CICS® Transaction Gateway

OS/390® Gateway 管理

バージョン 4.0

SC88-8987-00
(英文原典：SC34-5935-00)

CICS® Transaction Gateway

OS/390® Gateway 管理

バージョン 4.0

SC88-8987-00
(英文原典：SC34-5935-00)

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、129ページの『付録B. 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 (OS/390® 版) (プログラム番号 5648-B43) のバージョン 4.0 に適用されます。また、新版で特に明示されない限り、これ以降のすべてのバージョン、リリース、および修正レベルにも適用されます。

本書は、SD88-7246 の改訂版です。ページの左側の縦線は、本版の新規箇所を示しています。

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： SC34-5935-00
CICS® Transaction Gateway
OS/390® 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

CICS® Transaction Gateway (OS/390® 版) バージョン 4.0 に加えられた機能上の変更内容は以下のとおりです。

- 構成ツールの新規オプションでは、環境変数の設定を行うことができます。
- 新規のトレース・オプションが提供されています。
- JCL を使用して複数の CICS Transaction Gateway を始動することができます。
- 構成変換ツールの ctgconv は提供されなくなりました。

機能上の変更についての情報のほかに、以下の情報も追加されました。

- PDF ライブラリーの抽出とインストール
- リモート・システムからの X Window システムの使用

使用可能度を向上させるため、本書の情報の大半が改訂されています。

本書は、SD88-7246 の改訂版です。ページの左側の縦線は、本版の新規箇所を示しています。

本書の構成

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 (OS/390® 版) 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 (OS/390® 版) のソフトウェア要件などの計画問題と、各種の移行問題について説明します。
- 第3章では、CICS® Transaction Gateway (OS/390® 版) 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 (OS/390® 版) をサポートするために CICS® Transaction Server (OS/390® 版) で実行する必要のある構成作業について説明します。
- 第5章では、CICS® Transaction Gateway (OS/390® 版) の構成方法について説明します。
- 第6章では、CICS® Transaction Gateway (OS/390® 版) 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7章では、CICS® Transaction Gateway (OS/390® 版) 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8章では、CICS® Transaction Gateway (OS/390® 版) の問題判別について説明します。
- 付録Aでは、CICS Transaction Gateway ライブラリー内のオンライン情報を表示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 (OS/390® 版) の計画、インストール、カスタマイズ、操作、またはプログラミングに関係する人を対象としています。

本書では、OS/390 オペレーティング・システム、および OS/390 UNIX® System Services を熟知していることを前提とします。

インターネット・テクノロジーおよび X Window システムについて理解していることも、本書を読む上で役に立ちます。

本書で使用する規則と用語

本書において、*CICS* ユニバーサル・クライアント という用語は、CICS Transaction Gateway の Client コンポーネントを指しています。

簡潔さのために、C プログラム言語と C++ プログラム言語の両方を指すために、 'C' を使用しています。

「CICS on System/390®」は、以下の CICS サーバー製品を表すために使用しています。

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

CICS Transaction Gateway は、 Windows NT® および Windows 2000 で実行されます。本書での Windows® への言及は、特定バージョンの Windows が指定されていないかぎり、 Windows NT と Windows 2000 の両方を意味します。

本文で OS/390 に言及するときは、OS/390 オペレーティング・システムと z/OS オペレーティング・システムの両方を指します。

前提事項および関連情報

以下に、CICS® Transaction Gateway (OS/390® 版) に関する資料をリストします。

IBM CICS® Transaction Server (OS/390® 版) およびその他の資料

CICS インターネットの手引き、SD88-7164

CICS 外部インターフェース・ガイド、SD88-7026

CICS Problem Determination Guide、GC33-1693

CICS リソース定義ガイド、SC88-7687

CICS System Definition Guide、SC33-1682

CICS ファミリー: システム/390 CICS からの通信®、SD88-7242

IBM CICS Transaction Gateway の関連資料

この製品について入手可能な資料の詳細については、123ページの『付録A. 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。その章でも、ソフトコピー資料を表示して印刷する方法や、印刷資料を IBM に注文する方法について詳しく説明しています。

その他の資料

- *OS/390 UNIX System Services Planning*、*SC28-1890*
- *OS/390 UNIX システム・サービス ユーザーズ・ガイド*、*SC88-6619*
- *OS/390 UNIX System Services Command Reference*、*SC28-1892*
- *OS/390 Security Server (RACF) Security Administrator's Guide*、*SC28-1915*
- *OS/390 Security Server (RACF) Command Language Reference*、*SC28-1919*
- *OS/390 MVS JCL Reference*、*GC28-1757*
- *OS/390 MVS Initialization and Tuning Reference*、*SC28-1752*
- *Secureway Communication Server IP Configuration*、*SC31-8513*
- *Secureway Communication Server IP Programmer Reference*、*SC31-8515*
- *OS/390 C/C++ Run Time Library Reference*、*SC28-1663*

第1章 CICS Transaction Gateway の概要

IBM CICS Transaction Gateway は、一定範囲の構成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、堅固で拡張が容易な、Web サーバーを補完するシステムであり、 (Java™ サーブレットの実行時環境である) IBM WebSphere™ 用の 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®、HP-UX、Linux、および Solaris の各プラットフォームに対応しています。また CICS Transaction Gateway は OS/390 プラットフォームにも対応しています。ただし、複数の CICS サーバーにアクセスできる他の CICS Transaction Gateway と異なり、CICS® Transaction Gateway (OS/390® 版) は Transaction Server (OS/390 版) にしかアクセスできません。

Windows および UNIX プラットフォーム上の CICS Transaction Gateway は、CICS ユニバーサル・クライアントを使用して、CICS サーバーに対し、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) の要求をルーティングします (13ページの『外部アクセス・インターフェース (EPI、ECI、および ESI)』を参照してください)。一方、CICS® Transaction Gateway (OS/390® 版) は ECI 要求だけしかルーティングできず、EPI 要求や ESI 要求はルーティングしません。CICS® Transaction Gateway (OS/390® 版) は実際には、CICS 外部呼び出しインターフェース (EXCI) を使用して、要求を CICS に渡します (14ページの『CICS 外部呼び出しインターフェース (EXCI)』を参照してください)。ただし、Java™ アプレットまたはアプリケーションにとっては、これらの要求は ECI 要求として現れます。

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。 CICS Transaction Gateway を Web サーバー・マシンにインストールする必要がある

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。

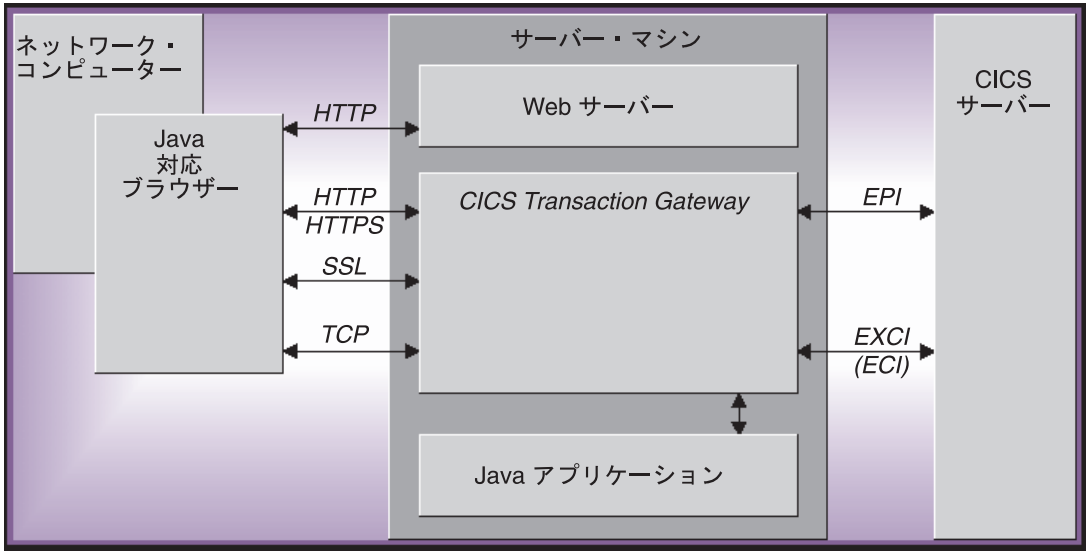

図 1. CICS Transaction Gateway

CICS Transaction Gateway との通信は、以下のプロトコルに基づいています。

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

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通信にも有効な方式です。ファイアウォールが存在する場所では、HTTP およびその代用で保護する HTTPS がインターネット・アプリケーションに適した通信プロトコルになります (15ページの『ネットワーク・セキュリティー』を参照してください)。

CICS® Transaction Gateway (OS/390® 版) が提供するもの

CICS® Transaction Gateway (OS/390® 版) は、次のものを提供しています。

1. EXCI (外部 CICS インターフェース) を介して、CICS サーバーで実行している CICS アプリケーションと通信する **Java Gateway アプリケーション**。この Java アプリケーションは、以前は IBM CICS Gateway for Java (MVS™) で利用できたものです。

2. 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 Gateway アプリケーションと Java アプリケーション (アプレット) 間との通信に使用するクラスが入っている、**CICS Java クラス・ライブラリー**。 Gateway プロセスとの通信を確立するには、クラス **JavaGateway** を使用します。このクラスは、Java のソケット・プロトコルを使用します。Gateway に流す ECI 呼び出しを指定する場合は、クラス **ECIRequest** を使用します。これらの Java クラスは、以前は IBM CICS Gateway for Java (MVS™) で利用できたものです。

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、複数の接続ユーザーを同時に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、セキュリティーの意味などの、 Java 言語について説明します。

Java 言語

Java 言語は、Java アプレット や Java アプリケーション を構成するために使用できます。

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の確保に不可欠な特殊なクラス、GUI 機能、入出力機能、およびネットワーク通信が入っています。

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 インタープリターによって提供されるこのような独立性から、これを Java 仮想マシンという場合もあります。

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、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、データ操作や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、アプレットに制御が与えられ、アプレットで使用され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 アプレットとは異なり、Web サーバー・マシン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 (Common Gateway Interface) プログラムを置き換えるものとして一般的に使用されるようになりました。CGI アプリケーションと比較した場合の、 Java サーブレットの利点は、サーブレットがデーモン・プ

ロセスでスレッドとして呼び出される (これは、メモリー内に常駐し、複数の要求を実行できることを意味します) ため、より迅速に実行できることです。

JavaGateway オブジェクトを使用するサーブレットでは、そのゲートウェイ・オブジェクトへの参照を必ずスレッド・セーフにすることに注意してください。

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し、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。

Java アプリケーションは、CICS 提供の Java クラスを使用して、CICS システムでトランザクション処理を行うことができます。 **JavaGateway** クラスを使用して 2 種類の接続を確立することができます。

- ネットワーク ・ゲートウェイ接続は、CICS Transaction Gateway へのネットワークにまたがる接続です。
- *ローカル* ・ゲートウェイは、Java アプリケーションが、ネットワークを必要とせずに、ローカルにインストールされた CICS Transaction Gateway と直接通信できるようにします。

アプリケーションと CICS Transaction Gateway 間の接続が確立されると、アプリケーションは **ECIRequest** クラスを使用して、トランザクション処理を行うことができます。(CICS® Transaction Gateway (OS/390® 版) については、**EPIRequest** および **ESIRequest** クラスを使用することはできません。)

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に割り当てられた用語です。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 - 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 - 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なければなりません。15ページの『SSL (Secure Sockets Layer)』および 17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 (OS/390® 版) には、JDK バージョン 1.1 対応の Java 対応 Web ブラウザーが必要です。たとえば Windows および AIX では、Netscape Communicator 4.5.1 またはそれ以降が必要です。詳細については、20ページの『Web ブラウザー』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、複数の Java アプレット (4ページの『Java アプレット』を参照) へ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、Lotus Domino™ 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 (OS/390® 版) がサポートする Web サーバーの詳細については、20ページの『Web サーバー』を参照してください。

CICS® Transaction Gateway (OS/390® 版) の CICS へのアクセス方法

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。

8ページの図2 は、CICS® Transaction Gateway (OS/390® 版) を使用して Web ブラウザーが CICS トランザクション処理機能を呼び出すときの制御の流れを表します。

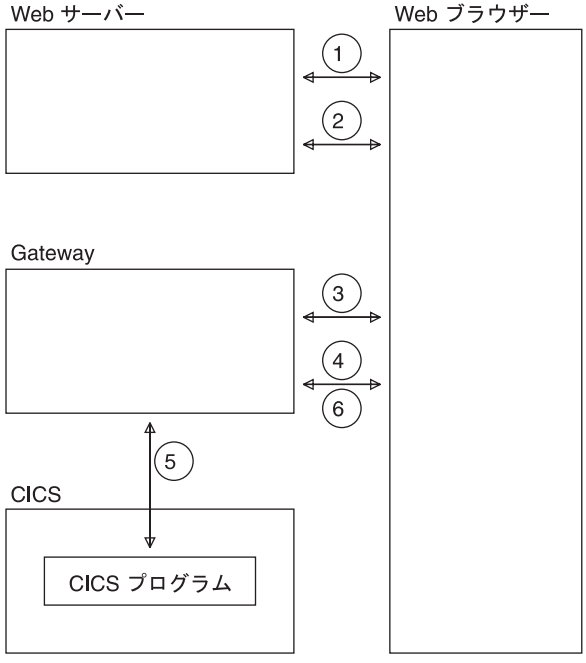

図 2. CICS® Transaction Gateway (OS/390® 版) を使用する制御の流れ

1. Web ブラウザーは、HTTP (Hypertext Transfer Protocol) を使用して Web サーバーを呼び出して、HTML ページを入手します。
2. HTML を解釈して画面に表示するブラウザーがアプレット・タグを検出すると、必要なアプレットとクラスを入手するために Web サーバーを呼び出します。それからアプレットを実行します。
3. CICS と通信しようとしているアプレットは、 **JavaGateway** オブジェクトを作成します。このオブジェクトを作成すると、CICS Transaction Gateway への呼び出しが長時間実行タスクになります。
4. アプレットは **ECIRequest** オブジェクトを作成して CICS プログラムについての要求を表し、**JavaGateway** オブジェクトの **flow** メソッドを呼び出し、 **ECIRequest** オブジェクトのインスタンスを渡します。
5. CICS Transaction Gateway は要求を受信し CICS プログラムを呼び出します。
6. CICS プログラムが終了すると、結果が Web ブラウザーに戻されます。

9ページの図3 は、ゲートウェイを使用して Web ブラウザーが CICS トランザクション処理機能を呼び出すときのデータの流れを表します。

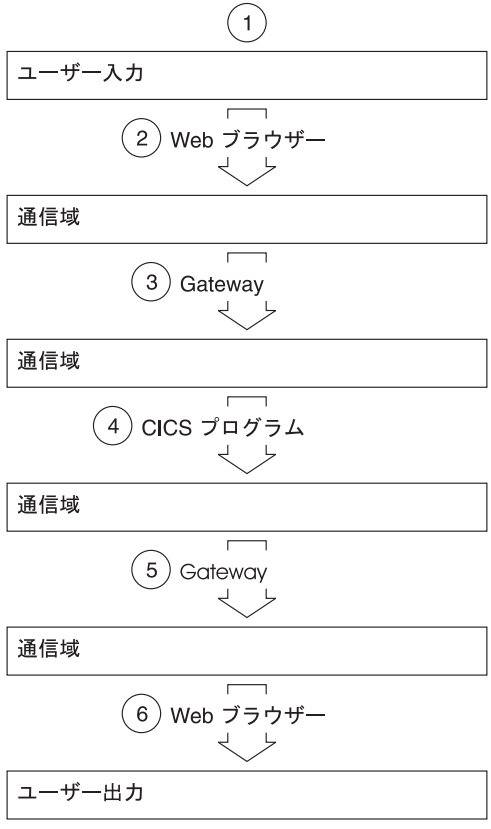

図 3. CICS® Transaction Gateway (OS/390® 版) を使用するデータの流れ

1. Web ブラウザーは、ユーザーから CICS プログラムについてのデータを獲得します。
2. Web ブラウザーは、トランザクション処理サービスを提供する CICS プログラムについて通信域 (COMMAREA) を構成します。
3. CICS Transaction Gateway は通信域を受信し、これに CICS プログラムを渡します。通信域の内容は、ASCII (ゲートウェイ内) から EBCDIC (CICS Transaction Server 内) に変換されます。データ変換は、CICS Transaction Server で実行されます。
4. CICS プログラムはトランザクション処理サービスを提供し、 CICS リソースについて質問し、おそらく変更します。プログラムが正常に終了すれば、リカバリー可能リソースへの変更はコミットされます。プログラムが異常終了する場合は、変更はバックアウトされます。

5. 通信域が EBCDIC から ASCII に変換され、ゲートウェイによって Web ブラウザーに戻されます。 (データ変換は、CICS Transaction Server で実行されます。)
6. Web ブラウザーはユーザーに情報を表示します。

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、このスレッドは使用可能な 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。初期 ConnectionManager 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**」の構成設定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**」設定を使用して指定できます (47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 (97ページの『コマンド行からの開始』を参照)。

Worker

Worker とは、Java クライアントからの要求を実際に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、行うべき作業が発生したときに各スレッドが活動化されます。 Worker スレッドは、完了すると、使用可能な Worker スレッドのプールに戻されます。 ConnectionManager の場合と同様、Worker リソース・プールにも初期サイズがあり、「**Worker スレッドの初期数**」の設定を使用して指定されます。 Worker プールの最大サイズは、「**Worker スレッドの最大数**」の設定で指定できます (47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 (97ページの『コマンド行からの開始』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。

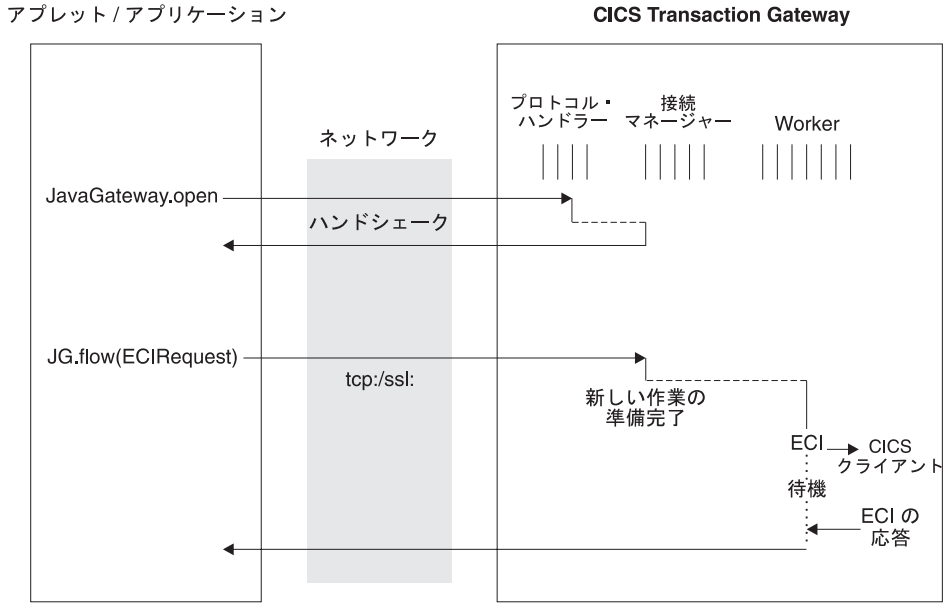

図 4. tcp/ssl の CICS Transaction Gateway スレッド化モデル

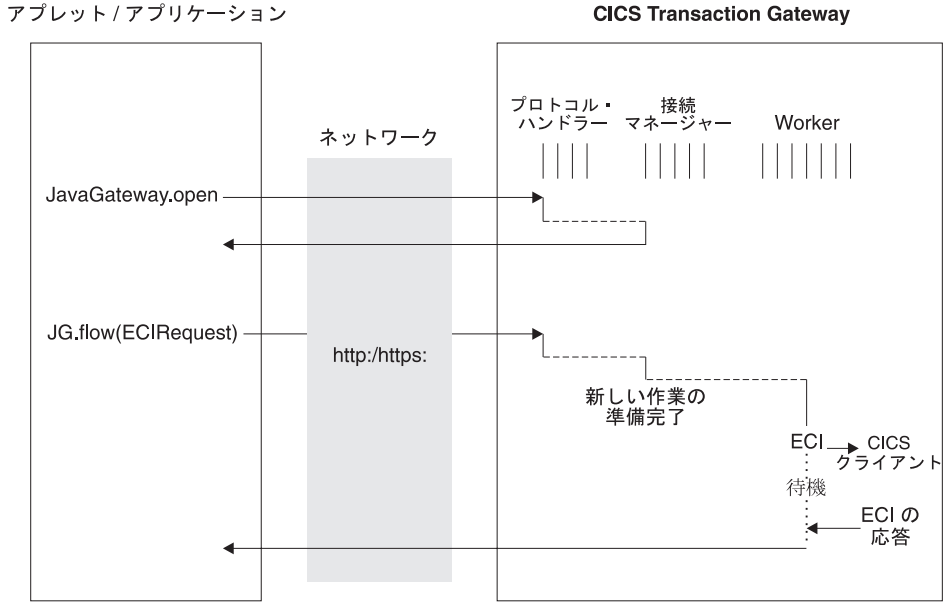

図 5. http/https の CICS Transaction Gateway スレッド化モデル

注: CICS® Transaction Gateway (OS/390® 版) では、図に示されている ECI 呼び出しが EXCI 呼び出しにマップされます。

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

表 1. CICS Transaction Gateway プラットフォームでのスレッドの限度

プラットフォーム	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
OS/390	MVS TCB の合計数により制限される (UNIX System Services スレッドごとに 1 つずつ作成される)	UNIX System Services パラメーターによって決まる: MAXTHREADS および MAXTHREADTASKS
Windows NT	制限なし	プロセスで使用可能な仮想メモリーの量により限定される。 (デフォルトでは、1 スレッドは 1M のスタックをもつ。これはプロセスごとに 2028 スレッドを作成できることを意味する。)
AIX	262,143	32768
Solaris	制限なし	制限なし
Linux	プロセスの最大数に等しい数	1024 (詳細は、ご使用の LinuxThreads 資料を参照)
HP-UX	制限なし (30 000 カーネル・スレッド)	30 000 (SAM ユーティリティーの Configurable Kernel Parameters を参照)

SYS1.PARMLIB の適切な BPXPRMxx メンバーを調べて、 UNIX System Services パラメーターの MAXTHREADS および MAXTHREADSTASK に設定されている値を、判別することができます。これらのパラメーターの詳細については、「*OS/390 UNIX System Services Planning*」を参照してください。

Java スレッドのスタック・サイズは、 Java -oss および -ss オプションを使って設定できます。留意点として、スレッドごとに割り振られるメモリーの量が、制約要因になることがあります。それは、スレッド限度に達する前にメモリーを使い果たしてしまうことがあるからです。

Java -oss および -ss オプションの使用法の詳細については、「*CICS Transaction Gateway: Gateway プログラミング*」を参照してください。

外部アクセス・インターフェース (EPI、ECI、および 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 通信機能とともに使用すれば、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。

外部セキュリティー・インターフェース (ESI) では、非 CICS アプリケーションが、拡張プログラム間通信 (APPC) のパスワード有効期限管理 (PEM) によって提供されるサービスを呼び出せるようにします。

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) は、CICS® Transaction Gateway (OS/390® 版) ではサポート**されません**。

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関係しており、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。

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 アプリケーションが 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同時に、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 を使用して呼び出し、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。 COMMAREA の長さは 32500 バイトを超えてはならないことに注意してください。

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、それ自身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分散プログラム・リンク (DPL) を使用します。また、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)、または (Transaction Server (OS/390® 版)™ 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) を使用します。

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*」を参照してください。

CICS 外部呼び出しインターフェース (EXCI)

外部 CICS インターフェース (EXCI) は ECI に類似しており、CICS 領域で実行するプログラムを呼び出したり、通信域を使ってデータを受信したりするために、 CICS® Transaction Gateway (OS/390® 版) のような、OS/390 で実行中のプログラムを使用可能にする、アプリケーション・プログラミング・インターフェースです。CICS® Transaction Gateway (OS/390® 版) は、EXCI のバージョン 2 をサポートします。

EXCI によって、ユーザーはセッション (またはパイプ) を CICS 領域に割り振って開くことができ、これらを介して分散プログラム・リンク (DPL) 要求を渡すことができます。 CICS 領域間通信 (IRC) 機能の複数領域操作 (MRO) 機能はこれらの要求をサポートし、各パイプは 1 つの MRO セッションにマップされます。 EXCI アドレス・スペースごとに 100 パイプまでの制限があります。

EXCI の詳細については、「*CICS 外部インターフェース・ガイド*」を参照してください。

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、 63ページの『第6章 CICS® Transaction Gateway (OS/390® 版) ネットワーク・セキュリティー』で詳しく解説しています。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、データの意味を理解できるのは、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) だけとなります。

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。公開鍵の暗号化は、暗号化と復号に対称鍵のペアを使う技法です。 SSL の場合、秘密鍵 (対称鍵) が、クライアントとサーバーの間で (公開鍵暗号化を使って) 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ています。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、メッセージがいたずらされた可能性があることを示す警告が発行されます。

責任能力 (Accountability)

責任能力はディジタル署名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。これにより、クライアントが CICS Transaction Gateway との接続を確立するときに、サーバーの詳細についての認証が必要になります。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、ディジタル証明書 (X.509v3 証明書) の交換に基づいています。これらのディジタル証明書には、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署名が含まれます。ディジタル証明書は、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。ユーザーは、CA の公開鍵を使用して証明書を復号できる場合、証明書に含まれている情報が信頼できるものである、つまり、その証明書が、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することになります。

System SSL および SSLight

CICS® Transaction Gateway (OS/390® 版) では次の 2 つのタイプの SSL がサポートされています。

System SSL は、SSL プロトコルをプラットフォーム固有に実装したものです。これは固有コードで作成されており、 OS/390 で使用可能なハードウェア暗号テクノロジーをサポートしています。

SSLight は、SSL を純粋な Java で実装したものです。

クライアント側のコード (つまり、分散アプレットで実行するコード) は常に SSLight を使用します。一方、サーバー側では、SSLight よりも System SSL を使用する必要があります。なぜなら、System SSL の方がパフォーマンスの大幅な向上が見られるからです。

HTTPS

現在の大半のブラウザーは、SSL を使用して HTTP サーバーに接続するために、URL アクセス方式 HTTPS をサポートしています。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” という形式の 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 (Internet Assigned Numbers Authority によって割り当てられた番号) です。

鍵および証明書

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ます。これは、X.509 プロトコルとも呼ばれます。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、外部的に署名された証明書を CA から入手するか、または自分自身を CA として確立して、「自己署名した」X.509 証明書を発行することができます。インターネットを使用する場合は、外部的に署名された証明書の方が適しています。一方、組織内での内部使用の場合は、自己署名された証明書の方が適しています。

X.509 ディジタル証明書は、System SSL の場合はデータベース・ファイル (.kdb) に「カプセル化」され、SSLight の場合は Java クラス・ファイルにカプセル化されます。こうして、SSL プロトコルおよび HTTPS プロトコルでこれらを使用することができます。データベース・ファイルまたは Java クラス・ファイルのことを *KeyRing* と言います。

System SSL では、CICS Transaction Gateway は OS/390 システム・ツール GSKKYMAN を使用して、ディジタル証明書を管理します。SSLight では、同様の目的のために、 CICS Transaction Gateway に iKeyMan ツールが含まれています。これらのツールを使用すると、KeyRing ファイルの作成、証明書の生成、および証明書のエクスポートを行ったり、その他の各種管理機能を実行することができます。詳細については、63ページの『第6章 CICS® Transaction Gateway (OS/390® 版) ネットワーク・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。これらの機能を説明するソース・ファイル例があります。

また、CICS Transaction Gateway がサーバー側のセキュリティー出口にあるクライアント証明書を公開することもできます。そうすると、クライアント証明書は RACF ユーザー ID にマップされます。詳細については、92ページの『RACF ユーザー ID へのクライアント証明書のマッピング』を参照してください。

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。

第2章 CICS® Transaction Gateway (OS/390® 版) の計画

この章は、ソフトウェア要件、サポートされている Web サーバーとブラウザー、また移行の問題について説明し、CICS® Transaction Gateway (OS/390® 版) のインストールの計画に役立つように編成されています。

ソフトウェア要件

この節では、CICS® Transaction Gateway (OS/390® 版) のソフトウェア要件をリストしています。

オペレーティング・システム

以下のレベルの OS/390 が必要です。

- OS/390 V2R8.0、またはそれ以降

Java 開発キット (JDK™)

Java アプレットとアプリケーションの開発には、**Java 開発キット (JDK)** バージョン 1.3.0 が必要です。

バージョンを調べるには、次の UNIX System Services コマンドを入力してください。

```
java -fullversion
```

Java 関連の環境変数に必要な設定値については、20ページの『OS/390 シェル環境の検査』を参照してください。

CICS® Transaction Server (OS/390® 版)

CICS® Transaction Gateway (OS/390® 版) を使用するには、それぞれ以下の APAR を CICS® Transaction Server (OS/390® 版) (CTS) に適用する必要があります。

CTS バージョン 1.2	APAR PQ31270
CTS バージョン 1.3	APAR PQ25809

OS/390 バージョン 1.3 の CTS 上での EXCI バージョン 2 のサポートは、APAR PQ38644 で提供されています。

Web サーバー

CICS® Transaction Gateway (OS/390® 版) は、以下の Web サーバーをサポートしています。

- Lotus Domino Go Webserver™
- WebSphere® Advanced Server for OS/390

Web ブラウザー

CICS Transaction Gateway クライアント・クラスで作成したアプレットは、JDK バージョン 1.1 準拠の Java 対応ブラウザーで動作します。これには、以下のものが含まれます。

- Windows NT: Microsoft® Internet Explorer バージョン 4.0.1、またはそれ以降 (トランスポートとしての HTTPS は動作しないことに注意してください。)
- Windows NT: Netscape バージョン 4.0.8、Netscape Communicator バージョン 4.51
- AIX: Netscape バージョン 4.0.8、Netscape Communicator バージョン 4.51
- Solaris: HotJava™ Browser バージョン 1.1

JDK AppletViewer を使用してアプレットを実行することもできます。

制約事項

Internet Explorer バージョン 4 (またはそれ以降) と CICS Transaction Gateway の HTTPS プロトコルの間には互換性の問題があります。これは、Internet Explorer の Java 仮想マシンで実行しているアプレットから CICS Transaction Gateway への https: 接続を確立できないということです。

OS/390 シェル環境の検査

CICS Transaction Gateway をインストールする前に、OS/390 シェル環境が正しくセットアップされていることを確認する必要があります。デフォルトのシステム全体のユーザー環境は、/etc/profile ファイルに指定されています。以下のことを検査してください。

- PATH ステートメントには、Java が配置されているバイナリー・ディレクトリーを組み込む必要があります。たとえば、以下のとおりです。

  ```
  export PATH=/usr/lpp/java/J1.3/bin:$PATH
  ```

  旧バージョンの CICS Transaction Gateway も実行する場合は、JDK 1.3 へのパスが、旧バージョンの CICS Transaction Gateway の JDK より*前* に指定されていなければなりません。

- PATH ステートメントには、uudecode ユーティリティー用のディレクトリーを組み込む必要があります。このユーティリティーはインストール時に使用されます。
- Java JIT コンパイラーを使用する場合、 JAVA_COMPILER 環境変数を jitc に設定し、 LIBPATH を適切に設定する必要があります。

注: JIT コンパイラーは HTTPS プロトコルおよび SSL プロトコルを使用した場合に問題を起こす可能性があるため、これらのセキュリティー・プロトコルを使用したい場合は、 JIT コンパイラーを使用不可にしてください。

```
export JAVA_COMPILER=none
```

CICS Transaction Gateway アプリケーションを local:// モードで実行する場合、次のように環境変数を設定してください。

```
JAVA_PROPAGATE=NO
```

これは、Java アプリケーションが実行される環境で設定する必要があります。この環境変数が設定されないと、 ECI 呼び出しは失敗し、ECI_ERR_SECURITY_ERROR 戻りコードが返されます。

ctgstart スクリプトを実行するプロセスの環境で、JAVA_HOME 変数を必ず使用可能にしてください。

OS/390 シェル環境の設定についての詳細は、「*OS/390 UNIX システム・サービス ユーザーズ・ガイド*、*SC88-6619*」を参照してください。

CICS Gateway for Java (MVS™) からの移行

CICS Gateway for Java (MVS™) から CICS® Transaction Gateway (OS/390® 版) に移行している場合は、環境変数を設定するために、Gateways.properties または JGate スクリプトをカスタマイズしてある場合があります。これらのファイルをカスタマイズしてある場合は、CICS® Transaction Gateway (OS/390® 版) をインストールする前にこれらのファイルのコピーを保管してください。

CICS® Transaction Gateway (OS/390® 版) バージョン 3.1 では、Gateway.properties ファイルの名前が CTG.INI に変更され、JGate スクリプトの名前は ctgstart に変更されました。古い Gateway.properties ファイルを新しい形式に変換してください。新しい設定の詳細については、47ページの『構成ツールの使用』を参照し、正しい形式については CTGSAMP.INI ファイルを参照してください。

CICS Transaction Gateway パッケージの名前が変更されたため、プログラム内のすべての import ステートメントを変更し、再コンパイルしなければなりません。したがって、以下のステートメントをすべて変更する必要があります。

import ibm.cics.jgate.client.* を import com.ibm.ctg.client.* に変更します。

import ibm.cics.jgate.security.* を import com.ibm.ctg.security.* に変更します。

ClientSecurity インターフェースおよび ServerSecurity インターフェースにおける変更により、これらのメソッドをインプリメントするユーザー・クラスを変更する必要があります。ハンドシェーク・データを生成するために呼び出すメソッドは、ハンドシェークしているユーザーの TCP/IP アドレスに渡されます。また、afterDecode メソッドが両方のインターフェースに追加されています。

CICS® Transaction Gateway (OS/390® 版) バージョン 3.x からの移行

環境変数を設定するために、 JGate スクリプトをカスタマイズしてある場合があります。このファイルをカスタマイズしてある場合は、CICS® Transaction Gateway (OS/390® 版) バージョン 4.0 をインストールする前にこれらのファイルのコピーを保管してください。

バージョン 3.1 では、Gateway.properties ファイルの名前が CTG.INI に変更され、JGate スクリプトの名前は ctgstart に変更されました。以前の Gateway.properties ファイルを新しい形式に変換してください。新しい設定の詳細については、47ページの『構成ツールの使用』を参照し、正しい形式については CTGSAMP.INI ファイルを参照してください。

注: 他の CICS Transaction Gateway プラットフォームで構成ツールを使って、OS/390 プラットフォームで使用するための CTG.INI ファイルを生成することができます。詳細については、他のプラットフォーム用の「*CICS Transaction Gateway 管理*」を参照してください。

各国語サポート

CICS® Transaction Gateway (OS/390® 版) の場合、英語、日本語、および中国語 (簡体字) だけがサポートされています (これらの言語が、OS/390 上の UNIX System Services でサポートされる言語であるため)。

第3章 CICS® Transaction Gateway (OS/390® 版) のインストール

この章では、CICS® Transaction Gateway (OS/390® 版) のインストール方法について説明します。

インストール

CICS Transaction Gateway は、階層ファイル・システム (HFS) にインストールされる CICS Transaction Gateway ソフトウェアを含む、圧縮ファイル ctg-400m.tar.Z として提供されています。

一般的には、最初にこの圧縮ファイルを、CICS Transaction Gateway をインストールしたいシステム/390 システムと通信できるワークステーション上のディレクトリーにコピーします。ただし、接続されている任意のシステムに CD をマウントして、ファイルをシステム/390 のシステムに転送することも可能です。また、ファイアウォールおよびその他のセキュリティー上の考慮が許せば、FTP を使用して IBM Web サイトから直接、OS/390 システムにファイルを転送することもできます。

CICS Transaction Gateway がルートの HFS データ・セット内に物理的に配置されていて、 UNIX System Services (USS) がアップグレードまたは取り替えられる場合には、ルート・ファイル構造を保持している物理データ・セットが置き換えられ、 CICS Transaction Gateway が削除されます。したがって、CICS Transaction Gateway を別個の HFS データ・セットにインストールし、それをルート・ファイル構造にマウントするのが望ましい方法です。 26ページの『個別の PDSE への CICS Transaction Gateway のインストール』を参照してください。この方法をとるときは、 CICS Transaction Gateway をインストールする前に行うことが必要です。

インストール手順

以下の手順は、圧縮ファイルをワークステーションにコピーしたことを前提としています。プログラム要素を HFS に転送するには、他の方式 (たとえば、FTP) を使用することもできます。

CICS Transaction Gateway をインストールするには、以下を実行します。

1. 次のように、プログラム要素をワークステーションから HFS の適切なディレクトリーに転送します。
   a. ファイルを MVS 順次データ・セットにアップロードします。これを行うには、ユーザーの端末エミュレーターの転送オプションを使用することができます。この順次データ・セットにはおよそ 220 トラック必要であることに注意してください。
   b. TSO 上の OPUT コマンドを使用してデータ・セットを HFS に入れます。たとえば、次のとおりです。

      ```
      OPUT 'CTG-400M.TAR.Z' '/path/ctg-400m.tar.Z' BINARY
      ```

      ここで、*path* は、CICS Transaction Gateway ディレクトリー構造のルートになる HFS パスです。たとえば、/usr/lpp です。
2. TSO の OMVS コマンドを使用するか、または直接、 UNIX System Services Telnet サーバーにログインして、OS/390 UNIX® System Services シェルに入ります。それから、次のコマンドを使用して

   ```
   cd /path
   ```

   ctg-400m.tar.Z を置いたディレクトリーに変更します。
3. 次のコマンドを使用して

   ```
   uncompress ctg-400m.tar.Z
   ```

   ファイルを圧縮解除します。
4. 次のコマンドを使用します。

   ```
   tar -xopfv ctg-400m.tar
   ```

   ここで、
   - `-x` は、すべてのファイルが抽出されることを指定します。
   - `-o` は、所有者、グループ、およびすべてのものが復元されるように、オリジナルのファイル許可ビットを許可します。これはすべての実行ビットに有効です。
   - `-p` は、ファイルを抽出するときにオリジナルの所有者およびグループ ID の復元を行いません。デフォルトではこれを行いますが、ほとんどの場合、抽出を行う人は許可をもっていないか、あるいはそのユーザー ID がシステムに存在しません。
   - `-f` は、アーカイブ・ファイルのファイル名を指定します。
   - `-v` は、長いメッセージを指定します (これはオプションです)。

ファイルが圧縮解除されると、ctg_install スクリプトが作成されるほか、ライセンス・ファイルを含む一時ディレクトリー /ctgtmp も作成されます。

5. ctg_install スクリプトを実行して、インストールを開始します。このスクリプトでは、インストール済みの OS/390 のレベル、システムに uudecode ユーティリティーがインストールされているか、およびインストールを継続するのに十分なスペースがあるかチェックされます。

   これらの前提条件のいずれかが利用できないと、ctg_install スクリプトはそれを通知するメッセージを生成して終了します。これらの問題が修正されたことを確認してから、ctg_install スクリプトを実行してください。
6. 前提条件の検査が完了すると、ユーザーのロケール用のライセンス・ファイルが表示され、ライセンスの条件を受け入れるようにプロンプトが出されます。次に CICS Transaction Gateway が取り出されて現行ディレクトリーにインストールされ、適切なサブディレクトリーが作成されます。
7. README.TXT ファイルには資料に記載されていない情報が含まれるため、このファイルを印刷して読むこともできます。

インストール・ディレクトリー構造

以下は、インストールされた CICS Transaction Gateway のディレクトリー構造です。

表 2. ディレクトリーの内容

相対パス	内容
(ルート)	ダウンロード・ディレクトリー。
./ctg	インストール・ディレクトリー。
./ctg/bin	CICS Transaction Gateway 実行可能コード。
./ctg/bin/resource	NLS リソース・ファイル。
./ctg/classes	Java クラス。
./ctg/docs	CICS Transaction Gateway についての資料。
./ctg/samples	サンプル用の Java ソース・ファイル。

インストール後、(/ctg/bin サブディレクトリー内の) の以下のファイルをブラウズしてみると役立ちます。

CTGSAMP.INI サンプル構成ファイル

ctgenvvarsamp サンプル環境変数ファイル

ctgcfg 構成ツールを開始するためのスクリプト

ctgikey ikeyman を開始するためのスクリプト

ctgstart	CICS Transaction Gateway 自体を開始するためのスクリプト

/ctg/samples/java/com/ibm/ctg/samples/security サブディレクトリーのファイルには、CICS Transaction Gateway セキュリティー出口の基本的な機能を示すサンプルが含まれています。このサンプルでは、java.util.zip パッケージを使用してクライアント / サーバーのデータ・フローの圧縮をデモンストレーションします。

CICS® Transaction Gateway (OS/390® 版) 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次のようにします。

1. TSO 上の OMVS コマンドを使用して OS/390 UNIX® System Services シェルに入ります。それから、次のコマンドを使用して

   ```
   cd /path
   ```

   CICS Transaction Gateway をインストールしたディレクトリー、たとえば、/usr/lpp に移動します。
2. 次のコマンドを使用して

   ```
   rm -fr ctg
   ```

   アンインストールします。

個別の PDSE への CICS Transaction Gateway のインストール

CICS Transaction Gateway のインストールは、個別の HFS データ・セットにインストールし、ルート・ファイル構造にマウントするのが最善です。CICS Transaction Gateway が配置されているルート HFS ファイル構造のポイントの下に ctg ディレクトリーを作成し、このディレクトリーに、755 のような適切な許可を与える必要があります。次に、 JCL ジョブを実行して、CICS Transaction Gateway HFS 用の PDSE を割り振り、この PDSE をルート・ファイル構造にマウントできます。それから CICS Transaction Gateway ファイルを圧縮解除し、その tar ファイルを作成し、インストール・スクリプトを実行できます。

また、STDERR および STDOUT を個別のディレクトリーに送ることも良い考えです。これにより、CICS Transaction Gateway エラー・ログおよびトレースを表示するための許可を個別に認可できるようになるので、CICS Transaction Gateway 実行可能ファイルを含むディレクトリーへの汎用の表示アクセスをセットアップする必要がなくなります。

28ページの図6 は、適切な PDSE を割り振り、それを HFS に MOUNT し、個別の /logs ディレクトリーをセットアップする、JCL の例です。この JCL を検討し、必要に応じて、PDSE 名、ソース CICS Transaction Gateway tar ファイル名、およびマウント・ポイントの場所を、各インストール・システムの標準に従って変更してください。

```
//useridA JOB (0),BATCH,MSGCLASS=X,REGION=4M,TIME=1439,
//  CLASS=A,NOTIFY=&SYSUID
//*
//* To run this job you need UID(0) or equivalent authority in USS.
//*
//* This job will remove an existing CTG from the HFS and add in
//* another CTG.
//* The CTG HFS will be created and inserted at /usr/lpp/ctg
//* You need only file transfer the CTG tar file to an MVS data set and
//* point the OPUT statement at the data set.
//* -------------------------------------------------------------
//* Unmount the existing CTG HFS at the mountpoint directory
//*   Edit the FILESYSTEM card to refer to your existing CTG PDSE
//* -------------------------------------------------------------
//UNMOUNT  EXEC PGM=IKJEFT01,REGION=4M,DYNAMNBR=99,COND=(0,LT)
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
  PROFILE MSGID WTPMSG
  UNMOUNT +
      FILESYSTEM('OMVS.USR.LPP.CTG400')
//* -------------------------------------------------------------
//*   Remove any existing CTG directory and re-create with correct
//*   permissions. Create separate logs directory.
//* -------------------------------------------------------------
//MKDIR    EXEC PGM=IKJEFT01,REGION=4M,DYNAMNBR=99
//SYSEXEC   DD DISP=SHR,DSN=SYS1.SBPXEXEC
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
  OSHELL rmdir '/usr/lpp/ctg'
  OSHELL mkdir '/usr/lpp/ctg'
  OSHELL chmod 755 '/usr/lpp/ctg'
  OSHELL mkdir '/usr/lpp/ctg/logs'
  OSHELL chmod 755 '/usr/lpp/ctg/logs'
/*
//* -------------------------------------------------------------
//* Delete any existing CTG HFS data set with the same name then ..
//* Allocate and mount the CTG HFS at the mountpoint directory
//*   Edit the DELETE, FREE ,ALLOCATE and FILESYSTEM cards to
//*   refer to the new CG PDSE
//* If you used ftp to transfer the CTG tar file directly into
//* /usr/lpp, remove the 'OPUT' command from the step below.
//* If you used file emulator file transfer to transfer the CTG tar
//* file into an MVS sequential data set, you need to execute the
//* 'OPUT' command to copy the tar file to /usr/lpp.
//* -------------------------------------------------------------
//*
//MOUNT    EXEC PGM=IKJEFT01,DYNAMNBR=99
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
```

図 6. PDSE を割り振り、マウントする JCL の例 (1/2)

```
  PROFILE MSGID WTPMSG

  DELETE 'OMVS.USR.LPP.CTG400' PURGE

  ALLOC DA('OMVS.USR.LPP.CTG') DSNTYPE(HFS)    +
         SPACE(20,5) DSORG(PO) CYL
  FREE  DA('OMVS.USR.LPP.CTG')

  MOUNT TYPE(HFS) +
    MODE(RDWR) +
    MOUNTPOINT('/usr/lpp/ctg') +
    FILESYSTEM('OMVS.USR.LPP.CTG400')

  OPUT 'hlq.CTG-400.TAR.Z' '/usr/lpp/ctg-400.tar.Z' BINARY
/*
//* ---------------------------------------------------------------
//* Uncompress and un-tar the tar file.
//* The tar command will populate the /usr/lpp/ctg directory.
//*  Edit the 'uncompress' command to point at the CTG tar file.
//* ---------------------------------------------------------------
//TAR      EXEC PGM=IKJEFT01,REGION=4M,DYNAMNBR=99
//SYSEXEC   DD DISP=SHR,DSN=SYS1.SBPXEXEC
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
  OSHELL cd /usr/lpp/
  OSHELL uncompress ctg-400.tar.Z
  OSHELL tar -xopfv ctg-400.tar
/*
```

図6. PDSE を割り振り、マウントする JCL の例 (2/2)

注:

1. このジョブで MOUNT 操作を実行するにはスーパーユーザー (UID=0) 権限をもっている必要があります。この JCL では、OMVS.USR.LPP.CTG400 と呼ばれる PDSE を作成してから、それをルート・ファイル・システムの /usr/lpp/ctg マウント・ポイント・ディレクトリーにマウントします。
2. CICS Transaction Gateway PDSE, CYL(20,5) のスペース割り振りでは、CICS Transaction Gateway そのものに 15 シリンダー、エラー・ログおよびトレース・レコード用に 5 シリンダー割り振ることが可能です。エラー・メッセージまたはトレースに大量のスペースが必要になることが予想される場合は、スペース割り振りで基本数量を大きくしてください。
3. CICS Transaction Gateway HFS の保持用に割り振りを選択した PDSE は、任意のデータ・セット名とすることができますが、それをルート・ファイル・システムにマウントするときに、UNIX System Services カーネル・タスクに、ユーザーが作成するデータ・セットへの UPDATE アクセスが必要となります。通常、 CICS Transaction Gateway HFS に使用する高位修飾子 (28ページの図6 に HFS として示されている) は、ルート・ファイル・シス

テムの HFS データ・セットの高位修飾子と同じになるため、RACF はおそらく、その高位修飾子をもつデータ・セットへの必要なアクセスを OMVS にすでに許可しています。ただし、ルート・ファイル・システムのものとは異なる高位修飾子をもつデータ・セットを選択する場合は、RACF コマンドを実行して、OMVS カーネル・タスク・ユーザー ID に、CICS Transaction Gateway HFS データ・セットへの UPDATE アクセスを許可する必要があります。

UNIX System Services のディレクトリー構造への、類似した名前をもつ PDSE の割り振りを試みてください。 MVS データ・セット名の最大長は、UNIX ディレクトリー・パス名の最大長より小さいので、これをいつも行うことはできません。ただし、特定のファイル構造をどの物理 PDSE が保持しているかは容易に識別できるため、この処置をお勧めします。したがって、28ページの図6では、 CICS Transaction Gateway PDSE は OMVS.USR.LPP.CTG400 であり、これが、ルート・ファイル・システムの /usr/lpp/ctg にインストールされます。この例では、 PDSE 名で修飾子 `CTG400` を使用していますが、これは、異なる修正レベルまたはリリース・レベルの CICS Transaction Gateway を保持するために個別の PDSE を作成できるからです。しかし、UNIX System Services では、実動レベルの CICS Transaction Gateway を 1 つだけ /usr/lpp/ctg の下に配置したい場合がほとんどなので、この例ではディレクトリー名にリリース・レベルが含まれていません。この理由は、/usr/lpp/ctg の使用によって、リリースごとに CICS Transaction Gateway を変更するために、 ctgstart シェル・スクリプトまたは CICS Transaction Gateway 領域 JCL を編集する必要がないからです。あるリリースの CICS Transaction Gateway を別のリリースに変更するには、 CICS Transaction Gateway HFS データ・セットを交換することだけが必要となります。

CICS Transaction Gateway HFS の永続的なマウント

28ページの図6 の JCL 例の MOUNT コマンドでは、現行 IPL のためにのみ CICS Transaction Gateway HFS をマウントします。各 IPL で CICS Transaction Gateway HFS が確実にマウントされるようにするには、SYS1.PARMLIB のメンバー BPXPRM*xx* にそれを追加する必要があります。ここで、*xx* は、SYS1.PARMLIB の IEASYS*xx* メンバーの BPX= オプションで指定されたサフィックスです。このような項目を追加して、各 IPL 後に確実に CICS Transaction Gateway がマウントされるようにします。

```
/* CTG HFS  */
  MOUNT FILESYSTEM('OMVS.USR.LPP.CTG400')
        MOUNTPOINT('/usr/lpp/ctg')
        TYPE(HFS)
```

'/logs' ディレクトリーの使用

28ページの図6 に示されているように、 CICS Transaction Gateway からの STDERR と STDOUT を保持するために、個別の /logs ディレクトリーを使用したい場合は、新規ディレクトリーを参照するように CICS Transaction Gateway JCL を変更することも必要です。

CICS Transaction Gateway JCL において、以下のような DD カードを組み込んでください。

```
//CTGC001  JOB  ......
// .......
//STDOUT   DD PATH='/usr/lpp/ctg/logs/ctgC001.out',
//            PATHOPTS=(OWRONLY,OCREAT),PATHMODE=SIRWXU
//STDERR   DD PATH='/usr/lpp/ctg/logs/ctgC001.err',
//            PATHOPTS=(OWRONLY,OCREAT),PATHMODE=SIRWXU
```

上記の DD カードの例では、CICS Transaction Gateway の名前 ctgC001' が、ログ・ファイル名で使用されるので、どのログがこの CICS Transaction Gateway に属するかが明確に分かります。この例では、ジョブ名 CTGC001 は、ストリング 'CTG と、EXCI を介してそれが接続する CICS 領域の SYSIDNT とから派生しています。このような規則を採用すると、各 CICS Transaction Gateway にどのメッセージが属しているかを識別することが常に可能です。

CTG.INI ファイルでは、同じ /logs ディレクトリーに位置するように、CICS Transaction Gateway トレース・ファイルを設定したい場合もあります。

```
# Enable trace file
tfile="¥usr¥lpp¥ctg¥logs¥ctgC001.trace"
```

リモート・システムからの X Window システムの使用

リモート・システム上の X Window システムを使用して、OS/390 で稼働するアプリケーションにアクセスすることができます。 X Window システムは、ウィンドウ操作とグラフィックスとサポートする、ネットワーク透過型のプロトコルです。このプロトコルは、 TCP/IP ネットワークでの X クライアント (アプリケーション) と X サーバー間で伝送されます。X Window システム・サポートは、 AIX 製品の一部として、また Windows プラットフォーム上で、たとえば、 Hummingbird® Exceed 製品によって提供されます。

X Window システム環境では、X サーバーは通常、ワークステーション上に配置されます。このサーバーが、同一システム上かネットワークのどこかに位置する 1 つまたは複数の X クライアント・プログラムとの間で、ユーザー入力を配布したり、要求を受け入れたりします。OS/390 ユーザーは通常、ホストで実行されるすべてのものをサーバーと見なし、ワークステーション上のすべてのアプリケーションをクライアントと見なすので、これは重要なポイントです。

リモート・システムから X Window システムを使用する際には、たとえば、構成ツールおよび iKeyman にアクセスするには、アプリケーションがそのシステム上にウィンドウを表示できるように、 DISPLAY 環境変数を設定しなければなりません。

表示システム (ウィンドウを表示するもの) で、次のコマンドを入力してください。

```
xhost +appl
```

ここで、*appl* は、アプリケーションの実行時に使用されるシステムのネットワーク名です。

ご使用の OS/390 システムの IP アドレスを調べるには、 TSO コマンド HOMETEST を使用します。

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てください。

```
DISPLAY=disp:0
```

その後に、次のコマンドを続けます。

```
export DISPLAY
```

ここで、*disp* は、ウィンドウが表示されるシステムのホスト名です (その後にコロンと表示装置 ID (通常は 0) が続きます)。そうすると、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。

Windows プラットフォーム上の IP アドレスを調べるには、`ipconfig /all` コマンドを実行します。UNIX プラットフォームでは、`hostname` コマンドを実行します。

.profile ファイルに次のステートメントを追加することができます。

```
export DISPLAY=disp:0
```

PDF ライブラリーの抽出およびインストール

CICS Transaction Gateway の PDF ライブラリーは、 /ctg/docs サブディレクトリーにあります。このライブラリー (cclia60i.tar.Z) は、圧縮された TAR アーカイブ形式です。

OS/390 用には現在、使用可能な Adobe Acrobat Reader がないため、PDF 文書を表示するには、Adobe Acrobat Reader を実行できるワークステーションに PDF ライブラリーをインストールする必要があります。

cclia60i.tar (バイナリー・フォーマット) を複数のプラットフォームに転送し、それを抽出することができます。TAR アーカイブは UNIX プラットフォームの標準です。また、Windows の下に、このアーカイブ形式を抽出することのできるいくつかのツール (たとえば、Windows 用の PKZIP) があります。

Adobe Acrobat Reader は、CD-ROM で提供されており、 Adobe ディレクトリーの下のそれぞれのプラットフォームごとのサブディレクトリーにあります。ご使用のプラットフォームを選択し、Reader インストール・プログラムを実行してください。

これらの Reader よりも後のバージョンが使用可能な場合がありますので、www.adobe.com をご覧ください。

プロダクト CD-ROM がある場合は、PDF ディレクトリーに個々の PDF ブックも見つかります。

tar ファイルを抽出し、Windows ワークステーションに転送するには、次のようにします。

1. TSO READY から TSO コマンド `OMVS` を入力するか、 ISPF オプション 6 を使用して、 UNIX System Services シェルに入ります。
2. UNIX System Services のモード変更サブコマンドを使用します。たとえば次のとおりです。

   ```
   chmod  755  /usr/lpp/ctg/docs
   ```

   これにより、自分自身に /docs ディレクトリーへの書き込みアクセス権を与え、圧縮解除コマンドが失敗しないようにします。
3. ディレクトリー変更サブコマンドを入力します。たとえば次のとおりです。

   ```
   cd /usr/lpp/ctg/docs
   ```
4. 次の UNIX System Services の圧縮解除コマンドを入力します。

   ```
   uncompress cclia60i.tar.Z
   ```

これでファイルの名前が、cclia60i.tar に変更されます。

5. PF2 を押し、QUIT サブコマンドを入力して、UNIX System Services シェルを終了します。
6. ISPF オプション 3.2 を使用して、名前 *userid*.CCLIA60I.TAR を持つデータ・セットを割り振ります。このデータ・セットには以下の属性があります。

```
Organization  . . . : PS
Record format . . . : VB
Record length . . . : 32756
Block size  . . . . : 32760
1st extent tracks . : 170
Secondary tracks  . : 10
```

7. TSO コマンドを入力して、

```
OGET '/usr/lpp/ctg/docs/cclia60i.tar'
'userid.CCLIA60I.TAR'     BINARY
```

cclia60i.tar ファイルを UNIX System Services から MVS 順次データ・セット *userid*.CCLIA60I.TAR にコピーします。

8. FTP を使用するか、または以下の方法で、 *userid*.CCLIA60I.TAR をワークステーションに転送します。
   a. OS/390 システムで TSO READY に進みます。
   b. 最上部のバーの「**Transfer**」を選択します。
   c. 「**Receive Files from Host**」を選択します。
   d. 以下を指定します。

```
Host file     : 'userid.CCLIA60I.TAR'
PC   file     : C:¥temp¥cclia60i.tar
Transfer type : binary
```

「**RECEIVE**」を選択します。受信が完了すると、ワークステーションの temp ディレクトリーに、ファイル cclia60i.tar があるはずです。ここで、OS/390 システムから、*userid*.CCLIA60I.TAR を削除できます。

9. Windows 用の PKZIP を使用して、c:¥temp¥cclia60i.tar ファイルからファイルをデコードし、抽出します。この方法の説明については、PKZIP ヘルプ情報を参照してください。

サービスおよびアップデート

次の Web サイトでサービスとアップデートが使用可能になっています。

```
www.software.ibm.com/ts/cics/support/details
```

第4章 CICS® Transaction Server (OS/390® 版) の構成

この章では、CICS Transaction Gateway をサポートするために CICS® Transaction Server (OS/390® 版) で実行する必要のある構成作業について説明します。

CICS Transaction Gateway のための CICS 定義

CICS Transaction Gateway では、EXCI についてセッション定義と接続定義が必要です。いくつかの例を挙げます。

```
VIEW GROUP(MYEXCI) CONNECTION(JCOS)
 OBJECT CHARACTERISTICS                                    CICS RELEASE = 0530
  CEDA  View Connection( JCOS )
   Connection     : JCOS
   Group          : MYEXCI
   DEscription    : Sample EXCI Specific connection
  CONNECTION IDENTIFIERS
   Netname        : JGATE400
   INDsys         :
  REMOTE ATTRIBUTES
   REMOTESYSTem   :
   REMOTEName     :
   REMOTESYSNet   :
  CONNECTION PROPERTIES
   ACcessmethod   : IRc               Vtam | IRc | INdirect | Xm
   PRotocol       : Exci              Appc | Lu61 | Exci
   Conntype       : Specific          Generic | Specific
   SInglesess     : No                No | Yes
   DAtastream     : User              User | 3270 | SCs | STrfield | Lms
+  RECordformat   : U                 U | Vb
                                                SYSID=CW2C APPLID=IYCWZCFY
PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

図 7. 接続定義の例--最初の画面

```
 OBJECT CHARACTERISTICS                                    CICS RELEASE = 0530
  CEDA  View Connection( EXC1 )
   Queuelimit     : No                No | 0-9999
   Maxqtime       : No                No | 0-9999
  OPERATIONAL PROPERTIES
   AUtoconnect    : No                No | Yes | All
   INService      : Yes               Yes | No
  SECURITY
   SEcurityname   :
   ATtachsec      : Identify          Local | Identify | Verify | Persistent | Mixidpe
   BINDPassword   :                   PASSWORD NOT SPECIFIED
   BINDSecurity   : No                No | Yes
   Usedfltuser    : No                No | Yes
  RECOVERY
   PSrecovery     :                   Sysdefault | None
   Xlnaction      : Keep              Keep | Force
                                                    SYSID=CW2C APPLID=IYCWZCFY
PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

図 8. 接続定義の例--2 番目の画面

これらの定義を作成する方法については、「*CICS* 外部インターフェース・ガイド」を参照してください。

```
VIEW GROUP(MYEXCI) SESSIONS(JCOS)
 OBJECT CHARACTERISTICS                                    CICS RELEASE = 0530
  CEDA  View Sessions( JCOS     )
   Sessions       : JCOS
   Group          : MYEXCI
   DEscription    : Sample EXCI Specific sessions definition
  SESSION IDENTIFIERS
   Connection     : JCOS
   SESSName       :
   NETnameq       :
   MOdename       :
  SESSION PROPERTIES
   Protocol       : Exci              Appc | Lu61 | Exci
   MAximum        : 000 , 000         0-999
   RECEIVEPfx     : JG
   RECEIVECount   : 004               1-999
   SENDPfx        :
   SENDCount      :                   1-999
   SENDSize       : 04096             1-30720
+  RECEIVESize    : 04096             1-30720
                                                    SYSID=CW2C APPLID=IYCWZCFY
PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

図 9. セッション定義の例--最初の画面

```
OBJECT CHARACTERISTICS                                  CICS RELEASE = 0530
 CEDA  View Sessions( EXC1SESS )
  SESSPriority   : 000               0-255
  Transaction    :
 OPERATOR DEFAULTS
  OPERId         :
  OPERPriority   : 000               0-255
  OPERRsl        : 0                                              0-24,...
  OPERSecurity   : 1                                              1-64,...
 PRESET SECURITY
  USERId         :
 OPERATIONAL PROPERTIES
  Autoconnect    : No                No | Yes | All
  INservice      : Yes               No | Yes
  Buildchain     : Yes               Yes | No
  USERArealen    : 000               0-255
  IOarealen      : 04096 , 04096     0-32767
  RELreq         : No                No | Yes
  DIscreq        : No                No | Yes
                                                 SYSID=CW2C APPLID=IYCWZCFY
PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

図 10. セッション定義の例--2 番目の画面

上記の例では、CICS Transaction Gateway が使用する特定の接続を定義しています。環境変数 DFHJVPIPE で CICS Transaction Gateway に指定される値は、接続定義の **Netname** オプション (図の中では、**JGATE400** として示されている) と同じでなければなりません。ネット名そのものは任意の値です。汎用パイプはほぼ同じように定義されます。ただし、**Netname** オプションはブランクのままになります。

CONNECTION 定義をインストールした後、以下のように、領域間通信 (IRC) をオープンにする必要があります。

```
CEDA INSTALL GROUP(groupname)
```

これで、この定義を含むグループをインストールし、

```
CEMT SET IRC OPEN
```

IRC をオープンにします。詳細については、「*CICS リソース定義ガイド*」を参照してください。

注: サンプルの DFH$EXCI を使用する場合は、DFHJVPIPE 環境変数を BATCHCLI に設定して、特定のパイプを使用する必要があります。変数を設定しない場合は汎用パイプが使用されます。詳細については、41ページの『環境変数の設定』を参照してください。

第5章 CICS® Transaction Gateway (OS/390® 版) の構成

この章では、CICS® Transaction Gateway (OS/390® 版) を構成するためにできることを説明します。

- 環境変数を設定します。たとえば、EXCI パイプの使用や、呼び出し側がアクセスできる CICS サーバーの名前を制御するための環境変数を設定します。
- 構成ツールを使用するか、直接 CTG.INI を編集して、 CTG.INI ファイルで CICS Transaction Gateway の特性を設定します。
- CICS Transaction Gateway を開始するために使用される ctgstart スクリプトを編集します。
- RACF を使用するように CICS Transaction Gateway を構成します。

環境変数の設定

CICS® Transaction Gateway (OS/390® 版) に関係のある環境変数は、ctgenvvar スクリプトを使って設定します。 ctgenvvar スクリプトを単独で実行するか、または ctgstart スクリプトでそのスクリプトを参照して、変数を設定できます。

ctgenvvar スクリプトは、次の 2 つの方法で作成できます。

1. 提供されている ctgenvvarsamp スクリプトを編集し、次にそれを ctgenvvar スクリプトにコピーまたは名前変更する方法。 45ページの『ctgenvvarsamp スクリプトの編集』を参照してください。
2. 構成ツールを使用して、環境設定を定義し、ctgenvvar スクリプトを作成する方法。 56ページの『OS/390 環境設定』を参照してください。

   ctgenvvarsamp 中の値はデフォルトとしてロードされます。

ctgstart の実行時に ctgenvvar スクリプトが検出されない場合、 ctgstart スクリプト自体の中にある環境変数の設定値が使用されます。

以下の方法で環境変数を設定することもできます。

1. OS/390 UNIX® System Services コマンド行でコマンドを入力する。たとえば、 /ctg/bin ディレクトリーで発行される次のコマンドは、DFHJVPIPE 環境変数を、 JAVAGAT1 という特定のパイプを使用するように設定します。

```
export DFHJVPIPE=JAVAGAT1
```

環境変数の値にスペースが含まれる場合は、単一の引用符または二重引用符で囲む必要があります。以下に例を示します。

```
export DFHJVSYSTEM_00="MYCICS-Test CICS system"
```

2. JCL を作成して環境変数の設定を組み込むゲートウェイを開始する。99ページの『JCL での開始』を参照してください。 ctgenvvar または ctgstart スクリプトで環境変数を設定する場合、これらの設定によって JCL 内の設定値がオーバーライドされます。

すべての CICS 領域は、領域に固有な DFHJVSYSTEM_*nn* を除いて、変数の同じインスタンスを参照します。これは、サーバー名とその記述から構成されます。

環境変数

CICS Transaction Gateway 用に以下の環境変数をエクスポートすることができます。これらの変数がどのように構成ツールおよび ctgenvvar の設定に関連するかについての詳細は、44ページの表3 を参照してください。

AUTH_USERID_PASSWORD

EXCI 呼び出しで渡されるユーザー ID とパスワードが RACF で認証されていなければならないかどうかを指定します。

AUTH_USERID_PASSWORD を `Yes` に設定する場合は、_BPX_SHAREAS も `Yes` に設定します。また、パスワードの許可が必要なときにとるべき処置について知らせるポップアップ・メッセージも表示されます。

CLASSPATH

CICS 提供 Java クラス定義についての HFS のパス、すなわち、/ctg/classes を組み込みます。 CLASSPATH を設定して Java アプリケーションをコンパイルおよび実行**しなければなりません**。

CTG_RRMNAME

CICS Transaction Gateway のこのインスタンスを管理するリソース・マネージャーの名前。リソース・リカバリー・サービスの場合、名前はシスプレックス内で固有にする必要があります。

名前は以下の印刷可能文字で構成することができます。

1. 英数字: A-Z および 0-9
2. 国別文字: $ (X'5B')、# (X'7B')、 @ (X'7C')
3. ピリオド (.)

4. 下線 (_)

名前をブランクで始めたり、名前に組み込みブランクを含めたりすることはできません。小文字は大文字に変換されます。

名前の競合を避けるには、先頭文字に A-C または G-I を使用してください。

CTG_RRMNAME の長さは 32 文字を超えてはなりません。

DFHJVPIPE

CICS Transaction Gateway が EXCI 呼び出しに使用する特定のパイプの名前。この変数が設定されない場合は、CICS Transaction Gateway は汎用パイプを使用します。

DFHJVSYSTEM_*nn*

DFHJVSYSTEM_*nn* の形式の環境変数を 100 まで設定します。 *nn* の範囲は 00 から 99 です。最初の変数は DFHJVSYSTEM_00 で、続く変数は連続番号でなければなりません。そうでないと認識されません。変数の値は、**CICS_EciListSystems** 関数の要求に応えて戻される CICS システムの名前と説明です。この値は、等号 (=) の直後に CICS システムの大文字の APPLID、その後にハイフン (-)、その後に CICS システムの説明を含む文字列でなければならず、長さは 60 バイトを超えてはなりません。たとえば次のように指定する場合:

```
DFHJVSYSTEM_00=MYCICS-Test CICS system
```

CICS_EciListSystems は、 MYCICS という CICS システムの詳細と、説明 `Test CICS system` を戻します。

LD_LIBRARY_PATH

CICS Transaction Gateway 実行可能コードについての HFS のパスを組み込みます。

LIBPATH

CICS Transaction Gateway 実行可能コードについての HFS のパスを組み込みます。この変数は `/ctg/bin` に設定しなければなりません。

STEPLIB

デフォルトの EXCI オプションと EXCI ロード・モジュールを含むライブラリー。 STEPLIB は、EXCI_OPTIONS 変数と EXCI_LOADLIB 変数の連結です。 EXCI_OPTIONS は、EXCI オプション・テーブル DFHXCOPT を含むデータ・セットを参照します。デフォルトで提供されているテーブルは、SDFHEXCI に含まれています。
EXCI_LOADLIB は、SDFHEXCI および SDFHLOAD ライブラリーを参照する必要があります。

環境変数の要約

表3 は、構成ツールの設定 (56ページの『OS/390 サーバー設定』と 56ページの『OS/390 環境設定』を参照) および ctgenvvar スクリプトに含まれる変数に、環境変数がどのように関連するかを表しています。

表 3. 環境変数

環境変数	構成ツールの設定	ctgenvvar 環境変数
AUTH_USERID_PASSWORD	RACF 認証の使用	AUTH_USERID_PASSWORD
CLASSPATH	CTG ユーザー・クラスパス	CLASSPATH、CTG_USER_CLASSPATH、CTG_CLASSPATH の連結
CTG_RRMNAME	RRM 名	RRMNAME
DFHJVPIPE	EXCI ネット名	DFHJVPIPE
DFHJVSYSTEM_*nn*	CICS サーバー, CICS 説明	DFHJVSYSTEM_*nn*
LD_LIBRARY_PATH	CTG ユーザー・ライブラリー・パス	LD_LIBRARY_PATH、CTG_USER_LIBPATH、CTG_LIBPATH の連結
LIBPATH	CTG ユーザー・ライブラリー・パス	LIBPATH、CTG_USER_LIBPATH、CTG_LIBPATH の連結
STEPLIB	EXCI ロード・ライブラリー (高位修飾子、ライブラリー名)、EXCI オプション	STEPLIB、EXCI_OPTIONS、EXCI_LOADLIB の連結

ctgenvvarsamp スクリプトの編集

以下は、環境変数がどのようにエクスポートされるかを示す、 ctgenvvarsamp スクリプトからの抜粋です。

```
#########################################################################
# Warning: Directly editing this file may affect the Configuration Tool #
# To set variables using this file, rename it to 'ctgenvvar'            #
#########################################################################

RRM_NAME="CCL.CTG"

EXCI_OPTIONS="your.user.loadlib"
EXCI_LOADLIB="CICSTS13.CICS.SDFHEXCI"
CTG_USER_CLASSPATH=""
CTG_USER_LIBPATH=""
CTG_CLASSPATH="${CTGSTART_HOME}/../classes/ctgclient.jar:
 ${CTGSTART_HOME}/../classes/ctgserver.jar:${CTGSTART_HOME}/../classes/cfwk.zip"
CTG_LIBPATH="${CTGSTART_HOME}"
DFHJVPIPE=""
AUTH_USERID_PASSWORD=No
DFHJVSYSTEM_00="NEW SERVER-Default example server"

export CTG_RRMNAME="${RRM_NAME}.IBM.UA"
export STEPLIB=${STEPLIB}:${EXCI_OPTIONS}:${EXCI_LOADLIB}
export CLASSPATH=${CTG_USER_CLASSPATH}:${CTG_CLASSPATH}:$CLASSPATH
export LIBPATH=${CTG_USER_LIBPATH}:${CTG_LIBPATH}:$LIBPATH
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${CTG_USER_LIBPATH}:${CTG_LIBPATH}
export DFHJVPIPE
export DFHJVSYSTEM_00

unset AUTH_USERID_PASSWORD
```

EXCI オプションのカスタマイズ

EXCI オプション・テーブル DFHXCOPT を調整して、 CICS Transaction Gateway で使用される EXCI オプションをカスタマイズするという選択が可能です。この場合は、 EXCI_OPTIONS 環境変数を更新して、カスタマイズされたオプション・テーブルを含むライブラリーを指定しなければなりません。また、以下の方法でも指定できます。

- 構成ツールを使用して、「**EXCI オプション (EXCI Options)**」を設定する。 56ページの『OS/390 環境設定』を参照してください。
- ctgenvvar スクリプトを編集する。次の行を検出して、

  ```
  EXCI_OPTIONS="your.user.loadlib"
  ```

  それからこの行を、適切なライブラリーを指定するように変更します。

EXCI オプションについての詳細とカスタマイズの方法については、「*CICS 外部インターフェース・ガイド*」を参照してください。

注: CICS 領域にセキュリティーがない (SEC=NO) 場合は、 DFHXCOPT テーブル・オプション SURROGCHK=NO を指定しなければなりません。詳細については、「*CICS RACF Security Guide*」を参照してください。

EXCI_LOADLIB 環境変数に指定されるライブラリーを更新しなければなりません。このライブラリーには次のものが含まれます。

- デフォルトの EXCI オプション。
- EXCI ロード・モジュール。

以下の方法で EXCI_LOADLIB を更新できます。

- 構成ツールを使用して、「**高位修飾子 (High Level Qualifier)**」および「**ライブラリー名 (Library Name)**」を設定する。56ページの『OS/390 環境設定』を参照してください。
- ctgenvvar スクリプトを編集する。次の行を検出して、

  ```
  EXCI_OPTIONS="CICSTS13.CICS.SDFHEXCI"
  ```

  次にこの行を、適切なライブラリーおよび高位修飾子を指定するように変更します。

EXCI_OPTIONS 変数と EXCI_LOADLIB 変数は、 STEPLIB 環境変数の定義で使用されます。

構成ツールの使用

構成ツールを使用して、CICS® Transaction Gateway (OS/390® 版) 用の構成パラメーターを設定します。

構成ツールを使用するには、32ページの『リモート・システムからの X Window システムの使用』の説明にあるコマンドを使用して、画面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** コマンドを入力してください。

構成は、CICS Transaction Gateway をインストールした bin サブディレクトリーの CTG.INI ファイルにデフォルトで保管されています。場合によっては、構成ツールを使用せずに、このファイルを直接、編集することもできます。

構成ファイルには、CICS Transaction Gateway バージョン 3.0 の Gateway.properties ファイル ファイルと同等の項目が含まれています。

構成ツールを開始するときに構成ファイルがすでに存在している場合、そのファイルでの設定が構成ツールにロードされます。

構成ツールのインターフェース

構成ツールのユーザー・インターフェースは、メニュー・バー、ツールバー、ツリー構造、および設定パネルで構成されます。

5 番目のエリアとしてステータス・バーがあり、ユーザーが使用中の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 つの入力域間の移動には、F10、エスケープ・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には、F10 を押します。
- 選択を行わずにプルダウン・メニューをクローズするには、 エスケープ・キーを押します。
- ツールバー、ツリー構造、および設定パネル間を移動するには、 タブ・キーを使用します。タブ・キーを繰り返し押すと、設定パネルのすべてのオブジェクトを移動してから、すべてのアイコンを移動し、最後にツリー構造に戻ります。
- タブ・キーを押しても次のエリアに移動できない場合は、 F10 を押してからエスケープ・キーを押して (メニューをアクティブにしてから、クローズして)、その後にタブ・キーをもう一度押します。

ツリー構造
ツリー構造を使用すると、構成内のすべての設定にナビゲートすることができます。 **Java Gateway** ノードの下には、最高で 6 つのサブノードがあります。つまり、CICS Transaction Gateway が使用できるプロトコルごとにサブノードが 1 つずつあることになります。

アクセス可能度:
ノードとサブノードを移動するには、上矢印キーと下矢印キーを使用します。構成ツールをオープンすると、すべてのノードが展開されます。ノードを縮小表示するには、そのノードに移動して、左矢印キーを押します。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**」プルダウンには以下のオプションがあります。

新規 新しい構成を作成します。

オープン 既存の構成をオープンします。

保管 現在の構成を保管します。新しい構成が保管されている場合、ファイル名は CTG.INI です (bin サブディレクトリーにあります)。

別名保管 構成ファイルのデフォルトのパスおよび名前をオーバーライドできます。

終了 構成ツールを終了します。その際に、構成を保管するかどうか尋ねられます。

「**編集**」プルダウンには、**切り取り**、**コピー**、および**貼り付け**のオプションがあり、標準のテキスト機能を実行します。

「**ツール**」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 トレース設定ダイアログを表示します。

新規サーバー 新しいサーバーを作成します。

サーバーの削除
サーバーを削除します。

「**ヘルプ**」プルダウンには以下のオプションがあります。

項目ヘルプ 現在の画面コンテキスト (たとえば、特定の構成設定) に応じてオンライン・ヘルプ情報を表示します。

目次 オンライン・ヘルプ情報の目次リストを表示します。

索引 オンライン・ヘルプ情報の主題索引を表示します。

その他の情報 バージョン番号および構成ツールについてのその他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が、「**別名保管**」または「**終了**」オプションがないという点が異なります。アイコンには吹き出しヘルプがあります。カーソルを吹き出しヘルプの上に移動させると、オプションについて説明するテキスト・ボックスが表示されます。

設定パネル

ツリー構造で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。これらのパネルの設定値は、CTG.INI ファイルおよび ctgenvvar ファイル内のパラメーターに対応しています。

各設定パネルには、「**変更のやり直し (Undo Changes)**」ボタンがあり、変更を取り消すことができます。

CICS Transaction Gateway 設定の構成

構成ツールを使用すると、Gateway コマンド行オプションで指定できる任意のパラメーターの値を事前設定することができます。

Gateway を開始するときに、構成ツールを使って定義されたパラメーターが、関連する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***。

Gateway の一般設定

Gateway の一般設定は以下のとおりです。

接続マネージャー・スレッドの初期数: ConnectionManager スレッドの初期数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 スレッドの最大数を入力します。デフォルトは 100 です。

「**非制限 (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。

1 つの MVS アドレス・スペースにつき 100 個のパイプの制限があるので、接続マネージャー・スレッドの最大数は、(100 を、CICS Transaction Gateway と直接通信する CICS® Transaction Server (OS/390® 版) 領域の数で除算した値) に設定することをお勧めします。これは、パイプが、CICS サーバーごとのスレッド単位で割り振られるからです。

スレッドの限度については、12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 スレッドの初期数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 スレッドの最大数を入力します。デフォルトは 100 です。

「**非制限 (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が適用されません。

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

スレッドの限度については、12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。

デフォルトでは、コンソールからの入力の読み取りを使用可能にします。

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: メッセージで 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、メッセージで TCP/IP アドレスをどのように表示するかを選択することができます。デフォルトでは、 CICS Transaction Gateway はメッセージに数字形式で TCP/IP アドレスを表示します。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、メッセージ中の数字の 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、メッセージが読み取りやすくなります。

注: このオプションを選択すると、システムによっては重大なパフォーマンスの低下が起こる場合があります。

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。特定の応答は、 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 がまだそのプログラムのために要求を処理していることがあります。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。この期間を過ぎても、まだ進行中の要求がある場合には、ConnectionManager はその処理を続行し、新しい接続による使用が可能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、デフォルトでは 10000 ミリ秒に設定されていますが、値を入力して、このデフォルト値をオーバーライドできます。

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、要求を受け入れたら、その要求を実行するために、Worker スレッドを割り振る必要があります。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、デフォルトでは 10000 ミリ秒に設定されていますが、値を入力して、このデフォルト値をオーバーライドできます。

この値をゼロに設定した場合には、Worker が即時使用可能にならないと、要求が拒否されます。

TCP プロトコルの設定

TCP サブノードを選択すると、以下の TCP の設定が表示されます。

プロトコル・ハンドラーを使用可能にする: このプロトコルで使用するために 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 ポート番号を入力します。

TCP/IP のデフォルト・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。ウェイクアップには 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、新しい接続が受信された時にハンドラーがウェイクアップするだけで、その時間まで Gateway の完全なシャットダウンは実行されません。

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。この値をゼロに設定すると、ConnectionManager がすぐに使用可能にならないと接続が拒否されます。

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流された時点からカウントされます。タイムアウトが満了すると接続は切断されますが、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、アイドル接続は切断されません。

タイムアウト後の終結処理については、サーバーにそ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 メッセージの送信時間までに PONG 応答が受信されないと、接続が切断されます。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 (**作業中接続のドロップ**値の設定に応じて) 接続されたままになります。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・ボックスは、この接続のための作業がまだ進行中の場合**であっても**、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するときに選択します。

SO_LINGER 設定: このハンドラーで使用する任意のソケットの SO_LINGER 設定を入力します。 SO_LINGER 設定は、ソケットを終了するための遅延値を秒単位で指定します。この値が入力されないか、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、しかもデータ伝送がまだ完了していない場合、データが伝送されるまで、または接続がタイムアウトになるまで、 Close 呼び出しによって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、呼び出し側をブロックすることなく戻り、 TCP/IP は引き続きデータを送信しようとします。通常、この転送は成功しますが、保証はできません。なぜなら、TCP/IP は、指定された時間内だけしか Send 要求を繰り返さないためです。

クライアントにセキュリティー・クラスの使用を要求する: Gateway にセキュリティー・クラスを使用する接続だけを受け入れさせたい場合には、このチェック・ボックスを選択します。

Java クライアント・プログラムがこの Gateway に接続する際には、その接続で使用する必要のある 1 対のセキュリティー・クラスを指定できます。しかし、Gateway は、デフォルトでは、セキュリティー・クラスのこの対を指定していないプログラムからの接続も受け入れます。

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

CICS Transaction Gateway セキュリティー出口の詳細については、 18ページの『セキュリティー出口』を参照してください。

System SSL プロトコルの設定

System SSL サブノードを選択すると、System SSL の設定が表示されます。設定は TCP の場合と同じです。ただし、SSL は以下のように設定されています。

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。

クライアント認証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。

クライアント用のクライアント証明書を取得する方法については、76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 データベース: サーバー KeyRing のデータベース・ファイル名 (.kdb) を入力します。

サーバー KeyRing は、このサーバーを接続クライアントに識別するのに使用する有効な x.509 証明書から構成されていなければなりません。この KeyRing データベースは、System SSL ツール (GSKKYMAN) を使用して生成されます。

SSL およびその関連した KeyRing データベースの詳細については、63ページの『第6章 CICS® Transaction Gateway (OS/390® 版) ネットワーク・セキュリティー』を参照してください。

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

System SSL のデフォルトのポートは 8050 です。

SSL プロトコルの設定

SSL サブノードを選択すると、SSLight の設定が表示されます。設定は System SSL と同じですが、**KeyRing データベース**の代わりに、以下の設定が使用される点が異なります。

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 KeyRing は、このサーバーを接続先のクライアントに識別させるために使用する、有効な x.509 証明書から構成され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 およびそれに関連する KeyRing クラスの詳細については、63ページの『第6章 CICS® Transaction Gateway (OS/390® 版) ネットワーク・セキュリティー』を参照してください。

SSL のデフォルトのポートは 8050 です。

HTTP プロトコルの設定

HTTP サブノードを選択すると、HTTP の設定が表示されます。設定は、TCP の場合と同じです。

HTTP のデフォルトのポートは 8080 です。

System SSL HTTPS プロトコルの設定

System SSL HTTPS サブノードを選択すると、 HTTPS over System SSL が表示されます。設定は HTTP の場合と同じですが、System SSL HTTPS には、**クライアント認証の使用**、**KeyRing データベース**、および **KeyRing パスワード**の各設定もあります。

HTTPS のデフォルトのポートは 443 です。

HTTPS プロトコルの設定

HTTPS サブノードを選択すると、HTTPS over SSLight の設定が表示されます。設定は HTTP の場合と同じですが、HTTPS には、**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**の各設定もあります。

HTTPS のデフォルトのポートは 443 です。

トレース設定

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

Gateway: このチェック・ボックスを選択すると、 CICS Transaction Gateway のトレースが使用可能になります。

これは、ctgstart コマンドで -trace オプションを指定した場合と同じ機能を果たします。

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、トレース・メッセージの書き込み先となるトレース・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。

ctgstart コマンドの -tfile オプションを使用しても、トレース・ファイルを指定することができます。

CICS Transaction Gateway が開始するたびにトレース・ファイルが追加されるわけではありません。ファイルは上書きされます。

パフォーマンス: Gateway トレースをオンにすると、パフォーマンスに大きな影響があります。実稼働環境では、 Gateway トレースを使用可能にすることをお勧めしません。

Java ゲートウェイ・トレース・ラップ・サイズ: 0 ～ 1000000 K バイトの範囲で値を入力します。デフォルトは 0 です。

この値は、ゲートウェイ・トレース・ファイルがそこまで大きくなると予想されるサイズを K バイト単位で指定します。後続のトレース項目は、ファイルの先頭から書き込まれます。

OS/390 サーバー設定

新規 CICS サーバーを構成するには、「**ツール (Tools)**)」メニューか、ツールバーから、「**新規サーバー (New Server)**」を選択します。

最大で 100 のサーバーしかないことに注意してください。

CICS サーバー: 1 ～ 8 文字の範囲で名前を入力します。これは、**CICS_EciListSystems** 関数の要求に応答して戻される、CICS システムの名前です。

CICS 説明: ? 文字までの説明を入力します。これは、**CICS_EciListSystems** 関数の要求に応答して戻される、CICS システムの説明です。

OS/390 環境設定

OS/390 環境設定を構成するには、CICS サーバー・ノードを選択します。

これらの設定に関連した環境変数の詳細については、 41ページの『環境変数の設定』を参照してください。

RACF 認証の使用: EXCI 呼び出しで渡されるユーザー ID とパスワードが RACF で認証されていなければならないことを指定するときに、このチェック・ボックスを選択します。このオプションを選択する場合は、 58ページの『RACF® で使用するための CICS Transaction Gateway の構成』を参照して、認証が必要な場合に実行する作業についての詳しい情報を調べてください。

高位修飾子: デフォルトの EXCI ロード・モジュールが入っているライブラリーの名前を入力します。

ライブラリー名: デフォルトの EXCI ロード・モジュールが入っているライブラリーの名前を入力します。デフォルトは SDFHEXCI です。

RRM 名: CICS® Transaction Gateway (OS/390® 版) のこのインスタンスを管理するリソース・マネージャーの名前を入力します。この名前の最大長は 32 文字です。リソース・リカバリー・サービスの場合、名前はシスプレックス内で固有にする必要があります。

重要: 名前の競合を避けるには、先頭文字に A-C または G-I を使用してください。

EXCI オプション: デフォルトの EXCI オプションが入っているライブラリーの名前を入力します。

CTG ユーザー・クラスパス: 現行の CLASSPATH 環境設定に追加するパス名を入力します。構成ツールは、ctgclient.jar、ctgserver.jar、および classes.zip のパスを自動的に設定することに注意してください。

CTG ユーザー・ライブラリー・パス: 現行の LIBPATH 環境設定に追加するパス名を入力します。構成ツールは、native_threads のパスおよび ctgstart の位置を自動的に設定することに注意してください。

EXCI ネット名: CICS® Transaction Gateway (OS/390® 版) が EXCI 呼び出しに使用する特定のパイプの名前を入力します。特定のパイプが指定されないと、CICS® Transaction Gateway (OS/390® 版) は汎用パイプを使用します。

構成ファイルの編集

構成ツールを使用することをお勧めしていますが、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。サンプルの構成ファイル CTGSAMP.INI が用意されています。

CICS Transaction Gateway を再始動して、構成ファイルに加えた変更を取り出す必要があります。

ctgstart スクリプトのカスタマイズ

CICS Transaction Gateway を開始するために使用される ctgstart スクリプトは、 bin サブディレクトリーにあります。

ctgstart スクリプトには、環境変数の設定値を含めることができますが、それらが使用されるのは、始動時に ctgenvvar スクリプトが検出されない場合だけです。

System SSL を使用する場合は、以下の変更が必要です。

1. ステートメント `GSKLOAD="SYS1.SGSKLOAD"` を追加する。
2. ctgenvvar スクリプトに以下のような変更を加える。つまり、

   ```
   export STEPLIB=${STEPLIB}:${EXCI_OPTIONS}:${EXCI_LOADLIB}
   ```

   以下に変更します。

   ```
   export STEPLIB=${STEPLIB}:${EXCI_OPTIONS}:${EXCI_LOADLIB}:${GSKLOAD}
   ```

RACF® で使用するための CICS Transaction Gateway の構成

CICS Transaction Gateway では、RACF で使用されるユーザー ID とパスワードの認証をサポートしてます。また、登録済みの X.509 証明書を RACF ユーザー ID にマップするオプションもあります (92ページの『RACF ユーザー ID へのクライアント証明書のマッピング』を参照してください)。

どちらの機能でも、RACF を呼び出すために OS/390 UNIX C/C++ pthread_security_np 関数を使用します。この OS/390 UNIX 関数では、呼び出し側アドレス・スペースに「プログラム制御」というマークが付いていなければなりません。 ctgstart で呼び出された CICS Transaction Gateway を実行する JVM (Java 仮想マシン) は、このインスタンスの呼び出し側アドレス・スペースになります。

いくつかの HFS ファイルに、プログラムで制御されていることを示す「拡張属性」を設定しなければなりません。以下の **extattr** コマンドは、 CICS Transaction Gateway が使用するロード・モジュールに、「プログラム制御」というマークを付けます。

```
extattr +p path/ctg/bin/lib*.so
extattr +p path/ctg/bin/SECURES
```

ここで、*path* は、たとえば、/usr/lpp です。

ctgstart スクリプトで定義され、STEPLIB 環境変数で参照されるデータ・セット (.SDFHEXCI) のロード・モジュールにも、「プログラム制御」というマー

クが付けられている必要があります。プログラム制御のマーク付けの例については、60ページの『プログラム制御としてのプログラムのマーク付け』を参照してください。

ctgstart スクリプトがアドレス・スペースを他のプロセスと共用**しない**ことを指定しなければなりません。これは、呼び出し側アドレス・スペース (JVM) が、プログラムで制御されていないロード・モジュールによって「誤用」されないようにするためのものです。 JVM 独自の非共用アドレス・スペースを使用させるには、次のように入力します。

```
extattr -s path/ctg/bin/ctgstart
```

ここで、*path* は、たとえば、/usr/lpp です。

CICS Transaction Gateway ロード・モジュールのほかに、CICS Transaction Gateway アドレス・スペースによってロードされた Java HFS ファイルにも、「プログラム制御」というマークを付ける必要があります。以下の例では、JVM が /usr/lpp/java/J1.3 にあるものと想定しています。このディレクトリーで次のコマンド・セットを入力してください。

```
extattr +p bin/*
extattr +p lib/*
```

CICS Transaction Gateway が System SSL を使用するように構成されている場合、この別個の製品で提供されるバイナリーとデータ・セットにも、「プログラム制御」というマークを付ける必要があります。以下に例を示します。

```
extattr +p /usr/lpp/gskssl/lib/*
extattr +p /usr/lpp/gskssl/bin/*
```

System SSL を使用して生成された .kdb keyring ファイルと、 CICS Transaction Gateway によってロードされた .kdb keyring ファイルにも、「プログラム制御」というマークを付ける必要があります。

extattr コマンドを使用するには、ユーザーが、変更されるファイルの所有者またはスーパーユーザーでなければなりません。

注: CICS Transaction Gateway を実行するためのユーザー ID は、BPX.SERVER FACILITY プロファイルへの READ アクセスが許可されている必要があります。詳細については、「*OS/390 C/C++ ランタイム・ライブラリー・リファレンス*」(SC88-6319-04) の、 pthread_security_np についての節を参照してください。

CICS Transaction Gateway を実行するためのユーザー ID は、TCPIP.STANDARD.TCPXLBIN データ・セットを保護する RACF プロファイルに対する READ アクセスが必要です。

プログラム制御としてのプログラムのマーク付け

OS/390 においては、プログラム制御とは、 RACF 制御のプログラムを誰が実行できるかを、インストール・システムで制御することを可能にする RACF 機能です。以下は、プログラムにプログラム制御とマークを付けるために使用するコマンドの例です。これは、データ・セット (PADS) に対するプログラム・アクセスを使用せずにプログラムを保護する例です。

1. `SETROPTS WHEN(PROGRAM)`

   これによってプログラム制御がアクティブになります。

2. `ADDSD 'SYS1.LINKLIB' UACC(EXECUTE)`

   これによって、ユーザーによるプログラムのコピーを防止します。

3. `RDEFINE PROGRAM MYPROG ADDMEM('SYS1.LINKLIB'/123456/NOPADCHK) UACC(NONE)`

   これにより、MYPROG が被制御プログラムになります。MYPROG は、ボリューム 123456 上の 'SYS1.LINKLIB' のメンバーでなければなりません。

4. `SETROPTS WHEN(PROGRAM) REFRESH`

   これによって、新しい PROGRAM プロファイルがストレージに入ります。

詳細については、「*RACF セキュリティー管理者のガイド*」を参照してください。

その他の構成タスク

CICS Transaction Gateway の構成を完了するには、以下を実行してください。

- CICS Transaction Gateway から EXCI 要求用に CICS を構成する。このタスクについての情報は、「*CICS 外部インターフェース・ガイド*」に記載されています。
- Web ブラウザーが要求した CICS プログラムによって使用される通信域を変換するための変換テンプレートを作成する。このタスクについての情報は、「*CICS ファミリー: システム/390 CICS からの通信*」に記載されています。

トラフィック量が多いときに CPUTIME タイムアウトを大きくする方法

BPXPRMxx parmlib メンバーには、処理を制御するパラメーターおよびファイル・システムが含まれています。システムは初期化の際にこれらの値を使用します。

トラフィックの量が多い場合、SYS1.PARMLIB の BPXPRMxx メンバー内の MAXCPUTIME 値を変更することが必要になります。この値により、処理で使用できる最大 CPU 時間を秒単位で決定します。BPXPRMxx は、IPL 時の初期値を設定するために使用されます。この値は、SETOMVS コマンドを使用して変更できます。

OS/390 バージョン 2.8 以降、RACF ユーザー ID プロファイルの OMVS セグメントの CPUTIMEMAX 値を使用して、ユーザー ID ベースでこの値を設定することができます。詳細については、「*OS/390 UNIX System Services Planning*」を参照してください。

第6章 CICS® Transaction Gateway (OS/390® 版) ネットワーク・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・プロトコルの SSL および HTTPS を使用して、CICS Transaction Gateway をセットアップする方法について説明します。

- 64ページの『System SSL および SSLight』では、CICS® Transaction Gateway (OS/390® 版) をサポートするさまざまなタイプの SSL について説明し、それらを比較します。
- 65ページの『概説』では、ネットワーク・セキュリティーの概念とその用語について概説します。また、暗号鍵、ディジタル証明書、および KeyRing の概念について紹介します。
- 69ページの『SSL と認証』では、SSL プロトコルとそれが提供する認証のタイプについて説明します。
- 71ページの『ctgikey ツール』では、ディジタル証明書を管理するために CICS Transaction Gateway で提供されている SSLight ツール、iKeyMan について説明します。
- 72ページの『外部的に署名された証明書の使用 (SSLight)』では、外部的に署名されたディジタル証明書を入手する方法、および CICS Transaction Gateway で使用するように、それらを KeyRing クラスに保管する方法について説明します。
- 78ページの『自己署名された証明書の使用 (SSLight)』では、CICS Transaction Gateway で使用するように、自己署名されたディジタル証明書を生成する方法について説明します。
- 83ページの『gskkyman ツール』では、ディジタル証明書を管理するために OS/390 V2R7.0 で提供されている System SSL ツール、gskkyman について紹介します。
- 84ページの『外部的に署名された証明書の使用 (System SSL)』では、外部的に署名されたディジタル証明書を入手する方法、および CICS Transaction Gateway で使用するように、それらを KeyRing データベース・ファイルに保管する方法について説明します。
- 88ページの『自己署名された証明書の使用 (System SSL)』では、CICS Transaction Gateway で使用するように、自己署名されたきディジタル証明書を生成する方法について説明します。

- 92ページの『RACF ユーザー ID へのクライアント証明書のマッピング』では、クライアント証明書を RACF ユーザー ID にマップする方法について説明します。
- 93ページの『SSL および HTTPS 用の CICS® Transaction Gateway (OS/390® 版) の構成』では、セキュア・プロトコルを使用するように CICS Transaction Gateway を構成する方法について説明します。

System SSL および SSLight

1ページの『第1章 CICS Transaction Gateway の概要』で説明したとおり、CICS Transaction Gateway では 2 つの Secure Sockets Layer (SSL) をインプリメントします。

純粋な Java で作成された汎用的なものを **SSLight** と言います。これは、iKeyman という、純粋な Java による鍵 / 証明書管理ツールによってサポートされています。

もう 1 つを **System SSL** と言います。これは、OS/390 上の CICS Transaction Gateway にのみ適用され、SSL サーバーでのみ使用できます。System SSL は固有コードで作成され、暗号ハードウェア・アクセラレーター・カードを余分にサポートしているため、パフォーマンスが大幅に向上します。これは、gskkyman という、独自の鍵 / 証明書管理ツールによってサポートされています。 (このツールは OS/390 V2R7.0 に付属します。)

そのため、CICS® Transaction Gateway (OS/390® 版) では、System SSL あるいは SSLight 用の SSL: および HTTPS: プロトコル・ハンドラーを構成することができます。しかし、パフォーマンス上の理由から、System SSL の方を SSLight より優先して使用してください。 SSLight と System SSL のプロトコル・ハンドラーをサーバー側で構成し、使用する方法は非常に似ています。93ページの『SSL および HTTPS 用の CICS® Transaction Gateway (OS/390® 版) の構成』を参照してください。

iKeyman と gskkyman ではどちらも、外部的に署名された証明書と自己署名された証明書の使用をサポートしています。

クライアント側のコード (つまり、分散アプレットで実行するコード) の場合は、常に SSLight が使用されます。 CICS Transaction Gateway で現在サポートされている暗号機能の最大のレベルは、DES 56bit です。しかし、OS/390 用の CICS Transaction Gateway の場合、VeriSign Step-Up Server 証明書をインポートすることができます。それにより、Java SSLight クライアント・コード

(およびその他のエクスポート機能付き SSL クライアント) が使用可能になり、暗号機能が大幅に向上します (現在は、128bit までです)。

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 CICS Transaction Gateway の概要』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。

- **責任能力 (Accountability)**

  責任能力とは、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。責任能力がないと、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。責任能力はディジタル・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。

暗号化とは ?

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、責任能力を保証します。

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したい場合、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. 自分あ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。

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用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慎重に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、安全に情報を共用できるようになります。

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。

SSLight の場合: 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。 93ページの『SSL および HTTPS 用の CICS® Transaction Gateway (OS/390® 版) の構成』を参照してください。

システム SSL の場合: CICS Transaction Gateway は鍵データベース・ファイルを使用して、証明書および鍵データを SSL サーバーに保管します。鍵データベース (.kdb) ファイルは、gskkyman ツールを使用して作成します。そのため、CICS Transaction Gateway デーモンは、サーバー KeyRing (たとえば、**ServerKeyRing.kdb** という名前) を使用します (.kdb ファイルがシステム LIBPATH に含まれていなければなりません)。 SSL プロトコルおよび HTTPS プロトコルはこのキー・データベース・ファイルにアクセスして、セキュリティー接続を確立する必要があります。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。 93ページの『SSL および HTTPS 用の CICS® Transaction Gateway (OS/390® 版) の構成』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、図11 に示します。

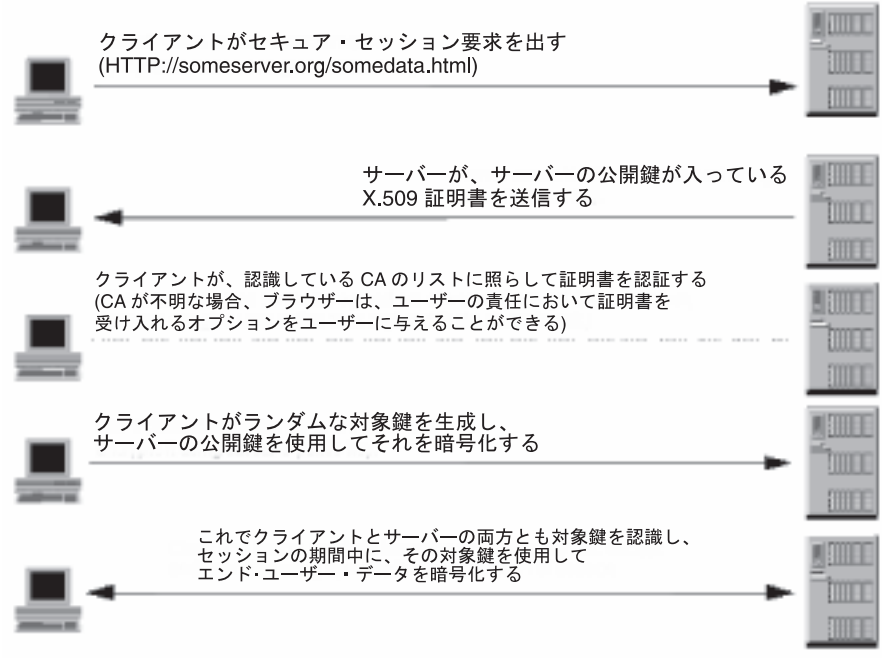

図 11. サーバー認証による SSL ハンドシェーク

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 は異なるプロトコルであり、通常は異なるポート (それぞれ 443 と 8080) を使用するため、SSL 要求と非 SSL 要求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。72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。78ページの『自己署名された証明書の使用 (SSLight)』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 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**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、76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の名前を指定して、「**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. 「**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サーバー認証だけを使用するので、クライアントは、提示されたサーバー証明書を受け入れることだけ必要です。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 (SSLight)

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をエクスポートする

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、「**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ます。
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。

古い自己署名された証明書を移行する

CICS Transaction Gateway バージョン 3.0 では、自己署名された証明書の使用だけが許可され、外部的に署名された証明書はサポートされませんでした。

バージョン 3.0 を使用して作成した KeyRing クラス・ファイルは iKeyman ツールにインポートできるので、 CICS Transaction Gateway バージョン 4.0 では、古い自己署名された証明書も使用できます。

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。

System SSL のソフトウェア前提条件とハードウェア前提条件

System SSL バージョン 2 リリース 7 は、 OS/390 の Cryptographic Services Base エレメントの一部です。

System SSL では追加のハードウェアは必要ありませんが、適切な暗号ハードウェアをインストールすれば、パフォーマンスは向上します。 Cryptographic Coprocessor Feature は、 IBM S/390® Multiprise® 2000 システム、および IBM S/390® Parallel Enterprise Server™ Generation 3 システムのオプショナル機能です。これは、IBM S/390® Parallel Enterprise Server Generation 4 システムでは標準です。 IBM 4753-014 ネットワーク・セキュリティー・プロセッサーは、Cryptographic Coprocessor Feature が含まれていない S/390 システムに付属していることがあります。

gskkyman ツール

gskkyman ツールは、rlogin OS/390 シェル環境からでも、OMVS シェル・コマンド行環境からでも実行できます。

gskkyman が実行できる作業には、以下のものがあります。

- キー・データベースの作成
- 認証要求 (および公開鍵 / 秘密鍵のペア) の作成
- 自己署名された証明書の作成
- ファイルへの認証要求のエクスポート
- 認証要求実行後の証明書の受け取り
- ファイルへの証明書のエクスポート
- ファイルからの証明書のインポート
- トラステッド CA 証明書としての証明書の作成
- キー・データベース用のデフォルト証明書としての証明書 (および秘密鍵) の作成
- 証明書情報のリスト
- キー・データベースからの証明書 (および秘密鍵) の削除
- キー・データベースの削除

gskkyman の詳細を得るには、次のコマンドを入力して、ヘルプ情報を表示してください。

```
gskkyman -h
```

または、「*Cryptographic Services System Secure Sockets Layer Programming Guide and Reference*」(SC24-5877) を参照してください。

外部的に署名された証明書の使用 (System SSL)

外部的に署名された証明書を使用する場合は、gskkyman を使用して、 CICS® Transaction Gateway (OS/390® 版) を System SSL サーバーとしてセットアップする必要があります。以下のようにしてください。

- キー・データベースを作成する。
- 認証要求を作成する。
- 要求の実行後、証明書を受け取る。
- 証明書をトラステッド CA 証明書としてインポートする。

キー・データベースの作成

新しいキー・データベースを作成するには、以下のようにします。

1. gskkyman を入力します。「IBM Key Management Utility」メニューが表示されます。
2. 1 (Create new key database) を入力し、以下のようにプロンプトに応答します。

```
Enter key database name or press ENTER for "key.kdb": mykey.kdb
Enter password for the key database.......> entered password
Enter password again for verification.....> entered password
Should the password expire? (1 = yes, 0 = no) [1]:
Enter password expiration time (number of days) or press ENTER
for 60 days: 35
```

3. Enter (実行) キーを押してから、パスワード有効期限を入力すると、以下のメッセージとプロンプトが表示されます。

```
The database has been successfully created, do you want to continue to work
with the database now? (1 = yes, 0 = no) [1]:
```

1 を入力するか、Enter (実行) キーを押すと、キー・データベース・メニューが表示されます。このメニューで、SSL サーバーのセットアップを行えます。

gskkyman を使ってキー・データベースを作成すると、データベースの作成に使用するパスワードを入力するように指示されます。データベースにアクセスするときには、必ずパスワードを指定する必要があります。キー・データベースを使用する前に、gskkyman を使用して、隠しパスワード・ファイルを作成してください。そうすれば、パスワードを指定しなくてもデータベースにアクセスできます。

キー・データベースは、OS/390 Unix System Services の階層ファイル・システム (HFS) に、ファイルとして作成されます。

認証要求 (および公開鍵 / 秘密鍵のペア) の作成

認証要求を作成するには、「Key database」メニューで、3 (Create new key pair and certificate request) を入力し、以下のようにプロンプトに応答します。

```
Enter certificate request file name or press ENTER for "certreq.arm":
Enter a label for this key.............> tjh1
Select desired key size from the following options (512):
    1:    512
    2:    1024

Enter the number corresponding to the key size you want: 1
Enter certificate subject name fields in the following.

    Common Name (required).............> servermachine.hursley.ibm.com
    Organization (required)............> IBM
    Organization Unit (optional).......> Hursley
    Country Name (required 2 characters)..> UK

Please wait while key pair is created...

Your request has completed successfully, exit gskkyman?

(1 = yes, 0 = no) [0]: 0
```

認証要求を作成する際に、その認証要求に含める項目の数を入力するように指示されます。最初に、認証要求を保管するファイルの名前が尋ねられます。デフォルトの名前は certreq.arm です。ラベルは、キー・データベースに保管されるすべての証明書と認証要求に関連付けられます。そのため、認証要求のラベルを選択する必要があります。次に、キーのサイズと、証明書の一部となる識別名の個々の部分を入力するように指示されます。

認証要求が作成されると、指定した名前の付いたファイルが現行作業ディレクトリーに入っています。

作成した認証要求は、base64 エンコード形式のファイルに保管されます。この形式は、通常、証明書を作成する認証局で必要とされるものです。これまで実行されたステップによって作成されるファイルの内容を以下に示します。

```
   $ cat certreq.arm
-----BEGIN NEW CERTIFICATE REQUEST-----

   MIH7MIGmAgEAMEExCzAJBgNVBAYTAlVTMQwwCgYDVQQKEwNJQk0xETAPBgNVBAsT
   CEVuZGljb3R0MREwDwYDVQQDEwhKb2huIERvZTBcMA0GCSqGSIb3DQEBAQUAA0sA
   MEgCQQCrIZdRnXhH1EMAwTuKMKYznCFp4CFk0YG66BhvMGgfTwql9aSRWkVcer8I
   I7Qk9aYzQ2LIpRhloJ9ugojy1I9VAgMBAAGgADANBgkqhkiG9w0BAQQFAANBAFc1
   xOfunjyt54dUqGDdPgbnMr5A3trUhzHHkX8xlfH9Albrpsv2a3FjvnmYWFPuFXAf
   3ABCD5nnsbk3AP++ic5UTM=

-----END NEW CERTIFICATE REQUEST-----
   $
```

このファイルは別のシステム (テキスト・ファイルなど) に転送してから認証局に転送することもできますし、カット・アンド・ペースト方式を使って、認証局に送られるメール・メッセージに直接添付することもできます。

要求に応答して認証局から証明書が作成されたら、次の節の説明に従って、それを受け取り、キー・データベースに入れる必要があります。

要求後の証明書の受け取り

CA から証明書が戻されたら、証明書をキー・データベースに保管しなければなりません。通常、CA は Base64 エンコード証明書を含む電子メール・メッセージを送ります。

証明書を受け取るには、Base64 エンコード証明書を、OS/390 システムの HFS ファイルに保管して、gskkyman コマンドで読み取れるようにする必要があります。 gskkyman の開始時には、このファイルは現行作業ディレクトリーになければなりません。

証明書を受け取るには、以下のようにします。

1. 「Key database」メニューで、4 (Receive a certificate issued for your request) を入力します。
2. 以下のようにプロンプトに応答します。

```
Enter certificate file name or press ENTER for "cert.arm":
Do you want to set the key as the default in your key database?
(1 = yes, 0 = no) [1]:

Please wait while certificate is received......

Your request has completed successfully, exit gskkyman?
(1 = yes, 0 = no) [0]: 0
```

Base64 エンコード証明書情報を含むファイルの名前を入力するように指示されます。証明書を受け取ったら、キー・データベースで作業を継続するか、 gskkyman を停止することができます。

証明書のインポート

証明書を生成するために使用する CA が、証明書をすでにキー・データベースに保管しているいずれのデフォルト認証局でもない場合は、 CA の証明書をキー・データベース・ファイルにインポートしなければなりません。

クライアント認証を使用する場合は、CA 証明書をサーバー・プログラムのキー・データベースにインポートする必要があります。 SSL 接続でクライアント認証を使用するかどうかに関係なく、クライアント・プログラムのキー・ファイルに、 CA 証明書がインポートされていなければなりません。

キー・データベースが作成されると、既知の認証局 (CA) の証明書がキー・データベースに保管されます。キー・データベースの作成時に証明書が保管される CA のリストは、以下のとおりです。

```
Integrion Certification Authority Root
IBM World Registry™ Certification Authority
Thawte Personal Premium CA
Thawte Personal Freemail CA
Thawte Personal Basic CA
Thawte Premium Server CA
Thawte Server CA
Verisign Test CA Root Certificate
RSA Secure Server Certification Authority
Verisign Class 1 Public Primary Certification Authority
Verisign Class 2 Public Primary Certification Authority
Verisign Class 3 Public Primary Certification Authority
Verisign Class 4 Public Primary Certification Authority
```

ある証明書をキー・データベース・ファイルに CA 証明書としてインポート (保管) するには、まず OS/390 HFS のファイルにある証明書を、 Base64 エンコード形式のファイルに加えます。

証明書を CA 証明書としてキー・データベースに保管するには、以下のようにします。

1. 「Key database」メニューで、6 (Store a CA certificate) を入力します。
2. 以下のようにプロンプトに応答します。

```
Enter certificate file name or press ENTER for "cert.arm":
Enter a label for this key.................> xxx

Please wait while certificate is stored...

Your request has completed successfully, exit gskkyman?
(1 = yes, 0 = no) [0]: 0
```

この操作が終了すると、証明書は「承認された」ものと見なされて、着信証明書を検査するために使用できます。 SSL サーバーの役割をするプログラムの場合、この証明書はクライアントの証明書の検査時に使用されます。 SSL クライアントの役割をするプログラムの場合、この証明書は SSL のハンドシェーク処理時にクライアントに送られるサーバーの証明書を検査するために使用されます。

自己署名された証明書の使用 (System SSL)

自己署名された証明書を使用する場合は、gskkyman を使用して、 CICS® Transaction Gateway (OS/390® 版) を System SSL サーバーとしてセットアップする必要があります。以下のようにしてください。

- 新しいキー・データベースを作成する。
- 自己署名された証明書を作成する。
- 証明書を SSL クライアントにエクスポートする。

キー・データベースの作成

手順は、外部的に署名された証明書の場合と同じです。

自己署名された証明書の作成

認証要求を作成するには、「Key database」メニューで、5 (Create a self=signed certificate) を入力し、以下のようにプロンプトに応答します。

```
Enter version number of the certificate to be created (1, 2, or 3) [3]: 3
Enter a label for this key.............> tjh2
Select desired key size from the following options (512):
    1:    512
    2:    1024

Enter the number corresponding to the key size you want: 1
Enter certificate subject name fields in the following.

    Common Name (required).............> servermachine.hursley.ibm.com
    Organization (required)............> IBM
    Organization Unit (optional).......> Hursley
    Country Name (required 2 characters)..> UK

Enter number of valid days for the certificate [365]:  244
Do you want to set the key as the default in your key database? (1 = yes, 0 =
no) [1]:
Do you want to save the certificate to a file? (1 = yes, 0 = no) [1]: 1
Should the certificate binary data or Base64 encoded ASCII data be saved? (1 =
ASCII, 2 = binary) [1]:1

Enter certificate file name or press ENTER for "cert.arm":

Please wait while self-signed certificate is created...

Your request has completed successfully, exit gskkyman?

(1 = yes, 0 = no) [0]: 0
```

自己署名された証明書を作成する際に、外部認証要求の作成時に要求された情報と同じものを入力するように指示されます。さらに、証明書のバージョン番号、その証明書をキー・データベースにデフォルトとして設定するか、および証明書を別のファイルに保管するか (加えて、キー・データベースに保管するかどうか) などの情報を入力するようにも指示されます。バージョン番号は X.509 標準のバージョン番号を表します。バージョン 3 を選択してください。証明書をデフォルト証明書として設定すると、System SSL API 呼び出しで識別名を指定しなくても、SSL Toolkit はその証明書を使用できます。証明書をファイルに保管することを選択した場合、ファイル形式に関する追加情報を入力するように指示されます。 ASCII を選択すると、デフォルト名が cert.arm のファイルが作成され、現行作業ディレクトリーに置かれます。このファイルの形式は以下のとおりです。

```
$ cat cert.arm
-----BEGIN CERTIFICATE-----
MIIBezCCASWgAwIBAgIJ+fHy9/n19/LwMA0GCSqGSIb3DQEBBQUAMEIxCzAJBgNV
BAYTAlVTMQwwCgYDVQQKEwNJQk0xETAPBgNVBAsTCEVuZGljb3R0MRIwEAYDVQQD
EwlUaW0gSm9uZXMwHhcNOTgxMjAzMTgyMjAwWhcNOTkwODA1MTgyMjAwWjBCMQsw
CQYDVQQGEwJVUzEMMAoGA1UEChMDSUJNMREwDwYDVQQLEwhFbmRpY290dDESMBAG
A1UEAxMJVGltIEpvbmVzMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBALk1wGHpv09T
kxK1jbGfaonWM08vmJLseRm6s2ZKz81oWvOKz3eW+bsbkt8uLzS7LWEXGprvpiSi
sem6We81zxMCAwEAATANBgkqhkiG9w0BAQUFAANBACism2QymyCRZBg+oDFA8jxZ
WABC+S4kDPMkw2HcoO645PVcLstWBVol14MOZMKjLwwS291BsX1lblwWxKCI/ME=
-----END CERTIFICATE-----
$
```

バイナリー・データ形式を選択すると、DER エンコード証明書がファイルに直接保管されます。このファイルのデフォルト名は cert.crt で、現行作業ディレクトリーにあります。

クライアント・プログラムへの証明書のエクスポート

自己署名された証明書を使用する場合、ご使用の SSL サーバーに接続しているすべての SSL クライアントは、自己署名された証明書を「CA 証明書」と見なすことができるはずです。そのため、自己署名された証明書は、すべての SSL クライアント・プログラムに送信する必要があります。

自己署名された証明書を SSL クライアント・プログラムのキー・ファイルに送る最初のステップは、その証明書を SSL サーバーのキー・データベースから、他のプログラムに送信できるファイルにエクスポートすることです。

キー・データベース内の証明書をファイルにエクスポートするには、以下のようにします。

1. 「Key database」メニューで、1 (List/Manage keys and certificates) を入力します。鍵および証明書のリストが表示されます。
2. 証明書の作成時に選択したラベル (この例では、tjh2) を見つけて、そのラベルに関連する鍵番号を入力します。
3. 次に、「キー (Key)」メニューで、5 (Copy the certificate of this key to a file) を選択します。
4. 以下のようにプロンプトに応答します。

```
Should the certificate binary data or Base64 encoded ASCII data be saved? (1 =
ASCII, 2 = binary) [1]:1
Enter certificate file name or press ENTER for "cert.arm":

Please wait while the certificate is written to a file...

Your request has completed successfully, exit gskkyman?
(1 = yes, 0 = no) [0]: 1
```

エクスポートする証明書にオプション 1 (ASCII) を選択した場合、デフォルトのファイル名は cert.arm になり、結果としてファイルは Base64 エンコード形式になります。オプション 2 (binary) を選択した場合、デフォルトのファイル名は cert.crt になり、結果としてファイルはバイナリーの DER エンコード形式になります。これで、このファイルを SSL プログラムが実行するシステムに転送し、証明書を SSL クライアント・プログラムのキー・ファイルにトラステッド CA 証明書としてインポートすることができます。

クライアント証明書の公開

CICS Transaction Gateway では、 SSL と HTTPS の両方のプロトコルのクライアント認証をサポートしています。クライアント認証を使用可能にするには、構成ツールで「クライアント認証の使用」設定を選択しなければなりません。

クライアント認証が使用可能になると、接続先のクライアントは自分のクライアント証明書を提供するように要求されます。サーバー (Gateway) はその妥当性検査を行って、ユーザー出口で公開します。

証明書を公開すると、サーバーは証明書の内容を問い合わせ、指定された認証規則に従って接続を拒否することもあります。

System SSL

クライアント証明書は SystemSSLServerSecurity ユーザー出口で公開されます。必要に応じて、afterDecode() メソッドで IOException をスローすることによって接続が拒否されます。

サーバー側のセキュリティー・クラスは、com.ibm.ctg.security.SystemSSLServerSecurity インターフェースをインプリメントしなければなりません。 afterDecode() メソッドでは、com.ibm.gkssl.SSLCertificate 証明書オブジェクトとともに、GatewayRequest オブジェクトを公開します。

クライアント側のセキュリティー・クラスは、com.ibm.ctg.security.ClientSecurity インターフェースをインプリメントしなければなりません。このインターフェースと SystemSSLServerSecurity インターフェースの詳細については、「*CICS Transaction Gateway: Gateway プログラミング*」を参照してください。

サンプルの SystemSSLServerCompression.java が /samples/java/com/ibm/ctg/security で提供されています。これはクライアント証明書の公開方法を説明しています。

SSLight

クライアント証明書は SSLLightServerSecurity ユーザー出口で公開されます。必要に応じて、afterDecode() メソッドで IOException をスローすることによって接続が拒否されます。

サーバー側のセキュリティー・クラスは、com.ibm.ctg.security.SSLightServerSecurity インターフェースをインプリメントし

なければなりません。 afterDecode() メソッドでは、com.ibm.sslight.SSLCert[] 証明書チェーン・オブジェクトとともに、GatewayRequest オブジェクトを公開します。

クライアント側のセキュリティー・クラスは、com.ibm.ctg.security.ClientSecurity インターフェースをインプリメントしなければなりません。このインターフェースと SSLightServerSecurity インターフェースの詳細については、「*CICS Transaction Gateway: Gateway プログラミング*」を参照してください。

サンプルの SSLightServerCompression.java が /samples/java/com/ibm/ctg/security で提供されています。これはクライアント証明書の公開方法を説明しています。

RACF ユーザー ID へのクライアント証明書のマッピング

TSO で RACDCERT コマンドを実行することにより、クライアント証明書を RACF ユーザー ID に関連付けることができます。 (このコマンドは、UNIX System Services シェルの下では実行されません。)

58ページの『RACF® で使用するための CICS Transaction Gateway の構成』の説明に従って、すでにこの処置は実行しているはずです。

RACDCERT を実行する前に、処理したい証明書を TSO からアクセスできる MVS 順次ファイルに (RECFM=VB を指定して) ダウンロードする必要があります。 RACDCERT の構文は次のとおりです。

```
RACDCERT ADD('datasetname') TRUST [ ID(userid) ]
```

ここで、*datasetname* はクライアント証明書を含むデータ・セットの名前です。また、*userid* は証明書に関連付けるユーザー ID です。オプションの ID(*userid*) パラメーターを省略する場合、証明書は RACDCERT コマンドを発行するユーザーに関連付けられます。

RACDCERT コマンドを発行したら、RACF は証明書とユーザー ID との関連を作成するプロファイルを DIGTCERT クラスに作成します。このプロファイルは、パスワードを指定しなくても証明書をユーザー ID に変換するために使用できます。

RACDCERT コマンド (ダウンロードした証明書データ・セットで使用できるデータの形式を含む) の詳細については、「*OS/390 Security Server (RACF) Command Language Reference*」を参照してください。

CICS Transaction Gateway では、X.509 証明書を RACF ユーザー ID にマップするために com.ibm.ctg.util.RACFUserid クラスを提供しています。このクラスには以下のメソッドがあります。

- **setCertificate(byte[] clientCertificateData);**
- **getRACFUserid();**

ユーザーは以下の 2 つの方法で RACFUserid オブジェクトを作成できます。

1. `RACFUserid myUseridObject = new RACFUserid();`
2. `RACFUserid myUseridObject = new RACFUserid(myCertificate);`

2 番目の方法では、オブジェクトを構成してから、そこに自動的に証明書データを入れるので、**setCertificate(byte[] clientCertificateData);** メソッドを呼び出す必要がありません。そのため、この 2 番目の方法が優先されます。オブジェクトが作成されると、**getRACFUserid()** は固有の OS/390 呼び出しを行って、提供された証明書を RACF ユーザー ID にマップしようとします。 (このとき、ユーザーが X.509 証明書と RACF ユーザー ID の間にすでにリンクを確立していることを前提としています。)

getRACFUserid() メソッドでは、 RACF ユーザー ID を含むストリングを戻します。

samples/java/com/ibm/ctg/security ディレクトリーの SystemSSLServerCompression.java クラスには、 RACFUserid オブジェクトの作成例があります。

com.ibm.ctg.util.RACFUserid クラスの詳細については、HTML プログラミングの参照情報をご覧ください。プログラミング例については、「*CICS Transaction Gateway: Gateway プログラミング*」を参照してください。

SSL および HTTPS 用の CICS® Transaction Gateway (OS/390® 版) の構成

SSL プロトコルおよび 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しなければなりません (47ページの『構成ツールの使用』を参照してください)。以下のものを使用可能にすることができます。

SSL SSLight

System SSL
System SSL

HTTPS
HTTP over SSLight

System HTTPS
HTTP over System SSL

これで、CTG.INI ファイルに正しいハンドラー定義が作成されます。

CICS Transaction Gateway が実行されると、プロトコルが開始されます。SSL か HTTPS のどちらかまたは両方を使用することを指定できます。さらに、SSLight と System SSL を同時に使用することもできます。 CICS Transaction Gateway が開始されると、ハンドラーが使用可能になっている限り、Gateway はポート 8050 で SSL 要求を listen し、ポート 443 で HTTPS 要求を listen します。ただし、SSLight ハンドラーと System SSL ハンドラーを同時に使用する場合は、固有のポート番号が指定されていることを確認してください。

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 データベース
System SSL サーバーのキー・データベースの名前を指定します。

KeyRing クラス名
SSLight の場合、SSL サーバーの KeyRing クラス・ファイルの名前を指定します。 CLASSPATH は、このクラスが見つかるように設定しなければなりません。

KeyRing パスワード
暗号化されたサーバーの KeyRing を復号するために使用するパスワードを指定します。

クライアント認証の使用
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 プロトコルおよび HTTPS プロトコルの項目の完全な例は、サンプル構成ファイル CTGSAMP.INI に入っています。

以下は、クライアント認証が使用可能になっている SSLight の、CTG.INI にあるプロトコル・ハンドラー項目の例です。

```
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;
```

以下の例は、System SSL の場合です。

```
protocol@systemssl_ssl.handler=com.ibm.ctg.server.GskSslHandler
protocol@systemssl_ssl.parameters=port=8050;sotimeout=1000;connecttimeout=2000;
idletimeout=600000;pingfrequency=60000;keyring=key.kdb;keyringpw=password;
clientauth=on;
```

SSLight の場合、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・ファイルが提供されています。 **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にすることをお勧めします。したがって、SSL および HTTPS プロトコルを使用するには、前の節の説明に従って、独自の KeyRing を生成してください。

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、クライアントの 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機密保護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保護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 クライアントの両方の KeyRing が必要です。

クライアントの 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 は以下に示す 2 つのメソッドを提供しています。 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 証明書を復号する場合に使用します。

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およびインプリメンテーションについては、「*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。

その他のセキュリティーについての考慮事項

その他のセキュリティーに関する考慮事項については、それぞれ以下を参照してください。

- AUTH_USERID_PASSWORD 環境変数。 42ページの『環境変数』を参照してください。
- HFS ファイルでの拡張属性の設定。58ページの『RACF® で使用するための CICS Transaction Gateway の構成』を参照してください。
- EXCI オプション・モジュールの DFHXCOPT での代理オプションの設定。45ページの『EXCI オプションのカスタマイズ』を参照してください。
- ECI/EXCI 呼び出しの CICS セキュリティー。「*CICS Transaction Gateway: Gateway プログラミング*」を参照してください。

第7章 CICS® Transaction Gateway (OS/390® 版) の操作

この章では、CICS® Transaction Gateway (OS/390® 版) の開始方法と停止方法について説明します。ユーザーが指定することのできる始動オプションについて説明します。

CICS Transaction Gateway の開始

次のようにして Gateway を開始することができます。

- OS/390 UNIX® System Services コマンド行から
- バッチ・ジョブを開始するために JCL をサブミットして

Gateway は、オプションにデフォルト値を指定して、あるいはユーザーが値を提供して、開始することができます。オプションとその値については、『ユーザー指定オプションでの Gateway の開始』で説明しています。

コマンド行からの開始

デフォルト・オプションを指定して CICS Transaction Gateway を開始するには、コマンド・プロンプトで `ctgstart` と入力して、Enter (実行) キーを押します。始動メッセージが表示されます。

```
CCL6500I: CICS Transaction Gateway をデフォルト値で始動します。
```

この後に、次のように、使用されている値を示す行が 2 行続きます。

```
CCL6502I: [ 初期 ConnectionManagers = 1, 最大 ConnectionManagers = 100,
CCL6502I: 初期 Workers = 1, 最大 Workers = 100, ポート = 2006 ]
```

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*
: CICS Transaction Gateway に割り当てられる TCP/IP ポート番号。

-initconnect=*number*
: ConnectionManager スレッドの初期数。

-maxconnect=*number*
: ConnectionManager スレッドの最大数。この値が -1 に設定されると、ConnectionManager スレッドの数に制限が適用されません。

-truncationsize=*nnn*
トレースで示されるデータ・ブロックの最大サイズを指定します。ここで、*nnn* は任意の正の整数です。 -trace または -X のいずれかに加えてこのオプションを使用できますが、それらのオプションで設定されたデフォルトのサイズはこのオプションによりオーバーライドされます。たとえば、標準トレースをオンにし、最大の 20,000 バイトをダンプするときは、以下のように指定します。

```
ctgstart -trace -truncationsize=20000
```

値 0 を指定すると、トレースにデータ・ブロックが表示されません。

-dumpoffset=*offset*
データ・ブロックの表示が開始されるオフセットを指定します。指定されたオフセットがデータの全長よりも大きい場合は、データは、オフセットが 0 であるかのようにダンプされます。

-initworker=*number*
Worker スレッドの初期数。

-maxworker=*number*
Worker スレッドの最大数。この値が -1 に設定されると、Worker スレッドの数に制限が適用されません。

-trace
これは、標準トレース・オプションです (108ページの『CICS Transaction Gateway でのトレース』を参照してください)。デフォルトでは、COMMAREA の最初の 80 バイトだけがトレースされます。

ご使用の Gateway を構成するの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。それ以外の場合、トレース出力は STDERR に書き込まれます。

-notime
メッセージ内のタイミング情報を (時刻はミリ秒単位で正確に示されます) 使用不可にします。

-stack
例外スタック・トレースのみを使用可能にします。 CICS Transaction Gateway の通常操作時に予想される例外を含め、ほとんどの Java 例外がトレースされますが、他のトレースは実行されません。

-tfile=*pathname*
トレースが使用可能になっている場合は、トレース・メッセージを

pathname で指定されたファイルに書き込みます。これにより、トレース出力のデフォルト宛先がオーバーライドされます (**-trace** オプションを参照してください)。

-tfilesize=*number*
トレース・テキスト・ファイルの最大サイズを K バイト単位で指定します。

-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。この場合、コンソール・セッションへの入力によって Gateway を停止することはできません。103ページの『CICS Transaction Gateway の停止』を参照してください。

-x 完全なデバッグ・トレースを使用可能にします。これには、-trace オプションでトレースされるすべての情報と、 CICS Transaction Gateway の内部作業についての情報を含む追加情報が含まれます。このオプションのデフォルトでは、COMMAREA 全体をトレースします。

このオプションを指定すると、パフォーマンスが大幅に低下します。

-dnsnames
メッセージでの TCP/IP の記号ホスト名の表示を使用可能にします。

始動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、その後に必要な始動オプションを入力して、Enter (実行) キーを押します。

始動メッセージが表示されます。

```
CCL6501I: CICS Transaction Gateway を指定の値で始動します。
```

この後に、次のように、使用されている値を示す行が 2 行続きます。

```
CCL6502I: [ 初期 ConnectionManagers = 10, 最大 ConnectionManagers = 100,
CCL6502I: 初期 Workers = 10, 最大 Workers = 100, ポート = 2006 ]
```

始動オプションに関するヘルプを表示するには、`ctgstart ?` と入力します。

トレースに関連する始動オプションの詳細については、 108ページの『CICS Transaction Gateway でのトレース』を参照してください。

JCL での開始

次に、バッチ・ジョブとして CICS Transaction Gateway を開始するために使用できる JCL の例を示します。

```
//DFHJGATE JOB (Accounting info),CLASS=A,USER=user,PASSWORD=passwd,
//         MSGCLASS=H
//OEEXCI   EXEC PGM=BPXBATCH,
//           PARM='SH /usr/lpp/ctg/bin/ctgstart -noinput',
//           REGION=8M
//STDIN  DD PATH='/dev/null',
//          PATHOPTS=(ORDONLY)
//STDOUT DD PATH='/jgateo.log',PATHOPTS=(OWRONLY,OCREAT),
//          PATHMODE=SIRWXU
//STDERR DD PATH='/jgatee.log',PATHOPTS=(OWRONLY,OCREAT),
//          PATHMODE=SIRWXU
//STDENV DD *
DFHJVPIPE=JAVAGAT1
DFHJVSYSTEM_00=IJKLMNOP-Primary CICS server
DFHJVSYSTEM_01=OTHER-Some other CICS system
/*
//
```

この例で、

- BPXBATCH は、バッチ・ジョブとして OS/390 UNIX® System Services スクリプトを実行する MVS プログラムです。 PARM フィールドは、シェル (SH) が、-noinput オプションを指定した ctgstart コマンドを実行することを指定します。パスは、最高位の /ctg ディレクトリーが HFS ルートから直接アクセス可能であると想定しています。
- STDENV は、任意の環境変数で値を設定することができるデータ・セットです。この例では、DFHJVPIPE、DFHJVSYSTEM_00、および DFHJVSYSTEM_01 環境変数が設定されています (詳細については、41ページの『環境変数の設定』を参照してください)。環境変数に指定する値を引用符で囲まないでください。行に順序番号がある場合、その順序番号は環境変数の値の一部として解釈されます。 JCL で指定された値は、ctgstart スクリプトまたは ctgenvvar スクリプトで指定された値によってオーバーライドされます。

注: CICS Transaction Gateway が JCL によって開始されると、指定されたジョブのほかに、その他のジョブもいくつか開始されます。たとえば、ジョブ名 CICSTGQ の場合、CICSTGQ1 から CICSTGQ*n* までの名前が付けられたジョブもいくつか開始されます。

領域サイズについての考慮事項

JCL の EXEC カードの REGION パラメーターは、 CICS Transaction Gateway で実行するために必要なスレッド数に従って設定する必要があります。

以下のパラメーターも、REGION パラメーターに選択できる値に影響を与える可能性があります。

- RACF ユーザー ID の OMVS セグメントの ASSIZEMAX パラメーター。詳細は、「*RACF Command Language Reference*」を参照してください。
- SYS1.PARMLIB(BPXPRMxx) で検出される UNIX System Services MAXASSIZE パラメーター。詳細は、「*UNIX System Services Planning*」を参照してください。

ASSIZEMAX に指定した値によって、 MAXASSIZE パラメーターで提供される値がすべてオーバーライドされることに注意してください。

注: REGION=0M と設定すると、java.lang.OutOfMemoryError が発生することがあります。 114ページの『java.lang.OutOfMemory 例外』を参照してください。

複数の CICS Transaction Gateway の開始

複数の CICS Transaction Gateway を開始するには、以下の例のように、開始済みタスクを使用します。

```
CTGST
//CTG24   JOB CLASS=A
/*JOBPARM SYSAFF=P24
//        COMMAND 'S CTG,JOBNAME=CTGT2401,ID=01,SYSID=T24A'
//        COMMAND 'S CTG,JOBNAME=CTGT2402,ID=02,SYSID=T24A'
//        COMMAND 'S CTG,JOBNAME=CTGT2403,ID=03,SYSID=T24B'
//        COMMAND 'S CTG,JOBNAME=CTGT2404,ID=04,SYSID=T24B'
//        EXEC PGM=IEFBR14
```

これにより、それぞれの CICS Transaction Gateway ごとに以下の JCL が開始されます。

```
CTG
//CTG   PROC REG='0M',ID='01',SYSID='T24A'
//*
//*    REG      - STORAGE REQUIRED
//*    ID       - UNIQUE ID FOR THIS INSTANCE OF THE CTG
//*
//*************************************************************
//*******************  EXECUTE CICS  ************************
//*************************************************************
//CTG      EXEC PGM=BPXBATCH,
//         PARM='SH /usr/lpp/ctg/bin/ctgstart -noinput',
//         REGION=&REG.
//STDIN    DD PATH='/dev/null',
//         PATHOPTS=(ORDONLY)
//STDOUT   DD PATH='/usr/lpp/ctg/bin/ctg&SYSID.&ID.o.log',
//          PATHOPTS=(OWRONLY,OCREAT),
//          PATHMODE=(SIRWXU,SIRWXG,SIRWXO)
//*STDERR   DD PATH='/dev/null',
//*          PATHOPTS=(OWRONLY)
//STDERR   DD PATH='/usr/lpp/ctg/bin/ctg&SYSID&ID.e.log',
```

```
//          PATHOPTS=(OWRONLY,OCREAT),
//          PATHMODE=(SIRWXU,SIRWXG,SIRWXO)
//STDENV    DD DSN=PEL.TAU.PARMLIB(CTG&SYSID.),DISP=SHR
//*
```

単一の CICS Transaction Gateway の場合の JCL に比較して、 PATHMODE および PATHOPTS の変更点に注意してください。これらがないと、スーパーユーザー権限をもたないユーザーはファイルを読み取ることができません。また、STDERR を /dev/null にコーディングすると、接続 / 切断トレース項目を停止できない場合に、トレース・ファイルが書き込まれません。この理由から、 STDERR について 2 つのオプションがコーディングされています。

また、上記の JCL では、ターゲットの CICS 領域に固有な STDENV データ・セットも使用しています (この STEPLIB 定義を実行するために、STEPLIB を変更 / 上書きしないように ctgstart を変更する必要もあります)。

```
PEL.TAU.PARMLIB(CTGT24A)
DFHJVPIPE=BATCHCLI
DFHJVSYSTEM_00=CICST24A
DFHJVSYSTEM_01=CICST24B
DFHJVSYSTEM_02=CICST25A
DFHJVSYSTEM_03=CICST25B
STEPLIB=PEL.TAU.CICS.CICST24A.SDFHEXCI:CICSTS13.SDFHEXCI:CICSTS13.SDFHLOAD
```

着信要求でクライアントが特定の CICS APPLID を指定しなくてもよいようにするため、サンプル内の分岐を除去して DFHXCURM サンプルを変更し、それに CICS 領域のリストを与えるようにコーディングします。それぞれの CICS 領域ごとに 1 つの DFHXCURM 出口が必要です。ユーザーがこの出口の名前を変更することはできないので、この出口を固有の区分データ・セットに入れ、それらのデータ・セットが最初に (上記のファイル内の) STEPLIB 環境変数に置かれていることを確認します。

共用されている 1 つのポートに送信する場合に、コンソール入力によって OMVS から複数の CICS Transaction Gateway を開始するには、TCP/IP プロファイルで共用ポートへのアクセスをもつものとしてジョブ名 ″OMVS″ を定義してください。コンソール入力が必要でない場合は、JCL によって CICS Transaction Gateway を開始し、 TCP/IP プロファイルで JCL ジョブ名を定義してください。

CICS Transaction Gateway の停止

CICS Transaction Gateway をコマンド行から開始した場合、および -noinput パラメーターを指定しなかった場合は、 Gateway コンソール・セッションで Q を入力してから、 Enter (実行) キーを押すと、Gateway を停止することができます。

-noinput パラメーターを使用していた場合、または CICS Transaction Gateway を開始するために JCL を使用した場合は、 JES CANCEL コマンドを使用して Gateway プロセスを停止しなければなりません。

注: メイン CICS Transaction Gateway ジョブを取り消すと、他のすべてのジョブも取り消されます。たとえば、ジョブ名 CICSTGQ の場合、 CICSTGQ1 から CICSTGQ*n* までの名前が付けられたジョブも取り消されます。

第8章 CICS® Transaction Gateway (OS/390® 版) の問題判別

問題判別は問題解決とは異なるものですが、問題を調べていけば、問題を解決するための十分な情報を見つけることができます。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

問題を調べる前に、明白な原因があるかどうか調べるのがよいでしょう。

- 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。
- CICS® Transaction Gateway (OS/390® 版) バージョン 3.0 から移行した場合、構成ファイル (Gateway.properties)、および JGate スクリプトが正しく移行しましたか。
- CLASSPATH や LIBPATH のような環境変数は正しく設定されていますか。(41ページの『環境変数の設定』を参照してください。)

  CLASSPATH 変数は、CICS Transaction Gateway トレース出力の最上部に表示されます。トレースを使用可能にした Gateway の開始については、108ページの『CICS Transaction Gateway でのトレース』を参照してください。
- 障害を説明する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。

JDK AppletViewer を使用してサンプル・アプレットを実行した場合の問題

AppletViewer を使用してユーザーのローカル・ファイル・システムから CICS Transaction Gateway のサンプル・アプレットのいずれかを実行したときに、メッセージ CCL6664E「*xyz* プロトコルをサポートするために関係のあるクラスをロードできません」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・バーで「**アプレット (Applet)**」を選択してから、「**特性 (Properties)**」を選択します。次に、「**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「非制限 (Unrestricted)」に設定します。

デフォルトのポートとの競合

CICS Transaction Gateway は、サポートされるプロトコルについてデフォルトのポートを使用します。これらのポートがすでに使用中のポートと競合することがあり、この場合、1 つまたは複数のプロトコルが正常に開始しません。CTG.INI ファイルでポート番号を変更することができます。 47ページの『構成ツールの使用』を参照してください。

CICS サーバー上のリソースの不足

CICS サーバーに対する ECI 呼び出しから、断続的な ECI_ERR_RESOURCE_SHORTAGE 戻りコードを受け取ることがあります。その場合には、CICS サーバーの SESSIONS リソース定義の RECEIVECOUNT 値を、理論的に必要な値よりも十分に大きい値、つまり 100 に増やす必要があります。詳細については、37ページの『CICS Transaction Gateway のための CICS 定義』を参照してください。

次に行うこと

CICS Transaction Gateway に問題があると考えられる場合は、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。

trace を使用可能にせずに Gateway を開始した場合は、 Gateway を停止して、**trace** オプションを選択した状態で再始動し、問題を再成させてください。

問題がネットワーク内のどこかにあると考えられる場合は、他の製品に添付されている問題判別手順に従ってください。「*CICS Problem Determination Guide*」を参照してください。

プログラム・サポート

各種レベルのプログラム・サポートが利用可能であるため、IBM に連絡する前に、どのレベルを使用しているか確認してください。保証およびサポート情報は、製品に付属のライセンス 文書に記載されています。製品によっては、**サービスおよびサポート**・カードが含まれているものもあります。あるいは、次の Web サイトを調べることもできます。

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

Support リンクに進んでください。

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 プログラミング*」を参照してください。

情報のソース

問題判別情報は、次のソースから入手することができます。

- 標準出力 (STDOUT) ファイルおよび標準エラー (STDERR) ファイル。
  - 標準出力には、CICS Transaction Gateway メッセージ、および CICS から EXCI に戻されたメッセージのログが含まれています。
  - 標準エラーには、標準出力のメッセージが、Java 仮想マシンからのエラー・メッセージと一緒に含まれています。 Java 仮想マシンのメッセージについては、Java 開発ツールキット (JDK) で説明しています。
- CICS Transaction Gateway トレース。以下の節を参照してください。
- EXCI トレース: EXCI 活動を表します。 CICS Transaction Gateway トレースは、EXCI トレースの一部です。トレース・ポイントは、 111ページの表4 にリストされています。 EXCI トレースの詳細については、「*CICS 外部インターフェース・ガイド*」を参照してください。
- CICS トレース。以下を介して要求の進行を表します。
  - EXCI 要求を処理する CICS ミラー・トランザクション。
  - 呼び出し先 CICS プログラム。

  CICS トレースで AP トレース・ポイントを検査してください。

メッセージおよびトレース情報をファイルに転送する場合は、 Gateway の開始時に以下を指定します。

```
ctgstart -trace -tfile=pathname
```

これにより、*pathname* ディレクトリーにファイルが生成されます。

完全なデバッグ・トレースを作成するために、 `ctgstart -x` オプションを使用できます。

CICS Transaction Gateway でのトレース

Gateway およびアプリケーションのトレースには以下の 3 つの主要なレベルがあります。

スタック・トレース
このトレース項目は、Java 例外が起こったときにのみ書き込まれます。これらの項目は、例外のソースを判別するのに役立ちます。パフォーマンスを保つことが重要な場合、このトレースを使用してください。

標準トレース
Java 例外およびメインの Gateway 機能および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、またはネットワーク・フローなど) 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
Java 例外およびメインの Gateway 機能およびイベントが、スタック・トレースや標準トレースよりも詳細にトレースされます。デフォルトでは、 Gateway はデータ・ブロックの全体をトレースに表示します。パフォーマンスがそれほど重要でないときや、問題を解決するのに十分な情報が標準トレースによって得られない場合にのみ、このトレースを使用してください。

Gateway のトレース

CICS Transaction Gateway プロセス、または Gateway への呼び出しを行うユーザー・アプリケーションで、トレースを開始することができます。

ctgstart コマンドのオプションを使用して、 Gateway でのトレースを使用可能にし、制御します。たとえば、標準トレースを使用可能にするには、Gateway を開始するときに次のコマンドを入力します。

```
ctgstart -trace
```

Gateway を構成するのに構成ツールを使用した場合 (または、 CTG.INI ファイルで TFILE パラメーターを設定した場合)、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。それ以外の場合、トレース出力は、`stderr` に

書き込まれます。 **ctgstart -tfile** オプションを使用すると、トレース出力のデフォルト宛先をオーバーライドすることができます。たとえば、次のコマンドは

```
ctgstart -x -tfile pathname
```

デバッグ・トレースを使用可能にして Gateway を開始し、トレース出力を、*pathname* で指定したファイルに書き込みます。

ctgstart で指定できるオプションは以下のとおりです。

-trace
標準トレースを使用可能にします。デフォルトでは、COMMAREA の最初の 80 バイトだけがトレースされます。

-stack
例外スタック・トレースのみを使用可能にします。 CICS Transaction Gateway の通常操作時に予想される例外を含め、ほとんどの Java 例外がトレースされますが、他のトレースは実行されません。

-x 完全なデバッグ・トレースを使用可能にします。これには、-trace オプションでトレースされるすべての情報と、 CICS Transaction Gateway の内部作業についての情報を含む追加情報が含まれます。このオプションのデフォルトでは、COMMAREA 全体をトレースします。

このオプションを指定すると、パフォーマンスが大幅に低下します。

-tfile=*pathname*
トレースが使用可能になっている場合は、トレース・メッセージを *pathname* で指定されたファイルに書き込みます。これにより、トレース出力のデフォルト宛先がオーバーライドされます (**-trace** オプションを参照してください)。

-tfilesize=*number*
トレース・ファイルの最大サイズを K バイト単位で指定します。

-truncationsize=*nnn*
トレースで示されるデータ・ブロックの最大サイズを指定します。ここで、*nnn* は任意の正の整数です。 -trace または -X のいずれかに加えてこのオプションを使用できますが、それらのオプションで設定されたデフォルトのサイズはこのオプションによりオーバーライドされます。たとえば、標準トレースをオンにし、最大の 20,000 バイトをダンプするときは、以下のように指定します。

```
ctgstart -trace -truncationsize=20000
```

値 0 を指定すると、トレースにデータ・ブロックが表示されません。

-dumpoffset=*offset*
データ・ブロックの表示が開始されるオフセットを指定します。指定されたオフセットがデータの全長よりも大きい場合は、データは、オフセットが 0 であるかのようにダンプされます。

パフォーマンスについての考慮事項

トレース、特にデバッグ・トレースでは、パフォーマンスが低下します。パフォーマンスへのトレースの影響を小さくするため、 CICS Transaction Gateway バージョン 4.0 では次のような変更が加えられました。

- **-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、トレース出力は、`stderr` へも書き込まれません。
- トレース出力のデータ・ブロックの最初の行だけにタイム・スタンプが付けられます。
- **-truncationsize** オプションと **-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できます。

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方法は 2 つあります。つまり、JVM を開始するときに Java ディレクティブを使用する方法と、 CICS Transaction Gateway トレース API に呼び出しを追加する方法です。 Java ディレクティブを指定するには、**Java** コマンドで **-D** オプションを使用します。トレース API は、 CICS Transaction Gateway の T クラスの中にいくつかの静的メソッドからなります。詳細は、「*CICS Transaction Gateway: Gateway プログラミング*」を参照してください。

JNI のトレース

JNI トレースは、通常の CICS Transaction Gateway トレース・オプションでは使用可能になりません。使用可能にするには、環境変数 CTG_JNI_TRACE を、トレース出力を書き込むパスおよびファイルに設定します。以下に、例を示します。

```
export CTG_JNI_TRACE=/tmp/ctgjni.trc
```

WebSphere Application Server の下のサーブレットからローカル・モード接続の JNI トレースを取得するには、以下を行います。

1. httpd.envvars ファイルに CTG_JNI_TRACE=*filename* を設定します。

2. IHS JCL を変更して、言語環境プログラム (LE) に対するパラメーターを次のように指定します。

   ```
   LEPARM='ENVAR("_CEE_ENVFILE=/your/httpd.envvars")'
   ```

3. WebSphere Application Server が実行されている ユーザー ID が、トレース出力ファイルへの書き込みアクセス権を持っていることを確認します。

詳細は、WebSphere Application Server の資料を参照してください。

EXCI のトレース

CICS Transaction Gateway は EXCI トレースにトレース項目を書き込みます。トレース項目は CICS トレース EXCI 形式なので、ダンプ内のトレース項目は標準 OS/390 ユーティリティーを使用して印刷されます。 SDSF オペレーター・コンソールのコマンド行から以下の操作手順を使用することができます。

1. 次のコマンドを使用します。

   ```
   /D OMVS,A=ALL
   ```

   これで、OMVS タスクが表示されます。

2. CICS Transaction Gateway タスクを見付け、ASID をメモします。
3. DUMP コマンドと適切なコメントを入力します。以下に、例を示します。

   ```
   /DUMP COMM=(JGATE DUMP)
   ```

4. 次のように、ASID 付きのメッセージに応答します。

   ```
   /R nn,ASID=aa,END
   ```

   ここで、*nn* は応答用のメッセージ番号で、*aa* は ASID です。

表4 は、CICS Transaction Gateway によって EXCI トレースに書き込まれるトレース・ポイントを表します。

表 4. CICS Transaction Gateway の EXCI トレース・ポイント

ポイント ID	モジュール	Lvl	タイプ	データ
8000	JVDLL	EX 1	渡される ECI パラメーター	**1** スレッド名 **2** Call_Type **3** Extend_Mode **4** Luw_Token **5** Commarea_Length **6** Cics_Rc **7** Message_Qualifier

表 4. CICS Transaction Gateway の EXCI トレース・ポイント (続き)

ポイント ID	モジュール	Lvl	タイプ	データ
8001	JVDLL	Exc	GetStringPlatform エラー	**1** 戻りコード **2** データ域 **3** 長さ
8002	JVDLL	Exc	GetStringPlatformLength エラー	**1** 戻りコード
8003	JVDLL	EX 1	変換された パラメーター	**1** スレッド名 **2** パラメーター名 **3** パラメーター値
8004	JVDLL	EX 1	インバウンド COMMAREA	**1** スレッド名 **2** 通信域の長さ **3** 通信域のアドレス **4** 250 バイトのデータ
8006	JVDLL	EX 1	アウトバウンド COMMAREA	**1** スレッド名 **2** 通信域の長さ **3** 通信域のアドレス **4** 250 バイトのデータ
8007	JVDLL	EX 1	ECI パラメーター出力	**1** スレッド名 **2** `Call_Type` **3** `Extend_Mode` **4** `Luw_Token` **5** `Commarea_Length` **6** `Cics_Rc` **7** `Message_Qualifier`
8010	JVDLL	Exc	受信されるエラー応答	**1** ファンクション番号 **2** EXCI 応答 **3** EXCI 理由 **4** EXCI 副次理由フィールド 1 **5** EXCI 副次理由フィールド 2 **6** `Cics_Rc`

表 4. CICS Transaction Gateway の EXCI トレース・ポイント (続き)

ポイント ID	モジュール	Lvl	タイプ	データ
8011	JVDLL	Exc	DPL_REQUEST エラー	**1** RESP **2** RESP2 **3** ABCODE **4** Cics_Rc
8012	JVDLL	Exc	WBA1 パラメーター割り振りエラー	**1** malloc 長 **2** Cics_Rc
8013	JVDLL	Exc	無効な呼び出しタイプ	**1** スレッド名 **2** Call_Type **3** Cics_Rc
8014	JVDLL	Exc	無効な COMMAREA 長	**1** スレッド名 **2** Commarea_Length **3** Commarea のサイズ **4** Cics_Rc

共通問題の診断

Java 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 スタック・ダンプを書き込みます。ダンプ出力の宛先は、ご使用の JVM インプリメンテーションに応じて異なります。詳細については、Java 資料を調べてください。

ジャストインタイム (JIT) コンパイラーを使用不可にして、 Java スタック・ダンプに書き込まれる情報を増やします。この情報には、例外が発生した Java ソース・コードの行も組み込まれる可能性があります。 JIT コンパイラーを使用不可にする方法は、ご使用の JVM インプリメンテーションに応じて異なります。詳細については、Java 資料を調べてください。

114ページの図12 は、JIT コンパイラーを使用不可にしてとられた Java スタック・ダンプのサンプルです。

```
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 12. Java スタック・ダンプのサンプル

CICS Transaction Gateway が例外を処理する場合は、トレースが使用可能になっている場合にのみ Java 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、問題の再現を試みてください。このことは、例外の発生前に何が起こっていたのかを理解するのに役立つはずです。トレースの詳細については、 108ページの『CICS Transaction Gateway でのトレース』を参照してください。

java.lang.OutOfMemory 例外

CICS Transaction Gateway が長時間実行された後にこの例外が発生した場合、メモリー・リークが原因と考えられます。Gateway の負荷が大きいときにのみこの例外が発生する場合は、使用可能な Java 仮想マシン (JVM) 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・リークがある場合

Gateway プロセスに対してメモリー使用モニターを実行します。メモリー・リークがある場合、Gateway で使用されるメモリーの量が時間の経過につれ増えていきます。 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、使用を完了したときに JavaGateway() 接続オブジェクトを必ずクローズするようにします。これでも問題が解決されないときは、Gateway デバッグ・トレースを実行して、IBM サポート組織に連絡をとってください。 **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れば、トレースで表示されるデータ・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、パフォーマンスへの影響が削減されます。

スレッドが多すぎる場合

このような症状は以下が原因と考えられます。

1. MAXTHREADS が BPXPRM*xx* 値に達した。解決法は、MAXTHREADS の値を増やすことです。詳細については、10ページの『CICS Transaction Gateway: スレッド化モデル』を参照してください。
2. CICS Transaction Gateway 始動 JCL での REGION=0M の使用。 0M が使用されるときに使用可能な実際の領域サイズは予測できず、 CICS Transaction Gateway で実行されているスレッド数に必要なサイズより小さ

いものと考えられます。解決法は、明示的な領域サイズを割り振ることです。この値は、 CICS Transaction Gateway で実行されるために必要なスレッド数に応じて異なります (100ページの『領域サイズについての考慮事項』を参照してください)。

3. JVM デフォルト・スタック・サイズ値として -ss (256 KB) および -oss (400 KB) の使用。これらのサイズの使用により、スレッドにつき 656 KB のメモリー割り振りとなります。 ctgstart スクリプトの `exec java` ステートメントにこれらの値があると、メモリー使用量が削減されます。

JVM で必要な仮想メモリーの量は以下のように見積もることができます。以下のすべてを加算してください。

- 接続マネージャー・スレッドの数
- Worker スレッドの数
- Java で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

次にこの合計を、JVM によって各スレッドに割り振られる Java スタックのサイズで乗算します。

この計算では、アクティブな Java バックグラウンド・スレッドの数がおよそ 5 個であるものと想定します。より正確な値については、ご使用の Java 資料を参照してください。CICS Transaction Gateway の構成で、接続マネージャー・スレッドおよび 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、49ページの『Gateway の一般設定』を参照してください。

Java がメモリーを割り振る方法は、ご使用の JVM 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 では以下の方法が可能です。

- **java -mx** オプションを使用して、 JVM に使用可能なメモリー量を設定する。
- **java -oss** オプションと **java -ss** オプションを使用して、Java が各スレッドに割り振るメモリー量を設定する。 Java スタックのサイズはこれら 2 つの値の合計です。

スレッドの制限についての詳細は、12ページの表1 を参照してください。 Java メモリー割り振りの詳細については、「*CICS Transaction Gateway: Gateway プログラミング*」およびご使用の Java 資料を参照してください。

OS/390 システムでは、各プロセスに使用可能なメモリーの量は制限される可能性があります。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。

プロセスがロックされた場合

ロックがどこで発生しているか判別するには、アプリケーション、および Gateway でトレースを使用可能にしてください。詳細については、108ページの『CICS Transaction Gateway でのトレース』を参照してください。Gateway 内にロックがある場合、 IBM サポート組織に連絡し、Gateway トレースを提供してください。

OS/390 JVM などの、一部の Java 仮想マシン (JVM) では、現行スレッドの状態を表示するスタック・トレースを強制的に Java 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 プロセスに送信して、スタック・トレースを `stderr` に書き込ませることができます。これにより、現行スレッドの状態が表示されます。この方法は、作業中の実動システムでは行わないでください。完全にロックされたシステムでのみ実行してください。

SSL 例外

一般

CICS Transaction Gateway でのスタック・トレースを使用可能にします。これにより、例外の発生時に何が起こっていたかか表示されます。また、CLASSPATH 変数の値など、構成についての情報も示されます。これによって、問題を診断するのに十分な情報が得られないときは、標準トレースを取得して、IBM サポート組織に連絡してください。

CICS Transaction Gateway の開始時

Gateway の開始時に、次のメッセージが出されることがあります。

```
CCL6525W: ssl: プロトコルのハンドラーが開始できません。
            [java.lang.ClassNotFoundException:
              server KeyRing class name]
```

このメッセージは、Gateway がサーバー KeyRing クラスをロードできなかったことを示しています。スタック・トレースを使用可能にして Gateway を開始します。トレース出力の最上部に表示される CLASSPATH 変数に、そのクラスを含むディレクトリーの項目が組み込まれているかチェックします。また、CLASSPATH に、ファイル `cfwk.zip` の項目が組み込まれているかチェックします。 Gateway は、このファイルを検出できないと、 KeyRing ファイルをロードできません。

構成ツールの SSL プロトコル設定パネルの「**キー・リング・クラス名 (KeyRing classname)**」フィールドに、サーバー KeyRing クラスの名前を指定します。 KeyRing クラス名を指定するとき、ディレクトリーまたはファイ

ル拡張子は組み込まないでください。CLASSPATH は、 KeyRing クラスを含むディレクトリーの項目を組み込みます。

ユーザー・アプリケーション内

API 呼び出し `SslJavaGateway.setKeyRing("ClientKeyRingName", "thePassword");` が、ユーザー・アプリケーション内で次のようなエラーによって失敗することがあります。

```
java.lang.NoClassDefFoundError: com/ibm/sslight/SSLightKeyRing
```

CLASSPATH 変数に、ファイル `cfwk.zip` の項目が組み込まれているかチェックします。このファイルは CICS Transaction Gateway と 一緒に提供されています。

ECI_ERR_SECURITY_ERROR 戻りコード

ECI 呼び出しからこの戻りコードが返される原因として以下のことが考えられます。

- EXCI オプション・テーブル DFHXCOPT で代理検査が使用可能になっている。

  DFHXCOPT テーブルの SURROGCHK オプションを使用すると、代理検査が使用可能になります。このオプションのデフォルトは YES です。このオプションの使用法についての詳細は、「*CICS 外部インターフェース・ガイド*」を参照してください。

- SDFHEXCI データ・セットに対して RACF プログラム制御がアクティブでなければならない。

  以下のようにプログラム制御をアクティブにします。

  ```
  SETROPTS CLASSACT(PROGRAM)
  RDEFINE PROGRAM * UACC(READ)
  SETROPTS WHEN(PROGRAM)
  ```

  以下のように、プログラム制御がアクティブなときに CICS ライブラリーを追加します。

  ```
  RALTER PROGRAM * ADDMEM*'hlq.SDFHEXCI'/volser/NOPADCHK)
  SETROPTS WHEN(PROGRAM) REFRESH
  ```

- 特定の HFS ファイルに拡張属性の設定値が指定されていない。

  以下の extattr コマンドは、 CICS Transaction Gateway が使用するロード・モジュールに、「プログラム制御」というマークを付けます。たとえば以下のとおりです。

  ```
  extattr +p path/ctg/bin/lib*.so
  extattr +p path/ctg/bin/SECURES
  ```

ここで、*path* は、たとえば、/usr/lpp です。

ctgstart スクリプトがアドレス・スペースを他のプロセスと共用**しない**ことを指定しなければなりません。これは、呼び出し側アドレス・スペース (JVM) が、プログラムで制御されていないロード・モジュールによって「誤用」されないようにするためのものです。 JVM 独自の非共用アドレス・スペースを使用させるには、次のように入力します。

```
extattr -s path/ctg/bin/ctgstart
```

ここで、*path* は、たとえば、/usr/lpp です。

特定の MVS Java ファイルもプログラム制御にする必要があります。

```
extattr +p javapath/bin/*
extattr +p javapath/lib/*
```

ここで、*javapath* は JVM のロケーションです。詳細については、 58ページの『RACF® で使用するための CICS Transaction Gateway の構成』を参照してください。

- ctgstart スクリプトが、'Share Address Space' 拡張属性ビットを、 extattr -s によってオフにしていない。これに当てはまらない場合、「ダーティー」アドレス・スペースが原因で、 ECI_SYNC または ECI_STATE-SYNC 呼び出しでエラーが発生しています。 119ページの『ダーティー・アドレス・スペースの問題』も参照してください。
- RACF ファシリティー・クラスが定義されていない。

  CICS Transaction Gateway そのものによって検査され、検査のために CICS に渡されることのないユーザー ID とパスワードを使用している場合は、RACF ファシリティー・クラスを定義する必要があります。ファシリティー・クラスの名前は、EXCI リンク用の CONNECTION 定義の netname パラメーターの値、および CICS Transaction Gateway で使用される DFHJVPIPE 環境変数の値に一致しなければなりません。詳細については、37ページの『CICS Transaction Gateway のための CICS 定義』を参照してください。
- JAVA_PROPAGATE 環境変数が、 local:// モードで実行される CICS Transaction Gateway アプリケーションに設定されていない。以下の変数を、

  ```
  JAVA_PROPAGATE=NO
  ```

  Java アプリケーションが実行される環境で設定する必要があります。

この環境変数が設定されていないと、 OS/390 トレースは、 CREATE_SECURITY_ENV パラメーターを指定した pthread_security_np 呼び出しが、 159 (EMVSERR) 戻りコードを返して失敗したことを表示します。

ECI_ERR_SYSTEM_ERROR 戻りコード

この戻りコードは、CICS アドレス・スペース内に問題があるために生成されることがあります。たとえば、PGMIDERR、セキュリティー違反、またはプログラム自動インストール拒否、などです。

また、CICS Transaction Gateway ホスト・アドレス・スペースのユーザー ID が、 DFHAPPL.*dfhjvpipe* または DFHAPPL.*applid* ファシリティーに対して PERMIT されていない場合、EXCI コードに生成されることもあります。

ダーティー・アドレス・スペースの問題

この問題があると、CICS Transaction Gateway がどのようにその特権状況を失うかによって、さまざまな結果となります。

- ctgstart スクリプトにまだ 's' ビットが設定されている。
- SDFHEXCI ロード・ライブラリーと、 EXCI_OPTIONS 環境変数に定義されているすべてのロード・ライブラリーがプログラム制御でない。
- CICS Transaction Gateway HFS が NOSETUID でマウントされている。 CICS Transaction Gateway HFS は、デフォルトの SETUID でマウントしなければなりません。

EDC5111I PERMISSION DENIED メッセージ

以下の理由でメッセージが生成されることがあります。

- CICS Transaction Gateway ポートがすでに別のタスクによって使用中である。
- ポートが、 TCPIP.PROFILE で CICS Transaction Gateway ジョブ用に予約されているが、 CICS Transaction Gateway ジョブ名が 8 文字未満である。この場合、ポートをオープンしようとするジョブは、 UNIX System Services によって割り振られる数字のサフィックスをもつジョブ名を持つことになり、TCPIP.PROFILE ジョブ名に一致しなくなります。この問題を解決するためには次のようにします。
  1. ポートを予約しない。
  2. 8 文字のジョブ名を使用する。
  3. TCPIP.PROFILE でポートに割り振られた ID のリストに 'OMVS' を追加する。 UNIX System Services によって追加される数字を ID として使用することはお勧めしません。

CEE3250C ABEND S806 メッセージ

システム・プロダクトおよびロード・ライブラリーにアクセスする十分な権限をユーザー ID がもっていない場合、Gateway の始動時に次のメッセージを受け取ることがあります。

```
CEE3250C The system or user abend S806  R=00000004 was issued.
         From entry point MVS_CcicsInit at compile unit offset -FFFF8D84 at address 1AEB158C.
```

STEPLIB 変数が正しい CICS EXCI ロード・ライブラリーを指していない場合にも、このメッセージが表示されます。詳細は、42ページの『環境変数』を参照してください。

EXCI パイプの制限

ローカル・モードでの CICS Transaction Gateway の実行時にサーブレットを使用する場合に、 EXCI パイプの制限によって問題が発生することがあります。

単一の OS/390 アドレス・スペースは、CICS 領域間通信 (IRC) によって、すべての接続されている CICS 領域に最大 100 個の EXCI パイプを割り振るように制限されます。それと対比して、CICS 領域での最大パイプ使用量は、CICS 複数領域操作 (MRO) セッション定義で定義されるセッション数 (最大 999 まで指定できます) によってのみ制限されます。そのため、サーブレットからの CICS Transaction Gateway による EXCI の使用には、以下の制限が適用されます。

- Web サーバー・ディレクティブ MaxActiveThreads は、処理できる並行要求の最大数を設定します。これは、ユーザーの最大数と同じではありません。OS/390 Web サーバーは、TCP/IP SOMAXCONN パラメーターまたは Web サーバー ListenBackLog ディレクティブの設定に従って、スレッドを使用する要求をキューに入れるからです。
- ECI 要求を処理する各スレッドは、1 つの EXCI パイプを割り振ります。このパイプは、障害が発生するか、CICS MRO 接続が終了するまで、割り振られたままになります。このようなパイプが、サーブレットに対する後続の要求で使用されます。ただし、異なる Web サーバー・スレッド間でパイプを共用することはできません。
- CICS Transaction Gateway を使用するスレッドの数が 100 を超えると、101 番目の EXCI Allocate_Pipe 呼び出しは、 SYSTEM_ERROR 応答コード、および理由コード 608 で失敗します。ECI アプリケーションは、戻りコード -9 (ECI_ERR_SYSTEM_ERROR) を受け取ります。これは、送信側のアドレス・スペースが限界に達したために、 Web サーバーで MaxActiveThreads を、100 より小か等しく構成する必要があることを意味します。

- CICS Transaction Gateway が、 CICS セッション定義で定義された使用可能なセッションよりも多くのパイプを割り振ろうとした場合、EXCI Open_Pipe 呼び出しは、RETRYABLE 応答コード、および理由コード 202 で失敗します。ECI アプリケーションは、戻りコード -16 (ECI_ERR_SYSTEM_ERROR) を受け取り、DFHXCURM が呼び出されます。この場合、CICS セッションの RECEIVECOUNT パラメーターを、少なくとも 100 に構成する必要があります。詳細については、37ページの『CICS Transaction Gateway のための CICS 定義』を参照してください。
- Web サーバーで 100 を超えるスレッドが必要であると分かった場合、TCP/IP ポート共用を使用して Web サーバー間の作業負荷のバランスを取るか、スケーラブル・モードで Web サーバーを使用することをお勧めします。 (それぞれのアドレス・スペースが、CICS Transaction Gateway で 100 個のパイプを使用できます)。

付録A. 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 (Windows® 版) 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 (AIX® 版) 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 (Solaris 版) 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 (Linux 版) 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 (HP-UX 版) 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 (OS/390® 版) 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・クライアントのマニュアルのソフトコピー・ファイル名を示しています。

表 5. CICS Transaction Gateway および CICS ユニバーサル・クライアントの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** リンクを参照してください。

その他の出版物

以下の International Technical Support Organization (ITSO) Redbook™ 資料には、クライアント / サーバー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*

  このレッドブックは、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・クライアントと一緒に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 (英語のみ) でアクセス可能です。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・クライアントのマニュアル。
- 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 の使用方法に関する情報もあります。

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、

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

にアクセスし、**Library** リンクに進んでください。

注: 一部のマニュアルは他の言語に翻訳されています。そ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 ファイルとして使用可能です。

PDF マニュアルの表示

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B. 特記事項

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

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

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

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

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

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

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

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

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

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

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

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

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

商標

以下は、IBM Corporation の商標です。

AIX	CICS
IBM	MVS
OpenEdition	OS/2
OS/390	System/390
TXseries	WebSphere

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]

[タ行]

[ナ行]

[ハ行]

［マ行］

［ヤ行］

［ラ行］

A

B

C

D

E

G

H

I

プログラム番号: 5724-A75

Printed in Japan

SC88-8987-00

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