CICS® Transaction Gateway

HP-UX Gateway 管理

バージョン 4.0

SC88-8988-00
(英文原典：SC34-5936-00)

CICS® Transaction Gateway

HP-UX Gateway 管理

バージョン 4.0

SC88-8988-00
(英文原典：SC34-5936-00)

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、141ページの『付録C. 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 for HP-UX 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、他のオペレーティング・システム用の管理マニュアルの以前の版を基にした新規のマニュアルです。ページの左にある垂直線は、本版で新規に追加された内容を示しています。

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： SC34-5936-00
CICS® Transaction Gateway
HP-UX Gateway Administration
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

図

表

変更の要約

以下に、CICS Transaction Gateway バージョン 4.0 に対して行われた機能変更を示します。

- TCP62 がサポートされました。
- 以下がサポート対象外となりました。
  - Cobol
  - PL/I
  - REXX
  - CICSTELD

本書は、他のオペレーティング・システム用の管理マニュアルの以前の版を基にした新規のマニュアルです。ページの左にある垂直線は、本版で新規に追加された内容を示しています。

本書について

本書は、以下の章から構成されています。

- 第1章では、CICS Transaction Gateway について紹介し、それを使用するメリット、およびその機能について簡単に説明します。
- 第2章では、CICS Transaction Gateway
- 第3章では、CICS Transaction Gateway のインストール方法について説明します。
- 第4章では、CICS Transaction Gateway の構成方法について説明します。
- 第5章では、CICS Transaction Gateway で SSL および HTTPS プロトコルに対応するセキュリティーを設定する方法について説明します。
- 第6章では、CICS Transaction Gateway の開始および停止など、CICS Transaction Gateway の操作方法について説明します。
- 第7章では、CICS Transaction Gateway に付属している端末サーブレットについて紹介します。
- 第8章では、CICS Transaction Gateway の問題判別について説明します。
- 付録B では、CICS Transaction Gateway および CICS ユニバーサル・クライアントのライブラリー内のオンライン情報を表示する方法、および資料を印刷および注文する方法について説明します。

本書の対象読者

本書は、CICS Transaction Gateway の計画、インストール、カスタマイズ、または操作を行なうユーザーを対象としています。

本書では、CICS Transaction Gateway の実行環境であるオペレーティング・システムを熟知していることを前提とします。

インターネット用語を理解していることも、本書を読む上で役立ちます。

本書で使用される表記規則と用語

本書では、*CICS* ユニバーサル・クライアントという用語は、CICS Transaction Gateway のクライアント・コンポーネントを示します。

「System/390® 上の CICS」は、以下の CICS サーバー製品を示すために使用します。

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

CICS Transaction Gatewayは、 Windows NT® および Windows 2000 上で稼働します。本書の Windows® は、Windows の特定のバージョンを指定しない限り、Windows NT および Windows 2000 の両方を示します。

本文の OS/390 は、OS/390 および z/OS の両方のオペレーティング・システムを示します。

前提条件および関連情報

本製品に利用できる資料についての詳細は、135ページの『付録B. CICS Transaction Gateway および 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。その章では、ソフトコピー・ブックの表示および印刷方法が述べられています。

第1章 概説

CICS Transaction Gateway は、ある構成範囲内で標準のインターネット・プロトコルを使用して、Web ブラウザーおよびネットワーク・コンピューターから、CICS Transaction Server や TXSeries™ サーバーで実行されているビジネスにとって重要なアプリケーションに安全かつ簡単にアクセスできるようにするシステムです。

CICS Transaction Gateway は、Web サーバーに対する強力でスケーラブルな補助システムであり、そうしたことから、Java™ サーブレットの実行時環境である、IBM WebSphere™ の e-business コネクターとしてインプリメントすることができます。

CICS Transaction Gateway は、Windows NT®、AIX®、Linux、Solaris、HP-UX、および OS/390® プラットフォームに対応しています。

2ページの図1 に、Web クライアントが CICS プログラムとデータにアクセスする様子を示します。この図では、CICS Transaction Gateway は Web サーバー・マシンにインストールされていることに注意してください。このようにする必要があるのは、CICS Transaction Gateway を Java アプレットと一緒に使用する場合だけです。

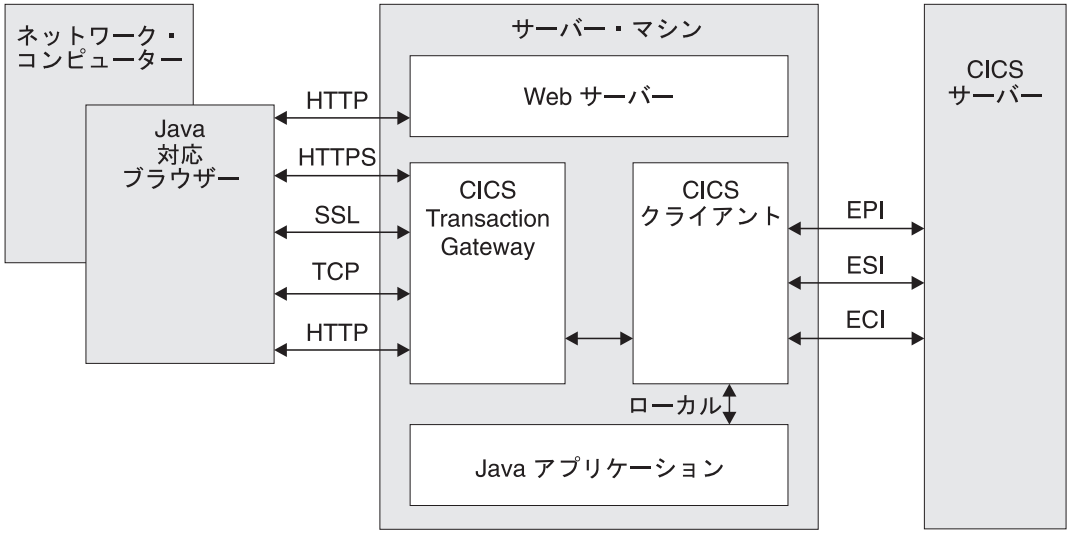


図 1. CICS Transaction Gateway

CICS Transaction Gateway との通信は以下のプロトコルに基づいて行われます。

- TCP/IP ソケット
- HTTP (Hypertext Transfer Protocol)
- SSL (Secure Sockets Layer)
- HTTPS (HTTP over SSL)

TCP/IP ソケットと SSL は、イントラネット・アプリケーションのための効率的な通信方式を提供します。ファイアウォールが存在する場合、HTTP と、そのより安全な代替手段である HTTPS は、インターネット・アプリケーションのための効率的な通信プロトコルです (16ページの『ネットワーク・セキュリティー』を参照)。

CICS Transaction Gateway が提供するもの

CICS Transaction Gateway は、以下のものを提供します。

1. **Java ゲートウェイ・アプリケーション**。通常は (セキュリティー上の理由から) Web サーバー・ワークステーション上に常駐しています。 CICS ユニバーサル・クライアントが提供する ECI (外部呼び出しインターフェース)、EPI (外部表示インターフェース)、または ESI (外部セキュリティー・インターフェース) を介して、CICS サーバーで実行されている CICS アプ

リケーションと通信します。この Java アプリケーションは、以前は IBM CICS Gateway for Java で利用されていました。

2. **CICS ユニバーサル・クライアント**。 ECI、EPI、および ESI インターフェースだけでなく、端末エミュレーション機能も提供します。 **ECI インターフェース**により、非 CICS クライアント・アプリケーションでも、 CICS プログラムをサブルーチンとして同期または非同期に呼び出すことができます。 **EPI インターフェース**により、非 CICS クライアント・アプリケーションでも、論理 3270 端末として動作することができるので、 CICS 3270 アプリケーションを制御することができます。 **ESI インターフェース**により、非 CICS クライアント・アプリケーションが、拡張プログラム間通信機能 (APPC) パスワード有効期限管理 (PEM) が提供するサービスを呼び出すことができます。 CICS ユニバーサル・クライアントは、オペレーティング・システムに応じて、TCP/IP、 APPC、およびその他のプロトコルで CICS サーバーと通信することができます。
3. **CICS Java クラス・ライブラリー**。アプリケーション・プログラミング・インターフェース (API) を提供し、Java ゲートウェイ・アプリケーションと Java アプリケーション (アプレットまたはサーブレット) 間との通信に使用するクラスが入っています。ゲートウェイ・プロセスとの通信を確立する場合は、クラス **JavaGateway** を使用します。このクラスは、Java ソケット・プロトコルを使用します。 **ECIRequest**、**EPIRequest**、および **ESIRequest** の各クラスは、それぞれ、ゲートウェイに流す ECI、EPI、および ESI 呼び出しを指定するために使用します。
4. **端末サーブレット**。Web ブラウザーを、CICS サーバー上で実行されている 3270 CICS アプリケーションのエミュレーターとして使用できるようにします。端末サーブレットは、Java サーブレット開発キット (JSDK) バージョン 1.1 またはそれ以降と等価のサポートを提供する、Web サーバーまたはサーブレット・エンジンと一緒に使用することができます。これは、IBM CICS クライアント、バージョン 2 の CICS Internet Gateway で提供されていた機能の拡張版です。端末サーブレットは、EPI プログラミング・インターフェースの使用に代わる方法を提供しています。
5. **Java EPI bean** のセット。プログラミングを行わずに、既存の CICS 3270 アプリケーションに合った Java フロントエンドを作成できます。

CICS Transaction Gateway は、接続先の Web ブラウザーとの通信リンクを同時に複数管理できると同時に、複数の CICS サーバー・システムとの非同期会話を制御することができます。 CICS Transaction Gateway のマルチスレッド・アーキテクチャーにより、 1 つの Gateway で、接続ユーザーを同時に複数サポートすることができます。

Java™ テクノロジー

この節では、開発可能なプログラムのタイプや、セキュリティー問題などの、Java 言語について説明します。

Java 言語

Java 言語は、Java サーブレット や、 Java アプレット、および Java アプリケーション を作成するために使用されます。

Java は、C++ に似た、インタープリター型のオブジェクト指向言語であり、この言語を使用することで、ソース形式およびオブジェクト形式のどちらでも、プラットフォームに依存しないプログラムを作成することができます。 Web ブラウザーだけでなく、Web サーバーにもまたがる、その操作上の固有の性質により、新しく強力な機能をインターネット・アプリケーションで実現することができます。

プラットフォームからの独立性を確保するため、 Java 言語では、プラットフォーム固有の操作を実行することができません。プリプロセッサー、演算子の多義化、多重継承、ポインターなどの一部の C++ 機能は除外されています。Java プログラミングはすべてクラス内にカプセル化されており、 Java 開発キット (JDK) には、プラットフォームの独立性、インクルード GUI 機能、入出力機能、およびネットワーク通信の確保に不可欠な特殊なクラスが入っています。

Java コンパイラーは、マシンに依存しない中間バイトコード形式を生成します。これは、実行時に、Java インタープリターによって処理されます。インタープリターはまた、実行時にバイトコードを検査して、その有効性と安全性をマシン環境に保証します。 Java インタープリターが提供する分離性から、これを Java 仮想マシン (JVM) ということもあります。

Java アプレット

Java アプレットは、Java 対応の Web ブラウザーやネットワーク・コンピューターにダウンロードされて、そこで実行される小さなアプリケーション・プログラムです。 Java アプレットは一般に、クライアント・コードがクライアント / サーバー・アーキテクチャーで実行するタイプの操作を実行します。入力を編集したり、画面を制御したり、データまたはデータベース操作を実行するサーバーにトランザクションを送信したりします。

アプレットの先頭には、<applet> という HTML タグを使用します。これにより、アプレット・コントロールが提供されて、アプレットが使用する表示域が指定されます。 Java 対応サーバーは、ページのダウンロード中にこのタグを

発見すると、 HTML イメージ・タグで参照されるイメージをダウンロードする場合と同じ方法でアプレットのバイトコードをダウンロードします。その後、Java 対応ブラウザーは、アプレットのバイトコードを解釈して実行します。アプレットは、画面入力を編集したり、画面出力を生成したり、ダウンロード元のコンピューターと通信したりすることができます。アプレットは、同時に複数実行することができます。

アプレット処理の例として、サーバーと常に通信して株式情報を取得し、それを画面上のウィンドウで更新するアプレットがあるとしましょう。

アプレットは一般的にあまり大きくないので、ダウンロードしてもエンド・ユーザーへの応答時間に重大なパフォーマンス上の影響が及ぶことはありません。それよりも、アプレットは、Web サーバーとの通信の反復を避けることができるので、ブラウザーのパフォーマンス全体を向上させることができます。また、イメージを Web ブラウザーにキャッシュする場合と同様に、アプレットをキャッシュしておけば、何度もダウンロードする必要がありません。

Java サーブレット

Java サーブレットは、Web ブラウザーにダウンロードされる Java アプレットと異なり、Web サーバー・マシン上で実行される小さな Java アプリケーションです。

Java サーブレットは、CGI (コモン・ゲートウェイ・インターフェース) プログラムに代わるものとして、人気が高まっています。 CGI プログラムよりも Java サーブレットが優れている点は、デーモン・プロセスでスレッドとして起動されるために、より速く実行できることです。つまり、メモリーに永久的に残るので、複数の要求を満たすことができます。

サーブレットを実行する 3 つの方法があります。

URL (Uniform Resource Locator) でサーブレットを呼び出す

サーブレットの URL の一般的な形式は次のとおりです。

```
http://machine-name:port/servlet/servlet-name
```

この方式は、ユーザーに情報を入力させる必要がない場合に使用します。要求パラメーターは、照会ストリングの形式で、URL にエンコードされます。

フォームを持つサーブレットを呼び出す

この方式では、Web ページで HTML フォームを作成する必要があります。

```
<FORM METHOD="GET" ACTION="/servlet/servlet-name">
attributes
</FORM>
```

この方式は、ユーザーに情報を入力させる必要がある場合に使用します。

サーバー側インクルードを使用してサーブレットを起動する

サーブレット側のインクルードは、 Web ページがユーザーに送信される前に Web サーバーによって処理されます。 HTML ソースでは、サーバー側のインクルードは次のようになります。

```
<SERVLET NAME="TerminalServlet" >
<PARAM NAME="request" VALUE="send">
<PARAM NAME="transaction" VALUE="CECI">
<PARAM NAME="display" VALUE="none">
</SERVLET>
```

Java アプリケーション

Java アプリケーションは、コンピューター上でローカルに実行されるプログラムです。このアプリケーションには、アプレットとしての機能のほかに、プラットフォーム固有の機能もあります。ローカル・ファイルにアクセスしたり、一般的なネットワーク接続を作成および受け入れたり、マシン固有のライブラリーにあるネイティブの C または C++ 機能を呼び出したりすることができます。

JavaBeans™

JavaBeans API は、Sun Microsystems が開発したもので、これを用いて Java でコンポーネント・ソフトウェアを作成することができます。コンポーネントは、自己完結型の再使用可能なソフトウェア単位であり、 VisualAge® for Java などのビジュアルなアプリケーション組み立てプログラムを使用して、ビジュアルに組み合わせ、アプレットやサーブレットを作成することができます。 JDK バージョン 1.1 対応のブラウザーまたはツールならすべて JavaBeans をサポートしています。

JavaBeans のコンポーネントのことを bean と言います。ほとんどのビルダー・ツールでは、 bean をパレットまたはツールボックスで保守することができます。ツールボックスから特定の bean を選択し、それをフォームにドロップし、外観や振る舞いを変更して、他の bean との対話方法を定義します。選択した bean とその他の bean とを組み合わせて、アプレット、サーブレット、または新しい bean を作成することができます。これらの作業はすべて、コードを作成せずに行えます。 JavaBeans の詳細については、Sun の Web サイト (www.java.sun.com) を参照してください。

CICS Transaction Gateway には、高いレベルの EPI インターフェースをベースとする **EPI Java bean** が用意されています。これらの bean を利用することにより、既存の CICS 7270 アプリケーションのデータにアクセスする、 Java プログラム (アプレットおよびアプリケーション) を簡単に作成することができます。 VisualAge for Java などの bean 組み立てツールを使用すれば、CICS に接続し、トランザクションを実行し、3270 画面からデータを表示し、ユーザー入力を CICS サーバーに送り返す、新しい Java フロントエンドを簡単に作成することができます。詳しくは、「*CICS Transaction Gateway: Gateway プログラミング*」をご覧ください。

ファイアウォール

Java アプレット通信を使用する場合に、設計上、現在考慮しなければならないことは、ファイアウォール の影響です。ファイアウォールとは、信用できるネットワークと信用できないネットワーク間で、認められていないトラフィックが流れないようにするソフトウェアの構成のことを指す用語です。ファイアウォールは、会社の資産を外部の侵入者から守るために適切な場所に設置されますが、同時に、正規の通信が制限されることもあります。ファイアウォールは、次の 2 通りの役割を果たします。

1. サーバーから外部ユーザーへの一般アクセス、すなわちインバウンド制限。
2. ファイアウォールの内側のエンド・ユーザーが、ファイアウォールの外側にある特定のネットワーク機能を実行できるようにする機能、すなわちアウトバウンド制限。

Gateway プロセッサーはファイアウォールの外側に置き、ファイアウォールを介して CICS サーバーに接続させることができるので、CICS Transaction Gateway の構成は、 1 つ目のインバウンド制限での問題発生の防止手段として適しています。ただし、エンド・ユーザーが対処しなければならないアウトバウンド・ファイアウォールは、問題になることがあります。大規模な会社では、ファイアウォールを使用して、使用できる接続やプロトコルのタイプを制限しています。

イントラネット (インターネットのローカルなインプリメンテーション) での Java の使用は、一般的にファイアウォールは要因ではないので、問題ありません。ただし、会社の外部のエンド・ユーザー向けのインターネット・アプリケーションを設計している場合は、エンド・ユーザー・ファイアウォールがインプリメンテーション要因であるかどうか判断する必要があります。要因である場合は、Java コードを Web サーバー上の Java サーブレットとして実行するなど、エンド・ユーザーに対する代替処理が必要になります。また、CICS Transaction Gateway がサポートする HTTP および HTTPS プロトコルの使用

も考慮してください。 17ページの『SSL (Secure Sockets Layer)』 および 18ページの『HTTPS』 を参照してください。

Web ブラウザーとネットワーク・コンピューター

CICS Transaction Gateway には、JDK バージョン 1.1 が使用可能な、 Java 対応の Web ブラウザーが必要です。たとえば、 Netscape Communicator 4.5.1 またはそれ以降 (Windows および AIX 版) などです。詳細については、21ページの『サポートされているソフトウェア』を参照してください。

Web ブラウザーは、HTTP (HyperText Transport Protocol) を使用して Web サーバーと通信し、 HTML (ハイパーテキスト・マークアップ言語) ページのダウンロードを要求します。これらの HTML ページには、Java アプレット (4ページの『Java アプレット』を参照)、またはサーブレットの呼び出しを組み込むことができます。アプレットは同時に複数実行することができます。

同じ情報でも、各ブラウザーごとに表示方法が異なる場合があります。

ネットワーク・コンピューターは、インターネット・ユーザー向けの低価格のコンピューターであり、 Web ブラウザーと同じことを実行します。

Web サーバー

Web サーバーは、Web ブラウザーが生成した情報要求に応答するソフトウェア・プログラムです。ブラウザーから要求を受け取ると、Web サーバーは要求を処理して、取るべきアクションを判別します。

- 要求された文書を戻す。
- 要求を拒否する。
- 外部アプリケーションでさらに処理を実行するように、要求を渡す。要求は、たとえば、検索要求を実行するデータベースに対するものであったり、ロータス ドミノなどの、より動的なフォームの情報伝達に対するものであったりします。

Web サーバーと外部アプリケーション間の通信は透過的であり、ユーザーは、要求を転送する Web サーバーの URL しか認識する必要はありません。また、すべての Web サーバーは、多くのブラウザーからの要求を同時に処理することができます。

限られたユーザー集団にアクセスを制限したり、商品やサービスの購入に合わせてセキュリティーを提供するように、専用サーバーを構成することもできます。

Web サーバーは、ほとんどすべてのプラットフォームに対応するものが存在しており、多くのメーカーから市販されています。 CICS Transaction Gateway によってサポートされている Web サーバーについては、21ページの『サポートされているソフトウェア』を参照してください。

CICS Transaction Gateway が CICS にアクセスする方法

この節では、CICS Transaction Gateway が CICS プログラムおよびデータにどのようにアクセスするかについて説明します。

Java アプレットの場合、アクセスは次のように実現されます。

1. Web ブラウザーまたはネットワーク・コンピューターは、 HTTP (Hypertext Transfer Protocol) プロトコルを使用して、Web サーバーの HTML ページを要求します。
2. Web サーバーは、Java アプレットを識別するタグが入っている HTML ページを戻します。
3. ブラウザーは、Web サーバーに関連する Java クラスの要求を開始します。
4. Web サーバーは、要求されたとおりの CICS Transaction Gateway クラスを含む、 Java クラスを戻します。
5. クラスが戻されると、Java アプレットが開始されます。
6. Java アプレットは **JavaGateway** オブジェクトを作成し、CICS Transaction Gateway に接続します。これにより、Java のソケット・プロトコルを使用して、ブラウザーと長期にわたって実行されている Gateway プロセスとの間の通信が確立されます。
7. Java アプレットは、それぞれ ECI、EPI、または ESI 呼び出しを含む **ECIRequest**、**EPIRequest**、または **ESIRequest** オブジェクトを作成し、 **JavaGateway.flow** メソッドを使用してそれを Gateway に送信します。
8. Gateway は、要求を受信し、アンパックして、CICS ユニバーサル・クライアントへの対応する ECI、EPI、または ESI 呼び出しを行います。
9. CICS ユニバーサル・クライアントは、目的の CICS サーバーに ECI、EPI、または ESI 呼び出しを渡します。
10. CICS サーバーは、呼び出しを処理し、必要に応じてユーザー ID およびパスワードの検証を行って、制御およびユーザー・データを CICS アプリケーション・プログラムに渡します。

11. CICS アプリケーション・プログラムは、処理を完了すると、制御およびデータを CICS に戻します。次に CICS はクライアントに制御およびデータを戻し、クライアントはそれを Gateway に戻します。
12. Gateway は、これらの結果をパックし、Web ブラウザー上で実行されている Java アプレットに戻します。

Java サーブレットの場合、アクセスは次のように実現されます。

1. Web サーバーは、サーブレットをロードして、初期化します。これは、Web サーバーの起動時、またはサーブレットに最初に要求が発行されたときに行われます。サーブレットは、この時点で **JavaGateway** オブジェクトを作成して、CICS Transaction Gateway に接続することができます。
2. サーブレットが正しい HTTP 要求によって呼び出されると、Web サーバーは、要求の詳細と共に、その **Service** メソッドを呼び出します。
3. Java サーブレットは、それぞれ ECI、EPI、または ESI 呼び出しを含む **ECIRequest**、**EPIRequest**、または **ESIRequest** オブジェクトを作成し、**JavaGateway.flow** メソッドを使用してそれを Gateway に送信します。
4. Gateway は、要求を受信し、アンパックして、CICS ユニバーサル・クライアントへの対応する ECI、EPI、または ESI 呼び出しを作成します。
5. CICS ユニバーサル・クライアントは、目的の CICS サーバーに ECI、EPI、または ESI 呼び出しを渡します。
6. CICS サーバーは、呼び出しを処理し、必要に応じてユーザー ID およびパスワードの検証を行って、制御およびユーザー・データを CICS アプリケーション・プログラムに渡します。
7. CICS アプリケーション・プログラムは、処理を完了すると、制御およびデータを CICS に戻します。次に CICS はクライアントに制御およびデータを戻し、クライアントはそれを Gateway に戻します。
8. Gateway は、これらの結果をパックし、Java サーブレットに戻します。
9. サーブレットは、CICS Transaction Gateway に対して作成した要求の結果を受け取ると、 Web ブラウザーに戻す HTTP 応答を生成します。

注: ECI、EPI、および ESI 呼び出しは、それぞれ **ECIRequest**、**EPIRequest**、および **ESIRequest** によってすべてサポートされているわけではありません。詳細については、「*CICS Transaction Gateway: Gateway プログラミング*」を参照してください。

CICS Transaction Gateway: スレッド化モデル

CICS Transaction Gateway は、ネットワーク接続を処理するため、また Java クライアントとの要求 / 応答用のスレッドを割り当てるために、マルチスレッド・モデルを提供しています。スレッド化モデルには、以下のコンポーネントが含まれています。

ConnectionManager

ConnectionManager は、特定の Java クライアント (アプレットまたはアプリケーション) からのすべての接続を管理します。要求を受け取ると、それは使用可能な Worker スレッドのプールから Worker スレッドを割り振って、その要求を実行します。 ConnectionManager の初期リソース・プールのサイズは、**接続マネージャー・スレッドの初期数**構成設定値により定義されます。ConnectionManager プールの最大サイズは、**接続マネージャー・スレッドの最大数**設定値で指定することができます (32ページの『構成ツールの使用』を参照)。これらの限度を CICS Transaction Gateway を開始する際に指定することもできます (91ページの『ユーザー指定オプションでの Gateway の開始』を参照)。

Worker

Worker とは、Java クライアントから実際に要求を実行するオブジェクトです。それぞれの Worker オブジェクトごとに専用のスレッドがあり、それは行うべき作業が発生したときに活動化されます。 Worker スレッドが完了すると、それは使用可能な Worker スレッドのプールに戻されます。 ConnectionManager の場合と同じように、Worker リソース・プールにも初期サイズがあって、それは **Worker スレッドの初期数**設定値により指定されます。Worker プールの最大サイズは、**Worker スレッドの最大数**設定値で指定することができます (32ページの『構成ツールの使用』を参照)。これらの限度を CICS Transaction Gateway を開始する際に指定することもできます (91ページの『ユーザー指定オプションでの Gateway の開始』を参照)。

スレッド化モデルが以下の図で例示されています。

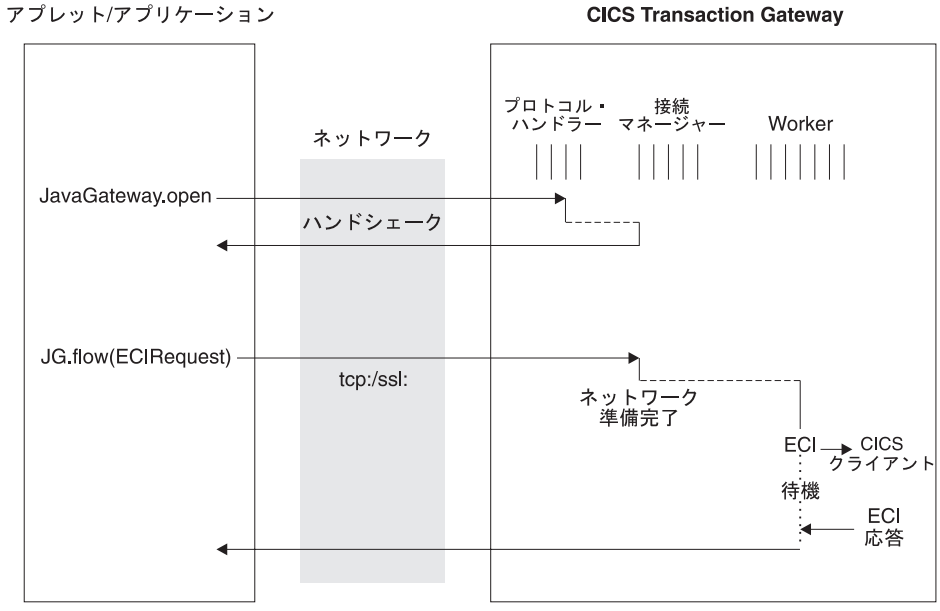


図 2. CICS Transaction Gateway スレッド化モデル (TCP/IP および SSL 用). これらのプロトコルは、永続ソケットを持っています。

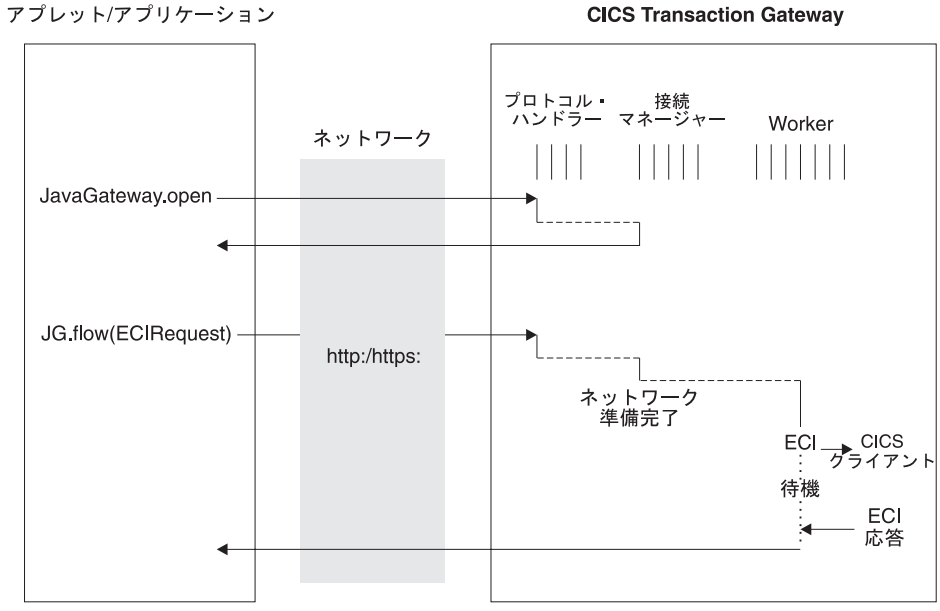


図 3. CICS Transaction Gateway スレッド化モデル (HTTP/HTTPS 用). これらのプロトコルは、永続ソケットを持っていません。

表1 は、各種のプラットフォームにおいて ConnectionManager および Worker スレッドの数を設定するに当たり、考慮すべきスレッドの限度を示しています。

表 1. CICS Transaction Gateway プラットフォームに対するスレッド制限

オペレーティング・システム	スレッドの最大数についてのシステム全体の限度	スレッドの数についてのプロセス限度
HP-UX	無制限 (30 000 カーネル・スレッド)	30 000 (SAM ユーティリティーの構成可能カーネル・パラメーターを参照)

Java がスレッドにメモリーを割り当てる方法の詳細については、「*CICS Transaction Gateway: Gateway プログラミング*」の『パフォーマンスの問題』を参照してください。

外部アクセス・インターフェース (EPI、ECI、ESI)

外部アクセス・インターフェースにより、非 CICS アプリケーションは、CICS トランザクションを開始するか、または CICS プログラムを呼び出すことで、CICS リソースにアクセスし、更新することができます。 CICS 通信機能と共に使用すれば、非 CICS プログラムは、どの CICS システム上のリソースにもアクセスし、それらを更新することができます。これにより、CICS アプリケーションの GUI (グラフィカル・ユーザー・インターフェース) フロントエンドの開発、 CICS システムと非 CICS システムとの統合などの活動がサポートされます。

外部表示インターフェース (EPI) では、既存の CICS システムまたは新規のアプリケーションの GUI を開発することができます。これは、特に、変更する必要がない、既存の CICS トランザクションに対して新しい GUI フロントエンドを開発する場合に役立ちます。アプリケーションは、EPI を使用して CICS トランザクションとやり取りすることができ、クライアント・システムの表示機能を活用して、エンド・ユーザーとやり取りすることができます。

CICS システムと非 CICS システムとの統合では、非 CICS システムのプログラムと CICS プログラム間でユーザー定義データを受け渡しすることが必要となり、この場合には外部呼び出しインターフェース (ECI) が役立ちます。

このようなケースでは、EPI と ECI のどちらを選択すべきかは、必ずしも明確に定義されていません。というのは、どちらのインターフェースも、非 CICS アプリケーションと CICS プログラム間のデータの受け渡しに使用できるからです。ただし、メカニズムは異なります。 EPI の場合は 3270 データ・ストリームであり、 ECI の場合は COMMAREA のアプリケーション定義形式です。

外部表示インターフェース (EPI)

EPI を利用すれば、接続先の CICS サーバー・システムで、非 CICS アプリケーション・プログラムを 3270 端末として表示することができます。 EPI アプリケーションおよび CICS 端末は両方とも、 CICS サーバーのトランザクションをスケジュールすることができます。アプリケーションは、同時に複数のサーバーの機能を使用することができ、まるで別々の 3270 端末であるかのように動作することができます。アプリケーションは、CICS トランザクションをスケジュールできるので、これらのトランザクションにとって、このアプリケーションは基本的な機能です。

EPI を介してアクセスをサポートする CICS サーバーにより、サーバーで実行されているその他の CICS トランザクションも、 CICS START コマンドを使

用して、非 CICS アプリケーションを開始端末として使用することができます。非 CICS アプリケーションが EPI を介して CICS サーバーのトランザクションを開始すると、3270 データ・ストリームおよびイベントが、サーバーとアプリケーションの間で受け渡しされます。アプリケーションは、アプリケーションの操作環境に合った方法で、ユーザーに端末入出力の内容を表示することができます。トランザクションは、標準のトランザクション経路指定により、他の CICS システムに経路指定することができます。他の CICS システム上のリソースは、機能シップによりアクセスすることができます。

サーバー・トランザクションは、3270 入出力を使用する既存のトランザクションでも構わないことに注意してください (多少の制限はあります)。

外部呼び出しインターフェース (ECI)

ECI を利用することにより、非 CICS アプリケーションは CICS サーバーの CICS プログラムを呼び出すことができます。アプリケーションは同時に複数のサーバーに接続することができると共に、同時に複数のプログラム呼び出しを未解決にしておくことができます。

CICS プログラムは、端末入出力を実行することはできませんが、その他のすべての CICS リソースにアクセスし、それらを更新することはできます。同じ CICS プログラムを、非 CICS アプリケーションは ECI を使用して、 CICS アプリケーションは EXEC CICS LINK を使用して呼び出すことができます。データは、CICS と同じように、COMMAREA によって 2 つのプログラム間で交換されます。ユーザーは、COMMAREA データの長さを指定して、パフォーマンスを最適化することができます。

呼び出しは、同期でも非同期でも可能です。同期呼び出しは、呼び出し先プログラムが終了すると制御を戻すので、戻された情報はすぐに利用することができます。非同期呼び出しは、呼び出し先プログラムの終了を参照せずに制御を戻すので、アプリケーションは、情報が利用できるようになったら通知するよう求めることができます。

また、呼び出しは *延長* することもできます。すなわち、単一の作業論理単位で 2 つ以上の連続した呼び出しを扱うことができます。ただし、各作業論理単位ごとに 1 つの呼び出ししか活動状態になることはできません。非同期呼び出しを使用すれば、アプリケーションは同時に数多くの作業論理単位を管理することができます。

呼び出し先プログラムは、独自のシステム上のリソースを更新することができます。他のシステム上の CICS プログラムを呼び出す場合には DPL (distributed program link) を使用し、他の CICS システム上のリソースにアク

セスする場合には、機能シップ、分散トランザクション処理 (DTP)、あるいは (CICS® Transaction Server for OS/390® または CICS® Transaction Server for VSE/ESA™ 環境では) フロントエンド・プログラミング・インターフェース (FEPI) を利用します。

外部アクセス・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*」をご覧ください。

外部セキュリティー・インターフェース (ESI)

ESI により、非 CICS アプリケーションは、拡張プログラム間通信機能 (APPC) のパスワード有効期限管理 (PEM) が提供するサービスを呼び出すことができます。

APPC PEM と CICS により、 APPC アーキテクチャーに基づくサインオン・トランザクションのサポートが提供されます。それは CICS サーバーへユーザー ID をサインオンし、以下の方法によるパスワード変更の要求を処理します。

- ユーザーを識別し、そのユーザーの身元を認証する
- 認証の際に、特定のユーザーに、パスワードの有効期限が切れたことを通知する
- パスワード失効の際 (あるいはその前に)、ユーザーにパスワードの変更を行わせる
- 現行のパスワードがいつまで有効かを、ユーザーに知らせる
- 特定のユーザー ID を使用してサーバーにアクセスしようとする無許可の試行に関する情報を提供する

APPC PEM を使用するには、 CICS ユニバーサル・クライアントが APPC を介して CICS に接続されている必要があります。リソース・アクセス管理機能 (RACF®) などの外部セキュリティー管理プログラム (ESM) を、 CICS サーバーで使用できることも必要です。 ESI 呼び出しを ECI または EPI アプリケーションに組み込むことができます。 **CICS_EciListSystems** および **CICS_EpiListSystems** 機能によって戻される CICS サーバーだけが受け入れ可能です。

ネットワーク・セキュリティー

CICS Transaction Gateway は SSL (Secure Sockets Layer) および HTTPS プロトコルの使用をサポートすることにより、インターネット操作の成功に不可欠であるセキュリティー通信を提供します。

CICS Transaction Gateway のネットワーク・セキュリティーとそのインプリメンテーションについては、 69ページの『第5章 セキュリティー』で詳しく論じられています。以下の節では、SSL および HTTPS が提供する機能を要約します。

SSL (Secure Sockets Layer)

SSL は、インターネット上でセキュリティーおよびプライバシーを実現するために開発された**ハンドシェーク・プロトコル**です。 SSL プロトコルを使用することにより、以下の点が保証されます。

機密性 (Confidentiality)

クライアントとサーバー間で交換されるデータは暗号化されるので、そのクライアント (アプリケーションまたはアプレット) とそのサーバー (CICS Transaction Gateway) にのみ、そのデータは意味があります。

SSL は保護メカニズムとして公開鍵暗号化を使用して、サーバーとクライアントの間で秘密鍵を配布します。公開鍵暗号化は、暗号化と復号に対称キーのペアを使う技法です。 SSL の場合、秘密鍵 (対称キー) が、クライアントとサーバーの間で (公開鍵暗号化を使用して) 渡され、次いでそれが、 SSL 接続を介するすべてのトラフィックの暗号化と復号に使用されます。この暗号化は、盗み読みしようとする他者からデータを保護するものであり、データの復号に必要な秘密鍵を持っている人は他にいないことになります。これにより、クレジット・カード番号などの秘密情報を安全に転送することができます。

保全性 (Integrity)

メッセージ転送には、セキュア・ハッシュ・アルゴリズムをベースとするメッセージ整合性チェックが組み込まれます。このアルゴリズムは、メッセージが送信されたとき、およびそのメッセージが受信されたときに実行されます。 2 つのハッシュ値が一致しないと、受信側には、メッセージが変更された可能性があることを示す警告が発行されます。

アカウンタビリティー (Accountability)

アカウンタビリティーはディジタル・シグニチャーによって保証されるので、問題が生じた場合、だれに責任があるかを特定できます。

認証 (Authentication)

CICS Transaction Gateway の SSL プロトコルのインプリメンテーションは、*サーバー認証* を提供します。それにより、クライアントが CICS Transaction Gateway との接続を確立するとき、サーバーの詳細を認証することが必要になります。また、*クライアント認証* も使用可能にすることができます。この場合は、サーバーがクライアントの詳細について認証します。

認証メカニズムは、ディジタル認証 (X.509v3 認証) の交換に基づきます。これらのディジタル認証には、システム名や公開鍵などのエンティティーに関する情報や、サーバーのディジタル・シグニチャーが含まれます。ディジタル認証は、認証局 (CA) が発行し、CA の公開鍵を使用して暗号化されます。 CA の公開鍵を使用して認証を復号できる場合は、認証に含まれている情報が信頼できるものである、つまり、認証が実際にその所有を要求している人に属するものであることを認識していることになります。

HTTPS

現在の大半のブラウザーは、URL アクセス・メソッドである HTTPS をサポートしています。これは、SSL を使用して HTTP サーバーに接続するためのものです。 HTTPS (HTTP+SSL) は、安全なトランザクションを扱うための HTTP の変形です。

セキュア接続は一般に、「https://someAddress」のような URL を使用して確立されます。デフォルトの HTTPS ポート番号は 443 であり、これは Internet Assigned Numbers Authority によって割り当てられた番号です。

鍵と認証

SSL プロトコルは公開鍵による暗号を使用しており、 ISO 認証フレームワークでの使用が推奨されていて、X.509 プロトコルとも呼ばれています。このフレームワークは、ネットワーク間の認証のためのものです。

X.509 の最も重要な部分は、公開鍵証明書の構造です。委託されている認証局 (CA) が、各ユーザーに一意の名前を割り振り、名前およびユーザーの公開鍵を含む、署名された証明書を発行します。

CICS Transaction Gateway では、外部署名された証明書を CA から取得するか、自分自身を CA として確立して、「自己署名された」X.509 証明書を発行することができます。外部署名された証明書はインターネットでの使用に、より適していますが、自己署名された証明書でも、組織内での内部使用には適切です。

X.509 ディジタル証明書は、Java クラス・ファイルに「カプセル化」されるので、 SSL プロトコルおよび HTTPS プロトコルで使用することができます。これらの Java クラス・ファイルのことを *KeyRing* クラスと言います。

CICS Transaction Gateway には、KeyRing クラスの管理のための iKeyman ツールが組み込まれています。このツールを使用して、KeyRing ファイルの作

成、証明書の生成、KeyRing への証明書保管、その他各種の管理機能を実行することができます。詳細については、69ページの『第5章 セキュリティー』を参照してください

セキュリティー出口

ユーザーが公開鍵の暗号化などのセキュリティー操作を定義することができる、セキュリティー出口も用意されています。これらは、データ圧縮にも使用することができます。これらの機能をデモンストレーションしたソース・ファイル例が用意されています。

さらに、セキュリティー出口メカニズムを使って、クライアント認証が使用可能なときの X.509 クライアント証明書の認証 / 解析を行うことができます。

詳しくは、「*CICS Transaction Gateway: Gateway プログラミング*」を参照してください。

CICS Transaction Gateway およびオブジェクト・リクエスト・ブローカー

Web サーバーの中には、オブジェクト・リクエスト・ブローカー (ORB) を持つものがあります。 ORB は、オブジェクト管理グループ (OMG) の共通オブジェクト・リクエスト・ブローカー (CORBA) 規格に従っています。そのような Web サーバーがある場合、 CORBA Internet InterORB Protocol (IIOP) を使用してリモート・ブラウザーから呼び出せるサーブレット・オブジェクトを開発することができます。IIOP は HTTP を使用してブラウザーに送信されます。

ブラウザーには ORB を組み込んでいるものもあります。そうしたブラウザーでは、IIOP 要求を発行するプログラムを実行することができます。この能力は、Java RMI (リモート・メソッド呼び出し) を使用して通信する Java アプレットによって使用される、バインディング・メカニズムであり、最終的に IIOP にマップされます。この技法によって、アプレットは IIOP を使用してサーブレットと通信し、サーブレットが CICS トランザクションを呼び出すことができます。

その他の機能

Host on-Demand もサポートしており、 Java 対応 Web ブラウザー上で EPI を介して CICS 3270 アプリケーションにアクセスできる手段を提供します。IBM eNetwork™ Host on-Demand バージョン 2.0 以降が必要です。

ローカル CICS Transaction Gateway もサポートされています。これにより、Java プログラムは、ネットワークを使用することなく、ローカルにインストールされている CICS Transaction Gateway と通信することができます。ローカルの CICS Transaction Gateway の構成方法については、 67ページの『ローカルの CICS Transaction Gateway サポートを使用するための準備』 を参照してください。

ユーロのサポート

CICS Transaction Gateway はデータ・フローの中のユーロ文字の使用をサポートします。CICS Transaction Gateway クラスの CicsCpRequest は Java エンコードの名前を戻そうとします。この名前は、ユーロ対応の JDK に適しており、 CICS ユニバーサル・クライアントが CICS サーバーに渡すコード・ページ/CCSID (コード化文字セット識別子) に相当します

たとえば、CICS ユニバーサル・クライアントが CICS サーバーに、コード・ページとして 5348 (コード・ページ 1252 のユーロ対応バージョン) を渡した場合、 Java クライアントに戻される Java エンコードは Cp1252 になります。 JDK 自体がユーロ対応であれば、Java エンコード Cp1252 はユーロ対応です。 Java プログラム用のユーロ通貨記号サポートの詳細については、該当する Java の資料を参照してください。

アクセス容易性

CICS Transaction Gateway のアクセス容易性については、Windows オペレーティング・システムでのみテストされています。本製品は、構成ツールを除き、アクセス可能です。これに対しては、ユーザーが手動で ini ファイルを編集することをお勧めします。ini ファイルは、アクセスに関する要件を満たすテキスト・エディターに読み込むことができます。この問題は、将来のリリースで対処される予定です。

第2章 インストール前の計画

この章では、サポートされている Web サーバーとブラウザーを含む、ハードウェアおよびソフトウェア要件について説明します。これらは、CICS Transaction Gateway のインストール計画に役立ちます。 CICS Transaction Gateway が接続可能な CICS サーバーをリストします。

ハードウェア要件およびソフトウェア要件の最新の変更については、製品の README ファイルを参照してください。

ハードウェア要件

CICS Transaction Gateway は、対応するオペレーティング・システム、およびその他の前提条件ソフトウェアを実行できるハードウェア上で実行されます。

CICS Transaction Gateway のハード・ディスク要件は、(英語のみの資料を含めて) **30MB** です。

インストール中に使用される一時ファイル用にさらに 30MB 必要になります。このうちの少なくとも 12MB が、Windows ドライブに必要です。

サポートされているソフトウェア

以下の製品がサポートされています。ここに示されている限りでは、以下の製品に対して CICS Transaction Gatewayはテスト済みです。

表 2. サポートされている製品

オペレーティング・システム	CICS Transaction Gatewayは、次のオペレーティング・システムで実行し、テスト済みです。 • AIX V4.3.3 + Service Pack 25 (UNIX-UNIX のコピー・プログラム bos.net.uucp を含む) • HP-UX 11.0 • Linux for S/390® - SuSe 7.0 (カーネル・レベル 2.2.16) • OS/390 V2.8 • OS/390 V2.10 • Solaris 7 (32 ビット・モードおよび 64 ビット・モード) • Solaris 8 (32 ビット・モードおよび 64 ビット・モード) • TurboLinux Server 6 for zSeries および S/390 • Windows NT Workstation V4.0 + Service Pack 6a、または Windows NT Server V4.0 + Service Pack 6a **注:** Windows NT 端末サーバーはサポートされていません。 • Windows 2000 Professional + Service Pack 1 または Windows 2000 + Service Pack 1 (Windows 2000 端末サービス機能付き) • z/OS V1
Web ブラウザー	• HTML/HTTP 機能: HTML V1.0 をサポートするブラウザーであれば、本製品で動作します。 • Java™ 機能: JDK™ 1.1 に準拠する Web ブラウザーであれば、本製品で動作します。 CICS Transaction Gatewayは、以下のブラウザーでテスト済みです。 • HotJava™ ブラウザー V 1.1 (Solaris 版) • Microsoft® Internet Explorer 5.0 • Netscape Communicator 4.7 および 6.0 **注:** Netscape Communicator ブラウザーで提供されている JVM 内で実行されるアプレットは、 CICS Transaction Gateway と接続することができません。アプレットにはメッセージ番号 CCL6652E が戻されます。この問題に対しては、現在、解決策はありません。

表 2. サポートされている製品 (続き)

Telnet クライアント	Telnet で CICS クライアントを実行する場合は、telnet クライアントによっては、表示に問題があるものがあります。たとえば、Telnet セッションがある長さ以上のメッセージ行を切り捨てることがあります。通常これは、使用している Telnet クライアント、あるいは Telnet の端末のタイプに関する問題です。この問題に対しては、現在、解決策はありません。
JDK のレベル	CICS Transaction Gatewayは、以下の JDK レベルをサポートしています。 • OS/390: IBM Java SDK 1.3 • AIX: IBM Java SDK 1.3 (サービス・リリース 6 以上) • HP-UX: Java SDK 1.3 • Linux for S/390: IBM Java SDK 1.2.2 • Solaris: Sun Java SDK 1.3 • Windows: IBM Java SDK 1.3 (サービス・リリース 6 以上) • OS/390 を除くすべてのオペレーティング・システムでは、IBM Java SDK 1.2.2 (サービス・リリース 10 および 10a) もサポートされています。 **注:** CICS Transaction Gateway は、JDK 1.1 準拠の Web ブラウザーで実行するアプレットをサポートしています。

表 2. サポートされている製品 (続き)

CICS サーバー	CICS サーバーは、リアルタイムでマルチユーザー・アプリケーションを実行し、関連するリソースおよびデータを管理します。CICS Transaction Gatewayは、以下のサーバーでテスト済みです。 • SNA および TCP62 を介した通信機能を持つ CICS® for MVS/ESA™ V4.1 (サインオン機能を持つ端末に対して APAR PQ30167 を適用) • CICS® for OS/400® V4.4 • CSD 3 を備えた CICS® Transaction Server for OS/2® V4.1 • CICS® Transaction Server for OS/390® V1.2 (サインオン機能を持つ端末に対して APAR PQ30168 を適用) • CICS® Transaction Server for OS/390® V1.3 (サインオン機能を持つ端末に対しては APAR PQ30168 を、EXCI に対しては OS/390® APAR PQ38644 を適用) • CICS® Transaction Server for VSE/ESA™ 1.1.0 (サインオン機能を持つ端末に対して APAR PQ30170 を適用) • CICS® Transaction Server for VSE/ESA™ 1.1.1 • CICS/VSE 2.3 (サインオン機能を持つ端末に対して APAR PQ30169 を適用) • TXSeries V4.2 + PTF 9 (HP-UX) • TXSeries V4.3 + PTF 4 (Windows® NT、AIX®、Solaris) • VisualAge® CICS Enterprise Application Development (OS/2® 版および Windows NT 版) (アプリケーション開発のみ) V3.1 + CSD 3
Web サーバー	CICS Transaction Gatewayは、以下のものでテスト済みです。 • Apache HTTP サーバー 1.3.12 (SuSe V7.0) • Domino™ Go Webserver V4.6.2 (Windows、AIX および Solaris) • IBM HTTP Server 1.3 (Windows、AIX および Solaris) • Microsoft® Internet Information Server V5.0 (Windows)

表 2. サポートされている製品 (続き)

アプリケーション・サーバー	CICS Transaction Gatewayは、以下のものでテスト済みです。 • Windows および AIX 上の IBM WebSphere™ Application Server V3.5.3 Enterprise Edition (PTF 3 付き) • Windows および AIX 上の IBM WebSphere™ Application Server V3.5.3 Advanced Edition (PTF 3 付き) **注:** 端末サーブレット (およびサーブレットのサンプル) も、上記のアプリケーション・サーバーでテスト済みです。
TCP/IP 通信	オペレーティング・システムは TCP/IP をサポートしています。
TCP62 通信	CICS Transaction Gatewayは TCP62を サポートしています。
コンパイラーおよびアプリケーション開発ツール	CICS Transaction Gatewayは、以下のコンパイラーおよびアプリケーション開発ツールでテスト済みです。 **AIX** • IBM C for AIX V4.4 • IBM C/Set++ for AIX V3.6.6 • IBM VisualAge C++ V4.0 • IBM VisualAge C++ V5.0 **HP-UX** • HP ANSI C B.11.01.07 • HP aC++ B.11.01.06 **Linux for S/390** • GNU コンパイラー C ランタイム V2.95.2 • GNU コンパイラー C++ ランタイム V2.95.2 **Solaris** Sun WorkShop C++ V5.0 **注:** C++ コンパイラーには、使用可能なサービス・パッチをすべて適用する必要があります。 **Windows** • IBM VisualAge for Java V3.5.3 • IBM VisualAge Interspace™ V5.2 • Microsoft Visual C++ V6.0 • Microsoft Visual Basic V6.0 • Microsoft Visual Basic Script (VBScript) V5.0 **注:** CICS Transaction Gatewayは、MTS を備えた COM ライブラリー、あるいは COM+ の MTS コンポーネントをサポートしていません。

表 2. サポートされている製品 (続き)

その他のツール	CICS Transaction Gatewayは、以下のツールをサポートしています。 • Adobe Acrobat

第3章 インストール

この章では、CICS® Transaction Gateway for HP-UX のインストール方法について説明します。

CICS® Transaction Gateway for HP-UX は、実行可能スクリプト・ファイルを含む圧縮 tar ファイルとして配布されています。このスクリプト・ファイルを実行すると、ライブラリー・ファイル、コマンド、メッセージ、およびカスタマイズ・ファイル (.INI) が作成されます。

CICS® Transaction Gateway for HP-UX のインストール

CICS® Transaction Gateway for HP-UX をインストールする場合は、以下の手順に従ってください。

1. ルートとしてログインします。
2. 配布メディア上で tar ファイルを探します。その名前は ctg-400h.tar.Z です。
3. ファイル ctg-400h.tar.Z. を、ハード・ディスク上の作業ディレクトリーにコピーします。
4. ファイルを解凍して、中のファイルを取り出します。

   ```
   uncompress ctg-400h.tar
   tar -xvf ctg-400h.tar
   ```

   これにより、現行のディレクトリーに **ctg_install** というスクリプト・ファイルが作成されます。
5. 次のように入力して、CICS Transaction Gateway をインストールします。

   ```
   ctg_install
   ```

注: インストール・プロセスでは *uudecode* 関数が使用されます。この関数は、*UNIX 間コピー・プログラム*のパーツです。CICS Transaction Gateway をインストールする前に、使用しているシステムにこのプログラムが含まれていることを確認してください。

注: 大きなクライアント・データ・フローを扱えるように HP-UX システムを構成するためのアドバイスについては、*CICS Transaction Gateway: HP-UX クライアント管理*の『UNIX® システムにおける内部クライアント通信』を参照してください。

各国語サポート

CICS® Transaction Gateway for HP-UX でユーザー・メッセージを表示するときの言語は、以下のコマンドを入力することで選択できます。

```
mkclimsgs XX Code Set
```

ここで、*XX* は 2 文字のメッセージ言語です。使用可能な言語のリストを取得するには、パラメーターを指定せずに次のコマンドを入力します。

```
CICS Client for HP-UX - User messages
-------------------------------------

    Language                Locale     Code Set
    ---------------------  ---------   ------------
us  US English               en_US      iso81
                            EN_US      utf8
fr  French                  fr_FR      iso81
                            FR_FR      utf8
de  German                  de_DE      iso81
                            DE_DE      utf8
it  Italian                 it_IT      iso81
                            IT_IT      utf8
es  Spanish                 es_ES      iso81
                            ES_ES      utf8
tr  Turkish                 tr_TR      iso89
                            TR_TR      utf8
jp  Japanese                ja_JP.SJIS sjis
                            ja_JP      eucJP
                            JA_JP      utf8
kr  Korean                  ko_KR      eucKR
                            KO_KR      utf8
cn  Simplified Chinese      zh_CN      hp15CN
                            ZH_CN      utf8

Syntax: mkclimsgs <Language> <Code Set>
```

リモート・システムからの X-Windows の使用

たとえば 構成ツールと iKeyman にアクセスする場合のように、リモート・システムから X-Windows を使用するときは、DISPLAY 環境変数を設定して、そのシステムでアプリケーションが独自のウィンドウを表示できるようにする必要があります。

表示システム (ウィンドウを表示する予定のシステム) で、次のコマンドを入力します。

```
xhost +appl
```

ここで、*appl* は、アプリケーションを実行するために使用されるシステムのネットワーク名です。

アプリケーション・システムで、アプリケーションを実行する前に、次のコマンドを入力します。

```
DISPLAY=disp:0
```

続いて、

```
export DISPLAY
```

と入力します。ここで、*disp* は、ウィンドウを表示する予定のシステムのホスト名です。 (そのあとに、コロンと 表示 ID (通常は 0) を付けます。)これで、アプリケーション・ウィンドウが *disp* システムに表示されます。

CICS Transaction Gateway のアンインストール

CICS Transaction Gateway をアンインストールするには、以下のように入力します。

```
mkcicscli -u
```

第4章 構成

この章では、次の内容について説明します。

- 『CICS Transaction Gateway のプログラミング環境の構成』
- 32ページの『構成ツールの使用』
- 54ページの『構成変換ツール』
- 55ページの『構成ファイルの編集』
- 58ページの『構成ファイルおよびマッピング・ファイルの名前変更』
- 59ページの『クライアント・キーボード・マップの構成』
- 63ページの『画面のカラーのカスタマイズ』
- 67ページの『ローカルの CICS Transaction Gateway サポートを使用するための準備』

CICS Transaction Gateway のプログラミング環境の構成

Java 仮想マシン (JVM) は CLASSPATH 環境変数を使用して、クラスと、クラスの入った zip または jar アーカイブを検出します。 JVM がクラス・ファイルにアクセスできるようにするために、クラス・ファイルまたはアーカイブの入ったディレクトリーの絶対パスを指定する必要があります。

CICS Transaction Gateway と一緒に使用するよう Java アプリケーションをコンパイルして実行するには、 CLASSPATH 環境変数に以下の絶対パスを追加してください。

- ctgclient.jar
- ctgserver.jar (ローカル・ゲートウェイを使用している場合)
- cfwk.zip (ゲートウェイとの SSL 接続を使用している場合)

これらのアーカイブは、CICS Transaction Gateway をインストールしたディレクトリーの classes サブディレクトリーにあります。

HP-UX での CLASSPATH の設定

CLASSPATH 環境変数を設定するには、次のようなシェル・コマンドを使用します。

```
export CLASSPATH=.:/opt/classes:/opt/ctg/classes/ctgclient.jar:${CLASSPATH}
```

このシェル・コマンドはクラスを、現行ディレクトリー、ディレクトリー /opt/classes、および JVM がアクセス可能なアーカイブ /opt/ctg/classes/ctgclient.jar に作成します。

シェル・コマンドをユーザーのホーム・ディレクトリー内の .profile ファイルに入れることもできます。そうすれば、システムにログオンするときに、CLASSPATH 環境変数が自動的に設定されます。

CLASSPATH 環境変数の値を表示するには、次のシェル・コマンドを使用します。

```
echo $CLASSPATH
```

構成ツールの使用

構成ツールを使用して、CICS Transaction Gateway および CICS ユニバーサル・クライアント用の構成パラメーターを設定します。

構成ツールを立ち上げる前に、次の情報を入手してください。

- CICS サーバーとの接続に使用するプロトコル (TCP/IP、TCP62)
- サーバーのホスト名または IP アドレス
- CTG が接続する必要のあるサーバーにおけるポート番号
- Gateway で使用するプロトコル (TCP、SSL、HTTP および HTTPS)

構成ツールをリモートで使用するには、28ページの『リモート・システムからの X-Windows の使用』で説明されているコマンドを使用して、表示をエクスポートする必要があります。

構成ツールを開始するには、**ctgcfg** コマンドを入力します。

タスクガイド

初めて構成ツールを実行する場合は、タスクガイドを使用すると、システムに関する基本情報を定義する際に役立ちます。使用している CICS サーバーの詳細を入力し、使用予定のプロトコルを指定することができます。ここで入力する設定値はすべて、構成ツールを使用して後に変更することができます。タスクガイドの開始画面で「キャンセル (Cancel)」をクリックすると、タスクガイドを飛ばして直接構成ツールに進むことができます。

タスクガイドで画面を完了するためのヘルプについては、オンライン・ヘルプまたは 33ページの『構成ツールのインターフェース』の関連箇所を参照してください。

構成ファイル

構成の詳細は、デフォルトでは、CTG をインストールしている bin サブディレクトリーの CTG.INI ファイルに保管されます。このファイルを編集する場合は、構成ツールだけを使用することをお勧めします。

CICSCLI 環境変数を設定すると、構成ファイルに別の場所を指定することができます。 CICS Transaction Gateway のログは、CTG.INI の場所を示しています。CTG.INI が見つからない場合は CICS Transaction Gateway は実行されません。

CTG.INI を bin ディレクトリーで見つけることができない場合、タスクガイドも立ち上がります。構成ファイルに対して別の場所を指定している場合に、既存の構成を編集するときは、「**キャンセル (Cancel)**」をクリックしてタスクガイドを飛ばしてください。構成ツールが起動した後、構成ファイルをオープンしてください。

構成ツールのインターフェース

画面には次の 4 つの領域があります (以下で個別に説明します)。

1. 「メニュー (Menu)」バー (「ファイル (File)」、「編集 (Edit)」、「ツール (Tools)」および「ヘルプ (Help)」プルダウン・メニューで構成されています)
2. 「新規 (New)」、「オープン (Open)」、「保管 (Save)」、「切り取り (Cut)」、「コピー (Copy)」、「貼り付け (Paste)」、「新規サーバー (New Server)」、「トレース設定 (Trace Settings)」アイコンを持つツールバー
3. ツリー構造
4. 「設定 (Settings)」パネル

5 番目の領域であるステータス・バーには、使用している構成ファイルおよびオペレーティング・システムが表示されます。ステータス・バーは表示専用です。

4 つの入力領域の間を移動するには、F10 キー、Esc キー、およびタブ・キーを使用します。

- メニューをアクティブにする場合は F10 を押します。
- 選択をせずにプルダウン・メニューをクローズする場合は Esc キーを押します。
- ツールバー、ツリー構造、および設定パネルの間を移動する場合はタブ・キーを使用します。タブを繰り返し押すと、設定パネルのすべてのオブジェクト、次にすべてのアイコンへと移動し、最後にツリー構造に戻ります。

- タブ・キーを押しても次の領域に移動できない場合は、F10、Esc、と押してから、もう一度 タブを押してください (メニューをアクティブにしてからクローズするため)。

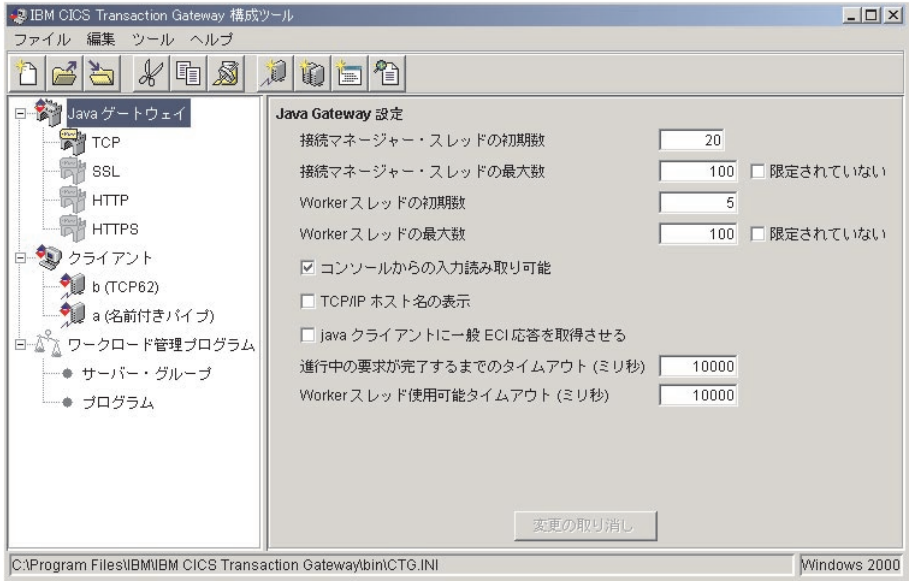


図 4. 構成ツール

注: ご使用のプラットフォームでは、表示される構成ツールは、図4 と全く同一にはならない場合があります。

ツリー構造

ツリー構造 (図4 を参照) では、構成のすべての設定値に移動することができます。ルート・ノードのタイプとしては、以下のものがあります。

「Java Gateway」
4 つのサブノードから成ります。CICS Transaction Gateway が使用できる各プロトコル (TCP、SSL、HTTP、および HTTPS) ごとに 1 つのサブノードがあります。

「クライアント (Client)」
それぞれのサーバー定義ごとにサブノードがあります。

キー操作によるアクセス: ノードおよびサブノードに移動するには、上矢印および下矢印キーを使用します。構成ツールを使用すると、すべてのノードが展開されます。ノードを縮

小するには、その上に移動してから左矢印キーを押します。そのノードを展開するには、右矢印キーを押します。

メニュー・バー

メニュー・バーには、**ファイル、編集、ツール、**および**ヘルプ**の各プルダウンがあります。

「**ファイル (File)**」プルダウンには、以下のオプションがあります。

「新規（New)」
新しい構成を作成します。

「オープン (Open)」
既存の構成をオープンします。

「保管 (Save)」
現在の構成を保管します。新しい構成が作成された場合、ファイル名は CTG.INI です (bin サブディレクトリーに入る)。

「別名保管 (Save As)」
構成ファイルのデフォルト・パスおよび名前をオーバーライドすることができます。

「終了 (Exit)」
構成ツールを終了します。構成を保管するかどうかを尋ねてきます。

「**編集 (Edit)**」プルダウンには、「**切り取り (Cut)**」、「**コピー (Copy)**」および「**貼り付け (Paste)**」オプションがあり、それぞれ標準のテキスト機能を実行します。

「**ツール (Tools)**」プルダウンには、以下のオプションがあります。

「トレース設定 (Trace Settings)」
トレース設定ダイアログを表示します。

「新規サーバー (New Server)」
新規サーバー定義を作成するためのタスクガイドを開始します。

「サーバーの削除 (Delete Server)」
ツリー構造からサーバー・ノードを削除します。

「ヘルプ (Help)」プルダウンには、以下のオプションがあります。

「コンテキスト・ヘルプ (Context Help)」
現在の画面コンテキスト (たとえば特定の構成設定値など) に従って、オンライン・ヘルプ情報を表示します。

「目次 (Contents)」
オンライン・ヘルプ情報の内容リストを表示します。

「索引 (Index)」
オンライン・ヘルプ情報の件名の索引を表示します。

「製品情報 (About)」
構成ツールのバージョン番号およびそれ以外の情報を表示します。

ツールバー
ツールバーにはメニュー・バーと同じ機能があります。ただし、ツールバーには「**別名保管 (Save As)**」と「**終了 (Exit)**」オプションはありません。それぞれのアイコンには、吹き出しヘルプが用意されています。カーソルをアイコンに合わせると、そのオプションを説明するテキスト枠が表示されます。

設定パネル
ツリー構造の中のノードを選択すると、関係のある設定パネルが表示されます。これらのパネル内の設定は、CTG.INI ファイル内のパラメーターに対応しています。

それぞれの設定パネルには、「**変更のやり直し** (Undo Changes)」ボタンがあり、行った変更を元に戻すことができます。

CICS Transaction Gateway 設定の構成

「ゲートウェイの設定 (Gateway Settings)」パネルを表示するには、ツリー構造の中のゲートウェイ・ノードを選択します。これらの設定値は、CTG.INI ファイルの Gateway セクションのパラメーターにマップされます。

構成ツールを使用して、Gateway コマンド行オプションからも指定できるパラメーターの値を、事前設定することができます。

構成ツールを使用して定義されたパラメーターが、 Gateway 開始時に、関連したコマンド行オプションを介して指定された場合は、コマンド行の設定が***優先されます***。

一般 Gateway 設定

一般 Gateway 設定を以下に列挙します。

接続マネージャー・スレッドの初期数: ConnectionManager スレッドの初期番号を入力してください。デフォルトは 1 です。

この設定は、`ctgstart -initconnect` コマンドでオーバーライドできます。

接続マネージャー・スレッドの最大数: ConnectionManager スレッドの最大番号を入力してください。デフォルトは 100 です。

「**無制限 (Unrestricted)**」チェック・ボックスを選択すると、ConnectionManager スレッドの数には制限が適用されません。

この設定は、`ctgstart -maxconnect` コマンドでオーバーライドできます。

スレッド化の限度については、13ページの表1 を参照してください。

Worker スレッドの初期数: Worker スレッドの初期番号を入力してください。デフォルトは 1 です。

この設定は、`ctgstart -initworker` コマンドでオーバーライドできます。

Worker スレッドの最大数: Worker スレッドの最大番号を入力してください。デフォルトは 100 です。

「**無制限 (Unrestricted)**」チェック・ボックスを選択すると、Worker スレッドの数に制限は適用されません。

この設定は、`ctgstart -maxworker` コマンドでオーバーライドできます。

スレッド化の限度については、13ページの表1 を参照してください。

コンソールからの入力読み取り可能: コンソールからの入力の読み取りを可能にするには、このチェック・ボックスを選択します。

コンソールからの入力の読み取りはデフォルトで使用可能です。

この設定は、`ctgstart -noinput` コマンドでオーバーライドできます。

TCP/IP ホスト名の表示: メッセージに TCP/IP ホスト名を表示できるようにするには、このチェック・ボックスを選択します。

このオプションを使用すると、TCP/IP アドレスをメッセージ内にどのように表示するかを選択できます。デフォルトでは、CICS Transaction Gateway は数値形式で TCP/IP アドレスをメッセージに表示します。このオプションを使用可能にすると、CICS Transaction Gateway はドメイン・ネーム・システム (DNS)

を使用して、メッセージ内で数値の TCP/IP アドレスを記号の TCP/IP ホスト名に変換します。これによりメッセージが読みやすくなります。

注: システムによっては、このオプションを選択すると、重大なパフォーマンス低下を起こすことがあります。

Java クライアントに総称 ECI 応答を取得させる: Java クライアントが総称 ECI 応答を CICS Transaction Gateway から取得できるようにしたい場合には、このチェック・ボックスを選択します。

総称応答は、Call_Type: ECI_GET_REPLY または ECI_GET_REPLY_WAIT を使用して得られるものです。特定応答は、Call_Type: ECI_GET_SPECIFIC_REPLY または ECI_GET_SPECIFIC_REPLY_WAIT を使用して得られるものです。この設定はローカル Gateway には適用されません。

進行中の要求が完了する前にタイムアウト: ミリ秒で値を入力します。

Java クライアント・プログラムが Gateway から切断されても、Gateway はそのプログラムに代わって要求の処理を続けることができます。処理がまだ進行中の場合には、その Java クライアントのために要求を管理していた ConnectionManager は、このタイムアウト期間まで、進行中の要求が完了するのを待機します。この時間が過ぎてもまだ進行中の要求があると、ConnectionManager はその処理を続け、新規の接続で利用可能であるとして自分自身にマークを付けます。このタイムアウトは、デフォルトで 10000 ミリ秒に設定されていますが、値を入力してこのデフォルトをオーバーライドすることができます。

この値をゼロに設定すると、ConnectionManager は、進行中要求の完了を待機しません。

Worker スレッド使用可能タイムアウト: ミリ秒で値を入力します。

ConnectionManager は要求を受信したときに、Worker スレッドを割り当て、その要求を実行しなければなりません。しかし、タイムアウト期間内に Worker が使用可能にならない場合には、その要求を拒否するエラー・メッセージが送られ、要求は実行されません。このタイムアウトは、デフォルトで 10000 ミリ秒に設定されていますが、このデフォルトをオーバーライドする値を入力できます。

この値がゼロに設定されている場合、Worker をすぐに利用できないと接続は拒否されます。

TCP プロトコル設定

TCP 用の設定を表示するには、**TCP** サブノードを選択します。

プロトコル・ハンドラーを使用可能にする: このプロトコルを使用して CICS Transaction Gateway を構成するには、このチェック・ボックスを選択します。

ポート: プロトコルの TCP/IP ポート番号を入力:

TCP/IP のデフォルトのポートは 2006 です。

TCP プロトコルのこの設定は、 `ctgstart -port` コマンドによってオーバーライドできます。

ハンドラー・ウェイクアップ・タイムアウト: ミリ秒で値を入力します。

この設定は、プロトコル・ハンドラーがインバウンド接続の受信からウェイクアップする頻度を制御します。再始動時には Gateway が停止しているかどうかが検査されるので、この値は、Gateway を完全にシャットダウンするまでの時間に影響します。この値をゼロに設定すると、ハンドラーは新しい接続が受け入れられた場合しか再始動しないので、 Gateway はその時点まで完全にシャットダウンしません。

接続タイムアウト: ミリ秒で値を入力します。

新しい接続が受け入れられると、この値は、 ConnectionManager スレッドが使用可能になるまでプロトコル・ハンドラーが待機する時間を指定します。ConnectionManager スレッドがこの時間内に使用可能にならない場合には、接続が拒否されます。この値がゼロに設定された場合、 ConnectionManager をすぐに利用できないと接続は拒否されます。

アイドル・タイムアウト: ミリ秒で値を入力します。

この設定は、どれだけの期間接続を休止状態にしておけるかを指定します。アイドル・タイムアウト期間は、要求が最後に接続をフロー・ダウンした時からカウントされます。アイドル・タイムアウトが満了すると接続は切断されますが、接続のための作業がまだ進行中である場合は、接続されたままになる場合があります。この値が設定されていない場合、またはゼロに設定された場合、アイドル接続は切断されません。

クリーンアップ処理は、サーバーにこの機能のサポートがインストールされている場合にのみ、タイムアウト後に行われます。

Ping 時間頻度: ミリ秒で値を入力します。

この値は、Gateway が接続クライアントに PING メッセージを送信して、クライアントがまだアクティブかどうかを検査する頻度を指定します。次の PING メッセージを送信する予定の時刻まで PING 応答がない場合は、接続は切断されます。この場合も、その接続のための作業がまだ進行中の場合には、接続されたままになります (これは「**作動中接続のドロップ (drop working connections)**」値の設定によって異なります)。この値が設定されていないか、ゼロに設定されている場合には、PING メッセージは送信されません。

作業中接続のドロップ: このチェック・ボックスを選択すると、この接続のための作業がまだ進行中の場合**でさえも**、アイドル・タイムアウトまたは PING/PONG 障害によって接続を切断できることを指定します。

SO_LINGER 設定: このハンドラーが使用するすべてのソケットの SO_LINGER 設定を入力します。SO_LINGER 設定はソケットをクローズするための遅延値を秒で指定します。この値を入力していない場合、あるいはゼロに設定した場合には、このプロトコル・ハンドラーで使用されるどのソケットについても SO_LINGER が使用不可になります。

SO_LINGER が使用可能で、データ伝送が完了していない場合は、Close 呼び出しはデータが転送されるまで、または接続がタイムアウトになるまで呼び出しプログラムをブロックします。 SO_LINGER が使用不可の場合は、Close 呼び出しは呼び出し元をブロックせずに戻り、TCP/IP はデータをまだ送信しようとします。通常はこの転送は成功しますが、TCP/IP は指定された期間のみ送信要求を繰り返すため、それを保証することはできません。

クライアントがセキュリティー・クラスを使用するように要求: セキュリティー・クラスを使用する接続だけを Gateway が受け入れるようにする場合には、このチェック・ボックスを選択します。

Java クライアント・プログラムが Gateway に接続している場合は、接続で使用する必要があるセキュリティー・クラスのペアを指定することができます。ただし、デフォルトでは、 Gateway は、このセキュリティー・クラスのペアを指定しないプログラムからの接続も受け入れます。

どのセキュリティー・クラスを有効にするかは、 Gateway がアクセスできる *xxx*ServerSecurity クラスのセットを制御することによって制御できます。

CICS Transaction Gateway のセキュリティー出口の詳細については、19ページの『セキュリティー出口』を参照してください。

SSL プロトコル設定

SSL 用の設定を表示するには、**SSL** サブノードを選択します。その設定は、SSL は以下の設定も持っている点を除き、TCP の場合と同じです。

クライアント認証の使用: CICS Transaction Gateway のクライアント認証を使用可能にするには、このチェック・ボックスを選択します。デフォルトは、クライアント認証を使用不可にすることです。

クライアントの認証が使用可能になっている時に、 ssl: または https: ハンドラーに対して接続を試みると、クライアントが自身のクライアント証明書 (ディジタル ID としても知られる) をもっている必要があります。

クライアントがクライアント証明書を取得する方法については、 81ページの『SSL クライアントを構成する』 を参照してください。

KeyRing クラス名: サーバー KeyRing のクラス名を入力します。

サーバーの KeyRing は、このサーバーを接続先のクライアントに確認させる場合に使用する、有効な x.509 認証から成り立っていなければなりません。この KeyRing クラスは、この製品と一緒に提供される SSL ツールを使用して生成されます。

SSL とそれに関連した KeyRing クラスの詳細については、 69ページの『第5章 セキュリティー』を参照してください。

KeyRing パスワード: 作成プロセスの際に指定したサーバー KeyRing クラスのパスワードを入力します。

SSL 用のデフォルト・ポートは 8050 です。

HTTP プロトコル設定

HTTP 用の設定を表示するには、**HTTP** サブノードを選択します。その設定は TCP の場合と同じです。

HTTP 用のデフォルト・ポートは 8080 です。

HTTPS プロトコル設定

HTTPS 用の設定を表示するには、**HTTPS** サブノードを選択します。設定は、HTTP は**クライアント認証の使用**、**KeyRing クラス名**、および **KeyRing パスワード**の設定も行なえる点を除き、HTTP の場合と同じです。詳細については、『SSL プロトコル設定』 を参照してください。

HTTPS 用のデフォルト・ポートは 443 です。

クライアント設定の構成

「クライアント設定 (Client Settings)」パネルを表示するには、ツリー構造の中のクライアント・ノードを選択します。これらの設定値は、CTG.INI ファイルの Client セクションのパラメーターにマップされます。

デフォルト・サーバー
ドロップダウン・リストからデフォルト・サーバーを選択してください。

アプリケーション ID
8 文字以内を入力するか、あるいはこのフィールドをデフォルト値 * のままにします。

この値は、CICS ユニバーサル・クライアント・ワークステーションのアプリケーション ID を、 CICS サーバーでのシステムとして自動インストールされる形式で指定します。この名前は、CICS サーバー・システム内で固有でなければなりません。値 * は、固有であることが保証された名前を自動的に生成します。

注: クライアントを複数の CICS サーバーに自動インストールし、アプリケーション ID に特定の名前を入力する場合には、その名前は、それが接続されるすべてのサーバーについて固有でなければなりません。この名前が固有でない場合には、すでに別のクライアントが同じ名前でインストールされているために、サーバーへの接続の試みが拒否されます。名前 * を使用すると、クライアントは、各サーバーで異なった固有の名前で認識されます。

最大バッファー・サイズ
4 ～ 32 の範囲の K バイト数を入力します。デフォルトは 32 KB です。

この値は、アプリケーションまたは端末データが流れる伝送バッファーのサイズを指定します。この値は、起こり得る最大の COMMAREA または使用する端末入出力域 (TIOA) を入れるために十分な大きさでなければなりません。この値には、一部のプロトコルでクライアントに必要な 512 バイトのオーバーヘッドは含まれません。

クライアントがメモリー制約のある環境で実行されている場合にのみ、この設定値をデフォルトから変更してください。

端末出口
1 ～ 4 文字の文字ストリングを入力します。デフォルトは EXIT です。

どの時点においても、ドランザクション名が入力可能な位置にこのストリングが入力されると、これによって端末エミュレーターは終了します。このストリングにブランク文字を入れることはできません。

ストリングには、大文字小文字の区別があります。端末エミュレーターの CICS 端末定義に大文字変換がある場合には、このストリングを大文字で入力する必要があります。

最大サーバー数
1 ～ 256 の範囲の値を入力してください。デフォルトは 10 です。

この値は、クライアントから同時にアクセスできるサーバーの最大数を指定します。

最大要求数
1 ～ 10 000 の範囲の値を入力します。デフォルトは 256 です。

この値は、クライアント上で同時に実行できる項目の最大数を指定します。項目は以下のいずれかです。

- 端末エミュレーター
- EPI 端末
- ECI 作業単位
- ESI 作業単位

この値は、アプリケーションがエラーになっており、過剰な数の要求をサーバーに処理依頼するようなランナウェイ状態の検出に使用されます。他のオペレーティング・システムの制限 (たとえば、メモリーの制約または通信セッション) が有効となっている場合には、実際の限度は、この設定よりも小さくなる可能性があります。

印刷コマンド
長さが 1 ～ 256 文字の文字ストリングを入力します。

指定するストリングは、クライアントが実行中のオペレーティング・システム固有のコマンドです。クライアントで印刷要求が受信されると、クライアントは、印刷要求ごとに固有の名前をもった一時印刷ファイルを作成します。

一時ファイル名とともにパラメーター・ストリングが付加され、結果のコマンドが実行されます。たとえば、これによって印刷要求をファイルにコピーしたり、ローカル・プリンターに指示したり、文書への組み込み用にフォーマットするなどできます。

一時印刷ファイルの処理が終了した後で、これを削除するのは、印刷コマンドの責任です。

印刷コマンド (たとえば、**lpr**) に続けて削除のためのコマンド (**rm**) を実行するシェル・スクリプトを使用します。

詳細については、**印刷ファイル**の説明を参照してください。

印刷ファイル
長さが 1 ～ 256 文字の文字ストリングを入力します。

このオプションが適用できるのは、印刷コマンドの設定を省略した場合だけです。

指定するストリングは、クライアントで受信した印刷要求の出力先となるファイルを指定します。各印刷要求は、現行ファイルの終わりに付加されます。

注: この設定はデフォルトとしてだけ機能します。端末および印刷エミュレーターは、この値をオーバーライドするオプションを提供します。

コード・ページ ID オーバーライド
ユーザーのローカル・コード・ページ ID をオーバーライドするコード化文字セット ID (CCSID) を指示する値を入力します。

ユーザーのプラットフォームがユーロ通貨記号サポート用に更新されていて、CICS サーバーにユーロ通貨記号サポートがある場合には、この設定を使用しなければなりません。たとえば、Latin-1 諸国では、CCSID 値 858 を使用して、コード・ページ 850 がユーロ通貨記号サポートを含むことを指示します。コード・ページ 1252 の場合には、CCSID 値 5348 を指定します。

cicsterm は常に、**コード・ページ ID オーバーライド**設定で指定された値に関係なく、ワークステーションのローカル・コード・ページに基づいて文字を表示することに注意してください。

また、使用されているコード・ページ ID を変更するために CCSID を使用した場合に、前にサーバーに保管されたデータ中に、コード・ポイントによって別の文字になるような文字が入っていると、クライアントに検索された時に変更される可能性があります。

CCSID およびデータ変換の詳細については、129ページの『付録A. CICS ユニバーサル・クライアントのデータ変換』 を参照してください。

サーバー再試行間隔
秒数を入力してください。これは、クライアントが接続されていたサーバーへの、クライアントによる再接続の試行相互間の時間 (秒数) です。デフォルト間隔は 60 秒です。

その接続先サーバーがもうアクティブでないことにクライアントが気付いた時には、非アクティブになった 1 秒後にサーバーへの再接続が試みられます。その後は、**サーバー再試行間隔**で定義された間隔で、再試行が引き続き試みられます。

ログ・ファイル
問題診断に使用するログ・ファイルの名前を入力します。

指定しないと、ログ・ファイル名のデフォルトとして、/var/cicscli サブディレクトリーの CICSCLI.LOG が使用されます。

サーバー設定の構成

「サーバー設定 (Server Settings)」パネルを表示するには、ツリー構造の中のサーバー・ノードを選択します。これらの設定値は、CTG.INI ファイルの Server セクションのパラメーターにマップされます。

サーバー名
1 ～ 8 文字の名前を入力します。これは、クライアントにとってローカルなサーバーの、通信プロトコル独立名を提供します。

ECI、EPI、ESI、または端末エミュレーターからサーバーにアクセスする要求は、この名前を使用してサーバーを参照します。

記述
1 ～ 60 文字のサーバーの説明を入力します。この記述は任意指定です。

この記述は、CICS_EpiListSystems および CICS_EciListSystems 機能によって、クライアント上で実行中のアプリケーションに戻されます。 (「*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)。

初期トランザクション
1 ～ 128 文字のトランザクション ID を入力します。

このストリングには、大文字小文字の区別があり、端末エミュレーターがサーバーに接続された時に実行する初期トランザクション (および任意のパラメーター) を識別します。なにも入力しない場合には、初期トランザクションは実

行されません。このストリング中の先頭 4 文字または最初のブランクまでの文字がトランザクションとみなされます。残りのデータは、その呼び出し時にトランザクションに渡されます。

モデル端末定義
1 ～ 16 文字のストリングを入力します。

このストリングには、大文字小文字の区別があり、サーバーでのモデル端末定義の名前を指定しますが、これはクライアントから自動インストールされる端末の特性を識別します。このモデルがサーバーで見つからない場合、またはなにも入力しない場合には、デフォルトの端末定義が使用されます。このデフォルトは、サーバー固有です。

モデル端末定義の設定の解釈は、サーバー固有です。たとえば、TXSeries for AIX サーバーの場合には、この値は 1 ～ 16 文字で、モデルとして使用する CICS 端末定義項目の DevType になります。

大文字セキュリティーの使用
ECI アプリケーションからのユーザー ID またはパスワード、あるいはユーザー・プロンプトからのユーザー ID またはパスワードを、すべて大文字に変換するよう指定するには、このチェック・ボックスを指定します。

デフォルトではこの設定はオフです。

ネットワーク・プロトコル
サーバー接続に使用されるプロトコルをドロップダウン・リストから選択してください。

- TCP/IP
- TCP62

パネルに表示されるプロトコル設定値は、選択したプロトコルによって異なります。

TCP/IP 設定

ホスト名または IP アドレス: CICS サーバーを実行するホストの文字または数字の TCP/IP ID を入力します。たとえば、cicssrv2.company.com (HostName) または 9.20.4.1 (IPAddress) を入力します。

Hostnames は、ネーム・サーバーまたは etc サブディレクトリーのホスト・ファイルのいずれかによって、 IP アドレスにマップされます。ただし、IP アドレスは変更される場合もあるため、ホスト名を使用する方がよりよい方法です。

ポート: クライアントの接続先サーバーでのポート番号を定義する 0 ～ 65 535 の範囲の数値を入力します。デフォルト値は 0 です。

値 0 は、/etc ディレクトリー内のサービス・ファイルを使用して、 TCP プロトコルを使用する CICS サービスのポート番号を見つける必要があることを示しています。

サービス・ファイルに項目が全く見つからない場合は、値として 1435 が想定されます。これは、TCP/IP アーキテクチャーにおいて CICS ユニバーサル・クライアントに割り当てられたポートです。

接続タイムアウト: 接続の確立に許容される最大秒数を指定する 0 ～ 3600 の範囲の値を入力します。デフォルト値の 0 は、クライアントによる制限が設定されないことを意味します。

接続の確立に、指定した時間より長くかかった場合には、タイムアウトが起こります。 TCP/IP ソケットがクローズされ、 ECI_ERR_NO_CICS または CICS_EPI_ERR_FAILED のいずれかの戻りコードがクライアント・アプリケーションに渡されます。

クリーンアップ処理は、サーバーにこの機能のサポートがインストールされている場合にのみ、タイムアウト後に実行されます。

TCP/IP キープアライブ・パケットの送信: TCP/IP が定期的に KeepAlive パケットをサーバーに送信して、接続を検査したい場合には、このチェック・ボックスを選択します。

TCP62 設定

ホスト・ファイル: TCP62 CICS サーバーを構成する場合は、ローカル・ホスト・ファイルも更新する必要があります。ファイルへのパスは次のようになっています。

Unix システム

/etc/hosts

このファイルは、IP アドレスをホスト名にマップします。各項目は、それぞれ別の行に指定してください。項目の形式は次のようになります。

```
9.20.101.6          IYCQST34.GBIBMIYA.hursley.ibm.com
```

この項目の各エレメントについて以下で説明します。

9.20.101.6
CICS システムの IP アドレスです。これは 1 桁目から始まり、その他の要素とは少なくとも 1 つのスペースで分離する必要があります。

IYCQST34.GBIBMIYA.
パートナー LU 名を*逆* 順にしたものです。上の例では、GBIBMIYA.IYCQST34 がパートナー LU 名で、構成ツールで入力されたものです。

hursley.ibm.com
Anynet ドメイン・ネームの接尾部です。

パートナー LU 名: クライアント で SNA 構成に認識されたサーバーの LU 名を入力します。これは、たとえば、 ABC3XYZ4.PQRS1234 のように修飾した 17 文字の名前としなければなりません。

ローカル LU 名またはテンプレート: CICS サーバーへの接続時に使用する実 LU 名またはローカル LU 名用のテンプレートを入力します。

テンプレートを入力するときには、 **LU 名テンプレートの IP アドレス・マスク (任意指定)** フィールドに入力する必要があります。

テンプレートには、テンプレート置き換え文字、すなわちアスタリスク (*) が含まれ、LU 名の動的生成に使用されます。たとえば、テンプレート CTS8BC** は、 2 つのアスタリスクの置き換えによって名前が動的に生成されることを指示します。テンプレート、IP アドレス・マスク、およびワークステーション IP アドレスに基づいたアルゴリズムが使用され、固有のローカル LU 名、たとえば、CTS8BC38 が作成されます。

ローカル LU 名の名前動的生成によって、クライアント・マシンの IP アドレスに基づいて異なったローカル LU 名が作成されるので、すべての CICS クライアント・マシンで同じ構成を使用できます。したがって、VTAM® 内で複数の LU を定義する必要はありません。

数百のクライアントが LU 名の動的生成を使用して CICS サーバーに接続する場合には、ローカル LU 名にもっと多くのテンプレート置き換え文字を使用する必要があります。たとえば、CTS8**** とします。実際に CICS® Transaction Server for OS/390® は LU 名の最後の 4 文字を接続名として使用するので、これらの文字は固有でなければなりません。

LU 名テンプレートの IP アドレス・マスク (任意指定): IP アドレス・マスクをビッグ・エンディアンで、すなわち最大有効バイトを最初に入力します。これは、たとえば FFFFFF80 のような 16 進数でなければなりません。

このアドレス・マスクは、LU 名の動的生成に使用されます。詳細については、**ローカル LU 名またはテンプレート**の説明を参照してください。

このパラメーターは、**ローカル LU 名**がテンプレートでない場合には、無視されます。デフォルトは 00000000 です。

モード名: サーバーに接続する時に使用するモード名を示す 1 ～ 8 文字を入力します。 ModeName を指定しない場合には、デフォルト・モード名 TCP62 が使用されます。

最大論理 SNA セッション数、最大 SNA RU サイズ、および **SNA ペーシング・サイズ**設定を使用して、**モード名**で指定されたモードを定義できます。

共通 TCP62 設定

完全修飾 CP 名またはテンプレート: 開始するノードの完全修飾 CP 名または完全修飾 CP 名のテンプレートを入力します。

テンプレートを入力するときには、 **CP 名の IP アドレス・マスク (任意指定)** フィールドに入力する必要があります。

テンプレートには、テンプレート置き換え文字、すなわちアスタリスク (*) が含まれ、CP 名の動的生成に使用されます。たとえば、テンプレート USIBMJKA.CP62**** は、 4 つのアスタリスクの置き換えによって名前が動的に生成されることを指示します。テンプレート、IP アドレス・マスク、およびワークステーション IP アドレスに基づいたアルゴリズムが使用され、固有の CP 名、たとえば、USIBMJKA.CP62AH38 が作成されます。

CP 名の名前動的生成によって、クライアント・マシンの IP アドレスに基づいて異なった CP 名が作成されるので、すべての CICS クライアント・マシンで同じ構成を使用できます。したがって、VTAM に対して複数の CP を定義する必要はありません。

数百のクライアントが CP 名の動的生成を使用する場合には、もっと多くのテンプレート置き換え文字を使用する必要があります。たとえば、USIBMJKA.CP****** とします。

テンプレートのネット ID 部分 (最初の部分) には、テンプレート置換文字を含めてはなりません。

CP 名の IP アドレス・マスク (任意指定): IP アドレス・マスクをビッグ・エンディアンで、すなわち最大有効バイトを最初に入力します。これは、たとえば FFFF8080 のような 16 進数でなければなりません。

このアドレス・マスクは、CP 名の動的生成に使用されます。詳細については、**完全修飾 CP 名またはテンプレート**の説明を参照してください。

このパラメーターは、**完全修飾 CP 名またはテンプレート**がテンプレートでない場合には、無視されます。デフォルトは 00000000 です。

Anynet ドメイン・ネーム接尾部: AnyNet® ドメイン・ネーム接尾部を入力します。このパラメーターは、パートナー (**パートナー LU 名**) とともに、その LU と関連したホストの IP アドレスの判別に使用されます。

たとえば、ドメイン・ネーム接尾部が sna.ibm.com で、パートナー LU 名が NETID.LUA の場合には、ホスト名 LUA.NETID.sna.ibm.com に対して gethostbyname 呼び出しが出されて、 LU 名 NETID.LUA をもったノードの IP アドレスが取得されます。

デフォルトは sna.ibm.com です。

TCP62 ポート: 0 から 65535 の範囲のポート番号を入力して、クライアントが接続するサーバー上のポート番号を定義してください。デフォルト値は 0 です。この場合は TCP/IP サービス MPTN 用のポート (サービス・ファイルに定義されています) を使用する必要があることを示しています。

リモート・ノード非活動ポーリング間隔: 0 ～ 65535 の間の秒数を入力してください。デフォルト値は 60 秒です。値 0 を指定すると、ポーリングはオフになります。

この構成設定は、非アクティブ接続をポーリングするまでに CICS ユニバーサル・クライアントが待機する時間を制御します。

拡張 TCP62 設定

最大論理 SNA セッション数: 1 ～ 255 の範囲の値を入力してください。デフォルト値は 8 です。

このパラメーターは、論理 SNA セッションの数を定義するために TCP62 で使用されます。

最大 SNA RU サイズ: 256 ～ 4096 の範囲の値を入力してください。デフォルト値は 1024 です。

このパラメーターは、**モード名**設定によって指定されたモードの定義中に TCP62 で使用されます。

SNA ペーシング・サイズ: 0 ～ 63 の範囲の値を入力してください。デフォルト値は 8 です。

このパラメーターは、**モード名**設定によって指定されたモードの定義中に TCP62 で使用されます。

トレース設定

トレース設定を構成するには、「**ツール (Tools)**」メニューから「**トレース設定 (Trace Settings)**」オプションを選択します。

トレース設定

チェック・ボックスを選択して、トレースがオンになった時にトレースする CICS Transaction Gateway および CICS ユニバーサル・クライアント・コンポーネントを指定します。

すべてのトレース	すべてのコンポーネント
クライアント API レベル 1	クライアント API 層 (レベル 1)
クライアント API レベル 2	クライアント API 層 (レベル 1 および 2)
CICSCLI コマンド行	cicscli コマンド・インターフェース
CICSTERM および CICSPRINT	cicsterm および cicsprnt エミュレーター
CPP クラス	C++ クラス・ライブラリー
クライアント・デーモン	CICS ユニバーサル・クライアント・デーモン
トランスポート層	プロセス間通信
ゲートウェイ	CICS Transaction Gateway

JNI のトレース
JNI のトレースを可能にするには、環境変数 CTG_JNI_TRACE=FILENAME を設定します。ここで、FILENAME はトレースが書き込まれるファイルに対応します。

プロトコル・ドライバー・レベル 1
プロトコル・ドライバー (たとえば、TCP)。これは送信および受信されたデータをトレースし、障害についての補足情報を提供します。

プロトコル・ドライバー・レベル 2
これはプロトコル・ドライバー、および他のソフトウェア・コンポーネントを介する内部フローをトレースします。現在、これをサポートしているのは、CCLTCP62 プロトコル・ドライバーのみです。

注: 上記のどの設定を使用しても JNI のトレースを使用可能にすることはできません。 JNI のトレースを使用可能にするには、情報を保管するファイルを指すように環境変数 CTG_JNI_TRACE を設定する必要があります。

また、cicscli コマンドで -m パラメーターを使用することによってトレース・コンポーネント (ゲートウェイコンポーネントを除く) を指定できます。cicscli を使用して指定したコンポーネントのトレースは、構成ツールで指定したトレースをオーバーライドします。コンポーネントのトレースが cicscli コマンドまたは構成ツールによって指定されていない場合には、デフォルトのコンポーネント・セット (すなわち**プロトコル・ドライバー**、**クライアント・デーモン**、および**クライアント API レベル 1**) がトレースされます。いずれかのチェック・ボックスを選択した場合には、これがデフォルトのコンポーネント・セットをオーバーライドします。

API コンポーネントの場合には、トレースする情報のレベルを指定できます。**クライアント API レベル 1** チェック・ボックスは、基本的な API 関連情報 (たとえば、ECI、EPI、および ESI 呼び出しの前と後) のトレースを指定します。**クライアント API レベル 2** チェック・ボックスは、レベル 1 の項目に加えて追加の API トレース項目が作成されることを指定します。

トレースするデータ域の最大サイズを指定するために cicscli -d=*nnn* コマンドが使用されることに注意してください。予定されるデータよりも小さい *nnn* を設定すると、トレース・データが切り捨てられることがあります。

ゲートウェイ・トレース・ファイル
トレースが使用可能な場合に、トレース・メッセージを書き込むトレース・ファイルのパス名を入力します。パスを指定しないと、トレースは、CICS Transaction Gateway がインストールされている bin サブディレクトリーの CTG.TRC ファイルに書き込まれます。

また、ctgstart コマンドの -tfile オプションを使用して、トレース・ファイルを指定できます。

CICS Transaction Gateway が開始されるたびに、トレース・ファイルは付加されることなく、上書きされます。

パフォーマンス: Gateway のトレースをオンにすると、パフォーマンスに重大な影響が生じます。実稼働環境では Gateway のトレースを使用可能にすることはお勧めできません。

ゲートウェイのトレースのラップ・サイズ
0 ～ 1 000 000 K バイトの範囲の値を入力してください。デフォルトは 0 です。

この値はゲートウェイのトレース・ファイルが達するサイズを K バイトで指定します。この後に続くトレース項目はファイルの先頭から続けて書き込まれます。

クライアント・トレース・ファイル
トレースが使用可能な場合に、トレース・メッセージを書き込むトレース・ファイルのパス名を入力します。

常にファイル・タイプ .BIN (またはトレース・ファイルが折り返す場合には .WRP) が生成されるので、このファイル名に拡張子を入力する必要はありません。

パスを指定しないと、トレースは CICSCLI.BIN ファイルに書き込まれます。このファイルは /var/cicscli サブディレクトリーにあります。

パフォーマンスに与える影響を最小にするために、クライアントのトレース・ファイルはバイナリー形式で書き込まれます。このファイルを読み取るには、cicsftrc コマンドを使用して、それを ASCII に変換する必要があります。

トレースの詳細については、ご使用のプラットフォームの「*CICS Transaction Gateway クライアント 管理*」の『問題判別』を参照してください。

最大クライアント折り返しサイズ
0 ～ 999 K バイトの範囲の値を入力してください。デフォルトは 0 です。

この値は、折り返し前のディスク上でのトレース・ファイルの大きさを指定します。デフォルト値 0 を指定すると、トレースの折り返しは、使用不可になります。

構成変換ツール

CICS Transaction Gateway、IBM CICS ユニバーサル・クライアント、および CICS Gateway for Java の旧バージョンの構成ファイルを CICS Transaction Gateway バージョン 4.0 構成ファイルの新しい形式へ変換するには、構成変換ツール (ctgconv) を使用します。

3.1 よりも前の CICS Transaction Gateway、IBM CICS ユニバーサル・クライアント、および CICS Gateway for Java のバージョンで作成された構成ファイルに対して変換ツールを使用します。バージョン 3.1 からアップグレードする場合は、これを使用する必要はありません。

変換ツールは、以下の変換を行います。

Gateway.properties	CICS Transaction Gateway バージョン 3.0 および CICS Gateway for Java のプロパティー・ファイル。
CICSCLI.INI	CICS クライアント、バージョン 2 およびバージョン 3.0 のクライアント初期設定ファイル。

デフォルトで変換ツールは、CTG.INI という 1 つの出力ファイルを生成します。このファイルのサンプルが 55ページの『構成ファイルの編集』 に示されています。

古いファイルは .BAK という拡張子により名前変更されますが、それが古いものであることを示すバナーが挿入されます。

変換ツールの使用

ctgconv のパラメーターは以下のとおりです。

```
ctgconv -c=file [-g=file] [-o=file]
```

各パラメーターにおいて、*file* は次のいずれかです。

- ファイル名に拡張子を付けたもの。この場合、 /opt/ctg/bin ディレクトリーが想定されます。拡張子は .INI でなければなりません。ただし、CICS Transaction Gateway 入力ファイルでは例外で、その場合は .properties 拡張子になります。
- 絶対パス名。
- ディレクトリー名 (末尾に / は付けない)。この場合はデフォルト・ファイル名が想定されます。

-c=*file*
変換対象のクライアント初期設定ファイルを指定します。

file にディレクトリーを指定した場合、ファイル名として CICSCLI.INI が想定されます。

パラメーターを指定しない場合、CICSCLI 環境変数によってクライアント初期設定ファイルの位置を突き止めます。 CICSCLI が設定されていなければ、ファイルとして /opt/ctg/bin/CICSCLI.INI が想定されます。

-g=*file*
変換対象の Gateway.properties ファイルを指定します。

-g パラメーターを指定しないと、CTG.INI で Gateway セクションは作成されません。

-o=*file*
変換後のファイルのパス名を指定します。デフォルトは /opt/ctg/bin/CTG.INI.

ctgconv の使用方法についてのヘルプを表示するには、`ctgconv -?` と入力します。

変換の際、入力ファイルは、ファイル拡張子の最後の 3 文字が .BAK になるよう、名前変更されます。その名前のファイルがすでにある場合は上書きされ、変換で名前変更が行われる前に出力ファイルが存在していた場合も同様に上書きされます。

冗長なパラメーターは古い構成ファイルから除かれ、他のパラメーターには変換ファイルで新しい名前が与えられます。

構成ファイルの編集

構成ツールを使用することが推奨されているものの、構成ファイルを編集して構成を実行することもできます。

構成ファイルは CICS ユニバーサル・クライアントと CICS Transaction Gateway の両方に使用され、以下のセクションがあります。

1. GATEWAY
2. CLIENT
3. SERVER
4. DRIVER

これらのセクションの形式は次のとおりです。

```
SECTION sectioname [=value]
     property1=value
     property2=value
     ...
     propertyN=value
ENDSECTION
```

構成ファイルに加えた変更を有効にするには、CICS ユニバーサル・クライアントと CICS Transaction Gateway の両方を再始動する必要があります。

以下の節では、各セクションの中のプロパティーの例を示します。プロパティーについての詳細は、構成ツールの対応する設定の説明を参照してください。

注: コメントを示すために # 文字を使用する場合は、それを行の先頭に置くか、スペースまたはタブ文字の後に置きます。この理由は、有効な名前の中には (たとえばモードネーム) # 文字で開始できるものがあるためです。

GATEWAY セクション

```
SECTION GATEWAY
# ------------------------------------------------------------------------------
# Properties file for the CICS Transaction Gateway
# ------------------------------------------------------------------------------

        initconnect=1   # Initial number of ConnectionManager threads
        maxconnect=100  # Maximum number of ConnectionManager threads
        initworker=1    # Initial number of Worker threads
        maxworker=100   # Maximum number of Worker threads
        trace=on        # Enable extra tracing messages
        notime=on       # Disable timing information in messages
        noinput=on      # Disable the reading of input from the console
        nonames=on      # Do not display TCP/IP hostnames
# ------------------------------------------------------------------------------
# Entries to define the protocols which this CICS Transaction Gateway daemon will
# accept.
#
# Entries for a particular protocol begin with protocol@<protocol-name>.
# followed by which property is being set. The following properties MUST be set
# for each protocol handler to be used :
#
#  (1) protocol@<protocol-name>.handler = ....
#         The name of the class which provides the handler for this protocol
#
#  (2) protocol@<protocol-name>.parameters = ....
```

```
#         The parameters passed to the protocol handler. The possible parameters
#         vary upon the protocol handler.
#
        protocol@tcp.handler=com.ibm.ctg.server.TCPHandler
        Protocol@tcp.parameters=port=2006;sotimeout=1000;
        connecttimeout=2000;idletimeout=600000;pingfrequency=60000

        protocol@http.handler=com.ibm.ctg.server.HttpHandler
        Protocol@http.parameters=port=8080;sotimeout=1000;connecttimeout=2000;
        idletimeout=120000;requiresecurity
        protocol@ssl.handler=com.ibm.ctg.server.SslHandler
        Protocol@ssl.parameters=port=8050;sotimeout=1000;connecttimeout=2000;
        Idletimeout=600000;pingfrequency=60000;keyring=ServerKeyRing;keyringpw=default;
        clientauth=on;

        Protocol@https.handler=com.ibm.ctg.server.HttpsHandler
        Protocol@https.parameters=port=443;sotimeout=1000;
        connecttimeout=2000;Idletimeout=120000;keyring=ServerKeyRing;keyringpw=default;
        clientauth=off;

# ------------------------------------------------------------------------------
# Advanced settings.

        ecigenericreplies=off
        workertimeout=10000
        closetimeout=10000

ENDSECTION
```

CLIENT セクション

```
SECTION CLIENT = *
#-------------------------------------------------------------------------
# Client section - This section defines the local CICS client.  There
#                  should only be one Client section.

        MaxServers = 1                  # Only allow one server connection
        MaxRequests = 256               # Limit the maximum server interaction
        MaxBufferSize = 32              # Allow for a 32K maximum COMMAREA
#       CCSID=850

ENDSECTION
```

SERVER セクション

```
SECTION SERVER = cicstcp1
#-------------------------------------------------------------------------
# Server section - This section defines a server to which the client may
#                  connect.  There may be several Server sections.

        Description = A TCP/IP Server   # Arbitrary description for the server
        Protocol = TCPIP                # Matches with a Driver section below
        NetName = cicstcp.ibm.com       # The server's TCP/IP name or address
        Port = 0                        # Use the default TCP/IP CICS port

ENDSECTION
```

DRIVER セクション

```
SECTION DRIVER = TCPIP          ; Matches the Server's Protocol value
#-------------------------------------------------------------------------
#       Driver section - This section defines a communications protocol DLL
#                used to communicate with a server.  There may be
```

```
#                several Driver sections.

       DriverName = CCLIBMIP          #Use the IBM TCP/IP communications library

ENDSECTION
```

注: DRIVER セクションは、構成ツールのどの設定にも対応しません。構成ツールは正しいプロトコル・ドライバーを自動的に選択します。

構成ファイルおよびマッピング・ファイルの名前変更

以下のファイルは、CICS ユニバーサル・クライアントに付属しており、デフォルトでは /opt/ctg/bin ディレクトリーにあります。

CTGSAMP.INI サンプルのクライアント構成ファイルです。(デフォルトの構成ファイル名は CTG.INI です。)

CICSKEY.INI キーボード・マッピング・ファイルです。(サンプル・ファイル CICSKEYSAMP.INI も提供されています。)

CICSCOL.INI カラー・マッピング・ファイルです。(サンプル・ファイル CICSCOLSAMP.INI も提供されています。)

サービス更新をインストールすると、これらのファイルが上書きされ、カスタマイズの内容が失われることがあるので、これらのファイルを独自にカスタマイズした場合は、それらの別名バージョンを作成することをお勧めします。

カスタマイズしたファイルは次の環境変数で参照してください。

ファイル	環境変数
クライアント構成ファイル	CICSCLI
キーボード・マッピング・ファイル	CICSKEY
カラー・マッピング・ファイル	CICSCOL

クライアント・キーボード・マップの構成

端末エミュレーター操作のためのキーボード・マッピングは、キーボード・マッピング・ファイルで定義されます。付属のこのファイルは、CICSKEYSAMP.INI であり、デフォルトでは /opt/ctg/bin ディレクトリーにあります。ただし、独自にカスタマイズしたマッピング・ファイルを作成することをお勧めします。

キーボード・マッピング・ファイルは、以下の方法で指定することができます。

- cicsterm コマンドの -k オプション。これは、キーボード・マッピング・ファイルと特定の端末を結び付けます (『クライアント・キーボード・マップの構成』のページを参照)。
- CICSKEY 環境変数。以下に、例を示します。

```
export CICSKEY=/var/cicscli/MYKEYS.INI
```

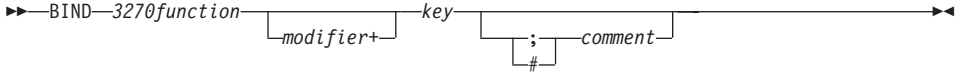
これらの環境変数がどちらも指定されていない場合は、/opt/ctg/bin ディレクトリーの CICSKEY.INI のファイル名が想定されます。

キーボード・マッピング・ファイルはいつでも変更することができますが、変更は、次回に端末エミュレーターが始動して初めて有効になります。

キーボード・マッピング・ファイルの構文

この節では、キーボード・マッピング・ファイルの構文について説明します。(英字および数値キーを除き) デフォルトの割り当てがされていないため、必要なキーごとに 1 つのステートメントが必要になります。大文字小文字は区別されないため、キーワードおよび値は、大文字、小文字、あるいは大文字小文字混合で入力してもかまいません。各バインディングは、それぞれ別の行に、次の形式で指定する必要があります。

キーボード・マッピング・ファイルの構文



たとえば、3270 関数 EraseEof を Ctrl+Delete キー (2 つのキーを同時に押す) にマップするには、バインディングは次のようになります。

```
bind     EraseEof              Ctrl+Delete ;erase to end of field
```

キーボード・マッピング・ファイル

キーボード・マッピング・ファイルでは、*3270function* は、以下のいずれか 1 つになります。

```
backspace       pa1             pf1     pf13
backtab         pa2             pf2     pf14
clear           pa3             pf3     pf15
cursordown                      pf4     pf16
cursorleft      printscreen     pf5     pf17
cursorright     reset           pf6     pf18
cursorselect    tab             pf7     pf19
cursorup                        pf8     pf20
delete          ignore          pf9     pf21
enter                           pf10    pf22
eraseeof                        pf11    pf23
eraseinput                      pf12    pf24
home
insert
newline
```

キーボード上の不要な制御キーを無視するために、ignore という値が与えられています。 (予期しない文字は生成されません。)

modifier は、以下のいずれか 1 つになります。

```
Ctrl
Shift
```

key は、表3 に示したキーのいずれか 1 つになりますが、 modifier+key の組み合わせの中にはサポートされていないものがあります。

表 3. マップ可能な CICS ユニバーサル・クライアントのキー

グループ	キー
Esc キー	Escape
ファンクション・キー	f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12
数字キー	0 1 2 3 4 5 6 7 8 9
英字キー	a b c d e f g h i j k l m n o p q r s t u v w x y z
タブ・キー	Tab
移動キー	newline backspace insert home pageup delete end pagedown up left down right

表 3. マップ可能な CICS ユニバーサル・クライアントのキー (続き)

グループ	キー
キーパッド・キー	keypad/ keypad* keypad- keypad7 keypad8 keypad9 keypad4 keypad5 keypad6 keypad+ keypad1 keypad2 keypad3 keypad0 keypad. keypadenter

注: CICS ユニバーサル・クライアント (HP-UX 版) では、 Ctrl/Act、Print Screen、Scroll Lock および Pause キーは使用できません。 3270 関数の Clear は、デフォルトでは Esc キーに割り当てられています。

キーの組み合わせ

次の修飾キー (modifier) とキー (key) の組み合わせをマップすることができます。

修飾キーなし すべてのキーをマップに使用することができます。

Ctrl 修飾キー ファンクション・キー、移動キー、英字キー、タブ・キー、およびキーパッド・キーだけをマップすることができます。

Shift 修飾キー ファンクション・キー、数字キー、タブ・キー、および英字キーだけをマップすることができます。

サンプルのキー・マッピング・ファイルを図5 に示します。

```
;*************************************************************************
;* IBM CICS Client - 3270 Emulator Keyboard Binding File                 *
;*************************************************************************
; Format:
;   bind 3270_key [modifier+]workstation_key
;
; Refer to the "IBM CICS Clients Administration" book for full details
; of available values.

; Note: There are no default key bindings, ensure all required 3270
;       keys are mapped.  "Enter" and "Clear" are particularly
;       important.
```

図 5. CICS ユニバーサル・クライアントのサンプルのキーボード・マッピング・ファイル (1/3)

```
; Miscellaneous keys
bind Enter          Newline
bind Enter          KeypadEnter
bind Enter          Ctrl+KeypadEnter
bind Clear          Escape
bind Reset          Ctrl+R
bind Newline        Ctrl+N
bind Tab            Tab
bind Backtab        Shift+Tab
bind Backspace      Backspace
bind Delete         Delete
bind Delete         Keypad.
bind Insert         Insert
bind Insert         Keypad0
bind EraseEof       Ctrl+E
bind EraseInput     Ctrl+A
bind Ignore         End
bind PrintScreen    Ctrl+P
; Cursor movement
bind CursorUp       Up
bind CursorUp       Keypad8
bind CursorDown     Down
bind CursorDown     Keypad2
bind CursorLeft     Left
bind CursorLeft     Keypad4
bind CursorRight    Right
bind CursorRight    Keypad6
bind CursorSelect   Ctrl+F3
bind Home           Home
bind Home           Keypad7
```

図 5. CICS ユニバーサル・クライアントのサンプルのキーボード・マッピング・ファイル (2/3)

```
; PF and PA keys
bind Pf1            F1
bind Pf2            F2
bind Pf3            F3
bind Pf4            F4
bind Pf5            F5
bind Pf6            F6
bind Pf7            F7
bind Pf7            Keypad9
bind Pf7            PageUp
bind Pf8            F8
bind Pf8            Keypad3
bind Pf8            PageDown
bind Pf9            F9
bind Pf10           F10
bind Pf11           F11
bind Pf12           F12
bind Pf13           Shift+F1
bind Pf14           Shift+F2
bind Pf15           Shift+F3
bind Pf16           Shift+F4
bind Pf17           Shift+F5
bind Pf18           Shift+F6
bind Pf19           Shift+F7
bind Pf20           Shift+F8
bind Pf21           Shift+F9
bind Pf22           Shift+F10
bind Pf23           Shift+F11
bind Pf24           Shift+F12
bind Pa1            Ctrl+Insert
bind Pa2            Ctrl+Home
bind Pa3            Ctrl+PageUp
```

図 5. CICS ユニバーサル・クライアントのサンプルのキーボード・マッピング・ファイル (3/3)

画面のカラーのカスタマイズ

画面のカラーおよび属性は、カラー・マッピング・ファイルで定義されます。独自に設定できるようサンプルが付属しています。付属ファイルは CICSCOL.INI であり、デフォルトでは /opt/ctg/bin ディレクトリーにあります。独自にカスタマイズしたマッピング・ファイルを作成することをお勧めします。

カラー・マッピング・ファイルは、以下のもので識別することができます。

- cicsterm コマンドの -c オプション。これは、カラー・マッピング・ファイルと特定のサーバーを結び付けます (『画面のカラーのカスタマイズ』のページを参照)。
- CICSCOL 環境変数: たとえば

```
export CICSCOL=/var/cicscli/MYCOLS.INI
```

これらの環境変数がどちらも指定されていない場合は、現行ディレクトリーの CICSCOL.INI のファイル名が想定されます。

カラー・マッピング・ファイルは、明滅や下線など、3270 の画面属性を正確に複製できないハードウェア環境において、代替表現を提供する場合に使用します。したがって、カラー・マッピング・ファイルでは、3270 画面属性をクライアント・ハードウェアでどのようにエミュレートするかを定義します。

カラー・マッピング・ファイルは任意指定です。ただし、ほとんどのハードウェア環境では、エミュレーターが明滅や下線のサポートを要求する場合に、マッピング・ファイルが必要になります。

注:

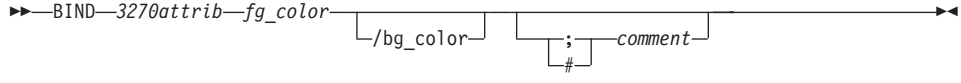
1. カラー・マッピング・ファイルが属性にマッピングを指定している場合は、クライアントが稼働しているハードウェアが実際に画面属性をサポートしている場合でも、このマッピングが使用されます。
2. アプリケーションが、3270 のフィールドを、たとえば下線を付けて表示するように要求しており、エミュレーションの設定値が指定されていず、ハードウェアが下線を表示できない場合は、このフィールドは全く何も強調されずに表示されます。

カラー・マッピング・ファイルはいつでも変更することができますが、変更は、次回に端末エミュレーターが始動して初めて有効になります。

カラー・マッピングの構文

カラー・マッピング・ファイルの構文は、以下のようになります。大文字小文字は区別されないため、値は大文字、小文字、あるいは大文字小文字混合で入力してもかまいません。各バインディングは、それぞれ別の行に、次の形式で指定する必要があります。

カラー・マッピング・ファイルの構文



カラー・マッピング・ファイル

カラー・マッピング・ファイルでは、*3270attrib* は、以下のいずれか 1 つになります。

```
normal_protected    intensified_protected
normal_unprotected  intensified_unprotected

default    blinking_default    underscored_default
blue       blinking_blue       underscored_blue
green      blinking_green      underscored_green
cyan       blinking_cyan       underscored_cyan
red        blinking_red        underscored_red
magenta    blinking_magenta    underscored_magenta
white      blinking_white      underscored_white
yellow     blinking_yellow     underscored_yellow

default_highlight

operator_information_area
```

各 *fg_color* および *bg_color* (前景色および背景色) は、以下のいずれか 1 つになります。

```
black       light_gray
blue        light_blue
brown       yellow
cyan        light_cyan
green       light_green
magenta     light_magenta
red         light_red
gray        white
```

bg_color を省略した場合は、デフォルト値の黒になります。

サンプルのカラー・マッピング・ファイルを 66ページの図6 に示します。

```
;************************************************************************
;* IBM CICS Client - 3270 Emulator Color Binding File                   *
;************************************************************************

; Format:
;   bind 3270_field foreground_color/background_color
;
; Refer to the "IBM CICS Clients Administration" book for full details
; of available values.

; Operation information area at the bottom of the screen.
bind operator_information_area  black/green

; Color used as default for a terminal defined as monochrome, or to display
; characters displayed in the default color when character or field
; attributes are used
bind default                    light_green/black

; Color used for an intense field when either the terminal is defined as
; monochrome, or the screen has been formatted with extended attributes but
; the data to be displayed has default color and highlight.
bind default_highlight          white/black

; Colors used for datastream formatted with field attributes when no
; extended character or field attributes are in use
bind normal_unprotected         light_green/black
bind intensified_unprotected    light_red/black
bind normal_protected           light_cyan/black
bind intensified_protected      white/black

; Colors used for datastream formatted with extended field attributes
; or datastream formatted with character attributes.
bind red                        light_red/black
bind green                      light_green/black
bind blue                       light_blue/black
bind magenta                    light_magenta/black
bind cyan                       light_cyan/black
bind yellow                     yellow/black
bind white                      white/black

; The following bindings provide a mapping for 3270 blink and under-
; score attributes, as most workstation displays do not provide blink
; or underscore capabilities.  For those that do these bindings could
; be deleted.
bind blinking_default           light_green/gray
bind blinking_red               light_red/gray
bind blinking_green             light_green/gray
bind blinking_blue              light_blue/gray
bind blinking_magenta           light_magenta/gray
bind blinking_cyan              light_cyan/gray
bind blinking_yellow            yellow/gray
bind blinking_white             white/gray
```

図6. サンプルのカラー・マッピング・ファイル (1/2)

```
bind underscored_default       light_green/light_gray
bind underscored_red           light_red/light_gray
bind underscored_green         light_green/light_gray
bind underscored_blue          light_blue/light_gray
bind underscored_magenta       light_magenta/light_gray
bind underscored_cyan          light_cyan/light_gray
bind underscored_yellow        yellow/light_gray
bind underscored_white         white/light_gray

;* End of file                                                    *
```

図 6. サンプルのカラー・マッピング・ファイル (2/2)

ローカルの CICS Transaction Gateway サポートを使用するための準備

ローカルの CICS Transaction Gateway サポートを使用するアプリケーションを実行するには、環境を事前に正しく構成しなければなりません。要件を以下に示します。

1. **正しく構成された Java 環境**

   ローカルの CICS Transaction Gateway サポートを使用する Java アプリケーションは、標準の Java プログラムと見なすことができるので、標準の JDK Java 実行可能プログラムで実行されます。したがって、環境は、Java バイナリー、ライブラリー、およびクラスを検出できるように正しく設定しなければなりません。正しい環境設定については、JDK の資料を参照してください。

2. **設定を修正して CICS Transaction Gateway バイナリーを検出できるようにする**

   CICS Transaction Gateway の .class ファイルを利用できるようにする必要があります。CLASSPATH 環境変数の設定に、CICS Transaction Gateway の ctgserver.jar および ctgclient.jar アーカイブを含める必要があります。また、関係のあるバイナリーも利用できなければなりません。

   SHLIB_PATH 環境変数の設定に、CICS Transaction Gateway の bin サブディレクトリーを含めて、アプリケーションが libCTGJNI.so を見つけられるようにしてください。

アプリケーションの実行: JDK `java` コマンドを適切なオプションを指定して使用します。

第5章 セキュリティー

この章では、ネットワーク・セキュリティー・プロトコルとして SSL および HTTPS を使用するよう CICS Transaction Gateway をセットアップする方法について説明します。

- 『概説』では、ネットワーク・セキュリティーの概念と用語を概説します。ここでは、暗号鍵、ディジタル証明書、および KeyRing の概念を紹介します。
- 74ページの『SSL と認証』では、SSL プロトコルと、それによって提供される認証のタイプについて説明します。
- 76ページの『ctgikey ツール』では、CICS Transaction Gateway が提供する、ディジタル証明書の管理のためのツールである iKeyMan を紹介します。
- 77ページの『外部的に署名された証明書の使用 (SSLight)』では、外部で署名されたディジタル証明書の入手方法と、それを CICS Transaction Gateway で使用するために KeyRing ファイルに保管する方法を説明します。
- 83ページの『自己署名された証明書の使用』では、CICS Transaction Gateway で使用するための自己署名のディジタル証明書の生成方法を説明します。
- 88ページの『SSL および HTTPS 用に CICS Transaction Gateway を構成する』では、保護プロトコルを使用するよう CICS Transaction Gateway を構成する方法を説明します。

CICS Transaction Gateway が提供する SSL のインプリメンテーションは pure Java で書かれており、 **SSLight** と呼ばれています。もう 1 つのインプリメンテーションである **System SSL** は、 OS/390 の CICS Transaction Gateway だけに適用され、そのプラットフォームの SSL サーバー専用です。

概説

CICS Transaction Gateway は、インターネット操作の成功に不可欠である、セキュリティー通信を包括的にサポートしています。セキュア・ネットワーク・プロトコルを使用すると、クライアントのアプレットおよびアプリケーションが SSL を使用して CICS Transaction Gateway と安全に通信できるようになります。 SSL および HTTPS プロトコルについては、1ページの『第1章 概説』で紹介されています。この章では、これらのプロトコルを使用するように CICS Transaction Gateway をセットアップする方法を詳細に説明します。

安全な通信の特性を以下に示します。

- **機密性 (Confidentiality)**

  機密性とは、メッセージをインターネットまたはイントラネットを介して送信するとき、その内容がプライベートに保たれることを意味します。機密性は、メッセージの暗号化によって保証されます。

- **保全性 (Integrity)**

  保全性とは、メッセージが伝送中に変更されないことを意味します。経路の途中にあるルーターで、テキストが挿入または削除されたり、メッセージが通過する際に誤り伝えられる可能性があります。保全性がないと、こちらから送信するメッセージと相手方で受信するメッセージとの一致は保証されません。保全性は暗号化およびディジタル・シグニチャーによって保証されます。

- **アカウンタビリティー (Accountability)**

  アカウンタビリティーとは、交換が行われたことを送信側と受信側の両方が認めることを意味します。アカウンタビリティーがないと、受信側はメッセージが到達しなかったと主張することもできます。アカウンタビリティーはディジタル・シグニチャーによって保証されるので、問題が生じた場合、だれに責任があるかを特定できます。

- **認証性 (Authenticity)**

  認証性とは、通信の相手方を識別でき、その相手方を信用できることを意味します。認証性を得るには、相手側が主張どおりの者であることを確認できるように、身元の検査が必要となります。認証は、ディジタル・シグニチャーおよびディジタル証明書を使用することによって達成されます。

暗号とは

暗号化によって、インターネットを介して送信する伝送の機密性が保証されます。最も簡単な形式の暗号化は、メッセージをスクランブル (順序を変える) して、受信側がスクランブル解除するまでそのメッセージを読めないようにすることです。送信側は、算法パターンつまり*鍵* を使用してメッセージを暗号化し、受信側は復号キーを使用してメッセージをスクランブル解除します。

暗号化 (およびディジタル・シグニチャーと認証) には、2 種類の鍵を使用できます。

1. 対称 (Symmetric)
2. 非対称 (Asymmetric)

対称鍵 を使用すると、送信側と受信側は一定のパターンを共有して、送信側はそれによってメッセージを暗号化し、受信側はそれによってメッセージを復号

します。対称鍵を使用する際のリスクは、秘密の鍵を通信相手に送るときに安全な伝送手段を選択しなければならないということです。

非対称鍵 では、鍵のペアを作成します。この鍵ペアは、*公開鍵* と *秘密鍵* で構成されます。対称鍵とは異なり、これらの鍵は互いに異なっていて、秘密鍵は公開鍵よりも多くの秘密の暗号化パターンを保持しています。

送信側は、安全に通信したい相手に公開鍵を配布します。秘密鍵は保存して、パスワードで保護します。送信側だけが秘密鍵を持っているので、公開鍵で暗号化された受信メッセージを復号できるのは送信側だけとなります。

SSL などのプロトコルは、非対称 (公開鍵としても知られる) 暗号と対称鍵暗号の両方を使用します。公開鍵暗号は、TCP/IP ハンドシェークに使用されます。ハンドシェークの際に、マスター・キーがクライアントからサーバーに渡されます。クライアントおよびサーバーは、そのマスター・キーを使用してそれぞれ独自のセッション・キーを作成します。その後、セッション・キーを使用して、セッションの残りの期間でデータの暗号化および復号が行われます。

ディジタル・シグニチャーおよびディジタル証明書

ディジタル・シグニチャー は、数学的に計算された固有の署名であり、アカウンタビリティーを保証します。

ディジタル証明書 によって、エンティティーを一意的に識別することが可能になります。それは本質的には、信用されている第三者が発行する電子的 ID カードです。

ディジタル証明書は、所有者のアイデンティティーを確立することと、所有者の公開鍵を使用可能にすることの 2 つの目的を果たします。ディジタル証明書は、信用される権威である、 VeriSign Inc、Thawte などの認証局 (CA) が発行します。発行の際には一定の有効期間が定められて、その期限日付が過ぎると置き換えが必要になります。

ディジタル証明書は、以下の要素から構成されます。

- 認証される人の公開鍵。
- *識別名 (DN)* とも呼ばれる、認証される人の名前とアドレス。
- CA のディジタル・シグニチャー。
- 発行日付。
- 有効期限。

識別名は、個人または組織の名前とアドレスです。識別名は証明書要求の一部として入力します。ディジタル式に署名された証明書には、ユーザーの識別名だけでなく、 CA の検証を可能にする CA の識別名も含まれます。

安全に通信するために、伝送の受信側は、送信側が使用している証明書を発行した CA を信用しなければなりません。その結果、送信側がメッセージに署名するたびに、受信側には対応する CA の*署名者の証明書* およびトラステッド・*ルート鍵* に指定された公開鍵が必要となります。 たとえば、ご使用の Web ブラウザーにはトラステッド CA の署名者の証明書を示すデフォルトのリストが備わっています。他の CA からの証明書を信用する場合、その CA からの証明書を受け取って、それをトラステッド・ルート鍵に指定しなければなりません。

あなたの公開鍵を含むディジタル証明書を他者に送信する場合、その者がディジタル証明書を誤用してあなたであるかのように振る舞うことのないように保護するものは何でしょうか。その答えは、あなたの秘密鍵です。ディジタル証明書だけでは、誰かのアイデンティティーを証明することはできません。ディジタル証明書は、所有者のディジタル・シグニチャーを検査するために必要な公開鍵を提供して、所有者のアイデンティティーの検証を可能にするだけです。そのため、ディジタル証明書の所有者は、ディジタル証明書内の公開鍵に属する秘密鍵を保護しなければなりません。そうしないと、秘密鍵が盗まれた場合、誰か他の者がそのディジタル証明書の正当な所有者であるかのように振る舞う可能性があります。

ディジタル証明書の取得

証明書は、次の 2 つの方法で取得できます。

1. CA から証明書を購入する
2. 自分宛てに証明書を発行する、つまり自分自身の CA となる。

CA から証明書を購入する

インターネット上で商業ビジネスを企画している場合、VeriSign Inc (ホーム・ページは、 https://www.verisign.com/) などの CA からサーバー証明書を購入してください。

証明書要求を VeriSign に送ると、証明書が発行される前に要求者の身元を証明することが求められます。この承認プロセスは要求者、要求者の組織、および VeriSign を保護するために必要であるとはいえ、予想以上の期間がかかることがあります。 VeriSign は証明書要求にディジタル式に署名して、一意的な証明書を E メールで送り返してきます。

外部的に署名された証明書の取得については、77ページの『外部的に署名された証明書の使用 (SSLight)』で説明します。

注: VeriSign サーバー証明書を、異なるマシン上の複数のサーバーが共有することはできません。

自分自身で証明書を発行する

CA として行動する場合、自分自身の、または他者の証明書要求に署名できます。外部的なインターネット商取引のためではなく、組織内での証明書だけが必要な場合、これは良い選択です。そのようなシナリオでは、イントラネット内の注意深く選択した主要な人員にアクセスを限定するとよいでしょう。

主要な人員は、自己署名された CA 証明書を受信してそれをトラステッド・ルートに指定できる Netscape Navigator などのブラウザーを持っているとします。それらの要員は、あなたとの通信を信用して、安全に情報を共有できるようになります。

自己署名された証明書の生成については、83ページの『自己署名された証明書の使用』で説明します。

KeyRing

では、ディジタル証明書とそれに関連する鍵はどこに保管しますか。答えは、公開鍵、秘密鍵、証明書、およびトラステッド・ルート鍵を、 *KeyRing* ファイルに保管するということです。

CICS Transaction Gateway は Java クラスを使用して、証明書と鍵データを SSL サーバーと SSL クライアントの両方に保管します。 CICS Transaction Gateway デーモン (SSL サーバーとして行動する) は、サーバー KeyRing (**ServerKeyRing.class** など) を使用し、すべての SSL クライアントはクライアント KeyRing (**ClientKeyRing.class** など) を使用します。SSL および HTTPS プロトコルでは、安全な接続を確立するため、これらの Java クラスへのアクセスが必要です。これらのアクセスは、CICS Transaction Gateway を構成するときに確立します。 88ページの『SSL および HTTPS 用に CICS Transaction Gateway を構成する』を参照してください。

この章の続く節では、以下の方法について説明します。

- KeyRing ファイルの作成。
- ディジタル証明書の取得。
- ディジタル証明書を受信して KeyRing ファイルに入れる。

SSL と認証

SSL では、クライアントがサーバーのアイデンティティーを認証することができます。これは、*サーバー認証* と呼ばれます。

SSL バージョン 3 ではさらに、サーバーがクライアントを認証することができます。これは*クライアント認証* と呼ばれます。これは、応答する前にクライアントが誰であるかを確認する必要がある場合に使用されます。 SSL クライアント認証が設定されている場合、サーバーはクライアントが SSL 接続を行うたびにクライアントの証明書を要求します。サーバーは文書をサービス提供する前に、クライアントの証明書に含まれる DN 情報を使用して、クライアントの要求に含まれる DN 情報を検査します。

SSL はセキュリティー・ハンドシェークを使用して、クライアントとサーバーとの間の TCP/IP 接続を開始します。ハンドシェークの際、クライアントとサーバーは、セッションで使用するセキュリティーの鍵および暗号化に使用するアルゴリズムについて合意します。クライアントは、サーバーを認証します。さらに、クライアントが SSL クライアント認証によって保護される文書を要求する場合、サーバーはクライアントの証明書を要求します。ハンドシェークの後、クライアント要求とサーバー応答の両方に含まれるすべての情報が、SSL によって暗号化および復号されます。それには以下の情報が含まれます。

- クライアントが要求している URL
- 発信中のフォームの内容
- ユーザー名やパスワードなどの、アクセス許可情報
- クライアントとサーバーの間で送信されたすべてのデータ

SSL ハンドシェークを説明した図を、75ページの図7 に示します。

図 7. サーバー認証との SSL ハンドシェーク

HTTPS

HTTPS は、SSL と HTTP とを結合する固有のプロトコルです。 SSL で保護された文書にリンクする HTML 文書のアンカーとして、 https:// を指定しなければなりません。クライアント・ユーザーは、https:// を指定して URL をオープンすることによっても、 SSL で保護された文書を要求できます。

HTTPS (HTTP + SSL) と HTTP とは別個のプロトコルであって、通常は異なるポートを使用する (それぞれ、443 と 8080) ため、 SSL 要求と非 SSL 要求とを同時に実行できます。その結果、情報をセキュリティーを使用しないですべてのユーザーに提供するとともに、特定の情報をセキュリティー要求を行うブラウザーだけに提供することができます。この方法によってインターネッ

トの小売業者は、ユーザーが商品を閲覧するときにはセキュリティーを使用せず、注文書に記入してクレジット・カード番号を送信するときにはセキュリティーを使用するようにしています。

SSL で HTTP をサポートしないブラウザーは、当然 HTTPS を使用する URL を要求することはできません。非 SSL ブラウザーでは、安全に発信しなければならないフォームを発信することはできません。

ctgikey ツール

CICS Transaction Gateway には、ディジタル証明書を保守するためのツール **iKeyman** が備わっています。 **ctgikey** コマンドを使用して、適切な環境 (JAVA_HOME 環境変数を含む) の設定および iKeyman の起動を行えます。

iKeyMan を使用して、以下のことができます。

- CA にディジタル証明書を要求して受け取る。77ページの『外部的に署名された証明書の使用 (SSLight)』を参照。
- 自己署名された証明書を生成する。83ページの『自己署名された証明書の使用』を参照。
- KeyRing ファイルに証明書を追加する。
- KeyRing パスワードを変更する。
- デフォルトの秘密鍵を設定する。
- 鍵を削除する。
- 鍵をファイルにコピーしてエクスポートする。
- エクスポートされたコピーから鍵をインポートして、それを KeyRing に追加する。

iKeyman をクライアント・ワークステーションに配布する

KeyRing 管理の大部分は CICS Transaction Gateway マシン上で実行されますが、 iKeyman ツールを SSL サーバーに接続しているクライアントに配布しなければならないこともあります。クライアント・マシンには、CICS Transaction Gateway Java サポートの最低レベル、つまり IBM Java SDK バージョン 1.2.2 が必要です。さらに、iKeyman Java 始動クラスを起動するためのスクリプトまたはコマンド・ファイルも必要です。

例として、JDK を使用して iKeyman を起動するための Windows NT コマンド行を以下に示します。

```
java.exe -classpath
"e:¥ikeyman¥cfwk.zip;e:¥ikeyman¥gsk4cls.jar;e:¥ikeyman
¥swingall.jar;" -Dkeyman.javaOnly=true com.ibm.gsk.ikeyman.Ikeyman
```

JRE を使用して起動する場合は、以下のとおりです。

```
jre.exe -classpath
"e:¥ikeyman¥cfwk.zip;e:¥ikeyman¥gsk4cls.jar;e:¥ikeyman
¥swingall.jar;" -Dkeyman.javaOnly=true com.ibm.gsk.ikeyman.Ikeyman
```

これらの例では、クライアント・ワークステーションの ikeyman ディレクトリーに以下のファイルがあると想定しています。

- cfwk.zip
- cfwk.sec
- gsk4cls.jar
- swingall.jar
- ikminit.properties

これらのファイルはすべて、CICS Transaction Gateway がインストールされている classes サブディレクトリーにあります。

注: cfwk.zip が CLASSPATH 設定に出現する最初の IBM 提供のアーカイブであることを確認してください。

外部的に署名された証明書の使用 (SSLight)

CICS Transaction Gateway は、SSL クライアントを認証して、VeriSign Inc. などの認証局から外部的に署名された証明書を受け入れることのできる SSL サーバーとして機能できます。

この節では、証明書管理インターフェース iKeyMan を使用して、 SSL サーバーと SSL クライアントの両方を構成する方法について説明します。これは純粋な Java で作成されているため、適切な JVM がインストールされた複数のクライアント / サーバー・ワークステーションに配布することができます。

SSL サーバーおよびクライアントを構成することには、KeyRing クラスの作成、ディジタル証明書の取得、およびそれらを KeyRing クラスに受け入れることが含まれます。

サーバー KeyRing クラスには、サーバー証明書および対応する秘密鍵、そしていくつかの署名者の証明書が含まれます。サーバー証明書とは、 SSL サーバーを接続しているクライアントに識別させるために使用するディジタル証明書です。

クライアント KeyRing クラスには、少なくとも、 SSL サーバーの署名者の証明書、およびクライアント認証が必要な場合にはクライアント x.509 証明書が含まれます。

SSL サーバーを構成する

この節では、VeriSign Web サイト (www.verisign.com) から試用 (テスト) サーバー証明書を取得する方法について説明します。 VeriSign は、試用サーバー証明書を 14 日間使用することを許可しています。これはデモンストレーション用のサーバー証明書なので、信用された VeriSign 認証局ではなく VeriSign Test CA によって署名されます。

すべての SSL クライアントは、 VeriSign Test 署名者の証明書ルート鍵がクライアント KeyRing に、または HTTPS 接続の場合はブラウザーのリポジトリーにインストールされている必要があります。

本書の例では、Netscape Communicator バージョン 4.5 の使用を想定しています。

完全な VeriSign サーバー証明書を取得する場合も、試用版に関してここに説明するのと同じ手順で行います。

サーバー KeyRing を作成する

最初のステップは、サーバー KeyRing クラスを作成することです。これには後に、署名者の証明書とサーバー証明書 (および関連した秘密鍵) が含まれることになります。このリポジトリーはパスワードによって保護されていて、iKeyMan によって .class を作成するとき、パスワードの「強度」が示されます。英数字の順序列を使用することをお勧めします。これにより、パスワードは「乱暴な強制辞書」攻撃に対して耐久力を増します。

証明書を取得するには、以下のようにします。

1. ctgikey を開始します。
2. 「**Key Database File --> New**」を選択します。
3. 「**Key Database Type**」から、「**SSLight key database class**」を選択します。
4. `ServerKeyRing.class` を「**ファイル名 (File name)**」として入力します。
5. **「Location」**に、 ServerKeyRing.class を保管するための適切な場所を入力します。
6. 「**了解 (OK)**」を選択します。

生成される ServerKeyRing.class にはデフォルトで、いくつかの一般的な署名者の証明書が含まれます。それらには、VeriSign ルート署名者の証明書や Test 署名者の証明書などが含まれます。さらに、サーバーがクライアントを VeriSign クライアント証明書によって検証することを可能にする、 VeriSign クラス 1 ～ 4 の公開認証局の署名者の証明書も含まれます。これについては、81ページの『SSL クライアントを構成する』 で詳しく説明されています。

証明書要求を作成する

サーバー証明書を www.verisign.com から取得できるようになる前に、 SSL サーバーは証明書要求を作成してローカルに保管しなければなりません。

1. iKeyMan から**「Create」**を選択します。
2. **「Create New Certificate Request」**を選択します。
3. 証明書要求を完成させなければなりません。一部のフィールドは任意指定ですが、少なくとも以下のフィールドに入力しなければなりません (例を示します)。

Key Label	verisignServerCert
Key Size	1024
Common Name	servermachine.hursley.ibm.com
Organization	IBM UK
Country	GB

4. 証明書要求を保管するファイル名を指定して、**「了解 (OK)」**を選択します。それから iKeyMan は公開と秘密との鍵ペアを作成します。この処理には、ご使用のプロセッサー速度に応じていくらかの時間がかかります。

サーバー証明書を取得する

次の段階は、VeriSign の Web サイトに接続して、試用サーバー証明書を要求することです。ブラウザーで http://www.verisign.com を表示してから、https://digitalid.verisign.com/server/trial/trialIntro.htm のページに移行します。これで、試用サーバー証明書の登録の用意ができました。

1. **「Continue」**を選択します。
2. もう一度**「Continue」**を選択すると、**「Step 2 of 5 - Submit CSR」**が表示されます。
3. 証明書要求ファイルの内容を、表示されるテキスト枠に貼り付けます。証明書要求の内容は、以下のようなものです。

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIB1jCCAT8CAQAwgZUxIDAeBgNVBAMTF2NocmlzdHAuaHVyc2xleS5pYm0u
Y29tMRswGQYDVQQLExJDbGllbnRzICsgR2F0ZXdheXMxDzANBgNVBAoTBklC
TSBVSzETMBEGA1UEBxMKV0luY2hlc3RlcjEOMAwGA1UECBMFSGFudHMxETAP
BgNVBBETCFNPMjEgMkpOMQswCQYDVQQGEwJHQjCBnzANBgkqhkiG9w0BAQEF
AAOBjQAwgYkCgYEAkAth9Ar6k6ijNZ3JxdPGH6yiikwYTuA0RZDLZBSpaSEx
4qNKN/CrdF1LgfQYbZcN5NGCeC4sC478NhT+ltf5dnR3pNWBzEzmWn5mN0lH
tqJ3oibOUmDui+tQc2J9z6iRBKjkcQwjPlJp0sp5KKsev1ahAETL7LqmqMIq
pJlzKi0CAwEAAaAAMA0GCSqGSIb3DQEBBAUAA4GBAHaMHpizPs8Q3bi3I6dh
4yw0UNhojTlS1+ffiph3hK98lMHJuMztr0UMBLl/SZGNw85OJRiWuDjGYUQW
inJ0uNH34IUsnygBmt78+WlXT5nJuayg+UrAc5Ao2H8QZpRE5Sfaoc8lQcvY
plTggCdMxpYN7I33LrZDl3Po0TT8gjxQ
-----END NEW CERTIFICATE REQUEST-----
```

4. **「Continue」**を選択します。
5. 次のページでは、証明書要求の内容を検査することができます。要求された個人に関する詳細も入力しなければなりません。完全な VeriSign サーバー ID (試用版ではなく) を要求した場合、これらの詳細はご使用のアプリケーションを認証するために使用されます。
6. 詳細を入力して、VeriSign 合意文書を読み終えたなら、**「Accept」**を選択します。
7. 次の段階 (**5 つのうちの 4 番目**) は、 Test 署名者の証明書ルートを、SSL サーバーへの接続に使用するブラウザーにインストールすることです。
   Java アプレットから CICS Transaction Gateway への SSL 接続の場合、クライアント・アプレットにはデフォルトで Test 署名者の証明書を含むクライアント KeyRing クラスが必要なので、そのインストールは必要ありません。しかし、HTTPS プロトコルはブラウザー内のリポジトリーを使用するので、 Test 署名者の証明書をインストールしなければなりません。完全な VeriSign サーバー証明書を申し込む場合には、 Test 署名者の証明書をインストールする必要がないことに注意してください。

VeriSign 試用版サーバー証明書は、個人の詳細情報として入力したアドレスに E メールで送られます。これには通常、1 時間 から 3 時間かかります。

サーバー証明書をサーバー KeyRing に受け入れる

サーバー証明書を取得したなら、 iKeyMan を使用してそれをサーバー KeyRing に「受け入れる」必要があります。

1. まず、テキスト・エディターを使用してサーバー証明書データを空のテキスト・ファイルにコピー・アンド・ペーストします。
2. iKeyMan から、プルダウン・メニューの「**Personal Certificates**」を選択します。これは、「**Key database content**」ラベルの下にあります。
3. **「受信 (Receive...)」**を選択します。

4. サーバー証明書データの入ったテキスト・ファイルの位置を指定します。このデータは Base64 エンコードの ASCII 形式です。
5. 「**了解 (OK)**」を選択します。
6. 「**Key Database File**」を選択してから、「**終了 (Exit)**」を選択します。

これで、**ServerKeyRing.class** が CICS Transaction Gateway で使用できるようになります。それにはデフォルトの署名者の証明書、および VeriSign サーバー証明書とそれに対応する秘密鍵が含まれます。

SSL クライアントを構成する

通常の (デフォルトの) 操作では、CICS Transaction Gateway は SSL ハンドシェークを実行するときにはサーバー認証だけを使用し、クライアントに必要なことは、提供されたサーバー証明書を受け取ることだけです。サーバー認証が機能するためには、クライアント KeyRing クラスに、または HTTPS の場合はブラウザー証明書リポジトリーに、サーバー証明書の署名者の証明書がなければなりません。この例では、署名者の証明書は VeriSign Test 署名者の証明書です。

サーバー KeyRing クラスの場合と同様に、 iKeyMan を使用してクライアント KeyRing クラスを作成すると、それには最も一般的な署名者の証明書のデフォルト・セレクションが含まれることになります。

サーバー認証に加えて、CICS Transaction Gateway はクライアント認証もサポートします。このオプションが使用可能になっているときに、 ssl: または https: ハンドラーに対して接続を試みる場合、クライアントが自身のクライアント証明書 (ディジタル ID としても知られる) を持っている必要があります。

以下の節では、VeriSign ディジタル ID を取得して必要なクライアント KeyRing クラスを作成する方法について説明します。

クライアント証明書を取得する

サーバー証明書の取得の場合とは対照的に、どの形式の証明書要求を生成する場合にも iKeyMan を使用する必要はありません。 VeriSign は、Netscape または Internet Explorer のどちらかのブラウザーを使用してクラス 1 ディジタル ID を取得するための手段を提供しています。

クライアント証明書を取得して、ブラウザー・リポジトリーにインストールすると、#PKCS12 ファイルとして知られる安全な保管場所に、証明書データを、それに関連する秘密鍵とともにエクスポートできるようになります。このファ

イルはパスワードで保護されていて、 iKeyMan ツールを使用してクライアント KeyRing クラスにインポートできます。

1. Netscape Communicator バージョン 4.5 を使用して、ブラウザーで www.verisign.com を表示し、 **Individual Certificates** のリンクをたどってください。
2. https://digitalid.verisign.com/client/class1Netscape.htm の HTML ページに到達したら、要求されている詳細を入力します。その詳細は VeriSign によって、クライアント・アプリケーションの認証に使用されます。
3. 「**Accept**」を選択します。
4. ここで、Netscape は秘密鍵を生成します。この秘密鍵を保護するためのパスワードが要求されます。
5. VeriSign は申し込みの際に通知したアドレスに、クラス 1 ディジタル ID のダウンロードおよびインストールに関する情報を E メールで送ります。

インストール・プロセスの際、Netscape Communicator ではクライアント証明書を、パスワードで保護されたファイル (#PKCS12 形式) に格納できます。iKeyMan ツールは #PKCS12 形式ファイルをサポートしており、証明書 (およびその秘密鍵) をクライアント KeyRing クラスにインポートすることができます。

クライアント KeyRing を作成する

クライアント KeyRing クラスを作成するには、以下のようにします。

1. ctgikey を開始します。
2. 「**Key Database File --> New**」を選択します。
3. 「**Key Database Type**」から、「**SSLight key database class**」を選択します。
4. `ClientKeyRing.class` を**ファイル名**として入力します。
5. 「**Location**」に、 ClientKeyRing.class を保管するための適切な場所を入力します。
6. 「**了解 (OK)**」を選択します。

これで、以下のデフォルトの署名者の証明書を含む ClientKeyRing.class が作成されます。

```
VeriSign Class 1 Public Primary Certification Authority
VeriSign Class 2 Public Primary Certification Authority
VeriSign Class 3 Public Primary Certification Authority
RSA Secure Server Certification Authority
Thawte Personal Basic CA
VeriSign Test CA Root Certificate
Thawte Personal Premium CA
Thawte Premium Server CA
Thawte Personal Freemail CA
Thawte Server CA
```

クライアント証明書をクライアント KeyRing にインポートする

クライアント証明書を含む #PKCS12 保管ファイルをインポートするには、以下のようにします。

1. 「**Key database content**」ラベルの下にあるプルダウン選択肢から「**Personal Certificates**」を選択します。
2. 「**インポート (Import...)**」を選択します。
3. 「**Key file type**」を PKCS12 ファイルに設定します。
4. 保管した #PKCS12 ファイルの位置を指定します。
5. 「**了解 (OK)**」を選択します。
6. 「**Key Database File**」を選択してから、「**Close**」を選択します。
7. 「**Key Database File**」を選択してから、「**終了 (Exit)**」を選択します。

これで、**ClientKeyRing.class** が CICS Transaction Gateway で使用できるようになります。それにはデフォルトの署名者の証明書、および VeriSign クライアント証明書 (クラス 1 ディジタル ID) とそれに対応する秘密鍵が含まれます。

自己署名された証明書の使用

CICS Transaction Gateway は、証明書に「自分で署名する」メカニズムを提供します。自分自身を独自の認証局として確立し、サーバー側 (CICS Transaction Gateway) およびクライアント側 (ブラウザー) の X.509 ディジタル証明書を生成します。

この節では、証明書管理インターフェース iKeyMan を使用して、 SSL サーバーと SSL クライアントの両方を構成する方法について説明します。

SSL サーバーを構成する

SSL サーバーの構成には、サーバー KeyRing クラスの作成、自己署名された証明書の生成、およびそれを KeyRing クラスに受け入れることが含まれます。

サーバー KeyRing を作成する

最初のステップは、サーバー証明書と鍵情報を保管するためのサーバー KeyRing クラス・ファイルを作成することです。

1. ctgikey を開始します。
2. 「**Key Database File --> New**」を選択します。
3. 「**Key Database Type**」から、「**SSLight key database class**」を選択します。
4. `ServerKeyRing.class` を**ファイル名**として入力します。
5. **「Location」**に、 ServerKeyRing.class を保管するための適切な場所を入力します。
6. 「**了解 (OK)**」を選択します。
7. KeyRing ファイルのパスワードを入力します。

サーバー証明書を生成する

これで、自己署名されたサーバー証明書を作成して、秘密鍵とともにサーバー KeyRing クラスに格納する準備ができました。

1. iKeyMan から、プルダウン・メニューの「**Personal Certificates**」を選択します。これは、「**Key database content**」ラベルの下にあります。
2. 「**New Self-Signed...**」を選択します。
3. 証明書の詳細を完成させなければなりません。一部のフィールドは任意指定ですが、少なくとも以下のフィールドに入力しなければなりません (例を示します)。

Key Label	exampleServerCert
Version	X509 V3
Key Size	1024
Common Name	clientmachine.hursley.ibm.com
Organization	IBM UK
Country	GB
Validity Period	365 (日)

4. 「**了解 (OK)**」を選択します。それから iKeyMan は公開と秘密との鍵ペアを作成します。この処理には、ご使用のプロセッサー速度に応じていくらかの時間がかかります。

5. iKeyman が自己署名されたサーバー証明書を正常に作成すると、それは「Personal Certificates」ウィンドウに表示されます。証明書は生成プロセスで指定した**キー・ラベル**に応じて名付けられます。この例では、exampleServerCert です。
6. **exampleServerCert** を強調表示して、「**View/Edit**」を選択します。「**issued to**」および「**issued by**」テキスト・ボックスの証明書情報が同じであることに注意してください。したがって、証明書要求者 (issued to) は署名者 (issued by) と同じです。この証明書を提示するサーバーとの間に SSL 接続を確立するためには、クライアントが exampleServerCert の署名者を信用しなければなりません。これを行うには、クライアント鍵リポジトリーに、exampleServerCert を提示するサイトの署名者の証明書が含まれていなければなりません。

サーバー署名者の証明書のエクスポート

ここで、SSL サーバーの署名者 (公開) 証明書をエクスポートします。

1. **exampleServerCert** を強調表示して、「**証明書の抽出 (Extract Certificate...)**」を選択します。
2. エクスポートする証明書の**データ・タイプ**を選択します。これは通常、Base64 エンコード ASCII 形式のデータです。
3. エクスポートする証明書の名前 / 場所を入力します。この例では、exampleServercert.arm を使用します。
4. 「**了解 (OK)**」を選択します。

エクスポートした証明書は安全な場所に保管して、この特定の SSL サーバー証明書とハンドシェークする必要のあるクライアント鍵リポジトリーにインポートします。

SSL クライアントを構成する

CICS Transaction Gateway によって使用される SSL ハンドラーがサーバー認証だけをサポートするように構成されている場合、自己署名されたクライアント証明書を生成する必要はありません。この場合、クライアント KeyRing クラスに含まれる必要があるのは、サーバーの署名者の証明書だけです。それはこの例では、エクスポートされた証明書ファイル exampleServercert.arm です。

以下のステップは、クライアント KeyRing を作成してサーバーの署名者の証明書をインポートするプロセスを示しています。

クライアント KeyRing を作成する

1. ctgikey を開始します。
2. 「**Key Database File --> New**」を選択します。
3. 「**Key Database Type**」から、「**SSLight key database class**」を選択します。
4. ClientKeyRing.class を**ファイル名**として入力します。
5. **「Location」**に、 ClientKeyRing.class を保管するための適切な場所を入力します。
6. 「**了解 (OK)**」を選択します。
7. KeyRing のパスワードを入力してください。
8. ClientKeyRing.class が生成されたなら、デフォルト署名者のリストが表示されます。

サーバーの署名者の証明書をインポートする

次の段階は、サーバーの署名者の証明書をインポートすることです。

1. 「**Add**」を選択します。
2. 保管したサーバーの Base64 エンコード ASCII 形式の証明書ファイル、この例では exampleServercert.arm の位置を指定します。
3. この署名者の証明書に固有のラベル、たとえば「My Self-Signed Server Authority」を指定します。
4. 「**了解 (OK)**」を選択します。この新しい署名者の証明書が、デフォルト署名者リストに追加されます。

生成された ClientKeyRing.class は、サーバー認証をサポートするように構成された CICS Transaction Gateway の SSL プロトコルで使用できます。 HTTPS プロトコルを使用してアプレットからのセキュリティー接続を確立した場合、アプレットを実行している特定のブラウザーは exampleServercert.arm を、その鍵 / 証明書リポジトリーにインポートする必要があります。

Base64 エンコード ASCII 形式の証明書ファイルをインポートする方法については、ご使用のブラウザーのオンライン・ヘルプを参照してください。

クライアント証明書を生成する

クライアント認証では、クライアントの KeyRing クラスに、接続中のクライアントの識別に使用する自己署名された証明書が含まれていることも必要です。

自己署名されたサーバー証明書の生成の場合と同じステップによって、クライアント KeyRing クラス (この例では、 ClientKeyRing.class) を作成 (または既存のものをオープン) します。その後、

1. iKeyMan から、プルダウン・メニューの「**Personal Certificates**」を選択します。これは、「**Key database content**」ラベルの下にあります。
2. 「**New Self-Signed...**」を選択します。
3. 証明書の詳細を完成させます。
4. 「**了解 (OK)**」を選択します。それから iKeyMan は公開と秘密との鍵ペアを作成します。この処理には、ご使用のプロセッサー速度に応じていくらかの時間がかかります。
5. SSL サーバーと同様に、クライアントは署名者の証明書を、 SSL サーバーの鍵 / 証明書リポジトリーにインストールしなければなりません。これにより、SSL サーバーはクライアントの詳細情報を検査できるようになります。 **exampleClientCert** を強調表示して、「**Extract Certificate...**」を選択します。
6. エクスポートする証明書の**データ・タイプ**を選択します。これは通常、Base64 エンコード ASCII 形式のデータです。
7. エクスポートする証明書の名前 / 場所を入力します。この例では、exampleClientcert.arm を使用します。
8. 「**了解 (OK)**」を選択します。

エクスポートした証明書は安全な場所に保管して、この特定の SSL クライアント証明書とハンドシェークする必要のあるクライアント鍵リポジトリーにインポートします。

アクセスをサーバー KeyRing に制限する

サーバー KeyRing の内容はパスワードによって暗号化されます。しかし、以下のことを実行するように強くお勧めします。

- 正しいファイル許可が割り当てられていることを確認する
- できれば、アクセスを CICS Transaction Gateway マシンに制限する。

複数のサーバーで証明書を共有することは良いことではありません。複数のサーバーで、特に異なるマシン上で実行している場合、秘密鍵を共有することは望ましくありません。秘密鍵は決して他の人に渡さないでください。

SSL および HTTPS 用に CICS Transaction Gateway を構成する

CICS Transaction Gateway はそのクライアントとの通信用に、TCP、HTTP、SSL、および HTTPS を含む各種のプロトコルをサポートします。 SSL および HTTPS プロトコルを使用するには、構成ツールを使ってそれらを使用可能にする必要があります (32ページの『構成ツールの使用』を参照)。それから、CICS Transaction Gateway を実行したときに、SSL および HTTPS プロトコルが開始されます。セキュリティー・プロトコルは、どちらか 1 つ、または両方を指定することができます。これらのハンドラーを指定して CICS Transaction Gateway を開始すると、ポート 8050 で SSL 要求を聴取し、ポート 443 で HTTPS 要求を聴取します。

SSL および HTTPS プロトコルに固有の設定は、以下のとおりです。

KeyRing クラス名
この設定値はサーバー KeyRing クラス・ファイルの名前を指定します。 CLASSPATH 環境変数は、このクラスが見つかるように設定する必要があります。

KeyRing パスワード
この設定値は、暗号化されたサーバー KeyRing を復号するために使用されるパスワードを指定します。

クライアント認証の使用
この設定値は、クライアント認証を使用することを指定します。 (デフォルトでは、サーバー認証を使用します。)

構成ツールを使用すると、必要な項目が構成ファイルに作成されます。 SSL および HTTPS プロトコル項目の完全な例が、サンプルの構成ファイル CTGSAMP.INI に入っています。

CICS Transaction Gateway には、SSL および HTTPS 接続の確立に使用できる、2 つのデフォルトの KeyRing クラス・ファイルが用意されています。**ClientKeyRing** および **ServerKeyRing** は、どちらもパスワード **default** を使用して暗号化されるので、これらはテスト環境でしか使用しないようお勧めします。したがって、SSL および HTTPS プロトコルを使用するためには、前述の節で述べられているように、独自の KeyRing を生成するようお勧めします。

クライアント KeyRing の指定

どちらのセキュリティー・プロトコルを使用するかにより、クライアント KeyRing が必要かどうかが決まります。 HTTPS プロトコルは、ブラウザー (クライアント) そのものが、CICS Transaction Gateway (サーバー) との安全な

接続を確立するために必要な機能を持っている場合に、 Java アプレット内からの通信をセキュリティーすることを目的として作られています。このため、HTTPS プロトコルでは、サーバー側の KeyRing しか指定する必要がなく、クライアント側はブラウザー・ソフトウェアが処理します。

SSL プロトコルは、CICS Transaction Gateway がサーバーおよびクライアントをセキュリティーした形で処理するコードを持つ、もっと低いレベルで作られています。 SSL プロトコルの場合は、サーバー および クライアント両方の KeyRing クラス・ファイルが必要です。

クライアント KeyRing は、 SslJavaGateway.class に静的なフィールドを設定することで指定します。このクラスは、CICS Transaction Gateway クライアント側コードの一部を成します。

SslJavaGateway.class は 2 つのメソッドを提供します。 1 つはクライアントの KeyRing を「取得」するためのメソッドで、もう 1 つはクライアントの KeyRing を「設定」するためのメソッドです。

```
public static void setKeyRing(String strSetKeyRing, String strSetKeyRingPW)
public static String getKeyRing()
```

SSL プロトコルが使用するクライアントの KeyRing クラスを設定するには、クライアント・アプリケーションまたはアプレットが、以下のメソッドの静的呼び出しを作成します。

SslJavaGateway.setKeyRing(CLASSname, PASSword);

ここで、

- CLASSname は、クライアント用に生成された Java KeyRing クラスのクラス名を示します。
- PASSword は、埋め込まれている X.509 認証を復号するために使用します。

SslJavaGateway.class は、「取得」メソッドである **getKeyRing()** も提供し、現在指定されているクライアント KeyRing の CLASSname を返します。

SSL/HTTPS プロトコルを使用してクライアント・アプリケーションまたはアプレットと CICS Transaction Gateway との接続を確立することと、 TCP または HTTP プロトコルを使用して確立することに違いはありません。クライアント・アプリケーションまたはアプレットは、関連する URL を使用してその要求を単に CICS Transaction Gateway に「流す」だけです。たとえば、SSL の場合は、アプリケーションは `ssl://transGatewayMachine:8050` を使用し、HTTPS の場合は、`https://transGatewayMachine:443` を使用します。

クライアント側プログラムの設計と実装については、「*CICS Transaction Gateway: Gateway プログラミング*」、および「CICS Transaction Gateway プログラミング・インターフェース」の HTML ページを参照してください。

第6章 CICS Transaction Gateway の操作

この章では、以下について説明します。

- 『Gateway の開始』
- 94ページの『Gateway の停止』

Gateway の開始

Gateway は、その Gateway がインストールされているオペレーティング・システムのコンピューターのコマンド・プロンプトから開始します。

最初に、作業ディレクトリーを CICS Transaction Gateway の bin ディレクトリーに設定します。

2 つの方法で CICS Transaction Gateway を開始できます。

- 『事前設定オプションでの Gateway の開始』
- 『ユーザー指定オプションでの Gateway の開始』

事前設定オプションでの Gateway の開始

Gateway を定義済みのオプションで始動するには、コマンド・プロンプトで `ctgstart` と入力して Enter キーを押すか、**CICS Transaction Gateway** アイコンを選択します。

この方法で Gateway を開始するときは、デフォルトの構成設定、または構成ツールを使って構成した設定が使用されます (32ページの『構成ツールの使用』を参照)。

Gateway コンソール・セッションが開始され、以下のメッセージが表示されます。

```
CCL6500I: Starting the Gateway with default values.
```

この後に、使用されている値を示す行が続きます。

```
CCL6502I: [ Initial ConnectionManagers = 1, Maximum ConnectionManagers = 100,
CCL6502I: Initial Workers = 1, Maximum Workers = 100, Port = 2345 ]
```

ユーザー指定オプションでの Gateway の開始

開始コマンドのユーザー定義可能なオプションは次のとおりです。

```
-port=port_number
-initconnect=number
-maxconnect=number
-initworker=number
-maxworker=number
-trace
-noinput
-tfile=pathname
-x
-tfilesize=number
-truncationsize=number
-dnsnames
-dumpoffset=number
-stack
```

ここで、

-port
ゲートウェイが listen する TCP/IP ポート番号を指定します。

-initconnect
ConnectionManager スレッドの初期番号を指定します。

-maxconnect
ConnectionManager スレッドの最大数を指定します。この値を *-1* に設定すると、ConnectionManager スレッドの数に制限は適用されません。

-initworker
Worker スレッドの初期番号を指定します。

-maxworker
Worker スレッドの最大数を指定します。この値を *-1* に設定すると、ConnectionManager スレッドの数に制限は適用されません。

-trace
標準トレースを使用可能にします (120ページの『トレース』を参照)。デフォルトでは、トレース出力は、どのデータ・ブロック (たとえば COMMAREA あるいはネットワーク・フロー) についても最初の 128 バイトだけを表示します。 CLASSPATH 変数の値およびコード・ページを含むその他の有益な情報は、トレース出力の先頭に表示されます。

Gateway の構成に構成ツールを使用した場合のトレース出力のデフォルト宛先は、CICS Transaction Gateway がインストールされている `bin` サブディレクトリーのファイル `CTG.TRC` です。構成ツールを使用しなかった場合は、トレース出力は `stderr` に書き込まれます。

-noinput
コンソールからの入力の読み取りを使用不可にします。

-tfile=*pathname*
トレースが使用可能の場合、*pathname* で指定されたファイルにトレース・メッセージが書き込まれます。これはトレース出力のデフォルト宛先をオーバーライドします (**-trace** オプションを参照)。

-x 完全デバッグ・トレースを使用可能にします (120ページの『トレース』を参照)。デフォルトでは、トレース出力は、データ・ブロック (たとえば COMMAREA あるいはネットワーク・フロー) 全体を表示します。また、Gateway の内部処理に関する情報を標準トレースよりも多く表示します。トレース出力の宛先については、**-trace** および **-tfile** オプションを参照してください。

デバッグ・トレースを行うと、パフォーマンスがかなり低下します。

-tfilesize=*number*
トレース出力ファイルの最大サイズを K バイトで指定します。

-truncationsize=*number*
値 *number* は、トレースに表示されるデータ・ブロックの最大サイズを指定します。このオプションを **-trace** または **-x** のいずれかのオプションと共に指定すると、デフォルト・サイズをオーバーライドすることができます。有効な値は正の整数です。値として *0* を指定すると、トレースにはデータ・ブロックは表示されません。

-dnsnames
TCP/IP のシンボリック・ホスト名をメッセージに表示できるようにします。詳細については、37ページの『TCP/IP ホスト名の表示』を参照してください

-dumpoffset=*number*
値 *number* は、データ・ブロックが始まるオフセットを指定します。オフセットが、表示されるデータの合計長よりも大きい場合は、オフセット *0* が使用されます。

-stack
Java の例外スタック・トレースを使用可能にします (120ページの『トレース』を参照)。 Java の例外は、Gateway の標準操作中に予想される例外も含めてトレースされます。それ以外のトレース出力は作成されません。トレース出力の宛先については、**-trace** および **-tfile** オプションを参照してください。

開始時のデフォルトをオーバーライドするには、コマンド・プロンプトで **ctgstart** と入力し、その後ろに必要な始動オプションを入力して、Enter キー

を押します。コマンド行に指定したオプションは、構成ツールを使用して指定したオプションをオーバーライド します。

以下の始動メッセージが表示されます。

```
CCL6501I: Starting the CICS Transaction Gateway with user specified values.
```

この後に、次のように、使用されている値を示す行が続きます。

```
CCL6502I: [ Initial ConnectionManagers = 10, Maximum ConnectionManagers = 100,
CCL6502I: Initial Workers = 10, Maximum Workers = 100, Port = 2345 ]
```

始動オプションに関するヘルプを表示するには、以下のコマンドを入力します。

```
ctgstart ?
```

Gateway の停止

Gateway を停止するには、以下のようにします。

- Gateway を **-noinput** オプションを指定せずに始動した場合は、Gateway コンソール・セッションで適正な文字を入力してから Enter キーを押すことで、Gateway を停止することができます。利用できる文字は、国ごとにローカライズされている可能性があります。利用可能なデフォルト文字は Q または - 文字です。

  どの文字で Gateway を停止するかは、単に Gateway コンソール・セッションで Enter キーを押すだけで判別できます。次のようなメッセージが表示されます。

  ```
  CCL6508I: Type Q or - to stop the CICS Transaction Gateway.
  ```

- -noinput パラメーターを使用した場合は、別の方法を使用して Gateway を停止する必要があります。以下に例を示します。
  - **kill** コマンドを使用します。

第7章 CICS Transaction Gateway 端末サーブレット

この章では、端末サーブレットによって Web ブラウザーを使用して CICS 3270 アプリケーションにアクセスする方法を説明します。

- 『CICS Transaction Gateway 端末サーブレットとは』では、サーブレットとは何かについて、特に CICS Transaction Gateway 端末サーブレットの機能について説明します。
- 97ページの『端末サーブレットのインストールおよび構成』では、端末サーブレットのインストールおよび構成方法について説明します。
- 105ページの『端末サーブレットの使用』では、端末サーブレットを使用して何を行えるかについて、また、それを Web ページから起動する方法について説明します。
- 111ページの『プロパティーおよびパラメーターのリファレンス』には、端末サーブレットに対して指定するサーブレット・プロパティーに関する詳細なリファレンスを記載しています。
- 117ページの『CICS® Transaction Server for OS/390® の Web インターフェース』では、 CICS Web インターフェース用に生成された HTML テンプレートを、端末サーブレットが使用する方法を説明します。

端末サーブレットを構成して使用する前に、この章の全体を読み通すことをお勧めします。

CICS Transaction Gateway 端末サーブレットとは

サーブレットとは、Web サーバー・マシン上で、または WebSphere Application Server (97ページの図8 を参照) などの Java 対応の Web サーバーまたはサーブレット・エンジン内で実行される Java プログラムです。 Java サーブレットは、CGI (コモン・ゲートウェイ・インターフェース) プログラムに代わるものとして人気が高まっています。端末サーブレットは、IBM CICS クライアント バージョン 2 に付属の CGI プログラムであった CICS Internet Gateway に置き代わるものです。

サーブレットは、Web サーバーの開始時に自動的にロードすること、またはクライアントがサーブレットのサービスを初めて要求するときにロードすることができます。 CGI プログラムとは異なり、サーブレットは継続的です。つまり、一度ロードされると実行を続けて、追加のクライアント要求を待機してい

ます。これにより、プロセスの作成および破棄のための時間のロスがないので、サーブレットは CGI プログラムよりも高速になります。

Java プログラムであるため、サーブレットは複数のオペレーティング・システム間で移植性があります。さらに、サーブレット・インターフェースは標準化されているため、異なるプラットフォーム上の異なる Web サーバーが使用できます。

端末サーブレットにより、Web ブラウザーを使用して CICS 3270 アプリケーションにアクセスすることができます。そのためには、端末サーブレットを起動して CICS トランザクションを開始する Web ページを作成し、 CICS サーバーから送られる画面情報を表示して、画面を CICS に送り返します。

端末サーブレットは以下のことを行えます。

- **単純な端末エミュレーターのように振る舞う。**

  CICS 画面は HTML フォームの一部としてブラウザーに表示されます。フォームのボタンの 1 つを押すと、フォームは発信されてデータが CICS に送られます。

- **CICS 画面からのデータを、HTML テンプレート・ファイルに置き換える。**

  HTML テンプレート・ファイルを使用して、 CICS データをブラウザー上に表示する方法を判別します。端末サーブレットは、HTML テンプレート・ファイル内の変数を CICS 画面情報で置き換え、その出力をブラウザーに送ります。詳しくは、108ページの『変数置換の使用』 をご覧ください。

- **サーバー側インクルードを使用して、変数情報を Web ページに組み込む、またはユーザーとの対話なしで端末エミュレーターを駆動する。**

  サーバー側インクルードとは、HTML ファイル内のステートメント (.shtml) で、 Web ページがブラウザーに戻される前に Web サーバーによって処理されます。たとえば、サーバー側インクルードを使用して端末サーブレットを起動し、フィールドの値を CICS 画面に表示できます。詳しくは、107ページの『サーバー側インクルードを使用してサーブレットを起動する』 をご覧ください。

- **CICS 画面を Web ページにマップする。**

  端末サーブレットのページ・マッピング・プロパティーを使用して、特定の Web ページを特定の CICS 画面に関連付けることができます。たとえば、端末がアイドル状態でトランザクションが実行されていないときに Web ページを表示するように指定できます。詳しくは、114ページの『ページ・マッピングのプロパティー』 をご覧ください。

EPI プログラミングと同様に、端末サーブレットによって既存の CICS トランザクションに新しいフロントエンドを作成できるようになります。さらに、それは Java プログラミングの能力、柔軟性、およびプラットフォーム非依存性を活用することを可能にします。

注: 端末サーブレットは、3270 データ・ストリームにおける DBCS フィールドをサポートしていません。

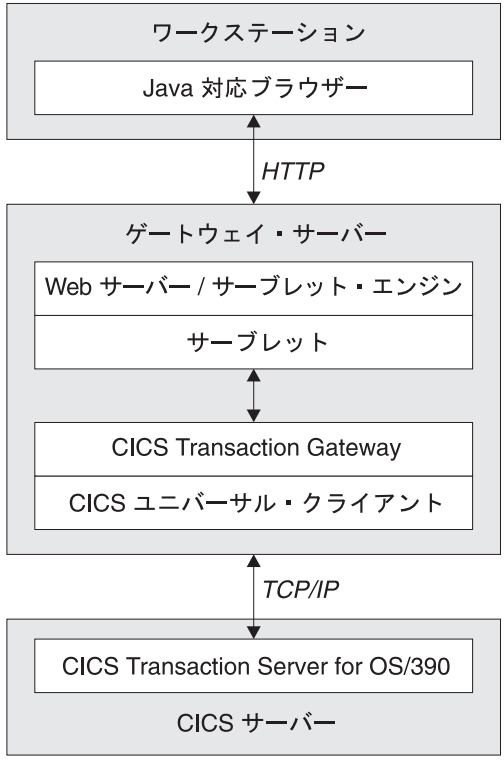


図 8. URL によって起動される端末サーブレットを使用する CICS Transaction Gateway

端末サーブレットのインストールおよび構成

この節では、端末サーブレットを Web サーバー・マシン上にインストールする方法、および端末サーブレットの初期化パラメーターを設定する方法について説明します。

端末サーブレットをインストールおよび構成する手順は、Web サーバーによって異なります。しかし、少なくとも以下の事柄が必要です。

- Web サーバーの CLASSPATH および PATH 設定値を構成する
- 端末サーブレットを Web サーバーの構成に追加する
- サーブレット初期化パラメーターを構成する
- セキュリティー、ログ記録、およびWeb サーバーがサーバー側インクルードを処理できるかどうかなど、他の構成オプションを検討する

特定の Web サーバー上に端末サーブレットをインストールして構成する例は、入手可能なすべての構成資料で提供されています。 138ページの『サンプル構成の資料』を参照してください。その文書で説明されている原則は、他の Web サーバーにも適用されます。

使用する Web サーバーには関係なく、サーブレットのインストールおよび構成についてのガイダンスとして、その文書を参照してください。

Web サーバーの CLASSPATH および PATH 設定値を構成する

2 つの CICS Transaction Gateway jar ファイル (**ctgclient.jar** および **ctgserver.jar**) が、Web サーバーのクラスパスに存在することを確認してください。 WebSphere Application Server の場合、これは WebSphere の管理ツールの Java Engine セクションで行います。

クラスパスを適切に設定しないと、サーブレットを更新できない、およびクラス **com.ibm.ctg.servlet.TerminalServlet** をロードできないという趣旨のメッセージを受け取ることがあります。

さらに、CICS Transaction Gateway bin サブディレクトリーが Web サーバーのパスに存在することも確認してください。

端末サーブレットを Web サーバーの構成に追加する

端末サーブレットを Web サーバーの構成に追加するには、サーブレット名を指定して、そのサーブレット・インスタンスを端末サーブレット・クラス (**com.ibm.ctg.servlet.TerminalServlet**) に関連付けなければなりません。WebSphere Application Server の場合、これは WebSphere の管理ツールの Servlet Configuration セクションで行います。

端末サーブレットの複数のインスタンスを構成できます。それぞれが異なるサーブレット名を持っていても、同じサーブレット・クラスを参照します。実際には、異なる端末サーブレット・インスタンスを接続する CICS サーバーごとに作成しなければなりません。端末サーブレットの各インスタンスは、異なる

初期化パラメーターを使用することがあり、このことは異なる CICS サーバーを構成する場合に役立つことがあります。

サーブレット初期化パラメーターを構成する

端末サーブレットを Web サーバーの構成に追加した後、端末サーブレットの初期化パラメーターを設定しなければなりません。 WebSphere Application Server の場合、これは WebSphere の管理ツールの Servlet Configuration セクションで行います。

続く節では、端末サーブレットの働きについて掘り下げて説明します。これは、適切なサーブレット初期化パラメーターを設定するために役立ちます。

端末サーブレットは、サインオン機能を持つ端末をサポートするようになりました。このタイプの端末を提供する初期化パラメーターも含め、使用可能な初期化パラメーターの完全セットについては、111ページの『プロパティーおよびパラメーターのリファレンス』を参照してください。

最初に、以下のパラメーターをいくつか設定してください。

表 4. サーブレット初期化パラメーター

パラメーター名	注記
servlet@propertyFile	サーブレットは構成情報をプロパティー・ファイルからロードできます。サンプルの servlet.properties ファイルが、サブディレクトリー /samples/terminalservlet にあります。 プロパティー・ファイルを使用するには、このパラメーターを、端末サーブレットをロードするプロパティー・ファイルの完全パス名に設定します。プロパティー・ファイルを指定する場合、そのファイルを他のすべてのサーブレット・パラメーターについて使用したいかもしれません。その場合、**servlet@propertyFile** が、設定しなければならない唯一のパラメーターです。
servlet@extendedLogging	サーブレットは、エラー・メッセージなどのログ情報を標準サーブレット・ログに書き込みます。より多くの情報をログ・ファイルに書き込むためには、このプロパティーを「true」に設定します。サーブレットを最初に構成するときにはこれが役立つことがありますが、サーブレットを正しく設定した後にはオフにしてください。

表 4. サーブレット初期化パラメーター (続き)

パラメーター名	注記
pool@maxTerminals	このプロパティーを、CICS サーバーへの接続数の最大可能値に設定します。 CICS サーバーに、この数の同時接続を処理するための空タスクが十分にあることを確認してください。さらに、CICS クライアント**最大要求数**構成設定値がこのプロパティー以上であることも確認してください。
pool@serverName	端末サーブレットの各インスタンスが接続できる CICS サーバーは 1 つだけです。これにより、リソースの使用およびセキュリティーをより十分に制御できるようになります。構成内の適切な**「サーバー名」**設定値で定義した必要な CICS サーバーの名前に、このプロパティーを設定します。このプロパティーを設定しない場合、デフォルトの CICS サーバーが使用されます。
pool@gatewayUrl	サーブレットがネットワークを介して CICS Transaction Gateway に接続するようにしたい場合、このプロパティーを設定します。

プロパティー・ファイルの名前とは別に、すべてのサーブレット・プロパティーはプロパティー・ファイルからロードするか、またはサーブレット初期化パラメーターとして設定できます。初期化パラメーターは、プロパティー・ファイルの項目をオーバーライドします。

サーブレット・プロパティーの名前には、大文字小文字の区別はありません。

端末プールの使用

端末サーブレットは端末プールを使用します。その端末プールの振る舞いは、サーブレット構成プロパティーによって制御できます。デフォルトの振る舞いは、最大 **pool@maxTerminals** に達するまでは、ユーザーが要求する度に、新規の端末が CICS に接続されるということです。ユーザーが端末の使用を終えると、接続は切断されます。

pool@minTerminals プロパティーを設定した場合、サーブレットの開始時に、指定した数の端末が CICS に接続されるので、その数を最大にいくつかの端末が CICS に接続されているにもかかわらず未使用の状態になっている可能性があります。これにより、新規のユーザーは接続を待つことなく、すぐに端末を割り当てられます。

pool@reusingTerminals プロパティーを設定すると、端末の再使用が可能になります。つまり、ユーザー・セッションが終了すると、端末は切断される代わりに新規のユーザーに割り当てられます。 **pool@minTerminals** が 0 の場合、このプロパティーは無効になります。端末で実行中のすべてのトランザクションは、再使用される前に終了し、画面は消去されます。しかし、ユーザーが CICS (たとえば CESN を使用して) にサインオンしている場合、端末は再使用されるときにもサインオンされています。そのため、ご使用の環境に適切であることが明らかでなければ、このプロパティーをオンにしないでください。

pool@idleTimeout プロパティーによって、ユーザーに割り当てられた端末が指定された期間使用されていない場合、その割り当てを解除することができます。このプロパティーのデフォルトは 0、つまりタイムアウトなしです。

値 **pool@connectedTerminals**、**pool@freeTerminals**、および **pool@terminalsInUse** を指定して画面要求を使用することにより、端末サーブレットの特定のインスタンスで接続、空き、または使用中の端末数を照会できます。

ページ・マッピング

ページ・マッピングは端末サーブレットに、特定の CICS 画面または端末状態に対して表示する Web ページを指示します。

表示するページは、現行のページ・マッピング設定値、現行の画面ハンドラー・クラス、および端末の状態に基づいてサーブレットが選択します。

端末サーブレットは最初に、ページ・マッピング・プロパティーを見て、セッションの現在の状態に対して以下のように特定のページが設定されているかどうかを調べます。

セッション状態	ページ・マッピング・プロパティー
エラーが発生した	**page@error**
端末がアイドル状態 - 実行中のトランザクションがない	**page@idle**
端末が切断されている	**page@disconnected**
現行の画面に画面ハンドラーがある	画面ハンドラーのクラス名に対するページ・マッピング

特定のページが見つからない場合、ページ・マッピング・プロパティー **page@default** で識別されるページが使用されます。これが設定されていない場合、代わりにエラー・メッセージが表示されます。

ページ・マッピング・プロパティーの値は、以下の 2 つの方法で設定できます。

1. URL、たとえば、

   ```
   http://webserver/index.html
   ```

2. 変数置換のためにロードされるテンプレート・ファイル。これは ″file://″ で始まる必要があります。たとえば、

   ```
   file://d:/pages/template.html
   ```

   servlet@templateDir プロパティーを設定した場合、テンプレート・ファイル名はテンプレート・ディレクトリーからの相対パスで指定できます。

ページ・マッピング・プロパティーの設定値の例は、以下のとおりです。

```
page@error=http://server/errors.html
page@idle=http://server/idle.html
page@disconnected=/ctglab/html/servlet/epissam.html
page@default=file://epissam1.html
```

特定の画面に対してページ・マッピングを設定するには、その画面を認識できる画面ハンドラー bean を作成しなければなりません。その後、**pool@handlerPath** プロパティーを設定して、画面ハンドラー bean が端末サーブレットによって確実にロードされるようにします。たとえば、MAP1ScreenHandler と呼ばれる画面ハンドラー bean を、testmaps と呼ばれるパッケージ内に作成した場合、その画面ハンドラーに対するマッピング・プロパティーは次のとおりです。

```
page@testmaps.MAP1ScreenHandler.
```

クラスの名前には大文字小文字の区別があります。標準ページ・マッピングについては、以下のように、表示する Web ページの名前に対してプロパティーを設定します。

```
page@testmaps.MAP1ScreenHandler=http://server/test.html
```

ページ・マッピングは端末サーブレットへの要求パラメーターにも設定できます。その場合、それはその要求だけに適用され、使用されているサーブレットのインスタンスに関連したページ・マッピングをオーバーライドします。適切な要求を使用して、ページ・マッピングの現在の設定値を照会できます (116ページの『表示可能プロパティー』 を参照)。

画面ハンドラー bean および端末の切断

端末が正常に切断されるためには、実行中のトランザクションから終了しなければなりません。これを達成するため、デフォルトでは端末サーブレットは、AID (アテンション ID) PF3 キーを CICS に送信します。しかし、これを行えないトランザクションを実行している場合、関連する画面の終了方法を知る 1 つまたは複数の画面ハンドラー bean を作成してください。

CICS Transaction Gateway の付属で提供される BMS マップ変換ユーティリティー (BMSMapConvert) を使用して、画面ハンドラー bean を自動的に作成できます。または、独自のクラスを記述することもできます。画面ハンドラーとその作成方法については、「*CICS Transaction Gateway: Gateway プログラミング*」を参照してください。

画面ハンドラーをサーブレットにロードするには、 **pool@handlerPath** プロパティーを設定します。ハンドラー・パスには、.jar ファイル、.zip ファイル、およびディレクトリーを含めることができます。しかし、サブディレクトリーは検索されません。たとえば、サーブレット・プロパティー・ファイルに、

```
pool@handlerPath=d:/handlers/handlers.jar;d:/handlers/test
```

を追加して、画面ハンドラー・クラスを .jar ファイルおよびディレクトリーからロードできます。このプロパティーの設定方法については、サンプルのプロパティー・ファイルを参照してください。

認識する画面ハンドラーが他にない場合に画面を終了するために使用する、デフォルトの画面ハンドラーも指定できます (プロパティー **pool@defaultHandler** を使用する)。サンプルのデフォルト画面ハンドラーは **com.ibm.ctg.epi.DefaultScreenHandler** で、 AID (アテンション ID) PF3 を CICS に送ります。このクラスのソースは、ディレクトリー samples/java/com/ibm/ctg/epi にあります。クラスパスまたはハンドラー・パスからロードできる任意の画面ハンドラー bean を、デフォルトの画面ハンドラーとして使用できます。

サーブレットは、実行中のトランザクションの終了を無期限に試行することはありません。プロパティー **pool@exitRetryLimit** が、トランザクションを終了させるための試行を何回行うかを制御します。このプロパティーのデフォルト値は 10 です。しかし、ご使用の環境に対してこの回数が十分であることを確認してください。終了させる試行を指定の回数行った後に、端末上でまだ実行しているトランザクションをサーブレットが見つけた場合、その端末は ‘非活動’ とみなされて、切断されることもユーザーに割り当てられることもなく

なります。非活動の端末を廃棄する唯一の方法は、サーブレットをアンロードしてから、CICS Transaction Gateway を停止および再始動して、サーブレットを再ロードすることです。

サーブレット・プロパティーが構成済みで、画面ハンドラー bean を使用している場合、非活動の端末は存在しないはずです。

その他の構成オプションを検討する

検討する必要のある他の構成オプションは、以下のとおりです。

- **セキュリティー**

  端末サーブレットへのアクセスを制御しなければならないことがあります。

- **ログ記録**

  端末サーブレットは、ログ情報を標準サーブレット・ログに書き込みます。ログ記録をオンまたはオフに設定できます。

- **セッション追跡**

  セッション追跡を使用するように端末サーブレットを構成してください。

- **サーバー側インクルード**

  サーバー側インクルードを処理するように Web サーバーを構成することができます。

注: IBM WebSphere Application Server、バージョン 1.01 は、サーブレットによる DBCS データの入力または出力を完全にはサポートしていません。他のサーブレット・エンジンにも同様の制約がある可能性があります。この場合、端末サーブレットは DBCS 文字を ???? (疑問符) として表示します。

端末サーブレットのロード

端末サーブレットを構成して設定値を保管した後、それをロードしなければなりません。それをすぐにロードするか、または Web サーバーが開始するたびにロードするかを選択できます。

端末サーブレットを適切に構成していない場合、Web サーバーが端末サーブレットをロードしようとするとエラー・メッセージが表示されます。詳細については、「*CICS Transaction Gateway: Gateway Messages*」の端末サーブレットに関する項を参照してください。

端末サーブレットの使用

この節では、端末サーブレットの使用方法について説明します。ここでは、端末サーブレットを起動するためのいろいろな方法について説明し、また端末サーブレットを使用する Web ページの開発方法についても説明します。

サンプル・ファイルは、サブディレクトリー /samples/terminalservlet にあります。これらのサンプル・ファイルは、ファイル /samples/Samples.txt で説明されています。

CICS に接続してトランザクションを開始する

端末サーブレットのインスタンスは、CICS サーバーだけに接続できます。アクセスを許可する CICS サーバーごとに、サーブレットのインスタンスを作成することになります。さらに、初期化パラメーターの異なるサーブレットの複数のインスタンスも作成することができます。

CICS サーバー上でトランザクションを開始するには、適切な要求を使用して端末サーブレットを起動しなければなりません。

端末サーブレットの起動

端末サーブレットを起動するには、以下の 3 つの方法があります。

- URL による
- HTML フォームを使用する
- サーバー側インクルードを使用する

それぞれの場合に、使用するサーブレット・インスタンスの名前を知っている必要があります。そしてサーブレットに、実行するアクションを指示する 1 つまたは複数の要求パラメーターを渡します。パラメーターの名前および値は、すべての場合に同じですが、それらを指定する方法は異なっています。

CICS サーバー上でトランザクションを開始するには、以下のパラメーターを指定しなければなりません。

パラメーター名	パラメーター値
request	send
transaction	開始するトランザクションのトランザクション ID。 しかし ATI トランザクションは、端末サーブレットに関して開始できないことに注意してください。

必要であれば、サーブレットは端末をユーザーに割り当てます。

要求パラメーターの完全なセットは、115ページの『要求パラメーター』 にリストされています。

URL によってサーブレットを呼び出す

TerminalServlet と呼ばれるサーブレット・インスタンスを起動するには、次の URL にリンクします。

```
http://your_webserver/servlet/TerminalServlet
```

ここで、*your_webserver* はご使用の Web サーバーのホスト名です。

照会ストリングの形式で URL 内に要求パラメーターをエンコードできます。以下に例を示します。

```
http://your_webserver/servlet/TerminalServlet?request=send&transaction=CECI
```

ユーザーに情報を入力させる必要がない場合は、サーブレットを起動するためのこの方法を使用してください。

HTML フォームを使用してサーブレットを起動する

この方式では、Web ページ内に HTML フォームを作成します。以下に、例を示します。

```
<FORM METHOD="GET" ACTION="/servlet/TerminalServlet">
(テキスト入力領域、ボタン、およびその他のプロンプトをここに配置します)
</FORM>
```

これは、TerminalServlet と呼ばれるサーブレットを起動するためのフォームです。 METHOD は、GET または POST です。フォームの種々のエレメントには名前と値があり、これらはフォームが実行依頼されるときにサーブレットに送信されます。特定の要求パラメーターを設定することが分かっていれば、それを隠し項目としてフォームに追加します。以下に例を示します。 <input type=″hidden″ name=″request″ value=″send″> 隠し入力は、常にサーブレットに送られます。一般的に、フォームエレメントの名前は要求パラメーターの名前と同じで、その値は要求パラメーターの値と同じです。

これはユーザーが入力する情報をサーブレットに送信する唯一の方法です。

付属のサンプル epissam3.html は、この HTML フォーム方式を示しています。製品 CD の Samples.txt ファイルを参照してください。

サーバー側インクルードを使用してサーブレットを起動する

サーバー側インクルードは、 Web ページがユーザーに送信される前に Web サーバーによって処理されます。サーブレットが起動されて、生成される出力は Web ページ内のサーバー側インクルードのタグ位置に組み込まれます。ユーザーに表示されるのは、Web ページ内のサーブレット出力だけです。

HTML ソースでは、サーブレットのサーバー側インクルードは次のようになります。

```
<SERVLET NAME="TerminalServlet" >
<PARAM NAME="request" VALUE="send">
<PARAM NAME="transaction" VALUE="CECI">
<PARAM NAME="display" VALUE="none">
</SERVLET>
```

サーバー側インクルードを使用して、ユーザーの介入なしに、 Web ページに可変情報を追加したり、 Web ページが表示されるたびに端末を駆動してアクションを実行することができます。

サーバー側インクルードをいくらでも Web ページに追加できますが、**以下について注意してください。**

- 1 つのページ内にある複数のサーバー側インクルードは、並列に処理することができます。したがって、それらが実行される順序を確定することはできませんが、通常その順序はソースに記述された順序と同じになります。
- サーブレットは、ユーザーに割り当てられた端末にアクセスすることを可能にするセッション情報を格納します。セッションがサーバー側インクルードによって起動される場合、同じページにある後続のサーバー側インクルードはセッション情報にアクセスできません。ページがユーザーに送られると、セッションが確立され、複数のサーバー側インクルードがあるページは正常に機能するようになります。

付属のサンプル `epissam2.shtml` は、このサーバー側インクルード方式を示しています。製品 CD の Samples.txt ファイルを参照してください。

次に生じる事柄

URL にリンクすることによってフォームからサーブレットが起動されたとき、何らかの出力をブラウザーに送り返す必要があります。ユーザーに何を表示するかを判別するため、サーブレットは設定されたページ・マッピング・プロパティーを使用します。 101ページの『ページ・マッピング』 を参照してください。

画面およびフィールドの表示

CICS 画面情報を Web ページに含めるには、次の 2 つの方法があります。

- サーバー側インクルード
- *変数置換*

これら 2 つの方法を組み合わせることも可能です。

サーバー側インクルードの使用

サーバー側インクルードを処理するには、ご使用の Web サーバーを構成する必要があります。サーブレットを起動するサーバー側インクルードを含む SHTML ページを作成します。サーブレットからの出力は、そのページのサーバー側インクルードの位置に挿入されます。

たとえば、画面を (HTML フォームの一部として) 表示するには、以下のように行います。

```
<SERVLET NAME="TerminalServlet" >
<PARAM NAME="display" VALUE="screen">
</SERVLET>
```

または、画面の 22 番目のフィールドの内容を表示するには、以下のように行います。

```
<SERVLET NAME="TerminalServlet" >
<PARAM NAME="display" VALUE="22">
</SERVLET>
```

変数置換の使用

サーブレットがロードする HTML テンプレート・ファイルを作成できます。サーブレットは、ファイル内の変数を CICS 画面情報で置き換え、その出力をブラウザーに送ります。マッピング・プロパティーを使用して、ロードするテンプレート・ファイルをサーブレットに指示します (101ページの『ページ・マッピング』 を参照)。

たとえば、CESN サインオン画面のテンプレート・ファイルは以下のようになります。

```
<html>
<head>
<title>CICS Signon</title>
</head>
<body>
<h1>CICS Signon</h1>
<form method="POST" action="/servlet/TermServlet">
<input type="hidden" name="request" value="send">
<pre>
 Userid: <input type="text" name="USERID" value="&USERID;" size="8" >
```

```
 Password: <input type="text" name="PASSWORD" value="&PASSWORD;" size="8" >
</pre>
<input type="submit" name="DFH_ENTER" value="Sign on">
<input type="submit" name="DFH_PF1" value="Help">
<input type="submit" name="DFH_PF3" value="Exit">
</form>
</body></html>
```

ここで、変数は &USERID; および &PASSWORD; です。フィールドは番号で識別できます。この例のように、フィールドの名前を使用するには、画面を認識してフィールドを名前で設定できる画面ハンドラー bean が必要です。

ページ・マッピング・プロパティーの値は、HTML テンプレート・ファイルの名前、または任意の URL とすることができます。ページ・マッピング・プロパティーが HTML テンプレートを表していることをサーブレットが認識できるようにするためには、この値を file://*template_filename* に設定してください。たとえば、`file://d:/html/templates/test.html` とします。

何を表示できるか

サーバー側インクルードによって表示できる情報は、変数置換にも使用できます。その逆も可能です。テンプレートで使用される変数名は、サーバー側インクルードの **display** パラメーターの値と同じです。

通常、以下のものを表示できます。

- 画面 (HTML フォームの一部として)
- フィールド (番号によって識別されるか、または画面ハンドラー bean があれば名前によって識別される)
- サーブレット構成設定値
- 現在接続されていて、使用されていない端末の数

表示可能なプロパティーの完全なセットは、111ページの『プロパティーおよびパラメーターのリファレンス』 にリストされています。

画面を CICS に送り返す

画面を CICS に戻すには、**request** パラメーターを ″send″ に設定して、サーブレットを起動しなければなりません。 **transaction** パラメーターは、トランザクションがすでに実行中の場合には無視されるので、設定する必要はありません。画面を CICS に送り返す前に新しい情報によって更新するには、フィールド (名前または番号による) および設定値を識別する追加の要求パラメーターを指定します。

たとえば、以下のサーバー側インクルード:

```
<SERVLET NAME="TerminalServlet" >
<PARAM NAME="request" VALUE="send">
<PARAM NAME="USERID" VALUE="sysad">
<PARAM NAME="PASSWORD" VALUE="sysad">
<PARAM NAME="display" VALUE="none">
</SERVLET>
```

は、USERID および PASSWORD という名前のフィールドを指定された値に設定して、画面を CICS に送ります。 HTML フォームでは、フォーム・エレメント <input type=″text″ name=″USERID″ value=″″ size=″8″> を使用できます。ユーザーがフォームを実行依頼すると、要求パラメーター USERID は入力された値に設定されます。

AID の設定

デフォルトの AID は Enter です。

AID を HTML フォームから設定するには、以下のように指定した Submit ボタンをインクルードします。

```
<input type="submit" name="DFH_PF3" value="Exit" >.
```

ユーザーがボタンを押すと、サーブレットには要求パラメーター ″DFH_PF3″ に値 ″Exit″ が指定されて送信されます (サーブレットはそれを無視します)。AID は PF3 に設定されます。サーバー側インクルードでは、AID を以下のように設定できます。

```
<SERVLET NAME="TerminalServlet" >
<PARAM NAME="request" VALUE="send">
<PARAM NAME="DFH_PF3" VALUE="ignored">
<PARAM NAME="display" VALUE="none">
</SERVLET>
```

要求パラメーターの完全なセットは、111ページの『プロパティーおよびパラメーターのリファレンス』 にリストされています。

切断

サーブレットが端末をユーザーに割り当てると、以下のことが生じるまでその割り当ては継続します。

- パラメーター ″request″ を ″disconnect″ に設定してサーブレットが起動される。
- Web サーバー上のユーザーのセッションが満了する。
- **pool@idleTimeout** プロパティーが設定されている場合、端末が指定の期間アクセスされないと自動的に切断される。
- CICS クライアントが停止する。

- CICS サーバーへの接続が失われる - またはサーバーが停止する。
- サーブレットが Web サーバーによってアンロードされる。

端末が必要なくなったときは、それを明示的に解放してください。

あるユーザーがサーブレットを使用するセッションを確立していて、すでに端末を割り当てられている場合は、ユーザーがアクセスできるサーブレットのどのインスタンスによってもセッションを継続できることに注意してください。ユーザーが切断するか、または他の理由でセッションが終了するまで、同じ端末が使用されます。特定の要求に使用されるページ・マッピングおよび他のプロパティーは、要求を処理しているサーブレットのインスタンスに設定されたものです。

ページ・マッピングなど、要求パラメーターとして設定できるプロパティーはサーブレットの設定値をオーバーライドしますが、通常は現在の要求だけに適用され、他のユーザーによる要求または同じユーザーによる後続の要求には影響を与えません。

プロパティーおよびパラメーターのリファレンス

この節では、端末サーブレットの各プロパティーについて説明します。

サーブレット構成プロパティー

一般的に、これらのプロパティーは Web サーバーを構成するときに、プロパティー・ファイル内にまたは初期化パラメーターとして指定できます。名前には、大文字小文字の区別はありません。サーブレット初期化パラメーターとして指定した値は、プロパティー・ファイルからロードされた設定値をオーバーライドします。 **servlet@debug** および **servlet@extendedLogging** プロパティーを除けば、これらのプロパティーをサーブレットの実行中に変更することはできません。しかし、要求パラメーター **display** を使用して、プロパティーの現在の設定値を表示できます。 116ページの『表示可能プロパティー』 を参照してください。

servlet@coloring

Y に設定された場合は、色付きのフィールドが、3270 データ・ストリームで指示されている色で表示されます。

HTML の制限により、無保護フィールドを色づけすることはできません。このフィールドは常に、背景が白の黒字のテキストで表示されます。

このプロパティーが指定されていない場合は、端末サーブレットは、背景を白にし、黒字のテキストで画面を表示します。

servlet@debug
「true」に設定すると、サーブレットのすべてのインスタンスについてトレースをオンにします。サーブレットの実行中にトレースをオンまたはオフにすることが可能です。出力は、標準出力に送られます。トレースをオンにするとパフォーマンスに重大な影響が及ぶことに注意してください。

servlet@extendedLogging
「true」に設定すると、拡張ログ記録をオンにします。出力は、サーブレット・ログに送られます。

servlet@propertyFile
ロードするプロパティー・ファイルの絶対パス名。このプロパティーは、通常はサーブレット初期化パラメーターとしてだけ設定されます。

servlet@templateDir
HTML テンプレート・ファイルのロード元となるディレクトリーの名前。このプロパティーを設定すると、テンプレート・ディレクトリーまたはそのサブディレクトリーに存在しないファイルを表示することを防止できます。このプロパティーを設定しないと、サーブレットはアクセスできるどのファイルをも Web サーバーにロードする可能性があります。

pool@defaultHandler
特定のハンドラーが見つからない場合に終了のために使用される画面ハンドラー。これを画面ハンドラー・パスからロードした画面ハンドラーの 1 つに設定する場合、ハンドラーの前にハンドラー・パスを設定するようにしてください。

pool@deviceType
使用する端末モデル定義。このプロパティーを設定しない場合、デフォルトが使用されます。

pool@encoding
Java エンコード方式を使用するように指定することができます。この詳細については、「*CICS Transaction Gateway: Gateway プログラミング*」の『付録 A. Java エンコード方式』を参照してください。

pool@exitRetryLimit
端末切断の際にサーブレットが実行中のトランザクションの終了を試行する回数。これは画面を終了できない場合にループすることを防止します。

pool@gatewayUrl
サーブレットが接続する CICS Transaction Gateway の URL。デフォルトは、ローカルのゲートウェイつまり `local://` を使用することです。

pool@handlerPath
画面ハンドラーのロード元となるパス。ディレクトリー、.jar ファイル、および .zip ファイルを含めることができますが、サブディレクトリーは検索されません。

pool@idleTimeout
端末が自動的に切断される前にアイドル状態となっている最大時間 (秒単位)。デフォルト値は 0、つまりタイムアウトにならないということです。

pool@installTimeout
失敗するまでに端末がインストールできる最大時間 (秒単位)。有効な範囲は、0 ～ 3600 です。デフォルト値は 0、つまりタイムアウトにならないということです。

pool@maxTerminals
CICS に同時に接続できる端末の最大数。デフォルト値は 5 です。

pool@minTerminals
使用されていないときにも CICS に接続したままにする端末の数。デフォルト値は 0 です。

pool@password
作成時に、パスワードをサインオン機能を持たない端末に関連付けることを指定します。

pool@readTimeout
会話型トランザクションに許可された、タイムアウトになるまでの最大時間 (秒単位)。有効な範囲は、0 ～ 3600 です。この値が 0 の場合は、タイムアウトになりません。

pool@reusingTerminals
「true」に設定すると、解放された端末は次のユーザーのために接続されたままとなります。デフォルトの振る舞いは、ユーザー・セッションの終了時に端末を切断することです。このプロパティーを有効にするには、 **pool@minTerminals** も設定しなければなりません。

pool@serverName
接続先の CICS サーバーの名前。このプロパティーを設定しない場合、構成で定義したデフォルトの CICS サーバーが使用されます。

pool@signonCapability
拡張端末に対してサインオン機能を指定します。有効な項目は、EPI_SIGNON_CAPABLE または EPI_SIGNON_INCAPABLE です。

pool@pool@userid
作成時に、ユーザー ID をサインオン機能を持たない端末に関連付けることを指定します。

screen@cursoring
「true」に設定すると、JavaScript は画面カーソル位置を設定します。デフォルトの設定値は「false」です。

screen@NeutralColor
画面ハンドラーによるニュートラルな 3270 カラーの変換後の HTML カラーを指定します。

screen@stripBlanks
「true」に設定した場合、Web ブラウザーはフィールド・テキストをフォーマットします。

screen@stripEmptyRows
「true」に設定すると、画面ハンドラーは、データを含まない画面行を除去します。

screen@TrimFields
「true」に設定すると、画面ハンドラーは、すべての入力フィールドの最初から最後までの余分な空白を除去します。

ページ・マッピングのプロパティー

ページ・マッピングは、Web ページを特定の画面または端末状態に関連付けます。ページ・マッピングは、Web サーバーを構成する際に、プロパティー・ファイル内にまたはサーブレット初期化パラメーターとして指定できます。しかし、要求パラメーター **display** を使用して、ページ・マッピングの現在の設定値を表示できます。 116ページの『表示可能プロパティー』 を参照してください。

標準のページ・マッピング・プロパティー	値
page@disconnected	端末が切断されたときに表示するページ。
page@error	エラー発生時に表示するページ。
page@idle	端末がアイドル状態のとき、つまり実行中のトランザクションがないときに表示するページ。

標準のページ・マッピング・プロパティー	値
page@default	他のページが指定されていない場合のページ。
page@*screen*	この画面に表示するページ。

要求パラメーター

これらのパラメーターは要求が行われるときだけに設定されます。さらに、要求時に設定できる他のプロパティーも指定できます。例として、**servlet@extendedLogging** があります。

パラメーター名	パラメーター値
request	画面を CICS に送る場合は「send」、セッションを終了する場合には「disconnect」に設定します。 必要な場合、新しい端末セッションが開始します。
display	表示可能なプロパティーの名前。116ページの『表示可能プロパティー』 を参照。
translateText	文字 <、>、および & をそれぞれ <、>、および & に変換する場合、「true」に設定します。これはテキストを表示する場合に使用します。
transaction	開始するトランザクションのトランザクション ID。これは要求が「send」で端末状態がアイドルの場合だけに使用されます。 しかし ATI トランザクションは、端末サーブレットに関して開始できないことに注意してください。
transactionData	トランザクションに渡す追加のパラメーター。これは要求が「send」で端末状態がアイドルであり、トランザクションが指定されている場合だけに使用されます。
DFH_CURSOR	画面カーソルを位置付けるフィールド。フィールドの指標 (番号)、または現在の画面で画面ハンドラーが使用可能な場合はフィールドの名前で指定します。 このパラメーター名には大文字を使用します。
DFH_ENTER、 DFH_CLEAR、 DFH_PA1 - 3、 DFH_PF1 - 24	この値は無視されます。 AID キーを CICS に送ることを指定するために、これらのパラメーターのいずれかを使用します。これは大文字でなければなりません。

パラメーター名	パラメーター値
フィールドの指標	フィールドに設定するテキスト。 フィールドの指標を知っている場合、フィールドの内容を設定するために使用します。
usingSession	端末セッションが必要ない場合、「false」に設定します。 このパラメーターを設定すると、この要求に使用する端末は使用後に廃棄されます。
matchOnIdle	デフォルトでは「false」に設定されています。 通常、端末状態が「client」のとき、つまり会話型または疑似会話型トランザクションが応答を待っているとき、サーブレットは現在の画面の ScreenHandler を探すことだけを試行します。このプロパティーを「true」に設定すると、状態に関係なく画面が変更するときにはいつもサーブレットが現在の ScreenHandler を検査するようになります。 BMS マップを使用していても、会話型または疑似会話型でないトランザクションがある場合に上記の設定にすることができます。多数の ScreenHandler がある場合、このプロパティーがオンに設定されているとサーブレットの実行速度が遅くなることがあります。そのため、デフォルト値は「false」です。

加えて、現在の画面で画面ハンドラー bean が使用可能であれば、任意のプロパティーを要求内に指定することによりそれらを bean に設定できます。 BMSMapConvert によって生成される bean では、BMS マップで定義されるフィールド名はフィールド内の必須テキストに設定できるプロパティーです。

表示可能プロパティー

以下の表は、「display」要求パラメーターに指定できる値、または変数置換に使用できる値を示しています。 108ページの『変数置換の使用』 を参照してください。

display の値	表示
none	なし。サーバー側インクルードからの出力を抑止するために使用します。
screen	HTML フォームの一部としての画面。

display の値	表示
errorMessage	エラーが生じた場合、そのエラー・メッセージ。
errorCause	エラーが生じた場合、そのエラーに関する詳細情報。
DFH_CURSOR	カーソルのあるフィールドの、判別可能であれば名前、そうでなければフィールドの指標。
フィールドの指標	フィールド・テキスト
画面ハンドラー・プロパティー名	判別可能であれば、プロパティーの値。
上にリストされたサーブレット構成プロパティーのいずれか	現在のサーブレット・インスタンスのプロパティーの値。
任意のページ・マッピング・プロパティー	現在のサーブレット・インスタンスのプロパティーの値。
「request」および「display」を除く、任意の要求パラメーター	現在の要求のプロパティーの値。
pool@connectedTerminals	このサーブレット・インスタンスで、現在接続されている端末の数。
pool@freeTerminals	このサーブレット・インスタンスで、現在接続されていて使用されていない端末の数。
pool@terminalsInUse	このサーブレット・インスタンスで、現在ユーザーに割り当てられている端末の数。

CICS® Transaction Server for OS/390® の Web インターフェース

CICS® Transaction Server for OS/390® バージョン 1.2 およびそれ以上に含まれる CICS Web インターフェースは、 Web サーバーから CICS トランザクション処理サービスへの直接アクセスをサポートするトランザクションおよびプログラムの集合です。

CICS Web インターフェースで使用するための HTML テンプレートを生成した場合、それらを端末サーブレット用の HTML テンプレートとして使用することもできます。CICS Web インターフェースの代わりにサーブレットを起動するように、いくらかの変更を行う必要がありますが、フォーマットは基本的に同じです。

テンプレート・ファイルを使用するには、以下のようにします。

1. テンプレート・ファイルを Web サーバー上のディレクトリーにコピーします。
2. それを編集して、<FORM> アクションを /servlet/TerminalServlet に変更し、名前「request」を値「send」に設定した隠れ <FORM> エレメントを追加します。
3. HTML テンプレートが参照する BMS マップのための Map クラスおよび画面ハンドラー bean を生成します。
4. 生成した Java ソース・ファイルをコンパイルします。
5. コンパイルしたクラス・ファイルを Web サーバー上の適切なディレクトリーにコピーします。
6. サーブレット・プロパティー **pool@handlerPath** を設定して、それが新しいクラスをロードするようにします。
7. ページ・マッピング・プロパティーを設定して、新しい画面ハンドラー bean を HTML テンプレート・ファイルに関連付けます。

第8章 問題判別と問題解決

問題を調査中に、その問題を解決するために十分な情報が見つかる場合がありますが、問題判別を問題解決と混同しないでください。発生する可能性がある問題のタイプの例を以下に挙げます。

- エンド・ユーザー・エラー
- プログラミング・エラー
- 構成エラー

事前チェック

問題の調査に取りかかる前に、明白な原因があるかどうか調べてください。

- システムは以前は正しく実行していましたか。
- システムの構成に変更を加えたり、新しい機能やプログラムを追加したりしましたか。
- CLASSPATH パスが正しく設定されていますか

  CLASSPATH の設定については、31ページの『CICS Transaction Gateway のプログラミング環境の構成』を参照してください。

  CLASSPATH 変数は、CICS Transaction Gateway のトレース出力の先頭に表示されます。トレースを使用可能にして Gateway を始動する方法については、91ページの『ユーザー指定オプションでの Gateway の開始』を参照してください。
- CICS ユニバーサル・クライアントの環境が正しく設定されていますか。

  次のコマンドを使用してください。

  ```
  cicscli -s=servername
  ```

  ここで、*servername* は CICS サーバーの名前です。この後に

  ```
  cicscli -l
  ```

  を続けて入力し、サーバーに接続可能かどうかを確認します。
- 障害の発生を表すメッセージが発行されていますか。
- 障害を再現できますか。

次に行うこと

問題が CICS Transaction Gateway にあると思う場合は、できる限り多くの情報を収集して、サポート組織に連絡する必要があります。

トレースを使用可能にせずに Gateway を始動した場合は、Gateway を停止してからトレース・オプションをいずれか 1 つ指定して再始動し、問題を再現します。

問題が CICS ユニバーサル・クライアントまたは別の製品にあると思われる場合は、その製品の問題判別手順に従ってください。 CICS ユニバーサル・クライアントの問題判別については、「*CICS Transaction Gateway: クライアント管理の手引き*」を参照してください。 CICS サーバーについては、該当するサーバーの「*問題判別*」を参照してください。

メッセージ

CICS Transaction Gateway メッセージには、従来は CICS クライアントに使用されていた、*CCL* という接頭部が付いています。

CICS Transaction Gateway が生成するメッセージのリストについては、「*CICS Transaction Gateway: Gateway Messages*」をご覧ください。

トレース

Gateway およびアプリケーションのトレースには、次の 3 つの主要レベルがあります。

スタック・トレース
Java の例外が発生した場合に限りトレース項目が書き込まれます。トレース項目を使用すると、例外の発生源を突き止めるのに役立ちます。パフォーマンスを維持することが重要な場合には、これを使用してください。122ページの図9 を参照してください。

標準トレース
Java の例外および Gateway のメイン機能とイベントがトレースされます。デフォルトでは、Gateway は、データ・ブロック (たとえば COMMAREA あるいはネットワーク・フロー) の最初の 128 バイトだけをトレースに表示します。

デバッグ・トレース
スタック・トレースあるいは標準トレースよりも詳細に Java の例外および Gateway のメイ

ン機能とイベントがトレースされます。デフォルトでは、Gateway は、データ・ブロック全体をトレースに表示します。これは、パフォーマンスが重要ではない場合、あるいは標準トレースでは問題解決のための情報が十分に得られない場合にのみ使用してください。

Gateway のトレース

トレースは、CICS Transaction Gateway プロセスまたは Gateway に呼び出しを行うユーザー・アプリケーションで開始することができます。

Gateway でトレースを使用可能にし、それを制御するには、**ctgstart** コマンドでオプションを使用します。たとえば、標準トレースを使用可能にするには、Gateway の始動時に次のコマンドを入力します。

```
ctgstart -trace
```

Gateway の構成に構成ツールを使用した場合は、トレース出力のデフォルト宛先は、CICS Transaction Gateway がインストールされている `bin` サブディレクトリーのファイル `CTG.TRC` です。構成ツールを使用しなかった場合は、トレース出力は `stderr` に書き込まれます。トレース出力のデフォルト宛先は、**ctgstart-tfile** オプションを使用すると変更することができます。たとえば

```
ctgstart -x -tfile pathname
```

というコマンドを入力すると、Gateway の始動時にデバッグ・トレースが使用可能になり、トレース出力は、*pathname* で指定されたファイルに書き込まれます。

パフォーマンスの考慮事項: トレース、特にデバッグ・トレースを行うと、パフォーマンスが低下します。CICS Transaction Gateway バージョン 4.0 では、パフォーマンスに与えるトレースの影響を小さくするために、以下の変更が行われました。

- **-tfile** オプションを使用してトレース出力ファイルを指定した場合は、トレース出力は `stderr` に書き込まれません。
- トレース出力内のデータ・ブロックの最初の行だけにタイム・スタンプが押されます。
- **-truncationsize** および **-dumpoffset** オプションを使用すると、トレースに書き込まれるデータ・ブロックのサイズを制限することができます。
- **-tfilesize** オプションを使用すると、トレース出力ファイルのサイズを制限することができます。

次に行うこと

トレース・オプションを指定して CICS Transaction Gateway を始動する方法の詳細については、91ページの『Gateway の開始』を参照してください。

```
10:38:12:262 : main: S-C: java.io.IOException
10:38:12:263 : main: S-C:  at com.ibm.gskssl.SSLSocketImpl.socketCreate(Native Method)
10:38:12:264 : main: S-C:  at com.ibm.gskssl.SSLSocketImpl.create(SSLSocketImpl.java:133)
10:38:12:265 : main: S-C:  at com.ibm.gskssl.SSLServerSocket.<init>(SSLServerSocket.java:109)
10:38:12:265 : main: S-C:  at com.ibm.gskssl.SSLServerSocket.<init>(SSLServerSocket.java:73)
10:38:12:266 : main: S-C:  at com.ibm.ctg.server.GskSslHandler.initialize(GskSslHandler.java:487)
10:38:12:266 : main: S-C:  at com.ibm.ctg.server.JGate.main(JGate.java:941)
```

図 9. Java スタック・トレースのサンプル

アプリケーションのトレース

アプリケーションでトレースを使用可能にするには、2 つの方法があります。JVM の始動時に Java のディレクティブを使用する、または CICS Transaction Gateway のトレース API に呼び出しを追加する、のいずれかの方法です。Java のディレクティブを指定する場合は、**java** コマンドで **-D** を使用します。トレース API は、CICS Transaction Gateway の T クラスのいくつかの静的メソッドで構成されています。詳細については、「*CICS Transaction Gateway: Gateway プログラミング*」の『Java クライアント・プログラムにおけるトレース』を参照してください。

JNI のトレース

Gateway のトレース・オプションでは、JNI のトレースを使用可能にすることはできません。これを使用可能にするには、環境変数 CTG_JNI_TRACE が、出力を書き込むファイルを示すように設定します。

WebSphere におけるサーブレットのトレースおよびロギング

IBM WebSphere Application Server V3.5 は*拡張管理コンソール* を備えており、これによりアプリケーションの管理に関連するタスクが容易になります。

WebSphere でサーブレットのトレースを使用可能にするには、以下を実行してください。

1. 標準出力および標準エラーを、選択したファイルに送信するよう WebSphere を構成する。

   管理コンソールで、トレースを使用可能にするアプリケーション・サーバーを選択します。「**一般 (General)**」タブには、「**標準出力 (Standard output)**」および「**標準エラー (Standard error)**」というラベルのついたテ

キスト入力ボックスがあります。これらのボックスにファイル名を入力すると、デフォルトの宛先をオーバーライドすることができます。

2. アプリケーションにおけるトレースを使用可能にする。
   - CICS Transaction Gateway 端末サーブレットの場合は、**servlet@debug** プロパティーを使用してトレースを使用可能にしてください。

     詳しくは、111ページの『プロパティーおよびパラメーターのリファレンス』 をご覧ください。
   - ユーザーが作成したアプリケーションの場合は、Java ディレクティブを使用してトレースを使用可能にしてください。これらは、管理コンソールの「**コマンド行引き数 (Command line arguments)**」ボックスに入力することができます。Java ディレクティブには、**java** コマンド行呼び出しの場合と同じフォーマットを使用してください。たとえば、完全デバッグ・トレースを使用可能にするには、最初のコマンド行引き数として、`-Dgateway.T=on` を入力します。

     トレースのための Java ディレクティブの詳細については、「*CICS Transaction Gateway: Gateway プログラミング*」の『Java クライアント・プログラムにおけるトレース』を参照してください。
3. 「**適用 (Apply)**」ボタンをクリックして WebSphere の設定値を更新し、サーブレットを実行します。

WebSphere は、サーブレットの実行時に、ロギング情報を標準出力と同じ宛先に書き込みます。この情報には、開始時刻、停止時刻、およびエラーがあればエラー・メッセージが含まれます。CICS Transaction Gateway 端末サーブレットの場合は、 **servlet@extendedLogging** プロパティーを使用して拡張ロギングを使用可能にしてください。拡張ロギングを使用可能にすると、端末サーブレットは、要求を受け取るごとにログに書き込みます。このログは、問題の診断に役立つ情報を提供しますが、パフォーマンスを低下させます。

端末サーブレットの実行中は、両方のトレースおよび拡張ログ記録を使用可能にすることができます。端末サーブレットのプロパティーの詳細については、111ページの『プロパティーおよびパラメーターのリファレンス』を参照してください。

プログラム・サポート

IBM サポートへの連絡方法の詳細については、「*CICS Transaction Gateway: クライアント管理の手引き*」の『プログラム・サポート』を参照してください。

共通問題の診断

インストールの問題

インストール・プロセスでは *uudecode* 関数が使用されます。この関数は、*UNIX 間コピー・プログラム* のパーツです。CICS Transaction Gateway のインストール時に

```
ERROR: uudecode cannot be found in your PATH,
please refer to the Gateway Administration book
Problem Determination section for further information
```

というメッセージを受け取った場合は、システムにこのプログラムが存在していない可能性があります。 CICS Transaction Gateway をインストールする前に、システムにこのプログラムが存在し、PATH が正しく構成されていることを確認してください。

Java の例外

アプリケーションまたは CICS Transaction Gateway が例外を処理しない場合、Java 仮想マシン (JVM) が Java のスタック・ダンプを書き込みます。ダンプ出力の宛先は、使用している JVM のインプリメンテーションによって異なります。詳細については、ご使用の Java の資料を参照してください。

Java のスタック・ダンプに書き込まれる情報を増やすには、ジャストインタイム (JIT) コンパイラーを使用不可にしてください。この情報には、例外が発生した箇所の Java ソース・コード行が含まれていることがあります。 JIT コンパイラーを使用不可にする方法は、使用している JVM のインプリメンテーションによって異なります。詳細については、ご使用の Java の資料を参照してください。

図10 は、JIT コンパイラーを使用不可にして取られた Java のスタック・ダンプのサンプルを示しています。

```
Exception in thread "main" java.lang.OutOfMemoryError
        at java.lang.Thread.start(Native Method)
        at com.ibm.ctg.server.ThreadManager.createObject(ThreadManager.java:345)
        at com.ibm.ctg.server.ThreadManager.<init>(ThreadManager.java:131)
        at com.ibm.ctg.server.ManagedResources.<init>(ManagedResources.java:106)
        at  com.ibm.ctg.server.JGate.main(JGate.java:895)
```

図 10. Java のスタック・ダンプのサンプル

CICS Transaction Gateway が例外を処理すると、トレースが使用可能になっている場合にのみ Java のスタック・ダンプが書き込まれます。トレースを使用

可能にして、問題の再現を試行してください。これは、例外が発生する前に何が起こっていたのかを明らかにするために役立ちます。トレースの詳細については、120ページの『トレース』を参照してください。

Gateway が Windows NT® サービスとして実行されている場合の問題

Gateway が Windows NT® サービスとして実行されていて、Gateway が始動しない、あるいはクライアントが Gateway に接続できない場合は、コンソールからの入力が使用可能になっている可能性があります。項目が、 図11 のような場合は、Gateway がコンソールから読み取りを行おうとしていることを示しています。

```
11:37:58:472 : main: S-C: -> [ServerMessages: getKeys] ()
11:37:58:472 : main: S-C: <- [getKeys] = Q-
11:37:58:482 : main: S-C: java.io.IOException: The handle is invalid
11:37:58:482 : main: S-C:  at java.io.FileInputStream.readBytes(Native Method)
11:37:58:482 : main: S-C:  at java.io.FileInputStream.read(FileInputStream.java:190)
11:37:58:482 : main: S-C:  at java.io.BufferedInputStream.read1(BufferedInputStream.java:229)
11:37:58:482 : main: S-C:  at java.io.BufferedInputStream.read(BufferedInputStream.java:286)
11:37:58:482 : main: S-C:  at java.io.FilterInputStream.read(FilterInputStream.java:99)
11:37:58:482 : main: S-C:  at java.io.InputStreamReader.fill(InputStreamReader.java:180)
11:37:58:482 : main: S-C:  at java.io.InputStreamReader.read(InputStreamReader.java:256)
11:37:58:482 : main: S-C:  at java.io.BufferedReader.fill(BufferedReader.java:145)
11:37:58:482 : main: S-C:  at java.io.BufferedReader.readLine(BufferedReader.java(Compiled Code))
11:37:58:482 : main: S-C:  at java.io.BufferedReader.readLine(BufferedReader.java:368)
11:37:58:482 : main: S-C:  at com.ibm.ctg.server.JGate.main(JGate.java:940)
11:37:58:482 : main: S-C: CCL6559E: Unexpected exception occurred. [java.io.IOException: The handle is invalid]
```

図 11. Windows NT® サービスとして実行されている Gateway からのトレース

コンソールからの入力の読み取りを使用不可にするには、以下のいずれかを実行します。

- 構成ツールの「Gateway の一般設定 (General Gateway settings)」パネルの、「コンソールからの入力の読み取りを使用可能にする (Enable reading input from console)」ボックスのチェックをはずす。 32ページの『構成ツールの使用』を参照してください。
- **ctgservice** コマンドで **-A** オプションを使用して、 **-noinput** オプションを **ctgstart** コマンドに渡す。

java.lang.OutOfMemory 例外

長時間に渡って CICS Transaction Gateway を実行した後でこの例外が発生した場合は、メモリー・リークが発生している可能性があります。 Gateway の負荷が重い場合にのみそれが発生する場合は、使用可能な Java 仮想マシン (JVM) のメモリーに対して、アクティブなスレッドが多すぎる可能性があります。

メモリー・リーク

Gateway プロセスに対して、メモリー使用量をモニターしてください。メモリー・リークが発生している場合は、Gateway が使用するメモリーの量が時間と

共に増加します。CICS Transaction Gateway に接続しているすべてのユーザー・アプリケーションが、メモリーの使用後は JavaGateway() 接続オブジェクトをクローズしていることを確認してください。それでも問題が解決しない場合は、Gateway のデバッグ・トレースを実行し、IBM サポートに連絡してください。 **ctgstart -tfilesize** オプションを使用すると、トレース出力ファイルのサイズを制限することができます。また、**ctgstart -truncationsize** オプションを使用すると、トレースに表示されるデータ・ブロックのサイズをゼロにすることができます。これにより、パフォーマンスに与える影響が小さくなります。

多すぎるスレッド

JVM が必要とする仮想メモリーの量は、以下に示す数を使用して見積もることができます。以下の数を加算してください。

- 接続マネージャーのスレッドの数
- Worker スレッドの数
- Java が使用中のバックグラウンド・スレッドの数
- 使用中の異なるネットワーク・プロトコルの数

次に、JVM によってスレッドごとに割り振られている Java スタックのサイズをこの合計にかけてください。

この計算では、Java のアクティブなバックグラウンド・スレッドの数を約 5 と想定しています。より正確な値については、Java の資料を参照してください。使用している CICS Transaction Gateway の構成に、接続マネージャーおよび Worker スレッドの最大数を設定してください。詳細については、 36ページの『一般 Gateway 設定』を参照してください

Java がメモリーをどのように割り振るかは、ご使用の Java のインプリメンテーションによって異なります。ほとんどの JVM では、Java のスタックおよび Java のヒープのサイズに制限を設けることにより、メモリーの割り振りを制御することができます。

スレッドの制限の詳細については、13ページの表1 を参照してください。Java のメモリー割り振りについては、「*CICS Transaction Gateway: Gateway プログラミング*」およびご使用の Java の資料を参照してください。

OS/390® および UNIX® システムでは、各プロセスで使用可能なメモリーの量は制限されていることがあります。詳細については、ご使用のオペレーティング・システムの資料を参照してください。

プロセスがロックする場合

すべてのオペレーティング・システム

ロックが発生している箇所を突き止めるには、アプリケーション、Gateway、およびクライアントのトレースを使用可能にします。Gateway およびアプリケーションのトレースについては、120ページの『トレース』を参照してください。クライアントのトレース、およびロックの発生箇所がクライアントにあるのかサーバーにあるのかを突き止める方法については、「*CICS Transaction Gateway: クライアント管理の手引き*」を参照してください。ロックが Gateway で発生している場合は、IBM サポートに連絡し、Gateway のトレースを提供してください。

AIX

Java 仮想マシン (JVM) によっては、現行スレッドの状態を示すスタック・ダンプを Java に強制的に書き込ませることができます。たとえば、IBM Java SDK 1.3 では、**SIGQUIT (-3)** シグナルを Java プロセスに送信すると、その Java プロセスはスタック・ダンプを `stderr` に書き込むことができます。このスタック・ダンプは、現行スレッドの状態を示しています。これは作業中の実動システムには行わないようにしてください。完全にロックされてしまったシステムにのみ実行してください。

コード・ページの問題

Gateway のトレースには、JVM が実行されているコード・ページが表示されます。コード・ページは、トレースの最上部の「システム・プロパティー (System Properties)」セクションに表示されます。クライアントがサーバーに送信したコード・ページを見つける方法については、「*CICS Transaction Gateway: クライアント管理の手引き*」を参照してください。

SSL 例外

一般

CICS Transaction Gateway でスタック・トレースを使用可能にしてください。これで、例外が発生したときに何が起こっていたのかが明らかになります。また、CLASSPATH 変数の値などの、構成に関する情報も得ることができます。これで問題の診断に十分な情報が得られない場合は、標準トレースを実行し、IBM サポートに連絡してください。

CICS Transaction Gateway の開始時

Gateway の開始時に、次のメッセージが表示されます。

```
CCL6525W: Unable to start handler for the ssl: protocol.
            [java.lang.ClassNotFoundException:
              server KeyRing class name]
```

このメッセージは、Gateway がサーバーの KeyRing クラスをロードできなかったことを示します。スタック・トレースを使用可能にして、Gatewayを始動してください。CLASSPATH 変数がトレース出力の先頭に表示されるので、そこにこのクラスを含むディレクトリーの項目が含まれているか確認してください。また、CLASSPATH に cfwk.zip ファイルの項目が含まれていることも確認してください。Gateway は、このファイルを見つけることができなければ、KeyRing ファイルをロードすることができません。

サーバーの KeyRing クラスの名前は、構成ツールの SSL プロトコルの設定パネルの「**KeyRing のクラス名 (KeyRing classname)**」フィールドで指定します。KeyRing クラスの名前を指定する場合は、ディレクトリーまたはファイル拡張子は含めないでください。CLASSPATH には、KeyRing クラスを含むディレクトリーの項目が含まれている必要があります。

ユーザー・アプリケーションにおいて

API 呼び出し `SslJavaGateway.setKeyRing("ClientKeyRingName", "thePassword");` がユーザー・アプリケーションで失敗すると、次のエラーが発生します。

```
java.lang.NoClassDefFoundError: com/ibm/sslight/SSLightKeyRing
```

CLASSPATH 変数に、ファイル cfwk.zip の項目が含まれていることを確認してください。このファイルは、CICS Transaction Gateway に付属しています。

JDK™ AppletViewer の問題

AppletViewer を使用してローカル・ファイル・システムから CICS Transaction Gateway サンプル・アプレットを実行すると、

```
CCL6664E Unable to load relevant class to support the protocol protocol
```

というメッセージが表示されることがあります。

この問題を解決するには、「AppletViewer」メニューから**「アプレット (Applet)」**を選択してから、**「プロパティー (Properties)」**を選択して、**「ネットワーク・アクセス (Network Access)」**と**「クラス・アクセス (Class Access)」**の両方を 「無限 (Unrestricted)」に設定します。

付録A. CICS ユニバーサル・クライアントのデータ変換

ECI および EPI では、クライアント・システムで実行している CICS 以外のアプリケーションは、 CICS サーバー・システムが管理している CICS の機能およびデータにアクセスすることができます。

クライアントおよびサーバー間で文字データを渡す場合には、そのデータを変換する必要が生じる場合があります。たとえば、データが、CICS ユニバーサル・クライアント・システムでは ASCII で、CICS/390 サーバー・システムでは EBCDIC でエンコードされている場合です。データ変換は、サーバー・システムによって実行されます。

可能なエンコード・スキームである ASCII および EBCDIC については、「Character Data Representation Architecture Reference and Registry (CRDA)」(SC09-2190) で詳細に説明されています。その内容を要約すると、各エンコード・スキームは、コード化文字セット ID で識別され、コード化文字セット ID は、図形文字のセット、および図形文字を表すために使用されるコード・ポイントを指定する CPGID (Code Page Global Identifier) を定義しています。

サーバー・システムで管理されるデータには、それぞれが異なる ASCII エンコード・スキームを使用するいくつかのクライアント・システムからアクセスすることができます。そのようなアクセスをサポートするには、各クライアント・システムは、データが正しく変換されるように、CCSID「タグ」を提供できなければなりません。

サポートされている変換

データ変換に使用される方式は、サーバーのプラットフォームによって異なります。サポートされているデータ変換の範囲もプラットフォームによって異なります。以下に示す表は、「*CICS ファミリー システム/390 CICS からの通信*」(SD88-7242-00) からの抜粋であり、ASCII および EBCDIC の CCSID が地理的または言語学的グループに割り当てられています。

ASCII および EBCDIC の間のデータ変換がサポートされており、両方の CCSID は同じグループに属します。その意図するところは、他の CICS サーバーが等価なサポートを提供することにあります。

アラビア語:

	CCSID	CPGID	
ASCII	00864	00864	PC data: Arabic
	01089	01089	ISO 8859-6: Arabic
	01256	01256	MS Windows: Arabic
EBCDIC	00420	00420	Host: Arabic

バルト海沿岸:

	CCSID	CPGID	
ASCII	00921	00921	PC data: Latvia, Lithuania
	01257	01257	MS Windows: Baltic Rim
EBCDIC	01112	01112	Host: Latvia, Lithuania

キリル文字:

	CCSID	CPGID	
ASCII	00866	00866	PC data: Cyrillic
	00915	00915	ISO 8859-5: Cyrillic
	01251	01251	MS Windows: Cyrillic
EBCDIC	01025	01025	Host: Cyrillic multilingual

エストニア:

	CCSID	CPGID	
ASCII	00922	00922	PC data: Estonia
	01257	01257	MS Windows: Baltic Rim
EBCDIC	01122	01122	Host: Estonia

ギリシャ語:

	CCSID	CPGID	
ASCII	00869	00869	PC data: Greece
	00813	00813	ISO 8859-7: Grek
	01253	01253	MS Windows: Greek
EBCDIC	00875	00875	Host: Grek

ヘブライ語:

	CCSID	CPGID	
ASCII	00856	00856	PC data: Hebrew
	00916	00916	ISO 8859-8: Hebrew
	01255	01255	MS Windows: Hebrew
EBCDIC	00424	00424	Host: Hebrew

Latin-1:

	CCSID	CPGID	
ASCII	00437	00437	PC data: USA, many other countries
	00819	00819	ISO 8859-1: Latin-1 countries
	00850	00850	PC data: Latin-1 countries
	01252	01252	MS Windows: Latin-1 countries
EBCDIC	00037	00037	Host: USA, Canada, etc
	00273	00273	Host: Austria, Germany

	00277	00277	Host: Denmark, Norway
	00278	00278	Host: Finland, Sweden
	00280	00280	Host: Italy
	00284	00284	Host: Spain, Latin America (Spanish)
	00285	00285	Host: United Kingdom
	00297	00297	Host: France
	00500	00500	Host: International Latin-1
	00871	00871	Host: Iceland
	01047	01047	Host: Latin-1 Open Systems

ユーロ通貨記号サポートを含む Latin-1

	CCSID	CPGID	
ASCII	00858	00858	PC data: Latin-1 countries
	05348	01252	MS Windows: Latin-1 countries, version 2
EBCDIC	01140	01140	Host: USA, Canada, etc
	01141	01141	Host: Austria, Germany
	01142	01142	Host: Denmark, Norway
	01143	01143	Host: Finland, Sweden
	01144	01144	Host: Italy
	01145	01145	Host: Spain, Latin America (Spanish)
	01146	01146	Host: United Kingdom
	01147	01147	Host: France
	01148	01148	Host: International Latin-1
	01149	01149	Host: Iceland

Latin-2:

	CCSID	CPGID	
ASCII	00852	00852	PC data: Latin-2 multilingual
	00912	00912	ISO 8859-2: Latin-2 multilingual
	01250	01250	MS windows: Latin-2
EBCDIC	00870	00870	Host: Latin-2 multilingual

Latin-5:

	CCSID	CPGID	
ASCII	00857	00857	PC data: Latin-5 (Turkey)
	00920	00920	ISO 8859-9: Latin-5 (Turkey)
	01254	01254	MS Windows: Turkey
EBCDIC	01026	01026	Host: Latin-5 (Turkey)

Latin-9:

	CCSID	CPGID	
ASCII	00923	00923	ISO 8859-15: Latin-9
EBCDIC	00924	00924	Host: Latin-9

日本語:

	CCSID	CPGID	
ASCII	00932	00897	PC data: SBCS
		00301	PC data: DBCS
	00942	01041	PC data: extended SBCS
		00301	PC data: DBCS

	00943	00897	PC data: SBCS
		00941	PC data: DBCS for Open environment
	00954	00895	G0: JIS X201 Roman
		00952	G1: JIS X208-1990
		00896	G2: JIS X201 Katakana
		00953	G3: JIS X212
EBCDIC	00930	00290	Host: Katakana, extended SBCS
		00300	Host: DBCS
	00931	00037	Host: Latin-1, SBCS
		00300	Host:DBCS
	00939	01027	Host: Latin-1, extended SBCS
		00300	Host: DBCS

韓国:

	CCSID	CPGID	
ASCII	00934	00891	PC data: SBCS
		00926	PC data: DBCS
	00944	01040	PC data: extended SBCS
		00926	PC data: DBCS
	00949	01088	PC data: SBCS, IBM KS code
		00951	PC data: DBCS, IBM KS code
	00970	00367	G0: ASCII
		00971	G1: KSC X5601-1989
	01363	01126	MS Windows: Korean SBCS
		01362	MS Windows: Korean DBCS
EBCDIC	00933	00833	Host: extended SBCS
		00834	Host: DBCS

中国語 (簡体字):

	CCSID	CPGID	
ASCII	00946	01042	PC data: extended SBCS
		00928	PC data: DBCS
	01381	01115	PC data: extended SBCS, IBM GB
		01380	PC data: DBCS, IBMGB
	01383	00367	G0: ASCII
		01382	G1: GB 2312-80 set
	01386	01114	PC data: SBCS, S-Chinese GBK, T-Chinese IBM BIG-5
		01385	PC data: DBCS, S-Chinese GBK
EBCDIC	00935	00836	Host: extended SBCS
		00837	Host: DBCS

中国語 (繁体字):

	CCSID	CPGID	
ASCII	00938	00904	PC data: SBCS
		00927	PC data: DBCS
	00948	01043	PC data: extended SBCS
		00927	PC data: DBCS
	00950	01114	PC data: SBCS, IBM BIG-5
		00947	PC data: DBCS
	00964	00367	G0: ASCII
		00960	G1: CNS 11643 plane 1

		00961	G2: CNS 11643 plane 2
EBCDIC	00937	00037	Host: Latin-1 SBCS
		00835	Host: DBCS

付録B. CICS Transaction Gateway および 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および関連資料をすべてリストし、それらの各種形式を紹介します。

この章の見出しは、以下のとおりです。

CICS Transaction Gateway のマニュアル

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

  このマニュアルは、CICS® Transaction Gateway for Windows® の管理について説明します。

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

  このマニュアルは、CICS® Transaction Gateway for AIX® の管理について説明します。

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

  このマニュアルは、CICS® Transaction Gateway for Solaris の管理について説明します。

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

  このマニュアルは、CICS® Transaction Gateway for Linux の管理について説明します。

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

  このマニュアルは、CICS® Transaction Gateway for HP-UX の管理について説明します。

CICS Transaction Gateway および 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*

  このマニュアルは、CICS® Transaction Gateway for OS/390® の管理について説明します。

- *CICS Transaction Gateway: Gateway Messages*

  このオンライン・ブックは、CICS Transaction Gateway から出されるエラー・メッセージをリストし、説明しています。

  このマニュアルは注文できません。

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

  このマニュアルは、CICS Transaction Gateway での Java™ プログラミングの概要について説明します。

  その他に、プログラミングについての参照情報を含む 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ます。

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

  このマニュアルは、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ます。

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

  このマニュアルは、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ます。

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

  このマニュアルは、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ます。

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

  このマニュアルは、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ます。

- *CICS Transaction Gateway: Client Messages*

  このオンライン・ブックは、CICS ユニバーサル・クライアントから出されるエラー・メッセージおよびトレース・メッセージをリストし、説明しています。

  このマニュアルは注文できません。

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

  ECI および EPI 用のオブジェクト指向プログラムを C++ 言語で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

  ECI および EPI 用のオブジェクト指向プログラムを Component Object Model (COM) 規格に基づいて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

  このマニュアルは、CICS のクライアントおよびサーバーのプログラミングに関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)、および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する設計者および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・クライアントに関連するマニュアルのソフトコピーの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・クライアントに関連するマニュアルとファイル名

マニュアル名	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: この表のファイル名には、2 桁のサフィックスは含まれていません。	

サンプル構成の資料

いくつかのサンプル構成の資料を、PDF 形式で入手できます。

これらの資料では、各種のプロトコルを使用して CICS ユニバーサル・クライアントが CICS サーバーと通信するように構成する方法などが、ステップバイステップで説明されています。これらの資料には、CICS Transaction Gateway および 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を拡張する詳細な指示が含まれています。

追加のサンプル構成資料が入手可能になるたびに、それを弊社の Web サイトからダウンロードできます。

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

で、**Library** リンクをたどってください。

その他の出版物

以下の International Technical Support Organization (ITSO) レッドブック™ 資料には、クライアントおよびサーバーの構成に関する多数の例が含まれています。

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

  この本は、以下のマニュアルに置き代わるものです。
- *CICS Clients Unmasked*、*GG24-2534*

ITSO レッドブックは、いくつかの入手先から取得できます。最新の情報については、以下を参照してください。

www.ibm.com/redbooks/

CICS 製品についての情報は、以下を参照してください。

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

オンライン資料の表示

CICS Transaction Gateway および CICS ユニバーサル・クライアントで提供されている資料はすべて、弊社のオンライン・ライブラリーでアクセス可能です (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 および適切な Web ブラウザーが必要です (それらを構成することが必要な場合もあります)。

オンライン・ライブラリーに進むには、**ctgdoc** スクリプトを実行します。ライブラリーのホーム・ページが表示されます。

オンライン・ライブラリーから、以下の資料にリンクできます。

- PDF 形式の CICS Transaction Gateway および CICS ユニバーサル・クライアントマニュアル。
- ハイパーテキスト・マークアップ言語 (HTML) ファイルのプログラミング参照資料 (CICS Transaction Gateway 用のみ)。
- README ファイル。

- PDF 形式のサンプル構成資料。
- CICS の Web サイト。

Acrobat Reader の使用方法についても説明されています。

これらのマニュアルの最新版が随時提供されています。弊社の Web サイトも確認してください。

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

で、**Library** リンクを参照してください。

注: マニュアルの一部は他の言語にも翻訳されています。これらのマニュアルはオンライン・ライブラリーには含まれていませんが、上記の Web サイトから、独立した PDF ファイルとして入手することができます。

PDF マニュアルを見るには

PDF 情報には、以下の強力な機能があります。

- 情報内のナビゲート。 PDF 文書の中にハイパーテキスト・リンクがあり、他の PDF 文書および Web ページにリンクしています。
- 特定の情報の検索。
- PDF 文書の全体または一部を PostScript プリンターで印刷。

Acrobat Reader について詳しくは、次の Adobe の Web サイトを参照してください。

www.adobe.com/acrobat/

付録C. 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。

本書に含まれる情報には、技術的に不正確なもの、または、誤植が含まれる場合があります。これらに対する変更は、定期的に行われます。これらの変更は、資料の改訂版に含まれます。 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。

商標

以下は、IBM Corporation の商標です。

AIX	Anynet
CICS	eNetwork
IBM	MQSeries
MVS	MVS/ESA
OS/390	OS/400
RETAIN	TXSeries
VisualAge	VSE/ESA
VTAM	WebSphere

Lotus Domino、Lotus Notes、および Domino Go Webserver は、 Lotus Development Corporation の商標です。

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。

UNIX は、The Open Group がライセンスしている米国およびその他の国における登録商標です。

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]

D

E

G

H

I

J

K

L

O

P

R

S

T

U

W

X

プログラム番号: 5724-A75

Printed in Japan

SC88-8988-00

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