# Sun™ Mainframe Transaction Processing ソフトウェア 管理者ガイド

Release 8.1.0

Sun Microsystems, Inc.
www.sun.com

Part No. 819-2514-10
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 Administrator's Guide*
Part No: 817-7431-10
Revision A

# 目次

# 図目次

# 表目次

# コード例

# はじめに

このマニュアルでは、Sun™ Mainframe Transaction Processing ソフトウェア (Sun MTP)、Release 8.1.0 の管理機能について説明します。また、アプリケーションを実行するためにシステムを調整する機能についても説明します。このマニュアルは、ご使用のオペレーティングシステムと IBM CICS 環境の構成について十分な知識を持つ Sun MTP システム管理者またはシステムプログラマを対象としています。また、アカウンティングおよび VSAM ファイル管理についての情報を提供ます。

新しい機能と変更点については、『ご使用にあたって』で説明しています。

外部セキュリティー管理システム Sun Mainframe Security Facility (Sun MSF) の使い方については、『Sun Mainframe Security Facility 管理者ガイド』を参照してください。

## マニュアルの構成

第 1 章では、Sun MTP サーバーとクライアントのプロセス、開発システム、サポートするファイルタイプ、およびリソース定義の上限について説明します。また、補完製品についても概要を説明します。Sun MTP のユーザーインタフェースの標準に関する情報も含まれています。

第 2 章では、システムトランザクションについて説明します。

第 3 章では、VSAM データセットと VSAM カタログの管理方法を説明します。また、VSAM ファイルの回復についても説明します。

第 4 章では、トランザクションクラスの設定と使用法について説明します。

第 5 章では、レコードエディタを使用してレコードを追加、変更、および削除する方法を説明します。

第 6 章では、機能シップ、トランザクション経路指定、分散プログラムリンクなどの ISC 機能の使用方法を説明します。

第 7 章では、オペレーティングシステムのセキュリティーと Sun MTP の基本的なセキュリティー機能について説明します。

第 8 章では、外部セキュリティーマネージャーへの Sun MTP Secure のインタフェースについて説明します。

第 9 章では、アカウンティング機能と領域が生成するアカウンティングデータについて説明します。

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) とユーザールーチンをサポートするために、領域をカスタマイズする方法について説明します。

第 11 章 では、Sun Mainframe Administration Tool について説明します。

付録 A では、Sun MTP シェルスクリプトのカスタマイズ方法を説明します。

付録 B では、日付/時刻設定ユーティリティー「`kixdate`」の使用方法について説明します。

用語集には、単語および語句とその定義が一覧にまとめてあります。

# 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です。いずれかを指定する必要があります。 | kixfile [-r{Y\|N}] |
| \| | 区切り文字 (セパレータ) です。この文字で区切ら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 構成ガイド』 | 819-2515-10 |
| | 『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』 | 819-2516-10 |
| | 『Sun Mainframe Transaction Processing ソフトウェア インストールガイド』 | 819-2517-10 |
| | 『Sun Mainframe Transaction Processing ソフトウェア メッセージガイド』 | 819-2518-10 |
| | 『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』 | 819-2519-10 |
| | 『Sun Mainframe Transaction Processing ソフトウェア 障害追跡とチューニング』 | 819-2520-10 |
| | 『Sun Mainframe Transaction Processing ソフトウェア XA リソースマネージャーの使用』 | 819-2358-12 |
| | 『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 |

| 製品 | タイトル | Part No. |
|---|---|---|
| Sun Mainframe Administration Tool | 『Sun Mainframe Administration Tool ユーザーズガイド』 | 819-2523-10 |
| Sun Cluster 用の高可用性エージェント | 『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 |
| 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 | * |
| 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 管理者ガイド』, Part No. 819-2514-10

# 第1章

# 概要

この章では、Sun MTP ソフトウェアについて説明します。次のトピックについて説明します。



## Sun MTP サーバープロセス

Sun MTP メインサーバープロセスは、クライアント以外のすべてのコンポーネントまたはプロセスを起動します。これらのコンポーネントまたはプロセスは、メインサーバープロセスの子プロセスと呼ばれます。

構成によって違いはありますが、領域の実行中には、次のプロセスが実行できます。

- メインサーバー (`unikixmain`)
- トランザクションサーバー (`unikixtran`)
- 開始サーバー (`unikixstrt`)
- 回復サーバー (`unikixrcv`)
- プリントサーバー (`unikixprt`)
- パーティション外キューサーバー (`unikixept`)
- 通信マネージャー (`unikixCommMgr`)
- TCP/IP サーバー (`unikixtcp`)
- DCL プロトコルスタックを使用する SNA サーバー用のシステム間通信 (ISC) サーバー (`unikixdcl`)

- ソケットサーバー (unikixsock)
- SSL (Secure Sockets Layer) サーバー (unikixssl)
- トランザクションイニシエータサーバー (unikixtrin)
- TN3270 (mux'd) サーバー (unikixtnemux)
- 管理サーバー (unikixadmin)
- IBM WebSphere MQ リスナー (unikixqm)

機能のライセンスが提供され、設定されていれば、スタート、回復、ISC、プリント、ソケット、TCP/IP、DCL、パーティション外キュー、通信マネージャー、トランザクションイニシエータサーバー、WebSphere MQ リスナー、およびメインサーバープロセスは、それぞれの領域に対して実行されます。ユーザー要求に応じて、追加のプロセスが起動されることもあります。

メインサーバープロセス unikixmain の詳細については、『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。

# Sun MTP クライアントプロセス

Sun MTP クライアントプロセスは、ユーザーを処理します。ローカルクライアント以外のクライアントは、通信サーバーの 1 つに接続します。このような通信サーバーには、TCP/IP、DCL、トランザクションイニシエータ、ソケットリスナー、WebSphere MQ リスナー、および TN3270 サーバーがあります。特定のクライアントを使用するための Sun MTP ソフトウェアの設定については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

すべての Sun MTP 通信サーバーは、複数の (最高数千の) ユーザーを処理します。ローカルクライアントは、Sun MTP トランザクションサーバーに直接接続して、1 ユーザーだけを処理します。ユーザーごとにクライアントを選択する必要があります。

## ローカル端末クライアント

ローカル端末クライアントプロセス unikixl は、unikix コマンドとそのオプションを入力して、領域に接続すると実行されます。

unikix コマンドとそのオプションの詳細については、『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。ローカルクライアントの起動については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

# TN3270 および TN3270E エミュレータクライアント

Sun MTP TN3270 サーバー `unikixtnemux` は、TCP/IP TN3270 および TN3270E プロトコルを使用して、商用 3270 エミュレータをサポートします。TN3270 プロトコルは、従来の TCP/IP Telnet プロトコルの拡張です。これにより、ASCII 以外の、ブロックモードのデバイス (たとえば IBM-3270 端末および Sun MTP のアプリケーション) による TCP/IP 経由の通信を可能にします。TN3270 または TN3270E クライアントとして動作する 3270 エミュレータは、プロトコルを正しく実装していれば、プラットフォームにかかわりなく、`unikixtnemux` で正常に機能します。

さらに、TN3270E は次のものをサポートします。

- 3287 プリンタ
- LU 名接続

TN3270 または TN3270E クライアントを使用できるように領域を設定する方法については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

# Sun MTP J3270 端末エミュレータクライアント

Sun MTP J3270 ソフトウェアは、3278 モデル 2、3、4、および 5 を拡張モードと非拡張モードの両方でサポートする 3270 端末エミュレータです。このソフトウェアは、Java™ 実行時環境 (JRE) が実装されている、Solaris、Microsoft Windows システム、およびその他のプラットフォームで動作します。このクライアントを使用して、TN3270 サーバーに接続できます。Sun MTP 領域は、TN3270 サーバーである `unikixtnemux` を使用して端末エミュレータクライアントをサポートします。

ソフトウェアのインストールと使用法については、『Sun MTP J3270 ユーザーズガイド』を参照してください。

# IBM 3270 SNA デバイスクライアント

Sun MTP ソフトウェアは、オプションの TPS サーバー製品を使用して、3270 SNA デバイスをサポートします。SNA デバイスのサポートには、次のコンポーネントを使用します。

- SNA サーバー
- トランザクションイニシエータサーバー (`unikixtrin`)
- 3270 クライアントイニシエータ (`unikixi`)
- 3270 クライアント (`unikixb`)

SNA サーバーについては、TPS SNA サーバーのマニュアルを参照してください。3270 SNA クライアントを使用するための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。

## TCP/IP ソケットクライアント

クライアントアプリケーションは、TCP/IP ソケットを使用して領域と通信します。Sun MTP リスナーサーバープロセス `unikixsock` は、あらかじめ定義された TCP ポートで着信要求を待機します。この待機プロセスは、要求をトランザクションサーバーに転送し、そこで要求が処理されます。要求を受け取ったトランザクションサーバーは、遠隔システムのユーザーアプリケーションとの間で直接通信を開始します。

詳細は、次を参照してください。

- リスナー機能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』
- メッセージの入出力形式とメッセージの送受信用プログラムについては、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』
- ソケットユーザー出口をカスタマイズして、標準以外のメッセージ形式をサポートする方法については、261 ページの「ソケットユーザー出口のカスタマイズ」

## SSL (Secure Sockets Layer) クライアント

SSL (Secure Sockets Layer) によって、アプリケーションでは、認証が必要な耐タンパ性暗号化通信にソケットを使用できます。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。SSL が有効なサーバーは、SSL が有効なクライアントに対して自身を認証できるほか、クライアント側からもサーバーに対する認証が可能になるため、暗号化された接続を両サイドから確立できます。

詳細は、次を参照してください。

- SSL 要求を受け入れられるように Sun MTP を設定する方法については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』
- SSL をサポートする方法については、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』
- ソケットユーザー出口をカスタマイズして、標準以外のメッセージ形式をサポートする方法については、264 ページの「SSL ユーザー出口のカスタマイズ」

## IBM WebSphere MQ クライアント

Sun MTP ソフトウェアは、WebSphere MQ クライアントをサポートします。WebSphere MQ とは、Sun MTP トランザクションプログラムから呼び出されてサービスを要求するルーチンのセットです。詳細については、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』、および IBM WebSphere MQ のドキュメントを参照してください。

## 外部呼び出しインタフェースクライアントおよび外部表示インタフェースクライアント

Sun MTP ソフトウェアは、外部呼出しインタフェース (ECI) クライアントと外部表示インタフェース (EPI) クライアントをコーディングするためのアプリケーションプログラミングインタフェース (API) を提供しています。この製品は Sun MTP クライアントといいます。Sun MTP は、IBM の CICS クライアントや Universal Client 製品もサポートしています。

『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』では、ECI クライアントおよび EPI クライアントが使用できるように領域を設定する方法について説明しています。ECI クライアントおよび EPI クライアントのマニュアルも参照してください。

# Sun MTP 標準規格

この節では、次の Sun MTP 標準規格について説明します。


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


ご使用のアプリケーションによって、標準が異なる場合があります。

# 画面の形式

データ入力画面は、次の図のようになります。

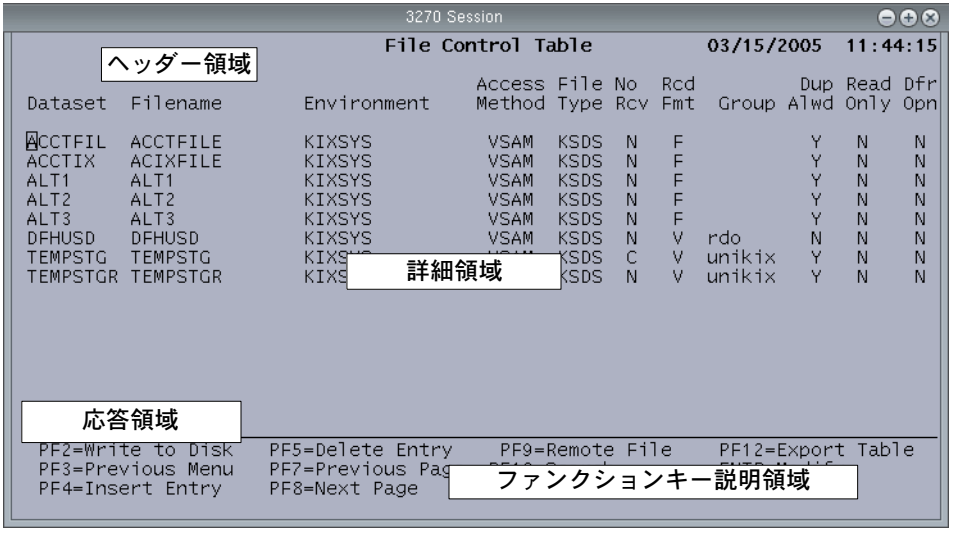


**図 1-1** Sun MTP 画面形式の例

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。

| 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 (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。また、操作の状態も表示されます。 |
| ファンクションキー説明領域 (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。 |

# ファンクションキー

次のファンクションキーは、指定された操作を Sun MTP 画面上で実行します。

| ファンクションキー | アクション |
|---|---|
| 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>ここで PF3 を押すと、変更内容が保存されずに前の画面に戻ります。ほかのキーを押すと、処理を続行できます。 |
| 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が強調表示されます。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は、次の表に示す指定に従う必要があります。

| 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なりません。パス名の任意の部分に代えて、先頭にドル符号 ( $ ) を付けた環境変数を使用できます。たとえば、次の 2 行はどちらも有効であり、同一のディレクトリ名を示します。<br>• `mtp/mtp8/finance/sys`<br>• `$KIXSYS`<br>$ 記号は、環境変数 `KIXSYS` 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)。<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)。 |

# 開発システム

開発システムは、複数のツールが 1 つのメニューインタフェースで使用できるようになっています。これらのツール群を使用すると、9 ページの「メニューオプション」で説明するように、領域管理とアプリケーション開発タスクが実行できます。

## 開発システムの起動

開発システムのメニューを表示するには、領域にローカルに接続する必要があります。各ツールは、直接アクセストランザクションで起動できます。詳細は、第 2 章を参照してください。

## ▼ 開発システムの起動

- **ローカルクライアントの空白のトランザクション画面で、左上部に** CMNU **トランザクションを入力し、Enter キー (Return キーではありません) を押します。**

  図 1-2 に示す「Development System」メインメニューが表示されます。

  領域が実行中でなく、エラーメッセージが表示される場合、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』で説明されているように領域を起動します。

---

**注 –** CMNU は、3270 端末または TN3270 クライアントをサポートしていません。

---

# メニューオプション

「Development System」メインメニューには、実行可能な機能が表示されます。

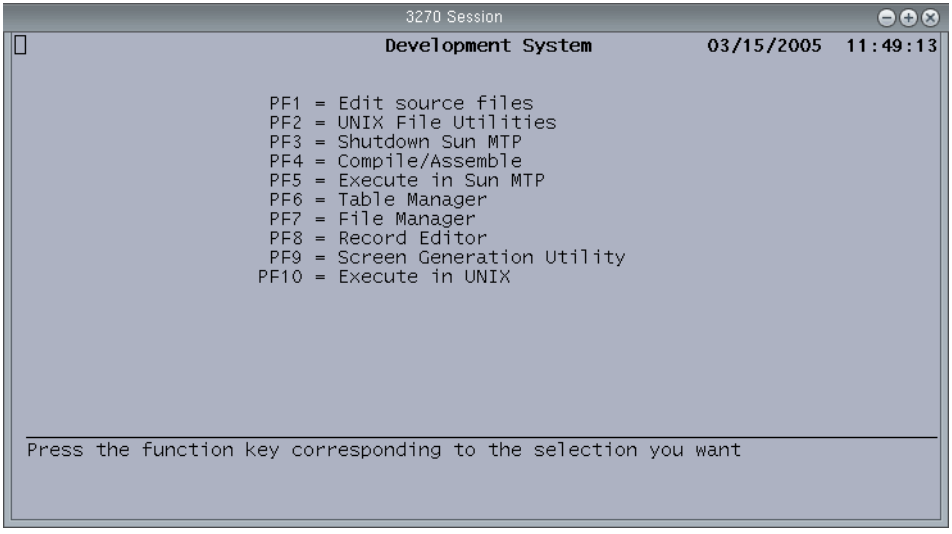


**図 1-2** 「Development System」メインメニュー

メニューオプションでは、次の機能を実行できます。

| | |
|---|---|
| Edit source files | PF1 を押すと、ファイルエディタが起動します。ファイルエディタは、アプリケーションファイルの編集に使用します。詳細は、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。 |
| UNIX File Utilities | PF2 を押すと、Browse、Copy、Concatenate、Delete、Print、Rename、Search などの UNIX ファイル機能が有効になります。詳細は、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。 |
| Shutdown Sun MTP | PF3 を押すと、Sun MTP クライアントと Sun MTP サーバーの停止を確認するメッセージが表示されます。もう一度 PF3 を押すと停止します。 |
| Compile/Assemble | PF4 を押すと、COBOL と PL/I のコンパイル機能および BMS (基本マッピングリソース) アセンブラにアクセスします。詳細は、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。 |
| Execute in Sun MTP | PF5 を押すと、メニューシステムが終了します。これで、端末からトランザクション識別子を入力できます。 |
| Table Manager | PF6 を押すと Table Manager が開き、アプリケーション制御用のテーブルを管理できます。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。 |
| File Manager | PF7 を押すと File Manager が開き、完全なアプリケーションの一部である VSAM ファイルが表示されます。File Manager を使用すると、VSAM カタログを作成、管理できます。詳細は、第 3 章を参照してください。 |
| Record Editor | PF8 を押すと、「Data File Editor」メニュー画面にレコードエディタが開きます。レコードエディタを使用して、レコードを変更できます。詳細は、第 5 章を参照してください。 |
| Screen Generation Utility | PF9 を押すと、Screen Generation Utility (SGU) が起動します。詳細は、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。 |
| Execute in UNIX | PF10 を押すと UNIX コマンドプロンプトが表示され、`unikix` 以外の UNIX コマンドを入力できます。`unikix` を入力すると、Sun MTP ローカルクライアントがすでに端末で起動されていることを示すエラーメッセージが表示されます。 |

# サポートするファイルタイプ

表 1-1 は、サポートされるファイルタイプを示します。拡張子を変えれば、異なるタイプのファイルに同じ名前を付けることができます。ソースコード、変換済みソースコード、オブジェクトコード、基本マッピングサポート (BMS) ソースコード、および BMS オブジェクトコードに対し、拡張子だけを変えて同じ名前を付けることができるため、アプリケーション開発に便利です。一部の言語プロセッサには、入力ファイルに特定の拡張子が必要です。

Sun MTP ファイル名は、拡張子 (通常、3 文字以内とピリオド) を含め 14 文字以内にする必要があります。

Sun MTP のファイル名を定義する場合、プログラム名 (Java プログラムと共有オブジェクトを除く)、データセット名、およびマップセットには大文字を使用する必要があります。特殊文字は使用できません。VSAM ファイル名には、大文字と小文字の両方を使用できます。

**表 1-1** Sun MTP – サポートされるファイルタイプ

| ファイルタイプ | 拡張子 |
|---|---|
| ACUCOBOL-GT 中間ファイル | `.acu` |
| BMS マップセット実行可能ファイル | `.map` |
| BMS マップセットソース | `.bms` |
| `EXEC CICS` コマンドが入った C のコード | `.ccs` |
| C のネイティブコード | `.c` |
| C のヘッダーファイル | `.h` |
| IBM OS/VS COBOL の Common Language Translator への入力 | `.clt` |
| IBM COBOL II の Common Language Translator への入力 | `.cl2` |
| Java ソースコード | `.java` |
| Java コンパイル済みプログラム | `.class` |
| Java アーカイブ | `.jar, .zip` |
| `kixclt` (共通言語トランスレータ) 出力。COBOL コンパイラへの入力 | `.cbl` |
| Liant Open PL/I コード | `.pli, .pp1` |
| `EXEC CICS` コマンドが入った Liant Open PL/I コード | `.plt` |
| Liant Open PL/I。データベースプログラムを想定 | `.ppl` |
| Micro Focus Animator デバッグファイル | `.idy` |
| Micro Focus Server Express 中間ファイル | `.int` |

表 1-1　Sun MTP – サポートされるファイルタイプ (続き)

| ファイルタイプ | 拡張子 |
|---|---|
| Micro Focus Server Express ネイティブファイル | .gnt |
| プリンタ出力 | .prt |
| Screen Generation Utility (SGU) 形式ファイル | .sgu |
| Sun MTP テーブル、ネイティブ形式 | .tbl |
| Sun MTP テーブル、ASCII 形式 | .lst |
| 一時データパーティション外のジョブストリーム宛先 | .job |
| VSAM カタログ、ASCII 形式 | .lst |
| VSAM KSDS 索引ファイル | .idx |
| VSAM KSDS データファイル | .dta |
| VSAM KSDS データファイルセグメント。*n*=1,...7 | .dt*n* |

# リソース定義の上限

次の表では、領域で定義できるリソースとその上限を示します。

表 1-2　Sun MTP リソース定義の上限

| カテゴリ | 説明 | 上限 |
|---|---|---|
| 3270 端末 | 3270 端末クライアント (unikixb プロセス) | 上限なし |
| データ変換リソース | データ変換テンプレート | 255 |
| 環境変数 | 長さ | 1024 バイト |
| EPI クライアント | トランザクションでクライアントに送信可能なデータの量 | 32,400 バイト |
| グループ | 領域に対して定義される数 | 上限なし |
| マップのリソース | マップセット内のマップ | 255 |
| マップのリソース | マップ内のフィールド | 1023 |
| マップのリソース | マップセット | 上限なし |
| メモリーのリソース | メモリーサイズ (ローカルおよび共有) | 2G バイト/割り当て |
| プリンタのリソース | プリンタ | 上限なし |
| プログラム | プログラム | 上限なし |

表 1-2　Sun MTP リソース定義の上限 (続き)

| カテゴリ | 説明 | 上限 |
|---|---|---|
| 回復ファイル | サイズ | 400K バイトからオペレーティングシステムがサポートする最大サイズまで |
| SNA 通信 | `unikixdcl` サーバープロセス | 255 |
| SNA 通信 | SNA: インバウンドおよびアウトバウンドの ISC セッションの合計数 | 200 |
| TCP/IP 通信 | TCP/IP: インバウンドおよびアウトバウンドの ISC セッションの合計数 | 200 |
| TN3270 クライアント | 同時ユーザー/`unikixtnemux` サーバープロセス | 1000 |
| TN3270 サーバー | `unikixtnemux` サーバープロセス | 255 |
| トランザクションクラス | トランザクションクラス (ユーザー定義) | 62 |
| トランザクション処理プログラム | トランザクション処理プログラム | 224 |
| 一時データキュー | 一時データキュー | 上限なし |
| ユーザー | 端末/ユーザー | 上限なし |
| VSAM 共有バッファー | バッファー | VCT では 999,999。ただし、ほかのメモリー要件によって制限されます。 |
| VSAM データセット | データセット | 上限なし |
| VSAM ESDS データセット | ファイルサイズ | **Solaris**: 4G バイト<br>**AIX**: 2G バイト |
| VSAM KSDS データセット | 代替索引/データセット | 12 |
| VSAM KSDS データセット | `FAST_UNIKIXVSAM` オープンデータセット | 10 |
| VSAM KSDS データセット | ファイルサイズ | **Solaris**: オペレーティングシステムの最大/セグメント<br>**AIX**: 2G バイト |
| VSAM KSDS データセット | メインセグメントを含むスパンファイルのセグメント | 8 |
| VSAM RRDS データセット | 相対レコードの最大数 | 2,147,483,647 |
| VSAM キーの長さ | サイズ | 255 バイト |
| VSAM レコードのサイズ | 可変長レコード | 32,767 バイト |
| VSAM レコードのサイズ | 固定長レコード | 32,767 バイト |

# 補足コンポーネントと製品

次のソフトウェアコンポーネントと製品は、Sun MTP を補足するものです。

- Sun Mainframe Administration Tool。このソフトウェアでは、ローカルおよび遠隔領域のアクティビティーを監視できます。グラフィカルユーザーインタフェースから、リソースの使用状況、パフォーマンス、その他の要因に関する情報が表示できるため、ご使用のアプリケーション環境のパフォーマンスを最適化できます。
- Sun MTP Secure および Sun Mainframe Security Facility (Sun MSF)。Sun MTP Secure では、Sun MSF、セキュリティー出口、Sun 以外の外部セキュリティーソフトウェアパッケージを使用して外部セキュリティー管理を実行できます。Sun MSF には、役割によるアクセス制御 (RBAC) が用意されています。
- Sun Mainframe Batch Manager ソフトウェア (Sun MBM) は、制御された環境でバッチジョブを実行する機能を備えています。Sun MBM は、バッチプログラムの実行を管理およびスケジュールする独立したプロセスで構成されています。ジョブのスケジュールは、開始時刻やバッチプロセスの最大数、ジョブの優先順位やクラスなどのパラメータを割り当てることで行われます。Sun MBM ではジョブ状態機能も用意されているので、ジョブ属性の割り当て、ジョブ属性の変更、およびジョブの現在の状態の判別が可能です。
- クライアント製品。詳細は、5 ページの「外部呼び出しインタフェースクライアントおよび外部表示インタフェースクライアント」を参照してください。

# Sun 以外の製品

Sun MTP は、Sun 以外のさまざまな製品をサポートしています。Sun 以外の製品をインストールする場合は、事前に Sun Microsystems にプラットフォーム固有の情報をお問い合わせください。Sun 以外の製品が認定されている場合でも、すべての製品ですべての機能が動作確認されているわけではなく、またパフォーマンスを保証しているわけでもありません。Sun 以外の製品の適切なバージョンで、アプリケーションの機能をテストし、検証する必要があります。

第2章

# システムトランザクション

Sun MTP には、開発システムのシステム制御とアクセスのためにトランザクションのセットが用意されています。これらのトランザクションは、「システムトランザクション」と呼ばれます。

この章では、次のトピックについて説明します。



## システムトランザクションの概要

次のトランザクションを使用して、制御とデバッグを行います。

| トランザクション | 使用法 |
|---|---|
| CBCH | オンラインバッチの実行 |
| CEBR | 一時記憶域のブラウズ |
| CECI | CICS コマンドを会話的に実行 |

| トランザクション | 使用法 |
|---|---|
| CEDA | リソース定義の動的操作 |
| CEDF | デバッグ機能の実行 |
| CEMT | 状態の設定/表示 |
| CINI | トランザクション処理プログラムの再初期化 |
| CRTE | Sun MTP 領域からのトランザクション経路指定 |
| CSMT | システム終了 |
| CSPG | システムページング |
| CESN | サインオン |
| CSSN | サインオン |
| CESF | サインオフ |
| CSSF | サインオフ |

次のトランザクションは、Sun MTP が内部的に使用します。

| トランザクション | 使用法 |
|---|---|
| CCIN | CICS クライアントトランザクション |
| CPLT | PLT 処理の初期化 |
| CPMI | LU6.2 ミラートランザクション |
| CRSR | システム間通信 (ISC) 遠隔スケジューラトランザクション |
| CTIN | CICS クライアントトランザクション |
| CVMI | LU6.2 同期レベル 1 ミラートランザクション |

次のトランザクションを使用すると、開発システムの一部に直接アクセスできます。トランザクションセキュリティーを有効にする場合、エンドユーザーは直接アクセストランザクションを使用する必要がないため、これらのトランザクションへのアクセスを制限する必要があります。

| | |
|---|---|
| CFMS | File Manager |
| CMNU | 開発システムのメインメニュー。ローカルに接続されている端末だけで利用できます。TN3270 接続では利用できません。 |
| CRED | レコードエディタ |
| CSGU | 画面生成ユーティリティー (SGU) |
| CTBL | Table Manager |

# CBCH – オンラインバッチの実行

CBCH トランザクションを使用して、あらゆるプログラムタイプのオンライン環境から、バッチプログラムまたはシェルスクリプトを即時実行します。このトランザクションは、自動的にバッチプログラムあるいはシェルスクリプトを実行するために使用される通常の優先順位を無視し、それらをフォアグラウンドトランザクションのように開始します。

形式:

CBCH *program*

ここで、*program* は、開始するバッチプログラムまたはシェルスクリプトの絶対または相対パス名です。これらのバッチプログラムまたはシェルスクリプトは、KIXSYS 環境変数によって定義されるディレクトリから実行されます。相対パス名を入力すると、$KIXSYS ディレクトリに相対的になります。

---

**注 –** システムが、CEMT SET によって BATCH 状態に設定されている場合、CBCH トランザクションでバッチプログラムまたはシェルスクリプトを実行することはできません。詳細は、54 ページの「システム状態の設定」を参照してください。

---

CBCH で、デバッガオプションを指定して CEDF トランザクションを使用し、バッチ COBOL プログラムをデバッグできます。バッチプログラムをデバッグするには、COBOL 実行時システムである unikixvsam の -D パラメータを使用して、バッチシェルスクリプトでプログラムを呼び出す必要があります。バッチプログラムとデバッグの詳細については、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。

CEDF トランザクションについては、43 ページの「CEDF – デバッグ機能の実行」で概要を説明しています。詳細については、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。

# CEBR – 一時記憶域のブラウズ

CEBR トランザクションを使用して一時記憶域をブラウズし、データをパーティション内キューで検索したり、パーティション内キューに挿入することができます。

これを開始するには、CEBR トランザクションを入力します。パラメータは使用できません。トランザクションを終了するには、PF3 キーを押します。

トランザクションを開始すると、初期の一時記憶域キューがデフォルトで CEBR*xxxx* に設定されます。ここで *xxxx* は、要求を実行している端末の端末識別子です。

次の図は、デフォルトの CEBR 画面を示しています。

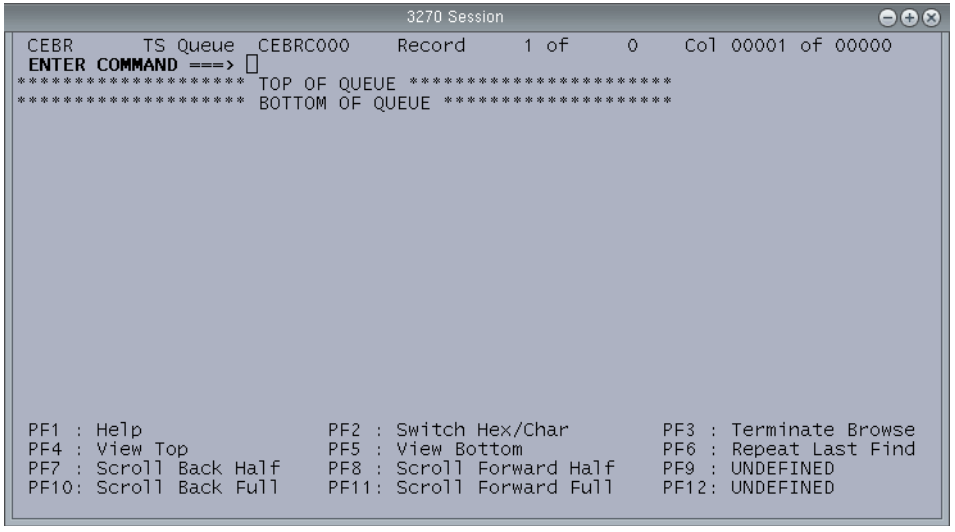


**図 2-1** CEBR – ブラウズユーティリティーのデフォルト画面

CEBR を開始すると、コマンド行と PF キーの両方からコマンドを受け取ることができます。画面に表示される情報は、参照しているキューに書き込まれたデータによって異なります。

CEBR は、コマンド行で次のコマンドを使用できます。

| コマンド | アクション |
|---|---|
| BOTTOM または B | キューの最後を表示します。PF7 キーを押すと、キューの前のページへとスクロールされます。 |
| COLUMN または C | 指定した列を表示します。たとえば、表示したキューに 80 文字 (16 進数で 40 文字) 以上のエントリがある場合、エントリには最初の 79 (または 39) 文字だけが表示されます。キューのエントリにあるそれ以降の文字を表示するには、表示する位置に列番号を設定します。 |
| FIND または F | 指定した文字列をキューの中で検索します。指定した文字列が「/」のような特殊文字で始まっている場合は、それが文字列終端として認識されるため、これと対になる文字列終端をトランザクションが検索します。そうでない場合は、最初の空白文字が文字列終端となります。<br>CEBR は、常に入力を大文字に変換します。したがって、小文字を含む文字列は検索できません。CEBR が文字列を検出すると、最初の行と最初の列に、検索された文字列の開始が表示されるようにキューの表示が変わります。 |
| GET または G | パーティション内キューを現在の一時記憶域キューの中に移動します。GET コマンドを使用してキュー名を入力し、一時データキューを指定します。 |
| LINE または L | 指定した行を画面の 2 行目に表示します。 |
| PURGE | 現在の一時記憶域キューを削除します。 |
| PUT または P | 現在の一時記憶域キューを指定した一時データキューに移動します。 |
| QUEUE または Q | 現在のキューを指定したキューに変更します。 |
| TERMINAL または TERM | 現在のキューを CEBR*xxxx* に変更します。 *xxxx* は、指定されたパラメータです。 |
| TOP または T | キューの最初のエントリを表示します。PF8 キーを押すと、キューの次のページへとスクロールされます。 |

CEBR トランザクションの実行時には、次のファンクションキーが利用できます。

| ファンクションキー | アクション |
|---|---|
| PF1 | 有効なコマンド行コマンドのリストを表示します。 |
| PF2 | 文字表示と 16 進表示を切り替えます。 |
| PF3 | CEBR トランザクションを終了します。 |
| PF4 | キューの最初のエントリを表示します。 |
| PF5 | キューの最後のエントリを表示します。 |
| PF6 | 前に実行した find コマンドに対して、次に一致する文字列を検索します。 |

| ファンクションキー | アクション |
|---|---|
| PF7 | 現在の行を示すポインタを半ページ分後退させます。 |
| PF8 | 現在の行を示すポインタを半ページ分前進させます。 |
| PF10 | 現在の行を示すポインタを 1 ページ後退させます。 |
| PF11 | 現在の行を示すポインタを 1 ページ前進させます。 |

# CECI – コマンドインタプリタ

CECI トランザクションを使用すると、IBM CICS CECI トランザクションに類似した方法で、3270 端末ウィンドウから CICS コマンドを会話的に実行できます。

## ▼ CECI インタプリタを起動する

- **空白のトランザクション画面で、トランザクション識別子 CECI を入力します。**
  コマンド画面に、サポートされる CICS コマンドが表示されます。

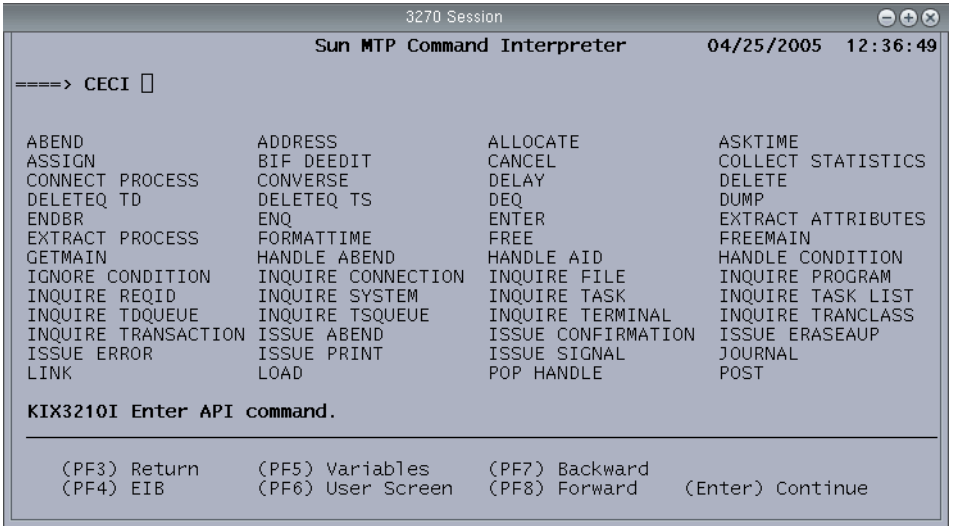


図 2-2 CECI—Commands 画面

CECI Commands 画面では、次のファンクションキーが利用できます。

| ファンクションキー | アクション |
|---|---|
| PF3 | 前の画面に戻ります。 |
| PF4 | Exec Interface Block 画面を表示します。 |
| PF5 | ユーザー定義の変数を表示します。 |
| PF6 | コマンドによって生成されたユーザー画面を表示します。 |
| PF7 | 1 ページ分前のページにスクロールします。 |
| PF8 | 1 ページ分次のページにスクロールします。 |
| Enter | 選択したコマンドの Syntax 画面を表示します。 |

# CECI コマンドの定義と実行

ここでは、CECI 画面およびこれらの画面でのコマンドの定義方法と実行方法を説明します。

## Commands 画面

CECI Commands 画面は、Sun MTP がサポートするコマンドを表示します。この画面で次の操作が可能です。

- コマンド入力領域にコマンド名を入力してから Enter キーを押すと、そのコマンドの Syntax 画面が表示されます。
- ファンクションキーを押すと、次の画面が表示されます。
  - EIB (Exec Interface Block) 画面 (28 ページの「EIB 画面」)
  - Variables 画面 (28 ページの「Variables 画面」)
  - ユーザー画面 (32 ページの「ユーザー画面」)

## Syntax 画面

Syntax 画面が表示されると、Commands 画面で入力したコマンド、または空白のトランザクション画面で CECI トランザクションを実行するときに入力したコマンドが一覧表示されます。画面の 19 行目にある「STATUS」には、「Command Syntax Check」と表示されます。

この節では、CECI からコマンドを作成および実行する方法を説明します。また、コマンドを作成および実行する際に使用できる、ほかの画面について説明した節を示します。次に示す各節では、CECI が引数を管理する方法と一定のコマンドから期待されるプログラム制御について説明しているので、合わせて参照してください。

- 33 ページの「動的レシーバ引数」
- 34 ページの「リテラルオプション引数」
- 34 ページの「プログラム制御」
- 35 ページの「作業論理ユニット (LUW)」

次の図に、GETMAIN コマンドを入力すると表示される Syntax 画面を示します。「STATUS」が、「Command syntax check」であることを確認します。

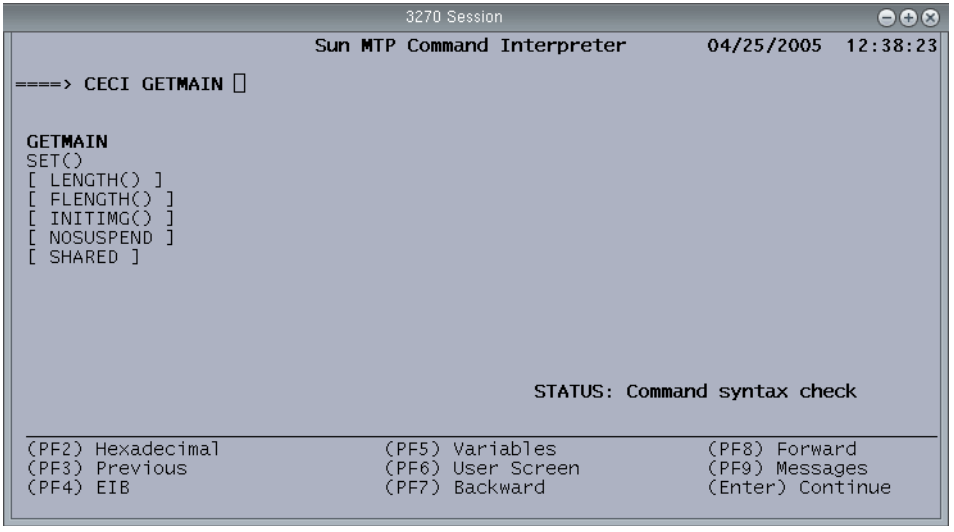


**図 2-3** CECI—Syntax 画面 (Command syntax check)

Syntax 画面では、次の記号を使用します。

| 記号 | 説明 |
|---|---|
| [ ] | 指定は可能だが、必須ではないオプション。 |
| ( ) | 引数を取るオプション。 |
| **太字** | オプションリストのオプションをコマンド行に入力すると、太字で表示されます。コマンド行も太字です。 |

CECI の Syntax 画面では、次のファンクションキーが利用できます。

| ファンクションキー | アクション |
|---|---|
| PF2 | 表示を文字/10 進 (デフォルト) から 16 進に切り替えます。Char/Decimal と表示されている場合は、16 進から文字/10 進に切り替えます。 |
| PF3 | 前の画面に戻ります。 |
| PF4 | Exec Interface Block 画面を表示します。 |
| PF5 | ユーザー定義の変数を表示します。 |
| PF6 | 3270 データを端末画面に送信するコマンドを実行したあと、CECI によって取り込まれた画面が表示されます。詳細は、32 ページの「ユーザー画面」を参照してください。 |
| PF7 | 1 ページ分前のページにスクロールします。 |
| PF8 | 1 ページ分次のページにスクロールします。 |
| PF9 | トランスレータが発行した診断メッセージを表示するメッセージ画面を表示します。詳細は、32 ページの「メッセージ画面」を参照してください。 |
| Enter | 構文シーケンスの次の画面が表示され、最後にコマンドが実行されます。手順を確認して、コマンドを作成および実行してください。 |

## ▼ コマンドを作成および実行する

1. **コマンド画面で、たとえば GETMAIN コマンドを入力して Enter キーを押します。**

   Syntax 画面に、CECI GETMAIN コマンドが表示されます。STATUS が、「Command syntax check」であることを確認します。

2. **コマンド行を作成します。**

   a. **コマンド行で、1 つ以上のオプションと引数を入力します。**

      オプションを入力するたびに Enter キーを押すか、またはすべてのオプションを入力してから Enter キーを押します。

   b. **コマンドを作成するときに、PF5 キーを押して Variables 画面を表示すると、変数を定義または表示できます。**

      変数の定義方法については、28 ページの「Variables 画面」を参照してください。

   c. **コマンドを作成するときに、変数値の表示を 16 進表示と文字/10 進表示との間で切り替えるには PF2 キーを押します。**

   d. **コマンドが完成するまで、必要な回数だけこの手順を繰り返します。**

e. **コマンドを作成し終わったら、Enter キーを押します。**

図 2-4 は、完成したコマンドを示します。

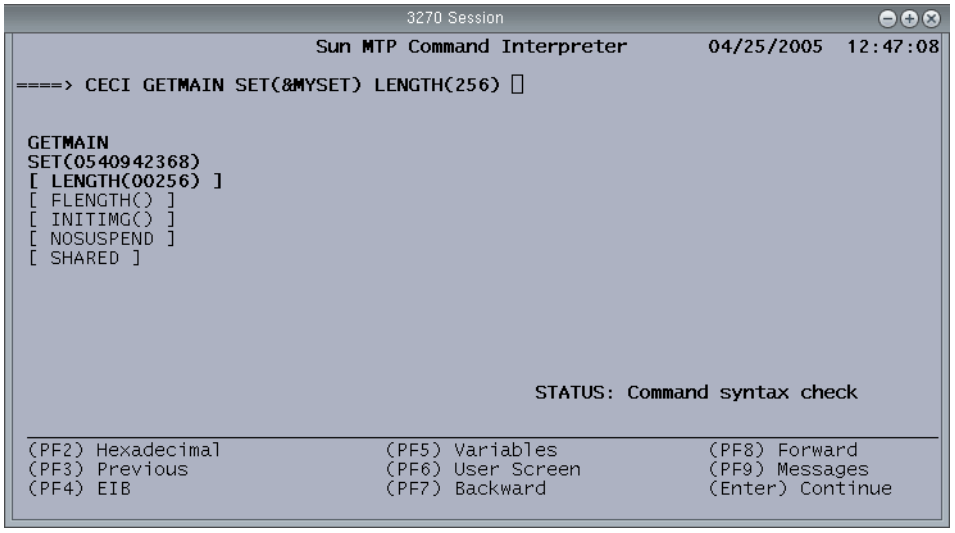


**図 2-4** CECI—Syntax 画面 (Command syntax check)

3. **コマンドを変換します。**

   a. **コマンド行が完成したら、Enter キーをもう 1 度押してコマンドをトランスレータに渡します。**

   変換に成功すると、Syntax 画面の 「STATUS」フィールドが、「About to execute command」に変わります。選択したオプションが太字で表示されていることを確認してください。詳細は、図 2-5 を参照してください。

   b. **PF9 キーを押して、トランスレータが生成したメッセージを確認します。**

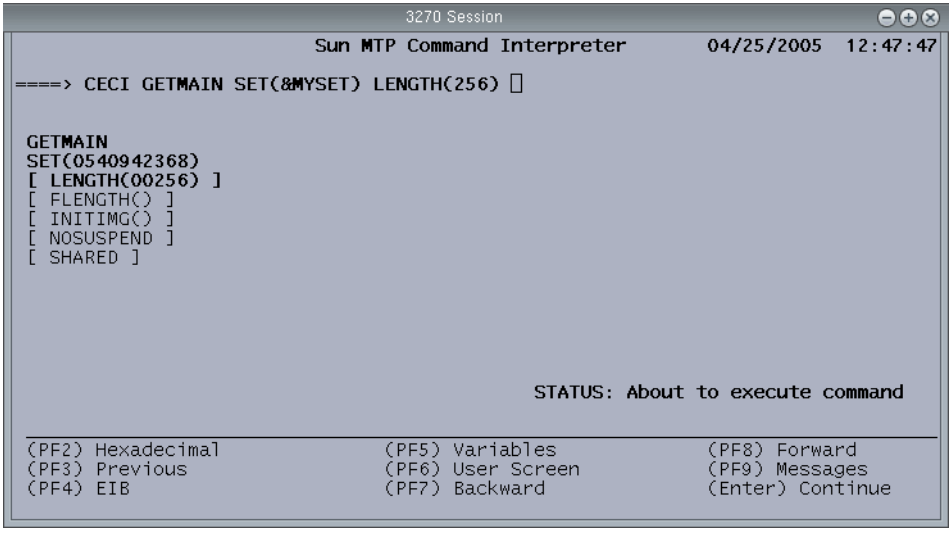


**図 2-5** CECI—Syntax 画面 (About to execute command)

4. **ここで、次のいずれかを実行します。**
   - Enter キーをもう 1 度押すと、コマンドが実行されます。コマンドが実行されると、拡張された構文がコマンドの結果を反映して再表示され、「STATUS」インジケータが「Command executed」に更新されます。EIB の「EIBRESP」フィールドが解釈され、応答が 20 行目、つまり「STATUS」インジケータの下に表示されます。この例では NORMAL と表示されています。詳細は、図 2-6 を参照してください。
   - コマンド行を変更します。変更を行うたびに Enter キーを押して、トランスレータを実行します。Enter キーをもう 1 度押すと、コマンドが実行されます。

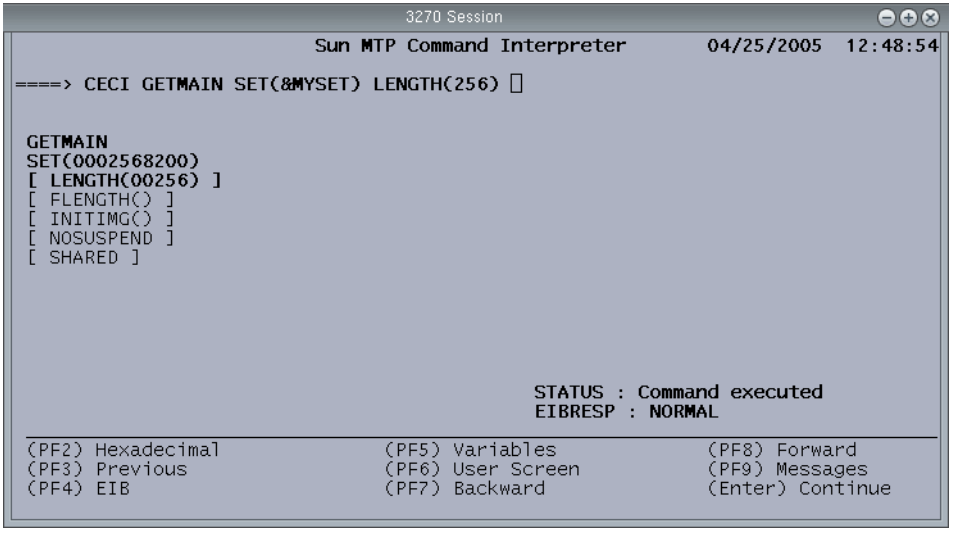


**図 2-6** CECI—Syntax 画面 (Command executed)

5. **コマンドの実行後、次の 2 つが実行できます。**
   - PF4 キーを押して EIB を表示する。詳細は、28 ページの「EIB 画面」を参照してください。
   - PF6 キーを押して、コマンドが生成したユーザー画面を表示する。詳細は、32 ページの「ユーザー画面」を参照してください。

## Hexadecimal 表示画面

Syntax 画面と Variables 画面は、文字 (デフォルト) および 16 進数の両方のモードで表示できます。表示モードを切り替えるには、PF2 キーを押します。Variables 画面で 16 進数モードを使用している場合は、表示モードインジケータが画面の右下に表示されます。

16 進数モードでは、ASCII 印刷可能文字に対応しない 16 進数の値を持つデータのバイトを含む変数を、定義または表示できます。

文字モードで、CECI は、たとえばピリオド (.) のような印刷可能文字に対応しない値を持つバイトを表示します。文字モードで、このような変数バイトをピリオドに変更することはできません。印刷不能文字を表す 16 進数の値を持つ変数バイトをピリオドに置き換えるには、16 進数モードに切り替えて、ピリオドを表す ASCII コード (x'2d') を挿入します。

次の図は、Hexadecimal 表示モードの Variables 画面を示します。

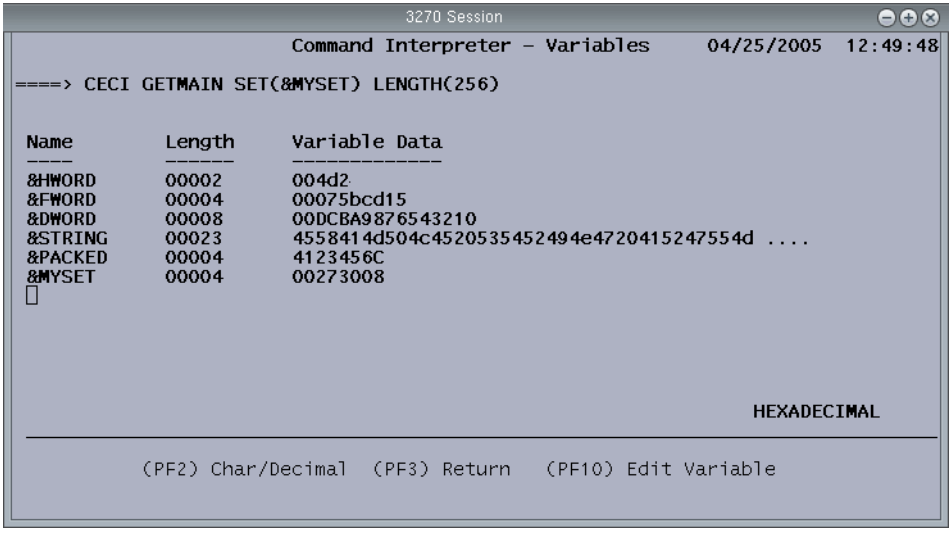


**図 2-7** CECI—Variables 画面 (Hexadecimal 表示モード)

Hexadecimal 表示モードが可能な画面では、PF2 キーのラベルが次のように表示されます。

| | |
|---|---|
| Hexadecimal | 表示を文字/10 進 (デフォルト) から 16 進モードに切り替えます。 |
| Char/Decimal | 表示を 16 進から文字/10 進モードに切り替えます。 |

## EIB 画面

EIB のデータは、CECI セッション中に、Commands 画面または Syntax 画面で PF4 キーを押せば、いつでも表示できます。図 2-8 に、EIB 画面を示します。

PF3 キーを押すと、前の画面に戻ります。

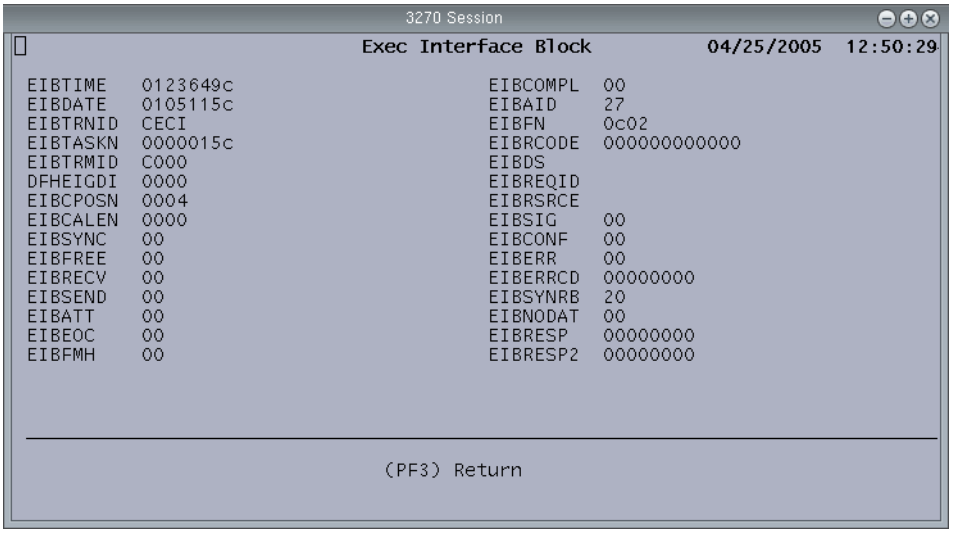


**図 2-8** CECI—EIB 画面

## Variables 画面

CECI は、コマンド構文でオプションの引数として使用する変数を定義するために使用します。定義した変数は CECI セッションの終了まで有効なので、1 つのコマンドを実行して返されたデータを、変数を使用して次のコマンドに渡すことができます。

定義した変数は、コマンド行で使用できます。コマンド構文が拡張されている場合、変数の値の代わりに変数名が使用されます。コマンドの実行中に変数値が変更されると、Syntax 画面と Variables 画面に反映されます。

各変数タイプの例は、CECI が初期化されるときに定義されます。この変数タイプの例を表示するには、Commands 画面または Syntax 画面で PF5 キーを押します。次の図は、5 つの定義済み変数と &MYSET という名前のユーザー定義の変数が表示された Variables 画面 (文字/10 進モード) を示します。

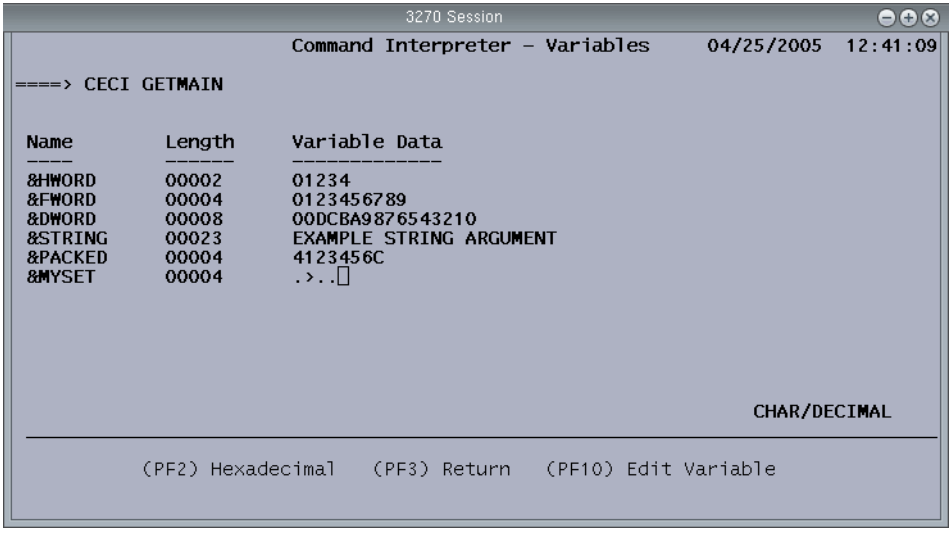


**図 2-9** CECI—Variables 画面

Variables 画面では、次のファンクションキーが利用できます。

| ファンクションキー | アクション |
|---|---|
| PF2 | 表示を文字/10 進 (デフォルト) から 16 進に切り替えます。 |
| PF3 | 前の画面に戻ります。Enter キーを押さずに PF3 キーを押すと、Variables 画面で行なった変更はすべて無効になります。 |
| PF10 | Edit Variables 画面が開きます。詳細は、30 ページの「Edit Variable 画面」を参照してください。 |
| Enter | Variables 画面で変更を行うたびに Enter キーを押すか、Variables 画面セッションの終わりに 1 度だけ Enter キーを押してすべての変更を処理します。 |

Variables 画面では、CECI 変数の定義、削除、および変更が可能です。

## ▼ 変数を追加する

1. 次の「Name」フィールドの最初にカーソルを移動します。
2. 9 文字以下の変数名を入力します。A ～ Z、a ～ z、0 ～ 9 を使用し、アンパサンド (&) 文字で始まる必要があります。

3. **Tab キーを押して「Length」列に移動し、型の省略名を入力するか、長さを表す数値を入力します。**

   省略名を指定した場合は、長さが自動入力されます。表 2-1 に変数の型を示します。

4. **Enter キーを押します。**

## ▼ 変数を削除する

1. **変数名のフィールドを空白で上書きします。**

   Sun MTP が提供する変数も含め、すべての変数が削除可能です。

2. **Enter キーを押します。**

## ▼ 変数を変更する

1. **変更するフィールドに新しい値を入力します。**
2. **Enter キーを押します。**

---

**注 –** 長さが 40 バイト (表示モードが 16 進数の場合は 20 バイト) を超える変数を追加または変更する場合は、30 ページの「Edit Variable 画面」で説明する Edit Variables 画面を使用する必要があります。

---

次の表に、変数の型を示します。

**表 2-1** `CECI` の変数の型

| 型 | 長さ (バイト) | 省略名 | 例 |
|---|---|---|---|
| Halfword | 2 | H | 04321 |
| Fullword | 4 | F | 0001234567 |
| Packed Decimal | 4 | P | 4000010C |
| Doubleword | 8 | D | 00ABCDE126395417 |
| Buffer | 最大 32767 | 長さを表す数値 | Example Buffer |

## Edit Variable 画面

Edit Variables 画面では、ユーザー定義の変数をフル画面モードで編集できます。この画面では、すべての変数タイプを編集できます。変数画面の上限である 40 バイト (表示モードが 16 進の場合は 20 バイト) を超える Buffer 変数を編集する場合は、この画面を使用する必要があります。

## ▼ 変数を編集する

1. Variables 画面で、編集する変数がある行にカーソルを移動します。
2. PF10 キーを押して、Edit Variables 画面を表示します。
3. 変数を変更します。
4. Enter キーを押して、変更を適用します。
5. PF3 キーを押すと、Variables 画面に戻ります。

次の図は、Edit Variables 画面を示します。

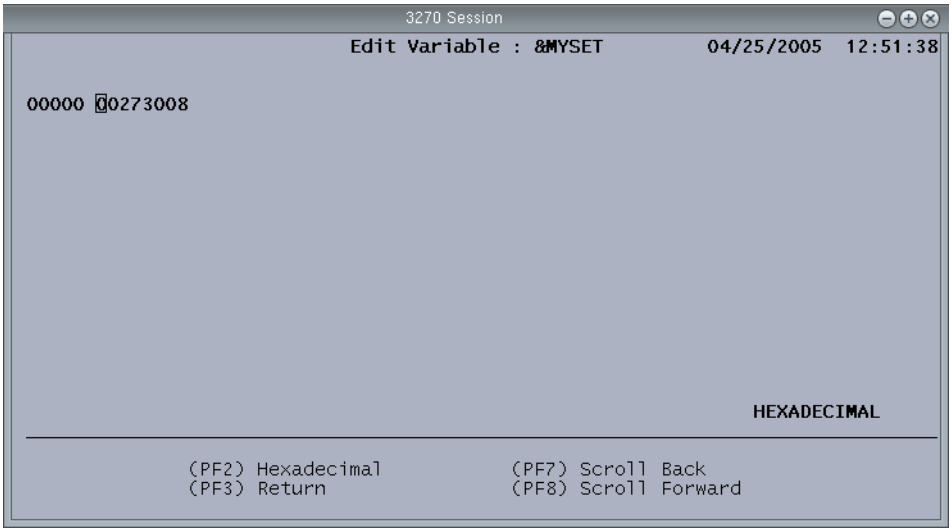


**図 2-10** CECI—Edit Variables 画面

Edit Variables 画面では、次のファンクションキーが利用できます。

| ファンクションキー | アクション |
|---|---|
| PF2 | 表示を文字/10 進 (デフォルト) から 16 進に切り替えます。Decimal/Char とラベルが付いている場合は、16 進から文字/10 進に切り替えます。 |
| PF3 | 前の画面に戻ります。 |
| PF7 | 1 ページ分前のページにスクロールします。 |
| PF8 | 1 ページ分次のページにスクロールします。 |

## ユーザー画面

SEND MAP のようなコマンドを実行すると、それによって 3270 データが端末画面に送られ、CECI が画面を取り込みます。取り込んだ画面を表示するには、Commands 画面または Syntax 画面で PF6 キーを押します。任意のキーを押すと、前の画面に戻ります。

次の図は、次のコマンド文字列の実行後に取り込まれたユーザー画面を示しています。

```
CECI SEND MAP(ACCTMNU) MAPSET(ACCTSET) MAPONLY
```

---

**注 –** ACCTMNU BMS マップは、Sun MTP Primer 領域の ACCTSET BMS マップセットで提供されます。

---

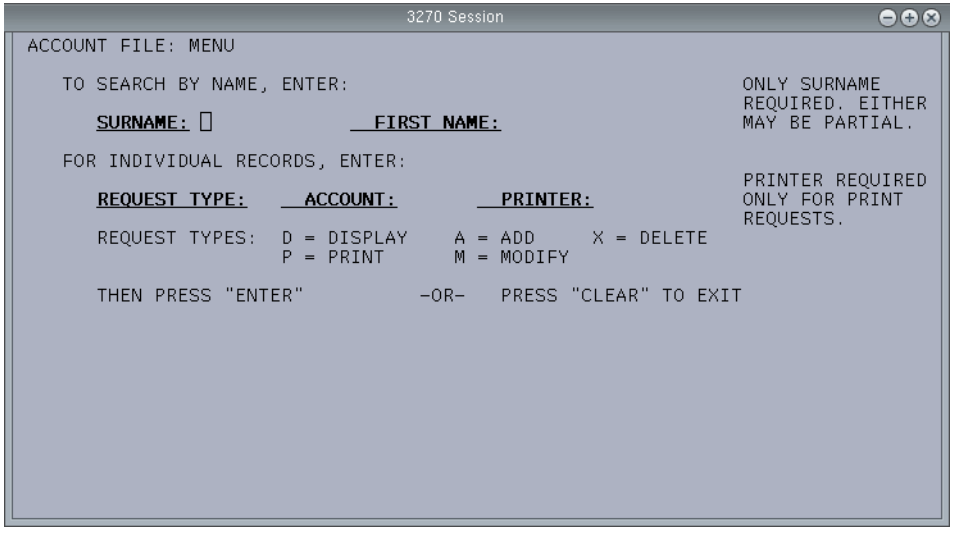


**図 2-11** CECI—ユーザー画面

## メッセージ画面

CECI はコマンドを実行する前に、コマンド行の構文を検査するトランスレータを呼び出します。トランスレータがエラーを検出した場合は、Syntax 画面に次のメッセージが表示されます。

```
KIX3220E Translation Error - see PF9.
```

PF9 キーを押して、診断メッセージを表示できます。

図 2-12 に示す Translator Diagnostic Messages 画面では、誤ったコマンドを入力した場合に表示されるメッセージを示します。

```
CECI STARTBR DAXASET(ACCTIX)
```

DATASET のスペルが DAXASET になっていることに注意してください。また、必要なオプションがありません。

PF3 キーを押すと、前の画面に戻ります。

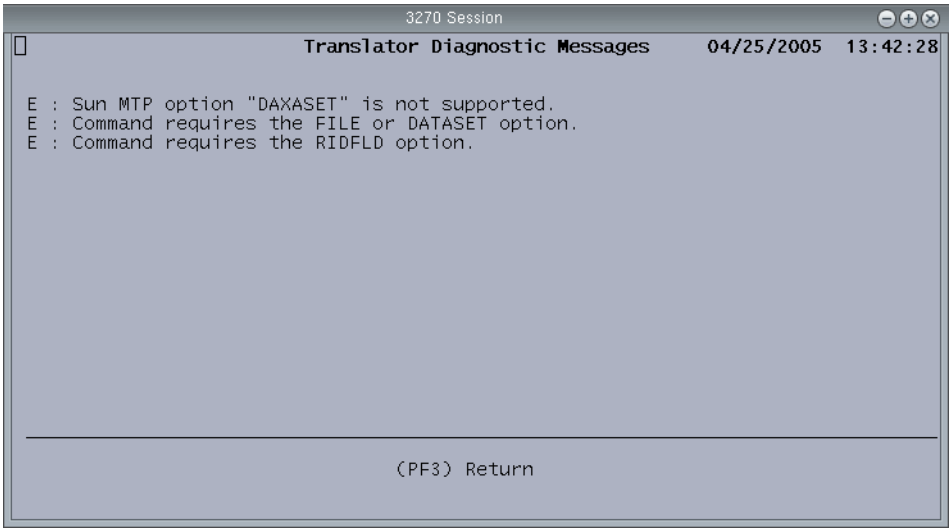


**図 2-12** CECI—Translator Diagnostic Messages 画面

# 動的レシーバ引数

データがオプションフィールドに流れると、CECI はその変数の記憶域を動的に割り当てることができます。したがって、すべての変数を Variables 画面で定義する必要はありません。

---

**注 –** 動的に割り当てられた変数は一時的なので、受け取ったデータをそれ以降のコマンドでは使用できません。また、これらの変数を、Edit Variables 画面で編集することはできません。

---

CECI がレシーバ変数の記憶域を動的に割り当てられるようにするには、コマンド行でオプションを指定する必要があります。このオプションには、次のように、空の括弧を最後に付ける必要があります。

```
CECI READ DATASET(ACCTIX) INTO() RIDFLD(&KEY) LENGTH(63)
```

このコマンドを実行すると、CECI が INTO() オプションにレシーバ変数を動的に割り当てます。

# リテラルオプション引数

CECI 変数を作成せずにデータを渡すもう 1 つの方法として、オプション引数をリテラルとして指定する方法があります。

- リテラルオプション引数は一時的なので、返されたデータをそれ以降のコマンドでは使用できません。また、これらの変数を、Edit Variables 画面で編集することはできません。
- リテラルオプション引数には、空白や括弧を含めることはできません。オプション引数に空白または括弧を含める場合は、CECI 変数を定義する必要があります。

# プログラム制御

この節では、CECI で LINK、XCTL、および ABEND コマンドを使用した場合のプログラム制御ポリシーについて説明します。

| コマンド | ポリシー |
|---|---|
| LINK | LINK コマンドの実行後、制御は CECI に戻されます。 |
| XCTL | XCTL コマンドと現在の CECI セッションが終了しても、制御は戻されません。このリリースではサポートされていません。 |
| ABEND | CANCEL オプションによって、ABEND は CECI セッションを終了します。CANCEL オプションを付けなくても、保護されたリソースの回復も含めて、ABEND 処理は完了します。 |

## 作業論理ユニット (LUW)

CECI は会話型トランザクションで、デフォルトでは、CECI セッションで実行するすべてのコマンドは、1 つの作業論理ユニット (LUW) に含まれています。1 つの CECI セッション内の LUW は、SYNCPOINT と ROLLBACK コマンドを使用して管理できます。

- SYNCPOINT 処理は、各 CECI セッションの終わりに自動的に行われます。
- ROLLBACK 処理は、回復可能なリソースに対する変更を取り消します。

# CEDA – リソースの操作

CEDA トランザクションを使用すると、リソース定義を動的に操作できます。次の節では、このトランザクションのオプションについて説明します。

リソースの動的な管理の詳細については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

---

**注 –** ほとんどの属性値の場合、入力したテキストは大文字に変換されます。その例外は、大文字と小文字の両方がサポートされる GROUP、DESCRIPTION、JVMCLASS、および SHLIB の値です。

---

## DEFINE オプション

CEDA DEFINE オプションを使用すると、リソースを定義できます。現在のリリースでは、マップセット、プログラム、およびトランザクションがサポートされます。

形式:

```
CEDA DEFINE GROUP(group-name)
[MAPSET(mapset-name)[attribute(value)] ... |
PROGRAM(program-name) [attribute(value)]... |
TRANSACTION(transaction-name) [attribute(value)]]
```

| オプション | 説明 |
|---|---|
| GROUP(*group-name*) | このマップセットを追加するグループ名。8 文字以下で、スペースを含みません。グループが存在しない場合は作成されます。大文字と小文字が混在できます。 |

| オプション | 説明 |
|---|---|
| MAPSET(*mapset-name*) | マップセット名。8 文字以下で、スペースを含みません。C で始まるシステム提供のプログラム名およびマップセット名は使用できません。小文字は大文字に変換されます。 |
| PROGRAM(*program-name*) | プログラム名。8 文字以下で、スペースを含みません。C で始まるシステム提供のプログラム名およびマップセット名は使用できません。小文字は大文字に変換されます。 |
| TRANSACTION(*transaction-name*) | トランザクション名。4 文字以下で、スペースは含みません。C で始まるシステム提供のプログラム名およびマップセット名は使用できません。小文字は大文字に変換されます。 |
| *attribute(value)*... | リソースの属性。属性の説明については、次の節を参照してください。 |

## MAPSET 属性

サポートされる MAPSET 属性は次のとおりです。

```
[DESCRIPTION(description)]
[STATUS(ENABLED|DISABLED)]
```

| オプション | 説明 |
|---|---|
| DESCRIPTION(*description*) | マップセットの説明 (58 文字以下)。大文字と小文字が混在できます。 |
| STATUS(ENABLED\|DISABLED) | マップセットの状態を指定します。ENABLED がデフォルトです。 |

**例**: CMP1INV グループにあるマップセット INVADD01 を定義するには、空白のトランザクション画面に次のように入力します。

```
CEDA DEFINE GROUP(CMP1INV) MAPSET(INVADD01)
```

## PROGRAM 属性

サポートされる PROGRAM 属性は次のとおりです。

```
[DESCRIPTION(description)]
[LANGUAGE([ASSEMBLER|C|COBOL|PLI])]
[STATUS(ENABLED|DISABLED)]
```

| オプション | 説明 |
|---|---|
| DESCRIPTION(*description*) | プログラムの説明 (58 文字以下、任意)。大文字と小文字が混在できます。 |
| LANGUAGE ([ASSEMBER\|C\|COBOL\|PLI]) | プログラミング言語の名前。指定しない場合のデフォルトは COBOL です。<br>• ASSEMBLER: CICS LOAD と CICS RELEASE コマンドを使用してアプリケーションプログラムによって参照されるテーブル<br>• C: C 言語<br>• COBOL: COBOL 言語<br>• PL/I: PL/I 言語 |
| STATUS(ENABLED\|DISABLED) | プログラムの状態を指定します。ENABLED がデフォルトです。 |

遠隔属性:

```
[EXECUTIONSET(FULLAPI|DPLSUBSET)]
[REMOTENAME(program-name)]
[REMOTESYSTEM(system-name)]
[TRANSID(transaction-name)]
```

| オプション | 説明 |
|---|---|
| EXECUTIONSET(FULLAPI\|DPL SUBSET) | REMOTESYSTEM 名がローカルシステム名と同じである場合、EXECUTIONSET は、遠隔システムであるかのように、ローカルシステムで DPL プログラムを実行するために指定します。詳細については、第 6 章を参照してください。<br>• FULLAPI: このプログラムは、すべての CICS API コマンドを使用できます。デフォルト値です。<br>• DPLSUBSET: このプログラムは、DPL 制限 API を使用できません。 |
| REMOTENAME(*program-name*) | 遠隔 Sun MTP または CICS システムのプログラムの名前。REMOTENAME を指定しないで REMOTESYSTEM を指定すると、Sun MTP は PROGRAM 名を使用します。詳細については、第 6 章を参照してください。 *program-name* は 8 文字に制限されます。 |

| オプション | 説明 |
|---|---|
| REMOTESYSTEM(*system-name*) | Sun MTP が分散プログラムリンク (DPL) 要求をシップする先の遠隔 Sun MTP または CICS システムを指定する場合の名前。この 4 文字の *system-name* は、「TCT–System Entries」テーブルに存在する必要があります。 |
| TRANSID(*transaction-name*) | 遠隔システムが要求されたプログラムを接続するトランザクションの名前。詳細については、第 6 章を参照してください。 |

Java 仮想マシン (JVM™) 属性

[JVMCLASS(*class-name*)]

JVMCLASS(*class-name*) オプションは、Java プログラムのメインクラスの名前を定義します。最大で 256 文字です。クラス名には大文字小文字が混在できます。

Sun MTP 属性

[PRELOAD(NO|YES)]
[SHLIB(*shared-library-name*)]

| オプション | 説明 |
|---|---|
| PRELOAD(NO\|YES) | 起動時に、トランザクションサーバーが共有ライブラリを開くかどうかを示します。 |
| SHLIB(*shared-library-name*) | 指定したプログラムを含む共有ライブラリの名前 (16 文字以下)。すべてのエントリは、KIXLIB 環境変数がパス名に設定されていない場合、$KIXSYS に相対的になります。共有ライブラリの名前には、大文字小文字が混在できます。 |

**例**: CMP1INV グループにある在庫追加プログラム INVPGM1 を定義するには、空白のトランザクション画面に次のように入力します。

```
CEDA DEFINE GROUP (CMP1INV) PROGRAM (INVPGM1) DESCRIPTION (Inventory add
program)
```

**注 –** コマンドが長くて 1 行に収まらない場合も、自動的に折り返されるので継続文字は不要です。

## TRANSACTION 属性

サポートされる TRANSACTION 属性は次のとおりです。

```
[DESCRIPTION(description)]
[PROGRAM(program-name)]
[STATUS(ENABLED|DISABLED)]
[TASKREQ(value)]
[TWASIZE(0-32767)]
```

| オプション | 説明 |
|---|---|
| DESCRIPTION(*description*) | トランザクションの説明 (58 文字以下、任意)。大文字と小文字が混在できます。 |
| PROGRAM(*program-name*) | プログラム名。小文字は大文字として処理されます。 |
| STATUS(ENABLED\|DISABLED) | トランザクションの状態を指定します。ENABLED がデフォルトです。 |
| TASKREQ(*value*) | プログラムをトランザクションとして開始するキー。このフィールドが空白の場合、プログラムをトランザクションとして開始するキーはありません。有効な値は次のとおりです。<br>PF1 ～ PF24<br>PA1 ～ PA3 |
| TWASIZE(0-32767) | トランザクション作業領域 (TWA) のサイズ (バイト) |

遠隔属性:

```
[LOCALQ(NO|YES)]
[REMOTESYSTEM(system-name)]
[REMOTENAME(program-name)]
```

| オプション | 説明 |
|---|---|
| LOCALQ(NO\|YES) | 遠隔システムが使用できない場合、遠隔システムにシップされたトランザクションをローカルのキューに入れるかを指定します。デフォルトは NO です。 |
| REMOTENAME(*transaction-name*) | 遠隔の Sun MTP 領域または CICS 領域のトランザクションを指定します。REMOTENAME を省略し、REMOTESYSTEM を指定すると、Sun MTP は ローカルシステムのトランザクション ID を使用します。トランザクション ID の長さは 1 ～ 4 文字です。 |
| REMOTESYSTEM(*system-name*) | 遠隔の Sun MTP または CICS システムを指定します。指定した場合、Sun MTP はそこにトランザクションを送信します。4 文字のシステム名は、TCT のシステムエントリテーブルで定義しておく必要があります。 |

スケジューリング属性:

```
[TRANCLASS(transaction-class)]
```

TRANCLASS(*transaction-class*) オプションは、トランザクションを割り当てるトランザクションクラスの名前を指定します。

回復属性:

```
[DUMP(YES|NO)]
```

DUMP(YES|NO) オプションは、トランザクションが異常終了する場合にトランザクションダンプを作成するかどうかを指定します。デフォルトは YES です。

次の属性は、Sun MTP 固有であり、IBM CICS セットにはありません。

```
[SCREENSIZE(DEF|ALT)]
[ACCOUNTING(DEF|NO|YES)]
[FILEID(0-99)]
[APPC(YES|NO)]
[TRANSEC(1-64)]
```

| オプション | 説明 |
|---|---|
| SCREENSIZE(DEF\|ALT) | デフォルト画面サイズまたは代替画面サイズ。<br>DEF: モデル 2 3270 端末タイプのデフォルト画面サイズ 24 行 × 80 列<br>ALT: モデル 5 3270 端末タイプの代替画面サイズ 27 行 × 132 列、またはモデル 4 3270 端末タイプの 43 行 × 80 列 |
| ACCOUNTING(DEF\|NO\|YES) | トランザクションに対してアカウンティングがオンであるかオフであるかを示します。<br>YES: SIT の 「Accounting」フィールドが Y に設定されている場合、トランザクションに対するアカウンティングはオンです。<br>NO: このトランザクションに対するアカウンティングはオフです。<br>DEF: (デフォルト) アカウンティングは SIT の「Accounting」フィールドおよび MCT によって制御されます。 |
| FILEID(0-99) | トランザクションに対するジャーナルファイル識別子で、1 ～ 99 の数。アカウンティングが N に設定されている場合、このフィールドは無視されます。<br>ファイル ID が JCT で指定されていない場合、ジャーナルファイル名は、ファイル ID 値と JRNL を結合して作成されます。たとえば、JRNL05 という名前のジャーナルファイルが作成されます。 |

| オプション | 説明 |
| --- | --- |
| APPC(YES\|NO) | バックエンドの DTP トランザクション。<br>YES: トランザクションは、バックエンドの DTP プログラムで、ISC 機能を実行します。<br>NO: トランザクションは、バックエンドの DTP プログラムではありません。アウトバウンド ISC 機能は実行するが、遠隔 Sun MTP または CICS 領域によって開始された場合、想定されたようには実行されません。 |
| TRANSEC(1-64) | トランザクションのセキュリティーレベルを指定する 1〜64 の数。この値は、トランザクションを入力した端末のユーザーに関連付けられた SNT のセキュリティーキー領域に設定されるビット値セットに対応します。<br>外部セキュリティー管理が使用可能な場合 (KIXSEC=YES)、「Trans Sec」の値は無視されます。<br>ユーザーが CSSN (CESN) を使用して明示的にサインオンするとき、あるいは Sun MTP が起動して入力したユーザー ID と SNT のユーザー ID が一致した場合に暗黙的にサインオンするときに、セキュリティーキーが SNT によってユーザーに対して設定されます。デフォルトでは、ユーザーはセキュリティーレベル 1 のトランザクションのみを入力できます。すべてのシステムトランザクションは、トランザクションセキュリティーレベルが 1 に最初は設定されます。このため、すべてのユーザーはトランザクションにアクセスできます。<br>CSMT、CEMT などの重要なトランザクションや、Sun MTP 環境の設定に使用されるその他のトランザクションのセキュリティーレベルを上げて、権限のないユーザーがアクセスできないようにします。Sun MTP セキュリティーの詳細は、第 7 章を参照してください。 |

## DELETE オプション

DELETE オプションは、リソースを削除します。

形式:

CEDA DELETE ALL | [MAPSET(*mapset-name*) | PROGRAM(*program-name*) | TRANSACTION(*transaction-name*)] GROUP(*group-name*)

| オプション | 説明 |
|---|---|
| ALL | グループ内のすべてのエントリを削除します。グループ定義も合わせて削除します。 |
| GROUP (*group-name*) | リソースを削除するグループの名前。 |
| MAPSET(*mapset-name)* \| PROGRAM(*program-name*) \| TRANSACTION(*transaction-name*) | 削除するリソース (マップセット、プログラムまたはトランザクション)。汎用リソース名を指定できます。最後のリソースまたは最後のリソースのグループを指定した場合は、GROUP 定義も削除されます。 |

**例**: CMP1INV グループにあるマップセット INVADD01 を削除するには、空白のトランザクション画面に次のコマンドを入力します。

```
CEDA DELETE GROUP(CMP1INV) MAPSET(INVADD01)
```

## DISPLAY オプション

DISPLAY オプションは、1 つ以上のリソースを表示します。

形式:

CEDA DISPLAY ALL | [MAPSET(*mapset-name*) | PROGRAM(*program-name*) | TRANSACTION(*transaction-name*)] GROUP(*group-name*)

| オプション | 説明 |
|---|---|
| ALL | すべてのリソースとキーワード GROUP によって指定したグループのグループ名のリストを表示します。 |
| GROUP (*group-name*) | グループ名だけを指定した場合、グループのリストが表示されます。総称名を指定できます。 |
| MAPSET(*mapset-name)* \| PROGRAM(*program-name*) \| TRANSACTION(*transaction-name*) | キーワード GROUP によって指定したグループで、指定された名前に一致するすべてのリソース (マップセット、プログラム、トランザクション) のリストを表示します。汎用リソース名を指定できます。 |

**例**: CMP* グループのすべてのリソースを表示するには、空白のトランザクション画面に次のように入力します。

```
CEDA DISPLAY GROUP(CMP*) ALL
```

## INSTALL オプション

INSTALL オプションを使用すると、領域の実行中に、領域のリソース定義の追加や置き換えができます。

形式:

CEDA INSTALL GROUP(*group-name*)
[MAPSET(*mapset-name*)]
[PROGRAM(*program-name*)
[TRANSACTION(*transaction-name*)]

| オプション | 説明 |
|---|---|
| GROUP (*group-name*) | 実行中の領域に動的に追加するリソースが存在するグループの名前。総称名は指定できません。 |
| MAPSET(*mapset-name*)<br>PROGRAM(*program-name*)<br>TRANSACTION(*transaction-name*) | 実行中の領域に動的に追加するリソースの名前。総称名を指定できます。省略した場合は、GROUP パラメータで指定したグループからすべてのリソースが追加されます。 |

**例**: CMP1INV グループのすべてのリソースをインストールするには、空白のトランザクション画面に次のコマンドを入力します。

```
CEDA INSTALL GROUP (CMP1INV)
```

# CEDF – デバッグ機能の実行

CEDF トランザクションを使用して、Sun MTP デバッグ機能にアクセスするか、ソースデバッガを選択します。デバッグ機能は、COBOL、C、および PL/I プログラムに埋め込まれた EXEC CICS 文のデバッグに使用します。CEDF は、ブレークポイント (デバッグのために制御を中断して、トランザクションサーバーに渡す位置) の設定について、数多くのオプションを提供しています。CEDF では、1 つまたは複数の端末セッションのデバッグと経路指定されたトランザクションをデバッグできます。

CEDF トランザクション、デバッグ、およびブレークポイント処理の詳細については、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。

---

**注 –** ソースコードデバッガを使用している場合は、ほかのシステムトランザクションを実行する前に、CEDF トランザクションを使用してソースコードデバッガをオフにします。

---

# CEMT – 状態の設定と表示

CEMT トランザクションを使用して、次のシステム保守作業ができます。

- フォアグラウンドタスクとバックグラウンドタスクの現在の状態の表示
- トランザクションクラスの表示と、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の変更
- プログラム、マップセット、または共有ライブラリのロード
- 特定のタスクまたは全部のタスクのパージ (タスクがループに入り込んだときなど)
- 3270 端末および遠隔システムとの接続のサービスの開始と終了
- トランザクションの状態の変更
- システム状態の変更
- 領域の停止

CEMT を実行するには、画面の左上にトランザクション識別子とその引数を入力して、Enter キーを押します。

形式:

CEMT INQ|SET|PERFORM *arguments*

| オプション | 説明 |
|---|---|
| I[NQ] | 照会を行います。詳細は、45 ページの「INQ (I) オプション」を参照してください。 |
| S[ET] | 操作環境を制御します。詳細は、49 ページの「SET (S) オプション」を参照してください。 |
| P[ERFORM] | システムアクティビティーを実行します。詳細は、55 ページの「PERFORM (P) オプション」を参照してください。 |
| *arguments* | 引数は実行するオプションによって異なります。 |

# INQ (I) オプション

INQ オプション (省略名は I) は、指定したタスクまたはキューの現在の状態を表示します。

形式:

```
CEMT I[NQ] [TASK ALL | (taskno)] |
[TRANID (tranid)] |
[FACILITY (facid)] |
[ACTIVE] |
[SUSPENDED] |
[TERM] |
[DEST] |
[CONN] |
[TDQUEUE | ALL] |
[PROG[RAM] (prog-name) LIB[rary]] |
[TDQUEUE (queue-id) | ALL] |
[TRANCLASS (class-name) | ALL]
```

次のコマンドは、状態を表示します。

| コマンド | 表示内容 |
|---|---|
| CEMT INQ TASK | 現在ログオンしているすべての端末とバックグラウンドプロセス。デフォルトは ALL です。 |
| CEMT INQ TASK ALL | 現在ログオンしているすべての端末とバックグラウンドプロセス。 |
| CEMT INQ TASK *taskno* | 指定したタスク番号を持つタスクのみ。タスク番号を確認するには、TASK ALL オプションを使用します。 |
| CEMT INQ TRANID *tranid* | 指定したトランザクション識別子を実行するタスクのみ。 |
| CEMT INQ FACILITY *facid* | 指定した機能識別子 (端末開始タスクの端末識別子およびスタート時またはトリガー時の起動タスクのトランザクション識別子) を持つタスクのみ。 |
| CEMT INQ ACTIVE | 現在実行中のすべてのタスク。 |
| CEMT INQ SUSPENDED | 現在中断され、リソースの利用が保留になっているすべてのタスク。 |
| CEMT INQ TERM | 開始されたタスク。 |
| CEMT INQ DEST | トリガーで開始された、すべてのタスク。 |
| CEMT INQ CONN | すべての遠隔システムの接続。 |

| コマンド | 表示内容 |
|---|---|
| CEMT INQ TDQUEUE *queue-id* | 指定した一時データキュー。*queue-id* はキューを特定する 4 文字の値です。 |
| CEMT INQ TDQUEUE ALL | すべての一時データキュー。 |
| CEMT INQ PROG[RAM] *prog-name* LIB[RARY] | 現在の共有ライブラリ名。プログラムが共有ライブラリを使用している場合、トランザクションが次のようにライブラリ名を表示します。<br>CEMT transaction terminated ShrLib = libc<br>使用していない場合、トランザクションが次のようにライブラリ名を表示します。<br>NoSharedLibPrsnt |
| CEMT INQ TRANCLASS (*class-name*)\| ALL | Max Active、現在の Max Active (この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数)、受信メッセージまたはトランザクション数、現在のキューの深さ、およびキューの最大の深さ。表示例については、図 4-5 を参照してください。 |

CEMT INQ TASK ALL などの CEMT INQ トランザクションをタスクに対して実行すると、「User Status Table」が表示されます。

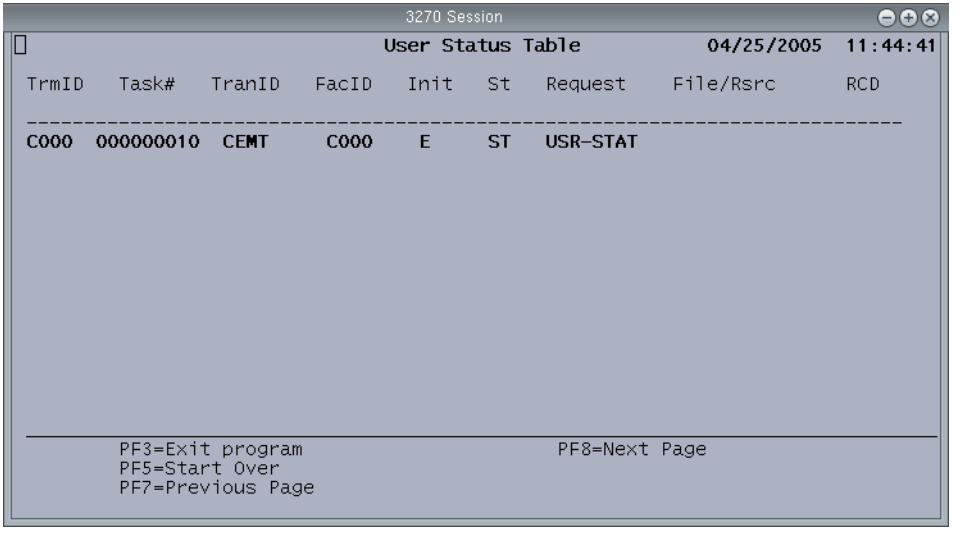


図 2-13 CEMT INQ TASK ALL の表示

「User Status Table」には、次のフィールドが含まれます。

| フィールド | 説明 |
|---|---|
| TrmID | タスク (トランザクション) に関連付けられている端末識別子を示します。このフィールドが空白の場合、トランザクションは端末に関連付けられていません。 |
| Task# | タスク番号を示します。各タスクには、システムで開始されたときに一意のシーケンシャル番号が割り当てられます。 |
| TranID | トランザクションを示します。 |
| FacID | 機能を示します。これは、端末開始タスクの端末識別子であるか、非同期で起動されたタスクまたはトリガーされたタスクを開始したトランザクションのトランザクション識別子です。 |
| Init | 起動のタイプを示します。次のいずれかの値が表示されます。<br>E: 端末<br>T: 起動<br>D: トリガー (一時データキュー)<br>C: 遠隔システム接続 |
| St | タスクの現在の状態を示します。次のいずれかの値が表示されます。<br>RQ: レディキュー上に存在<br>WB: I/O バッファーを待機<br>WR: リソースを待機<br>VI: VSAM アイドル<br>ST: 状態要求の実行中<br>ID: アイドル<br>EQ: イベントキュー上に存在<br>IN: サービス中の遠隔システムへの接続<br>OU: サービス中ではない遠隔システムへの接続 |
| Request | 実行されている CICS コマンドを示します。 |
| File/Rsrc | 要求されているファイルまたはリソースを示します。 |
| RCD | 要求されている物理レコード番号を示します。 |

CEMT INQ TDQUEUE ALL などの CEMT INQ トランザクションを一時データキューに対して実行すると、「Destination Control Table Information」画面が表示されます。

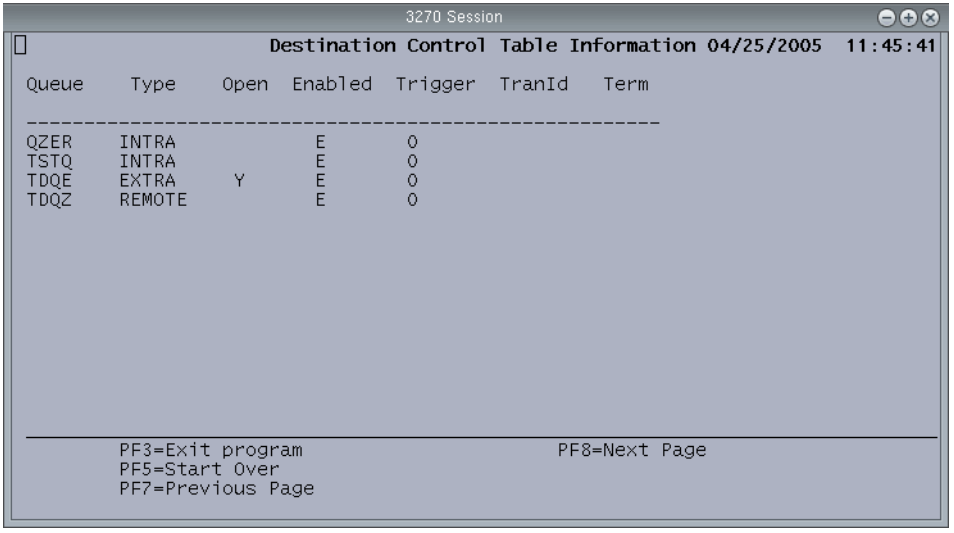


**図 2-14** CEMT INQ TDQUEUE ALL の表示

「Destination Control Table Information」画面には、次のフィールドが含まれます。

| フィールド | 説明 |
|---|---|
| Queue | キューの名前を示します。 |
| Type | キューのタイプを示します。次のいずれかの値が表示されます。<br>EXTRA: パーティション外キュー<br>INTRA: パーティション内キュー<br>REMOTE: 遠隔キュー |
| Open | パーティション外キューについて、このキューが開いているかどうかを示します。値は次のとおりです。<br>Y: キューは開いています。<br>N: キューは閉じています。 |
| Enabled | すべてのキューについて、キューが有効になっているかどうかを示します。値は次のとおりです。<br>Y: キューは有効化されています。<br>N: キューは無効化されています。 |
| Trigger | 次のいずれかを示します。<br>パーティション内キューのトリガーレベル<br>その他のキューのタイプの場合はゼロ (0) |
| TranId | 次のいずれかを示します。<br>パーティション内キューに関連付けられたトランザクション識別子<br>その他のキューのタイプの場合は空白 |

| フィールド | 説明 |
|---|---|
| Term | 次のいずれかを示します。<br>パーティション内キューに関連付けられた端末の名前<br>その他のキューのタイプの場合は空白 |

# SET (S) オプション

SET オプション (省略名は S) では、領域面を管理できます。SET オプションのあとのパラメータによって、アクションが決まります。

## リソースの新しいコピーのロード

プログラムまたはマップセットの新しいコピーをロードするには、次のトランザクションをサブミットします。

```
CEMT SET PROGRAM (resource-name) NEWCOPY
```

次回、プログラムまたはマップが参照されると、LOAD、LINK、SEND MAP、XCTL などのコマンドは、新しいバージョンに対して実行されます。

新しいプログラムやマップセットを導入する場合には、CEMT . . . NEWCOPY トランザクションの方が、CINI よりも適切です。

CEMT . . . NEWCOPY を使用してプログラムをロードすると、すべてのプログラムがフラッシュされます。次にプログラムが、適切なディレクトリからロードされます。したがって、領域が開始されたあと、またはプログラムに対する CINI あるいは CEMT トランザクションが実行されたあとに更新されたプログラムは、それが次回参照されたときにメモリーに読み込まれます。

次の例は、CEMT によるプログラムの処理を示しています。

1. $KIXPROGS ディレクトリに PROG1 バージョン 1 と PROG2 バージョン 1 があるとします。ユーザーが PROG1 と PROG2 を要求すると、それらがトランザクション処理プログラムのローカルメモリーにロードされます。
2. ここで管理者が PROG2 をバージョン 2 に更新したとします。
3. 管理者が CEMT SET PROGRAM (PROG1) NEWCOPY を実行すると、トランザクション処理プログラムが再起動します。これにより、すべてのプログラムがプロセッサのメモリーから削除されます。この結果、新しいコピーへの要求が 1 つのプログラムに対してであっても、ユーザーにはすべてのプログラム、たとえば PROG1 バージョン 1 および PROG2 バージョン 2 が、新しいコピーとして認識されます。

マップセットをロードすると、CEMT ． ． ． NEWCOPY によって、新しいマップセットのみが $KIXMAPS ディレクトリから $KIXSYS/_kix_reserved_maps ディレクトリ、または _KIX_RESERVED_MAPS 環境変数で指定されるディレクトリにコピーされます。$_KIX_RESERVED_MAPS 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。

次の例は、CEMT によるマップセットの処理を示しています。

1. $KIXMAPS ディレクトリに MAP1 バージョン 1 と MAP2 バージョン 2 があるとします。アプリケーションがマップセット MAP1 を要求すると、それが $KIXSYS/_kix_reserved_maps ディレクトリにコピーされ、共有メモリーにロードされます。
2. ここで管理者が MAP1 をバージョン 2 に更新したとします。
3. 管理者が CEMT SET PROGRAM (MAP1) NEWCOPY を実行すると、MAP1 バージョン 1 が共有メモリーから削除され、$KIXSYS/_kix_reserved_maps ディレクトリから削除されます。MAP1 バージョン 2 が $KIXSYS/_kix_reserved_maps ディレクトリにコピーされ、共有メモリーにロードされます。

CICS と異なり、プログラムに対して CEMT ． ． ． NEWCOPY を実行するとき、RESCOUNT のチェックは行われません。

本番システムでプログラムのバージョンを変更するときには、注意が必要です。プログラムが強制的に中止されたために (たとえば、ローカルプロセスの限界を超えてメモリーにアクセスしようとした場合) トランザクション処理プログラムが再起動すると、変更したバージョンのプログラムがそのトランザクション処理プログラムに対してだけロードされます。この結果、再起動したトランザクション処理プログラムは、新しいバージョンのプログラムを使用し、ほかのトランザクション処理プログラムはプロセスローカルメモリーにある古いバージョンのプログラムを使用することになります。

## 共有ライブラリまたは共有オブジェクトのロード

領域の実行中に、新しい共有ライブラリまたは共有オブジェクトを動的にロードする場合、または指定したプログラムを処理から削除する場合は、次のトランザクションを使用します。

CEMT SET PROGRAM *prog-name* LIBRARY *lib-name*

16 文字以下で指定した共有ライブラリまたは共有オブジェクトをロードします。

CEMT SET PROGRAM *prog-name* LIBRARY blankreset

指定したプログラムを共有ライブラリまたは共有オブジェクトの処理から削除します。キーワード blankreset により、このプログラムの Program Processing Table (PPT) の共有ライブラリフィールドが空きに設定されます。

**注 –** キーワード `blankreset` は正確に入力してください。誤っていると新しい共有ライブラリ名または共有オブジェクト名として使用されます。

どちらかの `CEMT` トランザクションを実行してから、`CINI` トランザクションを実行して新しい共有ライブラリまたは共有オブジェクトと内部テーブルを初期化する必要があります。

共有ライブラリおよび共有オブジェクトの詳細については、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。

## プログラムとトランザクションの有効化または無効化

プログラムやトランザクションを動的に有効化、または無効化するには、次のトランザクションを使用します。

`CEMT SET PROGRAM` *prog-name* `[DISABLED|ENABLED]`

`CEMT SET TRANSACTION` *tranid* `[DISABLED|ENABLED]`

`DISABLED` は `D` に、`ENABLED` は、`E` に省略できます。

`CEMT SET TRANSACTION DISABLED` コマンドを使用する場合、実行中のトランザクションは、処理を完了してから無効になります。

## トランザクションダンプの要求

次のトランザクションによって、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作成するかどうか指定できます。

`CEMT SET TRANSACTION` *tranid* `[TRANDUMP|NOTRANDUMP]`

| | |
|---|---|
| `TRANDUMP` | ダンプを作成します。 |
| `NOTRANDUMP` | ダンプを作成しません。 |

**注 –** このトランザクションは、Sun MTP に固有のものです。

## タスクの終了

タスクを終了するには、次のどちらかのトランザクションを使用します。

CEMT SET TASK *taskno* PURGE

特定のタスク番号 (*taskno*) を持つタスクのみをパージします。タスク番号を確認するには、CEMT INQ TASK ALL トランザクションを使用します。

CEMT SET TASK ALL PURGE

すべてのタスクをパージします。

## 一時データキューの管理

ここで説明するトランザクションを使用して一時データキューを管理すると、次の作業ができます。

- パーティション外キューのオープンとクローズ
- あらゆるタイプの一時データキューの有効化または無効化
- パーティション内キューに対するトリガーの設定

起動時にすべてのデータキューが有効になり、動的に無効にできます。実行時の変更は、ディスクに書き込まれません。したがって、領域が停止すると、キューの状態やトリガーレベルは、DCT に定義された元の状態に戻ります。

CEMT SET TDQUEUE (*queue-id*) [OPEN | CLOSED]

指定したキューをオープンまたはクローズします。パーティション外キューにのみ適用されます。
*queue-id* は、キューを示す 4 文字の値です。

CEMT SET TDQUEUE (*queue-id*) [ENABLED | DISABLED]

指定したキューを有効または無効にします。一時データキューのすべてのタイプに適用されます。 *queue-id* は、4 文字のキュー識別子です。

OPEN または CLOSED と DISABLED は同時に使用できません。

OPEN または CLOSED と ENABLED は同時に使用できます。

無効のキューは開くことも閉じることもできません。

開いているキューは無効にできません。

無効のキューとの間の読み書きまたは無効のキューの削除を行うと、トランザクションが強制的に中止し、使用不可の状態に戻ります。

CEMT SET TDQUEUE (*queue-id*) [TRIGGERLEVEL (*integer*)]

トランザクションが開始される前に許容される出力要求の数。レベルは、0 ～ 32767 の任意の整数です。パーティション内キューにのみ適用されます。 *queue-id* は、4 文字のキュー識別子です。

トランザクションを実行するときに、キュー ID やトリガーレベルに対する変数は括弧で囲んでも囲まなくてもかまいません。次に例を示します。

```
CEMT SET TDQUEUE (queue-id) TRIGGERLEVEL (integer)
```

```
CEMT SET TDQUEUE queue-id TRIGGERLEVEL integer
```

---

**注 –** *queue-id* に ALL を指定しないでください。

---

コマンドを再度実行するには、画面をクリアしてから、Enter キーをもう 1 度押す代わりに、CEMT SET TDQUEUE トランザクションを入力します。

## 3270 デバイスの制御

次の CEMT SET TERMINAL トランザクションを使用して、3270 デバイスを制御できます。*trmid* 変数は、アプリケーションが認識する 4 文字の端末 ID で、端末管理テーブル (TCT) の「3270 デバイス」画面で定義します。

---

**注 –** 次のトランザクションは、SNA サーバーがインストールされ、3270 デバイスが TCT で定義されている場合のみ使用できます。

---

```
CEMT SET TERMINAL trmid INSERVICE
```

指定した端末 ID の端末を領域にログオンします。

```
CEMT SET TERMINAL trmid OUTSERVICE
```

指定した端末を領域からログオフします。

## システム間接続の制御

次の CEMT SET CONNECTION トランザクションを使用して、システム間接続を制御できます。*sysid* 変数は、領域が認識する 4 文字の遠隔システム ID で、TCT の「システムエントリ」画面で定義します。

```
CEMT SET CONNECTION sysid INSERVICE
```

指定した遠隔システムへの接続を「稼動中」に設定します。

```
CEMT SET CONNECTION sysid OUTSERVICE
```

指定した遠隔システムへの接続を「停止中」に設定します。どの ISC 機能もこの接続を使用できません。

## トランザクション処理プログラムのトランザクションクラスへの割り当て

クラスに対する最大アクティブトランザクション処理プログラムを動的に変更できます。何らかのアクションを取る前に、パラメータの正確さと一貫性が検査されます。たとえば、MAXACTIVE を、設定されたトランザクション処理プログラムの数より大きな値に変更できません。

CEMT SET TRANCLASS *class-name1* MAXACTIVE *value* [[USE] *class-name2*]

指定したクラスに対する最大アクティブトランザクション処理プログラムを変更します。オプションの USE 節を使用すると、トランザクション処理プログラム数を増減させるクラスを指定できます。USE 節を指定しない場合は、KIXDFLT クラスのトランザクション処理プログラム数が増減されます。

トランザクションクラスの設定と使用については、第 4 章を参照してください。

## システム状態の設定

システム状態は、次のトランザクションを使用して設定します。

CEMT SET SYSTEM NORMAL

システム状態を NORMAL に設定します。

CEMT SET SYSTEM BATCH

システム状態を BATCH に設定します。

CEMT SET SYSTEM QUIESCE

システムを QUIESCE 状態に設定します。

次の表は、システム状態と停止ごとに、領域に課せられる条件と制約を示します。

**表 2-2**　システム状態

| 状態 | 新規ログオン | バッチプログラム | 新規トランザクション |
|---|---|---|---|
| NORMAL | 許可 | 許可 | 許可 |
| BATCH | 不許可 [1] | 許可 [2] | 不許可 [1] |
| QUIESCE | 不許可 [1] | 不許可 | 不許可 [1] |
| SHUT | 不許可 [1] | 不許可 | 不許可 [3] |
| SHUT IMM | 不許可 | 不許可 | 不許可 |

[1] Operator Class が SYS に設定されたユーザーは、このアクションを実行できます。『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』のサインオンテーブルに関する節を参照してください。

[2] システムが BATCH 状態の場合、バッチプログラムを CBCH トランザクションで実行できません。

[3] 適切なトランザクションセキュリティーキーを持つユーザーは、CEMT、CSMT、および CPLT のシステムトランザクションを実行できます。

# PERFORM (P) オプション

PERFORM オプション (省略名は P) は、次の処理に使用します。

- 領域を正常に停止します。
- 領域に関する外部セキュリティーマネージャーの結果キャッシュをクリアします。

形式:

```
CEMT  P[ERFORM]SHUT[DOWN]  [IMM[EDIATE]]

CEMT  P[ERFORM]  SECURITY  REBUILD
```

説明:

```
CEMT  PERFORM  SHUTDOWN
```

領域を SHUT 状態に設定します。新しいアクティビティーは開始できません。表 2-2 は、領域がこの状態のときに実行できるアクティビティーを示します。

```
CEMT  PERFORM  SHUTDOWN  IMMEDIATE
```

領域全体を即座に停止します。すべてのユーザーセッションとトランザクションが終了されます。処理中のトランザクションはロールバックされ、回復プロセスが開始されます。

CEMT PERFORM SECURITY REBUILD

アクセス要求の結果が保存されているキャッシュメモリー領域をクリアします。詳細は、204 ページの「セキュリティーアクセス結果のキャッシュ」を参照してください。

# CINI – トランザクション処理プログラムの再初期化

CINI トランザクションは、システムの各トランザクションサーバーに対して、現在実行中のトランザクション終了後、速やかに再起動させます。このとき、ローカルメモリーからすべてのプログラムを、また共有メモリーからすべてのマップセットをクリアし、$KIXSYS/_kix_reserved_maps (または $_KIX_RESERVED_MAPS) ディレクトリからすべてのマップセットを削除します。CINI トランザクションを実行すると、ユーザーによる要求時に、$KIXPROGS からプログラムが、また $KIXMAPS からマップセットがロードされます。マップセットのコピーは、$KIXSYS/_kix_reserved_maps ディレクトリにも配置されます。

CINI トランザクションには引数がありません。

# CRTE – 経路指定トランザクション

トランザクションの経路指定を行うには、CRTE トランザクションを使用します。トランザクション経路指定では、あるシステム上で接続されている端末がほかのシステムでトランザクションを実行できます。CRTE トランザクションの使用については、143 ページの「トランザクション経路指定」を参照してください。

# CSMT – ユーザーセッションまたは領域の停止

CSMT トランザクションを使用して、ユーザーセッションの終了や領域の停止を行います。次のいずれかの形式で、トランザクション識別子とその引数を入力します。

CSMT SHUT[,NO]

ユーザーセッションを終了します。

CSMT SHUT,YES

領域を停止し、すべてのユーザーを強制的にログオフします。有効なトランザクションはロールバックされます。

---

**注 –** CSMT SHUT,YES を正しく動作させるためには、コンマと YES の間に空白を入れないようにします。

---

トランザクションのセキュリティーが有効な場合、ほかのユーザーのサインオン中にユーザーが誤ってシステムを停止しないよう、CSMT トランザクションをセキュリティー保護する必要があります。このトランザクションがセキュリティー保護されている場合は、ユーザーセッションを CSSF GOODNIGHT または CSSF LOGOFF コマンドで終了します。詳細は、61 ページの「CESF および CSSF – 領域のサインオフ」を参照してください。

# CSPG – データのページの表示

CSPG トランザクションを使用して、複数ページにわたるデータを CICS BMS の端末に表示します。データのページは、SEND TEXT ACCUM または SEND MAP ACCUM コマンドによって作成されます。すべてのデータの作成後、SEND PAGE コマンドが実行されます。データのページが複数にわたる場合は、CSPG トランザクションが起動して、データ表示を制御します。

CSPG が開始されると、データの最初のページが表示されます。画面上の保護されていない最初のフィールドにコマンドを入力するか PF キーを使用して、ページを前後に移動したり、特定のページを表示したり、終了したりします。

コマンド文字列の形式

```
P/[+n | -n | n | P | N | L | C]
```

P/ コマンドの開始を示します。

- *n*: *n* で指定されたページを表示します。*n* の前に + または - が付いている場合、現在のページ番号からその数が増減されます。
- P: 前のページを表示します。
- N: 次のページを表示します。
- L: 最後のページを表示します。
- C: 現在のページを表示します。

PF キーを使用して表示するページを制御するには、PF キーをシステム初期化テーブル (SIT) で定義する必要があります。デフォルトでは、次の PF キーが定義されています。

| ファンクションキー | アクション |
|---|---|
| PF3 | 終了します。 |
| PF7 | 最後のページを表示します。 |
| PF8 | 次のページを表示します。 |

# CESN および CSSN – 領域へのサインオン

次の 2 つのサインオントランザクションが用意されています。

CESN

ユーザー ID とパスワードを使用するか、必要に応じてグループを使用して領域にサインオンします。また、パスワードの変更も可能です。

CSSN

オペレータ名とパスワードで領域にサインオンします。

Sun MTP の組み込みセキュリティーを使用している場合は、CESN および CSSN トランザクションは、SNT を使用してユーザー ID とパスワードを検査します。ユーザープログラムは、EXEC CICS ASSIGN コマンドを使用して SNT フィールドにアクセスできます。

Sun MSF などの外部セキュリティーマネージャーを使用している場合は、CESN トランザクションは ESM のリポジトリを使用してユーザーの ID、パスワード、グループ (指定された場合) を検査します。

## CESN

**注 –** CESN トランザクションをサインオン済みの端末から使用すると、前のユーザーがすぐにサインオフされます。

CESN だけを入力するか、データと合わせて入力します。CESN だけを入力した場合は、図 2-15 の画面が表示されます。

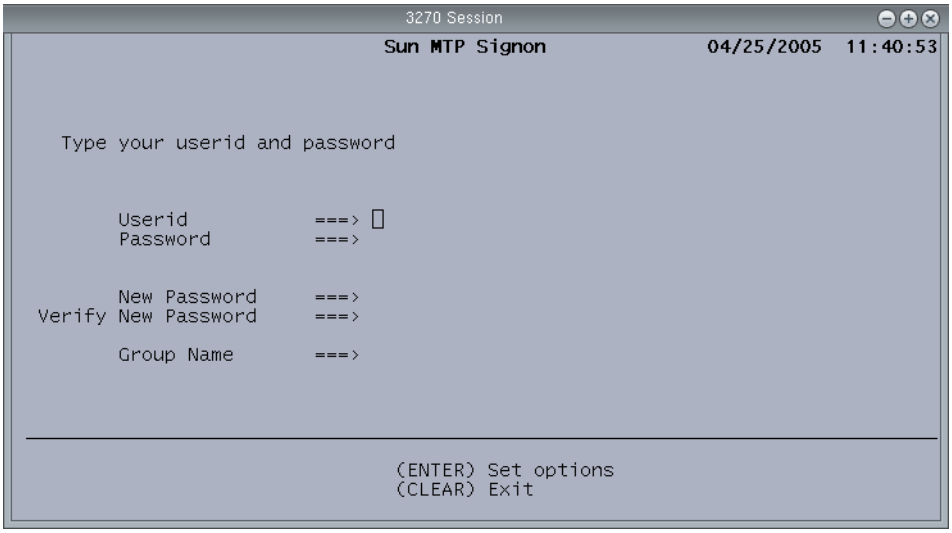


**図 2-15** CESN Signon 画面

パスワードと新しいパスワードを入力するときは、その値は表示されません。パスワードを変更するには、新しいパスワードを「New Password」フィールドと「Verify New Password」フィールドに入力します。「Userid」、「Password」、「New Password」の値は、1 ～ 8 文字である必要があります。パスワードは大文字小文字を区別します。Sun MTP の組み込みセキュリティーを使用している場合は、システム初期化テーブル (SIT) のエントリによって有効なパスワードの形式が定義されます。Sun MSF などの外部セキュリティーマネージャー (ESM) を使用している場合は、有効なパスワードの形式はその ESM によって定義されます。たとえば、Sun MSF の場合は、パスワードの形式は `MSFconfig.properties` ファイルで定義されます。

「Group Name」フィールドは、ESM が有効になっているときに使用できます。Sun MSF を使用している場合は、このフィールドに役割名を入力できます。ユーザーが認証されており、その役割が許可されていることが Sun MSF によって検証されると、その役割の権限がユーザーに割り当てられます。

CESN トランザクション識別子を使用してすべてのデータを入力する場合は、次のいずれかの形式を使用します。

組み込みセキュリティー:

```
CESN USERID=userid,PS=password[,NEWPS=newpassword]
```

Sun MSF のセキュリティー:

```
CESN USERID=userid,PS=password[,NEWPS=newpassword][,GROUPID=rolename]
```

## CSSN

CSSN だけを入力するか、オペレータ名およびパスワードと合わせて入力します。CSSN だけを入力した場合は、図 2-16 の画面が表示されます。

---

**注 –** ESM が有効になっているときは (KIXSEC=YES)、CSSN トランザクションは使用可能ですが、お勧めできません。これは、ESM リポジトリの主体に相当する、SNT のオペレータ名とユーザー ID の組み合わせを保持しなければならないためです。

---

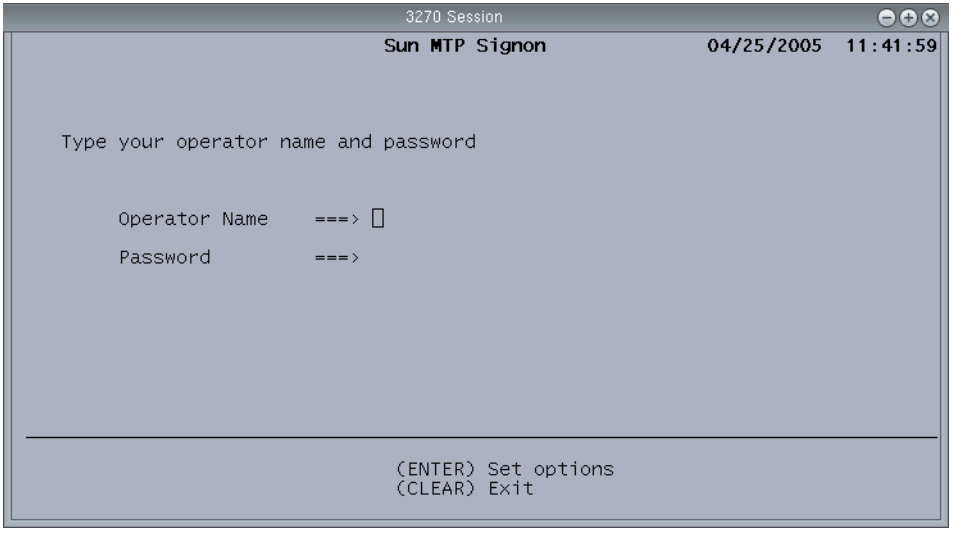


**図 2-16** CSSN Signon 画面

パスワードは大文字小文字を区別します。また、SIT で定義されている形式に従う必要があります。

**注 –** ログインできず、X 端末でセッションを終了する必要がある場合は、Ctrl+C キーを押してセッションを終了してください。

CSSN トランザクション識別子を使用してデータを入力する場合は、次の形式を使用します。

CSSN NAME=*operator-name*,PS=*password*

## CESF および CSSF – 領域のサインオフ

次の 2 つのサインオフトランザクションが用意されています。

CSSF または CESF

サインオフします。必要に応じてクライアントを終了できます。

ログオフするには、次のいずれかのトランザクションを入力します。

CSSF または CESF

ユーザーセッションをサインオフします。

CSSF GOODNIGHT または CSSF LOGOFF

ユーザーセッションをサインオフして、クライアントを終了します。

# CSSN/CESN トランザクションのカスタマイズ

付属の CSSN/CESN トランザクションはカスタマイズできます。サンプル COBOL プログラム CSSNCESN.cl2 は、$UNIKIX/src/trans ディレクトリにあります。

## ▼ トランザクションをカスタマイズする

1. **プログラム管理テーブル (PCT) でトランザクションを定義します。**

   グループ名は使用しません。

   PCT 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。

2. **$UNIKIX/src/trans から $KIXPROGS ディレクトリへ CSSNCESN.cl2 プログラムをコピーします。**

3. **プログラムを変更し、コンパイルします。**

4. **領域を再起動します。**

   標準のトランザクションに代わって、指定したトランザクションが実行されます。

---

**注 –** システムトランザクションを変更した場合、そのトランザクションは変更したユーザーのアプリケーションとなり、ユーザーに保守責任があります。
Sun Microsystems は、変更されたトランザクションをサポートしません。

---

# 第3章

# VSAMデータセットの管理

アプリケーションを実行するために領域を設定する重要な手順の 1 つが VSAM データセットの定義です。VSAM ファイルとその属性は、ファイル管理テーブル (FCT) と VSAM カタログで定義します。VSAM データセットの属性は、次のとおりです。

- データ編成。たとえば、キーシーケンスデータセット (KSDS)
- レコードの形式。固定長または可変長
- 最大レコードサイズ
- KSDS データセットの代替索引
- 自動回復機能

これらの属性の一部、たとえば、データ編成やレコード形式、レコードサイズ、代替索引などの属性は、構造上の属性でアプリケーションによって決定します。ほかの属性、たとえば、自動回復機能や、スパン、およびパフォーマンスを最適化するための特定のファイルシステムへのデータセット配置などは、オプションであり、ホストシステムでの運用上の必要性によって決定します。これらの設定の決定にあたっては、慎重に検討して計画する必要があります。

この章では、VSAM カタログの生成方法および VSAM データセットとその属性の管理方法について説明するとともに、設定を決めるために役立つ情報を提供します。次のトピックについて説明します。

# VSAM データの編成

VSAM データセットの構造上の属性には、次のものがあります。

- レコード長
- アクセス方法
  - KSDS (キーシーケンス)。KSDS ファイルの属性には、キー位置とキー長を含む
  - ESDS (入力順)
  - RRDS (相対レコード)
- レコード形式 (固定長または可変長)
- ファイル名
- 環境変数: データセットが存在するディレクトリ (`$KIXDATA` など)

Sun MTP は、1 つまたは複数のファイルの VSAM データを維持します。1 つの RRDS および ESDS データセットは、1 つの物理ファイルに対応します。各 KSDS と代替索引データセットは、2 ～ 9 の物理ファイルに対応します。これらのファイルにはすべて、データ参照に使用される特定の構造があります。

Sun MTP は、VSAM カタログを含むすべてのデータセットのタイプで、同じ VSAM ブロックサイズ (デフォルトは 4096 バイト) を使用します。レコードサイズが 1 ブロックを超えた場合、レコードは複数のブロックにスパンされます。データセットが複数ブロックにわたる場合、4072 バイトが最初のブロックに格納され、そのブロックが次のブロックをポイントします。1 ブロックのうち、20 バイトは管理に、4 バイトはレコード長および現在のブロックにあるレコードの量を示すために使用されるので、1 ブロックに格納できるデータは 4072 バイトのみになります。レコードに複数のブロックが必要な場合は、そのレコードからのデータのみが追加のブロックに格納されます。たとえば、レコードに 5000 バイトが含まれている場合は、2 つのブロックにスパンされますが、最初のブロックに 4072 バイト、2 番目のブロックに 928 バイトが格納されます。2 番目のブロックには、ほかのレコードは格納できません。複数の可変長レコードを持ち、その一部がスパンされているデータセットの例を次の図で示します。

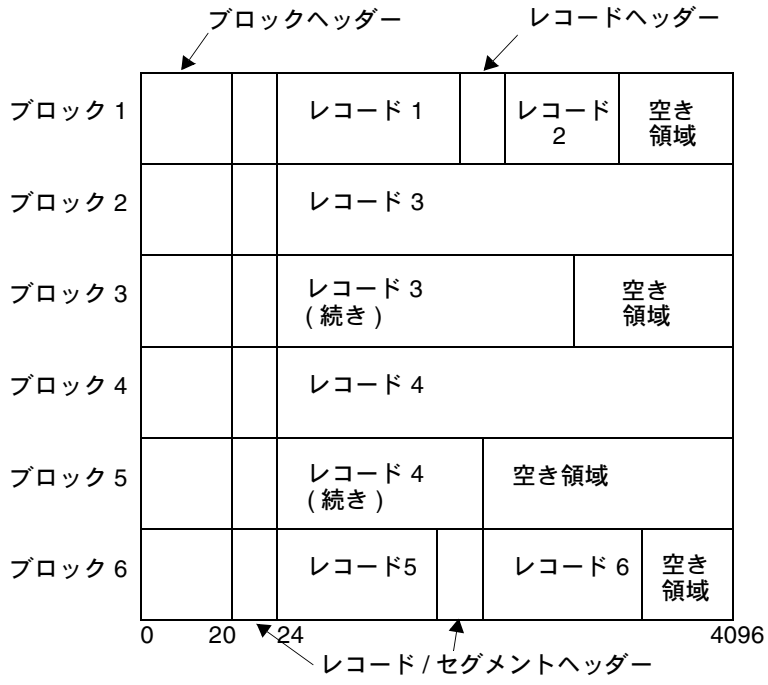


**図 3-1** VSAM Structure の例

---

**注 –** VSAM ブロック構造と、レコードの配置の詳細については、『Sun Mainframe Transaction Processing ソフトウェア 障害追跡とチューニング』を参照してください。

---

# VSAM データセットのサイズの計算

VSAM カタログを作成してそれにデータを埋め込む前に、アプリケーションの VSAM データセットのサイズを計算する必要があります。この値を使用して、パフォーマンスが最適になるデータセットとカタログのブロックサイズを選択します。レコードが 1 つのブロックを超えるサイズになると、VSAM ブロックサイズが大きいほど効率が向上します。ブロックサイズを増やす方法については、88 ページの「カタログの VSAM ブロックサイズの変更」を参照してください。

以下の節では、データセットのタイプと VSAM ブロックサイズをもとに、VSAM データセットのサイズを計算するためのさまざまな式について説明します。

# RRDS データセット

RRDS データセットは、レコードを順番に格納します。これらのファイルは固定長レコードです。データセットのブロックに、次のレコードが入らなくなるまでレコードが格納されると、ブロックの残りの領域は未使用のままになります。データセット内であるレコード番号 (RIDFLD) のレコードの位置を求めるには、1 つのブロックに入るレコード数を計算し、どのブロックがそのレコードを含むかを判定してから、そのブロック内でのレコードのオフセットを計算します。合計レコード数がわかっている場合は、次の式を使用して、RRDS データセットのサイズを計算します。

1. ブロック当たりのレコード数 = (VSAM ブロックサイズ - 20) / (レコード長 + 4) (余りは切り捨て)
2. 合計データセットサイズ = ((レコード数合計)/(ブロック当たりのレコード数),切り上げ)*(VSAM ブロックサイズ)

**例**: 長さ 80 バイトのレコードを 4,000 個持つ RRDS デーセットで、VSAM ブロックサイズが 4K バイトの場合、次のように計算します。

ブロック当たりのレコード数 = (4096 - 20 = 4076) / (80 + 4 = 84) = 48 (余りは切り捨て)

合計データセットサイズ = (4000 / 48 = 84 (切り上げ)) * 4096 = 344,064 バイト

RRDS レコードの長さが 1 ブロックを超える場合は、ファイルのサイズは次のように計算します。

1. レコード当たりのブロック数 = レコードサイズ / (ブロックサイズ -20 -4)
2. ファイルサイズ = レコード数 * レコード当たりのブロック数

# ESDS データセット

RRDS データセットと同じように、ESDS データセットはレコードを順番に格納します。ただし、ESDS データセットは、可変長レコードを格納することもできます。このことは、データセットのサイズの計算方法だけでなく、レコードの位置決めの方法にも影響します。

固定長レコードを持つ ESDS データセットについての計算方法は RRDS データセットと同じです。この場合は RRDS データセットと同様に、次のレコードが入らなくなるまでデータセット上のブロックにレコードが格納され、ブロックの残りの領域は未使用のままになるためです。したがって、データセット内であるレコード ID (RIDFLD) のレコードの位置を求めるには、1 つのブロックに入るレコード数を計算し、どのブロックがそのレコードを含むかを判定してから、そのブロック内でのレコードのオフセットを計算します。レコードの合計数がわかっている場合は、66 ページの「RRDS データセット」 に示す RRDS データセットの場合と同じ式を適用して固定長レコードを持つ ESDS データセットのサイズも計算できます。

**例**: 長さ 80 バイトの固定長レコードを 4,000 個持つ ESDS データセットで、VSAM ブロックサイズが 4K バイトの場合は、次のように計算します。

ブロック当たりのレコード数 = (4096 - 20 = 4076) / (80 + 4 = 84) = 48 (余りは切り捨て)

合計データセットサイズ = (4000 / 48 = 84 (切り上げ)) * 4096 = 344,064 バイト

ESDS データセットが可変長レコードを持つ場合は、計算方法が異なります。データセット上のブロックは、各ブロックの最後のバイトまでレコードで埋められ、必要に応じてブロックの最後のレコードは「分割」され、レコードの残りは、次のシーケンシャル VSAM ブロックの最初のレコードとして扱われます。レコード数合計と平均レコードサイズがわかっている場合は、次の式を使用して、可変長レコードを持つ ESDS データセットのサイズを計算できます。

1. ブロック当たりのレコード数 = (VSAM ブロックサイズ - 20) / (平均レコードサイズ + 4) (余りは保持)
2. 合計データセットサイズ = ((レコード数合計)/(ブロック当たりのレコード数),切り上げ)*(VSAM ブロックサイズ)

**例**: 長さが平均で 80 バイトのレコードを 4,000 個持つ ESDS デーセットで、VSAM ブロックサイズが 4K バイトの場合、次のように計算します。

ブロック当たりのレコード数 = (4096 - 20 = 4076) / (80 + 4 = 84) = 48.5238

合計データセットサイズ = (4000 / 48.5238 = 83 (切り上げ)) * 4096 = 339,968 バイト

# KSDS データセット

KSDS データセットファイルと代替索引ファイルは、RRDS や ESDS データセットよりも複雑です。それぞれ、データファイルと索引ファイルがあります。

**データファイル**:

レコードが入った 1 つまたは複数のデータブロックで構成されるファイルです。各データブロック内には、すべてのレコードが主キーシーケンスにソートされています。ただし、データブロックは、物理的にシーケンシャルではありません。各ブロックには、次のブロックの番号が入っていて、それが主シーケンスに次のブロックをポイントします。キーを使用して特定のレコードを検索するには、索引ファイルを読み取る必要があります。

**索引ファイル**:

ブロック番号と、そのブロックで最も高いキーで構成されるキーポインタを含みます。

索引ファイルは、データファイル内のデータブロックを指すキーポインタを含む、単一の索引ブロックで始まります。キーポインタでいっぱいになると、索引ブロックは分割されます。キーポインタの半分は新しい索引ブロックへ、残りは 2 つ目の新しいブロックに移動されます。元のブロックは、この 2 つの新しい索引ブロックを指す 2 つのキーポインタを持ちます。これにより、2 レベルの索引が作成されます。

追加の索引ブロックが必要な場合は、2 つ目の索引レベルに追加され、この索引レベルがいっぱいになるまで、データファイルを指します。2 つ目の索引レベルがいっぱいになると、3 つ目の索引レベルが作成されます。必要な数の索引レベルが作成されるまで、これが続きます。

KSDS データセット上の各ファイルは、頭に 4 バイトのレコードヘッダーが付き、そのあとにレコードデータが続きます。レコードが (VSAM ブロックサイズ − 24 バイト) より大きい場合は、そのレコードは、スパンレコードとみなされます。スパンレコードは、常にブロックのデータ部分の最初から始まります。そのスパンレコードの各部分は、レコード全体の格納に必要な数のブロックのデータ部分全部を使用します。そのレコードの最後の部分を含むブロックの残りの領域は未使用のままとなるので、スパンレコードは、それらのデータブロックがあるファイルのほかのレコードとは分離して格納されます。言い換えれば、非スパンレコードでは、ファイルのブロックは、次のレコードが入らなくなるまで格納され、ブロックの残りの領域は未使用のまま残されます。したがって、スパンレコードを使用するのか、あるいは大きな VSAM ブロックサイズを指定するのかの判断が重要です。索引ファイルレコードは、キーフィールドとブロック数 (4 バイト) を合わせた長さに等しい固定長レコードで、非スパンレコードとして処理されます。

スパンレコード (使用されている場合) と非スパンレコードの合計数、およびレコードサイズ (可変長の場合は、平均サイズ) がわかっていれば、次の式を使用して KSDS データセットのサイズを計算できます。

1. ブロック当たりの非スパンレコード数 = (VSAM ブロックサイズ - 20) / (レコードサイズ + 4) (余りは切り捨て)
2. 非スパンブロック数合計 = (非スパンレコード数合計) / (ブロック当たりの非スパンレコード数)
3. レコード当たりのスパンブロック数 = (レコードサイズ) / (VSAM ブロックサイズ - 24)
4. スパンブロック数合計 = (スパンレコード数合計) * (レコード当たりのスパンブロック数)
5. 合計データセットサイズ = ((非スパンブロック数合計) + (スパンブロック数合計)) * (VSAM ブロックサイズ)

**例**: 長さが平均で 80 バイトのレコードを 40,000 個持つ KSDS データセットで、VSAM ブロックサイズが 4K バイトの場合、次のように計算します。

非スパンブロック数合計 = (4096 - 20 = 4076) / (80 + 4 = 84) = 48 (余りは切り捨て)

非スパンブロック数合計 = 40,000 / 48 = 834 (切り上げ)

合計データセットサイズ = 834 * 4096 = 3,416,064 バイト

---

**注 –** このサンプルのデータセットには、スパンレコードはありません。

---

KSDS 索引ファイルもまた、KSDS データファイルのレコード数合計とキーサイズがわかっていれば、次の式を使用して計算できます。

1. ブロック当たりのレコード数 = (VSAM ブロックサイズ - 20) / (キーサイズ + 4) (余りは切り捨て)
2. 索引階層レベルの概数 = log (ブロック当たりのレコード数) (レコードの合計数)(余りは切り捨て)

---

**注 –** `log` 演算は、表計算ソフトの標準の関数です。

---

3. ブロック数合計 = Sum、i=1 から (索引階層レベルの概数) まで、(ブロック当たりのレコード数)(i-1) + ((レコード数合計) / (ブロック当たりのレコード数))。((レコード数合計) / (ブロック当たりのレコード数)) の結果を切り上げます。
4. 合計ファイルサイズ = (ブロック数合計) * (VSAM ブロックサイズ)

**例**: キーサイズが 20 バイトのレコードを 40,000 個持つ KSDS 索引ファイルで、VSAM ブロックサイズが 4K バイトの場合、次のように計算します。

ブロック当たりのレコード数 = (4096 - 20 = 4076) / (20 + 4 = 24) = 169 (余りは切り捨て)

索引階層レベルの概数 = log 169 (40,000) = 2 (小数点以下は切り捨て)

ブロック数合計 = (169) 0 + (169) 1 + (40,000 / 169), 切り上げ = 1 + 169 + 237 = 407

合計ファイルサイズ = 407 * 4096 = 1,667,072 バイト

# VSAM カタログの管理

Sun MTP でアプリケーションを実行するための、重要な設定手順の 1 つがカタログへの VSAM データセットの定義です。カタログとは、`CATALOG` という論理名を持つ VSAM データセットで、`CATALOG.dta` と `CATALOG.idx` というデータと索引部分の 2 つの物理ファイルで構成されます。

カタログは、領域の起動時に初期化されます。領域の最初の起動時には、カタログが作成されます。カタログの管理には、File Manager を使用するか、オペレーティングシステム環境で `kixexpcat` と `kiximpcat` の 2 つのユーティリティーとテキストエディタを使用します。また、これらの方法を組み合わせて使用することもできます。たとえば、最初に File Manager で VSAM データセットを定義してから、ユーティリティーとテキストエディタを使用してエントリの追加や削除ができます。

VSAM データセットをカタログに定義する前に、すべてのデータセットを FCT に入力する必要があります。FCT 内のエントリとカタログ内のエントリは 1 対 1 で対応します。FCT へのエントリの追加方法については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

VSAM ファイルの説明には、次の用語が使用されます。

| 用語 | 定義 |
|---|---|
| 代替索引 | ファイル属性、物理ファイルの位置、および基本クラスタ名を持つ、VSAM カタログ内のエントリ。代替索引を使用すると、主キーとは異なるキーによってレコード検索が可能です。 |
| クラスタ | ファイル属性と物理的なファイルの位置を持つ、VSAM カタログ内のエントリ |
| セグメント | スパンファイルの構成単位。1 つの KSDS データセットは、プライマリクラスタを含めて、最高 8 セグメントで構成されます。 |
| スパンファイル | VSAM データセットを最大ファイルサイズの最高 8 倍にまで拡大できる機能。 |
| VSAM データセット | Sun MTP VSAM 形式のユーザーデータを持つファイル。 |

# File Manager によるカタログへの VSAM データセットの定義

この節では、File Manager を使用して、領域の VSAM データセットをカタログに定義する方法を説明します。

## ▼ カタログで VSAM データセットを定義する

1. File Manager を開きます。
   - ローカルクライアントから、空白のトランザクション画面で CMNU と入力して「Development System」メインメニューを開いたのち、PF7 キーを押します。
   - 任意のタイプのクライアントから、空白のトランザクション画面で CFMS と入力して File Manager を直接開きます。

   File Manager を初めて開く場合、フィールドは空白です。

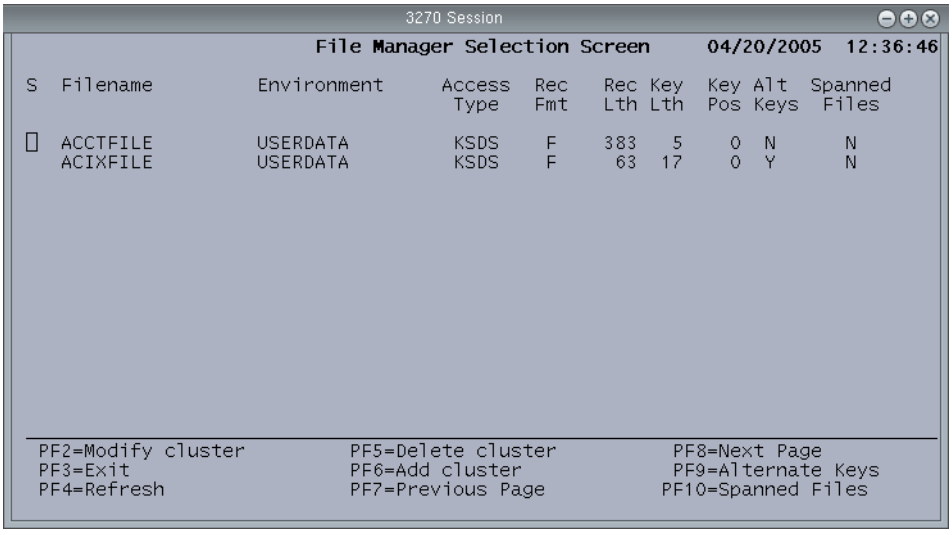


図 3-2　File Manager Selection 画面

2. 図 3-2 に示す「File Manager Selection」画面が表示されるので、PF6 キーを押します。
3. クラスタの追加画面が表示されたらデータセットを定義し、エントリが FCT のこのデータセットのエントリと正確に一致することを確認します。

   フィールド間の移動には、Tab キーを使用します。

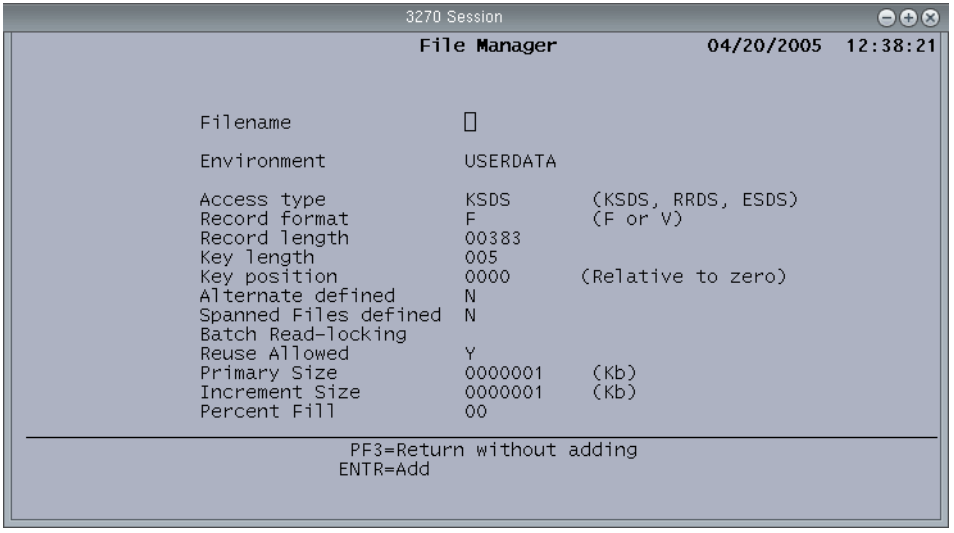


図 3-3　Add Cluster 画面

a. 「Filename」フィールドに、FCT に入力したのと同じデータセット名を入力します。

このフィールドでは、大文字と小文字が区別されます。

b. 「Environment」フィールドに、「Filename」フィールドのデータセットのディレクトリを示す環境変数を入力します。

このフィールドでは、大文字と小文字が区別されます。

c. 「Access type」フィールドで、KSDS、ESDS、または RRDS と入力して VSAM データセットのタイプを指定します。

d. 「Record format」フィールドには、データセットに固定長レコードが含まれている場合は F、可変長レコードが含まれている場合は V を入力します。

e. 「Record length」フィールドでは、データセットに固定長レコードが含まれている場合はレコード長を指定し、可変長レコードが含まれている場合は最大レコード長を指定します。

レコード長は、1 ～ 32,767 の値です。

f. 「Key length」フィールドに、KSDS データセットのレコードを検索するための主キーのキー長を入力します。

キー長は、1 ～ 255 バイトの値です。

g. 「Key position」フィールドに、レコード内で主キーフィールドが開始されるバイト位置 (0 に対する相対位置) であるキー位置を入力します。

この値は、レコード長を超えることはできません。また、キー長と相対キー位置の合計がレコード長を超えることもできません。

h. **「Alternate defined」フィールドには、その VSAM ファイルに代替キーが存在するかどうかが表示されます。**

この画面で、このフィールドは編集できません。代替キーを定義する方法については、79 ページの「既存のクラスタへの代替索引の追加」を参照してください。

i. **「Spanned files defined」フィールドには、ファイルがスパンされるかどうかが表示されます。**

この画面で、このフィールドは編集できません。スパンファイルを定義する方法については、81 ページの「File Manager を使用してスパンデータセットを定義する」を参照してください。

j. **「Batch Read-locking」フィールドでは、このデータセットのレコードが読み取り時にロックされるかどうかを指定します。次のいずれかを入力します。**

`Y`: バッチアクセス中、読み取りに対してこのレコードをロックします。

`N`: バッチアクセス中、読み取りに対してこのレコードをロックしません。

デフォルト値はありません。

VSAM RC (read consistency)、つまりバッチ読み取りロックの詳細については、83 ページの「データセットのバッチ読み取りロックの定義」と『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。

k. **「Reuse Allowed」フィールドには、`unikixbld` 操作中にこのデータセットを再利用 (スクラッチ) できるかが表示されます。有効な値は次のとおりです。**

`Y`: このデータセットを再利用できます。デフォルト値です。

`N`: このデータセットは再利用できません。

このオプションの Sun MBM 環境での使用法については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

`unikixbld` の形式と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。

l. **最後の 3 つのフィールドは、現在は使用されていません。デフォルトの値をそのまま使用してください。**

Primary Size: データセット作成時の初期サイズを K バイト単位で指定します。デフォルトは 1 です。

Increment Size: データセットに空き領域がなくなったときの 1 回の増分を K バイト単位で指定します。デフォルトは 1 です。

Percent Fill: `unikixbld` ロードまたは open output 操作の実行時に、各ブロックに残す空き領域を指定します。デフォルト値は 0 です。

4. **Enter キーを押して、クラスタ定義をカタログに追加し、「File Manager Selection」画面に戻ります。**

   追加した定義は、PF4 キーを押して画面をリフレッシュするまで画面に表示されません。

   定義を追加しない場合は、PF3 キーを押して、「File Manager Selection」画面に戻ります。

5. **PF3 キーを押して、エントリをカタログに保存し、File Manager を終了します。**

# File Manager によるデータセット属性の変更

File Manager の Modify Cluster 機能を使用すると、選択したクラスタの属性を変更できます。

クラスタ名は変更できません。クラスタ名を変更するには、まず既存のクラスタ定義を削除してから、新しい定義を新しい名前で追加する必要があります。

## ▼ クラスタを変更する

1. **「Development System」メインメニューから、または `CFMS` トランザクションを使用して、File Manager を開きます。**

2. **図 3-2 に示す「File Manager Selection」画面が表示されたら、データセット名の左にある選択フィールド (S) にカーソルを合わせてデータセットを選択します。**

3. **PF2 - Modify cluster を押します。**

4. **クラスタの変更画面に表示されるフィールドは、クラスタの追加画面のフィールド (図 3-3) と同じです。**

   Tab キーを使用して、変更するフィールドに移動します。

5. **スパンファイルのセグメントを追加、変更、または削除するには、PF1 キーを押して、スパンファイル画面を表示します。**

   詳細については、80 ページの「スパンファイル」を参照してください。

6. **クラスタに関連付けられている代替索引を追加、変更、または削除するには、PF2 キーを押して代替索引画面を表示します。**

   詳細については、76 ページの「代替索引の作成」を参照してください。

7. **Enter キーを押してクラスタを変更してから、PF3 キーを押して選択画面に戻ります。**

   属性を変更せずに選択画面に戻るには、PF3 キーを押します。

クラスタの属性を変更しても、VSAM データセットのデータは変更されません。たとえば、キー長を変更しても、VSAM データセットにある、古いキー長を使用して作成されたデータは変更されません。

### ▼ クラスタの変更時にデータを保持する

1. VSAM データセットから順編成ファイルを作成します (レコードエディタまたは `unikixbld` を使用)。
2. 属性を変更します。
3. この領域を停止します。
4. クラスタの物理ファイルを削除します。
5. 領域を再起動します。
6. 編集した順編成ファイルから新しいクラスタまでの VSAM データセットを作成します (レコードエディタまたは `unikixbld` を使用)。

## クラスタの削除

クラスタを削除すると、VSAM カタログからクラスタの定義が削除されます。このカタログエントリに関連付けられている物理ファイルは削除されません。

### ▼ クラスタを削除する

1. 「Development System」メインメニューから、または `CFMS` トランザクションを使用して、File Manager を開きます。
2. 「File Manager Selection」画面で、データセット名の左にある選択フィールド (S) にカーソルを合わせて、削除するデータセットを選択します。
3. PF5 キーを押して、クラスタの削除画面を表示します。
4. Enter キーを押して、クラスタを削除します。

   クラスタを削除せずに「File Manager Selection」画面に戻るには、PF3 キーを押します。

# 代替索引の作成

KSDS データセットに代替索引を作成するには、多くのステップが必要です。この節では、これらのステップを論理的に分けて説明します。

## ▼ 代替索引を使用してカタログエントリを作成する

1. 「Development System」メインメニューから、または `CFMS` トランザクションを使用して、File Manager を開きます。
2. 「File Manager Selection」画面が表示されたら、データセット名の左にある選択フィールドにカーソルを合わせてデータセットを選択します。
3. PF2 キーを押してクラスタの変更画面を表示します。

---

**注 –** PF9 - Alternate Keys 機能は、データセットに対してすでに定義された代替キーを表示するためにのみ使用します。

---

4. クラスタの変更画面で、PF2 キーを押して、代替索引画面を表示します。
5. 代替索引画面で、空白行に適切な情報を入力して、代替索引を定義します。
   a. 「Alternate index」列に、KSDS データセットからレコードを検索するための代替索引パス名を入力します。

      これは、FCT にあるファイル名と同じものです。

   b. 「Keylen」列に、対応するパス名が参照されたときに、KSDS データセットのレコードを検索するための代替キーの長さを指定します。

   c. 「Keypos」列に、レコード内で代替キーが始まるバイト位置 (0 に対する相対位置) を指定します。

   代替索引の属性を変更するには、変更する行にカーソルを合わせ、既存の値を上書きします。

   代替索引を削除するには、削除する行にカーソルを合わせ、スペースバーを使用して既存の情報を削除します。

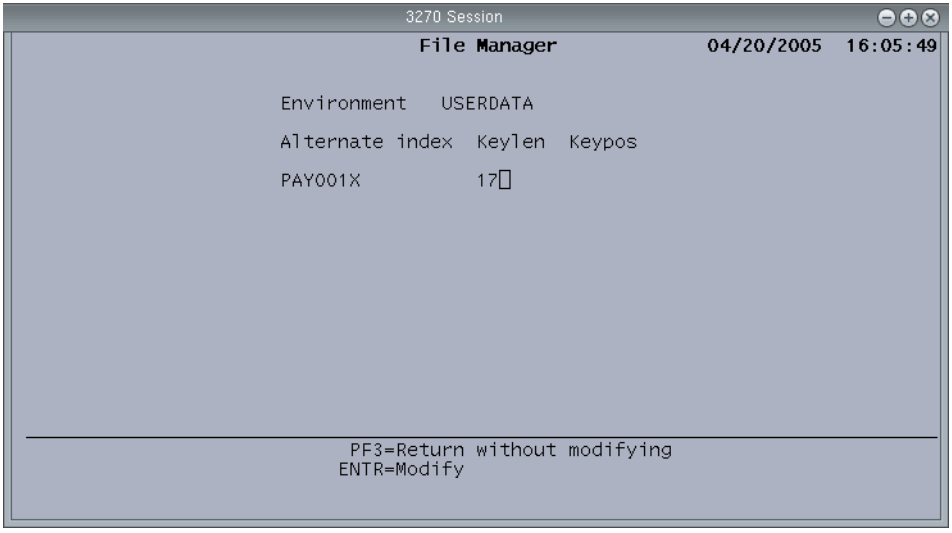


図 3-4　File Manager – Alternate Index 画面

6. Enter キーを押して、変更内容を適用し、クラスタの変更画面に戻ります。
   代替索引情報を変更せずに前の画面に戻るには、PF3 キーを押します。
7. PF3 キーを押すと、「File Manager Selection」画面に戻ります。
8. PF4 キーを押すと、画面がリフレッシュされ、変更内容が表示されます。
9. File Manager を終了します。

## ▼ FCT で代替索引を作成する

1. 「Development System」メインメニューから、または CTBL トランザクションを使用して、Table Manager を開始します。
2. 標準テーブルを選択し、ファイル管理テーブル (FCT) を選択します。
3. FCT で代替索引を定義します。
   File Manager – 代替索引画面で指定した代替索引は、代替索引のファイル名です。次の図は、FCT の代替索引とそれに関連するデータセット名を定義する必要があることを示します。

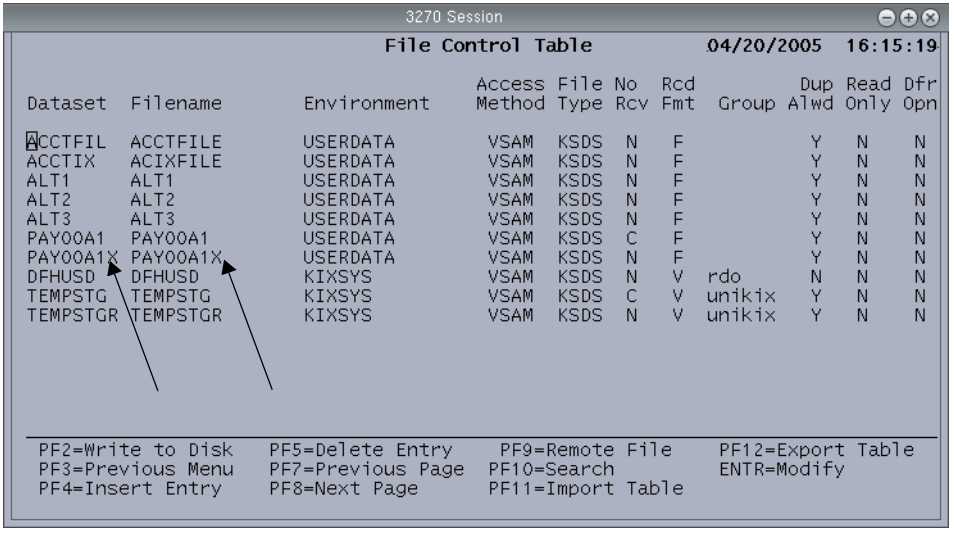


**図 3-5** File Control Table – 代替索引

図の矢印は、PAY00A1X という名前のデータセットと、代替索引 PAY00A1X を示します。次のプログラムの一部は、CICS コマンドがこのデータセット名を使用して、代替索引を参照する方法を示します。

```
EXEC CICS READ
       DATASET ('PAY00A1X')
       INTO (-----)
       RIDFLD (------)
END-EXEC
```

---

**注 –** データセット名とファイル名は、同じである必要はありません。

---

4. PF2 キーを押し、FCT を保存して Table Manager を終了します。
5. 領域を停止して再起動し、File Manager と FCT で定義したファイルと代替索引を初期化します。

### ▼ 代替索引を作成する

1. **次のいずれかの方法を使用して、レコードエディタを起動します。**
   - CRED トランザクションを入力します。
   - 「Development System」メインメニューの PF8 キーを押します。
2. **94 ページの「VSAM データセットの構築」 の手順に従い、VSAM データセットと代替索引を作成します。**

   ファイルが非常に大きい場合、または多くのクラスタを作成する場合は、unikixbld ユーティリティーを使用できます。unikixbld の使用方法の詳細については、『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』と『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。コード例 3-2 は、unikixbld スクリプトファイルの例を示します。

## 既存のクラスタへの代替索引の追加

既存のクラスタに 1 つ以上の新しい代替索引を追加できます。

### ▼ 代替索引を追加する

1. **次の手順に従って、File Manager と FCT で代替索引を定義します。**

   76 ページの「代替索引を使用してカタログエントリを作成する」

   77 ページの「FCT で代替索引を作成する」
2. **変更中の VSAM データセットから順編成ファイルを作成します。詳細は、98 ページの「順編成ファイルの構築」を参照してください。**
3. **この領域を停止します。**
4. **変更中の VSAM クラスタに関連する .dta ファイルと .idx ファイルをすべて削除します。**

   たとえば、DATASET1 に、1 つのメイン索引と 2 つの代替索引がある場合、次のコマンドを入力します。

   ```
   $ rm DATASET1.dta DATASET1.idx ALT1A.idx ALT1B.idx
   ```
5. **領域を起動し、File Manager と FCT で定義したデータセットと代替索引を初期化します。**
6. **レコードエディタを起動します。**

7. **手順 2 で作成された順編成ファイルから VSAM データセットを作成し直します。手順については、94 ページの「VSAM データセットの構築」を参照してください。**

   これにより、VSAM クラスタに新しい代替索引が追加されます。

# スパンファイル

スパンファイルとは、複数のファイルシステムにわたるデータセットです。
Sun MTP は KSDS VSAM データセットのスパンをサポートし、最高 8 セグメントまで使用できます。

---

**注 –** 1 つのセグメントに割り当て可能な容量がほかのセグメントより小さい場合、最大ファイルサイズはその小さいサイズにセグメント数を掛けた大きさになります。

---

Sun MTP は、起動時に設定されたブロックサイズを使用して、KSDS ファイルに書き込むデータを「ストライプ化」します。トランザクションからレコードへのアクセスは、1 つのデバイスのスループットによって制限されないので、ファイルアクセススループットが向上します。

## Sun MTP によるデータの分割方法

データは、ブロックごとに、スパンファイルに分割されます。各ブロックのサイズは、起動時に `unikixmain -b` オプションによって定義されます。ブロックサイズには、4、8、16、または 32K バイトが定義できます。

ブロックは物理ファイルにシーケンシャルに書き込まれ、論理ファイルのブロック 1 がセグメント 1 のブロック 1 に書き込まれます。論理ファイルのブロック 2 はセグメント 2 のブロック 1 に書き込まれます。ブロック書き込みの際、ブロック番号をセグメント数で割った余り + 1 がファイル番号となり、商 + 1 がそのファイルでのブロック番号となります。これを図 3-6 に示します。

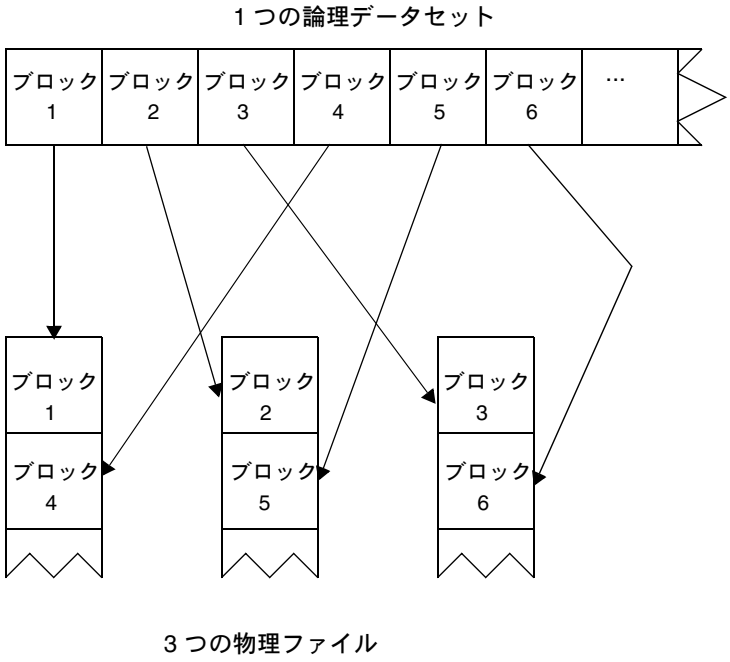


**図 3-6**　　スパンデータセット – 論理ファイルと物理ファイルの関連図

## ▼ File Manager を使用してスパンデータセットを定義する

1. **「Development System」メインメニューから、または `CFMS` トランザクションを使用して、File Manager を開きます。**

2. **「Add Cluster」機能を使用して、データセットのメインセグメントを定義します。**

   このメインセグメントは、索引ファイル (`.idx`) と最初のデータセグメント (`.dta`) の位置を定義します。これ以降のデータセグメントには、`.dt1` ～ `.dt7` という拡張子が付きます。詳細は、71 ページの「File Manager によるカタログへの VSAM データセットの定義」を参照してください。

3. **「File Manager Selection」画面で、スパンするデータセットを選択して PF2 キーを押します。**

---

**注 –** 追加または変更を行わずに「Spanned File Definition」画面を表示するには、「File Manager Selection」画面で PF10 キーを押します。

---

4. **クラスタの変更画面で、PF1 キーを押して、「Spanned File Definition」画面を表示します。**

   次の情報が表示されます。

   - 最初のデータセグメントの環境変数。データセットを探すために使用されます。
   - 「Segment」列は、表示のみのフィールドで、セグメント番号を示します。
   - セグメントの場所を指定する環境変数。このディレクトリは、通常、ほかのデータセグメントとは異なるファイルシステムにあります。セグメントの環境変数は、データセットを開く前に定義する必要があります。

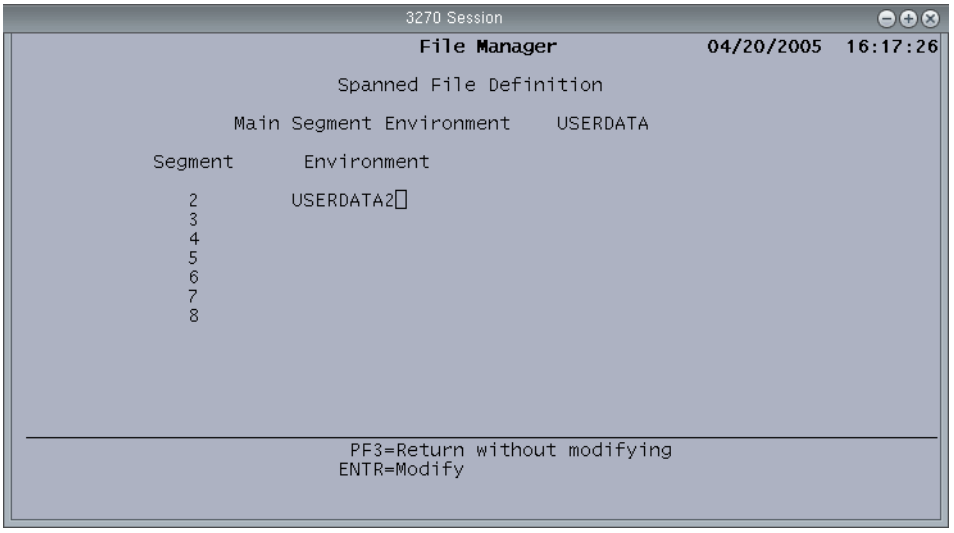


**図 3-7** File Manager – Spanned File Definition 画面

5. **セグメントを追加するには、そのセグメントの環境変数を入力します。セグメントを削除するには、スペースバーでその環境定義を削除します。**

6. **セグメントの追加または変更が終了したら Enter キーを押します。**

   スパンファイルの情報を変更せずにクラスタの変更画面に戻る場合は、PF3 キーを押します。

7. **Enter キーをもう 1 度押すと、「File Manager Selection」画面に戻ります。**

---

**注 –** セグメント定義を変更した場合、カタログのみが変更され、実際のデータは変更されません。

---

# データセットのバッチ読み取りロックの定義

バッチ読み取りロック、つまり VSAM RC (read consistency) は、バッチプログラムとオンライントランザクションが同じ VSAM データセットを同時に更新する場合に、データの整合性を完全に保ちます。読み取り整合性を保証する方法には、次の 3 つがあります。最初の 2 つの方法は、アプリケーションコードを変更する必要がありません。3 番目の方法は、アプリケーションコードを変更する必要があるので、領域またはデータセットのオプションでは対応できない場合のみ使用します。

- 領域内のすべての VSAM データセットに対して VSAM RC を指定します。カタログがリリース 7.0 より前のバージョンのソフトウェアで作成したものである場合は、VSAM カタログの変換が必要です。`kixcnvtcat81` ユーティリティーの詳細については、『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 –** 従来のバージョンのソフトウェアから移行する場合は、この変換が必要です。バッチ読み取りロックを `Y` と指定すると、すべてのデータセットにバッチ読み取りロック属性が設定されます。またこの値を `N` に設定すると、ロックされません。リリース 7.0 より前のリリースでは、ロックなし (ダーティー読み取り) がデフォルトの動作でした。

---

- VSAM カタログの特定のデータセットに対して、読み取りロック動作を指定します (`Y` または `N`)。この方法は、デフォルトのカタログ動作を無効にする場合に使用します。新しいクラスタを定義する場合は 71 ページの「File Manager によるカタログへの VSAM データセットの定義」を、または 74 ページの「File Manager によるデータセット属性の変更」を参照してください。`CATALOG.lst` ファイル内でこのフィールドを編集することも可能です。表 3-1 は、`CATALOG.lst` ファイルのレコード形式を示します。カタログをインポート、エクスポートする手順については、84 ページの「ユーティリティーを使用した VSAM カタログの管理」を参照してください。`kixexpcat` と `kiximpcat` 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』も参照してください。
- 特定のバッチジョブまたはステップで、特定のデータセットに対して読み取りロック属性を指定します。『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』のバッチ処理に関する章を参照してください。

# ユーティリティーを使用した VSAM カタログの管理

Sun MTP は、File Manager を使用せずに VSAM カタログを管理するために、次の 2 つのユーティリティーを用意しています。

| | |
|---|---|
| `kixexpcat` | VSAM カタログのエクスポート、つまり `$KIXSYS/CATALOG.dta` および `$KIXSYS/CATALOG.idx` を 1 つの ASCII テキストファイルにエクスポートします。このテキストファイルは定型で、任意のテキストエディタで編集できます。`kixexpcat` ユーティリティーは表 3-1 に示す最大文字数まで文字フィールドを空白で埋めます。 |
| `kiximpcat` | ASCII カタログファイルを VSAM カタログ形式にインポートします。 |

これらのコマンドとオプションの詳細は、『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。

## VSAM カタログの ASCII 形式

コード例 3-1 に示すファイルと同様の ASCII ファイルから VSAM カタログを作成できます。このファイルは、表 3-1 で説明されている形式に従う必要があります。

カタログファイルを作成または編集する場合は、次のガイドラインに従います。

- ASCII ファイル内のレコードは、2 番目と 3 番目のフィールド、つまり、プライマリクラスタの環境変数と主ファイル名から成るキーに従ってソートされます。この順序は維持する必要があります。
  - 同じプライマリクラスタに属するすべてのレコードが 1 つにグループ化され、プライマリクラスタを定義しているレコードがグループの最初に配置されます。
  - 主クラスタの定義は、それに関連するスパン定義や代替索引定義よりも前に位置する必要があります。
- 1 つの VSAM KSDS データセットには、スパンセグメントが最高 7 つまで定義できるので、主セグメントを含めると、合計 8 つのセグメントが定義できます。
- 1 つの VSAM KSDS データセットは、最高 12 の代替ファイル定義 (索引) を持つことが可能です。
- RRDS および ESDS データセットは、代替索引を持つことも、スパンすることもできません。
- RRDS および ESDS データセットは、キー長とキーオフセットフィールドを持つことができません。

詳細は、87 ページの「ユーティリティーを使用して VSAM カタログを管理する」を参照してください。

**表 3-1** CATALOG.lst のレコードタイプの説明

| レコードタイプ | フィールド名 | フィールドの内容 |
|---|---|---|
| B<br>ブロックサイズ<br>CATALOG.lst の 1 行目に記述 | Record type | B という 1 文字 |
| | Block size | 最大 3 文字。有効な値は次のとおりです。<br>4K、8K、16K、32K |
| P<br>プライマリクラスタ | Record type | P という 1 文字 |
| | Primary environment | 14 文字 |
| | Primary file name | 14 文字 |
| | Record format | 次の 1 文字<br>F: Fixed (固定)<br>V: Variable (可変) |
| | Access | 次の 4 文字<br>KSDS<br>ESDS<br>RRDS ESDS |
| | Record length | 5 桁の 10 進数 |
| | Key length | 5 桁の 10 進数 |
| | Key offset | 5 桁の 10 進数 |
| | Batch read-locking | 次の 1 文字<br>Y: バッチからこのファイルに READ が実行されると、VSAM レコードをロックします。<br>N: バッチからこのファイルに READ が実行されても、VSAM レコードをロックしません。 |
| | Reuse allowed | 次の 1 文字<br>Y: このファイルは再利用できます。<br>N: このファイルは再利用できません。 |
| | Primary size | デフォルト値は 1 です。 |
| | Increment size | デフォルト値は 1 です。 |
| | Percent fill | デフォルト値は 0 です。 |

表 3-1　CATALOG.lst のレコードタイプの説明 (続き)

| レコードタイプ | フィールド名 | フィールドの内容 |
|---|---|---|
| A<br>代替索引定義 | Record type | A という 1 文字 |
| | Primary environment | 14 文字 |
| | Primary file name | 14 文字 |
| | Alternate environment | 14 文字 |
| | Alternate file name | 14 文字 |
| | Alternate key length | 5 桁の 10 進数、クラスタのキー長 |
| | Alternate key offset | 5 桁の 10 進数、クラスタのキーのオフセット |
| S<br>スパンセグメント定義 | Record type | S という 1 文字 |
| | Primary environment | 14 文字 |
| | Primary file name | 14 文字 |
| | Spanned environment | 14 文字 |

## ▼ ASCII カタログファイルを作成する

1. **テキストエディタを使用して、テキストファイルを作成し、CATALOG.lst と名前を付けます。**
2. **レコードを入力します。フィールドの区切りには、コンマを使用します。**

   各レコードは、表 3-1 で説明するレコードタイプのいずれかで始まり、そのレコードタイプに応じた情報が格納されます。

3. ASCII ファイルは、次の形式で作成します。

コード例 3-1　CATALOG.lst ファイル

```
B,4K
P,KIXDATA0   ,EFILE001   ,F,ESDS,00080,00000,00000,Y,Y,0000001,0000001,00
P,KIXDATA0   ,EFILE002   ,F,ESDS,00080,00000,00000,Y,Y,0000001,0000001,00
P,KIXDATA0   ,KFILE001   ,F,KSDS,00080,00008,00000,Y,Y,0000001,0000001,00
A,KIXDATA0   ,KFILE001   ,KIXDATA0 ,KALT1001 ,00008,00009
A,KIXDATA0   ,KFILE001   ,KIXDATA0 ,KALT2001 ,00008,00011
P,KIXDATA0   ,KFILE002   ,F,KSDS,00080,00008,00000,Y,Y,0000001,0000001,00
A,KIXDATA0   ,KFILE002   ,KIXDATA0 ,KALT1002 ,00008,00002
A,KIXDATA0   ,KFILE002   ,KIXDATA0 ,KALT2002 ,00008,00003
A,KIXDATA0   ,KFILE002   ,KIXDATA0 ,KALT3002 ,00008,00004
A,KIXDATA0   ,KFILE002   ,KIXDATA0 ,KALT4002 ,00008,00005
S,KIXDATA0   ,KFILE002   ,KIXDATA1
S,KIXDATA0   ,KFILE002   ,KIXDATA2
S,KIXDATA0   ,KFILE002   ,KIXDATA3
S,KIXDATA0   ,KFILE002   ,KIXDATA4
S,KIXDATA0   ,KFILE002   ,KIXDATA5
```

4. ファイルを保存します。

5. kiximpcat を実行して、ASCII ファイルをインポートし、カタログを作成します。次に例を示します。

```
$ kiximpcat -l /tmp/CATALOG.lst -c $KIXSYS/CATALOG
```

この例では、/tmp ディレクトリにある CATALOG.lst ファイルをインポートし、CATALOG.dta と CATALOG.idx ファイルを作成して、$KIXSYS ディレクトリに保存します。

## ▼ ユーティリティーを使用して VSAM カタログを管理する

1. 実行中の領域は終了します。

2. VSAM カタログを ASCII カタログファイルにエクスポートします。次に例を示します。

```
$ kixexpcat -c $KIXSYS/CATALOG -l CATALOG.lst
```

3. テキストエディタを使用して、ファイルの追加または変更を行います。

4. ASCII ファイルを VSAM カタログ形式にインポートします。次に例を示します。

```
$ kiximpcat -l CATALOG.lst -c $KIXSYS/CATALOG
```

5. この領域を起動します。

6. FCT を開き、カタログに追加した各新規ファイルのエントリを作成し、変更したファイルのエントリを編集します。

   FCT の編集は、Table Manager を使用してファイル定義を入力するか、エクスポートした FCT (`fct.lst`) を編集して、それをインポートすることで行います。

7. 領域を停止して再起動して、新しいカタログを使用します。

   カタログの作成または変更時、FCT とカタログが一致しているかどうかは検証されません。したがって、カタログファイル内の各データセットの定義が、FCT 内の同じデータセットの定義と正確に一致することを確認する必要があります。データが一致しない場合、起動時にエラーが表示されます。

   同じ物理ファイルに対し、複数のデータセット名を FCT で定義できます。それらのデータセットのエントリにある FCT の構成情報は、互いに一致するだけでなく、カタログの構成情報に一致する必要があります。Read Only、Deferred Open、Batch Only などの、データセットへのアクセスに影響する属性は異なっていてもかまいません。

   FCT の詳細については、『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。

## カタログの VSAM ブロックサイズの変更

Sun MTP は、VSAM データセットのブロックサイズを領域単位で設定します。サポートするブロックサイズは、4、8、16、または 32K バイトのいずれかになり、デフォルトは 4K バイトです。KSDS データセットに 4K バイトを超えるレコード長があれば、ブロックサイズを大きくすることで、パフォーマンスの向上を図ることができます。これは、順次バッチ操作の場合もより効率的です。

領域が最初に起動されたときに、Sun MTP は領域全体のブロックサイズを `unikixmain` の `-b` オプションから取得します。カタログの作成後は、ブロックサイズがカタログに指定されたブロックサイズに合わせて自動的に設定されます。

システム全体で適用するブロックサイズを変更するには、カタログの VSAM ブロックサイズを変更する必要があります。

## ▼ カタログのブロックサイズを変更する

1. **カタログのすべての VSAM データセットを順編成ファイルにアンロードします。**

   unikixbld または 98 ページの「順編成ファイルの構築」で説明しているプロセスが使用できます。カタログ内のデータセットが多数ある場合は、unikixbld を使用する必要があります。

2. **この領域を停止します。**

3. **$KIXSYS ディレクトリで、コマンドプロンプトから、kixexpcat コマンドを実行して、カタログを ASCII ファイル CATALOG.lst にエクスポートします。**

   kixexpcat の詳細については、84 ページの「VSAM カタログの ASCII 形式」と『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。

4. **次のようにファイルを削除します。**

   - 既存のカタログファイル (CATALOG.dta と CATALOG.idx) を削除するか、名前を変更します。
   - rm TEMPSTG* コマンドを実行して、一時記憶域データセットを削除します (回復可能一時記憶域データセットを含む)。
   - 一時データキューを削除します。
   - VCT で定義された領域の回復ファイルを削除します。

5. **ASCII カタログファイルを編集して、ブロックサイズを変更します。**

   以下の例は、最初の行で、4K バイトであるものを次のように変更します。

```
B,4K
P,USERDATA   ,EFILE001 ,F,ESDS,00080,0000,0000,Y,Y,0000001,0000001,00
P,USERDATA   ,EFILE002 ,F,ESDS,00080,0000,0000,Y,Y,0000001,0000001,00
...
```

   8K バイトに変更した例です。

```
B,8K
P,USERDATA    ,EFILE001 ,F,ESDS,00080,0000,0000,Y,Y,0000001,0000001,00
P,USERDATA   ,EFILE002 ,F,ESDS,00080,0000,0000,Y,Y,0000001,0000001,00
...
```

6. **CATALOG.lst ファイルを保存します。**

7. **$KIXSYS ディレクトリで、コマンドプロンプトから kiximpcat を実行してカタログファイルを CATALOG という名前で作成し直します。**

   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。

8. **領域を再起動します。**

9. **対応する順編成ファイルから各 VSAM データセットを作成し直します。**

   データセット数が少ない場合は、94 ページの「VSAM データセットの構築」で説明するプロセスを使用します。そうでない場合は、unikixbld ユーティリティーを使用して、VSAM データセットを作成し直します。次の例では、VSAM データセットを作成し直す unikixbld スクリプトを示します。

**コード例 3-2** unikixbld スクリプトファイル

```
$!/bin/ksh
###############################################
#this script loads all the vsam datasets
###############################################
echo `date`
SLU=/net/sys1/porting/batch_ascii;export SLU
ARM=/net/sys1/porting/data;export ARM
USERDATA=/net/sys2/mig11/acct/data;export USERDATA
echo load of ACM010D starting
unikixbld -d ACM010D -r record -s $SLU/ACM010D.asc -tv
echo load of ARM010D starting
unikixbld -d ARM010D -r recordv -s $ARM/ARM010D.rdw4 -tv
echo load of ARX010D starting
unikixbld -d ARX010D -r record -s $SLU/ARX010D.srt -tv
echo load of BAM010D starting
unikixbld -d BAM010D -r record -s $SLU/BAM010D.srt -tv
echo load of BEM010D starting
unikixbld -d BEM010D -r record -s $SLU/BEM010D.srt -tv
echo load of BON010D starting
unikixbld -d BON010D -r record -s $SLU/BON010D.asc -tv
echo load of BRM010D starting
unikixbld -d BRM010D -r record -s $SLU/BRM010D.srt -tv
echo load of BSM010D starting
unikixbld -d BSM010D -r record -s $SLU/BSM010D.srt -tv
echo load of COM010D starting
unikixbld -d COM010D -r record -s $SLU/COM010D.asc -tv
echo load of DBM010D starting
unikixbld -d DBM010D -r record -s $SLU/DBM010D.asc -tv
echo load of DPM010D starting
unikixbld -d DPM010D -r record -s $SLU/DPM010D.srt -tv
echo load of GLM010D starting
unikixbld -d GLM010D -r record -s $SLU/GLM010D.asc -tv
echo load is done
echo `date`
```

unikixbld の詳細については、『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』と『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。

# VSAM データセットの操作

Sun MTP には、VSAM データセットを容易に操作できる次のような機能とユーティリティーが用意されています。

- レコードエディタの機能 : Build VSAM File、Build Sequential File、および Dump File
- kixfile ユーティリティー
- unikixbld ユーティリティー

## レコードエディタの機能へのアクセス

「Data File Editor Menu」は、開発システムのレコードエディタに含まれています。VSAM カタログに定義したデータセットを一覧します。

### ▼ データファイルエディタを開く

1. **この領域を起動します。**
2. **図 3-8 のようなデータファイルエディタを表示します。**
   - ローカルクライアントから、空白のトランザクション画面で CMNU と入力して「Development System」メインメニューを開いたのち、PF8 キーを押します。
   - クライアントから、空白のトランザクション画面で CRED を入力します。

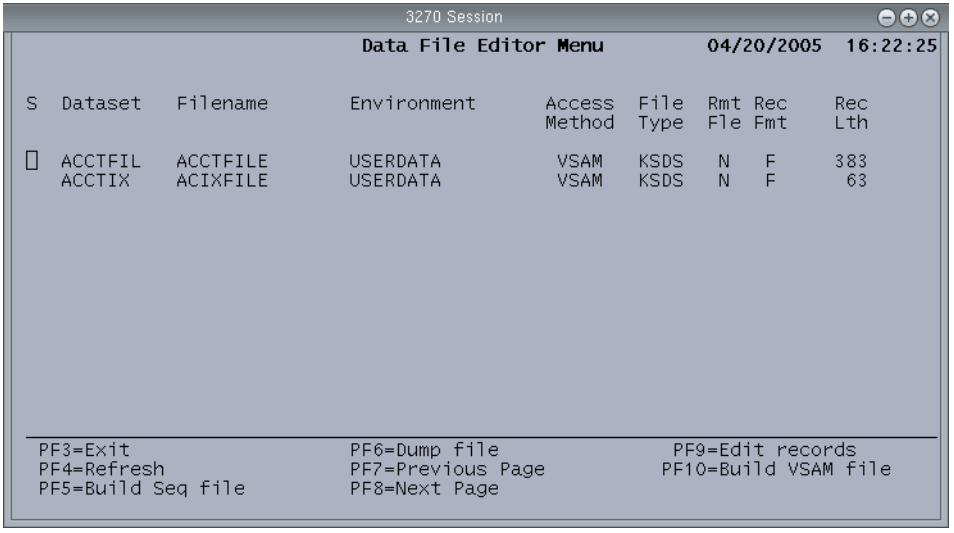


**図 3-8** Data File Editor Menu 画面

「Data File Editor Menu」には、次の情報が含まれています。

| 列 | 説明 |
|---|---|
| S | 選択フィールド。データセット名の左にある選択フィールドにカーソルを移動して、データセットを選択します。 |
| データセット | プログラムがファイルにアクセスするための論理名。物理ファイルの名前や場所を変更しても、アプリケーションを変更する必要はありません。アプリケーションは、このデータセット名の値を引き続き使用できます。<br>複数のデータセットが同じ物理ファイルに関連付けられている場合、各データセットがこのメニュー上に表示され、そのファイルへのアクセスが可能です。 |

| 列 | 説明 |
| --- | --- |
| Filename | 物理ファイル識別子<br>**KSDS データセットの場合**:<br>「Environment」フィールドで指定されたパス名とともに、VSAM カタログのファイルを識別する名前。このファイルはデータ部分と索引部分の 2 つの物理ファイルで構成され、これらのファイルにはそれぞれ、.dta と .idx という拡張子が付きます。<br>**ESDS および RRDS データセットの場合**:<br>物理ファイル名。<br>**代替索引ファイルの場合**:<br>「Environment」フィールドで指定されたパス名とともに、VSAM カタログのファイルを識別する名前。これは、カタログで定義された代替パス名に対応します。代替索引ファイルはデータ部分と索引部分の 2 つの物理ファイルで構成され、これらのファイルにはそれぞれ、.dta と .idx という拡張子が付きます。 |
| Environment | 「Filename」を含むディレクトリの環境変数。 |
| Access Method | Virtual Storage Access Method を表す VSAM という値が表示されます。それ以外のアクセス方法は、現在サポートされていません。 |
| File Type | VSAM データセットのファイル編成<br>KSDS: キー順のデータセット<br>ESDS: エントリ順のデータセット<br>RRDS: 相対レコードデータセット |
| Rmt Fle | データセットが遠隔ファイルであることを示します。つまり、データの読み書きが、FCT および TCT で定義された遠隔システムとの間で行われます。<br>Y: データセットは遠隔ファイルです。<br>N: データセットは遠隔ファイルではありません。 |
| Rec Fmt | レコード形式。<br>V: 可変長レコード形式<br>F: 固定長レコード形式 |
| Rec Lth | 固定レコードのデータセットではそのレコード長、可変長レコードのデータセットでは最大レコード長を表示します。値は 1 以上 32767 以下です。 |

「Data File Editor Menu」では、次のキーが使用できます。

| ファンクションキー | アクション |
|---|---|
| PF3 | 「Development System」メインメニューまたは空白のトランザクション画面に戻ります。 |
| PF4 | ファイル表示領域の内容を更新します。 |
| PF5 | 選択したデータセットから順編成データファイルを作成します。詳細は、98 ページの「順編成ファイルの構築」を参照してください。 |
| PF6 | 選択したデータセットのレコードの一部または全部を、ディスクファイルまたはシステムプリンタにダンプします。ファイルは、形式化されたレポートの形でダンプされます。詳細は、100 ページの「VSAM データセットのダンプ」を参照してください。 |
| PF7 | データセットの前のページを表示します。 |
| PF8 | データセットの次のページを表示します。 |
| PF9 | レコードエディタを開きます。これにより、既存のデータセットにあるレコードの追加、変更、および削除が可能になります。詳細は、第 5 章を参照してください。 |
| PF10 | 順編成データファイルから VSAM データセットを作成します。詳細は、94 ページの「VSAM データセットの構築」を参照してください。 |

# VSAM データセットの構築

「Build VSAM File」機能を使用すると、順編成データファイルから VSAM データセットが構築されます。構築するデータセットは、VSAM カタログに定義されている必要があります。定義されていないと、「Data File Editor Menu」画面に表示されません。新しいデータセットを構築する場合、データセットは構築を実行するまで空です。データセットがすでに存在する場合、上書きするか、既存のファイルのあとに追加するかを尋ねるメッセージが表示されます。

## ▼ VSAM データセットを構築する

1. **「Data File Editor Menu」画面で、データセットを選択して PF10 キーを押し、次の画面を表示します。**

   初めてこの画面を開くと、「Input Sequential Files」と「Fill %」フィールドが空白で、「Output Dataset」フィールドには、「Data File Editor Menu」画面で指定した値が入っています。「Output Dataset」フィールドは変更できません。

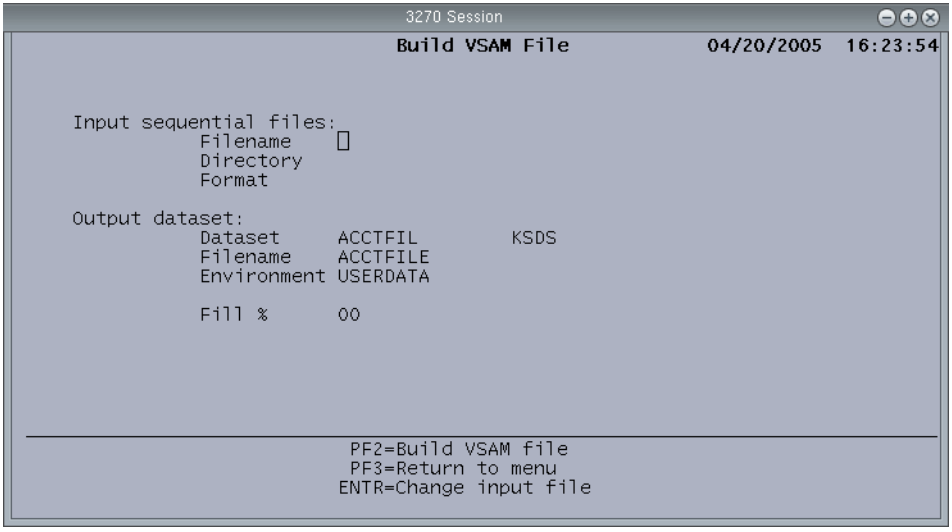


**図 3-9** Data File Editor—Build VSAM File 画面

2. **「Input Sequential Files」領域の「Filename」フィールドに、VSAM データセットの構築に使用する順編成ファイルの名前を入力します。**

3. **「Directory」フィールドに、順編成ファイルが配置されているディレクトリの名前を入力します。**

   ディレクトリを表す環境変数も入力できます。

4. **「Format」フィールドに、構築プロセスで使用するレコード処理ルーチンの名前を入力します。**

   次のいずれかを指定する必要があります。

   `line`、`mfrcd`、`mfrcdv`、`record`、`recordv`

   「Format」フィールドに表示される名前の末尾に `.in` という拡張子が付けられ、処理ルーチンの完全な名前が決定されます。これらのルーチンは、`$UNIKIX/bin/build` にあります。詳細は、96 ページの「レコード処理ルーチン」を参照してください。VSAM データセットの形式は、VSAM カタログの構成情報によって決定します。

5. 「Fill %」フィールドに、各 VSAM ブロックに割り当てる空き容量の割合を入力します。

   たとえば、0% は空き容量を割り当てないことを示し、60% は各ブロックの 60% を空いたままにしておくことを示します。ブロックに余分の領域を残しておくことにより、新しい情報を挿入するために分割を必要とするブロックの数を減らすことができます。空き領域が残っていない場合、ファイルの再構築後、挿入を行うたびにブロックの分割が発生します。入力した情報を変更する場合、Enter キーを押してカーソルを「Filename」フィールドに移動します。フィールド間の移動には、Tab キーを使用します。

6. PF2 キーを押して VSAM データセットを構築するか、PF3 キーを押してデータセットを構築せずに「Data File Editor Menu」画面に戻ります。

   構築しているデータセットに現在データが存在する場合は、データの削除 (KSDS および RRDS データセットの場合)、または新しいデータの追加 (ESDS データセットの場合) を確認するプロンプトが表示されます。

   KSDS または RRDS データセットでは、プロンプトで Enter キーを押すと、現在定義されているレコードが削除され、入力ファイルからデータセットにレコードが追加されます。ESDS データセットでは、Enter キーを押すと、入力ファイルからデータセットにレコードが追加されます。

7. PF3 キーを押して、「Data File Editor Menu」画面に戻ります。

8. PF3 キーを押して、レコードエディタを終了します。

# レコード処理ルーチン

VSAM データベースのロードとアンロードに使用される順編成ファイルの入出力は、レコード処理ルーチンによって処理されます。各レコード処理ルーチンは、1 つの特定のレコード形式を処理します。1 つのレコード形式には、入力処理と出力処理のための 2 つの実行可能ファイルがあります。たとえば、`recordv` 形式には、次の 2 つの実行可能ファイルがあります。

- `recordv.in` は、順編成ファイルの入力を処理する実行可能ファイルの名前です。
- `recordv.out` は、順編成ファイルの出力を処理する実行可能ファイルの名前です。

レコード処理ルーチンは、ロードプロセスを制御するループによって構成されます。たとえば、順編成ファイル入力ルーチンは、入力順編成ファイルを読み取って、データを Sun MTP ロードプロセスに提供します。正常に制御が戻ると、プログラムはループの始めに戻ります。詳細は、270 ページの「レコード処理ルーチンのカスタマイズ」を参照してください。

Sun MTP には、line、record、recordv、mfrcd、および mfrcdv の 5 つのレコード処理ルーチンがデフォルトで用意されています。4 つの処理ルーチンは、$UNIKIX/src/record ディレクトリにあります。最初の 3 つは、例として示したものです。line、record、および recordv は、実際の形式では、高速化のために組み込まれています。ただし、これらのルーチンは、わずかに異なるルーチンが必要な程度であれば、変更して別の名前で保存できます。mfrcdv ルーチンは組み込み型です。これは、「record」ディレクトリに物理的に存在しないので、再構築もできません。

次の表では、処理ルーチンとレコード形式を示します。

**表 3-2** レコード処理ルーチンとレコード形式

| ルーチン | 説明 |
|---|---|
| line | あらゆるユーティリティーで操作できる Line 形式化ファイル。line ファイルは、改行文字 (LF) で終わる行で形成されます。これらのファイルは、文字データだけを含み、可変長です。<br>line ファイルが固定長の VSAM ファイルから作成される場合、データに後続する空白は削除されます。<br>line ファイルが可変長ファイルから作成される場合、レコードサイズは、後続の改行を含めた VSAM ファイルから取得されます。 |
| record | 固定長レコードの順編成ファイル。各レコードにはバイナリデータが入り、レコードの終わりを示す終端は追加されません。レコードサイズは、作成しているファイルの VSAM カタログから取得されます。 |
| recordv | 可変長レコードの順編成ファイル。record 形式と同様、各レコードにはバイナリデータが入っています。各レコードは、4 バイトのバイナリフィールドが頭に付き、これに実際のレコード長が保存されています。最大レコード長は、作成しているファイルの VSAM カタログのレコード長フィールド (Rec Lth) から取得されます。 |
| mfrcd | Server Express が認識できる形式の固定長レコードが格納されている順編成ファイル。 |
| mfrcdv | Server Express が認識できる形式の可変長レコードが格納されている順編成ファイル。 |

これらの処理ルーチンで、順編成ファイルを構築したり、既存の順編成ファイルを Sun MTP VSAM 形式へ移行できます。

## 順編成ファイルの構築

「Build Sequential File」機能を使用すると、VSAM データセットから順編成データファイルが構築されます。

## ▼ 順編成ファイルを構築する

1. 「Data File Editor Menu」メニューで、データセット名の横にある選択フィールドにカーソルを合わせてデータセットを選択します。
2. PF5 キーを押して、「Build Sequential File」画面を表示します。

   「Input Dataset」フィールドには、「Data File Editor Menu」画面から取得したファイル情報が表示されます。カーソルは最初の空白のフィールドに配置されます。

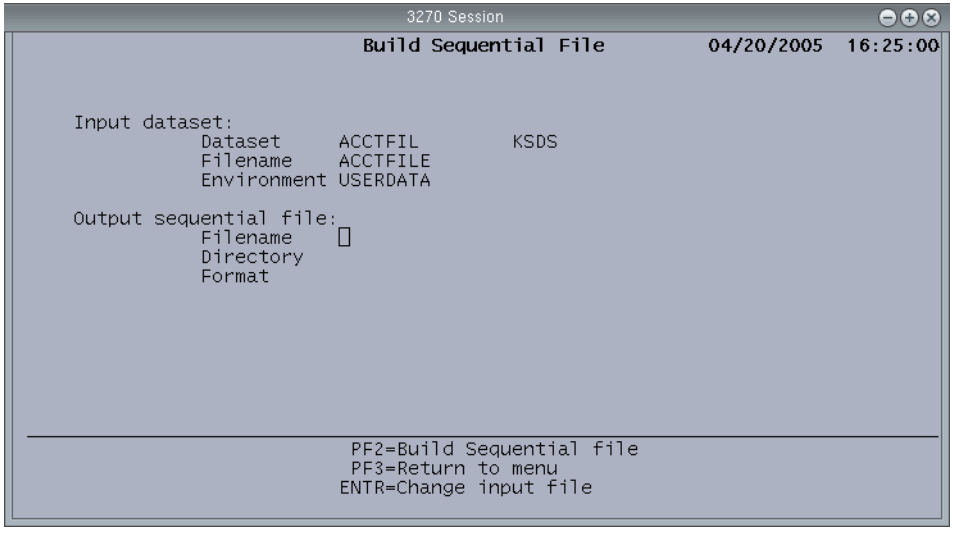


図 3-10　Data File Editor—Build Sequential File 画面 (ブランク)

3. 「Output Sequential File」領域の「Filename」フィールドに、構築する順編成ファイルの名前を入力します。
4. 「Directory」フィールドに、順編成ファイルを構築するディレクトリの名前を入力します。

   構築先ディレクトリには、環境変数も使用できます。

5. 「Format」フィールドに、構築プロセスで使用するレコード処理ルーチンの名前を入力します。

   次のいずれかを指定する必要があります。

   `line`、`mfrcd`、`mfrcdv`、`record`、`recordv`

   「Format」フィールドに表示される名前の末尾に `.out` という拡張子が付けられ、処理ルーチンの完全な名前が決定されます。これらのルーチンは、`$UNIKIX/bin/build` にあります。詳細は、96 ページの「レコード処理ルーチン」を参照してください。

   次の図は、入力後の「Build Sequential File」画面を示しています。

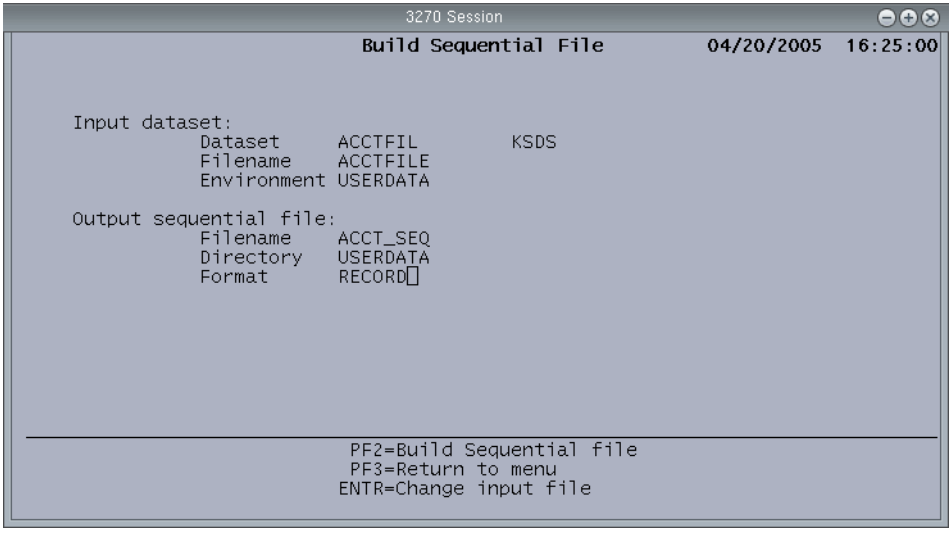


**図 3-11** Data File Editor—Build Sequential File 画面

6. 入力した情報を変更するには、Enter キーを押してカーソルを「Filename」フィールドに移動します。

   フィールド間の移動には、Tab キーを使用します。

7. PF2 キーを押してファイルを構築します。

   ファイルがすでに存在する場合、次のメッセージが表示されます。

```
KIX1426E Output file already exists. Press PF2 again to delete/rebuild
```

   メッセージに従って PF2 キーを押します。

8. PF3 キーを押して、「Data File Editor Menu」画面に戻ります。

# VSAM データセットのダンプ

Dump File 機能は、VSAM データセットのレコードの一部または全部を、書式付きレポートの形でディスクファイルまたはシステムプリンタにダンプします。ダンプファイルは、データセットのデータの検査に使用できます。

## ▼ データセットをダンプする

1. 「Data File Editor Menu」でデータセットを選択し PF6 キーを押します。

   「Dump File」画面の「Dataset to be Dumped」には、「Data File Editor Menu」から取得した情報が入っています。

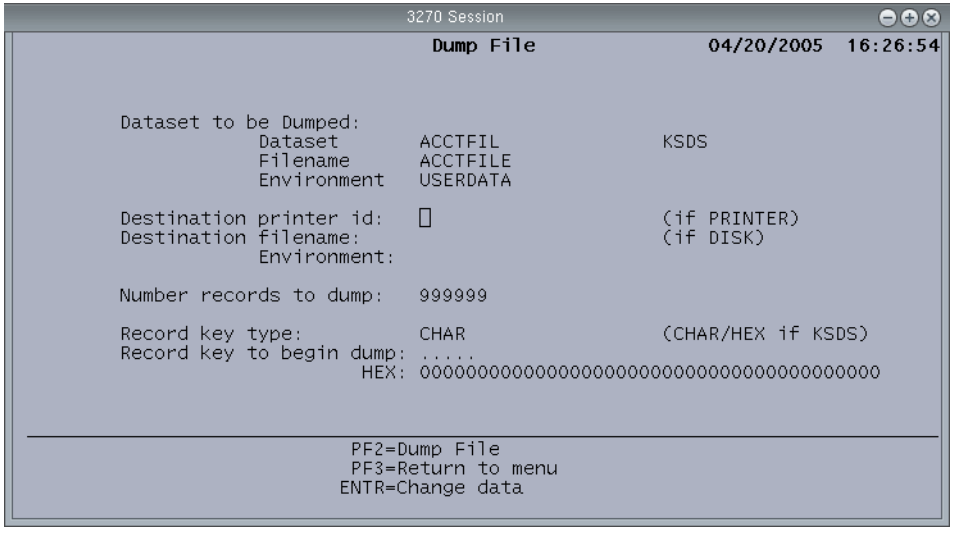


図 3-12　Data File Editor—Dump File 画面

2. 各フィールドに適切な値を入力します。

   表 3-3 は、フィールドの説明と可能な値を示します。

   入力した情報を変更するには、Enter キーを押してカーソルを最初の編集可能フィールドに移動します。フィールド間の移動には、Tab キーを使用します。

3. PF2 キーを押してデータセットをダンプします。

4. PF3 キーを押して、「Data File Editor Menu」画面に戻ります。

次の表に、「Dump File」画面のフィールドを示します。

**表 3-3** 「Dump File」画面のフィールド

| フィールド | 説明 |
|---|---|
| Destination Printer ID | 書式付きダンプを印刷するためのプリンタの名前。<br>プリンタ ID を `PRINTER` に設定すると、出力がシステムプリンタにスプールされます。<br>任意の名前を指定することも可能です。指定した名前は `kixprint` シェルスクリプトに `-p` パラメータとして渡されます。インストール時の設定によって、`kixprint` シェルスクリプトがこの名前を物理プリンタ名 (たとえば `lp0` または `lp1`) として使用できます。また `lp0-comp` または `lp0-norm` のような任意の名前を使用して、`lp0` 圧縮印刷や `lp0` 通常印刷を示すことも可能です。さまざまな制御オプションをプリントスプーラデーモンに送るには、`kixprint` シェルスクリプトをカスタマイズします。 |
| Destination Filename | 書式付きダンプを書き込むディスクファイル名。名前は、14 文字以下で、最初の文字は英文字である必要があります。 |
| Destination Environment | ファイルを格納するディレクトリを指定する環境変数。これは、「Destination Filename」フィールドにファイル名が指定されている場合に必要です。 |
| Number Records to Dump | ダンプ対象のレコード数。値を入力しない場合は、ファイル内のすべてのレコードがダンプされます。 |
| Record Key Type | レコードキーフィールドのデータの解釈方法を指定します。KSDS データセットの場合は必須フィールドです (ESDS または RRDS データセットでは使用しない)。次のいずれかを入力します。<br>`CHAR`: 文字キー (デフォルト)<br>`HEX`: 16進数 |
| Record Key to Begin Dump | ダンプを開始する最初のレコードのキー。どのデータタイプでも、キーを指定しない場合は、ファイルの最初のレコードからダンプが開始されます。<br>KSDS データセットの場合、ダンプを開始するために部分キーが使用されます。この部分キーの長さは、キー入力をスキャンして判定されます。最後の有意な (空白でない) 文字によって、ダンプに使用されるキー長が決定されます。<br>ESDS データセットの場合は、相対バイトアドレスを入力します。<br>RRDS データセットの場合は、相対レコード番号を入力します。 |

「Dump File」画面では、次のファンクションキーが利用できます。

| ファンクションキー | アクション |
|---|---|
| PF2 | データセットをダンプします。 |
| PF3 | 「Data File Editor Menu」に戻ります。 |
| Enter | ダンプ指定を変更します。 |

## ▼ 書式付ダンプファイルを表示する

- **次のいずれかの方法を使用します。**
  - コマンドプロンプトで、`more` *filename* と入力します。
  - テキストエディタでダンプファイルを開きます。

## `kixfile` での VSAM データセットの操作

`kixfile` ユーティリティーを使用すると、領域の実行中に VSAM データセットの割り当てと解放が可能です。初期化時には、FCT で deferred open が指定されたデータセットを除き、すべての VSAM データセットが開きます。VSAM データセットのクローズ、操作、およびオープンを行うには、バッチシェルスクリプトから `kixfile` を使用する必要があります。

VSAM データセットにアクセスする必要があるのは、次のような場合です。

- データセットがほかの領域で使用できるように、現在の領域から解放するため
- データセットに対して適切なコマンドまたは方法を使用して、新しいデータセットに置き換えるため
- バッチ用にデータセットを確保するため
- データセットの回復属性を変更するため

`kixfile` ユーティリティー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。

## `unikixbld` での VSAM データセットの操作

`unikixbld` ユーティリティーを使用すると、領域を実行しながら、VSAM データセットに対して次のようなさまざまな操作を実行できます。これには、バッチシェルスクリプトから `unikixbld` を使用する必要があります。

- 順編成ファイルからの VSAM データセットの構築 (サンプルスクリプトは コード例 3-2)
- 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および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
- 順編成ファイルから VSAM データセットを更新するための即時更新機能の提供

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。

# VSAM データセットの保全性の管理

VSAM データセットの保全性を管理するには、ソフトウェア組み込みの回復機能と管理手順を組み合わせて使用する必要があります。

組み込みの回復機能を構成すると、トランザクションの強制的な中止、デッドロック、バッチプログラムの強制的な中止、`unikixbld` 置換オプション、またはシステムクラッシュの際に効果を発揮します。

データセットのバックアップを保管することによって、障害が発生したときに、信頼できるコピーからデータセットを復元できます。VSAM データセットを復元する必要があるのは、次のような場合です。

- ディスククラッシュ
- アプリケーション中のバグにより、データレコードの一部が破壊された場合
- セキュリティーの侵害
- 回復不能な VSAM データセットに影響する UNIX システムクラッシュ

データセットの有効なコピーを常に確保する必要があります。このコピーに戻る必要がある場合は、そのデータが十分新しく、ニーズを満たしていることを確認します。たとえば、バックアップを毎晩作成している場合は、営業時間内に昨夜のファイルに戻すことによってニーズが満たされるかどうかを確認します。ニーズに合わない場合は、バックアップコピーを作成する間隔を短縮するか、最新のバックアップが利用できるように別のバックアップ方法を実装する必要があります。

## データセットのバックアップ

データセットの有効なバックアップコピーを確実に行うには、すべての更新アクティビティーを停止し、すべての VSAM ブロックをディスクに書き出す必要があります。これが保証できるのは、領域が停止している場合か、データセットがクローズ、ロック、または読み取り専用状態になっている場合に限られます。

---

**注 –** バッチシェルスクリプトで `kixfile` を使用する必要があります。

---

次のコマンドでデータセットを閉じます。

```
kixfile -fY datasetname
```

または次のコマンドを使用して、バックアップコマンドが入ったシェルスクリプトでデータセットを排他的にロックします。

```
kixfile -lY datasetname
```

または次のコマンドを使用して、データセットを読み取り専用に設定します。

`kixfile -dY` *datasetname*

上記のいずれかを実行したあとで、選択した方法に基づき、データセットをバックアップできます。KSDS データセットをバックアップする場合は、このデータセットを構成する `.dta` ファイルと `.idx` ファイルの両方をバックアップする必要があります。

**例**: `cp` コマンドを使用して `TEST1` という名前の KSDS データセットをバックアップします。`TEST1` を含むディレクトリに変更し、次のコマンドを入力します。

```
$ cp TEST1.dta TEST1.dta.sv
$ cp TEST1.idx TEST1.idx.sv
$ sync
```

# データセットの復元

信頼できる最新のデータセットのコピーに戻すには、領域を停止するか、データセットを閉じる必要があります。そのあと、選択した方法に基づき、データセットを復元できます。

**例**: 前の節の例で、データセットのコピーを復元するには、次のコマンドを入力します。

```
$ cp TEST1.dta.sv TEST1.dta
$ cp TEST1.idx.sv TEST1.idx
$ sync
```

# データセットの保全性の確認

Sun MTP ソフトウェアは、複数のファイルを使用して、VSAM データの記憶方式を実装します。

- KSDS データセットの場合、2 つのファイルを使用します。
- KSDS データセットがスパンされている場合、最高 7 つの追加ファイルが使用できます。
- KSDS データセットに代替索引がある場合は、それぞれの代替索引に対して、2 つの追加ファイルが使用されます。

保全性を維持するために、アクティビティーカウントが使用されます。この方法では、各ファイルヘッダーにアクティビティーカウントが用意され、領域が対象のファイルを開いたり閉じたりするたびにそれが増分されます。ファイルを開くと、アクティビティーカウントが増分されたのち、その負の補数に設定されます。データセットを閉じると、アクティビティーカウントが正の補数に再設定されます。領域がVSAM データセットを開いた場合は、そのデータセットのすべてのコンポーネントでアクティビティーカウントが等しくなります。つまり、主ファイルとすべての代替索引ファイルの索引およびデータコンポーネントについて、アクティビティーカウントが等しくなければなりません。データセットがスパンされている場合は、各セグメントのアクティビティーカウントが同じでなければなりません。

データセットの 1 つのコンポーネントで、アクティビティーカウントがほかのコンポーネントと異なる場合、データセットを開くことはできません。データセットを開く処理は無視され、なぜアクティビティーカウントに不整合が生じたのかを判定し、エラーを無視するか修正するかを決定する時間が与えられます。データセットのすべてのコンポーネントが同じバックアップから復元されていない場合、アクティビティーカウントに不整合が生じることがあります。

アクティビティーカウントを表示および変更するには、`kixverify` ユーティリティーを使用します。`kixverify` 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。

## 破壊されたデータセットの特定

データセットが破壊された疑いがある場合は、閉じたデータセットに対し、`kixvalfle` に `-ik` オプションを指定して使用することによって、保全性をチェックします。ご購入先の担当者が問題を分析するために必要になることがありますので、ファイルの出力を取得してください。 `kixvalfle` は、データファイルのブロック間のリンクの有効性をチェックします。ファイルが KSDS または代替索引ファイルの場合、索引ファイルのブロック間のリンクもチェックされます。

`kixvalfle` がエラーを返さない場合、データセットは破壊されていないので、特に措置は必要ありません。ただし、ファイルの大部分が空きブロックの場合、再編成が必要です。詳細は、106 ページの「データセットの再編成」を参照してください。

`kixvalfle` がエラーをレポートした場合、そのデータセットに対して修正のための措置が必要です。エラーメッセージによって、適切な措置が指示されます。

- 致命的エラーがレポートされた場合は、107 ページの「破壊されたデータセットの回収」で説明する手順を使用してデータセットを回復するか、バックアップからデータセットを復元します。
- 索引ファイルまたは代替索引ファイルがメインデータファイルと同期していない場合、`unikixbld` ユーティリティーを使用して、データファイルを再構築せずに索引ファイルを再構築します。

索引ファイルを再構築するための unikixbld のオプションは次のとおりです。

| | |
|---|---|
| -ta | データセットのすべての代替索引を再構築します。 |
| -tx | 主データファイルの索引ファイルを再構築します。 |

kixvalfle と unikixbld の詳細については、『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。

また、『Sun Mainframe Transaction Processing ソフトウェア 障害追跡とチューニング』も参照してください。

# データセットの再編成

データセットの再編成が必要な理由を理解するには、VSAM の KSDS データセットの物理ブロック編成を理解する必要があります。VSAM は、KSDS レコードをそのキーの値に基づいて、昇順に格納します。論理レコードは VSAM ブロック内で昇順に保存されていて、ブロックは互いに連結されていますが、物理的には必ずしも順番に並んでいるわけではありません。

**例**: ブロック 1 がブロック 7 に連結し、さらにブロック 7 がブロック 3 に連結する場合を想定します。ブロック 1 には、最小のキー値を持つ論理レコードのグループがあり、ブロック 1 の中ではこれらのレコードが昇順に並んでいます。ブロック 7 には、論理レコードが昇順に並んでいる次のグループがあり、次の論理レコードのグループはブロック 3 にあります。

論理レコードが削除されると、ブロックが空になることがあります (論理レコードが存在しない)。このような空のブロックは、*free* としてマークされ、オペレーティングシステムには返されません。この空のブロックは、新しいレコードがデータセットに挿入されるときに再利用されます。空のブロックが存在しない場合、オペレーティングシステムに対して、新しいブロックが要求されます。

データセットを再編成すると、空きディスク領域が再生されるので、ディスクアクセスが向上することがありますが、その度合いはデータセットの断片化の程度とファイル作成時、ファイル更新時、およびファイル再編成時のファイルシステムによって異なります。また、オペレーティングシステムや、ご使用のディスクコントローラのタイプによって異なります。物理ディスクアクセスの動作は予測がむずかしく、データセットの寿命の間にも変化します。

## ▼ データセットを再編成する

1. kixfile コマンドを実行して、データセットを読み取り専用にします。

```
$ kixfile -d y DataSetName
```

2. unikixbld ユーティリティーを使用して、出力を順編成ファイルに書き込みます。

```
$ unikixbld -d DataSetName -ts -s SeqFile -r recordv
```

3. unikixbld を使用して、順編成ファイルの内容を復元します。

```
$ unikixbld -d DataSetName -tv -s SeqFile -r recordv
```

unikixbld には、充てん率を指定するオプションがあります。手順 3 で使用されているコマンドでは、デフォルトの充てん率である 0 が使用されます。これにより、すべてのブロックに、可能な限りの論理レコードが格納されます。将来挿入される新しいレコードが、現在のセットよりも大きな値のキーを持つとわかっている場合には、この方法が適しています。ただし、既存の VSAM ブロックにレコードが新しく挿入されると考えられる場合には、これらを挿入するための領域を残しておきます。充てん率を使用すれば、この領域の量を指定できます。

# 破壊されたデータセットの回収

kixsalvage ユーティリティーは、データセットの回収に使用します。kixsalvage を使用すると、ファイルからすべての論理レコードの抽出が試みられますが、ファイルが破損されているために、データが失われていることがあります。kixsalvage の出力を慎重に調査して、最新のバックアップに戻ったほうがよいかどうかを判定します。

## ▼ 破壊されたデータセットを回収する

1. **データセットを閉じるか領域を停止します。**
2. **kixsalvage コマンドを入力してフラットファイルの出力を生成します。**

   -c オプションを使用して、Micro Focus の変数レコード形式でファイルを生成します。これは、COBOL SORT 文を使用して Micro Focus ランタイムでソートできます。

   ```
   $ kixsalvage -c DataSetName
   ```

   -c オプションを省略すると、kixsalvage によって recordv 形式のファイルが生成され、外部ソートユーティリティーでソートできます。

3. **バッチスクリプトから unikixbld を使用して、破壊されたデータセットを初期化し、作成したソート済みファイルから内容を再ロードします。**

   kixsalvage ユーティリティーについては、『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。また、『Sun Mainframe Transaction Processing ソフトウェア 障害追跡とチューニング』も参照してください。

# VSAM の回復

Sun MTP には、VSAM データセットの組み込みの回復機能が用意されています。これらの機能を活用するには、VSAM 構成テーブル (VCT) で回復についての設定をする必要があります。回復の設定手順については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。FCT で、個別のデータセットのレベルで回復を指定するか、領域が有効なときに kixfile ユーティリティーを使用して回復を有効にする必要もあります。

このソフトウェアは、データベースの回復だけでなく、一時記憶域キュー、一時データキュー (TDQ)、および非同期トランザクションの開始 (ATI 要求) の回復をサポートしています。KSDS VSAM ファイルである TEMPSTGR ファイルは、一時記憶域キューおよび ATI 回復の両方に使用します。このソフトウェアは、一時記憶域テーブル (TST) で、これらの両方に対して構成パラメータを提供します。宛先管理テーブル (DCT) は、TDQ の回復を制御します。TST と DCT 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。

回復プロセスには、次の 2 つのタイプがあります。

- **トランザクションの強制的な中止後の回復**

  トランザクションが強制的に中止した場合に行われます。失敗したトランザクションがデータベースに影響を及ぼさないよう、トランザクションによって実行されたデータベース更新が取り消されます。これは、動的トランザクションバックアウトと呼ばれます。

  トランザクションの更新が正常に終了するまで、ほかのトランザクションは更新したレコードにアクセスできません。これにより、失敗したトランザクションのデータが強制的に中止する前にほかの正常なトランザクションで更新されることで起こるデータベースの汚染を防止します。

  110 ページの「トランザクションの強制的な中止からの回復」を参照してください。

- **システムクラッシュからの回復**

  ハードウェアの問題、Sun MTP コンポーネントのソフトウェア上の問題、またはオペレーティングシステムの障害によって生じたシステムクラッシュ後に行われます。これは、緊急再起動/回復と呼ばれます。

  110 ページの「システムクラッシュからの回復」を参照してください。

回復機能を使用していない場合、上で説明する回復処理が行われず、アプリケーション環境が壊滅的な状態になります。トランザクションの強制的な中止、システムクラッシュ、または領域クラッシュにより、データベースが無効になることがあります。これらのいずれにも該当しない場合でも、アプリケーションがほかのトランザクションの処理中の更新情報を読み込んでしまうことがあります。アプリケーション設計者は、このような不整合を理解して、それに備える必要があります。

アプリケーションを設計する場合に検討すべき回復に関するその他の問題は次のとおりです。

- **デッドロック検出**: 1 つのトランザクションが所有するリソース (レコードやキュー登録のためのリソース) を、ほかの複数のトランザクションが待っている場合の問題です。詳細は、111 ページの「デッドロックが回復に与える影響」を参照してください。
- **会話型トランザクション**: マルチユーザー環境で会話型トランザクションによって生じる問題です。詳細は、112 ページの「会話型トランザクションと回復」を参照してください。
- **データベースの整合性**: Sun 以外の RDBMS パッケージを使用する場合に必要な情報を提供します。詳細は、112 ページの「データベースの保全性の管理」を参照してください。

---

**注 –** *record* という用語は、VSAM データベースファイルにある論理レコードを指します。論理レコードのサイズは、VSAM ブロックサイズよりも大きくすることができます。

---

# トランザクションの強制的な中止からの回復

トランザクションが失敗した場合、動的なトランザクションバックアウトを使用して、データベース更新をロールバックします。

レコードがデータベースに書き込まれる前に、元のレコードのコピーが回復ファイルに書き込まれます。この「変更前イメージ」と呼ばれるコピーは、どのトランザクションがそれを作成したのかを識別します。データベースを更新する各トランザクションの始まりと終わり、およびあらゆる同期点を示すマーカレコードも回復ファイルに書き込まれます。回復ファイルは循環ファイルです。つまり、ファイルがあらかじめ定義された最大サイズに達すると、レコードが最初から再利用されます。

トランザクションが強制的に中止されると、このソフトウェアは、そのトランザクションに対応する各レコードの回復ファイルを読み取ります。失敗したトランザクションに対応する変更前イメージが、データベースに復元されます。この時点で、失敗したトランザクションによって更新されたすべてのレコードが取り消され、レコードの状態は、トランザクション実行前の状態に戻ります。

バックアウト処理の一部として、Sun MTP は、一時記憶域テーブル (TST) で回復可能と定義されたキューについて、失敗したトランザクションが一時記憶域に対して行なったすべての更新をロールバックします。また、失敗したトランザクションが発したパーティション内一時データに対する更新と、すべての回復可能な非同期 `START` 要求をロールバックします。非同期 `START` 要求は、`REQID` パラメータで回復可能キューを指定した場合に回復可能です。失敗したトランザクションが、回復可能なリソース (VSAM データセット、一時記憶域、パーティション内一時データ、および非同期 `START`) に対して行なったすべての更新が、Sun MTP によって動的トランザクションバックアウトの実行中にロールバックされます。

トランザクションバックアウトが行われている間、システムが一時停止する場合がありますが、システムのほかのトランザクションには影響しません。

# システムクラッシュからの回復

回復が有効になるように VCT を構成し、XA プロトコルを使用していない場合は、この節で説明されている回復処理が実行されます。領域が XA 用に構成されている場合、システム障害からの回復の詳細については、『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。

領域を再起動すると、システムクラッシュの時点で未完了だったすべてのトランザクションのデータベース更新が回復サーバーによって取り消されます。システムが正常に終了しなかったことを、回復ファイルのヘッダーファイル情報が示している場合、回復サーバーは、トランザクションの強制的な中止の場合と同様に、「変更前イメージ」をデータベースに復元します。ただし、この場合は回復ファイルの最後のレコードが書き込まれた時点で処理中だったすべてのトランザクションが取り消されます。結果は、正常に完了しなかったトランザクションを個別にバックアウトした場合と同じです。

**注 –** 一時データキューに回復可能のフラグが付いている場合、Sun MTP は、上で説明した標準の KSDS 回復機構を使用して、VSAM データセットを回復します。

回復の実行中にシステムクラッシュが発生した場合、システムを再起動して回復を実行すると、回復処理が続行されます。回復を実行しない場合は VCT の回復フラグをオフにし、さらに次の操作を行います。クラッシュ後は、回復ファイルがデータベースに対してバックアウトを適用しようと待機している状態になっているので、回復ファイルを消去するか、回復ファイルの名前を変更する必要があります。バックアウトが次回の Sun MTP の実行に適用されず、将来の実行に適用されると、そのバックアウト処理が不適切なために、データベースに対して予期しない重大な損傷を与える可能性があります。

## デッドロックが回復に与える影響

複数のトランザクションが、互いにほかのトランザクションが所有中のリソースを待っている場合に、デッドロックが生じます。ほかのトランザクションがその必要なリソースを解放するまで、各トランザクションが待っているので、介入しない限り、それらのトランザクションがハングアップしたままになります。

最も単純なデッドロックは、2 つのトランザクションと 2 つのリソースが関係する場合です。次に例を示します。

- トランザクション 1 がリソース A を所有し、リソース B に対して要求を発している
- トランザクション 2 がリソース B を所有し、リソース A に対して要求を発している
- トランザクション 1 はトランザクション 2 がリソース B を解放するまで処理を続行できず、一方、トランザクション 2 はトランザクション 1 がリソース A を解放するまで処理を続行できない

これらのリソースは、`ENQ` コマンドまたは VSAM レコードで定義されるタイプのリソースです。

Sun MTP は、デッドロック状態を検出する特別なロジックを持っています。デッドロックを検出すると、トランザクションのいずれかを不正終了します。不正終了になるのは、グループに最後に参加したトランザクション、つまり、ほかのトランザクションが未処理のうちに発した要求がデッドロックを生じさせたトランザクションです。このタイプの不正終了が発生した場合、不正終了したトランザクションに設計上のエラーがあるわけではありません。設計上のエラーは、グループとしてみた場合の、デッドロックを生じさせたすべてのトランザクションにあります。

# 会話型トランザクションと回復

会話型トランザクションとは、トランザクションが有効のまま、ユーザーとの会話 (通常、SEND/RECEIVE シーケンス) が行われるトランザクションです。これは、トランザクション中にユーザーとの会話がない擬似会話型トランザクションと異なります。通常、擬似会話型トランザクションは、SEND、次に RETURN TRANSID で終わります。ユーザーは応答を入力しますが、トランザクションは有効ではありません。ユーザーが Enter キーまたは PF キーを押すと、TRANSID で指定されるトランザクションの新規インスタンスが始まります。

会話型トランザクションは、Sun MTP の回復機能の使用を十分に考慮して設計する必要があります。

- 回復機能をオンにすると、トランザクションは、トランザクションが終了するまで、または同期点に達するまで、それが変更するすべての VSAM レコードの制御を保持します。これは、ユーザーからの応答を待っている間、会話型トランザクションが何秒または何分にもわたって、レコードの制御を保持することがあるということを意味します。レコードがすでに書き込まれている場合にも、これは当てはまります。回復機能がオフのシステムでは、レコードが書き込み済みとなった時点で、ほかのトランザクションがレコードを利用できます。
- トランザクションの完了にユーザーの応答が必要な場合、トランザクションの処理時間の制限がなくなります。ユーザーには更新が完了したと見えるが、更新されていないことがあります。トランザクションが完了するまで、この更新はデータベースにコミットされません。トランザクションがあとで失敗すると、更新がロールバックされていることにユーザーが気付かない場合があります。

このような問題を回避するために、会話型トランザクションを一切使用しない方法もあります。また会話型トランザクションを使用してシステムがすでに作成されている場合は、レコードが保持される可能性のある RECEIVE コマンドの直前に SYNCPOINT コマンドを追加する方法もあります。

# データベースの保全性の管理

領域は、VSAM アプリケーションのデータベースの保全性を管理します。Sun 以外の RDBMS 製品を使用するアプリケーションの保全性は、RDBMS ソフトウェアによって管理されます。

Sun MTP は、回復ファイルを使用して、データベースレコードの「変更前イメージ」を格納します。回復ファイルは、強制的な中止の場合のロールバック機能を備えています。「変更前イメージデータ」は、データベースのコミットメントまたはロールバックののちも保持されるデータではありません。Sun 以外の RDBMS は、それぞれ専用のログファイルを持ち、データベースロールバックとロールフォワード機能を実現しています。RDBMS ログファイルの構成と使用方法については、RDBMS のドキュメントを参照してください。

アプリケーションの設計者は、特定のアプリケーションでのデータベースの保全性を暗黙的または明示的に管理します。暗黙のデータベースアクションは、次のように、アプリケーションの実行中のさまざまな時点で実行されます。

- トランザクション開始時の暗黙のコミットメントの開始
- トランザクション正常終了時の暗黙のデータベースコミットメント
- トランザクション異常終了時の暗黙のデータベースロールバック

## XA 以外の環境でのデータベースの保全性の管理

XA 以外の環境 (単一フェーズコミット) では、Sun MTP は、VSAM データベースの暗黙のコミットメントやロールバックを自動的に処理し、RDBMS ソフトウェアは、RDBMS トランザクションの暗黙のコミットメントやロールバックをユーザーモジュールによって管理します。

データベース管理者は、ユーザーモジュールを用意して、トランザクションサーバー (`unikixtran`) とバッチ処理プログラム (`unikixvsam`) にバインドする必要があります。ユーザーモジュールは、アプリケーションとの整合性を保証するために、アプリケーション設計者の協力を得て開発する必要があります。ユーザーモジュールの開発についての詳細は、第 10 章を参照してください。

アプリケーションプログラムは、データベースのコミットを明示的に要求できます。推奨される方法は、`SYNCPOINT` 関数の呼び出しを使用することです。これにより、トランザクション処理プログラムに統合されたすべての RDBMS ソフトウェアが呼び出されて、その変更を適切なデータベースにコミットし、トランザクションが正常に完了したことがログファイルに記載されます。

RDBMS アプリケーションでは、`COMMIT WORK SQL` 文を使用して、データベースの更新をコミットするか、`ROLLBACK WORK SQL` 文を使用して、データベースの変更内容を削除できます。どちらの場合も、操作は、特定の RDBMS によって管理されるデータに対してのみ実行されます。1 つの RDBMS のみがアプリケーションに使用されている場合は、この方法によって、明示的にコミットメントを要求します。そうでない場合は、アプリケーションデータベースの不整合が生じることがあります。この方法を使用するアプリケーションは、XA 環境で実行できなくなります。

## マルチデータベース環境 (XA) でのデータベースの保全性の管理

マルチデータベース環境でデータベースの保全性を維持するには、XA プロトコル (2 フェーズコミット) を使用するように Sun MTP を構成する必要があります。
Sun MTP XA の実装の詳細については、『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。

# VSAM キャッシュの使用

アプリケーションのスループットに対する要求が、システムクラッシュやトランザクションの強制的な中止からの回復に対する要求よりも大きい場合は、このソフトウェアの VSAM キャッシュ機能を使用して、データセットおよびジャーナルへの物理ファイル書き込みを遅延します。ファイルシステムのキャッシュは、そのキャッシュフラッシュ操作規則に基づいて、このような物理書き込みをスケジュールします。

日に 1 回以上の頻度で、システム上の VSAM データセットを適切にバックアップするための方針を導入するのは、システム管理者の責任です。したがって、`kixfile` または UNIX `sync` コマンドを別途使用して、キャッシュに入っている VSAM データセットやジャーナルについての更新内容が、必要に応じて物理的に書き込まれるようにする必要があります。

Sun MTP は、回復不能な一時記憶域ファイルである `TEMPSTG` と、回復不能な一時データファイルを自動的にキャッシュします。これらのファイルは領域の起動時に初期化されるので、領域やシステムのクラッシュ後はこれらのデータは不要になります。同様に、ユーザージャーナルのような一定のカスタマアプリケーションファイルは、最後の数ブロックが失われただけの場合、システム障害のあとでも使用可能です。

VSAM データセットとジャーナルファイルのキャッシュをファイル管理テーブル (FCT) とジャーナル管理テーブル (JCT) で指定できるほか、VSAM データセットのキャッシュを `kixfile` ユーティリティーで動的に指定できます。

## ▼ VSAM データセットのキャッシュを指定する

1. **空白のトランザクション画面で `CTBL` トランザクションを入力して、Table Manager を開きます。**
2. **PF4 キーを押して、「Standard Tables」を選択します。**
3. **PF5 キーを押して、FCT を表示します。**

4. キャッシュ書き込みを行うデータセットを選択し、「No Rcv」列に C を入力します。

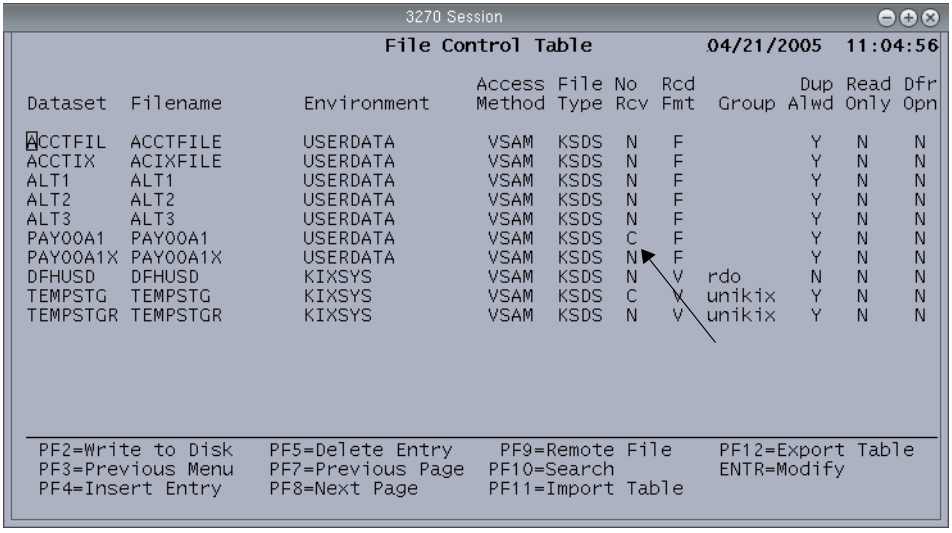


図 3-13　File Control Table—ファイルキャッシュの指定

5. キャッシュする各データセットに対して、手順 4 を繰り返します。
6. PF2 キーを押して、変更内容をディスクに保存します。
7. Table Manager を終了します。
8. 領域を停止して再起動し、変更内容を有効にします。

## ▼ ジャーナルファイルのキャッシュを指定する

1. 空白のトランザクション画面で CTBL トランザクションを入力して、Table Manager を開きます。
2. PF5 キーを押して、拡張テーブルを選択します。
3. PF6 キーを押して、JCT を開きます。

4. キャッシュ書き込みを行うジャーナルファイルを選択し、「Opt」列に C を入力します。

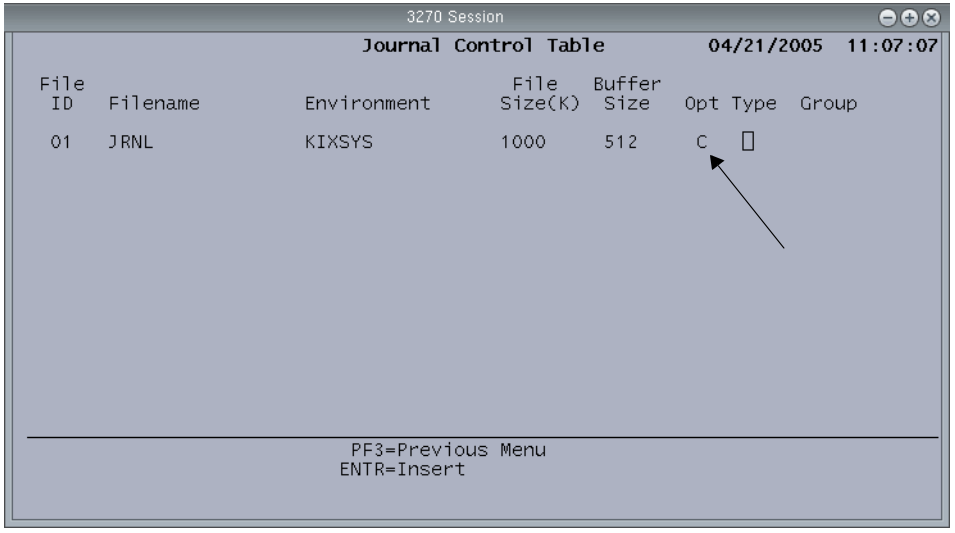


図 3-14　Journal Control Table—Journal キャッシュの指定

5. キャッシュする各ジャーナルファイルに対して、手順 4 を繰り返します。
6. PF2 キーを押して、変更内容をディスクに保存します。
7. Table Manager を終了します。
8. 領域を停止して再起動し、変更内容を有効にします。

## キャッシュ書き込みの動的指定

kixfile ユーティリティーの -rC オプションをほかの適切なオプション ( -fN など) とともに使用すると、データセットに対する書き込みがキャッシュされるようにそのデータセットの状態を動的に変更できます。これには、領域の実行中に、kixfile をシェルスクリプトからバッチジョブとして実行する必要があります。

形式:

```
kixfile {({-f|-o|-b|-l|-d} Y|N [-r Y|N|C])|[-p]} dataset-name
```

kixfile 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。

**例**: 回復可能な VSAM データセット PRODDATA への書き込みのキャッシュを有効にするには、次のコマンドを記述したバッチシェルスクリプトを実行します。

```
#Close the VSAM dataset:
kixfile -fY PRODDATA
#Reopen the VSAM dataset:
kixfile -fN -rC PRODDATA
```

PRODDATA 属性を復元する、つまりキャッシュを無効にするには、次のコマンドを記述したバッチシェルスクリプトを実行します。

```
#Close the VSAM dataset:
kixfile -fY PRODDATA
#Reopen the VSAM dataset:
kixfile -fN -rY PRODDATA
```

# 第4章

# アプリケーションのワークロード管理

トランザクションクラスおよびスケジューラプロセスは、領域の異なる負荷要件を管理します。トランザクションクラスを定義すると、領域の起動時にスケジューラプロセスが自動的に起動します。

この章では、次のトピックについて説明します。



**注 –** EPI クライアントから開始されたトランザクションは、それらがユーザー定義のトランザクションクラスに割り当てられている場合でも、常に KIXDFLT トランザクションクラスで実行されます。

XA リソースマネージャーを使用する場合は、トランザクションクラスを有効に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。

# トランザクションクラスとは

「トランザクションクラス」とは、共通の優先順位を持つトランザクションの物理グループです。トランザクションクラスの優先順位は、そのクラスに割り当てるトランザクション処理プログラムを増減することによって調節できます。

アプリケーション環境が次のような場合は、トランザクションクラスの使用を検討してください。

- トランザクションまたはトランザクションセットの優先順位が高い
- 実行時間が長いか、多くの CPU 時間を消費するため、シングルスレッドにする必要があるトランザクションがある
- バッチおよびオンラインプロセスが同時に起こる

# トランザクションクラスの設定

KIXDFLT と KIXADMIN という 2 つのシステムトランザクションクラスがあります。これらのシステムクラスには、デフォルトで、トランザクション処理プログラムが 1 つずつ割り当てられます。トランザクション処理プログラムを増加させることはできますが、1 未満に減らすことはできません。

---

**注 –** KIXDFLT と KIXADMIN の 2 つのクラスを削除することはできません。

---

KIXDFLT は、デフォルトのトランザクションクラスです。

- `CEMT`、`CINI`、および `CSMT` 以外のすべてのシステムトランザクションは、KIXDFLT クラスに割り当てられます。
- ユーザー定義のトランザクションで、PCT 中で明示的にトランザクションクラスを定義していない場合は、すべて KIXDFLT クラスに割り当てられます。
- 管理者は、システムトランザクションの割り当てを KIXDFLT クラスから KIXADMIN クラスまたはユーザー定義のクラスに変更できます。

KIXADMIN は、管理者トランザクションクラスです。

- `CEMT`、`CINI`、および `CSMT` の優先順位の高いシステムトランザクションは、KIXADMIN クラスに割り当てられます。これらのトランザクションは、割り当てられたトランザクション処理プログラムがビジーである場合、利用可能な任意のトランザクション処理プログラムで実行されます。すべてのトランザクション処理プログラムがビジーである場合、トランザクションはキューに入れられます。

- KIXADMIN クラスは、領域内に優先順位があります。管理者は必要に応じて、ほかのトランザクションを KIXADMIN クラスに割り当てたり、このクラスにトランザクション処理プログラムを追加することができます。
- 管理者は、CEMT、CINI、および CSMT の割り当てをほかのクラスに変更できます。
- 管理者は、任意のトランザクションを KIXADMIN に割り当てることができます。

## ▼ トランザクションクラスを定義する

1. **トランザクションクラスを定義する領域を起動します。**
2. **空白のトランザクション画面で** CTBL **と入力して Table Manager を起動します。**
3. **PF5 キーを押して、「Extended Tables」メニューを表示します。**
4. **PF1 キーを押して、図 4-1 に示す「Transaction Class Table」画面を開きます。**

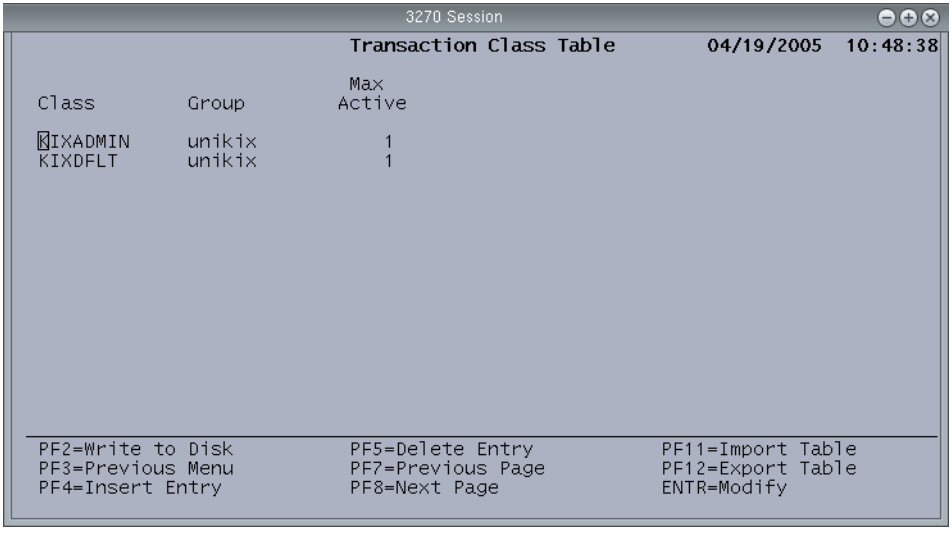


**図 4-1** トランザクションクラステーブル (TXC)

5. **PF4 キーを押して、トランザクションクラスを追加します。**
   トランザクションクラスは、最高 62 まで定義できます。

6. 次のフィールドに値を入力します。

   a. 「Class」フィールドに、8 文字までのトランザクションクラスの名前を入力します。

   b. グループを使用している場合は、8 文字までの「Group」フィールドにグループ名を入力します。

   c. 「Max Active」フィールドには、このクラスで同時に実行できるトランザクションの最大数を指定します。

      この数はまた、この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でもあります。指定できる値は 0 ～ 1024 の範囲です。値 0 は、このクラスに対して届けられるメッセージがすべてキューに入れられ、`CEMT SET TRANCLASS` コマンドまたは `EXEC CICS SET TRANCLASS` コマンドによってこのクラスにトランザクション処理プログラムが割り当てられるまでキューに留まることを示します。

7. トランザクションクラスを追加し終わったら、Enter キーを押して、エントリをテーブルに追加します。

8. PF3 キーを押すと、「Transaction Class Table」画面に戻ります。

9. PF2 キーを押して、変更をディスクに書き込みます。

10. PF3 キーを 2 回押して、「Table Manager」メインメニューに戻ります。

---

**注 –** VSAM 構成テーブル (VCT) は、領域の起動時に開始されるトランザクション処理プログラムの合計数を定義します。この数は、TXC テーブルの Max Active カウントの合計数以上でなければなりません。VCT 内のトランザクション処理プログラムの数が、TXC テーブル内の Max Active カウント合計より大きい場合は、残りのトランザクション処理プログラムが KIXDFLT クラスに割り当てられます。

---

# トランザクションのクラスへの割り当て

領域の PCT 内のクラスに、トランザクションを割り当てます。クラスを明示的に割り当てられていないトランザクションは、KIXDFLT クラスに割り当てられます。

## ▼ トランザクションをクラスに割り当てる

1. 「Table Manager」メニューで PF4 キーを押して、「Standard Tables」メニューを表示します。
2. PF6 キーを押して、PCT を表示します。
3. トランザクションを選択し、PF9 キーを押して、トランザクションクラス画面を表示します。

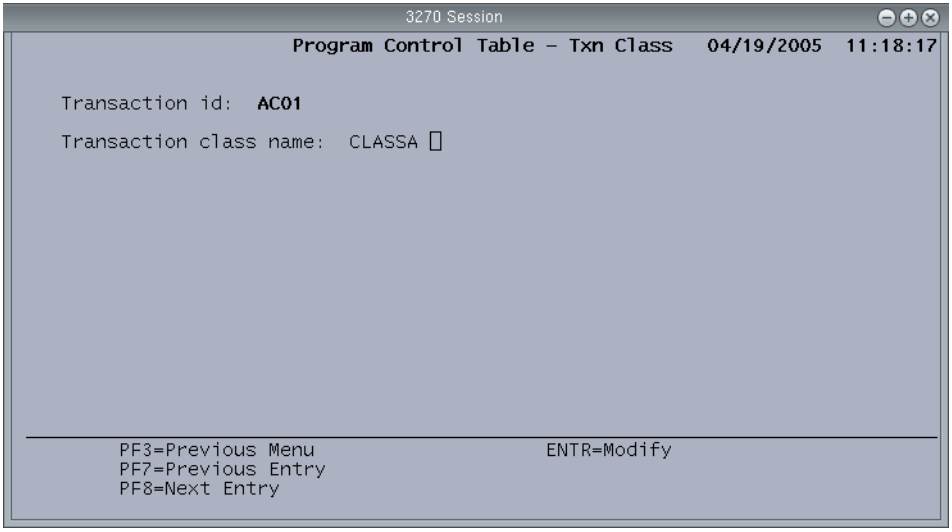


図 4-2　PCT – トランザクションクラス画面

4. トランザクションを割り当てるクラスの名前を入力します。

   PF7 と PF8 キーを使用して、割り当てるトランザクションがほかにあるかどうかを確認します。
5. Enter キーを押して変更内容を確定し、PCT メイン画面に戻ります。
6. PF2 キーを押して、変更をディスクに書き込みます。
7. PF3 キーを 2 回押して、「Table Manager」メインメニューに戻ります。

# トランザクションクラスの削除

トランザクションクラスを削除する前に、そのクラスに割り当てられている PCT のすべてのエントリを変更する必要があります。この節では、2 つの手順について説明します。変更するトランザクションが少数の場合に使用する方法と、多数のトランザクションを変更する場合に使用する方法です。

- エントリを変更するトランザクションが少数の場合は、Table Manager で手動で行います。
- 多数のトランザクションが割り当てられているトランザクションを削除する場合は、テーブルの `.lst` ファイルを使用して変更を行います。

---

**注 –** PCT のトランザクションエントリを変更せずにトランザクションクラスを削除しても、メッセージは表示されません。それらのトランザクションは、KIXDFLT クラスで実行されます。KIXDFLT クラスに割り当てられているトランザクション処理プログラムの数が少ない場合は、これによってパフォーマンスに影響が出ることがあります。

---

## ▼ トランザクションクラスを手動で削除する

1. 「Table Manager」メニューで PF4 キーを押して、「Standard Tables」メニューを表示します。
2. PF6 キーを押して、PCT を表示します。
3. クラスを削除するトランザクションごとに、トランザクションを選択して PF9 キーを押し、「Transaction Class」画面を表示します。
4. トランザクションを割り当てるクラスの名前を変更します。

   「Transaction Class」画面で、PF7 と PF8 キーを使用すると、変更するトランザクションがほかにあるかどうかを確認できます。
5. Enter キーを押して変更内容を確定し、PCT メイン画面に戻ります。
6. PF2 キーを押して、変更をディスクに書き込みます。
7. PF3 キーを 2 回押して、「Table Manager」メインメニューに戻ります。
8. PF5 キーを押して、「Extended Tables」メニューを表示します。
9. PF1 キーを押して、「Transaction Class Table」画面を開きます。

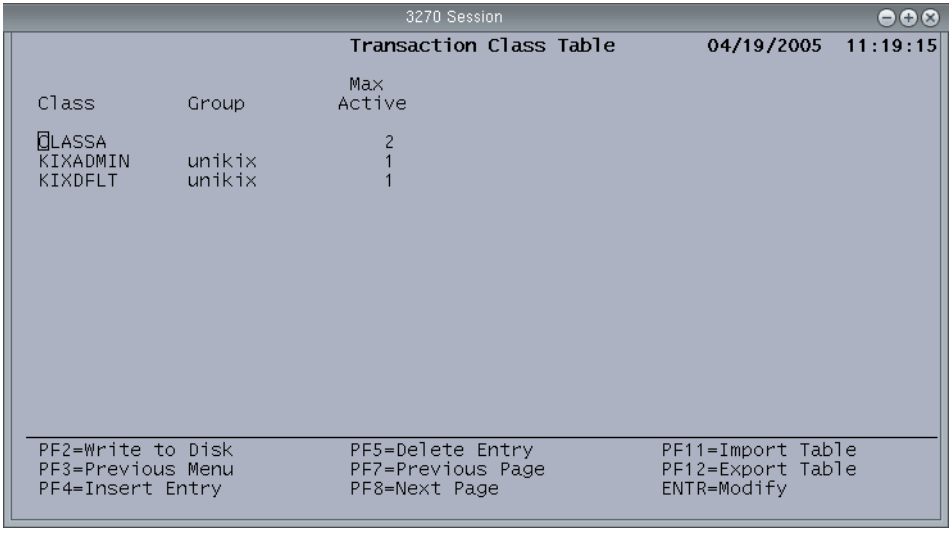


図 4-3　　トランザクションクラスの削除

10. 削除するトランザクションクラスを選択し、PF5 キーを押します。
    確認のプロンプトが表示されたら、Enter キーを押してクラスを削除します。
11. PF2 キーを押して、変更をディスクに保存します。
12. Table Manager を終了します。
13. これらの変更内容を有効にするには、領域を停止して再起動する必要があります。

## ▼ .lst ファイルでトランザクションクラスを削除する

1. 「Table Manager」メニューで PF4 キーを押して、「Standard Tables」メニューを表示します。
2. PF6 キーを押して、PCT を表示します。
3. PF12 キーを押して、テーブルをエクスポートします。
4. テキストエディタを使用して、次の操作を行います。
   a. `pct.lst` ファイルを開きます。
   b. クラスの名前を、削除するクラスから、そのトランザクションを割り当てる新しいクラスに置き換えます。
   c. `pct.lst` ファイルを保存します。

5. PCT のメイン画面で PF11 キーを押して、変更した `pct.lst` ファイルをインポートします。
6. PF2 キーを押して、変更をディスクに書き込みます。
7. PF3 キーを 2 回押して、「Table Manager」メインメニューに戻ります。
8. PF5 キーを押して、「Extended Tables」メニューを表示します。
9. PF1 キーを押して、「Transaction Class Table」画面を開きます。

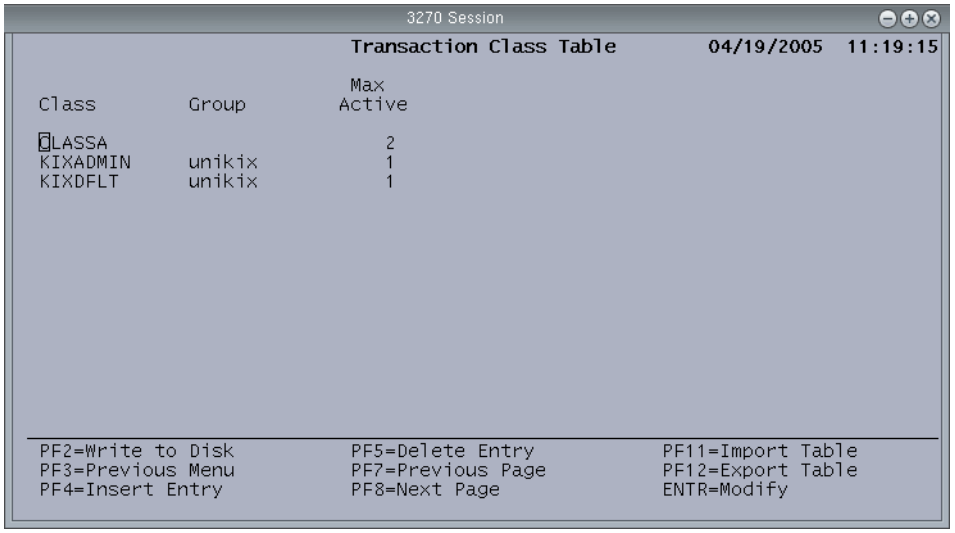


**図 4-4** トランザクションクラスの削除

10. 削除するトランザクションクラスを選択し、PF5 キーを押します。
    確認のプロンプトが表示されたら、Enter キーを押してクラスを削除します。
11. PF2 キーを押して、変更をディスクに保存します。
12. Table Manager を終了します。
13. これらの変更内容を有効にするには、領域を停止して再起動する必要があります。

# トランザクションクラスの監視

トランザクションクラスを監視する方法は、次の 3 つがあります。

- `kixdump -St` コマンド
- `CEMT INQUIRE` トランザクション
- Sun MAT。管理ツールの詳細については第 11 章を参照してください。トランザクションクラスの監視については、『Sun Mainframe Administration Tool ユーザーズガイド』を参照してください。

`kixdump -St` コマンドは、次の統計情報に加え、サポート技術者のための、その他の統計情報を表示するレポートを作成します。

| 統計 | 説明 |
|---|---|
| Max active | TXC でクラスに割り当てられているトランザクションの数。 |
| Current max active | そのクラスに割り当てられているトランザクション処理プログラムの現在の数。クラスに構成されているトランザクション処理プログラムの数が異ならない限り、Max active と同じです。 |
| Current queue depth | そのクラスでの処理を待っているメッセージまたはトランザクションの数。 |
| Maximum queue depth | これまでにキューで待機したことのあるメッセージまたはトランザクションの最大数。この履歴上の数は、`unikixmain` の `-i` オプションによって定義した統計間隔時間ごとに判定、更新されます。 |
| Messages received | このクラスに対して受信されたメッセージまたはトランザクションの合計数。 |
| Last send time | このクラスに割り当てられたメッセージキューに対して送信された、最新メッセージの日付/時刻スタンプ。`"No activity"` という表示は、このクラスに送られたメッセージが 1 つもないことを示します。 |
| Last receive time | このクラスに割り当てられたメッセージキューから受信またはピックアップされた最新メッセージの日付/時刻スタンプ。`"No activity"` という表示は、このクラスから受け取ったメッセージが 1 つもないことを示します。 |

次に、kixdump -St コマンドの出力例を示します。

**コード例 4-1** kixdump -St 出力の例

```
05/09/2005 16:07:25 kixdump     :entering version 8.1.0 - 05/05/2005

Transaction Class Statistics:

Class:CLASS_1  queue #:256  reconfig id:000000  user area:<c12b6008>

   Max active        :0001  Current q depth :000000  Msgs received: 000000004
   Cur max active    :0001  Maximum q depth : 000000
   Last send time    :Mon May 09 16:04:37 2005
   Last receive time :Mon May 09 16:04:37 2005

Class:CLASS_4  queue #:257  reconfig id:000000  user area:<c12b8100>

   Max active        :0004  Current q depth :000000  Msgs received: 000000001
   Cur max active    :0004  Maximum q depth : 000000
   Last send time    :Mon May 09 16:03:32 2005
   Last receive time :Mon May 09 16:03:32 2005

Class:CLASS_2  queue #:258  reconfig id:000000  user area:<c12c04e0>

   Max active        :0002  Current q depth :000006  Msgs received: 000000014
   Cur max active    :0002  Maximum q depth : 000006
   Last send time    :Mon May 09 16:06:03 2005
   Last receive time :Mon May 09 16:05:19 2005

Class:KIXADMIN queue #:259  reconfig id:000000  user area:<c12c46d0>

   Max active        :0001  Current q depth :000000  Msgs received: 000000001
   Cur max active    :0001  Maximum q depth : 000000
   Last send time    :Mon May 09 16:04:05 2005
   Last receive time :Mon May 09 16:04:05 2005

Class:KIXDFLT  queue #:260  reconfig id:000000  user area:<c12c67c8>

   Max active        :0001  Current q depth :000000  Msgs received: 000000019
   Cur max active    :0002  Maximum q depth : 000000
   Last send time    :Mon May 09 16:04:45 2005
   Last receive time :Mon May 09 16:04:45 2005
```

サンプル出力では、CLASS_2 に、トランザクション処理プログラムが 2 つ割り当てられていることがわかります (Max active の値)。現在のキューの深さが 6 と表示されていますが、これは、6 つのトランザクションが実行を待っていることを示します。その他のすべてのクラスは、現在のキューの深さが 0 と表示されています。パフォーマンスのバランスを図るために、どのトランザクションが CLASS_2 に割り当てられるかを判定し、1 つ以上のトランザクションをほかのクラスに移動するかどうか、また CLASS_2 に割り当てるトランザクション処理プログラムを増加させるかどうかを決定します。

---

**注 –** KIXDFLT クラスでは、Max active の値が 1、Cur max active の値が 2 と表示されています。これは、VCT で定義されているが、どのクラスにも割り当てられていない余分のトランザクション処理プログラムが、1 つ存在していたことを示します。領域の起動時に、この余分のトランザクション処理プログラムは、KIXDFLT に割り当てられます。

---

`CEMT INQ` トランザクションでは、1 つまたはすべてのクラスについての情報を表示できます。

## ▼ 特定のトランザクションクラスを監視する

- **次のトランザクションを実行します。**

```
CEMT INQ TRANCLASS CLASS_2
```

## ▼ すべてのトランザクションクラスを監視する

- **次のトランザクションを実行します。**

```
CEMT INQ TRANCLASS ALL
```

図 4-5 に示されている画面では、Class、max active、current max active、number of messages、current queue depth、および maximum queue depth といった `kixdump -St` 情報のサブセットがレポートされます。

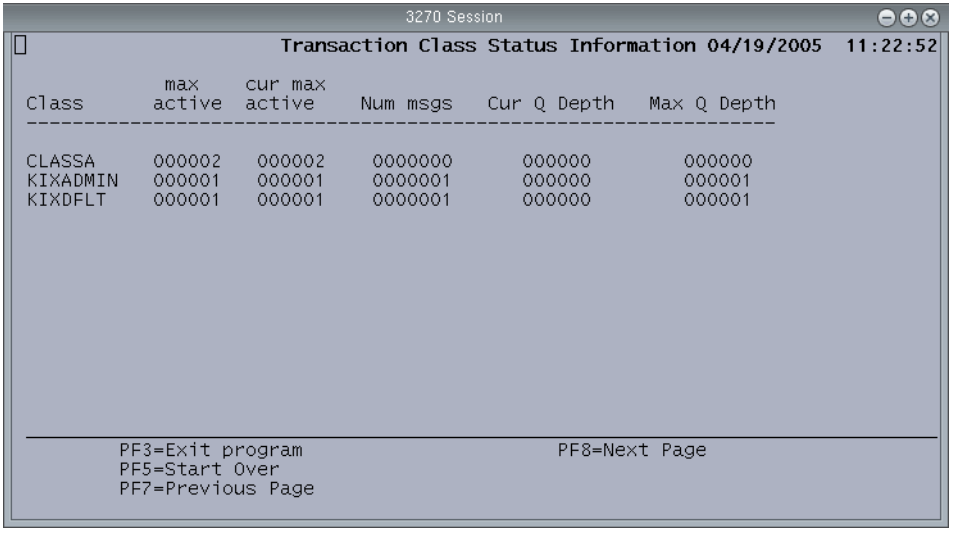


**図 4-5** Transaction Class Status Information レポート

# トランザクションクラスの管理

領域のトランザクションクラスを構成してから、そのクラス対トランザクション処理プログラム比率では領域が効率的に動作しないことが明らかになった場合、クラスに割り当てるトランザクション処理プログラムの数を動的に再構成できます。CEMT SET TRANCLASS トランザクションは、トランザクション処理プログラムの割り当てをあるクラスから別のクラスに変更します。この節の例では、領域のトランザクションクラスを管理するいくつかの方法について説明します。

## トランザクションクラスの使用制限

トランザクションが実行可能な時間を限定するには、トランザクションクラスの作成時に、TXC テーブルの Max Active フィールドに 0 を割り当ててから、対象のトランザクションをそのクラスに割り当てます。1 日のある時刻にこのクラスでの割り当てを多くして、トランザクションを実行することができます。

領域に対して定義した各トランザクションクラスについて、領域の起動時に、追加の IPC メッセージキューが作成されることに注意してください。これにより、管理者がクラスを再構成し、1 つ以上のトランザクション処理プログラムをそのクラスに割り当てるときに利用可能な、システムリソースを確保します。また、クラスの Max Active フィールドを 0 に設定すると、そのクラスに対するトランザクションが要求されたときに、それらのトランザクションがキューに入れられます。要求されたトランザクションが多数の場合、メッセージキューに対して定義されたシステム上限に達すると、システムに問題が発生したり、パフォーマンスが低下したりすることがあります。

## トランザクション処理プログラムを再割り当てする時期の判定

kixdump -St または CEMT INQ TRANCLASS コマンドを使用して、現在のキューの深さを監視します。理想としてこの数は、実行を待つトランザクションがないことを示す、0 でなければなりません。最大のキューの深さに関する統計情報は、キューの履歴を示し、実行を待つトランザクションのこれまでの最大数を示します。現在のキューの深さが常に 0 を上回るようであれば、そのクラスへのトランザクション処理プログラムの追加を検討する必要があります。

トランザクションクラスに割り当てるトランザクション処理プログラムの数は、CEMT SET TRANCLASS コマンドを使用して容易に再構成できます。また EXEC CICS SET TRANCLASS 文を使用し、プログラムによってトランザクションクラスの再構成もできます。

**例**: クラス ABC には 2 つのトランザクション処理プログラムがあり、クラス XYZ には 6 つのトランザクション処理プログラムがあります。クラス ABC には、パフォーマンス上の問題があり、CEMT INQ TRANCLASS には、現在のキューの深さが 8 と表示されています。そこで、クラス XYZ のトランザクション処理プログラムのうち 2 つを ABC に再割り当てすることにしました。次のトランザクションを実行します。

```
CEMT SET TRANCLASS ABC MAXACTIVE 2 USE XYZ
```

ここで、CEMT INQ TRANCLASS または kixdump -St を使用してクラスを表示すると、クラス ABC の現在の Max Acitive が 4 となり、XYZ の現在の Max Active が 4 となっているのが確認できます。トランザクション処理プログラムを 2 つ増やしたことで、処理能力が増強されるので、クラス ABC の現在のキューの深さが減少します。

## バックグラウンドタスクとバッチジョブの管理

実行可能なバックグラウンドタスクとバッチジョブの数は、VCT の設定によって制御します。VCT での設定は、START によって起動したトランザクション ID または CBCH を持つクラスに対して TXC で設定した、Max Active の値に優先します。さらに、START によって起動されたトランザクションまたは CBCH と同じクラスにほかのトランザクションを割り当てると、ほかのトランザクションがトランザクション処理プログラムを使用するためにパフォーマンスが低下する場合があります。

**例**: VCT では、「Maximum background tasks」フィールドが 4 に、「Maximum batch jobs」フィールドが 1 に設定されています。TXC では、クラス 2 に 6 つのトランザクション処理プログラムが定義されています。PCT では、100 のトランザクションをクラス 2 に割り当て、その一部が START によって起動されたトランザクションです。本番稼動では、バックグラウンドタスクに使用できるトランザクション処理プログラムが 4 つしかなく、START によって起動されていないトランザクションが利用可能な 6 つのトランザクション処理プログラムを使用しているので、START によって起動されたトランザクションがすぐに実行されない場合もあります。

この種の問題を回避するには、START によって起動されたトランザクションと CBCH を独自のクラスに割り当て、各クラスの Max Active 値を、VCT でのこれに対応する設定と同じ値に設定します。

### KIXDFLT トランザクションクラスの管理

デフォルトでは、KIXADMIN に割り当てられている 3 つのトランザクションを除くすべてのシステムトランザクションが、KIXDFLT クラスに割り当てられます。1 つのトランザクション処理プログラムは、KIXDFLT に割り当てられます。PCT で定義するときに、ユーザートランザクションをクラスに指定しなかった場合、そのトランザクションを KIXDFLT に割り当てます。Clear キーの入力や未定義または未知のトランザクションのようなクラスに属さないほかのアクションは、KIXDFLT クラスに割り当てられて実行されます。パフォーマンスの問題が生じないよう、KIXDFLT クラスに十分なトランザクション処理プログラムを割り当てます。

kixdump -St または CEMT INQ TRANCLASS で KIXDFLT クラスを監視している場合、現在の Max Active の値が Max Active の値を上回ることがあり、TXC テーブルに指定されているよりも多くのトランザクション処理プログラムが KIXDFLT に割り当てられているように見えます。この矛盾は、実際には、VCT に構成されているトランザクション処理プログラムの数が、TXC ですべてのクラスに対して構成されている Max Active プロセッサの合計よりも大きく、余分のトランザクション処理プログラムが KIXDFLT クラスに割り当てられていたことを示します。このような「クッション」に頼るのではなく、KIXDFLT クラスを定期的に監視して、領域のパフォーマンスデータをもとに、トランザクション処理プログラムを割り当てる必要があります。

## API サポート

Sun MTP は、トランザクションクラスに対して次の CICS API をサポートします。

- EXEC CICS INQUIRE TRANCLASS
- EXEC CICS INQUIRE TRANSACTION
- EXEC CICS SET TRANCLASS

サポートされるオプションの詳細については、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。

第5章

# レコードの編集

レコードエディタを使用して、「Data File Editor Menu」に表示されるデータセットのレコードの追加、変更、および削除ができます。レコードの編集は 16 進形式でも文字形式でも可能です。

この章では、次のトピックについて説明します。



## レコードエディタの起動

### ▼ レコードエディタを起動する

1. **領域を開始して、クライアント接続を確立します。**
2. **図 5-1 のような「Data File Editor Menu」を表示します。**
   - ローカルクライアントでは、空白のトランザクション画面で `CMNU` と入力して「Development System」メインメニューを開いてから、PF8 キーを押します。
   - 空白のトランザクション画面に、`CRED` と入力します。
3. **「Data File Editor Menu」が表示されるので、選択フィールド (S) を使用して、編集するファイルを選択します。**

   この画面のすべてのフィールドとファンクションキーについては、91 ページの「レコードエディタの機能へのアクセス」を参照してください。

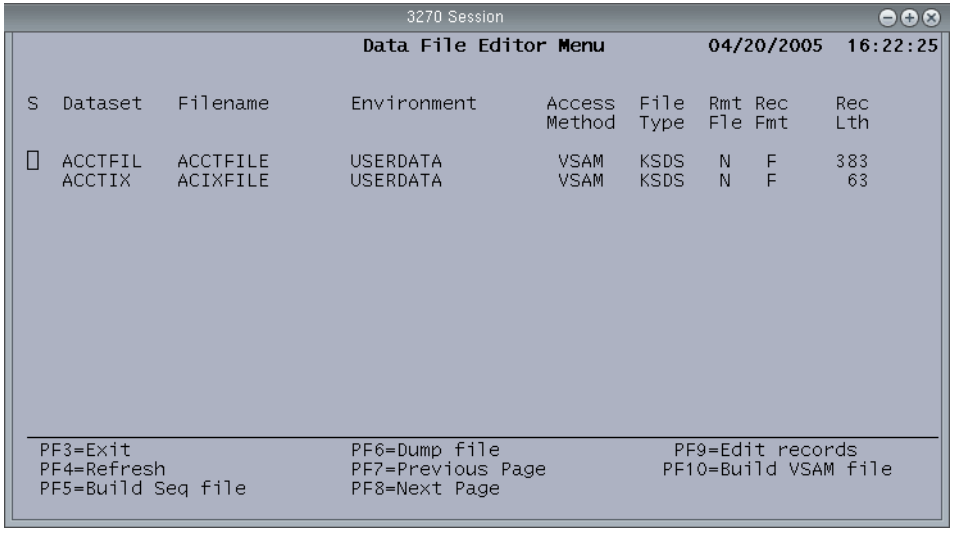


**図 5-1** 「Data File Editor Menu」画面

4. **PF9 キーを押し、図 5-2 に示す「Record Editor」画面を表示します。**

   矢印が指している「Dsn」フィールドは、編集中のデータセットの名前を示します。「Dsn」フィールドの右側のフィールドは、データセットのタイプ (KSDS、ESDS、または RRDS) を示すフィールドです。また、この画面には、レコード長 (Rcd lth) に加え、KSDS データセットの場合、キーオフセット (key ofs) およびキー長 (key lth) が表示されます。

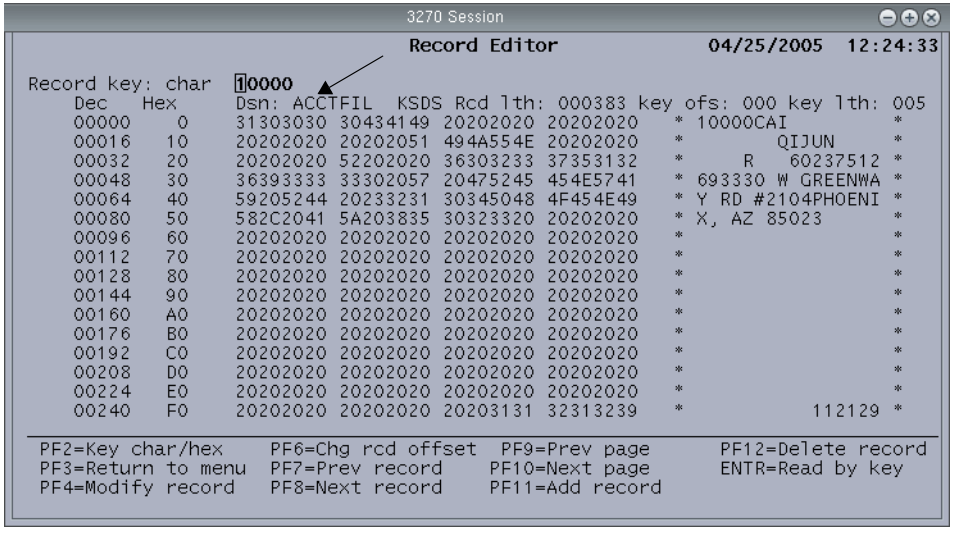


**図 5-2** 「Record Editor」画面

「Record Editor」メイン画面では、次のキーが利用できます。

| ファンクションキー | アクション |
|---|---|
| PF2 | 「Record key」フィールドの表示形式を、文字から 16 進へまたはその逆へと変更します。KSDS データセットのみに有効です。<br>図 5-2 のように「Record key」が「char」になっている場合、フィールドに入力したデータは文字として解釈されます。同様に、図 5-3 のように表示が「hex」の場合は、データを 16 進数値で入力する必要があります。 |
| PF3 | 「Data File Editor Menu」に戻ります。 |
| PF4 | 変更画面が表示され、レコードデータが変更できます。詳細は、136 ページの「データセット内のレコードの変更」を参照してください。 |
| PF6 | 表示する領域のレコードへのバイトオフセットを変更します。10 進数の列の最初のフィールドの保護が解除されます。オフセットを入力して Enter キーを押すと、オフセットが変更されます。PF3 キーを押すとオフセットを変更せずに「Record Editor」画面に戻ります。 |
| PF7 | データセット内の前のレコードを表示します。 |
| PF8 | データセット内の次のレコードを表示します。 |
| PF9 | レコードデータの前の 256 バイトを表示します。 |
| PF10 | レコードデータの次の 256 バイトを表示します。 |
| PF11 | データセットにレコードを追加します。詳細は、139 ページの「データセットへのレコードの追加」を参照してください。 |
| PF12 | 現在のレコードを削除します。詳細は、141 ページの「データセットからのレコードの削除」を参照してください。 |
| Enter | 「Record key」フィールドのキーを使用して、データセットレコードを読み取ります。KSDS データセットの場合は、文字キーか 16 進キーかを指定できます。ESDS データセットの場合はオフセットを指定し、RRDS データセットの場合は、レコード番号を指定します。要求したレコードが見つからない場合、エディタはファイル内で、指定したキーより大きいキーを持つ次のレコードを表示します。 |

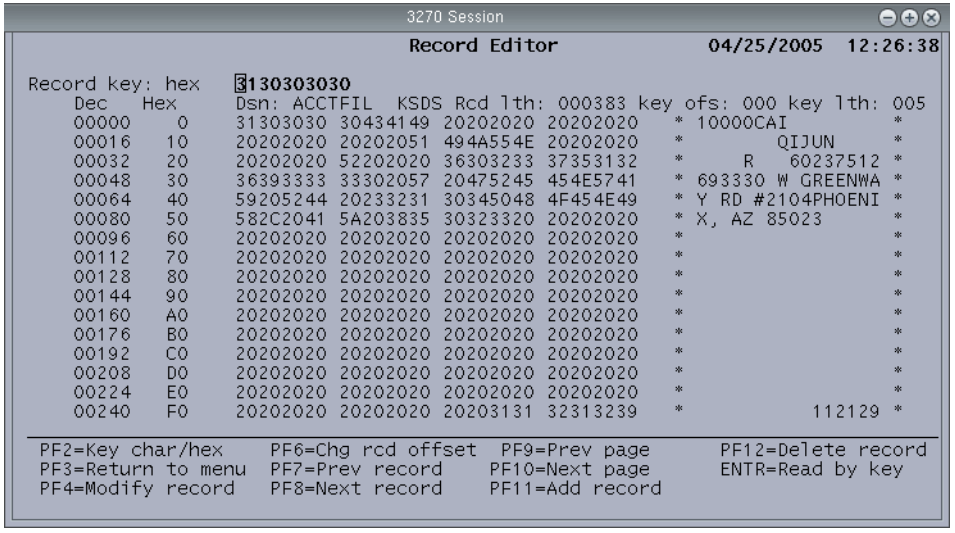


**図 5-3** Record Editor—Hex モード

# データセット内のレコードの変更

変更画面は、「Record Editor」メイン画面で PF4 キーを押すと表示されます。画面の文字領域、つまりアスタリスクで区切られた右側の画面領域または左側の 16 進領域は、それぞれ PF4 または PF5 キーを押して変更できます。編集可能な画面領域が強調表示され、保護が解除されます。

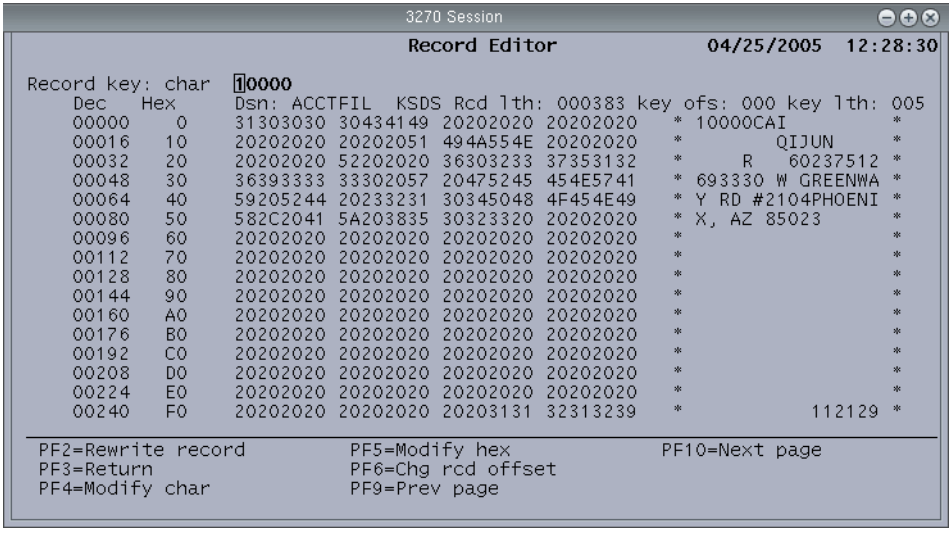


**図 5-4** Record Editor—変更画面

変更画面では、次のファンクションキーが利用できます。

| ファンクションキー | アクション |
|---|---|
| PF2 | 画面で入力したデータでレコードを書き換えます。 |
| PF3 | データを変更せずに「Record Editor」画面に戻ります。 |
| PF4 | 表示するレコードの文字セクションを変更します。アスタリスクで区切られた文字表示領域が強調表示され、この領域が保護されていないことが示されます。このファイルのキーを持つレコードの領域は変更できません。変更しようとするとエラーメッセージが表示されます。図 5-5 の文字モード変更画面を参照してください。<br>文字モード変更画面で Enter キーを押すと、行なった変更内容を保持したままレコードエディタの変更画面に戻ります。<br>PF3 キーを押すと、変更内容を保持せずに変更画面に戻ります。<br>PF2 キーを押すと、変更内容がディスクに書き込まれます。 |

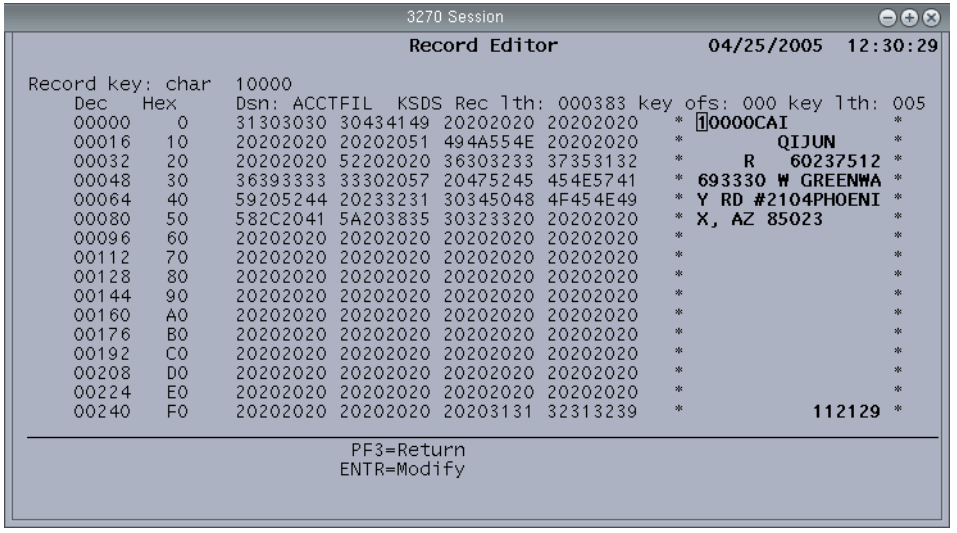


**図 5-5** Record Editor—文字モード変更画面

| ファンクションキー | アクション |
|---|---|
| PF5 | 表示するレコードの 16 進セクションを変更します。16 進領域が強調表示され、この領域が保護されていないことが示されます。図 5-6 の Hex モード変更画面を参照してください。<br>Hex モード変更画面で Enter キーを押すと、変更内容を保持したままレコードエディタの変更画面に戻ります。<br>PF3 キーを押すと、変更内容を保持せずに変更画面に戻ります。<br>PF2 キーを押すと、変更内容がディスクに書き込まれます。 |

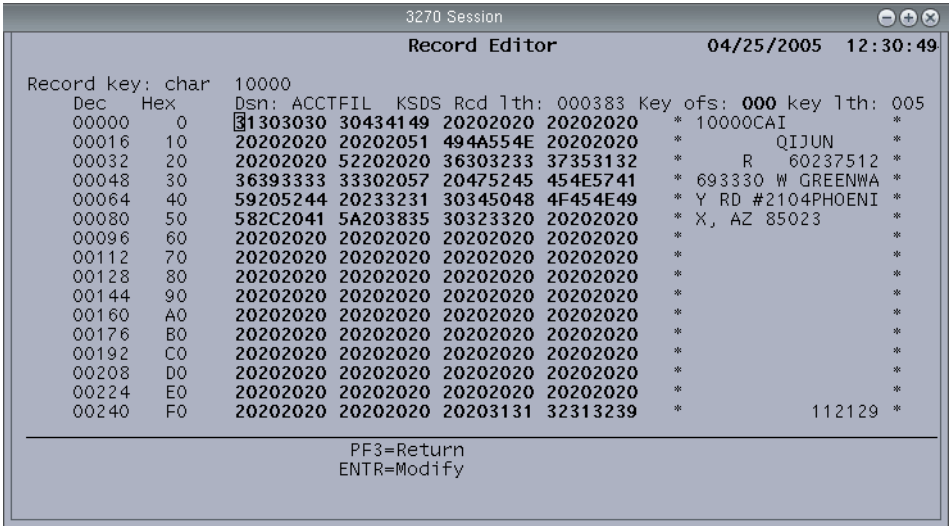


**図 5-6** Record Editor—Hex モード変更画面

| ファンクションキー | アクション |
|---|---|
| PF6 | 表示する領域のレコードへのバイトオフセットを変更します。10 進数の列の最初のフィールドの保護が解除されます。<br>オフセットを入力したのち、Enter キーを押してオフセットを変更します。または PF3 キーを押して、オフセットを変更せずに変更画面に戻ります。 |
| PF9 | レコードデータの前の 256 バイトを表示します。 |
| PF10 | レコードデータの次の 256 バイトを表示します。 |

# データセットへのレコードの追加

レコードをデータセットに追加するには、「Record Editor」画面で PF11 キーを押します。レコードの追加画面は、変更画面と外観や動作が似ています。ただし、レコードを追加したときはデータが確認されます。レコードの追加画面が表示されたら、新しいデータを入力します。画面の文字領域または 16 進領域にレコードを追加します。文字/16 進領域の選択には、PF4/PF5 キーを使用します。

KSDS データセットでは重複キーは使用できないので、キーが一意になるように編集する必要があります。図の右側に表示されている矢印はキーを示します。Enter キーを押すと、左側の矢印で示されているレコードキーが新しいキーに変更されます。

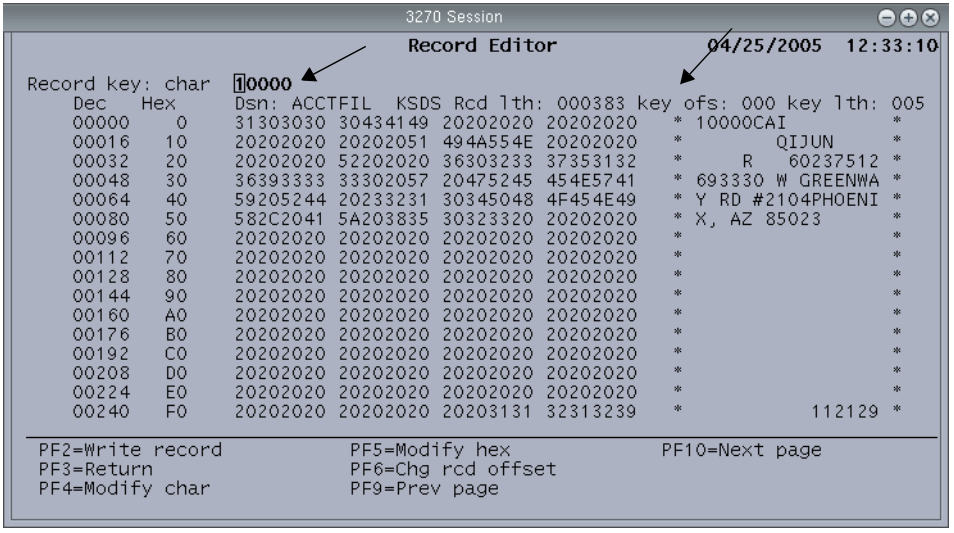


**図 5-7** Record Editor—レコード追加画面

レコードの追加画面では、次のキーが利用できます。

| ファンクションキー | 説明 |
|---|---|
| PF2 | レコードをデータセットに書き込みます。レコードがすでに存在する場合は、エラーが返されます。<br>ESDS データセットの場合は、レコードは末尾に追加されます。 |
| PF3 | 「Record Editor」画面に戻ります。 |
| PF4 | 変更画面を表示します。これによって、レコードデータが文字形式で変更できるようになります。キーシーケンスデータセットのキーも変更できます。それ以外の処理は前の節の説明と同様です。 |
| PF5 | PF4 と同じです。ただし、16 進形式でレコードデータを変更できます。 |
| PF6 | 表示する領域のレコードへのバイトオフセットを変更します。10 進数の列の最初のフィールドの保護が解除されます。<br>オフセットを入力し、Enter キーを押してオフセットを変更します。<br>または PF3 キーを押して、オフセットを変更せずにレコードの追加画面に戻ります。 |
| PF9 | レコードデータの前の 256 バイトを表示します。 |
| PF10 | レコードデータの次の 256 バイトを表示します。 |

# データセットからのレコードの削除

## ▼ レコードを削除する

1. 「Record Editor」メイン画面で PF12 キーを押し、レコードの削除画面を表示します。

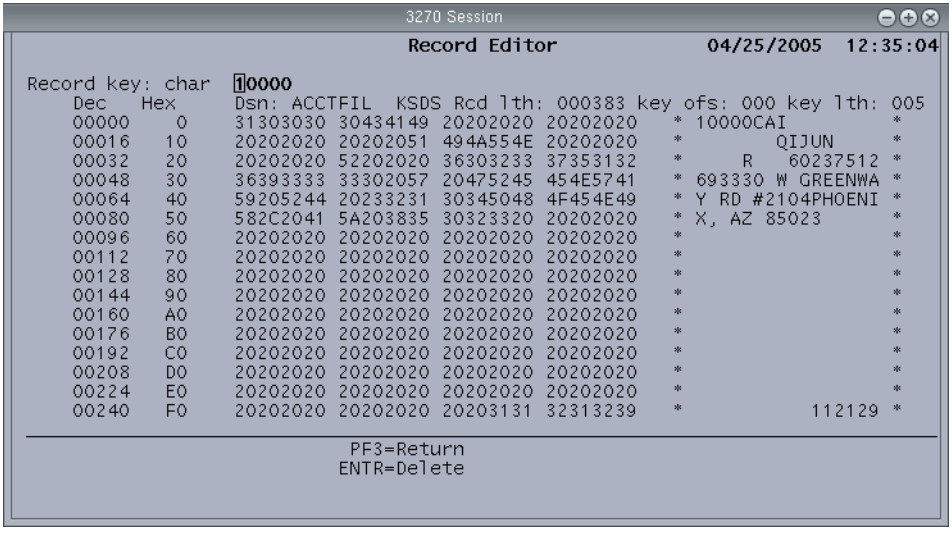


図 5-8　Record Editor—レコード削除画面

2. Enter キーを押して、表示されているレコードを削除します。

   PF3 キーを押すと、レコードを削除せずに「Record Editor」メイン画面に戻ります。

第6章

# システム間通信 (ISC) の使用

『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』では、システム間通信 (ISC) の設定方法を説明しています。この章では、サポートされる ISC の機能とその機能を実装するための構成オプションについて説明します。次のトピックについて説明します。



**注 –** TCP/IP では、2 フェーズコミットがサポートされています。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。

## トランザクション経路指定

トランザクション経路指定では、1 つのシステムに接続されている端末が、別の Sun MTP または CICS システム上のトランザクションを実行できます。システム初期化テーブル (SIT) には、ローカルシステム識別子 (SysId) が含まれています。この識別子とプログラム管理テーブル (PCT) を使用して、トランザクションをローカルに処理するか、遠隔システム上で処理するかを判定します。

また、トランザクション経路指定では、`START` コマンドまたは一時データトリガーレベルを使用し、自動トランザクション開始 (ATI) によって起動されたトランザクションが、ほかの領域に所有される端末を取得できます。`CRTE` トランザクションを

使用すれば、別の Sun MTP または CICS 領域へトランザクションを経路指定できるほか、CRTE によって経路指定されたトランザクションを受け付けることも可能です。

Sun MTP で使用可能なトランザクション経路指定機能は、CICS での経路指定機能のサブセットです。次の機能はサポートされていません。

- LU2 3270 ディスプレイと LU3 3270 プリンタ以外の端末装置
- BMS 経路指定コマンド
- LDC コマンド
- 端末が所有する領域内で行われる BMS ページング
- メッセージ保全性
- 遠隔 EDF
- Local または Identify 以外のセキュリティー

端末管理テーブル (TCT) ユーザー領域と通信領域は、ディスプレイ使用データが入るものと想定されます。これらのデータ領域は、Sun MTP 領域と CICS の間で交換できるよう、必要に応じて ASCII と EBCDIC のコード変換が行われます。これらの領域のデータに変換の必要がない場合、またはデータがディスプレイを使用しない場合のために、ユーザー出口プログラムが用意されています。このユーザー出口プログラムの詳細については、265 ページの「トランザクション経路指定のための変換ルーチン」を参照してください。

## CRTE を使用したトランザクションの経路指定

CRTE は、トランザクションを遠隔システムに経路指定します。トランザクションの実行が頻繁でない場合、またはトランザクションが多くのシステム上に、同じ名前 (たとえば、CEMT) で存在する必要がある場合は、CRTE を使用します。

形式:

```
CRTE SYSID=sysid
```

*sysid* は、TCT のシステムエントリテーブルでローカルシステムについて定義されている遠隔システム識別子です。

コマンドが正しく実行されると、その端末で入力されたトランザクションはすべて、遠隔システムに経路指定されます。トランザクションの経路指定を中止するには、CANCEL コマンドを使用します。これにより、経路指定が中止されたというメッセージが、CRTE によって表示されます。接続が切断された場合または遠隔システムが終了された場合にエラーが発生したときは、CRTE セッションは自動的に取り消されます。

CANCEL は、1 つの CRTE セッションだけを終了します。したがって、CRTE を使用してトランザクションを多くのシステムに経路指定している場合は、システムごとに CANCEL を入力します。

CRTE セッションにある間は、端末は ATI 要求を受信できません。このようなトランザクションは、CRTE セッションの終了後に端末上で開始されます。

## アウトバウンドトランザクション経路指定の定義

ローカル端末定義については、別途構成する必要はありません。デフォルトで、すべてのローカル端末はシップ可能に定義されています。

## ▼ トランザクションを遠隔に定義する

1. 空白のトランザクション画面に CTBL を入力し、Table Manager を開きます。
2. PF4 – 標準テーブルを押してから、PF6 – プログラム管理テーブルを押します。
3. PF4 キーを押して、挿入画面を表示します。

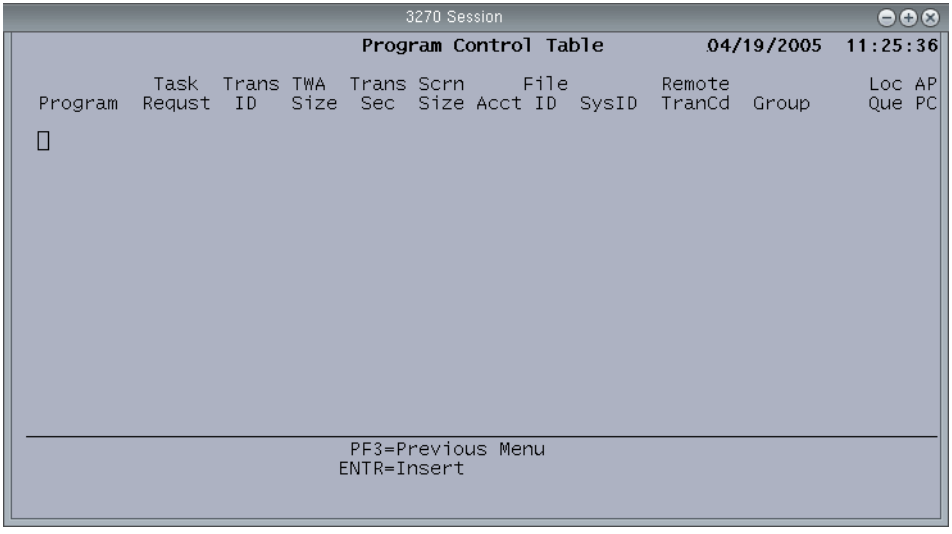


図 6-1　PCT—遠隔トランザクションの定義

4. 次の情報を入力して、遠隔トランザクションを定義します。
   a. 「Trans ID」フィールドで、ローカルシステムのトランザクション名を入力します。
   b. 「SysID」フィールドで、遠隔システムのローカル識別子を入力します。
      この SysID は、TCT のシステムエントリテーブルで定義しておく必要があります。

c. 「Remote Trans ID」フィールドで、遠隔システムのトランザクション名を入力します。

5. Enter キーを押してデータを挿入し、PCT 画面に戻ります。
6. PF2 キーを押してテーブルを保存します。
7. PF3 キーを押して、「Standard Tables」メニューに戻ります。
8. PF1 キーを押して、SIT を表示します。
9. SIT 画面で、アプリケーション名を指定する必要があります。

   アプリケーション名は、ローカル領域のネットワーク名です。このフィールドには、ほかの Sun MTP または CICS 領域がこの領域を認識するために使用しているネットワーク名と同じ値を指定します。CICS システムでのネットワーク名を確認するには、CICS が要求を Sun MTP 領域に経路指定するための接続定義を確認します。ネットワーク名は、Netname フィールドに表示されています。

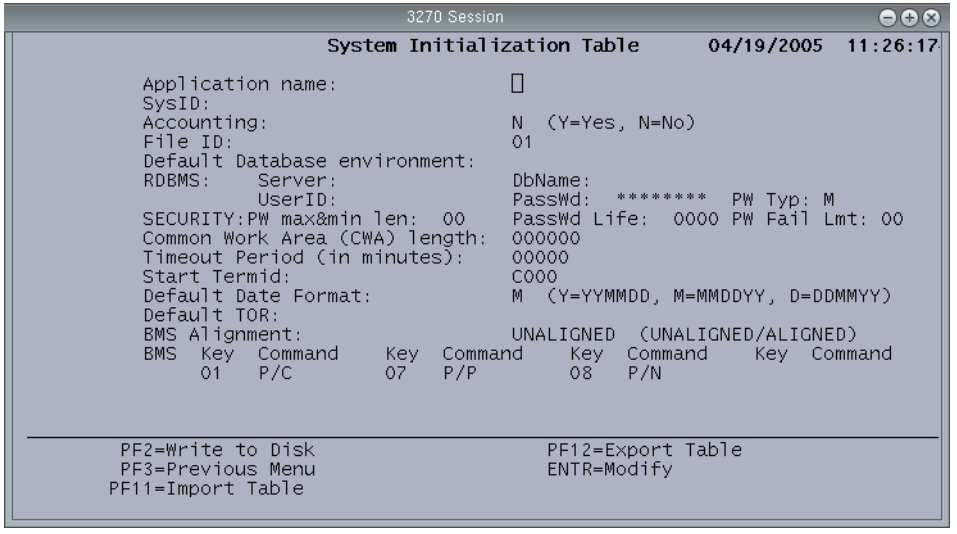


図 6-2　SIT—遠隔トランザクションの定義

10. 操作が終了したら、PF2 キーを押してテーブルを保存します。
11. PF3 キーを 2 回押して、「Table Manager」メニューに戻ります。
12. これらの変更内容を有効にするには、領域を停止して再起動する必要があります。

# インバウンドトランザクション経路指定の定義

トランザクションを遠隔システムから実行するには、要求を行う端末の定義が領域上に必要です。これには、すべての端末が、シップ可能の TYPETERM を使用して、領域にトランザクションを経路指定するように定義する方法が推奨されます。

端末をシップ可能に定義できない場合、または領域への経路指定を行う遠隔システムに、端末 ID が同じである端末が 2 つ存在する場合、端末の遠隔定義を TCT に設定する必要があります。

## ▼ 端末を遠隔に定義する

1. 空白のトランザクション画面に CTBL を入力し、Table Manager を開きます。
2. PF4 – 標準テーブルを押してから、PF8 – 端末管理テーブルを押します。
3. TCT メニューで PF9 – 3270 デバイスを押します。
4. PF4 – Insert Entry を押し、次のフィールドを使用して、挿入画面で遠隔端末を定義します。
   a. 「Term ID」フィールドで、端末に割り当てられている名前を入力します。
   b. 「SysID」フィールドで、Sun MTP にトランザクションを経路指定するときに使用する遠隔システムのローカル識別子を入力します。
      この SysID は、TCT のシステムエントリテーブルで定義しておく必要があります。
   c. 「Rem Name」フィールドで、遠隔システムの端末の端末 ID を入力します。

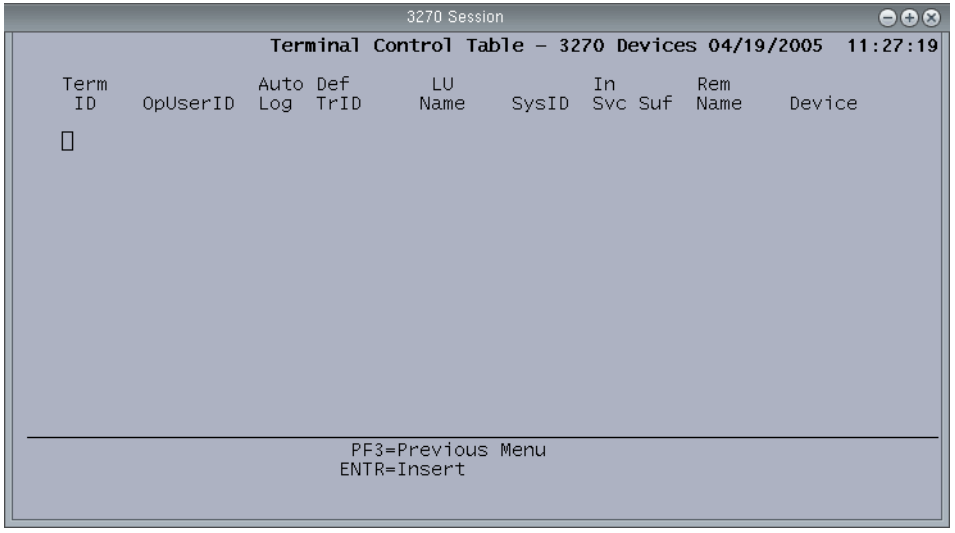


図 6-3　　TCT—3270 Devices からのトランザクションの定義

5. Enter キーを押してデータを挿入し、「3270 Devices」画面に戻ります。
6. 「Host & Port」画面で情報が必要な場合は、PF9 キーを押して情報を指定してから、Enter キーを押して 「3270 Devices」メイン画面に戻ります。
7. PF3 キーを押して、TCT メニュー画面に戻ります。
8. PF2 キーを押してテーブルを保存します。
9. PF3 キーを押して、「Standard Tables」メニューに戻ります。
10. 領域を停止して再起動し、変更を有効にします。

インバウンドトランザクション経路指定を実行すると、領域にすでに設置されている端末のリストに照らして、端末 ID がチェックされます。端末 ID が設置されている端末に一致した場合は、経路指定要求が拒否されます。場合によっては、端末 ID が Sun MTP のデフォルト端末名に一致する場合があります。このようなデフォルト端末名は、SIT のエントリによって変更が可能です。

1. Table Manager – 標準テーブルメニューで PF1 キーを押して、SIT を開きます。
2. SIT 画面で、遠隔システムの端末 ID と異なる名前を「Start Termid」で指定します。

「Start Termid」は、デフォルトの端末 ID 名です。後続のすべての端末名は、それらが一意になるように、値 1 が加算されていきます。

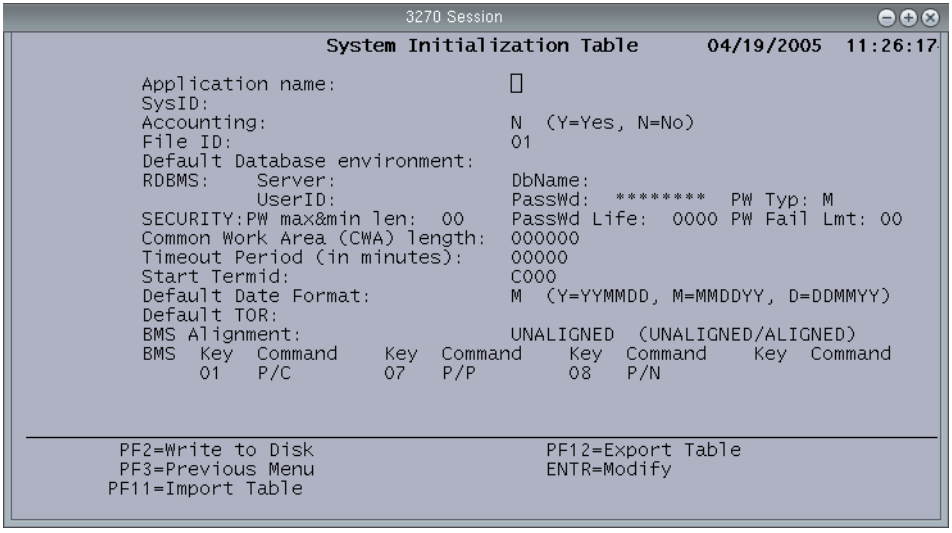


図 6-4　　SIT—遠隔端末の定義

3. 操作が終了したら、PF2 キーを押してテーブルを保存します。
4. PF3 キーを 2 回押して、「Table Manager」メニューに戻ります。
5. これらの変更内容を有効にするには、領域を停止して再起動する必要があります。

# 機能シップ

機能シップを使用すると、アプリケーションプログラムが、ほかの領域の所有するリソースにアクセスしたり、更新したりできるので、複数のシステムによるリソースの分散や共有が可能になります。このようなリソースには、VSAM ファイル、一時データ、補助一時記憶域データなどがあります。

このソフトウェアは、必要に応じて、ASCII と EBCDIC の間のコード変換機能を提供します。

Sun MTP で使用可能な機能シップは、CICS で利用可能なサブセットです。次の機能はサポートされていません。

- IMS データベース
- DL/I データベース

# アウトバウンド機能シップの定義

機能シップは、次のどちらかの場合に実行されます。

- EXEC CICS コマンドの SYSID オプションに有効な接続名を指定する。
- リソースが適切なテーブルで、遠隔として定義されている。

## ▼ 遠隔ファイルを定義する

---

**注 –** 遠隔に定義されているデータセットはエイリアスとみなされません。

---

1. **FCT を開き、データセットを選択します。**
2. **PF9 キーを押し、図 6-5 に示す「Remote File Characteristics」画面を表示します。**

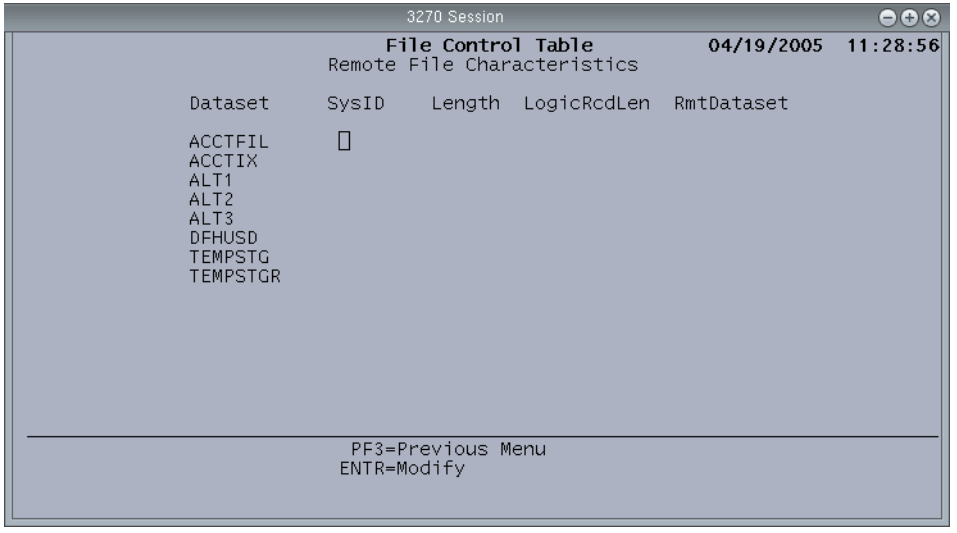


**図 6-5** FCT—Remote File Characteristics の定義

3. **次のフィールドに情報を指定します。**

   a. **「SysID」フィールドで、遠隔システムのローカル識別子を入力します。**

   この SysID は、TCT のシステムエントリテーブルで定義しておく必要があります。

b. 「Length」フィールドの値は、定義する VSAM ファイルのタイプによって決まります。

KSDS ファイルの場合、ファイルのキー長をバイトで入力します。これは、ローカルに定義されているシステムのファイルのキーと一致している必要があります。

ESDS と RRDS ファイルでは、0 を指定します。

c. 「LogicRcdLen」フィールドで、最大レコード長をバイトで入力します。

これは、ファイルを所有するシステムのファイルの長さと一致する必要があります。

d. 「RmtDataset」フィールドで、遠隔システムのファイル名を入力します。

## ▼ 遠隔一時データキューを定義する

1. 「DCT–Remote Destinations」画面を開きます。
2. PF4 キーを押して、挿入画面を表示します。

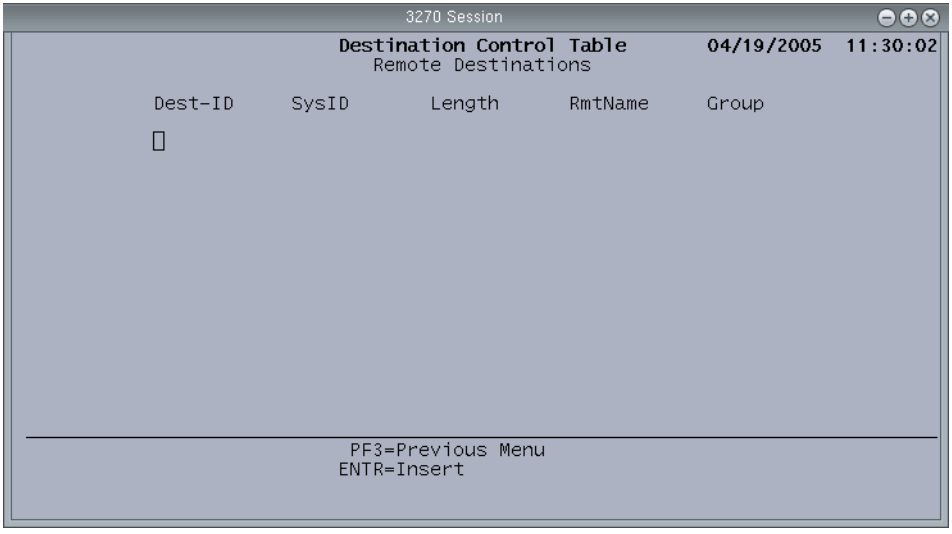


図 6-6　DCT—遠隔一時データキューの定義

3. 次のフィールドに情報を指定します。

a. 「Dest-ID」フィールドで、ローカルシステムのキューの名前を入力します。

b. 「SysID」フィールドで、遠隔システムのローカル識別子を入力します。

この SysID は、TCT のシステムエントリテーブルで定義しておく必要があります。

c. 「Length」フィールドで、最大レコード長をバイトで入力します。

これは、ファイルを所有するシステムのキューの長さと一致する必要があります。

d. 「RmtName」フィールドで、遠隔システムのキューの名前を入力します。

4. Enter キーを押して、情報を挿入します。
5. DCT メニューで、PF2 キーを押してテーブルを保存します。

## ▼ 遠隔一時記憶域キューを定義する

1. TST を開きます。
2. PF4 キーを押し、エントリを挿入します。

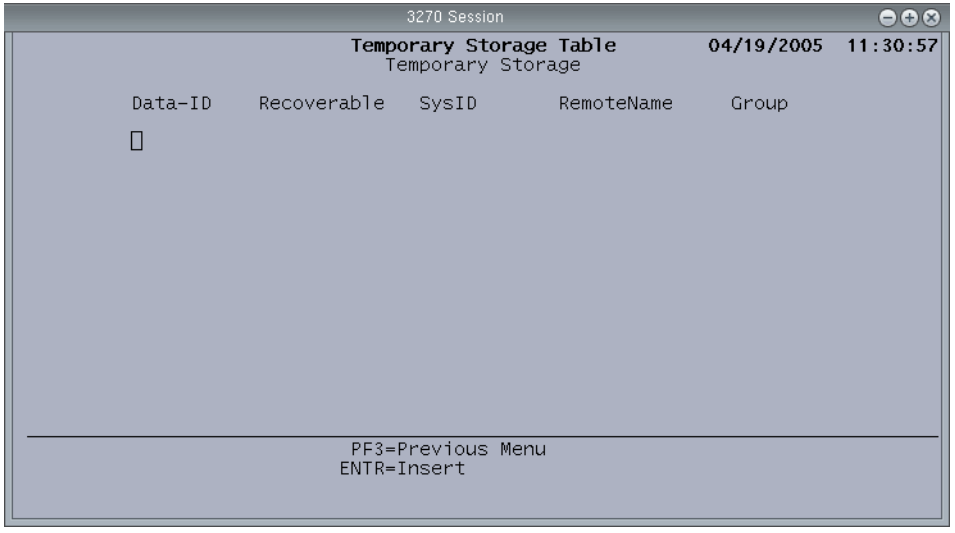


図 6-7　TST—遠隔一時記憶域キューの定義

3. 次の情報を入力します。

a. 「Data-ID」フィールドで、ローカルシステム上の一時記憶域キューの総称名または全体名を入力します。

b. 「SysID」フィールドで、遠隔システムのローカル識別子を入力します。

この SysID は、TCT のシステムエントリテーブルで定義しておく必要があります。

c. 「RemoteName」フィールドで、遠隔システム上の一時記憶域キューの総称名または全体名を入力します。

4. Enter キーを押して、エントリを挿入します。

5. TST のメイン画面で、PF2 キーを押してテーブルを保存します。

## インバウンド機能シップの定義

機能シップのインバウンドに、特別な定義は不要です。

## 機能シップ中のデータ変換

Sun MTP 領域と EBCDIC システム間で、データ変換が必要になることがあります。ほとんどのデータでは、変換が自動的に行われます。しかし、ユーザーデータがシステム間で渡される場合、Sun MTP 領域はそのユーザーデータの形式を知る必要があります。

画面方式のテーブルジェネレータによって、ユーザーはこれを使用して VSAM ファイルのレコードレイアウトを定義できます。生成されたテーブルは、Sun MTP と CICS 間で伝送されるレコードを、機能シップ、非同期処理、および DPL が変換する際に使用されます。このジェネレータはデータ変換テンプレートテーブル (CVT) です。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。

---

**注 –** EBCDIC システムに対して、または EBCDIC システムからファイル要求をシップする場合、ファイルのレコードがすべて文字であっても、ファイルには CVT エントリが必要です。ファイルのレコードがすべて文字である場合、キーの形式を指定するキーレコードと、データレコード全体を指定するデフォルトのレコードの両方が必要です。

---

機能シップの会話は、CICS `CVMI` または `CPMI` ミラートランザクションを操作する LU6.2 確認レベルセッションによって行われます。

| | |
|---|---|
| `CVMI` | TCT のシステムエントリテーブルの「Ptnr CdPge」フィールドが、特定の接続について空白でない場合に使用します。これは、遠隔システムが EBCDIC システムであり、データ変換が必要なことを指定します。 |
| `CPMI` | 「Ptnr CdPge」フィールドが空白の場合、つまり、遠隔システムが ASCII システムであり、データ変換の必要がない場合に使用します。 |

# 非同期処理

非同期処理では、Sun MTP または CICS トランザクションから遠隔システム上のトランザクションを起動したり、データをそれに渡したりすることによって、複数のシステム間での処理を分散します。この形態の ISC では、2 つのトランザクション間で要求と応答に直接の相関関係がないので、Sun MTP はこれを非同期と考えます。

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。このソフトウェアは、双方向の非同期処理をサポートします。

## アウトバウンド非同期処理の定義

非同期処理は、次のどちらかの場合に実行できます。

- `EXEC CICS` コマンドの `SYSID` オプションに有効な接続名を指定する。
- PCT でトランザクションが遠隔と定義されている。

`INTERVAL` がコマンドで指定されている場合、要求は即座に遠隔システムにシップされ、指定の間隔が経過するまでキューに入れられます。

遠隔システムに対する `START` コマンドで `NOCHECK` パラメータを指定すると、このコマンドを実行する ISC データフローの数が低減します。ただし、遠隔システムからローカルシステムに対してエラー応答が返されなくなるので、遠隔システムでのエラーに対する回復ができなくなります。

## ▼ 遠隔トランザクションを定義する

1. PCT を開きます。
2. PF4 キーを押して、挿入画面を表示します。

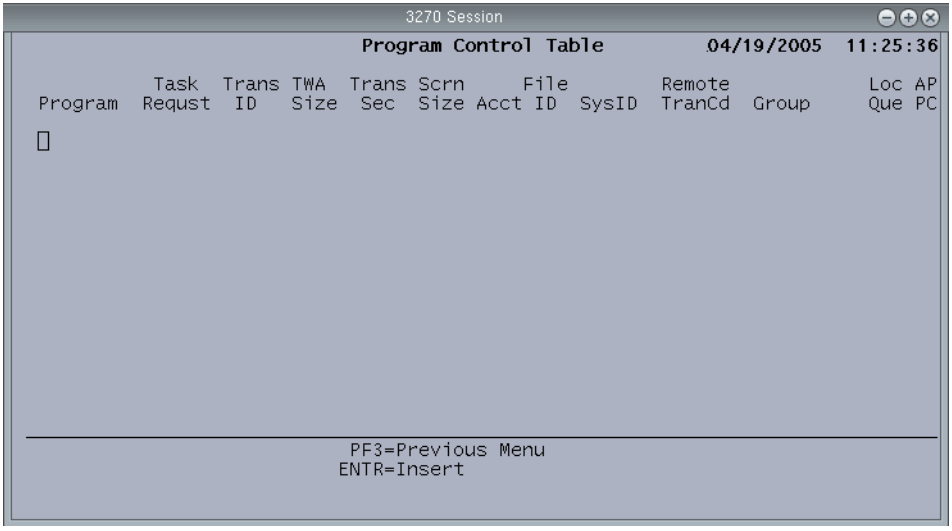


図 6-8　　PCT—遠隔トランザクションの定義

3. 次のフィールドを使用して、遠隔トランザクションを定義します。

   a. 「Trans ID」フィールドで、ローカルシステムのトランザクション名を入力します。

   b. 「SysID」フィールドで、遠隔システムのローカル識別子を入力します。

      この SysID は、TCT のシステムエントリ画面で定義しておく必要があります。

   c. 「Remote TranCd」フィールドで、遠隔システムのトランザクション名を入力します。

   d. 「Local Queue」フィールドで、要求をシップするためのセッションがすぐに手配できない場合、要求をローカルのキューに入れるかどうかを指定します。

      Y: 要求をローカルのキューに入れます。要求をローカルのキューに入れる場合は、START コマンドで NOCHECK パラメータを使用する必要があります。

      N: 要求をローカルのキューに入れません。

      要求を遠隔システムに送信する際に遅延が生じると、トランザクションの実際の起動時間に影響が出るので、時間依存の要求がローカルのキューに入れられないように設定する必要があります。

4. Enter キーを押して、テーブルにエントリを追加します。

5. PCT のメイン画面で、PF2 キーを押してテーブルを保存します。

## インバウンド非同期処理の定義

非同期処理のインバウンドに、特別な定義は不要です。

# 分散プログラムリンク (DPL)

DPL を使用すると、異なる Sun MTP または CICS 領域にあるプログラム間で同期リンクが形成できます。また DPL によって、Sun MTP 領域は、BDAM、DB2 ファイル、および IMS、DL/I、および SQL データベースにアクセスできるメインフレームプログラムにリンクできます。このソフトウェアの DPL 機能は双方向です。

---

**注 –** この双方向機能は、メインフレームリリース CICS/ESA 3.3 以降および CICS/VSE 2.2 以降だけでサポートされますが、それより前のバージョンでも DPL 要求の受信は可能です。

---

SYNCONRETURN パラメータを LINK コマンドに指定した場合、リンク先プログラムはそのリソースをコミットしたのち、リンク元のプログラムに制御を返します。リンク先プログラムは、それが SYNCONRETURN 指定で開始されたかどうかを判定できます。

ASSIGN コマンドで STARTCODE パラメータを使用すると、プログラムが SYNCONRETURN なしで起動された場合は D が、SYNCONRETURN を指定して起動された場合は DS が返されます。

リンク先プログラムは LINK 要求を発した端末にはアクセスできません。したがって、端末制御と BMS CICS コマンドはリンク先プログラムからは利用できず、これらのコマンドを実行すると、INVREQ コードが返されます。

1 つのテストシステム上で DPL のリンク元プログラムとリンク先プログラムをテストするには、リンク先のプログラムが DPL 制限付き API を実行できないように、リンク先プログラムの定義で APISet=D を指定します。

COMMAREA パラメータは、システム間でデータをシップするために使用します。プログラムで大きな COMMAREA を使用しているが、リンク先プログラムに対して COMMAREA の先頭の小さな領域しか送る必要がない場合は、LINK で DATALENGTH パラメータを使用することによって伝送時間を短縮できます。送信するデータの長さを DATALENGTH に入力します。リンク先のプログラムは、COMMAREA 全体を返します。

## アウトバウンド DPL の定義

DPL は、次のいずれかの方法で実行します。

- EXEC CICS コマンドの SYSID オプションに有効な接続名を指定する。
- PPT でエントリを追加して、プログラムを遠隔に定義する。

## ▼ 遠隔プログラムを定義する

1. PPT を開きます。
2. PF4 キーを押して、挿入画面を表示します。

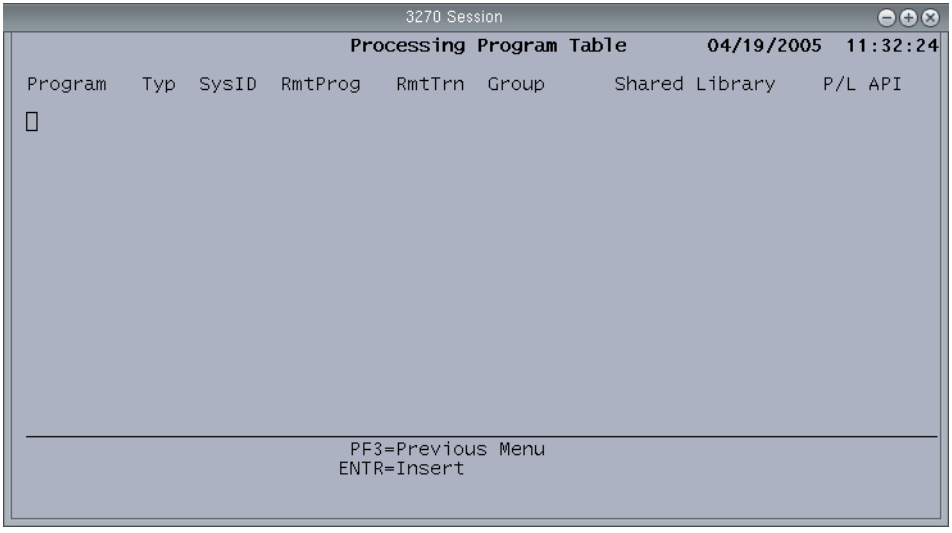


図 6-9　PPT—遠隔プログラムの定義

3. 次のフィールドを使用して、遠隔プログラムを定義します。
   a. 「Program」フィールドで、ローカルシステムのプログラム名を入力します。
   b. 「SysID」フィールドで、遠隔システムのローカル識別子を入力します。
      この SysID は、TCT のシステムエントリテーブルで定義しておく必要があります。
   c. 「RmtProg」フィールドで、遠隔システムのプログラム名を入力します。

d. 「RmtTrn」フィールドで、遠隔システムでプログラムが実行するトランザクション名を入力します。

空白の場合、プログラムは標準のミラートランザクションの下で実行されます。このフィールドは、アカウンティングまたはセキュリティー用に使用できます。

4. Enter キーを押して、テーブルにエントリを追加します。
5. PPT のメイン画面で、PF2 キーを押してテーブルを保存します。
6. Table Manager を終了します。

## インバウンド DPL の定義

リンク先プログラムに対して、PPT にエントリを追加します。LINK コマンドに TRANSID を指定している場合、または「RmtTrn」フィールドが PPT に設定されている場合は、指定されたトランザクションが遠隔システムで定義され、実行しているプログラムはそのシステムのミラープログラムでなければなりません。そのためには、PCT にエントリが必要です。

## ▼ インバウンド DPL を定義する

1. PPT を開きます。
2. PF4 キーを押して、挿入画面を表示します。

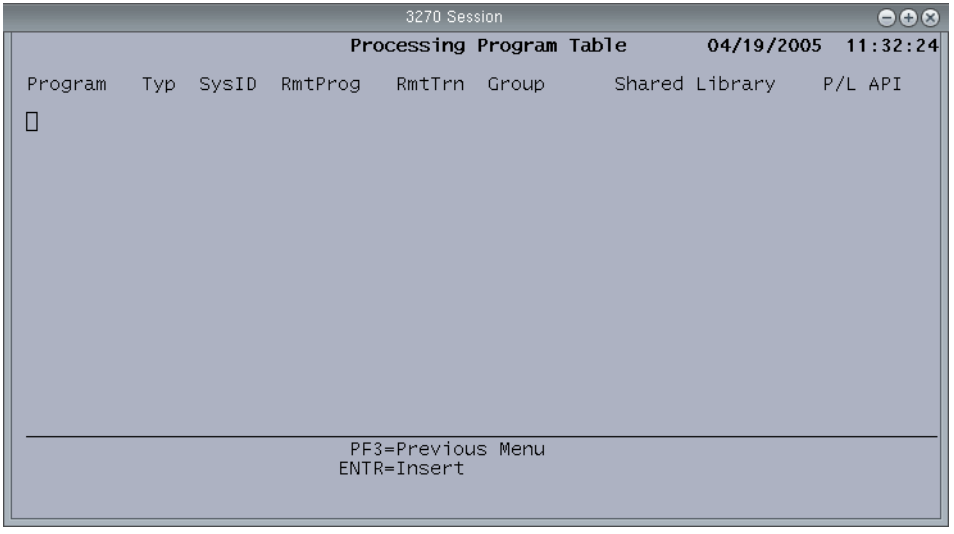


図 6-10 PPT—遠隔プログラムの定義

3. 次のように遠隔プログラムを定義します。

   a. 「Program」フィールドで、ローカルシステムのプログラム名を入力します。

   b. 「SysID」フィールドで、遠隔システムのローカル識別子を入力します。

      この SysID は、TCT のシステムエントリテーブルで定義しておく必要があります。

   c. 「RmtProg」フィールドで、遠隔システムのプログラム名を入力します。

   d. 「RmtTrn」フィールドで、遠隔システムでプログラムが実行するトランザクション名を入力します。

      空白の場合、プログラムは標準のミラートランザクションの下で実行されます。このフィールドは、アカウンティングまたはセキュリティー用に使用できます。

   e. 「API」フィールドで、次の値のいずれかを指定します。

      D: リンク先プログラムは DPL 制限付き API コマンドのみ実行できます。

      F: (デフォルト) リンク元プログラムと同じ領域で稼動しているリンク先プログラムであれば、完全なコマンドセットを実行できます。

4. Enter キーを押して、テーブルにエントリを追加します。
5. PPT のメイン画面で、PF2 キーを押してテーブルを保存します。
6. PF3 キーを押して「Standard Tables」メニューに戻ります。
7. PCT でプログラムを定義する必要がある場合は、PF6 キーを押して PCT を表示します。
8. PF4 キーを押して、挿入画面を表示します。
9. 次のようにトランザクションを定義します。

   a. 「Program」フィールドで、プログラム名 KXFSMIRS を入力します。

   b. 「TransID」フィールドで、ローカルシステムのトランザクション名を入力します。

   c. 「SysID」フィールドが空白であることを確認します。

10. Enter キーを押して、エントリを挿入します。
11. PCT のメイン画面で、PF2 キーを押してディスクに変更を保存します。
12. Table Manager を終了します。

# 分散トランザクション処理 (DTP)

DTP を使用すると、1 つの領域のトランザクションからほかのトランザクションの起動とそのトランザクションとの会話処理ができます。ほかのトランザクションは、LU6.2 (APPC) プロトコルをサポートしていれば、どのシステムで実行されていてもかまいません。DTP は、次のアプリケーションで有用です。

- 遠隔リソースの遠隔処理が必要なアプリケーション
- システム間でのデータ転送
- システム間でのバッチデータ転送

DTP では、トランザクション間で渡されるメッセージが直接相関関係を持っているので、トランザクション同期通信が実現されます。サポートする DTP CICS コマンドレベル文の詳細については、『Sun Mainframe Transaction Processing ソフトウェア開発者ガイド』を参照してください。

# 遠隔トランザクションのデバッグ

トランザクション経路指定を除くアウトバウンドトランザクションでは、トランザクションを実行している端末に対して、Sun MTP デバッグ機能を使用可能にできます。これにより、`EXEC CICS` コマンドがデバッグできるようになります。

トランザクション経路指定では、トランザクションが遠隔システム上で実行されるので、その遠隔システム上でデバッグ機能を呼び出す必要があります。Sun MTP では、端末ユーザーが `CRTE` トランザクションを使用して、遠隔システムへのトランザクションの経路指定を強制的に実行できます。

## ▼ 遠隔トランザクションをデバッグする

1. `CRTE` **トランザクションを入力して、遠隔システムに接続します。**

```
CRTE SYSID=sysid
```

2. `CEDF` **を入力して、遠隔システムのデバッグを有効にします。**

   デバッグとブレークポイント処理の詳細については、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。

3. **トランザクションを入力します。**

   トランザクションが遠隔システムで実行されます。

## ▼ インバウンドトランザクションをデバッグする

1. **受信側システムで** CEDF **コマンドを入力します。**

   *sysid* は、トランザクションの送信側システムを示します。*termid* を指定する場合は、「注」を参照してください。

```
CEDF [sysid] [termid] ,ON
```

2. **トランザクションが** *sysid* **から受信され、これによって** CEDF **を実行した端末上でデバッグ機能が有効になり、トランザクションをデバッグできます。**

   デバッグ端末がビジーの間、トランザクションのデバッグ機能は、端末が使用できるようになるまでキューで待機します。

   トランザクション経路指定では、*sysid* パラメータを使用して CEDF を起動すると、デバッグセッションをデュアル画面モードにできます。端末定義は不要で、COBOL ソースデバッガが使用できます。

---

**注 –** デバッグセッションを端末識別子 (*termid*) の上で実行する場合は、端末定義が必要になり、デバッガは使用できません。その端末の最初のトランザクションをデバッグする場合は、端末を TCT であらかじめ設定する必要があります。最初のトランザクションののち、TCT に端末を設定した場合、そのあとのトランザクションは、デバッグ機能の下で実行されます。

---

# ISC と 3270 デバイスの同時サポート

Sun MTP の 1 つの領域で ISC と 3270 デバイスの両方をサポートする場合、その領域は IBM システムからは 2 つの遠隔システムとして認識され、それぞれに異なる修飾されたネットワーク名を持ちます。

- 3270 デバイスのサポートは、IBM システムとは別のドメインにあるネットワーク名の 1 つとして認識される
- ISC 領域は、IBM システムと同じドメインにある独立した論理ユニットを持つ PU2.1 ノードとして認識される

IBM システムは、3270 デバイスと ISC 機能が同じシステムで動作しているとは認識しません。

Sun MTP 領域は、異なる 2 つの論理接続によって IBM ネットワークと接続されています。これは通常、2 つの物理接続が必要です。たとえば、Sun MTP は、ドメイン間トラフィックのサポート用と、独立の論理ユニットのサポート用の 2 つの SDLC ラインを使用します。Ethernet 環境では、これらの物理接続が、2 つのサービスアクセスポート (SAP) を持つ 1 つの物理接続になります。

第7章

# セキュリティー

Sun MTP セキュリティーは、組み込みセキュリティー機能とオペレーティングシステムのセキュリティー機構が統合されていて、ユーザーがログインしてからトランザクションを実行し、ユーザーがログオフするまで、セキュリティー保護された環境を保証します。

この章では、次のトピックについて説明します。

- 163 ページの「UNIXセキュリティー」では、UNIX セキュリティーの概要について簡単に説明します。
- 171 ページの「Sun MTPセキュリティー」では、このソフトウェアの組み込みセキュリティー機能について説明します。
- 186 ページの「セキュリティーの例」では、組み込みセキュリティー機能の使用法について説明します。

## UNIXセキュリティー

この節では、あらゆる UNIX 実装に見られる基本的な UNIX セキュリティーと機構を説明します。ご使用のオペレーティングシステムが、より高度なセキュリティー機能によってアプリケーション環境をセキュリティー保護していることもあります。すべての UNIX セキュリティーとシステム管理についての情報が記述された最も信頼できるマニュアルは、システムに付属しているマニュアルです。

# スーパーユーザー

すべての UNIX システムには、システムのすべての部分に無制限のアクセス権を持つユーザーが 1 人います。このユーザーは、「スーパーユーザー」と呼ばれ、ユーザー名は *root* です。このユーザーを指す用語は、スーパーユーザーでも root でもかまいません。root パスワードを持つのは、UNIX のシステム管理者だけに限定します。

# ユーザー

UNIX セキュリティーは、ユーザーという概念に基づいています。すべてのユーザーには、ユーザー名が割り当てられます。システム管理者は、各ユーザーに一意のユーザー名を割り当てる責任があります。

## システムへのログイン

ユーザーは UNIX システムにログインする必要があります。ログインプロセスは、2 段階から成ります。まず、ユーザーはユーザー名を入力し、次にパスワードを入力します。パスワードは監視には表示されません。

## ユーザー名とパスワードの保持

ユーザー名、パスワード、およびその他のアカウント情報は、パスワードデータベースで保持されます。ほとんどのシステムで、システム管理者はファイルをテキストエディタで編集するのではなく、管理ツールを使用して、パスワードデータベースを管理します。パスワードデータベースは、通常、`/etc/password` ファイルに格納され、ここにユーザーごとのエントリが含まれます。エントリは、コロンで区切られたフィールドのリストで構成されます。一般的な形式は次のとおりです。

*username* : *password* : *userid* : *groupid* : *identification* : *homedirectory* : *loginshell*

| 要素 | 説明 |
|---|---|
| *username* | ユーザーの一意のログイン名。ログイン名は、通常 8 文字以下です。 |
| *password* | ユーザーのパスワード。システム管理者は、`passwd` コマンドでパスワードを設定する必要があります。<br>`passwd` コマンドによって、スーパーユーザーはすべてのパスワードを変更でき、ユーザーは自分のパスワードだけを変更できます。 |
| *userid* | ユーザー名の数値識別子。通常、ドキュメントには記述されていない場合でも、UNIX システムの各値に対する最大しきい値があります。安全なガイドラインとして、ユーザー ID を 16K (16384) 未満に維持するとよいでしょう。UNIX システムにはすべて、パスワードファイルにデフォルトのユーザーのセットが用意されています。これらのユーザーのユーザー ID は、通常は 100 未満です。一般的なガイドラインとして、ユーザーのユーザー ID を 100 から開始し、そこから増分するとよいでしょう。 |
| *groupid* | ユーザーがログイン時に割り当てられるグループの数値識別子。最大しきい値と数値の割り当て方法に関するユーザー ID のガイドラインは、グループ ID にも当てはまります。詳細については、166 ページの「グループ」を参照してください。 |
| *identification* | ユーザーの正式の識別子。このフィールドには、通常、ユーザーのフルネームとその他の情報が入ります。形式は自由ですが、コロンは使用できません。 |
| *homedirectory* | ユーザーのホームディレクトリ。ユーザーが正常にログオンすると、このディレクトリがユーザーのカレントディレクトリになります。 |
| *loginshell* | ユーザーが正常にログインしたときに、UNIX オペレーティングシステムがそのユーザーに対して生成するシェル。このフィールドを空白のままにした場合、ログインシェルのデフォルト設定は、通常 Bourne シェル `/bin/sh` になります。 |

## ファイルの所有権

一般的なファイルシステム (Berkeley Fast File System、System V) では、ファイル所有権という概念を使用して、ファイルへの一次アクセスを制御します。UNIX は、すべてのファイルシステムエンティティーをファイルだと考えます。たとえば、ディレクトリ、実行可能ファイル、およびデータベースは、すべてファイルと考えられます。すべてのファイルには、ユーザーとグループの 2 種類の所有者があります。

ファイルの所有権によって、ユーザーは、ファイルにアクセスし、アクセス権 (「モード」とも呼ぶ) を設定する権利を付与されます。ファイルの所有者は、ファイルに制約を課してほかのユーザーのアクセスを制限できます。ファイルのアクセス権とアクセスについては、168 ページの「ファイルのアクセス権」で説明します。

## 別のユーザーへの切り替え

別のユーザーが所有するファイルを操作するときに、そのユーザーになると便利な場合があります。スーパーユーザーは、`su` (switch user) コマンドを使用すれば、パスワードを知らなくても、どのユーザーにでもなることができます。その逆は絶対にあってはなりません。多くの実装では、別の制御ファイルを使用して `su` をさらに制限しています。実装によって異なる情報については、ご使用のシステムのマニュアルページで、「`su`(1M)」を参照してください。

# グループ

「グループ」とは、共通の要件を備えたユーザーの集合です。グループはセキュリティーの第 2 階層です。たとえば、システム管理者は、テストデータベースにアクセスするグループを、開発チームのユーザーに対して作成できます。そして、グループのメンバーだけがデータベースにアクセスできるように、データベースに対してグループのアクセス権を設定できます。

ユーザーは、ログインすると、パスワードファイルの自分のエントリで、group フィールドに示されたグループに属します。

## グループの管理

グループ情報は、通常 `/etc/group` と呼ばれるグループファイルに保存されます。パスワードファイルの各行が個々のユーザーを表すように、グループファイルの各行は、個々のグループを表します。

グループファイルエントリの形式は、次のようなコロンで区切られたフィールドのリストです。

| *groupname* : *password* : *groupid* : *usernamelist* |
|---|

| 要素 | 説明 |
|---|---|
| *groupname* | グループの名前。通常は 8 文字以内。 |
| *password* | グループのパスワード。グループにはパスワードがないのが普通です。システムによっては、passwd コマンドで、グループファイルにあるグループのパスワードを変更できます。ご使用のシステムの passwd コマンドにこの機能がない場合、グループパスワードを変更する方法については、お手持ちの UNIX 管理者マニュアルを参照してください。 |
| *groupid* | グループの数値識別子。groupid はパスワードファイルで使用され、ユーザーがログインしたときに、ユーザーのデフォルトグループを識別します。 |
| *usernamelist* | グループに属するユーザーのユーザー名をコンマで区切ったリスト。 |

## グループの所有権

グループ所有権を使用すると、複数のユーザーが特定の制約のもとに、ファイルにアクセスできます。ユーザー所有権はユーザーがファイルにアクセスし、アクセス権を設定する権利を付与するのに対し、グループのアクセス権は、アクセスについてだけ機能します。ユーザーがアクセス権を変更できるのは、自分が所有するファイルだけです。すべてのファイルのアクセス権を変更できるのは、スーパーユーザーだけです。

## グループの変更

1 人のユーザーが複数のグループに属することができます。しかし、通常は一度に 1 つのグループのアクセス権しか持つことができません。

グループを変更するには、newgrp コマンドを使用します。ユーザー名が /etc/group のグループのエントリに入っている場合、newgrp は、ユーザーが newgrp の引数として指定されたグループに属する新しいシェルを起動します。ユーザー名が /etc/group のグループエントリになく、そのグループにパスワードが設定されている場合、そのグループに入るには、グループのパスワードを入力する必要があります。パスワードが設定されていない場合、newgrp が新しいシェルを起動しますが、ユーザーは対象のグループには入れられません。id と入力して Return キーを押すと、現在のグループが確認できます。newgrp コマンドは、システムによって動作が異なることがあります。

# ファイルのアクセス権

ファイルのアクセス権またはモードは、ファイルへのアクセスを制御します。ファイルシステムのすべてのファイルには、それに対応するモードがあります。ファイルにモードを設定できるのは、所有者とスーパーユーザーだけです。ファイルのモードは、次の 3 種類のアクセス権を制御します。

| | |
|---|---|
| r | 読み取り |
| w | 書き込み |
| x | 実行 |

次の 3 種類のユーザーのどれに対しても、これら 3 種類のアクセス権をすべて適用できます。

| | |
|---|---|
| u | ユーザー: ファイルの所有者 |
| g | グループ: ユーザーが属するグループ |
| o | その他: すべてのユーザー |

これら 3 つのアクセス権が次の形式で指定されます。

```
rwx rwx rwx
```

最初の `rwx` は所有者のアクセス権、2 番目はグループのアクセス権、3 番目はその他のアクセス権を設定します。

これら 3 つのセットは、多くの場合、3 つの 8 進数で表されます。ビットがオンになっているところでは、アクセス権がアクティブです。たとえば、ファイルモード 750 は、所有者が読み取り権、書き込み権、および実行権を、またグループのメンバーが読み取り権と書き込み権を持ち、その他のユーザーがファイルに対してまったくアクセス権を持たないことを示します。詳細は、170 ページの「アクセス権の変更」と169 ページの「新しいファイルに対するデフォルトのアクセス権」を参照してください。

## ファイルのアクセス権の設定

ファイルシステムのすべてのオブジェクトがファイルとして表されますが、アクセス権はファイルの種類によって異なり、特に、ディレクトリとその他のすべてのファイルでは大きく異なります。

ほとんどのファイルで、次のことが当てはまります。

- 読み取り権とは、ファイルの内容を表示できることを意味します。
- 書き込み権では、ファイルの内容を変更および削除できます。
- 実行権では、通常、ファイルを実行できます。

ただし、これらのアクセス権は、ファイルのディレクトリにアクセスできることが前提条件です。ファイルが入ったディレクトリにアクセスできない場合は、ファイルにはアクセスできません。詳細は、169 ページの「ディレクトリに対するアクセス権の設定」を参照してください。

## ディレクトリに対するアクセス権の設定

ディレクトリに対する読み取り権によって、ユーザーは、ディレクトリの内容にアクセスできるようになります。書き込み権によって、ユーザーは、ディレクトリ内のファイルの作成と削除およびディレクトリ自体を削除できます。ユーザーがディレクトリへの書き込み権を持っているが、そのディレクトリに書き込み権のないサブディレクトリがあり、かつそのサブディレクトリに 1 つ以上のファイルが入っている場合、ユーザーはそのディレクトリを削除できません。アクセス権は、1 つのディレクトリレベルで機能し、そのディレクトリのサブディレクトリには適用されません。

ディレクトリに対する実行権には、検索権が備わっています。ユーザーが読み取り権も持っている場合、実行権によってユーザーは、`cd` コマンドで、そのディレクトリに移動できます。

読み取り、書き込み、実行、およびその組み合わせは、システムによって動作が異なることがあります。特定のアクセス権設定を有効にする前に、常に検証してください。

## 新しいファイルに対するデフォルトのアクセス権

ユーザーが新しいファイルを作成すると、ユーザーの *umask* によってそのファイルのデフォルトのアクセス権が設定されます。umask は、3 つの 8 進数のセットで、それぞれの数値がファイルモードに対応します。通常のファイルでデフォルトのアクセス権を判定するには、umask と 666 の間の排他的論理和 (XOR) を実行します。ディレクトリのデフォルトモードを決定するには、umask と 777 の排他的論理和を求めます。XOR は実際には引き算です。

**例**: ユーザーの umask が 002 の場合、新しく作成するファイルのアクセス権は、666 - 002 = 664 となります。
つまり、所有者および所有者のグループには読み取り権と書き込み権が与えられ、その他のユーザーは読み取り専用アクセスが可能になります。

新しいディレクトリのモードは、777 - 002 = 775 となり、所有者と所有者のグループのメンバーはディレクトリに完全にアクセスできますが、その他のユーザーには読み取り権と実行権だけが与えられます。

umask は `umask` コマンドによって設定します。このコマンドは、通常、ユーザーの `.profile` (Bourne および Korn シェル) または `.login` (C シェル) ファイルにあります。

## アクセス権の変更

アクセス権を変更するコマンドは、chmod です。このコマンドには、アクセス権を指定するための構文が 2 つあります。最初の構文は、ファイルまたはディレクトリに適用するモードを表す 8 進数を chmod に渡します。

**例**: tpcatest というファイルをすべてのユーザーに読み取り可能で、所有者だけが編集できるように設定するには、シェルプロンプトで chmod 644 tpcatest と入力します。

2 番目の方法では、アクセス権に割り当てられた文字と、-、+ と = の記号を使用して、r、w、x のアクセス権を追加します。

**例**: シェルプロンプトで chmod u=rw,go=r tpcatest と入力すると、ファイル tpcatest がすべてのユーザーに対して読み取り可能で、所有者だけが編集できるようにファイルアクセス権が設定されます。

## set-user-id と set-group-id モード

set-user-id と set-group-id モードは特別な用途のフラグで、これらもセキュリティーに関係します。これらのモードは、実行可能ファイルに対して設定でき、実行の間、実行可能ファイルの所有者またはグループ特権をユーザーに与えます。たとえば、kixclean の set-user-id ビットがスーパーユーザーに設定されている場合に、Tony が kixclean を実行すると、スーパーユーザーはこのファイルを所有しているので、スーパーユーザー特権を使用してこれが実行されます。set-user-id ビットがスーパーユーザーに設定されていない場合は、kixclean が Tony のアクセス権を使用して実行されます。set-group-id ビットが設定されていると、実行可能ファイルは実行可能ファイルのグループアクセス権を使用して実行されます。

これらのモードを設定するには、chmod コマンドに 8 進数を指定して使用する必要があります。

**例**: kixclean がスーパーユーザー特権によって実行されるように set-user-id ビットを設定するには、シェルプロンプトで chmod 4755 kixclean と入力します。このビットを設定すると、kixclean のアクセス権が -rws r-x r-x となります。アクセス権の最初のグループにある s は set-user-id ビットです。

kixclean がグループの特権によって実行されるように set-group-id ビットを設定するには、シェルプロンプトで chmod 2755 kixclean と入力します。このビットを設定すると、kixclean のアクセス権が -rwx r-s r-x となります。アクセス権の 2 番目のグループにある s は set-group-id ビットです。

これらの特別なモードの詳細については、ご使用のシステムのマニュアルで「chmod(1)」を参照してください。

### ファイルシステムレベルでのアクセス制御

ファイルシステム全体が読み取り専用の場合、個別のファイルを制御するより、ファイルシステムレベルでアクセス制御を行うほうが簡単です。

すべての mount コマンドでは、スーパーユーザーがファイルシステムを読み取り専用にマウントできます。ファイルシステムが読み取り専用にマウントされている場合、スーパーユーザーを含むすべてのユーザーは、ファイルシステムのファイルを変更できません。すべてのシステムには、一般に `/etc/fstab` と呼ばれるファイルシステムのマウントテーブルがあり、読み取り専用マウントを指定できます。

NFS- または RFS- マウントのファイルシステムを扱う場合、アクセス権が複雑になります。多くの場合、ユーザー名とグループ名だけでなく、ユーザー ID と グループ ID がシステム間で共通している必要があります。つまり、システム A にログインした *bill* が、システム B にある NFS- マウントのファイルシステムにアクセス権を持つには、システム A のユーザー *bill* がシステム B の *bill* と同じユーザー ID を持つ必要があります。NFS 実装は、それぞれ一意です。

このほか、NFS- マウントのファイルシステムで考慮が必要な点として、一般にスーパーユーザーは、複数の NFS マウントで、nobody (user)/nobody (group) のアクセス権を持ちます。システム管理者が NFS を使用して管理ファイルを配布する場合、これらのファイルはほとんど常にスーパーユーザーに属するので、これによって問題が生じることがあります。

## Sun MTPセキュリティー

Sun MTP セキュリティーは、次の要素から成ります。

- 領域へのユーザーサインオンの確認
- 実行可能ファイル、シェルスクリプト、およびデータベースのセキュリティー
- 領域環境内の各トランザクションへのセキュリティーレベルの割り当て
- 領域が、外部セキュリティーマネージャーに統合されている場合、セキュリティー制御は、トランザクションまたはバッチジョブがアクセスする各リソースにも割り当てられます。外部セキュリティーマネージャーの使用方法については、『Sun Mainframe Security Facility 管理者ガイド』を参照してください。

# サインオンセキュリティーの制御

Sun MTP には、サインオン時のセキュリティーを強制するための方法がいくつか用意されています。

- サインオンテーブル (SNT) には各ユーザーの名前とパスワードの情報が入っていて、これにより自動サインオンが可能になります。
- CESN および CSSN トランザクションでは、ユーザーが有効なパスワードを入力する必要があります。
- システム初期化テーブル (SIT) には、パスワードの長さ、パスワードの有効期限、およびパスワード失敗回数の上限を領域全体にわたって制御するフィールドがあります (Sun MTP 組み込みセキュリティー使用時)。これらのフィールドは、ESM を使用するときには無視されます。
- プログラムリストテーブル (PLT) は、ユーザーのサインオン時に、ユーザー作成の検証プログラムが実行されるように構成できる。

## CESN/CESF と CSSN/CSSF の使用

CESN、CSSN、CESF、および CSSF トランザクションは、ユーザーのセキュリティー属性の設定に使用され、トランザクションをサブミットする各ユーザーに対して、Sun MTP は妥当性検査を行うことができます。CESN または CSSN ログオントランザクションおよび CESF または CSSF ログオフトランザクションの詳細については、第 2 章を参照してください。

ユーザーがログイントランザクションの 1 つを使用してログインすると、ユーザー ID とパスワードの組み合わせが、SNT または外部セキュリティーマネージャーを使用している場合はそれが管理する情報によって検査されます。検査に合格すると、SNT の情報が端末管理テーブル (TCT) の情報にマージされます。これにより、アプリケーションプログラムが、EXEC CICS ASSIGN コマンドを使用してこの情報にアクセスできます。サインオフ時に CESF または CSSF トランザクションが実行されると、TCT から取得した情報がクリアされ、外部セキュリティーマネージャー (使用している場合) に通知されます。

## SIT の使用

SIT には、領域内のパスワードセキュリティーを制御するフィールドが含まれています (ESM ではなく Sun MTP 組み込みセキュリティーを使用時)。これらのフィールドは、ESM を使用するときには無視されます。パスワードセキュリティー制御は、ESM によって提供されます。

- password maximum/minimum length フィールドでは、パスワードの長さの最大・最小値を指定できます。CESN トランザクションまたは EXEC CICS CHANGE PASSWORD コマンドを使用してパスワードを変更するときには、設定した長さの要件を満たすパスワードが許容されます。

このフィールドを 0 に設定すると、パスワードの長さの確認が行われません。

- password life フィールドでは、新しく変更されたパスワードが有効な期間を日数で指定します。

  このフィールドを 0 に設定すると、パスワードの有効期限の確認が行われません。

- password fail limit フィールドでは、誤ったパスワードを何度連続して入力すると、システムが自動的にそのユーザーのパスワードを中断するかを指定します。

  このフィールドを 0 に設定すると、パスワード失敗回数の確認が行われません。

SIT フィールドの詳細は、『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。

## PLT の使用

PLT では、ユーザーがその領域にサインオンしたときに実行されるアプリケーションプログラムを指定できます。アプリケーションプログラムは、ユーザーを確認し、ユーザーを領域にサインオンさせます。確認が行われない場合、そのユーザーは強制的にサインオフさせられます。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。

## 自動サインオンの使用

組み込みセキュリティーを使用すると、ユーザーを SNT で定義して、自動サインオンを使用できます。ユーザーが領域に接続を試みると、ユーザー ID が SNT のすべてのエントリと比較されます。一致するエントリが見つかると、ユーザーが自動的に領域にサインオンされます。一致するエントリが見つからない場合、ユーザーは、デフォルトのアクセス権 (デフォルトのトランザクションセキュリティーキーまたはデフォルトのユーザー名のアクセス権) でサインオンされます。『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』の SNT に関する節を参照してください。

ESM を使用する場合、SNT エントリが一致するかどうかに関係なく、ユーザー ID は ESM で設定されているアクセス権に関連付けられます。

## 定義済みの端末デバイスのユーザー名の認証

端末管理テーブル (TCT) のユーザー名をプリンタおよび 3270 デバイスに割り当てることができます。これにより、領域に接続されたときに、デバイスのユーザー名が SNT エントリに関連付けられます。ユーザー名を指定せずにデバイスを構成した場合、デフォルトのセキュリティーキーが使用されます。『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。

ESM を使用する場合、これらの SNT エントリには、ESM でユーザー ID を認証するパスワードが必要です。SNT エントリがそのユーザー ID について存在しない場合、デバイスは ESM で設定されているデフォルトのユーザー ID とそのアクセス権に関連付けられます。

## デフォルトのセキュリティーキーを使用する通信パス

ユーザーサインオンに対応していない、またはそれを必要としない Sun MTP 通信パスが複数存在します。外部セキュリティーマネージャーが使用されない場合、これらの通信パスを使用するすべてのトランザクションは、その他のユーザー確認の手段が用意されていない限り、デフォルトのセキュリティーキー (セキュリティーレベル 1) に関連付けられます。デフォルトトランザクションキーを使用すると、Sun MTP 管理者は、トランザクションまたはリソースへのアクセスを制限できます。

| | |
|---|---|
| EPI クライアント | デフォルトセキュリティーキーの使用が想定されます。 |
| ECI クライアント | ECI 要求で指定されたユーザー名を使用します。 |
| ISC サーバー | EPI のデフォルトセキュリティーキーの使用が想定されます。<br>ECI の要求には、ユーザー名とパスワードを使用し、それらによって確認されます。<br>トランザクション経路指定、機能シップ、および ATI では、要求を開始したトランザクションのユーザー名が使用されます。<br>接続セキュリティーの詳細については、『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。ISC 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。 |
| TN3270 サーバー | ログインやパスワードの確認をしないようオプションを指定して TN3270 サーバーを起動するときは、デフォルトのセキュリティーキーを使用します。TN3270 接続に領域を設定する方法については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。 |

**注 –** さらに、そのユーザーの端末に対して、`CESF/CSSF/EXEC CICS SIGNOFF` を実行すると、すべての通信パスのすべてのトランザクションに対して、デフォルトのセキュリティーキーが使用されます。

# シェルスクリプト、実行可能ファイル、およびファイルのセキュリティー保護

オペレーティング環境のセキュリティー機能を使用して、シェルスクリプト、実行可能ファイル、およびデータベースファイルのセキュリティー保護を行います。各ファイルに対してグループを作成し、適切なアクセス権を設定して保護を必要とするファイルに悪意のあるユーザーがアクセスしたり、それを変更したりできないようにします。

シェルスクリプトに対し、シェルスクリプトファイルと同じグループに属するユーザーのみがシェルスクリプトを実行できるようにアクセス権を設定します。この方法では、ユーザーが実行できるのは、同じグループにあるシェルスクリプトのみです。たとえば、Sun MTP シェルスクリプトを変更して、適切な環境変数を自動的に設定したり、ユーザーを領域にログインさせたり、最初に実行するトランザクションを指定したりすることが可能です。

シェルスクリプトに対しては、次のアクセス権の設定を推奨します。

- アクセス権を 750 に設定して、所有者 (システム管理者) に読み取り権、書き込み権、および実行権を付与します。同じグループ ID を持つユーザーには、読み取り権と実行権を設定し、ほかのユーザーにはアクセス権をまったく付与しません。
- アクセス権を 755 に設定して、すべてのユーザーがそのシェルスクリプトを実行できるようにします。ここで、すべてのユーザーは領域にログオンできます。ログイン後、アプリケーションプログラムまたはログイントランザクションを使用して、セキュリティーをチェックできます。

また、シェルスクリプトを実行するときに、グループパスワードを別のレベルのセキュリティーとしても使用できます。グループにパスワード保護を設定すると、ユーザーは保護されたシェルスクリプトを実行する前に、パスワードを入力する必要があります。

Sun MTP 実行可能ファイルにアクセス権を設定するときには、次のようなサイト固有の要請も考慮します。

- 特定のシステムに対応するように領域を初期化する場合、グループレベルと、場合によっては所有者レベルでアクセス権を設定します。Sun MTP 実行可能ファイルと同じグループ ID を持つ所有者とユーザーだけが領域を起動できるようにするには、実行可能ファイルのアクセス権を 550 に設定します。
- 開発専用のシステムの場合は、Sun MTP 実行可能ファイルのアクセス権を 555 に設定して、すべてのユーザーが領域を起動できるようにします。
- 設置場所で必要なセキュリティーの種類に応じて、実行可能ファイルのディレクトリアクセス権を 750 または 755 に設定します。750 ではグループアクセスのみが可能であり、755 ではすべてのユーザーがアクセスおよび実行できます。

# ユーザーの SNT エントリの状態変更

Sun MTP には、SNT でのユーザーのパスワードと中断状態を動的に変更するために、次の 2 つの方法が用意されています。

- SNT でユーザーのパスワードを変更する
- 関数呼び出しを使用して、ユーザーのパスワードまたは中断状態を変更する

また、ユーザーに対して暗号化パスワードを生成する機能もあります。

## SNT でのユーザーのパスワードの変更

ユーザーが自分のパスワードを忘れてしまうことがあります。管理者は、領域を再起動することなく、すぐに使用できるパスワードをユーザーに提供できます。

### ▼ ユーザーに新しいパスワードを付与する

1. `CTBL` トランザクションを入力し、Table Manager を開きます。
2. 「Table Manager」メニューで、PF5 – 拡張テーブルを押します。
3. 「Standard Tables」メニューで、PF5 – サインオンテーブルを押します。
4. SNT メイン画面で、新しいパスワードを必要としているユーザーの名前にカーソルを合わせ、PF9 キーを押して「Security/Accounting」画面を表示します。
5. 図 7-1 に示す「Security/Accounting」画面が表示されたら、「Password」フィールドに新しいパスワードを入力します。

   新しいパスワードを入力するときに、入力したパスワードが画面に表示されます。これは、「Password」フィールドが保護されていないことを意味しています。デフォルトでは、このフィールドには英数字が入力でき、すべて数値のパスワードが使用できます。すべて数値のパスワードまたはすべて英字のパスワードを必要とする場合は、領域を起動する前に、SIT の password type フィールドを設定する必要があります。このフィールドの詳細については、『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。

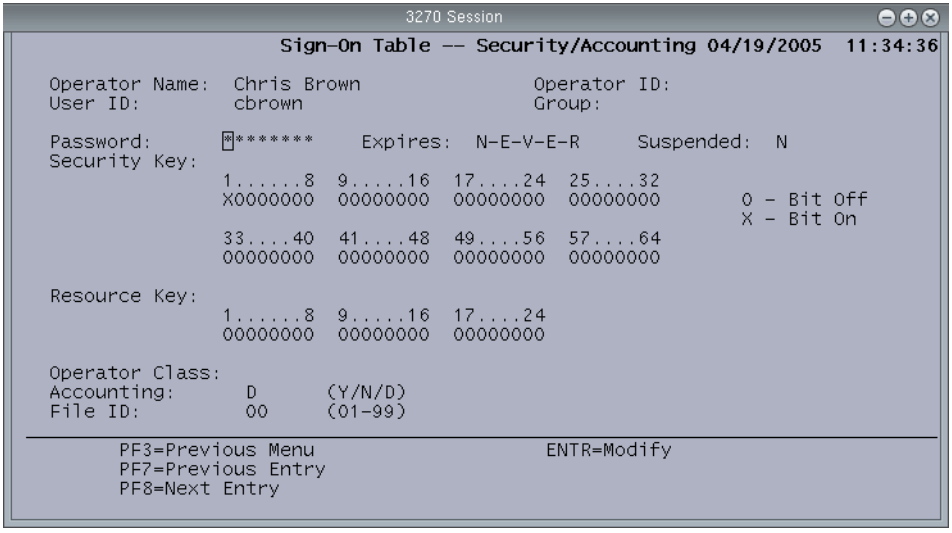


**図 7-1** 「Sign-On Table—Security/Accounting」画面

6. **Enter キーを押して、変更を実行します。**

   「Password」フィールドは保護状態に戻ります。

7. **PF3 キーを押して、SNT メイン画面に戻ります。**

8. **PF2 キーを押して、変更内容をディスクに書き込みます。**

9. **Table Manager を終了するまで PF3 キーを押します。**

PF2 キーを押して変更を保存すると、変更内容がディスクに書き込まれ、共有メモリーの SNT が更新されます。メモリー内の SNT は、領域にサインオンする際、CESN/CSSN トランザクションが使用します。

---

**注 –** 書き込みに失敗した場合、共有メモリーは更新されません。

---

オペレータ名は SNT 内で一意で、その名前がパスワードに関連付けられているので、SNT のエントリを検索してパスワードを更新するときに、オペレータ名を使用します。新しいユーザーを SNT に追加したり、ユーザーのパスワードとオペレータ名の両方を変更したりした場合、これらの変更内容は、共有メモリー内の SNT には加えられません。変更内容を有効にするためには、領域を再起動する必要があります。

動的パスワード変更機能を無効にするには、領域を起動する前に、環境変数 KIXSNT_NOMEMUPDATE を任意の値に設定する必要があります。次に例を示します。

```
$ KIXSNT_NOMEMUPDATE=1;export KIXSNT_NOMEMUPDATE
```

動的なパスワード変更機能を無効にした場合は、SNT に変更を反映させるために、領域を再起動する必要があります。

## 関数呼び出しを使用したユーザーの SNT 状態の変更

**注 –** これらの関数は、ESM が有効になっていないときにのみ使用されます。

Sun MTP には、セキュリティー保護されているトランザクションから呼び出して、ユーザーの状態を変更するために、次の 2 つの関数が用意されています。

- kx_pw_reset は、ユーザーの SNT パスワードを変更します。
- kx_pw_stachg は、ユーザーの SNT パスワードの状態を変更します。

kx_pw_reset 関数呼び出しでは、次の 2 つの引数を取ります。

| | |
|---|---|
| userid | パスワードをリセットするユーザーのユーザー ID |
| password | プレーンテキストで表示した新しいパスワード |

kx_pw_stachg 関数呼び出しでは、次の 2 つの引数を取ります。

| | |
|---|---|
| userid | パスワード状態を変更するユーザーのユーザー ID |
| state | 新しい SNT パスワードの状態の文字値<br>Y: パスワードは停止されます。<br>N: パスワードは停止されません。<br>M: 次回の確認時に、ユーザーがパスワードを変更する必要があります。 |

これらの関数のどちらかまたは両方が入ったプログラムは、セキュリティー保護されたトランザクションによって呼び出す必要があります。それぞれの関数は、トランザクションを実行しているユーザーの SNT エントリの Resource Key フィールドを確認します。次のビットは、関数を実行する前に、ユーザーの SNT エントリに設定する必要があります。

- kx_pw_reset 関数を呼び出すトランザクションをユーザーが実行する場合は、Resource Key フィールドのビット 20 をオンに設定します。
- kx_pw_stachg 関数を呼び出すトランザクションをユーザーが実行する場合は、Resource Key フィールドのビット 19 をオンに設定します。

SNT エントリの Resource Key が無効な場合も、ユーザーのトランザクションには何も通知されません。関数は正常に処理されているように見えますが (0 を返す)、SNT は変更されず、次のセキュリティー確認メッセージがログに記録されます。

```
KIX0481E Userid userid trmID=terminalID trnID=transactionID not authorized
```

呼び出した関数に対して Resource Key が正しい場合でも、次のようなほかのエラーが関数に生じる場合があります。

- SNT にユーザー ID が見つからない場合、関数は 1 を返します。
- 状態が無効な状態の場合、関数は 2 を返します。

次に、これらの関数を COBOL プログラムでコーディングする方法の例を示します。

**コード例 7-1**　　セキュリティー機能のコーディング—COBOL

```
01 PW-FUNCTIONS.
       05 USID PIC X(8).
       05 PASW PIC X(8).
       05 STATE PIC X.
       05 RET PIC 9(9) COMP.
...
CALL 'kx_pw_reset' USING USID PASW GIVING RET.
...
CALL 'kx_pw_stachg' USING USID STATE GIVING RET.
```

## アプリケーションを使用した暗号化パスワードの生成

kxuser_encryptpw 関数をアプリケーションプログラムで使用して、ユーザーの暗号化パスワードを生成できます。この関数は、13 文字の暗号化された形式のパスワードを返します。この関数を使用して新しいユーザーとその暗号化パスワードを snt.lst ファイルに追加したのち、snt.lst を snt.tbl 形式にインポートできます。

この関数の形式は次のとおりです。

```
char *kxuser_encryptpw(char *puserid, char *ppasswd, char *psalt)
```

説明:

| | |
|---|---|
| puserid | 1 ～ 14 文字のユーザー ID |
| ppasswd | 暗号化するクリアテキストのパスワード。長さは 8 文字で、空白が埋め込まれたフィールドです。 |
| psalt | 暗号化に使用される 2 文字の “salt” へのオプションのポインタ。“salt” の詳細については、ご使用のシステムのマニュアルで「crypt(3C)」を参照してください。 |

**注 –** 暗号化アルゴリズムの解読を防ぐため、この関数は共有ライブラリとして提供されていて、それを使用するアプリケーションに結合されません。また、この共有オブジェクトも無権限使用から保護する必要があります。

**例**: 次の例では、ユーザーの暗号化パスワードを返す C プログラムをコンパイル、リンクする方法を示します。

コード例 7-2 では、サンプルのユーザー ID、パスワード、およびオプションの “salt” を暗号化したパスワードを返す C のソースファイル `test.c` を示します。

**コード例 7-2** パスワード暗号化コード— C 言語

```
#include <stdio.h>
main()
{
char luserid[] = "steve\0";
char lpasswd[] = "mypasswd\0";
char lsalt[] = "xy\0";

printf("Testing shared object call of password encryption function\n")

printf("Result of encryption of userid %8.8s password %8.8s (NULL salt) is
%13.13s\n", luserid, lpasswd, kxuser_encryptpw(luserid, lpasswd2, NULL));

printf("Result of encryption of userid %8.8s password %8.8s (using salt) is
%13.13s\n", luserid, lpasswd, kxuser_encryptpw(luserid, lpasswd2, lsalt));
}
```

このプログラムは、`kxuser_encryptpw()` 関数 (`$UNIKIX/lib/libencryptpw.so`) を含む共有オブジェクトライブラリを指定して、コンパイル、リンクします。このライブラリは実行時に動的にリンクされます。次のコマンドは、このライブラリを Solaris プラットフォームでコンパイル、リンクするために必要なオプションを示します。

```
$ cc -Bdynamic -Kpic -R $UNIKIX/lib -L $UNIKIX/lib -lencryptpw test.c
```

次の例は、コンパイル、リンクされたプログラムを実行した結果を示します。

**コード例 7-3** パスワード暗号化の出力

```
Testing shared object call of password encryption function
Result of encryption of userid  steve password mypasswd (NULL salt) is diekfQgaiPUUT
Result of encryption of userid  steve password mypasswd (using salt) is xyQ[R‘rpt_LTK
```

# トランザクションセキュリティーの管理

本番稼働環境で運用するときには、一定のトランザクションを特権ユーザーだけが実行できるように制限する必要があります。すべてのユーザーに影響を与える操作を行うトランザクションは制限します。逆に、すべてのユーザーが利用できることが必要な CESN、CSSN、CESF、および CSSF のようなトランザクションは制限しません。システムトランザクションの詳細については、第 2 章を参照してください。アプリケーショントランザクションを特定のユーザーに制限するには、組み込みのトランザクションセキュリティー機能を使用するか、Sun MTP Secure によって外部セキュリティーマネージャーを使用します。Sun Mainframe Security Facility (Sun MSF) については、『Sun Mainframe Security Facility 管理者ガイド』を参照してください。

次の節では、組み込みのトランザクションセキュリティーについて説明します。

## トランザクションセキュリティーのための SNT と PCT の使用

Sun MTP は、SNT と PCT によってトランザクションセキュリティーを制御します。SNT は、64 ビットのセキュリティーキーを持ち、ユーザーが実行を許可されたトランザクションのクラスをそれによって指定します。ユーザーがサインオンすると、このセキュリティーキーが、ユーザーの TCT エントリとマージされ、ユーザーが実行するすべてのトランザクションに適用されます。

PCT のトランザクションセキュリティー (Trans Sec) フィールドには、1 ～ 64 の数字が含まれており、トランザクションが割り当てられている特定のクラスを示します。デフォルトで、すべてのトランザクションはセキュリティークラス 1 に割り当てられます。また、すべてのユーザーは、デフォルトで、セキュリティークラス 1 のトランザクションを実行できます。

ユーザーがトランザクションをサブミットすると、セキュリティークラスが PCT エントリから抽出されて、セキュリティーキービットマスクへの索引として使用されます。たとえば、ユーザーがセキュリティークラス 5 のトランザクションを実行すると、セキュリティーキーの最上位ビットから 5 番目のビットがオンかどうかが確認されます。このビットがオンの場合、ユーザーはアクセスを許可され、要求したトランザクションが正常に実行されます。このビットがオンでない場合、ユーザーは要求したトランザクションにアクセスできず、次のメッセージが表示されます。

```
TRANSACTION NOT AUTHORIZED FOR USER
```

## セキュリティーキー

セキュリティーキーは、ビットマスク内の 64 ビットで構成されます。SNT の中で、これらのビットは 8 文字ずつ 8 つのグループに分割され、最初のグループには 1〜8、2 番目のグループには 9 〜 16 というように番号が付けられます。グループは、一番左のビットが最小のビットになります。

**例**: ユーザーである Chris Brown は、通常の業務を完了するために、複数の異なるトランザクションを使用する必要があります。これらのトランザクションは、PCT の中のセキュリティークラス 12、38、42、および 44 に割り当てられています。図 7-2 では、Chris Brown がそれらのトランザクションを使用できるようにするために、「SNT-Security/Accounting」画面で設定するエントリを示します。

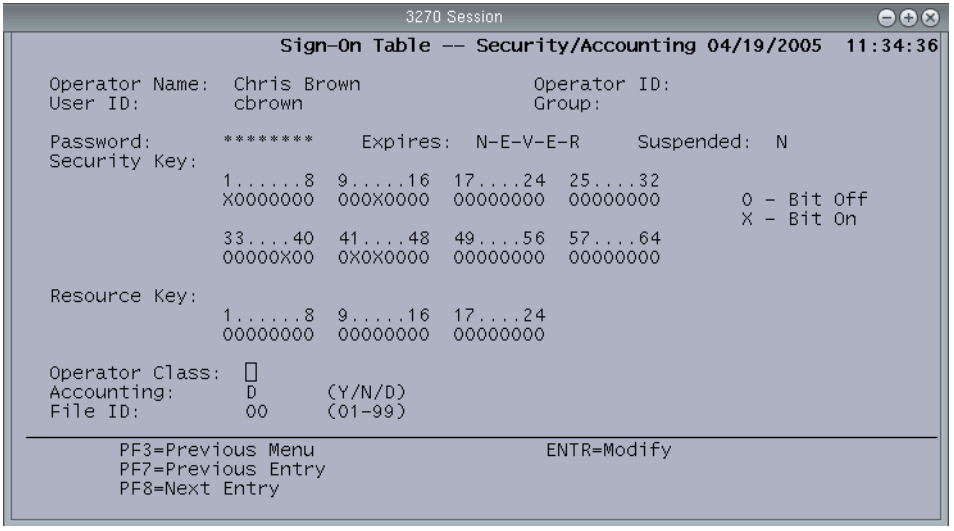


**図 7-2** SNT— トランザクションセキュリティーの例

画面で X の印が付いたセキュリティークラスに割り当てられたトランザクションに、ユーザーがアクセスできるようにするだけでなく、例で示されたセキュリティーキーもまたビットを 1 に設定します。これにより、それらのトランザクションのセキュリティーキーが PCT で変更されない限り、ユーザーはシステムトランザクションにアクセスできます。

## Sun MTP セキュリティー機能のトリガー

組み込みのトランザクションセキュリティー機能は、SNT の設定方法に基づいて自動的にトリガーされます。SNT エントリには、オペレータ名と Sun MTP ユーザー名が入っています。領域に入るユーザーのユーザー名は、テーブル内のすべてのユーザー名と比較されます。一致するユーザー名が検出された場合、そのユーザーは自分が許可されたセキュリティークラスのトランザクションを実行できます。一致するユーザー名が見つからない場合、ユーザーはサインオンはできても、PCT セキュリティークラスが 1 のトランザクションしか実行できません。

このユーザーが別の特権を持つ別のユーザーとしてサインオンするには、`CESN` または `CSSN` トランザクションに Sun MTP ユーザー名とパスワードを指定して入力します。ユーザー名とパスワードが SNT で定義されたユーザー名とパスワードに一致すれば、ユーザーはサインオンして、SNT のユーザー名に割り当てられたセキュリティークラスのトランザクションを使用できます。

## 自動トランザクションの管理

セキュリティーを設定する場合は、PLT で定義されているプログラムを評価する必要があります。`CPLT` トランザクションでは、ユーザーが領域にサインオンしたときに、PLT で定義されたプログラムを自動的に実行できます。`CPLT` トランザクションを含め、端末から実行するトランザクションは、すべてセキュリティーアクセスについて確認されます。

クライアントの起動時に、トランザクションが自動実行されます。`unikix -t` オプションを使用すると、`-t` 引数によって指定されたトランザクションが実行されます。このオプションを使用しない場合は、`CPLT` トランザクションが実行されます。どちらの場合も、トランザクションは、端末から実行されるすべてのトランザクションと同様にセキュリティーを確認されます。組み込みセキュリティー機能によるセキュリティー確認が省略されるのは、`START` コマンドまたは一時データキュートリガーによって、トランザクションが自動的に実行される場合のみです。ただし、外部セキュリティーが使用されている場合は、これらのトランザクションも確認の対象となります。

## Sun MTP システムトランザクションのセキュリティー

第 2 章で説明されている Sun MTP が提供するトランザクションには、考慮する必要があるセキュリティー要件があります。これらのトランザクションは、次のいずれかに分類されます。

- 無制限。すべてのユーザーが実行できます。
- 制御。承認されたユーザーにアクセスを制限します。

無制限トランザクションは、サインオンしているかどうかに関係なく、すべてのユーザーからアクセスできる必要があります。組み込みトランザクションセキュリティーでは、このトランザクションについて PCT エントリで設定されているデフォルトのセキュリティークラスで問題ないため、対策は必要ありません。Sun MTP Secure が有効で ESM を使用する場合、次のトランザクションは、デフォルトのユーザー ID (`$KIXSECDFLTUSER` に設定) と、ローカルクライアントで Sun MTP 領域に接続する (`unikix`/`unikixl`) すべての UNIX ユーザー ID に認められたアクセス権を実行する必要があります。

`CESN`
`CSSN`
`CESF`
`CSSF`
`CSPG`
`CCIN`
`CRSR`
`CPLT`
`CSMT`。
このトランザクションを使用して端末からユーザーを切断する場合、ユーザーにはこのデフォルトのアクセスレベルが必要ですが、KIX-COMMANDS SHUTDOWN セキュリティーも使用して、`CSMT SHUT,YES` トランザクションを権限のないユーザーが使用し、領域を停止してすべてのユーザーを切断することがないようにすることを強く推奨します。

制御トランザクションには、Sun MTP が提供するその他のトランザクションのすべてが含まれます。これらのトランザクションは、決まったセキュリティーロールの機能を提供します。また、これらトランザクションを実行する適切なアクセス権を承認されたユーザーだけに認めるように、グループ化する必要があります。次の表には、これらのトランザクションをグループ化する方法の例を示します。サイトごとに要件が異なります。

**表 7-1** 制御システムトランザクションのグループ化

| トランザクション | ユーザーの役割 |
|---|---|
| `CEBR` - 一時記憶域のブラウズ | 開発者 |
| `CECI` - CICS コマンドを会話的に実行 | 開発者 |
| `CEDF` - デバッグ機能 | 開発者 |

**表 7-1**　制御システムトランザクションのグループ化 (続き)

| トランザクション | ユーザーの役割 |
|---|---|
| CMNU - 開発システムのメインメニュー | 開発者 |
| CSGU - Screen Generation Utility | 開発者 |
| CEDA - 代替リソース定義 | 領域管理者 |
| CEMT - 領域状態の設定/表示 | 領域管理者 |
| CFMS - File Manager | 領域管理者 |
| CBCH - バッチジョブの実行 | 領域管理者 |
| CINI - トランザクション処理プログラムの再初期化 | 領域管理者 |
| CRED - レコードエディタ | 領域管理者 |
| CTBL - Table Manager | 領域管理者 |
| CPMI - LU6.2 ミラートランザクション | ISC ユーザー |
| CRTE - 別のシステムへのトランザクション経路指定 | ISC ユーザー |
| CVMI - LU6.2 同期レベル 1 ミラートランザクション | ISC ユーザー |
| CTIN - EPI クライアントトランザクション | EPI クライアントユーザー |

Sun MTP 組み込みトランザクションセキュリティーでは、これらトランザクションに関する PCT エントリには、役割を示すセキュリティークラスを割り当てる必要があり、権限を持つユーザーの SNT エントリにはそのセキュリティーキーが必要です。Sun MTP Secure では、ESM でこれらの役割とアクセス権を定義する必要があります。

**注 –** ESM を使用しており、構成済みの起動プログラムまたは停止プログラムがプログラムリストテーブル (PLT) にある場合は、203 ページの「起動プログラムと停止プログラム」を参照してください。

# セキュリティーの例

この節の例では、オペレーティング環境と Sun MTP のセキュリティー手法を応用して、アプリケーションやファイルへのアクセスを制御する方法を示します。セキュリティーの程度は環境によって決まり、本番稼働環境では開発環境より強固なセキュリティーが要求されます。

図 7-3 では、財務アプリケーションと会計アプリケーションがある領域を示します。

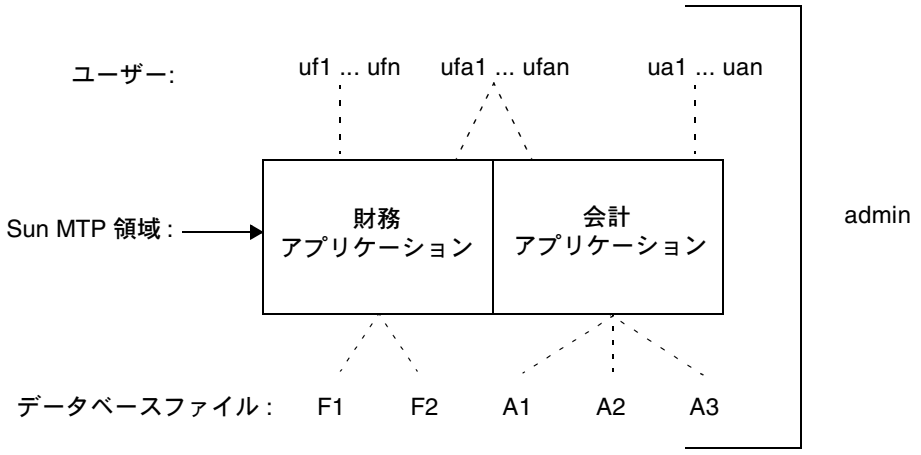


**図 7-3** Sun MTP セキュリティーの例

サンプル領域のユーザーには、次のアクセス要件があります。

| | |
|---|---|
| ユーザー uf1 ～ uf*n* | 財務アプリケーションに関係するトランザクションのみを実行します。 |
| ユーザー ua1 ～ ua*n* | 会計アプリケーションに関係するトランザクションのみを実行します。 |
| ユーザー uf1 ～ uf*n* | 両方のアプリケーションでトランザクションを実行します。 |
| admin ユーザー | 領域を管理します。admin ユーザーは、あらゆるアプリケーショントランザクション、システムトランザクション、およびデータベースファイルにアクセスできる必要があります。 |
| その他のすべてのユーザー | それぞれのアプリケーションのみにアクセスが制限されます。 |

# ユーザーサインオンセキュリティー

ユーザーサインオンセキュリティーを制御するには、グループ ID セキュリティー、CESN や CSSN トランザクション、ユーザーが供給するセキュリティートランザクションなど、いくつかの方法があります。この節の例では、それぞれの方法を説明します。

## グループ ID セキュリティー

必要なアクセス権に応じて、ユーザーをグループに割り当てます。次に例を示します。

- ユーザー uf1 ～ uf*n*: fin グループ
- ユーザー ua1 ～ ua*n*: acct グループ
- ユーザー ufa1 ～ ufa2: finacct グループ
- admin: unikix グループ

グループ ID 名は、/etc/group ファイルで、次のように定義されています。

```
fin::123:uf1,uf2,...ufn,admin
acct::124:ua1,ua2,...uan,admin
finacct::125:ufa1,ufa2,...ufan,admin
unikix::200:admin
```

ユーザーは、/etc/passwd ファイルで、次のように定義されています。

```
uf1::105:123:Albert Lavine:/usr/uf1:/bin/sh
uf2::106:123:Phil Jess:/usr/uf2:/bin/s
       .
       .
ua1::205:124:Dave Meir:/user/ua1:/bin/sh
ua2::206:124:Joe Terin:/user/ua2:/bin/sh
       .
       .
ufa1::305:125:Bill Raves:/user/ufa1:/bin/sh
ufa2::306:125:Hal Sample:/user/ufa2:/bin/sh
       .
       .
```

**注 –** admin ユーザーは、すべてのグループに割り当てられます。

各グループに対し、admin ユーザーは、特定のグループのユーザーを適切なアプリケーションに入れるための、特定のサインオントランザクションを実行するように Sun MTP サインオンスクリプトを変更します。各サインオンスクリプトのアクセス権は、指定したグループだけがそれを実行できるように設定します。またシステム管理者は、ユーザーがサインオンしたとき、変更したサインオンスクリプトが自動実行されるように、/etc/password ファイルを編集できます。たとえば、サインオンシェルスクリプトが /mtp/mtp81/bin/shell.grp1 の場合、パスワードファイルのエントリは次のようになります。

```
ufa1::305:125:Bill Ray:/user/ufa1:/mtp/mtp81/bin/shll.grp1
```

グループ ID にパスワードを割り当て、ユーザーをそのグループにサインオンさせないことによって、別のレベルのセキュリティーを追加できます。これにより、ユーザーは、newgrp を実行してグループに参加することが必要になります。

## CESN または CSSN トランザクション

CESN と CSSN トランザクションを使用して、ユーザーがログインしてトランザクションを実行するためのアクセス権を持つことを確認します。CESN または CSSN トランザクションをグループ ID セキュリティーとともに使用します。

システム管理者は、適切なアクセス権を使用してサインオンスクリプトを変更し、CESN または CSSN トランザクションがそのユーザーのために実行される最初のトランザクションであることを示すか、PLT を変更して、ユーザーがサインオンするたびに、CESN または CSSN トランザクションの実行を要求します。CESN と CSSN トランザクションについては、第 2 章を、PLT 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。

## ユーザー指定のセキュリティートランザクション

システム管理者は、ユーザーがサインオンするたびに実行されるサイトセキュリティートランザクションを指定できます。このトランザクションは、ユーザーに情報入力を求め、次に入力された情報を有効なデータが入ったファイルと照らし合わせます。確認ができなかった場合、ユーザーはサインオフされます。ユーザーのサインオン時に実行されるセキュリティートランザクションは、PLT によって指定します。

# シェルスクリプトと実行可能ファイルのセキュリティー

システム管理者は、シェルスクリプトと実行可能ファイルをユーザーが、誤ってまたは悪意で破壊したり、場合によっては読み取ったりしないよう、これらのファイルを保護する必要があります。たとえば、ユーザーを領域にログオンさせるシェルスクリプトは、同じグループ以外のユーザーが読み取ったり実行したりできないようにする必要があります。システムをセキュリティー保護するために、領域をセットアップします。

**例**: サインオンに使用するシェルスクリプトに対して、アクセス権を設定します。

```
$ chmod 750 unikix
```

- システム管理者には、読み取り権、書き込み権、および実行権があります。
- 同じグループに属するユーザーには、読み取り権と実行権があります。
- その他のすべてのユーザーには、アクセス権はありません。

システム管理者は、管理者のみが領域を起動できるようにアクセス権を設定することも、あらゆるユーザーが領域を起動できるように設定することも可能です。

**例**: 本番稼働環境では、システム管理者のみが領域を起動できるよう設定する必要があります。システム管理者が起動する場合を除き、開発システムが本番稼動システム上で実行できないようにアクセス権を設定します。

```
$ chmod 740 kixstart
```

- システム管理者には、読み取り権、書き込み権、および実行権があります。
- 同じグループに属するユーザーには、読み取り権があります。
- その他のすべてのユーザーには、アクセス権はありません。

**例**: 本番稼動領域と開発領域を同じシステム上で実行する場合、ほかのグループのユーザーが領域を起動できるようにアクセス権を設定します。

```
$ chmod 755 kixstart
```

- 所有者またはシステム管理者には、読み取り権、書き込み権、および実行権があります。
- 同じグループ ID に属するユーザーには、読み取り権と実行権があります。
- その他のすべてのユーザーには、読み取り権と実行権があります。

# データベースファイルセキュリティー

図 7-3 に示すデータベースファイル A1、A2、A3、F1 および F2 をセキュリティー保護するために、適切なアクセス権 (たとえば、760) を使用してファイルを作成し、そのファイルをユーザーのグループとは別のグループ (たとえば、finoff) に割り当てます。このアクセス権は、所有者と finoff グループに属するユーザーだけがファイルを読み書きでき、ほかのすべてのユーザーにはアクセス権が与えられないことを示します。この方法では、領域が管理者によって管理者の一次グループで起動されるので、領域を finoff グループで実行したり、ファイルにアクセスしたりできます。

# トランザクションセキュリティー

Sun MTP トランザクションセキュリティーシステムは、どのユーザーがどのトランザクションを実行できるかを制御します。この節の最初で説明したトランザクションセキュリティーの要件を実装するためには、SNT と PCT を次のように変更します。

PCT でクラスをトランザクションに割り当てます。

| | |
|---|---|
| 財務アプリケーション | セキュリティークラス 5 |
| 会計アプリケーション | セキュリティークラス 6 |
| CMNU、CTBL トランザクション | セキュリティークラス 60 |

SNT でクラスをユーザーに割り当てます。

| | |
|---|---|
| ユーザー uf1 ～ uf*n* | セキュリティークラス 1、5 |
| ユーザー ua1 ～ ua*n* | セキュリティークラス 1、6 |
| ufa1 ～ ufa*n* | セキュリティークラス 1、5、6 |
| システム管理者 | セキュリティークラス 1、5、6、60 |

この設定で、次のことが可能になります。

- ユーザー uf1 ～ uf*n* は、セキュリティークラス 5 のトランザクションである財務だけを実行できます。
- ユーザー ua1 ～ ua*n* は、セキュリティークラス 6 のトランザクションである会計だけを実行できます。
- ユーザー ufa1 ～ ufa*n* は、セキュリティークラス 5 と 6 のトランザクションを実行できます。
- システム管理者以外のユーザーは、システムトランザクション CMNU または CTBL を実行できません。

第8章

# Sun MTP Secure

この章では、外部セキュリティーの概要と、外部セキュリティー管理 (ESM) ソフトウェアとのインタフェースとして使用する Sun MTP Secure について説明します。次の節によって構成されます。



## 外部セキュリティー管理について

外部セキュリティー管理によって、SNT ベースの認証およびトランザクションレベルセキュリティー (TSL) よりも強化されたセキュリティー機能を Sun MTP から提供できます。ESM によるユーザー認証では、領域の SNT に代わる LDAP ディレクトリやその他の集約されたグローバルリポジトリなどの優れた機能が使用でき、またシングルサインオンをサポートできます。ESM によるアクセス制御では、すべての Sun MTP リソースにリソースレベルセキュリティー (RSL) を提供します。これらのリソースには、トランザクションのほか、VSAM ファイル、アプリケーションプログラム、端末が含まれます。

企業のセキュリティーポリシーをサポートするために役割にはそれぞれ必要なアクセス権が与えられていますが、ESM は、一般にユーザーとリソースをそれらの役割にグループ化するセキュリティールールをモデル化します。

# ESM と Sun MTP の統合

ここでは、ESM をある領域について実装するために必要な高いレベルのタスクについて説明します。

1. **使用しているアプリケーション環境のセキュリティーポリシーを定義します。**

   このポリシーは多くの場合、サイトのセキュリティー管理者と共同で作成します。ポリシーには、アプリケーションのすべてのセキュリティー要件を定義しておく必要があります。

2. **ESM ソフトウェアをインストールします。**
3. **ESM ソフトウェアを設定します。**
4. **ESM セキュリティーリポジトリが設定されていることを確認します。**
5. **領域をセットアップします (このタスクは、前のタスクと同時に実行できます)。**

   a. **サインオンテーブル (SNT) にデフォルトのユーザーを定義します。**

   b. **端末管理テーブル (TCT) と SNT で、プリセットセキュリティーがある端末を定義します。**

   c. **領域の設定ファイルで、ESM 環境変数を設定します。**

      これらの変数の詳細については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

   d. **領域の設定ファイルで、Sun MTP Secure 環境変数を設定します。**

      これらの変数の詳細については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

6. **この領域を停止します。**
7. **新しい設定ファイルを実行し、Sun MTP Secure と ESM を使用するように領域の環境を設定します。**
8. **この領域を起動します。**

# Sun MTP Secure の使用

Sun MTP Secure は、領域と次に示す外部セキュリティー管理の実装とのインタフェースを提供します。

- Sun MSF ソフトウェア。詳細は、『Sun Mainframe Security Facility 管理者ガイド』を参照してください。
- Sun MTP が提供するセキュリティー出口。詳細は、252 ページの「セキュリティーユーザー出口ルーチン」を参照してください。
- Sun 以外の外部セキュリティー管理システム。Sun 以外のソフトウェアについては、Sun Microsystems にお問い合わせください。

Sun MTP Secure は、ESM を使用して次の機能をサポートします。

**サインオン時のユーザー認証。**ユーザーが `CESN` サインオントランザクション、または `EXEC CICS SIGNON` コマンドでサインオンする際、Sun MTP Secure はユーザー名とパスワードの組み合わせの認証に、SNT エントリではなく、外部セキュリティーマネージャーを使用します。`CSSN` によるサインオンは、ESM を使用しているときには (直接) サポートされていません。SNT が設定され、`CSSN` を使用してオペレータ名の照会を行う場合、ESM は SNT の対応するユーザー ID と入力されるパスワードを使用して認証します。ただし、オペレータ名とユーザー ID を組み合わせたエントリを、ESM リポジトリに対応する SNT で保持する必要があるため、この方法はお勧めしません。

**VSAM ファイル、アプリケーションプログラム、キューなどの領域リソースに対するアクセス制御。**トランザクションでリソースにアクセスする場合、Sun MTP Secure を通じて外部セキュリティーマネージャーにリソース確認を発行して、トランザクションを実行しているユーザーがアクセス権限を持っているかを判定します。また、Sun MTP Secure は、ESM で確認するリソース名に接頭辞を追加するオプションも提供します。

---

**注 –** PCT、SNT の「Security/Accounting」画面、SIT のセキュリティー関連のフィールドは、Sun MTP Secure が有効な場合は無視されます。

---

Sun MTP Secure は、デフォルトのユーザー ID を使用した権限のない接続に対するリソースアクセス制御もサポートします。このユーザー ID には、企業のセキュリティーポリシーによって保護されていないと考えられるすべてのリソースへのアクセス権が ESM リポジトリで設定されており、サインオントランザクション `CESN` を実行する権限が与えられています。Sun MTP Secure には、そのユーザー ID とパスワードを含む SNT エントリが必要です。このユーザー ID は、`KIXSECDFLTUSER` 環境変数によって領域に対して識別され、領域のセットアップ時に ESM で認証されます。

Sun MTP Secure は、プリセット端末セキュリティーもサポートしています。これは、プリンタなどの端末が、START された、またはトリガーされたトランザクションを、そのリソースのアクセス権に対して設定されているユーザー ID を使用して実行できる機能です。Sun MTP Secure は、それらのユーザー ID に SNT でパスワードで設定されていること、および領域の起動時にそれらを ESM で認証することを必要とします。

## Sun MTP Secure の有効化

外部セキュリティー管理システムを有効にするには、領域の設定ファイルで、KIXSEC 環境変数を YES に設定して Sun MTP Secure を有効にする必要があります。これにより、領域のすべてのリソースに対するリソース確認とユーザー認証が ESM から有効になります。

リソースタイプについて環境変数を NO に設定すれば、リソース確認を無効にできます。たとえば、領域の設定ファイルに KIXTSTSEC=NO を設定すれば、一時記憶域ファイルに対する確認が無効になります。『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』の Sun MTP Secure 環境変数に関する節を参照してください。

Sun MTP Secure を有効にした領域を起動する前に、デフォルトのユーザー名とパスワード、プリセットセキュリティー TCT のユーザー名とパスワードを SNT に設定する必要があります。

## デフォルトのユーザー名

領域への接続を、UNIX クライアント以外による方法、つまり CESN トランザクション、あるいは EXEC CICS SIGNON によるサインオン以外で実行される場合、または接続が CESF トランザクション、あるいは EXEC CICS SIGNOFF でサインオンされる場合、接続で特定されるユーザー名は確認されません。この場合、Sun MTP Secure は KIXSECDFLTUSER 環境変数に定義されたデフォルトのユーザー名を使用します。

デフォルトのユーザー名とパスワードは、領域の起動時に、Sun MTP Secure を通じて外部セキュリティーマネージャーによって認証されます。デフォルトのユーザー名とパスワードが認証できない場合、領域は実行できず、unikixmain が終了します。

## プリセットセキュリティー端末のユーザー名の認証

Sun MTP Secure を有効にしている場合は、プリセットセキュリティーについて TCT で設定されているユーザー名が、起動時に外部セキュリティーマネージャーによって認証されます。ユーザー名と正しいパスワードを SNT で構成する必要があります。これは、上記で説明されているデフォルトユーザー名と要件は同じです。ユーザー名とパスワードが認証できない場合、領域は実行できず、`unikixmain` が終了します。

## ▼ ユーザー名とパスワードを設定する

1. **環境変数を `KIXSEC=NO` と設定してセキュリティー確認を無効にします。**
2. **この領域を起動します。**
3. **SNT にユーザー名とパスワードを、また必要な TCT エントリに対応するユーザー名を追加します。**

   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。
4. **この領域を停止します。**
5. **環境変数を `KIXSEC=YES` と設定してセキュリティー確認を有効にし、すべての `KIX`*xxx*`SEC` 環境変数を `NO` に設定してリソースセキュリティーをすべて無効にします。**
6. **`KIXSECDFLTUSER` 環境変数をデフォルトのユーザー ID に設定します。**
7. **必要な ESM 環境変数を設定します。**
8. **この領域を起動します。**

# デフォルトのユーザー名を必要とする通信パス

領域へのユーザーサインオンに対応していない、またはそれを必要としない通信パスが複数存在します。この通信パスを使用するすべてのトランザクションは、これ以外にユーザー確認の方法が用意されていない場合や、使用する端末が、プリセットセキュリティーとともに TCT で設定されているものではない場合、デフォルトのユーザー名と関連付けられます。デフォルトのユーザー名を使用すると、セキュリティー管理者は、トランザクションまたはリソースへのアクセスを制限できます。

| | |
|---|---|
| EPI クライアント | デフォルトのユーザー名を使用します。 |
| ECI クライアント | ECI 要求で指定されたユーザー名を使用します。 |
| ISC サーバー | EPI にデフォルトのユーザー名を使用します。<br>ECI の要求には、ユーザー名とパスワードを使用し、それらによって確認されます。<br>トランザクション経路指定、機能シップおよび ATI は、デフォルトのユーザー名 (ローカルセキュリティー) を使用します。<br>接続セキュリティーの詳細については、『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。ISC 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。 |
| TN3270 サーバー | ログインやパスワードの確認をしないようオプションを指定して TN3270 サーバーを起動したときは、デフォルトのユーザー名を使用します。TN3270 接続に領域を設定する方法については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。 |

**注 –** そのユーザーの端末に対して、`CESF/CSSF` または `EXEC CICS SIGNOFF` を実行すると、すべての通信パスのすべてのトランザクションに対して、デフォルトのユーザー名が使用されます。

# Sun MTP Secure リソースクラスタイプ

次の表に、Sun MTP Secure リソースクラスタイプを示します。このリソースクラスタイプは、Sun MSF のセキュリティーリポジトリに定義されているリソースタイプに対応します。

**表 8-1** Sun MTP Secure リソースクラス

| リソースクラスタイプ | Sun MSF リソースタイプ | 説明 |
|---|---|---|
| KIX-FILES | KIX_FILE | VSAM ファイル。指定した VSAM ファイルに、どのユーザーがアクセスできるかを制御します。 |
| KIX-PROGRAMS | KIX_PROGRAM | Sun MTP アプリケーション。アプリケーションが LINK、XCTL または LOAD コマンドを使用して呼び出す、指定されたアプリケーションプログラムへのアクセスを制御します。 |
| KIX-JOURNALS | KIX_JOURNAL | Sun MTP ジャーナル。指定したジャーナルに、どのユーザーがアクセスできるかを制御します。 |
| KIX-COMMANDS | KIX_COMMAND | コマンドセキュリティー確認の対象である Sun MTP 管理コマンドのサブセット。表 8-4 の定義に従い、指定された管理コマンドにどのユーザーがアクセスできるかを制御します。 |
| KIX-START-TRANS | KIX_START_TRANS | 起動されたトランザクション。指定されたトランザクションを、どのユーザーが EXEC CICS START を使用して起動できるかを制御します。 |
| KIX-ATTACH-TRANS | KIX_ATTACH_TRANS | 端末が接続されたトランザクション。指定された端末からのトランザクションを、どのユーザーがサブミットできるかを制御します。 |
| KIX-TD-QUEUE | KIX_TDQUEUE | Sun MTP パーティション外およびパーティション内一時データの転送先のキュー。一時データキュー (TDQ) とも呼ばれます。指定した TDQ 名に、どのユーザーがアクセスできるかを制御します。 |
| KIX-TS-QUEUE | KIX_TSQUEUE | 一時記憶域の転送先キュー。指定した一時記憶域キュー名に、どのユーザーがアクセスできるかを制御します。 |
| KIX-TERMINALS | KIX_TERMINAL | Sun MTP 端末。指定した端末名に、どのユーザーがアクセスできるかを制御します。 |
| UNIX-APPLS | KIX_REGION | アクセスを制御する領域。この値は、領域の $KIXSYS ディレクトリパス名です。領域で起動、終了、実行できるユーザーを制御します。 |

# トランザクションとリソースのセキュリティーのための Sun MTP Secure の使用

デフォルトユーザー $KIXSECDFLTUSER を含め、各ユーザーのアクセス権は、ESM で領域のリソースクラス (トランザクション、VSAM ファイル、一時データ、一時記憶域キュー、端末、ジャーナル、およびプログラム) に対して、アクセス権 (読み取り、書き込み、実行、など) を定義することによって構成されます。

領域が Sun MTP Secure を使用している場合、SNT と PCT のセキュリティーキーのフィールドは無視されます。代わりに、各ユーザー名とパスワードは ESM で設定されます。対応する SNT エントリがそのユーザー名について設定されている場合、オペレータ名、オペレータ ID、オペレータクラスなどのその他のフィールドは、ユーザーが Sun MTP Secure からサインオンして確認されるときに、TCT にコピーされます。これらのフィールドは、IBM の RACF が CICS 拡張レコードで提供する値に対応します。

---

**注 –** Sun MTP ユーザーごとに SNT エントリは必要ありません。そのユーザーにエントリが存在しない場合、それらの TCT フィールドは空のままです。

---

このユーザー名は、次に、ユーザーがサブミットしたトランザクションとそのトランザクションがアクセスするすべてのリソースのセキュリティー確認に使用されます。たとえば、端末が接続されたトランザクションのセキュリティーが有効な場合 (KIXPCTSEC=YES)、ユーザーがトランザクションをサブミットすると、PCT のトランザクション名 (Trans ID) は、ユーザー名とともに、KIX-ATTACH-TRANS リソースクラスで外部セキュリティーマネージャーに Sun MTP Secure からサブミットされます。

Sun MTP Secure が、トランザクションへのユーザーアクセスについて ESM から拒否を受け取ると、TRANSACTION NOT AUTHORIZED FOR USER メッセージが表示されます。

このユーザーが別の特権を持つ別のユーザーとしてサインオンするには、CESN トランザクションに有効なユーザー名とパスワードを指定して入力します。

ユーザーの接続に関して特定のユーザー名が確認されていない場合、領域は KIXSECDFLTUSER 環境変数で定義されたデフォルトのユーザー名を使用します。そのデフォルトのユーザー名のアクセス権は、ESM で設定されています。$KIXSECDFLTUSER に関する詳細は、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』の Sun MTP Secure 環境変数に関する節を参照してください。

**注 –** 端末が接続されたトランザクションのセキュリティーが有効な場合 (KIXPCTSEC=YES)、すべての Sun MTP システムトランザクションに必要なアクセス権が、ESM のセキュリティーリポジトリに設定されている必要があります。詳細は、184 ページの「Sun MTP システムトランザクションのセキュリティー」を参照してください。

## リソースセキュリティーの管理

領域が Sun MTP Secure を使用する場合、トランザクションに対するアクセスに加えて、トランザクションによるリソースへのアクセスを制御できます。これらのリソースは、定義済みのリソースクラスタイプにあるその識別情報ごとに、外部セキュリティーマネージャーで構成されます。また、リソースクラスタイプによるアクセス確認は、対応する Sun MTP Secure 環境変数で有効と無効が切り替えられます。

| リソースクラスタイプ | 説明 | Sun MTP Secure 環境変数 | Sun MTP テーブル |
|---|---|---|---|
| KIX-ATTACH-TRANS | サブミットされたトランザクション名 | KIXPCTSEC | PCT |
| KIX-FILES | VSAM データセット名 | KIXFCTSEC | FCT |
| KIX-PROGRAMS | アプリケーションプログラム名 | KIXPPTSEC | PPT |
| KIX-JOURNALS | ジャーナル ID | KIXJCTSEC | JCT |
| KIX-START-TRANS | 起動したトランザクション名 | KIXSTTSEC | PCT |
| KIX-TD-QUEUE | 一時データキュー名 | KIXDCTSEC | DCT |
| KIX-TS-QUEUE | 一時記憶域キュー名 | KIXTSTSEC | TST |
| KIX-TERMINALS | 端末名 | KIXTCTSEC | TCT |
| KIX-COMMANDS | SET/INQUIRE/PERFORM によってアクセス可能な内部 Sun MTP データ | KIXCMDSEC | n/a |
| UNIX-APPLS | アクセスを制御する領域 | KIXAPPSEC | n/a |

領域は、個々のリソースのアクセスについて Sun MTP Secure を呼び出し、その名前、リソースクラス、トランザクションを実行しているユーザー名、および必要なアクセス権を示します。各アクセス操作に必要なアクセス権について、次の表に示します。

**表 8-2** EXEC CICS コマンドのアクセス定義

| リソースクラスタイプ | EXEC CICS コマンド | 必要なアクセス権 |
|---|---|---|
| KIX-FILES | READ | 読み取り |
| | STARTBR* | 読み取り |
| | WRITE | 書き込み |
| | DELETE | 書き込み |
| | REWRITE | 書き込み |
| KIX-PROGRAMS | LOAD | 読み取り |
| | XCTL | 実行 |
| | LINK | 実行 |
| KIX-JOURNALS | JOURNAL | 書き込み |
| KIX-START-TRANS | START | 実行 |
| | DELAY | 実行 |
| | RETRIEVE | 読み取り |
| | CANCEL | 削除 |
| KIX-TD-QUEUE | READQ | 書き込み |
| | WRITEQ | 書き込み |
| | DELETEQ | 書き込み |
| KIX-TS-QUEUE | READQ | 読み取り |
| | WRITEQ | 書き込み |
| | DELETEQ | 書き込み |
| KIX-TERMINALS | - | 読み取り |
| KIX-COMMANDS | INQUIRE | 読み取り |
| | SET | 書き込み |

* その他の VSAM ブラウズ操作 (ENDBR、READNEXT、READPREV、および RESETBR) では、アクセス権の確認は行われません。これらを使用するには STARTBR が正常に実行される必要があるためです。

次の表は、残りの Sun MTP Secure リソースクラスに必要なアクセス権について説明します。

**表 8-3**　その他のアクションについてのアクセス定義

| リソースクラスタイプ | アクション | 必要なアクセス権 |
|---|---|---|
| KIX-ATTACH-TRANS | 新しいトランザクションのサブミット | 実行 |
| UNIX-APPLS | 有効な Sun MTP 領域へのアクセス許可 | 実行 |
| | その領域の起動 | 作成 |
| | Sun MTP 領域の終了。次に例を示します。<br>• CEMT PERFORM SHUTDOWN<br>• CSMT SHUT,YES<br>• 開発システムの PF3 キーの使用 | 削除 |

## KIX-FILES リソースクラスの使用

KIX-FILES リソースクラスは VSAM カタログへのアクセスを制御します。これは、CATALOG という名前の VSAM データセットです。CATALOG データセットに対する KIX-FILES 読み取り権を付与または拒否することによって、ユーザーが CFMS トランザクションを使用して VSAM カタログ内の VSAM ファイル定義を表示する権限が付与されたり、拒否されたりします。CATALOG データセットでの KIX-FILES 書き込み権を付与または拒否することによって、CFMS トランザクションを使用して、VSAM カタログで定義されている VSAM ファイルを作成、削除、変更する権限がユーザーに付与されたり、拒否されたりします。

バッチプログラムがアクセスする VSAM ファイルは、Sun MTP Secure によって制御されません。

## KIX-COMMANDS リソースクラスの使用

領域は Sun MTP Secure を使用して、CEMT トランザクションごとに、または EXEC CICS INQUIRE/PERFORM/SET API を使用するアプリケーションごとに内部管理リソースへのアクセスも制御します。これら管理コマンドの示すリソースクラスは、KIX-COMMANDS です。次の表は、KIX-COMMANDS クラスのリソース名を特定の EXEC CICS と CEMT 操作にマップしています。すべての INQUIRE 機能には読み取り権が、すべての SET 機能には書き込み権が必要です。

これらのコマンドのほとんどは、EXEC CICS と CEMT インタフェースの両方に共通です。しかし、いずれかのインタフェースに対して固有のコマンドの場合、このコマンドの前には、EXEC CICS と CEMT のどちらかが置かれます。

表 8-4　セキュリティー確認の対象である KIX-COMMANDS リソース

| リソース名 | EXEC CICS/CEMT コマンド | アクセス権 |
|---|---|---|
| TASK | EXEC CICS TASK LIST ... | 読み取り |
| PROGRAM | INQUIRE/SET PROGRAM (*name*) | 読み取り/書き込み |
| TERMINAL | INQUIRE/SET TERMINAL (*name*) | 読み取り/書き込み |
| CONNECTION | CEMT INQ/SET CONNECTION ... | 読み取り/書き込み |
| TDQUEUE | INQUIRE/SET TDQUEUE (*name*) | 読み取り/書き込み |
| TSQUEUE | INQUIRE TSQUEUE | 読み取り |
| TRANSACTION | EXEC CICS INQUIRE TRANSACTION *(name)* | 読み取り |
| | CEMT INQ/SET TASK ... | 読み取り/書き込み |
| | CEMT INQ TRANID ... | 読み取り |
| | CEMT INQ FACILITY | 読み取り |
| | CEMT INQ ACTIVE | 読み取り |
| | CEMT INQ SUSPENDED | 読み取り |
| FILE | EXEC CICS INQUIRE/SET FILE (*name*) | 読み取り/書き込み |
| REQID | EXEC CICS INQUIRE/SET REQID | 読み取り/書き込み |
| TRANCLASS | INQUIRE/SET TRANCLASS | 読み取り/書き込み |
| SECURITY | CEMT PERFORM SECURITY ... | 書き込み |
| SHUTDOWN | CEMT PERFORM SHUTDOWN* | 書き込み |
| SYSTEM | SET SYSTEM ... | 書き込み |
| | EXEC CICS INQUIRE SYSTEM | 読み取り |

* このコマンドリソースは、領域を終了する方法の代わりとなる同等の Sun MTP 内部の方法、つまり CSMT SHUT,YES と「Development System」メインメニューの PF3 キーを使用する方法 (CMNU) についても確認・実施されます。領域の停止の詳細は、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

## 入口セキュリティーの使用

入口セキュリティーは、特定の Sun MTP 領域のユーザーと特定の端末名が領域にアクセスするのを許可または拒否します。

UNIX-APPLS リソースクラスは、ESM で設定すれば、その `$KIXSYS` 値に基づいて、特定の領域に接続しようとしたユーザーアクセスの許可または拒否を実行できます。また、ESM は、ユーザーが確立する接続に対して指定または取得された端末名が、そのユーザーに使用が認められているかどうかを確認します。
KIX-TERMINALS リソースクラスは、ESM で構成されて、特定の端末名に対するユーザーの一連のアクセスを許可または拒否します。

リソース確認は、`KIXAPPSEC` 環境変数を `NO` に設定すれば無効にできます。Sun MTP Secure の環境変数の詳細については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

ESM は、領域を起動または終了するために、ユーザーに UNIX-APPLS リソースに対するアクセス権があることを確認します。ユーザーに `kixstart` コマンドで領域を起動するアクセス権がない場合、次のメッセージが表示されます。

```
KIX0145F Userid not authorized to start this Sun MTP (UNIX-APPLS permission)
```

**注 –** `kixclean` は、その領域の UNIX-APPLS 権を持っているユーザーがすべて起動できるわけではなく、Sun MTP を起動したユーザー名のみが起動可能です。

## 起動プログラムと停止プログラム

起動プログラムまたは停止プログラムをプログラムリストテーブル (PLT) で定義している場合は、これらのプログラムを呼び出すユーザー ID (領域管理者) に、そのプログラムの実行権限があることを確認する必要があります。タイプが `SYSSTART` であると PLT で定義されている起動プログラムは、`PLT1` という Sun MTP のトランザクション識別子で実行されます。このため、セキュリティーリポジトリで KIX-ATTACH-TRANS というリソースタイプに対して `PLT1` トランザクションを定義する必要があります。領域を起動するユーザーは、`PLT1` トランザクションの実行権限と、そのトランザクションがアクセスするすべてのリソースの適切な権限を持っている必要があります。同じように、タイプが `SYSTERM` であると PLT で定義されている停止プログラムは、`CEMT` というトランザクション識別子で実行されます。領域を停止するユーザーは、`CEMT` トランザクションの実行権限と、そのトランザクションがアクセスするすべてのリソースの適切な権限を持っている必要があります。

## リソース名への接頭辞の追加

Sun MTP Secure では、オプションとしてすべてのリソース名の「接頭辞を付ける」ことができます。同じ ESM で複数の Sun MTP 領域が管理されている場合、各領域のリソースは一意に修飾されている必要があります。たとえば、ユーザー A、B、C に Test 領域の PAYROLL データセットへのアクセス権があっても、Prod 領域のデータセットに対するアクセス権はないことがあります。これは、このデータセットが実際には 2 つの異なる VSAM ファイルであるためです。

接頭辞の追加は、KIXSECPREFIX 環境変数で有効にします。YES に設定すると、すべてのリソース名には、領域を起動した UNIX ユーザー ID のあとにピリオドが追加された接頭辞が付きます。上記の PAYROLL データセットの例では、ユーザー A、B、C は ESM によって確認が行われ、ESM リポジトリの設定に従って、mtptest.PAYROLL データセットへのアクセスは認証されますが、mtpprod.PAYROLL データセットへのアクセスは拒否されます。

## ESM 結果のログ記録

Sun MTP Secure は、ESM ログインおよびアクセス確認の結果を領域の unikixmain.log ファイルに記録します。書き込む結果のタイプは、KIXSEC_LOGGING 環境変数で定義します。この環境変数を ALL に設定すると、すべての ESM 結果、つまり成功と失敗の両方を記録します。DENIALS に設定すると、ESM の拒否した結果のみを記録します。NONE は、ログ記録を無効にします。

---

**注 –** このログ記録は、ESM 監査ログ記録に追加するものです。ESM ログではわからない、特定の領域のイベントを監査して記録するのに便利です。

---

## セキュリティーアクセス結果のキャッシュ

外部セキュリティーマネージャーが使用中の場合、ユーザーがさまざまなリソースにアクセスしようとすると、そのアクセスの結果が、結果キャッシュと呼ばれる領域のキャッシュメモリーに保存されます。結果キャッシュは、同じリソースへの連続アクセスに使用し、アクセス時間を短縮します。キャッシュ領域は、ユーザーがログオフするとクリアされます。

セキュリティーリポジトリのルールが変更され、それらのルールが領域のユーザーに影響する場合、次のシステムトランザクションを実行する必要があります。これにより、領域の結果キャッシュがクリアされ、ユーザーによるリソースへのアクセス時に新しいルールが照会されます。

```
CEMT PERFORM SECURITY REBUILD
```

第9章

# アカウンティング

アカウンティングとは、ユーザーアカウント情報を体系的に収集、記録、解釈、および表現する手順です。オペレーティングシステムには、通常、課金、パフォーマンス管理、容量計画、およびコンピュータシステムの正しい使用を保証するためのユーザーアクティビティー追跡機能が用意されています。

この章では、次のトピックについて説明します。



オペレーティングシステムのアカウンティングでは、プロセスの実行、ファイルアクセス、およびログオンのようなユーザーアクションに基づいて、リソース利用状況に関する情報が取得できます。サン以外のパッケージの場合、情報処理の程度はさまざまですが、一般に、基本的なオペレーティングシステムのアカウンティングよりも多くの情報と機能を提供します。トランザクションサーバーのような一部の製品は、多くの個別のユーザーに対して機能を提供しているので、独自のアカウンティング機能を必要とします。

## UNIX アカウンティング

UNIX 環境では、各ユーザーアカウントについて収集される情報は、通常、コンピュータリソースに関する情報です。ほとんどのシステムは、プロセスアカウンティングオプションを提供します。これを有効にすると、プロセスが終了するたびに、カーネルによってアカウンティングレコードが書き込まれます。

アカウンティングについては、オペレーティングシステムのマニュアルを参照してください。

# Sun 以外のアカウンティングパッケージを使用する場合

オペレーティングシステムのアカウンティングがサイトの必要条件を一部満たしていても、ネットワーク環境にある 1 つ以上のシステム上で、多くのユーザーとアプリケーションが作業を続けている実稼動サイトでは、より高度なアカウンティング機能が必要です。高度なアカウンティングパッケージは、情報の収集に加え、システムの使用状況を分析してレポートを作成することによって、パフォーマンス管理とキャパシティー計画を支援します。

- パフォーマンスの管理は、現在の構成を円滑に機能させることに関係しています。
- キャパシティー計画は、現在のリソースと将来の要求の把握に関係しています。

アプリケーションが複数のプラットフォーム (メインフレーム、UNIX システム、PC) 上で動作するので、包括的なキャパシティー計画には、すべてのプラットフォームを考慮する必要があります。Sun 以外のパッケージのプログラミングによって、多くの異なるファイル形式を認識し、キャパシティー計画のためのデータを抽出できます。通常、これらの統計パッケージは、サイトの必要条件に合わせたプログラミングが必要です。

# Sun MTP アカウンティング

Sun MTP は、1 人以上のユーザーのために処理を行うトランザクションサーバーです。Sun MTP Transaction Server は、専用のユーザー ID の下で実行されます。トランザクションで使用されるリソースについての情報が必要な場合、Sun MTP は、トランザクション処理に使用するユーザー ID とシステムリソースが入ったアカウンティングレコードを生成する必要があります。

実行される個々のトランザクションについて、アカウンティングレコードを生成する必要があるかどうかを指定できます。アカウンティングレコードを生成する場合の条件を指定し、特定のユーザー ID またはトランザクション自体に応じて、アカウンティングレコードを生成するかどうかを指定できます。

すべてのアカウンティングレコードは、アカウンティングジャーナルに書き込まれます。アカウンティングジャーナルは 1 つ以上指定して、それぞれのアーカイブ方法を制御できます。

# アカウンティングの有効化

領域にアカウンティングを有効にする手順は次のとおりです。各タスクの詳細については、参照先の節をご覧ください。

## ▼ アカウンティングを有効化する

1. **アカウンティングを有効にする領域を起動します。**
2. **次の各テーブルに、適切なアカウンティングオプションを指定します。**
   - システム初期化テーブル (SIT)
   - 監視管理テーブル (MCT)
   - プログラム管理テーブル (PCT)
   - サインオンテーブル (SNT)

   このオプションは、アカウンティングレコードを生成する方法とタイミングを制御します。また、ファイル ID によって、アカウンティングに使用するジャーナルも指定できます。208 ページの「アカウンティングオプション」 を参照してください。
3. **ジャーナル管理テーブル (JCT) で、手順 2 で指定したファイル ID を入力します。**

   これにより、ファイル ID がファイルにマップされ、ジャーナルサイズが指定できます。
4. **アカウンティングジャーナルがファイルサイズに達したら、それを保管するよう、`kixjournal` シェルスクリプトを変更します。**

   詳細は、227 ページの「`kixjournal` シェルスクリプト」を参照してください。
5. **領域を停止し、再起動して、ステップ 2 および 3 で設定したオプションを起動します。**
6. **必要に応じて `kixjas` プログラムを使用し、アカウンティングジャーナルの raw データを ASCII ファイル形式に変換します。**

   詳細は、216 ページの「`kixjas` アカウンティング変換プログラムの使用」を参照してください。

# アカウンティングオプション

Sun MTP では、グローバルレベル、トランザクションレベル、およびユーザーレベルのアカウンティングをシステムテーブルで構成することができます。アカウンティングオプションの階層を 図 9-1に示します。

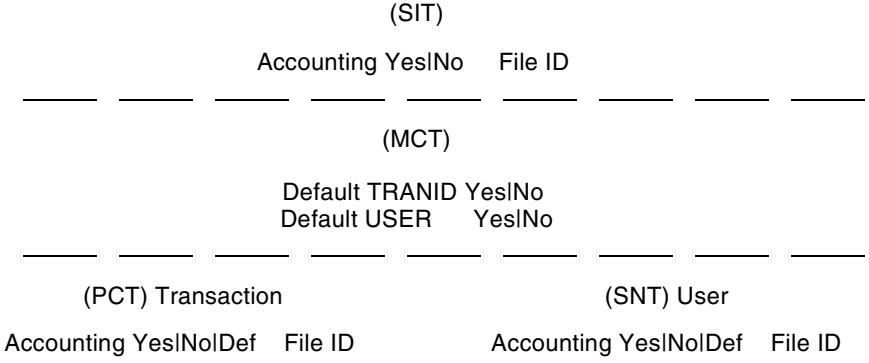


**図 9-1** アカウンティングオプションの階層

SIT はアカウンティング階層の最上位です。SIT のアカウンティングフィールドが N に設定されている場合、アカウンティングはほかの設定に関係なく実行されません。Y に設定されている場合、次回起動時からその領域に対するアカウンティングが有効になります。デフォルトの ファイル ID として使用される 2 桁のファイル ID を指定する必要があります。ファイル ID を JCT で指定しない場合、デフォルトのジャーナルファイルが `JRNL`*xx* という形式で作成されます。この *xx* はデフォルトのファイル ID です。

アカウンティング階層の次のレベルにある MCT は、ユーザー (SNT) またはトランザクション (PCT) に基づいてアカウンティングレコードを生成するためのデフォルト値を示します。デフォルトの設定により、現在のニーズに基づいて、ユーザー ID とトランザクションを含めたり除外したりできます。

アカウンティングオプションを指定する場合は、次のガイドラインに従います。

- SIT で Accounting オプションを Y(es) に、MCT で Default TRANID Accounting オプションを Y(es) に設定し、その他の PCT のオプションは設定しない場合、処理するすべてのトランザクションに対し、1 つのアカウンティングレコードを生成します。
- SIT で Accounting オプションを Y(es) に、MCT で Default TRANID Accounting オプションを N(o) に設定し、その他の PCT のオプションは設定しない場合、処理するすべてのトランザクションに対してアカウンティングレコードは生成されません。

- 特定のトランザクション ID をアカウンティングに含めるには、PCT で Trans ID のアカウンティングオプションに対して Y(es) を指定するだけです。

  同じガイドラインが、Default User Accounting オプションにも適用されます。

これらのデフォルトのオプションを設定する際には、必要以上のアカウンティング情報が生成される場合があるため、注意が必要です。たとえば、SIT の Accounting オプションを Yes 、Default TRANID Accounting オプションを Yes 、Default User Accounting オプションを Yes に設定し、その他のアカウンティングオプションは設定しない場合、アカウンティングレコードは、処理されるすべてのトランザクション、およびユーザーがサブミットするすべてのトランザクションについて生成されます。これにより、各トランザクションに対し、アカウンティングレコードが 2 つずつ生成されます。

PCT と SNT は、アカウンティング階層の最下位のレベルにあります。これらの表では、個別のトランザクションおよびユーザーについてアカウンティングオプションを設定できます。

PCT の Acct フィールドでは、特定の Trans ID に対するアカウンティングを指定できます。

- PCT で Y(es) を指定し、SIT の Accounting オプションを Y(es) に設定すると、トランザクションが処理されるたびに、アカウンティングレコードが生成されます。
- PCT で N(o) を指定すると、トランザクションについてアカウンティングレコードは生成されません。
- PCT で D(efault) を指定すると、SIT の Accounting オプションが Y(es) に、MCT の Default TRANID オプションが Y(es) に設定されている場合、トランザクションにアカウンティングレコードが生成されます。

---

**注 –** PCT にトランザクションを追加すると、アカウンティングが D(efault) に設定されます。

---

SNT には、システムサインオントランザクション `CESN` が使用される特定のユーザー関連情報が入っています。SNT でアカウンティングオプションを指定すると、ユーザー ID に基づいてアカウンティング情報を生成できます。

- SNT で Y(es) を指定し、SIT の Accounting オプションを Y(es) に設定すると、ユーザーのためにトランザクションが実行されるたびに、アカウンティングレコードが生成されます。
- SNT で N(o) と指定すると、指定したユーザー ID のために実行されたトランザクションのアカウンティングレコードは生成されません。
- SNT で D(efault) を指定すると、SIT のアカウンティングオプションが Y(es) に、MCT の Default User オプションが Y(es) に設定されている場合、そのユーザーのために実行されたトランザクションにアカウンティングレコードが生成されません。

**注 –** ユーザーを SNT に追加すると、アカウンティングは自動的に D(efault) に設定されます。

わかりにくいケースもいくつかあります。表 9-1 に、設定とその結果を示します。

- SIT の Accounting オプションを N(o) に設定すると、アカウンティングレコードは生成されません。
- 「アカウンティングオプション」のセクションは、システムテーブルで考えられるアカウンティングオプションの設定を示しています。User1、User2、および User3 は、SNT で定義された具体的なユーザーを表しています。Tran1 は、PCT で定義されたトランザクションを表しています。
- 「結果」のセクションは、ユーザーまたはトランザクションあるいはその両方に対して、アカウンティングレコードが生成されるかどうかを示しています。
- 最初のエントリの結果は、次のアカウンティングレコードが生成されることを示しています。
  - User1 と User3、それらのユーザーのために実行されたすべてのトランザクション。
  - トランザクション Tran1 (実行されるたびに生成)。User1 または User3 が Tran1 を実行した場合は、2 つのアカウンティングレコードが生成されます。

**表 9-1** アカウンティングオプションと結果

| アカウンティングオプション | | | | | | | 結果 | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| SIT | MCT | | SNT | | | PCT | | | | |
| | TRANID | User | User1 | User2 | User3 | Tran1 | User1 | User2 | User3 | Tran1 |
| Yes | Yes | Yes | Yes | No | Def | Yes | On | Off | On | On |
| Yes | No | Yes | Yes | No | Def | Yes | On | Off | On | On |
| Yes | Yes | No | Yes | No | Def | Yes | On | Off | Off | On |
| Yes | No | No | Yes | No | Def | Yes | On | Off | Off | On |
| Yes | Yes | Yes | Yes | No | Def | No | On | Off | On | Off |
| Yes | No | Yes | Yes | No | Def | No | On | Off | On | Off |
| Yes | Yes | No | Yes | No | Def | No | On | Off | Off | Off |
| Yes | No | No | Yes | No | Def | No | On | Off | Off | Off |
| Yes | Yes | Yes | Yes | No | Def | Def | On | Off | On | On |
| Yes | No | Yes | Yes | No | Def | Def | On | Off | On | Off |
| Yes | Yes | No | Yes | No | Def | Def | On | Off | Off | On |
| Yes | No | No | Yes | No | Def | Def | On | Off | Off | Off |
| No | 任意 | 任意 | 任意 | 任意 | 任意 | 任意 | Off | Off | Off | Off |

# アカウンティングジャーナル

アカウンティングジャーナルは、アカウンティングレコードが書き込まれるファイルです。アカウンティングオプションを指定するとき、アカウンティングジャーナルをそれに関連付けます。

JCT でジャーナルファイルと関連付けられているファイル ID にオプションを関連付けます。

- SIT で ファイル ID を指定し、JCT、PCT、または SNT で ファイル ID を入力しない場合、すべてのアカウンティングレコードは、SIT で指定したデフォルトのファイル ID に書き込まれます。
- SIT の Accounting オプションが Yes の場合は、JCT にファイル ID を指定する必要があります。指定しない場合は、ファイル ID が指定されていないことを示すメッセージが表示され、デフォルトのジャーナル (`JRNL`*xx*。ここで *xx* は ファイル ID) が作成されます。
- SIT Accounting オプションに ファイル ID を指定しただけの場合、ファイル ID が関連付けられたジャーナルに、すべてのアカウンティングレコードが書き込まれます。

また、JCT でジャーナルファイルのサイズも指定できます。JCT に定義されているすべてのジャーナルサイズには、代替ファイルが関連付けられています。代替ファイルは、ジャーナルファイルと名前は同じですが、`.jnl` という接尾辞が付いています。元のジャーナルファイルがそのサイズの限界に達すると、元のファイルは代替ファイルに移動されます。元のファイルはクリアされ、アカウンティングレコードは元のジャーナルファイルに書き込まれます。

ジャーナルファイルが代替ファイルに移動されると、代替ファイル名を引数として、`kixjournal` シェルスクリプトが呼び出されます。標準のシェルスクリプトは、代替ファイルに対して、まったく機能を実行しません。しかし、サイトの要請に応じて、ジャーナルファイルを処理するようシェルスクリプトを変更できます。たとえば、ジャーナルファイルをあとで処理できるよう、テープに書き込むこともできます。詳細は、227 ページの「`kixjournal` シェルスクリプト」を参照してください。

---

**注 –** 代替ファイルの処理に十分な時間が取れるよう、ジャーナルファイルは、数時間分のアカウンティングレコードを保存できるだけの大きさが必要です。

---

アカウンティングレコードを別のジャーナルに割り当てることができます。アカウンティングレコードが送られるジャーナルファイルが「File ID」フィールドによってどのように決まるかを表 9-2 に示します。

次の表に、Default User、User1、User2、User3、および User4 と、Default TRANID、Tran1、Tran2、Tran3、および Tran4 に対して設定されるアカウンティングオプションを示します。この表ではまた、アカウンティングレコードを書き込むジャーナルを示します。

**表 9-2** ジャーナルへのアカウンティングレコードの割り当て

| テーブル | アカウンティング オプション | | ファイル ID | ジャーナル | ジャーナルの内容 |
|---|---|---|---|---|---|
| SIT | Accounting | Yes | 03 | JOURNL03 | Tran3 と Tran4 のアカウンティングレコード。Trans ID の PCT エントリでファイル ID が指定されておらず、JOURNL03 が SIT で指定されているアカウンティングジャーナルであるため、これらのレコードは、このジャーナルに記録されます。<br>Tran3 のアカウンティングは Def に設定されます。TRANID の MCT の Accounting オプションが Yes なので、アカウンティングレコードが生成されます。 |
| MCT | Default TRANID | Yes | | | |
| MCT | Default User | No | | | |
| SNT | User1 Accounting | Yes | 04 | JOURNL04 | User1 のために実行されたすべてのトランザクションのアカウンティングレコード。たとえば、Tran1、Tran2、Tran3、または Tran4 です。ここでは、2 つのアカウンティングレコードが生成され、それぞれ異なるジャーナルに書き込まれます。つまり、User1 が Tran1 を実行する場合、トランザクションレコードが JOURNL07 に、ユーザーレコードが JOURNL04 に書き込まれます。 |
| SNT | User2 Accounting | Yes | 05 | JOURNL05 | User2 のために実行されるすべてのトランザクションのアカウンティングレコード。 |
| SNT | User3 Accounting | No | | | |
| SNT | User4 Accounting | Yes | 06 | JOURNL06 | User4 のために実行されるすべてのトランザクションのアカウンティングレコード。 |

表 9-2　ジャーナルへのアカウンティングレコードの割り当て (続き)

| テーブル | アカウンティング オプション | | ファイル ID | ジャーナル | ジャーナルの内容 |
|---|---|---|---|---|---|
| PCT | Tran1 Accounting | Def | 07 | JOURNL07 | トランザクション Tran1 と Tran2 のアカウンティングレコード。 |
| PCT | Tran2 Accounting | Def | 07 | JOURNL07 | |
| PCT | Tran3 Accounting | Def | | | |
| PCT | Tran4 Accounting | Yes | | | |

サイトの要件に応じて、アカウンティングレコードを特定のジャーナルに保存できます。ユーザージャーナルレコード (アプリケーションによって作成されるレコード) と領域のアカウンティング レコードは同じファイルに混在させるのではなく、それぞれ別のジャーナルファイルに書き込みます。

# アカウンティングジャーナルの形式

次の表は、アカウンティングジャーナルファイルを説明するものです。これには、次の 3 種類のレコードが入っています。

表 9-3　アカウンティングジャーナルファイルのレコード

| レコード | 説明 |
|---|---|
| Physical Write Header Record (PWHR) | PWHR は C 構造体で、ジャーナルファイルに書き込まれた物理ブロックを表します。この構造体の定義は、$UNIKIX/src/trans ディレクトリのファイル statrcrd.h の中で定義されます。構造体名は、jct_rcd です。 |
| Accounting Header Record (AHR) | AHR は、処理中のデータのタイプと、レコードが作成された状況を記述した C 構造体です。この構造体の定義は、$UNIKIX/src/trans ディレクトリの statrcrd.h ファイルの中で定義されます。構造体名は、acntg_hdr_type です。 |
| Accounting Data Record | AHR のタイプが user または tranid の場合、Accounting Data Record は、必ず AHR のあとにあります。Accounting Data Record の構造体は $UNIKIX/src/trans/statrcrd.h ファイルの構造体 acntg_body_type で定義されています。 |

アカウンティングジャーナルには、次の 6 種類の AHR が使用されます。

**表 9-4** Accounting Header Record (AHR) のタイプ

| AHR タイプ | 使用される場面と場所 |
|---|---|
| 0005 | すべてのジャーナルファイルの始め。 |
| 0006 | 領域またはシステムがクラッシュし、領域を起動したあと、すべてのアカウンティングファイルに最初に書き込まれるレコードとしてジャーナルに追加されます。 |
| 0015 | ジャーナルサイズを使い果たしたとき、ジャーナルの最後に追加されます。 |
| 0016 | 領域が通常の停止を行なったとき、最後のレコードとして追加されます。 |
| 0002 | アカウンティングレコードがユーザー ID のために書き込まれたとき。これには、SNT で指定されたオプションに基づくあらゆる状況が含まれます。このレコードのあとには、ユーザー ID に関する特定の情報を持つ Data Record が続きます。 |
| 0001 | アカウンティングレコードがトランザクションのために書き込まれたとき。これには、PCT で指定されたオプションに基づくあらゆる状況が含まれます。このレコードのあとには、Trans ID に関する特定の情報を持つ Data Record が続きます。 |

PWHR のあとには常に、AHR タイプ 0005、0006、0016、および 0015 が存在します。また、1 つまたは複数の AHR タイプ 0001 と 0002 もあとに存在します。PWHR のフラグは、レコードのあとのデータが Sun MTP アカウンティング情報であることを示します。次の図に、アカウンティングジャーナルレコードの形式を示します。

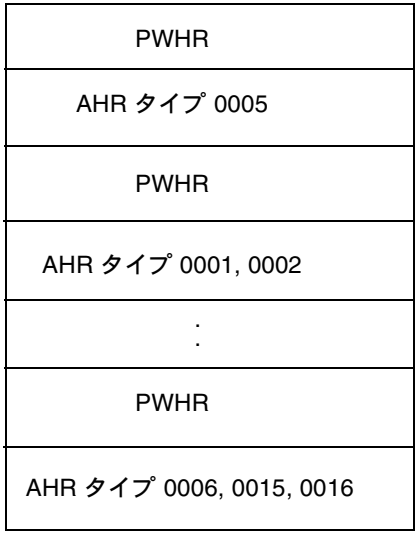


**図 9-2** アカウンティングジャーナルレコードの構造

Sun MTP アカウンティングレコードには、次の規則が適用されます。

- Sun MTP アカウンティング情報を指定する PWHR のあとで、AHR が生成されます。
- AHR タイプ 0005 は、すべてのジャーナルファイルの最初のレコードとして生成されます。これは、最初のトランザクションまたはユーザー ID Data Record を書き込む必要があると領域が識別したときに発生します。このレコードの前に、PWHR が付いています。
- AHR タイプ 0015 は、ファイルがそのサイズの上限まで使い果たされたときに、ジャーナルファイルに書き込まれる最後のレコードです。このとき、ファイルはその代替ファイルに移動され、AHR タイプ 0005 が、PWHR のあとに、オリジナルジャーナルファイルの最初のレコードとして書き込まれます。
- 領域またはオペレーティング環境がクラッシュしたときは、ジャーナルファイルにレコードが書き込まれません。
- 領域の起動時に、AHR タイプ 0006 レコードは、前に PWHR を伴って、すべての既存のジャーナルファイルのあとに追加されます。最後にファイルサイズの条件に達するか、領域が停止するまで、ジャーナルファイルに次々とレコードが追加されます。
- 通常の領域停止の場合は、すべてのジャーナルファイルに、AHR タイプ 0016 が、前に PWHR を伴って書き込まれます。さらに、すべてのジャーナルファイルがその代替ファイルに移動され、`kixjournal` シェルスクリプトが各代替ファイルに対して呼び出されます。
- メッセージを `unikixmain.log` ファイルに書き込みます。これらのメッセージは、ジャーナルファイルを使い果たしたこと、`kixjournal` シェルスクリプトを呼び出されたとき、およびファイル ID が JCT で定義されていないことを、その他の情報メッセージとともに示します。

# アカウンティングジャーナルの回復

システムまたは領域のクラッシュが発生すると、アカウンティングジャーナルはクラッシュの時点のまま残ります。ジャーナルの最後のレコードは、常に完全なアカウンティングレコードです。ジャーナルには、部分的なレコードはありません。領域が再起動すると、クラッシュの時点でアクティブだったアカウンティングジャーナルの終わりに、AHR タイプ 0006 が書き込まれます。AHR タイプ 0006 は、システムまたは領域のクラッシュが発生したことを示します。

システムまたは領域のクラッシュの時点で、アカウンティングレコードのすべてがそれぞれのジャーナルに書き込まれないこともあります。これらの内部バッファーに入っているレコードは失われます。

# kixjas アカウンティング変換プログラムの使用

ジャーナルファイルのアカウンティングレコードには、ジャーナルファイルとアカウンティングデータに固有で、ASCII とバイナリデータが混在したヘッダー情報が入っています。変換プログラム kixjas は、ジャーナルファイルを読み込んで、ヘッダー情報を除去する以外は同じ情報を持つ ASCII ファイルを出力します。 kixjas は、Transid レコード (レコードタイプ 0001) または User レコード (レコードタイプ 0002) という特定のタイプのアカウンティングレコードを抽出します。kixjas プログラムについては、『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。

kixjas は、各ジャーナルファイルを処理したあと、それぞれのジャーナルファイルで処理したレコードタイプの合計数を示すレポートを生成します。ジャーナルファイルがすべて処理されると、kixjas により、次の例に示す形式で、すべてのジャーナルのすべてのレコードタイプを記述した概要レポートを生成します。

**コード例 9-1** kixjas 概要レポート (1 / 3)

```
Record Types:
       Record type  1 ==> Transid record.
       Record type  2 ==> Userid record.
       Record type  5 ==> SunMTP start-up after normal termination.
       Record type  6 ==> SunMTP system crash occurred.
       Record type 15 ==> Journal file size exhausted.
       Record type 16 ==> SunMTP normal shut-down.
##########################################
JOURNAL FILE:JRNL02.jnl
Records processed:
   acct. header type  5 is skipped.
   acct. header type  6 is skipped.
   acct. header type  6 is skipped.
SunMTP Physical Header(size:418 bytes)
      0 type 1 record processed.
      1 type 2 record processed.
SunMTP Physical Header(size:418 bytes)
      0 type 1 record processed.
      1 type 2 record processed.
```

**コード例 9-1**　　kixjas 概要レポート (2 / 3)

```
SunMTP Physical Header(size:418 bytes)
      0 type 1 record processed.
      1 type 2 record processed.
SunMTP Physical Header(size:418 bytes)
      0 type 1 record processed.
      1 type 2 record processed.
SunMTP Physical Header(size:418 bytes)
      0 type 1 record processed.
      1 type 2 record processed.
SunMTP Physical Header(size:418 bytes)
      0 type 1 record processed.
      1 type 2 record processed.
SunMTP Physical Header(size:418 bytes)
      0 type 1 record processed.
      1 type 2 record processed.
   acct. header type 16 is skipped.
Total records processed:
   record-type         # of records
        1 ----------------- 0
        2 ----------------- 7
        5 ----------------- 1
        6 ----------------- 2
       15 ----------------- 0
       16 ----------------- 1
##########################################
JOURNAL FILE:JRNL05.jnl
Records processed:
   acct. header type  5 is skipped.
   acct. header type  6 is skipped.
   acct. header type  6 is skipped.
   acct. header type 16 is skipped.
Total records processed:
   record-type         # of records
        1 ----------------- 0
        2 ----------------- 0
        5 ----------------- 1
        6 ----------------- 2
       15 ----------------- 0
       16 ----------------- 1
##########################################
```

**コード例 9-1**　　kixjas 概要レポート (3 / 3)

```
JOURNAL FILE:JRNL11.jnl
Records processed:
   acct. header type  5 is skipped.
   acct. header type  6 is skipped.
   acct. header type  6 is skipped.
   SunMTP Physical Header(size:418 bytes)
      0 type 1 record processed.
      1 type 2 record processed.
   acct. header type 16 is skipped.
Total records processed:
   record-type         # of records
        1 ----------------- 0
        2 ----------------- 1
        5 ----------------- 1
        6 ----------------- 2
       15 ----------------- 0
       16 ----------------- 1
##########################################
JOURNAL FILE:JRNL22.jnl
Records processed:
   acct. header type  5 is skipped.
   acct. header type  6 is skipped.
   acct. header type  6 is skipped.
   SunMTP Physical Header(size:418 bytes)
      0 type 1 record processed.
      1 type 2 record processed.
   acct. header type 16 is skipped.
Total records processed:
   record-type         # of records
        1 ----------------- 0
        2 ----------------- 1
        5 ----------------- 1
        6 ----------------- 2
       15 ----------------- 0
       16 ----------------- 1
##########################################
==========================================
Total records for all journal files:
   record-type         # of records
        1 ----------------- 0
        2 ----------------- 9
        5 ----------------- 4
        6 ----------------- 8
       15 ----------------- 0
       16 ----------------- 4
```

# ASCII レコード形式

この節の表は、`kixjas` が生成する ASCII ヘッダーレコードと ASCII データレコード形式を示します。ASCII ヘッダーレコードは、すべての ASCII データレコードの前に付きます。

次の表は、ASCII ヘッダーレコードを構成するフィールドについて説明します。これらのフィールドは、レコード内での出現順になっています。

**表 9-5** ASCII ヘッダーレコード

| オフセット | サイズ (バイト) | フィールド | 説明 |
|---|---|---|---|
| 0 | 8 | Record Length | ヘッダーを含みます。 |
| 8 | 4 | Record Type | トランザクション ID と user のどちらのオプションを指定してレコードが生成されたかを示します。レコードタイプは次のとおりです。<br>0001 Transaction ID<br>0002 ユーザー<br>0005 ジャーナルファイルの始め<br>0006 システムまたは領域クラッシュのあとの起動<br>0015 サイズを使い果たしたあとのジャーナルファイルの終わり<br>0016 通常の領域停止後のジャーナルファイルの終わり |
| 12 | 8 | “unikix” | 左揃え。 |
| 20 | 8 | Sun MTP Release Number | 左揃え。<br>0,1 リリース番号 (たとえば、08)<br>2,3 更新番号 (たとえば、01、02、...)<br>4,5 保守番号 (たとえば、01、02、...)<br>6,7 予約済み |
| 28 | 8 | Reserved | |
| 36 | 8 | System ID | レコードを作成したシステムを示します。 |
| 44 | 4 | Journal ID | 左揃え。レコードが書き込まれるジャーナル ID を示します。 |
| 48 | 8 | Date | 形式<br>*mmddyyyy*<br>説明:<br>*mm* 月<br>*dd* 日<br>*yyyy* 年 |

表 9-5　ASCII ヘッダーレコード (続き)

| オフセット | サイズ (バイト) | フィールド | 説明 |
|---|---|---|---|
| 56 | 12 | Time | 形式<br>*hhmmssdd*<br>説明:<br>*hh*　時<br>*mm*　分<br>*ss*　秒<br>*dd*　1/100 秒 |

次の表は、ASCII データレコードを構成する ASCII フィールドを示します。これらのフィールドは、レコード内での出現順になっています。

表 9-6　ASCII データレコード (1 / 8)

| オフセット | サイズ (バイト) | フィールド | 説明 |
|---|---|---|---|
| 0 | 4 | Transaction Identification | トランザクション ID |
| 4 | 4 | Terminal Identification | このフィールドは、トランザクションサーバーがターミナルまたはセッションに関連付けられていない場合には null になります。 |
| 8 | 8 | ユーザー ID | トランザクション開始時のユーザー ID。 |
| 16 | 4 | Transaction Start Type | 下位のバイトは次のように設定されます。<br>0　端末入力からの接続<br>1　データを伴わない自動トランザクション起動 (ATI) による接続<br>2　データを伴う自動トランザクション起動 (ATI) による接続<br>3　一時データトリガーレベルによる接続<br>4　ユーザーの要求による接続<br>5　端末 TCTTE トランザクション ID からの接続<br>レコードタイプが transaction の場合のみ有効です。 |

表 9-6　ASCII データレコード (2 / 8)

| オフセット | サイズ (バイト) | フィールド | 説明 |
|---|---|---|---|
| 20 | 12 | Start Time of Measurement Interval | TRANID がトランザクションサーバーに接続した時刻。この時刻は 12 文字の値で表されますが、先行するゼロは表示されません。<br>*hhmmssnnnnnn*<br>説明:<br>*hh* 時<br>*mm* 分<br>*ss* 秒<br>*nnnnnn* 「Unit of Time」フィールドの値に応じて、16 ミリ秒単位、ミリ秒単位、マイクロ秒単位のいずれかです。 |
| 32 | 12 | Finish Time of Measurement interval | トランザクションサーバーがトランザクションを終了した時刻。この時刻は 12 文字の値で表されますが、先行するゼロは表示されません。<br>*hhmmssnnnnnn*<br>説明:<br>*hh* 時<br>*mm* 分<br>*ss* 秒<br>*nnnnnn* 「Unit of Time」フィールドの値に応じて、16 ミリ秒単位、ミリ秒単位、マイクロ秒単位のいずれかです。 |
| 44 | 9 | System CPU Time | このトランザクションのシステム CPU 時間。この時間は、秒および 1/1000 秒単位 (99999.999) で表されます。 |
| 53 | 9 | User CPU Time | このトランザクションのユーザー CPU 時間。この時間は、秒および 1/1000 秒単位 (99999.999) で表されます。 |
| 62 | 12 | CICS Time | このトランザクションに対する CICS コマンドの実行に要した時間。この時刻は 12 文字の値で表されますが、先行するゼロは表示されません。<br>*hhmmssnnnnnn*<br>説明:<br>*hh* 時<br>*mm* 分<br>*ss* 秒<br>*nnnnnn* 「Unit of Time」フィールドの値に応じて、16 ミリ秒単位、ミリ秒単位、マイクロ秒単位のいずれかです。 |
| 74 | 10 | Sequence Number | トランザクションのシーケンス番号。 |

表 9-6　ASCII データレコード (3 / 8)

| オフセット | サイズ (バイト) | フィールド | 説明 |
|---|---|---|---|
| 84 | 4 | Operator Identification at Task Creation | トランザクションサーバーの主な機能が端末またはセッションでない場合、またはサインオンエントリに OPIDENT という値がない場合は空白です。 |
| 88 | 8 | Program Name | 接続時に起動された最初のプログラムの名前。 |
| 96 | 20 | System Name | システムを VTAM ネットワークに認識させるための完全修飾名。この名前は、接続時に、TCT システムエントリの情報に基づいて割り当てられます。 |
| 116 | 8 | Unit of Work Name | システム内で作業単位を認識するための名前。この名前は、接続時に、ISC APPC または IRC 接続ヘッダーの一部として渡される作業単位 ID によって割り当てられます。 |
| 124 | 4 | Original Abend Code | |
| 128 | 4 | Current Abend Code | |
| 132 | 4 | Performance Record Type | 下位のバイトは次のように設定されます。<br>C　端末変換に対するレコード OUTPUT。<br>D　ユーザー EMP DELIVER 要求に対するレコード出力。<br>T　タスク終了に対するレコード出力。<br>M　半固定的ミラー中断に対するレコード出力。<br>Sun MTP では、値は常に T です。<br>レコードタイプが transaction の場合のみ有効です。 |
| 136 | 10 | Number of Messages | ユーザータスクによって、主な端末設備に送信されたメッセージ数。ISC APPC トランザクションには適用されません。<br>レコードタイプが transaction の場合のみ有効です。 |
| 146 | 10 | # Characters Received from Principal Terminal | ユーザータスクが、主な端末設備から受信した文字数。ISC APPC トランザクションには適用されません。<br>レコードタイプが transaction の場合のみ有効です。 |
| 156 | 10 | # Messages Received from Principal Terminal | ユーザータスクが、主な端末設備から受信したメッセージ数。ISC APPC トランザクションには適用されません。<br>レコードタイプが transaction の場合のみ有効です。 |
| 166 | 10 | # Characters Sent to Principal Terminal | ユーザータスクによって、主な端末設備に送信された文字数。ISC APPC トランザクションには適用されません。<br>レコードタイプが transaction の場合のみ有効です。 |
| 176 | 10 | TIOA | このユーザータスクに関連付けられている端末に割り当てられた端末記憶領域 (TIOA) の量 (適切な場合)。<br>レコードタイプが transaction の場合のみ有効です。 |
| 186 | 10 | # GETMAIN Requests | ユーザータスクによって実行されたユーザー記憶領域の GETMAIN 要求の数。<br>レコードタイプが transaction の場合のみ有効です。 |

表 9-6　ASCII データレコード (4 / 8)

| オフセット | サイズ (バイト) | フィールド | 説明 |
|---|---|---|---|
| 196 | 10 | Maximum User Storage | ユーザータスクに割り当てられるユーザー記憶領域の最大量。<br>レコードタイプが transaction の場合のみ有効です。 |
| 206 | 10 | # GET Requests | ユーザータスクによって実行されたファイルの GET 要求の数。これは VSAM 読み込みの数。<br>レコードタイプが transaction の場合のみ有効です。 |
| 216 | 10 | # PUT Requests | ユーザータスクによって実行されたファイルの PUT 要求の数。これは VSAM 書き換えの数です。<br>レコードタイプが transaction の場合のみ有効です。 |
| 226 | 10 | # Browse Requests | ユーザータスクによって実行されたファイルの参照要求の数。これには、START と END の 2 つの参照要求は含まれません。<br>レコードタイプが transaction の場合のみ有効です。 |
| 236 | 10 | # ADD Requests | ユーザータスクによって実行されたファイルの ADD 要求の数。これは VSAM 書き込みの数です。<br>レコードタイプが transaction の場合のみ有効です。 |
| 246 | 10 | # DELETE Requests | ユーザータスクによって実行されたファイルの DELETE 要求の数。これは VSAM DELETE の数です。<br>レコードタイプが transaction の場合のみ有効です。 |
| 256 | 10 | # File Control Requests | ユーザータスクによって実行されたファイルの制御要求の合計数。この数は、ファイルの OPEN、CLOSE、ENABLE、または DISABLE のためのすべての要求を除外します。これは、22、23、24、25、および 26 の合計です。<br>レコードタイプが transaction の場合のみ有効です。 |
| 266 | 10 | # Times File Access-method Interfaces Invoked | ユーザータスクがファイルアクセスメソッドインタフェースを起動した回数。この数字では、OPEN と CLOSE に対する要求が除外されます。<br>これは、VSAM ファイルにのみ関係があります。 |
| 276 | 10 | # Transient Data GET Requests | ユーザータスクによって実行された一時データの GET(READQTD) 要求の数。<br>レコードタイプが transaction の場合のみ有効です。 |
| 286 | 10 | # Transient Data PUT Requests | ユーザータスクによって実行された一時データの PUT(WRITEQTD) 要求の数。<br>レコードタイプが transaction の場合のみ有効です。 |
| 296 | 10 | # Transient Data PURGE (DELETE) Requests | ユーザータスクによって実行された一時データの PURGE(DELETEQ) 要求の数。<br>レコードタイプが transaction の場合のみ有効です。 |

表 9-6　ASCII データレコード ( 5 / 8 )

| オフセット | サイズ (バイト) | フィールド | 説明 |
|---|---|---|---|
| 306 | 10 | # Transient Requests Issued by User Task | ユーザータスクによって実行された一時データ要求の合計数。このフィールドは、一時データの GET、PUT、および PURGE 要求の合計です。<br>レコードタイプが transaction の場合のみ有効です。 |
| 316 | 10 | # Temporary Storage GET Requests | ユーザータスクによって実行された一時記憶域の GET(READQ TS) 要求の数。<br>レコードタイプが transaction の場合のみ有効です。 |
| 326 | 10 | # Auxiliary Temporary Storage PUT Requests | ユーザータスクによって実行された、補助一時記憶域に対する PUT(WRITEQ TS) 要求の数。<br>レコードタイプが transaction の場合のみ有効です。 |
| 336 | 10 | # Main Temporary Storage PUT Requests | ユーザータスクによって実行された、主一時記憶域に対する PUT(WRITEQ TS) 要求の数。<br>レコードタイプが transaction の場合のみ有効です。 |
| 346 | 10 | # Temporary Storage Requests by User Task | ユーザータスクによって実行された一時記憶域要求の合計数。このフィールドは、一時記憶域の GET および PUT 要求の合計です。<br>レコードタイプが transaction の場合のみ有効です。 |
| 356 | 10 | # BMS MAP Requests | ユーザータスクによって実行された BMS MAP 要求の数。このフィールドは、端末入出力を伴わない RECEIVE MAP 要求の数と RECEIVE MAP FROM 要求の数に対応します。<br>レコードタイプが transaction の場合のみ有効です。 |
| 366 | 10 | # BMS IN Requests | ユーザータスクによって実行された BMS IN 要求の数。<br>レコードタイプが transaction の場合のみ有効です。 |
| 376 | 10 | # BMS OUT Requests | ユーザータスクによって実行された BMS OUT 要求の数。このフィールドは、SEND MAP 要求の数に対応します。<br>レコードタイプが transaction の場合のみ有効です。 |
| 386 | 10 | # BMS Requests by User Task | ユーザータスクによって実行された BMS 要求の合計数。このフィールドは、BMS MAP、IN、および OUT 要求の合計です。<br>レコードタイプが transaction の場合のみ有効です。 |
| 396 | 10 | # LINK Requests | ユーザータスクによって実行されたプログラムの LINK 要求の数。レコードタイプが transaction の場合のみ有効です。 |
| 406 | 10 | # XCTL Requests | ユーザータスクの最初のプログラムへの制御の転送 (XCTL) など、ユーザータスクによって実行されたプログラムの XCTL 要求の数。<br>レコードタイプが transaction の場合のみ有効です。 |
| 416 | 10 | # LOAD Requests | ユーザータスクによって実行されたファイルのプログラムの LOAD 要求の数。<br>レコードタイプが transaction の場合のみ有効です。 |

表 9-6　ASCII データレコード (6 / 8)

| オフセット | サイズ (バイト) | フィールド | 説明 |
|---|---|---|---|
| 426 | 10 | # Journal Output Requests | ユーザータスク中のジャーナル出力要求の数。<br>レコードタイプが transaction の場合のみ有効です。 |
| 436 | 10 | # Interval Control START Requests または INITIATE Requests | ユーザータスク中の間隔制御の START 要求または INITIATE 要求の数。<br>レコードタイプが transaction の場合のみ有効です。 |
| 446 | 10 | # SYNCPOINT Requests | ユーザータスク中に実行された SYNCPOINT 要求の数。 |
| 456 | 12 | Elapsed Time | ユーザータスクがディスパッチされた経過時間。この時刻は 12 文字の値で表されますが、先行するゼロは表示されません。<br>*hhmmssnnnnnn*<br>説明:<br>*hh* 時<br>*mm* 分<br>*ss* 秒<br>*nnnnnn* 「Unit of Time」フィールドの値に応じて、16 ミリ秒単位、ミリ秒単位、マイクロ秒単位のいずれかです。 |
| 468 | 1 | Unit of Time | バイトに指定された値によって、時間の単位が示されます。<br>**バイト　内容**<br>01　16 ミリ秒単位<br>02　ミリ秒単位<br>03　マイクロ秒 |
| 469 | 8 | Sun MTP System Name | SIT に指定した名前。 |
| 477 | 12 | Response Time | 実行のためにトランザクションがトランザクションサーバーにロードされてから、最初の出力メッセージが端末に送信されるまでの時間差。この時刻は 12 文字の値で表されますが、先行するゼロは表示されません。<br>*hhmmssnnnnnn*<br>説明:<br>*hh* 時<br>*mm* 分<br>*ss* 秒<br>*nnnnnn* 「Unit of Time」フィールドの値に応じて、16 ミリ秒単位、ミリ秒単位、マイクロ秒単位のいずれかです。 |

表 9-6　ASCII データレコード ( 7 / 8 )

| オフセット | サイズ (バイト) | フィールド | 説明 |
|---|---|---|---|
| 489 | 4 | Record Type | **タイプ　内容**<br>0001　transaction (通常の入力トランザクションに対して生成されます)<br>0002　batch (標準のバッチまたは Sun MBM に対して生成されます) |
| 493 | 10 | # OPEN INPUT Requests | バッチジョブからの OPEN INPUT 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 503 | 10 | # OPEN OUTPUT Requests | バッチジョブからの OPEN OUTPUT 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 513 | 10 | # OPEN I/O Requests | バッチジョブからの OPEN I/O 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 523 | 10 | # OPEN EXTEND Requests | バッチジョブからの OPEN EXTEND 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 533 | 10 | # CLOSE Requests | バッチジョブからの CLOSE 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 543 | 10 | # CLOSE WITH LOCK Requests | バッチジョブからの CLOSE WITH LOCK 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 553 | 10 | # READ Requests | バッチジョブからの READ 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 563 | 10 | # READ PREVIOUS Requests | バッチジョブからの READ PREVIOUS 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 573 | 10 | # READ Random Requests | バッチジョブからの READ ランダム要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 583 | 10 | # WRITE Requests | バッチジョブからの WRITE 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 593 | 10 | # REWRITE Requests | バッチジョブからの REWRITE 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 603 | 10 | # START full length key Requests | バッチジョブからの完全長のプライムキーに等しい START の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 613 | 10 | # START Key/Record # Requests | バッチジョブからの任意のキー/レコード番号に等しい START の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 623 | 10 | # START > Requests | バッチジョブからの START (>) 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |

表 9-6　ASCII データレコード (8 / 8)

| オフセット | サイズ (バイト) | フィールド | 説明 |
|---|---|---|---|
| 633 | 10 | # START >= Requests | バッチジョブからの START (>=) 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 643 | 10 | # START < Requests | バッチジョブからの START (<) 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 653 | 10 | # DELETE Requests | バッチジョブからの DELETE 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 663 | 10 | # COMMIT Requests | バッチジョブからの COMMIT (UNLOCK all files) 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 673 | 10 | # ROLLBACK Requests | バッチジョブからの ROLLBACK (UNLOCK all files) 要求の数。<br>レコードタイプが batch の場合のみ有効です。 |
| 683 | 1 | Newline Character | |

# kixjournal シェルスクリプト

ジャーナルファイルのリフレッシュが必要になると、領域は kixjournal シェルスクリプトを呼び出します。ジャーナルファイルは、いっぱいになるか、通常の領域停止のときにリフレッシュされます。ジャーナルファイルがリフレッシュされると、元のジャーナルファイルは、同じ名前で .jnl という接尾辞が付いた代替ファイルに移動されます。代替ファイル名は、引数として kixjournal シェルスクリプトに渡されます。

出荷時の設定では、kixjournal シェルスクリプトは、代替ファイルに対して、まったく機能を実行しません。ただし、unikixmain.log ファイルにメッセージを書き込むコードは入っています。kixlog ユーティリティーを使用して、シェルスクリプトから unikixmain.log、unikixmain.err、および unikixmain.dbg にメッセージを書き込むことができます。kixlog の詳細については、『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。

対象のサイトの要件を確認し、それに応じて kixjournal シェルスクリプトを変更します。たとえば、バックアップとアーカイブ設備のインタフェースによっては、その設備に渡せるファイルが 1 つだけという場合があります。この場合は、バックアップとアーカイブ設備がジャーナルファイルをアーカイブするように kixjournal を変更します。それからあとに、このファイルを適切なファイルに復元して、データを処理します。

# ユーザージャーナルの設定

システム規模のアカウンティングジャーナルに加えて、ユーザージャーナルを定義できます。ユーザージャーナルは、EXEC CICS JOURNAL コマンドによって書き込みます。対象の領域にユーザージャーナルを定義するには、JCT でジャーナルファイルの属性を指定します。JCT フィールドの詳細については、『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。

ジャーナルファイルが一杯になると、そのファイルは拡張子 .jnl を付けてアーカイブされ、新しいファイルが開かれます。システムのアカウンティングファイルとは異なり、ユーザーのジャーナルファイルは領域が停止されてもアーカイブされません。領域が再起動されると、ユーザーのジャーナルファイルが追加モードで開かれるため、レコードは失われません。領域を停止する前に最新のファイルをアーカイブする場合は、以下のタスクを実行するスクリプトを作成できます。

- ジャーナルファイルの名前を *filename* から *filename*.jnl に変更します。
- 入力として *filename*.jnl を使用して kixjournal ユーティリティーを実行します。

次に例を示します。

```
mv userjournal userjournal.jnl
kixjournal userjournal.jnl
```

アプリケーション固有のジャーナルレコードについては、そのジャーナルファイルを読み取るユーティリティーを用意する必要があります。kixjas ユーティリティーは、アカウンティングレコードのみが含まれるジャーナルファイルを読み取ることができます。このため、ジャーナルファイルに Sun MTP レコードとアプリケーション固有のレコードが混在する場合は、このようなファイルを読み取るユーティリティーを用意する必要もあります。独自のユーティリティーを作成するときは、表 9-5 と表 9-6 で説明されている Sun MTP のアカウンティングレコードの形式を参照してください。

---

**注 –** ユーザーのジャーナルレコードと、システムのアカウンティングジャーナルレコードを同じファイルに混在させることはお勧めしません。

---

# 第10章

# アプリケーション環境のカスタマイズ

Sun MTP には、ソースファイルとも呼ばれるカスタマイズツールが用意されていて、これを使用して Sun 以外の製品やユーザー作成のルーチンを統合できます。カスタマイズ作業が終了したら、システム実行ファイルを再構築する必要があります。この章では、次のトピックについて説明します。



Sun MTP は、多くの Sun 以外の製品をサポートしています。それぞれの製品についてどのバージョンをサポートするかは、そのバージョンの入手可能性と適合認定の内容によって決まります。適合認定といっても、すべての製品ですべての機能が動作確認されているわけではなく、またパフォーマンス特性を保証しているわけでもありません。したがって、Sun 以外の製品の候補となるバージョンで、実際にアプリケーションの機能をテストして確認する必要があります。Sun 以外の製品をインストールする場合は、そのバージョンにかかわらず、事前に適合認定の内容を Sun Microsystems へお問い合わせください。

# カスタマイズツール

次のカスタマイズツールは、$UNIKIX/src ディレクトリにあります。

- 標準のユーザー出口ドライバモジュール ($UNIKIX/src/trans/kxusrexit.c)。詳細は、231 ページの「主要なユーザー出口モジュール – kxusrexit.c」を参照してください。
- Oracle、DB2 UDB、および Sybase との統合に使用されるモジュール用のソースファイル (特定のデータベースのディレクトリの下にある $UNIKIX/src/rdbms)。詳細は、232 ページの「RDBMS ユーザー出口ルーチン」を参照してください。
- セキュリティーユーザー出口 ($UNIKIX/src/security/kxsec_exits.c)。詳細は、252 ページの「セキュリティーユーザー出口ルーチン」を参照してください。
- ソケットユーザー出口 ($UNIKIX/src/socket/kxsktxit.c)。詳細は、261 ページの「ソケットユーザー出口のカスタマイズ」を参照してください。
- SSL (Secure Sockets Layer) ユーザー出口 ($UNIKIX/src/socket/kxsslxit.c)。詳細は、264 ページの「SSL ユーザー出口のカスタマイズ」を参照してください。
- トランザクション経路指定のための変換ルーチンやその他の ISC 機能が含まれる ISC ユーザー出口 ($UNIKIX/src/isc)。詳細は、265 ページの「ISC ユーザー出口のカスタマイズ」を参照してください。
- 端末非存在状態の出口 ($UNIKIX/src/terminal)。詳細は、269 ページの「端末非存在状態の出口」を参照してください。
- レコード処理ルーチン ($UNIKIX/src/record)。詳細は、270 ページの「レコード処理ルーチンのカスタマイズ」を参照してください。
- 回復プロセッサユーザー出口 ($UNIKIX/src/recovery/kxesdsxlt.c)。詳細は、274 ページの「回復プロセッサユーザー出口」を参照してください。

アプリケーション環境をカスタマイズするときは、次の点に注意してください。

- Sun MTP に付属のソースモジュールへの変更とそれから生成される実行可能ファイルは、サポートされるアプリケーションの一部です。したがって、変更したソースモジュールは、アプリケーションソースと同じソース制御システム内に取り込む必要があります。
- ソース、オブジェクト、および実行可能モジュールには、アプリケーションに使用するのと同じレベルのバックアップ保護を設定する必要があります。これにより、ハードウェアまたはソフトウェアに障害が発生したときも、完全なアプリケーションを正確にすばやく再構築できます。

# 主要なユーザー出口モジュール – kxusrexit.c

ユーザー出口ドライバモジュール (`$UNIKIX/src/trans/kxusrexit.c`) は、トランザクション処理プログラムの起動と終了、およびトランザクションの開始と終了を管理します。

開発者は、戻り値を確認したり、必要なアクションを取ったりする条件付きコードを実装できます。提供されているソースコードのドライバプログラムは、特定のユーザー出口を呼び出し、リターンコードを確認し、そのコードを次のように設定します。

| リターンコード | ラベル | 実行文 |
|---|---|---|
| 0 | KXSUCCESS | 継続実行 |
| 1 | KXFAILURE | 終了トランザクション |
| -1 | KXABORT | 終了トランザクションサーバー |

実行文は、トランザクションおよびバッチ処理プログラムにリターンコードが渡されたときに、それらが取るアクションです。KXABORT は、ソースモジュールプログラムが、データベースの割り当て (接続または使用) に失敗したときに発生します。KXFAILURE は、データベースが BEGIN TRANSACTION 関数の実行に失敗したときに生じます。これはトランザクション障害です。

また、開発者は必要に応じて、RDBMS ユーザーモジュールまたは kxusrexit.c にあるエラーの重大度を確認したあと、トランザクションの中止または終了の決定もできます。kxusrexit.c には、各 RDBMS の条件付きコードがあります。詳細は、232 ページの「RDBMS ユーザー出口ルーチン」を参照してください。makefile では、RDBMSFLAG 変数が必要なデータベースに設定されていて、これによってコンパイラが特定のコードを取り込みます。複数の RDBMS を取り込む場合は、RDBMSFLAG 変数が 1 つ以上の値を持つことができます。

kxusrexit.c プログラムには次の各行も含まれていて、これによってすべてのトランザクションに対するメッセージをトレースできるようになっています。

```
/*************************************************************/
/* To enable TRACEMSG after every transaction                */
/* uncomment the following code                              */
/*************************************************************/
/* #define TRACEMSG 1                                        */
```

トレースメッセージは、あらゆる段階 (開始、終了、コミット、またはロールバック) で行われる正常なトランザクションすべてについて状態を返します。

# RDBMS ユーザー出口ルーチン

**注 –** この節は、XA 以外の環境にのみ該当します。Sun MTP を XA 環境で使用している場合は、『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 ユーザー出口ルーチンの主な目的は、暗黙的なデータベースのコミットとロールバック機能を RDBMS ソフトウェアで使用できるようにすることです。これらのルーチンは、ソースファイル `$UNIKIX/src/trans/kxusrexit.c` にあります。次の表では、ルーチンについて説明します。

**表 10-1** RDBMS ユーザー出口

| ユーザー出口ルーチン | 説明 |
|---|---|
| トランザクション/バッチサーバーの初期化 | RDBMS ソフトウェアへの接続またはログインを実行します。これは、RDBMS ソフトウェアへの明示的な接続またはログイン操作が必要な場合に便利です。各バッチおよびトランザクションサーバー内のアプリケーションが使用する各データベースに対して、ログインが要求されるのは一度です。 |
| トランザクション/バッチサーバーの終了 | RDBMS ソフトウェアから切断またはログオフします。トランザクションサーバーの障害など、状況によっては、この出口を呼び出せません。このような場合、RDBMS サーバーソフトウェアは、クライアントプロセスが終了したことを検出し、接続をクリーンアップしてから、異常終了の時点で実行中だった作業を暗黙的にロールバックします。 |
| トランザクションの初期化 | トランザクションの実行前にロジックを実行します。 |

表 10-1　RDBMS ユーザー出口 (続き)

| ユーザー出口ルーチン | 説明 |
|---|---|
| トランザクションの終了 | トランザクションの終了後にロジックを実行します。 |
| データベースのコミット | アプリケーションが使用していた RDBMS に対して、暗黙的にコミットを実行します。これにより、アプリケーション内に明示的な COMMIT 文をコーディングしなくても、RDBMS のデータベースの統合性が維持できます。VSAM ファイルのデータベースコミット処理は、Sun MTP エグゼクティブの中に埋め込まれているので、このユーザー出口の中でコードを実行する必要はありません。<br>このユーザー出口は、SYNCPOINT が実行されたとき、またはトランザクションの終了時にデータベースのコミットが実行されたときに呼び出されます。 |
| データベースのロールバック | アプリケーションが使用した RDBMS のそれぞれに対して、暗黙的にデータベースのロールバックを実行します。<br>このユーザー出口は、SYNCPOINT ROLLBACK が実行されたとき、またはトランザクションの異常終了時にデータベースのロールバックが実行されたときに呼び出されます。 |

kxusrexit.c モジュールは、Sun MTP ユーザー出口メカニズムのドライバだと考えられます。このモジュールで定義されている関数は、あらかじめ定義されたポイントでエグゼクティブから直接呼び出されます。次に、kxusrexit.c が、アプリケーションが要求するモジュールを呼び出します。モジュールのサンプルは、$UNIKIX/src/rdbms ディレクトリにあります。

# 関数呼び出し

使用するプログラミング言語にかかわらず、コンパイルの前に適切な RDBMS プリプロセッサを使用して、モジュールを処理する必要があります。この節では、さらに論理チェックを行うための情報を Sun MTP から取得するために、アプリケーションプログラムで使用する関数呼び出しについて説明します。

## kxsysinfo

次の値を SIT から返します。

- システムアプリケーション名
- サーバー名
- RDBMS 名
- ユーザー ID
- パスワード
- 各トランザクション/バッチサーバーの一意の番号

COBOL プログラムの例

```
CALL 'kxsysinfo'  USING  KIX-SYS-INFO.
```

この関数は、たとえば取得した情報に基づいてデータベースに接続する場合に、トランザクション/バッチサーバーの初期化または終了のユーザー出口から呼び出します。

各トランザクション/バッチサーバーに対して返された一意の番号を使用して、データベースのソケットや接続文字列を識別できます。

Sybase 用の COBOL プログラムの例

```
call 'kxsysinfo' USING KIX-SYS-INFO.
MOVE KIX-SIT-SRV-NAME TO SERVER-NAME.
MOVE KIX-SIT-USR-NAME TO USERID.
MOVE KIX-SIT-USR-PASS TO USERPASS.
MOVE KIX-TRANIDX      TO CONNECT-NAME.
EXEC SQL  CONNECT :USERID IDENTIFIED BY:USERPASS
       AT :CONNECT-NAME
       USING :SERVER-NAME
END-EXEC.
```

SQL*NET を使用する Oracle 対応の COBOL プログラムでは、この番号によって接続のための一意のソケット番号を指定できます。

## kxtctinfo

次の情報を返します。

- ユーザー名
- オペレータセキュリティーキー
- オペレータクラス
- 端末 ID
- LU 名
- オペレータ ID
- トランザクションコード

COBOL プログラムの例

```
CALL 'kxtctinfo'  USING  KIX-TCT-INFO.
```

この関数は、トランザクション初期化または終了のユーザー出口から呼び出します。データベース固有のセキュリティーを実装するために使用します。各 RDBMS のコメント付きコードに、この例が説明されています。

RDBMS のセキュリティー実装の詳細については、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。

## kxsetmsg

ユーザー固有のメッセージをユーザー端末に設定します。

COBOL プログラムの例

```
MOVE  'User not authorized to execute SQL
       programs' TO  KIX-MSG-STR.
CALL 'kxsetmsg' USING KIX-MSG-INFO.
```

代わりにユーザー定義のエラーメッセージ番号を KIX-MSG-INFO ブロックの KIX-MSGNO フィールドで指定してから、メッセージ機能を呼び出すこともできます。

```
CALL 'kxsetmsg' USING KIX-MSG-INFO.
```

## kxttyinfo

次の情報を返します。

- 端末 PID
- デバイス名
- TERM 環境変数

COBOL プログラムの例

```
CALL 'kxttyinfo'  USING  KIX-TTY-INFO.
```

この関数は、トランザクション初期化または終了のユーザー出口から呼び出します。

---

**注 –** 呼び出しによってこの関数に値が返される前に、デバイス名が、/dev/pty/ttyu2 のような有効なデバイスを表していることを確認してください。

---

次の例のように、COBOL コピーブック $UNIKIX/src/rdbms/KXINFO.CPY には、それぞれの関数に対して返されたすべての変数とその属性が入っています。

**コード例 10-1** KXINFO.CPY コピーブック (1 / 2)

```
*  User exit table contains TCT information, which users can  *
*  use in the COBOL program                                   *
*                                                             *
*  User code should call the 'kxtctinfo' during start         *
*  transaction user exit get the latest KIX-TCT-INFO record   *
*  updated.                                                   *
*                                                             *
*  example:-  call "kxtctinfo" using KIX-TCT-INFO             *
*************************************************************


 01  KIX-TCT-INFO.
     05  KIX-USRNAM                PIC X(8).
     05  KIX-OPSEC                 PIC X(8).
     05  KIX-OPCLS                 PIC X(3).
     05  KIX-TRMID                 PIC X(4).
     05  KIX-LUNAME                PIC X(8).
     05  KIX-OPID                  PIC X(3).
     05  KIX-TRANCD                PIC X(4).
*************************************************************
*  User exit contains system application id from SIT table     *
*  which user can use in the cobol program.             *
*
*                                                             *
*  User code should call the 'kxsysinfo' during allocate      *
*  transaction user exit gets the KIX-SYS-INFO record updated. *
*                                                             *
*  example:-  call "kxsysinfo" using KIX-SYS-INFO             *
*************************************************************


01  KIX-SYS-INFO.
     05  KIX-SITNAME               PIC X(8).
     05  KIX-SIT-SRV-NAME          PIC X(8).
     05  KIX-SIT-DB-NAME           PIC X(8).
     05  KIX-SIT-USR-NAME          PIC X(8).
     05  KIX-SIT-USR-PASS          PIC X(8).
     05  KIX-TRANIDX               PIC X(3).
```

**コード例 10-1**　KXINFO.CPY コピーブック (2 / 2)

```
*****************************************************************
*  User exit sets the user message and displays it on the terminal*
*                                                                *
*  User code should call the 'kxsetmsg' any time in the program   *
*  to put the message in the queue. The message is printed once.
*  *
*  The next message is received on the buffer.              *
*
*                                                                *
*  example:-  call "kxsetmsg" using KIX-MSG-INFO                 *
*****************************************************************
 01  KIX-MSG-INFO.
     05  KIX-MSGNO                  PIC 9(4) VALUE 1098.
     05  KIX-MSG-ROUTINE            PIC X(20).
     05  KIX-MSG-STR                PIC X(60).
*****************************************************************
*  User exit sets the UNIX tty information for the terminal     *
*                                                                *
*  User code should call the 'kxttyinfo' any time in the program*
*  to get the UNIX tty information.                              *
*                                                                *
*  example:-  call "kxttyinfo" using KIX-TTY-INFO                *
*****************************************************************


01  KIX-TTY-INFO.
     05  KIX-TTY-PID                 PIC 9(6).
     05  KIX-TTY-NAME                PIC X(20).
     05  KIX-TTY-TERMENV             PIC X(14).
```

次の例のように、C のヘッダーファイル $UNIKIX/src/rdbms/kxinfo.h には、それぞれの関数に対して返されたすべての変数とその属性が入っています。

**コード例 10-2**　　kxinfo.h C のヘッダーファイル

```
/* Structure for passing parameter information from tct table */

struct lextpar {
       char    cur_usrnam [8];
       char    cur_opsec  [8];
       char    cur_opcls  [3];
       char    cur_trmid  [4];
       char    cur_luname [8];
       char    cur_opid   [3];
       char    cur_trancd [4];
       } lextpar;

struct lsyspar {
       char    cur_sysname [8];
       /* Case 2240 */
       /* Added entries for RDBMS related variables */
       char    cur_svrname [8];
       char    cur_dbname  [8];
       char    cur_usrname [8];
       char    cur_usrpass [8];
       char    cur_tranidx [3];
       } lsyspar;

struct lmsgstr {
       char    cur_errno [4];
       char    cur_func  [20];
       char    cur_msg   [60];
       } lmsgstr;

struct lttystr {
       char    cur_termpid [6];
       char    cur_ttyname [20];
       char    cur_termenv [14];
```

次の小節のコードは、RDBMS の条件文に囲まれています。これらの関数の主な目的は、別のモジュールで指定する必要がある RDBMS 関数を呼び出すことです。

---

**注 –** kxusrexit.c コードは、次の小節に記載がない限り、変更する必要はありません。

---

## kxusrexit.c ー Oracle RDBMS 関数

関数 KXORALGN、KXORALGF、KXORABTRN、KXORAETRN、KXORASAVE、および KXORAUNDO は、$UNIKIX/src/rdbms ディレクトリにある KXORACLE.pco プログラムで定義されています。

Oracle データベースのコードを取り込むには、makefile の RDBMSFLAG を -DORACLE に設定します。

**コード例 10-3**　Oracle ユーザー出口 (1 / 4)

```
/*****************************************************************/
/* The following code is specific to the support of Oracle       */
/* databases.The routines called by the functions below can be   */
/* found in the KXORACLE.pco source file which is located in the */
/* $UNIKIX/src directory.                                        */
/*                                                               */
/*****************************************************************/
#ifdef ORACLE
/*****************************************************************/
/*     ORACLE          allocate/open function                    */
/*****************************************************************/

static int oracle_allocate() {

#ifdef XOPEN
      strcpy(CurFunc,"XA_oracle_allocate\0");  /*  Do not change */
      dbms_rcode = KXORALGN_XA();
       return(dbms_rcode);
#endif

      strcpy(CurFunc,"oracle_allocate\0");  /*  Do not change */
#ifdef ACUCOBOL
      dbms_rcode = kxcobload("KXORACLE",KXDEBUG_NONE,0,NULL,0,LOGERR,NULL,0);
#else
      dbms_rcode =KXORALGN();
#endif
       return(dbms_rcode);
}
```

コード例 10-3　　Oracle ユーザー出口 (2 / 4)

```
/*****************************************************************/
/*     ORACLE          deallocate/close function                 */
/*****************************************************************/

static int oracle_deallocate() {

#ifdef XOPEN
      strcpy(CurFunc,"XA_oracle_deallocate\0");  /*  Do not change */
      dbms_rcode = KXORALGF_XA();
       return(dbms_rcode);
#endif

      strcpy(CurFunc,"oracle_deallocate\0");  /*  Do not change */
#ifdef ACUCOBOL
      dbms_rcode = kxcobload("KXORALGF",KXDEBUG_NONE,0,NULL,0,LOGERR,NULL,0);
#else
      dbms_rcode =KXORALGF();
#endif
       return(dbms_rcode);

}

/*****************************************************************/
/*     ORACLE          start_txn function                        */
/*****************************************************************/

static int oracle_start_txn() {

#ifdef XOPEN
      strcpy(CurFunc,"XA_oracle_start_txn\0");  /*  Do not change */
      dbms_rcode = KXORABTRN_XA();
       return(dbms_rcode);
#endif

      strcpy(CurFunc,"oracle_start_txn\0");  /*  Do not change */
#ifdef ACUCOBOL
      dbms_rcode = kxcobload("KXORABTRN",KXDEBUG_NONE,0,NULL,0,LOGERR,NULL,0);
#else
      dbms_rcode =KXORABTRN();
#endif
       return(dbms_rcode);

}
```

コード例 10-3　　Oracle ユーザー出口 ( 3 / 4 )

```
/*****************************************************************/
/*     ORACLE          end of txn function                       */
/*****************************************************************/

static int oracle_end_txn() {

#ifdef XOPEN
      strcpy(CurFunc,"XA_oracle_end_txn\0");  /*  Do not change */
      dbms_rcode = KXORAETRN_XA();
       return(dbms_rcode);
#endif

      strcpy(CurFunc,"oracle_end_txn\0");  /*  Do not change */
#ifdef ACUCOBOL
      dbms_rcode = kxcobload("KXORAETRN",KXDEBUG_NONE,0,NULL,0,LOGERR,NULL,0);
#else
      dbms_rcode =KXORAETRN();
#endif
       return(dbms_rcode);

}

/*****************************************************************/
/*     ORACLE          commit function
                          */
/*****************************************************************/

static int oracle_commit() {

#ifndef XOPEN
      strcpy(CurFunc,"oracle_commit\0");  /*  Do not change */
#ifdef ACUCOBOL
      dbms_rcode = kxcobload("KXORASAVE",KXDEBUG_NONE,0,NULL,0,LOGERR,NULL,0);
#else
      dbms_rcode =KXORASAVE();
#endif
       return(dbms_rcode);
#endif
      return(0);

}
```

コード例 10-3　　Oracle ユーザー出口 (4 / 4)

```
/*****************************************************************/
/*     ORACLE          rollback function
                          */
/*****************************************************************/

static int oracle_rollback() {

#ifdef XOPEN
      strcpy(CurFunc,"XA_oracle_rollback\0");  /*  Do not change */
      dbms_rcode = KXORAUNDO_XA();
       return(dbms_rcode);
#endif

      strcpy(CurFunc,"oracle_rollback\0");  /*  Do not change */
#ifdef ACUCOBOL
      dbms_rcode = kxcobload("KXORAUNDO",KXDEBUG_NONE,0,NULL,0,LOGERR,NULL,0);
#else
      dbms_rcode = KXORAUNDO();
#endif
       return(dbms_rcode);

}

#endif           /* End Oracle code */
```

## kxusrexit.c ー DB2 UDB RDBMS 関数

DB2 UDB データベースのコードを取り込むには、makefile の RDBMSFLAG を -DDBTWO に設定します。

**コード例 10-4**　　DB2 UDB ユーザー出口 (1 / 3)

```
/*****************************************************************/
/* The following code is specific to the support of db2 databases.*/
/* The routines called by the functions below can be found in     */
/* the KXDB2.cbl source file, located in the $UNIKIX/src          */
/*directory.                                */
  */
/*****************************************************************/
#ifdef DBTWO
/*****************************************************************/
/*     DB26000    allocate/open function                         */
/*****************************************************************/

static int dbtwo_allocate() {

      strcpy(CurFunc,"dbtwo_allocate\0");  /*  Do not change */
      dbms_rcode = KXDB2LGN();
      return(dbms_rcode);

}

/*****************************************************************/
/*     DB26000     deallocate/close function                     */
/*****************************************************************/

static int dbtwo_deallocate() {

      strcpy(CurFunc,"dbtwo_deallocate\0");  /*  Do not change */
      dbms_rcode = KXDB2LGF();
      return(dbms_rcode);

}
```

**コード例 10-4**　　DB2 UDB ユーザー出口 (2 / 3)

```
/*****************************************************************/
/*     DB26000     start_txn function                            */
/*****************************************************************/

static int dbtwo_start_txn() {

      strcpy(CurFunc,"dbtwo_start_txn\0");  /*  Do not change */
      dbms_rcode = KXDB2BTRN();
      return(dbms_rcode);

}

/*****************************************************************/
/*     DB26000     end of txn function                           */
/*****************************************************************/

static int dbtwo_end_txn() {

      strcpy(CurFunc,"dbtwo_end_txn\0");  /*  Do not change */
      dbms_rcode = KXDB2ETRN();
      return(dbms_rcode);

}

/*****************************************************************/
/*     DB26000     commit function                               */
/*****************************************************************/

static int dbtwo_commit() {

      strcpy(CurFunc,"dbtwo_commit\0");  /*  Do not change */
      dbms_rcode = KXDB2SAVE();
      return(dbms_rcode);

}
```

**コード例 10-4**　DB2 UDB ユーザー出口 ( 3 / 3 )

```
/*****************************************************************/
/*     DB26000     rollback function                             */
/*****************************************************************/

static int dbtwo_rollback() {

      strcpy(CurFunc,"dbtwo_rollback\0");  /*  Do not change */
      dbms_rcode = KXDB2UNDO();
      return(dbms_rcode);

}

#endif           /* End db26000  code */
```

## kxusrexit.c ー Sybase RDBMS 関数

Sybase のユーザー出口関数 KXSYBLGN、KXSYBLGF、KXSYBBTRN、KXSYBETRN、KXSYBSAVE、および KXSYBUNDO は、$UNIKIX/src/rdbms ディレクトリにある KXSYBASE.cop プログラムで定義されています。

Sybase データベースのコードを取り込むには、makefile の RDBMSFLAG を -DSYBASE に設定します。

-DSYBASE10 フラグの使用方法について記載されているコード例 10-6 も参照してください。

**コード例 10-5**　Sybase ユーザー出口 ( 1 / 4 )

```
/*****************************************************************/
/*                                                               */
/*The following code is specific to the support of SYBASE databases.*/
/* The routines called by the functions below can be found in the   */
/*KXSYBASE.cop source file which is located in the $UNIKIX/src      */
/*directory.                                                       */
/*                                                               */
/*****************************************************************/
#ifdef SYBASE
```

コード例 10-5　Sybase ユーザー出口 (2 / 4)

```
/*****************************************************************/
/*     SYBASE          allocate/open function                    */
/*****************************************************************/

static int sybase_allocate() {

/*---------------------------------------------------------------*/
/* RS6000 and SYBASE10 specific for changing process group id    */
/* Do not change any part of the code in ifdef                   */
/*---------------------------------------------------------------*/
#ifdef SYBASE10
#ifdef RS6000
        int lrc;
        pid_t l_pgid;

        l_pgid = getpid();
        lrc    = setpgid(l_pgid, l_pgid);
        if (lrc != 0)
           {
           strcpy(CurFunc,"sybase_allocate\0");  /*  Do not change */
          kxprtf("Error changing process group id  - Errno %d, returncode %d\n"
                  , errno,lrc);
           return(KXCOBOLABORT);
           }
#endif  /*  RS6000 condition */
#endif  /*  SYBASE10 condition */
/*---------------------------------------------------------------*/

      strcpy(CurFunc,"sybase_allocate\0");  /*  Do not change */
      dbms_rcode = KXSYBLGN();
      return(dbms_rcode);

}

/*****************************************************************/
/*     SYBASE          deallocate/close function                 */
/*****************************************************************/

static int sybase_deallocate() {

      strcpy(CurFunc,"sybase_deallocate\0");  /*  Do not change */
      dbms_rcode = KXSYBLGF();
      return(dbms_rcode);

}
```

コード例 10-5　　Sybase ユーザー出口 (3 / 4)

```
/*****************************************************************/
/*     SYBASE          start_txn function                        */
/*****************************************************************/

static int sybase_start_txn() {

      strcpy(CurFunc,"sybase_start_txn\0");  /*  Do not change */
      dbms_rcode = KXSYBBTRN();
      return(dbms_rcode);

}

/*****************************************************************/
/*     SYBASE          end of txn function                       */
/*****************************************************************/

static int sybase_end_txn() {

      strcpy(CurFunc,"sybase_end_txn\0");  /*  Do not change */
      dbms_rcode = KXSYBETRN();
      return(dbms_rcode);

}

/*****************************************************************/
/*     SYBASE        commit function                             */
/*****************************************************************/

static int sybase_commit() {

      strcpy(CurFunc,"sybase_commit\0");  /*  Do not change */
      dbms_rcode = KXSYBSAVE();
      return(dbms_rcode);

}
```

コード例 10-5　Sybase ユーザー出口 (4 / 4)

```
/*****************************************************************/
/*     SYBASE        rollback function                           */
/*****************************************************************/


static int sybase_rollback() {

      strcpy(CurFunc,"sybase_rollback\0");  /*  Do not change */
      dbms_rcode = KXSYBUNDO();
      return(dbms_rcode);

}
```

**Sybase System 10 以降**

Oracle と異なり、Sybase には、すべての Sybase ESQL/COBOL 関数を実行可能ファイルに取り込む追加コードがあります。このコードが使用できるのは、Sybase 4.9.x と ESQL/COBOL コンパイラだけです。

次のコードは、Sybase system 10 の条件付き定義に囲まれています。この条件付き定義は、`makefile` の `RDBMSFLAG` で、`-DSYBASE10` の値が設定されている場合には取り込まれません。

コード例 10-6　Sybase System 10 以降のユーザー出口 (1 / 3)

```
#ifndef SYBASE10
/*************************************************************/
/* Do not remove the following function definition; it is    */
/* required to satisfy SYBASE object syntax                  */
/*************************************************************/
void sybase_null_func()
{
int void_number;
void_number = sql_put_tds_vsn();
void_number = sqlchkxact();
void_number = sqlallrel();
void_number = sqlbadassert();
void_number = sqlbind();
void_number = sqlca();        /* Comment this function in 4.9.2 */
```

**コード例 10-6**　Sybase System 10 以降のユーザー出口 ( 2 / 3 )

```
void_number = sqlcancel();
void_number = sqlchkfetch();
void_number = sqlcobvalchk();
void_number = sqlconvert ();
void_number = sqlcstring ();
void_number = sqlcurcloseconn ();
void_number = sqlcurfind ();
void_number = sqlcurisopenconn ();
void_number = sqlcurset ();
void_number = sqldispnum ();
void_number = sqldynexec ();
void_number = sqldyngetexpct ();
void_number = sqlerror ();
void_number = sqlexecarg ();
void_number = sqlexecdo ();
void_number = sqlexecinit ();
void_number = sqlfetchfree ();
void_number = sqlfetchsuspend ();
void_number = sqlfindconn ();
void_number = sqlflogin ();
void_number = sqlgetcmd ();
void_number = sqlgetdbtype ();
void_number = sqlgetexpct ();
void_number = sqlgetfetch ();
void_number = sqlgetfetchqual ();
void_number = sqlgetstype ();
void_number = sqlgetsvargs ();
void_number = sqlimmexec ();
void_number = sqlinitca ();
void_number = sqllinkv ();
void_number = sqllogin ();
void_number = sqlmakcur ();
void_number = sqlmem ();
void_number = sqlmkcmd ();
void_number = sqlnewconn ();
void_number = sqlnext1 ();
void_number = sqlnextr ();
void_number = sqlnprocrow ();
void_number = sqlnullstr ();
void_number = sqloproc ();
void_number = sqlpproc ();
void_number = sqlrelease ();
```

**コード例 10-6**　Sybase System 10 以降のユーザー出口 ( 3 / 3 )

```
void_number = sqlrproc ();
void_number = sqlrterr ();
void_number = sqlrtlibstr ();
void_number = sqlrtlibvsn ();
void_number = sqlsetbadassert ();
void_number = sqlsetcmd ();
void_number = sqlsetcurbind ();
void_number = sqlseterr ();
void_number = sqlstproc ();
void_number = sqlulinkv ();
void_number = sqluxact ();
void_number = sqlwarn ();
void_number = sqlwarning ();
void_number = sqlxact ();
void_number = sqlxactcmd ();
void_number = sqlxactuse ();
void_number = sqlxcmd ();
}
```

次のダミー関数定義は、未定義の条件を満たすために使用します。ご使用の Sybase のリリースに適合する定義を使用してください。

```
int sql_put_tds_vsn()
{
        return -1;
}

int sqlchkxact()
{
        return -1;
}
.
.
#endif           /* End SYBASE10 Check */
#endif           /* End SYBASE code */
```

# ユーザーモジュールのソース例

次の RDBMS ユーザーソースモジュールは、$UNIKIX/src/rdbms ディレクトリにあります。

表 10-2　RDBMS ユーザーモジュール

| RDBMS | COBOL | C | PL/I |
|---|---|---|---|
| Oracle | KXORACLE.pco | kxoracle.pc | KXORAPLI.ppl |
| DB2 UDB | KXDB27.sqb | kxdb2.sqc | - |
| Sybase | KXSYBASE.cop | kxsybase.cp | - |

ソースモジュールは、ご使用のサイト要件に合わせて変更できます。次の表は、RDBMS 関数を 表 10-1 に定義されているユーザー出口関数にマップします。

表 10-3　RDBMS 関数とユーザー出口関数とのマップ

| ユーザー出口関数 | Oracle | DB2 | Sybase |
|---|---|---|---|
| プロセスの初期化 | KXORALGN | KXDB2LGN | KXSYBLGN |
| プロセスの終了 | KXORALGF | KXDB2LGF | KXSYBLGF |
| トランザクションの初期化 | KXORABTRN | KXDB2BTRN | KXSYBBTRAN |
| トランザクションの終了 | KXORAETRN | KXDB2ETRN | KXSYBETRN |
| データベースのコミット | KXORASAVE | KXDB2SAVE | KXSYBSAVE |
| データベースのロールバック | KXORAUNDO | KXDB2UNDO | KXSYBUNDO |

本章でこれまで説明したソース例は、呼び出し可能な関数の使用方法を示します。

**注意 –** kxsysinfo、kxtctinfo、kxsetmsg、および kxttyinfo 関数は、示したようなユーザー出口関数のコンテキストだけで使用できます。このコンテキスト以外で使用する場合、予測できない結果が生じる場合があります。

ソースモジュールは、RDBMS 固有の API および言語の文を使用するため、このような文を変更する際には注意が必要です。

必要な変更を行なってから、kixinstall ユーティリティーを実行してオブジェクトバイナリを生成する必要があります。 kixinstall は、RDBMS プリコンパイラおよび言語コンパイラからソースモジュールを処理し、オブジェクトバイナリを生成します。

# セキュリティーユーザー出口ルーチン

Sun MTP 領域が Sun MTP Secure を使用して設定されている場合、あらかじめ定義された固定ポイントで、セキュリティー出口ルーチンが呼び出されます。これは、その領域に対して設定されている外部マネージャーと Sun MTP が通信する方法です。この外部マネージャーは、Sun MSF であるか、カスタマイズしたセキュリティー出口ルーチンを使用して Sun MTP を再構築している場合は、そのセキュリティー出口ルーチンです。

独自のセキュリティーユーザー出口をコーディングするためのテンプレートは、`$UNIKIX/src/security/kxsec_exitsTMPL.c` です。Sun MTP のリソースタイプに必要な付属の定義とアクセス権は、ファイル `$UNIKIX/src/security/security.h` にあります。

この領域の外部セキュリティーが有効になっているときは (`KIXSEC=YES`)、セキュリティーユーザー出口モジュールで定義されている関数は Sun MTP エグゼクティブから直接呼び出されます。セキュリティー要求を検査して適切な応答を返すのは、これらの関数です。ここで、エグゼクティブは、セキュリティー検証を通過した操作を続けたり、セキュリティー検証に失敗した操作を終了したり適切なアクションを実行します。

独自のセキュリティーユーザー出口を設計して実装する前に、第 8 章をお読みください。ここには、セキュリティーユーザー出口によって行われるセキュリティーチェックで必要なリソースタイプとデフォルトのユーザー名についての説明があります。

## 関数とパラメータ

セキュリティーユーザー出口の関数を次に示します。

### セキュリティーマネージャー状態

この関数は、外部セキュリティーマネージャーが稼動中であるかどうかを調べるために呼び出されます。Sun MTP は、残りの関数を使用して、すべてのセキュリティー検証チェックを行います。これが使用可能でない場合 (デフォルト)、トランザクションセキュリティーキーチェックが使用されます。

**関数**:

```
kxsec_mgrstatus
```

**用法**:

```
boolean kxsec_mgrstatus()
```

**入力**:

なし

**出力**:

リターンコード:

TRUE: 外部セキュリティーマネージャーが稼動中であり、使用可能です。

FALSE: 外部セキュリティーマネージャーは現在稼動していません。

## ユーザーログイン

この関数は、ユーザー名とパスワードを確認したり、いったん確認したパスワードを必要に応じて新しいパスワードで更新したりするために呼び出されます。この関数が呼び出されるのは、トランザクションを領域に送信するためにユーザーを確認する必要がある場合、すなわち、ローカル端末が接続されるとき、遠隔システムのトランザクション要求を受信したとき、および CSSN または CESN トランザクションに入ったときです。ユーザーが確認されると、この関数は TRUE を返し、ユーザーが拒否された場合は FALSE を返します。

必要に応じて、この関数は独自の認証規則の代わりに、または独自の認証規則に加えて既存の SNT エントリを使用してユーザー名とパスワードを認証することができます。この SNT 確認は、kxchksntpw() 関数 (260 ページの「SNT を使用したユーザー名とパスワードの確認」を参照) を呼び出すことによって提供されます。

**関数**:

kxsec_login

**用法**:

```
int kxsec_login(
       char *username, int username_length,
       char *rolename, int rolename_length,
       char *password, int password_length,
       char *newpw, int newpw_length)
```

**入力**:

username: ログオンしているユーザーの名前

username_length: そのユーザー名の文字数

rolename: ユーザーの役割

rolename_length: その役割名の文字数

password: そのユーザーのパスワード

password_length: そのパスワードの文字数

newpw: そのユーザーの新しいパスワード (入力した場合)

newpw_length: その新しいパスワードの文字数。入力しなかった場合は 0 (ゼロ) です。

**出力**:

リターンコード:

この関数は、表 10-4 のいずれかのコードを返す必要があります。

---

**注 –** 下記のリターンコードは、EXEC CICS SIGNON API から返された RESP2 値にマッピングされています。

---

**表 10-4** kxsec_login のリターンコード

| リターンコード | 型 | RESP2 値 | 説明 |
|---|---|---|---|
| LOGIN_SUCCESS | | 0 | ユーザーが正常にサインオンしました。 |
| LOGIN_USERID_NOT_FOUND | USERIDERR | 8 | ユーザー ID が無効です。 |
| LOGIN_PW_FAILURE | NOTAUTH | 2 | パスワードが正しくありません。 |
| LOGIN_PASSWORD_EXPIRED | NOTAUTH | 3 | パスワードの期限が切れています。 |
| LOGIN_INVALID_PW_FORMAT | NOTAUTH | 4 | 新しいパスワードの形式が正しくありません。パスワードが長すぎるか、短すぎる可能性があります。 |
| LOGIN_PW_SUSPENDED | NOTAUTH | 19 | ユーザー ID が停止されています。 |
| LOGIN_ROLEID_NOT_FOUND | NOTAUTH | 23 | 役割名が無効です。 |
| LOGIN_USERID_NOT_IN_ROLE | NOTAUTH | 24 | この役割を使用できないユーザー ID です。 |
| LOGIN_MIN_PW_DUR_FAIL | INVREQ | 5 | ユーザーがパスワードを変更しようとしましたが、前のパスワード変更から次の変更までに必要な最低限の時間が経過していませんでした。 |
| LOGIN_MUST_CHG_PW | 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。 |
| LOGIN_ESM_FAILED | INVREQ | 33 | ESM がこの要求に失敗しました。 |

## ユーザーログアウト

この関数は、ユーザーを領域からログオフするために CSSF トランザクションまたは CESF トランザクションに入ったときに呼び出されます。

**関数**:

kxsec_logout

**用法**:

```
boolean kxsec_logout(
       char *username, int username_length)
```

**入力**:

username: ログオフするユーザーの名前

username_length: そのユーザー名の文字数

**出力**:

リターンコード:

TRUE: ユーザーは正常にログオフしました。

FALSE: ユーザーは正常にログオフしませんでした。

## リソースアクセス

下記の関数は、要求されたリソースが、提示されたユーザー名と要求されたアクセス権によってアクセス可能かを確認するために呼び出されます。この関数が呼び出されるのは、トランザクションがセキュリティーチェックが有効 (KIXxxxSEC=YES) なリソースの使用を要求したときです。リソースタイプと、アクセス要求に対して求められるアクセス権の詳細については、第 8 章を参照してください。各リソースタイプのセキュリティーチェックを有効および無効にする環境変数については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

**関数**:

kxsec_rescheck

**用法**:

```
boolean kxsec_rescheck(
       enum asset_type asset_type_req,
       char *asset_name, int asset_name_length,
       char *username, int username_length,
       enum access_type access_type_req)
```

**入力**:

asset_type - KIX-FILES などの Sun MTP の資産のタイプ

asset_name - データセット名などの、Sun MTP の資産の名前

asset_name_length - その資産名の文字数

username - アクセスを確認するユーザーの名前

username_length - そのユーザー名の文字数

access_type - WriteAccess などの、要求されているアクセス権

**出力**:

リターンコード:

TRUE: ユーザーには、資産への要求されたアクセス権が付与されています。

FALSE: ユーザーには、資産への要求されたアクセス権が付与されていません。

**例**: 領域に対して KIXFCTSEC=YES と設定されている場合、VSAM ファイル PRODDATA に対する WRITE 要求は、そのユーザー名の書き込みアクセス権を使用して、PRODDATA という名前の KIX-FILES リソースタイプにアクセスを要求します。許可された場合は、この関数は TRUE を返し、拒否された場合は FALSE を返します。

**関数**:

kxsec_rescheck_q

**用法**:

```
boolean kxsec_rescheck_q(
       enum asset_type resource_type_req,
       char *resource_name, int resource_name_length,
       char *resource_class, int resource_class_length,
       char *username, int username_length,
       boolean log_this,
       enum access_type access_type_req)
```

**入力**:

resource_type_req - KIX-FILES などの、Sun MTP のリソースタイプ

resource_name - データセット名などの、リソース名

resource_name_length - リソース名の文字数

resource_class - ユーザー定義のリソースタイプの名前

resource_class_length - ユーザー定義のリソースタイプの文字数

username - アクセスを確認するユーザーの名前

username_length - ユーザー名の文字数

log_this - ログ記録を行なっている場合は TRUE、行なっていない場合は FALSE

access_type_req - WriteAccess などの、要求されているアクセス権

**出力**:

リターンコード:

TRUE: ユーザーには、リソースへの要求されたアクセス権が付与されています。

FALSE: リソースへの要求されたアクセス権がユーザーに付与されていないか、問題が発生しました。

**注意**:

この関数のコードでは、次の事項を考慮する必要があります。

resource_type_req は、resource_class が null であるか、resource_class_length がゼロである場合にのみ、使用してください。

## リソース確認結果のキャッシュの開始と停止、およびキャッシュのクリア

下記の関数は、セキュリティー確認チェックの結果を保持する必要がある場合に使用できます。確認結果をキャッシュで保持すると、トランザクションが同じリソースまたはアクセス権を何度も呼び出すことがなくなります。たとえば、同じトランザクションの中で、同じ VSAM データセットに対し、READ 要求を複数回実行する場合がこれにあたります。この読み取り権が確認されると、それ以降、トランザクションが同じ VSAM データセットに対して READ 要求を行なった場合、要求ごとに再確認することなく読み取りが許可されます。

「キャッシュ処理を開始」する関数 (kxsec_docache) は、トランザクションサーバーの起動時に呼び出されます。この関数は、ユーザー出口に対し、そのトランザクションサーバーからのリソース確認要求の結果を保存するよう指示します。

**関数**:

kxsec_docache

**用法**:

```
void kxsec_docache()
```

**入力**:

なし

**出力**:

なし

「キャッシュをクリア」する関数 (kxsec_clearcache) は、CEMT PERFORM SECURITY REBUILD が実行されたときに呼び出され、保存していた結果をすべて廃棄するようにユーザー出口に指示します。これにより、次のトランザクションでのリソース要求が再び確認の対象となり、その結果が保存されます。

**関数**:

kxsec_clearcache

**用法**:

```
void kxsec_clearcache()
```

**入力**:

なし

**出力**:

なし

「キャッシュ処理を停止」する関数 (kxsec_dontcache) は、現在、Sun MTP では呼び出されませんが、将来的には、対象のサーバーで結果をそれ以上キャッシュしないようにする場合に使用される可能性があります。このキャッシュ処理はデフォルトでは無効になっているため、この呼び出しは kxsec_docache() の呼び出しによってキャッシュ処理が明示的に有効になっている場合にのみ必要です。

**関数**:

kxsec_dontcache

**用法**:

```
void kxsec_dontcache()
```

**入力**:

なし

**出力**:

なし

セキュリティーユーザー出口では、これらの結果を保存するために、ローカルな静的記憶領域を使用する必要があります。共有メモリーの使用は、不必要で望ましくありません。保存された結果は、キャッシュが有効なローカルトランザクションサーバープロセスの外から認識する必要がないからです。

---

**注 –** セキュリティーユーザー出口はまた、確認要求のキャッシュをサポートしないほかの Sun MTP サーバープロセス (たとえば、`unikixstrt`) から呼び出されることもあります。それらのサーバーはこれらの「キャッシュ処理を開始」する関数や「キャッシュをクリア」する関数を呼び出さず、すべてのリソース要求に対して、完全なセキュリティー確認を行うことを想定しています。したがって、セキュリティーユーザー出口は、そのサーバープロセスで「キャッシュ処理を開始」する関数が呼び出され、「キャッシュ処理を停止」する関数が呼び出されていない場合を除き、前の呼び出しについての情報を保存しません。

---

# コーディングのガイドライン

これらのセキュリティー出口にコードを作成する場合は、一定の制限に従う必要があります。Sun MTP は、領域の起動時やトランザクションの実行時に、これらの関数をその複数のシステムサーバーから呼び出します。特に、`unikixstrt` は、起動プロセスの中で、通常の CICS またはバッチ VSAM API サービスが呼び出せるようになる前の初期の段階で、これらの関数を呼び出します。したがって、セキュリティーユーザー出口コードは、これらのサービスに依存せずに、それ自体のセキュリティー規則をロードできる必要があります。このようなセキュリティー規則として、たとえば、ユーザー出口にハードコードされたテーブルを用意したり、ライン順編成ファイルからセキュリティーユーザー出口のローカルなバッファーに読み込んだりすることが考えられます。

セキュリティーユーザー出口で CICS API を使用する場合にこのような制約があるのは、セキュリティーフックがトランザクションの使用の範囲を超えているからです。次の表に、このようなセキュリティーフックを持つサーバーとプロセス、およびそれらが行う呼び出しを示します。

**表 10-5** セキュリティーフックを持つ Sun MTP プロセス

| サーバー/プロセス | 呼び出し |
|---|---|
| `unikixtran` | ログイン、ログアウト、およびすべてのリソース |
| `unikixCEMT` (`CEMT` トランザクション) | リソース (KIX-COMMANDS、UNIX-APPLS) |
| `unikixstrt` | ログイン (KIXSECDFLTUSER)、リソース (UNIX-APPLS)<br>プリンタ TCT のログインユーザー名 |
| `unikixFMS` (`CFMS` トランザクション) | リソース (KIX-FILES) |
| `unikixRED` (`CRED` トランザクション) | リソース (KIX-FILES) |

# SNT を使用したユーザー名とパスワードの確認

独自のユーザー名とパスワード確認規則の代わりに、またはそれに加えて、セキュリティーユーザー出口コードから kxchksntpw() 関数を呼び出すことができます。この関数は、SNT エントリで要求されたユーザー名を検索し、その要求されたパスワードをそのエントリに求められるパスワードと比較します。これらのパスワードが一致した場合、kxchksntpw は要求を有効だと確認して、その通知を返すとともに、新しいパスワードが入力された場合は、それを使用して SNT エントリを更新します。パスワードが一致しない場合またはユーザー名が SNT エントリのどれとも一致しない場合、kxchksntpw は、要求を拒否し、その通知を返します。

**用法**:

```
boolean kxchksntpw(username, password, result, newpw, newpw_length)
```

**入力**:

username: ログオンするユーザーの名前 (8 文字。不足分には空白文字が埋め込まれます)

password: そのユーザーのパスワード (8 文字。不足分には空白文字が埋め込まれます)

newpw: そのユーザーの新しいパスワード (入力された場合。8 文字。不足分には空白文字が埋め込まれます)

newpw_length: 新しいパスワードの文字数。入力されなかった場合はゼロ (4 バイトのバイナリの整数)

**出力**:

リターンコード:

1: ユーザー名とパスワードは有効です。新しいパスワードが入力された場合は、パスワードが更新されました。

0: ユーザー名とパスワードが有効ではありません。ユーザー名またはパスワードのいずれかが無効です。

次のコードは、C で kxchksntpw() 関数を呼び出す方法を示します。

```
if (kxchksntpw (username, password, result, newpw, newpw_length))
          {                 /* other processing              */
          return TRUE;      /* username & password valid     */
          }
      else
          {                 /* other processing              */
          return FALSE;     /* username or password invalid  */
          }
```

### ▼ 新しいセキュリティーユーザー出口ルーチンを作成する

1. ディレクトリを `$UNIKIX/src/security` に変更します。
2. `$UNIKIX/src/security/kxsec_exitsTMPL.c` ファイルを `kxsec_exits.c` にコピーします。
3. エディタを使用して `$UNIKIX/src/security/kxsec_exits.c` ファイルに必要な変更を加えます。
4. ユーザーオブジェクトファイルを作成します。

```
$ make kxsec_exits.o
```

5. カスタマイズツールの使用を終了し、Sun MTP 実行可能ファイルを再構築する準備が整ったら、次の手順に従います。

   a. `$UNIKIX/src` ディレクトリに移動します。

   b. 次のコマンドを実行します。

```
$ make security
```

   このコマンドは `kixinstall` によって生成されたメークファイルを使用して、すべての Sun MTP 実行可能ファイルのほか、セキュリティーユーザー出口のコードを呼び出す実行可能ファイルも構築します。

これで、デフォルトのセキュリティーユーザー出口ルーチンを変更しました。

# ソケットユーザー出口のカスタマイズ

ソケットユーザー出口 `kxusrexit_in_socket` は、`$UNIKIX/src/socket/kxsktxit.c` ファイルにあり、領域に供給するさまざまなメッセージ形式を柔軟にサポートします。ソケットユーザー出口は、ソケット要求が行われたときに呼び出されます。この出口は、初期ソケットデータを読み込んで、メッセージの詳細をトランザクション処理プログラムに返します。

このユーザー出口をカスタマイズすれば、次の機能を実行できます。

- 初期ソケットメッセージで、最高 32,732 バイトのデータを許容します。
- さまざまな量のデータを適切な CICS/COBOL プログラムに渡します。

- ソケットクライアントからのメッセージに対応して実行するトランザクションを指定します。たとえば、受信したデータ量に応じたトランザクションをスケジューリングします。
- 初期ソケットメッセージの形式を変更します。たとえば、メッセージにはデータのみが入り、データの内容により TRANSID をセットします。

kxusrexit_in_socket() ユーザー出口は、次の出力パラメータを提供する必要があります。

| パラメータ | 説明 |
|---|---|
| Transaction Id | 関数から返された時点で、この 4 文字のフィールドには、PCT で定義されたトランザクション識別子が入っている必要があります。必須です。 |
| Data Buffer | 関数から返された時点で、ソケットレコードの CLIENT-IN-DATA フィールドにあるアプリケーションプログラムに渡すデータをポイントします。データを渡さない場合、このフィールドは NULL になります。 |
| Data Length | 関数から返された時点では、データバッファーに存在するバイト数が入っています。トランザクション処理プログラムは、これらのバイトをデータバッファーから抽出し、必要に応じて次の語との境界まで NULL を埋め込んでから、アプリケーションプログラムに供給される COMMAREA にデータを保存します。最大データ長は 32,732 バイト、デフォルトのデータ長は 35 バイトです。<br>渡されるデータがない場合は、フィールドをゼロ (0) にする必要があります。 |
| Start Type | 関数から返された時点で、このフィールドは KC、TD、IC のいずれかになります。<br>KC: トランザクションを即座にスケジュールします。これがデフォルトです。<br>TD: 一時データ。名前がトランザクション ID の出力パラメータと一致する名前のキューに、ソケットレコードが書き込まれます。このキューをパーティション内キューとして定義します。これは、そのキューと同じ名前を持つトリガートランザクションと、1 というトリガーレベルを持つことができます。<br>IC: 間隔制御。トランザクション ID 出力パラメータで指定したトランザクションは、Interval パラメータに指定された時刻に開始するようにスケジューリングされます。 |
| Interval | 関数から返された時点で、このフィールドには、時刻の値が *HHMMSS* という形式で入っています。この値は、トランザクション ID 出力パラメータ内に指定されたトランザクションの開始までの経過時間を指定します。このフィールドは、Start Type が IC の場合だけ指定する必要があります。デフォルト値は「000000」です。 |

次の例は、kxuserexit_in_socket() ユーザー出口の典型的なサンプルコードを示します。

**コード例 10-7**　　ソケットユーザー出口

```
/* Since the default initial socket message has no way of telling
the receiver how much data is being sent (i.e. there is no leading
length indicator in the data nor is there any trailing pattern to
look for), we attempt to receive all the data in one go.If the
length of data being received can be determined, 'recv' should be
coded to be in a loop till all data is received or an error
condition occurs.
*/
length_received = recv (psocket_fd, read_buffer,
                                    sizeof(read_buffer), 0);
if (length_received < 0)
       {
       kxerror1(E_2327,E_PRINT+E_PRINTF, "unikixtrans", errno);
       return(-1);
       }
if (length_received == 0) /*possibly, the client has gone away or*/
       {                                /*closed the connection*/
       return(-2);
       }
first_comma = memchr (read_buffer, ',', length_received);
if (first_comma == 0)     /*only transid specified */
       {len_transid = length_received;
       }
else
       {len_transid = first_comma - read_buffer;
       }
/* Validate transid */
if ((len_transid > LEN_TRANID) || (len_transid == 0))
       {
       kxerror (E_2328,E_PRINT+E_PRINTF, "unikixtran");
       return(-1);
       }
memcpy (ptransid, read_buffer, len_transid);
```

ソケットサポートの詳細は、『Sun Mainframe Transaction Processing ソフトウェア開発者ガイド』を参照してください。

# SSL ユーザー出口のカスタマイズ

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。このユーザー出口では、たとえば、独自の証明書取消し確認に基づいて、クライアント証明書を、承認または拒否するコードを記述できます。

ユーザー出口のソースコードと共有ライブラリを作成するメークファイルは、$UNIKIX/src/socket ディレクトリにあります。ソースファイルは kxsslxit.c で、メークファイルは、makefile.xit です。ソースファイルを変更すれば、ご使用の環境に合わせてユーザー出口をカスタマイズすることができます。また、メークファイルを使用して、新しい共有ライブラリを作成できます。ただし、カスタマイズしたソースをコンパイルするには、NSS および NSPR ソースのインクルードファイルの場所が LD_LIBRARY_PATH 変数で指定されている必要があります。不明な点については、ご購入先にお問い合わせください。

---

**注 –** ソースのサンプルを変更する場合は、NSS および NSPR ソースのインクルードファイルが必要です。

---

クライアント証明書検証を実行する共有ライブラリの関数は、Exit_VerifyCertificate() です。SSL ハンドシェーク時、unikixssl はこの関数を呼び出し、ポインタ (CERTCertificate *cert) をクライアント証明書に渡します。この関数は、クライアント証明書が承認されると SECSuccess を、拒否されると SECFailure を返します。

## ▼ SSL ユーザー出口をカスタマイズする

1. **既存の** $UNIKIX/lib/libkxsslxit.so **共有ライブラリをバックアップします。次に例を示します。**

```
$ mv $UNIKIX/lib/libkxsslxit.so $UNIKIX/lib/libkxsslxit.so.bak
```

2. **ディレクトリを SSL ユーザー出口ソースファイルの場所に変更します。**

```
$ cd $UNIKIX/src/socket
```

3. **ソースファイル** kxsslxit.c **を開き、** Exit_VerifyCertificate() **関数への変更を行います。**

4. ファイルを保存します。

5. 共有ライブラリを作成します。

```
$ make -f makefile.xit
```

これで、$UNIKIX/src/socket ディレクトリに libkxsslxit.so が作成されます。

6. 新しい共有ライブラリを $UNIKIX/lib ディレクトリにコピーします。

```
$ cp libkxsslxit.so $UNIKIX/lib
```

# ISC ユーザー出口のカスタマイズ

この節では、次のユーザー出口を説明します。

- トランザクション経路指定のための変換ルーチン
- 機能シップ、分散プログラムリンク (Distributed Program Link: DPL)、非同期トランザクション開始 (Asynchronous Transaction Initiation: ATI) など、その他の ISC タイプ用の変換ルーチン。分散トランザクション処理 (Distributed Transaction Processing: DTP) は含みません。
- 動的なトランザクション経路指定

## トランザクション経路指定のための変換ルーチン

トランザクション経路指定では、Sun MTP が、USERAREA、COMMAREA、または端末入出力領域 (TIOA) 内の文字を対象のコードセットに変換できます。変換ルーチンは、デフォルトで、USERAREA と COMMAREA のディスプレイデータを想定します。すべてのトランザクションコードに対し、ルーチンは発信領域で ASCII から EBCDIC へ、受信領域で EBCDIC から ASCII へと両方の領域を変換します。この変換テーブルは、3270 データストリームで使用されているのと同じ変換テーブルです。変換テーブルは、$UNIKIX/src/convert/kxcnvtbl.c ソースファイル内の asc2ebcd と ebcd2asc にあります。

kxusrxlt.c 変換ルーチンには、次の項目を指定する引数が渡されます。

- 領域の始まり
- 領域の終わり
- 領域の長さ
- 変換される領域のタイプ (USERAREA、COMMAREA、またはTIOA)
- 送信方向 (送信または受信)
- トランザクションコード
- 使用中の遠隔コードページ

COMMAREA または USERAREA に、ディスプレイデータではないものが入っている場合、ユーザーは、ディスプレイデータ以外のデータを処理するようにユーザー変換ルーチンを変更できます。これはたとえば、バイナリデータフィールドを変換するのではなく、そのままコピーする場合に使用します。

## ▼ バイナリフィールドをコピーする

1. **ディレクトリを $UNIKIX/src/convert に変更します。**
2. **ユーザー変換モジュール $UNIKIX/src/convert/kxusrxlt.c のバックアップコピーを作成します。**
3. **元のモジュールをエディタで開きます。**
4. **デフォルトの変換は、EBCDIC から ASCII です。変換する必要がない場合は、kxusrxlt.c でコメントアウトされた memcpy ルーチンを使用します。**
5. **変換が必要なデータ内にバイナリフィールドが埋め込まれている場合は、Table Manager – 拡張テーブルメニューからデータ変換テンプレート (CVT) 画面にアクセスし、その領域をレコードとして定義します。**
6. **CVT で定義したリソース名を使用して kxusrxlt.c の kxcvt ルーチンを呼び出します。**
7. **更新した $UNIKIX/src/convert/kxusrxlt.c の内容をディスクに保存します。**
8. **次のコマンドを入力して、更新したモジュールをコンパイルし、それに依存するすべての実行可能ファイルを再構築します。**

```
$ cd $UNIKIX/src
$ make unikixtran
```

kxusrxlt.c ユーザー出口には、特定のトランザクション用のサンプル変換コードが用意されています。(このコードは、if 文の中では無効です)。

## ほかの ISC タイプのための変換ルーチン

Sun MTP 領域と遠隔システムの間でレコードを渡すとき、領域 は CVT を使用して、2 つのシステム間で渡されるデータを送信側では ASCII から EBCDIC へ、受信側では EBCDIC から ASCII へ変換します。CVT が個々のレコードの構造を定義するので、Sun MTP ルーチンは、変換の必要なレコードのフィールドだけを正しく変換できます。複雑なレコード構造では、CVT ユーザー出口の使用が必要になることがあります。`kxcvtxlt.c` ユーザー出口には、特定のトランザクション用のサンプル変換コードが用意されています。(このコードは、`if` 文の中では無効です)。

特定のレコードを処理するように変更できるサンプル CVT ユーザー出口が、`$UNIKIX/src/convert/kxcvtxlt.c` に用意されています。この出口を呼び出すには、50 ～ 80 のタイプで、CVT 内のテンプレートエントリをコーディングします。

この出口には、次の項目を指定する引数が渡されます。

- リソースタイプ
- リソース名
- テンプレートのシーケンス番号
- フィールドのアドレス
- フィールドの長さ
- 方向 (アウトバウンドまたはインバウンド)
- 遠隔コードページ
- テンプレートエントリタイプ

## ▼ ユーザー出口を変更する

1. **ディレクトリを `$UNIKIX/src/convert` に変更します。**
2. **ユーザー出口 `$UNIKIX/src/convert/kxcvtxlt.c` のバックアップコピーを作成します。**
3. **元のモジュールをエディタで開きます。**
4. **対象のレコードを正しく変換するように出口を変更します。**
5. **更新した `$UNIKIX/src/convert/kxcvtxlt.c` の内容をディスクに保存します。**
6. **次のコマンドを実行して、更新したモジュールをコンパイルし、それに依存するすべての実行可能ファイルを再構築します。**

```
$ cd $UNIKIX/src
$ make unikixtran
```

# 動的トランザクション経路指定のユーザー出口

動的トランザクション経路指定 (DTR) のユーザー出口は $UNIKIX/src/isc/kxdynrte.c です。これが呼び出されるのは、LU6.2 ALLOCATE コマンドが実行される直前です。この出口を使用すると、要求の経路指定先のシステムを変更することができます。この出口は、次の目的で使用できます。

- 一次領域が使用できないときに、要求を別の領域に経路指定し直す
- 複数の遠隔領域間で作業負荷を分散する
- 必要なほかのタスクを実行する

たとえば、システム SYS1 で、トランザクション TR01 を実行するように領域が構成されているとします。しかし、SYS1 が適切なターゲットシステムではない場合、このトランザクションを別のシステムに経路指定できるような柔軟性が必要です。利用度、負荷、またはほかのシステム特性に基づいて TR01 を SYS2 に経路指定するユーザー出口をコーディングできます。このユーザー出口には、TR01 の呼び出しに関連付けられた 3270 データストリームが渡されます。このデータストリームを調べると、トランザクションの経路指定先を特定できます。たとえば、データストリームに TR01 SYS1 が含まれている場合、トランザクション TR01 は SYS1 に経路指定され、データストリームに TR01 SYS2 が含まれている場合は SYS2 に経路指定されます。

## ▼ 動的トランザクション経路指定のユーザー出口を変更する

1. ディレクトリを $UNIKIX/src/isc に変更します。
2. ユーザー出口 $UNIKIX/src/isc/kxdynrte.c のバックアップコピーを作成します。
3. 元のモジュールをエディタで開きます。
4. 必要なタスクを実行するように出口を変更します。
5. 更新した $UNIKIX/src/isc/kxdynrte.c の内容をディスクに保存します。
6. 次のコマンドを実行して、更新したモジュールをコンパイルし、それに依存するすべての実行可能ファイルを再構築します。

```
$ cd $UNIKIX/src
$ make unikixtran
```

# 端末非存在状態の出口

Sun MTP と CICS の間で、START コマンドまたは一時データ自動トランザクション開始 (TDATI) を使用して、指定した端末のトランザクションを起動できます。端末が存在しない場合、START コマンドが失敗して TERMIDERR というメッセージが発せられ、端末がログオンするまで、TDATI がその要求をキューに入れます。

START 要求が実行されたときに、トランザクション経路指定された端末が Sun MTP に存在しないことがあります。この場合、要求を拒否するのではなく、端末が存在する領域を Sun MTP に対して通知する出口が提供されます。これにより、START 要求が、遠隔システムに渡されます。遠隔システムに端末があれば要求はすぐに実行され、それ以外の場合はキューに登録されます。

この出口は、SIT の DefaultTOR フィールドを使用します。サンプルの出口は、DefaultTOR フィールドを確認し、それが設定されている場合は、要求をそのシステムにシップします。このフィールドが設定されていない場合、要求は TERMIDERR エラーを発して失敗するか、キューに登録されます。

サンプルの端末非存在状態の出口は $UNIKIX/src/terminal/kxtnfxit.c にあります。この出口には、次の項目を指定する引数が渡されます。

- トランザクションが開始された状況
- 要求を実行したトランザクションが開始された状況
- トランザクション名と端末名。端末名は、要求を実行した端末のものです。
- DefaultTOR フィールド
- 要求の発信元システムの *netname* と *sysid* (要求が機能シップまたはトランザクション経路指定によって開始された場合)
- 要求の送信先遠隔システムの *netname* または *sysid* のための 2 つの戻りフィールド

渡された引数を使用して、要求のシップ先システムの *sysid* または *netname* を戻りフィールドに保存します。これらのフィールド内の値は、TCT のシステムエントリに指しておく必要があります。*sysid* は、SysID フィールドのエントリに一致する必要があり、*netname* は、Ptnr_LU Name フィールドのエントリに一致する必要があります。情報を保存したあと、次のどれかの値が返されます。

- 4: *netname* が変更された場合
- 8: *sysid* が変更された場合
- 0: 要求が失敗した場合

## ▼ 端末非存在状態の出口を変更する

1. ディレクトリを `$UNIKIX/src/terminal` に変更します。
2. ユーザー出口 `$UNIKIX/src/terminal/kxtnfxit.c` のバックアップコピーを作成します。
3. 元のモジュールをエディタで開きます。
4. ATI 要求が、要求された遠隔システムに正しくシップされるように出口を変更します。
5. 更新した `$UNIKIX/src/terminal/kxtnfxit.c` の内容をディスクに保存します。
6. 次のコマンドを実行して、更新したモジュールをコンパイルし、それに依存するすべての実行可能ファイルを再構築します。

```
$ make unikixtran
```

# レコード処理ルーチンのカスタマイズ

**注 –** この節は、Server Express を使用する、COBOL で記述されたアプリケーションに当てはまります。

付属のテンプレートを変更するか、新しいルーチンを作成して、入出力レコード処理ルーチンのカスタマイズができます。

すべてのテンプレートのソースファイルは、`$UNIKIX/src/record` ディレクトリにあります。`line`、`record`、および `recordv` ルーチンは C 言語で書かれており、`mfrcd` ルーチンは COBOL で書かれています。これら 2 種類のレコード処理ルーチンの作成手順は互いに異なっています。詳細はこの節で説明します。

実行可能ファイルは、次のディレクトリにあります。

- C 言語の実行可能ファイルは、`$UNIKIX/bin/build` ディレクトリにあります。
- COBOL の実行可能ファイルは、`$UNIKIX/bin` ディレクトリにあります。また `$UNIKIX/bin/build` ディレクトリには、入力用と出力用の 2 つのシェルスクリプトがあります。これらは入力パラメータを実行可能ファイルに渡します。

これらのルーチンを変更する場合は、カスタマイズした方の実行可能ファイルを `$UNIKIX/local/bin` ディレクトリに置きます。これは、領域がこのディレクトリを検索してから `$UNIKIX/bin/build` ディレクトリを検索するからです。

レコード処理ルーチンをカスタマイズするときに使用する、
$UNIKIX/src/makefile ファイル内の定義済み変数を次に示します。

| 変数 | 値 |
|---|---|
| USERSRCDIR | ソースファイルとオブジェクトファイルのデフォルトのホームディレクトリである $UNIKIX/src に設定します。 |
| USERRECORD | $USERSRCDIR/record に設定します。 |
| BINDIR | カスタマイズ済み実行可能ファイルのデフォルトのホームディレクトリである $UNIKIX/local/bin に設定します。 |

## ▼ 既存のレコード処理ルーチンを変更する

1. **エディタを使用して、たとえば linein.c などの適切なファイルに必要な変更を加えます。**

2. **実行可能ファイルをコンパイルして構築します。これには次のコマンドを入力します。**

```
$ cd $UNIKIX/src
$ make line.in
```

3. **通常 Sun MTP は、line、record、および recordv ルーチンの内部バージョンを使用するため、$UNIKIX/local/bin にあるカスタマイズしたバージョンを新しい名前に変更する必要があります。次に例を示します。**

```
$ cd $UNIKIX/local/bin
$ mv line.in myline.in
```

**注 –** mfrcrd ルーチンに対しては、この手順は不要です。

4. **領域に対して、カスタマイズしたルーチンを使用するよう指示します。**

   a. unikixbld -r *format* **を実行します。**

   *format* 引数として、ルーチン名をたとえば myline のように入力します。拡張子は、使用しないでください。

   b. **レコードエディタを使用します。**

   - 入力形式をカスタマイズした場合は、レコードエディタの「Build VSAM File」画面の Format フィールドにファイル名を指定します (たとえば myline)。

     .in 拡張子は、入力しないでください。

- 出力形式をカスタマイズした場合は、レコードエディタの「Build Sequential File」画面の Format フィールドにファイル名を指定します。

  .out 拡張子は、入力しません。

## 新しいレコード処理ルーチンの作成

まったく新しいレコード処理タイプをサポートするには、2 つの実行可能ファイルを作成する必要があります。COBOL ソースファイルを作成する場合は、新しいレコード入出力形式のために 2 つのシェルスクリプトを作成する必要があります。

次の手順に従って、myrecord という名前の新しいレコード形式を定義する 2 つのレコード処理ルーチン (myrecord.in と myrecord.out) を新しく作成します。

## ▼ 新しいレコード処理ルーチンを作成する

1. **ファイル $UNIKIX/src/makefile の中の 3 つの変数を、追加するユーザーモジュールのパス名を持つように更新する必要があります。**

   a. **ソースファイルは、USERSOURCES 環境変数によって位置が示され、$USERRECORD ディレクトリに存在する必要があります。次に例を示します。**

```
USERSOURCES = \
       $(USERRECORD)/myrecordin.c \
       $(USERRECORD)/myrecordout.c
```

   b. **オブジェクトファイルは、USEROBJECTS 環境変数によって位置が示され、$USERRECORD ディレクトリに存在する必要があります。次に例を示します。**

```
USEROBJECTS = \
       $(USERRECORD)/myrecordin.o \
       $(USERRECORD)/myrecordout.o
```

   c. **実行可能ファイルは、USEREXECUTABLES 環境変数によって位置が示され、$BINDIR ディレクトリに存在する必要があります。次に例を示します。**

```
USEREXECUTABLES = \
       $(BINDIR)/myrecord.in \
       $(BINDIR)/myrecord.out
```

2. システムに追加した実行可能ファイルのターゲットを $UNIKIX/src/makefile に追加します。

   たとえば、myrecord.in 実行可能ファイルの場合は、次のように追加します。

```
$(BINDIR)/myrecord.in myrecord.out
       rm -f $@
       $(CC) $(CFLAGS) -o $@ $?
```

3. 既存のテンプレートをコピーします。

   たとえば、次のコマンドを入力して、既存のファイルから line を myrecord ソースファイルにコピーします。

```
$ cp $UNIKIX/src/record/linein.c $UNIKIX/src/record/myrecordin.c
$ cp $UNIKIX/src/record/lineout.c $UNIKIX/src/record/myrecordout.c
```

4. エディタを使用して、myrecord ソースファイルを変更します。

5. COBOL ユーザーモジュールの場合は、追加のシェルスクリプトのセットもカスタマイズする必要があります。

   既存のテンプレートスクリプトファイルを myrecord スクリプトファイルにコピーします。

```
$ cp $UNIKIX/src/record/mfrcd.in $UNIKIX/src/record/myrecord.in
$ cp $UNIKIX/src/record/mfrcd.out $UNIKIX/src/record/myrecord.out
```

6. 次のコマンドを実行して、myrecord ソースファイルをコンパイルします。

```
$ cd $UNIKIX/src
$ make myrecord.in
$ make myrecord.out
```

7. 手順 6 で作成した実行可能ファイルを実行するようにスクリプトファイルを編集します。

   $UNIKIX/local/bin/myrecordin
   $UNIKIX/local/bin/myrecordout

8. 実行可能ファイルは、USEREXECUTABLES 環境変数によって位置が示され、$BINDIR ディレクトリに存在する必要があるので、スクリプトファイルを $USEREXECUTABLES 定義に追加します。

```
USEREXECUTABLES = \
       $(BINDIR)/myrecordin \
       $(BINDIR)/myrecordout \
       $(BINDIR)/myrecord.in \            *defined in Step 1
       $(BINDIR)/myrecord.out             *defined in Step 1
```

9. mfrcdin と mfrcd.in のエントリから COBOL のターゲットエントリをコピーし、適切な名前に変更します。

10. 次のコマンドを実行して、myrecord ソースファイルをコンパイルします。

```
$ cd $UNIKIX/src
$ make myrecordin
$ make myrecordout
$ make myrecord.in
$ make myrecord.out
```

# 回復プロセッサユーザー出口

回復プロセッサユーザー出口の主な目的は、レコードが論理的に削除されたことを示す、ESDS データセットレコードの特別なタグを付けることです。Sun MTP は、そのレコードを削除しません。このため、ユーザーアプリケーションが、レコードを削除すべきかどうか、それに合わせてデータセットを処理するかどうかを判断する必要があります。

ソース関数は、$UNIKIX/src/recovery/kxesdsxlt.c に位置し、トランザクションバックアウト操作中に呼び出されます。

# 回復バックアウト関数呼び出し

トランザクションが ESDS データセットに書き込み操作を実行した場合、暗黙的関数 kxesds_backout_user_exit は、動的なトランザクションバックアウト中に、次の情報とともに呼び出されます。

| | |
|---|---|
| レコードバッファー | 削除済みにマークする必要があるデータレコードへのポインタ (文字ポインタ) |
| 長さ | レコードの長さ (整数) |
| データセット名 | FCT に定義されているデータセットの名前 (文字ポインタ) |
| リターンコード | 回復プロセッサに戻されるリターンコード (整数) |

この関数を変更したり、独自のソース行を追加したりすることで、ユーザーアプリケーションに対して論理的な削除を示すレコードバッファーの内容を変更することができます。

ソースモジュールには、データレコードをどのように変更できるかという例が提供されています。独自のロジックを記述して、使用している環境のニーズを満たすことができます。また、条件式を含むロジックを使用して、record_dataset (ESDS データセット名) の値を確認することもできます。

# バックアウト関数からのリターンコード

リターンコードの初期値は 0 に設定します。record_buffer の内容の変更に成功したあと、更新されたデータバッファーの内容が ESDS ファイル上で物理的に変更可能になるように、リターンコードは 4 に設定する必要があります。リターンコードの値が 4 以外の場合は、更新があった場合も、その内容は ESDS ファイルに送られず、バックアウト関数が実行される前のレコードの元の内容が保存されます。

# 回復プロセスのカスタマイズ

kxesdsxlt.c ソースファイルを変更後、回復プロセッサ (unikixrcv) を再構築する必要があります。

## ▼ 回復プロセッサを再構築する

1. **カスタマイズした** kxesdsxlt.c **ソースファイルが配置されているディレクトリに変更します。**

```
$ cd $UNIKIX/src
```

2. kxesdsxlt.c **ソースファイルをコンパイルし、**$UNIKIX/src **ディレクトリの** makefile **を使用して** unikixrcv **実行可能ファイルを再リンクします。**

```
$ make unikixrcv
```

これは、更新した unikixrcv モジュールを、makefile の BINDIR 環境変数で指定されたディレクトリに配置します。通常は、$UNIKIX/local/bin ディレクトリです。

3. **領域を再起動し、変更を有効にします。**

# 定義済みのコードページ変換テーブルのカスタマイズ

unikixmain (または kixstart) に対して -A オプションまたは -B オプションを使用して領域を起動するときは、$UNIKIX/lib ディレクトリに用意されている EBCDIC から ASCII へのコードページ変換テーブルのいずれかを指定して、Sun MTP が EBCDIC から ASCII または ASCII から EBCDIC に文字データを変換する方法も設定します。

たとえば、次のオプションで領域を起動するとします。

```
$ unikixmain -A IBM-037toIBM-850.table
```

領域によって所有されている ASCII 文字データは、1 バイト文字の ASCII コードページ IBM-850 で保持されているとみなされ、適切な時点で 1 バイト文字の EBCDIC コードページ IBM-037 に変換されたり、このコードページから変換されたりします。この変換が発生するのは、EBCDIC 形式のデータを受信することになっている TN3270 端末に、Sun MTP の領域が文字データを送信するときなどです。

同じように、次のオプションで領域を起動するとします。

```
$ unikixmain -B IBM-930-IBM-932.table
```

領域によって所有されている ASCII 文字データは、2 バイト文字の ASCII コードページ IBM-932 で保持されているとみなされ、適切な時点で EBCDIC の 2 バイト文字のコードページ IBM-930 に変換されたり、このコードページから変換されたりします。

---

**注 –** これらのテーブルは文字データの変換にのみ使用されます。3270 データストリームを構成する 3270 コマンドや命令などの、その他のデータは、ASCII および EBCDIC の 3270 データストリームの規則に従って、領域によって自動的に変換されます。

---

定義済みのコードページ変換テーブル (接尾辞 `.table` が付いている) は、`$UNIKIX/lib` ディレクトリにあります。

## テーブルを変更する理由

ほとんどの場合、いずれかの定義済みのコードページ変換テーブルによってアプリケーションの要件が満たされます。しかし、場合によってはテーブルをカスタマイズしなければならないこともあります。たとえば、バイナリの EBCDIC データが送信される ATM 機などの、Sun 以外のハードウェアデバイスと通信する必要があるが、ユーザーが選択したコードページ内にそれに対応するものがない場合があります。常に、アプリケーションの要件を念入りに調査して、定義済みのテーブルがニーズを満たすことができるかどうかを判定してください。

---

**注意 –** 変換テーブルを正しくカスタマイズしないと、副次的な悪影響が生じることがあります。不明な点がある場合は、カスタマイズを実行する前にご購入先にお問い合わせください。

---

## Sun MTP のコードページ変換テーブルの形式

この節では、定義済みの 1 バイト文字および 2 バイト文字の変換テーブルの形式を説明します。

`-A` オプションとともに使用される 1 バイト文字テーブルのサイズは、256 バイトまたは 512 バイトです。`-B` オプションとともに使用される 2 バイト文字テーブルのサイズは、131,072 バイト (128K バイト) です。

## 長さが 256 の 1 バイト文字変換テーブル

-A オプションを使用して領域を起動するときに、サイズが 256 バイトである 1 バイト変換テーブルを指定すると、このテーブルは次のように使用されます。

値が X である EBCDIC 文字は、IBM-037toIBM-850.table を使用して EBCDIC から ASCII に変換されます。Sun MTP は、オフセット X の位置のバイトをこのテーブルから読み取ります。たとえば、Sun MTP が EBCDIC 値 0xC1 (EBCDIC IBM-037 で大文字の A) を変換する必要があるときは、テーブルのオフセット 0xC1 から値を読み取り、値 0x41 (ASCII IBM-850 で大文字の A) を取得します。

ASCII から EBCDIC に変換するため、Sun MTP はこのテーブルの逆のテーブルを自動的に構成します。このため、たとえば ASCII の 0x41 を EBCDIC の 0xC1 に変換できます。

## 長さが 512 の 1 バイト文字変換テーブル

-A オプションを使用する、長さが 512 の 1 バイト文字テーブルのサポートは従来のものであり、このテーブルは使用しないように強く推奨されています。これらのテーブルの最初の 256 バイトは、256 バイトのテーブルと同じ形式です。つまり、最初の 256 バイトは EBCDIC から ASCII への変換に使用できます。残りの 256 バイトは、ASCII から EBCDIC への明示的な逆方向マッピングです。512 バイトのテーブルを使用して領域を起動すると、Sun MTP は次の方法を使用して文字を ASCII から EBCDIC に変換します。

ASCII 文字の値 X が取得され、オフセット 0x100 + X の位置のバイトがテーブルから取り出され、必要な EBCDIC 文字が取得されます。たとえば、0x41 ASCII (ASCII の大文字の A) を変換するには、オフセット 0x100 + 0x41 の位置のバイト (オフセット 0x141 の位置のバイト) が取り出されます。通常は、これによって EBCDIC 値 0xC1 (EBCDIC の大文字の A) が取得されます。0x100 は 256 進数であることに注意してください。

**注意 –** これらの 512 バイトのテーブルでは明示的な順方向マッピングと逆方向マッピングが可能であるため、このテーブルは整合性のない状態になることがあります。この結果、EBCDIC から ASCII に変換されてから、その逆の変換が行われた文字は、元の EBCDIC 値にならないことがあります。

## 2 バイト文字変換テーブル

2 バイト文字変換テーブルの長さは 131,072 バイト (128K バイト) です。これらのテーブルは、1 バイト文字の変換と 2 バイト文字の変換に使用されます。

## 1 バイト文字の変換

-B オプションを使用して領域を起動し、2 バイト文字テーブルを指定すると、このテーブルが次のように使用されて 1 バイト文字が変換されます。

値が X である 1 バイト文字を EBCDIC から ASCII に変換するため、Sun MTP はオフセット 2 * X の位置で始まる 2 つのバイトを取り出し、ゼロである 1 番目のバイトを破棄します。1 バイト文字は 2 バイト文字と同じ形式でテーブルに格納され、それぞれが 2 バイトを占有するため、1 番目のバイトはゼロです。1 番目のバイトの値は 0x00 であり、2 番目のバイトは 1 バイトの値です。

たとえば、-B IBM-930-IBM-932.table を使用して領域を起動し、EBCDIC の大文字の A (値 0xC1) を変換する場合は、オフセット 2 * 0xC1 で始まる 2 つのバイト、つまり オフセット 0x182 で始まる 2 つのバイトが取り出されます。これらのバイトの値は 0x00 と 0x41 です。そして、1 番目のバイトは破棄され、ASCII の大文字の A である 0x41 が残ります。

## 2 バイト文字の変換

-B オプションを使用して領域を起動し、2 バイト文字テーブルを指定すると、このテーブルが次のように使用されて 2 バイト文字が変換されます。

値が X である 1 バイト文字を EBCDIC から ASCII に変換するため、Sun MTP はオフセット 2 * X の位置で始まる 2 つのバイトを取り出します。これらは、必要な ASCII 文字の上位バイトと下位バイトです。

たとえば、-B IBM-930-IBM-932.table を使用して領域を起動し、値が 0x42e0 である EBCDIC 文字 (全角のドル記号) を変換する場合は、オフセット 2 * 0x42e0、つまり オフセット 0x85c0 の位置の 2 つのバイトが取得されます。これらのバイトは 0x81 と 0x90 です。つまり、ASCII IBM-932 で全角のドル記号を示す ASCII 値 0x8190 です。

## 逆方向マッピング

1 バイト文字と 2 バイト文字の逆方向の変換、つまり ASCII から EBCDIC への変換を行う場合、Sun MTP は通常の逆のテーブルを構成して使用します。現在は、ASCII から EBCDIC に明示的にマッピングする 2 バイト文字テーブルはありません。

## ▼ コードページ変換テーブルをカスタマイズする

1. **使用している環境で変換テーブルをカスタマイズする必要があることを確認します。**
2. **カスタマイズする 1 バイト文字または 2 バイト文字のテーブルを選択し、新しい名前のファイルにコピーします。**

   たとえば、`IBM-930-IBM-932.table` ファイルをカスタマイズする場合は、これを `IBM-930-IBM-932.CUSTOM.table` などの新しい名前のファイルにコピーします。カスタマイズしたバージョンのテーブルが、製品に付属するデフォルトのバージョンと区別できなくなると、サポートの問題が発生する可能性があるため、この作業は重要です。
3. **この節に記載されているテーブルの形式の情報を利用し、16 進数エディタを使用して、変更する文字の値を修正します。**

   この手順を繰り返す必要がある場合に備えて、行なった変更とその理由の正確な記録を保存してください。
4. **新しいテーブルを `$UNIKIX/lib` ディレクトリにコピーします。**
5. **`unikixmain` に `-A` オプションまたは `-B` オプションを付けて領域を起動し、新しいテーブルを指定します。**

   次に例を示します。

```
$ unikixmain -B IBM-930-IBM-932.CUSTOM.table ...
```

# 複数のマッピングに関する問題

長さが 256 の 1 バイト文字テーブルと、2 バイト文字テーブルでは、ASCII から EBCDIC への逆方向のマッピングは、EBCDIC から ASCII への順方向のマッピングを逆にすることによって Sun MTP で処理されます。

複数の EBCDIC が同一の ASCII 文字にマッピングされていると、問題が発生します。この場合、Sun MTP が ASCII 文字を EBCDIC に変換するときに、最も高い数値の EBCDIC 文字に変換されます。つまり、ASCII 文字にマッピングするように定義されている最後の EBCDIC 文字に変換されてしまいます。

順方向の複数のマッピングから不要なものを削除するには、1 バイト文字テーブルの場合はテーブルの値を 0x00 に置き換え、2 バイト文字テーブルの場合は 0x0000 に置き換えます。このようにゼロの ASCII 値に変換される EBCDIC 文字は、未定義とみなされます。

# 変換テーブルのカスタマイズ

kixmakecnv ユーティリティーは、変換テーブルファイルを作成します。変換テーブルファイル $UNIKIX/lib/cnvtbl には、小文字から大文字への変換、ASCII から EBCDIC への変換、および EBCDIC から ASCII への変換のための変換テーブルが入っています。提供されているファイルに入っているのは、英語 (US) の変換テーブルです。このファイルは、変換を実行する各領域のプロセスによってロードされます。

英語 (US) 以外の言語をサポートするには、次の手順に従って変換テーブルを変更します。

## ▼ 変換テーブルファイルを変更する

1. **ディレクトリを $UNIKIX/src/convert に変更します。**

```
$ cd $UNIKIX/src/convert
```

2. **エディタを使用して、ソースファイル kxcnvtbl.c に入っている変換テーブルを変更します。**

   3270 データストリームを正しく解釈または作成できるように、ASCII-EBCDIC テーブルと EBCDIC-ASCII テーブルは互いに可逆でなければなりません。つまり、文字を ASCII から EBCDIC に変換し、さらに EBCDIC から ASCII に変換したとき、元どおりの文字が得られる必要があります。

3. **ファイルを保存します。**

4. **新しいバージョンの kixmakecnv をコンパイルしてリンクします。**

```
$ cc -o kixmakecnv kixmakecnv.c kxcnvtbl.c kxsyscnvtbl.c
```

5. **kixmakecnv プログラムを実行します。**

```
$ kixmakecnv
```

   これによって、kxcnvtbl.c と kxsysconvtbl.c で指定された変換テーブルが含まれる新しい $UNIKIX/lib/cnvtbl ファイルが作成されます。

6. **変更を有効にするため、領域を再起動します。**

# Sun MTP 実行可能ファイルの再構築

次の場合、領域に実行可能ファイルを再構築する必要があります。

- ソースモジュールに変更を加えた場合
- PL/I 共有ライブラリのサポートが必要な場合
- C の共有オブジェクトのサポートが必要な場合
- 特定の RDBMS のサポートが必要な場合
- Sun MBM がインストールされている場合
- オプションの Sun 以外の製品 (IBM WebSphere MQ など) をサポートする必要がある場合

## `make` コマンドによる実行可能ファイルの構築

この節では、ソフトウェアに付属のユーザー `makefile` と `make` 機能について説明します。`make` 機能は、領域を再構築する必要がある場合に使用します。

`make` 機能の目的は、カスタマイズ可能なコンポーネントの構築に必要な実行可能ファイル、コマンド、および規則をあらかじめ定義することです。これにより、ソースを変更したのち、変更したソースモジュールを特定するコマンドを 1 つ入力するだけで、適切な実行可能ファイルを作成するための手順が実行されます。ユーザー `makefile` は、`$UNIKIX/src` ディレクトリにあります。

経験豊富なユーザーであれば `makefile` を手動でも変更できますが、`makefile` をカスタマイズして、RDBMS 製品、ユーザールーチン、およびオプションの Sun 以外の製品をサポートする手順は `kixinstall` ユーティリティーによって自動化されています。

`kixinstall` ユーティリティーによって作成される `makefile` は、デフォルトのオプションですべての実行可能ファイル、Sun MTP トランザクションとバッチサーバー、およびすべてのレコード処理ルーチンを構築します。デフォルトでは、これらの実行可能ファイルは、`$UNIKIX/local/bin` ディレクトリに作成されます。

`kixinstall` の詳細については、『Sun Mainframe Transaction Processing ソフトウェア インストールガイド』と『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。

## すべての実行可能ファイルの構築

すべての実行可能ファイル、トランザクションとバッチサーバー、およびすべてのレコード処理ルーチンを構築するには、次のコマンドを入力します。

```
$ cd $UNIKIX/src
$ make clean
$ make all
```

$UNIKIX/src/makefile にある BINDIR 環境変数の値を変更した場合、実行可能ファイルはそのディレクトリに作成されます。それ以外の場合は、$UNIKIX/local/bin に作成されます。

## 選択した実行可能ファイルの構築

選択した実行可能ファイルのみを構築するには、実行可能ファイルの名前を指定して make を実行します。たとえば、unikixtran のみを構築する場合は、次のように入力します。

```
$ cd $UNIKIX/src
$ make unikixtran
```

$UNIKIX/src/makefile にある $BINDIR の値を変更した場合、新しい unikixtran はそのディレクトリに作成されます。それ以外の場合は、$UNIKIX/local/bin に作成されます。

## ユーザー作成の C 関数のバインド

アプリケーションプログラムからユーザー作成の C 関数を呼び出すには、それらをトランザクションサーバーまたはバッチサーバーにバインドする必要があります。この機能を実現するため、新しいトランザクションサーバーとバッチサーバーを作成する unikixtran.o と unikixvsam.o のファイルが、$UNIKIX/lib ディレクトリにあります。

**Server Express**: トランザクションとバッチサーバーを Server Express 実行時システムにバインドすると、ユーザーのアプリケーションプログラムが .int、.gnt または .o 形式で作成されます。.int および .gnt 形式は、実行にほかの COBOL サブルーチンとバインドする必要はありませんが、.o 形式はバインドが必要です。

**ACUCOBOL-GT**: トランザクションとバッチサーバーを ACUCOBOL-GT® 実行時システムにバインドすると、ユーザーのアプリケーションプログラムが .acu 形式で作成されます。この形式は、実行にほかの COBOL サブルーチンや言語実行時ルーチンとバインドする必要はありません。

# kixinstall による実行可能ファイルの構築または再構築

kixinstall を使用して実行可能ファイルを構築する方法の詳細については、『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。

## ▼ ユーザー作成モジュールを統合する

1. $UNIKIX/src ディレクトリに移動します。
2. kixinstall コマンドを実行し、構成ユーティリティーメニューを表示します。

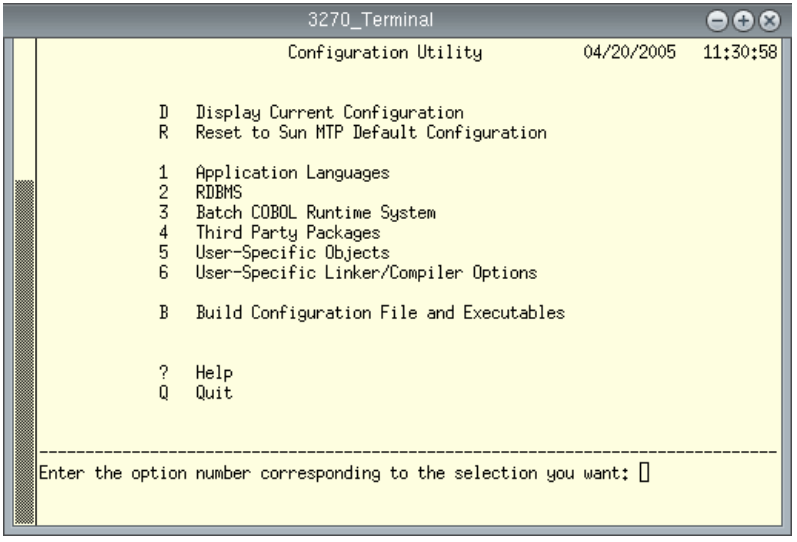


図 10-1 Configuration Utility メインメニュー

3. メインメニューで R を選択し、Return キーを押して現在の設定を削除します。
これでデフォルトの設定になります。

---

注 – 現在の設定を表示するには、随時、オプション D を選択します。

---

4. **ユーザー固有のオブジェクトを追加または変更するには、構成ユーティリティーメインメニューで、`5` を選択して Return を押します。**

   `vi` エディタでファイルが開きます。詳細は、図 10-2 を参照してください。

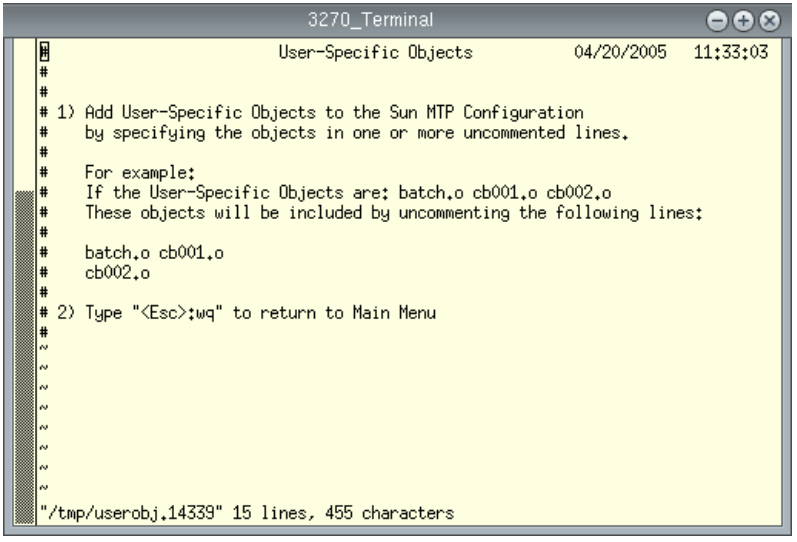


図 10-2　User-Specific Objects 画面

5. **統合するオブジェクトの名前、すなわち `$UNIKIX/src` に対する相対パスを入力します。**

6. **入力し終わったら、Esc キーを押して `:wq` と入力します。**

   メインメニューが表示されます。

7. **ユーザー固有のリンカーとコンパイラオプションを追加するには、メインメニューで `6` を選択し、Return を押します。**

   `vi` エディタでファイルが開きます。

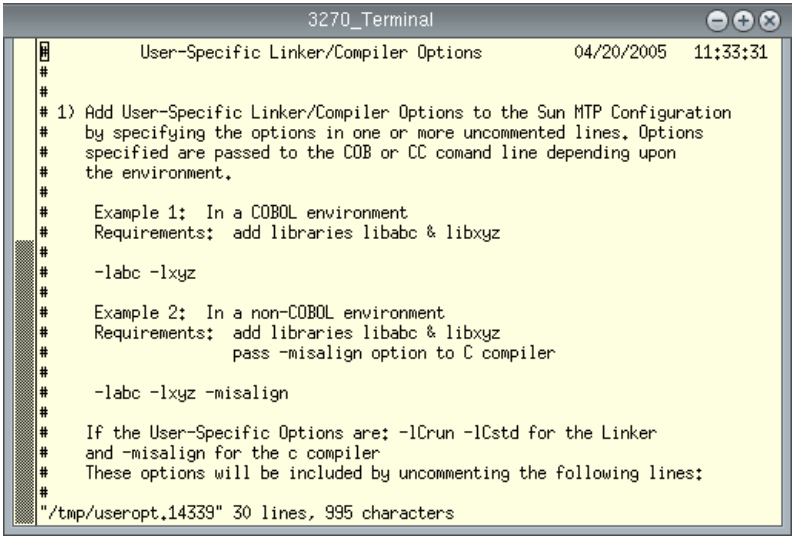


図 10-3　User-Specific Linker and Compiler Options の指定

8. **手順 5 で指定したユーザー固有のオブジェクトに適用するリンカーとコンパイラオプションを入力します。**

   リンカーとコンパイラのオプションの詳細は、C コンパイラのマニュアルまたは COBOL のマニュアルで、`cob` コマンドを参照してください。

9. **セッションを終了するには、Esc キーを押して `:wq` と入力します。メインメニューが表示されます。**

10. **構成ファイルを構築するために、メインメニューで `B` を選択して Return を押します。**

    構成ファイルの構築画面が表示されます。これにより、先の選択に基づいて構成ファイルが構築されます。

11. **構築し終えてから Return を押すとメインメニューが表示されます。**

12. **`Q` を入力して構成ユーティリティーを終了します。**

    終了後、次のメッセージが表示されます。

```
#Sun MTP Configuration successfully completed.
#
#Logfile was /tmp/kixinstall.log.#####
```

これは、この構成セッション中に作成されたログファイルのパス名です。このファイルはテキストエディタで表示できます。問題が発生したときは、このファイルのコピーを、ご購入先にお送りください。

13. kixinstall でシステムを再構築する際に実行しなければならない追加手順については、『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。

### ▼ 新しいオンライントランザクションサーバーを構築する

makefile は、新しいトランザクションサーバーを $UNIKIX/local/bin ディレクトリ (デフォルトの場合) に作成します。

1. 次の 2 つのディレクトリが存在しない場合は、次のようにそれらを作成します。

```
$ mkdir $UNIKIX/local
$ mkdir $UNIKIX/local/bin
```

2. 次のコマンドを使用して、新しいバージョンのトランザクションサーバー (unikixtran) を作成します。

```
$ cd $UNIKIX/src
$ make unikixtran
```

3. この領域を起動します。

makefile の BINDIR 環境変数を変更し、新しいバージョンを $UNIKIX/bin または $UNIKIX/local/bin 以外のディレクトリに作成した場合は、-l オプションを付け、ディレクトリをその引数に指定して領域を起動します。次に例を示します。

```
$ unikixmain -l /usr/workdir ....
```

unikixmain のコマンド行オプションの詳細については、『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。

## 新しいバッチ COBOL 実行時システムの構築

この節では、次の新しいバッチ COBOL 実行時システムの作成手順を説明します。

- 標準のバッチ COBOL 実行時システム
- Sun MBM COBOL 実行時システム

## ▼ 標準のバッチ COBOL 実行時システムを構築する

makefile は、実行時システムを $UNIKIX/local/bin ディレクトリ (デフォルトの場合) に作成します。

1. 次の 2 つのディレクトリが存在しない場合は、次のようにそれらを作成します。

```
$ mkdir $UNIKIX/local
$ mkdir $UNIKIX/local/bin
```

2. $UNIKIX/src ディレクトリまたは kixinstall セッションで指定したディレクトリに変更します。

3. 新しいバージョンの実行時システム、unikixvsam を作成します。

```
$ make unikixvsam
```

4. 実行時システムを $UNIKIX/local/bin に作成し、CBCH を使用して COBOL プログラムを実行する場合は、通常どおり領域を起動できます。

   makefile の BINDIR 環境変数を変更した場合、または kixinstall で実行可能ファイルに別のターゲットディレクトリを指定した場合は、-l オプションを指定して領域を起動する必要があります。unikixmain のコマンド行オプションの詳細については、『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。

## ▼ Sun MBM COBOL 実行時システムを構築する

makefile は、実行時システムを $UNIKIX/local/bin ディレクトリ (デフォルトの場合) に作成します。

1. 次の 2 つのディレクトリが存在しない場合は、次のようにそれらを作成します。

```
$ mkdir $UNIKIX/local
$ mkdir $UNIKIX/local/bin
```

   makefile の $BINDIR を変更した場合、または kixinstall セッションで実行可能ファイルに別のターゲットディレクトリを選択した場合は、makefile によってそのディレクトリの中に新しいバージョンが作成されます。

2. $UNIKIX/src ディレクトリまたは kixinstall セッションで指定したディレクトリに変更します。

3. 新しいバージョンの実行時システム、rtsvsam を作成します。

```
$ make rtsvsam
```

次に、出力の例を示します。

```
rm -f /tmp/company1/bin/rtsvsam_*
rm -f /tmp/company1/bin/RUNB
(sh RTSVSAM/inst_rtsvsam "SOL" "y" "y" \
.
.
.
+ cob -xD -o /tmp/company1/bin/rtsvsam_yy ./RTSVSAM/CPR.o ./RTSVSAM/COB.o
   ./RTSVSAM/CCFinit.o ./RTSVSAM/CCFextfh.o cobwrap.o coberror.o
   ./RTSVSAM/kxpassrem.o ./RTSVSAM/kxpassubs.o ./RTSVSAM/ebmlsfile.o
   ./RTSVSAM/runb.o /tmp/company1/kxusrexit.o /tmp/company1/cvlwrupr.o
   /tmp/company1/kxusrxlt.o /tmp/company1/kxcvtxlt.o /tmp/company1/kxcnvtbl.o
   /net/haven/home/ebm81/KIX/lib/libbsec.a -lcurses -lsocket -lnsl -lm
+ set +x
#
# Sun MBM COBOL Runtime System /tmp/company1/bin/RUNB Installed.
# To be effective set RUNBDIR in your setup file:
# setenv RUNBDIR /tmp/company1/bin
#
# Add the RUNBDIR pathname to the RUNBPATH in your setup file.
#
```

4. Sun MBM サブシステムユーザー設定ファイル内の RUNBDIR と RUNBPATH の 2 つの環境変数を、COBOL 実行時システムを作成したディレクトリに設定します。

5. unikixjob コマンドを使用してジョブをサブミットします。

   unikixjob コマンドの詳細については、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

# 第11章

# Sun MTP 領域の監視

この章では、Sun Mainframe Administration Tool (Sun MAT) の概要を説明します。このツールは、Sun MTP 領域の監視に使用できます。次のトピックが含まれます。



このソフトウェアのインストール、構成、および使用法の詳細については、『Sun Mainframe Administration Tool ユーザーズガイド』を参照してください。

## Sun MAT の概要

Sun MAT ソフトウェアは、ネットワークを介して分散している 1 つまたは複数のシステム上で実行中の、1 つまたは複数の Sun MTP 領域に対する遠隔管理アクセスを可能にするように設計されています。このツールは、次の 3 つのコンポーネントで構成されています。

- 領域の管理に使用されるグラフィカルユーザーインタフェース (GUI) である Sun MAT
- GUI と管理される領域とのインタフェースとして機能する Sun Mainframe Administration Agent (Sun MAA)
- Sun MTP の制御下で実行される `unikixadmin` プロセス

1 つまたは複数の Sun MTP 領域にアクセスできるようにするには、これらの領域をホストするそれぞれのシステムが、実行中の Sun MAA のインスタンスもホストする必要があります。Sun MAT は、ネットワーク上の任意の場所で実行できます。

1 つのシステムに管理対象の領域が 1 つあるシンプルな環境では、3 つのコンポーネント (Sun MAT、Sun MAA、および Sun MTP) は、図 11-1 に示すように、すべて同じ場所に配置されます。

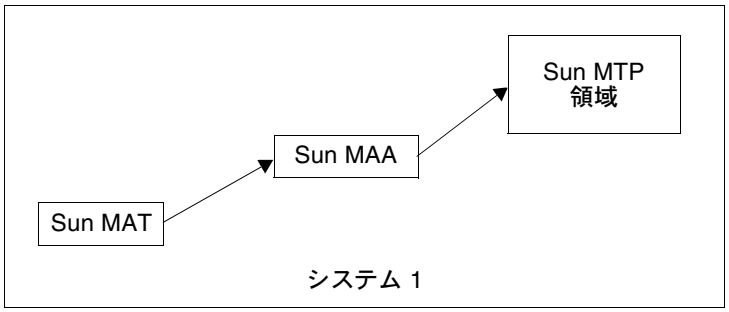


**図 11-1**　1 つの領域と 1 つのシステムで構成される環境

次の図は、より複雑な環境の例です。複数のシステムにそれぞれ複数の実行中の領域があり、これらすべてが 1 つのポイントから監視されます。

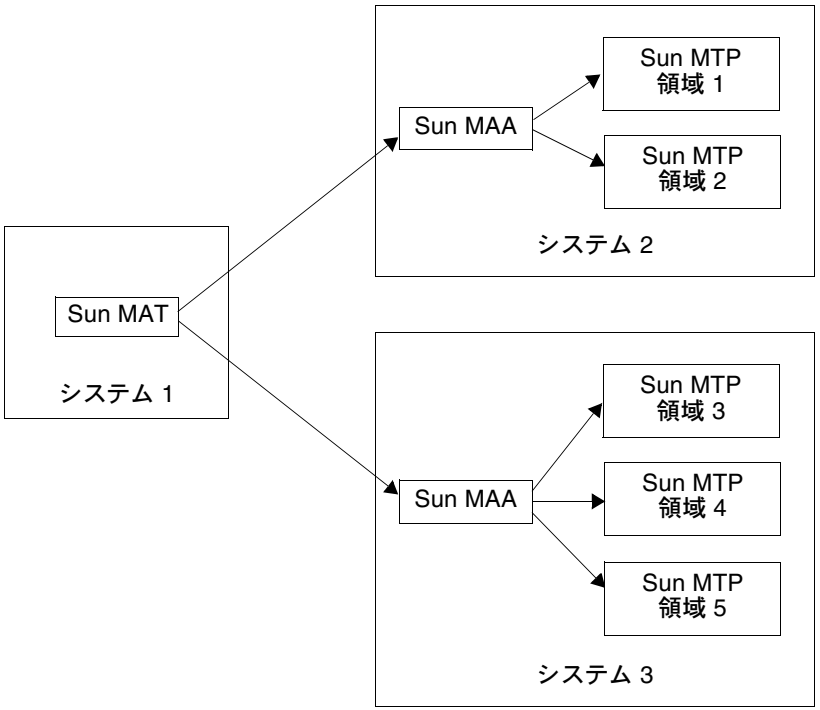


**図 11-2**　複数の領域と複数のシステムで構成される環境

複数の管理者が Sun MAT を同時に実行して、同じ領域にアクセスすることも可能です。図 11-3 に、この例を示します。

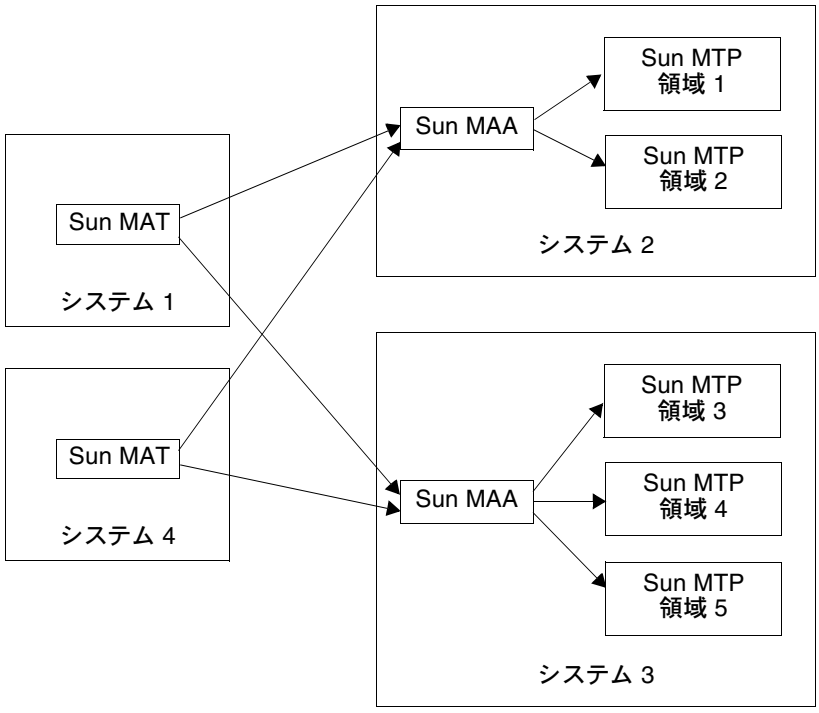


**図 11-3**　複数の Sun MAT から複数の領域へのアクセス

# 前提条件

Sun MAA を介して遠隔管理する Sun MTP の領域は、リリース 8.1.0 以降にする必要があります。

Sun MAA は、Java 2 Standard Edition (J2SE) バージョン 1.4 以降のリリースを必要とする Java アプリケーションです。Sun MAA を起動する前に、Sun MAA を実行する予定のシステムに J2SE 1.4 以降をインストールしておく必要があります。

また、Sun MAA を起動するユーザーの `$PATH` に、J2SE のバージョンを定義しておく必要もあります。J2SE のバージョンが正しく定義されているかどうかをテストするには、次のコマンドを入力します。出力は、次のようになります。

```
$ which java
/usr/java1.4.0/bin/java
$ java -version
java version "1.4.0"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.0-b92)
Java HotSpot(TM) Client VM (build 1.4.0-b92, mixed mode)
```

J2SE がインストールされる場所は、使用しているシステムによって異なります。J2SE がインストールされているかどうか、およびインストールされている場所を確認するには、システム管理者にお問い合わせください。サポートされているバージョンの J2SE がシステムにインストールされていない場合、次のサイトからダウンロードできます。

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

Sun MAT も Java アプリケーションであり、J2SE ランタイム (バージョン 1.4 以降) が使用可能なシステムで実行できます。

# Sun MTP の領域へのアクセス

領域を Sun MAT で監視するには、unikixadmin サーバープロセスを有効にして領域を起動する必要があります。

## ▼ unikixadmin サーバーを有効化する

1. **Solaris プラットフォームの LD_LIBRARY_PATH などの、共有ライブラリの環境変数が Java 仮想マシン (JVM) の位置を指していることを確認します。**

2. **unikixmain に -X オプションを使用して領域を起動します。**

   -X オプションでは、unikixadmin で使用するポートの番号を指定する必要があります。このとき、一意のポート番号を指定する必要があります。次に例を示します。

```
$ unikixmain -X 9876 other options
```

   unikixadmin サーバーを領域初期化の一部として起動すると、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。

```
02/16/2005 13:08:17 unikixmain :KIX0173I Process 1234 of type a started
```

   サーバーが起動したことが、次のようなメッセージで通知されます。

```
02/16/2005 13:08:19 unikixadmin :KIX0301I Entering (VER. 8.1.0 - 01/31/2005
```

# 付録 A

# シェルスクリプトの変更

この付録では、アプリケーション環境に合わせて Sun MTP シェルスクリプトを変更する方法について説明します。次のトピックについて説明します。



## シェルスクリプトの変更

Sun MTP は、`$UNIKIX/bin` ディレクトリにあるシェルスクリプトを使用して、ユーティリティープログラムを起動します。ご使用のアプリケーション環境に合わせて、これらのスクリプトを変更できます。通常、カスタマイズが必要なシェルスクリプトは、`kixprint`、`kixjournal`、および `kixjob` です。ほかのスクリプトの変更が必要になるのは、サイト固有の運用上の要請による場合だけです。

---

**注 –** 変更したシェルスクリプトは、変更した方のアプリケーションソースコードとなり、保守についてはその方の責任となります。

---

### ▼ シェルスクリプトを変更する

1. **変更するスクリプトをコピーして、`$UNIKIX/local/bin` ディレクトリに保存します。**
2. **テキストエディタを使用して、スクリプトを変更して保存します。**
3. **`$PATH` で `$UNIKIX/local/bin` が `$UNIKIX/bin` の前にあることを確認します。**
4. **この領域を起動します。**

unikixmain のコマンド行オプションの詳細については、『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。

kixprint シェルスクリプトの変更については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

# デフォルトエディタの変更

Sun MTP では、vi をデフォルトのエディタとして使用します。デフォルトは、kixed シェルスクリプトを変更して変えることができます。

## ▼ デフォルトエディタを変更する

1. **kixed スクリプトをコピーして、$UNIKIX/local/bin ディレクトリに保存します。**

2. **スクリプトを変更して、vi を実行している文を、環境変数を評価する if 文に置き換えます。**

   この例で、環境変数は $EDITOR です。if 文は、$EDITOR が設定されているかどうかをチェックします。設定されている場合は、EDITOR 環境変数に指定されているエディタをシェルスクリプトが呼び出します。$EDITOR が設定されていない場合は、vi が使用されます。

```
if test -n "$EDITOR"
then
       $EDITOR $FILENAME <$2 >$2
else
       vi $FILENAME <$2 >$2
fi
```

3. **スクリプトを保存します。**

   次回 kixstart -l を使用して領域を起動すると、指定したエディタがデフォルトになります。

付録 B

# 日付/時刻の構成

この付録では、kixdate を使用して日付と時刻を変更する方法について説明します。次のトピックについて説明します。



## 日付/時刻の構成ユーティリティーの起動

kixdate 構成ユーティリティーは、Sun MTP と Sun MBM システムの日付と時刻を変更します。デフォルトで、Sun MTP と Sun MBM は、システム時刻を反映します。Sun MTP は、ログ、画面、トレース、ダンプ、およびレポートの日付をすべて 4 桁年 (*YYYY*) の形式で表示します。

Sun MTP に日付/時刻構成ユーティリティーを使用するには、$KIXSYS を設定する必要があります。$KIXSYS 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。

Sun MBM の日付/時刻構成ユーティリティーを使用するには、Sun MBM の環境を設定する必要があります。詳細については、『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』を参照してください。

## ▼ 日付/時刻の構成ユーティリティーを起動する

- 次のコマンドを入力し、図 B-1 に示す画面を表示します。

```
$ kixdate
```

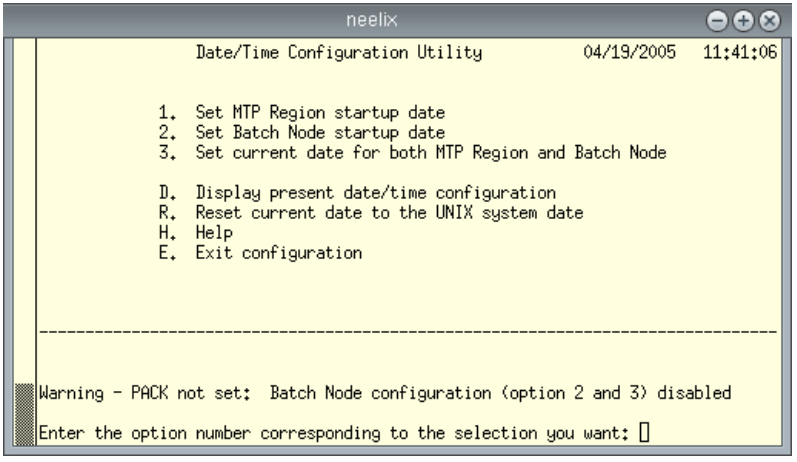


図 B-1　Date/Time Configuration Utility (`kixdate`)

メニューオプションについては、次の節で説明します。

# Sun MTP 起動日の設定

この節では、Sun MTP 起動日の設定の形式と手順を説明します。入力した日付/時刻が有効な場合は、領域起動のたびに、それが絶対起動日として扱われます。

## ▼ 起動日を設定する

1. 「Date/Time Configuration Utility」メニューで、オプション `1` を選択します。
2. 起動日の構成画面が表示されたら、*MM/DD/YYYY-HH:MM:SS* という形式で日付を入力します。

   設定できる年は、1970 ～ 2030 年です。

```
MTP Region Startup Date Configuration
-------------------------------------------------


Enter MTP Region Startup Date in the format MM/DD/YYYY-HH:MM:SS

04/19/2005-11:53:20
Successfully set MTP Region startup date to 04/19/2005-11:53:20
You will need to restart MTP Region for the new date setting to take effect
Press <Enter> to continue
```

**図 B-2**　　起動日の設定

3. **Return キーを押して日付を確定し、それを表示します。**
4. **領域を再起動して、新しい日付を有効にします。**

   **例**: 3/31/2002-23:30:00 と指定すると、起動日は、2002 年 3 月 31 日 11:30:00 PM となります。

   8 時間後、時刻は 2002 年 4 月 1 日 07:30:00 AM となります。

   領域をいったん停止し、数日後に再起動すると、時刻が再度 2002 年 3 月 31 日 11:30:00 PM に設定され、その時点から時刻の増分が始まります。

---

**注 –** 時計が起動日に時刻を刻み始めるのは、ツールがそれを受け付けた時点からであって、システムの起動時からではありません。

---

# Sun MTP と Sun MBM の現在の日付の設定

この節では、Sun MTP と Sun MBM の現在の日付を設定する形式と手順を説明します。入力した日付と時刻が有効な場合は、それが両製品の現在の日付となります。「現在の」とは、現在の日付が設定されていると構成ツールが確認したとき、この値から日付が増分することを意味します。この値は、システムが停止されても増分し続けます。

## ▼ 現在の日付を設定する

1. 「Date/Time Configuration Utility」メニューで、オプション 3 を選択します。
2. 現在の日付の構成画面が表示されたら、*MM/DD/YYYY-HH:MM:SS* という形式で日付を入力します。

   設定できる年は、1970 ～ 2030 年です。
3. Return キーを押して日付を確定し、それを表示します。
4. 領域と Sun MBM を再起動して、新しい日付を有効にします。

   現在の日付オプションは、Sun MTP と Sun MBM の日付と時刻が同期することを保証します。

---

**注 –** オプション「1」または「2」を使用して日付を設定した場合、オプション「3」は使用できません。逆にオプション「3」を使用した場合、オプション「1」または「2」は使用できません。

---

# 現時点の日付/時刻の表示

## ▼ 日付と時刻を表示する

- 「Date/Time Configuration Utility」メニュー画面で、「Display (D)」オプションを選択します。

  これで、Sun MTP と Sun MBM の両方のシステムの日付/時刻構成と起動時刻が表示されます。

```
------------------------------------------------------------------------

Warning - PACK not set; Batch Node configuration (option 2 and 3) disabled

Enter the option number corresponding to the selection you want:d

PACK environment variable not set.Unable to display Batch Node date
Sun MTP startup date is set to 04/19/2005-11:53:20
 as an absolute date
Press <Enter> to continue
```

**図 B-3**　現在の日付/時刻の構成の表示画面

オプション「3」を使用して日付を設定した場合、そこで指定された日付に設定日からの経過時間を加えたシステムの現在の日付が表示されます。表示には、日付/時刻が相対的であることが示されます。

オプション「1」または「2」を使用して日付を設定した場合、起動日が表示され、それが絶対日付であることが示されます。

Sun MTP または Sun MBM に対して日付/時刻を構成しなかった場合、そのことが表示されます。

# 現在の日付のシステム日付へのリセット

## ▼ 日付をリセットする

- **「Date/Time Configuration Utility」メニューで、「R」オプションを選択します。**

  これで、Sun MTP と Sun MBM の両方のシステムの日付/時刻構成がシステム日付にリセットされます。日付/時刻構成が存在しない場合は、そのことが表示されます。

  絶対日付構成から相対日付構成またはその逆に変更する場合、まず「R」オプションを使用して Sun MTP と Sun MBM の日付をシステム日付にリセットし、そのあと、日付をリセットします。

# ヘルプ画面の表示

## ▼ ヘルプを表示する

- 「Date/Time Configuration Utility」メニューで、「Help (H)」オプションを選択します。

  これで、ヘルプファイルが表示されます。このヘルプファイルでは、日付/時刻構成ユーティリティーオプションについて説明されています。

  ヘルプ画面を操作するには、次のキーを使用します。

| | |
|---|---|
| Return | 1 行下に移動します。 |
| スペースバー | 1 画面下に移動します。 |
| q | Enter (キーパッドの Enter キーではなく、Return キー) を押して、「Date/Time Configuration Utility」メニューに戻るように促すプロンプトが表示されます。 |

# 日付/時刻の構成ユーティリティーの終了

## ▼ ユーティリティーを終了する

- 「Date/Time Configuration Utility」メニューで、「E」オプションを選択します。

  これで、コマンドプロンプトに戻ります。

# 用語集

## 数字

**1 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 1 バイト文字セット (SBCS) と呼ばれます。たとえば、英語、スペイン語、およびフランス語が 1 バイト文字セット (SBCS) です。

**2 バイト文字セット (DBCS)**
(名詞) 一部の言語スクリプトでは、文字を表すのに 2 バイトが必要です。そのスクリプトは 2 バイト文字セット (DBCS) と呼ばれます。たとえば、日本語、中国語、および韓国語が 2 バイト文字セット (DBCS) です。

**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) タスクを異常終了するのに使用される `EXEC CICS` コマンド。

**Accounting Header Record (AHR)**
(名詞) 処理されるデータのタイプと、レコードの作成状況を記述した C 構造体。

# 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です。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ます。

# 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です。

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ます。

# 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ます。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ます。

# G

GUI (名詞) グラフィカルユーザーインタフェース。

# 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。

# 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 を中心に開発されました。

# S

**Screen Generation Utility (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ます。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します。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ます。

**SOSI フィールド** (名詞) SOSI (Shift-Out、Shift-In) フィールドは、1 バイト文字と 2 バイト文字の両方を含んだ 3270 フィールドです。SOSI フィールドのすべての 2 バイト文字は、SOSI 文字で挟まれている必要があります。

**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です。

**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します。

Sun Mainframe Transaction Processing ソフトウェア (Sun MTP)
: (名詞) プロセス間通信サービス、ソケット、COBOL、C、PL/I などの機能を使用してアプリケーションを実行するユーザーアプリケーション。クライアント以外の Sun MTP のすべてのコンポーネントは、メインサーバープロセスである `unikixmain` によって起動します。

Sun MTP 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ます。ディレクトリは GCT にリストされます。

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) は信頼性のある全二重のデータストリームを提供します。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です。TCP プロトコルは、ユーザープロセスではなく、IP と連携します。

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ます。

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します。TN3270E もサポートします。

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します。TN3270E も含まれます。

# U

`unikixdcl` サーバー
: (名詞) DCL プロトコルスタックを使用する SNA サーバーへの遠隔接続の数と状態を監視する 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) WebSphere MQ 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おさ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します。

**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です。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ます。

**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。Sun MTP の VSAM 環境では、データセットはファイルの論理名です。このファイル名は物理ファイルです。たとえば、ACCTFILE はデータセット名です。これが KSDS データセットである場合、このデータセットは ACCTFILE.dta と ACCTFILE.idx という 2 つの物理ファイルで構成されています。

**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します。

**アクセス制御エントリ (ACE)** (名詞) アクセス制御エントリ。「ACL」を参照。

**アクセス制御リスト (ACL)**
(名詞) アクセス制御リスト (ACL) は、1 ユーザーまたはグループのオブジェクトへのアクセス権または監査権を指定するアクセス制御エントリ (ACE) で構成されます。

オブジェクトの所有者が任意の ACL をコントロールします。

**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します。

**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ます。

**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います。

## 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

**外部表示インタフェース (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 と通信します。

**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) トランザクションが有効な間、ユーザーとの会話 (通常、SEND/RECEIVE のシーケンス) が行われ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、アクセスされます。

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ます。

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ます。

**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。

**環境変数** (名詞) プログラムファイルおよびアプリケーションの位置を定義する変数。クライアントとサーバーは、どちらも環境変数を使用します。

# き

**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編成 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。リソースがほかの 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します。

**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します。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します。

# く

**クラスタ** 「VSAM クラスタ」を参照。

**グループ** (名詞) Sun MTP での、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ます。ディレクトリは GCT に定義されます。UNIX システムでは、「グループ」という用語は、共通の要件を備えたユーザーの集合を指します。

**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します。

# こ

**コードページ** (名詞) 16 進数の値とグリフとのマッピングテーブル。たとえば、ASCII のコードページ ISO8859-1 は 0x41 を使用して文字「A」を表し、EBCDIC のコードページ IBM-1047 は 0xC1 を使用して文字「A」を表します。

**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ます。

# 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ます。キーはレコードを指します。

# 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。

**システムトランザクション** (名詞) システム制御と、Sun MTP 開発システムへのアクセスに使用される Sun MTP トランザクションのセット。システムトランザクションは、予約文字「C」で開始されます。

**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、ほか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ます。

**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。

# す

**スーパーユーザー** (名詞) システムのすべての部分に対する無制限のアクセス権を持つ、UNIX システムの単一のユーザー。ユーザー名は root です。

**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。

**スパンレコード** (名詞) VSAM ブロックサイズから 24 バイトを引いた値よりも大きい KSDS VSAM ファイル内のレコード。スパンレコードは、ブロックのデータ部分の最初から始まります。そのスパンレコードの各部分は、レコード全体の格納に必要な数のブロックのデータ部分全部を使用します。そのレコードの最後の部分を含むブロックの残りの領域は未使用のままになります。

# 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ます。

**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

**代替索引** (名詞) ファイル属性、物理ファイルの位置、および基本クラスタ名を持つ、VSAM カタログ内のエントリ。代替索引を使用すると、主キーとは異なるキーによってレコード検索が可能です。

**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します。デバッグに使用されます。

# 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ります。

**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ます。

**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ほかのコードセットに (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になります。

**伝送制御プロトコル (TCP)** 「TCP/IP」を参照。

# と

**同期データリンク制御 (SDLC)** (名詞) 米国規格協会 (ANSI) の拡張データ通信制御手順 (ADCCP) および国際標準化機構 (ISO) のハイレベルデータリンク制御手順 (HDLC) のサブセットに準拠する規定。リンク接続上で、同期をとり透過的なコードで情報をビットごとにシリアル転送する管理に使用します。

**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です。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、ほか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ります。プログラムが異常終了した場合、回復と再起動の機能は最新の同期点以前の更新をバックアウトしません。

**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います。

**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します。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ります。一つの `unikixtrin` プロセスは、遠隔にある 3270 デバイスクライアントすべてをサポートします。

**トランザクションクラス** (名詞) 共通の優先順位を持つトランザクションの物理グループ。

**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ほかの Sun MTP または CICS 領域でトランザクションの実行を可能にします。

## 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ます。

## 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ません。

## 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ます。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します。

**ファイルシステム** (名詞) 物理ディスクドライブをパーティションと呼ぶ小単位の領域に分割する機能。パーティションには、ファイルシステム、スワップ空間、ブートセクタその他の情報を含めることができます。

**ファイルのアクセス権 (またはモード)** (名詞) オペレーティングシステムの定義に従って、ファイルへのアクセスを制御します。

**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ます。異常終了と同じ意味です。

**物理書き込みヘッダーレコード (PWHR)** (名詞) ジャーナルファイルに書き込む物理ブロックを記述する C の構造体。

**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です。

**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ほか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。

## へ

**変更前イメージ** (名詞) 回復ファイルに書き込まれるレコードのコピー。

## 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ます。たとえば、grep コマンドについて表示するときは、プロンプトで man grep と入力します。

## も

**モード** (名詞) ファイルのアクセス権。

**監視管理テーブル (MCT)** (名詞) 領域で有効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ます。ルートファイルシステムは、完全なファイル名の最初の文字としてスラッシュ ( / ) を付けて、参照されます。

## 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ます。

**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

## D



## E



## F

## I



## J



## K



## L

## M



## N



## O



## P



## Q



## R



## S

## T



## U

## V

## き

## す



## せ

## そ



## た



## つ



## て

## と

## に



## は



## ひ



## ふ