Sun™ MTP Client ユーザーズガイド

Release 7.2

Sun Microsystems, Inc.
www.sun.com

Part No. 819-2524-10
2005 年 6 月, Revision A

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

原典: *Sun™ MTP Client User's Guide*
Part No: 816-2797-12
Revision A

目次

図目次

表目次

はじめに

このマニュアルでは、Sun™ MTP Client ソフトウェアのインストール方法および使用法について説明します。

マニュアルの構成

第 1 章では、Sun MTP Client のクライアント機能、サポート対象のプロトコル、および動作要件について説明します。

第 2 章では、ソフトウェアのインストール方法について説明します。

第 3 章では、ソフトウェアの構成方法について説明します。

第 4 章では、ホストで Sun MTP Client および Sun MTP を起動する方法について説明します。

第 5 章では、Sun MTP Client の管理方法について説明します。

第 6 章では、3270 端末の構成方法、起動方法、および停止方法について説明します。

第 7 章では、3270 プリンタアプリケーションの使用法について説明します。

第 8 章では、C プログラミング言語で使用する場合の Sun MTP Client 外部呼び出しインタフェース (ECI) について説明します。

第 9 章では、EPI の C プログラム関数仕様、および必要なデータ構造体について説明します。

付録 A では、`KIXTERM.INI` 初期設定ファイルについて説明します。

付録 B では、Sun MTP Client エラーメッセージのリストを示します。

UNIX コマンド

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

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

  `http://docs.sun.com`

シェルプロンプトについて

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

書体と記号について

書体または記号*	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例。	.login ファイルを編集します。 ls -a を実行します。 % You have mail.
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して表します。	% **su** Password:
AaBbCc123	コマンド行の可変部分。実際の名前や値と置き換えてください。	rm *filename* と入力します。

書体または記号*	意味	例
『 』	参照する書名を示します。	『Solaris ユーザーマニュアル』
「 」	参照する章、節、または、強調する語を示します。	第 6 章「データの管理」を参照。 この操作ができるのは「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	% **grep '^#define ** **XV_VERSION_STRING'**

* 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。

関連マニュアル

製品	タイトル	Part No.
Sun Mainframe Transaction Processing ソフトウェア	『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』	819-2514-10
	『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』	819-2515-10
	『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』	819-2519-10
3270 Data Stream	『IBM 3270 Information Display System Data Stream Programmers Reference』	CA23-0069

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 MTP Client ユーザーズガイド』、Part No. 819-2524-10

第1章

概要

この章では、Sun MTP Client のクライアント機能、サポート対象のプロトコル、および動作要件について説明します。

Sun MTP Client の機能

Sun MTP Client は、次のクライアント機能を提供します。

- 3270 端末 (Windows の場合のみ使用可能)
- 3270 プリンタ (Windows の場合のみ使用可能)
- 外部表示インタフェース (EPI) API
- 外部呼び出しインタフェース (ECI) API
- ダイナミックリンクライブラリ (DLL) 形式の Pascal バインディング

これらの機能によって、端末やプリンタ、PC や UNIX マシン上で実行している EPI アプリケーションまたは ECI アプリケーションを、1 つまたは複数の UNIX マシン上で実行している Sun MTP 領域に直接接続できます。さらに、Sun MTP Client は、独自のアプリケーションを作成する場合に参考となるサンプルを提供します。

図 1-1 の各 Sun MTP Client は、図 1-2 に示されているとおり複数の ECI アプリケーション、EPI アプリケーション、および 3270 端末またはプリンタをサポートできます。

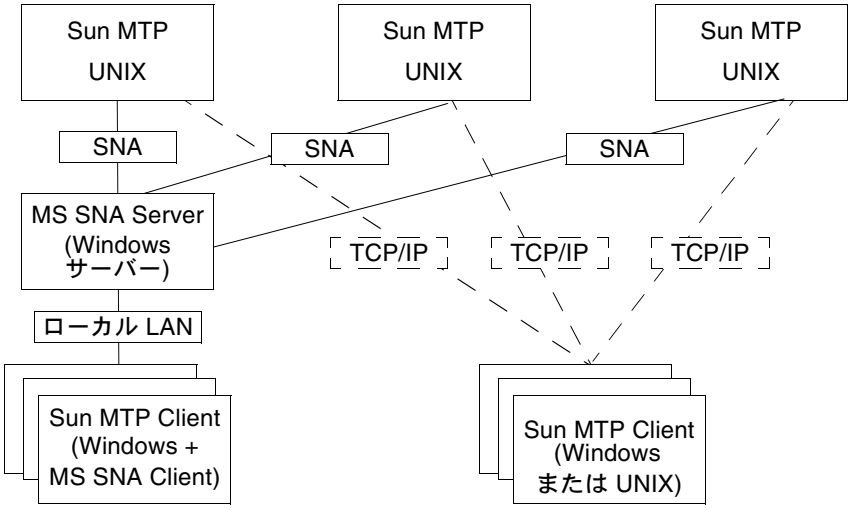

図 1-1　　Sun MTP Client ネットワーク接続方法

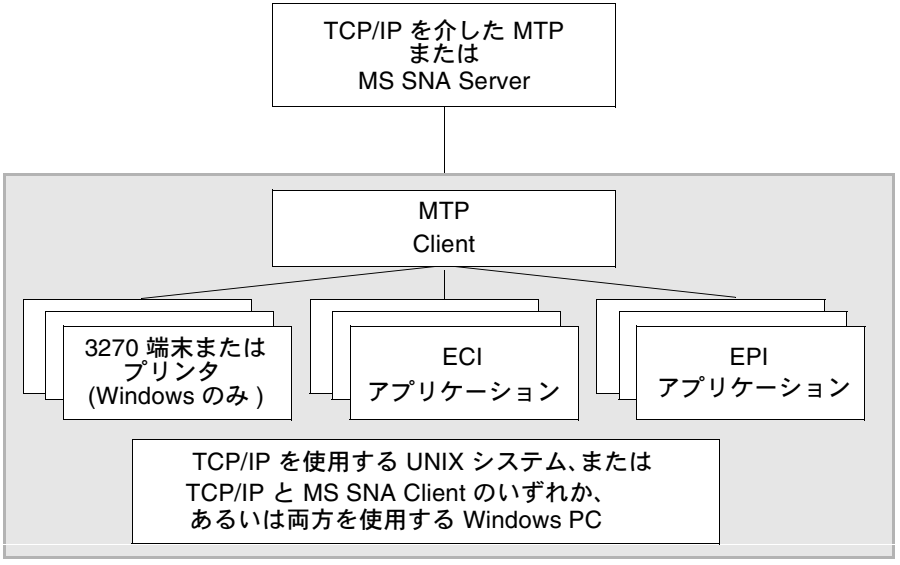

図 1-2　　Sun MTP Client アプリケーションサポート

サポートするトランスポートプロトコル

この節では、Sun MTP Client が、TCP/IP または SNA を介して Sun MTP へ接続する方法について説明します。

TCP/IP

Sun MTP Client は、TCP/IP プロトコルをサポートします。UNIX および Windows で利用できます。

Windows では、WINSOCK に準拠したソケットのダイナミックリンクライブラリ (DLL) で実装されます。第 2 章では、複数のバージョンの DLL が PC に実装されている場合に、Sun MTP Client が使用する DLL の決定方法を説明します。

UNIX では、TCP/IP はオペレーティングシステムの一部です。

SNA

Windows 用の Sun MTP Client は、Microsoft SNA Server または IBM Communications Server のどちらかを使用して、Sun MTP に接続します。このためには、アクセスする各領域に対して LU 6.2 接続を構成した、Windows サーバーまたは SNA サーバーを実行するマシンが必要です。Windows 用の Sun MTP Client を用いて、複数の Windows マシンが、SNA サーバーを介して Sun MTP に接続できます。図 5-3 を参照してください。また、Sun MTP Client を実行している各マシンには、SNA Client ソフトウェアがインストールされている必要があります。

各 SNA Client および SNA サーバー間の接続は、いくつかの方法で実現できます。詳細は、SNA サーバーのマニュアルを参照してください。

SNA は、現在、UNIX 用の Sun MTP Client ではサポートされていません。

動作要件

Sun MTP Client が利用できるプラットフォームは複数あります。最小動作要件は、次のとおりです。

- Microsoft Windows
  - 3 ページの「サポートするトランスポートプロトコル」で説明している `WINSOCK.DLL` または Microsoft SNA Client for Windows。
  - プログラム開発環境。C および C++ がサポートされています。
- Solaris
  - プログラム開発環境。C および C++ がサポートされています。

用語

このマニュアルでは、次の用語を使用します。

- 「Windows」は、Microsoft Windows 製品を表します。
- 「UNIX」は、Solaris または AIX オペレーティングシステムを表します。

このマニュアルでは、ファイルの指定に次の規則を使用します。

- 場合により、すべてのファイルが DOS 名バージョンで指定されます。つまり、パス指定に `\` 文字をディレクトリ区切り文字として使用し、`C:` などのドライブ名を含む場合もあります。
- 場合により、`$INSTROOT` という語を使用して Sun MTP Client がインストールされたディレクトリを指定します。

第2章

インストール

この章では、Sun MTP Client のインストール手順について説明します。

Sun MTP Client の Windows へのインストール

▼ Sun MTP Client を Windows にインストールする

1. **.exe ファイルをダブルクリックして、インストールを開始します。**
   インストールの手順を指示するインストール画面が表示されます。
2. **インストール先のフォルダを選択します。**
   デフォルト値とは別のフォルダにインストールする場合、「Browse」をクリックしてフォルダを選択します (図 2-1 を参照)。

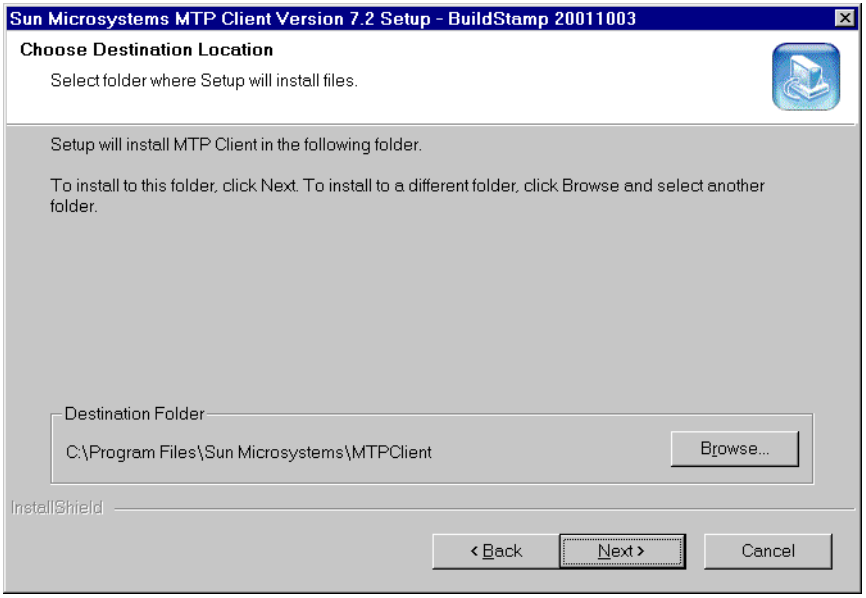

図 2-1　「Choose Destination Location」画面

3. **「Setup Type」画面で選択を行い、「Next」をクリックします。**

   インストールのタイプには、次の 3 種類があります。

   - Typical: すべての Sun MTP Client をインストールします。アプリケーションの開発と実行に適しています。
   - Compact: アプリケーションの実行に必要なものだけをインストールします。アプリケーションの開発はできません。
   - Custom: 「Select Components」画面が表示され、インストールするコンポーネントを選択できます。Production Files コンポーネントは、他のすべてのコンポーネントに必要です。

4. **インストールが完了すると、「Setup Complete」画面が表示されます。**

   システムを今すぐ再起動するかどうかを尋ねるメッセージが表示されます。
   Sun MTP Client ソフトウェアを使用するには、再起動が必要です。

   図 2-2 に示すように Sun MTP Client は、「スタート」メニューに登録されます。

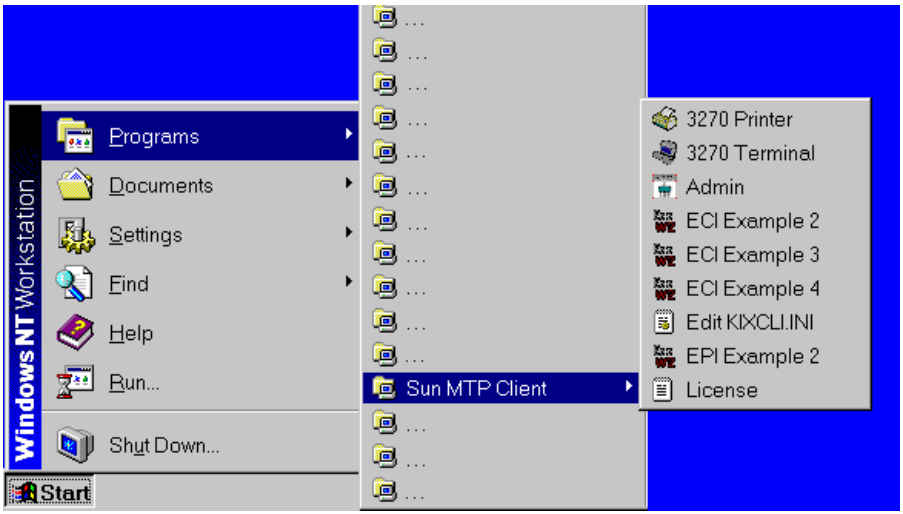

図 2-2　　Sun MTP Client の「スタート」メニュー

5. **インストールの完了後、`PATH` 環境変数に Sun MTP Client ソフトウェアの場所を追加する必要があります。**

   「Control」パネルで、環境変数 `PATH` を変更できます。

Sun MTP Client の UNIX へのインストール

UNIX 用の Sun MTP Client は、`tar` のイメージで提供されます。対象マシンの必要な場所にファイルをインストールできます。標準のインストールディレクトリは `/opt/kixcli` です。Sun MTP Client ソフトウェアの場所を `PATH` 環境変数に追加してください。

Sun MTP Client を標準ディレクトリにインストールすると、構成が簡単になります。ソフトウェアを標準とは別のディレクトリにインストールした場合、Sun MTP Client が提供する共有ライブラリにユーザーがアクセスできるように構成する必要があります。詳細は、ご使用のプラットフォームのシステムマニュアルを参照してください。インストールには標準ディレクトリの使用を推奨します。

第3章

Sun MTP Client と Sun MTP の構成

Sun MTP Client を使用して Sun MTP に接続する前に、2 つの手順を実行する必要があります。

- Sun MTP Client の構成 (9 ページの「Sun MTP Client の構成」)
- Sun MTP の構成 (15 ページの「Sun MTP Client に TCP/IP 接続するための Sun MTP の構成」)

Sun MTP Client の構成

Sun MTP Client により、Windows または UNIX を実行しているマシンを複数の Sun MTP 領域に同時に接続できます (図 1-1 を参照)。これらの各領域は、制御ファイル `KIXCLI.INI` の Systems セクションで、Sun MTP Client に対して定義する必要があります。このファイルの位置は、プラットフォームによって異なります。

- UNIX プラットフォーム: `/opt/kixcli/config`
- Windows: `C:\KIXCLI\CONFIG`

このファイルのデフォルトバージョンをコード例 3-1 に示します。

10 ページの「KIXCLI.INI ファイルへのシステムの追加」に説明されているように、Sun MTP Client を実行する前に、このファイルを編集して Sun MTP 領域を定義する必要があります。さらに、12 ページの「KIXCLI.INI ファイルでのクライアント動作の定義」に説明されているように、ファイルの General セクションのエントリを編集して Sun MTP Client のいくつかの動作を変更できます。

コード例 3-1　　KIXCLI.INI 制御ファイル

```
;-----------------------------------------------------------------
;
:Sun MTP Client Configuration File
;-----------------------------------------------------------------

[Systems]
Accounts=TCP,abc.mycompany.com,9111,Customer Accounting System
Payroll=TCP,def.mycompany.com,9111,Employee Payroll System
Ordering=TCP,ghi.mycompany.com,9111,Internal Ordering System
Test=TCP,555.555.55.5,9111,Test System
SnaSyst=MSSNA,RLUALIAS,LLUALIAS,MODENAME,MS SNA Connected System

[General]
DefaultSystem=Accounts
;TraceDir=C:\tmp
;TraceMask=0
;TraceType=External
;MsgDir=C:\tmp
;MaxRequests=20
;MaxSystems=3
;EnableConnect=false
```

KIXCLI.INI ファイルへのシステムの追加

Systems セクションの各行は、Sun MTP 領域を定義します。使用する領域ごとに Systems セクションにエントリを追加します。各領域のエントリ形式は、システムとの接続に使用するトランスポートプロトコル (TCP/IP または SNA) によって異なります。SNA のトランスポートとしての使用は、Sun MTP Client の Microsoft Windows バージョンだけでサポートされます。

TCP/IP 接続システムに必要なフィールド

すべてのフィールドが必須です。コメントを入力しない場合でも、ポート番号フィールドのあとにコンマ区切り記号を入力する必要があります。

次の表に、各エントリのフィールドを示します。

フィールド	説明
名前	Sun MTP Client が認識する領域名 (最大 8 文字) です。
トランスポートプロトコル	`TCP` の文字は、TCP/IP 接続システムの定義であることを示します。
ホストアドレス	領域が実行されているホストの TCP/IP アドレスです。
ポート番号	領域が Sun MTP Client からの TCP 接続を待機している TCP/IP ポート番号です。UNIX ホストでは、Sun MTP サーバー `unikixmain` を `-P` オプションとポート番号を指定して起動する必要があります。17 ページの「Windows での Sun MTP Client の起動」を参照してください。
コメント	最大 60 文字のコメントです。文字列を指定しなくてもかまいません。ただし、ポート番号フィールドを区切るために、コンマを入力する必要があります。

コード例 3-1 では、`Systems` セクションの最初の行で、ローカルで Accounts として認識されるシステムを定義しています。このシステムは TCP トランスポートを使用して、ホスト `abc.mycompany.com` で実行され、ポート 9111 で Sun MTP Client からの TCP 接続を待機している Sun MTP 領域に接続します。最後のフィールドで、Customer Accounting System というシステムの説明を指定しています。

SNA 接続システムに必要なフィールド

すべてのフィールドが必須です。コメントを入力しない場合でも、ポート番号フィールドのあとにコンマ区切り記号を入力する必要があります。

次の表に、各エントリのフィールドを示します。

フィールド	説明
名前	Sun MTP Client が認識する領域名 (最大 8 文字) です。
トランスポートプロトコル	`MSSNA` の文字は、SNA 接続システムの定義であることを示します。
遠隔 LU エイリアス	SNA サーバーで定義されているとおりに接続する Sun MTP 領域の LU エイリアスです。これは、SNA サーバー側で遠隔またはパートナー LU エイリアスと呼ばれます。

フィールド	説明
ローカル LU エイリアス	SNA サーバーで定義されているとおり Sun MTP Client が使用するローカル LU エイリアスです。これは、SNA サーバー側で LU エイリアスと呼ばれます。
モード名	SNA サーバーで定義されているように、ローカル LU と遠隔 LU 間の接続に使用する APPC モード名です。
コメント	最大 60 文字のコメントです。文字列を指定しなくてもかまいません。ただし、ポート番号フィールドを区切るために、コンマを入力する必要があります。

コード例 3-1 では、`Systems` セクションの最後の行で、ローカルで SnaSyst として認識されるシステムを定義しています。このシステムは、ローカル LU LLUALIAS を使用する Sun MTP に、Microsoft SNA または IBM Communications Manager を使用して接続します。この場合、Sun MTP には、RLUALIAS の遠隔 LU エイリアスがあります。使用される APPC モードは、MODENAME です。最後のフィールドでは、MS SNA Connected System というシステムの説明を指定しています。

KIXCLI.INI ファイルでのクライアント動作の定義

`KIXCLI.INI` ファイルの `General` セクションのエントリは、Sun MTP Client の動作を制御します。これは、オプションです。エントリは、「属性=値」の形式をとります。内容は次のとおりです。

注 – 不要なエントリをコメントにするには、セミコロンを使用します。初期状態では、コード例 3-1 に示すように、`General` セクションのほとんどのエントリがコメントアウトされています。

`DefaultSystem=`*system-name*

Sun MTP Client が使用するデフォルトシステムを識別します。システムは `Systems` セクションで定義されている必要があります。`DefaultSystem` が指定されていない場合、`KIXCLI.INI` ファイルの `Systems` セクションの最初のエントリがデフォルトとして設定されます。たとえば、このエントリがコメントされている場合でも、コード例 3-1 の場合は `Accounts` がデフォルトシステムになります。

`TraceDir=C:\TMP`

Sun MTP 診断追跡ファイルの保存先ディレクトリを制御します。各追跡ファイルは、`*.trc` という名前になります。`*` は、関連するタスクまたは処理を識別するファイル名のルートです。`TraceDir` が指定されていない場合、診断追跡ファイルは `$INSTROOT/BIN` に保存されます。

`TraceMask=0`

Sun MTP Client による診断追跡ファイルの生成を制御する 10 進整数です。ご購入先から診断追跡を取得するよう求められていない限り、このエントリはコメントアウトするか 0 に設定します。

診断追跡を使用する場合、次の点に注意してください。

- 追跡ファイル (`*.trc`) は、大量のディスク容量を一度に使用することがあります。そのため、できるだけ早く追跡を無効にして、必要のない追跡ファイルを削除してください。
- 診断追跡ファイルの生成は、Sun MTP Client のパフォーマンスを低下させます。

`TraceType=`*type*

追跡タイプは、診断追跡情報をどこに書き込むかを指定します。`External` または `Internal` を指定できます。

`External` を指定すると、追跡情報を直接ディスク上のファイルに書き込みます。これは、以前のリリースの動作と同じです。

`Internal` を指定すると、追跡情報をメモリー内の循環追跡バッファーに書き込みます。このバッファーは、次の場合にディスクにフラッシュされます。

- UNIX プラットフォームでは、コマンド `kixctl -D` が実行されたとき。
- Windows では、Windows サービスマネージャーなどから Windows サービスを一時停止したとき。

`MsgDir=C:\TMP`

標準機能の一部として、Sun MTP Client はメッセージを生成して、`MsgDir` ディレクトリの `KIXCLI.MSG` ファイルに書き込みます。`MsgDir` が指定されていない場合、ファイルは `$INSTROOT\BIN` ディレクトリに書き込まれます。`KIXCLI.MSG` ファイルは、Sun MTP Client が起動するたびに上書きされます。

`MaxRequests=20`

Sun MTP Client で可能な並行要求の最大数 (デフォルトは 20) を定義します。要求は、ECI 処理単位、EPI、または端末エミュレータセッションとして定義されます。このパラメータは、新しい処理単位を繰り返し開始して完了することのないループを発生する ECI アプリケーションなど、アプリケーションが誤って引き起こす可能性のある問題を制限するために設定します。

`MaxSystems=3`

Sun MTP Client が同時に接続できるシステムの最大数 (デフォルトは 3) を定義します。

EnableConnect=false

Systems 表示でマウスの右ボタンを押したときの Sun MTP Client の動作を構成できます。

true: Sun MTP 領域の接続または切断のメニューを表示します。この設定がデフォルトです。

false: 接続または切断メニューを表示しません。この機能は、ユーザーがシステムを選択できない環境のために用意されています。本番稼動では、この設定をお勧めします。

診断追跡の有効化と無効化

サポートの目的で、ご購入先から、Sun MTP Client の診断追跡を実行して診断情報を取得するよう求められることがあります。

▼ 診断追跡を実行する

1. Sun MTP Client を停止します。
2. Sun MTP Client 構成ファイル KIXCLI.INI の TraceDir 属性のコメント文字が削除されていて、指定したディレクトリが書き込み可能の状態で存在し、ディスクに診断追跡を出力するための十分な容量があることを確認します。
3. Sun MTP Client 構成ファイル KIXCLI.INI の TraceMask 属性のコメント文字が削除されていて、ご購入先のサポート担当者が指定した値に設定されていることを確認します。
4. Sun MTP Client 構成ファイル KIXCLI.INI の TraceType 属性のコメント文字が削除されていて、ご購入先のサポート担当者が指定した値に設定されていることを確認します。
5. 必要のない既存の診断追跡ファイル *.trc を TraceDir から削除します。
6. Sun MTP Client を再起動します。

注 – Windows では、管理プログラム KIXCTLG の「Control」パネルからでも診断追跡を実行できます。

▼ 診断追跡を無効にする

1. Sun MTP Client を停止します。
2. KIXCLI.INI ファイルを編集して TraceMask の設定を 0 に戻すか、TraceMask 行をコメントにして、診断追跡を無効にします。
3. Sun MTP Client を再起動する前に、追跡ファイル *.trc を別の場所にコピーします。

Sun MTP Client に TCP/IP 接続するための Sun MTP の構成

Sun MTP Client を使用して Sun MTP 領域に接続する前に、その領域で TCP/IP 接続の受信を受け入れるように構成する必要があります。ホストで Sun MTP を起動する際に、ポートを番号または名前で指定することで、Sun MTP Client は領域にアクセスできます。この名前は、/etc/services または NIS テーブルで既知のポート番号を検索するために使用します。

▼ Sun MTP を構成する

1. ホストで、/etc/services または NIS テーブルに既知のポートを、たとえば cicstcp 1435/tcp のように定義します。

   既知のポートは、1 つの領域に対して使用できます。同じマシンで実行している追加の領域で TCP 接続を待機する場合、各 Sun MTP 領域にそのマシンで一意のポートを指定する必要があります。

2. TCPRTERM 環境変数を Sun MTP 設定ファイルで定義します。

   Sun MTP Client および遠隔領域からの TCP/IP 接続に対する並行インバウンド要求の最大数を設定します。利用できるセッションよりも多くの要求がある場合、Sun MTP は余分な要求をキューに入れますが、パフォーマンスに影響することがあります。最小値は 2 に設定する必要があります。$TCPRTERM と $TCPSTERM を合わせて、200 以下にする必要があります。

   32K バイトの共有メモリーが、構成した各 $TCPRTERM には必要です。

3. `TCPSTERM` **環境変数を Sun MTP 設定ファイルで定義します。**

   遠隔領域への TCP/IP 接続に対する並行アウトバウンド要求の最大数を設定します。利用できるセッションよりも多くの要求がある場合、Sun MTP は余分な要求をキューに入れますが、パフォーマンスに影響することがあります。最小値は 2 に設定する必要があります。`$TCPRTERM` と `$TCPSTERM` を合わせて、200 以下にする必要があります。

   32K バイトの共有メモリーが、構成した各 `$TCPSTERM` には必要です。

4. `KIXMAXIST` **環境変数を Sun MTP 設定ファイルで定義します。**

   自動インストールされる Sun MTP Client および遠隔領域の最大数を設定します。遠隔領域からの要求は、その領域のインストールに利用できるエントリがない場合は拒否されます。

   詳細は、18 ページの「Sun MTP の接続受信の有効化」および『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

SNA で接続する Sun MTP Client のための Sun MTP および SNA の構成

Sun MTP Client のための Sun MTP および SNA の構成については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』の「Configuring Sun MTP for Remote Clients」を参照してください。

第4章

Sun MTP Client と Sun MTP の起動

この章では、ホストで Sun MTP Client および Sun MTP を起動する方法について説明します。この章の内容は、次のとおりです。

領域を起動する前に Sun MTP Client を起動することもできますが、起動していない領域には接続できません。領域を起動すると、TCP 要求および SNA 要求を待機します。アプリケーションを自動起動しても、領域が起動されていない場合は起動しません。

Windows での Sun MTP Client の起動

Windows 用の Sun MTP Client は、Windows サービスとして実行するように設計されています。通常、Sun MTP Client を起動する必要はなく、自動的に起動されます。ただし、Sun MTP Client の起動および停止の制御が必要な場合もあります。

ソフトウェアをインストールすると、マシンの起動時に開始するサービスが作成されます。しかし、このような動作モードを必要としない場合もあります。サービスが開始されたかどうかは、コントロールパネルのサービスで確認できます。
Sun MTP Client サービスに自動フラグが設定されている場合、マシンの起動時に開始されます。この設定を変更するには、「Startup」ボタンを押してオプションを選択します。

Sun MTP Client は、コントロールパネルのサービスオプションで起動できます。すべてのサービスの操作と同じように、サービスを選択して「開始」ボタンを押します。

Sun MTP Client の起動および停止には、コントロールパネルの他に Sun MTP Client Administrator も使用できます。26 ページの「UNIX 上での Sun MTP Client の管理」を参照してください。

UNIX での Sun MTP Client の起動

UNIX 用の Sun MTP Client は、デーモンプロセスとして実行するように設計されています。`kixcli` コマンドを使用してシェルから手動で起動することもできますが、一般的な操作方法ではありません。通常は、`inittab` から起動したデーモンとして Sun MTP Client を実行します。これにより、マシンの実行中は常に ECI/EPI 機能が使用できます。

`inittab` エントリを作成するには、ご使用のオペレーティングシステムのマニュアルでプラットフォームに適した構文を調べてください。`inittab` のエントリ例は、次のとおりです。

```
kixcli:2:once:/opt/kixcli/bin/kixcli >/dev/console 2>&1
```

Sun MTP の接続受信の有効化

`unikixmain` コマンドに `-P` オプションを指定して、ホストで Sun MTP を起動します。このオプションは Sun MTP に、`unikixtcp` サーバーを起動して、着信要求のポート番号を待機するように指示します。

`unikixmain` コマンドには、ファイル記述子の数を設定するための `-L` オプションも用意されています。

`-L` *connections*

Sun MTP が TCP 接続をサポートできるソケット接続の数です。この数は、1 つのプロセスがオープンできる接続の最大数を表します。接続できる TCP クライアント数を制限する場合は数値を小さくし、クライアントが拒否される場合は数を大きくします。

デフォルトは、現在のシステムのファイルに対するソフトリミット値です。

`-P` *port#*

TCP/IP 接続を介して ECI および EPI アプリケーションを実行するクライアントに対して、待機ポートとして使用する Sun MTP のポート名または番号です。デフォルトのポートがないので、名前または番号のどちらかを次のように指定する必要があります。

```
-P cicstcp
-P 5100
```

ポート番号は、クライアントの `INI` ファイルで識別されているシステムのポート番号と一致する必要があります。たとえば、コード例 3-1 の Payroll システムからの要求を待機する場合は、`-P` オプションの引数としてポート番号 `9111` を指定します。`cicstcp` は、15 ページの「Sun MTP Client に TCP/IP 接続するための Sun MTP の構成」の `/etc/services` で定義されている名前です。

`unikixmain` を直接開始するか、シェルスクリプト `kixstart` を使用して間接的に開始できます。`kixstart` は、コマンド行オプションを `unikixmain` に渡します。Sun MTP の起動手順については『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を、`unikixmain` 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。

第5章

Sun MTP Client の管理

この章では、Sun MTP Client の管理方法について説明します。内容は次のとおりです。

Windows 上での Sun MTP Client の管理

Windows 用の Sun MTP Client は、システムサービスとして実行されます。したがって、クライアント管理の一部は Windows のコントロールパネルから実行できます。ただし、実行できない管理機能もあります。

Sun MTP Client には、クライアントのモニターおよび管理を行うプログラム (`KIXCTLG.EXE`) が用意されています。`KIXCTLG.EXE` は、最大 4 つのタブがある単一ウィンドウのアプリケーションです。これらのタブを使用して、4 種類の情報を表示できます。それぞれについては、後続の節で説明します。

- 「Control」パネル
- 「TCP Systems」パネル
- 「MS SNA Systems」パネル
- 「Messages」パネル

「Control」パネル

「Control」パネルでは、以下の 2 つのタスクを実行できます。

- Sun MTP Client の起動と停止。Windows では、「Start」または「Stop」ボタンを押してシステムサービスを開始または停止します。
- 「Trace」による Sun MTP Client の追跡出力の動的な管理。追跡は、Sun MTP システム管理者の管理下でのみ使用します。

次の図は、「Control」パネルの例を示します。

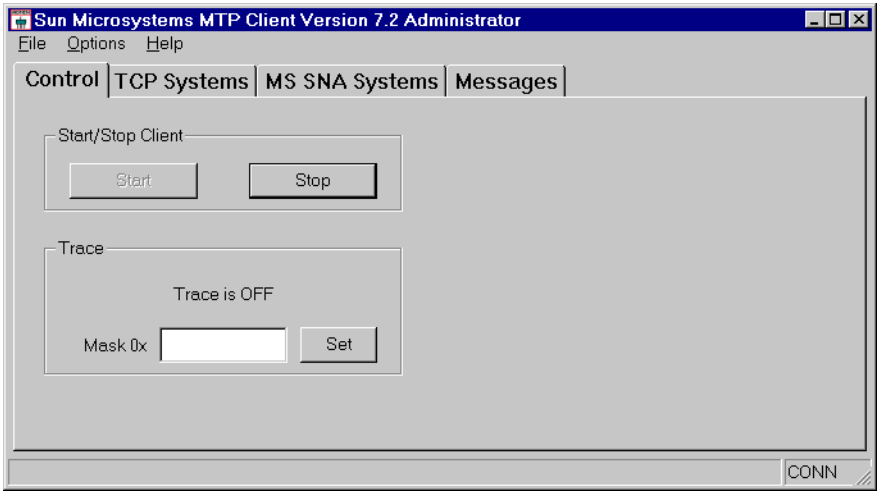

図 5-1　「Control」パネル

「TCP Systems」パネル

「TCP Systems」パネルは、KIXCLI.INI ファイルに TCP システムが定義されている場合にだけ表示されます。このパネルには、定義されている TCP システムおよびそれらを構成する値が表示されます。また、Sun MTP Client がこれらのシステムに接続されているかどうかも示されます。

各システムの左には次のようなアイコンが表示されます。

アイコン	説明
X	システムは接続されていません。
-	システムは接続処理中です。
(チェックマーク)	システムは接続されています。

リストにあるシステムは、名前のボタンを押すことで変更できます。ボタンを押すと、システムの接続または強制切断を行うメニューが表示されます。

注 – 切断は本当に必要な場合にだけ使用してください。

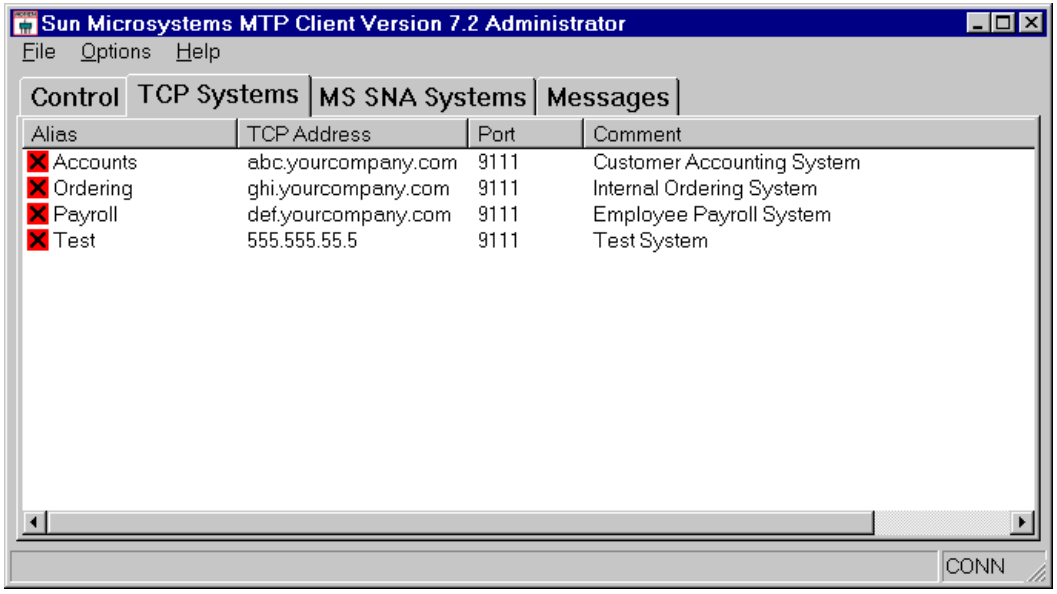

図 5-2　「TCP Systems」パネル

「MS SNA Systems」パネル

「MS SNA Systems」パネルは、KIXCLI.INI ファイルに MS SNA システムが定義されている場合にだけ表示されます。

このパネルには、定義されている MS SNA システムと、その構成値が表示されます。また、Sun MTP Client がこれらのシステムに接続されているかどうかも示されます。

各システムの左には次のようなアイコンが表示されます。

アイコン	説明
X	システムは接続されていません。
-	システムは接続処理中です。
(チェックマーク)	システムは接続されています。

リストにあるシステムは、名前のボタンを押すことで変更できます。ボタンを押すと、システムの接続または強制切断を行うメニューが表示されます。

注 – 切断は本当に必要な場合にだけ使用してください。

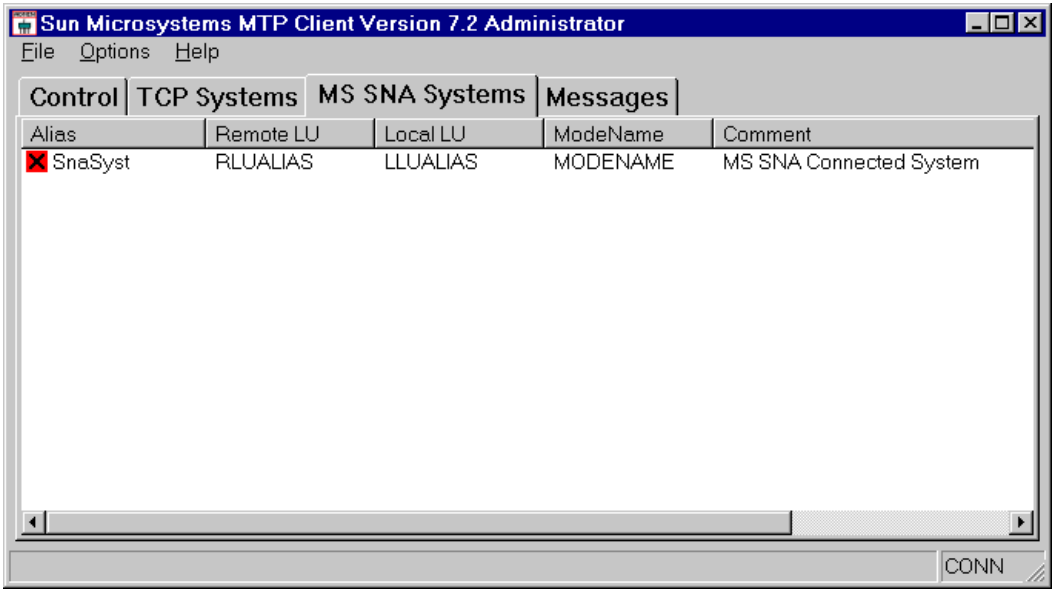

図 5-3　「MS SNA Systems」パネル

「Messages」パネル

「Messages」パネルには、Sun MTP Client が発行した最新 100 のメッセージが表示されます。3 つのメッセージカテゴリがあります。

- 情報 (I)
- 警告 (W)
- エラー (E)

メッセージファイル `KIXCLI.MSG` にも、同じメッセージが日付および時刻スタンプ付きで書き込まれます。Sun MTP Client メッセージについては、付録 B を参照してください。

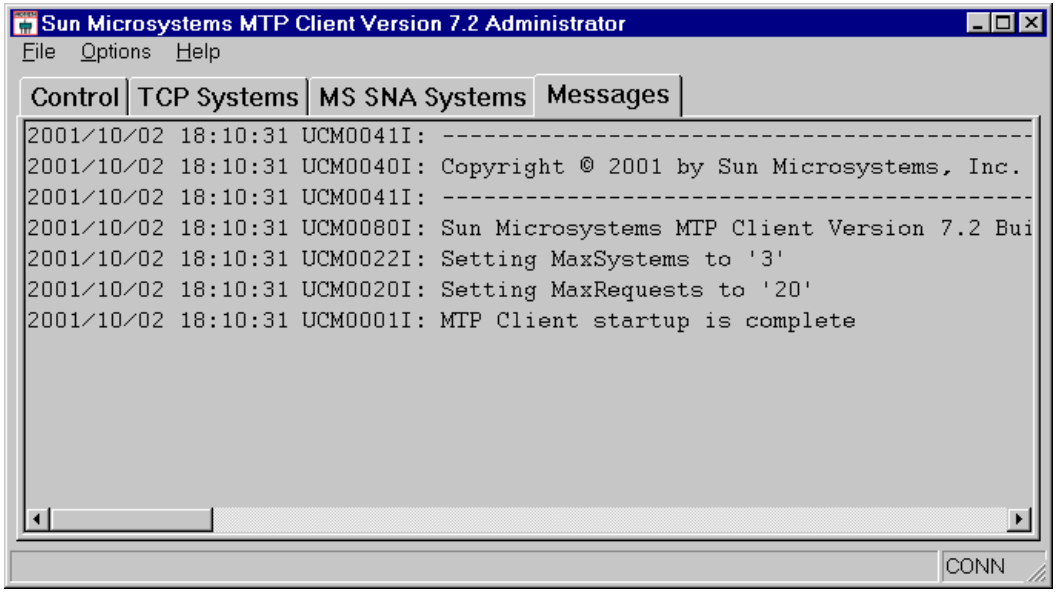

図 5-4　「Messages」パネルの例

UNIX 上での Sun MTP Client の管理

Solaris では、グラフィカルな Sun MTP Client の管理ツールは提供されていません。`kixctl` を使用する必要があります。

形式

`kixctl [-v] [-s] [-l] [-m] [-D] [-c` *system*`] [-d` *system*`] [-t` *mask*`]`

オプション	説明
`-v`	Sun MTP Client のバージョン情報を出力します。
`-s`	Sun MTP Client を停止します。
`-l`	定義済みのシステムと状態を表示します。
`-m`	システムの状態が変化すると、メッセージが表示されます。
`-D`	ダンプを実行します。 この機能は、システム管理者だけが実行します。
`-c` *system*	*system* で指定したシステムに接続します。
`-d` *system*	*system* で指定したシステムから切断します。
`-t` *mask*	追跡マスクを *mask* に設定します。 この機能は、システム管理者だけが実行します。

システム管理者は、システムで発生する重大なエラーやイベントに関する情報が書き込まれている `kixcli.msg` ファイルをモニターする必要があります。

第6章

3270 端末

3270 端末は、Sun MTP Client の EPI インタフェースを使用して、Sun MTP への 3270 ゲートウェイを提供する Windows アプリケーションです。この端末によって、Windows を実行している PC から標準 3270 Sun MTP トランザクションへの遠隔アクセスが可能になります。端末は、コマンド行のパラメータおよび初期設定ファイルで構成できます。

この章の内容は、次のとおりです。

注 – 3270 端末は、UNIX プラットフォームでは使用できません。

3270 端末の構成

端末の構成は、次の方法で行います。

- コマンド行のパラメータを指定して起動条件 (ウィンドウ名、システム、トランザクションなど) を構成します。28 ページの「コマンド行のパラメータを使用して構成する」を参照してください。
- 初期設定ファイルを使用して色、キーマッピング、およびフィールド属性を構成します。29 ページの「初期設定ファイルを使用して構成する」を参照してください。

▼ コマンド行のパラメータを使用して構成する

1. Sun MTP Client プログラムグループから、「3270 Terminal」アイコンを選択します。
2. 「Program Manager File」メニューから「Properties」を選択します。
3. kixterm.exe のダイアログボックスが表示されたら、次の構文に従ってコマンド行の必要なパラメータを入力します。
   - 各パラメータは、「-」または「/」文字のあとに大文字または小文字のパラメータ識別子を続けて指定します。
   - パラメータフラグを使用して渡すデータは、1 つ以上の空白文字で区切ります。
   - パラメータの引数に空白文字を含む場合は、次のように引数を引用符で囲みます。

     ```
     kixterm /t "CEBR" -w "Accounts Payable" /s Accounts
     ```

   次のコマンド行のパラメータを指定できます。

   /d *device-type*

   端末の Sun MTP デバイスタイプを指定します。16 文字を超える部分は切り捨てられます。Sun MTP の有効なモデルの一覧については、89 ページの「CICS_EpiAddTerminal()」を参照してください。

   /i *file-name*

   使用する初期設定ファイル名を指定します。指定されていない場合、端末はデフォルトの初期設定ファイル KIXTERM.INI を使用します。

   /n *netname*

   端末に割り当てる Sun MTP ネット名を指定します (最大 8 文字)。これは、TCT の「LU Name」フィールドの値です。通常はシステム管理者から指定されます。

   /s *system-name*

   コード例 3-1 の Accounts など、接続する Sun MTP 領域名を指定します (最大 8 文字)。端末を起動する前に、この名前を KIXCLI.INI に定義する必要があります。起動コマンド行で指定しない場合、システムを選択する選択ダイアログボックスが表示されます。

   /t *tranid*

   接続初期のトランザクションおよびデータを指定します。Sun MTP に正常に接続した直後に実行するトランザクションを指定します。

   /w *title*

   端末のタイトルバーに表示するウィンドウタイトルを指定します (最大 50 文字)。

▼ 初期設定ファイルを使用して構成する

端末は、デフォルトの初期設定ファイル KIXTERM.INI とともに提供されます。このファイルには、端末で使用する色およびキーボードの設定を含みます。

1. **デフォルトの初期設定ファイルを開くか、独自に作成します。**
2. **必要な設定を指定します。**
3. **初期設定ファイルを、Windows のデフォルトのシステムディレクトリに保存します。**

   インストール時、コピーが存在しない場合、KIXTERM.INI ファイルはデフォルトのシステムディレクトリに配置されます。KIXTERM.INI ファイルの形式と詳細については、付録 A で説明しています。

3270 端末の起動

この節では、「3270 Terminal」アイコンおよびコマンド行から端末を起動して、システムに接続する方法について説明します。

▼ 「3270 Terminal」アイコンから 3270 端末を起動する

1. **27 ページの「3270 端末の構成」で示すように 3270 端末を構成します。**
2. **Sun MTP Client グループから、「3270 Terminal」アイコンをダブルクリックします。**
3. **図 6-1 に示す「System Selection」ダイアログボックスが表示されたら、次のいずれかの方法でシステムへ接続します。**
   - システムをダブルクリックします。
   - システムをシングルクリックして、「OK」ボタンをクリックします。

注 – Program Manager でシステム名を使用して端末を構成した場合、「System Selection」ダイアログボックスは表示されません。

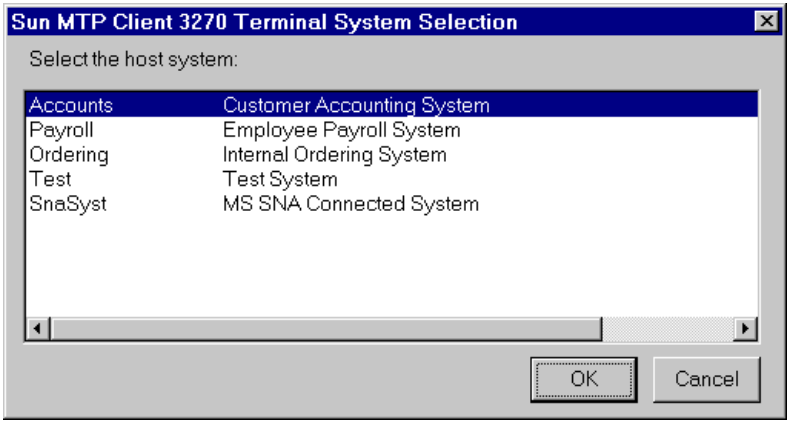

図 6-1　「3270 Terminal System Selection」ダイアログボックス

▼ コマンド行から 3270 端末を起動する

1. 「Program Manager File」メニューから「`Run`」を選択します。
2. `kixterm` コマンドと必要なパラメータをコマンドボックスに入力します。

   パラメータおよび構文については、28 ページの「コマンド行のパラメータを使用して構成する」を参照してください。

注 – 「System Selection」ダイアログボックスの表示を省略するには、`/s` パラメータとシステム名を指定します。これにより、3270 端末は指定されたシステムへの接続を自動的に行います。

「3270 Terminal」画面

起動時の「3270 Terminal」画面は、次のいずれかです。

- コマンド行にトランザクションを指定していない場合は、空白画面。
- 指定したトランザクション画面。次の図に画面の例を示します。

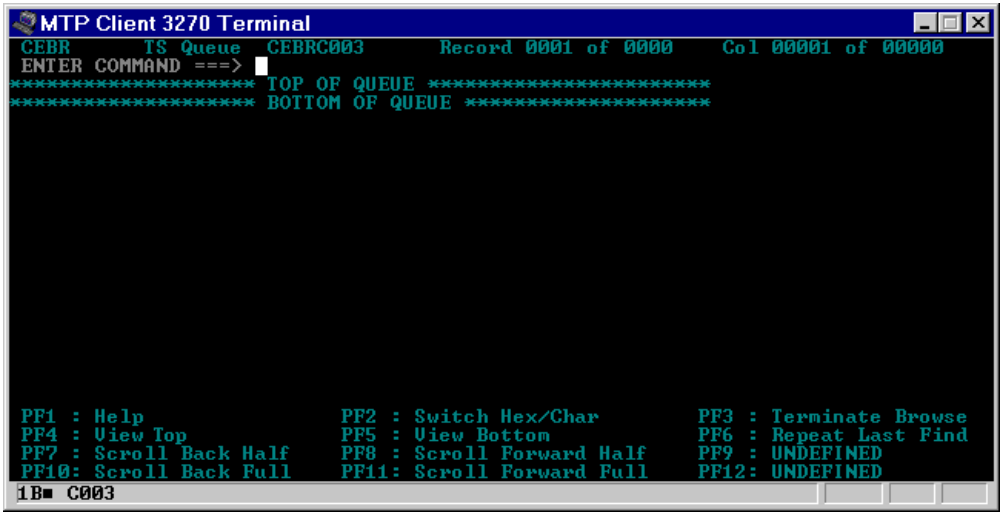

図 6-2 「3270 Terminal」画面

各部の説明は次のとおりです。

タイトルバー	端末に割り当てられたタイトルを表示します。デフォルトは、「Sun MTP Client 3270 Terminal」です。コマンド行のパラメータで `-W` または `/W` を指定して変更できます。
表示領域	Sun MTP トランザクションの入力と、トランザクションによるデータを表示する領域です。
ステータスバー	現在の端末セッション状態を、表 6-1 に示す情報で表示します。

表 6-1　3270 端末のステータスバー

表示される情報	例	説明
接続状態	1B#	接続が確立されたときに左下隅に表示される記号です。
端末ネット名	=2AAA3M	遠隔領域の端末名です。
キーボードロック	XSystem	キーボードがロックされたときに表示される記号です。
エラーメッセージ	EMU0014E:System unavailable	キーボードの状態フラグです。
キーボード状態フラグ	CAPS	Caps lock インジケータです。
	NUM	Num lock インジケータです。
	INS	挿入モードフラグです。

3270 端末の停止

▼ 3270 端末を停止する

1. CSSF LOGOFF トランザクションで接続を閉じます。
2. 3270 端末画面の左上隅の「Control-menu」ボックスをダブルクリックします (図 6-2)。

第7章

3270 プリンタ

3270 プリンタは、Sun MTP Client の EPI インタフェースを使用して、Sun MTP への 3270 ゲートウェイを提供する Windows アプリケーションです。このプリンタによって、プリンタベースのアプリケーションは、データを仮想プリンタデバイスへ送信できます。このプリンタは、次のどちらかを実行できます。

- データをフォーマットし、ローカルホストのテキストファイルに書き込む
- データをフォーマットして一時ファイルに書き込み、次にファイルを処理するコマンドを実行する

ファイルに書き込まれたデータを編集して、スプレッドシート、ワードプロセッサドキュメント、またはその他のアプリケーションに取り込むことができます。

このプリンタは、『IBM 3270 Information Display System Data Stream Programmers Reference』に指定されている出力規則に従ってデータをフォーマットします。

この章の内容は、次のとおりです。

注 – 3270 プリンタは、UNIX プラットフォームでは使用できません。

3270 プリンタの構成

この節では、コマンド行のパラメータを使用してプリンタを構成する方法について説明します。

▼ プリンタを構成する

1. **Sun MTP Client プログラムグループから、「Sun MTP Printer」アイコンを選択します。**
2. **「Program Manager File」メニューから「`Properties`」を選択します。**
3. **`kixprnt.exe` のダイアログボックスが表示されたら、必要なコマンド行のパラメータを次のように入力します。**
   - 各パラメータは、「–」または「`/`」文字のあとに大文字または小文字のパラメータ識別子を続けて指定します。
   - パラメータフラグを使用して渡すデータは、1 つ以上の空白文字で区切ります。
   - パラメータの引数に空白文字を含む場合は、次のように引数を引用符で囲みます。

     ```
     kixprnt /t "hp sys1" /f c:\acct.txt  /s Accounts
     ```

   次のコマンド行のパラメータを指定できます。

   `/d` *device-type*

   Sun MTP 仮想プリンタデバイスのデバイスタイプを指定します (最大 16 文字)。

   `/f` *print-file*

   フォーマット済み出力を追加するファイルを指定します。ファイル名または出力コマンドを指定しない場合、プリンタはデフォルトのファイル `kixprnt.txt` を開いて追加します。他のアプリケーションで使用する目的で、プリントファイルを編集したりデータを抽出したりできます。プリンタによるファイルの削除はありません。`/f` および `/p` の両方が指定されている場合、プリンタは `/f` で指定されたファイルを無視し、`/p` で示されるようにプリンタが作成した一時ファイルに書き込みます。

   `/n` *netname*

   仮想プリンタデバイスに割り当てる Sun MTP ネット名を指定します (最大 8 文字)。これは、Sun MTP TCT の「LU Name」フィールドの値です。システム管理者に確認してください。

/p *command*

出力要求の受信時にプリンタが実行するコマンドを指定します。処理対象の一時ファイル名をコマンド文字列の最後に入力します。たとえば、次のコマンドを考えます。

```
kixprnt /p hppmt
```

プリンタは、このコマンドの一時ファイル名を次のように追加します。

```
kixprnt /p hppmt  c:\tmp\123
```

ユーザーコマンドで、一時ファイルを削除する必要があります。35 ページの「kixprnt コマンド例」も参照してください。

/s *system-name*

接続する Sun MTP 領域の名前を指定します (最大 8 文字)。起動時に指定しない場合、「System Selection」ダイアログボックスでシステムを選択します。システムを KIXCLI.INI ファイルに追加する方法については、10 ページの「KIXCLI.INI ファイルへのシステムの追加」を参照してください。

/t *tranid* [*data*]

プリンタが正常に Sun MTP へ接続されたときに実行するトランザクションを指定します。初期トランザクションでデータを渡す場合、次のようにトランザクション ID とデータを引用符で囲む必要があります。

```
kixprnt /t "CEBR testq"
```

kixprnt コマンド例

この節では、kixprnt コマンドの例を示します。

例 1

テキストを c:\ ディレクトリの 3270.txt ファイルに書き込みます。

```
$ kixprnt /f c:\3270.txt
```

例 2

出力テキストを一時ファイルに書き込み、コマンド行の最後にある一時ファイル名を使用して、ユーザーが作成したコマンド usercmd を実行します。

```
$ kixprnt /p usercmd
```

例 3

先に実行したコマンドが一時ファイル `c:\tmp\453` を作成した場合、次のユーザーコマンドを実行します。

```
$ usercmd c:\tmp\453
```

例 4

一時ファイル名は、常にコマンドの最後に入力します。たとえば、次のように `kixprnt` コマンドに `/c` および `/g` として定義されたフラグがある場合を考えます。

```
$ kixprnt /p "usercmd /c /g"
```

この場合、実行するユーザーコマンドは次のとおりです。

```
$ usercmd /c /g c:\tmp\453
```

3270 プリンタの起動

この節では、プリンタの起動およびシステムへの接続の 2 つの方法について説明します。

- 「3270 プリンタ」アイコン
- コマンド行

▼ 「3270 Printer」アイコンから 3270 プリンタを起動する

1. **Sun MTP Client グループから、「3270 Printer」アイコンをダブルクリックします。**
2. **図 7-1 の「System Selection」ダイアログボックスが表示されたら、次のいずれかの方法で接続するシステムを選択します。**
   - システムをダブルクリックします。
   - システムをシングルクリックして、「OK」を選択します。

注 – Program Manager でシステム名を使用してプリンタを構成した場合、「System Selection」ダイアログボックスは表示されません。

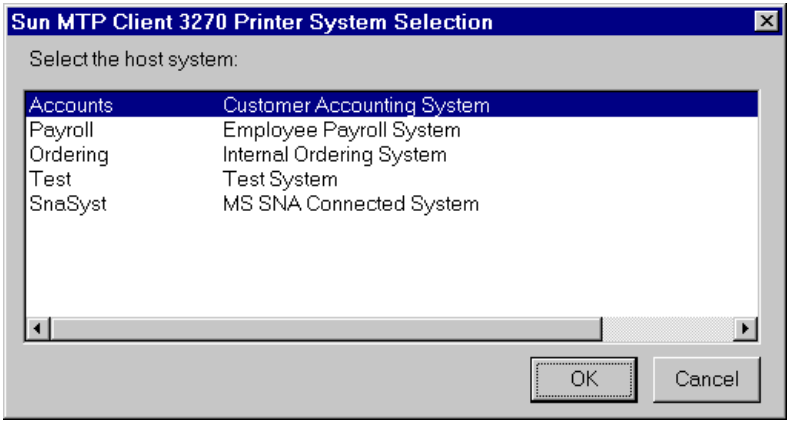

図 7-1 「3270 Printer System Selection」ダイアログボックス

▼ コマンド行から 3270 プリンタを起動する

1. 「Program Manager File」メニューから「`Run`」を選択します。
2. 必要なパラメータを使用して、`kixprnt` コマンドをコマンドボックスに入力します。

   パラメータの説明は、34 ページの「3270 プリンタの構成」を参照してください。

「3270 Printer」画面

起動時には、デフォルトでプリンタがアイコンとして表示されます。Sun MTP 領域に接続されると、プリンタがインストールされているネット名がアイコンの下に表示されます。

図 7-2 「3270 Printer」アイコン

プリンタウィンドウには、次の図に示すように出力ファイル、出力コマンド、およびエラーメッセージの詳細が表示されます。

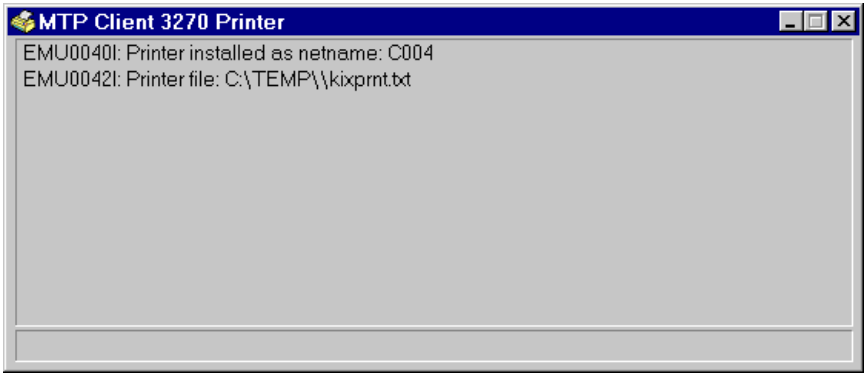

図 7-3　「3270 Printer」画面

3270 プリンタの停止

▼ プリンタを停止する

- **次のいずれかの方法を使用します。**
  - メインウィンドウの左上隅にあるシステムメニューから、「`close`」オプションを選択します。
  - アイコンを使用します。

第8章

外部呼び出しインタフェース (ECI)

この章では、C プログラミング言語で使用する場合の Sun MTP Client 外部呼び出しインタフェース (ECI) について説明します。C の関数呼び出しおよび定義は、C++ で使用できます。この章の内容は、次のとおりです。

ECI のコード例

C プログラミング言語での ECI の使用方法を示すコード例が、$INSTROOT\EXAMPLES に用意されています。$INSTROOT は、Sun MTP Client がインストールされているディレクトリ名を指します。

Sun MTP ECI の動作の仕組み

Sun MTP ECI によって、アプリケーションは Sun MTP 領域の Sun MTP プログラムを呼び出せます。ECI は、Sun MTP プログラムにアクセスする方法です。つまり、EXEC CICS コマンドを発行しないで、Sun MTP に処理を行うプログラムを実行するように命令します。Sun MTP プログラムは、COMMAREA オプションを使用して EXEC CICS LINK が呼び出したかのように実行されます。

ECI アプリケーションは、Sun MTP プログラムが次の分散プログラムリンク (DPL) の規則に準拠して作成されている限り、これらのプログラムを呼び出せます。

- 拡張 LUW で、EXEC CICS SYNCPOINT コマンドまたは EXEC CICS SYNCPOINT ROLLBACK コマンドを使用しない。
- リンク先プログラムに対して、PPT の APISet フィールドに D を指定して DPL 制限 API コマンドだけを実行する。
- 基本機能に対して CICS コマンドを実行しない。

アプリケーションは、Sun MTP 領域が同じでも異なっても、複数の Sun MTP プログラムを並行して実行できます。

ECI を使用するには、次の点に注意が必要です。

- 1 つの ECI 処理単位は、複数の領域にまたがることができません。処理単位とは、アクションがコミットされる前に完了する必要のある一連のアクションです。
- 1 つの ECI 処理単位では、複数アクションを一度に実行できません。

完全な ECI 機能は、2 つの API 関数呼び出しで実現されます。

`CICS_ExternalCall()`

ECI のほとんどの機能を提供します。41 ページの「CICS_ExternalCall 呼び出しタイプ」で説明します。この関数は、1 つのパラメータ (ECI_PARMS ブロックへのポインタ) を受け取ります。この制御構造体のフィールドで、実行する機能を定義します。49 ページの「CICS_ExternalCall()」で、この関数について説明します。

`CICS_EciListSystems()`

使用する領域を照会するインタフェースを提供します。52 ページの「CICS_EciListSystems()」で、この構文について説明します。

CICS_ExternalCall 呼び出しタイプ

CICS_ExternalCall() の呼び出しには、次の 3 つのタイプがあります。

プログラムリンク

Sun MTP 領域のプログラムを実行します。

状態情報

特定のシステムに関する接続情報を取得します。

応答請求

以前のプログラムリンク呼び出しの結果および状態を取得します。

これらタスクを実行するには、ECI_PARMS 制御ブロックの eci_call_type フィールドに必要な値を設定する必要があります。設定できる値を、表 8-1 に示します。

表 8-1 関数に対する eci_call_type

機能	eci_call_type の値
プログラムリンク	ECI_SYNC
	ECI_ASYNC
	ECI_ASYNC_NOTIFY_MSG (Windows のみ)
	ECI_ASYNC_NOTIFY_SEM (Windows のみ)
状態要求	ECI_STATE_SYNC
	ECI_STATE_ASYNC
	ECI_STATE_ASYNC_MSG (Windows のみ)
	ECI_STATE_ASYNC_SEM (Windows のみ)
応答請求	ECI_GET_REPLY
	ECI_GET_REPLY_WAIT
	ECI_GET_SPECIFIC_REPLY
	ECI_GET_SPECIFIC_REPLY_WAIT

表 8-1 の分類に加えて、同期呼び出しと非同期呼び出しを区別する必要があります。

同期呼び出し

この機能は、要求が完了するまでアプリケーションへ制御を戻しません。つまり、要求がネットワークを介して Sun MTP 領域に伝えられ、プログラムがスケジューリングされて完了すると応答が送信されます。これには、特に帯域幅が狭いネットワークで輻輳するシステムでは、相当な時間がかかる場合があります。この間、同期呼び出しを行っているアプリケーションは要求に対応できないので、他のタスクは実行できません。

注 – このように動作することから、同期呼び出しはお勧めできません。

非同期呼び出し

要求がスケジューリングされるとすぐに、関数がアプリケーションに戻ります。つまり、要求がネットワークに送られる前に、アプリケーションが制御を得ます。アプリケーションは他の処理を実行でき、要求が完了すると非同期で通知されます。要求結果は、応答請求呼び出しを使用して取得します。

プログラムリンク呼び出し

プログラムリンク呼び出しは、同期または非同期のどちらかです。非同期呼び出しの場合、応答請求呼び出しを使用して応答を請求するには、非同期で呼び出したアプリケーションで処理することが必要です。詳細は、42 ページの「応答請求呼び出し」を参照してください。

プログラムリンク呼び出しは、処理単位の一部としてプログラムを開始するか、処理単位を完了します。プログラムリンク呼び出しで実行できる機能は、次のとおりです。

- 単発の処理単位の実行
- 新しい処理単位の開始
- 既存の処理単位の続行
- 処理単位の同期点化
- 処理単位のロールバック

応答請求呼び出し

応答請求呼び出しは、非同期プログラムリンクまたは非同期状態情報呼び出し後、情報を戻します。応答請求呼び出しには、次の 2 つのタイプがあります。

- 一般: 未処理の情報を取り出します。
- 固有: 指定された非同期要求の情報を取り出します。

非同期方式の呼び出しを使用するアプリケーションでは、いくつかのプログラムリンクおよび状態情報呼び出しが未処理になっている場合があります。ECI パラメータブロックの `eci_message_qualifier` パラメータを非同期呼び出しで使用すると、呼び出しに対するユーザー定義の識別子を指定できます。

注 – 単一のアプリケーション内でそれぞれの非同期呼び出しに異なる識別子を割り当てるのは、プログラマの責務です。

一般的な応答請求呼び出しが行われると、ECI は、eci_message_qualifier フィールドを使用して、応答が属する呼び出し名を戻します。固有の応答請求呼び出しが行われた場合、求められている情報に関する非同期呼び出しを特定するために、eci_message_qualifier フィールドに値を設定する必要があります。

状態情報呼び出し

状態情報呼び出しは、同期または非同期のどちらかで呼び出すことができます。非同期呼び出しの場合、呼び出したアプリケーションが応答請求呼び出しにより応答の送信を取得します。詳細は、42 ページの「応答請求呼び出し」を参照してください。

状態情報呼び出しによって、次のタスクを実行できます。

- アプリケーションを実行しているシステムのタイプおよび所定の領域との接続についての照会。状態を戻す COMMAREA を指定する必要があります。
- 状態が指定された状態と異なる場合に通知する要求の設定。指定された状態について記述する COMMAREA を指定する必要があります。この目的では、非同期呼び出しだけが使用可能です。
- 状態変化の通知に対する要求の取り消し。COMMAREA は不要です。

状態要求ブロックの形式については、47 ページの「ECI データ構造体」を参照してください。

アプリケーション設計

アプリケーションの設計は、Solaris 環境と Windows 環境では異なります。これは、通常これらの環境でアプリケーションが準拠するパラダイムが異なることによります。この節では、すべてのプラットフォームに共通する論理処理単位の管理について説明し、Windows 環境および Solaris 環境でのプログラミングに対するガイドラインを示します。

論理処理単位の管理

ECI アプリケーションは、Sun MTP 領域の回復可能なリソースの更新に関連することがあります。アプリケーションのプログラマは、ECI が論理処理単位を管理する機能について理解する必要があります。論理処理単位は、回復可能なリソースに対する一連の更新を設定するために必要な領域でのすべての処理です。論理処理単位が正常に終了すると、変更がすべてコミットされます。たとえば、プログラムが不正終了するなど、論理処理単位が異常終了した場合、変更はすべて取り消されます。ECI を使用して、領域での論理処理単位を開始および終了できます。

論理処理単位の回復可能なリソースに対する変更は、次の呼び出しの影響を受けます。

- 単一のプログラムリンク呼び出し。
- 連続したプログラムリンク呼び出し。論理処理単位に対する一連の呼び出しの最初の呼び出しから正常に復帰すると、制御ブロックの `eci_luw_token` フィールドには、同じ論理処理単位に関連する以降のすべての呼び出しで使用されるトークンが含まれます。同じ論理処理単位のすべてのプログラムリンク呼び出しが、同じ領域に送信されます。

注意 – 論理処理単位を複数のプログラムリンク呼び出しに拡大すると、それが長時間に及ぶことがあるので注意してください。論理処理単位は、ロックを保持し、領域上の他の Sun MTP リソースも保持します。これによって、同じロックおよびリソースを待機している他のユーザーに遅れを生じさせる場合があります。

論理処理単位が終了すると、Sun MTP は変更をコミットしようとします。コミットが正常に行われたら、論理処理単位の最後のプログラムリンクだけを呼び出してください。

1 つの論理処理単位につき 1 つのプログラムリンク呼び出しが、未処理にできます。非同期プログラムリンク呼び出しは、応答請求呼び出しが応答を処理するまでは未処理です。

ECI から処理単位を操作するのに必要な手法を、55 ページの「一般 ECI シナリオ」に示します。この手法を使用することで、アプリケーションは処理単位内の制御フローを管理して、必要なタスクを実行できます。

各論理処理単位は、実行中に、Sun MTP の 1 つの非機能タスクと関連付けます。つまり、予想される最大数の並行呼び出しに対応できる十分なフリータスクを領域に定義する必要があります。タスク数は、15 ページの「Sun MTP Client に TCP/IP 接続するための Sun MTP の構成」に示す環境変数 `TCPRTERM` と、Sun MTP が起動したときに作成されるトランザクション処理プログラムの数に影響されます。たとえば、同じ Sun MTP 領域に対して 6 つの並行処理単位が必要な場合、`$TCPRTERM` を最低 6 に設定します。VSAM 構成テーブル (VCT) のトランザクション処理プログラムの数も最低 6 に設定する必要があります。

ECI アプリケーションは、一度に 1 つの処理単位だけ実行するようには制限されていません。アプリケーションは、1 つまたは複数の Sun MTP 領域に指定された複数の処理単位を同時に実行できます。これらの処理単位は互いに依存しないで、協調して動作するようにアプリケーションで制御できます。たとえば、アプリケーションが、2 つの異なる領域からデータを取得し画面に表示する必要がある場合があります。各領域に対する ECI 呼び出しを使用してデータの取得が行われます。両方が同時に実行されますが、異なる処理単位内です。次に、アプリケーションによってデータが受信され、エンドユーザーに表示されます。

Windows 用アプリケーションの設計

この節では、Windows 環境で最適に機能するプログラミング方法について説明します。

Windows はマルチスレッド化された環境なので、同期 ECI 呼び出しを使用してアプリケーションを設計することが可能で、それが望ましい場合があります。これらの呼び出しが複数のスレッドから実行される場合、複数の同期処理単位を常に動作させることが可能です。ただし、通常マルチスレッド化されたアプリケーションでは、複数のスレッドの同期をとるために多くの作業が必要となります。

Windows 環境内の非同期呼び出しで使用できる通知機能を使用するのが、より簡単で効果的です。これによって、アプリケーションを通常の Windows アプリケーションとして書き込むことができ、さまざまなメカニズムを介して DPL の完了を通知させることができます。したがって、ECI は残りのアプリケーションの設計に簡単に適合できます。

通知メカニズムで最も重要なのは、Windows メッセージです。このメカニズムを使用すると、Windows メッセージを送信することによって、作業が完了したことがアプリケーションに通知されます。メッセージパラメータは、各 ECI 要求に対して発生した内容をアプリケーションに通知します。次に、アプリケーションは応答請求呼び出しを使用して、作業の完了結果を取得します。アプリケーションが主にユーザーに対する表示に関連している場合、このメッセージ送信方法は、通常の Windows プログラミングモードに適合します。ECI の完了についてアプリケーションに対して送信されたメッセージは、他のすべての Windows メッセージと同じです。

他に、セマフォーおよびコールバックの 2 つの通知メカニズムがあります。セマフォー通知は、実際には通知が Windows イベントを介して行われるため、正しくはセマフォーではありません。

アプリケーションはイベントを作成し、ECI 要求が完了すると、ECI はイベントに信号を送信します。次に、アプリケーションは `WaitForSingleObject()` システムコールおよび `WaitForMultipleObjects()` システムコールを使用して、要求の完了を待機します。

コールバック通知メカニズムの利用頻度は低いですが、アプリケーションで必要とされる場合があります。このメカニズムを使用すると、ECI 要求が完了したときにユーザー定義関数を実行できます。コールバックルーチンで、大量の作業を実行しないでください。アプリケーションでは、コールバックで ECI を呼び出さないでください。コールバックは通常、応答を受信するために必要な作業があることをアプリケーションに通知します。Windows では、このコールバックは、ECI が所有する別のスレッドで実行されます。

呼び出しの完了をアプリケーションに通知するのに、ECI は使用されません。すなわち、適切なときに応答の送信を請求するのはアプリケーションの役割です。応答が準備できていない場合、関連するリターンコードが ECI の送信請求呼び出しから戻されます。

UNIX 用アプリケーションの設計

この節では、UNIX 環境で最適に機能するプログラミング方法について説明します。

アプリケーションの設計には、次の 2 つのモードを選択できます。

- 同期モード

  DPL 処理時アプリケーションはブロックします。多くの場合、これで十分です。

- 非同期モード

  アプリケーションには、一度に多くの処理単位を必要とするものがあります。UNIX の実装では、非同期要求の通知方法は 1 つだけ (コールバック) です。ただし、このコールバックメカニズムにはいくつかの設計上の制約事項があります。

Sun MTP Client は、名前付きパイプを使用してアプリケーションと通信します。Sun MTP Client はシングルスレッド化されているので、アプリケーションが ECI/EPI インタフェースを呼び出す場合、データはパイプからのみ読み取ることができます。すべてのコールバックの実行は、この場合のみです。そのため、通知コールバックは他の ECI/EPI 関数の処理中にだけ実行されます。この制限によって、効果的に動作するアプリケーションを設計するときに問題が発生する場合があります。この処理を簡単にするために、`KixCli_QueryFD()` 関数を使用します。これは、通信に使用するパイプの標準 UNIX ファイル記述子を戻します。アプリケーションは、`select()` 関数呼び出しを使用して、処理する ECI/EPI の情報が設定されるまで待機します。アプリケーションは、データがあることを伝えられると、ECI/EPI API を呼び出してコールバックが実行されるようにします。

サンプルアプリケーション `ECIEX2` では、このメカニズムがユーザー入力を同時に処理しながら、複数の並行処理単位を実行する `curses` アプリケーションにどのように適合するかを示しています。

ECI データ構造体

次の ECI データ構造体をそれぞれの表に示します。

- ECI_STATUS、表 8-2
- ECI_PARMS、表 8-3

表 8-2　ECI_STATUS 構造体のフィールド

フィールド	値	説明
ConnectionType	ECI_CONNECTED_NOWHERE	Sun MTP Client では使用しません。
	ECI_CONNECTED_TO_CLIENT	アプリケーションは、Sun MTP Client システムで実行されています。
	ECI_CONNECTED_TO_SERVER	使用しません。
CicsServerStatus	ECI_SERVERSTATE_UNKNOWN	使用しません。
	ECI_SERVERSTATE_UP	Sun MTP 領域は現在接続されていて、作業に使用できます。
	ECI_SERVERSTATE_DOWN	Sun MTP 領域は、作業に使用できません。
CicsClientStatus	ECI_CLIENTSTATE_UNKNOWN	使用しません。
	ECI_CLIENTSTATE_UP	ECI Client は使用可能です。
	ECI_CLIENTSTATE_INAPPLICABLE	使用しません。

表 8-3　ECI_PARMS 構造体のフィールド

フィールド	型	説明
eci_call_type	sshort	eci_extend_mode フィールドによって変更される一次呼び出しタイプです。要求されている呼び出しをこのフィールドとともに定義します。
eci_program_name	char[8]	Sun MTP 領域上で実行するプログラム名です。
eci_userid	char[8]	領域のセキュリティーチェック用ユーザー ID です。
eci_password	char[8]	領域のセキュリティーチェック用パスワードです。
eci_transid	char[4]	プログラムが実行されるトランザクション名です。EIBTRNID フィールドを介して Sun MTP プログラムで使用できます。

表 8-3　ECI_PARMS 構造体のフィールド (続き)

フィールド	型	説明
eci_abend_code	char[4]	障害が発生したプログラムから戻される不正終了コードです。ECI Client 側のエラーにも設定できます。
eci_commarea	char *	COMMAREA へのポインタです。呼び出しのタイプによって、データがこの領域から送信またはこの領域に設定 (あるいはその両方) されます。
eci_commarea_length	sshort	eci_commarea ブロックの長さです。
eci_timeout	sshort	要求に対するタイムアウト (秒) です。通常は、タイムアウトなしを示す 0 に設定します。一部のインスタンスのみで、他の値に設定します。
eci_sys_return_code	sshort	障害に関する拡張情報を提供するリターンコードです。このフィールドは、ECI Client から設定されません。
eci_extend_mode	sshort	eci_call_type フィールドに対する修飾子です。eci_call_type とともに、このフィールドは必要な機能を完全に定義します。
eci_window_handle	HWND	非同期呼び出しの完了時に、送信メッセージを受信するウィンドウのウィンドウハンドルです (Windows のみ)。
eci_hinstance	HINSTANCE	ECI Client では無視されます。
eci_message_id	ushort	非同期呼び出しの完了時に、eci_window_handle ウィンドウに送信されるメッセージ ID です。
eci_message_qualifier	sshort	異なる非同期呼び出しの識別に使用するユーザー定義の値です。
eci_luw_token	slong	動作している処理単位を識別するために、ECI 呼び出しによって戻されたり、入力として使用するトークンです。
eci_sysid	char[4]	ECI Client では無視されます。
eci_version	sshort	使用中の ECI のバージョンです。新しいアプリケーションの場合、ECI_UNIKIX_VERSION_1 に設定します。ただし、Sun MTP Client は、IBM Client ECI との互換性を保つために、ECI_VERSION_1 もサポートします。
eci_system_name	char[8]	INI ファイルの [Systems] セクションで指定されている Sun MTP 領域の名前です。CICS_EciListSystems() 呼び出しを使用して取得します。このパラメータに NULL を指定すると、デフォルトシステムを使用できます。一部の呼び出しでは、このパラメータは無視されます。
eci_callback	func*	非同期要求の完了時に呼び出されるコールバックルーチンへのポインタです。
eci_userid2	char[16]	無視されます。
eci_password2	char[16]	無視されます。

表 8-3　ECI_PARMS 構造体のフィールド (続き)

フィールド	型	説明
eci_tpn	char[4]	CPMI の代わりにプログラムで実際に実行されるトランザクション ID です。この値は、ECI_VERSION_1 および ECI_UNIKIX_VERSION_1 では無視されます。

ECI 関数

この節では、2 つの ECI 関数の構文について説明します。

CICS_ExternalCall()

CICS_ExternalCall() によって、プログラムリンク呼び出し、状態情報呼び出し、および応答請求呼び出しへアクセスできます。ECI パラメータブロックの eci_call_type フィールドで、実行する関数を制御します。

形式

```
cics_sshort_t  CICS_ExternalCall(ECI_PARMS  *EciParms);
```

EciParms は ECI パラメータブロックへのポインタです。呼び出しのタイプ別に、ブロック内にフィールドが必要です。後続の節で、CICS_ExternalCall() の通常の使用に必要なフィールドについて説明します。

この関数を呼び出す前に、ブロック全体を NULL に設定してから、関連のフィールドを設定します。ECI_PARMS 構造体の各フィールドについては、表 8-3 を参照してください。

55 ページの「一般 ECI シナリオ」では、CICS_ExternalCall() の例を示し、特定の使用で必要な呼び出しおよび制御ブロック値について説明します。

リターンコード

ECI_NO_ERROR

CICS_ExternalCall() の呼び出しが正常に実行されました。

ECI_ERR_INVALID_DATA_LENGTH

eci_commarea の長さが無効です。次のいずれかが原因です。

eci_commarea の長さ < 0

eci_commarea の長さ > 32500

eci_commarea の長さ = 0 で、

eci_commarea が NULL ポインタでない

eci_commarea の長さ > 0 で、

eci_commarea が NULL ポインタである

ECI_ERR_INVALID_EXTEND_MODE

eci_extend_mode の値が無効であるか、eci_call_type の値に対して適切ではありません。

ECI_ERR_NO_CICS

Sun MTP Client が使用不可か、領域が使用できません。

ECI_ERR_CICS_DIED

拡張処理単位を処理している Sun MTP 領域を使用できません。このエラーが発生すると、リソースの変更がコミットされたのか取り消されたのかを確認できません。このイベントのログをとって、関連するリソースの手動回復に利用します。

ECI_ERR_NO_REPLY

応答請求によって応答が求められましたが、応答できるものがありません。

ECI_ERR_RESPONSE_TIMEOUT

処理を完了する前に、要求がタイムアウトになりました。このエラーが発生すると、リソースの変更がコミットできたのか取り消されたのかを確認できません。このイベントのログをとって、関連するリソースの手動回復に利用します。

ECI_ERR_TRANSACTION_ABEND

この呼び出しが参照する要求によって、Sun MTP トランザクションが不正終了しました。不正終了コードが、ECI_PARMS 構造体の eci_abend_code フィールドに戻されます。

ECI_ERR_LUW_TOKEN

eci_luw_token フィールドに無効な値が含まれます。

ECI_ERR_SYSTEM_ERROR

内部エラーです。

ECI_ERR_NULL_WIN_HANDLE

eci_window_handle フィールドの指定が、既存のウィンドウを参照しません。

ECI_ERR_NULL_SEM_HANDLE

eci_sem_handle フィールドの指定が NULL です。

ECI_ERR_NULL_MESSAGE_ID

eci_message_id フィールドが 0 です。

ECI_ERR_INVALID_CALL_TYPE

eci_call_type フィールドに無効な値が含まれています。

ECI_ERR_ALREADY_ACTIVE

処理単位に対する既存の要求を処理中に、処理単位を続行しようとしました。

ECI_ERR_RESOURCE_SHORTAGE

要求の実行には、システムリソースが不十分でした。

ECI_ERR_NO_SESSIONS

Sun MTP Client が並行処理単位の最大許容数に達しているときに、アプリケーションが新しい処理単位を作成しようとしました。

ECI_ERR_INVALID_DATA_AREA

ECI_PARMS 構造体へのポインタまたは eci_commarea ポインタに、想定されていない NULL が指定されています。

ECI_ERR_INVALID_VERSION

eci_version フィールドが、ECI_UNIKIX_VERSION_1、ECI_UNIKIX_VERSION_1A、ECI_VERSION_1、ECI_VERSION_1A のいずれでもありません。

ECI_ERR_UNKNOWN_SERVER

eci_system_name フィールドに不明なシステムが指定されています。

ECI_ERR_CALL_FROM_CALLBACK

CICS_ExternalCall() の呼び出しが、コールバックルーチンから実行されました。この呼び出しは許可されません。

ECI_ERR_INVALID_TRANSID

eci_transid の値が、以前の同じ処理単位の呼び出しで指定した値と異なります。

ECI_ERR_SECURITY_ERROR

指定された eci_userid/eci_password の組み合わせが無効でした。

ECI_ERR_MAX_SYSTEMS

Sun MTP Client が並行して接続できるシステムの最大数に達しています。

ECI_ERR_ROLLEDBACK

処理単位をコミットしようとしたときに、Sun MTP 領域が変更をコミットできずロールバックされました。

CICS_EciListSystems()

CICS_EciListSystems() 関数によって、プログラマは、ECI 要求に指示できる遠隔システムのリストを取得できます。戻されたリストにあるシステムは、KIXCLI.INI ファイルで定義されたシステムです。10 ページの「KIXCLI.INI ファイルへのシステムの追加」を参照してください。システムがリストに表示されていても、遠隔システムへの接続が確立されているとは限りません。システムが KIXCLI.INI ファイルに定義されていることを示しているだけです。

この関数から正常に復帰すると、List パラメータが指す配列にシステムの詳細が含まれます。Systems パラメータが更新されて、使用可能なシステム数が反映されます。

アプリケーションは、この関数によって戻されるシステムを有効な Systems パラメータとして CICS_ExternalCall() 関数に指定できます。Systems パラメータに指定できる有効な値は、この呼び出しの戻り値か、空白 8 文字で構成される特別な値だけです。

形式

```
cics_sshort_t CICS_EciListSystems(cics_char_t *NameSpace,
cics_ushort_t *Systems,
CICS_EciSystem_t *List);
```

各部の説明は次のとおりです。

NameSpace	NULL ポインタに設定する必要があります。これは無視されます。
Systems	数へのポインタです。関数の入口では、List パラメータが指す記憶領域に配置可能な CICS_EciSystem_t 構造体の数が含まれます。復帰時には、要求数にかかわらず、存在するシステム数が含まれます。
リスト	定義されたシステムの詳細を含む記憶領域に対するポインタです。Systems 領域にある情報だけを含みます。

次の表に、CICS_EciSystem_t 構造体のフィールドを示します。

表 8-4 CICS_EciSystem_t 構造体のフィールド

フィールド	型	説明
SystemName	char[9]	KIXCLI.INI ファイルから取得された遠隔システムの名前です。文字列は NULL で終わります。遠隔要求を実行するために、CICS_ExternalCall() 関数でも使用されます。
Description	char[61]	KIXCLI.INI ファイルから取得された、システムを説明する文字列です。

リターンコード

ECI_NO_ERROR

CICS_EciListSystems() の呼び出しが正常に実行されました。

ECI_ERR_INVALID_DATA_LENGTH

Systems の値が無効です。たとえば、List パラメータの記憶領域の量が 32,767 バイトを超えています。

ECI_ERR_NO_CICS

Sun MTP Client を利用できません。

ECI_ERR_SYSTEM_ERROR

内部エラーです。

ECI_ERR_INVALID_DATA_AREA

List パラメータが NULL で、Systems パラメータが 0 以外です。

ECI_ERR_CALL_FROM_CALLBACK

CICS_EciListSystems() の呼び出しが、コールバックルーチンから実行されました。この呼び出しは許可されません。

ECI_ERR_NO_SYSTEMS

KIXCLI.INI ファイルでシステムが定義されていません。

ECI_ERR_MORE_SYSTEMS

KIXCLI.INI ファイルのシステム数が、List パラメータで要求された数を超えています。この場合、List パラメータの値が存在するシステム数です。

KixCli_QueryFD()

コールバックメカニズムは、UNIX では動作が異なります。Sun MTP Client は、名前付きパイプを使用してアプリケーションと通信します。Sun MTP Client はシングルスレッド化されているので、アプリケーションが ECI インタフェースを呼び出す場合、データはパイプからのみ読み取ることができます。コールバックを実行できるのは、この場合のみです。そのため、イベント通知コールバックは、他の ECI 関数の処理中にのみ実行されます。この制限によって、効果的に動作するアプリケーションを設計するときに問題が発生する場合があります。

この処理を簡単にするために、`KixCli_QueryFD()` 関数を使用します。これは、通信で使用するパイプの標準 UNIX ファイル記述子 (FD) を戻します。アプリケーションは、`select()` 関数呼び出しで、ECI が処理する情報を待機します。アプリケーションは、データが設定されたことが伝えられると、ECI を呼び出してコールバックを処理できます。

サンプルアプリケーションの `ECIEX2` では、ECI 要求を並行して実行しながら curses 入出力処理を可能にする、この呼び出し例を示しています。

形式

```
cics_sshort_t KixCli_QueryFD( int *pFD );
```

各部の説明は次のとおりです。

`pFD`

この呼び出しによって戻される FD へのポインタです。

リターンコード

`ECI_NO_ERROR`

`KixCli_QueryFD()` の呼び出しが正常に実行されました。

`ECI_ERR_FROM_CALLBACK`

コールバックの処理中に、`KixCli_QueryFD()` の呼び出しが実行されました。

`ECI_ERR_NO_CICS`

Sun MTP Client と接続できませんでした。

一般 ECI シナリオ

この節では、Sun MTP Client に付属のサンプルアプリケーションについて説明し、ECI を使用するためのいくつかの方法を示します。ご使用のプラットフォームに関連する部分だけを参照してもかまいません。

ECI は、処理単位を作成および使用するためのさまざまなメカニズムを提供します。アプリケーションを設計する際は、応答通知に対する要件を決定する必要があります。

注 – 特定のプラットフォームだけに適用される方法もあるので、最初に 43 ページの「アプリケーション設計」のガイドラインを参照してください。

これらのシナリオのうち最も要件に一致するものを判断する場合、Sun MTP 領域上で実行するプログラムを次のように考慮します。

- 実行するのは単一のプログラムか
- プログラムが作業のコミットまたは取り消しを制御する必要はあるか
- 論理処理単位を形成するために多くのプログラムを実行する必要があるか

これらの質問に答えることによって、適切なプログラミングモデルを選択できます。

単発の DPL の実行

ECI から使用できる最も頻繁に必要とされる機能は、Sun MTP 領域に対する単一の DPL で構成される処理単位の実行です。これは通常、単一領域を指す、更新の 1 セットおよびクエリーの 1 セットに使用されます。多くのアプリケーションで、この機能が必要とされます。

メッセージ通知を使用する単発の非同期 DPL の実行

注 – これは、Windows プラットフォーム上でだけ使用できます。

非同期 DPL の実行は、次の 3 つのアクションで構成されます。

1. アプリケーションは、最初に `CICS_ExternalCall()` を呼び出して作業を実行するように伝え、アプリケーションプログラムへの応答の通知方法を知らせる必要があります。

2. アプリケーションは、要求が完了したという通知を受信するまで待機する必要があります。

3. アプリケーションは、CICS_ExternalCall() を再度呼び出して、要求結果を取得するために応答請求呼び出しを実行する必要があります。

この処理の最初の部分で、ECI に要求を実行するように指示します。この処理は、表 8-5 に示す値を設定した ECI_PARMS ブロックと、CICS_ExternalCall() 関数を使用して実行します。他のすべてのパラメータは NULL に設定します。

表 8-5 メッセージ通知を使用する単発の非同期 DPL の ECI_PARMS 値

フィールド	値
eci_call_type	ECI_ASYNC_NOTIFY_MSG
eci_program_name	Sun MTP 領域の PPT で指定された、実行するプログラム名です。
eci_userid	プログラムを実行するユーザー ID です。
eci_password	eci_userid に対するパスワードです。
eci_commarea	Sun MTP 領域に渡すデータ用にアプリケーション内で割り当てた空間へのポインタです。
eci_commarea_length	eci_commarea フィールドが指すデータブロックの長さです。
eci_extend_mode	ECI_NO_EXTEND
eci_message_qualifier	通知メッセージの受信時に、アプリケーションが通知に対応する処理単位を確認できるように指定する、ユーザー定義の値です。
eci_version	UNIKIX_ECI_VERSION_1
eci_system_name	要求の送信先のシステム名です。この名前は、CICS_EciListSystems() 呼び出しから戻された名前の 1 つか、すべて NULL に設定します。NULL に設定した場合は、KIXCLI.INI の定義に基づいて、デフォルトのシステムが使用されます。
eci_window_handle	要求の完了時に、通知されたメッセージを受信するウィンドウのウィンドウハンドルです。
eci_message_id	要求の完了時に、eci_window_handle ウィンドウに送信されるメッセージ ID です。

呼び出しが完了すると、Sun MTP Client は ECI_PARMS 構造体の eci_luw_token フィールドを設定します。このフィールドには ECI Client LUW トークンが含まれ、このタイプの要求には利用できません。

表 8-5 に示した eci_call_type 値は、eci_message_id 識別子を含むメッセージが、eci_window_handle フィールドで指定されたウィンドウに送信されることを意味します。

形式:

`wParam` 非同期呼び出しのリターンコードです。

`lParam` 下位 16 ビットには、非同期呼び出しで指定された `eci_message_qualifier` が含まれています。

このメッセージを受信すると、アプリケーションは ECI 要求を実行して DPL の結果を取得します。これは、`CICS_ExternalCall()` の別の呼び出しによって実行されますが、ここでは表 8-6 に示す値を含むパラメータブロックを指定します。他のフィールドはすべて NULL に設定します。

表 8-6 特定の応答を取得するための `ECI_PARMS` 値

フィールド	値
`eci_call_type`	`ECI_GET_SPECIFIC_REPLY`
`eci_commarea`	Sun MTP 領域が戻すデータ用に、アプリケーション内に割り当てられる空間へのポインタです。このデータは、DPL に送信される元のデータと同じ長さである必要があります。
`eci_commarea_length`	`eci_commarea` フィールドが指すデータブロックの長さです。
`eci_message_qualifier`	元の DPL 呼び出しで指定されるメッセージ修飾子です。
`eci_version`	`ECI_UNIKIX_VERSION_1`

この呼び出しが完了したあと、`eci_commarea` フィールドで指定された commarea には、戻されたデータが含まれます。Sun MTP 領域では、処理単位が完了して変更がコミットされます。

この呼び出しは、次のフィールドを変更することもあります。

`eci_abend_code` この処理単位を実行しているトランザクションが不正終了すると、不正終了コードがこのフィールドに設定されます。

この応答を受信すると、処理単位は完了します。

セマフォー通知を使用する単発の非同期 DPL の実行

注 – これは、Windows プラットフォーム上でだけ使用できます。

非同期 DPL の実行は、次の 3 つのアクションで構成されます。

1. アプリケーションは、最初に `CICS_ExternalCall()` を呼び出して作業を実行するように伝え、アプリケーションプログラムへの応答の通知方法を知らせる必要があります。
2. アプリケーションは、要求が完了したという通知を受信するまで待機する必要があります。
3. アプリケーションは、`CICS_ExternalCall()` を再度呼び出して、要求結果を取得する必要があります。

この処理の最初の部分で、ECI に要求を実行するように指示します。この処理は、表 8-7 に示す値を設定した `ECI_PARMS` ブロックと、`CICS_ExternalCall()` 関数を使用して実行します。他のすべてのパラメータは NULL に設定します。

表 8-7 セマフォー通知を使用する単発の非同期 DPL の `ECI_PARMS` 値

フィールド	説明
`eci_call_type`	`ECI_ASYNC_NOTIFY_SEM`
`eci_program_name`	Sun MTP 領域の PPT で指定された、実行するプログラム名です。
`eci_userid`	プログラムを実行するユーザー ID です。
`eci_password`	`eci_userid` に対するパスワードです。
`eci_commarea`	Sun MTP 領域に渡すデータ用に、アプリケーション内で割り当てられた空間へのポインタです。
`eci_commarea_length`	`eci_commarea` フィールドが指すデータブロックの長さです。
`eci_extend_mode`	`ECI_NO_EXTEND`
`eci_message_qualifier`	通知メッセージの受信時に、アプリケーションが通知に対応する処理単位を確認できるように指定する、ユーザー定義の値です。
`eci_version`	`UNIKIX_ECI_VERSION_1`
`eci_system_name`	要求の送信先のシステム名です。この名前は、`CICS_EciListSystems()` 呼び出しから戻された名前の 1 つか、すべて NULL に設定します。NULL に設定した場合は、`KIXCLI.INI` の定義に基づいて、デフォルトのシステムが使用されます。
`eci_sem_handle`	要求が完了すると設定されるシステムイベントのハンドルです。

この呼び出しが完了すると、Sun MTP Client は ECI_PARMS 構造体の eci_luw_token フィールドを設定します。このフィールドには ECI Client LUW トークンが含まれます。これは、このタイプの要求には利用できません。

表 8-7 に指定された eci_call_type 値は、要求が完了すると、eci_sem_handle パラメータに指定されたイベントが設定されることを意味します。アプリケーションは、Windows 関数 WaitForMultipleObjects() および WaitForSingleObject() を使用して実行を待機します。

アプリケーションが完了の通知を受信すると、ECI 要求を実行して DPL の結果を取得します。これは、CICS_ExternalCall() の別の呼び出しによって実行されますが、ここでは表 8-6 に示す値を含むパラメータブロックを指定します。他のフィールドはすべて NULL に設定します。

この呼び出しが完了したあと、eci_commarea フィールドで指定された commarea には、戻されたデータが含まれます。Sun MTP 領域では、処理単位が完了して変更がコミットされます。

この呼び出しは、次のフィールドを変更することもあります。

eci_abend_code	この処理単位を実行しているトランザクションが不正終了すると、不正終了コードがこのフィールドに設定されます。

この応答を受信すると、処理単位は完了します。

コールバック通知を使用する単発の非同期 DPL の実行

非同期 DPL の実行は、次の 3 つのアクションで構成されます。

1. アプリケーションは、最初に CICS_ExternalCall() を呼び出して作業を実行するように伝え、アプリケーションプログラムへの応答の通知方法を知らせる必要があります。
2. アプリケーションは、要求が完了したという通知を受信するまで待機する必要があります。
3. アプリケーションは、CICS_ExternalCall() を再度呼び出して、要求結果を取得する必要があります。

コールバック通知に複雑さをもたらす、次の 2 つの制約があります。

- コールバック内では、ECI 呼び出しを実行できません。コールバック内では、応答が準備できたことと、それを取得できることをプログラムに伝えることができます。
- コールバックに関する状況および実際にコールバックが実行される条件は、プラットフォームによって異なります。

Windows の場合、コールバックは他のアプリケーションとは別のスレッドで呼び出されます。そのため、コールバックは、アプリケーションのスレッドとは別のスレッドで任意の時点で実行されます。したがって、コールバック内で実行されるすべての作業がスレッドに対して安全であることが重要です。

UNIX 環境の場合、Sun MTP Client は名前付きパイプを使用してアプリケーションと通信します。ECI API はシングルスレッド化されているので、アプリケーションが ECI インタフェースを呼び出す場合、データはパイプからのみ読み取ることができます。すべてのコールバックの実行は、この場合のみです。そのため、通知コールバックは、他の ECI 関数の処理中にのみ実行されます。

この制限によって、効果的に動作する Solaris 上のアプリケーションを設計するときに問題が発生する場合があります。この処理を簡単にするために、`KixCli_QueryFD()` 関数を使用します。これは、通信に使用するパイプの標準 UNIX ファイル記述子を戻します。アプリケーションは、`select()` 関数呼び出しで、ECI が処理する情報を待機します。データが存在することがアプリケーションに伝えられると、ECI インタフェースを呼び出して、必要なコールバックを処理することができます。サンプルアプリケーションでは、`curses` アプリケーションを ECI とともに正常に動作させる方法を示します。処理の最初で、ECI に要求を実行するように指示します。この処理は、表 8-8 に示す値を設定した `ECI_PARMS` ブロックと、`CICS_ExternalCall()` 関数を使用して実行します。他のすべてのパラメータは NULL に設定します。

表 8-8 コールバック通知を使用する単発の非同期 DPL の `ECI_PARMS` 値

フィールド	値
`eci_call_type`	`ECI_ASYNC`
`eci_program_name`	Sun MTP 領域の PPT で指定された、実行するプログラム名です。
`eci_userid`	プログラムを実行するユーザー ID です。
`eci_password`	`eci_userid` に対するパスワードです。
`eci_commarea`	Sun MTP 領域に渡すデータ用にアプリケーション内で割り当てた空間へのポインタです。
`eci_commarea_length`	`eci_commarea` フィールドが指すデータブロックの長さです。
`eci_extend_mode`	`ECI_NO_EXTEND`
`eci_message_qualifier`	通知メッセージの受信時に、アプリケーションが通知に対応する処理単位を確認できるように指定する、ユーザー定義の値です。
`eci_version`	`UNIKIX_ECI_VERSION_1`
`eci_system_name`	要求の送信先のシステム名です。この名前は、`CICS_EciListSystems()` 呼び出しから戻された名前の 1 つか、すべて NULL に設定します。NULL に設定した場合は、`KIXCLI.INI` の定義に基づいて、デフォルトのシステムが使用されます。
`eci_callback`	応答可能になったときに呼び出す関数へのポインタです。

この呼び出しが完了すると、Sun MTP Client は ECI_PARMS 構造体の eci_luw_token フィールドを設定します。このフィールドには ECI Client LUW トークンが含まれます。これは、このタイプの要求には利用できません。

表 8-8 の eci_call_type 値は、要求が完了すると、eci_callback パラメータに指定したコールバックを実行することを意味します。

アプリケーションが完了の通知を受信すると、ECI 要求を実行して DPL の結果を取得します。これは、CICS_ExternalCall() の別の呼び出しによって実行されますが、表 8-6 に示す値を含むパラメータブロックを指定します。他のフィールドはすべて NULL に設定します。

この呼び出しの完了後、eci_commarea フィールドで指定したメモリー位置に、戻されたデータが設定されます。Sun MTP 領域では、処理単位が完了して変更がコミットされます。

この呼び出しによって、次のフィールドが変更されることもあります。

eci_abend_code　この処理単位を実行しているトランザクションが不正終了すると、不正終了コードがこのフィールドに設定されます。

この応答を受信すると、処理単位は完了します。

単発の同期 DPL の実行

同期 DPL の処理は、単一のアクションで構成されます。アプリケーションが、作業の実行を通知するには、CICS_ExternalCall() を単一で呼び出す必要があります。CICS_ExternalCall() の呼び出しは、表 8-9 に示す値を設定した ECI_PARMS ブロックを使用して実行します。他のすべてのパラメータは NULL に設定します。

表 8-9　単発の同期 DPL に対する ECI_PARMS 値

フィールド	値
eci_call_type	ECI_SYNC
eci_program_name	Sun MTP 領域の PPT で指定された、実行するプログラム名です。
eci_userid	プログラムを実行するユーザー ID です。
eci_password	eci_userid に対するパスワードです。
eci_commarea	Sun MTP 領域に渡すデータ用に、アプリケーション内で割り当てられた空間へのポインタです。
eci_commarea_length	eci_commarea フィールドが指すデータブロックの長さです。

表 8-9　単発の同期 DPL に対する ECI_PARMS 値 (続き)

フィールド	値
eci_extend_mode	ECI_NO_EXTEND
eci_version	UNIKIX_ECI_VERSION_1
eci_system_name	要求の送信先のシステム名です。この名前は、CICS_EciListSystems() 呼び出しから戻された名前の 1 つか、すべて NULL に設定します。NULL に設定した場合は、KIXCLI.INI の定義に基づいて、デフォルトのシステムが使用されます。

この呼び出しの完了後、eci_commarea フィールドで指定したメモリー位置に、戻されたデータが設定されます。Sun MTP 領域では、処理単位が完了して変更がコミットされます。

この呼び出しは、次のフィールドを変更することもあります。

eci_abend_code　この処理単位を実行しているトランザクションが不正終了すると、不正終了コードがこのフィールドに設定されます。

この呼び出しが戻されると、処理単位は完了します。

複数パーツの処理単位の開始

複数パーツの処理単位を開始する方法は、単一のプログラムで構成される処理単位を開始する場合と同じです。非同期の場合、アプリケーションは、要求の発行、結果通知の待機、および結果の取得に対して同じステップを実行する必要があります。最も大きな違いは、結果取得後に処理単位は終了しないで、ユーザーの要件に合わせて延長できることです。

そのため、長時間実行する処理単位を開始するには、単発の DPL と同じ手順に従います。ただし、CICS_ExternalCall() の最初の呼び出しで、eci_extend_mode パラメータを ECI_NO_EXTEND ではなく ECI_EXTEND に設定します。CICS_ExternalCall() の呼び出し開始から戻ると、eci_luw_token にはそのあとの呼び出しで渡す必要がある値が含まれます。処理単位の完了を円滑にするために、この値を保存する必要があります。

応答を受信すると、処理単位を延長または完了できます。

注意 – この時点で、Sun MTP トランザクション処理プログラムは応答をブロックされます。したがって、他のトランザクションを実行できません。トランザクション処理プログラムを、必要以上に長い間この状態のままにしないでください。

長時間実行の処理単位の継続

長時間実行する処理単位を継続するには、62 ページの「複数パーツの処理単位の開始」に説明されている方法と同様に処理単位を開始する必要があります。処理単位を開始した CICS_ExternalCall() の呼び出しで戻された eci_luw_token を保存して、この処理単位の延長に使用する必要があります。

処理単位を続行するために必要な手順は、処理単位を開始する手順と同じです。プログラムを実行するには、CICS_ExternalCall() を呼び出す必要があります。同期または非同期で実行できます。処理単位の続行と開始の大きな違いは、ECI_PARMS 構造体の eci_luw_token フィールドに、元の開始プログラム呼び出しの戻り値を設定する必要があることです。直前の DPL 開始呼び出しの eci_extend_mode が ECI_NO_EXTEND に設定されている場合、応答が受信されたあとに処理単位が完了し、格納されていた eci_luw_token が無効になります。eci_extend_mode が ECI_EXTEND に設定されている場合、応答の受信後、その処理単位に適用するアクションの受け入れ準備が整います。

処理単位の明示的な同期点化

処理単位を同期点化するには、62 ページの「複数パーツの処理単位の開始」に説明されている方法で処理単位を開始する必要があります。処理単位を開始した CICS_ExternalCall() の呼び出しで戻された eci_luw_token を保存して、処理単位の延長に使用する必要があります。

処理単位を同期点化するために必要な手順は、処理単位を開始するための手順と同じです。CICS_ExternalCall() を呼び出して、処理単位を同期点化する必要があります。必要に応じて、応答が準備できるまでアプリケーションを待機し、そのあとで応答を取得します。大きな違いは、ECI_PARMS 構造体の eci_luw_token フィールドに、元のプログラム呼び出しの戻り値を設定する必要があることです。

処理の最初で、ECI に要求を実行するように指示します。この処理は、表 8-10 に示す値を設定した ECI_PARMS ブロックと、CICS_ExternalCall() 関数を使用して実行します。他のすべてのパラメータは NULL に設定します。

表 8-10 処理単位を同期点化するための ECI_PARMS 値

フィールド	値
eci_call_type	ECI_ASYNC_NOTIFY_MSG、ECI_ASYNC_NOTIFY_SEM、または ECI_SYNC
eci_commarea	NULL ポインタです。実行するプログラムはありません。
eci_commarea_length	0 です。実行するプログラムはありません。
eci_extend_mode	ECI_COMMIT
eci_luw_token	処理単位の最初の ECI 呼び出しで戻されるトークンです。

表 8-10　処理単位を同期点化するための ECI_PARMS 値 (続き)

フィールド	値
eci_message_qualifier	通知メッセージの受信時に、アプリケーションが通知に対応する処理単位を確認できるように指定する、ユーザー定義の値です。この値は、元の要求時の値と同じである必要はありません。
eci_version	ECI_UNIKIX_VERSION_1
eci_window_handle	要求の完了時に、通知されたメッセージを受信するウィンドウのウィンドウハンドルです (ECI_ASYNC_NOTIFY_MSG の場合)。
eci_message_id	要求の完了時に、eci_window_handle ウィンドウに送信されるメッセージ ID です (ECI_ASYNC_NOTIFY_MSG の場合)。
eci_sem_handle	要求が完了すると設定されるシステムイベントのハンドルです (ECI_ASYNC_NOTIFY_SEM の場合)。

この呼び出しが完了し応答を受信したあと、Sun MTP 領域では、必要に応じて処理単位を完了し変更をコミットします。

処理単位のロールバック

処理単位をロールバックするには、62 ページの「複数パーツの処理単位の開始」で説明されている方法と同様に、処理単位を開始する必要があります。処理単位を開始した CICS_ExternalCall() の呼び出しで戻された eci_luw_token を保存して、処理単位の延長に使用する必要があります。

処理単位をロールバックするために必要な基本的な手順は、処理単位を同期点化するための手順と同じです。処理単位をロールバックするには、CICS_ExternalCall() を呼び出す必要があります。アプリケーションは応答が準備されるまで待機し、応答を取得します。

処理の最初で、ECI に要求を実行するように指示します。この処理は、表 8-11 に示す値を設定した ECI_PARMS ブロックと、CICS_ExternalCall() 関数を使用して実行します。他のすべてのパラメータは NULL に設定します。

表 8-11　処理単位をロールバックするための ECI_PARMS 値

フィールド	値
eci_call_type	ECI_ASYNC_NOTIFY_MSG
eci_commarea	NULL ポインタです。実行するプログラムはありません。
eci_commarea_length	0 です。実行するプログラムはありません。
eci_extend_mode	ECI_BACKOUT
eci_luw_token	処理単位の最初の ECI 呼び出しで戻されるトークンです。

表 8-11　処理単位をロールバックするための ECI_PARMS 値 (続き)

フィールド	値
eci_message_qualifier	通知メッセージの受信時に、アプリケーションが通知に対応する処理単位を確認できるように指定する、ユーザー定義の値です。この値は、元の要求時の値と同じである必要はありません。
eci_version	ECI_UNIKIX_VERSION_1
eci_window_handle	要求の完了時に、通知されたメッセージを受信するウィンドウのウィンドウハンドルです (ECI_ASYNC_NOTIFY_MSG の場合)。
eci_message_id	要求の完了時に、eci_window_handle ウィンドウに送信されるメッセージ ID です (ECI_ASYNC_NOTIFY_MSG の場合)。
eci_sem_handle	要求が完了すると設定されるシステムイベントのハンドルです (ECI_ASYNC_NOTIFY_SEM の場合)。

この呼び出しが完了し応答を受信したあと、Sun MTP 領域では、必要に応じて処理単位を完了し変更をロールバックします。

遠隔システムへの接続の問い合わせ

ECI を使用すると、指定した Sun MTP 領域への接続の有無について、Sun MTP Client に呼び出しを行って問い合わせることができます。これを実行するには、アプリケーションは 表 8-12 に示されているパラメータを設定して、CICS_ExternalCall() を呼び出す必要があります。他のすべてのパラメータは NULL に設定します。

表 8-12　ECI_STATE_ASYNC 呼び出しの ECI_PARMS 値

フィールド	値
eci_call_type	ECI_STATE_ASYNC_MSG
eci_commarea	ECI_STATUS 構造体用に、アプリケーション内に割り当てられた空間へのポインタです。
eci_commarea_length	eci_commarea フィールドが指すデータブロックの長さ (ECI_STATUS のサイズ) です。
eci_extend_mode	ECI_STATE_IMMEDIATE
eci_message_qualifier	通知メッセージの受信時に、アプリケーションが通知に対応する処理単位を確認できるように指定する、ユーザー定義の値です。

表 8-12　ECI_STATE_ASYNC 呼び出しの ECI_PARMS 値 (続き)

フィールド	値
eci_version	ECI_UNIKIX_VERSION_1
eci_window_handle	要求の完了時に、通知されたメッセージを受信するウィンドウのウィンドウハンドルです。
eci_message_id	要求の完了時に、eci_window_handle ウィンドウに送信されるメッセージ ID です。

ECI が要求を完了したあと、通知メッセージが eci_window_handle ウィンドウに送信されます。このメッセージを受信すると、アプリケーションは表 8-13 の値を ECI_PARMS 構造体に設定し、CICS_ExternalCall() の呼び出しを使用して要求結果の送信を請求します。

表 8-13　STATE_ASYNC_MESSAGE 応答請求の ECI_PARMS 値

フィールド	値
eci_call_type	ECI_GET_SPECIFIC_REPLY
eci_commarea	ECI_STATUS 構造体用に、アプリケーション内に割り当てられた空間へのポインタです。
eci_commarea_length	eci_commarea フィールドが指すデータブロックの長さ (ECI_STATUS のサイズ) です。
eci_message_qualifier	元の呼び出しで指定され、通知機能の一部として戻されたメッセージ修飾子です。
eci_version	ECI_UNIKIX_VERSION_1

ECI_STATUS 構造体に値が設定されます。これらのフィールドの内容については、表 8-2 を参照してください。

この基本状態要求に加え、ECI は、システムの状態の特定の変化をアプリケーションに通知できます。たとえば、システムが作業できるようになったときに通知することができます。次のパラメータを設定して、CICS_ExternalCall() の呼び出しを実行します。

```
eci_call_type        ECI_STATE_ASYNC_MSG
eci_extend_mode      ECI_STATE_CHANGED
```

呼び出しの実行後、システムの状態が eci_commarea フィールドで指定した状態から変化すると、通知が送信されます。

コールバックの使用法

ECI を呼び出して、Windows メッセージの代わりにコールバックを生成できます。ECI_PARMS 構造体の eci_window_handle パラメータを使用して、Windows メッセージによる通知を生成する方法については、55 ページの「単発の DPL の実行」から 65 ページの「遠隔システムへの接続の問い合わせ」を参照してください。

Windows メッセージの代わりにコールバックを生成するには、eci_window_handle パラメータを NULL に設定し、eci_callback パラメータをコールバック関数へのポインタとして設定します。これを設定したあとで ECI 要求が完了すると、コールバック関数が呼び出されます。コールバック関数は、元の呼び出しで eci_message_qualifier フィールドに設定された単一の値を、パラメータとして受け取ります。

コールバック関数内では、アプリケーションは ECI 機能を呼び出すことはできません。呼び出した場合、リターンコード ECI_ERR_IN_CALLBACK が戻されます。この制限のため、コールバック関数は、Windows プログラムのメインセクションへの通知方法を備えておく必要があります。これは通常、Windows メッセージを送信することによって実現します。

Sun MTP ECI インタフェース

Sun MTP Client には、4 種類の ECI API があります。これらは、ECI パラメータブロックの eci_version フィールドを、次の 1 つに設定することで取得できます。

ECI_VERSION_1	IBM 互換バージョン ECI API Version 1。
ECI_VERSION_1A	IBM 互換バージョン ECI API Version 1A。
ECI_UNIKIX_VERSION_1	ECI_VERSION_1 と同じインタフェースですが、68 ページの「応答メッセージ形式」で説明されているように、応答メッセージ形式について拡張されています。
ECI_UNIKIX_VERSION_1A	ECI_VERSION_1A と同じインタフェースですが、68 ページの「応答メッセージ形式」で説明されているように、応答メッセージ形式について拡張されています。

応答メッセージ形式

アプリケーションが非同期 ECI 呼び出しを行ってメッセージによる通知を要求すると、メッセージは指定したウィンドウに次のように配信されます。

ECI_VERSION_1 および ECI_VERSION_1A

wParam	非同期呼び出しのリターンコードです。
lParam	非同期呼び出しの 4 文字の不正終了コードです。不正終了が発生しなかった場合は、空白 4 文字になります。

ECI_UNIKIX_VERSION_1 および ECI_UNIKIX_VERSION_1A

wParam	非同期呼び出しのリターンコードです。
lParam	下位 16 ビットに、非同期呼び出しで指定した eci_message_qualifier が含まれます。

メッセージ修飾子は通常、不正終了コードよりも重要な情報です。メッセージ修飾子を指定して ECI_GET_SPECIFIC_REPLY を実行すると、不正終了コードを取得することもできます。

第9章

外部表示インタフェース (EPI)

Sun MTP の外部表示インタフェース (EPI) は、CICS 以外のアプリケーションプログラムを、1 つ以上の標準 3270 端末として Sun MTP に認識させる API です。EPI アプリケーションは、標準の 3270 端末と同様に Sun MTP と通信し、次の動作ができます。

- 端末へのログオン
- トランザクションの開始
- 標準 3270 データストリームのトランザクション間の送受信

EPI アプリケーションは、受信する 3270 データを表示する役割を持っています。3270 端末をエミュレートするか、ユーザーに適した他の方法によって、データをデバイスに表示します。

この章では、EPI の C プログラム関数仕様および必要なデータ構造体について説明します。この章の内容は、次のとおりです。

EPI 例

C プログラミング言語での EPI の使用例が、`$INSTROOT\examples` ディレクトリにあります。ここで、`$INSTROOT` は、マシンにインストールされている Sun MTP Client のディレクトリ名を示します。このディレクトリには、使用方法および必要な Sun MTP COBOL ソースコードを説明する readme ファイルもあります。

EPI アプリケーションの開発

EPI は、すべてのプログラムから呼び出すことができる、ライブラリに収められた一連の関数です。これらのルーチンは、C 言語のインタフェースです。

- Windows の場合
  - ECI/EPI コードは、`CCLAPI32.DLL` 内にあります。
  - アプリケーションを `CCLWIN32.LIB` にリンクします。
- UNIX の場合
  - ECI/EPI コードは、`libcclapi.so` にあります。
  - アプリケーションを `libcclapi.so` にリンクします。

このマニュアルで説明する定義およびプロトタイプを取得するために、すべてのアプリケーションで `cics_epi.h` を組み込む必要があります。

EPI の開始および終了

EPI 関数の `CICS_EpiInitialize()` および `CICS_EpiTerminate()` は、それぞれ EPI を開始および終了します。

`CICS_EpiInitialize()` 関数は、EPI インタフェースを開始するためにタスクごとに 1 度呼び出す必要があります。初期化関数が完了するまで、他のすべての EPI 呼び出しは無効です。`CICS_EpiInitialize()` 呼び出しが、正常なリターンコードで戻ると、初期化は完了です。

EPI を使用した処理が完了したとき、そのタスクが終了する直前に `CICS_EpiTerminate()` 関数を呼び出す必要があります。この終了関数によって、正常に EPI を終了します。

今後の EPI バージョンとの互換性は、`CICS_EpiInitialize()` 関数によって維持されます。これには、アプリケーションをプログラミングした際の EPI バージョンを定義するパラメータが必要です。

EPI 端末の追加および削除

アプリケーションは、初期化を完了したあとに 1 つ以上の EPI 端末をインストールできます。これは、Sun MTP に 3270 端末として認識されます。アプリケーションは、各端末を追加するごとに、`CICS_EpiAddTerminal()` 関数を 1 度呼び出す必要があります。この関数は、アプリケーション内で端末を一意に特定するための値を戻します。

この端末を使用して EPI 関数を適用するために、アプリケーションから EPI 関数にこの TermIndex を渡します。

アプリケーションで端末を必要としなくなった場合、CICS_EpiDelTerminal() 関数を呼び出して削除します。削除が正常に完了すると、TermIndex 値は解放され、EPI によって再使用されます。ただし、アプリケーションが CICS_EPI_EVENT_END_TERM イベントを受信するまで削除は完了しません。

端末がトランザクションを実行している場合は、削除関数でエラーが発生します。この場合、関数はエラーコードを戻します。

端末ごとに情報を追跡するために、アプリケーションは単純な配列索引付けスキーマを使用できます。TermIndex 値を使用して、対応する端末情報を保存します。

トランザクションの開始

アプリケーションは、インストールされた端末に対してトランザクションを開始できます。Sun MTPからは、端末ユーザーが画面でトランザクションを入力し、AID キーを押したように、トランザクションが見えます。

アプリケーションは、CICS_EpiStartTran() 関数を呼び出してトランザクションを開始します。CICS_EpiStartTran() パラメータは、トランザクション名およびそのトランザクションと関連付けられた最初の 3270 データです。

トランザクション名が指定されていない場合、EPI は AID シーケンスのあとの 3270 データの先頭最大 4 文字から必要なトランザクション名を特定します。これによって、真の 3270 端末エミュレーションを実現する EPI アプリケーションが簡単になります。最初の 3270 データは、通常は空ではありません。端末からの入力を確定する 3270 キー用の AID バイトが少なくとも含まれています。

3270 データストリームの形式の詳細は、『IBM 3270 Information Display System Data Stream Programmers Reference』を参照してください。

イベントの処理

多様なイベントが、追加された端末に影響を与えます。このイベントは、端末ユーザーの操作ではなく、遠隔システム内のアクションの結果です。イベントが発生すると、EPI アプリケーションは EPI から通知を受けます。

イベント通知を受信すると、EPI アプリケーションは、CICS_EpiGetEvent() 関数を呼び出して、処理対象のキューから 1 つのイベントを取り出します。この関数は、発生したイベントとイベントに関連するデータの情報を含むデータ構造体 CICS_EpiEventData_t を戻します。さらに、キューに待機中のイベントが存在するかどうかも示します。CICS_EpiEventData_t データ構造体の情報およびイベントの詳細は、82 ページの「EPI イベント」を参照してください。

この端末に対するイベントがさらに存在する場合、アプリケーションはイベントキューが空になるまで CICS_EpiGetEvent() を呼び出し続けます。この手順に従わない場合、端末に対する以後のイベントをアプリケーションに通知できなくなります。

アプリケーションは、できるだけすみやかにイベントを処理する必要があります。この同期化によって、EPI の状態とアプリケーションの状態が一致しないという状況を防止します。この不整合が発生すると、アプリケーションが EPI 関数を実行しようとした場合に、予期しないエラーのリターンコードを受信します。

イベント通知は、サポートしているプラットフォームによって異なります。この節では、これらの違いについて説明します。

Windows のイベント通知

Windows では、コールバックメカニズムを介してイベント通知が実行されます。特定の端末に対するイベントが存在する場合、CICS_EpiAddTerminal() 呼び出しで指定されたコールバックルーチンが開始されます。このコールバックは、EPI が所有する個別のスレッド上にあります。このスレッドからは、EPI 作業を実行しません。通常、コールバックルーチンは、何らかの処理が可能であるということをメッセージまたはイベントを使用してメインコードに通知します。

Solaris のイベント通知

Solaris でのコールバックメカニズムは異なります。Sun MTP Client は、名前付きパイプを使用してアプリケーションと通信します。Sun MTP Client はシングルスレッド化されているので、アプリケーションが ECI/EPI インタフェースを呼び出す場合、データはパイプからのみ読み取ることができます。コールバックを実行できるのは、この場合のみです。そのため、イベント通知コールバックは、他の ECI/EPI 関数の処理中にのみ実行されます。この制限によって、効果的に動作するアプリケーションを設計するときに問題が発生する場合があります。

この処理を簡単にするために、KixCli_QueryFD() 関数を使用します。これは、通信で使用するパイプの標準 UNIX ファイル記述子 (FD) を戻します。アプリケーションは、select() 関数呼び出しを使用して、処理する ECI/EPI の情報が設定されるまで待機します。アプリケーションは、データが設定されたことが伝えられると、ECI/EPI を呼び出してコールバックを処理できます。

サンプルアプリケーション EPIEX2 は、ユーザー入力と同期を取りつつ、複数の並行処理単位を実行する curses アプリケーションに、このメカニズムをどのように使用するかを示しています。

データの送受信

トランザクションがデータを端末に送信すると、EPI は、CICS_EPI_EVENT_SEND または CICS_EPI_EVENT_CONVERSE イベントを生成します。
CICS_EPI_EVENT_SEND イベントには、端末からの応答は必要ありません。
CICS_EPI_EVENT_CONVERSE イベントには、端末からの応答が必要です。

EPI アプリケーションは、CICS_EpiReply() 関数を呼び出すことで応答し、応答データを提供します。この関数は、CICS_EPI_EVENT_CONVERSE イベントに対する応答だけに使用します。他のイベントに対する応答でこの関数を呼び出すと、エラーが戻されます。

3270 データストリームの形式の詳細は、『IBM 3270 Information Display System Data Stream Programmers Reference』を参照してください。

EPI の定数およびデータ構造体

EPI には、EPI を使用するための定数およびデータ構造体を定義するファイルが用意されています。この節では、定数、標準データ型、およびデータ構造体について説明します。

定数

EPI は、EPI アプリケーションで使用する次の定数を定義します (C の #define 文)。

CICS_EPI_SYSTEM_MAX	8
CICS_EPI_DESCRIPTION_MAX	60
CICS_EPI_NETNAME_MAX	8
CICS_EPI_TRANSID_MAX	4
CICS_EPI_ABEND_MAX	4
CICS_EPI_DEVTYPE_MAX	16
CICS_EPI_ERROR_MAX	60
CICS_EPI_TERM_INDEX_NONE	0xFFFF

標準データ型

EPI は、EPI アプリケーションで使用する次のデータ型を定義します。

cics_char_t	文字型
cics_sbyte_t	符合付き 8 ビット整数
cics_sshort_t	符合付き 16 ビット整数
cics_slong_t	符合付き 32 ビット整数
cics_ubyte_t	符合なし 8 ビット整数
cics_ushort_t	符合なし 16 ビット整数
cics_ulong_t	符合なし 32 ビット整数
cics_ptr_t	一般ポインタ型
cics_shandle_t	一般 16 ビットハンドル
cics_lhandle_t	一般 32 ビットハンドル

データ構造体

EPI は、EPI アプリケーションで使用する次のデータ構造体を定義します。

75 ページの「CICS_EpiSystem_t」
76 ページの「CICS_EpiDetails_t」
77 ページの「CICS_EpiEventData_t」
78 ページの「CICS_EpiSysError_t」
79 ページの「CICS_EpiNotify_t」
79 ページの「CICS_EpiEvent_t」
80 ページの「CICS_EpiEnd_t」
80 ページの「CICS_EpiATIState_t」
81 ページの「CICS_EpiSenseCode_t」

CICS_EpiSystem_t

このデータ構造体には、遠隔 Sun MTP 領域名および説明が含まれます。CICS_EpiListSystems() 関数は、この構造体による情報を戻します。

C 定義

```
typedef struct
{
       cics_char_t   SystemName[CICS_EPI_SYSTEM_MAX+1];
       cics_char_t   Description[CICS_EPI_DESCRIPTION_MAX+1];
} CICS_EpiSystem_t;
```

各部の説明は次のとおりです。

SystemName　Sun MTP 領域名です。この文字列をパラメータとして CICS_EpiAddTerminal() 関数に渡し、端末が接続される領域を特定します。この文字列の残りは NULL バイトで埋められ、末尾に NULL バイトが付加されます。

Description　システムの簡単な説明です。この文字列の残りは NULL バイトで埋められ、末尾に NULL バイトが付加されます。

CICS_EpiDetails_t

この構造体へのポインタを CICS_EpiAddTerminal() 関数に渡します。インストールが完了した端末についての詳細が、この構造体に含まれます。

C 定義

```
typedef struct
{
       cics_char_t    SystemName[CICS_EPI_SYSTEM_MAX+1];
       cics_char_t    Description[CICS_EPI_DESCRIPTION_MAX+1];
       cics_char_t    NetName[CICS_EPI_NETNAME_MAX+1];
       cics_sshort_t  NumLines;
       cics_sshort_t  NumColumns;
       cics_ushort_t  MaxData;
       cics_sshort_t  ErrLastLine;
       cics_sshort_t  ErrIntensify;
       cics_sshort_t  ErrColor;
       cics_sshort_t  ErrHilight;
       cics_sshort_t  Hilight;
       cics_sshort_t  Color;
       cics_sshort_t  Printer;
} CICS_EpiDetails_t;
```

各部の説明は次のとおりです。

SystemName	Sun MTP 領域名です。この文字列をパラメータとして CICS_EpiAddTerminal() 関数に渡し、端末が接続される領域を特定します。この文字列の残りは NULL バイトで埋められ、末尾に NULL バイトが付加されます。
Description	システムの簡単な説明です。この文字列の残りは NULL バイトで埋められ、末尾に NULL バイトが付加されます。
NetName	端末をインストールする VTAM スタイルの NetName を指定する 8 文字の文字列です。この文字列の残りは NULL バイトで埋められ、末尾に NULL バイトが付加されます。
NumLines	端末がサポートする行数です。
NumColumns	端末がサポートする列数です。
MaxData	Sun MTP が端末に送信するデータの最大サイズです。
ErrLastLine	端末が最終行にエラーメッセージを表示する場合は、ゼロ以外の値です。
ErrIntensify	端末が強調エラーメッセージを表示する場合は、ゼロ以外の値です。
ErrColor	表示するエラーメッセージの色の値を定義する 3270 属性です。
ErrHilight	表示するエラーメッセージの強調表示値を定義する 3270 属性です。

Hilight	端末が拡張強調表示をサポートする場合は、ゼロ以外の値です。
Color	端末が色をサポートする場合は、ゼロ以外の値です。
Printer	端末がプリンタの場合は、ゼロ以外の値です。

CICS_EpiEventData_t

Data および Size フィールドを設定して、この構造体へのポインタを CICS_EpiGetEvent() 関数に渡します。関数が正常に終了すると、イベントに関する詳細情報がこの構造体に設定されます。すべてのフィールドが、すべてのイベントに対して有効であるとは限りません。無効なフィールドは、0 または NULL バイトに設定されます。

C 定義

```
typedef struct
{
       u_short_t          TermIndex;
       CICS_EpiEvent_t    Event;
       CICS_EpiEnd_t      EndReason;
       char               TransId[CICS_EPI_TRANSID_MAX+1];
       char               AbendCode[CICS_EPI_ABEND_MAX+1];
       u_byte_t*          Data;
       u_short_t          Size;
} CICS_EpiEventData_t;
```

各部の説明は次のとおりです。

TermIndex	イベントが発生した端末の索引番号です。
Event	イベントを指定します。イベントインジケータの説明は、82 ページの「EPI イベント」を参照してください。
EndReason	イベントが CICS_EPI_EVENT_END_TERM の場合の終了理由です。
TransId[]	トランザクション名を指定する 4 文字の文字列です。この文字列の残りはスペースで埋められ、NULL バイトで終了します。
AbendCode[]	ABEND コードを指定する 4 文字の文字列です。この文字列の残りはスペースで埋められ、NULL バイトで終了します。
Data	イベントに関連付けられた端末データストリーム用バッファーへのポインタです。
Size	Data でアドレス指定されたバッファーの最大サイズです。CICS_EpiGetEvent() 関数が終了すると、このフィールドにはデータの実際の長さが戻されます。

CICS_EpiSysError_t

この構造体へのポインタをパラメータとして CICS_EpiGetSysError() 関数に渡します。関数が正常に終了すると、システムエラーの原因についての詳細情報がこの構造体に設定されます。

C 定義

```
typedef struct
{
       u_long_t   Cause;
       u_long_t   Value;
       char       Msg[CICS_EPI_ERROR_MAX+1];
} CICS_EpiSysError_t;
```

各部の説明は次のとおりです。

Cause	最後のエラーの原因を示す動作環境固有の値です。
Value	最後のエラーの特質を示す動作環境固有の値です。
Msg	最後のエラーを説明するテキストメッセージです。この文字列の残りは NULL バイトで埋め込まれ、末尾に NULL バイトが追加されます。メッセージがない場合、文字列はすべて NULL です。

Cause フィールドの値は、次のとおりです。

CICS_EPI_SYSERROR_UNEXPECTED_DATASTREAM

Sun MTP Client が復号化できないデータストリームを Sun MTP 領域から受信しました。

CICS_EPI_SYSERROR_NO_MEMORY

Sun MTP Client が記憶域割り当て障害を検出しました。

CICS_EPI_SYSERROR_DUPLICATE_NETNAME

端末追加コマンドで指定した NetName がすでに使用中です。

CICS_EPI_SYSERROR_UNKNOWN_NETNAME

端末追加コマンドで指定した NetName が終端システムで不明です。

CICS_EPI_SYSERROR_UNKNOWN_DEVTYPE

端末追加コマンドで指定した DevType が終端システムで不明です。

CICS_EPI_SYSERROR_INVALID_TPNAME

終端システムが端末インストールトランザクションを認識しませんでした。

CICS_EPI_SYSERROR_UNEXPECTED_ERROR

内部機能で予想外のエラーが発生しました。

CICS_EPI_SYSERROR_UNKNOWN_SYSTEM

追加端末コマンドで指定された System が定義されていません。

CICS_EPI_SYSERROR_TERMINAL_OUT_OF_SERVICE

サービス休止に設定された定義のため、端末を終端システムにインストールできませんでした。

CICS_EPI_SYSERROR_SYSTEM_UNAVAILABLE

端末追加コマンドで指定した System は、ここでは使用できません。

CICS_EPI_SYSERROR_INTERNAL_LOGIC_ERROR

Sun MTP Client の内部論理エラーです。

CICS_EPI_SYSERROR_AUTOINSTALL_FAILED

Sun MTP 領域で、自動インストールプロセスの障害を発生しました。

CICS_EPI_SYSERROR_TERM_INSTALL_FAILED

Sun MTP 領域で、端末インストールプロセスの障害が発生しました。

CICS_EpiNotify_t

この構造体は、パラメータとして CICS_EpiAddTerminal() 関数に渡されます。これは、EPI イベントが未処理の場合に呼び出される関数を定義します。

C 定義

```
typedef void (CICS_EpiCallback_t)(cics_ushort_t Trm);
typedef CICS_EpiCallback_t *CICS_EpiNotify_t;
```

CICS_EpiEvent_t

この構造体は、EPI が生成するイベントコードを定義します。各イベントコードの詳細は、82 ページの「EPI イベント」を参照してください。

C 定義

```
typedef cics_ushort_t CICS_EpiEvent_t;
```

値

CICS_EPI_EVENT_SEND

CICS_EPI_EVENT_CONVERSE

CICS_EPI_EVENT_END_TRAN

CICS_EPI_EVENT_START_ATI

CICS_EPI_EVENT_END_TERM

CICS_EpiEnd_t

この構造体は、CICS_EPI_EVENT_END_TERM イベントに対する理由コードを定義します。各理由コードの詳細は、85 ページの「CICS_EPI_EVENT_END_TERM」を参照してください。

C 定義

```
typedef cics_ushort_t CICS_EpiEnd_t;
```

値

CICS_EPI_END_SIGNOFF

CICS_EPI_END_SHUTDOWN

CICS_EPI_END_OUTSERVICE

CICS_EPI_END_UNKNOWN

CICS_EPI_END_FAILED

CICS_EpiATIState_t

この構造体は、CICS_EpiATIState() 関数に渡す値を定義します。この関数も、完了時に次の値を戻します。

C 定義

```
typedef cics_ushort_t CICS_EpiATIState_t;
```

値

CICS_EPI_ATI_ON

EPI は、通常の方法で非同期トランザクション開始 (ATI) 要求を処理します。

CICS_EPI_ATI_HOLD

EPI は、CICS_EPI_ATI_ON が設定されるまで、すべての ATI 要求をキューに入れます。

CICS_EPI_ATI_QUERY

EPI の ATI 要求状態は変わりません。この値を使用して、現在の設定を検索します。

CICS_EpiSenseCode_t

この構造体は、CICS_EpiSenseCode() 関数に渡す値を定義します。CICS_EpiSenseCode() 関数には実行する処理がなく、互換性のためにのみ提供されています。

C 定義

```
typedef cics_ushort_t CICS_EpiSenseCode_t;
```

値

CICS_EPI_SENSE_OPCHECK

EPI は、3270 データストリームのエラーを検出します。

CICS_EPI_SENSE_REJECT

EPI は、無効な 3270 コマンドを検出します。

CICS_EpiWait_t

この構造体は、CICS_EpiGetEvent() 関数に渡す値を定義します。値に意味はありませんが、定義された値の 1 つである必要があります。

C 定義

```
typedef cics_ushort_t CICS_EpiWait_t;
```

値

CICS_EPI_WAIT

CICS_EPI_NOWAIT

EPI イベント

EPI アプリケーションには、すべての EPI イベントを収集して処理する役割があります。端末に対してイベントが発生すると、EPI は未処理のイベントが存在することをアプリケーションに通知します。

イベント通知を受信すると、EPI アプリケーションは、`CICS_EpiGetEvent()` を呼び出して `CICS_EpiEventData_t` 構造体を取得します。この構造体は発生したイベントを示し、イベントの詳細情報を含みます。

発生する EPI イベントは、次のとおりです。

CICS_EPI_EVENT_SEND

Sun MTP トランザクションが 3270 データを端末に送信しました。応答は想定されていません。ただし、そのデータを表示します。このイベントは、通常、`EXEC CICS SEND` コマンドタイプの結果です。

このイベントに対して、`CICS_EpiEventData_t` 構造体の次のフィールドが設定されます。

`Event`	`CICS_EPI_EVENT_SEND`
`Data`	このフィールドが指す、トランザクションが送信したデータを含むバッファーです。最初の 2 バイトは、3270 コマンドおよび WCC (書き込み制御文字) です。
`Size`	`Data` バッファーのデータ長です。

CICS_EPI_EVENT_CONVERSE

Sun MTP トランザクションが 3270 データを端末に送信します。応答が想定されています。このイベントは、EXEC CICS RECEIVE コマンドタイプまたは EXEC CICS CONVERSE コマンドタイプの結果です。これにより、アプリケーションは、CICS_EpiReply() を呼び出して、応答データを Sun MTP に戻します。

Sun MTP が想定する応答のタイプは、指定した Data フィールドの最初のバイトの 3270 コマンドによって異なります。3270 コマンドが Read Buffer コマンドである場合、Sun MTP はただちに応答を想定します。コマンドが Read Modified または Read Modified All の場合、Sun MTP は、ユーザーが次に AID キーを押したときの応答を想定します。関連付けられたデータの存在にかかわらず、このイベントは発生します。データが存在しない場合、Size フィールドは 0 に設定されます。

このイベントに対して、CICS_EpiEventData_t 構造体の次のフィールドが設定されます。

Event	CICS_EPI_EVENT_CONVERSE
Data	このフィールドが指す、トランザクションが送信したデータを含むバッファーです。
Size	Data バッファーのデータ長です。0 の場合、データが送信されなかったことを示しますが、応答は想定されます。

CICS_EPI_EVENT_END_TRAN

端末に対する Sun MTP トランザクションが終了しました。トランザクションが異常終了すると、イベントは AbendCode を示す場合があります。トランザクションが正常終了すると、AbendCode フィールドは空白 4 文字になります。トランザクションが擬似会話型の場合、TransId フィールドには、次のトランザクション名が設定されます。ユーザーが次に AID キーを押したとき (CICS_EpiStartTran() 関数を使用) に、アプリケーションはこのトランザクションを開始します。

他のトランザクションが実行されていない場合だけ、アプリケーションはトランザクションを開始できます。CICS_EPI_EVENT_END_TRAN イベントは、EPI が新しいトランザクションを受け入れ可能な状態であることを示します。

このイベントに対して、CICS_EpiEventData_t 構造体の次のフィールドが設定されます。

Event	CICS_EPI_EVENT_END_TRAN
TransId	トランザクションが擬似会話型の場合、このフィールドは次に開始するトランザクション名です。名前は 4 文字で、末尾に追加した NULL バイトで終了します。次のトランザクションが存在しない場合、フィールドはすべて NULL バイトです。
AbendCode	トランザクションが異常終了すると、このフィールドに 4 文字の AbendCode が戻されます。文字列の末尾には NULL バイトが追加されます。トランザクションが正常終了すると、空白 4 文字と NULL バイトが設定されます。

CICS_EPI_EVENT_START_ATI

非同期トランザクション開始 (ATI) のトランザクションが端末に対して開始されました。端末が他のトランザクションを実行中に ATI 要求を受信すると、その要求は現在のトランザクションが終了するまで EPI によって保持されます。現在のトランザクションが終了すると、端末に対する ATI トランザクションが開始され、EPI はこのイベントを生成してアプリケーションに通知します。

CICS_EPI_EVENT_START_ATI イベントは、ATI トランザクションが開始され、EPI は他のトランザクションを開始できない状態であることを示します。EPI が CICS_EPI_EVENT_START_ATI イベントを生成してアプリケーションがそれを受信する前に、アプリケーションが CICS_EpiStartTran() 関数を呼び出すと、開始トランザクション関数で障害が発生します。

このイベントに対して、CICS_EpiEventData_t 構造体の次のフィールドが設定されます。

Event	CICS_EPI_EVENT_START_ATI
TransId	開始された ATI トランザクション名です。名前は 4 文字で、末尾に NULL バイトが追加されます。

CICS_EPI_EVENT_END_TERM

このイベントは、端末が存在しないことを示します。このイベントが取り出されたあとは、その端末に対する TermIndex は無効になります。

このイベントに対して、CICS_EpiEventData_t 構造体の次のフィールドが設定されます。

Event

CICS_EPI_EVENT_END_TERM

EndReason

端末が終了した、または存在しない理由として次のどれかを含みます。

CICS_EPI_END_SIGNOFF

端末がサインオフしました。CICS_EpiDelTerminal() の呼び出しによって発生した可能性があります。

CICS_EPI_END_SHUTDOWN

Sun MTP が停止しています。

CICS_EPI_END_OUTSERVICE

端末がサービス休止に切り替えられています。

CICS_EPI_END_UNKNOWN

予想外のエラーが発生しました。

CICS_EPI_END_FAILED

端末の削除で障害が発生しました。

EPI 関数

C プログラムから呼び出せる EPI ライブラリ関数は、次のとおりです。

CICS_EpiInitialize()

CICS_EpiInitialize 関数は EPI を初期化します。プロセスごとにこの関数を呼び出す必要があります。この関数の前に他の EPI を呼び出すと無効になります。

形式

```
cics_sshort_t CICS_EpiInitialize(cics_ulong_t Version);
```

各部の説明は次のとおりです。

Version

アプリケーションをプログラミングした際の EPI ライブラリのバージョンです。これにより、今後の EPI ライブラリバージョンとの互換性を保つことができます。

このバージョンの EPI の場合、パラメータに定数 CICS_EPI_VERSION_101 を設定します。

リターンコード

CICS_EPI_NORMAL

正常に完了しました。

CICS_EPI_ERR_FAILED

EPI は、インタフェースを初期化できません。

CICS_EPI_ERR_VERSION

EPI は、要求されたバージョンをサポートできません。

CICS_EPI_ERR_IS_INIT

EPI は、このプロセスに対してすでに初期化されています。

CICS_EpiTerminate()

この関数は EPI を終了します。通常、この関数は各プロセスの終了直前に呼び出します。この関数よりあとで EPI 呼び出しは無効です。この関数の呼び出しでは、アプリケーションによって定義された EPI 端末は削除されません。アプリケーションは、インタフェースを終了する前に CICS_EpiDelTerminal() 呼び出しを実行する必要があります。

形式

```
cics_sshort_t CICS_EpiTerminate(void);
```

リターンコード

CICS_EPI_NORMAL

正常に完了しました。

CICS_EPI_ERR_NOT_INIT

初期化が完了していません。

CICS_EPI_ERR_FAILED

EPI を終了できません。

CICS_EpiListSystems()

この関数は、端末の接続が可能なシステム (Sun MTP 領域) のリストを取得します。

形式

```
cics_sshort_t CICS_EpiListSystems  (cics_char_t    *NameSpace,
cics_ushort_t                                     *Systems,
CICS_EpiSystem_t                                  *List);
```

各部の説明は次のとおりです。

Namespace	パラメータは無視されますが、NULL ポインタに設定する必要があります。
Systems	EPI が戻す CICS_EpiSystem_t 構造体の最大数を設定します。関数から戻ると、このフィールドにはシステムが検出した実際の数が設定されます。
リスト	CICS_EpiSystem_t データ構造体の配列です。関数は構造体のフィールドを設定して、アプリケーションに戻します。呼び出し側は配列の記憶領域を指定し、配列の最大サイズを示す Systems パラメータを設定します。

リターンコード

CICS_EPI_NORMAL

正常に完了しました。

CICS_EPI_ERR_NOT_INIT

初期化が完了していません。CICS_EpiInitialize() を呼び出してください。

CICS_EPI_ERR_FAILED

システムの候補が見つかりませんでした。

CICS_EPI_ERR_NO_SYSTEMS

EPI は、システムの候補を検索できません。Systems フィールドの戻り値が 0 です。

CICS_EPI_ERR_MORE_SYSTEMS

配列に十分な領域がないため、すべてのシステムの候補を格納できません。システムの実際の数は、Systems フィールドを参照してください。この値を使用して、システムの候補をすべて格納できる十分な記憶領域を割り当て、関数を再実行します。

CICS_EpiAddTerminal()

この関数は、新しい端末をインストールします。各端末での以降の EPI 呼び出しで使用する TermIndex 値を戻します。索引番号は、0 から始まる連続した整数です。アプリケーションは、索引番号と端末のマッピングを保持して、端末ごとの情報を処理します。

形式

```
cics_sshort_t CICS_EpiAddTerminal (cics_char_t       *NameSpace,
                                   cics_char_t       *System,
                                   cics_char_t       *NetName,
                                   cics_char_t       *DevType,
                                   CICS_EpiNotify_t   NotifyFn,
                                   CICS_EpiDetails_t *Details,
                                   cics_ushort_t     *TermIndex);
```

各部の説明は次のとおりです。

Namespace　パラメータは無視されますが、NULL ポインタに設定する必要があります。

システム　端末を接続するシステム名です。この名前は、CICS_EpiListSystems() 関数呼び出しで戻されたシステムリストの 1 つと一致する必要があります。指定する文字列の長さを 0 にすると、KIXCLI.INI 構成ファイルで指定しているデフォルトのシステムが使用されます。これは、NULL で終了する文字列へのポインタです。

NetName　Sun MTP TCT にある端末の VTAM スタイルのネット名です。文字列へのポインタです。文字列の残りは NULL バイトで埋められ、末尾に NULL バイトが追加されます。このパラメータが NULL ポインタに設定されている場合、デフォルトが使用されます。このパラメータは一意で、システムの端末データベースに登録されている端末定義のネット名の 1 つと一致する必要があります。NetName を指定しない場合、端末は自動インストールされます。

DevType　モデル端末を指定する文字列へのポインタです。この文字列の残りは NULL バイトで埋められ、末尾に NULL バイトが追加されます。NetName パラメータが NULL 以外の場合、このパラメータは無視されます。

このフィールドが NULL ポインタに設定されている場合、デフォルトのモデル IBM-3278-2 が使用されます。

端末モデル名はシステム固有で、Sun MTP に有効なモデルは次のとおりです。

IBM-3278-2、IBM-3278-2-E

IBM-3278-4、IBM-3278-4-E

IBM-3278-5、IBM-3278-5-E

IBM-3287

接尾辞 -E は、拡張画像属性の端末であることを示します。

NotifyFn　端末に対する EPI イベントが発生したとき、呼び出すルーチンのアドレスです。

Details　EPI によって、インストールされた端末の詳細情報が設定される構造体へのポインタです。

CICS_EPI_WIN_INSTALLED メッセージを受信すると、アプリケーションはこの記憶領域を安全に使用したり、破棄できます。

TermIndex　インストールされた端末の索引番号です。すべての EPI 関数で、この端末に固有の索引番号を使用します。生成される索引番号は、0 から始まる連続した整数です。

リターンコード

CICS_EPI_NORMAL

正常に完了しました。

CICS_EPI_ERR_NOT_INIT

初期化が完了していません。

CICS_EPI_ERR_FAILED

端末をインストールできません。

CICS_EPI_ERR_SYSTEM

System パラメータで指定された名前が存在しません。

CICS_EPI_ERR_MAX_TERMS

サポートされている EPI 端末の最大数に達しています。

非同期作業を処理中に CICS_EpiAddTerminal() で障害が発生すると、次のリターンコードが CICS_EpiGetSysError() によって戻されます。

CICS_EPI_ERR_SYSTEM

System パラメータで指定された名前が存在しません。

CICS_EPI_ERR_MAX_TERMS

サポートされている EPI 端末の最大数に達しています。

CICS_EpiDelTerminal()

この関数は、インストールされた EPI 端末を削除します。端末が自動インストールされた場合、この定義は削除されます。ただし、端末が現在トランザクションを処理中の場合、アプリケーションは端末を削除できません。端末がトランザクションを処理中に、アプリケーションがこの関数を呼び出すと、関数は機能しません。現在のトランザクションが終了すると、アプリケーションはこの関数を正常に呼び出せます。アプリケーションでは、正常なリターンコードおよび CICS_EPI_EVENT_END_TERM イベントを受信するまで、端末が完全に削除されたと判断しません。

形式

```
cics_sshort_t CICS_EpiDelTerminal(cics_ushort_t TermIndex);
```

各部の説明は次のとおりです。

TermIndex

削除する端末の索引番号です。

リターンコード

CICS_EPI_NORMAL

正常に完了しました。

CICS_EPI_ERR_NOT_INIT

初期化が完了していません。

CICS_EPI_ERR_FAILED

端末を削除できません。

CICS_EPI_ERR_BAD_INDEX

TermIndex パラメータは、有効な端末を指定していません。

CICS_EPI_ERR_TRAN_ACTIVE

端末に対するトランザクションを処理中です。

CICS_EpiStartTran()

この関数は、インストールされた端末のトランザクションを開始します。トランザクションが開始されると、CICS_EPI_EVENT_END_TRAN イベントが生成されるまで、EPI は他の開始要求を受け入れません。

アプリケーションは、CICS_EPI_ERR_ATI_ACTIVE の予期しないリターンコードを受信することがあります。これは、端末がトランザクションを処理中でないと、アプリケーションが認識している場合にも該当します。つまり、ATI 要求が端末に対して開始され、CICS_EPI_EVENT_START_ATI イベントが発生しましたが、アプリケーションはこのイベントを処理していない場合です。

形式

```
cics_sshort_t CICS_EpiStartTran (cics_ushort_t  TermIndex,
                                 cics_char_t    *TransId,
                                 cics_ubyte_t   *Data,
                                 cics_ushort_t  Size);
```

各部の説明は次のとおりです。

TermIndex	トランザクションを実行する端末の索引番号です。
TransId	端末に対して実行するトランザクションコードです。このフィールドは、文字列へのポインタです。文字列の残りは空白文字で埋められ、末尾に NULL バイトが追加されます。 フィールドが NULL 文字の場合、EPI はデータバッファーから正しいトランザクションコードを抽出しようとします。ユーザーデータの開始を検出するために、データの最初のバイトを調べ、3270 制御コードをスキップして抽出します。別の 3270 制御コードが検出されるかデータがなくなるまで、次に続く最大 4 文字をコピーします。結果は、4 文字の長さまで空白文字で埋め込まれます。 擬似会話型トランザクションの継続部分を開始するには、CICS_EPI_EVENT_END_TRAN イベントが発生したときに戻される TransId をこのフィールドに含む必要があります。
Data	トランザクションに関連付けられた最初の 3270 データバッファーへのポインタです。このパラメータは NULL ポインタにはできません。少なくとも端末読み込みを発生した 3270 AID キーを含む必要があります。
Size	Data バッファーのバイト数です。

リターンコード

CICS_EPI_NORMAL

正常に完了しました。

CICS_EPI_ERR_NOT_INIT

初期化が完了していません。

CICS_EPI_ERR_FAILED

トランザクションを開始できません。

CICS_EPI_ERR_BAD_INDEX

TermIndex パラメータは、有効な端末を指定していません。

CICS_EPI_ERR_TTI_ACTIVE

端末で TTI トランザクションがすでに動作中です。

CICS_EPI_ERR_ATI_ACTIVE

端末で ATI トランザクションが動作中です。

CICS_EPI_ERR_NO_DATA

初期データが指定されていません。

CICS_EpiReply()

この関数は、端末から Sun MTP トランザクションへデータを送信します。これは、CICS_EPI_EVENT_CONVERSE イベントに応答する場合にだけ使用します。

形式

```
cics_sshort_t CICS_EpiReply  (cics_ushort_t  TermIndex,
                              cics_ubyte_t   *Data,
                              cics_ushort_t  Size);
```

各部の説明は次のとおりです。

TermIndex	データを送信している端末の索引番号です。
Data	トランザクションに送信される 3270 データバッファーへのポインタです。このフィールドは、NULL ポインタにはできません。少なくとも端末読み込みを発生させる 3270 AID バイトを含む必要があります。
Size	Data バッファーのサイズです。

リターンコード

CICS_EPI_NORMAL

正常に完了しました。

CICS_EPI_ERR_NOT_INIT

初期化が完了していません。

CICS_EPI_ERR_FAILED

応答データを送信できません。

CICS_EPI_ERR_BAD_INDEX

TermIndex パラメータは、有効な端末を指定していません。

CICS_EPI_ERR_NO_CONVERSE

端末に対する未処理の CICS_EPI_EVENT_CONVERSE イベントはありません。

CICS_EPI_ERR_NO_DATA

応答データがありません。

CICS_EpiATIState()

この関数によって、アプリケーションは端末での ATI 要求の処理を照会および変更できます。ATI 要求が有効 (CICS_EPI_ATI_ON) で ATI 要求が発行されると、端末が使用可能になるとすぐに EPI は自動的に要求を開始します。ATI 要求が保持されている場合 (CICS_EPI_ATI_HOLD)、すべての ATI 要求がキューに入れられ、ATI 要求が有効になると開始されます。

EPI は、常に CICS_EPI_ATI_HOLD 状態から開始します。ATI 処理が可能な状態で、端末定義で ATI 要求が可能に定義されている場合、アプリケーションは ATI 要求の処理を変更できます。

形式

```
cics_sshort_t CICS_EpiATIState   (cics_ushort_t  TermIndex,
              CICS_EpiATIState_t  *ATIState);
```

各部の説明は次のとおりです。

TermIndex　端末の索引番号です。

ATIState　80 ページの「CICS_EpiATIState_t」で定義されている CICS_EPI_ATI_ON、CICS_EPI_ATI_HOLD、または CICS_EPI_ATI_QUERY のいずれかです。関数の入口では、このフィールドに必要な新しい ATI 状態を設定します。この関数の出口では、このフィールドに以前の状態が設定されます。

リターンコード

CICS_EPI_NORMAL

正常に完了しました。

CICS_EPI_ERR_NOT_INIT

初期化が完了していません。CICS_EpiInitialize() を呼び出してください。

CICS_EPI_ERR_FAILED

ATI 状態を設定または確認できません。

CICS_EPI_ERR_BAD_INDEX

TermIndex パラメータは、有効な端末を指定していません。

CICS_EPI_ATI_STATE

無効な ATIState が指定されました。

CICS_EpiSenseCode()

送信されたデータストリームにアプリケーションがエラーを検出したときに、この関数を呼び出します。

注 – この関数は、互換性のためだけにあり、無視されます。

形式

```
cics_sshort_t CICS_EpiSenseCode (cics_ushort_t         TermIndex,
                                 CICS_EpiSenseCode_t   SenseCode);
```

各部の説明は次のとおりです。

TermIndex	端末の索引番号です。
SenseCode	センスコード障害の理由です。81 ページの「CICS_EpiSenseCode_t」 で定義されている、CICS_EPI_ATI_OPCHECK または CICS_EPI_ATI_REJECT のいずれかです。

リターンコード

CICS_EPI_NORMAL

正常に完了しました。

CICS_EPI_ERR_NOT_INIT

初期化が完了していません。

CICS_EpiGetEvent()

この関数は、処理対象のイベントキューからイベントを取り出します。

形式

```
cics_sshort_t CICS_EpiGetEvent      (cics_ushort_t  TermIndex,
              CICS_EpiWait_t         Wait,
              CICS_EpiEventData_t    *Event);
```

各部の説明は次のとおりです。

TermIndex	イベントを取得する端末の索引番号です。定数 CICS_EPI_TERM_INDEX_NONE を設定すると、登録した端末に対して次のイベントが戻されることを指定することができます。この場合、アプリケーションは、戻された CICS_EpiEventData_t 構造体の TermIndex フィールドを調べて、該当する端末を判断します。
Wait	無視されますが、次のどちらかに設定する必要があります。 CICS_EPI_WAIT CICS_EPI_NOWAIT
Event	発生したイベントの詳細情報を含む構造体へのポインタです。 構造体の Data フィールドは、イベントに対する端末データストリームによって更新されるデータバッファーを指します。 Size フィールドは、データバッファーの最大サイズを示し、戻されたデータの実際の長さに更新されます。

リターンコード

CICS_EPI_NORMAL

正常に完了しました。

CICS_EPI_ERR_NOT_INIT

初期化が完了していません。

CICS_EPI_ERR_FAILED

次のイベントを取得できません。

CICS_EPI_ERR_BAD_INDEX

TermIndex パラメータは有効な端末を指定していません。

CICS_EPI_ERR_WAIT

Wait パラメータが無効です。

CICS_EPI_ERR_NO_EVENT

端末に対する未処理のイベントはありません。

CICS_EPI_ERR_MORE_DATA

Data バッファーに、端末のデータを格納する十分な領域がありません。データは切り捨てられます。

CICS_EPI_ERR_MORE_EVENTS

イベントは正常に取得されましたが、この端末に対する他のイベントが未処理です。

CICS_EpiGetSysError()

この関数は、最後に発生したエラーについての詳細なエラー情報を取得します。EPI コマンドで障害が発生して CICS_EPI_ERR_FAILED のリターンコードが戻ると、EPI によってエラー情報が保存されます。

EPI 関数がこのリターンコードを戻した場合、元の EPI 要求に関連した TermIndex を指定して、この関数を呼び出します。SysErr パラメータで戻される値は、EPI 関数からのリターンコードを詳細に説明します。この値は、オペレーティングシステムと環境に固有な値です。それぞれの環境のマニュアルを参照してください。

形式

```
cics_sshort_t CICS_EpiGetSysError(cics_ushort_t      TermIndex,
                                  CICS_EpiSysError_t *SysErr);
```

各部の説明は次のとおりです。

TermIndex	追加のエラー情報を取得する端末の索引番号です。定数 CICS_EPI_TERM_INDEX_NONE を指定して、CICS_EpiInitialize()、CICS_EpiTerminate()、CICS_EpiListSystems()、または CICS_EpiAddTerminal() 関数から詳細なエラー情報が必要であることを示すことができます。
SysErr	システムのエラー情報を含む CICS_EpiSysError_t 構造体へのポインタです。SysErr 構造体の Cause フィールドおよび Value フィールドの値を使用して、EPI 関数のリターンコードを修飾できます。 Msg フィールドが使用可能な場合、発生したエラーを説明する動作環境固有のテキストメッセージを戻します。

リターンコード

CICS_EPI_NORMAL

正常に完了しました。

CICS_EPI_ERR_NOT_INIT

初期化が完了していません。

CICS_EPI_ERR_BAD_INDEX

TermIndex パラメータは、有効な端末を指定していません。

CICS_EPI_ERR_FAILED

SysErr 情報を取得できません。

Other （**その他**）

最後のエラーを発生させたコマンドからのリターンコードです。元の呼び出しのセクションを参照して、これらの値の意味を判断します。これは、特に CICS_EpiAddTerminal() 関数に該当しています。

SysErr パラメータに戻された値が該当する場合、各 Cause フィールドの意味を以下に示します。

CICS_EPI_SYSERROR_UNEXPECTED_DATASTREAM

Sun MTP Client が、Sun MTP から復号化できないデータストリームを受信しました。

CICS_EPI_SYSERROR_NO_MEMORY

Sun MTP Client が、内部処理のためのメモリーを取得できませんでした。

CICS_EPI_SYSERROR_DUPLICATE_NETNAME

CICS_EpiAddTerminal() 呼び出しで指定された NetName は、すでに使用されています。

CICS_EPI_SYSERROR_UNKNOWN_NETNAME

CICS_EpiAddTerminal() 呼び出しで指定された NetName は、通信している Sun MTP 領域で不明です。

CICS_EPI_SYSERROR_UNKNOWN_DEVTYPE

CICS_EpiAddTerminal() 呼び出しで指定された DevType は、通信している Sun MTP 領域で不明です。

CICS_EPI_SYSERROR_INVALID_TPNAME

端末インストールトランザクション CTIN は、遠隔 Sun MTP 領域で使用できません。

CICS_EPI_SYSERROR_UNEXPECTED_ERROR

内部エラーです。

CICS_EPI_SYSERROR_UNKNOWN_SYSTEM

指定されたシステムは、Sun MTP Client で不明です。

CICS_EPI_SYSERROR_TERM_OUT_OF_SERVICE

指定された NetName に対応する端末は、サービス休止に設定されています。

CICS_EPI_SYSERROR_SYSTEM_UNAVAILABLE

現在、指定されたシステムにアクセスできません。

CICS_EPI_SYSERROR_INTERNAL_LOGIC_ERROR

内部エラーです。

CICS_EPI_SYSERROR_AUTOINSTALL_FAILED

端末自動インストールで、原因不明の障害が発生しました。

CICS_EPI_SYSERROR_TERM_INSTALL_FAILED

端末インストールで、原因不明の障害が発生しました。

CICS_EpiInquireSystem()

この関数は、端末についての制限情報を提供します。TermIndex が指定されている場合、この関数は端末が接続されるシステム名を戻します。

形式

```
cics_sshort_t CICS_EpiInquireSystem  (cics_ushort_t  TermIndex,
                                      cics_char_t    *System);
```

各部の説明は次のとおりです。

`TermIndex` システム情報を必要とする端末の索引番号です。

システム `TermIndex` が接続されているシステム名を含むバッファーへのポインタです。

リターンコード

`CICS_EPI_NORMAL`

正常に完了しました。

`CICS_EPI_ERR_NOT_INIT`

初期化が完了していません。`CICS_EpiInitialize()` を呼び出してください。

`CICS_EPI_ERR_BAD_INDEX`

`TermIndex` パラメータは、有効な端末を指定していません。

`CICS_EPI_ERR_FAILED`

システム情報を取得できません。

付録 A

KIXTERM.INI

この付録では、初期設定ファイル KIXTERM.INI について説明します。このファイルには、ECI/EPI クライアント端末エミュレータで使用する色およびキーボードの定義を記述します。ファイルには次の 4 つのセクションがあります。

ファイルのセクション	内容
[keys]	キーマッピングです。102 ページの「キーマッピング」を参照してください。
[system_colors]	RGB 値から記号名への色マッピングです。104 ページの「端末の標準色およびライト色の定義」 を参照してください。
[colors]	記号名から 3270 値への色マッピングです。105 ページの「色のマッピング」 を参照してください。
[general]	その他のすべての初期設定です。106 ページの「キーボードのリセット」 を参照してください。

また、初期設定ファイルには、101 ページの「ファイルコメントの識別」で説明している形式のコメントが含まれます。

ファイルコメントの識別

テキストの前にセミコロンを付けてコメントにします。たとえば、次のようになります。

```
; ---------------------------
; 3270 color designation
; ---------------------------
```

キーマッピング

KIXTERM.INI ファイルの [keys] セクションには、キーマッピングを指定します。

形式

3270 key = *System Key* [+ *Modifier key*]

各部の説明は次のとおりです。

3270 key	特定の 3270 操作を実行するキーとして、端末エミュレータによって定義されています。すべての 3270 キーには記号表記があります。詳細は、表 A-1を参照してください。
System Key	端末エミュレータによってサポートされている、PC キーボード上のキーの記号表記として定義されています。定義については、表 A-2 を参照してください。
Modifier key	キーの組み合わせによって、3270 キーを指定できます。たとえば、次のように、リセットキーを Alt R として定義します。 `reset = R + Alt` 定義については、表 A-3 を参照してください。

次の表に、3270 キーを示します。

表 A-1 KIXTERM.INI 3270 キー

記号名	3270 操作
backspace	非保護フィールドにあるカーソルの左側の文字を削除し、カーソルを 1 つ左に移動します。
backtab	フィールドが非保護で、カーソルが最初の文字位置にない場合に、カーソルを現在のフィールドの最初の文字位置に移動します。それ以外の場合は、カーソルを前の非保護フィールドに移動します。
clear	画面を消去して、カーソルをホームポジションに移動します。AID を送信します。
cursorleft	カーソルを 1 つ左側に移動します。
cursordown	カーソルを 1 行下に移動します。
cursorright	カーソルを 1 つ右側に移動します。
cursorup	カーソルを 1 行上に移動します。
delete	カーソルの下にある文字を削除します。
enter	入力 AID を送信します。
eraseeof	現在のカーソルの位置からフィールドの最後までのすべての文字を消去します。

表 A-1　KIXTERM.INI 3270 キー (続き)

記号名	3270 操作
eraseinput	すべての非保護フィールドを消去します。
insert	端末の挿入モードと上書きモードを切り替えます。
newline	カーソルを行の最初の非保護フィールドへ移動します。
pa1 から pa3	適切な PA キー AID を送信します。
pf1 から pf24	適切な PF キー AID を送信します。
printscreen	現在の表示をローカルプリンタに出力します。
reset	キーボードロックをリセットします。
tab	カーソルを次の非保護フィールドの最初の文字位置へ移動します。

次の表に、システムキーを示します。

表 A-2　KIXTERM.INI システムキー

記号名	システムキー
A から Z	A から Z のキー
0 から 9	0 から 9 のキー
backspace	Backspace キー
delete	Delete キー
down	下矢印
end	End キー
escape	Esc キー
f1 から f24	ファンクションキー
home	Home キー
insert	Insert キー
left	左矢印
rightctrl	右コントロールキー
newline	Newline (Enter) キー
numpad0 から numpad9	数字キーパッドの 0 から 9 のキー
numpad+	数字キーパッドの + キー
numpad-	数字キーパッドの - キー
numpad*	数字キーパッドの * キー
numpad/	数字キーパッドの / キー

表 A-2　KIXTERM.INI システムキー (続き)

記号名	システムキー
numpad.	数字キーパッドの . キー
pagedown	Page down キー
pageup	Page up キー
pause	Pause キー
printscreen	Print screen キー
right	右矢印
scroll_lock	Scroll lock キー
tab	Tab キー
up	上矢印

次の表に、修飾キーを示します。

表 A-3　KIXTERM.INI 修飾キー

記号名	システムキー
Shift	Shift キー
Control	左コントロールキー
Ctrl	左コントロールキー
Alt	右 Alt キー

端末の標準色およびライト色の定義

システム色マッピングによって、端末で使用する標準形式およびライト形式の色を定義できます。表 A-4 に、システム色を示します。

形式:

system-color = RGB(*red*,*green*,*blue*)

red、*green*、および *blue* は、表示されるシステム色の量を示す 0 から 255 の数値です。たとえば、次のように指定します。

```
light_white = RGB(255,255,255)
```

色のマッピング

色マッピングによって、記号名から 3270 値へマップできます。色マッピングは次の 2 つの形式によって、エミュレータが使用する色表示を定義します。

- 色属性を含む 3270 フィールドを使用します。たとえば、次のように指定します。

  ```
  3270 color = System color
  ```

- 色属性を含まないフィールドを使用します。この形式では、フィールドの色はフィールド属性によって判定されます。たとえば、次の文によって標準のすべての非保護フィールドがデフォルトでライトグリーンで表示されます。

  ```
  normal_unprotected = light_green
  ```

次の表に、3270 の色、3270 システム色、およびフィールド属性の記号表記を示します。

表 A-4 KIXTERM.INI の色

3270 の色	システム色	フィールド属性
Default	red / light_red	normal_unprotected
red	green / light_green	normal_protected
green	blue / light_blue	intensified_unprotected
blue	pink / light_pink	intensified_protected
pink	cyan / light_cyan	
turquoise	yellow / light_yellow	
yellow	white / light_white	
white	black	

キーボードのリセット

[general] セクションには、キーまたは色マッピングに関連しない初期設定オプションを指定します。このリリースのエミュレータでは、次の 1 つのエントリだけが有効です。

```
kbd_reset = end_of_transaction
```

kbd_reset オプションによって、トランザクションが終了するたびにキーボードがリセットされます。このオプションによって端末の使用がさらに簡単になるので、エミュレータに組み込まれています。このアクションは標準 3270 アーキテクチャーではないので、必要でない場合は、次のようにセミコロンでコメントアウトすることによって KIXTERM.INI ファイルから削除できます。

```
; kdb_reset = end_of_transaction
```

コメントアウトした場合、トランザクションによってキーボードをロックされると、キーボードのリセットが必要な場合があります。

付録 B

メッセージ

この章では、Sun MTP Client のメッセージについて説明します。この章の内容は、次のとおりです。

メッセージの確認

Sun MTP Client は、さまざまな状態メッセージおよびエラーメッセージを発行します。これらのメッセージの確認には、次の 2 つの方法があります。

- Sun MTP Client Messages Display で確認する
- メッセージログファイル `KIXCLI.MSG` で確認する

Sun MTP Client Messages Display に表示されるのは、最新の 100 個のメッセージのみです。

メッセージの形式

すべてのメッセージは同じ形式です。

形式

`UCM####`*a* *Text of message*

各部の説明は次のとおりです。

`UCM`	エラーが Sun MTP Client ソフトウェアで生成されたことを示します。
`EMU`	エラーが 3270 端末エミュレータで生成されたことを示します。
####	エラー番号を表します。
a	エラーの重要度を示す文字です。 I: 情報メッセージ W: 警告メッセージ E: 致命的ではないエラー F: 致命的なエラー T: Sun MTP に影響しないトランザクションの不正終了
Text of message	Sun MTP Client によって発行されたメッセージです。メッセージを発信したサブルーチン名が、メッセージテキストの前の角括弧内に示される場合があります。この場合、メッセージは次のように表示されます。 `UCM`####*a*［*subroutine_name*］*Message*

使いやすいように、メッセージは番号順に掲載します。メッセージ本文で使用される変数の一覧は、次のとおりです。

%*c*: メッセージ内で 1 文字に置き換えられます。

%*d*: メッセージ内で 10 進数に置き換えられます。

%*s*: メッセージ内で文字列に置き換えられます。

%*r*: メッセージ内でサブルーチン名に置き換えられます。

%*x*: メッセージ内で 16 進数に置き換えられます。

Sun MTP Client メッセージ

メッセージログファイル KIXCLI.MSG は、KIXCLI.INI 構成ファイルの MsgDir パラメータによって指定されたディレクトリに書き込まれます。このメッセージログには、最近の Sun MTP Client の実行に対するすべてのメッセージが含まれます。各メッセージには、日付および時刻スタンプが含まれます。KIXCLI.MSG ファイルは、Sun MTP Client が起動するたびに上書きされます。

UCM0001I:Sun MTP Client startup is complete

説明: 情報メッセージです。

UCM0002W:Unable to create timer

UCM0003E:Unable to resolve TCP host %s for system %s

UCM0004W:Unable to connect to TCP host %s for system %s

UCM0005E:Lost contact with TCP host %s for system %s:

UCM0006I:TCP/IP transport support loaded

説明: 情報メッセージです。

UCM0007I:Connected to system %s as ApplID %s

説明: 情報メッセージです。

UCM0008E:Redefinition of system %s; the first definition is being used

UCM0009W:No systems found

UCM0010I:Winsock used is %s

説明: 情報メッセージです。

UCM0011E:The definition of system %s has insufficient parameters

対処: 必要なパラメータを入力します。

UCM0012E:The transport %s for system %s is invalid

UCM0013E:The port %s for system %s is not numeric

UCM0014E:Default system %s does not exist.Using %s

```
UCM0015W:No default system specified.Using %s

UCM0016E:Alias %s is too long; it is being ignored

UCM0017W:Trace is enabled.TraceMask is 0x%4.4x

UCM0018W:Alias %s has port 0 specified.The default port %d will
be used

UCM0019W:Invalid MaxRequests value of %s specified

UCM0020I:Setting MaxRequests to %d
```

説明: 情報メッセージです。

```
UCM0021W:Invalid MaxSystems value of %s specified

UCM0022I:Setting MaxSystems to %d
```

説明: 情報メッセージです。

```
UCM0023E:The TCP address %s for system %s is longer than the
maximum %d characters:

UCM0024W:The description for system %s is longer than the
maximum %d characters

UCM0025I:A request has been made to cease communication with
system %s
```

説明: 情報メッセージです。

```
UCM0026E:Resource shortage while in function %s

UCM0027E:Specified TraceDir is invalid.Using default TraceDir

UCM0028E:Specified MsgDir %s is invalid.Using %s

UCM0029E:Trace file could not be opened.Trace is disabled

UCM0030E:Client Install Transaction CCIN is invalid in
system %s

UCM0031E:Unexpected return code received while decoding a CCIN
reply from system %s

UCM0032E:Client Install Transaction CCIN could not be executed
on system %s
```

```
UCM0033W:LU6.2 Error Reply received from system %s.
Sense data is {%2.2x,%2.2x,%2.2x,%2.2x}
```

```
UCM0040I:Copyright © 2001 by Sun Microsystems, Inc.
```

説明: 情報メッセージです。

```
UCM0041I: ------------------------------------------
```

説明: 区切り線です。

```
UCM0042W:MaxSystems has been reached.Connection to system %s
has been aborted.
```

```
UCM0043E:The Sun MTP Client could not be contacted.
```

```
UCM0044I:kixctl [-s] [-l] [-m] [-D] [-c <system>]
[-d <system>] [-t <mask>]
```

説明: `kixctl` コマンドの使用法メッセージです。コマンドが正しく入力されていない可能性があります。

対策: 構文を確認して、コマンドを正しく入力します。

```
UCM0045E:kixctl [-s] [-l] [-m] [-D] [-c <system>]
[-d <system>] [-t <mask>]
```

説明: `kixctl` コマンドの使用法メッセージです。コマンドが正しく入力されていない可能性があります。

対策: 構文を確認して、コマンドを正しく入力します。

```
UCM0046I:A connection request has been issued for system ‘%s’
```

説明: 情報メッセージです。

```
UCM0047E:System name ‘%s’ is not valid
```

```
UCM0048E:System name ‘%s’ could not be connected
```

```
UCM0049I:A disconnection request has been issued for system
‘%s’
```

説明: 情報メッセージです。

```
UCM0050E:System ‘%s’ could not be disconnected
```

```
UCM0051E:Contact with the Sun MTP Client has been lost
```

```
UCM0052E:Remote system %s sent invalid block
header %2.2x%2.2x%2.2x%2.2x%2.2x%2.2x%2.2x%2.2x
```

UCM0053I:A request to shutdown the Sun MTP Client has been issued

説明: 情報メッセージです。

UCM0054I:Sun MTP Client shutdown is complete

説明: 情報メッセージです。

UCM0055E:A reply from the Sun MTP Client could not be obtained

UCM0056I:A dump is being taken into %s

説明: 情報メッセージです。

UCM0057E:A request to dump could not be performed

UCM0058E:The Sun MTP Client could not create pipe %s

UCM0059I:Destroying terminated application pipe %s

説明: 情報メッセージです。

UCM0060E:Could not delete file %s

UCM0061E:Could not create file %s

UCM0062E:The dump request could not be processed

UCM0063E:A dump request has been issued

UCM0064E:The TCP/IP transport support library could not be loaded

UCM0065I:SNA transport support loaded

説明: 情報メッセージです。

UCM0066W:An SNA conversation could not be allocated. primary_rc = 0x%4.4x, secondary_rc = 0x%8.8lx

UCM0067E:The SNA Remote LU name %s for system %s is longer than the maximum %d characters

UCM0068E:The SNA Local LU name %s for system %s is longer than the maximum %d characters

UCM0069E:The SNA Mode name %s for system %s is longer than the maximum %d characters

UCM0070E:The Remote LU Alias %s for system %s is invalid

UCM0071E:The Mode Name %s for system %s is invalid

UCM0072W:The local SNA Transaction Program could not perform a TP_START. primary_rc = 0x%4.4x, secondary_rc = 0x%8.8lx

UCM0073E:The Local LU Alias %s for system %s is invalid

UCM0074E:The SNA transport support library could not be loaded

UCM0075E:WinAPPCStartup() failed with return code 0x%8.8x

UCM0076E:The SNA PU2.1 node could not be contacted

UCM0077E:The underlying network subsystem is not ready for network communication

UCM0078E:A connection attempt to system %s failed due to a transport initialization failure

UCM0079E:An attempt to run the terminal install transaction CTIN failed on system %s

UCM0080I: %s BuildStamp %s

説明: 情報メッセージです。

エミュレータメッセージ

この節では、Sun MTP Client 3270 端末エミュレータおよびプリンタエミュレータによって生成されるメッセージについて説明します。

EMU0001E Unknown error

EMU0002E Terminal not initialized

EMU0003E Bad Index

EMU0004E Failed

EMU0005E Unexpected datastream

EMU0006E Out of memory

EMU0007E Duplicate netname

EMU0008E Unknown netname

説明: ネット名が定義されていません。

対処: ネット名を定義または修正します。

EMU0009E Unknown devtype

EMU0010E Terminal install failed

EMU0011E Unexpected error

EMU0012E Unknown system

説明: システムが定義されていません。

対処: システム名を定義または修正します。

EMU0013W Terminal out of service

EMU0014E System unavailable

EMU0015E Internal Logic error

EMU0016E Auto Install failed

EMU0017E Terminal Install error

EMU0018E EPI version not supported

EMU0019E Is Init

EMU0020E No Systems

EMU0021E No more terminal resources

EMU0022E System Error

EMU0023E Transaction active

EMU0024E TTI active

EMU0025E No server connection

EMU0026E Invalid data length

EMU0027E Invalid datastream

EMU0028E Terminal install error

EMU0029E Closing terminal

EMU0032W Are you sure you want to quit?

EMU0033E Terminal Initialization Error:0x%4.4x

EMU0034E No connections available

EMU0040I Printer installed as netname: %s

EMU0041W Printer not installed

EMU0042I Printer file: %s

説明: 情報メッセージです。

EMU0043I Printer Command: %s

説明: 情報メッセージです。

```
EMU0044E Cannot execute command:0x%4.4x

EMU0045E Cannot open file: %s

EMU0046E Cannot open temporary file: %s
```

説明: システムが指定されたコマンドまたはファイルを検出できません。

対処: コマンドの構文を確認して正しいファイル名を使用しているかを検証し、もう一度サブミットします。

用語集

数字

3270 SNA デバイス　(名詞) IBM SNA 3270 データストリームを表示する端末デバイス。

A

API　「アプリケーションプログラミングインタフェース」を参照。

C

CICS　「顧客情報管理システム」を参照。

D

DPL　「分散プログラムリンク」を参照。

DTP　「分散トランザクション処理」を参照。

E

ECI 「外部呼び出しインタフェース」を参照。

EPI 「外部表示インタフェース」を参照。

I

ISC 「システム間通信」を参照。

L

LU 「論理ユニット」を参照。

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。

S

SNA 「システムネットワークアーキテクチャー」を参照。

Sun MTP 領域 (名詞) システム上の個別の CICS アプリケーションを定義する、一連の UNIX プロセス、ファイル、および環境変数。

T

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。

あ

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定義済みのインタフェース。API は、ルーチン名およびそれに関連する引数で構成され、該当するアプリケーションプログラミング言語の構文に準拠します。

か

外部プレゼンテーション インタフェース (EPI)
(名詞) CICS 以外のアプリケーションプログラムが Sun MTP に対して 1 つ以上の標準 3270 端末として認識されるよう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) CICS 以外のアプリケーションプログラムが、分散プログラムリンク (DPL) の規則に従いながら領域で実行されている CICS プログラムを呼び出せるプログラムを作成するための API。

会話型トランザクション
(名詞) ユーザーとの会話 (通常、`SEND/RECEIVE` シーケンス) を続けながら実行するトランザクション。

環境変数
(名詞) オペレーティングシステムやその他のプログラムの動作、またはオペレーティングシステムが認識するデバイスを指定する文字列。環境変数は、プロセスの動作環境も指定できます。たとえば、環境変数で、ホームディレクトリ、コマンドの検索パス、および使用中の端末を表すことができます。

こ

顧客情報管理システム (CICS)
(名詞) 汎用トランザクション処理環境。

し

システム間通信 (ISC) (名詞) 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による、別々のシステム間の通信。

システムネットワーク体系 (SNA) (名詞) 情報単位を送信し、ネットワークの構成および操作を制御するための、論理構造、形式、プロトコル、および操作シーケンス。

そ

ソケット (名詞) 異なるネットワークプロトコルの使用を可能にするプロセス間通信のメカニズム。

ひ

非同期処理 (名詞) 一連の非同期の操作。非同期操作は特定のイベントに関連せず、不定期または予期しないタイミングで発生します。非同期操作の実行中も、アプリケーションプログラムは実行を継続できます。

ふ

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、相互に同期して通信するトランザクション間で処理を分散すること。

分散プログラムリンク (DPL) (名詞) 領域のプログラムが他の領域のプログラムに同期リンクするシステム間通信の方法。

ろ

論理処理単位 (名詞) 回復可能なリソースに対する一連の更新を設定するために必要な領域でのすべての処理。

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、およびエンドユーザーがシステムサービス制御点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

I

K

L

M

N

P

S

T

U

V

W

あ

へ

ほ

め

よ

ら

れ

ろ