CICS Transaction Server for z/OS
バージョン 4 リリース 2

インターネット・ガイド

SA88-4317-01
(英文原典：SC34-7173-01)

CICS Transaction Server for z/OS
バージョン 4 リリース 2

インターネット・ガイド

SA88-4317-01
(英文原典：SC34-7173-01)

お願い

本書および本書で紹介する製品をご使用になる前に、 515 ページの『特記事項』に記載されている情報をお読みください。

本書は、CICS Transaction Server for z/OS バージョン 4 リリース 2 (製品番号 5655-S97)、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： SC34-7173-01
CICS Transaction Server for z/OS
Version 4 Release 2
Internet Guide

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2011.9

目次

前書き

本書について

このマニュアルには、プログラムを作成するユーザーが バージョン 4 リリース 2 のサービスを使用するためのプログラミング・インターフェースが記述されています。

このマニュアルでは、CICS® 領域が HTTP サーバーおよび HTTP クライアントとして動作できるようにするための CICS Web サポートの設定および管理の方法を説明します。また、Web クライアントおよび Web サーバーと対話する CICS アプリケーション・プログラムの作成方法についても説明します。

本書を理解するための前提知識

本書は、CICS に関して、システム管理者としての知識、またはシステム・プログラマーやアプリケーション・プログラマーとしての知識があるユーザーを対象としています。

CICS Transaction Server for z/OS, バージョン 4 リリース 2 の変更点

このリリースに加えられた変更点に関する情報は、インフォメーション・センターの「*リリース・ガイド*」または以下の資料を参照してください。

- *CICS Transaction Server for z/OS リリース・ガイド*
- *CICS Transaction Server for z/OS V4.1 からのアップグレード*
- *CICS Transaction Server for z/OS V3.2 からのアップグレード*
- *CICS Transaction Server for z/OS V3.1 からのアップグレード*

リリース後に本文を技術的に変更した箇所は、その箇所の左側に縦線 (|) 引いて示しています。

第 1 部 概要: CICS および HTTP

インターネット、HTTP、TCP/IP および CICS Web サポートの各概念の概要、および CICS Web サポートの構造と動作についての情報。

第 1 章 CICS から Web への接続

CICS は Web のインターフェースになり、サーバーとして Web クライアントから要求を受け取ったり、クライアントとしてサーバーに要求を出したりできます。

CICS アプリケーションによって処理される Web クライアント要求

CICS Web サポートの使用

CICS Web サポートによって、CICS 領域は HTTP サーバーとして機能するようになります。

- CICS Web サポートは、CICS 文書または静的ファイルを使用して Web クライアントに静的応答を提供します。
- Web 対応ユーザー・アプリケーション・プログラムは HTTP 要求を受け取って分析し、動的なアプリケーション生成の応答を提供します。
- CICS Web サポートには、非 Web 対応アプリケーションへの Web クライアントのアクセスをサポートする、特定の範囲の CICS サービスが含まれます。 Web クライアントは、仮想 3270 端末と通信するように設計されている CICS プログラム、および COMMAREA またはチャネルを使用して別の CICS アプリケーションからリンクされるように設計されている CICS プログラムへのアクセス要求を出すことができます。
- CICS Web サポートは、クライアントからの非 HTTP 要求もサポートします。

Web サービスの使用

Web サービスは、ネットワーク上のマシン間での連携可能な対話をサポートするように設計されたソフトウェア・システムです。このサービスには、マシン処理が可能な形式 (特に WSDL と呼ばれる Web サービス記述言語) で記述されたインターフェースがあります。CICS Transaction Server バージョン 3 は、Web サービスのリクエスターにもプロバイダーにもなることができます。

Web サービスについては、「*CICS Web サービス・ガイド*」を参照してください。

IBM® HTTP Server の使用

IBM HTTP Server は、External CICS Interface (EXCI) および CICS ビジネス・ロジック・インターフェースを介して CICS アプリケーションへのアクセスを可能にします。

詳細については、 511 ページの『付録 K. CICS Web サーバー・プラグイン』および 399 ページの『第 25 章 CICS ビジネス・ロジック・インターフェースの概要』を参照してください。

CICS Transaction Gateway の使用

CICS Transaction Gateway は、Web クライアントから CICS アプリケーションにアクセスするための Web サーバー機能のセットを提供します。これらの機能には、アプリケーション特有のサーバー・プログラム (サーブレット) やブラウザー・プログラム (アプレット) だけでなく、共通機能用 IBM

提供コードを作成するための Java クラスと Java Bean が組み込まれています。従来型およびオブジェクト指向の CICS アプリケーションのどちらにもアクセスできるクラスが用意されています。アプレットとサーブレットは、CICS 提供のクラスを使用して ECI (External Call Interface) 要求と EPI (External Presentation Interface) 要求を作成します (CICS Transaction Gateway for z/OS® は ECI はサポートしますが EPI はサポートしないことに注意してください)。詳しくは、「*CICS Transaction Gateway: z/OS Administration*」を参照してください。

Web ソリューションを選択する手順については、IBM Redbooks 資料「*Revealed! Architecting Web Access to CICS*」(SG24-5466) を参照してください (http://www.redbooks.ibm.com/redbooks/pdfs/sg245466.pdf から入手可能)。

Web にアクセスする CICS アプリケーション

CICS Web サポートによって、CICS 領域は HTTP クライアントとして機能するようになります。CICS 内のユーザー・アプリケーション・プログラムは HTTP サーバーへの要求を開始し、HTTP サーバーからの応答を受け取ります。 CICS Web サポートは、ユーザー・アプリケーション・プログラムの **EXEC CICS WEB** コマンドに応答して、メッセージを処理します。

第 2 章 インターネット、TCP/IP、および HTTP の概念

TCP/IP (Transmission Control Protocol/Internet Protocol) および HTTP (Hypertext Transfer Protocol) に関連する重要な要素をよく理解しておいてください。必要であれば、ここのトピックで示した HTTP 仕様を確認してください。

TCP/IP プロトコル

TCP/IP は、ネットワーク内のコンピューター・システム同士を接続するために使用される通信プロトコルのファミリーです。このファミリーに含まれるプロトコルのうちの 2 つ (伝送制御プロトコル (TCP) とインターネット・プロトコル (IP)) にちなんで名前が付けられています。 Hypertext Transfer Protocol (HTTP) は、TCP/IP ファミリーのメンバーです。

TCP/IP ファミリー内のプロトコルは、多くの場合、オープン・システム間相互接続 (OSI) モデルの層に対応します。 表 1 は、TCP/IP ファミリーの HTTP 層と基礎層を OSI モデルの観点から示したものです。 OSI レイヤーとほぼ合致するシステム・ネットワーク体系 (SNA) 層も示されています。

表 1. TCP/IP プロトコル・ファミリーの層

層	OSI	SNA	TCP/IP
7	アプリケーション	アプリケーション	HTTP
6	プレゼンテーション	プレゼンテーション	(空)
5	セッション	データ・フロー	(空)
4	トランスポート	伝送	TCP
3	ネットワーク	パス制御	IP
2	データ・リンク	データ・リンク	サブネットワーク
1	物理	物理	

インターネット・プロトコル (IP)
IP は、TCP で使用されるコネクションレス・データ伝送サービスを提供するネットワーク層プロトコルです。データはリンクごとに伝送され、呼び出し時には終端間接続はセットアップされません。データ伝送の単位はデータグラム です。

伝送制御プロトコル (TCP)
TCP は、信頼性のある全二重コネクション型データ伝送サービスを提供するトランスポート層プロトコルです。ほとんどのインターネット・アプリケーションは TCP を使用します。

Hypertext Transfer Protocol (HTTP)
HTTP は、分散協調ハイパーメディア情報システムに使用されるアプリケーション層プロトコルです。HTTP は、Web クライアントと Web サーバー間で使用されるプロトコルです。

多くの TCP/IP のインプリメンテーションでは、TCP プロトコル (つまり、トランスポート層) とのアプリケーション・プログラミング・インターフェースを備えて

います。このインターフェースは、一般にソケット・インターフェース と呼ばれています。 CICS の TCP/IP ソケット・インターフェース、z/OS Communications Server IP CICS ソケット・インターフェースです。このインターフェースは z/OS Communications Server で提供され、z/OS の重要な部分となっています。 このインターフェースは、CICS Web サポートの一部ではありません。また、CICS SO ドメインは使用しません。「*z/OS Communications Server IP CICS ソケット・ガイド*」(SC88-9053) に、この CICS ソケット・インターフェースについての説明があります。

IP アドレス

TCP/IP インターネット上の各サーバーまたはクライアントは、数字の IP (インターネット・プロトコル) アドレスで識別されます。IP アドレスには、IPv4 (IP バージョン 4) アドレスと IPv6 (IP バージョン 6) アドレスの 2 つのタイプがあります。

IP アドレスは Internet Assigned Numbers Authority (IANA) とその代行者によって管理され、ユーザーに割り振られます。インターネット・アドレスではネットワークと個別ホストの両方が指定されます。アドレスの仕様は、ネットワークのサイズによって異なります。

IPv6 アドレス

IPv6 アドレスは 128 ビットのアドレスで、通常は 16 進表記で表されます。

```
16 進表記による IP アドレス: '00010022033344440000000abc0def0'x
ハーフワード 0: 16 進数 0001
ハーフワード 1: 16 進数 0022
ハーフワード 2: 16 進数 0333
ハーフワード 3: 16 進数 4444
ハーフワード 4: 16 進数 0000
ハーフワード 5: 16 進数 0000
ハーフワード 6: 16 進数 abc0
ハーフワード 7: 16 進数 def0
コロン 16 進表記による IP アドレス: 1:22:333:4444::abc0:def0

16 進表記による IP アドレス: '00000000000000000000ffff01020304'x
ハーフワード 0: 16 進数 0000
ハーフワード 1: 16 進数 0000
ハーフワード 2: 16 進数 0000
ハーフワード 3: 16 進数 0000
ハーフワード 4: 16 進数 0000
ハーフワード 5: 16 進数 ffff
ハーフワード 6: 16 進数 0102
ハーフワード 7: 16 進数 0304
コロン 16 進表記による IP アドレス: ::ffff:1.2.3.4 または ::ffff:0102:0304
```

アドレスは 8 つのハーフワード・フィールドで構成されます。ゼロはアドレスの出力の際に以下のように扱われます。

- フィールドに含まれている先行ゼロは無視されます。例えば、`0001` は `1` として表されます。
- アドレスの 1 つ以上の連続フィールドに値 `0000` が含まれている場合、これらのフィールドは `::` 表記を使用して表されます。

  例えば、`0000000000000ffff` は `::ffff` として表されます。

置換文字 :: は、置換されたフィールドの数を計算する際の混乱を避けるため、1 つのアドレスで 1 回だけ使用されます。

IPv4 アドレス

IPv4 アドレスは 32 ビットのアドレスで、通常はドット 10 進表記で表されます。

```
16 進表記による IP アドレス       :  '817EB263'x
バイト 0:  16 進数 81 = 10 進数 129
バイト 1:  16 進数 7E = 10 進数 126
バイト 2:  16 進数 B2 = 10 進数 178
バイト 3:  16 進数 63 = 10 進数 99
ドット 10 進表記による IP アドレス:  129.126.178.99
```

この例では、129.126 はネットワークを指定し、178.99 はそのネットワーク上のホストを指定しています。

CICS が受け入れる IP アドレス形式

CICS は、特定形式の IPv4 アドレスおよび IPv6 アドレスを処理用に受け入れます。

IPv6 アドレス形式

CICS は、以下の形式の IPv6 アドレスのみを受け入れます。

- 大括弧または /nn 表記のない、IPv6 固有のコロン 16 進アドレス (例、::a:b:c:d)

IPv6 アドレス構文の詳細については、RFC 4291 (「*IP Version 6 Addressing Architecture*」http://www.ietf.org/rfc/rfc4291.txt から入手できます) に記載されています。

IPv4 アドレス形式

CICS は、以下の形式の IPv4 アドレスのみを受け入れます。

- /nn 表記のない IPv4 固有のドット 10 進アドレス (例、1.2.3.4)。
- IPv6 形式に移行済みの IPv4 アドレス (IPv4 射影 IPv6 アドレス。例、::ffff:1.2.3.4)。
  - 内部的に、CICS はこのアドレスを 0:0:0:0:0:ffff:0102:0304 と等価の 2 進数に変換します。
- IPv6 互換アドレス (IPv4 互換 IPv6 アドレス。例、::1.2.3.4)。
  - 内部的に、CICS はこのアドレスを 0:0:0:0:0:0:0102:0304 と等価の 2 進数に変換します。

  この例外は、以下に適用されます。
  - CICS は以下のエントリーを許可しません。
    - 0.0.0.0
    - ::0.0.0.0
    - ::0

IPv4 アドレスにどの形式が指定されていようと、CICS はすべての IPv4 アドレスを IPv4 固有のドット 10 進アドレスとして表示します (例えば 1.2.3.4)。

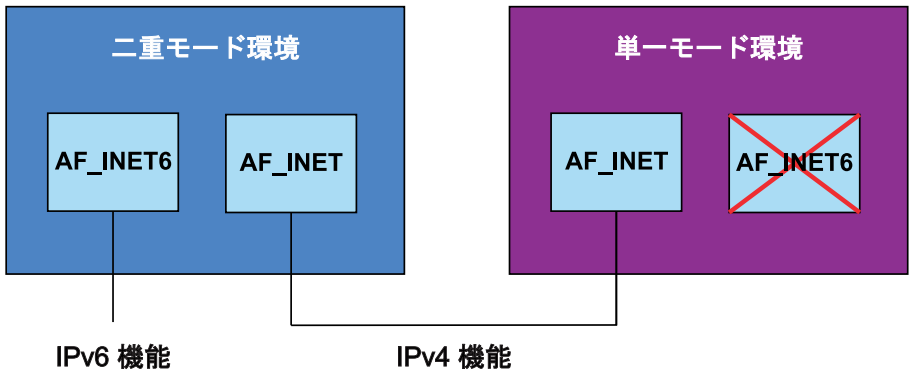
IPv6 および CICS についての理解

IPv6 は、IPv4 の後継となるプロトコルです。 IPv6 アドレッシングを使用するには、送信および受信環境が二重モード・アドレッシング (IPv4 および IPv6) をサポートしている必要があり、CICS 領域は正しいレベルの CICS で実行している必要があります。

IPv6 のインフラストラクチャー要件

IPv4 と IPv6 アドレッシングの両方を許可するためには、二重モードの TCP/IP インプリメンテーションが必要です。 AF_INET ソケットと別の領域にある別の AF_INET ソケット間の接続を確立するとき、単一モード (IPv4) 環境は、AF_INET アドレス・ファミリーを使用します。 IPv6 アドレスは、AF_INET ソケットではサポートされていません。これらのアドレスは接続を確立するための送信および受信領域で、AF_INET6 アドレス・ファミリーおよび AF_INET6 ソケットが必要です。二重モード環境は、AF_INET と AF_INET6 ソケットの両方を提供します。AF_INET および AF_INET6 についての詳細は、「*z/OS Communications Server IPv6 ネットワークとアプリケーション開発ガイド*」を参照してください。

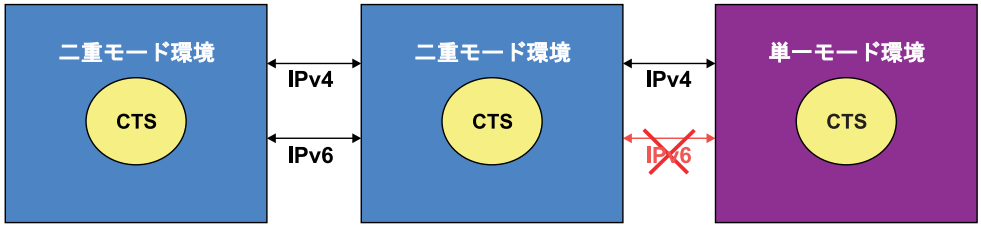
この図は、単一モード環境には AF_INET6 ソケットがないために、IPv6 機能がないことを示しています。



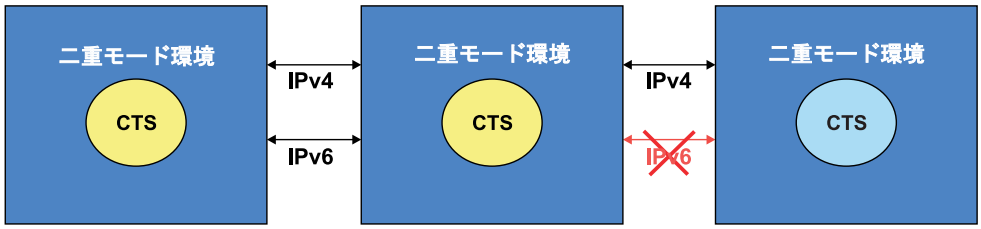
IPv6 の CICS 要件

IPv6 を使用して通信するには、最小レベルの CICS TS 4.1 が必要です。 CICS 領域は、二重モード (IPv4 および IPv6) 環境で実行されていなければなりません。また、CICS が通信するクライアントまたはサーバーも、二重モード環境で実行されていなければなりません。

次の図は、CICS-CICS 間通信を示しています。そこでは、2 つの二重モード CICS 環境が IPv4 アドレッシングか IPv6 アドレッシングを使用して通信できます。単一モード CICS 環境も接続されていますが、IPv4 のみを使用して通信できます。



次の図は、CICS-CICS 間通信を示しています。そこでは、2 つの二重モード CICS 環境が IPv4 アドレッシングか IPv6 アドレッシングを使用して通信できます。二重モード CICS TS 3.2 環境も接続されていますが、IPv4 のみを使用して通信できます。



ホスト名

インターネット上のホスト、または Web サイトは、www.example.com のようなホスト名によって識別されます。ホスト名は、ドメイン・ネームと呼ばれる場合もあります。ホスト名は IP アドレスにマッピングされますが、ホスト名と IP アドレスの間に 1 対 1 の関係はありません。

ホスト名は、Web クライアントがホストへの HTTP 要求を作成する際に使用されます。要求を作成するユーザーがホスト名ではなくサーバーの IP アドレスを指定することもできますが、現在のインターネットでは普通行われません。ホスト名は数字の IP アドレスよりもユーザーにとって便利です。企業や組織、個人などは、多くの場合、ユーザーが容易に記憶できる、Web サイトのホスト名を選択します。

最新の HTTP 実装でより重要なこととして、HTTP 要求でホスト名を使用すると、結果として以下のようになります。

- 1 つのホストの名前で行われるサービスを、異なる IP アドレスを持つ多数のサーバーによって提供できる。
- ある IP アドレスを持つ 1 つのサーバーが、多数のホストの名前でサービスを提供できる。この使用法は、*仮想ホスティング* として知られています。 10 ページの『仮想ホスティング』では、この処理について説明しています。

ホスト名は、*DNS* サーバーまたはドメイン・ネーム・サーバーと呼ばれるサーバーによって、IP アドレスにマッピングされます。大規模ネットワークでは、多くの DNS サーバーが共同でホスト名と IP アドレス間のマッピングを行います。

仮想ホスティング

HTTP には仮想ホスティングの概念が含まれています。これは、単一の HTTP サーバーが、同じ IP アドレスを持つ複数のホストの機能を果たすことができるというものです。URIMAP リソースをセットアップすることにより、Web サポートで仮想ホスティングを使用できます。

DNS サーバーは、複数の異なるホスト名を同じ IP アドレスに割り振ることができます。 HTTP クライアントが特定のホストに対して要求を行う場合には、DNS サーバーを使用してそのホスト名に対応する IP アドレスを見つけ、要求をその IP アドレスに送信します。

HTTP/1.0 では、HTTP メッセージにホスト名が現れません。IP アドレスが解決した時点で失われるからです。 IP アドレスで表されるサーバーにリソースのセットが複数保持されていると、どのリソースがどのホストに属しているかをサーバーが区別するのは困難です。

しかし、HTTP/1.1 の要求では、要求の中に (通常は Host ヘッダーに) ホスト名が記述されます。メッセージ内にホスト名を記述することで、HTTP サーバーは、異なるホスト名を含む要求を、各ホストの適切なリソースに送信することができます。 HTTP のこの機能は、仮想ホスティングとして知られています。CICS Web サポートでは、URIMAP リソースを使用することで仮想ホスティングをサポートしています。

ポート番号

サーバーでは、同時に複数のユーザー・プロセスが TCP を使用できます。各プロセスに関連したデータを識別するために、ポート番号が使用されます。ポート番号は 16 ビットで、 65535 までの数を使用できます。しかし、実際には、これらの数の小さなサブセットのみが使用されるのが普通です。

サーバー・プロセスに初めて連絡を取る場合、クライアント・プロセスはウェルノウン・ポート番号 を使用して通信を開始できます。ウェルノウン・ポート番号は、IANA (Internet Assigned Numbers Authority) によってインターネット上の特定のサービスに割り当てられます。ウェルノウン・ポート番号の範囲は 0 ～ 1023 です。表 2 に例を示します。

表 2. サービスおよびウェルノウン・ポート番号

サービス	ウェルノウン・ポート番号
ファイル転送プロトコル (FTP)	21
Telnet	23
Hypertext Transfer Protocol (HTTP)	80
Secure Sockets Layer (SSL) 対応の HTTP	443
CORBA Internet Inter-ORB Protocol (IIOP)	683
SSL 対応の CORBA IIOP	684

CICS External Call Interface (ECI) には、登録済みポート番号 1435 が割り当てられています。

ウェルノウン・ポートは、クライアント・プロセスとサーバー・プロセス間の通信を確立するためにのみ使用されます。その後サーバーは、後続の使用に*一時ポート番号* を割り振ります。一時ポート番号は、プロセスが通信を開始するときに動的に割り当てられる固有のポート番号です。これらのポート番号は、通信が終了すると解放されます。

IANA メディア・タイプおよび文字セット

Internet Assigned Numbers Authority (IANA) とは、インターネットで使用するプロトコルに名前を割り当てる役割を担う国際機関のことです。名前についてさらに調べるには、これらのリンクを使用してください。

- IANA メディア・タイプは、インターネット上で広く伝送されているデータ・タイプの名前です。これらのタイプは、http://www.iana.org/assignments/media-types/ で説明されています。

  テキスト・メディア・タイプ (`text/` で始まるタイプ、または `+xml` を含むタイプなど) は RFC 3023 で識別されています。RFC 3023 は、http://www.ietf.org/rfc/rfc3023.txt から入手できます。
- IANA 文字セットは、文字セット・レジストリーの名前です。これらの文字セットは、http://www.iana.org/assignments/character-sets で説明されています。

  CICS は、コード・ページ変換のための IANA 文字セットをすべてサポートしているわけではありません。 CICS がサポートしている文字セットについては、413 ページの『付録 A. HTML コード化文字セット』で説明しています。

URL の構成要素

URL (Uniform Resource Locator) は、URI (Universal Resource Identifier) の固有のタイプです。URL は通常、インターネット上の既存のリソースを位置指定します。URL は、Web クライアントがサーバーにリソースを要求する際に使用されます。

このトピックは、URL と URI について要約したものです。詳細については、Internet Society および IETF (Internet Engineering Task Force) の Request for Comments 文書である RFC 2396「*Uniform Resource Identifiers (URI): Generic Syntax*」(http://www.ietf.org/rfc/rfc2396.txt) で、URI および URL の概念が定義されています。

簡単に言うと、URI はリソースを識別する任意の文字ストリングとして定義されています。 URL は、リソースをその名前またはその他の属性ではなく、位置またはそのリソースへのアクセスに使用する手段によって識別する URI として定義されています。

リソース ID の新しい形式 IRI (Internationalized Resource Identifier) は、英語以外の各国語に適した文字およびフォーマットの使用を許可します。要求および応答に関係するアプリケーションが IRI をサポートする場合は、URI または URL の代わりに IRI を使用できます。 IRI について詳しくは、262 ページの『Internationalized Resource Identifiers (IRI)』を参照してください。

HTTP (または HTTPS) の URL は通常、以下の 3 つまたは 4 つの構成要素で構成されます。

1. **スキーム。** スキームは、インターネット上にあるそのリソースへのアクセスに使用されるプロトコルを識別します。これは、HTTP (SSL なし) または HTTPS (SSL あり) のいずれかです。
2. **ホスト。** ホスト名は、そのリソースを保持するホストを識別します。例えば、`www.example.com` などです。サーバーはホストの名前でサービスを提供しますが、ホストとサーバーの間に 1 対 1 のマッピングがあるわけではありません。9 ページの『ホスト名』を参照してください。

   ホスト名の後に**ポート番号**が続く場合もあります。 10 ページの『ポート番号』を参照してください。サービスの予約済みポート番号は、通常 URL から省略されています。ほとんどのサーバーは、HTTP および HTTPS 用に予約済みのポート番号を使用するため、HTTP URL からポート番号は除外されます。
3. **パス。** パスは、Web クライアントがアクセスする、ホストの特定のリソースを識別します。例えば、`/software/htp/cics/index.html` などです。
4. **照会ストリング。** 照会ストリングが使用される場合はパス構成要素の後に付加され、そのリソースの使用目的 (例えば、検索用のパラメーターとして、または、処理するデータとしてなど) に関する情報ストリングを提供します。照会ストリングは通常、名前と値がペアになったストリングで、例えば `term=bluebird` のようになります。名前と値のペアは、アンパーサンド (&) で互いに分離されます。例えば `term=bluebird&source=browser-search` のようになります。

URL の構成要素のスキームとホストの定義では大/小文字の区別はされませんが、パスと照会ストリングでは大/小文字の区別がされます。通常は、URL 全体が小文字で指定されます。

URL の構成要素は、次のように結合され、区切り文字で区切られます。

```
scheme://host:port/path?query
```

- スキームの後には、コロンと 2 つのスラッシュが付きます。
- ポート番号を指定する場合、その番号はホスト名の後に、コロンで区切って付加されます。
- パス名は、単一のスラッシュから始まります。
- 照会ストリングを指定する場合は、その前に疑問符 (?) が付きます。

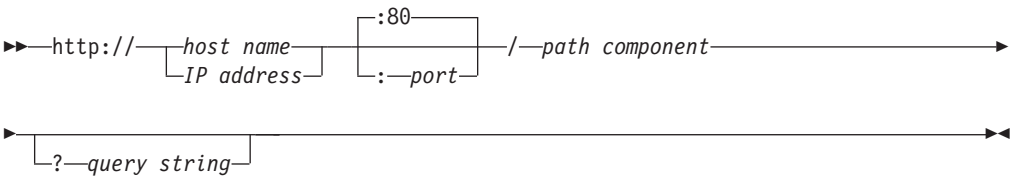


図 1. HTTP URL の構文

HTTP URL の例を次に示します。

```
http://www.example.com/software/index.html
```

ポート番号を指定した場合、URL は次のようになります。

```
http://www.example.com:1030/software/index.html
```

URL の後にフラグメント ID を付けることができます。 URL とフラグメント ID の間に使用される分離文字は、# 文字です。フラグメント ID は、検索したばかりの項目の参照または機能を、Web ブラウザーに指示するために使用されます。例えば、URL が HTML ページを識別している場合、フラグメント ID を使用してそのページ内のサブセクションを指示するには、そのサブセクションの ID を使用します。このケースでは、Web ブラウザーは通常、そのサブセクションが見える状態でそのページをユーザーに対して表示します。フラグメント ID に対して Web ブラウザーが実行するアクションは、その項目のメディア・タイプ、およびそのメディア・タイプに対するフラグメント ID の定義方法によって異なります。

FTP や Gopher などの他のプロトコルも URL を使用します。ただし、これらのプロトコルで使用する URL は、HTTP で使用する URL とは構文が異なります。

HTTP プロトコル

HTTP 要求および応答の正しいフォーマットは、クライアントとサーバーで使用される HTTP プロトコル (または HTTP 仕様) のバージョンによって異なります。

インターネットで一般的に使用される HTTP プロトコルのバージョン (つまり「HTTP バージョン」) は HTTP/1.0 および HTTP/1.1 です。前者は古いプロトコルで機能も少なく、後者は新しいプロトコルで、機能も多くなっています。クライアントとサーバーで異なるバージョンの HTTP プロトコルが使用されることもあります。それで、クライアントとサーバーの両方で、要求または応答の HTTP バージョンをそのメッセージの最初の行で明示する必要があります。

Internet Society および IETF (Internet Engineering Task Force) Request for Comments (RFC) の文書では、HTTP プロトコルの公式な定義が提供されています。

HTTP/1.0

RFC 1945 (「*Hypertext Transfer Protocol - HTTP/1.0*」)。これは http://www.ietf.org/rfc/rfc1945.txt から入手できます。

HTTP/1.1

RFC 2616 (「*Hypertext Transfer Protocol - HTTP/1.1*」)。これは http://www.ietf.org/rfc/rfc2616.txt から入手できます。

これらの RFC では、クライアントおよびサーバーが要求と応答を適切な方法で交換するために実行するアクションが、HTTP プロトコルのバージョンごとに明記されています。 HTTP 要求は、サーバー上にある指定されたホストに対して、クライアントによって作成されます。要求の目的は、サーバー上にあるリソースにアクセスすることです。HTTP 応答は、サーバーからクライアントに対して作成されます。応答の目的は、要求されたリソースをクライアントに提供すること、または要求されたアクションが実行されたことをクライアントに通知すること、あるいは要求の処理中にエラーが発生したことをクライアントに通知することです。これらのアクションはすべて、「要件」とされています。クライアントまたはサーバーが該当するバージョンの HTTP プロトコルの要件を満たしている場合、それは HTTP 仕様に準拠しているといえます。

クライアントに送信される HTTP 応答では、状況コード (3 桁の数字) にコードの意味を要約した理由句 (状況テキストとも呼ばれます) が伴っています。これらの項目は、応答の HTTP バージョンと共に応答の最初の行に配置されます。そのため最初の行は、状況表示行と呼ばれます。

HTTP 要求

HTTP 要求は、サーバー上にある指定されたホストに対して、クライアントによって作成されます。要求の目的は、サーバー上にあるリソースにアクセスすることです。

要求を作成するために、クライアントは URL (Uniform Resource Locator) の構成要素を使用します。この構成要素には、リソースにアクセスするために必要な情報が含まれています。 URL については、11 ページの『URL の構成要素』で説明しています。

正常に作成された HTTP 要求には、以下の要素が含まれています。

1. 要求行。
2. 一連の HTTP ヘッダー、またはヘッダー・フィールド。
3. メッセージ・ボディ (必要な場合)。

各 HTTP ヘッダーの後には、復帰改行 (CRLF) が続きます。 HTTP ヘッダーが終わると、その後に追加の CRLF が使用され (空の行を置くため)、それからメッセージ・ボディが始まります。

要求行

要求行は、要求メッセージの最初の行です。これは、少なくとも以下の 3 つの項目で構成されます。

1. **メソッド**。メソッドは、リソースを使用して何を行うかをサーバーに伝える、1 語のコマンドです。例えば、サーバーは、リソースをクライアントに送信するように要求される場合があります。
2. 要求の URL のパス構成要素。パスは、サーバー上のリソースを識別します。
3. クライアントがそのメッセージをどの HTTP 仕様に準拠させようとしているのかを示す、HTTP バージョン番号。

要求行の例は次のようになります。

```
GET /software/htp/cics/index.html HTTP/1.1
```

この例では以下のようになります。

- メソッドは `GET`
- パスは `/software/htp/cics/index.html`
- HTTP バージョンは `HTTP/1.1`

要求行には、次のような追加の項目が含まれる場合もあります。

- 照会ストリング。これは、リソースが何かの目的に使用できる情報のストリングを提供します。照会ストリングは先頭に疑問符 (?) を付けて、パスの後に続きます。

- パスに追加される、URL のスキームおよびホスト構成要素。リソースの位置がこの方法で指定される場合、これは**絶対 URI** 形式と呼ばれています。 HTTP/1.1 の場合、この形式は、要求がプロキシー・サーバーを通る場合に使用されます。また、HTTP/1.1 の場合、URL のホスト構成要素が要求行に含まれていない場合は、メッセージの Host ヘッダーに組み込む必要があります。

HTTP ヘッダー

HTTP ヘッダーは、メッセージ、送信者、および送信側が求める受信側との通信方法に関する情報を、受信側に提供するため、メッセージに書き込まれます。各 HTTP ヘッダーは、名前と値で構成されます。 HTTP プロトコルの仕様では、標準セットの HTTP ヘッダーを定義し、それらのヘッダーを正しく使用する方法について記述しています。または、HTTP メッセージには、HTTP/1.1 または HTTP/1.0 の仕様の一部には含まれていない、拡張ヘッダーを組み込むこともできます。

クライアント要求の HTTP ヘッダーには、その要求に対する応答方法を決定するためにサーバー側で使用できる情報が含まれます。例えば、以下の一連のヘッダーを使用すると、要求された文書をエンド・ユーザーがフランス語またはドイツ語でのみ読み取ること、および、その文書を最後にクライアントが取得した日時以降に変更があった場合にのみ、文書を送信する必要があることを、指定できます。

```
Accept-Language: fr, de
If-Modified-Since: Fri, 10 Dec 2004 11:22:13 GMT
```

要求メッセージの一連の HTTP ヘッダーの後には、ヘッダーとメッセージ・ボディを区切るために、空の行 (つまり、CRLF のみの行) が置かれます。

メッセージ・ボディ

HTTP メッセージのボディの内容はいずれも、メッセージ・ボディまたは**エンティティー・ボディ**として参照できます。技術の見地からいえば、エンティティー・ボディはメッセージの実際の内容です。メッセージ・ボディには、エンティティー・ボディが含まれます。エンティティー・ボディは、元の状態のままであることも、移送のためになんらかの方法で (例えば、チャンクに分割されている (チャンク転送コーディング) など) エンコードされることもあります。要求のメッセージ・ボディは、便宜上、要求ボディとして参照される場合があります。

メッセージ・ボディは、一部の要求メソッドには適切で、その他の要求メソッドには不適切な場合があります。例えば、入力データをサーバーに送信する、POST メソッドを持つ要求には、データが含まれているメッセージ・ボディがあります。リソースの送信をサーバーに求める、GET メソッドを持つ要求には、メッセージ・ボディはありません。

HTTP 応答

HTTP 応答は、サーバーからクライアントに対して作成されます。応答の目的は、クライアントから要求されたリソースを提供すること、または要求されたアクションが実行されたことをクライアントに通知することです。そうでない場合は、要求の処理中にエラーが発生したことをクライアントに通知します。

HTTP 応答には、以下のものが含まれています。

1. 状況表示行。

2. 一連の HTTP ヘッダー、またはヘッダー・フィールド。
3. メッセージ・ボディ (これは通常必要です)。

要求メッセージと同様に、各 HTTP ヘッダーの後には、復帰改行 (CRLF) が続きます。 HTTP ヘッダーが終わると、その後に追加の CRLF が使用され (空の行を置くため)、それからメッセージ・ボディが始まります。

状況表示行

状況表示行は、応答メッセージの最初の行です。これは、以下の 3 つの項目で構成されます。

1. サーバーがそのメッセージをどの HTTP 仕様に準拠させようとしているのかを示す、HTTP バージョン番号。
2. **状況コード**。これは、要求の結果を示す、3 桁の番号です。
3. **理由句**。これは状況テキストとも呼ばれるもので、状況コードの意味を要約した、人間が読むことができるテキストです。

応答行の例は次のようになります。

```
HTTP/1.1 200 OK
```

この例では以下のようになります。

- HTTP バージョンは `HTTP/1.1`
- 状況コードは `200`
- 理由句は `OK`

状況表示行のこれらの要素について詳しくは、17 ページの『状況コードおよび理由句』で説明しています。

HTTP ヘッダー

サーバーの応答の HTTP ヘッダーには、クライアントがその応答に関する情報をさらに得るための情報、および、その応答を送信したサーバーに関する情報が含まれます。この情報は、クライアントがその応答をユーザーに表示するため、その応答を将来の使用に備えて保管 (またはキャッシング) するため、および、今すぐまたは今後そのサーバーにさらなる要求をするために役立ちます。例えば、以下の一連のヘッダーは、クライアントに応答が送信された日時、その応答が CICS から送信されたこと、および、その応答が JPEG 画像であることを通知しています。

```
Date: Thu, 09 Dec 2004 12:07:48 GMT
Server: IBM_CICS_Transaction_Server/3.1.0(zOS)
Content-type: image/jpg
```

要求が失敗した場合、ヘッダーは、その要求を正常に完了するために何が必要かをクライアントに通知するために使用されます。

応答メッセージの一連の HTTP ヘッダーの後には、ヘッダーとメッセージ・ボディを区切るために、空の行 (つまり、CRLF のみの行) が置かれます。

メッセージ・ボディ

応答のメッセージ・ボディは、便宜上、応答ボディとして参照される場合があります。

メッセージ・ボディは、ほとんどの応答に使用されます。例外は、(ヘッダーは要求するが、応答のボディは要求しない) HEAD メソッドを使用したクライアント要求にサーバーが応答する場合と、サーバーが特定の状況コードを使用している場合です。

正常に終了した要求に対する応答の場合、メッセージ・ボディにはクライアントによって要求されたリソースか、またはクライアントによって要求されたアクションの状況に関するなんらかの情報が含まれます。失敗した要求に対する応答の場合、メッセージ・ボディでは、エラーの理由に関する詳しい情報を提供するか、その要求を正常に完了するためにクライアントが行う必要があるアクションに関する詳しい情報を提供する可能性があります。

状況コードおよび理由句

クライアントに送信される HTTP 応答では、状況コード (3 桁の数字) にコードの意味を要約した理由句 (状況テキストとも呼ばれます) が伴っています。これらの項目は、応答の HTTP バージョンと共に応答の最初の行に配置されるので、その行が状況表示行であることが分かります。

状況コードは数字の範囲で分類され、コードの各クラスの基本的な意味は同じです。

- 100 から 199 までの範囲は、通知として分類されます。
- 200 から 299 までは正常終了です。
- 300 から 399 まではリダイレクトです。
- 400 から 499 まではクライアント・エラーです。
- 500 から 599 まではサーバー・エラーです。

ある範囲を全体として記述する場合は、「1xx」、「2xx」といったように名前を付けることができます。HTTP プロトコル仕様では 600 以上の状況コードは定義されていません。

HTTP/1.0 および HTTP/1.1 仕様で定義されている状況コードは、各範囲にわずかしかありません。HTTP/1.1 仕様では、HTTP/1.0 仕様よりも多くの状況コードが使用されます。

HTTP 仕様で定義されている理由句 (例えば、「Not Found (未検出)」や「Bad Request (不正な要求)」) が推奨されていますが、これらはオプションです。HTTP/1.1 仕様では、各状況コードの理由の句は、ローカルの同等句と置き換えることができると記述されています。

200 (OK) 状況コードは、Web クライアントが要求した全リソースを提供する、通常応答に使用されます。それ以外のほとんどの状況コードは、要求の完遂の妨げとなるエラーが存在している状況、またはクライアントがその要求を正常に完了するために、何か別のことをする必要がある状況で使用されます。このような状況としては、例えばリダイレクト URL をたどる、要求を修正してその要求をサーバーで受け入れ可能にする、などがあります。

応答の HTTP ヘッダー、応答ボディの HTTP ヘッダー、またはその両方の HTTP ヘッダーで、クライアントに対する追加の指示および情報を提供することができま

す。HTTP 仕様には、各状況コードを含む応答の内容に対する要件および提案が含まれています。これらの要件で指定されている内容は以下のとおりです。

- 応答で使用する必要のある、または使用することのできる HTTP ヘッダー。例えば、状況コード 405 (Method not allowed (メソッドが許可されていません)) を使用する場合は、Allow ヘッダーを使用して、許可されている メソッドを示します。
- 応答ボディを使用する必要があるか、そうでないか。例えば、メッセージ・ボディに状況コード 204、205、および 304 を含めることはできません。
- 応答ボディが使用される場合は、その応答ボディで提供できる情報はどのようなものか。例えば、リダイレクトのメッセージ・ボディで、リダイレクト URL のハイパーリンクを提供することができます。

状況コードの意味および正しい使用方法についての詳細は、ご使用の HTTP の仕様書を参照してください。HTTP 仕様についての詳細は、13 ページの『HTTP プロトコル』を参照してください。

予約文字と除外文字

HTTP 要求の送信と解釈を正しく行えるように、URL での特定の文字の使用が制限されています。そうした文字は、要求の送信時に、安全な形式に変換される必要があります。

このトピックは、予約文字と除外文字についての要約です。詳しくは、Internet Society および IETF (Internet Engineering Task Force) の Request for Comments 文書である RFC 2396 (「*Uniform Resource Identifiers (URI): Generic Syntax*」) に、URI および URL において予約または除外されている文字のリストがあります。RFC 2396 は、http://www.ietf.org/rfc/rfc2396.txt にあります。

URI または URL に含まれる、1 つ以上の URI 構成要素や URL 構成要素のコンテキストで特別な目的を持つ文字は、予約文字として知られています。例えば、文字 /、?、&、および : は、さまざまな構成要素の区切り文字として使用されます。何らかの理由で予約文字が使用された場合、マシン・インタープリターは URI または URL を誤って解釈する可能性があります。

また、マシンやユーザーの混乱を招く可能性がある文字や、一部のマシン・インタープリターで問題を生じることがわかっている文字などの特定の文字は、URI または URL のどの部分でも使用を許可されないか、または除外されます。例えば、スペース文字を URL で使用することは許可されていません。

予約文字をその特別な目的以外の理由で URL に含めたい場合や、除外文字を URL に含めたい場合は、その URL の構成要素を含む要求をサーバーに送信するときに、それらの文字をエスケープする必要があります。照会ストリングで送信されるデータに含まれるそのような文字も、エスケープする必要があります。

文字をエスケープするには、その文字を %xx という形式の 3 文字のストリングで置き換えます。この中の xx は、その予約文字の ASCII 16 進数表記です。このフォーマットのため、エスケープはパーセント・エンコードとしても知られます。

要求がサーバーに到達すると、サーバーはエスケープ文字をアンエスケープ することができます。予約文字や除外文字を構文解析アプリケーションが誤って解釈するリスクを避けるために、アンエスケープは必ず、URL および照会ストリングの情報が構文解析された後で行われます。

フォームのデフォルトのエンコード方式 (application/x-www-form-urlencoded) では予約文字や除外文字がエスケープされるため、要求のフォーム・データは通常、特殊文字をエスケープして送信されます。『HTML フォーム』を参照してください。

HTML フォーム

HTML では、フォームとは <form> タグによって区切られた領域であり、テキスト入力ボックス、ボタン、チェック・ボックス、およびグラフィカル・ユーザー・インターフェースのその他の機能が含まれます。フォームは、サーバーに送信されるデータをエンド・ユーザーが提供できるようするため、Web アプリケーションで使用されます。

このトピックは、HTML フォームについて要約したものです。詳しくは、下記のリンクを使用してください。

フォームでは、ユーザーがデータを提供するために対話できる要素は、フォーム・フィールドと呼ばれています。 HTML の各フォーム・フィールドには名前が割り当てられており、サーバー・アプリケーションはこの名前によって各フォーム・フィールドを識別できます。しかし、ユーザーはこの名前を見ることはできません。

フォームの各種要素はユーザーには異なって見えますが、どの要素でも、情報は & 文字で区切られた一連の名前/値のペアとしてサーバー・アプリケーションに送信されます。それぞれの名前はフォーム・フィールドの名前であり、値はユーザーのアクションによって生成されるデータです。例えば、次に示すのは、ユーザーがファーストネームとラストネームを入力する 2 つのテキスト入力ボックスが含まれるフォームの場合です。

```
firstname=Maria&lastname=Smith
```

フォーム・データは、<form> タグ内でどのメソッド (GET または POST) が指定されているかに応じて、以下に示した 2 とおりの方法のいずれかでサーバーに送信されます。

- メソッドが GET であれば、フォームのデータは URL の照会ストリングで送信されます。
- メソッドが POST であれば、フォームのデータはメッセージ・ボディで送信されます。

フォーム・データのエンコードに必要な文字セットは、CHARACTERSET オプションで指定します。この文字セットは、対応する HTML フォームによって決まるフォーム・エンコード方式と一致しなければなりません。詳細については、 20 ページの『クライアント・エンコード方式の決定方法』を参照してください。

フォーム・データは通常、特殊文字をエスケープして送信されます。エスケープの目的については、 18 ページの『予約文字と除外文字』を参照してください。

フォームを GET メソッドで定義する場合、データは URL の照会ストリングとして送信されるため、予約文字または除外文字を常にエスケープする必要があります。

フォームを POST メソッドで定義する場合、データはメッセージ・ボディで送信されます。ただし、HTML 2.0 仕様での定義のとおり、すべてのフォームのデフォルト・エンコード方式は `application/x-www-form-urlencoded` です http://www.w3.org/MarkUp/html-spec/html-spec_8.html#SEC8.2.1を参照してください。このエンコード方式を POST メソッドで定義されたフォームに使用すると、データはメッセージ・ボディで送信されますが、予約文字または除外文字は、URL 内にある場合と同じようにエスケープされます。

代わりのエンコード・タイプである `multipart/form-data` をフォームに指定する (HTML の <form> タグの ENCTYPE 属性を使用して指定する) 場合、フィールド名の中の非 ASCII 文字はエスケープする必要がありますが、フィールド値の非 ASCII 文字をエスケープする必要はありません。このデータも、メッセージ・ボディの一連の個別セクションで表されます。従来のアプリケーションでは、このエンコード方式がサポートされない場合があります。 CICS ではこの方式がサポートされています。 multipart/form-data エンコード方式については、Internet Society および IETF の Request for Comments 文書である RFC 1867「*Form-based File Upload in HTML*」(http://www.ietf.org/rfc/rfc1867.txt) で説明されています。

クライアント・エンコード方式の決定方法

HTTP クライアントが (GET および POST の両メソッドの) フォーム・データに使用する文字エンコード方式 (**charset** パラメーター) は、HTML フォームの情報によって決定されます。

HTTP クライアントは通常、HTML フォームに使用されたのと同じ文字エンコード方式を使用してフォーム・データを送信します。この文字エンコード方式は、Content-Type ヘッダーの **charset** パラメーターで指定されるか、または HTML に組み込まれた等価な `META` タグを使用して指定されます。例えば、以下のように指定します。

```
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

HTML FORM 要素の accept-charset 属性を使用して、追加の許容文字エンコード方式を指定することもできます。コード・ページを指定しなかった場合、CICS はこの情報を **charset** パラメーターから取得します。HTML フォームの文字エンコード方式は、通常は ISO-8859-1 (CCSID 819) または UTF-8 (CCSID 1208) ですが、これらの値に限定されるわけではありません。

文字エンコード方式の情報は通常、送信されたフォーム要求の一部として存在しているわけではありません。そのため、インターネットのデフォルトの文字セット (ISO-8859-1) が使用されていない場合、フォームを読み取るアプリケーションは、CHARACTERSET キーワードを使用してエンコード方式を指定する必要があります。CHARACTERSET が省略されているが、HTTP クライアントが Content-Type ヘッダーで **charset** 値を提供している場合は (HTML フォーム送信の標準的なプラクティスではない)、その **charset** 値が使用されます。そうでない場合、CICS は ISO-8859-1 を想定します。

チャンク転送コーディング

チャンク転送コーディングはチャンク化とも呼ばれており、メッセージのボディを、各チャンクがそれぞれ独自のチャンク・サイズ・ヘッダーを持った状態で、一連のチャンクとして転送します。メッセージの終わりは、長さがゼロで、空の行を含むチャンクによって示されます。

このトピックは、チャンク転送コーディングについて簡潔に要約したものです。チャンク転送コーディングを使用するには、クライアントおよびサーバーの両方が HTTP/1.1 を使用している必要があります。詳しくは、HTTP/1.1 仕様 (RFC 2616) を参照してください。

この定義済みプロセスが意味することは、アプリケーションによって生成されたエンティティー・ボディや、大きなエンティティー・ボディを、扱いやすいセグメントで送信することができる、ということです。クライアントまたはサーバーは、長さがゼロのチャンクを受信したとき、チャンク化済みメッセージが終了することを認識しています。

チャンク化済みメッセージのボディの後に、末尾ヘッダーと呼ばれる、補足の HTTP ヘッダーを含むオプションのトレーラーを続けることができます。クライアントおよびサーバーがトレーラーを受け入れることは必須ではないので、クライアントがトレーラーを受け入れることをサーバーが認識していない場合は、補足の HTTP ヘッダーが提供するのは非必須情報のみです。

パイプライン化

パイプライン化では、クライアントは複数の HTTP 要求を、応答を待たずにサーバーに送信します。応答は、要求が受信されたのと同じ順序でサーバーから返す必要があります。

このトピックは、パイプライン化について要約したものです。HTTP/1.1 仕様書 (RFC 2616) には、HTTP 要求に対するべき等に関する規則が定義されています。HTTP 仕様についての詳細は、13 ページの『HTTP プロトコル』を参照してください。

要求側は、要求がべき等になるようにする責任があります。べき等とは、一連の要求がすべて、またはその一部が繰り返されたときに得られる結果は、常に同じになることをいいます。したがって、サーバーへの接続に何らかの障害が起こった場合、クライアントは、サーバーが一連の要求のすべてまたは一部を実施していたか、あるいは何も実施していなかったかどうかを知らなくても、一連の要求を再試行できます。

ほとんどの要求メソッドは、それらの要求メソッドがそれら自身に対して使用された場合は、べき等です。その理由は、そのメソッドが使用されるたびに、得られる結果は同じになるからです (例外は POST メソッドです。これは、このメソッドではサーバー上のリソースが変更されるからです)。ただし、パイプライン化中に一連の要求が発行された場合、そのシーケンスが非べき等になることがあります。特に、リソースが変更されている場合は、非べき等になります。

要求をパイプライン化しようとする場合は、要求シーケンスをどの時点で終了しても、論理エラーにならずに最初から再開できることを確認してください。そうでない場合は、個別に要求を行い、各要求の後に確認を待ってください。

関連タスク

153 ページの『サンプル・プログラム: HTTP サーバーへの要求のパイプライン化』
サンプル・プログラム DFH$WBPA (アセンブリ言語)、DFH$WBPC (C)、および DFH0WBPO (COBOL) は、CICS で HTTP サーバーへのクライアント要求をどのようにしてパイプライン化できるかを示します。

持続接続

Web クライアントとサーバー間の持続接続は、要求と応答の複数回の交換に再使用できます。

持続接続により、要求ごとに新しい接続を確立する必要がなくなるため、ネットワークのパフォーマンスが改善されます。新しい接続を確立すると、既存の接続を使用して要求を行う場合に比べ、消費される追加のネットワーク・リソースは大幅に増加します。

HTTP/1.0 では、サーバーに対するデフォルト・アクションは、サーバーが Web クライアントから要求を受信し、応答を送信するとその接続を閉じるというものでした。Web クライアントがサーバーに接続を開かせておくようにしていた場合は、要求で Connection: Keep-Alive ヘッダーを送信する必要がありました。

HTTP/1.1 の場合、持続接続がデフォルトです。Web クライアントとサーバーの間で接続が行われると、デフォルトではそのサーバーは、その接続を開いたままにしておきます。接続が閉じられるのは、Web クライアントが Connection: close ヘッダーを送信して閉止を要求した場合、サーバーのタイムアウト設定に達した場合、またはサーバーがエラーを検出した場合のみです。

HTTP 基本認証

HTTP 基本認証は、簡単な質問 / 応答メカニズムで、サーバーはこれを使用してクライアントに認証情報 (ユーザー ID およびパスワード) を要求することができます。クライアントは、Authorization ヘッダーで認証情報をサーバーに渡します。認証情報は base64 方式でエンコードされています。

このトピックは、HTTP 基本認証について要約したものです。詳しくは、RFC 2617「*HTTP Authentication: Basic and Digest Access Authentication*」(http://www.ietf.org/rfc/rfc2617.txt) を参照してください。

注: HTTP 基本認証体系は、Web クライアントとサーバー間の接続がセキュアである場合のみ、セキュアであると見なすことができます。接続がセキュアでない場合、この認証体系によるセキュリティーでは、無許可ユーザーがサーバーの認証情報を発見するのを十分に防ぐことはできません。パスワードが傍受される可能性があると思われる場合は、基本認証と一緒に SSL 暗号化を使用して、ユーザー ID とパスワードを保護します。

サーバーによって認証情報が必要とされる要求をクライアントが行うと、サーバーは、401 の状況コード、認証エラーを示す理由フレーズ、および WWW-Authenticate ヘッダーを持つ HTTP 応答を送信します。大部分の Web クライアントは、この応答を処理するために、ユーザー ID とパスワードをエンド・ユーザーに要求します。

HTTP 基本認証の WWW-Authenticate ヘッダーの形式は次のとおりです。

```
WWW-Authenticate: Basic realm="Our Site"
```

WWW-Authenticate ヘッダーには領域属性が含まれます。領域属性は、ユーザー ID とパスワードが適用されるリソースのセットを示します。 Web クライアントは、このストリングをエンド・ユーザーに表示します。領域ごとに異なる認証情報が必要になる場合があります。Web クライアントは、各領域ごとに認証情報を保管しておくことができます。こうすることにより、エンド・ユーザーは、各要求のたびに情報を再入力する必要がなくなります。

Web クライアントはユーザー ID とパスワードを取得すると、Authorization ヘッダーを付けて元の要求を再送します。あるいは、クライアントは、もともとの要求を行うときに、Authorization ヘッダーを送信することもできます。サーバーがこのヘッダーを受け入れる場合は、ユーザー確認と応答の処理を回避できます。

Authorization ヘッダーの形式は次のとおりです。

```
Authorization: Basic userid:password
```

第 3 章 CICS Web サポートの概念と構造

CICS Web サポートは、CICS 領域が HTTP サーバーと HTTP クライアントのどちらの役割も果たせるようにする、CICS サービスの集合体です。

HTTP サーバーとしての CICS

CICS が HTTP サーバーである場合、Web クライアントは HTTP 要求を CICS に送信し、応答を受け取ることができます。応答は、文書テンプレートまたは静的ファイルから CICS が作成した静的応答か、ユーザー・アプリケーション・プログラムによって動的に作成された、アプリケーション生成の応答のいずれかです。

HTTP サーバーとしての CICS のアクションは、以下によって制御されます。

1. CICS Web サポートを構成し、CICS に対して要求および応答の処理方法を指示するために使用される、システム初期設定パラメーターおよびリソース定義 (TCPIPSERVICE 定義および URIMAP 定義を含む)。
2. HTTP 要求および応答の分析と処理に使用される、CICS ユーティリティー・プログラム。
3. HTTP 要求を受け取り、HTTP 応答に材料を提供するために使用される、ユーザー作成のアプリケーション・プログラム。これらのアプリケーション・プログラムは、CICS Web サポートで使用するように設計された Web 対応アプリケーション・プログラムか、あるいは、本来は CICS Web サポートで使用するようには設計されていない、Web 非対応の CICS アプリケーション・プログラムです。

HTTP サーバーとしての CICS Web サポートの動作は、RFC 2616 に記述されているように、条件付きで HTTP/1.1 の仕様に準拠しています。 HTTP 仕様についての詳細は、13 ページの『HTTP プロトコル』を参照してください。

HTTP クライアントとしての CICS

CICS が HTTP クライアントである場合、CICS 内のユーザー・アプリケーション・プログラムは、HTTP サーバーへの要求を開始し、その HTTP サーバーからの応答を受け取ることができます。

HTTP クライアントとしての CICS のアクションは、ユーザー作成のアプリケーション・プログラムによって制御されます。 **EXEC CICS** WEB アプリケーション・プログラミング・インターフェースには、アプリケーション・プログラムが CICS からの HTTP 要求を構成および開始するために使用できるコマンドや、サーバーから送信された応答の受け取りに使用できるコマンドが含まれています。 URIMAP リソース定義を使用しては、URL やクライアント証明書ラベルなどの情報を提供することができます。

CICS Web サポートと非 HTTP メッセージ

CICS Web サポートは、クライアントからの非 HTTP 要求もサポートします。CICS Web サポートの多数のコンポーネント (TCPIPSERVICE 定義、CICS ユーテ

ィリティー・プログラム、およびユーザー作成のアプリケーション・プログラムなど) を使用して、ユーザーが定義した任意の要求形式に対する要求処理を提供することができます。 CICS Web サポートによって処理される非 HTTP メッセージでは、TCPIPSERVICE リソース定義にある特殊なプロトコル (USER プロトコル) を使用するため、HTTP メッセージに対して CICS が実行する検査の対象とはなりません。

CICS Transaction Server for z/OS, バージョン 4 リリース 2 では、この機能は、非標準の要求形式を使用するユーザー作成クライアントからの要求をサポートすることを主に目的にしています。要求に対して行われる処理は、ユーザーによって定義されます。この機能は、クライアント/サーバー通信に使用される、正式に定義されているプロトコルに対する固有のサポートを提供するものではありません。

CICS Web サポートが非 HTTP メッセージに対して提供するサポートは、CICS 用の TCP/IP ソケット・インターフェースと同じものではありません。 z/OS Communications Server に提供されている IP CICS ソケット・インターフェースには、クライアントが TCP/IP を使用して CICS アプリケーション・プログラムと通信することを可能にする、アプリケーション・プログラミング・インターフェースがあります。 CICS Web サポートは、このプロセスには関与しません。「*z/OS Communications Server IP CICS ソケット・ガイド*」 (SC88-9053) に、この CICS ソケット・インターフェースについての説明があります。

CICS Web サポートのコンポーネント

CICS Web サポートには、CICS Web サポート・タスクのすべてで使用される基本コンポーネントの一部、および各 CICS Web サポート・タスクに対して選択および構成したタスク固有のコンポーネントの一部が含まれます。

基本コンポーネント

- CICS の **TCP/IP サポート**は、CICS SO (ソケット) ドメインにより、z/OS が提供するネットワーク・サービス (z/OS Communications Server および DNS サーバーへのアクセス) と共に提供されます。
- **z/OS UNIX システム・サービス**は TCP/IP サポートの一部として使用され、CICS 領域はこれらにアクセスする必要があります。
- **Secure Sockets Layer (SSL) サポート**を使用して、CICS Web サポートの実装のためにセキュリティーを提供します。CICS は Secure Sockets Layer (SSL) 3.0 プロトコル、および Transport Layer Security (TLS) 1.0 プロトコルをサポートします (SSL 2.0 はサポートされません)。 CICS の文書で SSL という用語が使用された場合、通常、SSL および TLS の両方を指すことに注意してください。 SSL および TLS について詳しくは、「*CICS RACF Security Guide*」を参照してください。
- **DOCCODEPAGE システム初期化パラメーター**は、CICS 文書テンプレートのサポートにより使用されるデフォルト・ホスト・コード・ページを指定します。
- **LOCALCCSID システム初期化パラメーター**は、(CICS がアプリケーション・プログラムのデフォルトと見なすコード・ページである) ローカル CICS 領域に対してコード化文字セット ID を指定します。
- **TCPIP システム初期化パラメーター**は、CICS TCP/IP サービスを開始するときにアクティブにします。

- **WEBDELAY システム初期化パラメーター**は、Web 3270 ブリッジ機能が含まれる場合にのみ、非アクティブな CICS Web タスクのタイムアウト期間を定義します。他の CICS Web タスクのタイムアウトは、関連トランザクションの RTIMOUT 値、または (HTTP サーバーとしての CICS の) TCPIPSERVICE 定義の SOCKETCLOSE 属性により処理されます。
- **Socket リスナー・タスク (CSOL)** は、インバウンド TCP/IP 接続要求を検出して、Web 接続タスクを接続することにより、CICS Web サポートを起動します。
- **Web 接続タスク (CWXN、CWXU、または別名)** は、データを Web クライアントから受信して、URIMAP マッチング、HTTP ヘッダーのコード・ページ変換、要求の分析、およびメッセージ・ボディのコード・ページ変換を含む要求の初期処理を処理します。また、タスクは Web クライアントから受信したチャンク化済みメッセージおよびパイプライン・メッセージを前処理します。(URIMAP 定義を使用して) 静的応答が送信されると、Web 接続タスクはこの処理も同様に処理します。

リソース定義

- **TCPIPSERVICE リソース定義**を使用して、HTTP サーバーとしての CICS に使用する各ポートを定義します。これには、そのポートの接続のセキュリティー・オプション、およびインバウンド要求のタイムアウトと最大サイズ制限が含まれます。この定義は、HTTP クライアントとしての CICS では使用されません。

  注: TCPIPSERVICE リソース定義は、CICS 提供の TCP/IP サービスのみと共に使用され、z/OS Communications Server IP CICS ソケット・インターフェースとは関係ありません。CICS の TCP/IP ソケット・インターフェースは、z/OS Communications Server と共に提供され、z/OS の不可欠な部分であり、CICS SO ドメインを使用しません。
- **URIMAP リソース定義**は、Web クライアントからの要求、または HTTP サーバーへの要求の URL を突き合わせ、CICS に要求の処理方法についての情報を提供します。URIMAP 定義は、TCPIPSERVICE 定義に関連付けられたアナライザー・プログラムによって CICS Transaction Server for z/OS バージョン 3 リリース 1 の前に提供された CICS Web サポート処理機能を取り込んで置き換えることができます。また、URIMAP 定義を使用して、アプリケーション・プログラムを使用せずに、静的応答を Web クライアントからの要求に送信することもできます。
- **TRANSACTION リソース定義**を使用して、HTTP 要求処理の別名トランザクションを定義します。CICS はデフォルト別名トランザクションの CWBA のリソース定義を提供します。Web 接続タスクが要求の初期処理を完了すると、アプリケーション生成応答が生成された場合に、別名トランザクションが処理の残りのステージを処理します。この処理には、要求の受信、アプリケーションのビジネス・ロジックの実行、HTTP 応答の構造化、および HTTP 応答のコード・ページ変換が含まれます。

ユーザー・アプリケーション・プログラム

- **Web 対応アプリケーション・プログラム** EXEC CICS WEB および EXEC CICS DOCUMENT アプリケーション・プログラミング・インターフェースを使用して、CICS Web サポートのために設計できます。 HTTP サーバーとしての CICS の場合、これらのプログラムは HTTP 要求を受信して分析し、アプリケーション生成応答を Web クライアントに提供します。HTTP クライアントとしての CICS

に対して、CICS のユーザー・アプリケーション・プログラムは HTTP 要求をサーバーに送信し、サーバーから応答を受信できます。

- **COMMAREA アプリケーション**、 COMMAREA インターフェースを使用して、別のプログラムからリンクされるように設計されており、CICS Web サポートをコンバーター・プログラムと共に使用してアクセスされ、出力を Web クライアントへの送信のために HTML に変換できるプログラム。または、COMMAREA アプリケーションにリンクし、出力を使用して HTTP 応答を提供する Web 対応アプリケーション・プログラムを記述できます。
- **3270 表示アプリケーション**、 Web 端末変換アプリケーションを使用して 3270 端末と通信し、アクセスできるように設計されているプログラム。 Web 端末変換アプリケーションにより作成された HTML 出力は、Web ブラウザーで表示できます。

プログラミング・インターフェース

- **EXEC CICS WEB** アプリケーション・プログラミング・インターフェースは、HTTP 要求を解釈し、HTTP 応答を作成します。HTTP サーバーとしての CICS に使用されるもの、および HTTP クライアントとしての CICS に使用されるものがあり、両方とも CICS Web サポートを構成します。
- **EXEC CICS DOCUMENT** アプリケーション・プログラミング・インターフェースは、CICS 文書を構成して、CICS から送信された応答または要求ボディを提供します。

CICS Web サポート・ユーティリティー・プログラム

- **アナライザー・プログラム** TCPIPSERVICE 定義に関連付けられています。URIMAP 定義がアナライザー・プログラムの使用を指定した場合、または URIMAP 定義が存在しない場合に、HTTP 要求の解釈で使用されます。CICS はデフォルト・アナライザー・プログラムの DFHWBAAX を提供し、このプログラムは基本エラー処理を提供します。また、サンプル・アナライザー・プログラムの DFHWBADX も提供し、このプログラムは CICS TS 3.1 の前に CICS Web サポートが使用した URL 形式を使用して要求をサポートします。これらのアナライザーのどちらも独自のアナライザー・プログラムの基礎として使用できます。
- **コンバーター・プログラム** HTTP 要求のデコードおよびユーザー・アプリケーション・プログラムへの入力の構成に使用できます。Web 対応アプリケーション・プログラムは、通常、コンバーター・プログラムを必要としませんが、CICS Web サポートのために設計されていない Web 非対応アプリケーションで必要になる場合があります。 CICS は、コンバーター・プログラムを提供しません。コンバーター・プログラムを複数記述して、任意のコンバーター・プログラムを CICS 領域から選択し、要求を処理できます。
- **Web エラー・プログラム** 要求エラーまたは異常終了が CICS Web サポート処理で発生した場合に、エラー応答を Web クライアントに提供します。 CICS は、Web エラー・プログラムの DFHWBEP を提供し、このプログラムはほとんどのエラー状態で使用されます。また、Web エラー・アプリケーション・プログラムの DFHWBERX も提供し、このプログラムは URIMAP のマッチングが失敗した場合にデフォルト・アナライザーの DFHWBAAX と共に使用されます (他の状態のために指定することもできます)。Web エラー・プログラムはユーザーが置き

換えることができ、修正してカスタマイズしたり、各エラー状態で Web クライアントに送信するエラー応答を変更したりします。

- **Web 端末変換アプリケーション DFHWBTTA** (および代替処理 DFHWBTTB および DFHWBTTC の別名) を使用して、HTML 出力を 3270 端末との通信のために設計されたプログラムから作成できます。プログラムは、CICS 3270 ブリッジ機構を使用します。BMS を使用するアプリケーションも使用しないアプリケーションもサポートされます。この機能を使用するためにアプリケーション・プログラムを変更する必要はありません。
- **パスワード有効期限管理プログラム DFHWBPW** 基本認証が接続に対して指定されており、ユーザーのパスワードの有効期限が切れている場合に使用されます。このプログラムにより、ユーザーは新規パスワードを設定できます。DFHWBPW により、ユーザーに表示される Web ページをカスタマイズまたは置き換えることができます。

文書構成機能

- **z/OS UNIX システム・サービス・ファイル**を Web クライアントからの HTTP 要求への応答のボディとして使用できます。
- **文書テンプレートのサポート**。オフラインで作成した HTML のフラグメントからメッセージ・ボディを構築できるようにします。
- **BMS マクロ**。BMS マップ・セットから HTML 文書テンプレートを作成します。

コード・ページ変換

CICS は、ユーザー・アプリケーション・プログラムに適切なコード・ページ、またはインターネットで使用する場合に適切なコード・ページに HTTP メッセージを変換する機能を提供します。CICS は、z/OS 変換サービスを使用してコード・ページ変換を処理します。

コード・ページ変換テーブル (DFHCNV) は以前の CICS のリリースで必要でしたが、CICS Transaction Server for z/OS, バージョン 4 リリース 2 の CICS Web サポートでは通常必要ありません。ただし、以前の CICS のリリースでコーディングしたアナライザー・プログラムを使用して DFHCNV を参照する場合を除きます。この場合、コード・ページ変換テーブルの提供を継続するか、またはアナライザー・プログラムを更新する必要があります。詳しくは、67 ページの『コード・ページ変換テーブル (DFHCNV) のエントリーのアップグレード』を参照してください。

CICS Web サポートのタスク構造

HTTP サーバーとしての CICS に対して、CICS Web サポートが CICS 領域でアクティブになっているときは、インバウンド接続要求を listen する、ソケットからのデータの受信と初期処理を実行する、および要求との接続でアプリケーション・プログラムによって実行される作業をカバーするため、それぞれ別個のタスクが使用されます。 HTTP クライアントとしての CICS では、適用されるタスクは 1 つのみです。適用されるタスクは、HTTP 要求を作成するアプリケーション・プログラム用のタスクです。

Socket リスナー・タスク (CSOL)

これは、長時間実行の CICS タスクです。 CICS システムには Socket リスナー・タスクのインスタンスが 1 つあります。

このタスクは、CICS に対して定義されたすべてのポート上のインバウンド TCP/IP 接続要求を検出し、そのポートに関連する CICS サービスを起動します。ポートが CICS Web サポートを対象としている (つまり、プロトコルとして HTTP または USER が指定されている) 場合には、Web 接続タスクがそのポートの TCPIPSERVICE リソース定義でトランザクションとして定義されるため、リスナーはそのタスクに接続します。

Web 接続タスク (CWXN、CWXU、または別名)

ポートの TCPIPSERVICE 定義にプロトコル HTTP が含まれている場合、Web 接続タスクのデフォルトのトランザクション ID は CWXN です。プロトコルが USER の場合、デフォルトは CWXU です。別名を代わりに使用することもできますが、トランザクションは常にプログラム DFHWBXN を実行します。

ソケット・リスナー・タスクによって Web 接続タスクが呼び出される場合には、まず最初に SOCKET RECEIVE 要求を発行して、Web クライアントからデータを受信します。データを受信すると、Web 接続タスクは Web クライアントの要求の初期処理を行います。

- (HTTP プロトコル上の) HTTP 要求の場合、その初期処理には URIMAP マッチング、HTTP ヘッダーのコード・ページ変換、要求の分析、およびメッセージ・ボディのコード・ページなどが含まれます。またこのタスクは、Web クライアントから受信した、チャンク (大きい塊) 化されているパイプライン・メッセージの前処理も行います。アナライザー・プログラムを使用する場合、そのプログラムはこのトランザクションでカバーされます。
- (USER プロトコル上の) 非 HTTP 要求の場合は、初期処理は行われません。

(URIMAP 定義を使用して) 静的応答が HTTP 要求に送信される場合は、Web 接続タスクがこの処理も行います。アプリケーション生成の応答が必要な場合は、Web 接続タスクは別名トランザクションに接続します。

最初の処理ステージにある Web クライアントからのそれぞれの要求ごとに、Web 接続タスクのインスタンスがあります。 CICS Transaction Server for z/OS バージョン 3 リリース 1 の前には、Web クライアントと CICS の間に持続接続がある場合は、その持続接続が存在する間は CWXN トランザクションがシステム内に残っていました。現在は、Web クライアントからの要求が別名トランザクションに渡されるか、あるいは静的応答が送信されると、CWXN トランザクションは終了します。ソケット・リスナー・タスクはソケットをモニターし、持続接続上のそれぞれの要求ごとに、CWXN の新規インスタンスを開始します。この動作は、非同期受信と呼ばれるもので、システムに残っている CWXN トランザクションが別名トランザクションに接続してさらに要求を処理することができない場合に、最大タスク指定 (MXT) の限界に達した状況において、デッドロックが発生する可能性を回避します。

アプリケーション生成の応答用の別名トランザクション

アプリケーション生成の応答が作成される場合、Web 接続タスクによる要求の最初の処理が完了すると、Web 接続タスクは、その要求の残りの処理ステージ用に指定されている別名トランザクションに接続します。CICS はデフォルト別名トランザクションの CWBA のリソース定義を提供します。静的応答が提供される状況では、別名トランザクションは使用されません。

別名トランザクションは、アプリケーション生成の応答の各処理ステージの処理を行います。この処理ステージには、要求の受信、アプリケーションのビジネス・ロジックの実行、HTTP 応答の構成、および HTTP 応答のコード・ページ変換などがあります。要求の処理にコンバーター・プログラムを使用する場合は、これも別名トランザクションによって処理されます。これらの処理ステージにある各 HTTP 要求ごとに、別名トランザクションのインスタンスが 1 つあります。

HTTP クライアントとしての CICS

HTTP クライアントとしての CICS の場合、HTTP クライアント要求を作成するアプリケーション・プログラムによって起こされるすべてのアクティビティーが単一のタスクでカバーされます。このアクティビティーには、アプリケーション・プログラムのアクション、要求の送信および応答の受信における CICS のアクション、およびソケット・アクティビティーなどがあります。アプリケーション・プログラムが、**EXEC CICS** LINK コマンドを使用して別のプログラムにリンクする場合は、それもこのタスクによってカバーされます。このタスクには、アプリケーション・プログラムを起動するトランザクション ID があります。

タスクは、そのアプリケーション・プログラムのアクティビティーの最初から最後まで、システム内に存在します。タスクには、複数の要求と応答が含まれる可能性があります。また、アプリケーション・プログラムは、1 つのサーバーに対して複数の接続を開き、維持する可能性があります。タスクが終了すると、開いている接続はすべて自動的に閉じられます。

HTTP サーバーとしての CICS に対する HTTP 要求および応答の処理

HTTP サーバーとしての CICS に対する HTTP 要求は、CICS に要求を行う Web クライアントから開始されます。CICS は Web クライアントに、その Web クライアントが行った要求に対する応答を提供します。この応答は、URIMAP リソース定義によって識別される静的文書から作成するか、またはユーザー・アプリケーション・プログラムによって動的に作成することができます。

32 ページの図 2 に、Web クライアントからの要求を受信し、応答を提供するために CICS Web サポートが実行する処理を示しています。

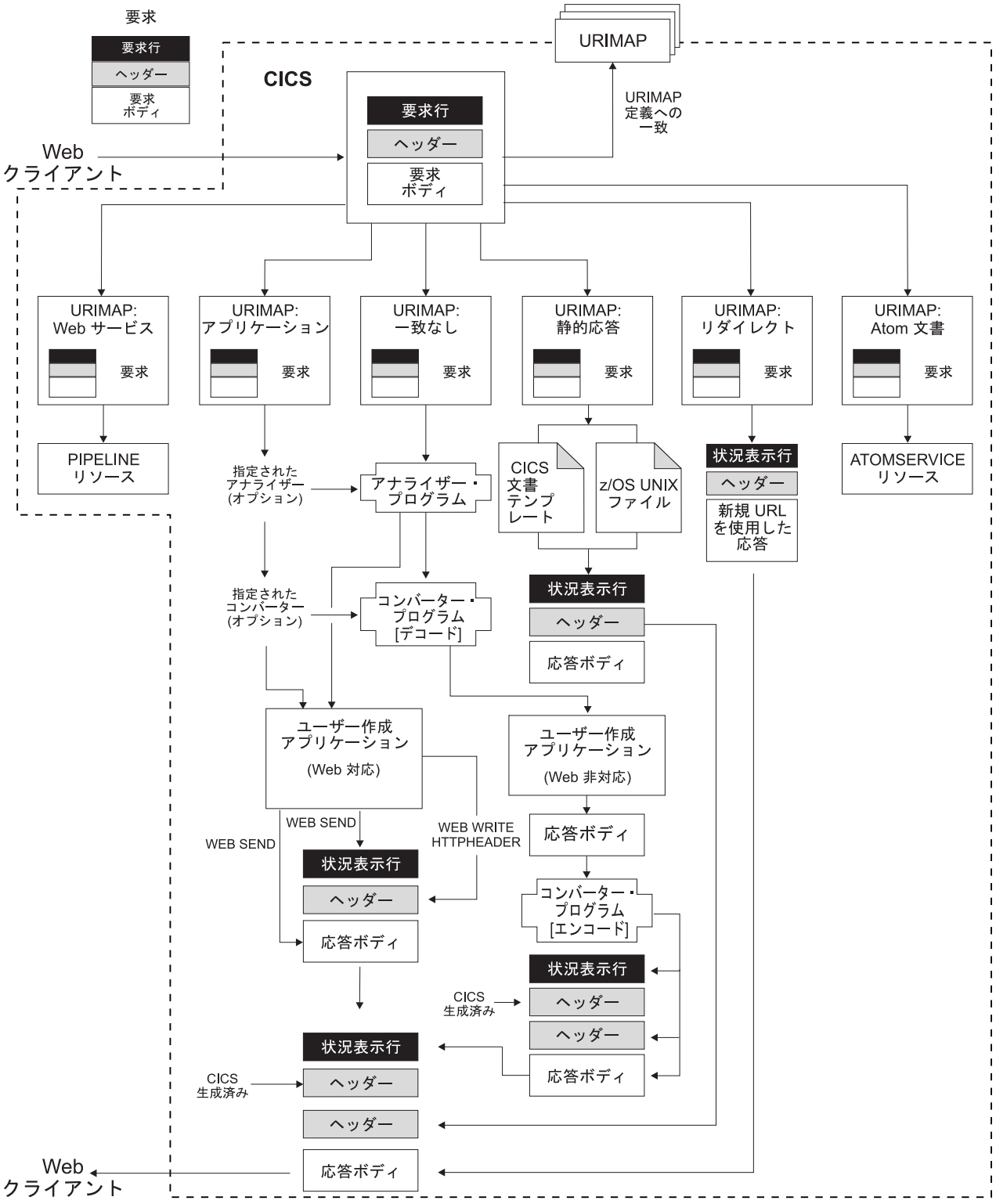


図 2. HTTP サーバーとしての CICS に対する処理

HTTP サーバーとしての CICS に対する処理は、以下のように行われます。

1. **CICS が TCP/IP 接続要求を受信します。** CICS ソケット・ドメインはポートの TCPIPSERVICE リソース定義を使用して、CICS Web サポートによりその

要求を処理する必要があることを決定します。TCPIPSERVICE 定義では、要求に対して適用されるセキュリティー属性、および要求メッセージを受信するためのタイムアウト設定が指定されており、単一の要求に対して受信することのできる最大データ量が制限されています。

2. **CICS は、使用可能な場合は要求の URL を URIMAP に突き合わせます。** CICS は、TCPIPSERVICE 定義に関連しており、HTTP サーバーとしての CICS に適用される URIMAP リソース定義に対して、 HTTP 要求で指定されている URL を突き合わせます。 両者が一致した場合は、URIMAP 定義で CICS が要求を処理する方法が決まります。一致しなかった場合、CICS は、アナライザー・プログラムによる処理ステージ 5 から開始するデフォルトのプロセスを続行します。
3. **URIMAP 定義でリダイレクトが指定されている場合、CICS は、指定された URL に Web クライアントをリダイレクトします。** CICS はリダイレクト・メッセージを作成し、そのリダイレクト・メッセージを Web クライアントに転送します。これにより、その HTTP 要求に対する処理が完了します。
4. **URIMAP 定義が Atom 文書に関連している場合、CICS は要求を処理する指定の ATOMSERVICE リソースを見つけます。** Atom 文書の処理については、254 ページの『CICS における Atom フィードの処理』で説明されています。
5. **URIMAP 定義が Web サービスに関連している場合、CICS は要求を処理する指定の PIPELINE リソースを見つけます。** Web サービスのための処理については、*CICS Web サービス・ガイド*で説明しています。
6. **URIMAP 定義で静的応答が指定されている場合、CICS は応答を構成して、それを返します。** CICS では、文書テンプレートまたは z/OS UNIX システム・サービス・ファイルとともに、適切な HTTP ヘッダーを使用することで、HTTP 応答を構成します。この応答は適切なコード・ページ変換処理され、CICS によって Web クライアントに送信されます。これにより、その HTTP 要求に対する処理が完了します。
7. **URIMAP 定義でアナライザー・プログラムの使用が指定されている場合、または一致する URIMAP 定義が見つからなかった場合は、そのアナライザー・プログラムを実行することができます。** アナライザー・プログラムは要求を動的に解釈するか、またはそれをモニター目的または監査目的に使用することができます。

   URIMAP 定義が要求に対してセットアップされていない場合は、TCPIPSERVICE 定義に対応するアナライザー・プログラムを、要求の処理パスで使用する必要があります。コード・ページ変換、または CICS TS バージョン 3 よりも前の互換処理に対して特殊な要件を持つ、Web 非対応アプリケーション・プログラムを使用している場合も、このアナライザー・プログラムが必要になることがあります (441 ページの『付録 E. アナライザー・プログラム』では、これらの状態について説明しています)。 それ以外の場合は、アナライザー・プログラムを使用する、または使用しないは任意ですが、 URIMAP 定義が見つからなかった場合は、アナライザー・プログラムが呼び出されて要求が処理されることに注意してください。

アナライザー・プログラムが使用されている場合、HTTP 要求および HTTP ヘッダーはそのアナライザー・プログラムに渡されます。アナライザー・プログラムはその要求を解釈して、以下を決定することができます。

- 要求の処理にどの CICS リソースを使用するか。
- どのユーザー ID を要求に関連付けるか。
- 残りの処理ステージのどのステージが必要であるか。

8. **要求のデコード、およびアプリケーション・プログラムへの入力の構成にコンバーター・プログラムを使用することができます。** Web 対応アプリケーション・プログラムは、デコードなしで HTTP 要求を受け取る必要があります。ただし、COMMAREA 入力を要求する Web 非対応アプリケーション・プログラムを使用して HTTP 要求を処理する場合は、コンバーター・プログラムを使用して要求をデコードし、アプリケーション・プログラムの要件を満たす入力を構成することができます。コンバーター・プログラムは、URIMAP 定義を使用して指定するか、またはアナライザー・プログラムによって選択することができます。
9. **要求を処理するために、アプリケーション・プログラムが実行されます。** アプリケーション・プログラムは、URIMAP 定義またはアナライザー・プログラムを使用して指定することができます。**EXEC CICS** WEB および **EXEC CICS** DOCUMENT アプリケーション・プログラミング・インターフェースを使用する Web 対応アプリケーション・プログラムを使用して要求を処理し、応答を構成することができます。Web 非対応アプリケーション・プログラムは、コンバーター・プログラム (Web クライアントの要求を、受け入れ可能な入力に変換し、そのプログラムの出力に基づいて HTTP 応答を作成します)、または Web 非対応プログラムを呼び出してその出力を使用する Web 対応アプリケーション・プログラムのいずれかを使用して、Web 対応にすることができます。

   アプリケーション・プログラムは、別名トランザクションの下で実行します。

   アプリケーション・プログラムは以下のタスクを実行できます。

   - アプリケーション・プログラムが Web 対応である場合は、このアプリケーション・プログラムで要求の HTTP ヘッダーを調べて、要求行から情報 (照会ストリングなど) を抽出し、要求のボディを処理するためにバッファーに受け取り、応答の状況表示行の状況コードおよびテキストを選択して、応答用の HTTP ヘッダーを作成することができます。応答の構成には、WEB SEND や WEB WRITE HTTPHEADER などの **EXEC CICS** WEB API コマンドが使用されます。
   - アプリケーションは、Web 対応であるか非対応であるかに関係なく、応答のボディを構成する出力を作成することができます。Web 対応アプリケーション・プログラムは、CICS 文書テンプレートから形成されたエンティティー・ボディ、またはデータのバッファーからで形成されたエンティティー・ボディを作成することができます。Web 対応でないアプリケーション・プログラムは、Web 対応アプリケーション・プログラムまたはコンバーター・プログラムによってエンティティー・ボディへ変換することが可能な出力を作成することができます。
10. **コンバーター・プログラムを使用して、アプリケーション・プログラムからの出力をエンコードし、HTTP 応答を構成することができます。** アプリケーション・プログラムが Web 対応ではなく、その出力が Web クライアントに送信す

るための正しい形式でない場合は、コンバーター・プログラムを使用して、状況表示行および HTTP ヘッダーを含む適切な HTTP 応答を作成することができます。コンバーター・プログラムは、必要な場合は、出力に対して他のタイプの処理を実行することもできます。

コンバーター・プログラムで、処理ステージ 6 (デコード、またはコンバーター・プログラムを使用する他の処理)、7 (アプリケーション・プログラム)、および 8 (エンコード、またはコンバーター・プログラムを使用する他の処理) を繰り返すことを指定することができます。コンバーター・プログラムはアプリケーション・プログラムの名前を変更することができるため、この機能を使用して、複数のアプリケーション・プログラムで同じ要求を順に処理し、単一の応答を提供することができます。

11. **CICS Web サポート・プロセスで要求エラーまたは異常終了が発生した場合は、Web クライアントにエラー応答が送信されますが、この動作は、ユーザーによる置換が可能な Web エラー・プログラムを使用してカスタマイズすることができます。** DFHWBEP または DFHWBERX はエラー状態に関する情報、および CICS が Web クライアントに送信する予定のデフォルトの HTTP 応答 (状況コードおよび状況テキストを含む) を受信します。ユーザーによる置換が可能なプログラムで応答をカスタマイズするか、新しい応答を作成し、その応答を CICS に返すことができます。

    Web エラー・プログラムは、すべてのエラー状態で使用されるわけではありません。Web エラー・プログラムが使用されるのは、要求の初期処理で問題が発生した場合、および後続の処理で異常終了または障害が発生した場合です。このエラー・プログラムは、 (ユーザー作成アプリケーション・プログラムによる処理などの) 処理が正常に完了した状態や、意図していた結果が、エラーまたはリダイレクト応答であるような状態には、使用されません。

12. **CICS はいくつかの必須の HTTP ヘッダーを生成し、それらのヘッダーをメッセージに追加します。** 応答の HTTP バージョンに応じて、適切なヘッダーが生成されます。応答が HTTP/1.1 の場合、CICS は、HTTP/1.1 メッセージのために必要なヘッダーを追加します。応答が HTTP/1.0 の場合、クライアントが持続接続を要求すると、CICS は Connection: Keep-Alive ヘッダーとそれ以外のわずかのヘッダーを追加します。これらのヘッダーのうちいくつかのヘッダー (Date ヘッダーなど) の値は、CICS によって直接生成され、それ以外の値は、(WEB SEND コマンドを使用する) Web 対応アプリケーション・プログラム、または URIMAP 定義によって提供される情報に基づいています。これらのヘッダーは、Web 対応アプリケーションからの出力とコンバーター・プログラムからの出力の両方に追加することができます。

13. **CICS は、Web クライアントに完全な HTTP 応答を送信します。** Web クライアントが持続接続をサポートしている場合、ユーザー・アプリケーションまたは Web クライアントがクローズを要求するか、またはタイムアウト期間に達するまで、CICS はさらに送信されてくる可能性のある HTTP 要求のためにこの接続を開いたままにしておきます。

この処理では、CICS Web サポートの処理およびユーザー作成アプリケーション (一般に EBCDIC エンコードを使用) が Web クライアント (一般に ASCII エンコードを使用) と通信できるように、CICS 環境にメッセージが入るときと出るときに、通常はコード・ページ変換が必要になります。 49 ページの『CICS Web サポ

ートのコード・ページ変換』で、この変換がいつどのように行われるかを説明しています。必要なコード・ページ変換のタイプは、WEB SEND または WEB RECEIVE コマンドのオプションを使用して指定することができます。

HTTP クライアントとしての CICS に対する HTTP 要求および応答の処理

HTTP クライアントとしての CICS の場合、 CICS は Web クライアントであり、HTTP サーバーと通信します。ユーザー作成アプリケーション・プログラムは、CICS を介して HTTP サーバーに要求を送信し、そのサーバーから応答を受信します。CICS は、サーバーとの持続接続を維持します。アプリケーション・プログラムによって発行されたコマンドではセッション・トークンを使用し、その接続を識別します。

HTTP 要求を実行して応答を受信するアプリケーション・プログラムは、 **EXEC CICS** WEB API コマンドを使用して、サーバーとの対話を明示的に指図する必要があります。 HTTP 要求の実行には Web 対応のアプリケーション・プログラムが使用可能であり、その結果を処理して Web 対応ではないアプリケーションに情報を提供します。

HTTP 要求を開始するアプリケーション・プログラムは、 CICS がその要求の応答としてサーバーから受信するものはすべて処理するよう設計する必要があります。これには、エラー応答、別の URL へのリダイレクト、埋め込みハイパーテキスト・リンク、HTML フォーム、イメージ・ソース、またはアプリケーション・プログラムからアクションを要求するその他の項目などがあります。 CICS は必要に応じて、要求および応答のコード・ページ変換を実行できます。

37 ページの図 3 には、このトピックで説明されている処理が示されています。

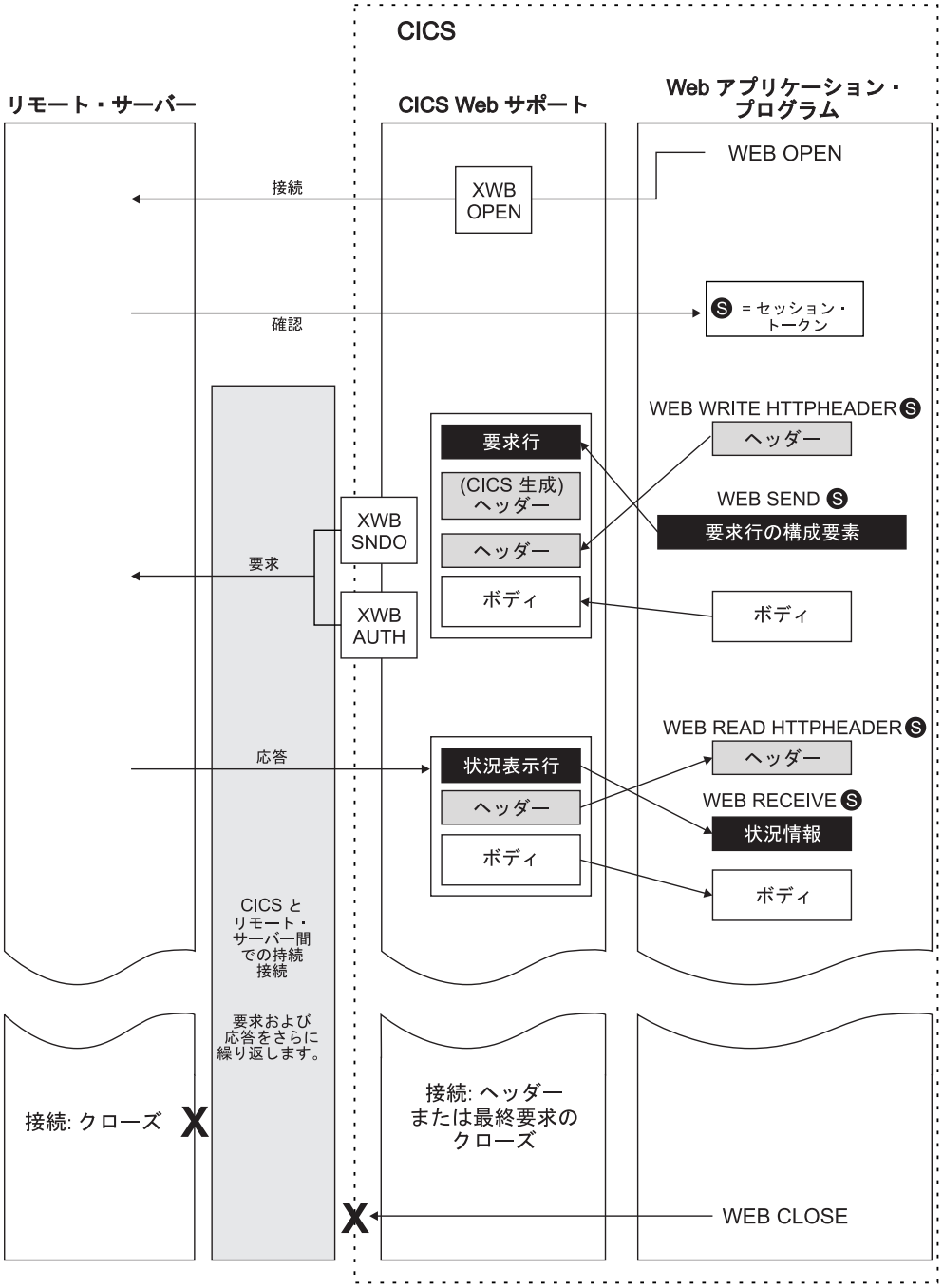


図 3. HTTP クライアントとしての CICS に対する処理

HTTP クライアントとしての CICS の処理は、以下のように実行されます。

1. **アプリケーション・プログラムが、CICS 経由で HTTP サーバーへの接続を開始します。** アプリケーション・プログラムは、**EXEC CICS** WEB OPEN コマンドを発行することにより、これを実行します。作成した URIMAP リソースを参照して、その接続のスキームおよびホスト名を指定するか、またはアプリケーション・プログラムでこの情報を指定することができます (URIMAP リソースについて詳しくは、URIMAP リソース定義を参照してください。) アプリケーションは、一度に複数の接続をオープンさせることが可能です。

2. **CICS が、サーバーとの接続を確立するか、適切なプール接続がないか検査します。** アプリケーション・プログラムによって提供された情報を使用し、CICS はソケットで TCP/IP 接続をオープンしてサーバーと通信します。この処理では、必要に応じて XWBOPEN ユーザー出口が使用され (ENABLE PROGRAM コマンドを使用して活動化されていた場合)、プロキシー・サーバーを経由したアプリケーション・プログラムの要求のリダイレクト、およびホストへの接続に対するセキュリティー・ポリシーの適用が実行されます。あるいは、接続プーリング (SOCKETCLOSE 属性) が指定された URIMAP リソースをアプリケーション・プログラムが使用していた場合、CICS は、再使用できる休止接続がプールにあるかどうかを検査します。 TCP/IP 接続が確立されるか、既存の接続が再使用のためにアプリケーションに提供されると、CICS は、その接続を使用時に一意的に識別するためのセッション・トークンをアプリケーション・プログラムに返します。このセッション・トークンは、その接続で接続するアプリケーション・プログラムによって発行される残りのすべてのコマンドで使用されます。 See 39 ページの『セッション・トークン』 for more information about the session token.
3. **アプリケーション・プログラムは、その要求の HTTP ヘッダーを書き込むことができます。** ユーザー作成の HTTP ヘッダーは、WEB WRITE HTTPHEADER コマンドを使用して作成可能であり、送信用に保管されます。
4. **アプリケーション・プログラムは、要求行の構成要素を指定します。** 要求のメソッド、パス情報、および照会ストリングは、 WEB SEND または WEB CONVERSE コマンドを使用して指定されます。その要求の HTTP バージョンは、 CICS により提供されます。
5. **アプリケーション・プログラムは、その要求のエンティティー・ボディを作成できます。** 要求の内容は、WEB SEND または WEB CONVERSE コマンドを使用して指定されます。その内容は、CICS 文書から (DOCUMENT インターフェースを使用して) 形成されるか、またはバッファーの内容から形成されます。サーバーが HTTP/1.1 である場合、バッファーの内容から形成された要求ボディに対してチャンク転送コーディングが使用されることがあります (ただし、CICS 文書については、使用されません)。
6. **アプリケーション・プログラムは、要求の送信を開始します。** アプリケーション・プログラムが WEB SEND または WEB CONVERSE コマンドを発行すると、その要求が CICS に受け渡され、セッション・トークンで指定された接続を使用して送信されます。
7. **CICS は、いくつかの必要な HTTP ヘッダーを生成して要求に追加してから、その要求をサーバーに送信します。** 一部のヘッダーの値は CICS によって直接生成され (例えば、Date ヘッダーなど)、その他の値は、アプリケーション・プログラムにより (WEB SEND または WEB CONVERSE コマンドを使用して) 提供された情報、または URIMAP リソースにより提供された情報に基づきます。要求の送信時には、必要に応じて次の 2 つのユーザー出口が起動されることがあります。 XWBSNDO は、個々の要求に対してセキュリティー・ポリシーを適用する場合に呼び出され、 XWBAUTH は基本認証で必要なユーザー名およびパスワードの詳細を指定します。
8. **サーバーは、要求を受信して処理し、応答を提供します。** CICS は、その応答をアプリケーション・プログラムに受け渡します。
9. **アプリケーション・プログラムは、その応答を検査します。** WEB READ HTTPHEADER コマンドまたは HTTP ヘッダーのブラウズ・コマンドを使用す

ることにより、その応答のヘッダーを検査できます。 WEB RECEIVE または WEB CONVERSE コマンドは、アプリケーション・プログラムで処理可能な応答のボディ (それが 1 つである場合)、およびその応答の状況コードと状況テキストを受信します。

10. **アプリケーション・プログラムは、さらに要求および応答を開始できます。** サーバーで持続接続がサポートされている場合は、セッション・トークンによって識別される接続が追加の要求用にオープンしたまま残ります。サーバーが持続接続をサポートしていない場合、サーバーは接続を閉じるよう CICS に指示します。
11. **アプリケーション・プログラムが、その接続使用を完了します。**要求と応答がすべて完了すると、アプリケーション・プログラムが WEB CLOSE コマンドを発行してその接続使用を終了します。接続プーリングが指定された URIMAP リソースを使用して接続が開かれていて、サーバーもアプリケーション・プログラムも接続を閉じる要求を行っていなければ、CICS は接続を閉じません。代わりに、CICS は接続が良好な状態であるかどうかを検査したうえで、それを休止接続のプールに入れます。プールされた接続を別のアプリケーション・プログラム、または同じアプリケーション・プログラムの別のインスタンスが再使用して、同じサーバーに接続することができます。接続が接続プーリングに適していない場合 (接続が閉じられている場合や URIMAP リソースを使用して開かれていない場合、または良好な状態でない場合)、CICS は接続を閉じます。

この処理では、CICS Web サポートの処理およびユーザー作成アプリケーション (一般に EBCDIC エンコードを使用) が HTTP サーバー (一般に ASCII エンコードを使用) と通信できるように、CICS 環境にメッセージが入るときと出るときに、通常はコード・ページ変換が必要になります。 49 ページの『CICS Web サポートのコード・ページ変換』で、この変換がいつどのように行われるかを説明しています。必要なコード・ページ変換のタイプは、WEB SEND、WEB RECEIVE または WEB CONVERSE の各コマンドのオプションを使用して指定できます。 HTTP クライアントとしての CICS の場合、デフォルトでは、メッセージの送受信時にはコード・ページ変換は実行されます。

セッション・トークン

セッション・トークンは 8 バイトのバイナリー値であり、HTTP クライアントとしての CICS と HTTP サーバーの間で使用中のクライアント HTTP 接続を一意的に識別するためのものです。アクティブな接続ごとにセッション・トークンが使用されるということは、さまざまなアプリケーション・プログラムによって使用されている接続を CICS Web サポートが区別できるということを意味します。また、アプリケーション・プログラムが複数の接続を制御できるということにもなります。

接続は、ユーザー・アプリケーション・プログラムが発行する **WEB OPEN** コマンドに対する応答で開始されます。 **WEB OPEN** コマンドが正常終了するとセッション・トークンが返され、その接続に関してアプリケーション・プログラムが発行するすべての **EXEC CICS WEB** コマンドで使用されます。

接続を使用して、ユーザー・アプリケーション・プログラムはサーバーへの HTTP クライアント要求を行い、サーバーからの応答を受信します。接続は、要求と応答の複数回の交換にわたって持続できます。 CICS Web サポートによる持続接続の処

理、および持続接続の終了について詳しくは、 46 ページの『CICS Web サポートによる持続接続の処理方法』を参照してください。

サーバーが接続を終了すると、アプリケーション・プログラムがその接続を使用してそれ以上要求を送信することはできませんが、サーバーが接続を終了する前に送信した応答を読むことはできます。アプリケーションが **WEB CLOSE** コマンドを発行するまで、セッション・トークンは引き続き有効であり、そのデータにアクセスするコマンドで使用できます。

WEB CLOSE コマンド発行後は、接続に適用されていたセッション・トークンは無効になります。アプリケーション・プログラムが **WEB CLOSE** コマンドを発行しなければ、セッション・トークンはタスクの終了時に無効になります。クライアント HTTP 接続用の接続プーリングが実装されている場合、CICS は、別のアプリケーションまたは同じアプリケーションの別のインスタンスが再使用できるように、休止状態の接続をプールすることがあります。接続がプールされると、セッション・トークンは持続しなくなります。したがって、クライアント接続を再使用するアプリケーションには、その接続使用のための新しいセッション・トークンが与えられます。

1 つの CICS 領域に同時に存在できる開かれた (セッション・トークンを与えられて使用中の) HTTP クライアント接続の最大数は、32,768 です。

CICS Web サポートの URL

CICS Web サポートによって提供されるリソースの要求 URL では、URL のパス構成要素はユーザーに任されています。 CICS Web サポートでは、URIMAP 定義またはアナライザー・プログラムによって、要求 URL と CICS が提供するリソースの間のリンクが作成されるため、この URL が CICS リソースと直接の関係を持つ必要はありません。ただし、処理目的または管理目的で情報を提供するよう URL を設計することは可能です。

11 ページの『URL の構成要素』では、要求 URL のさまざまな構成要素および各構成要素の役割について説明しています。

リソース ID の新しい形式 IRI (Internationalized Resource Identifier) は、英語以外の各国語に適した文字およびフォーマットの使用を許可します。 IRI は、要求および応答に関係するアプリケーションがこれをサポートする URI または URL の代わりに使用できます。 CICS は、URIMAP リソース定義での IRI の使用をサポートします。 IRI について詳しくは、 262 ページの『Internationalized Resource Identifiers (IRI)』を参照してください。

アプリケーション生成応答の URL

要求 URL における情報は、アナライザー・プログラムおよびユーザー作成アプリケーション・プログラムで使用することができます。

要求の処理パスでアナライザー・プログラムが使用されている場合は、追加処理を行うために指定するプログラムおよびトランザクションをアナライザー・プログラムに示す URL を設計することができます。CICS 提供のサンプル・アナライザー・プログラム DFHWBADX では、`/converter/alias/program/other path`

information という形式のパス構成要素を含む URL を分析することができます。ここで、converter は別名トランザクションを指定し、alias は別名トランザクションを指定し、program はユーザー・アプリケーション・プログラムを指定し、other path information は、アナライザーが使用しない追加情報を提供します。

応答を提供する Web 対応アプリケーション・プログラムでは、URL のパス構成要素からの情報を使用することもできます。アプリケーションは、WEB EXTRACT コマンドを使用してパス構成要素を抽出し、そのパス構成要素を分析して、適切なアクションを決定することができます。例えば、パス構成要素を使用して、アプリケーションが提供している特定の機能を指定することができます。または、Web 対応アプリケーションが他の複数のアプリケーションにフロントエンドを提供している場合、URL のパス構成要素によって、要求の適用先アプリケーションを識別することができます。

URIMAP 定義を使用して管理されるアプリケーション生成の応答の場合、URL のパス構成要素は、複数の要求 URL を同じアプリケーションにマップするよう設計することができます。これを行うには、URL のパス構成要素がそれぞれ同じように始まるようにし、ワイルドカードを含む単一の URIMAP 定義を作成して、すべての要求 URL を単一のリソースにマップします。例えば、パス /staffapps/ordering/* および関連するアプリケーションを指定している URIMAP マップ定義を作成することにより、/staffapps/ordering/ で始まるパスを持つすべての要求を特定の CICS アプリケーションにマップすることができます。これにより、このアプリケーションは、URL の残りの部分にある情報を抽出して分析し、要求ごとに適切なアクションを決定することができます。

静的応答の URL

CICS Web サポートでは、URL が CICS リソースと直接関係を持つ必要はありません。静的応答でこのことが意味するのは、応答を提供するファイルへの絶対パスが URL に含まれている必要はないということです。その代わりに、URIMAP 定義は要求 URL を適切なファイルに突き合わせます。

ただし、z/OS UNIX ファイルが静的応答として使用されている場合は、要求 URL のパス構成要素が z/OS UNIX で使用されているディレクトリーに一致するよう、パス構成要素を設計することができます。CICS Web サポートにより提供されるすべての z/OS UNIX ファイルが同一のディレクトリーのサブディレクトリー (例えば、CICS 領域ユーザー ID のホーム・ディレクトリー) にある場合は、このディレクトリーを省略し、要求 URL をこれらのファイルへのパスの残りの部分に一致させることができます。例えば、ホーム・ディレクトリーが /u/cts/CICSHome で、静的応答として以下の z/OS UNIX ファイルを提供する場合、

```
/u/cts/CICSHome/FAQs/ordering.html
/u/cts/CICSHome/help/directory/viewing.html
```

以下のような要求 URL を使用することができます。

```
http://www.example.com/faqs/ordering.html
http://www.example.com/help/directory/viewing.html
```

URL のパス構成要素は大/小文字を区別するので、z/OS UNIX の名前も大/小文字を区別します。URL は通常、小文字で指定されます。URIMAP 定義で各項目を指定する場合、特にファイル名が大/小文字混合で URL が小文字の場合は、大/小文字を正しく使用してください。

以下の場合には、使用しているファイル・ディレクトリー構造に要求 URL を一致させることもできます。

- リソースの管理をより簡単にする場合。
- Web サーバーの標準的な慣例に従う場合。
- 作成する必要のある URIMAP 定義の数を減らす場合。

ワイルドカードを含む単一の URIMAP 定義を作成し、パス突き合わせのメカニズムを使用して複数の静的応答を配信することができます。これが可能となるのは、それらのすべての静的応答の URL のパス構成要素が同じように始まる場合、およびそれらの静的応答のファイルが同じ z/OS UNIX ファイル・ディレクトリーに保管されている場合です。ワイルドカードは URL のパス構成要素の最後で使用されますが、z/OS UNIX ファイルのファイル・パスの最後でも使用されます。上の例では、FAQs ディレクトリーに保管されているすべての HTML 文書は、パス /faqs/* を指定し、HFSFILE 属性を /u/cts/CICSHome/FAQs/* と指定している単一の URIMAP 定義で提供することができます。名前が同じように始まる CICS 文書テンプレートでも同様の手法を使用することができます。静的応答の URIMAP 定義では、メディア・タイプ (例えば text/html) が指定されるので、この方法で別のファイル・タイプを提供する必要がある場合は、それらのファイル・タイプがそれぞれ別個のディレクトリーに保管されていることを確認してください。

照会ストリング

要求 URL 内の照会ストリングを使用すると、代替の URIMAP 定義を選択することができます。URIMAP の突き合わせに照会ストリングを使用するには、URIMAP 定義のパス属性に、パスそのものと併せて、完全で正確な照会ストリングをで指定する必要があります。

アプリケーション生成の応答の場合、アプリケーションは WEB EXTRACT コマンドまたは WEB READ FORMFIELD コマンドを使用して、照会ストリングから情報を抽出し、その情報を分析することができます。これは、照会ストリングが URIMAP の突き合わせに使用されていたかそうでないかにかかわらず実行することができます。

文書テンプレートで静的応答を提供する場合、CICS は、照会ストリングの内容を、指定された CICS 文書テンプレートにシンボル・リストとして自動的に挿入します。文書テンプレートで照会ストリングの内容を使用する場合は、文書テンプレートに適切な変数を組み込んで、照会ストリングの内容の代わりに使用することができます。これを行うことができるのは、照会ストリングがまだ URIMAP の突き合わせに使用されていない場合のみです。

URL の長さ: CICS Web サポート

CICS Web サポートでは URL の長さに以下の制限があります。

- インバウンド HTTP 要求の URL の場合 (HTTP サーバーとしての CICS の場合)、CICS は最大で 32K の長さまで受け入れます。この長さは、少なくとも一般的に使用されている Web ブラウザー・クライアントでサポートされている長さの 8 倍になります。 CICS は、処理可能な長さよりも長い URL を受信した場合、414 (Request-URI Too Long (要求 URI が長すぎる)) 状況コードを返します。
- (HTTP クライアントとしての CICS が行う) アウトバウンド HTTP 要求の URL の場合、CICS は、URIMAP リソース定義では最大 255 文字のパス構成要素をサポートします。要求を行うユーザー・アプリケーション・プログラムは、URIMAP 定義をオーバーライドし (またはまったく使用せずに)、それよりも長いパス構成要素を提供することがあります。サーバーが処理できる URL の長さを確認してください。

URL の長さ: URIMAP 定義

URIMAP 定義を使用して提供されるリソースの URL を選択する場合は、URL の長さに対する以下の追加制限に注意してください。

- CICS は URIMAP リソース定義では最大 255 文字のパス構成要素をサポートしています。これよりも長いパス構成要素は使用しないでください。HTTP/1.1 仕様書では、全体の長さが 255 文字よりも長い URL (スキーム、ホストおよびパス構成要素、および区切り文字で構成されます) には注意が必要であると記述されています。その理由は、古い Web クライアントおよびプロキシーにはこれらを正しくサポートしていないものがあるからです。パーセント・エンコードされた Unicode 文字を含む IRI を使用している場合、ここでいう文字とは、元の Unicode 文字ではなく、単一の ASCII 文字を意味していることに注意してください。例えば、パーセント・エンコード表記のキリル文字 `%D0%B4` は、255 文字制限の中の 6 文字としてカウントされます。
- これよりも長いパス構成要素を使用する必要がある場合は、通常はそのようなパス構成要素を使用することができます。これは、URIMAP リソース定義では完全なパスを指定する必要がないからです。パスの最後にワイルドカード文字としてアスタリスク (*) を使用することができます。 URIMAP 定義が正しく動作するのは以下の場合です。
  - URL の指定部分がそのリソースに固有である。
  - URL の指定部分はそのリソースに固有ではないが、静的応答を提供し、パス突き合わせのメカニズムを使用して完全な URL が得られる。
- URIMAP の突き合わせを目的として照会ストリングを使用し、URIMAP 定義のパス属性でそれを指定している場合は、合計の長さが 255 文字の制限内に収まっている必要があります (パス構成要素の一部をアスタリスクで置き換えることは可能ですが、その場合正しく動作しますが、アスタリスクを照会ストリング内で使用することはできません)。この目的で照会ストリングを使用していない場合は、CICS Web サポート全体で 32K までという、URL 長に対する制限までは、任意の長さの照会ストリングを受け入れることができます。
- (URIMAP 定義の LOCATION および REDIRECTTYPE 属性を使用する) リダイレクトの場合、CICS は最大 255 文字のリダイレクト URL をサポートしています。この URL は、スキーマ、ホスト、およびパス構成要素、および適切な区切り文字を含む、**完全な** URL である必要があります。リダイレクトされたクライ

アントの宛先としてリソースを使用する場合は、そのリソースの完全な URL が、この 255 文字制限に収まるようにする必要があります。

CICS Web サポートによる、チャンク転送コーディングの処理方法

チャンク転送コーディングを使用するメッセージを、CICS で送信および受信することができます。

HTTP サーバーとしての CICS は、チャンク化済みメッセージを要求として受信したり、応答として送信したりできます。HTTP クライアントとしての CICS は、チャンク化済みメッセージを要求として送信したり、または応答として受信したりできます。CICS Web サポートはこれらのさまざまなケースを以下のように処理します。

- HTTP サーバーとしての CICS が、HTTP 要求としてチャンク化済みメッセージを受信すると、CICS Web サポートはそのチャンク・エンコード方式を認識します。CICS Web サポートは、すべてのチャンクが受信されるまで (これは、長さがゼロのチャンクが受信されたことで示されます) 待機し、それらのチャンクをアセンブルして 1 つの完全なメッセージを形成します。アセンブルされたメッセージ・ボディは、ユーザー・アプリケーション・プログラムで WEB RECEIVE コマンドを使用して受信することができます。
  - 単一のチャンク化済みメッセージとして CICS が受け入れるデータ量の合計は、要求が着信するポートに関連する TCPIPSERVICE リソース定義の MAXDATALEN オプションを使用して制限することができます。
  - CICS が HTTP サーバーである場合は、チャンク化済みメッセージを受信するためのタイムアウト値を、TCPIPSERVICE 定義の SOCKETCLOSE 属性で設定します。
  - チャンク化済みメッセージの末尾ヘッダーは、HTTP ヘッダー・コマンドを使用して読み取ることができます。 Trailer ヘッダーは、末尾ヘッダーとして含まれていたヘッダーの名前を識別します。要求の処理パスでアナライザー・プログラムを使用している場合は、末尾ヘッダーはメインの要求ヘッダーと一緒にはアナライザー・プログラムに渡されないことに注意してください。
- HTTP クライアントとしての CICS がアプリケーション・プログラムの要求に対する応答としてチャンク化済みメッセージを受信すると、それらのチャンクがアプリケーション・プログラムにエンティティー・ボディとして渡される前に、チャンクもアセンブルされるので、HTTP ヘッダー・コマンドを使用して末尾ヘッダーを読み取ることができます。アプリケーションが応答をどれくらい待って受信するのかを、アプリケーション・プログラムに関係するトランザクション ID のトランザクション・プロファイル定義の RTIMOUT 属性を使用して指定することができます。
- CICS がチャンク化済みメッセージを HTTP サーバーとして (応答) または HTTP クライアントとして (要求) 送信する場合、アプリケーション・プログラムは、メッセージのチャンクごとに WEB SEND コマンドの CHUNKING (CHUNKYES) オプションを使用して、チャンク転送コーディング方式を指定することができます。メッセージは、アプリケーション・プログラムにとって最も都合のよい、任意の方法で分割することができます。 CICS は、チャンク転送コーディング方式が使用されていることを受信側に示すために、適切な HTTP ヘッダーを追加してメッセージの各チャンクを送信します。アプリケーション・プログ

ラムは、CHUNKING(CHUNKEND) を指定した WEB SEND を発行して、メッセージの終了を示します。次に CICS は、必要とされている末尾ヘッダーとともに、(ブランク行を含む) 空のチャンクを送信してチャンク化済みメッセージを終了します。

102 ページの『チャンク転送コーディングによる HTTP 要求または応答の送信』では、CICS から HTTP メッセージを送信するときに、チャンク転送コーディング方式に使用されるプロセスについて説明しています。チャンク化済みメッセージが受信側に受け入れられるようにするためには、この手順に従う必要があります。

CICS Web サポートによるパイプライン化の処理方法

パイプライン化された要求シーケンスを CICS によって送受信することができます。HTTP サーバーとしての CICS は、Web クライアントからのパイプライン化された要求シーケンスを受信することができ、HTTP クライアントとしての CICS は、パイプライン化された要求シーケンスをサーバーに送信することができます。

CICS Web サポートは、パイプライン化された要求シーケンス、およびそれらの要求シーケンスに対する応答を以下のように処理します。

- HTTP サーバーとしての CICS が、パイプライン化された HTTP 要求のシーケンスを受信すると、それらの要求は順次処理されます。このような処理が行われるのは、要求の送信順序と同じ順序で応答が返されるようにするためです。CICS は、パイプライン化されたシーケンスの各メッセージを、独立したトランザクションとして処理し、URIMAP 定義で指定されている静的応答を提供するか、またはアプリケーション・プログラムにメッセージを渡して、アプリケーション・プログラムが応答を生成するまで待機します。各トランザクションは単一の要求を処理し、応答を提供します。パイプライン化されたメッセージ・シーケンスの残りの要求は、前の要求に対する応答が送信され、新規のトランザクションが各追加要求の処理を開始するまで CICS によって保持されます。
- HTTP クライアントとしての CICS がパイプライン化された要求シーケンスを送信すると、自動的にパイプライン化が有効になります。各 HTTP 要求は即時に送信されるので、アプリケーション・プログラムは応答を受信する前に複数の HTTP 要求を送信することができます。パイプライン化されたシーケンスの最後のメッセージが送信されると、アプリケーションは応答の受信を開始することができます。
- HTTP クライアントとしての CICS が、パイプライン化された要求シーケンスに対する HTTP 応答を受信すると、CICS がサーバーから応答を受信した順にアプリケーション・プログラムに応答が返されます。パイプライン化をサポートしているサーバーは、要求が受信されたのと同じシーケンスで応答を提供します。アプリケーション・プログラムは、その HTTP 要求をすべて送信し終えると、応答の受信を開始します。

HTTP クライアントとしての CICS の場合、パイプライン化された要求のシーケンスがべき等であることを保証するのは、アプリケーション・プログラムの担当範囲となります。 21 ページの『パイプライン化』では、べき等について説明しています。要求のシーケンスがべき等であるかどうかが不明な場合サーバーのためだけでなく、アプリケーション・プログラムのロジックのためにも、個別の要求を行い、次の要求を送信する前に各要求に対する応答を待ってみることをお勧めします。

関連タスク

153 ページの『サンプル・プログラム: HTTP サーバーへの要求のパイプライン化』
サンプル・プログラム DFH$WBPA (アセンブリ言語)、DFH$WBPC (C)、および DFH0WBPO (COBOL) は、CICS で HTTP サーバーへのクライアント要求をどのようにしてパイプライン化できるかを示します。

CICS Web サポートによる持続接続の処理方法

Web クライアントと HTTP サーバーとしての CICS 間、または HTTP クライアントとしての CICS とサーバー間で接続が行われると、デフォルトでは、CICS はその接続を持続接続として開いたままにしておこうとします。

CICS が HTTP サーバーの場合、持続接続は以下の状態のときに閉じられます。

- Web クライアントからの要求を処理するユーザー作成アプリケーションが、クライアントに対して接続を閉じることを要求する (WEB SEND コマンドの CLOSESTATUS(CLOSE) オプションを指定)。
- Web クライアントが CICS に対して接続を閉じることを要求する (Connection: close ヘッダーによる通知)。
- Web クライアントが、Connection: Keep-Alive ヘッダーを送信しない HTTP/1.0 クライアントである。
- タイムアウト期間に達した (これは、接続が失敗したこと、または Web クライアントが接続を意図的に終了したことを示しています)。
- CICS 領域は持続接続の最大数として指定された制限に達したので、各応答を受信した後にそれぞれの接続を閉じることを Web クライアントに対して要求している。接続スロットリングについて詳しくは、47 ページの『HTTP サーバーとしての CICS: インバウンド HTTP 接続の接続スロットリング』を参照してください。

上記以外の場合、CICS は、Web クライアントが更に要求を送信できるよう持続接続を開いたままにしておきます。クライアントとの持続接続がある場合、CICS は、Web エラー・プログラムを介してエラー応答が送信された後も接続を開いたままにしておきます。ただし、CICS が 501 (Method Not Implemented (メソッドのインプリメントなし)) 状況コードを応答に対して選択した場合は例外で、この場合には、接続が CICS によって閉じられます。

CICS Web サポートの TCPIPSERVICE リソース定義では、TCPIPSERVICE 定義の SOCKETCLOSE および MAXPERSIST 属性にゼロを指定すべきではありません。SOCKETCLOSE にゼロを設定すると、HTTP サーバーとしての CICS が、追加データが待機していない限り、Web クライアントからデータを受信した直後に接続を閉じることを意味します。 MAXPERSIST にゼロを設定すると、CICS はすべての Web クライアントに対して、CICS からの各応答を受信した後に接続を閉じるよう要求することになります。これらの状態のいずれにおいても、持続接続を維持することはできません。これらの属性にゼロを設定するのは、現在外部要求を処理していない CICS 領域 (例えばテスト環境) でそれを特に必要とする場合だけにしてください。

CICS が HTTP クライアントの場合、持続接続は以下の状態のときに閉じられます。

- サーバーが CICS に対して接続を閉じることを要求する (HTTP/1.1 サーバーが Connection: close ヘッダーを送信すること、または HTTP/1.0 サーバーが Connection: Keep-Alive ヘッダーの送信に失敗したことによる通知)。
- ユーザー・アプリケーション・プログラムがサーバーに対して接続を閉じることを要求する (**WEB SEND** または **WEB CONVERSE** コマンドの CLOSESTATUS(CLOSE) オプションを指定)。

ユーザー・アプリケーション・プログラムが、サーバーが接続の終了を要求したかどうかをテストする必要がある場合は、**READ HTTPHEADER** コマンドを使用して、サーバーからの最後のメッセージで Connection: close ヘッダーを探すことができます。サーバーが接続のクローズを要求しているのに、アプリケーション・プログラムがまだ **WEB CLOSE** コマンドを発行していない場合は、CICS が接続を閉じますが、接続に関係しているデータ (サーバーおよびその HTTP ヘッダーから受信した最後の応答を含む) は維持します。アプリケーション・プログラムは、WEB CLOSE コマンドを発行するか、またはタスクの最後に達するまで、そのデータを使用し続けることができます。

ユーザー・アプリケーション・プログラムが WEB CLOSE コマンドを発行して接続の使用を終了しても、接続がまだ開いていれば、CICS は必ずしもその接続を閉じるとは限りません。接続の接続プーリングが指定されていなかった場合、または接続プーリングにとって接続が良好な状態でない場合、CICS は接続を閉じます。しかし、接続を開くのに使用された URIMAP リソースで接続プーリングが指定されていて、接続が良好な状態であれば、CICS は接続を閉じません。代わりに、CICS は接続を休止接続のプールに入れます。この接続は、別のアプリケーション・プログラム、または同じアプリケーション・プログラムの別のインスタンスが再使用して、同じサーバーに接続することができます。接続プーリングについて詳しくは、48 ページの『HTTP クライアントとしての CICS: アウトバウンド HTTP 接続の接続プーリング』を参照してください。

HTTP クライアントとしての CICS が HTTP/1.0 サーバーと通信している場合、CICS は、HTTP メッセージで Connection: Keep-Alive ヘッダーを自動的に送信します。接続を要求したアプリケーション・プログラムがこれらのヘッダーを送信する必要はありません。Keep-Alive は、持続接続が必要であることをサーバーに通知します。

HTTP サーバーとしての CICS: インバウンド HTTP 接続の接続スロットリング

HTTP サーバーとしての CICS への長期持続接続が複数の Web クライアントによってセットアップされ、使用されるそれらの接続の負荷が重い場合は、接続を扱う CICS 領域が過負荷になり、パフォーマンス上の問題が起こる可能性があります。この問題が発生した場合は、接続スロットリングをセットアップすることにより、過剰な Web クライアントを、ポートを共用して同じサービスを提供する他の CICS 領域に接続できます。

接続スロットリングを使用して、特定のポートに対して CICS 領域が受け入れる持続 HTTP 接続の数の制限を設定できます。制限に達してもさらに Web クライアントが要求を送信する場合、CICS は各応答とともに Connection: close ヘッダーを送信して、新たなクライアントに対してその接続を閉じることを要求します。 CICS 領域への持続接続を既に持っている Web クライアントは、その持続接続を維持で

きます。新たなクライアントが再接続するとき、ポートを共用していて制限に達していない別の CICS 領域に接続した場合に、そこで持続接続を維持できます。制限に達した CICS 領域は、自分に向けて持続接続を行っている Web クライアントがその接続を閉じると、新しい持続接続の受け入れを再開します。

接続スロットリングは、ポートの TCPIPSERVICE リソース定義の MAXPERSIST 属性で管理されます。デフォルト設定の MAXPERSIST(NO) は、CICS 領域が受け入れる持続接続の数に制限がないことを意味します。接続スロットリングをセットアップするには、CICS 領域が同時に処理できる持続接続の数に基づいて、MAXPERSIST 属性に適切な値を指定します。この設定は HTTP および HTTPS プロトコルにのみ適用され、他のプロトコルには適用されません。

HTTP/1.1 サーバーは通常、持続接続を許可する必要があります。したがって、長期持続接続が原因でパフォーマンス上の問題が起こった CICS 領域にのみ接続スロットリングをセットアップしてください。 MAXPERSIST にゼロを設定することは、CICS 領域は持続接続を許可しないことを意味し、HTTP/1.1 仕様に準拠していません。ゼロを設定するのは、現在外部要求を処理していない CICS 領域 (例えばテスト環境) でそれを特に必要とする場合だけにしてください。 MAXPERSIST にゼロより大きい値を指定すれば、HTTP サーバーとしての CICS は依然として HTTP/1.1 仕様に準拠しています。つまり、デフォルト動作でやはり持続接続が許可され、Web クライアントは持続接続を取得できない場合に Connection: close ヘッダーを受け取ります。ただし、持続接続を拒否することは、HTTP/1.1 サーバーの通常のプラクティスとしては推奨されないことを知っておく必要があります。また、Web クライアントが目的の持続接続の取得に失敗すると、Web クライアント自身のパフォーマンスが影響を受けることがあります。

HTTP クライアントとしての CICS: アウトバウンド HTTP 接続の接続プーリング

デフォルトでは、CICS がクライアント HTTP 接続を閉じるのは、CICS アプリケーションが接続を使い終わった後、またはサービス要求元アプリケーションが Web サービス要求を行って応答を受け取った後、あるいは HTTP EP アダプターがビジネス・イベントを出力した後です。接続プーリングをセットアップすると、CICS は接続を閉じる代わりに、その接続を休止状態でプールに入れることができます。 休止接続は、同じアプリケーションが再使用することも、同じホストとポートに接続する別のアプリケーションが再使用することも可能です。接続プーリングによるパフォーマンス上の効果が得られるのは、CICS Web サポート・アプリケーション、Web サービス・アプリケーション、または HTTP EP アダプターの複数の呼び出しにより特定のホストおよびポートに対する接続要求が行われるときや、Web サービス・アプリケーションが複数の要求および応答を行うときです。

接続プーリングをセットアップするには、クライアント HTTP 接続の URIMAP リソース定義の SOCKETCLOSE 属性を指定します。クライアント HTTP 接続がプールされるようにするには、CICS アプリケーション・プログラムが INVOKE SERVICE または WEB OPEN コマンドで URIMAP リソースを指定し、CICS Web サポート・アプリケーションが WEB CLOSE コマンドを発行して自身の接続使用を明示的に終了する必要があります。サーバーが CICS に対して接続を閉じることを要求した場合、またはアプリケーション・プログラムが **WEB SEND** または **WEB CONVERSE** コマンドの CLOSESTATUS(CLOSE) オプションを指定して、サーバーに

対して接続を閉じることを要求した場合は、接続をプールできません。また、CICS は、開いている接続をプールに入れる前に、その状態を検査します。不完全な状態 (例えば、最後の HTTP 応答が OK でなかった) であることが分かったまたは疑われる接続は、プールされません。

アプリケーションが URIMAP リソースを使用してクライアント HTTP 接続を行うと、CICS はそのホストとポートに使用できる休止接続がプールにあるかどうかを検査し、ある場合は新しい接続を開くのではなく、その休止接続をアプリケーションに提供します。アプリケーションは新しい接続を使用するのとまったく同じようにプール接続を再使用するので、接続は使用後に再びプールすることができます。接続が再使用されず、SOCKETCLOSE 属性で指定した制限時間に達すると、CICS はその接続を廃棄します。また、CICS 領域の MAXSOCKETS に達した場合、または接続の URIMAP リソースをユーザーが廃棄した場合、あるいはサーバーが CICS に対して接続を閉じることを要求した場合、CICS はプール内の休止接続を閉じます。

関連概念

HTTP クライアントのパフォーマンスのための接続プーリング

関連情報

HTTP 要求の拒否

CICS Web サポートのコード・ページ変換

CICS が、Web クライアントまたはサーバーとメッセージ交換をする場合、メッセージ内の文字データは、通常、CICS 環境での開始および終了時にコード・ページ変換を実行する必要があります。

次の 2 つの理由から、メッセージ内のテキストのコード・ページ変換が必要とされます。

- CICS および CICS のユーザー作成アプリケーションは、通常、EBCDIC エンコード方式を使用していますが、Web クライアントおよびサーバーは ASCII エンコード方式を使用しています。
- それぞれのエンコード内では、多くの異なるコード・ページを使用して各国語がサポートされています。

メッセージのテキスト以外のコンテンツ (イメージやアプリケーション・データなど) は、コード・ページ変換は必要ありません。

CICS Transaction Server for z/OS バージョン 3 リリース 1 の前の CICS のリリースでは、CICS Web サポートのコード・ページ変換は、コード・ページ変換テーブル (DFHCNV) を使用して処理されていました。CICS Transaction Server for z/OS, バージョン 4 リリース 2 では、アップグレードを目的とした限られた状況を除いて、コード・ページ変換テーブルは、CICS Web サポートでは必要ありません。CICS Web サポートは、z/OS 変換サービスを使用してコード・ページ変換を処理します。

CICS Web サポートでは、テキストのコード・ページ変換のデフォルト設定は以下のとおりです。

- デフォルトの文字セットは ASCII Latin-1 文字セット、ISO-8859-1 です。 HTTP メッセージでは、要求または状況表示行および HTTP ヘッダーは通常、US-ASCII 文字セット (ISO-8859-1 の古いサブセット) 形式です。テキストを含むメッセージ・ボディは、通常、ISO-8859-1 形式です。
- CICS 環境のデータのデフォルト EBCDIC コード・ページは、CICS 領域の **LOCALCCSID** システム初期設定パラメーターで指定されます。**LOCALCCSID** のデフォルトは、EBCDIC Latin 文字セット、コード・ページ 037 です。

より適切な代替コード・ページが指定される場合もあります。

- Web クライアントまたはサーバーが、要求または応答の Content-Type ヘッダーに文字セットを指定する場合があります。この文字セットは、メッセージ・ボディで使用された文字セットです。
- Web クライアントは、要求時に Accept-Charset ヘッダーを送信し、どの文字セットが応答に許容できるかを示します。
- HTTP 以外の要求および一部の古い HTTP インプリメンテーションの場合、メッセージ送信時に使用される文字セットは、メッセージ・ヘッダー内で指定されない場合があり、ご自身のメッセージ・ソースの知識からこれを指定する必要があります。
- アプリケーション・プログラマーは、アプリケーションがメッセージ・データを受信できる適切なコード・ページを指定する必要があります (デフォルトが適していない場合)。

CICS は IANA で指定された文字セットをすべてサポートするわけではありません。CICS によってサポートされるコード・ページ変換の IANA 文字セットは、413 ページの『付録 A. HTML コード化文字セット』にリストされています。

ほとんどの環境においてメッセージのメディア・タイプにより、コード・ページ変換が実行されるかどうかが決定されます。テキスト以外のメディア・タイプの要求または応答ボディは、通常、コード・ページ変換を行いません。旧リリースでコード化された Web 認識のアプリケーションとの互換性で例外が発生します。コマンドで使用されたオプションによって、アプリケーションがCICS Transaction Server for z/OS バージョン 3 リリース 1 以前にコード化されていることが示される場合は、メディア・タイプによるコード・ページ変換に影響はありません。

メッセージのタイプおよび処理パスによっては、CICS によってコード・ページ変換情報が自動的に判別されるか、URIMAP 定義で指定されるか、アナライザー・プログラムにより指定されるか、Web 認識アプリケーション・プログラムによって発行されるコマンドで指定されるかが決まります。『HTTP サーバーとしての CICS に対するコード・ページ変換』では、HTTP サーバーとしての CICS の処理、52 ページの『HTTP クライアントとしての CICS のためのコード・ページ変換』では、HTTP クライアントとしての CICS の処理について説明しています。

HTTP サーバーとしての CICS に対するコード・ページ変換

HTTP サーバーとしての CICS が Web クライアントとメッセージを交換する際、通常は、メッセージ・ボディのコード・ページ変換が必要になります。これを指定する方法は、アプリケーション生成の応答を行っているのか、静的応答を行っているのか、Web 対応アプリケーションまたは Web 非対応アプリケーションのいずれを使用しているのか、によって異なります。

要求行および HTTP ヘッダー

要求行または状況表示行のコード・ページ変換および HTTP ヘッダーのコード・ページ変換は、以下のように処理されます。

- CICS は、要求を受信するとすぐに (照会ストリングを含む) 要求行および HTTP ヘッダーを、それぞれの文字セットから、ローカルの CICS 領域全体に適用され、デフォルトの 037 を持つ、 LOCALCCSID システム初期設定パラメーターで指定されている EBCDIC コード・ページに変換します。変換を正常に行うには、LOCALCCSID システム初期設定パラメーターを、ASCII Latin-1 文字セット ISO-8859-1 (コード・ページ 819) の変換先にすることができる EBCDIC コード・ページに設定する必要があります。 LOCALCCSID が不適切なコード・ページに設定されている場合、CICS はその代わりにデフォルトの EBCDIC コード・ページ 037 を使用します。
- アプリケーションが WEB EXTRACT、WEB READ HTTPHEADER、または WEB READ FORMFIELD コマンドを使用して (照会ストリングを含む) 要求行および HTTP ヘッダーから情報を抽出する際、その情報は、その変換済みの形式である、LOCALCCSID システム初期設定パラメーターで指定されている EBCDIC コード・ページ (またはデフォルトの 037) で提供されます。
- CICS が応答の送信準備をしているときに、状況表示行および HTTP ヘッダーは、CICS で生成することも、または WEB WRITE HTTPHEADER コマンドを使用してアプリケーションで指定することもできます。送信の前に、すべてのヘッダーおよび状況表示行が、指定された EBCDIC コード・ページから US-ASCII 文字セットに変換されます。

メッセージ・ボディ: アプリケーション生成の応答

要求が、ユーザー作成アプリケーションからの動的応答を受け取る場合、以下のように、メッセージ・ボディのコード・ページ変換が処理されます。

- Web 対応アプリケーションが要求を受信した場合、RECEIVE コマンドのいずれかのコード・ページ変換オプションを使用して変換が指定されていれば、CICS はコード・ページ変換を実行します。いずれのオプションも指定されていない場合は、コード・ページ変換は行われません。デフォルトが適切でない場合は、文字セットを提供するか、または CICS が文字セットを識別できるようにして、コード・ページを要求することができます。
- 要求の処理パスでアナライザー・プログラムが使用される場合、後続の処理ステージに対してストレージ・ブロックで渡される要求のコピーについて、そのアナライザー・プログラムでコード・ページ変換を指定するか、またはコード・ページ変換を抑止することができます。使用される文字セットおよびアプリケーション・コード・ページの両方を提供してください。 CICS は、まだ要求ボディの元のバージョンを保持しています。**EXEC CICS** WEB API コマンドを使用するアプリケーションまたはコンバーター・プログラムは、ストレージ・ブロックではなく元のボディにアクセスし、**EXEC CICS** WEB API コマンドでコード・ページ変換を指定することができます。
- コンバーター・プログラムがストレージ・ブロックで要求を引き渡すときに、処理パスにアナライザー・プログラムがない場合、 CICS はその文字セットを識別し、デフォルトのコード・ページに変換して、ストレージ・ブロック内の要求ボディを変換します。

- Web クライアントが要求ボディに使用した文字セットを識別するために、CICS は要求ヘッダーを調べます。要求ヘッダーにこの情報が含まれていない場合や、指定されている文字セットがサポートされていない場合、CICS は、デフォルトとして、メッセージ・ボディが ISO-8859-1 文字セットであると想定します。メッセージ・ボディが想定された文字セットではなく、しかもヘッダーに情報が含まれていない場合は、ユーザーが正しい文字セットを識別する必要があります。
- デフォルトでは、CICS は、LOCALCCSID システム初期設定パラメーターに指定されている EBCDIC コード・ページに、要求ボディを変換します。この指定は、ローカルの CICS 領域全体に適用されるもので、デフォルトとして 037 を持っています。アプリケーションで、別のコード・ページ (EBCDIC または ASCII の可能性があります) が必要な場合は、そのコード・ページを指定することができます。
- アプリケーションまたはコンバーター・プログラムが、**EXEC CICS** WEB API コマンドを使用して応答を送信した場合、いずれかのコード・ページ変換オプションを使用して WEB SEND コマンドで変換が指定されていれば、CICS はコード・ページ変換を実行します。いずれのオプションも指定されていない場合は、コード・ページ変換は行われません。
- コンバーター・プログラムがストレージ・ブロックに応答を作成し、それを送信するために CICS に渡すと、CICS は、その要求に対して実行されたコード・ページ変換をそのまま反映します。このとき使用されるのは、アナライザー・プログラムの文字セットおよびホスト・コード・ページの設定、またはアナライザー・プログラムがない場合のデフォルトの設定です。アナライザー・プログラムが要求に対するコード・ページ変換を抑止した場合、応答ボディのコード・ページ変換は実行されません。

メッセージ・ボディ: 静的応答

要求が、URIMAP 定義で決定された静的応答を受け取る場合は、以下のようにメッセージ・ボディのコード・ページ変換が処理されます。

- 静的応答の場合、CICS は、Web クライアントの要求に含まれているメッセージ・ボディは調べないので、コード・ページ変換は要求されません。
- 応答のボディに対するコード・ページ変換は、静的応答を生成する URIMAP 定義で指定します。応答にテキストが含まれる場合は、以下のすべてを URIMAP 定義で指定する必要があります。
  - テキスト・メディア・タイプ。MEDIATYPE 属性を使用します。この属性にはデフォルトはありません。
  - Web クライアントの文字セット。CHARACTERSET 属性を使用します。
  - HOSTCODEPAGE 属性を使用する、応答用の CICS 文書テンプレート、または z/OS UNIX ファイルがエンコードされているコード・ページ。

CICS は、z/OS UNIX ファイルを取得するか、または CICS 文書テンプレートを取得して文書を作成し、適切なコード・ページ変換を実行します。

HTTP クライアントとしての CICS のためのコード・ページ変換

HTTP クライアントとしての CICS がサーバーとメッセージを交換する際、通常、メッセージ・ボディのコード・ページ変換が必要です。接続を開くときにアプリケ

ーションのコード・ページを指定してください。それらの文字セットは通常、CICS が識別とができますが、デフォルトに許可することもできます。

要求行および HTTP ヘッダー

要求行または状況表示行のコード・ページ変換および HTTP ヘッダーのコード・ページ変換は、以下のように処理されます。

- CICS が要求の送信準備をしているとき、要求行および HTTP ヘッダーを CICS が生成することもできますし、WEB WRITE HTTPHEADER コマンドを使用してアプリケーションによって指定することもできます。送信する前に、すべてのヘッダーが、指定されていた EBCDIC コード・ページから US-ASCII 文字セットに変換されます。
- 応答を受け取るとすぐに CICS は、状況表示行および HTTP ヘッダーを US-ASCII 文字セットから EBCDIC コード・ページ 037 に変換します。アプリケーションは状況表示行およびその他の情報を受け取り、EBCDIC コード・ページ 037 に変換された形式で HTTP ヘッダーを調べます。

メッセージ・ボディ

メッセージ・ボディのコード・ページ変換は、以下のように処理されます。

- アプリケーション・プログラムで使用される EBCDIC コード・ページは、サーバーとの通信を開始する WEB OPEN コマンドで指定されます。デフォルトは、LOCALCCSID システム初期設定パラメーターで指定されている EBCDIC コード・ページです。これは、ローカルの CICS 領域全体に適用され、デフォルトとして 037 を持っています。 CICS は、この接続における要求および応答のメッセージ・ボディを変換するためにこの情報を使用します。
- アプリケーションが送信する要求ごとに、WEB SEND または WEB CONVERSE コマンドの CLIENTCONV オプションによって、CICS が要求ボディに対してコード・ページ変換を実行するかどうかを指定します。デフォルトでは、コード・ページ変換が実行されます。WEB CONVERSE コマンドを使用する場合は、要求ボディおよび応答ボディのいずれかにコード・ページ変換を指定することも、両方にコード・ページ変換を指定することもできますし、あるいはどちらのボディにも指定しないよう選択することもできます。
- 要求の変換を指定した場合、デフォルトは、CICS が要求ボディを ISO-8859-1 文字セットへ変換することです。サーバーで別の文字セットを要求していることが分かっている場合は、WEB SEND または WEB CONVERSE コマンドで CHARACTERSET オプションを使用して、代替の文字セットを選択することができます。
- アプリケーションが受信する応答ごとに、WEB RECEIVE または WEB CONVERSE コマンドの CLIENTCONV オプションで、CICS が応答ボディのコード・ページ変換を実行して、接続が開かれたときに指定した EBCDIC コード・ページへ変換するかどうかを指定します。デフォルトでは、コード・ページ変換が実行されます。 CICS は、応答ヘッダーを調べて、サーバーが応答ボディに使用した文字セットを識別します。応答ヘッダーにこの情報が含まれていない場合や、名前が指定された文字セットがサポートされていない場合には、CICS はデフォルトとして、メッセージ・ボディが ISO-8859-1 文字セットであるものと想定します。

HTTP サーバーとしての CICS の HTTP/1.1 準拠

CICS Web サポートは、条件付きで HTTP/1.1 仕様に準拠しています。

HTTP/1.1 仕様は、Internet Society および IETF (Internet Engineering Task Force) の Request for Comments 文書である RFC 2616「Hypertext Transfer Protocol - HTTP/1.1」に記述されています。

HTTP/1.1 仕様に条件付きで準拠しているとは、CICS はすべての "MUST" レベルの要件を満たしているが、HTTP/1.1 仕様で詳細化されている "SHOULD" レベルのすべての要件は満たしていないという意味です。これらの要件は、CICS 自体が提供している機能に関係しています。そのプロトコルに対するすべての MUST または REQUIRED レベルおよびすべての SHOULD レベルの要件を満たす実装は、無条件で準拠しているといわれます。実装が、それが実装しているプロトコルに対する 1 つ以上の MUST または REQUIRED レベルの要件を満たしていない場合、その実装は準拠していません。

CICS Web サポートによる HTTP/1.1 仕様への準拠には 3 つの側面があります。

- CICS Web サポートは、HTTP サーバーから要求されているアクションを実行します。例えば、CICS Web サポートはインバウンド要求を受信し、持続接続を維持し、いくつかの HTTP ヘッダーを書き込み、応答を転送します。これらのアクション中の CICS Web サポートの動作は、条件付きで HTTP/1.1 仕様に準拠しています。必要に応じて、これは CICS の旧リリースからの動作の変更を表します。 55 ページの『HTTP/1.1 に準拠する CICS Web サポートの動作』では、CICS Web サポートの動作がどのように HTTP/1.1 に準拠しているか、そしてそれがどういう点で CICS の CICS Transaction Server for z/OS バージョン 3 リリース 1 より前のリリースと異なっているかが説明されています。
- HTTP/1.1 仕様の一部は CICS Web サポートには関係していません。例えば、CICS Web サポートは、プロキシー・サーバーとして動作することはなく、キャッシング機能を提供することもありません。それらの領域については 57 ページの『CICS Web サポートでサポートされていない HTTP 機能』で説明されており、それを見ると、CICS Web サポートやユーザー・アプリケーション・プログラムにとって HTTP/1.1 準拠が問題にならない領域について知ることができます。
- CICS において Web 対応ユーザー・アプリケーション・プログラムを使用して、アプリケーションが生成する HTTP 応答を作成し、応答を提供する方法を CICS Web サポートに指示することができます。 CICS Web サポートの実装を HTTP プロトコル仕様、特に HTTP/1.1 に準拠させる場合は、ユーザー・アプリケーション・プログラムが、それらが実行するアクションでの準拠に対して責任を共用します。この文書ではいくつかの基本的なガイダンス情報が提供されていますが、詳細についてはご使用の HTTP の仕様を確認することが重要です。 87 ページの『第 6 章 HTTP サーバーとしての CICS のための Web 対応アプリケーション・プログラムの作成』では、CICS Web サポート対応のユーザー・アプリケーション・プログラムを作成するためのプロセスが説明されています。

  実際面では、HTTP/1.1 仕様のさまざまなレベルの要件 (MUST、SHOULD、または MAY) は、以下のようにアプリケーション・プログラムで処理する必要があります。

  – 準拠を維持するためには、MUST レベルの要件を実装する必要があります。CICS Web サポートは、単純なアクティビティーに適用されるすべての MUST

レベルの要件をユーザーが処理する、またはそれらの要件に準拠するのを支援するように設計されています。CICS Web サポートがまだ扱っていないオプションの要件を満たす場合には、いくつかの追加の MUST レベルの要件が適用されることがあります。また、いくつかの MUST レベルの要件は、特定の環境下でアプリケーションが実行してはいけない アクションに関係しています。

- SHOULD レベルの要件は、HTTP 仕様に対する条件付き準拠には不要です。CICS は、HTTP/1.1 仕様のすべての SHOULD レベルの要件に準拠しているわけではありません。ただし、アプリケーションが不都合なく SHOULD レベルの要件に準拠できる場合は、準拠させることをお勧めします。
- MAY アクションは、その準拠レベルに関係なく、どの HTTP 実装に対してもオプションです。これらのアクションは、提案または推奨として扱う必要があります。

HTTP/1.1 に準拠する CICS Web サポートの動作

CICS Web サポートは、ユーザーに代わって、HTTP/1.1 仕様の数多くの要件に準拠します。

ここで説明しているほとんどの動作は、HTTP サーバーとしての CICS に対する HTTP 要求を処理するために、URIMAP 定義またはアナライザー・プログラムのどちらを使用しているのかには関係なく適用できますが、いくつかの例外があります。

- **CICS は、インバウンド・メッセージが HTTP/1.1 に準拠しているかどうかを検査し、準拠していないメッセージを処理または拒否します。** 要求を受信すると、URIMAP 定義またはアナライザー・プログラムが関係する前に、即時に検査が行われます。さまざまな基本的な受け入れ検査が行われ、しかもメッセージが受け入れ不可で、 CICS が問題を自身で処理することが適切でない場合、可能であれば Web クライアントにエラー応答が返されます。これらの基本的な受け入れ検査は HTTP/1.0 メッセージに対しては実行されず、(HTTP プロトコルの代わりの) USER プロトコルが TCPIPSERVICE 定義で指定されている場合でも実行されません。

  注: CICS では、メッセージ・ボディを持つすべてのインバウンド HTTP/1.1 メッセージに、Content-Length ヘッダーが必要です。メッセージ・ボディが存在するが、ヘッダーが提供されていない場合、またはその値が不正確である場合、欠陥のあるメッセージ、または以降のメッセージに対して受信されるソケットによって、予期しない結果が生じることがあります。メッセージ・ボディを持つ HTTP/1.0 メッセージの場合、Content-Length ヘッダーはオプションです。
- **CICS は、URL の比較のために HTTP/1.1 の規則に従います。** スキーム名およびホスト名は大/小文字を区別せずに比較されますが、パスは大/小文字を区別して比較されます。比較の前にすべての構成要素がアンエスケープされます。CICS は、要求内の絶対 URI 形式も処理します (この場合、ホスト名は要求行で指定されます)。URIMAP 定義の代わりにアナライザー・プログラムを使用してインバウンド HTTP 要求を処理するときに、この点において準拠を実現する必要がある場合は、HTTP/1.1 仕様で規定されている規則に従って URL 比較を実行するようにアナライザー・プログラムをコーディングする必要があります (HTTP 仕様についての詳細は、 13 ページの『HTTP プロトコル』を参照してください)。

- **CICS は、アウトバウンド・メッセージの開始行で適切な HTTP バージョン番号を提供します。** CICS は、Web クライアントまたはサーバーが HTTP/1.0 レベルであると識別していない限り、通常はメッセージを HTTP/1.1 として識別します。その場合、CICS はメッセージを HTTP/1.0 として識別します。 Web クライアントによる、あるバージョンの HTTP から別のバージョンへのアップグレード、または別のセキュリティー・プロトコルへのアップグレード要求は、CICS ではサポートされていません。
- **CICS は、メッセージが HTTP/1.1 に準拠するためには通常存在していなければならない HTTP ヘッダーを、アウトバウンド HTTP/1.1 メッセージで提供します。** 準拠上必須ではないもののアプリケーション・プログラムが要求したアクションをサポートするためのヘッダーも、CICS によっていくつか作成されます (例えば Expect ヘッダー)。 415 ページの『付録 B. CICS Web サポートにおける HTTP ヘッダーの解説』で、CICS が書き込むヘッダーとそれらのヘッダーが作成される環境について説明しています。準拠のためのヘッダーは、Web 対応アプリケーションおよび Web 非対応アプリケーションの両方が送信するメッセージに提供されます。これらのヘッダーは、(HTTP プロトコルではなく) USER プロトコルが TCPIPSERVICE 定義に対して指定されている場合は提供されません。HTTP/1.0 メッセージの場合は、Connection: Keep-Alive、Content-Length、Date、および Server ヘッダーのみが提供されます。
- **CICS は、インバウンドおよびアウトバウンドのどちらの要求に対しても、Expect ヘッダーのアクションを実行します。** HTTP サーバーとしての CICS は、Expect ヘッダーを含む要求を受信すると、100-Continue 応答を Web クライアントに送信し、要求の残りを待ちます。HTTP クライアントとしての CICS の場合、WEB SEND コマンドで EXPECT オプションを使用して、CICS に Expect ヘッダーをサーバーに送信させ、要求を送信する前に 100-Continue 応答を待ちます。サーバーが異なる応答を返した場合、CICS はアプリケーション・プログラムに通知して送信を取り消します。
- **CICS は Web クライアントからの OPTIONS 要求を処理し、適切な応答を返します。** OPTIONS * (特定のリソースではなく、サーバー全体についての照会) という形式のみが受け入れられます。この応答には、HTTP サーバーとしての CICS に関する基本情報 (HTTP バージョンおよびサーバー・ソフトウェアの説明) が含まれています。ユーザー・アプリケーション・プログラムは関係しません。
- **CICS は Web クライアントからの TRACE 要求を処理し、適切な応答を返します。** CICSWeb サポートは、TRACE メソッドを含む正しい形式の要求を受信すると、要求とそのオリジナルのヘッダー、およびその要求が取得したヘッダー (例えば Via ヘッダー) を含む応答を返します。ユーザー・アプリケーション・プログラムは関係しません。
- **CICS は、チャンク転送コーディングを含むインバウンド・メッセージを受け取り、それらをアセンブルして、ユーザーがチャンク転送コーディングを使用してアウトバウンド・メッセージを送信するのを支援します。** チャンク化済みメッセージの末尾ヘッダーは、HTTP ヘッダーの読み取り、書き込み、およびブラウズ・コマンドを使用して操作することができます。つまり、アプリケーションは、チャンク化済みメッセージを、標準のメッセージの場合と同じように受信して処理することができます。CICS では、ユーザー・アプリケーションからのチャンク化済みメッセージの送信もサポートしていますが、正しい手順に従って、チャンク化済みメッセージを受信側が受け入れ可能になるようにする必要があり

ます。 44 ページの『CICS Web サポートによる、チャンク転送コーディングの処理方法』では、この点における CICS Web サポートの動作について説明しています。

- **CICS は、インバウンドおよびアウトバウンドの両方のメッセージに対するパイプライン化をサポートしています。** CICS を使用すると、Web クライアントからパイプライン化された要求を受信できます。CICS は各要求を順にアプリケーション・プログラムに渡し、要求の受信順に要求に応答して HTTP/1.1 に準拠していることを保証します。CICS では、パイプライン化された要求をサーバーに送信することもできますが、正しい手順に従って、パイプライン化された要求シーケンスを HTTP/1.1 に準拠させる必要があります。 45 ページの『CICS Web サポートによるパイプライン化の処理方法』では、この点における CICS Web サポートの動作について説明しています。
- **CICS は仮想ホスティング (IP アドレスが同じである複数のホスト名) をサポートしています。** 仮想ホストのサポートは、URIMAP 定義に基づいています。 10 ページの『仮想ホスティング』では、提供されているサポートについて説明しています。
- **接続はデフォルトで永続的です。** これは、HTTP サーバーとしての CICS、および HTTP クライアントとしての CICS の両方に当てはまります。CICS は、HTTP/1.1 レベルの Web クライアントまたはサーバーと通信している場合は、CICS 内の Web クライアント、サーバー、またはユーザー・アプリケーションが明確にクローズまたはタスクの終了を要求していない限り、接続を開いたままにしておきます。CICS は、HTTP/1.0 レベルの Web クライアントまたはサーバーと通信している場合、持続接続がサポートされている場合は Connection: Keep-Alive ヘッダーを送信します。デフォルトでは、HTTP サーバーとしての CICS と通信する Web クライアントはすべて、必要な場合に持続接続を取得できます。ただし、長期持続接続が原因でパフォーマンス上の問題が起こった CICS 領域がある場合は、接続スロットリングを使用して領域への持続接続の数を制限することにより、過負荷を回避できます。 46 ページの『CICS Web サポートによる持続接続の処理方法』で、持続接続に関する CICS Web サポートの動作について説明しています。

CICS Web サポートでサポートされていない HTTP 機能

HTTP/1.1 仕様では HTTP プロトコルを利用する通話者にさまざまな役割が定義されています。CICS Web サポートは、起点サーバー、クライアント、およびユーザー・エージェントにとって適切な数多くの機能を実行します (ただし、どの HTTP クライアント要求にも人間のユーザーが関係しない場合もあります)。HTTP/1.1 仕様には、プロキシー、ゲートウェイ、トンネル、およびキャッシュに関係する要件も含まれていますが、これらの要件は CICS Web サポートには関係がないので、無視してかまいません。

- **CICS はプロキシーとしては動作しません。** プロキシーの動作に関連する、HTTP/1.1 仕様の要件はすべて無視してかまいません。
- **CICS は、ゲートウェイ (別のサーバーのための中継) としても、トンネル (HTTP 接続間の中継) としても動作しません。** ゲートウェイおよびトンネルの動作に関連する、HTTP/1.1 仕様の要件はすべて無視してかまいません。
- **CICS には、キャッシュ機能が用意されておらず、またユーザー作成キャッシュ機能のサポートもありません。** キャッシュの動作に関連する、HTTP/1.1 仕様の

要件はすべて無視してかまいません。 サーバーから受信する情報はどれも保管することはできますが、サーバーから現行情報を受信することを期待して要求を出すユーザーには、保管されている情報を送信しないように注意する必要があります。

- **CICS は Web ブラウザーとして使用するようには設計されていません。** ユーザー・アプリケーション・プログラムは、HTTP クライアントとしての CICS を介して、サーバーから利用できる個々の既知のリソースに対して要求を行うことができますが、一般的にインターネットをブラウズすることは期待されていません。CICS は、履歴リスト、お気に入りのリスト、または Web ブラウザーのその他の機能を備えていないので、これらの機能に関連する要件はすべて無視できます。

HTTP 仕様についての詳細は、 13 ページの『HTTP プロトコル』を参照してください。

Atom 形式および出版プロトコルへの準拠

CICS では、RFC 4287 および RFC 5023 で記述されている Atom 形式および Atom 出版プロトコルがサポートされています。

Atom 配信形式および Atom 出版プロトコルについては、以下のインターネット協会と IETF (Internet Engineering Task Force) の Request for Comments documents (RFC) で説明されています。

RFC 4287 (「*The Atom Syndication Format*」)
Atom フィード文書およびエントリー文書の構造に関する仕様。この仕様は http://www.ietf.org/rfc/rfc4287.txt で参照できます。

RFC 5023 (「*The Atom Publishing Protocol*」)
HTTP を介して Atom 文書形式のリソースを公開および編集するためのプロトコル。Atom サービス文書およびカテゴリー文書の構造に関する説明が含まれています。この仕様は http://www.ietf.org/rfc/rfc5023.txt で参照できます。

Atom 形式およびプロトコルに準拠した CICS の動作

CICS は、Atom 形式およびプロトコルに準拠するために必要な複数のタスクを処理します。

- CICS は、Atom フィード文書、Atom エントリー文書、およびコレクションの Atom 構成ファイルの形式を検証して、 RFC 4287 の要件に準拠しているかどうかをチェックします。また、実行可能な場合は常に、必須の要素が提供されているかどうかをチェックし、必要な場合にはデフォルト値を適用します。Atom フィードをセットアップするための CICS 文書には、CICS がユーザーのデータを検証できない状況に関する情報が記載されています。このような状況では、確実に正しいデータを提供する必要があります。
- CICS は、作成された Atom サービス文書および Atom カテゴリー文書を処理して、クライアントに送信します。ただし、それらの文書が Atom 構成ファイル内にあるか静的文書として送信されたかにかかわらず、文書の形式を検証することはありません。それらの Atom 文書タイプの形式が RFC 5023 に記載されている要件に準拠していることを確認する必要があります。

- CICS の EXEC CICS API コマンドを使用すると、Atom 文書の Date 構造に適した形式でタイム・スタンプを作成できます。EXEC CICS API コマンドを使用せずにタイム・スタンプを作成する場合は、 RFC 4287 に記載されている Date 構造の要件に準拠していることを確認する必要があります。
- CICS は、<atom:id> 要素および <atom:link> 要素で使用される IRI のマッピングと比較のタスクを処理します。
- CICS は、Atom フィード文書に、<atom:generator> 要素を提供します。これは、生成元エージェントの名前を、URI およびバージョン番号とともに指定します。
- CICS は、Atom フィードをサービス提供する際に、RFC 4287 の仕様を満たす形式で各エントリーの Atom ID を生成できます。 269 ページの『Atom エントリーの Atom ID』に記載されている条件下では、Atom ID は固有になり、各 Atom エントリーの存続期間において同じです。
- CICS は、コレクション内の通常の Atom エントリーを操作することを求めるクライアント要求を処理するためにサーバーで必要なアクションを、RFC 5023 に指定されているように実行します。ただし、『CICS でサポートされていない Atom の機能』に記載されている点は異なります。特に、CICS ではコレクション内のメディア・リソースとメディア・リンクのエントリーはサポートされていません。
- CICS を使用すると、CICS リソース・セキュリティーおよび CICS Web サポート・セキュリティーを介して、 Atom フィードのデータが含まれる Atom フィードおよびリソース用のセキュリティーを実装できます。ユーザーは、 Atom フィードを保護するために適切なセキュリティー機能を選択してセットアップする必要があります。RFC 5023 では、認証メカニズムの使用が推奨されるとともに、脅威となるいくつかのシナリオが説明されています。

CICS でサポートされていない Atom の機能

Atom 形式およびプロトコルの一部の要件は、CICS には関係ないか、CICS ではサポートされていません。

- RFC 4287 および RFC 5023 の「Atom プロセッサー (Atom Processors)」に関する要件は、CICS には関係ありません。CICS は処理用の Atom フィード文書を受信しないので、Atom プロセッサーとしては機能しません。
- CICS では Atom 文書のデジタル署名および暗号化はサポートされていませんが、RFC 4287 と準拠するため、署名が含まれる Atom 文書が拒否されることはありません。
- CICS では、テキスト構造 (<atom:title> 要素など) 内の HTML または XHTML コンテンツはサポートされていません。ユーザーは、子要素を指定せずにプレーン・テキストのコンテンツを提供する必要があります。
- CICS では、HTML、XHTML、およびその他のテキスト・メディア・タイプ (XML など) が <atom:content> 要素のコンテンツとして許可されています。ただし、コンテンツはユーザー自身が検証する必要があります。CICS では、マークアップが有効であるかどうかコンテンツが解析されることはなく、コンテンツはユーザーが提供したままの状態でクライアントに提供されます。
- コンテンツが有効な Base64 エンコードで作成されている場合は、<atom:content> 要素を使用して、非テキスト・メディア・タイプのコンテンツを提供することができます。CICS ではこれは禁止されていませんが、非テキストのコンテンツは

サポートされていないため、Base64 でエンコードされているコンテンツで必須である <atom:summary> 要素の有無はチェックされません。

- CICS ではコレクション内のメディア・リソースとメディア・リンクのエントリーはサポートされていません。メディア・リソースは、主にコレクション内で非テキストのコンテンツを整理するための手段として Atom 出版プロトコル (RFC 5023) で指定されています。クライアントがコレクション内でエントリーを作成しようとすると、CICS は Atom エントリー (メディア・タイプは application/atom+xml で、 type=entry パラメーターはある場合とない場合があります) でないクライアント要求を状況コード 415 で拒否します。CICS 用のサービス文書では、<app:accept> 要素で追加のメディア・タイプを指定しないでください。
- CICS における Atom フィードのサポートは、データのフィードをクライアントに提供することを目的としています。CICS では、リモート・コンテンツを参照するために "src" 属性を使用するエントリーなど、データを含まない Atom エントリーの配信はサポートされていません。
- CICS では、エントリーについてオプションの <atom:source> 要素はサポートされていません。<atom:source> 要素は、 Atom エントリーがあるフィードから別のフィードにコピーされる場合にメタデータを保存するために使用されます。
- CICS では、リンク要素として <atom:link rel="self"> および <atom:link rel="edit"> のみがサポートされています。related、enclosure、または via リンクなどのその他のリンク要素、および代替形式へのリンクはサポートされていません。CICS では、リンク要素の title 属性または length 属性はサポートされていません。
- CICS では、カテゴリーに基づいてコレクションの Atom エントリーが拒否されることはありません。サービス文書またはカテゴリー文書で <app:categories> 要素を使用して、コレクション内のエントリーについて受け入れられるカテゴリーを指定することができますが、CICS ではクライアントがこれらのカテゴリーに準拠しているかどうかは監視されません。
- CICS では Slug ヘッダーが無視され、Atom エントリー用に独自の URI が提供されます。
- パフォーマンス上の理由から、CICS では、最後に編集された時刻 (エントリーの <app:edited> 要素で示されます) の順で自動的にコレクション内に Atom エントリーが返されることはありません。RFC 5023 において、この機能はエントリーの完全なリストについては推奨される要件となっていますが、エントリーの部分リストについては必須の要件となっています。 CICS では、許容される応答時間を維持するとともに部分リストの有用な機能を提供し続けるため、この要件から外れています。サービス・ルーチンを使用して CICS にエントリーを提供する場合は、最後に編集された時刻 (リソースにこの情報を格納できる場合) の順でエントリーを提供することにより、コレクションをこの要件に準拠させることができます。
- CICS では、Atom エントリーで <app:control> 要素および <app:draft> 要素はサポートされていないため、無視されます。
- CICS に Atom エントリー・データを提供する一部のリソースでは、RFC 4287 にある推奨内容に従って Atom エントリーとともに Atom ID を格納することができません。ただし CICS には、 269 ページの『Atom エントリーの Atom ID』

に記述されている条件下で、Atom エントリーがサービス提供されるたびにそのエントリーに同一の Atom ID を生成する機能があります。

- RFC 4287 では、Atom エントリーを再利用するか、または別の場所に移動する場合に、Atom ID がその Atom エントリーと共に存続することを求めています。Atom ID を Atom エントリーと共に格納するなら、Atom エントリーを別の場所に移動しても、この要件に準拠できます。 Atom ID を Atom エントリーと共に格納しない場合は、Atom エントリーを別の場所に移動しないでください。

第 2 部 CICS Web サポート

Web コンテキストで CICS を使用するには、システムを計画し、構成する必要があります。その後で、CICS Web サポート用のアプリケーション・プログラムとユーティリティー・プログラムを作成できます。

第 4 章 CICS Web サポートのコンポーネントの構成

CICS Web サポートのこれらのコンポーネントは、すべての CICS Web サポート・タスクに必要です。CICS Web サポートでの作業を開始する前に、これらを構成してください。

このタスクについて

26 ページの『CICS Web サポートのコンポーネント』に、すべてのコンポーネントの完全なリストを記載しています。

以前の CICS リリースでコーディングしたアナライザー・プログラムを使用してコード・ページ変換テーブル DFHCNV を参照する場合は、いくつかの DFHCNV 項目をセットアップしなければならない場合があります。新規の CICS Web サポート開発には、コード・ページ変換テーブルの項目は必要ありません。

これらの基本コンポーネントを、次のリストで示したようにセットアップします。その後、サブトピックを順を追って進んでセットアップを完了し、CICS Web サポートの動作を検証します。

手順

1. 「*CICS Transaction Server for z/OS インストール・ガイド*」 の手順に従って、CICS 領域の TCP/IP サポートを有効にします。 この処理には、Communications Server のセットアップ、および z/OS を介した DNS (ドメイン・ネーム) サーバーへのアクセスの確立が含まれます。
2. 「*CICS Transaction Server for z/OS インストール・ガイド*」のGiving CICS regions access to z/OS UNIX System Servicesの指示に従って、 CICS 領域のユーザー ID のユーザー・プロファイルに OMVS セグメントを組み込み、CICS が z/OS UNIX System Services にアクセスできるようにします。
3. 「*CICS RACF Security Guide*」の指示に従って、SSL サポートをセットアップします。 「*CICS RACF Security Guide*」でも、SSL が提供する機能を説明しています。

CICS Web サポートのシステム初期設定パラメーターの指定

CICS Web サポートを使用可能にするには、これらのシステム初期設定パラメーターを指定します。

手順

1. 必須: **TCPIP=YES** を指定して CICS TCP/IP サービスを活動化します。 デフォルトの設定は NO です。CICS Web サポートを使用可能にするには YES を指定する必要があります。
2. **LOCALCCSID** システム初期設定パラメーターを使用して、ローカルの CICS 領域のコード化文字セット ID を指定します。 CICS は、このコード・ページをアプリケーション・プログラムのデフォルトと見なします。デフォルトは EBCDIC コード・ページ 037 です。代替コード・ページ変換オプションが選択されてい

ない場合、CICS は着信 HTTP 要求のデータ内容をこのコード・ページに変換した後、アプリケーション・プログラムに渡します。 CICS は、アプリケーションはこのコード・ページで HTTP 応答を提供したと見なします。

3. CICS 文書テンプレート・サポートを使用する場合は、HTTP 要求に静的応答を提供するか、アプリケーション生成応答の一部として、 **DOCCODEPAGE** システム初期設定パラメーターを使用して文書ドメインのデフォルトのホスト・コード・ページを指定します。デフォルトは EBCDIC コード・ページ 037 です。
4. 3270 表示アプリケーションに Web クライアントがアクセスできるようにする場合、または CICS ビジネス・ロジック・インターフェースを使用する場合は、**WEBDELAY** システム初期設定パラメーターを使用して適切なタイムアウト期間を指定します。
   - 活動が発生しなかった場合、Web 作業とそれに関連するデータに削除を指定するまでの時間 (分単位)。デフォルト値は 5 分です。
   - ガーベッジ・コレクション・トランザクション CWBG を実行して、指定されている作業およびそのデータを削除する頻度 (分単位)。デフォルトは 60 分です。

   WEBDELAY は、3270 表示アプリケーションまたは CICS ビジネス・ロジック・インターフェースを使用しない CICS Web サポート・タスクには適用されません。
5. CICS Web サポートでセキュリティーを使用する場合は、さらに次のシステム初期設定パラメーターの値も設定します。
   - **CRLPROFILE**
   - **ENCRYPTION**
   - **KEYRING**
   - **MAXSSLTCBS**
   - **SSLCACHE**

   CICS で SSL を処理する方法 (これらのシステム初期設定パラメーターの指定方法なども含む) については、「*CICS RACF Security Guide*」を参照してください。

CICS Web サポート用のポートの予約

CICS Web サポートに必要な数だけ z/OS Communications Server ポートを予約します。可能であれば、これらのポートが CICS Web サポート専用になるようにしてください。

手順

- HTTP の場合のウェルノウン (デフォルト) ポート番号は 80、HTTPS の場合のウェルノウン・ポート番号は 443 です。 ウェルノウン・ポートを使用する可能性のある同じ IP アドレスの他のサーバーとの競合を解決するように気を付けてください。
- アプリケーション・プログラマーは、1024 から 32,767 までのポート番号を標準外サーバーに使用できます。 1024 より小さい番号のポートは、特定機能用に IANA によって割り当てられたウェルノウン・ポート番号です。したがって、

HTTP 用ポート 80 と HTTPS 用ポート 443 を除き、それらのポートを CICS Web サポートに使用しないでください。 SSL 要求と非 SSL 要求とでは、別々のポートを使用する必要があります。

- CICS Web サポートが着信クライアント要求を listen するポートを予約するには、「*z/OS Communications Server: IP 構成解説書*」 (SC88-8927) で説明されているように、PROFILE.TCPIP データ・セットで PORT ステートメントまたは CICS ジョブ名を指定します。
- プログラムが listen する TCP/IP ポートに対する要求が入っているすべてのキューは、最大長が、PROFILE.TCPIP データ・セットの SOMAXCONN パラメーターで制御されます。 CICS は TCP/IP ポートで listen するので、このパラメーターの値は、TCPIPSERVICE 定義の BACKLOG パラメーターで選択した値と調整しなければなりません。

コード・ページ変換テーブル (DFHCNV) のエントリーのアップグレード

CICS Web サポートにコード・ページ変換テーブルは現在必要ありません。しかし、変換テーブルのエントリーを参照するアナライザー・プログラムが含まれる既存の要求処理構造をお持ちかもしれません。アナライザー・プログラムを変更したくないユーザーは、引き続き DFHCNV エントリーを提供することができます。

このタスクについて

CICS Transaction Server for z/OS バージョン 3 リリース 1 より前の CICS リリースでは、コード・ページ変換テーブル (DFHCNV) が、CICS で使用されるコード・ページと Web クライアントで使用される ASCII コード・ページ間のコード・ページ変換を定義するのに使用されました。CICS Transaction Server for z/OS, バージョン 4 リリース 2 の CICS Web サポートでは、コード・ページ変換テーブル内に新しいエントリーを作成する必要はありません。CICS Web サポートは、z/OS 変換サービスを使用してコード・ページ変換を処理します。

ただし、CICS の前のリリースでコード化した、DFHCNV を参照するアナライザー・プログラムを使用し続ける場合は、コード・ページ変換テーブルのエントリーを指定し続けるか、アナライザー・プログラムを変更する必要があります。アナライザー・プログラムを変更するには、DFHCNV エントリーの名前を指定していた出力パラメーターの代わりに、クライアントおよびサーバーのコード・ページを指定する新しい出力パラメーターを 2 つコーディングします。この変更を行う場合、DFHCNV エントリーをアップグレードする必要はありません。 444 ページの『アナライザー・プログラムの作成』では、この方法で出力パラメーターをコーディングする方法について説明されています。

注: CICS 提供のサンプル・アナライザー DFHWBADX は、提供されているとおりに、サンプル・コード・ページ変換テーブル DFHCNVW$ で定義されたエントリーを指定します。サンプル変換テーブルは、構成しなくても使用可能ですが、DFHWBADX を変換して新しい出力パラメーターを作成することができます。それにより、制御が強力になり、サンプル変換テーブルを使用しなくて済みます。

DFHCNV を使い続けたい場合は、以下のようにします。

手順

1. 以前の CICS リリースで変換テーブルを定義するのに使用した DFHCNV リソース定義マクロをソース内で検索します。 マクロのシーケンスには、コード・ページのペアごとに DFHCNV TYPE=ENTRY マクロが含まれます。
2. 「*CICS 相互通信ガイド*」に記載されているプロセスに従って、マクロを使用して DFHCNV 変換テーブルをセットアップします。 テーブルの定義、アセンブル、およびリンク・エディットを行います。

CICS Web サポートのオペレーションの検査

CICS Web サポートの動作テストをサポートするために、サンプル・プログラム DFH$WB1A (アセンブラー) および DFH$WB1C (C 言語) が提供されています。サンプル・プログラムは **EXEC CICS WEB** および **EXEC CICS DOCUMENT** コマンドを使用して要求を受け取り、単一の応答を作成して送信します。

このタスクについて

提供サンプルの URIMAP リソース DFH$URI1 を使用して、DFH$WB1A または DFH$WB1C にアクセスできます。URIMAP リソースは DFH$WB1A を指示しているので、DFH$WB1C にアクセスするには、DFH$WB1C を指示するようにリソースを変更します。 DFH$WB1A または DFH$WB1C は、CICS 提供のサンプル・アナライザー・プログラム DFHWBADX を使用してアクセスすることもできます。

HTTP クライアントとしての CICSを使用する場合、クライアント要求のパイプライン化のための CICS 提供サンプル・プログラムは、DFH$WB1A と DFH$URI1 がセットアップされた CICS 領域で動作します。

サンプル・プログラムでは、次のように HTTP 応答が構成されます。

```
DFH$WB1A on system applid successfully invoked through CICS web support
```

ここで、 *applid* は、CICS Web サポートが実行されている CICS システムのアプリケーション ID です。

サンプル・プログラムを実行するには、以下のようにします。

手順

1. グループ DFH$SOT で提供されるサンプル TCPIPSERVICE 定義 HTTPNSSL を変更してインストールします。CICS 提供のサンプル・アナライザー・プログラム DFHWBADX は TCPIPSERVICE 定義で指定されています。次のオプションを変更しなければならない場合があります。
   a. **PORTNUMBER**: HTTPNSSL は、HTTP の予約済みポート番号であるポート 80 を使用します。ポート 80 を CICS で使用するように予約していない場合、CICS で使用するように予約済みの、 z/OS Communications Server に属する別のポートを指定します。
   b. **HOST** または **IPADDRESS**: HTTPNSSL は IP アドレスを指定しないため、このオプションは、デフォルトの z/OS Communications Server TCP/IP スタックに対応する IP アドレスにデフォルトで設定されます。この設定が最も

一般的なものです。z/OS イメージに複数の TCP/IP スタックがあり、デフォルトでないスタックを使用する場合は、そのスタックに対応する IP アドレスを指定します。

2. オプション: サンプル・プログラム DFH$WB1C を使用する場合は、以下のようにします。
   a. DFH$WEB リソース定義グループで提供される DFH$WB1C の PROGRAM リソースをインストールします。 DFH$WB1A の PROGRAM リソース定義は、DFHLIST の一部としてインストール済みの、 DFHWEB リソース定義グループ内にあります。
   b. URIMAP リソースを使用する場合は、DFH$WB1C を指示するようにサンプル URIMAP 定義 DFH$URI1 を変更し、リソースをインストールします。サンプル・リソースは DFH$WEB グループにあります。
3. Web ブラウザーで、次の URL 構成要素を使用して、 CICS Web サポートに接続する URL を入力します。

   スキーム

   HTTP

   Host z/OS イメージに割り当てられたホスト名。ホスト名がわからない場合は、HTTPNSSL TCPIPSERVICE 定義のドット 10 進 IP アドレスを使用できます。IP アドレスを明示的に指定しなかった場合は、CICS によって IP アドレスが入力されるので、それをインストール済み TCPIPSERVICE 定義で見ることができます。

   ポート番号

   TCPIPSERVICE 定義で指定されているポート番号。80 の場合、明示的に指定する必要はありません。

   パス

   - DFH$WB1A にアクセスするには、パス `/CICS/CWBA/DFH$WB1A` を使用します。
   - DFH$WB1C にアクセスするには、パス `/sample_web_app` を使用するか (サンプル URIMAP 定義がインストール済みの場合)、パス `/CICS/CWBA/DFH$WB1C` を使用します (代わりにサンプル・アナライザー・プログラムを使用する場合)。
4. オプション: CICS Web サポートが動作していることを確認できたら、サンプル TCPIPSERVICE 定義 HTTPNSSL を廃棄し、URIMAP 定義 DFH$URI1 を使用不可にしてもかまいません。 後で、HTTPNSSL をユーザー独自の TCPIPSERVICE 定義に置き換えることができます。

関連タスク

153 ページの『サンプル・プログラム: HTTP サーバーへの要求のパイプライン化』
サンプル・プログラム DFH$WBPA (アセンブリ言語)、DFH$WBPC (C)、および DFH0WBPO (COBOL) は、CICS で HTTP サーバーへのクライアント要求をどのようにしてパイプライン化できるかを示します。

HTTP TRACE メソッドの構成

デフォルトでは、すべての HTTP TRACE 要求が HTTP 200 (OK) 応答を受け取ります。トレースが使用可能にならないように、この設定をオーバーライドすることができます。

このタスクについて

HTTP TRACE 要求が HTTP 501 (Not Implemented (インプリメントなし)) 応答を受け取るように、HTTP TRACE 要求の設定を変更します。

手順

アセンブリー言語データのみのモジュール DFHWBMTH を作成します。このモジュールでは、ハーフワード長の後に、文字 'NOTRACE' を含んだ 7 バイト・フィールドが続きます。

例

以下は、モジュールを作成するのに役立つ例です。

```
//DFHWBMTH  JOB   'accounting info',name,MSGCLASS=A
      //ASM      EXEC PGM=ASMA90,REGION=2048K,
      //            PARM=(DECK,NOOBJECT,ALIGN)
      //SYSPRINT DD SYSOUT=*
      //SYSLIB   DD DSN=SYS1.MACLIB,DISP=SHR
      //SYSUT1   DD SPACE=(CYL,(3,2))
      //SYSUT2   DD SPACE=(CYL,(1,1))
      //SYSUT3   DD SPACE=(CYL,(1,1))
      //SYSPUNCH DD DSN=&&OBJMOD,DISP=(,PASS),
//             SPACE=(CYL,(1,1)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//SYSIN    DD DATA,DLM='@@'
DFHWBMTH CSECT
DFHWBMTH AMODE 31
DFHWBMTH RMODE ANY
LENGTH   DC AL2(ENDDATA-*)
OPTIONS  DC CL7'NOTRACE'
ENDDATA  EQU *
         END     DFHWBMTH
@@
//LKED     EXEC PGM=IEWL,REGION=2048K,
//            PARM=(LIST,XREF),
//            COND=(7,LT)
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD SPACE=(CYL,(1,1))
//SYSLIN   DD DSN=&&OBJMOD,DISP=(OLD,DELETE)
//         DD DDNAME=SYSIN
//SYSLMOD  DD DSN=CICS.DFHRPL,DISP=SHR
//SYSIN    DD DATA,DLM='@@'
    MODE AMODE(31),RMODE(ANY)
    NAME DFHWBMTH(R)
@@
```

第 5 章 HTTP サーバーとしての CICS に対して CICS Web サポートを使用可能にする

HTTP サーバーとしての CICS のための CICS Web サポート・アーキテクチャーは、実行するタスクによって異なります。すべてのタスクで、CICS Web サポートの構成可能ないくつかの構成要素が必要となります。例えば、インバウンド要求を受け取るポートの TCPIPSERVICE 定義などです。その他の構成可能な構成要素は、特定のタスクの場合にのみ必要です。

始める前に

31 ページの『HTTP サーバーとしての CICS に対する HTTP 要求および応答の処理』を読んで、関連する処理ステージを理解してください。

このタスクについて

サブトピックでは、HTTP 要求に対する応答の提供方法、および Web クライアントと COMMAREA を使用する CICS プログラムの間の通信方法について説明しています。さらに、これらのトピックでは、HTTP を使用する Web クライアントが CICS 内の既存の 3270 表示アプリケーションにアクセスできるようにする方法、およびクライアントから非 HTTP 要求を受け取ってアプリケーション生成応答を提供する方法についても説明しています。

手順

- 193 ページの『第 13 章 CICS Web サポートおよび 3270 表示アプリケーション』
- 187 ページの『第 12 章 CICS Web サポートと非 HTTP 要求』

Web 対応アプリケーション・プログラムによる動的 HTTP 応答の提供

Web 対応アプリケーション・プログラムを使用して、Web クライアントからの HTTP 要求に対して、アプリケーションが生成した応答を提供することができます。

始める前に

65 ページの『第 4 章 CICS Web サポートのコンポーネントの構成』の説明に従って、CICS Web サポートの基本コンポーネントを構成します。

このタスクについて

このタスクには CICS Web サポートの以下のコンポーネントが関連し、いくつかのサブタスクが含まれます。

- TCPIPSERVICE リソース定義
- URIMAP リソース定義

- EXEC CICS WEB プログラミング・インターフェースを使用する Web 対応アプリケーション・プログラム。
- アプリケーション・プログラムの別名トランザクション
- アナライザー・プログラム
- セキュリティー機能
- Web エラー・プログラム

73 ページの図 4には、この CICS Web サポート・タスクのアーキテクチャー要素が示されています。 31 ページの『HTTP サーバーとしての CICS に対する HTTP 要求および応答の処理』では、さまざまな処理要素がどう連携動作するかが説明されています。

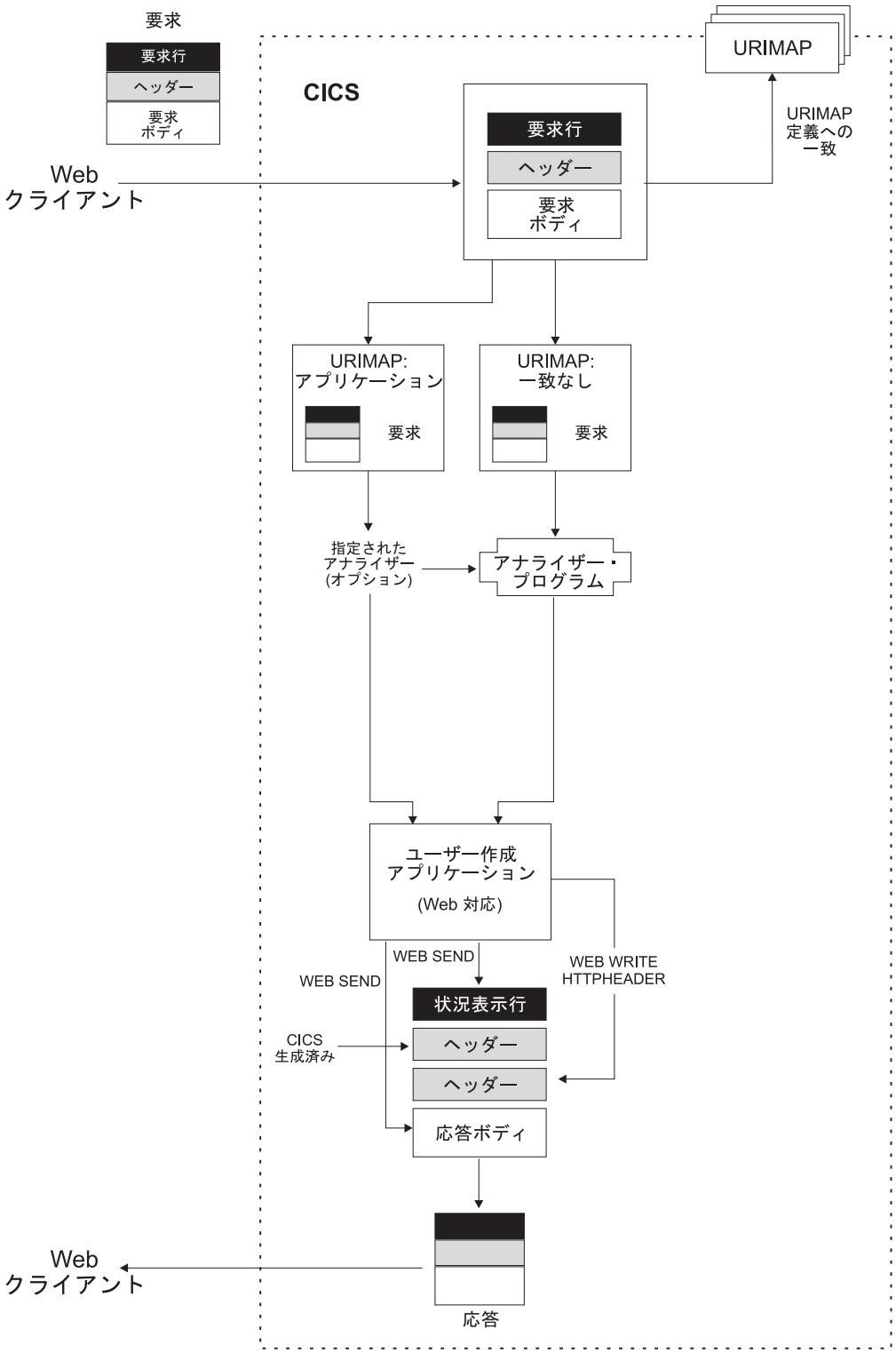


図 4. Web 対応アプリケーション・プログラムによる動的 HTTP 応答

手順

1. 87 ページの『第 6 章 HTTP サーバーとしての CICS のための Web 対応アプリケーション・プログラムの作成』の情報を使用して、Web クライアントによる各要求に対する応答を提供するための Web 対応アプリケーション・プログ

ラムを 1 つ以上設計し、コーディングします。 Web 対応アプリケーション・プログラムでは、EXEC CICS WEB および EXEC CICS DOCUMENT コマンドを使用して、HTTP 要求を受信して分析し、要求に対する応答を書き込んで送信することができます。各プログラムは、単一の要求と応答を処理します。

注: EXEC CICS WEB コマンドを使用する Web 対応アプリケーション・プログラムは、Web クライアントの要求を受信する CICS 領域で実行する必要があります。ただし、これらのアプリケーション・プログラムは、別の CICS 領域のアプリケーション・プログラムにリンクする場合もあります。

2. この CICS Web サポート・タスクのセキュリティー問題を検討します。 CICS は、ユーザーが ID とパスワードを提供する必要がある接続のための HTTP 基本認証を実装することができます。ユーザー ID を使用すると、アプリケーション生成の応答または静的応答に使用される個々のリソースへのアクセスを制御することができます。 インターネットで渡される情報 (基本認証に使用されるユーザー ID およびパスワードを含む) を保護する必要がある場合は、SSL (Secure Sockets Layer) を使用してください。詳細については、 171 ページの『第 11 章 CICS Web サポートのセキュリティー』を参照してください。
3. Web クライアントが各要求に使用する URL を、スキーム、ホストおよびパス構成要素、および任意の照会ストリングを含めて決定します。次を参照してください。 11 ページの『URL の構成要素』. 40 ページの『CICS Web サポートの URL』で、URL を選択する際に考慮すべき要因と制限について説明しています。
4. 要求に使用されるポートを決定します。次を参照してください。 66 ページの『CICS Web サポート用のポートの予約』. HTTP の場合のデフォルトのポート番号は 80 で、HTTPS (SSL を使用) の場合のデフォルトのポート番号は 443 です。スキームのデフォルト以外のポート番号は、要求の URL で明示的に指定されます。 必要に応じて、CICS Web サポートに関連付けられているポートを要求で使用することができます。
5. 要求を受信するポートに TCPIPSERVICE 定義がない場合は、 109 ページの『CICS Web サポートの TCPIPSERVICE リソース定義の作成』の手順に従います。 この定義を使用して、ポートに対して選択したセキュリティー手段 (例えば SSL および基本認証を使用する) を指定します。 関連する TCPIPSERVICE 定義の名前は、要求の URIMAP 定義で指定されます。 TCPIPSERVICE 定義を指定しなかった場合、URIMAP 定義に一致した要求は、TCPIPSERVICE 定義が存在している任意のポートを使用することができます。
6. 関係のあるユーザー・アプリケーション・プログラムの別名トランザクション ID を選択します。デフォルトの別名トランザクションは CWBA です。独自の別名トランザクションを作成する場合は、 114 ページの『CICS Web サポート用の TRANSACTION リソース定義の作成』を参照してください。 URIMAP 定義またはアナライザー・プログラムを使用して、HTTP 要求ごとに別名トランザクションを指定することができます。 Web クライアントのユーザー ID を使用してリソース・レベルのセキュリティーを実装する場合は、このトランザクションにはこれらのユーザー ID が適用されます。また、これらのユーザー ID には、トランザクションによって使用される、保護されている CICS リソースおよびコマンドへのアクセス権が必要になります。
7. TCPIPSERVICE 定義に関連付けられたアナライザー・プログラム (441 ページの『付録 E. アナライザー・プログラム』を参照) をどのように使用するかを決

定し、適切なプログラムを選択します。 Web 対応アプリケーションの場合は、以下の戦略から選択することができます。

a. CICS 提供のデフォルトのアナライザー・プログラム DFHWBAAX を使用してエラー処理を行う。このポートを使用している要求がすべて URIMAP 定義を使用して処理される場合は、DFHWBAAX が適しています。 一致する URIMAP 定義が見つかった場合、DFHWBAAX はアクションを実行しません。一致する URIMAP 定義が見つからなかった場合、DFHWBAAX は、ユーザーによる置換が可能な Web エラー・プログラム DFHWBERX に制御を渡して、エラー応答を生成します。

b. CICS 提供のサンプル・アナライザー・プログラム DFHWBADX を使用して、URIMAP 定義を使用する要求、および CICS TS 3.1 よりも前に CICS Web サポートが使用していたのと同じプロセスに従う要求に、基本サポートを提供します。 一致する URIMAP 定義が見つかった場合、DFHWBADX はアクションを実行しません。一致する URIMAP 定義が見つからなかった場合は、CICS TS 3.1 よりも前に要求されていた形式で URL を分析します。分析に失敗した場合、DFHWBADX は、ユーザーによる置換が可能な Web エラー・プログラム DFHWBEP に制御を渡し、エラー応答を生成します (URIMAP 定義で指定された URL が、CICS TS 3.1 よりも前に必要とされていた形式に合わない場合は、意味のある応答が提供されるよう、DFHWBADX または DFHWBEP をカスタマイズします)。

c. 独自のアナライザー・プログラムを使用して、カスタマイズされたサポートを提供する。

- URIMAP 定義を使用している要求に対する処理に動的な変更を加える。
- 要求に対する処理中に、モニター・アクションまたは監査アクションを提供する。
- CICS TS 3.1 よりも前に CICS Web サポートが使用していたのと同じプロセスに従う要求をサポートする。
- ユーザーによる置換が可能な Web エラー・プログラム DFHWBEP および DFHWBERX を使用してエラー応答を提供する。

URIMAP 定義の ANALYZER(YES) 属性を使用して、カスタマイズされたアナライザー・プログラムを指定できます。こうすると、このプログラムが要求の処理パスに含まれるようになります。前に説明したとおり、DFHWBAAX および DFHWBADX は、URIMAP 定義から呼び出された場合はアクションを実行しません。

8. HTTP 要求について、およびユーザー・アプリケーション・プログラムがそれらの HTTP 要求に提供する応答について行うコード・ページ変換の方法を決定します。 コード・ページ変換 (49 ページの『CICS Web サポートのコード・ページ変換』を参照) では通常、ASCII 文字セットを使用して行われた Web クライアント要求を、アプリケーション・プログラムで使用される EBCDIC コード・ページに変換し、その後このプロセスを逆にして、アプリケーション・プログラム出力を Web クライアントに返します。コード・ページ変換の設定は、Web 対応アプリケーション・プログラムが発行する **EXEC CICS** WEB API コマンドで指定できます。
9. 各要求を処理するための URIMAP 定義をセットアップします。 115 ページの『HTTP サーバーとしての CICS のための URIMAP リソースの作成』の手順に従ってください。 URIMAP 定義を使用しなくても、CICS TS 3.1 よりも前

に CICS Web サポートが使用していたのと同じプロセスに従って、アナライザー・プログラムに HTTP 要求を直接渡すことができます。ただし、URIMAP 定義を使用する方が、HTTP 要求の管理を容易に行うことができます。URIMAP 定義を使用しない場合、CICS が特定の HTTP 要求に応答する方法を変更するには、アナライザー・プログラムのロジックを変更する必要があります。URIMAP 定義を使用すると、これらの変更をシステム管理タスクとして動的に実行することができます。

10. ユーザーによる置換が可能な Web エラー・プログラム (131 ページの『第 9 章 Web エラー・プログラム』を参照) が Web クライアントに適切な応答を提供するようにしておきます。 初期処理時エラー、異常終了、または障害が CICS Web サポート・プロセスで発生した場合は、Web エラー・プログラムが使用されます。すべてのエラー状態で Web エラー・プログラムが使用されるわけではありません。

CICS 文書テンプレートまたは z/OS UNIX ファイルによる静的 HTTP 応答の提供

CICS 文書テンプレートまたは z/OS UNIX システム・サービス・ファイルを使用して、Web クライアントからの HTTP 要求に対する静的応答を提供することができます。

始める前に

65 ページの『第 4 章 CICS Web サポートのコンポーネントの構成』の説明に従って、CICS Web サポートの基本コンポーネントを構成します。

このタスクについて

このタスクには CICS Web サポートの以下のコンポーネントが関連し、いくつかのサブタスクが含まれます。

- TCPIPSERVICE リソース定義
- URIMAP リソース定義
- z/OS UNIX ファイル
- CICS 文書テンプレート・サポート
- セキュリティー機能
- Web エラー・プログラム

77 ページの図 5には、この CICS Web サポート・タスクのアーキテクチャー要素が示されています。 31 ページの『HTTP サーバーとしての CICS に対する HTTP 要求および応答の処理』では、さまざまな処理要素がどう連携動作するかが説明されています。

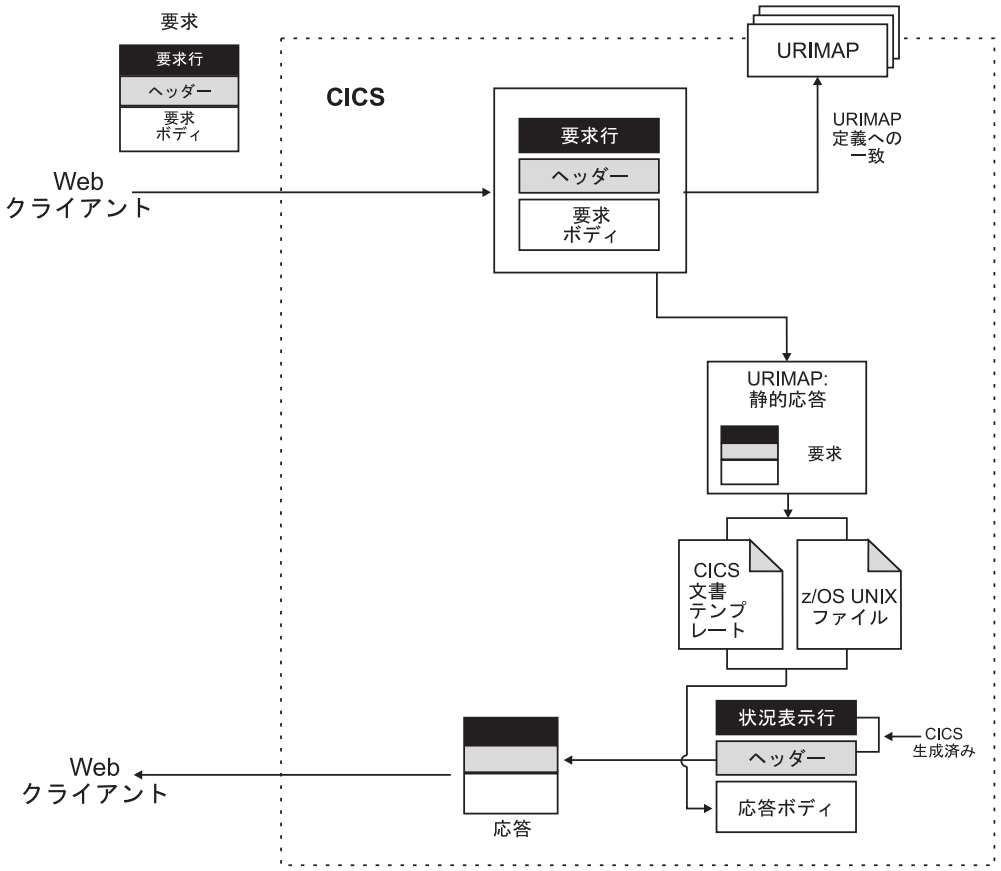


図 5. 静的 HTTP 応答

手順

1. この CICS Web サポート・タスクのセキュリティー問題を検討します。 CICS は、ユーザーが ID とパスワードを提供する必要がある接続のための HTTP 基本認証を実装することができます。ユーザー ID を使用すると、アプリケーション生成の応答または静的応答に使用される個々のリソースへのアクセスを制御することができます。 インターネットで渡される情報 (基本認証に使用されるユーザー ID およびパスワードを含む) を保護する必要がある場合は、SSL (Secure Sockets Layer) を使用してください。詳細については、 171 ページの『第 11 章 CICS Web サポートのセキュリティー』を参照してください。
2. z/OS UNIX システム・サービス・ファイルを使用して応答を提供する場合は、そのファイルを作成し、z/OS UNIX ファイル・システム内の任意の場所にそのファイルを配置します。 Web クライアントの要求に一致する URIMAP 定義によってこの応答が識別されると、CICS はそのファイルを取得し、適切なコード・ページ変換を実行します。以下の点に注意してください。
   a. z/OS UNIX ファイルには、HTTP ヘッダーや状況表示行情報は含めないでください。 応答が送信されると、要求された情報が CICS によって生成されます。z/OS UNIX ファイルは応答のボディを提供するだけです。
   b. このタスクで後ほど説明するように、パスの突き合わせを使用する場合は、このファイルの場所が重要になります。パスの突き合わせを使用しない場合は、このファイルの場所が要求の URL と関係を持つ必要はありません。

c. CICS 領域は、z/OS UNIX へのアクセス権、およびこのファイルを含むディレクトリーおよびこのファイルそのものに対するアクセス権を持っている必要があります。 *Java Applications in CICS*を参照してください。

d. Web クライアントのユーザー ID を使用してアクセス制御を実装している場合は、そのユーザー ID も、そのファイルを含むディレクトリーおよびそのファイルそのものへのアクセス権限が必要になります。 177 ページの『CICS システムと CICS Web サポートのリソース・セキュリティー』を参照してください。

3. CICS 文書テンプレートを使用して応答を提供する場合は、 「*CICS アプリケーション・プログラミング・ガイド*」の手順に従って文書テンプレートを作成します。 文書テンプレートは DOCTEMPLATE リソース定義を使用して定義されます。文書テンプレートは、区分データ・セット、CICS プログラム、ファイル、一時記憶域キュー、一時データ・キュー、出口プログラム、または z/OS UNIX システム・サービス・ファイルに保持することができます。 Web クライアントの要求に一致する URIMAP 定義によってこの応答が識別されると、CICS はテンプレートを使用して文書を作成し、その文書を取得して、適切なコード・ページ変換を実行します。

   a. 文書テンプレートには HTTP ヘッダーまたは状況表示行情報は含めないでください。 応答が送信されると、要求された情報が CICS によって生成されます。文書テンプレートは、応答のボディを提供するだけです。

   b. 名前と値のペアで構成される照会ストリングをシンボル・リストとして使用して、文書テンプレート内で置き換えることができます。 (照会ストリングが、URIMAP 定義の PATH 属性の一部としてすでに URIMAP の突き合わせに使用されている場合は、その照会ストリングをこのように使用することはできません。) URL 内の予期される形式の照会ストリングをクライアントが送信するようにするには、ユーザーが入力する HTML フォームを GET メソッドで送信します。 照会ストリング内の名前はすべてシンボルとして文書テンプレートにコーディングすることができるので、そのテンプレートを使用する場合、CICS は、その照会ストリングで指定されている値の代わりに各シンボルを使用します。例えば、名前と値のペア `username=Peter` を含む照会ストリングを取得した場合は、`username` をシンボルとしてコーディングして、この照会ストリングを文書テンプレートで使用することができます。

      ```
      財政システムにようこそ、&username;。
      ```

      この結果得られる、ユーザーに送達される静的応答は、以下のようになります。

      ```
      財政システムにようこそ、Peter。
      ```

      注: 文書テンプレート内のシンボルは、大/小文字を区別します。名前は、元の照会ストリング内と同じ大/小文字を使用して指定してください。
      文書テンプレート内のシンボルに対応しない名前と値のペアはすべて無視されます。

   c. Web クライアントのユーザー ID を使用してリソース・レベルのセキュリティーを実装している場合は、そのユーザー ID には、文書テンプレートへのアクセス権が必要になります。 177 ページの『CICS システムと CICS Web サポートのリソース・セキュリティー』を参照してください。文書テ

ンプレートが z/OS UNIX システム・サービス・ファイルである場合、Web クライアントにはファイルに対するアクセス権ではなく、DOCTEMPLATE リソース定義に対するアクセス権のみを与えればよいことに注意してください。

4. z/OS UNIX ファイル、または CICS 文書テンプレートによって提供されるメディア・タイプ (データ内容のタイプ) を識別します。 11 ページの『IANA メディア・タイプおよび文字セット』を参照してください。 URIMAP 定義を使用して静的応答を送信する場合は、品質係数 (「q」パラメーター) の使用はサポートされていないので注意してください。品質係数を使用して、Accept ヘッダーで指定されている許容メディア・タイプまたは文字セットのクライアント・リストの中から選択します。このタイプの分析を行う場合は、代わりにアプリケーション生成応答を使用できます。
5. 静的応答のコード・ページ変換を行うために CICS が要求する情報を識別します。 コード・ページ変換は、テキスト・メディア・タイプが指定されている場合のみ実行されます。 49 ページの『CICS Web サポートのコード・ページ変換』を参照してください。
   a. CICS が静的応答を Web クライアントに送信する前に、その静的応答の変換先となる文字セットを識別します。 413 ページの『付録 A. HTML コード化文字セット』では、コード・ページ変換用に CICS でサポートされている IANA 文字セットがリストされています。
   b. 応答ボディを提供する文書テンプレートまたは z/OS UNIX ファイルがエンコードされている IBM コード・ページ (EBCDIC) を識別します。

   静的応答の場合、この情報はその要求の URIMAP 定義で指定されています。
6. Web クライアントが各要求に使用する URL を、スキーム、ホストおよびパス構成要素、および任意の照会ストリングを含めて決定します。次を参照してください。 11 ページの『URL の構成要素』. 40 ページの『CICS Web サポートの URL』で、URL を選択する際に考慮すべき要因と制限について説明しています。
7. URIMAP 定義でパスの突き合わせを使用する場合は、要求 URL の計画を立て、CICS 文書テンプレートの名前、またはこれをサポートするための z/OS UNIX ファイルの場所を準備します。 パスの突き合わせでは、URIMAP 定義のパス構成要素、および URIMAP 定義で指定されている CICS 文書テンプレートまたは z/OS UNIX ファイルの名前においてもワイルドカード文字が使用されます。ワイルドカード文字に一致したパスの部分は、応答を提供するための文書テンプレートまたは z/OS UNIX ファイルを選択する場合に使用されます。
   a. CICS 文書テンプレートの場合、ワイルドカード文字に一致したパスの部分は、テンプレート名の最後の部分として置換されます。名前が同じように始まる文書テンプレートのコレクションを作成し、単一の URIMAP 定義を使用して、パスが同じように始まる要求 URL を使用してそれらの文書テンプレートにアクセスできます。
   b. z/OS UNIX ファイルの場合、ワイルドカード文字に一致したパスの部分は、ファイル名の最後の部分として置換されます。同一ディレクトリーにこれらのファイルを多数保管し、単一の URIMAP 定義を使用して、パスが同じように始まる要求 URL を使用してそれらのファイルにアクセスできます。 URIMAP 定義はデータ内容のタイプ (MEDIATYPE 属性) を指定する

必要があるため、同じタイプのデータ内容を作成する z/OS UNIX ファイルのグループを処理できるのは、単一の URIMAP 定義のみであることを忘れないでください。

8. 要求に使用されるポートを決定します。次を参照してください。 66 ページの『CICS Web サポート用のポートの予約』. HTTP の場合のデフォルトのポート番号は 80 で、HTTPS (SSL を使用) の場合のデフォルトのポート番号は 443 です。スキームのデフォルト以外のポート番号は、要求の URL で明示的に指定されます。 必要に応じて、CICS Web サポートに関連付けられているポートを要求で使用することができます。
9. 要求を受信するポートに TCPIPSERVICE 定義がない場合は、 109 ページの『CICS Web サポートの TCPIPSERVICE リソース定義の作成』の手順に従います。 この定義を使用して、ポートに対して選択したセキュリティー手段 (例えば SSL および基本認証を使用する) を指定します。 関連する TCPIPSERVICE 定義の名前は、要求の URIMAP 定義で指定されます。 TCPIPSERVICE 定義を指定しなかった場合、URIMAP 定義に一致した要求は、TCPIPSERVICE 定義が存在している任意のポートを使用することができます。
10. 各要求を処理するための URIMAP 定義をセットアップします。 115 ページの『HTTP サーバーとしての CICS のための URIMAP リソースの作成』の手順に従ってください。 URIMAP 定義は、z/OS UNIX ファイルまたは文書テンプレートのどちらかを識別できます。
11. この CICS Web サポート・タスクのエラー処理手順をチェックします。
    a. 要求が受信されるポートの TCPIPSERVICE 定義に関連付けられているアナライザー・プログラムの動作を確認します。要求に対して URIMAP の突き合わせが失敗した場合、その要求はアナライザー・プログラムに渡されます。 ポートが静的応答にのみ使用されている場合は、CICS 提供のデフォルトのアナライザー・プログラム DFHWBAAX は、適切なエラー処理を行います。ポートが静的応答以外にも使用されている場合は、アナライザー・プログラムの選択は、ユーザー・アプリケーション・プログラムの要件に応じたものになる可能性があります。静的応答に対して適切なエラー処理を提供するようにアナライザー・プログラムのカスタマイズを行うことが必要な場合もあります。 441 ページの『付録 E. アナライザー・プログラム』を参照してください。
    b. アーキテクチャーに含まれている、ユーザーによる置換が可能な Web エラー・プログラムが Web クライアントに適切な応答を提供することを確認します。 131 ページの『第 9 章 Web エラー・プログラム』を参照してください。

関連情報

CICS 領域に対する z/OS UNIX ディレクトリーおよびファイルへのアクセス権限の付与

z/OS UNIX 上の CICS Web サポートのリソース

z/OS UNIX ファイルを使用して Web クライアントからの要求に対して静的応答を提供する場合、それらの要求を受信して応答を提供する CICS 領域には、ファイルとそのディレクトリーに対する*読み取り* 権限が必要です。

CICS 領域のホーム・ディレクトリー下のディレクトリー構造に、各 CICS 領域に関係するファイルをすべて保管している場合は、その CICS 領域を各ディレクトリーおよびファイルの (適切な所有者アクセス権を持つ) 所有者とすることができます。複数の CICS 領域がいくつかの z/OS UNIX ファイルを使用する場合は、グループ・アクセス権またはアクセス制御リスト (ACL) を使用します。「その他の」アクセス権を使用することは、どの z/OS UNIX ユーザーにもアクセス権限を与えることになるため、実稼働環境内の CICS Web サポートにはおそらく適していません。

関連情報

CICS 領域に対する z/OS UNIX ディレクトリーおよびファイルへのアクセス権限の付与

COMMAREA アプリケーションへの Web クライアント・アクセスの提供

CICS Web サポートを使用すると、Web クライアントは、 COMMAREA インターフェースを使用して他のプログラムと通信する CICS アプリケーションと相互作用することができます。Web 対応アプリケーション・プログラムは、このアプリケーションにリンクし、その出力を使用して HTTP 応答を提供できます。あるいは、コンバーター・プログラムが Web クライアントからの入力を適切な COMMAREA に変換し、アプリケーションからの出力を HTTP 応答に変換することもできます。

始める前に

65 ページの『第 4 章 CICS Web サポートのコンポーネントの構成』の説明に従って、CICS Web サポートの基本コンポーネントを構成します。

このタスクについて

このタスクには CICS Web サポートの以下のコンポーネントが関連し、いくつかのサブタスクが含まれます。

- TCPIPSERVICE リソース定義
- URIMAP リソース定義
- COMMAREA アプリケーション・プログラム
- **EXEC CICS** WEB プログラミング・インターフェースを使用し、 COMMAREA アプリケーション・プログラムにリンクして、そのアプリケーション・プログラムの出力を使用する Web 対応アプリケーション・プログラム
- または、適切な COMMAREA 入力を提供し、アプリケーション・プログラムからの出力を HTTP 応答に変換することのできるコンバーター・プログラムのいずれか
- このプロセスに関係するユーザー・アプリケーション・プログラムを補う別名トランザクション
- アナライザー・プログラム
- セキュリティー機能
- Web エラー・プログラム

82 ページの図 6には、この CICS Web サポート・タスクのアーキテクチャー要素が示されています。 31 ページの『HTTP サーバーとしての CICS に対する HTTP

要求および応答の処理』では、さまざまな処理要素がどう連携動作するかが説明されています。

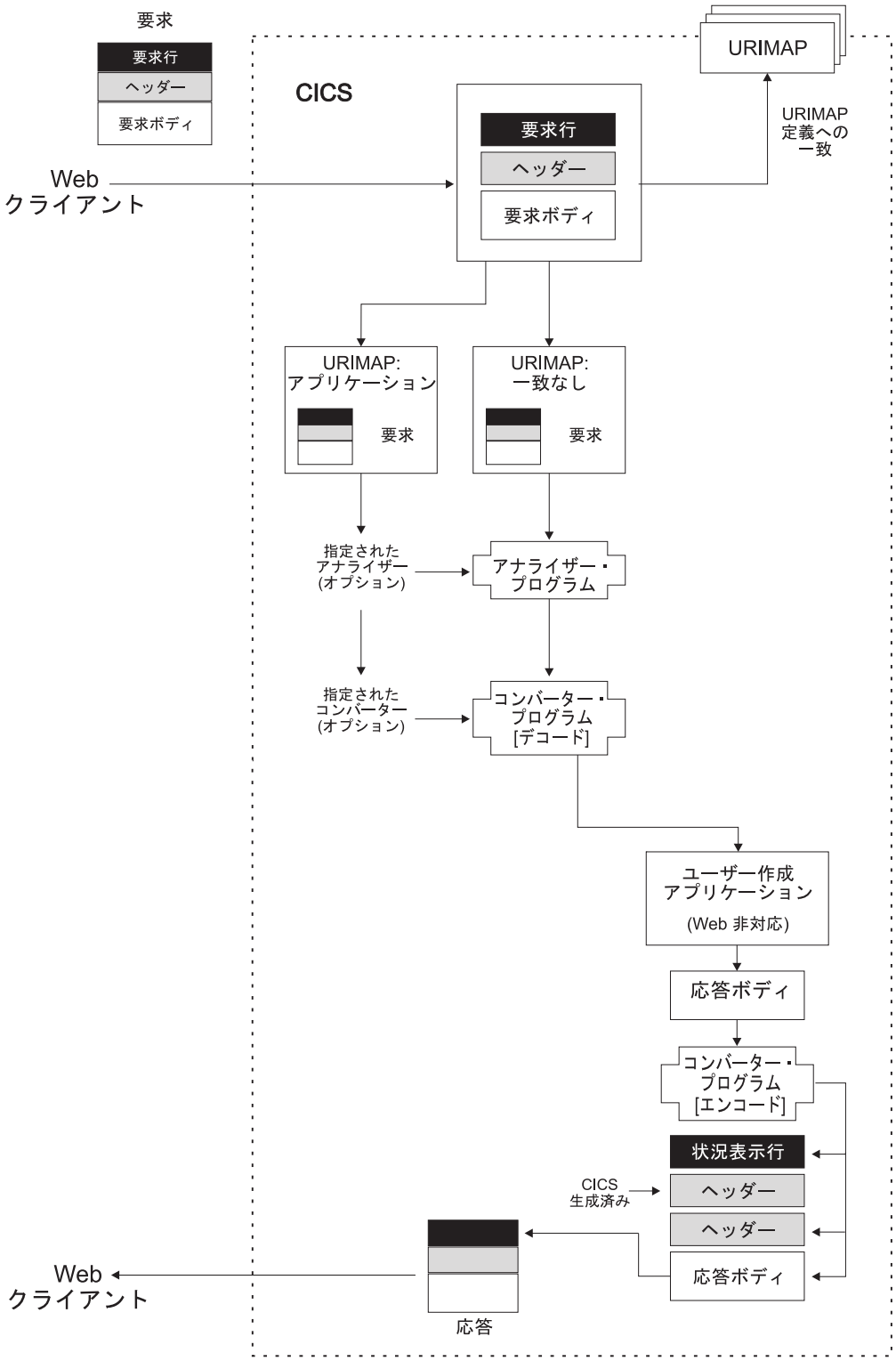


図 6. COMMAREA アプリケーションからの HTTP 応答

手順

1. HTTP 要求を処理して COMMAREA アプリケーション・プログラムにリンクする Web 対応アプリケーション・プログラムを作成できます。または、Web クライアントからの入力を適切な COMMAREA に変換した後にアプリケーションからの出力を HTTP 応答に変換するためのコンバーター・プログラムを作成することもできます。コンバーター・プログラムは、 **EXEC CICS** WEB API コマンドを使用して Web クライアントの要求を読み取り、応答を作成するか、またはストレージ・ブロックで要求および応答を処理することができます。

   a. Web 対応アプリケーション・プログラムを使用する場合は、71 ページの『Web 対応アプリケーション・プログラムによる動的 HTTP 応答の提供』の手順に従ってください。 COMMAREA アプリケーションにリンクして、その出力を使用するよう Web 対応アプリケーション・プログラムをコーディングします。 Web 対応アプリケーション・プログラムは、アナライザー・プログラムが (ユーザー・トークンまたは共用作業域で) 次の処理ステージに渡すために作成した情報を受け取ることはできません。コンバーター・プログラムを使用する必要があります。新規の CICS Web サポート・アプリケーションを開発する場合は、この制限は関係ありません。

   b. コンバーター・プログラムを使用する場合は、このタスクの手順に従ってください。

2. この CICS Web サポート・タスクのセキュリティー問題を検討します。 CICS は、ユーザーが ID とパスワードを提供する必要がある接続のための HTTP 基本認証を実装することができます。ユーザー ID を使用すると、アプリケーション生成の応答または静的応答に使用される個々のリソースへのアクセスを制御することができます。 インターネットで渡される情報 (基本認証に使用されるユーザー ID およびパスワードを含む) を保護する必要がある場合は、SSL (Secure Sockets Layer) を使用してください。詳細については、171 ページの『第 11 章 CICS Web サポートのセキュリティー』を参照してください。

3. TCPIPSERVICE 定義に関連付けられたアナライザー・プログラムをどのように使用するかを決定し、適切なプログラムを選択します。 441 ページの『付録 E. アナライザー・プログラム』を参照してください。 URIMAP 定義またはアナライザー・プログラムを使用して、Web クライアントからの要求を適切なコンバーター・プログラムおよびユーザー作成アプリケーション・プログラムにマップすることができます。 Web 非対応アプリケーションでは、URIMAP 定義があっても、以下のような状況では、要求の処理パスで、カスタマイズされたアナライザー・プログラムを使用することが必要です。

   a. コンバーター・プログラムでストレージ・ブロック内の要求および応答を処理し、非標準コード・ページ変換が必要な場合。 コンバーター・プログラムには、HTTP 要求および応答を含むストレージ・ブロックにコード・ページ変換を指定する仕組みがありません。アナライザー・プログラムがない場合、CICS では、 466 ページの『コンバーター・プログラムの作成』で説明されている標準設定を使用して、入力時および出力時の両方で、ストレージ・ブロックに提供されているメッセージ・ボディを変換します。この動作が適していない場合は、処理パスでアナライザー・プログラムを使用して代替設定を指定するか、またはストレージ・ブロックを使用して処理する代わ

りに **EXEC CICS** WEB API コマンドを使用するかのいずれかを行います。詳細については、 49 ページの『CICS Web サポートのコード・ページ変換』を参照してください。

b. 標準入力のほかにコンバーター・プログラムとなんらかの情報を伝達する必要がある場合。 ユーザー・トークンを提供します。これは、少量の情報を交換するため、またはさらに多くの情報が入っている共用作業域のアドレスを交換するために、アナライザー・プログラムまたはコンバーター・プログラムで使用できるものです。

c. アナライザー・プログラムによって実行可能な、モニター・アクションまたは監査アクションが必要な場合。

d. 使用されているコンバーター・プログラム、要求を処理するアプリケーション・プログラム、または要求に使用される別名トランザクションおよびユーザー ID などのプロセスの要素に、動的な変更を加える必要がある場合。

これらの機能のいずれもが不要の場合は、CICS 提供のデフォルト・アナライザー・プログラム DFHWBAAX を使用するか、または CICS 提供のサンプル・アナライザー・プログラム DFHWBADX を使用して、基本的なエラー処理を行うことができます。このポートを使用している要求がすべて URIMAP 定義を使用して処理される場合は、DFHWBAAX が適しています。DFHWBADX は、URIMAP 定義を使用する要求、および CICS Web サポートが CICS TS 3.1 よりも前に使用していたのと同じプロセスに従う要求の両方に基本サポートを提供しています。

4. 適切なコンバーター・プログラムを作成するには、 465 ページの『付録 F. コンバーター・プログラム』の情報を使用してください。 コンバーター・プログラムは 2 回呼び出されます。最初は、デコード機能を使用して、Web クライアントの要求、および URIMAP 定義またはアナライザー・プログラムによって提供される追加情報を調べて、アプリケーションに渡す適切な COMMAREA を作成するために呼び出されます。次に、エンコード機能のためにコンバーター・プログラムが呼び出されます。この機能で、アプリケーション・プログラムの出力を受信し、HTTP 応答を作成します。複数のアプリケーション・プログラムがデータを提供することになっている場合、コンバーター・プログラムはデコード機能を繰り返し呼び出すことができます。 472 ページの『コンバーター・プログラムからの複数のアプリケーション・プログラムの呼び出し』を参照してください。

5. Web クライアントが各要求に使用する URL を、スキーム、ホストおよびパス構成要素、および任意の照会ストリングを含めて決定します。次を参照してください。 11 ページの『URL の構成要素』. 40 ページの『CICS Web サポートの URL』で、URL を選択する際に考慮すべき要因と制限について説明しています。

6. 要求に使用されるポートを決定します。次を参照してください。 66 ページの『CICS Web サポート用のポートの予約』. HTTP の場合のデフォルトのポート番号は 80 で、HTTPS (SSL を使用) の場合のデフォルトのポート番号は 443 です。スキームのデフォルト以外のポート番号は、要求の URL で明示的に指定されます。 必要に応じて、CICS Web サポートに関連付けられているポートを要求で使用することができます。

7. 関係のあるユーザー・アプリケーション・プログラムの別名トランザクション ID を選択します。デフォルトの別名トランザクションは CWBA です。独自の

別名トランザクションを作成する場合は、 114 ページの『CICS Web サポート用の TRANSACTION リソース定義の作成』を参照してください。 URIMAP 定義またはアナライザー・プログラムを使用して、HTTP 要求ごとに別名トランザクションを指定することができます。 Web クライアントのユーザー ID を使用してリソース・レベルのセキュリティーを実装する場合は、このトランザクションにはこれらのユーザー ID が適用されます。また、これらのユーザー ID には、トランザクションによって使用される、保護されている CICS リソースおよびコマンドへのアクセス権が必要になります。

8. 各要求を処理するための URIMAP 定義をセットアップします。 115 ページの『HTTP サーバーとしての CICS のための URIMAP リソースの作成』の手順に従ってください。 URIMAP 定義を使用しなくても、CICS TS 3.1 よりも前に CICS Web サポートが使用していたのと同じプロセスに従って、アナライザー・プログラムに HTTP 要求を直接渡すことができます。ただし、URIMAP 定義を使用する方が、HTTP 要求の管理を容易に行うことができます。 URIMAP 定義を使用しない場合、CICS が特定の HTTP 要求に応答する方法を変更するには、アナライザー・プログラムのロジックを変更する必要があります。URIMAP 定義を使用すると、これらの変更をシステム管理タスクとして動的に実行することができます。
9. 要求を受信するポートに TCPIPSERVICE 定義がない場合は、 109 ページの『CICS Web サポートの TCPIPSERVICE リソース定義の作成』の手順に従います。 この定義を使用して、ポートに対して選択したセキュリティー手段 (例えば SSL および基本認証を使用する) を指定します。 関連する TCPIPSERVICE 定義の名前は、要求の URIMAP 定義で指定されます。 TCPIPSERVICE 定義を指定しなかった場合、URIMAP 定義に一致した要求は、TCPIPSERVICE 定義が存在している任意のポートを使用することができます。
10. この CICS Web サポート・タスクのエラー処理手順をチェックします。
    a. 要求が受信されるポートの TCPIPSERVICE 定義に関連付けられているアナライザー・プログラムの動作を確認します。要求に対して URIMAP の突き合わせが失敗した場合、その要求はアナライザー・プログラムに渡されます。 詳細については、 441 ページの『付録 E. アナライザー・プログラム』を参照してください。
    b. ユーザーによる置換が可能な Web エラー・プログラムが Web クライアントに適切な応答を提供するようにします。 131 ページの『第 9 章 Web エラー・プログラム』を参照してください。

第 6 章 HTTP サーバーとしての CICS のための Web 対応アプリケーション・プログラムの作成

CICS では、Web 対応アプリケーション・プログラムは、CICS を通じて Web クライアントまたはサーバーと対話するために **EXEC CICS** WEB コマンドを使用します。HTTP サーバーとしての CICS の場合、これらのプログラムは HTTP 要求を受信して分析し、アプリケーション生成応答を Web クライアントに提供します。

始める前に

HTTP サーバーとしての CICS 用の Web 対応アプリケーション・プログラムのコーディングを開始する前に、処理を理解するために 31 ページの『HTTP サーバーとしての CICS に対する HTTP 要求および応答の処理』をお読みください。

Web クライアントに提供するサービスを HTTP プロトコル仕様、特に HTTP/1.1 に準拠させる場合は、CICS およびユーザー・アプリケーションが実行できるアクションの詳細について、 54 ページの『HTTP サーバーとしての CICS の HTTP/1.1 準拠』をお読みください。

このタスクについて

CICS では、要求の URIMAP 定義で指定されている Web 対応アプリケーション・プログラムか、またはアナライザーを使用する場合は、アナライザー・プログラムによって指定されている Web 対応アプリケーション・プログラムを、アプリケーション生成の応答を必要とする HTTP 要求ごとに呼び出します。アプリケーション・プログラムの指定に URIMAP 定義を使用する場合、特定の URL を使用してすべての要求にサービスを提供するように、単一のアプリケーション・プログラムを選択することができます。 URIMAP 定義の代わりに、または URIMAP 定義に加えて、アナライザー・プログラムを使用すれば、要求を分析して代替アプリケーション・プログラムを決定できます。

要確認: EXEC CICS WEB コマンドを使用する Web 対応アプリケーション・プログラムは、Web クライアントの要求を受信する CICS 領域で実行する必要があります。ただし、これらのアプリケーション・プログラムは、例えばビジネス・ロジックを実行するための、別の CICS 領域のアプリケーション・プログラムにリンクする場合もあります。

HTTP サーバーとしての CICS の場合、アプリケーション・プログラムは、要求に対する応答を送信して CICS に制御を返すと、それ以降の Web クライアントからの要求を待機しません。要求が論理シーケンスを形成していたり、持続接続を使用して行われていたり、パイプライン化されていたりしても同じです。一連の要求と応答の間、別々のプログラム (または同じプログラムの新規インスタンス) 間で情報を共用する必要がある場合は、CICS 管理リソースを使用するか、または Web クライアントによって送信された要求の要素を使用することにより可能になります。

HTTP サーバーとしての CICS に、**EXEC CICS** WEB コマンドを使用する場合、SESSTOKEN オプションはありません。 SESSTOKEN オプションは、コマンドが HTTP クライアントとしての CICS に対して使用されていることを示します。

収集した情報を使用して、要求処理のビジネス・ロジックを実行できます。処理実行のために他のアプリケーション・プログラムも使用したい場合があります。Web 対応アプリケーション・プログラムは、Web 非対応プログラムから受信された情報に基づいて、 HTTP 要求に対する応答を作成できます。表示ロジックからビジネス・ロジックを分離するようにしてください。Web 対応アプリケーション・プログラムでは、表示ロジックによって Web クライアントとの対話が制御されます。詳細については、「*CICS アプリケーション・プログラミング・ガイド*」を参照してください。

サブトピックのガイダンスに従うことにより、HTTP 要求を処理する Web 対応アプリケーション・プログラムをコーディングできます。

HTTP 要求の要求行の検査

CICS は、アプリケーション・プログラムが必要に応じてアクセスできるよう、HTTP 要求ごとに使用される要求行を保管します。アプリケーション・プログラムでは、WEB EXTRACT コマンドを使用して要求 URL の構成要素 (パス、ホスト名、ポート番号、および照会ストリングを含む)、要求に使用されるメソッド、または要求の HTTP バージョンを抽出することができます。非 HTTP 要求もこの方法で識別することができます。

このタスクについて

要求行内の項目については、 13 ページの『HTTP プロトコル』を参照してください。要求 URL は、要求行の主要な要素です。 11 ページの『URL の構成要素』で、URL の各部分について説明しています。アプリケーション・プログラムは、要求を処理して適切な応答を提供できるように、要求行に含まれる何らかの項目を検査する場合があります。要求行から情報を抽出する一般的な理由として、以下が挙げられます。

- 論理要求シーケンスの一部、または同じリソースに関係するさまざまな要求など、いくつかの異なる要求を処理するために同じアプリケーション・プログラムが呼び出されるようにするため。
- どのアクションが HTTP メソッドによってアプリケーションから要求されているのかを知るため。 435 ページの『付録 D. CICS Web サポートの HTTP メソッド・リファレンス』では、ある要求に対して Web クライアントが使用するさまざまなメソッドについて説明されており、それぞれの場合に適切なアクションが提案されています。
- URL のパス構成要素を使用するため。この構成要素で、要求の適用先のリソースを識別できます。要求を処理アプリケーションにマップするのに使用されるようにすることに加えて、アプリケーションに処理情報を提供するように URL のパス構成要素を設計できます。例えば、アプリケーションが提供する特定の機能を、パス構成要素で指定できます。または、Web 対応アプリケーションが他の複数のアプリケーションにフロントエンドを提供している場合、URL のパス構成要素によって、要求の適用先アプリケーションを識別することができます。 40 ページの『CICS Web サポートの URL』で、そのための方法を説明しています。

- アプリケーションによる処理のための照会ストリングを取得するため。
- アプリケーションが適切な応答を提供できるようにするために、Web クライアントの HTTP バージョンを識別するため。Web クライアントが使用する HTTP バージョンが、応答の HTTP ヘッダー、状況コード、およびメッセージ内容に影響が与えることがあります。 HTTP/1.0 クライアントは、HTTP/1.1 仕様に記述された拡張機能に対応していない可能性があります。
- 非 HTTP 要求を識別するため。 187 ページの『第 12 章 CICS Web サポートと非 HTTP 要求』には、非 HTTP 要求の処理に関する詳しい情報が記載されています

オプションの詳細な参照情報については、WEB EXTRACT を参照してください。以下の項目を取得するには、WEB EXTRACT コマンドを使用します。

手順

- 要求の HOST ヘッダー・フィールドまたは要求行のいずれかで指定されている、要求 URL のホスト構成要素を取得する場合は、HOST オプションを使用します (要求に絶対 URI 形式が使用された場合)。
- 要求の HTTP メソッド (例えば GET または PUT) を取得する場合は HTTPMETHOD オプションを使用します。
- HTTP バージョン (HTTP/1.1 または HTTP/1.0) を識別する場合は HTTPVERSION オプションを使用します。
- URL のパス構成要素を取得する場合は PATH オプションを使用します。
- URL に適用されているポート番号を取得する場合は PORTNUMBER オプションを使用します。 サービスの予約済みポート番号は、通常 URL から省略されています。このポート番号が URL に含まれていない場合は、WEB EXTRACT コマンドがスキーマに基づいてそのポート番号を識別し、それを返します。HTTP の場合の予約済みポート番号は 80 です。 HTTPS の場合の予約済みポート番号は 443 です。
- 照会ストリング全体を取得する場合は QUERYSTRING オプションを使用します。 正しい構文解析の妨げになる可能性のある特定の文字を表すために、照会ストリングは `%xx` というシーケンスが含まれるエスケープ形式で返されます。詳細については、 18 ページの『予約文字と除外文字』を参照してください。 照会ストリングに名前と値のペアとしてフォーム・データが含まれている場合 (例えば、`account=40138025`)、代わりに WEB READ FORMFIELD コマンドを使用して、このデータをアンエスケープされた形式で取得することもできます。 92 ページの『HTTP 要求のフォーム・データの検査』では、コマンドの使用方法について説明しています。
- 非 HTTP 要求を識別する場合は、REQUESTYPE オプションを使用します。

メッセージの HTTP ヘッダーの検査

要求または応答メッセージの各 HTTP ヘッダーは、ヘッダー名およびヘッダー値から構成されています。 CICS は、必要な場合にアプリケーションがアクセスできるように、この情報を保管します。アプリケーションでは、指定されたヘッダーの値を受信したり、要求または応答のヘッダーすべての名前と値を参照したりできます。また、ヘッダーから取得した、設計済みの日時スタンプ・ストリングを ABSTIME 形式に変換することもできます。

このタスクについて

アプリケーションは、要求や応答を処理したり後続のメッセージを構成したりする目的で、ヘッダー内の情報を調べなければならない場合もあります。

- TE ヘッダーは、チャンク化された応答メッセージで末尾ヘッダーが許可されるかどうかをアプリケーションに示します。
- 条件付きヘッダーでは、アプリケーションに対して指示 (応答文書が変更されている場合にのみ応答するなど) を出すことができます。

HTTP 要求または応答の正確な形式が分かっていない限り、アプリケーションは特定のヘッダーの存在に頼ってはなりません。Web クライアントおよびサーバーが送信するヘッダーで不整合が発生する場合があるためです。

HTTP ヘッダーには、日時スタンプが含まれるものがあります。 CICS には、設計された日時スタンプのストリングの共通形式を、アプリケーションで使用する ABSTIME 形式へ変換する CONVERTTIME コマンドが用意されています。

標準の HTTP ヘッダーは HTTP/1.1 仕様書 (RFC 2616) および HTTP/1.0 仕様書 (RFC 1945) で説明されています。 415 ページの『付録 B. CICS Web サポートにおける HTTP ヘッダーの解説』では、CICS Web サポートにおける HTTP ヘッダーの一般的使用方法、および CICS Web サポートによるメッセージで受信した特定のヘッダーに対するアクションについて説明します。CICS はいくつかの HTTP ヘッダーを無視するので、ユーザー・アプリケーションはそれに応じて適切なアクションをとる必要があります。各 HTTP ヘッダーの意味と正しい使用法の詳細なガイダンスおよび要件を、HTTP 仕様で確認してください。

メッセージに末尾ヘッダーが含まれている場合、標準ヘッダーの場合と同様に、**EXEC CICS** WEB コマンドを使用して読み取ることができます。メッセージの Trailer ヘッダーには、末尾ヘッダーとして送信されたすべての HTTP ヘッダーの名前が指定されています。

HTTP ヘッダーの検査と処理は、以下のように行います。

手順

- 特定の HTTP ヘッダーの内容を検査するには、WEB READ HTTPHEADER コマンドを使用します。 ヘッダーの内容を受信するバッファーをアプリケーション・プログラムで用意する必要があります。該当するヘッダーが要求に存在しない場合、 CICS から NOTFND 条件が返されます。
- 要求または応答のすべてのヘッダーを参照するには、次のようにします。
  1. WEB STARTBROWSE HTTPHEADER コマンドを使用して、ヘッダー行の参照を開始します。
  2. WEB READNEXT HTTPHEADER コマンドを使用して、各行のヘッダー名およびヘッダー値を取得します。 アプリケーション・プログラムは、2 つのバッファーを提供する必要があります。1 つはヘッダーの名前を受信し、もう 1 つはその内容を受信します。CICS は、すべてのヘッダーの読み取りが完了すると ENDFILE 状態を返します。
  3. プログラムで関連のヘッダー情報すべてを取得した後、WEB ENDBROWSE HTTPHEADER コマンドを使用して、参照を終了します。

- HTTP ヘッダーに提供されている設計済みの日時スタンプ・ストリングを変換するには、WEB READ HTTPHEADER コマンドを使用してそのストリングをバッファーに受信し、CONVERTTIME コマンドを使用して処理します。 ユーザーが日時スタンプの形式を識別する必要はありません。 CONVERTTIME コマンドによって、インターネットで通常使用される 3 種類の日時スタンプ形式が識別されて変換されます。3 種類とは、RFC 1123 (Web 標準)、RFC 850 (古い形式)、および ASCtime (C 関数からの出力) です。 アプリケーションは、FORMATTIME コマンドを使用して、ABSTIME を他の形式に変換できます。

HTTP 要求に関する技術およびセキュリティー情報の取得

使用中のセキュリティー・オプションを含む、HTTP 要求のための TCP/IP 環境に関する情報、および Web クライアントが提供したクライアント証明書に関する情報をアプリケーションから取得することができます。

このタスクについて

CICS は、Web クライアントとサーバーの間の TCP/IP 接続を管理し、適切なセキュリティー手段を講じ、Web クライアントの ID 認証プロセスを管理します。接続ごとに CICS によって実行されるアクションは、Web クライアントの要求が受信されるポートに対する TCPIPSERVICE 定義で設定したオプションによって決まります。この処理によって取得された情報が要求の処理方法の決定に役立つ場合は、ユーザー作成アプリケーションでその情報を調べることができます。例えば、HTTP 要求を送信した Web クライアントのホスト名および IP アドレスを取得したり、接続のセキュリティーと暗号化のレベルを確認したりできます。

TCP/IP 接続に関する情報、および TCPIPSERVICE 定義で指定されているセキュリティー・オプションに関する情報は、EXTRACT TCPIP コマンドで取得できます。EXTRACT CERTIFICATE コマンドを使用すると SSL (Secure Sockets Layer) ハンドシェーク中に Web クライアントから受信した X.509 クライアント証明書から取得した情報を提供できます。これらのコマンドで使用可能なオプションの詳細な参照情報および説明については、「*CICS アプリケーション・プログラミング・リファレンス*」を参照してください。

手順

- HTTP 要求を送信した Web クライアントのホスト名および IP アドレスを取得するには、EXTRACT TCPIP コマンドを、CLIENTNAME および CLIENTADDR オプションを指定して使用します。 IP アドレスは、2 進数として、またはそのコロン付き 16 進数や小数点付き 10 進数表記が含まれる文字ストリングとして使用可能です。
- アプリケーションが実行されているホスト・システム (つまり、CICS そのもの) のホスト名および IP アドレスを取得するには、SERVERNAME および SERVERADDR オプションを指定した EXTRACT TCPIP コマンドを使用します。 この場合も、IP アドレスは、2 進数として、またはそのコロン付き 16 進数や小数点付き 10 進数表記として使用可能です。
- 要求を受信したポートの番号を取得するには、PORTNUMBER オプションを指定した EXTRACT TCPIP コマンドを使用します。 このポート番号は、2 進数として、または文字ストリングとして使用可能です。または、PORTNUMBER オプションを指定した WEB EXTRACT コマンドを使用することもできます。

- 要求に関連付けられている TCPIPSERVICE リソース定義の名前を取得するには、TCPIPSERVICE オプションを指定した EXTRACT TCPIP コマンドを使用します。
- TCPIPSERVICE 定義で指定された認証のタイプ (基本認証、クライアント証明書認証、または認証を行わない) を識別するには、 AUTHENTICATE オプションを指定した EXTRACT TCPIP コマンドを使用します。 171 ページの『HTTP サーバーとしての CICS: 認証および識別』では、さまざまなタイプの認証について詳しく説明しています。
- TCPIPSERVICE 定義で SSL (Secure Sockets Layer) サポートが指定されているかどうか、および使用されている SSL 暗号化のレベルを識別するには、 SSLTYPE および PRIVACY オプションを指定した EXTRACT TCPIP コマンドを使用します。 184 ページの『CICS Web サポートでの SSL』では、SSL について詳しく説明しています。
- SSL ハンドシェーク中に Web クライアントから受信した X.509 証明書から情報を取得するには、EXTRACT CERTIFICATE コマンドを使用します。 提供された証明書は、CICS により、セキュリティー・マネージャーのデータベースおよびセットアップ可能な証明書失効リストと照合され、すでに検証済みです。 証明書には、その証明書の対象 (所有者またはユーザーと呼ばれることもあります) を識別するフィールド、およびその証明書を発行した認証局 (発行者) を識別するフィールドが含まれています。必要な情報は、OWNER または ISSUER オプションを指定することで選択することができます。また、SERIALNUM および USERID オプションを使用すると、証明書のシリアル番号、およびその証明書に関連付けられている RACF® ユーザー ID を取得することもできます。 *CICS RACF Security Guide*では、証明書の内容およびその使用方法が詳しく説明されています。

HTTP 要求のフォーム・データの検査

フォーム・データは、HTML フォームの要素 (テキスト入力ボックス、ボタン、またはチェック・ボックスなど) とユーザーとの対話によって取得される情報です。情報は一連の名前と値のペアとして送信されます。CICS では、HTTP 要求をスキャンしてフォーム・フィールドを抽出できるため、要求のボディ全体を受信および分析しなくても、 CICS コマンドを使用してアプリケーションでデータを取得できます。

このタスクについて

19 ページの『HTML フォーム』では、フォームおよびフォーム・フィールドについてさらに詳しく説明しています。

アプリケーションは、指定されたフォーム・フィールドの値を受信したり、要求に含まれているすべてのフォーム・フィールドの名前と値を参照したりできます。アプリケーションで使用するためにデータを異なるコード・ページに変換する必要がある場合、コード・ページ変換オプションを指定できます。

Web クライアントは、GET メソッドが使用されている場合は照会ストリングで、また POST メソッドが使用されている場合はメッセージ・ボディで、フォーム・データを送信します。CICS ではこれらのいずれのロケーションからもデータを抽出できるため、どのメソッドが使用されたかを指定する必要がありません。代わりに、

フォーム・データが照会ストリングで送信されている場合、WEB EXTRACT コマンドを使用して照会ストリング全体を取得することもできます。 88 ページの『HTTP 要求の要求行の検査』では、この方法について説明しています。

CICS が HTTP クライアントではなく HTTP サーバーである場合のみ、CICS はフォーム・データを読み取ります。

手順

- HTML フォームの特定のフィールドの値を取得するには、 WEB READ FORMFIELD コマンドを使用します。 アプリケーション・プログラムで、値を受信するバッファーを提供するか、または値のアドレスに対して CICS で設定されるポインターを提供できます。指定した名前のフィールドがフォーム・データにない場合、CICS は NOTFND 状態を返します。 フォーム・データは、返される前に、 CICS によってアンエスケープになり、%xx シーケンスはオリジナルの文字に変換し直された状態となります。詳細については、 18 ページの『予約文字と除外文字』を参照してください。
- フォーム・データのすべてのフィールドを参照するには、次のようにします。
  1. WEB STARTBROWSE FORMFIELD コマンドを使用して、フィールドの参照を開始します。
  2. WEB READNEXT FORMFIELD コマンドを使用して、各フィールドの名前と値を順番に取得します。 アプリケーション・プログラムは、2 つのバッファーを提供します。1 つはフィールドの名前を受信し、もう 1 つはその内容を受信します。すべてのフィールドが読み取られると、 CICS は ENDFILE 状態を返します。
  3. プログラムで関連のフィールドすべてを取得した後、WEB ENDBROWSE FORMFIELD コマンドを使用して、参照を終了します。
- CICS は、受け取ったデータのコード・ページ変換を実行します。 WEB STARTBROWSE FORMFIELD および WEB READ FORMFIELD コマンドで CHARACTERSET および HOSTCODEPAGE オプションを使用すると、Web クライアントおよびアプリケーション・プログラムで使用されるコード・ページを指定できます。
  1. GET メソッドおよび POST メソッドの両方に対してクライアント・アプリケーションで使用される文字エンコードは、HTML フォームの情報によって決定されます。ただし、通常は、送信されるフォーム要求の一部としてこの情報が含まれないため、CHARACTERSET オプションを使用してアプリケーションによって提供されます。この情報は、対応する HTML フォームによって決まるフォーム・エンコード方式と一致しなければなりません。詳細については、 20 ページの『クライアント・エンコード方式の決定方法』を参照してください。
  2. HOSTCODEPAGE オプションでは、アプリケーション・プログラムによって使用される CICS (ホスト) コード・ページを指定します。このコード・ページは通常、EBCDIC コード・ページです。コード・ページが指定されていない場合、 LOCALCCSID システム初期設定パラメーターで指定されている EBCDIC コード・ページで、データが返されます (指定のコード・ページが CICS ウェブ・インターフェースでサポートされている場合)。それ以外の場合、CICS は、データをデフォルトの EBCDIC コード・ページ 037 に返します。

CHARACTERSET オプションおよび HOSTCODEPAGE オプションの詳細については、WEB READ FORMFIELD コマンドおよび WEB STARTBROWSE FORMFIELD コマンドを参照してください。

HTTP 要求のエンティティー・ボディの受信

アプリケーションで WEB RECEIVE コマンドを発行して HTTP 要求のエンティティー・ボディを受信できます。エンティティー・ボディの最初の一部のみを受信することも、一連の WEB RECEIVE コマンドを使用して、エンティティー・ボディ全体を少量ずつ受信することもできます。

このタスクについて

WEB RECEIVE コマンドではタイムアウト値を設定しません。ユーザー・アプリケーションが呼び出されるのは、要求が Web クライアントから完全に受信され、CICS によって保持されているときのみです。HTTP サーバーとしての CICS の場合、ポートに対する TCPIPSERVICE 定義の SOCKETCLOSE 属性によって、Web クライアントからの要求送信の完了までの時間が決定されます。この時間が満了すると、CICS から 408 (Request Timeout) 応答が Web クライアントに返されます。

チャンク転送コーディングを使用して要求メッセージが送信されている場合、CICS は、アプリケーションへの引き渡し前に、チャンクを単一メッセージにアセンブルします。パイプライン処理の一連の要求が送信された場合、CICS では、各要求が別個のトランザクションとして処理され、ユーザー・アプリケーションからの応答が得られてから、次の要求が次のユーザー・アプリケーションで処理できるようになります。

「*CICS アプリケーション・プログラミング・リファレンス*」には、WEB RECEIVE コマンドで使用可能なオプションについての詳細な参照情報と説明が記載されています。以下のアクションを実行するには、WEB RECEIVE コマンドを使用します。

手順

1. この要求のエンティティー・ボディを受信する必要があるかどうかを識別します。
   a. 特定の要求メソッド (GET メソッドなど) の場合、エンティティー・ボディは不適切であり、アプリケーションでは、存在するエンティティー・ボディをすべて無視することが許可されています。 435 ページの『付録 D. CICS Web サポートの HTTP メソッド・リファレンス』では、これに該当するメソッドについて説明しています。不適切なエンティティー・ボディがある場合でも、必要に応じて受信することはできます。 88 ページの『HTTP 要求の要求行の検査』では、要求メソッドの識別方法について説明しています。
   b. HTTP/1.1 要求の場合、エンティティー・ボディの存在は、要求上で、ゼロ以外の Content-Length ヘッダー (メッセージがチャンク化されている場合は Transfer-Encoding ヘッダー) によって示されます。 Content-Length ヘッダーの値がゼロの場合、または Transfer-Encoding ヘッダーも Content-Length ヘッダーも提供されていない場合、エンティティー・ボディは存在しません。89 ページの『メッセージの HTTP ヘッダーの検査』では、メッセージの HTTP ヘッダーの読み取り方法について説明しています。

c. HTTP/1.0 要求では、Content-Length ヘッダーを指定する必要はありませんが、多くの場合は指定します。要求にゼロ以外の Content-Length ヘッダーがある場合は、エンティティー・ボディが存在することを示しています。Content-Length ヘッダーが存在しなくても、エンティティー・ボディが適切であることを要求メソッド (特に POST メソッド) が示していれば、多くの場合にエンティティー・ボディが存在します。

2. INTO オプション (データ・バッファーの場合) または **SET** オプション (ポインター参照の場合) のいずれかと、LENGTH オプションを指定することによって、エンティティー・ボディを受信します。 戻り時、LENGTH オプションは、受信するデータの長さに設定されます。
3. エンティティー・ボディから受信するデータ量を制限する場合、MAXLENGTH オプションを指定します。
   a. エンティティー・ボディの最初の部分のみ受信し、この長さを超えるデータは廃棄する場合は、NOTRUNCATE オプションを省略します。NOTRUNCATE がデフォルトです。
   b. この長さを超えるデータを廃棄せずに保存する場合、NOTRUNCATE オプションを指定します。 残りのデータは、WEB RECEIVE コマンドをさらに使用して取得できます。

   チャンク転送コーディングを使用してデータが送信されている場合、CICS は、アプリケーションへの引き渡し前に、チャンクを単一メッセージにアセンブルします。したがって、MAXLENGTH オプションは、各チャンクに個別に適用されるのではなく、チャンク化されたメッセージのエンティティー・ボディの合計の長さに適用されます。単一のメッセージに対して CICS で受け入れ可能なデータの合計量は、TCPIPSERVICE 定義の MAXDATALEN 属性によって制限します。
4. コード・ページ変換に対して設定するオプションを指定します。
   a. SERVERCONV オプションによって、コード・ページ変換全体が制御されます。このオプションは、コード・ページ変換を実行するかどうかを指定する場合に使用します。 HTTP サーバーとしての CICS の場合、前のリリースでコーディングされた Web 対応アプリケーションとの互換性のため、SERVERCONV が指定されていないが別のコード・ページ変換オプションが指定されている場合、コード・ページ変換を実行すると想定されます。コード・ページ変換を実行しない場合は、SERVERCONV(NOSRVCONVERT) を指定するか、またはすべてのコード・ページ変換オプションを省略します。

      そのメッセージの Content-Encoding のヘッダーで示されているように、 ZIP 化または圧縮されているエンティティー・ボディを受信した場合は、コード・ページ変換を抑止したことを確認してください。 CICS では、このようなタイプのメッセージはデコードされません。したがって、コード・ページ変換が適用された場合、その結果は予測不能です。zip ファイルにされているか圧縮されているエンティティー・ボディを復号できない場合、415 状況コードを返すことによって、Web クライアントに通知できます。
   b. コード・ページ変換を実行したい場合に、CICS で Web クライアントの文字セットが判別できないときは、CHARACTERSET オプションを使用して文字セットを指定します。 古い Web クライアントの場合、要求ヘッダーでこの

情報が提供されていない場合があります。この場合、CICS は ISO-8859-1 文字セットを想定します。この想定が正しくない場合にのみ、文字セットを指定する必要があります。

c. コード・ページ変換を実行する場合に、ローカルの CICS 領域のデフォルト・コード・ページ (LOCALCCSID システム初期設定パラメーターで指定されています) がアプリケーションに適合しないときは、 HOSTCODEPAGE オプションを使用して、代わりのホスト・コード・ページを指定します。

コード・ページ変換は、非テキストのメディア・タイプが指定されているメッセージに対しては実行されません。ただし、SERVERCONV が指定されていない場合を除きます。指定されていない場合、互換性の目的のため、メディア・タイプは考慮されません。互換性の目的のため、CICS では、インバウンド・メッセージでメディア・タイプが指定されていない場合に `application/octet-stream` をデフォルトにする HTTP/1.1 要件が適用されません。代わりに、CICS は `text/plain` をデフォルトとして使用するので、メッセージに対してコード・ページ変換が実行できます。

5. MAXLENGTH および NOTRUNCATE オプションを指定した場合に、さらに受信するデータがあるときは、WEB RECEIVE コマンドをさらに発行します。 **SET** オプションを使用し、MAXLENGTH オプションを使用しない単一の RECEIVE コマンドでは、残りのすべてのデータをその長さに関係なく受信します。別の方法として、NOTRUNCATE オプションを指定した一連の RECEIVE コマンドを使用して、残りのデータを適切なチャンクに分けて受信することもできます。 LENGERR 応答を受信しなくなるまで、RECEIVE コマンドを発行し続けます。 MAXLENGTH オプションで要求された長さより短いデータを受信した場合でも、データの終わりを示しているとは限りません。このような状態は、CICS がデータの終わりで部分文字を返すのを避ける必要がある場合に起こる可能性があります。

応答に対する HTTP ヘッダーの書き込み

アプリケーション・プログラムで作成される動的応答の場合、CICS では、メッセージで使用される HTTP プロトコル・バージョンに応じて、基本的なメッセージに必要な HTTP ヘッダーが自動的に提供されます。アプリケーションでこれらのヘッダーを書き込む必要はありません。しかし、応答にさらに HTTP ヘッダーを追加することが必要になる場合もあります。

このタスクについて

CICS によって作成されるヘッダーの完全なリストを、以下に示します。

- ARM 相関関係子
- Connection
- Content-Type (CICS によって書き込まれるが、複合ヘッダーが必要な場合はクライアント・アプリケーションで提供可能)
- Content-Length
- Date
- Expect
- Host

- サーバー
- TE (CICS によって書き込まれるが、インスタンスを追加することが可能)
- Transfer-Encoding
- User-Agent
- WWW 認証

これらのヘッダーのなかには、CICS が HTTP クライアントである場合にのみ該当し、作成されるものもあります。上記のヘッダーが作成される条件については、415 ページの『付録 B. CICS Web サポートにおける HTTP ヘッダーの解説』で説明されています。応答でこれらのヘッダーを書き込んだ場合、 CICS はこれらのヘッダーを上書きせず、アプリケーションで提供されたバージョンを使用します。

応答の送信時に CICS が用意するヘッダーは、基本メッセージを該当 HTTP プロトコル仕様に準拠させるために通常書き込まれるものです。次のような目的で、応答にさらに HTTP ヘッダーを追加することが必要になる場合もあります。

- キャッシュおよび文書の有効期限の制御 (例えば、Cache-Control、Expires、Last-Modified)
- コンテンツ・ネゴシエーション (例えば、Accept-Ranges、Vary)
- Web クライアントの情報 (例えば、Title、Warning、追加の Content ヘッダー)

アプリケーション・プログラムで複雑なアクションを実行する場合や、応答に対して特定の状況コードを選択する場合は、処理対象の HTTP 仕様で、メッセージに対して特定の HTTP ヘッダーの使用が必要になる可能性があります。応答になんらかの HTTP ヘッダーを追加するときは、それらのヘッダーに適用される重要な要件がないかどうか、処理対象の HTTP 仕様を確認してください。HTTP 仕様についての詳細は、13 ページの『HTTP プロトコル』を参照してください。

メッセージ送信のための WEB SEND コマンドを発行する*前*に、メッセージに対して追加の HTTP ヘッダーを書き込みます。この規則の例外は、チャンク化されたメッセージの末尾ヘッダーとして送信されるヘッダーを書き込む場合です。その場合は、下記のような特別な処理が適用されます。応答の HTTP ヘッダーを書き込むには、次のようにします。

手順

- メッセージに追加するヘッダーごとに WEB WRITE HTTPHEADER コマンドを使用します。 使用している HTTP 仕様に記載されている形式で、各ヘッダーの名前および値が指定されているか確認します新規ヘッダーやユーザー定義ヘッダーを使用する可能性があるため、CICS では HTTP ヘッダーの内容は検証されません。 このコマンドによって単一のヘッダーが追加されます。このコマンドを繰り返すと、ヘッダーをさらに追加することができます。すでに書き込んだヘッダーを再度書き込むと、CICS は、既存のヘッダーに追加して新規のヘッダーを要求または応答に追加します。ヘッダーを再書き込みできるのは、ヘッダーを繰り返せることが HTTP 仕様に記載されている場合のみです。 送信時に、要求に追加する準備ができているヘッダーが CICS で保管されます。
- 使用する HTTP ヘッダーのいずれかが、HTTP/1.1 レベル以下の Web クライアントに対して不適当な可能性がある場合は、これらのヘッダーを書き込む前に、Web クライアントから提供されている HTTP バージョン情報を確認します。 この情報を取得するには、WEB EXTRACT コマンドを使用します。 ユーザー定義

(非標準) ヘッダーの使用を許可するように、CICS では、不適切なユーザー作成ヘッダーは除去されません。HTTP ヘッダーによっては、HTTP/1.1 以下のサーバーでは識別されず、要求の処理時にエラーが発生する場合があります。

CICS には、HTTP/1.0 レベル以下のサーバーまたは Web クライアントに対する対策は特に用意されていません。 CICS ではそれらが HTTP/1.0 レベルであるものとして処理され、HTTP バージョンとして HTTP/1.0 が返されます。

- HTTP ヘッダーのいずれか (例えば、Last-Modified ヘッダー) で使用するために日時スタンプを作成する場合、FORMATTIME コマンドを使用します。 このコマンドの STRINGFORMAT オプションでは、現在の日時 (ABSTIME 形式) またはアプリケーション・プログラムで生成された日時を Web で使用する適切な日時スタンプ形式に変換します。それ以外の日時スタンプ形式は、一部の Web クライアントまたは CICS が通信するサーバーでは受け入れられない場合があります。
- ETag HTTP ヘッダーで使用するストロング・エンティティー・タグを作成する場合は、BIF DIGESTBIF DIGEST コマンドによって作成された SHA-1 ダイジェストを使用できます。 ストロング・エンティティー・タグがあると、クライアントは、If-Match、If-None-Match、または If-Range ヘッダー内のエンティティー・タグを使用して、リソースの条件付き要求を行うことができます。これは、Last-Modified 日時ストリングよりも厳密な、リソース状況の検査方式です。条件付き要求を許可する場合には、アプリケーション・プログラムでサポートを提供する必要があります。CICS では、HTTP GET 要求について If-Match、If-None-Match、または If-Range ファンクションの独自のサポートを行っていません。
- HTTP 要求または応答を送信するためにチャンク転送コーディングを使用している場合や、チャンク化されたメッセージの最後に末尾ヘッダーを組み込む場合は、 102 ページの『チャンク転送コーディングによる HTTP 要求または応答の送信』に説明されている特別な指示に従ってください。 メッセージの最初のチャンクを送信する前に、Trailer ヘッダーを書き込みます。最初のチャンクの WEB SEND コマンドの後に書き込まれたすべての HTTP ヘッダーは、末尾ヘッダーとして処理されます。
- ご使用のアプリケーション・プログラムで、ユーザー作成ヘッダーによって暗黙に指示されるすべてのアクションが実行されることを確認します。 例えば、コンテンツ・ネゴシエーション・ヘッダーを書き込んだ場合、アプリケーション・プログラムでは、そのリソースの異なるバージョンを提供する必要があります。

HTTP メッセージのエンティティー・ボディの作成

Web 対応アプリケーション・プログラムでは、CICS 文書から形成されたエンティティー・ボディを作成することも、データのバッファーからエンティティー・ボディを作成することもできます。

始める前に

HTTP メッセージのエンティティー・ボディとして、 CICS 文書を使用することができます。それらを作成するには、**EXEC CICS** DOCUMENT コマンドを使用します。 アプリケーション・プログラムで直接指定したデータを文書に取り込んだり、文書テンプレートを使用することによって文書にデータを取り込むことができま

す。文書テンプレートは、CICS リソースとして定義された文書の一部、または別の CICS プログラムによって作成された文書の一部です。文書および文書テンプレートは、再利用に備えて保管できます。

代わりに、アプリケーション・プログラムによって作成されるデータのバッファーを指定することもできます。このオプションを使用すると、短いエンティティー・ボディや単純なエンティティー・ボディを、より手軽に作成できる可能性があります。また、メッセージのチャンク転送コーディングには、このオプションを使用する必要があります。ただし、この方法で作成されたデータを、再利用のために保管することはそれほど容易ではありません。

このタスクについて

手順

1. CICS 文書を作成するには、 「*CICS アプリケーション・プログラミング・ガイド*」の指示に従ってください。 文書を作成するには、**EXEC CICS** DOCUMENT アプリケーション・プログラミング・インターフェース (**EXEC CICS** DOCUMENT CREATE、INSERT、および **SET** コマンド) を使用します。完成した文書の文書トークンを指定するには、WEB SEND コマンドの DOCTOKEN オプションを使用します。CICS では、WEB SEND コマンドで指定されたオプションに従って、文書が取得され、適切なコード・ページ変換が実行されます。チャンク化されたメッセージのボディは CICS 文書からは形成できません。
2. あるいは、アプリケーション・プログラムでメッセージ・ボディをアセンブルします。 データのバッファーを指定するには、WEB SEND コマンドの FROM オプションを使用します。 データ・バッファーのサイズには最大限度が設定されていませんが、そのサイズを実際に制限する可能性のある以下の要因を考慮する必要があります。
   - CICS 領域の EDSA 限界。
   - CICS 領域で同時にアセンブルする可能性がある他のメッセージ・ボディの数。 CICS Web サポート・トランザクションに適用されるすべてのトランザクション・クラス定義に対して、MAXACTIVE 設定によってスケジューリング制約が課される場合があります。
   - メッセージ・ボディに使用されるコード・ページ変換のタイプ。EBCDIC コード・ページ 037 から ASCII コード・ページ ISO-8859-1 への変換の場合、CICS では、同一のデータ・バッファーが上書きされ、追加ストレージは使用されません。その他のタイプのコード・ページ変換の場合、 CICS では、変換されたメッセージ・ボディを格納する追加のストレージが必要です。使用される文字セットに応じて、この追加ストレージ域のサイズの範囲は、オリジナルのメッセージ・ボディと同じサイズから、理論上の最大値であるオリジナルのメッセージ・ボディの 4 倍のサイズ (非常に稀な場合) までになります。例えば、FROM オプションを使用して送信されるデータの 2 MB バッファーは、少なくとも合計 4 MB のストレージを必要とします。2 バイト文字セット (DBCS) またはマルチバイト文字セットでは、通常、この範囲内で、より大容量のストレージ域が必要です。

HTTP サーバーとしての CICS からの HTTP 応答の送信

WEB SEND コマンドを使用して、CICS で HTTP 応答の HTTP ヘッダー、エンティティー・ボディ、状況コードおよび理由句をアセンブルし、コード・ページ変換を実行し、Web クライアントへ応答を送信します。

始める前に

WEB WRITE HTTPHEADER コマンドを使用して、応答の追加 HTTP ヘッダーを書き込みます。これについては、 96 ページの『応答に対する HTTP ヘッダーの書き込み』で説明しています。また、メッセージに必要なエンティティー・ボディを作成します。これについては、 98 ページの『HTTP メッセージのエンティティー・ボディの作成』で説明しています。

WEB SEND コマンドで状況コードおよび理由句を指定します。これらについては、 17 ページの『状況コードおよび理由句』で説明しています。 423 ページの『付録 C. CICS Web サポートの HTTP 状況コード・リファレンス』では、アプリケーションで使用する状況コードの概要について説明してます。状況コードの使用を計画するため、詳細情報を調べるには、作業対象の HTTP 仕様を参照する必要があります。HTTP 仕様についての詳細は、 13 ページの『HTTP プロトコル』を参照してください。

このタスクについて

必要に応じて、応答はチャンクで送信できます (チャンク転送コーディング)。パイプライン処理の応答を Web クライアントに送信することはできません (Web クライアントから送信された各要求ごとに、単一の応答を送信する必要があります)。

「*CICS アプリケーション・プログラミング・リファレンス*」には、WEB SEND コマンドで使用可能なオプションについての詳細な参照情報と説明が記載されています。

コマンドに関する以下の点に注意してください。

手順

1. STATUSCODE オプションを指定して、応答に対する適切な状況コードを選択し、状態に応じて、STATUSTEXT および STATUSLEN オプションを指定して、理由句を提供します。 CICS は選択された状況コードを検証しないので、値が有効であることと HTTP 状況コードの規則に準拠していることを、ユーザー・アプリケーションが確認する必要があります。 選択した状況コードに応じて、WEB SEND コマンドを発行する前に、次のステップの一部またはすべてを実行する必要があります。
   a. Web クライアントの要求の HTTP バージョンを確認して、状況コードが識別されるかどうかを確認します。 HTTP/1.1 仕様では、HTTP/1.0 仕様よりも多くの状況コードが使用されます。
   b. HTTP 仕様で、状況コードには特定の HTTP ヘッダーが付属する必要があると示されている場合、WRITE HTTPHEADER コマンドを使用して、これらのヘッダーを作成します。
   c. HTTP 仕様で、状況コードには特殊情報を提供するメッセージ・ボディが付属する必要があると示されている場合、適切なエンティティー・ボディを作

成します。 特殊情報が必要になるのは通常、状況コードがエラーを示しているか、クライアントにさらにアクションを要求しているときです。 状況コード 204、205、および 304 ではメッセージ・ボディは許可されていません。メッセージ・ボディが許可されていない状況コードをユーザーが選択して、しかもメッセージ・ボディを使用しようとすると、CICS は WEB SEND コマンドに対してエラー応答を返します。

2. 応答のエンティティー・ボディのソースを識別します。作成した CICS 文書の場合は DOCTOKEN オプションを指定し、アセンブルしたデータのボディの場合は FROM オプションを指定することにより、ソースを識別します。FROM オプションを使用する場合、エンティティー・ボディの長さ、またはチャンク転送コーディングが使用されている場合はチャンクの長さを指定するために FROMLENGTH オプションを指定します。 チャンク転送コーディングの場合、DOCTOKEN オプションは使用できません。
3. MEDIATYPE オプションを使用して、応答のボディのメディア・タイプを指定します。 CICS では、データ内容に対して仕様の妥当性は検査されません。MEDIATYPE にはデフォルトがありません。このオプションを指定しない場合、CICS は、応答に対して Content-Type ヘッダーを作成しません。
4. メッセージをタスク終了時 (デフォルト) ではなく即時に送信する場合、ACTION オプションに IMMEDIATE を指定します。 チャンク転送コーディングを使用する場合は IMMEDIATE がデフォルトなので、この選択を行う必要はありません。

   1 つのタスクで送信できる応答は 1 つのみです。この応答は、1 つの WEB SEND コマンドを使用した標準応答、または一連の WEB SEND コマンドを使用したチャンク化された応答になります。
5. 応答を送信した後、接続を閉じる場合、CONNECTION オプションに CLOSE を指定します。 CICS は、応答に Connection: close ヘッダーを書き込みます。これによって、接続が閉じられたこと、および要求をこれ以上送信できないことが、Web クライアントに通知されます HTTP/1.0 レベルの Web クライアントの場合、CICS では、 Connection: Keep-Alive ヘッダーを省略することによって、同様の通知が行われます。
6. メッセージ・ボディのコード・ページ変換の設定を適切に指定します。
   a. SERVERCONV オプションによって、コード・ページ変換全体が制御されます。このオプションは、コード・ページ変換を実行するかどうかを指定する場合に使用します。 HTTP サーバーとしての CICS の場合、以前のリリースでコーディングされた Web 対応アプリケーションとの互換性のために、SERVERCONN ではなく別のコード・ページ変換オプションを指定した場合は、コード・ページ変換を実行すると見なされます。コード・ページ変換を実行しない場合は、SERVERCONV(NOSRVCONVERT) を指定するか、またはすべてのコード・ページ変換オプションを省略します。
   b. コード・ページ変換を実行したいが、CICS で選択されている文字セットが適切ではない場合、CHARACTERSET オプションを使用して、代わりの文字セットを指定します。 デフォルトでは、Web クライアントからのもともとの要求の Content-Type ヘッダーで指定された文字セットが、CICS で使用されます。その文字セットがサポートされないかまたは指定されていない場合、CICS では ISO-8859-1 文字セットが使用されます。

Web クライアントが、Accept-Charset ヘッダーに、代わりの許容文字セットを指定する場合があります。これらのいずれかを指定する場合、アプリケーションはヘッダー (Web クライアント設定を示す品質値を含んでいることがある) を分析して、サポートされる適切な文字セットを選択する必要があります。 CICS は IANA で指定された文字セットをすべてサポートするわけではありません。 413 ページの『付録 A. HTML コード化文字セット』では、コード・ページ変換用に CICS でサポートされている IANA 文字セットがリストされています。

c. コード・ページ変換が必要であり、FROM オプションを使用してメッセージ・ボディを指定する場合は、HOSTCODEPAGE オプションを使用してアプリケーションのコード・ページを示します (これが、LOCALCCSID システム初期設定パラメーターで指定されているローカル CICS 領域のデフォルト・コード・ページではない 場合)。 CICS 文書 (DOCTOKEN オプション) を使用する場合、 CICS では、文書のホスト・コード・ページの CICS 文書ドメインのレコードからホスト・コード・ページを識別します。

コード・ページ変換は、非テキストのメディア・タイプが指定されているメッセージに対しては実行されません。ただし、SERVERCONV が指定されていない場合を除きます。指定されていない場合、互換性の目的のため、メディア・タイプは考慮されません。 HTTP ヘッダーおよび状況表示行は、CICS によって常時、ISO-8859-1 文字セットに変換されます。

7. チャンク転送コーディング (またはチャンク化) を使用する場合、このトピックの基本的な指示に加えて、『チャンク転送コーディングによる HTTP 要求または応答の送信』 に説明されている特別な指示に従ってください。 チャンク化されたメッセージが受信側で受け入れ可能になるように、そのトピックで説明されている手順に確実に従う必要があります。チャンク化されたメッセージは、特定のオプションを指定した、WEB SEND コマンドの複数のインスタンスを使用して送信されます。

チャンク転送コーディングによる HTTP 要求または応答の送信

HTTP クライアントとしての CICS による HTTP 要求、または HTTP サーバーとしての CICS からの HTTP 応答のためのチャンク転送コーディングをセットアップできます。

始める前に

まず、送信する項目の以下の属性を検討します。

- メッセージの先頭で使用される HTTP ヘッダー。CICS は、その通常のメッセージ・ヘッダーを提供しています。これらのヘッダーは、 415 ページの『付録 B. CICS Web サポートにおける HTTP ヘッダーの解説』にリストしてあります。チャンク化済みメッセージについては、 CICS では、チャンク・ヘッダー Transfer-Encoding: を含め、チャンク転送コーディングに適したヘッダーを提供します。メッセージの先頭に追加のヘッダーが必要な場合は、最初の WEB SEND コマンドの前に、アプリケーションでそれらのヘッダーを書き込むことができます。

- メッセージの最後にトレーラーで送信されるヘッダー。これらのヘッダーは末尾ヘッダーと呼ばれます。HTTP/1.1 仕様には、受信側が末尾ヘッダーを無視しても問題ないなど、末尾ヘッダーの使用に関する要件が定められています。
- メッセージの分割方法。アプリケーション・プログラムにとって最も都合の良い方法で分割します。例えば、数多くの他のアプリケーション・プログラムからの出力を、それが作成されたときのまま送信したり、あるいはテーブルの各行のデータを個別に読み取って送信したりできます。
- 送信されるデータの各チャンクの長さ。末尾ヘッダーの長さは含めないでください。

このタスクについて

この手順を使用して、HTTP/1.1 仕様で定義されているとおりに正しく構成されたチャンク化済みメッセージを作成します。詳細については、 13 ページの『HTTP プロトコル』を参照してください。 チャンク化済みメッセージが正しく構成されていない場合、受信側はそれを破棄することができます。

100 ページの『HTTP サーバーとしての CICS からの HTTP 応答の送信』は、サーバー応答を送信するためのアプリケーション・プログラムを作成するための主な指示のセットであり、 141 ページの『HTTP クライアントとしての CICS を介した HTTP 要求』は、クライアント要求を行うためのアプリケーション・プログラムを作成するための主な指示のセットです。現在のトピックにある指示を、それらの指示セットのいずれかと併用することができます。

チャンク化済みメッセージのボディを CICS 文書から直接形成することはできません。したがって、DOCTOKEN オプションは使用できません。 FROM オプションを使用して、チャンク化済みメッセージのボディを形成するデータを指定する必要があります。

チャンク化済みメッセージの一部分の送信をすでに開始している場合は、最後の空のチャンクを送信してチャンク化済みメッセージが完了するまで、別のメッセージを送信したり、項目を受信したりすることはできません。

手順

1. チャンク化済みメッセージを開始する前に、Web クライアントまたはサーバーが HTTP/1.1 バージョンであることを確認してください。 すべての HTTP/1.1 アプリケーションは、チャンク転送コーディングを扱わなければなりません。チャンク化済みメッセージを HTTP/1.0 受信側に送信することはできません。
   a. HTTP サーバーとしての CICS が送信した応答については、WEB EXTRACT コマンドを使用して、Web クライアントの要求に対して指定されている HTTP バージョンをチェックします。
   b. HTTP クライアントとしての CICS によって送信された要求の場合、接続を行うための Web OPEN コマンドで HTTPVNUM および HTTPRNUM オプションを指定していた場合は、このコマンドで、サーバーの HTTP バージョンが返されます。これらのオプションを指定しなかった場合は、WEB EXTRACT コマンドを使用してサーバーの HTTP バージョンをチェックします。

c. 代わりに、このチェックを省略して、メッセージの最初のチャンクを送信するために WEB SEND コマンドを発行する際に、CICS が Web クライアントまたはサーバーのバージョンをチェックすることを許可することもできます。 受信側が HTTP/1.0 の場合は、エラー応答を受け取ります。

2. メッセージのボディの*前* に送信する必要のある HTTP ヘッダーを書き込むため、必要な回数だけ WRITE HTTPHEADER コマンドを使用します。 チャンク転送コーディングのヘッダーは書き込まないでください。これらのヘッダーは、CICS 自身が、アプリケーション・プログラムから提供されたチャンクの長さについての情報を使用して書き込みます。

3. チャンク化済みメッセージに末尾ヘッダー (メッセージのボディの後 に送信されるヘッダー) を組み込みたい場合は、WRITE HTTPHEADER コマンドを使用して Trailer ヘッダーを書き込みます。トレーラーで送信する予定のすべての HTTP ヘッダーの名前を、Trailer ヘッダーの値として指定します。 Transfer-Encoding、Trailer、および Content-Length ヘッダーを除く任意のヘッダーを、末尾ヘッダーとして送信することができます。

   a. HTTP サーバーとしての CICS によって送信される応答の場合は、Web クライアントがその要求で TE: トレーラー・ヘッダーを送信したことを確認します。 このヘッダーは、クライアントが末尾ヘッダーを扱うことを示しています。 クライアントが TE: トレーラーを送信しなかったときに、Trailer ヘッダーを書き込もうとすると、CICS は RESP2 の値を 6 にした INVREQ 応答を WRITE HTTPHEADER コマンドに返します。代わりに、READ HTTPHEADER コマンドを使用して、TE: トレーラー・ヘッダーの有無を確認することもできます。

   b. HTTP クライアントとしての CICS によって送信される要求については、TE ヘッダーへの参照を含めずに末尾ヘッダーを組み込むことができます。

   末尾ヘッダーそのものは、チャンク化送信プロセス中に書き込まれます。

4. WEB SEND コマンドを使用して、メッセージの最初のチャンクを送信します。

   a. CHUNKING(CHUNKYES) を指定して、これがメッセージのチャンクであることを CICS に知らせます。

   b. FROM オプションを使用して、メッセージのボディのデータの最初のチャンクを指定します。

   c. FROMLENGTH オプションを使用して、チャンクの長さを指定します。

   d. HTTP クライアントとしての CICS による要求の場合は、適切なメソッドを METHOD オプションで指定します。チャンク転送コーディングは、メッセージのボディを含まない要求には関係しないため、 GET、HEAD、DELETE、OPTIONS、および TRACE メソッドには関連しませんが、POST および PUT メソッドには使用することができます。

   e. ご使用の主な指示セットに提供したとおり、チャンク化済みメッセージおよび非チャンク・メッセージの両方に適用する、その他のオプションを指定します。

5. WEB SEND コマンドを必要な回数使用して、メッセージの残りの各チャンクを送信します。各 WEB SEND コマンドで、以下の項目を指定します。

   a. CHUNKING(CHUNKYES)

   b. FROM。データのチャンクを指定します。

   c. FROMLENGTH。チャンクの長さを指定します。

このコマンドには、他のオプションは指定しないでください。CICS は、このコマンドを発行すると、各チャンクを送信します。

6. オプション: 最初のチャンクに対して WEB SEND コマンドを発行した後は、最後の空のチャンクに対して WEB SEND コマンドを発行する前であればいつでも (次のステップを参照)、WRITE HTTPHEADER コマンドを使用して、末尾ヘッダーとして送信する追加の HTTP ヘッダーを作成することができます。Trailer ヘッダーがメッセージの最初のチャンクに書き込まれていた場合、チャンク送信プロセスで書き込まれた HTTP ヘッダーは、CICS によって末尾ヘッダーとして扱われ、最後の空のチャンクとともに送信されます (Trailer ヘッダーが書き込まれていなかった場合、CICS は末尾ヘッダーの書き込みを許可しません)。CICS は、トレーラー・ヘッダーが、メッセージの最初のチャンクで初期 Trailer ヘッダーで指定した名前に一致しているかどうかはチェックしません。
7. データの最後のチャンクの送信が完了したら、CHUNKING(CHUNKEND) オプションを指定し、FROM または FROMLENGTH オプションは指定しないで、さらに WEB SEND コマンドを指定します。 CICS は、空のチャンクを生成し、それを受信側に送信し、チャンク化済みメッセージを終了します。空のチャンクは、書き込んだ末尾ヘッダーを含むトレーラーとともに送信されます。
8. HTTP サーバーとしての CICS の場合、エラーは以下のように処理されます。
   a. WEB SEND コマンドのいずれかがシーケンス中に失敗した場合、エラー応答が返され、それ以降の送信も失敗します。この状況はアプリケーションで適切に処理する必要があります。
   b. チャンクがすべて正常に送信された場合でも、アプリケーションが最後の WEB SEND コマンドを、CHUNKING(CHUNKEND) を指定して発行しなかった場合、トランザクションは異常終了コード AWBP を出して異常終了します。 この異常終了が必要なのは、チャンク化済みメッセージが完全で正しいことを CICS は保証できず、アプリケーションに代わって最後の空のチャンクを発行することができないからです。

   不完全なチャンク化済みメッセージは、受信側では無視されて廃棄されます。Web クライアントは、要求を再試行するかどうかを判断します。
9. HTTP クライアントとしての CICS の場合、エラーは以下のように処理されます。
   a. アプリケーション・プログラムが、チャンク転送コーディング・プロセスでエラーを報告された場合は、WEB CLOSE コマンドを使用してプロセスを停止し、接続を閉じます。 サーバーは、最後の空のチャンクを受信しないため、これまで送信されたデータを無視して破棄します。要求を再試行するかしないかはユーザーが決定できます。
   b. 最後の空のチャンクを送信しなかった場合、または WEB CLOSE コマンドを発行しなかった場合は、タスク終了時に CICS Web サポート・メッセージ用の一時データ・キューである CWBO に警告メッセージが書き込まれます。 サーバーは受信をタイムアウトにし、送信されたデータを無視して破棄します。

HTTP 要求シーケンス中のアプリケーション状態の管理

CICS では、 Web クライアントによって作成される要求ごとに、新規の別名トランザクションおよび新規のプログラムが開始されます。この開始は、パイプライン化された要求、持続接続を使用して行われた要求、論理シーケンスを形成する要求の場合、および個別のスタンドアロン要求の場合です。要求間でアプリケーション状態をどのように管理するかを考慮に入れてください。

このタスクについて

要求シーケンスの間、別々のプログラム間または同じプログラムの別々のインスタンス間でデータを共用するには、CICS 管理リソースを使用するか、または Web クライアントによって送信された要求の要素を使用します。

タスクを正常に完了するために、Web クライアントと CICS の間で要求と応答を複数回交換する必要がある場合は、Web クライアントがシーケンス内の新しいステップの 1 つ 1 つを開始します。 CICS から送信される応答を、Web クライアント、および Web クライアントのユーザーを次のステップに導くように設計できます。例えば、ユーザーが次の要求を作成するのに使用できるリンクやボタンなどのコントロールを、エンティティー・ボディに含めることができます。ただし、適切な要求シーケンスを選択させることは容易ではありません。特に、次のような理由で、計画されたシーケンスがくずれることがあります。

- クライアントが Web ブラウザーであり、ユーザーが前の応答で提供された HTML ページのコントロールを選択せずに、特定の要求を開始するために既知の URL を入力した場合。
- ユーザーが Web クライアントをシャットダウンしたり、Web クライアントで代替アクティビティーに変更したりして、アクティビティーを中止した場合。

また、ユーザーがシーケンスの要求の開始を遅らせる場合もあります。

要求シーケンスの遅延または中止を処理できるように、アプリケーション・プログラムを設計する必要があります。例えば、要求シーケンスの間データを共用するなら、要求シーケンスが完了しなかったり過度に遅延されたりした場合に、データが確実にクリーンアップされるようにする必要があります。アプリケーション・プログラムが保護リソースを更新する場合は、一緒にコミットまたはバックアウトする必要がある更新が、同じトランザクション内で行われるようにする必要があります。したがって、Web クライアントからの単一の要求で更新を完了するように設計する必要があります。

アプリケーションにとって最適な状態は、要求および応答の交換ごとに、必要なものを完備していて、タスクの独立要素が完了する状態です。ただし、このような設計は、特にタスクが複雑な場合や、パイプライン処理される要求シーケンスが Web クライアントから送信される場合など、常に可能とは限りません。要求間でアプリケーション状態を管理する必要がある疑似会話型モデルが必要な場合もあります。以下の技法を使用します。

手順

- アプリケーションの状態または共用データが要求に取り込まれるように、Web クライアントから送信される要求を設計できます (例えば、Web クライアントから

HTML フォームが送信されるときに使用される要求 URL の一部として組み込みます)。 その次のプログラムで、共用データを取得する要求 URL を検査できます。

- Web クライアントに応答として返される HTML フォーム内の隠しフィールドを使用して、アプリケーション状態を少量保管できます。 計画されたシーケンス内の次のアクションをユーザーが実行したとき、CICS に送信される要求に、隠しフィールドを組み込むことができます。次のアプリケーション・プログラムは、この隠しフィールドを見つけて読み取ることができます。
- より大量にある状態の場合、および存続時間が延長される状態の場合は、CICS 管理リソースを作成して、アプリケーションの状態を保持し、リソースを表すトークンを受け渡すことができます。 CICS には、アプリケーション状態を主記憶域または一時記憶域のキューに保管し、アプリケーション・プログラムが情報にアクセスするために使用できるトークンを提供する、サンプルの状態管理プログラム (DFH$WBST および DFH$WBSR) が提供されています。 トークンは、疑似会話中のプログラム間で HTML フォームの隠しフィールドとして、または対話間で URL の照会ストリングとして伝達できます。 この技法は、疑似会話の間中情報を保持するために使用するほか、ユーザーとさまざまな CICS アプリケーション・プログラムの拡張対話の間中 (おそらく複数の疑似会話にわたって) 情報を保持するためにも使用します。

第 7 章 HTTP サーバーとしての CICS のリソースの定義

実行する CICS Web サポート・タスクごとに、追加のリソース定義をいくつか作成します。

このタスクについて

CICS 提供のリソース定義グループである DFHWEB には、以下の CICS Web サポート・リソースが含まれています。

- CICS Web サポート・タスクのトランザクション定義 (例えば、CWBA および CWXN など)
- CICS Web サポート・ユーティリティー・プログラム
  - デフォルトのアナライザー・プログラム DFHWBAAX、およびサンプルのアナライザー・プログラム DFHWBADX
  - Web エラー・プログラムの DFHWBEP および DFHWBERX
- 一時記憶域モデル DFHWEB

CICS Web サポート・メッセージの一時データ・キューである CWBO (大半のメッセージ用) および CWBW (警告ヘッダー・メッセージの個別キュー) は、グループ DFHDCTG の中にあります。

このリソース定義グループ DFH$WEB には、サンプル CICS Web サポート・アプリケーションのほとんどの PROGRAM リソース定義と URIMAP 定義が含まれています。

71 ページの『第 5 章 HTTP サーバーとしての CICS に対して CICS Web サポートを使用可能にする』は、タスクごとに必要なリソース定義に関する手引きとなります。詳細は以下のトピックに記載しています。

CICS Web サポートの TCPIPSERVICE リソース定義の作成

ポートと CICS サービス (CICS Web サポートを含む) 間の関連を定義するには、TCPIPSERVICE リソース定義を使用します。CICS Web サポートに使用するポートごとに、TCPIPSERVICE リソース定義を定義し、インストールします。

このタスクについて

CICS システムでアクティブな各 TCPIPSERVICE 定義では、固有のポート番号を指定する必要があります。CICS は、ポートの TCPIPSERVICE 定義を使用して、そのポートでインバウンド TCP/IP 接続要求を受け取ったときにどの CICS サービスを呼び出すかを判別します。サービスの識別には、PROTOCOL 属性を使用します。標準の CICS Web サポートについては HTTP を指定し、CICS Web サポートを使用して処理される非 HTTP 要求については USER を指定します。

CICS Web サポートの場合、インターネット・サービスに使用されるデフォルト (つまり予約済み) のポート番号の TCPIPSERVICE 定義を作成します。HTTP のデ

フォルトのポート番号は 80、HTTPS のデフォルトのポート番号は 443 です。非標準のポート番号を使用することもできます。

各 TCPIPSERVICE 定義では、Web 接続タスクについて 1 つのアナライザー・プログラム、および 1 つのトランザクション定義のみを指定できます。これらの項目を複数使用する必要がある場合は、別の TCPIPSERVICE 定義 (したがって別のポート) を使用する必要があります。

CICS では、グループ DFH$SOT で CICS Web サポート用のサンプル TCPIPSERVICE 定義が用意されています。

HTTPNSSL
SSL サポートのない CICS Web TCPIPSERVICE

HTTPSSL
SSL サポートのある CICS Web TCPIPSERVICE

重要: 各ポートに適用されるセキュリティー手段を指定するには、TCPIPSERVICE リソース定義を使用します。SSL を使用するかどうかを選択できます。SSL を使用する場合は、適用される正確なセキュリティー手段 (認証方式、クライアントおよびサーバーによる証明書の送信、メッセージの暗号化など) を選択します。CICS Web サポート機能を安全に保つために使用できるセキュリティー機能の詳細については、171 ページの『第 11 章 CICS Web サポートのセキュリティー』を参照してください。

手順

1. CICS Web サポートに使用する TCP/IP ポートを指定します。 CICS Web サポートで使用するポート番号は予約しておくことをお勧めします。ポートの使用法については、66 ページの『CICS Web サポート用のポートの予約』を参照してください。
2. TCPIPSERVICE リソースを作成します。 このポートのインバウンド HTTP 要求に対する URIMAP 定義を設定する場合は、 TCPIPSERVICE 定義の名前を指定します。
3. STATUS 属性を使用して、定義をインストールした直後に CICS がこのサービスの listen を開始するかどうかを指定します。 CLOSED を指定する場合は、使用する前にサービスをオープン状態に設定する必要があります。 CEMT トランザクションまたは **SET** TCPIPSERVICE システム・プログラミング・コマンドを使用して、サービスをオープン状態またはクローズ状態に設定できます。
4. この定義によって扱われる TCP/IP ポート番号として PORTNUMBER 属性を指定します。
5. HOST 属性を使用して、TCPIPSERVICE が着信接続を listen するドット 10 進またはコロン 16 進 IP アドレスを指定します。IPADDRESS 属性を使用して、既存プログラムのドット 10 進 IP アドレスを指定することもできます。または、複数の IP スタックを持つ構成の場合は、INADDR_ANY を指定して、 CICS が定義されているすべてのスタック上のポートに結合を試みるように設定することもできます。あるいは、複数スタック CINET 環境で、デフォルトの TCP/IP スタックにのみ親和性を割り当てる場合は、DEFAULT を指定してこれを行うこともできます。 この TCPIPSERVICE リソース定義属性に関する参照情報には、追加の詳しい考慮事項が記載されています。複数の CICS

領域でこの TCPIPSERVICE 定義を共用させる場合、または複数の CICS 領域を CICS が指定するポート番号に結合する場合に、この参照情報が重要になります。

6. DNSGROUP 属性および GRPCRITICAL 属性を使用して、sysplex ドメイン内でサービスが使用する DNS グループ名、およびサービスのクリティカル状況を指定します。 この情報を使用して、CICS を Workload Manager for DNS の接続の最適化に登録できます。この領域の詳細については、*Java Applications in CICS* を参照してください。
7. PROTOCOL 属性を使用して、CICS Web サポートがこのポート上の要求を処理することを指定します。
   a. 通常の HTTP 要求には HTTP を指定します。 ポート 80 または 443 を指定すると、 CICS によって HTTP が強制されます。このオプションは、SSL のある HTTP および SSL のない HTTP の両方をカバーします。SSL オプションは SSL を含めるかどうかを指定します。
   b. CICS Web サポートを使用して処理される非 HTTP 要求には USER を指定します。 USER を指定すると、要求の処理には CICS Web サポートが使用されますが、このプロトコルを使用して送受信されるメッセージの受け取りチェックは行われません。要求には非 HTTP というフラグが立てられ、直接アナライザー・プログラムに渡されます。 これらの要求には URIMAP 定義は使用されません。
8. TRANSACTION 属性で Web 接続タスクの 4 文字の ID を指定します。これは、通常、HTTP 要求では CWXN、非 HTTP (USER プロトコル) 要求では CWXU です。 このタスクは、要求の初期処理を取り扱います。ポート 80 または 443 を指定すると、 CICS はデフォルトとして CWXN を提供します。アカウンティングおよびモニターを行うために必要な場合は、 CWXN または CWXU の別名を指定できます。どちらの場合も、プログラム DFHWBXN を実行する必要があります。
9. URM 属性を、この TCPIPSERVICE 定義に関連したアナライザー・プログラムの名前として指定します。 非 HTTP (USER プロトコル) 要求の場合、常にこのアナライザー・プログラムが使用されます。 HTTP 要求の場合、URIMAP 定義でアナライザー・プログラムの使用を指定するか、 URIMAP 定義が存在しないときに、要求を解釈するためにこのアナライザー・プログラムが使用されます。アナライザー・プログラムはユーザーが指定する必要があります。各 TCPIPSERVICE 定義について選択できるアナライザー・プログラムは 1 つだけですが、アナライザー・プログラムは任意の要求を処理するようにコーディングすることができます。 URIMAP 定義を使用してすべての HTTP 要求を処理する場合、 441 ページの『付録 E. アナライザー・プログラム』 によって、アナライザー・プログラムが提供しなければならない基本サポートが分かります。 71 ページの『第 5 章 HTTP サーバーとしての CICS に対して CICS Web サポートを使用可能にする』のアーキテクチャー・ガイダンスによって、特定の HTTP 要求でアナライザー・プログラムを使用する必要があるかどうかを判別することができます。
10. SOCKETCLOSE 属性を使用して、ソケットで着信データの受信を発行した後ソケットを閉じるまで、 CICS が待機する時間を指定します。 NO は、データを受信するまで、または Web クライアントがソケットを閉じるまで、ソケットは開かれたままであることを意味します。低速または障害発生中の Web クライアントによってソケットがブロックされるのを防ぐには、 NO を指定する代

わりにタイムアウト値を指定します。接続が確立された後に Web 接続タスクによって発行される最初の受信コマンドではこのタイムアウト値は無視され、タスクは、CICS によって決定される期間 (HTTP の場合は 30 秒) Web クライアントからのデータ受信を待ちます。この遅延により、データが即時に使用可能でない場合でも、ソケット接続が開始直後に閉じられることがなくなり、Web クライアント側での接続リセット・エラーの発生も防止されます。

注: CICS Web サポートの場合、SOCKETCLOSE にゼロを設定するということは、たとえ Web クライアントが要求したとしても、持続接続を維持できなくなるということになります。この設定は、HTTP/1.1 仕様に準拠していません。HTTP プロトコル環境で SOCKETCLOSE(0) を指定するのは、現在外部要求を処理していない CICS 領域 (例えばテスト環境) でそれを特に必要とする場合だけにしてください。

11. BACKLOG 属性を使用して、TCP/IP が Web クライアントからの着信要求のリジェクトを開始するまでに、キューに入れることのできる接続数を指定します。 デフォルトは、1 です。
12. このポートに対して CICS 領域が常に許可する Web クライアントからの持続接続の数の制限を指定する必要がある場合は、MAXPERSIST 属性を使用します。 デフォルトでは、制限がありません。これは、HTTP/1.1 サーバーの標準動作です。 HTTP サーバーとしての CICS に長期持続接続が原因でパフォーマンス上の問題が起こった領域にのみ、制限を指定してください。制限に達すると、CICS は接続スロットリングを実施します。 46 ページの『CICS Web サポートによる持続接続の処理方法』で、制限を指定するとどうなるかを説明しています。

    注: MAXPERSIST にゼロを設定することは、Web クライアントに持続接続を許可しないことを意味します。この設定は、HTTP/1.1 仕様に準拠していません。 MAXPERSIST(0) を指定するのは、現在外部要求を処理していない CICS 領域 (例えばテスト環境) でそれを特に必要とする場合だけにしてください。
13. MAXDATALEN 属性を使用して、この接続で受信できるデータの最大長を指定します。 デフォルト値は 32 KB で、最大値は 524,288 KB です。このオプションによって、大量のデータ送信によるサービス妨害アタックを防ぐことができます。
14. SSL 属性を使用して、このポートで Secure Sockets Layer (SSL) を使用するかどうかを指定します。 YES を指定すると、SSL が使用され、CICS は Web クライアントにサーバー証明書を送信します。CLIENTAUTH を指定すると、SSL が使用され、 Web クライアントはクライアント証明書を CICS に送信し、さらに CICS は Web クライアントにサーバー証明書を送信します。ポート番号 443 を使用すると、 CICS はデフォルトとして YES を提供し、ポート番号 80 を指定すると NO を強制します。 SSL を使用するときに行うべきことについては、 171 ページの『第 11 章 CICS Web サポートのセキュリティー』を参照してください。
15. SSL(YES) または SSL(CLIENTAUTH) を指定した場合、CERTIFICATE 属性を使用して、 SSL ハンドシェークの際に CICS がサーバー証明書として使用する X.509 証明書のラベルを指定します。 この属性が省略されている場合、CICS 領域ユーザー ID の鍵リングで定義されているデフォルトの証明書が使用されます。証明書は、外部セキュリティー・マネージャーのデータベースの鍵

リングに保管する必要があります。証明書の使用の詳細については、 171 ページの『第 11 章 CICS Web サポートのセキュリティー』を参照してください。

16. AUTHENTICATE 属性を使用して、このポートで要求を出す Web クライアントに使用する認証レベルを指定します。 認証および識別については、 171 ページの『第 11 章 CICS Web サポートのセキュリティー』を参照してください。
    a. Web クライアントに認証または識別情報を送信するよう要求しない場合は NO を指定します。 クライアントが、セキュリティー・マネージャーに登録済みの有効な証明書を送信すると、 CICS はそれを使用することができます。
    b. CICS に HTTP 基本認証を行わせるには BASIC を指定します。この場合、CICS は Web クライアントからユーザー ID およびパスワードを要求します。 基本認証について詳しくは 22 ページの『HTTP 基本認証』を参照してください。
    c. SSL クライアント証明書認証を使用するには CERTIFICATE を指定します。 Web クライアントは、セキュリティー・マネージャーに登録済みで、ユーザー ID に関連付けられている有効な証明書を送信する必要があります。有効な証明書を受信しなかったか、または証明書がユーザー ID に関連付けられていない場合、接続はリジェクトされます。 このオプションを使用する場合は、SSL(CLIENTAUTH) を指定する必要があります。
    d. セキュリティー・マネージャーへの自動登録機能を持つ SSL クライアント証明書認証を使用するには、 AUTOREGISTER を指定します。 Web クライアントは有効な証明書を送信する必要があります。 CICS が証明書がまだセキュリティー・マネージャーに登録されていないことを検出すると、ユーザー ID とパスワードを要求するために HTTP 基本認証が使用されます。CICS はこの情報を使用して証明書を登録します。 このオプションを使用する場合は、SSL(CLIENTAUTH) を指定する必要があります。
    e. AUTOMATIC を指定して、セキュリティー・マネージャーへの自動登録機能を持つ SSL クライアント証明書認証を使用する (AUTOREGISTER オプションの場合と同様) か、証明書を送信しない場合に HTTP 基本認証を使用します (BASIC オプションの場合と同様)。
17. HTTP 基本認証に使用されるレルムを指定するには REALM 属性を使用します。 エンド・ユーザーは、基本認証のプロセス中にレルムを確認できます。これは、要求された認証情報 (つまり、ユーザー ID およびパスワード) が適用されるリソースのセットを示します。
    a. 異なる TCPIPSERVICE 定義を使用して提供されたリソースについて異なる認証情報が必要な場合は、この要件をエンド・ユーザーに明確にするために異なるレルムを指定します。
    b. エンド・ユーザーが複数のリソースに同じ認証情報を使用する場合は、複数の TCPIPSERVICE 定義に同じレルムを指定することができます。
    c. REALM 属性を指定しないと、デフォルトのレルムが使用されます。 デフォルトのレルムは次のとおりです。

       ```
       realm="CICS application aaaaaaaa"
       ```

       ここで、 *aaaaaaaa* は CICS 領域のアプリケーション ID です。

CICS Web サポート用の TRANSACTION リソース定義の作成

TRANSACTION リソース定義は、 CICS Web サポート用の別名トランザクションを定義します。別名トランザクションは、要求の受け取り、アプリケーション・ビジネス・ロジックの実行、HTTP 応答の作成、および HTTP 応答のコード・ページ変換などの、HTTP 要求処理の後のステージを取り扱います。別名トランザクションは、非 HTTP 要求の処理にも使用できます。

このタスクについて

CICS はデフォルト別名トランザクションの CWBA のリソース定義を提供します。代わりの別名トランザクション名は、以下のような目的に使用できます。

- 監査、モニター、またはアカウンティング
- セキュリティーのためのリソースおよびコマンドの検査
- 開始優先順位の割り振り
- DB2® リソースの割り振り
- 各種 CICS アプリケーション・プログラムへの各種ランナウェイ値の割り当て
- トランザクション・クラスの制限

別名トランザクション定義をいくつでもセットアップできます。 URIMAP 定義またはアナライザー・プログラムを使用して、特定の要求に必要な別名トランザクションを指定できます。

重要: アプリケーション生成の応答に使用される別名トランザクション (CWBA など) の優先順位が、 Web 接続タスクに関連するトランザクション (CWXN または CWXU など) の優先順位と同じか、それより高くなるようにしてください。 242 ページの『CICS Web サポート・トランザクション (CWXN、CWXU、CWBA、CW2A) の優先順位』を参照してください。

『TRANSACTION resource definitions』で、このタイプのリソース定義について説明しています。加えて、以下の点に注意してください。

手順

- CWBA の定義に基づいて別名トランザクション定義を作成し、優先順位の変更などの、必要な変更を行ってください。 CWBA の定義は以下のとおりです。

```
DEFINE TRANSACTION(CWBA)   GROUP(DFHWEB)
       PROGRAM(DFHWBA)     TWASIZE(0)
       PROFILE(DFHCICST)   STATUS(ENABLED)
       TASKDATALOC(BELOW)  TASKDATAKEY(USER)
       RUNAWAY(SYSTEM)     SHUTDOWN(ENABLED)
       PRIORITY(1)         TRANCLASS(DFHTCL00)
       DTIMOUT(NO)         INDOUBT(BACKOUT)
       SPURGE(YES)         TPURGE(NO)
       RESSEC(NO)          CMDSEC(NO)
```

- 別名トランザクション定義では、CICS 提供の別名プログラム DFHWBA を使用する必要があります。 この別名プログラムは、要求を処理するためにユーザーが指定したユーザー・アプリケーション・プログラムを呼び出します。
- 別名トランザクション定義はローカル・トランザクションでなければなりません。

HTTP サーバーとしての CICS のための URIMAP リソースの作成

CICS が HTTP サーバーとしてインバウンド要求を処理する場合、URIMAP リソースは HTTP 要求の処理方法を指定します。

このタスクについて

HTTP 要求への応答を、以下の方法で生成できます。

- Web 対応アプリケーション・プログラムを作成することにより、動的応答を提供できます。
- CICS 文書テンプレートまたは z/OS UNIX ファイルで、静的応答を提供できます。

どちらの方式を使用するとしても、URIMAP リソースを作成する必要があります。指定する属性の多くは、両方の方式に共通しています。それ以外の属性は、どちらか一方の方式に適用されます。

HTTP サーバーとしての CICS の共通 URIMAP 属性の指定

HTTP 要求をどのように処理するかは、URIMAP リソース定義で定義されます。URIMAP で指定される属性の多くは、CICS が HTTP サーバーとして動作するすべての構成に適用されます。

このタスクについて

URIMAP は、インバウンド HTTP 要求で受信される URL に対応するリソースであり、要求の処理に関する情報を提供します。

手順

1. STATUS 属性を使用して、URIMAP 定義を使用可能な状態でインストールするか、使用不可の状態でインストールするかを指定します。
2. SERVER の USAGE 属性を指定します (HTTP サーバーとしての CICS)。
3. URIMAP で処理する URL を指定します。 URL は CICS を介して Web クライアントが使用できるようにするリソースを表し、いくつかの構成要素から成ります。 例えば、`http://www.example.com/software/index.html?n=John&s=Smith` には、以下の構成要素が含まれます。
   - スキーム構成要素は `http`
   - ホスト構成要素は `www.example.com`
   - パス構成要素は `/software/index.html`
   - 照会ストリングは `?n=John&s=Smith`

   URL の構成要素については、11 ページの『URL の構成要素』で説明しています。
4. URL の構成要素を、URIMAP リソースの対応する属性で指定します。
   a. SCHEME 属性でスキーム構成要素を指定します。 値として HTTP または HTTPS を指定できます。スキーム構成要素の後に区切り文字 `//` を含めないでください。SCHEME(HTTP) を指定した場合、URIMAP は、HTTP スキームまたはより安全な HTTPS スキームのどちらかを使用して行われた Web

クライアント要求を受け入れます。 SCHEME(HTTPS) を指定した場合、URIMAP は、HTTPS スキームを使用して行われた Web クライアント要求のみ受け入れます。

b. 異なるホスト名が含まれる複数の URL を区別する必要がある場合は、HOST 属性でホスト構成要素を指定します。 ホスト構成要素を指定するときは、以下のようにします。
   - ポート番号は含めないでください。
   - ホスト名、あるいは IPv4 または IPv6 アドレスを指定できます。
   - 単一アスタリスク (*) を指定できます。

   このオプションは、複数のホスト名を使用しない場合や、複数のホスト名を区別しない場合に使用します。このオプションを使用すると、URIMAP 定義は着信 URL のどのホスト名とも一致します。

c. PATH 属性でパス構成要素を指定します。
   - パス構成要素の先頭の区切り文字 / (スラッシュ) は省略できます。CICS によって自動的に付加されるためです。
   - パスの最後に、ワイルドカード文字としてアスタリスクを使用できます。例えば、
     - `/software/*` を指定した場合、URIMAP リソースは、ストリング `/software/` で始まるパスを含んだすべての要求と一致します。
     - `/*` を指定すると、URIMAP リソースは、HOST 属性で指定したホストに送られるすべての要求と一致します。

     ワイルドカードが含まれる複数の URIMAP リソースが HTTP 要求と一致した場合は、最も具体的に一致したものが採用されます。
   - インバウンド URL に照会ストリングがあり、URIMAP 定義を特定の照会に適用する場合は、先頭が疑問符 (?) 文字の照会ストリングを PATH 属性に含めます。ワイルドカードのアスタリスクを含んだパス構成要素の後に照会ストリングを指定できますが、照会ストリング自体にアスタリスクを含めることはできません。厳密な照会ストリングを指定する必要があります。URIMAP 定義で照会ストリングを指定しない場合、マッチングはパスについてのみ行われ、要求内の照会ストリングは無視されます。
   - CICS 文書テンプレートを使用する静的応答の場合は、照会ストリングを使用して URIMAP 定義を選択するか、照会ストリングを文書テンプレートに置き換えることができます。

5. オプション: TCPIPSERVICE 属性で、この URIMAP 定義が関係するインバウンド・ポートを定義した TCPIPSERVICE 定義の名前を指定します。

   この属性を指定しないと、URIMAP 定義は任意のインバウンド・ポート上の一致する HTTP 要求に適用されます。

   SCHEME(HTTPS) が指定された URIMAP がインバウンド要求と一致すると、CICS は、要求で使用されたインバウンド・ポートが SSL を使用しているかどうかをチェックします。SSL がポートに指定されていない場合、要求は 403 (Forbidden) 状況コードで拒否されます。 URIMAP 定義がすべてのインバウンド・ポートに適用されると、このチェックにより、Web クライアントは非セキュアなポートを使用してセキュアなリソースにアクセスすることができなくなります。スキームとして HTTP を指定した URIMAP 定義に対しては検査が行わ

れないため、Web クライアントは、非セキュアなポート、またはセキュアな (SSL) ポートのどちらを使用しても、保護されたリソースにアクセスできます。

次のタスク

インバウンド HTTP 要求への応答をどのようにして提供するかによっては、さらに属性を指定します。

- Web 対応アプリケーション・プログラムを作成することにより、動的応答を提供できます。
- CICS 文書テンプレートまたは z/OS UNIX ファイルで、静的応答を提供できます。

  『HTTP 要求に対するアプリケーション応答のための URIMAP 属性の指定』
  URIMAP で指定される属性の中には、HTTP 要求への応答をアプリケーション・プログラムを使用して提供する場合のみ適用されるものもあります。

  118 ページの『HTTP 要求に対する静的応答のための URIMAP 属性の指定』
  URIMAP で指定される属性の中には、CICS 文書テンプレートまたは z/OS UNIX ファイルを使用して HTTP 要求に対する静的応答を提供する場合のみ適用されるものもあります。

HTTP 要求に対するアプリケーション応答のための URIMAP 属性の指定

URIMAP で指定される属性の中には、HTTP 要求への応答をアプリケーション・プログラムを使用して提供する場合のみ適用されるものもあります。

このタスクについて

URIMAP リソースには、HTTP 要求への応答をアプリケーション・プログラムを使用して提供する場合に使用できる属性がいくつかあります。

手順

1. 応答を提供するアプリケーション・プログラムの名前を PROGRAM 属性に指定します。 指定したプログラムに HTTP 要求が渡されます。 あるいは、アプリケーション・プログラムの名前を提供するアナライザー・プログラムまたはコンバーター・プログラムを指定することもできます。アナライザー・プログラムまたはコンバーター・プログラムは、この属性で指定されたアプリケーション・プログラムの名前を変更することもできます。
2. 別名トランザクションの名前を TRANSACTION 属性に指定します。 別名トランザクションとは、応答を提供するプログラムが実行される CICS に対して定義されたトランザクションのことです。デフォルトの別名トランザクションは CWBA です。

   アナライザー・プログラムを使用する場合は、別名トランザクションの名前を提供したり変更したりできます。
3. 別名トランザクションが接続される際に使用されるユーザー ID を USERID 属性に指定します。 URIMAP 定義で指定するユーザー ID は、Web クライアントから取得されるユーザー ID (その接続の TCPIPSERVICE 定義の AUTHENTICATE 属性で指定) でオーバーライドされます。アナライザー・プロ

グラムを使用する場合は、これらのユーザー ID のいずれかを変更するか、または 1 つ提供することができます。ユーザー ID が指定されない場合、デフォルトのユーザー ID は CICS のデフォルト・ユーザーです。

4. アナライザー・プログラムを使用する場合は、ANALYZER(YES) を指定します。 アナライザー・プログラムは、この URIMAP 定義が関係する TCPIPSERVICE リソースの URM 属性で指定されます。 アナライザー・プログラムを使用する場合も、PROGRAM、TRANSACTION、USERID、および CONVERTER 属性を指定できます。これらの属性に指定する値はアナライザー・プログラムへの入力として使用されますが、アナライザー・プログラムによってオーバーライドされる可能性があります。あるいは、これらの属性を省略して、アナライザー・プログラムに指定させることもできます。
5. コンバーター・プログラムを使用する場合は、プログラムの名前を CONVERTER 属性に指定します。 アナライザー・プログラムとは異なり、コンバーター・プログラムと TCPIPSERVICE 定義は関連がありません。コンバーター・プログラムを使用する場合も、PROGRAM 属性を指定することができます。この属性に対して指定する値は、コンバーター・プログラムへの入力として使用されます。コンバーター・プログラムは、要求を処理する別のアプリケーション・プログラムを指定するために、 PROGRAM 属性を変更することがあります。

HTTP 要求に対する静的応答のための URIMAP 属性の指定

URIMAP で指定される属性の中には、CICS 文書テンプレートまたは z/OS UNIX ファイルを使用して HTTP 要求に対する静的応答を提供する場合のみ適用されるものもあります。

始める前に

パス・マッチングを使用する場合は、URIMAP リソースで PATH 属性を指定するときに、アスタリスク文字 (*) を指定しておく必要があります。

このタスクについて

URIMAP リソースには、HTTP 要求に対する静的応答を提供する場合に指定する属性がいくつかあります。

URIMAP リソースは、CICS 文書テンプレートのセキュリティー、および静的応答として配信される z/OS UNIX ファイルのセキュリティーを制御しません。これらの項目を基本認証およびリソース・レベルのセキュリティーを使用して保護する方法については、171 ページの『第 11 章 CICS Web サポートのセキュリティー』を参照してください。

手順

1. MEDIATYPE 属性で静的応答のデータ内容を指定します。 例えば、データ内容が HTML の場合は `text/html`、XML の場合は `text/xml` を指定します。メディア・タイプの詳細については、11 ページの『IANA メディア・タイプおよび文字セット』を参照してください。 この属性は必須であり、デフォルト値はありません。 CICS はこの情報を使用して、応答の Content-Type ヘッダーを作成します。

2. MEDIATYPE 属性でデータ内容としてテキスト・タイプを指定した場合は、コード・ページ変換に必要な以下の属性を指定します。
   a. CHARACTERSET 属性でターゲット文字セットを指定します。 ターゲット文字セットとは、CICS が静的応答を Web クライアントに送信する前に変換する際の変換先の文字セットのことです。 CICS は IANA で指定された文字セットをすべてサポートするわけではありません。 413 ページの『付録 A. HTML コード化文字セット』では、CICS でサポートされている IANA 文字セットがリストされています。 この情報は、応答の Content-Type ヘッダーに組み込まれます。
   b. 静的文書のエンコードに使用される IBM コード・ページ (EBCDIC) を、HOSTCODEPAGE 属性で指定します。

   その他のコンテンツ・タイプでは、コード・ページ変換は存在しません。
3. CICS 文書テンプレートを使用して応答を提供する場合は、文書テンプレートの名前を TEMPLATENAME 属性で指定します。 指定する名前は、文書テンプレートの属性を定義した DOCTEMPLATE リソースの名前です。 パス・マッチングを使用する場合は、CICS 文書テンプレートの名前の最後に、ワイルドカード文字としてアスタリスクを組み込みます。CICS は、それぞれの HTTP 要求のパスでワイルドカード文字になっている部分を取って、それをテンプレート名の一番後ろの部分と置換します。 URIMAP 定義属性に、パス・マッチングの動作を示す例が記載されています。

   URL に照会ストリングがある場合、CICS は、指定された CICS 文書テンプレートに照会ストリングの内容をシンボル・リストとして渡します。 CICS が内容を渡すのは、URIMAP 定義の PATH 属性でその照会ストリングがまだ使用されていないときだけです。
4. z/OS UNIX ファイルを使用して静的応答を形成する場合は、ファイルの名前を HFSFILE 属性で指定します。

   z/OS UNIX ファイルは、絶対 (完全修飾) パス、または CICS 領域のユーザー ID の HOME ディレクトリーからの相対的なパスとして指定できます。絶対パスはスラッシュ文字 (/) から始まりますが、相対パスはそうではありません。

   CICS 領域のユーザー ID には、z/OS UNIX へのアクセス権限に加え、ファイルが含まれている z/OS UNIX ディレクトリー、およびファイルそのものへのアクセス権限が必要です。詳しくは、*Java Applications in CICS*を参照してください。

   パス・マッチングを使用する場合は、z/OS UNIX ファイルのパスの最後に、ワイルドカード文字としてアスタリスクを組み込む必要があります。CICS は、それぞれの HTTP 要求のパスのうち、ワイルドカード文字でカバーされている部分を取って、それを z/OS UNIX ファイル・パスの一番後ろの部分と置換します。ディレクトリー構造を少なくとも 1 レベル明示的に指定する必要があります。HFSFILE 属性でアスタリスクを単独で使用することはできません。

   URIMAP 定義属性に、パス・マッチングの動作を示す例が記載されています。

   照会ストリングを z/OS UNIX ファイルに置き換えることはできません。

第 8 章 HTTP サーバーとしての CICS の管理

さまざまな CICS Web サポート・タスクを実行するよう CICS を構成し、Web クライアントからの要求への応答を開始すると、CICS Web サポート構造を管理することと、リソースが使用不可の場合に要求に対して適切な処理を提供することが必要になることがあります。

このタスクについて

HTTP サーバーとしての CICS の管理には、以下に示すように、HTTP 要求を管理する URIMAP 定義が役立ちます。

- 特定の HTTP 要求 (例えば、CICS プログラム) で必要なリソースが使用不可の場合に、実行中の CICS システムで、その HTTP 要求を動的にリダイレクトまたはリジェクトします。
- CICS によって仮想ホストを作成します。仮想ホストは CICS コマンドを使用して管理できます。

URIMAP 定義がない場合、特定のポートのすべての要求を管理する TCPIPSERVICE リソース定義のレベルで CICS Web サポートを管理できますが、URIMAP リソース定義のレベルで管理する方が制御性に優れています。

仮想ホスティングの管理

INQUIRE HOST コマンドおよび仮想ホスト・ブラウズ・コマンドを使用して、CICS が URIMAP 定義から作成した仮想ホストを調べることができます。仮想ホストの状況を変更するには、SET HOST コマンドを使用します。

HTTP サーバーとしての CICS 用にセットアップする URIMAP 定義 (URIMAP 定義に USAGE(SERVER) を指定) にはそれぞれ、その要求で Web クライアントが提供することが予期されるホスト名を組み込みます。 CICS では、1 つの CICS 領域内で同じホスト名および同じ TCPIPSERVICE 定義を指定しているすべての URIMAP 定義を、単一のデータ構造にまとめてグループ化することで、自動的に仮想ホストを作成します。 TCPIPSERVICE 定義を指定しておらず、すべての TCPIPSERVICE 定義に適用される URIMAP 定義は、一致するホスト名を指定しているすべてのデータ構造に追加されます。そのため、そうした URIMAP 定義は、複数のデータ構造の一部となる可能性があります。次に、URIMAP 定義のこうした各グループは、単一のユニットとして管理可能な仮想ホストを形成します。

以下の CICS コマンドを使用して、CICS が URIMAP 定義から作成した仮想ホストを管理することができます。

- INQUIRE HOST コマンド。仮想ホストの状況を調べます。このコマンドは、仮想ホストのホスト名、関連付けられている TCPIPSERVICE 定義 (またはその CICS 領域内のすべての TCPIPSERVICE 定義に関連付けられているかどうか)、および、使用可能か使用不可かを通知します。
- **SET** HOST コマンド。仮想ホストの状況を使用可能または使用不可に設定します。仮想ホストを使用不可に設定すると、その仮想ホストを構成しているすべて

の URIMAP 定義が、アプリケーションからアクセスできなくなります。(ただし、この方法で使用不可にした URIMAP 定義は廃棄できませんので注意してください)。仮想ホストを使用不可に設定した場合、CICS は Web クライアントに HTTP 503 応答 (Service Unavailable (サービス利用不可)) を戻します。

- 仮想ホスト・ブラウズ・コマンド。CICS システム内の仮想ホストをブラウズします。

統計プログラム DFH0STAT には、CICS が作成した仮想ホストを表示するレポートが組み込まれています。

仮想ホストを構成しているすべての URIMAP 定義が削除されると、CICS は自動的にその仮想ホストを削除します。CICS によって自動的に作成された仮想ホストを管理したくない場合があります。そうであれば、仮想ホストを無視し、URIMAP 定義のレベルで管理できます。

また、アナライザー・プログラムを使用して仮想ホストを処理することもできます。 HTTP 要求のホスト名がアナライザー・プログラムに渡されるので、要求に対してホスト依存の応答が提供されるようにこのプログラムをコーディングすることができます。ただし、このように仮想ホストをセットアップした場合は、INQUIRE HOST、**SET** HOST、および仮想ホスト・ブラウズ・コマンドで仮想ホストを管理することはできません。

別の URL への HTTP 要求のリダイレクト

URIMAP 定義を使用することにより、HTTP サーバーとしての CICS に対する HTTP 要求を、一時的または永続的に、別の URL にリダイレクトできます。

このタスクについて

リダイレクトを使用して、要求は常に Web クライアントを別の URL にリダイレクトすることにより処理されるようにすることができます。または、意図していたリソースを使用できない間は、リダイレクトを使用して、要求に対して一時的な応答 (例えば、要求されているアプリケーションがオフラインになっていることを要求側に示すページ) を提供することができます。どちらの場合も、URIMAP リソースの LOCATION および REDIRECTTYPE 属性でリダイレクトを指定します。

これらの属性は URIMAP の作成時に指定でき、URIMAP がアクティブである間は変更可能です。

手順

1. URIMAP リソースの LOCATION 属性を使用して、一致する HTTP 要求のリダイレクト先の URL (最大 255 文字) を指定します。 この指定は、スキーム、ホスト、およびパス構成要素を含んだ完全なものでなければなりません。区切り文字をすべて含めます。CICS は、その URL が完全であること、および正しく区切られていることをチェックしますが、CICS では、宛先が有効であるかどうかのチェックは行いません。

   必要に応じて、LOCATION 属性でフラグメント ID (先頭に # 文字) を使用して、その URL で識別される項目内の参照または機能を Web ブラウザーに示すことができます。例えば、フラグメント ID を、文書のサブセクションの ID に

することができます。フラグメント ID を使用できるかどうか、およびその使用方法を確認するには、提供する内容のタイプ (例えば HTML) の技術仕様書を調べてください。

2. URIMAP リソースの REDIRECTTYPE 属性を使用して、一時的または永続的リダイレクトを指定します。 要求が一時的にリダイレクトされている場合、その応答に使用される HTTP 状況コードは 302 (Found (検出)) です。要求が永続的にリダイレクトされている場合、その応答に使用される HTTP 状況コードは 301 (Moved Permanently (永続的に移動)) です。 CICS は、リダイレクト応答を構成しますが、その応答をカスタマイズすることはできません。
   REDIRECTTYPE(TEMPORARY) または REDIRECTTYPE(PERMANENT) が指定されている場合、URIMAP 定義の LOCATION 属性によって URIMAP 定義のその他の属性がオーバーライドされ、HTTP 要求がリダイレクトされます。
   - 要求が一時的にリダイレクトされている場合、その応答に使用される HTTP 状況コードは 302 (Found (検出)) です。
   - 要求が永続的にリダイレクトされている場合、その応答に使用される HTTP 状況コードは 301 (Moved Permanently (永続的に移動)) です。
3. リダイレクトを取り消すには、REDIRECTTYPE 属性を NONE に設定します。 LOCATION 属性で指定した URL を変更する必要はありません。リダイレクトがアクティブのときだけ使用されるからです。

HTTP 要求の拒否

CICS システム内のアプリケーションまたはリソースが一時的に使用不可になったときにリダイレクトでカバーできない場合や、アプリケーションまたはリソースが永続的に除去された場合は、いくつかの異なるレベルで HTTP 要求を拒否できます。Web クライアントからの長期持続接続のために CICS 領域が過負荷になるリスクがある場合は、接続スロットリングを実装することもできます。

このタスクについて

HTTP 要求を以下のレベルで拒否できます。

- 特定の要求 URL のレベル。このレベルの細分度を達成するには、要求 URL を URIMAP 定義でカバーします。 URIMAP 定義がない場合は、要求を処理するアナライザー・プログラムに変更を加えることによって HTTP 要求の処理を変更できますが、この手法はあまり便利ではありません。
- 仮想ホストのレベル (特定のホスト名に対するすべての要求に対応)。要求を仮想ホストに取り込むには、要求が URIMAP 定義に定義されている必要があります。
- ポートのレベル。1 つのポートが 1 つの TCPIPSERVICE 定義にマップされます。例えば、デフォルト HTTP ポート 80 の TCPIPSERVICE 定義を無効にすると、SSL または標準外ポートを使用する HTTP 要求以外の HTTP 要求を、CICS が受信できなくなります。
- すべてのポートのレベルで完全に拒否する。 CEMT トランザクションまたは CPSM で CICS の内部 TCP/IP ソケット・サポートをシャットダウンし、したがって CICS Web サポートを完全にシャットダウンできます。

一般に、HTTP 要求を拒否する際のレベルを細かくすればするほど、CICS が Web クライアントに与えるエラー応答の有用性が高まります。例えば、URIMAP 定義ま

たは仮想ホストを無効にすることによって HTTP 要求を拒否すると、CICS は Web エラー・プログラムを介して Web クライアントに HTTP 503 応答 (Service Unavailable (サービス利用不可)) を返します。 Web エラー・プログラムを調整することにより、この応答を変更できます。ただし、TCPIPSERVICE 定義を無効にすることによって HTTP 要求を拒否した場合、Web クライアントは、サーバー・エラーを示す一般エラー応答のみを受信します。

長期持続接続をセットアップした Web クライアントが多すぎるためにパフォーマンス上の問題が起こった CICS 領域がある場合は、そのような要求を制御するための接続スロットリングを実装できます。接続スロットリングは、特定のポートに対して単一の CICS 領域が受け入れる持続 HTTP 接続の数を自動的に制限するので、手操作による介入は必要ありません。 Web クライアントは、ポートを共用していて同じサービスを提供する別の CICS 領域に再接続することによって、持続接続を取得できます。

接続スロットリングを実装すると、設定した制限を超えた Web クライアントは、各応答の後にその接続を閉じるよう要求されます。 HTTP/1.1 サーバーは通常、持続接続を許可する必要があります。したがって、パフォーマンス上の問題が起こった CICS 領域に自動介入を設定する必要がある場合のみ、接続スロットリングをセットアップしてください。 46 ページの『CICS Web サポートによる持続接続の処理方法』で、接続スロットリングの影響について説明しています。

HTTP 要求を以下のようにして拒否またはスロットリングできます。

手順

- 特定の要求 URL への要求を拒否するには、以下の手順を実行します。
  1. URL の URIMAP 定義がある場合は、URIMAP 定義を無効にします。 パスにワイルドカード文字を含んだ、あまり具体的でない URIMAP 定義が、要求 URL と一致しないことを確認してください。 CICS は、Web エラー・プログラムを介して Web クライアントに HTTP 503 応答 (Service Unavailable (サービス利用不可)) を返します。 Web エラー・プログラムを変更することにより、この応答を調整できます。
  2. URL の URIMAP 定義がない場合は、要求が行われたポートの TCPIPSERVICE 定義に関連付けられたアナライザー・プログラムを変更することによって、要求を拒否できます。 各 URL に対して個別の拒否メッセージを提供するようにアナライザー・プログラムをコーディングすることも、使用不可のどの URL に対しても単一のメッセージを提供することもできます。 441 ページの『付録 E. アナライザー・プログラム』で、拒否された要求の処理に適したアクションについて説明しています。
- 仮想ホストへの要求 (つまり、特定のホスト名に対するすべての要求) を拒否するには、**SET HOST** コマンドを使用して仮想ホストを無効にします (121 ページの『仮想ホスティングの管理』を参照してください)。 CICS は、Web エラー・プログラムを介して Web クライアントに HTTP 503 応答 (Service Unavailable (サービス利用不可)) を返します。 Web エラー・プログラムを変更することにより、この応答を調整できます。
- CICS 領域への特定のポートでの持続接続の数を制限するには、TCPIPSERVICE 定義をセットアップするときに MAXPERSIST 属性の値を指定します。 制限に達すると、接続スロットリングが実施されます。 CICS は、それ以降そのポート

で接続する Web クライアントへの応答で、Connection: close ヘッダーを送信します。この応答を受信した Web クライアントは、接続を閉じなければなりません。これらの新たな Web クライアントのパフォーマンスは、それらが同一 CICS 領域に再接続する場合は特に、接続スロットリングの影響を受けることがあります。これらの Web クライアントは、ポートを共用していて制限に達していない別の CICS 領域に接続した時点で、代わりにそこで持続接続を維持できます。CICS 領域への持続接続を既に持っていた Web クライアントは、その持続接続を維持できます。 持続接続を持っている Web クライアントがその接続を閉じると、CICS 領域は新たな持続接続の受け入れを再開します。

- 特定のポートでの要求をすべて拒否するには、CEMT または CPSM で SET TCPIPSERVICE コマンドを使用して、TCPIPSERVICE 定義を無効にします。 ポートの listen の停止に関して、アクティブ・タスクに完了を許可して普通に停止するか、アクティブ・タスクを異常終了して直ちに停止するかを選択できます。
- インバウンド要求とアウトバウンド要求をすべて拒否し、CICS Web サポートを完全に停止するには、CEMT または CPSM で SET TCPIP コマンドを使用して TCP/IP を閉じます。 アクティブ・タスクに完了を許可して普通に閉じるか、アクティブ・タスクを異常終了して直ちに閉じるかを選択できます。

お気に入りアイコンの提供

多くの Web ブラウザーでは、ユーザーが Web ページにアクセスしたりブックマークを付けたりすると、お気に入りアイコン (favicon) が自動的に要求されます。URIMAP 定義を使用すると、favicon を静的応答として提供することができます。

このタスクについて

favicon は Web サイト・アイコンやショートカット・アイコン、URL アイコン、ブックマーク・アイコンとも呼ばれ、特定の Web サイトや Web ページに関連付けられた 16×16 ピクセル、32×32 ピクセル、または 64×64 ピクセルの正方形のアイコンです。 Web デザイナーは、favicon を作成して Web サイトや Web ページにインストールでき、ほとんどのグラフィカル Web ブラウザーは、そのインストール済み favicon を使用します。

Web ブラウザーは、以下の URL を使用して、デフォルトの favicon を要求します。

```
http://www.example.com/favicon.ico
```

ここで、`www.example.com` は、サイトのホスト名です。該当する場合は、代わりに HTTPS スキームを使用することもできます。以下の方法で favicon を提供できます。

- CICS 領域で使用されているホスト名に対して返される、デフォルトの favicon。
- CICS 領域で使用されているホスト名ごとに異なる、デフォルトの favicon。

Web ブラウザーが要求した favicon が提供されなかった場合、CICS はブラウザーに対して、エラー応答を送信します。

- CICS 提供のデフォルトのアナライザー DFHWBAAX を使用している場合は、404 (Not Found (未検出)) 応答が返されます。この状態では CICS メッセージは発行されません。

- サンプル・アナライザー DFHWBADX を使用する場合、または CICS TS バージョン 3 よりも前は必須だった URL 形式のみ解釈できる類似のアナライザーを使用する場合、アナライザーはパス `favicon.ico` を、間違って指定されたコンバーター・プログラム名と誤って解釈する可能性があります。この場合は、メッセージ DFHWB0723 が発行され、400 (Bad Request (不正な要求)) 応答がブラウザーに返されます。この状態を回避するには、favicon 要求を認識して、より適切なエラー応答を提供するようアナライザー・プログラムを修正するか、または URIMAP 定義を使用して favicon を提供します。どちらのアクションも、このような要求の場合はサンプル・アナライザー・プログラムがバイパスされることになります。

URIMAP 定義を使用して、すべてのホスト名または一部のホスト名に対応する favicon を提供するには、以下の手順を実行します。

手順

1. favicon を作成して、z/OS UNIX ファイル・システム内の適切な場所に保管します。
   a. favicon は、アイコン・エディター・パッケージを使用して作成するか、またはアイコン・コンバーター・プログラムを使用して、別の形式で作成されたイメージを変換することができます。
   b. favicon のサイズは 16 x 16 ピクセルにする必要があります。 ブラウザーは、サイズが正しくない favicon を無視することがあります。
   c. favicon は Windows のアイコン形式 (ファイル拡張子は `.ico`) で保管し、`favicon.ico` という名前にする必要があります。

   ほとんどの Web サーバーは、ホスト名のルート・ディレクトリーに favicon を保管します。CICS の場合、z/OS UNIX 上の任意の場所に保管されている favicon を、URIMAP 定義によって提供することができます。 CICS 領域には、z/OS UNIX へのアクセス権が必要であり、さらにこのファイルが保管されている z/OS UNIX ディレクトリーおよびファイルそのものへのアクセス権も必要です。 これらの権限を付与する方法については、*Java Applications in CICS*を参照してください。
2. CICS Explorer を使用して、favicon を静的応答として提供するための URIMAP 定義を作成します。これについては、次に記載されています。 115 ページの『HTTP サーバーとしての CICS の共通 URIMAP 属性の指定』および 118 ページの『HTTP 要求に対する静的応答のための URIMAP 属性の指定』。 次のサンプル URIMAP 定義属性は、CICS 領域によって使用されるすべてのホスト名に対応する favicon を提供します。

表 3. URIMAP 定義の favicon 値の例

属性	値	説明
URIMAP	favicon	URIMAP の名前。
グループ	MYGROUP	任意の適切なグループ名。
説明	Favicon	
状況	有効	
使用法	サーバー	HTTP サーバーとしての CICS 用。
スキーム	HTTP	HTTPS 要求とも一致します。

表 3. URIMAP 定義の favicon 値の例 (続き)

属性	値	説明
Host	*	* は、どのホスト名とも一致します。複数の異なる favicon を提供する場合は、ホスト名を指定します。
パス	/favicon.ico	ブラウザーは、このパスを使用して favicon を要求します。
TCPIPSERVICE		ブランクにしておくと、どのポートとも一致します。
メディア・タイプ (Media type)	image/x-icon	適切なメディア・タイプを選択します。
HFS ファイル (HFS file)	/u/cts/CICSHome/favicon.ico	HFS 内の favicon の場所。

注: favicon にはコード・ページ変換は不要であるため、CHARACTERSET や HOSTCODEPAGE オプションを指定しないでください。

robots.txt ファイルの提供

Web ロボットは、サービスの自動要求を行うプログラムです。例えば、検索エンジンはロボット (Web クローラーとも呼ばれる) を使用して、検索データベースに組み入れるページを取得します。 robots.txt ファイルを提供して、ロボットがアクセスできない URL を指定できます。

このタスクについて

Web サイトへのアクセス時に、ロボットは次の URL を使用して、robots.txt 文書の要求を行います。

```
http://www.example.com/robots.txt
```

ここで、www.example.com は、サイトのホスト名です。複数のポート番号を使用してアクセスできるホスト名が存在する場合、ロボットは、ホスト名とポート番号の組み合わせごとに robots.txt ファイルを要求します。ファイルにポリシーをリストし、すべてのロボットに適用するか、特定のロボットを指定することができます。 disallow ステートメントを使用して、ロボットがアクセスできない URL を指定します。 robots.txt ファイルを提供しても、ロボット除外標準に準拠していないロボットは、Web ページにアクセスして索引を作成する可能性があります。

Web ブラウザーが robots.txt ファイルを要求してもそのファイルが提供されなかった場合、CICS は以下のようなエラー応答を Web ブラウザーに送信します。

- CICS 提供のデフォルトのアナライザー DFHWBAAX を使用している場合は、404 (Not Found (未検出)) 応答が返されます。この状態では CICS メッセージは発行されません。
- サンプル・アナライザー DFHWBADX を使用する場合、または CICS TS バージョン 3 よりも前は必須だった URL 形式のみ解釈できる類似のアナライザーを使用する場合、アナライザーはパス robots.txt を、間違って指定されたコンバーター・プログラム名と誤って解釈する可能性があります。この場合は、メッセージ DFHWB0723 が発行され、400 (Bad Request (不正な要求)) 応答が Web ブラ

ウザーに返されます。この状態を回避するには、robots.txt 要求を認識して、より適切なエラー応答を提供するようアナライザー・プログラムを修正するか、または URIMAP 定義を使用して robots.txt ファイルを提供します。どちらのアクションも、このような要求の場合はサンプル・アナライザー・プログラムがバイパスされるようになります。

すべてまたは一部のホスト名に対する robots.txt ファイルを提供するには、以下の手順を実行します。

手順

1. robots.txt ファイルのテキスト内容を作成します。 robots.txt ファイルの作成に関する情報と例を、いくつかの Web サイトから入手できます。「robots.txt」または「ロボット除外標準」で検索し、適切なサイトを選択してください。
2. robots.txt ファイルを格納および提供する方法を決定します。このファイルは、URIMAP 定義のみを使用するかアプリケーション・プログラムを使用して提供できます。
   - robots.txt ファイルを z/OS UNIX システム・サービスに格納し、そのファイルを静的応答として提供できます (URIMAP 定義を使用)。ほとんどの Web サーバーはホスト名のルート・ディレクトリーに robots.txt ファイルを格納します。CICS では、z/OS UNIX 上の任意の場所に保管されているファイルを、URIMAP 定義によって提供することができます。また、複数のホスト名に対して、同じファイルを使用できます。

     z/OS UNIX に保管されたファイルを使用する場合は、CICS 領域に、z/OS UNIX へのアクセス権限と、ファイルを含む z/OS UNIX ディレクトリーおよびファイルそのものへのアクセス権限が必要になります。 これらの権限を付与する方法については、*Java Applications in CICS*を参照してください。
   - robots.txt ファイルを CICS 文書に変換し、その文書を静的応答として提供するか (URIMAP 定義を使用)、またはアプリケーション・プログラムからの応答として提供することができます。 「*CICS アプリケーション・プログラミング・ガイド*」では、CICS 文書テンプレートを作成する方法を説明しています。文書テンプレートは、区分データ・セット、CICS プログラム、ファイル、一時記憶域キュー、一時データ・キュー、出口プログラム、または z/OS UNIX システム・サービス・ファイルに保持することができます。
   - アプリケーション・プログラムを使用して robots.txt ファイルの内容を提供する場合は、適切な Web 対応アプリケーション・プログラムを作成します (87 ページの『第 6 章 HTTP サーバーとしての CICS のための Web 対応アプリケーション・プログラムの作成』を参照)。例えば、EXEC CICS WEB SEND コマンドに FROM オプションを指定して使用すると、robots.txt の情報を含んだデータ・バッファーを指定できます。あるいは、アプリケーション・プログラムを使用して、テンプレートから CICS 文書を配信することもできます。 text/plain のメディア・タイプを指定します。

     アプリケーション・プログラムを使用してロボットからの要求を処理し、どのロボットが Web ページにアクセスしているかを追跡することもできます。ロボットからの要求の User-Agent ヘッダーはそのロボットの名前を示し、From

ヘッダーにはそのロボットの所有者の連絡先情報が含まれています。アプリケーション・プログラムでこれらの HTTP ヘッダーを読み取り、ログに記録することができます。

3. CICS Explorer を使用して、Web ロボットが行う robots.txt ファイルの要求に対応する URIMAP 定義を作成します。 115 ページの『HTTP サーバーとしての CICS の共通 URIMAP 属性の指定』を参照してください。 次の URIMAP 定義属性のサンプルは、任意のホスト名の robots.txt ファイルに対する要求にどのようにして一致させるかを示しています。

表 4. URIMAP 定義のロボット値の例

属性	値	説明
URIMAP	robots	URIMAP の名前。
グループ	MYGROUP	任意の適切なグループ名。
説明	Robots.txt	
状況	有効	
使用法	サーバー	HTTP サーバーとしての CICS 用。
スキーム	HTTP	HTTPS 要求とも一致します。
Host	*	* は、どのホスト名とも一致します。別々の robots.txt ファイルを提供する場合は、ホスト名を指定します。
パス	/robots.txt	ロボットはこのパスを使用して robots.txt を要求します。
TCPIPSERVICE		ブランクにしておくと、どのポートとも一致します。ポートに応じて異なる robots.txt ファイルを提供する場合は、TCPIPSERVICE 定義名を指定します。

URL のパス構成要素には大/小文字の区別があることに注意してください。/robots.txt というパスは小文字で指定する必要があります。

4. robots.txt ファイルを静的応答として提供する場合は、URIMAP 定義を作成して、ファイルの場所と、CICS Web サポートが応答を構成するために使用するその他の情報を指定します。 118 ページの『HTTP 要求に対する静的応答のための URIMAP 属性の指定』で、この処理について説明しています。 例えば、次のような URIMAP 属性を指定することにより、EBCDIC コード・ページ 037 を使用して作成されて /u/cts/CICSHome ディレクトリーに格納された robots.txt ファイルを提供できます。

表 5. URIMAP 定義の静的文書プロパティーの例

属性	値
メディア・タイプ (Media type)	/text/plain
文字セット (Character set)	iso-8859-1
ホスト・コード・ページ (Host codepage)	037
HFS ファイル (HFS file)	u/cts/CICSHome/robots.txt

HFS ファイル名では、大/小文字が区別されます。

5. アプリケーション・プログラムを使用して robots.txt ファイルの内容を提供する場合は、プログラムが要求を処理しなければならないことを指定した URIMAP 定義を作成します。 117 ページの『HTTP 要求に対するアプリケーション応答のための URIMAP 属性の指定』で、この処理について説明しています。 例えば、以下のような URIMAP 定義属性を使用することにより、アナライザーもコンバーター・プログラムも関与させることなく、Web 対応アプリケーション・プログラムの ROBOTS に要求を処理させることができます。

表 6. URIMAP 定義の関連 CICS リソース・プロパティーの例

属性	値	説明
アナライザー	いいえ	要求にアナライザーは使用されません。
コンバーター (Converter)		ブランクにしておくと、コンバーター・プログラムなしになります。
トランザクション		ブランクにしておくとデフォルトの CWBA になります。
プログラム	ROBOTS	ROBOTS

Warning ヘッダー

HTTP メッセージに Warning ヘッダーがある場合、これには通常、ユーザーに読んでもらいたい情報が含まれています。 CICS Web サポートが Warning ヘッダーを持つメッセージを受け取ると、そのヘッダーに関連付けられているテキストが、CWBW 一時データ・キューに書き込まれます。

要求の警告ヘッダーを記録するために使用されるメッセージ番号は DFHWB0750 であり (HTTP サーバーとしての CICS の場合)、応答の警告ヘッダーの記録に使用されるメッセージ番号は DFHWB0752 です (HTTP クライアントとしての CICS の場合)。各警告ヘッダーのメッセージには、以下の情報が含まれます。

- その警告ヘッダーに関連付けられるテキスト。
- サーバーおよびクライアントの IP アドレス。

メッセージは、CSSL に間接的に送信される、CICS 提供の一時データ・キュー CWBW に書き込まれます。DFHDCTG グループには、キューのサンプル定義が含まれています。

CWBO は通常、 CICS Web サポートのメッセージ用に使用されるキューであり、CWBW は警告メッセージを分離しておくために提供されます。受信する警告ヘッダーが多すぎる場合、あるいは、もう不要な警告ヘッダー (クライアント要求に対して必ず送信される応答として、サーバーから繰り返し送信される警告など) を受信する場合は、CWBW 一時データ・キューを削除して、こうしたレコードを抑制することができます。

第 9 章 Web エラー・プログラム

要求エラーまたは異常終了が CICS Web サポート処理で発生した場合に、ユーザー置換可能 Web エラー・プログラムがエラー応答を Web クライアントに提供します。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

Web エラー・プログラムは、以下の状態で使用されます。

- CICS Web サポートが Web クライアントからの要求の初期処理において問題を検出した場合 (例えば、必要な情報が要求に欠落している場合や、要求の送信に時間がかかりすぎて受信タイムアウトになった場合)。
- インストールされた URIMAP 定義が要求に一致するが、URIMAP 定義か仮想ホストが使用不可である場合、または静的応答のリソースにアクセスできない場合。
- URIMAP マッチングが失敗し、TCPIPSERVICE 定義で指定されたアナライザーが要求を処理できず、制御を Web エラー・プログラムに渡す場合。
- URIMAP 定義でも、アナライザー・プログラム処理とコンバーター・プログラム処理でも、要求にサービスを提供するアプリケーション・プログラムを判別できない場合。
- アナライザー・プログラム、コンバーター・プログラム、またはユーザー作成アプリケーション・プログラムで異常終了が発生した場合。このケースでは、処理が失敗したとしても Web クライアントに応答を返すことができます。

Web エラー・プログラムは、以下の状態で使用されません。

- ソケット送信エラーまたは受信エラーが発生した場合。ソケットが閉じられており、Web クライアントに応答は送信されません。
- URIMAP でリダイレクト応答を指定している場合。これらの応答は、CICS によって構成され、カスタマイズできません。
- ユーザー作成アプリケーション・プログラムが処理を正常に完了して、例えば、クライアントがリソースでサポートされていないメソッドを指定したときに、エラーを示す応答を返す場合。これらの応答は、アプリケーションによって作成および送信されます。
- HTTP クライアントとしての CICS を含む処理の場合。Web クライアントは、サーバーにエラー応答を送信する場合には必要ありません。サーバーから受信した応答は、クライアント・アプリケーション・プログラムによって処理されます。
- EXEC CICS WEB RECEIVE コマンドの場合。受信された内容が損害を与える可能性があります。サブトピックで詳しく説明します。

CICS にクライアントとの持続接続が存在する場合、CICS は、Web エラー・プログラムを介してエラー応答が送信された後も接続を開いたままにしておきます。た

だし、CICS が 501 (Method Not Implemented (メソッドのインプリメントなし)) 状況コードを応答に対して選択した場合は例外で、この場合には、接続が CICS によって閉じられます。

ユーザー置換可能プログラムの作成に関する一般情報については、「*CICS Customization Guide*」のユーザー置換可能プログラムによるカスタマイズを参照してください。

2 つのユーザー置換可能 Web エラー・プログラムが、CICS で提供されています。

DFHWBERX、Web エラー・アプリケーション・プログラム

DFHWBERX は、**EXEC CICS WEB** コマンドおよび DOCUMENT API コマンドを使用して Web クライアント要求に関する情報を取得した後、エラー応答を作成して送信します。これはアプリケーション・プログラムと呼ばれています。
DFHWBERX は、アナライザー・プログラムで指定することもできますし、要求に対してエラー応答が常に求められる場合には、URIMAP 定義の PROGRAM 属性として指定することもできます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

DFHWBERX が使用されるのは、 TCPIPSERVICE 定義において CICS 提供のデフォルト・アナライザー・プログラム DFHWBAAX がアナライザー・プログラムとして指定されており、かつ要求と一致する URIMAP 定義が見つからない場合です。
DFHWBAAX は、wbra_server_program 出力パラメーターを使用して、
DFHWBERX をその要求を処理するアプリケーション・プログラムとして設定します。

DFHWBERX は、ユーザーによる置換が可能です。CICS では、DFHWBERX のアセンブリ言語のソース・コードのみが提供されています。

DFHWBERX が、CICS からパラメーター・リストやデフォルトの HTTP 応答を受け取るわけではありません。

DFHWBERX は以下のようにエラー応答を提供します。

- Web クライアントの要求が、メディア・タイプ text/xml を持つ POST 要求である場合、その要求は SOAP 1.1 要求であると見なされ、SOAP 1.1 障害応答が返されます。
- 要求が、メディア・タイプ application/soap+xml を持つ POST 要求である場合、その要求は SOAP 1.2 要求と見なされ、SOAP 1.2 障害応答が返されます。
- その他の要求はすべて標準の HTTP 要求であると想定され、適切な HTTP 応答が作成されて、404 (Not Found (未検出)) 状況コードとともに戻されます。

DFHWBERX は、以下のコマンドを使用します。

- **EXEC CICS WEB EXTRACT**。エラー応答が必要な Web クライアント要求の URL を取得します。
- **EXEC CICS DOCUMENT**。メッセージ・ボディを構成します。

- **EXEC CICS WEB WRITE HTTPHEADER** (SOAP 障害応答の場合)。適切な SOAP アクション・ヘッダーを書き込みます。
- **EXEC CICS WEB SEND**。適切な状況コードを指定し、Web クライアントに応答を送信します。応答ボディのコード・ページの変換のために、UTF-8 文字セットを指定します。

DFHWBERX は、Web クライアントの要求の内容を受信するのに **EXEC CICS WEB RECEIVE** コマンドを使用しません。 DFHWBERX をユーザー独自のアプリケーション・プログラムに置き換える場合は、このコマンドを使用しないでください。 CICS 提供のデフォルト・アナライザー・プログラム DFHWBAAX を使用している場合は、URIMAP 定義と一致しない任意の要求へのエラー応答を送信するために、DFHWBERX が使用されます。このような要求の内容を知ることはできませんし、その意図には悪意がある可能性があります。したがって、要求を受信しようとするのは望ましくありません。

DFHWBEP、Web エラー・プログラム

DFHWBEP は、エラー状態についての情報を提供するパラメーター・リストを CICS から受信し、また、CICS が Web クライアントに送信しようとする (状況コードおよび状況テキストを含む) デフォルト HTTP 応答を含むストレージのブロックも受信します。プログラムからデフォルト応答を使用や変更したり、EXEC CICS WEB および DOCUMENT API コマンドを使用して独自の応答を作成および送信したりできます。DFHWBEP はユーザーが置換可能です。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

エラー状態の評価

CICS により DFHWBEP に渡されるパラメーター・リストには、デフォルト応答で CICS が使用した 3 桁の HTTP 状況コードが含まれます。また、パラメーター・リストは、エラー・コード、異常終了コード、CICS メッセージ番号、応答と理由コード、およびエラーが発生したプログラムの名前などのエラー状態を識別する情報も提供します。

DFHWBEP をカスタマイズする場合、適切な入力パラメーターの範囲を使用して、状況コードのみに依存するのではなく、カスタマイズされた応答が適用される状態を識別するようにします。各状況コードは、CICS Web サポートがさまざまな目的で使用する可能性があります。プログラムが認識しない状況コードがある HTTP 応答は、変更しないで渡す必要があります。

CICS が提供するパラメーター・リストの確認の他に、EXEC CICS WEB EXTRACT コマンドおよび EXEC CICS EXTRACT TCPIP コマンドを使用して、要求行を確認し、エラー応答が必要な Web クライアントの要求に関連するその他の情報を取得する場合があります。WEB READ HTTPHEADER コマンドまたは HTTPHEADER 表示コマンドを使用して、要求の HTTP ヘッダーを読み取ることもできます。要求が無効状態またはタイムアウトになっている場合、これらのコマンドは使用できない可能性があります。

Web クライアント要求の内容を受け取る EXEC CICS WEB RECEIVE コマンドを、Web エラー・プログラムで使用しないでください。 DFHWBEP が処理するエラー状態の範囲では、Web クライアントの要求がタイムアウトになるか、必要な情報が欠落しているか、予期しない悪意のある内容の可能性があるか、または別のアプリケーション・プログラムにより既に受信されている場合があります。これらのいずれかの状態の要求を受信すると、問題が起こったり予測不能な結果になったりするおそれがあります。

エラー応答の作成および送信

CICS が提供するパラメーター・リストには、検出されたエラーのデフォルト HTTP 応答のほか、完全な応答メッセージの長さを示すパラメーターも格納されたストレージ・ブロックへのポインターが含まれます。ストレージ・ブロックには、状況表示行、HTTP ヘッダー、およびメッセージ・ボディを含む完全な HTTP 応答が格納されます。

Web エラー・プログラムは、以下のアクションのいずれかを選択できます。

1. デフォルト応答を変更しないで、CICS が Web クライアントに送信できるようにします。プログラムが認識しない状況コードがある HTTP 応答に対して、または状態を評価してデフォルト応答が適切であるとわかった場合にこのアクションを実行します。
2. **EXEC CICS** WEB および DOCUMENT API コマンドを使用して、新しい応答を作成し、Web クライアントに送信します。CICS は、メッセージを検査することで、さらに支援することができます。デフォルトの応答は廃棄されます。応答の HTTP ヘッダーを書き込むには WEB WRITE HTTPHEADER コマンドを使用し、応答をアセンブルして送信するには WEB SEND コマンドを使用します。デフォルトの ACTION(EVENTUAL) は DFHWBEP では許可されないため、コマンドに ACTION(IMMEDIATE) を指定する必要があります。API コマンドを使用した応答の作成および送信方法について詳しくは、96 ページの『応答に対する HTTP ヘッダーの書き込み』、 98 ページの『HTTP メッセージのエンティティー・ボディの作成』、および 100 ページの『HTTP サーバーとしての CICS からの HTTP 応答の送信』を参照してください。
3. ストレージのブロックのデフォルト応答を手動で変更して、それに応じてパラメーターの長さを更新し、CICS が Web クライアントに送信できるようにします。このアクションは、デフォルトのメッセージ・ボディを短い 1 テキストに置き換えるなど、デフォルト応答に小さい変更を加えるだけの場合に使用してください。 HTTP 応答を有効なままにしておくこと、および適切な長さを指定することに注意する必要があります。
4. ストレージの新しいブロックの新しい HTTP 応答を手動で構成して、ストレージの新しいブロックのアドレスおよび新しい応答の長さを渡し、CICS が Web クライアントに送信できるようにします。デフォルトの応答は廃棄されます。このアクションは、CICS がこの方法で構成されたメッセージをチェックする場合に十分な支援を提供できないため、推奨されなくなりました。CICS Transaction Server for z/OS, バージョン 3 リリース 2 より前にカスタマイズされたバージョンの DFHWBEP があり、それがこのように動作する場合は、**EXEC CICS** WEB および DOCUMENT API コマンドを使用して構成および送信される HTTP 応答に置き換えることを検討してください。

エラー応答の内容の訂正

EXEC CICS WEB および DOCUMENT API コマンドを使用して新しい応答を作成しようと、ストレージ・ブロック内のデフォルト応答を手動で変更しようと、新しいストレージ・ブロック内の新しい HTTP 応答を手動で構成しようと、エラー応答のすべての項目を変更できます。ただし、HTTP 応答を有効で適切な状態のままにしておくこと、また、ストレージ・ブロック内の応答を手動で処理する場合は、適切な長さを指定することが必要です。

応答には HTTP バージョン、状況コード、状況テキスト、必要な HTTP ヘッダー、およびメッセージ・ボディを含める必要があります。応答の形式は、目的の HTTP プロトコル仕様 (HTTP/1.0 または HTTP/1.1) に準拠していなければなりません。 API コマンドを使用している場合、CICS はこれらすべての要素に支援を提供します。

エラー応答の各項目について、以下のガイダンスに注意してください。

HTTP バージョン (HTTP/1.1 または HTTP/1.0)
デフォルト応答では、Web クライアントの HTTP バージョンに基づいて、CICS がバージョンを判断します。ストレージのブロックのデフォルト応答を処理している場合、応答のこの要素を変更しないでください。 API コマンドを使用して新しい応答を作成している場合、WEB EXTRACT コマンドを使用して、Web クライアントの HTTP バージョンを識別し、それに応じて応答を調整します。Web クライアントにより使用される HTTP バージョンは、応答の HTTP ヘッダー、状況コード、およびメッセージの内容の選択に影響を与える場合があります。HTTP/1.0 クライアントは、HTTP/1.1 仕様にある拡張機能を認識しない可能性があります。

数値状況コード (例: 404 または 500)
CICS はデフォルト応答に対して状況コードを選択します。デフォルト応答のこの要素を変更したり、新しい応答の状況コードを選択したりする場合は注意してください。 423 ページの『付録 C. CICS Web サポートの HTTP 状況コード・リファレンス』に、状況コードとそれが使用される理由をリストしています。 HTTP/1.1 規格には、すべての状況コード、および適切な使用のための要件についての追加情報が含まれます。CICS Web サポートが選択したものとは異なる状況コードを選択する場合は、その使用が HTTP/1.1 仕様に準拠するようにしてください。特に、状況コードが Web クライアントの HTTP バージョンに対して適切であることを確認してください。HTTP 以外のエラーの場合、CICS では状況コード 400 が常に使用されます。

理由の句、または状況テキスト (例:「Not Found (未検出)」)
デフォルト応答のこの要素を変更したり、新しい応答に対して独自の理由の句を指定したりすることができます。HTTP/1.1 規格により示される理由の句 (例: 「Not Found (未検出)」または「Bad Request (不正な要求)」) が推奨されますが任意指定です。HTTP/1.1 仕様には、各状況コードの理由句はローカルの同等のものに置き換えることができると記述されています。

HTTP ヘッダー
デフォルト応答には、CICS が応答に対して書き込んだヘッダー (例えば、日付ヘッダーやサーバー・ヘッダーなど) が含まれます。API コマンドを使用して新しい応答を作成する場合、応答を送信するときに CICS によってそれらのヘッ

ダーが自動的に追加されます。 415 ページの『付録 B. CICS Web サポートにおける HTTP ヘッダーの解説』に、CICS が書き込めるヘッダーをリストしています。CICS によって書き込まれるヘッダーは、HTTP バージョンのメッセージ向けです。これらのヘッダーは、HTTP 仕様準拠のうえで必須である可能性があるため、デフォルト応答を処理する場合は除去しないでください。必要に応じて、応答の HTTP ヘッダーを追加できます。HTTP/1.1 規格により、コンテキストのヘッダーの使用が許可されていることを確認します。異なる状況コードを選択すると、一部のヘッダーは HTTP/1.1 規格により必要とされる場合があります。

メッセージ・ボディ
デフォルト応答のメッセージ・ボディは、リリース行に示された状況コードおよび理由の句を繰り返します。デフォルト応答のこの要素を変更したり、新しい応答用の独自のメッセージ・ボディを提供したりできます。多くの状況コードでは、メッセージ・ボディを使用してさらに情報を提供できます。メッセージ・ボディに付随できない状況コードもあります。

コード・ページ変換

ストレージのブロックのデフォルト HTTP 応答は、EBCDIC コード・ページ 037 の DFHWBEP に渡されます。

Web エラー・プログラムの **EXEC CICS** WEB API コマンドを使用して新しいエラー応答を作成し、Web クライアントに送信した場合、**EXEC CICS** WEB API コマンドを使用するその他のプログラムと同じ方法で、コマンドで指定したようにコード・ページ変換が実行されます。

DFHWBEP は、ストレージ・ブロックに作成された応答に対して、コード・ページ変換設定を指定できません。ストレージ・ブロック内のデフォルトのエラー応答を変更するか、新しいストレージ・ブロック内の新しいエラー応答を提供して、それを送信のために CICS に返す場合、CICS は、ストレージ・ブロックにあった応答も EBCDIC コード・ページ 037 が使用されていると見なします。CICS は応答でコード・ページ変換を実行して、適切な ASCII 文字セットに変換し、クライアントに戻します。アナライザー・プログラムが処理パスに含まれ、コード・ページ変換に対して設定パラメーターが (各サーバーおよびクライアントのコード・ページ・パラメーターとして、または DFHCNV キーとして) ある場合、CICS ではコード・ページ変換にこれらのオプションが使用されます。含まれるアナライザー・プログラムがない、またはエラーが発生する前にアナライザーが起動されていなかった場合、応答に対して ISO-8859-1 文字セットが使用されます。この結果が適切でない場合は、代わりに **EXEC CICS** WEB API コマンドを使用して、応答を作成します。

DFHWBEP、web エラー・プログラムの入出力パラメーター

CICS は、エラー・コード、異常終了コード、およびエラー・メッセージを含む複数のパラメーターを DFHWBEP に渡します。DFHWBEP 用の出力パラメーターは、`wbep_response_ptr` および `wbep_response_len` です。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

DFHWBEP に関するすべてのパラメーターのリストと技術的説明については、 495 ページの『付録 H. Web エラー・プログラムの DFHWBEP に関する参照情報』を参照してください。 IP アドレス表記の詳細については、 7 ページの『CICS が受け入れる IP アドレス形式』を参照してください。

デフォルトでは、出力パラメーターは、CICS により生成されるデフォルト HTTP 応答を含むストレージのブロックに関連します。

- DFHWBEP の **EXEC CICS** WEB および DOCUMENT API コマンドを使用して新しい応答を作成し、Web クライアントに送信した場合、CICS はストレージのブロックの HTTP 応答を無視および廃棄するため、出力パラメーターは変更されないままです。
- ストレージのブロックのデフォルト応答を変更した場合、wbep_response_len の長さを更新して、バッファー全体の新しい長さを指定する必要があります。メッセージ・ボディの長さを計算したり、応答の Content-Length ヘッダーを変更したりする必要はありません。 CICS は指定したメッセージ・ボディの長さを検査し、必要に応じて Content-Length ヘッダーを訂正します。
- ストレージの新しいブロックで新しい HTTP 応答を手動で構成した場合、wbep_response_ptr のストレージの新しいブロック、および wbep_response_len の新しい応答の長さを渡す必要があります。

CICS Web サポートのデフォルトの状況コードおよびエラー応答

アナライザーまたはコンバーター・プログラムによって設定される応答コードおよび理由コードは、デフォルトの状況コードおよび関連した応答にマップされます。URIMAP 定義を使用して静的応答が生成されたときにエラーが発生する場合は、CICS はデフォルトの状況コードおよび関連した応答も選択します。状況コードおよび応答は、ユーザーによる置換が可能な Web エラー・プログラムの DFHWBEP によって変更できます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

HTTP プロトコル仕様では、要求を正常に完了できない場合に、HTTP 応答に対してサーバーが返すことのできる状況コードが定義されています。 423 ページの『付録 C. CICS Web サポートの HTTP 状況コード・リファレンス』を参照してください。

HTTP 応答の構造について詳しくは、 15 ページの『HTTP 応答』を参照してください。

CICS Web サポート処理の際にエラーが発生すると、適切なエラー応答の決定に役立つ情報が、パラメーター・リスト内の Web エラー・プログラム DFHWBEP に渡されます。

- アナライザーまたはコンバーター・プログラムによる処理の間にエラーが発生する場合は、パラメーター・リスト内のプログラムの応答および理由コードを使用してエラーを識別できます。
- URIMAP 定義を使用して静的応答を生成する際にエラーが発生する場合は、パラメーター・リスト内の関連する CICS メッセージ番号およびテキストを使用して、エラーを識別できます。

すべてのタイプのエラーで、状況コードを含む完全なデフォルト・エラー応答が Web エラー・プログラムに渡され、受け入れられたり、変更されたり、または置換されたりします。エラー応答には、 CICS メッセージおよび例外トレース項目が付けられます。

デフォルトの状況コードは、以下のとおりです。

表 7. アナライザー・プログラム処理エラーのデフォルトの状況コード

wbra_response	**デフォルトの状況コード**
URP_OK 以外の値	400 Bad Request (不正な要求)

表 8. コンバーター・デコード機能のデフォルトの状況コード

decode_response	**decode_reason**	**デフォルトの状況コード**
URP_EXCEPTION	URP_CORRUPT_CLIENT_DATA	400 Bad Request (不正な要求)
URP_EXCEPTION	URP_SECURITY_FAILURE	403 Forbidden (禁止)
URP_EXCEPTION	その他の任意の値	501 Not Implemented (インプリメントなし)
URP_INVALID	任意の値	501 Not Implemented (インプリメントなし)
URP_DISASTER	任意の値	501 Not Implemented (インプリメントなし)
その他の任意の値	任意の値	500 Internal Server Error (内部サーバー・エラー)

表 9. コンバーター・エンコード機能のデフォルトの状況コード

encode_response	**encode_reason**	**デフォルトの状況コード**
下記以外の任意の値 URP_OK URP_OK_LOOP	任意	501 Not Implemented (インプリメントなし)

表 10. 静的応答処理エラーのデフォルトの状況コード (URIMAP 定義の使用)

CICS メッセージ番号	**エラー**	**デフォルトの状況コード**
0364	ユーザー ID のセキュリティーを確立しようとしましたが、失敗しました。	403 Forbidden (禁止)
0758	静的応答を生成するのに必要なリソース (CICS 文書テンプレートまたは z/OS UNIX ファイル) に対する読み取り権限を、ユーザーが持っていません。	403 Forbidden (禁止)
0759	静的応答を生成するのに必要なリソース (CICS 文書テンプレートまたは z/OS UNIX ファイル) が見つかりません。	404 Not Found (未検出)
0760	静的応答を生成するのに必要な z/OS UNIX ファイルを読み取ることができません。	500 Internal Server Error (内部サーバー・エラー)

表 10. 静的応答処理エラーのデフォルトの状況コード (URIMAP 定義の使用) (続き)

CICS メッセージ番号	エラー	デフォルトの状況コード
0761	その他のエラー。	500 Internal Server Error (内部サーバー・エラー)

第 10 章 CICS アプリケーションからの HTTP クライアント要求

CICS は HTTP クライアントとして動作し、インターネット上の HTTP サーバーと通信することができます。ユーザー作成アプリケーション・プログラムは、CICS を介して HTTP サーバーに要求を送信し、そのサーバーから応答を受信します。

ユーザー作成アプリケーション・プログラムは、以下の処理を行えます。

- HTTP プロトコルを使用してハードウェアまたはソフトウェアと対話する。例えば、プリンターは多くの場合に、この方法で制御できます。
- 情報の項目 (例えば株価) を提供する HTTP アプリケーションにアクセスし、アプリケーションで使用するためにこの情報を取得する。

CICS Web サポートの HTTP クライアント機能は、Web ブラウザーとして使用するようには設計されていません。ユーザー・アプリケーション・プログラムは、サーバーから入手できる既知のリソースを個別に要求できますが、一般的にはインターネットをブラウズすることはありえません。サーバーから受信する可能性のある応答の範囲、およびそれらの応答を処理するために実行する必要のあるアクションは、事前選択されたリソースと、それらのリソースに関連付けられる可能性のあるエラー応答、および行う要求のタイプに関連付けられる可能性のあるエラー応答にのみ関係します。

HTTP クライアント要求を行うアプリケーション・プログラムを作成する前に、これらの要求の処理ステージを理解しておいてください。その理由は、ほとんどのステージは、アプリケーション・プログラムそのものから開始されるからです。 36 ページの『HTTP クライアントとしての CICS に対する HTTP 要求および応答の処理』を参照してください。HTTP サーバーとしての CICS に使用される TCPIPSERVICE リソース定義は、HTTP クライアントとしての CICS には適用されません。

CICS が行う要求と受信する応答では、コード・ページ変換が実行されます。 52 ページの『HTTP クライアントとしての CICS のためのコード・ページ変換』を参照してください。

HTTP クライアントとしての CICS を介した HTTP 要求

CICS からインターネット上のサーバーに対して行われる HTTP クライアント要求は、ユーザー作成アプリケーション・プログラムによって開始されます。

始める前に

HTTP クライアント要求を行うアプリケーション・プログラムを作成する前に、これらの要求の処理ステージを理解しておいてください。その理由は、ほとんどのステージはアプリケーション・プログラム自身から開始されるからです。 36 ページの『HTTP クライアントとしての CICS に対する HTTP 要求および応答の処理』では、アプリケーション・プログラムが実行する必要がある内容、およびプロセスで

CICS が実行するアクションについて説明しています。

このタスクについて

HTTP クライアントとしての CICS の場合、アプリケーション・プログラムはサーバーに要求を出し、それに対する応答を待ちます。アプリケーション・プログラムは、複数の接続を制御することができますが、これらの接続を区別するためにセッション・トークンを使用します。

HTTP 要求を行って応答を受信するには、サブトピック (サンプルが続く) に記載したプロセスに従うアプリケーション・プログラムを作成します。

HTTP サーバーへの接続のオープン

CICS Web サポートで HTTP クライアント要求を実行するときは、サーバーへの接続をオープンしてから最初の要求を送信する必要があります。 CICS は、その接続を表すセッション・トークンを戻します。

このタスクについて

サーバーへの接続を開くときに、接続のホスト・サーバーおよびポートに関する情報が含まれる URIMAP リソースを指定できます。この情報は、URIMAP リソースを使用する代わりに、アプリケーション・プログラムに直接コーディングできます。ただし、URIMAP リソースを使用すると、次のような利点があります。

- 接続のエンドポイントのどのような変更もシステム管理者が管理できるので、要求の URL が変わってもアプリケーションを再コンパイルする必要がありません。
- SSL を使用する場合は、URIMAP リソースに SSL クライアント証明書または暗号スイート・コードを指定できるので、システム管理者はこれらの証明書やコードのどのような変更も管理できます。
- URIMAP リソースで開かれた接続を使用後も開いたままにしておき、別のアプリケーションまたは同じアプリケーションの別のインスタンスが再使用できるようにプールに入れるということを、CICS に行わせることもできます。接続プーリングが使用可能になるのは、SOCKETCLOSE 属性が設定された URIMAP リソースを指定した場合のみです。接続プーリングのパフォーマンス上の利点について詳しくは、「インターネット・ガイド」の『HTTP クライアントのパフォーマンスのための接続プール』を参照してください。

WEB OPEN コマンドを発行して、サーバーとの接続を開始します。

手順

1. サーバーのホスト名、そのホスト名の長さ、および使用されるスキーム (HTTP または HTTPS) を指定します。 ホストのポート番号が指定スキームのデフォルトではない場合は、そのポート番号も指定します。 WEB OPEN コマンドで URIMAP オプションを指定すると、既存の URIMAP リソースから直接この情報を使用できます。 接続に対して接続プーリングが有効になるのは、URIMAP オプションを指定し、SOCKETCLOSE 属性が設定された URIMAP リソースを指定した場合です。 代わりに、SCHEME、HOST、HOSTLENGTH および PORTNUMBER の各オプションを使用してこの情報を指定することもできます。これらの詳細は、WEB PARSE URL コマンドを使用して既知の URL から

抜き出すか、または WEB EXTRACT URIMAP コマンドを使用して既存の URIMAP 定義から抜き出すことができます (ただし、URIMAP 定義から情報を抜き出しても接続プーリングが使用可能になるわけではありません)。

2. 必要な場合は、CODEPAGE オプションを使用して、この接続に対する EBCDIC コード・ページをローカルの CICS 領域のデフォルト・コード・ページ (LOCALCCSID システム初期設定パラメーターで設定されています) 以外のコード・ページに変更します。例えば、別の各国語の EBCDIC コード・ページなどに変更します。 サーバーが応答を返す際に変換が指定されていると、 CICS は応答ボディをこのコード・ページに変換してからアプリケーションに渡します。
3. HTTPS スキームを使用する場合は、以下のように適切なセキュリティー・オプションを指定します。
   a. SSL クライアント証明書を提供する必要がある場合は、CERTIFICATE オプションを指定します。 WEB OPEN コマンドで URIMAP オプションを指定すると、既存の URIMAP 定義から直接この情報を使用できます。
   b. CIPHERS および NUMCIPHERS オプションを使用して、接続の暗号スイート・コードのリストを指定します。 WEB OPEN コマンドで URIMAP オプションを指定する場合は、URIMAP 定義での設定を受け入れることも、独自の暗号スイート・コードをオーバーライドとして指定することもできます。
4. 最初に計画した要求に HTTP プロトコルのすべてのバージョンではサポートされていないアクションが含まれていて、そのアクションが機能することを確認するためにサーバーの HTTP バージョンを検査する場合、その情報が返されるようにするには、HTTPVNUM オプションと HTTPRNUM オプションのどちらか一方または両方を指定します。 この情報は、サーバーの HTTP バージョンがまだ分かっていないときに HTTP プロトコル・バージョンに依存するアクションをとる場合に、必要になることがあります。例えば、以下のような場合です。
   - HTTP/1.1 より前のレベルのサーバーでは正常に行われない可能性があるアクションを要求する HTTP ヘッダーの作成。
   - HTTP/1.1 より前のレベルのサーバーには適さない可能性がある HTTP メソッドの使用。
   - チャンク転送コーディングの使用。
   - 要求のパイプライン・シーケンスの送信

   サーバーの HTTP バージョンを必ずしも確認する必要はありません。使用している HTTP 仕様を調べて、誤ったバージョンのサーバーでそのアクションの試行が可能であるかどうか確認してください。例えば、一部の不適合の HTTP ヘッダーは、受信側で無視される可能性があります。確認しないで要求を試行し、サーバーからのエラー応答に対処することも可能です。 HTTPVNUM オプションおよび HTTPRNUM オプションは、パフォーマンスに影響するため、この情報が必要ない場合は指定しないでください。
5. オプション: サーバーへの接続をプロキシー・サーバー経由にする場合や、ホスト名にセキュリティー・ポリシーを適用する場合は、XWBOPEN ユーザー出口のユーザー出口プログラムを作成します。 詳細については、 165 ページの『HTTP クライアントのオープン出口 XWBOPEN』を参照してください。

タスクの結果

SOCKETCLOSE 属性を設定して URIMAP リソースを指定した場合、CICS は使用できる休止接続がプールにあるかどうかを検査し、ある場合はその休止接続をアプリケーション・プログラムに提供して再使用します。使用可能な休止接続がない場合、または適切な URIMAP リソースを指定しなかった場合、CICS はサーバーへの新規接続を開きます。アプリケーション・プログラムは、新規接続も再使用接続もまったく同じように使用します。

CICS は、アプリケーションの接続使用を表す新しいセッション・トークンを、そのアプリケーション・プログラムに返します。

次のタスク

セッション・トークンを保管し、その接続使用に関係する後続のすべてのコマンドで使用します。

要求に対する HTTP ヘッダーの書き込み

クライアントの HTTP 要求に対して、 CICS は、メッセージで使用されている HTTP プロトコル・バージョンに応じて、基本メッセージで必要な HTTP ヘッダーを自動的に提供します。ユーザーの要求に対して、さらに HTTP ヘッダーを追加することが必要になる場合があります。

このタスクについて

メッセージで HTTP ヘッダーが必要な場合、これらのヘッダーは、CICS によって自動的に作成されます。

- ARM 相関関係子
- Connection
- Content-Type (CICS によって書き込まれるが、複合ヘッダーが必要な場合はクライアント・アプリケーションで提供可能)
- Content-Length
- Date
- Expect
- Host
- サーバー
- TE (CICS によって書き込まれるが、インスタンスを追加することが可能)
- Transfer-Encoding
- User-Agent
- WWW 認証

上記の一部のヘッダーは、HTTP サーバーとしての CICS に対してのみ、該当します。上記のヘッダーが作成される条件については、415 ページの『付録 B. CICS Web サポートにおける HTTP ヘッダーの解説』で説明されています。
Content-Type ヘッダーおよび TE ヘッダーを除き、CICS 提供の要求ヘッダーのユーザー独自バージョンを書き込むことはできません。

CICS が提供する要求用ヘッダーは通常、基本的な HTTP/1.1 メッセージ用に書き込まれ、HTTP/1.1 仕様に準拠しています。 (CICS は、HTTP バージョンとして指定された HTTP/1.1 でユーザーの要求を送信します)。以下のような目的で、さらに HTTP ヘッダーの追加が必要な場合があります。

- サーバーに対する設定の指定 (例えば、Accept-Encoding、Accept-Language)
- 条件付き要求の作成 (例えば、If-Match、If-Modified-Since)
- サーバーまたはプロキシーに対する基本認証情報の提供 (Authorization、Proxy-Authorization)

ユーザーのメッセージでの使用を決定した追加の HTTP ヘッダーに関する要件について、ご使用の HTTP 仕様を確認してください。 13 ページの『HTTP プロトコル』を参照してください。

メッセージ送信のための WEB SEND コマンドを発行する*前*に、メッセージに対して追加の HTTP ヘッダーを書き込みます。ただし、チャンク化されたメッセージの末尾ヘッダーとして送信されるヘッダーを書き込む場合は、下記のような特別な処理が適用されます。以下の点に注意してください。

手順

- すべてのコマンドにおいて、SESSTOKEN オプションを使用して、その接続使用のセッション・トークンを指定します。
- メッセージに追加するヘッダーごとに WEB WRITE HTTPHEADER コマンドを使用します。 使用している HTTP 仕様に記載されている形式で、各ヘッダーの名前および値が指定されているか確認します。 このコマンドによって単一のヘッダーが追加されます。このコマンドを繰り返すと、ヘッダーをさらに追加することができます。要求に対して既に書き込まれているヘッダーを書き込むと、CICS は既存のヘッダーに追加してその要求に新規のヘッダーを追加します。ヘッダーを繰り返せるのは、ヘッダーを繰り返してもよいことが HTTP 仕様に記載されている場所のみです。 送信時に、要求に追加する準備ができているヘッダーが CICS で保管されます。
- サーバーの HTTP バージョンが不明であり、 HTTP/1.1 レベル以前のサーバーでは正常に実行できないと思われるアクションを要求するヘッダーを使用する場合は、 WEB EXTRACT コマンドを使用してそのサーバーの HTTP バージョンを確認します。 バージョンに依存するアクションを実行する前に、サーバーの HTTP バージョンを必ずしも確認する必要はありません。使用している HTTP 仕様を調べて、誤ったバージョンのサーバーでそのアクションの試行が可能であるかどうか確認してください。例えば、一部の不適合の HTTP ヘッダーは、受信側で無視される可能性があります。確認しないで要求を試行し、サーバーからのエラー応答に対処することも可能です。
- HTTP ヘッダーのいずれか (例えば、If-Modified-Since ヘッダー) で使用する日時およびタイム・スタンプを生成する場合は、FORMATTIME コマンドを使用できます。 このコマンドの STRINGFORMAT オプションでは、現在の日時 (ABSTIME 形式) またはアプリケーション・プログラムで生成された日時を Web で使用する適切な日時スタンプ形式に変換します。それ以外の日時スタンプ形式は、一部の Web クライアントまたは CICS が通信するサーバーでは受け入れられない場合があります。
- チャンク転送コーディングを使用して HTTP 要求を送信し、そのチャンク化済みメッセージの最後に末尾ヘッダーを含める場合は、 102 ページの『チャンク転送

コーディングによる HTTP 要求または応答の送信』を参照してください。 メッセージの最初のチャンクを送信する前に、Trailer ヘッダーを書き込む必要があります。最初のチャンクの WEB SEND コマンドの後に書き込まれたすべての HTTP ヘッダーは、末尾ヘッダーとして処理されます。

- ご使用のアプリケーション・プログラムで、ユーザー作成ヘッダーによって暗黙に指示されるすべてのアクションが実行されることを確認します。

HTTP 要求の作成

HTTP クライアントとしての CICS の場合、WEB SEND コマンドまたは WEB CONVERSE コマンドを使用して要求を行えます。 WEB CONVERSE コマンドでは WEB SEND コマンドおよび WEB RECEIVE コマンドで使用可能な各オプションを組み合わせているため、単一のコマンドを使用して要求を発行し、応答を受信できます。

始める前に

要求に対する追加の HTTP ヘッダーは、 144 ページの『要求に対する HTTP ヘッダーの書き込み』で説明されているように、その要求を実行する前に WEB WRITE HTTPHEADER コマンドを使用して書き込みます。

このタスクについて

要求をチャンク (チャンク転送コーディング) で送信するか、要求のパイプライン・シーケンス (148 ページの『要求のパイプライン・シーケンスの送信』を参照) を送信できます。

WEB SEND コマンドおよび WEB CONVERSE コマンドで使用可能なオプションの詳細な参照情報および説明については、「*CICS アプリケーション・プログラミング・リファレンス*」を参照してください。以下の手順に示すように WEB SEND または WEB CONVERSE コマンドを発行します。

手順

1. SESSTOKEN オプションを使用して、この接続使用のセッション・トークンを指定します。
2. 要求の HTTP メソッド (OPTIONS、GET、HEAD、POST、PUT、DELETE、または TRACE) を指定します (435 ページの『付録 D. CICS Web サポートの HTTP メソッド・リファレンス』を参照)。メソッドは、要求への対処内容をサーバーに伝えます。詳細なガイダンスについては、目的の HTTP 仕様 (13 ページの『HTTP プロトコル』) を参照してください。 CICS は要求を HTTP/1.1 で送信します。
3. サーバー上の必要なリソースのパス情報を指定します。 デフォルトは、その接続の WEB OPEN コマンドで参照した URIMAP 定義で指定されているパスです。 URIMAP オプションを使用してパスを取り出す別の URIMAP 定義を指定することにより、代わりのパスを指定できます (この新規の URIMAP 定義では、現行接続の正しいホスト名を指定する必要があります)。代わりに、PATH オプションおよび PATHLENGTH オプションを使用してパス情報を提供することもできます。

4. QUERYSTRING オプションおよび QUERYSTRLEN オプションを使用し、その要求の照会ストリングを指定します。
5. HTTP 要求のエンティティー・ボディ、およびその長さを指定します。 435 ページの『付録 D. CICS Web サポートの HTTP メソッド・リファレンス』で、要求ボディを使用することが該当するケースを示しています。 要求ボディが必要な場合は、CICS 文書からボディの内容を形成できます。CICS DOCUMENT インターフェースを使用し、DOCTOKEN オプションを指定して文書を特定します。あるいは FROM オプションを指定して、バッファーの内容からボディの内容を形成することもできます。 98 ページの『HTTP メッセージのエンティティー・ボディの作成』を参照してください。
6. MEDIATYPE オプションを使用し、提供するエンティティー・ボディのメディア・タイプを指定します。 ボディが必要である POST メソッドおよび PUT メソッドによる要求の場合は、 MEDIATYPE オプションを指定する必要があります。ボディの内容がないその他のメソッドを指定した要求の場合、MEDIATYPE オプションは必要ありません。
7. 要求ボディに対してコード・ページ変換が必要ない場合は、対応する変換オプション (WEB SEND コマンドを使用しているか WEB CONVERSE コマンドを使用しているかに依存する) を指定し、CICS が要求ボディを変換しないようにします。 HTTP クライアントとしての CICS の場合、非テキストのメディア・タイプでない限り、デフォルト設定では要求ボディが変換されます。
8. コード・ページ変換が必要であり、デフォルトの ISO-8859-1 文字セットが適切でない場合は、そのサーバーに適合する文字セットを指定します。 ISO-8859-1 は、ほとんどのサーバーで許容されます。
9. サーバーが要求を受け入れるかどうかを Expect ヘッダーを使用してテストする場合は、ACTION オプションに EXPECT を指定します。 この設定にすると、CICS は要求の要求行とヘッダーとともに Expect ヘッダーを送信し、100-Continue 応答を待機してからサーバーにメッセージ・ボディを送信します。 100-Continue 以外の応答を受信した場合、CICS はアプリケーション・プログラムに通知し、その送信を取り消します。待機の期限内に応答がない場合、CICS はメッセージ・ボディを送信します。 Expect ヘッダーは、HTTP/1.1 より下位のサーバーではサポートされていません。 CICS がサーバーの HTTP バージョンをまだ認識していない場合、CICS は要求を送信する前にバージョン番号を要求します。 Expect ヘッダーが適合しない場合、CICS はこのヘッダーなしで要求を送信します。
10. オプション: この要求がこのサーバーに対して行う最後の要求である場合は、接続プーリングを使用しているかどうかによって、接続を閉じるようサーバーに要求することが必要になることもあります。
    - この接続に接続プーリングを使用していない場合は、CLOSESTATUS オプションに CLOSE を指定できます。このオプションを使用すると、CICS は要求に Connection: close ヘッダーを書き込みます。HTTP/1.0 レベルのサーバーの場合は、Connection: Keep-Alive ヘッダーを省略します。このオプションを指定することは、サーバーが最終応答の送信直後にその接続を閉じることができるということを意味します。この動作は Web クライアントの要件ではありませんが、接続を再使用できるようにしておくことを絶対にしたくない場合は、ベスト・プラクティスです。

- この接続に接続プーリングを使用している場合は、CLOSESTATUS オプションを指定しないでください。CLOSESTATUS(CLOSE) を指定すると、サーバーが接続を閉じるので、接続をプールできなくなってしまいます。

SOCKETCLOSE 属性が設定された URIMAP リソースを使用して接続を開くと、接続プーリングが使用可能になります。

11. チャンク転送コーディングを使用して一連のチャンクとして要求ボディを送信する場合は、 102 ページの『チャンク転送コーディングによる HTTP 要求または応答の送信』に記載された追加の説明に従ってください。 以下の環境では、チャンク転送コーディングはサポートされません。
    - HTTP/1.1 より前のサーバー
    - WEB CONVERSE コマンド。
    - CICS 文書 (DOCTOKEN オプション)。

タスクの結果

CICS は、要求の行、HTTP ヘッダー、および要求ボディをアセンブルし、その要求をサーバーに送信します。

要求のパイプライン・シーケンスの送信

サーバーからの応答を待つことなく、さらに要求を送信することができます。この技法はパイプライン化と呼ばれます。パイプライン要求の送信には WEB SEND コマンドが使用されます。 WEB CONVERSE コマンドは使用できません。このコマンドには応答の待機が含まれているためです。

このタスクについて

45 ページの『CICS Web サポートによるパイプライン化の処理方法』で、詳しく説明しています。

HTTP/1.1 仕様には、パイプライン要求のシーケンスは「べき等」でなければならないと記述されています。つまり、シーケンスのすべてまたは一部を繰り返しても、同じ結果が得られます。べき等について詳しくは、 21 ページの『パイプライン化』を参照してください。

手順

1. サーバーとの持続接続があることを確認します。 HTTP/1.1 仕様では、その接続が永続的であるか確認することなくパイプライン・シーケンスを送信する試行が可能です。この試行に失敗した場合は、要求を再試行する前に確認する必要があります。その接続の性質を判別するには、以下のようにします。
    a. その接続に対して WEB OPEN コマンドで HTTPVNUM オプションおよび HTTPRNUM オプションを指定した場合は、戻された情報を検査してそのサーバーの HTTP バージョンを判別します。
    b. WEB OPEN コマンドでこれらのオプション指定していない場合は、 WEB EXTRACT コマンドを使用してそのサーバーの HTTP バージョンを判別します。

c. サーバーから既に前の応答を受信している場合は、 WEB READ HTTPHEADER コマンドを使用して、サーバーが Connection: close または Connection: Keep-Alive のヘッダーを送信したか確認します。

サーバーが HTTP/1.1 レベルであり Connection: close ヘッダーを送信していない場合、およびサーバーが HTTP/1.0 レベルであり Connection: Keep-Alive ヘッダーを送信している場合は、持続接続がサポートされています。 この検査を CICS が自動的に行うわけではありません。パイプライン要求にはそれを識別するための特別なヘッダーがないので、クライアント・アプリケーション・プログラムが要求のパイプライン・シーケンスを送信しているかどうかを CICS が判別できないためです。

2. 接続プーリングを使用していない場合、サーバーが接続を閉じるようにするには、パイプライン・シーケンスの最終要求以外は、どの要求でも CLOSESTATUS(CLOSE) を指定しないでください。 接続プーリングを使用している場合は、CLOSESTATUS(CLOSE) を一切指定しないでください。

基本認証のための資格情報の提供

HTTP 401 WWW-Authenticate メッセージを受信した場合、アプリケーションは、サーバーが基本認証を行うのに必要なユーザー名とパスワード (資格情報) を提供する必要があります。アプリケーションは、401 メッセージを待たなくても、これらの資格情報を提供することもできます。

手順

1. WEB OPEN コマンドで SESSTOKEN オプションを使用して、サーバーとの Web セッションを開きます。 セッションが正常に開くと SESSTOKEN が返されるので、このセッション・トークンをこの接続に関係するすべての CICS WEB コマンドで使用する必要があります。
2. WEB SEND コマンドを発行して、この接続使用の SESSTOKEN を指定します。 この WEB SEND コマンドにより、サーバーからレルムが取得されます。
3. WEB RECEIVE コマンドを発行します。サーバーは状況コードを返します。WEB RECEIVE コマンドで STATUSCODE オプションを使用して、401 応答について確認します。
4. 状況コードが 401 の場合は (サーバーが認証の詳細を要求しています)、最初の WEB SEND 要求を繰り返してください。ただし、今回は AUTHENTICATE (BASICAUTH) オプションを追加します。クライアント・アプリケーションによって XWBAUTH グローバル・ユーザー出口が呼び出されます。 この 2 回目の WEB SEND コマンドでは、最初の WEB SEND コマンドから受信したレルムと XWBAUTH 出口を使用して、要求されたユーザー名とパスワードが決定されます。
5. 最初の WEB SEND コマンドでは、401 応答を待つ代わりに、AUTHENTICATE(BASICAUTH) を指定することもできます。次の選択肢があります。
   - WEB SEND コマンドで AUTHENTICATE(BASICAUTH) オプションを使用して、ユーザー名とパスワードを提供する。
   - AUTHENTICATE(BASICAUTH) オプションを指定して XWBAUTH グローバル・ユーザー出口を呼び出す。ただし、資格情報は省略します。ユーザー出口が呼び出されますが、出口に渡されるレルムは空です。レルムがサーバーから

まだ受信されていないためです。このユーザー出口は、要求された資格情報を、他のパラメーター (例えば HOST と PATH) から取り出す必要があります。

6. 401 応答で送信されたレルムをアプリケーションで認識する必要がある場合は、WEB EXTRACT コマンドを使用してください。

タスクの結果

CICS は、Authentication ヘッダーでユーザー名とパスワードの資格情報をサーバーに渡します。

HTTP 応答の受信

サーバーから応答を受信するには、WEB RECEIVE コマンドまたは WEB CONVERSE コマンドを使用します。ヘッダーを検査するには、WEB READ HTTPHEADER コマンドまたは HTTP ヘッダー・ブラウズ・コマンドを使用します。

始める前に

応答の受信でアプリケーションに準備された待機時間は、別名トランザクション用のトランザクション・プロファイル定義で指定された RTIMOUT 値によって示されます。このタイムアウト限界は、応答のヘッダーの読み取りには適用されません。

RTIMOUT で指定した期限が過ぎると、CICS からアプリケーションに TIMEDOUT 応答が戻されます。 RTIMOUT 値がゼロである場合、アプリケーションは無限に待機するよう準備されます。トランザクション・プロファイル定義における RTIMOUT のデフォルト設定はゼロであるため、設定を確認し変更することが重要です。

このタスクについて

HTTP 応答を受信するには、WEB RECEIVE または WEB CONVERSE コマンドを以下のように使用します。

手順

1. SESSTOKEN オプションを使用して、この接続使用のセッション・トークンを指定します。
2. サーバーが返す HTTP 状況コードを受信するデータ域、および状況コードの説明でサーバーから戻されるテキストを受信するデータ域を指定します。 データは、そのアンエスケープ形式で返されます。
3. 応答ボディのメディア・タイプを受信するデータ域を指定します。
4. INTO オプション (データ・バッファーの場合) または **SET** オプション (ポインター参照の場合)、および LENGTH オプションを指定することにより、応答ボディを受信します。 このデータは、そのエスケープ形式で戻され、特に要求した場合を除き、アプリケーションに適合するコード・ページに変換されます。
5. 応答ボディから受信するデータ量を制限する場合は、MAXLENGTH オプションを指定します。その長さを超過したデータを廃棄しないで保存する場合は、NOTRUNCATE オプションも加えて指定します。 残りのデータは、WEB RECEIVE コマンドをさらに使用して取得できます。 チャンク転送コーディング

を使用してデータが送信されている場合、CICS は、アプリケーションへの引き渡し前に、チャンクを単一メッセージにアセンブルします。したがって、MAXLENGTH オプションは、各チャンクに個別に適用されるのではなく、チャンク化されたメッセージのエンティティー・ボディの合計の長さに適用されます。

6. 応答ボディのコード・ページ変換が必要ない 場合は、CICS が応答ボディを変換しないように、適切な変換オプションを指定します。 デフォルトでは、変換が行われます。その場合、CICS は、サーバー応答のボディをローカル CICS 領域のデフォルト・コード・ページに変換するか、WEB OPEN コマンドで指定された代替 EBCDIC コード・ページに変換します。

   注: そのメッセージの Content-Encoding のヘッダーで示されているように、圧縮されているエンティティー・ボディを受信した場合は、コード・ページ変換を抑止したことを確認してください。 CICS では、このようなタイプのメッセージはデコードされません。したがって、コード・ページ変換が適用された場合、その結果は予測不能です。圧縮されたエンティティー・ボディを受信したくない場合は、サーバーへの要求で Accept-Encoding ヘッダーを使用します。

   WEB RECEIVE または WEB CONVERSE コマンドを発行した場合、 CICS は、応答ボディ、および状況表示行からの情報を戻します。

7. サーバー応答の HTTP ヘッダーを、以下のように検査します。
   - サーバーが提供した、既知である特定の HTTP ヘッダーを読み取るには、WEB READ HTTPHEADER コマンドを使用してそのヘッダーの内容を検査します。アプリケーション・プログラムは、ヘッダーの内容を受信するためのバッファーを提供する必要があります。該当するヘッダーが要求に存在しない場合、 CICS から NOTFND 条件が返されます。
   - 応答内のすべての HTTP ヘッダーをブラウズする場合は、 WEB STARTBROWSE HTTPHEADER コマンドを使用してヘッダー行のブラウズを開始します。各行のヘッダー名およびヘッダー値を取得するには、 WEB READNEXT HTTPHEADER コマンドを使用します。アプリケーション・プログラムは、2 つのバッファーを提供する必要があります。1 つはヘッダーの名前を受信するためのもの、もう 1 つはその内容を受信するためのものです。CICS は、すべてのヘッダーの読み取りが完了すると ENDFILE 状態を返します。関係のあるヘッダー情報をプログラムがすべて取得した後で、WEB ENDBROWSE HTTPHEADER コマンドを使用します。

   HTTP ヘッダー・コマンドのそれぞれにセッション・トークンを含める必要があることに注意してください。

8. サーバーの応答を処理し、アプリケーションのビジネス・ロジックを実行します。 その応答が「正常」または通知の状況コード (200 (OK) など) であった場合は、正常にその応答を処理できます (状況コードは、WEB RECEIVE コマンドを発行した場合に受信されます)。エラーを示している状況コードやさらにアクションを要求している状況コードが応答にある場合は、代わりの処理を実行する必要があります。 423 ページの『付録 C. CICS Web サポートの HTTP 状況コード・リファレンス』に、状況コードへの対処に関する基本的なガイダンスを記載しています。

9. パイプライン化された要求のシーケンスを送信した場合は、さらに WEB RECEIVE コマンドを使用して、サーバーから応答の残りを受信します。 CICS

は各応答を保持し、 CICS がサーバーから受信した順序で、応答をアプリケーション・プログラムに返します。パイプライン要求を処理するサーバーでは、各要求が受信された同じ順序で応答を提供します。

ヒント: パイプライン要求に対する応答を受信するときに、複数の WEB RECEIVE コマンドを使用して長さが超過したメッセージ・ボディを受信する場合は、発行した WEB RECEIVE コマンドの数を注意深く追跡してください。各応答のそれぞれに対するボディ全体を単一の WEB RECEIVE コマンドで受信した方が便利である場合もあります。

HTTP サーバーとの接続のクローズ

CICS が HTTP クライアントである場合、CICS とサーバー間の接続は、サーバーが閉じるか、またはアプリケーション・プログラムによって発行されたコマンドの後に CICS が閉じることができます。 接続プーリングを使用している場合、CICS は適切な接続を閉じるのではなく、再使用できるように休止接続のプールで保持します。

このタスクについて

デフォルトでは、CICS とサーバー間の接続は、サーバーの要求で、またはアプリケーション・プログラムが接続を使い終わった後に閉じられます。ただし、接続プーリングが指定された URIMAP リソースを使用して接続が開かれていて、サーバーもアプリケーション・プログラムも接続を閉じる要求を行っていなければ、CICS は接続を閉じません。代わりに、CICS は接続が良好な状態であるかどうかを検査したうえで、それを休止接続のプールに入れます。プールされた接続は開かれたままなので、別のアプリケーション・プログラム、または同じアプリケーション・プログラムの別のインスタンスが再使用して、同じサーバーに接続することができます。

手順

1. サーバーが接続の終了を要求したかどうかを接続の使用中にテストする場合は、WEB READ HTTPHEADER コマンドを使用して、サーバーからの最後のメッセージに Connection: close ヘッダーがないか探します。 サーバーが接続を閉じると、アプリケーション・プログラムはその接続による以降の要求は送信できなくなりますが、サーバーが接続を閉じる前に送信した応答は受信できます。
2. オプション: 接続プーリングを使用していない場合は、サーバーに対して行う最後の要求時に、WEB SEND または WEB CONVERSE コマンドの CLOSESTATUS オプションに CLOSE を指定します。 CICS は要求に Connection: close ヘッダーを書き込みます。または HTTP/1.0 レベルのサーバーの場合、Connection: Keep-Alive ヘッダーを省略します。このオプションを指定することは、サーバーはさらに要求があるかどうかをタイムアウトまで待機するのではなく、最終応答の送信直後にその接続を閉じることができるということを意味します。 CLOSESTATUS オプションを指定した場合は、その効果の範囲は、WEB CLOSE コマンドを発行した場合とは異なります。

   注: 接続プーリングを使用している場合は、CLOSESTATUS オプションに CLOSE を指定しないでください。 CLOSESTATUS(CLOSE) を指定すると、サーバーが接続を閉じるので、接続をプールに入れることができなくなってしまいます。

3. HTTP 要求および応答がすべて完了すると、セッション・トークンを指定して WEB CLOSE コマンドを発行します。 WEB CLOSE コマンドを発行すると、アプリケーション・プログラムが接続を使い終わったことが CICS に通知されます。このコマンドを発行すると、その接続使用に適用されるセッション・トークンは無効になり、使用できなくなります。サーバーからの応答を受信し、その応答の HTTP ヘッダーを読み取るには、セッション・トークンが必要です。したがって、サーバーおよびサーバーが送信した最後の応答との対話がすべて完了するまでは、WEB CLOSE コマンドを発行しないでください。
   - WEB CLOSE コマンドを発行したときに接続がまだ開いていて、接続プーリングが指定された URIMAP リソースを使用してその接続が開かれていた場合、CICS は接続を閉じません。CICS は接続の状態を検査し、再使用できるようにプールに入れます。
   - WEB CLOSE コマンドを発行したときに接続が接続プーリングに適していない場合 (接続を閉じる要求をサーバーまたはアプリケーション・プログラムが既に行っていたり、適切な URIMAP リソースを使用して接続が開かれていなかったり、あるいは接続が良好な状態でなかったりするため)、CICS は接続を閉じ、プールには入れません。
   - WEB CLOSE コマンドを発行しない場合、CICS はタスクの終わりに接続を閉じます。閉じられた接続をプールに入れることはできません。接続プーリングを使用可能にするには、アプリケーションが WEB CLOSE コマンドを発行する必要があります。

サンプル・プログラム: HTTP サーバーへの要求のパイプライン化

サンプル・プログラム DFH$WBPA (アセンブリ言語)、DFH$WBPC (C)、および DFH0WBPO (COBOL) は、CICS で HTTP サーバーへのクライアント要求をどのようにしてパイプライン化できるかを示します。

始める前に

サンプル・プログラムは、CICS Web サポートを実行している CICS 領域に要求を送信します。要求は、CICS 提供のサンプル・プログラムである DFH$WB1C によって処理されます。サンプル・プログラムを使用する前に、65 ページの『第 4 章 CICS Web サポートのコンポーネントの構成』の手順に従って、 CICS 領域を HTTP サーバーとしてセットアップする必要があります。サンプル・プログラム DFH$WB1C をセットアップし、サンプル URIMAP 定義の DFH$URI1 を変更して手順を完了します (68 ページの『CICS Web サポートのオペレーションの検査』を参照)。CICS 領域が既に HTTP サーバーとしてセットアップされて運用されていて、独自の TCPIPSERVICE 定義がある場合は、サンプル TCPIPSERVICE 定義の HTTPNSSL を再度インストールしないでください。DFH$WB1C と DFH$URI1 のみをセットアップしてください。

このタスクについて

CICS 領域を HTTP サーバーとしてセットアップしたら、パイプライン化のサンプル・プログラムを使用するため以下の手順を実行します。

手順

1. HTTP クライアントとなる CICS 領域を識別します。 サンプル・プログラムを試行するには、以下の 3 つのオプションがあります。
   - 同一の CICS 領域をサーバーとクライアントの両方として使用する。この場合、要求は 2 つの異なる CICS 領域で送受信されるときと同様に 1 つの領域で送受信され、結果も同じです。この場合、追加の CICS Web サポートのセットアップは不要です (HTTP サーバーとして作動する CICS 領域は、HTTP クライアントとしても作動することができるため)。
   - CICS Web サポート用にすでにセットアップされている別の CICS 領域をクライアントとして使用します。この場合も、追加の CICS Web サポートのセットアップは不要です。
   - CICS Web サポート用にまだセットアップされていない別の CICS 領域をクライアントとして使用します。この場合は、CICS Web サポートの基本的なセットアップを実行する必要があります (ステップ 2 を参照)。
2. オプション: 別の CICS 領域を HTTP クライアントとして使用する予定で、その領域が CICS Web サポート用にまだセットアップされていない場合は、基本セットアップを実行します。
   a. 「*CICS Transaction Server for z/OS インストール・ガイド*」の手順に従って、CICS 領域の TCP/IP サポートを有効にします。 この処理には、Communications Server のセットアップ、および z/OS を介した DNS (ドメイン・ネーム) サーバーへのアクセスの確立が含まれます。
   b. 領域のシステム初期設定パラメーター **TCPIP=YES** を指定して、CICS TCP/IP サービスをアクティブにします。

   このセットアップにより、CICS 領域が HTTP クライアントとして機能します。
3. HTTP サーバーとしてセットアップした CICS 領域で、クライアント領域が要求を行うために使用できるポートの TCPIPSERVICE 定義を識別します。HTTP プロトコルで定義されているが、SSL を使用しない任意のポート、つまり PROTOCOL(HTTP) と SSL(NO) を指定する TCPIPSERVICE 定義を選択します。サンプル・プログラム DFH$WB1C にアクセスするためにサーバーで使用されるサンプル URIMAP 定義である DFH$URI1 は、任意のホスト名とポート番号に一致するため、任意の適切なポートを選択できます。
4. HTTP クライアント領域で、提供されているサンプル URIMAP 定義 DFH$URI2 (DFH$WEB リソース定義グループに提供されています) を変更します。 DFH$WEB は保護されたグループであるため、編集できるようにするには、定義を別のグループにコピーします。 DFH$URI2 は、使用法の属性が CLIENT である URIMAP 定義です。この定義は、サンプル・プログラムで HTTP サーバー領域に対する要求を行うために使用する URL の構成要素を指定します。以下のステップを実行して、サンプルを変更します。
   a. DFH$URI2 で HTTP として指定されたスキーム (SCHEME 属性) を変更しないでください。
   b. DFH$URI2 で指定されているホスト (HOST 属性) は、ダミーのホスト名です。この名前を変更して実際のホスト名を挿入します。

- HTTP サーバー領域の z/OS イメージに割り当てられたホスト名を指定します。ホスト名が不明な場合は、ステップ 3 で選択した TCPIPSERVICE 定義にあるコロン 16 進数またはドット 10 進数の IP アドレスを使用できます。
- 選択した TCPIPSERVICE 定義が 80 (HTTP 用のウェルノウン・ポート番号) 以外の番号のポートで使用されている場合は、TCPIPSERVICE 定義にあるポート番号をホスト名の後に指定します (ホスト名とポート番号はコロンで区切ります)。

c. DFH$URI2 で /sample_web_app (DFH$URI1 と一致) として指定されたパス (PATH 属性) を変更しないでください。

5. HTTP クライアント領域で、変更済みの URIMAP 定義 DFH$URI2 をインストールします。
6. HTTP クライアント領域に、DFH$WEB リソース定義グループに提供されている PROFILE 定義 DFH$WBPF をインストールします。
7. いずれかのサンプル・プログラムを使用する言語に変換してコンパイルします。 パイプライン化のサンプル・プログラムは、ユーザーが受け取った時点ではコンパイルされていません。サンプル・プログラムは SDFHSAMP ライブラリーに提供されています。サンプル・プログラムの名前およびそれぞれの対応するトランザクションは、以下のとおりです。

言語	プログラム	トランザクション
アセンブラー	DFH$WBPA	WBPA
C	DFH$WBPC	WBPC
COBOL	DFH0WBPO	WBPO

8. HTTP クライアント領域に、選択したサンプル・プログラム用の PROGRAM リソース定義および対応する TRANSACTION リソース定義をインストールします。 リソース定義は DFH$WEB リソース定義グループに提供されています。
9. HTTP クライアント領域で、選択したサンプル・プログラムのトランザクションを実行します。 サンプル・プログラムによって 3 つの HTTP 要求がそれぞれ正常に送受信されて処理が完了すると、端末に情報メッセージが送信されます。いずれかの送信または受信が失敗した場合は、代わりにエラー・メッセージが表示されます。 HTTP 要求および応答の内容は表示されません。
10. サンプル・プログラムを使用し終えたら、セキュリティーのために、URIMAP 定義の DFH$URI1 および DFH$URI2 を無効にし、サンプル TCPIPSERVICE 定義の HTTPNSSL を使用した場合は、この定義をアンインストールします。

関連概念

21 ページの『パイプライン化』
パイプライン化では、クライアントは複数の HTTP 要求を、応答を待たずにサーバーに送信します。応答は、要求が受信されたのと同じ順序でサーバーから返す必要があります。

45 ページの『CICS Web サポートによるパイプライン化の処理方法』
パイプライン化された要求シーケンスを CICS によって送受信することができます。HTTP サーバーとしての CICS は、Web クライアントからのパイプライン化された要求シーケンスを受信することができ、HTTP クライアントとしての CICS は、パイプライン化された要求シーケンスをサーバーに送信することができます。

関連タスク

68 ページの『CICS Web サポートのオペレーションの検査』
CICS Web サポートの動作テストをサポートするために、サンプル・プログラム DFH$WB1A (アセンブラー) および DFH$WB1C (C 言語) が提供されています。サンプル・プログラムは **EXEC CICS WEB** および **EXEC CICS DOCUMENT** コマンドを使用して要求を受け取り、単一の応答を作成して送信します。

サンプル・プログラム: チャンク単位での HTTP 要求の送信と受信

サンプル・プログラム DFH$WBCA (アセンブリ言語)、DFH$WBCC (C)、および DFH0WBCO (COBOL) は、HTTP クライアントとしての CICS が HTTP サーバーにセクションまたはチャンク単位での送受信を行う方法、およびチャンク化されたメッセージを応答で受信する方法を示します。サンプル・プログラム DFH$WBHA (アセンブリ言語)、DFH$WBHC (C)、および DFH0WBHO (COBOL) は、HTTP サーバーとしての CICS が HTTP クライアントからチャンク単位で要求を受信する方法、およびチャンク化された応答を送信する方法を示します。

始める前に

サンプル・プログラムは、CICS Web サポートが実行している CICS 領域の間で送受信を行います。クライアント側チャンク化のサンプルである DFH$WBCA、DFH$WBCC、および DFH0WBCO は HTTP クライアント領域にインストールされ、サーバー側チャンク化のサンプルである DFH$WBHA、 DFH$WBHC、および DFH0WBHO は HTTP サーバー領域にインストールされます。例えば、クライアント・サンプル DFH$WBCA は、対応するサーバー・サンプル DFH$WBHA とのセッションをオープンします。DFH$WBHA は DFH$WBCA からチャンク化された要求を受信し、チャンク化された応答を送信します。クライアント・サンプルである DFH$WBCA は、チャンク化されたメッセージとして応答を受信します。

サンプル・プログラムを使用する前に、 65 ページの『第 4 章 CICS Web サポートのコンポーネントの構成』の手順に従って、 CICS 領域を HTTP サーバーとしてセットアップする必要があります。CICS 領域が既に HTTP サーバーとしてセットアップされて運用されていて、適切に設計された独自の TCPIPSERVICE 定義がある場合は、サンプル TCPIPSERVICE 定義の HTTPNSSL を再度インストールしないでください。

このタスクについて

CICS 領域を HTTP サーバーとしてセットアップした場合、チャンク化サンプル・プログラムを使用するには以下の手順を実行してください。

手順

1. HTTP クライアントとなる CICS 領域を識別します。 サンプル・プログラムを試行するには、以下の 3 つのオプションがあります。
   - 同一の CICS 領域をサーバーとクライアントの両方として使用します。その場合、要求は 2 つの異なる CICS 領域で送受信されるときと同様に 1 つの領域で送受信され、結果も同じです。この場合、追加の CICS Web サポートのセットアップは不要です (HTTP サーバーとして作動する CICS 領域は、HTTP クライアントとしても作動することができるため)。
   - CICS Web サポート用にすでにセットアップされている別の CICS 領域をクライアントとして使用します。この場合も、追加の CICS Web サポートのセットアップは不要です。
   - CICS Web サポート用にまだセットアップされていない別の CICS 領域をクライアントとして使用します。この場合は、CICS Web サポートの基本的なセットアップを実行する必要があります (ステップ 2 を参照)。
2. オプション: 別の CICS 領域を HTTP クライアントとして使用する予定で、その領域が CICS Web サポート用にまだセットアップされていない場合は、基本セットアップを実行します。
   a. 「*CICS Transaction Server for z/OS インストール・ガイド*」の手順に従って、CICS 領域の TCP/IP サポートを有効にします。 この処理には、Communications Server のセットアップ、および z/OS を介した DNS (ドメイン・ネーム) サーバーへのアクセスの確立が含まれます。
   b. 領域のシステム初期設定パラメーター TCPIP=YES を指定して、CICS TCP/IP サービスをアクティブにします。

   このセットアップにより、CICS 領域が HTTP クライアントとして機能します。
3. HTTP サーバーとしてセットアップした CICS 領域で、クライアント領域が要求を行うために使用できるポートの TCPIPSERVICE 定義を識別します。PROTOCOL(HTTP) と SSL(NO) を指定した PROTOCOL(HTTP) 定義のように、HTTP プロトコルで定義されているが、SSL を使用しない、任意のポートを選択します。サーバー側チャンク化サンプル・プログラムにアクセスするためにサーバーで使用されるサンプル URIMAP 定義である DFH$URI4 は、任意のホスト名とポート番号に一致するため、任意の適切なポートを選択できます。
4. HTTP クライアント領域で、提供されているサンプル URIMAP 定義 DFH$URI3 (DFH$WEB リソース定義グループに提供されています) を変更します。DFH$WEB は保護されたグループであるため、編集できるようにするには、定義を別のグループにコピーします。 DFH$URI3 は、使用法の属性が CLIENT である URIMAP 定義です。この定義は、サンプル・プログラムで HTTP サーバー領域に対する要求を行うために使用する URL の構成要素を指定します。

a. DFH$URI3 で HTTP として指定されたスキーム (SCHEME 属性) を変更しないでください。

b. DFH$URI3 は、ダミー・ホスト名 (HOST 属性) を指定しています。これを以下のように変更して実際のホスト名を挿入します。

- HTTP サーバー領域の z/OS イメージに割り当てられたホスト名を指定します。ホスト名が不明な場合は、ステップ 3 で選択した TCPIPSERVICE 定義にある IP アドレスを使用できます。
- 選択した TCPIPSERVICE 定義が、HTTP 用のポート番号として一般的に使用される 80 以外のポート番号を使用している場合は、TCPIPSERVICE 定義にあるポート番号をホスト名の後に指定します (ホスト名とポート番号をコロンで区切ります)。

c. DFH$URI3 で /chunking_sample_application (DFH$URI4 と一致) として指定されたパス (PATH 属性) を変更しないでください。

5. HTTP クライアント領域に、変更した URIMAP 定義 DFH$URI3 をインストールします。
6. HTTP クライアント領域に、DFH$WEB リソース定義グループに提供されている PROFILE 定義 DFH$WBPF をインストールします。
7. クライアントおよびサーバー用の必要な言語のサンプル・プログラムを変換してコンパイルします。 チャンク化のサンプル・プログラムは、ユーザーが受け取った時点ではコンパイルされていません。サンプル・プログラムは SDFHSAMP ライブラリーに提供されています。サンプル・プログラムの名前および対応するトランザクションを、以下に示します。

サンプルのタイプ	言語	プログラム	トランザクション
クライアント側チャンク化サンプル	アセンブリー	DFH$WBCA	WBCA
クライアント側チャンク化サンプル	C	DFH$WBCC	WBCC
クライアント側チャンク化サンプル	COBOL	DFH0WBCO	WBCO
サーバー側チャンク化サンプル	アセンブリー	DFH$WBHA	-
サーバー側チャンク化サンプル	C	DFH$WBHC	-
サーバー側チャンク化サンプル	COBOL	DFH0WBHO	-

8. HTTP サーバー領域に、選択したサーバー側チャンク化サンプル・プログラム用の PROGRAM リソース定義をインストールし、提供されているサンプル URIMAP 定義 DFH$URI4 をインストールします。 リソース定義は DFH$WEB リソース定義グループに提供されています。

a. C または COBOL のサンプルを選択した場合は、提供されたサンプル URIMAP 定義 DFH$URI4 を変更してからインストールします。 DFH$WEB は保護されたグループであるため、編集できるようにするには、定義を別のグループにコピーします。

b. DFH$URI4 で指定されているプログラム (PROGRAM 属性) を、DFH$WBHA (アセンブリ言語のチャンク化サンプル・プログラム) から、優先するサーバー側チャンク化サンプル・プログラムに変更します。

c. 変更した URIMAP 定義 DFH$URI4 をインストールします。

9. HTTP クライアント領域に、選択したクライアント側チャンク化サンプル・プログラム用の PROGRAM リソース定義および対応する TRANSACTION リソース定義をインストールします。 リソース定義は DFH$WEB リソース定義グループに提供されています。

10. HTTP クライアント領域で、選択したクライアント側チャンク化サンプルのトランザクションを実行します。 サンプル・プログラムからメッセージ・チャンク 4 個とヘッダー・トレーラー 2 個がすべて HTTP サーバーに正常に送信されると、端末に情報メッセージが送信されます。受信が行われたことを確認するメッセージも表示されます。いずれかの送信が失敗した場合は、代わりにエラー・メッセージが表示されます。実際の HTTP 要求および応答の内容は表示されません。

11. HTTP サーバー領域で、選択したサーバー側チャンク化サンプルが、対応するクライアント側チャンク化サンプルによって呼び出されます。 サンプル・プログラムからメッセージのチャンク 4 個とヘッダー・トレーラー 2 個がすべて待機している HTTP クライアントに正常に送信されると、端末に情報メッセージが送信されます。いずれかの送信が失敗した場合は、代わりにエラー・メッセージが表示されます。実際の HTTP 要求および応答の内容は表示されません。

12. サンプル・プログラムを使い終えたら、セキュリティー上の理由から、URIMAP 定義の DFH$URI3 と DFH$URI4 を無効にし、サンプル TCPIPSERVICE 定義の HTTPNSS を使用した場合は、それをアンインストールします。

HTTP クライアントとしての CICS のための URIMAP リソースの作成

HTTP クライアント要求の URI の構成要素 (スキーム、ホスト、パス) と、要求で使用される SSL クライアント証明書 (必要な場合) を指定した URIMAP リソースを作成できます。URIMAP リソースを使用してクライアント HTTP 接続を開く場合、管理者はサーバーの URI のどのような変更も管理できます。また、他のアプリケーションが再使用できるように、開かれた接続を使用後にプールすることを指定できます。

このタスクについて

WEB OPEN コマンドで URIMAP リソースを指定することにより、接続のスキームとホスト名、およびデフォルト・パスを提供できます。 **WEB SEND** コマンドでリソースを指定して、該当する要求のパスを提供することもできます。さらにまた、**WEB EXTRACT URIMAP** コマンドを使用して URIMAP リソースから情報を抽出し、HTTP クライアント要求を作成するアプリケーション・プログラムで直接、この情報を使用することもできます。URIMAP リソースを **INVOKE SERVICE** コマンドで使用してサービスを呼び出すこともできます。

デフォルトでは、CICS アプリケーションがそのクライアント HTTP 接続を使い終わると、CICS は接続を閉じます。URIMAP リソースで SOCKETCLOSE 属性を指

定すると、CICS は接続を閉じる代わりに、その接続を休止状態でプールに入れることができます。休止接続は、同じアプリケーションが再使用することも、同じホストとポートに接続する別のアプリケーションが再使用することも可能です。これにより、新しい接続を開かずに済みます。アプリケーション・プログラムは、新しい接続を使用するのとまったく同じようにプール接続を使用します。クライアント HTTP 接続を接続プーリングの対象にするには、CICS アプリケーション・プログラムが **INVOKE SERVICE** または **WEB OPEN** コマンドで URIMAP リソースを指定する必要があります。

手順

1. HTTP クライアント要求用に使用する予定の URL を識別します。 URL は、アクセスする予定のサーバー上のリソースを表します。
2. その要求用にクライアント証明書が必要かどうかを識別し、適切な証明書ラベルを取得します。 要求に使用されているスキームが HTTPS である場合は、サーバーが SSL クライアント証明書を要求する可能性があります。その場合には、URIMAP リソースで指定されている証明書ラベルを CICS が提供します。
3. 要求の URL を、スキーム、ホスト、およびパスの構成要素に分割します。 11 ページの『URL の構成要素』を参照してください。URL で明示的にポート番号が指定されている場合は、そのポート番号も使用します。 例えば、URL `http://www.example.com:1030/software/index.html` の場合は以下のようになります。
   - スキーム構成要素は `http`
   - ホスト構成要素は `www.example.com`
   - ポート番号は `1030`
   - パス構成要素は `/software/index.html`

   要求用に URL で照会ストリングを提供する場合は、WEB SEND コマンドで QUERY オプションを使用することで、照会ストリングを指定することができます。
4. 任意の名前とグループを指定した URIMAP リソースを、以下のように作成します。
   a. CICS は HTTP クライアントなので、USAGE 属性に CLIENT を指定します。
   b. SCHEME 属性に、要求の URL のスキーム構成要素を指定します。HTTP または HTTPS を使用します。 スキーム構成要素の後に区切り文字 `://` を含めないでください。
   c. 要求の URL のホスト構成要素として、HOST 属性を指定します。 ホスト構成要素として、明示的な IPv4 または IPv6 アドレス、あるいは文字ホスト名を使用できます。 サーバーへの要求の URL にポート番号を指定する場合は、HOST 属性に含めます (ポート番号の前にコロンを使用)。ポート番号を指定する必要があるのは、ポート番号がスキームのデフォルトでない場合のみです。デフォルトは、SSL を使用しない HTTP の場合は 80、HTTPS (SSL を使用する HTTP) の場合は 443 です。
   d. 要求の URL のパス構成要素として、PATH 属性を指定します。

- パス構成要素に照会ストリングを含めないでください。**WEB SEND** コマンドで QUERY オプションを使用することで、照会ストリングを指定できます。
- CICS が HTTP クライアントの場合は、URIMAP リソースにワイルドカード文字 (アスタリスク) を使用しないでください。
- パス構成要素の先頭にスラッシュを含めても省略してもかまいません。省略しても、CICS が実行時に追加します。

WEB OPEN コマンドで URIMAP リソースが参照される場合は、このパスがその接続の **WEB SEND** コマンドのデフォルト・パスとなります。URIMAP リソースが WEB SEND コマンドで参照される場合、パスがその **WEB SEND** コマンドで使用されますが、その URIMAP リソースのホスト属性は、接続の **WEB OPEN** コマンドで指定されるホストと一致しなければなりません。

e. オプション: SSL が使用されている場合は、この要求の SSL クライアント証明書として使用される証明書のラベルとして、CERTIFICATE 属性を指定します。

f. オプション: SSL または TLS が使用されている場合は、この要求用に使用する暗号コードとして、CIPHERS 属性を指定します。

g. オプション: URIMAP リソースを使用して開かれた接続を再使用に備えてプールする場合は、アプリケーション・プログラムが接続を使い終わった後にそれを CICS がプールに保持しておく時間の長さを、SOCKETCLOSE 属性に指定します。 CICS によるプール接続の管理と、接続プーリングによるアプリケーション・パフォーマンスの向上については、HTTP クライアントのパフォーマンスのための接続プーリングを参照してください。

次の例は、URIMAP リソースとして指定された URL `http://www.example.com:1030/software/index.html` を示しています。

```
Urimap:        softw
Group:         MYGROUP
Description:   Client request for software page
Status:        Enabled
Usage:         Client
Scheme:        HTTP
Host:          www.example.com:1030
Path:          /software/index.html
Socketclose:   001500
```

タスクの結果

CICS が HTTP クライアントとして動作して HTTP 要求を送信できるように URIMAP リソースを作成しました。リソースをインストールするうえで問題がある場合は、使用可能 URIMAP が同じ URI を示していないか確認してください。複数の使用可能 URIMAP リソースが CICS 領域内の同じ URI を示すことはできません。

HTTP クライアントの送信出口 XWBAUTH

XWBAUTH を使用すると、ターゲット・サーバーまたはサービス・プロバイダー用に基本認証の資格情報 (ユーザー名およびパスワード) を指定できます。
XWBAUTH は、HTTP を使って転送される許可ヘッダーを作成するために、要求に応じてこれらを CICS に渡します。

EXEC CICS WEB SEND (Client) コマンドまたは WEB CONVERSE コマンドで AUTHENTICATE(BASICAUTH) が指定されている場合は、ユーザー名およびパスワードをアプリケーションから提供することができます。提供されていない場合は、こうした資格情報を指定する別の方法として XWBAUTH が呼び出されます。

ユーザー名とパスワードは、通常はリモート・サーバー環境に特定のもので、RACF システムで使用される標準の 8 文字よりも長いことがあります。ユーザー名フィールドおよびパスワード・フィールドの長さは、最長 256 文字まで可能です。これらのフィールドの構文の妥当性検査は行われません。

ホストは UEPHOST パラメーターとしてユーザー出口プログラムに渡され、パスは UEPPATH パラメーターとして渡されます。レルムはオプションで、UEPREALM パラメーターとして渡されます。応答では、ユーザー出口プログラムによってユーザー名は UEPUSNM パラメーターとして、パスワードは UEPPSWD パラメーターとして戻されます。

次に紹介する出口プログラムのサンプルが、CICS サンプル・ライブラリーの SDFHSAMP に同梱されています。

- DFH$WBPI
- DFH$WBEX
- DFH$WBX1
- DFH$WBX2
- DFH$WBGA。DFH$WBPI、DFH$WBX1、DFH$WBX2、および DFH$WBEX の各サンプルが使用するグローバル作業域をマップした、コピーブックです。

クライアントの出口プログラムのサンプルについて詳しくは、*CICS Customization Guide* を参照してください。LDAP プロファイルのセットアップについて詳しくは、*CICS RACF Security Guide* を参照してください。

出口 XWBAUTH

呼び出される状況

EXEC CICS WEB SEND コマンドまたは WEB CONVERSE コマンドで AUTHENTICATE(BASICAUTH) は指定されているが、USERNAME および PASSWORD は指定されていない場合。

出口固有のパラメーター

UEPHOST (CICS 提供の入力)
ホスト名のアドレス (接続の WEB OPEN コマンドの HOST オプションで指定された IPv4 または IPv6 アドレス) が含まれるフィールドのアドレス。ホスト名は、このフィールドに保管されるときに小文字に変換されます。ユーザー出口プログラムは、ホスト名とのマッチング時に、この変換を考慮に入れる必要があります。

UEPHOSTL (CICS 提供の入力)
ホスト名のハーフワードの長さが入っているフィールドのアドレス。

UEPPATH (CICS 提供の入力)
WEB SEND コマンドまたは WEB CONVERSE コマンドの PATH オプションで指定されたパスのアドレスが含まれるフィールドのアドレス。パスは、指定どおりの大/小文字混合です。

UEPPATHL (CICS 提供の入力)
パスのハーフワードの長さが入っているフィールドのアドレス。

UEPREALM (CICS 提供の入力)
宛先に関連付けられているレルム名のアドレスが含まれるフィールドのアドレス (サーバーからの前回の HTTP 401 応答で、レルム名が返された場合)。

UEPREALML (CICS 提供の入力)
レルム名のハーフワードの長さが入っているフィールドのアドレス。

UEPAUTHT (CICS 提供の入力)
認証タイプを示す、1 バイト・コードのアドレス。これはバイナリーの 01 であり、基本認証を示します。

UEPUSNM (ユーザー出口提供の出力)
HTTP サーバーへのアクセスに必要なユーザー名のアドレスが含まれるフルワード・フィールドのアドレス。ユーザー名を保管するために、定義済みのアドレスと 64 バイトの領域が CICS によって作成されます。この 64 バイトの領域にユーザー名を入れて、UEPUSNM にあるアドレスは変更せずにそのままにすることができます。また、ユーザー名を独自の領域に入れて、UEPUSNM にあるアドレスを独自のユーザー名のアドレスで置き換えることもできます。独自のユーザー名領域を作成する場合、そのフィールドの長さは 256 バイトまで可能です。

UEPUSNML (CICS 提供の入力とユーザー出口提供の出力)
ハーフワード・フィールドのアドレス。最初、このフィールドには、UEPUSNM で提供されるバッファー・アドレスの長さが入ります。ユーザー出口プログラムは、このバッファーの長さを、UEPUSNM で提供されるユーザー名の長さに設定する必要があります。

UEPPSWD (ユーザー出口提供の出力)
HTTP サーバーへのアクセスに必要なパスワードのアドレスが含まれるフルワード・フィールドのアドレス。パスワードまたはパスワード・フレーズを保管するために、事前定義されたアドレスおよび 100 バイト領域が CICS によって作成されます。 この 100 バイトの領域にパスワードを入れて、UEPPSWD にあるアドレスは変更せずにそのままにすることができます。また、パスワードを独自の領域に入れて、UEPPSWD にあるアドレスを独自のパスワードのアドレスで置き換えることもできます。独自のパスワード領域を作成する場合、そのフィールドの長さは 256 バイトまで可能です。

UEPPSWDL (CICS 提供の入力とユーザー出口提供の出力)
ハーフワード・フィールドのアドレス。最初、このフィールドには、UEPPSWD で提供されるバッファー・アドレスの長さが入ります。ユーザー出口プログラムは、このバッファーの長さを、UEPPSWD で提供される実際のパスワードの長さに設定する必要があります。

UEPHOSTT (CICS 提供の入力)
UEPHOST パラメーターに含まれているホスト・タイプを示す 1 バイト・コードのアドレス。

2 進数 01 はホスト名を、2 進数 02 は IPv4 アドレスを、2 進数 03 は IPv6 アドレスを示します。

戻りコード

UERCNORM
出口は正常にユーザー名およびパスワードを戻しました。

UERCBYP
出口がユーザー名およびパスワードを識別できません。 Authorization ヘッダーは送信されません。

UERCERR
出口がユーザー名およびパスワードを識別できません。WEB SEND (Client) コマンドまたは WEB CONVERSE コマンドは停止される必要があります。

XPI 呼び出し
すべての XPI 呼び出しが使用可能です。

API コマンドおよび SPI コマンド
EXEC CICS SHUTDOWN および **EXEC CICS XCTL** を除く、すべての API コマンドおよび SPI コマンドを使用できます。

XWBAUTH による LDAP XPI 関数の一般的な使用方法

DFHDDAPX 関数の想定される使用方法 (XWBAUTH グローバル・ユーザー出口との関連における) には、LDAP セッションの開始と終了、資格情報の結果の表示、結果のスキャンと探索、表示の終了、正しい値の戻し、および検索の終了が含まれます。

BIND_LDAP
LDAP サーバーとのセッションを確立します。グローバル・ユーザー出口 XWBAUTH への最初の呼び出しで一度使用されます。 LDAP セッション・トークンは、XWBAUTH のグローバル作業域が用意されている場合はそこに保管され、それ以降の LDAP_SEARCH に対する呼び出しで使用されます。

UNBIND_LDAP
LDAP サーバーとの接続を解放します。この関数は、CICS シャットダウン処理時にのみ必要になります。この関数は、XSTERM (システム終了) グローバル・ユーザー出口の際に使用できます。

SEARCH_LDAP
必要なユーザー情報の URL およびレルムを識別する LDAP 識別名を指定して、資格情報を検索します。識別名は次の形式で指定されます。

```
racfcid=uuuuuuuu, ibm-httprealm=rrrrrrrr, labeledURI=xxxxxxxx, cn=BasicAuth
```

ここで、

- uuuuuuuu は、XWBAUTH パラメーター UEPUSER から取得された現在のユーザー ID です。
- rrrrrrrr には、XWBAUTH パラメーター UEPREALM から取得される HTTP 401 レルムが入ります (これが存在する場合)。
- xxxxxxxx には、http:// とホスト名を連結することで XWBAUTH パラメーター UEPHOST から取得されるターゲット URL、および、XWBAUTH パラメーター UEPPATH から取得されるパスが入ります。

- cn=BasicAuth は、基本認証資格情報を保管するために、LDAP サーバー内に構成される任意の接尾部です。

START_BROWSE_RESULTS
SEARCH_LDAP から戻される結果のスキャンを開始します。

GET_NEXT_ENTRY
SEARCH_LDAP から戻された一連のエントリー内の次の結果エントリーを見つけます。通常は、SEARCH_LDAP で指定される URL で固有の入力が位置指定されるため、GET_NEXT_ENTRY 関数は使用されません。

GET_NEXT_ATTRIBUTE
現在の結果エントリー内の次の属性を見つけます。 通常は、固有の属性が選択されるため、GET_NEXT_ATTRIBUTE 関数は使用されません。

END_BROWSE_RESULTS
SEARCH_LDAP で開始された表示セッションを終了します。

GET_ATTRIBUTE_VALUE
ターゲット識別名の各種属性の値を戻します。 XWBAUTH の場合、この属性の値は、属性 uid に保管されているユーザー名、および userpassword に保管されているパスワードです。 XWBAUTH はこれらの属性値を資格情報として戻します。

FREE_SEARCH_RESULTS
SEARCH_LDAP によって開始された検索を終了し、関連付けられているストレージを解放します。

HTTP クライアントのオープン出口 XWBOPEN

XWBOPEN を使用して、HTTP クライアントとしての CICS による HTTP 要求に使用されるプロキシー・サーバーを指定できます。これらの要求に対して指定されるホスト名にセキュリティー・ポリシーを適用することもできます。

XWBOPEN は、アプリケーション・プログラムがサーバーとの接続を開くために使用する **EXEC CICS WEB OPEN** コマンドの処理中に呼び出されます。XWBOPEN は、**EXEC CICS INVOKE SERVICE** コマンドの処理中にも呼び出されます。

CICS には、HTTP クライアントとしての CICS による HTTP 要求のためのプロキシー・サーバーの使用 (およびその他) に関する要件は一切ありません。また、CICS がこれらの要求にセキュリティー・ポリシーを適用することもありません。ご使用のシステムまたは組織でこのような機能が必要な場合は、ユーザーがセットアップする必要があります。

EXEC CICS WEB OPEN コマンドは、CICS Web ドメインに、サーバーとの接続を開くように指示します。 XWBOPEN は接続が開かれる前に呼び出されます。 **EXEC CICS** WEB OPEN コマンドの HOST オプションで指定される接続のホスト名 (例えば www.example.com) は、UEPHOST パラメーターとしてユーザー出口プログラムに渡され、検査されます。この時点で、ユーザー出口プログラムを次の 2 つの目的で使用できます。

- HTTP 要求にプロキシー・サーバーが使用されるかどうかを判別し、必要なプロキシー・サーバーの名前を返す。 プロキシー・サーバーが必要な場合は、戻りコード UERCPROX が使用され、プロキシー・サーバーの名前が CICS Web ドメ

イン (UEPPROXY によって識別されるバッファー内) に返されて、サーバーとの接続を作成するために使用されます。プロキシー・サーバーが必要でない場合は、戻りコード UERCNORM が使用されます。

- セキュリティー・ポリシーをホスト名に適用する。 戻りコード UERCBARR は、ホストへのアクセスが許可されていないことを示し、NOTAUTH 応答が WEB OPEN コマンドに返されます。アプリケーション・プログラマーは、その接続を開こうとするのを止めなければなりません。ホストだけでなく (またはホストではなく) 個々のリソースにセキュリティー・ポリシーを適用する場合には、**EXEC CICS** WEB SEND コマンドおよび **EXEC CICS** WEB CONVERSE コマンド上の XWBSNDO ユーザー出口を使用して、その URL のパス構成要素にセキュリティー・ポリシーを適用することができます。

XWBOPEN ユーザー出口は、**EXEC CICS** コマンドの使用をサポートしていません。

関連するコピーブックの DFH$WBGA を使用するサンプル・プログラム DFH$WBPI および DFH$WBEX は、グローバル作業域でプロキシー・サーバー情報またはセキュリティー・ポリシーをセットアップする方法を示しています。例えば、CICS システムからのすべての要求に単一のプロキシー・サーバーを使用しなければならない場合には、そのプロキシー・サーバー名を初期設定パラメーターとして指定することができます。多数のプロキシー・サーバーを使用しているか、またはセキュリティー・ポリシーを異なるホスト名に適用する場合、ホスト名を適切なプロキシー・サーバーに一致させるか、禁止のマークを付けるテーブルをロードまたは作成できます。これにより、**EXEC CICS** WEB OPEN コマンドの処理の間に、検索テーブルとして使用されます。サンプル・プログラムは、プログラム・リスト・テーブルの初期設定後 (PLTPI) 処理中に実行するか、または **EXEC CICS** WEB OPEN コマンドの使用が予期される前の任意の時点で実行することができます。

出口 XWBOPEN

呼び出される状況

EXEC CICS WEB OPEN コマンドまたは **EXEC CICS INVOKE SERVICE** コマンドの処理中。

出口固有のパラメーター

UEPHOST (CICS 提供の入力)
ホスト名 (WEB OPEN コマンドの HOST オプションで指定された IPv4 または IPv6 アドレス) が含まれるフィールドのアドレス。

注: ホスト名は、このフィールドに保管されるときに小文字に変換されます。ユーザー出口プログラムは、ホスト名とのマッチング時に、この変換を考慮に入れる必要があります。

UEPHOSTL (CICS 提供の入力)
ホスト名のハーフワードの長さが入っているフィールドのアドレス。

UEPPROXY (ユーザー出口提供の出力)
プロキシー・サーバー名を指し示すアドレスが入っているフィールドのアドレス。プロキシー・サーバー名は、URL 形式で指定する必要があります。 ユーザー出口プログラムへの入力時、このパラメーターは、2046 バイト領域のアドレスを含んだフィールドのアドレスに設定されます。この領域にプロキシー・サーバー名を入れ、UEPPROXY にある

アドレスは変更しないまま残すことができます。また、独自の領域にプロキシー・サーバー名を入れ、UEPPROXY にあるアドレスを、独自の領域のアドレスが含まれているフィールドのアドレスで置き換えることもできます。

UEPPROXYL (ユーザー出口提供の出力)
プロキシー・サーバー名のハーフワードの長さが入っているフィールドのアドレス。

UEPHOSTT (CICS 提供の入力)
UEPHOST パラメーターに含まれているホスト・タイプを示す 1 バイト・コードのアドレス。

注: 2 進数 01 はホスト名を、2 進数 02 は IPv4 アドレスを、2 進数 03 は IPv6 アドレスを示します。

戻りコード

UERCNORM
この HTTP 要求には、プロキシー・サーバーは必要ではなく、ホスト名は制限されません。

UERCPROX
この HTTP 要求にはプロキシー・サーバーが必要です。 UEPPROXY は必要なプロキシー・サーバーの名前に設定され、UEPPROXYL はプロキシー・サーバー名の長さに設定されます。

UERCBARR
そのサーバーのホスト名は制限されます。

UERCERR
終了処理中にエラーが発生しました。

XPI 呼び出し
すべての XPI 呼び出しが使用可能です。

API コマンドおよび SPI コマンド
EXEC CICS コマンドは使用できません。

HTTP クライアントの送信出口 XWBSNDO

XWBSNDO を使用して、HTTP クライアントとしての CICS による HTTP 要求に対するセキュリティー・ポリシーを指定できます。 XWBSNDO は、**EXEC CICS** WEB SEND コマンドまたは **EXEC CICS** WEB CONVERSE コマンドの処理中に呼び出されます。ホスト名とパスの情報が出口に渡されます。セキュリティー・ポリシーは、これらの構成要素のどちらか、または両方に適用することができます。

CICS は、HTTP クライアントとしての CICS による HTTP 要求に対して、どのようなセキュリティー・ポリシーも適用しません。ご使用のシステムまたは組織でこの機能が必要な場合は、ユーザーがセットアップする必要があります。

WEB OPEN コマンドの XWBOPEN 出口を使用して、ホストへのアクセスを全面的に制限することができます。 XWBSNDO 出口を使用して、同じ操作をすることも、ホスト内の特定のパスへのアクセスを制限することもできます。ホストへのアクセスを全面的に制限するには、XWBOPEN 出口を使用すると時間の節約になりま

す。アプリケーション・プログラムは接続を開くことができないので、送信する必要のある要求を作成することに時間を浪費しないからです。異なるホストで使用されるまったく同じパスを区別できるように、XWBSNDO 出口にホスト名が提供されます。

HTTP 要求に対してチャンク転送コーディングが使用されている場合、XWBSNDO は、そのチャンク化されているメッセージ用の最初の WEB SEND コマンドでのみ呼び出されます。

XWBSNDO ユーザー出口は、**EXEC CICS** コマンドの使用をサポートしていません。

ホストは UEPHOST パラメーターとしてユーザー出口プログラムに渡され、パスは UEPPATH パラメーターとして渡されます。戻りコード UERCNORM はそのパスが許可されていることを示し、戻りコード UERCBARR はそのパスが許可されていないことを示します。パスが許可されていない場合は、WEB SEND コマンドまたは WEB CONVERSE コマンドに NOTAUTH 応答が返されます。アプリケーション・プログラマーは、WEB CLOSE コマンドで接続を閉じることで、この応答に対処します。

出口 XWBSNDO

呼び出される状況

HTTP クライアントとしての CICS による HTTP 要求に対する、**EXEC CICS** WEB SEND コマンドまたは **EXEC CICS** WEB CONVERSE コマンドの処理中。クライアント要求は、WEB SEND コマンドで SESSTOKEN パラメーターを使用することで示されます。

出口固有のパラメーター

UEPHOST
ホスト名 (接続の WEB OPEN コマンドの HOST オプションで指定された IPv4 または IPv6 アドレス) が含まれるフィールドのアドレス。

注: ホスト名は、このフィールドに保管されるときに小文字に変換されます。ユーザー出口プログラムは、ホスト名とのマッチング時に、この変換を考慮に入れる必要があります。

UEPHOSTL
ホスト名のハーフワードの長さが入っているフィールドのアドレス。

UEPPATH
WEB SEND コマンドの PATH オプションで指定されたパスが入っているフィールドのアドレス。パスは、指定どおりの大/小文字混合です。

UEPPATHL
パスのハーフワードの長さが入っているフィールドのアドレス。

UEPHOSTT
UEPHOST パラメーターに含まれているホスト・タイプを示す 1 バイト・コードのアドレス。

注: 2 進数 01 はホスト名を、2 進数 02 は IPv4 アドレスを、2 進数 03 は IPv6 アドレスを示します。

戻りコード

UERCNORM
そのパスは許可されています。

UERCBARR
そのパスは許可されていません。

XPI 呼び出し
すべての XPI 呼び出しが使用可能です。

API コマンドおよび SPI コマンド
EXEC CICS コマンドは使用できません。

第 11 章 CICS Web サポートのセキュリティー

CICS がインターネットに接続される場合は、CICS アプリケーションおよびデータへの無許可アクセスを防止するために、また第三者がプライベート情報を取得できないようにするために、セキュリティー手段を講じてください。

CICS Web サポート・アーキテクチャーの開発プロセス全体を通して、セキュリティーは CICS Web サポートのアプリケーションおよびユーティリティー・プログラムの設計の一部であると考えてください。また、関連する CICS 機能のリソース定義を作成する際にも、このように考える必要があります。サブトピックでは、CICS Web サポート実装環境のセキュリティーを強化するために使用できる手段を要約しています。

HTTP サーバーとしての CICS: 認証および識別

HTTP サーバーとしての CICS の場合、TCPIPSERVICE 定義の AUTHENTICATE 属性で認証スキームを指定します。識別は認証プロセスに関連して取得されます。また、認証が不要の場合は、CICS が提供できます。

Web クライアントからの認証および識別の取得は、許可されていないユーザーによるアクセスから CICS システムを保護するための重要なステップです。

HTTP サーバーとしての CICS に対して適用されるセキュリティー手段を、TCPIPSERVICE リソース定義を使用して指定します。 CICS Web サポートに使用するポートごとに、TCPIPSERVICE リソース定義で以下の属性が指定されます。

- ポートに SSL を使用するか、しないか
- ポートに使用される認証スキーム
- 基本認証のためのレルム

認証

HTTP プロトコルで使用するものとして、CICS では、次の 2 つの認証スキームがサポートされています。

- **基本認証**は HTTP の一部であり、クライアントはユーザー ID とパスワードまたはパスワード・フレーズを提供することによって、サーバーに対して自身を認証および識別することができます。この情報は、デコードが容易な base-64 エンコード方式を使用してエンコードされます。このため、基本認証を認証の唯一の手段として使用するのが適しているのは、パスワードが傍受されることがない場合のみです。ほとんどの環境では、基本認証を SSL と組み合わせて使用します。そうすることで、SSL 暗号化によりユーザー ID とパスワードの情報が保護されます。 22 ページの『HTTP 基本認証』を参照してください。
- **SSL クライアント証明書認証**はより安全なクライアント認証方式で、信頼のおける第三者機関 (または認証局) が発行し、SSL 暗号化を使用して送信されるクライアント証明書が使用されます。*CICS RACF Security Guide*を参照してください。クライアント証明書には、CICS での識別に使用することのできるユーザー ID は含まれません。識別を行うには、証明書が使用される前か、またはクライア

ントが要求を行ったときに (基本認証を使用して) 自動的に、クライアント証明書を RACF またはそれと同等のセキュリティー・マネージャー内のユーザー ID と関連付けます。証明書が使用されるたびに、RACF ユーザー ID がクライアントのユーザー ID になります。これについては、*CICS RACF Security Guide*に記載されています。

109 ページの『CICS Web サポートの TCPIPSERVICE リソース定義の作成』には、CICS Web サポートに対して、これらの認証スキームの 1 つを指定する TCPIPSERVICE 定義をセットアップする方法が記載されています。

基本認証またはクライアント証明書認証を使用すると、CICS は、ユーザーからの認証要求のプロセスを処理し、必要に応じて認証情報をデコードし、提供された認証をセキュリティー・マネージャーのデータベースと照合し、認証が受け入れ不可の場合はその要求を拒否します。アナライザー・プログラムまたはユーザー作成アプリケーション・プログラムは、認証が検査されて受け入れられて初めて呼び出されます。

Web クライアントが使用するすべてのユーザー ID のユーザー・プロファイルは、RACF または相当する外部セキュリティー・マネージャーに作成されている必要があります。 *CICS RACF Security Guide*を参照してください。

基本認証の場合、ユーザーが指定したパスワードまたはパスワード・フレーズが期限切れになっていると、CICS は新規パスワードまたはパスワード・フレーズを求めるプロンプトをユーザーに出して、ユーザーが要求を再実行依頼できるようにします。 CICS 提供のユーティリティー・プログラム DFHWBPW が使用されます。このプロセスで CICS がユーザーに表示する Web ページ上のテキストを、カスタマイズできます。これについては、174 ページの『HTTP 基本認証のパスワード有効期限管理』に記載されています。

クライアント証明書認証の場合、CICS は、提供された証明書をセキュリティー・マネージャーのデータベースと照合し、また必要な場合は、ユーザーがセットアップしてある証明書取り消しリストとも照合して、その証明書を検査します。この処理によって取得された情報が要求の処理方法の決定に役立つ場合は、ユーザー作成アプリケーションでその情報を調べることができます。以下の項目を取得するには、EXTRACT CERTIFICATE コマンドを使用します。

- 発行者または対象の識別名の構成要素。識別名の詳細については、*CICS RACF Security Guide*を参照してください。
- 証明書に関連付けられている RACF ユーザー ID。

識別

Web クライアントのユーザー ID を取得するときに、識別が行われます。ID は以下のようにして Web クライアントから取得されます。

- 基本認証の実行時に取得する。
- ユーザー ID をクライアント証明書に関連付ける。

アプリケーション生成の応答の場合のみ、CICS は Web クライアントの代わりに、次の方法でユーザー ID を提供できます。

- アプリケーション生成の応答の処理パスで使用されるアナライザー・プログラムで、提供する (この ID は、取得された Web クライアントのユーザー ID をオーバーライドできます)。
- 要求の URIMAP 定義で提供する (この ID は、取得された Web クライアントのユーザー ID をオーバーライドできません)。
- 他の ID を決定できない場合、CICS のデフォルトのユーザー ID として、提供する。

Web クライアントの代わりにユーザー ID を提供する場合は、クライアントの ID は認証されないことに注意してください。ユーザー ID を提供するのは、ユーザー自身のクライアント・システムと通信する際に、クライアント・システムが既にそのユーザーを認証していて、セキュア環境でサーバーと通信する場合のみにしてください。*CICS RACF Security Guide* では、TCPIPSERVICE 定義の設定に応じて、ユーザー ID を決定する方法の詳細を説明しています。

クライアントを識別済みの場合、そのクライアントのユーザー ID には、その他のユーザー ID と同じ様に、RACF またはそれ等価な外部セキュリティー・マネージャーを使用して、 CICS リソースへのアクセスを許可することができます。 CICS 文書テンプレートとして保管されている Web ページや z/OS UNIX ファイル、応答を提供するアプリケーションで使用される CICS コマンドなど、CICS において Web クライアントがアクセスする個々のリソースのいずれかまたはすべてに、リソース・レベルのセキュリティーを適用することもできます。 177 ページの『CICS システムと CICS Web サポートのリソース・セキュリティー』で、これらのリソースを保護する方法、および不要になった場合にリソース・レベルのセキュリティーを除去する方法を説明しています。

HTTP クライアントとしての CICS: 認証および識別

CICS を介して HTTP クライアント要求を行うと、基本認証、プロキシー認証、または SSL クライアント証明書認証を実行するようサーバーまたはプロキシーが要求することがあります。

基本認証は、WEB SEND または WEB CONVERSE コマンドの AUTHENTICATE オプションを使用して行えます。プロキシー認証は、ユーザー・アプリケーションが行います。クライアント証明書は、URIMAP 定義を使用して提供します。

クライアント・アプリケーションは、以下の方法で自身の認証を要求されることがあります。

- **基本認証**では、特定の情報にアクセスするため、ユーザー名およびパスワードを提供することができます。サーバーに要求を出すと、そのサーバーは 401 状況コードおよび WWW-Authenticate ヘッダーを含む応答を送信することがあります。このヘッダーには、基本認証が要求されているレルムが指定されています。要求した情報を受信するには、ユーザー名とパスワードを提供します。CICS は、ユーザーがそのレルムにアクセスできるように、ユーザー名とパスワードを指定した Authorization ヘッダーとともに要求を再送信します。 CICS は、Authorization ヘッダーを予期しているサーバーに、Authorization ヘッダーを直接送信することもできます。これにより、401 応答の必要性がなくなります。CICS は、ユーザー名とパスワードを ASCII に変換し、基本認証プロトコルで必要とされている base-64 エンコードを適用します。したがって、WEB SEND または WEB

CONVERSE コマンド、あるいは XWBAUTH ユーザー出口を介して、通常の文字で資格情報を提供できます。 149 ページの『基本認証のための資格情報の提供』 および 22 ページの『HTTP 基本認証』を参照してください。

- **プロキシー認証**は、プロキシー・サーバーによって開始されます。プロキシー認証の場合、応答の状況コードは 407 で、プロキシー・サーバーからの確認のための質問ヘッダーは Proxy-Authenticate、応答ヘッダーは Proxy-Authorization です。CICS はこのプロトコルをサポートしていません。
- **SSL クライアント証明書認証**では、信頼のおける第三者機関 (または認証局) が発行したクライアント証明書が使用されます。HTTPS 要求を行う際、サーバーはこの認証を提供するよう要求する場合もあります。「*CICS RACF Security Guide*」 では、証明書の取得方法と、RACF データベース内の鍵リングまたは相当する外部セキュリティー・マネージャーに証明書を保管する方法について、説明しています。 サーバーがクライアント証明書を要求した場合は、接続を行うための WEB OPEN コマンドで使用される URIMAP 定義、または WEB OPEN コマンドそのもので、適切な証明書のラベルを指定することができます。あるいは、WEB OPEN コマンドのオプションとして証明書ラベルを直接指定することもできます URIMAP 定義を使用しているのに、証明書ラベルが指定されていない場合、CICS 領域のユーザー ID の鍵リングで定義されているデフォルトの証明書が使用されます。

サーバーによっては、他のタイプの認証または識別を提供するよう要求する場合があります。受け入れ可能な認証または識別をサーバーに提供できない場合、要求は拒否されます。基本認証またはプロキシー認証の場合、サーバーが要求を拒否したときに使用されていた状況コードは、確認のための質問に対する状況コードと同じです (サーバーの場合は 401、プロキシーの場合は 407)。確認のための質問に応答したにもかかわらず、これらの状況コードの 1 つを含む応答をさらに受信した場合は、使用した許可情報が無効になっています。

HTTP 基本認証のパスワード有効期限管理

HTTP 接続で基本認証が使用されている場合、CICS Web サポートは、外部セキュリティー・マネージャー内のユーザー ID とパスワードを検査します。パスワードの有効期限が切れている場合は、CICS 提供のユーティリティー・プログラム DFHWBPW を使用して、ユーザーに対して新しいパスワードの選択を求めるプロンプトを出します。 DFHWBPW からユーザーに提示されるページは、ユーザー自身がカスタマイズまたは置換することができます。

DFHWBPW は、要求に適用される TCPIPSERVICE 定義が AUTHENTICATE 属性の BASIC、AUTOREGISTER、または AUTOMATIC オプションで定義されている場合の、パスワード有効期限管理にのみ使用されます。DFHWBPW の構造はコンバーター・プログラムと似ていますが、通常の CICS Web サポート処理パスの一部ではないので、その他の目的でこれにコードを追加する必要はありません。ユーザーが新しいパスワードを選択すると、DFHWBPW はクライアントを元の要求の URL にリダイレクトすることによって要求の実行依頼を再開するため、その要求の処理パスはすべて通常通りに発生します。

DFHWBPW は、以下の 2 つの Web ページをユーザーに提示します。

1. パスワード・プロンプト・ページ。このページには、以下の 2 つの要素が含まれています。
   a. パスワードの有効期間に関するメッセージ。ユーザーに対して表示される最初のメッセージには、パスワードの有効期限が切れていることが記載されます。ユーザー ID は、標準パスワードとパスワード・フレーズの両方を持つことができます。長さが 9 から 100 文字までのパスワードがパスワード・フレーズであり、8 文字以下のパスワードが標準パスワードです。標準パスワードとパスワード・フレーズは、互いに無関係の働きがあります。標準パスワードの有効期限が切れた場合は、新規の標準パスワードに置き換える必要があります。同様に、パスワード・フレーズの有効期限が切れた場合も、新規パスワード・フレーズに置き換える必要があります。ユーザーがパスワードを変更しようとして失敗した場合 (例えば、入力された新規パスワードの 2 つのコピーが一致しない)、さらにメッセージが表示されて問題が示されます。
   b. ユーザーがパスワードを変更するための HTML フォーム。
2. 確認および要求の最新表示ページ。このページでは、有効期限が切れたパスワードが正常に置換されたことを確認します。また、要求を自動または手動で再作成するための、最新表示タグおよび URL リンクが表示されます。

DFHWBPW は 3 つの CICS 文書テンプレート DFHWBPW1、DFHWBPW2、および DFHWBPW3 を使用して、これらの Web ページを作成します。CICS 提供の定義は、これらのテンプレートをロード可能プログラム (つまり、タイプ PROGRAM(DFHWBPW1) など) として定義します。定義は、CICS 提供のリソース定義グループ DFHWEB にあります。これらの定義は、別のグループにコピーし、リソース定義 ALTER コマンドを使用してテンプレートを別のソースから派生させることによって変更することができます。また、リソース定義は変更せず、その代わりにロードされるプログラムを変更することもできます。
DFHWBPW1、DFHWBPW2、および DFHWBPW3 の 3 つのプログラムは、アセンブリ言語データのみのモジュールで、そのソースは、対応する CICS サンプル・ライブラリー SDFHSAMP のメンバーに含まれています。これらのサンプルを変更して、DFHRPL データ定義ステートメントに連結されている通常の CICS プログラム・ライブラリーのいずれかの中に、再アセンブルおよびリンク・エディットすることができます。

ヒント: アセンブリ言語のアンパーサンド (&) をコーディングする場合は、2 つのアンパーサンド (&&) として入力する必要があります。

各 DFHWBPW テンプレートの内容と機能は、以下のとおりです。

DFHWBPW1

パスワード・プロンプト・ページの一部。そのページにつながる HTML ページ見出しを提供し、可能なパスワード妥当性メッセージのシンボルを (サーバー・サイドに組み込まれているシンボル設定用の技法を使用して) 設定します。メッセージは、以下の情報を提供します。

message.1
パスワードの有効期限が切れています。

message.2
入力されたユーザー ID が無効です。

message.3
2 回入力された新規パスワードが一致しません。

message.4
入力された以前のパスワード (有効期限が切れたもの) が正しくありません。

message.5
入力された新規パスワードは、パスワード品質規則に従っていないため、外部セキュリティー・マネージャーによって許可されませんでした。

message.6
そのユーザー ID は現在取り消されています。

DFHWBPW プログラムは、適切なシンボルを選択して、パスワード・プロンプト・ページの文書に挿入します。DFHWBPW1 をカスタマイズしてページ見出しやタイトルを変更することや、body タグを変更してページの色や背景を変更することが可能です。また、メッセージ・シンボルの内容を変更することもできます。

DFHWBPW2
パスワード・プロンプト・ページの一部。ユーザーがユーザー ID、古い (有効期限が切れた) パスワード (またはパスワード・フレーズ)、および希望する新規パスワード (同じものを 2 回) を入力する HTML フォームを作成します。 DFHWBPW2 をカスタマイズしてユーザーへのプロンプトに使用するテキストを変更するか、そのページのレイアウトを変更することができます。ただし、form タグや input タグの内容を変更することはできません。これを変更すると、DFHWBPW が本来の処理を行わない場合があります。

DFHWBPW3
確認および要求の最新表示ページ。このテキストは、期限切れのパスワードが正常に置換されたことをユーザーに通知し、パスワードの再入力を求めるプロンプトがクライアントからすぐに出されることをユーザーに明らかにします。このテキスト、およびページのレイアウトは、カスタマイズすることができます。

DFHWBPW3 によって要求プロセスが再開されます。これには、meta http-equiv="Refresh" タグが含まれています。このタグは、10 秒後に、有効期限の切れたパスワードが検出されたときにユーザーが要求していたページに自動的にリダイレクトします。このタグの制限時間を変更することができます。また、ユーザーが自動リダイレクトされないようにする場合は、このタグを削除できます。ただし、変更後のページには、本来要求されていたページへのリンクが必ず含まれている必要があります。そのページの URL は、シンボル &dfhwbpw_target_url; に入っています。要求プロセスを再開するということは、Web クライアントが旧パスワードをキャッシュに入れている場合は、直ちに新規パスワードに置換できるということを意味します。また、CICS Web サポートの処理パスが影響を受けないということにもなります。

CICS システムと CICS Web サポートのリソース・セキュリティー

CICS が HTTP サーバーである場合、CICS システムを、無許可のユーザーによるアクセスから保護する必要があります。システムが適切に保護されていない場合、ユーザーが機密データにアクセスしたり、システムを妨害して他のユーザーに対するサービス妨害を発生させたりすることが可能性があります。

CICS Web サポートへのアクセス全般を監視するには、HTTP クライアント要求を行う各ユーザーに ID を要求し、ユーザーが提示した ID を認証します。インバウンド・ポートの TCPIPSERVICE 定義を使用して、これらの要件を指定します。171 ページの『HTTP サーバーとしての CICS: 認証および識別』を参照してください。

Web クライアントが使用するすべてのユーザー ID のユーザー・プロファイルは、RACF または相当する外部セキュリティー・マネージャーに作成されている必要があります。 RACF ユーザー・プロファイルを参照してください。

Web クライアントの認証済みユーザー ID を取得したら、この ID を使用して、応答を提供するために使用する CICS 領域のリソースに対するリソース・レベルのセキュリティーを実装できます。手順は、以下の応答タイプごとに異なります。

- アプリケーションが生成する応答。
- 応答として CICS 文書テンプレートを提供する URIMAP 定義を使用した静的応答。
- z/OS UNIX システム・サービスのファイルを応答として提供する URIMAP 定義を使用した、静的応答。

アプリケーションが生成する応答の場合、CICS システムのデフォルト指定では、リソース・セキュリティー検査は実行されず、トランザクション・セキュリティー検査 (具体的には、別名トランザクションのトランザクション接続セキュリティー) が実行されます。トランザクション・セキュリティーが CICS 領域でアクティブであると想定した場合、Web クライアントの認証済みユーザー ID をセキュリティー検査に使用しない場合でも、アプリケーション生成の応答に明確に関連したアクションを実行する必要があります。

静的応答の場合、Web クライアントのユーザー ID に対してトランザクション接続セキュリティーは適用されません。ただし、CICS システムのデフォルト指定では、Web クライアントのユーザー ID が使用可能な場合には、応答レベルのセキュリティー検査が*実行*されます。認証済みユーザー ID を Web クライアントから取得する場合は、それらのユーザー ID のリソース権限を設定するか、リソース・レベルのセキュリティー検査を無効にするアクションを実行する必要があります。

CICS Web サポートによって提供されるすべての応答に Web クライアントのユーザー ID を使用したリソース・レベルのセキュリティーを実装するかどうかにかかわらず、以下の保護を備える必要があります。

- 無許可アクセスまたは悪意のあるアクセスからインバウンド・ポートを保護する手段を実装する。
- 無許可のユーザーによる変更から CICS システム・コンポーネントを保護し、認証済みユーザーが正しいアクセス権を持っていることを確認する。

関連情報

z/OS UNIX ファイルのセキュリティー

z/OS UNIX ファイルのセキュリティーの実装

インバウンド・ポートのセキュリティー

TCPIPSERVICE リソース定義で、CICS Web サポートに使用される各ポートを定義します。TCPIPSERVICE 定義では、SSL を使用するかどうかや、クライアントに要求される認証レベルなど、ポートのセキュリティー・オプションを指定します。ポートは、無許可アクセスや悪意のあるアクセスに対してガードされていなければなりません。

109 ページの『CICS Web サポートの TCPIPSERVICE リソース定義の作成』で、ポートの定義の作成方法について説明しています。

ポートの保護状態が維持されるようにするには、以下のようにします。

- すべての TCPIPSERVICE 定義に MAXDATALEN 属性を指定します。これは、CICS が単一の要求に対して受け入れるデータの最大量を制限するオプションであり、大容量データの送信を伴うサービス妨害アタックに対する CICS の防御に役立ちます。
- Web クライアントとの対話を確実に機密の状態に維持し、第三者が傍受できないようにするには、Secure Sockets Layer (SSL) を使用します。機密データを送信する場合、またはユーザー ID やパスワードなどの許可情報をサーバーに渡す場合は、SSL を使用することが特に重要です。 184 ページの『CICS Web サポートでの SSL』を参照してください。

1 つ以上の CICS Web サポートのポートで異常な活動を発見した場合は、CICS システム・コマンドを使用して、CICS Web サポートをさまざまなレベル (単一の要求、仮想ホスト、ポート、または CICS Web サポート全体) でシャットダウンし、CICS システムをシャットダウンする必要はありません。 123 ページの『HTTP 要求の拒否』を参照してください。

URIMAP リソース定義では、要求のスキームとして HTTP または HTTPS を指定します。 HTTP が指定された URIMAP は、HTTP またはより安全な HTTPS のどちらかを使用して作成された Web クライアント要求を受け入れます。HTTPS が指定された URIMAP は、HTTPS を使用して作成された Web クライアント要求のみ受け入れます。

HTTPS が指定された URIMAP 定義が Web クライアントからの要求と一致する場合、CICS は、要求で使用されたインバウンド・ポートが SSL を使用していることをチェックします。SSL がポートに指定されていない場合、要求は 403 (Forbidden) 状況コードで拒否されます。 URIMAP 定義がすべてのインバウンド・ポートに適用されると、このチェックにより、Web クライアントは非セキュアなポートを使用してセキュアなリソースにアクセスすることができなくなります。 HTTP を指定した URIMAP 定義に対しては検査が行われないため、Web クライアントは、非セキュアなポート、またはセキュアな (SSL) ポートのどちらを使用しても、保護されたリソースにアクセスできます。

CICS システム・コンポーネントのセキュリティー

他の CICS リソースと同様に、CICS Web サポートで使用される CICS システム・コンポーネントを、無許可ユーザーによる変更から保護する必要があります。 また、許可されているユーザー (特に CICS 領域) に、これらのコンポーネントの使用に必須の権限があることを確認することも必要です。

CICS Web サポートを制御するには、アプリケーション・プログラムやリソース定義など、多数のコンポーネントが使用されます。26 ページの『CICS Web サポートのコンポーネント』を参照してください。これらのコンポーネントを無許可アクセスから保護しないと、CICS Web サポート・アーキテクチャーのセキュリティーが危うくなる可能性があります。例えば、ポートの TCPIPSERVICE 定義へのアクセス権を持つユーザーによって、SSL の使用または ID の提供に必要な、Web クライアントの要件が削除される可能性があります。 CICS トランザクション、リソース、およびコマンドを許可されていない使用から保護する方法については、*CICS RACF Security Guide*を参照してください。

一部の CICS システム・コンポーネントでは、許可されているユーザーに対して、追加のアクセス権限をセットアップする必要があります。

- URIMAP リソースでは、Web クライアントのユーザー ID を設定するために、追加の権限が必要とある場合があります。代理ユーザー検査が、(システム初期設定パラメーターとして XUSER=YES が指定されている) CICS 領域で使用可能に設定されている場合、CICS は、URIMAP 定義をインストールするために使用されたユーザー ID が、USERID 属性に指定されているユーザー ID の代理として許可されているかどうかを確認します。
- 文書テンプレートを使用して、HTTP サーバーとしての CICS からの応答のボディを作成したり、HTTP クライアントとしての CICS からの要求のボディを作成したりできます。文書テンプレートは、DOCTEMPLATE リソース定義で定義します。 文書テンプレートが区分データ・セットで保管される場合、CICS 領域のユーザー ID は、そのデータ・セットの READ 権限を持っている必要があります。
- z/OS UNIX システム・サービス・ファイルを使用して、HTTP サーバーとしての CICS からの静的応答のボディを作成できます。これらは、それぞれ独自の名前で指定するか、DOCTEMPLATE リソース定義で定義することができます。z/OS UNIX ファイルを使用する場合は、z/OS UNIX へのアクセス権限と、ファイルを含む z/OS UNIX ディレクトリーおよびファイルそのものへのアクセス権限が、CICS 領域に必要です。 *Java Applications in CICS*を参照してください。

アプリケーションが生成する応答のリソース・セキュリティーとトランザクション・セキュリティー

セキュリティー・マネージャーにプロファイルが作成されている Web クライアントの認証済みユーザー ID を取得した場合、このユーザー ID は、アプリケーションが生成する応答に使用される別名トランザクションに適用されます。

このタスクについて

Web クライアントのユーザー ID に適切な権限を与えるか、独自の標準ユーザー ID をオーバーライドとして指定します。 Web クライアントのユーザー ID をリソ

ース・セキュリティー検査に使用するかどうかにかかわらず、別名トランザクション用のユーザー ID が適切な権限を持つようにしておく必要があります。

TRANSACTION リソース定義で別名トランザクションを定義します。アプリケーションが生成する各応答の別名トランザクションは、要求の URIMAP 定義によって指定するか、またはアナライザー・プログラムによって指定されます。デフォルトは CICS 提供の別名トランザクション CWBA です。これは、Web 対応アプリケーションまたは COMMAREA アプリケーションのどちらかを使用して応答を提供するときに適用されます。

別名トランザクションを実行するときのユーザー ID には、以下のタスクの実行権限が必要です。

- 別名トランザクションを接続する (CICS 領域に対してトランザクション接続セキュリティーが指定されている場合)。トランザクション接続セキュリティーは、システム初期設定パラメーターである XTRAN によって制御されます。デフォルトは、YES (トランザクション接続セキュリティーがアクティブ) です。
- 別名トランザクションで使用される任意の CICS リソースにアクセスする (別名トランザクションに対してリソース・セキュリティーが指定されている場合)。リソース・セキュリティーは、別名トランザクションの TRANSACTION リソース定義の RESSEC 属性によって制御されます。デフォルトは NO (リソース・セキュリティーなし) であり、CWBA に対して提供されている設定も NO です。
- 別名トランザクションで使用される任意の CICS システム・プログラミング・コマンドにアクセスする (別名トランザクションに対してコマンド・セキュリティーが指定されている場合)。これらのシステム・プログラミング・コマンドは、応答を生成するユーザー作成アプリケーション・プログラムで使用されます。コマンド・セキュリティーは、別名トランザクションの TRANSACTION リソース定義の CMDSEC 属性によって制御されます。デフォルトは NO (コマンド・セキュリティーなし) であり、CWBA に対して提供されている設定も NO です。

Web クライアントが CICS Web サポートに対して要求を行い、アプリケーションによって応答が提供されると、CICS は以下の優先順に従って別名トランザクション用のユーザー ID を選択します。

1. アナライザー・プログラムを使用して設定したユーザー ID。このユーザー ID は、Web クライアントから取得された、または URIMAP 定義によって提供されたユーザー ID をオーバーライドできます。
2. 基本認証を使用して Web クライアントから取得したユーザー ID、または Web クライアントによって送信されたクライアント証明書に関連付けられたユーザー ID。
3. 要求に対する URIMAP 定義で指定したユーザー ID。
4. 他に判別できるユーザー ID がない場合、CICS デフォルト・ユーザー ID。

CICS Web サポートのアーキテクチャーに応じて、さまざまな要求に対して 1 つまたは複数のタイプのユーザー ID を使用する場合があります。 Web クライアントの認証済みユーザー ID を取得した場合、その ID をオーバーライドするアクションを実行しない限り、その ID が別名トランザクションに使用されます。

アプリケーションが生成する応答に対して、以下のセキュリティー・アクションを実行します。

手順

1. Web クライアントの認証済みユーザー ID を取得しても、それをアプリケーション生成応答のセキュリティー検査に*使用しない* 場合は、アナライザー・プログラムを使用して、関連する別名トランザクションの標準ユーザー ID で Web クライアントのユーザー ID をオーバーライドします。 (CICS のデフォルトのユーザー ID を使用できます)。このオーバーライドを指定する要求の処理パスに、アナライザー・プログラムを含めます。 441 ページの『付録 E. アナライザー・プログラム』を参照してください。 このユーザー ID のユーザー・プロファイルがセキュリティー・マネージャーに定義されていることを確認してください。 標準ユーザー ID を設定した場合は、この手順の残りのステップに従って、標準ユーザー ID に必要な権限を与えることができます。
2. Web クライアントの認証済みユーザー ID を取得しない場合は、別名トランザクション用の標準ユーザー ID とする適切なユーザー ID を選択してください。CICS のデフォルト・ユーザー ID を使用するのでない限り、選択したユーザー ID を要求の URIMAP 定義で指定するか、選択したユーザー ID を指定するアナライザー・プログラムを設定します。 標準ユーザー ID のユーザー・プロファイルがセキュリティー・マネージャーに定義されていることを確認してください。
3. CICS 領域にトランザクション接続セキュリティーが指定されているとすれば、別名トランザクション用の可能性のあるユーザー ID すべてが、トランザクションを接続するための権限を持つようにしておく必要があります。 ユーザー ID をすべて挙げると、Web クライアントのユーザー ID (取得されてオーバーライドされない場合)、URIMAP 定義またはアナライザー・プログラムで指定した標準ユーザー ID、または CICS のデフォルト・ユーザー ID そのものなどが考えられます。「*CICS RACF Security Guide*」を参照してください。
4. オプション: 別名トランザクションで使用されるリソースに対してリソース・レベルのセキュリティー検査を適用するには、以下のようにします。
   a. 別名トランザクションで使用されるすべての CICS リソースを識別し、その中のどのリソースが CICS 領域でリソース・セキュリティー検査の対象になるかを決定します。 CICS Web サポート用のアプリケーション・プログラムで使用される可能性のあるいくつかのリソース、およびそれらのリソースに対するリソース・セキュリティー検査を制御するシステム初期設定パラメーターを以下に示します。
      - CICS 文書テンプレート (XDOC システム初期設定パラメーター)。
      - ビジネス・ロジックを実行するためにメイン・アプリケーション・プログラムによって呼び出される他のアプリケーション・プログラム (XPPT システム初期設定パラメーター)。
      - HTTP 要求シーケンス全体でアプリケーション状態を共用するために使用される一時記憶域キュー (XTST システム初期設定パラメーター)。
      - CICS ファイル制御によって管理されるファイル (XFCT システム初期設定パラメーター)。

      HFS ファイルがアプリケーション・プログラムによって使用される場合、HFS ファイルのリソース・セキュリティー検査 (XHFS システム初期設定パラメーター) は適用されません。これらのファイルは、CICS 文書テンプレートとして定義されている場合に限りアプリケーション・プログラムが操作で

き、この状態においては、CICS 文書テンプレート・セキュリティーがファイルへのアクセスを制御するためです。「*CICS RACF Security Guide*」を参照してください。

アナライザー・プログラムを使用する場合、アナライザー・プログラムが別名トランザクションのメインプログラムであり、したがってアナライザー・プログラムはリソース・セキュリティー検査の対象になりません (対象となるのはトランザクション接続セキュリティー検査のみです)。ただし、ユーザー作成の Web アプリケーション・プログラムそのもの、およびユーザーが使用するなんらかのコンバーター・プログラムはすべて、別個のリソース・セキュリティー検査の対象となることに注意してください。同様に、コンバーター・プログラムを使用し、アナライザー・プログラムを使用しない場合は、コンバーター・プログラムが別名トランザクションのメインプログラムですが、コンバーター・プログラムによって呼び出されるアプリケーション・プログラムは別個のリソース・セキュリティー検査の対象となります。

b. 別名トランザクションを接続する権限があるすべてのユーザー ID に、別名トランザクションで使用される保護されたリソースを使用する権限を与えます。

c. トランザクションの TRANSACTION リソース定義で RESSEC(YES) を指定します。

5. オプション: 別名トランザクションで使用される CICS システム・プログラミング・コマンドに対してコマンド・セキュリティー検査を適用するには、以下のようにします。

a. コマンド・セキュリティーが CICS 領域でアクティブであることを確認します。コマンド・セキュリティーは、XCMD システム初期設定パラメーターによってアクティブ化します。

b. トランザクションに関連付けられたアプリケーション・プログラム、アナライザー・プログラム (使用する場合)、およびコンバーター・プログラム (使用する場合) で使用する CICS システム・プログラミング・コマンドを識別します。 「*CICS RACF Security Guide*」には、コマンドのチェックリストがあります。

c. 別名トランザクションを接続する権限があるすべてのユーザー ID に、別名トランザクションで使用されるコマンドを使用する権限を与えます。

d. 別名トランザクションの TRANSACTION リソース定義で CMDSEC(YES) を指定します。

次のタスク

CICS 領域でセキュリティー検査が行われるようにするには、システム初期設定パラメーター SEC=YES を設定します。

文書テンプレートを使用する静的応答のリソース・レベル・セキュリティー

基本認証またはクライアント証明書認証を実装した状態で、特定の Web ページへのユーザーのアクセスを制御する場合は、Web クライアントの認証済みユーザー ID を使用して、静的応答を提供するために使用する個々の CICS 文書テンプレートへのアクセスを制御できます。

このタスクについて

URIMAP 定義で指定された CICS 文書テンプレートを使用して CICS Web サポートが送信する静的応答に対しては、デフォルトでリソース・セキュリティー検査が有効になります。

XRES システム初期設定パラメーターは、CICS 文書テンプレートのリソース・セキュリティーを制御します。このパラメーターのデフォルトは YES であり、リソース・セキュリティーがアクティブです。 CICS 領域内で目的を問わず使用される CICS 文書テンプレートについて、リソース・セキュリティー検査を行わない場合は、このシステム初期設定パラメーターを NO に設定することで、検査を無効にすることができます。

すべての静的応答のトランザクションは、デフォルトの Web リスナー・トランザクションである CWXN か、または TCPIPSERVICE 定義の TRANSACTION 属性を使用して CWXN の代わりに指定した代替トランザクションです。CICS 文書テンプレートの場合は、TRANSACTION リソース定義の RESSEC 属性でリソース・セキュリティー検査を制御することもできます。 CWXN の場合、CICS での提供時に RESSEC(YES) が指定されます。これは、リソース・セキュリティーがアクティブになるということです。静的応答に対してリソース・セキュリティー検査を使用しない場合、検査を非アクティブにする最適な方法は、TCPIPSERVICE 定義の CWXN を、プログラム DFHWBXN と RESSEC(NO) を指定した代替トランザクションに置き換えることです。この設定により、CICS 文書テンプレートに対するリソース・セキュリティー検査が、静的応答についてのみ非アクティブになります HFSFILE 属性で指定した z/OS UNIX ファイルのセキュリティー検査については、RESSEC 属性で制御することはできません。

区分データ・セット、CICS プログラム、CICS ファイル、z/OS UNIX システム・サービス・ファイル、一時記憶域キュー、一時データ・キュー、出口プログラムなど、さまざまなソースから文書テンプレートを取得できます。リソース・セキュリティー検査が文書テンプレートに対して行われる場合、文書テンプレートを提供するリソースについては、CICS は追加のセキュリティー検査を実行しません。CICS 領域で、そのタイプのリソースに対してリソース・セキュリティーが指定されていても、同様です。

CICS 文書テンプレートを使用して、静的応答のリソース・レベル・セキュリティーを設定するには、以下の手順を実行します。

手順

1. Web クライアントが使用する認証済みユーザー ID を識別します。これらの ID は、リソース・セキュリティー検査の基礎になります (アプリケーションが生成する応答の場合とは異なり、アナライザー・プログラムを使用してオーバーライドを提供することはできません)。 認証済みユーザー ID は、すでにセキュリティー・マネージャーにユーザー・プロファイルが定義されています。
2. 静的応答を提供するために使用する CICS 文書テンプレートを、すべて特定します。
3. CICS 領域で CICS 文書テンプレートのセキュリティーを実装します (Security for CICS document templates にある手順を参照)。 静的応答を提供するために使用する CICS 文書テンプレートのそれぞれについて、セキュリティー・マネージ

ャーにプロファイルを定義するとともに、認証済みの各ユーザー ID に対して、適切な CICS 文書テンプレートにアクセスする権限を与える必要があります。

4. CWXN の TRANSACTION リソース定義、または CWXN の代わりに指定した代替トランザクションの TRANSACTION リソース定義に、RESSEC(YES) が指定されていることを確認します。 CICS 提供の CWXN には RESSEC(YES) が指定されていますが、TRANSACTION リソース定義では一般に RESSEC(NO) がデフォルトになります。 このステップにより、静的応答に対するリソース・セキュリティー検査がアクティブになります。そのため、Web クライアントがユーザー ID を提供するときは必ず、適切な権限をセットアップしておく必要があります。

次のタスク

CICS 領域でセキュリティー検査が行われるようにするには、システム初期設定パラメーターの SEC を YES に設定します。

CICS Web サポートでの SSL

HTTP とともに SSL (Secure Sockets Layer) を使用することで、暗号化とメッセージ認証、また、証明書を使用するクライアントおよびサーバーの認証を使用可能にできます。SSL を使用するように CICS を構成してある場合は、HTTP サーバーとしての CICS、および HTTP クライアントとしての CICS の両方に対して、SSL の機能が使用可能になります。

「*CICS RACF Security Guide*」 では、SSL が提供する機能を説明し、 では、CICS で SSL を使用する方法を紹介しています。

CICS が HTTP サーバーである場合、SSL を使用して Web クライアントとの相互作用を保護することができます。 CICS がクライアントの要求を受信するポートの TCPIPSERVICE 定義で、適切なセキュリティー・オプションを指定します。

SSL を使用することを指定するだけでなく、基本認証またはクライアント証明書を要求することもできます。 Web クライアントをさらに支援するには、クライアント証明書をクライアントが提供できるようにし、次に、CICS 環境における識別を提供するためにクライアント自体をセキュリティー・マネージャーに登録します。識別を提供するために必要に応じて、クライアントが自己登録または基本認証を使用することを許可することもできます。これらのアクティビティーはすべて CICS によって処理されるので、アプリケーション生成応答を提供する場合は、アプリケーションがこの登録を処理する必要はありません。 109 ページの『CICS Web サポートの TCPIPSERVICE リソース定義の作成』を参照してください。

CICS が HTTP クライアントの場合、サーバーは、いくつかの接続に対しては SSL を使用するように要求することがあります。そのような場合は、以下のアクションの一部またはすべてを実行する必要があります。

- 接続のスキームとして、HTTPS を使用する。
- 接続に使用する暗号スイートのリストを提供する。これらは、接続を行うための WEB OPEN コマンドで使用する URIMAP 定義で指定することができます。
- クライアント証明書を提供する。クライアント証明書はすべての SSL トランザクションで必須なわけではありませんが、サーバーが、特定のトランザクション

に対してクライアント証明書を要求する可能性があります。サーバーがクライアント証明書を要求した場合は、接続を行うための WEB OPEN コマンドで使用される URIMAP 定義、または WEB OPEN コマンドそのもので、適切な証明書のラベルを指定することができます。クライアント証明書は、セキュリティー・マネージャーの鍵リングに保管する必要があります URIMAP 定義を使用しているのに、証明書ラベルが指定されていない場合、CICS 領域のユーザー ID の鍵リングで定義されているデフォルトの証明書が使用されます。

第 12 章 CICS Web サポートと非 HTTP 要求

標準外の要求形式を使用するユーザー作成クライアントからの要求を主にサポートするために、CICS Web サポートを使用して、HTTP 形式でないインバウンド TCP/IP クライアント要求を処理できます。処理と応答を定義します。クライアント/サーバー通信用に公式に定義されたプロトコルに固有のサポートは、一切提供されません。

CICS Web サポートは、CICS がサーバーであるときに、非 HTTP メッセージのみを処理します。CICS Web サポートによって行われる CICS クライアント要求では、HTTP プロトコルを使用します。

非 HTTP 要求を処理する場合は、CICS Web サポートに関する以下の点に注意してください。

- TCPIPSERVICE リソース定義を使用して、要求を受け取るポートを制御できます。
- アナライザー・プログラムを使用して、要求のアセンブルおよび構文解析を行い、コード・ページ変換を指定し、以後の要求処理を決定できます。定義した要求形式に合わせて要求の構文解析を行うように、アナライザー・プログラムをコーディングできますが、CICS では、公式の定義が存在する特定のプロトコルに対する具体的なサポートは提供されないことに注意してください。
- Web 対応アプリケーション・プログラムを使用するか、またはコンバーター・プログラムを使用した非 Web 対応アプリケーションを使用して、要求に対する応答を提供できます。要求および応答を、EXEC CICS WEB プログラミング・インターフェースの特定の要素を使用して処理したり、COMMAREA に入れて CICS アプリケーション間で受け渡したりすることができます。
- Web エラー・プログラム DFHWBEP は、アナライザー・プログラム、コンバーター・プログラム、またはユーザー作成アプリケーション・プログラムで異常終了が発生した場合のほか、要求にサービスを提供するアプリケーション・プログラムをアナライザー・プログラムおよびコンバーター・プログラムが判別できない場合にも、エラー応答を提供します。デフォルトでは標準の HTTP エラー・メッセージが使用されますが、必要に応じてこれらのメッセージを調整できます。

CICS Web サポート機能のなかには、非 HTTP 要求に使用不可のものがあります。使用不可の場合、それぞれ以下のとおりです。

- HTTP 要求を解釈して応答を作成するのを援助する一部の機能を使用することができません。例えば、個別にメッセージ・ヘッダーにアクセスすることができません。
- チャンク転送コーディングを含め、CICS TS バージョン 3 で導入された機能拡張は、非 HTTP 要求には一般的に使用できません。
- 持続接続はサポートされていません。
- URIMAP 定義を非 HTTP 要求に使用することはできません。

CICS Web サポートが非 HTTP メッセージに対して提供するサポートは、CICS 用の TCP/IP ソケット・インターフェースと同じではありません。 z/OS

Communications Server IP CICS ソケット・インターフェースは、クライアントが TCP/IP によって直接 CICS アプリケーション・プログラミング・インターフェースと通信できるようにするアプリケーション・プログラミング・インターフェースを提供します。 CICS Web サポートは、このプロセスの一部ではありません。

CICS ソケット・インターフェースは、z/OS Communications Server には提供されていますが、CICS には提供されていません。 CICS ソケット・インターフェースについては、「*z/OS Communications Server: IP CICS ソケット・ガイド*」(SC88-9053) に説明があります。

非 HTTP 要求の処理

CICS Web サポートを使用して非 HTTP 要求を処理するには、要求に対する処理を決定するためのアナライザー・プログラムと、応答を提供するためのアプリケーション・プログラムをコーディングします。またリソース定義をいくつか作成する必要もあります。

始める前に

65 ページの『第 4 章 CICS Web サポートのコンポーネントの構成』の説明に従って、CICS Web サポートの基本コンポーネントを構成します。

このタスクについて

非 HTTP 要求の処理には、CICS Web サポートの以下のコンポーネントが使用されます。

- TCPIPSERVICE リソース定義
- アナライザー・プログラム
- 必要に応じて、コンバーター・プログラム
- ユーザー作成アプリケーション・プログラム
- アプリケーション・プログラムの別名トランザクション
- Web エラー・プログラム DFHWBEP

HTTP 要求に対する処理、および非 HTTP 要求に対する処理は別々に行われます。非 HTTP 要求は、TCPIPSERVICE 定義で指定されている USER プロトコルを使用して受信されます。したがって、CICS は HTTP 要求および応答の基本的な受け入れ検査を行うことができ、非 HTTP 要求はこれらの検査の対象になりません。受け入れ検査を行うことにより、非 HTTP 要求に対してエラー応答が作成されて、要求が処理されないことがあります。

CICS Web サポートを使用して非 HTTP 要求を処理するには、以下を実行します。

手順

1. 使用するポートを決定します。 ポートごとにただ 1 つのアクティブな TCPIPSERVICE 定義が存在でき、非 HTTP 要求は HTTP 要求と同じポートを使用することはできません。ウェルノウン・ポート番号 80 (HTTP 用) および 443 (HTTPS 用) は、非 HTTP 要求を受け入れることができません。非 HTTP 要求を行う Web クライアントは、その要求に対して、ポート番号を URL で明示的に指定する必要があります。

2. 『非 HTTP 要求に対するリソース定義』の情報を使用して、要求に対してリソース定義をセットアップします。
3. 190 ページの『アナライザー・プログラムおよび非 HTTP 要求』の情報を使用して、各要求を処理するためのアナライザー・プログラムをコーディングします。
4. 1 つ以上のアプリケーション・プログラムを設計およびコーディングし、 191 ページの『非 HTTP 要求用のアプリケーション・プログラミング』の情報を使用して、各要求に対する応答を提供します。
5. Web エラー・プログラム DFHWBEP が、エラー状態で、適切な応答を提供することを確認してください。 非 HTTP 要求では、アナライザー・プログラム、コンバーター・プログラム、またはユーザー作成アプリケーション・プログラムで異常終了が発生した場合のほか、要求を処理するアプリケーション・プログラムをアナライザー・プログラムおよびコンバーター・プログラムが決定できない場合にも、DFHWBEP が使用されます。 デフォルトでは、DFHWBEP は同じ状態の HTTP 要求に対するエラー応答として送信される標準 HTTP メッセージを出しますが、これらのメッセージは必要に応じて調整できます。 131 ページの『第 9 章 Web エラー・プログラム』を参照してください。

非 HTTP 要求に対するリソース定義

非 HTTP 要求は、TCPIPSERVICE および TRANSACTION リソース定義を必要とします。非 HTTP 要求に対する TCPIPSERVICE リソース定義では、CICS 提供のトランザクション CWXU に関連付けられている USER (ユーザー定義) プロトコルを指定する必要があります。要求が USER プロトコルを介して受信される場合には、URIMAP リソース定義は使用されません。

このタスクについて

手順

1. 非 HTTP 要求に使用するポートごとに、USER プロトコルを含む TCPIPSERVICE リソース定義を作成します。 USER プロトコルで使用できる属性は、HTTP プロトコルで使用できる属性と同じです。 109 ページの『CICS Web サポートの TCPIPSERVICE リソース定義の作成』を参照してください。
2. TCPIPSERVICE リソース定義ごとに、CICS 提供のトランザクション CWXU を使用するのか、CICS Web ユーザー定義プロトコル接続トランザクションを使用するのか、または代替トランザクションを使用するのかを決定します。 DFHCURDI サンプルには CWXU のサンプル定義が含まれています。CWXU は CICS プログラム DFHWBXN を実行します。DFHWBXN を実行する代替トランザクションを使用できます (TCPIPSERVICE リソース定義でプロトコルに定義されている他のデフォルト・トランザクションを除く)。
3. オプション: 要求の処理に使用する別名トランザクションに対して TRANSACTION リソース定義を作成します。 114 ページの『CICS Web サポート用の TRANSACTION リソース定義の作成』を参照してください。

アナライザー・プログラムおよび非 HTTP 要求

非 HTTP 要求を処理するにはアナライザー・プログラムが必要です。このプログラムは、ネットワーク上の伝送のために分割されていた要求を再構成し、要求のコード・ページ変換を指定し、以後の要求処理を決定するために必要な構文解析を実行することができます。

非 HTTP 要求の再構成

着信要求は、ネットワーク上の伝送用にいくつかの部分に分割されることがあります。非 HTTP 要求の場合、CICS はアナライザー・プログラムを呼び出すまで要求を再構成しないので、それを踏まえてアナライザー・コードを作成する必要があります。

アナライザーへの入り口で、user_data ポインターは、着信要求の先頭部分が格納されている COMMAREA のアドレスを指し示します。要求の次の部分を受け取るには、戻りコードを URP_EXCEPTION に設定し、理由コードを URP_RECEIVE_OUTSTANDING に設定します。 CICS Web サポートが再びアナライザーを呼び出すと、user_data ポインターはメッセージの次の部分を指し示します。要求全体の受け取りが終了するまで、このプロセスを繰り返すことができます (最大サポート長の 32,767 バイトまで)。

この処理の結果は、CICS WEB API コマンドには見えません。しかし、再構成されたメッセージをコンバーター・プログラムに渡すことはできます。

非 HTTP 要求に対するコード・ページ変換の指定

非 HTTP 要求の場合、 CICS Web サポートは、アナライザー・プログラムを起動する前に要求のコード・ページ変換を行いません。

アナライザーは、コード・ページ変換テーブル (DFHCNV) キー、またはクライアントおよびサーバーのコード・ページ出力パラメーターのいずれかを使用して、HTTP 要求の場合と同様に、非 HTTP 要求についてもコード・ページ変換を指定することができます。 444 ページの『アナライザー・プログラムの作成』を参照してください。

あるいは、 Web 対応アプリケーション・プログラムが、着信の非 HTTP 要求のコード・ページ変換を WEB RECEIVE コマンドで指定することもできます。

非 HTTP 要求は、要求行、ヘッダー、およびボディの各要素に構文解析されません。コード・ページ変換は要求全体が対象となります。

非 HTTP 要求処理の決定

HTTP 要求に関連する以下の入力フィールドは、非 HTTP 要求用のアナライザー・プログラムでは未定義になっています。

- HTTP バージョン
- メソッド
- 要求のパス構成要素
- 要求ヘッダー

したがって、要求の内容を調べることによって、以後の処理ステージを決定する必要があります。

アナライザー・プログラムは、コンバーター・プログラムまたは Web 対応アプリケーション・プログラムによる後続の要求処理を指定できます。 444 ページの『アナライザー・プログラムの作成』で、アナライザー・プログラムの入出力について、およびそれらが要求処理の決定にどのように使用されるかについて説明しています。

非 HTTP 要求用のアプリケーション・プログラミング

非 HTTP 要求用のアプリケーション・プログラムでは、**EXEC CICS WEB** プログラミング・インターフェースの特定の要素を使用することができます。また、これらのプログラムは Web 非対応アプリケーションであってもよく、コンバーター・プログラムによってエンコードされた出力を生成することができます。

非 HTTP 要求には、疑似会話型プログラミング・モデルは適しません。単一の要求を受け取って単一の応答を提供するようにアプリケーションを設計します。

Web 対応アプリケーション

Web 対応アプリケーションを使用して非 HTTP 要求に応答するには、以下の CICS API コマンドを使用します。

- WEB RECEIVE コマンドは、非 HTTP 要求を受け取ることができます。HTTP 要求と非 HTTP 要求の両方に応答するアプリケーションの場合は、WEB RECEIVE コマンドの TYPE オプションで、その 2 つの要求タイプを区別できます。CICS は、非 HTTP メッセージに対して構文解析を行いません。ネットワーク上の伝送用に分割された要求は、自動的にはアセンブルされません。アナライザー・プログラムが要求をアセンブルしても、その結果は CICS WEB API コマンドには見えません。
- **EXEC CICS DOCUMENT** コマンドは、CICS 文書を組み立てて応答のボディを形成することができます。
- WEB SEND コマンドは、非 HTTP クライアントに応答を送信します。ただし、HTTP 固有のアクションに関連する次のオプションは適していません。
  - STATUSCODE および STATUSTEXT は無視されます。
  - CLOSESTATUS は無視されます。
  - CHUNKING を指定すると、コマンドのエラーになります。
- WEB RETRIEVE コマンドは、**EXEC CICS** WEB SEND コマンドによって既に送信された CICS 文書を取得します。

その他の **EXEC CICS** WEB コマンドは、HTTP 要求にのみ関連するものであり、非 HTTP 要求に使用した場合は INVREQ 状態になることがあります。

アプリケーション・プログラムでは、 **WEB RECEIVE** コマンドを使用することで、非 HTTP 要求のコード・ページ変換を指定できます。

コンバーター・プログラムを使用する非 Web 対応アプリケーション

非 Web 対応アプリケーションでは、コンバーター・プログラムを使用して、Web クライアントからの入力をアプリケーションに適した COMMAREA に変換し、アプリケーションからの出力を HTML に変換して応答を提供することができます。ネットワーク上の伝送用に分割された要求をアナライザー・プログラムが再構成した場合、その結果をコンバーター・プログラムに渡すことができます。

HTTP 要求に関連する以下の入力フィールドは、非 HTTP 要求用のコンバーター・プログラムでは未定義になっています。

- HTTP バージョン
- メソッド
- URL のパス構成要素
- 要求ヘッダー

詳細については、 465 ページの『付録 F. コンバーター・プログラム』を参照してください。

第 13 章 CICS Web サポートおよび 3270 表示アプリケーション

Web クライアントから 3270 トランザクションにアクセスすると、 CICS は出力を HTML フォームで表示できます。 Web 端末変換アプリケーションのバリアント (DFHWBTTA、DFHWBTTB、または DFHWBTTC) を使用して、本来 3270 表示システムを使用するように設計されたアプリケーションへのアクセスを、Web クライアントに提供します。3270 アプリケーション用 CICS Web サポートは、端末管理コマンドの SEND、CONVERSE、および RECEIVE をサポートしています。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

次の 2 とおりの方法のいずれかで、3270 トランザクションの出力から HTML フォームを作成できます。

- BMS を使用するアプリケーションの場合、HTML テンプレートは BMS マップから生成され、テンプレート・ライブラリーに保管されます。テンプレートの生成をカスタマイズすることができます。ただし、生成された HTML に加えるべき変更が、ヘッディング・セクションまたはフッティング・セクションに対する変更のみの場合は、BMS マップからテンプレートを生成する必要はありません。実行時にマップを処理して、HTML フォームを生成できるからです。
- BMS を使用しないアプリケーションの場合は、実行時にアウトバウンド 3270 データ・ストリームを処理して HTML フォームを生成します。

Web 端末変換アプリケーションを使用して、Web ブラウザーに HTML フォームを表示できます。

注: Web 端末変換アプリケーションは HTTP/1.0 レベルで動作します。このアプリケーションは、**EXEC CICS** WEB API など、CICS Web サポートで使用できる機能をフルに利用するわけではないので、HTTP/1.1 仕様には準拠していません。

- Web クライアントからの要求、およびアプリケーションからの応答は、HTTP プロトコル仕様に基づいてチェックされません。
- クライアントが HTTP/1.1 レベルであっても、CICS は、通常状態またはエラー状態のいずれの場合にも HTTP/1.1 応答を提供しません。

Web 端末変換アプリケーションの 3 つのバリアントはすべて、非会話型トランザクション、会話型トランザクション、および疑似会話型トランザクションをサポートしています。

- DFHWBTTA および DFHWBTTB は、3270 データ・ストリームと HTML 間、および BMS マップから生成したテンプレートと HTML 間の変換を行います。ご使用の HTML テンプレートのデータが 32,767 バイト (32 KB) 以下の場合は DFHWBTTA を使用し、HTML テンプレートが 32 KB より大きい場合は DFHWBTTB を使用します (小さい HTML テンプレートに DFHWBTTB を使用すると、無用の性能低下を招きます)。

- DFHWBTTC は、テンプレートが生成されない場合に、BMS マップと HTML 間の変換を行います。ここで使用される BMS マップでは、TERM=3270 を指定するか、TERM パラメーターを省略する必要があります。 DFHWBTTC は、任意の長さの HTML 出力をサポートします。 HTML テンプレートを生成する必要のない場合は、DFHWBTTC を使用してください。

DFHWBTTB と DFHWBTTC は DFHWBTTA の別名です。いずれの場合にも DFHWBTTA が呼び出されます。 CICS は、プログラムが呼び出されるときに用いられる名前を使用して、必要な処理を判別します。

DFHWBTTA、DFHWBTTB、および DFHWBTTC は、HTML 3.2 仕様に準拠した HTML を生成します。 HTML 3.2 をサポートしていない Web ブラウザーを使用すると、一部の機能が正しく働かないことがあります。

ページ・サイズが 4095 (x'FFF') より大きなフィールド位置になる、端末用に生成された HTML は、正しく動作しない場合があります。特に DFHWBTTC を使用する場合にこのことが当てはまります。古いスタイルのテンプレートを使用する場合は例外です。 (古いスタイルのテンプレートとは、 PTF UQ53534 以前の CICS TS 1.2 または CICS TS 1.3 の DFHWBTLG によって生成されたものです)。 DFHWBTTA または DFHWBTTB を使用する際に (DFHWBTTC を使用する際にではない) このようなテンプレートの BMS 送信を許容するため、コードが提供されています。

要求を処理するために呼び出すプログラムとして DFHWBTTA、 DFHWBTTB、または DFHWBTTC を指定する (PROGRAM 属性)、URIMAP 定義を作成することができます。プログラムへのアクセスに Web クライアントが使用する方式は類似していますが、URIMAP 定義を使用すると、要求を防止したりリダイレクトしたりするために使用できるオンライン管理機能が得られます。URIMAP 定義を使用する場合、アナライザー・プログラムを使用するかどうかはオプションです。 196 ページの『3270 表示アプリケーションの URL パス構成要素』を参照してください。

3270 アプリケーション用 CICS Web サポートは、端末管理コマンドの SEND、CONVERSE、および RECEIVE をサポートしています。また、最小関数 BMS および SEND TEXT コマンドもサポートしています。 SEND コマンドと CONVERSE コマンドの DEFRESP オプションは無視されます。アプリケーション・リカバリーが影響を受ける可能性があります。

CICS Web サポートは、区分、論理装置コード、磁気スロット読取装置、外部形式設定機能、またはその他のハードウェア機能をサポートしていません。ライト・ペン・サポートのある検出可能フィールドを使用できます。

3270 アプリケーション・プログラムの CICS Web サポート処理

CICS Web サポートは、端末向けトランザクションをこの順序で処理します。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

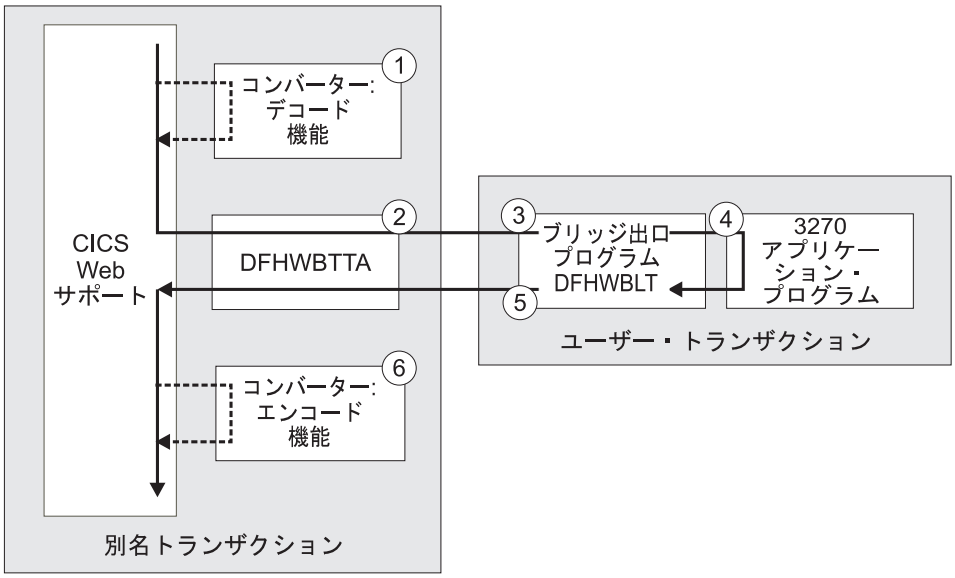


図 7. CICS Web サポートと 3270 アプリケーション・プログラムとの対話の仕組み

以下のステップを図に示しています。

1. オプションで、コンバーター・プログラムは、DFHWBTTA プログラムに渡す入力データを作成します。
2. DFHWBTTA はユーザー・トランザクションのタスクを生成し、DFHWBLT をブリッジ出口プログラムとして指定し、 DFHWBLT からの応答を待ちます。このトランザクションは、3270 ブリッジ環境で実行されます。
3. ブリッジ出口ルーチンは、アプリケーション・プログラムのための 3270 環境をセットアップします。
4. このアプリケーション・プログラムは入力データを処理し、3270 出力データを作成します。
5. ブリッジ出口ルーチンは 3270 出力データを解釈し、HTTP 応答を DFHWBTTA に渡します。
6. オプションで、コンバーター・プログラムは、Web クライアントに渡す出力データを変更します。

CICS Web サポートを 3270 アプリケーションで使用すると、アプリケーション・プログラムはそれ自身のトランザクションのもとで実行され、別名トランザクションのもとでは実行されません。

3270 ブリッジの詳細については、「*CICS External Interfaces Guide*」のBridging to 3270 transactionsを参照してください。

3270 表示アプリケーションの URL パス構成要素

CICS 3270 アプリケーションを Web ブラウザーから呼び出すには、アプリケーション・プログラム名 DFHWBTTA、DFHWBTTB、または DFHWBTTC を、適切な別名トランザクションおよびコンバーター・プログラム (必要な場合) とともに呼び出すことから始めるパス構成要素を含んだ URL を入力します。この別名トランザクションは、独自のトランザクションのもとで実行される 3270 アプリケーションには適用されません。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

アナライザー・プログラムの使用

CICS 提供のサンプル・アナライザー DFHWBADX のようなアナライザー・プログラムを使用して要求を処理する場合は、 URL のパス構成要素にはアプリケーション・プログラムの名前 (DFHWBTTA、 DFHWBTTB、または DFHWBTTC) が含まれます。また、使用するコンバーター・プログラムの名前、要求処理用の別名トランザクションの名前 (CICS 提供のデフォルトの別名トランザクション CWBA など) も含まれます。 453 ページの『CICS 提供のサンプル・アナライザー・プログラム DFHWBADX』で説明されているように、パスのこれらの要素はアナライザー・プログラムによって抽出され、以後の処理段階を開始するために使用されます。

URIMAP 定義の使用

URIMAP 定義を使用して要求を処理する場合は、URL のパス構成要素は PATH 属性で指定します。 URIMAP 定義を使用する場合、URL のパス構成要素には、アプリケーション・プログラム、コンバーター・プログラム、および別名トランザクションに関する明示的な情報を含める必要はありません (ただし、含めることもできます)。これらの要素はすべて、URIMAP 定義の PROGRAM、CONVERTER、および TRANSACTION 属性を使用して指定できます。これにより、パス構成要素のこの部分を任意のパスに置き換えることができます。DFHWBTTA の要件を満たすには、URIMAP 定義で指定するパスの最後でワイルドカード文字としてアスタリスクを使用します。ワイルドカードを使用することで、パス構成要素の残りの部分が可変になり、DFHWBTTA を制御できるようになります。

URIMAP 定義とアナライザー・プログラムの両方の使用

URIMAP 定義で ANALYZER(YES) オプションを指定して、要求の処理パスでアナライザー・プログラムを使用できます。アナライザー・プログラムは、URIMAP 定義で指定されたコンバーター・プログラム、別名トランザクション ID、およびプログラム名を動的に変更できます。 DFHWBTTA は、これらの変更を認識することができます。

アプリケーション・プログラムを呼び出すのに必要な情報を提供した後、URL のパス構成要素の次の部分は、以下の制御情報を DFHWBTTA に提供します。

- 非定様式モードが使用されるかどうかを指定するキーワード。
- 使用する 3270 アプリケーションのトランザクション ID。
- 指定したトランザクションの入力パラメーター (区切り文字として正符号 (+) を使用)。

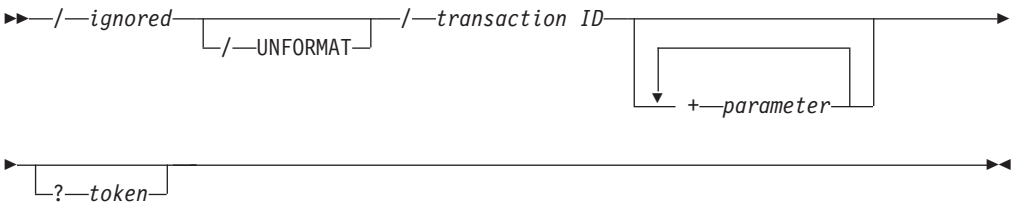


図 8. DFHWBTTA によって解釈されるパスの構文

DFHWBTTA は、 URL のパス構成要素を以下のように解釈します。

ignored
パスの最初の部分は DFHWBTTA では無視されます。この部分は、アプリケーション・プログラムの呼び出しに必要な情報を提供するために、アナライザーによって解釈されるか、URIMAP 定義と突き合わされます。

UNFORMAT
3270 表示システムは、定様式モードと非定様式モードの 2 つのモードで操作できます。このキーワードが存在していれば、 DFHWBTTA は 3270 表示操作を非定様式モードでシミュレートします。このキーワードが省略されていれば、DFHWBTTA は 3270 表示操作を定様式モードでシミュレートします。

詳しくは、「*CICS アプリケーション・プログラミング・ガイド*」の不定形式モードを参照してください。

トランザクション ID
初期要求で、この情報は実行される CICS トランザクションを指定します。パスのこの要素は継続要求では無視されます。

parameter
トランザクションに対する入力パラメーターを指定します。トランザクション ID とこのデータ、およびこのデータの要素間を区切るために、区切り文字としてスペースでなく正符号 (+) を使用します。

token
DFHWBTTA はこの情報を無視します。アナライザー・プログラムによって使用される場合があります。

常にこの形式で URL をコーディングしてください。

例えば、CICS 提供の DFHWBADX アナライザー・プログラムを使用する場合は、以下の URL パスを使用して CEMT INQ TAS コマンドを出すことができます。

```
/cics/cwba/dfhwbtta/CEMT+INQ+TAS
```

- `cics` は、コンバーター・プログラムが必要でないことを示すために使用されます。
- `cwba` は、要求処理用の別名トランザクションの名前です。
- `dfhwbtta` はアプリケーション・プログラムの名前です。
- `CEMT+INQ+TAS` は、CEMT トランザクションにアクセスし、 INQ TAS コマンドを出すように DFHWBTTA に指示します。

あるいは、次の属性を含む URIMAP 定義を設定することもできます。

```
Path:        /terminal/*
Transaction:  CWBA
Program:      DFHWBTTA
```

この URIMAP 定義が使用可能になると、以下の URL パスを使用して CEMT INQ TAS コマンドを出すことができます。

```
/terminal/CEMT+INQ+TAS
```

- terminal は URIMAP 定義に一致し、別名トランザクションとアプリケーション・プログラムの名前を示します。
- CEMT+INQ+TAS は、URIMAP 定義では無視されますが、CEMT トランザクションにアクセスし、 INQ TAS コマンドを出すように DFHWBTTA に指示します。

初期要求と継続要求

DFHWBTTA は、2 つのタイプの HTTP 要求 (初期要求と継続要求) をそのトランザクション内のコンテキストによって識別します。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

初期要求

初期要求は CICS トランザクションを開始します。以下のいずれかの方法で初期要求を送信します。

- URL を明示的に入力する。
- HTML ページからリンクを選択する。
- HTML フォーム内のボタンを選択する。フォーム内に入力したデータはすべて無視されます。

継続要求

継続要求は既存の CICS トランザクションを継続します。以下のようにして継続要求を送信します。

- 前の要求に対する応答として表示された HTML フォーム内のボタンを選択する。

継続要求は HTML POST メソッドを使用します。フォームのデータは HTML 要求のエンティティー・ボディとして伝送されます。

Web クライアントと CICS の間で対話が何回か行われる会話型または疑似会話型トランザクションでは、1 つの初期要求の後に 1 つ以上の継続要求が続きます。対話が 1 回だけの単純なトランザクションでは、初期要求が 1 つで継続要求はありません。

初期要求によって表示され、その後に続く要求によって返される HTML フォーム内の隠れ要素 (DFH_STATE_TOKEN) は、初期要求と継続要求を区別し、継続要求を適切なトランザクションに関連付けます。

継続要求のトランザクション ID

継続要求の場合、URL はその前の要求で表示された形式でコーディングされます。ただし、継続要求の URL に組み込まれたトランザクション ID は無視されます。

その代わり、トランザクションの決定は次のようにして行われます。

- 継続要求が会話型トランザクションの一部であれば、同じトランザクションが実行を継続します。
- 継続要求が疑似会話型トランザクションの一部であれば、以下のように異なるトランザクション ID が使用されます。
  - TRANSID オプションを指定した **EXEC CICS** RETURN コマンドで直前のトランザクションが終了した場合は、指定したトランザクション ID が使用されます。
  - その前のトランザクションで **EXEC CICS** RETURN コマンドにトランザクション ID を指定せず、 AID がトランザクション ID に関連付けられている場合は、そのトランザクション ID が使用されます。
  - **EXEC CICS** RETURN コマンドでトランザクション ID が指定されておらず、AID にトランザクション ID が関連付けられていない場合、CICS は HTML フォームからトランザクション ID を取得します。

HTML フォーム内のトランザクション ID

トランザクションを 3270 表示システムから生成すると、CICS は、そのトランザクション ID が 3270 データ・ストリームの最初の変更フィールドに入っているものと見なします。

Web クライアントがフォームのデータを伝送する順序は必ずしも予測できないため、CICS は、以下のようにして、フォーム・フィールドの名前と 3270 画面上の対応する位置との間のマッピングを使用します。

- BMS マップを使用しないトランザクションの場合は、マッピングはフィールド名を直接使用します。それは、その名前が 3270 画面上のフィールドの位置を反映しているからです。
- BMS マップを使用するトランザクションの場合は、フィールド名は必ずしも 3270 画面上の位置を反映していないため、間接マッピングが使用されます。このマッピングでは、隠れ変数 DFH_NEXTTRANSID.*n* を使用します。 BMS マップから HTML テンプレートが作成されると、最大 5 個の変数が作成されます。各変数の値は入力フィールドの名前であり、 3270 バッファー位置の順序になっています。

  CICS は、HTTP 要求を受信すると、各 DFH_NEXTTRANSID フィールドを順番に調べて、参照先の入力フィールドの名前を判別するほか、 HTTP 要求にそのフィールドの値が入っているかどうかも判別します。値が入っていれば、ユーザーが要求を変更したということであり、したがって、要求には次のトランザクションのトランザクション ID が入っていると見なされます。

  いくつかの BMS および非 BMS SEND コマンドからの出力をマージすることによって画面を構成すると、場合によっては入力フィールドが抑止されます。詳細については、 219 ページの『フッティング・セクションの選択方法』を参照してください。 CICS が 3270 データ・ストリーム内のトランザクション ID を正しく識別できるように、トランザクション ID が含まれている可能性のある入力フィールドが、マージ後の HTML ページで抑止されないようにする必要があります。

BMS マップから生成した HTML テンプレート

3270 表示システムと HTML フォームには、多くの類似点があります。 BMS マップ・テンプレートは、3270 表示の機能を表すことができます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

3270 表示システムの機能は、以下のように、多くの点で HTML フォームに類似しています。

- どちらの場合も、表示域に固定テキストと、ユーザーがデータを入力できる領域を格納することができます。
- 3270 キーボードの AID キーは、HTML フォームで表示される各種のボタンに類似した機能を持っています。
- どちらの場合も、ユーザーがデータ入力フィールドの内容を変更したかどうかを検出できます。

BMS マップから生成されたテンプレートには、以下のような、3270 表示システムの各種機能を示す多くの要素が含まれています。

- マップ内の保護フィールドは、通常の HTML テキストとして表示されます。
- マップ内の無保護フィールドは、テキスト入力要素として表示されます。 CICS は、各要素に次のような 2 部構成の名前を付けます (最大長は 32 文字)。
  - 名前の最初の部分は 11 文字の長さを持ち、次のような形式になっています。

    F*rrcclllll*_

    ここで、

    - *rr* は、フィールドが表示されている 3270 画面上の行を示す 2 桁の数字です。
    - *ccc* は、フィールドが表示されている 3270 画面上の列を示す 3 桁の数字です。
    - *llll* は、フィールドの長さを示す 4 桁の数字です。
  - マップに指定されている BMS フィールドの場合、2 番目の部分は、マップで使用されている名前からなり、必要な場合は、21 文字に切り捨てられます。
  - 指定されていない BMS フィールドの場合、2 番目の部分は DFH_*nnnn* の形式になっています。ここで、*nnnn* は 4 桁の数字です。これらのフィールドには、 BMS マップに配置されるときに順序番号が付けられます。

  例えば、指定されていない 3 番目の無保護フィールドが画面の 2 行目、11 列目にあり、長さが 16 文字であると仮定します。生成される 2 つの部分の名前は、次のようになります。

  ```
  F020110016_DFH0003
  ```

  ここで、同じフィールドが BMS マップに TOTAL_MONTHLY_PURCHASES という名前を持っているとします。この HTML 要素に対して CICS が生成する名前は、次のようになります。

  ```
  F020110016_TOTAL_MONTHLY_PURCHAS
  ```

注: フィールドが 3270 画面上に表示される順序は、それらが BMS マップ定義にコーディングされている順序と異なる場合があります。対応するテンプレートが Web クライアントに表示されている場合は、それらのフィールドはコーディングの順序で表示されます。

- 3270 表示システムでサポートされている各アテンション・キーは、サブミット・ボタンとしてシミュレートされます。それらのボタンは、以下のような名前になっています。
  - DFH_PF01 から DFH_PF24 まで
  - DFH_PA1 から DFH_PA3 まで
  - DFH_ENTER、DFH_CLEAR

  エンド・ユーザーがこれらのボタンのいずれかを選択すると、対応する変数が HTTP 要求として送信されます。 CICS は、この変数を使用して、どの AID を 3270 アプリケーションでシミュレートするかを決定します。

  さらに、DFH_PEN というサブミット・ボタンが検出可能フィールドで使用されます。
- 検出可能フィールドは、先頭にチェック・ボックスを持つテキスト要素としてシミュレートされます。 209 ページの『検出可能フィールドの使用』を参照してください。
- 隠れ要素 (DFH_STATE_TOKEN) は、アプリケーションが Web クライアントとの多くの対話を通して確認した表示状態を維持するために使用されます。
- 隠れ要素 (DFH_CURSOR) と JavaScript 関数 (dfhinqcursor()) の連携により、カーソル位置がアプリケーションに戻されます。
- 一連の隠れ要素 (DFH_NEXTTRANSID.1 から DFH_NEXTTRANSID.*n*) は、 Web クライアント・フィールドに入力されたトランザクション ID を取り込むために使用されます。

3270 データ・ストリームから生成された HTML ページ

BMS を使用しないアプリケーションの場合、 CICS Web サポートは、ヘッディング・セクション、画面イメージ・セクション、およびフッティング・セクションの 3 つの部分からなる HTML ページを生成します。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

ヘッディング・セクション

CICS Web サポートは、以下のようなヘッディング・セクションを生成します。

```
<!doctype html public "-//W3C//DTD HTML 3.2//EN">
<html>
<STYLE TYPE="text/css">
<!--
    TABLE, TR, TD
    { padding: 0mm    }
    TABLE
    { width: 60%    }
-->
.BRIGHT
{font-weight: bold}
```

```
{font-family: courier}
.INPUT
{font-family: courier}
</STYLE>
<head>
<title>CICS web support screen emulation - tranid</title>
<meta name="generator" content="CICS Transaction Server/2.2.0">
<script language="JavaScript">
<!--
 function dfhsetcursor(n)
  {for (var i=0;i<document.form3270.elements.length;i++)
     {if (document.form3270.elements[i].name == n)
         {document.form3270.elements[i].focus();
          document.form3270.DFH_CURSOR.value=n;
          break}}}
 function dfhinqcursor(n)
   {document.form3270.DFH_CURSOR.value=n}
 // -->
</script>
</head>
<body onLoad="dfhsetcursor('&DFH_CURSORPOSN;')">
```

独自のヘッディング・セクションを設けることで、ページの外観を変更することができます。 205 ページの『DFHWBTTA からの出力の変更』を参照してください。

画面イメージ・セクション

HTML ページのこのセクションは、3270 画面イメージの内部表現から直接生成されます。そのサイズは、トランザクションに関連付けられた FACILITYLIKE 端末定義の DEFSCREEN 定義と ALTSCREEN 定義から決定されます。このセクションには、以下の要素が含まれています。

通常の HTML テキスト
保護フィールドをシミュレートします。

テキスト入力要素
無保護フィールドをシミュレートします。各要素に 11 文字の名前が付けられます。形式は次のとおりです。

`F`*`rrcccllll`*`_`

ここで、

- *rr* は、フィールドが表示されている 3270 画面上の行を示す 2 桁の数字です。
- *ccc* は、フィールドが表示されている 3270 画面上の列を示す 3 桁の数字です。
- *llll* は、フィールドの長さを示す 4 桁の数字です。

例えば、

- 3270 表示システム上の 1 行目、1 列目のフィールドで、78 バイトの長さを持つフィールドは、次のように指定されます。

  `F010010078_`

チェック・ボックスを持つテキスト要素
検出可能フィールドをシミュレートします。 209 ページの『検出可能フィールドの使用』を参照してください。

隠れ要素

DFH_STATE_TOKEN という隠れ要素は、アプリケーションが Web クライアントとのいくつかの対話を通して確認した表示状態を維持します。

隠れ要素 (DFH_CURSOR) と JavaScript 関数 (dfhinqcursor()) の連携により、カーソル位置がアプリケーションに戻されます。 CICS は JavaScript focus() メソッドを使用して、 DFH_CURSOR によって指定された入力ボックスまたはフィールドでのカーソル位置を決定します。 focus() は、入力ボックスまたはフィールドの特定の文字上でカーソル位置を決定することはできません。決定できるのは、カーソルの最初の文字位置のみです。

```
<!doctype html public "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title>CICS web support screen emulation - tranid</title>
<meta name="generator" content="CICS Transaction Server/2.1.0">
<script language="JavaScript">
<!--
 function dfhsetcursor(n)
  {for (var i=0;i<document.form3270.elements.length;i++)
     {if (document.form3270.elements[i].name == n)
         {document.form3270.elements[i].focus();
          document.form3270.DFH_CURSOR.value=n;
          break}}}
 function dfhinqcursor(n)
   {document.form3270.DFH_CURSOR.value=n}
 // -->
</script>
</head>
<body onLoad="dfhsetcursor('&DFH_CURSORPOSN;')">
```

3270 画面イメージから生成された HTML は、BMS マップのテンプレートで生成された HTML に類似しています。ページ上の情報の水平位置合わせや垂直位置合わせは、HTML テーブルを使用して行われます。

- HTML テーブルは、フィールドの開始点を含む 3270 画面の各列に対して、1 つの列を保持します。例えば、列 2、11、21、および 55 で始まるフィールドが 3270 画面にある場合、HTML テーブルには 4 つの列が含まれます。したがって、3270 画面で開始位置が縦方向に配置されているフィールドはすべて、HTML ページでも縦方向に配置されます。
- HTML テーブルは、フィールドの開始点を含む 3270 画面の各行に対して、1 つの行を保持します。したがって、3270 画面で開始位置が横方向に配置されているフィールドはすべて、HTML ページでも横方向に配置されます。フィールドが含まれていない 3270 画面上の行は、HTML テーブルには示されません。
- テーブルでは、テキストはプロポーショナル・フォントで表示されます。

以下のフィールドが含まれている 3270 画面を考えます。

フィールド	行	開始列
Field_1	2	2
Field_2	3	2
Field_3	3	35
Field_4	4	2
Field_5	4	35
Field _6	9	2

フィールド	行	開始列
Field_7	9	18
Field_8	9	35

すべてのフィールドは、3270 画面の列 2、18、または 35 から開始されます。したがって、その結果作成される HTML テーブルには 3 つの列が設けられます。同様に、すべてのフィールドが 3270 画面の行 2、3、4、または 9 に位置指定されているため、HTML テーブルには 4 つの行が設けられます。

コンバーター・プログラムのエンコード機能を使用して、画面イメージ・セクションを変更することができます。 207 ページの『DFHWBTTA でのコンバーター・プログラムの使用』を参照してください。

フッティング・セクション

CICS Web サポートは以下のようなフッティング・セクションを生成します。3270 表示システムでサポートされている各アテンション・キーは、サブミット・ボタンとしてシミュレートされます。ユーザーがボタンを選択すると、対応する変数が HTTP 要求で送信されます。 CICS は、この変数を使用して、どの AID を 3270 アプリケーションでシミュレートするかを決定します。さらに、DFH_PEN というサブミット・ボタンが検出可能フィールドで使用されます。

```
<input type="submit" name="DFH_PF1"  value="PF1">
<input type="submit" name="DFH_PF2"  value="PF2">
<input type="submit" name="DFH_PF3"  value="PF3">
<input type="submit" name="DFH_PF4"  value="PF4">
<input type="submit" name="DFH_PF5"  value="PF5">
<input type="submit" name="DFH_PF6"  value="PF6">
<input type="submit" name="DFH_PF7"  value="PF7">
<input type="submit" name="DFH_PF8"  value="PF8">
<input type="submit" name="DFH_PF9"  value="PF9">
<input type="submit" name="DFH_PF10"  value="PF10">
<input type="submit" name="DFH_PF11"  value="PF11">
<input type="submit" name="DFH_PF12"  value="PF12">
<br>
<input type="submit" name="DFH_PF13"  value="PF13">
<input type="submit" name="DFH_PF14"  value="PF14">
<input type="submit" name="DFH_PF15"  value="PF15">
<input type="submit" name="DFH_PF16"  value="PF16">
<input type="submit" name="DFH_PF17"  value="PF17">
<input type="submit" name="DFH_PF18"  value="PF18">
<input type="submit" name="DFH_PF19"  value="PF19">
<input type="submit" name="DFH_PF20"  value="PF20">
<input type="submit" name="DFH_PF21"  value="PF21">
<input type="submit" name="DFH_PF22"  value="PF22">
<input type="submit" name="DFH_PF23"  value="PF23">
<input type="submit" name="DFH_PF24"  value="PF24">
<br>
<input type="submit" name="DFH_PA1"   value="PA1">
<input type="submit" name="DFH_PA2"   value="PA2">
<input type="submit" name="DFH_PA3"   value="PA3">
<input type="submit" name="DFH_CLEAR" value="Clear">
<input type="submit" name="DFH_ENTER" value="Enter">
<input type="submit" name="DFH_PEN"   value="Pen">
<input type="reset"  value="Reset">
</form>
</body>
</html>
```

独自のフッティング・セクションを設けることで、ページの外観を変更することができます。『DFHWBTTA からの出力の変更』を参照してください。

DFHWBTTA からの出力の変更

DFHWBTTA からの出力は、HTML をカスタマイズしたり独自のヘッディング・セクションやフッティング・セクションを設定したりすることにより、変更できます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

このタスクについて

BMS を使用するアプリケーションの場合は、BMS マップから作成された HTML テンプレートをカスタマイズすることができます。214 ページの『カスタマイズ済み HTML テンプレートの生成』を参照してください。

非 BMS アプリケーションの場合、および DFHWBTTC を使用して呼び出した BMS アプリケーションの場合は、独自のヘッディング・セクションとフッティング・セクションを設けることで、ページの外観を変更することができます。画面イメージ・セクションを直接変更することはできません。ただし、ヘッディング・セクションにタグを挿入すると、その後のセクションの外観に影響を与えることがあります。

ユーザー独自のヘッディング・セクションとフッティング・セクションを設けるには、以下のテンプレートを 1 つ以上定義してインストールします。これらの名前は、DOCTEMPLATE 定義の TEMPLATENAME フィールドで定義されます。

*tran***HEAD**
: このテンプレートは、トランザクション *tran* の出力となる HTML ページの先頭に挿入されます (このテンプレートがインストールされている場合)。

CICSHEAD
: このテンプレートは、対応する *tran*HEAD テンプレートがインストールされていないトランザクションの出力となる HTML ページの先頭に挿入されます。

*tran***FOOT**
: このテンプレートは、トランザクション *tran* の出力となる HTML ページのフッターに挿入されます (このテンプレートがインストールされている場合)。このテンプレートがインストールされていない場合は、代わりに CICSFOOT が使用されます。

CICSFOOT
: このテンプレートは、対応する *tran*FOOT テンプレートがインストールされていないトランザクションの出力となる HTML ページのフッターに挿入されます。

文書テンプレート作成の詳細については、「*CICS アプリケーション・プログラミング・ガイド*」の Programming with documents and document templates を参照してください。

CICS Web サポートが生成するヘッディング・セクションは、EBCDIC ローマ字セット (コード・ページ 037) を使用します。 CICS システムで別のコード・ページを使用する場合は、以下のように、独自のコード・ページを使用して、同様のヘッディング・セクションを作成する必要があります。

1. ヘッディング・セクションを含む文書テンプレート CICSHEAD を作成する。
2. テンプレートに DOCTEMPLATE 定義を定義およびインストールする。

CICS 生成のヘッディング・セクションで使用される以下の文字は、037 とは別のコード・ページに別の表記法を持っています。

! [ ] { }

独自のヘッディング・テンプレートの提供

独自のヘッディング・テンプレートを提供する場合は、HTML ページの必須要素のいくつかを提供する必要があります。

このタスクについて

ヘッディング・テンプレートは、通常以下の HTML 要素を含みます。

- doctype タグ。例えば、次のとおりです。

  ```
  <!doctype html public "-//W3C//DTD HTML 3.2//EN>
  ```

- <html> タグ。
- <head> タグ。
- <STYLE> タグ。BRIGHT クラスおよび INPUT クラスのスタイル・シート規則を含む必要があります。例えば、次のとおりです。

  ```
  <STYLE TYPE="text/css">
  <!--
      TABLE, TR, TD
      { padding: 0mm    }
      TABLE
      { width: 60%    }
  -->
  .BRIGHT
  {font-weight: bold}
  {font-family: courier}
  .INPUT
  {font-family: courier}
  </STYLE>
  ```

  TABLE 要素の width 属性を使用して、画面イメージ・セクションの外観を細密に調整することができます。

- </head> タグ。
- <body> タグ。このタグを使用して、テキストの色を指定したり、ページのバックグラウンドとして使用するイメージを指定することができます。例えば、次のとおりです。

  ```
  <body background="/dfhwbimg/background2.gif" bgcolor=#FFFFFF"
   text="#000000" link="#00FFFF" vlink="#800080" alink="#FF0000>
  onLoad="dfhsetcursor('&DFH_CURSORPOSN;')"
  ```

  注: この例では、 210 ページの『DFHWBIMG を使用したグラフィックスの表示』で説明している DFHWBIMG を使用しています。

- ページのカスタマイズに必要な他の任意の HTML 要素 (オプション)

独自のフッティング・テンプレートの提供

独自のフッティング・テンプレートを提供する場合は、HTML ページの必須要素のいくつかを提供する必要があります。

このタスクについて

フッティング・テンプレートは、通常以下の HTML 要素を含みます。

- プログラム・ファンクション・キーまたは Enter キーを表す入力ボタン。例えば、次のとおりです。

```
<input type="submit" name="DFH_PF1"  value="Help">
<input type="submit" name="DFH_PF3"  value="Quit">
<input type="submit" name="DFH_ENTER"  value="Continue">
```

  これらのボタンは、CICS によって開始される HTML フォームの一部を形成します。これらのボタンをユーザーが選択すると、201 ページの『3270 データ・ストリームから生成された HTML ページ』で説明されている AID 標識が、そこに記述されている名前で作成されます。 *value* パラメーターは、生成されたボタンに表示する凡例を指定します。これは、DFHWBTTA では使用されません。
- `</form>` タグ。
- ページのカスタマイズに必要な他の任意の HTML 要素 (オプション)。
- ページをクローズするための `</body>` タグ。
- `</html>` タグ。

DFHWBTTA でのコンバーター・プログラムの使用

コンバーター・プログラムのデコード機能を使用して、DFHWBTTA に渡された要求を変更することができます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

このタスクについて

以下の方法で要求を変更できます。

- アテンション・キーを表すボタンのいずれかを使用して、クライアントが HTML フォームを送信すると、その要求には選択されたボタンを示すフィールドが組み込まれます。その要求を変更することにより、別のアテンション・キーを使用した場合の結果をシミュレートすることができます。このフィールドの値を希望のアテンション・キーに変更するか、または Web クライアントによって送られたフィールドの後に新しいフィールドを挿入します。
- クライアントが HTML フォームを送信すると、カーソルが含まれているフィールドの名前が `DFH_CURSOR` フィールドに組み込まれます。その要求を変更することにより、別のカーソル位置を使用した場合の結果をシミュレートすることができます。 `DFH_CURSOR` フィールドの値に別のフィールド名を含めるように変更するか、または Web クライアントによって送られた `DFH_CURSOR` フィールドの後に新しいフィールドを挿入します。
- 継続要求の中の `DFH_NEXTTRANSID.`*n* 変数を変更することにより、その次のトランザクション ID を選択することができます。変数を挿入または削除したり、それ

らのいずれかの値を変更したりできます。これらのフィールドを使用して次のトランザクション ID を決定する方法の詳細については、 199 ページの『HTML フォーム内のトランザクション ID』を参照してください。

DFH_STATE_TOKEN の値は変更しないでください。

コンバーター・プログラムのエンコード機能を使用して、DFHWBTTA からの出力を変更することができます。

- 応答は、バッファーの長さを指定する 32 ビットの符号なし番号で始まっているバッファーの中に入っています。バッファーの残りの部分は HTTP 応答です。応答の中の HTML は、トランザクション・プログラムからの出力 BMS マップまたは 3270 データ・ストリームに対応するものです。
- HTTP 応答の HTTP ヘッダーは、DFHWBTTA によって自動的に生成されます。以下のヘッダーは、DFHWBTTA によって生成されます。
  - Content-type: text/html
  - Content-length: <エンティティー・ボディの長さ>
  - Pragma: no-cache
  - Connection: Keep-Alive (この接続が HTTP 1.0 持続接続の場合)

  追加ヘッダーが必要な場合は、コンバーターのエンコード機能を使用して HTTP 応答に追加します。

検出可能フィールドの使用可能化

CICS Web サポート 3270 ブリッジを介した検出可能フィールド処理を使用可能にするには、ライト・ペン・サポートを使用可能にしたブリッジ機能を定義します。

このタスクについて

この機能を定義するには、次のようにします。

手順

1. 以下の定義を新規グループにコピーします。 CICS システムで実行するすべてのアプリケーションがライト・ペン・サポートを必要とする場合を除き、両方の定義の名前変更も行います。
   - CICS 提供の CBRF ブリッジ機能 (DFHTERM グループ内)。
   - そのデフォルト TYPETERM、DFHLU2 (DFHTYPE グループ内)。
2. TYPETERM 定義では、DEVICE PROPERTIES の下の LIGHTPEN オプションを YES に変更します。
3. TERMINAL 定義では、TYPETERM パラメーターが新規の TYPETERM をポイントするように変更します。
4. これらの定義を CICS 領域にインストールします。
5. 新規のブリッジ機能定義を作成した場合は、新規の TERMINAL および TYPETERM 定義でブリッジ機能をモデル化するために、CICS Web サポートを使用して実行する 3270 トランザクションの PROFILE 定義を、次のようにして更新します。
   a. CEDA を使用して TRANSACTION 定義の PROFILE パラメーターを表示することにより、トランザクションで使用する PROFILE を識別します。

b. プロファイルが CICS 提供のプロファイルであれば、それをユーザー独自のグループにコピーして名前変更します。
c. 新規 PROFILE を変更し、新規ブリッジ機能の名前を FACILITYLIKE パラメーターに入力します。
d. TRANSACTION 定義で新規の PROFILE 定義を使用するように変更します。

検出可能フィールドの使用

CICS は、3270 データ・ストリームから HTML ページを生成すると、テキスト入力フィールドの前にチェック・ボックスを付けて検出可能フィールドをシミュレートします。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

始める前に

検出可能フィールドを使用するには、トランザクションに関連付けられたブリッジ機能を構成します。 208 ページの『検出可能フィールドの使用可能化』を参照してください。

このタスクについて

検出可能フィールドには、以下の特性があります。

- フィールド属性バイトがフィールドを検出可能または高輝度表示として識別する。
- 3270 フィールドの先頭文字に有効な指定文字が含まれている。この指定文字は、アンパーサンド (&)、不等号右括弧 (>)、疑問符 (?)、ブランク、またはヌルなどです。

検出可能フィールドについて詳しくは、次の資料を参照してください。「*CICS アプリケーション・プログラミング・ガイド*」のField selection features。

チェック・ボックスおよびテキスト入力フィールドが Web クライアントに表示される場合は、以下の文字に注意してください。

- 3270 フィールドの指定文字は表示されません。したがって、 Web クライアントのフィールド長は、3270 データ・ストリームのそれよりも 1 文字短くなります。
- 指定文字が不等号右括弧 (>) である場合、チェック・ボックスにはチェック記号 (✔) が入ります。それ以外の場合は、チェック・ボックスは空になります。

Web クライアントで検出可能フィールドを使用するには、次のようにします。

手順

- チェック・ボックスにチェックマークを付けて、3270 データ・ストリームでの変更データ・タグ (MDT) ビットの設定をシミュレートします。チェック・ボックスのチェックマークを外して、変更データ・タグをオフに設定します。 HTML ページのテキスト・フィールドにデータを入力しても、変更データ・タグは変更されません。

- データを CICS アプリケーションに送信するには、チェック・ボックスにチェックマークを付け、「**DFH_PEN**」ボタンを選択します。
  - 1 つのアテンション・フィールドだけにチェックマークを付けると、CICS アプリケーションはそのフィールドの内容を受け取ります。EIBAID フィールドは DFHPEN に設定されます。
  - 複数のアテンション・フィールドにチェックマークを付けると、CICS アプリケーションは、3270 画面の行 1、列 1 に最も近いフィールドの内容を受け取ります。EIBAID フィールドは DFHPEN に設定されます。
  - アテンション・フィールドにチェックマークを付けないと、CICS は、すべてのフィールドの内容を受け取ります。EIBAID フィールドは DFHENTER に設定されます。

DFHWBIMG を使用したグラフィックスの表示

CICS は、Web アプリケーションで使用できるグラフィックスを備えています。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

このタスクについて

CICS は、以下のグラフィックスを提供します。

CICS.GIF
CICS ロゴ

MASTHEAD.GIF
'CICS Web Interface' というテキストが入った CICS ロゴ

BACKGROUND1.GIF
'CICS' の文字が入ったバックグラウンド

BACKGROUND2.GIF
'CWI' の文字が入ったバックグラウンド

TEXTURE1.JPEG
テクスチャー・バックグラウンド

TEXTURE2.JPEG
テクスチャー・バックグラウンド

TEXTURE3.JPEG
テクスチャー・バックグラウンド

TEXTURE4.JPEG
テクスチャー・バックグラウンド

TEXTURE5.JPEG
テクスチャー・バックグラウンド

TEXTURE6.JPEG
テクスチャー・バックグラウンド

グラフィックスを Web ブラウザーに表示するには、パスが以下の形式になっている (大文字への変換後に) URL を入力します。

```
/DFHWBIMG/filename
```

ここで、filename は、リストされるグラフィックスの名前の 1 つです。例えば、次のとおりです。

```
 /DFHWBIMG/Texture1.jpeg
```

いずれかのグラフィックスを出力データに取り込むには、パスを適切な HTML タグに組み込みます。例えば、以下のようなタグを使ってテクスチャー・バックグラウンドを組み込むことができます。

```
<body background="/DFHWBIMG/background1.gif" ... >
```

CICS は、特殊なケースとして、パスが /DFHWBIMG で始まっている HTTP 要求を処理します。ただし、アナライザーは呼び出されず、DFHWBIMG がコンバーター・プログラムとして実行されます。

CICS は、これらのグラフィックスのいくつかを、CICS 提供のトランザクションで使用するテンプレートで使用します。

CICS 提供のグラフィックスは DFHWBIMG の一部としてハードコーディングされており、別個のファイルとしては使用できません。 DFHWBIMG は、指定されているもの以外のグラフィックスの表示をサポートしていません。

第 14 章 BMS 定義からの HTML テンプレートの作成

ソース・コードのない既存の BMS マップ・セットから HTML テンプレートを作成する場合、対応するロード・モジュールからソースを再構成できる可能性があります。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

このタスクについて

BMS マクロ生成ユーティリティー・プログラム (DFHBMSUP) を使用してください。このプログラムについては、「*CICS Operations and Utilities Guide*」の BMS macro generation utility を参照してください。

CICS は、BMS マップ・セットから作成された HTML テンプレートをインストールするためのカタログ・プロシージャー DFHMAPT を備えています。詳細については、「*CICS アプリケーション・プログラミング・ガイド*」の Installing map sets and partition sets を参照してください。

BMS 生成テンプレート

BMS マップから生成されるテンプレートには、定数と入力フィールド、ボタン、非表示変数、JavaScript 関数、および JavaScript 例外ハンドラーが含まれます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

テンプレートには、以下の項目が含まれます。

- マップから入手した定数と入力フィールド
- 以下のものを表すボタン
  - ENTER および CLEAR キー
  - PA1、PA2、および PA3 キー
  - プログラム・ファンクション・キー PF1 ～ PF24
  - HTML リセット
- *DFH_NEXTTRANSID.1* から *DFH_NEXTTRANSID.5* までの最大 5 個の隠れ変数。これらの変数の値は、マップ内の最初の 5 つのフィールドの名前です。193 ページの『第 13 章 CICS Web サポートおよび 3270 表示アプリケーション』で、これらの変数の使用法について説明しています。
- 隠れ変数 *DFH_CURSOR*。この変数の値は、マップ内でカーソルが置かれているフィールドの名前です。カーソルが名前のないフィールドにある場合、*DFH_CURSOR* はゼロです。
- JavaScript 関数 `dfhsetcursor()`。 *DFH_CURSOR* にフィールドの名前が入っていると、関数はそのフィールドにカーソル位置をセットします。
- onFocus 例外用の JavaScript 例外ハンドラー。この機能は、`dfhsetcursor` を呼び出し、カーソル移動を追跡します。

カスタマイズ済み HTML テンプレートの生成

BMS マップから生成された HTML テンプレートを、3 とおりの方法でカスタマイズできます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

このタスクについて

次の 3 とおりの方法でテンプレートをカスタマイズできます。

- 独自の DFHMSX マクロをコーディングすれば、BMS マップから HTML テンプレートを生成する方法を変更することができます。
- BMS マップ定義内で DFHWBOUT マクロを使用すれば、HTML テキストを生成済みマップに追加することができます。
- 生成済み HTML を手動で編集することができます。
  - プログラムが MAP SEND コマンドを発行すると行われる属性の動的変更をオーバーライドする場合
  - Web 3270 環境外で HTML テンプレートを使用する場合

  どちらの場合も、テンプレート生成プロセスによって追加される *Frrcccllll* 変数を変更する必要があります。

すべての **SEND MAP** コマンドが ERASE オプションを使用する場合を除き、CICS 生成 HTML テンプレートは編集しないでください。ERASE を使用せずに **SEND MAP** コマンドを実行すると、HTML がマージされます。ランタイム・ロジックでは、 CICS テンプレート・ジェネレーターで生成された HTML の存在が想定されています。特に、`<tr>` タグの変更は避けてください。

カスタマイズ・テンプレートの例については、 228 ページの『カスタマイズの例』を参照してください。

CICS は、BMS を使用する CETR CICS 提供トランザクション用の HTML テンプレートを提供します。これらのテンプレートは、EBCDIC Latin 文字セット (コード・ページ 037) を使用します。 CICS システムで別のコード・ページを使用する場合は、これらのテンプレートの独自バージョンを生成する必要があります。 CICS 生成のヘッディング・セクションで使用される以下の文字は、037 とは別のコード・ページに別の表記法を持っています。

```
! [ ] { }
```

DFHMDX マクロで **CODEPAGE** パラメーターを使用してコード・ページを指定してください。

DFHMSX マクロによるカスタマイズ

独自の DFHMSX マクロをコーディングすれば、BMS マップから HTML テンプレートを生成する方法を変更することができます。

このタスクについて

以下の項目を指定できます。

- ボタンで表される 3270 キー
- 各ボタンに表示するテキストまたはイメージ
- HTML ページの表題
- HTML ページのトップに表示するマストヘッド・グラフィック
- 図形ファイルまたはカラーとしてのページ・バックグラウンド
- 通常テキスト、未アクセス・リンク、アクセス・リンク、およびアクティブ・リンクのカラー
- ページに HTML リセット・ボタンを含めるかどうか、およびそこにテキストを表示するかどうか
- BMS マップに使用するカラー間のマッピングおよび HTML テンプレート内の対応するテキストに使用するカラー
- HTML ページから抑止する BMS フィールド
- JavaScript `onLoad()` および `onUnload()` 例外ハンドラー
- テンプレートのテキストをプロポーショナル・フォントで表示するか、非プロポーショナル・フォントで表示するか
- テンプレートを生成するときに使用するコード・ページ、および特殊文字 `[]{}` と ! に使用するコード・ポイント
- HTML ページで保護フィールドを右寄せするかどうか

注:

1. BMS フィールドの ATTRB=BRT オプションは、名前のない無保護 (入力) フィールドには影響を与えません。
2. 論理マップで指定されたフィールドの属性バイトの 3270 属性ビット DFHBMEOF は、フィールドが空 (例えば DEL キーのため) の場合や、フィールドが前の SEND コマンドの際に既に空 (ヌルまたはスペース) になっていて、そのフィールドの変更データ・タグ (MDT) がオフになっている場合は設定されません。

ユーザー独自バージョンの DFHMSX マクロをコーディングする場合は、オプションを適用するマップとして、以下を指定できます。

- すべてのマップ・セット内のすべてのマップ
- 特定のマップ・セット内のすべてのマップ
- 個々のマップ

HTML テンプレートのインストール

アプリケーション・プログラムによっては、カスタマイズされた HTML ページを必要とする場合があります。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

このタスクについて

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

カスタマイズした HTML ページを作成するには、以下のようにします。

手順

1. CICS アプリケーション・プログラムとそのアプリケーションによる BMS の使用を調べ、カスタマイズが必要であるかどうかを確認します。
2. カスタマイズした HTML ページを必要とするアプリケーションの場合は、カスタマイズ・マクロ定義を作成し、アセンブリ・プログラムの SYSLIB DD ステートメントで指定されたマクロ・ライブラリーの連結の中のライブラリーにそれを保管します。適切な DFHWBOUT マクロ呼び出しを書き、それをマップ定義の適切な場所に書き込みます。
3. DFHMSD マクロで TYPE=TEMPLATE を指定するか、あるいはアセンブリ・プログラムに渡すパラメーターで SYSPARM=TEMPLATE を指定して、既存のマップ定義をアセンブルします。 DFHMSD マクロのラベルが、処理中のマップ・セット内のマップごとに生成される HTML テンプレートの名前になります。HTML テンプレートの名前は、DFHMSD マクロからのラベル、およびそれに続く A から Z および 0 から 9 の文字を使用して生成される 1 または 2 文字の接尾部で構成されています。マップ・セットに含まれるマップが 36 より多いときに 2 文字の接尾部が使用されます。その場合、マップ・セット名は 6 文字以下でなければなりません。BMS SEND または BMS RECEIVE がプログラムから発行されたときに、ブリッジ出口で HTML テンプレートと BMS マップとを突き合わせる場合は、 HTML テンプレートのメンバーが、SEND および RECEIVE ステートメントで使用するマップ・セットの名前と一致しなければなりません。カスタマイズ・マクロを使用する場合は、カスタマイズ・マクロの名前を TYPE に追加します。アセンブリ・プログラムは、マップ・セット内の各マップごとに 1 つずつテンプレートを作成する IEBUPDTE ソース・ステートメントを生成します。
4. IEBUPDTE を使用して、テンプレートをテンプレート・ライブラリーに保管します。テンプレート・ライブラリーのレコード形式が固定ブロックでない場合は、テンプレートを別の区分データ・セットにいったん保管した後、例えば ISPF COPY を使用して、テンプレート・ライブラリーのレコード形式にテンプレートを変換する必要があります。
5. DFHHTML DD 名に指定したもの以外の区分データ・セットにテンプレートを入れたい場合は、そのテンプレートに対する DOCTEMPLATE 定義を定義して、代替 DD 名を指定する必要があります。 CICS JCL でも代替 DD 名を指定する必要があります。

   テンプレートが含まれる区分データ・セットを特定の DD 名に割り振り、そこからテンプレートをインストールできるようにするには、ADYN サンプル・トランザクションを使用します。まず、ADYN とそれに関連するプログラムが含まれる DFH$UTIL グループをインストールし、次に ADYN を実行します。

   ```
   ADYN
   ALLOC DDNAME(ddname) DATASET('template-pds') STATUS(SHR)
   ```

ここで、*ddname* は DOCTEMPLATE 定義で指定した DD 名、 *template-pds* は、インストールするテンプレートが入っている区分データ・セットの名前です。 ADYN のインストールおよび使用について詳しくは、「*CICS Customization Guide*」を参照してください。

大きい HTML テンプレートの処理

3270 ブリッジを使って実行されるトランザクションが使用するテンプレートのサイズに、制限は適用されません。ただし、ストレージが 32KB を超えるテンプレートは、それより小さいテンプレートとは別の方法で処理されます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

このタスクについて

32KB より大きいテンプレートを処理する場合は、HTTP 要求の中で DFHWBTTB のプログラム名にマップするパスを指定します。 196 ページの『3270 表示アプリケーションの URL パス構成要素』を参照してください。

必要なストレージが 32 KB 未満のテンプレートでも、シンボル置換のためにデータ量が大幅に増加した場合は、32 KB より大きくなることがあります。

テンプレートを生成すると、DFHWBTLG は、各テンプレートを DFHHTML データ・セットから読み込むために必要なストレージ量を示すメッセージを出します。プログラム名として DFHWBTTA を使用するか DFHWBTTB を使用するかは、これらのメッセージを使用して判断します。

カスタマイズ・マクロ定義の作成

CICS 提供のアセンブラー・マクロによって呼び出される完全なアセンブリー言語マクロ定義を用意する必要があります。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

このタスクについて

アセンブリー言語マクロ定義の規則に従って、カスタマイズ・マクロの定義を作成してください。定義内のマクロの起動も、アセンブリ言語マクロ・ステートメントの規則に従っている必要があります。

カスタマイズ・マクロ定義には、次の要素が含まれています。

1. 定義を開始させる MACRO ステートメント。
2. マクロの名前。
3. 任意の数の DFHMDX マクロの起動。

   DFHMDX の構文の説明は 221 ページの『DFHMDX マクロ』にあり、その使用法の説明は 228 ページの『カスタマイズの例』にあります。DFHMDX は、DFHMSX の中から起動されます。

4. 定義を終了させる MEND ステートメント。

空白の処理

マクロ定義をカスタマイズするときは、空白に関する HTML 仕様を考慮してください。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

このタスクについて

3270 端末の場合は、ブランク (EBCDIC X'40') とヌル (EBCDIC X'00') を使用して、画面データの位置を形式設定できます。このようなデータ・ストリームを HTML に変換し、それをクライアントが解釈すると、3270 端末の場合とは異なる出力が生成されます。

ブランクのストリングは、開始タグの直後にあると、クライアントによって無視され、後続の一連のブランクは 1 つのブランクとして解釈されます。すべてのブランクをレンダリングするには、<pre> タグおよび </pre> タグを使用します。

ヌル文字の処理は指定されていないので、クライアントの処理に整合性はありません。場合によって、表示されたり、表示されなかったりします。

BMS 出力と非 BMS 出力の結合

トランザクションで一連の BMS コマンドと非 BMS コマンドを出して、3270 表示画面の内容を作成することができます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

以下のように、すべてのコマンドからの出力が結合され、Web ブラウザーで表示される HTML ページを構成します。

1. BMS または非 BMS SEND コマンドを出すと、(ヘッディング・セクション、画面イメージ・セクション、およびフッティング・セクションが含まれている) HTML ページが生成されますが、Web クライアントへは送信されません。
2. トランザクションから RECEIVE コマンドを出すか、トランザクションが終了すると、
   - ヘッディング・セクションが、それまでに生成されたものから選択されます。
   - 新しい画面イメージ・セクションが、それまでに生成されたものすべてをマージして作成されます。
   - フッティング・セクションが、それまでに生成されたものから選択されます。
3. こうして作成された HTML ページが Web クライアントに送信されます。

ヘッディング・セクションの選択方法

ヘッディング・セクションは、以下のように、各 HTML ページの開始位置と作成順序に基づいて選択されます。

1. 画面の先頭行に最も近い開始位置を持つページが選択されます。
2. 複数のページが選択プロセスに残っている場合は、それらのページの開始位置が再度比較されます。このとき、画面の先頭列に最も近い開始位置を持つページが選択されます。
3. 最後に、複数のページが残っている場合は、最初に生成されたページが選択されます。

Web クライアントに送信する HTML ページでは、残りの選択ページのヘッディング・セクションが使用されます。

注:

- BMS マップから生成された HTML ページの開始位置は、マップの左上隅の行と列です。
- 非 BMS コマンドから生成された HTML ページの開始位置は、画面の左上隅 (行 1、列 1) です。

フッティング・セクションの選択方法

フッティング・セクションは、以下のように、各 HTML ページの終了位置と作成順序に基づいて選択されます。

1. 画面の最終行に最も近い終了位置を持つページが選択されます。
2. 複数のページが選択プロセスに残っている場合は、それらのページの終了位置が再度比較されます。このとき、画面の最終列に最も近い終了位置を持つページ選択されます。
3. 最後に、複数のページが残っている場合は、最後に生成されたページが選択されます。

Web クライアントに送信する HTML ページでは、残りの選択ページのフッティング・セクションが使用されます。

注:

- BMS マップから生成された HTML ページの終了位置は、マップの右下隅の行と列です。
- 非 BMS コマンドから生成された HTML ページの終了位置は、画面の右下隅です。

画面イメージ・セクションのマージ方法

一連の BMS および非 BMS SEND コマンドを実行して作成された画面イメージ・セクションをマージすると、新規の画面イメージ・セクションが作成されます。この画面イメージ・セクションには、可能な限り、それを作成するために使用されたすべての画面イメージ・セクションのすべてのフィールドが取り込まれています。

しかし、複数の構成要素画面イメージのフィールドが全部または部分的にオーバーラップしている場合は、マージを行えません。したがって、重複フィールドのいくつかは、以下のように、変更されるか完全に抑止される可能性があります。

- いくつかのフィールドがオーバーラップすると、それ以前に使用した BMS または非 BMS SEND コマンドに関連するフィールドが、それ以降に使用するコマンドのために変更または抑止されます。

- 入力フィールドの一部または全部がオーバーラップしている場合は、その入力フィールド全体が廃棄され、最終の HTML には表示されなくなります。
- 入力フィールドが何らかの通常テキストと部分的にオーバーラップしている場合は、入力フィールドの先頭までのすべての可視テキストが最終 HTML に表示され、残りのデータは廃棄されます。この処置は、3270 装置上で入力フィールドの後にまだ可視テキストがあるかどうかに関係なく行われます。
- テーブル・セルに水平罫線タグ (<hr>) が含まれている場合に、セルの内容をオーバーラップすると、予期しない結果を招きます。

これらの規則の要約が表 11 に示されています。

表 11. マージした画面イメージ・セクションでのフィールド・オーバーラップ

先の SEND からのフィールド	**後の SEND からのフィールド**	**結果**
入力 (無保護)	入力 (無保護) またはテキスト (保護)	以前のフィールドは全部抑止されます
テキスト (保護)	入力 (無保護)	3270 画面の文字位置に基づいて、 • 入力フィールドの左方の保護文字が保存されます。 • 入力フィールドによって上書きされた保護文字が抑止されます。 • 入力フィールドの右方の保護文字が抑止されます。
テキスト (保護)	テキスト (保護)	3270 画面の文字位置に基づいて、後で送信した文字が以前に送信した文字を上書きします。

BMS マップから作成した HTML テンプレートをアプリケーション・プログラムで使用する前に編集することができます。しかし、画面イメージ・セクションをマージするアルゴリズムでは、画面イメージ・セクションの HTML が特定の構造を持っていなければなりません。したがって、テンプレートの画面イメージ・セクションを編集して、そのセクションを他のセクションとマージする場合は、以下のガイドラインに従ってください。

- 各 HTML ページには、それぞれ DFHROW および DFHCOL ストリングを持つ 2 つのコメントが含まれています。これらのストリングの後に続く値は、3270 画面上の各フィールドの位置計算に使用されるので、マージ・プロセス中は重要です。これらのコメントを変更または削除した場合、画面イメージ・セクションはマージされませんが、最終 HTML ページでは、上から下へ順に追加される形で表示されます。
- テーブル・セルの終了タグ (</td>) とテーブル行 (</tr>) はオプションです。
- テーブル・セルには、通常テキストの一部 (追加属性タグの有無は問わない) が含まれているか、もしくは入力フィールドが含まれていなければなりません。ま

た、テキストと入力フィールドが、両者間に追加タグの挿入がなく互いに連続している場合は、同一テーブル・セル内にテキストと入力フィールドの混合を含めることもできます。

- 空のテーブル・セルには、開始タグと終了タグの間にヌル値 (X'00') またはスペースを含めることができません。つまり、空のセルは、<td></td> としてコーディングする必要があります。
- テキストのセクションまたは入力フィールドを以下のような 1 つ以上のタグ・ペアで区切ることができます。

  強調 ...

  強調 ...

  フォント
   ...

  下線 <u> ... </u>

  明滅 <blink> ... </blink>

  どのタグにも、対応する終了タグが必要です。また、開始タグと終了タグが正しくネストされていることも確認する必要があります。例えば、<u> ... </u> は適切にネストされますが、<u> ... </u> は適切にネストされません。
- 他のタグをテーブル・セルに挿入する場合は、テキストまたは入力フィールドの前か後でなければなりません。ただし、同じテーブル・セル内で前後両方に置くことはできません。
- HTML コメントをテーブル・セルに入れることができ、通常テキストまたは入力フィールドの前後どちらか、あるいは両側に置くことができます。
- セルにコメントを含める場合は、通常テキストまたは入力フィールドも含めなければなりません。

DFHMDX マクロ

DFHMDX マクロは、DFHMSX 内から呼び出されます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

その構文を 222 ページの図 9 に示します。

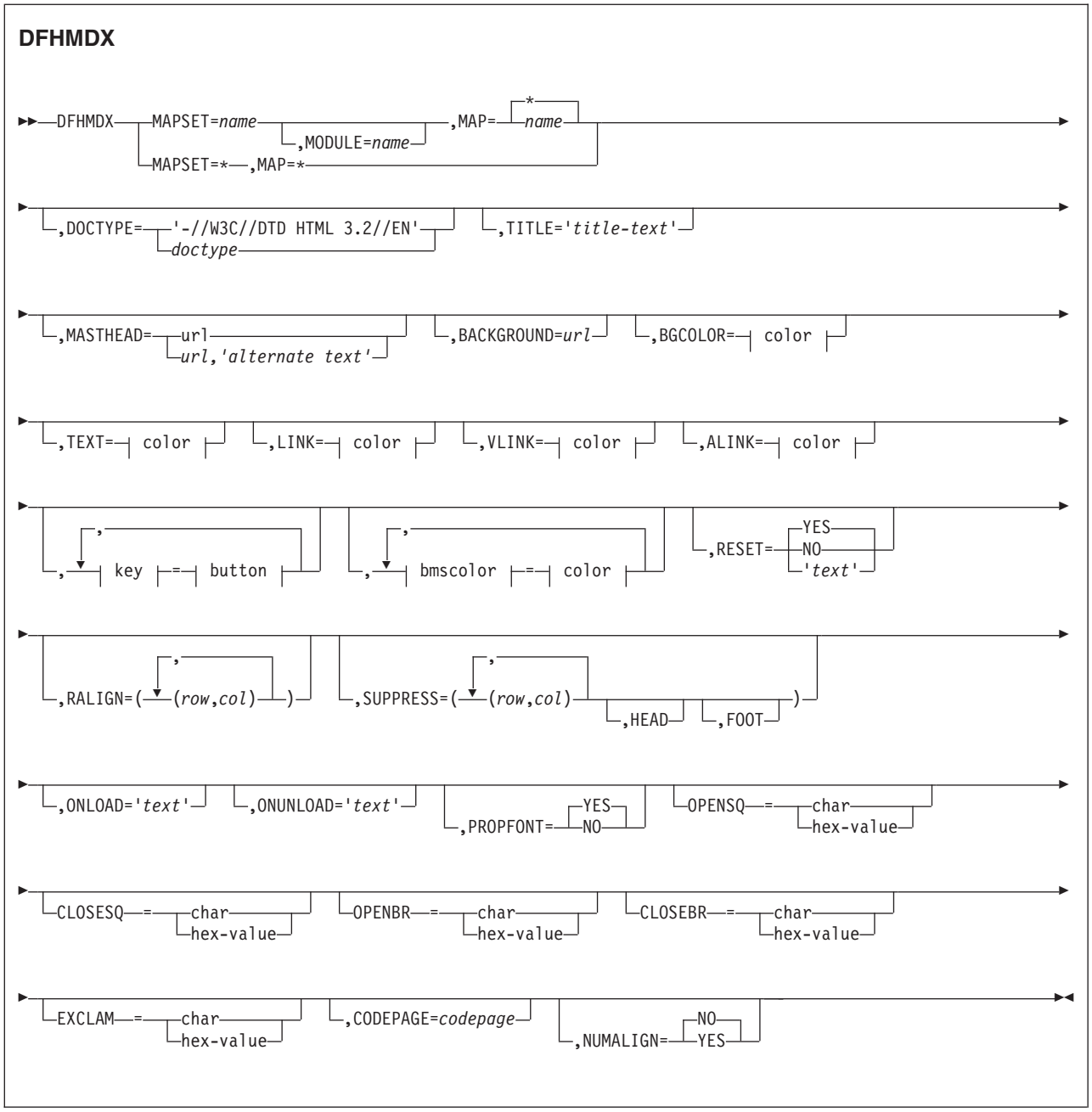


図 9. DFHMDX の構文

このマクロのキーワード・パラメーターの指定順序は任意です。

MAPSET

他のオプションが参照するマップが含まれるマップ・セットの名前を指定します。アスタリスクを指定すると、これらのオプションは、後続のすべてのマップ・セットのデフォルト値になります。

MODULE

マップ・セットのリンク・エディット先のロード・モジュールの名前を指定します。このパラメーターは、MAPSET=* を指定していない場合に限り使

用できます。指定する名前 (7 文字のみです) を使用し、単一文字の接尾部を追加することで、テンプレートの名前を作成します。デフォルト値は、マップ・セットの名前です。

MAP オプションが参照する MAPSET に指定されたマップ・セット内のマップの名前を指定します。アスタリスクを指定すると、これらのオプションは、後続のすべてのマップのデフォルト値になります。

DOCTYPE
HTML テンプレート内に表示されるようにしたい <!doctype> タグの DTD 共通 ID 部分を指定します。デフォルト値は `-//W3C//DTD HTML 3.2//EN` であり、これは HTML 3.2 を指定します。特定の HTML タグのカラー・サポートには、レベル 3.2 が必要です。

TITLE
HTML 表題として、また最初の `<h1>` タグの内容として使用される表題を指定します。

MASTHEAD
ページの冒頭の、最初の `<h1>` タグの前に表示されるマストヘッド・グラフィックの URL を指定します。*代替テキスト* を提供すれば、クライアントは、指定されたグラフィックをロードできないとき、そのテキストを使用します。

BACKGROUND
ページ・バックグラウンドの図形ファイルの URL を指定します。

BGCOLOR
ページのバックグラウンドのカラーを指定します。

TEXT 通常テキストのカラーを指定します。

LINK ページ上の接続されていないハイパーテキスト・リンクのカラーを指定します。

VLINK
ページ上の接続されているハイパーテキスト・リンクのカラーを指定します。

ALINK
ページ上の活動化されているハイパーテキスト・リンクのカラーを指定します。

PF1-PF24
対応する 3270 プログラム・ファンクション・キーのシミュレート・ボタンに割り当てられる名前またはイメージを指定します。

PA1-PA3
対応する 3270 プログラム・アテンション・キーのシミュレート・ボタンに割り当てられる名前またはイメージを指定します。

CLEAR
3270 CLEAR キーのシミュレート・ボタンに割り当てられる名前またはイメージを指定します。

ENTER
3270 Enter キーのシミュレート・ボタンに割り当てられる名前またはイメージを指定します。

PEN ペン選択用のシミュレート・ボタンに割り当てられる名前またはイメージを指定します。

BLUE BMS マップでブルーが指定されている HTML ページ内の場所に表示されるカラーを指定します。デフォルト値は #0000FF です。

制約事項: DFHMDX は、名前なしフィールドのカラーのみオーバーライドします。名前付きフィールドは変更されずそのままです。

GREEN
BMS マップでグリーンが指定されている HTML ページ内の場所に表示されるカラーを指定します。デフォルト値は #008000 です。

制約事項: DFHMDX は、名前なしフィールドのカラーのみオーバーライドします。名前付きフィールドは変更されずそのままです。

NEUTRAL
BMS マップで灰色が指定されている HTML ページ内の場所に表示されるカラーを指定します。デフォルト値は #000000 です。

制約事項: DFHMDX は、名前なしフィールドのカラーのみオーバーライドします。名前付きフィールドは変更されずそのままです。

PINK BMS マップでピンクが指定されている HTML ページ内の場所に表示されるカラーを指定します。デフォルト値は #FF00FF です。

制約事項: DFHMDX は、名前なしフィールドのカラーのみオーバーライドします。名前付きフィールドは変更されずそのままです。

RED BMS マップで赤が指定されている HTML ページ内の場所に表示されるカラーを指定します。デフォルト値は #FF0000 です。

制約事項: DFHMDX は、名前なしフィールドのカラーのみオーバーライドします。名前付きフィールドは変更されずそのままです。

TURQUOISE
BMS マップで青緑色が指定されている HTML ページ内の場所に表示されるカラーを指定します。デフォルト値は #00FFFF です。

制約事項: DFHMDX は、名前なしフィールドのカラーのみオーバーライドします。名前付きフィールドは変更されずそのままです。

YELLOW
BMS マップで黄色が指定されている HTML ページ内の場所に表示されるカラーを指定します。デフォルト値は #FFFF00 です。

制約事項: DFHMDX は、名前なしフィールドのカラーのみオーバーライドします。名前付きフィールドは変更されずそのままです。

RESET
HTML リセット機能がサポートされるかどうかを指定します。 YES を指定して、デフォルトの Reset という説明を持つデフォルトのリセット・ボ

タンを入手します。リセット・ボタンなしの場合は、NO を指定します。ユーザー独自の説明付きリセット・ボタンには、独自のテキストを指定してください。

RALIGN

HTML ページでデータを右寄せする BMS マップ・フィールドを指定します。指定する値 *rr* および *cc* は、フィールドを右寄せするための DFHMDF マクロに指定された POS=(*rr*,*cc*) に対応していなければなりません。それぞれの対を括弧で囲み、対のリスト全体も括弧で囲む必要があります。特定の列で終わるすべての適格フィールドを右寄せしたい場合は、末尾列番号を指定し、行指定のためのアスタリスクを入れます。フィールドの末尾列番号を計算するには、DFHMDF マクロに定義されたその LENGTH にこのフィールドの先頭列番号を追加します。フィールドが右寄せされるのは、それらのフィールドが保護されていて、名前が付けられておらず、DFHMDF マクロの INITIAL、XINIT、または GINIT 値で初期設定されている場合のみです。RALIGN パラメーターを MAP=* または MAPSET=* と一緒に指定した場合、RALIGN パラメーターは無視されます。

マクロ定義の文字ストリングに関するアセンブリ言語プログラムの限界値 256 文字を超えるリストを指定したい場合は、同じ MAPSET 値および MAP 値で追加の DFHMDX マクロをコーディングし、RALIGN パラメーターの値を増やします。

SUPPRESS

HTML ページに現れない BMS マップ・フィールドを指定します。抑制されるフィールドの開始位置を表す行/列の対の任意の数を指定します。指定する値 *rr* と *cc* は、抑制されるフィールド用の DFHMDF マクロに指定された POS=(*rr*,*cc*) に対応していなければなりません。それぞれの対を括弧で囲み、対のリスト全体も括弧で囲む必要があります。ある行の中のフィールドすべてを抑制したい場合には、その行番号を指定し、列指定にはアスタリスクを書き込んでください。 SUPPRESS パラメーターを MAP=* または MAPSET=* と一緒に指定した場合は、SUPPRESS パラメーターは無視されます。

テンプレートのヘッディング・セクションを非表示にするには、キーワード HEAD を使用します。テンプレート内のフッティング・セクションを非表示にするには、キーワード FOOT を使用します。

マクロ定義の文字ストリングに関するアセンブリ言語プログラムの限界値 256 文字を超えるリストを指定したい場合は、同じ MAPSET 値および MAP 値で追加の DFHMDX マクロをコーディングし、SUPPRESS パラメーターの値を増やします。

ONLOAD

HTML ページの標準 `onLoad` 例外ハンドラーに代わるものとして使用される JavaScript テキストを指定します。通常のアセンブリ言語プログラム規則に従い、このテキストには、二重引用符 (") を含めるのではなく、単一引用符 (') を重ねる ('') 必要があります。このパラメーターを使用する場合は、標準の `onLoad` 例外ハンドラーが提供する DFH_CURSOR によって示されるフィールドへのカーソルの設定を抑制することになります。`dfhsetcursor` 機能を使用して、カーソル位置を設定することができます。

ONUNLOAD
HTML ページの onUnload 例外ハンドラーとして使用する JavaScript テキストを指定します。通常のアセンブリ言語プログラム規則に従って、このテキストには、二重引用符 (") ではなく、単一引用符 (') を重ねて指定する (' ') 必要があります。

PROPFONT
フォントを指定します。YES を指定した場合、テンプレートは、テキストをプロポーショナル・フォントで表示することを指定し、連続したスペースは単一のスペースに切り詰められます。 NO を指定した場合、テンプレートは、テキストを固定ピッチのフォントで表示することを指定し、連続したスペースはそのまま表示されます。

OPENSQ
左大括弧を表示するために使用する 16 進数の値または文字。デフォルトは X'BA' (コード・ページ 37) です。

CLOSESQ
右大括弧を表示するために使用する 16 進数の値または文字。デフォルトは X'BB' (コード・ページ 37) です。

OPENBR
左中括弧を表示するために使用する 16 進数の値または文字。デフォルトは X'C0' (コード・ページ 37) です。

CLOSEBR
右中括弧を表示するために使用する 16 進数の値または文字。デフォルトは X'D0' (コード・ページ 37) です。

EXCLAM
感嘆符を表示するために使用する 16 進数の値または文字。デフォルトは X'5A' (コード・ページ 37) です。

CODEPAGE
テンプレート生成プロセスで生成した任意のテキストをエンコードするときの IBM コード・ページ番号を指定します。このコード・ページは、テンプレートを CICS で使用するときに使用する、**EXEC CICS** DOCUMENT コマンドの HOSTCODEPAGE オプションで指定したコード・ページ、またはアナライザー・プログラムが選択した DFHCNV マクロの SRVERCP オプションで指定したコード・ページのいずれかと一致しなければなりません。

CICS での標準形のホスト・コード・ページ名は、必要に応じて 3 から 5 桁の 10 進数で記述されたコード・ページ番号 (より一般的には CCSID) で構成され、末尾に最大 8 文字のスペースが埋め込まれます。3 桁未満であるコード・ページ 37 の場合、標準形は 037 です。CICS は、標準形でない場合でも、最大 8 桁 (末尾にスペースが埋め込まれた) である 1 から 65,535 の範囲の任意の 10 進数をコード・ページ名として受け入れます。

シンボルやシンボル・リスト処理に使用される区切り文字は EBCDIC で記述されていると想定されるため、シンボル処理が必要な場合は、CODEPAGE パラメーターで EBCDIC ベースのコード・ページを指定する必要があります。

デフォルトのコード・ページは 037 です。

NUMALIGN

DFHMDF マクロの中で数値として明示的に定義されているフィールドを、HTML テンプレート内のテーブル・セルの中でどのように位置合わせするかを指定します。

NO これらのテーブル・セルでは、数値フィールドは右に位置合わせしないことを指定します。これはデフォルトです。

YES これらのテーブル・セルでは、数値フィールドは右に位置合わせすることを指定します。

- 保護されたフィールドでは、生成される HTML テキストはセルで右に位置合わせされます。テキストに末尾ブランクがある場合、それらは保存されないこともあります。それらを単一のブランクで置き換えるクライアントもあります。

  注: RALIGN パラメーターは末尾ブランクを保存します。**NUMALIGN** パラメーターは保存しません。 1 つのフィールドに両方のパラメーターが適用される場合 (つまり、数値フィールドが **RALIGN** パラメーターで識別されていて、しかも NUMALIGN=YES が指定されている場合) は、末尾ブランクは保存されません。
- 無保護フィールドの場合は、HTML テキスト入力要素 (ただし、要素のテキストではない) は、セルで右に位置合わせされます。

color は明示的に *#rrggbb* として指定することができます。ここで、*rr*、*gg*、および *bb* は要求する色の赤、緑、青の輝度を示す 2 桁の 16 進数です。または、次のようなカラー名のいずれかにすることもできます。AQUA、BLACK、BLUE、 FUCHSIA、GRAY、GREEN、LIME、 MAROON、NAVY、OLIVE、PURPLE、 RED、SILVER、TEAL、WHITE、YELLOW。

key は、PF1 ～ PF24、 PA1 ～ PA3、CLEAR、ENTER、および PEN のいずれかにすることができます。

button は、(IMAGE,*url*) (ここで、*url* には、ボタンに使用するグラフィック・イメージの URL を指定します)、または '*text*' (ここで、*text* はボタンに挿入するテキストです)、または NO (ボタンを表示しない場合) にすることができます。

bmscolor は、BLUE、GREEN、NEUTRAL、PINK、RED、TURQUOISE、および YELLOW のいずれかにすることができます。

DFHWBOUT マクロを使用したテンプレートのカスタマイズ

DFHWBOUT マクロを使用して、BMS マップから生成される HTML ページにテキストを追加します。テキストは、HTML ページの一部としてのみ表示されます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

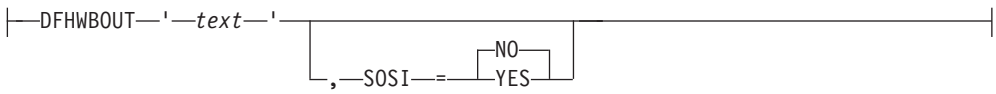
このタスクについて

マップ中の DFHMDF の最初の出現の前にこのマクロを使用すると、テキストは HTML ページの <head> セクションの中に置かれます。このマクロをマップ中の他の箇所で使用すると、テキストは直前の DFHMDF によって生成されたテキストのすぐ後に挿入されます。

アプリケーション・プログラムで複数の BMS マップを使用して画面表示を作成する場合は、DFHWBOUT マクロを使用しないでください。

DFHWBOUT

DFHWBOUT マクロ



このマクロのパラメーターは次のとおりです。

text HTML ページに挿入されるテキスト。

SOSI テキストが、シフトアウト (X'0E') およびシフトイン (X'0F') で区切られた DBCS 文字を含むかどうか。デフォルト値は SOSI=NO です。

DFHWBOUT マクロを使用する場合は、挿入する HTML テキストが、BMS マップ・フィールドから生成されるページ・レイアウトに影響を与えることがあるという点に注意してください。正しいページ・レイアウトにするために、挿入したテキストを調整しなければならない場合があります。

カスタマイズの例

この例に従えば、BMS マップ定義用のテンプレートをカスタマイズするのに役立ちます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

以下の例は、カスタマイズ・マクロ定義を示しています。DFHMDX の最初の起動では、マップ・セット名とマップ名に * を指定することにより、DFHMDX の後続の起動に適用される値のデフォルトが設定されます。その後の起動では、マップ・セット中の特定マップのパラメーターがオーバーライドされるか、またはそのパラメーターに追加されます。継続文字は 72 桁目にあり、継続テキストは 16 桁目から再開されます。

```
MACRO
DFHMSX
DFHMDX MAPSET=*,MAP=*,                                          *
      F1='Help',F3='Exit',F4='Save',F9='Messages'
DFHMDX MAPSET=DFHWB0,MAP=*,                                     *
      TITLE='CICS Web Interface',                               *
      F3='Messages'
```

```
DFHMDX MAPSET=DFHWB0,MAP=DFHWB02,                                *
      TITLE='CICS Web Interface Enable',                         *
      F3='Save'
MEND
```

CICS は、各 BMS マップ定義のテンプレートを作成する際、DFHMAPT プロシージャーの SYSPARM パラメーターで指定されているカスタマイズ・マクロを呼び出します。SYSPARM パラメーターでカスタマイズ・マクロの名前が指定されていない場合は、DFHMSX が使用されます。各マクロは順次処理され、可能な場合は、パラメーター値が保管されます。特定のマップまたはマップ・セットに対して重複パラメーターを指定した場合は、そのマップまたはマップ・セットだけに関する直前の値が新規値によって置き換えられます。

- この例の最初の DFHMDX マクロでは、`MAPSET=*,MAP=*` と `F3='Exit'` が指定されています。 F3 のこの値は、すべてのマップ・セット、および後続の DFHMDX マクロで異なる値が指定されていないすべてのマップに適用されます。
- 2 番目の DFHMDX マクロでは、`MAPSET=DFHWB0,MAP=*` と `F3='Messages'` が指定されています。 F3 のこの値は、後続の DFHMDX マクロで異なる値が指定されていないマップ・セット DFHWB0 内のすべてのマップに適用されます。
- 3 番目の DFHMDX マクロでは、`MAPSET=DFHWB0,MAP=DFHWB02` と `F3='Save'` が指定されています。この値は、マップ・セット DFHWB0 内のマップ DFHWB02 にのみ適用されます。

BMS マップから生成されたデフォルト・テンプレートには、以下のすべてのキーを表すボタンが含まれています。

- ENTER および CLEAR キー
- PA1、PA2、および PA3 キー
- ファンクション・キー F1 から F24
- HTML リセット

ただし、DFHMDX マクロを使用してテンプレートに必要なボタンを指定すると、指定したボタンだけがそのテンプレートに組み込まれます。例えば、次のように指定した場合、

```
DFHMDX MAPSET=*,MAP=*,F3='Exit',ENTER='Continue'
```

テンプレートには、F3 と ENTER キーのみのボタンが含まれます。

以下、さらにいくつか例を挙げて、BMS マップから生成された HTML テンプレートをカスタマイズする方法について説明します。

- **標準出力に含まれていないキーをアプリケーションで使用できるようにする。**

  次のようにしてボタンを AD001 マップに追加することができます。

  ```
  DFHMDX MAP=AD001,F1xx='Resubmit'
  ```

  ここで、Fxx は指定する新しいファンクション・キーの番号です。Web クライアントは、「Resubmit (再実行依頼)」という説明付きのボタンを表示します。ユーザーがこのボタンをクリックすると、それは Fxx としてアプリケーションに報告されます。

- **HTML リセット機能を抑制する。**

次のようにして AD001 マップのリセット機能を抑制することができます。

```
DFHMDX MAP=AD001,RESET=NO
```

Web クライアントは、リセット・ボタンを含まないページを表示します。

- **ボタンの外観またはボタンに関連付けられたテキストを変更する。**

  次のようにして F1 ボタンの説明を変更できます。

  ```
  DFHMDX F1='Help'
  ```

  Web クライアントは、「Help (ヘルプ)」という説明付きのボタンを表示します。ユーザーがこのボタンをクリックすると、それは F1 としてアプリケーションに伝えられます。

- **HTML ページに HTML 表題を付ける。**

  次のようにして表示マップに表題を追加することができます。

  ```
  DFHMDX MAP=DFHWB01,TITLE='CICS Web Interface'
  ```

  Web クライアントは、ページの表題として「CICS Web インターフェース」を表示します。

- **HTML ページにマストヘッド・グラフィックを付ける。**

  マストヘッドを持つマップの DFHMDX マクロを書きます。例えば、次のとおりです。

  ```
  DFHMDX MASTHEAD=(/dfhwbimg/masthead.gif,'CWI')
  ```

  Web クライアントは指定されたマストヘッドを使用します。あるいは図形ファイルを検出できない場合には、マストヘッドとして「CWI」を表示します。

- **バックグラウンドのカラーを変更、または特別のバックグラウンドを指定する。**

  特別のバックグラウンドを持つマップの DFHMDX マクロを書きます。例えば、次のとおりです。

  ```
  DFHMDX MAP=AD001,BACKGROUND=/dfhwbimg/texture4.jpeg
  ```

  Web クライアントは、指定されたファイルをページ用のバックグラウンドとして使用します。

  バックグラウンドの色を変更するには、BGCOLOR パラメーターを使用します。

- **BMS カラーを変更する。**

  BMS カラーを変更するには、次のような DFHMDX マクロを作成します。

  ```
  DFHMDX MAP=AD001,BLUE=AQUA,YELLOW=#FF8000
  ```

  Web クライアントは、BMS のブルー・テキストを HTML 水色 (BMS 青緑色と同じ) で表示し、 BMS のイエロー・テキストを明るいオレンジ色で表示します。

- **BMS マップの一部を抑制する。**

  次のようにしてマップ内のフィールドを抑制できます。

  ```
  DFHMDX MAP=AD001,SUPPRESS=((5,2),(6,2),(7,*))
  ```

表示されるページは、マップの行 5 列 2 のフィールド、行 6 列 2 のフィールド、および行 7 にあるフィールドをどれも含みません。

- **Web クライアント制御機能を追加する。**

  ページのロード時に JavaScript 機能が起動されるようにしたい場合は、カスタマイズ・マクロの中で DFHMDX マクロの ONLOAD パラメーターを使用します。例えば、次のようにコーディングした場合、

  ```
  DFHMDX MAP=AD001,ONLOAD='jset(''CWI is wonderful'',''Hello there!'')'
  ```

  ページのロード時に、与えられたパラメーターを持つ JavaScript 機能 `jset()` が起動されます。

  このカスタマイズを完了するには、`jset` 関数の定義を DFHWBOUT マクロで HTML ページのヘッダーに追加します。 BMS マップ定義の最初の DFHMDF マクロの前にこのマクロの呼び出しを書き込む必要があります。例えば、

  ```
  DFHWBOUT '<script language="JavaScript">'
  DFHWBOUT 'function jset(msg,wng)'
  DFHWBOUT '         {window.status = msg; alert(wng)}'
  DFHWBOUT '</script>'
  ```

  ページのロード時に、ウィンドウの下部の状況域にはメッセージ "CWI is wonderful" が表示され、メッセージ "Hello there!" が含まれるアラート・ウィンドウがオープンされます。

- **HTML ページだけに現れ、BMS マップの一部ではないテキストを追加する。**

  BMS マップ定義内で、テキストを表示したいポイントに DFHWBOUT マクロを書き込みます。例えば、次のとおりです。

  ```
  DFHWBOUT '<p>This text illustrates the use of the DFHWBOUT macro,'
  DFHWBOUT 'which can be used to output text that should only appear'
  DFHWBOUT 'in HTML templates, and will never appear in the'
  DFHWBOUT 'corresponding BMS map.'
  ```

  これにより、以下の行が HTML テンプレートに生成されます。

  ```
  <p>This text illustrates the use of the DFHWBOUT macro,
  which can be used to output text that should only appear
  in HTML templates, and will never appear in the
  corresponding BMS map.
  ```

- **HTML ヘッダー情報を HTML ページに追加する。**

  BMS マップ定義の中の最初の DFHMDF のオカレンスの前に DFHWBOUT マクロを書き込みます。例えば、次のとおりです。

  ```
  DFHWBOUT '<meta name="author" content="E Phillips Oppenheim">'
   DFHWBOUT '<meta name="owner"  content="epoppenh@xxxxxxx.yyy.co*
         m">'
   DFHWBOUT '<meta name="review" content="19980101">'
   DFHWBOUT '<meta http-equiv="Last-Modified" content="&WBDATE&W*
         BTIME GMT">'
  ```

  これにより、HTML テンプレートのヘッド・セクションに以下の行が生成されます。

  ```
  <meta name="author" content="E Phillips Oppenheim">
  <meta name="owner"  content="epoppenh@xxxxxxx.yyy.com">
  <meta name="review" content="19980101">
  <meta http-equiv="Last-Modified" content="23-Dec-1997 12:06:46 GMT">
  ```

DFHMSD は、&WBDATE および &WBTIME の値を、マクロがアセンブルされた日時に設定します。

- **国特有の文字を JavaScript および HTML で使用する。**

  テンプレートの作成に使用されるデフォルトの US コード・ページ 37 は、異なるコード・ページ用に変更できます。例えば、次のとおりです。

  ```
  DFHMDX OPENSQ=[,CLOSESQ=],OPENBR={,CLOSEBR=},EXCLAM=!
  ```

  必要な置換を指定します。これらの文字は、コード・ページが DFHCNV 呼び出しの SERVERCP に対応している端末で入力する必要があります。

- **HTML ページのフィールドを右寄せする。**

  フィールドのデータは、次のようにして右寄せできます。

  ```
  DFHMDX MAPSET=MAPSETA,MAP=AD001,RALIGN=((3,5),(*,15),(*,3),(6,7),(*,83))
  ```

  この例では、以下のすべてのフィールドでデータの右寄せが行われています。

  ```
  DFHMDF POS=(3,5),LENGTH=4,INITIAL='TEXT',ATTRB=PROT
  DFHMDF POS=(5,80),LENGTH=3,INITIAL='123',ATTRB=PROT
  DFHMDF POS=(2,10),LENGTH=5,INITIAL='  EXT',ATTRB=ASKIP
  DFHMDF POS=(4,8),LENGTH=7,INITIAL='INITEX ',ATTRB=PROT
  DFHMDF POS=(1,1),LENGTH=2,XINIT='C1C2',ATTRB=ASKIP
  DFHMDF POS=(6,7),LENGTH=4,XINIT='0E44850F',ATTRB=PROT,SOSI=YES
  DFHMDF POS=(2,9),LENGTH=6,XINIT='0E448544830F',SOSI=YES,ATTRB=PROT
  DFHMDF POS=(2,9),LENGTH=6,XINIT='448544834040',PS=8,ATTRB=PROT
  ```

- **数値フィールドを右寄せする。**

  NUMERIC 属性を持つすべてのフィールドを、それらの HTML テーブル・セル内で次のように右に位置合わせすることができます。

  ```
  DFHMDX MAPSET=MAPSETA,MAP=AD001,NUMALIGN=YES
  ```

第 15 章 CICSPlex の CICS Web サポート

CICSPlex® で CICS Web サポートを使用するアプリケーションを、さまざまな方法を個別にまたは組み合わせて使用して分散できます。

- Web クライアントから複数の CICS 領域に要求を分散するために、ネットワーク・ロード・バランシングを使用できます。
- CICS Web サポートおよびビジネス・アプリケーションを、同じ CICS 領域で実行できます。
- CICS Web サポートは、ルーター領域で実行し、ビジネス・アプリケーションは、1 つ以上のアプリケーション所有の領域 (AOR) で実行できます。ただし、AOR で **EXEC CICS** WEB API を使用できないため、Web 対応アプリケーション・プログラムは AOR で実行できません。AOR の中で **EXEC CICS** DOCUMENT API を使用することはできますが、HTML 出力をルーター領域に転送して戻すメカニズムをユーザー自身で用意する必要があります。 234 ページの『AOR への web クライアントの要求のルーティング』を参照してください。

図 10 は、このような構成を図示したものです。

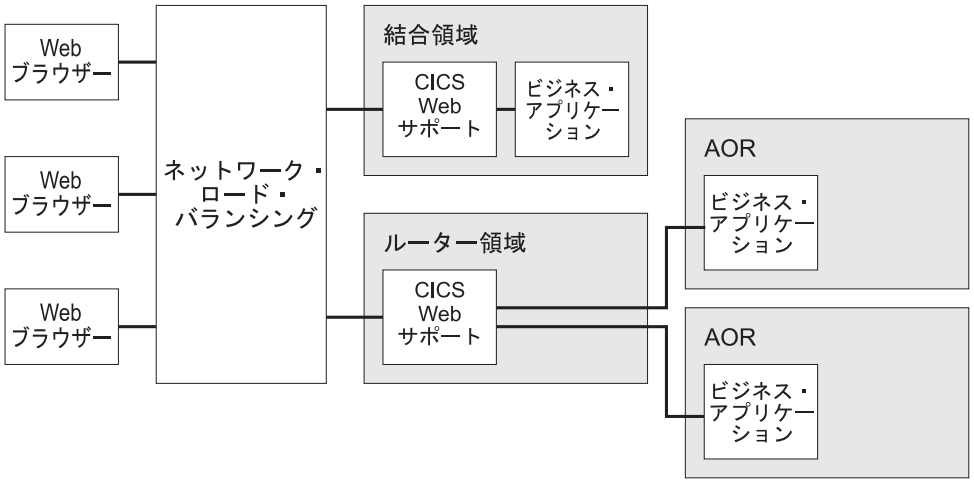


図 10. CICSPlex の CICS Web サポート構成

CICS ビジネス・ロジック・インターフェースを使用する要求を同様の方法で分散できます (234 ページの図 11を参照)。

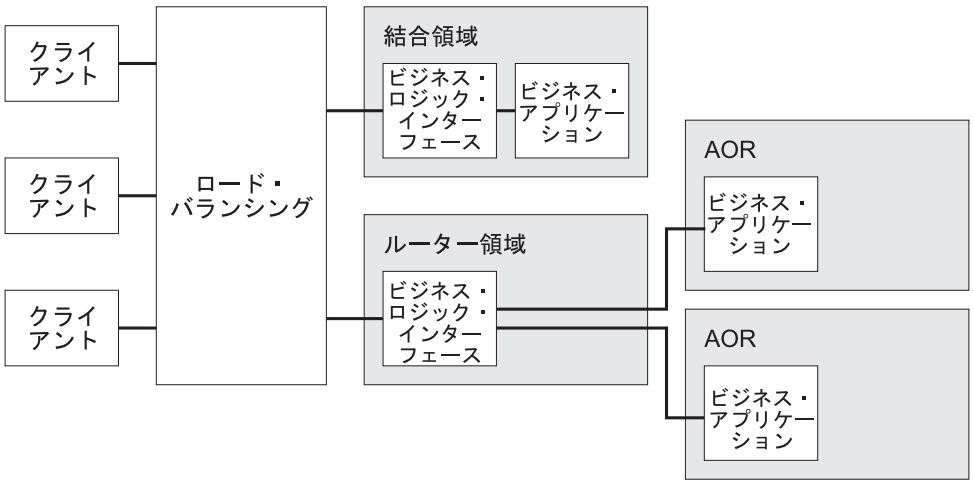


図 11. CICSPlex の CICS ビジネス・ロジック・インターフェースの構成

アプリケーションをこのように分散する場合は、アプリケーションの各部分の間に存在する類縁性を考慮してください。類縁性について詳しくは、「*CICS Interdependency Analyzer for z/OS User's Guide and Reference*」を参照してください。

要求間でアプリケーション状態をどのように管理するかも考慮に入れてください。106 ページの『HTTP 要求シーケンス中のアプリケーション状態の管理』で、疑似会話型モデルを使用する CICS Web サポート・アプリケーションについての考慮事項を解説しています。以下の場合に追加の考慮事項が該当することがあります。

- ビジネス・アプリケーションが実行されている AOR を選択するために動的ルーティングが使用される場合
- ルーター領域および間接的に AOR を選択するためにワークロードおよび接続バランシングが使用される場合

CICS は、ユーザーのアプリケーション状態を管理するために使用できる状態管理プログラム (DFH$WBSR) のサンプルを提供しています。 DFH$WBSR は、いくつかの CICS 領域が共用できるリソースを介してアプリケーションの状態を共用するのに役立ちます。 507 ページの『付録 J. 状態管理サンプルの DFH$WBST および DFH$WBSR に関する参照情報』を参照してください。(もう 1 つのサンプルの DFH$WBST は、類縁性を作成するので、CICSPlex での使用には適していません)。

CICS Web サポートの構成および CICSPlex の CICS ビジネス・ロジック・インターフェースについて詳しくは、「*Workload Management for Web Access to CICS*」を参照してください。

AOR への web クライアントの要求のルーティング

Web 対応アプリケーションを使用して Web クライアント要求に応答できます。Web 非対応アプリケーションの場合は、ルーター領域でコンバーター・プログラムを使用できます。

このタスクについて

要求に応答するために Web 対応アプリケーションを使用する場合、解決策の 1 つとして、(ルーター領域で実行されている) Web 対応アプリケーション・プログラムの中で自分用の表示ロジックをコーディングして、(AOR で実行されている) 自分用のビジネス・ロジックをコーディングし、全体に渡って表示とは独立させることができます。 Web 対応アプリケーション・プログラムは、要求を処理するプログラムとして指定され、それ自身とビジネス・ロジックを実行するアプリケーション・プログラムとの間の通信を管理する必要があります。 31 ページの『HTTP サーバーとしての CICS に対する HTTP 要求および応答の処理』では、Web 対応アプリケーションの処理ステージについて説明します。

EXEC CICS WEB API を、AOR で使用することはできません。これは、ルーター領域でのみ使用できます。

Web 非対応アプリケーションの場合は、ルーター領域のコンバーター・プログラムを使用して、AOR のアプリケーション・プログラムが提供する情報から HTTP 応答を生成できます。 236 ページの図 12 は、アプリケーション所有領域で Web 非対応アプリケーション・プログラムが実行されるときに Web クライアントからの要求に関連付けられる処理ステージおよびタスク構造を示しています。

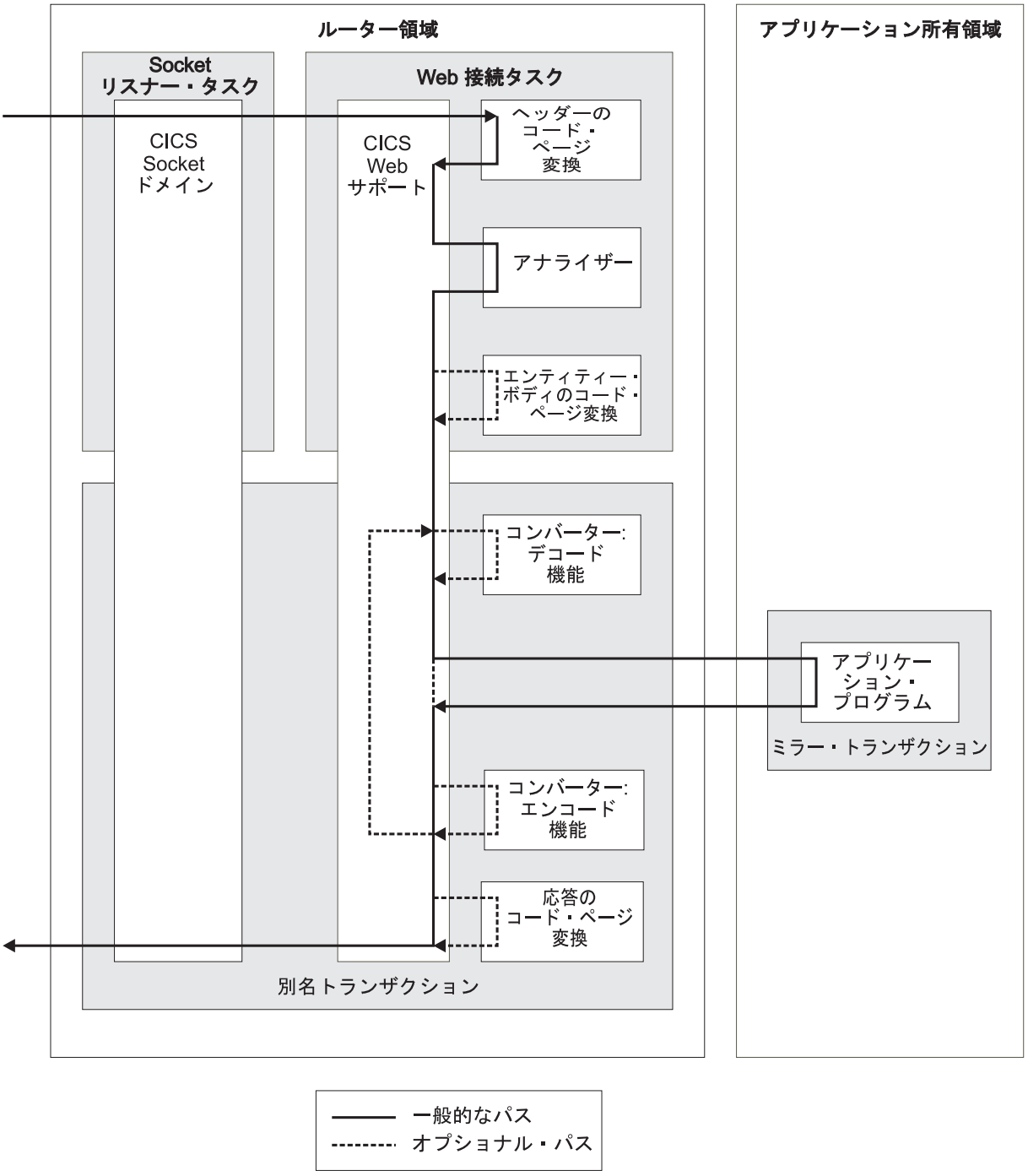


図 12. CICS Web サポートが AOR に Web 非対応アプリケーション要求を経路指定する方法

CICS ビジネス・ロジックに対応するステージについて詳しくは、 237 ページの図 13を参照してください。

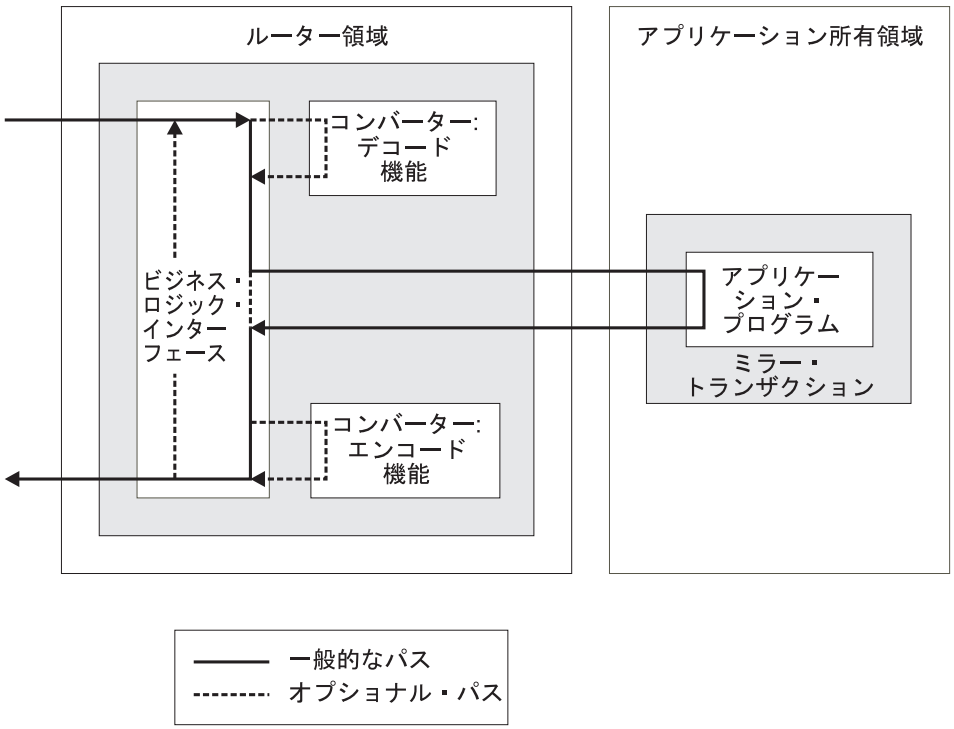


図 13. CICS ビジネス・ロジック・インターフェースが AOR にアプリケーション要求を経路指定する方法

CICS は、Distributed Program Link (DPL) を使用して、AOR にあるアプリケーション・プログラムを呼び出します。そのアプリケーションは、ミラー・タスクの下で実行されます。 DPL について詳しくは、DPL の概要を参照してください。

ビジネス・アプリケーションを AOR で実行するときは、

- REMOTESYSTEM 属性を指定するか、アプリケーション・プログラムの PROGRAM 定義で DYNAMIC(YES) を指定してください。 DYNAMIC(YES) を指定すると、アプリケーション・プログラムをどこで実行するかを、動的ルーティング・プログラムが判断します。
- (アナライザー・プログラム、Web 対応アプリケーション・プログラムまたはコンバーター・プログラム、および別名トランザクション用の) その他のリソース定義でそれらがルーター領域で実行されることを指定する必要があります。
- ルーター領域と AOR の間の MRO または APPC 接続を定義する必要があります。

AOR で実行されるアプリケーション・プログラムが表示から完全に独立している場合、アプリケーション・プログラムは Web 対応アプリケーション・プログラムまたはコンバーター・プログラムに出力を返します。次にこれらのプログラムが HTML 出力を構成します。または、コンバーター・プログラムを使用している場合、AOR の EXEC CICS DOCUMENT API を使用して、HTML 出力を作成する場合があります。コンバーター・プログラムは、この出力を使用して完全な HTTP 応答を作成できます。

アプリケーション・プログラムの出力をルーター領域に戻すには、独自のメカニズムを作成する必要があります。出力は COMMAREA に転送できます。または、一

時記憶域キューなど、他のメカニズムを使用して、そのメカニズムの中でデータを表すトークンを転送することができます。ルーター領域のプログラムは、トークンを使用して出力を取得し、処理してから Web クライアントに渡します。 CICS は、この処理を実行できる状態管理プログラム (DFH$WBSR) のサンプルを提供しています。 507 ページの『付録 J. 状態管理サンプルの DFH$WBST および DFH$WBSR に関する参照情報』を参照してください。もう 1 つのサンプルの DFH$WBST は、類縁性を作成するので、CICSPlex での使用には適していません。

ネットワーク・ロード・バランシング

単一の CICS ルーター領域に依存するのを避けるために、複数のルーター領域を使用して、ネットワークからのワークロードを共用することができます。

このタスクについて

ルーター領域間でワークロードのバランスを取るには、以下のようないくつかの手法を使用できます。

シスプレックス・ディストリビューター

シスプレックス・ディストリビューターは、動的なアプリケーション固有の仮想 IP アドレス (VIPA) であるクラスター IP アドレスを使用する z/OS Communications Server の機能であり、シスプレックス内の CICS サーバー・インスタンスのクラスターを表します。シスプレックス・ディストリビューターについて詳しくは、「*z/OS Communications Server:構成ガイド*」(SC88-8926) を参照してください。

アプリケーション固有の仮想 IP アドレス (VIPA)

アプリケーション固有の VIPA は、z/OS Communications Server の機能であり、バインドがアクティブにされた仮想 IP アドレスを提供します。CICS 領域が開始されるとき、VIPA は TCP/IP スタックでアクティブになり、VIPA は特定の CICS 領域を持つシスプレックスの周りを移動できます。 VIPA について詳しくは、「*z/OS Communications Server: IP 構成ガイド*」(SC88-8926) を参照してください。

DNS のアプローチ

DNS 接続最適化は、z/OS シスプレックス IP ドメインにおいて、登録されたアプリケーションの正常性について MVS™ WLM からのフィードバックを基にして、IP 接続のバランスをとります。 CICS の使用に対してまだサポートされています。DNS/WLM のサポートについて詳しくは、「*z/OS Communications Server: IP 構成ガイド*」(SC88-8926) を参照してください。

ポートの共用

TCP/IP ポート共用は、同じ z/OS イメージで実行されている CICS ルーター領域のグループ全体にわたって HTTP 要求を分散するための直接的な方式として使用できます。異なる領域にある CICS TCP/IP サービスは、同じポートで listen するよう構成されているので、TCP/IP は SHAREPORT オプションを付けて構成されています。次に TCP/IP スタックは、リスナー全体にわたって接続要求のバランスをとります。 TCP/IP ポート共用について詳しくは、「*z/OS Communications Server: IP 構成解説書*」(SC88-8927) を参照してください。

第 16 章 CICS Web サポートのパフォーマンスとチューニング

CICS Web サポートのパフォーマンスを向上させるために、システムのさまざまな面を調整できます。

- それぞれの CICS 領域において、HTTP サーバーとしての CICS と Web クライアント間、または HTTP クライアントとしての CICS と Web 上のサーバー間の同時接続の最大数は、理論上は 64,000 までです。以下のように、MXT 設定が同時接続数を直接制限するわけではありません。
  - HTTP サーバーとしての CICS の場合、CWXN トランザクション (Web 接続タスク) は持続接続の期間中はシステムに残っておらず、しかも各要求の後に終了します。つまり、要求相互間で 1 つの持続接続が存在でき、この持続接続はアクティブ・タスクに関連付けられることなく、Socket リスナー・タスク (CSOL) によってモニターされることになります。
  - HTTP クライアントとしての CICS の場合、アプリケーション・プログラムは Web 上のサーバーへの持続接続を複数開いて維持できます。これらの持続接続は、アプリケーション・プログラムの単一アクティブ・タスクによってカバーされます。

  ただし、CICS Web サポート・トランザクションのトランザクション・クラス定義で設定する MXT 設定および何らかの制限を使用して、CICS 領域内の CICS Web サポート・アクティビティーの量を制限できます。
- HTTP サーバーとしての CICS への長期持続接続が複数の Web クライアントによってセットアップされ、使用されるそれらの接続の負荷が重い場合は、接続を扱う CICS 領域が過負荷になり、パフォーマンス上の問題が起こる可能性があります。この問題が発生した場合は、接続スロットリングをセットアップすることにより、過剰な Web クライアントを、ポートを共用して同じサービスを提供する他の CICS 領域に接続できます。
  - TCPIPSERVICE リソース定義の MAXPERSIST 属性を使用して、特定のポートに対して CICS 領域が受け入れる持続 HTTP 接続の数の制限を設定します。その後の Web クライアントは接続スロットリングの対象となり、各応答の後にそれぞれの接続を閉じる必要があります。新たなクライアントが再接続するとき、ポートを共用していて制限に達していない別の CICS 領域に接続した場合に、そこで持続接続を維持できます。
  - HTTP/1.1 サーバーは通常、持続接続を許可する必要があります。したがって、長期持続接続が原因でパフォーマンス上の問題が起こった CICS 領域にのみ接続スロットリングをセットアップしてください。持続接続の数を制限しても、HTTP サーバーとしての CICS は HTTP/1.1 仕様に準拠します (ゼロ制限を設定しなければ)。しかし、持続接続を拒否することは、通常のプラクティスとしては推奨されません。
  - Web クライアントが目的の持続接続の取得に失敗すると、接続スロットリングが実施されている CICS 領域に再接続する場合は特に、Web クライアント自身のパフォーマンスが影響を受けることがあります。ポートを共用していて制限に達していない別の CICS 領域に Web クライアントが接続すると、通常のパフォーマンスに戻ります。

持続接続および接続スロットリングについて詳しくは、 46 ページの『CICS Web サポートによる持続接続の処理方法』を参照してください。

- 実際面では、単一 CICS 領域と Web 間でアクティブ状態が可能な接続の数は、CICS 領域で使用できるストレージによって第一に制限されます。『CICS Web サポートのストレージ要件』で、CICS Web サポート・アクティビティーの最重要ストレージ要件を説明しています。
- HTTP サーバーとしての CICS の場合、CICS Web サポートに伴うさまざまなトランザクションの優先順位が重要になります。設定を誤ると、ストレージ不足状態になるおそれがあります。 242 ページの『CICS Web サポート・トランザクション (CWXN、CWXU、CWBA、CW2A) の優先順位』で、この領域の問題を回避するための優先順位の設定方法を説明しています。
- HTTP サーバーとしての CICS の場合、HTTP 応答はアプリケーションが生成することも、静的文書 (HFS ファイルまたは CICS 文書テンプレートのどちらか) から提供することもできます。 CICS 文書テンプレートとして、さまざまな異なる CICS リソース (プログラム、区分データ・セット、ファイルなど) の 1 つを使用できます。 242 ページの『CICS Web サポートの応答方式の相対パフォーマンス』で、これらの応答タイプ間のパフォーマンスの違いを説明しています。
- HTTP クライアントとしての CICS の場合、デフォルトでは、アプリケーションが接続を使い終わると、CICS はクライアント HTTP 接続を閉じます。接続プーリングがセットアップされている場合、CICS は接続を開いたままにしておき、休止状態でプールに入れることができます。この接続は、同じアプリケーションが再使用することも、同じホストとポートに接続する別のアプリケーションが再使用することも可能です。 244 ページの『HTTP クライアントのパフォーマンスのための接続プーリング』で、CICS Web サポート・アプリケーションおよび Web サービス・アプリケーションにとって接続プーリングによるパフォーマンス上の効果が得られるシチュエーションを説明しています。
- SSL (Secure Sockets Layer) を使用する場合は、セキュリティー手段 (暗号化と暗号化解除および SSL ハンドシェーク) のためにトランザクション当たりの CPU がわずかに増加します。HTTP クライアントとしての CICS の場合、接続プーリングを実装すると、SSL を使用して既に接続されているクライアント HTTP 接続を、CICS アプリケーションが再使用できるようになります。接続が再使用されるとセキュリティー手段を繰り返す必要がないので、プールから接続を与えられたアプリケーションでは、このような CPU 増加は起こりません。SSL は S8 TCB を使用しますが、この TCB はディスパッチャー TCB 統計を使用してモニターできます。 CICS が使用する TCB の数を制御するシステム初期設定パラメーターの設定に関するガイダンスについては、「*CICS パフォーマンス・ガイド*」のCICS ディスパッチャー: パフォーマンスおよび調整を参照してください。

CICS Web サポートのストレージ要件

単一 CICS 領域と Web 間で持続可能な接続の数は、CICS 領域で使用できるストレージによって第一に制限されます。

CICS Web サポートのストレージ使用量に影響を与えるものとして、主に以下があります。

- **接続ごとの基本ストレージ要件**。CICS と Web 間の接続ごとに、約 4K のストレージが必要となります。

- **HTTP サーバーとしての CICS の場合: Web クライアントから受け取る要求のサイズ。** 単一の要求に対して CICS が受け入れるデータ合計量は、TCPIPSERVICE 定義の MAXDATALEN 属性で制限できます。
  - 要求行と HTTP ヘッダーは、コンテナーに保管されます。
  - 要求ボディは、CICS 主記憶域に保管されます。要求ボディに含まれる不要データはアプリケーション・プログラムでは無視できますが、Web 接続タスクは完全メッセージを常に読み取ります。ソケットからデータをクリアし、すべてのデータをストレージに入れるためです。
  - Web クライアントから受信した要求に使用されるストレージは、要求に応答が送信されると解放されます。
- **HTTP クライアントとしての CICS の場合: Web サーバーから受け取る応答のサイズ。** 1 つの応答に対して CICS が受け入れるデータ量を制限するための具体的な方法はありません。(CICS Web サポートの HTTP クライアント機能は、ブラウザーとして使用するようには設計されていません。そのため、要求は選択された既知のリソースのみ返します。)
  - 状況表示行と HTTP ヘッダーは、コンテナーに保管されます。
  - 応答ボディは、CICS 主記憶域に保管されます。応答ボディに含まれる不要データはアプリケーション・プログラムでは無視できますが、完全メッセージが読み取られて CICS ストレージに入れられます。
  - Web サーバーとの接続の期間中は、Web サーバーから受信する応答に使用されるストレージが必要となります。最初の応答のために取得されたストレージは、その接続で受信される後続の各応答によって上書きされます。より大きいストレージ量を後続の応答が必要とする場合は、解放されて再取得されます。アプリケーション・プログラムが WEB CLOSE コマンドを使用して接続を閉じると、またはタスクの終了時に (接続がまだ閉じられていない場合)、ストレージは解放されます。したがって、接続ごとに取得されるストレージの最大量は、その接続で受信する最大メッセージのサイズに等しくなります。
- **HTTP サーバーとしての CICS および HTTP クライアントとしての CICS の場合: CICS によって生成されるメッセージ (要求または応答) のサイズ。**CICS からの送信のために要求または応答がアセンブルされている間は、HTTP ヘッダーのほかメッセージ・ボディのためのストレージが必要となります。
  - HTTP ヘッダーは、コンテナーに保管されます。
  - メッセージ・ボディは、CICS 主記憶域に保管されます。
  - WEB SEND (クライアント/サーバー) および WEB CONVERSE コマンドの場合、ストレージを 2 とおりの方法で解放できます。これらのオプションは、送信された文書の内容を (そのトランザクション内で後で) 取り出すつもりがない場合のみ、使用するようにしてください。
    - WEB SEND または CONVERSE コマンドに ACTION(IMMEDIATE) オプションを指定します。
    - WEB SEND コマンドに ACTION(EVENTUAL) および DOCSTATUS(DOCDELETE) オプションを指定します。
- **コード・ページ変換。**CICS によって受信または送信されるメッセージ・ボディのコード・ページ変換が行われることがあります (要求の処理パスに含まれるアプリケーション・プログラム、アナライザー・プログラム、または URIMAP 定義で指定された場合)。メッセージ・ボディは、CICS 主記憶域にあります。

– EBCDIC コード・ページ 037 と ASCII コード・ページ ISO-8859-1 間の変換の場合、CICS は変換後のメッセージ・ボディを元のメッセージ・ボディと同じストレージ域に書き込むので、追加ストレージは使用されません。
– その他のタイプのコード・ページ変換の場合、 CICS では、変換されたメッセージ・ボディを格納する追加のストレージが必要です。使用される文字セットに応じて、この追加ストレージ域のサイズの範囲は、オリジナルのメッセージ・ボディと同じサイズから、理論上の最大値であるオリジナルのメッセージ・ボディの 4 倍のサイズ (非常に稀な場合) までになります。例えば、メッセージ・ボディのデータが 2 MB の場合は、少なくとも合計 4 MB のストレージが必要となります。2 バイト文字セット (DBCS) またはマルチバイト文字セットでは、通常、この範囲内で、より大容量のストレージ域が必要です。

CICS Web サポート・トランザクション (CWXN、CWXU、CWBA、CW2A) の優先順位

Web クライアントから要求を受け取って初期検査を行うために、Web 接続タスクが使用されます。 Web 接続タスクのデフォルトのトランザクション ID は、CWXN (HTTP プロトコルの場合) または CWXU (USER プロトコルの場合) です。アプリケーション生成応答のための後続処理をカバーするために、別名トランザクションが使用されます。 CICS Web サポート別名トランザクションのデフォルトのトランザクション ID は CWBA であり、Atom フィード別名トランザクションのデフォルトのトランザクション ID は CW2A です。

「*CICS インターネット・ガイド*」のCICS Web サポートのタスク構造で、CICS Web サポート処理に使用されるトランザクションとそれらの相互作用について詳しく説明しています。

CWXN または CWXU トランザクション (あるいはその別名) の優先順位を、CWBA や CW2A などの別名トランザクションの優先順位より高く設定すると、受信されたが未処理の要求が蓄積されることになる可能性があり、その場合はストレージ不足状態になるおそれがあります。

CWXN および CWXU トランザクションのデフォルト優先順位は 1 に設定されており、CICS 提供の CWBA および CW2A トランザクションのデフォルト優先順位も 1 に設定されています。これらの優先順位は、ワークロードに応じて調整できます。 CWBA や CW2A のような別名トランザクションの優先順位を、CWXN や CWXU のような Web 接続タスク関連のトランザクションの優先順位以上にしておく必要があります。 CICS 提供のデフォルトの代わりに独自のトランザクション定義をセットアップする場合は、このことを念頭に置いてください。

CICS Web サポートの応答方式の相対パフォーマンス

アプリケーション生成応答は、静的応答よりも多くのリソースを使用します。アプリケーション生成応答では、別名トランザクションを接続する必要があります。要求の処理と応答の生成には、アナライザー・プログラムやコンバーター・プログラム、あるいは複数のユーザーが作成したアプリケーション・プログラムを伴う場合があります。通常は、応答の生成に、より多くの経過時間とプロセッサー時間を要します。

アプリケーション生成応答は、静的応答よりも多くのリソースを使用します。アプリケーション生成応答では、別名トランザクションを接続する必要があります。要求の処理と応答の生成には、アナライザー・プログラムやコンバーター・プログラム、あるいは複数のユーザーが作成したアプリケーション・プログラムを伴う場合があります。通常は、応答の生成に、より多くの経過時間とプロセッサー時間を要します。

静的応答には、Web 接続タスク、URIMAP 定義、および応答ボディのソース文書のみが関与します。静的応答のパフォーマンスは通常、アプリケーション生成応答よりも優れています。したがって、アプリケーション・プログラムとアナライザー・プログラムが含まれるアーキテクチャーを使用して単純な応答文書を配信する場合は、それを静的応答に変換することを考慮する必要があります。

このカテゴリーでは、応答ボディに使用されるソース文書の選択によって、パフォーマンスがさらに影響を受けます。以下の選択肢があります。

- z/OS UNIX システム・サービスの HFS ファイル。HFSFILE オプションを使用している URIMAP 定義から直接呼び出されます。
- z/OS UNIX システム・サービスの HFS ファイル。CICS 文書テンプレートと定義されたファイルで、DOCTEMPLATE オプションを使用している URIMAP 定義から呼び出されます。
- MVS 区分データ・セットまたは PDSE に保管されている文書テンプレート。
- 一時データ・キューに保管されている文書テンプレート。
- 一時記憶域キューに保管されている文書テンプレート。
- CICS ファイル (ESDS、RRDS、または別のタイプのデータ・セット) に保管されている文書テンプレート。
- CICS プログラムに含まれる文書テンプレート。
- 出口プログラムによって生成される文書テンプレート。文書テンプレートの内容は、DB2 または別のデータベース・マネージャーなどの場所からロードすることもできます。出口プログラムによって生成された文書テンプレートは、それらが Web 対応アプリケーション・プログラムを通じてではなく URIMAP 定義を通じて機能するときは、静的応答と同類になります。ただし、それらはアプリケーション・プログラムを必要とするので、リソースとパフォーマンスの観点から見ると、アプリケーション生成応答と類似していると言えます。

「*CICS アプリケーション・プログラミング・ガイド*」で、さまざまなタイプの CICS 文書テンプレートとそれらのセットアップ方法について詳しく説明しています。 CICS 文書テンプレートを使用して静的応答を提供する場合は、使用前に定義をインストールしておく必要があります。

文書テンプレートがインストールされると、CICS 文書ハンドラーは、パフォーマンスを高めるために、ほとんどの文書テンプレートのコピーをキャッシュに入れます。その後のテンプレート参照では、キャッシュに入れられたコピーが使用されます。これは、文書テンプレート・タイプの相対アクセス速度は、ソースから最初に取り出された後は、重要ではないことを意味します。 CICS プログラムから取り出される文書テンプレートの場合、キャッシングは行われません。プログラムは CICS

ローダーによって既に管理されており、取り出し時間が速いからです。出口プログラムによって生成される文書テンプレートの場合は、コピーをキャッシュに入れるかどうかを指定できます。

ストレージが制約されているときは、文書テンプレートのパフォーマンスが影響を受けることがあります。文書テンプレートが含まれるプログラムは他の CICS ロード済みプログラムのように管理され、プログラム圧縮によりフラッシュアウトされる場合があります。文書ハンドラーによってキャッシュに入れられた文書テンプレートは解放される場合があり、それらの文書テンプレートの次回参照時は、ソースから取り出すことが必要になります。

HTTP クライアントのパフォーマンスのための接続プーリング

HTTP クライアントとしての CICS の場合、CICS Web サポート・アプリケーション、Web サービス・アプリケーション、または HTTP EP アダプターの複数の呼び出しにより同じホストおよびポートに対する接続要求が行われるときや、Web サービス・アプリケーションが複数の要求および応答を行うとき、接続プーリングによるパフォーマンス上の効果が得られます。

デフォルトでは、クライアント HTTP 接続が使用された後、CICS は以下のように接続を閉じます。

- HTTP クライアント要求を行う CICS Web サポート・アプリケーションの場合、アプリケーション・プログラムが接続を使い終わったことを **WEB CLOSE** コマンドを発行して CICS に通知すると、CICS は接続を閉じます。アプリケーションが **WEB CLOSE** コマンドを発行しない場合、タスクの終わりに達すると、CICS は接続を閉じます。
- サービス要求元である CICS Web サービス・アプリケーションの場合、アプリケーション・プログラムが INVOKE SERVICE コマンドを使用してサービス要求を発行して応答を受け取った (該当する場合) 後、CICS は接続を閉じます。
- CICS イベント処理の場合、HTTP EP アダプターがビジネス・イベントを出力した後、CICS は接続を閉じます。

接続プーリングをセットアップすると、CICS はクライアント HTTP 接続を使用後に閉じる代わりに、その接続を開いたままにしておき、休止状態でプールに保管します。休止接続は、同じアプリケーションが再使用することも、同じホストとポートに接続する別のアプリケーションが再使用することも可能です。クライアント HTTP 接続を開くコマンドをアプリケーションが発行すると、CICS はそのホストとポートに使用できる休止接続がプールにあるかどうかを検査し、ある場合は新しい接続を開くのではなく、その休止接続をアプリケーションに提供します。

プール接続が再使用される場合は、新しい接続を開くことと比較して、CPU の使用が少なくなります。接続が SSL (Secure Sockets Layer) を使用する場合は、さらに削減が見られます。接続が再使用されると、セキュリティー手段を繰り返す必要がないからです。アプリケーションは新しい接続を使用するのとまったく同じようにプール接続を再使用するので、接続は使用後に再びプールすることができます。

接続プーリングは、URIMAP リソースの SOCKETCLOSE 属性で指定します。この属性は、CICS アプリケーションがクライアント HTTP 接続を使い終わった後に

CICS がそのソケットを開いたままにしておくかどうかとその期間を定義します。アプリケーションは、接続を開くときに URIMAP リソースを指定する必要があります。

HTTP EP アダプターは、接続を開くための URIMAP リソースを必要とします。URIMAP リソースは、CICS Web サポートおよび Web サービス・アプリケーションからの HTTP クライアント要求の要件ではありませんが、いくつかの利点があります。クライアント接続に URIMAP リソースを使用すると、接続プーリングの使用可能化に加えて、以下のような利点があります。接続のエンドポイントのどのような変更もシステム管理者が管理できるので、サービス・プロバイダーの URI が変わってもアプリケーションを再コンパイルする必要がありません。 CICS Web サポート・アプリケーションの場合は、URIMAP リソースを使用して、SSL のクライアント証明書および暗号スイート・コードの保管と管理も行えます。

以下のシナリオでクライアント HTTP 接続を使用する CICS アプリケーションの場合に、接続プーリングによるパフォーマンス上の効果が得られます。

- 複数の CICS Web サポート・アプリケーション (同じアプリケーションかまたは異なるアプリケーションの呼び出し) が、同じホストとポートに対して HTTP クライアント要求を行う。
- サービス要求元である複数の CICS Web サービス・アプリケーションが、同じ Web サービス・プロバイダーに対して要求を行う。
- サービス要求元である CICS Web サービス・アプリケーションが、単一タスク内で Web サービス・プロバイダーに対して複数の要求を行う。
- イベント・バインディングで指定された HTTP サーバーに、HTTP EP アダプターが頻繁にイベントを送信する。

ユーザー作成 CICS Web サポート・アプリケーションの単一呼び出しや、Web サービスの単一の要求および応答、あるいは単一イベント出力のパフォーマンスが、接続プーリングによって向上するわけではありません。

CICS 領域内のいずれかのクライアント HTTP 接続が接続プーリングに適した候補である場合は、以下のように実装します。

- CICS Web サポート、CICS Web サービス、または CICS イベント処理に選択したクライアント HTTP 接続の URIMAP リソース定義で、SOCKETCLOSE 属性を指定します。 CICS が各プール接続を開いたままにしておくことを開始してから廃棄するまでの時間の長さを選択できます。 SOCKETCLOSE 属性については、URIMAP リソースを参照してください。
- CICS Web サポートおよび Web サービス・アプリケーションの場合は、クライアント HTTP 接続を開くコマンドで CICS アプリケーションが URIMAP リソース定義を指定するようにします。 CICS Web サポート・アプリケーションは **WEB OPEN** コマンドを使用してクライアント HTTP 接続を開き、CICS Web サービス・アプリケーションは **INVOKE SERVICE** コマンドまたはそのシノニムの **INVOKE WEBSERVICE** を使用してクライアント HTTP 接続を開きます。アプリケーションはコマンドで URI を直接指定してはなりません。サービス要求元の場合は、Web サービス記述に含まれる URI を使用します。
- CICS Web サポート・アプリケーションの場合のみ、アプリケーションが発行する **WEB SEND** または **WEB CONVERSE** コマンドのいずれでも、オプション

CLOSESTATUS(CLOSE) が使用されていないことを確認します。 CLOSESTATUS(CLOSE) は接続を閉じるようサーバーに要求しますが、閉じられた接続はプールできません。

- CICS Web サポート・アプリケーションの場合のみ、アプリケーションがクライアント HTTP 接続を使い終わったときに **WEB CLOSE** コマンドを発行していることを確認します。 CICS Web サポート・アプリケーションが **WEB CLOSE** コマンドを発行しない場合、接続が良好な状態でなければ、CICS は接続をプールに入れません。 CICS Web サービス・アプリケーションでは、**INVOKE SERVICE** コマンドがアプリケーションの接続使用を完了するので、CICS Web サービス・アプリケーションは接続使用を完了するための追加コマンドを発行する必要はありません。

CICS による接続プーリングの管理

接続プーリングはアプリケーションに認識されないので、URIMAP リソース定義で SOCKETCLOSE 属性を指定した後は、管理者からの関与は必要ありません。CICS は、ゼロ以外の SOCKETCLOSE 属性を持つ URIMAP リソースごとに接続のプールを保持します。

HTTP クライアント接続を接続のプールに入れる前に、CICS は接続の状態を検査します。不完全な状態であることが分かったまたは疑われる接続は、プールに入れられません。例えば、以下の状態の場合、接続は不完全な状態になります。

- サーバーからの最後の HTTP 応答が OK でなかった。
- 接続での要求の数と応答の数が等しくない。
- CICS アプリケーションが受け取っていないデータが接続にまだ存在する。
- 接続を開いた CICS Web サポート・アプリケーションが、接続を使い終わったことを **WEB CLOSE** コマンドを発行して CICS に通知していない。

HTTP クライアント接続を開くコマンドをアプリケーションが発行し、適切な接続がプールにあるとき、CICS は最も新しくプールに入れられた接続を選択します。このプラクティスにより、前のピーク・レベルから接続使用量が減少した場合に、古い接続から有効期限が切れるようになります。プール接続が再使用されず、SOCKETCLOSE 属性で指定した制限時間に達すると、CICS はソケットを閉じて接続を除去します。

以下の状態の場合、CICS はプール内の休止接続も閉じます。

- Web サーバーが接続を閉じる。
- 接続のプールに関連付けられた URIMAP リソースが廃棄される。
- CICS 領域の MAXSOCKETS に達し、異なる接続で使用されるソケットが必要である。

URIMAP 定義のリソース統計 (DFHWBRDS 内) には、接続プーリングに関する以下の統計が含まれます。

- URIMAP リソースの SOCKETCLOSE 設定
- この URIMAP リソースのプール内のプール接続の現行数とピーク数
- CICS 領域が MAXSOCKETS 制限に達したために CICS がプールから再利用した休止接続の数

- プールに入れられている間に再使用されず有効期限が切れた休止接続の数

これらの統計は、DFHSTUP および DFH0STAT レポートで報告されます。

第 3 部 CICS からの Atom フィード

CICS は Web クライアントに Atom フィードをサービス提供することができます。 Atom フィードは、CICS リソースまたはアプリケーション・プログラムによって提供されるデータで構成されます。 CICS リソースまたはアプリケーション・プログラムを Atom フィードまたはコレクションとして公開する際、ユーザーはフィード・リーダーや Web マッシュアップ・アプリケーションなどの外部クライアント・アプリケーションから HTTP 要求を行うことによって、データの読み取りや更新を行うことができます。

CICS には、独自のデータを使って Atom フィードまたはコレクションをセットアップするために使用できるサンプルが付属しています。 Atom フィードに関する CICS サポートを試行し、Web クライアントとの相互作用を確認するには、サンプル Atom コレクションをセットアップして使用してください。その際、CICS TS for z/OS バージョン 4.1 Information Center (`https://publib.boulder.ibm.com/infocenter/cicsts/v4r2/index.jsp`) の Web 2.0 シナリオ『従業員情報を処理する Atom フィードの作成』に記されている指示に従ってください。

第 17 章 Atom フィードの概要

Web フィード (場合によっては単に「フィード」とも呼ばれます) は、コンテンツ提供者がインターネット上に公開している一連の関連項目です。Atom フィードは、Atom 配信フォーマットおよび Atom 出版プロトコルを使用する Web フィードです。

Atom は XML ベースのフォーマットで構成されていて、Atom フィードとその中の情報項目について、さらに Atom フィードを公開および編集するためのプロトコルについて記述しています。このフォーマットとプロトコルについては、以下の 2 つのインターネット協会と IETF (Internet Engineering Task Force) の Request for Comments documents (RFC と呼ばれる) で説明されています。

RFC 4287 (「*The Atom Syndication Format*」。http://www.ietf.org/rfc/rfc4287.txt から入手できます)

RFC 5023 (「*The Atom Publishing Protocol*」。http://www.ietf.org/rfc/rfc5023.txt から入手できます)

コンテンツ提供者は、RSS (Really Simple Syndication) と呼ばれる以前のフォーマットで、Web フィードをしばしば配信します。 CICS は Atom をサポートしていますが、RSS はサポートしていません。

Atom フィードを構成する情報の項目は、*Atom エントリー* と呼ばれています。コンテンツ提供者は Atom フィードを公開または「配信」します。そのためには、インターネット上の特定の URL を使用可能にして、新しい項目で更新します。 Web ページは Atom フィード内の項目を表示でき、Web ユーザーはフィード・リーダーまたは Web ブラウザーを使用してフィードから項目を取得できます。 Atom フィードはマッシュアップ の一部として使用できます。マッシュアップは、コンテンツを多数のデータ・ソースからマージする Web アプリケーションで、それによりユーザーにはそのデータが新しい仕方で表示され、より良く理解できます。マッシュアップでは、Atom フィードからのデータは、ウィジェット として処理されます。これは、Web ページで実行されるスクリプト・アプリケーションです。

Atom 公開プロトコルは、Atom フィード内の個々の Atom エントリーを保管するサーバーに対して HTTP 要求を行うことにより、それらの項目をユーザーが追加、削除、編集、または表示できるように指定します。GET 要求では表示用にエントリーを取得し、POST 要求では完全な新規エントリーを追加し、 PUT 要求では既存エントリーを編集し、DELETE 要求ではエントリーを削除することができます。サーバーは、要求された変更を適切な方法で処理し、変更の確認によってユーザーのクライアントに応答します。

Atom 文書

Atom 文書は、エントリー文書、フィード文書、コレクション、サービス文書、カテゴリー文書のいずれかのタイプになります。

Atom エントリー文書

Atom エントリー文書は、Atom フィード用の単一の情報項目 (エントリーと呼ばれます) が含まれる XML 文書です。

Atom エントリー文書は、複数の子要素が含まれる <atom:entry> 要素で構成されます。これらの子要素により、エントリーの内容や、エントリーに関するメタデータ (エントリーのタイトルや最初に公開された時刻など) が提供されます。

Atom エントリーの内容には、プレーン・テキスト、HTML、XHTML、または他の IANA (Internet Assigned Numbers Authority) メディア・タイプを使用できます。IANA メディア・タイプは、 http://www.iana.org/assignments/media-types/ でリストされています。 Atom エントリーには、ムービーやイメージなどのメディア・リソースへのリンクをその内容として含めることができます。この場合、そのエントリーはメディア・リンク・エントリーと呼ばれます。

Atom エントリー文書のメディア・タイプは、application/atom+xml です。

Atom フィード文書

Atom フィード文書は、Atom フィードに関するメタデータとそのフィードの 1 つ以上のエントリーを提供する XML 文書です。クライアントがフィードの情報を要求すると、サーバーはその要求を満たす適切な数の Atom エントリーが含まれるフィード文書を生成します。

Atom フィード文書は、複数の子要素が含まれる <atom:feed> 要素で構成されます。<atom:entry> 要素は最も重要な子要素ですが、通常、フィードのエントリーは別個の XML 文書として存在しているため、サーバーはフィード文書を提供する際にそれらの XML 文書を追加します。Atom フィード文書は、<atom:entry> 要素が含まれていない場合でも、受け入れられる文書となります。

その他の子要素には、フィードに関するメタデータ (フィードのタイトルやサブタイトル、または主要な作成者など) が含まれます。Atom フィード文書内のメタデータの一部の項目 (作成者の詳細や知的所有権に関する情報) は、該当のメタデータ項目の独自バージョンが含まれているエントリーを除いて、フィード内のすべてのエントリーに適用できます。

Atom フィード文書のメディア・タイプは、application/atom+xml です。

Atom コレクション

Atom コレクションは、編集可能な Atom エントリーの URL のリストが含まれる特殊な種類の Atom フィード文書です。 Atom コレクションの形式は通常の Atom フィード文書の形式と似ていますが、いくつかの特殊要素が追加されています。これは、Atom サービス文書でリスト表示されることによってコレクションとして識別されます。

Atom コレクションには、いくつかの特殊な <atom:link> 要素が含まれています。すべてのエントリーを返すのに複数のフィード文書が必要になるほどコレクションが大きい場合は、<atom:link rel="first">、<atom:link rel="last">、<atom:link rel="next">、および <atom:link rel="previous"> の各要素によってフィード文書間のナビゲーションが提供されます。エントリーには、編集要求を送信できる <atom:link rel="edit"> リンクも含まれていま

す。<app:edited> 要素は、各エントリーの最終編集時刻を示すためにコレクション内のエントリーに追加されます。

Atom エントリーがコレクションとして使用可能になると、クライアントは既存エントリーの編集や削除、およびコレクション用の新しいエントリーの作成を行うことができます。クライアントは、以下のようにサーバーに HTTP 要求を送信することでエントリーを操作できます。

GET 1 つの Atom エントリーまたは Atom エントリーのリストを取得します。 Atom エントリーのリストを取得するための GET 要求は、Atom サービス文書で指定されているコレクションの URL に送信されます。 1 つの Atom エントリーを取得するための GET 要求は、エントリーの <atom:link rel="edit"> リンクで指定されているコレクション内の個々の Atom エントリーの URL に送信されます。

POST 新しい Atom エントリーを作成します。POST 要求は、コレクションの URL に送信されます。

PUT クライアントが GET 要求を使用して取得した既存の Atom エントリーを編集します。PUT 要求は、コレクション内の個々の Atom エントリーの URL に送信されます。

DELETE
既存の Atom エントリーを削除します。DELETE 要求は、コレクション内の個々の Atom エントリーの URL に送信されます。

それぞれのケースで、サーバーから適切な HTTP 応答がクライアントに送信されます。サーバーは、クライアントがエントリー用に提供するメタデータを変更できるので、クライアントの POST または PUT 要求が成功すると、サーバーは新しいエントリーのコピーも応答のボディとして返します。

標準の Atom エントリーのほか、コレクションにはムービーやイメージなどのメディア・リソースを含めることもできます。メディア・リソースをサポートしているサーバーは、メディア・リンク・エントリーと呼ばれる特殊な Atom エントリーをコレクション内に作成して、それらのリソースに対するリンクを提供します。CICS では、メディア・リソースはサポートされていません。

Atom サービス文書

Atom サービス文書は、サーバーで使用可能なコレクションをリストする XML 文書です。

Atom サービス文書には、ルート要素 <app:service> があります (app: 接頭部は、Atom 出版プロトコル用の名前空間の接頭部です)。これには、複数の <app:collection> 要素を含むワークスペースを定義する 1 つ以上の <app:workspace> 要素が含まれています。ワークスペースはコレクションのグループ化にのみ使用され、ワークスペースに対してアクションを実行することはできません。

<app:collection> 要素では各コレクションの URL とタイトルがリストされ、コレクションが受け入れる入力のタイプやエントリー用に使用可能なカテゴリーが記述されることもあります。

Atom サービス文書のメディア・タイプは、application/atomsvc+xml です。

Atom カテゴリー文書

カテゴリー文書には、コレクション内のエントリーのカテゴリーのリストが含まれます。カテゴリーは、サービス文書で指定することもできます。同じカテゴリーを使用して複数の Atom フィードを定義する場合は、別々のカテゴリー文書を使用すると便利です。

Atom カテゴリー文書には、ルート要素 <app:categories> があります (app: 接頭部は、Atom 出版プロトコル用の名前空間の接頭部です)。
<app:categories> 要素には、コレクション内のエントリーについて許可されている <atom:category> 要素のリストが含まれます。カテゴリーのリストは、サーバーがその他のカテゴリーのエントリーを拒否できるように固定することができます。または、その他のカテゴリーを使用できるようにオープンにすることができます。

Atom サービス文書のカテゴリーのリストの代わりに別個の Atom カテゴリー文書が使用される場合、そのカテゴリー文書はサービス文書内でその URL によって参照されます。

Atom コレクションのメディア・タイプは、application/atomcat+xml です。

CICS における Atom フィードの処理

Atom 文書をサービス提供するには、CICS は適切な URL を組み込み、Atom エントリー用データを識別および取得し、さらに Atom 文書における Atom エントリーの配置を判別する必要があります。 Atom フィードまたはコレクションをセットアップする際、Atom 構成ファイルまたはサービス・ルーチンに、CICS がこうしたタスクを完了するために役立つ情報を提供しなければなりません。

関連概念

251 ページの『第 17 章 Atom フィードの概要』
Web フィード (場合によっては単に「フィード」とも呼ばれます) は、コンテンツ提供者がインターネット上に公開している一連の関連項目です。Atom フィードは、Atom 配信フォーマットおよび Atom 出版プロトコルを使用する Web フィードです。

273 ページの『第 18 章 CICS によって Atom フィードがサポートされる方法』
CICS では、Atom フィードをサポートするために、CICS Web サポートの HTTP サーバー機能と、 Atom 形式およびプロトコルをサポートするサーバーで必要なアクションを実行するいくつかの追加機能を使用します。ユーザーは、 Atom フィードのデータを提供するリソースを選択またはセットアップし、フィードを CICS に対して定義する必要があります。

CICS からの Atom フィードのデータ処理

Atom エントリーを含む Atom フィードまたはコレクション文書を生成するために、 CICS は Atom エントリーのデータをファイルまたは一時記憶域キューから直接、あるいは他のリソースからデータを取り出すサービス・ルーチンから取得します。

255 ページの図 14は、CICS が Atom 構成ファイルを使用してファイル内のレコードから関連データを識別して取り出す方法を示しています。

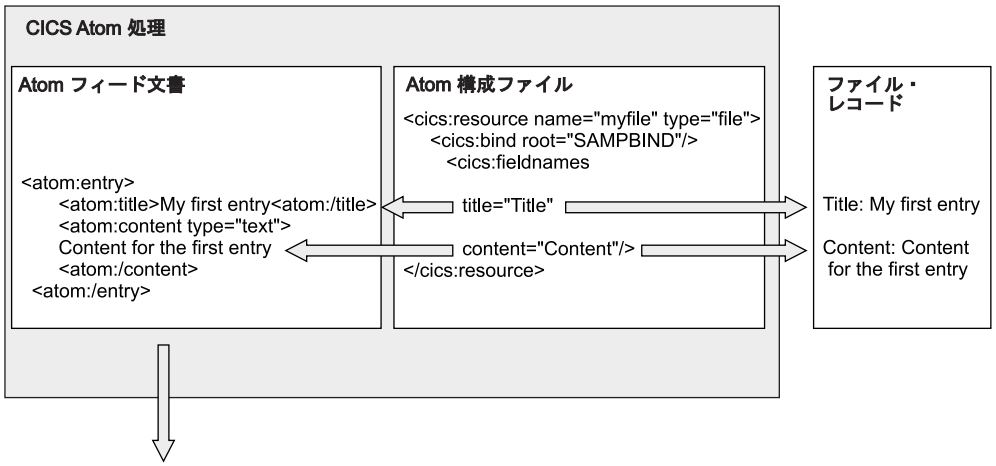


図 14. Atom エントリー・データをファイルから直接取り出す

- ファイル内のレコードには、Atom エントリーのデータを保持する「Title」および「Content」という名前のフィールドが含まれています。
- Atom 構成ファイルには、ファイルを識別する <cics:resource> 要素、ファイルの XML バインディングを参照する <cics:bind> 要素、および <cics:fieldnames> 要素が含まれます。 <cics:fieldnames> 要素のタイトル属性は、ファイル・レコードの「Title」フィールドを Atom エントリーのタイトルのデータを保持するフィールドとして識別します。 content 属性は、ファイル・レコードの「Content」フィールドを Atom エントリーの内容のデータを保持するフィールドとして識別します。
- CICS は Atom 構成ファイルおよび XML バインディングの情報を使用して、ファイル・レコード内の「Title」および「Content」フィールドからデータを見つけて取り出し、そのデータを使用して Atom フィード文書の Atom エントリーの <atom:title> および <atom:content> 要素に値を取り込みます。
- CICS が必要な数の Atom エントリーを生成するためにこの処理をファイルから一連のレコードに対して実行し終えると、CICS は Atom エントリーを含む Atom フィード文書を Web クライアントに送ります。

256 ページの図 15は、CICS がユーザー作成のサービス・ルーチン・プログラムを介してデータベース内のレコードからデータを取得する方法を示しています。

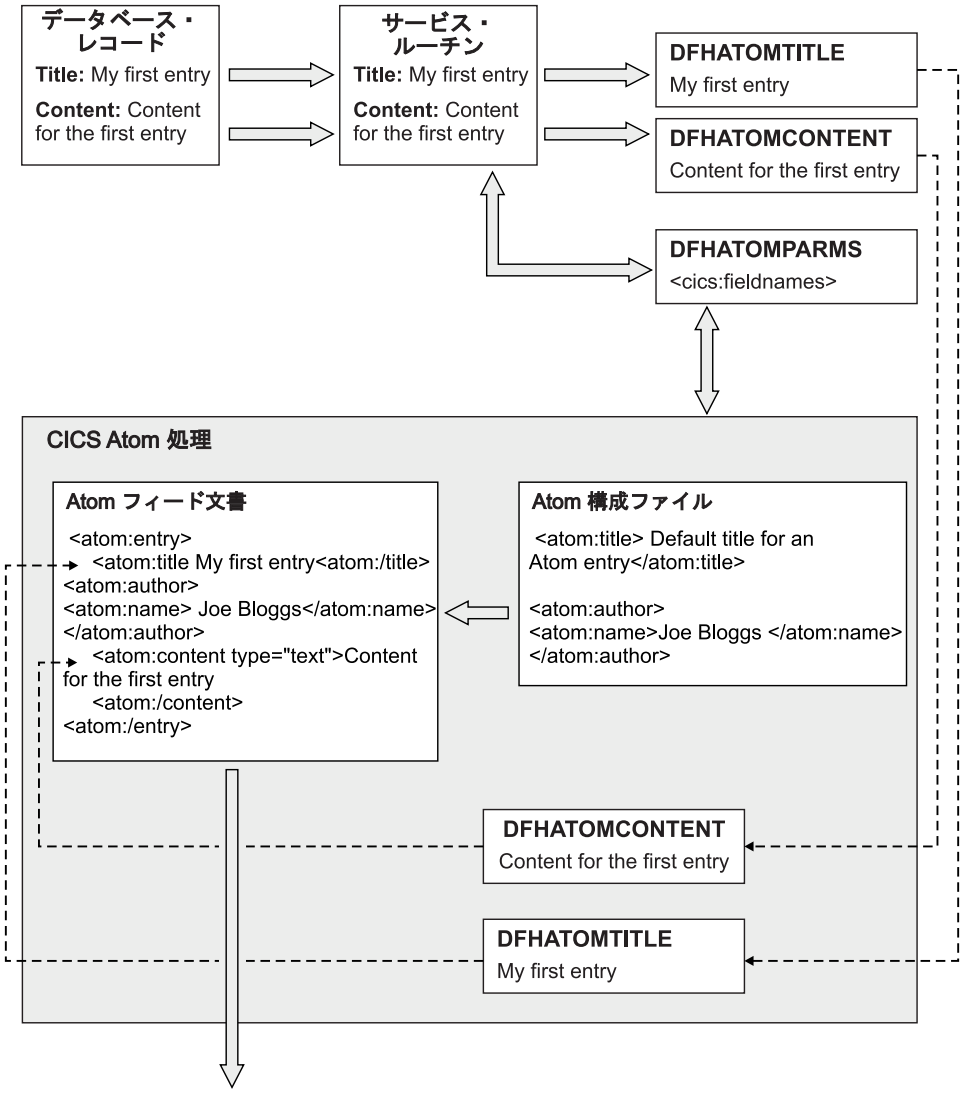


図 15. サービス・ルーチンを使用して Atom エントリー・データを提供する

- データベース内のレコードには、Atom エントリーのデータを保持する「Title」および「Content」という名前のフィールドが含まれています。
- サービス・ルーチンは、データベース内の「Title」および「Content」フィールドからデータを取り出します。サービス・ルーチンが XML バインディングのあるリソースを処理している場合、関連フィールドの名前を Atom 構成ファイルの <cics:fieldnames> 要素の情報から取得できます。 CICS はそれを、DFHATOMPARMS という名前のコンテナー内のパラメーターとしてサービス・ルーチンに渡します。 DFHATOMPARMS には、Web クライアントの要求に関する他の情報も含まれます。
- サービス・ルーチンは DFHATOMTITLE および DFHATOMCONTENT という名前のコンテナーを作成して、データを「Title」および「Content」フィールドからコンテナーに書き込みます。その後、コンテナーを CICS Atom 処理に戻します。

- Atom 構成ファイルには、Atom エントリーのデフォルト・タイトルを示す <atom:title> 要素、および作成者名 Joe Bloggs を示す <atom:name> 要素を含む <atom:author> 要素が含まれています。
- CICS は、サービス・ルーチンによって DFHATOMTITLE および DFHATOMCONTENT コンテナー内に提供されたタイトルおよび内容を使用して Atom エントリーを構成します。サービス・ルーチンが作成者名を提供していないので、CICS は Atom 構成ファイルからの作成者名を使用します。サービス・ルーチンがデータを提供しているので、CICS は Atom 構成ファイルからのデフォルト・タイトルを必要としません。
- CICS がサービス・ルーチンを複数回呼び出して、さまざまなデータベース・レコードからのデータを提供することにより必要な数の Atom エントリーを生成すると、 CICS は Atom エントリーを含む Atom フィード文書を Web クライアントに送ります。

CICS からの Atom フィードの URL

Atom フィード文書、コレクション、およびフィードまたはコレクション内の Atom エントリー文書には、Web クライアントが文書と対話するために使用できる URL (Uniform Resource Locators) が含まれています。各 URL は、Atom 文書の <atom:link> 要素で提供されます。 Atom 文書では複数の <atom:link> 要素を指定でき、この要素の rel 属性 (リンク関係と呼ばれる) でさまざまな URL の目的を指定できます。

CICS によってサービス提供される Atom フィード文書またはコレクションには、最大 4 つのタイプの以下の URL が含まれます。

- Atom フィードまたはコレクション全体を位置指定する URL。このフィードの URL は、<atom:feed> 要素の子要素である <atom:link rel="self" > 要素で提供されます。 Web クライアントは、この URL を使用して、複数のエントリーを含む Atom フィード文書を Atom フィードまたはコレクションから取得できます。
- フィードまたはコレクション内の各 Atom エントリーを位置指定する個別の URL。これらのエントリーの URL は、<atom:entry> 要素の子要素である <atom:link rel="self" > 要素で提供されます。 Web クライアントは、これらの URL を使用して、フィードまたはコレクションから単一の Atom エントリーを取得できます。
- Web クライアントがコレクションを編集するための要求を行うのに使用できる、編集 URL。これらの URL は、<atom:link rel="edit"> 要素で提供されます。CICS は、1 つの編集 URL をコレクションの <atom:feed> 要素の子要素として、コレクション全体に提供し、個別の編集 URL を <atom:entry> 要素の子要素として、コレクションの各 Atom エントリーに提供します。さらに CICS は、コレクションと、コレクション内の Atom エントリーに <atom:link rel="self" > URL を提供します。
- Web クライアントが Atom フィードまたはコレクションの Atom エントリーの部分リストを取得するために使用できる、ナビゲーション URL。これらの URL は、<atom:link> 要素において、rel 属性 "first"、"previous"、"next"、および "last" を指定して提供されます。これらの URL により、Web クライアントは Atom フィードまたはコレクションの全体を、すべての Atom エントリーを一度に取得する必要なく探索できます。 CICS は、Web クライアントが Atom エントリーの次のウィンドウをフィードから取得するために使用できる URL を、Atom フィ

ード文書内の <atom:link rel="next"> に提供します。コレクションからのエントリーの部分リストを含む Atom 文書では、CICS は rel 属性 "first"、"previous"、"next"、および "last" を指定した <atom:link> 要素を追加して、コレクションからの Atom エントリーの他の部分リストへのナビゲーションを提供します。

Atom フィードの場合、フィード全体の URL は通常、インターネットまたは会社のイントラネットで公表されます。 Web クライアントがフィード URL を使用して Atom フィード文書を取得するとき、Atom フィード文書の Atom エントリーにはそれら自身の個別の URL が含まれており、Web クライアントはそれらを使用して単一の Atom エントリーを取得できます。

コレクション (編集できる Atom エントリーが含まれている) の場合、サーバーから使用可能なサービス文書が、サーバー上で各コレクションの編集 URL を提供します。 Web クライアントは、それらの URL のいずれか 1 つを使用してコレクション内の Atom エントリーを表示することができ、さらなるエントリーをそれに追加する要求を行うことができます。 Web クライアントは編集 URL を個別の Atom エントリーに使用して、エントリーを更新または削除する要求を行うことができます。

Atom 配信形式により、Internationalized Resource Identifiers (IRI) の使用が可能になります。これは、英語以外の各国語に適した Unicode 文字およびフォーマットを許可します。通常の URL の代わりに、CICS からの Atom フィードのリソース・ロケーターとして Unicode 文字を含む IRI を使用することもできます。 Atom 配信形式および Atom 出版プロトコルの RFC では、Atom フィードおよび Atom エントリーのリソース・ロケーターは、IRI と呼ばれます。 262 ページの『Internationalized Resource Identifiers (IRI)』には、IRI について、およびそれを Atom フィードに使用する方法について説明されています。

Atom URL を指定および解決する方法

CICS では、Atom 構成ファイルの <atom:feed> および <atom:entry> 要素の <atom:link> 子要素を使用して、Atom フィードまたはコレクションの全体の URL を指定し、個別の Atom エントリーの標準 URL も指定します。 Atom 構成ファイルでは、これらの子要素に <atom:link rel="self"> を常に指定し、CICS が Atom 文書を送信するときに、CICS は <atom:link rel="edit" > 要素の同じリンクをコレクションおよびコレクション内の Atom エントリーに追加します。 Atom 構成ファイルから URL のスキームとホスト構成要素を省略して、パス構成要素のみを指定することもできます。 CICS は、フィードまたはエントリー文書をクライアントに返す際に、スキームとホスト構成要素を URL に追加します。

Atom 構成ファイルで、rel 属性 "first"、"previous"、"next"、および "last" を指定して、ナビゲーション URL の <atom:link> 要素のいずれかを指定する必要はありません。 CICS はこれらのリンクをユーザーのために作成します。

フィード全体に指定される URL、および個別のエントリーの標準 URL として指定される URL には、同じ方法で開始されるパス構成要素がなければなりません。 Web クライアントの Atom フィードへの要求を処理するために CICS が使用する URIMAP リソース定義にパス構成要素のこの共通部分を指定し、アスタリスクを使用してパスの残りの部分がパスのマッチングに使用されることを示します。パス構

成要素の共通部分は、CICS が Atom フィードまたはコレクションを識別するものなので、これは、指定されたホスト名を使用して提供するすべての Atom フィードおよびコレクションの中で、この Atom フィードまたはコレクションに固有のものでなければなりません。

Web クライアントが、パス構成要素のこの共通部分を含む URL を使用して要求を行うとき、CICS は一致する URIMAP リソース定義を検出し、他の多くのリソースを使用して要求 URL を Atom フィードのデータにマップします。 図 16には、以下のようにフィード URL のこのプロセスが示されています。

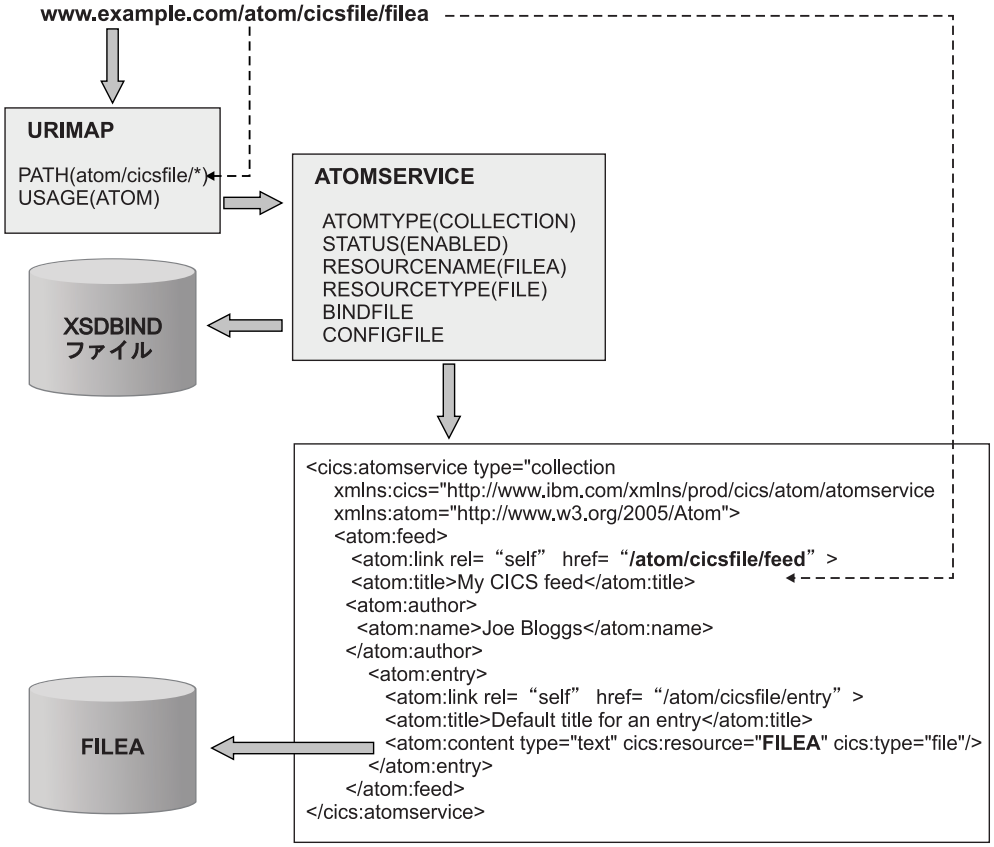


図 16. Atom フィードの要求 URL

- Web クライアントからの着信要求を処理するには、フィードおよびエントリーの URL に共通なパス構成要素の部分を指定する URIMAP リソース定義を作成します。この例では、パス構成要素の共通部分は `atom/cicsfile/` です。 Web クライアントが、Atom フィードまたはコレクションに定義した URL、または Atom エントリーに定義した URL を使用して要求を行うとき、CICS はパス構成要素の共通部分が一致する URIMAP リソースを検出します。この例では、Web クライアントはフィード URL `www.example.com/atom/cicsfile/filea` を使用して Atom フィードを要求します。
- URIMAP リソースは ATOMSERVICE リソースを指定し、それは Atom 構成ファイル、XML バインディング (XSDBIND ファイル)、および Atom フィードを

提供する CICS リソースを指定します。この例の ATOMSERVICE リソースは、FILEA ファイルを Atom エントリーのデータを保持するリソースとして指定します。

- CICS は ATOMSERVICE リソースを使用して Atom 構成ファイルの位置を指定し、Web クライアントによって使用されるインバウンド URL のパス構成要素を、Atom 構成ファイルのすべての <atom:link> 要素で指定されている URL と比較します。 CICS が一致するパス構成要素を持つ <atom:link> 要素で URL を検出すると、適切なアクションを実行し、Atom フィードまたはエントリー文書を返すか、編集要求を実装します。この例では、Web クライアントによって使用される要求 URL が Atom 構成ファイルの Atom フィードに指定された URL と一致するので、CICS は Atom フィード文書を返す必要があります。
- ATOMSERVICE リソースのように、Atom 構成ファイルは、FILEA ファイルを Atom エントリーのデータを保持するリソースとして指定します。 254 ページの『CICS からの Atom フィードのデータ処理』で説明されているように、CICS は、Atom エントリーのデータを含むファイルまたは一時記憶域キューからデータを抜き出すか、要求をサービス・ルーチンに受け渡すために、直接操作する場合があります。

259 ページの図 16 では、Atom フィード全体の URL のパスは、Atom 構成ファイルの <atom:feed> 要素の <atom:link> 子要素で指定されているとおり、`/atom/cicsfile/filea` です。 Atom 構成ファイルの <atom:entry> 要素には、<atom:link> 子要素もあります。これには、パス `/atom/cicsfile/entry` が含まれています。これは、Atom エントリーの標準パスです。 Atom エントリーのこの標準パスは、パス構成要素の共通部分である `atom/cicsfile/` で始まります。 Atom エントリーの標準パスの残りの部分は、<atom:feed> 要素の <atom:link> 子要素で指定されている Atom フィードのパスとは異なっている必要があります。 CICS は Atom 構成ファイル内のパス・マッチングにパスのこの部分を使用して、Atom フィード文書または Atom エントリー文書が必要とされているかどうかを判別します。

261 ページの図 17 には、CICS が単一の Atom エントリー用の Web クライアントからの要求を処理する方法と、適切な Atom エントリーを識別する方法が示されています。

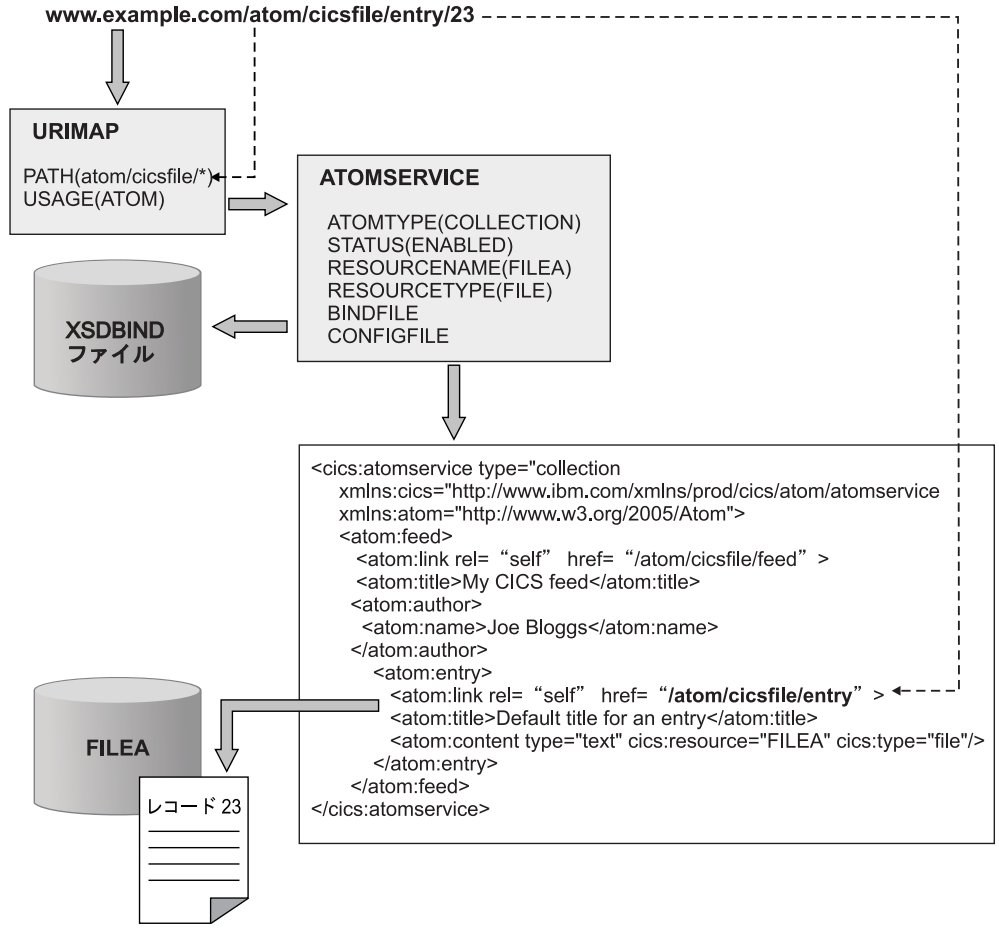


図 17. Atom エントリーの要求 URL

- この Web クライアントは、URL `www.example.com/atom/cicsfile/entry/23` を使用して 1 つの Atom エントリーを要求します。 Web クライアントは、Atom エントリーの <atom:link> 子要素からこの URL を取得しました。これは、Atom フィード文書の一部として Web クライアントがもともと受信したものです。
- Atom エントリー URL には Atom フィードのパス構成要素の共通部分である `atom/cicsfile/` が含まれているので、フィード URL と同じ URIMAP リソースおよび ATOMSERVICE リソースによって処理されます。この例では、Web クライアントによって使用される要求 URL が Atom 構成ファイルの Atom エントリーに指定された標準パスと一致するので、CICS は Atom エントリー文書を返す必要があります。
- CICS は、標準パスに続く要求 URL の残りの部分を調べて、Web クライアントに戻す Atom エントリーを識別します。この例では、要求 URL には番号「23」が含まれています。これは、エントリーのセレクター値です。セレクター値は ID (通常は数値) であり、Atom エントリーのデータを含むファイル、一時記憶域キュー、または他のリソースのレコードを識別します。この例では、Atom エントリーに選択されたセレクター値がこのレコード番号でした。セレクター値を選択する場合は、セレクター値が追加されたときに、Atom フィード全体の URL と Atom エントリーの標準パスが、引き続き異なっていることを確認してくださ

い。 264 ページの『Atom エントリーのセレクター値』には、セレクター値の選択および使用方法の詳細が説明されています。

さらに CICS は、rel 属性 "first"、"previous"、"next"、および "last" を指定した <atom:link> 要素においてセレクター値を使用して、Atom フィードまたはコレクションからのエントリーの部分リストへのナビゲーション・リンクを作成します。CICS は Atom フィードまたはコレクションの全体に指定したパスをとり、部分リストの始めに表示される Atom エントリーのセレクター値を追加することにより、これらのナビゲーション・リンクを作成します。 CICS はこの情報を Atom フィードまたはコレクションに指定されたウィンドウ設定とともに使用して、部分リストを Web クライアントに返します。ここで使用されている Atom フィードの例で、セレクター値「9」を指定した Atom エントリーで始まるエントリーの部分リストの場合、CICS はリンク `www.example.com/atom/cicsfile/filea/9` を作成します。

これらの例は、デフォルト形式 ("segment" 形式と呼ばれる) で URL に追加されているセレクター値を示しています。ここでは、セレクター値はパス構成要素の最終セグメントとして置かれます。代わりに、CA8K SupportPac を使用して開発されたアプリケーションと互換性のある URL スタイル (Atom エントリーのセレクター値が照会ストリングに置かれます) を選択することもできます。 Atom 構成ファイルで <cics:selector> 要素を使用して、この代わりの "query" 形式を指定できます。<cics:selector style="query"/> が、ここで使用されている Atom フィードの例に指定されている場合、CICS は形式 `www.example.com/atom/cicsfile/entry?s=23` で個別の Atom エントリーのリンクを作成します。同じ形式が、ナビゲーション・リンクに使用されます。

Internationalized Resource Identifiers (IRI)

Internationalized Resource Identifier (IRI) は、英語以外の各国語に適した文字およびフォーマットを使用できるようにするための、インターネット用のリソース ID の形式です。 IRI は、要求および応答に関連したアプリケーションによってサポートされる場合、URI または URL の代わりに使用できます。

IRI については、http://www.ietf.org/rfc/rfc3987.txt で入手可能な RFC 3987、「*Internationalized Resource Identifiers (IRIs)*」で説明されています。 CICS は、HTTP サーバーとしての CICS に対するインバウンド Web クライアント要求用の URIMAP リソース内、および Atom フィード文書内での IRI の使用をサポートします。

ホスト名

ドメイン・ネーム・サーバーの要件を満たすために、Web クライアントは IRI のホスト名を Punycode と呼ばれるフォーマットに変換します。 Punycode については、http://www.ietf.org/rfc/rfc3492.txt で入手可能な RFC 3492、「*Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)*」で説明されています。このアルゴリズムは、ホスト名をエンコードして、英数字、ハイフン、およびピリオドだけから構成されるストリングにします。

IRI を CICS によってサービスされる Web リソースまたは Atom フィードへのリンクとして使用する場合、CICS に対する Web クライアントの要求を定義する URIMAP リソース定義内に、ホスト名を Punycode で指定する必要があります。

CICS は、この変換を実行するためのツールを提供していませんが、Unicode から Punycode への変換をサポートするフリー・アプリケーションがインターネット上で使用可能です。 URIMAP リソースがどのホスト名とも一致するようにホスト名の代わりに単一のアスタリスクを使用する場合、Punycode を使用する必要はありません。

パス・コンポーネント

Web クライアントは IRI のパス・コンポーネントを Punycode に変換しませんが、パス内の Unicode 文字をエスケープ、つまりパーセント・エンコードします。

IRI を CICS によってサービス提供される Web リソースのために使用する場合、URIMAP リソース定義で、指定するパスに含まれる Unicode 文字をパーセント・エンコードする必要があります。 Unicode 文字をパーセント・エンコード表記に変換できるアプリケーションがない場合、このタスクを実行するためのフリー・アプリケーションがインターネット上で使用可能です。 40 ページの『CICS Web サポートの URL』でリストされている URL 長の制限は Atom フィードの URL にも適用されること、つまり URL のパス・コンポーネントで URIMAP リソース定義に指定する部分は 255 文字以下でなければならないことに注意してください。この文脈で文字とは、元の Unicode 文字ではなく、単一の ASCII 文字のことです。例えば、パーセント・エンコード表記のキリル文字 %D0%B4 は、255 文字制限の中の 6 文字としてカウントされます。

CICS が URIMAP リソース定義をインストールすると、 CICS はパスを URI 用に推奨される正規の形式で保管していくつかの文字をアンエスケープしますが、URIMAP リソースを表示するときには入力したままの状態でパスが表示されます。

IRI を Atom フィードまたはエントリー文書のリンクとして使用する場合、 Atom 構成ファイル内および URIMAP リソース定義内に IRI を指定します。 Atom 構成ファイル内にある IRI の Unicode 文字をパーセント・エンコードする必要があります。

CICS が IRI を含む Atom 文書を発行すると、 CICS はパーセント・エンコード文字を XML 文字参照に変換して、XML が有効になるようにします。結果として生じるリンクを Web クライアント要求で使用するには、 XML 文字参照をパーセント・エンコード文字に変換して戻す必要があります。

この Unicode リソースの例に含まれているパスは、Unicode 文字を使用して Atom フィード用の IRI の先頭を指定し、末尾のアスタリスクによって IRI の残り部分にパス・マッチングが使用されることを示しています。

```
  Urimap         : ALEXANDR
  Group          : IRIMAPS
  DEScription    :
  STatus         : Enabled            Enabled | Disabled
  USAge          : Atom               Server | Client | Pipeline | Atom
 UNIVERSAL RESOURCE IDENTIFIER
  SCheme         : HTTP               HTTP | HTTPS
  POrt           : No                 No | 1-65535
  HOST           : *
  (Mixed Case)   :
  PAth           : %D0%90%D0%BB%D0%B5%D0%BA%D1%81%D0%B0%D0%BD%D0%B4%D1%80%D0%
  (Mixed Case)   : A1%D0%BE%D0%BB%D0%B6%D0%B5%D0%BD%D0%B8%D1%86%D1%8B%D0%BD*
```

この Atom エントリーの例には、URIMAP リソースの例に表記された Unicode 文字と等しい XML 参照を使用する IRI が含まれます。

```
<entry>
<link rel="self" href="http://example.com:5050/&#x0410;&#x043B;&#x0435;
&#x043A;&#x0441;&#x0430;&#x043D;&#x0434;&#x0440;&#x0421;&#x043E;&#x043B;&#x0436;
&#x0435;&#x043D;&#x0438;&#x0446;&#x044B;&#x043D;/000100"/>
<id>tag:example.com,2009-02-13:file:FILEA:000100</id>
<title>FILEA item 000100</title>
<rights>Copyright (c) 2009, Joe Bloggs</rights>
<published>2008-11-06T12:35:00.000Z</published>
<author>
 <name>Joe Bloggs</name>
 <email>JBloggs@example.com</email>
</author>
<app:edited>2009-03-11T14:42:38+00:00</app:edited>
<updated>2009-03-11T14:42:38+00:00</updated>

<content type="text/xml">
  <DFH0CFIL xmlns="http://www.ibm.com/xmlns/prod/cics/atom/filea">
  <filerec>
  <numb>000100</numb><name>S. D. BORMAN</name><amount>$0100.11</amount>
  </filerec>
  </DFH0CFIL>
</content>

</entry>
```

Atom エントリーのセレクター値

Atom エントリーのセレクター値は、CICS またはサービス・ルーチンが使用できる ID であり、Atom エントリーのデータを含むファイル、一時記憶域キュー、またはその他リソースの中で、レコードを見つけるためのものです。適切なセレクター値は、Atom エントリーのデータを保持するリソースの特定のレコードに常に適用される固有の ID (項目番号や固有キーなど) です。

CICS が Web クライアントに対する応答として Atom 文書を発行する際、CICS は個々の Atom エントリーに対してセレクター値を使用して Atom エントリーへのリンクを直接構成するか、またはエントリーについて生成された Atom ID の一部として構成します。 Web クライアントはそのリンクを使用して、単一の Atom エントリーに対して要求を行うことができます。各リンクのセレクター値は、Atom エントリーのデータを含むリソース内の正しいレコードを識別します。 257 ページの『CICS からの Atom フィードの URL』では、セレクター値がどのようにリンクに付加されるかが説明されています。

CICS がファイルまたは一時記憶域キューから直接 Atom エントリーを配信する際、CICS はリソースのタイプに応じて適切なセレクター値を識別します。一時記憶域キューの場合、セレクター値は、一時記憶域キュー内のレコードを識別する番号です。この番号については、10 進数が CICS で想定されます。ファイルの場合、セレクター値はファイルのキーです。これは固有キーでなければなりません。 CICS は、各ファイル・タイプのセレクター値の形式が以下のとおりであると想定します。

- RRDS ファイルと VRRDS ファイルの場合、10 進数。
- ESDS ファイルと拡張 ESDS ファイルの場合、2 進数。
- その他のタイプの VSAM ファイルの場合、文字ストリング。

ファイルのキーが CICS が想定する形式ではない場合、Atom 構成ファイルの <cics:selector> 要素で正しい形式を指定できます。

サービス・ルーチンが Atom エントリーにデータを返す際に、セレクター値の内容を選択することができます。セレクター値には、ユーザー・プログラムが Atom エントリーにデータを提供するためにリソース内の正しいレコードを検索するために使用できる任意の値を設定できます。例えば、リソースがデータベースである場合は、レコードのキーを提供する固有 ID を使用できます。キーが文字ストリングでない場合には、使用している Atom 構成ファイルの <cics:selector> 要素に 16 進数のセレクター値を指定する必要があります。

サービス・ルーチンがフィードまたはコレクションから Atom エントリーを返す際に、DFHATOMPARMS コンテナーの **ATMP_NEXTSEL** パラメーターを使用して、フィードで使用可能にされている次の Atom エントリーのセレクター値を返す必要があります。 Web クライアントが複数のエントリーを要求した場合、CICS はユーザー・プログラムがリソースのレコードとして保持されている次の Atom エントリーを識別して返せるようにするため、このセレクター値を使用して再びユーザー・プログラムにリンクします。このプロセスは、CICS にフィード用の十分なエントリーを取得するまで、またはリソースの使用可能な Atom エントリーがなくなったことを示すヌル値をユーザー・プログラムが返すまで続行されます。

サービス・ルーチンがコレクションから Atom エントリーを返す際には、DFHATOMPARMS コンテナーの **ATMP_PREVSEL**、**ATMP_FIRSTSEL**、および **ATMP_LASTSEL** パラメーターを使用して、コレクション内の前、最初、および最後の Atom エントリーのセレクター値を返す必要があります。 CICS はこれらの値を使用することにより、コレクション内のエントリーに関する他の部分リストのリンクが入る <atom:link> 要素を構成します。フィードから Atom エントリーのその他のウィンドウを取得するために Atom エントリーに関するそれらの値が Web クライアントにとって有効であると思われる場合には値をフィードから返すこともできますが、フィードではそれらの値は不要です。 **ATMP_PREVSEL** を使用してリンクを生成する処理には応答時間が余計にかかるので、この値は、この形式のナビゲーションを使用するように Web クライアントがセットアップされている場合に限りフィードに対して指定してください。

フィードまたはコレクション内の最初、前、および最後の Atom エントリーの ID は、Atom エントリーを返す順序によって決まります。 『Atom エントリーの順序』では、CICS が Atom エントリーを戻す順序を判別する方法を解説し、サービス・ルーチンが Atom エントリーを戻すことのできる順序について、提案が記されています。

Atom エントリーの順序

CICS、またはご使用のサービス・ルーチンは、Atom フィード文書内で複数の Atom エントリーが配置される順序を判別する必要があります。

http://www.ietf.org/rfc/rfc5023.txt から入手できる RFC 5023「*The Atom Publishing Protocol*」では、Atom コレクション内のエントリーは、エントリー内の <app:edited> 要素によって示されているように、編集された順序で Web クライアントに返されるべきであることが述べられています。直近に編集された Atom エントリーが最初に戻され、Atom フィード文書で表示される最初の Atom エントリーと

なるようにすべきです。次に新しく編集された Atom エントリーが 2 番目に戻され、編集されたエントリーのうち最も古いものが最後に戻されるようにします。RFC 5023 において、この機能は Atom エントリーの完全なリスト (1 つのフィード文書においてコレクション全体が戻されます) については推奨される要件となっていますが、Atom エントリーの部分リストについては必須の要件となっています。RFC 5023 および RFC 4287 には、コレクションとして定義されていない Atom フィード内の Atom エントリーの順序に関する要件はないので、Atom フィード内のエントリーに関してサーバーは、整合性があり論理的な任意の順序を選択できます。

パフォーマンス上の理由から、CICS では、最後に編集された時刻の順で自動的にコレクション内に Atom エントリーが返されることはありません。 CICS では、許容される応答時間を維持するとともに部分リストの有用な機能を提供し続けるため、この要件から外れています。Atom フィードおよびコレクションの両方に関して、CICS が Atom 文書を作成するためにリソースから直接データを取り出すと、CICS は判別可能な限り、Atom エントリーがリソース内のレコードとして書き込まれた時刻によって順序付けしてこれらのエントリーを戻します。直近に書き込まれた Atom エントリーが最初に戻され、次に新しく書き込まれた Atom エントリーが 2 番目に戻るといった具合です。 CICS は、以下のようにして書き込まれた順序を判別します。

- 一時記憶域キューの場合、レコード番号が最大の Atom エントリーが最初に戻されます。
- ESDS および拡張 ESDS ファイルの場合、最大の RBA (相対バイト・アドレス) または XRBA (拡張相対バイト・アドレス) を持つ Atom エントリーが最初に戻ります。
- RRDS および VRRDS ファイルの場合、最大の RRN (相対レコード番号) を持つ Atom エントリーが最初に戻されます。
- KSDS および AIX ファイル (書き込み順序という概念がない) の場合、Atom エントリーはレコード・キーの順序で戻され、**最小**のレコード・キーを持つ Atom エントリーが最初に戻されます。

サービス・ルーチンを使用して Atom エントリーにデータを提供する場合、Atom エントリーを戻す順序を選択できます。 RFC 5023 に準拠して、Atom エントリーが編集された時刻に従ってコレクションでそれらのエントリーを戻す場合には、サービス・ルーチンを使用して、ファイル内に格納されている Atom エントリーに関してこれを実行できます。編集された順序で Atom エントリーを戻すには、以下のアクションを実行します。

1. ご使用のファイルに、エントリーが最後に編集された時刻を示すタイム・スタンプ (ABSTIME 形式) が含まれるレコードが入ったフィールドを組み込みます。Atom エントリー内の <app:edited> 要素としてこの情報を出力できます。
2. タイム・スタンプが含まれるフィールドを、ファイルの代替索引として定義します。
3. ご使用のサービス・ルーチンでその代替索引を使用して、Atom エントリーのデータが含まれるレコードを見つけ、直近のタイム・スタンプによって示されている最も新しく編集されたエントリーが含まれるレコードを最初に戻すようにします。

またこの方法は、Atom エントリーが最初に書き込まれたときではなく更新されたときに基づいて、フィード内の Atom エントリーを戻す場合にも使用できます。Atom エントリーのデータを保持するファイルに適切なタイム・スタンプを保管できない場合、またはその情報を使用してエントリーを順序付けすることで許容できないほどの応答時間がかかる場合には、一貫性があり論理的な任意の順序、例えば CICS がリソースからデータを直接取り出す際に使用した順序などで Atom エントリーを返します。

サービス・ルーチンが DFHATOMPARMS コンテナーの **ATMP_PREVSEL**、**ATMP_NEXTSEL**、**ATMP_FIRSTSEL**、および **ATMP_LASTSEL** の各パラメーターに提供する値は、Atom エントリーを戻すために選択した順序によって異なります。タイム・スタンプに示されている編集または更新された時刻に基づいて Atom エントリーを戻す場合、これらのパラメーターの値は以下のようになります。

- 前の Atom エントリーは、現在のエントリーが編集された後に編集された Atom エントリーです。
- 次の Atom エントリーは、現在のエントリーが編集される直前に編集された Atom エントリーです。
- 最初の Atom エントリーは、直近に編集された Atom エントリーです。
- 最後の Atom エントリーは、編集時刻が最も古い Atom エントリーです。

Atom エントリーを最初に書き込まれた時刻に従って戻す場合、これらのパラメーターの値は以下のようになります。

- 前の Atom エントリーは、現在のエントリーが書き込まれた後に書き込まれた Atom エントリーです。
- 次の Atom エントリーは、現在のエントリーが書き込まれる直前に書き込まれた Atom エントリーです。
- 最初の Atom エントリーは、直近に書き込まれた Atom エントリーです。
- 最後の Atom エントリーは、書き込み時刻が最も古い Atom エントリーです。

Atom エントリーの日時スタンプ

Atom エントリーのメタデータには、Atom エントリーが最初に公開された日時、最後に更新された日時、および最後に編集された日時を示すスタンプを組み込むことができます。

Atom 配信形式および Atom 出版プロトコルは、これらの日時スタンプを次のように定義します。

<atom:published>
Atom エントリーが最初に作成された、または最初に使用可能になった日時。例えば、Atom フィードがデータベース内のレコードを使用して Atom エントリーにデータを提供する場合、この日時はデータが含まれているレコードがデータベースに追加された時点となります。

<atom:updated>
Atom エントリーに対して最後に重要な変更が加えられたと見なされる日時。例えば、データベースのレコード内のフィールドの値が変更された場合にこの日時スタンプを記録することもできます。

<app:edited>

Atom エントリーが最後に編集された日時。この日時スタンプはコレクションの一部である Atom エントリーにのみ適用され、そのようなエントリーでは必須 (SHOULD 要件) となっています。

Atom エントリーのデータが含まれるように新しいリソースをセットアップする場合は、リソースのレコード内にフィールドを含めて日時スタンプを保持することができます。サービス・ルーチンは、DFHATOMPARMS コンテナーの **ATMP_PUBLISHED**、**ATMP_UPDATED**、および **ATMP_EDITED** パラメーターを上書きすることにより、このデータを返すことができます。 DFHATOMPARMS コンテナーのリソース・ハンドリング・パラメーターを使用している場合、 **ATMP_PUBLISHED_FLD**、**ATMP_UPDATED_FLD**、および **ATMP_EDITED_FLD** パラメーターには、リソース内のレコードの関連フィールドの名前と長さが含まれています。

リソース内のレコードにメタデータを保持するためのフィールドが含まれていない場合、CICS はこれらすべての要素に対して現行の日時をデフォルトのタイム・スタンプとして提供します。この場合、サービス・ルーチンは、DHFATOMPARMS コンテナー内の関連パラメーターに対してスペースを返します。 <atom:published> 要素については、Atom 構成ファイルのプロトタイプ Atom エントリーに代替デフォルト・タイム・スタンプを指定することができます。 Atom 構成ファイルのプロトタイプ Atom エントリーの <atom:updated> および <app:edited> 要素については、代替デフォルト値を指定できません。

これらの要素に使用する日時スタンプは、RFC 3339 形式 (XML dateTime データ型としても知られる) で指定する必要があります。 RFC 3339 (*Date and Time on the Internet: Timestamps*) は、ISO 8601 標準に基づく、UTC (協定世界時) による日時スタンプの形式仕様です。この仕様は http://www.ietf.org/rfc/rfc3339.txt で参照できます。

EXEC CICS ASKTIME コマンドを使用した後、EXEC CICS FORMATTIME コマンドを使用して、RFC 3339 形式の日時スタンプを生成します。または、サービス・ルーチンが TRANSFORM DATATOXML コマンドを使用できる場合は、リソース・レコードに保持されている CICS ABSTIME 値をこの形式の日時スタンプに変換することができます。

リソース内のレコードに Web クライアント (POST 要求) によって提供される新しい Atom エントリーのデータを取り込んでいる場合、または Web クライアントの要求 (PUT 要求) に応じてリソース内のレコードのフィールドを編集している場合には、Web クライアントが <atom:updated>、<atom:published>、または <app:edited> 要素で日時スタンプを提供することもあります。 <atom:updated> および <app:edited> 要素の場合は、正確かつ有効なものとするために、その日時スタンプは無視して新しい日時スタンプを生成することをお勧めします。特に PUT 要求の場合、日時スタンプは、変更されずに返されて、リソース内の既存のレコードの日時スタンプと全く同じになることもあります。

関連資料

287 ページの『DFHATOMPARMS コンテナー』
DFHATOMPARMS は、Atom フィードのデータを提供するサービス・ルーチンと通信するために CICS が使用するパラメーターを含む DATATYPE(CHAR) のコンテナーです。

Atom エントリーの Atom ID

各 Atom エントリーには固有の Atom ID が付与されていますが、この ID は Atom エントリーの存続期間中は同じままでなければなりません。

Atom エントリーの Atom ID は、<atom:id> 要素で指定します。この ID は有効な Internationalized Resource Identifier (IRI) の形式でなければなりませんが、リソースの実際の場所と関連していなくても構いません。

タグ URI

CICS は、Atom フィードを提供する際に、Atom 構成ファイルの <cics:authority> 要素で指定されている情報を使用し、タグ URI 形式で各 Atom エントリーの固有 Atom ID を生成できます。タグ URI スキームについては、RFC 4151 (「*The 'tag' URI Scheme*」) で説明されています。

Atom エントリーの Atom ID としてタグ URI を生成するため、CICS は以下の項目を順番に使用します。

1. 「tag」のスキーム
2. Atom 構成ファイルの <cics:authority> 要素で指定した権限名と日付
3. Atom 構成ファイルの <cics:resource> 要素で指定されたリソース・タイプおよびリソース名の特定の構成と、個々の Atom エントリーのセレクター値

権限名と日付はコンマで区切られ、その他の要素はコロンで区切られます。以下に、CICS によって生成されるタグ URI の例を示します。

```
tag:example.com,2009-01-08:tsqueue:WB20TSQ:23
```

タグ URI の権限名は、ユーザーまたはユーザーの会社に登録済みのドメイン名または E メール・アドレスです。また日付は、ユーザーまたはユーザーの会社が権限名の所有を開始した日付です。 327 ページの『<cics:authority> 要素』には、権限名と日付に関する要件が説明されています。

CICS によってファイルまたは一時記憶域キューからデータが直接取得される Atom フィードについては、リソース・タイプおよびリソース名は、そのファイルまたは一時記憶域キューのものとなります。ユーザー作成サービス・ルーチンによってデータが提供される Atom フィードについては、リソース・タイプおよびリソース名は、そのサービス・ルーチンのものとなります。

CICS が Atom ID として生成するタグ URI には、以下の特性があります。

- ファイル、一時記憶域キュー、またはサービス・ルーチンの名前を変更したり、Atom 構成ファイル内の関連情報を変更したり、あるいは Atom エントリーを別のリソースに移動したりしない限り、Atom ID は各 Atom エントリーの存続期間中は変わりません。

- 同一のリソースが別の CICS 領域から同じようにサービス提供される場合、Atom ID は変わりません。
- ファイル、一時記憶域キュー、またはサービス・ルーチンの名前を変更すると、Atom ID が変更されます。リソースの名前を変更した場合、Atom 形式に準拠させるには、そのリソースを同じ Atom フィード (URL が同じ) として提供し続けることはできません。これは、リソースの Atom ID が異なるためです。
- Atom ID は、CICS 領域内で固有ですが、異なる CICS 領域間で固有であるとは限りません。名前とタイプは同じであるが異なる CICS 領域にあるリソースから Atom フィードをセットアップする場合は、Atom 構成ファイルの <cics:authority> 要素で各フィードについて異なる権限名または異なる日付を指定します。日付が異なるタグ URI は、その他のすべての情報が同じであっても、等価にはなりません。
- 単一の Atom フィードを扱うユーザー作成サービス・ルーチンによって提供される Atom エントリーでは、Atom ID は固有です。しかし、ユーザー作成サービス・ルーチンによって複数のフィードが提供される場合には、Atom ID は固有ではありません。ユーザー作成サービス・ルーチンによって複数のフィードが提供される場合は、Atom ID 用に代替形式を選択するか、または Atom 構成ファイルの <cics:authority> 要素で各フィードについて異なる権限名または日付を指定してください。

Atom ID 用の代替形式

CICS によって生成されるタグ URI 形式を使用する代わりに、Atom 構成ファイルのプロトタイプ Atom エントリーに <atom:id> 要素を使用することにより、代替形式を Atom ID 用に指定することもできます。 CICS は、この代替形式に対してセレクター値を追加して、Atom エントリーごとに固有の Atom ID を生成します。

代替の Atom ID 形式を使用する場合は、得られる Atom ID が固有であり、Atom 形式仕様の要件 (RFC 4287) を満たしていることを確認してください。

形式設定が正しく行われるようにするために、CICS は、Web クライアントによって提供された Atom ID をすべて無視し、フィードの Atom 構成ファイルで指定された形式を代わりに使用します。

Atom ID の格納

CICS は Atom エントリーをサービス提供するたびにその Atom エントリーの同じ Atom ID を生成できるため、Atom ID を Atom エントリーと共に格納することは必須ではありません。この機能により、メタデータを格納するフィールドが含まれないリソースから Atom エントリー・データを提供することが可能となります。ただし、Atom ID を変更しないでおくこと、構成ファイル内で Atom ID を変更しないこと、Atom エントリーを別のリソースに移動しないこと、あるいはタグ URI においてリソースまたはサービス・ルーチンの名前を変更しないことが前提となります。

しかし RFC 4287 では、Atom ID を Atom エントリーと共に格納することが推奨されています。 Atom エントリーのデータを保持するリソースに Atom ID を格納できるのであれば、この推奨事項に従うことができます。 Atom エントリーをファイルに格納する場合、このフィールドはレコード用の固有キーにできます。CICS ま

たはサービス・ルーチンは、このフィールドに Atom エントリーの完全な Atom ID を格納します。Atom エントリーと共に格納された Atom ID は、その Atom エントリー用に CICS が生成することになっている Atom ID と異なっていてもよく、これをオーバーライドできます。

サービス・ルーチンの場合、CICS は **ATMP_ATOMID** パラメーターを使用して Atom エントリーのプロトタイプ Atom ID を送信しますが、その際に Atom 構成ファイルの <cics:authority> 要素または <atom:id> 要素で指定された情報を使用します。完全な Atom ID を生成するには、サービス・ルーチンによって、セレクター値を追加してプロトタイプ Atom ID を完全にするか、またはそれを無視して有効な独自の Atom ID で置換できます。例えば、16 進数の Universally Unique Identifier (UUID) を使用して urn:uuid スキームで URI を生成できます。これは、RFC 4122「*A Universally Unique IDentifier (UUID) URN Namespace*」に説明されています。サービス・ルーチンは、**ATMP_ID_FLD** パラメーターで指定されたフィールドを使用してリソース・レコードに Atom ID を格納し、次いで **ATMP_ATOMID** パラメーターを使用してそれを戻すことができます。

正確さを保つために CICS は、Web クライアントによって提供された Atom ID を無視します。それらの Atom ID を、ファイルまたは一時記憶域キューのレコードに格納したり、サービス・ルーチンに渡したりしません。

RFC 4287 では、Atom エントリーを再利用するか、または別の場所に移動する場合に、Atom ID がその Atom エントリーと共に存続することを求めています。 Atom ID を Atom エントリーと共に格納するなら、Atom エントリーを別の場所に移動しても、この要件に準拠できます。 Atom ID を Atom エントリーと共に格納しない場合は、Atom エントリーを別の場所に移動しないでください。

Atom フィードの Atom ID

Atom フィードにも固有 ID が付与されます。 Atom 構成ファイルで <cics:authority> 要素を使用して、CICS が Atom ID としてタグ URI を生成するようにした場合、CICS では、Atom エントリーの場合と同じ形式で Atom フィードの Atom ID が生成されますが、Atom エントリーの場合に追加されるセレクター値や固有 ID は生成されません。例えば、次のとおりです。

```
tag:example.com,2009-01-08:tsqueue:WB20TSQ
```

代わりの Atom ID 形式を使用する場合は、Atom フィードの <atom:id> 要素を使用して、Atom フィードの完全な Atom ID を指定します。この場合は、Atom ID が固有であり、Atom 形式仕様の要件 (RFC 4287) を満たしていることを確認してください。

関連資料

327 ページの『<cics:authority> 要素』
Atom 構成ファイルの <cics:authority> 要素は、CICS が個々の Atom エントリーの Atom ID として使用するタグ URI を作成する際に使用する権限名と関連日付を提供します。

第 18 章 CICS によって Atom フィードがサポートされる方法

CICS では、Atom フィードをサポートするために、CICS Web サポートの HTTP サーバー機能と、 Atom 形式およびプロトコルをサポートするサーバーで必要なアクションを実行するいくつかの追加機能を使用します。ユーザーは、 Atom フィードのデータを提供するリソースを選択またはセットアップし、フィードを CICS に対して定義する必要があります。

CICS から Atom フィードを提供するには、事前に CICS Web サポートの基本コンポーネントを構成して、CICS を HTTP サーバーとしてセットアップする必要があります。

既存のリソースが保持または生成するデータから Atom フィードを作成できます。これらのリソースには、一時記憶域キュー、ファイル、データベース・アプリケーションのレコード、Web サービス、または既存のアプリケーション・プログラムによって生成された出力が含まれます。リソース内の 1 つのレコードは、1 つの Atom エントリーのデータを保持します。あるいは、新規のリソースをセットアップして、Atom エントリーを含めることもできます。

リソースが CICS に定義されたファイルまたは一時記憶域キューであり、リソース内のレコードの記述が COBOL、C、C++、または PL/I で書かれた言語構造の場合、 CICS はリソースから直接データを抽出して Atom フィードを生成できます。CICS XML アシスタント・プログラムへの入力として言語構造を使用して、リソースの構造を定義する XML バインディングを生成すると、CICS は Atom 文書内の正しい要素にデータをマップできます。

リソースの各レコードからデータを抽出して Atom エントリーを形成し、一連のコンテナーで CICS にデータを提供するプログラム (サービス・ルーチンと呼ばれます) を作成することにより、そのリソースを Atom フィードとして使用することもできます。リソースで XML バインディングを生成することができる場合、サービス・ルーチンは XML バインディング内の情報を使用できますが、サービス・ルーチンは XML バインディングを必要とするわけではありません。

リソースを識別または作成し、XML バインディングを生成またはサービス・ルーチンを書いた後は、以下の項目を作成することにより、CICS に対して Atom フィードを定義します。

- ATOMSERVICE リソース定義: Web クライアント要求に応じて Atom 文章を生成する際に CICS がデータを入手する場所を指定します。
- URIMAP リソース定義: Web クライアントが Atom フィードの http 要求を行った場合の CICS の処理方法を指定します。 URIMAP リソースは ATOMSERVICE リソース定義を参照します。 URIMAP リソース定義をサポートするには、CICS Web サポート用のインバウンド・ポートを定義する TCPIPSERVICE 定義が必要です。 CICS はそのポートで HTTP 要求を受信することができます。
- Atom 構成ファイル: Atom フィード文書の XML 構文が含まれると同時に、フィードのデータがあるリソースを特定する要素など CICS に特有の要素がいくつか

含まれます。 CICS は Atom 構成ファイル内の情報を使って、複数の Atom エントリーを含む Atom フィード文書を作成します。CICS はリソースのデータを使ってこの Atom エントリーを作成します。

フィードの Atom エントリーを管理し編集するために Web クライアントを使用可能にするには、Atom フィードをコレクションとしてセットアップするための追加の手順を実行できます。コレクションをセットアップするには、新しい URIMAP 定義を作成して、フィードとは別にコレクションを使用可能にします。新しい ATOMSERVICE 定義および Atom 構成ファイルを作成するために、同じデータから Atom フィード用の対応するファイルをコピーして、コレクション用であることを示すために再定義し、小さい変更を加えることもできます。その後、Atom サービス文書および Atom カテゴリー文書 (任意) を作成してコレクションを定義し、それらの文書を CICS を介して使用可能にします。サービス・ルーチンを使用する場合、コレクションから Atom エントリーを追加、編集、削除するための Web クライアント要求を処理するには、それをコーディングする必要があります。

Atom フィードとの相互作用

Atom フィードをセットアップした後、Web クライアントはその Atom フィードにアクセスして Atom エントリーのリストを取得することができます。 CICS はサービス・ルーチン (使用している場合) と共にサーバーとして機能し、 Web クライアントの HTTP 要求を受信して、複数の Atom エントリーが含まれる Atom フィード文書を返します。無料または市販の多数の Web クライアント・アプリケーションは、Atom フィードを要求、受信、および表示することができます。これには、最近のほとんどの Web ブラウザー、専用のフィード・リーダーのほか、マッシュアップ作成用アプリケーションなど、追加機能を提供するアプリケーションがあります。使用するアプリケーションで Atom 形式がサポートされていることを確認してください。独自の Web クライアント・アプリケーションを作成して、Atom フィードのデータに対する GET 要求を行うこともできます。

Atom フィードをコレクションとしてセットアップした場合、Atom フィードに対する HTTP POST、PUT、および DELETE 要求をサポートする Web クライアントでフィード内のエントリーの管理および編集を行えるようにすることもできます (Atom 出版プロトコルで説明されています)。使用中の Web クライアントにこの機能がない場合は、独自の HTTP 要求の作成と送信、および応答の表示を行える Web クライアント・アプリケーションを使用できます。独自の Web クライアント・アプリケーションを作成して、Atom コレクションに対する POST、PUT、および DELETE 要求を行うこともできます。CICS がリソースを直接管理している場合は、コレクションで使用可能にしたデータに CICS によって Web クライアントの編集要求が適用され、適切な応答が返されます。サービス・ルーチンを使用してデータを提供している場合、Web クライアントの要求は CICS によってコンテナー・インターフェースを使用してサービス・ルーチンに渡されます。この場合は、要求に対する応答としてリソースを変更するようにサービス・ルーチンをコーディングしてください。

Atom フィードの CICS サポートおよび Web クライアントが Atom フィードと相互作用する方法について学習するには、`https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp` にある CICS TS for z/OS バージョン 4.1 インフォメーション・センターの Web 2.0 シナリオ、『従業員情報を処理する Atom

フィードの作成』の指示に従って、サンプル Atom コレクションをセットアップし、使用してください。

CA8K SupportPac からの Atom フィード

CICS TS for z/OS バージョン 3.1 または CICS TS for z/OS バージョン 3.2 で CA8K SupportPac を使用して Atom フィードをセットアップした後、CICS TS for z/OS バージョン 4.1 で Atom フィードのサポートを使用できるようにアップグレードを希望する場合は、引き続き独自のサービス・ルーチンを使用することができます。ただし、PIPELINE リソース定義、パイプライン構成ファイル、およびリソース・レイアウト・マッピング構造の代わりに、ATOMSERVICE リソース定義、Atom 構成ファイル、および XML バインディングを使用する必要があります。また、サービス・ルーチンのコードに変更を加えて、コンテナーの名前を変更し、コンテナーの 1 つにある新規パラメーターについて記述した後、モジュールを再コンパイルする必要があります。

第 19 章 Atom エントリー・データを提供するリソースのセットアップ

Atom フィードまたはコレクションは、一連の Atom エントリー、つまりデータの項目と適切なメタデータで構成されます。 CICS によってサービス提供される Atom フィードの場合、Atom エントリーのデータはリソース内のレコードから取られます。このリソースは、ファイルの場合もあれば、一時記憶域キューであったり、データベース表などの別のリソースである場合もあります。提供される Atom エントリーは、1 つのレコードにつき 1 つです。

このタスクについて

リソースのレコードは、Atom エントリーのメタデータの項目や Atom エントリーの内容を保持することもあり、Atom エントリーの内容のみを保持することもあります。 Atom フィードをセットアップする際、リソース・レコードに保持されていないメタデータの必須項目が CICS によって提供されるようにすることができます。

Atom フィード内の Atom エントリーにデータを提供する際は、以下のいずれかのリソースを使用できます。

- Atom エントリーを含めるために作成する新規の VSAM ファイルまたは一時記憶域キュー。
- CICS に対して定義されている既存の VSAM ファイルまたは一時記憶域キュー。CICS はこれらのリソースからデータを直接抽出して Atom フィードを生成することができます。 CICS は、あらゆるタイプの VSAM ファイルから Atom フィードのデータを抽出できますが、NONUNIQUEKEY 属性によって定義された代替索引ファイルだけは例外です。ファイルには、そのレコード用の固有キーが必要です。 CICS では、BDAM ファイルから直接 Atom フィードのデータを抽出することはできません。
- CICS アプリケーション・プログラムからアクセスできるその他すべてのリソース。 CICS または非 CICS リソースは、サービス・ルーチンと呼ばれる CICS アプリケーション・プログラムを使用して配信することができます。このサービス・ルーチンは、Atom エントリー用のデータをリソースから抽出し、コンテナーに入れて CICS に提供します。

手順

Atom フィードのデータを提供するためのリソースに適した手順に従ってください。

- Atom エントリーを含める新規の VSAM ファイルまたは一時記憶域キューを作成する場合は、 278 ページの『Atom エントリーを格納するための CICS リソースの作成』の指示に従ってください。
- VSAM ファイルまたは一時記憶域キュー (TSQ) 内のデータ用の XML バインディングを作成または再利用します。データ用の XML バインディングが含まれる既存の XMLTRANSFORM リソースがある場合は、Atom フィードで使用できます。既存の XMLTRANSFORM リソースがない場合は、以下のようにします。

  1. リソース内のレコードの構造を記述する言語構造を見つけるか作成します。

– COBOL、C、C++、または PL/I の高水準言語構造、またはコピーブックを使用できます。言語構造は、区分データ・セットに入っている必要があります。 CICS アプリケーション・プログラムによって使用されるファイルまたは一時記憶域キューの場合、言語構造はすでに存在しているはずです。レコードの言語構造が存在しない場合には、それを作成することができます。
– あるいは、リソース内のレコードの構造を記述する XML スキーマまたは WSDL 文書を使用することもできます。

2. 「*CICS アプリケーション・プログラミング・ガイド*」の『言語構造からのマッピングの生成』で説明されているように、言語構造を使用してリソースの XML バインディングを生成します。 あるいは、CICS Explorer™ のファイル・インポート・ウィザードを使用してソース言語ファイルを CICS バンドル・プロジェクトにインポートし、Atom フィード用の XML バインディングと関連スキーマを作成することもできます。次に、このバンドル・プロジェクトを CICS 領域にエクスポートします。ファイル・インポート・ウィザードについて詳しくは、 311 ページの『CICS Explorer を使用して Atom フィード用の XML バインディングを作成する』を参照してください。

- これ以外の CICS または非 CICS リソースを配信する場合は、リソース内のレコードから各 Atom エントリーのデータを抽出するサービス・ルーチンを作成し、データを一連のコンテナーに入れて CICS に提供します。 サービス・ルーチンの作成については、 283 ページの『Atom エントリー・データを提供するプログラムの作成』を参照してください。

次のタスク

Atom エントリー・データを保持するリソースを選択し、このデータの配信をサポートする XML バインディングまたはサービス・ルーチンを作成した後、 315 ページの『第 20 章 Atom フィード用の CICS 定義のセットアップ』の指示に従って Atom フィードをセットアップします。 BUNDLE リソースを作成およびインストールし、XML バインディング・ファイルの場所を定義した XMLTRANSFORM リソースを作成した場合は、その XMLTRANSFORM リソースを他の Atom フィードで再利用できます。

Atom エントリーを格納するための CICS リソースの作成

データを Atom エントリーとして格納するには、CICS でファイルまたは一時記憶域キューを作成し、その構造を説明する言語構造を COBOL、C、C++、または PL/I で作成します。

このタスクについて

新しいファイルまたは一時記憶域キューの各レコードは、単一の Atom エントリーを表します。レコードの各フィールドには、 Atom エントリー内の 1 つの要素 (コンテンツ、またはタイトルなどのメタデータ項目) のデータが含まれます。Atom フィードをセットアップする際は、Atom 構成ファイルで <cics:fieldnames> 要素を使用して、これらのフィールドの名前を CICS に対して指定します。CICS は各レコードからデータを抽出して Atom エントリーを組み立てます。

Atom エントリーで使用可能な要素の完全なリストと説明は、 RFC 4287 (「*The Atom Syndication Format*」。http://www.ietf.org/rfc/rfc4287.txt から入手できます) に記載されています。CICS でこれらの要素がすべてサポートされているわけではありません。また、CICS でサポートされている一部の要素は、ユーザーのファイルまたは一時記憶域キューではサポートされず、Atom 構成ファイルでのみ指定できます。CICS がファイルおよび一時記憶域キュー内でサポートしている要素のリストと説明については、 329 ページの『<cics:fieldnames> 要素』を参照してください。 CICS が Atom フィードおよび Atom エントリーでサポートしている要素、およびサポートしていない要素の完全なリストについては、 341 ページの『CICS 用の Atom 要素のリファレンス』を参照してください。

手順

1. Atom エントリーのデータを格納するためのリソースとして、一時記憶域キューを使用するか、ファイルを使用するかを決めます。
   - 一時記憶域キューは使用前に CICS に対して定義する必要がないため、 CICS で Atom フィードをテストする場合は、一時記憶域キューが適しています。ただし、セキュリティー手段を適用する場合は、CICS リソース定義をセットアップする必要があります。一時記憶域キューは、Atom エントリーが長期間必要なものではない場合 (例えば、アプリケーションでのイベントについてアラートを発行する場合) の Atom フィードにも適しています。
   - ファイルは、使用前に CICS に対して定義する必要があり、通常は物理データ・セットが必要であるため、一時記憶域キューよりもセットアップに時間がかかります。ただし、ファイルは、編集可能なコレクションとしてセットアップする可能性のあるフィードを含め、任意の Atom フィードを長期保管するのに適しています。Atom エントリーを保持するファイルはレコードの固有キーを持つ必要があり、NONUNIQUEKEY 属性によって定義された代替索引ファイルを使用することはできません。 Atom エントリーを保持する VSAM ファイルは種類を問いませんが、ESDS (入力順データ・セット) ファイルは、編集可能コレクションとしてセットアップするフィードに対しては良い選択とは言えないことにご注意ください。その理由については、 283 ページの『Atom フィードが含まれる ESDS ファイル』で記されています。 BDAM ファイルは使用できません。
2. ファイルまたは一時記憶域キューのレコードの内容を計画します。 Atom 構成ファイルですべてのメタデータを指定することが可能であるため、 CICS がレコード内で必要とする唯一の項目は Atom エントリーの内容です。ただし、Atom エントリーを含めるための専用ファイルまたは一時記憶域キューをセットアップする場合は、各 Atom エントリーに固有のメタデータを提供するために使用できるよう、レコード内にメタデータ用のフィールドを含めることができます。 以下のリストでは、レコード内にフィールドとして含めることができるデータの項目が要約されています。それが必須かオプションかも示されています。

   Atom ID

   Atom エントリーの固有 ID。 Atom ID の形式について詳しくは、 269 ページの『Atom エントリーの Atom ID』を参照してください。

   Atom エントリーには、固有の Atom ID がなければなりません。 CICS は、Atom フィードをサービス提供する際に、Atom 構成ファイルに指定されている情報を使用して、それぞれのエントリーに固有の Atom ID を生成できます。 CICS によって作成された Atom ID は、ユーザーが

ファイルまたは一時記憶域キューの名前を変更したり、Atom 構成ファイル内の関連情報を変更したり、Atom エントリーを別のリソースに移動したりしない限り、Atom エントリーの存続期間中は同じままです。そのため、レコードに Atom ID を格納するためのフィールドを含める必要はありません。

ただし、Atom 形式に完全に準拠するには、エントリーが再利用される、または別の場所に移動する場合に、 Atom ID がそのエントリーと共に残る必要があります。Atom エントリーをこのファイルまたは一時記憶域キュー以外の場所で使用する可能性がある場合や、Atom ID をエントリーと共に格納することを求める RFC 4287 の推奨内容に従うことが望ましい場合は、レコードに Atom ID を保持するためのフィールドを含めてください。 Atom エントリーをファイルに格納する場合、このフィールドはレコード用の固有キーにできます。

作成者の詳細

Atom エントリーの基本作成者の個人名、E メール・アドレス、および Web サイト。各項目別に 3 つのフィールドに分けられます。 Atom フィードまたはすべての Atom エントリーに関して作成者の名前を指定する必要がありますが、他のフィールドはオプションです。 Atom エントリーにさまざまな作成者が含まれている場合は、レコードに名前用のフィールド、および必要に応じてその他の詳細に関するフィールドを含めます。作成者の名前およびその他の詳細情報がすべての Atom エントリーについて同じ場合は、代わりに Atom 構成ファイルで作成者の名前および詳細情報を指定してください。

カテゴリー

エントリーを分類するカテゴリー用語。このフィールドはオプションです。この Atom フィードを編集可能なコレクションとしてセットアップし、コレクションを説明するためにカテゴリーを使用する予定がある場合は、このフィールドを含めます。このフィードをコレクションとしてセットアップする予定がない場合でも、フィードの利用者に役立つ可能性がある場合は、このフィールドを含めることができます。

コンテンツ

Atom エントリーで公開されるコンテンツ全体。CICS では、すべての Atom エントリーについてコンテンツが必要です。コンテンツは、プレーン・テキスト、HTML、XHTML、XML、または他のテキスト・メディア・タイプとすることが可能です。 CICS は、非テキスト・コンテンツや、コンテンツのない Atom エントリーはサポートしていません。メタデータ用のフィールドを含める場合は、コンテンツを保持するフィールド、またはレコード内のネスト・フィールドの副構造が必要です。メタデータのフィールドを含めない場合、CICS ではファイルまたは一時記憶域キューのレコード全体がエントリーのコンテンツとして公開されます。

コンテンツ・タイプ

テキストまたは XML などの、Atom エントリーのコンテンツのメディア・タイプ。このフィールドはオプションです。すべての Atom エントリーのコンテンツのタイプが同じ場合は、代わりに Atom 構成ファイルでメディア・タイプを指定できます。

最終編集日付

レコードが最後に編集された時刻を示すタイム・スタンプ。タイム・スタンプは、RFC 3339 で説明されている XML dateTime 形式のものか、または CICS ABSTIME 値を使用できます。日時スタンプについて詳しくは、267 ページの『Atom エントリーの日時スタンプ』を参照してください。この Atom フィードを編集可能なコレクションとしてセットアップする予定がある場合、このフィールドを組み込むことで Atom エントリーが最後に編集された時刻に基づいてそれらのエントリーを返すことができます。これは、コレクションの Atom 出版プロトコルの推奨事項です。このフィードをコレクションとしてセットアップする予定がない場合は、このフィールドを含めないでください。

公開開始日付

レコードが初めて Atom エントリーとして作成または公開された時刻を示すタイム・スタンプまたは ABSTIME 値。このフィールドはオプションです。フィードの利用者に役立つと思える場合は、このフィールドを含めます。

タイトル

Atom エントリーのタイトル。タイトルでは、CICS はプレーン・テキストのみをサポートします。各 Atom エントリーで 1 つのタイトルが必要なので、通常はこのフィールドを含める必要があります。すべての Atom エントリーのタイトルが同じ場合は、代わりに Atom 構成ファイルでこのタイトルを指定できます。

要約

Atom エントリーの要約。要約では、CICS はプレーン・テキストのみをサポートします。このフィールドは、エントリーのコンテンツが非テキスト・メディア・タイプの場合は必須ですが、それ以外の場合はオプションです。 CICS では、Atom エントリー内で非テキスト・コンテンツはサポートされていません。

最終更新日付

レコードが最後に更新された時刻を示すタイム・スタンプまたは ABSTIME 値。各 Atom エントリーで 1 つの更新タイム・スタンプが必要なので、通常はこのフィールドを含める必要があります。ファイルまたは一時記憶域キューでこのタイム・スタンプまたは ABSTIME 値を含めることができない場合は、このフィールドを省略できます。その場合、CICS では Atom フィード文書内のエントリーを公開するときに、現在日時か、 Atom 構成ファイルで指定されている適切な代替デフォルト値を提供することができます。

3. ファイルまたは一時記憶域キューの言語構造またはコピーブックを COBOL、C、C++、または PL/I で作成します。言語構造は、ファイルまたは一時記憶域キュー内のレコードのフィールドを記述します。各フィールドの名前、コンテンツ・タイプ、および長さが表示順に示されています。
   COBOL、C、C++、または PL/I のアプリケーションを使用してファイルまたは一時記憶域キューにレコードを作成する予定の場合、アプリケーションはこの言語構造を使用してファイルまたは一時記憶域キューに書き込みを行います。 この言語構造は、ファイルまたは一時記憶域キュー用の XML バインディングを作成するためにも必要です。そのため、使用するアプリケーションが別の言語で作成されている場合やアプリケーションを使用しない (例えば、CICS で Atom フ

ィードをテストする際に、レコードの構造が単純で、CECI トランザクションを使用してファイルまたは一時記憶域キューに書き込むことができる) 場合でも、この言語構造が必要になります。

注: Atom エントリーのメタデータを提供するために使用されるフィールドは、言語構造内でネストさせないでください。レコード内のメタデータ・フィールドは、言語構造内ですべて同じレベルでリストされなければなりません。 Atom エントリーのコンテンツを提供するフィールド内では、ネストされたフィールドの構造を使用できます。
言語構造は、固定レコード長が 80 バイトの区分データ・セットに格納します。以下の COBOL 言語構造の例では、各要素のデータを含めるための適切な長さの英数字フィールドを宣言しています。

```
******************************************************************
* Name: SAMPBIND.cob                                             *
*                                                                *
*                                                                *
* This is a COBOL copy book to describe the data record.         *
* You can generate a binding file from this.                     *
*                                                                *
******************************************************************

     03 TITLE-FIELD                    PIC X(50).
     03 SUMMARY-FIELD                  PIC X(500).
     03 ATOMID-FIELD                   PIC X(20).
     03 CONTENT-FIELD                  PIC X(500).
     03 AUTHOR-NAME-FIELD              PIC X(30).
     03 AUTHOR-EMAIL-FIELD             PIC X(256).
     03 AUTHOR-URI-FIELD               PIC X(256).
     03 EDITED-FIELD                   PIC X(25).
     03 UPDATED-FIELD                  PIC X(25).
     03 PUBLISHED-FIELD                PIC X(25).
     03 CATEGORY-FIELD                 PIC X(20).
```

4. 「*CICS アプリケーション・プログラミング・ガイド*」の手順に従って、言語構造を CICS XML アシスタントへの入力として使用し、XML バインディングを作成します。
5. ファイルを使用して Atom エントリーを格納することにした場合は、次のようにします。
   a. 「*CICS System Definition Guide*」の手順に従って、適切な VSAM データ・セットをセットアップします。
   b. FILE リソース定義 の情報を使用して FILE リソース定義を作成し、インストールすることにより、CICS に対してファイルを定義します。
6. 一時記憶域キューを使用して Atom エントリーを格納し、それに対してセキュリティーとリカバリーの設定を指定することにした場合は、TSMODEL リソース定義 の情報を使用して、一時記憶域モデル (TSMODEL) を定義します。

次のタスク

ご使用のファイルまたは一時記憶域キュー内にあるレコードを処理できるアプリケーションが既にある場合、ユーザー・アプリケーションやその他の適切な方法を使用して、少なくとも 1 つのレコードをファイルまたは一時記憶域キューに書き込むことにより、セットアップをテストします。その際、一時記憶域キューの場合は WRITEQ TS コマンドを、ファイルの場合は WRITE コマンドを使用します。

Atom フィードが含まれる ESDS ファイル

Atom フィード用の Atom エントリー・データを保持するために ESDS (入力順データ・セット) ファイルを使用できますが、Atom エントリーの削除に関して制約事項が幾つかあります。これらは、フィードを編集可能なコレクションとしてセットアップする際に適用されます。

Web クライアントは、HTTP 要求を DELETE メソッドを使用して実行すると、コレクション内の Atom エントリーを削除できます。 ESDS ファイルを使用する場合、DELETE メソッドを用いた HTTP 要求がサポートされるのは、ESDS に代替索引が定義されていない場合のみです。

DELETE 要求の応答として、CICS は ESDS から関連レコードを削除します。その際、論理削除を示すために最初のバイトを 'FF'x に書き直します。 Web クライアントがその後の HTTP 要求で GET メソッドを使用して、削除済みレコード内にあった Atom エントリーを取り出そうとすると、CICS はその GET 要求に対して「not found (未検出)」応答を戻します。

ESDS ファイルを Atom コレクションとして定義するときは、以下のいずれかの方法を使用して、その ESDS ファイルを使用する他のアプリケーション・プログラムが削除済みレコードを適切に処理することを確かめる必要があります。

- ESDS の FILE リソース定義で、DELETE を NO に設定します。
- あるいは、'FF'x で始まるレコードを論理削除されたものとして処理するように、アプリケーションをコーディングします。

コレクション用の Atom エントリー・データを格納する新しいリソースをセットアップする場合に、こうした制約事項を回避にするには、ESDS 以外の VSAM ファイル・タイプを選択してください。

コレクションとして定義されていない Atom フィードに関してのみ ESDS ファイルが使用されている場合、Web クライアントは DELETE メソッドを使用して要求を行えないので、こうした制約事項は当てはまりません。ただし、Atom フィード用 Atom エントリー・データを格納するために新しいリソースをセットアップする場合、Atom フィードを後ほどコレクションとしてセットアップする場合に備え、ESDS ファイルの使用は避けてください。

Atom エントリー・データを提供するプログラムの作成

サービス・ルーチンを使用して、CICS プログラムからのアクセスが可能な任意のデータ (DB2 データベースのレコード、ファイル内のレコード、COMMAREA など) から Atom フィードを提供することができます。以下の手順は、Atom フィードに関する HTTP GET 要求に応答するプログラムを作成する方法を示しています。

このタスクについて

Web クライアントは、フィードの複数の Atom エントリーを要求したり、特定のエントリーを要求したりできます。 CICS は、Web クライアントから要求を受信して、各クライアント要求に関する情報と共にプログラムにリンクします。CICS は、クライアントが要求する Atom エントリーごとに 1 回ずつプログラムにリンクし、プログラムはそのたびに 1 つのエントリーを返します。

プログラムは、このフィードの Atom エントリーのデータを保持するリソース (データベースやファイルなど) のレコードから抽出したデータを使用して Atom エントリーを提供します。このプロセスの概要については、 254 ページの『CICS からの Atom フィードのデータ処理』を参照してください。

CICS はコンテナー・インターフェースを使用してサービス・ルーチンと通信します。 EXEC CICS GET CONTAINER および EXEC CICS PUT CONTAINER コマンドを使用してコンテナーと対話します。 C 言語サンプル・サービス・ルーチン DFH$W2S1 は、コンテナーを使用して HTTP GET 要求に応答する方法を示しています。 COBOL サンプル・サービス・ルーチン DFH0W2F1 もコンテナーの使用法を示していますが、HTTP GET 要求だけでなく、PUT、POST、および DELETE 要求にも応答するので、DFH0W2F1 サンプルの方が複雑です。

Web クライアント要求は HTTP 要求であるため、WEB READ HTTPHEADER コマンドや WEB READ QUERYPARM コマンドなどの CICS Web API コマンドを使って対話することもできます。これらのコマンドの使い方を知っている場合は、サービス・ルーチンで使用して、Web クライアント要求から直接情報を入手することができます。CICS が DFHATOMPARMS コンテナーには提供しない情報を入手することもできます。

Atom エントリーのデータが含まれるリソースの XML バインディングを作成できる場合、Atom エントリーのデータが含まれるリソース・レコード内のフィールドの名前と長さに関する情報を、CICS からプログラムに DFHATOMPARMS コンテナーで渡すことができます。プログラムは、この情報を使用して、リソース・レコード内のメタデータ・フィールドを見つけることができます。これらのリソース・ハンドリング・パラメーターを使用することにより、複数のリソースを処理できる汎用サービス・ルーチンを作成できます。ただし、リソース・ハンドリング・パラメーターを使用せずに、リソース構造に関する情報をプログラム内に直接コーディングすることもできます。

GET 要求に応答するには、サービス・ルーチンで以下のタスクを実行する必要があります。

手順

1. EXEC CICS GET CONTAINER コマンドを使用して、DFHATOMPARMS コンテナー内のデータを取得します。 CICS はこのコンテナーを使用して、要求に関する情報をサービス・ルーチンに提供します。 サンプル・サービス・ルーチン DFH$W2S1 では、DFHATOMPARMS 内のパラメーターを読み取る方法を示しています。 287 ページの『DFHATOMPARMS コンテナー』には、CICS がこのコンテナーで渡すすべてのパラメーターが記載されています。
2. **ATMP_HTTPMETH** パラメーターの値を調べ、要求メソッドが GET であることを確認します。 GET、POST、PUT、および DELETE 以外のメソッドについては、CICS からエラーまたは適切な応答が返されます。Atom コレクションに対する HTTP PUT、POST、または DELETE 要求への応答についての指示はここには含めませんが、 382 ページの『サービス・ルーチンで Atom コレクション編集要求を処理する方法』に記載されています。
3. プログラムが CICS に返す必要がある Atom エントリーのデータが含まれているリソースのレコードを識別するには、DFHATOMPARMS コンテナー内の **ATMP_ATOMTYPE** パラメーターと **ATMP_SELECTOR** パラメーターの値を使用しま

す。 特定の Atom エントリーを識別するセレクター値が **ATMP_SELECTOR** パラメーターに含まれていることもあります。 264 ページの『Atom エントリーのセレクター値』では、セレクター値となり得るものについて、また CICS およびサービス・ルーチンがそれを使う方法について説明されています。

a. **ATMP_SELECTOR** の値がヌルで、**ATMP_ATOMTYPE** の値が "feed" の場合、クライアントは特定の Atom エントリーを指定していないので、フィードに追加された最新の Atom エントリーを保持するリソース内のレコードを見つけます。 例えば、Atom エントリーのデータがデータベースに保持されている場合には、データベースに追加された最新のレコードを使用します。

b. **ATMP_SELECTOR** にセレクター値が含まれていて、**ATMP_ATOMTYPE** の値が "feed" の場合は、セレクター値で識別されるリソース内のレコードを見つけます。 この値の組み合わせは、CICS がクライアント要求を完了するためにフィードからの 2 番目以降の Atom エントリーを必要としており、サービス・ルーチンによって前の反復で提供されたセレクター値を使用して CICS がこうした Atom エントリーのいずれかを要求していることを示す場合があります。またこの値の組み合わせは、部分リストなどの特定の範囲の Atom エントリーが含まれるフィード文書をクライアントが要求した場合に、その要求内の最初の Atom エントリーにも使用されます。

c. **ATMP_SELECTOR** にセレクター値が含まれていて、**ATMP_ATOMTYPE** の値が "entry" の場合は、セレクター値で識別されるリソース内のレコードを見つけます。 この値の組み合わせは、クライアントがフィード内の 1 つの既知の Atom エントリーを要求していることを示します。

4. Atom エントリーのデータが含まれているリソースの XML バインディングがあり、リソース・ハンドリング・パラメーターを使用してリソース内のフィールドに関する情報を渡す場合は、**ATMP_TITLE_FLD** パラメーターおよびその他 **_FLD** で終わるパラメーターの値を使用して Atom エントリーの要素のデータが含まれる各フィールドの名前と長さを識別するようにサービス・ルーチンをコーディングします。 リソースのデータを使用する Atom フィードの Atom 構成ファイルをセットアップする際は、Atom 構成ファイルの <cics:fieldnames> 要素にこれらのフィールドの名前を指定する必要があります。CICS はリソース・ハンドリング・パラメーターを使用して、その名前をサービス・ルーチンに渡します。 287 ページの『DFHATOMPARMS コンテナー』には、リソース・ハンドリング・パラメーターが記載されています。
5. PUT CONTAINER コマンドを使用して、Atom エントリーの内容が含まれる DFHATOMCONTENT という名前のコンテナーを、リソースから識別したレコードに記述されているとおりに、DATATYPE(CHAR) を指定して作成します。このコンテナーは必須です。サンプル・サービス・ルーチン DFH$W2S1 ではコンテナーを更新する方法を示しており、 300 ページの『DFHATOMCONTENT コンテナー』ではコンテナーに入れる内容について説明しています。
6. リソースから識別したレコードに、Atom エントリーのメタデータ (タイトルなど) を提供するフィールドが含まれている場合、 301 ページの『コンテナー内の Atom エントリー・メタデータを戻す』の手順に従って、オプションのコンテナーを使用してこのメタデータを CICS に返します。
7. リソースから識別したレコードに、データの作成時点または更新時点の日時スタンプを提供するフィールドが含まれている場合、DFHATOMPARMS コンテナー内の **ATMP_PUBLISHED** および **ATMP_UPDATED** パラメーターの新しい値としてそれらを返します。 サンプル・サービス・ルーチン DFH$W2S1 では、これ

らのパラメーターの新しい値を返す方法を示しています。 これらの日時スタンプの形式について詳しくは、 267 ページの『Atom エントリーの日時スタンプ』を参照してください。

8. サービス・ルーチンへの入力に対する DFHATOMPARMS コンテナーの **ATMP_SELECTOR** パラメーターの値がヌルの場合、つまり、Web クライアントが特定の Atom エントリーを要求しなかった場合は、ヌル値を、返されるエントリーの適切なセレクター値で置き換えます。 サンプル・サービス・ルーチン DFH$W2S1 では、**ATMP_SELECTOR** パラメーターがヌルの場合にセレクター値を返す方法を示しています。 264 ページの『Atom エントリーのセレクター値』では、セレクター値の選択方法が説明されています。 **ATMP_SELECTOR** パラメーターにサービス・ルーチンへの入力に対するセレクター値が含まれていた場合は、変更しないでください。
9. DFHATOMPARMS コンテナーの **ATMP_ATOMTYPE** パラメーターの値が "feed" (クライアントが複数のエントリーを要求していることを示します) であった場合は、より古く、Atom エントリーを提供するために使用できるデータがリソース内に他に含まれているかどうかを確認します。
   a. より古いデータがある場合は、Atom エントリーを提供する次のデータ項目を見つけ、このデータ項目の適切なセレクター値を返して **ATMP_NEXTSEL** パラメーターで使用します。 265 ページの『Atom エントリーの順序』では、Atom エントリーを返すべき順序が説明されています。
   b. 使用可能なデータが他にない場合、ヌル値を返すために **ATMP_NEXTSEL** パラメーター内のデータの現在の長さをゼロに設定します。
10. DFHATOMPARMS コンテナーの **ATMP_ATOMID** パラメーターを読み取り、エントリーのプロトタイプ Atom ID を確認します。プロトタイプ Atom ID は、**ATMP_SELECTOR** パラメーターで指定されているように、Atom エントリーのセレクター値を追加して完成させる必要があります。必要であれば、サービス・ルーチンでそのプロトタイプ Atom ID を無視し、その Atom エントリーに対する独自の有効な Atom ID に置き換えることができます。 Atom ID の要件について詳しくは、 269 ページの『Atom エントリーの Atom ID』を参照してください。
    a. この Atom エントリー用に完全な Atom ID をリソース・レコードに既に格納している場合、**ATMP_ATOMID** パラメーターにこの Atom ID とその長さを返します。 DFHATOMPARMS コンテナーのリソース・ハンドリング・パラメーターを使用している場合、**ATMP_ID_FLD** パラメーターにはリソース内の関連フィールドの名前と長さが含まれています。
    b. リソースに Atom ID が格納されていない場合、**ATMP_ATOMID** パラメーターのデータの現在の長さをゼロに設定します。 CICS によってセレクター値が追加され、完全な Atom ID が生成されます。
11. 適切な応答コードを返し、DFHATOMPARMS コンテナーの **ATMP_RESPONSE** パラメーターで使用されるようにします。 サンプル・サービス・ルーチン DFH$W2S1 は、これを行う方法を示しています。 このコードは、正常終了を示すゼロに初期設定されています。 エラー応答が返された場合、CICS は適切なデフォルトの HTTP エラー応答を作成して、Web クライアントに送信します。 298 ページの『DFHATOMPARMS コンテナーの ATMP_RESPONSE パラメーター』には、使用可能な応答コード、および CICS が各ケースで送信する

HTTP エラー応答のリストが記載されています。 サンプル・サービス・ルーチン DFH$W2S1 は、応答コードの設定後に制御を CICS に返します。

次のタスク

サービス・ルーチンを作成したのであれば、適切な PROGRAM リソース定義を CICS に作成およびインストールして、そのサービス・ルーチンに関して記述します。 PROGRAM リソース定義では、EXECKEY(USER) 属性を使用します。ご使用の Atom フィードに関して、ATOMSERVICE リソース定義でこの PROGRAM リソースを指名する必要があります。

サービス・ルーチンを使用して Atom フィードにデータを提供する CICS 定義をセットアップしたら、CEDX トランザクションを使用して、サービス・ルーチンが HTTP 要求に応答する際にモニターおよびデバッグすることができます。 CW2A は Atom フィード用のデフォルトの別名トランザクションであり、代替の別名トランザクションをセットアップしていない限りは、このトランザクションでサービス・ルーチンが実行されます。 CEDX は指定したトランザクションの次のインスタンスをモニターするので、他のユーザーが同じ別名トランザクションを使用してこの CICS 領域内で Atom フィードを処理する場合、代替の別名トランザクションをセットアップして、ご使用のサービス・ルーチンをデバッグしている間に使用します。

DFHATOMPARMS コンテナー

DFHATOMPARMS は、Atom フィードのデータを提供するサービス・ルーチンと通信するために CICS が使用するパラメーターを含む DATATYPE(CHAR) のコンテナーです。

DFHW2AP シリーズのコピーブックは、DFHATOMPARMS コンテナーで渡されたパラメーターをサービス・ルーチンにマップします。以下のコピーブックが定義されています。

- DFHW2APD (アセンブラー用)
- DFHW2APH (C 言語用)
- DFHW2APL (PL/I 用)
- DFHW2APO (COBOL 用)

DFHW2CN シリーズのコピーブックには、DFHW2AP シリーズのコピーブックで参照される定数値が含まれています。以下のコピーブックが定義されています。

- DFHW2CND (アセンブラー用)
- DFHW2CNO (COBOL 用)
- DFHW2CNH (C 言語用)
- DFHW2CNL (PL/I 用)

DFHATOMPARMS コンテナーの入力専用パラメーター

CICS は、これらのパラメーターを使用して Web クライアントの要求に関する情報をサービス・ルーチンに提供します。これらのパラメーターには、**ATMP_TITLE_FLD** などのリソース・ハンドリング・パラメーターが含まれます。

DFHATOMPARMS コンテナー内の各入力専用パラメーターは、領域を指すポインターおよび領域内のデータの現在の長さで構成されるダブルワードのアドレスです。サービス・ルーチンでこれらのポインター、長さ、またはストレージを変更することはできません。

FLD で終わるパラメーターは、リソースを処理するために使用されます。 CICS は、これらのリソース・ハンドリング・パラメーターを使用して、一時記憶域キューなどの CICS リソースを処理する CICS 提供のサービス・ルーチンに、リソース・レコードのフィールドに関する情報を提供します。 CICS は、Atom 構成ファイル内の <atom:content> 要素および <cics:fieldnames> 要素について指定された属性、およびリソースの XML バインディングからフィールドの名前を取得します。リソース構造に関する情報をサービス・ルーチンに直接コーディングするのではなく、その情報を Atom 構成ファイルから取得するサービス・ルーチンを作成する場合は、これらのパラメーターを使用できます。これらのパラメーターを使用する場合、データを含むリソース用の XML バインディングを作成する必要があります。

ATMP_RESNAME
Atom フィードのデータを提供する CICS リソースの名前。サービス・ルーチンでは、これは常にサービス・ルーチンの名前になります。 CICS では、さまざまなリソースを直接処理する CICS 提供サービス・ルーチンのためにこのパラメーターが必要です。 CICS はこの情報を <atom:content> 要素の cics:resource 属性から取得します。

ATMP_RESTYPE
CICS リソースのタイプ (大文字)。リソース・タイプは、PROGRAM、TSQUEUE、または FILE です。サービス・ルーチンでは、リソース・タイプは常に PROGRAM になります。 CICS では、さまざまなリソースを直接処理する CICS 提供サービス・ルーチンのためにこのパラメーターが必要です。 CICS はこの情報を <atom:content> 要素の cics:type 属性から取得します。

ATMP_ATOMTYPE
処理されている Atom 文書のタイプ (小文字)。タイプ・ストリングの値は、『feed』、『collection』、または 『entry』 です。『feed』 は、クライアントが Atom フィードの複数のエントリーを要求したことを示します。『collection』 は、クライアントがコレクション内のエントリーのリストを要求したことを示します。『entry』 は、クライアントがフィードまたはコレクション内の、指定された 1 つの Atom エントリーを要求したことを示します。

ATMP_HTTPMETH
埋め込まれた、クライアント要求の HTTP メソッド。 HTTP メソッドは、GET、POST、PUT、または DELETE のいずれかです。

ATMP_TAG_AUTHORITY
Atom 構成ファイル内の <cics:authority> 要素の name 属性で指定された権限名。権限名は、タグ URI を構成するために使用できる完全修飾ドメイン名または E メール・アドレスです。この形式を選択した場合、権限名がプロトタイプ Atom ID の一部になります。

ATMP_TAG_DATE
Atom 構成ファイル内の <cics:authority> 要素の date 属性で指定された日付。この日付は、権限名と共に使用されてタグ URI を構成します。この形式を選択した場合、日付がプロトタイプ Atom ID の一部になります。

ATMP_XMLTRANSFORM
XMLTRANSFORM リソースの名前。 XMLTRANSFORM リソースは、CICS リソース用の XML バインディングを生成し、それを指定する ATOMSERVICE リソース定義をインストールすると作成されます。 XMLTRANSFORM リソースは、リソース内のレコードのレイアウトを XML 構造として記述します。この名前の長さがゼロの場合、リソース用の XML バインディングは作成されていないので、サービス・ルーチンは、リソース・レコードと Atom エントリーの要素間で独自のマッピングを実行する必要があります。

ATMP_ROOT_ELEMENT
XMLTRANSFORM リソースによってマップされた XML 構造のルート要素の名前。

ATMP_MTYPEIN
Web クライアントの HTTP 要求ボディのメディア・タイプ。要求ボディは、**ATMP_HTTPMETH** パラメーターで指定されているように、HTTP メソッドが POST または PUT の場合にのみ存在します。メディア・タイプは常に "application/atom+xml" で、Atom エントリーであることを示します。 CICS は要求ボディを DFHREQUEST コンテナー内のサービス・ルーチンに渡します。GET および DELETE 要求には要求ボディは存在しない、これらの HTTP メソッドの場合、ポインターと長さの値はいずれもゼロになります。

ATMP_MTYPEOUT
Atom フィードの Atom 構成ファイルにある <atom:content> 要素の type 属性で指定された、Atom エントリーで予期されるコンテンツのメディア・タイプ。RFC 4287 にあるとおり、プレーン・テキストでは IANA メディア・タイプ「text/plain」の代わりにメディア・タイプ「text」、「text/html」の代わりに「html」、「application/xhtml+xml」の代わりに「xhtml」を使用します。 Atom 構成ファイルにこの情報が含まれない場合、CICS はデフォルトのメディア・タイプ「application/xml」をサービス・ルーチンに渡します。サービス・ルーチンはこのメディア・タイプを使って、DFHATOMCONTENT コンテナーに返すデータの適切なマークアップを決定できます。リソース・ハンドリング・パラメーターを使用していて、個別の Atom エントリーのメディア・タイプを格納するフィールドがリソース・レコード内にある場合、**ATMP_CONTENT_TYPE_FLD** パラメーターにはこのフィールドの名前が入ります。

ATMP_WINSIZE
フィードのウィンドウ・サイズ。値は、各フィードで返されるデフォルトのエントリー数または Web クライアントが要求した代替エントリー数を含む数値ストリングです。CICS では、個々のエントリーについて一連の要求がサービス・ルーチンに対して行われるため、このパラメーターは情報のためにのみ存在します。

ATMP_ID_FLD
Atom エントリーの Atom ID が入っている、リソース・レコード内のフィールドの名前。 CICS は、Atom フィードの Atom 構成ファイルにある <cics:fieldnames> 要素の atomid 属性からフィールドの名前を取得します。CICS がこの情報をサービス・ルーチンに渡す場合、サービス・ルーチンでは、この名前付きフィールドを使用してエントリーの Atom ID を格納または検索し、その ID を **ATMP_ATOMID** パラメーターで返すことができます。このデータは、エントリーの <atom:id> 要素で使用されます。

ATMP_PUBLISHED_FLD
リソースが最後に公開された時刻が入っている、リソース・レコード内のフィールドの名前。 CICS は、Atom フィードの Atom 構成ファイルにある <cics:fieldnames> 要素の published 属性からフィールドの名前を取得します。 CICS がこの情報をサービス・ルーチンに渡す場合、サービス・ルーチンではこの名前付きフィールドを使用して、**ATMP_PUBLISHED** パラメーターで返される値を構成するために使用できるタイム・スタンプの値や ABSTIME 値を検索できます。このデータは、エントリーの <atom:published> 要素で使用されます。

ATMP_UPDATED_FLD
リソースが最後に更新された時刻が入っている、リソース・レコード内のフィールドの名前。 CICS は、Atom フィードの Atom 構成ファイルにある <cics:fieldnames> 要素の updated 属性からフィールドの名前を取得します。 CICS がこの情報をサービス・ルーチンに渡す場合、サービス・ルーチンではこの名前付きフィールドを使用して、**ATMP_UPDATED** パラメーターで返される値を構成するために使用できるタイム・スタンプの値や ABSTIME 値を検索できます。このデータは、エントリーの <atom:updated> 要素で使用されます。

ATMP_EDITED_FLD
リソースが最後に編集された時刻が入っている、リソース・レコード内のフィールドの名前。 CICS はフィールドの名前を <cics:fieldnames> 要素の edited 属性から取得します。 CICS がこの情報をサービス・ルーチンに渡す場合、サービス・ルーチンではこの名前付きフィールドを使用して、**ATMP_EDITED** パラメーターで返される値を構成するために使用できるタイム・スタンプの値や ABSTIME 値を検索できます。このデータは、エントリーの <app:edited> 要素で使用されます。

ATMP_TITLE_FLD
要求された Atom エントリーのタイトルが入っている、リソース・レコード内のフィールドの名前。 CICS はフィールドの名前を <cics:fieldnames> 要素の title 属性から取得します。 CICS がこの情報をサービス・ルーチンに渡す場合、サービス・ルーチンではこの名前付きフィールドを使用してエントリーのタイトルを検索し、そのタイトルを DFHATOMTITLE コンテナーで返すことができます。DFHATOMTITLE コンテナーのデータは、エントリーの <atom:title> 要素で使用されます。

ATMP_SUMMARY_FLD
要求された Atom エントリーの要約が入っている、リソース・レコード内のフィールドの名前。 CICS はフィールドの名前を <cics:fieldnames> 要素の summary 属性から取得します。 CICS がこの情報をサービス・ルーチンに渡す場合、サービス・ルーチンではこの名前付きフィールドを使用してエントリーの要約を検索し、その要約を DFHATOMSUMMARY コンテナーで返すことができます。DFHATOMSUMMARY コンテナーのデータは、エントリーの <atom:summary> 要素で使用されます。

ATMP_CONTENT_FLD
要求された Atom エントリーのコンテンツ全体が入っている、リソース・レコード内のフィールドの名前。 CICS はフィールドの名前を <cics:fieldnames> 要素の content 属性から取得します。 CICS がこの情報をサービス・ルーチンに渡す場合、サービス・ルーチンではこの名前付きフィールドを使用してエントリーのコンテンツを検索し、そのコンテンツを DFHATOMCONTENT コンテナー

で返すことができます。DFHATOMCONTENT コンテナーのデータは、エントリーの <atom:content> 要素で使用されます。

ATMP_CONTENT_TYPE_FLD
Atom エントリーのコンテンツのメディア・タイプ (application/xml や text など) が入っている、リソース・レコード内のフィールドの名前。 **ATMP_MTYPEOUT** パラメーターについては、完全な IANA メディア・タイプの代わりに、メディア・タイプ「text」、「html」、および「xhtml」が使用されます。メディア・タイプは、Atom エントリーの <atom:content> 要素の type 属性で指定されます。 CICS はフィールドの名前を <cics:fieldnames> 要素の content_type 属性から取得します。 CICS がこの情報をサービス・ルーチンに渡す場合、サービス・ルーチンではこの名前付きフィールドを使用してコンテンツのメディア・タイプを検索し、DFHATOMCONTENT コンテナーにあるコンテンツに適したマークアップを決定することができます。 Atom エントリーのコンテンツのメディア・タイプを格納するためのフィールドがリソース・レコードにない場合、Atom 構成ファイルにある <atom:content> 要素の type 属性で指定されたメディア・タイプが適用されます。 CICS は、このメディア・タイプを **ATMP_MTYPEOUT** パラメーターでサービス・ルーチンに渡します。

ATMP_CATEGORY_FLD
要求された Atom エントリーに適用されるカテゴリー用語が入っている、リソース・レコード内のフィールドの名前。 CICS はフィールドの名前を <cics:fieldnames> 要素の category 属性から取得します。 CICS がこの情報をサービス・ルーチンに渡す場合、サービス・ルーチンではこの名前付きフィールドを使用してカテゴリーを検索し、そのカテゴリーを DFHATOMCATEGORY コンテナーで返すことができます。DFHATOMCATEGORY コンテナーのデータは、エントリーの <atom:category> 要素で使用されます。

ATMP_AUTHOR_FLD
Atom エントリーの基本作成者の名前が入っている、リソース・レコード内のフィールドの名前。 CICS はフィールドの名前を <cics:fieldnames> 要素の author 属性から取得します。 CICS がこの情報をサービス・ルーチンに渡す場合、サービス・ルーチンではこの名前付きフィールドを使用して作成者の名前を検索し、その名前を DFHATOMAUTHOR コンテナーで返すことができます。 DFHATOMAUTHOR コンテナーのデータは、エントリーの <atom:name> 要素で使用されます。

ATMP_AUTHORURI_FLD
Atom エントリーの基本作成者に関連付けられた Web サイトの URI が入っている、リソース・レコード内のフィールドの名前。 CICS はフィールドの名前を <cics:fieldnames> 要素の authoruri 属性から取得します。 CICS がこの情報をサービス・ルーチンに渡す場合、サービス・ルーチンではこの名前付きフィールドを使用して URI を検索し、その URI を DFHATOMAUTHORURI コンテナーで返すことができます。DFHATOMAUTHORURI コンテナーのデータは、エントリーの <atom:uri> 要素で使用されます。

ATMP_EMAIL_FLD
Atom エントリーの基本作成者の E メール・アドレスが入っている、リソース・レコード内のフィールドの名前。 CICS はフィールドの名前を <cics:fieldnames> 要素の email 属性から取得します。 CICS がこの情報をサービス・ルーチンに渡す場合、サービス・ルーチンではこの名前付きフィールドを

使用して E メール・アドレスを検索し、そのカテゴリーを DFHATOMEMAIL コンテナーで返すことができます。 DFHATOMEMAIL コンテナーのデータは、エントリーの <atom:email> 要素で使用されます。

DFHATOMPARMS コンテナーの入出力パラメーター

サービス・ルーチンは、これらのパラメーターを使用して、返される Atom エントリーに関する情報を CICS に提供します。

DFHATOMPARMS コンテナー内の各入出力パラメーターは、領域を指すポインター、領域内のデータの現在の長さ、および領域の最大長で構成されるトリプルワードのアドレスです。

パラメーターを使用して情報を CICS に提供するため、サービス・ルーチンでは以下のいずれかを行うことができます。

- ポインターで示される領域に一部のデータをコピーし、その領域の現在の長さをデータの長さに設定する。 DFHATOMPARMS コンテナー内の入出力パラメーターの値を格納するストレージはユーザー・キーの中にあるため、サービス・ルーチンが EXECKEY(USER) で定義されている場合にはそこにアクセスできます。
- サービス・ルーチン独自のストレージで、かつプログラムの存続時間を超えて存続する必要のあるストレージ (TWA ストレージなど) にある何らかのデータにポインターを設定し、その領域の現在の長さをデータの長さに設定する。領域の指定最大長より長い値がある場合には、これを行う必要があるかもしれません。

サービス・ルーチンには特定のパラメーターに関連した情報がなく、CICS がそのパラメーターについて独自に指定するデフォルトを使用する必要がある場合、サービス・ルーチンでそのことを CICS に示す必要があります。これはデータの現在の長さをゼロに設定することにより行います。

ATMP_ATOMID

エントリーの Atom ID。 Atom ID は、Atom エントリーの固有 ID です。 Atom ID の形式について詳しくは、 269 ページの『Atom エントリーの Atom ID』を参照してください。

入力では、CICS はこの領域を使用して、エントリーのプロトタイプ Atom ID をサービス・ルーチンに送信します。プロトタイプ Atom ID の形式は、固有 ID を生成するためにタグ URI 形式と代替形式のどちらを使用するかに応じて、<cics:authority> 要素または <atom:id> 要素を Atom 構成ファイルに含めることによって決定します。 CICS は、Web クライアントによって提供される Atom ID を無視し、サービス・ルーチンには渡しません。

RFC 4287 に記載されている Atom 形式仕様では、Atom ID を、その Atom エントリーのリソース・レコードに格納することが推奨されています。 POST 要求でリソースに Atom ID を格納できる場合には、サービス・ルーチンでは、Atom エントリーのセレクター値 (**ATMP_SELECTOR** パラメーターで指定) を追加することによってプロトタイプ Atom ID を完成させてから、リソース・レコードの適切なフィールド (**ATMP_ID_FLD** パラメーターで指定) に完全な Atom ID を格納する必要があります。必要であれば、サービス・ルーチンでそのプロトタイプ Atom ID を無視し、その Atom エントリーに対する独自の有効な Atom ID に置き換えることができます。サービス・ルーチンは、入力として **ATMP_TAG_AUTHORITY** および **ATMP_TAG_DATE** パラメーターの値を使用して Atom ID を構成できます。

タグ URI 形式を使用している場合には、生成される Atom ID は、単一の Atom フィードを扱うユーザー作成サービス・ルーチンによって提供される Atom エントリーでは固有になりますが、ユーザー作成サービス・ルーチンが複数のフィードを提供する場合には固有にはならない点に注意してください。ユーザー作成サービス・ルーチンによって複数のフィードが提供される場合は、Atom ID 用に代替形式を選択するか、または Atom 構成ファイルの <cics:authority> 要素で各フィードについて異なる権限名または日付を指定してください。

出力では、サービス・ルーチンは以下のように **ATMP_ATOMID** パラメーターを使用する必要があります。

- Atom エントリーのリソース・レコードに完全な Atom ID を格納した場合、サービス・ルーチンは、リソース・レコードのフィールド (**ATMP_ID_FLD** パラメーターで指定) から完全な Atom ID とその Atom ID の長さを返す必要があります。
- リソースに Atom ID が格納されていない場合、サービス・ルーチンで **ATMP_ATOMID** パラメーターのデータの現在の長さをゼロに設定する必要があります。この場合、CICS によってセレクター値がプロトタイプ Atom ID に追加され、完全な Atom ID が生成されます。

ATMP_ETAGVAL

選択したリソース・レコードのエンティティー・タグ (または Etag) 値。サービス・ルーチンを使用してエンティティー・タグを作成するには、EXEC CICS BIF DIGEST コマンドを使用してレコードの SHA-1 ダイジェストを計算するか、または別の適した方式を使用して HTTP/1.1 プロトコル要件に準拠するエンティティー・タグを作成できます。

入力では、CICS は、**ATMP_ETAGVAL** パラメーターを使用して Atom エントリーのエンティティー・タグをサービス・ルーチンに提供します。 Web クライアントが PUT または DELETE 要求を行って Atom エントリーを編集する場合、CICS では、クライアントが要求時にエンティティー・タグを含む If-Match HTTP ヘッダーを提供することが必要です。 CICS がこのパラメーターを使用してエンティティー・タグを提供する場合、サービス・ルーチンでは、既存レコードのエンティティー・タグを計算し、Web クライアントのエンティティー・タグと比較する必要があります。タグが一致しない場合、エントリーは別のエージェントによって変更されているので、サービス・ルーチンでは、応答コード `atmp_resp_etag_no_match` を出してその要求を拒否する必要があります。 Web クライアントでは、エンティティー・タグの代わりにアスタリスクを提供することにより、他のエージェントによって変更されている場合でもエントリーを編集または削除するように指示できます。サービス・ルーチンではこの要求に応じる必要があります。

出力では、サービス・ルーチンは以下のように **ATMP_ETAGVAL** パラメーターを使用する必要があります。

- エンティティー・タグは Atom フィードのエントリーには使用されません。現在の Atom エントリーが Atom フィードの一部である場合、サービス・ルーチンはデータの現在の長さをゼロに設定する必要があります。
- CICS ではコレクション内のエントリーについてエンティティー・タグが必要になります。現在の Atom エントリーがコレクションの一部である場合、サ

ービス・ルーチンはエンティティー・タグを計算してそれを返す必要があります。エンティティー・タグはリソース・レコードに格納せず、必要になったときに計算してください。

ATMP_PUBLISHED
サービス・ルーチンは、このパラメーターを使用して、返された Atom エントリーが最初に公開された日時を返すことができます。「公開された日時」とは、データが最初に作成された、または最初に使用可能になった時点を指します。リソースにこのデータが格納されていない場合、サービス・ルーチンでそのことを示す必要があります。これはデータの現在の長さをゼロに設定することにより行います。このとき、CICS は現在時刻のデフォルトを提供します。サービス・ルーチンで日時スタンプを返す場合は、RFC 3339 形式 (XML dateTime データ型としても知られる) で指定する必要があります。この形式で日時スタンプを提供するには、EXEC CICS FORMATTIME コマンドを使用できます。あるいは、サービス・ルーチンで TRANSFORM DATATOXML コマンドを使用できる場合、CICS ABSTIME 値をこの形式で日時スタンプに変換することもできます。

ATMP_UPDATED
サービス・ルーチンは、このパラメーターを使用して、返された Atom エントリーが最後に更新された日時を返すことができます。「更新された日時」とは、データに重要な変更が加えられたと見なされる時点を指します。リソースにこのデータが格納されていない場合、サービス・ルーチンでそのことを示す必要があります。これはデータの現在の長さをゼロに設定することにより行います。このとき、CICS は現在時刻のデフォルトを提供します。サービス・ルーチンで日時スタンプを返す場合は、RFC 3339 形式で指定する必要があります。

ATMP_EDITED
サービス・ルーチンは、このパラメーターを使用して、返された Atom エントリーが最後に編集された日時を返すことができます。リソースにこのデータが格納されていない場合、サービス・ルーチンでそのことを示す必要があります。これはデータの現在の長さをゼロに設定することにより行います。このとき、CICS は現在時刻のデフォルトを提供します。サービス・ルーチンで日時スタンプを返す場合は、RFC 3339 形式で指定する必要があります。

ATMP_SELECTOR
サービス・ルーチンで提供する必要がある、Atom エントリーのセレクター値。セレクター値の説明については、264 ページの『Atom エントリーのセレクター値』を参照してください。

- クライアントがフィードに関する一般的な要求を行う場合、入力時に CICS は **ATMP_SELECTOR** パラメーターについてヌル値を送信します。このとき、入力パラメーター **ATMP_ATOMTYPE** の値は「feed」になります。サービス・ルーチンは、この値の組み合わせを受け取ったときに、以下の処置を行う必要があります。
  - フィードに追加された最新の Atom エントリーについてのデータを返す。
  - **ATMP_SELECTOR** パラメーターを使用して、そのエントリーを識別するセレクター値を返す。リソースでエントリーの Atom ID が保持されていない場合、 CICS ではこのセレクター値がエントリーの生成済み Atom ID で使用されます。
  - **ATMP_NEXTSEL** パラメーターを使用して、フィード内における次のエントリーのセレクター値を返す。

- CICS がクライアント要求を完了するためにフィードの 2 番目または後続の Atom エントリーを必要とする場合、またはクライアントが特定の範囲の Atom エントリーが入ったフィード文書を要求した場合、入力時に CICS は **ATMP_SELECTOR** パラメーターを使用してそのフィード文書内のいずれかの Atom エントリーのセレクター値を送信します。このとき、入力パラメーター **ATMP_ATOMTYPE** の値は「feed」になります。サービス・ルーチンは、この値の組み合わせを受け取ったときに、以下の処置を行う必要があります。
  - セレクター値で表される Atom エントリーについてのデータを返す。
  - CICS が **ATMP_SELECTOR** パラメーターに提供したデータまたは長さを変更しない。
  - **ATMP_NEXTSEL** パラメーターを使用して、フィード内における次のエントリーのセレクター値を返す。
- クライアントがフィードの特定のエントリーを要求する場合、CICS は **ATMP_SELECTOR** パラメーターを使用して、そのエントリーの URL から抽出したセレクター値を送信します。このとき、入力パラメーター **ATMP_ATOMTYPE** の値は「entry」になります。サービス・ルーチンは、この値の組み合わせを受け取ったときに、以下の処置を行う必要があります。
  - セレクター値で表される Atom エントリーについてのデータを返す。
  - CICS が **ATMP_SELECTOR** パラメーターに提供したデータまたは長さを変更しない。
  - **ATMP_NEXTSEL** パラメーターでは、データの現在の長さをゼロに設定することによりヌル値を返す。

注: コレクションでは、Atom フィードで "feed" 値が使用されるような状況の場合、 CICS は **ATMP_ATOMTYPE** パラメーターで "collection" 値を使用します。"entry" 値は、コレクションからのエントリーでも、Atom フィードからのエントリーでも同じです。

ATMP_NEXTSEL

使用可能な次の Atom エントリーがある場合、サービス・ルーチンでは、そのエントリーのセレクター値をこのパラメーターを使用して返す必要があります。265 ページの『Atom エントリーの順序』には、複数の Atom エントリーを返す順序が説明されています。

この値は、サービス・ルーチンで扱っているのがフィードかコレクションかに関係なく提供する必要があります。これが必要ないのは、クライアントが単一の特定のエントリーを要求する場合 (**ATMP_ATOMTYPE** の値が「entry」)、および Atom エントリーを提供するために使用可能なデータがこれ以上ない場合です。この値が必要ない場合、サービス・ルーチンでは、データの現在の長さをゼロに設定することによりこのパラメーターにヌル値を返す必要があります。

CICS は、サービス・ルーチンによって提供された値を使用して、サービス・ルーチンからさらに Atom エントリーを要求して Atom 文書を完成させます。Atom 文書が完成した場合、CICS はこの値を使って Atom 文書に <atom:link rel="next"> リンクを生成し、Web クライアントはそれを使ってフィードから Atom エントリーの次のウィンドウを取得するか、またはコレクションから Atom エントリーの次の部分リストを取得できます。

ATMP_PREVSEL
コレクション内に直前の Atom エントリーがある場合、そのコレクションを処理しているサービス・ルーチンでは、そのエントリーのセレクター値をこのパラメーターを使用して返す必要があります。 265 ページの『Atom エントリーの順序』には、複数の Atom エントリーを返す順序が説明されています。

この値は、**ATMP_ATOMTYPE** パラメーターの値が "collection" の場合に提供する必要があります。これが必要ないのは、クライアントが単一の特定のエントリーを要求する場合 (**ATMP_ATOMTYPE** の値が「entry」)、および直前の Atom エントリーがない場合です。この値が必要ない場合、サービス・ルーチンでは、データの現在の長さをゼロに設定することによりこのパラメーターにヌル値を返す必要があります。

Atom 文書が完成した場合、CICS はこの値を使ってサービス・ルーチンに対して一連の要求を実行して Atom 文書で <atom:link rel="previous"> リンクを生成し、Web クライアントはそれを使って、コレクションから Atom エントリーの次の部分リストを取得できます。

この値は、通常の Atom フィードを扱っているサービス・ルーチンからは必要ありません。 Web クライアントがフィードから Atom エントリーの前のウィンドウを取得するのに <atom:link rel="previous"> リンクが役立つ場合に、これを指定することができます。ただし、このリンクを生成する処理によって応答時間が増えるため、 Web クライアントがこの形式のナビゲーションを使用するように設定している場合にのみ、フィードにこの値を指定してください。

ATMP_FIRSTSEL
コレクションを処理しているサービス・ルーチンでは、そのコレクション内の最初の Atom エントリーのセレクター値をこのパラメーターを使用して返す必要があります。 264 ページの『Atom エントリーのセレクター値』には、複数の Atom エントリーを返す順序が説明されています。

この値は、**ATMP_ATOMTYPE** パラメーターの値が "collection" の場合に提供する必要があります。同じ Web クライアント要求に関連した後続の呼び出しでは、CICS は **ATMP_FIRSTSEL** パラメーターを使用してこのセレクター値をサービス・ルーチンに提供するため、サービス・ルーチンではこれを再度提供する必要はありません。

クライアントが単一の特定のエントリーを要求する場合 (**ATMP_ATOMTYPE** の値が「entry」)、この値は必要ありません。この値が必要ない場合、サービス・ルーチンでは、データの現在の長さをゼロに設定することによりこのパラメーターにヌル値を返す必要があります。

Atom 文書が完成した場合、CICS はこの値を使って Atom 文書で <atom:link rel="first"> リンクを生成し、Web クライアントはそれを使って、コレクションから Atom エントリーの最初 (最新) の部分リストを取得できます。

この値は、通常の Atom フィードを扱っているサービス・ルーチンからは必要ありません。 Web クライアントがフィードから Atom エントリーの最初 (最新) のウィンドウを取得するのに <atom:link rel="first"> リンクが役立つ場合に、これを指定することができます。 CICS は、このリンクを生成するための追加の処理を実行しません。

ATMP_LASTSEL
コレクションを処理しているサービス・ルーチンでは、そのコレクション内の最

後の Atom エントリーのセレクター値をこのパラメーターを使用して返す必要があります。 265 ページの『Atom エントリーの順序』には、複数の Atom エントリーを返す順序が説明されています。

この値は、**ATMP_ATOMTYPE** パラメーターの値が "collection" の場合に提供する必要があります。同じ Web クライアント要求に関連した後続の呼び出しでは、CICS は **ATMP_LASTSEL** パラメーターを使用してこのセレクター値をサービス・ルーチンに提供するため、サービス・ルーチンではこれを再度提供する必要はありません。

クライアントが単一の特定のエントリーを要求する場合 (**ATMP_ATOMTYPE** の値が「entry」)、この値は必要ありません。この値が必要ない場合、サービス・ルーチンでは、データの現在の長さをゼロに設定することによりこのパラメーターにヌル値を返す必要があります。

Atom 文書が完成した場合、CICS はこの値を使って Atom 文書で <atom:link rel="last"> リンクを生成し、Web クライアントはそれを使って、コレクションから Atom エントリーの最後 (最古) の部分リストを取得できます。 CICS は、単一のエントリー、つまりフィードの最後のエントリーのみを含むこの最後の部分リストを発行します。 Web クライアントは、<atom:link rel="previous"> リンクを使って、前のすべての部分リストを取得できます。

この値は、通常の Atom フィードを扱っているサービス・ルーチンからは必要ありません。 Web クライアントがフィードから最後 (最古) の Atom エントリーを取得するのに <atom:link rel="last"> リンクが役立つ場合に、これを指定することができます。 CICS は、このリンクを生成するための追加の処理を実行しません。

DFHATOMPARMS コンテナーの ATMP_OPTIONS パラメーター

ATMP_OPTIONS パラメーターは、64 個のオプション・ビットが含まれるダブルワードのアドレスです。これは、Atom エントリーのタイトルなどのデータを含むオプションのコンテナーがサービス・ルーチンによって提供されることを示すために使用されます。オプション・ストリングは **ATMP_OPTIONS_BITS** DSECT によってマップされます。

ATMP_OPTIONS が指すオプションのビットマップは、ATMP_OPTIONS_BITS と ATMP_OPTIONS_WORDS の 2 つの方法でマップされます。ATMP_OPTIONS_BITS は、ビット値を認識できる言語で使用するための一連のバイトおよびビット定義です。ATMP_OPTIONS_WORDS は、ビット値のコード化が容易ではない COBOL で使用するための、フルワード値のペアです。

ATMP_OPTIONS_BITS

ATMP_OPTIONS_BITS では、意味があるビット値は、バイト ATMP_OUTOPT_BYTE1 に入っています。以下を参照してください。

OPTTITLE
サービス・ルーチンは、DFHATOMTITLE コンテナーを使用して、エントリーのタイトルとして使用される文字ストリングを返します。

OPTSUMMA
サービス・ルーチンは、DFHATOMSUMMARY コンテナーを使用して、エントリーの要約として使用される文字ストリングを返します。

OPTAUTHOR
サービス・ルーチンは、DFHATOMAUTHOR コンテナーを使用して、エントリーの作成者の名前として使用される文字ストリングを返します。

OPTAUTHEML
サービス・ルーチンは、DFHATOMEMAIL コンテナーを使用して、エントリーの作成者の E メール・アドレスとして使用される文字ストリングを返します。

OPTAUTHURI
サービス・ルーチンは、DFHATOMAUTHORURI コンテナーを使用して、エントリーの作成者と関連付けられている Web サイトの URI として使用される文字ストリングを返します。

OPTCATEG
サービス・ルーチンは、DFHATOMCATEGORY コンテナーを使用して、エントリーのカテゴリー用語として使用される文字ストリングを返します。

ATMP_OPTIONS_WORDS

ATMP_OPTIONS_WORDS には、次の 2 つのフルワード値が含まれます。

ATMP_OPTIONS_IN
入力オプション値のフルワードです。これは使用されません。

ATMP_OPTIONS_OUT
出力オプション値を格納するためのフルワードです。ATMP_OUTOPT_BYTE1 内のビット値と等価のフルワード値は、コピーブック DFH0W2CO で指定されています。これらの値を必要に応じて組み合わせて追加することにより、適切なビットマップ値を生成できます。

コピーブック DFH0W2CO には、 ATMP_OPTIONS_OUT で使用される ATMP_OUTOPT_BYTE1 内のビットの値を表す次のような 2 進整数が含まれています。

OPTTITLE_NUM
OPTTITLE と等価

OPTSUMMA_NUM
OPTSUMMA と等価

OPTAUTHOR_NUM
OPTAUTHOR と等価

OPTAUTHEML_NUM
OPTAUTHEML と等価

OPTCATEG_NUM
OPTCATEG と等価

OPTAUTHURI_NUM
OPTAUTHURI と等価

DFHATOMPARMS コンテナーの ATMP_RESPONSE パラメーター

ATMP_RESPONSE パラメーターは、成功かエラーかを示す応答コードを CICS に返すために使用される、ダブルワードのアドレスです。サービス・ルーチンからエラーを示す応答コードが送信されると、CICS は適切なデフォルトの HTTP エラー応答

を生成して、Web クライアントに送信します。サービス・ルーチンは DFHHTTPSTATUS コンテナーを使用して、デフォルトのエラー応答をオーバーライドする代替の状況コードとテキストを返すことができます。

ダブルワードのアドレスは変更しないでください。最初のフルワード **ATMP_RESPONSE_CODE** には、応答コードが含まれます。これは、正常終了を示すゼロに初期設定されています。2 番目のフルワード **ATMP_REASON_CODE** も、ゼロに初期設定されています。このフルワードは将来使用するために予約されているので、サービス・ルーチンで変更することはできません。

ATMP_RESPONSE_CODE のシンボル値は、DFHW2CN シリーズのコピーブックで定義されています。値は以下のとおりです。

```
atmp_resp_normal            constant(0);  ! 通常の成功応答
atmp_resp_not_found         constant(4);  ! リソースが見つかりません
atmp_resp_not_auth          constant(8);  ! リソースが許可されていません
atmp_resp_disabled          constant(12); ! リソースが使用不可です
atmp_resp_already_exists    constant(16); ! リソースがすでに存在します
atmp_resp_etag_no_match     constant(20); ! If-Match 比較に失敗しました
atmp_resp_invalid_request   constant(24); ! 要求が無効です
atmp_resp_access_error      constant(32); ! その他のリソース・エラー
atmp_resp_conversion_failed constant(36); ! XML 変換エラー
```

このパラメーターが変更されずに返された場合、CICS は要求の正常終了を示す HTTP 応答を送信します。サービス・ルーチンからエラーを示す応答コードが送信されると、CICS は適切なデフォルトの HTTP エラー応答を生成して、Web クライアントに送信します。デフォルトの HTTP エラー応答は、以下のとおりです。

表 12. Atom フィードに関するサービス・ルーチンからのデフォルトの HTTP エラー応答

ATMP_RESPONSE_CODE value	**HTTP 状況コード**	**HTTP 状況テキスト**
atmp_resp_normal	200 (POST 要求の場合は 201)	OK (POST 要求の場合は Created)
atmp_resp_not_found	404	Not found
atmp_resp_not_auth	403	Forbidden
atmp_resp_disabled	503	Service unavailable
atmp_resp_already_exists	409	Duplicate resource
atmp_resp_etag_no_match	412	Precondition failed
atmp_resp_invalid_request	400	Invalid request
atmp_resp_access_error	500	Resource error
atmp_resp_conversion_failed	500	Resource error

エラーを返す場合、サービス・ルーチンでは DFHHTTPSTATUS コンテナーを使用して代替の状況コードとテキストを返すことにより、デフォルトの HTTP エラー応答を置き換えることができます。エラー応答に使用可能な状況コードのリストについては、423 ページの『付録 C. CICS Web サポートの HTTP 状況コード・リファレンス』を参照してください。要求が成功した場合のデフォルトの HTTP 応答 (応答コードがゼロのもの) はオーバーライドできません。

DFHATOMCONTENT コンテナー

DFHATOMCONTENT は DATATYPE(CHAR) のコンテナーであり、 Atom エントリーのコンテンツを提供するためにサービス・ルーチンで使用します。

このコンテナーは必須です。CICS はコンテナーのデータを、Atom エントリーの <atom:content> 要素として返します。

DFHATOMPARMS コンテナーのリソース・ハンドリング・パラメーターを使用している場合、**ATMP_CONTENT_FLD** パラメーターにはこのコンテナーのデータがあるリソース・レコード内のフィールドの名前と長さが含まれています。データは単一のフィールドまたはネストされたフィールド構造から使用できます。リソース・ハンドリング・パラメーターを使用していない場合、リソース・レコード全体をエントリーのコンテンツとして戻すか、あるいはリソース・レコードから適切なフィールドを選択してコンテンツをアセンブルするように、サービス・ルーチンをコーディングすることができます。

子要素を伴わないプレーン・テキストとしてコンテンツを提供することも、XML または別のマークアップ (HTML や XHTML など) を使ってデータをフォーマットすることも可能です。 DFHATOMPARMS コンテナーの **ATMP_MTYPEOUT** パラメーターには、Atom エントリーで予期されるコンテンツのメディア・タイプが含まれます。これは、Atom フィードの Atom 構成ファイルにある <atom:content> 要素の型属性で指定されています。 RFC 4287 にあるとおり、プレーン・テキストでは IANA メディア・タイプ「text/plain」の代わりにメディア・タイプ「text」、「text/html」の代わりに「html」、「application/xhtml+xml」の代わりに「xhtml」を使用します。 Atom 構成ファイルでメディア・タイプを指定しない場合、CICS は、デフォルトのメディア・タイプである「application/xml」をこのパラメーターに設定します。

個々の Atom エントリーのメディア・タイプをリソース・レコードに保管することもできます。リソース・ハンドリング・パラメーターを使用している場合、**ATMP_CONTENT_TYPE_FLD** パラメーターには、Atom エントリーのコンテンツのメディア・タイプを収容するフィールドの名前および長さが含まれます。

Atom エントリーのコンテンツ用の <content> タグ

対象のコンテンツが、DFHATOMPARMS コンテナーにある **ATMP_MTYPEOUT** パラメーターのサービス・ルーチンに CICS によって設定された所定のメディア・タイプではない 場合、DFHATOMPARMS コンテナー内でタグ <content> をコンテンツの先頭に、終了タグ </content> を末尾に含める必要があります。コンテンツがプレーン・テキスト以外であるなら、型属性を <content> タグに追加してコンテンツのメディア・タイプを指定する必要もあります。設定可能な型属性のいくつかを以下に示します。

- <content type="html"> は HTML を指定します。
- <content type="xhtml"> は XHTML を指定します。
- <content type="text/xml"> は通常、人間が読んで理解できる XML 文書に使用されるメディア・タイプです。

プレーン・テキストを提供する際に <content type="text"> を指定しても構いませんが、型属性を何も指定しない場合に、このメディア・タイプが指定されたものと

Atom 文書の受信側によって見なされます。コンテンツが他のフォーマットの場合、インターネットでそのフォーマットに通常使用する IANA メディア・タイプを指定します。メディア・タイプのリストについては、http://www.iana.org/assignments/media-types/ を参照してください。なお、CICS では非テキスト・メディア・タイプがサポートされないことに注意してください。

コンテンツが、DFHATOMPARMS コンテナーにある **ATMP_MTYPEOUT** パラメーターのサービス・ルーチンに CICS によって設定された所定のメディア・タイプである場合は、<content> タグおよび </content> タグを省略できます。この場合 CICS は、開始タグおよび終了タグを付けて、型属性を **ATMP_MTYPEOUT** パラメーターのメディア・タイプとして指定します。

Atom エントリーのコンテンツ用のマークアップ

プレーン・テキスト以外のフォーマットをコンテンツに使用する場合、RFC 4287 のセクション 4.1.3.3 にある処理情報、「*Atom 配信形式 (The Atom Syndication Format)*」(http://www.ietf.org/rfc/rfc4287.txt から入手可能) を参照してください。これらの規則は、マークアップの配置方法と、Atom 文書の受信者 (「Atom プロセッサー」として知られる) が使用マークアップの種類に応じてどのようにコンテンツを解釈し提供するかを説明します。特に、HTML マークアップはエスケープする必要があります。例えば、
 タグは
 と記述する必要があります。 CICS は使用するマークアップの妥当性検査を行いません。

リソース・レコードのフィールドから XML コンテンツを生成する場合、リソースに関連 XMLTRANSFORM リソースとの XML バインディングがあるなら、CICS 機能を使ってアプリケーション・データを XML に変換できます。リソースのデータの構造を記述する言語構造つまりコピーブックが DFHLS2SC プロシージャーでサポートされている高水準言語のいずれか、つまり、COBOL、C、C++、または PL/I の場合、XML バインディングを作成することができます。 言語構造からマッピングを生成で、その方法が説明されています。 CICS では、XML バインディングを指定する ATOMSERVICE リソース定義をインストールすると、XMLTRANSFORM リソースを動的に作成します。

XMLTRANSFORM リソースが使用可能な場合、CICS はその名前を DFHATOMPARMS コンテナー内の **ATMP_XMLTRANSFORM** パラメーターに提供します。 TRANSFORM DATATOXML コマンドに関する詳細と、データ・マッピング機能の使用方法については、 「*CICS アプリケーション・プログラミング・ガイド*」を参照してください。

コンテナー内の Atom エントリー・メタデータを戻す

Atom エントリー用のデータを含むリソースのレコードに、個別の Atom エントリー用にメタデータを提供するタイトルや要約などのフィールドが含まれる場合、サービス・ルーチンがこのデータを CICS に提供するために、 DFHATOMTITLE などのオプションのメタデータ・コンテナーを使用します。

このタスクについて

Atom エントリーの内容を保持する DFHATOMCONTENT コンテナーは、サービス・ルーチンから CICS に戻す必要のある唯一のコンテナーです。いくつかの他のオプションのコンテナーも、サービス・ルーチンが Atom エントリーのデータを含

むリソース内のレコードから取り出すことのできるメタデータを戻すために、使用可能です。例えば、元々 Atom フィードとして使用するためにセットアップされたのではない既存のファイルから Atom フィードを作成する場合など、リソース内のレコードにメタデータを保持するためのフィールドが含まれないことがあります。レコード内にメタデータがない場合はそれらのコンテナーを戻す必要がありませんが、場合によっては、Atom フィード用の Atom 構成ファイルをセットアップするときにデフォルトのメタデータを提供する必要があります。

サンプル・サービス・ルーチン DFH$W2S1 は、PUT CONTAINER コマンドを使用してオプションのコンテナーを作成する方法、およびリソース内のレコードから取り出したデータをそれらに取り込む方法を示しています。

手順

1. リソース内のレコードに Atom エントリー用の個別のタイトルが含まれる場合は、PUT CONTAINER コマンドを使用して、この Atom エントリーのタイトルが含まれる DFHATOMTITLE という名前のコンテナーを、DATATYPE(CHAR) を指定して作成します。 このコンテナーはオプションですが、このデータをサービス・ルーチンから提供しない場合は、デフォルトのタイトルを Atom 構成ファイルに指定する必要があります。コンテナーに何を入れるかについては、 303 ページの『DFHATOMTITLE コンテナー』を参照してください。
2. リソース内のエントリーに個別の要約が設定されている場合は、PUT CONTAINER コマンドを使用して、Atom エントリーの要約が含まれる DFHATOMSUMMARY という名前のコンテナーを、DATATYPE(CHAR) を指定して作成します。 このコンテナーはオプションですが、エントリーの内容がテキストまたは XML ではない場合、 Atom 仕様では要約が必要となります。コンテナーに何を入れるかについては、 303 ページの『DFHATOMSUMMARY コンテナー』を参照してください。
3. サービス・ルーチンを使用してエントリーの作成者の名前を提供する場合は、PUT CONTAINER コマンドを使用して、作成者の名前が含まれる DFHATOMAUTHOR という名前のコンテナーを、DATATYPE(CHAR) を指定して作成します。 このコンテナーはオプションですが、このデータをサービス・ルーチンから提供しない場合は、デフォルトの名前を Atom 構成ファイルに指定するか、または CICS のデフォルトを受け入れる必要があります。コンテナーに何を入れるかについては、 304 ページの『DFHATOMAUTHOR コンテナー』を参照してください。
4. サービス・ルーチンを使用してエントリーの作成者の E メール・アドレスおよび URI (Web サイト・アドレス) を提供する場合は、PUT CONTAINER コマンドを使用して、これらの項目のデータが含まれる DFHATOMEMAIL および DFHATOMAUTHORURI という名前のコンテナーを、DATATYPE(CHAR) を指定して作成します。 これらのコンテナーはオプションであり、いずれか 1 つまたは両方を提供することも、どちらも提供しないこともできます。これらのコンテナーに何を入れるかについては、 304 ページの『DFHATOMEMAIL コンテナー』および 305 ページの『DFHATOMURI コンテナー』を参照してください。
5. サービス・ルーチンを使用してエントリーのカテゴリーを提供する場合は、PUT CONTAINER コマンドを使用して、エントリーのカテゴリー用語が含まれる DFHATOMCATEGORY という名前のコンテナーを、DATATYPE(CHAR) を指定

して作成します。 このコンテナーはオプションです。コンテナーに何を入れるかについては、 305 ページの『DFHATOMCATEGORY コンテナー』を参照してください。

6. DFHATOMPARMS コンテナー内で、CICS に返されるオプションの各コンテナーについて、 **ATMP_OPTIONS_OUT** パラメーターから適切なオプション・ビットを設定します。 287 ページの『DFHATOMPARMS コンテナー』では、このパラメーターについて説明されています。サンプル・サービス・ルーチン DFH$W2S1 には、これらのオプション・ビットを設定する方法が示されています。

DFHATOMTITLE コンテナー

DFHATOMTITLE は DATATYPE(CHAR) のコンテナーであり、 Atom エントリーのタイトルを提供するためにサービス・ルーチンで使用できます。

このコンテナーはオプションです。CICS はコンテナーのデータを、 Atom エントリーの <atom:title> 要素として返します。

リソース・レコードで Atom エントリーの個々のタイトルを保持していない場合、Atom 構成ファイルを使用して、フィード内のすべてのエントリーについて同じタイトルを指定することもできます。 Atom エントリーにはタイトルが必要なため、CICS では Atom 構成ファイル内でデフォルトのタイトルが指定されている必要があります。適切なデフォルトのタイトルがない場合は、空白のデフォルトのタイトルを使用できますが、この場合は Atom 形式仕様に準拠するために DFHATOMTITLE コンテナーを使用してすべてのエントリー用のタイトルを提供する必要があります。

DFHATOMPARMS コンテナーのリソース・ハンドリング・パラメーターを使用している場合、**ATMP_TITLE_FLD** パラメーターにはこのコンテナーのデータがあるリソース・レコード内のフィールドの名前と長さが含まれています。

コンテナーのデータの周りにはタグを付けず、フォーマットするためのマークアップも使用しないでください。タイトルでは、CICS はプレーン・テキストのみをサポートします。

このコンテナーを CICS に提供する場合、DFHATOMPARMS コンテナー内の **ATMP_OPTIONS** パラメーターに **OPTTITLE** ビット値を設定します。

DFHATOMSUMMARY コンテナー

DFHATOMSUMMARY は DATATYPE(CHAR) のコンテナーであり、 Atom エントリーの要約を提供するためにサービス・ルーチンで使用できます。

このコンテナーはオプションです。CICS はコンテナーのデータを、 Atom エントリーの <atom:summary> 要素として返します。

リソース・レコードで Atom エントリーの個々の要約を保持していない場合、Atom 構成ファイルを使用して、フィード内のすべてのエントリーについて同じ要約を指定することもできます。 Atom 仕様では、エントリーの内容がテキストでも XML でもない場合に要約が必要です。そのため、これらのカテゴリーに当てはまらないコンテンツを提供することが予想される場合は、 DFHATOMSUMMARY コンテナーを提供するか、Atom 構成ファイルを使用してこのデータを提供する必要がありま

す。CICS では、非テキストおよび非 XML のエントリーについて要約が提供されているかどうかはチェックされません。この点に関する仕様への準拠については、ユーザーが責任を持つ必要があります。

DFHATOMPARMS コンテナーのリソース・ハンドリング・パラメーターを使用している場合、**ATMP_SUMMARY_FLD** パラメーターにはこのコンテナーのデータがあるリソース・レコード内のフィールドの名前と長さが含まれています。

コンテナーのデータの周りにはタグを付けず、フォーマットするためのマークアップも使用しないでください。

このコンテナーを CICS に提供する場合、DFHATOMPARMS コンテナー内の **ATMP_OPTIONS** パラメーターに **OPTSUMMA** ビット値を設定します。

DFHATOMAUTHOR コンテナー

DFHATOMAUTHOR は DATATYPE(CHAR) のコンテナーであり、 Atom エントリーの作成者の名前を提供するためにサービス・ルーチンで使用できます。

このコンテナーはオプションです。CICS はコンテナーのデータを、 Atom エントリーの <atom:author> 要素の <atom:name> 子要素として返します。

Atom 仕様では、各 Atom エントリーで作成者の名前が必要です。リソース内のレコードで個別の作成者の名前が提供されているフィールドがある場合は、このデータを提供します。提供されていない場合は、サービス・ルーチンでフィードのすべてのエントリーについて同じ作成者の名前を提供できます。または、フィード用の Atom 構成ファイルを使用して、フィードのすべてのエントリーについて単一の作成者の名前を提供することもできます。いかなる方法でも作成者の名前が提供されていない場合、CICS はデフォルトの作成者名 "CICS Transaction Server" が設定された応答を Web クライアントに送信します。

DFHATOMPARMS コンテナーのリソース・ハンドリング・パラメーターを使用している場合、**ATMP_AUTHOR_FLD** パラメーターにはこのコンテナーのデータがあるリソース・レコード内のフィールドの名前と長さが含まれています。

コンテナーのデータの周りにはタグを付けず、フォーマットするためのマークアップも使用しないでください。

このコンテナーを CICS に提供する場合、DFHATOMPARMS コンテナー内の **ATMP_OPTIONS** パラメーターに **OPTAUTHOR** ビット値を設定します。

DFHATOMEMAIL コンテナー

DFHATOMEMAIL は DATATYPE(CHAR) のコンテナーであり、 Atom エントリーの作成者の E メール・アドレスを提供するためにサービス・ルーチンで使用できます。

このコンテナーはオプションです。CICS はコンテナーのデータを、 Atom エントリーの <atom:author> 要素の <atom:email> 子要素として返します。

Atom 仕様はこのデータを必要としないため、データがない場合や配布したくない場合は省略することができます。リソースのレコードに個別の作成者の E メール・アドレスが含まれる場合、このデータを提供することができます。サービス・ルーチ

ンを通して、またはフィードの Atom 構成ファイル内で、フィードのすべてのエントリーについて単一の作成者の名前を提供する場合、 E メール・アドレスについても同様のことを行えます。

DFHATOMPARMS コンテナーのリソース・ハンドリング・パラメーターを使用している場合、**ATMP_EMAIL_FLD** パラメーターにはこのコンテナーのデータがあるリソース・レコード内のフィールドの名前と長さが含まれています。

コンテナーのデータの周りにはタグを付けず、フォーマットするためのマークアップも使用しないでください。

このコンテナーを CICS に提供する場合、DFHATOMPARMS コンテナー内の **ATMP_OPTIONS** パラメーターに **OPTEMAIL** ビット値を設定します。

DFHATOMURI コンテナー

DFHATOMURI は DATATYPE(CHAR) のコンテナーであり、 Atom エントリーの作成者に関係がある URI (Web サイト・アドレス) を提供するためにサービス・ルーチンで使用できます。

このコンテナーはオプションです。CICS はコンテナーのデータを、 Atom エントリーの <atom:author> 要素の <atom:uri> 子要素として返します。

作成者の E メール・アドレスについては、Atom 仕様はこのデータを必要としないため、データがない場合や配布したくない場合は省略することができます。リソースのレコードに個別の作成者の Web サイトが含まれる場合、このデータを提供することができます。すべての作成者が自分と同じ会社の人の場合、会社のホーム・ページの URI を提供することができます。サービス・ルーチンを通して、またはフィードの Atom 構成ファイル内で、フィードのすべてのエントリーにおいて単一の作成者の名前を提供する場合、 URI についても同様のことを行えます。

DFHATOMPARMS コンテナーのリソース・ハンドリング・パラメーターを使用している場合、**ATMP_AUTHORURI_FLD** パラメーターにはこのコンテナーのデータがあるリソース・レコード内のフィールドの名前と長さが含まれています。

コンテナーのデータの周りにはタグを付けず、フォーマットするためのマークアップも使用しないでください。

このコンテナーを CICS に提供する場合、DFHATOMPARMS コンテナー内の **ATMP_OPTIONS** パラメーターに **OPTAUTHURI** ビット値を設定します。

DFHATOMCATEGORY コンテナー

DFHATOMCATEGORY は DATATYPE(CHAR) のコンテナーであり、 Atom エントリーのカテゴリーを提供するためにサービス・ルーチンで使用できます。

このコンテナーはオプションです。CICS はコンテナーのデータを、 Atom エントリーの <atom:category> 要素の term 属性として返します。CICS では、<atom:category> 要素のオプションの scheme 属性と label 属性はサポートされていません。

リソース・レコードで Atom エントリーのカテゴリーを保持していない場合、メリットがあることが分かっているなら、Atom 構成ファイルを使用して、フィード内の

すべてのエントリーについて同じカテゴリーを指定することもできます。 Atom 仕様では、カテゴリーは必要ではありません。

DFHATOMPARMS コンテナーのリソース・ハンドリング・パラメーターを使用している場合、**ATMP_CATEGORY_FLD** パラメーターにはこのコンテナーのデータがあるリソース・レコード内のフィールドの名前と長さが含まれています。

コンテナーのデータの周りにはタグを付けず、フォーマットするためのマークアップも使用しないでください。

このコンテナーを CICS に提供する場合、DFHATOMPARMS コンテナー内の **ATMP_OPTIONS** パラメーターに **OPTCATEG** ビット値を設定します。

Atom フィード用の DFH$W2S1 C サンプル・サービス・ルーチン

サンプル・サービス・ルーチン DFH$W2S1 は、DFHATOMPARMS コンテナー内のパラメーターを読み取る方法、メタデータおよび内容コンテナー (DFHATOMTITLE や DFHATOMCONTENT など) を更新する方法、および DFHATOMPARMS コンテナーを更新して戻す方法を示す、C 言語のスケルトン・プログラムです。

EXECKEY(CICS) が DFH$W2S1 用に指定されている必要のある、適切な RDO 定義を作成してインストールする場合は、サンプル・サービス・ルーチンを実行することにより、Atom フィードへの GET 要求に対する応答としていくつかのデータをテスト目的で提供できます。出荷された状態のプログラムは、そのコードでセットアップされたデフォルトのデータだけを戻し、 POST、PUT、または DELETE 要求は処理しません。 Atom エントリーのデータを保持するリソースから必要なレコードを識別して、レコードから適切なフィールドを取り出す処理、または POST、PUT、DELETE 要求に対する応答としてレコードを更新する処理は、リソースの選択およびそのリソース内のレコードの構造に特定のものです。サービス・ルーチンとリソースとの相互作用を示す例として、 390 ページの『Atom フィード用の DFH0W2F1 COBOL サンプル・サービス・ルーチン』で DFH0W2F1 に関する説明を参照してください。 DFH0W2F1 は、CICS サンプル・ファイル FILEA と相互作用します。

DFH$W2S1 サンプル・サービス・ルーチンは、以下のタスクを実行します。

- DFHATOMPARMS のパラメーター・リストを含むコピーブック DFHW2APH、および応答コードの定数を含むコピーブック DFHW2CNH のための C ヘッダー・ファイルを組み込みます。
- 一時作業域 (TWA) に入れられる「outdata」という名前の構造を定義します。プログラムはこの構造を使用して DFHATOMPARMS パラメーター用の新しいデータを保管し、 CICS の新規データへのポインターを戻して、パラメーターの既存値を置き換えます。 DFHATOMPARMS パラメーターの新規データ用のストレージは、 CICS Atom 処理によって読み取り可能となるように、TWA 内になければなりません。

```
typedef struct
{
    char atomid??(50??);
```

```
    char published??(30??);
    ...
    char selector??(20??);
} outdata;
```

- DFHATOMPARMS パラメーターからすべての値を取り出す方法を示すプレースホルダーである、「indata」という名前の構造を定義します。実際には、サービス・ルーチンは各処理ステップを実行するために必要な値を取り出すことができます。

```
typedef struct
{
    char* resname;
    char* restype;
    char* atomtype;
    ...
    char* emailfld;
} indata;
```

- ヘルパー・メソッド getParameter を提供します。これは、DFHATOMPARMS 内のパラメーターからポインターおよび長さ情報を取得し、パラメーターの値をメモリーに読み込み、C ストリングとして使用できるようにヌル終了文字 (ゼロ・バイト) を追加して、新しいメモリー位置へポインターを戻すために使用できます。

```
char* getParameter(atmp_parameter* ParamPtr)
{
    char* DataPtr;
    int DataLen;
    DataLen  = ParamPtr->atmp_parameter_len;

    EXEC CICS GETMAIN SET(DataPtr) FLENGTH(DataLen+1)
                      INITIMG(0x00);
    if (DataLen != 0) {
        memcpy(DataPtr, ParamPtr->atmp_parameter_ptr, DataLen);
    }

    return DataPtr;
}
```

- ヘルパー・メソッド updatedAtomParam を提供します。これは、CICS が DFHATOMPARMS パラメーター用に送ったポインターおよび長さを、新規データを含む新しいストリング値へのポインターおよびそのストリングの長さで更新するために使用できます。

```
void updatedAtomParam(char *value, atmp_parameter *ParamPtr)
{
    ParamPtr->atmp_parameter_ptr = (unsigned long*)value;
    ParamPtr->atmp_parameter_len = strlen(value);
}
```

- ヘルパー・メソッド updateAtomContainer を提供します。これは、EXEC CICS PUT CONTAINER コマンドを使用して、メタデータおよび内容コンテナー (DFHATOMTITLE や DFHATOMCONTENT など) の 1 つを、 Atom エントリーのデータを保持するリソース内のレコードのフィールドから取得された Atom エントリー用のデータで更新して、対応するオプション・ビットを DFHATOMPARMS のポインター提供先に設定します。

```
void updateAtomContainer(char *cName, char *cChannel, char *value,
                         atmp_options_bits *optPtr)
{
    int len = strlen(value);

    EXEC CICS PUT CONTAINER(cName)
```

```
                CHANNEL(cChannel)
                FROM(value)
                FLENGTH(len)
                FROMCODEPAGE("IBM037");

    /* work out which option to set */
    if (strcmp(cName, "DFHATOMTITLE    ") == 0) {
        (*optPtr).atmp_options_outbit.atmp_outopt_byte1.opttitle = 1;
    }
    else if (strcmp(cName, "DFHATOMSUMMARY  ") == 0) {
        (*optPtr).atmp_options_outbit.atmp_outopt_byte1.optsumma = 1;
    }
    else if (strcmp(cName, "DFHATOMCATEGORY  ") == 0) {
        (*optPtr).atmp_options_outbit.atmp_outopt_byte1.optcateg = 1;
    }
    else if (strcmp(cName, "DFHATOMAUTHOR    ") == 0) {
        (*optPtr).atmp_options_outbit.atmp_outopt_byte1.optauthor = 1;
    }
    else if (strcmp(cName, "DFHATOMAUTHORURI ") == 0) {
        (*optPtr).atmp_options_outbit.atmp_outopt_byte1.optautheml = 1;
    }
    else if (strcmp(cName, "DFHATOMEMAIL     ") == 0) {
        (*optPtr).atmp_options_outbit.atmp_outopt_byte1.optauthuri = 1;
    }

}
```

- メインプログラムで、メタデータおよび内容コンテナー (DFHATOMTITLE や DFHATOMCONTENT など) に入れられるデータのストレージなどの変数、および「outdata」と「indata」構造をセットアップします。
- EXEC CICS ADDRESS TWA コマンドを発行して、「outdata」構造のストレージを TWA 内にセットアップしてから、この構造にデフォルト値を取り込みます。これらのデフォルト値は、サンプル・サービス・ルーチンを提供されたままの状態で実行した場合に戻されます。 **ATMP_NEXTSEL** パラメーターの値を 0 に設定することにより、サンプル・サービス・ルーチンは Atom フィード文書のウィンドウとして指定されたエントリーの数が取得されるまで繰り返し、CICS 要求をこのデフォルト・データから構成される同じ Atom エントリーにします。実データを提供するサービス・ルーチンをコーディングするときは、要求メソッドによってはいくつかのデータ項目を変更しないこともあるので、デフォルト値を使用しないでください。

```
 EXEC CICS ADDRESS TWA(outData);

    strcpy(outData->atomid,
           "cics-atomservice-sample:2009-02-02T00:00:00Z");
    strcpy(outData->published, "2009-02-02T00:00:00Z");
    strcpy(outData->updated, "2009-02-20T14:54:34Z");
    strcpy(outData->edited, "2009-02-20T14:54:34Z");
    strcpy(outData->etagval, "");
    strcpy(outData->selector, "");
    strcpy(outData->nextsel, "0");
    strcpy(outData->prevsel, "");
    strcpy(outData->firstsel, "");
    strcpy(outData->lastsel, "");
```

- EXEC CICS GETMAIN コマンドを発行して、メタデータおよび内容コンテナー (DFHATOMTITLE や DFHATOMCONTENT など) の値を保持するストレージを取得し、これらのコンテナーにデフォルトのデータを取り込みます。コンテナー内のデータは、TWA にある必要はありません。 DFHATOMCONTENT コンテナー内にある Atom エントリー用の内容は、内容のタイプを示すタイプ属性を指定

できる <atom:content> の開始タグと終了タグで囲まれている必要があります。サンプル・サービス・ルーチンで行われているように、<atom:content> を省略して <content> だけにすることもできます。

```
EXEC CICS GETMAIN SET(AtomContent)   FLENGTH(512) INITIMG(0x00);
EXEC CICS GETMAIN SET(AtomTitle)     FLENGTH(50) INITIMG(0x00);
EXEC CICS GETMAIN SET(AtomSummary)   FLENGTH(100) INITIMG(0x00);
EXEC CICS GETMAIN SET(AtomCategory)  FLENGTH(30) INITIMG(0x00);
EXEC CICS GETMAIN SET(AtomAuthor)    FLENGTH(40) INITIMG(0x00);
EXEC CICS GETMAIN SET(AtomAuthorUri) FLENGTH(256) INITIMG(0x00);
EXEC CICS GETMAIN SET(AtomEmail)     FLENGTH(256) INITIMG(0x00);

strcpy(AtomContent,
       "<content type='text/xml'>Hello world</content>");
strcpy(AtomTitle,     "Sample Service Routine Entry");
strcpy(AtomSummary,
       "This is an entry from the sample service routine");
strcpy(AtomCategory,  "sample-entry");
strcpy(AtomAuthor,    "CICS Sample service routine");
strcpy(AtomAuthorUri, "");
strcpy(AtomEmail,     "");
```

- EXEC CICS ASSIGN CHANNEL コマンドを発行して、後の GET および PUT CONTAINER コマンドで使用するために現行チャネルの名前を取得します。

```
EXEC CICS ASSIGN CHANNEL(cChannel);
```

- CICS が Web クライアントの POST または PUT 要求のボディをサービス・ルーチンに渡すために使用する、DFHREQUEST コンテナーの存在を検査します。最初の EXEC CICS GET CONTAINER コマンドはコンテナーの長さを要求して、ゼロ以外の応答コード (コンテナーが存在しないかまたは問題があることを示す) が戻された場合には続行しません。要求ボディが渡された場合、サンプル・サービス・ルーチンはその内容をストレージに読み取ります。

```
EXEC CICS GET CONTAINER("DFHREQUEST      ")
             CHANNEL(cChannel)
             NODATA
             FLENGTH(DataLen)
             RESP(Resp) RESP2(Resp2);

if(Resp != 0 || Resp2 != 0)
{
    DataLen = -1;
}

/* If we have a request body then get it */
if(DataLen != -1)
{
    DataLen++;
    EXEC CICS GETMAIN SET(RequestBody) FLENGTH(DataLen)
                      INITIMG(0x00);

    EXEC CICS GET CONTAINER("DFHREQUEST      ")
                      INTO(RequestBody)
                      FLENGTH(DataLen);
}
```

- EXEC CICS GET CONTAINER コマンドを発行して、ポインター ParamList を DFHATOMPARMS パラメーターのデータの先頭に設定します。

```
EXEC CICS GET CONTAINER("DFHATOMPARMS    ")
             SET(ParamListData)
             FLENGTH(DataLen);

ParamList = (atmp_parameter_list*)ParamListData;
```

- DFHATOMPARMS パラメーターのデータ全体を対象に、ヘルパー・メソッド getParameter を使用して各パラメーターの値を取得してから、それに対するポインターを「inData」構造の中に設定します。

```
optPtr = (atmp_options_bits*)ParamList->atmp_options;

ParamPtr = (atmp_parameter*)ParamList->atmp_resname;
inData.resname = getParameter(ParamPtr);

ParamPtr = (atmp_parameter*)ParamList->atmp_restype;
inData.restype = getParameter(ParamPtr);

...

ParamPtr = (atmp_parameter*)ParamList->atmp_email_fld;
inData.emailfld = getParameter(ParamPtr);
```

- DFHATOMPARMS パラメーターおよび DFHREQUEST コンテナーの情報に基づいて、 Atom エントリーのデータを保持するリソースと対話するコードを提供する必要のある場所を示します。サービス・ルーチンはリソース内で必要なレコードを見つけて、 GET 要求に対する応答とてレコードから適切なフィールドを取り出すか、または POST、PUT、または DELETE 要求に対する応答としてレコードを更新します。 **ATMP_HTTPMETH** パラメーターの値は、要求メソッドが何であるかをサービス・ルーチンに知らせます。
- リソース・レコードから取得したデータを、outData 構造内のポインターで指定されるメモリー位置と、メタデータおよび内容コンテナー (DFHATOMTITLE や DFHATOMCONTENT など) のストレージに入れる方法を示します。この例は、サービス・ルーチン用の入力で **ATMP_SELECTOR** パラメーターがヌルの場合に、現行レコードのためにセレクター値を追加する方法を示しています。

```
 *  strcpy(outData->atomid,    ".....");
 *  strcpy(outData->published, ".....");
 *  strcpy(outData->updated,   ".....");
 *  strcpy(outData->edited,    ".....");
 *  strcpy(outData->etagval,   ".....");
 *  strcpy(outData->nextsel,   ".....");
 *  strcpy(outData->prevsel,   ".....");
 *  strcpy(outData->firstsel,  ".....");
 *  strcpy(outData->lastsel,   ".....");
 *
 *  if (strlen(inData->selector) == 0) {
 *      strcpy(outData->selector, ".....");
 *  }
 *
 *  strcpy(AtomContent,   "....");
 *  strcpy(AtomTitle,     "....");
 *  strcpy(AtomSummary,   "....");
 *  strcpy(AtomCategory,  "....");
 *  strcpy(AtomAuthor,    "....");
 *  strcpy(AtomAuthorUri, "....");
 *  strcpy(AtomEmail,     "....");
```

- updatedAtomParam ヘルパー・メソッドを使用して、各 DFHATOMPARMS パラメーターのストレージを、データの新規アイテムのポインターおよび長さによって更新します。

```
updatedAtomParam(outData->atomid,
                 (atmp_parameter*)ParamList->atmp_atomid);
updatedAtomParam(outData->published,
                 (atmp_parameter*)ParamList->atmp_published);
updatedAtomParam(outData->updated,
                 (atmp_parameter*)ParamList->atmp_updated);
updatedAtomParam(outData->edited,
```

```
                    (atmp_parameter*)ParamList->atmp_edited);
    updatedAtomParam(outData->etagval,
                    (atmp_parameter*)ParamList->atmp_etagval);
    updatedAtomParam(outData->nextsel,
                    (atmp_parameter*)ParamList->atmp_nextsel);
    updatedAtomParam(outData->prevsel,
                    (atmp_parameter*)ParamList->atmp_prevsel);
    updatedAtomParam(outData->firstsel,
                    (atmp_parameter*)ParamList->atmp_firstsel);
    updatedAtomParam(outData->lastsel,
                    (atmp_parameter*)ParamList->atmp_lastsel);

    if (strlen(outData->selector) != 0) {
        updatedAtomParam(outData->selector,
                        (atmp_parameter*)ParamList->atmp_selector);
    }
```

- updateAtomContainer ヘルパー・メソッドを使用して、新規データをメタデータおよび内容コンテナーに追加し、各コンテナーの存在を示すようにオプション・ビットを設定します。

```
updateAtomContainer("DFHATOMTITLE    ",cChannel,AtomTitle,optPtr);
updateAtomContainer("DFHATOMSUMMARY  ",cChannel,AtomSummary,optPtr);
updateAtomContainer("DFHATOMCATEGORY ",cChannel,AtomCategory,optPtr);
updateAtomContainer("DFHATOMAUTHOR   ",cChannel,AtomAuthor,optPtr);
updateAtomContainer("DFHATOMAUTHORURI",cChannel,AtomAuthorUri,optPtr);
updateAtomContainer("DFHATOMEMAIL    ",cChannel,AtomEmail,optPtr);
updateAtomContainer("DFHATOMCONTENT  ",cChannel,AtomContent,optPtr);
```

- DFHW2CNH コピーブックで定義された選択から、応答コードを提供します。理由コードは現在は CICS によって無視され、将来の使用のために予約されているので、その内容は任意のものです。

```
ResponsePtr = (atmp_responses*)ParamList->atmp_response;
ResponsePtr->atmp_response_code = ATMP_RESP_NORMAL;
/*
 * ResponsePtr->atmp_response_code = ATMP_RESP_NOT_FOUND;
 * ResponsePtr->atmp_response_code = ATMP_RESP_NOT_AUTH;
 * ResponsePtr->atmp_response_code = ATMP_RESP_DISABLED;
 * ResponsePtr->atmp_response_code = ATMP_RESP_ALREADY_EXISTS;
 * ResponsePtr->atmp_response_code = ATMP_RESP_ACCESS_ERROR;
 */
ResponsePtr->atmp_reason_code = 0;
```

- メインプログラムが終了すると、制御を CICS に自動的に戻します。 CICS Atom 処理は、メタデータおよび内容コンテナーからのデータと、サービス・ルーチンが新しいポインターおよび長さを提供する DFHATOMPARMS パラメーターの TWA 内のデータを使用して、Atom エントリーを構成します。その後 CICS は、Atom フィード文書を構成するエントリーのウィンドウが完了するまで、同じ方法でサービス・ルーチンから追加の Atom エントリーを要求します。

CICS Explorer を使用して Atom フィード用の XML バインディングを作成する

CICS Explorer のファイル・インポート・ウィザードを使用してソース言語ファイルを CICS バンドル・プロジェクトに組み入れ、Atom フィード用の XML バインディングと関連スキーマを作成します。その後このバンドル・プロジェクトを CICS システムにデプロイして、XMLTRANSFORM リソースを作成できます。

始める前に

Atom フィードのデータ・ソース内の各レコードを定義した言語構造が必要です。例えば、VSAM ファイル内に保管されているレコードの構造を定義した COBOL コピーブックが考えられます。

このタスクについて

ウィザードは、COBOL、C/C++、または PL/I 言語構造を使用して、XML バインディングと関連スキーマを生成できます。ウィザードは、XML バインディングとスキーマをバンドル・プロジェクトにインポートします。

手順

1. オプション: XML バインディングおよび関連スキーマ用のプロジェクトがまだない場合は、CICS Explorer で新規 CICS バンドル・プロジェクトを作成します。
   a. リソース・パースペクティブに切り替えるには、メインメニュー・バーで、「**ウィンドウ (Window)**」 > 「**パースペクティブを開く (Open Perspective)**」 > 「**その他 (Other)**」をクリックします。「パースペクティブを開く (Open Perspective)」ウィンドウから「**リソース (Resource)**」を選択し、「**OK**」をクリックします。
   b. メインメニュー・バーで、「**エクスプローラー (Explorer)**」 > 「**新規作成ウィザード (New Wizards)**」 > 「**その他 (Other)**」 > 「**CICS リソース (CICS Resources)**」 > 「**CICS バンドル・プロジェクト (CICS Bundle project)**」をクリックします。 バンドル・プロジェクト・ウィザードが開きます。
   c. 「**プロジェクト名 (Project name)**」フィールドに、新規プロジェクトの名前を入力します。
   d. 「**完了 (Finish)**」をクリックします。

   「プロジェクト・エクスプローラー (Project Explorer)」ビューに新規 CICS バンドル・プロジェクトがリストされます。
2. リソース・パースペクティブの「プロジェクト・エクスプローラー (Project Explorer)」ビューでバンドル・プロジェクトを右クリックし、「**インポート (Import)**」をクリックします。
3. 「一般 (General)」フォルダーを展開し、「**XML 変換ソースのインポート (Import XML Transform Source)**」をクリックします。ファイル・インポート・ウィザードが開きます。
4. 「**参照 (Browse)**」をクリックして、XML 変換に関連付けるソース言語ファイルを見つけます。次に、使用するファイルにナビゲートします。このファイルとして、以下のファイル・タイプのいずれかが可能です。

ファイル・タイプ	プログラミング言語
.c	C/C++ ソース・コード・ファイル
.cpp	C/C++ ソース・コード・ファイル
.h	C/C++ ヘッダー・ファイル
.cbl	COBOL ソース・コード

ファイル・タイプ	プログラミング言語
.cob	COBOL ソース・コード
.cpy	COBOL コピーブック・データ・ファイル
.pli	PL/I データ記述ファイル

5. 「**親フォルダーの入力または選択 (Enter or select the parent folder)**」フィールドで、XML バインディングとスキーマを組み入れるバンドル・プロジェクトの名前を上書き入力するか、リストからバンドル・プロジェクトを選択します。
6. 「**バンドル内の XML 変換ソース名 (XML Transform source name in bundle)**」フィールドで、XML バインディングと関連スキーマの名前を指定します。このフィールドには、先ほど指定したソース言語ファイルの名前が取り込まれますが、異なる名前をこのフィールドに上書き入力できます。ただし、ファイル拡張子は維持するようにします。
7. マッピング・レベルを設定します。マッピング・レベルは、ソース言語ファイルと XML スキーマの間で変換される情報量を定義するためのものです。現在使用可能で最も高度なマッピングを活用するには、マッピング・レベルを最新レベルに設定します。マッピング・レベルについて詳しくは、CICS 支援機能用のマッピング・レベルを参照してください。
8. 「**完了 (Finish)**」をクリックします。XML バインディング (.xsdbind) とスキーマ (.xsd) が作成され、バンドル・プロジェクトに組み入れられます。これらのファイルは、バンドル・プロジェクトの xsdbind フォルダーに入れられます。ソース言語ファイルのコピーも参照用にバンドル・プロジェクトに組み入れられ、META-INF フォルダー内のマニフェスト・ファイル (cics.xml) が更新されて新規 XMLTRANSFORM リソースが反映されます。

タスクの結果

XML バインディングとスキーマがバンドル・プロジェクトに組み入れられます。

次のタスク

バンドル・プロジェクトを CICS 領域にデプロイする必要があります。バンドル・プロジェクトをデプロイすると、XMLTRANSFORM リソースが CICS によって自動的に生成されます。バンドル・プロジェクトのデプロイについて詳しくは、345 ページの『CICS Explorer からの CICS バンドル・プロジェクトのデプロイ』を参照してください。

第 20 章 Atom フィード用の CICS 定義のセットアップ

CICS から Atom フィードを提供するには、CICS Explorer を使用してバンドル・プロジェクト内に Atom 構成ファイルを作成し、プロジェクトを CICS 領域にデプロイします。BUNDLE リソースをインストールすると、ATOMSERVICE および URIMAP リソースが CICS によって自動的に作成されます。Atom フィードの別名トランザクションをセットアップすることもできます。

始める前に

CICS から Atom フィードを提供するためには、CICS Web サポートの基本コンポーネントを構成しておく必要があります (まだ構成していない場合)。65 ページの『第 4 章 CICS Web サポートのコンポーネントの構成』で、CICS Web サポートのこれらの基本コンポーネントの構成方法について説明しています。

CICS 領域には、 Web クライアントが Atom フィードの HTTP 要求を行うポートの TCPIPSERVICE リソース定義が必要です。 109 ページの『CICS Web サポートの TCPIPSERVICE リソース定義の作成』では、TCPIPSERVICE リソースの定義方法が説明されています。

Atom エントリーのデータを提供するリソースを選択し、このデータの配信をサポートする XML バインディングかサービス・ルーチンのいずれかを作成する必要もあります。 277 ページの『第 19 章 Atom エントリー・データを提供するリソースのセットアップ』で、XML バインディングまたはサービス・ルーチンの作成方法について説明しています。

このタスクについて

選択したリソースのデータで構成され、作成済みの XML バインディングまたはサービス・ルーチンを使用して配信される Atom フィードを提供するように CICS 定義をセットアップするには、以下の手順に示すタスクを実行します。

構成ファイルには EBCDIC を使用するようにしてください。また、「encoding=」が指定されていない限り、コード・ページ 1047 で構成ファイルが提供されるものとします。

手順

1. CICS Explorer の Atom 構成ウィザードを使用して、バンドル・プロジェクト内に基本 Atom 構成ファイルを作成します。 詳細については、316 ページの『Atom 構成ファイルの作成』を参照してください。
2. CICS Explorer の Atom 構成エディターを使用して Atom 構成ファイルを編集し、既存のデータを修正するかさらに XML 要素を追加します。 詳細については、320 ページの『Atom 構成ファイルの編集』を参照してください。
3. オプション: Atom フィードの別名トランザクションを作成します。 詳細については、344 ページの『Atom フィード用の別名トランザクションの作成』を参照してください。

4. CICS 領域にバンドル・プロジェクトをデプロイします。 詳細については、 345 ページの『CICS Explorer からの CICS バンドル・プロジェクトのデプロイ』を参照してください。

タスクの結果

Atom フィード BUNDLE リソースが CICS 領域に定義されてインストールされます。ATOMSERVICE、XMLTRANSFORM、および URIMAP リソースが、CICS によって自動的に作成されます。BUNDLE リソースを使用可能にすると、Web クライアントが Atom フィードにアクセスできるようになります。

次のタスク

無料または市販の多数の Web クライアント・アプリケーションは、Atom フィードを要求、受信、および表示することができます。これには、専用のフィード・リーダーのほか、マッシュアップ作成用アプリケーションなど、追加機能を提供するアプリケーションがあります。使用するアプリケーションで Atom 形式がサポートされていることを確認してください。独自の Web クライアント・アプリケーションを作成して、Atom フィードのデータに対する GET 要求を行うこともできます。

関連タスク

371 ページの『Atom フィードまたはコレクションに対する GET 要求の発行』
Web クライアントは、コレクションの URL に対して HTTP GET 要求を出すことにより Atom フィードまたはコレクションの中の既存の Atom エントリーのリストを取得できます。 または、Atom エントリーの URL に対して HTTP GET 要求を出すことにより、Atom フィードまたはコレクションから個々の Atom エントリーを取得できます。

Atom 構成ファイルの作成

Atom 構成ファイルを作成するには、CICS Explorer™ の Atom 構成ウィザードを使用します。このファイルは、Atom フィードのメタデータを提供するためのさまざまな XML 要素から成ります。新規作成ウィザードを使用して基本詳細情報を入力してファイルを作成した後、エディターを使用してさらに情報を追加します。

始める前に

z/OS UNIX 上に XMLTRANSFORM リソースが必要です。XMLTRANSFORM リソースは、アプリケーション・データを XML に変換するための XML バインディングおよびスキーマを定義したものです。 XMLTRANSFORM リソースは既に存在する場合があります。そうでなければ、CICS Explorer のファイル・インポート・ウィザードを使用して作成することができます。ファイル・インポート・ウィザードの使用法について詳しくは、 311 ページの『CICS Explorer を使用して Atom フィード用の XML バインディングを作成する』を参照してください。

このタスクについて

Atom 構成ファイルを作成する場合は、バンドル・プロジェクトを作成し、次に Atom 構成ウィザードを使用して Atom フィードの基本詳細情報を入力します。入力した詳細情報に関連付けられた XML 要素が含まれる Atom 構成ファイルが、ウィザードによって作成されます。

このタスクで作成される Atom 構成ファイル例は、サンプル Atom 構成ファイル `filea.xml` に基づいています。 CICS Transaction Server をインストールすると、ディレクトリー `/usr/lpp/cicsts/cicsts42/samples/web2.0/atom` にサンプル Atom 構成ファイルがインストールされます (`/usr/lpp/cicsts/cicsts42` は、z/OS UNIX 上の CICS ファイルのデフォルトのインストール・ディレクトリー)。

手順

1. オプション: Atom 構成ファイル用のプロジェクトがまだない場合は、CICS Explorer で新規 CICS バンドル・プロジェクトを作成します。
   a. リソース・パースペクティブにしておきます。リソース・パースペクティブに切り替えるには、メインメニュー・バーで、「**ウィンドウ (Window)**」 > 「**パースペクティブを開く (Open Perspective)**」 > 「**その他 (Other)**」をクリックします。「パースペクティブを開く (Open Perspective)」ウィンドウから「**リソース (Resource)**」を選択し、「**OK**」をクリックします。
   b. メインメニュー・バーで、「**エクスプローラー (Explorer)**」 > 「**新規作成ウィザード (New Wizards)**」 > 「**その他 (Other)**」 > 「**CICS リソース (CICS Resources)**」 > 「**CICS バンドル・プロジェクト (CICS Bundle project)**」をクリックします。 バンドル・プロジェクト・ウィザードが開きます。
   c. 「**プロジェクト名 (Project name)**」フィールドに、新規プロジェクトの名前を入力します。
   d. 「**完了 (Finish)**」をクリックします。

   「プロジェクト・エクスプローラー (Project Explorer)」ビューに新規 CICS バンドル・プロジェクトがリストされます。
2. リソース・パースペクティブの「プロジェクト・エクスプローラー (Project Explorer)」ビューで、新規 Atom 構成ファイルを含めるバンドル・プロジェクトをクリックします。
3. 以下のいずれかの方法で、新規作成ウィザードを開きます。
   - CICS Explorer ツールバーにある「 **新規作成 (New)**」ウィザード・アイコン  の下矢印をクリックし、「**その他 (Other)**」をクリックします。「CICS リソース (CICS Resources)」フォルダーを展開し、「**Atom 構成ファイル (Atom Configuration file)**」をクリックします。
   - 「**エクスプローラー (Explorer)**」>「**新規作成ウィザード (New Wizards)**」をクリックし、「**その他 (Other)**」をクリックします。「CICS リソース (CICS Resources)」フォルダーを展開し、「**Atom 構成ファイル (Atom Configuration file)**」をクリックします。
   - 「プロジェクト・エクスプローラー (Project Explorer)」ビューでプロジェクトを右クリックし、「**新規作成 (New)**」>「**その他 (Other)**」をクリックします。「CICS リソース (CICS Resources)」フォルダーを展開し、「**Atom 構成ファイル (Atom Configuration file)**」をクリックします。
4. ウィザードのフィールドに入力します。次の表は、Atom 構成ウィザードのフィールドを示しています。 319 ページの図 18 は、サンプル Atom 構成ファイル `filea.xml` 内の XML 要素に基づいてすべてのフィールドが入力された CICS Explorer の Atom 構成ウィザードを示しています。

表 13. *Atom* ウィザードのフィールド

フィールド	説明
親フォルダー (Parent Folder)	Atom 構成ファイルが含まれるプロジェクトの名前。このフィールドに上書き入力してプロジェクトを変更できます。
ファイル名 (File name)	作成する Atom 構成ファイルの名前。
サービス・タイプ (Service Type)	コレクションとフィードの 2 つのサービス・タイプがあります。コレクションは、Web クライアントが編集したり削除したりできるデータを提供します。フィードは、読み取り専用データを Web クライアントに提供します。
フィード・タイトル (Feed Title)	Web クライアントで表示される Atom フィードのタイトル。 title では、CICS はプレーン・テキストのみをサポートします。
エントリー・タイトル (Entry Title)	Atom エントリーのタイトル。CICS は、プレーン・テキストのタイトルのみサポートします。 Atom エントリーのタイトルを CICS リソースが提供するとしても、エントリー・タイトルを指定する必要があります。どの Atom エントリーにも適応可能なデフォルト・タイトルを使用してください。
リソース・タイプ (Resource Type)	フィードに使用するリソースのタイプ。
リソース名 (Resource Name)	フィードに使用するリソースの名前。
デフォルト URI (Default URI)	Atom フィードの部分 URI。 URI のスキームとホスト構成要素を省略して、パス構成要素のみを指定できます。パスは、Atom フィードに固有でなければなりません。
XML 変換名 (XML Transform Name)	データ・ソースに関連付けられた XML バインディングを指す、CICS 領域内の既存の XMLTRANSFORM リソースの名前。

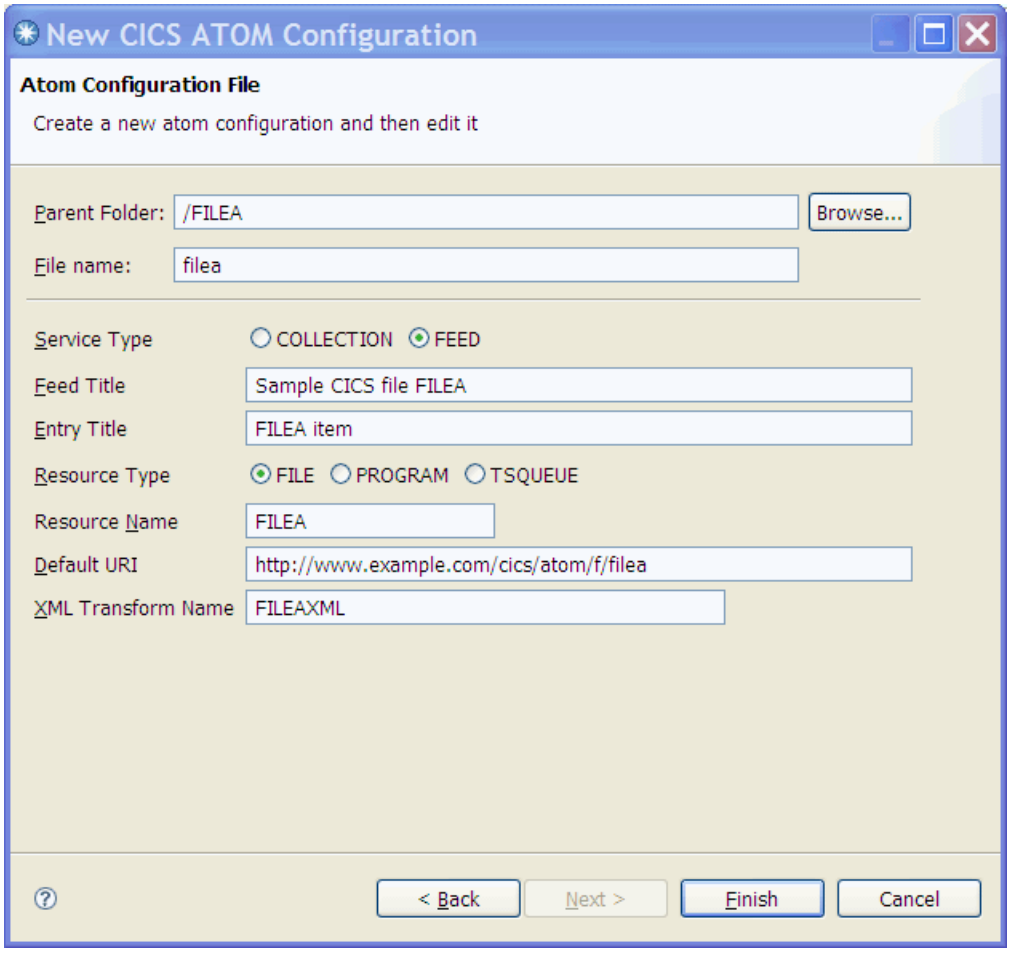


図 18. サンプル Atom 構成ファイル filea.xml 内の XML 要素に基づいてすべてのフィールドが入力された CICS Explorer の Atom 構成ウィザード

5. 「**完了 (Finish)**」をクリックします。

タスクの結果

Atom 構成ファイルが作成されます。このファイルの属性が Atom 構成エディターに表示されます。

次のタスク

Atom 構成エディターを使用して、Atom 構成ファイルを編集することや、ファイルにさらに XML 要素を追加することが可能です。 Atom 構成エディターは基本エディターであるため、このエディターを使用してすべての XML 要素を編集したり追加したりできるわけではありません。詳しくは、320 ページの『Atom 構成ファイルの編集』を参照してください。Atom 構成エディターでサポートされない XML 要素を追加するには、代わりの XML エディターを使用して、Atom 構成ファイルを直接編集する必要があります。

これ以上の編集が不要であれば、Atom 構成ファイルが含まれるバンドル・プロジェクトを CICS 領域にデプロイする必要があります。

バンドル・プロジェクトをインストールすると、ATOMSERVICE および URIMAP リソースが CICS によって自動的に生成されます。バンドル・プロジェクトのデプロイについて詳しくは、 345 ページの『CICS Explorer からの CICS バンドル・プロジェクトのデプロイ』を参照してください。

関連概念

257 ページの『CICS からの Atom フィードの URL』
Atom フィード文書、コレクション、およびフィードまたはコレクション内の Atom エントリー文書には、Web クライアントが文書と対話するために使用できる URL (Uniform Resource Locators) が含まれています。 各 URL は、Atom 文書の <atom:link> 要素で提供されます。 Atom 文書では複数の <atom:link> 要素を指定でき、 この要素の rel 属性 (リンク関係と呼ばれる) でさまざまな URL の目的を指定できます。

Atom 構成ファイルの編集

CICS Explorer™ の Atom 構成エディターを使用して、既存のデータを修正したり、Atom 構成ファイルにさらにデータを追加したりします。エディターで情報を更新または追加して変更を保管すると、Atom 構成ファイルの XML が更新されます。

このタスクについて

Atom 構成エディターは、Atom 構成ファイルを編集するためのデフォルトのエディターです。このエディターを使用する場合、入力する必要があるのは属性値のみです。エディターが XML を自動的に作成します。あるいは、テキスト・エディターを使用して XML を直接編集することもできます。以下の手順は、Atom 構成エディターを使用して Atom 構成ファイルを編集する方法を示しています。

手順

1. CICS Explorer の「プロジェクト・エクスプローラー (Project Explorer)」で Atom 構成ファイルをダブルクリックします。デフォルトでは、Atom 構成エディターでファイルが開きます。
2. エディターにリストされた追加属性の値を入力します。あるいは、Atom 構成ウィザードで値が設定されたフィールドを編集することもできます。Atom ウィザードのフィールド表は、Atom 構成ウィザードで値が設定されるフィールドを示しています。表 14 は、Atom 構成エディターでの追加フィールドを示しています。 322 ページの図 19 は、サンプル Atom 構成ファイル `filea.xml` 内の XML 要素に基づいてすべてのフィールドが入力された CICS Explorer の Atom 構成エディターを示しています。

表 14. Atom 構成エディターのフィールド

フィールド	説明
ルート XML 要素 (Root XML Element)	XML バインディングの最上位データ構造の名前。このオプション属性は、XML バインディングに変換セットが複数ある場合のみ必要となります。

表 14. Atom 構成エディターのフィールド (続き)

フィールド	説明
フィード - リンク URI (Feed - Link URI)	Web クライアントが Atom フィードを取得するために使用できる完全パス。パスの先頭部分は、Atom フィードの URIMAP リソース定義で指定した部分パスと一致している必要があります。URI のスキームとホスト構成要素を省略して、パス構成要素のみを指定できます。CICS は、Atom フィードをクライアントに返すときに、Atom 形式仕様に準拠するように、スキームとホスト構成要素を URI に追加します。
ウィンドウ・サイズ (Window Size)	CICS が Atom フィードで返すデフォルトのエントリー数。このフィールドをブランクにしておいた場合は、デフォルト・ウィンドウ・サイズの 8 が使用されます。ウィンドウ・サイズが適用されるのは、フィード・リンク URI、または Atom エントリーの部分リストのナビゲーション URI を使用して Web クライアントが要求を行った場合のみです。
エントリー - リンク URI (Entry - Link URI)	拡張して Atom エントリー文書に適用できる標準 URI パス。これにより、Web クライアントがそれらの文書を個別に取得できるようになります。パスの先頭部分は、Atom の URIMAP リソース定義で指定した部分パスと一致している必要があります。標準パスの残りの部分は、フィード・リンク URI で指定した完全パスとは異なっていなければなりません。URI のスキームとホスト構成要素を省略して、パス構成要素のみを指定できます。CICS は、Atom フィードをクライアントに返すときに、Atom 形式仕様に準拠するように、スキームとホスト構成要素を URI に追加します。
URI	Atom フィードの共通部分パス。ウィザードは、「**デフォルト URI (Default URI)**」フィールドに入力された値と同じ値をこのフィールドに設定します。ウィザードは次に、値に /* を付加します。ウィザードによって入力された値を使用するか、異なる URI を入力した場合、CICS は URIMAP リソース定義を動的に作成します。存在する URIMAP リソースを使用する場合は、このフィールドをクリアできますが、URIMAP リソースを手動で管理する必要があります。

表 14. Atom 構成エディターのフィールド (続き)

フィールド	説明
トランザクション ID	Atom フィードの別名トランザクションの名前。トランザクション ID を指定しない場合は、Atom フィードのデフォルトの別名トランザクション ID である CW2A が使用されます。
ユーザー ID	別名トランザクションが接続される際に使用されるユーザー ID。このフィールドで指定された ID は、Web クライアントから取得される認証済みユーザー ID によってオーバーライドされます。どちらの方法でもユーザー ID が指定されていない場合は、CICS のデフォルト・ユーザー ID が使用されます。

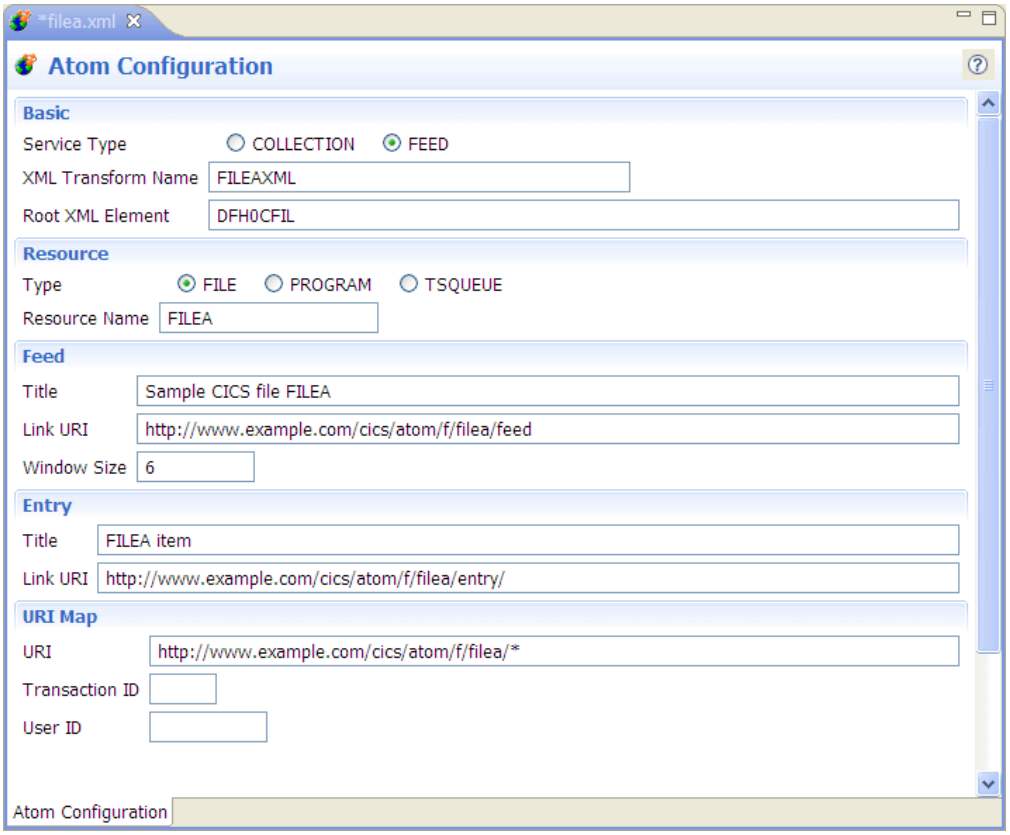


図 19. サンプル Atom 構成ファイル `filea.xml` 内の XML 要素に基づいてすべてのフィールドが入力された CICS Explorer の Atom 構成エディター

3. 「**保管 (Save)**」アイコン  をクリックして変更を保管します。

次のコードは、サンプル Atom 構成ファイル `filea.xml` に基づいてすべてのフィールドが入力された Atom 構成エディターによって生成された XML 要素を示しています。

```
<?xml version="1.0"?>
<cics:atomservice xmlns:atom="http://www.w3.org/2005/Atom"
     xmlns:app="http://www.w3.org/2007/app"
     xmlns:cics="http://www.ibm.com/xmlns/prod/cics/atom/atomservice"
     version="2" type="feed">
    <cics:feed window="6">
        <cics:resource name="FILEA" type="file">
            <cics:bind xmltransform="FILEAXML"/>
        </cics:resource>
    </cics:feed>
    <cics:urimap uri="http://www.example.com/cics/atom/f/filea/*"/>
    <atom:feed>
        <atom:title>Sample CICS file FILEA</atom:title>
        <atom:link rel="self" href="http://www.example.com/cics/atom/f/filea/feed"/>
        <atom:entry>
            <atom:title>FILEA item</atom:title>
            <atom:link rel="self" href="http://www.example.com/cics/atom/f/filea/entry/"/>
            <atom:content cics:resource="FILEA" cics:type="file"/>
        </atom:entry>
    </atom:feed>
</cics:atomservice>
```

タスクの結果

エディターを使用して行われた変更で Atom 構成ファイルが更新されます。

次のタスク

Atom 構成エディターは基本スターター・エディターにすぎず、Atom 構成ファイルに存在しうるすべての属性をサポートしているわけではありません。 Atom 構成エディターに表示されない属性を追加する場合は、XML エディターまたはテキスト・エディターを使用する必要があります。 XML エディターまたはテキスト・エディターを使用して Atom フィード用に構成できる XML 要素について詳しくは、『Atom 構成ファイルで使用される要素』を参照してください。

これ以上編集する必要がなければ、CICS システムへのデプロイメントのためにバンドル・プロジェクトを .zip ファイルとしてローカル・ファイル・システムにエクスポートするか、またはプロジェクトを z/OS UNIX に直接エクスポートすることができます。

バンドル・プロジェクトのデプロイについて詳しくは、345 ページの『CICS Explorer からの CICS バンドル・プロジェクトのデプロイ』を参照してください。

関連概念

257 ページの『CICS からの Atom フィードの URL』
Atom フィード文書、コレクション、およびフィードまたはコレクション内の Atom エントリー文書には、Web クライアントが文書と対話するために使用できる URL (Uniform Resource Locators) が含まれています。 各 URL は、Atom 文書の <atom:link> 要素で提供されます。 Atom 文書では複数の <atom:link> 要素を指定でき、 この要素の rel 属性 (リンク関係と呼ばれる) でさまざまな URL の目的を指定できます。

Atom 構成ファイルで使用される要素

Atom 構成ファイルで使用される XML 要素は、構成される Atom フィードのタイプとその Atom フィードを提供する CICS リソースの場所を定義し、Atom フィードに必要なメタデータを提供します。 これらの要素は、CICS Explorer™ の Atom

構成ウィザードを使用して Atom 構成ファイルに追加できます。また、CICS Explorer™ の Atom 構成エディターを使用して編集することも、XML エディターやテキスト・エディターを使用して Atom 構成ファイルを直接編集することもできます。

<cics:atomservice> 要素

<cics:atomservice> 要素は、Atom 構成ファイルのルート要素です。

属性

type=*typevalue*
構成されている Atom 文書のタイプ。この属性は必須です。*typevalue* は、この Atom 構成ファイルを参照する ATOMSERVICE リソース定義の ATOMTYPE 属性で指定されている文書タイプと一致する必要があります。

type="feed"
Atom フィード文書。

type="service"
Atom サービス文書。

type="collection"
Atom コレクション文書。

type="category"
Atom カテゴリー文書。

version=*version*
Atom 構成ファイル・フォーマットのバージョン。この属性を指定しない場合、バージョンはデフォルトで 1 になります。

version="1"
Atom 構成ファイルが以前のリリースの CICS の Atom フィードと互換性がある場合に、このバージョンを設定します。この値はデフォルトです。

version="2"
Atom 構成ファイルが以下のいずれかの XML 要素および属性を設定する場合に、このバージョンを設定します。

- <cics:urimap> 要素が含まれる
- <cics:bind> 要素に `xmltransform` 属性が含まれる
- <cics:bind> 要素が含まれない

Atom フィードは、このリリースの CICS でのみサポートされます。

xmnls:cics="http://www.ibm.com/xmlns/prod/cics/atom/atomservice"
構成ファイルを CICS Atom XML 名前空間にバインドする名前空間宣言。この名前空間宣言は変更しないでください。

xmnls:atom="http://www.w3.org/2005/Atom"
構成ファイルを Atom XML 名前空間にバインドする名前空間宣言。この名前空間宣言は変更しないでください。

Atom 文書を Atom 出版プロトコルの名前空間にバインドするために、名前空間宣言 **xmnls:app="http://www.w3.org/2007/app"** が使用されます。ただし、この名前空

間宣言は Atom フィードまたはコレクションの Atom 構成ファイルには出現しません。これらの Atom 構成ファイルには app: 接頭部を持つ要素は含まれないためです。 CICS がコレクション用にサービス提供する Atom 文書で <app:edited> 要素が使用されるとき、この名前空間宣言が CICS によって自動的に追加されます。
Atom サービスまたはカテゴリー文書の Atom 構成ファイルでは、名前空間宣言が必要です。

含まれる内容:

『<cics:feed> 要素』

332 ページの『<cics:urimap> 要素』

332 ページの『<atom:feed> 要素』

例

```
<cics:atomservice type="feed" version="2"
      xmlns:cics="http://www.ibm.com/xmlns/prod/cics/atom/atomservice"
      xmlns:atom="http://www.w3.org/2005/Atom">
</cics:atomservice>
```

<cics:feed> 要素

Atom 構成ファイルの <cics:feed> 要素には、フィードとして公開される CICS リソースを記述するその他の要素が含まれます。この要素では、CICS が各フィード文書で返すエントリー数のウィンドウが指定されるとともに、エントリーの要求 URL および Atom ID のスタイルを指定する要素が含まれています。

参照元:

324 ページの『<cics:atomservice> 要素』

属性

window="*number***"|"8"**

CICS が Atom フィード文書で返すデフォルトのエントリー数。この属性はオプションで、デフォルトのウィンドウは 8 個のエントリーです。クライアントは異なるウィンドウ・サイズを指定できます。ウィンドウ・サイズは、クライアントが Atom フィード全体の URL、または Atom エントリーの部分的なリストのナビゲーション URL を使って要求した場合にのみ適用されます。クライアントが個々の Atom エントリーの URL を使って要求を行う場合、CICS は要求された 1 つの Atom エントリーのみを返します。

含まれる内容:

328 ページの『<cics:selector> 要素』

327 ページの『<cics:authority> 要素』

326 ページの『<cics:resource> 要素』

例

```
<cics:feed window="10">
</cics:feed>
```

<cics:resource> 要素

Atom 構成ファイルの <cics:resource> 要素は、フィードとして公開される CICS リソースの名前とタイプを指定します。

参照元:

325 ページの『<cics:feed> 要素』

属性

name="*cics-resource-name***"**
フィードのデータを提供する CICS リソースの名前。この属性は必須です。この名前には大/小文字の区別があり、通常は大文字で指定します。このリソース名は、この Atom 構成ファイルを参照する ATOMSERVICE リソース定義の RESOURCENAME 属性の値と一致する必要があります。

type="*cics-resource-type***"**
CICS リソースのタイプ。この属性は必須です。このリソース・タイプは、小文字で指定します。このリソース・タイプは、この Atom 構成ファイルを参照する ATOMSERVICE リソース定義の RESOURCETYPE 属性の値と一致する必要があります。

type="tsqueue"
一時記憶域キュー。

type="file"
ファイル。

type="program"
プログラム (サービス・ルーチン)。

type="notapplic"
リソース・タイプは適用されません。

含まれる内容:

<cics:bind> 要素
<cics:bind> 要素はオプションであり、次の 2 つのオプション属性が含まれます。

root="*root-element-name***"**
XML バインディングの最上位データ構造の名前を指定します。XML バインディングに変換セットが複数ある場合に、この属性を設定します。

xmltransform="*XMLTRANSFORM***"**
XMLTRANSFORM リソースの 1 文字から 32 文字の名前を指定します。既存の XMLTRANSFORM リソースを再利用する場合、または Atom フィードのリソースのインストール時に動的に作成されるリソースを指定する場合に、この属性を設定します。

329 ページの『<cics:fieldnames> 要素』

例

```
<cics:resource name="feedq" type="tsqueue">
     <cics:bind xmltransform="MYXMLTRANSFORM"/>
</cics:resource>
```

<cics:authority> 要素

Atom 構成ファイルの <cics:authority> 要素は、CICS が個々の Atom エントリーの Atom ID として使用するタグ URI を作成する際に使用する権限名と関連日付を提供します。

タグ URI のその他の要素は、接頭部「tag」、Atom 構成ファイルの <cics:resource> 要素で指定したリソース・タイプとリソース名で構成される詳細、および個々の Atom エントリーのセレクター値です。 269 ページの『Atom エントリーの Atom ID』では、Atom ID で使用するためのタグ URI の形式と要件が説明されています。

Atom ID に代替形式を使用する場合は、プロトタイプ Atom エントリーで <atom:id> 要素を使用してプロトタイプ Atom ID を指定するとともに、<cics:authority> 要素を省略します。代替の Atom ID 形式を使用する場合は、Atom ID が固有であり、 Atom 形式仕様の要件 (RFC 4287) を満たしていることを確認してください。

タグ URI は、CICS 領域内で固有ですが、異なる CICS 領域間で固有であるとは限りません。名前とタイプは同じであるが異なる CICS 領域にあるリソースから Atom フィードをセットアップする場合は、Atom 構成ファイルの <cics:authority> 要素で各フィードについて異なる権限名または異なる日付を指定します。日付が異なるタグ URI は、その他のすべての情報が同じであっても、等価にはなりません。

単一の Atom フィードを扱うユーザー作成サービス・ルーチンによって提供される Atom エントリーではタグ URI は固有になりますが、ユーザー作成サービス・ルーチンが複数のフィードを提供する場合には固有にはなりません。タグ URI ではリソース名としてサービス・ルーチンの名前が使用されるからです。ユーザー作成サービス・ルーチンが複数のフィードを提供する場合には、Atom ID に代わりの形式を選択するか、または Atom 構成ファイルの <cics:authority> 要素で各フィードについて異なる権限名または異なる日付を使用してください。

参照元:

325 ページの『<cics:feed> 要素』

属性

name="*authority-name***"**

権限名。RFC 4151 に準拠するには、権限名はユーザーまたはユーザーの会社に登録済みの完全修飾ドメイン名または E メール・アドレスである必要があります。他のユーザーが所有しているドメイン名や E メール・アドレスは使用できません。RFC 4151 では、権限名を小文字で指定することが推奨されており、同じ権限名がそれぞれ大文字および小文字で指定されているタグ URI は、等価ではありません。社内の他のユーザーがタグ URI を生成するために使用する可能性があるドメイン名を使用するには、会社の同意を得る必要があります。

date="*YYYY-MM-DD***"**

ユーザーまたはユーザーの会社が権限名の所有を開始した日付。日付には、権限名を初めて所有したときから今日までの間の任意の日付を使用できますが、RFC 4151 では、所有開始日とそれに続く数日以内の日付を使用しないことが推奨されています。未来の日付を使用することもできません。CICS では、日付が有効な過去または現在の日付であるかどうかがチェックされます。

例

```
<cics:authority name="example.com" date="2009-01-08"/>
```

<cics:selector> 要素

Atom 構成ファイル内の <cics:selector> 要素は、Atom エントリーのセレクター値の性質と URL 内の位置を特定します。 CICS はこの情報を使用して、具体的な Atom エントリーに対する HTTP 要求でセレクター値を特定して抽出し、Atom フィード文書またはコレクションの一部として Atom エントリーを発行する際に正しいフォーマットで URL を生成します。

参照元:

325 ページの『<cics:feed> 要素』

属性

style="*style-type***"**

スタイルは以下のとおりです。

"segment"

<cics:selector style="segment"/> は、個々の Atom エントリーの標準 URL スタイルを表します。Atom エントリーのセレクター値は URL のパス・コンポーネントの最終セグメントとして置かれます。この URL では、セレクター値は「25」です。

```
http://www.example.com/web20/sample_atom_feed/entry/25
```

デフォルトのスタイルは "segment" であるため、このスタイルを必要とし、セレクター値がリソース・タイプとして CICS で認識される形式になっている場合、Atom 構成ファイルから <cics:selector> 要素を省略できます。

"query"

<cics:selector style="query"/> は、CA8K SupportPac を使用して開発したアプリケーションと互換性のある URL スタイルを表します。Atom エントリーのセレクター値は照会ストリングに置かれます。この URL では、セレクター値は「25」です。

```
http://www.example.com/web20/sample_atom_feed/entry?s=25
```

name="*query-parm-name***"**

style="query" を選択した場合は、この属性を使用して、セレクター値を識別する照会ストリング・キーワードの名前を指定できます。name 属性を省略した場合は、デフォルトのキーワード "s" が使用されます。style="segment" を選択した場合は、この属性を使用しないでください。

format="*format-name***"**

この属性は、セレクター値の形式を識別します。セレクター値が、Atom エントリーが含まれるリソースのタイプであると CICS で認識される形式になっている場合、この属性は指定しないでください。リソースに非標準的なセレクター値が含まれる場合のみ、この属性を指定してください。形式は以下のとおりです。

"decimal"

セレクター値は 10 進数です。 CICS は、一時記憶域キューの場合と、RRDS および VRRDS の各ファイルの場合にこの形式が使用されると

想定します。その他のタイプのリソースを使用していて、そのセレクター値が 10 進数である場合には、この設定を指定してください。

"hexadecimal"
セレクター値は 2 進数です。 CICS は、ESDS および拡張 ESDS の各ファイルの場合にこの形式が使用されると想定します。その他のタイプのリソースを使用していて、そのセレクター値が 2 進数である場合には、この設定を指定してください。

その他のすべてのタイプの VSAM ファイルでは、CICS は、セレクター値の形式は文字ストリングであると想定します。属性を使用して明示的に文字ストリング形式を指定することはできません。

ccsid="*nnnn***"**
この属性は、セレクター値がサービス・ルーチンに渡される前に、それがどのコード化文字セット ID (CCSID またはコード・ページ) に変換されるかを指定します。 format="decimal" または format="hexadecimal" が指定されているか、リソースがこれらのいずれかの形式であると CICS で認識される場合には、この属性を指定しないでください。この属性は、セレクター値が文字ストリング形式で、要求 URL でパーセント・エンコードされる非英数字がセレクター値に含まれる可能性がある場合のみ、指定してください。 "*nnnn*" は、パーセント・エンコードされる UTF-8 文字の変換後の EBCDIC CCSID の数値です。 CICS がセレクター値の中でパーセント・エンコードされる文字を検出しても ccsid 属性が省略される場合には、CICS は LOCALCCSID システム初期設定パラメーターで指定された値を使用します。 CICS でサポートされる CCSID については、「*CICS System Definition Guide*」の LOCALCCSID システム初期設定パラメーターの説明を参照してください。

例

```
<cics:selector style="query" name="sel" format="hexadecimal"/>
```

<cics:fieldnames> 要素

Atom 構成ファイルの <cics:fieldnames> 要素は、Atom エントリー文書のメタデータまたはコンテンツの項目を提供する CICS リソース内のレコードにあるフィールド名を識別します。この要素の属性では、重要なフィールド名を指定します。

参照元:

326 ページの『<cics:resource> 要素』

属性

<cics:fieldnames> 要素の使用をサポートするには、 Atom エントリーのデータを含むリソースの XML バインディングが必要です。作成方法は、「*CICS アプリケーション・プログラミング・ガイド*」で説明されています。 <cics:fieldnames> 要素では、XML バインディングを作成する際に CICS XML アシスタントによって生成された XML 名を使ってフィールド名を指定します。リソースの構造を記述する高水準言語構造の元のフィールド名やコピーブックは指定しません。

ファイルまたは一時記憶域キューの CICS リソース・タイプで、CICS がリソースから Atom エントリーのデータを直接抽出する場合、ファイルまたは一時記憶域キューのレコードにタイトルやタイム・スタンプなど Atom エントリーのメタデータ

を保持するフィールドがあると、<cics:fieldnames> 要素を使う必要があります。レコードにメタデータが含まれる場合、<cics:fieldnames> 要素を追加し、要素の属性を使って、Atom エントリーにメタデータを提供する CICS リソースのレコードのフィールドを指定します。 CICS リソース内のレコードにエントリーの内容のみが含まれており、メタデータが含まれていない場合は、<cics:fieldnames> 要素を省略します。

プログラム (サービス・ルーチンと呼ばれる) を使用して Atom エントリーを提供しており、 DFHATOMPARMS コンテナー内のリソース・ハンドリング・パラメーターを使用してフィールド名をプログラムに渡す場合は、<cics:fieldnames> 要素を追加します。Atom エントリーのメタデータおよび内容を取得するためにプログラムがアクセスするリソース内のレコードにあるフィールドに名前を付けるには、この要素の属性を使用します。これを行うには、リソースで XML バインディングが必要です。ユーザー・プログラムでリソース構造に関する独自の情報を保持しており、リソース・ハンドリング・パラメーターを使用しない場合は、<cics:fieldnames> 要素を省略します。

<cics:fieldnames> 要素の属性はすべてオプションです。作成者の名前といった特定の属性が指定されていない場合、CICS はメタデータのその項目を構成ファイルの <atom:entry> 要素内の対応する要素から提供するか、省略します。すべての属性を省略する場合は、 <cics:fieldnames> 要素も省略できます。

atomid="*fieldname*"
Atom エントリーの固有 ID が含まれるリソース・レコード内のフィールドの名前。

author="*fieldname*"
Atom エントリーの基本作成者の個人名が含まれるリソース・レコード内のフィールドの名前。CICS リソース・レコード内のフィールドから追加の作成者または投稿者の詳細を提供することはできません。これらの詳細は、構成ファイルの <atom:feed> 要素で提供することができますが、すべてのエントリーに適用されます。<cics:fieldnames> 要素を使用して CICS リソースから作成者の詳細を提供するとともに、構成ファイル内で <atom:author> のインスタンスを 1 つ以上指定した場合、CICS リソースの作成者の詳細は構成ファイルの作成者の詳細よりも前に置かれます。

authoruri="*fieldname*"
Atom エントリーの基本作成者 (ブログ・サイトまたは企業の Web サイト) に関連付けられている URL が含まれるリソース・レコード内のフィールドの名前。

category="*fieldname*"
エントリーを分類するカテゴリーが含まれるリソース・レコード内のフィールドの名前。

content="*fieldname*"
Atom エントリーで公開されるコンテンツ全体が含まれるリソース・レコード内のフィールドの名前。レコード内の副構造の名前を指定することもでき、その場合は副構造全体が必要に応じて、子要素を持つ XML 要素として公開コンテンツに渡されます。この属性が省略されている場合、CICS はリソース・レコード全体をエントリーのコンテンツとして公開します。

content_type="*fieldname*"
Atom エントリーのコンテンツのメディア・タイプが入っている、リソース・レコード内のフィールドの名前。

edited="*fieldname*"
レコードが最後に編集された時刻を示すタイム・スタンプが含まれるリソース・レコード内のフィールドの名前。この名前付きフィールドのデータ型は、ATOMSERVICE リソース定義の BINDFILE 属性で指定された XML バインディング内の記述から取得されます。名前付きフィールドが 20 文字以上の文字ストリングである場合、 CICS では、 XML dateTime 形式 (RFC 3339 で説明されています) のタイム・スタンプ (例えば "2008-05-20T11:39:50.325Z") が含まれていると見なされます。名前付きフィールドが長さ 8 のパック 10 進数フィールドで、XML バインディングを準備する際にオプションの DATETIME=PACKED15 パラメーターが使用された場合、CICS では、CICS ABSTIME 値が含まれていると見なされます。

email="*fieldname*"
Atom エントリーの基本作成者の E メール・アドレスが含まれるリソース・レコード内のフィールドの名前。

published="*fieldname*"
レコードが最初に公開された時刻を示すタイム・スタンプまたは ABSTIME 値が含まれるリソース・レコード内のフィールドの名前。CICS では、編集済みの属性の場合と同じ方法でデータの形式が識別されます。

title="*fieldname*"
Atom エントリーのタイトルが含まれるリソース・レコード内のフィールドの名前。

summary="*fieldname*"
Atom エントリーの要約が含まれるリソース・レコード内のフィールドの名前。

updated="*fieldname*"
レコードが最後に更新された時刻を示すタイム・スタンプまたは ABSTIME 値が含まれるリソース・レコード内のフィールドの名前。CICS では、編集済みの属性の場合と同じ方法でデータの形式が識別されます。

例

これらのフィールド名は COBOL 言語構造の例にあるフィールド名の XML バージョンで、 278 ページの『Atom エントリーを格納するための CICS リソースの作成』に例が示されています。

```
<cics:fieldnames title="title_field"
                 summary="summary_field"
                 atomid="atomid_field"
                 content="content_field"
                 author="author_name_field"
                 email="author_email_field"
                 authoruri="author_uri_field"
                 edited="edited_field"
                 updated="updated_field"
                 published="published_field"
                 category="category_field"/>
```

関連資料

287 ページの『DFHATOMPARMS コンテナー』
DFHATOMPARMS は、Atom フィードのデータを提供するサービス・ルーチンと通信するために CICS が使用するパラメーターを含む DATATYPE(CHAR) のコンテナーです。

<cics:urimap> 要素

Atom 構成ファイル内の <cics:urimap> 要素は、URIMAP リソースの URI、ユーザー ID、およびトランザクション名を示します。この要素を使用した場合、URIMAP リソースが CICS によって動的に自動作成されます。

参照元:

324 ページの『<cics:atomservice> 要素』

属性

uri=*value*
Atom フィードのパスの共通部分を指定します。パスを拡張して、Atom フィード内の個々のエントリーへのリンクを設定できます。パスの共通部分は、Atom フィードまたはコレクションに固有でなければなりません。パスにアスタリスクを追加する必要はありません。

transaction=*transaction*
Atom フィードの別名トランザクションの名前を指定します。Atom フィードのデフォルトの別名トランザクションは、CW2A です。

userid=*userid*
別名トランザクションが接続される際に使用されるユーザー ID を指定します。URIMAP リソースで指定されたユーザー ID は、Web クライアントから取得される認証済みユーザー ID によってオーバーライドされます。どちらの方法でもユーザー ID が指定されていない場合は、CICS のデフォルト・ユーザー ID が使用されます。

例

```
<cics:urimap uri="/atom/scenario/tsq/dfhfeedq" transaction="TEST" userid="USERABC">
</cics:urimap>
```

<atom:feed> 要素

Atom 構成ファイルの <atom:feed> 要素は、CICS が返す Atom フィード文書のプロトタイプです。この要素は Atom フィードのメタデータを提供するものであり、単一のプロトタイプ <atom:entry> 要素が含まれています。

参照元:

324 ページの『<cics:atomservice> 要素』

子要素

<atom:feed> 要素の子要素は、Atom 形式仕様 (RFC 4287) で定義されています。一部の子要素は必須であり、その他の子要素はオプションです。

RFC 4287 では、プレーン・テキスト、HTML、または XHTML コンテンツを、<atom:title> 要素などの「テキスト構造」として定義された子要素に使用することを許可しています。CICS では、これらの要素についてプレーン・テキストのコンテンツだけがサポートされているため、"type" 属性を使用して代わりのコンテンツ・タイプを指定することはできません。ユーザーは、子要素を指定せずにプレーン・テキストのコンテンツを提供する必要があります。CICS では、<atom:content> 要素内で HTML、XHTML、およびその他のテキスト・メディア・タイプ (XML など) を Atom エントリーのコンテンツとして使用できます。これについては、Atom エントリーのデータを提供する CICS リソースで指定します。

<atom:feed> 要素の子要素は以下のとおりです。

<atom:author>
Atom フィードの基本作成者 (個人または組織) の個人詳細情報。この要素は構成ファイルで複数指定できます。データは子要素で次のように指定されます。

<atom:name>
個人の名前。<atom:author> 要素を指定する場合、この子要素は必須です。CICS では、この要素が含まれているかどうかがチェックされます。

<atom:uri>
ブログ・サイトや企業の Web サイトなど、個人に関連付けられている URL。この子要素はオプションです。 CICS では、<atom:author> 要素内で複数の <atom:uri> 要素を指定していないかどうかがチェックされます。CICS では URL が有効であるかどうかは検証されないため、正しいことを自分で確認する必要があります。

<atom:email>
個人の E メール・アドレス。この子要素はオプションです。 CICS では、<atom:author> 要素内で複数の <atom:email> 要素を指定していないかどうかがチェックされます。

<atom:author> 要素は、Atom フィード内のすべての Atom エントリーに <atom:author> 要素がある場合を除き、必須です。 Atom エントリーのデータが一時記憶域キューまたはファイルによって提供される場合、CICS では、<atom:author> 要素が <atom:feed> 要素の子またはプロトタイプ <atom:entry> 要素の子として指定されているか、あるいは <cics:fieldnames> 要素の author 属性が指定されているかがチェックされます。Atom エントリーのデータがプログラムによって提供される場合、CICS ではこのチェックが行われません。構成ファイルで <atom:author> 要素を指定しない場合は、RFC 4287 に準拠するために、リソース内のすべての Atom エントリーにこのデータが含まれていることを確認する必要があります。

<atom:category term="" ">
Atom フィードを分類するカテゴリーの名前。この要素はオプションです。CICS は、この要素の単一インスタンスのみをサポートします。 term 属性はカテゴリーの名前を指定します。

<atom:contributor>
Atom フィードの補助作成者の個人詳細情報。この要素はオプションです。この要素は構成ファイルで複数指定できます。データは子要素で次のように指定されます。

<atom:name>
個人の名前。この子要素は、<atom:contributor> 要素を使用する際に必要です。

<atom:uri>
ブログ・サイトや企業の Web サイトなど、個人に関連付けられている URL。この子要素はオプションです。

<atom:email>
個人の E メール・アドレス。この子要素はオプションです。

<atom:entry>
Atom 構成ファイルでは、プロトタイプ <atom:entry> 要素が 1 つ必要です。この要素については、336 ページの『<atom:entry> 要素』を参照してください。

<atom:generator>
Atom フィードを生成するエージェントの名前。 Atom 構成ファイルではこの要素は指定しません。Atom フィード文書を合成するときに CICS が提供します。 CICS は、Atom フィードの生成者として自身の ID を提供します。

<atom:icon>
Atom フィードを表す小さいアイコンを指す URL。この要素はオプションです。CICS では、Atom フィードに対して複数の <atom:icon> 要素を指定していないかどうかがチェックされますが、イメージの縦横比はチェックされません。縦横比は 1 (水平) 対 1 (垂直) である必要があります。

<atom:id>
Atom フィードの固有の ID。 Atom 形式仕様では、Atom フィードについて 1 つの <atom:id> 要素が必要です。Atom 構成ファイルで <cics:authority> 要素を指定して、CICS が Atom ID としてタグ URI を生成するようにした場合は、Atom フィードの <atom:id> 要素を省略できます。CICS では、Atom エントリーの場合と同じ形式で Atom フィードの Atom ID が生成されますが、Atom エントリーの場合に追加されるセレクター値や固有 ID は生成されません。次に例を示します。

```
tag:example.com,2009-01-08:tsqueue:WB20TSQ
```

代わりの Atom ID 形式を使用する場合や、プロトタイプ Atom エントリーで <atom:id> 要素を使用して Atom ID 形式を指定し、これをコピーする場合は、Atom フィードの <atom:id> 要素を含めるとともに、Atom フィードの完全な Atom ID を指定します。この場合は、Atom ID が固有であり、Atom 形式仕様の要件 (RFC 4287) を満たしていることを確認してください。

<atom:link>
Atom フィード文書を特定する URL で、Web クライアントがそれを取得す

るのを可能にします。 257 ページの『CICS からの Atom フィードの URL』では、この URL を構成する方法を説明しています。

Atom フィード文書では、1 つの <atom:link rel="self"> 要素を <atom:feed> 要素の子要素として指定する必要があります。 href 属性には、 Web クライアントが、Atom 形式でフィード文書を取得するために使用できる URL が含まれます。CICS では、 Atom フィード文書用のその他のタイプの <atom:link> 要素はサポートされていません。

Atom 構成ファイル内の <atom:link rel="self"> 要素では、Web クライアントが Atom フィード文書を取得するために使用できる完全なパスを指定する必要があります。パスの先頭部分は、Atom フィードまたはコレクション用の URIMAP リソース定義で指定した部分パスと一致している必要があります。例えば、`/myatomfeed/*` を URIMAP リソース定義内でパス構成要素として指定した場合、Atom 構成ファイルで <atom:link rel="self" href="/myatomfeed/feed.atom"> を指定できます。 40 ページの『CICS Web サポートの URL』でリストされている URL 長の制限は、Atom フィードの URL にも適用されます。

Atom 構成ファイルでは、URL のスキームとホスト構成要素を省略して、パス構成要素のみを指定することもできます。CICS は、Atom 形式仕様に準拠するため、Atom フィードまたは Atom エントリー文書をクライアントに返す際に、スキームとホスト構成要素を URL に追加します。

<atom:logo>
Atom フィードを表す大きいロゴを指す URL。この要素はオプションです。CICS では、Atom フィードに対して複数の <atom:logo> 要素を指定していないかどうかがチェックされますが、イメージの縦横比はチェックされません。縦横比は 2 (水平) 対 1 (垂直) である必要があります。

<atom:rights>
著作権など、請求されている知的所有権が含まれるテキスト・ストリング。この要素では、CICS はプレーン・テキストのみをサポートします。この要素はオプションです。

<atom:subtitle>
Atom フィードのサブタイトル。subtitle では、CICS はプレーン・テキストのみをサポートします。この要素はオプションです。CICS では、Atom フィードに対して複数の <atom:subtitle> 要素を指定していないかどうかがチェックされます。

<atom:title>
Atom フィードのタイトル。title では、CICS はプレーン・テキストのみをサポートします。 1 つの <atom:title> 要素が必要です。CICS では、Atom フィードに対して 1 つの <atom:title> 要素を指定しているかどうかがチェックされます。

<atom:updated>
Atom フィードが最後に更新された時刻。 Atom 構成ファイルではこの要素は指定しません。Atom フィード文書を合成するときに CICS が提供します。 CICS は、Atom フィードを構成する囲まれた <atom:entry> 要素のすべての <atom:updated> 要素のうち最新の時刻をこのタイム・スタンプとし

て提供します。 Atom エントリーが含まれている CICS リソースによってこのデータが提供されない場合、CICS によりデフォルトで現在日時が設定されます。

含まれる内容:

『<atom:entry> 要素』

例

```
<atom:feed>
    <atom:title>CICS Atom feed</atom:title>
    <atom:subtitle>My first Atom feed from CICS</atom:subtitle>
    <atom:link rel="self" href="/web20/sample_atom_feed" />
    <atom:rights>Copyright (c) 2009, Joe Bloggs</atom:rights>
    <atom:author>
         <atom:name>Joe Bloggs</atom:name>
         <atom:uri>http://www.ibm.com/JBloggs/</atom:uri>
         <atom:email>JBloggs@uk.ibm.com</atom:email>
     </atom:author>
     <atom:contributor>
         <atom:name>John Doe</atom:name>
     </atom:contributor>
</atom:feed>
```

<atom:entry> 要素

Atom 構成ファイルの <atom:feed> 要素には、単一のプロトタイプ <atom:entry> 要素が含まれています。CICS は、この要素の子要素に、<cics:resource> 要素で記述されている CICS リソースから提供されるデータを取り込むことにより、Atom エントリー文書を生成します。この処理は、クライアントが要求する Atom エントリーの数だけ繰り返されます。

参照元:

332 ページの『<atom:feed> 要素』

子要素

<atom:entry> 要素の子要素は、CICS でオプションの <atom:source> 要素がサポートされていない点を除き、Atom 形式仕様 (RFC 4287) で定義されているとおりになっています。一部の子要素は Atom エントリーに必須の要素であり、その他の子要素はオプションです。

<atom:entry> 要素の子要素の大半はデフォルト・メタデータを含んでおり、CICS リソースやサービス・ルーチンが提供しない個々の Atom エントリー用のデータ項目を提供します。

- これらの子要素のいずれかのデータが常に CICS リソースやサービス・ルーチンによって提供される場合、通常は構成ファイル内の <atom:entry> 要素でその要素を省略できます。例外は <atom:title> 要素で、CICS はこれを必要とします。ただし、すべての Atom エントリーにタイトルがあり、デフォルトのタイトル設定が適切でない場合は、空の要素を使用できます。
- リソースに Atom エントリーが保持されており、そのリソースのレコードにおいてメタデータの項目の一部または全部でフィールドが欠如している場合、Atom 構成ファイルで指定することによりそれらのメタデータの項目を提供できます。

<atom:entry> 要素でメタデータを提供すると、CICS は、CICS リソースまたはサービス・ルーチンに対応するデータの項目がない場合に、常にこのメタデータを使用します。

- オプションのメタデータの項目はいずれも、それらの項目でデフォルトを提供しない場合は省略できます。

<atom:entry> 要素の子要素の一部は、<cics:fieldnames> 要素の属性に対応しています。子要素に <cics:fieldnames> 要素で対応する属性がある場合、CICS はリソースからそのデータ項目を配信することができ、サービス・ルーチンがある場合はそこから配信することもできます。子要素に <cics:fieldnames> 要素で対応する属性がない場合、Atom 構成ファイルでのみ指定できます。

RFC 4287 では、プレーン・テキスト、HTML、または XHTML コンテンツを、<atom:title> 要素などの「テキスト構造」として定義された子要素に使用することを許可しています。CICS では、これらの要素についてプレーン・テキストのコンテンツだけがサポートされているため、"type" 属性を使用して代わりのコンテンツ・タイプを指定することはできません。ユーザーは、子要素を指定せずにプレーン・テキストのコンテンツを提供する必要があります。CICS では、<atom:content> 要素内で HTML、XHTML、およびその他のテキスト・メディア・タイプ (XML など) を Atom エントリーのコンテンツとして使用できます。これについては、Atom エントリーのデータを提供する CICS リソースで指定します。

<atom:entry> 要素の子要素は以下のとおりです。

<app:edited>
Atom エントリーが最後に編集された時刻。この要素はコレクションの一部である Atom エントリーにのみ適用され、そのような Atom エントリーでは必須 (SHOULD 要件) となっています。 <app:edited> 要素は、構成ファイル内のプロトタイプ Atom エントリーに指定できません。この要素は、<cics:fieldnames> 要素の edited 属性に対応します。 CICS リソースまたはサービス・ルーチンによってこのデータが提供されない場合は、CICS によって現在日時がデフォルトとして提供されます。 CICS は <app:edited> 要素の使用をサポートしていますが、パフォーマンス上の理由から、コレクション内の Atom エントリーをリストとしてクライアントに返す際に、この要素によって自動的に順序付けすることはしません。コレクション内の Atom エントリーの順序付けの詳細については、265 ページの『Atom エントリーの順序』を参照してください。

<atom:author>
Atom エントリーの基本作成者 (個人または組織) の個人詳細情報。この要素は構成ファイルで複数指定できます。このデータは、<atom:feed> 要素についての説明にあるように、 <atom:name>、<atom:uri>、および <atom:email> の各子要素で提供されます。Atom エントリーに作成者が含まれていない場合は Atom フィードの作成者が Atom エントリーに適用されるため、Atom フィードに <atom:author> 要素がある場合、この要素はオプションです。この要素は、<cics:fieldnames> 要素の author、email、および authoruri の各属性に対応します。そのため、CICS リソースによって必要なデータが常に提供される場合は、省略できます。<cics:fieldnames> 要素を使用して CICS リソースから作成者の詳細を提供するとともに、構成ファイ

ル内で <atom:author> のインスタンスを 1 つ以上指定した場合、CICS リソースの作成者の詳細は構成ファイルの作成者の詳細よりも前に置かれます。

<atom:category term=" ">
Atom エントリーを分類するカテゴリーの名前。この要素はオプションです。CICS は、この要素の単一インスタンスのみをサポートします。 term 属性はカテゴリーの名前を指定します。この要素は、<cics:fieldnames> 要素の category 属性に対応します。そのため、CICS リソースによって適切なデータが常に提供される場合は、省略できます。

<atom:content type=" " cics:resource=" " cics:type=" "/>
Atom エントリーのコンテンツ。この要素は必須です。

<atom:content> 要素のデータは、 CICS がフィード文書を発行する際にリソースから提供されるため、構成ファイル内のこの要素にはデータが含まれていません。すべての Atom エントリーについてリソースからコンテンツが提供されることを確認してください。CICS では、リモート・コンテンツを参照するために "src" 属性を使用する Atom エントリーなど、データを含まない Atom エントリーの配信はサポートされていません。

type=" "
型属性は、CICS が Atom エントリーに対して求めるコンテンツの型を指定します。「text」、「html」、「xhtml」、または IANA メディア・タイプを指定できます。この属性が存在しない場合、CICS はデフォルトのメディア・タイプである「application/xml」を使用します。 RFC 4287 にあるとおり、プレーン・テキストでは IANA メディア・タイプ「text/plain」の代わりにメディア・タイプ「text」、「text/html」の代わりに「html」、「application/xhtml+xml」の代わりに「xhtml」を使用します。コンテンツが他のフォーマットの場合、インターネットでそのフォーマットに通常使用する IANA メディア・タイプを指定します。メディア・タイプのリストについては、http://www.iana.org/assignments/media-types/ を参照してください。なお、CICS では非テキスト・メディア・タイプがサポートされないことに注意してください。

CICS が Atom エントリーをリソースから直接送信する際、メディア・タイプまたはデフォルトがそのリソース内のデータに合うものでなければなりません。なぜなら CICS は、Atom 文書の発行時に、コンテンツをこのメディア・タイプでラベル付けするためです。サービス・ルーチンを使用して Atom エントリーにコンテンツを提供する際、このメディア・タイプまたはデフォルトがサービス・ルーチンに提供されます。サービス・ルーチンは、これをオーバーライドして代わりのメディア・タイプを指定できますし、あるいは CICS によって Atom 構成ファイルからのメディア・タイプでコンテンツをラベル付けすることもできます。

" cics:resource=" および " cics:type=" "
cics:resource と cics:type 属性は、Atom フィードにデータを提供する CICS リソースの名前とタイプを記述します。これらの属性の値は、<cics:resource> 要素について指定された name 属性および type 属性の値と一致している必要があります。これらの属性の値については、326 ページの『<cics:resource> 要素』を参照してください。

<atom:content> 要素は、<cics:fieldnames> 要素の content 属性および content_type 属性に対応します。

<atom:contributor>

Atom エントリーの補助作成者の個人詳細情報。この要素はオプションです。このデータは、<atom:feed> 要素で指定されているように、<atom:name>、<atom:uri>、および <atom:email> の各子要素で提供されます。個々のリソースで <atom:contributor> 要素のデータを指定することはできないため、<cics:fieldnames> 要素に対応する属性はなく、ユーザーが提供したデータがすべての Atom エントリーに適用されます。データがすべての Atom エントリーに適用されるため、<atom:feed> 要素下で contributor を指定することもできます。

<atom:id>

Atom エントリーの固有の ID。 Atom 形式仕様では、各 Atom エントリーについて 1 つの <atom:id> 要素が必要です。 269 ページの『Atom エントリーの Atom ID』には、Atom ID の使用要件について説明されています。

CICS では、Atom エントリーの固有の Atom ID として使用するためのタグ URI を生成できます。タグ URI には、Atom 構成ファイルの <cics:authority> 要素で指定する属性、<cics:resource> 要素で指定するリソース・タイプとリソース名、および個々の Atom エントリーのセレクター値が含まれます。この形式で必要を満たせる場合は、プロトタイプ Atom エントリーで <atom:id> 要素を省略します。

CICS によって生成されるタグ URI 形式を使用する代わりに、プロトタイプ Atom エントリーで <atom:id> 要素を使用して、代わりの形式でプロトタイプ Atom ID を指定することもできます。CICS は、セレクター値または適切な固有 ID を追加して固有の Atom ID を生成します。代替の Atom ID 形式を使用する場合は、Atom ID が固有であり、 Atom 形式仕様の要件 (RFC 4287) を満たしていることを確認してください。

<atom:link>

Atom エントリーを特定する標準の URL であり、Web クライアントがそれを取得するのを可能にします。 257 ページの『CICS からの Atom フィードの URL』では、この URL を構成する方法を説明しています。

CICS の場合、1 つの <atom:link rel="self"> 要素を <atom:entry> 要素の子要素として指定する必要があります。コレクション内の Atom エントリーの場合、Atom 形式仕様では「self」ではなく「edit」のリンク関係が必要であり、コレクション内の Atom エントリーを送信する際に CICS で自動的にこのリンク関係を提供します。 Atom エントリーが Atom フィードにあるかコレクションにあるかに関係なく、<atom:link rel="self"> を Atom 構成ファイルで指定する必要があります。

Atom 構成ファイルでは、すべての Atom エントリー文書に適用されるように拡張可能な標準パスとして、<atom:link rel="self"> 要素で URL を指定します。パスの先頭部分は、Atom フィードまたはコレクション用の URIMAP リソース定義で指定した部分パスと一致している必要があります。標準パスの残りの部分は、Atom フィード用の <atom:link rel="self"> 要素で指定した完全パスとは異なっている必要があります。例えば、`/myatomfeed/*` を URIMAP リソース定義でパス構成要素として指定し、<atom:link rel="self" href="/myatomfeed/feed.atom"> を Atom 構成ファイルで Atom フィード全体

のリンクとして指定した場合は、<atom:link rel="self" href="/myatomfeed/entries/"> を Atom エントリーの標準パスとして指定できます。 40 ページの『CICS Web サポートの URL』でリストされている URL 長の制限は、Atom フィードの URL にも適用されます。

CICS は、Atom エントリー文書をクライアントに返す際、Atom エントリーに適したセレクター値をこのパスに追加して、完全なリンクを作成します。 264 ページの『Atom エントリーのセレクター値』ではセレクター値について説明しています。セレクター値をパスに追加する方法を指定するには、Atom 構成ファイルで <cics:selector> 要素を使用します。デフォルトの "segment" スタイルを選択した場合や、要素を省略した場合には、CICS によって <atom:link rel="self" href="/myatomfeed/entries/23"> などのリンクが作成されます。代わりに "query" スタイルを選択すると、 CA8K SupportPac 用に開発されたアプリケーションと互換性のある形式が生成されます。また、セレクター値が 16 進数であることを指定する必要がある場合、<cics:selector> 要素を使用することもできます。

Atom 構成ファイルでは、URL のスキームとホスト構成要素を省略して、パス構成要素のみを指定することもできます。CICS は、Atom 形式仕様に準拠するため、Atom フィードまたは Atom エントリー文書をクライアントに返す際に、スキームとホスト構成要素を URL に追加します。

<atom:published>

Atom エントリーが最初に作成または公開された時刻。この要素はオプションです。この要素は、<cics:fieldnames> 要素の published 属性に対応します。CICS リソースによってこのデータが提供されない場合は、CICS によって現在日時がデフォルトとして提供されます。Atom 構成ファイルの <atom:published> 要素は、RFC 3339 で説明されているように、XML dateTime 形式の代替デフォルト・タイム・スタンプで指定できます。

<atom:rights>

著作権など、請求されている知的所有権が含まれるテキスト・ストリング。この要素では、CICS はプレーン・テキストのみをサポートします。この要素はオプションであり、指定しない場合は Atom フィードの <atom:rights> 要素が適用されます。 <cics:fieldnames> 要素には対応する属性がないため、ユーザーが提供したデータがすべての Atom エントリーに適用されます。

<atom:summary>

Atom エントリーのコンテンツの簡略説明。 summary では、CICS はプレーン・テキストのみをサポートします。この要素は、<cics:fieldnames> 要素の summary 属性に対応します。この要素は、Atom エントリーのコンテンツがテキスト、HTML、XHTML、または XML のいずれでもない場合に必要です。そのため、これらのカテゴリーに収まらないコンテンツを提供する予定で、CICS リソースによって必ずしも要約が提供されるわけではない場合は、この要素を使用してデフォルトの要約を提供します。 CICS リソースによって要約が提供されず、すべての Atom エントリーで同じ要約を表示させたい場合にも、この要素を使用できます。それ以外の場合、この要素は省略できます。CICS では、Atom エントリーに対して複数の <atom:summary> 要素を指定していないかどうかがチェックされます。

<atom:title>

Atom エントリーのタイトル。1 つの <atom:title> 要素が必要です。title では、CICS はプレーン・テキストのみをサポートします。この要素は、<cics:fieldnames> 要素の title 属性に対応します。CICS リソースによって Atom エントリーのタイトルが常に提供される場合でも、プロトタイプ Atom エントリーに <atom:title> 要素を含める必要があります。すべての Atom エントリーに適用できる、適切なデフォルトのタイトルを使用してください。すべての Atom エントリーにタイトルがあり、デフォルトのタイトル設定が適切でない場合は、空の要素を使用します。

<atom:updated>

Atom エントリーが最後に大幅に更新された時刻。この要素は Atom 仕様で必須ですが、 Atom 構成ファイルのプロトタイプ Atom エントリーでは指定できません。この要素は、<cics:fieldnames> 要素の updated 属性に対応します。CICS リソースによってこのデータが提供されない場合は、CICS によって現在日時がデフォルトとして提供されます。

例

```
<atom:entry>
    <atom:title>An entry from my feed</atom:title>
    <atom:summary>DEFAULT --- This is the default summary</atom:summary>
    <atom:link rel="self" href="/web20/sample_atom_feed/entry" />
    <atom:author>
          <atom:name>Joe Bloggs</atom:name>
          <atom:uri>http://www.hursley.ibm.com/JBloggs/</atom:uri>
          <atom:email>JBloggs@uk.ibm.com</atom:email>
    </atom:author>
    <atom:contributor>
          <atom:name>John Doe</atom:name>
    </atom:contributor>
    <atom:category term="Comments" />
    <atom:published>2008-12-02T15:41:00</atom:published>
    <atom:content type="text" cics:resource="WB20TSQ" cics:type="tsqueue" />
 </atom:entry>
```

CICS 用の Atom 要素のリファレンス

以下の表には、Atom フィード文書で使用される要素、Atom エントリー文書で使用される要素、 <cics:fieldnames> 要素の属性、およびリソース・ハンドリング用に CICS がサービス・ルーチンに渡すことができるパラメーターの間にある関係を示す参照情報が記載されています。

CICS で Atom フィード文書を作成する際は、RFC 4287 の Atom 形式仕様で定義されている要素を指定できます。これらの要素の一部は <atom:feed> 要素の子要素として使用され、Atom フィード全体用のメタデータ (フィードのタイトルなど) を提供します。一部の要素は <atom:entry> 要素の子要素として使用され、個々のエントリーのメタデータまたは内容を提供します。大半の要素はフィード用に指定されるとともに、個々のエントリー用にも指定されます。例えば、各エントリーには <atom:id> 要素で指定される固有 ID があり、フィードにも固有 ID があります。各要素の完全な説明については、RFC 4287 を参照してください。

CICS 内の Atom フィード文書では、 CICS は単一のプロトタイプ <atom:entry> 要素を使用して個々のエントリーを生成します。この要素で子要素を指定した場合、デフォルトではそのメタデータがすべてのエントリーに適用されます。ただし、Atom フィードの内容を提供するために使用している CICS リソースに適切なメタ

データが含まれている場合は、 <cics:fieldnames> 要素の属性を使用して、リソース・レコードに子要素のデータが存在するかどうか、およびその場所を CICS に伝えることができます。例えば、リソース・レコードに含まれているエントリーのタイトルを提供するリソース・レコード内のフィールドを指定できます。リソース・レコードに作成者の名前などの特定のメタデータが含まれていない場合は、<cics:fieldnames> 要素でその属性を省略できます。CICS は、その項目のメタデータを構成ファイルのプロトタイプ <atom:entry> 要素内の対応する要素から提供するか、省略します。リソース・レコードに適切なメタデータが含まれていない場合は、<cics:fieldnames> 要素を完全に省略できます。この場合、CICS はリソース・レコード全体をエントリーの内容として公開します。

CICS からサービス・ルーチンに渡されるパラメーターには、 <cics:fieldnames> 要素の属性に対応するパラメーターが含まれます。リソース構造に関する情報をサービス・ルーチンに直接コーディングするのではなく、その情報を Atom 構成ファイルから取得するサービス・ルーチンを作成する場合は、これらのリソース・ハンドリング・パラメーターを使用できます。この方法を使用すると、複数のリソースを処理できる汎用サービス・ルーチンを作成できます。

表 15. <atom:feed> 要素と <atom:entry> 要素の子要素

要素	意味	フィード用	エントリー用	<cics:fieldnames> 属性 (エントリー用)	サービス・ルーチンのパラメーター (エントリー用)
<app:edited>	エントリーが最後に編集された時刻	使用されません	コレクション内では必須。CICS が生成	edited	ATMP_ EDITED
<atom:author>	基本作成者の詳細	すべてのエントリーにこの要素がある場合を除き必須	フィードにこの要素がある場合はオプション	適用外 (データは子要素に存在)	適用外 (データは子要素に存在)
<atom:category>	フィードまたはエントリーを分類するカテゴリー	オプション	オプション	category	ATMP_ CATEGORY_ FLD
<atom:content type=" ">	エントリーの内容	使用されません	必須	content、content_type	ATMP_ CONTENT_ FLD および ATMP_ CONTENT_ TYPE_FLD
<atom:contributor>	補助作成者の詳細	オプション	オプション	適用外 (データは子要素に存在)	適用外 (データは子要素に存在)
<atom:email>	作成者または投稿者の E メール・アドレス	オプション	オプション	email (作成者の場合のみ、投稿者はサポート対象外)	ATMP_ EMAIL_ FLD
<atom:generator>	フィードを生成したエージェント	CICS が生成	使用されません	適用外 (エントリーでは使用されない)	適用外 (エントリーでは使用されない)

表 15. <atom:feed> 要素と <atom:entry> 要素の子要素 (続き)

要素	意味	フィード用	エントリー用	<cics:fieldnames> 属性 (エントリー用)	サービス・ルーチンのパラメーター (エントリー用)
<atom:icon>	フィードを表すアイコン	オプション	使用されません	適用外 (エントリーでは使用されない)	適用外 (エントリーでは使用されない)
<atom:id>	フィードまたはエントリーの固有 ID	必須。<cics:authority> 要素を指定した場合には CICS が生成	必須。<cics:authority> 要素を指定した場合には CICS が生成	atomid	ATMP_ ID_ FLD
<atom:link rel="self">	フィードまたはエントリー文書を取得するための URL	必須	CICS で必須	適用外 (リソースには格納されない)	適用外 (リソースには格納されない)
<atom:link rel="edit">	コレクション内のエントリーを編集するための URL (メンバー URI)	コレクション用に CICS が生成	エントリーがコレクションにあれば CICS が生成	適用外 (リソースには格納されない)	適用外 (リソースには格納されない)
<atom:logo>	フィードのロゴ	オプション	使用されません	適用外 (エントリーでは使用されない)	適用外 (エントリーでは使用されない)
<atom:name>	作成者または投稿者の名前	作成者または投稿者の要素では必須	作成者または投稿者の要素では必須	author (作成者の場合のみ、投稿者はサポート対象外)	ATMP_ AUTHOR_ FLD
<atom:published>	エントリーが最初に作成または公開された時刻	使用されません	オプション	published	ATMP_ PUBLISHED_ FLD
<atom:rights>	フィードの知的所有権	オプション	オプション	リソースではサポート対象外	リソースではサポート対象外
<atom:source>	別のフィードでのエントリーの使用から発生したメタデータ	使用されません	オプション。CICS ではサポート対象外	サポート対象外	サポート対象外
<atom:subtitle>	フィードのサブタイトル	オプション	使用されません	適用外 (エントリーでは使用されない)	適用外 (エントリーでは使用されない)
<atom:summary>	エントリーの内容の簡略説明	使用されません	内容がテキストでも XML でもない場合は必須	summary	ATMP_ SUMMARY_ FLD
<atom:title>	フィードのタイトル	必須	必須	title	ATMP_ TITLE_ FLD

表 15. <atom:feed> 要素と <atom:entry> 要素の子要素 (続き)

要素	意味	フィード用	エントリー用	<cics:fieldnames> 属性 (エントリー用)	サービス・ルーチンのパラメーター (エントリー用)
<atom:updated>	フィードが最後に更新された時刻	必須。CICS が生成	必須。CICS が生成	updated	ATMP_ UPDATED_ FLD
<atom:uri>	作成者または投稿者の Web サイトの URL	オプション	オプション	authoruri (作成者の場合のみ、投稿者はサポート対象外)	ATMP_ AUTHORURI_ FLD

Atom フィード用の別名トランザクションの作成

別名トランザクションは、Atom フィードの処理の後半の段階を処理します。CICS には、デフォルトの Atom フィード別名トランザクションである CW2A のリソース定義が備わっています。代わりの別名トランザクションを定義する場合は、TRANSACTION リソース定義をセットアップします。

このタスクについて

CICS Web サポートで処理された Atom 以外の HTTP 要求の場合、ユーザー作成のアプリケーション・プログラムがその要求を処理する際にのみ別名トランザクションを使用します。ただし、Atom フィードの場合、ユーザー作成のサービス・ルーチンが関係しているかどうかに関わらず、すべての要求の処理に別名トランザクションが使用されます。

代わりの別名トランザクション名は、以下のような目的に使用できます。

- 監査、モニター、またはアカウンティング
- リソースおよびコマンド・セキュリティー設定の変更
- 開始優先順位の割り振り
- DB2 リソースの割り振り
- 各種 CICS アプリケーション・プログラムへの各種ランナウェイ値の割り当て
- トランザクション・クラスの制限

別名トランザクション定義は、必要な個数だけ設定できます。特定の要求に必要な別名トランザクションは、URIMAP 定義を使用して指定できます。

CW2A は RESSEC(YES) および CMDSEC(YES) を指定することにより、リソースおよびコマンド・セキュリティーが CICS 領域に対してアクティブな場合に、それがこのトランザクションに適用されるようにします。別名トランザクションについてリソースおよびコマンド・セキュリティーを指定する場合、トランザクションで使用されるリソースおよびコマンドにアクセスするための適切な権限を Web クライアントに与える必要があります。Atom フィードおよびコレクションのセキュリティーの詳細については、395 ページの『第 24 章 Atom フィードのセキュリティー』を参照してください。

トランザクション・リソース定義を作成するには、 TRANSACTION リソース定義の説明に従ってください。これらの説明に従う際には、以下のことに注意してください。

手順

- CW2A の定義に基づいて別名トランザクション定義を作成し、必要な変更を行います。 CW2A の定義は次のとおりです。

```
DEFINE TRANSACTION(CW2A)   GROUP(DFHWEB2)
       PROGRAM(DFHW2A)     TWASIZE(512)
       PROFILE(DFHCICST)   STATUS(ENABLED)
       TASKDATALOC(ANY)    TASKDATAKEY(CICS)
       RUNAWAY(SYSTEM)     SHUTDOWN(DISABLED)
       PRIORITY(1)         TRANCLASS(DFHTCL00)
       DTIMOUT(NO)         TPURGE(NO)        SPURGE(YES)
       RESSEC(YES)         CMDSEC(YES)
       DESCRIPTION(CICS Web2.0 Atomservice alias transaction)
```

- 別名トランザクション定義では、CICS 提供の別名プログラム DFHW2A を使用する必要があります。 この別名プログラムは、ATOMSERVICE 定義で指定されているユーザー作成のサービス・ルーチンまたは CICS ルーチンにアクセスします。
- 別名トランザクション定義はローカル・トランザクションでなければなりません。
- 別名トランザクションのトランザクション作業域のサイズ (TWASIZE) は、512 バイト以上でなければなりません。
- 別名トランザクションの優先順位は、CWXN または CWXU などの Web 接続タスクに関連するトランザクションの優先順位と同じか、それより高くなるようにしてください。 このような設定が重要である理由については、 CICS Web サポート・トランザクションの優先順位 「*CICS パフォーマンス・ガイド*」に説明されています。

CICS Explorer からの CICS バンドル・プロジェクトのデプロイ

CICS Explorer からアプリケーションを CICS バンドル・プロジェクトとしてデプロイし、BUNDLE リソースを使用してリソースを動的に自動作成できます。 BUNDLE リソースはアプリケーションを表すので、BUNDLE リソースを使用可能/使用不可にすることにより、CICS でアプリケーションを使用できるかどうかを管理することもできます。

このタスクについて

バンドル・プロジェクトは、CICS リソース、成果物、参照、およびマニフェストのコレクションであり、CICS 領域にデプロイしてアプリケーションのすべてまたは一部を表すことができます。マニフェストは、アプリケーションの前提条件システム・リソースを含めてバンドルの内容を示したファイルです。 CICS はこれらのシステム・リソースを動的に作成するわけではありませんが、それらが CICS 領域に存在するかどうかを検査できます。このリソース分離は、バンドルの再パッケージ化も再デプロイもせずに、同一アプリケーションを複数の CICS 領域にインストールできることを意味します。

CICS Explorer で CICS バンドル・プロジェクトを作成した場合、それを CICS 領域にエクスポートする必要があります。バンドル・プロジェクトを .zip ファイルと

してローカル・ファイル・システムにエクスポートし、そのアーカイブ・ファイルを z/OS UNIX に転送し、それをディレクトリーに解凍します。あるいは、プロジェクトを z/OS UNIX に直接エクスポートします。バンドル・プロジェクトの BUNDLE リソースを定義し、使用可能にする必要もあります。その後、バンドル・プロジェクトのアプリケーション・リソースが、CICS によって動的に自動作成されます。

注: バンドルを z/OS UNIX ファイル・システムに直接エクスポートできるのは、ネットワークが IPv4 (Internet Protocol version 4) を使用する場合のみです。この手順は IPv6 ネットワークの場合は使用できません。ポートを保護して SSL トラフィックを受け入れることができます。こうすると、CICS Explorer が検出して SSL (Secure Sockets Layer) モードに切り替えます。その結果、非暗号化データが流れなくなります。 SSL 接続を確立できない場合は、基本認証が使用されます。

手順

1. 以下のいずれかの方法で、CICS Explorer からバンドル・プロジェクトをエクスポートします。
   - ローカル・ファイル・システムにエクスポート:
     a. 「プロジェクト・エクスプローラー (Project Explorer)」ビューで CICS バンドル・プロジェクトを右クリックし、「**エクスポート (Export)**」を選択します。「エクスポート (Export)」ウィンドウが開きます。
     b. 「**ファイルのアーカイブ (Archive File)**」をクリックします。
     c. 「**次へ (Next)**」をクリックします。
     d. 上部のペインでプロジェクトのファイルをすべて選択しておきます。「**保管先アーカイブ・ファイル (To archive file)**」フィールドにファイル名を入力し、「**.zip 形式で保管 (Save in .zip format)**」を選択します。
     e. 「**完了 (Finish)**」をクリックします。
   - z/OS UNIX ファイル・システムに直接エクスポート:
     a. 「プロジェクト・エクスプローラー (Project Explorer)」ビューで CICS バンドル・プロジェクトを右クリックし、「**z/OS UNIX ファイル・システムにエクスポート (Export to z/OS UNIX File System)**」を選択します。「z/OS UNIX ファイル・システムにエクスポート (Export to z/OS UNIX File System)」ウィンドウが開きます。「**バンドル・プロジェクト (Bundle project)**」フィールドにバンドル・プロジェクトの名前が入っています。その代わりにこのフィールドにバンドル・プロジェクトの名前を入力するか、「**参照 (Browse)**」をクリックすることもできます。
     b. 「**接続 (Connection)**」フィールドで、ツイスティーをクリックして FTP ポートへの既存の接続を選択するか、または接続テキスト・ハイパーリンクをクリックして「**System z - FTP**」タイプの新規接続を定義します。
     c. 「**ディレクトリー (Directory)**」フィールドで、転送の宛先として z/OS UNIX ファイル・システムのディレクトリーの名前を指定します。ディレクトリー・フォルダーの名前を入力すると、ディレクトリー・ツリーが最新表示され、そのフォルダーがルートとして表示されます。フォルダーをダブルクリックすると、そのフォルダーがツリーのルートになります。あるいは、ツリーから選択し、ディレクトリーにナビゲートすることもできます。

d. 「**既存の z/OS Unix バンドル・フォルダーを削除 (Delete existing z/OS Unix Bundle Folder)**」チェック・ボックスを選択すると、z/OS UNIX ファイル・システムの宛先ディレクトリーにある指定したフォルダーとその子フォルダーおよびファイルすべてが、ファイル転送前に除去されます。このチェック・ボックスを選択しなかった場合、その名前のフォルダーが存在すると、エクスポートは行われません。

e. 「**完了 (Finish)**」をクリックします。

2. アプリケーション・バンドルの BUNDLE リソースを定義し、使用可能にします。 指定する属性の詳細については、BUNDLE 属性を参照してください。 CICS はバンドル・ディレクトリー内のマニフェストを読み取り、アプリケーション・リソースを動的に作成します。アプリケーションを正常に実行できるように、アプリケーション外の必要な参照 (例えばプログラムやファイルの参照) が CICS 領域に存在するかどうかも検査します。

タスクの結果

バンドル・プロジェクトが z/OS UNIX にエクスポートされ、バンドル・プロジェクトの BUNDLE リソースが定義されて使用可能になります。

次のタスク

BUNDLE リソースを使用可能/使用不可にすることにより、CICS でバンドル・プロジェクト・アプリケーションを使用できるかどうかを管理できます。

第 21 章 Atom フィードの CICS サンプル

CICS では、サンプルの URIMAP および ATOMSERVICE リソース定義、Atom 構成ファイル、XML バインディングおよびサービス・ルーチンが提供されています。

Atom フィードのサンプル・サービス・ルーチンは、SDFHSAMP サンプル・ライブラリーにあります。Atom フィードのサンプル・リソースは、次の 2 つの場所にあります。

- CICS リソース・グループ DFH$WEB2。
- z/OS UNIX 上の CICS ファイルのルート・ディレクトリーの /samples/web2.0/ サブディレクトリー (CICS システム初期設定パラメーター USSHOME で指定)。USSHOME のデフォルト値は /usr/lpp/cicsts/cicsts42 です。

リソース・グループ DFH$WEB2 内の CICS リソースは、/samples/web2.0/ サブディレクトリー内のファイルを参照します。その際、USSHOME のデフォルト値を用いてパス /usr/lpp/cicsts/cicsts42/samples/web2.0/ を使用します。 CICS 領域の USSHOME システム初期設定パラメーターによって、z/OS UNIX 上の CICS ファイルのデフォルト以外のルート・ディレクトリーが指定されている場合には、これらのサンプルを使用するために以下のステップを実行する必要があります。

1. リソース・グループ DFH$WEB2 を、新しいリソース・グループにコピーします。
2. DFH$WEB2 グループのコピー内にある URIMAP リソース定義を変更して、HFSFILE 属性でデフォルト・ディレクトリー /usr/lpp/cicsts/cicsts が出現する箇所を、z/OS UNIX 上の CICS ファイル用に CICS 領域が使用しているルート・ディレクトリーの名前に変えます。
3. DFH$WEB2 グループのコピー内にある ATOMSERVICE リソース定義を変更して、CONFIGFILE および BINDFILE 属性でデフォルト・ディレクトリー /usr/lpp/cicsts/cicsts42 が出現する箇所を、z/OS UNIX 上の CICS ファイル用に CICS 領域が使用しているルート・ディレクトリーの名前に変えます。

サンプル Atom コレクション

サンプル Atom コレクションでは、従業員の地理的位置を含む従業員データが組み込まれた Atom コレクションを作成する方法が示されています。このサンプル Atom コレクションをセットアップおよび使用するための指示については、 CICS TS for z/OS バージョン 4.1 Information Center (https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp) の Web 2.0 シナリオ『従業員情報を処理する Atom フィードの作成』に記されています。

以下の表に示されているコンポーネントを使用して、サンプル Atom コレクションが作成されます。

表 16. サンプル Atom コレクション用のコンポーネント

コンポーネント	目的	場所
URIMAP リソース DFH$W2Q1	サンプル Atom コレクションおよび個別の Atom エントリーの HTTP 要求を処理します	CICS グループ DFH$WEB2 またはご使用のコピー
ATOMSERVICE リソース DFH$W2Q1	Atom コレクションを生成するために使用するリソースを指名します	CICS グループ DFH$WEB2 またはご使用のコピー
Atom 構成ファイル `dfh0w2q1.xml`	Atom フィード文書および Atom エントリー用のメタデータと構造を提供します	*USSHOME*`/samples/web2.0/atom`
XML バインディング `dfh0w2q1.xsdbind` および XML スキーマ `dfh0w2q1.xsd`	CICS に、一時記憶域キュー内のレコードの構造について通知します	*USSHOME*`/samples/web2.0/atom`
URIMAP リソース DFH$W2AC	Web ブラウザーで Atom 構成ファイルまたは XML バインディングを表示するための HTTP 要求を処理します	CICS グループ DFH$WEB2 またはご使用のコピー
COBOL 言語構造 (コピーブック) DFH0W2Q1	XML バインディングの生成に使用された、一時記憶域キュー内のレコードの構造について記述します	SDFHSAMP サンプル・ライブラリー
CICS アイコン `cics-icon.gif` および CICS ロゴ `cics-logo.gif`	サンプル Atom コレクションのアイコンおよびロゴのイメージ	*USSHOME*`/samples/web2.0/image`
URIMAP リソース DFH$W2GI	アイコンとロゴのイメージのための HTTP 要求を処理します	CICS グループ DFH$WEB2 またはご使用のコピー

以下の表に示されているコンポーネントを使用して、サンプル Atom コレクションとの対話を可能にするマッシュアップ Web ページが作成されます。

表 17. マッシュアップ Web ページ用のコンポーネント

コンポーネント	目的	場所
マッシュアップ Web ページ `dfh$w2q1.html`	サンプル Atom コレクションと対話するためのマッシュアップ	*USSHOME*`/samples/web2.0/html`
URIMAP リソース DFH$W2HT	マッシュアップ Web ページのための HTTP 要求を処理します	CICS グループ DFH$WEB2 またはご使用のコピー
スタイルシート `dfh$w2ss.css`	マッシュアップ Web ページの外観を制御します	*USSHOME*`/samples/web2.0/style`
URIMAP リソース DFH$W2SS	スタイルシートの HTTP 要求を処理します	CICS グループ DFH$WEB2 またはご使用のコピー
スクリプト `dfh$w2w2.js`	Atom エントリーからのデータを表示および受信する、マッシュアップ Web ページで使用されるウィジェット	*USSHOME*`/samples/web2.0/script`

表 17. マッシュアップ Web ページ用のコンポーネント (続き)

コンポーネント	目的	場所
URIMAP リソース DFH$W2JS	スクリプト dfh$w2w2.js のための HTTP 要求を処理します	CICS グループ DFH$WEB2 またはご使用のコピー

これらのコンポーネントを使用してサンプル Atom コレクションをセットアップした場合、Web クライアントのコレクション要求の URL は、次のようになります。

```
http://host:port/atom/q/personnel/feed
```

ここで、

- *host* は、このコレクションで使用する TCPIPSERVICE リソースの HOST 属性に定義されている、文字ホスト名、IPv4 アドレス、または IPv6 アドレスです。
- *port* は、このコレクションで使用する TCPIPSERVICE リソースの PORTNUMBER 属性で定義されているポート番号です。

サンプル Atom コレクションを表示するマッシュアップ Web ページの URL は、次のようになります。

```
http://host:port/web2.0/html/dfh$w2q1.html
```

ここで、*host* および *port* は、このコレクションで使用する TCPIPSERVICE リソースからのホスト名とポート番号です。

DFH0W2F1 COBOL サンプル・サービス・ルーチン

サンプル・サービス・ルーチン DFH0W2F1 は、CICS サンプル・ファイル FILEA からのデータを使用する Atom エントリーの GET、POST、PUT、および DELETE 要求を、ユーザー作成サービス・ルーチンが処理する方法について示している COBOL プログラムです。このサービス・ルーチンとその実行手順については、390 ページの『Atom フィード用の DFH0W2F1 COBOL サンプル・サービス・ルーチン』を参照してください。

CICS は、DFH0W2F1 を実行できるように、次の表に示すコンポーネントを提供します。

表 18. DFH0W2F1 を実行するためのコンポーネント

コンポーネント	目的	場所
URIMAP リソース DFH$W2P1	サービス・ルーチンとして DFH0W2F1 を用いながら、FILEA を使用して Atom フィードの HTTP 要求を処理します	CICS グループ DFH$WEB2 またはご使用のコピー
ATOMSERVICE リソース DFH$W2P1	Atom フィードを生成するために使用するリソースを指名します	CICS グループ DFH$WEB2 またはご使用のコピー
Atom 構成ファイル dfh0w2f1.xml	Atom フィード文書および Atom エントリー用のメタデータと構造を提供します	*USSHOME*/samples/web2.0/atom

ATOMSERVICE リソース DFH$W2P1 は FILEA ファイル用の XML バインディングを指名しません。それは、FILEA には Atom エントリーのメタデータとして使用

できるフィールドが含まれていないので、DFH0W2F1 が Atom 構成ファイルで <cics:fieldnames> 要素を使用することがないためです。

これらのコンポーネントを使用して Atom フィードをセットアップした場合、Web クライアントの Atom フィード要求の URL は、次のようになります。

```
http://host:port/atom/p/filea/feed
```

ここで、*host* および *port* は、この Atom フィードで使用する TCPIPSERVICE リソースからのホスト名とポート番号です。

DFH$W2S1 C サンプル・サービス・ルーチン

サンプル・サービス・ルーチン DFH$W2S1 は、ユーザー作成サービス・ルーチンが DFHATOMPARMS コンテナー内のパラメーターを読み取る方法、メタデータおよび内容コンテナー (DFHATOMTITLE や DFHATOMCONTENT など) を更新する方法、および DFHATOMPARMS コンテナーを更新して返す方法を示す、C 言語プログラムです。

出荷時の状態では、DFH$W2S1 はそのコードでセットアップされたデフォルト・データを返すことによって、Atom フィードの GET 要求に応答できます。 Atom エントリーのデータを保持するリソースから必要なレコードを識別してレコードから適切なフィールドを取り出す処理や、POST、PUT、または DELETE 要求に応じてレコードを更新する処理は、示していません。 DFH$W2S1 を使用することにより、Atom フィードの GET 要求に対する応答で、テスト目的のデフォルト・データを提供できます。

CICS は、DFH$W2S1 を実行するためのリソースを提供しません。 DFH$W2S1 を実行するには、DFH$W2S1 に適合した RDO 定義 (EXECKEY(CICS) を指定していなければならない) を作成してインストールし、DFH$W2S1 をサービス・ルーチンとして指定した URIMAP および ATOMSERVICE リソースと Atom 構成ファイルをセットアップする必要があります。

DFH$W2S1 の動作の詳細については、306 ページの『Atom フィード用の DFH$W2S1 C サンプル・サービス・ルーチン』を参照してください。

FILEA のサンプル Atom フィード

CICS では、CICS サンプル・ファイル FILEA を CICS からの直接的な Atom フィードとしてサービス提供するために、一連のリソースが提供されています。

以下の表に示されているコンポーネントを使用して、この Atom フィードをサービス提供します。

表 19. DFH0W2F1 を実行するためのコンポーネント

コンポーネント	目的	場所
URIMAP リソース DFH$W2F1	FILEA を使用して Atom フィードの HTTP 要求を処理します	CICS グループ DFH$WEB2 またはご使用のコピー
ATOMSERVICE リソース DFH$W2F1	Atom フィードを生成するために使用するリソースを指名します	CICS グループ DFH$WEB2 またはご使用のコピー

表 19. DFH0W2F1 を実行するためのコンポーネント (続き)

コンポーネント	目的	場所
Atom 構成ファイル `filea.xml`	Atom フィード文書および Atom エントリー用のメタデータと構造を提供します	*USSHOME*`/samples/web2.0/atom`
XML バインディング `filea.xsdbind` および XML スキーマ `filea.xsd`	CICS に、FILEA 内のレコードの構造について通知します	*USSHOME*`/samples/web2.0/atom`

これらのコンポーネントを使用して Atom フィードをセットアップした場合、Web クライアントの Atom フィード要求の URL は、次のようになります。

`http://`*host*`:`*port*`/atom/f/filea/feed`

ここで、*host* および *port* は、この Atom フィードで使用する TCPIPSERVICE リソースからのホスト名とポート番号です。

第 22 章 Atom フィードからのコレクションの作成

編集可能なコレクションを既存の Atom フィードから作成するには、新しい URIMAP リソース定義と ATOMSERVICE リソース定義、および新しい Atom 構成ファイルをセットアップします。新しい定義では、少しの変更点はありますが、元の Atom フィードのほとんどの設定を使用します。

始める前に

Atom フィードに、リソースからデータを抽出して CICS に提供するサービス・ルーチンが関係している場合は、データをコレクションとして使用可能にする前に 367 ページの『第 23 章 Atom フィードおよび Atom コレクションの管理』を参照して、コレクション内のエントリーに関する GET、POST、PUT、および DELETE 要求に対して適切なアクションを実行するようにサービス・ルーチンを変更する必要があります。手順については、 382 ページの『サービス・ルーチンで Atom コレクション編集要求を処理する方法』を参照してください。

このタスクについて

クライアントは、Atom エントリーのリストを取得して表示することで、通常の Atom フィードと同じようにコレクションを使用することが可能です。そのため、コレクションに関する要件を満たすように Atom フィードの ATOMSERVICE 定義と構成ファイルを変更して、CICS に、元の Atom フィードの代わりにコレクションをすべてのユーザーに送信させることができます。ただし、別々の CICS リソース定義を作成し、異なる URL を使用して Atom エントリーをコレクションとして使用可能にするとともに、引き続きそれらのエントリーをそれぞれフィードとして使用可能にすることが推奨されています。同じデータをフィードおよびコレクションとしてセットアップすることには、追加の作業が関係しますが、次のような重要な利点があります。

- 適切なセキュリティー手段を編集可能なコレクションに適用するとともに、読み取り専用のフィードを自由に使用可能にできます。
- パフォーマンスの優れたフィード文書を大部分のユーザーに配信することで、応答時間を改善することができます。 CICS ではコレクションに追加のナビゲーションが提供されるため、コレクションからエントリーを配信すると、フィードからエントリーを配信する場合よりも多くの処理が必要になります。

Atom フィードから編集可能なコレクションを作成するには、以下のようにします。

手順

1. コレクション用の適切なセキュリティー手段を計画して、コレクション内のエントリーの編集を認証された Web クライアントにのみ許可します。 セキュリティーの詳細については、 395 ページの『第 24 章 Atom フィードのセキュリティー』を参照してください。
2. TCPIPSERVICE リソース定義をセットアップして、Web クライアントがコレクションに関する要求を行うポートに適用されるセキュリティー手段を指定しま

す。 109 ページの『CICS Web サポートの TCPIPSERVICE リソース定義の作成』では、この実行方法が説明されています。

3. Atom フィードに使用しているものとは異なるコレクション用の適切な URL を選択し、コレクションの URL 用の新しい URIMAP リソースを作成します。選択する URL は、Atom フィード用の URL とは異なると共に、同じホスト名を使用してサービスする他の Atom フィードおよびコレクションの URL とも異なる必要があります。
4. 『コレクション用の ATOMSERVICE 定義および Atom 構成ファイルの作成』の手順を実行し、Atom フィード用の既存のファイルに基づいて新しい ATOMSERVICE 定義と Atom 構成ファイルをセットアップします。 コレクションを保護するためにリソースとコマンドのセキュリティーを使用する場合は、Web クライアントのユーザー ID に、参照する ATOMSERVICE 定義およびリソース (サービス・ルーチンで使用される CICS リソースおよびコマンドを含む) に対するアクセス権限があることを確認してください。
5. 358 ページの『Atom サービス文書の作成』の説明に従い、コレクションが含まれる Atom サービス文書を作成します。 362 ページの『Atom カテゴリー文書の作成』の説明に従い、コレクションのカテゴリーを指定する Atom カテゴリー文書を作成することもできます。
6. 365 ページの『Atom 構成ファイルとしての Atom サービス文書またはカテゴリー文書の送信』または 365 ページの『静的応答としての Atom サービス文書またはカテゴリー文書の送信』の説明に従い、Atom サービス文書とカテゴリー文書を配信するための CICS リソース定義をセットアップします。

タスクの結果

これらの作業を完了すると、コレクションは Web クライアントがエントリーを追加、更新、および削除できるようになります。Web クライアントは、サービス文書を取得することによってコレクションの URL を検出できます。

次のタスク

367 ページの『第 23 章 Atom フィードおよび Atom コレクションの管理』には、Web クライアントで HTTP GET、POST、PUT、および DELETE 要求を発行してコレクションを編集する方法と、サービス・ルーチンで編集要求を処理する方法が説明されています。

コレクション用の ATOMSERVICE 定義および Atom 構成ファイルの作成

コレクション用の ATOMSERVICE 定義および Atom 構成ファイルは、同じデータから Atom フィード用の対応するファイルをコピーして小さい変更を加えることにより作成します。

このタスクについて

ATOMSERVICE リソース定義には、さまざまなリソース定義方法に関する情報と、ATOMSERVICE リソース定義属性に関する詳細な参照情報があります。

手順

1. Atom フィード用の Atom 構成ファイルをコピーして、適切な名前に変更します。 ファイルに以下の変更を加えます。
   a. ルート要素を <cics:atomservice type="feed"> から <cics:atomservice type="collection"> に変更します。
   b. <atom:feed> 要素の <atom:title> 子要素をコレクションに適したタイトルに変更します。 <atom:entry> 要素の <atom:title> 子要素を変更する必要はありません。
   c. <atom:feed> 要素の <atom:id> 子要素をコレクションに適した固有 ID に変更します。
   d. <atom:feed> 要素の <atom:link rel="self" href=" "> 子要素の href 属性を変更して、 Web クライアントがコレクションを取得するために使用できる完全パスを指定します。パスの先頭部分は、コレクション用の URIMAP リソース定義で指定した部分パスと一致している必要があります。 例えば、/myatomcoll/* を URIMAP リソース定義内でパス構成要素として指定した場合、Atom 構成ファイルで <atom:link rel="self" href="/myatomcoll/collection.atom"> を指定できます。 URL のスキームとホスト構成要素を省略して、パス構成要素のみを指定することもできます。
   e. プロトタイプ <atom:entry> 要素の <atom:link rel="self" href=" "> 子要素の href 属性を変更して、この標準パスの先頭部分が、コレクション用の URIMAP リソース定義で指定した部分パスと一致するようにします。 パスの残りの部分も変更して、対応するデータからの通常の Atom フィードに使用するパスとは異なるものにすると便利です (ただし、必須ではありません)。パス全体は、コレクション用の <atom:link rel="self" href=" "> 要素で指定した完全パスとは異なっている必要があります。 CICS は、サービス・ルーチンによって提供されたセレクター値または Atom エントリーに適した固有 ID をこのパスに追加して、各 Atom エントリー用の完全なリンクを作成します。

      例えば、/myatomcoll/* を URIMAP リソース定義でパス構成要素として指定し、<atom:link rel="self" href="/myatomcoll/collection.atom"> を Atom 構成ファイルでコレクション全体のリンクとして指定した場合は、<atom:link rel="self" href="/myatomcoll/edit/"> をコレクション内のエントリーの標準パスとして指定できます。 CICS が Atom エントリー文書を発行するとき、CICS は Atom 出版プロトコルでの必要に応じて "self" 属性を "edit" に変更します。 Atom 構成ファイルの中でこの属性を独自に変更しないでください。この例では、CICS は <atom:link rel="edit" href="/myatomcoll/edit/23"> のようなリンクを発行します。
2. Atom フィード用の ATOMSERVICE リソース定義をコピーし、コレクション用の URIMAP リソース定義で指定した名前に変更します。 定義に以下の変更を加えます。
   a. ATOMTYPE 属性を FEED から COLLECTION に変更します。
   b. CONFIGFILE 属性を、コレクション用に作成したばかりの Atom 構成ファイルの名前に変更します。
3. ATOMSERVICE リソース定義、およびコレクション用に作成した対応する URIMAP リソース定義をインストールします。 また、 344 ページの『Atom フ

ィード用の別名トランザクションの作成』の説明に従い、コレクションの別名トランザクション用の TRANSACTION リソース定義を作成して、このリソース定義をインストールすることもできます。

Atom サービス文書の作成

サーバーで使用可能なコレクションをクライアントに知らせるには、Atom サービス文書を作成します。Atom サービス文書には、編集用のコレクションとして使用可能にする Atom フィードのみをリストします。編集用に使用できない通常の Atom フィードは含めません。

このタスクについて

通常は、CICS 領域を介して使用可能なコレクションについて、 1 つの Atom サービス文書のみを作成します。Atom サービス文書は、z/OS UNIX システム・サービスに格納されます。Atom サービス文書は、XML 文書であり、ファイル拡張子は .xml です。このファイルは、任意の XML エディターまたはテキスト・エディターを使用して作成できます。

Atom サービス文書は、CICS が ATOMSERVICE リソース定義を使用して管理する Atom 構成ファイルとして作成するか、CICS Web サポートの静的コンテンツ配信によって配信されるファイルとして作成するかを選択できます。Atom サービス文書を Atom 構成ファイルとして定義するには、追加のリソース定義が必要ですが、サービス文書は Atom フィード用の CICS サポートと統合されることになります。

Atom サービス文書を Atom 構成ファイルとして定義する場合は、文書をルート要素 <cics:atomservice type="service"> で始めます。CICS コレクション用の Atom サービス文書には、このルート要素を除いて CICS 環境に固有の要素は含まれず、Atom 出版プロトコルの仕様 (RFC 5023) で定義されている標準の要素が使用されます。そのため、CICS の資料には、これらの要素に関する詳細な参照情報は含まれていません。Atom サービス文書の要素に関する詳細な参照情報が必要な場合は、RFC 5023 (「*The Atom Publishing Protocol*」) を参照してください。

CICS では、Atom サービス文書の内容は検証されません。以下の手順を正しく実行すれば、 Atom 出版プロトコル仕様に準拠した有効な Atom サービス文書を作成できるはずです。クライアントでサービス文書を読み取る際に問題が発生する場合や、文書の内容が意図したとおりに解釈されない場合は、以下の手順および Atom 出版プロトコル仕様 (RFC 5023) を参照してサービス文書を再確認してください。

手順

1. Atom サービス文書を Atom 構成ファイルとして定義する場合は、文書をルート要素 <cics:atomservice type="service"> で始め、<app:service> 要素を子要素として追加します。

   ```
   <?xml version="1.0"?>
   <cics:atomservice type="service">
    <app:service>
    </app:service>
   </cics:atomservice>
   ```

   Atom サービス文書を Atom 構成ファイルとして定義しない場合は、次のように <app:service> 要素をルート要素として使用します。

```
<?xml version="1.0"?>
<app:service>
</app:service>
```

2. Atom サービス文書のルート要素、つまり <cics:atomservice> 要素または <app:service> 要素には、Atom XML 名前空間および Atom 出版プロトコル用の名前空間の名前空間宣言を含めます。 <cics:atomservice> 要素をルート要素として使用している場合は、CICS Atom XML 名前空間の名前空間宣言も含めます。次に例を示します。

```
<?xml version="1.0"?>
<cics:atomservice type="service"
      xmlns:cics="http://www.ibm.com/xmlns/prod/cics/atom/atomservice"
      xmlns:atom="http://www.w3.org/2005/Atom"
      xmlns:app="http://www.w3.org/2007/app">
 <app:service>
 </app:service>
</cics:atomservice>
```

<app:service> 要素がルート要素である場合、宣言は以下のようになります。

```
<?xml version="1.0"?>
<app:service
      xmlns:atom="http://www.w3.org/2005/Atom"
      xmlns:app="http://www.w3.org/2007/app">
</app:service>
```

3. 少なくとも 1 つの <app:workspace> 要素を <app:service> 要素の子要素として追加した後、 <atom:title> 要素を適切なタイトルを指定して各ワークスペースに追加します。 <app:workspace> 要素は、サービス文書内でコレクションをグループ化するにすぎないので、サーバーやクライアントでこれに関連したアクションを実行する必要はありません。ただし、Atom 出版プロトコルでは、少なくとも 1 つのワークスペースを使用し、各ワークスペースには人間が読んで理解できるタイトルを付けることが要求されています。 この例では、すべてのコレクションを含めることのできる 1 つのワークスペースを示します。

```
<?xml version="1.0"?>
<cics:atomservice type="service"
      xmlns:cics="http://www.ibm.com/xmlns/prod/cics/atom/atomservice"
      xmlns:atom="http://www.w3.org/2005/Atom"
      xmlns:app="http://www.w3.org/2007/app">
 <app:service>
  <app:workspace>
     <atom:title>CICS Atom collections</atom:title>
  </app:workspace>
 </app:service>
</cics:atomservice>
```

4. 各コレクションについて、<app:workspace> 要素 (複数作成した場合は適切な <app:workspace> 要素) の子要素として <app:collection> 要素を追加します。 各 <app:collection> 要素では、以下の必須項目を指定します。

   a. <app:collection> 要素の href 属性。コレクションの Atom 構成ファイルで指定した完全パスを含む、コレクションの完全な URL を示します。 Web クライアントは、GET 要求についてこの URL を使用し、コレクション内の Atom エントリーのリストを取得します。また、POST 要求についてもこの URL を使用し、新しい Atom エントリーをコレクションに送信します。 このリンクが有効であることと、コレクションをサポートするために必要なすべての項目 (URIMAP リソース定義や ATOMSERVICE リソース定義を含む) が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。

b. 子要素として、<atom:title> 要素。コレクションのタイトルを示します。

次に例を示します。

```
<app:workspace>
    <atom:title>CICS Atom collections</atom:title>
  <app:collection
      href="http://www.example.com/customers/customercol.atom">
         <atom:title>Customer collection</atom:title>
  </app:collection>
</app:workspace>
```

5. オプション: 1 つ以上のコレクションでクライアントが新規 Atom エントリーを作成しないようにするには、その <app:collection> 要素の子要素として空の <app:accept> 要素を含めます。 次に例を示します。

```
  <app:collection
      href="http://www.example.com/customers/customercol.atom">
         <atom:title>Customer collection</atom:title>
         <app:accept/>
  </app:collection>
```

クライアントがコレクションを編集するのを防ぐには、追加のセキュリティー手段を実装する必要があります。クライアントは空の <app:accept> 要素を、エントリーを作成できないことを意味する指示として解釈するはずですが、空の <app:accept> 要素があったとしても、 CICS がクライアントによるエントリー作成を防止するわけではありません。

クライアントに Atom エントリーを作成させる場合は、<app:accept> 要素を省略します。CICS ではメディア・リソースがサポートされていないため、<app:accept> 要素を使用してエントリー用の追加メディア・タイプを指定することのないようにしてください。CICS では、Atom エントリー文書 (メディア・タイプは application/atom+xml であり、type=entry 属性はある場合とない場合があります) でないエントリーを作成するクライアント要求は拒否されます。このメディア・タイプはデフォルトなので、<app:accept> 要素を省略した場合、クライアントはこのコレクションでは Atom エントリー文書しか作成できないと理解するはずです。

6. オプション: 1 つ以上のコレクションで Atom エントリーのカテゴリー (これはオプションです) を指定する場合は、サービス文書自体でカテゴリーのリストを提供するか別個のカテゴリー文書への参照を提供するかを決定してください。別個のカテゴリー文書は、カテゴリーのリストが長い場合や、さまざまなコレクション用に同じカテゴリーのリストを再利用する場合に便利です。 Atom 出版プロトコルでは、サービス文書で複数の <app:categories> 要素を使用することが認められているので、例えば、共用 Atom カテゴリー文書への参照とコレクションに固有のカテゴリーのリストの両方を提供することもできます。 方針が決定したら、以下の手順を実行します。

   a. サービス文書でカテゴリーを提供するには、<app:collection> 要素の子要素として <app:categories> 要素を追加した後、 1 つ以上の <atom:category> 要素を <app:categories> 要素の子要素として追加します。各 <atom:category> 要素では、カテゴリーの名前を指定する term 属性を設定します。 CICS では、オプションの scheme 属性と label 属性はサポートされていないため、これらの属性は使用しないでください。 次に例を示します。

```
  <app:collection
      href="http://www.example.com/customers/customercol.atom">
         <atom:title>Customer collection</atom:title>
```

```
      <app:categories>
         <atom:category term="Customers" />
         <atom:category term="Actions" />
      </app:categories>
   </app:collection>
```

指定したカテゴリーが CICS の動作の仕方に影響することはありません。CICS は、ユーザーのリストに含まれていないカテゴリーを指定するクライアント要求を受け入れます。このため、CICS がリソースを直接処理する要求の場合、<app:categories> 要素で fixed="yes" 属性 (これは、サーバーでその他のカテゴリーが許可されないことを示します) を使用しないでください。fixed 属性を省略した場合は、"no" の値が想定されます。サービス・ルーチンを使用してリソースに変更を加える場合は、カテゴリーに基づいてクライアント要求を拒否するようにサービス・ルーチンをコーディングして、そのことを <app:categories> 要素で示すことができます。

b. 別個のカテゴリー文書でカテゴリーを提供するには、 362 ページの『Atom カテゴリー文書の作成』の説明に従い、カテゴリー文書とそこで必要なリソース定義を作成します。次に、<app:collection> 要素の子要素として <app:categories> 要素を追加し、クライアントがカテゴリー文書を取得するために使用できる URL を href 属性で指定します。 次に例を示します。

```
<app:collection
    href="http://www.example.com/customers/customercol.atom">
       <atom:title>Customer collection</atom:title>
    <app:categories href="http://www.example.com/cat/customercol"/>
</app:collection>
```

カテゴリー文書への参照を指定する場合は、 <app:categories> 要素で属性や子要素を使用しないでください。

2 つのコレクションが含まれる Atom サービス文書の例

この Atom サービス文書は Atom 構成ファイルとして定義されており、ルート要素は <cics:atomservice type="service"> です。単一のワークスペースに 2 つのコレクションが含まれています。コレクションの 1 つには、サービス文書内のカテゴリーのリストが含まれています。もう 1 つのコレクションには、別個のカテゴリー文書への参照が含まれています。

```
<?xml version="1.0"?>
<cics:atomservice type="service"
      xmlns:cics="http://www.ibm.com/xmlns/prod/cics/atom/atomservice"
      xmlns:atom="http://www.w3.org/2005/Atom"
      xmlns:app="http://www.w3.org/2007/app">
 <app:service>
  <app:workspace>
    <atom:title>CICS Atom collections</atom:title>
   <app:collection
      href="http://www.example.com/customers/customercol.atom">
         <atom:title>Customer collection</atom:title>
      <app:categories>
         <atom:category term="Customers" />
         <atom:category term="Actions" />
      </app:categories>
   </app:collection>
   <app:collection
      href="http://www.example.com/sysadmin/messagecol.atom">
      <atom:title>Messages from the systems administrator</atom:title>
      <app:categories href="http://www.example.com/cat/messagecol"/>
```

```
   </app:collection>
  </app:workspace>
 </app:service>
</cics:atomservice>
```

次のタスク

Atom 出版プロトコルでは、Atom フィード文書で 1 つの <app:collection> 要素を <atom:feed> 要素の子要素として指定することが許可されています。この配置は、該当データの通常の Atom フィードにコレクションをリンクする際に役立つことがあります。このマークアップを認識できるクライアントは、要求した Atom フィードがコレクションとして使用可能であることがわかります。また、サービス文書を表示しなくても、この URL を使用して新規エントリーを作成したり編集用のエントリーを参照したりできます。

Atom フィード文書で <app:collection> 要素を指定する場合は、Atom サービス文書の <app:collection> 要素およびそのすべての子要素を、<atom:feed> 要素の子要素として、Atom フィード用の Atom 構成ファイルにコピーします。 <cics:feed> 要素やプロトタイプ <atom:entry> 要素の子要素ではないことに注意してください。また、<app:collection> 要素の指定は Atom サービス文書内でも保持する必要があります。将来、Atom サービス文書内の <app:collection> 要素の指定を変更する場合は、Atom 構成ファイル内にコピーした内容に対しても該当する変更を加えます。

次に、 365 ページの『Atom 構成ファイルとしての Atom サービス文書またはカテゴリー文書の送信』または 365 ページの『静的応答としての Atom サービス文書またはカテゴリー文書の送信』の説明に従い、 Atom サービス文書を Web クライアントに送信するためのリソース定義をセットアップします。

Atom カテゴリー文書の作成

コレクション用に長いカテゴリーのリストを提供したり、さまざまなコレクション用に同じカテゴリーのリストを再利用したりする場合は、Atom カテゴリー文書を作成します。短いカテゴリーのリストや専用のカテゴリーのリストの場合、別個のカテゴリー文書を定義する際に必要な値はわずかなので、代わりに Atom サービス文書でカテゴリーを指定します。

このタスクについて

Atom カテゴリー文書は、z/OS UNIX システム・サービスに格納されます。Atom カテゴリー文書は、XML 文書であり、ファイル拡張子は `.xml` です。このファイルは、任意の XML エディターまたはテキスト・エディターを使用して作成できます。

Atom サービス文書の場合と同様、Atom カテゴリー文書は、Atom 構成ファイルとして作成するか、CICS Web サポートの静的コンテンツ配信によって配信されるファイルとして作成するかを選択できます。 358 ページの『Atom サービス文書の作成』で Atom サービス文書について選択したのと同じ方法を選択してください。CICS では、Atom カテゴリー文書の内容は検証されません。

<cics:atomservice type="category"> ルート要素 (使用する場合) を除き、 CICS 内の Atom カテゴリー文書では、RFC 5023 (「*Atom PublishingProtocol*」) で定義されている標準の要素だけが使用されます。詳細な参照情報については、この資料を参照してください。

カテゴリー文書で指定するカテゴリーが CICS の動作に影響することはありません。CICS は、ユーザーの文書に含まれていないカテゴリーを指定するクライアント要求を受け入れます。このため、CICS がリソースを直接処理する要求の場合、<app:categories> 要素で fixed="yes" 属性 (これは、サーバーでその他のカテゴリーが許可されないことを示します) を使用しないでください。サービス・ルーチンを使用してリソースに変更を加える場合は、カテゴリーに基づいてクライアント要求を拒否するようにサービス・ルーチンをコーディングして、そのことを <app:categories> 要素で示すことができます。

リソースで保持されているデータの場合、CICS では各 Atom エントリーについて 1 つのカテゴリーだけがサポートされていることに注意してください。

手順

1. Atom カテゴリー文書を Atom 構成ファイルとして定義する場合は、文書をルート要素 <cics:atomservice type="category"> で始め、<app:categories> 要素を子要素として追加します。

   ```
   <?xml version="1.0"?>
   <cics:atomservice type="category">
    <app:categories>
    </app:categories>
   </cics:atomservice>
   ```

   Atom カテゴリー文書を Atom 構成ファイルとして定義しない場合は、次のように <app:categories> 要素をルート要素として使用します。

   ```
   <?xml version="1.0"?>
   <app:categories>
   </app:categories>
   ```

   CICS ではサポートされていないので、オプションの scheme 属性は使用しないでください。CICS ではカテゴリーの不許可はサポートされていないため、CICS がリソースを直接処理する要求の場合、fixed="yes" 属性は使用しないでください。fixed 属性を省略した場合は、"no" の値が想定されます。サービス・ルーチンを使用してリソースを処理する要求では、サービス・ルーチンでカテゴリーに基づいて要求を拒否する場合に、 fixed="yes" 属性を使用することができます。

2. Atom カテゴリー文書のルート要素、つまり <cics:atomservice> 要素または <app:categories> 要素に、Atom XML 名前空間および Atom 出版プロトコル用の名前空間の名前空間宣言を含めます。 <cics:atomservice> 要素をルート要素として使用している場合は、CICS Atom XML 名前空間の名前空間宣言も含めます。次に例を示します。

   ```
   <?xml version="1.0"?>
   <cics:atomservice type="category"
         xmlns:cics="http://www.ibm.com/xmlns/prod/cics/atom/atomservice"
         xmlns:atom="http://www.w3.org/2005/Atom"
   ```

```
      xmlns:app="http://www.w3.org/2007/app">
 <app:categories>
 </app:categories>
</cics:atomservice>
```

<app:categories> 要素がルート要素である場合、宣言は以下のようになります。

```
<?xml version="1.0"?>
<app:categories
      xmlns:atom="http://www.w3.org/2005/Atom"
      xmlns:app="http://www.w3.org/2007/app">
</app:categories>
```

3. 1 つ以上の <atom:category> 要素を <app:categories> 要素の子要素として追加して、各 <atom:category> 要素で、カテゴリーの名前を指定する term 属性を設定します。 次に例を示します。

```
 <app:categories>
    <atom:category term="Events" />
    <atom:category term="Comments" />
 </app:categories>
```

   CICS では、オプションの scheme 属性と label 属性はサポートされていないため、これらの属性は使用しないでください。

4. 該当する各コレクションで <app:categories> 要素と共に href 属性を指定して、Atom サービス文書にカテゴリー文書の URL 全体を含めます。 この例では、Atom サービス文書で <app:categories> 要素を <app:collection> 要素の子として指定する方法を示します。

```
  <app:collection
      href="http://www.example.com/events/eventcol.atom">
         <atom:title>Events collection</atom:title>
      <app:categories href="http://www.example.com/cat/eventcol"/>
  </app:collection>
```

例

この Atom カテゴリー文書は Atom 構成ファイルとして定義されており、ルート要素は <cics:atomservice type="category"> です。

```
<?xml version="1.0"?>
<cics:atomservice type="category"
      xmlns:cics="http://www.ibm.com/xmlns/prod/cics/atom/atomservice"
      xmlns:atom="http://www.w3.org/2005/Atom"
      xmlns:app="http://www.w3.org/2007/app">
 <app:categories>
    <atom:category term="Events" />
    <atom:category term="Comments" />
 </app:categories>
</cics:atomservice>
```

次のタスク

365 ページの『Atom 構成ファイルとしての Atom サービス文書またはカテゴリー文書の送信』または 365 ページの『静的応答としての Atom サービス文書またはカテゴリー文書の送信』の説明に従い、 Atom カテゴリー文書を Web クライアントに送信するためのリソース定義をセットアップします。

Atom 構成ファイルとしての Atom サービス文書またはカテゴリー文書の送信

Atom サービス文書またはカテゴリー文書をルート要素が <cics:atomservice> である Atom 構成ファイルとして作成した場合は、その文書を送信するための URIMAP および ATOMSERVICE リソース定義をセットアップします。

手順

1. Atom サービス文書またはカテゴリー文書用の URIMAP リソースを作成します。 Atom サービス文書用に選択する URL のパス構成要素は、同じホスト名を使用してサービス提供する Atom フィードまたはコレクションの URIMAP リソース定義で指定するパス構成要素の共通部分で開始して**はなりません**。
2. リソース定義の方法にリストされているいずれかの方法により、Atom 文書の URIMAP リソース定義で指定した名前、および選択したグループを使用して、ATOMSERVICE リソース定義を作成します。
3. STATUS 属性を使用して、ATOMSERVICE リソース定義を使用可能な状態でインストールするか、使用不可の状態でインストールするかを指定します。
4. Atom サービス文書の場合は SERVICE、Atom カテゴリー文書の場合は CATEGORY で、ATOMTYPE 属性を指定します。
5. CONFIGFILE 属性で、Atom 文書用に作成した Atom 構成ファイルの名前およびパスを指定します。
6. ATOMSERVICE リソース定義、および対応する URIMAP リソースをインストールします。 344 ページの『Atom フィード用の別名トランザクションの作成』で別名トランザクション用の TRANSACTION リソース定義を作成した場合は、そのリソースもインストールします。

静的応答としての Atom サービス文書またはカテゴリー文書の送信

ルート要素が <app:service> または <app:categories> である簡潔な Atom サービス文書またはカテゴリー文書を作成した場合は、 CICS Web サポートを介してその文書を静的応答として送信するための URIMAP リソース定義をセットアップします。

手順

1. 76 ページの『CICS 文書テンプレートまたは z/OS UNIX ファイルによる静的 HTTP 応答の提供』に記載されている、z/OS UNIX ファイルを静的応答として送信する場合の計画手順を実行します。
2. 115 ページの『HTTP サーバーとしての CICS の共通 URIMAP 属性の指定』および 118 ページの『HTTP 要求に対する静的応答のための URIMAP 属性の指定』に記載されている、 z/OS UNIX ファイルを静的応答として送信するための URIMAP 定義をセットアップする手順を実行します。 これらの手順を実行する際は、Atom サービス文書またはカテゴリー文書の URIMAP 定義で、以下の選択を行います。
   a. PATH 属性をサービス文書またはカテゴリー文書の適切な URL (`/servicedocument` など) として指定します。 Atom サービス文書または

Atom カテゴリー文書のパスは、指定したホスト名を使用してサービス提供する Atom フィードおよびコレクションのパス構成要素の共通部分で開始してはなりません。

b. USAGE 属性として ATOM ではなく SERVER (HTTP サーバーとして機能する CICS) を指定します。

c. MEDIATYPE 属性として、Atom サービス文書の場合は `application/atomsvc+xml` を、Atom カテゴリー文書の場合は `application/atomcat+xml` を指定します。

d. サービス文書またはカテゴリー文書を含む z/OS UNIX ファイルの名前として HFSFILE 属性を指定します。

第 23 章 Atom フィードおよび Atom コレクションの管理

クライアントは、Atom フィード内の Atom エントリーを参照できます。ただし、Atom フィード内の Atom エントリーは読み取り専用であり、編集できません。クライアントは、Atom コレクション内の Atom エントリーを参照、編集、作成、または削除できます。

サーバー上の使用可能な Atom フィードまたはコレクションを検出するために、クライアントはサーバーにサービス文書を要求します。サービス文書には、クライアントが使用できる Atom フィードおよびコレクションの URL がリストされています。その後、クライアントは、以下のようにサーバーに HTTP 要求を送信することで Atom エントリーと対話できます。

GET 1 つの Atom エントリーまたは Atom エントリーのリストを取得します。Atom エントリーのリストを取得するための GET 要求は、Atom サービス文書で指定されているコレクションの URL に送信されます。 1 つの Atom エントリーを取得するための GET 要求は、エントリーの <atom:link rel="edit"> リンクで指定されているコレクション内の個々の Atom エントリーの URL に送信されます。

POST 新しい Atom エントリーを作成します。POST 要求は、コレクションの URL に送信されます。

PUT クライアントが GET 要求を使用して取得した既存の Atom エントリーを編集します。PUT 要求は、コレクション内の個々の Atom エントリーの URL に送信されます。

DELETE
既存の Atom エントリーを削除します。DELETE 要求は、コレクション内の個々の Atom エントリーの URL に送信されます。

POST および PUT 要求の場合、クライアントは完全な Atom エントリー文書が含まれる要求ボディを送信します (CICS では、Atom コレクションでその他のメディア・タイプはサポートされていません)。通常、PUT 要求では既存のエントリーが部分的に更新されるだけですが、サーバーによって誤って解釈される可能性を防ぐために、クライアントは編集されていない既存エントリーの要素を含むエントリー全体を送信する必要があります。サーバーにより新しいエントリーが追加または既存エントリーが更新された後、適切な HTTP 状況コードの HTTP 応答がクライアントに送信されます。サーバーは、クライアントがエントリー用に提供するメタデータの項目 (Atom ID、日時タイム・スタンプなど) を変更、追加、または削除できます。したがって、クライアントの POST または PUT 要求が成功すると、サーバーは新しいエントリーのコピーも応答の本体として返します。 CICS では、コレクション内の Atom エントリーの PUT 要求用のエンティティー・タグ (ETags) が必要です。サーバーはこのタグにより、クライアントの編集要求が Atom エントリーの最新コピーに基づいて行われていることを確認できます。

RFC 5023 (「*The Atom Publishing Protocol*」。http://www.ietf.org/rfc/rfc5023.txt から入手可能です) には、コレクション内の Atom エントリーと対話を行うためのプロトコルの詳細と、使用される HTTP 要求および応答のいくつかの例が記載されています。

CICS がサービス・ルーチンを介さずにファイルまたは一時記憶域キューからデータを直接抽出するコレクションの場合、CICS は、クライアント要求の処理、リソースの更新、およびクライアントへの応答の返信を行うサーバー・アクションを実行します。これらのアクションによって、ファイルまたは一時記憶域キューの内容が永続的に変更されることに注意してください。 CICS は POST および PUT 要求に対する応答としてレコードを追加または編集します。ファイルに関連した DELETE 要求の場合、レコードを削除できない ESDS を除いて、CICS はレコードを削除します。一時記憶域キューに関連した DELETE 要求の場合、CICS は最初のバイトを 'FF'x に設定することによってレコードを除去します。ご使用の環境でこれらのアクションが適切でない場合は、代わりにサービス・ルーチンを使ってクライアント要求を処理してください。

CICS では、 Atom 出版プロトコル (RFC 5023) でコレクション用として記載されている可能なアクションのいくつかがサポートされていません。主なものは以下のとおりです。

- CICS ではコレクション内のメディア・リソースとメディア・リンクのエントリーはサポートされていません。メディア・リソースは、主にコレクション内で非テキストのコンテンツを整理するための手段として Atom 出版プロトコル (RFC 5023) で指定されています。クライアントがコレクション内でエントリーを作成しようとすると、CICS は Atom エントリー (メディア・タイプは application/atom+xml で、 type=entry パラメーターはある場合とない場合があります) でないクライアント要求を状況コード 415 で拒否します。CICS 用のサービス文書では、<app:accept> 要素で追加のメディア・タイプを指定しないでください。
- CICS では、カテゴリーに基づいてコレクションの Atom エントリーが拒否されることはありません。サービス文書またはカテゴリー文書で <app:categories> 要素を使用して、コレクション内のエントリーについて受け入れられるカテゴリーを指定することができますが、CICS ではクライアントがこれらのカテゴリーに準拠しているかどうかは監視されません。
- パフォーマンス上の理由から、CICS では、最後に編集された時刻 (エントリーの <app:edited> 要素で示されます) の順で自動的にコレクション内に Atom エントリーが返されることはありません。RFC 5023 において、この機能はエントリーの完全なリストについては推奨される要件となっていますが、エントリーの部分リストについては必須の要件となっています。 CICS では、許容される応答時間を維持するとともに部分リストの有用な機能を提供し続けるため、この要件から外れています。サービス・ルーチンを使用して CICS にエントリーを提供する場合は、最後に編集された時刻 (リソースにこの情報を格納できる場合) の順でエントリーを提供することにより、コレクションをこの要件に準拠させることができます。

59 ページの『CICS でサポートされていない Atom の機能』には、あまり重要でない、または特にコレクションとは関係のないその他の非サポート項目のリストが記載されています。

CICS によって提供されるコレクションに、サービス・ルーチンがリソースから抽出して CICS に提供するデータが含まれる場合、CICS はクライアント要求を一連のコンテナーに入れてサービス・ルーチンに渡します。サービス・ルーチンは、リソース内のデータに要求を適用し、クライアントに送信する応答を CICS に返すようにコーディングされている必要があります。この状況では、Atom 出版プロトコル (RFC 5023) と準拠する責任をいくつかの点で CICS と分担することになります。382 ページの『サービス・ルーチンで Atom コレクション編集要求を処理する方法』には、コレクションに関する GET、POST、PUT、および DELETE 要求をサービス・ルーチンで処理する方法が説明されています。

Atom コレクションと (必要な場合は) サービス・ルーチンのセットアップが完了したら、 HTTP プロトコルを処理する任意の適切なクライアントを使用してエントリーを編集できます。要求の作成や応答の表示を行う実際のプロセスは、選択したクライアントによって異なります。GET、POST、PUT、および DELETE 要求を使ってコレクション内のエントリーと対話する方法の詳細については、『Web クライアントによる Atom コレクションの編集』を参照してください。

355 ページの『第 22 章 Atom フィードからのコレクションの作成』
編集可能なコレクションを既存の Atom フィードから作成するには、新しい URIMAP リソース定義と ATOMSERVICE リソース定義、および新しい Atom 構成ファイルをセットアップします。新しい定義では、少しの変更点はありますが、元の Atom フィードのほとんどの設定を使用します。

Web クライアントによる Atom コレクションの編集

Atom コレクション内のエントリーの読み取り、作成、編集、および削除を行うには、HTTP GET、POST、PUT、および DELETE 要求を行える Web クライアントを使用します。

始める前に

CICS は、マッシュアップ Web ページでサポートされるサンプル Atom コレクションを提供します。これを使用すると、コレクションの Web クライアント要求を行い、要求と応答を表示することができます。 CICS 内の Atom コレクションを Web クライアントで編集する方法を調べるには、CICS TS for z/OS バージョン 4.1 インフォメーション・センターにある Web 2.0 シナリオ『従業員情報を処理する Atom フィードの作成』の指示に従って、サンプル Atom コレクションをセットアップし、使用してください。

このタスクについて

無料または市販の多数の Web クライアント・アプリケーションは、Atom フィードまたはコレクションを取得および表示する HTTP GET 要求を出すことができます。ただし、これらすべての Web クライアントに、Atom エントリーを編集するための HTTP POST、PUT、および DELETE 要求を出す機能があるとは限りません。使用するアプリケーションで Atom 形式だけでなく Atom 出版プロトコルがサポートされていることを確認してください。この機能を備えた Web クライアントをご使用の場合、要求を出して応答を確認する手順については、そのクライアントの資料を参照してください。 Web クライアントは、Atom 出版プロトコルをサポートす

る任意のサーバーに対する場合と同様に CICS と対話することができますが、 367 ページの『第 23 章 Atom フィードおよび Atom コレクションの管理』でリストされている特定の機能は使用できません。

使用中の Web クライアント・アプリケーションが Atom エントリーに対する POST、PUT、および DELETE 要求を明確にサポートしていない場合は、独自の HTTP 要求の作成と送信および応答の表示を行える Web クライアント・アプリケーションを使用できます。独自の Web クライアント・アプリケーションを作成して、Atom コレクションに対する POST、PUT、および DELETE 要求を行うこともできます。CICS アプリケーションから Web クライアント要求を行う方法については、141 ページの『HTTP クライアントとしての CICS を介した HTTP 要求』を参照してください。

独自の Web クライアント・アプリケーションを作成して Atom コレクションを編集する場合、または適切な Web クライアントで独自の HTTP 要求を行う場合は、ここで説明される手順に従って要求を行ってください。各手順の詳細情報については、RFC 5023 (「*The Atom Publishing Protocol*」。http://www.ietf.org/rfc/rfc5023.txt から入手可能です) を参照してください。この資料には、コレクション内の Atom エントリーと対話するためのプロトコルの詳細が示されています。 RFC 2616 (「*Hypertext Transfer Protocol -- HTTP/1.1* 」。http://www.ietf.org/rfc/rfc2616.txtから入手可能です) では、 HTTP 要求を作成するためのプロトコルが説明されています。

コレクションには、Web クライアント・アクセスを制御するためのセキュリティー手段が含まれている必要があります。各手順では、ご使用の Web クライアントに Atom エントリーの読み取りと変更を行うための全アクセス権限があることを想定しています。 CICS から提供される Atom フィードおよびコレクションの場合、一部の Web クライアントには項目に対する読み取り専用権限を与えて HTTP GET 要求のみを行えるようにし、他の Web クライアントには UPDATE 権限を与えて HTTP POST、PUT、および DELETE 要求も行えるようにすることができます。Web クライアントにコレクションに対してアクションを実行するための正しい権限が与えられていない場合、CICS から状況コード 403 (Forbidden (禁止)) の HTTP エラー応答が返されます。 Atom フィードおよびコレクションのセキュリティーの詳細については、395 ページの『第 24 章 Atom フィードのセキュリティー』を参照してください。

手順

1. 編集するコレクションの URL がわからない場合や、複数のコレクションを処理できるアプリケーションを作成している場合は、まず、358 ページの『Atom サービス文書の作成』でセットアップした Atom サービス文書に対する HTTP GET 要求を行います。 Atom サービス文書には、サーバーで使用可能なコレクションの URL のリストがあります。 コレクション内のエントリーに使用可能なカテゴリーを確認することもできます。これらのカテゴリーは、サービス文書または別個のカテゴリー文書にリストされています。
2. コレクション内の既存の Atom エントリーのリストを取得するには、371 ページの『Atom フィードまたはコレクションに対する GET 要求の発行』の指示に従って、コレクションの URL に対して HTTP GET 要求を行います。 また、これらの指示は、コレクションとして定義されていない Atom フィードに対する GET 要求にも当てはまります。

3. コレクションから個々の Atom エントリーを取得するには、『Atom フィードまたはコレクションに対する GET 要求の発行』の指示に従って、Atom エントリーの URL に対する HTTP GET 要求を行います。 また、これらの指示は、コレクションとして定義されていない Atom フィードに対する GET 要求にも当てはまります。
4. コレクション内に新しい Atom エントリーを作成するには、376 ページの『Atom コレクションに対する POST 要求』の指示に従って、コレクションの URL に対する HTTP POST 要求を行います。
5. コレクション内の既存の Atom エントリーを編集するには、379 ページの『Atom コレクションに対する PUT 要求』の指示に従って、Atom エントリーの URL に対する HTTP PUT 要求を行います。
6. コレクションから既存の Atom エントリーを削除するには、381 ページの『Atom コレクションに対する DELETE 要求』の指示に従って、Atom エントリーの URL に対する HTTP DELETE 要求を行います。

Atom フィードまたはコレクションに対する GET 要求の発行

Web クライアントは、コレクションの URL に対して HTTP GET 要求を出すことにより Atom フィードまたはコレクションの中の既存の Atom エントリーのリストを取得できます。または、Atom エントリーの URL に対して HTTP GET 要求を出すことにより、Atom フィードまたはコレクションから個々の Atom エントリーを取得できます。

始める前に

コレクションの編集時に編集対象のコレクションの URL がわからない場合や、複数のコレクションを処理できるアプリケーションを作成している場合には、まず、358 ページの『Atom サービス文書の作成』でセットアップした Atom サービス文書に対する HTTP GET 要求を行います。 Atom サービス文書には、サーバーで使用可能なコレクションの URL のリストがあります。 Atom フィードの URL は Atom サービス文書からは得られませんが、通常、該当する場所に公開されています (例えば Web サイト上でフィードに加入する招待として)。

手順

1. Atom フィードまたはコレクション内の既存の Atom エントリーのリストを取得するには、Atom フィードまたはコレクションの URL に対して HTTP GET 要求を行います。この要求は以下の手順で作成します。
   a. 最初の部分は GET メソッドで構成される要求行です。続いて、Atom フィードまたはコレクションの URL のパス構成要素、要求の HTTP バージョンを示す `HTTP/1.1` の順に指定します。 URL にはスキーム (HTTP または HTTPS) およびホスト名を含めることもできます。 要求行の詳細については、14 ページの『HTTP 要求』を参照してください。
   b. この Atom フィードまたはコレクションに対する単一の要求で CICS から送信されるデフォルトの Atom エントリーの数を増減するには、URL の最後に照会ストリングを指定し、名前には `w` (「window (ウィンドウ)」を表す)、値にはこの Atom 文書で受信する Atom エントリーの数を設定します。 次の照会ストリングは、6 つの Atom エントリーを要求します。

      ```
      ?w=6
      ```

c. 要求の HTTP ヘッダーを以下のように作成します。各行は改行します。
   - Host ヘッダーに、Atom フィードまたはコレクションの URL からホスト名を提供します (要求行にホスト名をまだ含めていない場合)。
   - Authorization ヘッダーに、Atom フィードまたはコレクションにアクセスするために必要なセキュリティー情報 (基本認証用のユーザー ID とパスワードなど) を指定します。

   最後の HTTP ヘッダーの後に追加の復帰改行 (CRLF) 文字を入力して、空の行を作成します。

要求ボディは含めないでください。 要求をサーバーに送信します。 Atom フィードまたはコレクション内の Atom エントリーの完全なリストまたは部分リストを含む単一の HTTP 応答である Atom フィード文書が、サーバーから返されます。

注: CICS によってサービス提供される Atom 文書内の要素には、通常、atom: namespace 接頭部は含まれません。 Atom 文書の先頭にある要素で Atom 名前空間がデフォルトの名前空間として定義されるため、子要素では atom: 接頭部は不要です。ただし、CICS 資料で要素について記述する場合には、明確にするために、要素名で atom: 接頭部が使用されています。

2. Atom フィードまたはコレクション内の Atom エントリーの部分リストを受け取った場合、さらに取得するには、次のようにして、追加の部分リストを指定する追加の HTTP GET 要求を Atom フィード文書内のリンクに対して行います。
   a. <atom:link rel="next"> 要素で指定された URL を使用して、次のウィンドウまたはエントリー部分リストを取得します。 CICS は、Atom フィードとコレクションの両方に関してこのリンクを提供します。 上述のように、CICS では子要素に atom: 名前空間接頭部が含まれないため、Atom フィード文書では、要素は <link rel="next"> として出現します。
   b. <atom:link rel="previous"> 要素で指定された URL を使用して、前のエントリー部分リストを取得します。 CICS は、コレクションに関してこのリンクを提供します。
   c. <atom:link rel="first"> 要素で指定された URL を使用して、最初のエントリー部分リストを取得します。 CICS は、コレクションに関してこのリンクを提供します。
   d. <atom:link rel="last"> 要素で指定された URL を使用して、最後のエントリー部分リストを取得します。 CICS は、コレクションに関してこのリンクを提供します。 CICS によってサービス提供される Atom 文書の場合、パフォーマンスを改善するために、この部分リストにはフィード内の最後の Atom エントリーだけが含まれます。 <atom:link rel="previous"> リンクを使用して、これより前のすべての部分リストを取得することができます。

   追加の部分リストへのリンクは、それらを含む Atom 文書が CICS によって発行された時点での Atom フィードまたはコレクションのスナップショットを表していることに注意してください。コレクションで Atom エントリーが追加または削除されると、リンクが無効になる可能性があります。したがって、すぐには変化しないコレクションをブラウズする目的で、短期間でのみリンクを使用できます。
3. Atom フィードまたはコレクションから個々の Atom エントリーを取得するには、エントリーの <atom:link rel="self"> または <atom:link rel="edit"> 要素で指

定されている Atom エントリーの URL に対して HTTP GET 要求を行います。 <atom:link rel="self"> はフィード内の Atom エントリーのリンクを提供し、 <atom:link rel="edit"> はコレクション内の Atom エントリーのリンクを提供します。また、コレクション内の Atom エントリーに <atom:link rel="self"> リンクが含まれる可能性もあります。 この要求は以下の手順で作成します。

a. 最初の部分は GET メソッドで構成される要求行です。続いて、Atom エントリーの URL のパス構成要素、要求の HTTP バージョンを示す HTTP/1.1 の順に指定します。 URL にはスキーム (HTTP または HTTPS) およびホスト名を含めることもできます。 要求行の詳細については、14 ページの『HTTP 要求』を参照してください。

b. 要求の HTTP ヘッダーを以下のように作成します。各行は改行します。

   - Host ヘッダーに、Atom フィードまたはコレクションの URL からホスト名を提供します (要求行にホスト名をまだ含めていない場合)。
   - Authorization ヘッダーに、Atom フィードまたはコレクションにアクセスするために必要なセキュリティー情報 (基本認証用のユーザー ID とパスワードなど) を指定します。

   最後の HTTP ヘッダーの後に追加の復帰改行 (CRLF) 文字を入力して、空の行を作成します。

要求ボディは含めないでください。 要求をサーバーに送信します。 サーバーから、個々の Atom エントリーのコピーが含まれる HTTP 応答が返されます。

例

これは、http://www.mycics.com:80/web20/myfeed という URL の Atom コレクションに対する HTTP 要求です。

```
GET /web20/myfeed HTTP/1.1
Host: www.mycics.com:80
```

これは、http://www.mycics.com:80/web20/entry/7 という URL の Atom エントリーに対する HTTP 要求です。

```
GET /web20/entry/7 HTTP/1.1
Host: www.mycics.com:80
```

このサンプル HTTP 応答には、単一の Atom エントリーに関する要求の応答として CICS が送信する Atom 文書が示されています。この例では Atom フィードと関係のある HTTP ヘッダーだけが示されています。応答には他の HTTP ヘッダーも含まれる可能性があります。 HTTP 応答の ETag ヘッダーは Atom エントリーのエンティティー・タグであり、エントリーを編集する PUT 要求を出す場合には If-Match ヘッダーでこれを使用する必要があります。

```
HTTP/1.1 200 OK
Content-Type: application/atom+xml
Content-Length: 1005
ETag: c4826af12991fb102ef13099c927c2ac24e4caa2

<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app">
    <generator uri="http://www.ibm.com/cics/" version="6.6.0">
    CICS Transaction Server Version 4.1.0
    </generator>
    <link rel="self" href="http://www.mycics.com:80/web20/entry/7"/>
    <link rel="edit" href="http://www.mycics.com:80/web20/entry/7"/>
```

```
    <id>tag:http://www.mycics.com/web20/myfeed,2009-01-20:tsqueue:WB20TSQ:7</id>
    <title>This is entry 7</title>
    <summary>
    Entry 7 is about something to do with feeds...
    </summary>
    <category term="Test Feeds"/>
    <rights>Copyright (c) 2009, Joe Bloggs</rights>
    <published>2008-12-02T15:41:00</published>
    <author>
     <name>Joe Bloggs</name>
     <email>JBloggs@example.com</email>
     <uri>http://www.example.com/JBloggs/</uri>
    </author>
    <contributor>
     <name>John Doe</name>
    </contributor>
    <app:edited>2009-02-02T16:29:36+00:00</app:edited>
    <updated>2009-02-02T16:29:36+00:00</updated>
    <content type="text/xml">
        <SAMPBIND xmlns="http://www.ibm.com/xmlns/prod/cics/atom/bindfile/sampbind">
        <data_field>
        Here is some content for entry 7
        </data_field>
        </SAMPBIND >
    </content>
</entry>
```

このサンプル HTTP 応答に示されている Atom 文書は、Atom コレクションからの Atom エントリー・リストに関する要求の応答として CICS によって送信されます。この例でもまた、Atom フィードと関係のある HTTP ヘッダーだけが示されています。応答には他の HTTP ヘッダーも含まれる可能性があります。

```
HTTP/1.1 200 OK
Content-Type: application/atom+xml
Content-Length: 8661

<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app">
    <generator uri="http://www.ibm.com/cics/" version="6.6.0">
    CICS Transaction Server Version 4.1.0
    </generator>
    <link rel="self" href="http://www.mycics.com:80/web20/myfeed"/>
    <link rel="edit" href="http://www.mycics.com:80/web20/myfeed"/>
    <id>tag:http://www.mycics.com/web20/myfeed,2009-01-20:tsqueue:WB20TSQ</id>
    <title type="text">CICS ATOM FEED TITLE</title>
    <subtitle>CICS ATOM FEED SUBTITLE</subtitle>
    <rights>Copyright (c) 2009, Joe Bloggs</rights>
    <category term="my-first-feed"/>
    <author>
     <name>Joe Bloggs</name>
     <email>JBloggs@example.com</email>
     <uri>http://www.example.com/JBloggs/</uri>
    </author>
    <contributor>
     <name>John Doe</name>
    </contributor>
    <!--------------------------------------------------------------------->
    <entry>
        <link rel="self" href="http://www.mycics.com:80/web20/entry/9"/>
        <link rel="edit" href="http://www.mycics.com:80/web20/entry/9"/>
        <id>tag:http://www.mycics.com/web20/myfeed,2009-01-20:tsqueue:WB20TSQ:9</id>
        <title>This is entry 9</title>
        <summary>
        Entry 9 is about something to do with feeds...
        </summary>
        <category term="Test Feeds"/>
```

```
    <rights>Copyright (c) 2009, Joe Bloggs</rights>
    <published>2008-12-02T15:41:00</published>
    <author>
     <name>Joe Bloggs</name>
     <email>JBloggs@example.com</email>
     <uri>http://www.example.com/JBloggs/</uri>
    </author>
    <contributor>
     <name>John Doe</name>
    </contributor>
    <app:edited>2009-02-02T16:29:36+00:00</app:edited>
    <updated>2009-02-02T16:29:36+00:00</updated>
    <content type="text/xml">
     <SAMPBIND xmlns="http://www.ibm.com/xmlns/prod/cics/atom/bindfile/sampbind">
     <data_field>
     Here is some content for entry 9
     </data_field>
     </SAMPBIND >
    </content>
</entry>
<!--------------------------------------------------------------------->
<entry>
    <link rel="self" href="http://www.mycics.com:80/web20/entry/8"/>
    <link rel="edit" href="http://www.mycics.com:80/web20/entry/8"/>
    <id>tag:http://www.mycics.com/web20/myfeed,2009-01-20:tsqueue:WB20TSQ:8</id>
    <title>This is entry 8</title>
    <summary>
    Entry 8 is about something to do with feeds...
    </summary>
    <category term="Test Feeds"/>
    <rights>Copyright (c) 2009, Joe Bloggs</rights>
    <published>2008-12-02T15:41:00</published>
    <author>
     <name>Joe Bloggs</name>
     <email>JBloggs@example.com</email>
     <uri>http://www.example.com/JBloggs/</uri>
    </author>
    <contributor>
     <name>John Doe</name>
    </contributor>
    <app:edited>2009-02-02T16:29:36+00:00</app:edited>
    <updated>2009-02-02T16:29:36+00:00</updated>
    <content type="text/xml">
     <SAMPBIND xmlns="http://www.ibm.com/xmlns/prod/cics/atom/bindfile/sampbind">
     <data_field>
     Here is some content for entry 8
     </data_field>
     </SAMPBIND >
    </content>
</entry>
<!--------------------------------------------------------------------->
<entry>
    <link rel="self" href="http://www.mycics.com:80/web20/entry/7"/>
    <link rel="edit" href="http://www.mycics.com:80/web20/entry/7"/>
    <id>tag:http://www.mycics.com/web20/myfeed,2009-01-20:tsqueue:WB20TSQ:7</id>
    <title>This is entry 7</title>
    <summary>
    Entry 7 is about something to do with feeds...
    </summary>
    <category term="Test Feeds"/>
    <rights>Copyright (c) 2009, Joe Bloggs</rights>
    <published>2008-12-02T15:41:00</published>
    <author>
     <name>Joe Bloggs</name>
     <email>JBloggs@example.com</email>
     <uri>http://www.example.com/JBloggs/</uri>
    </author>
```

```
        <contributor>
         <name>John Doe</name>
        </contributor>
        <app:edited>2009-02-02T16:29:36+00:00</app:edited>
        <updated>2009-02-02T16:29:36+00:00</updated>
        <content type="text/xml">
         <SAMPBIND xmlns="http://www.ibm.com/xmlns/prod/cics/atom/bindfile/sampbind">
         <data_field>
         Here is some content for entry 7
         </data_field>
         </SAMPBIND >
        </content>
    </entry>
    <!--------------------------------------------------------------------->
    <entry>
        <link rel="self" href="http://www.mycics.com:80/web20/entry/6"/>
        <link rel="edit" href="http://www.mycics.com:80/web20/entry/6"/>
        <id>tag:http://www.mycics.com/web20/myfeed,2009-01-20:tsqueue:WB20TSQ:6</id>
        <title>This is entry 6</title>
        <summary>
        Entry 6 is about something to do with feeds...
        </summary>
        <category term="Test Feeds"/>
        <rights>Copyright (c) 2009, Joe Bloggs</rights>
        <published>2008-12-02T15:41:00</published>
        <author>
         <name>Joe Bloggs</name>
         <email>JBloggs@example.com</email>
         <uri>http://www.example.com/JBloggs/</uri>
        </author>
        <contributor>
         <name>John Doe</name>
        </contributor>
        <app:edited>2009-02-02T16:29:36+00:00</app:edited>
        <updated>2009-02-02T16:29:36+00:00</updated>
        <content type="text/xml">
         <SAMPBIND xmlns="http://www.ibm.com/xmlns/prod/cics/atom/bindfile/sampbind">
         <data_field>
         Here is some content for entry 6
         </data_field>
         </SAMPBIND >
        </content>
    </entry>
</feed>
```

Atom コレクションに対する POST 要求

Web クライアントは、コレクションの URL に対して HTTP POST 要求を出すことにより、新しい Atom エントリーをコレクション内に作成できます。

始める前に

編集するコレクションの URL がわからない場合や、複数のコレクションを処理できるアプリケーションを作成している場合は、まず、358 ページの『Atom サービス文書の作成』でセットアップした Atom サービス文書に対する HTTP GET 要求を行います。 Atom サービス文書には、サーバーで使用可能なコレクションの URL のリストがあります。

手順

1. HTTP POST 要求の最初の部分は POST メソッドで構成される要求行です。続いて、コレクションの URL のパス構成要素、要求の HTTP バージョンを示す

HTTP/1.1 の順に指定します。 URL にはスキーム (HTTP または HTTPS) およびホスト名を含めることもできます。 要求行の詳細については、 14 ページの『HTTP 要求』を参照してください。

2. 要求の HTTP ヘッダーを以下のように作成します。各行は改行します。
   - Host ヘッダーに、コレクションの URL からホスト名を提供します (要求行にホスト名をまだ含めていない場合)。
   - Authorization ヘッダーに、コレクションにアクセスするために必要なセキュリティー情報 (基本認証用のユーザー ID とパスワードなど) を指定します。
   - Content-Type ヘッダーに、値 application/atom+xml;type=entry を指定します。
   - Content-Length ヘッダーに、メッセージ・ボディの長さをバイト単位 (オクテット) で指定します。このヘッダーの値は、メッセージ・ボディを記述した後で入力します。

   最後の HTTP ヘッダーの後に追加の復帰改行 (CRLF) 文字を入力して、空の行を作成します。
3. 送信する Atom エントリー用の XML マークアップが含まれるメッセージ・ボディをセットアップします。
   a. コレクション内に Atom エントリーが既にある場合、GET 要求を行って、そのコレクション用のフィードまたはエントリー文書を取得し、コレクションの既存の Atom エントリーをメッセージ・ボディにコピーします。
   b. コレクション内にまだ Atom エントリーがない場合、このトピック内の例に基づいて最初の Atom エントリーの XML マークアップを作成します。
   c. Atom エントリーの先頭にある <entry> タグに名前空間宣言 xmlns="http://www.w3.org/2005/Atom" が含まれていることを確認し、この名前空間宣言がない場合には追加します。 CICS から送信される Atom 文書では、通常 atom: 名前空間接頭部は要素には含まれていません。 Atom 名前空間は、Atom 文書の先頭で要素内のデフォルト名前空間として定義されるので、子要素に atom: 接頭部を使用する必要はありません。 CICS 文書では、明確にするためにこの接頭部を要素名で使用します。要求内の要素で atom: 名前空間接頭部を使用する場合、名前空間宣言を xmlns:atom="http://www.w3.org/2005/Atom" に変更してください。
   d. Atom エントリーの前に要素 <?xml version="1.0" ?> を追加します。
4. ご使用の Atom エントリーの該当する要素に必要とするデータを入力し、使用しない要素は削除します。 CICS が送信する Atom エントリー文書に存在するすべての要素に、データを指定する必要はありません。エントリーを CICS によって管理されているコレクションに送信する場合は、CICS リソースに格納されていない要素や、CICS リソースに格納されているが CICS またはサービス・ルーチンによって生成される要素を省略できます。通常では、少なくとも以下の要素を省略できます。
   - <atom:updated> 要素などの、タイム・スタンプが含まれる要素 (サービス・ルーチンがこれらの要素に関するユーザー入力を受け入れる場合は除く)
   - <atom:id> 要素
   - <atom:contributor> 要素
   - <atom:link> 要素

- <atom:rights> 要素

省略できるかどうかが不明な要素は含めてください。 CICS またはサービス・ルーチンでは、必要ない場合にその要素が無視されます。 RFC 5023 には、Atom エントリーで使用可能な要素の完全なリストが記載されています。 Atom エントリーで必須の要素、許可される要素、または使用されない要素の要約については、 341 ページの『CICS 用の Atom 要素のリファレンス』を参照してください。各要素の内容の詳細については、 336 ページの『<atom:entry> 要素』を参照してください。

5. 要求をサーバーに送信します。

タスクの結果

サーバーから、状況コード 200 (要求の正常終了を示す) の HTTP 応答、または適切なエラー応答が送信されます。要求が成功した場合、応答には新しい Atom エントリーのコピーが含まれています。サーバーからの応答に含まれる Atom エントリーを元の送信内容と比べて検査し、新しいエントリーに問題がないことを確認してください。

エラー応答を受信する場合、応答のメッセージ・ボディを読み取り、 423 ページの『付録 C. CICS Web サポートの HTTP 状況コード・リファレンス』で CICS が Web クライアントに提供している状況コードのリストを参照してください。 HTTP 状況コード 400 (Bad Request or Invalid Request (不正な要求または無効な要求)) など、要求の問題が存在する可能性があることをエラー応答が示している場合は、CICS 領域がメッセージ DFHML0100 を発行したかどうかを確認してください。CICS は z/OS XML システム・サービスのパーサーを使用して、ご使用の Atom エントリーの XML マークアップを構文解析します。パーサーがマークアップで問題を検出すると、CICS は、パーサーからの戻りコードと理由コードが含まれるメッセージ DFHML0100 を発行します。戻りコードと理由コードの解説については、http://www.ibm.com/servers/eserver/zseries/zos/xml/ で入手可能な「*z/OS XML System Services User's Guide and Reference*」を参照してください。

例

これは、`http://www.mycics.com:80/web20/myfeed` という URL のコレクションの中に新しい Atom エントリーを作成する要求です。

```
POST /web20/myfeed HTTP/1.1
Host: www.mycics.com:80
Content-Type: application/atom+xml;type=entry
Content-Length: 763

<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom">
    <title>This is my posted entry</title>
    <summary>
    This is my new posted entry
    </summary>
    <category term="Test Feeds"/>
    <author>
     <name>Joe Bloggs</name>
     <email>JBloggs@example.com</email>
     <uri>http://www.example.com/JBloggs/</uri>
    </author>
    <content type="text/xml">
     <SAMPBIND xmlns="http://www.ibm.com/xmlns/prod/cics/atom/bindfile/sampbind">
```

```
      <data_field>
       Here is content for my posted entry
     </data_field>
     </SAMPBIND >
    </content>
</entry>
```

CICS は次のような応答を送ります。

```
HTTP/1.1 201 Created
Content-Type: application/atom+xml;type=entry
Content-Length: 1029

<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app">
    <generator uri="http://www.ibm.com/cics/" version="6.6.0">
    CICS Transaction Server Version 4.1.0
    </generator>
    <link rel="self" href="http://www.mycics.com:80/web20/entry/10"/>
    <link rel="edit" href="http://www.mycics.com:80/web20/entry/10"/>
    <id>tag:http://www.mycics.com/web20/myfeed,2009-01-20:tsqueue:WB20TSQ:10</id>
    <title>This is my posted entry</title>
    <summary>
    This is my new posted entry
    </summary>
    <category term="Test Feeds"/>
    <rights>Copyright (c) 2009, Joe Bloggs</rights>
    <published>2009-04-23T15:00:44+00:00</published>
    <author>
     <name>Joe Bloggs</name>
     <email>JBloggs@example.com</email>
     <uri>http://www.example.com/JBloggs/</uri>
    </author>
    <app:edited>2009-04-23T15:00:44+00:00</app:edited>
    <updated>2009-04-23T15:00:44+00:00</updated>
    <content type="text/xml">
     <SAMPBIND xmlns="http://www.ibm.com/xmlns/prod/cics/atom/bindfile/sampbind">
      <data_field>
       Here is content for my posted entry
      </data_field>
     </SAMPBIND >
    </content>
</entry>
```

Atom コレクションに対する PUT 要求

Web クライアントは、エントリーの <atom:link rel="edit"> 要素で指定された Atom エントリーの URL に対して HTTP PUT 要求を出すことにより、コレクション内の既存の Atom エントリーを編集できます。

このタスクについて

個別の Atom エントリーの URL を使用して、一度に編集できるのは 1 つの Atom エントリーだけです。 1 つの要求で複数の Atom エントリーを編集することはできません。 CICS は、コレクションの URL に対して実行される PUT 要求はリジェクトします。

手順

1. エントリーの <atom:link rel="edit"> 要素で指定されている Atom エントリーの URL に対して HTTP GET 要求を出すことにより、エントリーの現在のコピーを取得して、応答の ETag HTTP ヘッダーからそのエンティティー・タグを取得

します。 371 ページの『Atom フィードまたはコレクションに対する GET 要求の発行』では、この実行方法が説明されています。

2. HTTP PUT 要求の最初の部分は PUT メソッドで構成される要求行です。続いて、Atom エントリーの URL のパス構成要素、要求の HTTP バージョンを示す `HTTP/1.1` の順に指定します。 URL にはスキーム (HTTP または HTTPS) およびホスト名を含めることもできます。 要求行の詳細については、 14 ページの『HTTP 要求』を参照してください。
3. 要求の HTTP ヘッダーを以下のように作成します。各行は改行します。
   - Host ヘッダーに、Atom エントリーの URL からホスト名を提供します (要求行にホスト名をまだ含めていない場合)。
   - Authorization ヘッダーに、コレクションにアクセスするために必要なセキュリティー情報 (基本認証用のユーザー ID とパスワードなど) を指定します。
   - If-Match ヘッダーに、既存の Atom エントリーに関する取得済みのエンティティー・タグを指定します。このチェックをオーバーライドする必要がある場合は、エンティティー・タグの代わりにアスタリスクを指定することにより、Atom エントリーが取得後に別のエージェントによって変更されている場合でも編集内容を適用するようにサーバーに指示することができます。このオプションは注意して使用してください。
   - Content-Type ヘッダーに、値 `application/atom+xml;type=entry` を指定します。
   - Content-Length ヘッダーに、メッセージ・ボディの長さをバイト単位 (オクテット) で指定します。このヘッダーの値は、メッセージ・ボディを記述した後で入力します。

   最後の HTTP ヘッダーの後に追加の復帰改行 (CRLF) 文字を入力して、空の行を作成します。
4. Atom エントリーをメッセージ・ボディにコピーし、その前に要素 <?xml version="1.0" ?> を追加します。 <entry> タグに、名前空間宣言 xmlns="http://www.w3.org/2005/Atom" が含まれていることを確認してください。
5. 変更する Atom エントリー内の要素の内容を編集します。 CICS は正確なタイム・スタンプを提供します。サービス・ルーチンも同じ動作を行う必要があるため、エントリー内のタイム・スタンプを更新しないでください。 各要素の内容の詳細については、 336 ページの『<atom:entry> 要素』を参照してください。
6. 要求をサーバーに送信します。

タスクの結果

要求が受け入れられると、サーバーから、状況コード 200 (要求の正常終了を示す) の HTTP 応答、または適切なエラー応答が送信されます。CICS は編集された Atom エントリーのコピーを応答ボディとして戻すため、必要に応じて変更内容を確認できます。

POST 要求でエラー応答を受信する場合、応答のメッセージ・ボディを読み取り、 423 ページの『付録 C. CICS Web サポートの HTTP 状況コード・リファレンス』で CICS が Web クライアントに提供している状況コードのリストを参照してください。要求の問題を示しているようなエラー応答の場合は、CICS 領域からのメッセージ DFHML0100 を確認し、戻りコードと理由コードの説明を「*z/OS XML System*

Services ユーザーズ・ガイドおよび解説書」で調べてください。

例

これは、http://www.mycics.com:80/web20/entry/10 という URL の Atom エントリーを編集する HTTP PUT 要求です。

```
PUT /web20/entry/10 HTTP/1.1
Host: www.mycics.com:80
Content-Type: application/atom+xml;type=entry
Content-Length: 1034
If-Match: c4826af12991fb102ef13099c927c2ac24e4caa2

<?xml version="1.0" encoding="utf-8"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:app="http://www.w3.org/2007/app">
    <generator uri="http://www.ibm.com/cics/" version="6.6.0">
    CICS Transaction Server Version 4.1.0
    </generator>
    <link rel="self" href="http://www.mycics.com:80/web20/entry/10"/>
    <link rel="edit" href="http://www.mycics.com:80/web20/entry/10"/>
    <id>tag:http://www.mycics.com/web20/myfeed,2009-01-20:tsqueue:WB20TSQ:10</id>
    <title>This is my updated entry</title>
    <summary>
    This is my new updated entry
    </summary>
    <category term="Test Feeds"/>
    <rights>Copyright (c) 2009, Joe Bloggs</rights>
    <published>2009-04-23T15:00:44+00:00</published>
    <author>
     <name>Joe Bloggs</name>
     <email>JBloggs@example.com</email>
     <uri>http://www.example.com/JBloggs/</uri>
    </author>
    <app:edited>2009-04-23T15:00:44+00:00</app:edited>
    <updated>2009-04-23T15:00:44+00:00</updated>
    <content type="text/xml">
     <SAMPBIND xmlns="http://www.ibm.com/xmlns/prod/cics/atom/bindfile/sampbind">
      <data_field>
       Here is new content for my updated entry
      </data_field>
     </SAMPBIND >
    </content>
</entry>
```

Atom コレクションに対する DELETE 要求

Web クライアントは、エントリーの <atom:link rel="edit"> 要素で指定された Atom エントリーの URL に対して HTTP DELETE 要求を出すことにより、既存の Atom エントリーをコレクションから削除できます。

このタスクについて

コレクション内の Atom エントリーは、個々の Atom エントリーの URL を使用して一度に 1 つしか削除できません。コレクション全体を一度に削除することはできません。 CICS は、コレクションの URL に対して出された DELETE 要求はリジェクトします。

手順

1. HTTP DELETE 要求の最初の部分はメソッド DELETE で構成される要求行です。続いて、Atom エントリーの URL のパス構成要素、要求の HTTP バージョンを示す HTTP/1.1 の順に指定します。 URL にはスキーム (HTTP または

HTTPS) およびホスト名を含めることもできます。 要求行の詳細については、14 ページの『HTTP 要求』を参照してください。

2. 要求の HTTP ヘッダーを以下のように作成します。各行は改行します。
   - Host ヘッダーに、Atom エントリーの URL からホスト名を提供します (要求行にホスト名をまだ含めていない場合)。
   - Authorization ヘッダーに、コレクションにアクセスするために必要なセキュリティー情報 (基本認証用のユーザー ID とパスワードなど) を指定します。

   最後の HTTP ヘッダーの後に追加の復帰改行 (CRLF) 文字を入力して、空の行を作成します。 メッセージ本体を含めないでください。
3. 要求をサーバーに送信します。

タスクの結果

要求が受け入れられると、サーバーから、状況コード 200 (要求の正常終了を示す) の HTTP 応答、または適切なエラー応答が送信されます。応答には、削除したエントリーのコピーは含まれません。

例

これは、`http://www.mycics.com:80/web20/entry/10` という URL の Atom エントリーを削除する要求です。

```
DELETE /web20/entry/10 HTTP/1.1
Host: www.mycics.com:80
```

サービス・ルーチンで Atom コレクション編集要求を処理する方法

サービス・ルーチンによって提供される既存の Atom フィードからコレクションを作成するときには、そのコレクション内のエントリーに関する GET、POST、PUT、および DELETE 要求に対して適切なアクションを実行するようにプログラムを更新する必要があります。

CICS とサービス・ルーチンの間のインターフェースは、基本的には通常の Atom フィードに対するのと同じようにコレクションに対しても機能します。CICS はユーザー・プログラムに対して、クライアント要求に関する情報を DFHATOMPARMS コンテナーで提供します。ユーザー・プログラムは要求を分析した後、DFHATOMPARMS コンテナーと (必要に応じて) その他の文字コンテナーを CICS に返すことによって応答する必要があります。サンプル・サービス・ルーチン DFH$W2S1 は、これを行う方法を示しています。

コレクションに関連した追加のタスクとして、プログラムでは、このフィードの Atom エントリーのデータを保持するリソース (データベース、ファイル、その他のリソース) 内のレコードを変更することによってクライアント要求を処理する必要があるかもしれません。リソースから取得される 1 つの Atom エントリーのデータを単純に戻す代わりに、プログラムは次のように応答する必要があるかもしれません。

- POST 要求の場合、新しいレコードを追加して、クライアント提供の新規 Atom エントリー用のデータをそれに入れます。

- PUT 要求の場合、クライアントによる変更要求に従って既存のレコードを編集します。
- GET 要求の場合、レコードを削除します。

要求の方式によっては、プログラムからの応答は、要求された変更を確認する 1 つの HTTP 状況表示行だけで、それ以上のデータが含まれない可能性もあります。サービス・ルーチンによる変更はリソースの内容を永続的に変更するため、適切なセキュリティー手段を実施する必要があります。

POST および PUT メソッドを使用するクライアント要求の場合、CICS は DFHREQUEST というコンテナーで要求ボディを提供します。クライアントは完全な Atom エントリー文書が含まれる要求ボディを送信する必要があります。プログラムでは、Atom エントリー文書のマークアップを解析し、Atom エントリーの要素をリソース内のレコードのフィールドにマッチさせるとともに、要素の内容を使用して新しいリソース・レコードの作成や既存のリソース・レコードの更新を行う必要があります。また、一部の要素 (日付やタイム・スタンプなど) の内容を提供する必要もあります。

Atom 出版プロトコル (RFC 5023) では、サーバーがクライアントから送信されたエントリーのメタデータや内容をかなり自由に変更することが許可されています。サービス・ルーチンでは通常、結果として生じるエントリーが Atom 形式の要件を満たす限り、リソース・レコードにとって最良のアプローチである場合にクライアント要求内の要素の内容を無視またはオーバーライドすることができます。無視またはオーバーライドするようにサービス・ルーチンをコーディングしたい特定の要素に関して不明な点がある場合は、RFC 4287 および RFC 5023 を参照してそのアクションが受け入れられることを確認してから、コーディングを進めてください。

サービス・ルーチンは、コレクションに関連するクライアント要求を受信したときに、このセクションで説明するタスクを実行する必要があります。これらの説明では、Web クライアントからの Atom フィードに関する HTTP GET 要求に応答するサービス・ルーチンを既に作成していることが想定されています。(283 ページの『Atom エントリー・データを提供するプログラムの作成』の説明を参照してください。また、サンプル・サービス・ルーチン DFH$W2S1 にこれが示されています。)

これらのタスクを実行するようにサービス・ルーチンがコーディングされていて、CICS 領域でリソースとコマンドのセキュリティーがアクティブになっているとともにコレクション用に使用されている場合は、 Web クライアントのユーザー ID に、サービス・ルーチンで使用される CICS リソースおよびコマンドに対する適切なアクセス権限があることを確認してください。Atom コレクションを保護するためのセキュリティー手段の詳細については、395 ページの『第 24 章 Atom フィードのセキュリティー』を参照してください。

Atom コレクションに対する GET 要求の処理

Web クライアントが Atom コレクションに対する GET 要求を出したとき、サービス・ルーチンは、コレクションから 1 つのエントリーまたは一連のエントリーを取り出して応答する必要があります。

手順

1. EXEC CICS GET CONTAINER コマンドを使用して、要求に関する情報が含まれる DFHATOMPARMS コンテナー内のデータを取得します。 サンプル・サービス・ルーチン DFH$W2S1 は、これを行う方法を示しています。 287 ページの『DFHATOMPARMS コンテナー』には、CICS がこのコンテナーで渡すすべてのパラメーターが記載されています。
2. **ATMP_HTTPMETH** パラメーターの値を確認して、要求メソッドを識別します。 GET、POST、PUT、および DELETE 以外のメソッドについては、CICS からエラーまたは適切な応答が返されます。
3. プログラムがこのインスタンスで CICS に返す必要のある Atom エントリーを識別するには、DFHATOMPARMS コンテナーの **ATMP_ATOMTYPE** パラメーターと **ATMP_SELECTOR** パラメーターの値を使用します。
   a. **ATMP_SELECTOR** の値がヌルで、**ATMP_ATOMTYPE** の値が "collection" の場合、クライアントは特定の Atom エントリーを指定しませんでした。可能な場合は、コレクション内の最後に編集された Atom エントリーを返します。 リソースにこのデータを格納できない場合は、コレクションに最後に追加された Atom エントリーを返します。
   b. **ATMP_SELECTOR** にセレクター値が含まれていて、**ATMP_ATOMTYPE** の値が "collection" の場合は、セレクター値で識別されるエントリーを返します。 この値の組み合わせは、クライアントがコレクションの 2 番目または後続のエントリーを要求していることを示します。
   c. **ATMP_SELECTOR** にセレクター値が含まれていて、**ATMP_ATOMTYPE** の値が "entry" の場合は、セレクター値で識別されるエントリーを返します。 この値の組み合わせは、クライアントがコレクション内の 1 つの既知の Atom エントリーを要求していることを示します。
4. GET 要求に必要な Atom エントリーを識別した後は、通常の Atom フィードの場合と同じようにコレクションからそのエントリーを返します。ただし、以下の追加手順を実行してください。
   a. DFHATOMPARMS コンテナーの **ATMP_EDITED** パラメーターを使用して、この Atom エントリーが最後に編集された日時を返します。 DFHATOMPARMS コンテナーのリソース・ハンドリング・パラメーターを使用している場合、**ATMP_EDITED_FLD** パラメーターにはリソース内の関連フィールドの名前と長さが含まれています。 リソースにこのデータが格納されていない場合は、スペースを返してください。CICS では現在の日時が想定されます。 287 ページの『DFHATOMPARMS コンテナー』には、このフィールドで必要な形式が記載されています。
   b. DFHATOMPARMS コンテナーの **ATMP_ETAGVAL** パラメーターを使用して、Atom エントリーのエンティティー・タグと長さを返します。 エンティティー・タグを作成するには、EXEC CICS BIF DIGEST コマンドを使用してリソース・レコードの SHA-1 ダイジェストを計算するか、または別の適切な方法を使用して HTTP/1.1 プロトコル要件に準拠するエンティティー・タグを作成できます。
   c. **ATMP_ATOMTYPE** の値が "collection" (クライアントが複数のエントリーを要求していることを意味する) である場合、エントリーが最後に編集された日時についてのデータがリソースに格納されているならば、コレクション内の次の Atom エントリーのセレクター値として **ATMP_NEXTSEL** パラメーターを返しま

す。 265 ページの『Atom エントリーの順序』には、複数の Atom エントリーを返す順序が説明されています。編集の順にエントリーを返すと、Atom 出版プロトコルに準拠するのに役立ちます。

d. **ATMP_ATOMTYPE** の値が "collection" である場合、コレクション内の 1 つ前、最初、および最後の Atom エントリーのセレクター値として **ATMP_PREVSEL**、**ATMP_FIRSTSEL**、および **ATMP_LASTSEL** パラメーターを返します。 CICS はこれらの値を使用することにより、コレクション内のエントリーに関する他の部分リストのリンクが入る <atom:link> 要素を構成します。 これらのパラメーターの詳細については、 287 ページの『DFHATOMPARMS コンテナー』を参照してください。

通常の Atom フィードのエントリーを返す手順については、 283 ページの『Atom エントリー・データを提供するプログラムの作成』を参照してください。

Atom コレクションに対する POST 要求の処理

Web クライアントが Atom コレクションに対して POST 要求を行うとき、サービス・ルーチンはコレクション内に新しいエントリーを作成して、そのコピーをクライアントに戻す必要があります。

このタスクについて

Web クライアントは完全な Atom エントリー文書を HTTP POST 要求のボディに提供し、 CICS はこの要求ボディを DFHREQUEST コンテナー内のサービス・ルーチンに渡します。

Atom エントリーのデータを保持するリソースに、関連付けられた XMLTRANSFORM リソースとの XML バインディングがある場合、CICS の機能を使用して、Web クライアントが提供する Atom エントリー文書のマークアップを構文解析するために XML をアプリケーション・データに変換できます。 XMLTRANSFORM リソースが使用可能な場合、CICS はその名前を DFHATOMPARMS コンテナー内の **ATMP_XMLTRANSFORM** パラメーターに提供します。 TRANSFORM XMLTODATA コマンドに関する詳細と、データ・マッピング機能の使用方法については、「*CICS アプリケーション・プログラミング・ガイド*」を参照してください。

Atom エントリーのデータを保持するリソースの XML バインディングがない場合、ユーザーのサービス・ルーチンの言語に応じて他の機能を使用できます。

- ユーザー・プログラムが C またはアセンブラーで作成されている場合は、 z/OS XML システム・サービスのパーサーを使用してエントリーのマークアップを解析できます。
- ユーザー・プログラムが Enterprise COBOL で作成されている場合は、XML PARSE verb (DFH0W2F1 という COBOL サービス・ルーチンのサンプルを参照) を使用します。
- プログラムが Enterprise PL/I で記述されている場合、PLISAXA ライブラリー関数を使用します。

手順

1. EXEC CICS GET CONTAINER コマンドを使用して、要求に関する情報が含まれる DFHATOMPARMS コンテナー内のデータを取得します。 サンプル・サービス・ルーチン DFH$W2S1 は、これを行う方法を示しています。 287 ページの『DFHATOMPARMS コンテナー』には、CICS がこのコンテナーで渡すすべてのパラメーターが記載されています。
2. **ATMP_HTTPMETH** パラメーターの値を確認して、要求メソッドを識別します。 GET、POST、PUT、および DELETE 以外のメソッドについては、CICS からエラーまたは適切な応答が返されます。
3. EXEC CICS GET CONTAINER コマンドを使用して、新しい Atom エントリーが含まれる DFHREQUEST コンテナー内のデータを取得します。 サンプル・サービス・ルーチン DFH$W2S1 は、これを行う方法を示しています。 CICS では、Atom フィードでその他のメディア・タイプがサポートされていないため、Atom エントリー (メディア・タイプは application/atom+xml であり、type=entry パラメーターはある場合とない場合があります) として記述されていないクライアント要求は状況コード 415 とともに拒否されます。
4. CICS 機能または別の適切な機能を使用して XML をアプリケーション・データに変換すると、Atom エントリーの XML マークアップが解析され、CICS でサポートされて Atom エントリーのデータを保持するリソースにデータを格納するための適切なフィールドがある、Atom エントリーの全要素が識別されます。 CICS でサポートされる Atom エントリーの全要素のリストと説明については、336 ページの『<atom:entry> 要素』を参照してください。 CICS でサポートされていない要素、サービス・ルーチンで認識されない要素、または Atom エントリーのデータを保持するリソース内のレコードのフィールドに格納できない要素は、すべて無視します。 Atom 構成ファイルは、リソースにデータを格納できないすべての必須要素に適切なデフォルト値を提供するように事前にセットアップされている必要があります。

   ヒント: Web クライアントが無効な XML マークアップまたはデータが含まれる要求を発信したことが分かった場合、応答コード `atmp_resp_invalid_request` でその要求をリジェクトします。
5. レコード内に Atom エントリーのデータを保持する新しいリソースを作成して、そのフィールドに識別した要素の内容を取り込みます。 サービス・ルーチンには、クライアントの性質や Atom エントリーのデータを保持するリソースの重要度に応じて、適切だと思える任意のレベルのエラー・チェックを含めることができます。 クライアントから提供されたデータをオーバーライドする必要がある場合はそのようにすることができますが、 RFC 4287 および RFC 5023 を参照し、そのオーバーライドが有効であることを検証する必要があります。 CICS では、Web クライアントから提供された Atom ID は無視されるので、サービス・ルーチンでも無視する必要があります。 DFHATOMPARMS コンテナー内の **ATMP_ATOMID** パラメーターで CICS によって渡されたプロトタイプ Atom ID を完成させて作成された Atom ID を代わりに使用するか、別の有効な形式を使用してください。 Atom ID の形式について詳しくは、 269 ページの『Atom エントリーの Atom ID』を参照してください。
6. リソースにエントリーが最後に編集された時刻 (<app:edited> 要素)、最後に更新された時刻 (<atom:updated> 要素)、または最初に公開された時刻 (<atom:published> 要素) に関するデータが格納される場合は、EXEC CICS

ASKTIME コマンドおよび EXEC CICS FORMATTIME コマンドを使用して、現在日時の日時スタンプを RFC 3339 形式で作成し、このタイム・スタンプをそれらのフィールドに取り込みます。 クライアントによって <atom:updated>、<atom:published>、または <app:edited> 要素で日時スタンプが提供される場合は、精度と妥当性を確実にするために新しい日時スタンプを生成することができます。 日時スタンプを扱う方法について詳しくは、 267 ページの『Atom エントリーの日時スタンプ』を参照してください。

7. クライアント要求が成功した場合は、通常の Atom フィードの場合と同様に新しいエントリーを返して、新しいリソース・レコードに配置したばかりのデータを提供します。ただし、通常の処理に対して以下の例外があります。
   a. **ATMP_NEXTSEL** パラメーターは返しません。
   b. EXEC CICS BIF DIGEST コマンドを使用して新しいリソース・レコードの SHA-1 ダイジェストを計算するか、または別の適切な方法を使用して HTTP/1.1 プロトコル要件に準拠するエンティティー・タグを作成します。 DFHATOMPARMS コンテナーの **ATMP_ETAGVAL** パラメーターを使用して、Atom エントリーのエンティティー・タグと長さを結果として返します。
   c. **ATMP_EDITED** パラメーターについては、リソースに格納した日時スタンプを返すか、スペースを返して CICS に現在日時を提供させることができます (ただし、Atom 構成ファイルで代わりのデフォルト値を指定していない場合)。

   サンプル・サービス・ルーチン DFH$W2S1 には、Atom エントリーを CICS に戻す方法が示されています。 CICS は、HTTP 状況コード 201 の応答を作成し、新しい Atom エントリーの URI をクライアントに渡すための Location ヘッダーと、一致する Content-Location ヘッダーを提供します (これにより、クライアントは応答が Atom エントリーを完全に表したものであることを認識できます)。クライアントは、メッセージ・ボディ内のエントリーを検査して、その要求で提供されたデータに対して行ったすべての変更を確認できます。
8. クライアント要求が失敗した場合は、DFHATOMPARMS コンテナーの **ATMP_RESPONSE** パラメーターを使用して、適切な応答コードを返します。 エラー応答が返された場合、CICS は適切なデフォルトの HTTP エラー応答を作成して、Web クライアントに送信します。 287 ページの『DFHATOMPARMS コンテナー』には、ユーザーが使用できる応答コード、および CICS が各ケースで送信する HTTP エラー応答のリストが記載されています。

Atom コレクションに対する PUT 要求の処理

Web クライアントが Atom コレクションに対して PUT 要求を行うとき、サービス・ルーチンはエントリーを更新して、要求が正常に完了したかどうかを示す必要があります。

このタスクについて

POST 要求によって、Web クライアントは完全な Atom エントリー文書を HTTP PUT 要求のボディに提供し、CICS はこの要求ボディを DFHREQUEST コンテナー内のサービス・ルーチンに渡します。 Atom エントリーのデータを保持するリソースに、関連付けられた XMLTRANSFORM リソースとの XML バインディングがある場合、CICS の機能を使用して、Web クライアントが提供する Atom エントリー文書のマークアップを構文解析するために XML をアプリケーション・データに変

換できます。 Atom エントリーのデータを保持するリソースの XML バインディングがない場合、 385 ページの『Atom コレクションに対する POST 要求の処理』にリストされている他の機能の 1 つを使用して Atom エントリー文書のマークアップを構文解析できます。

手順

1. EXEC CICS GET CONTAINER コマンドを使用して、要求に関する情報が含まれる DFHATOMPARMS コンテナー内のデータを取得します。 サンプル・サービス・ルーチン DFH$W2S1 は、これを行う方法を示しています。 287 ページの『DFHATOMPARMS コンテナー』には、CICS がこのコンテナーで渡すすべてのパラメーターが記載されています。
2. **ATMP_HTTPMETH** パラメーターの値を確認して、要求メソッドを識別します。 GET、POST、PUT、および DELETE 以外のメソッドについては、CICS からエラーまたは適切な応答が返されます。
3. EXEC CICS GET CONTAINER コマンドを使用して、更新された Atom エントリーが含まれる DFHREQUEST コンテナー内のデータを取得します。 サンプル・サービス・ルーチン DFH$W2S1 は、これを行う方法を示しています。
4. **ATMP_SELECTOR** パラメーターを使用して、Atom エントリーのデータを保持するリソースから、クライアントが更新する Atom エントリーのデータを含むレコードを選択します。
5. EXEC CICS BIF DIGEST コマンドを使用して Atom エントリーのデータを含むリソース内の現行レコードの SHA-1 ダイジェストを計算するか、またはサービス・ルーチンの別の方法を使用してエンティティー・タグを計算し、**ATMP_ETAGVAL** で提供されたエンティティー・タグと比較します。 タグが一致しない場合、Atom エントリーのデータは Web クライアントがそのコピーを取得した後で他のエージェントによって変更されているので、その要求を応答コード `atmp_resp_etag_no_match` で拒否する必要があります。 エンティティー・タグがアスタリスクの場合、Web クライアントはこのプロセスをオーバーライドすることを選択しているので、要求を受け入れる必要があります。
6. CICS 機能または別の適切な機能を使用して XML をアプリケーション・データに変換すると、Atom エントリーの XML マークアップが解析され、CICS でサポートされて Atom エントリーのデータを保持するリソースにデータを格納するための適切なフィールドがある、Atom エントリーの全要素が識別されます。 CICS でサポートされる Atom エントリーの全要素のリストと説明については、 336 ページの『<atom:entry> 要素』を参照してください。 CICS でサポートされていない要素、サービス・ルーチンで認識されない要素、または Atom エントリーのデータを保持するリソース内のレコードのフィールドに格納できない要素は、すべて無視します。 Atom 構成ファイルは、リソースにデータを格納できないすべての必須要素に適切なデフォルト値を提供するように事前にセットアップされている必要があります。

   ヒント: Web クライアントが無効な XML マークアップまたはデータが含まれる要求を発信したことが分かった場合、応答コード `atmp_resp_invalid_request` でその要求をリジェクトします。
7. レコード内のフィールドを変更するために識別した要素の内容を使用して、Atom エントリーのデータを保持するリソース内のレコードを更新します。 PUT 要求ボディでは、変更済み要素と未変更の要素を含む完全な Atom エント

リーをクライアントが提供することが想定されているため、理論的には、比較を行って変更されたフィールドを確認しなくても、リソース・レコード内のすべてのフィールドを更新できるはずです。ただし、クライアントの性質およびリソースの重要度に応じて、内容の形式に関する特定の要件があるフィールドについてエラー・チェックを実行し、変更することが論理的ではないフィールド (Atom ID や最初に公開された時刻など) の内容をクライアントが変更していないことを確認することもできます。 クライアントから提供されたデータをオーバーライドする必要がある場合はそのようにすることができますが、 RFC 4287 および RFC 5023 を参照し、そのオーバーライドが有効であることを検証する必要があります。

8. リソースにエントリーが最後に編集された時刻 (<app:edited> 要素) および最後に更新された時刻 (<atom:updated> 要素) のデータが格納される場合は、EXEC CICS ASKTIME コマンドおよび EXEC CICS FORMATTIME コマンドを使用して、現在日時の日時スタンプを RFC 3339 形式で作成し、このタイム・スタンプをそれらのフィールドに取り込みます。 クライアントによって <atom:updated> 要素または <app:edited> 要素で日時スタンプが提供される場合、 それらは変更されずに返された以前の日時スタンプである可能性があるので、無視します。 日時スタンプを扱う方法について詳しくは、 267 ページの『Atom エントリーの日時スタンプ』を参照してください。
9. クライアント要求が成功した場合は、通常の Atom フィードの場合と同様に更新済みのエントリーを返して、更新済みのリソース・レコードからのデータを提供します。ただし、通常の処理に対して以下の例外があります。
   a. **ATMP_NEXTSEL** パラメーターは返しません。
   b. EXEC CICS BIF DIGEST コマンドを使用して更新されたリソース・レコードの SHA-1 ダイジェストを計算するか、または別の適切な方法を使用して HTTP/1.1 プロトコル要件に準拠するエンティティー・タグを作成します。 DFHATOMPARMS コンテナーの **ATMP_ETAGVAL** パラメーターを使用して、Atom エントリーの新しいエンティティー・タグと長さを結果として返します。
   c. **ATMP_EDITED** パラメーターについては、リソースに格納した日時スタンプを返すか、スペースを返して CICS に現在日時を提供させることができます (ただし、Atom 構成ファイルで代わりのデフォルト値を指定していない場合)。

   サンプル・サービス・ルーチン DFH$W2S1 には、Atom エントリーを CICS に戻す方法が示されています。 CICS は、要求の正常終了を示す HTTP 状況コード 200 の応答を作成します。クライアントは、メッセージ・ボディ内のエントリーを検査して、その要求で提供されたデータに対して行ったすべての変更を確認できます。
10. クライアント要求が失敗した場合は、DFHATOMPARMS コンテナーの **ATMP_RESPONSE** パラメーターを使用して、適切な応答コードを返します。 エラー応答が返された場合、CICS は適切なデフォルトの HTTP エラー応答を作成して、Web クライアントに送信します。 287 ページの『DFHATOMPARMS コンテナー』には、ユーザーが使用できる応答コード、および CICS が各ケースで送信する HTTP エラー応答のリストが記載されています。

Atom コレクションに対する DELETE 要求の処理

Web クライアントが Atom コレクションに対する DELETE 要求を出したとき、サービス・ルーチンは、エントリーを削除して要求が成功したかどうかを示す必要があります。

手順

1. EXEC CICS GET CONTAINER コマンドを使用して、要求に関する情報が含まれる DFHATOMPARMS コンテナー内のデータを取得します。 サンプル・サービス・ルーチン DFH$W2S1 は、これを行う方法を示しています。 287 ページの『DFHATOMPARMS コンテナー』には、CICS がこのコンテナーで渡すすべてのパラメーターが記載されています。
2. **ATMP_HTTPMETH** パラメーターの値を確認して、要求メソッドを識別します。 GET、POST、PUT、および DELETE 以外のメソッドについては、CICS からエラーまたは適切な応答が返されます。
3. **ATMP_SELECTOR** パラメーターを使用することにより、Atom エントリーのデータを保持するリソースから、クライアントによる削除対象の Atom エントリーのデータが入ったレコードを選択します。
4. **ATMP_SELECTOR** パラメーターのセレクター値に対応するリソース内のレコードを削除した後、DFHATOMPARMS コンテナーの **ATMP_RESPONSE** パラメーターを使用してゼロの応答コードを返します。 CICS では DFHATOMPARMS コンテナー内の残りのパラメーターは無視されるので、それらのパラメーターに変更を加える必要はありません。 CICS から、状況コード 200 (要求の正常終了を示す) の応答がクライアントに送信されます。
5. 要求を実行しない場合は、代わりの応答コードを **ATMP_RESPONSE** パラメーターに返します。 287 ページの『DFHATOMPARMS コンテナー』には、ユーザーが使用できる応答コード、および CICS が各ケースで送信する HTTP エラー応答のリストが記載されています。

Atom フィード用の DFH0W2F1 COBOL サンプル・サービス・ルーチン

サンプル・サービス・ルーチン DFH0W2F1 は、CICS サンプル・ファイル FILEA からのデータを使用する Atom エントリーの GET、POST、PUT、および DELETE 要求を処理する COBOL プログラムです。これらの対話は、独自のサービス・ルーチン内でリソースを処理するためのモデルとして使用できます。

DFH0W2F1 を実行して Web クライアント要求を処理し、テストまたはデモンストレーションの目的で FILEA サンプル・ファイルからデータを提供できます。
DFH0W2F1 を実行する前に、以下のタスクを完了する必要があります。

1. EXECKEY(CICS) を必ず指定する DFH0W2F1 用の適切な RDO 定義を作成してインストールします。
2. FILEA VSAM ファイルを作成して、このファイル用の適切な RDO 定義をインストールします。適切な RDO 定義は、CICS 提供のサンプル RDO グループである DFH$FILA 内にあります。
3. FILEA ファイルが使用可能で、開かれていることを確認します。

CICS では、DFH$WEB2 グループにサンプルの URIMAP リソースと ATOMSERVICE リソースが提供されていて、DFH0W2F1 を実行するために使用できます。これらのリソースはどちらも DFH$W2P1 という名前です。これらのリソースは、サンプル Atom コレクションをセットアップするのと同じ方法でセットアップできます。サンプル Atom コレクションのセットアップの指示に従ってください。DFH0W2F1 を使用して Atom コレクションとして FILEA サンプルにアクセスする Web クライアント要求の URL は以下のとおりです。

```
http://host:port/atom/p/filea/feed
```

ここで、

- *host* は、このコレクションで使用する TCPIPSERVICE リソースの HOST 属性に定義されている、文字ホスト名、IPv4 アドレス、または IPv6 アドレスです。
- *port* は、このコレクションで使用する TCPIPSERVICE リソースの PORTNUMBER 属性で定義されているポート番号です。

ATOMSERVICE リソース DFH$W2P1 は Atom 構成ファイル dfh0w2f1.xml の名前を指定します。このファイルは、Atom フィード文書および Atom エントリーのメタデータと構造を提供します。 dfh0w2f1.xml は、z/OS UNIX 上の CICS ファイルのルート・ディレクトリーの /samples/web2.0/ サブディレクトリー (CICS システム初期設定パラメーター USSHOME で指定します) にあります。 USSHOME のデフォルト値は /usr/lpp/cicsts/cicsts42 です。ATOMSERVICE リソース DFH$W2P1 は FILEA ファイル用の XML バインディングを指名しません。それは、FILEA には Atom エントリーのメタデータとして使用できるフィールドが含まれていないので、DFH0W2F1 が Atom 構成ファイルで <cics:fieldnames> 要素を使用することがないためです。

DFH0W2F1 サンプル・プログラムでは、以下のタスクが実行されます。

Working-Storage section

- 応答コード用の定数が含まれるコピーブック DFHW2CNO をコピーします。
- 作業用ストレージにローカル変数をセットアップします。
- CICS に付属のコピーブック DFH0CFIL で説明されている、FILEA レコード・レイアウトの定義を組み込みます。
- Atom エントリーの内容を保持するための XML タグおよびフィールドを含むプロトタイプをセットアップします。それぞれの XML 要素の間には、復帰改行 (X'0D25') が使用されます。サービス・ルーチンは、FILEA ファイルのレコードを読み取り、レコード内のフィールドからこのプロトタイプにデータを挿入し、それを DFHATOMCONTENT コンテナーに戻すことによって、Atom エントリーの内容を生成します。
- DFHATOMTITLE および DFHATOMSUMMARY コンテナーで使用される、プロトタイプ Atom エントリー・タイトルおよび要約をセットアップします。 FILEA ファイル内のレコードにはタイトルおよび要約が含まれませんが、サービス・ルーチンがファイル・レコードからの情報を使用してそれらを構成します。サービス・ルーチンが提供するタイトルおよび要約は、Atom 構成ファイルで指定されたデフォルトのタイトルおよび要約を置き換えます。

- FILEA を読み取れない場合に Atom エントリーに戻すエラーを示す内容をセットアップします。これはデモンストレーションのためのものなので、実動 Atom フィードには適していません。
- Atom エントリーを構文解析するときに使用される XML スタックをセットアップします。
- Atom エントリーのエンティティー・タグ (ETag) を入れるストレージをセットアップします。

Linkage section

- DFHATOMPARMS コンテナーのレイアウトが記述されているコピーブック DFHW2APO をコピーします。
- DFHATOMPARMS コンテナー内の任意のパラメーターに使用できる可変長ストリングを定義します。
- DFHREQUEST コンテナーで渡された、Web クライアントからの要求ボディが含まれるストレージを記述します。
- CICS がコレクション用に追加のナビゲーション・リンクを作成するために使用する、このファイル内の最初、最後、直後、および直前のレコードのセレクター値を入れるためのストレージを定義します。

メインプログラム

- EXEC CICS GET CONTAINER コマンドを使用して、DFHATOMPARMS コンテナーからパラメーターを取得します。
- ATMP_SELECTOR パラメーターからセレクター値を取得します。セレクター値は、要求された Atom エントリーのデータを含む FILEA 内のレコードのキーです。 FILEA で、レコードのキーは固有のカスタマー番号です。 FILEA のキーは、正確に 6 桁でなければなりません。 DFH0W2F1 では、正しい長さになるようにキー値に埋め込み処理が行われることはありません。 CICS がセレクター値を提供しない場合、サービス・ルーチンはファイル内の最初のレコードを読み取り、そのキーを ATMP_SELECTOR パラメーター・ストレージにコピーします。サービス・ルーチンは、CICS が KSDS ファイルに対して行うように、キーを昇順にしてレコードを戻します。
- エンティティー・タグ (Etag) 値が提供されている場合は、If-Match ヘッダーのエンティティー・タグを保管します。
- 要求用の HTTP メソッドを取得します。サービス・ルーチンは、HTTP メソッド名を使用して後続のアクションを判別します。
  - GET 要求の場合、サービス・ルーチンはキーがセレクター値と等しいレコードを読み取り、その内容を Atom エントリーとして RETURN-FILE-CONTENT プロシージャーを使用して戻します。
  - PUT 要求の場合、サービス・ルーチンは更新用読み取り機能を使用して、キーがセレクター値と等しいレコードを読み取ります。次に、読み取ったばかりのレコードの ETag 値を計算し、If-Match ヘッダーから受け取った値と比較します。値が等しくない場合、ファイルは更新用読み取り機能によってアンロックされているので、サービスはエラー応答を戻します。 ETags が一致する場合、サービス・ルーチンは PARSE-REQUEST-BODY プロシージャーを使用して Web クライアントから提供された更新済みの Atom エントリーを解析し、そのデータを使用して FILEA 内のレコードを更新します。ファイル・レコー

ドは、REWRITE 関数で書き込まれます。その後、サービス・ルーチンは RETURN-FILE-CONTENT プロシージャーを使用して、更新済みの Atom エントリーのコピーを Web クライアントに戻します。

– POST 要求の場合、サービス・ルーチンは PARSE-REQUEST-BODY プロシージャーを使用して Web クライアントから提供された更新済みの Atom エントリーを解析し、そのデータを使用して新しいファイル・レコードを記述します。 POST 要求で Web クライアントによって提供されたカスタマー番号がこの新しいファイル・レコードのキー値となり、その番号のレコードが既に存在する場合にはサービス・ルーチンがエラーを戻します。その後、サービス・ルーチンは RETURN-FILE-CONTENT プロシージャーを使用して、新しい Atom エントリーのコピーを Web クライアントに戻します。

– DELETE 要求の場合、キーがセレクター値と等しいレコードが削除されます。

PARSE-REQUEST-BODY および XML-HANDLER プロシージャー

PARSE-REQUEST-BODY プロシージャーで、サービス・ルーチンは EXEC CICS GET CONTAINER コマンドを使用して、POST または PUT 要求のボディで Web クライアントから提供された Atom エントリーである DFHREQUEST コンテナーの内容を取得します。その後、それは XML PARSE 関数を使用して Atom エントリーを構文解析します。

XML-HANDLER プロシージャーは、提供される Atom エントリーの要素を分析し、 XML スタックを使用して、提供される Atom エントリーの <atom:content> 要素内に各 XML 要素があるかどうか (「IN-CONTENT」フラグ) を追跡し、各要素のネスト・レベルを記録します。「IN-CONTENT」で、名前が認識される要素が検出されると、 UPDATE-RECORD-FIELD プロシージャーはそのデータを FILEA レコード内に保管します。サービス・ルーチンは、<atom:content> 要素の外部にある Atom エントリー内のメタデータ要素、および <atom:content> 要素の内部にあって名前が認識されない要素を無視します。それは、これらの要素からのデータを FILEA レコード内に保管できないためです。

COBOL XML パーサーは、XML 名前空間の接頭部に対する精密な分析は試行しません。このパーサーを使用している場合、提供される Atom エントリー内の <atom:content> 要素には接頭部「atom」が含まれる必要があります。また、内容の中の XML 要素には名前空間の接頭部があってはなりません。

RETURN-FILE-CONTENT プロシージャー

GET、POST、および PUT 要求の場合、RETURN-FILE-CONTENT プロシージャーは FILEA ファイル内のレコードからのデータを Atom エントリーの内容として戻し、Atom エントリー用のタイトルおよび要約を作成して戻します。プロシージャーは、以下のステップを実行します。

- FILEA レコードのフィールドからのデータを、Working-Storage section にセットアップされたプロトタイプ Atom エントリー内容に入れます。結果として生じる XML 構造は、EXEC CICS PUT CONTAINER コマンドによって DFHATOMCONTENT コンテナーに入れられます。
- プロトタイプ Atom エントリーのタイトルを完成させて、DFHATOMTITLE コンテナーに保管します。このことを示すフラグが、ATMP-OPTIONS-OUT パラメーターに対して設定されます。

- Atom エントリーの要約を作成し、DFHATOMSUMMARY コンテナーに保管します。このことを示すフラグが、ATMP-OPTIONS-OUT パラメーターに対して設定されます。
- EXEC CICS BIF DIGEST コマンドを使用して FILEA レコード用に ETag を生成し、この値を DFHATOMPARMS コンテナーの ATMP-ETAGVAL パラメーターに戻します。
- 呼び出し

ADD-NAVIGATION-LINKS プロシージャー

ADD-NAVIGATION-LINKS プロシージャーは、FILEA ファイル内の最初、最後、直後、および直前のレコードのセレクター値 (つまりキー値) を戻します。セレクター値は、一時作業域 (TWA) に保管されます。 CICS は、Atom コレクション文書用に追加のナビゲーションを作成するために、セレクター値を使用します。

第 24 章 Atom フィードのセキュリティー

CICS Web サポートは、Atom コレクションおよび (必要な場合に) Atom フィードへの Web クライアントのアクセスを制御するための適切なセキュリティー・プロトコルおよび認証方式を提供します。ユーザーは、CICS リソースおよびコマンド・セキュリティーを使用して、Atom フィードまたはコレクションを配信するために使用するリソースを保護できます。

RFC 5023 では、認証を使用して Atom コレクションを保護することが推奨されています。Atom フィードのデータを編集可能なコレクションとして使用可能にすると、Web クライアントが新しいエントリーの挿入、既存のエントリーの変更、またはエントリーの削除を行えるようになります。そのため、Web クライアントの ID を検証し、信頼できるクライアントにのみコレクションへのアクセスを許可する必要があります。特に、コレクションにビジネス・データを含めている場合は注意してください。Web クライアントが編集できない通常の Atom フィードは、通常、セキュリティーの制限なく任意の加入者が使用できるようになります。ただし、機密のビジネス・データが含まれている、または特定のユーザーのみを対象にした Atom フィードについては、アクセスを制限する必要があります。

RFC 4287 および RFC 5023 には、Atom 文書でのデジタル署名および暗号化の使用に関する説明があります。CICS では Atom 文書のデジタル署名および暗号化はサポートされていませんが、RFC 4287 と準拠するため、署名が含まれる Atom 文書が拒否されることはありません。

CICS Web サポートには、Atom フィードまたはコレクションを無許可のアクセスや更新から保護するために使用できる以下のセキュリティー機能があります。

SSL または TLS セキュリティー・プロトコル
RFC 5023 では、コレクションに対する最小レベルのセキュリティー・プロトコルとして、 Transport Layer Security (TLS) 1.0 を使用することが推奨されています。CICS では、Secure Sockets Layer (SSL) 3.0 プロトコルおよび Transport Layer Security (TLS) 1.0 プロトコルがサポートされています。「*CICS RACF Security Guide*」には、SSL および TLS によって提供される機能の説明があります。

HTTP 基本認証
RFC 5023 では、コレクションに関する最小レベルの認証として、HTTP 基本認証を使用することが推奨されています。 22 ページの『HTTP 基本認証』では、このメカニズムについて説明されています。

クライアント証明書認証
クライアント証明書認証はより安全なクライアント認証方式で、信頼のおける第三者機関 (または認証局) が発行し、 SSL 暗号化を使用して送信されるクライアント証明書が使用されます。「*CICS RACF Security Guide*」ではこの仕組みについて説明されています。

CICS Web サポートのこれらの機能をセットアップする際、CICS が Atom フィードまたはコレクションの Web クライアント要求を受け取るポートの

TCPIPSERVICE 定義の属性を使用して、Atom フィードまたはコレクションにそれらの機能を適用することができます。 CICS Web サポート用の SSL サポートをセットアップする方法については、「*CICS RACF Security Guide*」のを参照してください。

第 4 部 CICS ビジネス・ロジック・インターフェース

CICS ビジネス・ロジック・インターフェースについての情報。

第 25 章 CICS ビジネス・ロジック・インターフェースの概要

CICS ビジネス・ロジック・インターフェースを使用すると、CICS HTTP リスナーを介して呼び出しをしなくても、Web 対応のビジネス・アプリケーションにリンクすることができます。

例えば、z/OS で実行されている Web サーバーは、外部 CICS インターフェース (EXCI) を使用して、CICS ビジネス・ロジック・インターフェースを使用するアプリケーションにリンクすることができます。このようにして、 Web クライアントは、CICS に直接接続する方法ではなく、中間 Web サーバーを介して CICS アプリケーションと通信することができます。

インターフェースの参照情報については、 485 ページの『付録 G. DFHWBBLI、CICS ビジネス・ロジック・インターフェースに関する参照情報』を参照してください。

CICS ビジネス・ロジック・インターフェースの使用方法

CICS アプリケーション・プログラムにリンクできる環境であればどこからでも CICS ビジネス・ロジック・インターフェースを呼び出すことができます。

例えば、次のとおりです。

- CICS アプリケーション・プログラムから **EXEC CICS LINK** コマンドを発行することができる。
- 外部 CICS インターフェース (EXCI) を使用することができる。
- クライアントから外部呼び出しインターフェース (ECI) を使用することができる。
- ONC RPC クライアントから CICS ONC RPC サポートを使用することができる。

CICS ビジネス・ロジック・インターフェースは、以下のものにより使用されます。

- CICS Web サーバー・プラグイン。このプラグインは、外部 CICS インターフェースを使用して CICS ビジネス・ロジック・インターフェースを起動します。

処理の例

EXCI または ECI を使用する MVS アプリケーションからの要求を CICS ビジネス・ロジック・インターフェースが処理する方法の例です。

400 ページの図 20 は、EXCI を使用する MVS アプリケーションからの要求を、CICS ビジネス・ロジック・インターフェースが処理する方法を示しています。

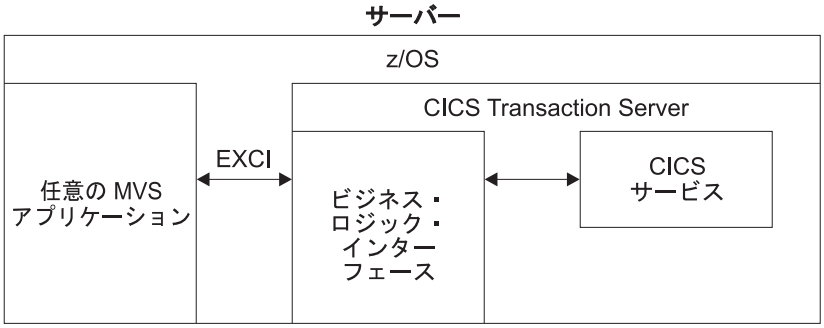


図 20. EXCI からの要求の処理

1. MVS アプリケーションは、 CICS ビジネス・ロジック・インターフェース用のパラメーターを含む COMMAREA を作成します。
2. MVS アプリケーションは、 EXCI を使用して CICS ビジネス・ロジック・インターフェースを呼び出します。
3. CICS ビジネス・ロジック・インターフェースは、要求されたサービスを呼び出し、すべての出力を COMMAREA に戻します。

図 21 は、ECI を使用する CICS クライアントからの要求を、CICS ビジネス・ロジック・インターフェースが処理する方法を示しています。

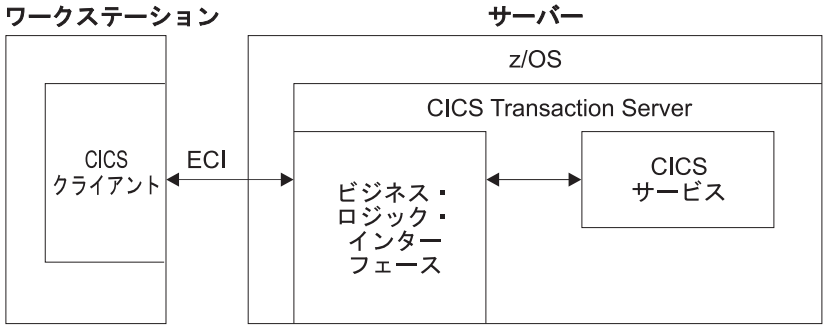


図 21. ECI からの要求の処理

1. ワークステーション環境で実行されているクライアントは、CICS ビジネス・ロジック・インターフェース用のパラメーターを含む COMMAREA を作成します。
2. クライアントは ECI を使用して CICS ビジネス・ロジック・インターフェースを呼び出します。
3. CICS ビジネス・ロジック・インターフェースは、要求されたサービスを呼び出し、すべての出力を COMMAREA に戻します。

ECI は SNA プロトコル、または TCP/IP を介する SNA 接続を可能とする TCP62 のいずれかで動作します (詳細については、「*CICS Family: Client/Server Programming*」を参照してください)。

要求処理での制御フロー

使用する機能と、その機能のカスタマイズ方法を決定するには、 CICS ビジネス・ロジック・インターフェースのコンポーネントの対話方法を理解しておく必要があります。

CICS ビジネス・ロジック・インターフェースを使用したプログラムの呼び出し

図 22 は、CICS ビジネス・ロジック・インターフェースを経由したプログラムへの制御フローを示しています。 CICS ビジネス・ロジック・インターフェースにアクセスするには、PROGRAM DFHWBBLI への LINK コマンドを使用します。

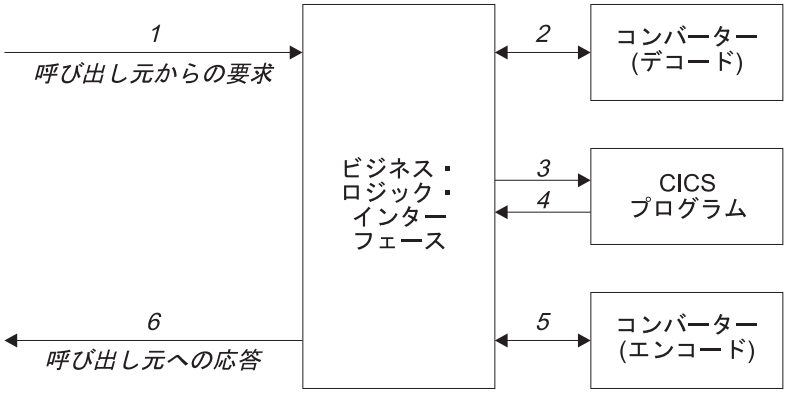


図 22. CICS ビジネス・ロジック・インターフェースによるプログラムの呼び出し制御フロー

1. CICS ビジネス・ロジック・インターフェースに対する要求が到着します。
2. 呼び出し元がコンバーターを要求すると、CICS ビジネス・ロジック・インターフェースはそのコンバーターを呼び出して、**デコード**機能を要求します。**デコード**機能は、CICS アプリケーション・プログラム用の COMMAREA をセットアップします。
3. CICS ビジネス・ロジック・インターフェースは、呼び出し元が指定した CICS アプリケーション・プログラムを呼び出します。アプリケーション・プログラムに渡された COMMAREA は、デコード機能がセットアップしたものです。CICS ビジネス・ロジック・インターフェースの呼び出し元が、コンバーターが不要であることを示している場合、要求の先頭の 32 K バイトが COMMAREA で CICS アプリケーション・プログラムに渡されます。
4. CICS アプリケーション・プログラムは要求を処理し、出力を COMMAREA に戻します。
5. 呼び出し元がコンバーターを要求すると、CICS ビジネス・ロジック・インターフェースは、コンバーターの**エンコード**機能を呼び出します。エンコード機能は、COMMAREA を使用して応答を作成します。コンバーター・プログラムが呼び出されなかった場合は、 CICS ビジネス・ロジック・インターフェースは、CICS アプリケーション・プログラムが必要な応答を COMMAREA に入れたものと見なします。

6. CICS ビジネス・ロジック・インターフェースは、呼び出し元に応答を戻します。

CICS ビジネス・ロジック・インターフェースを使用した端末向けトランザクションの実行

図 23 は、端末向けトランザクションに対する要求での、CICS ビジネス・ロジック・インターフェースを経由した制御フローを示しています。ビジネス・ロジック・インターフェースは、Web CICS トランザクションのもとで実行されるのではなく、CICS ミラー・トランザクションのもとで実行される点に注意してください。この処理の最初の部分はプログラム呼び出しの場合と同じですが、トランザクションを実行する場合は、**wbbl_server_program_name** で、呼び出す CICS アプリケーション・プログラムとして DFHWBTTA を指定する必要があります。

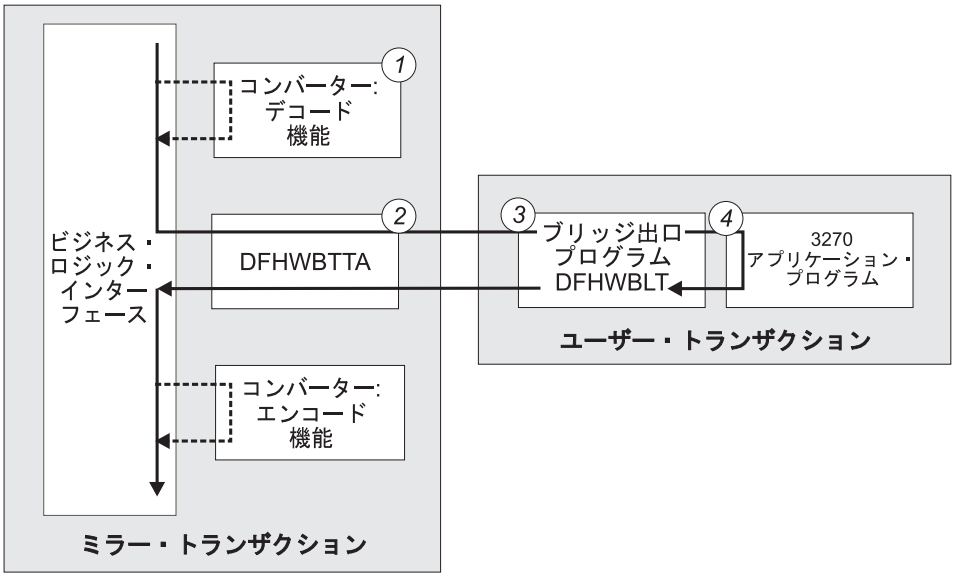


図 23. CICS ビジネス・ロジック・インターフェースによるトランザクションの実行制御フロー

1. 呼び出し元がコンバーターを要求すると、CICS ビジネス・ロジック・インターフェースはそのコンバーターを呼び出して、**デコード**機能を要求します。**デコード**機能は、DFHWBTTA 用の COMMAREA をセットアップします。
2. CICS ビジネス・ロジック・インターフェースは、DFHWBTTA を呼び出します。DFHWBTTA に渡された COMMAREA は、**デコード**機能がセットアップしたものです。コンバーター・プログラムが呼び出されなかった場合、COMMAREA には要求全体が含まれます。
3. DFHWBTTA は、端末向けトランザクションのトランザクション ID を HTTP 要求から抽出し、CICS Web ブリッジ出口ルーチンを実行するトランザクションを開始します。
4. プログラムがその基本機能への書き込みを試みると、データは CICS Web ブリッジ出口ルーチンにより代行受信されます。この出口は、CICS ビジネス・ロジック・インターフェースに戻す HTML 応答を作成します。呼び出し元がコンバーターを要求すると、CICS ビジネス・ロジック・インターフェースは、コンバーターの**エンコード**機能を呼び出します。エンコード機能は、COMMAREA を

使用して応答を作成します。コンバーター・プログラムが呼び出されなかった場合、CICS ビジネス・ロジック・インターフェースは、COMMAREA には必要な応答が含まれているものと見なします。

要求処理でのデータ・フロー

使用する機能と、その機能のカスタマイズ方法を決定するには、 CICS ビジネス・ロジック・インターフェースでどのようにデータが渡されるかを理解しておく必要があります。

コンバーター・プログラムおよび CICS ビジネス・ロジック・インターフェース

CICS ビジネス・ロジック・インターフェースの操作をサポートするために、CICS システム内に多数のコンバーター・プログラムを持つことができます。

CICS ビジネス・ロジック・インターフェース内のコンバーターの場所について詳しくは、 401 ページの図 22および 402 ページの図 23を参照してください。それぞれのコンバーターには次の 2 つの機能がなければなりません。

- **デコード**機能は、CICS アプリケーション・プログラムが呼び出される前に使用されます。この機能は、次の処理を行うことができます。
  - 着信要求からのデータを使用して、アプリケーション・プログラムが予期している形式で COMMAREA を作成する。
  - アプリケーション・プログラム用の COMMAREA の入出力データの長さを与える。
  - 要求に関連した管理用タスクを実行する。
- **エンコード**機能は、CICS アプリケーション・プログラムが呼び出された後に使用されます。この機能は、次の処理を行うことができます。
  - アプリケーション・プログラムからのデータを使用して応答を作成する。
  - 応答に関連した管理用タスクを実行する。

注:

- DECODE_DATA_PTR または ENCODE_DATA_PTR を変更して別のストレージ・ロケーションのアドレスを示すようにした場合は、元のストレージの主記憶域の解放はコンバーター・プログラムが行わなければなりません。
- ENCODE_DATA_PTR によって示されたアドレス (つまり、WBBL_OUTDATA_PTR フィールドに戻されたアドレスから 4 を引いたアドレス) のバッファーの解放は、CICS ビジネス・ロジック・インターフェースの呼び出し元が行う必要があります。
- コンバーターが異常終了した場合は、CICS は、DECODE_DATA_PTR および ENCODE_DATA_PTR でアドレス指定されたストレージを解放しようとします。したがって、すでに解放されたストレージのアドレスが、これらのポインターに含まれていないことを確認してください。

CICS ビジネス・ロジック・インターフェースを使用したプログラムの呼び出し

CICS ビジネス・ロジック・インターフェースを使用した、プログラムとリクエスター間のデータ・フロー。

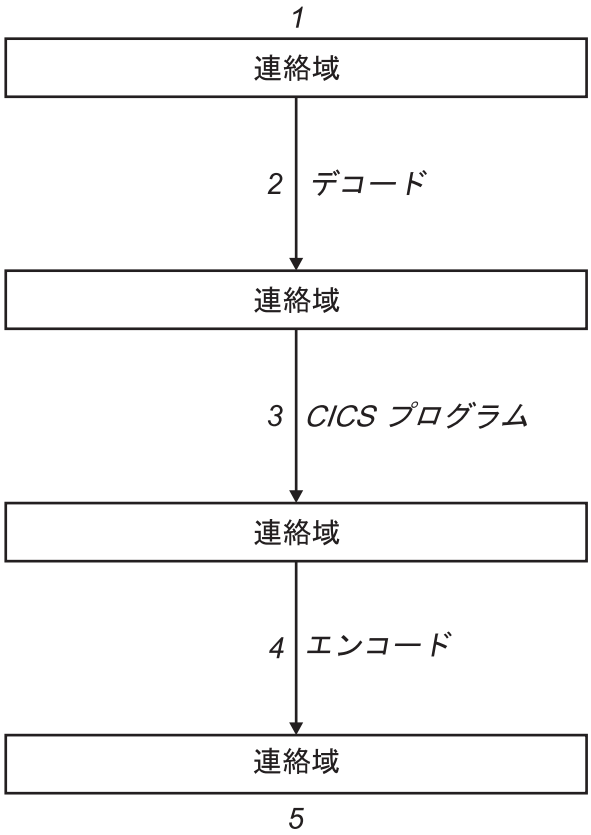


図 24. CICS ビジネス・ロジック・インターフェースによるプログラムの呼び出しデータ・フロー

1. CICS ビジネス・ロジック・インターフェースの呼び出し元は、処理対象の要求が含まれている COMMAREA を提供します。COMMAREA の内容は、後続の処理で受け入れ可能なコード・ページ形式である必要があります。通常、これはその内容が EBCDIC 形式でなければならないことを意味します。
2. 呼び出し元がコンバーターを要求すると、コンバーターの**デコード**機能が CICS アプリケーション・プログラム用の COMMAREA を作成します。
3. CICS アプリケーション・プログラムが COMMAREA を更新します。
4. 呼び出し元がコンバーターを要求した場合、コンバーターの**エンコード**機能は、呼び出し元に戻される COMMAREA を作成します。
5. CICS ビジネス・ロジック・インターフェースは、COMMAREA を呼び出し元に戻し、呼び出し元は COMMAREA の内容を使用できるようになります。

端末向けトランザクションの要求

図 25 に、端末向けトランザクションの開始要求におけるデータ・フローを示します。

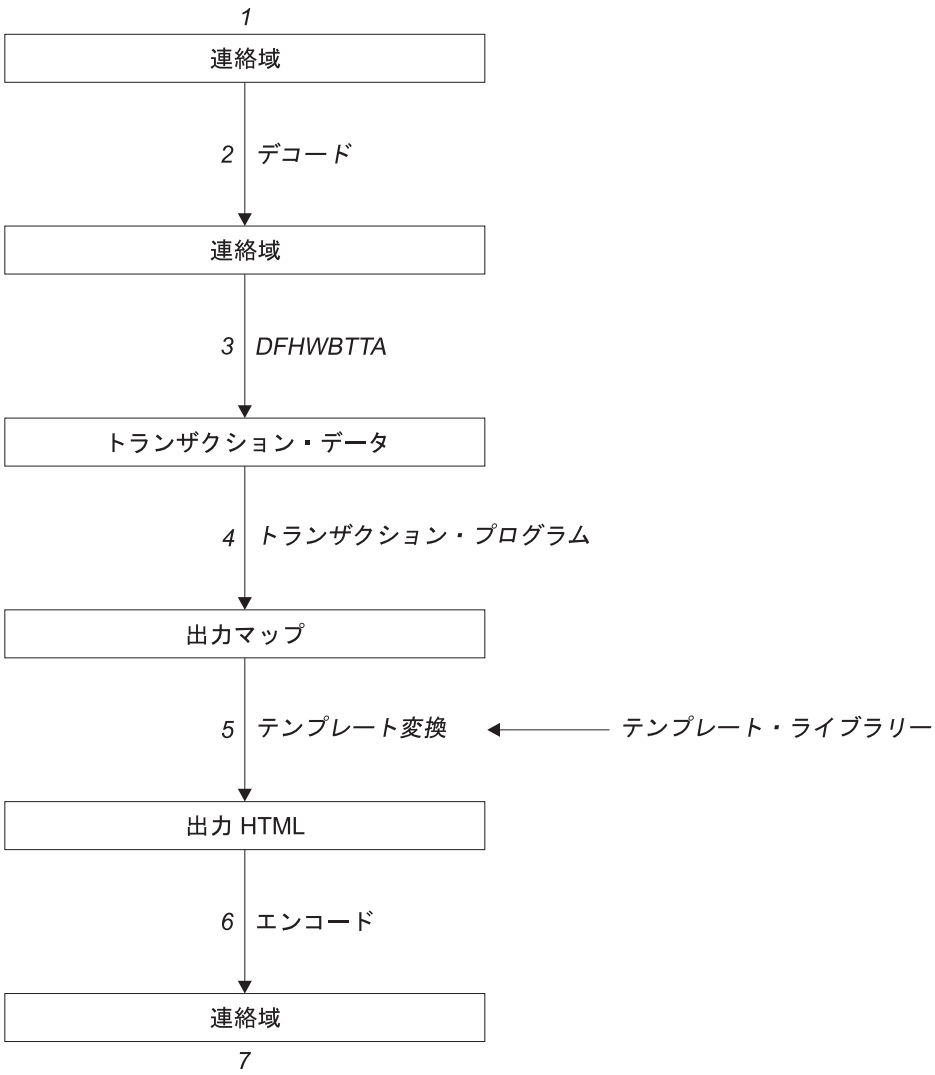


図 25. 端末向けトランザクションの開始データ・フロー

この図は、CICS ビジネス・ロジック・インターフェースを経由した、3270 BMS アプリケーションのデータ・フローを示しています。

1. CICS ビジネス・ロジック・インターフェースの呼び出し元は、処理対象の要求が含まれている COMMAREA を提供します。COMMAREA の内容は、後続の処理で受け入れ可能なコード・ページ形式である必要があり、DFHWBTTA では EBCDIC である必要があります。
2. 必要であれば、コンバーターの**デコード**機能を使用して要求を変更することができます。
3. これは、会話または疑似会話の最初のトランザクションなので、要求にはトランザクション ID が含まれており、おそらくトランザクション・プログラムで使用できるデータも含まれています。 DFHWBTTA はデータを抽出して、RECEIVE コマンドでそれをトランザクション・プログラムで使用できるようにします。

4. トランザクション・プログラムは RECEIVE コマンドを使用してデータを受け取ります。その後、トランザクション・プログラムは出力マップを作成し、 SEND MAP コマンドを用いてそれをリクエスターに送信します。
5. マップとそのデータ内容は HTML に変換されます。この変換では、DOCTEMPLATE 定義で定義したテンプレートが使用されます。
6. 必要であれば、コンバーターの**エンコード**機能を使用して応答を変更することができます。
7. CICS ビジネス・ロジック・インターフェースは、COMMAREA を呼び出し元に戻し、呼び出し元は COMMAREA の内容を使用できるようになります。

407 ページの図 26 に、端末向けトランザクションの継続要求におけるデータ・フローを示します。

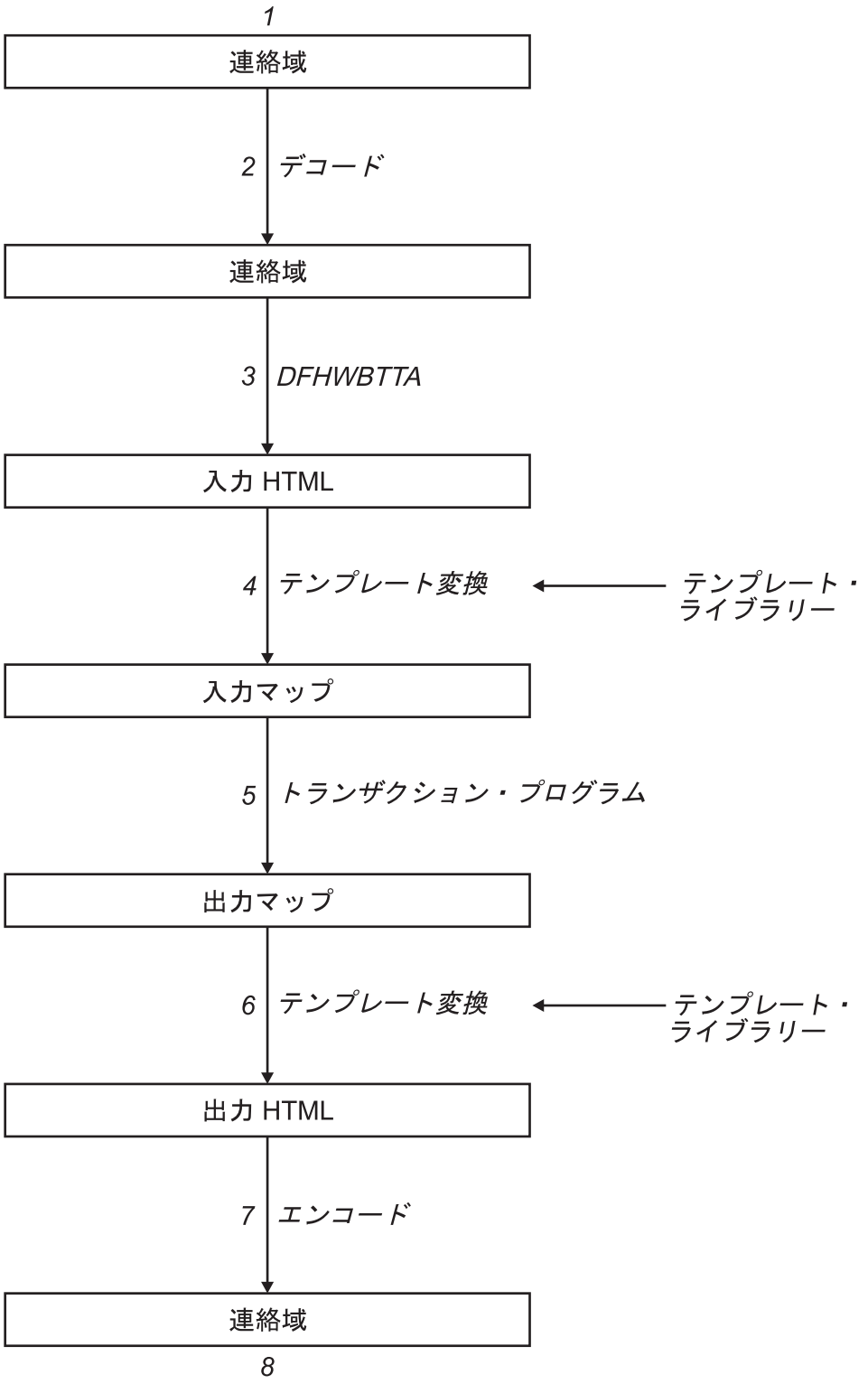


図 26. 端末向けトランザクションの継続データ・フロー

この図は、CICS ビジネス・ロジック・インターフェースが要求を処理する場合のデータ・フローを示しています。

1. CICS ビジネス・ロジック・インターフェースの呼び出し元は、処理対象の要求が含まれている COMMAREA を提供します。COMMAREA の内容は、後続の処理で受け入れ可能なコード・ページ形式である必要があります。通常、これはその内容が EBCDIC 形式でなければならないことを意味します。
2. コンバーターの**デコード**機能は、DFHWBTTA 用の COMMAREA を作成します。
3. これは、会話または疑似会話の最初のトランザクションではないので、要求には、トランザクション・プログラムが受け取ることを予期しているマップに対応する HTML が含まれています。 DFHWBTTA はデータを抽出して、RECEIVE MAP コマンドでそれをトランザクション・プログラムで使用できるようにします。
4. 着信形式の入力データは BMS マップに変換されます。この変換では、DOCTEMPLATE 定義のテンプレートが使用されます。
5. トランザクション・プログラムは RECEIVE MAP コマンドを使用してデータを受け取ります。その後、トランザクション・プログラムは出力マップを作成し、SEND MAP コマンドを用いてそれをリクエスターに送信します。
6. マップとそのデータ内容は HTML に変換されます。この変換では、DOCTEMPLATE 定義のテンプレートが使用されます。
7. コンバーターの**エンコード**機能は、変換プロセスからの HTML 出力を使用して、呼び出し元に戻される COMMAREA を作成します。
8. CICS ビジネス・ロジック・インターフェースは、COMMAREA を呼び出し元に戻し、呼び出し元は COMMAREA の内容を使用できるようになります。

オフセット・モードとポインター・モード

CICS ビジネス・ロジック・インターフェースは、以下の 2 つのモードで呼び出すことができます。

オフセット・モード

オフセット・モードの場合、DFHWBBLI の COMMAREA および CICS アプリケーション・プログラムの領域が入っている単一のストレージ域 (409 ページの図 27 のストレージ域 1) が用意されています。 DFHWBBLI の COMMAREA のフィールド *wbbl_indata_offset* には、ストレージ域の先頭からのアプリケーション・プログラム COMMAREA のオフセットが含まれています。ストレージ域の最大サイズは 32k バイトです。

オフセット・モードの場合、コンバーター・プログラムは DECODE_DATA_PTR または ENCODE_DATA_PTR の値を変更することはできません。

ポインター・モード

ポインター・モードの場合、2 つの独立ストレージ域が用意されています。1 つのストレージ域 (409 ページの図 27 のストレージ域 1) には DFHWBBLI の COMMAREA が含まれており、もう 1 つのストレージ域 (ストレージ域 2) には CICS アプリケーション・プログラムの領域が含まれています。 DFHWBBLI の COMMAREA の *wbbl_indata_ptr* フィールドには、アプリケーション・プログラムの COMMAREA のアドレスが含まれています。

ポインター・モードの場合、コンバーター・プログラムは DECODE_DATA_PTR または ENCODE_DATA_PTR の値を変更することができます。

これらの 2 つのモードが図 27 に示されています。

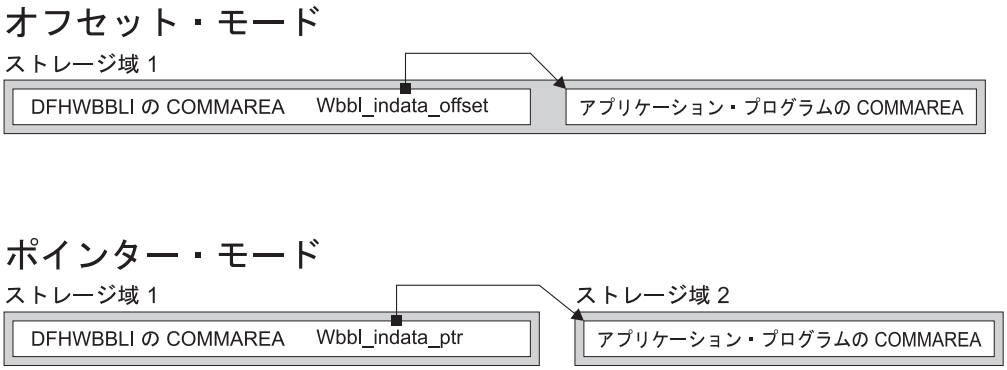


図 27. CICS ビジネス・ロジック・インターフェースにおけるオフセット・モードおよびポインター・モード

CICS ビジネス・ロジック・インターフェースを呼び出す場合は、以下のようにモードを指定する必要があります。

- wbbl_mode を "D" に設定して、オフセット・モードにすること、および (wbbl_user_data_offset によって参照される) HTTP 要求の本体が ASCII 形式であることを指定します。 これが必要となるのは、サーバー・プログラムでいずれかの FORMFIELD API コマンドを使用する場合です。
- wbbl_mode を "P" に設定してポインター・モードを示します

コンバーター・プログラムでは、`decode_volatile` または `encode_volatile` を調べて、以下のように、モードを判別することができます。

- 0 はオフセット・モードを示します
- 1 はポインター・モードを示します

CICS ビジネス・ロジック・インターフェースを呼び出す場合、以下のソースからの要求はすべてオフセット・モードを使用します。

- IBM HTTP Server を使用している Web クライアント
- ローカル・ゲートウェイ機能を使用している Java アプリケーション
- DCE RPC クライアント
- CICS Transaction Gateway を使用している Web クライアント

コード・ページ変換および CICS ビジネス・ロジック・インターフェース

CICS ビジネス・ロジック・インターフェースは、コード・ページ変換を行いません。ビジネス・アプリケーションに渡されるデータ、および戻されるデータは、アプリケーション・プログラミング・インターフェースが使用するコード・ページにあります。

ただし、**EXEC CICS WEB** アプリケーション・プログラミング・コマンドを使用して、クライアント・コード・ページを指定できるので、データは、アプリケーショ

ン・プログラミング・インターフェースそのものの中で変換されます。したがって、これらのコマンドを使用する場合は、アプリケーション・プログラムと CICS ビジネス・ロジック・インターフェースとの間でコード・ページ変換が行われます。インターフェースを通って渡されるデータは、**EXEC CICS WEB** コマンドの CHARACTERSET オプション (またはシノニム CLNTCODEPAGE) で指定されたコード・ページになります。

CICS ビジネス・ロジック・インターフェースの構成

このタスクについて

手順

1. 65 ページの『CICS Web サポートのシステム初期設定パラメーターの指定』 で説明されているように、WEBDELAY システム初期設定パラメーターを設定する必要があります。
2. プログラムの自動インストールを使用しない場合は、CICS ビジネス・ロジック・インターフェースの呼び出し元が使用する、ユーザーによる置換が可能なプログラム (コンバーター) をすべて定義することが必要です。 プログラムの自動インストールを使用する場合は、コンバーターを定義する必要はありません。すべてのコンバーターは、CICS ビジネス・ロジック・インターフェースが動作するシステムのローカル側になければなりません。

第 5 部 付録

付録 A. HTML コード化文字セット

このリファレンスでは、サポートされる IANA 登録済み文字セット名 (HTTP ヘッダーで charset= 値として指定される)、およびこれと同等の IBM CCSID をリストします。

これらの値はすべて、次のコマンドのコード・ページ変換オプションで有効です。

- WEB RECEIVE (クライアント)
- WEB RECEIVE (サーバー)
- WEB SEND (クライアント)
- WEB SEND (サーバー)
- WEB CONVERSE
- DOCUMENT RETRIEVE
- WEB READ FORMFIELD
- WEB STARTBROWSE FORMFIELD

表 20. コード化文字セット

言語	コード化文字セット	IANA charset	IBM CCSID
アルバニア語	ISO/IEC 8859-1	iso-8859-1	819
アラビア語	ISO/IEC 8859-6	iso-8859-6	1089
ブルガリア語	Windows 1251	windows-1251	1251
ベロルシア語	Windows 1251	windows-1251	1251
カタロニア語	ISO/IEC 8859-1	iso-8859-1	819
中国語 (簡体字)	GB	gb2312	1381 または 5477
中国語 (繁体字)	Big 5	big5	950
クロアチア語	ISO/IEC 8859-2	iso-8859-2	912
チェコ語	ISO/IEC 8859-2	iso-8859-2	912
デンマーク語	ISO/IEC 8859-1	iso-8859-1	819
オランダ語	ISO/IEC 8859-1	iso-8859-1	819
英語	ISO/IEC 8859-1	iso-8859-1	819
エストニア語	ISO/IEC 8859-1	iso-8859-1	819
フィンランド語	ISO/IEC 8859-1	iso-8859-1	819
フランス語	ISO/IEC 8859-1	iso-8859-1	819
ドイツ語	ISO/IEC 8859-1	iso-8859-1	819
ギリシャ語	ISO/IEC 8859-7	iso-8859-7	813
ヘブライ語	ISO/IEC 8859-8	iso-8859-8	916
ハンガリー語	ISO/IEC 8859-2	iso-8859-2	912
イタリア語	ISO/IEC 8859-1	iso-8859-1	819
日本語	Shift JIS EUC Japanese	x-sjis または shift-jis euc-jp	943 (943 のサブセットである 932 も可) 5050 (EUC)

表 20. コード化文字セット (続き)

言語	コード化文字セット	IANA charset	IBM CCSID
韓国語	EUC Korean	euc-kr	970 (AIX® または UNIX の場合)
ラトビア語	Windows 1257	windows-1257	1257
リトアニア語	Windows 1257	windows-1257	1257
マケドニア語	Windows 1257	windows-1257	1251
ノルウェー語	ISO/IEC 8859-1	iso-8859-1	819
ポーランド語	ISO/IEC 8859-2	iso-8859-2	912
ポルトガル語	ISO/IEC 8859-1	iso-8859-1	819
ルーマニア語	ISO/IEC 8859-2	iso-8859-2	912
ロシア語	Windows 1251	windows-1251	1251
セルビア語 (キリル文字)	Windows 1251	windows-1251	1251
セルビア語 (Latin-2)	Windows 1250	windows-1250	1250
スロバキア語	ISO/IEC 8859-2	iso-8859-2	912
スロベニア語	ISO/IEC 8859-2	iso-8859-2	912
スペイン語	ISO/IEC 8859-1	iso-8859-1	819
スペイン語	ISO/IEC 8859-15	iso-8859-15	923
スウェーデン語	ISO/IEC 8859-1	iso-8859-1	819
トルコ語	ISO/IEC 8859-9	iso-8859-9	920
ウクライナ語	Windows 1251	windows-1251	1251
Unicode	UCS-2	iso-10646-ucs-2	1200 (拡張) または 13488 (固定)
Unicode	UTF-16	utf-16	1200
Unicode	UTF-16 ビッグ・エンディアン	utf-16be	1201
Unicode	UTF-16 リトル・エンディアン	utf-16le	1202
Unicode	UTF-8	utf-8	1208

付録 B. CICS Web サポートにおける HTTP ヘッダーの解説

CICS Web サポートでは、アウトバウンド・メッセージの一部の HTTP ヘッダーが CICS によって自動的に提供されます。ユーザーが独自のヘッダーを追加することもできます。メッセージが CICS に送信されると、CICS では一部の HTTP ヘッダーに対応してアクションが実行され、ユーザー・アプリケーション・プログラムではその他のヘッダーに対応してアクションが実行されます。この参照情報では、CICS Web サポートによる HTTP ヘッダーの処理方法について説明します。

標準の HTTP ヘッダーは HTTP/1.1 仕様書 (RFC 2616) および HTTP/1.0 仕様書 (RFC 1945) で説明されています。HTTP プロトコル仕様ではない拡張ヘッダーを含め、多くの HTTP ヘッダーが考えられます。より完全なリストについては、使用している HTTP 仕様を参照してください。HTTP 仕様の詳細については、 13 ページの『HTTP プロトコル』を参照してください。

このトピックでは、CICS Web サポートにおける HTTP ヘッダーの一般的使用方法、および CICS Web サポートによる特定のヘッダーに対するアクションについて説明します。 ヘッダー値の正しい形式や各ヘッダーを使用するときのコンテキストなど、HTTP ヘッダーの使用方法に関する詳細な手引きおよび要件については、ご使用の HTTP の仕様書を参照してください。

CICS が受信したメッセージの HTTP ヘッダー

- CICS が HTTP 要求または応答を受信すると、いくつかの HTTP ヘッダーを使用して、CICS Web サポートが実行するアクションが決まります。 417 ページの表 21には、HTTP 要求のさまざまなヘッダーに対して CICS の取るアクションが示されています。 419 ページの表 22 には、HTTP 応答のさまざまなヘッダーに対して CICS の取るアクションについて示されています。その他のヘッダーは、CICS では使用されず、これらのヘッダーに応じて実行される適切なアクションは、ユーザー・アプリケーションによって異なります。
- メッセージに対して受信されたすべてのヘッダーは、CICS によって使用されたかどうかに関係なく、WEB READ HTTPHEADER コマンドおよび HTTP ヘッダー・ブラウズ・コマンドを使用する検査用に、ユーザー・アプリケーションで使用できるようになります。 CICS は、メッセージに特定のヘッダーがあることをユーザー・アプリケーションに警告することはありません。アプリケーションで必要としないか、または認識されないヘッダーは、無視してください。
- CICS では、サーバーまたはクライアントがメッセージの受信時に実行する必要があるアクションに関連する HTTP/1.1 仕様の MUST レベル要件にすでに対応しています。このため、ヘッダーを調べなくても要求または応答を受信して使用することができます。ただし、Web クライアントまたはサーバーとの将来の通信で実行するアクションに関連する情報用に、ヘッダーを検査する必要があります。
- HTTP ヘッダーはヘッダー名とヘッダー値で構成されており、これらはコロンで区切られます。 HTTP/1.1 仕様では、コロンとヘッダー値の間をシングル・スペースで区切り、この共通形式に従うことが推奨されています。HTTP/1.0 仕様では、このシングル・スペースが必須ですが、HTTP/1.1 仕様では、アプリケーショ

ンによって、より多くのスペースを使用したり、またはスペースを使用しないことが許可されています。後方互換性を保持するため、CICS では、メッセージ処理中に CICS でアクションを実行する一部のヘッダー (Content-Length ヘッダーなど) で、共通形式のシングル・スペースが必要です。 HTTP 要求を CICS に送信するアプリケーションを設計する場合は、HTTP/1.1 仕様で推奨されているように、すべての HTTP ヘッダーにおいて、この共通形式に従うようにしてください。 EXEC CICS WEB WRITE HTTPHEADER コマンドを実行すると、ヘッダーが適切な形式で作成されるので、CICS によって自動的に書き込まれるヘッダーは、すべて適切な形式になっています。

CICS から送信されるメッセージの HTTP ヘッダー

- HTTP/1.1 のバージョンを使用して CICS から送信される HTTP 要求または HTTP 応答において、CICS は、自動的に主要なヘッダーを提供します。これらのヘッダーは通常、HTTP/1.1 仕様に準拠する基本メッセージ用に書き込まれます。 HTTP/1.0 のバージョンを使用した HTTP 応答の場合、CICS は自動的に少数のヘッダーを提供します。 これらのヘッダーには、すべてのメッセージについて CICS によって生成されるものや、ユーザー・アプリケーション・プログラム内の WEB SEND コマンドで指定されているオプションにより作成されるものがあります。 420 ページの表 23、および 421 ページの表 24 は、HTTP バージョンごとに書き込まれるヘッダー、およびそのヘッダーのソースをリストしています。

  ユーザー・アプリケーション・プログラムによって書き込まれるヘッダーが、CICS の生成対象にもなっている場合、CICS はこのヘッダーを状況に応じて処理します。

  – CICS が HTTP サーバーとして機能している場合、CICS は応答に適したヘッダーを上書きしないで、アプリケーションのバージョンを使用できるようにします。
  – CICS が HTTP クライアントとして機能している場合、CICS は要求に適したヘッダーの書き込みをアプリケーションに許可しないで、WEB WRITE HTTPHEADER コマンドにエラー応答を戻します。例外は TE ヘッダーと Content-Type ヘッダーです。アプリケーション・プログラムは、TE ヘッダーのインスタンスをさらに追加することができます。必要なヘッダーにスペースまたは 56 を超える文字を含める必要があり、WEB SEND コマンドの MEDIATYPE オプションで指定できない場合、アプリケーション・プログラムは Content-Type ヘッダーを提供することもできます。
  – ヘッダーがメッセージのタイプ (要求または応答) として一般的に不適切な場合、ユーザー定義のすべてのヘッダーの場合と同様、CICS はそれを許可します。メッセージが、使用している HTTP 仕様に準拠している場合は、このような状態が発生することはありません。
- ユーザー・アプリケーション・プログラムは、**WEB WRITE HTTPHEADER** コマンドを使用して、要求または応答にさらに HTTP ヘッダーを追加することができます。CICS では、追加の HTTP ヘッダーが許容され、渡されます。 HTTP サーバーとしての CICS の場合、CICS 文書テンプレートまたは HFS ファイルで静的応答を返している場合は、CICS によって自動的に提供されるヘッダーよりも多くのヘッダーを応答に追加することはできません。

- CICS は、ユーザー作成ヘッダーの名前または値をチェックしません。アプリケーション・プログラムが、正確でかつ正しく形式設定された情報を、ご使用の HTTP の仕様を満たすように提供していることを確認する必要があります。アプリケーションが複雑なアクションを実行する場合は、該当する要件の HTTP 仕様を特に注意深く確認してください。これらのアクションについて説明するため、特定のヘッダーを提供するための重要な (MUST または SHOULD レベル) 要件がある可能性があります。例えば、次のようなアクションを実行する場合は、特殊な HTTP ヘッダーが必要です。
  - 文書の変更日付またはエンティティー・タグを使用する条件付き要求に応答する、またはそのような要求を作成する場合。
  - クライアントの機能、または Web クライアントの各国語要件に従って、応答の内容を変更する場合。
  - 文書全体ではなく文書のある範囲を含む応答を提供する、または要求を行う場合。
  - 応答に対するキャッシュ制御情報を提供する場合。

  応答に特定の状況コードを使用する場合は、特定の HTTP ヘッダーも必要になることがあります。例えば、状況コード 405 (Method not allowed (メソッドが許可されていません)) を使用する場合は、Allow ヘッダーを使用して、許可されているメソッドを示します。状況コードの使用方法の詳細については、 423 ページの『付録 C. CICS Web サポートの HTTP 状況コード・リファレンス』を参照してください。

Upgrade ヘッダー

- CICS Web サポートではプロトコルのアップグレードがサポートされていないという特別なことを知っておいてください。これは、次のことを意味します。
  - HTTP サーバーとしての CICS の場合、アプリケーションでは、Web クライアントによって送信された Upgrade ヘッダーに応じてアクションを実行することはできません。
  - HTTP クライアントとしての CICS の場合、Upgrade ヘッダーを要求に書き込むことはできません。

  CICS は、接続中の HTTP バージョンにおけるスイッチをサポートしておらず、セキュリティー層でのアップグレードはサポートされていません。

HTTP サーバーとしての CICS: HTTP 要求の受信時に CICS でアクションが実行されるヘッダー

表 21 に、Web クライアントから受信した要求に含まれるいくつかのヘッダーに対して CICS が実行するアクションを示します。

表 21. HTTP サーバーとしての CICS: HTTP 要求のヘッダーに対する CICS のアクション

Web クライアントから受信されるヘッダー	応答がユーザー・アプリケーション・プログラムによって処理される場合に CICS で実行されるアクション	応答が静的文書で提供される場合に CICS が実行するアクション
Authorization	提供されたユーザー ID とパスワードを、検証のために RACF に渡し、これらが無効の場合は要求を拒否します。	アプリケーション生成応答の場合と同じ。

表 21. *HTTP サーバーとしての CICS: HTTP 要求のヘッダーに対する CICS のアクション (続き)*

Web クライアントから受信されるヘッダー	**応答がユーザー・アプリケーション・プログラムによって処理される場合に CICS で実行されるアクション**	**応答が静的文書で提供される場合に CICS が実行するアクション**
Connection	応答の送信後に、Web クライアントからの接続クローズ要求を実行します。	アプリケーション生成応答の場合と同じ。
Content-Length	CICS では、メッセージ・ボディを持つすべてのインバウンド HTTP/1.1 メッセージに、Content-Length ヘッダーが必要です。メッセージ・ボディが存在するが、ヘッダーが提供されていない場合、またはその値が不正確である場合、欠陥のあるメッセージ、または以降のメッセージに対して受信されるソケットによって、予期しない結果が生じることがあります。メッセージ・ボディを持つ HTTP/1.0 メッセージの場合、Content-Length ヘッダーはオプションです。	メッセージ・ボディは静的応答に対する処理では使用されませんが、その場合でもソケットから受信する必要があるため、アプリケーション生成応答の場合と同じ要件が適用されます。
Content-Type	ヘッダーを構文解析して、コード・ページ変換のために、メディア・タイプおよび文字セットを識別します。	ヘッダーを構文解析して、応答のコード・ページ変換のための文字セットを識別します。
Expect	100-Continue 応答を Web クライアントに送信し、要求の残りの部分を待ちます。	アプリケーション生成応答の場合と同じ。
Host	このヘッダーが存在せず、クライアントが HTTP/1.1 である場合、 400 (「Bad Request (不正な要求)」) 応答を Web クライアントに送信します。	アプリケーション生成応答の場合と同じ。
If-Modified-Since	CICS によるアクションはありません。ユーザー・アプリケーションは、このヘッダーの存在を検査して、適宜応答するか、あるいはヘッダーを無視して、アプリケーション生成の応答が変更されたと見なします。	文書テンプレート: 応答が変更されていると見なし、要求された項目を送信します。 HFS ファイル: 変更日をチェックし、その結果に従って応答します。項目が変更されていない場合は 304 応答を送信します。
If-Unmodified-Since	ヘッダーが存在する場合は、常に 412 (「Precondition Failed (前提条件が失敗しました)」) 応答を Web クライアントに送信して、指定された時間以降に応答が変更されたことを示します (したがって、ユーザー・アプリケーションはこのヘッダーが存在しているかどうかをチェックする必要はありません)。	文書テンプレート: アプリケーション生成の応答に関して、応答が変更されていると見なし、412 応答を送信します。 HFS ファイル: 変更日をチェックし、その結果に従って応答します。
Trailer	WEB READ HTTPHEADER コマンドを使用して、アプリケーションで個々の末尾ヘッダーを使用できるようにします。	チャンク化されたメッセージは、静的応答に適していません。

表 21. *HTTP サーバーとしての CICS: HTTP 要求のヘッダーに対する CICS のアクション (続き)*

Web クライアントから受信されるヘッダー	**応答がユーザー・アプリケーション・プログラムによって処理される場合に CICS で実行されるアクション**	**応答が静的文書で提供される場合に CICS が実行するアクション**
Transfer-Encoding	「チャンク化」されている場合は、すべてのチャンクを受信し、それらを 1 つのメッセージにアセンブルして、アプリケーションに渡します。「チャンク化」以外の場合、501 (「Not Implemented (インプリメントなし)」) 応答を Web クライアントに送信します。 Transfer-Encoding ヘッダーは、メッセージ上に残りますが、通知のみが目的です。	チャンク化されたメッセージは、静的応答に適していません。
Warning	TS キュー CWBW に警告テキストを書き込みます。128 文字を超える場合、警告テキストは切り捨てられます。	アプリケーション生成応答の場合と同じ。

HTTP クライアントとしての CICS: HTTP 応答の受信時に CICS でアクションが実行されるヘッダー

表 22 は、サーバーから受信した応答のいくつかのヘッダーに対して CICS が実行するアクションを示しています。

表 22. *HTTP クライアントとしての CICS: HTTP 応答のヘッダーに対する CICS のアクション*

サーバーから受信したヘッダー	**CICS が実行するアクション**
Connection	応答の受信後に、サーバーからの接続クローズ要求を実行します。
Content-Length	CICS では、メッセージ・ボディを持つすべてのインバウンド HTTP/1.1 メッセージに、Content-Length ヘッダーが必要です。メッセージ・ボディが存在するが、ヘッダーが提供されていない場合、またはその値が不正確である場合、欠陥のあるメッセージ、または以降のメッセージに対して受信されるソケットによって、予期しない結果が生じることがあります。メッセージ・ボディを持つ HTTP/1.0 メッセージの場合、Content-Length ヘッダーはオプションです。
Content-Type	ヘッダーを構文解析して、コード・ページ変換のために、メディア・タイプおよび文字セットを識別します。
Trailer	WEB READ HTTPHEADER コマンドを使用して、アプリケーションで末尾ヘッダーを使用できるようにします。
Transfer-Encoding	「チャンク化」されている場合は、すべてのチャンクを受信し、それらを 1 つのメッセージにアセンブルして、アプリケーションに渡します。「チャンク化」以外の場合、501 (「Not Implemented (インプリメントなし)」) 応答を Web クライアントに送信します。 Transfer-Encoding ヘッダーは、メッセージ上に残りますが、通知のみが目的です。
Warning	TS キュー CWBW に警告テキストを書き込みます。128 文字を超える場合、警告テキストは切り捨てられます。

HTTP サーバーとしての CICS: CICS が HTTP 応答に書き込むヘッダー

表 23 は、Web クライアントからの要求に対して応答するときに CICS が書き込むヘッダー、それらのヘッダーを使用するときの HTTP バージョン、および CICS がヘッダーで提供する情報のソースを示しています。

表 23. HTTP サーバーとしての CICS: HTTP 応答に対する CICS 書き込みのヘッダー

CICS によって書き込まれるヘッダー	HTTP バージョン	ユーザー・アプリケーション・プログラムによって応答が処理されるソース	静的文書によって応答が提供されるソース
Connection	1.0 および 1.1	WEB SEND コマンドの CLOSESTATUS オプション。クローズが指定されておらず、クライアントが HTTP/1.0 の場合は、Keep-Alive が送信されます。クローズが指定されている場合は Connection: close が送信されるか、または HTTP/1.0 クライアントの場合は Keep-Alive が省略されます。TCPIPSERVICE 定義の MAXPERSIST 属性の制限で指定したポートに対して接続スロットリングが実施されている場合も、CICS は Connection: close を送信します。	Keep-Alive が静的な応答で送信されます。
Content-Length (チャンク転送コーディングが使用されない場合)	1.0 および 1.1	応答ボディがデータのバッファーである場合、長さは、WEB SEND コマンドの FROMLENGTH オプションから取得されます応答ボディが CICS 文書である場合、長さは CICS によって計算されます。	CICS によって計算されます。
Content-Type	1.0 および 1.1	WEB SEND コマンドの MEDIATYPE オプションおよび応答ボディの文字セット (ヘッダーは、MEDIATYPE オプションが指定されている場合にのみ作成されます)。	要求に対する URIMAP リソース定義の MEDIATYPE 属性であり、応答ボディに対する文字セット。
Date	1.0 および 1.1	CICS によって生成される現在日時。	CICS によって生成される現在日時。
Last-Modified (静的 HFS ファイルの場合のみ)	1.0 および 1.1	**動的応答には提供されません。** 実行可能な場合、アプリケーションがこれを生成することになります。	HFS ファイルの場合: ファイルの変更日付。文書テンプレートの場合: 提供されません。
サーバー	1.0 および 1.1	「IBM_CICS_ Transaction_Server/ 4.2.0 (zOS)」に事前設定されます。	「IBM_CICS_ Transaction_Server/ 4.2.0 (zOS)」に事前設定されます。
Transfer-Encoding	1.1 のみ	WEB SEND コマンドの CHUNKING オプション。	使用されません。
WWW 認証	1.0 および 1.1	TCPIPSERVICE リソース定義の AUTHENTICATE 属性。	TCPIPSERVICE リソース定義の AUTHENTICATE 属性。

HTTP クライアントとしての CICS: HTTP 要求に対して CICS が書き込むヘッダー

表 24 は、アプリケーション・プログラムがサーバーにクライアント要求を送信したときに CICS が書き込むヘッダー、ヘッダーが使用されるときの HTTP のバージョン、および CICS がこのヘッダーで提供する情報のソースを示しています。

表 24. HTTP クライアントとしての CICS: HTTP 要求に対する CICS 書き込みのヘッダー

CICS によって書き込まれるヘッダー	**HTTP バージョン**	**ソース**
Connection	1.0 および 1.1	WEB SEND コマンドの CLOSESTATUS オプション。ヘッダーの値は、サーバーの HTTP バージョンに従って選択されます。
Content-Length (チャンク転送コーディングが使用されない場合)	1.0 および 1.1	WEB SEND コマンドの FROMLENGTH オプション
Content-Type	1.0 および 1.1	WEB SEND コマンドの MEDIATYPE オプションおよび応答ボディの文字セット (ヘッダーは、MEDIATYPE オプションが指定されている場合にのみ作成されます)。必要なヘッダーにスペースまたは 56 を超える文字を含める必要があり、MEDIATYPE オプションで指定できない場合、アプリケーション・プログラムは CICS の代わりに Content-Type ヘッダーを提供できます。
Date	1.0 および 1.1	CICS が生成する、GMT 時刻を使用する RFC 1123 形式の現行日付および時刻。
Expect	1.1 のみ	WEB SEND コマンドの ACTION(EXPECT) オプション。このオプションは、要求にメッセージ・ボディが含まれている場合にのみ使用する必要があります。CICS は HTTP/1.0 サーバーにはこのヘッダーを送信しません。CICS がまだサーバーのバージョンを認識していない場合、OPTIONS メソッドで ACTION(EXPECT) オプションを指定すると、追加の要求が起動されます。
Host	1.0 および 1.1	WEB OPEN コマンドの HOST オプション。
TE	1.1 のみ	チャンク化済みメッセージおよびトレーラーが受け入れられたことを示すために、HTTP/1.1 サーバーに送信されるときに常に CICS によって追加されます (チャンク化されたメッセージは、HTTP/1.0 サーバーによって送信されません)。アプリケーション・プログラムは、さらに TE ヘッダーを追加することができます。
Transfer-Encoding	1.1 のみ	チャンク化されたメッセージを送信するための、シーケンス内の最初の WEB SEND コマンド (コマンドの CHUNKING オプションは、チャンク転送コーディングを示します)。Transfer-Encoding ヘッダーは、メッセージの最初のチャンクにのみ書き込まれます。
User-Agent	1.0 および 1.1	「IBM_CICS_ Transaction_Server/ 4.2.0 (zOS)」に事前設定されます。

付録 C. CICS Web サポートの HTTP 状況コード・リファレンス

HTTP 状況コードは、クライアントの要求の結果を説明するために、サーバーによってクライアントに提供されます。 CICS が HTTP サーバーである場合は、状況に応じて、CICS Web サポートまたはユーザー・アプリケーション・プログラムのいずれかが、各応答の適切な状況コードを選択します。 CICS が HTTP クライアントである場合は、サーバーから受信したほとんどの状況コードは、ユーザー・アプリケーション・プログラムに渡されて処理されます。

17 ページの『状況コードおよび理由句』では、状況コードが HTTP 応答でどのように使用されるかについて説明しています。

状況コードの意味および正しい使用方法についての詳細は、ご使用の HTTP の仕様書を参照してください。HTTP 仕様の詳細については、13 ページの『HTTP プロトコル』を参照してください。

このトピックでは、CICS Web サポートに関連する HTTP/1.1 状況コードの簡単な要約を示します。送信する状況コードを Web エラー・プログラムを経由して選択する場合も、ユーザー・アプリケーションから直接選択する場合も、使用する HTTP 仕様を確認することが重要です。 HTTP 仕様には、状況コードの使用方法についての詳しい指針と要件が示されています (応答ボディの内容、含めるべき HTTP ヘッダーなど)。

CICS から送信される応答の状況コード (CICS が HTTP サーバーである場合)

- CICS Web サポートは、以下の場合に Web クライアントへの応答を生成します。
  - CICS Web サポートが Web クライアントからの要求の初期処理において問題を検出した場合 (例えば、必要な情報が要求に欠落している場合や、要求の送信に時間がかかりすぎて受信タイムアウトになった場合)。
  - インストールされている URIMAP 定義が要求に一致するが、URIMAP 定義または仮想ホストが使用不可である場合、または静的応答のリソースが読み取れない場合。
  - 一致する URIMAP 定義が要求について ATOMSERVICE リソース定義を参照するが、 ATOMSERVICE 定義が無効になっているか、Atom フィード用の CICS リソースを読み取れない場合。
  - URIMAP マッチングが失敗し、TCPIPSERVICE 定義で指定されたアナライザーが要求を処理できず、制御を Web エラー・プログラムに渡す場合。
  - URIMAP 定義、アナライザー・プログラム処理およびコンバーター・プログラム処理のいずれによっても、要求を処理するために実行するアプリケーション・プログラムを判別できない場合。
  - アナライザー・プログラム、コンバーター・プログラム、またはユーザー作成のアプリケーション・プログラムで異常終了が発生した場合。これにより、処理が失敗しても応答が Web クライアントに確実に返されます。

– URIMAP でリダイレクト応答を指定している場合。
– Web クライアントが、応答を提供するために必要なリソースへのアクセスを許可されていない場合。

このような状態のとき、CICS は適切な状況コードを選択し、デフォルトの応答を作成します。 425 ページの表 25では、そのような目的のために CICS で使用される状況コードについて説明しています。CICS は、ユーザー作成アプリケーション・プログラムが処理を正常に終了したうえでエラーを示す応答を返そうとする状況では、応答を生成**しません** (例えば、クライアントがリソースに対してサポートされていないメソッドを指定した場合)。この場合はユーザー作成アプリケーションが応答を作成します。

- 4xx および 5xx という状況コードを持つ、CICS で生成されるほとんどの応答について、Web クライアントに送信された応答は、ユーザーが置換できる Web エラー・プログラムである DFHWBEP と DFHWBERX を調整することによって変更できます。 CICS が生成した、1xx、 2xx、および 3xx を含む応答は変更できません。Web エラー・プログラムでは、応答の状況コード、理由句、HTTP ヘッダーおよびメッセージ・ボディを変更できます。 Web エラー・プログラムを変更した場合は、状況コードと応答の内容の選択が、作業している HTTP 仕様の要件に従って行われたことを確認してください。 131 ページの『第 9 章 Web エラー・プログラム』では、Web エラー・プログラムを調整する方法を説明しています。
- クライアントの要求に応答するユーザー・アプリケーション・プログラムは、応答の適切な状況コードを選択する必要があります。状況コードによって Web クライアントに以下のメッセージを伝えることができます。
  – 要求は予期したとおりに完了しました。
  – エラーがあるために要求を完了できません。
  – 要求を正常に終了するためにクライアントは他の何かを行う必要があります。これには、リダイレクト URL に従うことや、サーバーで受け入れられるように要求を変更することが含まれます。

  状況コードは応答の他の内容、つまりメッセージ・ボディと HTTP ヘッダーに影響を与えます。 100 ページの『HTTP サーバーとしての CICS からの HTTP 応答の送信』では、状況コードと理由句を含む応答のアセンブルと送信の方法を説明しています。

CICS が受信する応答の状況コード (CICS が HTTP クライアントである場合)

- CICS が HTTP クライアントである場合、CICS Web サポートは、ほとんどの状況コードで応答を直接ユーザー・アプリケーション・プログラムに渡して処理します。状況コードのなかには CICS によって処理され、アプリケーションに返されないものがありますが、これらは少数です。状況コードがアプリケーションに渡された場合、それは、CICS がコードに対する応答としてどのようなアクションもとっていないことを示します。
- エラーを示す状況コードを持つエラーを受信した場合に適切に動作するように、ユーザー・アプリケーションを設計する必要があります。特に、以下の状況では常に状況コードをチェックする必要があります。
  – 今または今後の接続でサーバーに対して同じ要求を行う場合。
  – この接続を使用してサーバーに対してさらに要求を行う場合。

– ご使用のアプリケーションにおいて、応答で受信した情報に応じて、引き続き処理を実行する場合。

適切なアクションに関する指針については、作業に使用している HTTP 仕様を確認してください。 HTTP/1.1 仕様には、状況コード受信時にアプリケーションにさらにアクションを要求する MUST (必須) レベルの要件はありませんが、リダイレクトに従うことを求める要件のような SHOULD (推奨) 要件がいくつかあります。

HTTP サーバーとしての CICS: CICS から Web クライアントに提供される状況コード

表 25 は、CICS が Web クライアントの要求に応答を提供する場合の状況コードを示しています。これらの応答の一部は、Web エラー・プログラムを変更することによって調整できます。ここに示されている状況コードの多くはユーザー・アプリケーション・プログラムで使用される場合もあります。

HTTP/1.1 クライアントにのみ適切な状況コードもあります。 CICS は、HTTP/1.0 クライアントにはそれらの状況コードを返しません。

表 25. HTTP サーバーとしての CICS: CICS によって生成されて Web クライアントに送信される応答の状況コード

提供される状況コードと理由句	HTTP/1.0 クライアントに送信されるか?	この応答が提供される状況	Web エラー・プログラムで変更できるか?
100 Continue (続行)	いいえ	Web クライアントが Expect ヘッダーを送信しました。	いいえ
200 OK	はい	通常応答の配信。	いいえ
201 Created (作成済み)	はい	新規オブジェクトが作成されました。	いいえ
301 Moved Permanently (永続的に移動)	はい	URIMAP 定義が属性 REDIRECTTYPE (PERMANENT) でリダイレクトを指定。	いいえ
302 Found (検出)	はい	URIMAP 定義が属性 REDIRECTTYPE (TEMPORARY) でリダイレクトを指定。	いいえ
304 Not Modified (未変更)	はい	If-Modified-Since ヘッダーが要求で使用されており、静的応答が変更されていないことを CICS が検証できる場合。	はい
400 Bad Request (不正な要求) (状況によっては Invalid Request (無効な要求))	はい	要求の構文エラー (要求の行が正しく指定されていない、要求が不完全、Atom POST または PUT 要求での Atom エントリーの問題など)。または、Host ヘッダーが指定されていない (HTTP/1.1 のみ)。または、 If-Match ヘッダーのない PUT 要求を受け取りました。現在のエンティティー・タグを認識せずにオブジェクトを更新するクライアントは、 If-Match: * を指定する必要があります。	はい
401 Basic Authentication Error (基本認証エラー)	はい	基本認証にユーザー ID とパスワードが必要。これはポートの TCPIPSERVICE 定義のセキュリティー設定によって決定されます。	はい

表 25. HTTP サーバーとしての CICS: CICS によって生成されて Web クライアントに送信される応答の状況コード (続き)

提供される状況コードと理由句	HTTP/1.0 クライアントに送信されるか?	この応答が提供される状況	Web エラー・プログラムで変更できるか?
403 Forbidden (禁止) (一部の状況では Client Authentication Error (クライアント認証エラー))	はい	基本認証が失敗しました。または、クライアント証明書に問題があります。あるいは、応答を提供するために必要なリソース (ATOMSERVICE リソース定義、別名トランザクション、プログラムで使用される CICS コマンド、または応答データが含まれる CICS リソースなど) へのアクセスをユーザーが許可されていません。	はい
404 Not Found (未検出) (一部の状況では Program Not Found (プログラムが見つかりません)、File Not Found (ファイルが見つかりません))	はい	要求に応答するように指定されたプログラムが見つかりません。または、応答を提供するために必要なリソースが見つかりません。 または、Atom フィードのデータを提供するために使用される CICS リソース内にレコードが見つかりません。あるいは、イメージ・ファイルが見つかりません。	はい
408 Request Timeout (要求タイムアウト)	いいえ	要求が受信タイムアウトになりました。これはポートの TCPIPSERVICE 定義の SOCKETCLOSE 属性によって決定されます。	はい
409 Conflict (競合) (状況によっては Duplicate resource (重複リソース))	はい	指定した URL の既存オブジェクトがすでに存在するため、新規オブジェクトは作成されませんでした。	いいえ
412 Precondition Failed (前提条件の失敗)	はい	If-Unmodified-Since ヘッダーが要求で使用されていました。または、If-Match ヘッダーのエンティティー・タグ値が、更新中のオブジェクトのエンティティー・タグと一致しません。	はい
417 Expectation Failed (予期の失敗)	いいえ	値「100-continue」を持たない Expect ヘッダーを受け取りました。	いいえ
500 Internal Server Error (内部サーバー・エラー) (状況によっては Resource Error (リソース・エラー))	はい	要求の処理と応答の提供にかかわるいずれかのプログラムでの異常終了。または、z/OS UNIX ファイルで静的応答を読み取り中のエラー。または、Atom エントリー・コンテンツとして使用するリソース・レコードから XML マークアップを生成するエラーなどの、Atom フィードのリソースに関わるエラー。	はい
501 Method Not Implemented (メソッドのインプリメントなし)	はい	この HTTP バージョンのメソッドは CICS でサポートされていません (サポートはされていても、クライアントが要求した使用方法ではサポートされていない、特定のリソースを参照する OPTIONS 要求など)。または、要求のメディア・タイプが、サポートされていない「multipart/byteranges」です。または、要求の転送コーディングが「チャンク」でありません (注: 接続は CICS によってクローズされます)。	はい

表 25. HTTP サーバーとしての CICS: CICS によって生成されて Web クライアントに送信される応答の状況コード (続き)

提供される状況コードと理由句	HTTP/1.0 クライアントに送信されるか?	この応答が提供される状況	Web エラー・プログラムで変更できるか?
503 Service Unavailable (サービス利用不可)	はい	一致する URIMAP 定義が存在していても無効になっているか、URIMAP 定義が属している仮想ホストが無効になっています。または、マッチングしている URIMAP 定義が、無効になっている ATOMSERVICE リソース定義を参照しています。あるいは、応答データを提供するために URIMAP 定義または ATOMSERVICE 定義で指定されているリソースが無効になっています。	はい
505 Version Not Supported (サポートされていないバージョン)	いいえ	HTTP のバージョンが 1.1 より上であり、メソッドは CICS でサポートされている最上位のバージョンで認識されません。	はい

HTTP サーバーとしての CICS: ユーザー・アプリケーションでの状況コード

表 26 は、HTTP/1.1 仕様での推奨事項に従って状況コードを示し、ユーザー・アプリケーションとの関連性を説明するとともに、適切なアクションを提示しています。

CICS では、これらの状況コードによって暗黙的に指示される可能性のある特定のアクションは実行しません。また、CICS は通常、メッセージの内容に対する状況コードの妥当性をチェックしません。状況コードが正しいこと、および必要なアクションを実行したことを、確認してください。各状況コードに適用される詳細情報と要件について、作業に使用している HTTP 仕様を確認してください。

表 26. HTTP サーバーとしての CICS: Web クライアントに送信されるユーザー作成の応答の状況コード

状況コードと通常の理由句	HTTP/1.0 クライアントに適切か?	この応答を提供する状況	メッセージ・ボディおよび HTTP ヘッダーに対する影響 (状況コードがユーザー・アプリケーションに対して適切な場合) 詳しくは HTTP 仕様を参照してください。
100 Continue (続行)	いいえ	使用しないでください。CICS は Expect 要求を処理し、自身で 100-Continue 応答を送信します。	
101 Switching Protocols (切り替えプロトコル)	いいえ	使用しないでください。CICS は HTTP バージョンまたはセキュリティー・プロトコルのアップグレードをサポートしていません。	
200 OK	はい	要求が満たされました。通常応答です。	通常応答のボディを提供してください。
201 Created (作成済み)	はい	新しいリソースを作成しました (リソースがまだ作成されていない場合は、202 Accepted (受け入れ済み) を使用してください)。	メッセージ・ボディの内容と 1 つ以上のヘッダーが必要です。

表 26. HTTP サーバーとしての CICS: Web クライアントに送信されるユーザー作成の応答の状況コード (続き)

状況コードと通常の理由句	HTTP/1.0 クライアントに適切か?	この応答を提供する状況	メッセージ・ボディおよび HTTP ヘッダーに対する影響 (状況コードがユーザー・アプリケーションに対して適切な場合) 詳しくは HTTP 仕様を参照してください。
202 Accepted (受け入れ済み)	はい	要求を受け入れましたが、まだ処理はしておらず、処理する保証もありません。	メッセージ・ボディの内容が必要です。
203 Non-Authoritative Information (権限以外の情報)	いいえ	使用しないでください。ユーザーが提供するヘッダーが権限情報を提供します。	
204 No Content (コンテンツなし)	はい	メッセージ・ボディを送信していません (おそらく、更新されたヘッダーのみを送信する必要があるため)。	メッセージ・ボディが許可されていません。
205 Reset Content (内容をリセット)	いいえ	要求を開始したフォームをクライアントにクリアさせます。	メッセージ・ボディが許可されていません。
206 Partial Content (部分的な内容)	いいえ	バイト範囲の要求をサポートし、この応答は要求を満たします。	通常応答のボディです。1 つ以上のヘッダーが必要です。
300 Multiple Choices (多肢選択)	はい	複数バージョンのリソース (例えば、異なる言語の文書) を提供できます。	メッセージ・ボディの内容と 1 つ以上のヘッダーが必要です。
301 Moved Permanently (永続的に移動)	はい	ユーザー・アプリケーションによる発行は推奨されていません。 CICS がアプリケーション・プログラムを呼び出すことなく正しい応答を生成するように、URIMAP 定義の LOCATION 属性と REDIRECTTYPE 属性を使用して、リダイレクトを管理することができます。 REDIRECTTYPE (PERMANENT) はこの状況コードを選択します。	
302 Found (検出)	はい	ユーザー・アプリケーションによる発行は推奨されていません。リダイレクト用に URIMAP 定義を使用すると、REDIRECTTYPE (TEMPORARY) はこの状況コードを選択します。	
303 See Other (他を参照)	いいえ	応答 (特に、POST 要求の結果に関する応答) を提供する別のリソースに対する GET 要求をクライアントに行わせます。	メッセージ・ボディの内容と 1 つ以上のヘッダーが必要です。
304 Not Modified (未変更)	はい	クライアントが条件付き要求を行いましたが、ユーザーが提供するリソースは変更されていません。アプリケーションによって動的に作成される応答は、要求のたびに変更される可能性が高いことに注意してください。変更されないリソースについては、URIMAP 定義を使用して静的応答を配信することを検討してください。	メッセージ・ボディが許可されていません (DOCTOKEN オプションを使用して、内容のない文書を指定できます)。1 つ以上のヘッダーが必要です。

表 26. *HTTP サーバーとしての CICS: Web クライアントに送信されるユーザー作成の応答の状況コード (続き)*

状況コードと通常の理由句	HTTP/1.0 クライアントに適切か?	この応答を提供する状況	メッセージ・ボディおよび HTTP ヘッダーに対する影響 (状況コードがユーザー・アプリケーションに対して適切な場合) 詳しくは HTTP 仕様を参照してください。
305 Use Proxy (プロキシーを使用)	いいえ	クライアントが指定されたプロキシーを経由して要求を行うようにします。	1 つ以上のヘッダーが必要です。
307 Temporary Redirect (一時的リダイレクト)	いいえ	ユーザー・アプリケーションによる発行は推奨されていません。 CICS は、URIMAP リダイレクトに、この状況コードではなく状況コード 302 を使用します。	
400 Bad Request (不正な要求)	はい	クライアントの要求は構文エラーまたは同様の問題が含まれていて処理できません。	メッセージ・ボディの内容が必要です。
401 Unauthorized (未許可)	はい	使用しないでください。TCPIPSERVICE 定義のセキュリティー設定に指定されている場合、CICS は基本認証プロセスを処理します。	
403 Forbidden (禁止)	はい	クライアントの要求を拒否します。	メッセージ・ボディの内容が必要です。
404 Not Found (未検出)	はい	要求に応答するためのリソースがありません。または、説明を返さずに要求を拒否します。または、関連性のあるその他の状況コードはありません。	メッセージ・ボディの内容が必要です。
405 Method Not Allowed (メソッド未許可)	いいえ	クライアントが、このリソースでサポートされていないメソッドを使用しました。	メッセージ・ボディの内容と 1 つ以上のヘッダーが必要です。
406 Not Acceptable (受け入れ不可)	いいえ	クライアントが Accept ヘッダーを使用して条件付き要求を行いましたが、クライアントの基準に合うバージョンのリソースがありません。この状況コードを使用する別の方法として、条件に合わない応答を送信できることに、注意してください。	メッセージ・ボディの内容が必要です。
407 Proxy Authentication Required (プロキシー認証が必要)	いいえ	使用しないでください。CICS はプロキシー・サーバーとして動作しません。	
408 Request Timeout (要求タイムアウト)	いいえ	ユーザー・アプリケーションによる発行は推奨されていません。 CICS Web サポートによる処理に対して、TCPIPSERVICE 定義で SOCKETCLOSE 属性を使用してタイムアウトを指定する必要があります。	
409 Conflict (競合)	いいえ	リソースが変更され、クライアントの要求を現状のリソースに適用できません。	メッセージ・ボディの内容が必要です。
410 Gone (終了)	いいえ	リソースは永続的に使用不可です。	メッセージ・ボディの内容が必要です。

表 26. *HTTP サーバーとしての CICS: Web クライアントに送信されるユーザー作成の応答の状況コード (続き)*

状況コードと通常の理由句	**HTTP/1.0 クライアントに適切か?**	**この応答を提供する状況**	**メッセージ・ボディおよび HTTP ヘッダーに対する影響 (状況コードがユーザー・アプリケーションに対して適切な場合) 詳しくは HTTP 仕様を参照してください。**
411 Length Required (長さが必要)	いいえ	使用しないでください。CICS では、正常なソケット受信のために、 HTTP/1.1 要求で Content-Length ヘッダーを指定する必要があります。	
412 Precondition Failed (前提条件の失敗)	いいえ	クライアントが条件付き要求を行い、条件が満たされませんでした。	メッセージ・ボディの内容が必要です。
413 Request Entity Too Large (要求エンティティーが大きすぎる)	いいえ	ユーザー・アプリケーションによる発行は推奨されていません。 CICS Web サポートによる処理に対して、TCPIPSERVICE 定義で MAXDATALEN 属性を使用して要求サイズ制限を指定する必要があります。	
414 Request URI Too Long (要求 URI が長すぎる)	いいえ	クライアントの要求 URL が長すぎてアプリケーションで処理できません。	メッセージ・ボディの内容が必要です。
415 Unsupported Media Type (サポートされないメディア・タイプ)	いいえ	クライアントによって送信されたメッセージ・ボディは、サポートされていないメディア・タイプまたはコンテンツ・コーディングになっています。	メッセージ・ボディの内容が必要です。
416 Requested Range Not Satisfiable (要求された範囲を満たせない)	いいえ	クライアントは、Range ヘッダー・フィールドを使用して (If-Range ヘッダー・フィールドは使用しないで) 要求を行いました。バイト範囲はサポートされていますが、その範囲がリソースに存在しませんでした。	メッセージ・ボディの内容と 1 つ以上のヘッダーが必要です。
417 Expectation Failed (予期の失敗)	いいえ	使用しないでください。CICS は Expect 要求を処理します。	
500 Internal Server Error (内部サーバー・エラー)	はい	アプリケーション・エラーまたはシステム・エラーのため要求を処理できません。	メッセージ・ボディの内容が必要です。

表 26. *HTTP サーバーとしての CICS: Web クライアントに送信されるユーザー作成の応答の状況コード (続き)*

状況コードと通常の理由句	HTTP/1.0 クライアントに適切か?	この応答を提供する状況	メッセージ・ボディおよび HTTP ヘッダーに対する影響 (状況コードがユーザー・アプリケーションに対して適切な場合) 詳しくは HTTP 仕様を参照してください。
501 Not Implemented (インプリメントなし)	はい	クライアントの要求に必要なメソッドはサポートされていません。この状況コードは、クライアントが HTTP/1.0 であるか、USER プロトコルを使用する場合にのみ発行してください。 HTTP プロトコルの場合、CICS は、認識できない要求を初期処理中に拒否します。メソッドが認識されてもリソースに適用されない場合、HTTP/1.1 クライアントに対しては「405 Method Not Allowed (メソッド未許可)」を使用してください。	メッセージ・ボディの内容が必要です。
502 Bad Gateway (無効なゲートウェイ)	はい	使用しないでください。CICS はプロキシーまたはゲートウェイとして動作しません。	
503 Service Unavailable (サービス利用不可)	はい	一時的に利用不可である別のアプリケーションまたはシステムにアクセスする必要がない限り、ユーザー・アプリケーションが、この状況コードの使用が適した状況になる可能性は高くありません。	メッセージ・ボディの内容と 1 つ以上のヘッダーが必要です。
504 Gateway Timeout (ゲートウェイのタイムアウト)	いいえ	使用しないでください。CICS はプロキシーまたはゲートウェイとして動作しません。	
505 HTTP Version Not Supported (サポートされていない HTTP バージョン)	いいえ	使用しないでください。CICS は、応答の HTTP バージョンをクライアントの要求の HTTP バージョンと突き合わせます。	

HTTP サーバーとしての CICS: サーバーからの応答で受信された状況コードの処理

432 ページの表 27 は、サーバーからの応答で受け取る可能性がある状況コードを示し、HTTP/1.1 仕様にある推奨事項に従って適切なアクションを提示しています。WEB RECEIVE コマンドは状況コードと状況テキストを返します。サーバーが理由句のテキストを HTTP 仕様で提示されているテキスト以外のものに変更している場合があることに留意してください。

各状況コードに適用される詳細情報と要件について、作業に使用している HTTP 仕様を確認してください。

表 27. *HTTP クライアントとしての CICS: 応答時の状況コードの処理*

状況コードと考えられる理由句	サーバーがこの状況コードを送信する理由	ユーザー・アプリケーション・プログラムによる推奨アクション
100 Continue (続行)	WEB SEND コマンドで ACTION(EXPECT) オプションを使用しました。サーバーは全メッセージ送信を受け入れます。	CICS は、メッセージ・ボディを送信することによってこの応答を処理します。ユーザー・アプリケーションはこの状況コードを受信しません。
101 Switching Protocols (切り替えプロトコル)	使用しないでください。プロトコルのアップグレードは、CICS Web サポートではサポートされていません。	ユーザー・アプリケーションはこの状況コードを受信しません。
200 OK	要求は正常終了しました。通常応答です。	計画どおりに応答の処理を続行します。
201 Created (作成済み)	リソースの作成を要求し、それが実行されました。	計画どおりに応答の処理を続行します。
202 Accepted (受け入れ済み)	サーバーは要求を受け入れますが、処理はまだ実行されていません。	計画どおりに応答の処理を続行しますが、行った変更はコミット済みとは限りません (コミットされないこともあります)。
203 Non-Authoritative Information (権限以外の情報)	メッセージ・ボディに関係するヘッダーが、サーバー上のヘッダーと完全に一致しません。	計画どおりに応答の処理を続行します。
204 No Content (コンテンツなし)	応答のメッセージ・ボディがありません。	計画どおりに応答の処理を続行しますが、受信するボディがありません。
205 Reset Content (内容をリセット)	サーバーは、要求を送信したフォームをユーザーがクリアすることを求めています。	要求を行うために使用していたフォーム・フィールドがあればクリアします。
206 Partial Content (部分的な内容)	Range ヘッダー・フィールドを使用して要求を行い、その要求は正常終了しました。	計画どおりに応答の処理を続行します。
300 Multiple Choices (多肢選択)	異なるバージョンのリソースが使用可能です。	提示された情報から優先バージョンを選択し、新しい要求を行います。サーバーの優先選択の URL を含む Location ヘッダーが存在している可能性があります。
301 Moved Permanently (永続的に移動)	リソースは新しい場所に永続的に移動しました。	サーバーによって (通常は Location ヘッダーで) 提供される URL に対して新しい要求を行い、これを今後すべての要求に使用します。
302 Found (検出)	リソースは一時的に新しい場所に移動しました。	サーバーによって (通常は Location ヘッダーで) 提供される URL に対して新しい要求を行います。ただし、これは、今後の要求には使用しないでください。
303 See Other (他を参照)	サーバーは、応答 (特に、POST 要求の結果に関する応答) を提供する別のリソースに対する GET 要求をユーザーに求めています。	GET メソッドを使用して、サーバーによって (通常は Location ヘッダーで) 提供される URL に対して新しい要求を行います。
304 Not Modified (未変更)	ユーザーが条件付き要求を行いましたが、リソースは変更されていません。	既存の格納済みバージョンの応答で情報を参照してください。ただし、これを最新情報としてユーザーに提示しないでください (CICS ではキャッシングがサポートされないため)。
305 Use Proxy (プロキシーを使用)	サーバーは、指定されたプロキシーを要求に使用することをユーザーに求めています。	サーバーによって (Location ヘッダーで) 提供された URL を使用して、新しい要求を行います。

表 27. *HTTP クライアントとしての CICS: 応答時の状況コードの処理 (続き)*

状況コードと考えられる理由句	サーバーがこの状況コードを送信する理由	ユーザー・アプリケーション・プログラムによる推奨アクション
307 Temporary Redirect (一時的リダイレクト)	「302 Found」と同様。	「302 Found」と同様。
400 Bad Request (不正な要求)	要求の構文に誤りがあります。	要求を確認して、変更を加え、再試行してください。
401 Unauthorized (未許可)	サーバーは許可を必要としています。または、ユーザーが提供した許可が拒否されました。	173 ページの『HTTP クライアントとしての CICS: 認証および識別』を参照してください。
403 Forbidden (禁止)	サーバーはユーザーの要求を拒否しました。	要求を繰り返さないでください。メッセージ・ボディに、要求が拒否された理由が含まれていることがあります。
404 Not Found (未検出)	要求された URL が見つかりませんでした。	要求が意図したとおりに指定されたことを確認してください。この状態は一時的である可能性があるので、後から再試行してください。
405 Method Not Allowed (メソッド未許可)	このリソースでサポートされていないメソッドが指定されました。	応答の Allow ヘッダーにあるサポートされているメソッドの一覧を読み、必要な場合、いずれかのメソッドを使用して新しい要求を行います。
406 Not Acceptable (受け入れ不可)	Accept ヘッダーを使用して要求を行いましたが、基準に合うバージョンのリソースがサーバーにありません。	メッセージ・ボディでサーバーに存在するリソースの情報を確認し、必要に応じて、いずれかのリソースに対する新しい要求を行います。
407 Proxy Authentication Required (プロキシー認証が必要)	プロキシー・サーバーは許可を必要としています。または、ユーザーが提供した許可が拒否されました。	173 ページの『HTTP クライアントとしての CICS: 認証および識別』を参照してください。
408 Request Timeout (要求タイムアウト)	サーバーはユーザーの要求が完了するのをこれ以上待機しません。	必要に応じて要求を繰り返します。アプリケーションのアセンブルとメッセージ送信を行うのに長時間かかっていないか、確認してください。
409 Conflict (競合)	リソースが変更されており、現状のリソースには、ユーザーの要求を適用できません。	メッセージ・ボディで競合の原因に関する情報を確認し、必要に応じて、この情報に基づいて新しい要求を行います。
410 Gone (終了)	リソースは永続的に使用不可です。	今後はこの要求を繰り返さないでください。
411 Length Required (長さが必要)	サーバーはユーザーに Content-Length ヘッダーの提供を求めています。	CICS は通常そのヘッダーを提供します。ただし、TCPIPSERVICE 定義で USER プロトコルを使用している場合は例外です。その場合は、ヘッダーを自分で記述して新しい要求を行ってください。
412 Precondition Failed (前提条件の失敗)	条件付き要求を行いましたが、条件が満たされませんでした。	計画どおり処理を続行します。要求に指定したアクションはいずれも適用されていないことに注意してください。

表 27. *HTTP クライアントとしての CICS: 応答時の状況コードの処理 (続き)*

状況コードと考えられる理由句	サーバーがこの状況コードを送信する理由	ユーザー・アプリケーション・プログラムによる推奨アクション
413 Request Entity Too Large (要求エンティティーが大きすぎる)	メッセージ・ボディが大きすぎるためサーバーで処理できません。	Retry-After ヘッダーを参照し、この状態が一時的なものかどうかを確認してください。待機するか、メッセージ・ボディの長さを減らして、再試行します。新しい接続を開くことが必要な場合があります。
414 Request URI Too Long (要求 URI が長すぎる)	要求 URL が長すぎるため、サーバーで処理できませんでした。	要求を確認して再試行するか、または要求を中止します。
415 Unsupported Media Type (サポートされないメディア・タイプ)	送信したメッセージ・ボディは、このリソースに対してサーバーがサポートしていないメディア・タイプまたはコンテンツ・コーディングになっています。	指定したメディア・タイプを確認し、エラーがあったら訂正して要求を繰り返します。
416 Requested Range Not Satisfiable (要求された範囲を満たせない)	Range ヘッダー・フィールドを使用して要求を行いましたが、その範囲はリソースに存在しませんでした。	Content-Range ヘッダーを読んで、リソースの実際の長さを確認し、必要な場合、適切なバイト範囲で要求を繰り返します。
417 Expectation Failed (予期の失敗)	WEB SEND コマンドで ACTION (EXPECT) オプションを使用しましたが、サーバーは全メッセージ送信を受け入れません。	ACTION (EXPECT) オプションを使用しないで同じ要求を繰り返すことができますが、再び失敗する可能性があります。要求が正しく指定されていることを確認し、エラーがあったら訂正して要求を繰り返します。
500 Internal Server Error (内部サーバー・エラー)	予期しないエラーが発生したため、サーバーは要求を処理できません。	この状態は一時的である可能性があるので、後から要求を再試行してください。
501 Not Implemented (インプリメントなし)	サーバーはこの要求メソッドをサポートしていません。	要求を繰り返さないでください。
502 Bad Gateway (無効なゲートウェイ)	要求はプロキシーまたはゲートウェイを経由しており、そのプロキシーまたはゲートウェイが別のサーバーから無効な応答を受信しました。	この状態は一時的である可能性があるので、可能な場合はプロキシーまたはゲートウェイを回避して、後で要求を再試行してください。
503 Service Unavailable (サービス利用不可)	サーバーは一時的に要求を処理できません。	Retry-After ヘッダーを読み、この状態が一時的なものかどうかを確認して、一時的である場合は、ヘッダーに指定された時間が経過してから再試行してください。
504 Gateway Timeout (ゲートウェイのタイムアウト)	要求はプロキシーまたはゲートウェイを経由しており、そのプロキシーまたはゲートウェイが別のサーバーから時間内の応答を受信しませんでした。	可能であればプロキシーまたはゲートウェイを回避して、必要に応じて要求を繰り返します。
505 HTTP Version Not Supported (サポートされていない HTTP バージョン)	使用しないでください。CICS Web サポートは、バージョンとして HTTP/1.1 を指定してクライアント要求を送信します。	ユーザー・アプリケーションはこの状況コードを受信しません。

付録 D. CICS Web サポートの HTTP メソッド・リファレンス

HTTP 要求にはメソッドが含まれています。メソッドとは、要求に含まれているデータに対してクライアントがサーバーに実行させるアクションを説明するキーワードです。 CICS Web サポートは、HTTP/1.1 仕様で定義されているすべての標準要求メソッドと、CICS の旧リリースで受け入れられていたいくつかの追加メソッドを実装しています。

メソッドの正しい使用方法、およびそれらに対する応答における正しいアクションの詳細なガイダンス、および適用可能な要件については、必ずご使用の HTTP の仕様書を参照してください。

- HTTP/1.1 Web クライアントから受信した要求の場合 (CICS が HTTP サーバーの場合)、HTTP/1.1 仕様で定義されている標準メソッドは受け入れられます。これらのメソッドは、GET、HEAD、POST、PUT、TRACE、OPTIONS、および DELETE です。
- HTTP/1.0 以下の Web クライアントから受信した要求の場合 (CICS が HTTP サーバーの場合)、以下に示す、HTTP/1.0 仕様で定義されているメソッド、およびいくつかの追加メソッドが受け入れられます。
  - HTTP/1.0 仕様で定義されているメソッドは、GET、HEAD、および POST です。
  - CICS へのインバウンド HTTP/1.0 要求で受け入れられる追加メソッドは、PUT、DELETE、LINK、UNLINK、および REQUEUE です。
- HTTP クライアントとしての CICS が行う要求の場合:
  - HTTP/1.1 仕様書で定義されている標準メソッドを使用することができます。これらのメソッドは、GET、HEAD、POST、PUT、TRACE、OPTIONS、および DELETE です。
  - LINK、UNLINK、および REQUEUE メソッドは、この目的にはサポートされていません。
  - 要求のバージョンは常に HTTP/1.1 として与えられます。
  - HTTP/1.0 サーバーの中には、HTTP/1.0 仕様では定義されていないメソッドを受け入れるものもあります。HTTP/1.0 サーバーは、受け入れることのできないメソッドに対しては、状況コード 501 (「Not Implemented (インプリメントなし)」) を返すはずです。
- メッセージ・ボディは、一部の要求メソッドには適切で、その他の要求メソッドには不適切な場合があります。
  - HTTP サーバーとしての CICS の場合、クライアント (特にユーザー作成クライアント) によっては、適切ではないのにメソッドにメッセージ・ボディを送信するものがありますが、これを処理するか無視するかは自由です。
  - HTTP クライアントとしての CICS の場合、CICS は、不適切な場合はメソッドに対するメッセージ・ボディの送信は行わず、適切な場合はメソッドに対してメッセージ・ボディの送信を要求します。

- CICS が HTTP サーバーである場合、CICS Web サポートは、Web クライアントから受信した要求に対して、メソッドおよびそのクライアントの HTTP バージョンに応じて、メソッドの応答としてある範囲のアクションを実行します。
  - ほとんどのメソッドを含む要求は、アプリケーション・プログラムに直接渡されて処理されます。
  - CICS は、ユーザー・アプリケーション・プログラムを呼び出さずに、OPTIONS および TRACE メソッドに対する適切な応答を返します。
  - メソッドが要求の HTTP バージョンで実装されていない場合、CICS はユーザー・アプリケーション・プログラムを呼び出さずに、Web クライアントにエラー応答を返します。
- サーバーによっては、HTTP 仕様で定義されている標準の要求メソッドのほかにも、拡張メソッドと呼ばれる非標準要求メソッドを実装している場合があります。
  - HTTP サーバーとしての CICS の場合、CICS Web サポートは、HTTP プロトコルでは非標準メソッドを含む要求は受け入れません (CICS Transaction Server for z/OS, バージョン 4 リリース 2 よりも前は、これらの要求は受け入れられ、非 HTTP として処理されていました)。非標準メソッドを含む要求を受信する必要がある場合は、HTTP 受け入れ検査が行われないユーザー定義プロトコル (TCPIPSERVICE 定義の USER オプション) を使用して、そのような要求を受信することができます。
  - HTTP クライアントとしての CICS の場合、**EXEC CICS** WEB API コマンドで非標準メソッドを使用することはできません。

このリファレンス・リストの表には、各メソッドを使用できる環境がリストされています。このリファレンスで言及されているメソッドの詳細なガイダンスについては、ご使用の HTTP の仕様書を調べてください。

HTTP サーバーとしての CICS: Web クライアントから受信した要求メソッドの処理

表 28 は、要求メソッドに対して CICS が実行するアクション、およびユーザー・アプリケーション・プログラムに対して提案されているアクションを示しています。詳細なガイダンスおよび関連する要件については、ご使用の HTTP の仕様書を確認することが重要です。

表 28. HTTP サーバーとしての CICS: Web クライアントから受信した要求メソッド

メソッド	HTTP/1.0 クライアントでの CICS アクション	HTTP/1.1 クライアントでの CICS アクション	要求に対してメッセージ・ボディは適切ですか?	ユーザー・アプリケーション・プログラムによる適切なアクション
GET (リソースの要求)	受け入れられます。要求はアプリケーションに渡されます。	受け入れられます。要求はアプリケーションに渡されます。	いいえ	リソースを Web クライアントに送信するか、またはリソースを送信できない理由を説明したエラー応答を送信します。
HEAD (応答ヘッダーの要求)	受け入れられます。要求はアプリケーションに渡されます。	受け入れられます。要求はアプリケーションに渡されます。	いいえ	同一リソースに対する GET 要求に対する応答の場合のように、リソースを Web クライアントに送信します。CICS は応答ボディを削除して、ヘッダーのみを残します。

表 28. *HTTP サーバーとしての CICS: Web クライアントから受信した要求メソッド (続き)*

メソッド	HTTP/1.0 クライアントでの CICS アクション	HTTP/1.1 クライアントでの CICS アクション	要求に対してメッセージ・ボディは適切ですか?	ユーザー・アプリケーション・プログラムによる適切なアクション
POST (入力データの送信)	受け入れられます。要求はアプリケーションに渡されます。	受け入れられます。要求はアプリケーションに渡されます。	はい	メソッドのサポートはオプションです。(フォーム・フィールドである可能性のある) データを抽出し、それを処理して Web クライアントに応答を送信します。リソースの変更または作成にも使用できます。その場合は、PUT 要求の場合と同様に処理されます。
PUT (新規項目の送信)	受け入れられます。要求はアプリケーションに渡されます。	受け入れられます。要求はアプリケーションに渡されます。	はい	メソッドのサポートはオプションです。要求が有効である場合には、メッセージの内容を使用して、指定された URL でリソースを作成するか、または適宜、既存のリソースをメッセージの内容で置き換えます。 Web クライアントに肯定応答を送信します。HTTP/1.1 仕様には、正しい操作を行うための詳細な要件が含まれています。 **ヒント:** この要求タイプは、ご使用の CICS Web サポートの実装には適用できない可能性があります。必要な場合は、指定された URL に URIMAP 定義を作成し、そのリソースを、静的応答として提供されるように保管することで、この適用を実現できる可能性があります。
TRACE (要求のパスおよび最終状態を参照)	状況コード 501 (「Not Implemented (インプリメントなし)」) で拒否されます。ユーザー・アプリケーションは呼び出されません。	受け入れられます。CICS は応答します。ユーザー・アプリケーションは呼び出されません。	いいえ	ユーザー・アプリケーションには渡されません。CICS は、元のヘッダーおよび CICS が取得したヘッダー (Via ヘッダーなど) を持つ要求を含む応答を返します。
OPTIONS (サーバーに関する情報の要求)	状況コード 400 (「Bad request (不正な要求)」) で拒否されます。 ユーザー・アプリケーションは呼び出されません。	CICS では、パスを伴わない OPTIONS のみがサポートされます (パスを伴った OPTIONS は、405 で拒否されます)。 注 : OPTIONS * は受け入れられます。 CICS は応答します。 ユーザー・アプリケーションは呼び出されません。	未定義	ユーザー・アプリケーションには渡されません。CICS は基本情報 (HTTP バージョンおよびサーバー・ソフトウェアの説明) を含む応答を返します。

表 28. *HTTP サーバーとしての CICS: Web クライアントから受信した要求メソッド (続き)*

メソッド	HTTP/1.0 クライアントでの CICS アクション	HTTP/1.1 クライアントでの CICS アクション	要求に対してメッセージ・ボディは適切ですか?	ユーザー・アプリケーション・プログラムによる適切なアクション
DELETE (リソースの削除)	受け入れられます。要求はアプリケーションに渡されます。	受け入れられます。要求はアプリケーションに渡されます。	いいえ	メソッドのサポートはオプションです。要求が有効な場合は、既存のリソースを削除し、Web クライアントに肯定応答を送信します。
LINK、UNLINK、REQUEUE	受け入れられます。要求はアプリケーションに渡されます。	状況コード 501 (「Not Implemented (インプリメントなし)」) で拒否されます。ユーザー・アプリケーションは呼び出されません。	未定義	HTTP/1.1 仕様に記述されていないため、使用しないことをお勧めします。互換性のために、HTTP/1.0 要求は引き続きアプリケーションに渡されます。

HTTP クライアントとしての CICS: サーバーへの要求のメソッドの使用

表 29 は、HTTP クライアント要求に対して CICS API がサポートしている要求をリストしています。各メソッドの正しい使用方法についてのガイダンス、およびメソッドを使用する HTTP クライアントに適用される要件については、ご使用の HTTP の仕様書を参照してください。

表 29. *HTTP クライアントとしての CICS: サーバーに送信される要求メソッド*

メソッド	HTTP/1.0 サーバーに送信されますか?	HTTP/1.1 サーバーに送信されますか?	要求にメッセージ・ボディはありますか?	目的
GET (リソースの要求)	はい	はい	いいえ	サーバーからリソースを取得します。
HEAD (応答ヘッダーの要求)	はい	はい	いいえ	サーバーからリソースのヘッダーを取得します。ボディ全体を取得せずに、リソースの性質、状況、またはサイズをチェックできるようにします。
POST (入力データの送信)	はい	はい	はい	サーバーにデータを送信します。例えば、フォーム・データはこの方法で送信されることがあります。サーバーはこのメソッドをサポートする必要はありません。
PUT (新規項目の送信)	サーバーがサポートしていないことがあります。	はい	はい	サーバー上のリソースを作成または変更します。要求の URL は、リソースがサーバー上に持っている URL です。この要求を使用して、既存の項目を更新したり、または新規の項目を作成したりできます。サーバーはこのメソッドをサポートする必要はありません。

表 29. *HTTP* クライアントとしての *CICS:* サーバーに送信される要求メソッド *(続き)*

メソッド	HTTP/1.0 サーバーに送信されますか?	HTTP/1.1 サーバーに送信されますか?	要求にメッセージ・ボディはありますか?	目的
TRACE (要求のパスおよび最終状態を参照)	サーバーがサポートしていないことがあります。	はい	いいえ	要求の最終状態、および要求がサーバーに到達するまでにたどるパス (Via ヘッダーで示されます) を示している応答を取得します。要求の処理に使用されているプロキシー・サーバーを確認することができます。サーバーはこのメソッドをサポートする必要はありません。
OPTIONS (サーバーに関する情報の要求)	サーバーがサポートしていないことがあります。	はい	許可されていますが、現時点では目的は定義されていません。	サーバー情報を取得します。要求パスとして * (アスタリスク) を指定してサーバー全体に要求を適用するか、または要求の絶対パスを指定して、そのリソースに関する情報を取得します。サーバーはこのメソッドをサポートする必要はありません。
DELETE (リソースの削除)	サーバーがサポートしていないことがあります。	はい	いいえ	サーバー上のリソースを削除します。要求 URL は、削除される項目の URL です。サーバーはこのメソッドをサポートする必要はありません。
一般的には、LINK、UNLINK、REQUEUE、および拡張メソッド	許可されていません。INVREQ 応答は返されますが、要求は送信されません。	許可されていません。INVREQ 応答は返されますが、要求は送信されません。	未定義	HTTP クライアントとしての CICS の WEB API では使用不可です。

付録 E. アナライザー・プログラム

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

アナライザー・プログラムは、TCPIPSERVICE 定義と関連付けられます。これらのプログラムの主な役割は、URIMAP 定義によってアナライザー・プログラムの使用が指定されている場合や、URIMAP 定義が存在しない場合に、HTTP 要求の解釈を行うことです。

アナライザー・プログラムは、CICS が HTTP クライアントであるときは呼び出すことができず、また Web サービス処理に対しても呼び出すことができません。アナライザー・プログラムは、CICS が HTTP サーバーであるときにのみ呼び出すことができます。CICS に対し、CICS Web サポート・プロセス中のアナライザー・プログラムが HTTP サーバーとしての役割をすることについては、31 ページの『HTTP サーバーとしての CICS に対する HTTP 要求および応答の処理』で説明されています。 71 ページの『第 5 章 HTTP サーバーとしての CICS に対して CICS Web サポートを使用可能にする』には、HTTP サーバーとしての CICS のアーキテクチャーを計画するのに役立つ情報が含まれています。

アナライザー・プログラムと URIMAP 定義の関係

CICS Transaction Server for z/OS バージョン 3 リリース 1 の前は、HTTP サーバーとしての CICS に対するすべての HTTP 要求を、アナライザー・プログラムで解釈しました。このリリースでは、URIMAP リソースは HTTP 要求の処理を制御するための戦略的方式です。これらの定義では、要求の URL を、要求を処理するアプリケーション・プログラムにマッチングすること、およびコンバーター・プログラムと別名トランザクションの使用を指定することで、アナライザー・プログラムの主要な機能を置き換えます。

ただし、処理ステージの一部を引き継ぐため、またその他のアクション (モニターまたは監査のアクションなど) を実行するために、選択した HTTP 要求について URIMAP 定義でアナライザー・プログラムを呼び出すことがあります。アナライザー・プログラムに対して、アナライザーの機能を複製する URIMAP 定義の属性、すなわち CONVERTER (コンバーター・プログラム名)、TRANSACTION (別名トランザクション)、USERID (別名トランザクションのユーザー ID)、および PROGRAM (要求を処理するアプリケーション・プログラムの名前) を渡すことができます。アナライザー・プログラムでは、これらの属性のオーバーライドを選択することもできます。

CICS Transaction Server for z/OS バージョン 3 リリース 1 の前に CICS Web サポートが使用したのと同じプロセスに従って、URIMAP 定義を使用せずに HTTP 要求を直接、アナライザー・プログラムに渡すという選択も可能です。ただし、URIMAP 定義を使用しない場合、特定の HTTP 要求に対する CICS の応答方法を変更するには、アナライザー・プログラム内のロジックを変更することが必要になります。URIMAP 定義を使用すると、これらの変更をシステム管理タスクとして動的に実行することができます。また、URIMAP 定義の代わりにアナライザー・プロ

グラムを使用して要求の処理を継続するにあたって、この観点から、HTTP/1.1 への準拠が必要な場合は、HTTP/1.1 仕様 (RFC 2616) に記述されている規則に従って URL 比較を実行するように、アナライザー・プログラムをコーディングする必要があります。

注: URIMAP で ANALYZER(YES) が指定されていても、その要求と一致する URIMAP 定義が検出された場合は、CICS 提供のサンプル・アナライザー・プログラム DFHWBADX および CICS 提供のデフォルト・アナライザー・プログラム DFHWBAAX は、提供時には、要求の分析をまったく実行しません。

エラー処理に対するアナライザー・プログラムの使用

現在、すべての HTTP 要求のパス処理にアナライザー・プログラムが必要なわけではありませんが、CICS Web サポートに使用されるそれぞれの TCPIPSERVICE リソースでは、依然としてアナライザー・プログラムを指定する必要があります。

リソース定義の URM 属性で、アナライザー・プログラムの名前を指定します。各 TCPIPSERVICE 定義に別々のアナライザーを指定することもできますし、複数の TCPIPSERVICE 定義に同じアナライザーを指定することもできます。 URIMAP 定義からアナライザー・プログラムを呼び出す場合には、複数の異なるアナライザー・プログラムの中から選択することはできません。その TCPIPSERVICE 定義用に指定されているアナライザー・プログラムを使用するか、使用しないかのみの選択となります。

TCPIPSERVICE 定義に指定したアナライザー・プログラムは、CICS がその要求と一致する URIMAP 定義を検出できない場合に、その HTTP 要求を処理するために呼び出されます。この定義が見つからない状態は、要求の URL 入力時のユーザー・エラーが原因か、または適切な URIMAP 定義がインストールされていないために発生した可能性があります (URIMAP 定義は存在するが、使用不可に設定されている場合、その要求はアナライザー・プログラムではなく、Web エラー・プログラムによって処理されます)。

そのため、最小限の処置として、各 TCPIPSERVICE 定義に指定するアナライザー・プログラムには、認識できない HTTP 要求を処理するためのプロシージャーを組み込み、適切なエラー応答を提供する必要があります。またユーザーは、URIMAP 定義によって処理されるはずであったと思われる要求を具体的に識別し、より適切なエラー応答を提供することができます。エラー状態でアナライザー・プログラムから出された出力は、Web エラー・プログラムに渡されます。この Web エラー・プログラムを使用して、HTTP 応答を変更することができます。 131 ページの『第 9 章 Web エラー・プログラム』では、この調整方法が説明されています。

CICS 提供のデフォルト・アナライザー・プログラム DFHWBAAX は、TCPIPSERVICE 定義で PROTOCOL(HTTP) が指定されている場合のデフォルトです。 DFHWBAAX は、そのポートを使用するすべての要求が URIMAP 定義で処理する必要がある場合の、基本エラー処理を提供しています。CICS Web サポートが CICS TS 3.1 以前で使用した URL 形式を使用しては、要求をサポートしません。URIMAP 定義で処理されない要求の処理を、アナライザー・プログラムで提供する必要がある場合、TCPIPSERVICE 定義で指定するアナライザー・プログラムは、CICS 提供のサンプル・アナライザー・プログラム DFHWBADX か、またはカスタ

マイズした独自のアナライザー・プログラムである必要があります。

一部の Web 非対応アプリケーション用、および非 HTTP メッセージ用にアナライザー・プログラムを使用する方法

Web 非対応のアプリケーションは、URIMAP 定義から直接呼び出されれば、正常に機能する可能性があります。しかし、一部は、アナライザー・プログラムからのみ提供可能な機能に依存している場合があります。以下のような環境では、HTTP 要求の処理パスの中でのアナライザー・プログラムの使用が必要になる可能性があります。

- Web 非対応のアプリケーションおよびコンバーター・プログラムを使用して応答を作成する場合で、CICS TS バージョン 3 以前に受信したであろう応答と同一の応答を Web クライアントが必要とするために、 CICS TS バージョン 3 以前との互換性処理用のフラグを立てる必要がある場合 (例えば、ユーザー作成のクライアントが、新しいエラー応答または追加の HTTP ヘッダーに関する作業を行う際に、問題が発生する可能性があります)。このフラグは、コンバーター・プログラムがストレージ・ブロック内で手動で応答を作成する場合にのみ機能します。コンバーター・プログラムが **EXEC CICS** WEB API コマンドを使用して応答を送信する場合、このフラグは無効です。
- Web 非対応のアプリケーションとコンバーター・プログラムを使用して応答を作成する場合で、ストレージ・ブロック内のコンバーター・プログラムに渡される Web クライアントの要求か、またはストレージ・ブロック内でコンバーター・プログラムが手動で作成する HTTP 応答のいずれかが、標準でないコード・ページ変換を必要とする場合。コンバーター・プログラムは、ストレージ・ブロック内で渡される HTTP 要求または応答のコード・ページ変換の設定を指定することはできません。処理パスにアナライザー・プログラムがない場合に CICS がコード・ページ変換用に使用する標準設定については、 466 ページの『コンバーター・プログラムの作成』で説明しています。これらの標準設定が適していない場合や、コード・ページ変換が不要の場合は、処理パスの中でアナライザー・プログラムを使用して、代わりのコード・ページ変換設定を指定することができます。アナライザー・プログラムを使用する方法の代わりの選択として、ストレージ・ブロックを使用するのではなくコンバーター・プログラムで **EXEC CICS** WEB API コマンドを使用して、Web クライアントの要求の検査や、応答の作成を行うことができます。この場合、コード・ページ変換は通常どおりに **EXEC CICS** WEB API コマンドで指定することができます。

アナライザー・プログラムでこうした状態のいずれかに対処する必要はある場合、要求に対して URIMAP 定義をセットアップすることも可能ですが、その定義ではアナライザー・プログラムを指定する必要があります。

TCPIPSERVICE 定義でユーザー定義の (USER) プロトコルを使用する非 HTTP 要求の場合、要求の処理にはアナライザー・プログラムが常に必要であり、URIMAP 定義は使用できません。非 HTTP 要求の処理方法については、 187 ページの『第 12 章 CICS Web サポートと非 HTTP 要求』で説明しています。

追加の処理に対するアナライザー・プログラムの使用

処理パスの中でアナライザー・プログラムを使用するかどうかが任意指定であるような状況では、以下の理由で、アナライザー・プログラムの使用を選択する場合があります。

- 要求の内容を基に、処理パスの要素への動的な変更を行うため。 HTTP 要求の各 URL は、単一の処理パスを定義する単一の URIMAP 定義と突き合わせされます。アナライザー・プログラムは、要求の内容を解釈し、要素を変更することができます。そうした要素には、要求を処理するアプリケーション・プログラム、コンバーター・プログラムの関与、またはその要求に使用される別名トランザクションとユーザー ID などがあります。
- プロセスにモニター・アクションまたは監査アクションを導入するため。アナライザー・プログラムは、これを行う場所として適しています。
- 既存の CICS Web サポート・アーキテクチャーを CICS TS バージョン 2 からアップグレードするため、および既存のアナライザー・プログラムで、要求を処理している間に維持したい追加機能 (コンバーター・プログラムへの情報の引き渡しなど) が提供されているため。

『アナライザー・プログラムの作成』では、アナライザー・プログラムが実行できる全範囲の機能について説明しています。

アナライザー・プログラムと URIMAP 定義の置き換え

通常は、アナライザー・プログラムの要求処理機能を URIMAP リソース定義で置き換えることができます。このリソース定義は、CICS システム・プログラミング・コマンドで変更および制御できます。

URIMAP 定義は、要求の URL を一致させ、それらをアプリケーション・プログラムにマップしたり、コンバーター・プログラム、別名トランザクション、およびユーザー ID を指定するために使用できます。使用しているアナライザー・プログラムが追加機能を提供している場合、引き続き URIMAP 定義の代わりにその機能を使用したり、その機能を URIMAP 定義と結合することができます。

URIMAP の使用に移行している間は次のようになります。

- 一度に少数の要求に対して、段階的に URIMAP リソース定義を導入できます。アナライザー・プログラムで実行される処理のタイプ、および要求を処理するアプリケーションのタイプに応じて、各要求の処理パスでアナライザー・プログラムの使用を続行するかどうかを選択できます。
- URIMAP リソース定義で処理される要求について、既存の URL を保持する代わりに、新しい URL を選択して公開することもできます。要求について古い処理パスの使用を中止する場合は、URIMAP 定義をセットアップして、古い URL から新しい URL に要求を永久的にリダイレクトすることができます。
- 何らかの要求に対する処理パスで使用されることがないとしても、アナライザー・プログラムには認識されない要求に対する基本的な処理プロシージャーが含まれている必要があります。アナライザー・プログラムは、TCPIPSERVICE 定義で依然として必要であり、エンド・ユーザーが URL をミスタイプした場合などに要求を受け取ります。

アナライザー・プログラムの作成

アナライザー・プログラムは、アセンブラー、C、COBOL、または PL/I で作成できます。

このタスクについて

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

アナライザー・プログラムの入出力パラメーターは、COMMAREA で渡されます。言語依存のヘッダー・ファイル、組み込みファイル、および COMMAREA にマップするコピーブックについては、456 ページの『アナライザー・プログラムに関する参照情報』で説明します。

アナライザー・プログラムが実行できる機能の全範囲は、以下のとおりです。

- 要求の処理を継続するか、CICS がエラー応答を Web クライアント返すかを決定する。
- 要求の内容、および URIMAP 定義からコンバーター・プログラムに渡されたパラメーターを分析して、必要な後続の処理ステージ、および各ステージの実行に必要な CICS リソースを決定する (**EXEC CICS** WEB API コマンドをこの分析の間に使用する場合があります)。
- アプリケーション・プログラムに渡す前に要求を処理するためのコンバーター・プログラムの名前を指定する。コンバーター・プログラムは、通常、Web 非対応のアプリケーション・プログラムと共に使用されます。ユーザー・トークンがアナライザー・プログラムに対して提供され、必要に応じてコンバーター・プログラムと通信します。 Web クライアントの要求は、パラメーター・リストによって指示されている 32K のストレージ・ブロック内のコンバーター・プログラムに渡されます。 465 ページの『付録 F. コンバーター・プログラム』で、コンバーター・プログラムの機能について説明しています。
- 要求を処理して応答を提供するユーザー作成アプリケーション・プログラムの名前を指定する。
- 処理の残りのステージを処理する別名トランザクションのトランザクション ID を指定する。
- 別名トランザクションに関連付けるユーザー ID を指定する。
- ストレージのブロックでコンバーター・プログラムに渡される要求のコード・ページ変換、およびコンバーター・プログラムがストレージのブロックで手動で作成する応答を指定または抑止する。このことは、**EXEC CICS** WEB API コマンドを使用して HTTP 要求を表示し、応答を作成するコンバーター・プログラムまたはユーザー作成アプリケーションに影響を与えません。コード・ページ変換は CICS から直接要求されます。 49 ページの『CICS Web サポートのコード・ページ変換』では、コード・ページ変換の処理について説明しています。
- アップグレードの目的で提供されている、 Web 非対応アプリケーションが CICS TS バージョン 3 以前の互換性処理を必要とする場所を示すフラグを指定します。このことは、**EXEC CICS** WEB API コマンドを使用して HTTP 要求を表示し、応答を作成するコンバーター・プログラムまたはユーザー作成アプリケーションに影響を与えません。
- 要求ボディを変更する。変更されたすべての内容は、コンバーター・プログラムにストレージのブロックで渡されたデータで見ることができますが、**EXEC CICS** WEB API コマンドに渡されたデータでは見ることは**できません**。

CICS は、デフォルト・アナライザー・プログラム DFHWBAAX (詳しくは 452 ページの『CICS 提供のアナライザー・プログラム DFHWBAAX』を参照)、およびサンプル・アナライザー・プログラム DFHWBADX (詳しくは 453 ページの『CICS 提供のサンプル・アナライザー・プログラム DFHWBADX』を参照) を提供します。これらのアナライザーが要求を満たさない場合、独自のアナライザーを作成する必要があります。 DFHWBADX を例として使用することができる場合があります。

ユーザーによる置換が可能なプログラムはすべて、 CICS Web サポートが動作しているシステムのローカル側になければなりません。プログラムの自動インストールを使用しない場合は、アナライザー・プログラムおよびコンバーター・プログラムを含む CICS Web サポートが使用するすべてのユーザー置換可能プログラムを定義し、そのプログラム定義をインストールする必要があります。プログラムの自動インストールを使用する場合は、ユーザーによる置換が可能なプログラムを正しい属性でインストールする必要があります。アナライザー・プログラムを EXECKEY(CICS) で定義する必要があることに注意してください。

ユーザー置換可能プログラムの作成について詳しくは、「*CICS Customization Guide*」のCustomizing with user-replaceable programsを参照してください。

アナライザー・プログラムへの入力

入力パラメーターは COMMAREA のアナライザー・プログラムに渡され、要求の性質と内容、および URIMAP 定義によって提供されるすべての入力に関する情報を与えます。アナライザー・プログラムは、これらの値を受け取って出力パラメーターとして渡すことも、あるいは要求内容の分析に基づいて動的に指定変更することもできます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

COMMAREA 内のすべてのパラメーターのリストと技術的説明については、 457 ページの『アナライザー・プログラム用のパラメーター』を参照してください。

入力パラメーターには、次の項目またはこれらの項目へのポインターが含まれます。

- アナライザー・パラメーター・リストの目印。
- クライアントおよびサーバー (HTTP サーバーとしての CICS) のコロン 16 進またはドット 10 進 IP アドレス。
- 要求が HTTP 要求であるかどうかを示す標識
- 要求について一致する URIMAP 定義が見つかったかどうかを示す標識。この標識が正の場合は、 URIMAP 定義が、アナライザー・プログラムに追加の入力パラメーターを渡した可能性があります。
- HTTP バージョン。
- 要求方式。
- 要求に指定されたホスト名。ホスト・ヘッダー (絶対 URI の場合は要求 URL) から取得されます。HTTP/1.1 要求では、ホスト名は必須であるため、このパラメーターは常にアナライザーに渡されます。 HTTP/1.0 要求では、ホスト名が提供されない場合があります。

- URL のパス構成要素。
- 要求で指定された照会ストリング。
- 要求の HTTP ヘッダー。

  チャンク転送コーディングを使用して要求が送信された場合、後続のヘッダーがメインの要求ヘッダーと共にアナライザー・プログラムに渡されることはありません。
- 要求ボディ、またはそのうちの 32 KB のストレージ・ブロックに収まる部分。この要求ボディは、要求が含まれている別個のストレージ・ブロックへのポインターです。

SSL クライアント認証を使用した接続で受け取られた HTTP 要求の場合は、次のパラメーターも渡されます。

- クライアント証明書から入手したユーザー ID

要求に一致する URIMAP 定義が見つかり、それによってアナライザー・プログラムが呼び出された場合は、その URIMAP 定義の後続のパラメーター (存在する場合) がアナライザー・プログラムに渡されます。

- アプリケーション・プログラムに渡す前に要求を処理する推奨されるコンバーター・プログラムの名前 (URIMAP 定義の CONVERTER 属性)。
- 要求を処理し、応答を提供する推奨されるユーザー作成アプリケーション・プログラムの名前 (URIMAP 定義の PROGRAM 属性)。
- 処理の残りの段階をカバーする推奨される別名トランザクションのトランザクション ID (URIMAP 定義の TRANSACTION 属性)。
- 別名トランザクションに関連付ける推奨されるユーザー ID (URIMAP 定義の USERID 属性)。このユーザー ID は、クライアントによってユーザー ID が提供される場合はオーバーライドされることがあります。

wbra_urimap 入力パラメーターを使用して、要求の処理パスで URIMAP 定義が使用されたかどうかをテストすることができます。

URIMAP 定義の代わりにアナライザー・プログラムを使用して要求を処理する場合、この点に関して HTTP/1.1 に準拠するには、HTTP/1.1 仕様の規則に従って URL 比較を実行するようにアナライザー・プログラムをコーディングする必要があります。これらの規則では、スキーム名とホスト名は大文字小文字を区別せず比較されますが、パスは大文字小文字を区別して比較されます。比較の前にすべての構成要素がアンエスケープされます。CICS は、 URL と URIMAP 定義を比較する場合、これらの規則に従います。

必要に応じて、**EXEC CICS** WEB API コマンドを使用して HTTP 要求を調べることもできます。 **EXEC CICS** WEB コマンドを使用すると要求の分析の精度と完全性が上昇する場合があります。特に、幅広い内容と使用法がある HTTP ヘッダーを調べる際に有効です。また、**EXEC CICS** WEB コマンドを使用すると、後続処理の決定要素になる可能性のある、要求の照会ストリングまたはフォーム・フィールド情報を見つけ、抽出する処理が単純化されます。

EXTRACT TCPIP コマンドを使用して、処理中のクライアント要求に関する以下の情報を入手できます。

- Web クライアントの IP アドレス
- DNS サーバーによって認識されている Web クライアントのホスト名
- Web クライアントが接続要求の送信に使用したポートの番号
- サーバー (HTTP サーバーとしての CICS) の IP アドレス
- 使用されている認証のタイプ
- 使用している SSL サポートのレベル
- 要求に関連付けされた TCPIPSERVICE リソース定義

アナライザー・プログラムからの出力

アナライザー・プログラムは、出力を COMMAREA で提供します。出力には、応答コード、および処理ステージをさらに指定して、情報をコンバーター・プログラムと共用するためのオプションの出力パラメーターの範囲が含まれます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

COMMAREA 内のすべてのパラメーターのリストと技術的説明については、457 ページの『アナライザー・プログラム用のパラメーター』を参照してください。

アナライザー・プログラムは、COMMAREA で以下のような出力を生成する必要があります。

- 応答コード。
  - アナライザー・プログラムが URP_OK の応答コードを戻した場合、処理は次のステップへ進みます。
  - アナライザー・プログラムが他の値を戻した場合、CICS は Web クライアントにエラー応答を戻します。応答はユーザー置換可能 Web エラー・プログラムを使用して変更できます。137 ページの『CICS Web サポートのデフォルトの状況コードおよびエラー応答』 では、アナライザーからの戻りコードが CICS により Web クライアントに戻される状況コードにマップする方法を説明します。

アナライザー・プログラムは、以下のような出力を提供する場合もあります。

- ユーザー作成アプリケーション・プログラムに渡す前に要求を処理する場合に使用されるコンバーター・プログラムの名前。
  - コンバーター・プログラム名が URIMAP 定義から入力された場合、この名前を受け入れるか指定変更できます。
  - コンバーター・プログラムは不要であるということをアナライザーが示している場合は、要求の先頭の 32K バイトがストレージのブロックでユーザー作成アプリケーション・プログラムに渡されます。 Web 対応アプリケーションは、これを無視して、**EXEC CICS** WEB API コマンドを使用し、要求を読み取ります。
- 要求を処理して応答を提供するアプリケーション・プログラムの名前。
  - プログラム名が URIMAP 定義から入力された場合、この名前を受け入れるか指定変更できます。

- コンバーター・プログラムを使用している場合は、コンバーター・プログラムからプログラム名を指定または指定変更できます。コンバーターをこのように使用して、要求で処理中の複数のプログラムを含めることができます。

- 処理の残りのステージを処理する別名トランザクションのトランザクション ID。トランザクション ID が URIMAP 定義から入力された場合、この名前を受け入れるか指定変更できます。
- 別名トランザクションに関連付けるユーザー ID。ユーザー ID が URIMAP 定義から入力された場合、この名前を受け入れるか指定変更できます。ユーザー ID が指定されていない場合に、CICS がどのようにして決定するかを以下に示します。
  - ユーザー ID が URIMAP 定義から入力された場合、この ID が使用されます。
  - HTTP 要求がクライアント認証に SSL を使用している場合は、ユーザー ID はクライアント証明書から入手されます。
  - その他の場合は、CICS のデフォルト・ユーザー ID が使用されます。
- コンバーター・プログラムがストレージのブロックで手動で作成する場合の、要求を含むストレージの 32K ブロックのコード・ページ変換に関連するパラメーター、および応答ボディのコード・ページ変換に関連するパラメーター。

  注: このことは、**EXEC CICS** WEB API コマンドを使用して HTTP 要求を表示し、応答を作成するコンバーター・プログラムまたはユーザー作成アプリケーションに影響を与えません。コード・ページ変換は CICS から直接要求されます。以下の 2 つのいずれかの方法で、要求を含むストレージのブロックの変換のためのパラメーターを指定できます。
  - Web クライアントにより使用される文字セットを指定するパラメーターのペア (wbra_characterset)、およびアプリケーション・プログラムに適切なホスト・コード・ページ (wbra_hostcodepage) として。この方法でパラメーターを指定すると、コード・ページ変換テーブル (DFHCNV) のエントリーは必要とされません。
  - DFHCNV コード・ページ変換テーブルのエントリーに対するキーとして (wbra_dfhcnv_key)。これはアップグレードを目的とする場合を除いて推奨されません。

  これらのパラメーターのいずれも指定しないと、CICS では、 441 ページの『付録 E. アナライザー・プログラム』 に記載された標準設定を使用したテキスト・メッセージの変換をデフォルトで行います。ストレージのブロックの要求および応答のコード・ページ変換を抑止する場合は、wbra_dfhcnv_key をヌルまたはブランクに設定します。
- コンバーター・プログラムを使用する Web 非対応アプリケーションを示すフラグでは、CICS TS バージョン 3 以前の互換性処理 (wbra_commarea) が必要になります。このフラグはアップグレードの目的のために提供されます。このフラグは、Web クライアントが CICS TS バージョン 3 以前に受信した応答と同一の応答を必要とする特定の状況で、 **EXEC CICS** WEB API コマンドを使用しないアプリケーションによってのみ使用されます (つまり、ストレージのブロックで応答を手動で作成します)。このフラグを設定すると、以下のようになります。

– CICS は、HTTP/1.1 メッセージに通常挿入される応答ヘッダーを追加しません。 CICS Transaction Server for z/OS バージョン 3 リリース 1 の前にクライアントに送信されるヘッダーのみが使用されます。
– エラー処理が必要な場合、CICS は、Web クライアントの HTTP バージョンに関係なく、HTTP/1.0 応答に適切で、HTTP/1.0 応答とラベルが張られたエラー応答を送信します。CICS は、通常、HTTP/1.1 クライアントに HTTP/1.1 エラー応答を使用して応答しますが、このことにより、アプリケーションが通常 HTTP/1.1 応答を送信するとクライアントに見なされる場合があります。

- アナライザー・プログラムとコンバーター・プログラムで情報を共用するために使用される 8 バイトのユーザー・トークン。このメカニズムについては、『アナライザー・プログラムとコンバーター・プログラムでのデータの共用』を参照してください。
- 要求ボディの長さの変更値。

アナライザーは、要求の内容を変更できます。

- 変更されたデータは、オリジナルのデータよりも短くなる場合もあれば、同じ長さの場合もあります。要求ボディを長くすることはできません。
- 変更されたすべての内容は、コンバーター・プログラムに渡されたデータで見ることができますが、**EXEC CICS** WEB API コマンドに渡されたデータでは見ることは**できません**。

アナライザー・プログラムとコンバーター・プログラムでのデータの共用

CICS は、処理ステージによるデータの共用を可能にする 3 つのパラメーターをアナライザー・プログラムとコンバーター・プログラム間でやり取りします。

user_data ポインター

このパラメーターには、ステージ間でやり取りする 32K のストレージ・ブロックのアドレスが含まれています。アナライザー・プログラムへの入り口で、ポインターは HTTP 要求が含まれているストレージ・ブロックを指し示します。コンバーター・プログラムのエンコード機能が完了すると、CICS Web サポートはこのポインターを使用して、HTTP 応答が含まれているストレージ・ブロックを見つけます (代わりに、応答を作成するために **EXEC CICS** WEB API コマンドが使用された場合は除きます)。

アナライザー・プログラムではポインターの値を変更してはなりません。ただし、ポインターによって示されたストレージ・ブロックの内容は変更できます。

コンバーター・プログラムとユーザー作成アプリケーション・プログラムの間で、ポインターを無変更のままステージ間でやり取りすることもできますし、 1 つのプログラムで GETMAIN コマンドを出し、新規に取得したストレージをポインターに渡すこともできます。

user_data 長

このパラメーターは、user_data ポインターによって示されたストレージ・ブロックの長さです。

ユーザー・トークン

ユーザー・トークンは、アナライザー・プログラムとコンバーター・プログ

ラムによって共用される 8 バイトのフィールドです。ユーザー・トークンには、以下のような任意の情報を含めることができます。

- 少量の共用情報はユーザー・トークンに直接渡すことができます。
- 大量の情報を渡すには、1 つのプログラムで GETMAIN コマンドを出して共用作業域用のストレージを獲得することができます。ユーザー・トークンを使用して共用ストレージのアドレスを渡します。

各プログラムでユーザー・トークンの内容を変更することができます。例えば、アナライザー・プログラムからコンバーター・プログラムのデコード機能へ渡すときのユーザー・トークンの意味と、エンコード機能へ渡すときのユーザー・トークンの意味を違ったものにすることができます。

アナライザー・プログラムは、コンバーター・プログラムに渡されるパラメーター・リスト内の任意のパラメーターを変更することができます。ポインターを変更することはできませんが、そのポインターによって指し示されるデータは変更可能です。各フィールドの長さを変更することはできません。

注: アナライザー・プログラムとコンバーター・プログラムはそれぞれ異なる CICS タスクのもとで実行されます。したがって、アナライザー・プログラムで GETMAIN コマンドを出す場合は、ストレージがコンバーター・プログラムで可視になるのであれば、 SHARED オプションをコーディングする必要があります。一般に、 SHARED オプションで獲得されたストレージは、CICS によって自動的には解放されないため、ユーザー・プログラムがストレージを必要としなくなった時点で GETMAIN コマンドを出す必要があります。しかし、HTTP 応答が Web クライアントへ送信された後は、user_data ポインターによって示されたストレージを CICS が解放します。

アナライザー・プログラムからのエスケープ・データまたはアンエスケープ・データの選択

構文解析のためにアナライザー・プログラムに渡された HTTP 要求はエスケープ形式になっています。 URL またはメッセージ・ボディのフォーム・データ内の予約文字または除外文字は、%xx シーケンスとして表されます。ここで、xx は、予約文字の ASCII 16 進表記です。アナライザーは、ストレージの 32K ブロック内の要求を、エスケープ・シーケンスをそのまま使用したエスケープ形式で以後の処理段階に渡したり、エスケープ・シーケンスを元の文字に戻したアンエスケープ形式で渡したりできます。 **EXEC CICS** WEB API コマンドを使用した Web アプリケーション・プログラムは応答を受け取る際にこの仕組みを使用せず、CICS から直接、アンエスケープの処理を要求します。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

エスケープとその目的については、 18 ページの『予約文字と除外文字』を参照してください。エスケープおよびアンエスケープは HTTP 要求の以下の要素にのみ適用されます。

- 照会ストリングを含む、照会行の URL 部分。照会ストリングは GET メソッドを持つフォームからのデータの場合があります。

- POST メソッドを持ったフォームから戻されたフォーム・データおよびデフォルトのエンコードの **application/x-www-form-urlencoded**。このデータはメッセージ・ボディで表されます。フォーム・データについて詳しくは、19 ページの『HTML フォーム』を参照してください。

ストレージの 32K ブロック内の要求をアンエスケープ形式で渡す場合、アナライザーはデータをエスケープ形式からアンエスケープ形式に変換することもできるし、CICS に変換を行わせることもできます。

- 要求をエスケープ形式で渡すには、アナライザーで WBRA_UNESCAPE を WBRA_UNESCAPE_NOT_REQUIRED に設定します。WBRA_UNESCAPE_NOT_REQUIRED はデフォルト値です。
- 要求をアンエスケープ形式で渡し、その変換を CICS に行わせる場合は、アナライザーで WBRA_UNESCAPE を WBRA_UNESCAPE_REQUIRED に設定します。
- アナライザーによる変換が行われた後で要求をアンエスケープ形式で渡すには、WBRA_UNESCAPE を WBRA_UNESCAPE_NOT_REQUIRED に設定します。

EXEC CICS WEB API コマンドを使用した Web 対応アプリケーション・プログラムは、応答の送受信の際に COMMAREA の仕組みを使用せず、CICS から直接、アンエスケープの処理を要求します。**EXEC CICS** WEB API コマンドを使用する Web 対応アプリケーションの場合、WEB READ FORMFIELD コマンドまたはフォーム・フィールド・ブラウズ・コマンドを使用して要求からデータを抽出すると、 CICS がアンエスケープ処理を実行し、データはアンエスケープ形式で戻されます。WEB EXTRACT コマンドを使用して要求から照会ストリングを抽出する場合、データはエスケープ形式で戻されます。

CICS Web サポートまたは CICS ビジネス・ロジック・インターフェースのいずれかを介して実行できる COMMAREA インターフェースを持つアプリケーションを作成する場合は、WBRA_UNESCAPE が WBRA_UNESCAPE_NOT_REQUIRED に設定され、すべてのアンエスケープ処理がアプリケーションに委任されていることを確認する必要があります。この処置が行われていないと、 CICS ビジネス・ロジック・インターフェースはアンエスケープ・データをアプリケーションに渡し、CICS Web サポートはエスケープ・データをアプリケーションに渡すことになり、予期しない結果が生じることがあります。

CICS 提供のアナライザー・プログラム DFHWBAAX

CICS は、デフォルトのアナライザー・プログラム DFHWBAAX を提供しています。 DFHWBAAX は、CICS Web サポート用に使用される TCPIPSERVICE リソース定義のエラー処理関数を提供します。これは、1 つのポートを使用するすべての要求が、URIMAP 定義を使用して処理される場合に適しています。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

CICS は、アセンブラーの DFHWBAAX のソース・コードのみ、提供しています。

DFHWBAAX は、PROTOCOL(HTTP) を指定する TCPIPSERVICE 定義のデフォルトのアナライザー・プログラムです。

DFHWBAAX は、COMMAREA の標準アナライザー・プログラムと同じ入出力パラメーターを受け取ります。提供時の状態では、これらのパラメーターのほとんどは利用されておらず、CICS TS 3.1 より前の CICS Web サポートで使用されていた URL 形式を使用した要求のサポートは提供されません。その代わりに、以下のような簡素化された処置が行われます。

- DFHWBAAX は、URIMAP で ANALYZER(YES) が指定されている場合でも、その要求と一致する URIMAP 定義が検出された場合は、それ以上の処理は行いません。 wbra_urimap 入力パラメーターを使用して URIMAP 定義の存在を検査し、結果が肯定の場合は、その要求 URL に対する一切の分析を行わずに戻します。つまり、別名トランザクション、コンバーター・プログラム (使用されている場合)、およびアプリケーション・プログラム用の URIMAP 定義で指定されている設定が自動的に受け入れられ、それ以降の処理工程を決定するために使用されます。
- 一致する URIMAP 定義が見つからない場合、DFHWBAAX はユーザーによる置換が可能な Web エラー・トランザクション・プログラム DFHWBERX に制御を渡して、エラー応答を作成します。これは、wbra_server_program 出力パラメーターを使用して、 DFHWBERX を、その要求を処理するアプリケーション・プログラムとして設定することで達成されます。 DFHWBAAX はそれ以外の変更を COMMAREA に加えません。制御を受け取ると、DFHWBERX は 404 (Not Found (未検出)) 状況コードを持つ HTTP 応答か SOAP 障害応答のいずれかを、Web クライアントによって作成される要求に応じて提供します。

DFHWBAAX は、標準的な範囲の応答、**URP_OK**、**URP_EXCEPTION**、および **URP_INVALID** を使用します。提供時、DFHWBAAX には理由値は作成されません。理由が **URP_OK** 以外の場合は、処理中のエラーがあり、制御は、 Web エラー・アプリケーション・プログラム DFHWBERX ではなく、ユーザーによる置換が可能な Web エラー・プログラム DFHWBEP に渡されることを意味します。

CICS 提供のサンプル・アナライザー・プログラム DFHWBADX

CICS には、作業サンプル・アナライザー・プログラム DFHWBADX が提供されています。アナライザー・プログラムによる要求処理を、URIMAP 定義とともに、または URIMAP 定義の代わりに、提供する必要がある場合は、独自のアナライザー・プログラムを作成する際の開始点として、DFHWBADX を使用することができます。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

CICS は、以下の言語のソース・コードを提供しています。

- DFHWBADX (アセンブラー)
- DFHWBAHX (C)
- DFHWBALX (PL/I)
- DFHWBAOX (COBOL)

DFHWBADX は、提供された状態では、URIMAP で ANALYZER(YES) が指定されている場合でも、その要求と一致する URIMAP 定義が見つかったときは要求の分析を行いません。つまり、別名トランザクション、コンバーター・プログラム、お

よびアプリケーション・プログラム用の URIMAP 定義で指定されている設定が自動的に受け入れられ、それ以降の処理工程を決定するために使用されます。

DFHWBADX は wbra_urimap 入力パラメーターを使用して URIMAP 定義の存在を検査し、結果が肯定の場合は、その要求 URL に対する一切の分析を行わずに戻ります。独自のアナライザー・プログラムを作成して URIMAP 定義と相互作用するようにする場合、DFHWBADX の処理のこの特徴はコピーしないでください。アナライザー・プログラムの処理を別の方法で変更するために、wbra_urimap 入力パラメーターを検査したい場合があります。例えば、URIMAP 定義からの入力パラメーターを基に分析を行うか、要求 URL で直接分析を行うかを決定するために、パラメーターの検査を行う場合があります。

DFHWBADX による要求 URL の解釈方法

DFHWBADX は、URL のパス構成要素が以下のような構文を持つ HTTP 要求を解釈します。

図 28. DFHWBADX によって解釈されるパス構成要素の構文

アナライザー・プログラムによって処理されたフィールドはすべて、大文字に変換されます。変換後は、以下のようになります。

converter
要求に使用するコンバーター・プログラムの名前を指定します。最高 8 文字の長さにすることができます。

特殊なケースとして、4 文字値 'CICS' は、コンバーター・プログラムを使用していないことを表します。URIMAP 定義が設定されているコンバーター・プログラムを使用する方法については、465 ページの『付録 F. コンバーター・プログラム』を参照してください。

alias
以降の要求処理のための別名トランザクションのトランザクション ID を指定します。最高 4 文字の長さにすることができます。

program
要求を処理するために使用する CICS アプリケーション・プログラムの名前を指定します。最高 8 文字の長さにすることができます。

ignored
パスのこの部分を、DFHWBADX は無視します (ただし、コンバーター・プログラムまたはアプリケーション・プログラムが使用する可能性があります)。

token
先頭の 8 バイトは、コンバーター・プログラムに渡されるユーザー・トークンを指定します。トークンの先頭 8 バイトの後に続くデータを、DFHWBADX は無視します (ただし、コンバーター・プログラムまたはアプリケーション・プログラムが使用する可能性があります)。

パスの例として、/cics/cwba/dfh$wb1a では、以下のとおりです。

- コンバーター・プログラムは使用されません。
- 別名トランザクションは CWBA です。
- CICS アプリケーション・プログラムは DFH$WB1A です。

オリジナルの HTTP 要求から派生した出力に加え、DFHWBADX は以下の出力も設定します。

- コード・ページ変換テンプレートは DFHWBUD です。このテンプレートは、サンプル変換テーブル DFHCNVW$ に定義されており、ASCII Latin-1 文字セット (コード・ページ ISO 8859-1) と EBCDIC Latin 文字セット (コード・ページ 037) のデータを変換します。このサンプル変換テーブルは、なにも構成を加えずに使用できますが、出力パラメーター `wbra_characterset` および `wbra_hostcodepage` を `wbra_dfhcnv_key` 出力パラメーターの代わりに使用すると、より強力な制御が提供され、変換テーブルの使用を回避できることに、注意してください。
- DFHWBADX は、要求をエスケープ形式で渡し、WBRA_UNESCAPE_NOT_REQUIRED を設定します。

DFHWBADX からの応答

DFHWBADX によって作成された応答の意味は、以下のとおりです。

URP_OK
アナライザーは、要求がデフォルトの HTTP 要求形式に準拠していることを検出し、別名用の適切な出力を生成しました。

URP_EXCEPTION
アナライザーは、要求がデフォルト形式に準拠していないことを検出しました。次のような理由コードが提供されます。

1 リソースの長さが 6 未満でした (DFHWBADX によって認識される URL 形式で、可能な最短のリソース指定は `/A/B/C` です。これは、コンバーター A を使ってトランザクション B のもとでプログラム C を実行するよう要求します)。この応答と理由は、着信要求が HTTP 要求でないときに使用されるものです。

2 リソース指定が『/』で始まっていませんでした。

3 リソース指定が 1 つの『/』を含んでいましたが、3 個未満です。

4 リソース指定の中のコンバーター名の長さが 0 であったか、または 8 を超えていました。

5 リソース指定の中のトランザクション名の長さが 0 であったか、または 4 を超えていました。

6 リソース指定の中の CICS アプリケーション・プログラム名の長さが 0 であったか、または 8 を超えていました。

応答および理由コードは、メッセージ DFHWB0723 に表示されます。状況コードが 400 (Bad Request (不正な要求)) のエラー応答が、Web クライアントに返されます。ユーザーによる置換が可能な Web エラー・プログラム DFHWBEP を使用すると、この動作を変更することができます。

URP_INVALID
目印が無効です。これは内部エラーを示します。

アナライザー・プログラムに関する参照情報

このセクションでは、アナライザー・プログラムに関する参照情報を提供します。これには、入出力パラメーター、応答コードおよび理由コードなどが含まれます。

アナライザー・プログラム用のパラメーターのまとめ

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

アナライザー・プログラム用のパラメーターおよび定数の名前は、サポートされている各種のプログラム言語に適合した形式に変換され、CICS の一部として提供されているファイルに定義されています。

言語	パラメーター・ファイル	定数ファイル
アセンブラー	DFHWBTDD	DFHWBUCD
C	DFHWBTDH	DFHWBUCH
COBOL	DFHWBTDO	DFHWBUCO
PL/I	DFHWBTDL	DFHWBUCL

これらのファイルは、COMMAREA 内にあるフィールドのデータ型に関する言語固有の情報を提供します。これらのファイルを使用する場合は、変換プログラムのステップで XOPTS(NOLINKAGE) を指定する必要があります。指定しないと、コンパイルが失敗します。

次の表では、パラメーターの名前は省略形で示されています。表中のそれぞれの名前をパラメーター名にするには、接頭部として **wbra_** を付ける必要があります。

表 30. アナライザー・プログラム用のパラメーター

Input wbra_	Inout wbra_	Output wbra_
client_ip_address	**alias_tranid**	**application_style**
client_ipv6_address	**converter_program**	**alias_termid**
content_length	**server_program**	**characterset**
eyecatcher	**user_data_length**	**commarea**
function	**userid**	**dfhcnv_key**
hostname_length		**hostcodepage**
hostname_ptr		**reason**
http_version_length		**response**
http_version_ptr		**unescape**
method_ptr		**user_token**
method_length		
querystring_length		
querystring_ptr		
request_header_length		
request_header_ptr		
request_type		
resource_escaped_ptr		
resource_length		
resource_ptr		
server_ip_address		
server_ipv6_address		
urimap		
user_data_ptr		
version		

アナライザー・プログラム用のパラメーター

アナライザー・プログラム用のパラメーターの名前を、入力専用パラメーターか、出力専用パラメーターか、または入出力パラメーターであるかなどの簡単な説明と共に示します。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

wbra_alias_tranid
(入出力)

長さ 4 のストリング。この要求の処理の残りを処理する別名トランザクションのトランザクション ID。URIMAP 定義が含まれる場合、このストリングには TRANSACTION 属性の値が含まれます。このフィールドを設定しない場合や、それをブランクに設定した場合は、CWBA が使用されます。

wbra_alias_termid
(出力のみ)

長さ 4 のストリング。この要求の処理の残りを処理する別名トランザクションの START 要求で使用される端末 ID。

wbra_characterset
(出力のみ)

要求のエンティティー・ボディでクライアントが使用した IANA 文字セットの名前。この情報は、要求および応答のエンティティー・ボディのコード・ページ変換で使用されます。要求が HTTP 要求ではない場合、この文字セットを使用して要求および応答全体を変換します。**wbra_hostcodepage** も提供される必要があります。

wbra_client_ip_address
(入力のみ)

wbra_client_ipv6_address が指定されていない場合にクライアントの 2 進 IPv4 アドレスを指定する、フルワードの 32 ビット・フィールド。**wbra_client_address** では、IPv6 アドレスはサポートされていません。

wbra_client_address にゼロ以外の値が設定されている場合は、この値が使用されて **wbra_client_ipv6_address** の値は無視されます。そのため、IPv6 アドレッシングを使用する場合は、**wbra_client_address** の内容をクリアして、**wbra_client_ipv6_address** の値が使用されるようにする必要があります。

wbra_client_ipv6_address
(入力のみ)

IPv6 アドレッシングを使用する場合や、IPv4 アドレッシングを使用していて **wbra_client_address** が指定されていない場合に設定する必要がある、16 バイトのフィールド。このフィールドは IPv4 アドレスと IPv6 アドレスの両方をサポートしており、クライアントの 2 進 IPv6 アドレス、または IPv6 形式によるクライアントの IPv4 アドレスに設定されます。IP アドレス形式の詳細については、6 ページの『IP アドレス』を参照してください。

wbra_commarea
(出力のみ)

CICS TS バージョン 3 以前の互換性処理が、Web 非対応アプリケーションおよびコンバーター・プログラムを使用する応答に対して必要であることを示すフラグ。このフラグは、CICS TS バージョン 3 以前に受信した応答と同一の応答を Web クライアントが受信することを意味します。

wbra_content_length
(入力のみ)

受信したデータの Content-Length HTTP ヘッダーで指定されているエンティティー・ボディ長を表す 32 ビットの 2 進表現。

wbra_converter_program
(入出力)

長さ 8 のストリング。要求の処理に使用されるコンバーター・プログラムの名前。URIMAP 定義が含まれる場合、このストリングには CONVERTER 属性の値が含まれます。このフィールドが出力に設定されていない場合、コンバーター・プログラムは呼び出されません。

wbra_dfhcnv_key
(出力のみ)

長さ 8 のストリング。要求および応答のエンティティー・ボディのコード・ページ変換に使用する DFHCNV テーブル内にある変換テンプレートの名前。要求が HTTP 要求ではない場合、このテンプレートを使用して要求および応答全体を変換します。

CICS によりこのフィールドが高い値に初期設定されます。このフィールドを変換テンプレートの指定に使用する場合、選択する名前は、67 ページの『コード・ページ変換テーブル (DFHCNV) のエントリーのアップグレード』に説明されているように、DFHCNV テーブルで定義する必要があります。代わりの方法としては、**wbra_hostcodepage** および **wbra_characterset** フィールドを設定して、コード・ページ変換で使用するコード・ページのペアを指定します。**wbra_dfhcnv_key** をヌルまたはブランクに設定して、**wbra_hostcodepage** および **wbra_characterset** を設定しない場合、コード・ページ変換は抑止されます。

wbra_eyecatcher
(入力のみ)

長さ 8 のストリング。値は、`">analyze"` です。

wbra_function
(入力のみ)

アナライザー・プログラムが呼び出されることを示すコード。値は 1 です。

wbra_hostcodepage
(出力のみ)

要求を処理しているアプリケーション・プログラムに適切なホスト・コード・ページ (IBM EBCDIC コード・ページ) の名前。この情報は、要求および応答のエンティティー・ボディのコード・ページ変換で使用されます。要求が HTTP 要求ではない場合、このコード・ページを使用して要求および応答全体を変換します。`wbra_characterset` も提供される必要があります。

wbra_hostname_length
(入力のみ)

HTTP 要求に指定されたホスト名の長さ (バイト)。ホスト名が指定されていない場合、この値は未定義です。

wbra_hostname_ptr
(入力のみ)

クライアントが送信する HTTP 要求に指定されたホスト名のポインター。絶対 URI が要求で使用された場合、ホスト名はその URI から取得されます。使用されない場合、ホスト名は要求のホスト・ヘッダーで指定された名前になります。HTTP/1.1 要求では、ホスト名は必須であるため、このパラメーターは常にアナライザーに渡されます。HTTP/1.0 要求では、ホスト名は提供されない場合があり、このとき値は未定義です。

wbra_http_version_length
(入力のみ)

HTTP 要求の場合は、クライアントの要求の HTTP のバージョンを識別するストリングの長さ (バイト)。この要求が HTTP 要求でない場合、値はゼロです。

wbra_http_version_ptr
(入力のみ)

HTTP 要求の場合は、クライアントの要求の HTTP のバージョンを識別するストリングを指すポインター。この要求が HTTP 要求でない場合、値は未定義です。

wbra_method_length
(入力のみ)

HTTP 要求の場合は、この HTTP 要求に指定されたメソッドを識別するストリングの長さ (バイト)。この要求が HTTP 要求でない場合、値はゼロです。

wbra_method_ptr
(入力のみ)

HTTP 要求の場合は、この HTTP 要求に指定されたメソッドを指すポインター。この要求が HTTP 要求でない場合、値は未定義です。

wbra_querystring_length
(入力のみ)

HTTP 要求に指定された照会ストリングの長さ (バイト)。照会ストリングが送信されない場合、この値は未定義です。

wbra_querystring_ptr
(入力のみ)

クライアントが送信する HTTP 要求に指定された照会ストリングのポインター。照会ストリングが送信されない場合、この値は未定義です。

wbra_reason
(出力のみ)

アナライザー・プログラムによって戻される理由コード。 463 ページの『応答および理由コード』を参照してください。

wbra_request_header_length
(入力のみ)

HTTP 要求の場合は、この HTTP 要求の先頭の HTTP ヘッダーの長さ。この要求が HTTP 要求でない場合、値はゼロです。

wbra_request_header_ptr
(入力のみ)

HTTP 要求の場合は、この HTTP 要求に指定された先頭の HTTP ヘッダーを指すポインター。残りの HTTP ヘッダーは、要求バッファーの中でこの後に続きます。この要求が HTTP 要求でない場合、値は未定義です。

wbra_request_type
(入力のみ)

この要求が HTTP 要求の場合、値は WBRA_REQUEST_HTTP です。HTTP 要求でない場合、値は WBRA_REQUEST_NON_HTTP です。

wbra_resource_escaped_ptr
(入力のみ)

HTTP 要求の場合、アンエスケープされていない (つまり、まだエスケープされた形で存在している) 要求の HTTP ヘッダーのコピーを指すポインター。

wbra_resource_length
(入力のみ)

HTTP 要求の場合は、URL のパス構成要素の長さ (バイト)。この要求が HTTP 要求でない場合、値はゼロです。

wbra_resource_ptr
(入力のみ)

HTTP 要求の場合は、URL のパス構成要素のポインター。URIMAP 定義が含まれる場合、このポインターには PATH 属性の値が含まれます。この要求が HTTP 要求でない場合、値は未定義です。

wbra_response
(出力のみ)

アナライザー・プログラムによって生成される応答の値。 463 ページの『応答および理由コード』を参照してください。

wbra_server_ip_address
(入力のみ)

wbra_server_ipv6_address が指定されていない場合に HTTP サーバーの 2 進 IPv4 アドレスを指定する、フルワードの 32 ビット・フィールド。**wbra_server_address** では、IPv6 アドレスはサポートされていません。

wbra_server_address にゼロ以外の値が設定されている場合は、この値が使用されて **wbra_server_ipv6_address** の値は無視されます。そのため、IPv6 アドレッシングを使用する場合は、**wbra_server_address** の内容をクリアして、**wbra_server_ipv6_address** の値が使用されるようにする必要があります。

wbra_server_ipv6_address
(入力のみ)

IPv6 アドレッシングを使用する場合や、IPv4 アドレッシングを使用していて **wbra_server_address** が指定されていない場合に設定する必要がある、16 バイトのフィールド。このフィールドは IPv4 アドレスと IPv6 アドレスの両方をサポートしており、サーバーの 2 進 IPv6 アドレス、または IPv6 形式による IPv4 アドレスに設定されます。IP アドレス形式の詳細については、 6 ページの『IP アドレス』を参照してください。

wbra_server_program
(入出力)

長さ 8 のストリング。要求を処理する CICS アプリケーション・プログラムの名前。URIMAP 定義が含まれる場合、このストリングには PROGRAM 属性の値が含まれます。プログラム名は、**wbra_converter_program** で指定されるコンバーター・プログラムに渡されます。このフィールドを設定しなければ、渡される値はヌルです。プログラム名はこの場所かコンバーター・

プログラムで設定する必要があります。設定しないと、CICS アプリケーション・プログラムは呼び出されません。

wbra_unescape
(出力のみ)

- データをアンエスケープ形式で CICS アプリケーション・プログラムに渡すように指定するには、このパラメーターを WBRA_UNESCAPE_REQUIRED に設定します。
- データをエスケープ形式でアプリケーションに渡すように指定するには、このパラメーターを WBRA_UNESCAPE_NOT_REQUIRED に設定します。この値はデフォルトです。

  アナライザーによってデータがエスケープ形式に変換された場合も、このパラメーターを WBRA_UNESCAPE_NOT_REQUIRED に設定します。

wbra_urimap
(入力のみ)

要求の処理パスに含まれる URIMAP 定義に一致する名前。このフィールドが非ブランクの場合、CICS 提供のデフォルト・アナライザー DFHWBADX は、URL のパス構成要素を処理しないで戻します。

wbra_user_data_length
(入出力)

HTTP 要求のエンティティー・ボディの長さを表す 15 ビットの整数。要求が HTTP 要求でなければ、この値は要求の長さです。アナライザーに渡される長さには、エンティティー・ボディの終わりを定めることができる末尾の復帰改行 (CRLF) 文字が含まれます。アナライザーがエンティティー・ボディの長さを縮小した場合は、新たに冗長になったバイトがヌル文字 X'00' に置き換わります。変更後の値は、CICS ビジネス・ロジック・インターフェースにはフィールド **wbbl_user_data_length** で渡され、コンバーター・プログラムにはフィールド **decode_user_data_length** で渡されます。

wbra_user_data_ptr
(入力のみ)

HTTP 要求の場合は、この HTTP 要求のエンティティー・ボディを指すポインター。要求が HTTP 要求でない場合、これはその要求を指すポインターです。

wbra_user_token
(出力のみ)

decode_user_token としてコンバーター・プログラムに渡される 64 ビットのトークン。このフィールドを設定しないと、渡される値はヌルです。この要求用のコンバーター・プログラムがない場合、値は無視されます。

wbra_userid
(入出力)

長さ 8 のストリング。入力では、このストリングは (基本認証またはクライアントの証明書認証を使用して) クライアントが提供するユーザー ID です。また、URIMAP 定義が含まれる場合は、USERID 属性の値です (指定されている場合)。出力では、別名トランザクションで使用されるユーザー

ID です。これは、提供されるユーザー ID、またはアナライザー・プログラムが選択するユーザー ID です。このフィールドが出力でブランクまたはヌルの場合は、CICS のデフォルト・ユーザー ID が使用されます。

wbra_version
(入力のみ)

現在使用されているパラメーター・リストのバージョンを示すハーフワードの 2 進数。これは、定数値 **wbra_current_version** を使用して設定します。

応答および理由コード

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

アナライザー・プログラムは、次の表で示されている値のいずれかを **wbra_response** に返す必要があります。

シンボル値	数値	説明
URP_OK	0	別名トランザクションが開始されます。
URP_EXCEPTION	4	別名トランザクションは開始されません。Web 接続処理は、例外トレース・エントリー (トレース・ポイント 4510) を作成し、メッセージ (DFHWB0523) を発行します。 この要求が HTTP 要求である場合、エラー応答が Web クライアントに送信されます。デフォルトの状況コードは 400 (Bad Request (不正な要求)) であり、ユーザーによる置換が可能な Web エラー・プログラム DFHWBEP を使用してこのデフォルトを構成することができます。 要求が HTTP 要求でなければ、応答は送信されず、ソケットは閉じます。
URP_INVALID	8	別名トランザクションは開始されません。サーバー制御プログラムは例外トレース・エントリー (トレース・ポイント 4510) を書き込み、メッセージ (DFHWB0523) を出します。 この要求が HTTP 要求である場合、エラー応答が Web クライアントに送信されます。デフォルトの状況コードは 400 (Bad Request (不正な要求)) であり、ユーザーによる置換が可能な Web エラー・プログラム DFHWBEP を使用してこのデフォルトを構成することができます。 要求が HTTP 要求でなければ、応答は送信されず、ソケットは閉じます。

シンボル値	数値	説明
URP_DISASTER	12	別名トランザクションは開始されません。 CICS は、例外トレース・エントリー (トレース・ポイント 4510) を作成し、メッセージ (DFHWB0523) を発行します。 この要求が HTTP 要求である場合、エラー応答が Web クライアントに送信されます。デフォルトの状況コードは 400 (Bad Request (不正な要求)) であり、ユーザーによる置換が可能な Web エラー・プログラム DFHWBEP を使用してこのデフォルトを構成することができます。 要求が HTTP 要求でなければ、応答は送信されず、ソケットは閉じます。

その他の値を **wbra_response** に戻すと、サーバー制御プログラムは例外トレース・エントリー (トレース・ポイント 4510) を書き込み、メッセージ (DFHWB0523) を出します。この要求が HTTP 要求であれば、状況コード 400 (Bad Request (不正な要求)) を持つメッセージが Web クライアントに送信されます。要求が HTTP 要求でなければ、応答は送信されず、ソケットは閉じます。

エラーの場合にさらに情報を提供するために、 **wbra_reason** で 32 ビットの理由コードを与えることができます。 CICS Web サポートは、アナライザー・プログラムから返された理由コードに関してはなんの処理も行いませんが、ユーザーによる置換が可能な Web エラー・プログラム DFHWBEP では、このコードを使用してデフォルトの応答を変更する方法を決定できます。理由コードは、アナライザー・プログラムを起動した結果生成されたトレース・エントリーに出力され、また、メッセージ DFHWB0523 の中に含められます。

付録 F. コンバーター・プログラム

コンバーター・プログラムは主に、本来 Web で使用するようにはコーディングされていないアプリケーション・プログラムに対して使用するためのものです。 このプログラムは、複数のアプリケーション・プログラムからの出力を単一の HTTP メッセージに結合する際にも使用できます。

このタスクについて

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

コンバーター・プログラムは CICS が HTTP クライアントの場合には使用されません。また、Web サービス処理にも使用されません。このプログラムは、 CICS が HTTP サーバーの場合にのみ呼び出すことができます。CICS に対し、CICS Web サポート・プロセス中のコンバーター・プログラムが HTTP サーバーとしての役割をすることについては、31 ページの『HTTP サーバーとしての CICS に対する HTTP 要求および応答の処理』で説明されています。 71 ページの『第 5 章 HTTP サーバーとしての CICS に対して CICS Web サポートを使用可能にする』には、HTTP サーバーとしての CICS のアーキテクチャーを計画するのに役立つ情報が含まれています。

URIMAP リソースでコンバーター・プログラムを指定して、HTTP 要求の関連処理を行えます。アナライザー・プログラムを CICS Web サポート処理で使用する場合、アナライザー・プログラムはコンバーター・プログラムを呼び出すこともできます。コンバーター・プログラムは以下のような場合に役立ちます。

- 本来 Web で使用するようにはコーディングされていないアプリケーション・プログラムが、 COMMAREA の形式の入力を受け取る必要がある場合、または出力を HTTP 応答に変換する必要がある場合。 **EXEC CICS** WEB および **EXEC CICS** DOCUMENT アプリケーション・プログラミング・インターフェースを使用してコーディングされた Web 対応アプリケーション・プログラムの場合、この変換は必要ありません。コンバーター・プログラムを使用して、この変換または要求の内容に関する他の処理を実行できます。
- 順番に並んだ同一の要求データに関して複数のアプリケーション・プログラムを動作させる場合、および Web クライアントに対して単一の HTTP 応答を戻す場合。

コンバーター・プログラムを URIMAP 定義から直接呼び出す場合は、 (要求を処理するアプリケーション・プログラムの名前を指定する) URIMAP 定義の PROGRAM 属性をコンバーター・プログラムに渡すことができます。また、コンバーター・プログラムがこの属性をオーバーライドすることもできます。

コンバーター・プログラムは、要求についての詳細な情報を示すパラメーター・リストと一緒に、ストレージ・ブロックで Web クライアントの要求を受け取ります。コンバーター・プログラムは要求の内容を処理して、応答用のデータを提供するアプリケーション・プログラムに適した形式に変換し、 COMMAREA 内のアプ

リケーション・プログラムに渡します。この手順は、コンバーター・プログラムのデコード機能と呼ばれます。コンバーター・プログラムでアプリケーション・プログラムの COMMAREA を作成するためにデコード機能を使用していない場合は、HTTP 要求を受信するために使用された 32,767 バイトのバッファーがアプリケーション・プログラムに渡されます。

アプリケーション・プログラムはその結果をコンバーター・プログラムに返します。応答用のデータを生成するために複数のアプリケーション・プログラムが必要な場合、コンバーター・プログラムは追加のアプリケーション・プログラムを呼び出すことがあります。コンバーター・プログラムは、アプリケーション・プログラムから必要なすべてのデータを取得すると、 Web クライアントに送信する HTTP 応答を生成します。この手順は、コンバーター・プログラムのエンコード機能と呼ばれています。

コンバーター・プログラムは、アナライザー・プログラムと同じようには、TCPIPSERVICE 定義と関連付けられていません。任意のコンバーター・プログラムを使用して HTTP 要求を処理できますが、コンバーター・プログラムは、要求を受け取る CICS システムのローカル側になければなりません。デコード機能とエンコード機能の両方を使用するために、各要求に対して同一のコンバーター・プログラムが呼び出されます。

ユーザーによる置換が可能なプログラムはすべて、 CICS Web サポートが動作しているシステムのローカル側になければなりません。プログラムの自動インストールを使用しない場合は、アナライザー・プログラムおよびコンバーター・プログラムを含む CICS Web サポートが使用するすべてのユーザー置換可能プログラムを定義し、そのプログラム定義をインストールする必要があります。プログラムの自動インストールを使用する場合は、ユーザーによる置換が可能なプログラムを正しい属性でインストールする必要があります。

コンバーター・プログラムは、 CICS ビジネス・ロジック・インターフェースでも使用されます。CICS ビジネス・ロジック・インターフェースにおけるコンバーター・プログラムの役割は、 401 ページの『CICS ビジネス・ロジック・インターフェースを使用したプログラムの呼び出し』に記載されています。 CICS ビジネス・ロジック・インターフェースの呼び出し元は、コンバーター・プログラムが必要であるかどうか、およびどのコンバーター・プログラムを呼び出す必要があるかを決定します。

コンバーター・プログラムの作成

コンバーター・プログラムを作成するには、デコード機能およびエンコード機能を構成し、またコード・ページ変換について考慮する必要があります。

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

コンバーター・プログラムは、アセンブラー、C、COBOL、または PL/I で作成することができます。言語依存のヘッダー・ファイル、組み込みファイル、およびコピーブックについては、 473 ページの『コンバーター・プログラムに関する参照情報』で説明しています。

デコード機能: HTTP 要求の表示および処理

コンバーター・プログラムのデコード機能は、Web クライアントから HTTP 要求とその要求に関する詳しい情報を提供するパラメーター・リストを受け取ります。HTTP 要求は、パラメーター・リストによって指示されている 32K のストレージ・ブロック内のコンバーター・プログラムに渡されます。要求はすでに、メソッド、要求ヘッダー、およびボディなどの、いくつかの別々の要素に分割されています (要求がストレージ・ブロックに対して長すぎて収まらない場合、残りのデータはコンバーター・プログラムに渡されないことに、注意してください)。処理パスでアナライザー・プログラムが使用される場合、アナライザー・プログラムが要求の内容を変更している可能性があります。

CICS Web サポートのコンバーター・プログラムでは、必要であれば、**EXEC CICS** WEB API コマンドを使用して HTTP 要求を検査することができます。 WEB EXTRACT コマンドは、その要求に関する情報 (メソッドやバージョンなど) を取得します。 WEB READ HTTPHEADER コマンドまたは HTTPHEADER 表示コマンドは、HTTP ヘッダーの読み取りに使用できます。WEB RECEIVE コマンドは、要求のボディの受け取りに使用することができます。いずれかの **EXEC CICS** WEB API コマンドを使用する場合、これらのコマンドが Web クライアントの要求からオリジナルの情報を戻すこと、および、アナライザー・プログラムによって加えられた変更を参照するためには使用できないことに、注意してください。アナライザー・プログラムによる変更は、コンバーター・プログラムに直接渡されるパラメーター・リストおよびストレージ・ブロックでのみ見ることができます。

応答にデータを提供するユーザー作成のアプリケーション・プログラムの名前は、パラメーター・リストで提供します。パラメーター・リストは、URIMAP 定義から取得されるか、アナライザー・プログラムによって設定されます。アナライザー・プログラムを使用する場合、このアナライザー・プログラムが追加情報をユーザー・トークン内のコンバーター・プログラムに直接提供することができます。

Web クライアントの要求に関して取得した情報を使用して、コンバーター・プログラムのデコード機能は以下のことを行う必要があります。

- 要求の処理を継続するか、CICS がエラー応答を Web クライアント返すかを決定する。
- 要求を処理して応答を提供するユーザー作成アプリケーション・プログラムの名前を指定する。すでに URIMAP 定義またはアナライザー・プログラムから名前が入力されている場合、コンバーター・プログラムはその名前を受け入れるか、または変更することができます。
- ユーザー作成のアプリケーション・プログラムに渡される COMMAREA を構成する。 COMMAREA には、そのアプリケーション・プログラムが受け入れられる入力形式に変換された、Web クライアント要求からのデータが入っています。HTTP 要求が入っているストレージ・ブロックを再使用することもできますし、新しい COMMAREA を指定することもできます。

エンコード機能: 応答の作成

ユーザー作成のアプリケーション・プログラムが、コンバーター・プログラムによって提供された入力を使用してその処理を行った場合、コンバーター・プログラム

のエンコード機能がアプリケーション・プログラムから出力 COMMAREA を受け取ります。このデータを使用して、コンバーター・プログラムのエンコード機能は以下のことを行う必要があります。

- データを供するため給に複数のアプリケーション・プログラムが必要な場合は、残りのアプリケーション・プログラムを呼び出す。これを行うために、エンコード機能は、デコード機能を再度呼び出すためのループ応答を設定します。デコード機能はアプリケーション・プログラムの名前を変更して、COMMAREA 内の適切な入力を提供します。エンコード機能に再度出力が戻されます。詳しくは、472 ページの『コンバーター・プログラムからの複数のアプリケーション・プログラムの呼び出し』を参照してください。
- Web クライアントに送信される HTTP 応答を構成する。

CICS Web サポートのコンバーター・プログラムでは、**EXEC CICS** WEB API コマンドを使用して、Web クライアントへの応答を作成し、送信することができます。WEB WRITE HTTPHEADER は、応答の HTTP ヘッダーの作成に使用できます。WEB SEND コマンドは、応答のアセンブルと送信に使用することができます。

また、コンバーター・プログラムでは、ストレージのバッファー内に手動で HTTP 応答を構成し、それを CICS に戻して、CICS から Web クライアントに送信させることもできます。応答には HTTP バージョン、状況コード、状況テキスト、必要な HTTP ヘッダー、およびメッセージ・ボディを含める必要があります。応答の形式は、作業を行っている HTTP プロトコルの仕様 (HTTP/1.0 または HTTP/1.1) に準拠している必要があります。以下の方法で、HTTP 応答用にストレージのバッファーを取得することができます。

- GETMAIN コマンドを発行してストレージを取得する。
- 初期の処理ステージ (アナライザー・プログラムなど) で獲得したストレージを使用する。
- ユーザー作成のアプリケーション・プログラムによって戻された COMMAREA で応答を作成する。

応答に使用される領域の先頭のワードには、その領域の長さ (つまり、HTTP 応答の長さに 4 を加えたもの) が含まれていなければなりません。コンバーター・プログラムのエンコード機能を終了するときには、パラメーター・リスト内のデータ・ポインターがこのストレージのブロックを指示している必要があります。代わりに **EXEC CICS** WEB API コマンドを使用して応答を送信する場合、 CICS はこのポインターが示すストレージ・ブロックを無視し、破棄します。

どちらの方法を使用して HTTP 応答を構成するとしても、CICS は通常、HTTP/1.0 または HTTP/1.1 応答に適したいくつかの HTTP ヘッダーを挿入します (それらのヘッダーは、415 ページの『付録 B. CICS Web サポートにおける HTTP ヘッダーの解説』にリストされています)。コンバーター・プログラムによって作成された応答に、これらのヘッダーがすでに含まれている場合、CICS はそのヘッダーを置き換えません。CICS TS バージョン 3 以前に受信したであろう応答と同一の応答を Web クライアントが必要とするために、アナライザー・プログラムによって応答に、CICS TS バージョン 3 以前との互換性処理用のフラグが立てられている場合は、CICS Transaction Server for z/OS バージョン 3 リリース 1 以前にクライアントに送信されたヘッダーのみが使用されます。このフラグは、コンバーター・プログラムがストレージ・ブロック内で手動で応答を作成する場合にのみ機能します。

コンバーター・プログラムが **EXEC CICS** WEB API コマンドを使用して応答を送信する場合、このフラグは無効です。

コード・ページ変換

コンバーター・プログラムで **EXEC CICS** WEB API コマンドを使用して HTTP 要求を表示し、応答を作成する場合には、そのコマンドでの指定に従って、**EXEC CICS** WEB API コマンドを使用する他のすべてのプログラムと同じ方法でコード・ページ変換が行われます。

コンバーター・プログラムでは、32K のストレージ・ブロック内で渡される HTTP 要求のコード・ページ変換の設定を指定することはできません。コンバーター・プログラムが URIMAP 定義から直接呼び出される場合で、Web クライアント要求用のヘッダーが、メッセージ・ボディがテキストであることを示している場合、CICS は以下の標準設定を使用して、ストレージ・ブロックで提供されるメッセージ・ボディを変換します。

- 文字セットの場合、Web クライアントの要求に、CICS でサポートされている文字セットを指名する Content-Type ヘッダーがあれば、その文字セットが使用されます。Content-Type ヘッダーが Web クライアントの要求に付帯していないか、または指定した文字セットがサポートされていない場合、ISO-8859-1 文字セットが使用されます。
- ホスト・コード・ページの場合、CICS は、LOCALCCSID 初期設定パラメーターに指定されている、そのローカル CICS 領域のデフォルトのコード・ページを使用します。

これらの標準設定が適していない場合や、コード・ページ変換が不要の場合は、処理パスの中でアナライザー・プログラムを使用して、代わりのコード・ページ変換設定を指定するか、あるいは **EXEC CICS** WEB API コマンドを使用して要求を処理します。

コンバーター・プログラムが HTTP 応答をストレージのバッファー内に手動で構成する場合、CICS は 32K ストレージ・ブロックで渡された要求用に行われたコード・ページ変換をミラーリングします。応答は、アナライザー・プログラムに指定されている文字セットおよびホスト・コード・ページ設定を使用して、Web クライアントに送信されます。また、アナライザー・プログラムがない場合は、このトピックで説明している標準設定が使用されます。 アナライザー・プログラムが要求に対するコード・ページ変換を抑止した場合、応答ボディのコード・ページ変換は実行されません。この結果が適切でない場合は、代わりに **EXEC CICS** WEB API コマンドを使用して、応答を作成します。

CICS ビジネス・ロジック・インターフェース用のコンバーター・プログラム

CICS ビジネス・ロジック・インターフェースのあるコンバーター・プログラムを使用するときには制約事項があり、ユーザー作成のアプリケーション・プログラムに渡される COMMAREA の作成方法と、応答が含まれているストレージのバッファーの作成方法が、その制限による影響を受ける場合があります。詳細については、408 ページの『オフセット・モードとポインター・モード』を参照してください。

CICS ビジネス・ロジック・インターフェース用に作成されたコンバーター・プログラムで、**EXEC CICS** WEB API コマンドを使用しないでください。

コンバーター・プログラム・デコード機能の入力パラメーター

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

入力パラメーターは、パラメーター・リストでデコード機能に渡されます。

入力パラメーターには、以下のものがあります。

- Web クライアントの IP アドレス。
- Web クライアント要求の HTTP バージョンを指すポインター。
- 要求方式を指すポインター。
- URL のパス構成要素を指すポインター。
- 要求の HTTP ヘッダーを指すポインター。
- 要求メッセージのエンティティー・ボディを指すポインター。
- 要求のデータを提供する CICS アプリケーション・プログラムの名前 (アナライザー・プログラムによって設定されるか、URIMAP 定義で指定されています)。
- アナライザー・プログラムとコンバーター・プログラムで情報を共用するために使用される 8 バイトのユーザー・トークン。 450 ページの『アナライザー・プログラムとコンバーター・プログラムでのデータの共用』を参照してください。
- 各 HTTP 要求ごとにデコード機能が起動された回数を記録する反復カウンター。このカウンターは、デコード機能を初めて呼び出す前に 1 に設定され、後続の各呼び出しを行う前に増分されます。
- エンティティー・ボディのアドレスを FREEMAIN コマンドのターゲットにすることができるかどうかの指示。

アナライザー・プログラムは、パラメーター・リストをコンバーター・プログラムに渡す前に、以上のパラメーターの値を変更することがあります。Web クライアントからの元の要求を調べたい場合は、コンバーター・プログラムで **EXEC CICS** WEB API コマンドを使用します。

コンバーター・プログラム・デコード機能の出力パラメーター

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

デコード機能は、COMMAREA に出力として、応答コード、および COMMAREA のアドレスと長さを提供する必要があります。

- 応答コード (オプションで、理由コードによって修飾されます)。

  デコード機能が URP_OK の応答コードを戻した場合、処理は次のステップへ進みます。

  デコード機能が他の値を戻した場合は、HTTP 要求がリジェクトされ、エラー応答が出ます。この場合に CICS が行う応答の詳細は、 137 ページの『CICS Web サポートのデフォルトの状況コードおよびエラー応答』を参照してください。

- ユーザー作成アプリケーション・プログラムに渡される COMMAREA のアドレスと長さ。アプリケーション・プログラムが呼び出されなかった場合は、COMMAREA は未変更のままエンコード機能に渡されます。

デコード機能は次のような出力を生成する場合もあります。

- 要求のデータを生成するユーザー作成アプリケーション・プログラムの名前。アナライザー・プログラムが名前を提供した場合、コンバーター・プログラムはその名前を変更したり、アプリケーション・プログラムを呼び出さないことを指定したりできます。
- アナライザー・プログラムとコンバーター・プログラムで情報を共用するために使用される 8 バイトのユーザー・トークン。 450 ページの『アナライザー・プログラムとコンバーター・プログラムでのデータの共用』を参照してください。

コンバーター・プログラム・エンコード機能の入力パラメーター

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

入力パラメーターは、COMMAREA のエンコード機能に渡されます。

入力パラメーターには、以下のものがあります。

- ユーザー作成アプリケーション・プログラムによって戻される COMMAREA のアドレスと長さ。アプリケーション・プログラムが呼び出されなかった場合は、COMMAREA は、デコード機能から未変更のまま渡されます。
- アナライザー・プログラムとコンバーター・プログラムで情報を共用するために使用される 8 バイトのユーザー・トークン。 450 ページの『アナライザー・プログラムとコンバーター・プログラムでのデータの共用』を参照してください。
- 各 HTTP 要求ごとにエンコード機能が起動された回数を記録する反復カウンター。このカウンターは、エンコード機能を初めて呼び出す前に 1 に設定され、後続の各呼び出しを行う前に増分されます。

コンバーター・プログラム・エンコード機能の出力パラメーター

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

エンコード機能は、出力として、応答コード、ストレージ・バッファーのアドレス、HTTP 応答の長さ、および 8 バイトのユーザー・トークンを提供することができます。

- 応答コード (オプションで、理由コードによって修飾されます)。
  - エンコード機能が URP_OK の応答コードを戻すと、CICS は、送信のために **EXEC CICS** WEB API コマンドがすでに使用された場合をのぞいて、提供された HTTP 応答を Web クライアントに送信します。
  - エンコード機能が URP_OK_LOOP の応答コードを戻した場合、デコード機能による処理へ進みます。詳細については、 472 ページの『コンバーター・プログラムからの複数のアプリケーション・プログラムの呼び出し』を参照してください。

– エンコード機能が他の値を戻した場合は、HTTP 要求がリジェクトされ、エラー応答が出ます。この場合に CICS が行う応答の詳細は、 137 ページの『CICS Web サポートのデフォルトの状況コードおよびエラー応答』を参照してください。

- ストレージのバッファーに手動で HTTP 応答を作成した場合は、ストレージのバッファーのアドレス、および HTTP 応答の長さ。このバッファーの先頭のワードには、データの長さ (つまり、HTTP 応答の長さに 4 を加えたもの) が含まれていなければなりません。代わりに **EXEC CICS** WEB API コマンドを使用して応答を送信する場合、 CICS はこのポインターが示すストレージ・ブロックを無視し、破棄します。
- アナライザー・プログラムとコンバーター・プログラムで情報を共用するために使用される 8 バイトのユーザー・トークン。 450 ページの『アナライザー・プログラムとコンバーター・プログラムでのデータの共用』を参照してください。

コンバーター・プログラムからの複数のアプリケーション・プログラムの呼び出し

HTTP 要求に対する応答を作成するために必要なデータは、場合によっては、複数のユーザー作成アプリケーション・プログラムから出力されることがあります。

このタスクについて

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

このような場合には、必要に応じて以下の順序を繰り返すことができます。

- コンバーターのデコード機能。
- アプリケーション・プログラム。
- コンバーターのエンコード機能。

これを行うには、エンコード機能で応答を URP_OK_LOOP に設定します。 HTTP 応答が完成したら、応答を URP_OK に設定します。

デコード機能が 2 番目およびそれ以降に呼び出されるときは、以下の入力パラメーターは使用できません。

- HTTP バージョン。
- メソッド。
- URL のパス構成要素。
- 要求ヘッダー。
- エンティティー・ボディ。

ただし、WEB EXTRACT コマンドを使用すると、同じ情報を取得することができます。

パラメーター・リストのデータ・ポインターとユーザー・トークンを使用して、デコード機能とエンコード機能でデータを共用します。ストレージのバッファーに HTTP 応答を手動で作成する際に、エンコード機能の最終呼び出しを行ったら、データ・ポインター (**encode_data_ptr**) が有効な HTTP 応答のアドレスを指し示していることを確認する必要があります。 **EXEC CICS** WEB API コマンドを使用して応

答の生成および送信を行う場合は、このステージで確認を行います。この場合、CICS は、このポインターが示すストレージ・ブロックを無視し、破棄します。

コンバーター・プログラムに関する参照情報

このセクションには、コンバーター・プログラムの**デコード**機能に関する参照情報、およびコンバーター・プログラムの**エンコード**機能に関する参照情報が含まれています。

コンバーター・プログラムに渡された COMMAREA 内のパラメーターおよび定数の名前は、サポートされている各種のプログラム言語に適合する形式に変換され、CICS Web サポートの一部として提供されているファイルに定義されます。各種言語のファイルを以下の表にリストします。

言語	パラメーター・ファイル	定数ファイル
アセンブラー	DFHWBCDD	DFHWBUCD
C	DFHWBCDH	DFHWBUCH
COBOL	DFHWBCDO	DFHWBUCO
PL/I	DFHWBCDL	DFHWBUCL

これらのファイルは、COMMAREA 内にあるフィールドのデータ型に関する言語固有の情報を提供します。これらのファイルを使用する場合は、変換プログラムのステップで XOPTS(NOLINKAGE) を指定する必要があります。指定しないと、コンパイルが失敗します。

コンバーター・プログラム・デコード機能のパラメーター・リスト

アナライザー・プログラム、URIMAP 定義、または CICS ビジネス・ロジック・インターフェースの呼び出し元で要求用のコンバーター・プログラム名が指定された場合は、要求のデータを提供するユーザー作成アプリケーション・プログラムの前に**デコード**機能が呼び出されます。

パラメーターの要約

次の表では、パラメーターの名前は省略形で示されています。表中のそれぞれの名前をパラメーターの名前にするには、接頭部として **decode_** を付ける必要があります。

表 31. デコード機能のパラメーター

Input decode_	Inout decode_	Output decode_
client_address	**data_ptr**	**output_data_len**
client_ipv6_address	**input_data_len**	**reason**
client_address_string	**server_program**	**response**
client_ipv6_address_string	**user_token**	
eyecatcher		
entry_count		
function		
http_version_length		
http_version_ptr		
method_length		
method_ptr		
request_header_length		
request_header_ptr		
resource_length		
resource_ptr		
user_data_length		
user_data_ptr		
version		
volatile		

パラメーター

decode_client_address
(入力のみ)

decode_client_ipv6_address が指定されていない場合にクライアントの 2 進 IPv4 アドレスに設定する必要がある、フルワードの 32 ビット・フィールド。 **decode_client_address** では、IPv6 アドレスはサポートされていません。

decode_client_address にゼロ以外の値が設定されている場合は、この値が使用されて **decode_client_ipv6_address** の値は無視されます。そのため、IPv6 アドレッシングを使用する場合は、 **decode_client_address** の内容をクリアして、 **decode_client_ipv6_address** の値が使用されるようにする必要があります。

decode_client_ipv6_address
(入力のみ)

IPv6 アドレッシングを使用する場合や、IPv4 アドレッシングを使用していて **decode_client_address** が指定されていない場合に設定する必要がある、16 バイトのフィールド。このフィールドは IPv4 アドレスと IPv6 アドレスの両方をサポートしており、クライアントの 2 進 IPv6 アドレス、または IPv6 形式によるクライアントの IPv4 アドレスに設定されます。IP アドレス形式の詳細については、6 ページの『IP アドレス』を参照してください。

decode_client_address_string
(入力のみ)

クライアントのドット 10 進数形式の IPv4 アドレス。

decode_client_ipv6_address_string
(入力のみ)

IPv4 アドレスの場合はドット 10 進数形式、IPv6 アドレスの場合はコロン 16 進形式による、クライアントの IP アドレス。このフィールドの最大長は 39 バイトです。

decode_data_ptr
(入出力)

入力では、クライアントからの要求 (アナライザー・プログラムにより変更されている場合がある) を指すポインター、または、この呼び出しが**エンコード**・コンバーター機能からのループバックである場合は、**encode_data_ptr** の応答データを指すポインター。

出力では、ユーザー作成アプリケーション・プログラムに渡す COMMAREA を指すポインター。**decode_volatile** の値が **0** の場合は、このパラメーターを変更しないでください。

decode_entry_count
(入力のみ)

現行 Web 要求で**デコード**・コンバーターが実行された回数を示すカウント。

decode_eyecatcher
(入力のみ)

長さ 8 のストリング。**デコード**機能用の値は 『>decode 』です。

decode_function
(入力のみ)

デコード機能が呼び出されていることを示す、定数値 **URP_DECODE** に設定されるハーフワード・コード。

decode_http_version_length
(入力のみ)

クライアントがサポートする HTTP のバージョンを識別するストリングの長さ (バイト)。この要求が HTTP 要求でない場合、または **decode_entry_count** が 1 より大きければ、この値はゼロです。

decode_http_version_ptr
(入力のみ)

クライアントがサポートする HTTP のバージョンを識別するストリングを指すポインター。アナライザーが要求のこの部分を変更した場合、その変更はここで確認できます。 **decode_http_version_length** がゼロであれば、この値は未定義です。

decode_input_data_len
(入出力)

入力では、これは decode_data_ptr が指す要求データの長さ (バイト) です。

decode_method_length
(入力のみ)

HTTP 要求に指定されたメソッドの長さ (バイト)。この要求が HTTP 要求でない場合、または **decode_entry_count** が 1 より大きければ、この値はゼロです。

decode_method_ptr
(入力のみ)

HTTP 要求に指定されたメソッドを指すポインター。アナライザーが要求のこの部分を変更した場合、その変更はここで確認できます。**decode_method_length** がゼロであれば、この値は未定義です。

decode_output_data_len
(出力のみ)

ポインター **decode_data_ptr** で示されるユーザー作成アプリケーション・プログラムに渡される COMMAREA の長さ (バイト)。この出力が設定されていない場合のデフォルト値は 32 KB です。

decode_reason
(出力のみ)

理由コード。 478 ページの『応答および理由コード』を参照してください。

decode_request_header_length
(入力のみ)

HTTP 要求の最初の HTTP ヘッダーの長さ。この要求が HTTP 要求でない場合、または **decode_entry_count** が 1 より大きければ、この値はゼロです。

decode_request_header_ptr
(入力のみ)

HTTP 要求の最初の HTTP ヘッダーを指すポインター。アナライザー・プログラムが要求のこの部分を変更した場合、その変更はここで確認できます。**decode_request_header_length** がゼロであれば、この値は未定義です。

decode_resource_length
(入力のみ)

HTTP 要求内の URL のパス構成要素の長さ (バイト)。この要求が HTTP 要求でない場合、または **decode_entry_count** が 1 より大きければ、この値はゼロです。

decode_resource_ptr
(入力のみ)

HTTP 要求内の URL のパス構成要素のポインター。アナライザー・プログラムが要求のこの部分を変更した場合、その変更はここで確認できます。**decode_resource_length** がゼロであれば、この値は未定義です。

decode_response
(出力のみ)

応答。 478 ページの『応答および理由コード』を参照してください。

decode_server_program
(入出力)

長さ 8 文字のストリング。入力では、アナライザーが **wbra_server_program** として提供する値、または CICS ビジネス・ロジック・インターフェースの呼び出し元が提供する値。出力では、要求を処理するユーザー作成アプリケーション・プログラムの名前。アプリケーション・プログラム名はこの場所かアナライザー・プログラムで設定する必要があります。設定しないと、アプリケーション・プログラムは呼び出されません。

decode_user_data_length
(入力のみ)

この HTTP 要求のエンティティー・ボディの長さ (バイト)。アナライザー・プログラムによってこの値が変更された場合、変更された値はここで確認できます。エンティティー・ボディが要求の中になければ、長さはゼロです。この要求が HTTP 要求でなければ、値はこの要求の長さです。**decode_entry_count** が 1 より大きければ、この値はゼロです。

decode_user_data_ptr
(入力のみ)

この HTTP 要求の任意のエンティティー・ボディを指すポインター。アナライザーが要求のこの部分を変更した場合、その変更はここで確認できます。エンティティー・ボディが要求の中になければ、ポインターはゼロです。要求が HTTP 要求でなければ、このポインターは **decode_data_ptr** と同じ値を持ちます。 **decode_entry_count** が 1 より大きければ、この値は未定義です。

decode_user_token
(入出力)

64 ビットのトークン。入力では、**wbra_user_token** としてアナライザーが与えるユーザー・トークン、あるいは CICS ビジネス・ロジック・インターフェースの呼び出し元が与えるユーザー・トークン。出力では、**encode_user_token** として**エンコード**機能に渡されるトークン。

decode_version
(入力)

現在使用されているパラメーター・リストのバージョンを示すハーフワードの 2 進数。これは、定数値 **decode_current_version** を使用して設定します。

decode_volatile
(入力)

decode_data_ptr として指定されているデータ域を置換できるかどうかを示す単一文字コード。設定可能な値は以下のとおりです。

0 データ域は別の COMMAREA の一部であるので、置換することはできません。

1 **decode_data_ptr** が指しているストレージは、解放して、別サイズの作業域と置換できます。

応答および理由コード

以下のいずれかの値を **decode_response** で戻す必要があります。

シンボル値	数値	説明
URP_OK	0	処理を続行します。CICS アプリケーション・プログラムが要求されている場合は、そのプログラムが実行されます。そうでいない場合は、コンバーター・プログラムの**エンコード**機能で処理が続行されます。

シンボル値	数値	説明
URP_EXCEPTION	4	実行するアクションは、以下の理由コードによって決まります。 • **CICS Web サポート** – 1 (URP_SECURITY_FAILURE)。CICS は、例外トレース・エントリー (トレース・ポイント 455A) を作成し、メッセージ (DFHWB0121) を発行します。 この要求が HTTP 要求であれば、状況コード 403 が Web クライアントに送信されます。要求が HTTP 要求でなければ、応答は送信されず、TCP/IP ソケットはクローズします。 – 2 (URP_CORRUPT_CLIENT_DATA)。CICS は、例外トレース・エントリー (トレース・ポイント 4559) を作成し、メッセージ (DFHWB0121) を発行します。この要求が HTTP 要求であれば、状況コード 400 が Web クライアントに送信されます。要求が HTTP 要求でなければ、応答は送信されず、TCP/IP ソケットはクローズします。 – その他の任意の値。CICS は、例外トレース・エントリー (トレース・ポイント 455B) を作成し、メッセージ (DFHWB0121) を発行します。この要求が HTTP 要求であれば、状況コード 501 が Web クライアントに送信されます。要求が HTTP 要求でなければ、応答は送信されず、TCP/IP ソケットはクローズします。 • **CICS ビジネス・ロジック・インターフェース** – 2 (URP_CORRUPT_CLIENT_DATA)。CICS ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 4556) を書き込み、メッセージ (DFHWB0120) を出し、応答 400 をその呼び出し元に戻します。 – その他の任意の値。CICS は、例外トレース・エントリー (トレース・ポイント 455B) を書き込み、メッセージ (DFHWB0121) を出し、応答 501 をその呼び出し元に戻します。 CICS アプリケーション・プログラムおよびコンバーター・プログラムの**エンコード**機能は実行されません。

シンボル値	数値	説明
URP_INVALID	8	CICS アプリケーション・プログラムおよびコンバーター・プログラムの**エンコード**機能は実行されません。 • **CICS Web サポート** – CICS は、例外トレース・エントリー (トレース・ポイント 455C) を作成し、メッセージ (DFHWB0121) を発行します。この要求が HTTP 要求であれば、状況コード 501 が Web クライアントに送信されます。要求が HTTP 要求でなければ、応答は送信されず、TCP/IP ソケットはクローズします。 • **CICS ビジネス・ロジック・インターフェース** – CICS は、例外トレース・エントリー (トレース・ポイント 455C) を書き込み、メッセージ (DFHWB0121) を出し、応答 501 をその呼び出し元に戻します。
URP_DISASTER	12	CICS アプリケーション・プログラムおよびデコーダーの**エンコード**機能は実行されません。 • **CICS Web サポート** – CICS は、例外トレース・エントリー (トレース・ポイント 455D) を作成し、メッセージ (DFHWB0121) を発行します。この要求が HTTP 要求であれば、状況コード 501 が Web クライアントに送信されます。要求が HTTP 要求でなければ、応答は送信されず、TCP/IP ソケットはクローズします。 • **CICS ビジネス・ロジック・インターフェース** – CICS は、例外トレース・エントリー (トレース・ポイント 455D) を書き込み、メッセージ (DFHWB0121) を出し、応答 501 をその呼び出し元に戻します。

シンボル値	数値	説明
その他の任意の値		CICS アプリケーション・プログラムおよびデコーダーの**エンコード**機能は実行されません。 • **CICS Web サポート** – CICS は、例外トレース・エントリー (トレース・ポイント 455E) を作成し、メッセージ (DFHWB0121) を発行します。この要求が HTTP 要求であれば、状況コード 500 が Web クライアントに送信されます。要求が HTTP 要求でなければ、応答は送信されず、TCP/IP ソケットはクローズします。 • • **CICS ビジネス・ロジック・インターフェース** – CICS は、例外トレース・エントリー (トレース・ポイント 455E) を書き込み、メッセージ (DFHWB0121) を出し、応答 501 をその呼び出し元に戻します。

エラーの場合にさらに情報を提供するために、 **decode_reason** に 32 ビットの理由コードを設定できます。CICS Web サポートも CICS ビジネス・ロジック・インターフェースも**デコード**機能によって戻された理由コードに対しては何の処置も行いませんが、前述の URP_EXCEPTION で示された理由コードに対しては処置を行います。この理由コードは、**デコード**機能の呼び出しの結果書き込まれるトレース・エントリーに含まれています。

コンバーター・プログラム・エンコード機能のパラメーター・リスト

パラメーターの要約

以下のテーブルには、パラメーターの名前が省略形で示してあります。テーブルの中のそれぞれの名前には、パラメーターの名前にするために接頭部として **encode_** を付ける必要があります。

表 32. エンコード機能のパラメーター

Input encode_	Inout encode_	Output encode_
eyecatcher **entry_count** **function** **input_data_len** **user_token**	**data_ptr**	**reason** **response**

機能

アナライザー・プログラムまたは CICS ビジネス・ロジック・インターフェースの呼び出し元が要求のコンバーター・プログラム名を指定した場合は、ユーザー作成アプリケーション・プログラムが終了した後に**エンコード**機能が呼び出されます。**エンコード**機能は、アプリケーション・プログラムによって戻された COMMAREA

内のデータを使用して応答を作成します。

パラメーター

encode_data_ptr
(入出力)

入力では、CICS アプリケーション・プログラムによって戻される COMMAREA を指すポインターです。アプリケーション・プログラムが呼び出されなかった場合、これは、コンバーター・プログラムの**デコード**機能によって作成された COMMAREA を指すポインターです。

出力では、コンバーター・プログラムが、CICS によって Web クライアントに送信される HTTP 応答を手動でストレージのバッファーに作成した場合は、これは、応答を含むバッファーを指すポインターです。ポインターは、有効な場所を指していなければなりません。指定しないと、予想外の結果が発生することがあります。バッファーは、ダブルワード位置合わせされている必要があります。最初の 4 バイトは、バッファーの長さを指定する 32 ビットの符号なしの数でなければなりません (COBOL では、PIC 9(8) COMP として指定します)。バッファーの残りの部分は応答です。

代わりに、コンバーター・プログラムが **EXEC CICS** WEB API コマンドを使用して応答を送信する場合、CICS はこのポインターが示すストレージ・ブロックを無視し、破棄します。このような場合、ポインターは CICS アプリケーション・プログラムによって戻される COMMAREA を指すポインターとして残される場合がありますが、設定値は重要ではありません。

コンバーターをオフセット・モードで呼び出した CICS ビジネス・ロジック・インターフェースから呼び出した場合は、このフィールドを出力として使用しないでください。

encode_entry_count
(入力のみ)

現行 Web 要求でコンバーター・プログラムの**エンコード**機能が入った回数を示すカウント。

encode_eyecatcher
(入力のみ)

長さ 8 のストリング。**エンコード**機能用の値は「`>encode `」です。

encode_function
(入力のみ)

エンコード機能が呼び出されていることを示す、定数値 **URP_ENCODE** に設定するハーフワード・コード・セット。

encode_input_data_len
(入力のみ)

デコード機能によって **decode_output_data_len** に指定される COMMAREA の長さ。

encode_reason
(出力のみ)

理由コード (483 ページの『応答および理由コード』を参照してください)。

encode_response
(出力のみ)

応答 (『応答および理由コード』を参照してください)。

encode_user_token
(入力のみ)

デコード機能によって **decode_user_token** として出力される 64 ビットのトークン。

encode_version
(入力)

パラメーター・リストのレイアウトが変更された場合に変更される、単一文字のパラメーター・リスト・バージョン ID。この値は、CICS TS 1.3 以前のバージョンのパラメーター・リストを示す場合は 2 進ゼロ (X'00')、CICS TS 1.3 以降のバージョンのパラメーター・リストを示す場合は文字ゼロ (X'F0') となります。

encode_volatile
(入力)

encode_data_ptr として指定されているデータ域を置換できるかどうかを示す単一文字コード。可能な値は次のとおりです。

0 データ域は別の COMMAREA の一部であるので、置換することはできません。

1 **encode_data_ptr** が指しているストレージは、解放して、別サイズの作業域と置換できます。

応答および理由コード

以下のいずれかの値を **encode_response** で戻す必要があります。

シンボル値	数値	説明
URP_OK	0	**EXEC CICS** WEB API コマンドがまだ応答の送信に使用されていない場合は、**encode_data_ptr** によって指し示されたバッファー内の応答がクライアントに送信されます。
URP_DISASTER	12	**CICS Web サポート** • CICS は、例外トレース・エントリー (トレース・ポイント 455D) を作成し、メッセージ (DFHWB0122) を発行します。この要求が HTTP 要求であれば、状況コード 501 が Web クライアントに送信されます。要求が HTTP 要求でなければ、応答は送信されず、TCP/IP ソケットはクローズします。 **CICS ビジネス・ロジック・インターフェース** • CICS は、例外トレース・エントリー (トレース・ポイント 455D) を作成し、メッセージ (DFHWB0122) を発行して、応答 501 をその呼び出し元に戻します。

シンボル値	数値	説明
URP_OK_LOOP	16	CICS は**デコード**機能の先頭にループバックします。 **encode_user_token** に保管されている値が **decode_user_token** にコピーされ、**デコード**・コンバーター機能がそれを使用します。
その他の任意の値		**CICS Web サポート** • CICS は、例外トレース・エントリー (トレース・ポイント 455E) を作成し、メッセージ (DFHWB0122) を発行します。この要求が HTTP 要求であれば、状況コード 501 が Web クライアントに送信されます。要求が HTTP 要求でなければ、応答は送信されず、TCP/IP ソケットはクローズします。 **CICS ビジネス・ロジック・インターフェース** • CICS は、例外トレース・エントリー (トレース・ポイント 455E) を作成し、メッセージ (DFHWB0122) を発行して、応答 501 をその呼び出し元に戻します。

エラーの場合にさらに情報を提供するために、 **encode_reason** で 32 ビットの理由コードを与えることができます。 CICS Web サポートも CICS ビジネス・ロジック・インターフェースも**エンコード**機能によって戻された理由コードに対しては何の処置も行いません。理由コードは、**エンコード**機能の呼び出しの結果作成されるトレース・エントリーの中の出力です。

付録 G. DFHWBBLI、CICS ビジネス・ロジック・インターフェースに関する参照情報

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

ビジネス・ロジック・インターフェースを使用すれば、呼び出し元は、CICS アプリケーション・プログラムの前後にどの表示論理を実行するかを指定することができます。これには、次の 2 つの操作モードがあります。

- ポインター・モード。**デコード**機能の入力データは、ビジネス・ロジック・インターフェースの COMMAREA とは別個に割り振られたストレージの中にあります。 COMMAREA には、**デコード**機能の入力データを指すポインター (**wbbl_data_ptr**) が入っています。ビジネス・ロジック・インターフェースの呼び出しが終了したとき、 からの出力は、ビジネス・ロジック・インターフェースの COMMAREA とは別個に割り振られたストレージ内にあり、COMMAREA には、**エンコード**機能からの出力を指すポインター (**wbbl_outdata_ptr**) が入っています。
- オフセット・モード。**デコード**機能の入力データは、ビジネス・ロジック・インターフェース用の COMMAREA の一部に含まれます。 COMMAREA には、**デコード**機能の入力データのオフセット (**wbbl_data_offset**) が入っています。ビジネス・ロジック・インターフェースの呼び出しを終了したとき、**エンコード**機能からの出力は、ビジネス・ロジック・インターフェースの COMMAREA の一部となっており、 COMMAREA には**エンコード**機能からの出力のオフセット (**wbbl_outdata_offset**) が入っています。

ビジネス・ロジック・インターフェースの呼び出し元は、**wbbl_mode** を使用して、使用する操作モードを指示します。

ビジネス・ロジック・インターフェースのコンバーターの作成方法については、466 ページの『コンバーター・プログラムの作成』を参照してください。

パラメーターの要約

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

パラメーターおよび定数の名前は、サポートされている各種プログラム言語に適合した形式に変換され、CICS Web サポートの一部として提供されているファイルで定義されています。これらのファイルは、COMMAREA 内にあるフィールドのデータ型に関する言語固有の情報を提供します。

各種言語用のファイルを以下に示します。

言語	ファイル
アセンブラー	DFHWBBLD
C	DFHWBBLH

言語	ファイル
COBOL	DFHWBBLO
PL/I	DFHWBBLL

表 33. ビジネス論理インターフェースのパラメーター

入力	出力
wbbl_client_address **wbbl_client_ipv6_address** **wbbl_client_address_length** **wbbl_client_ipv6_address_length** **wbbl_client_address_string** **wbbl_client_ipv6_address_string** **wbbl_converter_program_name** **wbbl_eyecatcher** **wbbl_header_length** **wbbl_header_offset** **wbbl_http_version_length** **wbbl_http_version_offset** **wbbl_indata_length** **wbbl_indata_offset** **wbbl_indata_ptr** **wbbl_length** **wbbl_method_length** **wbbl_method_offset** **wbbl_mode** **wbbl_prolog_size** **wbbl_resource_length** **wbbl_resource_offset** **wbbl_server_address** **wbbl_server_ipv6_address** **wbbl_server_program_name** **wbbl_ssl_keysize** **wbbl_status_size** **wbbl_user_token** **wbbl_user_data_length** **wbbl_vector_size** **wbbl_version**	**wbbl_outdata_length** **wbbl_outdata_offset** **wbbl_outdata_ptr**

旧バージョンの CICS ビジネス・ロジック・インターフェース (DFHWBA1) を使用するように作成されたプログラムは、DHWBBLI を呼び出す互換性インターフェースを介してサポートされます。

ビジネス・ロジック・インターフェース DFHWBBLI 用のパラメーター

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

ビジネス・ロジック・インターフェース用の入出力パラメーターの名前と説明を以下にリストします。名前には接頭部 **wbbl_** が付いています。

入力を COMMAREA に挿入する前に、連絡域をクリアして 2 進ゼロにする必要があります。

wbbl_eyecatcher
(入力のみ)

ストリング **>DFHWBBLIPARMS** に設定する必要がある 14 文字のフィールド。

wbbl_client_address
(入力のみ)

wbbl_client_ipv6_address が指定されていない場合にクライアントの 2 進 IPv4 アドレスに設定する必要がある、フルワードの 32 ビット・フィールド。**wbbl_client_address** では、IPv6 アドレスはサポートされていません。

wbbl_client_address にゼロ以外の値が設定されている場合は、この値が使用されて **wbbl_client_ipv6_address** の値は無視されます。そのため、IPv6 アドレッシングを使用する場合は、**wbbl_client_address** の内容をクリアして、**wbbl_client_ipv6_address** の値が使用されるようにする必要があります。

wbbl_client_ipv6_address
(入力のみ)

IPv6 アドレッシングを使用する場合や、IPv4 アドレッシングを使用していて **wbbl_client_address** が指定されていない場合に設定する必要がある、16 バイトのフィールド。このフィールドは IPv4 アドレスと IPv6 アドレスの両方をサポートしており、クライアントの 2 進 IPv6 アドレス、または IPv6 形式によるクライアントの IPv4 アドレスに設定されます。IP アドレス形式の詳細については、6 ページの『IP アドレス』を参照してください。

wbbl_client_address_length
(入力のみ)

wbbl_client_address_string の長さに設定する必要がある 1 バイトの 2 進フィールド。

wbbl_client_ipv6_address_length
(入力のみ)

wbbl_client_ipv6_address_string の長さに設定する必要がある 1 バイトの 2 進フィールド。

wbbl_client_address_string
(入力のみ)

右側に 2 進ゼロを埋め込んだ、**wbbl_client_address** のドット 10 進表記である最大 15 文字のストリング。すべての新規プログラムについて、**wbbl_client_address_string** を **wbbl_client_ipv6_address_string** の代わりに使用します。

wbbl_client_ipv6_address_string
(入力のみ)

右側に 2 進ゼロを埋め込んだ、**wbbl_client_ipv6_address** のコロン 16 進表記またはドット 10 進表記である最大 39 文字のストリング。

wbbl_converter_program_name
(入力のみ)

コンバーターのデコード機能およびエンコード機能用に使用するプログラムの 8 文字の名前。

wbbl_header_length
(入力のみ)

この要求に関連付けられている HTTP ヘッダーの長さを入れなければならないフルワードの 2 進数。

wbbl_header_offset
(入力のみ)

この要求に関連付けられている HTTP ヘッダーの、要求データの先頭からのオフセットを入れなければならないフルワード 2 進数。

wbbl_http_version_length
(入力のみ)

この要求を処理する場合に使用する HTTP プロトコルのバージョンの長さを入れなければならないフルワード 2 進数。

wbbl_http_version_offset
(入力のみ)

この要求を処理する場合に使用する HTTP プロトコルのバージョンのオフセットを入れなければならないフルワード 2 進数。

wbbl_indata_length
(入力のみ)

wbbl_indata_ptr または **wbbl_indata_offset** で位置付けられるデータの長さに設定しなければならないフルワード 2 進数。アナライザーによってデータ長の値が変更された場合は、ここで確認できます。要求が HTTP 要求ではない場合は、このフィールドを設定してはなりません。

wbbl_indata_offset
(入力のみ)

wbbl_mode が "O" または "D" の場合、これは、アプリケーションに渡される HTTP 要求データの、パラメーター・リストの先頭からのオフセットです。

wbbl_indata_ptr
(入力のみ)

wbbl_mode が "P" の場合、このフィールドは、アプリケーションに渡される HTTP 要求データのアドレスです。

wbbl_length
(入力のみ)

BLI パラメーター・リストの全長に設定しなければならないハーフワード 2 進数。

wbbl_method_length
(入力のみ)

この要求を処理する場合に使用する HTTP メソッドの長さを入れなければならないフルワード 2 進数。メソッドは、GET、POST、HEAD、PUT、DELETE、LINK、UNLINK、または REQUEUE のいずれかです。

wbbl_method_offset
(入力のみ)

この要求を処理するために使用する HTTP メソッドの、要求データの先頭からのオフセットを入れなければならないフルワード 2 進数。メソッドは、GET、POST、HEAD、PUT、DELETE、LINK、UNLINK、または REQUEUE のいずれかです。

wbbl_mode
(入力のみ)

wbbl_indata および **wbbl_outdata** のアドレッシング・モードを示す単一文字。これは、これらの値がポインターであることを示す場合は "P"、パラメーター・リストの先頭からのオフセットであることを示す場合は "O" に設定する必要があります。

wbbl_outdata_length
(入力のみ)

wbbl_outdata_ptr または **wbbl_outdata_offset** によって位置付けられる応答データの長さを DFHWBBLI が戻す、フルワード 2 進数フィールド。

wbbl_outdata_offset
(入力のみ)

wbbl_mode が "O" または "D" の場合、これは、アプリケーションからの応答データの、パラメーター・リストの先頭からのオフセットを DFHWBBLI が戻すフルワードです。このアドレスは、**wbbl_indata_offset** と必ずしも同一ではありません。

wbbl_outdata_ptr
(入力のみ)

wbbl_mode が "P" の場合、このフィールドは、アプリケーションからの応答データのアドレスを DFHWBBLI が戻すフルワード・アドレスです。このアドレスは、**wbbl_indata_ptr** と必ずしも同一ではありません。

wbbl_prolog_size
(入力のみ)

56 (つまり、**wbbl_prolog** 副構造体の長さ) に設定する必要があるハーフワード 2 進数。

wbbl_resource_length
(入力のみ)

要求されている URI リソース (つまり、URL の最初の / 文字で始まる、URL の非ネットワーク部分) の長さを入れなければならないフルワード 2 進数。

wbbl_resource_offset
(入力のみ)

要求されている URI リソース (つまり、URL の最初の / 文字で始まる、URL の非ネットワーク部分) の要求データの先頭からのオフセットを入れなければならないフルワード 2 進数。

wbbl_response
(入力のみ)

DFHWBBLI が応答コードを戻す場合のフルワード 2 進数フィールド。

wbbl_server_address
(入力のみ)

wbbl_server_ipv6_address が指定されていない場合にサーバーの 2 進 IPv4 アドレスに設定する必要がある、フルワードの 32 ビット・フィールド。**wbbl_server_address** では、IPv6 アドレスはサポートされていません。

wbbl_server_address にゼロ以外の値が設定されている場合、この値が使用されて **wbbl_server_ipv6_address** の値は無視されます。そのため、IPv6 アドレッシングを使用する場合は、**wbbl_server_address** の内容をクリアして、**wbbl_server_ipv6_address** の値が使用されるようにする必要があります。

wbbl_server_ipv6_address
(入力のみ)

IPv6 アドレッシングを使用する場合や、IPv4 アドレッシング使用していて **wbbl_server_address** が指定されていない場合に設定する必要がある、16 バイトのフィールド。このフィールドは IPv4 アドレスと IPv6 アドレスの両方をサポートしており、サーバーの 2 進 IPv6 アドレス、または IPv6 形式による IPv4 アドレスに設定されます。IP アドレス形式の詳細については、6 ページの『IP アドレス』を参照してください。

wbbl_server_program_name
(入力のみ)

要求を処理し、応答を生成するときに使用する CICS アプリケーション・プログラムの 8 文字の名前。

wbbl_ssl_keysize
(入力のみ)

セキュア・ソケット・レイヤーが使用されている場合、SSL ハンドシェーク時にネゴシエーションされる暗号鍵のサイズ。 SSL を使用しない場合は、ゼロが入ります。

wbbl_status_size
(入力のみ)

wbbl_status 副構造体の長さに設定する必要がある 1 バイト 2 進数フィールド。

wbbl_user_data_length
(入力のみ)

エンティティー・ボディの長さに設定しなければならないフルワード 2 進数。アナライザーによって長さの値が変更された場合は、ここで確認できます。要求が HTTP 要求ではない場合は、このフィールドを設定してはなりません。

wbbl_user_token
(入力のみ)

DFHWBBLI の呼び出し元がクライアントとの現在の会話状態を識別するデータを渡す場合に使用する 8 文字フィールド。通常は、URL の**照会ストリング**部分 (つまり、疑問符 (?) より後ろのデータ) の先頭の 8 文字に設定されます。 .

wbbl_vector_size
(入力のみ)

64 (すなわち、**wbbl_vector** 副構造体の長さ) に設定しなければならないハーフワード 2 進数。

wbbl_version
(入力のみ)

現在使用されている BLI パラメーター・リストのバージョンを示すハーフワード 2 進数。これは、定数値 **wbbl_current_version** を使用して設定します。

ビジネス・ロジック・インターフェース応答

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

wbbl_response では、次の値のうちの 1 つが戻されます。これらの値は、HTTP クライアントに送信される予定の HTTP 応答と一致します。

400 コンバーター機能の 1 つが URP_EXCEPTION 応答を理由 URP_CORRUPT_CLIENT_DATA と一緒に戻しました。ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 4556) を書き込み、メッセージ (DFHWB0120) を出します。

403 **wbbl_server_program_name** で指定されたプログラムへの LINK コマンドが、NOTAUTH 応答を受け取りました。ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 4556) を書き込み、メッセージ (DFHWB0120) を出します。

404 **wbbl_server_program_name** で指定されたプログラムへの LINK コマンドが、PGMIDERR 応答を受け取りました。ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 4556) を書き込み、メッセージ (DFHWB0120) を出します。

500 次のいずれかが起こりました。

- ビジネス・ロジック・インターフェースが異常終了を検出しました。異常終了したプログラムに応じたメッセージが出されます。
  wbbl_server_program_name で指定されたプログラムの場合、メッセージは DFHWB0125 です。コンバーターの**エンコード**機能の場合、メッセー

ジは DFHWB0126 です。コンバーターの**デコード**機能の場合、メッセージは DFHWB0127 です。その他のプログラムの場合、メッセージは DFHWB0128 です。どの場合にも、例外トレース・エントリー (トレース・ポイント 4557) が書き込まれます。

- **wbbl_server_program_name** で指定されたプログラムへの LINK コマンドが、INVREQ 応答、LENGERR 応答、または予期せぬ応答を受け取りました。ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 4556) を書き込み、メッセージ (DFHWB0120) を出します。

501 次のいずれかが起こりました。

- **デコード**機能が、応答 URP_EXCEPTION と未定義の理由コードを戻しました。ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 455B) を書き込み、メッセージ (DFHWB0121) を出します。
- **デコード**機能が応答 URP_INVALID を戻しました。ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 455C) を書き込み、メッセージ (DFHWB0121) を出します。
- **デコード**機能が応答 URP_DISASTER を戻しました。ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 455D) を書き込み、メッセージ (DFHWB0121) を出します。
- **デコード**機能が未定義の応答を戻しました。ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 455E) を書き込み、メッセージ (DFHWB0121) を出します。
- **エンコード**機能が、応答 URP_EXCEPTION と未定義の理由コードを戻しました。ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 455B) を書き込み、メッセージ (DFHWB0122) を出します。
- **エンコード**機能が応答 URP_INVALID を戻しました。ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 455C) を書き込み、メッセージ (DFHWB0122) を出します。
- **エンコード**機能が応答 URP_DISASTER を戻しました。ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 455D) を書き込み、メッセージ (DFHWB0122) を出します。
- **エンコード**機能が未定義の応答を戻しました。ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 455E) を書き込み、メッセージ (DFHWB0122) を出します。

503 次のいずれかが起こりました。

- **wbbl_server_program_name** で指定されたプログラムへの LINK コマンドが、TERMERR 応答を受け取りました。ビジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 4555) を書き込み、メッセージ (DFHWB0120) を出します。
- **wbbl_server_program_name** で指定されたプログラムへの LINK コマンドが、SYSIDERR 応答または ROLLEDBACK 応答を受け取りました。ビ

ジネス・ロジック・インターフェースは、例外トレース・エントリー (トレース・ポイント 4556) を書き込み、メッセージ (DFHWB0120) を出します。

付録 H. Web エラー・プログラムの DFHWBEP に関する参照情報

重要: このトピックには、プロダクト・センシティブ・プログラミング・インターフェースと関連ガイダンス情報が含まれています。

サポートされているプログラム言語に適合する形式に変換される、Web エラー・プログラムの DFHWBEP に渡されるパラメーター・リスト内のパラメーターおよび定数の名前を、以下のテーブルにリストします。

言語	パラメーター・ファイル
アセンブラー	DFHWBEPD
C	DFHWBEPH
COBOL	DFHWBEPO
PL/I	DFHWBEPL

パラメーター

DFHWBEP パラメーターはすべて入力専用です。ただし、入出力である **wbep_response_ptr** および **wbep_response_len** と、出力専用である **wbep_suppress_abend** および **wbep_close_conn** は除きます。

wbep_abend_code
(入力のみ)

この例外に関連付けられている 8 文字の異常終了コード。

wbep_activity
(入力のみ)

エラーが発生したときに進行していた処理のタイプ。0 はサーバー処理を示し、2 はパイプライン処理を示します。

wbep_analyzer_reason
(入力のみ)

起動された場合、アナライザー・プログラムによって戻される理由コード。

wbep_analyzer_response
(入力のみ)

起動された場合、アナライザー・プログラムによって戻される応答コード。

wbep_client_address
(入力のみ)

wbep_client_ipv6_address が指定されていない場合にクライアントの 2 進 IPv4 アドレスに設定する必要がある、15 文字のフィールド。
wbep_client_address では、IPv6 アドレスはサポートされていません。

wbep_client_address にゼロ以外の値が設定されている場合は、この値が使用されて **wbep_client_ipv6_address** の値は無視されます。そのため、IPv6

アドレッシングを使用する場合は、**wbep_client_address** の内容をクリアして、**wbep_client_ipv6_address** の値が使用されるようにする必要があります。

wbep_client_ipv6_address
(入力のみ)

クライアントのコロン 16 進 IPv6 アドレスまたはドット 10 進 IPv4 アドレス。このフィールドの最大長は 39 文字です。

IPv6 アドレッシングを使用する場合や、IPv4 アドレッシングを使用していて **wbep_client_address** が指定されていない場合は、このフィールドを設定する必要があります。このフィールドは IPv4 アドレスと IPv6 アドレスの両方をサポートしており、クライアントの 2 進 IPv6 アドレス、または IPv6 形式によるクライアントの IPv4 アドレスに設定されます。IP アドレス形式の詳細については、6 ページの『IP アドレス』を参照してください。

wbep_client_address_len
(入力のみ)

wbep_client_address に含まれているドット 10 進 IP アドレスの長さ。アドレスが IPv6 形式の場合、このフィールドにはゼロが含まれます。

wbep_client_ipv6_address_len
(入力のみ)

wbep_client_ipv6_address または **wbep_client_address** に含まれている IP アドレスの長さ。

wbep_close_conn
(出力のみ)

応答がクライアントに送信された後で接続が閉じられたかどうかを示す、1 文字のフィールド (Y または N)。デフォルト値は N で、接続が閉じられていないことを示します。

wbep_converter_program
(入力のみ)

コンバーター・プログラムの名前 (それを失敗要求に使用している場合)。

wbep_converter_reason
(入力のみ)

起動された場合、コンバーター・プログラムによって戻される理由コード。

wbep_converter_response
(入力のみ)

起動された場合、コンバーター・プログラムによって戻される応答コード。

wbep_error_code
(入力のみ)

検出したエラーを識別するエラー・コード。

wbep_eyecatcher
(入力のみ)

診断の手助けをする目印が入っている文字フィールド。DFHWBA はこれを >wbepca に設定してから、 Web エラー・プログラムを呼び出します。

wbep_failing_program
(入力のみ)

エラーが発生したプログラムの名前が含まれる 8 文字のフィールド。

wbep_http_response_code
(入力のみ)

このエラーで CICS が戻したデフォルトの HTTP 状況コード。

wbep_length
(入力のみ)

DFHWBEPC コピーブックの長さ。

wbep_message_len
(入力のみ)

wbep_message_ptr によってアドレス指定された CICS メッセージ・テキストの長さ。

wbep_message_number
(入力のみ)

このエラーに関連する CICS WB ドメイン・メッセージのフルワードの番号。

wbep_message_ptr
(入力のみ)

このエラーに関連する CICS メッセージ・テキストを指すポインター。

wbep_response_len
(入出力)

入力では、このフィールドはこのエラーに関するデフォルトの HTTP 応答のフルワードの長さです。CICS は、 HTTP 要求に関するデフォルトの応答のみを提供します。HTTP でない要求の場合、このフィールドはゼロです。出力では、このフィールドは、デフォルトの HTTP 応答の長さ、またはストレージの同一ブロック内の変更された応答、あるいはストレージの新規ブロック内の置換応答の長さです。

wbep_response_ptr
(入出力)

入力では、このエラーに関するデフォルトの HTTP 応答が含まれるストレージのブロックを指すポインターです。CICS は HTTP 要求に関するデフォルトの応答のみを提供します。デフォルトの応答は、状況表示行、HTTP ヘッダー、およびメッセージ・ボディを含む、完全な HTTP 応答です。出力では、このポインターはデフォルトの応答の元のバージョンまたは変更されたバージョンが含まれるストレージの同一ブロックか、置換応答が含まれるストレージの新規ブロックを指します。DFHWBEP が **EXEC CICS** WEB SEND コマンドを適切に使用して新規応答を作成し、 Web クライアントに

送信すると、CICS はストレージのブロック内の HTTP 応答を無視して破棄します。そうでない場合は、ストレージのブロック内の応答が Web クライアントに送信されます。

wbep_server_address
(入力のみ)

wbep_server_ipv6_address が指定されていない場合に設定する必要がある、サーバー (HTTP サーバーとしての CICS) の 15 文字の IPv4 アドレス。**wbep_server_address** では、IPv6 アドレスはサポートされていません。

wbep_server_address にゼロ以外の値が設定されている場合は、この値が使用されて **wbep_server_ipv6_address** の値は無視されます。そのため、IPv6 アドレッシングを使用する場合は、**wbep_server_address** の内容をクリアして、**wbep_server_ipv6_address** の値が使用されるようにする必要があります。

wbep_server_ipv6_address
(入力のみ)

IPv6 アドレッシングを使用する場合や、IPv4 アドレッシングを使用していて **wbep_server_address** が指定されていない場合に設定する必要がある、16 バイトのフィールド。このフィールドは IPv4 アドレスと IPv6 アドレスの両方をサポートしており、サーバー (HTTP サーバーとしての CICS) の 2 進 IPv6 アドレス、または IPv6 形式による IPv4 アドレスに設定されます。IP アドレス形式の詳細については、6 ページの『IP アドレス』を参照してください。

wbep_server_address_len
(入力のみ)

wbep_server_address に含まれているドット 10 進 IPv4 アドレスの長さ。アドレスが IPv6 形式の場合、このフィールドにはゼロが含まれます。

wbep_server_ipv6_address_len
(入力のみ)

wbep_server_ipv6_address または **wbep_server_address** に含まれている IP アドレスの長さ。

wbep_target_program
(入力のみ)

Web クライアントの要求を処理するように指定されたターゲットのユーザー作成アプリケーション・プログラム。

wbep_tcpipservice_name
(入力のみ)

要求が受信されたポートの TCPIPSERVICE 定義の名前。

wbep_version
(入力のみ)

現在使用されているパラメーター・リストのバージョンを示すハーフワードの 2 進数。これは、定数値 **wbep_current_version** を使用して設定します。

wbep_suppress_abend
(出力のみ)

オンに設定されているときに異常終了の AWBM を抑止する、1 ビットのフラグ。

付録 I. DFHWBCLI Web クライアント・インターフェース

DFHWBCLI は、CICS 提供のユーティリティー・プログラムです。このプログラムを **EXEC CICS LINK** 経由で呼び出すと、Web クライアント・サービスまたはアウトバウンド HTTP を提供できます。これは、アップグレード用に、CICS Transaction Server for z/OS, バージョン 4 リリース 2でサポートされています。

DFHWBCLI Web クライアント・インターフェースの機能は、互換性上の理由で保持されています。機能を拡張するために、DFHWBCLI インターフェースを使用していた HTTP クライアント・アプリケーションをアップグレードして、クライアント要求に **EXEC CICS** WEB API コマンドを使用することができます (SESSTOKEN オプションを使用)。注意すべき重要な相違点は、**EXEC CICS** WEB API では、プロキシー・サーバーの使用は WEB OPEN コマンド (XWBOPEN) でユーザー出口によって指定され、プロキシー・サーバーの URL は、そのユーザー出口によって提供される点です。 141 ページの『HTTP クライアントとしての CICS を介した HTTP 要求』では、HTTP クライアント要求がどのように行われるかについて説明します。

DFHWBCLI を使用するには、まずネーム・サーバーを使用するように CICS をセットアップする必要があります。 65 ページの『第 4 章 CICS Web サポートのコンポーネントの構成』を参照してください。

DFHWBCLI を使用するには、この DFHWBCLI を、以下のコピーブックに内容がマップされているパラメーター・リストを含む COMMAREA とリンクさせる必要があります。

- DFHWBCLD (アセンブラー用)
- DFHWBCLO (COBOL 用)
- DFHWBCLL (PL/I 用)
- DFHWBCLH (C 言語用)

パラメーターの意味は、次のとおりです。

WBCLI_VERSION_NO
このパラメーター・リストのバージョン番号を指定する、1 バイトの 2 進数。これは、シンボリック定数 WBCLI_VERSION_CURRENT で指定している値に設定する必要があります。

WBCLI_FUNCTION
DFHWBCLI で実行する関数を指定する、1 バイトの 2 進数。これは、以下の値のいずれかに設定する必要があります。

0 (WBCLI_FUNCTION_CONVERSE)
HTTP 要求をターゲット・サーバーに送信し、対応する応答を受信する

1 (WBCLI_FUNCTION_SEND)
HTTP 要求をターゲット・サーバーに送信し、応答を待たずに制御を戻す

2 (WBCLI_FUNCTION_RECEIVE)
直前の SEND 関数で送信された HTTP 要求に対する応答を待機し、受信する

3 (WBCLI_FUNCTION_INQUIRE_PROXY)
`INITPARM=(DFHWBCLI,'PROXY=http://....')` システム初期設定パラメーターで指定されたプロキシー・サーバーの名前を要求する

4 (WBCLI_FUNCTION_CLOSE)
直前の SEND 関数で確立された接続を、HTTP 応答を待たずにクローズする

WBCLI_METHOD
HTTP 要求で指定される HTTP メソッドを指定する、1 バイトの 2 進数。これは、以下の値のいずれかに設定する必要があります。

1 (WBCLI_METHOD_GET)

2 (WBCLI_METHOD_POST)

WBCLI_FLAGS
HTTP 要求およびその予期される応答に関連したオプションを指定するために使用する、1 バイトの 2 進数ビット・ストリング。このビット・ストリングのビットは、以下の値に設定できます。

1... .... (WBCLI_OFFSET_MODE)
パラメーター・リスト内のポインター値は、そのパラメーター・リストの先頭からのオフセット値として指定される。これは、そのようなポインターのすべてのターゲットが、COMMAREA 内に含まれていることを意味します。

.1.. .... (WBCLI_DOCUMENT)
HTTP 要求ボディが、DOCUMENT CREATE コマンドで作成され、WBCLI_REQUEST_DOCTOKEN の文書トークンで指定されている CICS 文書である。

..1. .... (WBCLI_USE_PROXY)
HTTP 要求は、WBCLI_PROXY_URL_PTR で指定されている URL を持つプロキシー・サーバーを経由して送信される。

...1 .... (WBCLI_SET_RESP_BUFFER)
CICS は、HTTP 応答ボディを格納するのに適切なサイズのバッファーを獲得し、そのアドレスを WBCLI_REQUEST_BODY_PTR に戻す。

注: このアドレスは、WBCLI_OFFSET_MODE 設定にかかわらず、オフセットには作成されません。

.... ..1. (WBCLI_NATIVE_REQUEST_BODY)
アプリケーションが、元のままの形式で HTTP 要求ボディを提供するため、CICS でこれを EBCDIC から ASCII に変換する必要がない。

....1 (WBCLI_NATIVE_RESPONSE_BODY)
アプリケーションが、元のままの形式で HTTP 応答ボディを処理するため、CICS でこれを ASCII から EBCDIC に変換する必要がない。

WBCLI_RESPONSE
以下の値のいずれかに設定し、関数の出力を示す、ハーフワード 2 進数。

0 (WBCLI_RESPONSE_OK)

4 (WBCLI_RESPONSE_EXCEPTION)

8 (WBCLI_RESPONSE_DISASTER)

WBCLI_REASON
以下の値のいずれかに設定し、応答コードを修飾する、ハーフワード 2 進数。

1 (WBCLI_REASON_INVALID_URL)
WBCLI_URL_PTR で位置指定された URL の形式が無効であるか、ホスト・ロケーションがネーム・サーバーで解決できない。

2 (WBCLI_REASON_INVALID_HEADER)
WBCLI_HEADER_PTR で位置指定されたリスト内にある HTTP ヘッダーのいずれかの形式が無効である。

3 (WBCLI_REASON_INVALID_DOCUMENT)
WBCLI_REQUEST_DOCTOKEN で指定された文書トークンで、有効な CICS 文書を検索していない。

4 (WBCLI_REASON_GETMAIN_ERROR)
DFHWBCLI がその内部作業域のいずれかのストレージを取得しようとしたときに、エラーが発生した。

5 (WBCLI_REASON_PROXY_ERROR)
WBCLI_PROXY_URL_PTR で位置指定されたプロキシー・サーバーが見つからないか、エラー応答を戻した。

6 (WBCLI_REASON_SOCKET_ERROR)
ソケット操作を実行したときに、予期しない応答が戻された。

7 (WBCLI_REASON_HTTP_ERROR)
サーバーにより予期しない HTTP 応答が戻された。

8 (WBCLI_REASON_TRANSLATE_ERROR)
CICS がホストのコード・ページとサーバーのコード・ページとの間のデータ変換を行っているときに、エラーが戻された。これは、必要な変換が CICS でサポートされていないために起こる場合があります。

9 (WBCLI_REASON_TRUNCATED)
WBCLI_RESPONSE_BODY_LEN で指定されたユーザー提供の応答バッファーの長さが、サーバーから戻された応答を格納するのに十分でない。このバッファー長を超えたデータは廃棄されます。

WBCLI_SESSION_TOKEN
HTTP サーバーとの間に確立した接続を表す、不透明型の 8 バイト 2 進数トークン。これは、SEND 関数によって設定され、RECEIVE 関数および CLOSE 関数で必要です。その他の関数では使用されません。

WBCLI_URL_PTR
宛先 HTTP サーバーの URL (Uniform Resource Locator) を格納する、EBCDIC 文字ストリングのアドレス。 URL は、完全修飾されている必要があります。つまり、「`http://`」または「`https://`」で始まらなければなりません。

WBCLI_URL_LEN
WBCLI_URL_PTR で位置指定された URL の長さを格納する、フルワード 2 進数。

WBCLI_PROXY_URL_PTR
ファイアウォールの外側のリモート・サイトへのアクセスに必要なプロキシー・サーバーの URL (Uniform Resource Locator) を格納する、EBCDIC 文字ストリングのアドレス。 URL は、完全修飾されている必要があります。つまり、「http://」で始まらなければなりません。プロキシーを使用するには、WBCLI_USE_PROXY フラグも設定してください。

WBCLI_PROXY_URL_LEN
WBCLI_PROXY_URL_PTR で位置指定された URL の長さを格納する、フルワード 2 進数。

WBCLI_HEADER_PTR
HTTP 要求とともに送信される HTTP ヘッダーのリストのアドレス。ヘッダーは、EBCDIC でエンコードし、以下の形式で記述する必要があります。

headername: *headervalue*§*headername*: *headervalue*§ ...

ここで、
headername
はヘッダー名です。
headervalue
はヘッダー値です。

上記で示されているように、ヘッダー名とヘッダー値を分離するために、コロン (:) とスペースを使用する必要があります。また、上記の「§」は、1 つ以上の次の区切り文字で置き換えてください。
復帰 (X'0D')
改行 (LF) (X'25')
改行 (NL) (X'15')
フィールド分離文字 (X'1E')

注: これらの区切り文字は、ヘッダーの送信時には使用されません。 CICS は、アーキテクチャー上、正しい HTTP 区切り文字を使用します。
リスト内のヘッダーは、必要な数だけコーディングすることができます。ただし、CICS が以下のヘッダーを提供しているため、これらを含めないようにしてください。
Host
User-Agent
Content-Length
Content-Type

リストの最後のヘッダーの後には、区切り文字を指定する必要はありません。

WBCLI_HEADER_LEN
WBCLI_HEADER_PTR で位置指定されたヘッダー・リストの長さを格納する、フルワード 2 進数。

WBCLI_REQUEST_DOCTOKEN
DOCUMENT CREATE コマンドで作成され、HTTP 要求ボディとして使用される CICS 文書を表す、16 バイトの 2 進数文書トークン。
WBCLI_DOCUMENT フラグを設定して、このトークンを使用していることを示さなければなりません。

WBCLI_REQUEST_BODY_PTR
HTTP 要求ボディの全内容を格納する、EBCDIC 文字ストリングのアドレス。このパラメーターは、WBCLI_DOCUMENT フラグが設定されていない 場合に使用されます。

WBCLI_REQUEST_BODY_LEN
WBCLI_REQUEST_BODY_PTR で位置指定された要求ボディの長さを格納する、フルワード 2 進数。

WBCLI_RESPONSE_BODY_PTR
DFHWBCLI がサーバーから HTTP 応答ボディを戻すときに使用するバッファーのアドレス。

- フラグ WBCLI_SET_RESP_BUFFER が設定されていない場合、このアドレスおよび WBCLI_RESPONSE_BODY_LEN は呼び出し元によって設定しなければなりません。このバッファーが、応答ボディを格納するのに十分な大きさでない場合、応答ボディは切り捨てられます。
- フラグ WBCLI_SET_RESP_BUFFER が設定されている場合、このアドレスおよび WBCLI_RESPONSE_BODY_LEN は無視されます。 CICS は、応答全体を格納するのに十分な大きさのバッファーを新しく取得し、そのアドレスをこのフィールドに戻します。このアドレスは、WBCLI_OFFSET_MODE フラグに設定されている値にかかわらず、オフセットには変換されません。

通常、CICS は、DFHWBCLI を呼び出すタスクが終了したときにこのアドレスでストレージを開放します。別の方法では、アプリケーション・プログラムの **EXEC CICS** FREEMAIN コマンドを発行することによって、より早くストレージを開放することができます。CICS がストレージ不足になるのを回避するために、DFHWBCLI が長期実行タスクで繰り返し呼び出される場合は、上記のようにすることが推奨されます。

WBCLI_RESPONSE_BODY_LEN
WBCLI_RESPONSE_BODY_PTR で位置指定された応答バッファーの長さを格納する、フルワード 2 進数。

- 入力時、WBCLI_SET_RESP_BUFFER が設定されていない場合は、このパラメーターを使用してユーザー提供バッファーの長さを指定します。
- 出力時、このパラメーターには、戻された応答ボディの実際の長さが格納されます。

WBCLI_MEDIATYPE
HTTP 本体の IANA メディア・タイプ (MIME タイプとも呼ばれる) を格納する、40 バイトの EBCDIC ブランク埋め込み文字ストリング。

- 入力時、このパラメーターを使用して HTTP 要求ボディのメディア・タイプを指定します。このメディア・タイプは、HTTP Content-Type ヘッダーで送信されます。
- 出力時、このパラメーターには、HTTP Content-Type ヘッダーで受信したときの HTTP 応答ボディのメディア・タイプが格納されます。

メディア・タイプは、POST メソッドを使用する送信要求 (WBCLI_FUNCTION_SEND と WBCLI_METHOD_POST の両方が設定される要求) で指定される必要があります。

WBCLI_CHARSET

HTTP 本体の IANA 文字セットを格納する、40 バイトの EBCDIC 文字ストリング。

- 入力時、WBCLI_NATIVE_REQUEST_BODY が設定されていない場合は、このパラメーターを使用して、CICS で HTTP 要求ボディを変換する文字セットの名前を指定します。指定した文字セットは、HTTP Content-Type ヘッダー内のメディア・タイプを修飾するのに使用されます。値を指定しない場合は、デフォルトの `iso-8859-1` が想定されます (WBCLI_MEDIATYPE に値 TEXT が含まれている場合のみ)。
- 出力時、このパラメーターには、HTTP Content-Type ヘッダーで受信したときの HTTP 応答ボディの文字セットが格納されます。この文字セットは、HTTP 応答ボディを変換するのに使用されます (ただし、WBCLI_NATIVE_RESPONSE_BODY が設定されている場合を除きます)。

WBCLI_HOST_CODEPAGE

アプリケーションで使用する EBCDIC コード・ページ名を格納する、10 文字の EBCDIC ブランク埋め込み文字ストリング。これは、WBCLI_CHARSET と組み合わせて使用し、HTTP 文書本体でどのような変換が行われるかを判別します (ただし、変換が、WBCLI_NATIVE_REQUEST_BODY フラグまたは WBCLI_NATIVE_RESPONSE_BODY フラグにより抑止されている場合を除きます)。このパラメーターを省略した場合、CICS はコード・ページ 037 を使用します。

WBCLI_HTTP_STATUS_CODE

HTTP 状況コードが戻される、3 桁の数値 EBCDIC 文字ストリング。このコードは、HTTP 要求が成功したかどうかを示します。 200 状況コードは通常応答に使用され、2xx 範囲の他の状況コードも成功したかを示します。他の状況コードは、要求の実行を妨害するエラーが発生しており、要求を完了するために他の処理 (リダイレクト URL の実行など) をクライアントがする必要があることを示します。

付録 J. 状態管理サンプルの DFH$WBST および DFH$WBSR に関する参照情報

CICS Web サポートには、 2 つの状態管理サンプル・プログラム DFH$WBST および DFH$WBSR が提供されています。これらのサンプル・プログラムを使用すれば、トランザクションでデータを保管しておき、後でそれを同じトランザクションまたは別のトランザクションで検索することができます。

保管されたデータは、最初のトランザクションの状態管理プログラムが作成したトークンによってアクセスされます。最初のトランザクションは、データを検索するトランザクションにそのトークンを渡す必要があります。 DFH$WBST は GETMAIN コマンドを使用して、保管データ用のストレージを割り振ります。DFH$WBSR はデータを一時記憶域キュー (トークンごとに 1 つ) に保管するので、一時記憶域キューの適切な定義があれば、複数の CICS システムからデータにアクセスできます。このセクションの残りの部分は、どのプログラムにも当てはまります。

状態管理プログラムは、次のような操作を行います。
- 新規トークンを作成する。
- 情報を保管し、それをすでに作成済みのトークンと関連付ける。
- すでにトークンと関連付けられている情報を検索する。
- トークンと関連付けられた情報を破棄し、トークンを無効にする。

DFH$WBST は、有効期限が切れたトークンや情報も除去します。このプログラムを定期的に実行して、有効期限が切れた状態データをパージすることができます。
- 1 時間更新されていないすべての状態データをパージするには、プログラムをトランザクション CWBT として実行します。
- すべての状態データをパージするには、プログラムをトランザクション CWBP として実行します。

268 バイトの COMMAREA のレイアウトを以下の表に示します。必要な機能のための入力を設定する前に、COMMAREA を 2 進ゼロにクリアする必要があります。

表 34. 状態管理プログラムのパラメーター

オフセット	長さ	タイプ	値	注
0	4	C		目印
4	1	C	'C' 'R' 'S' 'D'	作成 (Create) 検索 (Retrieve) 保管 (Store) 破棄 (Destroy) これは機能コードです。機能コードはすべての呼び出しへの必須入力です。

表 34. 状態管理プログラムのパラメーター (続き)

オフセット	長さ	タイプ	値	注
5	1	X		戻りコード。戻りコードはすべての呼び出しからの出力です。
6	2	X		予約済み。
8	4	F		トークン。トークンは作成呼び出しからの出力であり、その他のすべての呼び出しへの入力です。
12	256	C		ユーザー・データ。ユーザー・データは作成・保管呼び出しへの入力であり、検索呼び出しからの出力です。ユーザー・データは、その他の呼び出しでは使用されません。

戻りコードを以下に示します。

0 要求された機能が実行されました。

- その機能が「作成」であれば、新規のトークンはオフセット 8 に入っています。
- その機能が「検索」であれば、オフセット 8 の入力トークンに関連付けられたエンティティー・ボディは、オフセット 12 のエンティティー・ボディ域に入っています。
- その機能が「保管」であれば、オフセット 12 の入力エンティティー・ボディは、オフセット 8 の入力トークンに関連付けられます。以前にそのトークンに関連付けられたエンティティー・ボディは上書きされます。
- その機能が「破棄」であれば、オフセット 8 の入力トークンに関連付けられたデータは廃棄されています。そのトークンは無効になりました。

2 オフセット 4 の機能コードは無効でした。COMMAREA をセットアップするプログラムを訂正してください。

3 機能は「作成」でしたが、GETMAIN コマンドによりエラー応答が出されました。

4 機能は「検索」、「保管」、または「破棄」でしたが、オフセット 8 の入力トークンが見つかりませんでした。入力トークンが「作成」によって戻されたトークンではないか、あるいはその有効期限が切れています。

5 内部データを一時記憶域キューに書き込む際に、 WRITEQ TS コマンドによりエラー応答が出されました。

7 ASKTIME コマンドによりエラー応答が出されました。

8 内部データを一時記憶域キューから読み取る際に、 READQ TS コマンドによりエラー応答が出されました。

9 タイムアウト処理時に、ASKTIME コマンドによりエラー応答が出されました。

11 機能は「作成」でしたが、WRITEQ TS コマンドによりエラー応答が出されました。この戻りコードが作成されるのは DFH$WBSR の場合だけです。

12 機能は「検索」でしたが、READQ TS コマンドによりエラー応答が出されました。この戻りコードが作成されるのは DFH$WBSR の場合だけです。

13 機能は「保管」でしたが、WRITEQ TS コマンドによりエラー応答が出されました。この戻りコードが作成されるのは DFH$WBSR の場合だけです。

14 機能は「破棄」でしたが、DELETEQ TS コマンドによりエラー応答が出されました。この戻りコードが作成されるのは DFH$WBSR の場合だけです。

付録 K. CICS Web サーバー・プラグイン

CICS Web サーバー・プラグインの機能は、互換性のために保持されています。CICS Web サービス、CICS Web サポート、または CICS Transaction Gateway を利用するソリューションに移行することが推奨されています。

提供されているこのプラグインは、CICS ビジネス・ロジック・インターフェースを使用することにより、IBM HTTP Server から外部 CICS インターフェース (EXCI) および CICS Web サポートへのパススルー・メカニズムを使用可能にします。このインターフェースで渡すことのできる最大データ量は 32 KB です。

IBM HTTP Server の構成

CICS Web サーバー・プラグインの機能は、互換性のために保持されています。新しいアプリケーションでは、CICS Transaction Gateway を使用されることをお勧めします。

このタスクについて

IBM HTTP Server が CICS ビジネス・ロジック・インターフェースを使用してサービスを提供する場合は、IBM HTTP Server 内の構成情報を変更する必要があります。構成のステートメントについて詳しくは、「*z/OS HTTP Server 計画、インストールと使用の手引き*」(SC88-9004) を参照してください。

以下の手順を使用することができます。

手順

1. CICS を次のように設定します。
   a. ISC=YES を指定して CICS 領域を初期設定する。
   b. RDO グループ DFHWEB をインストールする。
   c. EXCI の汎用接続を定義する。例えば、サンプル・グループ DFH$EXCI をインストールしてこれを行います。
   d. IRC がオープン状態であることを確認する。
2. CICSTS42.CICS.SDFHDLL1 ロード・ライブラリーおよび CICSTS42.CICS.SDFHEXCI を RACF プログラム制御に対して定義します。RACF プログラム制御は、そのライブラリーを含むボリュームのボリューム通し番号を記録し、別のボリュームを使用できないようにします。後でロード・ライブラリーまたは CICSTS42.CICS.SDFHEXCI ライブラリーを別のボリュームに移動した場合には、それを RACF プログラム制御に対して再定義する必要があります。
3. CICSTS42.CICS.SDFHDLL1 データ・セットおよび CICSTS42.CICS.SDFHEXCI ライブラリーを、IBM HTTP Server の JCL の STEPLIB 連結に追加します。SDFHEXCI および SDFHDLL1 は、サポートされているすべての CICS リリースに対する下位互換性があります。

4. IBM HTTP Server の httpd.conf ファイルが含まれるディレクトリーで、以下のコマンドを使用します。

   ```
   ln -e DFHWBAPI dfhwbapi.so
   ```

   このコマンドを STEPLIB 連結で使用した場合は、IBM HTTP Server のホーム・ディレクトリーから CICSTS42.CICS.SDFHDLL1 ライブラリーの DFHWBAPI メンバーの DLL dfhwbapi.so へのリンクが設定されます。

5. 1 つ以上のサービス・ディレクティブを httpd.conf ファイルに追加します。 サービス・ディレクティブは、エンド・ユーザーが入力した URL を、この要求を満たす CICS リソースにマップします。DFHWBAPI のサービス・ディレクティブは、次のような形式になっています。

   ```
   Service  /sourceurl/*  /home/dfhwbapi.so:DFHService/targeturl/*
   ```

   ここで、

   home IBM HTTP Server の httpd.conf ファイルが含まれるディレクトリーです。

   sourceurl
   DFHWBAPI で処理する着信 URL を選択する文字ストリングです。後続のアスタリスクは、着信 URL の残りの文字を表すワイルドカード文字です。 *sourceurl* は任意の形式にすることができるため、 *applid* や *transaction* などの詳細情報をエンド・ユーザーに隠すことができます。

   targeturl
   targeturl は、DFHWBAPI が、ユーザー要求を満たす CICS リソースを判別する場合に使用する文字ストリングです。ワイルドカードの置換後は、*targeturl* は次の形式になっていなければなりません。

   ```
   /applid/converter/tran/program/filename
   ```

   ここで、

   applid ターゲット CICS 領域のアプリケーション ID です。

   converter
   CICS 領域で使用するコンバーター・プログラムの名前、またはコンバーターを使用しない場合は CICS です。

   tran CICS 領域で実行するトランザクションです。トランザクションは EXCI 要求のターゲットであるため、トランザクションは、Web 別名トランザクション CWBA ではなく、 CSM3 などのミラー・トランザクションでなければなりません。トランザクションは、*sourceurl/** ではなく、 *targeturl/** を着信 URL として受信します。

   program
   CICS 領域で実行するプログラムの名前です。

   filename
   program が調べる任意の詳細情報です。

   DFHWBAPI を使用して 3270 アプリケーションにアクセスする場合は、CICS が、Web クライアントで表示する HTML フォームを生成します。CICS が HTML フォームに挿入する URL は、直前の要求で使用

した *targeturl* に一致します。このような状態を処理するには、上記のディレクティブに加え、以下の形式のサービス・ディレクティブも提供する必要があります。

```
Service  /targeturl/*  /home/dfhwbapi.so:DFHService
```

この場合、*targeturl* は未変更のまま DFHWBAPI に渡されます。

6. CICS 提供の一部のテンプレート定義から参照されるグラフィックス・ファイルを表示するには、以下のようにディレクティブを含めます。

   ```
   Service  /dfhwbimg/*  /home/dfhwbapi.so:DFHService/applid/DFHWBIMG/CSM3/*
   ```

   ここで、*applid* は、グラフィックス・ファイルを提供する CICS 領域を指定します (これは、ブリッジ動作を実行する CICS 領域とは異なる場合があります)。DFHWBIMG は、CICS Web ブリッジが使用する特殊目的の CICS 提供のコンバーター・プログラムです。

7. CICS Web サポートと CICS ビジネス・ロジック・インターフェースの両方を使用して CICS Web アプリケーションにアクセスする場合は、両方に同じホスト・コード・ページを指定する必要があります。 CICS のデフォルト・ホスト・コード・ページは IBM-037 ですが、IBM HTTP Server の場合は IBM-1047 です。

   - IBM HTTP Server のデフォルトのコード・ページを変更するには、DefaultFsCp 構成ディレクティブを使用します。 例えば、次のとおりです。

     ```
     DefaultFsCp IBM-1047
     ```

   - CICS で使用されるデフォルトのコード・ページを変更するには、DOCCODEPAGE システム初期設定パラメーターでそのコード・ページを指定します (例、DOCCODEPAGE=1047)。

     このデフォルトを使用して参照される文書と文書フラグメントは、指定されたコード・ページでエンコードされる必要があります。特に、BMS マップ定義から生成された文書テンプレートを使用している場合は、テンプレート・カスタマイズ・マクロを使用して、テンプレートが生成されるコード・ページを変更する必要があります。この指定を行うには、DFHMDX マクロの **CODEPAGE** パラメーターを使用します。次に例を示します。

     ```
     DFHMDX MAPSET=*,MAP=*,CODEPAGE=1047
     ```

     BMS マップ定義から生成されたテンプレートのカスタマイズの詳細については、213 ページの『第 14 章 BMS 定義からの HTML テンプレートの作成』を参照してください。

次のタスク

エスケープ・データおよび IBM HTTP Server

IBM HTTP Server および CICS ビジネス・ロジック・インターフェースを使用して同じ CICS アプリケーション・プログラムにアクセスする場合は、エスケープ・データが両者で一貫性をもって処理されていることを確認する必要があります。

IBM HTTP Server は、データをアンエスケープ形式で CICS アプリケーション・プログラムに渡します。したがって、CICS Web サポートにも同じ操作を行わせる必要があります。

詳細については、 451 ページの『アナライザー・プログラムからのエスケープ・データまたはアンエスケープ・データの選択』を参照してください。

IBM HTTP Server の処理の例

図 29 は、IBM HTTP Server に接続された Web クライアントからの要求を CICS Web サポートが処理する方法を示しています。

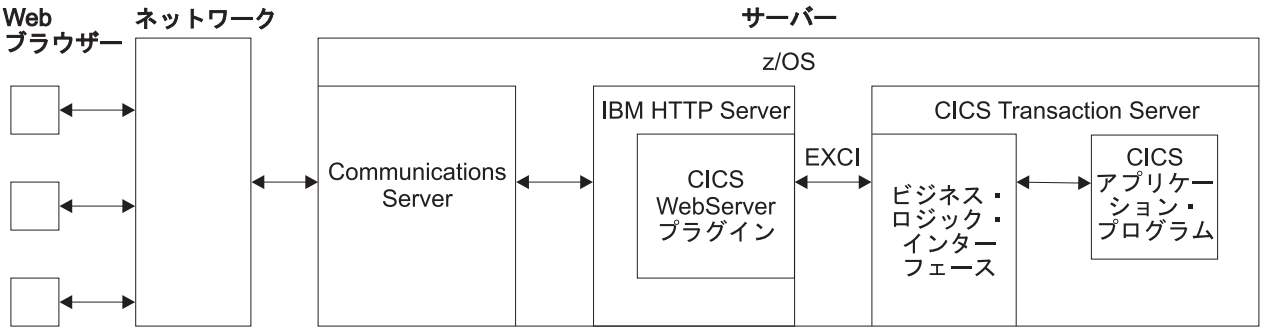


図 29. IBM HTTP Server からの要求の処理

1. Web クライアントが、ネットワークを介して Communications Server に渡す HTTP 要求を作成します。
2. Communications Server は要求を IBM HTTP Server にリレーします。
3. IBM HTTP Server が CICS Web サーバー・プラグインを呼び出します。
4. CICS Web サーバー・プラグインが CICS ビジネス・ロジック・インターフェースに対する要求を作成し、外部 CICS インターフェース (EXCI) を使用して CICS に渡します。
5. CICS ビジネス・ロジック・インターフェースが、要求された CICS アプリケーション・プログラムを起動し、出力を COMMAREA に戻します。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、本書に記載の製品、サービス、または機能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、および機能については、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、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービス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。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒242-8502
神奈川県大和市下鶴間1623番14号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

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

本書には、技術的に正確でない記述や誤植がある場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、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。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、 http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社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。

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。

UNIX は The Open Group の米国およびその他の国における登録商標です。

参考文献

CICS Transaction Server for z/OS の CICS ブック

一般

CICS Transaction Server for z/OS Program Directory、GI13-0565
CICS Transaction Server for z/OS リリース・ガイド、GA88-4308
CICS Transaction Server for z/OS CICS TS V3.1 からのアップグレード、GA88-4310
CICS Transaction Server for z/OS CICS TS V3.2 からのアップグレード、GA88-4311
CICS Transaction Server for z/OS CICS TS V4.1 からのアップグレード、GA88-4312
CICS Transaction Server for z/OS インストール・ガイド、GA88-4309

CICS へのアクセス

CICS インターネット・ガイド、SA88-4317
CICS Web サービス・ガイド、SA88-4315

管理

CICS System Definition Guide、SC34-7185
CICS Customization Guide、SC34-7161
CICS Resource Definition Guide、SC34-7181
CICS Operations and Utilities Guide、SC34-7213
CICS RACF Security Guide、SC34-7179
CICS Supplied Transactions、SC34-7184

プログラミング

CICS アプリケーション・プログラミング・ガイド、SA88-4313
CICS アプリケーション・プログラミング・リファレンス、SA88-4314
CICS System Programming Reference、SC34-7186
CICS Front End Programming Interface User's Guide、SC34-7169
CICS C++ OO Class Libraries、SC34-7162
CICS Distributed Transaction Programming Guide、SC34-7167
CICS Business Transaction Services、SC34-7160
CICS での Java アプリケーション、SA88-4321

診断

CICS Problem Determination Guide、GC34-7178
CICS パフォーマンス・ガイド、SA88-4318
CICS Messages and Codes Vol 1、GC34-7175
CICS Messages and Codes Vol 2、GC34-7176
CICS Diagnosis Reference、GC34-7166
CICS Recovery and Restart Guide、SC34-7180
CICS Data Areas、GC34-7163
CICS Trace Entries、SC34-7187

CICS Debugging Tools Interfaces Reference、GC34-7165

通信

CICS 相互通信ガイド、SA88-4316
CICS External Interfaces Guide、SC34-7168

データベース

CICS DB2 Guide、SC34-7164
CICS IMS Database Control Guide、SC34-7170
CICS Shared Data Tables Guide、SC34-7182

CICS Transaction Server for z/OS の CICSPlex SM ブック

一般

CICSPlex SM 概念および計画、SA88-4319
CICSPlex SM Web User Interface Guide、SC34-7214

管理

CICSPlex SM Administration、SC34-7193
CICSPlex SM Operations Views Reference、SC34-7202
CICSPlex SM Monitor Views Reference、SC34-7200
CICSPlex SM Managing Workloads、SC34-7199
CICSPlex SM Managing Resource Usage、SC34-7198
CICSPlex SM Managing Business Applications、SC34-7197

プログラミング

CICSPlex SM Application Programming Guide、SC34-7194
CICSPlex SM Application Programming Reference、SC34-7195

診断

CICSPlex SM Resource Tables Reference Vol 1、SC34-7204
CICSPlex SM Resource Tables Reference Vol 2、SC34-7205
CICSPlex SM Messages and Codes、GC34-7201
CICSPlex SM Problem Determination、GC34-7203

他の CICS 資料

以下の資料には CICS に関する詳しい情報が含まれますが、これらの資料は CICS Transaction Server for z/OS, バージョン 4 リリース 2 の一部としては提供されません。

Designing and Programming CICS Applications、SR23-9692
CICS Application Migration Aid Guide、SC33-0768
CICS ファミリー: API の構成、SC88-7261
CICS ファミリー クライアント・サーバー プログラミングの手引き、SC88-7429
CICS Family: Interproduct Communication、SC34-6853
CICS Family: Communicating from CICS on System/390、 SC34-6854
CICS Transaction Gateway (OS/390 版) 管理の手引き、SD88-7246

CICS Family: General Information、GC33-0155

CICS 4.1 Sample Applications Guide、SC33-1173

CICS/ESA 3.3 XRF Guide、SC33-0661

その他の IBM 資料

以下の資料には、関連する IBM 製品に関する情報が記載されています。

UNIX システム・サービス

「*z/OS UNIX システム・サービス ユーザーズ・ガイド*」、SA88-8640

「*z/OS UNIX システム・サービス コマンド解説書*」、SA88-8641

「*z/OS UNIX システム・サービス メッセージおよびコード*」、SA88-8645

「*z/OS UNIX システム・サービス プログラミング・ツール*」、SA88-8643

「*z/OS UNIX システム・サービス プログラミング: アセンブラー呼び出し可能サービス解説書*」、SA88-8642

「*z/OS UNIX システム・サービス ファイル・システム・インターフェース解説書*」、SA88-8646

「*z/OS REXX および z/OS UNIX システム・サービスの使い方*」、SA88-8644

「*z/OS V1R1 UNIX システム・サービス 並列処理環境: MPI プログラミングおよびサブルーチン解説*」、SA88-8650

z/OS Communications Server

「*z/OS Communications Server IP 構成解説書*」、SC88-8927

「*z/OS Communications Server IP 構成ガイド*」、SC88-8926

「*z/OS Communications Server IP マイグレーション*」、GC88-8924

「*z/OS Communications Server IP CICS ソケット・ガイド*」、SC88-9053

「*z/OS Communication Server IP アプリケーション・プログラミング・インターフェース・ガイド*」、SC88-8932

「*z/OS Communications Server: IP Programmer's Reference*」、SC31-8787

「*z/OS Communications Server IP ユーザーズ・ガイドとコマンド*」、SC88-8931

「*z/OS Communications Server クイック・リファレンス*」、SX88-8508

「*z/OS Communications Server: IP Diagnosis*」、GC31-8782

IBM Redbooks

IBM Redbooks は、IBM の International Technical Support Organization で開発され、公開されます。これらの資料は、お客様の実地に即したシナリオの統合、実装、および運用について、検討するものです。

以下の IBM Redbooks には、本書に取り上げた題材に関連する情報が記載されています。

「*Accessing CICS Business Applications from the World Wide Web*」、SG24-4547

「*How to Secure the Internet Connection Server for MVS/ESA*」、SG324-4803

「*Revealed! Architecting Web Access to CICS*」、SG24-5466

「*Workload Management for Web Access to CICS*」、SG24-6118

Web 情報

本書には、これらの URL が記載されていますが、現在もあるかどうかは保証されませんので、ご注意ください。

RFC (Request for Comments)

RFC 1945、「*Hypertext Transfer Protocol - HTTP/1.0*」(HTTP/1.0 仕様): http://www.ietf.org/rfc/rfc1945.txt

RFC 1867、「*Form-based File Upload in HTML*」: http://www.ietf.org/rfc/rfc1867.txt

RFC 2616、「Hypertext Transfer Protocol - HTTP/1.1」(HTTP/1.1 仕様): http://www.ietf.org/rfc/rfc2616.txt

RFC 2617、「HTTP Authentication: Basic and Digest Access Authentication」: http://www.ietf.org/rfc/rfc2617.txt

RFC 2396、「Uniform Resource Identifiers (URI): Generic Syntax」: http://www.ietf.org/rfc/rfc2396.txt

RFC 3023、「XML Media Types」: http://www.ietf.org/rfc/rfc3023.txt

RFC 文書は、Internet Society および Internet Engineering Task Force (IETF) によって公開されています。

IANA (Internet Assigned Numbers Authority)

IANA 登録済みメディア・タイプ: http://www.iana.org/assignments/media-types/

IANA 登録済み文字セット名: http://www.iana.org/assignments/character-sets

IANA 登録済みポート番号: http://www.iana.org/assignments/port-numbers

World Wide Web Consortium (W3C)

W3C ホーム・ページ: http://www.w3.org/

W3C HTTP 概説: http://www.w3.org/Protocols/Overview.html

W3C HTML ホーム・ページ: http://www.w3.org/MarkUp/

IBM IBM developerWorks: http://www.ibm.com/developerworks/

アクセシビリティー

アクセシビリティー機能は、運動障害または視覚障害など身体に障害を持つユーザーがソフトウェア・プロダクトを快適に使用できるようにサポートします。

CICS システムのセットアップ、実行、および保守に必要なほとんどの作業は、以下のいずれかの方法で行うことができます。

- CICS にログオンした 3270 エミュレーターを使用する
- TSO にログオンした 3270 エミュレーターを使用する
- 3270 エミュレーターを MVS システム・コンソールとして使用する

IBM パーソナル・コミュニケーションズは、身体障害のある方々のためのアクセシビリティー機能を持つ 3270 エミュレーションを提供します。 CICS システムで必要なアクセシビリティー機能を提供するためにこの製品を使用することができます。

Web Terminal Translation Application (DFHWBTTA) を使用して CICS 提供のトランザクションにアクセスする場合は、アクセシビリティー機能が Web ブラウザーから提供されます。

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]

[ワ行]

[数字]

A

B

C

D

E

F

G

H

I

X

Z

[特殊文字]

SA88-4317-01

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21