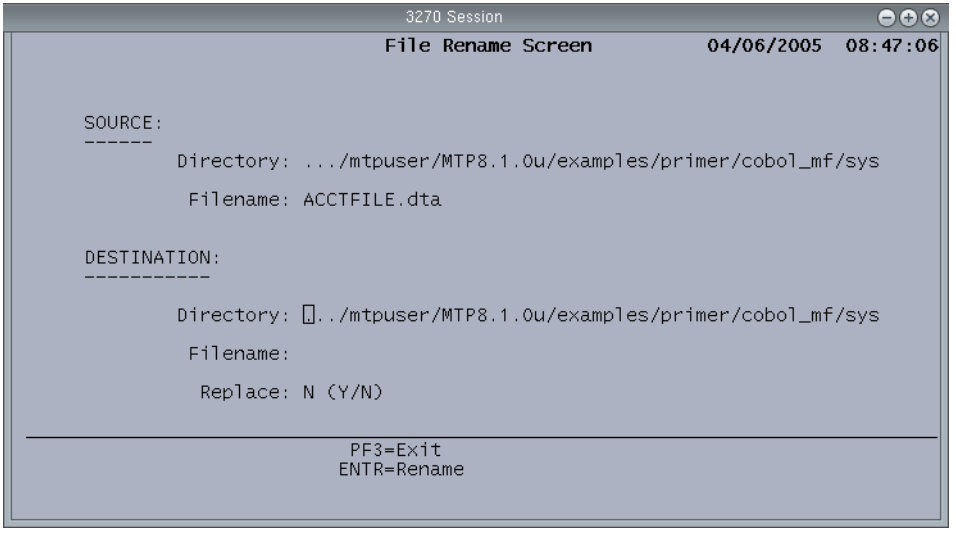


図 3-4　UNIX File Utilities—Rename 画面

ファイルのリネーム画面は、ファイルコピー画面と同様に次の 2 つの部分から構成されています。

| フィールド名 | SOURCE (リネーム元) | DESTINATION (リネーム先) |
|---|---|---|
| Directory | ソースファイルがあるディレクトリ | デフォルトでは source ディレクトリですが、変更可能です。 |
| Filename | ソースファイルの名前 | ファイル名を入力します。入力しない場合、ファイル名の入力を要求されます。 |
| Replace | | 既存のファイルをリネームファイルで置き換えるかどうかを指定します。<br>• N: 既存のファイルを置き換えません。同じ名前のファイルが存在する場合、置換オプションの変更またはファイル名の変更を要求するエラーメッセージが表示されます。<br>• Y: 既存のファイルが自動的に置き換えられます。 |

**注意 –** 置換オプションを `Y` にしてファイルの名前を変更すると、確認メッセージは表示されません。誤ってファイルの名前を変更した場合、バックアップから復元する以外にはファイルを復元できません。

データを入力して Enter キーを押すと、`kixrnm` シェルスクリプトが実行されてファイルの名前が変更されます。

# 検索オプション (S)

検索するファイルを 1 つ以上選択して Enter キーを押すと、検索画面が表示されます。このオプションでは、選択したファイル内で検索パターンの有無を検索します。

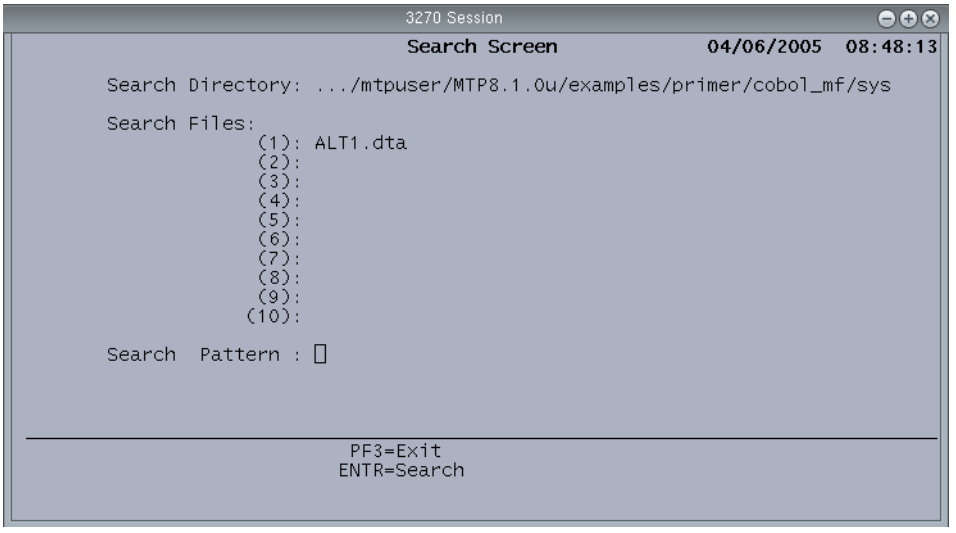


**図 3-5** UNIX File Utilities—Concatenate 画面

検索画面でのオプションは検索パターンのみですが、この指定は必須です。指定しないと、検索パターンの入力を要求するエラーメッセージが表示されます。

検索オプションでは `kixgrep` シェルスクリプトが実行され、選択したファイルで検索パターンを検索します。検索結果は一時ファイルに書き込まれるので、デフォルトエディタの画面に表示できます。

---

**注 –** 検索パターンは大文字・小文字を区別するので、ファイルでの記述どおりに検索パターンを入力する必要があります。

---

`kixgrep` シェルスクリプトを変更することにより、検索結果の表示に使用するエディタおよび検索ユーティリティーを変更できます。シェルスクリプトのカスタマイズについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

# 連結オプション (A)

連結オプションでは、複数のファイルを 1 つのファイルに結合します。連結させる複数のファイルを選択して Enter キーを押すと、図 3-6のような連結画面が表示されます。

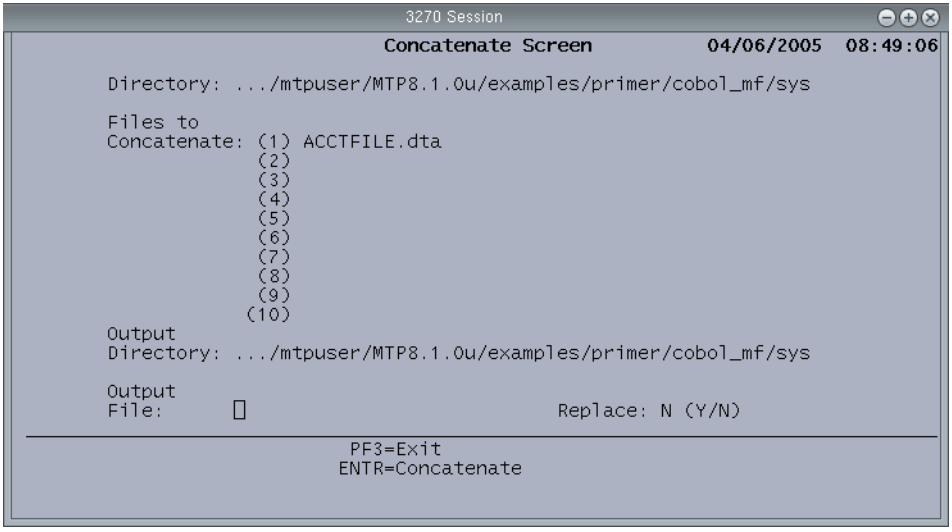


**図 3-6** UNIX File Utilities—Concatenate 画面

次の表に、連結画面のフィールドを示します。

| フィールド名 | 説明 |
|---|---|
| Directory | ソースファイルがあるディレクトリ |
| Files to Concatenate | 選択したファイルが一覧表示されます。連結対象ファイルは最大 10 ファイルまで選択可能です。ただし、1 回の連結で選択できるファイルは、UNIX ファイルユーティリティーメニューの同じページに一覧表示されるファイルのみです。 |
| Output Directory | 連結後のファイルの保存先ディレクトリは、デフォルトでは source ディレクトリですが変更可能です。 |
| Output File | 連結後のファイルの名前を指定します。 |
| Replace | 既存のファイルを「Output File」フィールドで名前を付けたファイルで置き換えるかどうかを指定します。<br>• N: 既存のファイルを置き換えません。同じ名前のファイルが存在する場合、置換オプションの変更またはファイル名の変更を要求するエラーメッセージが表示されます。<br>• Y: 既存のファイルが自動的に置き換えられます。 |

**注意 –** 置換オプションを `Y` にしてファイルを連結すると、確認メッセージは表示されません。誤ってファイルを連結した場合、バックアップから復元する以外にはファイルを復元できません。

Enter キーを押すと、`kixcat` シェルスクリプトが実行されてファイルが連結されます。デフォルトでは `kixcat` シェルスクリプトで指定されているエディタに連結結果を表示できます。

連結処理中にエラーが発生すると、エラーメッセージが表示されて連結画面が再表示されます。

## 印刷オプション (P)

印刷オプションでは、`kixprt` シェルスクリプトを実行して印刷を行います。Enter キーを押すと、ただちにファイルが印刷されます。そのあとの操作は不要です。シェルスクリプトを変更すると、特定のプリンタまたはプリンタオプションを選択できます。シェルスクリプトのカスタマイズについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

## 削除オプション (D)

削除オプションでは、`kixdlt` シェルスクリプトが実行されます。UNIX ファイルユーティリティーメニューで Enter キーを押すと、ファイルが削除されます。そのあとの操作は不要です。

**注意 –** 削除オプションでは、確認メッセージが表示されません。削除するファイルを選択して Enter キーを押すと、ファイルはただちに削除されます。バックアップから復元する以外にはファイルを復元できません。

第4章

# IBM CICS との互換性

この章では、次のトピックについて説明します。



## Sun MTP と IBM CICS の相違点

基本的なアーキテクチャーの違いにより、Sun MTP が IBM CICS と異なる点がいくつかあります。Sun MTP では、各端末のトランザクションは別のトランザクションサーバーで実行されますが、IBM CICS では、すべてのトランザクションがひとつの大きなパーティションで実行されます。

### 絶対アドレス

各ユーザーは異なる仮想アドレススペースで実行しているので、異なる端末で実行しているトランザクション間で絶対アドレスを渡すときは、注意してください。通常、これは行わないでください。

ただし、共有メモリーの領域を参照するアドレスは、トランザクションサーバー間で同じであることが保証されています。Sun MTP は、共通システム域 (CSA) および 共通作業域 (CWA) を共有メモリーに実装します。また、`POST` および `WAIT EVENT` コマンドで使われるイベント制御ブロックも実装します。これらのアドレスは、自由に渡すことができます。

EXEC CICS GETMAIN コマンドを使って取得した領域は、トランザクションサーバーに割り当てられます。この領域は他のトランザクションに渡すことはできません。ただし、同じトランザクション内のプログラム間では自由に渡すことができます。ほかのトランザクションがこれらの領域内のデータを参照できるように割り当てるには、EXEC CICS GETMAIN コマンドの SHARED オプションまたは EXEC CICS LOAD コマンドを使ってロードされるテーブルのいずれかを使用します。

## 端末識別子の割り当て

特定のユーザーセッションに関連する端末識別子は、ユーザーがクライアントを開始するときに割り当てられます。実際の端末識別子は、クライアントが領域に接続する順序に基づいて動的に割り当てられます。

端末には識別子 *Annn* が与えられます。*A* は任意のアルファベットの文字で、*nnn* は順番に割り当てられる 3 桁の 10 進数です。

領域の開始端末識別子は、システム初期化テーブル (SIT) で定義されます。デフォルトは C000 です。ただし、この値はメインフレーム端末識別子との重複を避けるために変更されることがあります。たとえば、領域にログインする最初の端末には識別子 C000 が割り当てられ、2 番目には C001 が割り当てられます。特定の端末識別子が必要な場合は、クライアントを開始するときに指定します。

3270 デバイスおよびプリンタに対しては、端末管理テーブル (TCT) で端末識別子を割り当てる必要があります。オプションで、TN3270E セッションに対して端末識別子を割り当てることもできます。端末識別子の割り当てについては、『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』の TCT に関する節を参照してください。

## バイト順序

コンピュータによっては、2 バイト整数および 4 バイト整数のバイト順序が IBM 370 アーキテクチャーのコンピュータのバイト順序と異なります。多くの場合、コンパイラが、これらの違いをプログラムから見えないようにするために、2 バイト整数および 4 バイト整数をプログラムが要求する順序にします。

ただし、COBOL プログラムに関連する次の 2 つのケースでは、プログラムの変更が必要な場合があります。

- プログラムがベースロケーターリンケージ (BLL) セルまたはポインタ変数を使用し、そのプログラムがこれらのアドレスで計算を実行する場合
- COBOL プログラムから C サブルーチンを呼び出す場合

## Server Express のベースロケータリンケージ (BLL) およびポインタ変数

**注 –** ACUCOBOL-GT® は、COBOL-85 のみをサポートします。

SET オプションが指定された CICS コマンドを使用する場合、BLL セルが COBOL 74 標準に準拠するプログラムで使用されます。SET オプションによってアドレスが返されます。

ポインタ変数は、COBOL 標準に準拠したプログラムで使用されます。このアドレスは、システムが使用するアドレスの形式である必要があります。Sun MTP はこのアドレスを Server Express に適した形式で返します。ただし、この形式は使用する COBOL プログラムのタイプおよび COBOL のバージョンによって異なります。次の表に、その違いを示します。

**表 4-1** バイト順序

| Micro Focus COBOL のバージョン | プログラムタイプ | セットタイプ | 形式 |
|---|---|---|---|
| 3.0 以前 | COBOL 74 (.clt) | BLL セル | 9(9) COMP-5 |
| | COBOL (.cl2) | ポインタ | 9(9) COMP-5 |
| 3.0 またはそれ以降 | COBOL 74 (.clt) | BLL セル | 9(9) COMP |
| | COBOL (.cl2) | ポインタ | 9(9) COMP-5 |

以前のバージョンの COBOL 74 プログラム から 3.0 以降に変換する際に、BLL セル形式が COMP-5 形式の場合には、picture 節を変更する必要があります。Server Express での整数再定義のポインタは、再定義に PIC 9(9) COMP-5 形式を使う必要があります。これらのアドレスが 9 桁よりも大きい場合があるので、notrunc コンパイラオプションを使用する必要があります。最後に、Sun MTP はプログラムのタイプ (.clt または .cl2) に基づいて、どの SET 形式を使うかを決定します。

## COBOL プログラムからの C サブルーチンの呼び出し

C サブルーチンは常に、システムアーキテクチャーで定義された順序で 2 バイト整数および 4 バイト整数を使用します。整数が COBOL から C に渡される場合、COBOL で COMP-5 として定義する必要があります。これを行わないと、C サブルーチンが正しく実行されない場合があります。

# サポートされている ISC 機能

Sun MTP では、次の CICS ISC 機能のサブセットを提供します。

- トランザクション経路指定
- 機能シップ
- 非同期処理
- 分散プログラムリンク (DPL)
- 分散トランザクション処理 (DTP)

これらは、Sun MTP 領域と CICS 領域間の双方向で機能します。CICS 以外のシステムとの相互運用性は検証されていません。

ISC 機能を使用すると、Sun MTP は `EXEC CICS` コマンドの `SYSID` パラメータの値を確認し、それを TCT の既定のシステムエントリと対照して検査します。

トランザクション経路指定では、あるシステムに接続されている端末が他のシステムでトランザクションを実行できます。

機能シップは、VSAM ファイル制御コマンド、一時データキューコマンド、および一時記憶域キューコマンドに提供されています。次の表に、機能シップに使用できるコマンドを示します。

**表 4-2**　機能シップコマンド

| コマンド機能 | コマンド |
|---|---|
| VSAM ファイル制御 | `READ`、`WRITE`、`DELETE`、`REWRITE`、`UNLOCK`、`STARTBR`、`RESETBR`、`ENDBR`、`READNEXT`、`READPREV` |
| 一時データキュー制御 | `READQ TD`、`WRITEQ TD`、`DELETEQ TD` |
| 一時記憶域キュー制御 | `READQ TS`、`WRITEQ TS`、`DELETEQ TS` |

Sun MTP 機能シップは、確認レベルの論理ユニット 6.2 (LU6.2) セッションで行われる会話です。このセッションでは、機能シップの要求を受け取る CICS システム上で、ミラートランザクション `CVMI` または `CPMI` が定義されている必要があります。不正終了または同期点に対する暗黙的あるいは明示的な要求により、ミラーとの確認レベルの同期点プロトコル交換が実行されます。

非同期処理は、遠隔の `START` および `CANCEL` コマンドに関連します。非同期処理では、トランザクションによって遠隔システムでのトランザクションを開始でき、その遠隔システムにデータを渡すことができます。遠隔要求の処理中にローカルリソースへのアクセスに対するブロックが必要でない場合に、または望ましくない状況にあるすべてのアプリケーションに対して、非同期処理を適用します。

DPL では、あるシステムのプログラムが他のシステムのプログラムにリンクできます。DPL によってユーザーは、DB2 や DL/I データベースなどのリソースにアクセスするメインフレーム上で実行されている既存のプログラムにリンクできます。

DTP では、複数システム間での分散処理をサポートします。これは、遠隔リソースの遠隔処理が必要なアプリケーション、システム間でのオンラインデータまたはバッチデータの送信が必要なアプリケーションに有効です。Sun MTP は、31 ページの「サポートされている CICS コマンド」に示す制限のもとで、DTP に対する `EXEC CICS` コマンドレベルのインタフェースをサポートします。

Sun MTP は、ISC の次の領域をサポートしません。

- 論理デバイス制御
- TOR (端末所有領域) に対する Sun MTP の BMS ページング
- トランザクション経路指定および端末制御での経路指定リスト
- 端末制御でのバッチデータ交換
- 同期レベル 2 のミラーおよびプロトコル

3 つの LU6.2 会話セキュリティーモードのうちの 2 つ (Local および Identify) が装備されています。Verify はサポートされていません。

個々の開発者には直接関係ありませんが、ホストに送られる表示使用データはコードセット変換が必要です。このタスクのためのツールも提供されています。この点では、Sun MTP は、ホストがインバウンドデータ変換を行う IBM ASCII 製品に対して異なるアプローチをとります。Sun MTP は CICS に対して EBCDIC ホストのように動作します。

ISC 機能の使用法についての詳細は、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

# 索引の初期化

この節では、作業記憶域の内容を COBOL プログラムで初期化する方法、さらに PL/I および C プログラムとの互換性のために作業記憶域の概念がどのように扱われるかについて説明します。

## COBOL プログラム

Sun MTP が COBOL アプリケーションプログラムを実行する前に、作業記憶域の内容が初期化されます。COBOL プログラムを最初に実行する以前に保存された作業用記憶域のコピーを使って、Sun MTP は `DFH-START` (`kixclt` が取り込む作業記憶域の最初の項目) から `DFHEIV` (リンケージセクションの直前の項目) までのすべての内容を保存します。ただし、COBOL 索引はこれらの境界に格納されていないので、Sun MTP は各 COBOL プログラムの実行に対して COBOL 索引を初期化できません。そのため、これらの索引値はアプリケーションによって初期化される必要があります。

## PL/I および C プログラム

PL/I および C には作業記憶域という概念が存在しませんが、Sun MTP との互換性を提供するために、インクルードファイルには 2 つの文字が定義されています。インクルードファイルは API の一部です。これらの文字は、作業記憶域の最初と最後に対応しています。

# CICS コマンドのサポートレベル

次の表に、CICS コマンドの 3 つのサポートレベルを示します。

| レベル | サポートの説明 |
|---|---|
| サポートする | コマンドが正しい構文であるかどうかをチェックします。構文が正しければ、コマンドを実行します。 |
| 認識しますがサポートしない | コマンドが正しい構文であるかどうかをチェックします。構文が正しければコマンドを実行しますが、コマンドは NULL コマンドとして扱われます。すなわち、Sun MTP はリターンコード 0 とともに呼び出し元プログラムに戻ります。 |
| 認識しない | コマンドはエラーとして通知されます。 |

CICS コマンドをアプリケーションプログラムに取り込む方法については、CICS のマニュアルを参照してください。

Sun MTP は、コマンド処理に関して次のオプションをサポートします。

- NOHANDLE
- RESP
- RESP2
- DFHRESP

NOHANDLE、RESP、および RESP2 オプションは、エラー状態を返すすべてのコマンドで使用できます。DFHRESP オプションは、特定のエラーを確認するために IF 文で使用できます。指定しないかぎり、コマンドは RESP2 に対してゼロを返します。

# サポートされている CICS コマンド

この節では、Sun MTP がサポートするコマンドレベルの CICS 文 (API) を説明します。

## ABEND

```
ABEND [ABCODE (char-4)]
[CANCEL]
[NODUMP]
```

ABEND は、タスクを異常終了します。そのタスクに関連する主記憶域が解放されます。これは完全にサポートされています。

| オプション | 説明 |
|---|---|
| ABCODE | 終了を識別します。 |
| CANCEL | HANDLE ABEND コマンドで設定された出口を無視します。 |
| NODUMP | ダンプを行わないで不正終了するように指定します。 |

## ADDRESS

```
[ORGANIZATION(pointer-ref)]
[CSA(pointer-ref)]
[CWA(pointer-ref)]
[EIB(pointer-ref)]
[TCTUA(pointer-ref)]
[TWA(pointer-ref)]
```

ADDRESS を使用すると、オプションで指定した記憶領域にアクセスできます。

| オプション | 説明 |
|---|---|
| COMMAREA | 実行中のプログラムに対して現在定義されている通信領域のアドレスを返します。COMMAREA は、アプリケーションプログラム間で情報を渡すために使用します。ポインタ参照は、現在の COMMAREA のアドレスになります。COMMAREA がない場合、ポインタ参照は NULL 値 X'FF000000' に設定されます。 |
| CSA | 1024 バイトの共通記憶領域 (CSA) のアドレスを返します。ただし、この領域のすべてのフィールドは NULL に設定されます。Sun MTP の内部構造は IBM CICS と大きく異なるので、内部構造へのアクセスはできません。 |
| CWA | システム初期設定テーブル (SIT) の共通作業領域 (CWA) の長さがゼロ以外の場合、CWA のアドレスが返されます。長さがゼロの場合、値 X'FF000000' が返されます。SIT については、『Sun Mainframe Transaction Processing 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。 |
| EIB | EXEC インタフェースブロック (EIB) のアドレスを返します。 |
| TCTUA | TCT の端末管理テーブルユーザー域 (TCTUA) の長さがゼロ以外の場合、TCTUA のアドレスが返されます。長さがゼロの場合、値 X'FF000000' が返されます。TCT については、『Sun Mainframe Transaction Processing 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。 |
| TWA | プログラム管理テーブル (PCT) のトランザクション作業領域 (TWA) の長さがゼロ以外の場合、TWA のアドレスが返されます。長さがゼロの場合、値 X'FF000000' が返されます。PCT については、『Sun Mainframe Transaction Processing 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。 |

次のオプションはサポートされていません。

ACEE

## ALLOCATE (APPC マップ)

```
ALLOCATE SYSID(char-4)
[PROFILE(char-4)]
[NOQUEUE | NOSUSPEND]
[STATE(cvda)]
```

ALLOCATE コマンドは、完全にサポートされています。DTP プログラムで使用するための LU6.2 セッションを割り当てます。

| オプション | 説明 |
|---|---|
| NOQUEUE | 指定した SYSID からセッションがただちに利用可能にならない場合、ALLOCATE に続く命令に制御を返します。 |
| NOSUSPEND | NOQUEUE の代替キーワードです。 |
| PROFILE | 指定されている場合、LU6.2 構成ファイルの MODE NAME に一致する必要があります。 |
| STATE | 現在の会話の状態を取得します。返される cvda 値は ALLOCATEDです。 |
| SYSID | TCT–システムエントリテーブルのエントリに一致する必要があります。 |

## ASKTIME

```
ASKTIME [ABSTIME(doubleword)]
```

ASKTIME は、EXEC インタフェースブロックのフィールド EIBDATE および EIBTIME を現在の日付および時刻で更新します。ABSTIME は、時間のダブルワード値を返します。この値は、1900 年 1 月 1 日の 00 時 00 分から一番近いミリ秒単位に切り上げられる時間です。

| オプション | 説明 |
|---|---|
| ABSTIME | COBOL:<br>COBOL で S9(15) COMP-3 として定義します。<br>Sun MTP の以前のバージョンでは、MOVE、COMPUTE、DISPLAY などの COBOL 文でフィールドを使用するには、フィールドを S9(15) COMP に変換する必要がありました。フィールドを S9(15) COMP に変換した場合、kixstart または unikixmain の -n オプションを使用する必要があります。<br>ASKTIME または FORMATTIME への呼び出しに、どちらの形式も使用できます。<br>PL/I:<br>PL/I で FIXED DEC(15) として定義します。 |

# ASSIGN

```
ASSIGN option(data-area)
[option(data-area)] ...
```

ASSIGN では、さまざまなシステム変数の値を取得できます。コマンドは部分的にサポートされています。以下のオプション一覧で、各オプションで返される値について説明します。サポートされていないオプションを指定した場合、関連するデータ領域には値が設定されません。

| オプション | 説明 |
|---|---|
| ABCODE (char-4) | 不正終了コードの値。不正終了が発生しない場合は、空白を返します。 |
| ABDUMP (char-1) | 常に値 X'00' を返します。 |
| APLKYBD (char-1) | 常に値 X'00' を返します。 |
| APLTEXT (char-1) | 常に値 X'00' を返します。 |
| APPLID (char-8) | SIT で指定されているアプリケーション名。詳細は、『Sun Mainframe Transaction Processing 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。 |
| BTRANS (char-1) | 背景透過性機能をもたないことを示す値 X'00' を常に返します。 |
| COLOR (char-1) | クライアントによっては、拡張色機能がないことを示す値 X'00' を返すか、あるいは拡張色のサポートを示す値 X'FF' を返します。 |
| CWALENG (halfword) | SIT の CWA の長さ |
| DELIMITER (char-1) | 常に値 X'00' を返します。 |
| DESTCOUNT (halfword) | 常に値 X'0001' を返します。 |
| DESTID (char-8) | 常にスペースを返します。 |
| DESTIDLENG (halfword) | 常にゼロを返します。 |
| DSSCS (char-1) | 常に値 X'00' を返します。 |
| DS3270 (char-1) | 常に値 X'00' を返します。 |
| EWASUPP (char-1) | 常に値 X'00' を返します。 |
| EXTDS (char-1) | 拡張データストリーム機能がないことを示す値 X'00' を常に返します。 |
| FACILITY (char-4) | このトランザクションを開始した端末、トランザクション、DTP 会話 ID、またはキューを含みます。 |
| FCI (char-1) | 端末がトランザクションに関連していることを示す値 X'01' を返すか、このトランザクションに関連する端末がないことを示す X'00' を返します。 |
| GCHARS (halfword) | 常にゼロを返します。 |
| GCODES (halfword) | 常にゼロを返します。 |

| オプション | 説明 |
| --- | --- |
| GMMI (char-1) | good morning メッセージがないことを示す値 X'00' を常に返します。 |
| HILIGHT (char-1) | クライアントによっては、拡張強調表示機能がないことを示す値 X'00' を返すか、拡張強調表示機能がサポートされていることを示す値 X'FF' を返します。 |
| INITPARM (address) | 常に空アドレスを返します。 |
| INITPARMLEN (halfword) | 常にゼロを返します。 |
| INPARTN (char-2) | 常にスペースを返します。 |
| KATAKANA (char-1) | 基本機能としてカタカナをサポートしないことを示す値 X'00' を常に返します。 |
| LDCMNEM (char-1) | オーバーフローが起こっていないことを示す値 X'00' を常に返します。 |
| LDCNUM (char-1) | オーバーフローが起こっていないことを示す値 X'00' を常に返します。 |
| MAPCOLUMN (halfword) | 常にゼロを返します。 |
| MAPHEIGHT (halfword) | 常にゼロを返します。 |
| MAPLINE (halfword) | 常にゼロを返します。 |
| MAPWIDTH (halfword) | 常にゼロを返します。 |
| MSRCONTROL (char-1) | 端末が MSR 制御をサポートしないことを示す値 X'00' を常に返します。 |
| NATLANGINUSE (char-1) | 常に値 X'00' を返します。 |
| NETNAME (char-8) | 端末のタイプが 3270 の場合、論理ユニット名が返されます。それ以外の場合は、論理 tty 名が返されます。 |
| NEXTTRANSID (char-4) | 常にスペースを返します。 |
| NUMTAB (char-1) | タブが必要でないことを示す値 X'00' を常に返します。 |
| OPCLASS (char-3) | ユーザーが CSSN でサインオンすると、サインオンテーブル (SNT) からオペレータクラスが返されます。それ以外の場合は、TCT からオペレータクラスが返されます。SNT および TCT については、『Sun Mainframe Transaction Processing 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。 |
| OPERKEYS (char-3) | ユーザーが CSSN でサインオンすると、SNT からオペレータリソースキーの最初の 3 文字が返されます。それ以外の場合は、TCT のオペレータリソースキーの最初の 3 文字が返されます。 |
| OPID (char-3) | ユーザーが CSSN でサインオンすると、SNT からオペレータ ID が返されます。それ以外の場合は、TCT からオペレータ ID が返されます。 |

| オプション | 説明 |
|---|---|
| OPSECURITY (char-8) | ユーザーが CSSN でサインオンすると、SNT から 8 バイトの完全なオペレータセキュリティーキーが返されます。それ以外の場合は、TCT から 8 バイトの完全なオペレータセキュリティーキーが返されます。<br>メインフレームプログラムでは、3 バイト値が要求されます。Sun MTP は 8 バイト値を返すので、このコマンドを使用するすべてのプログラムは、この違いを調整するために変更する必要があります。これを行わないと、メモリーの破損の原因になります。 |
| ORGABCODE (char-4) | 常にスペースを返します。 |
| OUTLINE (char-1) | 常に値 X'00' を返します。 |
| PAGENUM (char-2) | BMS ページングが有効でない場合、現在の BMS ページ番号または INVREQ が返されます。 |
| PARTNPAGE (char-2) | パーティションが使用されていないことを示す空白を常に返します。 |
| PARTNS (char-1) | 端末がパーティションをサポートしないことを示す値 X'00' を常に返します。 |
| PARTNSET (char-6) | 設定されているアプリケーションパーティションがないことを示す空白を常に返します。 |
| PRINSYSID (char-4) | ローカルシステムで認識されている遠隔システムの接続の SysID を返します。タスクの基本機能が DTP 会話でない場合、INVREQ が返されます。 |
| PROGRAM (char-8) | 実行中のプログラムの 8 文字の名前を返します。 |
| PS (char-1) | プログラム式シンボル機能がないことを示す値 X'00' を常に返します。 |
| QNAME (char-4) | このタスクを開始した一時データキューの名前を返します。 |
| RESSEC (char-1) | 常にスペースを返します。 |
| RESTART (char-1) | 再起動ではなく通常の起動であることを示す値 X'00' を常に返します。 |
| SCRNHT (halfword) | 画面の高さを示す値を返します。 |
| SCRNWD (halfword) | 画面の幅を示す値を返します。 |
| SIGDATA (halfword) | 常にゼロを返します。 |
| SOSI (char-4) | シフトアウト/シフトイン機能がないことを示す値 X'00' を常に返します。 |
| STARTCODE (char-2) | トランザクションが開始した方法を示すコードを返します。次の開始コードがサポートされています。<br>• D: SYNCONRETURN オプションで指定されていなかった分散プログラムリンク (DPL) 要求<br>• DS: SYNCONRETURN オプションで指定された DPL 要求<br>• QD: 一時データトリガーレベル<br>• S: データがない START コマンド<br>• SD: データがある START コマンド<br>• TD: 端末入力または固定トランザクション ID |
| STATIONID (char-1) | 2980 ステーション識別子がないことを示す値 X'00' を常に返します。 |
| SYSID (char-4) | SIT からのシステム識別子を提供します。 |

| オプション | 説明 |
|---|---|
| TASKPRIORITY (halfword) | 常にゼロを返します。 |
| TCTUALENG (halfword) | TCT からの TCTUA の長さを提供します。 |
| TELLERID (char-1) | 常にスペースを返します。 |
| TERMCODE (char-2) | 端末のタイプおよびモデルを返します。端末が 3277 遠隔端末の場合、タイプは常に X'91' に等しくなります。モデルは、'2'、'4'、または '5' のどれかで、それぞれ 3270 モデル 2、4、または 5 を表します。DTP セッションでは、タイプは X'C0'で、モデルは常に '0' に設定されます。 |
| TERMPRIORITY (halfword) | 常にゼロを返します。 |
| TEXTKYBD (char-1) | 常に値 X'00' を返します。 |
| TEXTPRINT (char-1) | 常に値 X'00' を返します。 |
| TRANPRIORITY (halfword) | 常にゼロを返します。 |
| TWALENG (halfword) | PCT からの TWA の長さ。PCT については、『Sun Mainframe Transaction Processing 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。 |
| UNATTEND (char-1) | 端末が存在していることを示す値 X'00' を常に返します。 |
| USERID (char-8) | ユーザーが CSSN、CESN、またはトランザクション経路指定からの自動サインオンのいずれかの方法でサインオンした場合、そのユーザー ID を返します。それ以外の場合、トランザクションを実行している Sun MTP 領域に対するデフォルトのシステムユーザー ID を返します。 |
| USERNAME (data-area) | USERID に対応するオペレータ名の 20 文字の識別子を返します。 |
| USERPRIORITY (halfword) | 常にゼロを返します。 |
| VALIDATION (char-1) | 検査機能がないことを示す値 X'00' を常に返します。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| ABPROGRAM | ALTSCRNHT | ALTSCRNWD |
| ASRAINTRPT | ASRAPSW | ASRAREGA |
| CMDSEC | DEFSCRNHT | DEFSCRNWD |
| SUSERNAME | | |

## BIF DEEDIT

```
BIF DEEDIT FIELD(data-area)
[LENGTH(halfword)]
```

BIF DEEDIT は、データフィールドからアルファベット文字および特殊文字を削除して右寄せし、残りの桁をゼロにします。

| オプション | 説明 |
|---|---|
| FIELD | 編集するフィールドを指定します。 |
| LENGTH | フィールド長をバイトで指定します。 |

## CANCEL

```
CANCEL REQID(char-8)
[TRANSID(char-4)]
[SYSID(char-4)]]
```

CANCEL では、直前の START コマンドをキャンセルできます。キャンセルするタスクがまだ開始されていない場合にのみ有効です。

| オプション | 説明 |
|---|---|
| REQID | 直前の START コマンドで指定された REQID に一致する必要があります。 |
| SYSID | 指定されている場合、TCT–System Entries 画面のエントリに一致する必要があります。非同期処理については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |
| TRANSID | 指定されている場合、PCT のエントリに一致する必要があります。また、直前の START コマンドで指定された TRANSID に一致する必要があります。 |

## CHANGE PASSWORD

```
CHANGE PASSWORD(current-pw) (char-8)
USERID(current-uid) (char-8)
NEWPASSWORD(new-pw) (char-8)
ESMREASON(err-reason) (fullword binary)
```

CHANGE PASSWORD では、ユーザーのパスワードを変更できます。この節の最後に示す例外を除き、完全にサポートされています。新しいパスワードはただちに有効になります。

| オプション | 説明 |
|---|---|
| PASSWORD | 現在のパスワード |
| USERID | 現在のユーザー ID |
| NEWPASSWORD | 新しいパスワード |
| ESMREASON | 新しいパスワードに関するエラーの詳細 |

ユーザー ID と現在のパスワードが有効で正しく指定された場合のみ、パスワードを変更できます。そうでない場合は、次のエラー条件が返されます。

| 条件 | RESP2 | 理由 |
|---|---|---|
| NOTAUTH | 2 | 現在のパスワードが間違っています。 |
| NOTAUTH | 19 | ユーザー ID が停止されていて、ESMREASON が 901 に設定されています。 |
| NOTAUTH | 4 | 新しいパスワードが最大文字数よりも長い。ESMREASON が 902 に設定されます。 |
| NOTAUTH | 4 | 新しいパスワードが最小文字数よりも短い。ESMREASON が 903 に設定されます。 |
| NOTAUTH | 4 | 新しいパスワードが古いパスワードと同じ。ESMREASON が 904 に設定されます。 |
| NOTAUTH | 4 | 新しいパスワードは、ユーザー ID と同じまたはすべて空白。または、新しいパスワードの形式は許可されません (数字でない、またはアルファベットでない)。 |
| USERIDERR | 8 | 現在のユーザー ID が無効です。 |
| INVREQ | 10 | タスクに関連する端末がありません。 |

次のオプションはサポートされていません。

ESMRESP

# COLLECT STATISTICS

```
COLLECT STATISTICS SET(pointer-ref)
TDQUEUE(data-value)
```

COLLECT STATISTICS に対して、Sun MTP はパーティション内一時データキューに関連する単一統計項目の集合 (パーティション内の出力数) をサポートします。パーティション内出力のカウントは、その領域が開始されるとゼロに初期化されます。一時データキューに対してアクティビティーが発生すると更新され、その領域が有効な間は COLLECT STATISTICS コマンドを通じて利用できます。領域が停止して再起動すると、カウントは再度ゼロに初期化されます。

| オプション | 説明 |
|---|---|
| SET | フォーマットされた統計が入っているデータ領域のアドレスに設定されるポインタを指定します。 |
| TDQUEUE | data value は、宛先管理テーブル (DCT) からのキューの 4 文字の名前です。 |

キュー名が DCT で見つからない場合、次のエラーが返されます。

```
NOTFND
RESP2=2
```

次のオプションはサポートされていません。

| | | |
|---|---|---|
| AUTOINSTALL | CONNECTION | DISPATCHER |
| DTB | FILE | IRCBATCH |
| JOURNALNUM | LASTRESET | LASTRESETHRS |
| LASTRESETMIN | LASTRESETSEC | LSRPOOL |
| MONITOR | NODE | POOL |
| POOL TARGET | PROGAUTO | PROGRAM |
| STORAGE | SYSDUMP | TABLEMGR |
| TCLASS | TERMINAL | TRANCLASS |
| TRANDUMPCODE | TRANSACTION | TSQUEUE |
| VTAM | | |

# CONNECT PROCESS

```
CONNECT PROCESS CONVID (char-4)
PROCNAME(char-8)
PROCLENGTH(halfword)
[PIPLIST(data-area) PIPLENGTH(halfword)]
SYNCLEVEL(halfword)
[STATE(cvda)]
```

CONNECT PROCESS は、一部の例外を除きサポートされます。DTP プログラムが遠隔システムのパートナーを起動するために使用するトランザクション名および同期レベルを指定します。

| オプション | 説明 |
|---|---|
| CONVID | コマンドに関連する会話を識別します。 |
| PIPLENGTH | 指定したプロセス初期化パラメータ (PIP) リストの合計の長さ。PIPLIST を指定している場合、PIPLENGTH を指定する必要があります。 |
| PIPLIST | 遠隔システムに送信される PIP データ。PIP リストは可変長レコードで構成され、各レコードに 1 つの PIP が含まれます。 |
| PROCLENGTH | PROCNAME オプションで指定したプロセス名の長さ。 |
| PROCNAME | 遠隔システムに接続されるトランザクション。 |
| STATE | 現在の会話の状態を取得します。 |
| SYNCLEVEL | 現在のセッションの同期レベル。 |

次のオプションはサポートされていません。

SESSION

# CONVERSE (APPC)

```
CONVERSE FROM(data-area)
FROMLENGTH(halfword)|FROMFLENGTH(fullword)
[INTO(data-area)|SET(pointer-ref)]
[TOLENGTH(halfword-data-area)|
TOFLENGTH(fullword-data area)]
[MAXLENGTH(halfword)|MAXFLENGTH(fullword)]
[NOTRUNCATE]
[CONVID(char-4)]
[STATE (cvda)]
```

CONVERSE は SEND と、あとに続く RECEIVE との組み合わせで使用します。3278/3279-タイプの端末および DTP 会話でサポートされます。他のタイプ端末ではサポートされません。

| オプション | 説明 |
|---|---|
| CONVID | コマンドに関連する会話を識別します。 |
| FROM | パートナートランザクションに送信されるデータ |
| FROMFLENGTH | FROMLENGTH の代替フルワード |
| FROMLENGTH | 送信されるデータのハーフワード 2 進値での長さ |
| INTO | 現在の会話のパートナーに接続しているアプリケーションプログラムからのデータを受け取るための、アプリケーションのターゲットデータ領域。 |
| MAXFLENGTH | MAXLENGTH の代替フルワード |
| MAXLENGTH | CONVERSE コマンドに応答して、CICS が受け取るデータの最大量。INTO が指定されていると、MAXLENGTH が入力として TOLENGTH より優先されます。SET が指定されていると、プログラムが 1 度に受け取るデータの量を MAXLENGTH により制限できます。<br>データ長が指定の値を超えていて、NOTRUNCATE オプションが存在しない場合、データは指定の値に切り捨てられ、LENGERR 条件が発生します。TOLENGTH オプションで指定されているデータ領域は、データの元の長さに設定されます。<br>データ長が指定の値を超えていて、NOTRUNCATE オプションが存在する場合、CICS が残りのデータを保持し、そのデータを後続の RECEIVE コマンドで取得できます。<br>MAXLENGTH に引数がない場合、CICS は TOGENGTH オプションのデフォルトの値を使用します。 |
| NOTRUNCATE | 利用できるデータが要求された長さを超える場合、データの残りを破棄しないで、後続の RECEIVE コマンドによって取り出されるように保持します。 |

| オプション | 説明 |
|---|---|
| SET | パートナートランザクションから取り出すデータのアドレスに設定されるポインタ参照。他のコマンドや文によって変更されないかぎり、ポインタ参照は、次の CONVERSE (APPC) コマンドまたはタスクの終了まで有効です。 |
| STATE | 現在の会話の状態を取得します。 |
| TOFLENGTH | TOLENGTH の代替フルワード |
| TOLENGTH | 受信されるデータのハーフワード 2 進値での長さ。INTO を指定して MAXLENGTH を省略すると、このオプションはプログラムが受け入れる最大長を指定します。 |

# CONVERSE (LUTYPE2/LUTYPE3)

```
CONVERSE FROM(data-area)
{FROMLENGTH(halfword)|
FROMFLENGTH(fullword)}
{INTO(data-area)|SET(pointer-ref)}
{TOLENGTH(halfword-data-area)|
TOFLENGTH(fullword-data area)}
[MAXLENGTH(halfword)|MAXFLENGTH(fullword)]
[NOTRUNCATE]
[[ERASE] [CTLCHAR(char-1)]]
```

CONVERSE は SEND と、あとに続く RECEIVE との組み合わせで使用します。後述の一部の例外を除き、3278/3279-タイプの端末および DTP 会話でサポートされます。他のタイプ端末ではサポートされません。

| オプション | 説明 |
|---|---|
| CTLCHAR | CONVERSE コマンドを制御する 1 バイトの書き込み制御文字。COBOL ユーザーは、この文字を含むデータ領域を指定する必要があります。このオプションを省略すると、変更されたすべてのデータタグはゼロにリセットされ、キーボードが復元されます。 |
| ERASE | 書き込みが発生する前に、バッファーまたは表示イメージを消去してカーソルを画面左上に戻します。 |
| FROM | パートナートランザクションに送信されるデータ |
| FROMFLENGTH | FROMLENGTH の代替フルワード |
| FROMLENGTH | 送信されるデータのハーフワード 2 進値での長さ |
| INTO | 現在の会話のパートナーに接続しているアプリケーションプログラムからのデータを受け取るための、アプリケーションのターゲットデータ領域。 |
| MAXFLENGTH | MAXLENGTH の代替フルワード |

| オプション | 説明 |
|---|---|
| MAXLENGTH | CONVERSE コマンドに応答して、CICS が受け取るデータの最大量。INTO が指定されていると、MAXLENGTH が入力として TOLENGTH より優先されます。SET が指定されていると、プログラムが 1 度に受け取るデータの量を MAXLENGTH により制限できます。<br>データ長が指定の値を超えていて、NOTRUNCATE オプションが存在しない場合、データは指定の値に切り捨てられ、LENGERR 条件が発生します。TOLENGTH オプションで指定されているデータ領域は、データの元の長さに設定されます。<br>データ長が指定の値を超えていて、NOTRUNCATE オプションが存在する場合、CICS が残りのデータを保持し、そのデータを後続の RECEIVE コマンドで取得できます。<br>MAXLENGTH に引数がない場合、CICS は TOGENGTH オプションのデフォルトの値を使用します。 |
| NOTRUNCATE | 利用できるデータが要求された長さを超える場合、データの残りを破棄しないで、あとで後続の RECEIVE コマンドによって取得できるように保持します。 |
| SET | パートナートランザクションから取り出すデータのアドレスに設定されるポインタ参照。他のコマンドや文によって変更されないかぎり、ポインタ参照は、次の CONVERSE (APPC) コマンドまたはタスクの終了まで有効です。 |
| TOFLENGTH | TOLENGTH の代替フルワード |
| TOLENGTH | ハーフワードの 2 進値として受信されるデータの長さ。INTO を指定して MAXLENGTH を省略すると、このオプションはプログラムが受け入れる最大長を指定します。 |

次のオプションはサポートされていません。

```
ASIS          DEFRESP       STRFIELD
```

## DELAY

```
DELAY [INTERVAL(packed-7)|TIME(packed-7)|
FOR[HOURS(hh)][MINUTES(mm)][SECONDS(ss)]|
UNTIL[HOURS(hh)][MINUTES(mm)][SECONDS(ss)]]
[REQID(char-8)]
```

DELAY は、完全にサポートされています。指定した期間、タスクの処理を中断します。

| オプション | 説明 |
|---|---|
| FOR | 延期する期間を指定します。 |
| HOURS | 0 ～ 99 のフルワードの 2 進値。 |
| INTERVAL | DELAY コマンドが発行されてから待機する時間の長さ。 |
| MINUTES | 0 ～ 59 または 0 ～ 5999 のフルワードの 2 進値。 |
| REQID | コマンドを識別する一意の名前。一時記憶域識別子として使用します。 |
| SECONDS | 0 ～ 59 または 0 ～ 359999 のフルワードの 2 進値。 |
| TIME | 中断する時間。 |
| UNTIL | 再開する時刻を指定します。 |

# DELETE

```
DELETE DATASET(dataset)|FILE(dataset)
[RIDFLD(data-area)]
[KEYLENGTH(halfword)]
[GENERIC [NUMREC (halfword-data-area)]]
[SYSID(char-4)]
[RBA|RRN]
```

DELETE は、VSAM データセットで完全にサポートされています。データセットからレコードまたはレコードのグループを削除します。

| オプション | 説明 |
|---|---|
| DATASET | データセット名。SYSID がコード化されていない場合、FCT のエントリに一致する必要があります。詳細については、『Sun Mainframe Transaction Processing 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。 |
| FILE | データセット名。 |
| GENERIC | 検索キーは、KEYLENGTH オプションで指定された長さの総称キーです。KSDS ファイルとともにのみ使用できます。 |
| KEYLENGTH | RIDFLD オプションで指定されるキーの長さ。 |
| NUMREC | CICS が削除したレコードの数をセットするデータ領域。 |
| RBA | RIDFLD オプションで指定されたレコード識別フィールドが相対バイトアドレスを示すように指定します。 |
| RIDFLD | レコード識別フィールド。キー、相対バイトアドレス、または相対レコード番号の指定が可能です。 |
| RRN | RIDFLD オプションで指定されたレコード識別フィールドが相対レコード番号を示します。 |
| SYSID | 指定されている場合、TCT–System Entries 画面のエントリに一致する必要があります。機能シップについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |

## DELETEQ TD

```
DELETEQ TD QUEUE(char-4)
[SYSID(char-4)]
```

DELETEQ TD は、完全にサポートされています。QUEUE オプションで指定された一時データキューに関連するすべてのデータを削除します。

| オプション | 説明 |
|---|---|
| QUEUE | SYSID がコード化されていない場合、DCT の宛先エントリに一致する必要があります。DCT については、『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』を参照 |
| SYSID | 指定されている場合、TCT–System Entries 画面のエントリに一致する必要があります。TCT については、『Sun Mainframe Transaction Processing 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。 |

## DELETEQ TS

```
DELETEQ TS QUEUE(char-8)
[SYSID(char-4)]
```

DELETEQ TS は、完全にサポートされています。QUEUE オプションで指定された一時記憶域キューに関連するすべてのデータを削除します。

| オプション | 説明 |
|---|---|
| QUEUE | SYSID がコード化されていない場合、キューが回復可能または遠隔であるときは、TST にエントリを行う必要があります。TST については、『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』を参照 |
| SYSID | 指定されている場合、TCT–System Entries 画面のエントリに一致する必要があります。機能シップについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |

## DEQ

```
DEQ RESOURCE(data-area)
[LENGTH(halfword)]
[MAXLIFETIME(cvda) [LUW|TASK]]
```

DEQ は、完全にサポートされています。以前にキューに入れられたリソースを解除して、利用できるようにします。

| オプション | 説明 |
|---|---|
| LENGTH | リソースの長さがデータ値によって指定されます。1 ～ 255 のハーフワードの 2 進値。 |
| MAXLIFETIME | 待機が解除されるまでの ENQ の期間。渡される値は LUW および TASK です。<br>LUW: デフォルト。ENQ を作業論理ユニットで取得します。<br>TASK: ENQ をタスク期間で取得します。 |
| RESOURCE | キューから解除されるリソース、またはリソースを含むデータ領域。データ領域である場合、LENGTH オプションを使用する必要があります。 |

## DUMP

```
DUMP DUMPCODE(char-4)
[FROM(data-area)[LENGTH(halfword) |
FLENGTH(fullword)]]
[COMPLETE]
[TASK]
[STORAGE]
[PROGRAM]
[TERMINAL]
[TABLES]
[DCT]
[FCT]
[PCT]
[PPT]
[SIT]
[TCT]
```

DUMP は、Sun MTP の特定のトランザクション記憶域をダンプします。内部テーブル領域の形式は CICS とは異なりますが、このコマンドは、完全にサポートされています。

**注 –** このコマンドは、DUMP TRANSACTION と同じ機能を実行します。ただし、DUMP TRANSACTION コマンドはサポートされていません。

| オプション | 説明 |
|---|---|
| COMPLETE | タスク、すべてのテーブル、およびすべての制御ブロックに関連する、すべての主記憶域をダンプします。 |
| DUMPCODE | ダンプを識別する名前。 |
| FLENGTH | FROM オプションで指定される記憶領域の長さ (フルワード 2 進値)。 |
| FROM | 指定したデータ領域をダンプします。 |
| LENGTH | FROM オプションで指定されるデータ領域の長さ (ハーフワード 2 進値)。 |
| PROGRAM | タスクに関連するプログラム記憶域をダンプします。 |
| STORAGE | タスクに関連する記憶領域をダンプします。 |
| TABLES | DCT、FCT、PCT、PPT、SIT、および TCT をダンプします。特定のテーブルをダンプする場合、特定のテーブルオプションを使用します。 |
| TASK | タスクに関連する記憶領域をダンプします。このオプションは、STORAGE オプションよりも多くの記憶領域をダンプします。詳細は、IBM CICS のマニュアルを参照してください。 |
| TERMINAL | 端末に関連する記憶領域をダンプします。 |

## ENDBR

```
ENDBR DATASET(dataset)|FILE(dataset)
[REQID(halfword)]
[SYSID(char-4)]
```

ENDBR は、VSAM データセットで完全にサポートされています。データセットのブラウズ操作を終了します。ISAM および DAM データセットはサポートされません。

| オプション | 説明 |
|---|---|
| DATASET | データセット名。SYSID がコード化されていない場合、FCT のエントリに一致する必要があります。詳細については、『Sun Mainframe Transaction Processing 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。 |
| FILE | ブラウズしているファイルのデータセット名。 |

| オプション | 説明 |
|---|---|
| REQID | ブラウズの固有の要求識別子。指定されていない場合、デフォルト値のゼロが使用されます。 |
| SYSID | 指定されている場合、TCT–System Entries 画面のエントリに一致する必要があります。機能シップについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |

## ENQ

```
ENQ RESOURCE(data-area)
[LENGTH(halfword)]
[MAXLIFETIME(cvda) | UOW | TASK]]
[NOSUSPEND]
```

ENQ は、完全にサポートされています。他のタスクから同時に使用されないように保護するリソースをキューに入れます。

UOW、TASK、または MAXLIFETIME オプションが指定されてなく、かつタスクはリソースが利用できるようになるまで待機する必要がある場合、そのリソースがキューに入れられた時点で、タスクは UOW または TASK がリソースに割り当てられているものとみなします。

| オプション | 説明 |
|---|---|
| LENGTH | リソースの長さがデータ値によって指定されます。1 ～ 255 のハーフワードの 2 進値。 |
| MAXLIFETIME | 自動的に待機が解除されるまでの ENQ の期間。渡される値は UOW および TASK です。 |
| UOW | デフォルト。ENQ の期間は作業単位です。LUW オプションは、逆互換性でサポートされています。 |
| TASK | ENQ の期間はタスク。 |
| NOSUSPEND | ENQ コマンドで指定したリソースが使用不能でも、アプリケーションプログラムは中断されずに ENQBUSY 状態が発生します。 |
| RESOURCE | キューに入れられるリソース。リソースを表すアドレスを持つ領域またはリソースを含む変数によって指定します。変数を指定する場合、LENGTH オプションを使用する必要があります。 |

## ENTER

```
ENTER TRACEID(halfword)
[FROM(char-8)]
[RESOURCE(char-8)]
[ENTRYNAME(char-8)]
```

ENTER は、ユーザーデータをトレーステーブルに配置します。DUMP コマンドを使用してこのデータを印刷できます。

| オプション | 説明 |
|---|---|
| ENTRYNAME | エントリ名。 |
| FROM | トレーステーブルエントリのデータフィールドに入力される内容を含んだデータフィールド。 |
| RESOURCE | トレーステーブルエントリのリソースフィールドに入力される 8 文字の名前。 |
| TRACEID | トレース識別子。 |

次のオプションはサポートされていません。

ACCOUNT　　FROMLENGTH　　MONITOR

PERFORM　　TRACENUM

## EXTRACT ATTRIBUTES

```
EXTRACT ATTRIBUTES [CONVID(char-4)|SESSION(char-4)]
STATE(cvda)
```

EXTRACT ATTRIBUTES は、完全にサポートされています。DTP 会話の現在の状態をアプリケーションプログラムに通知します。

| オプション | 説明 |
|---|---|
| CONVID | コマンドに関連する会話を識別します。このオプションは新しいプログラムで使用します。 |
| SESSION | セッションの記号識別子。古いリリースとの互換性のために使用されます。 |
| STATE | トランザクションプログラムの状態を取得します。 |

# EXTRACT CERTIFICATE

```
EXTRACT CERTIFICATE(pointer-ref)
[LENGTH(data-area)]
OWNER | ISSUER
[COMMONNAME(pointer-ref)]
[COMMONNAMELEN(data-area)]
[COUNTRY(pointer-ref)]
[COUNTRYLEN(data-area)]
[STATE(pointer-ref)]
[STATELEN(data-area)]
[LOCALITY(pointer-ref)]
[LOCALITYLEN(data-area)]
[ORGANIZATION(pointer-ref)]
[ORGANIZATLEN(data-area)]
[ORGUNIT(pointer-ref)]
[ORGUNITLEN(data-area)]
```

EXTRACT CERTIFICATE では、SSL サーバー (unikixssl) との SSL ハンドシェーク時に SSL クライアントから受信した X.509 証明書の情報をアプリケーションが取得できます。この証明書には、証明書の所有者 (または件名) を識別するフィールド、および証明書を発行した認証局 (CA) を識別するフィールドがあります。必要なフィールドを選択するには、OWNER または ISSUER オプションを指定します。1 つのコマンドで OWNER と ISSUER の両方のフィールドを取り出すことはできません。

| オプション | 説明 |
|---|---|
| CERTIFICATE | クライアントから受け取る完全証明書 (識別名) のアドレスに設定されるポインタ参照を指定します。<br>クライアントが証明書を提示しない場合、文字列 no certificate が返されます。no certificate が返される場合、そのほかのオプションはゼロ長を返します。 |
| LENGTH | 完全証明書 (識別名) の長さに設定されるフルワードの 2 進データ領域を指定します。 |
| OWNER | このコマンドによって返される値が、証明書の所有者を参照するように指定します。 |
| ISSUER | このコマンドによって返される値が、この証明書を発行した CA の証明書を参照するように指定します。 |
| COMMONNAME | クライアント証明書から受け取る共通名に設定されるポインタ参照を指定します。 |
| COMMONNAMLEN | クライアント証明書から受け取る共通名の長さに設定されるフルワードの 2 進データ領域を指定します。 |
| COUNTRY | クライアント証明書から受け取る国のアドレスに設定されるポインタ参照を指定します。 |

| オプション | 説明 |
|---|---|
| COUNTRYLEN | クライアント証明書から受け取る国の長さに設定されるハーフワードの 2 進データ領域を指定します。 |
| STATE | クライアント証明書から受け取る州または県のアドレスに設定されるポインタ参照を指定します。 |
| STATELEN | クライアント証明書から受け取る州または県の長さに設定されるハーフワードの 2 進データ領域を指定します。 |
| LOCALITY | クライアント証明書から受け取る州または県以下の区域のアドレスに設定されるポインタ参照を指定します。 |
| LOCALITYLEN | クライアント証明書から受け取る州または県以下の区域の長さに設定されるハーフワードの 2 進データ領域を指定します。 |
| ORGANIZATION | クライアント証明書から受け取る組織のアドレスに設定されるポインタ参照を指定します。 |
| ORGANIZATLEN | クライアント証明書から受け取る組織の長さに設定されるハーフワードの 2 進データ領域を指定します。 |
| ORGUNIT | クライアント証明書から受け取る組織単位のアドレスに設定されるポインタ参照を指定します。 |
| ORGUNITLEN | クライアント証明書から受け取る組織単位の長さに設定されるハーフワードの 2 進データ領域を指定します。 |

次のオプションはサポートされていません。

SERIALNUM SERIALNUMLEN USERID

EXTRACT CERTIFICATE API を使用するトランザクションを、SSL クライアントが unikixssl によって起動しなかった場合、INVREQ RESP という値が返されます。

# EXTRACT PROCESS

```
EXTRACT PROCESS[CONVID(char-4)]
[PROCNAME(char-8) PROCLENGTH(halfword)
[MAXPROCLEN(halfword)]]
[SYNCLEVEL(halfword)]
[PIPLIST(pointer-ref) PIPLENGTH(halfword)
```

EXTRACT PROCESS は、完全にサポートされています。現在の DTP 会話の情報を取得します。

| オプション | 説明 |
|---|---|
| CONVID | コマンドに関連する会話を識別します。 |
| MAXPROCLEN | PROCNAME のバッファー長。指定されていない場合、バッファーは 32 バイトであると見なされます。 |
| PIPLENGTH | プロセス初期化パラメータ (PIP) リストの合計の長さが返される、ハーフワードの 2 進データ領域。PIPLIST を指定している場合、PIPLENGTH を指定する必要があります。 |
| PIPLIST | PIP リストが入っているデータ領域のアドレスに設定されるポインタ参照。このリストには、CONNECT PROCESS コマンドのリストと同じ形式の可変長レコードが含まれます。ゼロの値が返されると、PIP データを受け取っていないことを示します。 |
| PROCLENGTH | プロセス名の長さに設定されるハーフワードのデータ領域。PROCNAME を指定している場合、このオプションを指定する必要があります。 |
| PROCNAME | タスクを開始した遠隔システムによって指定されるプロセス名を受け取るためのデータ領域。 |
| SYNCLEVEL | 同期レベルの値 |

## FORMATTIME

```
FORMATTIME ABSTIME(doubleword)
[YYDDD(char-6-data-area)]
[YYYYDDD(char-8-data-area)]
[YYMMDD(char-8-data-area)]
[YYYYMMDD(char-10-data-area)]
[YYDDMM(char-8-data-area)]
[YYYYDDMM(char-10-data-area)]
[DDMMYY(char-8-data-area)]
[DDMMYYYY(char-10-data-area)]
[MMDDYY(char-8-data-area)]
[MMDDYYYY(char-10-data-area)]
[DATE(char-8-data-area)]
[DATEFORM(char-6-data-area)]
[DATESEP [(char-1)]]
[DAYCOUNT(fullword-data-area)]
[DAYOFWEEK(fullword-data-area)]
[DAYOFMONTH(fullword-data-area)]
[MONTHOFYEAR(fullword-data-area)]
[YEAR(fullword-data-area)]
[TIME(char-8-data-area)TIMESEP(char-1)]]
```

FORMATTIME は、完全にサポートされています。ASKTIME コマンドの ABSTIME オプションから返される時刻をフォーマットします。ここでは、「年/月/日」に関する各種のオプションについては説明しません。

| オプション | 説明 |
|---|---|
| ABSTIME | パック 10 進数で表される時刻のデータ値。 |
| DATE | 日付を受け取る変数。 |
| DATEFORM | インストール時に定義した日付の形式。このオプションでは DATE を使う必要があります。 |
| DATESEP | 年、月、日の値の間に挿入される区切り文字。省略すると、区切り文字は挿入されません。 |
| DAYCOUNT | 1900 年 1 月 1 日からの日数を返します。 |
| DAYOFWEEK | 週の各曜日に対応する番号を返します。日曜が 0、土曜が 6 |
| DAYOFMONTH | 月の日にちに対応する番号を返します。 |
| MONTHOFYEAR | 年の各月に対応する番号を返します。1 月が 1、12 月が 12 |
| TIME | 24 時間表示による 8 文字の値。区切り文字は TIMESEP オプションで指定します。 |

| オプション | 説明 |
|---|---|
| TIMESEP | 返される時間の区切り文字。このオプションを省略すると、区切り文字は使用されません。(char-1)を省略する場合は、コロンが使用されます。 |
| YEAR | 年を表す 4 桁の数字。 |

## FREE

```
FREE [CONVID(char-4)]
[STATE(cvda)]
```

FREE は、完全にサポートされています。DTP 会話の割り当てを解除するために使用します。

| オプション | 説明 |
|---|---|
| CONVID | 解放するセッションを識別します。 |
| STATE | 現在の会話の状態を取得します。 |

## FREEMAIN

```
FREEMAIN DATA(data-area)
```

FREEMAIN は、完全にサポートされています。アプリケーションプログラムが以前に取得した主記憶域を解放します。

## GETMAIN

```
GETMAIN SET(pointer-ref)
LENGTH(halfword) | FLENGTH(fullword)
[INITIMG(char-1)]
[NOSUSPEND]
[SHARED]
```

GETMAIN は、後述の一部の例外を除きサポートされます。アプリケーションプログラムで使用する主記憶域を取得します。通常、このコマンドを使って取得する記憶領域は、トランザクションの仮想アドレス空間から取得され、他のトランザクションからはアクセスできません。記憶領域を SHARED オプションで取得する場合、記憶領域は共有メモリーから取得されます。共有メモリーアドレスは、トランザクション間での一貫性が保証されます。このオプションで取得した記憶領域は、明示的に解放する必要があります。

| オプション | 説明 |
|---|---|
| FLENGTH | フルワードの 2 進形式の必要な記憶領域のバイト数。 |
| INITIMG | オプションの 1 バイトの初期化値。 |
| LENGTH | ハーフワードの 2 進形式の必要な記憶領域のバイト数。 |
| NOSUSPEND | 記憶領域が利用できるかどうかはシステムの他の部分のイベントに左右されないので、影響はありません。 |
| SET | 取得した主記憶域のアドレスにポインタ参照を設定します。 |
| SHARED | 要求したタスクの終了時、GETMAIN コマンドで取得した記憶領域の自動解放を回避します。 |

次のオプションはサポートされていません。

BELOW

## HANDLE ABEND

```
HANDLE ABEND PROGRAM(char-8)|LABEL(program-label)|
CANCEL|RESET
```

HANDLE ABEND は、完全にサポートされています。異常終了が発生した場合に実行するアクションを指定します。

| オプション | 説明 |
|---|---|
| CANCEL | 制御しているアプリケーションプログラムの論理レベルで、以前に設定した出口を取り消します。 |
| LABEL | 制御分岐が異常終了したプログラムのラベル。 |
| PROGRAM | タスクが異常終了した場合に、制御を受け取るプログラム名を識別します。 |
| RESET | HANDLE ABEND CANCEL コマンドまたは CICS によって取り消された出口を再度有効にします。 |

## HANDLE AID

```
HANDLE AID option [(program-label)]
[option [(program-label)]] ...
```

HANDLE AID は、この節の最後に示す例外を除いてサポートされています。アテンション識別子 (AID) を端末から受け取ったときに制御を渡すラベルを指定します。1 つの HANDLE AID コマンドで、オプションを 12 まで指定できます。

| | | |
|---|---|---|
| CLEAR | ENTER | NULL |
| PA1-PA3 | PF1-PF24 | |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| CLRPARTN | LIGHTPEN | MRSE |
| OPERID | STRF | TRIGGER |

## HANDLE CONDITION

```
HANDLE CONDITION condition [(program-label)]
[condition [(program-label)]] ...
```

HANDLE CONDITION は、後続の CICS コマンドが例外条件を発生した場合に制御が渡されるプログラムのポイントを指定します。1 つの HANDLE CONDITION コマンドで、条件を 16 まで指定できます。

次のオプションはサポートされていません。

| | | |
|---|---|---|
| ALLOCERR | CBIDERR | DSSTAT |
| ENDINPT | ENVDEFERR | EODS |
| EOF | IGREQCD | INBFMH |
| INVERRTERM | INVLDC | INVPARTN |
| INVPARTNSET | INVTSREQ | LOADING |
| NONVAL | NOPASSBKRD | NOPASSBKWR |
| NOSPOOL | NOSTART | OPENERR |
| PARTNFAIL | RDATT | RETPAGE |
| ROLLEDBACK | RTEFAIL | RTESOME |
| SELNERR | SESSBUSY | SESSIONERR |

| | | |
|---|---|---|
| SPOLBUSY | SPOLERR | STRELERR |
| SUPPRESSED | UNEXPIN | WRBRK |
| USERIDERR | | |

## IGNORE CONDITION

```
IGNORE CONDITION condition [condition] ...
```

IGNORE CONDITION は、後続の CICS コマンドで例外条件を発生した場合に無視される例外条件を指定します。

次のオプションはサポートされていません。

| | | |
|---|---|---|
| ALLOCERR | CBIDERR | DSSTAT |
| ENDINPT | ENVDEFFERR | EOC |
| EODS | EOF | ERROR |
| EXPIRED | IGREQCD | IGREQID |
| INBFMH | INVERRTERM | INVLDC |
| INVPARTN | INVPARTNSET | INVTSREQ |
| LOADING | NONVAL | NOPASSBKRD |
| NOPASSBKWR | NOSPACE | NOSPOOL |
| NOSTART | OPENERR | PARTNFAIL |
| RDATT | RETPAGE | ROLLEDBACK |
| RTEFAI | RTESOME | SELNERR |
| SESSBUSY | SESSIONERR | SIGNAL |
| SPOLBUSY | SPOLERR | STRELERR |
| SUPPRESSED | SYSBUSY | TERMERR |
| UNEXPIN | WRBRK | USERIDERR |

# INQUIRE CONNECTION

```
INQUIRE CONNECTION(char-4)
[NETNAME(char-8)]
[ACCESSMETHOD](cvda)
[PROTOCOL(cvda)]
[SERVSTATUS(cvda)]
[CONNSTATUS(cvda)]
[ACQSTATUS(cvda)]
```

INQUIRE CONNECTION は、遠隔システムまたは CICS 領域への名前付き接続 (システム ID) についての情報を取得します。

| オプション | 説明 |
|---|---|
| ACCESSMETHOD | Sun MTP が次の CVDA 値をサポートします。<br>VTAM IRC INDIRECT XM |
| CONNSTATUS | Sun MTP が次の CVDA 値をサポートします。<br>ACQUIRED RELEASED |
| NETNAME | 返される遠隔システム名を示す 8 文字のフィールド。 |
| PROTOCOL | Sun MTP が次の CVDA 値をサポートします。<br>NOTAPPLIC APPC LU61 |
| SERVSTATUS | Sun MTP が次の CVDA 値をサポートします。<br>INSERVICE OUTSERVICE |
| ACQSTATUS | CONNSTATUS オプションと同じ値を返し、互換性のためにだけ保持されます。新しいアプリケーションでは CONNSTATUS を使用します。CONNSTATUS と ACQSTATUS は同時に使用できません。<br>Sun MTP は次の CVDA 値をサポートします。<br>ACQUIRED RELEASED |

次のオプションはサポートされていません。

AUTOCONNECT EXITTRACING PENDSTATUS

XLNSTATUS ZCPTRACINGS

# INQUIRE FILE

```
INQUIRE FILE (char-8)
[ENABLESTATUS(cvda)]
[KEYLENGTH(data-area]
[KEYPOSITION(data-area)]
[OPENSTATUS(cvda)]
[RECORDSIZE(fullword binary data-area)]
[REMOTESYSTEM(char-4)]
[TYPE(cvda)]
```

INQUIRE FILE は、VSAM データセットに関する情報を返します。次のオプションがサポートされています。代替索引ファイルではサポートされません。

| オプション | 説明 |
|---|---|
| ENABLESTATUS | ファイルがアプリケーションからアクセスできるかどうかを指定します。Sun MTP は次の CVDA 値をサポートします。<br>ENABLED DISABLED UNENABLED |
| FILE | 問い合わせる VSAM データセットを指定します。これは FCT の定義データセット名です。 |
| KEYLENGTH | VSAM KSDS データセットに関連付けられたファイルのレコードキーの長さを示す、フルワードのバイナリフィールドを返します。<br>• ファイルが閉じていて、キーの長さがファイル定義で定義されていない場合は、返される値は 0 (ゼロ) となります。<br>• ファイルが閉じていて、キーの長さがファイル定義で定義されている場合は、ファイル定義からの値が返されます。<br>• ファイルが開いている場合は、キーの長さの値が、関連付けられているデータセットから返されます。 |
| KEYPOSITION | レコードの開始に対する各レコードのキーフィールド開始位置を示す、フルーワードのバイナリフィールドを返します。開始は、位置 0 となります。キーがない場合、またはファイルが開いていない場合は、キー位置でゼロの値が返されます。 |
| OPENSTATUS | ファイルがオープンまたはクローズのどちらの状態にあるかを指定します。Sun MTP が次の CVDA 値をサポートします。<br>OPEN CLOSED |
| RECORDSIZE | 固定長レコードの実際のサイズ、または可変長レコードの最大サイズです。 |
| REMOTESYSTEM | ファイルが遠隔である場合、遠隔システムの名前。 |
| TYPE | VSAM データセットのレコードの編成。Sun MTP は次の値をサポートします。<br>ESDS KSDS RRDS |

Sun MTP はファイルの遷移状態という概念をサポートしないので、OPENING、CLOSING、DISABLING などの値をサポートしません。

## INQUIRE PROGRAM

```
INQUIRE PROGRAM (8-char-data-value)
[EXECUTIONSET(cvda)]
[LANGUAGE(cvda)]
[PROGTYPE(cvda)]
[REMOTENAME(data-area)]
[REMOTESYSTEM(data-area)]
[RESCOUNT(data-area)]
[STATUS(cvda)]
[TRANSID(data-area)]
[USECOUNT(data-area)]
```

INQUIRE PROGRAM コマンドは、システムに定義されているプログラムについての情報を返します。次のオプションがサポートされます。

| オプション | 説明 |
|---|---|
| PROGRAM | Sun MTP に指定されているプログラム名。指定したプログラムが見つからない場合、PGMIDERR が設定され、RESP2 コードが 1 に設定されます。 |
| EXECUTIONSET | プログラムが CICS API の分散プログラムリンクのサブセットに制限されるかを示す CVDA 値を返します。実行可能プログラムに対してのみ適用され、プログラムがローカルで起動されるときのみ API を制御します。CVDA 値は、次のとおりです。<br>DPLSUBSET: プログラムが常に制限されます。<br>FULLAPI: プログラムが遠隔で起動される場合を除き、プログラムは制限されません。 |
| LANGUAGE | モジュールが記述された言語を示す CVDA 値を返します。その値は、PPT の Type インジケータに相当する次の CVDA 値です。<br>ASSEMBLER: 150<br>C: 149<br>COBOL: 151<br>PLI または PL1: 152<br>JAVA: 1080<br>NOTAPPLIC: 1<br>NOTAPPLIC は 1 の CVDA 値を返します。これは、モジュールがマップセットまたは遠隔プログラムなので、LANGUAGE が適用されないことを示します。 |
| PROGTYPE | プログラムタイプを示す次の CVDA 値を返します。<br>マップ: 155<br>プログラム: 154 |

| オプション | 説明 |
|---|---|
| REMOTENAME | PROGRAM 定義の REMOTESYSTEM オプションで名付けられた領域で、プログラムが認識される 8 文字の名前を返します。 |
| REMOTESYSTEM | プログラムが定義される領域の 4 文字の名前を返します。 |
| RESCOUNT | 照会時に、そのモジュールを独立して同時に使用している数をフルワードの 2 進値カウントで返します。モジュールが遠隔プログラムの場合、-1 が返されます。 |
| STATUS | 常に ENABLED 23 を返します。 |
| TRANSID | プログラムが遠隔で実行されるトランザクションの 4 文字の名前を返します。この値は、PROGRAM 定義の TRANSID オプションから取得され、遠隔として定義されているプログラムに対してのみ適用されます。 |
| USECOUNT | 領域が起動してからのそのモジュールの使用回数の合計を、フルワードの 2 進値カウントで返します。モジュールが遠隔プログラムの場合、-1 が返されます。 |

## INQUIRE REQID

```
INQUIRE REQID (data-area)
TERMID (data-area)
TRANSID (data-area)
```

INQUIRE REQID コマンドは、要求についての情報を返します。REQID は、どの要求に対して照会が行われたかを示す、要求の 8 文字の識別子です。次のオプションがサポートされます。

| オプション | 説明 |
|---|---|
| TERMID | 要求を作成した START コマンドの TERMID オプションで指定された 4 文字の端末識別子を返します。 |
| TRANSID | 要求を作成した START コマンドの TRANSID オプションで指定された 4 文字のトランザクション識別子を返します。 |

REQID が見つかると、RESP/RESP2 フィールドに通常のエラーフリー条件が返されます。

REQID が見つからない場合、RESP は NOTFND 条件設定され、RESP2 は 1 に設定されます。

# INQUIRE SYSTEM

```
INQUIRE SYSTEM [CICSSTATUS(cvda)]
[JOBNAME(data-area)]
[APPLID(data-area)]
```

INQUIRE SYSTEM コマンドは、実行している領域についての情報を返します。

| オプション | 説明 |
|---|---|
| APPLID | システム初期設定テーブル (SIT) から 8 文字のアプリケーション名を返します。 |
| CICSSTATUS | 実行している領域の状態。サポートされる値は次のとおりです。<br>STARTUP: 領域は起動しているが、まだ完全に有効ではありません。<br>Value = 180<br>ACTIVE: 領域が完全に有効な状態。<br>Value = 181<br>FIRSTQUIESCE: 領域が停止の最初の休止状態。PLTSD ステージ 1 プログラムは、FIRSTQUIESCE の実行時に実行されます。<br>Value = 182<br>FINALQUIESCE: 領域が停止の最後の休止状態。PLTSD ステージ 2 プログラムは、FINALQUIESCE の実行時に実行されます。<br>Value = 183 |
| JOBNAME | SIT から 8 文字のアプリケーション名を返します。 |

# INQUIRE TASK

```
INQUIRE TASK (data-value)
[FACILITY(data-area)]
[FACILITYTYPE(cvda)]
[PROCESSID(data-area)]
[RUNSTATUS(cvda)]
[STARTCODE(data-area)]
[TRANSACTION(data-area)]
[USERID(data-area)]
```

INQUIRE TASK コマンドは、指定したユーザータスクに関する情報を検索します。ユーザータスクとは、通常オペレータが起動するユーザー定義のトランザクションまたはシステム供給のトランザクションに関連するタスクです。

| オプション | 説明 |
|---|---|
| FACILITY | 指定したタスクに関連する機能名を表す 4 文字の文字列を返します。タスクが端末から開始された場合 (TERM の FACILITYTYPE)、FACILITY は端末名を返します。タスクが一時データ定義 (TDD) で定義されている宛先トリガーレベルによって開始された場合 (DEST の FACILITYTYPE)、FACILITY は関連する一時データキューの名前を返します。<br>これ以外の場合は、空白が返されます。 |
| FACILITYTYPE | 指定したタスクを開始した機能のタイプを示す CVDA 値を返します。返される値は次のとおりです。<br>TASK: 別のタスクによってタスクが開始されました<br>TERM: 端末によってタスクが開始されました<br>DEST: TDD で定義されている宛先トリガーによってタスクが開始されました |
| PROCESSID | タスクに関連するトランザクションを実行しているプロセス (unikixtran) のプロセス ID を示す 32 ビット 2 進値を返します。Sun MTP 固有のオプションです。 |
| RUNSTATUS | タスクが実行中またはその他の状態のどちらであるか示す CVDA 値を返します。返される値は次のとおりです。<br>RUNNING: タスクが実行中です |

| オプション | 説明 |
| --- | --- |
| STARTCODE | 指定したタスクが開始された方法を示す 2 文字の文字列を返します。返される値は次のとおりです。<br>D: 分散プログラムリンク (DPL) 要求。プログラムは、基本機能に対して I/O 要求を発行することもできず、同期点要求を発行することもできません。<br>DS: プログラムが同期点要求を発行できる点を除き、D の場合と同じ DPL 要求。<br>QD: 一時データトリガーレベル。<br>S: データがない START コマンド。<br>SD: データがある START コマンド。<br>TD: 端末入力。<br>QB: バッチジョブ。 |
| TASK | タスク番号を 4 バイトパック 10 進数で指定します。 |
| TRANSACTION | タスクに関連するトランザクションがある場合、そのトランザクション名を表す 4 文字の文字列を返します。関連するトランザクションがない場合は、空白を返します。 |
| USERID | タスクに現在関連しているユーザーを識別する 8 文字の文字列を返します。 |

指定したタスクが見つからない場合、TASKIDERR 条件が返されます。

次のオプションはサポートされていません。

UOWSTATE TCLASS

# INQUIRE TASK LIST

```
INQUIRE TASK LIST LISTSIZE(fullword binary data-area)
[RUNNING]
[SUSPENDED]
[SET(pointer)]
```

INQUIRE TASK LIST コマンドは、システムに定義されているユーザータスクのリストを返します。タスク状態は明示的に指定する必要があります。

| オプション | 説明 |
|---|---|
| LISTSIZE | 照会に含めたカテゴリ内にあるタスク数を示すフルワードの 2 進フィールドを返します。このパラメータは必須です。 |
| RUNNING | 実際に実行中のタスクのみをリストします (中断されているタスクは除く)。これには、発行中のタスクが含まれます。 |
| SUSPENDED | 現在中断されているタスクのみ (イベントまたは条件を待機しているもの) をリストします。これには、発行中のタスクは含まれません。 |
| SET | 4 バイトパック 10 進数のタスク番号のリストのアドレス。リストの各エントリは、要求したカテゴリのどれかのタスクを識別します。要求したカテゴリにタスクが存在しない場合、SET ポインタが NULL になります。 |

**注 –** コマンドで、RUNNING と SUSPENDED のいずれも指定されていない場合、タスクリストにはすべてのタスクが含まれます。

次のオプションはサポートされていません。

```
DISPATCHABLE      SETTRANSID
```

# INQUIRE TDQUEUE

```
INQUIRE TDQUEUE(4-char-data-value)
[ATIFACILITY(cvda)
[ATITERMID(data-area)]
[ATITRANID(4-byte-data-area)]
[ENABLESTATUS](cvda)
[INDIRECTNAME(data-area)]
[IOTYPE(cvda)]
[NUMITEMS(fullword binary data-area)]
[RECORDFORMAT(cvda)]
[RECORDLENGTH(data-area)]
[RECOVSTATUS(cvda)]
[REMOTENAME(data-area)]
[REMOTESYSTEM(data-area)]
[OPENSTATUS(cvda)]
[TRIGGERLEVEL(fullword binary data-area)]
[TYPE(cvda)]
```

INQUIRE TDQUEUE は、指定した一時データキューについての情報を取り出します。

| オプション | 説明 |
|---|---|
| ATIFACILITY | パーティション内 TDQ でサポートされています。端末がキューに関連付けられているかどうかを示す値を返します。CVDA 値は次のとおりです。<br>NOTERMINAL: キューに関連付けられている端末がない<br>TERMINAL: キューに関連付けられている端末がある<br>NOTAPPLIC: キューがパーティション内キューではない |
| ATITERMID | 端末がキューに関連付けられている場合、ターミナルの 4 文字の名前が返されます。それ以外の場合は、空白を返します。端末名は、Sun MTP 宛先管理テーブル (DCT) のパーティション内の宛先画面で定義されます。 |
| ATITRANID | TRIGGERLEVEL 値に達したときに開始されるトランザクション名。 |
| ENABLESTATUS | アプリケーションがキューにアクセス可能かどうかを示します。CVDA 値は次のとおりです。<br>ENABLED DISABLED<br>アプリケーションが DISABLED キューにアクセスできません。キューがまだ OPEN の状態である可能性があります。 |
| INDIRECTNAME | Sun MTP は常に空白を返します。 |

| オプション | 説明 |
| --- | --- |
| IOTYPE | パーティション外 TDQ でサポートされています。キューの I/O タイプを示す値を返します。CVDA 値は次のとおりです。<br>INPUT<br>OUTPUT<br>NOTAPPLIC: キューがパーティション外キューではないことを示します。 |
| NUMITEMS | パーティション内キューのみ。キューにあるレコードの論理数を返します。指定したキューがパーティション内でない場合、-1 が返されます。 |
| OPENSTATUS | パーティション外キューのみ。キューがオープン、クローズ、または中間状態のいずれであるかを示します。次の CVDA 値だけがサポートされています。<br>OPEN CLOSED |
| RECORDFORMAT | パーティション外 TDQ でサポートされています。キューが固定または可変レコードであるかを示す値を返します。CVDA 値は次のとおりです。<br>FIXED: 固定長レコード<br>VARIABLE: 可変長レコード<br>NOTAPPLIC: キューがパーティション外キューではないことを示します。 |
| RECORDLENGTH | 固定長レコードを持つキューのバイトレコード長と、可変レコードを持つキューの最大レコード長を返します。 |
| RECOVSTATUS | パーティション内 TDQ でサポートされています。回復方法を示す値を返します。CVDA 値は次のとおりです。<br>LOGICAL: キューが回復可能であることを示します。<br>NOTRECOVERABLE: キューが回復不可であることを示します。<br>NOTAPPLIC: キューがパーティション内キューではないことを示します。 |
| REMOTENAME | 遠隔 TDQ でサポートされています。遠隔キューの 4 文字の名前を返します。これは、DCT の遠隔の宛先画面の RmtName フィールドの値。キューが定義されていない場合は、空白が返されます。 |
| REMOTESYSTEM | 遠隔 TDQ でサポートされています。遠隔キューがあるシステムの 4 文字の名前を返します。これは、DCT の遠隔の宛先画面の SysID フィールドの値です。端末管理テーブル (TCT) のシステムエントリでも遠隔システムを定義することが必要であることに注意してください。 |
| TRIGGERLEVEL | パーティション内キューのみ。自動トランザクション開始 (ATI) を発生させるために、キューの出力に存在しなければならない要求の数です。指定したキューがパーティション内でない場合、-1 が返されます。 |

| オプション | 説明 |
|---|---|
| TYPE | キューが、パーティション内 (INTRA)、パーティション外 (EXTRA)、遠隔 (REMOTE)、または間接 (INDIRECT) のいずれであるかを示します。 |

次のオプションはサポートされていません。

BLOCKFORMAT EMPTYSTATUS PRINTCONTROL

RDBACK

## INQUIRE TERMINAL

```
INQUIRE TERMINAL(char-4)
[ACQSTATUS(cvda)]
[DEVICE(cvda)]
[NEXTTRANSID (data-area)]
[SERVSTATUS(cvda)]
[TERMSTATUS(cvda)]
[TRANSACTION(char-4)]
[USERID(data-area)]
[USERNAME(data-area)]
```

INQUIRE TERMINAL コマンドは、指定した端末についての情報を返します。端末名はインストールタイプの端末定義と同じで、プリンタおよび端末を含みます。指定した端末が見つからない場合、TERMIDERR が設定され、RESP2 コードが 1 に設定されます。

| オプション | 説明 |
|---|---|
| ACQSTATUS | TERMSTATUS オプションと同じ値を返し、互換性の目的にだけ保持されます。新しいアプリケーションでは TERMSTATUS を使用します。 |
| DEVICE | TCT に格納されている端末タイプを返します。 |
| NEXTTRANSID | この端末の代わりに実行する次のトランザクションの 4 文字の識別子を返します。何も指定されていない場合、空白が返されます。 |
| SERVSTATUS | 端末が使用可能であるかを示す CVDA 値を返します (ローカルの CICS システムから見た場合。端末を所有するシステムとは異なる場合がある)。SERVSTATUS は、端末定義の INSERVICE オプションに対応します。VTAM 端末では、“available” (INSERVICE) は必ずしも端末を取得したことを示すわけではありません。 |

| オプション | 説明 |
|---|---|
| TERMSTATUS | CICS がこの端末によって表される論理ユニットのセッションにあるかどうか示す CVDA 値を返します。CVDA 値は次のとおりです。<br>ACQUIRED CICS: 論理ユニットのセッションにあります。<br>RELEASED CICS: 論理ユニットのセッションにありません。 |
| TRANSACTION | 指定した端末で実行されているトランザクション名を返します。トランザクションが実行されていない場合、引数として TRANSACTION に渡される 4 文字のデータフィールドが空白の状態で返されます。 |
| USERID | この端末にサインオンしたユーザーの 8 文字の識別子を返します。 |
| USERNAME | USERID に対応するオペレータ名の 20 文字の識別子を返します。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| ACCESSMETHOD | ALTPAGEHT | ALTPAGEWD |
| ALTPRINTER | ALTPRTCOPYST | ALTSCRNHT |
| ALTSCRNW | ALTSUFFIX | APLKYBDST |
| APLTEXTST | ATISTATUS | AUDALARMST |
| BACKTRANSST | COLORST | COPYST |
| CREATESESS | DEFPAGEHT | DEFPAGEWD |
| DEFSCRNHT | DEFSCRNWD | DISREQST |
| DUALCASEST | EXITTRACING | EXTENDEDDSST |
| FMHPARMST | FORMFEEDST | GCHARS |
| GCODES | HFORMST | HILIGHTST |
| KATAKANAST | LIGHTPENST | MODENAME |
| MSRCONTROLST | NATLANG | NATURE |
| OBFORMATST | OBOPERIDST | OPERID |
| OUTLINEST | PAGEHT | PAGESTATUS |
| PAGEWD | PARTITIONSST | PRINTADAPTST |
| PRINTER | PROGSYMBOLST | PRTCOPYST |
| QUERYST | RELREQST | REMOTENAME |
| REMOTESYSTEM | SCRNHT | SCRNWD |
| SCREENHEIGHT | SCREENDWIDTH | SECURITY |
| SESSIONTYPE | SIGNONSTATUS | SOSIST |
| TASKID | TCAMCONTROL | TERMMODEL |
| TERMPRIORITY | TEXTKYBDST | TEXTPRINTST |

```
TRACING          TTISTATUS        UCTRANST
USERAREA         USERAREALEN      VALIDATIONST
VFORMST          ZCPTRACING
```

## INQUIRE TRANCLASS

```
INQUIRE TRANCLASS(8-char data-value)
[MAXACTIVE(data-area)|ACTIVE(data-area)]
QUEUED(data-area)
```

INQUIRE TRANCLASS コマンドは、指定したトランザクションクラスについての情報を返します。

| オプション | 説明 |
|---|---|
| ACTIVE | 指定したトランザクションクラスに対して実行されているトランザクション処理プログラムの数。 |
| MAXACTIVE | 指定したクラスに対して割り当てられるトランザクション処理プログラムの最大数。 |
| QUEUED | トランザクションクラスに対してキューに入れられるトランザクションの数。 |

次のオプションはサポートされていません。

PURGETHRESH

クラス名が無効な場合、TCIDERR 条件が返されます。

# INQUIRE TRANSACTION

```
INQUIRE TRANSACTION(4-char data-value)
[DUMPING(cvda)]
[PROGRAM(8-char data-area)]
[REMOTENAME(8-char data-area)]
[REMOTESYSTEM(4-char data-area)]
[SCRNSIZE(cvda)]
[STATUS(cvda)]
[TRANCLASS(8-char data-area)]
[TWASIZE(data-area)]
```

INQUIRE TRANSACTION コマンドは、次に示す例外を除きサポートされます。指定したトランザクションについての情報を返します。

| オプション | 説明 |
|---|---|
| PROGRAM | このトランザクションの開始時に実行される最初のプログラム名。 |
| DUMPING | このトランザクションを実行中のタスクが異常終了した場合にトランザクションダンプをとるかどうかを示す CVDA 値を返します。CVDA 値は次のとおりです。<br>NOTRANDUMP: ダンプをとりません。<br>TRANDUMP: ダンプをとります。 |
| REMOTENAME | 遠隔システムでこのトランザクションが認識される名前 (該当する場合)。 |
| REMOTESYSTEM | トランザクションが定義されている遠隔システムの名前 (該当する場合)。 |
| SCRNSIZE | このトランザクションを実行中のタスクが代替画面サイズを使用するか、デフォルトサイズを使用するかを示す CVDA 値を返します。CVDA 値は次のとおりです。<br>ALTERNATE: 代替画面サイズを使用します。<br>DEFAULT: デフォルト画面サイズを使用します。 |
| STATUS | トランザクションが使用可能であるか示す CVDA 値を返します。CVDA 値は次のとおりです。<br>DISABLED: トランザクションを実行できません。<br>ENABLED: トランザクションを実行できます。 |
| TRANCLASS | トランザクションが属しているトランザクションクラスの名前。 |
| TWASIZE | このトランザクションに対するトランザクション作業領域 (TWA) のバイト単位のサイズを示すフルワードの 2 進フィールドを返します。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| CMDSEC | DTB | DTIMEOUT |
| PRIORITY | PROFILE | PURGEABILITY |
| ROUTING | RESSEC | RTIMEOUT |
| TASKDATAKEY | TASKDATALOC | TRACING |
| TRPROF | | |

## INQUIRE TSQUEUE

```
INQUIRE TSQUEUE(data-value) [LOCATION(cvda)]
[NUMITEMS(data-area)]
[RECOVSTATUS(cvda)]
```

INQUIRE TSQUEUE コマンドは、次に示す例外を除きサポートされます。名前を付けた一時記憶域キューに関する情報を返します。

| オプション | 説明 |
|---|---|
| LOCATION | 一時記憶域キューがある場所を示す値を返します。AUXILIARY の CVDA 値は、キューが固定記憶域にあることを意味します。MAIN は、キューがメイン記憶域にあることを意味します。 |
| NUMITEMS | キューにある項目の数を返します。 |
| RECOVSTATUS | 回復状態を返します。RECOVERABLE の CVDA 値は、回復可能であることを意味します。NOTRECOVERABLE の値は、キューが回復不可であることを意味します。 |
| TSQUEUE | キューの 8 文字の名前。 |

領域の一時記憶域キューをすべて参照するには、INQUIRE TSQUEUE コマンドで次のオプションを使用します。

| オプション | 説明 |
|---|---|
| START | 参照操作を開始します。 |
| AT(data-value) | START オプションとともに使用して、参照操作の開始点となるリソースの名前を指定します。 |
| NEXT | 次のリソースを取得します。 |
| END | 参照操作を終了します。 |

INQUIRE TSQUEUE NEXT コマンドが十分なローカル記憶域を取得できない場合は、NOSTG の応答コードが返されます。この応答コードは、Sun MTP に固有のものです。

次のオプションはサポートされていません。

| | | |
|---|---|---|
| FLENGTH | LASTUSEDINT | MAXITEMLEN |
| MINITEMLEN | TRANSID | SYSID |

## ISSUE ABEND

```
ISSUE ABEND [CONVID(char-4)]
[STATE(cvda)]
```

ISSUE ABEND は、完全にサポートされています。DTP 会話を不正終了させるときに使用します。

| オプション | 説明 |
|---|---|
| CONVID | 不正終了させる会話。 |
| STATE | 現在の会話の状態を取得します。 |

## ISSUE CONFIRMATION

```
ISSUE CONFIRMATION [CONVID(char-4)]
[STATE(cvda)]
```

ISSUE CONFIRMATION は、完全にサポートされています。DTP 会話の SEND CONFIRM に肯定応答を送信するために使用します。

| オプション | 説明 |
|---|---|
| CONVID | 応答の送信先の会話。 |
| STATE | 現在の会話の状態を取得します。 |

## ISSUE DISCONNECT

```
ISSUE DISCONNECT
```

領域と論理ユニットまたは領域と端末のセッションを終了します。

次のオプションはサポートされていません。

SESSION

## ISSUE ERASEAUP

```
ISSUE ERASEAUP
```

ISSUE ERASEAUP は、3270 バッファーにあるすべての無保護フィールドを消去します。3278/3279 タイプの端末でサポートされています。他のタイプ端末ではサポートされません。

次のオプションはサポートされていません。

WAIT

## ISSUE ERROR

```
ISSUE ERROR [CONVID(char-4)]
[STATE(cvda)]
```

ISSUE ERROR は、完全にサポートされています。DTP 会話のパートナーにエラーを通知するために使用します。

| オプション | 説明 |
|---|---|
| CONVID | コマンドに関連する会話。 |
| STATE | 現在の会話の状態を取得します。 |

## ISSUE PRINT

```
ISSUE PRINT
```

ISSUE PRINT では、トランザクションでシステムプリンタまたは 3270 プリンタから画面を印刷できます。ISSUE PRINT は、完全にサポートされています。

## ISSUE SIGNAL

```
ISSUE SIGNAL [CONVID(char-4)]
[STATE(cvda)]
```

ISSUE SIGNAL は、DTP 会話のパートナーのトランザクションからの方向の変更を要求するために使用します。

| オプション | 説明 |
|---|---|
| CONVID | コマンドに関連する会話。 |
| STATE | 現在の会話の状態を取得します。 |

次のオプションはサポートされていません。

SESSION

## JOURNAL

```
JOURNAL JFILEID(halfword)
JTYPEID(char-2)
FROM(data-area)
[LENGTH(halfword)]
[REQID(fullword)]
[PREFIX(data-value) [PFXLENG(halfword)]]
[NOSUSPEND]
```

JOURNAL は、JCT で定義されている外部の VSAM 以外のファイルにレコードを書き込みます。Sun MTP のすべての JOURNAL コマンドは同期しながら書き込まれます。JCT については、『Sun Mainframe Transaction Processing 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。

次のオプションはサポートされていません。

STARTIO WAIT

# LINK

```
LINK PROGRAM(char-8)
[COMMAREA(data-area) [LENGTH(halfword)]
[DATALENGTH(halfword)]]
[SYSID(char-4)][SYNCONRETURN]
[TRANSID(char-4)]
```

LINK は、順次、制御を各アプリケーションプログラムに一時的に渡します。

| オプション | 説明 |
|---|---|
| COMMAREA | 呼び出されたプログラムに通信領域として渡されるデータ領域。呼び出されたプログラムは、呼び出し元プログラムと同じ記憶領域の位置にある通信領域にアクセスします。その通信領域は他の記憶領域にコピーできません。 |
| DATALENGTH | リンクされたプログラムにデータが渡される COMMAREA の開始からの連続スペース。 |
| LENGTH | COMMAREA オプションで指定されたデータ領域の長さを示す、2 進ハーフワードまたはリテラル値。COMMAREA をコード化する場合、このオプションは必須です。 |
| PROGRAM | SYSID がコード化されていない場合、PPT のエントリに一致する必要があります。PPT については、『Sun Mainframe Transaction Processing 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。 |
| SYNCONRETURN | SYSID をコード化する場合、サーバープログラムの完了時に遠隔システムが同期点を取るように指定します。これを省略すると、デフォルトの同期レベルはリンクの同期レベルと同じになります。 |
| SYSID | 指定されている場合、TCT–System Entries 画面のエントリに一致する必要があります。DPL については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。SYSID および INPUTMSG は、相互に排他的です。 |
| TRANSID | 遠隔システムが接続してサーバープログラムを実行するトランザクションの名前。指定されていない場合、遠隔システムはデフォルトで CSMI または CVMI に接続します。 |

次のオプションはサポートされていません。

```
INPUTMSG     INPUTMSGLEN
```

## LOAD

```
LOAD PROGRAM(char-8)
[SET(pointer-ref)]
[LENGTH(halfword-data-area)|
FLENGTH(fullword-data-area)]
[ENTRY(pointer-ref)]
[HOLD]
```

LOAD は、プログラム、テーブル、またはマップを共有記憶域にロードします。コマンドから返されるアドレスは、プログラム、テーブル、またはマップを参照するすべてのトランザクションに対して同じであることが保証されています。プログラム、テーブル、およびマップは、拡張子を付けてはならず、KIXPROGS 環境変数で指定されているディレクトリのいずれかに存在する必要があります。

| オプション | 説明 |
|---|---|
| FLENGTH | ロードされるプログラム、テーブル、またはマップの長さ。ロードされるプログラムの長さが 32K バイトよりも長い場合に使用します。 |
| HOLD | LOAD コマンドを発行するタスクが終了するとき、ロードされてまだ存在しているプログラム、テーブル、またはマップを削除しません。 |
| LENGTH | ロードされるプログラム、テーブル、またはマップの長さ。 |
| SET | プログラム、テーブル、またはマップがロードされるアドレスに設定されるポインタ参照。 |

LOAD は、次のオプションを以下のようにサポートします。

| オプション | 説明 |
|---|---|
| ENTRY | 158 ページの「共有ライブラリでの LOAD PROGRAM ENTRY の使用法」に示されているとおり、PL/I プログラムだけサポートします。 |
| PROGRAM | タイプ A または L の PPT のエントリに一致する必要があります。PPT については、『Sun Mainframe Transaction Processing 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。 |

## POP HANDLE

```
POP HANDLE
```

POP HANDLE は、完全にサポートされています。PUSH HANDLE で中断された次の HANDLE コマンドを復元します。

| | | |
|---|---|---|
| HANDLE ABEND | HANDLE AID | HANDLE CONDITION |
| IGNORE | IGNORE CONDITION | |

## POST

```
POST [INTERVAL(packed-7)|TIME(packed-7)|
AFTER{[HOURS(hh)][MINUTES(mm)]
[SECONDS(ss)]}|AT{[HOURS(hh)]
[MINUTES(mm)][SECONDS(ss)]}]
SET(pointer-ref)
[REQID(char-8)]
```

POST は、完全にサポートされています。指定した時間間隔を経過すると、タイマーイベント制御領域が送信されるように要求します。

| オプション | 説明 |
|---|---|
| AFTER | 期限切れまでの時間。 |
| AT | 期限切れ時刻。 |
| HOURS(hh) | 0 ～ 99 のフルワードの 2 進値。 |
| INTERVAL | POST コマンドが発行されてから期限切れになるまでの経過時間。 |
| MINUTES(mins) | 0 ～ 59 または 0 ～ 5999 のフルワードの 2 進値。 |
| REQID | コマンドを識別する一意の名前。一時記憶域識別子として使用されます。 |
| SECONDS(ss) | 0 ～ 59 または 0 ～ 359,999 のフルワードの 2 進値。 |
| SET | CICS によって生成されるタイマーイベント制御領域のアドレスにポインタ参照を設定します。 |
| TIME | 期限切れ時刻。 |

CICS による処理に対して POST コマンドが有効でない場合、INVREQ 条件が発生することがあります。失敗時には、EIBRESP2 の値によって失敗の理由が示されます。

| 条件 | RESP2 | 理由 |
|---|---|---|
| INVREQ | 4 | 時間が範囲外。 |
| INVREQ | 5 | 分が範囲外。 |
| INVREQ | 6 | 秒が範囲外。 |

デフォルトのアクションでは、タスクを異常終了します。

## PURGE MESSAGE

```
PURGE MESSAGE
```

PURGE MESSAGE は、論理メッセージの構築を終了します。論理メッセージの一部に割り当てられた領域で、ここまで構築されたものは削除されます。

## PUSH HANDLE

```
PUSH HANDLE
```

PUSH HANDLE は、完全にサポートされています。このコマンドは、HANDLE コマンドを中断します。このコマンドは、POP HANDLE コマンドを使用して復元できます。

次のコマンドを中断できます。

```
HANDLE ABEND    HANDLE AID         HANDLE CONDITION
IGNORE          IGNORE CONDITION
```

# QUERY SECURITY

```
QUERY SECURITY
{RESTYPE(data-value) | RESCLASS(data-value)
  [RESIDLENGTH(data-value)]}
RESID(data-value)
[LOGMESSAGE(cvda)] [READ(cvda)] [UPDATE(cvda)] [CONTROL(cvda)]
[ALTER(cvda)]
```

QUERY SECURITY は、完全にサポートされています。このコマンドを使うと、ユーザーが外部セキュリティーマネージャー (ESM) で定義されているリソースへのアクセス権を持つかどうかを判断できます。

| オプション | 説明 |
|---|---|
| ALTER(cvda) | ユーザーがリソースの ALTER 権限を持つかどうかを照会します。返される CVDA 値は、ALTERABLE および NOTALTERABLE。ALTER 権限は、Sun MTPの CreateAcess を Sun MSFの Manage アクセスにマップします。 |
| CONTROL(cvda) | ユーザーがリソースの CONTROL 権限を持つかどうかを照会します。返される CVDA 値は、CTRLABLE および NOTCTRLABLE。CONTROL 権限は、Sun MTPの CreateAcess を Sun MSFの Manage アクセスにマップします。 |
| LOGMESSAGE(cvda) | セキュリティー違反メッセージを抑制できます。CVDA 値は、LOG (デフォルト) または NOLOG (メッセージを表示しない)。 |
| READ(cvda) | ユーザーがリソースの READ 権限を持つかどうかを照会します。返される CVDA 値は、READABLE および NOTREADABLE。READ 権限は、Sun MTPの ReadAccess を Sun MSFの Read アクセスにマップします。 |
| RESCLASS(data-value) | ESM で定義されている 8 文字のユーザー定義リソースクラスを指定します。 |
| RESID(data-value) | ユーザーのアクセスを照会するリソースを指定します。値は、CICS タイプのリソースでは 1〜12 文字の文字列、ユーザー定義のリソースでは 1〜246 文字。<br>RESID は、ユーザー定義のリソースを示しますが、RESTYPE(*SPCOMMAND*) の場合は指定されます。 |
| RESIDLENGTH(data-value) | フルワードバイナリとして、RESID の長さを指定します。このパラメータは、RESCLASS オプションを指定する場合にのみ使用します。 |

| オプション | 説明 |
|---|---|
| RESTYPE(data-value) | ユーザーのアクセスを照会するリソースの種類を指定します。1〜12 文字の値。RESTYPE の有効な値は次のとおりです。<br>• FILE。これは、Sun MSF KIX_FILE リソースタイプにマップします。<br>• JOURNALNAME。これは、Sun MSF KIX_JOURNAL リソースタイプにマップします。<br>• PROGRAM。これは、Sun MSF KIX_PROGRAM リソースタイプにマップします。<br>• SPCOMMAND。これは、Sun MSF KIX_COMMANDS リソースタイプにマップします。<br>• TDQUEUE。これは、Sun MSF KIX_TDQUEUE リソースタイプにマップします。<br>• TRANSACTION。これは、Sun MSF KIX_START_TRANS リソースタイプにマップします。<br>• TRANSATTACH。これは、Sun MSF KIX_ATTACH_TRANS リソースタイプにマップします。<br>• TSQUEUE。これは、Sun MSF KIX_TSQUEUE リソースタイプにマップします。 |
| UPDATE(cvda) | ユーザーがリソースの UPDATE 権限を持つかどうかを照会します。返される CVDA 値は、UPDATABLE および NOTUPDATABLE。UPDATE 権限は、Sun MTP の WriteAccess を Sun MSFの Write アクセスにマップします。 |

確認される実際のリソースは、RESCLASS または RESTYPE が指定されているかどうか、接頭辞付けが有効であるかどうか (領域設定ファイルの KIXSECPREFIX=YES) によって異なります。

- RESCLASS が指定されている場合は、確認されるリソースが、接頭辞付けがオンかオフかにかかわらず、常に RESID 値となります。
- RESTYPE が指定されていて、接頭辞付けがオフの場合は、確認されるリソースは、指定された RESID 値となります。接頭辞付けがオンである場合は、確認されるリソースは、領域を開始したユーザーの UNIX ユーザー ID が接頭辞に付いた RESID 値です。

失敗時には、RESP2 によって失敗の理由が示されます。

| 条件 | RESP2 | 理由 |
|---|---|---|
| INVREQ | 7 | CVDA 値は、LOGMESSAGEでは有効ではありません。 |
| INVREQ | 9 | RESID 値は、無効であるか、空白です。 |
| INVREQ | 10 | ESM が有効でない、または存在しません。 |

| 条件 | RESP2 | 理由 |
|---|---|---|
| LENGERR | 6 | RESIDLENGTH が無効です。1～246 の範囲が必要です。 |
| NOTFND | 1 | RESID が無効です。 |
| NOTFND | 2 | RESTYPE が無効です。 |
| NOTFND | 3 | RESTYPE(*SPCOMMAND*) の RESID が無効です。 |
| NOTFND | 5 | RESCLASS が ESM で定義されていません。 |

## READ

```
READ DATASET(dataset)|FILE(dataset)
SET(pointer-ref)|INTO(data-area)
[LENGTH(halfword-data-area)]
RIDFLD(data-area)
[KEYLENGTH(halfword) [GENERIC]]
[SYSID(char-4)]
[RBA|RRN]
[GTEQ|EQUAL]
[UPDATE]
```

READ は、この節の最後に示す例外を除いてサポートされています。VSAM データセットに対して、データセットからレコードを読み取ります。ISAM および DAM データセットはサポートされません。

| オプション | 説明 |
|---|---|
| DATASET | SYSID がコード化されていない場合、FCT のエントリに一致する必要があります。FCT については、『Sun Mainframe Transaction Processing 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。 |
| EQUAL | 検索は、RIDFLD オプションで指定されているのと同じキーを持つレコードの場合のみ一致します。 |
| FILE | アクセスするファイルの名前。これは FCT のデータセット名です。 |
| GENERIC | 検索キーは、KEYLENGTH オプションで指定された長さの総称キー。 |
| GTEQ | RIDFLD オプションで指定されているのと同じキーを持つレコードの検索に失敗した場合、それより大きいキーを持つ最初のレコードが取り出されます。 |
| INTO | データセットから取り出したレコードが書き込まれるデータ領域。 |
| KEYLENGTH | RIDFLD オプションで指定されたキーの長さ。 |
| LENGTH | レコードが入れられるデータ領域の長さ。READ コマンドが完了すると、LENGTH は実際のレコードの長さを示します。 |

| オプション | 説明 |
|---|---|
| RBA | RIDFLD オプションで指定されたレコード識別フィールドが相対バイトアドレスを示します。 |
| RIDFLD | レコードを特定するフィールド。 |
| RRN | RIDFLD オプションで指定されたレコード識別フィールドが相対レコード番号を示します。 |
| SET | CICS はレコードが読み込まれるバッファーを供給し、取り出したレコードのアドレスが含まれるポインタ参照を指定します。 |
| SYSID | 指定されている場合、TCT– System Entries 画面のエントリに一致する必要があります。機能シップについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |
| UPDATE | 更新または削除用のレコードを取得します。省略すると、読み取り専用操作になります。 |

次のオプションはサポートされていません。

```
DEBKEY        DEBREC        SEGSET
SEGSETALL
```

## READNEXT

```
READNEXT DATASET(dataset)|FILE(dataset)
SET(pointer-ref)|INTO(data-area)
[LENGTH(halfword-data-area)]
RIDFLD(data-area)
[KEYLENGTH(halfword)]
[REQID(halfword)]
[SYSID(char-4)]
[RBA|RRN]
```

READNEXT は、VSAM データセットで完全にサポートされています。データセットから順番にレコードを読み取ります。ISAM および DAM データセットはサポートされません。

| オプション | 説明 |
|---|---|
| DATASET | SYSID がコード化されていない場合、FCT のエントリに一致する必要があります。FCT については、『Sun Mainframe Transaction Processing 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。 |
| FILE | アクセスするファイルの名前。これは FCT のデータセット名。 |
| INTO | データセットから取り出したレコードが書き込まれるデータ領域。 |
| KEYLENGTH | RIDFLD オプションで指定されたキーの長さ。 |
| LENGTH | レコードが入れられるデータ領域の長さ。READNEXT コマンドが完了すると、LENGTH は実際のレコードの長さを示します。 |
| RBA | RIDFLD オプションで指定されたレコード識別フィールドが相対バイトアドレスを示します。 |
| REQID | ブラウズの一意の要求識別子。データセットで複数のブラウズ操作を制御する場合に使用します。 |
| RIDFLD | レコードを特定するフィールド。 |
| RRN | RIDFLD オプションで指定されたレコード識別フィールドが相対レコード番号を示します。 |
| SET | CICS はレコードが読み込まれるバッファーを供給し、取り出したレコードのアドレスが含まれるポインタ参照を指定します。 |
| SYSID | 指定されている場合、TCT–System Entries 画面のエントリに一致する必要があります。機能シップについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |

次のオプションはサポートされていません。

SEGSET　　SEGSETALL

## READPREV

```
READPREV DATASET(dataset)|FILE(dataset)
SET(pointer-ref)|INTO(data-area)
[LENGTH(halfword-data-area)]
RIDFLD(data-area)
[KEYLENGTH(halfword)]
[REQID(halfword)]
[SYSID(char-4)]
[RBA|RRN]
```

READPREV は、VSAM データセットで完全にサポートされています。データセットから逆順にレコードを読み取ります。ISAM および DAM データセットはサポートされません。

| オプション | 説明 |
|---|---|
| DATASET | SYSID がコード化されていない場合、FCT のエントリに一致する必要があります。FCT については、『Sun Mainframe Transaction Processing 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。 |
| FILE | アクセスするファイルの名前。これは FCT のデータセット名 |
| INTO | データセットから取り出したレコードが書き込まれるデータ領域。 |
| KEYLENGTH | RIDFLD オプションで指定されたキーの長さ。 |
| LENGTH | レコードが入れられるデータ領域の長さ。READPREV コマンドが完了すると、LENGTH は実際のレコードの長さを示します。 |
| RBA | RIDFLD オプションで指定されたレコード識別フィールドが相対バイトアドレスを示します。 |
| REQID | ブラウズの一意の要求識別子。データセットで複数のブラウズ操作を制御する場合に使用します。 |
| RIDFLD | レコードを特定するフィールド。 |
| RRN | RIDFLD オプションで指定されたレコード識別フィールドが相対レコード番号を示します。 |
| SET | CICS はレコードが読み込まれるバッファーを供給し、取り出したレコードのアドレスが含まれるポインタ参照を指定します。 |
| SYSID | 指定されている場合、TCT– System Entries 画面のエントリに一致する必要があります。機能シップについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |

次のオプションはサポートされていません。

```
SEGSET        SEGSETALL
```

## READQ TD

```
READQ TD QUEUE(char-4)
SET(pointer-ref)|INTO(data-area)
[LENGTH(halfword-data-area)]
[SYSID(char-4)]
[NOSUSPEND]
```

READQ TD は、完全にサポートされています。QUEUE オプションで指定される一時データキューからデータを読み取ります。

| オプション | 説明 |
|---|---|
| INTO | キューから読み取られたデータが配置されるユーザーデータ領域。 |
| LENGTH | INTO オプションを指定した場合、LENGTH は、プログラムが受け入れるデータの最大長を指定するデータ領域にする必要があります。 |
| NOSUSPEND | ローカルシステムに影響はなく、機能シップ要求で渡されます。 |
| QUEUE | SYSID がコード化されていない場合、DCT のエントリに一致する必要があります。DCT については、『Sun Mainframe Transaction Processing 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 | キューから読み取ったデータのアドレスにポインタ参照を設定します。 |
| SYSID | 指定されている場合、TCT–System Entries 画面のエントリに一致する必要があります。機能シップについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |

# READQ TS

```
READQ TS QUEUE(char-8)
SET(pointer-ref)|INTO(data-area)
[LENGTH(halfword-data-area)]
[NUMITEMS(halfword-data-area)]
[ITEM(halfword)|NEXT]
[SYSID(char-4)]
```

READQ TS は、完全にサポートされています。QUEUE オプションで指定されている一時記憶域キューからデータを取り出します。

| オプション | 説明 |
|---|---|
| INTO | キューから読み取られたデータが配置されるユーザーデータ領域。 |
| ITEM | キューから取り出される論理レコードの項目番号。 |
| LENGTH | INTO オプションを指定した場合、LENGTH は、プログラムが受け入れるデータの最大長を指定するデータ領域にする必要があります。 |
| NEXT | 次の論理レコードを取り出します。 |
| NUMITEMS | キュー内の項目数が格納されるフィールド。 |
| QUEUE | SYSID がコード化されていない場合、キューが回復可能または遠隔であるときは、TST のエントリと一致する必要があります。TST については、『Sun Mainframe Transaction Processing 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 | キューから読み取ったデータのアドレスにポインタ参照を設定します。 |
| SYSID | 指定されている場合、TCT–System Entries 画面のエントリに一致する必要があります。機能シップについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |

# RECEIVE (APPC)

```
RECEIVE INTO(data-area) | SET(pointer-ref)
LENGTH(halfword-data-area) | FLENGTH(fullword-data-area)
[MAXLENGTH [(halfword)] | MAXFLENGTH [(fullword)]]
[NOTRUNCATE]
[CONVID(char-4)]
[STATE(cvda)]
```

RECEIVE は、端末または DTP 会話からデータを読み取ります。3278/3279 タイプの端末で完全にサポートされています。他のタイプの端末はサポートされていません。

| オプション | 説明 |
|---|---|
| CONVID | コマンドに関連する会話を識別します。 |
| INTO | 現在の会話のパートナーに接続しているアプリケーションプログラムからのデータを受け取るための、アプリケーションのターゲットデータ領域。INTO データ領域の長さは、LENGTH、FLENGTH、MAXLENGTH、および MAXFLENGTH オプションで指定されている最大の長さに等しいか、それより長い必要があります。 |
| FLENGTH | LENGTH の代替フルワード。 |
| LENGTH | 受け取るデータの長さ。INTO オプションを指定して MAXLENGTH オプションを省略する場合、この引数はプログラムが受け入れる最大長を指定するデータ領域になります。指定する値がゼロより小さい場合、ゼロと見なされます。SET オプションを指定する場合、その引数がデータ領域になります。データを受け取ると、そのデータ領域はデータの長さに設定されます。最大長は、32K バイト。 |
| MAXFLENGTH | MAXLENGTH の代替フルワード。 |
| MAXLENGTH | RECEIVE コマンドに応答して返されるデータの最大量。INTO が指定されていると、MAXLENGTH は LENGTH の入力を無視します。<br>SET が指定されていると、プログラムが 1 度に受け取るデータの量を MAXLENGTH により制限できます。データ長が指定の値を超えていて、NOTRUNCATE オプションが存在しない場合、データは指定の値に切り捨てられ、LENGERR 条件が発生します。LENGTH オプションで指定したデータ領域は、データのもとの長さに設定されます。<br>データの長さが指定の値を超えていて、NOTRUNCATE オプションが存在する場合、残りのデータを保持して、そのデータを後続の RECEIVE コマンドのために使用します。<br>このオプションを省略すると、LENGTH オプションで示される値が使用されます。 |
| NOTRUNCATE | 利用できるデータが要求された長さを超える場合、データの残りを破棄しないで、後続の RECEIVE コマンドによって取り出されるように保持します。 |

| オプション | 説明 |
|---|---|
| SET | パートナートランザクションから受け取ったデータのアドレスにポインタ参照を設定します。 |
| STATE | 現在の会話の状態を取得します。 |

# RECEIVE (LUTYPE2/LUTYPE3)

```
RECEIVE INTO(data-area) | SET(pointer-ref)
LENGTH(halfword-data-area) | FLENGTH(fullword-data-area)
[MAXLENGTH [(halfword)] | MAXFLENGTH [(fullword)]]
[NOTRUNCATE]
[ASIS]
[BUFFER]
```

RECEIVE は、端末または DTP 会話からデータを読み取ります。3278/3279 タイプの端末で完全にサポートされています。他のタイプの端末はサポートされていません。

| オプション | 説明 |
|---|---|
| ASIS | 3270 入力データストリームの小文字を大文字に変換しません。これにより、現在のタスクは大文字と小文字データの両方を含むメッセージを受け取ることができます。 |
| BUFFER | 3270 バッファーの内容を読み取ります。バッファー位置 1 から開始して、バッファーのすべての内容を読み取るまで継続します。すべての文字および属性シーケンス (NULL を含む) が、3270 バッファーに現れるのと同じ順序で入力データストリームに現れます。 |
| INTO | 論理ユニットから読み取ったデータを受け取るフィールド。 |
| FLENGTH | LENGTH の代替フルワード。 |
| LENGTH | 転送されるデータの長さ。INTO オプションを指定して MAXLENGTH オプションを省略する場合、この引数はプログラムが受け入れる最大長を指定するデータ領域になります。指定する値がゼロより小さい場合、ゼロと見なされます。データ長が指定の値を超えていて、NOTRUNCATE オプションが存在しない場合、データは指定の値に切り捨てられ、LENGERR 条件が発生します。データを受け取ると、データ領域はもとのデータの長さに設定されます。最大長は、32K バイト。<br>SET オプションを指定する場合、その引数がデータ領域になります。データを受け取ると、そのデータ領域はデータの長さに設定されます。 |
| MAXFLENGTH | MAXLENGTH の代替フルワード。 |

| オプション | 説明 |
|---|---|
| MAXLENGTH | 復元するデータの最大量。INTO が指定されていると、MAXLENGTH は LENGTH の入力を無視します。SET が指定されていると、プログラムが 1 度に受け取るデータの量を MAXLENGTH により制限できます。データ長が指定の値を超えていて、NOTRUNCATE オプションが存在しない場合、データは指定の値に切り捨てられ、LENGERR 条件が発生します。LENGTH オプションで指定したデータ領域は、データのもとの長さに設定されます。<br>データの長さが指定の値を超えていて、NOTRUNCATE オプションが存在する場合、残りのデータを保持して、そのデータを後続の RECEIVE コマンドのために使用します。LENGTH オプションで指定したデータ領域は、返されたデータの長さに設定されます。<br>このオプションを省略すると、LENGTH オプションで示される値が使用されます。 |
| NOTRUNCATE | 利用できるデータが要求された長さを超える場合、データの残りを破棄しないで、後続の RECEIVE コマンドによって取り出されるように保持します。 |
| SET | 端末または論理ユニットから読み取ったデータのアドレスにポインタ参照を設定します。 |

## RECEIVE MAP

```
RECEIVE MAP (char-7)
[MAPSET(char-7)]
[SET(pointer) | INTO(data-area)]
[FROM(data-area) LENGTH(halfword) | TERMINAL [ASIS]]
```

RECEIVE MAP は、端末からのデータをデータ領域に割り当てます。3278/3279 タイプの端末でサポートされています。他のタイプの端末はサポートされていません。

| オプション | 説明 |
|---|---|
| ASIS | 3270 入力データストリームの小文字を大文字に変換しません。 |
| FROM | 割り当てられるデータを含むデータ領域。 |
| INTO | 割り当てられるデータを書き込むデータ領域。 |
| LENGTH | フォーマットされるデータのハーフワード 2 進値による長さ。 |
| MAPSET | 接尾辞のないマップセットの名前。 |
| SET | 割り当てられるデータの 12 バイト接頭辞のアドレスにポインタを設定します。 |
| TERMINAL | トランザクションを開始した端末からの入力データを読み取ります。 |

次のオプションはサポートされていません。

INPARTN

## RELEASE

```
RELEASE PROGRAM(char-8)
```

RELEASE は、完全にサポートされています。主記憶域から以前にロードされたプログラム、テーブル、またはマップを削除します。

| オプション | 説明 |
|---|---|
| PROGRAM | プログラム名を識別します。PPT のエントリに一致する必要があります。PPT については、『Sun Mainframe Transaction Processing 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。 |

## RESETBR

```
RESETBR DATASET(dataset) | FILE(dataset)
RIDFLD(data-area)
[KEYLENGTH(halfword) [GENERIC]]
[REQID(halfword)]
[SYSID(char-4)]
[GTEQ | EQUAL]
[RBA | RRN]
```

RESETBR は、VSAM データセットで完全にサポートされています。ブラウズ操作を再起動するデータセットのレコードを指定します。ISAM および DAM データセットはサポートされません。

| オプション | 説明 |
|---|---|
| DATASET | SYSID がコード化されていない場合、FCT のエントリに一致する必要があります。FCT については、『Sun Mainframe Transaction Processing 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。 |
| EQUAL | 検索は、RIDFLD オプションで指定されているのと同じキーを持つレコードの場合のみ一致します。 |
| FILE | アクセスするデータセットの名前。 |
| GENERIC | 検索キーは、KEYLENGTH オプションで指定された長さの総称キー。 |
| GTEQ | RIDFLD オプションで指定されているのと同じキーを持つレコードの検索に失敗した場合、それより大きいキーを持つ最初のレコードが取り出されます。 |

| オプション | 説明 |
|---|---|
| KEYLENGTH | RIDFLD オプションで指定されたキーの長さ。 |
| RBA | RIDFLD オプションで指定されたレコード識別フィールドが相対バイトアドレスを示します。 |
| REQID | ブラウズの一意の要求識別子。データセットで複数のブラウズ操作を制御する場合に使用します。 |
| RIDFLD | レコードを特定するフィールド。 |
| RRN | RIDFLD オプションで指定されたレコード識別フィールドが相対レコード番号を示します。 |
| SYSID | 指定されている場合、TCT– System Entries 画面のエントリに一致する必要があります。詳細については、『Sun Mainframe Transaction Processing 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。 |

## RETRIEVE

```
RETRIEVE [INTO(data-area) | SET(pointer-ref)]
[LENGTH(halfword)]
[RTRANSID(char-4)]
[RTERMID(char-4)]
[QUEUE(char-48)]
```

RETRIEVE は、前の START コマンドによって開始されたタスクで使用されます。START コマンドによって格納されたデータを取り出します。

| オプション | 説明 |
|---|---|
| INTO | 取り出したデータが書き込まれるユーザーデータ領域。 |
| LENGTH | 取り出されるデータの長さ。 |
| QUEUE | RETRIEVE コマンドを発行するトランザクションによってアクセスされるキューの 8 文字の名前。 |
| RTRANSID | 引き続き実行される START コマンドの TRANSID オプションで使用される 4 文字の領域。 |
| RTERMID | 引き続き実行される START コマンドの TERMID オプションで使用される 4 文字の領域。 |
| SET | 取り出されるデータのアドレスにポインタ参照を設定します。 |

次のオプションはサポートされていません。

WAIT

## RETURN

```
RETURN [TRANSID(char-4) [COMMAREA(data-area)
[LENGTH(halfword)]] [IMMEDIATE]]
[INPUTMSG(data-area) [INPUTMSGLEN(data-value)]]
```

RETURN は、あるアプリケーションプログラムから、そのプログラムを呼び出したプログラムまたはシステムに制御を返します。

| オプション | 説明 |
|---|---|
| COMMAREA | 制御を受け取る次のプログラムで使用できるようになる通信領域。 |
| IMMEDIATE | TRANSID オプションで指定されたトランザクションが、次のトランザクションとして確実に接続されるようにします。 |
| INPUTMSG | TRANSID オプションで指定されたトランザクションにデータが渡されます。 |
| INPUTMSGLEN | INPUTMSG のデータの長さを指定します。 |
| LENGTH | COMMAREA のバイト単位の長さ。 |
| TRANSID | 指定されている場合、PCT のエントリに一致する必要があります。詳細については、『Sun Mainframe Transaction Processing 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。 |

**注 –** INPUTMSG および INPUTMSGLEN オプションは、TRANSID および IMMEDIATE オプションも RETURN コマンドで指定されている場合にのみサポートされます。これらのオプションのほかの使用方法はサポートされていません。

## REWRITE

```
REWRITE DATASET(dataset) | FILE(dataset)
FROM(data-area)
[LENGTH(halfword)]
[SYSID(char-4)]
```

REWRITE は、VSAM データセットで完全にサポートされています。データセットのレコードを更新します。ISAM および DAM データセットはサポートされません。

| オプション | 説明 |
|---|---|
| DATASET | SYSID がコード化されていない場合、FCT のエントリに一致する必要があります。詳細については、『Sun Mainframe Transaction Processing 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。 |
| FILE | アクセスされるデータセットの名前。 |
| FROM | このファイルが参照するデータセットに書き込まれるレコード。 |
| LENGTH | 書き込むレコードの実際の長さ。 |
| SYSID | 指定されている場合、TCT–System Entries 画面のエントリに一致する必要があります。機能シップについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |

## SEND (APPC マップ)

```
SEND [FROM(data-area)
{LENGTH(halfword) | FLENGTH(fullword)}]
[CONVID(char-4)]
[INVITE | LAST]
[CONFIRM | WAIT]
[STATE(cvda)]
```

SEND は、端末または DTP 会話にデータを書き込みます。3278/3279 タイプの端末およびシステムプリンタでサポートされます。他のタイプの端末はサポートされていません。

| オプション | 説明 |
|---|---|
| CONFIRM | 同期レベル 1 または 2 の会話を使用するアプリケーションは、遠隔アプリケーションからの応答が必要であることを示します。 |
| CONVID | コマンドに関連する会話を識別します。 |
| FLENGTH | LENGTH の代替フルワード。 |

| オプション | 説明 |
|---|---|
| FROM | パートナートランザクションに送信されるデータ。INVITE、WAIT、CONFIRM、または LAST を使用する場合、このオプションは省略できます。 |
| INVITE | アプリケーションプログラムが、接続されている APPC システムのプロセスにすでに送信されたデータに制御データを追加できるようにします。CONFIRM または WAIT をともに指定していない場合、後続の WAIT または SYNCPOINT コマンドが実行されるまで、制御データは転送されません。 |
| LAST | トランザクションの最後の SEND コマンド。 |
| LENGTH | 送信するデータの長さ。最大長は、32K バイト。 |
| STATE | 現在の会話の状態を取得します。 |
| WAIT | 会話で現在までに送信されたすべてのデータおよびインジケータが、パートナートランザクションに確実にフラッシュされるようにします。WAIT が指定されていない場合、コマンドの処理が開始すると、制御はアプリケーションプログラムに返されます。 |

# SEND (SCS/LUTYPE1)

```
SEND FROM(data-area)
{LENGTH(halfword) | FLENGTH(fullword)}
[WAIT]
```

SEND は、データを LU タイプ 1 のデバイスに書き込みます。これは、TN3287 タイプのプリンタでサポートされています。他のタイプの端末はサポートされていません。

| オプション | 説明 |
|---|---|
| FLENGTH | LENGTH の代替フルワード。 |
| FROM | データを LU Type 1 のプリンタに書き込みます。 |
| LENGTH | 書き込むデータの長さ。最大長は、32K バイト。 |
| WAIT | 後続処理を試行する前に、コマンドの処理を完了します。指定されていない場合、コマンドの処理が開始すると、制御はアプリケーションプログラムに返されます。 |

次のオプションはサポートされていません。

| | |
|---|---|
| CNOTCOMPL | DEFRESP |
| FMH | INVITE |
| LAST | STRFIELD |

# SEND (LUTYPE2/LUTYPE3)

```
SEND FROM(data-area)
{LENGTH(halfword) | FLENGTH(fullword)}
[STRFIELD | [ERASE] [CTLCHAR(char-1)]]
[INVITE | LAST]
[WAIT]
```

SEND は、端末または DTP 会話にデータを書き込みます。3278/3279 タイプの端末およびシステムプリンタでサポートされます。他のタイプの端末はサポートされていません。

| オプション | 説明 |
|---|---|
| CTLCHAR | 3270 の SEND コマンドを制御する 1 バイトの書き込み制御文字。COBOL ユーザーは、この文字を含むデータ領域を指定する必要があります。このオプションを省略すると、変更されたすべてのデータタグはゼロにリセットされ、キーボードがリセットされます。 |
| ERASE | 書き込みが発生する前に、バッファーまたは画面を消去して画面左上にカーソルを戻します。 |
| FLENGTH | LENGTH の代替フルワード。 |
| FROM | 論理ユニットに書き込まれるデータ。 |
| INVITE | この機能に対して実行される次の端末制御コマンドは RECEIVE。 |
| LAST | トランザクションに対する最後の出力操作、およびこれによる括弧の終わり。 |
| LENGTH | 書き込むデータの長さ。最大長は、32K バイト。 |
| STRFIELD | FROM オプションで指定されたデータ領域に構造化フィールドが含まれます。 |
| WAIT | 後続処理を試行する前に、コマンドの処理を完了します。指定されていない場合、コマンドの処理が開始すると、制御はアプリケーションプログラムに返されます。 |

次のオプションはサポートされていません。

DEFRESP

# SEND CONTROL

```
SEND CONTROL [CURSOR(halfword)]
[ERASE | ERASEAUP]
[PRINT]
[FREEKB]
[ALARM]
[FRSET]
[ACCUM]
[SET(pointer) | PAGING | TERMINAL]
[REQID(char-2)]
[L80 | HONEOM]
```

SEND CONTROL は、デバイス制御を端末に送信します。3278/3279 タイプの端末でサポートされています。他のタイプの端末はサポートされていません。

| オプション | 説明 |
|---|---|
| ACCUM | 論理メッセージの構築に使用されるコマンドの 1 つ。 |
| ALARM | 3270 警告音を有効化します。 |
| CURSOR | SEND CONTROL コマンドの完了時に戻されるカーソルの位置。 |
| ERASE | 画面プリンタバッファー、またはパーティションを消去し、画面の左上にカーソルを戻します。 |
| ERASEAUP | パーティションまたは画面全体にあるすべての無保護の文字位置を削除します。 |
| FREEKB | 3270 キーボードのロックを解除します。 |
| FRSET | 現在 3270 バッファーにある全フィールドの変更されたデータタグを変更前の状態にリセットします。 |
| HONEOM | デフォルトのプリンタ行の長さを使用します。 |
| L80 | 3270 プリンタに対して 80 文字の行の長さを指定します。 |
| PAGING | 出力データを端末にすぐに送信しないで一時記憶域に配置し、端末ユーザーが入力するページングコマンドに応答してそのデータを表示するように指定します。 |
| PRINT | 印刷操作を開始します。 |
| REQID | メッセージの回復に対する一時記憶域識別子の一部として 2 文字の接頭辞を使用します。 |
| SET | 出力データのアドレスにポインタを設定します。 |
| TERMINAL | トランザクションを開始した端末に出力データを送信します。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| ACTPARTN | FORMFEED | LAST |
| LDC | L40 および L64 | MSR |
| OUTPARTN | WAIT | |

## SEND MAP

```
SEND MAP(char-7)
[MAPSET(char-7)]
[[FROM(data-area)][DATAONLY] | [MAPONLY]
[LENGTH(halfword)]
[CURSOR(halfword)]]
[ERASE | ERASEAUP]
[PRINT]
[FREEKB]
[ALARM]
[FRSET]
[ACCUM]
[SET(pointer) | PAGING | TERMINAL]
[L80 | HONEOM]
```

SEND MAP は、端末またはプログラムによって定義されたデータ領域に出力を割り当てます。あとに示す例外を除き、3278/3279 タイプの端末および PRINTER 環境変数によって識別されるシステムプリンタでサポートされます。LU Type 1 デバイスなどのその他の端末はサポートされておらず、予期しない結果になる場合があります。

| オプション | 説明 |
|---|---|
| ACCUM | 論理メッセージの構築に使用されるコマンドの 1 つ。 |
| ALARM | 3270 警告音を有効化します。 |
| CURSOR | SEND MAP コマンドの完了時に戻されるカーソルの位置。 |
| DATAONLY | アプリケーションプログラムデータのみを書き込みます。 |
| ERASE | 画面プリンタバッファー、またはパーティションを消去し、画面の左上にカーソルを戻します。 |
| ERASEAUP | パーティションまたは画面全体にあるすべての無保護の文字位置を削除します。 |
| FREEKB | 3270 キーボードのロックを解除します。 |
| FROM | 処理するデータを含むデータ領域。 |
| FRSET | 現在 3270 バッファーにある全フィールドの変更されたデータタグを変更前の状態にリセットします。 |

| オプション | 説明 |
|---|---|
| HONEOM | デフォルトのプリンタ行の長さを使用します。 |
| L80 | 3270 プリンタに対して 80 文字の行の長さを指定します。 |
| LENGTH | フォーマットするデータの長さ。 |
| MAPONLY | マップからのデフォルトのデータのみを書き込みます。 |
| MAPSET | 接尾辞のないマップセットの名前。 |
| PAGING | 出力データを端末にすぐに送信しないで一時記憶域に配置し、端末ユーザーが入力するページングコマンドに応答してそのデータを表示するように指定します。 |
| PRINT | 印刷操作を開始します。 |
| SET | 出力データのアドレスにポインタを設定します。 |
| TERMINAL | トランザクションを開始した端末に出力データを送信します。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| ACTPARTN | FMHPARM | FORMFEED |
| LAST | LDC | L40 および L64 |
| MSR | NLEOM | NOFLUSH |
| OUTPARTN | REQID | WAIT |

# SEND PAGE

```
SEND PAGE [SET(pointer-ref)]
[RELEASE [TRANSID(char-4)] | RETAIN]
[TRAILER(data-area)]
```

SEND PAGE は、指定した ACCUM オプションで SEND MAP または SEND TEXT を使用して構築された、フォーマットされたメッセージの処理を終了および転送します。3278/3279 タイプの端末でサポートされています。LU Type 1 デバイスなどのその他の端末はサポートされておらず、予期しない結果になる場合があります。

| オプション | 説明 |
|---|---|
| RELEASE | SEND PAGE コマンドでページが端末に書き込まれたあと、次に高い論理レベルにあるプログラムに制御が返されます。 |
| RETAIN | SEND PAGE コマンドでページが端末に書き込まれたあと、アプリケーションプログラムに制御が返されます。 |
| SET | 出力データのアドレスにポインタを設定します。 |
| TRAILER | 最終ページの末尾にだけ配置されるトレーラデータが含まれるテキストデータ領域。 |
| TRANSID | タスクが接続される端末からの次の入力メッセージとともに使用するトランザクション識別子。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| AUTOPAGE | CURRENT,ALL | FMHPARM |
| LAST | NOAUTOPAGE | OPERPURGE |

## SEND TEXT

```
SEND TEXT FROM(data-area)
LENGTH(halfword)
[REQID(char-2)]
[CURSOR(halfword)]
[SET(pointer) | PAGING]
[HEADER(data-area)]
[TRAILER(data-area)]
[JUSTIFY(halfword)]
[ACCUM]
[ERASE]
[PRINT]
[FREEKB]
[ALARM]
[L80 | HONEOM]
```

SEND TEXT は、制御マップを使用しないで出力をフォーマットします。3278/3279 タイプの端末でサポートされています。LU Type 1 デバイスなどのその他の端末はサポートされておらず、予期しない結果になる場合があります。

| オプション | 説明 |
|---|---|
| ACCUM | 論理メッセージの構築に使用されるコマンドの 1 つ。 |
| ALARM | 3270 警告音を有効化します。 |
| CURSOR | SEND TEXT コマンドの完了時に戻される 3270 カーソルの位置。 |
| ERASE | この出力ページが表示される前に、画面プリンタバッファー、またはパーティションを消去し、カーソルを画面の左上に戻します。 |
| FREEKB | データが書き込まれたあと、3270 キーボードのロックを解除します。 |
| FROM | 送信するデータが含まれるデータ領域。 |
| HEADER | テキストデータの各ページの最初にヘッダーデータを配置します。 |
| HONEOM | デフォルトのプリンタ行の長さを使用します。 |
| JUSTIFY | テキストデータが配置されるページの行を指定します。 |
| L80 | 3270 プリンタに対して 80 文字の行の長さを指定します。 |
| LENGTH | 送信するデータの長さ。 |
| PAGING | 出力データを端末にすぐに送信しないで一時記憶域に配置し、端末ユーザーが入力するページングコマンドに応答してそのデータを表示するように指定します。 |
| PRINT | 3270 プリンタで印刷操作を開始します。省略すると、データはプリンタバッファーに送信されますが、印刷されません。 |

| オプション | 説明 |
|---|---|
| REQID | メッセージの回復に対する一時記憶域識別子の一部として使用される 2 文字の接頭辞。 |
| SET | データのアドレスにポインタを設定します。 |
| TRAILER | トレーラデータが含まれるテキストデータ領域を各出力ページの末尾に配置します。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| ACTPARTN | FORMFEED | JUSFIRST |
| JUSLAST | LAST | LDC |
| L40 および L64 | MSR | NLEOM |
| OUTPARTN | TERMINAL LAST | TERMINAL WAIT |

## SEND TEXT MAPPED

```
SEND TEXT MAPPED FROM(data-area)
LENGTH(halfword)
[TERMINAL]
```

SEND TEXT MAPPED は、以前に BMS によって生成され、SEND または SEND TEXT コマンドの SET オプションを使ってプログラムに返されたデータを出力します。3278/3279 タイプの端末でサポートされています。LU Type 1 デバイスなどのその他の端末はサポートされておらず、予期しない結果になる場合があります。

| オプション | 説明 |
|---|---|
| FROM | 送信するデータが含まれるデータ領域。 |
| LENGTH | データの長さ。 |
| TERMINAL | トランザクションを開始した端末に入力データを送信します。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| LAST | PAGING | REQID |
| WAIT | | |

# SEND TEXT NOEDIT

```
SEND TEXT NOEDIT FROM(data-area)
[LENGTH(halfword)]
[TERMINAL]
[ERASE]
[PRINT]
[FREEKB]
[ALARM]
[L80]
```

SEND TEXT NOEDIT は、編集されていないデータストリームを端末に書き込みます。SEND TEXT NOEDIT は、3278/3279 タイプの端末でサポートされています。LU Type 1 デバイスなどのその他の端末はサポートされておらず、予期しない結果になる場合があります。

| オプション | 説明 |
|---|---|
| ALARM | 3270 警告音を有効化します。 |
| ERASE | この出力ページが表示される前に、画面プリンタバッファー、またはパーティションを消去し、カーソルを画面の左上に戻します。 |
| FREEKB | データが書き込まれたあと、3270 キーボードのロックを解除します。 |
| FROM | 送信するデータが含まれるデータ領域。 |
| L80 | 3270 プリンタに対して 80 文字の行の長さを指定します。 |
| LENGTH | データの長さ。 |
| PRINT | 3270 プリンタで印刷操作を開始します。省略すると、データはプリンタバッファーに送信されますが、印刷されません。 |
| TERMINAL | トランザクションを開始した端末にデータを送信します。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| HONEOM | LAST | L40 および L64 |
| OUTPARTN | PAGING | REQID |
| WAIT | | |

# SET CONNECTION

```
SET CONNECTION(data-value)
[ACQSTATUS(cvda) | CONNSTATUS(cvda)[ACQUIRED | RELEASED]]
[SERVSTATUS(cvda)[INSERVICE | OUTSERVICE]]
```

SET CONNECTION では、ローカル領域と別の領域または別のシステムとの間で確立する接続の定義および現在の状態を変更できます。CONNECTION 定義は、「システムエントリ」とも呼ばれます。

接続を有効にするために SET CONNECTION コマンドを使用するには、SNAP-IX のリンクステーション定義で制御点を定義する必要があります。さらに、Sun MTP 領域を起動するユーザーのユーザー ID を /etc/group ファイルの sna グループに追加する必要があります。

| オプション | 説明 |
|---|---|
| ACQSTATUS | CONNSTATUS オプションと同じ情報を指定し、互換性のためだけに保持されます。新しいアプリケーションでは CONNSTATUS を使用します。 |
| CONNECTION | 変更する接続の 4 文字の名前を指定します。これは、システムが CONNECTION 定義から認識する接続領域の名前です。 |
| CONNSTATUS | 接続によって表される遠隔システムのセッションに Sun MTP を置くかどうか、つまり接続を取得するかどうかを指定します。CVDA 値は、次のとおりです。<br>ACQUIRED: CICS がセッションに置かれます。また、接続が存在しない場合は接続を確立します。接続の SERVSTATUS 値が OUTSERVICE の場合、INSERVICE および ACQUIRED を指定する必要があります。<br>RELEASED: CICS がセッション内に置かれません。また、接続が存在する場合は接続を終了します。接続でセッションを現在使用しているすべてのタスクが終了しないと、接続は解放されません。 |
| SERVSTATUS | 接続をユーザーに使用可能な状態にするかどうかを指定します。CVDA 値は、次のとおりです。<br>INSERVICE: 接続が使用できるようにします。<br>OUTSERVICE: 接続が使用できません。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| PURGETYPE | CANCEL | FORCECANCEL |
| EXITTRACING | EXITTRACE | NOEXITTRACE |
| PENDSTATUS | NOTPENDING | FORCEPURGE |
| PURGE | FORCE | ZCPTRACING |
| NOZCPTRACE | ZCPTRACE | |

## SET FILE

```
SET FILE(char-8) [OPEN | CLOSED]
[ENABLED | DISABLED]
[WAIT | NOWAIT | FORCE]
[EMPTYREQ | NOEMPTYREQ | RESET]
```

SET FILE では、単一 VSAM ファイルの属性を変更できます。

**注 –** SET FILE は、代替索引ファイルではサポートされていません。

| オプション | 説明 |
|---|---|
| CLOSED | ファイルを閉じます。 |
| DISABLED | ファイルをアプリケーションプログラムで使用できないようにします。 |
| EMPTYREQ | OPEN のファイルを再初期化して、ファイルを空にします。<br>EMPTYREQ 時にファイルが (バッチにより) ビジー状態の場合、WAIT が指定されていないときは INVREQ 条件が返されます。 |
| ENABLED | ファイルをアプリケーションプログラムで使用可能にします。オンライントランザクションが、バッチストリームによってロックされているデータセットを有効にしようとすると、エラーが返されます。 |
| FORCE | CLOSED または DISABLED を実行する前に、ファイルですべてのアクティビティーが完了するまで待機するWAIT と同じです。 |
| NOEMPTYREQ | OPEN のファイルを再初期化しないで、ファイルを空にしません。デフォルト値です。 |
| NOWAIT | ファイルがビジー状態の場合は待機しないが、代わりに INVREQ 条件を返します。デフォルト値です。 |
| OPEN | ファイルを開きます。オンライントランザクションが、バッチストリームによってロックされているデータセットをオープンしようとすると、エラーが返されます。 |
| RESET | OPEN のファイルを再初期化して、ファイルを空にします。<br>EMPTYREQ と同じです。 |
| WAIT | CLOSED または DISABLED を実行する前に、ファイルですべてのアクティビティーが完了するまで待機します。 |

# SET PROGRAM

```
SET PROGRAM(data-value)
[COPY(cvda) | NEWCOPY]
[EXECUTIONSET(cvda) | DPLSUBSET | FULLAPI]
[JVMCLASS(data-area) | SHLIB(data-area)]
[STATUS(cvda)]
```

SET PROGRAM コマンドは、特定のプログラムの定義を変更します。

| オプション | 説明 |
|---|---|
| PROGRAM | プログラムの 8 文字の名前。 |
| COPY | 次にプログラムが要求されるときに、そのプログラムの新しいコピーがロードされるように指定します。CVDA 値は、次のとおりです。<br>NEWCOPY: 次にプログラムが呼び出しで要求されるときに、そのプログラムの新しいコピーが使用されます。<br>PHASEIN: 現在サポートされていません。トランスレータがこのオプションを認識すると、警告が出力されます。 |
| EXECUTIONSET | プログラムが CICS API の分散プログラムリンク (DPL) サブセットの実行に制限されるかを指定します。<br>EXECUTIONSET は、実行可能プログラムに対してのみ適用され、プログラムがローカルで実行されるときにのみ API を制御します。<br>CVDA 値は、次のとおりです。<br>DPLSUBSET: プログラムが常に制限されます。<br>FULLAPI: プログラムが遠隔で起動される場合を除き、プログラムは制限されません。 |
| JVMCLASS | プログラムの Java クラス名を指定する 255 文字 のデータ領域。Java プログラムにだけ有効です。 |
| SHLIB | プログラムに関連する共有オブジェクトの名前を示す 16 文字 のデータ領域。KIXLIB 環境変数がパス名に設定されていない場合、$KIXSYS 関連と見なされます。(Sun MTP 固有のオプション)。 |
| STATUS | プログラムが使用可能かどうかを指定します。CVDA 値は、次のとおりです。<br>DISABLED: プログラムが使用できません。<br>ENABLED: プログラムを実行できます。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| CEDFSTATUS | HOTPOOLING | JVMDEBUG |
| RUNTIME | SHARESTATUS | VERSION |

## SET TDQUEUE

```
SET TDQUEUE(char-4) [ENABLED | DISABLED]
[OPEN | CLOSED]
[TRIGGERLEVEL(integer)]
```

SET TDQUEUE では、パーティション内およびパーティション外の一時データキューの属性を変更できます。すべてのキューは起動時に有効です。

| オプション | 説明 |
|---|---|
| CLOSED | キューを閉じます。パーティション外のデータキューだけに適用します。DISABLED キューを閉じることはできません。 |
| DISABLED | キューをアプリケーションで使用できないようにします。DISABLED キューを開くことも閉じることもできません。DISABLED キューを読み込み、書き込み、または削除すると、トランザクションが強制的に中止されて DISABLED 条件が返されます。 |
| ENABLED | キューをアプリケーションで使用できるようにします。 |
| OPEN | キューを開きます。パーティション外のデータキューだけに適用します。DISABLED キューを開くことはできません。 |
| TRIGGERLEVEL | DCT で指定したトランザクションが開始される前に許可する出力要求の数。レベル番号は 0 ～ 32,767。<br>• トリガーレベルはパーティション内キューだけに適用されます。<br>• DISABLED パーティション内キューのトリガーレベルは変更可能。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| ATIFACILITY | ATITERMID | ATITRANID |

# SET TERMINAL

```
SET TERMINAL(termid)
[ACQSTATUS(cvda)|TERMSTATUS(cvda)[ACQUIRED | RELEASED]]
```

SET TERMINAL は、指定した端末定義の値の一部を変更します。

| オプション | 説明 |
|---|---|
| ACQSTATUS | 端末によって表される論理ユニットのセッションの状態を設定します。互換性のためにのみ使用します。新しいアプリケーションでは TERMSTATUS を使用します。CVDA 値については、「TERMSTATUS」を参照してください。 |
| TERMSTATUS | 端末によって表される論理ユニットのセッションの状態を設定します。新しいアプリケーションで使用します。CVDA 値は、次のとおりです。<br>ACQUIRED: Sun MTP が、この端末によって表される論理ユニットとのセッションを取得します。<br>RELEASED: Sun MTP がセッションを終了します。現在有効なトランザクションが終了すると、セッションが終了されます。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| ALTPRINTER | ALTPRTCOPYST | ATISTATUS |
| CREATESESS | DISCREQST | EXITTRACING |
| NEXTTRANSID | OBFORMATST | PAGESTATUS |
| PRINTER | PRTCOPYST | PURGE |
| PURGETYPE | RELREQST | SERVSTATUS |
| TCAMCONTROL | TERMPRIORITY | TRACING |
| TTISTATUS | UCTRANST | ZCPTRACING |

応答は、次のとおりです。

| | |
|---|---|
| NORMAL | このコマンドの実行が成功すると返されます。 |
| TERMIDERR | 指定した *termid* が TCT で見つからない場合に返されます。 |
| INVREQ | 次の状況で返されます。<br>• Sun MTP 領域が SNA 3270 端末をサポートしないので、unikixtrin が実行されていない場合。<br>• 指定した端末の TCT エントリで識別される Sun MTP 端末ハンドラ (unikixi) が実行されていない場合。<br>• コマンドが ACQUIRED 端末を ACQUIRED に設定しようとする場合。<br>• コマンドが RELEASED 端末を RELEASED に設定しようとする場合。 |

## SET TRANCLASS

```
SET TRANCLASS(8-char data value) MAXACTIVE(data-area)
[[USE](8-char target class name)]
```

SET TRANCLASS コマンドは、トランザクションクラスに割り当てられたトランザクション処理プログラムの数を変更するために使用します。PURGETHRESH を除くすべてのオプションがサポートされます。

| オプション | 説明 |
|---|---|
| MAXACTIVE | クラスに割り当てられる有効なトランザクション処理プログラムの新しい数。 |
| [USE](8-char) | トランザクション処理プログラムの再割り当て先または再割り当て元のトランザクションクラス。指定されていない場合、KIXDFLT クラスを使います。[USE] キーワードはオプションであり、8 文字のターゲットクラス名は単独で使用できます。このオプションは、API への Sun MTP 拡張子です。 |

これらの文から返される条件は、次のとおりです。

| | |
|---|---|
| TCIDERR | クラス名が無効な場合に返されます。 |
| INVREQ | MAXACTIVE パラメータが無効で、クラスを再構成できない場合に返されます。 |

次のオプションはサポートされていません。

PURGETHRESH

## SET TRANSACTION

```
SET TRANSACTION(data-value)
[DUMPING(cvda) | TRANDUMP | NOTRANDUMP]
[TRANCLASS(data-value)]
[STATUS(cvda) | DISABLED | ENABLED]
```

SET TRANSACTION コマンドは、トランザクション定義の属性を変更します。

| オプション | 説明 |
|---|---|
| TRANSACTION | 4 文字のトランザクション識別子。 |
| DUMPING | このトランザクションを実行中のタスクが異常終了した場合にトランザクションダンプをとるかどうかを指定します。CVDA 値は、次のとおりです。<br>NOTRANDUMP: ダンプをとりません。<br>TRANDUMP: ダンプをとります。 |
| TRANCLASS | このトランザクションが属しているトランザクションクラスの 8 文字の名前。 |
| STATUS | トランザクションが使用可能かどうかを示します。CVDA 値は、次のとおりです。<br>DISABLED: トランザクションを実行できません。<br>ENABLED: トランザクションを実行できます。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| PRIORITY | PURGEABILITY | RUNAWAY |
| RUNAWAYTYPE | SHUTDOWN | TRACING |

## SIGNOFF

```
SIGNOFF
```

SIGNOFFは、以前にサインオンした端末または基本端末からサインオフします。失敗時には、EIBRESP2 の値によって失敗の理由が示されます。

| 条件 | RESP2 | 理由 |
|---|---|---|
| INVREQ | 1 | サインオンしていません。 |
| INVREQ | 2 | このトランザクションには端末がありません。 |
| INVREQ | 3 | このタスクの端末は、プリセットセキュリティーを持ちます。 |

## SIGNON

```
SIGNON USERID(data-value)
[GROUPID(data-value)
[PASSWORD(data-value)]
[NEWPASSWORD(data-value)]
```

SIGNON は、USERID で指定されるユーザーの特性を端末に関連付けます。

| オプション | 説明 |
|---|---|
| USERID | 8 文字のユーザーサインオン名。 |
| GROUPID | 外部セキュリティーマネージャー (ESM) でユーザーが割り当てられた役割に該当するグループ名。 |
| PASSWORD | ESM で必要な 8 文字のパスワード。 |
| NEWPASSWORD | 新しいパスワードを定義する 8 文字のフィールド。PASSWORD が指定されている場合だけ有効です。<br>指定されている場合は、パスワードが新しいパスワード文字列に変更されます。 |

失敗時には、RESP2 によって失敗の理由が示されます。

| 条件 | RESP2 | 理由 |
|---|---|---|
| INVREQ | 2 | このタスクの端末は、プリセットセキュリティーを持ちます。 |
| INVREQ | 5 | ユーザーがパスワードを変更しようとしたが、パスワードの変更に必要な最小時間が経過していません。 |
| INVREQ | 6 | 新しいパスワードが必要ですが、入力されませんでした。 |
| INVREQ | 9 | 端末がすでにサインオンされています。 |
| INVREQ | 10 | タスクに関連する端末がありません。 |
| INVREQ | 11 | このタスクの端末は、プリセットセキュリティーを持ちます。 |
| INVREQ | 13 | ESM が、この要求の実行に失敗しました。 |
| NOTAUTH | 2 | パスワードが間違っています。 |
| NOTAUTH | 3 | パスワードの期限が切れています。 |
| NOTAUTH | 4 | 新しいパスワードが最大長を超えています。新しいパスワードが最小長に達していません。新しいパスワード値は、古いパスワード値と同じです。新しいパスワードが空白です。または、新しいパスワードの形式が間違っています (数字でない、アルファベットでない)。 |
| NOTAUTH | 16 | ユーザー ID は、この端末で許可されていません。 |

| 条件 | RESP2 | 理由 |
|---|---|---|
| NOTAUTH | 17 | ユーザー ID は、この領域で許可されていません。($KIXSYS) |
| NOTAUTH | 19 | ユーザー ID が保留になっています。 |
| NOTAUTH | 23 | 役割名が無効です。 |
| NOTAUTH | 24 | ユーザー ID はこの役割を使用できません。 |
| USERIDERR | 8 | ユーザー ID が無効です。 |

次のオプションはサポートされていません。

```
ESMRESP          NATLANG          NATLANGINUSE
OIDCARD
```

# START

```
START [INTERVAL(packed-7) | TIME(packed-7) |
AFTER [HOURS(data-value)]
 [MINUTES(data-value) [SECONDS(data-value)] |
AT [HOURS(data-value)]
 [MINUTES(data-value) [SECONDS(data-value)]]
TRANSID(char-4)
[REQID(char-8)]
[FROM(data-area) LENGTH(halfword)]
[TERMID(char-4)]
[SYSID(char-4)]
[RTRANSID(char-4)]
[RTERMID(char-4)]
[QUEUE(char-8)]
[NOCHECK]
[PROTECT]
```

START コマンドは、同じ端末、別の端末、またはプリンタで非同期タスクを開始します。これにより、デバイスから切断されているタスクを開始できます。

| オプション | 説明 |
|---|---|
| AFTER | HOURS、MINUTES、SECONDS オプションのいずれかを使用する場合、新しいタスクが開始されるまでの時間間隔を AFTER が指定します。<br>このオプションは、C または C++ プログラムの INTERVAL の代わりに使用する必要があります。<br>AFTER HOURS(0) がデフォルトです。 |
| AT | 新しいタスクが開始される時刻を指定します。<br>このオプションは、C または C++ プログラムの TIME の代わりに使用する必要があります。 |
| FROM | 今後開始されるタスクのために格納するデータ。 |
| HOURS | 時間数を指定する 32 ビット 2 進値。AFTER または AT と組み合わせて使用します。範囲は 0 ～ 99。 |
| INTERVAL | START コマンドが発行されてから期限切れになるまでの経過時間<br>夏時間の変更を考慮し、トランザクションが正しい時刻に開始されることを確実にするには、KIX_ADJ_DST 環境変数に値を設定する必要があります。ローカルな時間帯で有効でない時刻にアプリケーションがトランザクションを開始しようとする場合、START コマンドの結果は予想できません。 |
| LENGTH | 新しいタスクのために格納するデータの長さ。 |
| MINUTES | 分数を指定する 32 ビット 2 進値。AFTER または AT と組み合わせて使用します。HOURS または SECONDS も指定する場合、範囲は 0 ～ 59。それ以外の場合の範囲は 0 ～ 5,999。 |
| NOCHECK | SYSID をコード化すると、システムのエラーチェックおよび機能を少なくすることにより、START コマンドのパフォーマンスが向上します。 |
| PROTECT | ローカルトランザクションが同期点を取るまでは、開始されるトランザクションをスケジュールできません。同期点を取る前に開始されたタスクが不正終了すると、START 要求は取り消されます。 |
| QUEUE | 開始されるトランザクションが使用できる一時記憶域キューの名前。 |
| REQID | コマンドを識別する一意の名前。一時記憶域識別子として使用。 |
| RTERMID | TRANSID オプションで指定されているトランザクションが開始されると取得される、端末名などの値。 |
| RTRANSID | TRANSID オプションで指定されているトランザクションが開始されると取得される、トランザクション名などの値。 |
| SECONDS | 秒数を指定する 32 ビット 2 進値。AFTER または AT と組み合わせて使用します。HOURS または MINUTES も指定する場合、範囲は 0 ～ 59。それ以外の場合の範囲は 0 ～ 359,999。 |

| オプション | 説明 |
|---|---|
| SYSID | 指定されている場合、TCT– System Entries 画面のエントリに一致する必要があります。非同期処理については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |
| TIME | 新しいタスクが開始される時刻を指定します。 |
| TRANSID | SYSID がコード化されていない場合、PCT のエントリに一致する必要があります。PCT については、『Sun Mainframe Transaction Processing 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。 |
| TERMID | コマンドがローカルで実行される場合、TCT のエントリに一致する必要があります。TCT については、『Sun Mainframe Transaction Processing 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。 |

次のオプションはサポートされていません。

FMH

## STARTBR

```
STARTBR DATASET(dataset) | FILE(dataset)
RIDFLD(data-area)
[KEYLENGTH(halfword) [GENERIC]]
[REQID(halfword)]
[SYSID(char-4)]
[RBA | RRN]
[GTEQ | EQUAL]
```

STARTBR は、VSAM データセットで完全にサポートされています。ブラウズ操作を開始するデータセットのレコードを指定します。索引順次アクセス方式 (ISAM) および直接アクセス方式 (DAM) データセットは、サポートされていません。

| オプション | 説明 |
|---|---|
| DATASET | SYSID がコード化されていない場合、FCT のエントリに一致する必要があります。FCT については、『Sun Mainframe Transaction Processing 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。 |
| EQUAL | 検索は、RIDFLD オプションで指定されているのと同じキーを持つレコードの場合のみ一致します。 |
| FILE | アクセスするファイルの名前。これは FCT のデータセット名です。 |
| GENERIC | 検索キーは、KEYLENGTH オプションで指定された長さの総称キー。 |

| オプション | 説明 |
|---|---|
| GTEQ | RIDFLD オプションで指定されているのと同じキーを持つレコードの検索に失敗した場合、それより大きいキーを持つ最初のレコードが取り出されます。 |
| KEYLENGTH | RIDFLD オプションで指定されたキーの長さ。 |
| RBA | RIDFLD オプションで指定されたレコード識別フィールドが相対バイトアドレスを示します。 |
| REQID | ブラウズの一意の要求識別子。データセットで複数のブラウズ操作を制御する場合に使用します。 |
| RIDFLD | レコードを特定するフィールド。 |
| RRN | RIDFLD オプションで指定されたレコード識別フィールドが相対レコード番号を示します。 |
| SYSID | 指定されている場合、TCT–System Entries 画面のエントリに一致する必要があります。機能シップについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |

次のオプションはサポートされていません。

```
DEBKEY          DEBREC
```

## SUSPEND

```
SUSPEND
```

SUSPEND は、完全にサポートされています。より優先順位の高いタスクに制御を譲るために使用します。ただし、Sun MTP 領域のタスクは別のトランザクションサーバーで実行されるので、この影響を受けません。

## SYNCPOINT

```
SYNCPOINT
```

SYNCPOINT は、作業論理ユニットの完了を示します。最後の同期点以来、回復可能なリソースへのすべての変更を確定するタスクを指示します。

次の条件が認識されます。

| 条件 | RESP2 | 理由 |
|---|---|---|
| INVREQ | 200 | SYNCPOINT は、SYNCONRETURN オプションを指定していない遠隔システムからリンクされるプログラムにありました。または、ローカルにリンクされていて、EXECUTIONSET=DPLSUBSET で定義されています。<br>デフォルトのアクションでは、タスクを異常終了します。 |
| ROLLEDBACK | n/a | SYNCPOINT コマンドが、実行しているリソースマネージャーによってロールバックに強制された場合に発生します。作業での現在の単位における回復可能なすべてのリソースへの変更は取り消されます。<br>アプリケーションが RESP コードを指定していない、または HANDLE 状態を確立していない場合は、トランザクションがコード AEXJ で異常終了します。 |

## SYNCPOINT ROLLBACK

```
SYNCPOINT ROLLBACK
```

SYNCPOINT ROLLBACK は、作業論理ユニットの完了を示します。最後の同期点以来、回復可能なリソースへのすべての変更を取り消すタスクを指示します。

次の条件が認識されます。

| 条件 | RESP2 | 理由 |
|---|---|---|
| INVREQ | 200 | SYNCPOINT ROLLBACK は、SYNCONRETURN オプションを指定していない遠隔システムからリンクされるプログラムにありました。または、ローカルにリンクされていて、EXECUTIONSET=DPLSUBSET で定義されています。<br>デフォルトのアクションでは、タスクを異常終了します。 |

## TRACE

```
TRACE [ON | OFF]
[SYSTEM]
[EI]
[USER]
[SINGLE]
```

TRACE は、システムおよびユーザーのトレーステーブルエントリを制御します。Sun MTP のトレースは、端末ごとに行われます。この制限内で、TRACE は完全にサポートされます。

## UNLOCK

```
UNLOCK DATASET(dataset) | FILE(dataset)
[SYSID(char-4)]
```

UNLOCK は、VSAM データセットで完全にサポートされています。更新のためにレコードが保持されているファイルを解放します。ISAM および DAM データセットはサポートされません。

| オプション | 説明 |
|---|---|
| DATASET | FCT のエントリに一致する必要があります。SYSID が指定されている場合、そのファイルは遠隔であるとみなされる FCT については、『Sun Mainframe Transaction Processing 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。 |
| FILE | 解放するデータセットの名前。FCT のエントリに一致する必要があります。SYSID が指定されている場合、そのファイルは遠隔であるとみなされます。 |
| SYSID | 要求が送信されるシステム。指定されている場合、TCT– System Entries 画面のエントリに一致する必要があります。TCT については、『Sun Mainframe Transaction Processing 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。 |

## VERIFY PASSWORD

```
VERIFY PASSWORD(current-pw)(char-8)
USERID(current-uid) (char-8)
[DAYSLEFT(remaining days)](halfword binary)
[EXPIRYTIME(expiration date)](packed decimal 8 bytes)
ESMREASON(err-reason) (fullword binary)
```

VERIFY PASSWORD は、入力されたパスワードが有効であるかどうかを確認するために使用します。

| オプション | 説明 |
|---|---|
| PASSWORD | 現在のパスワード。 |
| USERID | 現在のユーザー ID。 |
| DAYSLEFT | パスワードの有効期限。 |
| EXPIRYTIME | パスワードの期限切れの日付。 |
| ESMREASON | 新しいパスワードに関するエラーの詳細。 |

DAYSLEFT および EXPIRYTIME は、省略可能な引数です。パスワードが期限切れでない場合、この 2 つの引数は -1 として返されます。

PASSWORD および USERID が有効であれば、その他の情報が返されます。有効でない場合は、次のエラー条件が返されます。

| 条件 | RESP2 | 理由 |
|---|---|---|
| NOTAUTH | 2 | パスワードが間違っています。 |
| NOTAUTH | 3 | パスワードの期限が切れています。 |
| NOTAUTH | 19 | ユーザー ID が停止されていて、ESMREASON が 901 に設定されています。 |
| USERIDERR | 8 | ユーザー ID が無効です。 |
| INVREQ | 10 | タスクに関連する端末がありません。 |

次のオプションはサポートされていません。

CHANGETIME ESMRESP INVALIDCOUNT

LASTUSETIME

## WAIT CONVID

```
WAIT CONVID(char-4) [STATE(cvda)]
```

WAIT CONVID は、完全にサポートされています。DTP 会話でバッファーされたデータをフラッシュするために使用します。

| オプション | 説明 |
|---|---|
| CONVID | コマンドに関連する会話を識別します。 |
| STATE | 現在の会話の状態を取得します。 |

STATE の CVDA 値は、次のとおりです。

| | | |
|---|---|---|
| ALLOCATED | CONFFREE | CONFRECEIVE |
| CONFSEND | FREE | PENDFREE |
| PENDRECEIVE | RECEIVE | ROLLBACK |
| SEND | SYNCFREE | SYNCRECEIVE |
| SYNCSEND | | |

## WAIT EVENT

```
WAIT EVENT ECADDR(pointer-value)
```

WAIT EVENT は、完全にサポートされています。指定されたタイマーイベント制御領域が送信されるまで、タスクの処理を中断します。

| オプション | 説明 |
|---|---|
| ECADDR | タイマーイベント制御領域のアドレス。このアドレスが送信されないとアクティビティーが再開されません。 |

## WAIT JOURNAL

```
WAIT JOURNAL JFILEID(halfword)
[REQID(fullword)]
[STARTIO]
```

WAIT JOURNAL は、JOURNAL ファイルとトランザクションを同期し、レコードが確実に書き込まれるようにします。Sun MTP の JOURNAL は同期をとっているので、WAIT JOURNAL コマンドは影響しません。

## WRITE

```
WRITE DATASET(dataset) | FILE(dataset)
FROM(data-area)
[LENGTH(halfword)]
RIDFLD(data-area)
[KEYLENGTH(halfword)]
[SYSID(char-4)]
[RBA | RRN]
[MASSINSERT]
```

WRITE は、VSAM データセットで完全にサポートされています。データセットにレコードを書き込みます。ISAM および DAM データセットはサポートされません。

| オプション | 説明 |
|---|---|
| DATASET | SYSID がコード化されていない場合、FCT のデータセット名に一致する必要があります。 |
| FILE | アクセスするファイルの名前。これは FCT のデータセット名です。 |
| FROM | 指定したデータセットに書き込むレコード。 |
| KEYLENGTH | RIDFLD オプションで指定されるキーの長さ (RBA または RRN が指定されている場合を除く)。 |
| LENGTH | 書き込むレコードの長さ。 |
| MASSINSERT | WRITE コマンドを大量の挿入操作に使用することを指定します。すなわち、連続して WRITE を指定し、それぞれに MASSINSERT を指定します。 |
| RBA | RIDFLD オプションで指定されたレコード識別フィールドが相対バイトアドレスを示すように指定します。ESDS ファイルへの書き込み時にのみ使用します。<br>サイズが 4 ギガバイトを超える場合、INVREQ が設定され、RESP2 が 1 に設定されます。 |
| RIDFLD | レコードを特定するフィールド。 |
| RRN | RIDFLD オプションで指定されたレコード識別フィールドが相対レコード番号を示します。RRDS ファイルへの書き込み時にのみ使用します。 |
| SYSID | 指定されている場合、TCT–System Entries 画面のエントリに一致する必要があります。機能シップについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |

## WRITE OPERATOR

```
WRITE OPERATOR TEXT(data-value) [TEXTLENGTH(data-value)]
```

WRITE OPERATOR は、各種のオプションをサポートします。コンソール (CONSOLE) への出力は、unikixmain.log に送信されます。unikixmain.log を表示するには、$KIXSYS ディレクトリから次のコマンドを入力します。

```
$ tail -f unikixmain.log
```

| オプション | 説明 |
|---|---|
| TEXT | コンソールへの出力が含まれるデータ領域。 |
| TEXTLENGTH | フルワードの 2 進値で表されるテキストの長さ。テキストの長さが 120 を超える場合、INVREQ が設定され、RESP2 が 1 に設定されます。 |

次のオプションはサポートされていません。

| | | |
|---|---|---|
| ACTION | CRITICAL | EVENTUAL |
| IMMEDIATE | NUMROUTES | REPLY |
| ROUTECODES | TIMEOUT | |

## WRITEQ TD

```
WRITEQ TD QUEUE(char-4)
FROM(data-area)
[LENGTH(halfword)]
[SYSID(char-4)]
```

WRITEQ TD は、完全にサポートされています。QUEUE オプションで指定された書き込み先に一時データを書き込みます。

| オプション | 説明 |
|---|---|
| FROM | 一時データキューに書き込まれるデータ。 |
| LENGTH | 書き込むデータの長さ。 |
| QUEUE | SYSID がコード化されていない場合、DCT のエントリに一致する必要があります。書き込み先タイプが PRINTER の場合、データはシステムプリンタに書き込まれます。DCT については、『Sun Mainframe Transaction Processing 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。 |
| SYSID | 指定されている場合、TCT– System Entries 画面のエントリに一致する必要があります。機能シップについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |

## WRITEQ TS

```
WRITEQ TS QUEUE(char-8)
FROM(data-area)
LENGTH(halfword)
[ITEM(halfword) | [REWRITE]]
[NUMITEMS(halfword)]
[SYSID(char-4)]
[MAIN | AUXILIARY]
[NOSUSPEND]
```

WRITEQ TS は、完全にサポートされています。QUEUE オプションで指定された一時記憶域キューにデータを書き込みます。

| オプション | 説明 |
|---|---|
| AUXILIARY | 一時記憶域キューが、補助記憶装置の直接アクセス記憶装置上にあります。 |
| FROM | 一時記憶域に書き込まれる補助データ。 |
| ITEM | キュー内で置換される項目。NUMITEMS を参照してください。 |

| オプション | 説明 |
|---|---|
| LENGTH | 書き込まれるデータの長さ。 |
| MAIN | 一時記憶域キューが、主記憶域にあります。 |
| NOSUSPEND | 一時記憶域はユーザーの仮想アドレス空間から取得され、システムの他の部分にあるイベントに関係なく利用できるので、ローカル要求に影響しません。 |
| NUMITEMS | ハーフワードの 2 進フィールド。ここには、WRITEQ TS コマンドの完了後にキューに存在するレコードの合計数が格納されます。書き込まれるレコードが新しいキューを開始した場合、完了時に返される番号は 1 で、後続の項目番号はこの数に続きます。<br>コマンド上で REWRITE も指定されている場合は、NUMITEMS は確実に有効とはなりません。<br>以前のリリースとの互換性のため、REWRITE を持たない WRITEQ TS コマンドの ITEM は、NUMITEMS と同様の働きをします。 |
| QUEUE | SYSID がコード化されていない場合、キューが回復可能または遠隔であるときは、TST のエントリと一致する必要があります。TST については、『Sun Mainframe Transaction Processing 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。 |
| REWRITE | キュー内の既存のレコードを提供されたデータで上書きします。REWRITE を指定する場合、ITEM を指定する必要があります。 |
| SYSID | 指定されている場合、TCT–System Entries 画面のエントリに一致する必要があります。機能シップについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。 |

## XCTL

```
XCTL PROGRAM(char-8)
[COMMAREA(data-area)[LENGTH(halfword)]]
```

XCTL は、あるアプリケーションプログラムからほかのアプリケーションプログラムに制御を渡しますが、この制御はあとで戻されません。

| オプション | 説明 |
|---|---|
| COMMAREA | 起動したプログラムで使用できるようにする通信領域。 |
| LENGTH | 通信領域のバイト単位の長さ。 |
| PROGRAM | PPT のエントリに一致する必要があります。PPT については、『Sun Mainframe Transaction Processing 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。 |

次のオプションはサポートされていません。

INPUTMSG　　INPUTMSGLEN

# 独自のコマンド

LEXECTERM は、CICS API 構文に準拠する独自のコマンドです。

## LEXECTERM

```
LEXECTERM FROM(COM-BUF)
LENGTH(COM-LEN)
```

LEXECTERM コマンドによって、アプリケーションプログラムでシステムコール経由のコマンド行から別のプログラムを実行できます。コマンド行を形成するシェルスクリプト、プログラム、およびコマンドは、$UNIKIX/bin に存在する必要があります。コマンドが完了すると、通常はプログラムに制御が返されます。

| オプション | 説明 |
|---|---|
| FROM | COM-BUF が実行するコマンド行を示します。 |
| LENGTH | COM-LEN は、COM-BUF に示されるコマンド行の 1 ～ 256 の長さ。長さフィールドの値が範囲外の場合、次のエラーメッセージが表示されます。<br>1480E Key length may not be greater than 255（**キーの長さは 255 を超えることはできません**）<br>組み込まれた EXEC CICS LEXECTERM を持つトランザクションを起動した端末が、ローカルの非 3270 タイプの端末ではない場合、次の INVREQ エラーメッセージが表示されます。<br>Transaction not authorized for 3270 terminal（**トランザクションは 3270 端末では許可されていません**） |

# サポートされている BMS 機能

Sun MTP は、3270 モデル 2、モデル 4、モデル 5 端末、および CICS プリンタに対する BMS ソフトウェア機能をほとんどすべてサポートしています。これらのプリンタおよび端末タイプに対しては、BMS の標準コマンドセットもサポートされます。

次の表に、最小、標準、および完全の各バージョンにおける BMS 機能のサポートを示します。機能がサポートされないのは、オペレーティングシステムによって利用できるハードウェアが異なるためです。内部的には、Sun MTP は 3270 タイプの端末を直接サポートし、*curses* を使用して標準 ASCII 端末もサポートします。curses は、画面処理および最適化パッケージです。すべての BMS データストリームは、内部使用のために、最初に 3270 データストリームに変換されます。さらに 3270 タイプの端末以外の端末では、これらのデータストリームは、端末への出力用の UNIX *curses* ライブラリを使用して端末ハンドラによって変換されます。

**表 4-3**　バージョン別のサポートされる BMS 機能

| BMS バージョン | 提供される機能 | サポート |
|---|---|---|
| 最小 | `SEND MAP` コマンド | あり。サポートされているオプションについては、100 ページの「`SEND MAP`」を参照してください。 |
| | `RECEIVE MAP` コマンド | あり。サポートされているオプションについては、92 ページの「`RECEIVE MAP`」を参照してください。 |
| | `SEND CONTROL` コマンド | あり。サポートされているオプションについては、99 ページの「`SEND CONTROL`」を参照してください。 |
| | デフォルトおよび代替画面 | デフォルトおよび代替。デフォルトでは 3270 モデル 2 を使用。代替では モデル 2、4、または 5 が使用可能です。 |
| | Extended Attributes | プログラムで使用可能。拡張属性によるマップのコピーを生成。一部の属性は特定のデバイスでだけサポートされます。詳細は、129 ページの「拡張属性」を参照してください。 |
| | マップセット接尾辞 | あり |
| | Null マップによる画面調整 | あり |
| | フィールドおよびブロックデータ | フィールドだけ |

表 4-3　バージョン別のサポートされる BMS 機能 (続き)

| BMS バージョン | 提供される機能 | サポート |
|---|---|---|
| 標準 | 外部フォーマット | なし |
| | パーティション | なし |
| | 磁気スロット読み取り装置の制御 | なし |
| | 3270 プリンタの NLEOM | なし |
| | SEND TEXT コマンド | あり。サポートされているオプションについては、103 ページの「SEND TEXT」を参照してください。 |
| | サブシステム LDC 制御 | なし |
| 完全 | 端末オペレータのページング | あり |
| | 累積マッピング | あり |
| | ページオーバーフロー | あり |
| | 累積テキスト処理 | あり |
| | 経路指定 (ROUTE コマンド) | なし |
| | メッセージ交換 | なし |
| | 出力の前に BMS 生成データストリームをプログラムに返す | あり |

# 拡張属性

拡張属性は、端末タイプごとにサポートが異なります。表 4-4 に、各端末タイプで利用できるサポートを示します。

次の拡張属性はサポートされていません。

- 透過性
- 検査

**表 4-4**　端末タイプ別のサポートされる拡張属性

| Terminal | 拡張属性 | サポート |
|---|---|---|
| 3270* | Color | 完全サポート |
| | Hilight | 完全サポート |
| | Outline | 完全サポート |
| | SOSI | 完全サポート |
| | PS | 1 領域につき、プログラムされたシンボルを 1 つだけサポート |
| UNIX 端末クライアントを使用する ASCII 端末 | Color | サポートされません |
| | Hilight | 使用する実際の端末の機能によっては完全サポート |
| | Outline | サポートされません |
| | SOSI | 完全サポート |
| | PS | 1 領域につき、プログラムされたシンボルを 1 つだけサポート |
| UNIX 端末クライアントを使用して VT220 をエミュレートする X 端末 | Color | サポートされません |
| | Hilight | 点滅以外をサポート |
| | Outline | サポートされません |
| | SOSI | 完全サポート |
| | PS | 1 領域につき、プログラムされたシンボルを 1 つだけサポート |

* 3270 端末のバインド時に、端末が拡張属性をサポートされるかどうかが決定されます。デフォルトでは、LU を 0x80 に設定して拡張属性を使用できるようにするために、3270 ログインは端末がバインド形式の 15 バイトの上位ビットに設定されていることを前提とします。

デフォルトの 3270 ログインを変更して、拡張属性を有効にすることができます。この変更を適切に行う方法については、3270 端末のユーザーズガイドを参照してください。

# 代替画面サイズ

3270 端末のエミュレート時は、Sun MTP のデフォルトの画面サイズは 24 行 x 80 文字です。Sun MTP は、次の 2 つの代替画面サイズもサポートしています。

- 43x80 (3270 モデル 4)
- 27x132 (3270 モデル 5)

Sun MTP は PCT の端末および画面サイズフィールドを使用して、代替画面サイズが利用できるか、さらにそれを使用するかどうかを決定します。端末タイプによってパラメータの設定は異なります。

トランザクションが PCT エントリで `ALT` の画面サイズ値を持つ場合、代替画面サイズは有効化されます。実際の端末が代替画面サイズを利用できる場合、Sun MTP は端末に対する最初の `ERASE/WRITE` コマンドを `ERASE/WRITE ALTERNATE` に変更して、画面を代替画面サイズに変更します。`EXEC CICS SEND`、`SEND MAP`、またはその他の端末出力コマンドによって、`ERASE/WRITE` コマンドを生成できます。

マップセット接尾辞により、特定のスクリーンで表示されるマップセットが決定します。マップセットの接尾辞は、BMS および端末定義で指定されます。規則により、モデル 4 のマップセットは通常 4 という接尾辞を使用し、モデル 5 のマップセットは 5 という接尾辞を使用します。モデル 2 のマップセットは接尾辞に文字 M を使用します。

たとえば、トランザクションは通常どのサイズの端末でも実行できるように構築されます。これを達成するには、トランザクションは BMS を使用して出力を送信する必要があります。BMS は、アプリケーションと異なる画面サイズを隠すように設計されている次のアルゴリズムを使用します。

- 代替画面サイズが使用され、マッピング操作が要求されると、Sun MTP はマップセットを検索する要求でマップセット名を使用します。このマップセット名には接尾辞が付加されています。
- マップセットが見つからない場合、3270 モデル 2 マップセットを検索するためにマップセットに M が付加されます。
- さらにそのマップセットが見つからない場合は、名前を直接使用します。

## 画面サイズの決定

画面サイズは、端末タイプに基づいて決定されます。

### 3270 端末

Sun MTP は、3270 端末のバインドによって画面サイズを決定します。バインドの バイト 24 は、端末の画面サイズ機能を記述するビットマスクを示します。デフォルトでは、3270 端末は 24 行 x 80 文字の画面を表示できるのみです。ただし、バイト 24 が 0x7F の場合、バイト 22 および 23 の値が代替画面サイズとして使用されます。43 行 x 80 文字および 27 行 x 132 文字の代替画面サイズのみがサポートされます。

### ASCII 端末

Sun MTP は、ASCII 端末の画面サイズを端末クライアントのオプションから決定します。端末クライアントを起動するときに、代替画面サイズは、端末がサポートできる最大サイズにデフォルトで設定されます。たとえば、画面が 43 行 x 80 文字と同じかこれよりも大きいが、132 文字幅でない場合、ASCII 端末クライアントはモデル 4 の代替画面サイズのサポートにデフォルトで設定されます。

画面が 27 行 x132 文字と同じかこれよりも大きい場合、ASCII 端末クライアントはモデル 5 の代替画面サイズのサポートにデフォルトで設定されます。このサポートを取り消すには、-4 および -5 オプションを使用してそれぞれモデル 4 またはモデル 5 を設定します。すべての場合で、画面は選択したサイズをサポートできる大きさである必要があります。

`LINES` および `COLS` 環境変数は、実際のサイズを決定するために使用されます。実際の 3270 端末とは異なり、モデル 2 トランザクションとモデル 4 トランザクション またはモデル 5 トランザクションとの間で切り換えが行われても、フォントは変更されません。モデル 2 に必要な画面の部分のみが使用され、その他の領域は保護されます。

## BMS 接尾辞の指定

BMS 接尾辞の設定は、端末タイプごとに異なります。

### 3270 タイプの端末

3270 タイプの端末は通常、TCT で定義する必要はありません。ただし、BMS 接尾辞を設定する場合は、TCT– 3270 Devices の画面で端末および接尾辞を定義する必要があります。TCT については、『Sun Mainframe Transaction Processing 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。

### ローカルクライアント端末

unikix コマンドに -x オプションを使用します。これにより、接尾辞になる英数字 1 文字が設定されます。-x オプションがないと、接尾辞は設定されません。

# サポートされている EIB フィールド

この節では、CICS EXEC インタフェースブロック (EIB) のフィールドについて説明し、さらに、サポートされているフィールドについて確認します。フィールドがサポートされていない場合 (次の表で「なし」と表記)、EIB エントリは空白または 2 進数のゼロに設定されます。エントリはデバッグ画面およびダンプで表示されますが、その他の有用なリファレンスはありません。

**表 4-5** CICS EXEC インタフェースブロック (EIB) フィールドのサポート ( 1 / 3 )

| EIB エントリ | 長さ/タイプ | 説明 | Sun MTP サポート |
|---|---|---|---|
| EIBAID | PIC X(1). | 表示デバイスからの BMS 入力操作または最終端末制御に関連するアテンション識別子 (AID) が含まれます。 | Y |
| EIBATT | PIC X(1). | RU に付加ヘッダーデータが含まれることを示します (X'FF')。 | N |
| EIBCALEN | PIC S9(4) comp. | COMMAREA および LENGTH オプションを使用して、最後のプログラムからアプリケーションプログラムに渡された通信領域の長さが含まれます。通信領域が渡されない場合、このフィールドはゼロになります。 | Y |
| EIBCOMPL | PIC X(1). | 端末制御の RECEIVE コマンドで、この EIB エントリはデータが完了しているかを示します (X'FF')。RECEIVE コマンドで NOTRUNCATE オプションを使用すると、CICS は LENGTH または MAXLENGTH オプションによって要求された量を超えるデータを保持します。EIBRECV を設定すると、さらに RECEIVE コマンドが必要であることが示されます。EIBCOMPL は、データの最後を取り出すまで設定されません。<br>NOTRUNCATE オプションのない RECEIVE コマンドを実行するとき、EIBCOMPL が常に設定されます。 | Y |
| EIBCONF | PIC X(1). | APPC 会話に対して CONFIRM 要求が受け取られたことを示します (X'FF')。 | Y |
| EIBCPOSN | PIC S9(4) comp. | 表示デバイスからの BMS 入力操作または最終端末制御に関連するカーソルアドレス (位置) が含まれます。 | Y |

表 4-5　CICS EXEC インタフェースブロック (EIB) フィールドのサポート (2 / 3)

| EIB エントリ | 長さ/タイプ | 説明 | Sun MTP サポート |
|---|---|---|---|
| EIBDATE | PIC S9(7) comp-3. | タスクが開始される日付が含まれます。このフィールドは、ASKTIME コマンドによって更新されます。日付はパック 10 進数形式 (*0CYYDDD+*) で表記されます。このとき、C は世紀を表し、0 は 1900 年代、1 は 2000 年代に該当します。たとえば、1999 年 12 月 31 日および 2000 年 1 月 1 日を表す EIBDATE 値は、それぞれ 0099365 および 0100001 となります。 | Y |
| EIBDS | PIC X(8). | ファイル制御要求で参照される最後のデータセットの記号識別子が含まれます。 | Y |
| EIBEOC | PIC X(1). | 直前に受け取った RU に連鎖終了インジケータがあることを示します (X'FF')。 | Y |
| EIBERR | PIC X(1). | APPC 会話でエラーを受け取ったことを示します (X'FF')。 | Y |
| EIBERRCD | PIC X(4). | EIBERR が設定されていると、EIBERRCD には受け取ったエラーコードが含まれます。次の値は、EIBERRCD の最初の 2 バイトで返すことができます。<br>X'0889' 会話エラーが検出されます<br>X'0824' SYNCPOINT ROLLBACK が要求されます | Y |
| EIBFMH | PIC X(1). | 直前に受け取ったユーザーデータ、または取り出されたユーザーデータに FMS が含まれることを示します (X'FF')。 | Y |
| EIBFN | PIC X(2). | 最後の CICS コマンドがタスクによって実行されることを示すコードを表示します (要求された機能の完了で更新)。 | Y |
| EIBFREE | PIC X(1). | 機能を使用してアプリケーションプログラムを続行できないことを示します。CICS が機能を解放できるように、アプリケーションプログラムは機能を解放するか終了する必要があります (X'FF')。 | Y |
| EIBNODAT | PIC X(1). | 遠隔アプリケーションからデータが送信されていないことを示します (X'FF')。制御情報のみを転送した遠隔システムからのメッセージは受信しています。このフィールドの使用は、APPC リンクによって会話を行っているアプリケーションプログラムのみに限定されます。 | Y |
| EIBRCODE | PIC X(6). | CICS 応答コードが含まれます。このコードは、タスクが実行する最後の CICS コマンドによって要求された機能が完了したあとに返されます。<br>通常の応答では、このフィールドには 16 進数のゼロ (6 X'00') が含まれます。 | Y |

表 4-5　CICS EXEC インタフェースブロック (EIB) フィールドのサポート ( 3 / 3 )

| EIB エントリ | 長さ/タイプ | 説明 | Sun MTP サポート |
|---|---|---|---|
| EIBRECV | PIC X(1). | RECEIVE コマンドの実行により、アプリケーションプログラムが機能からのデータの受信を継続することを示します (X'FF')。 | Y |
| EIBREQID | PIC X(8). | CICS によって時間間隔制御コマンドに割り当てられた要求識別子が含まれます。このフィールドは、アプリケーションプログラムで要求識別子が指定されている場合には使用されません。 | Y |
| EIBRESP | PIC 9(09) comp. | 発生した RESP 条件に対応する番号が含まれます。 | Y |
| EIBRESP2 | PIC 9(09) comp. | RESP 条件の発生理由に関する詳細情報が含まれます。 | Y |
| EIBRLDBK | PIC X(1). | ロールバックを示します。 | N |
| EIBRSRCE | PIC X(8). | もっとも最近実行されたコマンドによりアクセスされたリソースの記号識別子が含まれます。 | Y |
| EIBSIG | PIC X(1). | SIGNAL が受け取られたことを示します (X'FF')。 | Y |
| EIBSYNC | PIC X(1). | アプリケーションプログラムが同期点を取るか、終了する必要があることを示します。このいずれかを実行する前に、そのアプリケーションプログラムが所有するその他の機能が送信状態または解放状態になっている必要があります (X'FF')。 | N |
| EIBSYNRB | PIC X(1). | アプリケーションプログラムが SYNCPOINT ROLLBACK コマンドを実行する必要があることを示します (X'FF')。このフィールドは、APPC または MRO リンクで会話しているアプリケーションプログラムにのみ設定されます。 | N |
| EIBTASKN | PIC S9(7) comp-3. | CICS によってタスクに割り当てられたタスク番号が含まれます。この番号は、タスクの制御中に生成されたトレーステーブルエントリに表示されます。このフィールドの形式はパック 10 進数です。 | Y |
| EIBTIME | PIC S9(7) comp-3. | タスクが開始される時刻が含まれます。このフィールドは、ASKTIME コマンドによって更新されます。この時刻は、パック 10 進数 (*0HHMMSS+*) 形式で表されます。 | Y |
| EIBTRMID | PIC X(4). | タスクに関連する基本機能 (端末または論理ユニット) の記号端末識別子が含まれます。 | Y |
| EIBTRNID | PIC X(4). | タスクの記号トランザクション識別子が含まれます。 | Y |
| EIBSEND | PIC X(1). | IBM フィールドではありません。Sun MTP によって 0x00 に設定されるのみです。 | |

第5章

# COBOLプログラムの使用

Sun MTPは、Server Express および ACUCOBOL-GT® ランタイム環境で、COBOL アプリケーションプログラムをサポートします。この章の内容は、次のとおりです。



両ベンダーのコンパイラは、ランタイムがロードし、要求時に実行される解釈的な形式でコンパイルされた出力を生成します。両方のランタイム環境とも、ソースレベルのデバッガとランタイムエラー情報も提供します。

UNIX 環境でメインフレームから移行される COBOL プログラムを使用する前に、copybooks などのプログラムとすべてのメンバーが正しく分析され、移行されていることを確認する必要があります。メインフレームアプリケーションプログラムの移行については、Sun Microsystems の担当者までお問い合わせください。

バッチプログラムのコンパイルについては、326 ページの「Server Express バッチプログラムのコンパイル」を参照してください。

## Server Express の使用

この節では、コマンド行インタフェースと Development System's Compilation メニューを使用した、Server Express プログラムの変換とコンパイルの方法について説明します。次のトピックがあります。



環境の設定については『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』、Server Express 領域を構築する方法については『Sun Mainframe Transaction Processing ソフトウェア インストールガイド』を参照してください。

# コマンド行インタフェースを使用したプログラムのコンパイル

コマンド行インタフェースを使用してオンラインプログラムをコンパイルする前に、kixclt トランスレータで変換する必要があります。プログラムの変換が終了したら、プログラムをコンパイルできます。コンパイルの際、通常の一連の命令を使用します。これらの命令は、コンパイルする COBOL プログラムのタイプ (OS/VS、COBOL II) によって異なります。一部の命令は、プログラムを正しくコンパイルするために Sun MTP で必要とされます。コンパイラオプションおよび命令については、Server Express のマニュアルを参照してください。

## プログラムの変換

kixclt は、Sun MTP の共通言語トランスレータです。CICS COBOL 文を COBOL 文に変換する前に、すべてのコピー、ヘッダー、インクルードファイルを検出して取り込みます。そのため、kixclt の実行前に COBCPY 環境変数を設定する必要があります。$COBCPY を使用して、COBOL コピーブック、マップセットコピーブック (存在する場合)、および Sun MTP コピーブックディレクトリの場所を指定します。次に例を示します。

```
export COBCPY=path-name/copy:path-name/maps:$UNIKIX/copy
```

myprog.clt という名前の OS/VS COBOL プログラムを変換して myprog.cbl という名前の出力ファイルを作成するには、次のコマンドを入力します。

```
$ kixclt myprog.clt
```

変換時にエラーが発生すると、kixclt がゼロでないリターンコードを返します。プログラムを修正し、もう一度変更する必要があります。

kixclt とそのオプションについては、『Sun Mainframe Transaction Processing 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。

## プログラムのコンパイル

すべてのオンライン COBOL プログラムをコンパイルするには、次の命令を使用します。

- defaultbyte"0"
- ibmcomp

COBOL II (.cl2) プログラムをコンパイルするには、次の命令を使用します。

- nocics

OS/VS COBOL (.clt) プログラムをコンパイルするには、次の命令を使用します。

- cics

次に、移行元のメインフレーム環境を基にする一般的な命令と、アプリケーションの要件を示します。

- arithmetic"osvs"
- notrunc
- perform-type"osvs"
- vsc2
- osvs

コンパイラの構文および必要なその他のコンパイラ命令については、Micro Focus のマニュアルを参照してください。

## コンパイルプログラムのためのスクリプトおよび Makefile の使用法

コマンド行で命令を入力するか、命令リストが保存されているファイルの名前を指定できます。

**例**: COBOL II プログラムをコンパイルする次のコマンドには、命令と 4 つのオプション (-iaPV) がコマンド行に記述されています。

```
$ cob -C 'arithmetic"osvs"' -C 'defaultbyte"0"' -C ibmcomp -C nocics -C notrunc
      -C vsc2 -iaPV myprog.cbl
```

**例**: OS/VS COBOL プログラムをコンパイルする次のコマンドには、命令が記述されている cobopts という名前のファイルと 4 つのオプション (-iaPV) がコマンド行に記述されています。

```
$ cob -C "directives=/path-name/cobopts" -iaPV myprog.cbl
```

また、1 つのプログラムまたは 1 つのディレクトリ内にあるすべてのプログラムをコンパイルするためのシェルスクリプトを作成できます。

コード例 5-1 は、1 つのプログラムをコンパイルするスクリプトを示す図です。COBOPT 環境変数が適切な命令ファイルに設定されている必要があります。

**コード例 5-1** 1 つのオンライン Server Express プログラムをコンパイルするスクリプト

```
for source
       do
       echo ---$source
       cob  -iaPV $source
       if [ $?-ne 0 ]
       then
           echo “Error in compilation of:“ $source
           exit
       fi
done
```

コード例 5-2 は、ディレクトリ内のすべてのオンラインプログラムを変換およびコンパイルするスクリプトです。このスクリプトでは、コンパイルのエラーまたは完了を示すログファイルの作成を指定しています。変換が終了しないと、プログラムはコンパイルされません。

**コード例 5-2** ディレクトリ内のすべてのオンライン Server Express プログラムをコンパイルするスクリプト

```
for i in `ls *.cl2`
do
    echo about to kixclt `basename $i .cl2`.cl2
    echo about to kixclt `basename $i .cl2`.cl2 >> COMPALL.err
    echo about to kixclt `basename $i .cl2`.cl2 >> COMPALL.log
    kixclt options `basename $i .cl2`.cl2
  ReturnCode=$?
  if [ $ReturnCode -ne 0 ]
    then
       echo "Error detected during kixclt of `basename $i .cl2`" >> COMPALL.err
  else
    echo about to compile `basename $i .cl2`.cbl
    echo about to compile `basename $i .cl2`.cbl >> COMPALL.err
    cob options and directives `basename $i .cl2`.cbl
    ReturnCode=$?
    if [ $ReturnCode -ne 0 ]
      then
          echo "Error detected during cob of `basename $i .cbl`" >> COMPALL.err
    else
        echo "Compile completed for `basename $i .cl2`" >> COMPALL.err
        rm `basename $i .cl2`.cbl
    fi
  fi
done
```

スクリプトファイルを使用する代わりに、makefile を使用して COBOL プログラムをコンパイルできます。makefile を使用する利点は、ソースファイルで変更されたプログラムだけを選択してコンパイルできることです。

---

**注 –** makefile を使用する前に、`make` 機能の概要を理解しておく必要があります。

---

コード例 5-3 は、makefile を最後に実行したあとでソースコードが変更されている COBOL プログラムを変換およびコンパイルする makefile の図です。

**コード例 5-3** Server Express プログラムをコンパイルする `makefile` (1 / 3)

```
# ********************************************
# CICS Application Programming Primer
# Enter into Catalog:
# dataset     reclen     keylen  type
# ACCTFIL     383        5       KSDS
# ACCTIX      63         17      KSDS
# ********************************************

# ********************************************
# rules
# ********************************************

.SUFFIXES:.clt .cbl .int .gnt .bms .map

.clt.cbl:
       rm -f $*.cbl
       ( kixclt $*.clt )

.cbl.int:
       rm -f $*.int $*.idy
       cob -ia $(COBFLAGS) $<

.cbl.gnt:
       rm -f $*.gnt
       cob -u $(COBFLAGS) $<

.bms.map:
       rm -f $*.map $*.err $*.lst
       kixbms  -a $<
```

コード例 5-3　　Server Express プログラムをコンパイルする makefile (2 / 3)

```
# ********************************************
# variables to rules
# ********************************************

# Cobflags for CLT modules
COBFLAGS=-C "ibmcomp cics noalter osvs defaultbyte=0"

# Cobflags for CL2 modules
COBFLAGS1=-C "ibmcomp noalter osvs defaultbyte=0"

# ********************************************
# standard variables
# ********************************************

# ********************************************
# standard targets
# ********************************************

all:        $(INTTARGETS)

INTTARGETS: ACCT00.int ACCT01.int ACCT02.int ACCT03.int \
            ACCT04.int

GNTTARGETS: ACCT00.gnt ACCT01.gnt ACCT02.gnt ACCT03.gnt \
            ACCT04.gnt

clean:
       rm -f *.gnt
       rm -f *.int
       rm -f *.idy
```

コード例 5-3　Server Express プログラムをコンパイルする makefile（3 / 3）

```
# ***********************************************
# Targets with dependencies
# ***********************************************
ACCT00.int:   ACCT00.clt     ACCT00.cbl

ACCT01.int:   ACCT01.clt     ACCT01.cbl

ACCT02.int:   ACCT02.clt     ACCT02.cbl

ACCT03.int:   ACCT03.clt     ACCT03.cbl

ACCT04.int:   ACCT04.clt     ACCT04.cbl

ACCT00.gnt:   ACCT00.clt     ACCT00.cbl

ACCT01.gnt:   ACCT01.clt     ACCT01.cbl

ACCT02.gnt:   ACCT02.clt     ACCT02.cbl

ACCT03.gnt:   ACCT03.clt     ACCT03.cbl

ACCT04.gnt:   ACCT04.clt     ACCT04.cbl
```

# コンパイルメニューを使用したプログラムのコンパイル

開発システムのコンパイルメニューを使用して、Server Express プログラムをコンパイルすることもできます。このメニューのオプションを使用すると、ファイルの編集、copybook ディレクトリの指定、およびコンパイラオプションの設定が可能になります。

コンパイルメニューを使用する場合は、コンパイルされる前に kixclt トランスレータにより、プログラムが自動的に変換されます。

## ▼ コンパイルするファイルを選択する

1. **開発システムメインメニューで PF4 キーを押します。**

   図 5-1 に示されているようなコンパイルメニューが表示されます。標準のファイル選択画面は、Sun MTP の画面に似ています。

2. **「Directory」フィールドに、COBOL プログラムを含むディレクトリのパス名を入力します。**

   最初のエレメントまたはパス全体の名前を環境変数で示すことができます。

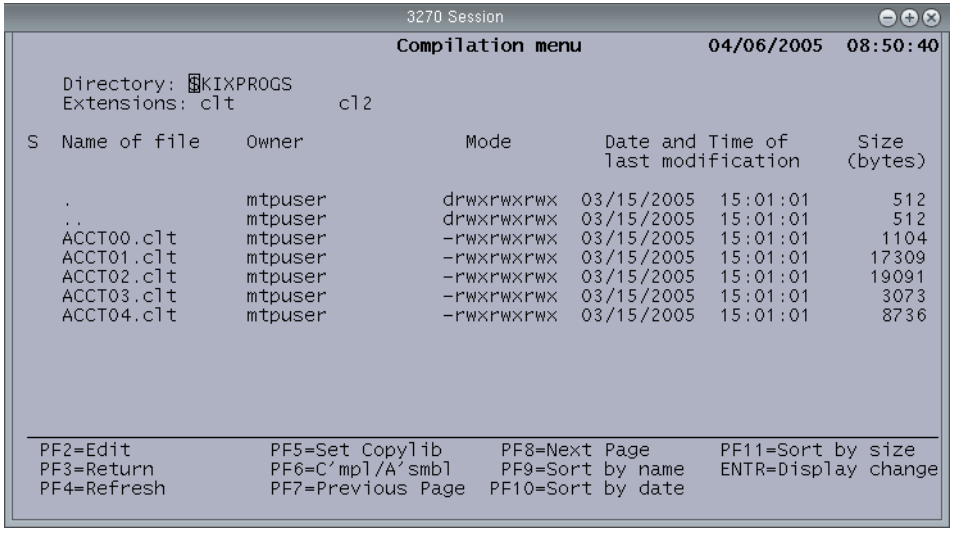


**図 5-1** Compilation メニュー

3. **「Extensions」フィールドには、ファイル拡張子または表示する COBOL プログラムの拡張子を入力します。**

   有効な COBOL 拡張子は次のとおりです。

   `.clt`: IBM OS/VS COBOL コンパイラ標準に準拠。CICS プログラムであると見なされます。

   `.cl2`: IBM COBOL II コンパイラ標準に準拠。CICS プログラムであると見なされます。

   `.cbl`: `kixclt` トランスレータによってすでに変換されたバッチプログラムまたはオンラインプログラム。

   ファイル情報フィールドについては、表 2-2 を参照してください。

4. **Enter キーを押します。**

   画面が更新され、指定したディレクトリおよびファイルタイプを表示します。

5. **ファイル名の左にある選択フィールドにカーソルを移動してファイルを選択し、ファイル名の横に大文字または小文字の `S` を入力します。**

   複数ファイルの選択が可能。

---

**注 –** Server Express ファイル名拡張子と Open PL/I ファイル名拡張子を 1 つの選択セットで混在させないでください。混在させると、エラーメッセージ「`Illegal file extension for compile`」が表示されます。

---

6. **実行するアクションのファンクションキーを押します。**

   - PF2: `kixed` シェルスクリプトを実行して、選択したファイルを開きます。デフォルトでは `vi` エディタを使用します。複数のファイルを選択した場合、最後に選択したファイルを最初に、最初に選択したファイルを最後に編集します。`kixed` スクリプトは、各ファイルに対して自動的に起動されます。
   - PF5: copybook ディレクトリを指定できる、Copylib の設定画面を表示します。詳細は、図 5-2 を参照してください。
   - PF6: コンパイラオプションの設定画面を表示します。詳細は、図 5-3を参照してください。

   次の表に、コンパイルメニューのその他のファンクションキーを示します。

**表 5-1** コンパイルメニューのファンクションキー

| ファンクションキー | 動作 |
|---|---|
| PF3 | コンパイルメニューを閉じて、開発システムメインメニューに戻ります。 |
| PF4 | 画面のファイルリスト部分のデータを更新します。ファイルの属性を変更したあと、PF4 キーを押して画面を更新します。 |
| PF7 | ファイルの前のページを表示します。ファイルの最初のページが表示されているときにこのキーを押すと、次のメッセージが表示されます。<br>`Beginning of data reached` |
| PF8 | ファイルの次のページを表示します。ファイルの最後のページが表示されているときにこのキーを押すと、次のメッセージが表示されます。<br>`End of data reached` |
| PF9 | ディレクトリの内容をファイル名順にソートして再表示します。 |
| PF10 | ディレクトリの内容をファイルの変更日の降順にソートして再表示します。 |
| PF11 | ディレクトリの内容をファイルサイズの降順にソートして再表示します。 |
| Enter | ファイルが選択されていない場合には、何も実行されません。ファイルを選択すると、コンパイラ画面が表示されます。<br>ディレクトリを選択すると、ディレクトリが選択したディレクトリに変更され、ディレクトリ内のファイルが表示されます。 |

## Copybook ディレクトリの指定

Copylib の設定画面では、COBOL ソースコードで参照されるコピーブックを検索するディレクトリを最大 8 つまで指定できます。ディレクトリは、次の順序で検索されます。

1. 現在のディレクトリ
2. Copylib の設定画面で定義されたディレクトリ
3. COBCPY 環境変数で定義されたディレクトリ

---

**注 –** 領域セットアップファイルで COBCPY 環境変数をすでに設定して、copybook ディレクトリをポイントしている場合は、この画面で入力を行う必要はありません。

---

COBCPY 環境変数については、Micro Focus のマニュアルを参照してください。

## ▼ Copybook ディレクトリを指定する

1. **コンパイルメニューで PF5 キーを押します。**

   図 5-2 に示す画面が表示されます。ディレクトリ名用の 8 つのフィールドがあります。

2. **ディレクトリを入力します。**

   名前の先頭には、ドル記号 ($) で示された環境変数も指定できます。環境変数のあとは、スラッシュ (/) で区切ってその他のサブディレクトリを続けられます。

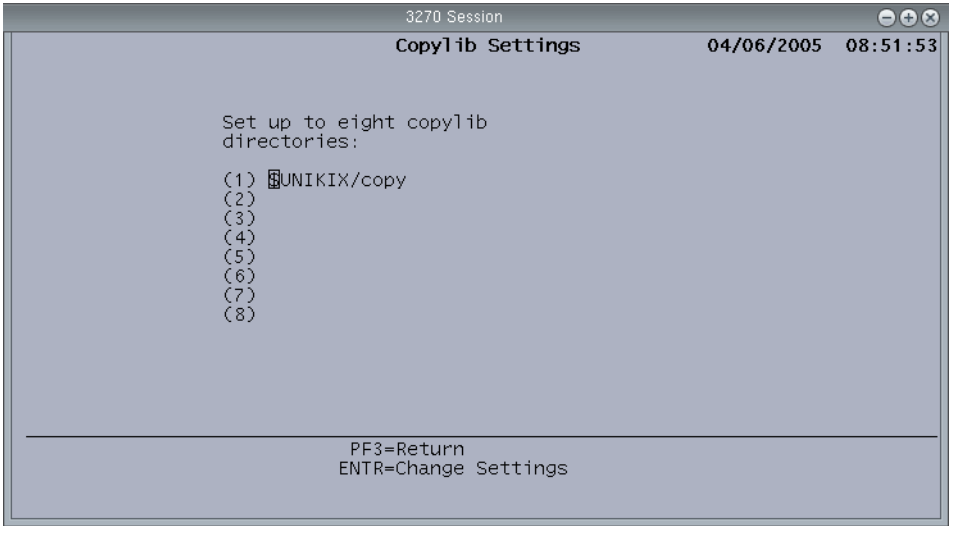


**図 5-2** Copylib Settings 画面

3. Enter キーを押して、設定を保存します。

他のユーザー指定オプションと同様、この画面で実行する Copylib の設定は `$KIXSYS/`*user-name*`.tbl` ファイル内に記述されています。*user-name* はユーザーのログイン名を示します。オプションは、セッション間で保存されます。

## プログラムのコンパイル

プログラムで必要な編集を完了し、copybook ディレクトリを設定したら、プログラムのコンパイル準備が整います。

### ▼ プログラムをコンパイルする

1. コンパイルメニュー (図 5-1) で COBOL プログラムファイルを選択します。
2. PF6 キーを押して「Set Compiler Options」画面を表示します。

この画面では、コンパイラの処理をカスタマイズできます。

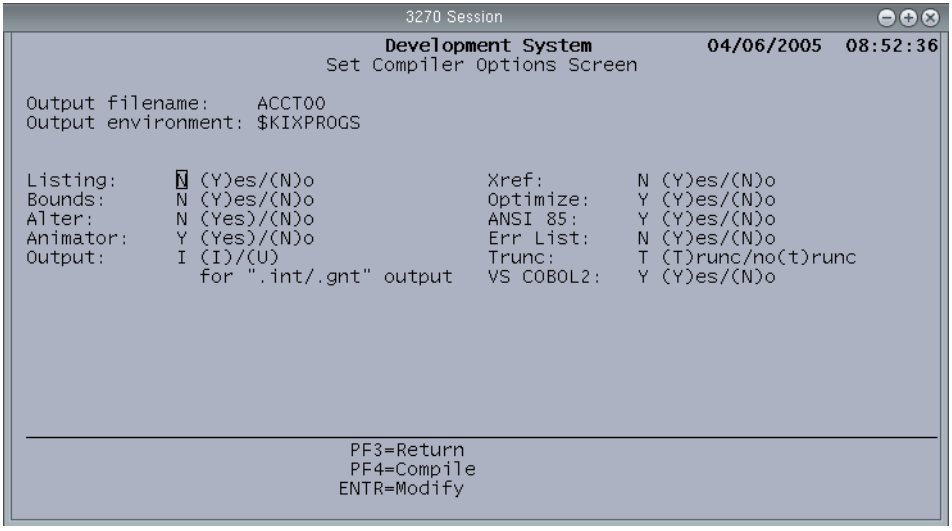


図 5-3　Set Compiler Options Screen (COBOL)

3. **必要に応じて、以下の画面の値を変更します。**

| | |
|---|---|
| Output filename | 現在選択しているファイルの名前がデフォルトとして表示されます。ファイル名は最大 14 文字です。 |
| Output environment | コンパイルメニュー画面のディレクトリ名がデフォルトとして表示されます。このフィールドの最初の要素を環境変数にして、スラッシュで区切ってサブディレクトリをあとに続けることができます。 |

一部のコンパイラオプションおよび命令は、選択したファイルの拡張子に基づいて自動的に設定されます。それ以外の値は、画面上でデフォルトの値を変更して設定します。

ファイル拡張子に基づいて設定されるオンラインプログラムのオプションおよび命令は、次のとおりです。

`.clt` では `-v ibmcomp perform-type=osvs defaultbyte=0 cics`

`.cl2` では `-v ibmcomp perform-type=osvs defaultbyte=0 cics`

ファイル拡張子に基づいて設定されるバッチプログラムのオプションおよび命令は、次のとおりです。

`.cbl` では `-v ibmcomp perform-type=osvs defaultbyte=32`

| 指示 | 説明 |
|---|---|
| `-v` | コンパイルに関する情報を表示する冗長オプションを設定します。 |
| `ibmcomp` | メインフレームと互換性のあるサイズでデータを割り当てます。 |
| `perform-type=osvs` | コンパイラは `PERFORM` 文をメインフレームで使用されるのと同じ方法で処理します。 |
| `defaultbyte` | 記憶域が初期化される方法を制御します。記憶域は COBOL プログラムで宣言され、それに関連する VALUE 文はありません。オンラインプログラムでは、この命令をゼロに設定して、記憶域が低い値に設定されるようにします。バッチプログラムでは、この命令を 32 に設定して、記憶域がスペースに設定されるようにします。 |
| `cics` | OS/VS COBOL プログラムで BLL セルを使用できるようにします。 |

「Set Compiler Options」画面のオプションおよび命令を、対応する Server Express 命令とともに次の表に示します。命令の使用については、Server Express のマニュアルを参照してください。

**表 5-2** Server Express コンパイラのオプション

| 画面オプション | Server Express 命令/オプション |
|---|---|
| Listing | `[NO]LIST` |
| Bounds | `[NO]BOUND` |

表 5-2　Server Express コンパイラのオプション (続き)

| 画面オプション | Server Express 命令/オプション |
|---|---|
| Alter | [NO]ALTER |
| Animator | -a |
| Output | -i (中間コード) /-u (ネイティブコード) |
| Xref | [NO]XREF |
| Optimize | [NO]CICSOPT |
| ANSI 85 | [NO]ANS85 |
| Err List | [NO]ERRLIST |
| Trunc | [NO]TRUNC |
| VS COBOL2 | [NO]VSC2 |

「Set Compiler Options」画面で設定したこれらのオプションおよび命令は、その他のユーザー指定の命令とともに、$KIXSYS/*user-name*.tbl ファイルに保存されます。*user-name* はユーザーのログイン名を示します。これらの命令は、セッション間で保存されます。

その他の命令は、$COBDIR/cobopt ファイルに保存できます。また、$UNIKIX/bin/kixcob シェルスクリプトを変更することにより、デフォルトの COBOL 命令を変更できます。

4. **「Set Compiler Options」画面で PF4 を押して、COBOL コンパイラを呼び出す kixcob シェルスクリプトを実行します。**

.clt または .cl2 プログラムを選択すると、それらは変換され、コンパイルされます。.cbl プログラムを選択すると、それらはコンパイルされますが、変換されません。ただし、.cbl拡張子を持つオンラインプログラムを選択すると、ファイルがコンパイルされますが、変換されません。.cbl 拡張子を持つオンラインプログラムは、kixclt トランスレータからの出力です。

複数のファイルを選択すると、ファイルは選択した順番と逆の順番で、最後に選択したファイルが最初にコンパイルされます。

コンパイルが完了すると、コンパイルおよび変換の結果がデフォルトのエディタ (vi) に表示されます。

## ネイティブコードへのプログラムのコンパイル

Server Express プログラムを実行できるようにするには、プログラムを本番稼動システムの .gnt モードにコンパイルする必要があります。.gnt ファイルはネイティブコードでコンパイルされるので、.int よりも早く実行できます。

テスト環境で Micro Focus Animator を使用する場合は、アプリケーションのオプションについて Server Express のマニュアルを参照してください。

# ACUCOBOL-GT の使用

この節では、コマンド行インタフェースを使用した、ACUCOBOL-GT プログラムの変換およびコンパイル方法について説明します。

---

**注 –** 開発システムのコンパイルメニューはサポートされていません。

---

この節の内容は、次のとおりです。


- 136 ページの「コマンド行インタフェースを使用したプログラムのコンパイル」
- 153 ページの「デバッグ用の ACUCOBOL-GT プログラムのコンパイル」


バッチプログラムのコンパイルについては、326 ページの「Server Express バッチプログラムのコンパイル」を参照してください。

環境の設定については『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』、ACUCOBOL-GT 領域を構築する方法については『Sun Mainframe Transaction Processing ソフトウェア インストールガイド』を参照してください。

## コマンド行インタフェースを使用したプログラムのコンパイル

コマンド行インタフェースを使用してオンラインプログラムをコンパイルする前に、kixclt トランスレータで変換する必要があります。プログラムの変換が終了したら、プログラムをコンパイルできます。コンパイルの際、通常の一連のオプションを使用します。一部のオプションは、プログラムを正しくコンパイルするために Sun MTP で必要とされます。すべてのコンパイラオプションについては、ACUCOBOL-GT のマニュアルを参照してください。

## プログラムの変換

kixclt は、Sun MTP の共通言語トランスレータです。CICS COBOL 文を COBOL 文に変換する前に、すべてのコピー、ヘッダー、インクルードファイルを検出して取り込みます。そのため、kixclt の実行前に COPYPATH 環境変数を設定する必要があります。$COPYPATH を使用して、COBOL コピーブック、マップセットコピーブック (存在する場合)、および Sun MTP コピーブックディレクトリの場所を指定します。次に例を示します。

```
export COPYPATH=path-name/maps:$UNIKIX/copy
```

myprog.cl2 という名前のプログラムを変換して myprog.cbl という名前の出力ファイルを作成するには、次のコマンドを入力します。

```
$ kixclt myprog.cl2
```

変換時にエラーが発生すると、kixclt がゼロでないリターンコードを返します。プログラムを修正し、もう一度変更する必要があります。

kixclt とそのオプションについては、『Sun Mainframe Transaction Processing 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。

## プログラムのコンパイル

コンパイラの構文および必要なその他のコンパイラ命令については、『ACUCOBOL-GT ユーザーズガイド』 (Book 2) というマニュアルを参照してください。

オンライン COBOL II プログラムをコンパイルするには、次のオプションを使用します。

- -Cv: IBM DOS/VS COBOL 関数を受け入れるように、コンパイラを設定します。
- -Dv=0: 新しくロードされたプログラムのデータ部分を入力するために使用するデフォルトバイトを設定します。
- -Fm: LOCK 節が FILE-CONTROL 段落のエントリから省略される場合は、デフォルトの LOCK MODE IS MANUAL に設定します。

Server Express プログラムから ACUCOBOL-GT に移行する場合は、--TruncANSI オプションを使用します。これは、COMP-5 項目に割り当てられたストレージの容量に、バイナリを切り捨てます。

ACUCOBOL-GT 環境変数 CBLFLAGS で使用するオプションを指定できます。次に例を示します。

```
export CBLFLAGS="-Cv -Dv=0"
```

コンパイラの -x オプションを使用して、特定のコンパイルで $CBLFLAGS を無視できます。

## コンパイルプログラムのためのスクリプトおよび Makefile の使用法

コマンド行で命令を入力するか、命令リストが保存されているファイルの名前を指定できます。

**例**: COBOL II プログラムをコンパイルする次のコマンドには、命令がコマンド行に記述されています。

```
$ ccbl -Cv -Dv=0 -v -e myprog.err -Ls -Lc myprog.cbl
```

また、1 つのプログラムまたは 1 つのディレクトリ内にあるすべてのプログラムをコンパイルするためのシェルスクリプトを作成できます。

コード例 5-4 は、1 つのプログラムをコンパイルするスクリプトを示す図です。

**コード例 5-4** 1 つのオンライン ACUCOBOL-GT プログラムをコンパイルするスクリプト

```
for source
       do
       echo ---$source
       ccbl -Dv=0 -Cv -v $source
       if [ $?-ne 0 ]
       then
           echo “Error in compilation of:“ $source
           exit
       fi
done
```

コード例 5-5 は、ディレクトリ内のすべてのオンラインプログラムを変換およびコンパイルするスクリプトです。このスクリプトでは、コンパイルのエラーまたは完了を示すログファイルの作成を指定しています。変換が終了しないと、プログラムはコンパイルされません。

**コード例 5-5** ディレクトリ内のすべてのオンライン ACUCOBOL-GT プログラムをコンパイルするスクリプト

```
for i in `ls *.cl2`
do
    echo about to kixclt `basename $i .cl2`.cl2
    echo about to kixclt `basename $i .cl2`.cl2 >> COMPALL.err
    echo about to kixclt `basename $i .cl2`.cl2 >> COMPALL.log
    kixclt options `basename $i .cl2`.cl2
  ReturnCode=$?
  if [ $ReturnCode -ne 0 ]
    then
       echo "Error detected during kixclt of `basename $i .cl2`" >> COMPALL.err
  else
    echo about to compile `basename $i .cl2`.cbl
    echo about to compile `basename $i .cl2`.cbl >> COMPALL.err
    kixclt options `basename $i .cl2`.cl2
    ReturnCode=$?
    if [ $ReturnCode -ne 0 ]
      then
         echo "Error detected during ccbl of `basename $i .cbl`" >> COMPALL.err
    else
        echo "Compile completed for `basename $i .cl2`" >> COMPALL.err
        rm `basename $i .cl2`.cbl
    fi
  fi
done
```

スクリプトファイルを使用する代わりに、makefile を使用して COBOL プログラムをコンパイルできます。makefile を使用する利点は、ソースファイルで変更されたプログラムだけを選択してコンパイルできることです。

---

**注 –** makefile を使用する前に、`make` 機能の概要を理解しておく必要があります。

---

コード例 5-6 は、makefile を最後に実行したあとでソースコードが変更されている ACUCOBOL-GT プログラムを変換およびコンパイルする makefile の図です。

**コード例 5-6** ACUCOBOL-GT プログラムをコンパイルする `makefile` (1 / 2)

```
# ********************************************
# CICS Application Programming Primer
# Enter into Catalog:
# dataset     reclen     keylen  type
# ACCTFIL     383        5       KSDS
# ACCTIX      63         17      KSDS
# ********************************************

# ********************************************
# rules
# ********************************************

.SUFFIXES:.clt .cbl .int .gnt .bms .map

.clt.cbl:
       rm -f $*.cbl
       ( kixclt $.cl2 )

.cbl.acu:
       rm -f $*.cbl
       ccbl -Dv=0 -Cv $<

.bms.map:
       rm -f $*.map $*.err $*.lst
       kixbms  -a $<

# ********************************************
# variables to rules
# ********************************************


# ********************************************
# standard variables
# ********************************************
```

コード例 5-6　　ACUCOBOL-GT プログラムをコンパイルする makefile (2 / 2)

```
# ********************************************
# standard targets
# ********************************************

all:        $(INTTARGETS)

ACUTARGETS: ACCT00.acu ACCT01.acu ACCT02.acu ACCT03.acu \
            ACCT04.gnt

clean:
       rm -f *.acu

# ********************************************
# Targets with dependencies
# ********************************************
ACCT00.acu:   ACCT00.cl2     ACCT00.cbl

ACCT01.acu:   ACCT01.cl2     ACCT01.cbl

ACCT02.acu:   ACCT02.cl2     ACCT02.cbl

ACCT03.acu:   ACCT03.cl2     ACCT03.cbl

ACCT04.acu:   ACCT04.cl2     ACCT04.cbl
```

# デバッグ用の ACUCOBOL-GT プログラムのコンパイル

ACUCOBOL-GT プログラムをデバッグ用にコンパイルする必要がある場合は、すべてのデバッグコンパイラオプションの説明について、ACUCOBOL-GT のマニュアルを参照してください。ccbl -help コマンドを入力して、コンパイランのオプションリストを表示することもできます。

次の例は、ソースレベルのデバッグで仲介コードを生成する簡単なコマンドを示します。

```
$ ccbl -Ga ACCT00.cbl
```

# 第6章

# PL/I 共有ライブラリの使用法

Sun MTP では、PL/I アプリケーションの実行に必要な Liant Open PL/I および共有ライブラリ機能をサポートしています。この章では、PL/I 共有ライブラリの使用法について説明します。

`$UNIKIX/bin` にある `kixplt` プリプロセッサのバージョンを使用していることを確認します。`kixinstall` 構成ユーティリティーを使用して構築された makefile は、`$UNIKIX/bin` 内の `kixplt` を自動的に検索します。すべてのユーザー作成スクリプトおよび makefile も、`$UNIKIX/bin` 内の `kixplt` を検索します。

この章の内容は、次のとおりです。



PL/I デバッガの CodeWatch については、315 ページの「PL/I プログラムのデバッグ」を参照してください。

また、docs.sun.com の『Linker and Libraries Guide』も参照してください。このマニュアルには、共有オブジェクトの構築と保守に関する詳細情報が記載されています。

# 共有ライブラリの設定

すべての PL/I アプリケーションプログラムは、UNIX 共有ライブラリ機能を使用して事前に構築された 1 つまたは複数の共有ライブラリから実行する必要があります。詳細は、162 ページの「Makefile を使った共有ライブラリの構築」を参照してください。

各 PL/I プログラムおよびそれらが存在する関連の共有ライブラリは、処理プログラムテーブル (PPT) の領域に一致する必要があります。PPT については、『Sun Mainframe Transaction Processing 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。

## 共有ライブラリの命名

共有ライブラリを命名する場合、次の要件を満たす名前にする必要があります。

- 環境変数 `KIXLIB` を定義していないと、PPT の共有ライブラリのエントリはすべて、`$KIXSYS` ディレクトリからの相対パスとなります。`$KIXLIB` を設定すると、この環境変数に共有ライブラリを検索するディレクトリを指定するパス名のリストが含まれます。環境変数の設定については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。
- 共有ライブラリ名の PPT エントリは、サブディレクトリ名を含め、拡張子 `.so` を含めずに最大 16 文字です。これにより、`$KIXSYS` または `$KIXLIB` より下位レベルのサブディレクトリの長さと数が制限されます。
- 共有ライブラリ名は小文字。
- 実際の共有ライブラリは拡張子 `.so` で構築されるが、PPT では拡張子 `.so` を使用しません。ライブラリを割り当てるとき、Sun MTP が PPT から取り出すライブラリ名にこの拡張子を付加します。
- パスの一部が異なっていれば、2 つのライブラリに同じ名前を付けることができます。

たとえば、次のライブラリ名は有効です。

`lev1/lev2/myshlb`: PPT エントリの最大長 (Sun MTP では `.so` は名前の一部ではない)。このエントリには、共有ライブラリの前に 2 つのサブディレクトリがあります。

`myshlb`: サブディレクトリを持たない共有ライブラリ。

# 共有ライブラリのオープン

PPT で一覧表示された各共有ライブラリは、それぞれの指定されたトランザクションに対応します。個々の共有ライブラリは、対応するトランザクションが呼び出されるまでオープンされません。共有ライブラリをオープンするこの方法では、呼び出された共有ライブラリのみをオープンすることによって起動時のオーバーヘッドを低く抑えます。ただし、共有ライブラリが一度オープンされると、トランザクションが終了してもアンロード (すなわちクローズ) されません。同じトランザクションの次の呼び出しのために、利用できる状態をそのまま維持するからです。

アプリケーション共有ライブラリのロードアルゴリズムは、対応するプログラムの呼び出しに基づいているので、すべての記号への参照の依存性は、特定の共有ライブラリを構築するときにリンクラインで宣言する必要があります (157 ページの「共有ライブラリ構築のためのリンクラインの例」を参照)。これを行わないと、動的ローダーではこれらの記号を解決に利用できません。

---

**注 –** 共有ライブラリが依存性を持たない場合、変更を行う必要はありません。

---

動的ローダーが依存性のある記号を利用できるようにする別の方法として、PPT エントリのプリロード (P/L) フィールドを共有ライブラリに対して `Y` に設定します。このフィールドを `Y` に設定すると、起動時にトランザクションサーバーはすべての「プリロード」された共有ライブラリをオープンします。PPT の各共有ライブラリでプリロードフィールドが `Y` に設定されると、その領域が起動するときにすべての共有ライブラリがロードされます。

---

**注 –** 共有ライブラリの構築時にリンクラインで依存性を宣言する必要も、実行時に共有ライブラリをオープンするための「pre-load」フラグを設定する必要もありません。

---

## 共有ライブラリ構築のためのリンクラインの例

次のリンクラインの例は、依存性のある `prog1.so` という名前の共有ライブラリを、`libmoreso.so` と呼ばれる別の共有ライブラリに構築する例です。

```
$ ld -G -Bdynamic  -o prog1.so prog1.o -R $KIXLIB/mysos1 \
-L $KIXLIB/mysos1 -lmoreso
```

---

**注 –** 各引数に固有の事項については、`ld`(1) コマンドのマニュアルページを参照してください。

---

## PPT プリロードフィールド

各 PPT エントリには、「P/L」というラベルの 1 文字のフィールドがあります。共有ライブラリがプリロードされるかどうかを示す場合、このフィールドを Y に設定する必要があります。開く必要がない場合は、このフィールドを N に設定するか、空白のままにします。

図 6-1 は、「P/L」フィールドが Y に設定されている共有オブジェクトが 1 つある PPT メイン画面です。

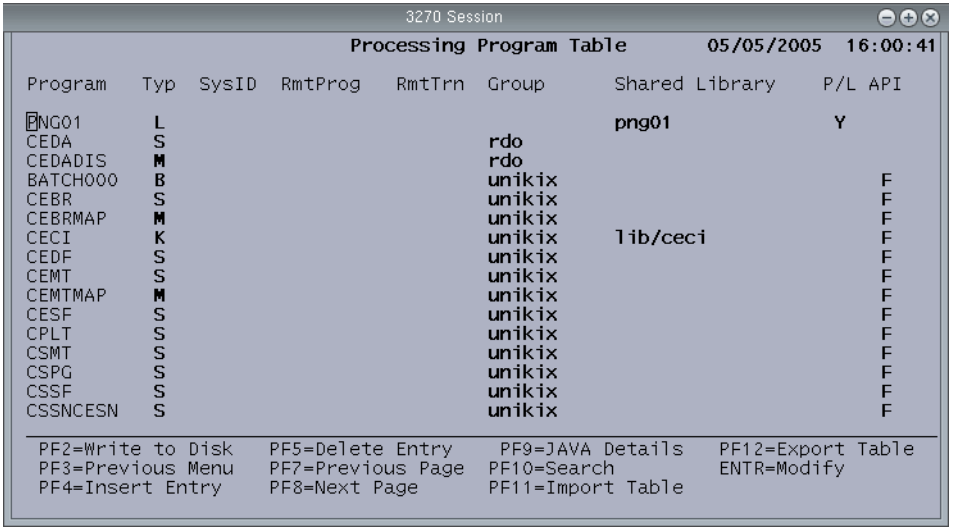


**図 6-1** PPT—Shared Library エントリ

# 共有ライブラリでの LOAD PROGRAM ENTRY の使用法

メインフレームでは、EXEC CICS LOAD PROGRAM ENTRY 文は要求されたプログラムのエントリアドレスを返します。ユーザーアプリケーションは、このアドレスをあとで使用できるように保存したり、プログラムをただちに呼び出したりできます。

ただし、Sun MTP では、コマンドは異なる動作をします。あるトランザクション処理プログラムで取得される共有ライブラリポインタは、あとで別のトランザクション処理プログラムで使用できない場合があります。これは UNIX 共有ライブラリの実装によるものです。擬似会話型トランザクションは、トランザクション中に 1 つまたは複数のトランザクション処理プログラムによって実行される場合があります)。この実装では、EXEC CICS LOAD PROGRAM ENTRY 文によって返されるポインタは、

特定のプロセス ID (すなわち、トランザクション処理プログラム) が必要なプログラムを呼び出すときに、「実際の」共有エントリポインタに変換される必要のある共有メモリーポインタです。返されるこのアドレスは、あとで使用できるように保存されます。ただし、エントリアドレスとして使用する前に、例に示すように KXSYM2FUNC という名前の特殊関数によって変換される必要があります。

コード例 6-1 およびコード例 6-2 では、LOAD コマンドの ENTRY オプションを示しています。省略されたコードは、省略記号 (...) で示しています。これらの例は、異なる Sun MTP トランザクションサーバーのプロセスで実行される別々の PL/I プロシージャーにある場合がありますが、LOAD は KXSYM2FUNC が呼び出される前に実行される必要があります。

次の例のコードでは、プロシージャーへのポインタ (ユーザーインタフェース、データ編集、またはデータベースアクセスを処理するルーチン) を含むグローバルテーブルの記憶領域を取得します。ロードされたプロシージャー MYPROC は、PPT のエントリを持っている必要があります。

**コード例 6-1**　　グローバルテーブルの記憶領域の取得

```
...
DCL TABLE_PTRPOINTER;
DCL 1 TABLE BASED(TABLE_PTR),
       2 ...
       2 PROC_PTRPOINTER;
       ...
EXEC CICS GETMAIN
           LENGTH(STG(TABLE))
           SET(TABLE_PTR)
           SHARED;
...
EXEC CICS LOAD
           PROGRAM('MYPROC')
           ENTRY(TABLE.PROC_PTR)
           HOLD;
...
```

コード例 6-2 のコードは、TABLE からプロシージャーポインタをフェッチして、それを現在の Sun MTP トランザクションサーバープロセスのアドレススペースに割り当てるために KXSYM2FUNC を呼び出し、さらにプロシージャーを呼び出します。

**コード例 6-2** KXSYM2FUNC 関数の呼び出し

```
       ...
DCL KXSYM2FUNC         ENTRY(POINTER VALUE) RETURNS(POINTER)
                       EXTERNAL('kxsym2func');
DCL NULL               BUILTIN;
DCL PROC               ENTRY VARIABLE;
DCL 1      PROC_OVERLAY        DEFINED(PROC),
           2 ENTRY_PTR         POINTER,
           2 DISPLAY_PTR       FIXED BIN(31);
           ...
PROC_OVERLAY.ENTRY_PTR   = KXSYM2FUNC(TABLE.PROC_PTR);
PROC_OVERLAY.DISPLAY_PTR = 0;
IF PROC_OVERLAY.ENTRY_PTR = NULL THEN
       /*
       *  Error!Was there a corresponding LOAD?
       *          Did a condition occur during that LOAD?
       *
       */
       ...
ELSE
       CALL PROC;
       ...
```

# 共有ライブラリでの CEMT の使用

CEMT SET および CEMT INQ へのオプションでは、プログラムの共有ライブラリの動的な変更および現在のライブラリ名の照会ができます。詳細は、161 ページの「同じ共有ライブラリの異なるバージョンの実行」を参照してください。

CEMT SET PROGRAM トランザクションへの LIBRARY オプションでは、領域の実行中にプログラムの共有ライブラリを動的に変更できます。

形式:

```
CEMT S[ET] PROG[RAM] prog-name LIB[RARY] lib-name PROG[RAM] prog-name
       LIB[RARY] blankreset
```

説明

| | |
|---|---|
| *prog-name* | プログラム名。最大 8 文字です。 |
| *lib-name* | 共有ライブラリ名。最大 16 文字の小文字です。 |
| `blankreset` | プログラムの PPT 共有ライブラリフィールドを空白に設定することにより、指定されたプログラムを共有ライブラリ処理から削除します。 |

**注 –** `blankreset` は、ここに記載されているとおりに正確に入力する必要があります。間違いがあると、新しい共有ライブラリ名として扱われます。

`CEMT` コマンドを実行したあと、`CINI` トランザクションを実行して新しい共有ライブラリおよび内部テーブルを初期化します。

`CEMT INQ PROGRAM` トランザクションの `LIBRARY` オプションを使用して、現在の共有ライブラリ名を照会します。

形式:

```
CEMT I[NQ] PROG[RAM] prog-name LIB[RARY]
```

*prog-name* は、最大 8 文字のプログラム名です。

プログラムに共有ライブラリがある場合、次のメッセージが表示されます。

`KIX1584I CEMT transaction terminated ShrLib =` *lib-name*

プログラムに共有ライブラリがない場合、次のメッセージが表示されます。

`KIX1584I CEMT transaction terminated ShrLib = NoSharedLibPrsnt`

# 同じ共有ライブラリの異なるバージョンの実行

Sun MTP では、領域を再起動しないで共有ライブラリの新しいバージョンをロードできます。たとえばアプリケーションが、PPT で `dir1/prog_pay1` として定義されている `prog_pay1.so` という名前の共有ライブラリを使用するとします。実行中、`prog_pay1.so` が期待通りに動作しないことが判明したので、`prog_pay1.so` の新しいバージョンをコンパイルおよび再構築します。領域を再起動して新しいバージョンを得るのではなく、異なるディレクトリ `dir2` に共有ライブラリの新しいバージョンを保存します。

新しい共有ライブラリを実行するために、次のトランザクションを使用します。

```
CEMT SET PROGRAM MY_PROG1 LIBRARY dir2/prog_pay1
```

このトランザクションによって、共有ライブラリが配置される新しいディレクトリを指定するために、共有メモリー PPT が変更されます。この変更は、領域が存在する間有効です。パス名は最大 16 文字で、`$KIXLIB` または `$KIXSYS` ディレクトリからの相対パスです。PPT での共有ライブラリの定義については、『Sun Mainframe Transaction Processing 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。

次のトランザクションでは、トランザクション処理プログラムによって共有ライブラリを再オープンし、それを利用できるようにします。

```
CEMT SET PROGRAM MY_PROG1 LIBRARY NEWCOPY
```

**注 –** 共有ライブラリの再コンパイルおよび再構築だけ行って、以前のバージョンを上書きしないでください。一部のプラットフォームにはこの方法を適用できますが、その他のプラットフォームでは予期しないエラーが発生することがあります。

# Makefile を使った共有ライブラリの構築

`$UNIKIX/examples/primer/pli` ディレクトリにあるサンプルの makefile は、共有ライブラリの構築方法を示します。この makefile は、サンプルの Primer アプリケーションのために、`ACCT` トランザクションの PL/I バージョンをコンパイルします。`$UNIKIX/examples/primer/pli` ディレクトリにある readme ファイルは、サンプルアプリケーションの実行方法を示します。

サンプルの makefile を使用するとき、次の点に注意してください。

- `-pic` (位置独立コード) コンパイラオプションを使用する必要があります。
- `-zp1` コンパイラオプションは、`UNALIGNED` 属性が各構造体のレベル 1 に指定されているかのように、すべての構造体を割り当てます。データファイルの配置により、このオプションが必要であるかどうかが判定されます。
- 各メインプログラムは、それぞれの共有ライブラリ (`acct00.so`、`acct01.so`、`acct02.so`、`acct03.so`、`acct04.so`) にリンクします。

- すべての共通サブルーチンは、1 つまたは複数の共有ライブラリにリンクされ、さらにそのライブラリは、Open PL/I ランタイムライブラリとともにトランザクションサーバーにリンクされます。会計アプリケーションの例では、共通サブルーチンは 1 つだけ (`ALPHANUM`) で、共有ライブラリ `libacct.so` にリンクします。

# コンパイルメニューを使用した PL/I プログラムのコンパイル

開発システムのコンパイルメニューを使用して、オンライン PL/I プログラムをコンパイルすることもできます。このメニューのオプションを使用すると、ファイルの編集、include ディレクトリの指定、およびコンパイラオプションの設定が可能になります。

コンパイルメニューを使用する場合は、コンパイルされる前に `kixclt` トランスレータにより、PL/I プログラムが自動的に変換されます。

## ▼ コンパイルするファイルを選択する

1. **開発システムメインメニューで PF4 を押します。**

   図 6-2 に示されているようなコンパイルメニューが表示されます。標準のファイル選択画面は、Sun MTP の画面に似ています。

2. **「Directory」フィールドに、PL/I プログラムを含むディレクトリのパス名を入力します。**

   最初のエレメントまたはパス全体の名前を環境変数で示すことができます。

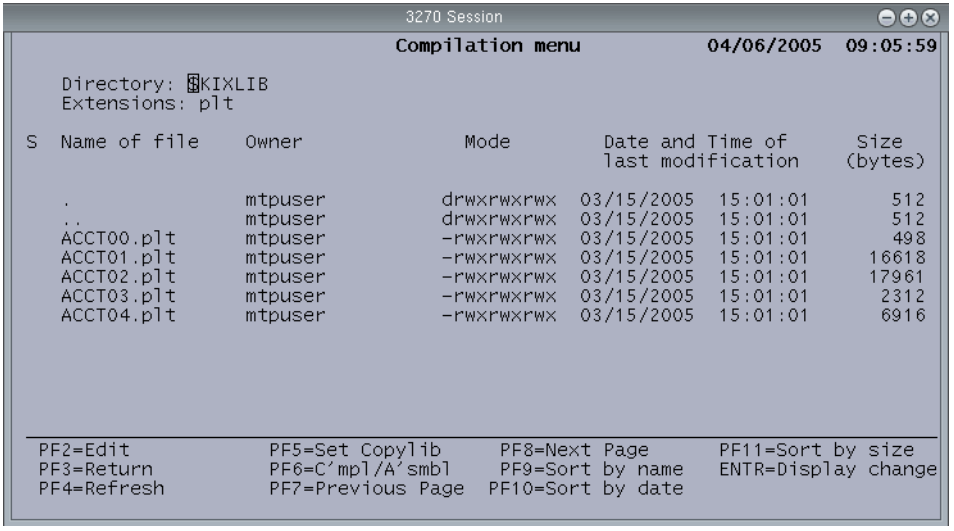


図 6-2 Compilation メニュー

3. 「Extensions」フィールドには、ファイル拡張子または表示する PL/I プログラムの拡張子を入力します。

   有効な PL/I 拡張子は次のとおりです。

   `.pli, .pl1`: Open PL/I コマンドだけ

   `.plt`: `EXEC CICS` コマンドの Open PL/I

   `.ppl`: `EXEC CICS` コマンドおよびデータベース (Oracle) コマンドの Open PL/I

   ファイル情報フィールドについては、表 2-2 を参照してください。

4. Enter キーを押します。

   画面が更新され、指定したディレクトリおよびファイルタイプを表示します。

5. ファイル名の左にある選択フィールドにカーソルを移動してファイルを選択し、ファイル名の横に大文字または小文字の `S` を入力します。

   複数ファイルの選択が可能です。

---

**注 –** COBOL ファイル名拡張子と Liant Open PL/I ファイル名拡張子を 1 つの選択セットで混在させないでください。混在させると、エラーメッセージ「`Illegal file extension for compile`」が表示されます。

---

6. **実行するアクションのファンクションキーを押します。**

   - PF2: `kixed` シェルスクリプトを実行して、選択したファイルを開きます。デフォルトでは `vi` エディタを使用します。複数のファイルを選択した場合、最後に選択したファイルを最初に、最初に選択したファイルを最後に編集します。`kixed` スクリプトは、各ファイルに対して自動的に起動されます。
   - PF5: インクルードファイルディレクトリを指定できる、Copylib 設定画面を表示します。詳細は、166 ページの「Include ディレクトリの指定」を参照してください。
   - PF6: コンパイラオプションの設定画面を表示します。詳細は、167 ページの「プログラムのコンパイル」を参照してください。

   次の表に、コンパイルメニューのその他のファンクションキーを示します。

**表 6-1** コンパイルメニューのファンクションキー

| ファンクションキー | 動作 |
|---|---|
| PF3 | コンパイルメニューを閉じて、開発システムメインメニューに戻ります。 |
| PF4 | 画面のファイルリスト部分のデータを更新します。ファイルの属性を変更したあと、PF4 キーを押して画面を更新します。 |
| PF7 | ファイルの前のページを表示します。ファイルの最初のページが表示されているときにこのキーを押すと、次のメッセージが表示されます。<br>`Beginning of data reached` |
| PF8 | ファイルの次のページを表示します。ファイルの最後のページが表示されているときにこのキーを押すと、次のメッセージが表示されます。<br>`End of data reached` |
| PF9 | ディレクトリの内容をファイル名順にソートして再表示します。 |
| PF10 | ディレクトリの内容をファイルの変更日の降順にソートして再表示します。 |
| PF11 | ディレクトリの内容をファイルサイズの降順にソートして再表示します。 |
| Enter | ファイルが選択されていない場合には、何も実行されません。ファイルを選択すると、コンパイラ画面が表示されます。<br>ディレクトリを選択すると、ディレクトリが選択したディレクトリに変更され、ディレクトリ内のファイルが表示されます。 |

# Include ディレクトリの指定

Copylib の設定画面では、PL/I ソースコードで参照されるインクルードファイルを検索するディレクトリを最大 8 つまで指定できます。ディレクトリは、次の順序で検索されます。

1. 現在のディレクトリ
2. Copylib の設定画面で定義されたディレクトリ

## ▼ Include ディレクトリを指定する

1. **コンパイルメニューで PF5 キーを押します。**

   図 6-3 に示す画面が表示されます。ディレクトリ名用の 8 つのフィールドがあります。

2. **Liant Open PL/I include ディレクトリのエントリを作成します。**

   これは、一般に `$LPI_PRODUCT_DIR/include` です。

3. **アプリケーションプログラムのインクルードファイルを含むその他のディレクトリを追加します。**

   名前の先頭には、ドル記号 ($) で示された環境変数も指定できます。環境変数のあとは、スラッシュ (/) で区切ってその他のサブディレクトリを続けられます。

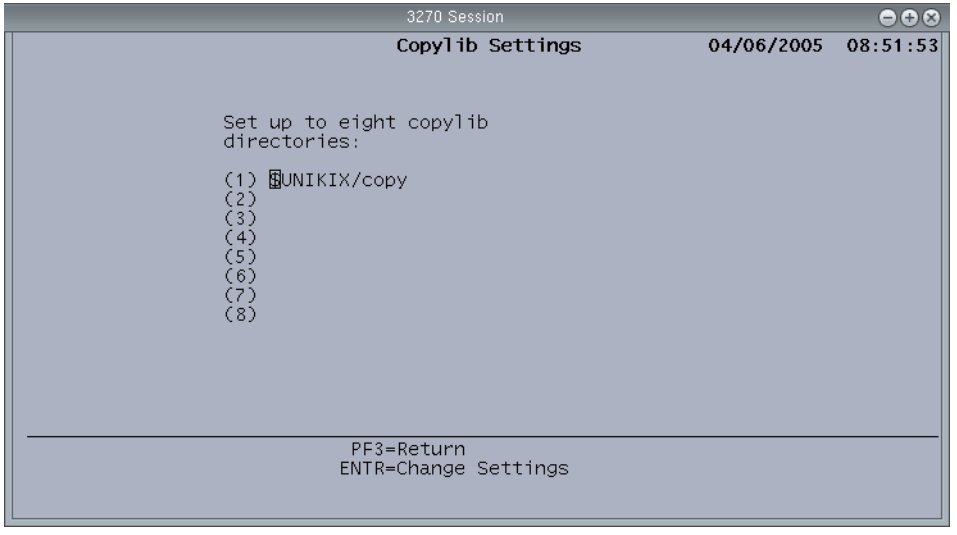


図 6-3　Copylib Settings 画面

4. Enter キーを押して、設定を保存します。

   他のユーザー指定オプションと同様、この画面で実行する Copylib の設定は `$KIXSYS/`*user-name*`.tbl` ファイル内に記述されています。*user-name* はユーザーのログイン名を示します。オプションは、セッション間で保存されます。

# プログラムのコンパイル

プログラムで必要な編集を完了し、include ディレクトリを設定したら、プログラムのコンパイル準備が整います。

## ▼ プログラムをコンパイルする

1. コンパイルメニュー (図 6-2) で PL/I プログラムファイルを選択します。
2. PF6 キーを押して「Set Compiler Options」画面を表示します。

   この画面では、コンパイラの処理をカスタマイズできます。

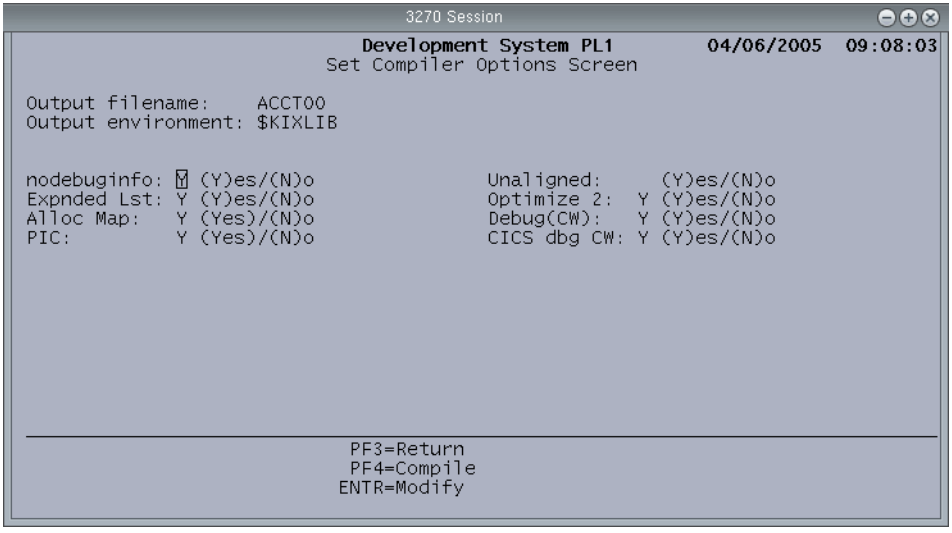


図 6-4　Set Compiler Options Screen (PL/I)

3. **必要に応じて、以下の画面の値を変更します。**

| | |
|---|---|
| Output filename | 現在選択しているファイルの名前がデフォルトとして表示されます。ファイル名は最大 14 文字です。 |
| Output environment | コンパイルメニュー画面のディレクトリ名がデフォルトとして表示されます。このフィールドの最初の要素を環境変数にして、スラッシュで区切ってサブディレクトリをあとに続けることができます。 |

一部のコンパイラオプションは、選択したファイルの拡張子に基づいて自動的に設定されます。`$UNIKIX/examples/primer/pli` ディレクトリの makefile に、これらのオプションが示されています。

画面のデフォルトの値を変更することにより、その他のオプションを設定できます。Liant PL/I 用の「Set Compiler Options」画面のオプションを、対応する Liant Open PL/I オプションとともに次の表に示します。詳細については、『Open PL/I User's Guide』を参照してください。

**表 6-2** PL/I コンパイラオプション

| 画面オプション | Open PL/I オプション |
|---|---|
| nodebuginfo | CodeWatch によるデバッグで使用 |
| Expnded List | 拡張リストを作成 |
| Alloc Map | リストでの記憶領域割り当てマップを作成 |
| PIC | 位置独立コード |
| Unaligned | レベル 1 で位置合わせさせていない PL/I の構造 |
| Optimize 2 | 最適化のレベル |
| Debug (CW) | CodeWatch の使用の指定 |
| CICS dbg CW | CodeWatch による `EXEC CICS` プリコンパイラ生成コードのデバッグを許可 |

これらのオプションおよびユーザー固有のその他のオプションは、`$KIXSYS/`*user-name*`.tbl` ファイル内に記述されています。*user-name* はユーザーのログイン名を示します。オプションは、セッション間で保存されます。

4. **「Set Compiler Options」画面で PF4 キーを押して、COBOL コンパイラを呼び出す `kixpl1` シェルスクリプトを実行します。**

このシェルスクリプトは、ソースファイルのファイル名拡張子に応じて、Liant Open PL/I コンパイラ、Oracle プリコンパイラ (拡張子 `.ppl`)、またはプリプロセストランスレータを呼び出します。複数のファイルを選択すると、ファイルを選択した順番と逆の順番で、最後に選択したファイルが最初にコンパイルされます。

コンパイルが完了すると、コンパイルおよび変換の結果がデフォルトのエディタ (`vi`) に表示されます。

# 第7章

# C の共有オブジェクトの使用法

Sun MTP では、C 言語で記述されたオンラインおよびバッチの CICS アプリケーションを実行する機能を提供します。オンラインの C プログラムは、COBOL プログラムとは異なり、`$KIXPROGS` ディレクトリから直接起動できません。その代わり、共有オブジェクトにリンクされているので、トランザクションサーバーから起動できます。

この章の内容は、次のとおりです。



**注 –** 共有オブジェクトと共有ライブラリという用語は同じものを指します。

標準バッチ環境でバッチの C プログラムを実行するには、C-ISAM インタフェースを使用する必要があります。詳細は、332 ページの「C 言語のバッチプログラムの実行」を参照してください。

また、docs.sun.com の『Linker and Libraries Guide』も参照してください。このマニュアルには、共有オブジェクトの構築と保守に関する詳細情報が記載されています。

# C アプリケーションの共有オブジェクトモデル

Sun MTP は、C アプリケーションプログラムを共有オブジェクトとして実行します。共有オブジェクトは、次の 3 段階の処理によって構築されます。

1. `kixclt` が、`EXEC CICS` コマンドを含む C コードを `.c` ファイルに変換します。詳細は、174 ページの「C プログラムの変換」を参照してください。
2. C コンパイラが、`.c` ファイルをオブジェクト (`.o`) ファイルに変換します。詳細は、180 ページの「共有オブジェクトの構築」を参照してください。
3. ローダが、共有オブジェクト (`.so` ファイル) を構築します。詳細は、180 ページの「共有オブジェクトの構築」を参照してください。

Sun MTP で共有オブジェクトを使用する方法は、次のとおりです。`$UNIKIX/examples/primer/C` ディレクトリにある C 言語のサンプルアプリケーションの例として使用します。

1. 共有オブジェクトは、プログラム管理テーブル (PCT) および処理プログラムテーブル (PPT) で定義されます。

| PCT プログラム | 変換 ID | PPT プログラム | 共有ライブラリ |
|---|---|---|---|
| ACCT00 | ACCT | ACCT00 | acct00 |
| ACCT01 | AC01 | ACCT01 | acct01 |
| | | ACCT02 | acct02 |

2. 領域を起動すると、各トランザクションサーバーが PPT の共有オブジェクトを準備します。`ACCT00` および `ACCT01` は PCT のエントリなので、それぞれに対応する共有オブジェクトに、エントリポイントとして `main()` 関数が 1 つだけ含まれている必要があります。デフォルトでは、トランザクションサーバーが、PPT で検出されるすべての C プログラムへのエントリポイントとしてシンボル `main` を検出する必要があります。`main` 関数が検出されない場合、指定されたプログラムで `main` 関数が検出されなかったことを示す警告メッセージが表示されます。
3. `ACCT00` および `ACCT01` プログラムには、それぞれ `main` 関数およびその他複数の関数のシンボルがあるが、`ACCT02` にはありません。`ACCT02` は、C スタイルの関数呼び出しにより `ACCT00` または `ACCT01` で使用される補助関数の目録の働きをします。

4. それぞれの main 関数のコーディングに必要な唯一の EXEC CICS コマンドは、EXEC CICS ADDRESS EIB(...) です。このコマンドがなくても、kixclt トランスレータは警告を出しません。ただし、実行時に予期しない結果になることがあります。

5. ソースファイルの接尾辞 .ccs は、main 関数、EXEC CICS コマンドを持つソースファイル、および EXEC CICS コマンドを持つ main 関数以外のソースに対して予約されています。EXEC CICS コマンドを持たない独立したコンパイル可能単位として存在する補助関数は、接尾辞 .c を使用します。これらのファイルは、それらを共有オブジェクトに組み込む前に kixclt で変換する必要はありません。

6. ACCT00、ACCT01、および ACCT02 のソースファイルの構造は、次に示すコード例のとおりです。これらのソースファイルが変換およびコンパイルされ、共有オブジェクトにリンクされます。

**コード例 7-1** acct00.ccs - ACCT00 のソースファイル

```
#include <stdio.h>
main()
{
. . .
func1();
. . .
funca();
}
void func1()
{
EXEC CICS ...
. . .
}
```

**コード例 7-2** acct01.ccs - ACCT01 のソースファイル

```
#include <stdio.h>
main()
{
EXEC CICS ADDRESS EIB(...)
func2();
funca();
}
```

**コード例 7-3**　　acct02.ccs - ACCT02 のソースファイル

```
#include <stdio.h>
void funca()
{
. . .
EXEC CICS ...
. . .
EXEC CICS RETURN ...
}
void func2()
{
. . .
. . .
}
```

# C での CICS API の使用法

この節では、サポートされていない API コマンドについて説明し、さらに、CICS コマンドのデータタイプで C と同等のものについて説明します。Sun MTPと CICS の互換性の詳細は、第 4 章を参照してください。

## サポートされない API コマンド

次の例外を除き、C では完全に CICS API がサポートされます。

- HANDLE CONDITION (ラベルありまたはラベルなし)
- HANDLE AID (ラベルありまたはラベルなし)
- IGNORE CONDITION
- PUSH HANDLE
- POP HANDLE
- HANDLE ABEND (ラベル)

これらのコマンドは、C 言語の構造化プログラムロジックフローの考え方と互換性を持ちません。これらは非構造化例外処理を表します。そのため、C プログラムは、各 CICS コマンドに続く RESP または RESP2 による独自の例外処理を行う必要があります。サポートされないコマンドを使用すると、トランスレータのエラー診断が行われます。

---

**注 –** HANDLE ABEND PROGRAM はサポートされます。

---

# C でのデータ型の置換

CICS コマンドとそのデータ型については、第 4 章で説明されています。次の表は、それに対応する C のデータ型を示します。

**表 7-1**　CICS API および C 言語のデータ型

| CICS API データ型 | C 言語データ型 |
|---|---|
| data-value (**ハーフワード**) | short |
| data-value (**フルワード**) | int |
| data-value (**文字列**) | char[n] |
| data-area (**ハーフワード**) | short |
| data-area (**フルワード**) | int |
| data-area (**フルワード**) | char[n] |
| cvda | int |
| prt-ref | C ポインタ型参照 (char * int *) |
| name | 二重引用符で囲まれた**文字**列 |

次の typdefs は、cicstype.h ヘッダーファイルに含まれています。

| | |
|---|---|
| cics_char_t | 文字 |
| cics_ubyte_t | 符号なし 1 バイト |
| cics_sshort_t | 符号付き short |
| cics_ushort_t | 符号なし short |
| cics_slong_t | 符号付き int |
| cics_ulong_t | 符号なし int |
| cics_bool_t | ブール型 |
| cics_uxlong_t | 符号なしのバイナリ (8 バイト) |
| cics_sxlong_t | 符号付きのバイナリ (8 バイト) |

# C プログラムの変換

C 言語および EXEC CICS コマンドが混在する C プログラムは、拡張子 .ccs で表されます。kixclt ユーティリティーは、これらのプログラムを .c ファイルに変換します。次の例のように、入力ファイル名および C 関連で必要なその他のオプションを指定して kixclt を実行します。

```
$ kixclt exp_main.css
```

kixclt とそのオプションについては、『Sun Mainframe Transaction Processing 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 7-4 は、.ccs ファイルのサンプルです。

**コード例 7-4** .ccs ファイル - 例 ( 1 / 2 )

```
#include <stdio.h>
/*     This is an example of a main() .ccs file with EXEC CICS
       commands embedded in it for illustrative purposes
*/
main()
{
       struct my_struct {
                         char stuff[23];
                         int  flen;
                        };
       struct my_struct right_here;
       struct my_struct *my_ptr;
       struct cics_eib *prm1;
       char *prm2;

       my_ptr = &right_here;
       EXEC CICS ADDRESS EIB(prm1);
       prm2 = 0;
       my_ptr->flen = 8192;
```

**コード例 7-4**　　.ccs ファイル - 例 ( 2 / 2)

```
        EXEC CICS GETMAIN
                SET     (&prm2)
                FLENGTH (my_ptr->flen)
                SHARED
                INITIMG ("|");
        /* More application code here */
        EXEC CICS FREEMAIN DATA (prm2);
        EXEC CICS RETURN
                TRANSID ("BC01");
}
```

変換時、トランスレータは kxdfhei1 への適切な関数呼び出しを使用して、.h ヘッダーファイルを出力ファイルに挿入します。

| | |
|---|---|
| cicstype.h | CICS typedefs |
| cics_eib.h | EIB 制御構造体 |
| cics_api.h | API 制御機能の定義 |

main() 関数を含む .ccs ファイルでは、トランスレータは領域内のトランザクション環境を初期化するための関数呼び出しも挿入します。main() 関数を持たない .ccs ファイルでは、.h ヘッダーファイルは挿入されますが、初期化関数呼び出しは挿入されません。

アプリケーションによって必要とされる場合、次のヘッダーファイルを .ccs ファイルに取り込みます。

| | |
|---|---|
| dfhaid.h | アテンションキーコードの定義 |
| dfhbmsca.h | BMS 制御コード |

これらすべてのヘッダーファイルは、$UNIKIX/src/CICS_structures ディレクトリにあります。

次のコード例は、コード例 7-4 の .ccs ファイルの変換結果です。出力ファイルには拡張子 .c が付きます。変換時にエラーが発生すると、*filename*.err ファイルが生成されます。

**コード例 7-5**　　.c ファイル - 例 ( 1 / 2 )

```
/* This is an example of a main() .ccs file with EXEC CICS
   commands embedded in it for illustrative purposes                */
#include <stdio.h>
#include "cicstype.h"
#include "cics_eib.h"
#include "cics_api.h"
main()

{
          int kxdfh_entry = cics_entr("EXP_MAIN");
       struct my_struct {
                  char stuff[23];
                  int  flen;
                  };
       struct my_struct right_here;
       struct my_struct *my_ptr;
       struct cics_eib *prm1;
       char *prm2;

       my_ptr = &right_here;
           /*EXEC CICS ADDRESS EIB(prm1);*/
              {
                  KIX_MOV_DFHEIV0("\"\"  E                  \"   #00000016");
                  kxdfhei1(&_dfheiv0,
                                 &prm1);
                  cics_dfhbak();
              }

       prm2 = 0;
       my_ptr->flen = 8192;
       /*EXEC CICS GETMAIN
              SET     (&prm2)
                  FLENGTH (my_ptr->flen)
                  SHARED
                  INITIMG ("|");*/
              {
```

**コード例 7-5**　　.c ファイル - 例 ( 2 / 2)

```
       KIX_MOV_STR(_dfheiv9,"|");
                       KIX_MOV_DFHEIV0(",\"IF S                $   #00000019");
                       kxdfhei1(&_dfheiv0,
                              &prm2,
                              &my_ptr->flen,
                              &_dfheiv9);
                       cics_dfhbak();
                       }
       /* More application code here */
       /*EXEC CICS FREEMAIN DATA    (prm2);*/
                       {
                       KIX_MOV_DFHEIV0(",$                    \"   #00000025");
                       kxdfhei1(&_dfheiv0,
                              *&prm2);
                       cics_dfhbak();
                       }

/*EXEC CICS RETURN
              TRANSID ("BC01");*/

                      {
                       KIX_MOV_STR(_dfheiv5,"BC01");
                       KIX_MOV_DFHEIV0(".(T                   $   #00000026");
                       kxdfhei1(&_dfheiv0,
                              &_dfheiv5,
                              &_dfheiv99,
                              &_dfheiv99);
                       cics_gobk();
                       }
}
```

# C トランスレータの制限事項

kixclt トランスレータは、C プログラムの EXEC CICS コマンドをインライン C コードに変換します。ただし、C 言語のさまざまな構造のすべてを変換するわけではないので、次の例のように EXEC CICS コマンドの引き数として複雑な C 言語の表現を組み込まないようにします。

```
EXEC CICS . . . (command)
       LENGTH (func1() + func2() + ((num_1 < num_2) ?(short)100 :(short)200)
```

このような場合、適切な型のローカル変数を宣言し、EXEC CICS コマンドの外側でその変数に複雑な C 表現の結果を割り当てます。さらに、次のようにその変数を EXEC CICS コマンドの引数として使用します。次に例を示します。

```
short my_var;
my_var =

       (short) func1() + func2() + ((num_1 < num_2) ?(short)100 :
           short(200)));
EXEC CICS .. .(command)
       LENGTH(my_var);
```

# 共有オブジェクトの設定

すべての C アプリケーションプログラムは、以前に構築した 1 つまたは複数の共有オブジェクトから実行する必要があります。詳細は、180 ページの「共有オブジェクトの構築」を参照してください。

各 C プログラムおよびそれらが存在する関連の共有オブジェクトを PPT で特定する必要があります。詳細は、『Sun Mainframe Transaction Processing 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。

## 共有オブジェクトの命名

共有オブジェクトを命名するには、次の要件を満たす名前にする必要があります。

- 環境変数 KIXLIB を定義していないと、PPT の共有オブジェクトのエントリはすべて、$KIXSYS ディレクトリからの相対パスとなります。$KIXLIB を設定すると、この環境変数に共有オブジェクトを検索するディレクトリを指定するパス名のリストが含まれます。環境変数の設定方法については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。
- 共有オブジェクト名の PPT エントリは、サブディレクトリ名を含め、拡張子 .so を含めずに最大 16 文字です。これにより、$KIXSYS または $KIXLIB より下位レベルのサブディレクトリの長さと数が制限されます。
- 共有オブジェクト名は小文字。
- パスの一部が異なっていれば、2 つの共有オブジェクトに同じ名前を付けることができます。
- 実際の共有オブジェクトは拡張子 .so で構築されるが、PPT では拡張子 .so は使用しません。共有オブジェクトを割り当てるとき、Sun MTP が PPT から取り出す名前にこの拡張子を付加します。

たとえば、次のオブジェクト名は有効です。

`lev1/lev2/myshobj`: PPT エントリの最大長 (Sun MTP では `.so` は名前の一部ではない)。このエントリには、共有オブジェクトの前に 2 つのサブディレクトリがあります。

`myshobj`: サブディレクトリを持たない共有オブジェクト。

## 共有オブジェクトのオープン

PPT で一覧表示された各共有オブジェクトは、それぞれの指定されたトランザクションに対応します。個々の共有オブジェクトは、対応するトランザクションが呼び出されるまでオープンされません。共有オブジェクトをオープンするこの方法では、呼び出された共有オブジェクトだけをオープンすることによって起動時のオーバーヘッドを低く抑えます。ただし、共有オブジェクトが一度オープンされると、トランザクションが終了してもアンロード (すなわち、クローズ) されません。同じトランザクションの次の呼び出しのために、利用できる状態をそのまま維持するからです。

アプリケーション共有オブジェクトのロードアルゴリズムは、対応するプログラムの呼び出しに基づいているので、すべての記号への参照の依存性は、特定の共有オブジェクトを構築するときにリンクラインで宣言する必要があります (180 ページの「共有オブジェクト構築のためのリンクラインの例」を参照)。これを行わないと、動的ローダーではこれらの記号を解決に利用できません。

---

**注 –** 共有オブジェクトが依存性を持たない場合、変更を行う必要はありません。

---

動的ローダーが依存性のある記号を利用できるようにする別の方法として、PPT エントリのプリロード (P/L) フィールドを依存性のある共有オブジェクトに対して `Y` に設定します。このフィールドを `Y` に設定すると、起動時にトランザクションサーバーはすべての「プリロード」された共有オブジェクトをオープンします。PPT の各共有オブジェクトでプリロードフィールドが `Y` に設定されると、その領域が起動するときにすべての共有オブジェクトがロードされます。

---

**注 –** 共有オブジェクトの構築時にリンクラインで依存性を宣言する必要も、実行時に共有オブジェクトをオープンするための「pre-load」フラグを設定する必要もありません。

---

### 共有オブジェクト構築のためのリンクラインの例

次のリンクラインの例は、依存性のある prog1.so という名前の共有オブジェクトを、libmoreso.so と呼ばれる別の共有オブジェクトに構築する例です。

```
$ ld -G -Bdynamic  -o prog1.so prog1.o -R $KIXLIB/mysos1 \
   -L $KIXLIB/mysos1 -lmoreso
```

---

**注 –** 各引数については、ld(1) コマンドのマニュアルページを参照してください。

---

### PPT プリロードフィールド

各 PPT エントリには、「P/L」というラベルの 1 文字のフィールドがあります。領域が起動するときに対応する共有オブジェクトがプリロードされるかどうかを示す場合、このフィールドを Y に設定する必要があります。開く必要がない場合は、このフィールドを N に設定するか、空白のままにします。

## 共有オブジェクトの構築

この節では、C および C++ 言語共有オブジェクトを構築する手順について説明します。

### ▼ C 言語共有オブジェクトを構築する

1. **kixclt トランスレータを実行して、.ccs ファイルを変換します。**
   .c ファイルは、トランスレータの出力です。
2. **C コンパイラで .c ファイルをコンパイルして、.o ファイルを作成します。**
3. **ローダーを起動して、.o ファイルの処理および共有オブジェクトファイル (拡張子 .so) の作成を行います。**
4. **共有オブジェクトを適切なディレクトリに移動します。**

次のコマンドは、共有オブジェクトを変換、コンパイル、作成し、$KIXLIB ディレクトリに移動します。この例では、このディレクトリは、1 つの共有オブジェクトディレクトリを指します。

```
$ kixclt acct00.ccs
$ cc -Bdynamic -Kpic -c -Xt -I $UNIKIX/src/CICS_structures \
      -o acct00.o acct00.c
$ ld -G -o acct00.so acct00.o
$ mv acct00.so $KIXLIB
```

## ▼ C++ 言語共有オブジェクトを構築する

1. kixclt トランスレータを実行して、.ccs ファイルを変換します。

   .c ファイルは、トランスレータの出力です。

2. C++ コンパイラで .c ファイルをコンパイルして、.o ファイルを作成します。

3. ローダーを起動して、.o ファイルの処理および共有オブジェクトファイル (拡張子 .so) の作成を行います。

4. 共有オブジェクトを適切なディレクトリに移動します。

   次のコマンドは、共有オブジェクトを変換、コンパイル、作成し、$KIXLIB ディレクトリに移動します。この例では、このディレクトリは、1 つの共有オブジェクトディレクトリを指します。

```
$ kixclt acct00.ccs
$ CC -Bdynamic -Kpic -c -Xt -I $UNIKIX/src/CICS_structures -o acct00.o acct00.c
$ ld -G -o acct00.so acct00.o
$ mv acct00.so $KIXLIB
```

---

**注 –** $UNIKIX/src/CICS_structures ディレクトリにあるインクルードファイルは、C++ および C 言語プログラムで同じです。

---

# BMS マップのアセンブル

C 環境での kixbms ユーティリティーのコマンド行機能は、COBOL 環境での場合と同じです。ただし、その出力は異なります。.bms ソースファイルを変換する形式は、次のとおりです。

```
$ kixbms -c TSTMNU.bms
```

このコマンドからの出力は、TSTMNU.bms ソースファイルの LANG オプションによって決定されます。LANG=C の場合、kixbms は次のファイルを作成します。

- TESTMNU.map: $KIXMAPS ディレクトリに格納されるマップファイル
- TESTMNU.h: C ヘッダーファイル (#include "TSTMNU.h")

kixbms からのすべての .h 出力は、C 構造体形式です。つまり、すべてのマップ定義が位置合わせされています。COBOL と C アプリケーションプログラムに対して同じマップおよびマップセットを使用する場合、kixbms コマンド行で -a (位置合わせ) オプションを使用して、マップを記述するすべての COBOL コピーブックを再構築する必要があります。kixbms のすべてのオプションについては、『Sun Mainframe Transaction Processing 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 ヘッダーファイルが作成されると、常に出力ファイル名は大文字で、小文字の拡張子 .h が付きます。このヘッダーファイル名は、BMS マップを使用するプログラムの .ccs ソースファイルに取り込まれる必要があります。

次の例は、BMS 入力ファイルです。2 行目の STORAGE=AUTO 文に注意してください。

**コード例 7-6**　　BMS ソースファイル

```
          TITLE 'FILEA - MAP FOR OPERATOR INSTRUCTIONS - C'
TSTMNU  DFHMSD MODE=INOUT,CTRL=(FREEKB,FRSET),STORAGE=AUTO,           *
               LANG=C,TIOAPFX=YES,EXTATT=MAPONLY,COLOR=BLUE
MENU    DFHMDI SIZE=(12,40)
        DFHMDF POS=(1,10),LENGTH=21,INITIAL='OPERATOR INSTRUCTIONS',  *
               HILIGHT=UNDERLINE
        DFHMDF POS=(1.32),LENGTH=1
        DFHMDF POS=(3,1),LENGTH=29,INITIAL='OPERATOR INSTR - ENTER MEN*
               U'
        DFHMDF POS=(4,1),LENGTH=38,INITIAL='FILE INQUIRY   - ENTER INQ*
               Y AND NUMBER'
        DFHMDF POS=(5,1),LENGTH=38,INITIAL='FILE BROWSE    - ENTER BRW*
               S AND NUMBER'
        DFHMDF POS=(6,1),LENGTH=38,INITIAL='FILE ADD       - ENTER ADD*
               S AND NUMBER'
        DFHMDF POS=(7,1),LENGTH=38,INITIAL='FILE UPDATE    - ENTER UPD*
               T AND NUMBER'
MSG     DFHMDF POS=(11,1),LENGTH=39,INITIAL='PRESS CLEAR TO EXIT'
        DFHMDF POS=(12,1),LENGTH=18,INITIAL='ENTER TRANSACTION:'
        DFHMDF POS=(12,20),LENGTH=4,ATTRB=IC,COLOR=GREEN,             *
               HILIGHT=REVERSE
        DFHMDF POS=(12,25),LENGTH=6,INITIAL='NUMBER'
KEY     DFHMDF POS=(12,32),LENGTH=6,ATTRB=NUM,COLOR=GREEN,            *
               HILIGHT=REVERSE
        DFHMDF POS=(12,39),LENGTH=1
        DFHMSD TYPE=FINAL
```

コード例 7-7 では、kixbms の実行後の .h 出力ファイルを示しています。STORAGE=AUTO 文を使用したことによって、最後の行の union 文で構造体のインスタンス化を示しています。

**コード例 7-7**　　kixbms による .h 出力ファイル - インスタンス化の例 (1 / 2)

```
union menuu {
       struct {
               char  dfhms1[12];
               short msgl;
               char  msgf;
               char  msgi[39];
               short keyl;
               char  keyf;
               char  keyi[6];
               }     menui;
```

**コード例 7-7** kixbms による .h 出力ファイル - インスタンス化の例 (2 / 2)

```
        struct {
               char  dfhms1[12];
               short dfhms2;
               char  msga;
               char  msgo[39];
               short dfhms3;
               char  keya;
               char  keyo[6];
               }     menuo;
               }     ;
union menuu menu;
```

BMS ソースファイルで STORAGE=AUTO 文を省略すると、コード例 7-8 のように、出力される .h ファイルの最後の行は構造体へのポインタになります。

**コード例 7-8** kixbms による .h 出力ファイル - ポインタの例

```
union menuu {
        struct {
               char  dfhms1[12];
               short msgl;
               char  msgf;
               char  msgi[39];
               short keyl;
               char  keyf;
               char  keyi[6];
               }     menui;
        struct {
               char  dfhms1[12];
               short dfhms2;
               char  msga;
               char  msgo[39];
               short dfhms3;
               char  keya;
               char  keyo[6];
               }     menuo;
               }     ;
} *bmsmapbr;
```

# C プログラムの起動

C プログラムの起動時には通常、2 つの引数 (argc および argv) がプログラムに渡されます。ただし、CICS プログラムは特別な環境で実行され、トランザクションサーバーによってロードされることを前提としています。そのため、領域内の C プログラムが開始するとき、コマンド行引数 (トランザクション ID) が 1 つだけ指定されているかのように扱われます。これにより、argc は 1 に設定され、argv[0] にはトランザクションコードが入ります。

EIB ブロックのアドレスおよび COMMAREA アドレスは、起動する C プログラムに渡されません。この 2 つのアドレスは、アプリケーションプログラムで必要とされる場合、それぞれ EXEC CICS ADDRESS EIB および EXEC CICS ADDRESS COMMAREA 文を使用して取得します。

Sun MTP トランザクション環境を初期化するための関数呼び出しは、アプリケーションコードで使用できる tcb_eib_ptr (cics_api.h 内) と呼ばれる変数への EIB ポインタを取得します。そのため、EXEC CICS ADDRESS EIB を使用してポインタを再び取得する必要はありません。ただし、アプリケーションの独自の EIB ポインタについては、dfheiptr と呼ばれる cics_api.h で提供される 2 つ目の変数を使用して取得する必要があります。

# 共有オブジェクトでの CEMT の使用法

CEMT SET および CEMT INQ へのオプションでは、プログラムの共有オブジェクトの動的な変更および現在のオブジェクト名の照会ができます。詳細は、186 ページの「同じ共有オブジェクトの異なるバージョンの実行」を参照してください。

CEMT SET PROGRAM トランザクションの LIBRARY オプションでは、Sun MTP の実行中にプログラムの共有オブジェクトを動的に変更できます。

形式:

```
CEMT S[ET] PROG[RAM] prog-name LIB[RARY] lib-name PROG[RAM] prog-name
       LIB[RARY] blankreset
```

| オプション | 説明 |
|---|---|
| *prog-name* | プログラム名。最大 8 文字です。 |
| *lib-name* | 共有オブジェクト名。最大 16 文字の小文字です。 |
| blankreset | プログラムの PPT 共有オブジェクトフィールドを空白に設定することにより、指定されたプログラムを共有ライブラリ処理から削除します。 |

**注 –** blankreset は、ここに記載されているとおりに正確に入力する必要があります。間違いがあると、新しい共有オブジェクト名として扱われます。

CEMT コマンドを実行したあと、CINI トランザクションを実行して新しい共有オブジェクトおよび内部テーブルを初期化します。

現在の共有オブジェクト名を照会するには、CEMT INQ PROGRAM トランザクションの LIBRARY オプションを使用します。

形式:

```
CEMT I[NQ] PROG[RAM] prog-name LIB[RARY]
```

*prog-name* は、最大 8 文字のプログラム名です。

プログラムに共有オブジェクトがある場合、次のメッセージが表示されます。

KIX1584I CEMT transaction terminated ShrLib = *lib-name*

プログラムに共有オブジェクトがない場合、次のメッセージが表示されます。

KIX1584I CEMT transaction terminated ShrLib = NoSharedLibPrsnt

# 同じ共有オブジェクトの異なるバージョンの実行

Sun MTP の領域を再起動しなくても、共有オブジェクトの新しいバージョンをロードできます。たとえばアプリケーションが、PPT で dir1/prog_pay1 として定義されている prog_pay1.so という名前の共有ライブラリを使用するとします。実行中、prog_pay1.so が期待通りに動作しないことが判明したので、prog_pay1.so の新しいバージョンをコンパイルおよび再構築します。領域を再起動して新しいバージョンを得るのではなく、異なるディレクトリ dir2 に共有オブジェクトの新しいバージョンを保存します。

新しい共有オブジェクトを実行するために、次のトランザクションを使用します。

```
CEMT SET PROGRAM MY_PROG1 LIBRARY dir2/prog_pay1
```

このトランザクションによって、共有オブジェクトが配置される新しいディレクトリを指定するために、共有メモリー PPT が変更されます。この変更は、領域が存在する間有効です。パス名は最大 16 文字で、$KIXLIB または $KIXSYS ディレクトリからの相対パスです。PPT での共有オブジェクトの定義については、『Sun Mainframe Transaction Processing 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。

次のトランザクションでは、トランザクション処理プログラムによって共有オブジェクトを再オープンし、それを利用できるようにします。

```
CEMT SET PROGRAM MY_PROG1 LIBRARY NEWCOPY
```

**注 –** 共有オブジェクトの再コンパイルおよび再構築だけ行って、以前のバージョンを上書きしないでください。予期しないエラーが発生することがあります。

# C 共有オブジェクトでのスタティックおよび外部変数の初期化

Sun MTP では、C 言語共有オブジェクトのスタティックおよび外部変数の初期化を制御できます。

- 領域レベルで、環境変数 KIX_PGM_MODE を使用できます。KIX_PGM_MODE が KIX_PGM_MODE=TXSERIES に設定されている場合は、C プログラム (共有オブジェクト) は、プログラムが実行されるたびに、初期イメージを取得します。

  C プログラムがスタティックまたは外部変数を初期化しない場合は、この環境変数を使用すると、プログラムの新しいコピーが実行されるたびに Sun MTP がそれをロードします。

**注 –** この環境変数を設定すると、パフォーマンスが低下することがあります。

- 代替リソース定義 (CEDA トランザクションまたは dfhusdup ユーティリティー) を使用して、プログラムのリソースを定義する場合は、以下の方法で RESIDENT オプションを使用できます。
  - RESIDENT が YES に設定されている場合は、共有オブジェクト (プログラム) がプロセスに添付されたまま残ります。
  - RESIDENT が NO に設定されている場合は、プログラムが実行されるたびに、共有オブジェクトが削除され、オープンされます。

例では、CEDA トランザクションを使用して、非常駐として ABCD01 プログラムを定義します。

```
CEDA DEFINE GROUP(PAYROLL) PROGRAM(ABCD01) RESIDENT(NO)
```

---

**注 –** $KIX_PGM_MODE が設定されていない場合は、RESIDENT オプションは無視されます。

---

この RESIDENT オプションは、Sun MTP に特有の実装です。メインフレームの実装またはオープンシステムの TxSeries の実装とは異なります。

第8章

# Java プログラムの使用法

Sun MTP ソフトウェアでは、Java プログラミング言語で記述されたオンラインおよびバッチのアプリケーションを実行する機能を提供します。この章では、オンライン環境での Java プログラムの使用に重点を置いて説明します。Sun Mainframe Batch Manager (Sun MBM) 環境での Java プログラムの使用法については、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

この章の内容は、次のとおりです。



## JCICS を使用する前提条件

Java CICS API (JCICS) を使用するには、システム上で最低でも Solaris 9 オペレーティングシステムが稼動している必要があります。また、次の事項も前提条件です。

- Java 1.4 Development Kit (JDK™)、最小リリース 1.4 のインストール
- `$JDKROOT/jre/lib/sparc` ディレクトリ内の JVM™
- 領域を起動する前に、`KIX_ENABLE_JAVA` 環境変数を設定します。

**注 –** `$JDKROOT` は、Java Development Kit のインストールディレクトリです。

# JCICS API の使用法

Sun MTP 環境でサポートされているほかのプログラミング言語とは異なり、Java プログラムでは `EXEC CICS` API を使用しません。代わりに、Java プログラミング言語で記述された Sun MTP アプリケーションでは、JCICS API を使用します。

IBM は、OS/390 製品の CICS Transaction Server version 1.3 に JCICS API を導入しました。JCICS API の詳細は、オンラインで参照できる IBM RedBook の『Java Application Development for CICS』(パーツ番号 SG24-5275-01) を参照してください。

JCICS API は、C および COBOL 言語の API のような変換された `EXEC CICS` API ではありません。他の Java クラスのセットと同様に、Java Sun MTP アプリケーションから呼び出される Java クラスのセットで構成されています。現在 Sun MTP でサポートされる JCICS API のサブセットの完全なマニュアルは、Javadoc™ 形式 HTML で `$UNIKIX/doc/jcics` ディレクトリにあります。

プログラマには完全な JCICS API が提供されていますが (すなわち、関連するクラスおよびメソッドが存在する)、まだ完全に実装されていない重要な領域も存在します。実装されていないメソッドに対してAPI 呼び出しを行なった場合、例外 `com.sun.emp.mtp.jcics.NotImplementedException` が発生します。JCICS API の Javadoc には、この例外が適用されるメソッドが示されています。

Sun MTP の JCICS のこのリリースでは、タスク制御および端末制御の選択したセクションとともに、JCICS API のプログラム制御セクションの大部分に対してサポートが提供されています。API のその他の部分も、ごく一部が実装されています。

# JCICS アプリケーションアーキテクチャー

JCICS プログラムは、その他の言語で記述されたプログラムと同じ方法で Sun MTP に対して定義されます。プログラムは PPT 定義を必要とし、さらにそのプログラムがトランザクションで最初に呼び出される場合、PCT エントリを追加する必要があります。

JCICS プログラムは、`EXEC CICS LINK` を使用して他の Sun MTP プログラム (COBOL などの異なる言語で記述されたプログラムを含む) からも呼び出すことができます。JCICS プログラムでは、このような他の Sun MTP プログラムへのリンクが可能です。JCICS API によって、プログラムクラスのメソッド `link()` を使用してこれを実現します。

JCICS プログラムが呼び出されると、2 つのメソッドのいずれか 1 つでユーザーの Java アプリケーションに制御が渡されます。通常、指定された Java クラスには、`CommareaHolder` をパラメータとして受け取る定義済みの **static** `main()` メソッドがあります。このメソッドは、次のように定義されています。

```
public static void main(CommareaHolder cah);
```

このメソッドが存在しない場合、次のように定義されたメソッドが呼び出されます。

```
public static void main(String[] array);
```

が実行されます。いずれのメソッドも存在しない場合、プログラムの呼び出しに失敗します。

次の例は、端末接続トランザクションとして実行された場合に、ユーザー端末に `Hello World` の文字列を表示する JCICS プログラムです。

**コード例 8-1**　　Hello World JCICS プログラム

```
import com.ibm.cics.server.CommAreaHolder;
import com.ibm.cics.server.Task;

public class HelloWorld
{
       public static void main(CommAreaHolder cah)
       {
           Task task = Task.getTask();
           task.out.println("Hello World");
       }
}
```

`Task` クラスは、JCICS アプリケーションを記述する上で重要です。アプリケーションでは、`Task` クラスから `out` と呼ばれる Java printwriter にアクセスできます。端末接続トランザクションでは、この printwriter に書かれたすべてのデータがユーザーの端末に表示されます。

---

**注 –** Sun MTP で提供される JCICS クラスは、`com.ibm` で始まる名前を持つ Java パッケージで提供されます。これにより、IBM Transaction Server から Sun MTP への JCICS プログラムの移植が容易になります。

---

# JCICS アプリケーションの構築

JCICS アプリケーションは、他の Java プログラムとまったく同じ方法で構築されます。javac コマンドは、必要なクラスをコンパイルするために使用します。JCICS プログラムでは、使用するアプリケーション用に記述された一部のクラスに Sun MTP で提供されるクラスへの参照が含まれています。これらのクラスは、$UNIKIX/lib/dfjcics.jar ファイル内にあります。そのため、JCICS アプリケーションクラスをコンパイルするには、コンパイラのクラスパスでこれらのクラスを指定する必要があります。環境変数 CLASSPATH を設定するか、javac コマンドで -classpath オプションを指定します。次のコマンドは、javac コマンドで -classpath オプションを指定しています。

```
$ javac -classpath $UNIKIX/lib/dfjcics.jar:.HelloWorld.java
```

コンパイルされたプログラムは、HelloWorld.class です。

プログラムのコンパイルに、IBM の VisualAge for Java のような統合開発環境 (IDE) または Sun 以外の製品をご使用の場合、ご使用のコンパイラでこれらのクラスを有効にする方法については、それぞれのユーザーマニュアルを参照してください。

# クラスパスおよびライブラリパスのカスタマイズ

JCICS フレームワークは、JVM にクラスパス属性を設定するために、Sun MTP の環境変数 KIXPROGS を使用します。JCICS クラスローダーに対して追加のクラスまたは JAR ファイルをロードするように指示するには、Classpath.apends および Libpath.appends ファイルを使用します。

## ▼ Classpath.appends および Libpath.appends ファイルを有効化する

1. **実行中の領域は終了します。**
2. **領域環境が設定されていることを確認します。**

3. 領域の $KIXSYS ディレクトリに移動して、kix_java ディレクトリを作成します。

```
$ cd $KIXSYS
$ mkdir kix_java
```

4. kix_java ディレクトリに移動します。

```
$ cd kix_java
```

5. $UNIKIX/lib/kix_java ディレクトリから Classpath.appends および Libpath.appends ファイルをコピーします。

```
$ cp $UNIKIX/lib/kix_java/*.appends .
```

6. 必要なエントリをこれらのファイルにすべて追加し、保存します。

# PPT での Java プログラムの定義

COBOL や C で記述された他のプログラムと同じ方法で、PPT の Sun MTPに JCICS プログラムを定義します。

## ▼ PPT で Java プログラムを定義する

1. 領域を開始して、クライアント接続を確立します。
2. CTBL トランザクションを入力して、テーブルマネージャーを開きます。
3. PF4 キーを押して、「Standard Tables」メニューを表示します。
4. PF7 キーを押して、PPT を開きます。
5. PF4 キーを押して、新しいエントリを挿入します。
6. PPT 挿入画面で、次のように入力します。
   a. 「Program」フィールドにプログラム名を入力します (たとえば、HELLOW2)。
   b. 「Typ」フィールドに Java プログラムを示す J を入力します。

7. **Enter キーを押します。**

   詳細は、図 8-1を参照してください。

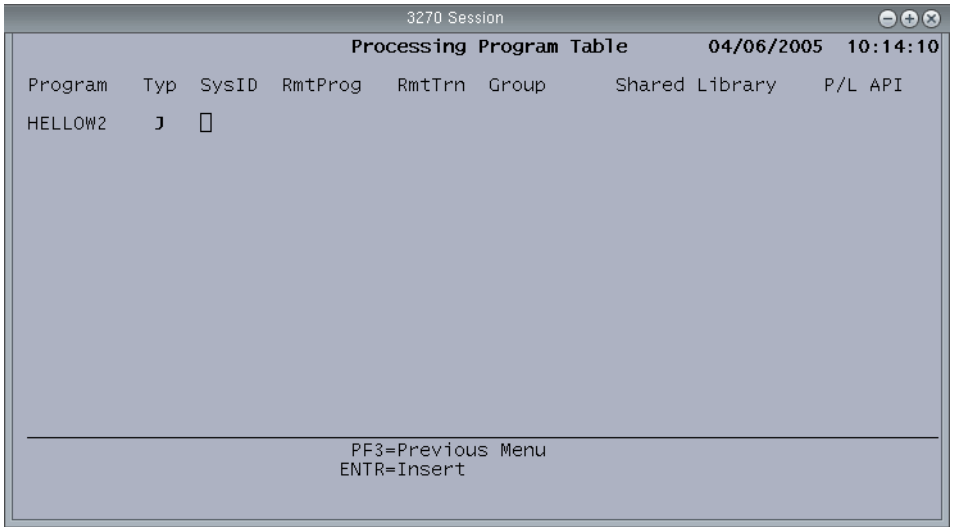


**図 8-1** Processing Program Table—Java プログラムの定義

8. **PF3 キーを押して、PPT のメイン画面に戻ります。**

9. **PPT のメイン画面で `HELLOW2` プログラムを選択し、PF9 キーを押して Java Class の詳細画面を開きます。**

10. **Java Class の詳細画面の「Fully qualified JAVA Class Name」フィールドに Java クラス名を入力します。このとき、`.class` ファイル名接尾辞を含めないでください。クラス接尾辞を含めると、実行時にエラーが発生します。**

    Java Class の詳細画面は、以下の理由で必要です。

    - Sun MTP プログラムの名前は 8 文字以内。Java クラス名はこれより長い名前でも可能です。
    - Sun MTP プログラム名では大文字・小文字を区別しないが、Java クラス名では区別します。

    「Program Name」フィールドの名前は、PPT のメイン画面で選択したプログラムです。

    「Remote Object」および「IIOP Connectivity」フィールドは、このリリースでは使用しません。

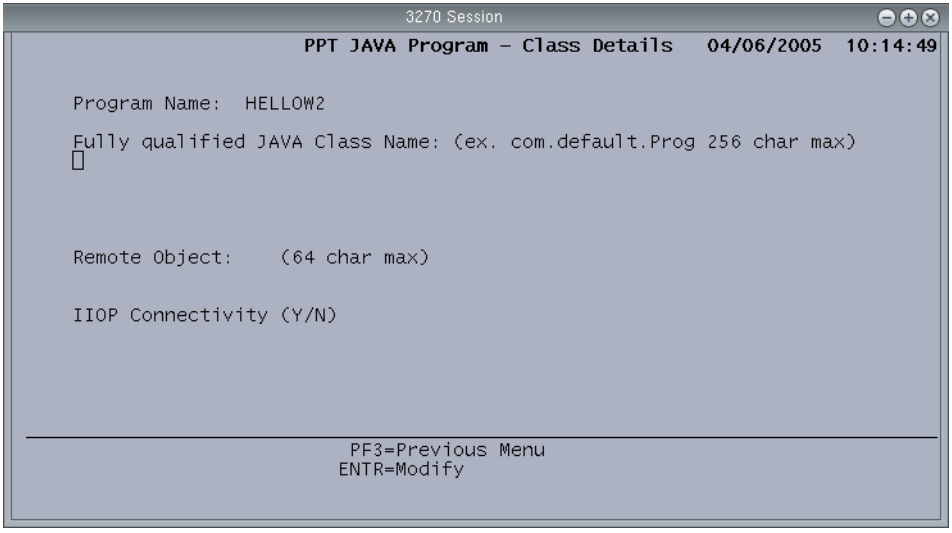


図 8-2　Processing Program Table—Java Class 詳細画面での Java Program の定義

11. Enter キーを押して、この情報を追加します。
12. PF3 キーを押して、PPT のメイン画面に戻ります。
13. PPT のメイン画面で PF2 キーを押し、ディスクにテーブルの変更を書き込みます。
14. PF3 キーを 2 回押して、テーブルマネージャーのメインメニューに戻ります。
15. PF3 キーを押して、テーブルマネージャーを終了します。
16. この領域を停止します。

この例では、`JHELLOW2` プログラムが実行されると呼び出される Java クラスの完全修飾名が `HelloWorld` です。このクラスが Java パッケージの一部である場合、このパッケージ名をこのエントリの一部として指定する必要があります。

JCICS プログラムのクラスファイルを領域の `$KIXPROGS` ディレクトリに置き、Sun MTP がそれを検出して実行できるようにします。Sun MTP では、`$KIXPROGS` の内容が `CLASSPATH` として JVM に渡されます。たとえば、PPT の Java Class の詳細画面で `HelloWorld` として定義されている JCICS プログラムのクラスファイルは、`$KIXPROGS/HelloWorld.class` として領域にアクセスできる必要があります。

アプリケーションクラスファイルは、Java アーカイブ、zip ファイルなどのアーカイブファイルにまとめることができます。その場合、次の領域設定ファイルのように、$KIXPROGS にアーカイブファイルのパス名およびファイル名を含めます。

```
KIXPROGS=$SYSTEM/progs:$SYSTEM/progs/myapp.jar; export KIXPROGS
```

**注 –** $KIXPROGS は、コロンで区切られたディレクトリのリストである場合もあります。

# JCICS サポートによる領域の開始

JCICS プログラムを実行可能な領域を開始するには、特別な構成が必要になります。

## ▼ 領域を開始する

1. **Java ランタイム環境が設定されていることを確認します。**
2. **KIX_ENABLE_JAVA 環境変数を YES に設定します。**

   たとえば、領域のセットアップファイルに以下の文を含めます。

```
KIX_ENABLE_JAVA=YES;export KIX_ENABLE_JAVA
```

3. **LD_LIBRARY_PATH 環境変数が $JDKROOT/jre/lib/sparc ディレクトリを含んでいることを確認します。**

   アプリケーションプログラムをデバッグする場合、JVM オプションを追加します。

   詳細は、197 ページの「JVM 起動オプションの追加」を参照してください。

4. **領域を起動します。**

   Java サポートでの領域の開始に成功すると、領域が開始するトランザクション処理プログラムごとに次のようなメッセージが unikixmain.log ファイルに書き込まれます。

```
03/02/2005 09:59:35 unikixtran0 :KIX3400I Java Support Loaded
```

## JVM 起動オプションの追加

各 Sun MTP トランザクション処理プログラムは、ユーザー作成 JCICS コードを実行するために、それぞれのプロセス内で JVM を開始して所有します。環境変数 KIX_JVM_OPTIONS を設定することにより、その JVM の動作を変更できます。

KIX_JVM_OPTIONS を設定して、java コマンドへのコマンド行パラメータとしてスタンドアロン JVM に通常渡されるすべての JVM 起動フラグを含める必要があります。java コマンドが受け入れるフラグを確認するには、java -? コマンドを実行します。

たとえば、ガベージコレクションの冗長デバッグ情報を持つトランザクション処理プログラムによって所有されている JVM を開始するには、領域を開始する前に、KIX_JVM_OPTIONS を次のように設定します。

```
$ export KIX_JVM_OPTIONS='-verbose:gc'
```

JVM で使用するクラスパスやライブラリパスを設定することは許可されていません。たとえば、-Djava.class.path=... や -Djava.library.path=... は指定できません。

# 制限事項

この節では、Sun MTP で JCICS プログラムを実行するときの制限事項について説明します。

## データ永続性

Sun MTP の制御のもとで実行する各トランザクションは、1 つのトランザクション処理プログラムのプロセスで実行されます。各トランザクション処理プログラムは、トランザクションによって必要とされる JCICS プログラムを実行するために JVM も実行できます。

このマルチプロセスモデルでは、ユーザー作成 JCICS クラスの実行時に設定される Java の静的データは、そのクラスを使用する後続のトランザクションに対して、その値では有効にならない場合があります。これは、異なるトランザクション処理プログラムで実行している可能性があり、そのため異なる JVM を実行していることによるものです。

データを永続的に格納するには、標準の Sun MTP 機能 (たとえば一時記憶域キューまたは一時データキュー) を使用します。

# スレッド化の問題

Java のプログラミング環境は、本質的にマルチスレッド化されています。ただし、Sun MTP の制御のもとで実行する Java (JCICS) プログラムには、補足的な制限事項があります。

最も重要なのは、JCICS API の呼び出しはメインスレッドからしか行えないことです。すなわち、JCICS プログラムの `main()` メソッドが起動されるスレッドだけから、呼び出しが可能です。

# 第9章

# マップおよびマップセットの作成と管理

Sun MTP では、マップおよびマップセットを管理する方法を 2 つ提供します。

- 画面生成ユーティリティー (SGU) で可能な処理
  - 画面フォーマットツールを使用したネイティブフォーマットでのマップセットおよびマップの作成。
  - ネイティブファイルからの基本マッピングリソース (BMS) マクロ命令ファイルの生成。
  - BMS マクロファイルからの物理および記号定義マップのアセンブル。記号マップとは、アプリケーションプログラムにコピー可能なコードセクションで、端末から送受信するデータの形式を定義するデータ構造。このファイルは IBM CICS システムにもアセンブル用にエクスポートできます。
  - BMS マクロファイルを修正用に元のネイティブファイル形式に変換します。
- `kixbms` ユーティリティーで可能な処理
  - BMS マクロファイルからの物理および記号定義マップのアセンブル
  - BMS マクロファイルを修正用に元のネイティブファイル形式に変換します。

BMS マクロ命令については、IBM のマニュアルを参照してください。

この章の内容は、次のとおりです。

# SGU の開始

**注 –** TN3270 エミュレーションを使用している場合、3270 タイプの端末から `CSGU` トランザクションを呼び出すことはできません。この場合、ローカルクライアントが実行中である必要があります。

## ▼ SGU を開始する

- **ブランクの Sun MTP 画面で `CSGU` トランザクションを入力するか、開発システムのメインメニューで PF9 キーを押します。**

  図 9-2 に、SGU マップ管理画面を示します。

  図 9-1 に、SGU の機能マップを示します。

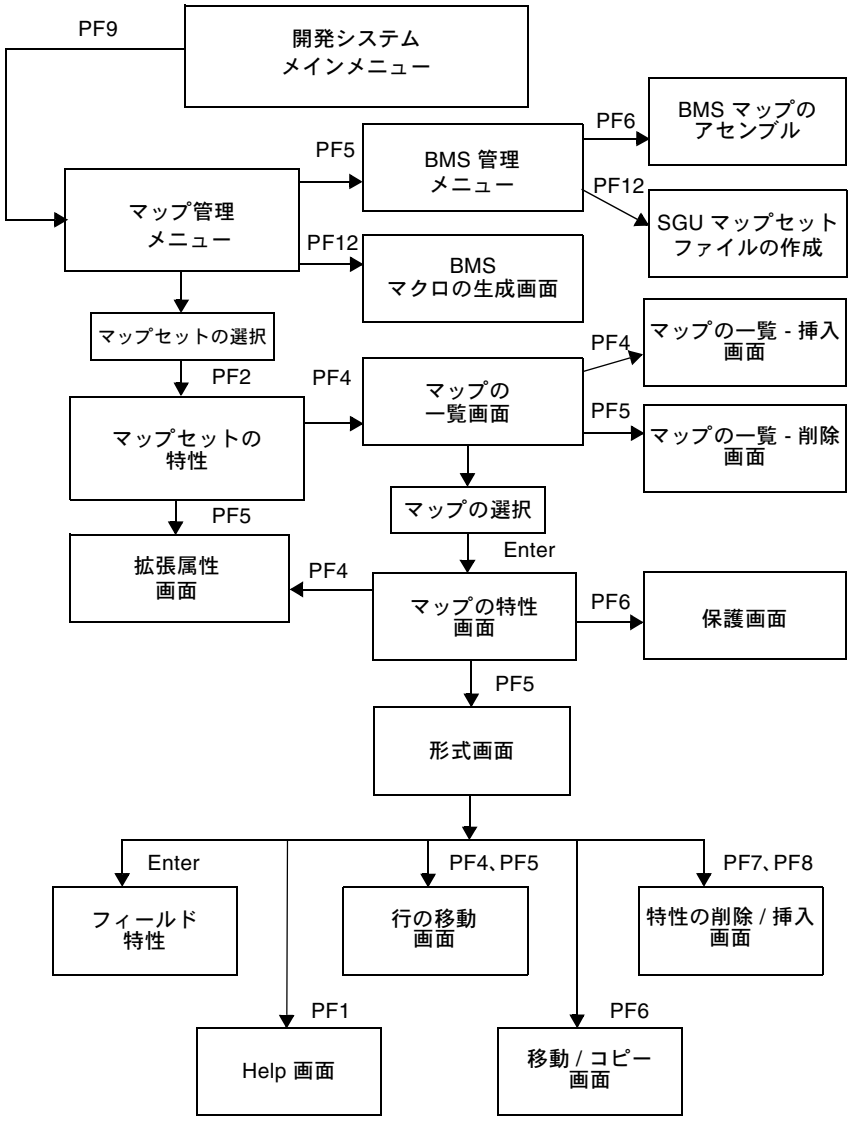


図 9-1　Screen Generation Utility—Menu Map

# マップ管理メニュー画面

SGU を開始すると、マップ管理メニュー画面が表示されます。マップ管理メニュー画面では、BMS マップセットの Sun MTP ネイティブ形式を示す拡張子 `.sgu` を持つファイルのみを表示します。これらのマップセットファイルが、マップセットに関連するマップの基本 BMS オプションを維持します。

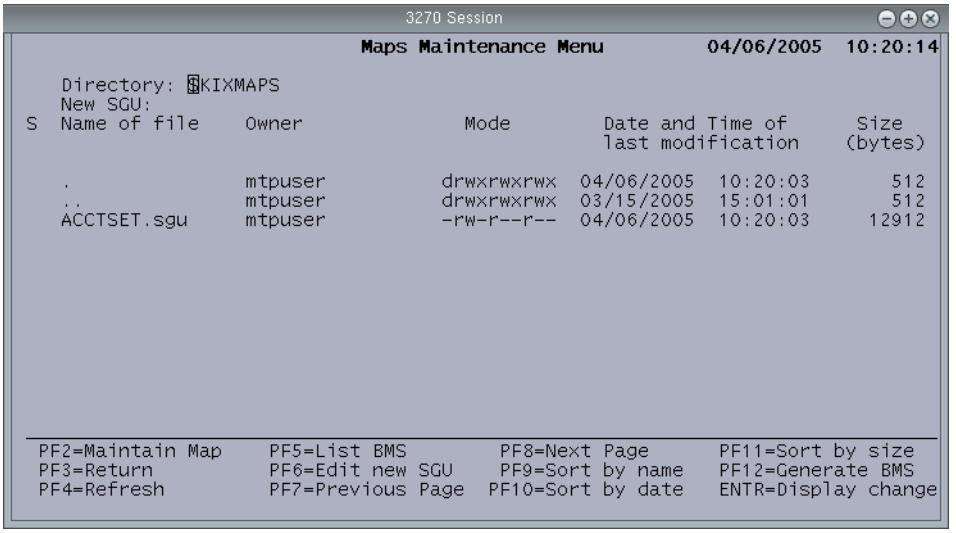


**図 9-2** Screen Generation Utility—Maps Maintenance Menu

この画面で変更できるのは、「Directory」および「New SGU」フィールドのみです。

| | |
|---|---|
| Directory | 50 文字までのディレクトリ名。最初の要素は環境変数にすることも可能で、そのあとにスラッシュで区切って他の複数のディレクトリを続けることができます。この値を変更した場合、Enter キーを押して変更を適用させる必要があります。 |
| New SGU | 新しい `.sgu` ファイルの名前で、英数字 7 文字までです。PF6 キーを押すと、新しい `.sgu` マップセットファイルが作成されます。 |
| S | ファイルを選択するために、ファイル名の左側の選択フィールドにカーソルを移動して、`S` または `s` を入力します。複数ファイルの選択が可能です。 |

マップ管理メニュー画面では、次のファンクションキーが使用できます。

| | |
|---|---|
| PF2 | 選択したファイルを作業ファイルとしてマップセットの特性画面を開きます。ファイルが選択されていない場合、カーソルが置かれているファイルが自動的に選択されます。このオプションは、既存のマップセットまたはそのマップの 1 つを変更するために使用します。マップセットの特性画面については、205 ページの「マップセットの特性」を参照してください。 |
| PF3 | マップ管理メニュー画面を閉じて、開発システムメインメニューまたはブランクのトランザクション画面に戻ります。 |
| PF4 | ファイル属性の変更のあと、画面のファイルリストの部分のデータを更新します。 |
| PF5 | 拡張子 `.bms` を持つファイルを一覧表示する BMS 管理メニューを表示します。詳細は、239 ページの「BMS ファイルの表示」を参照してください。 |
| PF6 | 「New SGU」フィールドで指定されたファイル名で新しい `.sgu` ファイルを作成し、デフォルトの値でマップセットの特性画面を表示します。詳細については、205 ページの「マップセットの特性」を参照してください。「New SGU」フィールドに名前を入力しないと、ファイル名の入力を求められます。ファイルの作成を取り消すには、エラーメッセージのあとに PF3 キーを押すか、Enter キーを押します。 |
| PF7 | 前の画面のファイルを表示します。ファイルの最初のページが表示されているときにこのキーを押すと、次のメッセージが表示されます。<br>`Beginning of data reached` |
| PF8 | 次の画面のファイルを表示します。ファイルの最後のページが表示されているときにこのキーを押すと、次のメッセージが表示されます。<br>`End of data reached` |
| PF9 | ディレクトリの内容をファイル名順にソートして再表示します。 |
| PF10 | ディレクトリの内容をファイルの変更日順にソートして再表示します。 |
| PF11 | ディレクトリの内容をファイルサイズ順にソートして再表示します。 |
| PF12 | `.sgu`ファイルから `.bms` マクロファイルを作成します。この画面の説明については、238 ページの「BMS マクロ命令の作成」を参照してください。 |
| Enter | 「Directory」フィールドに対するすべての変更を有効にします。ファイル表示域が一度消去され、新しいディレクトリ内のファイルが表示されます。 |

# マップセットの作成およびマップの追加

## ▼ 新しいマップセットを作成する

1. **マップ管理メニュー画面の「New SGU」フィールドに名前を入力して、PF6 キーを押します。**
2. **「`New file created`」というメッセージとともに、マップセットの特性画面が表示されます。**

   この画面については、205 ページの「マップセットの特性」を参照してください。
3. **デフォルトの特性を変更して、Enter キーを押します。**
4. **マップセットの拡張属性を定義するには、マップの特性画面で PF5 キーを押します。**

   拡張属性画面については、210 ページの「マップセットの拡張属性の定義」を参照してください。デフォルトの特性を変更して、Enter キーを押します。
5. **PF3 キーを押して、マップセットの特性画面に戻ります。**

## ▼ マップを追加する

1. **マップセットの特性画面で PF4 キーを押し、マップの一覧画面を表示します。**
2. **マップの一覧画面で PF4 キーを押して、マップを挿入します。**

   詳細は、214 ページの「マップセットへのマップ名の追加」を参照してください。

   マップを作成すると、次のことができます。

   - 特性のカスタマイズ。詳細は、216 ページの「マップの特性のカスタマイズ」を参照してください。
   - 画面のフォーマットと保存。詳細は、224 ページの「画面のフォーマット」を参照してください。
   - BMS マクロ命令ファイルの作成。詳細は、238 ページの「BMS マクロ命令の作成」を参照してください。
   - マクロ命令ファイルの物理マップへのアセンブル。詳細は、242 ページの「BMS マップのアセンブル」を参照してください。

# マップセットの特性

マップ管理メニュー画面で .sgu ファイルを選択して PF2 キーを押すか、「New SGU」フィールドに名前を入力して PF6 キーを押すと、図 9-3 のマップセットの特性画面が表示され、現在のマップセットの特性が示されます。この画面のデータは、BMS DFHMSD マクロ命令で定義されたデータと同じです。

マップセットが存在する場合、現在の特性が表示されます。新しいマップセットを定義する場合、BMS のデフォルトの特性が表示され、画面の応答領域に次のメッセージが表示されます。

```
KIX1451I New file created ...
```

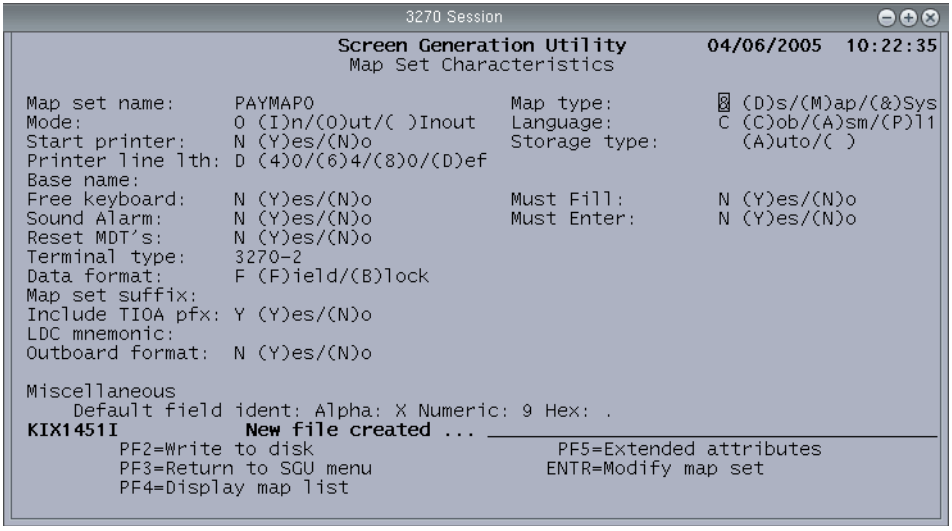


図 9-3　Screen Generation Utility—Map Set Characteristics

マップセットの特性画面では、次のファンクションキーが使用できます。

| ファンクションキー | 動作 |
|---|---|
| PF2 | ディスクにデータを書き込みます。現在作業している SGU マップセットファイルによって認識されるファイル名で、現在のマップセットの特性、マップセット、マップ、およびフィールドデータが保存されます。編集を終了して SGU 編集セッションを保存するときには、PF2 キーを押します。 |
| PF3 | マップ管理メニュー画面に戻ります。データを変更してもそれをディスクに書き込んでいない場合、次のメッセージが表示されます。<br>`SGU File has been modified ...press PF3 to quit or PF2 to save.` |
| PF4 | 現在マップセットに定義されているマップ名の一覧を表示します。この画面で利用できる機能については、214 ページの「マップセットへのマップ名の追加」を参照してください。 |
| PF5 | このマップセットの拡張属性を表示します。これらの属性は、このマップセットで定義されているすべてのマップのデフォルトです。 |
| Enter | マップセットの特性を変更します。データは検査されます。<br>データが有効な場合、その新しい情報が適用されて次のメッセージが表示されます。<br>`Data modified`<br>無効なデータは強調表示されます。カーソルが最初の無効なフィールド上に置かれ、次のメッセージが表示されます。<br>`Invalid data` |

次の表は、マップセットの特性画面上のデータフィールドおよび適用できる値を示します。マップセットの特性画面で Enter キーを押すと、これらのフィールドが検査されます。

SGU は、BMS DFHMSD、DFHMDI、および DFHMDF マクロ命令のほとんどのオプションをサポートします。ただし、Sun MTP BMS アセンブラおよび Sun MTP ランタイムモジュールは、これらのオプションの一部だけをサポートしています。サポートしているオプションについては、127 ページの「サポートされている BMS 機能」を参照してください。

**表 9-1** マップセットの特性画面のデータフィールド

| フィールド名 | 説明 |
|---|---|
| Map set name | マップセットの名前。表示専用フィールドです。 |
| Map type | BMS マクロがメインフレームにアップロードされたあと、アセンブラによって作成される出力のタイプ。<br>D: 記号記述マップを作成<br>M: 物理マップを作成<br>&: JCL に従って記号記述マップまたは物理マップを作成 |
| Mode | 生成する記憶領域タグ。<br>記号記述マップにのみ適用されます。<br>I: 入力フィールド名のみ作成<br>O: 出力フィールド名のみ作成<br>空白: 入力および出力の両方のフィールド名を作成 |
| Language | 作成する記号記述マップの形式を指定します。<br>記号記述マップにのみ適用されます。<br>C: COBOL 互換の記憶領域を適切なデータ名で作成<br>K: 記憶領域を記述する C 構造体のヘッダーファイルを作成<br>P: PL/I 互換の記憶領域を適切なデータ名で作成<br>A: アセンブラ互換の記憶領域を適切なデータ名で作成。このオプションは現在サポートされていません。 |
| Start printer | BMS 出力マッピング操作によって start printer コマンドをプリンタに送信するかどうかを指定します。<br>Y: start printer コマンドを送信します。<br>N: start printer コマンドを送信しません。 |
| Storage type | 記号記述マップの各マップを異なる記憶領域に配置するか、すべてのマップを同じ記憶領域に配置するかを指定します。自動記憶領域およびベース名は相互に排他的です。<br>記号記述マップにのみ適用されます。<br>A: 自動記憶領域。セット内の各マップが異なる記憶領域に配置されます。<br>空白: セット内のすべてのマップは同じ記憶領域に配置されます。 |

表 9-1　マップセットの特性画面のデータフィールド (続き)

| フィールド名 | 説明 |
|---|---|
| Printer line lth | プリンタの行の長さを指定します。<br>標準のプリンタでは、デフォルトのオプションのみをサポートします。<br>3270 タイプのデバイスでは、すべてのオプションをサポートします。<br>4: 40 文字の印刷行<br>6: 64 文字の印刷行<br>8: 80 文字の印刷行<br>D: デフォルトの印刷行の長さは、プリンタの設定によって決定されます。 |
| Base name | 記号記述マップに、複数のマップセットと同じ記憶領域ベースを使用します。ベース名は 1 ～ 8 文字 (最初の文字はアルファベット) です。<br>名前を空白にすると、プログラムで定義される各マップセットは一意の記憶域ベースで開始されます。 |
| Free keyboard | マップが端末に送信されたあと、キーボードを解放するかどうかを指定します。設定しない場合、そのフィールドはデフォルトで入力禁止状態に設定されます。<br>Y: 入力禁止状態をオフ - マップが送信されたあとにキーボードを解放します。<br>N: 入力禁止状態をオン - マップが送信されたあとにキーボードを解放しません。 |
| Must Fill | マップセット内のすべてのマップおよびフィールドに設定されるデフォルト値。<br>Sun MTP は、検査属性に対して記号マップでスペースを予約します。これを行わないと、検査属性はサポートされません。<br>Y: このオプションが設定されているデータフィールドには、データを完全に入力する必要があります。このフィールドにデータを完全に入力しないでカーソルを移動すると、入力禁止状態になります。<br>N: Must Fill チェックを実行しません。 |
| Sound Alarm | 端末に送信されたマップが表示されたあと、警告音を発するかどうかを指定します。<br>Y: 警告音を発します。<br>N: 警告音を発しません。 |
| Must Enter | マップセット内のすべてのマップおよびフィールドに設定されるデフォルト値を指定します。<br>Sun MTP は、検査属性に対して記号マップでスペースを予約しますが、その属性はサポートされません。<br>Y: このフラグが設定されている場合、フィールドへのデータの入力が必要です。このフィールドにデータを入力しないでカーソルを移動すると、入力禁止状態になります。<br>N: このフィールドにはデータを入力する必要はありません。 |
| Reset MDT's | 送信されたマップが端末に表示される前に、すべての変更データタグ (MDT) をリセットするかどうかを指定します。<br>Y: マップを表示する前に MDT をリセットします。<br>N: マップを表示する前に MDT をリセットしません。 |

表 9-1　マップセットの特性画面のデータフィールド (続き)

| フィールド名 | 説明 |
|---|---|
| Terminal type | 物理マップがサポートする端末のタイプ。このフィールドには、6 文字のコードを入力します。<br>フィールドへの入力が 3270-2 と同じ場合、Map set suffix に指定する接尾辞パラメータが設定されていないと判断して、BMS のアセンブル時に作成されるファイル名に M が追加されます。 |
| Data format | BMS 処理プログラムから送受信するデータの形式/解釈です。<br>F: フィールドデータ形式のデータ。<br>B: ブロック形式のデータ。 |
| Map set suffix | マップセット名の最後に追加される 1 文字の接尾辞。空白でもかまいません。この接尾辞は、異なるサイズまたは言語のマップを区別するために使用されます。 |
| Include TIOA pfx | 記号記述マップの生成時に 12 バイト TIOA 接頭辞を含めるかどうかを指定します。<br>Y: TIOA 接頭辞を含めます。<br>N: TIOA 接頭辞を含めません。 |
| LDC mnemonic | BMS 出力操作用の論理デバイスニーモニックを判断するために CICS が使用するコードです。TCAM および VTAM をサポートする 3600 端末およびバッチ論理ユニットに対してだけ使用されます。このフィールドには 2 文字のコードを入力するか、空白のままにします。 |
| Outboard format | 物理マップセットで外部フォーマットをサポートするかどうかを指定します。このオプションはサポートされていません。<br>Y: マップセットは外部フォーマットをサポートします。<br>N: マップセットは外部フォーマットをサポートしません。 |
| Default field ident | |
| Alpha | リピート記号を識別します。フィールドにデフォルト値を設定するとき、リピート記号は英数字の無保護フィールドとして認識されます。詳細は、228 ページの「フィールド属性のデフォルト」を参照してください。 |
| Numeric | リピート記号を識別します。フィールドにデフォルト値を設定するとき、リピート記号は数字の無保護フィールドとして認識されます。詳細は、228 ページの「フィールド属性のデフォルト」を参照してください。 |
| Hex | リピート記号を識別します。フィールドにデフォルト値を設定するとき、リピート記号は 16 進数の初期値を持つものとして認識されます。228 ページの「フィールド属性のデフォルト」を参照してください。 |

# マップセットの拡張属性の定義

拡張属性画面では、マップセットの拡張属性を定義します。

## ▼ マップセットの拡張属性を定義する

1. **マップセットの特性画面で PF5 キーを押します。**

   次の図は、拡張属性画面を示します。

---

**注 –** Sun MTP は、すべての拡張属性に対してマップでスペースを予約しますが、Color および Hilight の拡張属性だけをサポートします。

---

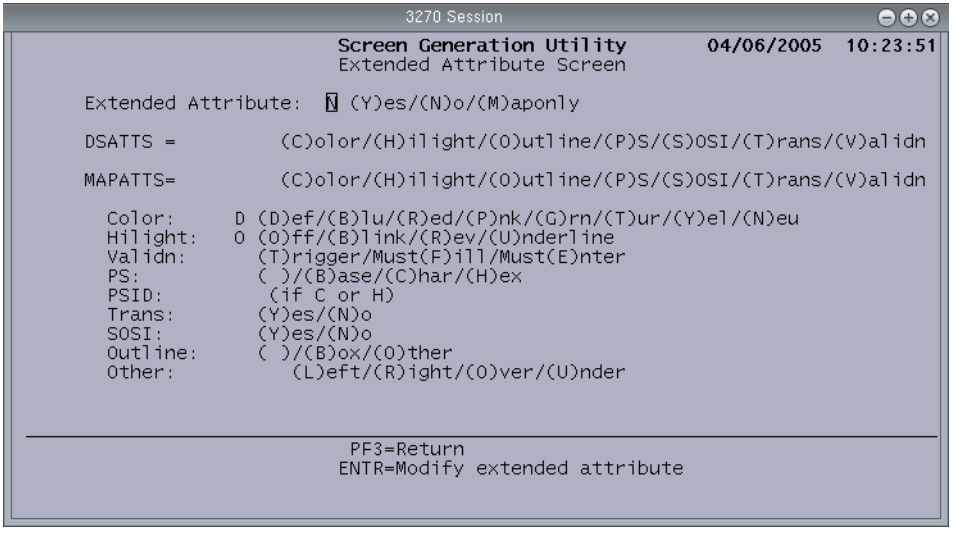


**図 9-4** Screen Generation Utility—Extended Attribute Screen

2. **拡張属性を指定します。**
3. **Enter キーを押して変更を確認します。**

次の表に、拡張属性画面のフィールドとそのオプションを示します。

**表 9-2　マップセットの拡張属性画面のフィールド**

| フィールド名 | 説明 |
|---|---|
| Extended Attributes | マップセットに対して拡張属性を指定できますが、表示されるのは 129 ページの「拡張属性」に示される属性のみです。<br>N: 拡張属性を使用しません。あるいは拡張属性が「DSATTS」および「MAPATTS」フィールドによって指定されることを示します。<br>Y: 物理マップセットと記号記述マップセットの両方で拡張属性を使用します。このフィールドを Y に設定すると、次のように設定する場合と同じ意味を持ちます。<br>DSATTS=(COLOR,HILIGHT,VALIDN,PS)<br>および<br>MAPATTS=(COLOR,HILIGHT,VALIDN,PS)<br>M: 物理マップセットのみで拡張属性を使用します。このフィールドを M に設定すると、次のように設定する場合と同じ意味を持ちます。<br>MAPATTS=(COLOR,HILIGHT,VALIDN,PS) |
| DSATTS | 記号記述マップで使用する絶対属性。7 文字までの文字列を指定できます。各文字は特定の拡張属性を表します。このフィールドで属性を指定すると、「MAPATTS」フィールドにも表示されます。<br>C: Color<br>H: Hilight<br>O: Outline<br>P: PS<br>S: SOSI<br>T: Transp<br>V: Validn |
| MAPATTS | マップセットで使用する絶対属性。7 文字までの文字列を指定できます。各文字は特定の拡張属性を表します。<br>C: Color<br>H: Hilight<br>O: Outline<br>P: PS<br>S: SOSI<br>T: Transp<br>V: Validn |

表 9-2　マップセットの拡張属性画面のフィールド (続き)

| フィールド名 | 説明 |
|---|---|
| Color | マップセット内で定義されたマップおよびフィールドに割り当てる拡張属性の色のデフォルト。Sun MTP では、割り当てる色属性に対して記号マップでスペースを予約し、特定のデバイスに対してサポートします。詳細については、129 ページの「拡張属性」を参照してください。<br>D: デフォルト<br>B: 青<br>R: 赤<br>P: ピンク<br>G: 緑<br>T: 青緑<br>Y: 黄<br>N: 無色 |
| Hilight | マップセット内で定義されたマップおよびフィールドに割り当てるデフォルトの強調表示属性。Sun MTP は、強調表示属性に対して記号マップでスペースを予約します。強調表示属性は、さまざまなデバイスでサポートされます。詳細については、129 ページの「拡張属性」を参照してください。<br>O: オフ<br>B: 点滅<br>R: 反転表示<br>U: 下線 |
| Validn | 自動検査を有効にするかどうかを指定します。<br>T: トリガー<br>F: 完全入力が必要<br>E: 入力が必要 |
| PS | プログラム式記号を使用するかどうかを指定します。<br>B: 基本記号セットを使用。<br>C: プログラム式記号セットを使用し、プログラム式記号識別子を 1 文字で指定。<br>H: プログラム式記号セットを使用し、プログラム式記号識別子を 2 桁の 16 進数で指定。 |
| PSID | 指定される PS オペランドに従って、プログラム式記号セット識別子を指定します。<br>C: 1 文字の EBCDIC 文字。<br>H: 2 桁の 16 進数。 |
| Trans | フィールドの背景。<br>Y: 透明のフィールド。<br>N: 不透明のフィールド。 |

表 9-2　マップセットの拡張属性画面のフィールド (続き)

| フィールド名 | 説明 |
|---|---|
| SOSI | フィールドに EBCDIC および DBCS データの両方を含めることができます。DBCS データは、データの最初のシフトアウト (SO) 文字と、データの最後のシフトイン (SI) 文字によって指定されます。<br>Y: フィールドに EBCDIC および DBCS のデータが含まれます。<br>N: フィールドには EBCDIC データのみが含まれます。 |
| Outline | フィールドの周りを線で囲むか、フィールドの上、下、左、右のどれかに線を引きます。<br>B: フィールドの周りを線で囲みます。<br>O: 「Other」フィールドで線を引く場所を定義します。 |
| Other (その他) | このフィールドは、「Outline」オプションが O に設定されている場合に有効です。<br>L: フィールドの左側に線を引きます。<br>R: フィールドの右側に線を引きます。<br>O: フィールドの上に線を引きます。<br>U: フィールドの下に線を引きます。 |

# マップセットへのマップ名の追加

マップの一覧画面では、マップセットに対してマップセット名の追加、変更、および削除ができます。この画面には、マップセットに対して定義されたマップ名が一覧表示されます。新しいマップセットにマップを追加しても、この画面にマップ名は表示されません。マップセットを変更する場合、画面には 170 までのマップ名を一覧表示できます。

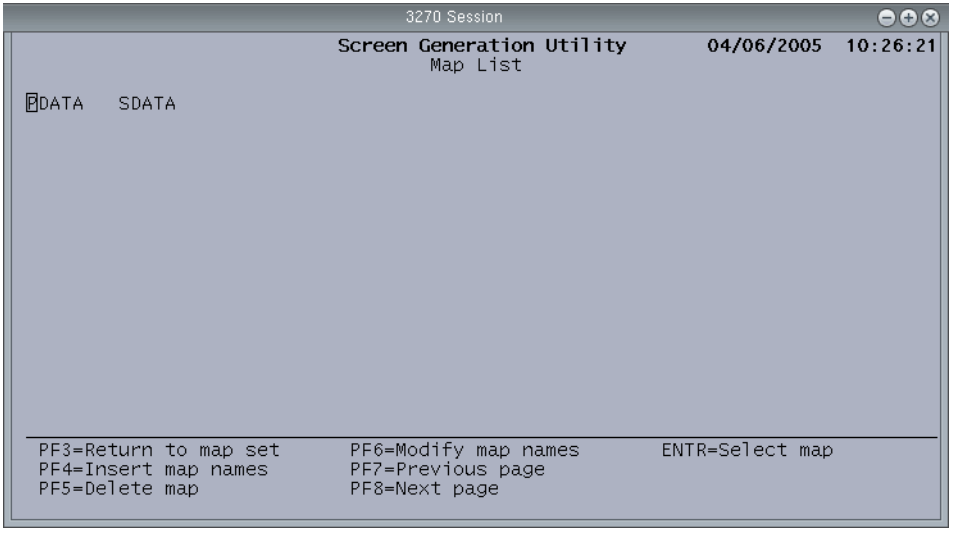


**図 9-5** Screen Generation Utility—Map List 画面

マップの一覧画面では、次のファンクションキーが使用できます。

| ファンクションキー | 動作 |
|---|---|
| PF3 | マップセットの特性画面に戻ります。 |
| PF4 | マップの挿入画面を表示します。この画面では、リストにマップ名を挿入できます。カーソルがあるマップ名の前に新しい名前が挿入されます。<br>• リストの最初にマップ名を挿入するには、リスト内の最初のマップ名にカーソルを置きます。<br>• リストの最後にマップ名を挿入するには、リストの最後のマップ名の後ろにカーソルを置きます。 |
| PF5 | 削除するマップをマークし、図 9-14 のマップの削除画面を呼び出します。マップセットからマップ名を削除するには、削除するマップ名にカーソルを置いて PF5 キーを押します。詳細は、237 ページの「マップの削除」を参照してください。 |
| PF6 | 現在定義されているマップ名を変更します。マップ名が一意であるかどうかが検査されます。 |

| ファンクションキー | 動作 |
|---|---|
| PF7 | マップ名の前のページを表示します。マップ名の最初のページを表示している場合、次のメッセージが表示されます。<br>`Beginning of data reached` |
| PF8 | マップ名の次のページを表示します。マップ名の最後のページを表示している場合、次のメッセージが表示されます。<br>`End of data reached` |
| Enter | マップの特性画面に戻ります (216 ページの「マップの特性のカスタマイズ」を参照)。現在のカーソルが置かれているマップ名が処理の対象となります。 |

## ▼ 新しいマップを追加する

1. **マップの一覧画面で PF4 キーを押します。**

   マップの一覧画面がブランクの状態で表示されます。

2. **新しいマップの名前を入力します。**

   1 つのマップセットに 255 までのマップを定義できます。新しいマップは現在のマップセットの特性を引き継ぎます。個々のマップでその特性を上書きすることができます。次の節では、個々のマップの特性をカスタマイズする方法を説明します。

3. **Enter キーを押して、マップを追加します。**

**注 –** マップ名は、各マップセット内で一意である必要があります。

# マップの特性のカスタマイズ

マップの特性画面では、1 つのマップの特性をカスタマイズできます。

## ▼ マップの特性をカスタマイズする

1. **マップの一覧画面で、カスタマイズするマップの名前にカーソルを置き、Enter キーを押します。**

   図 9-6 のマップの特性画面が表示されます。

   マップの特性画面のデータは、BMS `DFHMDI` マクロ命令で定義されるデータと同じです。

   マップがすでに定義されている場合、その特性が表示されます。マップセットにマップが新しく挿入されると、デフォルトの特性が表示されます。

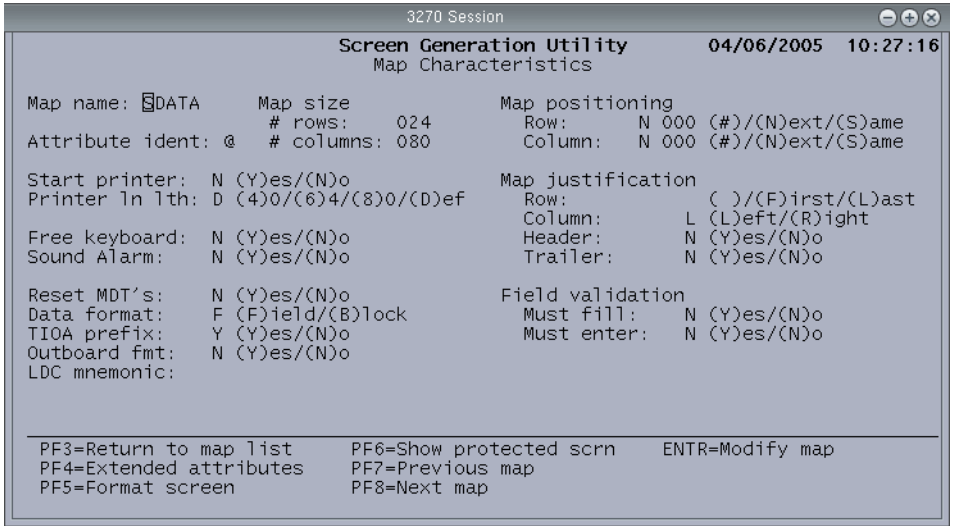


**図 9-6** Screen Generation Utility—Map Characteristics 画面

2. **必要に応じて、画面の値を変更します。**

   詳細は、表 9-3 を参照してください。

3. **Enter キーを押して、マップを変更します。**

マップの特性画面では、次のファンクションキーが使用できます。

| ファンクションキー | 動作 |
|---|---|
| PF3 | マップの一覧画面に戻ります。 |
| PF4 | マップの拡張属性画面を表示します。 |
| PF5 | フォーマット画面を表示します。マップの現在の定義が表示された全画面表示になります。この機能は画面レイアウトの定義に使用します。詳細は、224 ページの「画面のフォーマット」を参照してください。 |
| PF6 | マップの全画面バージョンを、保護、無保護、数字、強調表示フィールドなどをすべて備えたエンドユーザーに表示されるのと同じ形式で表示します。サンプルデータを入力して、データ入力フィールドのテストが可能です。テストで入力したデータは保存されません。 |
| PF7 | マップセットの前のマップを表示します。マップセットの最初のマップを表示している場合、次のメッセージが表示されます。<br>`Beginning of data reached` |
| PF8 | マップセットの次のマップを表示します。マップセットの最後のマップを表示している場合、次のメッセージが表示されます。<br>`End of data reached` |
| Enter | マップの特性を変更します。画面のデータは検査されます。データが有効な場合、その新しい情報が適用されて次のメッセージが表示されます。<br>`Data modified`<br>データが検査で不合格になると、無効なフィールドが強調表示され、カーソルが最初の無効なフィールドに置かれます。エラーが検出された場合、次のメッセージが表示されます。<br>`Invalid data` |

次の表は、マップの特性画面上のデータフィールドおよび適用できる値を示します。マップの特性画面で Enter キーを押すと、フィールドが検査されます。

**表 9-3**　マップの特性画面のデータフィールド

| フィールド名 | 説明 |
|---|---|
| Map name | マップの一意の名前 (1 ～ 7 文字)。最初の文字はアルファベットです。 |
| Map size | |
| # rows | マップが使用する画面の行数 (1 ～ 240)。 |
| # columns | マップが使用する画面の列数 (1 ～ 240)。 |
| Map positioning | |
| Row: N | マップを配置する位置の行番号 (1 ～ 240)。行の位置指定インジケータが # の場合のみ指定できます。 |

表 9-3　マップの特性画面のデータフィールド (続き)

| フィールド名 | 説明 |
|---|---|
| Row: *indicator* | 行の位置。<br>#: 行の位置と「Map justification Row」フィールドに従ってマップを配置します。<br>N: 次に利用できる行にマップを配置します。論理メッセージが構築されるときのみ適用できます。<br>S: スペースが利用できる場合に、最後のマップセットと同じ行にマップを配置します。または、完全に利用できる次の行にマップを配置します。論理メッセージが構築されるときのみ適用できます。 |
| Column: S | マップを配置する位置の列番号 (1 ～ 240)。「Column: indicator」が # の場合のみ指定できます。 |
| Column: *indicator* | 列の位置。<br>#: 列の位置と「Map justification Column」フィールドに従ってマップを配置します。<br>N: 列の位置揃えに従って、次に利用できる左または右の列にマップを配置します。論理メッセージが構築されるときのみ適用できます。<br>S: このマップと同じ列の位置および列の位置揃えを指定した、最後の送信マップと同じ列にマップを配置します。論理メッセージが構築されるときのみ適用できます。 |
| Attribute ident | フォーマット画面がフィールドの最初を識別する処理を行うときに使用するコード。有効な文字は次のとおりです。<br>!, @, #, $, %, *, + |
| Start printer | BMS 出力マッピング操作によって start printer コマンドをプリンタへ送信するマップセットオプションより優先されます。<br>Y: start printer コマンドを送信します。<br>N: start printer コマンドを送信しません。 |
| Printer ln lth | プリンタの行の長さを指定するマップセットオプションより優先されます。<br>3270 タイプのデバイスでは、すべてのオプションをサポートします。<br>4: 40 文字の印刷行<br>6: 64 文字の印刷行<br>8: 80 文字の印刷行<br>D: デフォルトの印刷行の長さは、プリンタの設定によって決定します。 |
| Free keyboard | マップが送信されたあとにキーボードを解放するマップセットオプションより優先されます。<br>Y: 入力禁止状態をオフ - マップが送信されたあとにキーボードを解放します。<br>N: 入力禁止状態をオン - マップが送信されたあとにキーボードを解放しません。 |
| Sound alarm | マップが表示されたあとに警告音を発するマップセットオプションより優先されます。<br>Y: 警告音を発します。<br>N: 警告音を発しません。 |

表 9-3　マップの特性画面のデータフィールド (続き)

| フィールド名 | 説明 |
|---|---|
| Reset MDT's | 端末でマップが表示される前にすべての変更データタグ (MDT) をリセットするマップセットオプションより優先されます。<br>Y: マップを表示する前に MDT をリセットします。<br>N: マップを表示する前に MDT をリセットしません。 |
| Data format | BMS 処理プログラムから送受信するデータの形式/解釈に対するマップセットオプションより優先されます。<br>F: フィールドデータ形式のデータ。<br>B: ブロック形式のデータ。 |
| TIOA prefix | 記号記述マップの生成時に 12 バイト TIOA 接頭辞を含めるかどうかのマップセットオプションより優先されます。<br>Y: TIOA 接頭辞を含めます。<br>N: TIOA 接頭辞を含めません。 |
| Map justification | |
| Row | マップを配置する行。<br>空白: 行位置に従ってマップを配置します。<br>F: 新しいページの最初にマップを配置します。論理メッセージが構築される場合のみ適用されます。<br>L: 現在のページの最後にマップを配置します。論理メッセージが構築される場合のみ適用されます。 |
| Column | マップを調整する列の余白。<br>L: 左余白でマップを調整します。<br>R: 右余白でマップを調整します。 |
| Header | 論理メッセージを構築する間、オーバーフロー状態を終了しないでマップを使用できるようにします。<br>Y: ヘッダーマップ。<br>N: ヘッダーマップ以外。 |
| Trailer | 論理メッセージを構築する間、オーバーフロー状態を終了しないでマップを使用できるようにします。<br>Y: トレーラマップ。<br>N: トレーラマップ以外。 |

表 9-3　マップの特性画面のデータフィールド (続き)

| フィールド名 | 説明 |
|---|---|
| Field validation | |
| Must fill | マップセット内のすべてのマップおよびフィールドに設定されるデフォルト値に対するマップセットオプションより優先されます。<br>Y: このオプションが設定されているデータフィールドには、データを完全に入力する必要があります。このフィールドにデータを完全に入力しないでカーソルを移動すると、入力禁止状態になります。<br>N: Must Fill チェックを実行しません。 |
| Must enter | マップセット内のすべてのマップおよびフィールドに設定されるデフォルト値に対するマップセットオプションより優先されます。<br>Y: このフラグが設定されているフィールドには、データの入力が必要。このフィールドにデータを入力しないでカーソルを移動すると、入力禁止状態になります。<br>N: このフィールドにはデータを入力する必要はありません。 |
| Outboard fmt | 物理マップセットで外部フォーマットをサポートするかどうかを指定するマップセットオプションより優先されます。このオプションはサポートされていません。<br>Y: マップセットは外部フォーマットをサポートします。<br>N: マップセットは外部フォーマットをサポートしません。 |
| LDC mnemonic | このフィールドの 2 文字のマップセットコードまたは空白コードより優先されます。 |

# 行および列の処理

マップの特性画面の「Map Size」の「# rows」フィールドおよび「# columns」フィールドを変更するには、特殊な方法で行う必要があります。これらのフィールドでは、フォーマットするデータ入力画面の最大サイズを決定します。

たとえば、「# rows」が 2 で「# columns」が 80 の場合、画面のフォーマット処理時に画面の 2 行だけでデータを定義できるようになります。「# rows」が 24 で「# columns」が 80 の場合、画面全体をフォーマットできるようになります。「# rows」が 10 で「# columns」が 80 の場合、最初の 10 行の領域のみがフォーマットデータを含むことができます。

すでにフォーマットされている画面でこれらのフィールドを変更すると、画面フォーマットに影響します。たとえば、前の例の列数を 80 から 40 に減らすと、41 列以降に定義されていたすべてのフィールドは削除されます。40 列より前に始まり 40 列より後ろで終わるすべてのフィールドは、40 列で切り捨てられます。

## ▼ マップの拡張属性を定義する

1. **マップの特性画面で PF4 キーを押し、マップの拡張属性画面を表示します。**

   詳細は、図 9-7 を参照してください。

---

**注 –** これは、図 9-5 のマップセットの特性画面で PF5 キーを押して表示される画面と同じです。

---

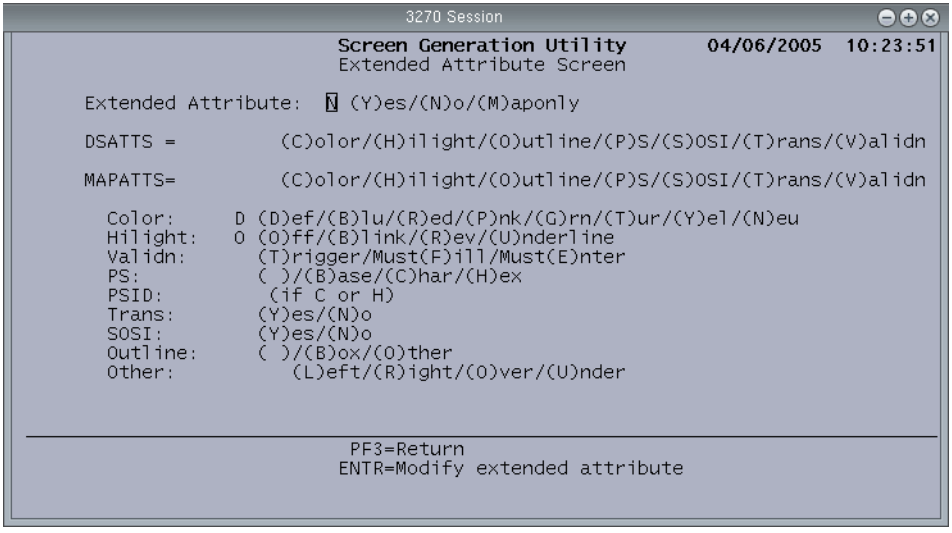


**図 9-7** Screen Generation Utility—Extended Attribute Screen

2. **属性を変更します。**

   属性のそれぞれのフィールドについては、表 9-4 を参照してください。

   この画面で指定したすべてのオプションは、マップセットに指定されたオプションより優先されます。

3. **Enter キーを押して、マップを変更します。**

表 9-4　マップの拡張属性画面のフィールド

| フィールド名 | 説明 |
|---|---|
| Extended Attribute | マップに対して拡張属性を指定できますが、129 ページの「拡張属性」に示す属性のみが表示されます。<br>N: 拡張属性をサポートしません。あるいは拡張属性を「DSATTS」および「MAPATTS」フィールドで指定します。<br>Y: 物理マップおよび記号記述マップで拡張属性をサポートします。このフィールドを Y に設定すると、次のように設定する場合と同じ意味を持ちます。<br>DSATTS=(COLOR,HILIGHT,VALIDN,PS)<br>および<br>MAPATTS=(COLOR,HILIGHT,VALIDN,PS)<br>M: 物理マップだけで拡張属性を使用します。このフィールドを M に設定すると、次のように設定する場合と同じ意味を持ちます。<br>MAPATTS=(COLOR,HILIGHT,VALIDN,PS) |
| DSATTS | 記号記述マップで使用する絶対属性。7 文字までの文字列を指定します。各文字は特定の拡張属性を表します。このフィールドで属性を指定すると、「MAPATTS」フィールドにも表示されます。<br>C: Color<br>H: Hilight<br>O: Outline<br>P: PS<br>S: SOSI<br>T: Transp<br>V: Validn |
| MAPATTS | マップで使用する絶対属性。7 文字までの文字列を指定します。各文字は特定の拡張属性を表します。<br>C: Color<br>H: Hilight<br>O: Outline<br>P: PS<br>S: SOSI<br>T: Transp<br>V: Validn |

表 9-4　マップの拡張属性画面のフィールド (続き)

| フィールド名 | 説明 |
|---|---|
| Color | マップ内で定義されたマップおよびフィールドに割り当てる拡張属性の色のデフォルト。Sun MTP は色属性の割り当てに対するスペースを記号マップで予約し、特定のデバイスに対するサポートを提供します。詳細については、129 ページの「拡張属性」を参照してください。<br>D: デフォルト<br>B: 青<br>R: 赤<br>P: ピンク<br>G: 緑<br>T: 青緑<br>Y: 黄<br>N: 無色 |
| Hilight | マップ内で定義されたマップおよびフィールドに割り当てるデフォルトの強調表示属性。Sun MTP は記号マップで強調表示属性に対するスペースを予約します。強調表示属性は、さまざまなデバイスでサポートされます。詳細は、129 ページの「拡張属性」を参照してください。<br>O: オフ<br>B: 点滅<br>R: 反転表示<br>U: 下線 |
| Validn | 自動検査を有効にするかどうかを指定します。<br>T: トリガー<br>F: 完全入力が必要<br>E: 入力が必要 |
| PS | プログラム式記号を使用するかどうかを指定します。<br>B: 基本記号セットを使用。<br>C: プログラム式記号セットを使用。プログラム式記号識別子を 1 文字で指定します。<br>H: プログラム式記号セットを使用。プログラム式記号識別子を 2 桁の 16 進数で指定します。 |
| PSID | 指定される PS オペランドに従って、プログラム式記号セット識別子を指定します。<br>C: 1 文字の EBCDIC 文字。<br>H: 2 桁の 16 進数。 |
| Trans | フィールドの背景が透明か不透明かを指定します。<br>Y: 透明のフィールド。<br>N: 不透明のフィールド。 |

表 9-4　マップの拡張属性画面のフィールド (続き)

| フィールド名 | 説明 |
|---|---|
| SOSI | フィールドに EBCDIC および DBCS データの両方を含めることができます。DBCS データは、データの最初のシフトアウト (SO) 文字と、データの最後のシフトイン (SI) 文字によって指定されます。<br>Y: フィールドに EBCDIC および DBCS のデータが含まれます。<br>N: フィールドには EBCDIC データのみが含まれます。 |
| Outline | フィールドの周りを線で囲むか、フィールドの上、下、左、右のどれかに線を引きます。<br>B: フィールドの周りを線で囲みます。<br>O: 「Other」フィールドで枠を定義します。 |
| Other (その他) | このフィールドは、「Outline」オプションが O に設定されている場合に有効です。<br>L: フィールドの左側に線を引きます。<br>R: フィールドの右側に線を引きます。<br>O: フィールドの上に線を引きます。<br>U: フィールドの下に線を引きます。 |

# 画面のフォーマット

フォーマット画面の機能により、画面にデータを入力してマップを設計できます。マップの特性画面の PF5 キーを押して、フォーマット機能を開きます。

画面がフォーマットされている場合、現在のフォーマットデータが表示されます。初めて画面をフォーマットする場合、画面は空白の状態です。マップセットの各マップで 1,023 までのフィールドを定義できます。

キーボードのすべてのカーソル制御は、通常どおり動作します。現在割り当てられているマップの属性は、画面のフォーマット時に適用されます。そのため、マップの特性画面の全画面より小さいマップサイズを指定した場合、画面のその部分のみがキーボード入力を受け入れます。画面の残りの部分は保護され、データの入力を受け付けません。

フォーマットできる画面は、次のとおりです。

- 3270 モデル 2 の画面 (24x80 文字)
- 3270 モデル 4 の画面 (43x80 文字)
- 3270 モデル 5 の画面 (27x132 文字)

画面のフォーマットに使用する物理端末は、少なくともフォーマットする画面より大きい必要があります。たとえば、モデル 5 端末は、モデル 2 または モデル 5 の画面のフォーマットに使用できますが、モデル 4 の画面には使用できません。

次のファンクションキーは、ブランクのフォーマット画面に表示されませんが、画面フォーマット処理時に使用できます。

| ファンクションキー | 動作 |
|---|---|
| PF1 | 226 ページの「フォーマットのヘルプ画面を表示する」に示すヘルプ画面を開きます。 |
| PF3 | マップの特性画面に戻ります。 |
| PF4 | 行を *n* 文字分左または右に移動します。詳細は、229 ページの「行を右または左に移動する」を参照してください。 |
| PF5 | 画面を *n* 文字分左または右に移動します。この機能は、行の移動機能と同じです。この画面で使用できるキーは、次のとおりです。<br>• PF7: 画面全体を *n* 文字分左に移動します。<br>• PF8: 画面全体を *n* 文字分右に移動します。<br>• PF3: データを移動しないでフォーマット画面に戻ります。 |
| PF6 | 現在の行を移動またはコピーします。詳細は、230 ページの「現在の行を移動またはコピーする」を参照してください。 |
| PF7 | *n* 文字を削除または挿入します。詳細は、231 ページの「文字を挿入または削除する」を参照してください。 |
| PF8 | *n* 行を削除または挿入します。この機能は、文字の削除/挿入機能と同じです。この画面で使用できるキーは、次のとおりです。<br>• PF7: 現在の行から *n* 行を削除します。残りのすべての行は、画面の上部に移動します。<br>• PF8: 現在の行の前に空白の *n* 行を挿入します。現在の行以降のすべての行は、画面の下部に移動します。マップの下からはみ出した行は削除されます。<br>• PF3: 2 つの画面で同じ機能を持たせます。つまりデータを変更しないでフォーマット画面に戻ります。 |
| PF9 | 現在の行にあるすべてのフィールドを中央揃えにします。最初のフィールドの前にある空白文字数および最後のフィールドのあとにある空白文字数は、左右の余白に均等に配置されます。空白文字数が奇数の場合、余分な空白文字は右側に追加されます。 |
| Enter | フォーマット画面のデータを適用し、フィールドの特性画面を表示します。画面フォーマットのデータを保存するには、Enter キーを押す必要があります。画面のフォーマットを変更したあと、Enter キーを押さないで PF3 キーを押すと変更が失われます。 |

## ▼ フォーマットのヘルプ画面を表示する

- **フォーマット画面で PF1 キーを押すと、ヘルプ画面が表示されます。**

  PF3 キーを押して、フォーマット画面に戻ります。

  この画面には、各 PF キーおよび Enter キーの機能についての説明があり、さらに現在の文字、行、およびフィールドを識別する方法についての説明があります。

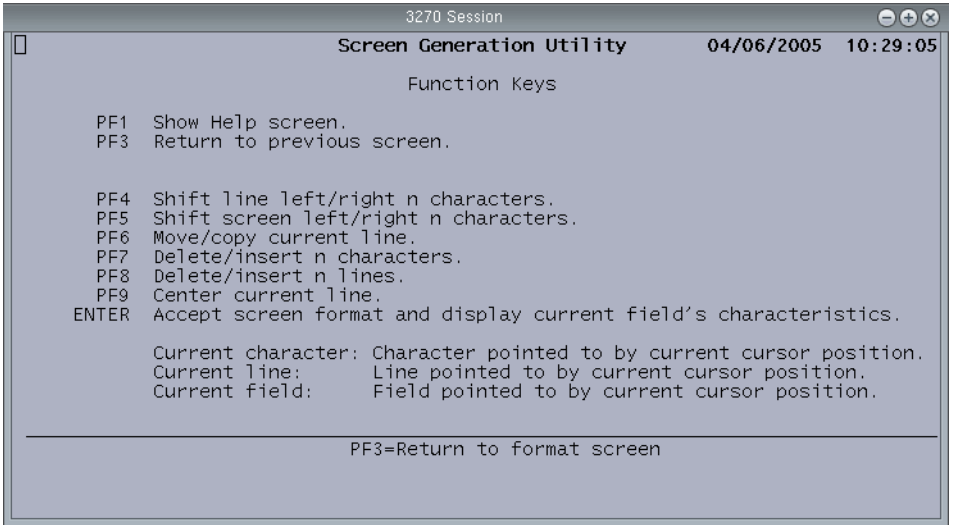


**図 9-8** Format Screen のヘルプ

# 属性識別子

属性識別子は、フィールドの始まりを識別する一意の文字です。データフィールドのすぐ前に置かれ、画面のその位置に物理的に配置されます。また、フィールドへの属性の自動割り当てに使用すると便利です。属性識別子は、マップに特性を割り当てるときに定義します。デフォルトの属性識別子は @ です。詳細は、216 ページの「マップの特性のカスタマイズ」を参照してください。

属性識別子には、フォーマットされた画面にリテラル定数として現れる文字を選択できません。

図 9-9 に、属性識別子として @ を持つフォーマットされた画面を示します。フォーマットされた画面を表示するには、始めにマップの一覧画面からマップを選択します。マップの特性画面が表示されたら、PF5 キーを押します。この例では、26 のフィールドが定義されています。

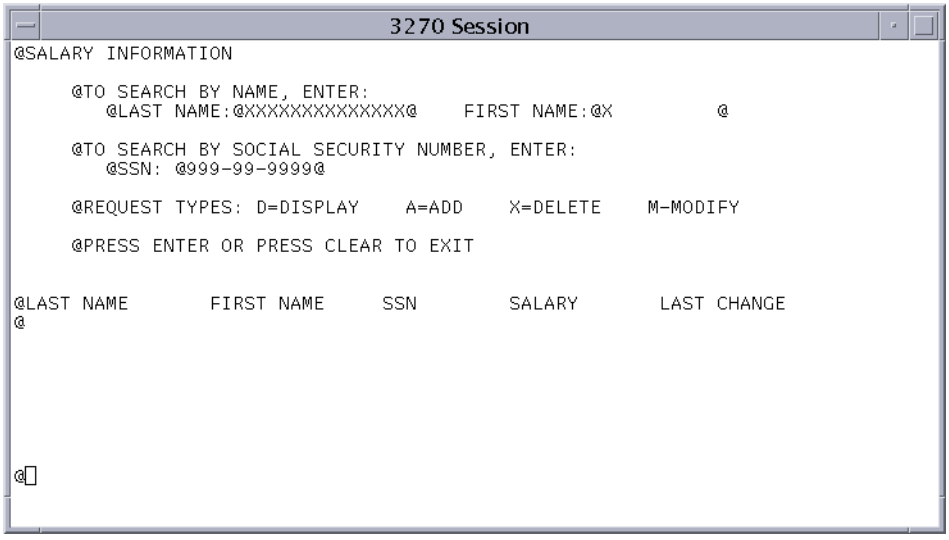


**図 9-9** Formatted Screen の例

属性識別子が連続する場合、または行の最後の文字が属性識別子である場合に Enter キーを押すと、エラーメッセージが表示されます。ただし、マップが端末の最大幅で表示されていて、フィールドを次の行に折り返す場合には、行の最後の文字が属性識別子になることもあります。

キーボードの Insert および Delete キーを使用して画面の属性識別子を移動すると、以前に割り当てたフィールド特性に次のような影響を与えることがあります。

- そのフィールドで、割り当てられたフィールド属性をデフォルトから変更している場合、フィールドはデフォルトに戻されます。これは、属性識別子の位置によりフィールドが識別されるために発生します。詳細は、228 ページの「フィールド属性のデフォルト」を参照してください。
- 属性識別子を画面のほかの位置に移動した場合、画面フォーマットは、これを、削除されるフィールドまたはほかの追加されるフィールドとしてしか認識できません。

画面フォーマットが終了するまでは、フィールド特性を変更しないでください。

フィールドを移動してその変更された値を保持するには、224 ページの「画面のフォーマット」に示すファンクションキーを使用します。

# フィールド属性のデフォルト

新しいフィールドを定義する場合、属性を手動で割り当てるか、SGU によって自動的に割り当てることができます。属性を手動で割り当てる場合、古いフィールド属性は変更されません。新しいフィールド属性の自動割り当てを実行するには、マップセットの特性画面の「Default field ident」フィールドを使用します。フォーマット画面のフィールドに入力するすべての文字が、「Default field ident」フィールドに定義されている 3 文字のいずれかと一致する場合、フィールド属性は次のように設定されます。

1 つのマップセットに 1 つのデフォルトフィールド識別子だけを定義できます。明示的に変更しないかぎり、「Default field ident」の文字は次のとおりです。

| | |
|---|---|
| X | 英数字 |
| 9 | 数字 |
| . | 16 進数 |

**すべての文字がデフォルトの英数字のフィールド識別子と一致する**

- フィールドの属性は、無保護、英数字、および明るいに設定されます。文字は連続している必要があり、次のフィールドと分けるには空白文字のみ使用します。
- フィールドの長さに連続した文字の数が設定されます。
- フィールド名には疑問符が設定されます。これにより、このデータ入力フィールドに名前を割り当てる必要があることを知らせます。

**すべての文字がデフォルトの数字のフィールド識別子と一致する**

- フィールドの属性は、無保護、数字、および明るいに設定されます。文字は連続している必要があり、次のフィールドと分けるには空白文字のみ使用します。
- フィールドの長さに連続した文字の数が設定されます。
- フィールド名には疑問符が設定されます。これにより、このデータ入力フィールドに名前を割り当てる必要があることを知らせます。

**すべての文字がデフォルトの 16 進数のフィールド識別子と一致する**

- フィールドの属性は、自動スキップ、英数字、および普通の輝度に設定されます。文字は連続している必要があり、次のフィールドと分けるには空白文字のみ使用します。
- このフィールドは、フィールド特性の保守による 16 進数初期値の割り当てを受け入れるものとして識別されます。
- フィールドの長さに連続した文字の数が設定されます。

**すべての文字が空白**

- フィールドの属性は、自動スキップ、英数字、および普通の輝度に設定されます。
- フィールドの長さは、1 に設定されます。

**その他**

- フィールドの属性は、自動スキップ、英数字、および普通の輝度に設定されます。
- 長さは、属性識別子と、次のフィールドの前にある最後の非空白文字との間の文字数です。

## ▼ 行を右または左に移動する

1. フォーマット画面で PF4 キーを押すと、図 9-10 の行の移動画面が表示されます。
2. 何文字分移動するかを指定します。
3. 次のいずれかの PF キーを押します。
   - PF7 キーを押すと、行が左に移動します。
   - PF8 キーを押すと、行が右に移動します。
4. PF3 キーを押して、フォーマット画面に戻ります。

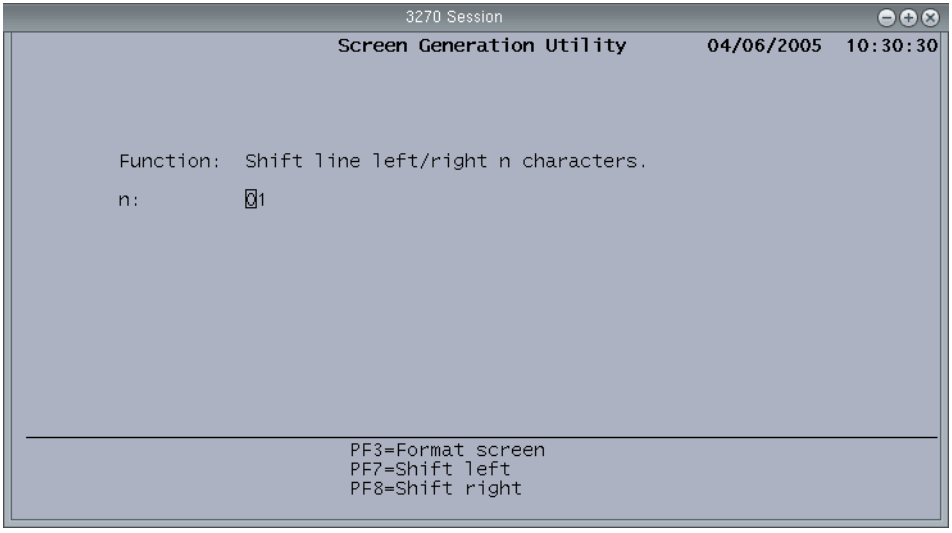


図 9-10　Format Screen—Shift Line 画面

行の移動画面では、次のファンクションキーが使用できます。

| ファンクションキー | 動作 |
|---|---|
| PF3 | データを移動しないでフォーマット画面に戻ります。 |
| PF7 | 行全体を *n* 文字分左に移動します。 |
| PF8 | 行全体を *n* 文字分右に移動します。 |

## ▼ 現在の行を移動またはコピーする

1. **フォーマット画面で PF6 キーを押すと、次の画面が表示されます。**

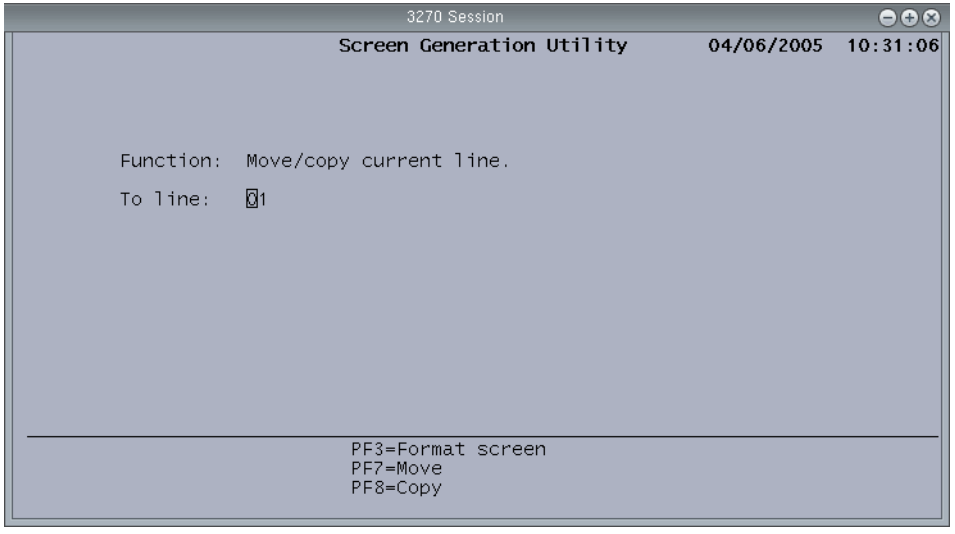


**図 9-11** Format Screen—Move/Copy Line 画面

2. **「To line」フィールドに、選択した行の移動先の行 (マップの最初の行からの相対値) を入力します。**

   この行は、必ずしもフォーマット画面に表示されません。移動/コピー画面が最初に表示されるとき、「To Line」フィールドは、移動する前の行の行番号を示しています。

3. **次のいずれかの PF キーを押します。**

   - PF7 キーを押すと、行が新しい位置に移動します。
   - PF8 キーを押すと、行を新しい位置にコピーし、元の行をそのまま残します。

4. PF3 キーを押して、フォーマット画面に戻ります。

この画面では、次のファンクションキーが使用できます。

| ファンクションキー | 動作 |
|---|---|
| PF3 | 操作を取り消して、フォーマット画面に戻ります。 |
| PF7 | 「To line」フィールドを指定したあと、現在の行を画面上の別の行に移動します。「To line」フィールドの行が空白の場合、現在の行で定義されているすべてのフィールドが「To line」フィールドの行にコピーされ、現在の行で定義されているフィールドは削除されます。「To line」フィールドの行が空白でない場合、次のメッセージが表示されます。<br>`KIX1457E Data exists on target line. Hit key again to overwrite.` |
| PF8 | 現在の行を、「To line」で指定した行にコピーします。現在の行フィールドが削除されないこと以外は、移動操作の処理と同様です。 |

## ▼ 文字を挿入または削除する

1. フォーマット画面で PF7 を押すと、次の画面が表示されます。

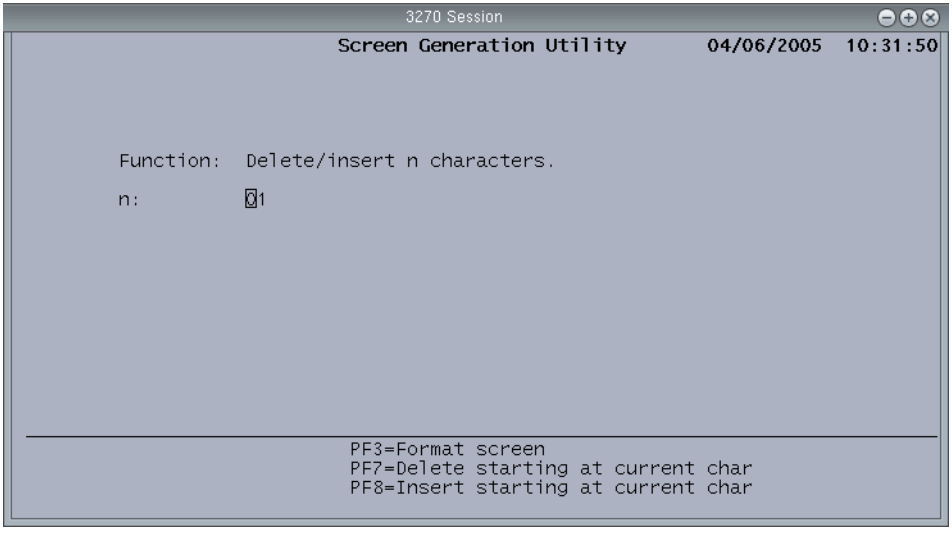


図 9-12　Format Screen—Delete/Insert Characters 画面

2. **フォーマット画面のカーソルが置かれている文字のあとで、文字を削除または挿入するときの文字数を入力します。**

3. **次のいずれかの PF キーを押します。**

   - PF7 キーを押すと、文字が削除されます。

     現在のカーソルが既存のフィールド内にあり、そこで文字を削除する場合、そのフィールド長が指定した文字数分のみ短くなります。現在のカーソル位置の右側にあるすべてのフィールドは、指定した文字数分のみ左に移動します。指定した文字数に次のフィールドの一部が含まれる場合、そのフィールドは削除されます。削除されるフィールドに関連するすべての初期値はその画面に残りますが、どのフィールドにも関連しなくなります。

   - PF8 キーを押すと、文字が挿入されます。

     フォーマット画面でカーソルが置かれている文字の前に、空白文字が挿入されます。現在のカーソルが既存のフィールド内にある場合、フィールドの長さは指定した文字数分のみ拡張します。現在のカーソル位置の右側にあるすべてのフィールドは、指定した文字数分のみ右に移動します。再配置されたフィールドが処理によって右余白を越える場合、そのフィールドはマップから削除されます。

4. **PF3 キーを押して、フォーマット画面に戻ります。**

   以前に変更した属性はすべて維持されます。詳細は、226 ページの「属性識別子」を参照してください。

## フィールド特性の変更

カーソルの位置は、画面フォーマット処理の際に重要です。カーソルが置かれているフィールドは、Enter キーを押したときにフィールドの特性画面で最初に表示されるフィールドになります。フォーマット画面に戻ると、フィールドの特性画面で最後に表示したフィールドの属性識別子にカーソルが置かれます。フォーマット画面とフィールドの特性画面を切り換えることにより、特定のフィールドを変更できます。

### ▼ フィールド特性を変更する

1. **フォーマット画面で Enter キーを押すと、図 9-13 のフィールドの特性画面が表示されます。**

   この画面のデータは、BMS `DFHMDF` マクロ命令で定義されるデータと同じです。

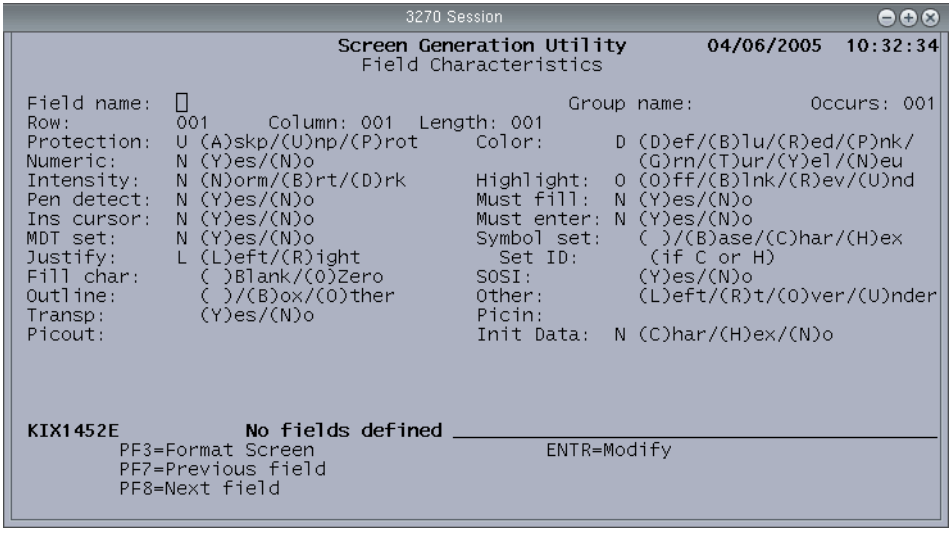


**図 9-13** Field Characteristics 画面

2. **データフィールドを変更します。**

   フィールドとその値のリストは、表 9-5 を参照してください。

3. **変更を終了したら、Enter キーを押してフィールドの変更を適用します。**

4. **PF3 キーを押して、フォーマット画面に戻ります。**

   フィールドの特性画面では、次のファンクションキーが使用できます。

| ファンクションキー | 動作 |
|---|---|
| PF3 | フォーマット画面に戻ります。 |
| PF7 | 前のフィールドを表示します。マップの最初のフィールドを表示している場合、次のメッセージが表示されます。<br>`Beginning of data reached` |
| PF8 | 次のフィールドを表示します。マップの最後のフィールドを表示している場合、次のメッセージが表示されます。<br>`End of data reached` |
| Enter | フィールド特性を変更し、データを検査します。データが有効な場合、その新しい情報が適用されて次のメッセージが表示されます。<br>`Data modified`<br>無効なデータがある場合、そのフィールドが強調表示され、カーソルが最初の無効なフィールドに置かれます。次のメッセージが表示されます。<br>`Invalid data` |

次の表は、フィールドの特性画面上のデータフィールドおよび適用できる値を示します。これらのフィールドは Enter キーを押すと有効になります。サポートされる拡張属性については、129 ページの「拡張属性」を参照してください。

**表 9-5** フィールドの特性画面のデータフィールド

| フィールド名 | 説明 |
|---|---|
| Field name | フィールドの一意の名前 (1 ～ 7 文字)。最初の文字はアルファベットです。 |
| Group name | グループの一意の名前 (1 ～ 7 文字)。最初の文字はアルファベットです。 |
| Occurs | フィールドが繰り返される回数 (1 ～ 999)。グループ名が指定されている場合、このフィールドの値は 1 である必要があります。 |
| Row | 属性識別子が定義されるマップ内の行の位置。表示専用フィールドです。 |
| Column | 属性識別子が定義されるマップ内の列の位置。表示専用フィールドです。 |
| Length | フィールドの長さ (1 ～ 256 文字)。指定した長さが現在のフィールドを越えて次のフィールドに渡らないかどうかが検査されます。 |
| Protection | フィールドの保護属性。<br>A: 自動スキップ<br>U: 無保護<br>P: 保護 |
| Color | 対応するマップオプションより優先されます。<br>D: デフォルト<br>B: 青<br>R: 赤<br>P: ピンク<br>G: 緑<br>T: 青緑<br>Y: 黄<br>N: 無色 |
| Numeric | このフィールドを数字のデータのみに限定するかどうかを指定します。<br>Y: 数字のデータのみ<br>N: 英数字のデータ |
| Intensity | 端末で表示されるフィールドの輝度。<br>N: 普通の輝度でフィールドを表示。<br>B: 高い輝度でフィールドを表示 (明るい)。<br>D: フィールドを表示しません (暗い)。 |

表 9-5　フィールドの特性画面のデータフィールド (続き)

| フィールド名 | 説明 |
|---|---|
| Highlight | マップ内で定義されたマップおよびフィールドに割り当てるデフォルトの強調表示属性に対するマップオプションより優先されます。<br>O: オフ<br>B: 点滅<br>R: 反転表示<br>U: 下線 |
| Pen detect | このフィールドをライトペンで検出できるようにするかどうかを指定します。このオプションはサポートされていません。<br>Y: このフィールドをライトペンで検出できるようにします。<br>N: このフィールドをライトペンで検出できるようにしません。 |
| Must fill | マップ内のマップおよびフィールドに設定されるデフォルト値に対するマップオプションより優先されます。<br>Y: このオプションが設定されているデータフィールドには、データを完全に入力する必要があります。このフィールドにデータを完全に入力しないでカーソルを移動すると、入力禁止状態になります。<br>N: Must Fill チェックを実行しません。 |
| Ins cursor | このフィールドにカーソルを置くかどうかを指定します。1 つのマップに対して複数の「Ins cursor」フィールドが存在し、このオプションが選択されている場合、最後のフィールドにカーソルが置かれます。<br>Y: このフィールドの最初の位置にカーソルを置きます。フィールドの最初の位置は、フィールド属性のすぐあとに続きます。<br>N: フィールドにカーソルを置きません。 |
| Must enter | マップ内のマップおよびフィールドに設定されるデフォルト値に対するマップオプションより優先されます。<br>Y: このフラグが設定されている場合、フィールドへのデータ入力が必要です。このフィールドにデータを入力しないでカーソルを移動すると、入力禁止状態になります。<br>N: このフィールドにはデータを入力する必要はありません。 |
| MDT set | 変更データタグ (MDT) セットでこのフィールドを送信するかどうかを指定します。<br>Y: MDT を設定し、端末の後続の読み取りによってフィールドのデータが返されるようにします。<br>N: MDT を設定しません。 |
| Symbol set | プログラム式記号を使用するかどうかを示すマップオプションより優先されます。<br>B: 基本記号セットを使用<br>C: プログラム式記号セットを使用。プログラム式記号識別子を 1 文字で指定します。<br>H: プログラム式記号セットを使用。プログラム式記号識別子を 2 桁の 16 進数で指定します。 |
| Justify | このフィールドに入力されたデータに対して実行される位置揃えのタイプ。<br>L: 記号記述マップ内のデータを左揃えします。<br>R: 記号記述マップ内のデータを右揃えします。 |

表 9-5　フィールドの特性画面のデータフィールド (続き)

| フィールド名 | 説明 |
|---|---|
| Set ID | 対応するマップオプションより優先されます。記号セットが (C)har または (H)ex の場合のみ使用します。 |
| Fill char | 戻されるデータの長さがフィールドの長さよりも短い場合に、位置決め後にフィールドの残りの記号記述データ領域に配置する文字。<br>空白: 空白を配置<br>0: 0 を配置 |
| SOSI | 対応するマップオプションより優先されます。フィールドに EBCDIC および DBCS データの両方を含めることができます。DBCS データは、データの最初のシフトアウト (SO) 文字と、データの最後のシフトイン (SI) 文字によって指定されます。<br>Y: フィールドに EBCDIC および DBCS のデータが含まれます。<br>N: フィールドには EBCDIC データのみが含まれます。 |
| Outline | 対応するマップオプションより優先されます。フィールドの周りを線で囲むか、フィールドの上、下、左、右のどれかに線を引きます。<br>B: フィールドの周りを線で囲みます。<br>O: 「Other」フィールドで枠を定義します。 |
| Other (その他) | 「Outline」オプションが O のときだけ有効です。<br>L: フィールドの左側に線を引きます。<br>R: フィールドの右側に線を引きます。<br>O: フィールドの上に線を引きます。<br>U: フィールドの下に線を引きます。 |
| Transp | フィールドの背景が透明か不透明かを指定するマップオプションより優先されます。<br>Y: 透明のフィールド。<br>N: 不透明のフィールド。 |
| Picin | 記号記述マップを定義するときの、フィールドの COBOL または PL/I の PICTURE 節 (1 ～ 20 文字)。<br>C では無効です。 |
| Picout | 記号記述マップを定義するときの、フィールドの COBOL または PL/I の PICTURE 節 (1 ～ 20 文字)。COBOL か PL/I かは、マップセットの特性画面の「Language」フィールドの値によって異なります。1 つのマップに、合計で最大 100 の Picin および Picout 節の組み合わせを定義できます。<br>C では無効です。 |
| Init Data | フィールドの初期データを出力マッピング操作で送信するかどうかを指定します。<br>C: 初期データを定義し、文字データを使用します。<br>H: 初期データを定義し、16 進数データを使用します。<br>N: フィールドに初期データを定義しません。 |

表 9-5　フィールドの特性画面のデータフィールド (続き)

| フィールド名 | 説明 |
|---|---|
| Initial value | 出力マッピング操作で送信する初期データ。「Init Data」フィールドで **C** または **H** と入力したあと、SGU が「Initial value」フィールドの最初にカーソルを置きます。このフィールドに入力されたデータは、「Init Data」フィールドで指定されたコードに基づいて検査されます。「Init Data」のコードによる違いは次のとおりです。<br>C: データは英数字で、フィールド長を超えることはできません。<br>H: データは有効な 16 進数で、フィールド長の 2 倍を超えることはできません。指定する文字の数は、偶数である必要があります。<br>N: 初期値はありません。 |

# マップの削除

## ▼ マップを削除する

1. **マップの一覧画面で PF5 キーを押して、削除するマップをマークします。**

   図 9-14 に示す画面のように、マークが付けられたマップは太字で強調表示されます。

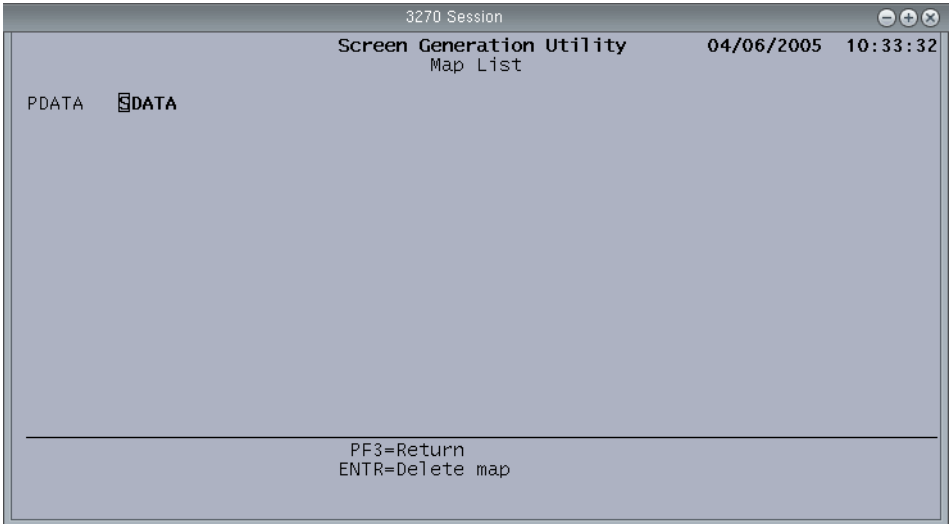


**図 9-14**　Map List のメンテナンス - Delete Map 画面

2. **Enter キーを押して、マップを削除します。**

   確認ウィンドウは表示されません。

# BMS マクロ命令の作成

.sgu ファイルを作成したあと、マクロ命令ファイル (.bms ファイル) を作成する必要があります。このファイルは、Sun MTP BMS アセンブラでアセンブルされるか、アセンブル用に IBM CICS システムにエクスポートされます。アセンブルされたファイルにはマクロが含まれます。このマクロは、プログラムの実行時に使用される物理マップおよびアプリケーションプログラムにコピーされる記号マップを作成します。

## ▼ .bms ファイルを作成する

1. **マップ管理メニュー画面で .sgu ファイルを選択し、PF12 キーを押します。**

2. **図 9-15 の画面が表示されたら、必要に応じて値を変更します。**

   SGU マップセットの「Filename」および「Environment」フィールドでは、マップ管理メニュー画面で選択した .sgu ファイルをデフォルトとします。BMS マクロの「Filename」および「Environment」フィールドでは、.sgu ファイルのファイル名およびディレクトリをデフォルトとします。

   BMS ファイルを生成するとき、「Quote/Apost」フィールドで設定された引用文字を使用して文字列を囲みます。このフィールドの「QUOTE」を使用すると、文字列は二重引用符 (") で囲まれます。このフィールドの「APOST」を使用すると、文字列はアポストロフィ (') で囲まれます。

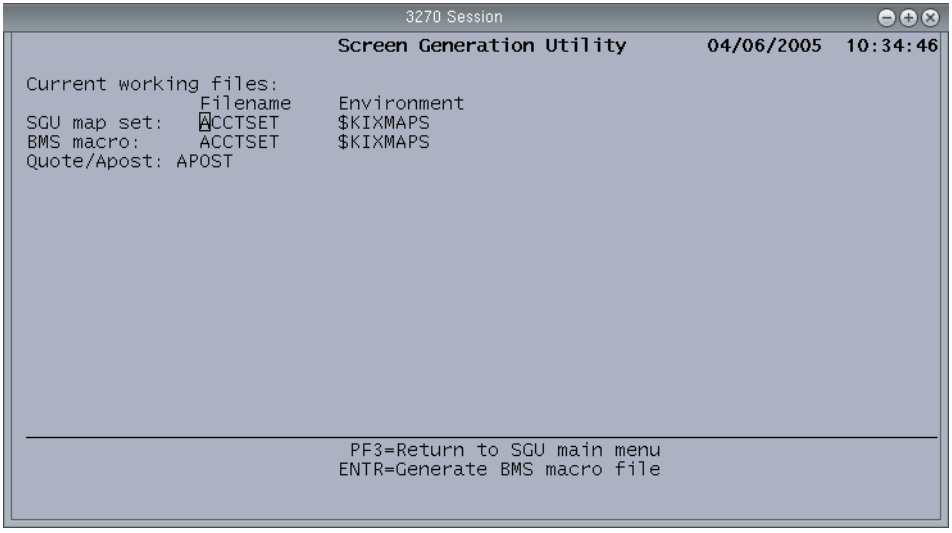


**図 9-15**　BMS マクロの生成画面

3. **Enter キーを押して、BMS マクロ命令を生成します。**

   ファイルは、領域を開始したユーザーによって許可される権限を使って作成されます。

4. **PF3 キーを押して、マップ管理メニュー画面に戻ります。**

# BMS ファイルの表示

SGU を開始すると、図 9-2 のようなマップ管理メニュー画面が表示されます。

## ▼ BMS 管理メニューを表示する

- **マップ管理メニュー画面で PF5 キーを押します。**

  拡張子 .bms のファイルを一覧表示する BMS 管理メニューが表示されます。

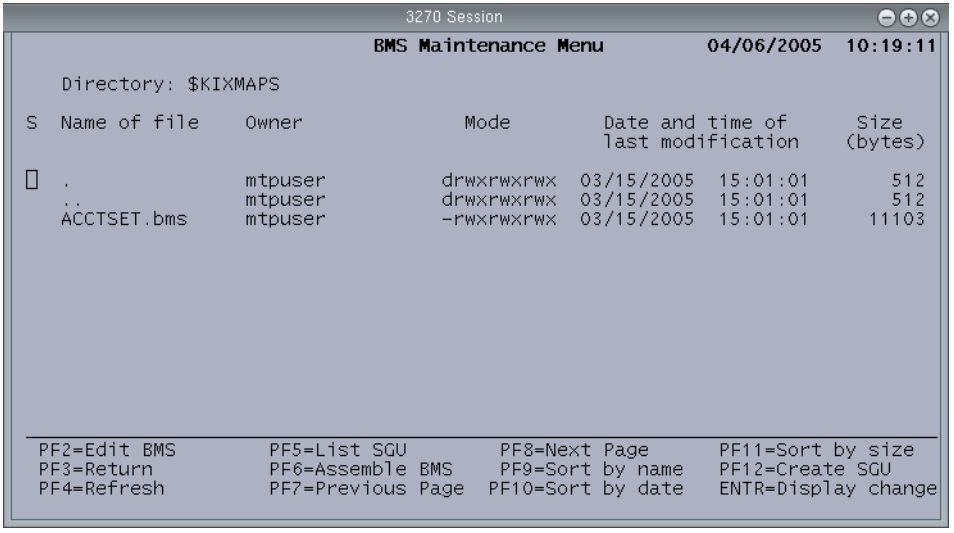


図 9-16　BMS Maintenance Menu

## ▼ 異なるディレクトリのファイルを表示する

1. 「Directory」フィールドに、表示する .bms ファイルがあるディレクトリの名前を入力します (最大 50 文字)。

   ディレクトリの最初の要素を環境変数にすることもでき、スラッシュで区切って複数のサブディレクトリをあとに続けることができます。

2. Enter キーを押して、変更を適用します。

   ファイルリストが変更され、選択したディレクトリの .bms ファイルが表示されます。

   BMS 管理メニューでは、次のキーが使用できます。

| ファンクションキー | 動作 |
|---|---|
| PF3 | BMS 管理メニューを閉じて、開発システムメインメニューまたはブランクのトランザクション画面に戻ります。 |
| PF2 | 選択したファイルを作業ファイルとして kixed シェルスクリプトを実行します。ファイルを選択しないと、カーソルが置かれているファイルが使用されます。デフォルトでは、vi エディタが呼び出されます。 |
| PF4 | 画面のファイルリスト部分のデータを更新します。ファイルの変更後、このキーを押して変更後の内容を表示します。 |

| ファンクションキー | 動作 |
|---|---|
| PF5 | マップ管理メニュー画面を表示します。マップ管理メニュー画面については、202 ページの「マップ管理メニュー画面」を参照してください。 |
| PF6 | 選択したファイルを物理マップにアセンブルします。物理マップには拡張子 .map が付きます。詳細は、242 ページの「BMS マップのアセンブル」を参照してください。 |
| PF7 | ファイルの前のページを表示します。最初のページを表示している場合、次のメッセージが表示されます。<br>Beginning of data reached |
| PF8 | ファイルの次のページを表示します。最後のページを表示している場合、次のメッセージが表示されます。<br>End of data reached |
| PF9 | ディレクトリの内容をファイル名順にソートして再表示します。 |
| PF10 | ディレクトリの内容をファイルの更新日順にソートして再表示します。 |
| PF11 | ディレクトリの内容をファイルサイズ順にソートして再表示します。 |
| PF12 | .bms ファイルから .sgu ファイルを作成します。詳細は、247 ページの「.sgu マップセットファイルの作成」を参照してください。 |
| Enter | 「Directory」フィールドに対するすべての変更を有効にします。ファイル表示域が一度消去され、新しいディレクトリ内のファイルが表示されます。 |

# BMS マップのアセンブル

プログラムで .bms ファイルを使用するには、ファイルをアセンブルする必要があります。アセンブルされたファイルには拡張子 .map が付きます。Sun MTP では、次の 3 つのアセンブル方法が実行可能です。

- SGU
- kixbms ユーティリティー
- Compilation メニュー

## ▼ SGU を使用してマップをアセンブルする

1. マップ管理メニュー画面で PF5 キーを押し、BMS 管理メニューを開きます。

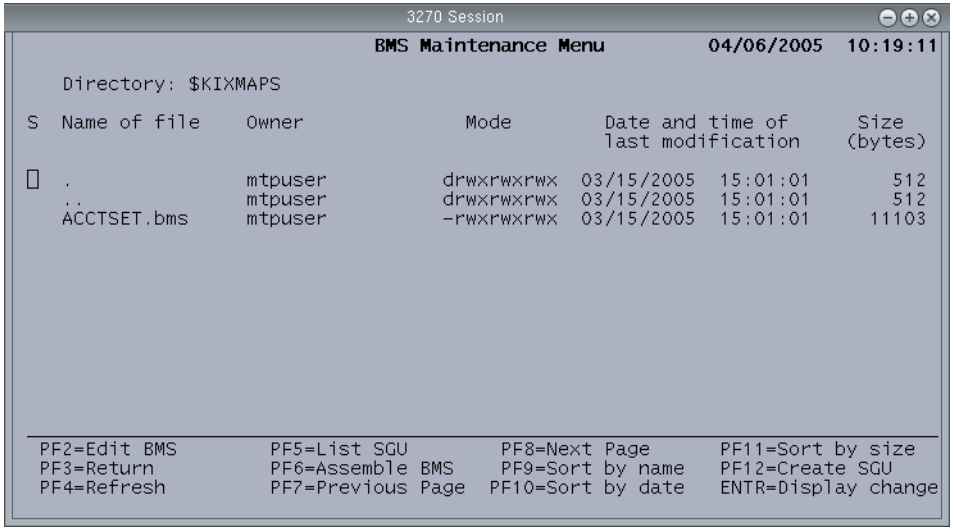


図 9-17　BMS Maintenance メニュー

2. .bms ファイルを選択して、PF6 キーを押します。
3. 図 9-18 の画面が表示されたら、次のように実行します。
   a. 必要に応じて、「Output filename」および「Output Environment」フィールドを変更します。

      これらのフィールドでは、BMS 管理メニューの値がデフォルトとして使用されます。

b. アセンブラで COBOL コピーブック、C ヘッダーファイル、または PL/I インクルードファイルを生成するには、このオプションで `Y` を選択します。

`.bms` ファイルの `LANG` の値に基づいて適切なメンバーが作成されます。

c. アセンブラでリストファイルを生成するには、このオプションで `Y` を選択します。

d. アセンブルする前に、Enter キーを押して変更を適用します。

ファイルをアセンブルしない場合、PF3 キーを押して BMS 管理メニューに戻ります。

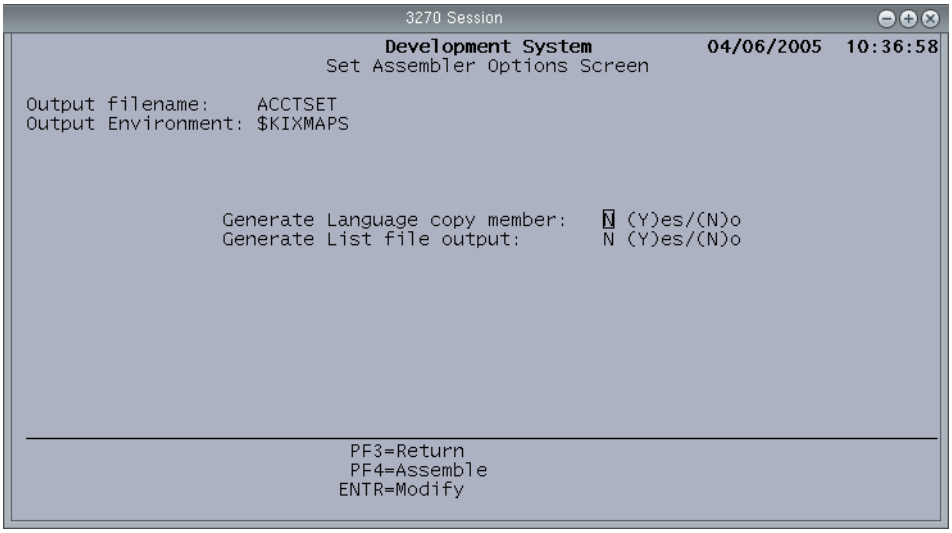


**図 9-18** Set Assembler Options Screen

4. PF4 キーを押して、`.map` ファイルを生成します。

アセンブルの結果を示す画面が表示されます。リストファイルを生成した場合は、それも表示されます。

ファイルは、領域を開始したユーザーによって許可される権限を使って作成されます。

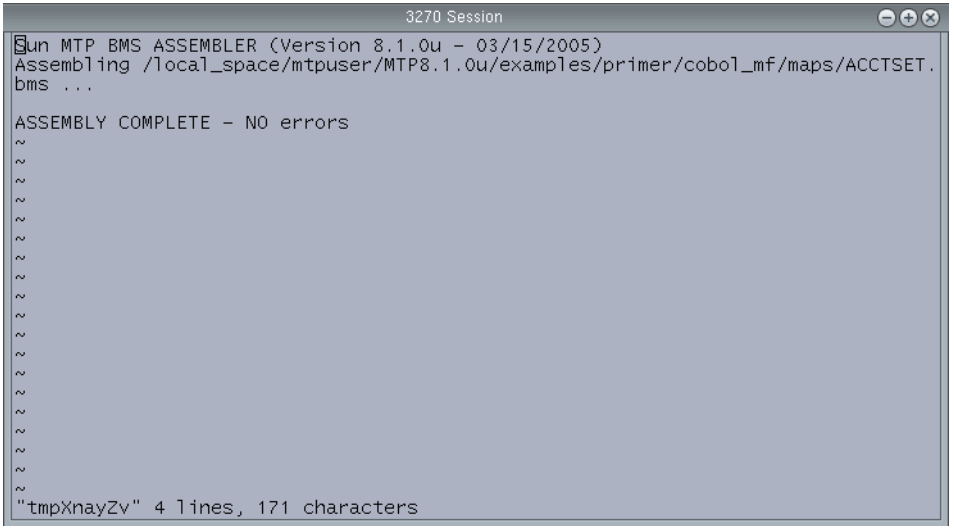


**図 9-19**　アセンブルの結果

5. **この一時リストファイルを保存するには、次のように入力します。**

```
:wq ファイル名
```

ファイルは、「Output Environment」フィールドで指定されたディレクトリに保存されます。

ファイルを保存しない場合は、`:q` と入力して画面を閉じると、BMS 管理メニューに戻ります。

## ▼ kixbms を使用してマップをアセンブルする

- **`.bms` ファイルを入力として使用して、適切なオプションとともに `kixbms` コマンドを入力します。**

  すべてのオプションについては、『Sun Mainframe Transaction Processing 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。

  **例**: COBOL マップをアセンブルして、COBOL コピーブックを生成します。`.bms` ファイルで `LANG=COBOL` と設定されている場合、コピーブックが作成されます。出力ファイルには自動的に `SB001.map` という名前が付けられます。

```
$ kixbms -c  SB001.bms
```

**例**: JB045A という名前の C マップをアセンブルし、C ヘッダーファイルを生成します。.bms ファイルで LANG=C と設定されている場合、ヘッダーファイルが作成されます。.map ファイルに異なる名前を使用する場合、-o オプションおよびマップ名を指定する必要があります。

```
$ kixbms -c -o JB045A.map JB045.bms
```

## ▼ Compilation メニューを使用してマップをアセンブルする

1. 開発システムのメインメニューで、PF4 キーを押して、Compilation メニューを表示します。
2. Directory および Extensions フィールドを編集し、アセンブルする BMS ファイルを含むディレクトリを選択します。
3. Enter キーを押して、ディレクトリの内容を表示します。
4. 拡張子 .bms のファイルを選択します。

   複数ファイルの選択が可能です。
5. PF6 キーを押して「Set Assembler Options」画面を表示します。

   「Set Assembler Options」画面では、BMS アセンブラの処理をカスタマイズできます。

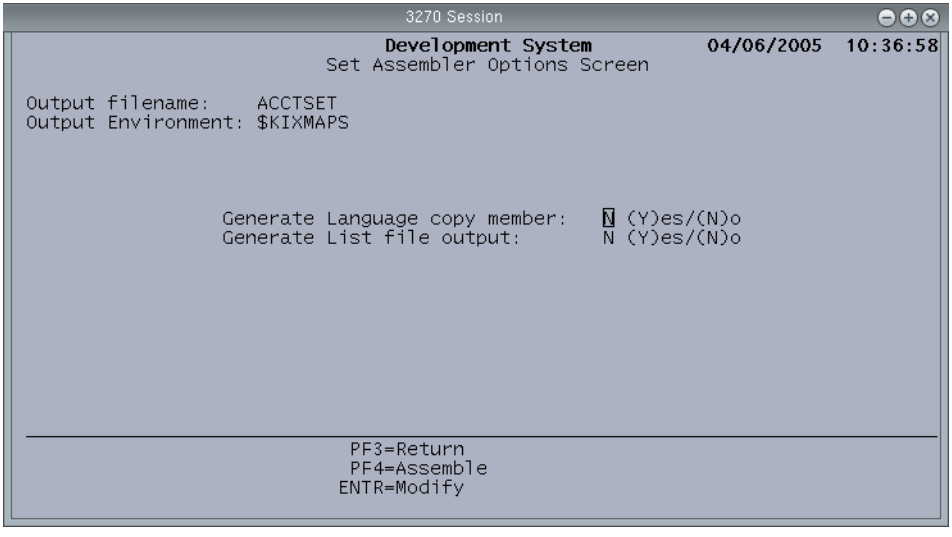


図 9-20 Set Assembler Options Screen

6. 画面の次の値を編集します。

| フィールド | 可能性のある値 |
|---|---|
| Output filename | 現在選択しているファイルの名前がデフォルトとして表示されます。このファイル名は変更できます。ファイル名は 最大 14 文字です。 |
| Output Environment | コンパイルメニューで入力したディレクトリ名がデフォルトとして表示されます。出力環境はどのディレクトリでも可能で、ディレクトリを指定する要素がドル記号 ($) ではじまる環境変数も設定できます。環境変数の前または後ろをスラッシュ (/) で区切って、サブディレクトリを指定できます。 |
| Generate Language copy member | COBOL コピーブック、PL/I インクルードファイル、または C ヘッダーファイルを作成します。作成されるコピーメンバーのタイプは、`.bms` ファイルの `LANG` の値に応じて異なります。<br>`Y`: コピーメンバーを作成します。<br>`N`: コピーメンバーを作成しません。 |
| Generate List file output | アセンブルの結果を示すリストファイルを作成します。<br>`Y`: `vi` エディタで一時ファイルを作成し、それを画面に表示します。<br>• ファイルを保存するには、`:wq` *filename* と入力します。<br>• ファイルの保存時に別のディレクトリを指定した場合を除き、Output Environment で指定されているディレクトリにファイルが保存されます。<br>`N`: リストファイルを作成しません。 |

7. PF4 キーを押して、マップをアセンブルします。

これにより、BMS アセンブラ `kixbms` を呼び出す `kixasm` シェルスクリプトが実行されます。複数のファイルを選択すると、ファイルを選択した順番と逆の順番で、最後に選択したファイルが最初にアセンブルされます。

各ファイルのアセンブルが完了するたびに、アセンブルの結果がデフォルトのエディタ (`vi`) に表示されます。

# .sgu マップセットファイルの作成

.bms ファイルから .sgu ファイルを作成できます。これは、IBM CICS システムからインポートされた BMS ソースコードを管理するのに便利です。.sgu ファイルを作成する方法は、次の 2 つがあります。

- SGU を使用
- -s オプションで kixbms ユーティリティーを使用

## ▼ SGU を使用して .sgu ファイルを作成する

1. BMS 管理メニューでファイルを選択し、PF12 キーを押します。
2. 図 9-21 の画面が表示されたら、必要に応じてフィールドを変更します。

   a. 「BMS macro」フィールドに入力ファイル名が表示されます。

   BMS 管理メニュー画面で選択したファイルがデフォルトになります。このファイルには BMS マクロ命令が含まれます。

   b. 作成されるファイルは「SGU map set」フィールドに表示されます。このとき、処理対象に選択した BMS マクロファイルの名前がデフォルトとして使用されます。

   c. 「Environment」フィールドは、BMS 管理メニューで指定されたディレクトリがデフォルトになります。

   d. .sgu ファイルを生成するとき、「Quote/Apost」フィールドで設定した引用文字を使用して文字列を囲みます。

   - そのフィールドで QUOTE と入力すると、文字列は二重引用符 (") で囲まれます。
   - そのフィールドで APOST と入力すると、文字列はアポストロフィ (') で囲まれます。

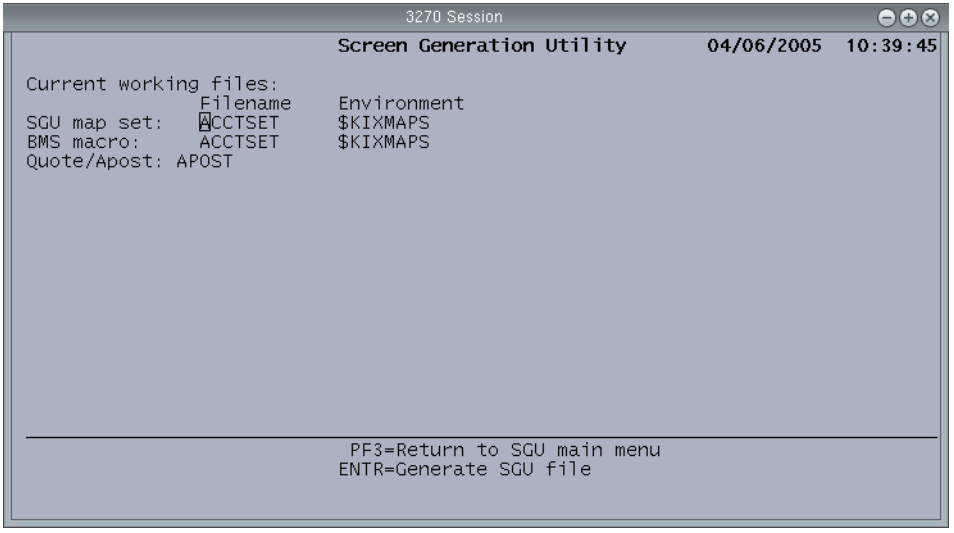


図 9-21　SGU の生成画面

3. **Enter キーを押して、.sgu ファイルを作成します。**

   Sun MTP が、アセンブルの結果を示す画面を表示します。

   ファイルを作成しない場合、PF3 キーを押して BMS 管理メニューに戻ります。

   ファイルは、領域を開始したユーザーによって許可される権限を使って作成されます。権限は、ユーザーの umask によって決定されます。

4. **:q と入力して画面を閉じると、BMS 管理メニューに戻ります。**

## ▼ kixbms を使用して .sgu ファイルを作成する

- **kixbms ユーティリティーを -s オプションとともに使用して、.bms ファイルから .sgu ファイルを作成します。**

  **例**: TB012.bms ファイルから .sgu ファイルを生成します。出力ファイルには自動的に TB012.sgu という名前が付けられます。

```
$ kixbms -s TB012.bms
```

  **例**: TB012.bms ファイルから、TB012A という名前の .sgu ファイルを生成します。出力ファイル名を変更する場合は、-o オプションを指定する必要があります。

```
$ kixbms -s -o TB012A.sgu TB012.bms
```

# 第10章

# SQL インタフェースを使用したRDBMS へのアクセス

この章では、リレーショナルデータベース管理システム (RDBMS) へのアクセスを管理する SQL インタフェースに対する Sun MTP のサポートの状況について説明します。Sun MTP は、これらの RDBMS をサポートします。

- Oracle®
- IBM DB2 Universal Database (UDB)
- Sybase

**注 –** PL/I プログラムをサポートしているのは、Oracle だけです。

一般的なアプリケーショントランザクションには、データベースアクセスのための RDBMS スタイルのプログラマティックインタフェースと組み合わせて、通信、プログラムフロー、およびエグゼクティブサービスを実行するために、CICS スタイルのコマンドレベルインタフェースで記述された CICS アプリケーションプログラムが 1 つ以上含まれています。

この章の内容は、次のとおりです。

# アプリケーション設計のテクニック

Sun MTP RDBMS 環境のアプリケーション設計では、2 つのプログラミングテクニックを組み合わせます。

- プログラムは CICS API を使用して、メッセージの送受信のために Sun MTP 実行環境と会話する必要があります。また、命令および EXEC インタフェースブロック (EIB) を使用して情報を取得および設定し、`LINK` および `XCTL` 命令を使用してトランザクションフローに影響を与えます。
- データベースにアクセスするには、プログラムで RDBMS 独自の SQL 文を使用する必要があります。SQL 文は、組み込み SQL または呼び出しインタフェースと呼ばれる手法を使用してプログラムに配置されます。たとえば、次のように設定します。

```
EXEC SQL
SELECT * FROM TABLE1, TABLE2
END-EXEC.
```

RDBMS によってプリプロセスされる必要のある手続き型 SQL 文は、RDBMS 実行時システムへの直接呼び出しを実行します。すなわち、SQL 呼び出しは Sun MTP によってインターセプトされません。

プログラム定義で “value” 文がデータ名に明示的に割り当てられていないかぎり、CICS プログラムの作業記憶域は 2 進ゼロに初期化されます。開発者は、プログラム内の各データ構造 (RDBMS とユーザープログラム間の通信に使用される `SQLCA` 構造など) に対して、このことを考慮する必要があります。`SQLCA` 構造は、次の文を使用してプログラムの COBOL データ部の適切な位置に組み込まれます。

```
EXEC SQL INCLUDE SQLCA END-EXEC.
```

組み込み SQL プリプロセッサは、`EXEC` 文を COBOL 構造に展開します。詳細は、256 ページの「COBOL プログラムのコンパイル」を参照してください。

# データベースの保全性の維持

**注 –** アプリケーションがアクセスするすべてのファイルは、データベースと呼ばれます。

データベースの保全性を維持するうえで、次の 2 つの操作が重要です。

| | |
|---|---|
| コミット | すべてのデータベース更新を適用します。 |
| ロールバック | データベースの初期状態に戻します。 |

CICS トランザクションが正常に終了すると、すべての更新はアプリケーションデータベースに書き込まれます。トランザクションが異常終了した場合、データベースはトランザクション開始時の状態のままになります。

コミットおよびロールバック操作は、暗黙的または明示的に行われます。トランザクションが正常に終了した場合、Sun MTP はすべてのデータベース操作を暗黙的にコミットします。また、トランザクションが異常終了した場合、データベースをロールバックします。さらに、アプリケーションプログラムでは、`SYNCPOINT` を実行して明示的なコミットを要求したり、`SYNCPOINT ROLLBACK` を実行して明示的なロールバックを要求したりできます。

**XA 以外の環境**:

SQL では、`COMMIT WORK` 機能を提供します。この命令の実行では、コードが対象とする特定の RDBMS の更新のみをコミットします。すなわち、Oracle `COMMIT WORK` では、Informix または Sybase データベースのコミットを行いません。`COMMIT WORK` は各 RDBMS に対して固有なので、複数の RDBMS を使用するアプリケーション内または 1 つの RDBMS と複数の VSAM ファイルを使用するアプリケーション内では使用しないでください。

**XA 環境**:

Sun MTPが、XA 環境用に設定される場合は、コミットまたはロールバックの呼び出しが、XA を禁止するように定義された RDBMS に変更されます。コミットまたはロールバック変更の要求は、`EXEC CICS SYNCPOINT` または `EXEC CICS ROLLBACK` 文を使用して、行います。

# 暗黙的操作の実装

トランザクションの実行中には、特定の暗黙的データベース操作を前提にできます。これらの暗黙的操作は、通常 Sun MTP トランザクションサーバープロセス (`unikixtran`) およびバッチサーバープロセス (`unikixvsam`) のユーザー出口モジュール内に実装されます。

**XA 以外の環境**:

ユーザー出口モジュールは、以下の 6 つの所定位置で Sun MTP ソフトウェアからの出口を提供します。

- トランザクションおよびバッチプロセッサの初期化 (通常データベースのログインに使用)
- トランザクションおよびバッチプロセッサの終端 (通常データベースのログオフに使用)
- トランザクションの開始 (通常そのトランザクションのデフォルトデータベースの設定またはセキュリティーの設定に使用)
- トランザクションの終了
- 暗黙的なコミット。ここでは、SQL の `COMMIT WORK` 文が適切
- 暗黙的なロールバック。ここでは、SQL の `ROLLBACK WORK` 文が適切

管理者およびアプリケーション設計者は、トランザクション内のデータベースの数や名前についての情報を共有している必要があります。これを行わないと、アプリケーションがデータベースを更新し、コミット時にデータベースのコミットが行われない場合に問題が発生します。

RDBMS のユーザー出口モジュールの実装については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

**XA 環境**:

Sun MTP XA 環境では、Sun MTP Transaction Manager (TM) が暗黙的に設定されたリソースマネージャーの コミットおよびロールバックを管理します。詳細は、『Sun Mainframe Transaction Processing ソフトウェア XA リソースマネージャーの使用』を参照してください。

## 1 つのトランザクション内での複数の RDBMS へのアクセス

Sun MTP 製品では、1 つのトランザクションによる複数の RDBMS へのアクセスは完全にサポートされています。複数の RDBMS をサポートするには、各 RDBMS およびオンライントランザクション処理 (OLTP) 実行環境が共通の 2 フェーズコミットプロトコルに準拠している必要があります。Sun MTP は、XA 互換の RDBMS を管理する XA プロトコルを使用します。詳細は、『Sun Mainframe Transaction Processing ソフトウェア XA リソースマネージャーの使用』を参照してください。

# RDBMS のセキュリティー管理

RDBMS は、オブジェクトまたはリソースに割り当てられたいくつかの形式のアクセス権によってセキュリティーを扱います。たとえば、`CREATE TABLE` などの命令をユーザー ID レベルで管理できます。

RDBMS クライアントコードは、2 つの Sun MTP 実行可能プログラム (トランザクションとバッチサーバー) と結合されます。領域は、Sun MTP 領域を開始したユーザーの UNIX ユーザー ID で実行されます。アプリケーションプログラムはトランザクションまたはバッチサーバープロセス内で実行されるため、それらはそのプロセスのユーザー ID で実行されます。

Sun MTP の `kxtctinfo` 機能により、ユーザー出口モジュールのトランザクションレベルまたは端末レベルでセキュリティーを強化できます。この機能は、その端末ユーザーに関する情報を返します。この機能は RDBMS で提供されるセキュリティー機能と組み合わされ、管理者がセキュリティーを実装できるようにします。`kxtctinfo` 機能の詳細は、『Sun Mainframe Transaction Processing ソフトウェア管理者ガイド』を参照してください。

セキュリティーを実装する方法は、各 RDBMS によって異なります。次の節では、各 RDBMS のセキュリティー実装の例を示します。

## Oracle のセキュリティーの注意点

Oracle では、データベース操作時にデータベースオブジェクトへの特権を設定できる `ROLE` 機能をサポートします。

**例**: 次の表のように、各テーブルおよびユーザーのアクセス権を設定します。

| テーブル/ユーザー | USRA | USRB | USRC |
|---|---|---|---|
| TABLEA | ALL | NONE | SELECT |
| TABLEB | ALL | SELECT | NONE |

Oracle のデータ制御文は、次のとおりです。

```
CREATE ROLE USRA;
CREATE ROLE USRB;
CREATE ROLE USRC;
GRANT ALL ON TABLEA TO USRA;
GRANT SELECT ON TABLEA TO USRC;
GRANT ALL ON TABLEB TO USRA;
GRANT SELECT ON TABLEB TO USRB;
```

このシナリオでは、Sun MTP 領域所有者のユーザー ID は、トランザクション/バッチサーバーの初期化ユーザー出口を使用して、Oracle サーバーへ初期接続するために使用されます。ユーザー ID は SIT 内のアプリケーション名をとる場合 (`kxsysinfo` 機能を使用して決定される) や、その他のアプリケーション独自のメソッドにより渡される場合があります。

ユーザー ID の USRA、USRB、および USRC は、それらのトランザクション特権とともに SNT で定義されます。トランザクション初期化ユーザー出口の処理の間、トランザクションイニシエータのユーザー ID が、トランザクションユーザーの `ROLE` を設定するために使用されます。そのユーザー ID は `kxtctinfo` 機能を使用して決定されます。これにより、データベースオブジェクトへのアクセス時にトランザクションユーザーの特権が限定されます。

USRB がトランザクションイニシエータであり、トランザクションコードが TABLEA にアクセスする場合、プログラム実行時に `SQLERROR` が返されます。

同様に、データベースでの特権を持たない USR*n* が存在する場合、データベースに `ROLE` が定義されていないので、トランザクションの初期化に失敗します。

# DB2 UDB のセキュリティーの注意点

DB2 UDB は、各データベースのシステムテーブルのセットにユーザー特権を含めます。トランザクションの初期化時に、データベースクエリを実行してアクセス特権を確認してから、実行を継続します。

# Sybase のセキュリティーの注意点

Sybase の SETUSER 機能は、Oracle の ROLE 機能と同じです。

アクセス特権メソッドについても、Oracle と同じです。

**例**: 次の表のように、各テーブルおよびユーザーのアクセス権を設定します。

| テーブル/ユーザー | USRA | USRB | USRC |
|---|---|---|---|
| TABLEA | ALL | NONE | SELECT |
| TABLEB | ALL | SELECT | NONE |

Sybase のデータ制御文は、次のとおりです。

```
USRA
sp_adduser USRB
sp_adduser USRC
GRANT ALL ON TABLEA TO USRA
GRANT ALL ON TABLEB TO USRA
GRANT SELECT ON TABLEA TO USRC
GRANT SELECT ON TABLEB TO USRB
go
```

**注 –** sp_adduser は、データベースにユーザーを追加するための、Sybase のパッケージ化されたプロシージャーです。

SNT および SIT の情報は、Oracle の例と同じです。データベース所有者のユーザー ID は、データベースに接続するためにトランザクションサーバーの初期設定ユーザー出口内で使用されます。

トランザクションの開始段階では、kxtctinfo を使用してユーザー ID を判定し、さらに Sybase にユーザー ID を通知するために SETUSER SQL 文を発行します。これにより、実行時にデータベースオブジェクトの特権を限定します。

イニシエータが USRB であり、トランザクションが TABLEA にアクセスした場合、TABLEA には USRB に対する権限が設定されていないので、SQLERROR がアプリケーションに返されます。

同様に、USR*n* がトランザクションを初期化しようとすると、初期化に失敗します。これは、SNT で USR*n* がトランザクションを使用する権限を持つように定義されていても、データベースユーザーの USR*n* がないためです。

# SQL カーソルの管理

SQLCA 通信ブロックは、RDBMS 実行時システムとアプリケーションプログラム間の通信に使用されます。これによって制御されるリソースのひとつとして、SQL カーソルの管理があります。

通常、SQL カーソルは、同じアプリケーションプログラム内でオープン、処理、およびクローズされます。このプログラムがシングルソースのプログラムの場合もあり、SQLCA 構造が同じプログラム内の複数のモジュールに対してグローバルに可視である場合もあります。いずれの場合も、処理は完全に 1 つの実行可能プログラム内で処理されます。

# 複数の実行可能プログラムの使用法

CICS では、複数の実行可能プログラムをモジュールとして処理するための機能を提供します。これらのプログラムは、CICS コマンドレベル API の LINK および XCTL 機能を使用して 1 つのトランザクションに結合されます。XCTL および LINK 機能は組み合わせることができます。

| | |
|---|---|
| XCTL | プログラムが、COMMAREA と呼ばれる共通域を渡して 2 番目のプログラムに制御を転送します。最初のプログラムには制御が戻りません。 |
| LINK | プログラムが、共通域を渡して 2 番目のプログラムに制御を転送します。2 番目のプログラムが終了すると、呼び出しプログラムに制御が戻されます。 |

通常、すべての RDBMS は複数のプログラム上でカーソル操作をサポートします。これは、カーソル管理が SQLCA によってではなく、実行時システム内で行われるためです。しかし、アプリケーションが直前の SQL 文の実行による情報を XCTL または LINK コマンドによりアクセスされているプログラムの SQLCA に置いておく必要がある場合、RDBMS はこのサービスを提供しません。アプリケーション設計者は、プログラム論理内でこの状況を処理する必要があります。

# COBOL プログラムのコンパイル

COBOL アプリケーションに RDBMS 独自の SQL 文、CICS コマンドレベル API 文、および標準 COBOL 文がある場合、実行に適した形式にアプリケーションを変換する必要があります。

## ▼ COBOL プログラムをコンパイルする

1. RDBMS プリプロセッサを通じてソースコードを渡します。このプリプロセッサは、組み込み SQL 文を RDBMS 実行時システムに対する COBOL 呼び出しに変換します。
2. 変換されたプログラムを Sun MTP のプリプロセッサである `kixclt` を通じて渡します。`kixclt` は、CICS コマンドレベル API 文を Sun MTP 実行時システムに対する COBOL 呼び出しに変換します。
3. 変換されたソースを適切なコンパイラでコンパイルします。

   この 3 つの手順を makefile に取り込むことにより、コンパイルプロセスを自動化できます。

## ▼ Oracle COBOL プログラムをコンパイルする

1. Oracle Pro*Cobol プリプロセッサを使用して、中間ソースファイルを生成します。
2. Sun MTP `kixclt` プリプロセッサを使用して、COBOL ソースプログラムを生成します。
3. COBOL コンパイラを使用して、プログラムをコンパイルします。

   または、次のディレクトリにあるサンプルの makefile をカスタマイズすることもできます。

   - **Server Express**: `$UNIKIX/examples/oracle/cobol_mf`
   - **ACUCOBOL-GT**: `$UNIKIX/examples/oracle/cobol_acu`

## ▼ DB2 UDB COBOL プログラムをコンパイルする

1. Sun MTP `kixclt` プリプロセッサを使用して、COBOL ソースプログラムを生成します。
2. COBOL コンパイラを使用して、プログラムをコンパイルします。このコンパイラでは DB2 文もコンパイルします。

   または、`$UNIKIX/examples/db2udb/cobol_mf` ディレクトリにあるサンプルの makefile をカスタマイズすることもできます。

## ▼ Sybase COBOL プログラムをコンパイルする

1. Sybase の組み込み SQL/COBOL プリプロセッサを使用して、中間ソースファイルを生成します。
2. Sun MTP `kixclt` プリプロセッサを使用して、COBOL ソースプログラムを生成します。
3. COBOL コンパイラを使用して、プログラムをコンパイルします。

   または、`$UNIKIX/examples/sybase/cobol_mf` ディレクトリにあるサンプルの makefile をカスタマイズすることもできます。

# Oracle PL/I プログラムのコンパイル

アプリケーションが Oracle RDBMS 独自の SQL 文、CICS コマンドレベル API 文、および標準 Open PL/I 文を含んでいる場合、それらを実行に適した形式に変換する必要があります。

## ▼ プログラムをコンパイルする

1. Oracle プリプロセッサを通じてソースコードを渡します。このプリプロセッサは、組み込み SQL 文を Oracle 実行時システムに対する PL/I 呼び出しに変換し、中間ソースファイルを生成します。
2. 変換されたプログラムを Sun MTP `kixplt` プリプロセッサを通じて渡し、CICS コマンドレベル API 文を Sun MTP 実行時システムに対する PL/I 呼び出しに変換します。
3. 変換されたソースを Open PL/I コンパイラでコンパイルします。

   この 3 段階の手順を UNIX makefile に取り込むことにより、プロセスを自動化できます。Liant Open PL/I のマニュアルを参照してください。

# 第11章

# ソケットによる通信

クライアントアプリケーションは、TCP/IP ソケットまたは SSL (Secure Socket Layer) ソケットを使用して Sun MTP 領域と通信できます。各ソケットには、それぞれ独自のサーバープロセスがあります。

Sun MTP ソケットサーバー `unikixsock` は、受信要求を既定の TCP ポートで待機します。要求を受信するとソケット接続が確立され、その要求がトランザクションサーバーに転送されて処理されます。SSL サーバー `unikixssl` は、要求をトランザクションサーバーに転送する前に証明書の交換が必要になる点を除いて、ほとんど同じ方法で処理します。証明書の交換は、「SSL ハンドシェーク」と呼ばれます。ハンドシェークが成功すると、暗号化ソケット接続が確立されます。

この章の内容は、次のとおりです。

# TCP/IP ソケット

この節では、unikixsock サーバープロセスの機能およびメッセージの送受信の方法について説明します。

## unikixsock サーバープロセス

unikixsock サーバープロセスは、次のように機能します。

1. -p オプションによって unikixmain コマンド行でポート番号が指定されている場合、unikixmain は unikixsock サーバープロセスを開始します。このプロセスは、指定されたポート番号にバインドされ、そのポートに対する待機呼び出しを発行します。unikixmain については、『Sun Mainframe Transaction Processing 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。
2. ソケットクライアントプログラムは、unikixsock プロセスに接続するとき、261 ページの「メッセージの送信」に示す標準 IBM 形式に準拠しているデータの初期ブロックを渡します。
3. unikixsock は、処理を行うトランザクションサーバーにクライアントソケットを渡します。
4. トランザクションサーバーはソケットユーザー出口を呼び出し、クライアントからのソケットメッセージを読み取ります。そのメッセージが標準 IBM 形式に準拠している場合、そのメッセージは未変更のユーザー出口によって解析されます。メッセージが標準形式以外の場合、メッセージの解析用にユーザー出口をカスタマイズする必要があります。ソケットユーザー出口のカスタマイズについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。
5. メッセージが受信されて解析されると、トランザクションサーバーは要求されたトランザクションを開始し、COMMAREA でトランザクションにクライアントソケットのファイル記述子および受信したユーザーデータを渡します。

この時点で、ソケットクライアントプログラムがトランザクションプログラムからのデータを待機するので、トランザクションが開始されたことを確認できます。また、この時点から、トランザクションプログラムとソケットクライアントプログラムは、ソケット接続を使用してデータを交換できます。トランザクションプログラムは、標準 UNIX ソケットライブラリコールを使用します。このコールは C 言語呼び出しなので、トランザクションプログラムには COBOL CALL 文を使用する必要があります。例として、$UNIKIX/src/socket ディレクトリにある SOCK00.cl2 プログラムを参照してください。

## ▼ ソケットインタフェースを設定する

1. **リスナー機能をセットアップします。**

   TCP/IP ソケットリスナー機能の設定については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

2. **トランザクションプログラムをコンパイルします。**

   `$UNIKIX/src/socket/SOCK00.cl2` にあるプログラムの例を参照してください。

3. **PCT および PPT のプログラムのエントリを作成します。**

4. **ソケットクライアントプログラムをコンパイルして実行します。**

   `$UNIKIX/src/socket/sock00.c` プログラムの例を参照してください。

5. **ソケットクライアントプログラムから、ソケット接続を通じてメッセージを送信します。**

   ソケット接続によるメッセージの送受信の手順については、次の節で説明します。

# メッセージの送信

待機プロセスでは、最初の送信が遠隔ユーザーアプリケーションによって既定の形式で行われる必要があります。最初の送信のあと、ユーザーは後続の送信を行う前に応答を待機します。

最初の送信のデフォルト (IBM 標準) の入力形式は、次のとおりです。

*TRANID*`[,`*User-Data*`][,`*XX* `[,`*HHMMSS*`]]`

| オプション | 説明 |
|---|---|
| *TRANID* | トランザクション識別子 (1 ～ 4 文字)。Sun MTP PCT に存在する必要があります。 |
| *User-Data* | オプションのテキスト (35 文字まで)。<br>コンマはフィールドの区切り文字として解釈されるので、データには文字形式および 2 進形式のどちらにもコンマを含めません。10 進数 44 [16 進数の 2C] がコンマとして解釈されます。 |
| *XX* | オプションの起動タイプ。<br>`IC` または `ic`: 間隔制御<br>`TD` または `td`: 一時データ<br>このフィールドが空白の場合は、ただちに起動します。 |
| *HHMMSS* | 間隔制御を使用してトランザクションを開始した場合、間隔の時間、分、秒を表すために使用されるオプションのフィールドです。 |

コード例 11-1 に示す例によって、初期メッセージが作成されます。この例は、$UNIKIX/src/socket/sock00.c にあります。このコードがエラーなく実行されるには、領域に SOCK トランザクションを設定する必要があります。

**コード例 11-1**　　初期メッセージの作成

```
char ibuffer[4+1+35]; /* transid            4 bytes
                                   ','      1 byte
                                   data     35 bytes */

/* build the initial socket message */
memset(ibuffer, 0, sizeof(ibuffer));
memcpy(ibuffer,   trans,  4);
memcpy(ibuffer+4, ",",    1);
memcpy(ibuffer+5, data,  35);

/* send initial message to invoke the transaction */
if (send(fd, ibuffer, sizeof(ibuffer), 0) <= 0) {
       printf("sock00:send error %d\n", errno);
       exit(EXIT_FAILURE);
}
```

# メッセージの受信

次の例に示すように、トランザクションプログラムに COMMAREA を定義する必要があります。このコード例は、サンプルプログラム $UNIKIX/src/socket/SOCK00.cl2 からの抜粋です。

**コード例 11-2**　　ソケット COMMAREA を定義する linkage セクション

```
linkage section.
01  DFHCOMMAREA.
       05 GIVE-TAKE-SOCKET      PIC 9(8) COMP.
       05 LSTN-NAME             PIC X(8).
       05 LSTN-SUBNAME          PIC X(8).
       05 CLIENT-IN-DATA        PIC X(36).
       05 SOCKADDR-IN-PARM.
           15 SIN-FAMILY        PIC 9(4) COMP.
           15 SIN-PORT          PIC 9(4) COMP.
           15 SIN-ADDRESS       PIC 9(8) COMP.
           15 SIN-ZERO          PIC X(8).
```

サンプルプログラムには、メッセージの受信、データの表示、およびメッセージの送信用のコードも含まれています。コード例 11-3 は、メッセージ受信用の SOCK00.cl2 のコードです。

出力域がロードされると、メッセージで特定されたトランザクションが実行されます。ユーザーアプリケーションは、RECEIVE 呼び出しをしないで、受信データである CLIENT-IN-DATA を表示できます。

**コード例 11-3**　メッセージの受信 - ソケット (1 / 2)

```
working-storage section.
       *
       * program buffers
       *
       77 ws-recv-msg-size           pic s9(8) comp value 4096.
       77 ws-recv-buf                pic x(4096).
       77 ws-recv-total              pic s9(8) comp value 0.
       77 ws-recv-left               pic s9(8) comp value 0.
       77 ws-flags                   pic s9(8) comp value 0.
...

       *
       * set up the receive buffer
       *
              move low-values to ws-recv-buf.
              set ws-recv-total to zero.
              compute ws-recv-left = ws-recv-msg-size.
       *
       * receive data
       *
       recv-1.
              call "recv" using by value GIVE-TAKE-SOCKET,
                  by reference ws-recv-buf(1+ws-recv-total:ws-recv-left),
                  by value ws-recv-left,
                  by value ws-flags.
       *
       * test what was received and decide what we should do
       *
              if return-code < zero
                  display 'SOCK00:recv error ',
                  go to socket-error.

              if return-code = zero
                  display 'SOCK00:client disconnected',
                  go to socket-error.
       *
```

**コード例 11-3**　メッセージの受信 - ソケット (2 / 2)

```
        * have we received all the data yet?
        *
               compute ws-recv-total = ws-recv-total + return-code.
               compute ws-recv-left = ws-recv-msg-size - ws-recv-total.
        *
        * not yet
        *
               if ws-recv-left > 0 go to recv-1.
        *
        * received all the data
        *
               display 'SOCK00:receive buffer   =', ws-recv-buf(1:50).
```

# SSL (Secure Socket Layer)

SSL (Secure Socket Layer) によって、認証処理が必要な改竄防止暗号化通信のために、アプリケーションがソケットを使用できるようになります。SSL は、インターネットのような安全が確保されていないネットワークを通じて、情報交換を安全に行えるように設計されています。これにより、SSL 対応サーバーおよび SSL クライアントが相互に認証し合って、双方から暗号化接続を確立できます。

SSL サーバー認証により、クライアントがサーバーの識別情報を確認できます。SSL 対応のクライアントソフトウェアは、公開鍵暗号の標準的な技術によって、サーバーの証明書が有効であること、およびその証明書が信頼できる認証局 (CA) のクライアントのリストにある CA から発行されていることを確認します。たとえば、ユーザーがネットワーク経由でクレジットカード番号を送信する場合や、受信したサーバーの識別情報をチェックする場合に、この確認が重要な役割を果たします。

SSL クライアント認証により、サーバーがクライアントの識別情報を確認できます。サーバー認証で使用されるのと同じ技術によって、SSL 対応のサーバーソフトウェアは、クライアントの証明書が有効であること、およびその証明書が信頼できる認証局 (CA) のサーバーのリストにある CA から発行されていることを確認します。たとえば、銀行が機密の財務情報を顧客に送信する場合や、受信者の識別情報をチェックする場合に、この確認が重要な役割を果たします。

暗号化 SSL 接続では、クライアント/サーバー間で送信されるすべての情報が送信ソフトウェアで暗号化され、受信ソフトウェアで複合化される必要があり、これにより高度な機密性が実現します。機密性は、民間取引の双方の当事者にとって重要です。さらに、暗号化 SSL 接続で送信されるすべてのデータは、改竄を検知するメカニズムによって保護されます。つまり、データが転送中に変更されたかどうかが自動的に判定されます。

# unikixssl サーバープロセス

クライアントアプリケーションは、SSL を使用して遠隔 Sun MTP 領域と通信できます。SSL サーバープロセス unikixssl は、受信要求を既定の TCP ポートで待機します。処理を行うトランザクションサーバーに要求を転送する前に、SSL ハンドシェークが発生します。ハンドシェークによって、クライアントおよびサーバーは相互に検査を行い、暗号技術のネゴシエーションを行うことができます。ハンドシェークが成功すると、要求がトランザクションサーバーに転送されます。クライアント/サーバー間を流れるすべてのデータは、ハンドシェーク時にネゴシエーションが行われた暗号技術を使用して暗号化されます。

unikixssl サーバープロセスは、次のように機能します。

1. -p オプションによって unikixmain コマンド行でポート番号が指定されている場合、unikixmain は unikixsock サーバープロセスを開始します。このプロセスは、指定されたポート番号にバインドされ、そのポートに対する待機呼び出しを発行します。unikixmain については、『Sun Mainframe Transaction Processing 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。unikixssl プロセスは、unikixsock を使用してトランザクションサーバーとの接続を確立します。

2. unikixssl は、unikixrc.cfg ファイルのエントリによって制御されます。SslServer*Active を True に設定して、unikixssl プロセスを開始します。SslServer*Port によって、unikixssl がクライアント接続を待機するポートが指定されます。SslServer*Sockport によって、unikixsock の待機ポートが指定されます。

3. SSL クライアントプログラムが unikixssl と接続するとき、SSL ハンドシェークが発生します。クライアントプログラムが unikixssl 証明書を認証し、unikixssl がクライアント証明書を認証して、暗号技術のネゴシエーションが行われます。ハンドシェークが成功すると、unikixssl は、処理を行うトランザクションサーバーにクライアントソケットを渡します。

4. トランザクションサーバーはソケットユーザー出口を呼び出し、クライアントからのソケットメッセージを読み取ります。そのメッセージが標準 IBM 形式に準拠している場合、そのメッセージは未変更のユーザー出口によって解析されます。メッセージが標準形式以外の場合、メッセージの解析用にユーザー出口をカスタマイズする必要があります。ソケットユーザー出口のカスタマイズについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

5. メッセージが受信されて解析されると、トランザクションサーバーは要求されたトランザクションを開始し、COMMAREA でトランザクションにクライアントソケットのファイル記述子および受信したユーザーデータを渡します。

---

**注 –** SSL クライアントの COMMAREA で渡される遠隔アドレスは、unikixssl のアドレスであり、クライアントのアドレスではありません。

---

この時点で、SSL クライアントプログラムがトランザクションプログラムからのデータを待機するので、トランザクションが開始されたことを確認できます。また、この時点から、トランザクションプログラムと SSL クライアントプログラムは、ソケット接続を使用してデータを交換できます。トランザクションプログラムは、標準 UNIX ソケットライブラリコールを使用します。このコールは C 言語呼び出しなので、トランザクションプログラムには COBOL `CALL` 文を使用する必要があります。例として、`$UNIKIX/src/socket` ディレクトリにある `SSLSOCK0.cl2` プログラムを参照してください。この例の詳細は、`$UNIKIX/src/socket/README.ssl` ファイルを参照してください。

# メッセージの送信

`unikixssl` では、最初のメッセージがクライアントプログラムによって既定の形式で送信される必要があります。最初の送信のあと、クライアントプログラムは後続の送信を行う前に応答を待機します。

最初の送信の入力形式は、次のとおりです。

*TRANID*`[`,*User-Data*`][`,*XX* `[`,*HHMMSS*`]]`

| オプション | 説明 |
|---|---|
| *TRANID* | トランザクション識別子 (1 ～ 4 文字)。Sun MTP PCT に存在する必要があります。 |
| *User-Data* | オプションのテキスト (35 文字まで)。<br>コンマはフィールドの区切り文字として解釈されるので、データには文字形式および 2 進形式のどちらにもコンマを含めません。10 進数 44 [16 進数の 2C] がコンマとして解釈されます。 |
| *XX* | オプションの起動タイプ。<br>`IC` または `ic`: 間隔制御<br>`TD` または `td`: 一時データ<br>このフィールドが空白の場合は、直ちに起動します。 |
| *HHMMSS* | 間隔制御を使用してトランザクションを開始した場合、間隔の時間、分、秒を表すために使用されるオプションのフィールド。 |

コード例 11-4 は、初期メッセージを作成する $UNIKIX/src/socket/sslsock00.c のコードです。このコードがエラーなく実行されるには、領域に SSL0 トランザクションを設定する必要があります。SSL0 トランザクションの設定については、$UNIKIX/src/socket/README.ssl ファイルを参照してください。

**コード例 11-4　　初期メッセージの作成**

```
char ibuffer[4+1+35];   /* transid            4 bytes
                                   ','         1 byte
                                   data        35 bytes */

/* build initial socket message */
memset(ibuffer, 0, sizeof(ibuffer));
memcpy(ibuffer,   trans,  4);
memcpy(ibuffer+4, ",",    1);
memcpy(ibuffer+5, data,  35);

/* Initial send to invoke transaction */
if (PR_Write(sock, ibuffer, sizeof(ibuffer)) <= 0) {
       exitErr("PR_Write");
}
```

# メッセージの受信

次の例に示すように、トランザクションプログラムに出力域を定義する必要があります。このコードは、サンプルプログラム $UNIKIX/src/socket/SSLSOCK0.cl2 からの抜粋です。

**コード例 11-5　　出力域の定義**

```
01  DFHCOMMAREA.
       05 GIVE-TAKE-SOCKET       PIC 9(8) COMP.
       05 LSTN-NAME              PIC X(8).
       05 LSTN-SUBNAME           PIC X(8).
       05 CLIENT-IN-DATA         PIC X(36).
       05 SOCKADDR-IN-PARM.
           15 SIN-FAMILY         PIC 9(4) COMP.
           15 SIN-PORT           PIC 9(4) COMP.
           15 SIN-ADDRESS        PIC 9(8) COMP.
           15 SIN-ZERO           PIC X(8).
```

初期クライアントデータは、CLIENT-IN-DATA で渡されます。サンプルプログラムには、メッセージの受信、データの表示、およびメッセージの送信のためのコードも含まれています。次の例は、メッセージ受信のための SSLSOCK0.cl2 のコードです。

**コード例 11-6**　メッセージの受信 - SSL (1 / 2)

```
working-storage section.
       *
       * program buffers
       *
       77 ws-recv-msg-size           pic s9(8) comp value 4096.
       77 ws-recv-buf                pic x(4096).
       77 ws-recv-total              pic s9(8) comp value 0.
       77 ws-recv-left               pic s9(8) comp value 0.
       77 ws-flags                   pic s9(8) comp value 0.
...

       *
       * set up the receive buffer
       *
              move low-values to ws-recv-buf.
              set ws-recv-total to zero.
              compute ws-recv-left = ws-recv-msg-size.
       *
       * receive data
       *
       recv-1.
              call "recv" using by value GIVE-TAKE-SOCKET,
                  by reference ws-recv-buf(1+ws-recv-total:ws-recv-left),
                  by value ws-recv-left,
                  by value ws-flags.
       *
       * test what was received and decide what we should do
       *
              if return-code < zero
                  display 'SSLSOCK0:recv error ',
                  go to socket-error.

              if return-code = zero
                  display 'SSLSOCK0:client disconnected',
                  go to socket-error.
       *
```

**コード例 11-6**　メッセージの受信 - SSL (2 / 2)

```
       * have we received all the data yet?
       *
              compute ws-recv-total = ws-recv-total + return-code.
              compute ws-recv-left = ws-recv-msg-size - ws-recv-total.
       *
       * not yet
       *
              if ws-recv-left > 0 go to recv-1.
       *
       * received all the data
       *
              display 'SSLSOCK0:receive buffer   =', ws-recv-buf(1:50).
```

クライアント証明書情報は、EXEC CICS EXTRACT CERTIFICATE API 呼び出しを使用して取得できます。詳細は、52 ページの「EXTRACT CERTIFICATE」を参照してください。

# SSL ユーザー出口

SSL クライアント証明書の検証ユーザー出口は、共有ライブラリ $UNIKIX/lib/libkxsslxit.so にあります。このユーザー出口によって、独自の証明書取り消しチェックなどに基づいてクライアント証明を拒否するコードを記述できます。

SSL ユーザー出口のカスタマイズについては、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

# サーバー証明書共通名の問題

ハンドシェーク時に SSL クライアントは、サーバー証明書の共通名 (CN) と、サーバーへの接続の際にクライアントで使用されるホスト名が一致することを確認します。この名前は完全に一致する必要があります。一致しない場合、次のエラーメッセージとともに、サーバー証明書がクライアントによって拒否されます。

```
-12276 SSL_ERROR_BAD_CERT_DOMAIN
```

たとえば、unikixssl サーバーが、名前が myhost、IP アドレスが 172.16.678.789 のホスト上で実行されているとします。この場合、サーバー証明書の共通名は、myhost にすることも、myhost の IP アドレスにすることもできます。クライアントは、unikixssl サーバーと接続するとき、サーバー証明書の共通名と一致するホスト名または IP アドレスを使用します。証明書の共通名が myhost のときにクライアントが myhost の IP アドレスを使用すると、接続は確立されてもハンドシェークは失敗します。 IP アドレスはサーバー証明書の共通名と完全に一致しないからです。

# 第 12 章

# WebSphere MQ の使用法

この章では、Sun MTP ソフトウェアによる IBM WebSphere バージョン 5 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。



次の図は、Sun MTP 領域と WebSphere MQ (MQ) の関係を示します。

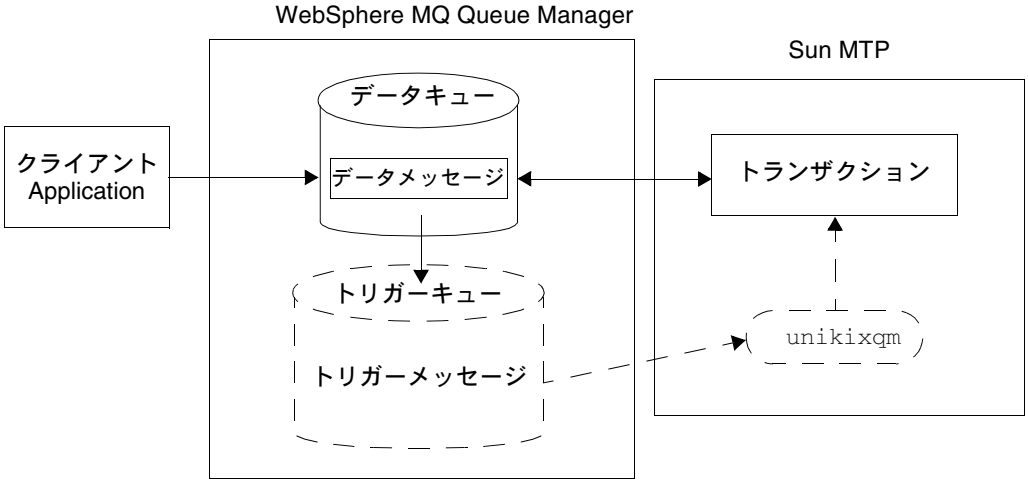


図 12-1　WebSphere MQ Process Flow

図 12-1 の各要素の意味は、次のとおりです。

**クライアントアプリケーション**: 入力データキューにメッセージを作成します。クライアントアプリケーションは、Sun MTP から独立できます。

**データキュー**: アプリケーションメッセージを保持します。アプリケーションメッセージの保存により、MQ がトリガーメッセージを生成します。

**トリガーキュー**: キューマネージャーによって生成されたメッセージを含みます。トリガーメッセージには、領域がトランザクションを開始するために使用する情報、およびトリガーイベントを生成した元のデータメッセージをトランザクション処理プログラムが検索するために使用する情報が含まれています。

**unikixqm**: トリガーキューを監視し、トリガーメッセージのアプリケーション ID に基づいてトランザクションを開始します。アプリケーション ID は、プログラム管理テーブル (PCT) に設定されているトランザクション ID の値と同じである必要があります。

**トランザクション**: `unikixqm` デーモンによって開始されたユーザーアプリケーション。アプリケーションは、MQ API を使用して、データキューからメッセージを読み取ります。

# Sun MTP によってサポートされる WebSphere MQ アプリケーションの種類

Sun MTP MQ アプリケーションは、次の 2 つの方法で開始できます。

- MQ トリガー機構の使用

  MQ トリガー機構は、特定の条件が満たされた場合に、トリガーする定義済みアプリケーションを有効にします。たとえば、メッセージが MQ データキューに書き込まれた場合などがあります。Sun MTP はこの機構を使用して、トランザクションを開始し、キューマネージャー名とキュー名をアプリケーションに提供します。

- ユーザー開始トランザクションの使用

  アプリケーションが、キューマネージャー名およびキュー名を認識します。Sun MTP をサポートしている機構によって開始されるトランザクションは、MQ API 呼び出しを発行できます。

# WebSphere MQ のアプリケーション設計ガイドライン

Sun MTP MQ 環境でアプリケーションを設計する場合、次のガイドラインに従ってください。

- プログラムは、CICS API を介して Sun MTP と対話する必要があります。
- プログラムは、MQ APIを介して MQ と対話する必要があります。
- トランザクションの整合性を維持するため、MQ を XA リソースマネージャーとして設定する必要があります。MQ XA リソースマネージャーで使用する領域を設定する方法については、『Sun Mainframe Transaction Processing ソフトウェア XA リソースマネージャーの使用』を参照してください。

MQ トリガーの使用法については、273 ページの「WebSphere MQ トリガー機構の使用法」を参照してください。トリガーの一部として実行されるトランザクションは、Sun MTP 環境向けに記述できる MQ アプリケーションのうちのひとつだけです。このようなトリガートランザクションのアプリケーション設計については、275 ページの「トリガートランザクションの例」を参照してください。

# WebSphere MQ トリガー機構の使用法

WebSphere MQ では、事前に定義された条件が満たされた場合に、MQ キューマネージャーが別のトリガーキューにメッセージを自動的に配置するように設定できるトリガー機構を実装しています。複数のキューでトリガーメッセージが生成され、それらのメッセージが 1 つのトリガーキューに自動的に配置されます。トリガー機構の詳細は、MQ のマニュアルを参照してください。

# トリガーキューの定義

Sun MTP トランザクションが MQ を通してトリガーされる前に、MQ の設定でトリガー機能を定義する必要があります。次の例は、トリガー機能を有効にするデータキューのコードです。

**コード例 12-1**　　WebSphere MQ トリガーの設定

```
DEF QL (UNIKIXMQ1) REPLACE                                  +
       DESCR ('Sun MTP MQSeries Sample Queue 1')    +
       TRIGGER                                       +
       TRIGTYPE (FIRST)                             +
       INITQ (MTP.TRIGGER.QUEUE)                 +
       PROCESS (MQTX.TRIGGER.PROC)
DEF QL (MTP.TRIGGER.QUEUE) REPLACE               +
       DESCR ('Sun MTP trigger queue')
DEF PROCESS (MQTX.TRIGGER.PROC) REPLACE             +
       DESCR ('Invoke transaction')                 +
       APPLICID (MQTX)                              +
       APPLTYPE (CICS)
```

この例のデータキュー名は UNIKIXMQ1 です。このキューに関連して、トリガーキューは MTP.TRIGGER.QUEUE という名前が付けられています。MQ がこの 2 つのキューを管理し、データキューが新しいトリガーを生成するように要求するたびに、トリガーキューにメッセージを配置します。トリガーメッセージの目的は、MQTX という名前の Sun MTP トランザクション (属性 APPLICID で定義) を開始することです。

MQSeries トリガーのサポートとは、トリガーメッセージがシステムによって作成されたときに Sun MTP トランザクションが呼び出されることです。Sun MTP トランザクションは、データキューからのメッセージを読み取ってそれを処理するように記述することもできます。トランザクションが領域によって自動的に開始されるときに、トランザクションは十分な情報を渡されてデータメッセージを読み取って処理します。

トリガーメッセージの結果としてトランザクションが開始されると、トリガーメッセージの詳細がトランザクションに渡されます。トランザクションは、EXEC CICS RETRIEVE コマンドを使用してデータを取得します。このデータの形式は、MQSeries コピーブック CMQTMLで定義されます。その構造体のフィールドは、次のとおりです。

```
10 MQTM.
15 MQTM-STRUCID       PIC X(4).
15 MQTM-VERSION       PIC S9(9) BINARY.
15 MQTM-QNAME         PIC X(48).
15 MQTM-PROCESSNAME   PIC X(48).
15 MQTM-TRIGGERDATA   PIC X(64).
15 MQTM-APPLTYPE      PIC S9(9) BINARY.
15 MQTM-APPLID        PIC X(256).
15 MQTM-ENVDATA       PIC X(128).
15 MQTM-USERDATA      PIC X(128).
```

この構造体では、キューマネージャー名を定義しません。そのため、Sun MTP トリガーのサポートでは、情報を含んでいる構造の末尾に追加フィールドを提供します。アプリケーションプログラムでは、このフィールドに次のようなデータ定義を指定する必要があります。

```
01 PARAMS.
COPY CMQTML.
10 MQTM-QMGR-NAME     PIC X(48).
```

# トリガートランザクションの例

MQ メッセージを処理するには、トリガートランザクションはキューマネージャーに接続してそのメッセージを読み取る必要があります。そのために、トランザクションでキューを空にして終了することをお勧めします。

このようなアプリケーションの擬似コードを、コード例 12-2 に示します。このアプリケーションの設計について、次の点に注意してください。

- キューが空になるまで、メッセージは処理され続けます。ほとんどのアプリケーションは、この設計に従う必要があります。
- メッセージを取得する MQGET() 呼び出しは、非ブロッキングである必要があります。つまり、キューがドレーンされるとトランザクションは終了します。さらに、領域のリソースは MQ メッセージの受信の待機によって消費されません。

コード例 12-2　WebSphere MQ トリガートランザクションの擬似コード

```
Retrieve the start data (Using EXEC CICS RETRIEVE)
Connect to Queue Manager
       (Using MQCONN() with the QueueManagerName as
MQTM-QMGR-NAME)
Open the Queue
       (Using MQOPEN() with the queue name as MQTM-QNAME)
Read a message from Queue (Using MQGET())
while (Message was read)
do
       Process the message
       Read a message from the queue (Using MQGET())
end
Close the Queue (Using MQCLOSE())
Disconnect from the Queue Manager (Using MQDISC())
```

# リソースとトランザクションのセキュリティー保護

MQ は、MQ リソースへのアクセスを制御します。MQ セキュリティー機能を使用して、MQ リソースを保護します。

Sun MTP トランザクションセキュリティーを使用して、Sun MTP MQ トランザクションを保護します。トランザクションレベルのセキュリティーはトリガートランザクションを制御しないので、プログラムは MQ のセキュリティー機能を使用して、トランザクションが承認されていない作業を行われることを防ぎます。

# WebSphere MQ アプリケーションを実行するための Sun MTP の設定

この節では、Sun MTP が MQ アプリケーションを実行できるようにするための手順について説明します。

- MQ ソフトウェアの設定。詳細は、277 ページの「WebSphere MQ の設定」を参照してください。
- MQ をサポートする Sun MTP システムソフトウェアの構築。詳細は、277 ページの「WebSphere MQ を使用するために Sun MTP を構築する」を参照してください。
- MQ を使用するための領域 (複数可) の設定。詳細は、278 ページの「WebSphere MQ を使用する領域を設定する」を参照してください。
- MQ トリガー機構を使用するための領域 (複数可) の設定。

## ▼ WebSphere MQ の設定

- **MQ トランザクションに対して適切に MQ を設定します。**

  キュー、プロシージャー、チャネルなどをすべて定義します。トランザクションのサポートを必要とする場合、XA 環境で MQ を設定する必要があります。詳細は、『Sun Mainframe Transaction Processing ソフトウェア XA リソースマネージャーの使用』を参照してください。

## ▼ WebSphere MQ を使用するために Sun MTP を構築する

1. **この領域を停止します。**
2. **`$UNIKIX/src` ディレクトリに移動します。**
3. **`kixinstall` ユーティリティーを実行します。**
4. **「Third Party Packages」画面で、MQ オプションを選択します。**
5. **『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』に記載されているとおりに、実行可能プログラムを構築します。**

## ▼ WebSphere MQ を使用する領域を設定する

1. MQ を使用する領域 (複数可) を特定します。
2. 領域設定ファイルで、次のように追加します。
   a. WebSphere MQ 製品のインストールパスを指す `MQSERIES` 環境変数を追加します。
   b. `$MQSERIES/bin` を `PATH` 環境変数に追加します。
   c. `$MQSERIES/lib` を `LD_LIBRARY_PATH` 環境変数に追加します。
   d. MQ クライアントライブラリを使って Sun MTP を構築する場合、`MQSERVER` 環境変数を適切な値で追加します。

      詳細は、『WebSphere MQ Clients Guide』を参照してください。

## ▼ WebSphere MQ トリガー機構を使用する領域を設定する

1. MQ トリガー機構を使用する領域を特定します。
2. `$KIXSYS/unikixrc.cfg` ファイルを編集して、`MQServer*` エントリを追加します。

   領域に `unikixrc.cfg` ファイルがない場合、`unikixrc` ファイルを `$UNIKIX/lib` から `$KIXSYS` にコピーし、そのファイル名を `unikixrc.cfg` に変更します。`MQServer*` エントリを次のように追加します。

```
MQServer*Active:                True
MQServer*QueueManagerName:      "QueueManagerName"
MQServer*QueueName:             "TriggerQueueName"
```

   *QueueManagerName* は、`MQSERVER` 環境変数で指定されたホストで実行するキューマネージャーの名前です (デフォルトのキューマネージャーを使用する場合は、空白 (`""`)。*TriggerQueueName* は、`unikixqm` プロセスがトリガーメッセージを監視するキューの名前です。

3. まだ定義されていない場合、`MQSERVER` 環境変数を適切な値で領域セットアップファイルに追加します。

   Sun MTP トリガーデーモンは、MQ クライアントライブラリを使って構築されているため、環境変数が必要です。

   詳細は、『WebSphere MQ Clients Guide』を参照してください。

## アプリケーションの準備

他の Sun MTP アプリケーションプログラムをコンパイルするのと同じ方法で、MQ API に対して記述したアプリケーションプログラムをコンパイルします。プログラムをコンパイルするために必要な設定情報 (C 言語インクルードファイル、COBOL コピーブックの位置など) については、MQ のマニュアルを参照してください。

# WebSphere MQ サンプルアプリケーションの実行

サンプルのトリガートランザクション `KIXMQ01.clt` のソースおよびオブジェクト形式は、次のディレクトリにあります。

- **Server Express**: `$UNIKIX/examples/mq/cobol_mf`
- **ACUCOBOL-GT**: `$UNIKIX/examples/mq/cobol_acu`

このサンプルは、ファイル転送プログラムのように動作し、プロセスで使用されるリソースの MQ 定義および Sun MTP 定義を含んでいます。このサンプルトランザクションを、トリガーアプリケーションのモデルとして使用できます。

サンプルには、次のものがあります。

- アプリケーションの実行に関する説明を含む `README.txt` ファイル
- ファイルから MQ キューに行をコピーする C 言語プログラム
- MQ キューからのメッセージを読み取り、Sun MTP の一時記憶域キューにデータを配置する COBOL プログラム
- MQ 構成ファイル

# 第13章

# MQ-JMS Bridge の使用法

**注 –** この章を読む前に 第 12 章を読んでください。

Sun MTP MQ-JMS Bridge では、Sun MTP WebSphere MQ (MQ) トリガー機構と Java Message Service (JMS) API 向けに開発されたアプリケーションとの間をマッピングする機能を提供します。MQ-JMS Bridge は、Sun MTP `unikixqm` デーモンが発する MQ トリガーメッセージを処理し、関連のアプリケーションメッセージを構成済みのユーザーアプリケーションプログラムに転送します。基盤となる JMS メッセージングスタイルとしては、JMS ポイントツーポイントモデルが使用されます。

管理者のレベルでは、MQ-JMS Bridge によって、トランザクション開始機構を抽象化することができます。ターゲットアプリケーションは、開始手段にかかわらず、実行する作業が記述されたアプリケーションメッセージだけに対応します。アプリケーション開発者のレベルでは、基盤となる MQ キューマネージャーおよびアプリケーションキューの管理から解放されます。アプリケーションは、JMS `MessageListener.onMessage()` メソッドのアプリケーション実装に対する呼び出しに応答するだけです。

この章の内容は、次のとおりです。

# MQ-JMS Bridge の動作

MQ-JMS Bridge は、既存の MQ トリガーフレームワークに構築されます。このフレームワークでは、MQ トリガーメッセージが監視対象キューに配置されると unikixqm デーモンに通知されます。それによって unikixqm は、指定された Sun MTP トランザクションを開始して関連アプリケーションキューを処理します。MQ-JMS Bridge をユーザーアプリケーションではなく初期アプリケーションと定義して、処理フローに間接参照のレベルを指定することによって、このフレームワークを拡張できます。

MQ-JMS Bridge が担当するのは、MQ トリガーメッセージを処理すること、および提供された情報を使用してアプリケーションキューにアクセスしてユーザーアプリケーションプログラムのインスタンスを取得することです。さらに MQ-JMS Bridge は、アプリケーションキューを処理して各メッセージを読み取り、これを MessageListener.onMessage メソッドを通じてユーザー JMS アプリケーションに渡します。次の図に、その基本的な概念を示します。

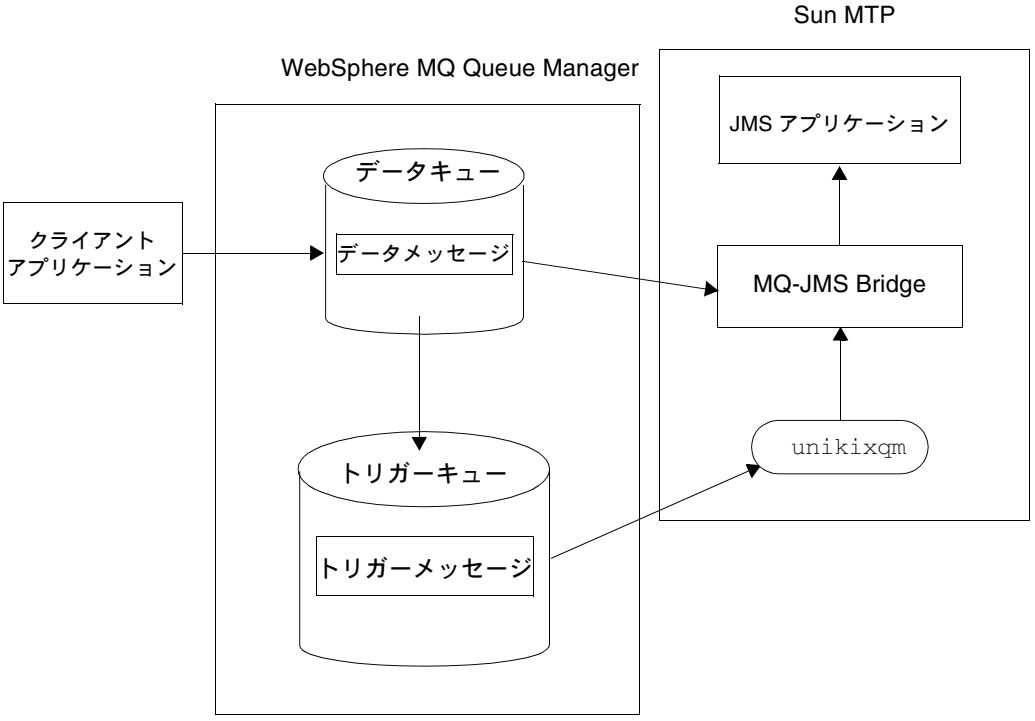


図 13-1　MQ-JMS Bridge プロセスフロー

図 13-1 の各要素の意味は、次のとおりです。

**クライアントアプリケーション**: 入力データキューにメッセージを作成します。クライアントアプリケーションは、Sun MTP から独立しています。

**データキュー**: Sun MTP トランザクションを定義するトリガーメッセージをトリガーキューに配置します。

**トリガーキュー**: キューマネージャーに設定されたプロシージャーを通じて自動化メッセージを受け取ります。トリガーメッセージには、Sun MTP がトランザクションを開始するために使用する情報、およびトリガーイベントを生成した元のデータメッセージをトランザクション処理プログラムが検索するために使用する情報が含まれています。

**unikixqm**: トリガーキューを監視し、トリガーメッセージのアプリケーション ID に基づいてトランザクションを開始します。JMS アプリケーショントランザクションの場合、`unikixqm` によって開始されるトランザクションは常に MQ-JMS Bridge です。

**MQ-JMS Bridge**: データキューからメッセージを取り出し、該当する JMS アプリケーションを開始し、呼び出し時にメッセージを渡します。

**JMS アプリケーション**: `MessageListener` インタフェースを実装します。そのため、`onMessage` メソッドにタイプ `Message` のパラメータを 1 つ渡す必要があります。JMS アプリケーショントランザクションは、MQ-JMS Bridge を通じてデータキューから元のデータメッセージを受け取っています。それ以降の MQ との対話は、JMS アプリケーションが担当します。JMS アプリケーションは、MQ-JMS Bridge で使用されるものと同じ設定情報を使用してキューマネージャーとの接続を確立できます。JMS アプリケーションからのキューマネージャーの対話の例については、`$UNIKIX/examples/mq/jms` ディレクトリにあるサンプルプログラム `GetAliasJMSListenerEXMQJMS` を参照してください。

# MQ-JMS Bridge の属性の定義

MQ-JMS Bridge は、コード例 12-1 の MQ トリガー設定を展開して、新しいトリガーメッセージが生成されたときにスケジューリングされるターゲットユーザーアプリケーションを定義します。

ユーザーアプリケーションを定義するには、次のいずれかの方法を使用します。

- 次の例に示すように、`PROCESS` メンバーの `USERDATA` 属性に完全修飾クラス名を明示的に指定します。
- 290 ページの「アプリケーションマッピング」に示す方法を使用します。

**コード例 13-1**　　MQ-JMS Bridge の属性の定義

```
DEF PROCESS (MQJV.TRIGGER.PROC) REPLACE                +
  DESCR ('Invoke instance of MQJMSBridge')             +
  USERDATA ('scouser.GetAliasJMSListenerExMQJMS')      +
  APPLICID (MQJV)                                      +
  APPLTYPE (CICS)
```

この例の設定では、unikixqm デーモンが MQJV トランザクションを開始します。トランザクションを開始するメッセージには、トリガープロパティー情報が含まれていて、それによりターゲットアプリケーション名 scouser.GetAliasJMSListenerExMQJMS を指定する USERDATA 属性が取り込まれます。

# 領域の設定

この節では、MQ-JMS Bridge を使用するための領域の設定方法について説明します。

MQ-JMS Bridge を設定する前に、次のことを実行してください。

- 領域で Java サポートが有効となるように設定します。第 8 章を参照してください。
- $KIXSYS/unikixrc.cfg ファイルの MQ エントリを変更して MQ を有効にします。『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。MQ の使用法については、第 12 章も参照してください。

MQ-JMS Bridge 自体を設定するには、次のことを実行してください。

- JMS グループをグループ管理テーブル (GCT) に定義します。
- Classpath.appends および Libpath.appends ファイルを変更して、必要なクラス、JAR ファイル、およびライブラリを取り込みます。
- デフォルトと異なる設定プロパティーを指定する場合、MQJMS.properties ファイルをカスタマイズします。

# GCT での jms グループの定義

MQ-JMS Bridge 製品に関連するプログラムは、jms グループに割り当てられます。領域がこのプログラムにアクセスできるようにするには、GCT に jms グループを定義する必要があります。

## ▼ jms グループを定義する

1. テーブルマネージャーで GCT を開き、PF4 キーを押してエントリを挿入します。
2. 挿入画面で、グループ名として jms、ディレクトリとして $KIXSYS/jms.dir を入力します。

   詳細は、図 13-2 を参照してください。

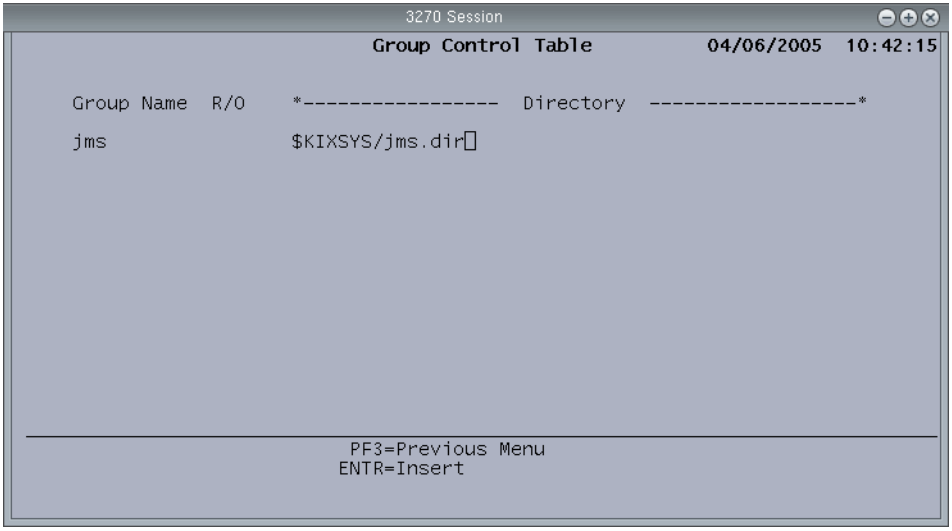


図 13-2　Group Control Table の jms Group 定義

3. Enter キーを押すとエントリが挿入され、GCT メイン画面に戻ります。
4. PF2 キーを押すと、変更がディスクに書き込まれます。
5. テーブルマネージャーを終了します。
6. 領域を停止して再起動し、変更を有効にします。

## ▼ MQ-JMS Bridge プログラムの情報を表示する

1. PPT を開き、JMQJMSB プログラムエントリにカーソルを置きます。
2. PF9 キーを押して、Java クラスの詳細画面を開きます。

   この画面には、MQ-JMS Bridge アプリケーションへの完全修飾クラスパスが表示されます。プログラム JMQJMSB は、com.sun.emp.mtp.MQJMS.MQJMSBridge という名前の Java クラスファイルに関連しています。

   図 13-3 は、JMQJMSB プログラムの Java クラスの詳細画面を示します。

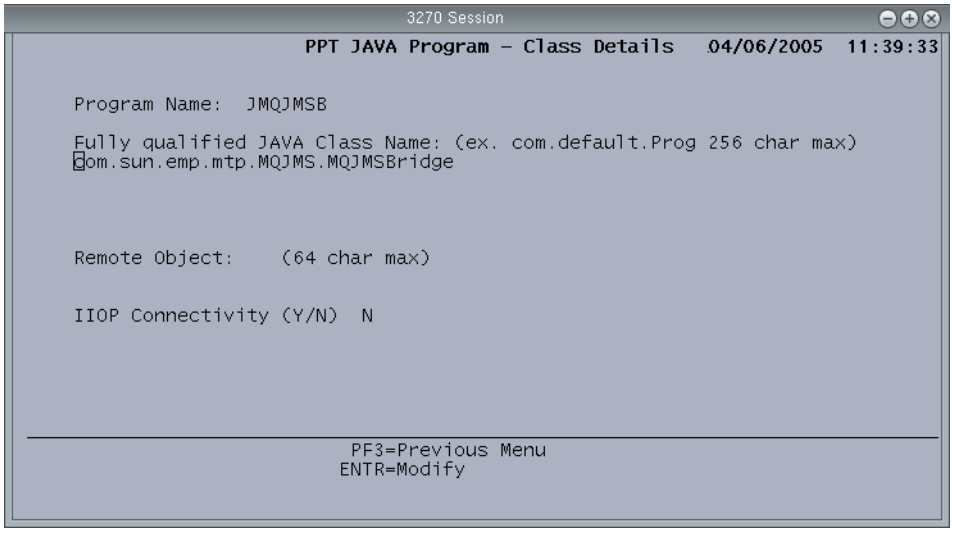


図 13-3　Processing Program Table での MQ-JMS Bridge エントリ

## ▼ トランザクション ID と MQ-JMS Bridge プログラムを関連付ける

---

**注 –** `$UNIKIX/examples/mq/jms` のサンプルアプリケーションの PCT には、ここで説明するエントリがすでに含まれています。

---

1. PCT を開き、PF4 キーを押してエントリを挿入します。
2. 「Program」フィールドに `JMQJMSB` と入力します。
3. 「Trans ID」フィールドにトランザクション識別子を入力します。

   図 13-4 は、`MQJV` トランザクションに割り当てられる `JMQJMSB` プログラムを示します。`JMQJMSB` は、トランザクション ID が発行されたときに最初に実行されるプログラムです。

   このトランザクション ID は、MQ トリガー設定のアプリケーション ID と一致する必要があります。たとえば、次の図に示すトランザクション ID (`MQJV`) は、コード例 13-1 の `APPLICID` の値セットに一致しています。

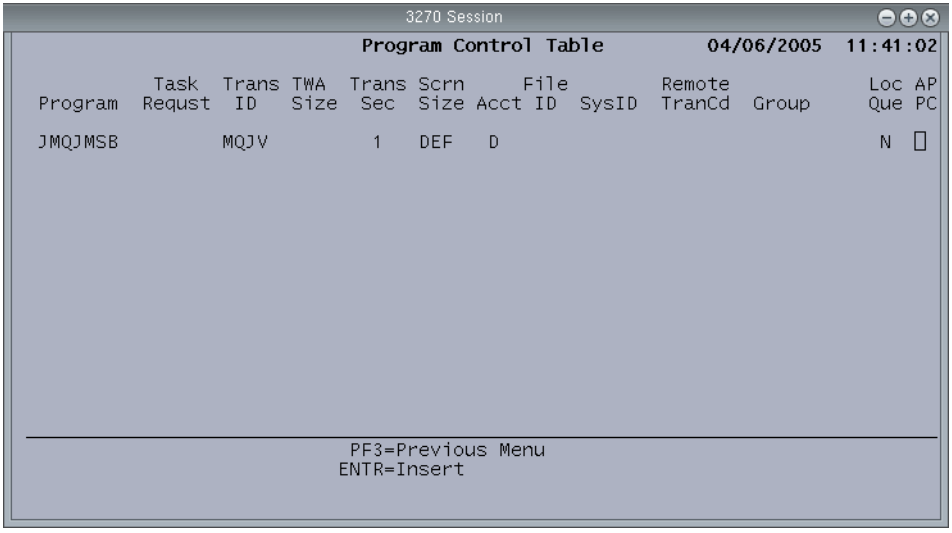


**図 13-4** Program Control Table での MQ-JMS Bridge エントリ

4. Enter キーを押して、エントリを挿入します。
5. PF2 キーを押して、テーブルをディスクに保存します。

6. **領域を停止して再起動します。**

   ここで、MQ トリガーキューと MQ-JMS Bridge アプリケーション間には関係があります。

## MQ-JMS Bridge プロパティーの指定

MQ-JMS Bridge は、製品の属性を含む構成ファイルを使用します。構成ファイルに含まれる属性は、次の 2 つのレベルで適用されます。

- MQ-JMS Bridge 全般レベル
- アプリケーションレベル

全般レベルの属性は、多くの場合、キューマネージャーの位置などの MQ に固有の設定に関連します。アプリケーション属性はマッピング機能を有効にするので、MQ-JMS Bridge がインスタンスを検出しようとする前に、ターゲットアプリケーションを他のアプリケーションに割り当てることができます。

---

**注 –** `$KIXSYS/kix_java` ディレクトリの `MQJMS.properties` ファイルを管理することによって、それぞれの Sun MTP 領域が独自の MQ-JMS 設定を持つことができます。

---

### ▼ プロパティーファイルを定義する

1. **`$UNIKIX/lib/kix_java/MQJMS.properties` ファイルを領域の `$KIXSYS/kix_java` ディレクトリにコピーします。**

   `kix_java` ディレクトリの作成については、192 ページの「クラスパスおよびライブラリパスのカスタマイズ」を参照してください。

2. **それぞれのサイトの要件に合うようにファイルを変更します。**

構成ファイルは、Java プロパティーファイルの形式に従って key=value のように記述します。次の表に、全般属性とそのデフォルト値を示します。

**表 13-1** `MQJMS.properties` ファイル

| 属性名 | デフォルト値 | 説明 |
|---|---|---|
| `application` | `MQTM-USERDATA` の値 (デフォルトアプリケーション) | 要求に対してロードされるユーザーアプリケーションの完全修飾クラス名。`MQTM-USERDATA` がトリガーメッセージで定義されない場合 (つまり、値がスペースの場合) にのみ、この値が使用されます。MQ-JMS Bridge のアプリケーションマッピング機能を使用して変更できます。 |
| `channel` | `CHANNEL1` | MQ キューマネージャーへの接続に使用するサーバーチャネルの名前。 |
| `host` | `localhost` | MQ キューマネージャーを実行しているサーバーの名前/アドレス。 |
| `port` | `1414` | MQ キューマネージャーによって使用されるポート番号。 |
| `debug` | `no` | イベントを処理する MQ-JMS Bridge が `unikixmain.dbg` ファイルにログを書き込むかどうかを指定します。サポートされる値は、次のとおりです。<br>`yes`: ログを書き込みます。<br>`no`: ログを書き込みません。<br>ログが書き込まれる処理イベントのタイプについては、295 ページの「デバッグ情報」を参照してください。 |
| `txscope` | `message` | Sun MTP 処理単位の適用範囲を指定します。サポートされる値は `message` で、これにより、各メッセージがそれぞれ 1 つの Sun MTP 処理単位内で処理されます。つまり、`onMessage()` 呼び出しが成功するたびに、`SYNCPOINT` (`Task.commit()`) が続きます。 |

使用される MQ トリガーのタイプ (`FIRST`、`EVERY`、または `DEPTH`) にかかわらず、Sun MTP に配置されたユーザー JMS アプリケーション (アプリケーションメッセージの使用先) では、MQ キューのメッセージを認識しません。ユーザー JMS アプリケーションは MQ キューから取り出されるそれぞれのメッセージによって呼び出されるので、連続する各呼び出しの間に相関関係はありません。ユーザー JMS アプリケーションが各 `onMessage()` 呼び出しを別の Sun MTP 処理単位にするために、MQ-JMS Bridge プロパティーファイルの `txscope` 属性を値 `message` に設定できます。

---

**注 –** このリリースでは、`txscope` の有効な属性は `message` のみです。

---

Sun MTP 領域および MQ キューマネージャーが同じホストで実行されている場合、デフォルトのキューマネージャーとして動作する MQ キューマネージャーが 1 つ設定されています。このとき、アプリケーションマッピングが必要ない場合には、MQ-JMS Bridge 構成ファイルも必要ありません。`$UNIKIX/examples/mq/jms` のサンプルアプリケーションは、このシナリオに基づいています。

Sun MTP 領域および MQ キューマネージャーが別々のホストにある場合、MQ キューマネージャーには、MQ クライアントアプリケーション (この場合は `unikixqm` および MQ-JMS Bridge) とキューマネージャーとの通信のために定義される特別なチャネルがあります。`MQJMS.properties` ファイルには、`MQSERVER` 環境変数に定義されているホストおよびチャネルの値が含まれる必要があります。たとえば、`MQSERVER=MYSVRCHNL/TCP/myserver` の場合、`MQJMS.properties` ファイルには次のエントリが含まれる必要があります。

```
host=myserver
channel=MYSVRCHNL
```

一般に、MQ-JMS Bridge 構成ファイルでは、`unikixrc.cfg` ファイルおよび `MQSERVER` 環境変数で `unikixqm` プロセスに対して定義されているパラメータが反映されます。Java は、実行環境からではなく、このプロパティーファイルから MQ 設定を取り出します。`$MQSERVER` については、278 ページの「WebSphere MQ を使用する領域を設定する」を参照してください。

## アプリケーションマッピング

アプリケーションマッピングは、1 つのアプリケーションターゲットを他のターゲットに割り当てる方法です。たとえば、この方法によって `MQJMSAppl_GetAlias` として識別されるアプリケーションが、`scouser.GetAliasJMSListener` として識別されるアプリケーションに置換されます。

マッピングを使用するとき、ターゲットユーザー JMS アプリケーションは、`USERDATA` 属性によって MQ `PROCESS` 定義に明示的に定義するのではなく、`MQJMS.properties` ファイルに定義します。これにより、バックエンドアプリケーションの変更は、多数存在する MQ `PROCESS` 定義を変更する代わりに、1 つの `MQJMS.properties` ファイルを変更するのみですみます。

アプリケーションマッピングを実装するには、MQ の`PROCESS` 定義の `USERDATA` 属性で、ユーザーアプリケーションの完全修飾名ではなくトークンまたはエイリアスを指定します。このトークンは、対応する値で定義されているターゲットアプリケーションのキーとして `MQJMS.properties` ファイルに定義されます。

コード例 13-2 は、論理トークン MQJMSAppl_GetAlias が JMS アプリケーション scouser.GetAliasJMSListener に割り当てられている例を示します。

**コード例 13-2**　　アプリケーションマッピング

```
MQSeries Configuration
  DEF PROCESS (MQJV.TRIGGER.PROC) REPLACE          +
       DESCR ('Invoke instance of MQJMSBridge')    +
       USERDATA ('MQJMSAppl_GetAlias')             +
       APPLICID (MQJV)                             +
       APPLTYPE (CICS)

MQ-JMS Bridge (MQJMS.properties) file
  MQJMSAppl_GetAlias=scouser.GetAliasJMSListener
```

---

**注 –** USERDATA 属性には、scouser.GetAliasJMSListener などの実際のアプリケーションターゲットが入ることもあります。詳細は、コード例 13-1 を参照してください。

---

アプリケーションマッピングの動作の詳細は、292 ページの「処理の流れ」を参照してください。

## クラスパスの設定

JCICS フレームワークでは、Sun MTP 環境変数 KIXPROGS を使用して JVM に対するアプリケーションクラスパス属性を設定します。通常、これはユーザー Java アプリケーションを領域に設定する際の方法であり、ユーザー JMS アプリケーションもこのようにして領域に対して定義されます。MQ-JMS Bridge アプリケーションは、システムアプリケーションであり、そのままの状態で標準 Sun MTP システム JAR ファイルの一部として Sun MTP に設定されます。

WebSphere MQ-JMS が MQ-JMS Bridge アプリケーションを正常にロードするには、MQSeries MQ-JMS アーカイブファイル (com.ibm.mqjms.jar、com.ibm.mq.jar) および Sun アーカイブファイル (jms.jar、jndi.jar) が使用可能である必要があります。このアーカイブは、Classpath.appends ファイルを使用して定義します。

Classpath.appends ファイルは、$KIXSYS/kix_java ディレクトリに配置します。このファイルを編集し、必要なディレクトリファイル (/opt/mqm/java/lib/com.ibm.mqjms.jar など) を追加することにより、JVM が従属アーカイブにアクセスできるようになります。kix_java ディレクトリの内容および設定については、192 ページの「クラスパスおよびライブラリパスのカスタマイズ」を参照してください。

## ライブラリパスの設定

Libpath.appends ファイルも、$KIXSYS/kix_java ディレクトリに配置します。このファイルに次のパスを含める必要があります。

```
MQ-installdir/java/lib
```

*MQ-installdir* は、WebSphere MQ がインストールされているディレクトリです。このパスは、JVM で必要なその他のライブラリにも追加する必要があります。

## MQSERIES 環境変数の設定

MQ-JMS Bridge を使用するには、次のように MQSERIES 環境変数を設定する必要があります。

```
MQ-installdir/java/lib
```

*MQ-installdir* は、WebSphere MQ がインストールされているディレクトリです。これは、MQ-JMS Bridge を使わずに MQ を使用する場合の設定とは異なります (第 12 章を参照)。

# 処理の流れ

各トリガーメッセージに対する制御の流れは、次のとおりです。

1. MQ トリガーメッセージを取得
2. 指定された JMS アプリケーションのインスタンスのロード
3. 指定されたアプリケーションキューへのアクセス
4. アプリケーションキューのメッセージの流れは、次のとおりです。
   a. メッセージ本文の読み取り
   b. アプリケーション MessageListener.onMessage(Message) メソッドの呼び出し

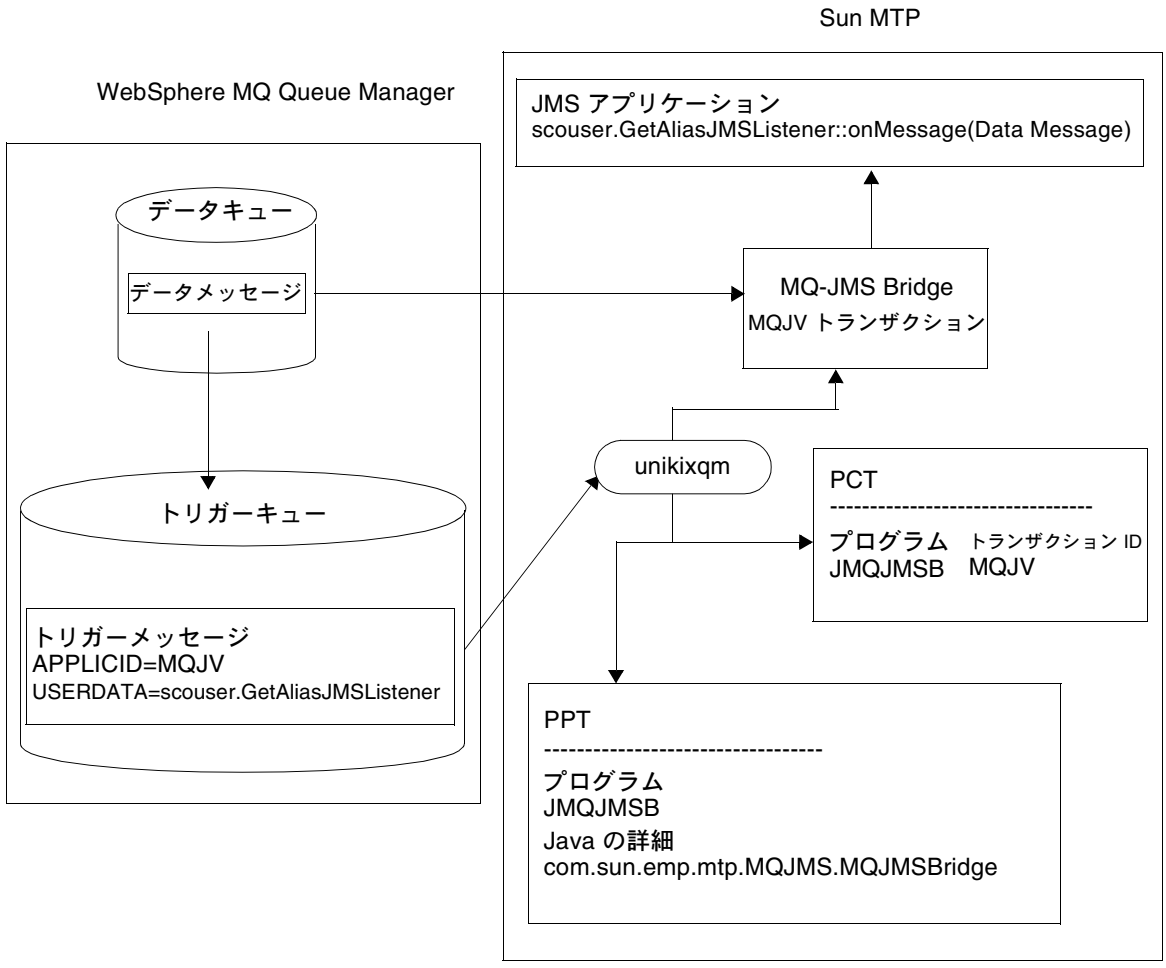


図 13-5　アプリケーションマッピングのない MQ-JMS Bridge 構成

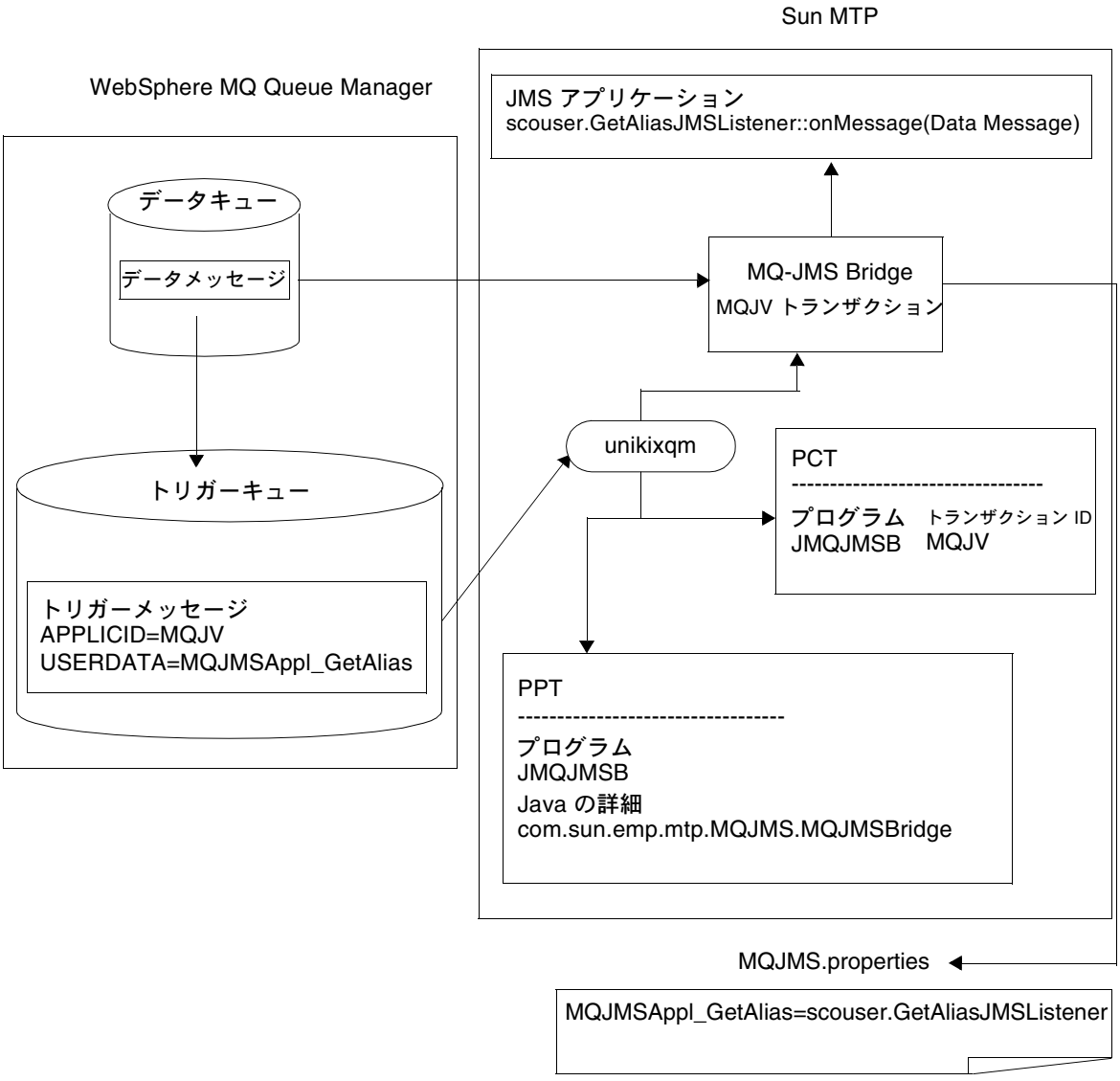


図 13-6　アプリケーションマッピングのある MQ-JMS Bridge 構成

# デバッグ情報

MQJMS.properties ファイルで debug 属性を yes に設定すると、ログ情報が $KIXSYS/unikixmain.dbg ファイルに書き込まれます。イベントエントリのタイプの例は、次の節で示します。

それぞれのイベントの詳細は、IBM WebSphere MQ のマニュアルおよび IBM MQ-JMS のマニュアルを参照してください。

## Triggered transaction start イベント

トランザクションの開始時に、MQJMS.properties 引数が次のようにリストされます。

```
07/25/2001 14:58:08 MQJMSBridge :Triggered transaction start:
Check MQJMS.properties File:
       chan=CHANNEL1
       host=nacelle
       port=1414
       txscope=message
```

## Get Trigger Msg イベント

このエントリは、受け取ったトリガーメッセージの形式を次のようにリストします。

```
07/25/2001 14:58:08 MQJMSBridge :Get Trigger Msg:
STRUCID=TM
VERSION=
QNAME=UNIKIXMQ1
PROCESSNAME=MQJV.TRIGGER.PROC
TRIGGERDATA=Trigdata from UNIKIXMQ1
APPLTYPE=
APPLID=MQJV
ENVDATA=Environ for MQJV
USERDATA=
QMGR-NAME=TESTQM
```

## Target application イベント

このエントリは、アクセスされるターゲットアプリケーションが次のようにリストされます。

```
07/25/2001 14:58:08 MQJMSBridge :Target application:
       scouser.GetAliasJMSListenerExMQJMS
```

## Receiver queue started イベント

次のようなエントリによって、接続された受信側キューが示されます。

```
07/25/2001 14:58:10 MQJMSBridge :Receiver queue started:queue:///UNIKIXMQ1
```

## Process 'onMessage' イベント

onMessage イベントエントリは、ターゲットアプリケーションの onMessage() メソッドに渡される入力メッセージに関する詳細をリストします。それぞれの開始イベントのエントリのあとに、処理単位コミットのエントリが続きます (297 ページの「Unit-of-Work Commit complete イベント」を参照)。

次の例は、バイトメッセージです。

```
07/25/2001 14:58:10 MQJMSBridge :Process 'onMessage':
07/25/2001 14:58:10 MQJMSBridge :
07/25/2001 14:58:10 MQJMSBridge :JMS Message class:jms_bytes
07/25/2001 14:58:10 MQJMSBridge :JMSType:null
07/25/2001 14:58:10 MQJMSBridge :JMSDeliveryMode: 2
07/25/2001 14:58:10 MQJMSBridge :JMSExpiration:   0
07/25/2001 14:58:10 MQJMSBridge :JMSPriority:     0
07/25/2001 14:58:10 MQJMSBridge
:JMSMessageID:ID:414d512054455354514d20202020202016405f3b13100000
07/25/2001 14:58:10 MQJMSBridge :JMSTimestamp:    996098078560
07/25/2001 14:58:10 MQJMSBridge :JMSCorrelationID:null
07/25/2001 14:58:10 MQJMSBridge :JMSDestination:null
07/25/2001 14:58:10 MQJMSBridge :JMSReplyTo:null
07/25/2001 14:58:10 MQJMSBridge :JMSRedelivered:false
07/25/2001 14:58:10 MQJMSBridge :JMS_IBM_Encoding:546
07/25/2001 14:58:10 MQJMSBridge :JMSXAppID:S7.2.0/test/mq/kixmqtst
```

```
07/25/2001 14:58:10 MQJMSBridge :JMS_IBM_Format:
07/25/2001 14:58:10 MQJMSBridge :JMS_IBM_PutApplType:11
07/25/2001 14:58:10 MQJMSBridge :JMS_IBM_MsgType:8
07/25/2001 14:58:10 MQJMSBridge :JMSXUserID:R_Holbert
07/25/2001 14:58:10 MQJMSBridge :JMSXDeliveryCount:1
07/25/2001 14:58:10 MQJMSBridge :JMS_IBM_Character_Set:437
07/25/2001 14:58:10 MQJMSBridge :Integer encoding:2, Floating point encoding 512
07/25/2001 14:58:10 MQJMSBridge :48756d7074792044756d70747920536d69746820
```

次の例は、テキストメッセージです。

```
07/25/2001 14:58:57 MQJMSBridge :Process 'onMessage':
07/25/2001 14:58:57 MQJMSBridge :
07/25/2001 14:58:57 MQJMSBridge :JMS Message class:jms_text
07/25/2001 14:58:57 MQJMSBridge :JMSType:null
07/25/2001 14:58:57 MQJMSBridge :JMSDeliveryMode: 1
07/25/2001 14:58:57 MQJMSBridge :JMSExpiration:   0
07/25/2001 14:58:57 MQJMSBridge :JMSPriority:     0
07/25/2001 14:58:57 MQJMSBridge
:JMSMessageID:ID:414d512054455354514d20202020202016405f3b13400000
07/25/2001 14:58:57 MQJMSBridge :JMSTimestamp:    996098215350
07/25/2001 14:58:57 MQJMSBridge :JMSCorrelationID:null
07/25/2001 14:58:57 MQJMSBridge :JMSDestination:null
07/25/2001 14:58:57 MQJMSBridge :JMSReplyTo:null
07/25/2001 14:58:57 MQJMSBridge :JMSRedelivered:false
07/25/2001 14:58:57 MQJMSBridge :JMSXAppID:amqsputc_nd\300\200\300\200\300\200
\300\200\300\200\300\200\300\200\300\200\300\200\300\200\300\200\300\200\300\200
\300\200\300\200\300\200\300\200
07/25/2001 14:58:57 MQJMSBridge :JMS_IBM_Format:MQSTR
07/25/2001 14:58:57 MQJMSBridge :JMS_IBM_PutApplType:6
07/25/2001 14:58:57 MQJMSBridge :JMS_IBM_MsgType:8
07/25/2001 14:58:57 MQJMSBridge :JMSXUserID:R_Holbert
07/25/2001 14:58:57 MQJMSBridge :JMSXDeliveryCount:1
07/25/2001 14:58:57 MQJMSBridge :Clark Kent
```

## Unit-of-Work Commit complete イベント

このエントリは、ターゲットアプリケーションに代わってコミットが行われたことを次のように示します。

```
07/25/2001 14:58:11 MQJMSBridge :Unit-of-Work Commit complete.
```

### State of Receiver queue is now 'empty' イベント

このエントリは、受信側キューに他のメッセージがないことを次のように示します。

```
07/25/2001 14:58:11 MQJMSBridge :State of Receiver queue is now 'empty'
```

### Close Receiver queue and wrap-up イベント

受信側キューが空なので、MQ-JMS Bridge が入力キューを閉じることを次のように示します。

```
07/25/2001 14:58:11 MQJMSBridge :Close Receiver queue and wrap-up
```

### Wrap-up successful イベント

トリガートランザクションサイクルが完了しました。新しいメッセージが受信側キューに入った場合、WebSphere MQ サービスが新しいトリガーメッセージを作成し、シーケンス全体が再開します。

```
07/25/2001 14:58:11 MQJMSBridge :Wrap-up successful
```

## MQ-JMS Bridge サンプルアプリケーション

MQ-JMS Bridge サンプルアプリケーションは、`$UNIKIX/examples/mq/jms` ディレクトリにあります。`README.txt` ファイルには、領域の設定およびアプリケーションの実行に関する説明が記述されています。

# JMS および COBOL アプリケーションの使用法

MQ-JMS Bridge は、JMS Java アプリケーションとのみ動作します。ただし、COBOL などのその他のアプリケーションも同じアプリケーション領域に存在でき、メッセージをキューに入れたり削除したりできます。参照の共通ポイントは、MQ トリガーキュー自体です。

`$UNIKIX/examples/mq/jms` ディレクトリにある JMS サンプルアプリケーションの 1 バージョンでは、`UNIKIXMQ2` という名前の MQ キューに出力を配置します。そのキューを入力キューとして使用すると、トリガー COBOL アプリケーションが `UNIKIXMQ2` からのデータを処理できます。

## ▼ COBOL プログラムをトリガーする MQ-JMS Bridge プログラムを使用する

1. **コード例 13-3 に示すように WebSphere MQ 構成ファイル (`kixmqtst.mqconfig`) を変更して、`UNIKIXMQ2` キューを定義します。**

   この変更は、メッセージが `UNIKIXMQ2` に配置された場合に、トリガーメッセージを作成すること、トリガープロセスは `KMQ2` トランザクションであることを、MQ サービスに対して指定しています。

**コード例 13-3**　UNIKIXMQ2 トリガーの設定

```
 * Declare a trigger queue and process for UNIKIXMQ2
 INITQ(UNIKIX.TRIGGER.QUEUE)                          +
 TRIGGER                                              +
 TRIGDATA('Trig data from UNIKIXMQ2')                 +
 TRIGTYPE(FIRST)                                      +
 PROCESS(KMQ2.TRIGGER.PROC)

 DEFINE PROCESS('KMQ2.TRIGGER.PROC')  REPLACE     +
        DESCR('Invoke KMQ2 transaction')          +
        ENVRDATA('Some environment for KQM2')     +
        USERDATA('Some data for KQM2')            +
        APPLTYPE(CICS)                            +
        APPLCID('KMQ2')
```

2. 次の例のように、MQ サーバーで runmqsc コマンドを使用して、構成ファイルをキューマネージャーに配置します。次に例を示します。

```
$ runmqsc TESTMQ < config-file
```

TESTMQ はキューマネージャー名、*config-file* は構成ファイル名です。

3. PCT で、COBOL プログラム KIXMQ01 を追加し、これにトランザクション ID KMQ2 を関連付けます。

4. PPT で、COBOL プログラム KIXMQ01 を追加します。

5. COBOL プログラム KIXMQ01.cl2 を $UNIKIX/examples/mq/cobol_mf/progs から $UNIKIX/examples/mq/jms にコピーします。

6. $UNIKIX/test/mq/jms ディレクトリの makefile ファイルを編集して、その COBOL プログラムの変換およびコンパイルを取り込みます。

   この例として、$UNIKIX/examples/mq/cobol_mf ディレクトリの makefile を参照してください。アプリケーション全体を再作成します。

7. 領域を開始し、前と同じようにサンプルアプリケーションを実行します。

   一時記憶域キュー MQTEST01 を参照するには、CEBR トランザクションを使用します。UNIKIXMQ2 に配置されていた出力が、MQTEST01 キューに移っています。

# 第 14 章

# オンラインプログラムのデバッグ

Sun MTP は、コマンドレベルのデバッグのための、独自のデバッグ機能を備えています。また、C 言語、COBOL、および Liant Open PL/I のソースレベルのデバッガもサポートしています。この章では、次の事項について説明します。



デバッグバッチプログラムについての詳細は、第 15 章を参照してください。

## デバッグ機能の使用法

Sun MTP デバッグ機能は、COBOL、C、および PL/I プログラム内の `EXEC CICS` 文を監視します。ローカルシステムまたはトランザクションの経路指定を行った遠隔システムで、トランザクションのデバッグにデバッグ機能を使用できます。

### ▼ デバッグ機能を起動する

1. **領域を開始します。**
2. **ブランクのトランザクション画面で、`CEDF` トランザクションを入力します。**

   `CEDF` にオプションを指定して、COBOL プログラムをデバッグできます。詳細は、302 ページの「`CEDF` オプション」を参照してください。
3. **ブレークポイントの設定画面 (図 14-1) が表示されたら、次のデバッグオプションを選択します。**

   詳細は、303 ページの「デバッグモードを選択する」を参照してください。

4. **Enter キーを押します。**

   オプションを選択すると、その端末で開始したすべてのトランザクションに対して有効になります。

5. **トランザクションを入力して、デバッグを開始します。**

   詳細は、307 ページの「デバッグセッション中にブレークポイントを設定する」を参照してください。

## CEDF オプション

**注 –** C または PL/I 環境では、CEDF をオプションとともに使用しないでください。CEDF トランザクションのみを入力します。

CEDF トランザクションの形式は、次のとおりです。

```
CEDF  [trmid|sysid][,ON|,OFF|,COBOL|,RCOBOL]

CEDF[,ON|,OFF|,COBOL|,RCOBOL]
```

| オプション | 説明 |
|---|---|
| *trmid* | 領域にログインした端末を示す 4 文字の端末、ローカル、または遠隔の識別子。識別子は EIBTERMID です。<br>この端末から送信されたトランザクションは、デバッグ機能が実行されるウィンドウでデバッグできます。<br>COBOL、C、および CodeWatch デバッガは、インターシステム通信接続 (TCP/IP や SNA) を介してこのオプションとは動作しません。 |
| *sysid* | TCT のシステムエントリ画面の「SysId」フィールドと一致する 4 文字の識別子。*sysid* を指定すると、その遠隔システムから受信する LU6.2 接続要求がデバッグされます。 |
| ,ON | コマンドレベルのデバッグを初期化します。CEDF,ON とのみ入力すると、デバッグは CEDF トランザクションを呼び出した端末をデフォルトとします。 |
| ,OFF | ,OFF: すべてのデバッグオプションを破棄します。 |
| ,COBOL | ,COBOL: COBOL デバッガを初期化します。詳細は、311 ページの「COBOL デバッガの使用」を参照してください。 |
| ,RCOBOL | ,RCOBOL: 遠隔 COBOL デバッガを初期化します。詳細は、312 ページの「遠隔 COBOL デバッガの使用法」を参照してください。 |

**注 –** CEDF コマンドで *trmid* または *sysid* を使用する場合、CEDF と *trmid* または *sysid* との間にスペースを入力します。それ以外の場合は、CEDF とコンマの間にスペースを入力しないでください。

CEDF では、1 つまたは複数の端末セッションをデバッグできます。

遠隔端末の端末定義が領域に存在する場合、インバウンドトランザクション経路で送られたトランザクションをデバッグするために CEDF *trmid* を使用できます。ISC シップ可能な端末については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

すべての他のインバウンド ISC 要求に対しては、CEDF *sysid* を使用します。端末で CEDF を呼び出したあとに、その接続で他の接続を受信した場合、これはデバッグ用のキューに入りません。ただし、同じ端末に対して複数のデバッグトランザクションが発生した場合、それらはキューに入ります。

START 要求を実行するデバッグトランザクション、実行トランザクションが初めに処理される場合は、開始されたトランザクションが処理されます。

## ▼ デバッグモードを選択する

1. **ブランクのトランザクション画面で、オプションなしの CEDF を入力します。**
2. **ブレークポイントの設定画面 (図 14-1) が表示されたら、次のデバッグモードを選択します。**
   - コマンドレベルのデバッグ (ブレークポイント画面上部の 5 つのオプション)

     Sun MTP は、設定に基づいて EXEC CICS コマンドをデバッグします。COBOL、C、PL/I ソースコードはデバッグされません。
   - ソースレベルのデバッグ
     - COBOL デバッガを初期化
     - COBOL 遠隔デバッガを初期化
     - C ソースデバッガを起動
     - CodeWatch (PL/I ソースデバッガ) を起動
3. **次の状態のときに、DISPLAY フィールドに値が入力されます。**
   - TN3270 または IBM 3270 端末からの COBOL デバッガを使用している場合
   - ローカル端末クライアントから COBOL デバッガを使用しており、別のウィンドウでデバッグを実行する場合
   - C ソースデバッガを使用する場合
   - CodeWatch PL/I デバッガを使用する場合

**注 –** DISPLAY フィールドは、ローカル端末クライアントを開始したシェルの DISPLAY 変数から自動的に入力されます。シェルで $DISPLAY が設定されていない場合、DISPLAY フィールドは空白になります。

DISPLAY フィールドについては、表 14-1 を参照してください。

4. **Enter キーを押します。**

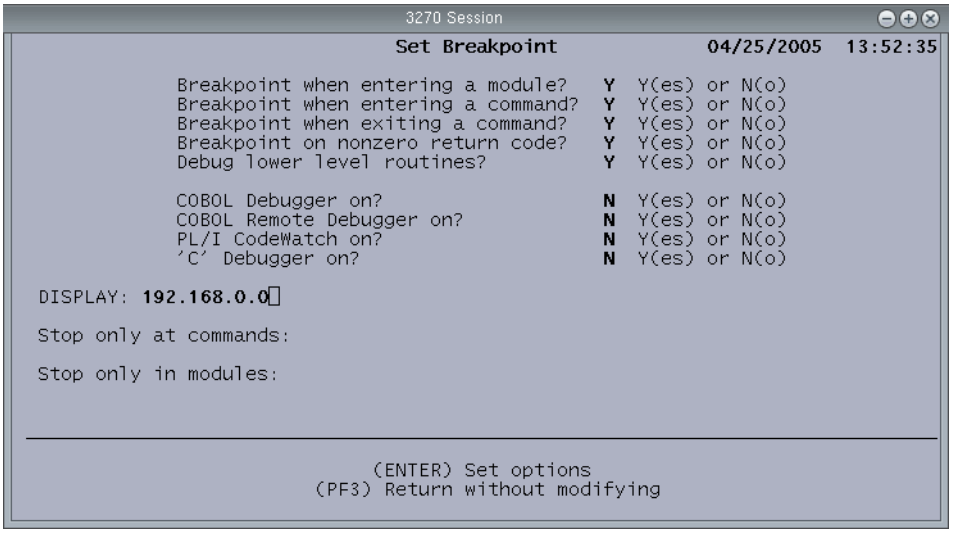


**図 14-1** デバッグ機能—Set Breakpoint 画面

ブレークポイントの設定画面のオプションについては、次の表を参照してください。

**表 14-1** ブレークポイントの設定画面のオプション

| オプション | 説明 | 有効値 |
|---|---|---|
| Breakpoint when entering a module | 新しいオブジェクトモジュールをロードするとき、プログラムの実行を停止します。 | Y: ブレークポイントを有効にします<br>N: ブレークポイントを無効にします |
| Breakpoint when entering a command | CICS コマンドが実行される直前に、実行を停止してデバッグ機能メインメニューに戻ります。 | Y: ブレークポイントを有効にします<br>N: ブレークポイントを無効にします |
| Breakpoint when exiting a command | CICS コマンドが終了するときに、実行を停止してデバッグ機能メインメニューに戻ります。 | Y: ブレークポイントを有効にします<br>N: ブレークポイントを無効にします |

表 14-1　ブレークポイントの設定画面のオプション (続き)

| オプション | 説明 | 有効値 |
|---|---|---|
| Breakpoint on nonzero return code | CICS コマンドからのリターンコードがゼロ以外 (エラー条件を示す) の場合、現在のプログラムの実行を停止します。 | Y: ブレークポイントを有効にします<br>N: ブレークポイントを無効にします |
| Debug lower level routines | EXEC CICS LINK コマンドによって入力された CICS プログラムをデバッグするかどうかを指定します。 | Y: LINK コマンドで入力されたプログラムのブレークポイント条件が 1 つでも true の場合、ブレークポイント処理プログラムに入力します。<br>N: ブレークポイント処理プログラムは、LINK コマンドで入力されたプログラムのブレークポイント条件を無視します。 |
| COBOL Debugger on | COBOL トランザクションの実行時に、COBOL デバッガを開始します。デバッガの使用法については、COBOL ベンダーのマニュアルを参照してください。<br>このオプションは、トランザクションの開始時にのみセットまたはリセットされます。トランザクションの途中でデバッガをオフにするには、COBOL ベンダの高速オプションを使用して、プログラムを完了するか、デバッグセッションをキャンセル (強制的に中止) します。<br>このオプションが Y に設定されている場合、他のデバッグオプションをオフにします。 | Y: COBOL デバッガの開始<br>N: COBOL デバッガの停止 |
| COBOL Remote Debugger on | このオプションは、Server Express のみでサポートされます。<br>COBOL トランザクションの開始時に、遠隔 COBOL デバッガを開始します。Remote Animator で利用できるオプションについては、Micro Focus Remote Animator のマニュアルを参照してください。遠隔 COBOL デバッガを使用してデバッグするには、312 ページの「遠隔 COBOL デバッガの使用法」を参照してください。<br>このオプションが Y に設定されている場合、他のデバッグオプションをオフにします。 | Y: 遠隔 COBOL デバッガの開始<br>N: 遠隔 COBOL デバッガの停止 |

表 14-1　ブレークポイントの設定画面のオプション (続き)

| オプション | 説明 | 有効値 |
|---|---|---|
| PL/I CodeWatch on | Liant Open PL/I トランザクションの実行時に、PL/I CodeWatch デバッガを開始します。315 ページの「PL/I プログラムのデバッグ」および Liant PL/I の『Liant PL/I CodeWatch Reference Manual』を参照してください。<br>PL/I CodeWatch が Y に設定されている場合、他のデバッグオプションをオフにします。 | Y: CodeWatch を有効にする<br>N: CodeWatch を無効にする |
| C Debugger on | C トランザクションの実行時に、C ソースデバッガを開始します。詳細は、314 ページの「C 言語プログラムのデバッグ」を参照してください。<br>C デバッガが Y に設定されている場合、他のデバッグオプションをオフにします。 | Y: C デバッガを有効にする<br>N: C デバッガを無効にする |
| DISPLAY | デバッガの画面が表示されることを示します。<br>• ローカルクライアント端末から COBOL デバッガを使用する場合は、このフィールドが空白になります。この例では、デバッガは、ローカルクライアントウィンドウに表示されます。異なるウィンドウを表示するには、このフィールドに値を入力します。<br>• TN3270 クライアントまたは IBM 3270 デバイスで実行する COBOL デバッガには、このフィールドに値が必要です。<br>• 任意の端末で実行される CodeWatch および C デバッガには、このフィールドに値が必要です。 | 有効な値は次のようになります。<br>`myhost:32.0`<br>`10.255.255.255:32.0` |
| Stop only at commands | システムがブレークポイントを制御するための CICS コマンド (`SEND MAP` など) を 3 つまで指定します。画面の上半分で 1 つ以上のオプションを有効にしている場合にのみ適用されます。<br>コマンドが何も入力されていない場合、画面の上半分で設定した 5 つのオプションに基づいて、プログラムの実行はすべてのコマンドで停止する。1 つ以上のコマンドが設定されている場合、そのコマンドだけがブレークポイント処理プログラムをトリガーします。 | CICS コマンド |

表 14-1　ブレークポイントの設定画面のオプション (続き)

| オプション | 説明 | 有効値 |
|---|---|---|
| Stop only in modules | システムがブレークポイント制御するためのプログラムを 5 つまで指定します。画面の上半分で 1 つ以上のオプションを有効にしている場合にのみ適用されます。<br>5 つのすべてのプログラム名がブランクの場合、画面の上半分で設定された 5 つのオプションに基づいて、プログラムはすべてのプログラム内のコマンドで実行を停止します。1 つ以上のプログラム名が入力されている場合、それらのプログラム内のコマンドのみがブレークポイント処理プログラムをトリガーします。<br>ソースデバッガを使用している場合は、このオプションは無視されます。 | プログラム名 |

## ▼ デバッグセッション中にブレークポイントを設定する

1. **トランザクションの実行前に** `CEDF` **を入力するか、前のブレークポイントの要求によってトランザクションが停止したときに PF10 キーを押します。**

   ブレークポイントの設定画面が表示されます。

2. **ブレークポイントの値を入力し、Enter キーを押します。**

   システムは、トランザクション識別子を入力するためのブランク画面を返すか、あるいは指定された条件の 1 つが発生したときには、デバッグ機能のメイン画面を表示してプログラムを停止します。

## デバッグ処理プログラムメイン画面

デバッグオプションを設定してプログラム名またはトランザクションを入力すると、デバッグ処理プログラムメイン画面が表示されます。デバッグ処理プログラムメイン画面には、次の 3 つの領域があります。

- EXEC インタフェースブロック (EIB) 表示領域。EIB 領域の各フィールドについては、132 ページの「サポートされている EIB フィールド」を参照してください。
- CICS コマンドおよび主記憶域アドレス領域。
- PF キー解釈領域。

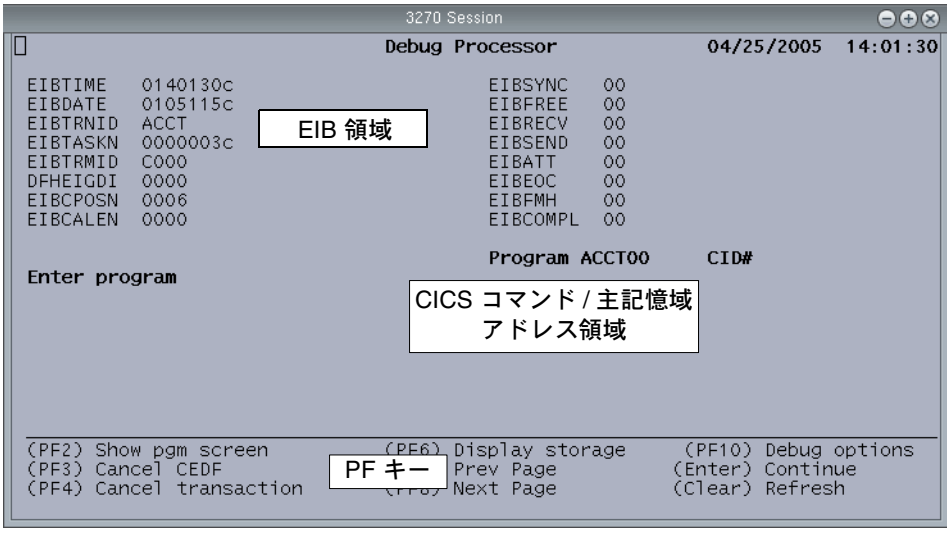


**図 14-2** Debug Processor—メイン画面

デバッグ処理プログラムメイン画面では、次のファンクションキーが使用できます。

| ファンクションキー | 動作 |
|---|---|
| PF2 | その時点でユーザーに表示される、プログラムの端末画面を表示します。アテンションキー (PF2 キーなど) を押すと、デバッグ処理プログラム画面に戻ります。 |
| PF3 | CEDF を取り消して、プログラムの実行を終了します。確認プロンプトは表示されません。 |
| PF4 | プログラムのキャンセルを確認するための「Cancel Transaction」画面を表示します。Enter キーを押すと、取り消されます。PF3 キーを押すと、デバッグ処理プログラム画面に戻ります。 |
| PF6 | 主記憶域画面を表示します。詳細は、309 ページの「主記憶域を表示する」を参照してください。 |
| PF7 | EIB 表示領域の前のページを表示します。 |
| PF8 | EIB 表示領域の次のページを表示します。 |
| PF10 | ブレークポイントの設定画面を表示します。ブレークポイントについては、表 14-1 を参照してください。 |
| Clear | デバッグ処理プログラム画面を再表示します。 |
| Enter | 次のブレークポイントが検出されるまで、プログラム実行を継続します。ブレークポイントに達するか、現在のトランザクションが終了するまで、デバッガはプログラム実行を継続します。 |

## ▼ 主記憶域を表示する

- **デバッガが `EXEC CICS` コマンドで停止したときに、デバッグ処理プログラム画面で PF6 キーを押します。**

  256 バイトページの 16 進数表現および文字表現が表示されます。デフォルトでは、現在のプログラムの開始アドレスから記憶域が表示されます。

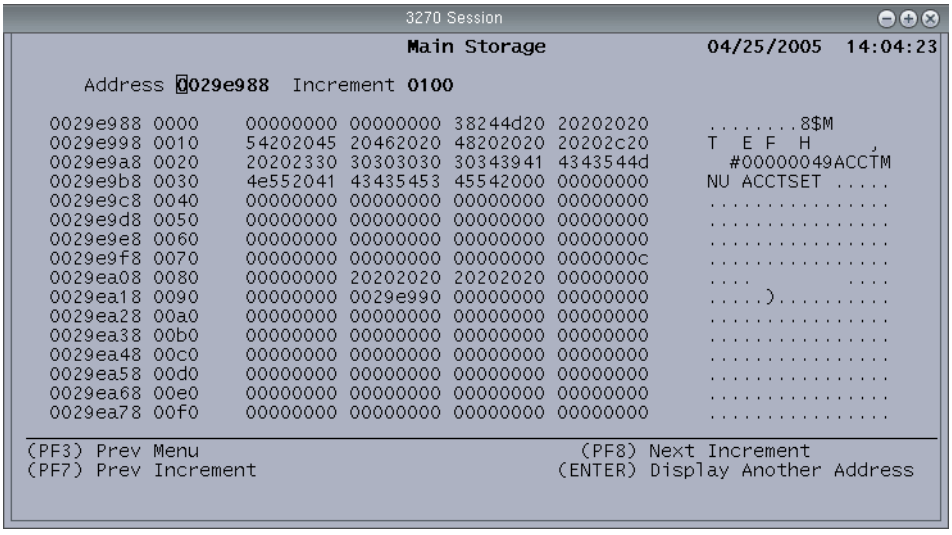


**図 14-3** Debug Processor—Main Storage 画面

次のフィールドは変更できます。

Address

表示領域の最初のバイトのアドレス。表示される記憶領域の範囲が有効なファンクションキー (PF7、PF8、または Enter) によって変更された場合、このフィールドは自動的に更新されます。

Increment

前のページまたは次のページを表示する際の変更量を表す、16 進数形式でのバイト数。増分値が 16 進数の 100 よりも大きく設定されていると、後続の表示では中間の一部のバイトが省略されます。

この画面では、次のファンクションキーが使用できます。

| ファンクションキー | 動作 |
|---|---|
| PF3 | デバッグ処理プログラムメイン画面に戻ります。 |
| PF7 | 「Increment」フィールドで指定された増分値のみページを戻ります。 |
| PF8 | 「Increment」フィールドで指定された増分値のみページを進みます。 |
| Enter | 「Address」および「Increment」フィールドに対して行われた変更を適用します。「Address」を変更すると、メモリーバッファーの領域が変更されて指定のアドレスが表示されます。 |

### ▼ デバッグをオフにする

- **次のいずれかの方法を使用します。**
  - ブランクのトランザクション画面で、`CEDF, OFF` トランザクションを入力します。
  - デバッグ処理プログラムメイン画面 (図 14-2) で PF3 キーを押します。
- **または**

1. **ブランクのトランザクション画面で、`CEDF` を入力します。**
2. **ブレークポイントの設定画面が表示されたら、各オプションを `N` に設定して無効にします。**
3. **Enter キーを押して変更を適用し、ブランクのトランザクション画面に戻ります。**

# COBOL プログラムのデバッグ

COBOL ソースレベルのデバッガは、Sun MTP環境に統合されているので、Sun MTP内でプログラムをデバッグできます。Server Express デバッガは、Animator で、ACUCOBOL-GT デバッガは Runtime Debugger という名前です。デバッガの使用法については、ベンダのマニュアルを参照してください。

トランザクション処理が COBOL サポートなしで構築されている領域で COBOL デバッガを起動しようとすると、次のメッセージが表示されます。

```
COBOL Debugger cannot be invoked on a non-COBOL system
```

## COBOL デバッガの使用

COBOL デバッガは、X 端末および ASCII 文字デバイス上で動作します。

ブレークポイントの設定画面にある DISPLAY フィールドに値を入力するか、DISPLAY 環境変数をローカル端末セッションで設定すると、ディスプレイデバイスで新しいウィンドウが起動されます。ユーザーは、別のウィンドウでソースと CICS 画面を表示できます。

ディスプレイが設定されていない場合は、ローカル端末セッションが使用され、COBOL ソースと CICS 画面が表示されます。トランザクションデバッグセッション中は、有効な画面が表示されます。EXEC CICS 文が実行されている場合は、Sun MTP 画面が表示され、COBOL コードが実行されている場合は、COBOL デバッガ画面が表示されます。この機能は、Server Express ランタイムを使ってのみサポートされます。Micro Focus Animator オプションの F2 を使って、デバッグ画面と CICS 画面を切り替えることができます。

## ▼ COBOL Debugger を使用してプログラムをデバッグする

1. **領域を開始します。**
2. **端末を使用して領域に接続します。**
3. **次のいずれかの方法を使用して、COBOL デバッガを開始します。**
   - CEDF,COBOL。詳細は、302 ページの「CEDF オプション」を参照してください。
   - オプションなしの CEDF。ブレークポイントの設定画面で、オプション COBOL Debugger ON を Y に設定します。デバッガを独自のウィンドウに表示する場合は、DISPLAY フィールドにも値を入力します。
4. **Enter キーを押します。**

   端末がデバッグモードになります。
5. **トランザクションを入力して、デバッグを開始します。**
6. **プログラムをデバッグします。**
7. **終了したら、デバッガをオフにします。**

   詳細は、310 ページの「デバッグをオフにする」を参照してください。

# 遠隔 COBOL デバッガの使用法

Sun MTPの遠隔 COBOL デバッグ機能は、Micro Focus 環境でのみサポートされています。Net Express IDE (統合開発環境) と統合することによって、開発者の生産性を向上させることができます。これにより、ネットワーク上の端末でサブミットされるトランザクションを通じて Net Express 環境から Sun MTP COBOL アプリケーションをデバッグできます。

Net Express は、Microsoft Windows オペレーティングシステムのパーソナルコンピュータで動作します。Net Express IDE および遠隔アニメーションの詳細は、Micro Focus のマニュアルを参照してください。

## ▼ Remote COBOL Debugger を使用してプログラムをデバッグする

1. **領域の設定ファイルまたはシェルプロンプトで、次のように環境変数を設定します。**

   a. `KIXREMANIMPORT` **に必要なポート番号を設定します。**

      ポート番号の値は 1025 ～ 65,535 です。

   b. `KIXREMANIMTOUT` **にタイムアウト値 (秒) を設定します。**

      この値は、エラーが返されるまで、領域がセッション間の Animator サーバーを待機する時間です。デフォルトは 5 秒です。

2. **領域設定ファイルで、次の事項を確認します。**

   a. `COBDIR` **環境変数が、Server Express のインストールされているディレクトリに設定されています。**

   b. `$COBDIR/lib` **が、**`LD_LIBRARY_PATH` **環境変数に取り込まれています。**

   c. `$COBDIR/bin` **が、**`PATH` **環境変数に取り込まれています。**

3. **ホストシステムとパーソナルコンピュータで、COBOL ソースファイルが同じであることを確認します。**

   a. Net Express **で、**`.int` **および** `.idy` **ファイルを作成するプロジェクトを構築します。**

   b. **その Net Express プロジェクトが配置されているパス名をメモします。**

      このパス名は、手順 9 で必要です。

4. **ホストシステムの新しいウィンドウで、次のコマンドを入力してデバッグサーバープロセスを開始します。**

```
$ animserv32 AUTO PORT=nnnn
```

*nnnn* は、KIXREMANIMPORT 環境変数で設定されている値です。

---

**注** – このプロセスは、フォアグラウンドで実行する必要があります。

---

5. **領域を開始します。**
6. **TN3270 エミュレータソフトウェア (Sun MTP J3270 など) を使用して、パーソナルコンピュータから領域に接続します。**
7. **エミュレータウィンドウで、次のいずれかの方法によってホストシステム上で遠隔デバッグを開始します。**
   - CEDF,RCOBOL
   - CEDF *termid*,RCOBOL
   - CEDF にオプションを設定しない。ブレークポイントの設定画面の「COBOL Remote Debugger On 」オプションを Y に設定
8. **遠隔デバッガが有効化されたことを示すメッセージが表示されたら、端末識別子、ホスト名、およびポート番号をメモします。**
9. **パーソナルコンピュータで Remote Animator を開始します。**
   - Sun MTP J3270 端末エミュレータを使用しないで領域に接続する場合、パーソナルコンピュータの DOS ウィンドウで次のコマンドを使用します。このコマンドは 1 行です。

```
C:\> MFNETX /DEBUGSERVER /DEBUGPROJ:Net-Express-project-path /DEBUGID:termid
/DEBUGMACHINE:machine-name /DEBUGPORT:port-number
```

   - Sun MTP J3270 端末エミュレータを使用して領域に接続する場合、このコマンドは自動的に実行されます。
10. **エミュレータウィンドウのブランクのトランザクション画面で、トランザクションを入力してデバッグを開始します。**
11. **Remote Animator ウィンドウが表示されたら、オプションを使用してデバッグを実行できます。**
12. **デバッグを終了したら、遠隔デバッグをオフにします。**

310 ページの「デバッグをオフにする」を参照してください。

# C 言語プログラムのデバッグ

Sun MTP は、DBX シンボリックデバッガにインタフェースを提供することにより、C 言語プログラムのデバッグをサポートします。デバッガの使用法については、デバッガのマニュアルを参照してください。

## ▼ プログラムをデバッグする

1. **コンパイルコマンドで `-g` オプションを指定してプログラムをコンパイルします。**
2. **共有オブジェクトを作成します。**

   C 共有オブジェクトに関する情報については、180 ページの「共有オブジェクトの構築」を参照してください。
3. **PCT および PPT で C トランザクションを定義します。**
4. **デバッガ初期設定ファイルを作成し、ホームディレクトリに保存します。**

   初期設定ファイルについては、デバッガのマニュアルを参照してください。次の例は、Solaris™ DBX デバッガで必要な初期設定ファイル `.dbxrc` のコードです。

```
use /.../.../src/CICS_structures
```

5. **この領域を起動します。**
6. **端末クライアントを使用して領域に接続します。**
7. **ブランクのトランザクション画面で、`CEDF` トランザクションを入力します。**
8. **ブレークポイントの設定画面で、オプション「C Debugger On」を `Y`、その他のすべてのオプションを `N` に設定します。**
9. **DISPLAY フィールドに、端末クライアントの値を入力します。**
10. **Enter キーを押します。**

    端末がデバッグモードになります。
11. **デバッグする C トランザクションを入力します。**

    デバッガが自動的に開始し、トランザクションサーバーに接続します。

12. **デバッガウィンドウが表示されたら、デバッガがトランザクションサーバーに接続したときにソースコードを表示するデバッガコマンドを入力します。**

    このコマンドは、次のようにコーディングします。

```
...
while(1) {} i = i;
return;
```

    実際の停止/接続ポイントは、`while(1)` 命令です。

13. **デバッグウィンドウでコマンドを入力して C アプリケーションコードのブレークポイントを設定し、デバッグを継続します。**

    コマンドの詳細は、デバッガのマニュアルを参照してください。

    `EXEC CICS RETURN TRANSID` または `EXEC CICS RETURN` (あるいは論理的にこれらと同等なコマンド) が検出された場合、トランザクションサーバーがトランザクションプロセスを継続するとき、現在のデバッグセッションは自動的に非インスタンス化され、新しいデバッグセッションがインスタンス化されます。デバッガは、トランザクションサーバー間のプロセス識別子 (PID) の境界を超えてアプリケーションコードを追跡できないので、この非インスタンス化/再インスタンス化プロシージャーが必要になります。

14. **終了したら、デバッグをオフにします。**

    310 ページの「デバッグをオフにする」を参照してください。

# PL/I プログラムのデバッグ

CodeWatch は、Liant Open PL/I ソースデバッガです。このデバッガでは、マルチユーザー、マルチトランザクションサーバーの環境で、PL/I アプリケーションをデバッグできます。

マルチユーザー環境では、一部のユーザーだけが CodeWatch を使用している場合があります。CodeWatch の同時ユーザー数は、システム上に存在できる CodeWatch プロセスの有効なコピー数に制限されます。CodeWatch を実行していない同時ユーザーは、制限を受けずにトランザクションを実行できます。

CodeWatch モードで実行している場合、`STARTCW` プログラム (`startcw.so` 共有オブジェクト) が、トランザクションの開始処理を行うすべての PL/I アプリケーションプログラムの最初のモジュールになります。このプログラムが CodeWatch プロセスをトリガーし、接続を待機します。CodeWatch ウィンドウが表示されたら、コマンドまたはブレークポイントを入力するか、`C` と入力してデバッグ処理を続行するだけです。

**注 –** デバッガを使用するには、使用する unikixtran 実行可能ファイルを含むディレクトリが、PATH 変数の $UNIKIX/bin の前に置かれていることを確認してください。たとえば、$UNIKIX/local/bin は、unikixtran 実行可能ファイルを含む場合、これが検索パスの最初のエントリである必要があります。

CodeWatch は、デバッグ用に、グラフィカルユーザーインタフェース (GUI) を提供します。GUI を使用する場合は、領域を開始する前に、CWSTARTD_COMMAND 変数を有効なコマンドに設定する必要があります。この変数の設定については、『CodeWatch Reference Manual』を参照してください。

## ▼ CodeWatch を設定する

1. **CodeWatch 環境でデバッグできるように、プログラムコードを CodeWatch に対応させます。**

   これについての説明は、Liant の『Open PL/I Language Reference Manual』、『Open PL/I User's Guide』、および『CodeWatch Reference Manual』を参照してください。

2. **環境変数 CODEWATCH_STBPATH および CODEWATCH_SRCPATH を設定し、検索パスに $UNIKIX/lib があることを確認します。**

   これらの環境変数の詳細は、『CodeWatch Reference Manual』を参照してください。

3. **CodeWatch GUI を使用する場合は、CWSTARTD_COMMAND 変数を設定します。**

4. **CodeWatch 設定ファイルを作成します。**

   このファイルのコマンドは、CodeWatch デバッグモードで領域を配置するために使用されます。CodeWatch の設定ファイルには、次の startcw.cwf ファイルに示されているコマンドを含めます。

```
shlib /aaaa/bbbb/..../lib/startcw.so
env STARTCW
b innerloop
c
goto loopexit
```

**注 –** この例の最初の行は、`startcw.so` オブジェクトのフルパス名を提供する必要があることを示しています。

共有ライブラリ `startcw.so` は、`$UNIKIX/lib` ディレクトリにあり、それに対するエントリも PPT にあります。これらの 2 つの項目は、Sun MTP ソフトウェアを構築すると自動的に設定されます。

5. `CODEWATCH_INIT` **環境変数を、CodeWatch コマンドを含むファイルに設定し、CodeWatch モードに領域を配置します。**

   次に例を示します。

```
CODEWATCH_INIT=$KIXSYS/startcw.cwf
```

## ▼ プログラムをデバッグする

1. **この領域を起動します。**
2. **端末を使用して領域に接続します。**
3. **ブランクのトランザクション画面で、`CEDF` トランザクションを入力します。**
4. **ブレークポイントの設定画面で、オプション「PL/I CodeWatch On」を `Y`、その他のすべてのオプションを `N` に設定します。**
5. **DISPLAY フィールドに、端末の値を入力します。**
6. **Enter キーを押します。**

   端末がデバッグモードになります。

7. **デバッグするトランザクションを入力します。**

   PL/I アプリケーションコードが処理を開始する直前に、CodeWatch プロセスが自動的に呼び出され、316 ページの「CodeWatch を設定する」で示されているように、現在のトランザクションサーバーに接続されます。

8. **CodeWatch 環境が実行されたあとでプログラムをデバッグするには、次のいずれかの方法を使用します。**
   - CodeWatch のデバッグコマンドを入力します。『Liant CodeWatch Reference Manual』を参照してください。

■ デバッグ環境から実行できるコマンドファイルを作成します。

たとえば、ACCT Primer トランザクションプログラムを CodeWatch 環境でデバッグするには、CodeWatch のプロンプトで次のコマンドを入力します。

```
read acct00.cwf
```

このコマンドにより、CodeWatch が次の内容を含む acct00.cwf ファイルから、デバッグコマンドを読み取るようになります。

```
shlib /pkgs/mtp/MTP8.1.0/primer/pl1/progs/acct00.so
env ACCT00
b ACCT00%ENTRY
c
p12
```

---

**注 –** この例では、acct00.so 共有オブジェクトのフルパス名は、PPT で定義されているパスです。

---

9. **トランザクションが終了すると、CodeWatch は自動的に終了し、CodeWatch ウィンドウが閉じます。**

10. **同じトランザクションサーバーまたは別のトランザクションサーバーが次のトランザクションを取得すると、新しい CodeWatch プロセスが開始されます。**

    このシーケンスは、デバッグを終了するまで続行されます。

11. **終了したら、デバッグモードをオフにします。**

    詳細は、310 ページの「デバッグをオフにする」を参照してください。

# 第15章

# バッチ処理

Sun MTP では、次の 2 つのバッチ環境をサポートします。

- **標準のバッチ**: COBOL プログラムのバッチ処理または実行可能コマンドファイル (シェルスクリプト) のバッチ処理を制御します。

  C 言語のバッチプログラムは、C-ISAM インタフェースを使用してサポートされます。詳細は、332 ページの「C 言語のバッチプログラムの実行」を参照してください。

  PL/I バッチプログラムは、Sun MBMの使用をサポートしています。詳細は、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

- Sun MBM: 制御された環境でバッチジョブを実行します。詳細については、Sun MBM のマニュアルを参照してください。

---

**注 –** 「Sun MBM以外」と表記されている機能は、標準のバッチ環境でだけ有効です。

---

この章の内容は、次のとおりです。

# バッチ環境の設定

バッチ処理に割り当てられているリソースは限定されています。Sun MTP は、フォアグラウンドトランザクション (端末で開始されるオンライントランザクションなど) を優先します。リソースに余裕がある場合、バックグラウンドトランザクションを開始します。バックグラウンドトランザクションには、プリンタで開始されるレポートプログラム、一時データキューがトリガーレベルに達したときに開始されるトランザクションなどがあります。最後に、さらにリソースに余裕がある場合、バッチプログラムを開始します。

バッチプログラムを実行する前に、次のようにバッチ環境を設定します。

- KIXBTCH 環境変数を定義します。この環境変数で、実行待ちのバッチプログラムが置かれているディレクトリを定義します。この手順は、Sun MBM 以外の環境でのみ有効です。
- VSAM 構成テーブル (VCT) でバッチパラメータを設定します。詳細は、322 ページの「VCT バッチフィールドの設定」を参照してください。

## KIXBTCH 環境変数の設定

領域の設定ファイルで KIXBTCH 環境変数を設定します。複数の $KIXBTCH ディレクトリが存在する場合、それらをコロンで区切ります。$KIXBTCH によって指定された 1 つまたは複数のディレクトリは、$KIXSYS ディレクトリと同じファイルシステム上に存在する必要があります。

これらのディレクトリに対して、設定された間隔で COBOL プログラムまたはシェルスクリプトの検索が順番に行われます。バッチ検索間隔の設定については、322 ページの「VCT バッチフィールドの設定」を参照してください。

---

**注 –** $KIXBTCH は、Sun MBM 以外の環境でだけ使用します。

---

## 複数の $KIXBTCH 環境のサポート

複数の $KIXBTCH ディレクトリからバッチジョブを実行していて、各ディレクトリに異なる検索間隔を設定する必要がある場合、次の形式で領域設定ファイルに $KIXBTCH を設定します。

```
export KIXBTCH=directory1[starttime1-endtime1] [:directory2[starttime2-endtime2]]:...
```

| 引数 | 説明 |
|---|---|
| *directory1, directory2, ...* | バッチプログラムまたはシェルスクリプトがあるディレクトリ。バッチプロセスを有効にするには、最低 1 つのディレクトリが存在する必要があります。すべてのバッチディレクトリは、$KIXSYS ディレクトリと同じファイルシステム上に存在する必要があります。 |
| *starttime1, endtime1, ...* | 24 時間表示の時刻 (*hhmm*)。ディレクトリを検索する時刻を制御します。すべての時刻は任意指定ですが、開始時刻は終了時刻より前である必要があります。現在時刻が開始時刻と終了時刻の間でない場合、そのディレクトリはスキップされます。<br>時刻が 1 つの場合は開始時刻とみなされます。終了時刻は 2400 がデフォルトです。深夜 0 時をまたがった検索間隔は許可されていません。 |

**例**: バッチ環境で、2 つの $KIXBTCH ディレクトリを必要としています。 1 つ目は、在庫管理の領域で実行しているローカルプログラムのディレクトリです。2 つ目は、CICS のメインフレームシステムから夜間ダウンロードされる給与計算タイムシートプログラムを受け取るディレクトリです。最高のパフォーマンスを得るために、2 分ごとに検索されるローカルディレクトリ /inv/batch および夜間のダウンロード後に一度だけ検索されるダウンロードディレクトリ /pay/times を持つことにします。領域設定ファイルで $KIXBTCH を次のように設定します。

```
export KIXBTCH=/inv/batch:/pay/times[0300-0400]
```

# VCT バッチフィールドの設定

VCT の複数のフィールドで、バッチプログラムの実行方法を制御します。テーブルマネージャーの標準テーブルメニューから VCT を開きます。

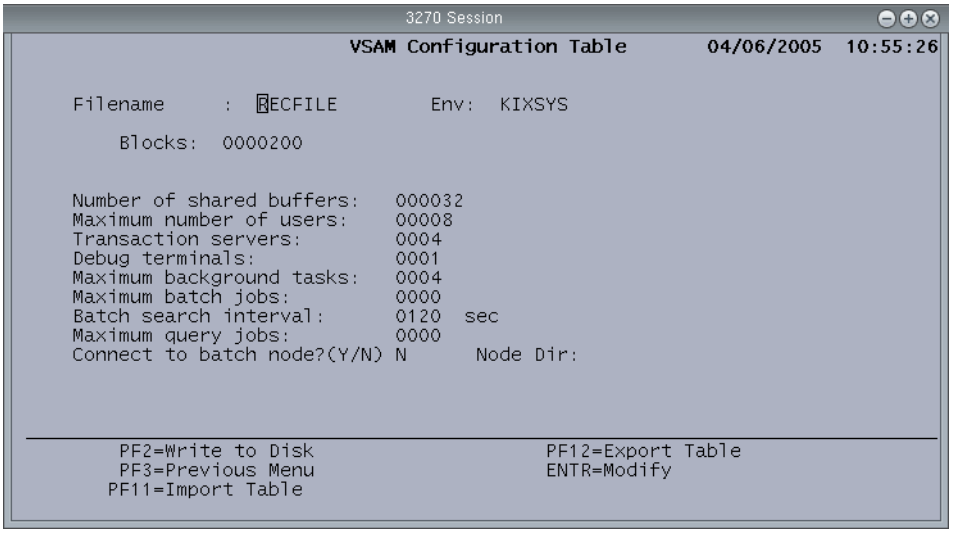


**図 15-1** VSAM Configuration Table

次の各フィールドにより、Sun MTP が実行できるフォアグラウンドトランザクション、バックグラウンドトランザクション、およびバッチプログラムの数、さらに、バッチ検索間隔が制御されます。

| フィールド | 説明 |
|---|---|
| Transaction servers | 領域が利用できるトランザクションサーバーの総数。トランザクションまたはバッチプログラムの 1 つを実行するには 1 つのトランザクションサーバーが必要なので、同時に実行できるトランザクションおよびプログラムの最大数もこのフィールドで定義されます。<br>このフィールドは、トランザクションサーバーが使用可能かどうかを判定するために使用されます。 |
| Maximum background tasks | 並行して実行できるバックグラウンドトランザクションおよびプログラムの最大数を設定します。<br>このフィールドにより、優先順位が低いタスクに割り当てられるトランザクションサーバーの数が制限されます。 |
| Maximum batch jobs | 並行して実行できるバッチジョブの最大数を指定します。<br>「Maximum background tasks」と「Maximum batch jobs」の合計数は、「Transaction servers」の数と等しいか、それより少ない値になる必要があります。 |

| フィールド | 説明 |
|---|---|
| Batch search interval | 実行待ちバッチプログラムを領域が照会する頻度を制御します。1～9999 までの秒数で、デフォルトは 120 秒です。 |

「Connect to Batch Node」および「Node Dir」フィールドは、バッチ処理のために Sun MBM が Sun MTP に接続しているかどうかを示します。Sun MBMで使用する領域の設定については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

# バッチタスク

バッチ環境では、次のようなさまざまなタスクを実行できます。

- ファイルのバックアップなど、オペレーティングシステム固有のジョブの実行
- VSAM データベースにアクセスするアプリケーションの実行
- VSAM データセットを管理するための `unikixbld` および `kixfile` ユーティリティーの使用
- リソース管理のための `dfhusdup` ユーティリティーの使用
- VSAM データセットに影響しない、その他のバッチユーティリティーの使用

## バッチシェルスクリプトでのエラー処理

バッチシェルスクリプトには、一連の Sun MTP 付属のユーティリティー (`unikixvsam` または `unikixbld` など)、オペレーティングシステムコマンド、Sun 以外のユーティリティー、制御文、環境設定文、およびその他の文を記述できます。

Sun MTP 付属のユーティリティーをバッチスクリプトで使用する場合、ユーティリティーまたはスクリプトの次のステップを実行するかどうかを判定するための復帰状態を取り込む必要があります。エラー条件が致命的でない場合、制御は次の手順に進みます。エラー条件が致命的である場合、その状態のスクリプトを終了します。

Sun MTP ユーティリティーにより引き起こされたエラー条件が適切に処理されない場合、次の Sun MTP ユーティリティーでロールバックが暗黙的に実行されます。問題のユーティリティーがスクリプトの最後のステップである場合、トランザクション処理プログラム (`unikixtran`) がロールバックを暗黙的に実行します。

# COBOL バッチプログラムの実行

この節では、VSAM データセットにアクセスする COBOL バッチプログラムを実行する方法について説明します。

## COBOL バッチプログラムからのデータファイルへのアクセス

Sun MTP ファイル編成は、CICS で使用される VSAM のアクセス方法と互換性を持つように設計されています。このファイル編成は、COBOL で使用する通常のファイル編成とは別のものです。 1 つのプログラムで、次の両方のファイル編成にアクセスできます。

- COBOL ランタイムファイル編成は、順編成ファイルでアクセスされます。
- Sun MTP ファイル編成は、索引付きレコードまたは相対レコードでアクセスされます。

バッチ COBOL プログラムを使用する際、`SELECT` 文の `ASSIGN` および `ORGANIZATION` 節、およびファイル記述エントリの `RECORD` 節は、どのファイル編成を使用するかを決定するために使用されます。

`SELECT` 文の `ASSIG` 節で使用できる形式は、次のとおりです。

- 物理的な順編成ファイル:

```
ASSIGN TO name
ORGANIZATION IS SEQUENTIAL
ORGANIZATION IS LINE SEQUENTIAL
ORGANIZATION IS RECORD SEQUENTIAL
```

- VSAM 相対編成ファイル:

```
ASSIGN TO name
ORGANIZATION IS RELATIVE
```

- VSAM 索引編成ファイル:

```
ASSIGN TO name
ORGANIZATION IS INDEXED
```

VSAM データセットにアクセスするとき、RECORD 節が正しい形式であると、Sun MTP ファイル処理が実行されます。

- プログラムが起動されるときに、*name* ファイルが読み取られます。これはアクセスされるデータセットです。
- *name* は 1 ～ 8 文字で、VSAM カタログおよび FCT を通じて物理ファイルに関連付けられます。VSAM アクセスには VSAM カタログおよび FCT が必要なので、ファイル指定で Micro Focus COBOL から利用できるオプションが Sun MTP からすべて利用できるとはかぎりません。

ASSIGN 節で EXTERNAL オプションを使用する場合、バッチプログラムを実行しているシェルスクリプトは *name* を上書きできます。このオプションでは、Sun MTP は dd_name または DD_nameという名前の環境変数を検索するために *name* パラメータを使用します。この環境変数の値が結果として使用されます。

ファイルが複数のファイル記述によって同時にオープンされる場合、プログラムのファイル記述エントリに異なる名前を使用する必要があります。これらのエントリを同じ物理ファイルに関連付ける場合、ASSIGN 節の EXTERNAL オプションが必要になります。

Sun MTP はオンラインアプリケーションの入力順データセット (ESDS) をサポートしますが、Sun MTP バッチプログラムはこれらのファイルに直接アクセスできません。

---

**注 –** ネイティブ ESDS ファイルのバッチ処理は、Sun MBM でサポートされています。

---

標準のバッチ環境で ESDS データセットにアクセスするには、以下の手順で説明されているように、順編成ファイルに変換する必要があります。

## ▼ ESDS ファイルを処理する

1. **シーケンシャル、行順、またはレコード順のいずれかの形式で順編成ファイルを作成することにより、ESDS ファイルをアンロードします。**

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

2. **作成した新しい順編成ファイルを使用して、バッチプログラムを実行します。**

   バッチプログラムは、ファイルのアクセスに VSAM ではなく COBOL のファイルハンドラを使用します。

3. **順編成ファイルを VSAM 編成に復元します。**

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

   この時点で、オンラインプログラムは VSAM 保護ファイルとしてそのファイルにアクセスできます。

ESDS ファイルが順編成ファイルで、バッチプログラムにより更新されているときに、その他のプログラムは元の VSAM ファイルにアクセスしてはいけません。この時点でアクセスが必要となる場合、読み取り専用アクセスを行います。

Sun MTP VSAM はマルチユーザーなので、バッチプログラムが更新をコミットまたはロールバックできる追加コマンドが提供されています。このコマンドについては、349 ページの「データ整合性の制御」を参照してください。

## Server Express バッチプログラムのコンパイル

次の表に、バッチプログラムのコンパイル時に使用できるオプションとその命令を示します。コンパイラオプションおよび命令の詳細は、Server Express のマニュアルを参照してください。

**表 15-1** バッチ Server Express プログラムのコンパイラオプションおよび命令

| オプション/命令 | 機能 | 注 |
|---|---|---|
| `-C ibmcomp`<br>`ibmcomp` | COBOL のコンパイラ命令 | 必須。IBM 370 互換機の OS/VS COBOL コンパイラと同じサイズのデータを割り当てます。 |
| `-C assign"external"` | COBOL のコンパイラ命令 | Sun MBM 製品に対して、または標準バッチで `DD` *filename* を使用している場合に必須です。 |
| `-C defaultbyte"32"` | COBOL のコンパイラ命令 | 必須。記憶域が初期化される方法を制御します。COBOL プログラムで宣言され、それに関連する VALUE 文を持たない記憶域です。バッチプログラムでは、この命令を 32 に設定して、記憶域にスペースがセットされるようにします。 |
| `-i`<br>`nognt` | 拡張子 `.int` を持つ中間コードを生成します | 開発または入出力集中のプログラムでこのオプションを使用します。`.gnt` ファイルを作成するより早くコンパイルできます。 |
| `-C sequential"record"` | COBOL のコンパイラ命令 | COBOL II バッチプログラム用。 |
| `-u`<br>`gnt` | 拡張子 `.gnt` を持つ実行可能プログラムを生成します | `.gnt` ファイルは `.int` または `.idy` ファイルより早い速度で実行します。実稼動環境でプログラムを使用する場合は、このオプションを使用します。 |
| `-a`<br>`anim` | 拡張子 `.idy` を持つ Animator 出力を生成します | Animator はバッチプログラムのデバッグに使用されます。Server Express は `.gnt`、`.idy`、および `.int` ファイルをアニメーション表示できます。 |
| `notrunc` | COBOL のコンパイラ命令 | IBM COBOL と同じ方法でバイナリフィールドを処理します。 |

次の例は、１つのディレクトリ内のすべてのバッチプログラムをコンパイルするスクリプトです。このスクリプトは、エラーまたは正常な完了を示すログファイルも取り込みます。

**コード例 15-1**　ディレクトリ内のすべての Server Express バッチプログラムをコンパイルするスクリプト

```
for i in `ls *.cbl`
do
       echo about to compile `basename $i .cbl`.cbl
       echo about to compile `basename $i .cbl`.cbl >> COMPALL.err
       cob <options and directives> `basename $i .cbl`.cbl
       ReturnCode=$?
       if [ $ReturnCode -ne 0 ]
           then
           echo "Error detected during cob of `basename $i .cbl`" >> COMPALL.err
           else
           echo "Compile completed for `basename $i .cbl`" >> COMPALL.err
       fi
done
```

## ACUCOBOL-GT バッチプログラムのコンパイル

次の表に、ACUCOBOL-GT バッチプログラムのコンパイル時に使用できるコンパイラオプションを示します。コンパイラオプションおよび命令の詳細は、ACUCOBOL-GT のマニュアルを参照してください。

**表 15-2**　ACUCOBOL-GT バッチプログラムのコンパイルオプション

| オプション | 注 |
|---|---|
| -Ca | 必須。DISPLAY は、ANSI スタイルです。 |
| -Cv | オプション。IBM DOS/VS COBOL と互換性があります。 |
| -Dv=32 | 必須。記憶域が初期化される方法を制御します。これは COBOL プログラムで宣言され、それに関連する VALUE 文はありません。バッチプログラムでは、この命令を 32 に設定して、記憶域がスペースに設定されるようにします。 |
| -Ga | オプション。デバッグ用にプログラムをコンパイルします。 |

次の例は、1つのディレクトリ内のすべてのバッチプログラムをコンパイルするスクリプトです。このスクリプトは、エラーまたは正常な完了を示すログファイルも取り込みます。

**コード例 15-2** ディレクトリ内のすべての ACUCOBOL-GT バッチプログラムをコンパイルするスクリプト

```
for i in `ls *.cbl`
do
       echo about to compile `basename $i .cbl`.cbl
       echo about to compile `basename $i .cbl`.cbl >> COMPALL.err
       cob <options> `basename $i .cbl`.cbl
       ReturnCode=$?
       if [ $ReturnCode -ne 0 ]
           then
           echo "Error detected during cob of `basename $i .cbl`" >> COMPALL.err
           else
           echo "Compile completed for `basename $i .cbl`" >> COMPALL.err
       fi
done
```

## 標準のバッチを使用したプログラムの実行

標準のバッチ環境では、Sun MTP は、ジョブが `$KIXBTCH` にコピーされたあとでジョブを自動的に実行します。リソースが使用できるようになると、各プログラムおよびシェルスクリプトが開始されます。リソースが利用できない場合、プログラムおよびシェルスクリプトは待機します。

Sun MTP は、COBOL プログラムまたはシェルスクリプトを実行できます。Server Express 実行可能プログラムのファイル拡張子は、`.gnt` または `.int` です。実行可能プログラムのファイル拡張子は、`.acu` です。シェルスクリプトは COBOL プログラムを呼び出すことができる一連のコマンドであり、シェルスクリプトを呼び出す必要はありません。シェルスクリプトのファイル名には、拡張子が付いている場合と付いていない場合があります。プログラムおよびシェルスクリプトが実行可能であることを確認します。

### ▼ プログラムを実行する

- **プログラムを `$KIXBTCH` ディレクトリにコピーします。**

  プログラムがバッチ検索間隔に基づいて実行されます。

**注意 –** プログラムまたはシェルスクリプトは、`$KIXBTCH` に移動するのではなく、コピーしてください。バッチジョブが開始すると、プログラムまたはスクリプトは `$KIXBTCH` から削除されます。

プログラムおよびシェルスクリプトは、領域を開始したユーザー ID で実行される必要があります。すべての出力は、領域を開始したユーザー ID の端末に書き込まれます。

シェルスクリプトまたは COBOL プログラムが VSAM データセットを参照する場合、データセット名には FCT で指定された 8 文字の名前が使用されます。プログラムでは、絶対または相対編成ファイル名を使用してデータセットを参照できます。相対編成ファイル名が使用されている場合、現在のディレクトリが基準です。バッチファイルでは、`$KIXSYS` ディレクトリが現在のディレクトリです。

シェルスクリプトまたは COBOL プログラムがほかのファイルを参照する場合、COBOL のマニュアルの指示どおりに名前を指定する必要があります。

COBOL プログラムをシェルスクリプトの内部で実行する場合、プログラムは `unikixvsam` COBOL 実行時システムを使用して実行される必要があります。`unikixvsam` の形式およびオプションについては、『Sun Mainframe Transaction Processing 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。

詳細は、353 ページの「バッチ COBOL プログラムのデバッグ」も参照してください。

## Sun MBM からのプログラムの実行

Sun MBM に接続している領域は、`unikixjob` コマンドを使用して Sun MBM にサブミットされたジョブだけを実行します。こ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。

## オンライントランザクションを使用した バッチプログラムの実行

バッチプログラムは、`JOB` タイプのパーティション外一時データキュー (TDQ) に対してシェルスクリプトを書き込むオンライントランザクションによって開始できます。指定されたキューへの TDQ 書き込みは、ジョブキューに再び渡されるファイルに対して書き込まれます。

宛先管理テーブル (DCT) で TDQ を定義すると、`kixjob` シェルスクリプトがジョブキューとのインタフェースとして使用されます。ジョブキューに対して書き込みが行われたあと、トランザクションが同期点を発行するときに `kixjob` が呼び出されます。この同期点は、`EXEC CICS SYNCPOINT` コマンドを使用して明示的に発行されるか、トランザクションの終了によって暗黙的に発行されます。

---

**注 –** 領域が Sun MBM に接続されている場合の `kixjob` の使用法については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

---

## 複数ステップのバッチジョブの実行

複数ステップのバッチジョブを実行するには、シェルスクリプトを使用する必要があります。シェルスクリプトでは、プログラムの実行順序を指定できます。また、シェルスクリプトによって、プログラム間でのファイルを引き渡し、ファイルの削除、およびファイルのコピーをバッチジョブで行うことが可能になります。

## unikixvsam プログラムへのパラメータの引き渡し

**注 –** この機能は、Sun MBM 以外の環境でのみ有効です。

unikixvsam プログラムが開始すると、パラメータはカードファイルか、コマンド行によってプログラムに渡されます。プログラムが標準 COBOL 入力ファイルからカードを読み込む場合、テキストエディタを使用してカードファイルを作成します。COBOL SELECT 文が行の順編成を指定していることを確認する必要があります。パラメータカード作成方法の 1 つに、次のような echo コマンドの使用があります。

```
$ echo "Parameter card data"> parm_data
```

これにより、テキスト PARAMETER CARD DATA を含む parm_data という名前の 1 レコード行の順編成ファイルが作成されます。

パラメータデータをプログラムに渡す別の方法として、プログラムの呼び出しがあります。プログラムの実行時に、次のようなコマンドでパラメータを渡します。

```
$ unikixvsam PGM00112 "Parameter card data"
```

プログラムは LINKAGE SECTION でパラメータカードの形式を定義し、PROCEDURE DIVISION USING 節を追加する必要があります。

プログラムの形式は、次のとおりです。

```
LINKAGE SECTION.
01  PARAMETER-AREA.
       05  PARAMETER-LENGTH PIC 9(04) COMP.
       05  PARAMETER-DATA PIC X(80).
PROCEDURE DIVISION USING PARAMETER-AREA.
1000-START.
```

unikixvsam は各パラメータに 160 バイトまでの領域を確保し、5 つまでのパラメータ領域を許可します。

**注 –** COBOL の ACCEPT FROM CONSOLE 文はサポートされていません。この節で示す例を使用して、この文を変更できます。

## リターンコードの設定

**注 –** この機能は、Sun MBM 以外の環境でのみ有効です。

COBOL では、プログラムにリターンコードを設定できます。リターンコードは、OS/VS COBOL オプションを設定することにより、呼び出し先プログラムまたは呼び出し元シェルスクリプトに渡されます。リターンコードを設定する方法は、次のとおりです。

```
SET RETURN-CODE TO 1.
GOBACK.
```

オペレーティングシステムは、特殊レジスタ RETURN-CODE をシングルバイトに切り捨てます。そのため、COBOL プログラムで RETURN-CODE 変数が 257 に設定されている場合、シェルスクリプトは値 1 を受け取ります。

unikixvsam では、この変数の詳細を定義します。この変数が 0 ～ 127 の場合、単にその値が返されます。値が 127 を超える場合、コミットされていないすべての VSAM 更新はロールバックされます。

Korn シェルスクリプトのリターンコードの値をテストするには、$? 構成子を使用します。これは、実行ストリームで次にどのプログラムを実行するかを制御できます。

次の例では、COBOL プログラム RAM59 のリターンコードをテストします。リターンコードがエラーの発生を示している場合、echo コマンドが実行され、exit コマンドがリターンコードをパラメータにしてシェルスクリプトの実行を終了します。

```
unikixvsam RAM59 "RAM0599"
cond=$?
if [$cond -ne 0]
then
       echo "RAM59 ERROR CODE = $cond"
       exit $cond
fi
```

# PL/I バッチプログラムの実行

PL/I バッチプログラムの実行には、Sun MBM を使用する必要があります。

# C 言語のバッチプログラムの実行

---

**注 –** この機能は、Sun MBM 以外の環境でのみ有効です。

---

Sun MTP は、C-ISAM インタフェースを使用して VSAM データセットにアクセスする C 言語のバッチプログラムの実行をサポートします。C バッチプログラムを実行する前に、それらをコンパイルする必要があります。プログラムをコンパイルしたシステムと同じバージョンの Sun MTP を実行しているシステムでだけそのプログラムを実行できます。C-ISAM 機能の詳細は、第 16 章を参照してください。

## ▼ C 言語のバッチプログラムをコンパイルする

1. `UNIKIX` 環境変数を、プログラムを実行する領域に設定します。
2. `isam.h` ヘッダーファイルを C プログラムに取り込みます。
   このファイルは、`$UNIKIX/src/CICS_structures` ディレクトリにあります。
3. コマンド行またはスクリプトファイルで次のコマンドを実行し、プログラムをコンパイルして実行可能プログラムを作成します。

```
$ cc input-filename.c -o output-filename $UNIKIX/lib/libbcisam.a
```

4. ファイルの実行権があることを確認します。

## ▼ プログラムを実行する

1. シェルスクリプトを作成します。

```
$ vi cbat01.sh
```

2. スクリプトにパス名およびバッチプログラム名を入力します。

```
batch/bat001
```

3. スクリプトを保存し、`:wq` と入力してエディタを終了します。

4. アクセス権を設定してスクリプトを実行可能にします。

```
$ chmod 777 cbat01.sh
```

5. シェルスクリプトを $KIXBTCH ディレクトリにコピーして、そこから自動的に実行されるようにします。

```
$ cp cbat01.sh $KIXBTCH
```

## unikixbld を使用したバッチジョブからの VSAM データセットへのアクセス

unikixbld ユーティリティーは、領域で実行されているバッチジョブから VSAM データセットを操作します。これらの機能を実行するプログラムの記述もできますが、通常は、このユーティリティーを使用するほうが便利です。unikixbld は、次の機能を実行します。

- 順編成ファイルからの VSAM データセットの構築
- VSAM データセットからの順編成ファイルの構築
- VSAM データセットの初期化
- VSAM データセットからの索引ファイルの再構築
- VSAM データセットからの代替索引ファイルの再構築
- 複数の順編成ファイルから 1 つの VSAM データセットへのマージ

---

**注 –** unikixbld には、IDCAMS REPRO の動作を実行するオプションがあります。

---

unikixbld の詳細は、『Sun Mainframe Transaction Processing 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。

次の表に、特定の機能を実行するために使用する unikixbld の形式を示します。

**表 15-3** unikixbld の機能

| 機能 | 形式 |
|---|---|
| VSAM データセットの初期化 | unikixbld -d *VSAM-dataset* -i |
| VSAM データセットからの 1 つまたは複数の順編成ファイルの構築 | unikixbld -t s -d *VSAM-dataset* -s *filename* [-s *filename* [-s *filename* [...]]] [-r *format*] [-l *bytes* [K\|M\|G]] [-v *n_rec*] |
| 1 つまたは複数の順編成ファイルからの VSAM データセットの構築 | unikixbld -t v -d *VSAM-dataset* -s *filename* [-s *filename* [-s *filename* [...]]] [-f *percent*] [-r *format*] [-m] [-M *altkey_mem*] [-v *n_rec*] |

表 15-3　unikixbld の機能 (続き)

| 機能 | 形式 |
|---|---|
| VSAM データセットの索引部の再構築 | `unikixbld -t x -d` *VSAM-dataset* `[-v` *n_rec*`]` |
| VSAM 1 次データセットの VSAM 代替索引ファイルの再構築 | `unikixbld -t a -d` *VSAM-dataset* `[-M` *altkey_mem*`]` `[-v` *n_rec*`]` |
| VSAM データセットの更新 | `unikixbld -d` *VSAM-dataset* `-p [reuse\|noreuse] [-k {replace\|noreplace}] -W` *nnn* `-V` |

## kixfile を使用したバッチジョブからの VSAM データセットの制御

kixfile は、関連する代替データセットなどの VSAM データセットの状態を動的に変更できるバッチユーティリティーです。kixfile を使用すると、次のような VSAM データセットの制御ができます。

- 状態の設定 (オープンまたはクローズ)
- オンライン状態の設定 (読み取りまたは読み取り/書き込み)
- バッチ状態の設定 (バッチアクセスのみまたはバッチ/オンラインアクセス)
- ロック状態の設定 (排他ロックまたはロック解除)
- アクセス権の設定 (読み取り専用または起動時の状態にリセット)
- 回復状態の設定 (回復の有効化、回復の無効化、または回復の無効化に伴う VSAM データセットのシステムキャッシュの有効化)
- 現在のデータセット状態の印刷

kixfile の形式およびオプションについては、『Sun Mainframe Transaction Processing 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。

## dfhusdup を使用したリソースの定義およびレポート

dfhusdup は、システム定義ファイル (DFHUSD) のリソースを定義およびレポートするために使用するバッチユーティリティーです。DFHUSD ファイルの初期化にも使用します。dfhusdup の形式およびオプションについては、『Sun Mainframe Transaction Processing 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。代替資源定義については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』も参照してください。

行の 1 列目にアスタリスク (*) を入力することにより、バッチファイルにコメントを入力できます。コマンド、属性、または属性タイプが次の行に続く場合は、72 列目にアスタリスクを入力する必要があります。

**例**: この例では、バッチファイルでのコメントおよびコマンドの継続を表すアスタリスクの使用法について示しています。

```
*Define a program
DEFINE GROUP (SALES1) PROGRAM (SAL001) DESCRIPTION (Daily sal*
       es program)
```

次の例では、dfhusdup ユーティリティーの使用法を示します。

**例**: スクリプトから実行される次の dfhusdup コマンドは、load という名前のファイルを指定します。このファイルには、リソースを初期化、定義、および一覧表示するコマンドが含まれます。dfhusdup ユーティリティーによって生成されるすべてのメッセージは、loadout ファイルにリダイレクトされます。

```
dfhusdup USD=READWRITE PAGESIZE=50 CMDFILE=load > loadout
```

ファイルロードの内容は次のとおりです。

```
*Resource definitions for Sales Department
INITIALIZE
DEFINE GROUP (SALES1) PROGRAM (SAL001) DESCRIPTION (Daily s*
       ales program)
DEFINE GROUP (SALES1) MAPSET (SALAD01) DESCRIPTION (Daily sal*
       es add screen)
DEFINE GROUP (SALES1) MAPSET (SALDE01) DESCRIPTION (Daily sal*
       es delete screen)
DEFINE GROUP (SALES1) MAPSET (SALMF01) DESCRIPTION (Daily sal*
       es modify screen)
LIST ALL OBJECTS
```

**例**: 次のスクリプトの例では、CMDFILE=stdin 引数の使用法およびリソースの削除方法を示しています。

```
dfhusdup USD=READWRITE CMDFILE=stdin <<!
*To delete an old program and mapset from the resource file
DELETE GROUP (SALES1) PROGRAM (OLDPGM)
DELETE GROUP (SALES1) MAPSET (OLDMAP)
!
```

**例**: 次のスクリプトの例では、あるグループから他のグループにリソースをコピーする方法を示しています。

```
dfhusdup USD=READWRITE CMDFILE=stdin <<!
*To copy group 'sales1' resources to new group 'sales2'
COPY GROUP (SALES1) TO (SALES2)
!
```

# VSAM データセットに影響しない Sun MTP バッチユーティリティー

VSAM データセットに影響しないバッチユーティリティーは、次の 2 つです。

- `kixlog` は、Sun MTP シェルスクリプトと Sun MTP ログファイル、`unikixmain.log`、`unikixmain.err`、および `unikixmain.dbg` との間のインタフェースを提供します。`kixlog` ユーティリティーでは、選択したログファイルに文字列を配置できます。詳細は、『Sun Mainframe Transaction Processing 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。
- `kixtran` は、バッチまたはシェルプロンプトから Sun MTP トランザクションを開始します。詳細は、『Sun Mainframe Transaction Processing 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。

# VSAM バッチジョブに関する統計情報

`unikixvsam`、`unikixbld`、`kixfile` ユーティリティー、さらに Sun MBM の `unikixjob` ユーティリティーに関して、統計情報を表示できます。これらの情報は、バッチジョブ実行時のパフォーマンスの問題を診断するのに役立ちます。

各ユーティリティーで、次の統計情報がモニターに表示されます。モニターには、実際の値が表示されます。

```
Command Executed:"command and parameters"
Batch Process 99999 Start Time hh:mm:ss End Time hh:mm:ss Elapsed Time 9.99 secs
System CPU Time 9.99 secs User CPU Time 9.99 secs
```

各ユーティリティーには、追加の統計情報を表示するために、コマンド行で入力できる引数があります。

- `unikixvsam` では、次の統計情報を表示できます。
  - VSAM コードでの経過時間 (`-X VSTIME`)
  - 待機時間 (`-X WTIME`)

- 論理入出力 (-X LIO)
- 物理入出力 (-X PIO)

- unikixbld では、次の統計情報を表示できます。
  - 物理入出力 (-X PIO)
  - 待機時間 (-X WTIME)
- kixfile では、VSAM コードでの経過時間を表示できます (-X VSTIME)。

これらの統計情報を表示するコマンド構文については、『Sun Mainframe Transaction Processing 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。unikixjob ユーティリティーについては、Sun MBM のマニュアルを参照してください。

次に示す一連の例では、それぞれのオプションによる出力を示します。

command *arguments* -X VSTIME

```
Total time in VSAM:9.99 secs Time in O/S calls in VSAM:9.99 secs VSAM code
execution time 9.99 secs
```

command *arguments* -X WTIME

```
Buffer wait time:9.99 secs I/O Wait Time:9.99 secs Semaphore Wait Time 9.99 secs
```

command *arguments* -X LIO

```
Logical File I/O Statistics
File Reads Writes Rewrites Deletes Startbrs Endbrs Resetbrs Unlocks Readnexts Readprevs
ABC     9    9        9       9        9      9       9        9       9         9
```

command *arguments* -X PIO

```
Physical File I/O Statistics
File          Reads      Writes     Allocs
ABC            9          9           9
```

**注 –** 統計情報の収集および表示に関するオプションを使用しても、パフォーマンスにはほとんど影響を与えません。

# バッチ検索間隔の上書き

バッチプログラムを今すぐ実行するには、CBCH トランザクションを使用して VCT で設定されているバッチ検索間隔を上書きします。CBCH は、バッチプログラムがフォアグラウンドトランザクションであるかのようにバッチプログラムを開始します。

形式

CBCH *batch-program*

batch-program は、開始するバッチプログラムの名前です。プログラム名のみを指定すると、$KIXSYS からの相対ディレクトリからプログラムが実行されます。プログラムが $KIXSYS のサブディレクトリに存在しない場合、フルパスおよびファイル名を入力します。

---

**注 –** 領域が Sun MBM に接続している場合、バッチ検索間隔の上書きに CBCH を使用できません。

---

# データ整合性の維持

VSAM RC (Read Consistency) により、バッチプログラムは、I-O モードでオープンされているファイルに対して排他レコードロックを獲得できます。

VSAM RC には、次の利点があります。

- あるバッチプログラムがオンラインおよびその他のバッチプログラムと同時にファイルを共有しているときに、データの整合性を維持します。
- アプリケーションのバッチウィンドウを短縮して、オンラインアプリケーションにより多くの処理時間を確保します。

VSAM RC を使用する場合、次に示す事項について理解しておいてください。

- 340 ページの「アプリケーション設計の問題」
- 342 ページの「VSAM RC のための既存バッチプログラムの変更」

読み取りの一貫性を必要としないプログラムは、「ダーティー読み取り」と呼ばれる読み取り専用のレコードを要求できます。これは、VSAM のデフォルトの動作で、ソフトウェアの以前のバージョンの動作です。VSAM RC を使用しない場合は、複数のバッチプログラム間およびバッチとオンラインプログラム間で、リソースを共有できません。

Sun MBM を使用して VSAM データセットにアクセスする場合、DD JCL 文の RLS パラメータの値について考慮する必要があります。RLS パラメータについては、Sun MBM のマニュアルを参照してください。

## VSAM RC

リリース 7.1 より前のバージョンでは、READ が発行されたときに、バッチプログラムは排他レコードロックを獲得できませんでした。プログラムが READ を要求すると、レコードはロックなしで読み取られていました。プログラムが REWRITE または DELETE を発行すると、レコードはロックされて再度読み取られ、次の操作が実行されました。この実装では、READ 要求と REWRITE または DELETE 要求との間のレコード保護が保証されないので、更新または削除に対して無防備な状態でした。

VSAM RC を使用すると、バッチプログラムは排他ロックを獲得してレコードを読み取ることができ、バッチプログラムの代わりに VSAM RC がこのレコードを保持します。オンライントランザクションであるか他のバッチプログラムであるかに関係なく、同じレコードに対するロックの要求は、ロックを所有するバッチプログラムがそれを解放するまでキューに入れられます。

- 回復がオフの場合、別の VSAM 入出力がそのファイルに対して発行されるまで、バッチプログラムはそのレコードのロックを保持します。
- 回復がオンで、バッチプログラムがレコードを更新する場合、次の COMMIT、ROLLBACK、またはプログラムの終了までロックを保持します。バッチプログラムがレコードを更新しない場合、別の VSAM 入出力がファイルに対して発行されるまでロックが保持されます。

## VSAM RC の制限事項

アプリケーションに VSAM RC を実装する場合、次の制限事項に注意してください。

- ファイル (データセット) が I-O モードでオープンされている場合にのみ、VSAM RC を適用できます。ファイルが INPUT、OUTPUT、または EXTEND モードでオープンされている場合、VSAM RC が有効であっても適用されません。
- Sun MTP ソフトウェアでは、プログラムの各データセットに対して 1 レコードのみに排他ロックが可能です。これは、同じファイルを指していて異なる SELECT 節を使用して複数回オープンされるファイルに該当します。READ がロックを使用して発行されると、バッチプログラムの代わりに VSAM がレコードを保持するので、これは非常に重要です。異なるキーに対して REWRITE または DELETE が発行された場合、Sun MTP はそのレコードのロックを破棄し、REWRITE または DELETE が発行されたレコードでロックを獲得します。
- 代替キーファイルで VSAM RC はサポートされません (1 次ファイルとしての代替キーファイルのオープンなど)。

- ANSI COBOL API を使用する場合、次のオプションがサポートされます。その他のオプションを使用するとエラーが発生します。
  - SELECT の節での LOCK MODE IS AUTOMATIC
  - SELECT の節での LOCK MODE IS MANUAL
  - READ 文での READ WITH LOCK、READ WITH NO LOCK、READ NEXT WITH LOCK、READ NEXT WITH NO LOCK、READ PREVIOUS WITH LOCK、READ PREVIOUS WITH NO LOCK
  - UNLOCK
  - COMMIT
  - ROLLBACK

---

**注 –** Server Express は、COMMIT および ROLLBACK 文をサポートしません。プログラムで kixvsam を使用する必要があります。詳細については、349 ページの「データ整合性の制御」を参照してください。

---

- すべての VSAM RC 機能は、標準のバッチでサポートされます。C-ISAM を使用するプログラムに対しては、VSAM カタログおよび環境変数設定を使用して読み取り一貫性を保証します。C-ISAM でのプログラムレベルのロックは、このリリースではサポートされていません。

# アプリケーション設計の問題

アプリケーションが VSAM RC 機能を必要としない場合、既存のオンラインおよびバッチ実行モデルを継続して使用できます。以前のリリースからアップグレードする場合は、VSAM カタログを変換する必要があります。詳細は、344 ページの「API ロックを使用しないプログラム」を参照してください。

アプリケーションがバッチプログラムおよびオンライントランザクション間のリソースの共有を必要とする場合、次の事項を検討してください。

- **バッチウィンドウ**

  バッチウィンドウとは、オンラインシステムを休止しバッチプログラムがファイルを更新し、レポートを印刷している時間です。バッチに割り当てられている時間が十分でない場合、バッチウィンドウを減らすために、並列実行できるプログラムおよびバッチステップを特定する必要があります。

- **共有リソース**

  バッチプログラムで共有されるリソースを特定してください。同時に実行するすべてのバッチプログラムによって I-O モードでオープンされるファイルのすべてを組み入れます。たとえば、プログラム A が File1 および File2 を更新し、プログラム B が File2 および File3 を更新する場合、File1、File2、および File3 に対して VSAM RC を有効にします。これにより、これらのファイルのいずれかにアクセスするその他のバッチプログラムまたはオンライントランザク

ションから保護されます。一部のバッチプログラムでファイルに対して VSAM RC を有効にし、他のバッチプログラムでは無効にしている場合、データ整合性は保証されません。

次の例は、2 つのバッチプログラム C、D が `account` ファイルを共有する方法を示しています。共有ファイルに対して VSAM RC を有効にしていない場合、バッチプログラムはデータの不整合を起こす可能性のあるダーティー読み取りを実行します。この例では、ある銀行口座に ＄1000 の残高があるとします。VSAM RC を使用する プログラム C はこの口座から $100 を引き落とし、VSAM RC を使用しない プログラム D はこの口座に $500 を入金します。これにより、次のように推移します。

1. プログラム C が口座残高 $1000 を読み取り、レコードをロックします。
2. プログラム D が口座残高 $1000 を読み取ります (VSAM RC を使用していないので、ダーティー読み取り)。
3. プログラム D が口座に $500 を預け入れて $1500 と書き込もうとしますが、プログラム C がレコードを解放するのを待つ必要があります。
4. プログラム C が口座から $100 を引き落とし、残高を $900 とします。
5. プログラム C が残高 $900 と書き込み、`COMMIT` またはプログラムの終了によってレコードを解放します。
6. プログラム D が残高 $1500 と書き込みます。

以上のように推移した結果の口座残高は $1500 です。ステップ 5 の前にステップ 6 が発生すると、口座残高は $900 になります。実際の口座残高は $1400 です。プログラム D が作業を行う前にプログラム C が作業を完了したり、あるいはその逆である場合には、正しい結果が得られます。

- **時間見積もり**

  バッチプログラムが 1 つのスレッドで実行されるか、排他ロックされたファイルで実行されるとき、すべてのリソースが占有されます。VSAM RC が有効な場合、バッチプログラムを並列で実行できます。実行時間は改善されていますが、予想以上に時間がかかる場合もあります。たとえば、バッチジョブでステップ 1 およびステップ 2 を行うのにそれぞれ 1 時間かかるとします。並列で実行する場合、理論的には 2 つのステップを完了するのに 1 時間ですみます。実際には、ステップ 1 が獲得したリソースが解放されるまで、ステップ 2 が待機しなければならない場合があります。ほかのシナリオでは、ステップ 1 がレコードで排他ロックを持っていて、オンライントランザクションがそのレコードを要求する場合、そのレコードが解放されるまでオンライントランザクションは待機する必要があります。オンライントランザクションの待機時間によっては、ユーザーへの応答時間が増加します。

- **操作手順の変更**

  一般的なバッチ実行プロセスのとき、ファイルは更新される前に定期的にバックアップされます。更新処理に失敗し、バッチジョブが特定のオプションで再起動すると、バックアップファイルが復元され、更新プロセスが再開されます。VSAM RC を使用すると、この手順が変更になり、開始点としてバックアップファイルを使用できません。これは、オンライントランザクションまたは異なるバッチプログラムによって更新されたレコードがファイルに含まれる場合があるためです。

回復が有効である場合、次の事項も考慮する必要があります。

- **共有メモリーサイズ**

  バッチプログラムのデータの整合性を維持するには、ファイルの回復を有効にする必要があります。VSAM は、ロックされた各リソース (レコード) に約 300 バイトの共有メモリーを使用します。バッチプログラムで更新される各レコードに対して、VSAM はロックを取得します。バッチプログラムは一般的に大量のレコードを更新するので、大量のロックが発生する場合があり、その結果、共有メモリーサイズが大幅に増加することがあります。

---

**注 –** メモリーは、プログラムまたはトランザクションが `COMMIT` を発行したあとで再利用されます。

---

- **回復ファイルのサイズ**

  回復ファイルは、レコードの更新前イメージおよびブロック分割情報を保存するために使用される、固定サイズの循環ファイルです。バッチプログラムにはコミットされていない大量のレコードが残されていることがあるので、回復ファイルのサイズの増加が必要な場合があります。

# VSAM RC のための既存バッチプログラムの変更

既存のバッチプログラムの多くは、実行前にすべての必要なリソースが利用できるようになることを前提としています。VSAM RC を使用する場合、次のような状況に対応できるようにバッチプログラムを変更する必要があります。

- **デッドロック**

  デッドロックとは、2 つのプログラムがそれぞれ、他方のプログラムがレコードを解放するのを待機している状態のことをいいます。たとえば、プログラム 1 がレコード 1 を排他ロックしたままレコード 2 の使用を待機し、プログラム 2 がレコード 2 を排他ロックしたままレコード 1 の使用を待機する状態です。VSAM はこのような状態を検出してエラーを返します。COBOL では、プログラム状態は 9D に設定されます。デッドロックを処理できるようにするには、バッチプログラムを変更する必要があります。バッチプログラムで `ROLLBACK` を発行して、すべてのロックされたレコードを解放し、プログラム開始以降または最後の `COMMIT` 以降の変更をすべて取り消します。

- **ファイルの共有**

  複数のバッチプログラムまたはバッチおよびオンラインプログラムでファイルを共有する場合、ファイルを共有可能なモードにする必要があります。ファイルが排他ロックされている場合には、排他ロックを解除するために `kixfile` コマンドを実行する必要があります。ファイルモードがバッチ専用またはオンライン読み取り専用モードである場合、モードを変更してファイルに対する制約を排除する必要があります。

- **選択的排他レコードロック**

  バッチプログラムで選択的レコードロックを使用する場合、ANSI COBOL 拡張機能を使用します。

回復が有効である場合、さらに次の事項を考慮する必要があります。

- **リソースの定期的解放**

  バッチプログラムが、ファイル内の 2000 レコードのうちの 1000 レコードを更新する場合、他のプログラムはそのバッチプログラムが 1000 レコードの使用を終了するまで待機する必要があります。これでは、応答時間が長くなり、共有メモリーなどのリソースに影響する恐れがあります。これを回避するために、定期的に `COMMIT` を発行して、他のプログラムに対してレコードを解放するようにバッチプログラムを変更します。`COMMIT` を発行することによって、データの整合性が維持されます。たとえば、バッチプログラムによって口座残高をオフラインで更新するとします。当座預金ファイルから $100 を引き出して、それを普通預金ファイルに預け入れます。当座預金からお金を引き出したあとで `COMMIT` を発行しても、対応する情報は普通預金ファイルに反映されません。顧客が普通預金を閉じると、バッチプログラムが普通預金ファイルに $100 を入金する前に、オンラインプログラムが普通預金ファイルのレコードを削除することがあります。

- **バッチステップの再起動**

  プログラムが定期的な `COMMIT` を発行し、プログラムが強制的に中止した場合、再起動時にコミットされたレコードに対して整合性を保証する必要があります。たとえば、プログラムが、各 1000 レコードを更新したあとで `COMMIT` を発行しながら、`new_items` 順編成ファイルから VSAM 在庫管理マスターファイルのレコードを更新するとします。プログラムが 4200 レコードを更新したあとで強制的に中止した場合、4001 以降のレコードの更新を再開できる必要があります。

# API ロックを使用しないプログラム

ほとんどのバッチアプリケーションプログラムは、ロックに ANSI COBOL 機能を使用しません。この機能 (「アプリケーションプログラムロック機構」と呼ばれる) には、SELECT 文の LOCK MODE IS AUTOMATIC および LOCK MODE IS MANUAL 節、PROCEDURE DIVISION の READ WITH LOCK および READ WITH NO LOCK 文などがあります。

プログラムがアプリケーションのプログラムロックメカニズムを使用しない場合は、この節を読んでください。アプリケーションプログラムのロックメカニズムがある、または使用する予定である場合は、346 ページの「API ロックを使用するプログラム」も読んでください。

VSAM カタログのクラスタ (ファイル) レベルでは、「Batch Read-locking」フィールドで、バッチ VSAM 読み取りによって読み取られるレコードをロックするかどうかを指定します。このフィールドが Y の場合、データセットのバッチ VSAM 読み取りはレコードをロックします。このフィールドが N の場合、データセットのバッチ VSAM 読み取りはレコードをロックしません。

リリース 7.0 以前のリリースからアップグレードする場合、kixcnvtcat72 ユーティリティーを使用して VSAM カタログを変換するとき、「Batch Read-locking」フィールドの値を決定する必要があります。選択した値は、変換されるすべてのファイル定義に適用されます。この値は、File Manager を使用してあとで変更できます。カタログに新しいファイルを追加するとき、「Batch Read-locking」フィールドに Y または N を指定する必要があります。VSAM カタログの管理については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

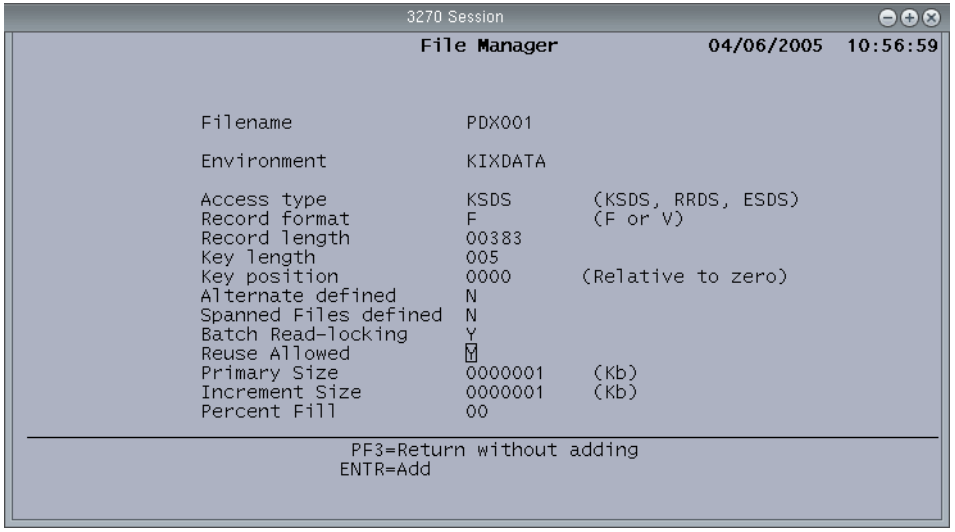


図 15-2　VSAM カタログでの読み取りロックの指定

**注 –** ソフトウェアの以前のリリースでのバッチ VSAM 読み取りロック機能を維持したい場合は、すべてのファイルに対して「Batch Read-locking」フィールドを N に設定します。

特定のジョブステップに対してクラスタレベルの読み取りロック指定を上書きする場合、次の 2 つの環境変数を操作できます。

```
KIX_READLOCKON=dataset 1:dataset 2:dataset 3:""..:dataset N

KIX_READLOCKOFF=dataset 1:dataset 2:dataset 3:""..:dataset N
```

KIX_READLOCKON 環境変数でデータセットがリストされている場合、そのデータセットの VSAM RC 読み取りは、クラスタレベルフラグの値にかかわらず、読み込んでいるレコードをロックします。KIX_READLOCKOFF 環境変数でデータセットがリストされている場合、そのデータセットの VSAM RC 読み取りは、クラスタレベルフラグの値にかかわらず、読み込んでいるレコードをロックしません。データセットが KIX_READLOCKON および KIX_READLOCKOFF 環境変数の両方でリストされている場合、バッチアプリケーションプログラムがデータセットを開くときにエラーが発生します。

これらの環境変数はクラスタレベルの指定の上書きに使用されるので、VSAM カタログの読み取りロックフィールドが Y である場合に、$KIX_READLOCKON でデータセットをリストする必要はありません。同様に、VSAM カタログでデータセットが N (ロックなし) である場合、$KIX_READLOCKOFF でこれを設定する必要はありません。

次の表に、クラスタレベル設定および環境変数に基づくバッチ VSAM 読み取りのロック動作をまとめます。

**表 15-4** クラスタ設定および環境変数に基づくロックの動作

| 条件 | CLUSTER = Y | CLUSTER = N |
|---|---|---|
| データセットがリストされる環境変数 $KIX_READLOCKON | ロック | ロック |
| データセットがリストされる環境変数 $KIX_READLOCKOFF | ロックなし | ロックなし |
| データセットはどちらの環境変数にもリストされません | ロック | ロックなし |

# API ロックを使用するプログラム

ANSI COBOL は、レコードを読み取るときのロック動作をアプリケーションに指定する方法を提供します。単純な READ に加えて、アプリケーションプログラムでは READ WITH LOCK および READ WITH NO LOCK を指定できます。API に基づき、READ WITH LOCK は読み込むレコードをロックし、READ WITH NO LOCK はレコードをロックしません。単純な READ のロックの動作は、LOCK MODE IS AUTOMATIC であるか、MANUAL であるかによって異なります。LOCK MODE は SELECT 節で指定します。MANUAL は単純な READ をロックしませんが、AUTOMATIC ではロックします。

LOCK MODE を指定するアプリケーションプログラムの読み取りロックの動作は、LOCK MODE、API、VSAM カタログでのクラスタレベルの設定、およびクラスタレベルの設定を上書きする環境変数に依存します。API ロックは、344 ページの「API ロックを使用しないプログラム」に示されている規則に基づいて読み取りロックが有効である場合にのみ有効です。

次の表に、ANSI COBOL ロック機構を使用するアプリケーションプログラムのロックの動作の概要を示します。

**表 15-5** クラスタが Y に設定されている場合の ANSI COBOL ロック

| CLUSTER = Y | ロックモードが MANUAL | ロックモードが AUTOMATIC |
|---|---|---|
| データセットがリストされる環境変数 $KIX_READLOCKON | READ (ロックなし)<br>READ WITH LOCK (ロック)<br>READ WITH NO LOCK (ロックなし) | READ (ロック)<br>READ WITH LOCK (ロック)<br>READ WITH NO LOCK (ロックなし) |
| データセットがリストされる環境変数 $KIX_READLOCKOFF | READ (ロックなし)<br>READ WITH LOCK (ロックなし)<br>READ WITH NO LOCK (ロックなし) | READ (ロックなし)<br>READ WITH LOCK (ロックなし)<br>READ WITH NO LOCK (ロックなし) |
| データセットはどちらの環境変数でもリストされません | READ (ロックなし)<br>READ WITH LOCK (ロック)<br>READ WITH NO LOCK (ロックなし) | READ (ロック)<br>READ WITH LOCK (ロック)<br>READ WITH NO LOCK (ロックなし) |

表 15-6　クラスタが N に設定されている場合の ANSI COBOL ロック

| CLUSTER = N | ロックモードが MANUAL | ロックモードが AUTOMATIC |
|---|---|---|
| データセットがリストされる環境変数 $KIX_READLOCKON | READ (ロックなし)<br>READ WITH LOCK (ロック)<br>READ WITH NO LOCK (ロックなし) | READ (ロック)<br>READ WITH LOCK (ロック)<br>READ WITH NO LOCK (ロックなし) |
| データセットがリストされる環境変数 $KIX_READLOCKOFF | READ (ロックなし)<br>READ WITH LOCK (ロックなし)<br>READ WITH NO LOCK (ロックなし) | READ (ロックなし)<br>READ WITH LOCK (ロックなし)<br>READ WITH NO LOCK (ロックなし) |
| データセットはどちらの環境変数でもリストされません | READ (ロックなし)<br>READ WITH LOCK (ロックなし)<br>READ WITH NO LOCK (ロックなし) | READ (ロックなし)<br>READ WITH LOCK (ロックなし)<br>READ WITH NO LOCK (ロックなし) |

クラスタレベルの設定は、API ロックを使用するプログラムと使用しないプログラムとでは異なる意味を持ちます。

表 15-5 では、次のことを示しています。

- クラスタレベル設定が Y の場合、またはデータセットが $KIX_READLOCKON で設定されている場合、API が有効になります。
- クラスタレベル設定が Y であるが、データセットが $KIX_READLOCKOFF で設定されている場合、API は無効になります。

表 15-6 では、次のことを示しています。

- クラスタレベル設定が N であるが、データセットが $KIX_READLOCKON で設定されている場合、API が有効になります。
- クラスタレベル設定が N である、またはデータセットが $KIX_READLOCKOFF で設定されている場合、API は無効になります。

## 読み取りロックの例

次に示す例では、読み取りロックの使用を検討する各種の状況について説明します。それぞれの状況は、API ロックに関する節に記載した動作に基づいています。

**状況**: アプリケーションプログラムに API ロックの指定が含まれず、読み取りロックも必要としません。

**推奨**: VSAM カタログの「Batch Read-locking」フィールドを、すべてのデータセットに対して N に設定します。環境変数を設定する必要はありません。このシナリオは、ソフトウェアの以前のバージョンにあったダーティー読み取りの動作と一致しています。

**状況**: アプリケーションプログラムに API ロックの指定が含まれていないが、バッチ読み取りでレコードをロックします。

**推奨**: VSAM カタログのクラスタレベル設定を Y に設定します。

**状況**: 試験的なバッチステップでバッチ読み取りロックを使用します。

**推奨**: そのバッチステップの $KIX_READLOCKON に関連データセットを取り込み、そのデータセットに対して VSAM カタログで、「Batch Read-locking」フィールドを N に設定します。

**状況**: ほとんどの場合にバッチ読み取りロックを有効にしますが、一部のバッチステップでロックを無効にします。

**推奨**: すべてのファイルに対して VSAM カタログの「Batch Read-locking」フィールドを Y に設定し、ロックを無効にするバッチステップに対して関連データセットを $KIX_READLOCKOFF に取り込みます。

**状況**: さまざまなバッチ読み取りロックの試験を行ったため、環境変数設定およびクラスタレベル設定が複雑になってしまいました。アプリケーションプログラムの実行速度が遅かったりハングアップするので、試験を開始する前の状態に戻すことにします。

**推奨**: すべての環境変数の設定を解除し、VSAM カタログでデータセットの「Batch Read-locking」フィールドを N に設定します。

**状況**: アプリケーションに API ロックが含まれているが、それらを無効にします。

**推奨**: VSAM カタログのクラスタレベル設定を N に設定します。クラスタレベル設定が Y に設定されている場合、ファイル名を KIX_READLOCKOFF 環境変数に追加します。

**状況**: アプリケーションに API ロックが含まれていて、それらを有効にします。

**推奨**: VSAM カタログのクラスタレベル設定が Y に設定されていることを確認するか、ファイル名を KIX_READLOCKON 環境変数に追加します。

**状況**: アプリケーションに API ロックが含まれています。API ロックは使用しないが、バッチ読み取りロックは使用します。

**推奨**: アプリケーションプログラムを変更して、API ロックを削除する必要があります。クラスタレベル設定が Y に設定されていることを確認するか、ファイル名を KIX_READLOCKON 環境変数に追加します。

# データ整合性の制御

Sun MTPは、バッチプログラムでデータ整合性を制御するための追加機能を提供します。

| 機能 | アクション |
|---|---|
| ROLLBACK | この機能の前に実行され、前の SYNCPT コマンド以降またはプログラム開始以降に実行された更新を取り消します。 |
| SYNCPT | すべての更新をコミットします。これらの更新がコミットされたあとは、それ以降の強制的な中止はこの更新に影響しません。 |

これらの機能の構文は、次のとおりです。

| 機能 | 呼び出し構文 |
|---|---|
| ROLLBACK | 77 ROLL-BACK PIC X(08) VALUE 'ROLLBACK'.<br>CALL 'kixvsam' USING ROLL-BACK. |
| SYNCPT | 77 SYNCPOINT PIC X(08) VALUE 'SYNCPT'.<br>CALL 'kixvsam' USING SYNCPOINT. |

次の制限事項は、これらの機能を呼び出したあとに適用されます。

- VSAM データセットに対して順次アクセスが行われた場合、現在のレコードポインタを閉じて、それを再び確立する必要があります。
- オープン出力データセットが書き込み中の場合は、これらの機能を呼び出さないでください。

1 つの COBOL プログラムが直接またはスクリプトによって実行されると、プログラムの終了時に同期点が発行されます。ただし、シェルスクリプトを使用して 1 つのジョブステップで複数の COBOL プログラムを実行する場合、そのプログラムは 1 つの実行単位とみなされ、ジョブステップが完了するまで同期点が発行されません。そのため、シェルスクリプトの停止時と同様に、各 COBOL プログラムの終わりで同期点が発行されるようにするには、STOP RUN 文ではなく GOBACK 文を使用して各 COBOL プログラムを停止する必要があります。

**注 –** Sun MBM のもとで実行している COBOL プログラムでは、GOBACK/STOP RUN は異なる動作をします。詳細については、『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』を参照してください。

GOBACK 文ではオープンしているファイルを自動的に閉じないので、GOBACK コマンドを実行する前に COBOL プログラムでそれらのファイルを閉じる必要があります。

# バッチジョブによる回復の使用法

バッチプログラムは、通常、実行時間が長くなります。回復が有効である場合、レコードを更新しているプログラムが同期点を発行するまでは、レコードはロックされたままになります。このため、同じレコードを更新するオンライントランザクションを実行するユーザーを長時間待たせることになります。

バッチプログラムの回復を有効にする場合、プログラムは定期的に同期点を発行してすべての更新をコミットし、更新済みのレコードに他のユーザーがアクセスできるようにする必要があります。回復を使用する場合、プログラムを再起動できるように記述する必要があります。すなわち、再起動時に、プログラムはすでにコミットされたレコードを更新しないようにします。

---

**注 –** 回復が有効な場合であっても、`OUTPUT` モードでオープンしているファイルに対する回復は無視されます。`SYNCPOINT` または `ROLLBACK` が発行されるとき、ファイルに対して特別な処理は行われません。

---

実行時間の長いプログラムは、大きな回復ファイルを作成する場合があります。領域は、変更前イメージを循環ファイルに維持します。ファイルの終わりに達すると、変更前イメージはファイルの始めから再び始まります。プログラムまたはトランザクションがまだ有効である場合、Sun MTP は変更前イメージが上書きされないことを保証します。それに代えて、有効な変更前イメージが上書きされそうになると、変更前イメージを作成したプログラムまたはトランザクションを終了します。これには、1 つのプログラムまたはトランザクションによって更新されるすべてのレコード、さらにその他の有効なプログラムおよびトランザクションによって更新されるレコードを保存するに十分な大きさの回復ファイルが必要です。

そのため、実行時間の長いバッチファイルでは、回復を無効にします。これを行うには 2 つの方法があります。使用する方法は、バッチジョブの実行環境によって異なります。

- ファイルまたは raw ファイルシステムデバイスを VCT で定義しないことで、領域に対するすべてのデータセットの回復を無効にします。変更を有効にするには、領域を停止して再起動します。
- `kixfile` コマンドで、特定のデータセットに対する回復を無効にします。`kixfile` コマンドは、FCT の個々のデータセットの回復設定を上書きします。`kixfile` については、『Sun Mainframe Transaction Processing 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。

バッチプログラムを起動する前に、更新されるファイルを保存します。バッチプログラムが正しく実行されると、保存したファイルを削除できます。プログラムがエラーで停止すると、保存したファイルから復元できます。

次の表に、バッチジョブの回復の手順をまとめます。

**表 15-7**　バッチジョブの回復手順

| プロセス | 回復の状態 | 手順 | |
|---|---|---|---|
| バッチジョブからのデータベースの更新 | オフ | データセットを保存します。データセットを保存する前に、`kixfile` を使用してデータセットをオンライン読み取り専用に設定します。(`kixfile -oY` *filename*)。あるいはバッチジョブによる排他的使用のためにそのデータセットをロックします (`kixfile -lY` *filename*)。<br>データセットを処理します。<br>ジョブが失敗した場合、保存したデータセットから復元します。 | |
| | オン | 同期点なし | 回復を有効に設定します。<br>データセットを処理します。<br>ジョブが失敗した場合、ファイルは Sun MTP によって回復されます。<br>実行時間の長いバッチジョブの最初の状態にデータセットを回復するには、長い時間がかかる場合があります。 |
| | | 同期点あり | 回復を有効に設定します。<br>データセットを処理します。<br>ジョブが失敗した場合、最後の同期点にファイルが復元されます。バッチジョブにより、最後の同期点からジョブを再開する前に環境が設定されます。 |
| バッチジョブからのデータベースの読み取り (更新なし) | 更新がない場合、回復ファイルは使用されません。 | | |

# バッチプログラムからの高速書き込みの実行

`FAST_UNIKIXVSAM` は、バッチプログラムからオープンしている VSAM ファイルに高速書き込みするオプションです。このオプションを使用するには、バッチシェルスクリプトまたは Sun MBM マクロジョブスクリプトで `FAST_UNIKIXVSAM` 環境変数を設定する必要があります。環境変数の値は、FCT で定義されているデータセット名 (複数可) です。出力用にオープンするデータセットを取り込むだけです。この環境変数は、バッチシェルスクリプトまたはマクロスクリプトの各ステップで設定する必要があります。

**注 –** このオプションは、順次ソート出力データセットに制限されています。データセットがランダム出力書き込みを行う場合、`$FAST_UNIKIXVSAM` を設定するときにこのデータセットを含めないでください。

各データセット名は、大文字で一意である必要があります。複数のデータセットは、それぞれの名前をコロンで区切ります。現在、このオプションでサポートされるオープンの出力データセットは 10 個までです。高速モードの出力では、KSDS 一時データセットだけオープンできます。

バッチシェルスクリプトに、次のように環境を設定します。

```
FAST_UNIKIXVSAM=ACCT1:ACCT2;export FAST_UNIKIXVSAM
```

Sun MBMでは、このオプションを指定する方法は、複数あります。

- マクロジョブスクリプトに次のように環境を設定します。

```
setenv FAST_UNIKIXVSAM ACCT1:ACCT2
```

- 下に示されているように、`SET` 文を使用して、MVS JCL で環境を設定してから、JCL を以下のように変換します。

```
//STEP1   SET  FAST_UNIKIXVSAM=ACCT1:ACCT2
```

- 以下のように、MVS JCL で Sun MBM `EBMSYSCMD` 文を使用して次のように環境を設定します。この場合、JCL を変換する必要があります。

```
//*EBMSYSCMD setenv FAST_UNIKIXVSAM ACCT1:ACCT2
```

- VSE JCL では、テクニックは類似しています。詳細は、Sun MBM のマニュアルを参照してください。

ジョブを実行すると、`Opened Dataset` *datasetname* `in fast mode` というメッセージが表示され、ログファイルに書き込まれます。このメッセージが表示されない場合、データセットは標準のモードでオープンします。

次のようなエラーメッセージが表示される場合があります。

`FastOpen Allocate failed = %n`

**説明**: `%n` = -1 の場合、データセットはすでにオープンしています。`%n` = -2 の場合、オープンデータセット数の限度 (10) を超過しています。このメッセージは、アプリケーションを実行するのに十分なメモリーがない場合にも表示されることがあります。

**対策**: オープンファイルの数を減らすか、処理メモリーを増やします。

`Mismatch of buffers = %n`

**説明**: バッファーが一致しません。

**対策**: この問題は、通常の操作では起こりません。この問題が発生した場合は、ご購入先に連絡してください。

# バッチ COBOL プログラムのデバッグ

**注 –** この節は、Sun MBM 以外の環境でだけ有効です。

Sun MTP は、バッチプログラムのデバッグ用に 2 つのツールを提供しています。どちらも、`unikixvsam` COBOL ランタイムプログラムのオプションです。

- `-d` オプションは、VSAM アクティビティーのトレースを出力します。
- `-D` オプションは、COBOL デバッガを有効化します。

## ▼ デバッグのプログラムをコンパイルする

- **使用する COBOL のブランドに適したコマンドを入力します。**
  - Server Express を使用する場合は、以下のように入力します。

```
$ cob -ia -C ibmcomp program.cbl
```

  - ACUCOBOL-GT を使用する場合は、以下のように入力します。

```
$ ccbl -CA -Ga options program.cbl
```

## ▼ VSAM トレース機能を使用してデバッグする

- 次のコマンドを入力します。

```
$ unikixvsam -d program
```

VSAM トレース機能は、トレースの出力を $KIXSYS/unikixmain.dbg ファイルに送信します。

次の例は、トレース出力ファイルを示します。

```
11/03/2001 10:59:38 unikixvsam :
11/03/2001 10:59:38 unikixvsam :A VSAM OPEN was issued for file  DWORK
11/03/2001 10:59:38 unikixvsam : The command had a status of 00
11/03/2001 10:59:38 unikixvsam :
11/03/2001 10:59:38 unikixvsam :A VSAM WRITE was issued for file  DWORK
11/03/2001 10:59:38 unikixvsam : The relative key is 1
11/03/2001 10:59:38 unikixvsam : The command had a status of 00
11/03/2001 10:59:38 unikixvsam :
11/03/2001 10:59:38 unikixvsam :A VSAM CLOSE was issued for file  DWORK
11/03/2001 10:59:38 unikixvsam : The command had a status of 00
11/03/2001 10:59:38 unikixvsam :
```

この出力ファイル例は、COBOL プログラムのファイルアクセス動作を示しています。ファイルに対して VSAM 呼び出しを行うたびに、ファイル名、呼び出しのタイプ、使用したキー、およびアクセスの結果が記録されます。プログラムが大量の呼び出しを行う場合、トレースファイルは急速に大容量になります。利用できるディスク容量をすべて消費しないように十分注意してください。実稼動環境で VSAM トレース機能を使用しないでください。

## ▼ COBOL Source Debugger を使用してデバッグする

**注 –** この手順は、Sun MBM 以外の環境でだけ有効です。

1. **プログラムをコンパイルします。**

   デバッガで使用できるオプションについては、COBOL ベンダーのマニュアルを参照してください。

2. **バッチプログラムを実行するために、** `-D` **オプション付きの** `unikixvsam` **を使用するシェルスクリプトを作成します。**

   たとえば、次の行を含む `dbbat` という名前のシェルスクリプトを作成します。

   ```
   unikixvsam -D BATCH001
   ```

   シェルスクリプトには、このような文を複数含めることができます。

3. **シェルスクリプトを実行可能にします。**

   ```
   $ chmod 777 dbbat
   ```

4. **この領域を起動します。**

5. `CEDF`、`COBOL` **トランザクションを実行します。**

   COBOL デバッガの起動に関する詳細は、301 ページの「デバッグ機能を起動する」を参照してください。

6. **ブランクのトランザクション画面で、** `CBCH` **トランザクションを使用してバッチプログラムを実行します。**

   ```
   CBCH dbbat
   ```

7. **プログラムをデバッグします。**

8. **デバッグが終了したら、** `CEDF`、`OFF` **トランザクションを実行して、デバッガ表示をオフにします。**

   シェルスクリプトの実行時、`unikixvsam` が `-D` オプションで呼び出されるたびに、デバッガは自動的に開始します。

   COBOL デバッガの使用法の詳細は、ベンダーのマニュアルを参照してください。

   詳細は、311 ページの「COBOL デバッガの使用」も参照してください。

# 第16章

# 外部プログラムからの VSAM ファイルへのアクセス

この章では、Sun MTP による C-ISAM インタフェースのサポート、および領域リソースとして定義されていない C プログラムの使用法について説明します。次の節によって構成されます。



## C-ISAM インタフェース

C-ISAM は、C 言語バッチおよびオンラインプログラムが領域の VSAM ファイルにアクセスできるようにするためのインタフェースです。この節では、次の事項について説明します。

- サポートされているすべての C-ISAM 機能のリスト
- オンラインおよびバッチ環境で C-ISAM を使用するときの注意事項
- Sun MTP が回復およびロールバックを処理する方法についての情報
- C-ISAM 標準で実装されている機能と異なる機能
- エラーコードおよびエラーコード処理

---

**注 –** C-ISAM インタフェースの使用については、『C-ISAM Programmer's Manual』を参照してください。

---

# サポートされている機能

C-ISAM 機能は、固定長および可変長の両方のレコードをサポートしています。

**表 16-1**　サポートされている C-ISAM 機能

| 機能 | KSDS | RRDS | ESDS |
|---|---|---|---|
| `isopen(filename, ISINPUT);` | OK | OK | OK |
| `isopen(filename, ISOUTPUT);` | OK | OK | OK |
| `isopen(filename, ISINOUT [+ISAUTOLOCK\|+ISMANULOCK]);` | OK | OK | OK |
| • `filename` は、Sun MTP に認識されている、8 文字までのデータセット名である必要があります。Sun MTP は、データセット名の、フルパス名およびファイル名へのマッピングを処理します。 | | | |
| `isclose(isfd);` | OK | OK | OK |
| `isstart(isfd,keydesc,length,rec,ISEQUAL);` | OK | OK | E102 |
| `isstart(isfd,keydesc,length,rec,ISGREAT);` | OK | OK | E102 |
| `isstart(isfd,keydesc,length,rec,ISGTEQ);` | OK | OK | E102 |
| `isstart(isfd,keydesc,length,rec,ISFIRST);` | OK | OK | E102 |
| • `keydesc` は必要な引数ですが、VSAM ファイルの場合、Sun MTP で無視されます。これは、Sun MTP によって自動的に処理されます。 | | | |
| `isread(isfd, record, ISNEXT [+ISLOCK]);` | OK | OK | OK |
| `isread(isfd, record, ISPREVIOUS [+ISLOCK]);` | OK | NOT OK | E102 |
| `isread(isfd, record, ISEQUAL [+ISLOCK]);` | OK | OK | E102 |
| `isread(isfd, record, ISGTE [+ISLOCK]);` | OK | OK | E102 |
| `isread(isfd, record, ISGREAT [+ISLOCK]);` | OK | OK | E102 |
| `isread(isfd, record, ISFIRST [+ISLOCK]);` | OK | OK | E102 |
| `iswrite(isfd, record);` | OK | OK<br>注を参照 | OK |
| `isrewrite(isfd, record);` **Using `KEY` | OK | OK | E102 |
| `isrewcurr(isfd, record);` | OK | OK | OK |
| `isdelete(isfd, record);` **Using `KEY` | OK | OK | E102 |
| `isdelcurr(isfd);` | OK | OK | E102 |
| `isrelease(isfd)` | OK | OK | OK |
| `isindexinfo(isfd, buffer, idx_number);` | OK | OK | OK |

**制限** – データセット名ごとに 1 つの索引のみを使用できます。代替索引をサポートするには、同じファイルにアクセスしている場合でも、各索引に異なるデータセット名を指定します。ファイルは、要求索引を表すデータセット名を使用してオープンされ、`isindexinfo` コマンドが `isfd` を使用して実行されます。

表 16-1　サポートされている C-ISAM 機能 (続き)

| 機能 | KSDS | RRDS | ESDS |
|---|---|---|---|
| `isindexinfo(isfd, buffer, 0);` | OK | OK | OK |

**制限** – キーの数 (VSAM に対して 0 または 1 のみが有効) およびデータセットの最大レコードサイズをレポートします。「`indexsize`」フィールドにゼロが返され、「`nrecords`」フィールドにファイルおよびスパンファイルの数 (1 ～ 8) が返されます。これは、Sun MTP が索引ノードという概念を直接サポートしていないためであり、また、そのレコード数へアクセスできないためです。

---

**注 –** RRDS ファイルの場合のみ − C-ISAM インタフェースでは、新しいレコードに使用するレコード数を指示するためにレコードへの書き込みを行う前に、グローバル変数 `isrecnum` をプログラムで設定するようには指定しません。VSAM はこの値を必要とします。あるいは、空のレコードの直前にそのファイルが配置される必要があります。

C-ISAM インタフェースユーザーは、グローバル変数 `isrecnum` を要求レコード数に設定するか、レコード数をファイルの適切な場所に配置して `isrecnum` をゼロに設定する必要があります。

---

# バッチおよびオンラインの注意点

Sun MTP C-ISAM インタフェースは、次の 2 つの環境のいずれかに実装して使用します。

- **オンライン**: 親プロセスを通して実行される Sun 以外のソフトウェア。
- **バッチ**: Sun MBM または Sun MTP 標準バッチ機能を使用して実行されるすべてのバッチプログラム。第 15 章を参照してください。

ほとんどの機能では、ユーザーはプログラムを実行する環境について考慮する必要はありません。ただし、`isbegin`、`iscommit`、および `isrollback` を正しく使用するには、次の要因について考慮する必要があります。

- オンライン
  - クエリまたはクエリ/更新は、トランザクションの範囲外であると見なされます。
  - クエリ/更新の間にファイルはオープンしたままになるので、`iscommit` および `isrollback` を使用できます。最後のファイルがクローズすると、C-ISAM インタフェースによって `COMMIT` が暗黙的に実行され、変更が適用されます。クエリの最後のファイルがクローズしたあとのロールバックは許可されません。最後のファイルがクローズしたあとで `isrollback` または `iscommit` が使用された場合、エラーが発生します。
  - `isbegin` 機能は使用できるが、これはファイルには影響しないで、通常の完了状態が通知されます。

- バッチ
  - プログラム (バッチ環境で実行する、クエリまたはクエリ/更新) は、トランザクションの範囲内で実行していると見なされます。インタフェースは暗黙的に isbegin を実行します。
  - ユーザーは isbegin を実行する必要はないが、isbegin を実行することはできます。
  - iscommit および isrollback は、すべてのファイルがクローズしたあとでも実行できます。バッチジョブがいったん停止すると、iscommit がユーザーによって実行された場合でも、Sun MTP によって暗黙的に COMMIT が実行されます。

---

**注 –** COMMIT が暗黙的に有効になる前に、1 つのバッチジョブが複数のステップ (プログラム) を実行することが可能です。

---

# 回復および C-ISAM

C-ISAM の使用時には、Sun MTP が回復およびロールバックを処理します。そのため、C-ISAM ログファイルを作成または使用する必要はありません。C-ISAM インタフェースは、islogopen および islogclose への呼び出しを含む、Sun 以外のソフトウェア用の機能を提供します。これらの機能は、機能が呼び出されると完了コードとともに返ります。

Sun MTP は、変更後イメージを使用したロールフォワード回復はサポートしません。

# C-ISAM 標準と異なる機能

C-ISAM インタフェースは、すべての Sun MTP 要件に準拠しています。C-ISAM インタフェースでは、領域が実行されている必要があります。データファイルは領域内で定義および作成されるので、iserase および isbuild などの機能は有効ではありません。

isbuild 機能は、次のように実装されます。

```
int isbuild(filename, reclen, buffer, keydesc, mode)
```

この機能は、渡された filename および mode 引数を渡して isopen を呼び出します。ほかの 3 つの引数は無視されます。この呼び出しは、通常の isopen 機能のように処理されます。filename がすでに存在していて、領域で定義されている必要があります。

次の機能が実装されていて、SUCCESSFUL 状態を返します。ただし、状態を返すこと以外は何も行いません。

```
iserase      islogopen
isbegin      islogclose
islock       isunlock
```

## Sun MTP 固有の機能

Sun MTP は、C-ISAM 標準ではサポートしていない次の 2 つの機能をサポートしています。

```
int isgetpath(char *buffer, int size, char*dataset)
```

dataset 名を受け取り、そのデータセットに対するファイル名を含むフルパス名を指定のバッファーに返します。バッファーサイズは、size 引数で指定されている必要があります。エラーが発生すると、-1 の値が返されます。グローバル変数 iserrno には、特定のエラーコードが入ります。データセットが Sun MTP 領域の一部でない場合、iserrno には値 111 (ENOREC) が入ります。バッファーがフルパス名を処理するのに十分な大きさでない場合、iserrno に値 98 (MAXFILESEXC) が返されます。

```
int isconn()
```

通常の open/close 機能が呼び出される前に Sun MTP 領域の共有メモリーに接続する必要のある、Sun 以外のソフトウェアのみによって使用されます。バッチジョブでのみ利用可能です。

オンラインジョブがこの機能を実行しようとすると、-1 が返され、iserrno には値 95 (BTCHONLY95) が入ります。

この機能は、プロセスで 1 度のみ呼び出すことができます。すべての追加呼び出しは値 -1 を返し、iserrno には値 1 が入ります。

# エラーコード

C-ISAM 機能は、整数コードのみを返します。0 以上の値のリターンコードは、機能が正常に実行されたことを示します。リターンコードが負数の場合、機能の実行に失敗したことを示します。失敗の原因を判定するには、グローバルエラー変数 `iserrno` を調べます。

返されるエラーには、次の 4 つのタイプがあります。

1. ファイルが存在しない場合のエラーコード `ENOENT`
2. Sun MTP システムエラー (コード 90 ～ 99)

**表 16-2** C-ISAM/Sun MTP インタフェースエラー

| Sun MTP システムエラー | コード | 説明 |
|---|---|---|
| `#define SNDMSG90` | 90 | TP サーバーを解放するためのメッセージを送るときのシステムエラーです。 |
| `#define RCVMSG91` | 91 | Sun MTP から `ACK` メッセージを受け取るときのシステムエラーです。 |
| `#define STRTCISAM` | 92 | C-ISAM は VSAM ファイルに対する Sun MTP の制御下で実行される必要があります。 |
| `#define SNDMSG93` | 93 | 初期メッセージのシステムエラー。Sun MTP は有効ではありません。 |
| `#define RCVMSG94` | 94 | Sun MTP から `XUANUM` メッセージを受け取るときのシステムエラーです。 |
| `#define BTCHONLY95` | 95 | 機能はバッチプロセスのみで利用可能です。 |
| `#define VERSIONERR` | 96 | C-ISAM と Sun MTP のバージョンが適合しません。 |
| `#define FCBERR` | 97 | システムエラー。`valfcb` 呼び出し時の FCB のエラーです。 |
| `#define MAXFILESEXC` | 98 | ファイル最大数の超過 |
| `#define INVALIDIX` | 99 | `isindexinfo` 呼び出し時の無効な索引番号です。 |

3. 『C-ISAM Programmer's Manual』で定義されている C-ISAM エラーコード (100 ～ 171) に直接対応する VSAM エラーコード。EOF、No record found matching Key などの制御エラーが C-ISAM エラーコードに対応します。

**表 16-3** VSAM エラー

| VSAM エラー | コード | 説明 |
|---|---|---|
| #define EDUPL | 100 | 重複レコード |
| #define ENOTOPEN | 101 | 指定されたファイルが見つかりません |
| #define EBADARG | 102 | 無効な引数です。この回復タイプではサポートされていません |
| #define EENDFILE | 110 | ファイルの終わりに達しました |
| #define ENOREC | 111 | レコードが見つかりません |
| #define ENOCURR | 112 | 現在のレコードが存在しません |
| #define JOBSEXCEED | 129 | 対話型 C-ISAM ジョブ数の超過 |

4. C-ISAM エラーに対応しない VSAM エラーコードは、Micro Focus File Status Codes として定義されている VSAM エラーコードと同じ整数で返されます。

**例**: iserrno で返されるコードが、エラーコード 2 とします。このエラーコードは、代替キーデータセットであるデータセットの read next に返されます。この代替キーデータセットに対しては、同じ重複キーを持つほかのレコードが少なくとも 1 つ存在します。

レコードは正しく返されますが、-1 のリターンコードが返されます。さらに iserrno には、重複キーを持つレコードがこの代替キーデータセットに存在することをユーザーに警告するために 2 が入ります。

これは、ユーザーが read next でファイルの終わりに達したときの論理と同じです。

---

**注 –** COBOL 変数 isstat1、isstat2、isstat3、および isstat4 は、COBOL プログラムでだけ使用されるので、設定されていません。

---

# C プログラムを使用した VSAM ファイルへのアクセス

この節では、CICS API を使用しない C プログラムを使用して、VSAM ファイルにアクセスする方法について説明します。これらの C プログラムは、領域内でリソースとして定義されていません。

Sun MTP は、C-ISAM インタフェースを使用して VSAM ファイルにアクセスする C 言語プログラムの実行をサポートします。これらの C プログラムを実行する前に、それらをコンパイルする必要があります。プログラムをコンパイルしたシステムと同じバージョンの Sun MTP を実行しているシステムでだけそのプログラムを実行できます。

詳細は、358 ページの「サポートされている機能」を参照してください。

## ▼ C 言語プログラムをコンパイルする

1. **このプログラムを実行する Sun MTP の環境に** `UNIKIX` **環境変数を設定します。**
2. `isam.h` **ヘッダーファイルを C プログラムに取り込みます。**
   このファイルは、`$UNIKIX/src/CICS_structures` ディレクトリにあります。
3. **コマンド行またはスクリプトファイルで次のコマンドを実行し、プログラムをコンパイルして実行可能プログラムを作成します。**

   **Solaris プラットフォーム**:

```
$ cc input-filename.c -o output-filename $UNIKIX/lib/libbcisam.a -lsocket
```

   **AIX プラットフォーム**:

```
$ cc input-filename.c -o output-filename $UNIKIX/lib/libbcisam.a -lcurses -brtl
-L$UNIKIX/lib -lkxtables
```

4. **ファイルの実行権があることを確認します。**

## ▼ C プログラムを実行する

1. **領域の VCT の「Maximum query jobs」フィールドが 1 以上の値であることを確認します。**

   この値は、同時に実行できるプログラムの数を示します。

2. **コマンドプロンプトで、KIXCISAM 環境変数に C 言語の実行可能プログラムのフルパス名およびファイル名を設定します。**

   たとえば、次のように設定します。

```
$ export KIXCISAM=/finance/src/cprogs/myprog
```

3. **プログラムがアクセスする VSAM ファイルがある領域を開始します。**

4. **コマンドプロンプトで、kixcisam ユーティリティーを実行します。**

```
$ kixcisam
```

   このユーティリティーは、KIXCISAM 環境変数で定義されているプログラムを実行し、その実行を監視します。kixcisam の詳細は、『Sun Mainframe Transaction Processing 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。

# 用語集

## 数字

**3270 SNA デバイス**　(名詞) IBM SNA 3270 データストリームを表示するターミナルデバイス。

## A

**ABEND**　(名詞) `EXEC CICS` コマンドを使用して、タスクを異常終了する。

**APPC**　(名詞) 拡張プログラム間通信機能。LU6.2 アーキテクチャーのプレゼンテーションサービス層の SNA プロトコル境界。APPC は一般的に、LUTYPE6.2 の同義語として使用される。

## C

**COMMAREA**　(名詞) 通信領域。指定した端末と通信するタスク間で、データの受け渡しに使用される領域。領域を使用して、タスク内のプログラム間でデータを受け渡すこともできる。

# E

EBCDIC　(名詞) 拡張 2 進化 10 進コード。多くのデータ処理システム、データ通信システム、および関連装置で情報交換に使用される、8 ビット符号化文字から構成された符号化文字セット。

exec interface block (EIB)　(名詞) CICS プログラム内の各タスクに関連する制御ブロック。EIB には、アプリケーションプログラムの実行中に役立つ情報 (トランザクション識別子など) と、プログラムのデバッグのためにダンプを使用する際に役立つ情報が収められる。

# F

File Manager　(名詞) VSAM カタログでの VSAM ファイルの定義に使用される。ファイルタイプ、キーの長さ、サイズなどのファイルの属性を定義できる。

# G

GUI　(名詞) グラフィカルユーザーインタフェース。

# J

JCICS　(名詞) Java プログラミング言語で記述されたアプリケーションによる CICS API へのアクセスを可能にする IBM 提供の Java クラス。

# L

LU6.2　(名詞) 分散処理環境のプログラム間での一般的な通信をサポートする論理ユニットタイプ。

LUTYPE2/LUTYPE3 (名詞) ログインユニットのタイプ。LUTYPE2 は、3270 ディスプレイとの通信に使用される。LUTYPE3 は、3270 プリンタにデータを送信するために使用される。メインフレーム CICS 関連の用語。

# M

`make` 機能 (名詞) Sun MTP システムの再構築に使用されるユーティリティー。

# P

POSIX (名詞) ポータブルオペレーティングシステムインタフェース。UNIX オペレーティングシステムに基づく標準オペレーティングシステムインタフェースのセット。POSIX インタフェースは IEEE を中心に開発された。

# S

SOSI フィールド (名詞) SOSI (Shift-Out, Shift-In) フィールドは、シングルバイトとダブルバイト文字両方を含んだ 3270 フィールド。SOSI フィールドのすべてのダブルバイト文字は、SOSI 文字で挟まれている必要がある。

SQL (名詞) 構造化照会言語。一連の情報へのアクセスと更新に使用されるリレーショナルデータベース言語。

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、バッチ生産負荷を処理し、開始時刻やバッチプロセスの最大数、およびジョブの優先順位といった割り当てられたパラメータによってジョブをスケジューリングする。

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。クライアント以外の Sun MTP のすべてのコンポーネントは、メインサーバープロセスである `unikixmain` によって起動する。

**Sun MTP グループ** (名詞) 特定のアプリケーションのテーブルファイルセット。ファイルはファイルシステムの単一のディレクトリに配置されている。ディレクトリは GCT にリストされる。

**Sun MTP シェルスクリプト** (名詞) `$UNIKIX/bin` に配置されたシェルスクリプトのユーティリティープログラム。

**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。

# T

**Table Manager** (名詞) Sun MTP テーブルで領域のリソースの定義に使用する Sun MTP 機能。

**TCP/IP** (名詞) インターネットの基礎となるネットワークプロトコル群。伝送制御プロトコル (TCP) は、信頼性の高い全二重データストリームを提供するプロトコル。インターネットプロトコル (IP) は、TCP のパケット配信サービスを提供するプロトコル。TCP プロトコルは、ユーザープロセスではなく、IP と連携する。

**TCTUA** (名詞) 端末ユーザー領域。端末に関連するトランザクション間で、データの受け渡しに使用される。

**TN3270 サーバー (`unikixtnemux`)** (名詞) Sun MTP で、TCP/IP -TN3270 プロトコルを使用して、PC、Macintosh、および UNIX システムで実行する 3270 エミュレータのサポートを有効にする。TN3270E もサポートする。

**TN3270 プロトコル** (名詞) 従来の TCP/IP Telnet プロトコルの拡張で、ASCII 以外の文字、IBM-3270 などのブロックモードデバイス、Sun MTP などのアプリケーションで、TCP/IP を介した通信を可能にする。TN3270E も含まれる。

# U

**`unikixdcl` サーバー** (名詞) DCL プロトコルスタックを使用する SNA サーバーに対する遠隔接続の数および状態を監視する Sun MTP サーバー。

**`unikixmain` サーバー** (名詞) Sun MTP メインサーバープロセス。

**`unikixqm` サーバー** (名詞) TCP/IP 接続の数と状態を監視する Sun MTP サーバー。

unikixrc.cfg ファイル
(名詞) unikixdcl、unikixqm、および unikixtnemux サーバーについての情報を収めたリソースファイル。起動時に、Sun MTP コミュニケーションマネージャーは、unikixrc.cfg ファイルを読み取り、該当するサーバーを起動する。

unikixtcp サーバー
(名詞) TCP/IP 接続の数と状態を監視する Sun MTP サーバー。

unikixtnemux サーバー
(名詞) TN3270 サーバープロセス。「TN3270 サーバー」を参照。

# V

VSAM クラスタ
(名詞) VSAM 構造での最上位の命名。クラスタの名前は主アクセス名。クラスタには、データ定義、索引定義、および任意の二次索引も含まれる。

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。

VSAM データセット
(名詞) VSAM 規則に従って編成、格納、およびアクセスされる関連データの名前付き集合。

VSAM ファイル
VSAM データセットを参照。

# あ

アカウンティング
(名詞) ユーザーのアカウント情報を体系的に収集、記録、解釈、および表示する方法。

アカウンティングジャーナル
(名詞) 関連付けられたジャーナルのアカウンティングレコードを Sun MTP が書き込むファイル。ジャーナルファイル名は物理ファイル名に対応する。

オブジェクトの所有者が任意の ACL をコントロールする。

アクティビティーカウント
(名詞) Sun MTP が保全性の管理に使用する方法。アクティビティーカウントは、各 VSAM ファイルヘッダーに置かれ、領域がファイルを開閉するたびに増加する。

宛先管理テーブル (DCT)
(名詞) 一時データコマンドで処理される、宛先の名前またはキューを含んだ Sun MTP テーブル。テーブルは、これらのキューに割り当てられた特性の定義や管理に使用される。

**アプリケーション プログラミング インタフェース (API)** (名詞) アプリケーションプログラムで使用される事前定義のインタフェース。API はルーチン名とルーチンの引数から構成され、関連付けられたアプリケーションプログラム言語の構文に従う。

# い

**一時記憶域テーブル (TST)** (名詞) ローカルおよび遠隔の一時記憶域キューの記憶域と回復を定義する Sun MTP テーブル。

**インターネット プロトコル (IP)** TCP/IP を参照。

# か

**外部プレゼンテーション インタフェース (EPI)** (名詞) CICS 以外のアプリケーションプログラムが、1 つ以上の標準 3270 端末として Sun MTP を表示することを可能にするプログラムを作成するための API。EPI アプリケーションは、実際の 3270 端末のように Sun MTP と通信する。

**外部呼び出し インタフェース (ECI)** (名詞) DPL の規則に従いサーバーで実行中の CICS プログラムを、CICS 以外のアプリケーションプログラムが呼び出すことを可能にするプログラムを作成するための API。

**会話型トランザクション** (名詞) ユーザーとの会話 (通常、`SEND/RECEIVE` シーケンス) を続けながら実行するトランザクション。

**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)。レコードは順次に記録され、アクセスされる。

RRDS (相対レコードデータセット)。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る。

KSDS (キーシーケンスデータセット)。レコードは索引またはキーによって検索される。

**仮想通信アクセス方式 (VTAM)** (名詞) 通信を制御し、SNA ネットワーク内のデータの流れを制御するプログラム。

**カタログファイル** (名詞) VSAM データセットの名前と情報を含むファイル。

**画面生成ユーティリティー (SGU)** (名詞) 開発者がユーザーインタフェースの画面を描き、コンパイルされた基本マッピングサポート (BMS) マップを作成することを可能にするユーティリティー。そのあと、このユーティリティーを使用して、開発者は画面メニューやデータエントリ画面を定義したり変更したりできる。SGU は、マップセット、マップセット内の複数のマップ、およびマップ内のさまざまなフィールドの定義を提供する。画面属性 (数値や明るさなど) もフィールドごとに指定される。

**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する。

# き

**キーシーケンスデータセット (KSDS)** (名詞) キーによって参照される可変長レコードの索引付き VSAM ファイル。

**機能シップ** (名詞) アプリケーションプログラムに透過的なプロセス。リソースが他の CICS システムに実際に配置されている際に、このプロセスを使って CICS はそのリソースにアクセスする。

**基本マッピングサポート (BMS)** (名詞) データストリームを端末とやり取りする機能。入出力表示データをフォーマットする。BMS マクロ命令は Sun MTP BMS アセンブラで使用され、物理および記号定義のマップファイルを作成する。

**共通作業領域 (CWA)** (名詞) システム内のタスクからのアクセスが必要なユーザーデータに対して、アプリケーションプログラムによって使用可能なタスク共有プール内の領域。

# く

**クラスタ** VSAM クラスタを参照。

**グループ** (名詞) 特定のアプリケーションのテーブルファイルセット。ファイルはファイルシステムの単一のディレクトリに配置されている。ディレクトリは GCT に定義される。

**グループ管理テーブル (GCT)** (名詞) グループを定義する Sun MTP テーブル。各グループは、特定のアプリケーションの情報を含んだファイルシステムのディレクトリを定義する。

## こ

**顧客情報管理システム (CICS)** (名詞) ユーザー作成のアプリケーションプログラムによって、遠隔端末で入力されたトランザクションの並行処理を可能にする IBM の使用許諾を受けたプログラム。データベースの構築、使用、維持の各機能が含まれる。

## さ

**サインオンテーブル (SNT)** (名詞) Sun MTP トランザクションを使用する認証されたユーザーのリストを含んだ Sun MTP テーブル。

**索引ファイル** (名詞) ブロック番号とそのブロックでの最上位のキーで構成されるキーポインタが収められる。キーはレコードを指す。

## し

**システム間通信 (ISC)** (名詞) TCP/IP や SNA ネットワーキング機能、または SNA アクセス方式のアプリケーション間機能を使った別個のシステム間の通信。

**システムサービス制御ポイント (SSCP)** (名詞) 構成管理、ネットワークオペレータ機能および問題判定要求の調整、およびネットワークのエンドユーザーに対するディレクトリサポートやその他のセッションサービスの提供のための、SNA ネットワーク内の制御点。

**システム初期化テーブル (SIT)** (名詞) システムの初期化情報を含み Sun MTP システム名を識別する Sun MTP テーブル。

**システムネットワーク体系 (SNA)** (名詞) 情報単位を伝達し、ネットワークの構成と動作を制御するための論理構造、形式、プロトコル、および操作順序。

**ジャーナル管理テーブル (JCT)**
(名詞) トランザクションが、他のテーブルで参照されている 1 つ以上のジャーナルファイルを書き込むことができるかを指定する Sun MTP テーブル。ジャーナルはアカウンティングデータの書き込みにも使用される。

**処理プログラムテーブル (PPT)**
(名詞) Sun MTP トランザクションが参照するアプリケーションプログラムと BMS マップセットをリストする Sun MTP テーブル。

**シングルバイト文字セット (SBCS)**
(名詞) 文字ごとに 1 バイトを必要とする言語スクリプトはシングルバイト文字セット (SBCS) と呼ばれる。たとえば、英語、スペイン語、およびフランス語がシングルバイト文字セット (SBCS) である。

## す

**スパンファイル**
(名詞) 複数のファイルシステムに渡ってセグメント化されるファイル。

## せ

**セグメント**
(名詞) スパンファイルの一部分。スパンファイルを参照。

## そ

**相対レコードデータセット (RRDS)**
(名詞) VSAM データセットで、そのレコードはデータセット内で占める位置番号で検索される。

**ソケット**
(名詞) 異なるネットワークプロトコルを使用可能にするプロセス間通信の仕組み。

# た

**ダブルバイト文字セット (DBCS)** (名詞) 一部の言語スクリプトは、文字の表現に 2 バイトを必要とする。そのスクリプトはダブルバイト文字セット (DBCS) と呼ばれる。たとえば、日本語、中国語、および韓国語がダブルバイト文字セット (DBCS) である。

**ダンプファイル機能** (名詞) Sun MTP からディスクファイルやシステムプリンタに、VSAM データセットのレコードの一部またはすべてを書き込む機能。

**端末管理テーブル (TCT)** (名詞) 端末、プリンタ、および遠隔システム接続の識別情報を収めた Sun MTP テーブル。

# つ

**追跡機能** (名詞) `kixdump` コマンドを使用してアクセスされるメモリーに追跡エントリを作成する。デバッグに使用される。

# て

**データエントリ画面** (名詞) テーブルデータの追加、変更、または削除を行う Sun MTP 画面。データエントリ画面は 1 つ以上のエントリとサブエントリ画面を持つ場合がある。

**データセグメント** (名詞) クラスタのデータ部分を収めたファイル。

**データセット** VSAM データセットを参照。

**データファイル** (名詞) レコードを収めた 1 つまたは複数のデータブロックから構成される。

**データファイルエディタ** (名詞) VSAM データセットの構築、変更、またはダンプが可能な Sun MTP メニュー。

**データ変換テンプレートテーブル (CVT)** (名詞) あるコードセットから他のコードセットに (たとえば、EBCDIC から ASCII に) データ変換を行うテンプレートを定義する Sun MTP テーブル。ある環境から別の異機種システム混在環境に、データを転送する際に必要となる。

**伝送制御プロトコル (TCP)** TCP/IP を参照。

# と

**同期点** (名詞) アプリケーションプログラムの実行での論理点。この論理点で、プログラムによるデータベースの変更は、整合性があり完全で、データベースにコミットが可能。この点まで持続した出力は宛先に送信され、入力はメッセージキューから削除され、他のアプリケーションでのデータベースの更新が可能になる。プログラムが異常終了した場合、回復機能と再起動機能によって、更新は前回完了した同期点の前にバックアウトしない。

**特別メニュー** (名詞) 単一テーブルのさまざまなエントリタイプ用のメニューとして使用されるメニュー。また、グローバルなテーブル操作を行う。

**トランザクションイニシエータサーバー (unikixtrin)** (名詞) Sun MTP サーバーシステムのエージェントとして動作し、必要な Sun MTP メッセージキューにメッセージを配置する。トランザクションサーバーと開始サーバーが、出力メッセージ用に同じデータグラム機構を使用することにより、遠隔クライアントを通して直接対話できるようになる。ひとつの unikixtrin プロセスは、遠隔の 3270 デバイスクライアントすべてをサポートする。

**トランザクションクラステーブル (TXC)** (名詞) 領域に定義されたトランザクションクラスに関する情報を含んだ Sun MTP テーブル。

**トランザクション経路指定** (名詞) Sun MTP または CICS 領域と接続した端末による、同じあるいは異なるシステム上の他の Sun MTP または CICS 領域でトランザクションの実行を可能にする。

# に

**入力順データセット (ESDS)** (名詞) データレコードを入力順に格納する可変長 VSAM ファイル

# は

**パーティション外キュー** (名詞) 「DCT –Extrapartition Destinations」画面で識別されるキューに書き込まれるすべてのデータを収めた順編成ファイル。ファイルは、エントリで指定したレコード形式と長さで開かれる。

# ひ

**非同期処理** (名詞) プロセスの継続中に、メインフレームによる Sun MTP 領域でのトランザクション開始を可能にする、あるいは Sun MTP 領域によるメインフレームでのトランザクション開始を可能にする双方向のプロセス。

**標準メニュー** (名詞) あるメニューから次のメニューまたはデータエントリ画面へのナビゲートに使用されるメニュー。「Table Manager 」メインメニューなど。標準メニューでは、データの要求または変更は行われない。

# ふ

**ファイル管理テーブル (FCT)** (名詞) Sun MTP アプリケーションプログラムがアクセスする VSAM データファイルについての情報を収めた Sun MTP テーブル。各ファイルには一連の特性が関連付けられる。この特性を Sun MTP のコマンドルーチンで使用して、アプリケーションプログラムが指定したコマンドを検査し実行する。

**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る。

**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する。

**不正終了** (名詞) タスクの異常な終了。アプリケーションは、`EXEC CICS ABEND` コマンドを実行してタスクを異常終了させることができる。異常終了と同じ意味。

**物理ファイル** (名詞) 物理データファイル。テープ、ディスク、CD-ROM などの媒体に格納された文字列またはバイナリデータ。

**プログラム管理テーブル (PCT)** (名詞) Sun MTP でトランザクションの識別と初期化に使用する制御情報を収めた Sun MTP テーブル。

**プログラムリストテーブル (PLT)** (名詞) システムの起動時、ユーザーの起動時、またはシステムの停止時に Sun MTP によって自動的に開始されるプログラム名を収めた Sun MTP テーブル。

**分散トランザクション処理 (DTP)** (名詞) システム間または領域間リンク上で互いに同期通信し合うトランザクション間での処理の配布。

**分散プログラムリンク (DPL)**
(名詞) 領域のプログラムが他の領域のプログラムに同期リンクするシステム間通信の方法。

# ま

**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る。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` と入力する。

# も

**モニター管理テーブル (MCT)**
(名詞) 領域で有効な状態にあるアカウンティングのデフォルトタイプ (トランザクションとユーザーロギング) を制御する Sun MTP テーブル。このテーブルには、アカウンティングを制御するフラグが付いた単一のエントリが含まれます。

# り

**領域**
Sun MTP 領域を参照。

# 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関連のファイルが入っている。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 ( / ) をつけて、参照される。

# ろ

**論理ファイル**　(名詞) 物理ファイルのデータを意味のある情報として定義づけるファイル。その定義には、フィールド名、フィールド長、レコード名、レコード長、およびブロック長が含まれる。

**論理ユニット (LU)**　(名詞) SNA で、エンドユーザーが SNA ネットワークにアクセスして別のエンドユーザーと通信するためのポートまたはエンドユーザーがシステムサービス制御点 (SSCP) によって提供される機能にアクセスするためのポート。

# 索引

## B



## C

## D

## E

## F



## G

## H



## I

## J



## K

## L

## M

## N



## O

## P

## Q



## R

## S

## T

## U

## V



## W

## X



## Y



## あ



## い



## え



## お



## か

## き

## へ



## ま