CICS® Transaction Gateway

Gateway プログラミング

バージョン 4.0

SC88-8990-00
(英文原典：SC34-5938-00)

CICS® Transaction Gateway

Gateway プログラミング

バージョン 4.0

SC88-8990-00
(英文原典：SC34-5938-00)

ご注意

本書の情報およびそれによってサポートされる製品を使用する前に、 85ページの『付録D. 特記事項』に記載する一般情報をお読みください。

本書は CICS Transaction Gateway のバージョン 4.0 (プログラム番号 5724-A75) に適用されます。また、新版で特に明示されない限り、これ以降のすべてのバージョン、リリース、および修正レベルにも適用されます。

本書は、SD88-7321 の改訂版です。ページ左側の縦線は、この版で新しくなった部分を示しています。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

`http://www.ibm.com/jp/manuals/main/mail.html`

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

`http://www.ibm.com/jp/manuals/` の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

原　典： SC34-5938-00
CICS® Transaction Gateway
Gateway Programming
Version 4.0

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2001.6

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この (書体*) は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

本書について

このマニュアルは、CICS Transaction Gateway の Java™ プログラミング入門です。本書では、Java クラスおよび Java Beans について取り上げ、 VisualAge® for Java™ を使用して CICS トランザクションにアクセスするためのアプリケーションを開発する方法を説明します。

本書の対象読者

本書は、CICS Transaction Gateway のプログラミングにかかわっている人を対象としています。

CICS Transaction Gateway を実行するオペレーティング・システムについての知識が前提条件になります。

インターネット用語の知識も役立ちます。

本書の表記規則と用語

本書では、*CICS ユニバーサル・クライアント* は、CICS Transaction Gateway のクライアント・コンポーネントのことを示します。

分かりやすいように、「C」は、C および C++ プログラム言語の両方を示すために使用されます。

「System/390 上の CICS®」は、次の CICS サーバー製品を示します。

- CICS® for MVS/ESA™
- CICS® Transaction Server (OS/390® 版)
- CICS® Transaction Server (VSE/ESA™ 版)
- CICS/VSE®

CICS Transaction Gateway は、 Windows NT® および Windows 2000 で稼働します。本書中、 Windows® は、特にバージョンが指定されてい場合は、Windows NT と Windows 2000 の両方を示します。

本書中、OS/390 は、OS/390 と z/OS オペレーティング・システムの両方を示します。

前提条件となる情報とその他の関連情報

ここでは、CICS Transaction Gateway 関連のマニュアルを取り上げます。

IBM CICS Transaction Gateway の資料

CICS Transaction Gateway と CICS ユニバーサル・クライアントのライブラリーに含まれているマニュアルについては、 79ページの『付録C. CICS Transaction Gateway と CICS ユニバーサル・クライアントのライブラリー』を参照してください。その付録では、 CICS ユニバーサル・クライアントに添付されているソフトコピー・マニュアルを表示したり印刷したりするための詳しい方法について説明しています。

変更の要約

CICS Transaction Gateway バージョン 4.0 における機能変更を以下に示します。

- EPI サポート・クラスが更新されました。23ページの『第4章 EPI サポート・クラス』を参照してください。
- EPI Bean が提供されるようになりました。
- ECI および EPI リソース・アダプターが提供されるようになりました。
- Java サンプルが更新されました。CICS Transaction Gateway の Samples ディレクトリーの Samples.txt ファイルを参照してください。
- HP-UX および Linux がサポートされるようになりました。
- 以下は、サポートされなくなりました。
  - Cobol
  - PL/I
  - REXX
  - CICSTELD
  - OS/2®、Windows 95、Windows 98
  - Netware for SAA®
  - Windows オペレーティング・システムにおける次の機能
    - 16-bit アプリケーション
    - NetBIOS および DCE プロトコル
    - IBM VisualAge® C++ for Windows®
  - Solaris オペレーティング・システムにおける次の機能
    - SUNLINK

本書は、SD88-7321 の改訂版です。ページ左の縦線は、この版での新規内容を示します。

第1章 CICS Transaction Gateway プログラミング・インターフェースの概要

この章では、 CICS Transaction Gateway の共通プログラミング・インターフェースを構成する、クラス、インターフェース、Java Beans について紹介します。

- **Java クライアント・プログラム・クラス**
  - com.ibm.ctg.client.JavaGateway
  - com.ibm.ctg.client.ECIRequest
  - com.ibm.ctg.client.EPIRequest
  - com.ibm.ctg.client.ESIRequest
  - com.ibm.ctg.client.CicsCpRequest
  - com.ibm.ctg.client.Callbackable (インターフェース)

  詳細については、 5ページの『第2章 Java™ クライアント・プログラムの作成』を参照してください。
- **Gateway セキュリティー・クラスを作成するためのインターフェース定義と認証オブジェクト**
  - com.ibm.ctg.security.ClientSecurity
  - com.ibm.ctg.security.ServerSecurity
  - com.ibm.ctg.security.SSLightServerSecurity
  - com.ibm.sslight.SSLCert[]
  - com.ibm.ctg.security.SystemSSLServerSecurity (OS/390 のみ)
  - com.ibm.gskssl.SSLCertificate (OS/390 のみ)
  - com.ibm.ctg.util.RACF.Userid (OS/390 のみ)

  詳細については、 19ページの『第3章 CICS Transaction Gateway セキュリティー・クラス』を参照してください。
- **CICS Transaction Gateway EPI サポート・クラス**
  - com.ibm.ctg.epi.AID
  - com.ibm.ctg.epi.EPIGateway
  - com.ibm.ctg.epi.Field
  - com.ibm.ctg.epi.FieldData
  - com.ibm.ctg.epi.Map
  - com.ibm.ctg.epi.MapData
  - com.ibm.ctg.epi.Screen
  - com.ibm.ctg.epi.Terminal

- com.ibm.ctg.epi.TerminalSession (インターフェース)
- com.ibm.ctg.epi.TerminalInterface
- com.ibm.ctg.epi.Session (インターフェース)
- com.ibm.ctg.epi.EPIException および次のサブクラス
  - com.ibm.ctg.epi.EPIGatewayException
  - com.ibm.ctg.epi.EPIRequestException
  - com.ibm.ctg.epi.EPISecurityException
  - com.ibm.ctg.epi.EPITxnFailedException
  - com.ibm.ctg.epi.TerminalException
  - com.ibm.ctg.epi.EPIMapException
  - com.ibm.ctg.epi.EPI3270Exception
  - com.ibm.ctg.epi.EPIFieldException
  - com.ibm.ctg.epi.EPIScreenException

これらのクラスは、 CICS® Transaction Gateway (OS/390® 版) では使用できません。

詳細については、 23ページの『第4章 EPI サポート・クラス』を参照してください。

- **CICS Transaction Gateway EPI bean クラス**
  - com.ibm.ctg.epi.EPIBasicScreenHandler クラス
  - com.ibm.ctg.epi.EPITerminal クラス
  - com.ibm.ctg.epi.ScreenEvent クラス
  - com.ibm.ctg.epi.ScreenEventListener インターフェース
  - com.ibm.ctg.epi.ScreenHandler クラス
  - com.ibm.ctg.epi.TerminalEvent クラス
  - com.ibm.ctg.epi.TerminalEventListener インターフェース

これらのクラスは、 CICS® Transaction Gateway (OS/390® 版) では使用できません。

詳細については、 41ページの『第5章 EPI beans』を参照してください。

注: CICS Transaction Gateway では、ECI プログラミングに比べて、 EPI プログラミング (サポート・クラス、beans) のほうに重点が置かれているように見えるかもしれません。これは、この 2 つのインターフェースの性質によるものです。ECI と EPI インターフェースの詳細については、「*CICS*® *ファミリー: クライアント / サーバー・プログラミング*」を参照してください。

CICS Transaction Gateway のクラスとインターフェースに関するオンライン情報が、ハイパーテキスト・マークアップ言語 (HTML) 形式で用意されています。詳細については、 69ページの『第7章 Java クラスの参照情報』を参照してください。

第2章 Java™ クライアント・プログラムの作成

この章では、 CICS Transaction Gateway のための Java クライアント・プログラムを作成する方法を紹介します。

CICS Transaction Gateway には、 Java クライアント・プログラムを作成するための以下のような基本クラスが用意されています。

com.ibm.ctg.client.JavaGateway
このクラスは、プログラムと CICS Transaction Gateway との間の論理接続を表します。アクセスしたいと思っているそれぞれの CICS Transaction Gateway に、 JavaGateway オブジェクトが必要です。

com.ibm.ctg.client.ECIRequest
このクラスは、 Gateway に対する ECI 要求の詳細を収容します。このクラスは、各種パラメーターのストリングを取得します。指定したストリングが長すぎる場合は、最大長に切り捨てられます。

com.ibm.ctg.client.EPIRequest
このクラスは、 Gateway に対する EPI 要求の詳細を収容します。このクラスは、各種パラメーターのストリングを取得します。指定したストリングが長すぎる場合は、最大長に切り捨てられます。

com.ibm.ctg.client.ESIRequest
このクラスは、 Gateway に対する ESI 要求の詳細を収容します。

com.ibm.ctg.client.CicsCpRequest
このクラスは、 CICS ユニバーサル・クライアントが使用するコード・ページの詳細を収容します。

com.ibm.ctg.client.Callbackable
CICS Transaction Gateway がサポートしている非同期モデルでは、コールバック・オブジェクトを使用できます。このインターフェースは、Callbackable オブジェクトが提供しなければならないメソッドを定義します。

プログラム制御の流れ

最も単純に考えれば、簡単な CICS Transaction Gateway Java クライアント・プログラムを作成するために必要なプログラム制御の流れは、次のようになります。

1. Java プログラムが、 com.ibm.ctg.client.JavaGateway オブジェクトのインスタンスを作成してオープンします。
   - デフォルトの JavaGateway コンストラクターが、ブランクの JavaGateway オブジェクトを作成します。ユーザーはその後、該当する set.. メソッドを使用して、このオブジェクトに正しいプロパティーを設定する必要があります。それができたら、open メソッドを呼び出して、JavaGateway をオープンします。
   - 他にも 2 つの JavaGateway コンストラクターにより、関連プロパティーを設定したり、ユーザーの代わりに open メソッドを暗黙的に呼び出したりして、JavaGateway の作成を簡単に行えます。これらのコンストラクターの呼び出しが正常に行われると、作成された JavaGateway がオープンして、要求対象の CICS Transaction Gateway に接続します。
2. Java プログラムが、それぞれの要求に合わせて、該当する要求を収容する Gateway 要求クラスのいずれか 1 つのインスタンスを作成します。
   - ECI 要求の場合は、com.ibm.ctg.client.ECIRequest が作成されます。
   - EPI 要求の場合は、com.ibm.ctg.client.EPIRequest が作成されます。
   - ESI 要求の場合は、com.ibm.ctg.client.ESIRequest が作成されます。
   - 接続を媒介する CICS ユニバーサル・クライアントのコード・ページを照会する場合は、 com.ibm.ctg.client.CicsCpRequest が作成されます。
3. Java プログラムが、 JavaGateway オブジェクトの flow メソッドを使用して、要求を CICS Transaction Gateway に流します。
4. Java プログラムが、流れ操作の戻りコードをチェックして、要求が正常に実行されたかどうかを確認します。
5. プログラムが、必要に応じて、要求オブジェクトの作成を続け、JavaGateway オブジェクトによって要求オブジェクトを流し続けます。
6. Java プログラムが、 JavaGateway オブジェクトをクローズします。

JavaGateway

JavaGateway は、リモート・プロトコルを指定する際の、プログラムと CICS Transaction Gateway 間の論理接続を示します。ローカル接続を指定した場合、CICS Transaction Gateway サーバーはバイパスして、直接 CICS サーバーに接続します。

たとえば、**tcp://acmectg.acme.com:2006** などの URL を指定して、リモート CICS Transaction Gateway サーバーを指定することができます。ローカル・プロトコルを使用するには、URL **local:** を指定します。

プロトコル

JavaGateway は、次のプロトコルをサポートします。

- TCP/IP
- SSL (Secure Sockets Layer)
- HTTP
- HTTPS (アプレットを使用。または、ユーザーの Java 環境内で HTTPS が明示的にサポートされる場合。 Java 環境によってはサポートされない場合がある。)
- ローカル

JavaGateway を作成する場合は、使用するプロトコルを決定して、必要に応じて、リモート CICS Transaction Gateway サーバーの接続詳細 (ネットワーク・アドレスおよびポート番号) を決めます。 JavaGateway クラスの場合、setAddress、setProtocol、および setPort メソッドを使用してこの情報を指定するか、 **Protocol://Address:Port** の URL 形式ですべての情報を指定することができます。 setURL メソッドを指定するか、いずれかの JavaGateway コンストラクターに URL を渡すことができます。

要求のフロー

JavaGateway を作成してオープンした後、これに要求を流します。適切な要求 (ECIRequest など) を作成して、flow メソッドに渡します。要求は、次のように送られます。

- リモート JavaGateway がある場合は、実行される CICS Transaction Gateway サーバーに送信。
- ローカル JavaGateway を使用する場合は、直接 CICS に送信。

JavaGateway オブジェクトを作成すると、複数のスレッドにより参照可能な単一オブジェクトができます。

JavaGateway セキュリティー

リモート CICS Transaction Gateway に接続する場合、特定の接続に割り振られたリソース、および特定の接続における要求オブジェクトに指定された ID は、その接続においてのみ使用することができます。

ECI 要求

論理作業単位 (LUW) ID およびメッセージ修飾子は、これらを作成または割り当てたものと同じ JavaGateway においてのみ使用可能です。これは、セキュリティー機能です。これにより、同じ CICS Transaction Gateway サ

ーバーに接続されているプログラムが、別のアプリケーションの LUW ID を操作したり、メッセージ修飾子を使用してメッセージを要求することのないようにしています。たとえば、異なる JavaGateway からのメッセージに対する固有の応答を取得しようとすると、 ECI_ERR_NO_REPLY コードが戻ります。

EPI 要求

端末 ID は、その端末を作成した JavaGateway においてのみ使用することができます。これも、同じ CICS Transaction Gateway に接続する他のプログラムがその端末を操作することのないようにするためのセキュリティー機能です。

ローカル・プロトコルを指定した場合の動作は異なります。同じ JVM (つまり同じプロセス) で作成されたすべての JavaGateway は、互いの割り振りリソースまたは指定 ID にアクセスすることができます。

SSL JavaGateway

JavaGateway オブジェクトの作成と、これを使用した作業が中心となります。ただし、SSL については、ゲートウェイに対し鍵リング・クラスおよびパスワードを通知する必要があります。これには、**SslJavaGateway** クラスで静的メソッド **setKeyRing** を呼び出します。

ECIRequest

ECI タイプの呼び出しを CICS に行うには、ECIRequest オブジェクトを作成する必要があります。これらのオブジェクトの詳細については、「*CICS*® ファミリー: クライアント / サーバー・プログラミング」を参照してください。

コールバック

ECIRequest は、コールバック・オブジェクトをサポートします。コールバック・オブジェクトは、**Callbackable** インターフェースを実装する必要があり、setResults メソッドを経由してフローの結果を受信します。結果が適用されると、run メソッドを実行するための新しいスレッドが開始されます。

コールバック・オブジェクトは、非同期呼び出しタイプだけではなく、すべての ECIRequest 呼び出しタイプに対して指定することができます。同期呼び出しの場合、結果はフロー要求における ECIRequest オブジェクト、および Callbackable オブジェクトにも渡されます。

メッセージ修飾子

メッセージ修飾子を使用することにより、非同期 ECI 要求呼び出しを作成して、後から結果を受け取ることができます。(非同期呼び出しの場合、要求の結果を待たずにフローを継続することができます。) これらの結果を受け取ることができるようにするため、要求の呼び出しに番号をタグ付けする必要があります。この番号 (*メッセージ修飾子* と呼ばれる) を使用して、その呼び出しについての結果を識別します。

リモート CICS Transaction Gateway 接続の場合、メッセージ修飾子は、CICS Transaction Gateway サーバー内で固有のものである必要があります。要求を作成して、すでに使用されているメッセージ修飾子の値を使用すると、そのメッセージ修飾子は使用できないことを示すエラー・メッセージを受け取ります。ECIRequest では、**メッセージ修飾子の自動生成**機能により、この問題を回避しています。ECIRequest オブジェクトでメッセージ setAutoMsgQual(true) を呼び出して、この機能をオンにします。これにより、すべての非同期要求 (ECI_ASYNC、ECI_ASYNC_TPN、ECI_STATE_ASYNC、ECI_STATE_ASYNC_JAVA) において固有のメッセージ修飾子が作成されます。呼び出しタイプ ECI_GET_SPECIFIC_REPLY および ECI_GET_SPECIFIC_REPLY_WAIT を使用する場合、フロー後の ECIRequest に含まれる値を使用して、結果を検索することができます。

リモート接続の場合、メッセージ修飾子を使用してオリジナル要求のフローを行った接続への応答を、異なる接続で受け取ることはできません。6ページの『JavaGateway』を参照してください。

ローカル接続の場合、要求ごとに固有のメッセージ修飾子が必要ですが、これは強制ではありません。また、異なる接続に要求のフローが行われた場合でも、いずれの接続においてもメッセージ修飾子を使用してメッセージを取得できますが、これには JavaGateway が同じ JVM 内にあることが条件となります。ただし、次を目的とする場合、メッセージ修飾子の自動生成機能を使用することをお勧めします。

- 同じメッセージ修飾子を再利用することによる問題を回避する。
- ローカルおよびリモート接続間でアプリケーションを切り替える。

メッセージ修飾子は、CICS® Transaction Gateway (OS/390® 版) ではサポートされません。

EPIRequest

EPI タイプの呼び出しを CICS に行うには、EPIRequest オブジェクトを作成する必要があります。これらのオブジェクトの詳細については、「*CICS*® ファミリー: クライアント / サーバー・プログラミング」のガイド を参照してください。CICS® Transaction Gateway (OS/390® 版) では、EPIRequest オブジェクトはサポートされません。

EPIRequest クラスの使用

EPI を使用してプログラムを作成する場合は、 EPI beans または CICS Transaction Gateway EPI サポート・クラスを使用することをお勧めします。ただし、EPIRequest クラスを使用する予定があれば、このセクションに目を通してください。

EPI を使用して CICS に接続する場合は、 Java アプリケーションが、 CICS に対する端末として動作します。したがって、 CICS から流れる 3270 データ・ストリームと、応答として予期される 3270 データ・ストリームについての情報が重要になります。 Java アプリケーションにイベントが戻れされると、 EPIRequest オブジェクトのサイズ・フィールドに、戻されたデータ配列のサイズが表示されます。

また、EPI プログラミングの基本原則や制限事項、および EPI コードの動作にはオペレーティング・システムによって若干の違いがあることなども知っておく必要があります。たとえば、Windows NT で CICS クライアントを実行する場合は、EPIRequest オブジェクトの Transid フィールドではなく、EPIRequest オブジェクトのデータ配列にトランザクション ID を送信する必要がある場合があります。

CICS からイベントを受け取る場合は、EPI_WAIT オプションを使用して、EPIRequest オブジェクトのサイズ・フィールドが、CICS から戻される 3270 データ・ストリームの最大サイズに設定されていることを確認するようにお勧めします。

パラメーター長: EPIRequest クラスを使用する場合、各種パラメーターには最大長があることに注意してください。この長さを超えるパラメーターは、切り捨てられます。

一般に、 CICS Transaction Gateway を使用して作成する EPI プログラムは、以下のように動作します。

1. Gateway との接続をオープンします。
2. 端末を追加します。

3. トランザクションを開始します。
4. 次のいずれかになるまで、イベントを取得します。
   - 受け取ったイベントが、終了トランザクションまたは会話である。
   - 受け取ったエラーが重大である。
5. 受け取ったイベントが会話の場合は、応答を送信してイベント取得ループに戻ります。
6. 受け取ったイベントが終了トランザクションの場合は、端末を削除して、終了端末イベントを受け取るための最後の取得イベントを実行します。
7. Gateway との接続をクローズします。

端末索引

リモート接続の場合、端末索引は、その割り当てられている接続においてのみ使用することができます。詳細については、6ページの『JavaGateway』を参照してください。ローカル接続の場合は、同じ JVM 内にあるのであれば、すべてのローカル JavaGateway はその他のローカル JavaGateway にある端末索引にアクセスが可能です。

CLASSPATH のセットアップ

Java クライアント・プログラムを作成する前に、 CLASSPATH 環境変数を更新して、CICS Transaction Gateway に添付されている jar ファイルを組み込むようにしてください。最初に、ctgclient.jar ファイルへのパスを組み込みます。ローカル CICS Transaction Gateway を使用するつもりであれば、さらに ctgserver.jar ファイルのパスを CLASSPATH ステートメントに指定しなければなりません。たとえば、CICS® Transaction Gateway (Windows NT® 版) の場合は、次のようになります。

```
CLASSPATH = C:¥Program Files¥IBM¥CICS Transaction
Gateway¥classes¥ctgclient.jar;C:¥Program Files¥IBM¥CICS
Transaction Gateway¥classes¥ctgserver.jar
```

CLASSPATH を設定する詳しい方法については、該当するオペレーティング・システムの「*CICS Transaction Gateway 管理*」を参照してください。

コード・ページ変換

アプリケーションのコード・ページが、サーバーのコード・ページと異なる場合は、COMMAREA のデータを変換する必要があります。変換するには、DFHCNV マクロ定義など、CICS 提供のリソース変換機能をサーバー上で実行します。

ブラウザーと CICS Transaction Gateway を同じワークステーションで使用する

ブラウザーと CICS Transaction Gateway を同じワークステーションで使用する場合は、 CLASSPATH 設定から ctgclient.jar と ctgserver.jar を削除します。その 2 つのファイルを削除しない場合、アプリケーションを実行したときに、次のようなエラーが表示される場合があります。

```
ERROR: java.io.IOException:
CCL6664E: Unable to load relevant class to support the tcp protocol
```

このエラーは、 Java がネットワーク経由でクラスをダウンロードする前に、CLASSPATH 環境変数を検索したことが原因です。必要なクラスがローカルにある場合、 Java はそのクラスを使用しようとします。しかし、ローカル・ファイル・システムにあるクラス・ファイルを使用することは、 Java アプリケーションのセキュリティー規則に違反しているので、例外が発生します。

パフォーマンスの問題

Java クライアント・アプリケーションを実行するときには、パフォーマンスに関する問題をいくつか考えなければなりません。Java 仮想マシン (JVM) は、アプリケーションの各実行スレッドごとに、固定サイズのスタック・スペースを割り振ります。通常は、以下のサイズに制限を設け、Java が割り振るスペースの量を制御できます。

- ネイティブ・スタック・サイズ。ネイティブ JIT (ジャストインタイム) コンパイル・コードの実行時に割り振られます。
- Java スタック・サイズ。 Java バイトコードの実行時に割り振られます。
- Java ヒープ・サイズの初期値。
- Java ヒープ・サイズの最大値。

制限の設定方法は、JVM によって異なります。詳細については Java のドキュメンテーションを参照してください。

Java クライアント・プログラムでのスレッドの使用

アプリケーションで Java のネイティブ・スレッド・サポートを使用する場合は、次の Web サイトより、Sun のホワイト・ペーパーを参照してください。

```
www.sun.com/software/solaris/java/wp-java/
```

Java クライアント・プログラムのトレース機能

Java クライアント・プログラムのトレース機能は、以下のものを使用して制御できます。

- com.ibm.ctg.client.T クラスの呼び出し

  たとえば、ユーザー・アプリケーションで、次のように入力します。

```
if (getParameter("trace") != null)
 {
   T.setOn(true);
 }
```

  trace は、ユーザー・プログラムに設定可能な始動パラメーターです。

- Gateway.T システム・プロパティー

  例

```
java -Dgateway.T=on com.usr.smp.test.testprog1
```

  これは、testprog1 の完全デバッグを指定します。

  システム・プロパティーの詳しい使用方法については、 Java の資料を参照してください。

使用可能な各種のトレース・レベルについて、以下に示します。これらはケース・センシティブであることに注意してください。

表 1. 使用可能なトレース・レベル

トレース・レベル	com.ibm.ctg.client.T 呼び出し	システム・プロパティー	定義
標準	T.setOn (true/false)	gateway.T.trace=on	アプリケーション・トレースの標準オプション。 デフォルトでは、データ・ブロック (*commarea* またはネットワーク・フローなど) の最初の 128 バイトのみが表示されます。 このトレース・レベルは、ctgstart -trace オプションによるゲートウェイ・トレースと同じです。

表 1. 使用可能なトレース・レベル (続き)

トレース・レベル	com.ibm.ctg.client.T 呼び出し	システム・プロパティー	定義
完全デバッグ	T.setDebugOn (true/false)	gateway.T=on	アプリケーション・トレースのデバッグ・オプション。 デフォルトでは、トレースしたデータ・ブロック全体を出力します。トレースには、標準のトレース・レベルよりも多くの CICS Transaction Gateway 情報が含まれます。 このトレース・レベルは、`ctgstart -x` オプションによるゲートウェイ・デバッグ・トレースと同じです。
例外のスタック	T.setStackOn (true/false)	gateway.T.stack=on	アプリケーション・トレースの例外スタック・オプション。 通常の CICS Transaction Gateway のオペレーションに予想される例外を含め、ほとんどの Java 例外をトレースします。その他のトレースは、書き込まれません。 このトレース・レベルは、`ctgstart -stack` オプションによるゲートウェイ・スタック・トレースと同じです。

次のオプションを使用して、さらにトレースを構成することができます。

表 2. トレースの構成

com.ibm.ctg.client.T 呼び出し	システム・プロパティー	オプションの使用法
T.setTFile(true,*filename*)	gateway.T.setTFile=*filename*	値 *filename* は、トレース出力を書き込むファイル位置を指定します。これは、stderr のデフォルト出力の代わりとなります。長いファイル名は、引用符で囲みます。たとえば、″trace output file.log″ のようにします。
T.setTruncationSize(*number*)	gateway.T.setTruncationSize=*number*	値 *number* は、トレースに書き込むデータ・ブロックの最大サイズを指定します。いずれの正の整数も有効です。値 *0* を指定すると、トレースにデータ・ブロックは書き込まれません。このオプションに負の値を割り当てた場合、例外 java.lang.IllegalArgumentException が出されます。
T.setDumpOffset(*number*)	gateway.T.setDumpOffset=*number*	値 *number* は、データ・ブロックの表示を開始するオフセットを指定します。オフセットが、表示するデータの合計長よりも大きければ、オフセット *0* が使用されます。このオプションに負の値を割り当てた場合、例外 java.lang.IllegalArgumentException が出されます。
T.setTimingOn (true/false)	gateway.T.timing=on	トレースにタイム・スタンプを表示するかどうかを指定します。

表2 のオプションを、いずれかのディレクティブと合わせて使用して、トレースをオンにします。

たとえば次にようにすると、標準トレースをオンにして、ダンプするデータ・ブロックの最大サイズを 20 000 バイトに設定します。

```
java -Dgateway.T.trace=on -Dgateway.T.setTruncationSize=20000
```

OS/390® における ECI 呼び出し

CICS Java クラスを使用することにより、ECI への呼び出しを表す **ECIRequest** オブジェクトを作成することができます。

プログラム・リンク呼び出し

同期および非同期のプログラム・リンク呼び出し (ECI_SYNC および ECI_ASYNC 呼び出しタイプ) の要求を表す **ECIRequest** オブジェクトを作成することができます。

これには、次の制約事項があります。

- ECI に渡される通信域 (COMMAREA) の長さは、 32 659 バイトを超えてはなりません。
- 応答請求呼び出しはサポートされません。非同期呼び出しにより、コールバック・ルーティングを指定してください。
- **eci_message_qualifier** の使用はサポートされません。ただし、コールバック・ルーティングで ECI_ASYNC を使用することができます。
- **eci_timeout** の使用はサポートされません。

状況情報呼び出し

ECI 状況情報呼び出し (ECI_STATE_SYNC および ECI_STATE_ASYNC 呼び出しタイプ) は、結果を CICS 通信域に戻します。通信域の内容はプラットフォームに依存しているため、 **ECIRequest** クラスには、状況情報呼び出しの結果を解釈するための補助機能があります。

- 状況情報呼び出しの結果をプラットフォーム固有の形式で保持するための public 変数
- 状況情報呼び出しの結果を、通信域ではなく public 変数に戻す 2 つの呼び出しタイプ (ECI_STATE_SYNC_JAVA および ECI_STATE_ASYNC_JAVA)
- public 変数の内容を表示用のストリングとして解釈するためのメソッド

状況情報呼び出しの拡張モードでは、ECI_STATE_IMMEDIATE 値のみサポートされます。

応答請求呼び出し

応答請求呼び出しを表す **ECIRequest** オブジェクトを作成することができます。

応答請求呼び出しはサポートされません。

CICS セキュリティーの考慮事項

CICS® Transaction Gateway (OS/390® 版) 領域は、EXCI ユーザーであるため、以下のセキュリティーの考慮事項が適用されます。

- CICS® Transaction Gateway (OS/390® 版) アドレス・スペースのユーザー ID が、CICS アドレス・スペースのユーザー ID と *同じ* 場合、次のようになります。
  - LINK セキュリティー検査は行われません。
  - 代理ユーザー検査が行われるため、CICS® Transaction Gateway (OS/390® 版) アドレス・スペースのユーザー ID は、 ECI 呼び出しに渡されるユーザー ID の使用権限が必要です。
- CICS® Transaction Gateway (OS/390® 版) アドレス・スペースのユーザー ID が CICS アドレス・スペースのユーザー ID と異なる 場合、接続の ATTACHSEC の設定に応じて、LINK セキュリティー検査が行われます。ECI 呼び出しで渡されるユーザー ID が検証されます。
- すべての EXCI 呼び出しについて、 ECIRequest オブジェクトにコーディングされたユーザー ID およびパスワードが、 RACF® により検査されます。
- 各 EXCI 呼び出しについて、ユーザー ID およびパスワード認証が使用不可になります。これは、ctgstart スクリプトの AUTH_USERID_PASSWORD 環境変数を使用して行います。 詳細については、「*CICS® Transaction Gateway (OS/390® 版): 管理*」の構成の章を参照してください。

OS/390® における ECI 戻りコードおよびサーバー・エラー

このセクションでは、**ECIRequest** オブジェクトのユーザーに戻される EXCI の戻りコードについて説明します。

表3 は、EXCI 戻りコードと ECI 戻りコードとのマッピングを示しています。EXCI 戻りコードについては、「*CICS 外部インターフェース・ガイド*」に記載されています。

表 3. EXCI 戻りコードおよび ECI 戻りコード

EXCI 戻りコード	ECI 記号名 / 戻りコード	rc
201、203	ECI_ERR_NO_CICS	—3
202	ECI_ERR_RESOURCE_SHORTAGE	—16
401、402、403、404、410、411、412、413、418、419、421	ECI_ERR_SYSTEM_ERROR	—9
422	ECI_ERR_TRANSACTION_ABEND	—7

表 3. EXCI 戻りコードおよび ECI 戻りコード (続き)

EXCI 戻りコード	ECI 記号名 / 戻りコード	rc
423	ECI_ERR_SECURITY_ERROR	—27
601、602、603、604、605、606、607、608、621、622、623、627、628	ECI_ERR_SYSTEM_ERROR	—9
609	ECI_ERR_SECURITY_ERROR	—27
624	ECI_ERR_REQUEST_TIMEOUT	—5

OS/390® における EPI 呼び出し

CICS Java クラスを使用することにより、EPI への呼び出しを表す **EPIRequest** オブジェクトを作成することができます。 **EPIRequest** オブジェクトに public 変数 `Call_Type` を指定して、作成する EPI 呼び出しを指定することができます。EPI 呼び出しの結果は、 **JavaGateway** オブジェクトの **flow** メソッドを使用した後、オブジェクトに戻されます。

ただし、予想されるように、**EPIRequest** オブジェクトをフローしようとすると、拒否されます。戻りコード EPI_ERR_FAILED が戻されます。詳細を得ようとして、**EPI_GetSysError** を使用しないでください。EPI の形式でトランザクションを実行する場合は、ECI を使用して、DFHWBTTA の要求を設定してください。これについては、「*CICS インターネット・ガイド*」に記載されています。

OS/390® における EXCI の考慮事項

EXCI のバージョン 2 がサポートされており、eci_transid のサポートが提供され、以前の ASCII/EBCDIC 変換の問題が解決されています。

EXCI バージョン 2 を使用して、ECI 要求で eci_tpn が指定されている場合、ユーザーのミラー・トランザクションの定義では、トランザクション定義がローカルであるかリモートであるかに関係なく、PROGRAM(DFHMIRS) を指定する必要があります。

第3章 CICS Transaction Gateway セキュリティー・クラス

CICS Transaction Gateway には、セキュリティーを確保するための以下のようなクラスが用意されています。

com.ibm.ctg.security.SSLightServerSecurity

SSL クライアント証明書の提示が必要な、CICS Transaction Gateway のサーバー側のセキュリティー・クラスを実装する場合は、**SSLightServerSecurity** インターフェースを実装する必要があります。

pure Java (SSLight) プロトコル・ハンドラーを使用する場合は、このインターフェースを実装するとよいでしょう。SSLight は、CICS Transaction Gateway のすべてのプラットフォームで使用できます。

com.ibm.ctg.security.SystemSSLServerSecurity

SSL クライアント証明書の提示が必要であり、しかも System SSL プロトコル・ハンドラーを使用する、CICS Transaction Gateway のサーバー側のセキュリティー・クラスを実装する場合は、**SystemSSLServerSecurity** インターフェースを実装する必要があります。

System SSL を使用する場合は、このインターフェースを実装します。System SSL は、CICS Transaction Gateway (OS/390® 版) でのみ使用できます。

com.ibm.ctg.security.ServerSecurity

SSL クライアント証明書の提示を必要としない、 CICS Transaction Gateway のサーバー側のセキュリティー・クラスを実装する場合は、**ServerSecurity** インターフェースを実装する必要があります。

com.ibm.ctg.security.ClientSecurity

CICS Transaction Gateway のクライアント側のセキュリティー・クラスを実装する場合は、**ClientSecurity** インターフェースを実装する必要があります。

com.ibm.ctg.util.RACFUserid

このクラスは、X.509 クライアント証明書と RACF ユーザー ID をマップしようとします。証明書と有効な RACF ユーザー ID との関連付けがすでに設定されている必要があります。

SSLightServerSecurity、 **SystemSSLServerSecurity**、 **ServerSecurity** のいずれかのインターフェースと、そのパートナーになる **ClientSecurity** インターフェースは、CICS Transaction Gateway の使用時にセキュリティーを確保するための、簡単で柔軟なモデルを定義します。インターフェースの実装は、必要に応じて、簡単なものにも堅固なものにもすることができます。簡単なものにする場合は、XOR (eXclusive-OR) にスクランブルをかけるだけですが、堅固なものにする場合は、Java 暗号体系を使用します。

SSLightServerSecurity インターフェースや SystemSSLServerSecurity インターフェースは、Secure Sockets Layer (SSL) プロトコルと併用するための設計になっています。これらのインターフェースを使用した場合は、サーバー側のセキュリティー・オブジェクトから、初期 SSL ハンドシェークのときに渡されるクライアント証明書にアクセスできます。クライアント証明書の提示は、CICS Transaction Gateway がクライアント認証をサポートするように構成されているかどうかにより異なります。

個々の JavaGateway インスタンスには、JavaGateway がクローズするまでの間、ClientSecurity クラスのインスタンスが関連付けられています。同様に、パートナーである SSLightServerSecurity、SystemSSLServerSecurity、ServerSecurity のいずれかのクラスには、接続がクローズするまでの間、接続先の Java クライアントが関連付けられています。

基本的なモデルは、以下のようになります。

- 関連情報を交換するための初期ハンドシェークが行われます。たとえば、このハンドシェークでは、公開鍵の交換が行われる場合があります。ただし、インターフェースのレベルでは、そのフローが簡単なバイト配列で構成されるので、実装システムには、そのハンドシェーク・フローの内容に対する完全な制御権があります。
- 関連する ClientSecurity インスタンスが、アウトバウンド要求のエンコードとインバウンド応答のデコードに対して呼び出されます。
- CICS Transaction Gateway では、パートナーである SSLightServerSecurity、SystemSSLServerSecurity、ServerSecurity のいずれかのインスタンスが、インバウンド要求のデコードとアウトバウンド応答のエンコードを行うために呼び出されます。インバウンド要求とクライアント証明書は、**afterDecode()** メソッドによって提示されます。SSLight の場合、afterDecode() メソッドは、GatewayRequest オブジェクトと、**com.ibm.sslight.SSLCert[]** 証明書チェーン・オブジェクトを提示します。System SSL の場合、afterDecode() メソッドは、GatewayRequest オブジェクトと、**com.ibm.gskssl.SSLCertificate** 証明書オブジェクトを提示します。

ClientSecurity クラスのインスタンスと、SSLLightServerSecurity、SystemSSLServerSecurity、ServerSecurity のいずれかのクラスのインスタンスは、フローのエンコードとデコードを正しく実行するために、初期ハンドシェークからの十分な情報をデータ・メンバーとして保守するべきです。

第4章 EPI サポート・クラス

本章では、以下の項目について説明します。

- EPI サポート・クラスの概要
- EPI サポート・クラスの使用方法
- 例外処理
- 端末の切断
- 3270 データ・ストリームのエンコード
- BMS マップの変換および Map クラスの使用方法
- EPIRequest クラスの使用方法
- Terminal を使用しない Screen クラスの使用方法

EPI サポート・クラスの概要

既存の CICS サーバー・アプリケーションの多くは、 3270 端末インターフェースに対応しており、 CICS には、基本マッピング・サポート (BMS) など、その種のデータ・ストリームを処理するための便利な機能が用意されています。

外部表示インターフェース (EPI) は、CICS クライアントがサーバー上のトランザクションと通信したり、3270 データ・ストリームを処理したりするためのメカニズムを提供します。EPI を使用する CICS クライアントは、それが 3270 端末であるかのように CICS に接続し、3270 データ・ストリームを送受信する必要があります。

Java プログラマーは、CICS Transaction Gateway の EPI サポート・クラスを使用して、EPI の機能に簡単にアクセスできます。

- CICS サーバーへの 3270 セッションの接続
- CICS トランザクションの開始
- 3270 データ・ストリームの送受信

3270 データ・ストリームに関する詳細な知識は必要ではありません。これらのクラスは、3270 データ・ストリームを処理するための以下のような高水準の構成を提供します。

- フィールドや属性などの 3270 データ・ストリームと、トランザクション ID などの CICS トランザクション・ルーティング・データを処理するための汎用 Java クラス。
- BMS マップ・ソース・ファイルから、特定の CICS アプリケーションのための Java クラスが生成されます。これらのクラスを使用すると、クライアント・アプリケーションから、CICS サーバーの BMS アプリケーションで使用するフィールド名と同じフィールド名を使用して、3270 パネル上のデータにアクセスできます。

注: これらのクラスには、DBCS フィールドを含む 3270 データ・ストリームに対する特定のサポートは組み込まれていません。ただし、これらは SBCS および DBCS の混合フィールドを含む 3270 データ・ストリームはサポートしません。

BMS 変換ユーティリティーは、CICS BMS マップ・セットから、Java クラスのソース・コードを静的に生成するためのツールです。37ページの『BMS マップの変換と Map クラスの使用』を参照してください。

EPI サポート・クラスは、C++ EPI クラスに類似しています（要求されるオブジェクト、および扱われるメソッドが同様）。

本章の例においては、以下と同様のステートメントが前提とされています。

```
import com.ibm.ctg.epi.*;
import java.io.*;
```

EPI サポート・クラスは、以下のとおりです。

com.ibm.ctg.epi.AID
CICS に送信される AID (アテンション ID) 文字を表します。

com.ibm.ctg.epi.EPIGateway
CICS Transaction Gateway に接続します。このクラスには、クライアントからアクセスできる CICS サーバーの情報を取得するためのメソッドもあります。

com.ibm.ctg.epi.Field
仮想画面の単一フィールドをサポートし、フィールドのテキストと属性へのアクセスを提供します。

com.ibm.ctg.epi.FieldData
単一フィールドの情報 (行と列の位置や長さ) を収容します。

com.ibm.ctg.epi.Map
BMS マップ情報を使用して Field オブジェクトへのアクセスを提供します。BMSMapConvert ユーティリティーは、Map からの派生クラスを生成します。

com.ibm.ctg.epi.MapData
BMS マップの情報 (サイズやフィールド数など) を収容します。

com.ibm.ctg.epi.Screen
端末の Terminal オブジェクトには、仮想画面が関連付けられます。Screen クラスは、Field オブジェクトの集合と、それらのオブジェクトにアクセスするためのメソッドを収容します。また、一般的な画面処理のためのメソッドも含んでいます。

com.ibm.ctg.epi.Terminal
3270 端末から CICS への接続を制御します。Terminal クラスは、CICS の会話型トランザクション、疑似会話型トランザクション、ATI トランザクションを処理します。1 つのアプリケーションで、多くの Terminal オブジェクトを作成できます。

com.ibm.ctg.epi.TerminalInterface
Terminal クラスは、このインターフェースを実装します。

com.ibm.ctg.epi.TerminalSession (インターフェース)
セッションの端末 (関連する Session オブジェクトを持つ端末) の動作を定義します。 TerminalInterface は、この拡張を Session オブジェクトの変更を可能にする関数により行います。

com.ibm.ctg.epi.Session (インターフェース)
同期モードと非同期モードのサーバー通信を制御します。アプリケーション・クラスは、Session インターフェースを実装して、サーバーからの応答を処理できます。

com.ibm.ctg.epi.EPIException
これは、EPI サポート・クラスにより throw される例外に対するスーパークラスです。

com.ibm.ctg.epi.EPIGatewayException
この例外は、ゲートウェイに関する一般的な問題が発生した場合に throw されます。

com.ibm.ctg.epi.EPIRequestException
この例外は、EPI 要求に関する問題が発生した場合に throw されます。

com.ibm.ctg.epi.EPISecurityException
この例外は、セキュリティー要求に関する問題が発生した場合に throw されます。

com.ibm.ctg.epi.EPITxnFailedException
この例外は、トランザクションの実行に問題が発生した場合に throw されます。

com.ibm.ctg.epi.TerminalException
この例外は、Terminal クラスが問題を検出した場合に throw されます。

com.ibm.ctg.epi.EPIMapException
この例外は、マップが無効な場合に、自動生成されたマップ・クラスにより throw されます。

com.ibm.ctg.epi.EPI3270Exception
この例外は、3270 データ・ストリームの処理に問題が発生した場合に throw されます。

com.ibm.ctg.epi.EPIFieldException
この例外は、フィールド・オブジェクトとの対話で問題が発生した場合に throw されます。

com.ibm.ctg.epi.EPIScreenException
この例外は、Screen オブジェクトとの対話で問題が発生した場合に throw されます。

EPI サポート・クラスの使用方法

ここでは、 EPI サポート・クラスの使用方法を説明します。コードの例も記載しています。

CICS への端末の確立

以下に、CICS サーバーに対する端末の確立方法を説明します。サインオン機能や読み取りタイムアウトなどの EPI および端末のプロパティーについては、「*CICS® ファミリー: クライアント / サーバー・プログラミング*」を参照してください。

EPIGateway
端末を CICS に接続する前に、JavaGateway オブジェクトを作成します (CICS Transaction Gateway への接続を開始する)。EPIGateway クラスには、CICS

Transaction Gateway からアクセス可能な CICS サーバーの情報にアクセスするためのメソッドがあるので、このクラスを JavaGateway クラスの代わりに使用することもできます。

```
try {
   EPIGateway eGate = new EPIGateway("local:" ,0);
   int numSystems = eGate.serverCount();
   System.out.println("Number of servers:" + numSystems);
   for (int sysCount = 1; sysCount <= numSystems; sysCount++) {
       System.out.print("ServerName=" + eGate.serverName(sysCount));
       System.out.println("   ServerDesc=" + eGate.serverDesc(sysCount));
   }
}
catch (IOException ioEx) {
    ioEx.printStackTrace();
}
catch (EPIException epiEx) {
    epiEx.printStackTrace();
}
```

次の 2 タイプの端末を確立できます。

基本端末

基本端末では、サーバーの名前のみが必要です。オプションで DeviceType タイプおよび NetName を提供することができます。

拡張端末

拡張端末では、サインオン機能、インストール・タイムアウト、読み取りタイムアウト、データ・ストリーム・コード・ページ (エンコード)、およびセキュリティー・クリデンシャル (ユーザー ID およびパスワード) などの機能が提供されます。

基本端末

基本端末を構成するには、次の 2 つの方法があります。

- デフォルト・コンストラクターの使用
- 基本端末コンストラクターの使用

デフォルト端末コンストラクターの使用

デフォルト・コンストラクターを使用して端末を作成するには、最初に端末をインスタンス化し、適切な setter メソッドを使用して必要なプロパティーを設定します。基本端末に適用される setter のみを使用します。これらのメソッドは、次のとおりです。

- setGateway
- setServerName
- setDeviceType
- setNetName

- setSession

setGateway を除くすべてのプロパティーはオプションであり、そのデフォルト設定はヌルです。端末を定義した後、connect() メソッドを使用してこれを CICS に接続します。connect() メソッドは、このバージョンのみを使用してください。その他のバージョンの connect メソッドは、拡張端末にのみ使用できます。詳細については、30ページの『CICS への拡張端末の接続』および 32ページの『同期とセッション』を参照してください。

```
try {
    EPIGateway eGate = new EPIGateway("tcp://MyGateway",2006);
    Terminal term = new Terminal();
    term.setGateway(eGate);
    term.setServerName("CICS1");
    term.connect();
}
catch (IOException ioEx) {
    ioEx.printStackTrace();
}
catch (EPIException epiEx) {
    epiEx.printStackTrace();
}
```

基本端末コンストラクター

もう 1 つの方法は、基本端末コンストラクターを使用するものです。これにより、必要なプロパティーがすべて設定され、CICS サーバーに自動的に接続します。

```
try {
    EPIGateway eGate = new EPIGateway("tcp://MyGateway",2006);
    Terminal term = new Terminal(eGate, "CICS1", null, null);
}
catch (IOException ioEx) {
    ioEx.printStackTrace();
}
catch (EPIException epiEx) {
    epiEx.printStackTrace();
}
```

例外: 例に示すように、基本端末の構成にどのメソッドを使用した場合でも、例外が catch される必要があります。

もう 1 つのオプションの setter として setSession があります。セッションについては、32ページの『同期とセッション』を参照してください。

拡張端末

拡張端末を構成するには、次の 2 つの方法があります。

- デフォルト・コンストラクターの使用
- 拡張端末コンストラクターの使用

デフォルト・コンストラクター

デフォルト・コンストラクターを使用して端末を作成するには、最初に端末をインスタンス化し、そのオブジェクトで適切な setter メソッドを使用します。基本端末の場合、setGateway が必須であり、setDeviceType、setNetName、および setServer はオプションです。拡張端末のプロパティーを定義するには、次の setter が使用されます。これらの setter を使用することは、拡張端末を使用することを暗黙に指定します。

- setSignonCapability (デフォルト = サインオン可能)
- setUserid (デフォルト = ヌル)
- setPassword (デフォルト = ヌル)
- setReadTimeout (デフォルト = 0)
- setEncoding (デフォルト = ヌル)
- setInstallTimeout (デフォルト = 0)

```
try {
    EPIGateway eGate = new EPIGateway("tcp://MyGateway",2006);
    Terminal term = new Terminal();
    term.setGateway(eGate);
    Term.setServerName("CICS1");
    term.setSignonCapability(Terminal.EPI_SIGNON_INCAPABLE);
    term.setUserid(userid);
    term.setPassword(password);
    term.connect();
}
catch (IOException ioEx) {
    ioEx.printStackTrace();
}
catch (EPIException epiEx) {
    epiEx.printStackTrace();
}
```

端末を定義した後、これを CICS に接続できます (30ページの『CICS への拡張端末の接続』を参照)。

拡張端末コンストラクター

拡張端末コンストラクターは、構成時に、すべての必要なプロパティーの設定を行います。

```
try {
    EPIGateway eGate = new EPIGateway("tcp://MyGateway",2006);
    Terminal term = new Terminal(eGate, "CICS1", null, null,
                                 Terminal.EPI_SIGNON_INCAPABLE, userid, password,
                                 0, null);
    term.connect();
}
catch (IOException ioEx) {
    ioEx.printStackTrace();
}
catch (EPIException epiEx) {
    epiEx.printStackTrace();
}
```

基本端末コンストラクターと異なり、拡張端末コンストラクターは、CICS に自動的には接続しません。これは、次のいずれかのメソッドにより明示的に行う必要があります。

CICS への拡張端末の接続: 使用可能な connect メソッドは、次のとおりです。

Connect()

このメソッドは、セッション・プロパティーおよびインストール・タイムアウト・プロパティーを使用して、 CICS へ端末を接続します。

Connect(installTimeout)

このメソッドは、セッション・プロパティーを使用して CICS へ端末を接続しますが、インストール・タイムアウト・プロパティーを提供のものに更新します。

Connection(Session, installTimeout)

このメソッドは、現行のセッション・プロパティーを提供されたセッション・オブジェクトに、また、インストール・タイムアウト・プロパティーを提供のものに更新して、CICS へ端末を接続します。セッションについては、32ページの『同期とセッション』を参照してください。

send の使用によるトランザクションの開始および対話

CICS への端末を確立した後、 send メソッドを使用して新規のトランザクションを開始するか、CICS に画面を送信します。

```
try {
    term.send("EP01",null);
}
catch (EPIException ex) {
    ex.printStackTrace();
}
```

独自の画面を作成して、トランザクションを開始することもできます。画面の操作とフィールドについては、『CICS 3270 画面のフィールドへのアクセス』を参照してください。次に、Screen オブジェクトおよび Field オブジェクトを使用してトランザクションを開始するもう一つの方法を示します。

```
try {
    Screen scr = term.getScreen();
    Field fld = scr.field(1);
    fld.setText("EP01");
    term.send();
}
catch (EPIException ex) {
    ex.printStackTrace();
}
```

CICS 3270 画面のフィールドへのアクセス

端末から CICS への接続を確立した後、 Terminal、Screen、Field の各オブジェクトを使用して、 CICS サーバー・アプリケーションが表示する画面間をナビゲートしたり、必要に応じて、画面データを読み取ったり更新したりします。

Screen オブジェクトは、 Terminal オブジェクトから作成し、 Terminal オブジェクトの getScreen メソッドによって取得します。 Screen オブジェクトには、 3270 画面 (カーソル位置など) に関する一般的な情報を取得したり、個々のフィールドに (画面内の行 / 列の位置や索引によって) アクセスしたりするためのメソッドがあります。以下の例では、フィールドの内容を印刷してから、 PF3 を戻して CESN トランザクションを終了します。

```
// Get access to the Screen object
Screen screen = terminal.getScreen();

        for ( int i=1; i <= screen.fieldCount(); i++ ) {
    Field field = screen.field(i); // get field by index
    if ( field.textLength() > 0 )
        System.out.println( "Field " + i + ": " + field.getText() );
}

// Return PF3 to CICS
        screen.setAID( AID.PF3 );
terminal.send();

// Disconnect the terminal from CICS
terminal.disconnect();
```

Field クラスは、3270 の個々のフィールドのテキストと属性へのアクセスを提供します。このクラスをいろいろな方法で使用して、3270 画面の情報を見つけたり操作したりすることができます。

```
for ( int i=1; i <= screen.fieldCount(); i++ ) {
    Field field = screen.field(i); // get field by index

    // Find unprotected (i.e. input) fields
    if ( field.inputProt() == Field.unprotect )
        ...
    // Find fields the same as a specific text string
    if ( field.getText().equals( "CICS Sign-on") )
        ...
    // Find red fields
    if ( field.foregroundColor() == Field.red )
        ...
}
```

同期とセッション

Terminal クラスは、CICS サーバーへの同期送信および非同期送信の両方をサポートします。同期送信は、すべての情報がサーバーから受信されるまでブロックします。非同期送信は、メソッドからただちに出されますが、応答は引き続きバックグラウンドで処理されます。サーバーから情報が受信される間に、Screen オブジェクトが更新されます。

同期モードを選択するには、setSession メソッドを使用してセッションにヌルを指定するか、send を呼び出す際にヌル・セッションを指定します。あるいは、Session インターフェースを実装して、これを同期セッションに指定することができます。

非同期モードを選択するには、Session インターフェースを実装して、これを非同期セッションに指定します。

Session インターフェースの実装

Session インターフェースは、実装する必要がある 2 つのメソッド getSyncType および handleReply を定義します。次のコードは、サンプルの実装を示します。

```
import com.ibm.ctg.epi.*;
public class myASession implements Session {
    public int getSyncType() {
        return Session.async;
    }
    public void handleReply(TerminalInterface term) {
        System.out.println("Reply received Terminal state = " + term.getState());
    }
    public void handleException(TerminalInterface a, Exception e) {
       System.out.println("Exception received:" + e.getMessage());
    }
}
```

この例では、getSyncType 呼び出しで Session.async を戻しているため、非同期セッションとしてセッションを定義しています。セッションを同期セッションとするには、Session.sync を戻します。

例では、実装が必要な handleReply メソッドを示しています (『handleReply メソッド』を参照)。また、handleException メソッドも示しています。このメソッドは、インターフェースには定義されておらず、オプションです。このメソッドが Session オブジェクト内にあれば、Terminal クラスがこれを呼び出します。 handleException メソッドについては、35ページの『例外処理』を参照してください。

handleReply メソッド

handleReply メソッドは、CICS から受け取る伝送ごとに呼び出されます。CICS サーバー・プログラムの設計によっては、Terminal の send 呼び出しの応答は、1 つまたは複数になります。Terminal の状態プロパティーは、サーバーが応答の送信を終えたかどうかを示します。

Terminal.server

CICS サーバー・プログラムがまだ実行中であり、送信するデータが残っていることを示します。クライアント・アプリケーションからは、現在の画面内容をすぐに処理することもできますし、残っている応答を待つこともできます。アプリケーションで、端末からの接続を終了したり、画面を CICS に送信したり、新しいトランザクションを開始したりすることはできません。

Terminal.client

CICS サーバー・プログラムが今のところ応答を待っていることを示します。クライアント・アプリケーションによって、画面内容が処理されて応答が送信されるはずです。アプリケーションで、端末からの接続を終了したり、新しいトランザクションを開始したりすることはできません。

Terminal.idle

CICS サーバー・プログラムの処理が終了したことを示します。クライアント・プログラムによって、画面内容が処理され、端末との接続が終了するか、あるいはさらにトランザクションが開始されるはずです。ほとんどのクライアント・アプリケーションでは、 CICS サーバー・プログラムのデータ送信が終了する (つまり、 Terminal の状態が client か idle になる) まで待ってから、画面を処理したほうがよいでしょう。ただし、実行に時間がかかるサーバー・プログラムの場合は、状態プロパティーがまだ server のときでも使用できる中間結果や進行状況を送信することもできます。

Terminal.failed
トランザクションが何らかの理由により (たとえば、変換トランザクションがアプリケーションからの応答を待ってタイムアウトになるなど)、開始または完了に失敗したことを示します。詳細については、endReason および endReasonString メソッドを呼び出します。

Terminal.discon
端末が切断されたことを示します。詳細については、endReason および endReasonString メソッドを呼び出します。

Terminal.error
端末がエラー状態にあり、使用できないことを示します。端末を切断して、すべてのリソースを終結処理したことを確認してください。

handleReply メソッドの実装システムでは、 Screen オブジェクトのデータを読み取ったり処理したり、必要に応じてフィールドを更新したり、 CICS への戻り伝送の準備としてカーソル位置や AID 鍵を設定したりしてから、 Terminal の send メソッドを使用して、サーバー・アプリケーションを起動することができます。

同期モードでは、handleReply は、send を呼び出したスレッドと同じスレッドで実行されます。非同期モードでは、handleReply は、別のスレッドで実行されます。

重要
handleReply が端末を切断することのないようにしてください。
handleReply から切断を呼び出した場合、アプリケーションがハングする場合があります。

セッションの使用
setSession メソッドを使用する、あるいは send または connect メソッドの一部として Session オブジェクトを渡すことにより、端末にセッションを設定できます。「ヌル」をセッションとして使用することもでき、この場合、応答および例外の代わりとなる listen オブジェクトは存在せず、すべての呼び出しは同期となります。

ATI および読み取りタイムアウト
ATI イベントおよび読み取りタイムアウト・イベントは非同期であり、ATI が使用可能で、拡張端末の作成時に読み取りタイムアウト値が指定されていれば、アプリケーションの実行中いつでも発生させることが可能です。これらの

機能を使用する場合は、非同期セッションを使用することをお勧めします。ただし、これらの機能は、同期セッションで使用することができます。この場合、ブロックされている間にイベントが発生すると、handleReply は、send または disconnect を呼び出したスレッドで実行されます。send または disconnect の呼び出し内にないアプリケーションの場合、handleReply は別のスレッドで実行されます。

例外処理

例外は、端末との対話において発生します。例外の階層を次に示します。

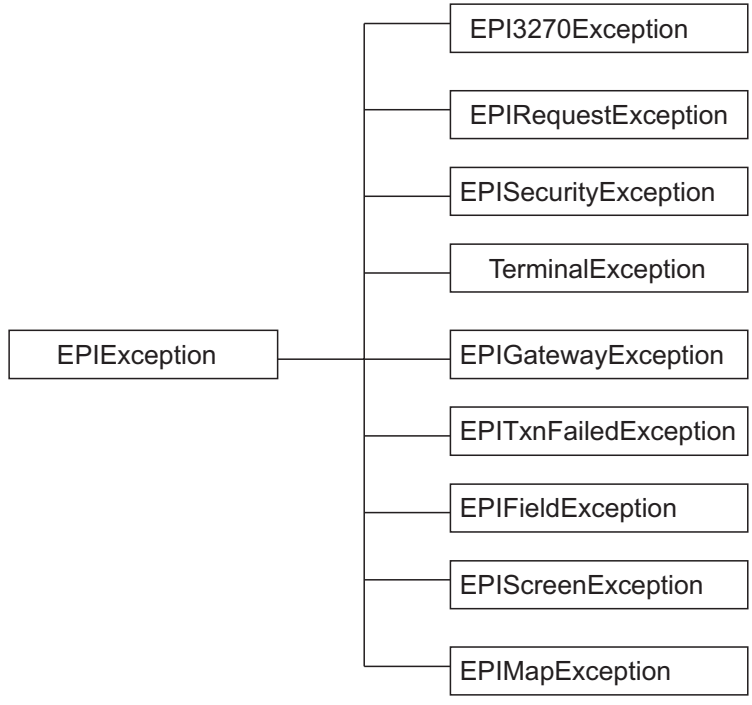

図 1. 例外の階層

各例外の説明は、23ページの『EPI サポート・クラスの概要』に記載されています。

発生するその他のタイプの例外として、IOException があります。

例外では、例外を識別する固有のエラー・コードを検索するためのフィールドが提供されます。このためのメソッドは、getErrorCode() です。

ヌル・セッションまたは同期セッションを使用する場合で、ATI が使用可能でなく、読み取りタイムアウトを使用していなければ、すべての例外はアプリケーションのスレッドで throw されます。connect、send、または disconnect などのメソッドを呼び出す場合は、try/catch/finally タイプのブロックで呼び出しをラップしてください。

ただし、同期セッションを使用する場合、ATI または読み取りタイムアウト (あるいはその両方) を使用可能にすると、問題が発生します。この場合、connect/send/disconnect メソッドが呼び出されている間に例外が発生する場合がありますが、これらの呼び出し外においても発生します。

非同期セッションを使用する場合、例外はアプリケーション・スレッドには throw されません。ATI または読み取りタイムアウト (あるいはその両方) を使用可能にした場合、非同期セッションを使用することをお勧めします。

Terminal メソッドを呼び出している間以外に、いつ例外が発生したかを知るには、セッションに handleException メソッドを実装します (32ページの『同期とセッション』を参照)。これは、同期および非同期セッションの両方で実装できます。端末がアプリケーション・スレッドで例外を throw できなければ (つまり、同期呼び出しでブロックされていない、または非同期セッションである)、このメソッドは別のスレッドで呼び出され、これに例外が渡されます。

端末の切断

端末の切断に問題のないこと、およびアプリケーションの終了前にすべての端末が切断されていることを必ず確認してください。これにより、すべてのリソースの終結処理が確実になります。切断には、次の 2 種類があります。

- PurgeOnDisconnect プロパティーが false に設定される (デフォルト)
- PurgeOnDisconnect プロパティーが true に設定される

除去を行うことにより、イベントが未解決またはトランザクションが実行中の場合でも、端末の切断が試みられます。切断時に除去を行うには、プロパティーを true に設定します。

```
term.setPurgeOnDisconnect(true);
term.disconnect();
```

切断が正常に行われた場合、いずれかの connect() メソッドを発行して、Terminal オブジェクトを再接続できます。

```
term.disconnect();
.....
term.connect();
```

Session は、切断呼び出しには適用されません。切断はすべて同期と見なされます。

拡張端末のエンコード・プロパティー

3270 データ・ストリームを構成するエンコードを指定することができます。端末が接続されると、CICS サーバー (EPI バージョン 2 をサポートする場合) に、3270 データ・ストリームに適用されるエンコード方式が通知されます。ヌルを指定した場合は、CICS Transaction Gateway サーバーの使用するエンコード方式が使用されます (またはローカル・ゲートウェイが使用されている場合は、アプリケーションのデフォルトのエンコード)。

基本端末は常に、CICS Transaction Gateway サーバーの使用するエンコード方式 (またはローカル・ゲートウェイが使用されている場合は、アプリケーションのデフォルトのエンコード) を使用します。

73ページの『付録A. Java エンコード』に、EPI サポート・クラスにより行われる変換のリストが記載されています。

サポートされるコード・ページの詳細については、CICS サーバーの資料を参照してください。

BMS マップの変換と Map クラスの使用

既存の CICS アプリケーションのほとんどは、3270 画面出力のために BMS マップを使用しています。したがって、サーバー・アプリケーションは、3270 データ・ストリームを直接処理する代わりに、BMS マップの名前付きフィールドに対応するデータ構造を使用することができます。EPI BMS 変換ユーティリティーは、BMS マップ・ソースの情報を使用して、個々のマップに固有のクラスを生成するので、フィールドにはそれぞれの名前でアクセスできます。

このユーティリティーは、アプリケーションが、マップ・オブジェクト内の名前付きフィールドとしてマップ・データにアクセスするための Java クラスを生成します。それぞれのマップに 1 つのクラスを定義するので、フィールドの

名前と長さがコンパイル時に認識されるようになります。生成されるクラスは、すべてのマップ・クラスに必要な一般機能を提供する、**Map** クラスを展開したものです。

次のように、BMS ソースで BMS マップ変換ユーティリティーを実行します。

```
java com.ibm.ctg.epi.BMSMapConvert -p package -name filename.BMS
```

このユーティリティーによって、マップ・クラスのソースを収容した .java ファイルが生成されます。-p パラメーターを使用して、新しいファイルを収めるパッケージを指定します。その場合は、ファイルを編集して ″package″ ステートメントを追加する必要はありません。

EPI BMS ユーティリティーを使用してマップ・クラスを生成した後、基本 EPI クラスを使用し、通常の方法で、該当する 3270 画面にアクセスします。その画面では、マップ・クラスを使用して、BMS マップにある名前で各フィールドにアクセスできます。マップ・クラスは、現在の Screen オブジェクトにあるデータに照らして、妥当性検査が行われます。

マップ・クラスの使用

BMS 変換ユーティリティーで生成されるクラスには、以下のような特色があります。

- クラス名は、BMS ソースにあるマップ名から派生します。
- クラスは、Map を展開したものです。
- 2 つのコンストラクターがあります。1 つのコンストラクターは、関連 BMS マップによって画面がまだ生成されていない場合に、Screen パラメーターを取って、EPIException を throw します。この引き数のないコンストラクターは、Map を作成します。この Map は、setScreen メソッドによって、後から画面に照らして妥当性検査を行うことができます。
- メソッド・フィールドは、BMS ソースのフィールド名 (クラス内部の定数として設定) を使用して、マップ内のフィールドへのアクセスを提供します。

生成された Map クラスを使用するには、通常の方法で Terminal を作成し、トランザクションを開始します。

```
try {
   EPIGateway epi = new EPIGateway("jgate", 2006 );
   // Connect to CICS server
   Terminal terminal = new Terminal( epi, "CICS1234", null, null );
   // Start transaction on CICS server
   terminal.send( null, "EPIC", null );
```

以下の例では、サーバー・プログラムで、マップ・クラス ″MAPINQ1Map″ が生成されている、最初のパネル用に BMS マップを使用しています。マップ・オブジェクトが作成されると、コンストラクターが、マップ内に定義されているフィールドに照らして、画面内容の妥当性検査を行います。妥当性検査に合格すると、各フィールドに対して、Screen オブジェクトの索引や位置によってではなく、BMS フィールド名でアクセスできるようになります。

```
            MAPINQ1Map map = new MAPINQ1Map( terminal.getScreen() );
            Field field;
            // Output text from "PRODNAM" field
            field = map.field(MAPINQ1Map.PRODNAM);
            System.out.println( "Product Name: " + field.getText() );
            // Output text from "APPLID" field
            field = map.field(MAPINQ1Map.APPLID);
            System.out.println( "Applid : " + field.getText() );
        } catch (Exception exception) {
            exception.printStackTrace();
        }
```

BMS Map オブジェクトは、Session の handleReply メソッド内部でも使用できます。

妥当性検査に合格するには、現在の画面で BMS マップ全体が使用できるようになっている必要があります。したがって、BMS マップの一部または全体が、別のマップや個々の 3270 フィールドによってオーバーレイされている場合は、マップ・クラスを使用できません。

Terminal なしで Screen を使用する

Screen クラスと Field クラスを使用して、 3270 データ・ストリームを解釈する場合に、 EPIRequest クラスをそのまま使用したければ、そのようにしても構いません。

1. 通常の方法で CICS とのセッションを確立し、必要なサイズの Screen オブジェクトを作成します。
2. Screen の analyze メソッドにデータ・ストリームを渡します。
3. この時点で、画面のフィールドへアクセスしたり、戻す AID を設定したりできます。
4. CICS に戻すデータ・ストリームを準備するには、 Screen の format メソッドを呼び出します。

第5章 EPI beans

この章では、EPI 関数で使用できる Bean について説明します。

CICS Transaction Gateway EPI beans の概要

CICS Transaction Gateway には、既存の CICS® 3270 アプリケーションからデータにアクセスする Java プログラムを簡単に作成するために使用できる、高水準の EPI クラスが組み込まれています。

しかし、EPI beans は、この機能を元にして、さらに優れた機能を提供します。すなわち、既存の CICS® 3270 画面にアクセスするアプリケーションを、まったくプログラミングを行うことなく 作成できるということです。VisualAge® for Java™ を含め、新しいビジュアル・アプリケーション・ビルダー・ツールが多数ありますが、そのいずれかを使うことにより、CICS® への接続、トランザクションの実行、3270 画面からのデータの表示、ユーザー入力のサーバーへの送信を実行できる、新しい Java フロントエンドをす早く容易に作成できます。EPI Bean は、3270 データ・ストリームの 2 バイト文字セット (DBCS) フィールドをサポートしません。

現在、Java beans を使用して Java アプリケーションを作成するためのビジュアル開発ツールは多数あります。

beans をビジュアル開発ツールで使用するためには、ツールに **ctgclient.jar** ファイルをインポートまたはロードしなければならない場合があります。その方法については、ツールの資料を参照してください。あるいは、上記のファイルを CLASSPATH 設定に含めなければならない場合もあります。

VisualAge® for Java™ をご使用の場合は、47ページの『VisualAge® for Java™ 入門』の情報が役立ちます。

beans の使用

beans を適切なビジュアル開発ツールにロードした後、beans を使用するアプリケーションの作成に着手できます。

CICS® への接続

アプリケーションでは、まず CICS Transaction Gateway に接続しなければ、ほかの作業を何も行うことができません。接続するには、EPITerminal bean が必

要です。Bean のプロパティーを設定して、接続先のゲートウェイの URL を指定します。図2 は、Bean のプロパティー・シートの例を示しています。

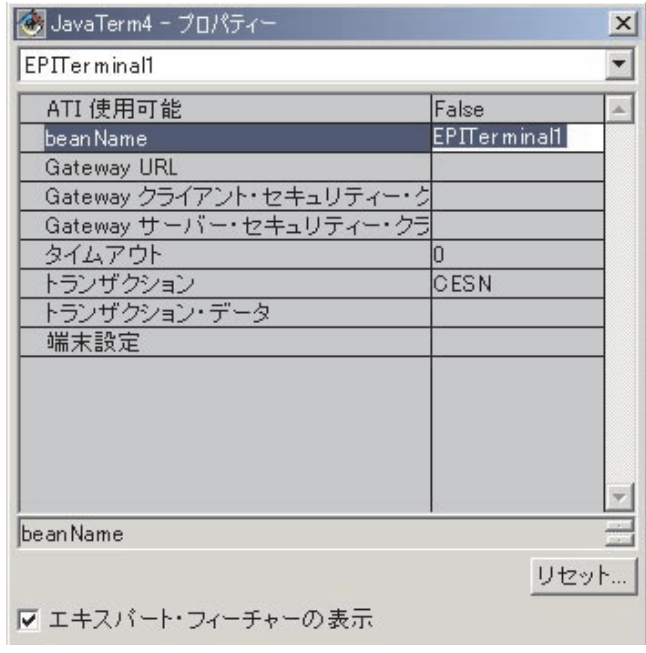

図 2. 端末 Bean のプロパティー・シート

接続先の CICS® など、必要な端末固有のプロパティーを定義します。43ページの図3 の端末設定値プロパティー・シートの例を参照してください。各種プロパティーの詳細については、Bean 解説のセクションを参照してください。CICS® に接続するには、EPITerminal の **connect** メソッドを呼び出します。

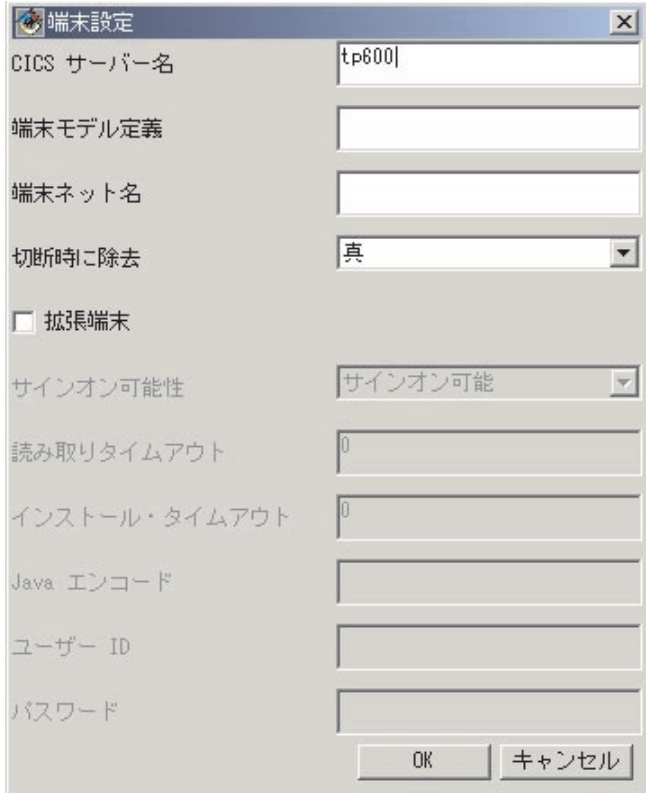

図 3. 端末設定プロパティー・シート

トランザクションの開始

トランザクションを開始するには、 EPITerminal の **startTran** メソッドを呼び出すことができます。あらかじめ、開始するトランザクション ID を transaction プロパティーに設定しておきます。VisualAge® for Java™ などの一部のツールでは、イベントが生じたときに、定義済みのパラメーターを指定して **setTransaction** メソッドを呼び出すことができます。あるいは、開始するトランザクション ID をアクション・コマンドに指定して、 ActionEvent を EPITerminal に送信することもできます。transaction プロパティーがその値に設定され、可能になり次第、トランザクションが開始します。ActionEvents は、ボタン、メニュー項目、および入力フィールド (**「Enter」**を選択した場合、テキストがアクション・コマンドになります。) によって生成されます。

画面の処理

CICS® サーバーでトランザクションが開始されると、CICS® は EPITerminal bean にデータを送り返し始めます。CICS® からデータを受け取ると、EPITerminal は登録済みのすべてのイベント・リスナーに handleScreen イベントを送ります。CICS® から送信されるデータを処理するには、画面ハンドラーという beans を作成し、これを EPITerminal の handleScreen イベントに接続します。

EPIBasicScreenHandler bean は、画面ハンドラーの一例です。これは、データをテキストおよび入力フィールドとして表示します。EPIBasicScreenHandler のプロパティーの詳細については、Bean 解説のセクションを参照してください。図4 は、EPIBSH プロパティー・シートの例を示しています。

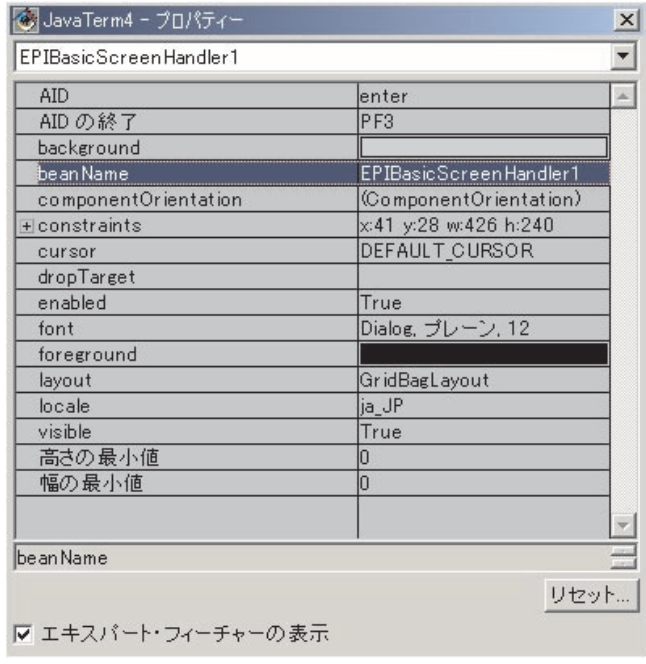

図 4. EPIBasicScreenHandler プロパティー・シート

特定の画面を異なる方法で処理するには、それに合った画面ハンドラーを作成します。自分独自のハンドラーを作成することもできますし、提供されているツールの 1 つを使用して、自動的に画面ハンドラー beans を生成することもできます。自動生成された画面ハンドラー beans を使用すると、画面のデータの一部に、バウンド・プロパティーの形式でアクセスできるようになります。この場合は、それらのプロパティーを表示および変更できるよう、独自のユーザー・インターフェースを (他の beans を使用して) 作成してください。

CICS® へのデータの返信

CICS® から画面を受け取って、これを表示し、必要に応じて編集を加えた場合、画面を CICS® に送り返さないと処理を続行できません。これは、EPIBasicScreenhandler、または自動的に生成された bean を使用している場合、次のことを意味します。

- 画面ハンドラーの AID プロパティーを必要な AID 値に設定する
- 画面ハンドラーまたは EPITerminal の send メソッドを呼び出す

同じ結果を得るための別の方法は、アクション・コマンドに AID 値を設定して (例: enter または PF3)、画面ハンドラーに ActionEvent を送信するというものです。そうすると、AID がその値に設定され、画面が CICS® に送信されます。ActionEvents は、ボタン、メニュー項目、および入力フィールドにより生成されます。

CICS® からの切断

アプリケーションが終了した後、EPITerminal の **disconnect** メソッドを呼び出して、CICS® から切断するようにしなければなりません。**disconnect** を呼び出す場合、端末がアイドル状態 (トランザクションが実行されていない状態) 以外の場合は、EPITerminal は登録済みのイベント・リスナーに handleScreen イベントを送信し、実行中のトランザクションをすべて終了するように要求します。EPIBasicScreenHandler および生成された画面ハンドラーのデフォルトの動作は、AID PF3 を CICS® サーバーに送信するというものです。これが自分の使用する特定のトランザクションで機能しないという場合は、関係する画面に合った画面ハンドラーを生成し、この画面を確実に終了できるよう、ハンドラーをカスタマイズする必要があります。切断時の除去を指定していない場合、切断は、トランザクションの終了時にのみ行われます。

例

この例は、beans の使用に関係する基本的な原則を示したものです。

1. 好みの bean 作成ツールを使用して EPITerminal を作成し、そのプロパティーを調べます。URL プロパティーを Gateway のアドレスに設定します (例: tcp://jbloggs.hursley.ibm.com:2006)。terminal settings プロパティーを使用して、さらに詳細な端末設定を行えます (CICS® サーバーの名前など)。
2. 「**Connect**」および「**Disconnect**」というラベルの付いた、 2 つのボタンを作成します。「**Connect**」ボタンの actionPerformed イベントを、EPITerminal の **connect** メソッドに接続します。同様に、「**Disconnect**」ボタンを EPITerminal の **disconnect** メソッドに接続します。これで、CICS® サーバーへの接続と切断ができるようになります。

3. EPIBasicScreenHandler を作成します。これは、CICS® から送信されたデータを表示できる、表示パーツです。EPITerminal の handleScreen イベントを、EPIBasicScreenHandler の handleScreen メソッドに接続します。 イベント・データが EPIBasicScreenHandler に渡されることを確かめる必要があります。
4. 「**Connect**」ボタンの actionPerformed イベントを、EPITerminal の **startTran** メソッドに接続します。EPITerminal のプロパティーで、transaction プロパティーが、CICS® サーバーで実行可能なトランザクションに設定されていることを確かめます。
5. EPIScreenButtons オブジェクトを作成します。これは、端末を使用するために使用できる、単純なボタンのセットです。EPIScreenButtons の actionPerformed イベントを、EPIBasicScreenHandler の actionPerformed メソッドに接続します。使用するツールによっては、イベント・データが EPIBasicScreenHandler に渡されることを確かめる必要があります。
6. これで、CICS® に接続してトランザクションを実行できるようになったはずです。EPIBasicScreenHandler は CICS® から送信されたデータを表示し、ユーザー入力を受け入れます。ボタンを 1 つ選択すると、画面が CICS® に送り返されます。テキスト・フィールドに入力フォーカスが合っている時に 「**Enter**」を選択した場合にも、画面が送信されます。処理が完了した後、端末を切断するようにします。

画面ハンドラー beans の生成

BMS 定義ファイルから画面ハンドラー beans を自動的に生成できます。49ページの『画面ハンドラー bean』を参照してください。こうして生成された画面ハンドラー beans は、対応する BMS マップ内のラベル付きフィールドのそれぞれにバウンド・プロパティーを定義し、CICS® からのデータとユーザー・インターフェース・コンポーネントの間で容易にマップが行えるようにします。

生成された画面ハンドラーを使用するには、上述のとおりに、それらのハンドラーを EPITerminal の handleScreen イベントに接続します。EPITerminal へは、いくつでも画面ハンドラーを接続できます。画面ハンドラーは、CICS® から新しい画面を受信すると、この画面が対応する BMS マップを表示しているのかどうかを判別します。画面ハンドラーは、画面を認識すると、**screenHandled** イベントを送信します。さらに、CICS® から送られた新しいデータに合わせて、バウンド・プロパティーの更新も行います。

VisualAge® for Java™ 入門

このセクションは、VisualAge® for Java™ の使用者の手引きというわけではありません。初めて VisualAge® for Java™ をご使用になる方にとって有益な情報が記載されてはいますが、 VisualAge® for Java™ の資料も合わせて参照してください。

バージョン: 記載されている例は、VisualAge® for Java™ バージョン 2.0 用です。他のバージョンの VisualAge® for Java™ では、異なる場合もあります。

CICS Transaction Gateway クラスを VisualAge® for Java™ にインポートする

CICS Transaction Gateway クラスを VisualAge® for Java™ で使用するには、最初に、これらをリポジトリーにインポートします。ワークベンチのメニューで、**「ファイル (File)」**–**「インポート (Import)」**と選択します。プロジェクト名を入力し
(例: `CICS Transaction Gateway`)、**「JAR ファイル (JAR file)」**を選択してから、**「次へ > (Next >)」**ボタンを選択します。CICS Transaction Gateway の **classes** サブディレクトリーでファイル **ctgclient.jar** を選択し、これをインポートします。

VisualAge® for Java™ がエラーを報告することがありますが、無視して差し支えありません。

CICS® Transaction Gateway v4.0 クラスを Visual Age for Java v3.0 にインポートする場合、com.ibm.sslight.* を含むパッケージが置換されることを示す警告が出る場合があります。これは、CICS® Transaction Gateway に同梱の SSLight (SSL) レベルが新しいためです。Visual Age for Java で開発を行う場合、このレベルの com.ibm.sslight がクライアント・アプリケーションで使用できることを確認してください。

ファイルが VisualAge にインポートされると、「Add to Palette」というダイアログが表示されます。リストされているクラスを全部選択し、**「OK」**を選択します。新しいカテゴリー・フォルダーの名前を入力するよう、プロンプトが出されます。`CICS Transaction Gateway EPI beans` などと入力できます。
「OK」を選択して、先に進みます。

ヒント

このセクションでは、 EPI beans をさらに活用する上で役に立つヒントをいくつか紹介します。

アプリケーションが機能せず、一部の接続が破線で表示される場合

何かが起こるようにするために (たとえば、ボタンが選択された時に CICS® に接続するなど)、単純にイベントをメソッドに接続することができます。しかし、2 つの bean の間で情報をやり取りするためにイベントを使用する場合もあります。たとえば、EPITerminal の **handleScreen** イベントは、CICS® から受け取った画面データを EPIBasicScreenHandler (または MAPINQ1ScreenHandler) に渡します。イベントをメソッドに接続すると、VisualAge® for Java™ は、イベント・データをメソッドに渡すことは意図されていないものと見なします。メソッドにデータを渡すことが可能である場合、接続は破線で表示されます。VisualAge® for Java™ がイベント・データをメソッドに渡すようにするには、接続を右マウス・ボタン・クリックして、**「Properties」**を選択します。「Event-to-Method Properties」ダイアログで、**「Pass event data」**というラベルの付いたチェック・ボックスを選択します。これで、接続が実線で表示されるようになるはずです。

入力フィールドに Screen Handler プロパティーを設定する方法

入力フィールドの **text** プロパティーはバウンド・プロパティーではないため、入力フィールドに Screen Handler プロパティーを接続することはできません。その代わりに、フィールドを設定する際には、画面ハンドラー上で **setXXX** メソッドを呼び出す必要があります。トリガーとして、**actionPerformed** イベントを (ボタンから、あるいは入力フィールド自体から) 使用してください。そして、**actionPerformed** イベントを画面ハンドラーの **setXXX** メソッドに接続します。これで、今作成した**接続に**、入力フィールドの **text** プロパティーを接続できるようになりました。項目が 1 つ含まれたポップアップ・メニューが表示されます。その項目は、メソッド **setXXX** のパラメーターです。これを選択すると、入力フィールドから接続へのリンクが存在しているはずです。

生成された画面ハンドラー beans をビジュアル・コンポジション・エディターに追加する方法

まず、生成されたファイルを、Map クラスと ScreenHandler クラスの両方とも、VisualAge® for Java™ にインポートします。ファイルは、パッケージの中に作成されたものでなければなりません (BMSMapConvert の -p パラメーターを使用)。そうでない場合は、ファイルをパッケージ (Default パッケージではない) にコピーしてください。次に、ビジュアル・コンポジション・エディターのメニューで、**「Options」**–**「Add bean」**と選択します。beans の名前の最初の数文字を入力してください (パッケージ名を指定する必要はありません)。**「Browse」**を選択し、クラスのリストから新しい bean のクラスを選択します。**「OK」**を選択すると、bean を表示域にドロップできるようになります。

画面ハンドラー bean

画面ハンドラー bean は、以下のことが可能な Java クラスです。

- CICS® の画面を認識する
- 画面の情報を bean のプロパティーとして使用可能にする
- 画面を終了する

画面ハンドラーを EPI beans と一緒に使用することにより、既存の CICS® 3270 アプリケーションへの Java フロントエンドを作成できます。また、画面ハンドラーを CICS Transaction Gateway 端末サーブレットから使用することにより、トランザクションを終了すること、および画面上のフィールドに記号名でアクセスすることができます。

画面ハンドラー・クラスを自分で作成する必要はありません。BMS マップ定義から自動的に生成できます。

画面ハンドラー bean の生成

画面ハンドラー・クラスを生成するには、 CICS Transaction Gateway に付属の BMS マップ変換ユーティリティーを使用します。たとえば、TEST1.BMS という BMS マップ・ファイルによって定義された画面用の bean を生成するには、次のコマンドを使用します。

```
java com.ibm.ctg.epi.BMSMapConvert -b test1.bms
```

BMS マップ変換ツールに指定できるパラメーターは次のとおりです (順不同)。

-b 画面ハンドラー beans を生成します。デフォルトでは、beans を生成しません。

-k *exit* 画面ハンドラーが画面を終了するために使用する鍵を設定します (例: `-k clear`)。デフォルトは PF3 です。

-p *package name*
ステートメント `package` *`package name`* を付けて、 Java ソース・コードを生成します (例: `-p testing.beans`)。デフォルトでは、package ステートメントを追加しません。

BMS マップ定義ファイル名
変換する BMS ファイルを選択します。プラットフォーム標準のファイル名の規則を使用してください。ファイル名を完全に指定しない場合、そのファイルのファイル・タイプは .BMS と見なされます。

BMS マップ変換ツールは、BMS ファイルに定義されているマップごとに、次の 2 つの Java ソース・ファイルを生成します。

- Map クラス
- ScreenHandler クラス

ソース・ファイルの名前は、マップ名から受け継がれます。たとえば、MAP1 というマップが BMS ファイルに定義されている場合、MAP1Map.java と MAP1ScreenHandler.java という Java ソース・ファイルが生成されます。

ScreenHandler クラスは Map クラスを使用するので、これらのファイルは一緒に保持しておいてください。

VisualAge® for Java™ にロードするつもりのファイルを生成するときには、パッケージの名前を付けてください。パッケージの名前が付いていないと、ファイルが beans として認識されません。

画面ハンドラーのカスタマイズと作成

以下の場合には、生成された ScreenHandler クラスをカスタマイズします。

- 単に AID (PF3 など) を CICS® に送信するだけでは、画面ハンドラーが自分に関連付けられている画面を終了できない場合。この場合は、必要なことを実行するよう exitScreen メソッドを変更する必要があります。
- 画面ハンドラーが、自分に関連付けられている画面を認識する方法を変更する場合。通常、画面ハンドラーは、BMS マップに定義されているのと同じフィールドが画面にあるかどうかを検査します。

画面ハンドラーを独自に作成する場合は、次の点に注意してください。

- 画面ハンドラーは Java beans として作成する必要があります。通常、これは、引き数をとらないコンストラクターを持たなければならないということと、java.io.Serializable インターフェースを実装しなければならないということを意味します。
- 生成された画面ハンドラーはクラス ScreenHandler を拡張しますが、これは必要ありません。実際、EPI beans と一緒に使用する上で画面ハンドラーに必要なのは、TerminalEvent パラメーターを受け入れるメソッドを備えているということだけです。

  端末サーブレットと一緒に使用する場合、画面ハンドラーは、TerminalEventListener インターフェースを実装しなければなりません。また、画面を認識したという信号を送るために、TerminalEvent の **isHandled** メソッドを使用する必要もあります。

EPI beans のリファレンス

このセクションでは、EPI beans の参照情報を示します。

EPITerminal bean

この bean は、CICS Transaction Gateway 経由で CICS® に接続された 3270 端末として機能します。

この bean を使用するには、次のことが必要です。

- bean のプロパティーを適切な値に設定する。最低でも、CICS Transaction Gateway の URL を設定することが必要です。
- 画面ハンドラー beans を端末の handleScreen イベントに接続する。
- 端末を CICS® に接続するために connect メソッドを呼び出す。
- transaction プロパティーをトランザクション ID に設定してから startTran を呼び出して、トランザクションを開始する。
- アプリケーションを終了する前に、 disconnect または terminate を呼び出して、端末を確実に切断する。

プロパティー

Gateway URL

接続先の CICS Transaction Gateway の URL。たとえば、URL を tcp://buster:2006 に設定すると、端末は TCP/IP アドレス buster、ポート番号 2006 の CICS Transaction Gateway に接続しようとします。

Gateway サーバー・セキュリティー・クラス

CICS Transaction Gateway が使用する、サーバー・セキュリティー・クラス。このプロパティーの使用には、専門的な知識が必要です。

標準端末プロパティー

端末がサーバーに接続している間にこれらの値 (ユーザー ID およびパスワードを除く) を変更しても、影響はありません。異なる設定を使用するには、端末を切断してから再接続する必要があります。

以下は、標準の端末プロパティーです。

- CICS®サーバー名 — 接続先の CICS® サーバー
- 端末モデル定義
- 端末ネット名
- 切断時に除去 — 標準の切断の代わりに除去を強制するには「真」に設定します。

拡張端末プロパティー

端末がサーバーに接続している間にこれらの値 (ユーザー ID およびパスワードを除く) を変更しても、影響はありません。異なる設定を使用するには、端末を切断してから再接続する必要があります。

以下は、拡張端末プロパティーです。

- サインオン可能性 — サインオン可能またはサインオン不能な端末のいずれかを選択します。 CICS® サーバーは、選択されたオプションをサポートする必要があります。
- 読み取りタイムアウト — 有効範囲 0 ～ 3600
- インストール・タイムアウト — 有効範囲 0 ～ 3600
- Java エンコード — 3270 データ・ストリームに使用するエンコード
- ユーザー ID — 端末に関連付けられているユーザー ID
- パスワード — 端末に関連付けられているパスワード

ATI 使用可能

CICS® サーバーが端末上でトランザクションを開始できるようにするには、このプロパティーを **true** に設定します。

トランザクション

トランザクション ID。これは、メソッド startTran が呼び出されたときに開始されるトランザクションです。

トランザクション・データ

トランザクションのパラメーター。startTran が呼び出されると、このストリングがパラメーターとしてトランザクションに渡されます。たとえば、トランザクション ID が CESN で、トランザクション・データが `USERID=fred PS=pswd` に設定されている場合、startTran を呼び出すことの結果は、CICS® 端末で `CESN USERID=fred PS=pswd` と入力することと同じになります。

AutoDisconnectTimeout

端末のタイムアウト・インターバル (ミリ秒単位)。この期間が過ぎても端末のアイドル状態 (CICS® からデータを受信しない状態) が続いていると、端末は CICS® から切断しようとします。端末がタイムアウトにならないようにするためには、この値を 0 に設定することができます。このタイムアウト値には、CICS Transaction Gateway の接続タイムアウト値よりも小さな値を設定しなければなりません。さもないと、端末が CICS® から切断しないうちに、CICS® Gateway for Java への接続が失われてしまう可能性があります。

接続された
端末を CICS® に接続する場合は、このブール値を true に設定します。

イベント

TerminalEvent

terminalConnected
端末が CICS® へ接続されたことを示します。

terminalDisconnected
端末が CICS® から切断されたことを示します。

handleScreen
端末が CICS® から画面を受け取ったことを示します。

exceptionOccurred
例外が起こったことを示します。

メソッド

connect()
端末をサーバーに接続します (まだ接続していない場合)。

startTran()
すでに設定されているトランザクション ID とトランザクション・データを使用して、端末上でトランザクションを開始します。

send() 現在の画面を CICS® に送信します。

disconnect()
サーバーから端末を切断します。トランザクションの実行中に、サーバーから端末を切断することはできません。トランザクションが実行中の場合、端末は、切断する前にそのトランザクションを終了しようとします。terminate メソッドが呼び出されるか、アプリケーションが終了するまで、端末は CICS Transaction Gateway に接続したままです。

terminate()
端末が切断を完了するのを待ってから、CICS Transaction Gateway への接続をクローズします。

actionPerformed(ActionEvent イベント)
トランザクション ID をイベントのアクション・コマンドに設定し、**startTran** メソッドを呼び出します。ボタンを押すことによって (または他の ActionEvent によって) トランザクションを開始するために、使用することができます。

connect(int installTimeout)
端末を CICS に接続します (まだ接続していない場合)。指定されたタイムアウト時間内に端末が接続しない場合、exceptionOccurred イベントが発生します。

getTermid()
接続されている端末の現行の端末 ID を戻します。

getUserid()
端末に関連付けられているユーザー ID を戻します。

getPassword()
端末に関連付けられているパスワードを戻します。

setUserid()
端末のユーザー ID を設定します。サインオン可能な端末であれば、このユーザー ID が次の開始トランザクションに使用されます。

setPassword()
端末のパスワードを設定します。サインオン可能な端末であれば、このパスワードが次の開始トランザクションに使用されます。

verifyPassword()
CICS® サーバーがパスワードの期限管理をサポートしている場合に、端末に関連付けられている現行のユーザー ID およびパスワードを検証します。EPISecurityAttrs インスタンスを戻します。

changePassword(String newPassword)
CICS® サーバーがパスワードの期限管理をサポートしている場合に、端末および CICS® サーバーに関連付けられているパスワードを変更します。EPISecurityAttrs インスタンスを戻します。

Send(String tranid, String tranData)
オプションの tranData を渡して、CICS® サーバーに提供されるトランザクションを開始します。

EPIBasicScreenHandler bean

この bean は、どんな画面でも処理できる、単純な画面ハンドラーです。次のことが可能です。

- CICS® から送信された画面を、 Java のユーザー・インターフェース・コンポーネントを使用して表示する。
- ユーザー入力を受け入れる。
- ユーザーにより変更が加えられた画面を CICS® に送り返す。
- CICS® に送信する AID を設定する。

- 画面のカーソル位置を設定する。

この bean を使用するには、次のことが必要です。

- この bean をユーザー・インターフェースに追加する。この bean は、パネルのように扱うことができます。
- 端末の handleScreen イベントを handleScreen メソッドに接続する。VisualAge を使用する場合は、イベント・データがこのメソッドに渡されるようにする必要があります。

プロパティー

AID CICS に送信される AID 鍵。

AID の終了
実行中のトランザクションを終了するために使用される AID 鍵。

handling
EPIBasicScreenHandler が画面を処理する場合は true。

幅の最小値
画面パネルの最小の幅。

高さの最小値
画面パネルの最小の高さ。

イベント

ScreenEvent

screenHandled
画面ハンドラーが画面の処理を開始したことを示します。

screenUnhandled
画面ハンドラーが画面の処理を停止したことを示します。

メソッド

terminalConnected(TerminalEvent イベント)
端末接続イベントを受け取ったときに、何のアクションも起こしません。

terminalDisconnected(TerminalEvent イベント)
画面をクリアします。

handleScreen(TerminalEvent イベント)
画面を表示します。

actionPerformed(ActionEvent イベント)
イベントの送信元がボタンの場合、可能であればボタン・アクション・コマンドを画面の AID にマップして、画面を CICS® に送信します。イベントの送信元が TextField の場合、関連する画面フィールドを更新し、AID を **enter** に設定して、画面を CICS® に送信します。

send() 画面を CICS® に送信します。

EPIMonitor bean

この bean は、EPITerminal 用の単純な状況モニターです。この bean は、状況メッセージを表示することにより、端末のイベントを処理します。開発中、端末が何を行っているかを把握する上で役立ちます。

この bean を使用するには、次のことが必要です。

- この bean をユーザー・インターフェースに追加する。
- 端末イベント (terminalConnected、terminalDisconnected、 handleScreen、および exceptionOccurred) を適切な EPIMonitor メソッドに接続する。 VisualAge を使用する場合は、イベント・データが渡されるようにする必要があります。

メソッド

terminalConnected(TerminalEvent イベント)
状況メッセージを更新します。

terminalDisconnected(TerminalEvent イベント)
状況メッセージを更新します。

handleScreen(TerminalEvent イベント)
状況メッセージを更新します。

exceptionOccurred(TerminalEvent イベント)
状況メッセージを例外テキストに設定します。

EPIScreenButtons bean

この bean は、 EPIBasicScreenHandler に AID を送信するために使用できるボタンのセットです。各ボタンのアクション・コマンドは AID 値に設定されます。たとえば、「**Enter**」というラベルのボタンのアクション・コマンドは、**enter** です。ボタンが選択されると、EPIScreenButtons bean は、そのボタンのアクション・イベントを listener に転送します。

この bean を使用するには、次のことが必要です。

- この bean をユーザー・インターフェースに追加する。

- actionPerformed イベントを EPIBasicScreenHandlers の actionPerformed メソッドに接続する。 VisualAge® for Java™ を使用する場合は、イベント・データが渡されるようにする必要があります。

ScreenHandler に基づいた画面ハンドラーはどれも、EPIScreenButtons から送られるアクション・イベントに応答しますが、ユーザー独自のボタンを使用することもできます。ボタンに AID 値と同等のアクション・コマンドを指定すると、得られる結果は同じになります。

イベント

ActionEvent

actionPerformed
ボタンが選択された場合です。

メソッド

actionPerformed(ActionEvent イベント)
任意の登録済み listener にイベントを転送します。

第6章 VisualAge® for Java™ の使用

このセクションでは、CICS トランザクションにアクセスする簡単なアプリケーションを、 VisualAge® for Java™ を使用してどのように作成できるかを説明します。以下のことを行う方法が説明されています。

- VisualAge® for Java™ をセットアップして CICS Transaction Gateway クラスを使用する
- VisualAge® for Java™ ビジュアル・コンポジション・エディターを使用して、アプリケーションを作成する
- VisualAge® for Java™ 環境の外部でアプリケーションを使用する

記載されている例は、VisualAge® for Java™ バージョン 2.0 Entry Version 用です。他のバージョンの VisualAge® for Java™ では、異なる場合もあります。

VisualAge® for Java™ と CICS Transaction Gateway クラス

VisualAge® for Java™ を使用して CICS アプリケーションを作成するためには、 CICS Transaction Gateway に付属のクラス・パッケージ ctgclient.jar をインポートする必要があります。このパッケージには、アプリケーションを作成するのに必要な Java クラスと beans が含まれています。

CICS Transaction Gateway クラス・パッケージを VisualAge® for Java™ にインポートする手順は、次のとおりです。

1. VisualAge® for Java™ を開始し、ワークベンチに移動します。
2. **「ファイル (File)」**メニューで**「インポート (Import)」**を選択します。
3. SmartGuide の「インポート (Import)」ダイアログで**「JAR ファイル (JAR file)」**ボタンを選択し、**「次へ (Next)」**を選択します。
4. SmartGuide の「jar/zip からのインポート (import from jar/zip file)」ダイアログで**「ファイル名 (Filename)」**の**「ブラウズ (Browse)」**ボタンを選択し、ctgclient.jar ファイルが入っているディレクトリーを指定します。
5. ctgclient.jar ファイルを選択してから**「オープン (Open)」**を選択し、SmartGuide の「jar/zip からのインポート (import from jar/zip file)」に戻ります。
6. **「.class」**と**「リソース (resource)」**のそれぞれのチェック・ボックスが選択されていることを確認します。

7. **「プロジェクト (Project)」**フィールドに「CICS Transaction Gateway」と入力します。 図5 に Smartguide によるインポートを示します。

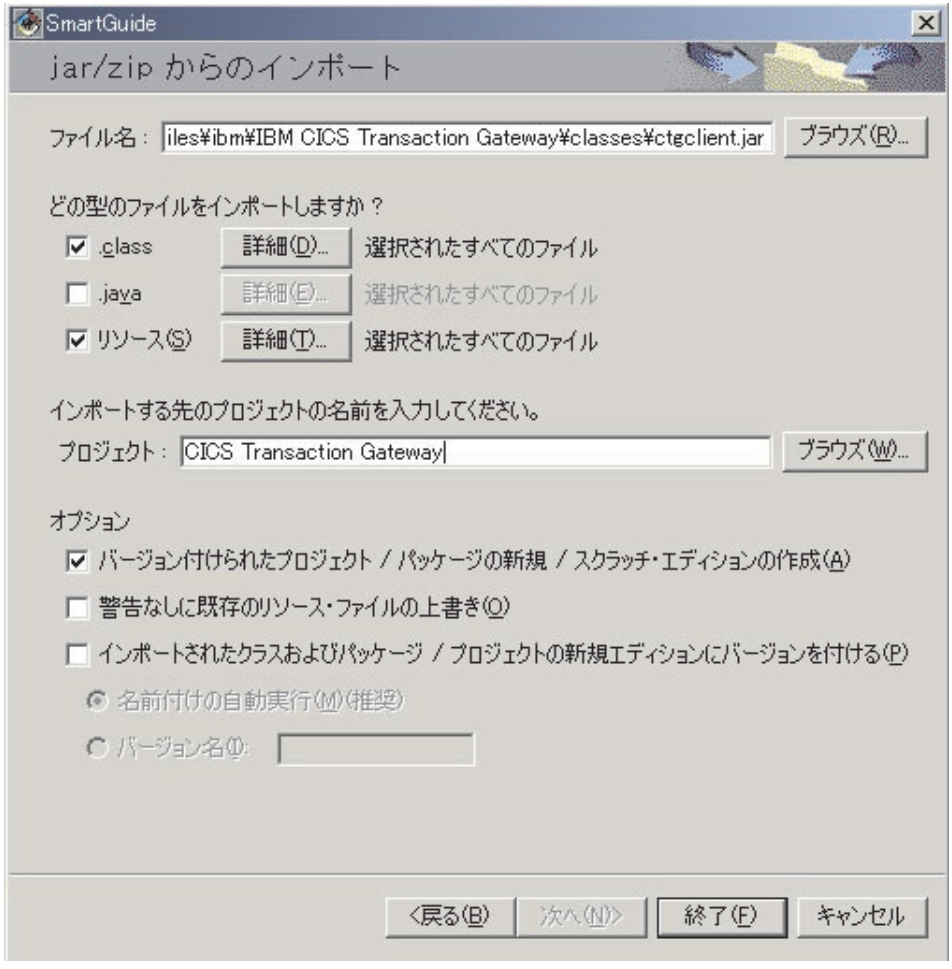

図 5. CICS Transaction Gateway クラスを VisualAge® for Java™ にインポートする

8. **「終了 (Finish)」**を選択します。CICS Transaction Gateway というプロジェクトは存在しないので、新しく作成するかどうかと尋ねる質問のボックスが表示されます。**「はい (Yes)」**を選択します。

   クラスのインポートの処理中に問題が報告されることがありますが、無視して差し支えありません。処理が完了すると、「パレットの変更 (Modify Palette)」ダイアログが表示されます。

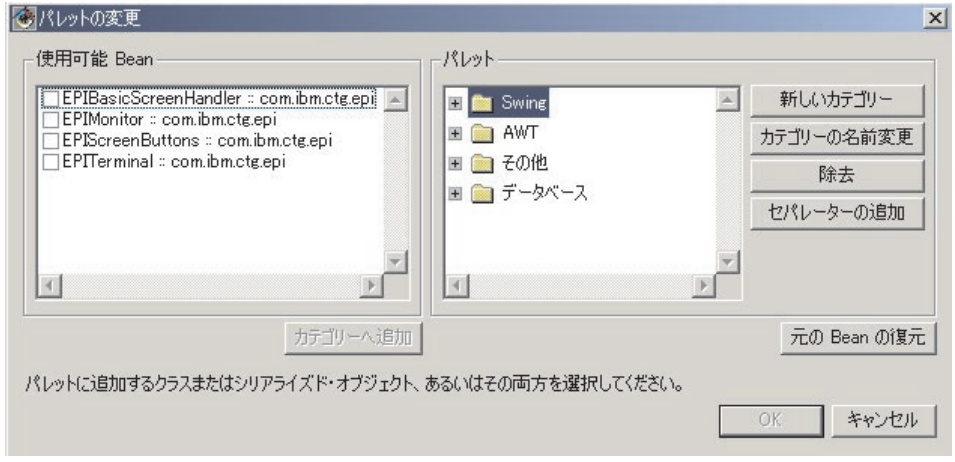

図 6. 「パレットの変更 (Modify Palette)」ダイアログ

9. **「新しいカテゴリー (New Category)」**を選択し、新しいカテゴリーの名前として「CTG beans」と入力します。
10. 左側のパネルのすべての使用可能な beans が選択されていることと、新しいカテゴリーである CTG beans が選択されていることを確認します。
11. **「カテゴリーへ追加 (Add to Category)」**を選択します。図7 は、「パレットの変更 (Modify Palette)」ダイアログに含まれる CICS Transaction Gateway Bean を示しています。

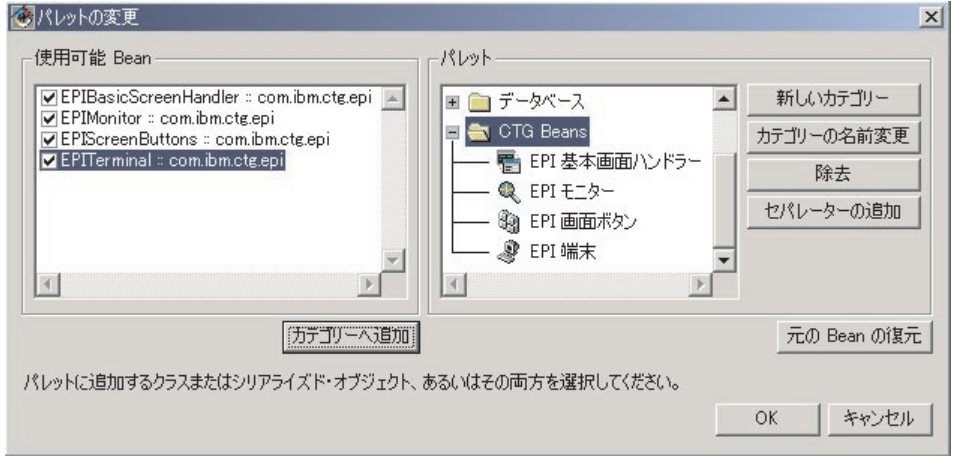

図 7. 「パレットの変更 (Modify Palette)」ダイアログおよび CICS Transaction Gateway Bean

12. **「OK」**を選択します。

すると、CICS Transaction Gateway クラスと beans がインポートされ、VisualAge® for Java™ のプログラム開発で使用できるようになります。

注: VisualAge® for Java™ エンタープライズ版をご使用の場合は、 Workspace から**「CICS Connector」**プロジェクトを除去するか、代わりに、そのプロジェクトに CICS Transaction Gateway クラスをインポートすることが必要です。こうすると、Workspace 内の別々のプロジェクトに、同一クラスのさまざまなレベルが同時に存在することが防げます。同一クラスのさまざまなレベルが同時に存在すると、問題の原因となる場合があります。

EPI アプリケーションの作成

EPI アプリケーションを作成する場合、VisualAge® for Java™ のクイック・スタート機能を使用して、アプリケーションのフレームワークを生成することができます。そして、VisualAge® for Java™ のビジュアル・コンポジション・エディターを使用して、アプリケーションのキャンバスのレイアウト、ボタンと CICS Transaction Gateway beans の追加、 bean のプロパティーの構成、およびボタンと beans の接続が行えます。

アプリケーションの作成

新しいアプリケーションを作成する手順は次のとおりです。

1. **「File」**メニューで**「Quick Start」**を選択し、「Quick Start」ダイアログを表示します。
2. 左側のパネルで**「Basic」**を選択し、右側のパネルで**「Create application」**を選択します。
3. **「OK」**を選択して、 SmartGuide の「Create application」ダイアログをオープンします。
4. **「Project」**フィールドに、適切なプロジェクト名を入力します。
5. **「Package」**フィールドに、適切なパッケージ名を入力します。
6. **「Class」**フィールドに、適切なクラス名を入力します。
7. **「Create AWT based application」**ラジオ・ボックスが設定されていることを確認します。
8. **「Finish」**を選択します。

処理が完了すると、アプリケーションのブランクのキャンバスが VisualAge® for Java™ のビジュアル・コンポジション・エディターに表示されます。

EPI Terminal の作成

EPI Terminal bean をアプリケーションに追加し、 CICS サーバーにアクセスできるようこの bean を構成する手順は次のとおりです。

1. ビジュアル・コンポジション・エディターの画面の左上隅のプルダウン・メニューで、**「CTG beans」**を選択します。
2. 画面の左側にある、**「EPI Terminal」**というラベルの付いたアイコンを選択します。(アイコンの上にカーソルを移動すると、ラベルが表示されます。)
3. キャンバスの中の、四角形のアウトラインの外側の任意の位置にカーソルを移動します。すると、カーソルが十字線に変わります。左マウス・ボタンでクリックすると、**「EPITerminal1」**というラベルの付いたアイコンが、その位置に作成されます。
4. 新しく作成した**「EPI Terminal」**アイコンを右マウス・ボタンで選択し、ポップアップ・メニューで**「Properties」**を選択します。
5. **「GatewayURL」**プロパティーを修正し、 CICS Transaction Gateway が稼働しているホストを指定します (例: tcp://dilbert:2006)。
6. **「terminal settings」**入力フィールドの右側を選択して、「Terminal settings」ダイアログをポップアップ表示します。リモート CICS サーバーの名前を**「CICS server name」**フィールドに入力し、**「OK」**を選択します。

これで、EPI Terminal bean である、EPITerminal1 が構成されました。

ログオン・ボタンの作成

ログオン・ボタン bean をアプリケーションに追加し、このボタン bean を EPI Terminal bean に接続する手順は次のとおりです。

1. ビジュアル・コンポジション・エディターの画面の左上隅のプルダウン・メニューで、**「AWT」**を選択します。
2. 画面の左側にある、**「Button」**というラベルの付いたアイコンを選択します。 (アイコンの上にカーソルを移動すると、ラベルが表示されます。)
3. キャンバスの中の、四角形のアウトラインの内側の任意の位置にカーソルを移動します。すると、カーソルが十字線に変わります。左マウス・ボタンでクリックすると、**「Button1」**というラベルの付いたボタンが、その位置に作成されます。
4. 新しく作成した**ボタン bean** を右マウス・ボタンで選択し、ポップアップ・メニューで**「Properties」**を選択します。

5. ボタン bean の label プロパティーを**「Logon」**に変更し、プロパティー・ダイアログをクローズします。
6. 新しく作成した**ボタン bean** (**「Logon」**というラベルが付いている) を右マウス・ボタンで選択し、ポップアップ・メニューで**「Connect」**、**「actionPerformed」**の順に選択します。すると、カーソルと**「Logon」**ボタンが破線で結ばれます。
7. このカーソルで、**「EPI Terminal」**bean を選択します。ポップアップ・メニューで**「Connectable Features」**を選択します。すると、「End connection to」ダイアログがオープンします。
8. 「End connection to」ダイアログで、画面の上部の**「Method」**ボタンを選択し、パネルで **connect()** メソッドを選択してから、**「OK」**を選択します。

EPI Basic Screen Handler の作成

EPI Basic Screen Handler bean をアプリケーションに追加し、これを EPI Terminal bean に接続する手順は次のとおりです。

1. ビジュアル・コンポジション・エディターの画面の左上隅のプルダウン・メニューで、**「CTG beans」**を選択します。
2. 画面の左側にある、**「EPI Basic Screen Handler」**というラベルの付いたアイコンを選択します。 (アイコンの上にカーソルを移動すると、ラベルが表示されます。)
3. キャンバスの中の、四角形のアウトラインの内側の任意の位置にカーソルを移動します。すると、カーソルが十字線に変わります。左上隅の付近を左マウス・ボタンでクリックして、その位置に新しい四角形のアウトラインを作成します。この四角形のアウトラインは EPI Basic Screen Handler bean を表します。
4. 右マウス・ボタンで **EPI Terminal** bean を選択し、ポップアップ・メニューで**「Connect」**、**「Connectable Features」**の順に選択します。そうすると、「Start connection from」ダイアログがオープンします。
5. 「Start connection from」ダイアログで、画面の上部の**「Event」**を選択し、パネルで**「handleScreen」**を選択してから、**「OK」**を選択します。
6. すると、カーソルと「Logon」ボタンが破線で結ばれます。このカーソルで、**「EPI Basic Screen Handler」**の四角形を選択します。ポップアップ・メニューで**「Connectable Features」**を選択します。すると、「End connection to」ダイアログがオープンします。

7. 「End connection to」ダイアログで、画面の上部の**「Method」**ボタンを選択し、パネルで **handleScreen(com.ibm.ctg.terminalEvent)** メソッドを選択してから、**「OK」**を選択します。
8. EPI Basic Screen Handler の四角形と EPI Terminal bean を結ぶ破線を右マウス・ボタンで選択し、「Properties」ダイアログでイベントからメソッドの接続を開きます。
9. 「Event-to-method connection」の「Properties」で、チェック・ボックス**「Pass event data」**を選択し、**「OK」**を選択します。

「Logon」ボタンと EPI Terminal の接続

「Logon」ボタン bean には、EPI Terminal との接続がもう 1 つ必要です。これは、CICS サーバーとの接続が確立された後、トランザクションも開始されるようにするためです。

「Logon」ボタンに 2 番目の接続を作成する手順は次のとおりです。

1. 右マウス・ボタンで**「Logon」ボタン bean** を選択し、ポップアップ・メニューで**「Connect」**、**「actionPerformed」**の順に選択します。すると、カーソルと「Logon」ボタンが破線で結ばれます。
2. このカーソルで、「**EPI Terminal bean**」を選択します。ポップアップ・メニューで**「Connectable Features」**を選択します。すると、「End connection to」ダイアログがオープンします。
3. 「End connection to」ダイアログで、画面の上部の**「Method」**ボタンを選択し、パネルで **startTran()** メソッドを選択してから、**「OK」**を選択します。

EPI Screen Buttons の作成

EPI Screen Buttons bean をアプリケーションに追加する手順は次のとおりです。

1. ビジュアル・コンポジション・エディターの画面の左上隅のプルダウン・メニューで、**「CTG Beans」**を選択します。
2. 画面の左側にある、**「EPI Screen Buttons」**というラベルの付いたアイコンを選択します。 (アイコンの上にカーソルを移動すると、ラベルが表示されます。)
3. キャンバスの中の、四角形のアウトラインの内側の任意の位置にカーソルを移動します。すると、カーソルが十字線に変わります。右下隅の付近を左マウス・ボタンでクリックして、その位置に EPI Screen Buttons bean を作成します。

4. 右マウス・ボタンで EPI Screen Buttons bean を選択し、ポップアップ・メニューで**「Connect」**、**「Connectable Features」**の順に選択します。そうすると、「Start connection from」ダイアログがオープンします。
5. 「Start connection from」ダイアログで、画面上部の**「Event」**ボタンを選択し、パネルで**「actionPerformed」**を選択してから、**「OK」**を選択します。
6. すると、カーソルと EPI Screen Buttons bean が破線で結ばれます。このカーソルで、**「EPI Basic Screen Handler」**の四角形を選択します。ポップアップ・メニューで**「Connectable Features」**を選択します。すると、「End connection to」ダイアログがオープンします。
7. 「End connection to」ダイアログで、画面の上部の**「Method」**を選択し、パネルで **actionPerformed(java.awt.event.ActionEvent)** メソッドを選択してから、**「OK」**を選択します。
8. EPI Basic Screen Handler の四角形と EPI Screen Buttons bean を結ぶ破線を右マウス・ボタンで選択し、「Event-to-method connection」の「Properties」ダイアログをオープンします。
9. 「Event-to-method connection」の「Properties」で、チェック・ボックス**「Pass event data」**を選択し、**「OK」**を選択します。

VisualAge® for Java™ 内でアプリケーションをテストする場合

VisualAge® for Java™ の中からアプリケーションをテストする手順は次のとおりです。

1. ワークベンチの**「Bean」**メニューで、**「Run」**および**「In application Viewer」**を選択します。
2. アプリケーションの左下隅をドラッグして、すべてのコンポーネントが見えるよう、アプリケーションをサイズ変更します。
3. アプリケーションの**「Logon」**ボタンを選択します。CESN トランザクションが CICS サーバーで実行され、EPI Screen Handler アプリケーションが結果を表示します。
4. CICS サーバーのユーザー ID とパスワードを入力して、Enter を押します。
5. 画面をクリアしてテキスト入力ボックスを表示するには、**「Clear」**を選択します。
6. 任意の CICS トランザクション名とオペランドを入力して、Enter を押します。

7. ビジュアル・コンポジション・エディターの中で、マウスによるドラッグの手法を使用して、EPI Basic Screen Handler のサイズ変更とアプリケーション beans の再配置を行い、アプリケーションのすべてのコンポーネントがアプリケーション・ビューアーの中に表示されるようにします。

アプリケーションのエクスポート

VisualAge® for Java™ の中でアプリケーションを作成した後、これを VisualAge® for Java™ の外部にエクスポートする必要があります。

VisualAge® for Java™ のエクスポート機能を使用すると、問題の生じることがあります。エクスポート機能を使用すると、アプリケーションに必要な CICS Transaction Gateway のすべての静的クラスは選択されますが、動的クラスは省略されてしまいます。この問題の解決法の 1 つは、エクスポートの実行中に必要な動的クラスを手作業で選択する、という方法です。

アプリケーションのエクスポートと配布の手順は次のとおりです。

1. VisualAge® for Java™ のワークベンチに移動し、**「Project」**タブを選択して、作成したアプリケーションのクラスに移動します。
2. 作成したアプリケーションのクラスを、右マウス・ボタンで選択します。ポップアップ・メニューで**「Export」**を選択し、SmartGuide の「Export」ダイアログをオープンします。
3. SmartGuide の「エクスポート (Export)」ダイアログで、**「ディレクトリー (Directory)」**を選択してから**「次へ (Next)」**を選択し、SmartGuide の「ディレクトリーへのエクスポート (Export to a directory)」ダイアログをオープンします。(68ページの図8 を参照してください。)

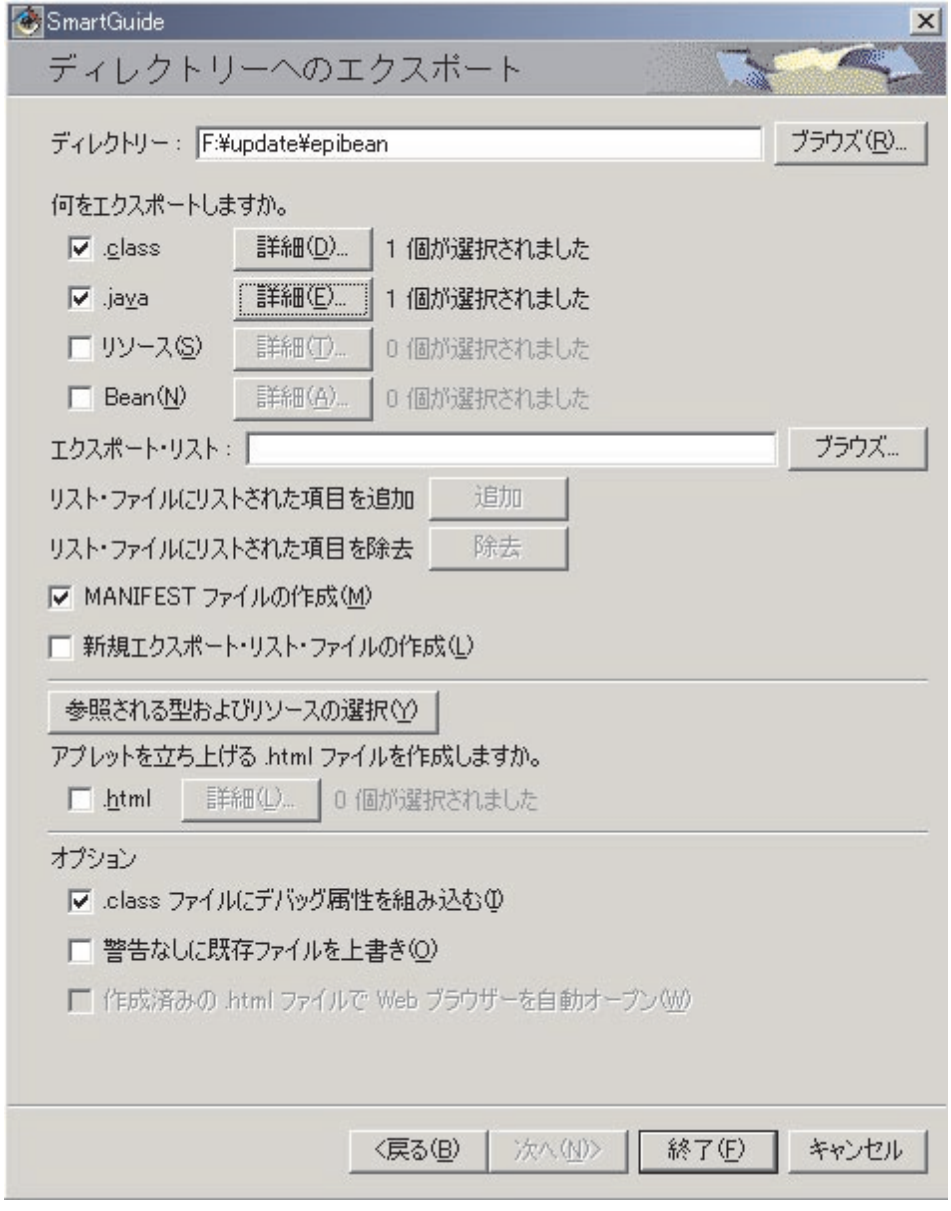

図 8. 「ディレクトリーへのエクスポート (Export to a directory)」ダイアログ

4. 適切なディレクトリーを選択した後、**「.class」**を選択します。「.java」がチェックされていることを確認して、**「終了 (Finish)」**を選択します。選択したディレクトリーを作成するかどうか尋ねられたら、**「はい (Yes)」**を選択します。VisualAge® for Java™ は、アプリケーション用の.class ファイルを作成し、これを指定したディレクトリーに配置します。

第7章 Java クラスの参照情報

CICS Transaction Gateway に添付されている Java クラスとインターフェースには、オンラインのプログラミング参照情報が用意されています。

参照情報は HTML 形式であり、JDK™ に添付されている Javadoc ツールで生成されています。

参照情報を入手するには、以下のようにします。

- Windows の場合は、**「資料 (Documentation)」**アイコンを選択します。
- AIX®、Solaris、Linux、および HP-UX の場合は、**ctgdoc** スクリプトを実行します。

表示されるライブラリー・ホーム・ページから、参照情報へのリンクをたどることができます。

以下の各セクションでは、参照情報の中に用意されている各種 HTML ページについて説明します。

README ファイル: プログラミング参照情報の使用に際しては、 README ファイルで最新情報を確認してください。

クラス / インターフェースのページ

参照情報ページでは、それぞれのクラスとインターフェースごとにページを設けています。各ページには、3 つのセクションがあります。

1. クラス / インターフェースの説明
   - クラスの継承図
   - 直接のサブクラス
   - 認識されているすべてサブインターフェース
   - 認識されているすべての実装クラス
   - クラス / インターフェースの宣言
   - クラス / インターフェースの説明
2. テーブルの要約
   - 内部クラスの要約
   - フィールドの要約

- コンストラクターの要約
- メソッドの要約

3. クラス / インターフェースの説明
   - フィールドの詳細
   - コンストラクターの詳細
   - メソッドの詳細

それぞれの要約項目には、その項目の詳細説明の最初の文が入ります。要約項目は、アルファベット順、詳細説明は、ソース・コードに登場する順番に配列されます。したがって、プログラマーが確立した論理的なグループ分けがそのまま維持されます。

使用法のページ

取り上げているそれぞれのクラスとインターフェースごとに、使用法のページを設けています。このページでは、そのクラスに何らかの形でかかわるパッケージ、クラス、メソッド、コンストラクター、フィールドについて説明しています。パッケージまたはインターフェース A の使用法のページには、以下の項目が入っています。

- A のサブクラス
- A として宣言されたフィールド
- A を戻すメソッド
- タイプ A のパラメーターを設定したメソッドとコンストラクター

このページにアクセスするには、まずそのクラスかインターフェースのページに入ってから、ナビゲーション・バーの**「使用法 (Use)」**リンクをクリックします。

ツリー (クラス階層)

特定のクラスやインターフェースのページを表示したときに、**「ツリー (Tree)」**を選択すると、 CICS Transaction Gateway のクラスとインターフェースの階層が表示されます。

使用すべきでない API

使用すべきでない API のページには、使用すべきでないすべての API のリストがあります。使用すべきでない API とは、品質改良などによって使用に適さなくなったものを指します。使用すべきでない API は、今後の実装では削除される場合があります。

索引のページ

索引のページには、すべてのクラス、インターフェース、コンストラクター、メソッド、フィールドのアルファベット順リストがあります。

付録A. Java エンコード

この付録では、サポートされる Java エンコードをリストしています。データ・ストリームが配置されるコード・ページを CICS サーバーが判別できるよう、標準名が対応の CCSid に変換されます。

注:

1. このエンコードを実装するには、CICS サーバーが EPI バージョン 2 をサポートしている必要があります。
2. CICS サーバーの資料を参照して、サーバーのサポートする CCSid を確認してください。

表 4. Java エンコード

標準名	説明	CCSid
Cp1252	Windows Latin-1	5348
ISO8859_1	ISO 8859-1、Latin アルファベット No. 1	819
UTF8	8 ビット ユニコード変換形式	1208
ASCII	情報交換用米国標準コード (ASCII)	437
Big5	Big 5、中国語 (繁体字)	950
Cp037	USA、カナダ (2 か国語、フランス語)、オランダ、ポルトガル、ブラジル、オーストラリア	37
Cp273	IBM オーストリア、ドイツ	273
Cp277	IBM デンマーク、ノルウェー	277
Cp278	IBM フィンランド、スウェーデン	278
Cp280	IBM イタリア	280
Cp284	IBM カタロニア語 / スペイン、スペイン語圏ラテンアメリカ	284
Cp285	IBM 英国、アイルランド	285
Cp297	IBM フランス	297
Cp420	IBM アラビア語	420
Cp424	IBM ヘブライ語	424
Cp437	MS-DOS 米国、オーストラリア、ニュージーランド、南アフリカ	437
Cp500	EBCDIC 500V1	500

表 4. Java エンコード (続き)

標準名	説明	CCSid
Cp838	IBM タイ拡張 SBCS	9030
Cp850	MS-DOS Latin-1	850
Cp852	MS-DOS Latin-2	852
Cp855	IBM キリル文字	855
Cp856	IBM ヘブライ語	856
Cp857	IBM トルコ語	857
Cp858	Cp850 変種とユーロ文字	858
Cp862	PC ヘブライ語	862
Cp864	PC アラビア語	864
Cp865	MS-DOS 北欧ゲルマン系言語	865
Cp866	MS-DOS ロシア語	866
Cp868	MS-DOS パキスタン	868
Cp869	IBM Modern ギリシャ語	869
Cp870	IBM マルチリンガル Latin-2	870
Cp871	IBM アイスランド	871
Cp874	IBM タイ語	9066
Cp875	IBM ギリシャ語	875
Cp918	IBM パキスタン (ウルドゥー語)	918
Cp921	IBM ラトビア、リトアニア (AIX、DOS)	921
Cp922	IBM エストニア (AIX、DOS)	922
Cp923	IBM Latin-9	923
Cp930	4370 UDC との混合日本語カタカナ漢字、5026 のスーパーセット	930
Cp933	1880 UDC との混合韓国語、5029 のスーパーセット	933
Cp935	1880 UDC との混合中国語 (簡体字) ホスト、5031 のスーパーセット	935
Cp937	6204 UDC との混合中国語 (繁体字) ホスト、5033 のスーパーセット	937
Cp939	4370 UDC との混合日本語英数小文字漢字、5035 のスーパーセット	939
Cp942	IBM OS/2 日本語、Cp932 のスーパーセット	942
Cp942C	Cp942 の変種	942

表 4. Java エンコード (続き)

標準名	説明	CCSid
Cp943	IBM OS/2 日本語、Cp932 および Shift-JIS のスーパーセット	943
Cp943C	Cp943 の変種	943
Cp948	OS/2 中国語 (台湾)、938 のスーパーセット	948
Cp949	PC 韓国語	949
Cp949C	Cp949 の変種	949
Cp950	PC 中国語 (香港、台湾)	950
Cp964	AIX 中国語 (台湾)	964
Cp970	AIX 韓国語	970
Cp1006	IBM AIX パキスタン (ウルドゥー語)	1006
Cp1025	IBM マルチリンガル・キリル文字: ブルガリア、ボスニア、ヘルツェゴビナ、マケドニア (FYR)	1025
Cp1026	IBM Latin-5、トルコ	1026
Cp1097	IBM イラン (ペルシア語) / ペルシア語	1097
Cp1098	IBM イラン (ペルシア語) / ペルシア語	1098
Cp1112	IBM ラトビア、リトアニア	1112
Cp1122	IBM エストニア	1122
Cp1123	IBM ウクライナ	1123
Cp1124	IBM AIX ウクライナ	1124
Cp1140	Cp037 変種とユーロ文字	1140
Cp1141	Cp273 変種とユーロ文字	1141
Cp1142	Cp277 変種とユーロ文字	1142
Cp1143	Cp278 変種とユーロ文字	1143
Cp1144	Cp280 変種とユーロ文字	1144
Cp1145	Cp284 変種とユーロ文字	1145
Cp1146	Cp285 変種とユーロ文字	1146
Cp1147	Cp297 変種とユーロ文字	1147
Cp1148	Cp500 変種とユーロ文字	1148
Cp1149	Cp871 変種とユーロ文字	1149
Cp1250	Windows 東欧	5346
Cp1251	Windows キリル文字	5347
Cp1253	Windows ギリシャ語	5349
Cp1254	Windows トルコ語	5350

表 4. Java エンコード (続き)

標準名	説明	CCSid
Cp1255	Windows ヘブライ語	5351
Cp1256	Windows アラビア語	5352
Cp1257	Windows バルト語	5353
Cp1258	Windows ベトナム語	5354
Cp1381	IBM OS/2、DOS 中華人民共和国 (PRC)	1381
Cp1383	IBM AIX、中華人民共和国 (PRC)	1383
EUC_CN	GB2312、EUC エンコード、中国語 (簡体字)	1383
EUC_JP	JIS X 0201、0208、0212、EUC エンコード、日本語	954
EUC_KR	KS C 5601、EUC エンコード、韓国語	970
GBK	GBK、中国語 (簡体字)	1386
ISO8859_2	ISO 8859-2、Latin アルファベット No. 2	912
ISO8859_5	ISO 8859-5、Latin / キリル文字アルファベット	915
ISO8859_6	ISO 8859-6、Latin / アラビア語アルファベット	1089
ISO8859_7	ISO 8859-7、Latin / ギリシャ語アルファベット	813
ISO8859_8	ISO 8859-8、Latin / ヘブライ語アルファベット	916
ISO8859_9	ISO 8859-9、Latin アルファベット No. 5	920
ISO8859_15_FDIS	ISO 8859-15、Latin アルファベット No. 9	923
JIS0201	JIS X 0201、日本語	5050
JIS0208	JIS X 0208、日本語	5050
JIS0212	JIS X 0212、日本語	5050
EUC_TW	CNS 11643 (Plane 1-3)、EUC エンコード、中国語 (繁体字)	964
MS932	Windows 日本語	943
MS936	Windows 中国語 (簡体字)	1386
MS949	Windows 韓国語	1363

付録B. CICS Transaction Gateway Java サンプル

ECI、ESI、EPI 要求クラス、および EPI と Support クラスの使用法を示す Java サンプルが提供されています。

- ECIRequest および EPIRequest 用の、基本、中間、および拡張サンプル。
- ESI 用の基本サンプル。
- EPI および Support クラス用の、基本サンプルおよび 2 つの JavaBean サンプル。 JavaBean サンプルの一方は、VisualAge 用であり、もう一方は開発環境用です。

4 つのセキュリティー・サンプルも提供されています。これらはすべて中間レベル Java サンプルに統合されています。

Java サンプルは、CICS Transaction Gateway の Samples ディレクトリーにあります。詳細については、Samples.txt を参照してください。

付録C. CICS Transaction Gateway と CICS ユニバーサル・クライアントのライブラリー

この章では、 CICS Transaction Gateway と CICS ユニバーサル・クライアントのマニュアルや関連資料をすべてリストアップし、それぞれの資料の形式について説明します。

この章には、以下の見出しがあります。

CICS Transaction Gateway のマニュアル

- *CICS Transaction Gateway: Windows® Gateway 管理*、*SC88-8984*

  このマニュアルでは、 CICS® Transaction Gateway (Windows® 版) の管理について説明しています。

- *CICS Transaction Gateway: AIX® Gateway 管理*、*SC88-8985*

  このマニュアルでは、 CICS® Transaction Gateway (AIX® 版) の管理について説明しています。

- *CICS Transaction Gateway: Solaris Gateway 管理*、*SC88-8986*

  このマニュアルでは、 CICS® Transaction Gateway (Solaris 版) の管理について説明しています。

- *CICS Transaction Gateway: Linux Gateway 管理*、*SC88-8989*

  このマニュアルでは、CICS® Transaction Gateway (Linux 版) の管理について説明しています。

- *CICS Transaction Gateway: HP-UX Gateway 管理*、*SC88-8988*

  このマニュアルでは、CICS® Transaction Gateway (HP-UX 版) の管理について説明しています。

CICS Transaction Gateway と CICS ユニバーサル・クライアントのライブラリー

- *CICS Transaction Gateway: OS/390® Gateway 管理*、*SC88-8987*

  このマニュアルでは、 CICS® Transaction Gateway (OS/390® 版) の管理について説明しています。

- *CICS Transaction Gateway: Gateway Messages*

  このオンライン・マニュアルでは、 CICS Transaction Gatewayから生成されるエラー・メッセージをリストアップし、それぞれについて説明しています。

  このマニュアルは、ご注文いただけません。

- *CICS Transaction Gateway: Gateway プログラミング*、SC88-8990

  このマニュアルは、CICS Transaction Gateway の Java™ プログラミング入門です。

  プログラミング参照情報を載せた追加の HTML ページもあります。

CICS ユニバーサル・クライアントのマニュアル

- *CICS Transaction Gateway: Windows® クライアント管理*、*SC88-8991*

  このマニュアルでは、 CICS ユニバーサル・クライアント (Windows 版) の管理について説明しています。

- *CICS Transaction Gateway: AIX® クライアント管理*、*SC88-8992*

  このマニュアルでは、 CICS ユニバーサル・クライアント (AIX 版) の管理について説明しています。

- *CICS Transaction Gateway: Solaris クライアント管理*、*SC88-8993*

  このマニュアルでは、CICS ユニバーサル・クライアント (Solaris 版) の管理について説明しています。

- *CICS Transaction Gateway: Linux クライアント管理*、*SC88-8995*

  このマニュアルでは、CICS ユニバーサル・クライアント (Linux 版) の管理について説明しています。

- *CICS Transaction Gateway: HP-UX クライアント管理*、*SC88-8994*

  このマニュアルでは、CICS ユニバーサル・クライアント (HP-UX 版) の管理について説明しています。

- *CICS Transaction Gateway: Client Messages*

  このオンライン・マニュアルでは、CICS ユニバーサル・クライアントから生成されるエラー・メッセージとトレース・メッセージをリストアップし、それぞれについて説明しています。

  このマニュアルは、ご注文いただけません。

- *CICS Transaction Gateway: C++ プログラミング*、SC88-8996

  このマニュアルでは、 C++ 言語で、 ECI と EPI 用のオブジェクト指向プログラムを作成する方法について説明しています。

- *CICS Transaction Gateway: COM オートメーション・プログラミング*、SC88-8997

  このマニュアルでは、Component Object Model (COM) 規格に従って、ECI と EPI 用のオブジェクト指向プログラムを作成する方法について説明しています。

CICS ファミリーの資料

- *CICS® ファミリー: クライアント / サーバー・プログラミング*、SC88-8998

  このマニュアルでは、 CICS クライアント / サーバー・プログラミングに関連するプログラミング・インターフェースである、外部呼び出しインターフェース (ECI)、外部表示インターフェース (EPI)、外部セキュリティー・インターフェース (ESI) について説明しています。 CICS サーバー・システムと通信するクライアント・アプリケーションを開発したいと思っているアプリケーション設計者やプログラマーを対象としています。

マニュアルのファイル名

表5 は、CICS Transaction Gateway および CICS ユニバーサル・クライアントの資料のソフトコピー・ファイル名を示しています。

表 5. CICS Transaction Gateway および CICS ユニバーサル・クライアントの資料およびファイル名

マニュアルのタイトル	ファイル名
CICS Transaction Gateway: Client Messages	CCLIAB
CICS Transaction Gateway: AIX® クライアント管理	CCLIAD
CICS Transaction Gateway: Windows® クライアント管理	CCLIAF
CICS Transaction Gateway: Solaris クライアント管理	CCLIAG
CICS Transaction Gateway: Linux クライアント管理	CCLIAR
CICS Transaction Gateway: HP-UX クライアント管理	CCLIAT
CICS Transaction Gateway: OS/390® Gateway 管理	CCLIAI
CICS Transaction Gateway: Gateway Messages	CCLIAJ
CICS Transaction Gateway: Gateway プログラミング	CCLIAK
CICS Transaction Gateway: Windows® Gateway 管理	CCLIAL
CICS Transaction Gateway: AIX® Gateway 管理	CCLIAN
CICS Transaction Gateway: Solaris Gateway 管理	CCLIAO
CICS Transaction Gateway: Linux Gateway 管理	CCLIAS
CICS Transaction Gateway: HP-UX Gateway 管理	CCLIAU
CICS Transaction Gateway: C++ プログラミング	CCLIAP
CICS Transaction Gateway: COM オートメーション・プログラミング	CCLIAQ
CICS® ファミリー: クライアント / サーバー・プログラミング	DFHZAD
注: この表にあるファイル名では、 2 桁の接尾部を省いています。	

サンプル構成の資料

いくつかのサンプル構成の資料が、 PDF 形式で用意されています。

これらの資料では、各種プロトコルを使用して CICS サーバーと通信する、CICS ユニバーサル・クライアントを構成する手順などを段階的に示しています。CICS Transaction Gateway と CICS ユニバーサル・クライアントのライブラリーにある情報よりもさらに詳細な説明を載せています。

新しいサンプル構成の資料が用意されたら、弊社の Web サイトからダウンロードできます。

www.ibm.com/software/ts/cics/

ここから、**Library** リンクをたどります。

その他の資料

International Technical Support Organization (ITSO) の以下のレッドブック™ 資料には、クライアント / サーバー構成のたくさんの例が載っています。

- *Revealed! CICS Transaction Gateway with more CICS Clients Unmasked*、*SG24-5277*

  このマニュアルは、以下のマニュアルの改訂版です。

- *CICS Clients Unmasked*、*GG24-2534*

ITSO のレッドブックは、いろいろな場所から入手できます。最新情報については、以下を参照してください。

www.ibm.com/redbooks/

CICS 製品の情報については、以下を参照してください。

www.ibm.com/software/ts/cics/

オンライン・マニュアルの表示

弊社のオンライン・ライブラリーでは、 CICS Transaction Gateway と CICS ユニバーサル・クライアントに添付されているすべてのマニュアルにアクセスできます (英語版のみ)。オンライン・ライブラリーを使用するには、 Adobe Acrobat Reader と適切な Web ブラウザーが必要です。場合によっては、これらを構成する必要もあります。

オンライン・ライブラリーを入手するには、 ライブラリー・ホーム・ページを表示します。

このオンライン・ライブラリーから、以下の資料やサイトにリンクできます。

- CICS Transaction Gateway と CICS ユニバーサル・クライアントの PDF 形式のマニュアル。
- プログラミング参照資料のハイパーテキスト・マークアップ言語 (HTML) ファイル (CICS Transaction Gateway のみ)。
- README ファイル。
- サンプル構成の PDF 形式の資料。
- CICS Web サイト。

Acrobat Reader の使用方法についても、説明があります。

マニュアルの改訂版が用意されることがありますので、弊社の Web サイトをチェックしてください。

www.ibm.com/software/ts/cics/

ここから、**Library** リンクをたどります。

注: 一部の資料には、翻訳済みのものがあります。これらの資料はオンライン・ライブラリーには含まれていませんが、上記の Web サイトより PDF ファイルを入手できます。

PDF マニュアルの表示

PDF 情報には、以下の便利な機能が用意されています。

- 情報のナビゲーション機能。 PDF 文書には、内部のハイパーテキスト・リンクや、他の PDF 文書や Web ページへのハイパーテキスト・リンクが設定されています。
- 情報の検索機能。
- PostScript プリンターによる、 PDF 文書の一部または全体の印刷機能。

Acrobat Reader の詳細については、 Adobe 社の Web サイトを参照してください。

www.adobe.com/acrobat/

付録D. 特記事項

本書はアメリカ合衆国で提供されている製品およびサービス用に作成されたものであり、本書に記載の製品、サービス、またはフィーチャーが日本においては提供されていない場合があります。日本で利用可能な製品、サービス、およびフィーチャーについては、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない機能的に同等のプログラムまたは製品を使用することができます。ただし、IBM 以外の製品、プログラムまたはサービスの操作性の評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む。) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権の許諾については、下記の宛先に書面にてご照会ください。

〒106-0032 東京都港区六本木 3 丁目 2-31
AP 事業所
IBM World Trade Asia Corporation
Intellectual Property Law & Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

本書は定期的に見直され、必要な変更 (たとえば、技術的に不適確な表現や誤植など) は、本書の次版に組み込まれます。IBM は、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するもので

はありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM United Kingdom Laboratories, MP151,
Hursley Park, Winchester, Hampshire,
England, SO21 2JN

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。また、IBM 以外の製品に関するパフォーマンスの正確性、互換性、またはその他の要求は確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、IBM のアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

商標

以下は、IBM Corporation の商標です。

AIX
IBM
VisualAge
CICS
OS/390

Microsoft、Windows、Windows NT、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc.の米国およびその他の国における商標または登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標または登録商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ハ行]

[マ行]

[ラ行]

[数字]

B

C

E

H

J

O

P

S

V

プログラム番号: 5724-A75

Printed in Japan

SC88-8990-00

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12