WebSphere® Application Server V4.0 for z/OS and OS/390

CORBA アプリケーションのアセンブル

SA88-8658-00
(英文原典：SA22-7848-00)

WebSphere® Application Server V4.0 for z/OS and OS/390

CORBA アプリケーションのアセンブル

SA88-8658-00
(英文原典：SA22-7848-00)

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、201ページの『付録D. 特記事項』に記載されている一般情報をお読みください。

本書は、WebSphere Application Server V4.0 for z/OS および OS/390 (5655-F31) に適用されます。また、新版で特に断りのない限り、これ以降のすべてのリリースとモディフィケーション・レベルにも適用されます。

WebSphere Application Server V4.0 for z/OS and OS/390 資料の最新バージョンは、下記の Web サイトにあります。
http://www.ibm.com/software/webservers/appserv/

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

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

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

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

(URL は、変更になる場合があります)

原　典： SA22–7848–00
WebSphere® Application Server V4.0 for z/OS and OS/390
Assembling CORBA Applications

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2001.6

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

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

目次

図

表

本書について

WebSphere Application Server V4.0 for z/OS and OS/390: CORBA アプリケーションのアセンブル, SA88-8658 では、WebSphere for z/OS サーバーで実行する共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (CORBA: Object Request Broker Architecture) アプリケーションの作成、アセンブル、および配置の方法を説明します。WebSphere for z/OS には、2 種類のアプリケーション・サーバーがあります。1 つは Java 2 エンタープライズ版 (J2EE) アプリケーション、もう 1 つは CORBA アプリケーション用のアプリケーション・サーバーです。CORBA アプリケーション用のサーバーは、マネージド・オブジェクト・フレームワーク (MOFW) サーバーと呼ばれています。J2EE アプリケーション、およびこれをインストールする WebSphere for z/OS サーバーについては、*WebSphere Application Server V4.0 for z/OS and OS/390: J2EE アプリケーションのアセンブル*, SA88-8654 を参照してください。

CORBA アプリケーションは、当座預金または顧客などの、特定のビジネス機能またはエンティティーを表す再利用可能な一部のコードである、C++ または Java ビジネス・オブジェクトで構成できます。WebSphere for z/OS サーバーは、これらのオブジェクトの高度な管理と、z/OS または OS/390 上のデータベース・システムおよびトランザクション・システムとの統合を可能にするアプリケーション環境を提供します。さまざまなプログラミング言語で作成されたオブジェクト指向のクライアント・アプリケーションは、WebSphere Application Server Component Broker 製品を実行する任意のオペレーティング・システム (たとえば、Windows NT や AIX) からこれらのサーバーに対する要求に対応することができます。

本書の説明を使用する前に、プログラマーは Component Broker プログラミングとコンポーネント・モデルについて精通していなければなりません。これらについては、*WebSphere Application Server エンタープライズ版 Component Broker: プログラミングの手引き* で詳しく説明しています。

本書の対象者

本書は、オブジェクト指向の分散アプリケーションが、z/OS または OS/390 で常駐または稼動する新規または既存のデータあるいはプログラム、または DB2、CICS、IMS などのサブシステムにアクセスできるようにする方法を習得したいプログラマーを対象にしています。このようなアクセスを使用可能にす

るために、プログラマーは WebSphere for z/OS サーバーで稼動するサーバー・アプリケーションを開発し、アセンブルし、配置します。これを実行するには、それぞれに異なるスキルが必要です。

- サーバー・アプリケーションの移植可能コンポーネント (すなわち、独立したプラットフォームであるコンポーネント・オブジェクト) を**開発する**には、以下の知識が必要です。
  - C++ または Java プログラム言語
  - *WebSphere Application Server エンタープライズ版 Component Broker: プログラミングの手引き* で詳しく説明している、Component Broker 共通プログラミング・モデルおよびコンポーネント・モデル
  - オブジェクト管理グループ (OMG) の共通オブジェクト・リクエスト・ブローカー・アーキテクチャー (CORBA: Common Object Request Broker Architecture)
  - オブジェクト・ビルダーおよび VisualAge コンポーネント開発 (以前は CBToolkit) を構成するその他のツールを含む、アプリケーション開発ツール。これらのツールの使用方法については、*WebSphere Application Server エンタープライズ版 Component Broker: アプリケーション開発ツールの手引き* を参照してください。

  クライアントおよびサーバー CORBA アプリケーション・プログラマーにとって必要な大量の情報は、一般的な Component Broker ブック、すなわちプラットフォームごとに独立した情報を収めたブックにあります。ただし、この資料には、z/OS または OS/390 上にサーバー・アプリケーションを配置することが分かっている場合に役立つ可能性がある、設計上の考慮事項および z/OS または OS/390 固有のガイドラインなどの情報の一部が含まれています。この情報は、17ページの『第2章 WebSphere for z/OS における CORBA アプリケーションの開発方法の学習』にあります。
- CORBA サーバー・アプリケーションの移植可能でプラットフォーム固有のコンポーネントを**アセンブルする**には、プログラマーにはワークステーションと z/OS または OS/390 アプリケーション開発の両方に関する専門知識が多少必要です。すなわち、これらの移植可能なコンポーネント (前述のリストで説明) を開発するために必要なプロセスとツールに関する知識と、UNIX システム・サーバー (USS) 環境での作業に多少精通していることが必要になります。USS 環境での作業には、階層ファイル・システム (HFS)、環境変数の設定、およびコードを完了するための `make` コマンドの使用が含まれます。

  USS 環境での作業に関する一般情報は、*z/OS UNIX システム・サービス ユーザーズ・ガイド*, SA88-8640 にあります。CORBA アプリケーションのア

センブリー・プロセスに関する情報は、47ページの『第3章 z/OS または OS/390 での CORBA アプリケーションのアセンブル』 にあります。

- z/OS または OS/390 上にサーバー・アプリケーションを**配置する**には、アプリケーションを配置するための知識と、サーバー・アプリケーションに必要な z/OS または OS/390 およびサブシステムの知識が必要です。z/OS または OS/390 システム・プログラマーとデータベース管理者は、アプリケーションの配置に必要な技術を有する可能性の最も高い人たちです。本書では、サーバー・アプリケーションのアセンブルおよび配置担当者を**アプリケーション・アセンブラー**と呼びます。これらの人たちは、配置するサーバー・アプリケーションについての詳細の他に、以下についての知識が必要です。
  - USS 環境 (前述のリストで説明) および z/OS または OS/390 コンポーネントおよびデータ・セットにアクセスするための TSO/E での作業方法。
  - アプリケーションに必要な z/OS または OS/390 サブシステム・リソースの設定方法。たとえば、これらのリソースにはデータベース、トランザクション・マネージャー、およびセキュリティー・プロダクトが含まれていることがあります。

    各分野における精通の度合いは、配置するサーバー・アプリケーションのタイプによって異なります。たとえば、CORBA アプリケーションが IMS トランザクションを介してアクセスしないで直接 DB2 データにアクセスする場合には、IMS についての知識は必要ありません。
  - WebSphere for z/OS アプリケーション・サーバーの定義と活動化の方法 (つまり、配置するアプリケーションのためのランタイム環境)。

  配置プロセスに関する情報は、85ページの『第4章 WebSphere for z/OS MOFW における CORBA アプリケーションの配置』にあります。

本書の主眼は、WebSphere for z/OS サーバーで稼動するサーバー・アプリケーションに置いてありますが、いくつかのトピックでは、z/OS または OS/390 とそのサブシステムのサービスを得るために、CORBA アプリケーションに対するメソッドを駆動するクライアント・アプリケーションにも注意を向けます。WebSphere for z/OS は、Windows NT および z/OS または OS/390 を含む、Java、C++、およびさまざまなプラットフォームで稼動するその他のクライアント・アプリケーションをサポートします。

クライアント・アプリケーション・プログラマーには、サーバー・アプリケーションを配置するプログラマーと同じワークステーション・スキルとリソースが必要です。また、各エリアで必要な専門知識は、クライアント・アプリケーションのタイプとそのランタイム環境によって異なります。ただし、クライア

ント・アプリケーション・プログラマーとサーバー・アプリケーション・プログラマーとでは、技術的に必要とするものが違います。すなわち、リソース・マネージャーまたはデータ・ストア (IMS や DB2 など) に関するプラットフォーム固有の詳細な知識は、クライアント・アプリケーション・プログラマーには必要ありませんが、サーバー・アプリケーション・プログラマーには必要です。クライアント・アプリケーションは、サーバー・アプリケーション・インターフェースのみを処理し、サーバー配置プラットフォームの内部作業は行いません。

ただし、z/OS または OS/390 で稼動するクライアント・アプリケーションを配置するには、プログラマーには、USS 環境に精通していることなど、多少の z/OS または OS/390 スキルが必要です。クライアント・アプリケーション・プログラマーに必要な大量の開発情報は、共通の Component Broker ブックにあります。このブックでは、z/OS または OS/390 でのクライアント・アプリケーションの準備と実行のための要件について説明します。

本書の編成

本書は、以下の章で編成されています。

- 1ページの『第1章 WebSphere for z/OS の CORBA アプリケーション入門』。WebSphere for z/OS 上で CORBA アプリケーションを使った作業の経験がほとんどないかまったくない場合には、ここから始めます。この章では、CORBA サーバー・アプリケーションのコード作成から WebSphere for z/OS サーバー上での実行に至るまで、CORBA サーバー・アプリケーションを開発および配置するプロセスについて説明します。

  この章は、次の 3 つの章で詳細に説明するトピックの概要となります。

  - 17ページの『第2章 WebSphere for z/OS における CORBA アプリケーションの開発方法の学習』
  - 47ページの『第3章 z/OS または OS/390 での CORBA アプリケーションのアセンブル』
  - 85ページの『第4章 WebSphere for z/OS MOFW における CORBA アプリケーションの配置』
- 17ページの『第2章 WebSphere for z/OS における CORBA アプリケーションの開発方法の学習』。z/OS または OS/390 上に配置する予定のサーバー・アプリケーション用のコードを設計および作成する場合に知っておく必要がある、z/OS または OS/390 固有のガイドラインを提供します。開発過程およびガイドラインの全体像を知りたい場合は、作成している CORBA アプリケーションのタイプによって、Component Broker アプリケーション開発に関する共通のブックを 1 冊以上参照する必要もあります。

- *WebSphere Application Server エンタープライズ版 Component Broker プログラミングの手引き*, SD88-7372 は、ビジネス・オブジェクト、データ・オブジェクト、ならびに、マネージド・オブジェクト・フレームワーク、IDL、および C++ CORBA プログラミングに関する情報を含むプログラミング・モデルについて説明しています。
- *WebSphere Application Server エンタープライズ版 Component Broker プログラミング解説書、第 1 巻および第 2 巻*, SD88-7376 および SD88-7377 は、Component Broker アプリケーション開発者が使用可能なアプリケーション・プログラミング・インターフェース (API) に関して説明しています。
- *WebSphere Application Server エンタープライズ版 Component Broker 上級プログラミングの手引き*, SD88-7373 は、CORBA オブジェクト・サービスおよび Component Broker オブジェクト・リクエスト・ブローカー (リモート・メソッド呼び出しおよび動的起動インターフェース (DII) のプロシージャーを含む) のための Component Broker インプリメンテーション、およびその他のトピックについて説明しています。
- *WebSphere Application Server エンタープライズ版 Component Broker アプリケーション開発ツールの手引き*, SD88-7378 は、CBToolkit で提供されるツールによる、Component Broker アプリケーションの作成方法およびテスト方法について、継承およびチーム開発などの共通の開発シナリオを中心に説明しています。
- *WebSphere Application Server エンタープライズ版 Component Broker 用語集*, SD88-7380 では、共通して使用される用語を定義しています。

- 47ページの『第3章 z/OS または OS/390 での CORBA アプリケーションのアセンブル』。ワークステーションで開発されたアプリケーション・ソース・コードを、WebSphere for z/OS サーバーで実行できる実行可能コードに変換するために必要なステップをリストします。このアセンブリー・ステップには、サーバー・アプリケーションを構成するコンポーネント・オブジェクトのパッケージ、z/OS または OS/390 でのコードのコンパイルが含まれます。
- 85ページの『第4章 WebSphere for z/OS MOFW における CORBA アプリケーションの配置』。サーバー・アプリケーションの WebSphere for z/OS サーバー (すなわち、ランタイム環境) の作成および活動化のためのガイドラインと手順を提供します。
- 103ページの『第5章 z/OS または OS/390 でのクライアント・アプリケーションの開発、アセンブル、および配置』。WebSphere for z/OS サーバー用に開発する CORBA アプリケーションを使用するクライアント・アプリケーションのセットアップのためのガイドラインを提供します。

- 115ページの『第6章 実動システムにおける CORBA アプリケーションに関する作業』。テスト・レベルではなく、実動レベルの WebSphere for z/OS サーバーへのサーバー・アプリケーションのインストールに関するバックグラウンド情報代替メソッドを提供します。
- 119ページの『第7章 CORBA アプリケーションの活動に関するデータの収集』。WebSphere for z/OS サーバーで稼動する CORBA アプリケーションに関する情報 (トレース・データなど) を収集するさまざまな方法について説明します。

関連情報の検索場所

WebSphere for z/OS のライブラリーにある資料を以下に示します。これらの資料は以下の Web サイトにあります。

`http://www.ibm.com/jp/software/websphere/appserv/`

- *WebSphere Application Server V4.0 for z/OS and OS/390: プログラム・ディレクトリー*, GI88-8549 では、WebSphere for z/OS のエレメントとインストールについて説明します。
- *WebSphere Application Server V4.0 for z/OS and OS/390: License Information*, LA22-7855 では、WebSphere for z/OS のライセンス情報について説明します。
- *WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652 では、WebSphere for z/OS の計画、インストール、およびカスタマイズの作業とガイドラインについて説明します。
- *WebSphere Application Server V4.0 for z/OS and OS/390: メッセージおよび診断*, GA88-8655 は、診断情報を提供し、WebSphere for z/OS に関連するメッセージとコードを説明します。
- *WebSphere Application Server V4.0 for z/OS and OS/390: 操作および管理*, SA88-8653 では、システムの運用および管理作業について説明します。
- *WebSphere Application Server V4.0 for z/OS and OS/390: J2EE アプリケーションのアセンブル*, SA88-8654 では、J2EE アプリケーションを開発、アセンブルし、WebSphere for z/OS J2EE サーバーにインストールする方法を説明します。この資料には、WebSphere Application Server for OS/390 の以前のリリースや、その他の WebSphere ファミリー・プラットフォームからのアプリケーションのマイグレーションに関する情報も記載されています。
- *WebSphere Application Server V4.0 for z/OS and OS/390: CORBA アプリケーションのアセンブル*, SA88-8658 では、CORBA アプリケーションを WebSphere for z/OS (MOFW) サーバーで開発、アセンブル、および配置する方法を説明します。

- *WebSphere Application Server V4.0 for z/OS and OS/390: システム管理ユーザー・インターフェース*, SA88-8656 では、システム管理ユーザー・インターフェースで提供されているシステム管理およびオペレーション作業について説明します。
- *WebSphere Application Server V4.0 for z/OS and OS/390: システム管理スクリプト API*, SA88-8657 では、WebSphere for z/OS システム管理スクリプト API 製品の機能を説明します。

また、その他の z/OS または OS/390 のエレメントおよび製品に関する情報も参照する必要が生じる場合があります。この情報はすべて以下のインターネット・サイトのリンクを通じてご利用いただけます。

```
http://www.ibm.com/servers/eserver/zseries/zos/
http://www.ibm.com/servers/s390/os390/
```

特に有用なその他の資料は以下のとおりです。

- *Getting Started with WebSphere Application Server*, SC09-4581 は、WebSphere for z/OS の概要を記載し、環境をセットアップするための要件を説明します。
- *WebSphere ビジネス構築のソリューション*, SD88-7362

第1章 WebSphere for z/OS の CORBA アプリケーション入門

この章には、WebSphere for z/OS 上で CORBA アプリケーションを開発および配置するプロセスの概要が記載されています。この概要では、CORBA アプリケーションに使用するプログラミング・ツールと配置ツールについて説明します。

次の表は、本章で説明されている作業、および関連するバックグラウンド情報または手順を示しています。

作業	関連するバックグラウンド情報または手順 (参照項目)
z/OS または OS/390 における CORBA アプリケーションの開発方法の学習	『CORBA アプリケーション開発のバックグラウンド』
WebSphere for z/OS サーバーにおける CORBA アプリケーションの配置方法の学習	9ページの『配置するサーバー・アプリケーションのバックグラウンド』
アプリケーション開発および配置環境の設定	12ページの『開発環境と配置環境のセットアップのバックグラウンド』

CORBA アプリケーション開発のバックグラウンド

開発過程の最初のステップは、CORBA アプリケーションを構成するコンポーネント・オブジェクトを作成することです。サーバー・アプリケーションで必要とするコンポーネント・オブジェクトのタイプは、いくつかの要因によって変わる可能性があるものです。たとえば、CICS または IMS リソースへのアクセスに必要なオブジェクトは、DB2 リソースへのアクセスに必要なオブジェクトとは異なります。コンポーネント・オブジェクト内での相違のため、以下のトピックで示すように、開発過程も多少変わります。

ただし、配置過程は、CORBA アプリケーションのタイプに関係なく同じです。開発したい CORBA アプリケーションに応じて、9ページの『配置するサーバー・アプリケーションのバックグラウンド』 と共に以下の開発トピックをいくつか見直してください。

- 2ページの『リレーショナル・データベースにアクセスする CORBA アプリケーションの開発』

- 6ページの『CICS または IMS リソースにアクセスする CORBA アプリケーションの開発』

Component Broker クライアント・プログラミングおよびコンポーネント・モデルに関する基本的な知識がある場合は、コンポーネント・オブジェクト (ビジネス・オブジェクト、マネージド・オブジェクト、データ・オブジェクトなど) についてはすでに理解しています。これらの用語に精通していない場合は、概要と詳細オブジェクトの説明について *Component Broker プログラミングの手引き* を参照してください。

リレーショナル・データベースにアクセスする CORBA アプリケーションの開発

DB2 などのリレーショナル・データベースにアクセスするアプリケーションを開発するには、まず、オブジェクト・ビルダーを使用して図1 に示すコンポーネントを定義します。

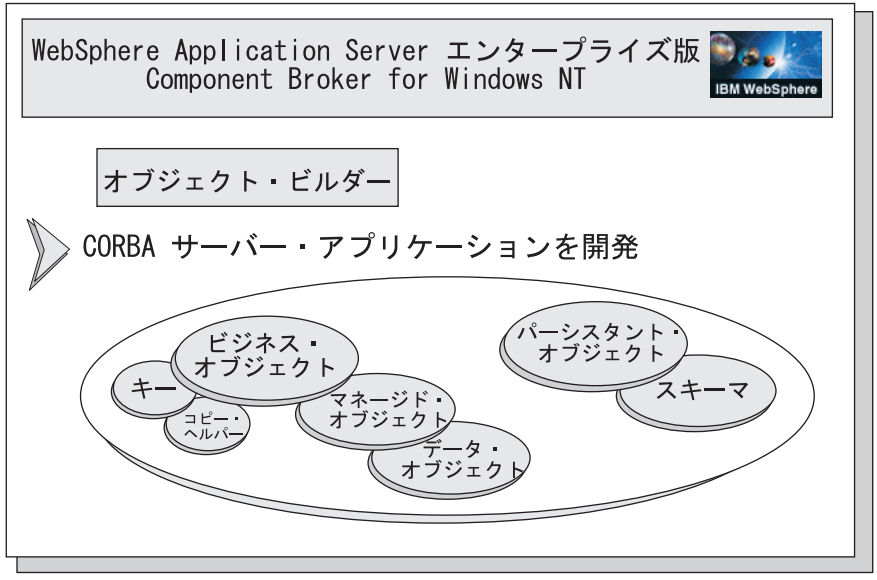

図 1. CORBA アプリケーションのコンポーネント・オブジェクトの開発

これらのコンポーネント・オブジェクト定義から、ソース・コードを生成し、CORBA サーバー・アプリケーションを使用するクライアント・アプリケーション、および CORBA アプリケーションを実行する WebSphere for z/OS サーバーに必要な動的ロード・ライブラリー (DLL) を定義するよう、オブジェクト・ビルダーに指示します。3ページの図2 は、プロセス内のこれらのステップ

を示しています。この時点で、オブジェクト・ビルダーは、後に z/OS または OS/390 で使用する make ファイルも生成し、実行可能コードを生成します。

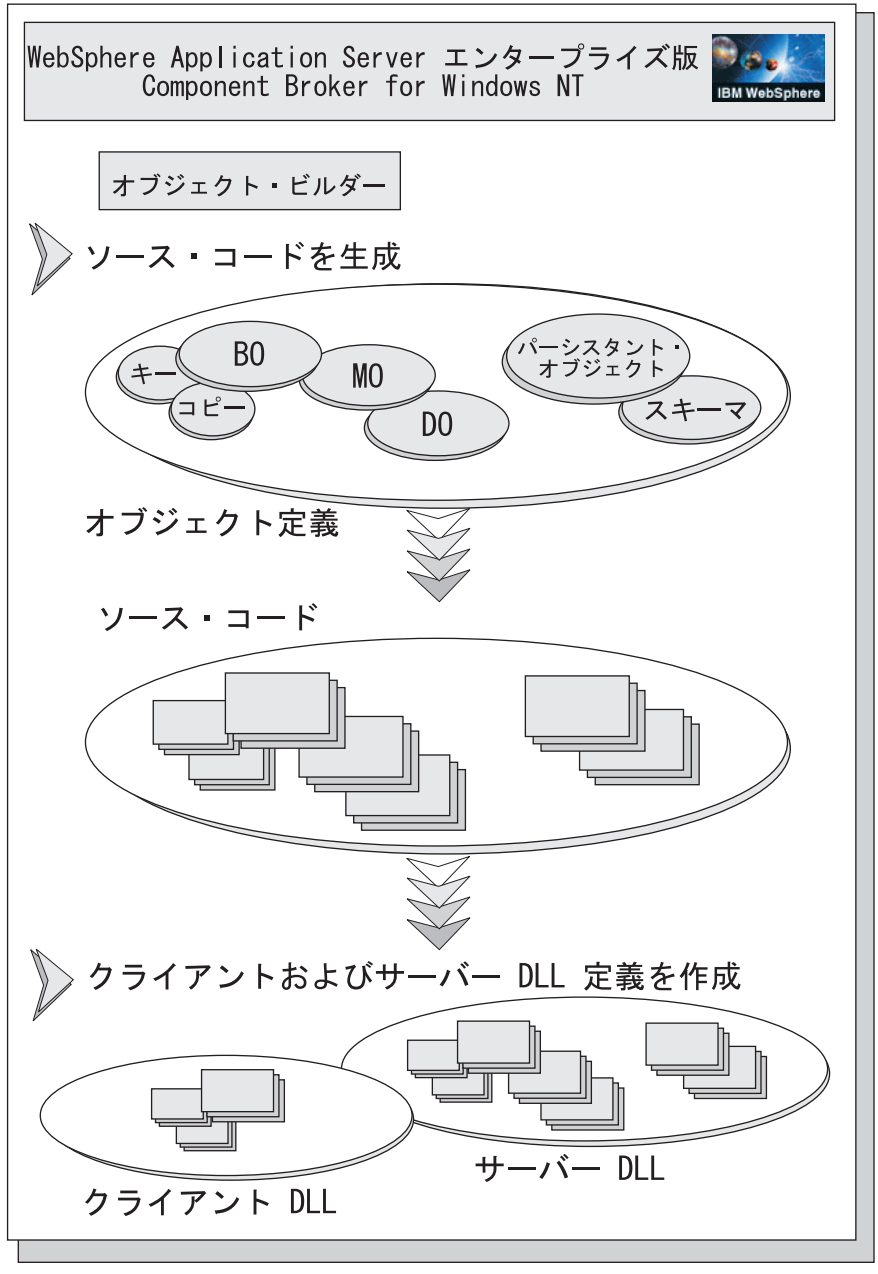

図 2. コンポーネント・オブジェクト用のソース・コードの生成

ソース・コード、クライアントおよびサーバーの DLL、および make ファイルの生成後は、オブジェクト・ビルダーからのもう 1 つの成果物であるメタデータのみが必要になります。このメタデータは、WebSphere for z/OS サーバーが CORBA アプリケーションの構造と識別、そのホームおよびコンテナーを理解するために必要です。5ページの図3 に示すように、CORBA アプリケーションをパッケージするステップを完了すると、オブジェクト・ビルダーはデータ定義言語でこのメタデータを生成します。

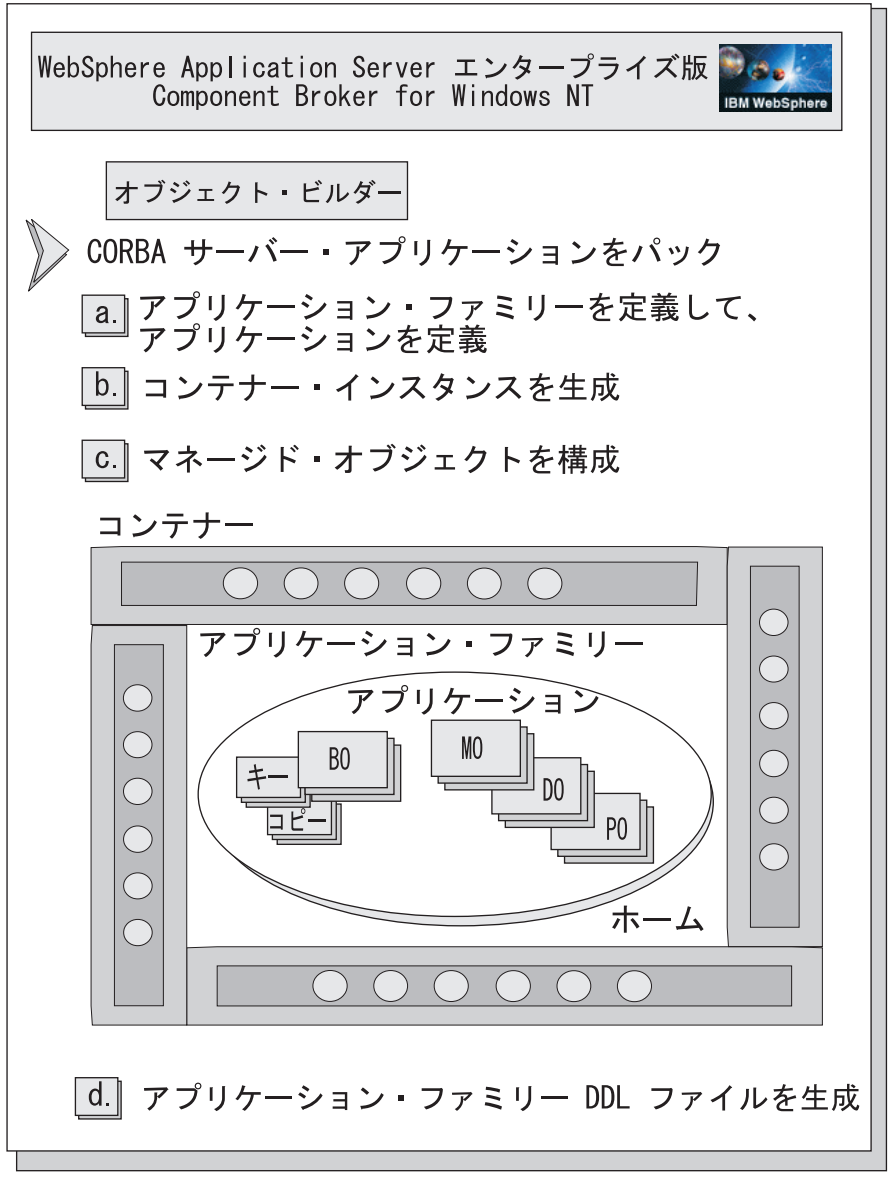

図 3. アプリケーション・ファミリーへのコンポーネント・オブジェクトのパッケージ

この時点で、開発過程で生成した成果物を WebSphere for z/OS がインストールされている z/OS または OS/390 システムに転送します。このシステムで、make ファイルを実行して実行可能コードを生成することができます。本書では、転送とコード生成を完了するために必要な詳細について説明します。コードを生成すると、9ページの『配置するサーバー・アプリケーションのバックグ

ラウンド』に示すように、WebSphere for z/OS アプリケーション・サーバーで CORBA アプリケーションを配置する準備ができます。

CICS または IMS リソースにアクセスする CORBA アプリケーションの開発

CICS または IMS を使用する CORBA アプリケーションを開発している場合は、開発過程の始まりは、2ページの『リレーショナル・データベースにアクセスする CORBA アプリケーションの開発』に示すものと多少異なります。

CICS サンプルおよび IMS サンプルのアプリケーションは、WebSphere for z/OS が提供する手続き型アプリケーション・アダプター (PAA) サポートを使用します。このようなサーバー・アプリケーションの場合は、まず、VisualAge for Java を使用して手続き型アダプター・オブジェクト (PAO) を作成する必要があります。VisualAge for Java を介して、このオブジェクトとそれに関連付けられたクラスを PA bean にパッケージします。次に、PA bean をオブジェクト・ビルダーにインポートします。オブジェクト・ビルダーは、PAO とそのクラスを 2 つのコンポーネント・オブジェクト (パーシスタント・オブジェクトとそのスキーマ) に変換します。7ページの図4 は、この過程を示しています。

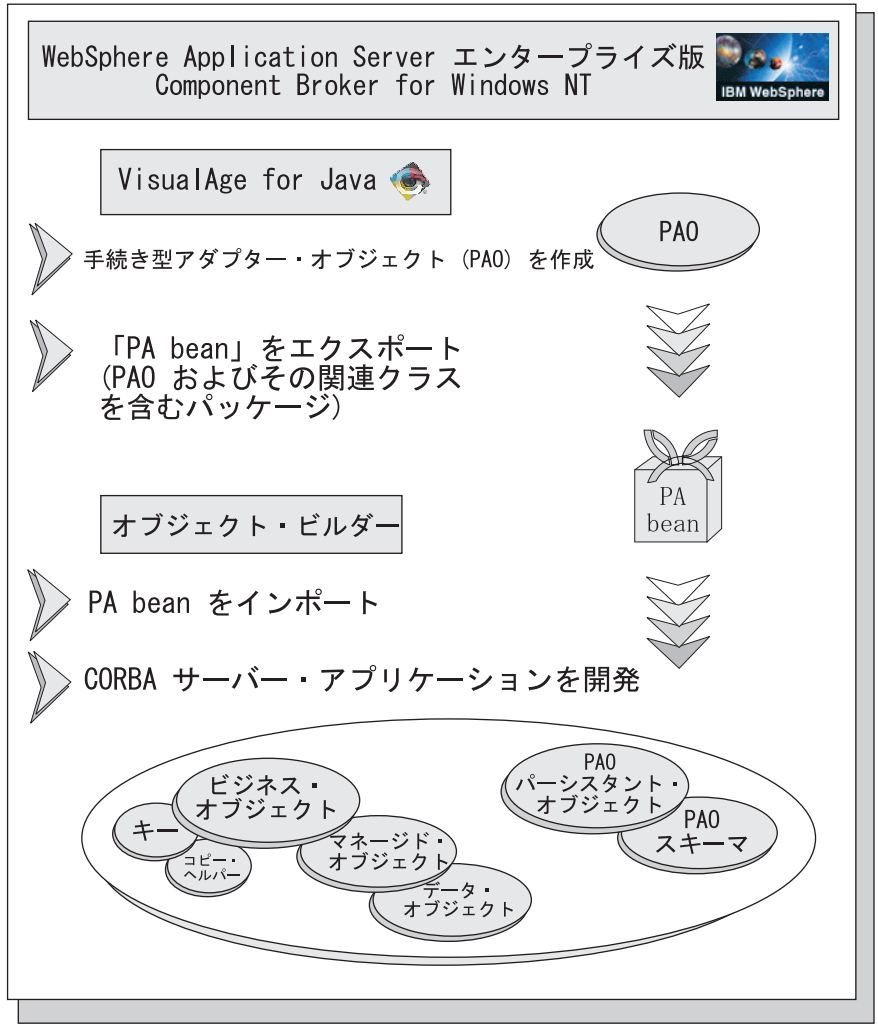

図 4. PAA サポートを使用する CORBA アプリケーションのコンポーネント・オブジェクトの開発

8ページの図5 は、PAO とその bean、またはクラス、およびその機能の詳細を示しています。

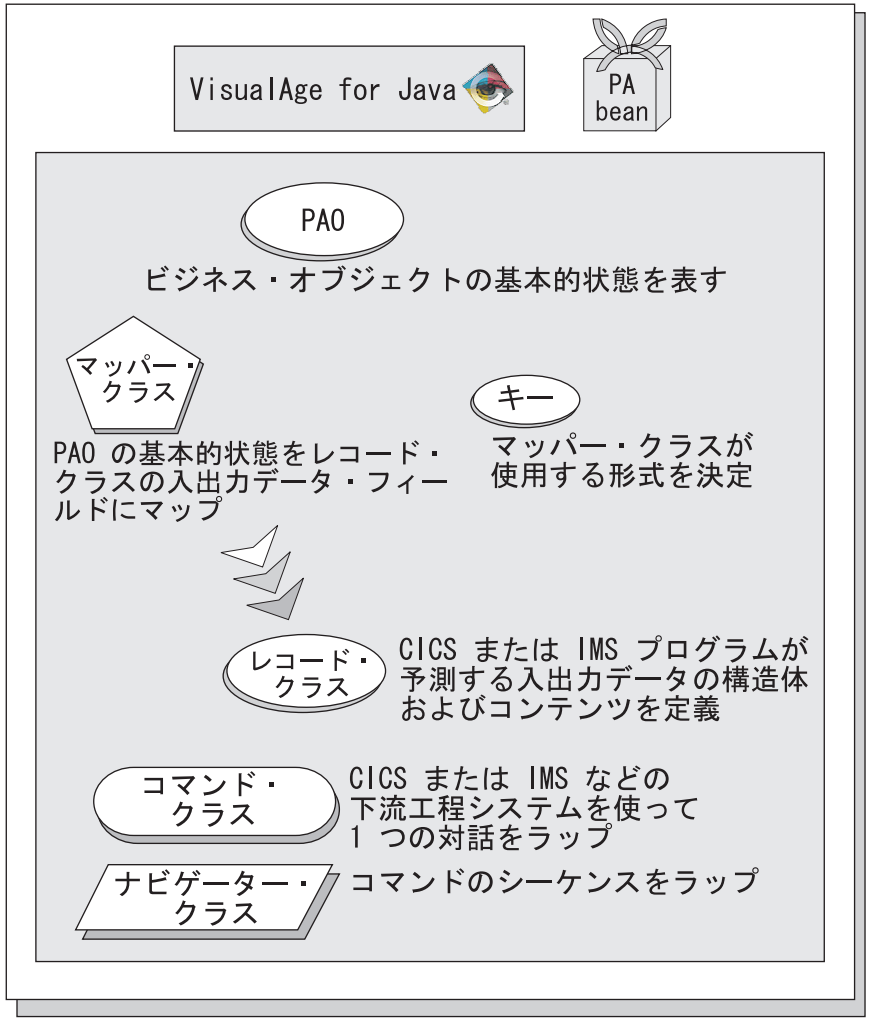

図 5. PA bean パッケージ : PAO およびその関連付けられたクラス

残りの開発過程は、3ページの図2 および 5ページの図3 で示すものと同じです。

ソース・コードを生成しアプリケーション・ファミリーをパッケージした後、成果物を開発過程から、WebSphere for z/OS がインストールされている z/OS または OS/390 システムに転送します。このシステムで、make ファイルを実行して実行可能コードを生成することができます。本書では、転送とコード生成を完了するために必要な詳細について説明します。コードを生成すると、9ページの『配置するサーバー・アプリケーションのバックグラウンド』に示す

ように、WebSphere for z/OS アプリケーション・サーバーで CORBA アプリケーションを配置する準備ができます。

配置するサーバー・アプリケーションのバックグラウンド

実行可能 CORBA アプリケーションができあがると、そのランタイム環境をセットアップする必要があります。サンプルによって、DB2 表のセットアップ、特定のクライアント・アプリケーションのセットアップ、および 1 つの CORBA アプリケーションに固有のその他の作業など、追加の作業が必要になる場合があります。このような場合には、本書でこれらの追加作業のアウトラインを説明します。

ただし、どのような場合でも WebSphere for z/OS 管理アプリケーションを使用します。これは Windows NT 上で稼動し、アプリケーション・サーバーを定義します。アプリケーション・サーバーを定義するときに、10ページの図6で示すエレメントを構成するモデルを作成します。

注: WebSphere for z/OS には、2 種類のアプリケーション・サーバーがあります。1 つは Java 2 エンタープライズ版 (J2EE) アプリケーション、もう 1 つは CORBA アプリケーション用のアプリケーション・サーバーです。CORBA アプリケーション用のサーバーは、マネージド・オブジェクト・フレームワーク (MOFW) サーバーと呼ばれています。WebSphere for z/OS 管理アプリケーションを使用して CORBA アプリケーションにサーバーを定義する際、管理アプリケーションではサーバーに対して J2EE サーバーおよび Server (MOFW サーバー・タイプの場合) の 2 つのラベルが使用されます。CORBA アプリケーションの場合は、Server を使用するようにしてください。

サーバーを定義するステップには、以下が含まれます。

1. 新規会話の開始
2. MOFW アプリケーション・サーバーの追加
3. MOFW サーバー・インスタンスの追加
4. コンテナーの追加
5. 論理リソース・マッピング (LRM)、インスタンスおよび接続の追加
6. CORBA サーバー・アプリケーションのインポート

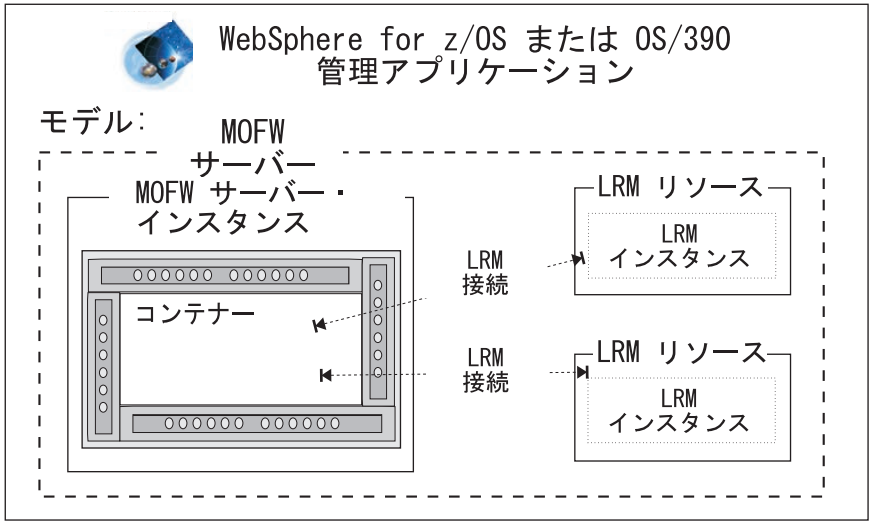

図 6. ランタイム環境、または MOFW アプリケーション・サーバーの定義

ユーザーが定義するサーバー・モデルには、サーバーの属性、すなわち、アプリケーションのランタイム環境を制御する環境変数が組み込まれています。本書の説明には、各種の CORBA アプリケーションに設定する環境変数に関する情報が含まれます。ユーザー独自のアプリケーション用の WebSphere for z/OS サーバーの作成を開始するときには、詳細について 87ページの『WebSphere for z/OS MOFW サーバーの環境変数の設定に関するバックグラウンド』を参照してください。

また、このサーバー定義の一部として、CORBA アプリケーションの構造を定義する DDL (メタデータ) をインポートします。11ページの図7 は、サーバー定義と DDL インポート過程の結果 (アプリケーション・サーバーのモデル、コンテナー、ホーム、および CORBA アプリケーション) を示しています。

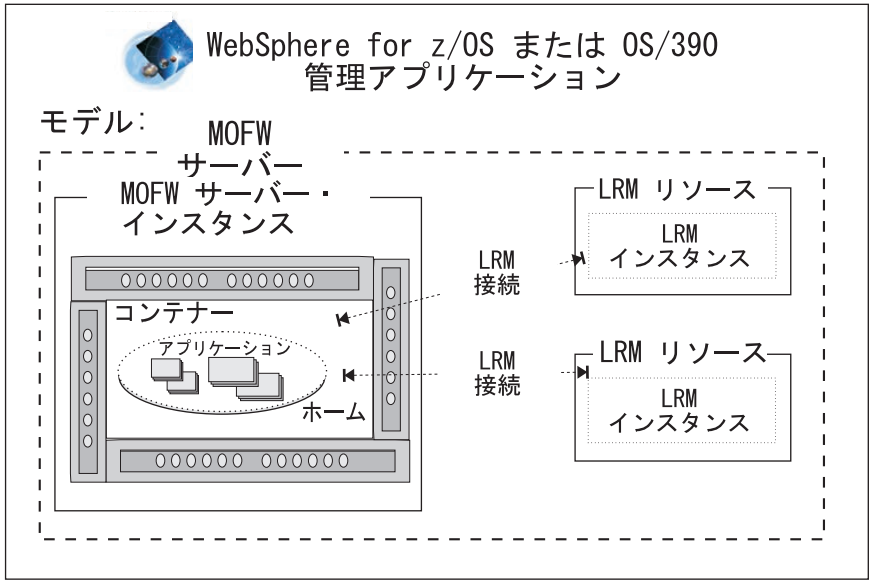

図 7. MOFW サーバーに対するアプリケーション・ファミリーの定義

ユーザーの CORBA アプリケーション用のランタイム環境のモデルを入手したら、12ページの図8 に示すように、この過程の最後の段階である、z/OS または OS/390 上の活動ランタイム環境へのモデルの変換を開始します。

まず、WebSphere for z/OS 管理アプリケーションを使用して、サーバー・モデルをコミットします。これは、定義を永久に保管することに似ています。コミット後のモデルには、どんな変更も加えることができません。次に、z/OS または OS/390 上で、セキュリティー定義などの手動の作業をいくつか完了しなければなりません。本書の説明では、これらの作業の手助けとなるアドバイスを提供しています。 z/OS または OS/390 システムのこのような局面に慣れていない場合は、システム・プログラマーまたはセキュリティー管理者の手助けも必要です。

最後に、WebSphere for z/OS 管理アプリケーションを介してサーバーを活動化することができます。

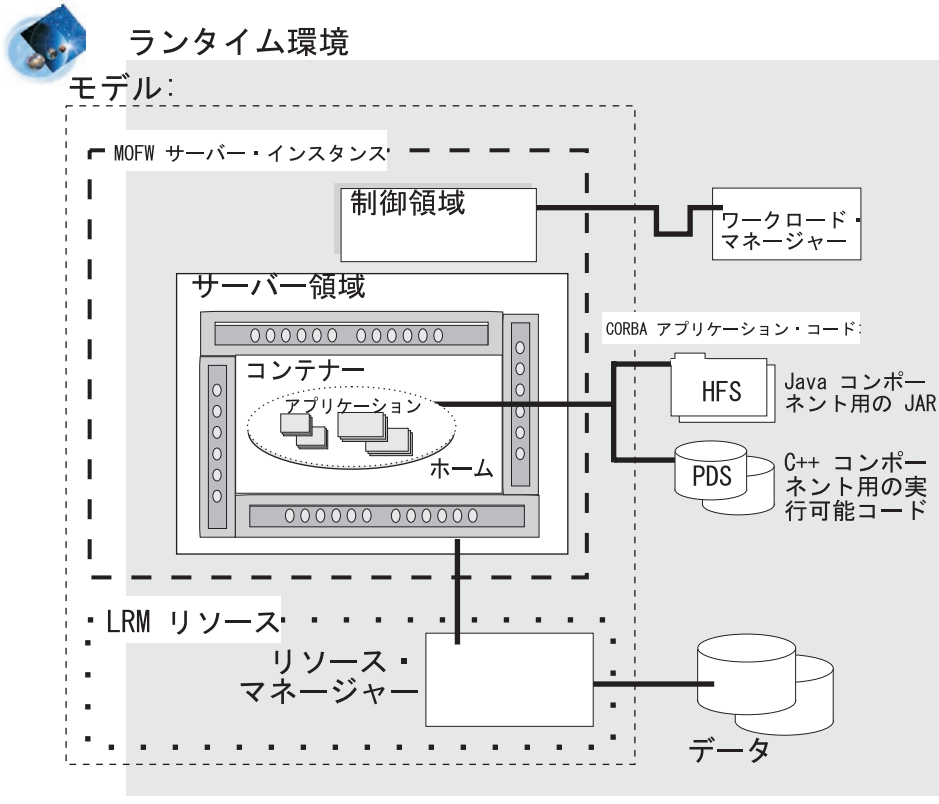

図 8. MOFW アプリケーション・サーバーの活動化

サーバーが活動化されると、開発したばかりの CORBA アプリケーションを使用するクライアント・アプリケーションを実行することができます。クライアント・アプリケーションの種類によっては、多少のセットアップ作業が必要となる可能性があります。本書の説明では、これらの作業の手助けとなるモデルまたはアドバイスを提供します。

開発環境と配置環境のセットアップのバックグラウンド

サイトの役割と責任によっては、システム・プログラマーが、ユーザーのためのアプリケーション開発環境と配置環境をすでにセットアップしている場合があります。このような場合には、すべてが正常にセットアップされているかどうかを検査するだけでかまいません。そうでない場合は、一部のインストール作業を自分で行う必要がある場合があります。どちらの場合も、このセクションでは最低限の要件を検討し、必要な場合には、詳細情報を検索する場所を示

します。これらのセットアップ作業を完了するために、システム・プログラマー、セキュリティー管理者、またはデータベース管理者の手助けが必要な場合があります。

CORBA アプリケーションの開発に関する本書の説明では、Component Broker for Windows NT と VisualAge コンポーネント開発ツールがインストールされている Windows NT ワークステーションで作業していることを想定しています。ただし、WebSphere Application Server エンタープライズ版がサポートするその他のワークステーション環境でアプリケーションを開発してもかまいません。CORBA アプリケーションを WebSphere for z/OS アプリケーション・サーバーに配置するには、z/OS または OS/390 UNIX 環境でも作業している必要があります。アプリケーション開発環境に関する詳細については、14ページの『アプリケーション開発とアセンブリー環境のセットアップ』を参照してください。

本書の説明では、単純化を図るため、アプリケーション開発作業を実行するのと同じ z/OS または OS/390 システムで、CORBA アプリケーションとそのクライアント・プログラムを配置および実行することも想定しています (つまり、WebSphere for z/OS アプリケーション・サーバーが実行されるシステムと同じシステムでクライアントおよびサーバー・アプリケーション・コードをコンパイルします)。独自のサーバー・アプリケーションを開発および配置するときは、このクライアント / サーバー構成は単純すぎてユーザーのニーズに応えられない場合があります。複雑な構成を使用するときは、サンプル・アプリケーションを管理する WebSphere for z/OS アプリケーション・サーバーと、サンプル・アプリケーションのすべてのクライアントがいずれも必要なサンプル・アプリケーション・コードに対するアクセス権を持つようにしてください。

サーバー・アプリケーションを WebSphere for z/OS アプリケーション・サーバーに配置するには、Windows NT 上で稼動するプログラム WebSphere for z/OS 管理アプリケーションを使用して、そのアプリケーション・サーバー (またはランタイム環境) に必要なセットアップの大部分を完了します。サンプル・アプリケーションの配置に関する説明では必要なすべての作業をリストしますが、セキュリティーのセットアップなど一部の作業の実行方法は指示していません。これは、このような作業は、それぞれのお客様のインストールごとに異なるからです。これらの作業に関する詳細については、15ページの『サーバー・アプリケーションのランタイム環境のセットアップ』をご覧ください。

アプリケーション開発とアセンブリー環境のセットアップ

ユーザーのサイトのシステム・プログラマーが、WebSphere for z/OS をインストールする際、オプションとしてアプリケーション開発環境をセットアップすることもできます。*WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652 で示される説明は、48ページの『アプリケーション開発環境をセットアップするためのステップ』 にリストされています。したがって、正しい環境を自分で検査するかまたはインストールすることができます。

CORBA アプリケーションを作成するには、Component Broker for Windows NT 3.5 で使用可能な VisualAge コンポーネント開発ツールを使用する必要があります。一部のソフトウェア製品をインストールしなければならないことが分かった場合は、*WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652 にあるソフトウェア要件に関する情報を調べ、正しい製品バージョンを確実にインストールしてください。

CORBA アプリケーションを作成するには、以下の作業を実行しなければならない場合があります。

作業	関連するバックグラウンド情報または手順 (参照項目)
z/OS または OS/390 プラットフォーム用にアプリケーション成果物を生成するために、ワークステーション環境をセットアップする。	『z/OS または OS/390 用にアプリケーション成果物を生成するためのワークステーション・ツールの設定』
z/OS または OS/390 上でソース・コードをコンパイルするために環境変数の一部を変更する。	15ページの『make 環境変数を変更して、z/OS または OS/390 上のソース・ファイルをコンパイルする』

z/OS または OS/390 用にアプリケーション成果物を生成するためのワークステーション・ツールの設定

開発している CORBA アプリケーションによって、以下の 1 つ以上のツールを使用して、コンポーネント・オブジェクトを開発し、そのためのソース・ファイルを生成します。z/OS または OS/390 上に配置するコンポーネントおよびコードの場合は、ツールが正しく設定されているかどうか以下のことを確認します。

VisualAge for Java

必ず、以下のことを行ってください。

- 「**ウィンドウ**」→「**オプション**」と選択します。「**ビジュアル・コンポジション**」を選択して、「**bean スーパークラスの BeanInfo の継承**」をクリアします。
- 作成している bean に適切なフィーチャーを追加します。個々のサンプル命令には、追加する正しいフィーチャーがリストされています。

オブジェクト・ビルダー

新規モデルを始動する場合は、「**プラットフォーム (Platform)**」を選択して、以下のものを設定します。

- 「**表示 (View)**」は 390
- 「**生成 (Generate)**」は 390
- 「**制約 (Constrain)**」は 390

make 環境変数を変更して、z/OS または OS/390 上のソース・ファイルをコンパイルする

作成過程の環境変数は、WebSphere for z/OS 製品とともに出荷される CB390make.env ファイルで設定されています。インストール過程の一部として、システム・プログラマーは、ユーザーのサイトで使用するためにこのファイルを調整することができます。

使用すべき環境変数値は、インストールのガイドラインまたは個々の開発環境によって異なります。特定の変数の意味に関する詳細が必要な場合は、65ページの『make 処理に関するバックグラウンド』の説明を参照してください。

サーバー・アプリケーションのランタイム環境のセットアップ

9ページの『配置するサーバー・アプリケーションのバックグラウンド』にある一連の図は、会話としても知られる、サーバー構成を定義するために WebSphere for z/OS 管理アプリケーションを介して完了するステップを示しています。ただし、その会話を活動化できるようになる前に、一部の手動の作業を完了して、そのサーバー用のランタイム環境もセットアップする必要があります。

特に、JCL プロシージャーを作成して、制御領域とサーバー領域を開始する必要があります。WebSphere for z/OS 製品とともに出荷されるサンプル JCL プロシージャーのコピーと変更を行うことができます。本書の説明では、JCL サンプルと必要な変更箇所がどこにあるかを示しています。87ページの『WebSphere for z/OS MOFW サーバーを始動する JCL プロシージャーのコーディング』を参照してください。

クライアント・アプリケーションのセットアップ

WebSphere for z/OS サーバーにインストールされた CORBA アプリケーションを駆動するクライアント・アプリケーションの開発を開始する際は、以下を参照してください。

- クライアント・アプリケーションの設計とコーディングに関する 107ページの『サーバー・アプリケーションのためのクライアントの設計およびコーディングに関するバックグラウンド』。
- z/OS または OS/390 上でサーバー・アプリケーションを使用するクライアント・アプリケーションの準備に関する 103ページの『第5章 z/OS または OS/390 でのクライアント・アプリケーションの開発、アセンブル、および配置』。

セキュリティー管理のセットアップ

サーバー・アプリケーションの DDL ファイルをインポートする前に、適切なセキュリティー定義をセットアップしているかどうか確認してください。インポート・プロセスのための入力および出力ファイルは、システム管理サーバー領域の ID と関連付けられているため、そのユーザー ID は以下のファイルにアクセスできる必要があります。

- インポートする DDL にデータ・セットを使用する場合、ユーザー ID には、入力データ・セットに対する読み取りアクセスと、出力データ・セットに対する更新アクセスが必要です。
- DDL に HFS ファイルを使用する場合は、ユーザー ID には、入力ファイルを検索するためのディレクトリー検索機能、入力ファイルの読み取り機能、および出力ファイルへの書き込み機能が備わっていなければなりません。

第2章 WebSphere for z/OS における CORBA アプリケーションの開発方法の学習

Component Broker の共通プログラミング・モデルおよびそのツール・セットの機能性によって、特定のプラットフォームのための複雑なプログラミングを学習する代わりに、ビジネス・モデルを中心に、再利用可能で移植可能なビジネス・オブジェクトを開発することができます。これらのオブジェクトは、WebSphere for z/OS サーバーに配置されると、z/OS または OS/390 上でデータベースおよびトランザクション・システムに高度に統合された環境で稼動します。このような統合のために、z/OS または OS/390 とそのサブシステムの知識があると、z/OS または OS/390 環境向けに最適化された効率的なアプリケーションを設計する際に役立ちます。

WebSphere for z/OS サーバーで稼動する独自の CORBA アプリケーションを開発する準備ができたら、本書のトピックをお読みください。この中には、設計の考慮事項、高水準の機能についての説明、z/OS または OS/390 固有のガイドライン、および現行レベルの WebSphere for z/OS の制約事項や制限が含まれています。この情報は、z/OS または OS/390 が提供する質の高いサービスを受けるアプリケーションの開発に役立ちます。

始める前に: WebSphere for z/OS サーバー向け CORBA アプリケーションの設計とコーディングを開始する前に、以下のことを行ってください。

- 1ページの『第1章 WebSphere for z/OS の CORBA アプリケーション入門』の情報を検討してください。この情報によって、独自のアプリケーションのために行う開発、アセンブリー、および配置プロセス全体を実行することができます。本章には、WebSphere for z/OS サーバーでのアプリケーションの開発、アセンブル、および配置のためにしておかなければならない作業の学習に役立つ、予備的な情報もいくつか含まれています。
- 開発中の CORBA アプリケーションのタイプによって、以下のリソースの 1 つ以上を使用可能にします。
  - *WebSphere Application Server エンタープライズ版 Component Broker プログラミングの手引き*, SD88-7372 は、ビジネス・オブジェクト、データ・オブジェクト、ならびに、マネージド・オブジェクト・フレームワーク、IDL、および C++ CORBA プログラミングに関する情報を含むプログラミング・モデルについて説明しています。

– *WebSphere Application Server エンタープライズ版 Component Broker プログラミング解説書、第 1 巻 および第 2 巻*, SD88-7376 および SD88-7377 は、Component Broker アプリケーション開発者が使用可能なアプリケーション・プログラミング・インターフェース (API) に関して説明しています。
– *WebSphere Application Server エンタープライズ版 Component Broker 上級プログラミングの手引き*, SD88-7373 は、CORBA オブジェクト・サービスおよび Component Broker オブジェクト・リクエスト・ブローカー (リモート・メソッド呼び出しおよび動的起動インターフェース (DII) のプロシージャーを含む) のための Component Broker インプリメンテーション、およびその他のトピックについて説明しています。
– *WebSphere Application Server エンタープライズ版 Component Broker アプリケーション開発ツールの手引き*, SD88-7378 は、CBToolkit で提供されるツールによる、Component Broker アプリケーションの作成方法およびテスト方法について、継承およびチーム開発などの共通の開発シナリオを中心に説明しています。
– *WebSphere Application Server エンタープライズ版 Component Broker 用語集*, SD88-7380 では、共通して使用される用語を定義しています。

次の表は、独自の CORBA アプリケーション開発のためのサブタスク、および関連情報を示しています。

サブタスク	関連情報 (参照項目)
z/OS または OS/390 環境がアプリケーション設計に与える影響についての学習	• 19ページの『WebSphere for z/OS トランザクション環境に関するバックグラウンド』 • 30ページの『CORBA アプリケーションおよびクライアント・アプリケーションの制約事項』 • 30ページの『CORBA アプリケーションから最適パフォーマンスを取得するためのガイドライン』 • 107ページの『サーバー・アプリケーションのためのクライアントの設計およびコーディングに関するバックグラウンド』

サブタスク	関連情報 (参照項目)
ワークステーション・ツールを使用した z/OS または OS/390 向け CORBA アプリケーションの作成	• 31ページの『z/OS または OS/390 にアプリケーション成果物を生成するワークステーション・ツールの設定に関するバックグラウンド』 • 31ページの『CORBA アプリケーションに必要なコンポーネント・オブジェクトに関するバックグラウンド』 • 32ページの『CORBA ビジネス・オブジェクトを C++ または Java で開発するためのガイドライン』 – 33ページの『IMS リソースを使用する CORBA アプリケーションの開発のためのガイドライン』 – 42ページの『CICS リソースを使用する CORBA アプリケーションの開発のためのガイドライン』 • 44ページの『コンテナー作成のバックグラウンド』

WebSphere for z/OS トランザクション環境に関するバックグラウンド

CORBA アプリケーションの開発の一環として、アプリケーションの一部である、ビジネス・オブジェクトごとに 1 つのコンテナーを定義します。コンテナー・ポリシーを定義することもできます。これは、(オブジェクト・ビルダーによる) 開発過程で、特定のコンテナーによりそのオブジェクトを管理する方法を示しています。ただし、ここでのポリシー設定は、WebSphere for z/OS サーバー内のコンテナーには影響しません。z/OS または OS/390 に配置されるサーバー・アプリケーションのコンテナー・ポリシーを定義するには、WebSphere for z/OS 管理アプリケーションを使用します。

設定する 1 つのコンテナー・ポリシーは、コンテナーが管理するオブジェクトのトランザクションの有効範囲を定義します。WebSphere for z/OS は、現在、CORBA アプリケーション用に 4 つのトランザクション・ポリシーをサポートしていますが、ほとんどのアプリケーションでその 1 つである TX 必須を使用します。 TX 必須を使用して、コンテナーは、クライアント・アプリケーションのグローバル・トランザクションを使用するか、クライアント用のグローバル・トランザクションを開始します。グローバル・トランザクションでは、WebSphere for z/OS サーバー、z/OS または OS/390 リソース・リカバリー・サービス (RRS)、およびその他の関連したリソース・マネージャー (DB2 およ

び IMS など) が一緒に稼動して、サーバー・アプリケーションの処理は、アトミック・オペレーションとして調整および処理されます。つまり、分散リソースに対するアプリケーションの更新については、すべて更新される (コミットされる) か更新されない (ロールバックされる) かのいずれかです。

注: この制限は、WebSphere for z/OS MOFW サーバーにのみ適用されます。J2EE アプリケーション (エンタープライズ bean、サーブレット、および JavaServer Pages) のみがインストールされる WebSphere for z/OS J2EE サーバーには、コンテナー定義は必要ありません。J2EE サーバーは、所定の J2EE アプリケーションまたはその個々のコンポーネントの配置記述子に設定された属性を使用します。J2EE サーバーの詳細については、*WebSphere Application Server V4.0 for z/OS and OS/390: J2EE アプリケーションのアセンブル*, SA88-8654 を参照してください。

その他の 3 つのポリシーは、グローバル・トランザクション環境の欠如をシミュレートするので、サーバー・アプリケーションの特定のタイプのパフォーマンスだけを向上させます。つまり、コンテナーは、オブジェクトに対するトランザクションを表したり管理したりしません。代わりに、コンテナーは、RRS およびその他の z/OS または OS/390 リソース・マネージャーにより、トランザクションの内容を管理できるようにします。これら 3 つの代替ポリシーを安全に使用するには、サーバー・アプリケーションの処理を完全に理解し、ポリシー別の規則に従わなければなりません。そうしなければ、アプリケーションは、期待どおりの振る舞いをしない場合があります。

推奨: 21ページの表1 で要約しているように、サーバー・アプリケーションが、その他のポリシーのいずれかの規則に従うことを保証できない限り、 TX 必須を使用してください。これらのその他のポリシーは、特定の特性および処理が適用されるサーバー・アプリケーションの場合のみ、パフォーマンスが向上します。WebSphere for z/OS 製品の完成度が高まるにつれて、すべてのアプリケーションに対してパフォーマンスは向上するはずです。

21ページの表1 は、WebSphere for z/OS MOFW サーバーが現在サポートしているコンテナー・トランザクション・ポリシーの要約を示しています。

表 1. MOFW サーバーのコンテナー・トランザクション・ポリシーおよびその使用規則の要約

コンテナー・トランザクション・ポリシーの種類	CORBA アプリケーションが従うべき規則
TX 必須 (デフォルト・コンテナー・ポリシー) WebSphere for z/OS MOFW サーバーは、クライアント・アプリケーションのグローバル・トランザクションを使用するか、クライアント用のグローバル・トランザクションを開始します。25ページの図9 を参照してください。 サーバー、RRS およびその他の関連するリソース・マネージャーは、すべてグローバル・トランザクションの管理に関係しています。トランザクション・コンテキストのサーバー管理について、詳しくは、24ページの『TX 必須コンテナー・ポリシーを使用する場合の規則』を参照してください。	サーバーは、クライアント・アプリケーションのグローバル・トランザクションを使用するか、グローバル・トランザクションを開始するので、オブジェクトは、まず、`suspend` を発行して、既存のグローバル・トランザクションを中断し、`begin` を発行して、新規トランザクションを開始します。 ビジネス・オブジェクトによりトランザクション・コンテキストを管理したい場合は、追加規則が適用されます。24ページの『TX 必須コンテナー・ポリシーを使用する場合の規則』 を参照してください。

表 1. MOFW サーバーのコンテナー・トランザクション・ポリシーおよびその使用規則の要約 (続き)

コンテナー・トランザクション・ポリシーの種類	CORBA アプリケーションが従うべき規則
TX MOFW 結合ハイブリッド・グローバル WebSphere for z/OS MOFW サーバーは、クライアント・トランザクションがあったとしても、それを無視し、RRS およびその他の関連するリソース・マネージャーがハイブリッド・グローバルなトランザクションを管理できるようにします。 同じコンテナーで実行されているメソッドは、すべて同じハイブリッド・グローバルなトランザクションを共用しますが、その他のコンテナーのオブジェクトに対するメソッドは、その他のコンテナーに指定されているトランザクション・ポリシーによっては、別のトランザクション・コンテキストが必要になる場合があります。また、メソッドが異なるサーバーのオブジェクトに対して駆動される場合、サーバーは、ハイブリッド・グローバルなトランザクションをこれらのリモート・サーバーに伝えません。トランザクション・コンテキストのサーバー管理について、詳しくは、26ページの『TX MOFW 結合ハイブリッド・グローバル・コンテナー・ポリシーを使用する場合の規則』を参照してください。	ビジネス・オブジェクトが、トランザクションを開始する `begin` を最初に発行するように設計されている場合、オブジェクトは、最初のオブジェクト状態を保持するために、DB2 データベースなどの保護リソースにアクセスするようにコンテナーに対して要求することはできません。この場合、サーバーは、ハイブリッド・グローバル・トランザクションを自動的に開始して、パーシスタント・データ・ストアから初期状態を取り出します。サーバーが一度ハイブリッド・グローバル・トランザクションを開始すると、サーバーはオブジェクトがトランザクションのコンテキストを管理することを許可しません。 追加規則が適用されます。詳しくは、26ページの『TX MOFW 結合ハイブリッド・グローバル・コンテナー・ポリシーを使用する場合の規則』を参照してください。

表 1. MOFW サーバーのコンテナー・トランザクション・ポリシーおよびその使用規則の要約 (続き)

コンテナー・トランザクション・ポリシーの種類	CORBA アプリケーションが従うべき規則
TX MOFW 分離ハイブリッド・グローバル TX MOFW 結合ハイブリッド・グローバル・ポリシーに関して、WebSphere for z/OS MOFW サーバーは、クライアント・トランザクションがあったとしても、それは無視して、RRS およびその他の関連するリソース・マネージャーにより、グローバルなトランザクションを管理できるようにします。また、メソッドが異なるサーバーのオブジェクトに対して駆動される場合、サーバーは、ハイブリッド・グローバルなトランザクションをこれらのリモート・サーバーに伝えません。 ただし、TX MOFW 結合ハイブリッド・グローバル・ポリシーと異なり、サーバーで実行される各オブジェクト・メソッドには、独自のハイブリッド・グローバル・トランザクションがあります。トランザクション・コンテキストのサーバー管理について、詳しくは、28ページの『TX MOFW 分離ハイブリッド・グローバル・コンテナー・ポリシーを使用する場合の規則』を参照してください。	ビジネス・オブジェクトが、トランザクションを開始する `begin` を最初に発行するように設計されている場合、オブジェクトは、最初のオブジェクト状態を保持するために、DB2 データベースなどの保護リソースにアクセスするようにコンテナーに対して要求することはできません。この場合、サーバーは、ハイブリッド・グローバル・トランザクションを自動的に開始して、パーシスタント・データ・ストアから初期状態を取り出します。サーバーが一度ハイブリッド・グローバル・トランザクションを開始すると、サーバーはオブジェクトがトランザクションのコンテキストを管理することを許可しません。 追加規則が適用されます。詳しくは、28ページの『TX MOFW 分離ハイブリッド・グローバル・コンテナー・ポリシーを使用する場合の規則』を参照してください。

表 1. MOFW サーバーのコンテナー・トランザクション・ポリシーおよびその使用規則の要約 (続き)

コンテナー・トランザクション・ポリシーの種類	CORBA アプリケーションが従うべき規則
TX MOFW サポート結合ハイブリッド・グローバル TX 必須以外のポリシーに関して、WebSphere for z/OS MOFW サーバーは、RRS およびその他の関連するリソース・マネージャーがグローバルなトランザクションを管理できるようにします。 ただし、この場合、サーバーは、クライアント・アプリケーションのトランザクションのコンテキストを尊重します。クライアントにグローバル・トランザクションがある場合は、同じサーバー・インスタンスで実行されるメソッドはすべて、その同じグローバル・トランザクションを共用します。トランザクション環境は、 TX 必須ポリシーの場合の環境と同じです。	サーバー・アプリケーションが使用するオブジェクトはすべて、同じサーバー・インスタンスで実行されるように構成されていなければなりません。 追加規則が適用されます。詳しくは、29ページの『TX MOFW サポート結合ハイブリッド・グローバル・コンテナー・ポリシーを使用する場合の規則』を参照してください。

TX 必須コンテナー・ポリシーを使用する場合の規則

TX 必須コンテナー・ポリシーに関して、WebSphere for z/OS MOFW サーバーは、25ページの図9 で示しているように、クライアント・アプリケーションのグローバル・トランザクションを使用するか、クライアント用のグローバル・トランザクションを開始します。オブジェクトが、異なる WebSphere for z/OS サーバーにある別のオブジェクトに対してメソッドを駆動する場合、サーバーは、このグローバル・トランザクションをリモート・サーバーに伝えます。

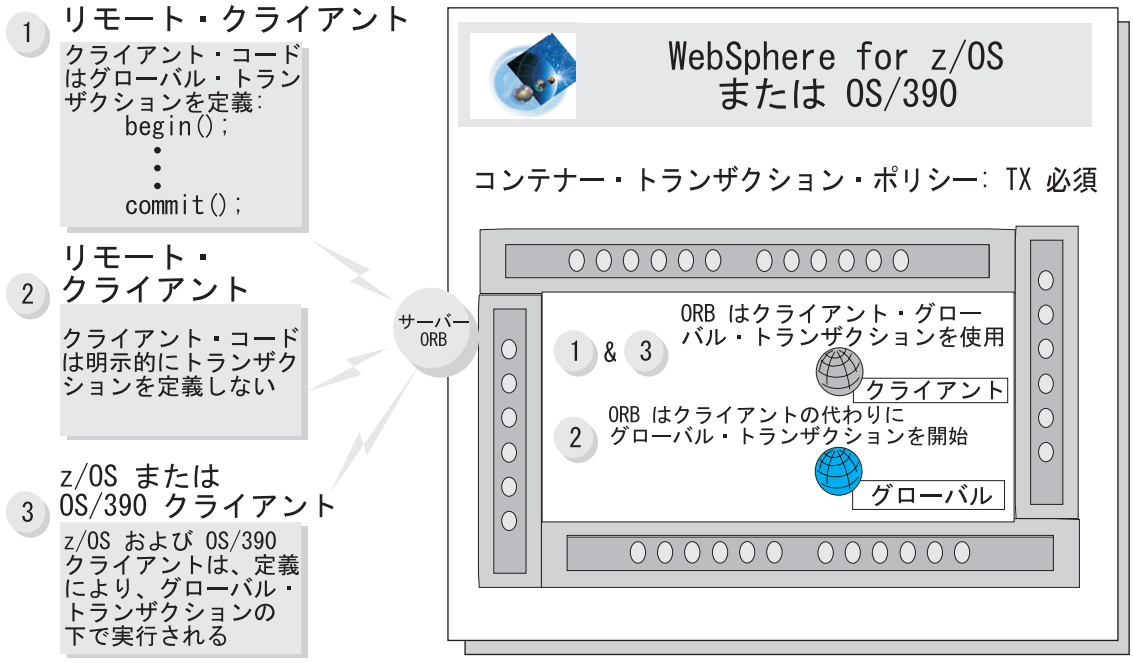

図 9. 大部分の CORBA アプリケーションの WebSphere for z/OS MOFW サーバー・トランザクション環境

ビジネス・オブジェクトによりトランザクション・コンテキストを管理したい場合は、最初に現行トランザクションを中断するように設計しておかなければなりません。そうすると、以下のようになります。

- 新規トランザクションを開始する `begin` を指定するか、既存のトランザクションを再開する `resume` を指定する前に、マネージド・オブジェクト・メソッドを起動することはできません。新規トランザクションを開始することも、既存のトランザクションを再開することもしないと、処理結果は予測不能になります。
- オブジェクトが実行を開始したときに、アクティブであったトランザクションをまず再開しないと、オブジェクトはその処理を終了しないことを確認しておく必要があります。最初のアクティブ・トランザクションの下でオブジェクト処理を終了しないと、サーバー領域は異常終了し、最初のアクティブ・トランザクションは強制的にロールバックされます。

トランザクション・オブジェクトは、トランザクション・コンテキストの管理に使用するビジネス・オブジェクトで使用可能です。たとえば、ビジネス・オブジェクトは、CosTransactions::Control オブジェクトを使用することができます。これにより、アプリケーションはトランザクション・コンテキストを表す

ことができます。トランザクション・オブジェクトについて、詳しくは、*Component Broker 上級プログラミングの手引き* を参照してください。

TX 必須コンテナー・ポリシーでは、トランザクションのネスティングを許可していないので、以下のように、システム例外またはユーザー例外が発生したときに、特定のアプリケーションを実行するように、サーバー・アプリケーションを設計することはできません。

- システム例外が発生した場合、WebSphere for z/OS MOFW サーバーはトランザクションをロールバックします。
- ユーザー例外が発生した場合、サーバーは以下のいずれかを実行します。
  - サーバー・アプリケーションに対して、インターフェース定義言語 (IDL) でこのユーザー例外を定義している場合に限り、トランザクションをコミットします。
  - ユーザー例外をシステム例外に変換して、トランザクションをロールバックします。

TX MOFW 結合ハイブリッド・グローバル・コンテナー・ポリシーを使用する場合の規則

TX MOFW 結合ハイブリッド・グローバル・コンテナー・ポリシーに関して、27ページの図10 に示しているように、WebSphere for z/OS MOFW サーバーは、クライアント・トランザクションがあったとしても、それは無視して、RRS およびその他の関連するリソース・マネージャーにより、ハイブリッド・グローバルなトランザクションを管理できるようにします。オブジェクトに設定された初期トランザクション・コンテキストは、TX MOFW 結合ハイブリッド・グローバルと TX MOFW 分離ハイブリッド・グローバルの両方のコンテナー・ポリシーで同じであることに注意してください。

同じコンテナーで実行されているメソッドは、すべて同じハイブリッド・グローバルなトランザクションを共用しますが、その他のコンテナーのオブジェクトに対するメソッドは、その他のコンテナーに指定されているトランザクション・ポリシーによっては、別のトランザクション・コンテキストが必要になる場合があります。メソッドが異なるサーバーのオブジェクトに対して駆動される場合、サーバーは、ハイブリッド・グローバルなトランザクションをこれらのリモート・サーバーに伝えません。

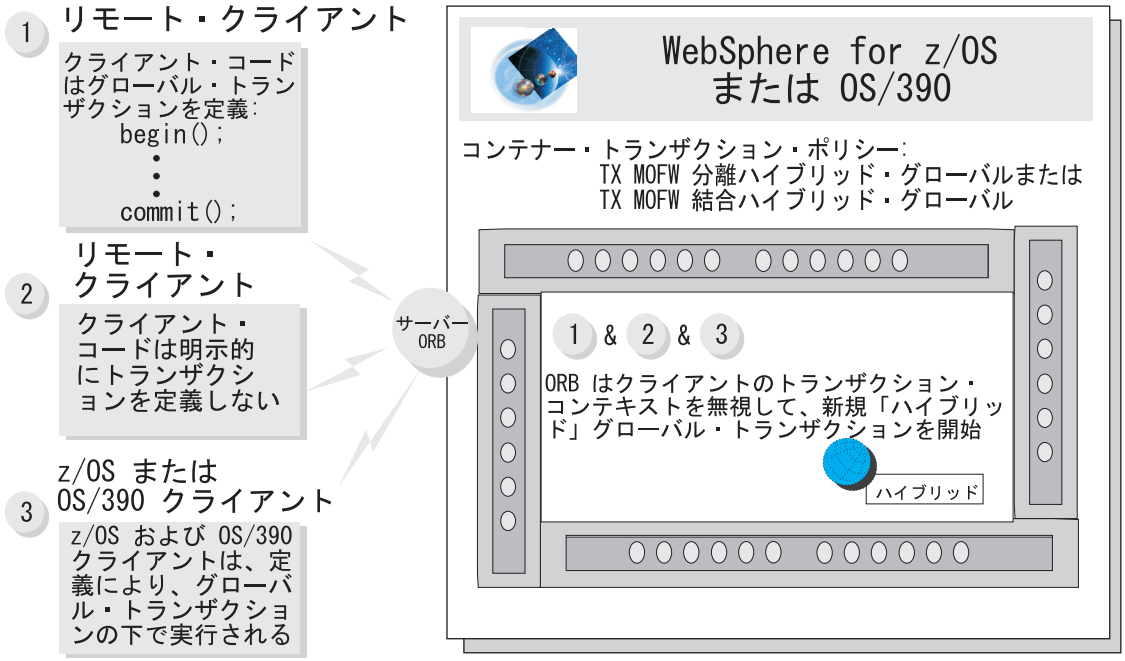

図 10. TX MOFW 結合ハイブリッド・グローバル コンテナー・ポリシーの WebSphere for z/OS MOFW サーバー・トランザクション環境

ビジネス・オブジェクトが、トランザクションを開始する begin を最初に発行するように設計されている場合、オブジェクトは、最初のオブジェクト状態を保持するために、DB2 データベースなどの保護リソースにアクセスするようにコンテナーに対して要求することはできません。この場合、サーバーは、ハイブリッド・グローバル・トランザクションを自動的に開始して、パーシスタント・データ・ストアから初期状態を取り出します。サーバーが一度ハイブリッド・グローバル・トランザクションを開始すると、サーバーはオブジェクトがトランザクションのコンテキストを管理することを許可しません。

追加規則

- サーバー・アプリケーションは、グローバル・トランザクションの下で実行されることを想定して、設計されていなければなりません。
- サーバー・アプリケーションは、そのトランザクション・コンテキストを管理しようとすることはできません。
- 相互作用するオブジェクト同士は、すべて同じサーバーに配置されていなければなりません。

TX MOFW 分離ハイブリッド・グローバル・コンテナー・ポリシーを使用する場合の規則

TX MOFW 結合ハイブリッド・グローバル・ポリシーに関して、WebSphere for z/OS MOFW サーバーは、クライアント・トランザクションがあったとしても、それは無視して、RRS およびその他の関連するリソース・マネージャーにより、グローバルなトランザクションを管理できるようにします。また、メソッドが異なるサーバーのオブジェクトに対して駆動される場合、サーバーは、ハイブリッド・グローバル・トランザクションをこれらのリモート・サーバーに伝えません。この初期トランザクション・コンテキストについては、27 ページの図10 を参照してください。これは、TX MOFW 結合ハイブリッド・グローバルと TX MOFW 分離ハイブリッド・グローバルの両方のコンテナー・ポリシーの場合と同じです。

ただし、TX MOFW 結合ハイブリッド・グローバル・ポリシーと異なり、サーバーで実行される各オブジェクト・メソッドには、独自のハイブリッド・グローバル・トランザクションがあります。

このポリシーは、ローカル・リモート透過性を保証しています。つまり、オブジェクトは、配置される場所に関係なく同じように振る舞います。TX MOFW 分離ハイブリッド・グローバルを使用すると、サーバー・アプリケーション・オブジェクトを配置するための柔軟性が最大限になるだけでなく、パフォーマンスが向上します。

ただし、TX MOFW 分離ハイブリッド・グローバル・ポリシーは、TX MOFW 結合ハイブリッド・グローバル・ポリシーが、複数のオブジェクトに関連するアプリケーションに対して提供するほど、パフォーマンスの向上を提供しません。TX MOFW 結合ハイブリッド・グローバルに関して、サーバーは、ほとんどのオブジェクトに対して、同じハイブリッド・グローバル・トランザクションを使用することができます。これとは対照的に、TX MOFW 分離ハイブリッド・グローバル・ポリシーの場合、サーバーは、オブジェクト・メソッド要求ごとに、現行のハイブリッド・グローバル・トランザクションを中断して、新規のトランザクションを作成しなければならないので、サーバーにさらにオーバーヘッドがかかります。

ビジネス・オブジェクトが、トランザクションを開始する `begin` を最初に発行するように設計されている場合、オブジェクトは、最初のオブジェクト状態を保持するために、DB2 データベースなどの保護リソースにアクセスするようにコンテナーに対して要求することはできません。この場合、サーバーは、ハイブリッド・グローバル・トランザクションを自動的に開始して、パーシスタント・データ・ストアから初期状態を取り出します。サーバーが一度ハイブリッ

ド・グローバル・トランザクションを開始すると、サーバーはオブジェクトがトランザクションのコンテキストを管理することを許可しません。

TX MOFW サポート結合ハイブリッド・グローバル・コンテナー・ポリシーを使用する場合の規則

TX 必須以外のポリシーに関して、WebSphere for z/OS MOFW サーバーは、RRS およびその他の関連するリソース・マネージャーがグローバルなトランザクションを管理できるようにします。ただし、この場合、サーバーは、クライアント・アプリケーションのトランザクションのコンテキストを尊重します。クライアントにグローバル・トランザクションがある場合は、同じサーバー・インスタンスで実行されるメソッドはすべて、その同じグローバル・トランザクションを共用します。図11 は、この初期トランザクション・コンテキストを示しています。

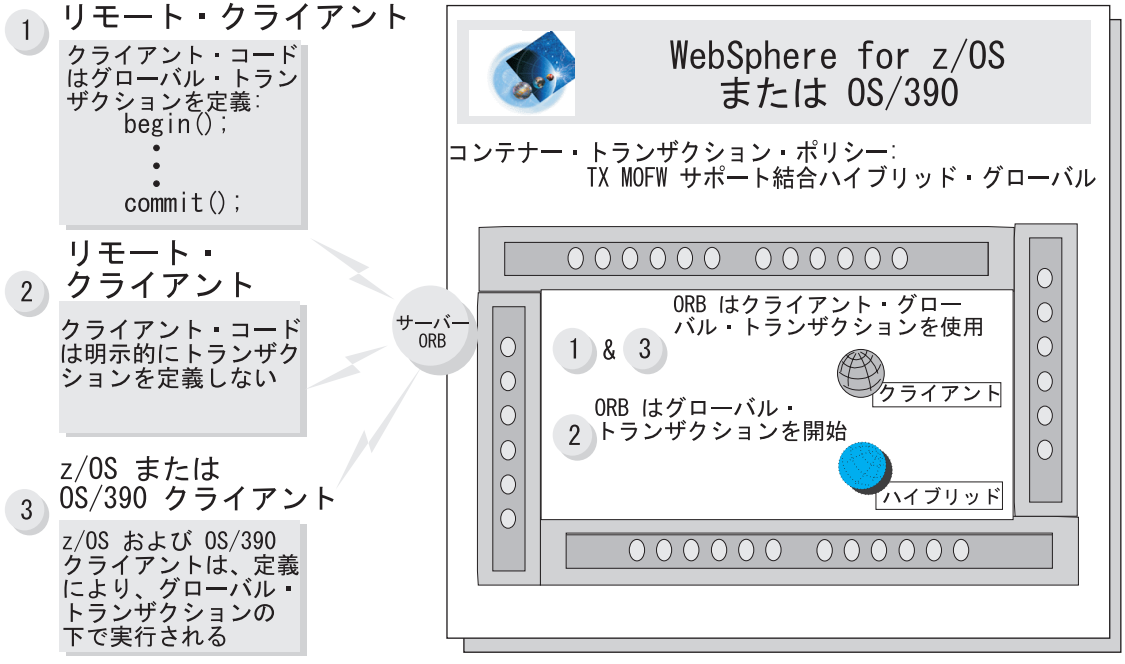

図 11. TX MOFW サポート結合ハイブリッド・グローバル コンテナー・ポリシーの WebSphere for z/OS MOFW サーバー・トランザクション環境

サーバー・アプリケーションが使用するオブジェクトはすべて、同じサーバー・インスタンスで実行されるように構成されていなければなりません。サーバー・アプリケーションの追加規則は、以下のとおりです。

- グローバル・トランザクションの下で実行されることを前提としている。

- そのトランザクション・コンテキストを管理しようとはしない。

CORBA アプリケーションおよびクライアント・アプリケーションの制約事項

- サーバー・アプリケーションで z/OS または OS/390 言語環境プログラム CEESETL 呼び出し可能サービスは使用しないでください。このサービスは、C 言語関数 setlocale() と類似していて、グローバル・ロケール操作環境を確立します。サーバー・アプリケーションは、WebSphere for z/OS がそのアプリケーション・サーバーに設定するグローバル・ロケール環境を更新する場合、結果が予測不能になります。
- z/OS または OS/390 でクライアント・アプリケーションを実行しようとする場合、IBM は、WebSphere for z/OS MOFW サーバーが使用しているものと同じコード・ページ EBCDIC IBM 1047 を使用するように、これらのクライアントを設計することを強くお勧めします。その他の EBCDIC コード・ページでも稼動するかもしれませんが、結果は予測不能になります。

  注:

  1. この制約事項は、MOFW サーバーで実行中のアプリケーションを使用するクライアントのみに適用されます。
  2. z/OS または OS/390 以外のプラットフォームで実行するクライアント・アプリケーションは、任意のコード・ページを使用することができます。

CORBA アプリケーションから最適パフォーマンスを取得するためのガイドライン

- サーバー・アプリケーションが使用するデータ・ストアと同じ、ネットワーク内のシステムにビジネス・アプリケーションのサーバー部分をインストールします。このように、z/OS または OS/390 データ・ストアを必要とするビジネス・アプリケーションの場合、WebSphere for z/OS サーバーにサーバー部分を配置します。
- ビジネス・アプリケーション・サーバーの場合に必要となる可能性があるアドレス・スペース数を考慮して、ロード・ライブラリーをリンク・パック域 (LPA) に配置します。IBM は、これを手助けするために parmlib を提供しています。

z/OS または OS/390 にアプリケーション成果物を生成するワークステーション・ツールの設定に関するバックグラウンド

開発中の CORBA アプリケーションのタイプによって、1 つまたは複数の以下のワークステーション・ツールを使用して、コンポーネント・オブジェクトを開発し、それらのソース・コードを生成します。WebSphere for z/OS サーバーでコンポーネントを開発する場合は、以下のとおり、ワークステーション・ツールが適切に設定されていることを確認してください。

VisualAge for Java

必ず、以下のことを行ってください。

- 「**ウィンドウ**」→「**オプション**」と選択します。「**ビジュアル・コンポジション**」を選択して、「**bean スーパークラスの BeanInfo の継承**」をクリアします。
- 作成している bean に適切なフィーチャーを追加します。個々のサンプル命令には、追加する正しいフィーチャーがリストされています。

オブジェクト・ビルダー

新規モデルを始動する場合は、「**プラットフォーム (Platform)**」を選択して、以下のものを設定します。

- 「**表示 (View)**」は 390
- 「**生成 (Generate)**」は 390
- 「**制約 (Constrain)**」は 390

CORBA アプリケーションに必要なコンポーネント・オブジェクトに関するバックグラウンド

CORBA アプリケーションは、コンポーネント・オブジェクトの集合で構成され、そのいくつかは、VisualAge for Java またはオブジェクト・ビルダーを使用して、アプリケーションを開発する際に取った選択内容によって自動的に生成されます。必要なコンポーネント・オブジェクトの数およびタイプは、32ページの表2 で示されるように、開発中のサーバー・アプリケーションのタイプによって決まります。

表 2. CORBA サーバー・アプリケーションのためのコンポーネント・オブジェクトの要約

サーバー・アプリケーションの状況	必要となるコンポーネント・オブジェクト					
	手続き型アダプター・オブジェクト	ビジネス・オブジェクト	マネージド・オブジェクト	データ・オブジェクト	パーシスタント・オブジェクト	スキーマ
バックエンドのリソースをまったく使用しない		✔	✔			
CICS または IMS リソースを使用する	✔	✔	✔	✔	✔	✔
DB2 を使用する		✔	✔	✔	✔	✔

CORBA ビジネス・オブジェクトを C++ または Java で開発するためのガイドライン

WebSphere Application Server Component Broker 製品は、共通プログラミング・モデルを定義しているため、サーバー・アプリケーションの設計およびコーディングに関する情報は、ほとんど以下の資料に出てきます。

- *Component Broker プログラミングの手引き*
- *Component Broker 上級プログラミングの手引き*
- *Component Broker プログラミング解説書*
- *Component Broker アプリケーション開発ツールの手引き*

上記の資料では、Component Broker ビジネス・オブジェクトを C++ または Java で設計およびコード化するために理解しておく必要がある概念、コーディング演習、プログラミング・インターフェース、およびツールについて定義しています。上記の資料は、z/OS または OS/390 プラットフォームに適用されない概念またはインターフェースについても記載しています。

CORBA サーバー・アプリケーションは、Component Broker クライアントとして機能する場合もあるので注意してください。つまり、WebSphere for z/OS MOFW サーバーで実行するようコード化したビジネス・オブジェクトは、それ自身が以下のことを実行できます。

- 任意の Component Broker プラットフォームにある他のビジネス・オブジェクトの検索または作成、使用、および削除。

- Java で作成されている場合は、WebSphere for z/OS J2EE サーバー で実行されているエンタープライズ bean の検索または作成、使用および削除。

共通 Component Broker プログラミングの資料を使用する以外にも、開発するサーバー・アプリケーションのタイプに応じて以下のトピックの情報に精通しておく必要があります。

- 『IMS リソースを使用する CORBA アプリケーションの開発のためのガイドライン』
- 42ページの『CICS リソースを使用する CORBA アプリケーションの開発のためのガイドライン』

IMS リソースを使用する CORBA アプリケーションの開発のためのガイドライン

開発する任意の CORBA サーバー・アプリケーションには、コンポーネント、すなわちビジネス・オブジェクトとそれに関連したクラス、および手続き型アダプター (PA) bean とそれに関連したクラスが必要です。PA bean の開発は、おそらく IMS リソースにアクセスするサーバー・アプリケーションの作成で、最も難易度の高い部分です。bean クラスの作成には、IMS および IMS の環境で実行されるトランザクションに関する知識が必要です。

WebSphere for z/OS は、手続き型アプリケーション・アダプターを介して IMS リソースにアクセスします。1 つは、オープン・トランザクション・マネージャー・アクセス (OTMA) を使用して、IMS と通信するもので、もう 1 つは、MVS の拡張プログラム間通信コンポーネント (APPC/MVS) を使用するものです。これらのアダプターは、WebSphere for z/OS MOFW アプリケーション・サーバーが IMS サブシステムと通信できるようにします。PA bean は、IMS との通信に使用するアダプターに入力および出力できるようにします。インストールについては、おそらく、OTMA が使用できる IMS をインストールしておくか、IMS に対して APPC 接続を確立しておきます。インストールで使用する構成を決定するには、システム・プログラマーに確認してください。

次の表は、IMS リソースにアクセスするサーバー・アプリケーションを開発するためのサブタスクおよび関連するガイドラインを示しています。

サブタスク	関連した手順 (参照項目)
特定の IMS 手続き型アプリケーション・アダプターの場合のアプリケーションの設計方法について	• 34ページの『IMS-OTMA アダプターの設計のためのガイドライン』 • 34ページの『IMS-APPC アダプターの設計のためのガイドライン』

サブタスク	関連した手順 (参照項目)
IMS プロセスの要求方法について	36ページの『IMS 要求処理に関するバックグラウンド』
PA bean およびそれに関連するクラスについて	• 37ページの『IBM VisualAge Java with EAB による PAO およびそれに関連するクラスのコーディングに関するバックグラウンド』 • 38ページの『入力 / 出力ユーザー・データを定義する COBOL ファイルを作成するためのステップ』 • 40ページの『PA bean マッパーおよび情報クラスのコーディングに関するバックグラウンド』 • 41ページの『PA bean コマンド・クラスのコーディングに関するバックグラウンド』

IMS-OTMA アダプターの設計のためのガイドライン

IMS のターゲット・トランザクション・プログラムと通信するときは、SendReceive 要求だけを使用することができます。WebSphere for z/OS は、IMS トランザクション・プログラムと送信専用または受信専用処理を行う要求をサポートしていません。

IMS-APPC アダプターの設計のためのガイドライン

MOFW サーバー・アプリケーションが IMS リソースにアクセスするには、インストール時に、WebSphere for z/OS サーバーを IMS-APPC 論理リソース・マッピング (LRM) が可能となるように定義します。この構成の場合、サーバーは、IMS サブシステムとの通信の方法として APPC/MVS を使用します。つまり、サーバーは、APPC/MVS 会話を IMS サブシステムと共に割り振ります。

IMS-APPC LRM を定義する場合は、インストール時に APPC 同期 (sync) レベルを選択します。これは、通信で使用する APPC/MVS 会話のタイプを決定します。サーバーが割り振る会話はすべて、保護リソースまたは無保護リソースのいずれかです。会話のタイプ、保護または無保護は、以下の 2 つの要因によって決まります。

- 35ページの表3 でアウトラインを示すように、アプリケーションの処理のタイプ。

- IMS-APPC LRM が接続されるコンテナー (複数も可) のトランザクション・ポリシー。

推奨: パフォーマンスを向上させるには、保護会話を要求するのは、必要な場合のみにするように、CORBA サーバー・アプリケーションを設計します。おそらく、ユーザーのアプリケーションは、たとえば、データを読み込んで、さらに再読み込みして、そのデータを操作し、結果を保管するので、表3 にある 2 つのカテゴリーの一方には、うまく適合しないでしょう。この場合、サーバー・アプリケーションを、読み取り専用処理用と、読み取り更新および保管処理用の 2 つのオブジェクトを使用するように設計することもできます。このように設計した場合、各オブジェクトは、同じ IMS-APPC LRM に接続される独自のコンテナーを保有していて、各コンテナーのトランザクション・ポリシーは、サーバーが保護または無保護会話のいずれを使用しているのかを検出します。

表 3. APPC/MVS 会話タイプおよび適切なサーバー・アプリケーション・プロセス

APPC/MVS 会話タイプ	最適なアプリケーション・プロセス
保護会話。 保護会話の場合、APPC/MVS、WebSphere for z/OS MOFW サーバー、IMS、およびその他のシステム・コンポーネントが共に機能して、確実に、分散リソースに対するアプリケーションの更新が調整され、アトミック・オペレーションとして処理されます。つまり、アプリケーションの更新については、すべて更新される (コミットされる) か更新されない (ロールバックされる) かのどちらかです。	データ保全性に対するアプリケーション・プロセスが重要です。たとえば、サーバー・アプリケーションが、当座預金と普通預金の間の振替処理を行う場合、これらの振替処理は、勘定残高の整合性を保つために単一操作であると考えられます。
無保護会話。 無保護会話の場合、アプリケーションの処理の調整に関連するオーバーヘッドが発生しないため、アプリケーション・パフォーマンスは速くなります。ただし、会話エラーまたは障害が発生した場合、アプリケーションで使用するリソースは不整合状態になる可能性があります。	パフォーマンスに関するアプリケーション・プロセスは、データ保全性よりも重要です。たとえば、サーバー・アプリケーションが、データを取り出すために読み取り専用要求を実行して、一定の状態のデータ値には依存しない場合、そのアプリケーションは、データ保全性には依存しないので、この環境にさらに適合することになります。

もう 1 つの設計ガイドラインでは、サーバー・アプリケーションが IMS トランザクション・プログラムに送信する要求のタイプを考慮します。IMS のター

ゲット・トランザクション・プログラムと通信するときは、送受信要求だけを使用することができます。WebSphere for z/OS は、IMS トランザクション・プログラムと送信専用または受信専用処理を行う要求をサポートしていません。

IMS 要求処理に関するバックグラウンド

IMS がトランザクションを実行する要求を受け取ると、要求されるトランザクション名**と 1 つのブランク**を指定した 8 バイト・フィールドを使用して、その要求に対する入力メッセージを作成します。このアクションにより、IMS の 3270 端末およびエミュレーターをご使用の元来のお客様に問題が生じることはありません。ただし、この振る舞いによって、要求された IMS トランザクションに渡したいユーザー・データはふぞろいになる可能性があります。37ページの図12 を見ると分かるように、IMS トランザクションの名前の長さが 7 文字以外であると、以下の問題が発生します。

- トランザクション名が 7 文字未満の場合、IMS は、8 バイトの名前フィールドの残りのバイトをユーザー・データの頭のバイトから埋めていきます。
- トランザクション名が 7 文字よりも多い場合、IMS は、IMS トランザクションがユーザー・データの開始位置を検出できるオフセットにブランクを入れます。

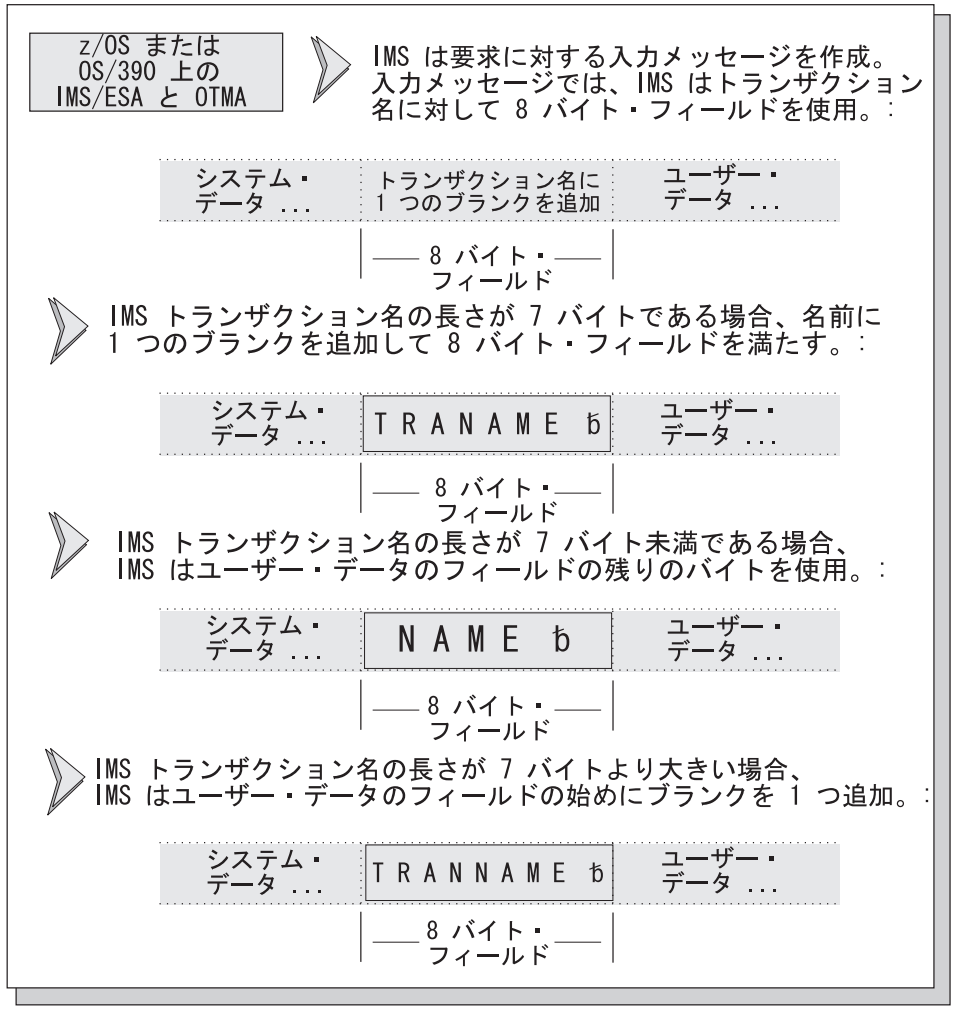

図 12. IMS が WebSphere for z/OS MOFW アプリケーション・サーバーから要求を受け取る方法

いずれの場合も、結果は予測どおりにはなりません。ユーザー・データに適当な分だけ埋めこみを行うことによって、7 バイトに満たないトランザクション名の場合に発生するふぞろいを簡単に避けることができます。埋め込みの計算の手順については、38ページの『入力 / 出力ユーザー・データを定義する COBOL ファイルを作成するためのステップ』で説明しています。

IBM VisualAge Java with EAB による PAO およびそれに関連するクラスのコーディングに関するバックグラウンド

IMS または CICS 手続き型アプリケーション・アダプターを使用する CORBA アプリケーションを作成する場合、最初のステップは、IBM VisualAge Java

with EAB を使用して、手続き型アダプター (PA) bean を作成することです。次に、この PA bean をオブジェクト・ビルダーにインポートして、サーバー・アプリケーションのその他のコンポーネント・オブジェクトを作成します。PA bean は、8ページの図5 に示しているように、いくつかのクラスから構成されています。この図から以下のことが分かります。

- 手続き型アダプター・オブジェクト (PAO) は、ビジネス・オブジェクトの基本的状態 (つまり、パーシスタント記憶装置に配置されるユーザー・データ) を表しています。
- 入力および出力情報クラスは、IMS または CICS トランザクションを支えるデータの構造体およびコンテンツを定義します。
- PA bean のマッパー・クラスにより、PAO と入力または出力情報クラスの間のデータ転送が可能になります。

入力 / 出力ユーザー・データを定義する COBOL ファイルを作成するためのステップ: 独自のサーバー・アプリケーション用の PA bean を作成する場合、既存の IMS トランザクションの通信 (またはデータ) 域を見て、その領域の構造体とコンテンツを定義する COBOL ファイルを作成する必要があります。VisualAge for Java は、このファイルを使用して、入力情報および出力情報クラスについてコードを正しく生成します。これらのクラスは、PA bean の重要なエレメントで、ビジネス・オブジェクトの基本的状態 (つまり、パーシスタント記憶装置に配置されるユーザー・データ) を、IMS トランザクションが使用できるフォーマットで提示する機構を提供します。

推奨: 多くの IMS トランザクションは、入力と出力の両方に同じデータ域を使用します。そのため、フィールド名および特性は、入力も出力も両方とも同じです。ただし、PA bean の場合、入力用フィールドの集合を定義し、出力用フィールドに別の集合を別の名前で定義する必要があります。入出力に同じフィールド名を使用すると、VisualAge for Java は、ユーザーが定義している 2 番目の集合に対して固有のフィールド名を自動的に生成します。VisualAge for Java を使用して、固有の名前を生成することができますが、その場合は、これらの名前を覚えておいて、PA bean の insert()、retrieve()、update()、および del() メソッドのために作成したコードでこれらの名前を使用するのを忘れないようにしてください。

始める前に: IMS トランザクションのソース・コード、特に、トランザクションが入出力用に使用する通信域を定義しているコードにアクセスする必要があります。

以下のステップを実行して、VisualAge for Java にインポートする COBOL ファイルを作成します。

1. 独自の COBOL ファイルを作成するか、サンプル COBOL ファイルをワークステーション上の作業ディレクトリーにコピーします。

---

2. 使用したい IMS トランザクション名が、7 バイト未満の場合、IMS がユーザー・データを適当に位置合わせするのに必要な埋め込みバイト数を計算します。埋め込みを計算するには、以下のようにします。
   - ソース・コードを見て、トランザクション名に使用されている入力フィールドのサイズを検索します (通常、このフィールドは、実際のトランザクション名に合わせるために必要な長さよりも長くなります)。
   - 以下のアルゴリズムを使用して、埋め込みバイト数を計算します。

     ```
     Size_of_tran_name_field - (size_of_tran_name + 1) = size_of_fill
     ```

     たとえば、使用したいトランザクションの名前が PCTIA であり、トランザクション名の入力フィールドのサイズが 8 バイトであるとします。この場合の計算は、以下のとおりです。

     ```
     8 - (5 + 1) = 2
     ```

     この例では、ユーザー・データの適当な位置合わせのために 2 バイトの埋め込みを定義する必要があります。

---

3. 作業ディレクトリーの COBOL ファイルを編集して、ユーザー・データおよび、必要に応じて、必要な埋め込みのための COBOL レコード定義を追加します。以下のサンプルは、PL/1 で作成した IMS トランザクションのデータ域に必要な COBOL レコード定義を示しています。埋め込みは入力データには必要ですが、出力データには不要です。

PL/1 データ構造体	**使用する COBOL レコード定義**
入力データ構造体	

`DCL 1 IN_MESSAGE STATIC,`
`  2 LL  BIN FIXED(31),`
`  2 ZZ  CHAR(2),`
`  2 TRANID CHAR(8),`
`  2 ITEMID  CHAR(6),`
`  2 PRICE  CHAR(6);`
	`01 INPUT-MSG.`
	`  02 INFILL PICTURE X(2).`
	`  02 IN_ITEMID  PIC X(6).`
	`  02 IN_PRICE  PIC X(6).`

出力データ構造体	

PL/1 データ構造体	使用する COBOL レコード定義
DCL 1 OUT_MESSAGE BASED..., 2 LL BIN FIXED(31), 2 ZZ CHAR(2), 2 DATE CHAR(19), 2 ITEMID CHAR(6), 2 PRICE PICTURE'999V.99', 2 ERRMSG CHAR(78), 2 ERRMSG2 CHAR(78);	01 OUTPUT-MSG. 02 OUT_DATE PIC X(19). 02 OUT_ITEMID PIC X(6). 02 OUT_PRICE X(6). 02 OUT_ERRMSG PIC X(78). 02 OUT_ERRMSG2 PIC X(78).

トランザクションのデータ域のユーザー・レコード・フィールドごとに、COBOL レコード定義を作成するときに、実行することは分かっています。このファイルをインポートするときに、VisualAge for Java は、使用する PA bean に対する入出力情報クラスを正しく生成します。

PA bean マッパーおよび情報クラスのコーディングに関するバックグラウンド: マッパー・クラスは、PAO 属性を入力または出力情報クラスの対応するフィールドに接続する必要があります。そのために、VisualAge for Java を使用してマッパー・クラスを作成する場合、実際には、PAO 属性だけを定義して、各属性から入力および出力情報クラスの対応するフィールドへのリンクを作成することになります。

情報およびマッパー・クラスを作成できるようにするには、既存の IMS または CICS トランザクションの通信 (またはデータ) 域を見て、その領域の構造体とコンテンツを定義する COBOL ファイルを作成する必要があります。VisualAge for Java は、このファイルを使用して、入力情報および出力情報クラスについてコードを正しく生成します。作成する必要のあるクラス数は、以下のように、既存の IMS または CICS トランザクションの機能によって決まります。

IMS または CICS トランザクションの機能	作成されるもの
IMS または CICS トランザクションは、単一の通信域を使用して、入出力の両方を処理します。	• 入出力両方の構造体およびコンテンツを定義している 1 レコードを含む COBOL ファイル。 • COBOL ファイル・レコードと一致する単一情報クラス。 • 情報クラスと一致する単一マッパー・クラス。 **重要:** IMS または CICS トランザクションで単一の通信域を使用していても、入力および出力情報クラスを別々に作成していて、入出力両方に同じフィールド名を使用した場合、VisualAge for Java は、ユーザーが定義している 2 番目の集合に固有のフィールド名を自動的に生成します。VisualAge for Java を使用して、固有の名前を生成することもできますが、その場合は、これらの名前を覚えておいて、PAO の insert()、retrieve()、update()、および del() メソッドのために作成したコードでこれらの名前を使用するのを忘れないようにしてください。
IMS または CICS トランザクションは、入力用に 1 つと出力用に 1 つの別の通信域を使用します。	• 構造体およびコンテンツを入出用に定義しているレコードと出力用に定義している別のレコードから成る 2 レコードを含む COBOL ファイル。 • 入力用の COBOL ファイル・レコードに一致するクラスと、出力用のレコードに一致する別のクラスから成る 2 つの情報クラス。 • 入力情報クラスに一致するクラスと、出力情報クラスに一致する別のクラスから成る 2 つのマッパー・クラス。 **推奨:** 入力および出力情報クラスを別に作成する場合は、必ず、以下のように選択してパフォーマンスを向上させてください。 • 出力情報クラスのレコード・スタイルを指定するときは、「**カスタム・レコード**」を選択してください。 • 出力情報クラスの「**生成時に通知 (Generate with notification)**」オプションを選択解除してください。

PA bean コマンド・クラスのコーディングに関するバックグラウンド: PA bean コマンド・クラスは、IMS との単一の相互作用をラップします。つまり、IMS に送信したり、サーバーが IMS から受信した結果をフォーマットするために、これらのクラスは、WebSphere for z/OS アプリケーション・サーバーの要求をパッケージ化します。VisualAge for Java を使用して、コマンド・クラスを作成する場合は、いくつかのキーから成る情報を指定します。

相互作用スペック

IMSOTMAInteractionSpec または IMSAPPCInteractionSpec

コネクター

com.ibm.connector.CB390.IMSOTMA.InteractionSpec または
com.ibm.connector.CB390.IMSAPPC.InteractionSpec

CICS リソースを使用する CORBA アプリケーションの開発のためのガイドライン

IMS または CICS 手続き型アプリケーション・アダプターを使用する CORBA アプリケーションを作成する場合、最初のステップは、IBM VisualAge Java with EAB を使用して、手続き型アダプター (PA) bean を作成することです。次に、この PA bean をオブジェクト・ビルダーにインポートして、サーバー・アプリケーションのその他のコンポーネント・オブジェクトを作成します。PA bean は、8ページの図5 に示しているように、いくつかのクラスから構成されています。この図から以下のことが分かります。

- 手続き型アダプター・オブジェクト (PAO) は、ビジネス・オブジェクトの基本的状態 (つまり、パーシスタント記憶装置に配置されるユーザー・データ) を表しています。
- 入力および出力情報クラスは、IMS または CICS トランザクションを支えるデータの構造体およびコンテンツを定義します。
- PA bean のマッパー・クラスにより、PAO と入力または出力情報クラスの間のデータ転送が可能になります。

PA bean マッパーおよび情報クラスのコーディングに関するバックグラウンド

マッパー・クラスは、PAO 属性を入力または出力情報クラスの対応するフィールドに接続する必要があります。そのために、VisualAge for Java を使用してマッパー・クラスを作成する場合、実際には、PAO 属性だけを定義して、各属性から入力および出力情報クラスの対応するフィールドへのリンクを作成することになります。

情報およびマッパー・クラスを作成できるようにするには、既存の IMS または CICS トランザクションの通信 (またはデータ) 域を見て、その領域の構造体とコンテンツを定義する COBOL ファイルを作成する必要があります。VisualAge for Java は、このファイルを使用して、入力情報および出力情報クラスについてコードを正しく生成します。作成する必要のあるクラス数は、以下のように、既存の IMS または CICS トランザクションの機能によって決まります。

IMS または CICS トランザクションの機能	作成されるもの
IMS または CICS トランザクションは、単一の通信域を使用して、入出力の両方を処理します。	• 入出力両方の構造体およびコンテンツを定義している 1 レコードを含む COBOL ファイル。 • COBOL ファイル・レコードと一致する単一情報クラス。 • 情報クラスと一致する単一マッパー・クラス。 **重要:** IMS または CICS トランザクションで単一の通信域を使用していても、入力および出力情報クラスを別々に作成していて、入出力両方に同じフィールド名を使用した場合、VisualAge for Java は、ユーザーが定義している 2 番目の集合に固有のフィールド名を自動的に生成します。VisualAge for Java を使用して、固有の名前を生成することができますが、その場合は、これらの名前を覚えておいて、PAO の insert()、retrieve()、update()、および del() メソッドのために作成したコードでこれらの名前を使用するのを忘れないようにしてください。
IMS または CICS トランザクションは、入力用に 1 つおよび出力用に 1 つと、別の通信域を使用します。	• 構造体およびコンテンツを入出用に定義しているレコードと出力用に定義している別のレコードから成る 2 レコードを含む COBOL ファイル。 • 入力用の COBOL ファイル・レコードに一致するクラスと、出力用のレコードに一致する別のクラスから成る 2 つの情報クラス。 • 入力情報クラスに一致するクラスと、出力情報クラスに一致する別のクラスから成る 2 つのマッパー・クラス。 **推奨:** 入力および出力情報クラスを別に作成する場合は、必ず、以下のように選択してパフォーマンスを向上させてください。 • 出力情報クラスのレコード・スタイルを指定するときは、「**カスタム・レコード**」を選択してください。 • 出力情報クラスの「**通知による生成 (Generate with notification)**」オプションを選択解除してください。

PA bean コマンド・クラスのコーディングに関するバックグラウンド

PA bean コマンド・クラスは、CICS との単一の相互作用をラップします。つまり、CICS に送信したり、MOFW サーバーが CICS から受信した結果をフォーマットするために、このクラスは、WebSphere for z/OS アプリケーション・サーバーに対する要求をパッケージ化します。VisualAge for Java を使用して、コマンド・クラスを作成する場合は、以下のうちのいくつかのキー項目を指定します。

相互作用スペック

CICSEXCIInteractionSpec

コネクター

com.ibm.connector.CB390.CICSEXCI.InteractionSpec

プログラム名

pgmName フィールドに、CICS 領域で実行したい 8 文字のプログラム名を入力します。このプログラム名は、CICS 領域にインストールされているかユーザー作成の自動インストール・プログラムに定義されている、事前定義された PROGRAM リソース定義で指定しなければなりません。

トランザクション ID

CashAcct サンプルでは、**transID** フィールドが NULL であることを確認しなければなりません。これは、CICS 処理にデフォルトの振る舞いを指定します。このデフォルトの振る舞いに関して、**pgmName** フィールドに指定したプログラムは、CICS 提供のミラー・トランザクション、すなわち CSMI の下で実行されます。ただし、独自のサーバー・アプリケーション用に PA bean を開発している場合は、別のミラー・トランザクションの下でプログラムを実行させてみたい場合があります。その場合は、transID の指定要件について *CICS External Interfaces Guide*, SC34-5709 を参照してください。

コンテナー作成のバックグラウンド

オブジェクト・ビルダーを使用して WebSphere for z/OS サーバーに配置するサーバー・アプリケーションを開発するときは、コンテナーをアプリケーションに関連付ける必要があります。オブジェクト・ビルダーを使用して、コンテナー名のみを提供します。コンテナーの特性を定義するには、WebSphere for z/OS 管理アプリケーションを使用する必要があります。

注: この制限は、WebSphere for z/OS MOFW サーバーのみに適用されます。J2EE アプリケーション (エンタープライズ bean、サーブレット、および JavaServer Pages) のみがインストールされる WebSphere for z/OS J2EE サーバーには、コンテナー定義は必要ありません。J2EE サーバーは、所定の J2EE アプリケーションまたはその個々のコンポーネントの配置記述子に設定された属性を使用します。J2EE サーバーの詳細については、*WebSphere Application Server V4.0 for z/OS and OS/390: J2EE アプリケーションのアセンブル*, SA88-8654 を参照してください。

ガイドライン: オブジェクト・ビルダーを使用して z/OS または OS/390 プラットフォームに既存のアプリケーションを構成する場合、さらにそのアプリケ

ーションがすでにデフォルト・コンテナーに関連付けられている場合は、必ず新規のコンテナーを定義して、そのデフォルトを置換してください。デフォルト・コンテナーはワークステーション・プラットフォーム上の Component Broker サーバーでの使用のみを目的として設計されています。WebSphere for z/OS によって、これらのデフォルト・コンテナーに関連付けられたアプリケーションを配置することができますが、作成されたランタイム環境は、ワークステーション・プラットフォームのものとは異なる場合があります。

WebSphere for z/OS 管理アプリケーションを使用した MOFW サーバーに対するコンテナー定義について詳しくは、89ページの『MOFW サーバーに対するコンテナーの定義』を参照してください。

第3章 z/OS または OS/390 での CORBA アプリケーションのアセンブル

CORBA サーバー・アプリケーションの移植可能なプラットフォーム固有のコンポーネントをアセンブルする場合、プログラマーはワークステーション上で生成されたソース・ファイルを、WebSphere for z/OS MOFW サーバーで実行できる実行可能コードにコンパイルします。

プログラマーがこの作業を完了するには、ワークステーションと z/OS または OS/390 アプリケーション開発の両方についての専門知識が必要です。特に、以下のようなことが必要になります。

- ワークステーションでの CORBA アプリケーション・コンポーネントの開発に必要なプロセス、ツール、および出力についての理解。
- ワークステーションと z/OS または OS/390 間のファイル転送のためのソフトウェア・プロダクトについて精通していること。
- UNIX システム・サーバー (USS) 環境での作業に精通していること。この中には、階層ファイル・システム (HFS) の使用、環境変数の設定、および `make` コマンドを使用したコードのコンパイルなどが含まれます。

次の表は、CORBA アプリケーションのアセンブルのためのサブタスク、および関連情報を示しています。

サブタスク	関連情報 (参照項目)
z/OS または OS/390 でのアセンブリー環境の設定	48ページの『アプリケーション開発環境をセットアップするためのステップ』
ワークステーション上で生成された CORBA アプリケーション・ファイルでの作業	• 55ページの『ワークステーションからサーバー・アプリケーション・ファイルの HFS ディレクトリー構造を作成するためのステップ』 • 57ページの『ワークステーションから z/OS または OS/390 にファイルを転送するためのステップ』

サブタスク	関連情報 (参照項目)
z/OS または OS/390 で CORBA アプリケーション・ファイルをコンパイルするための準備	• 59ページの『サーバー・アプリケーションに対して実行可能コードを配置する場所の決定に関するバックグラウンド』 • 63ページの『CORBA アプリケーションの実行可能コードに対するデータ・セットの割り振りに関するバックグラウンド』 • 65ページの『make 処理に関するバックグラウンド』
CORBA アプリケーションの実行可能コードへのコンパイル	77ページの『z/OS または OS/390 で CORBA アプリケーションのソース・ファイルをコンパイルするためのステップ』
CORBA アプリケーションを使用するための準備	• 80ページの『CORBA アプリケーションをシステム検索パスに追加するためのステップ』 • 81ページの『CORBA アプリケーションにデータ・オブジェクトをバインドするためのステップ』

アプリケーション開発環境をセットアップするためのステップ

ユーザーのサイトのシステム・プログラマーが、WebSphere for z/OS をインストールする際、オプションとして z/OS または OS/390 UNIX アプリケーション開発環境をセットアップすることもできます。 *WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652 で得られる手順をここにリストしているので、正しい環境を自己検証することができます。必要であれば、z/OS または OS/390 UNIX およびワークステーション・アプリケーション環境の両方の場合の、そのソフトウェア製品のバージョンおよびその他の要件について、*WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652 を調べてください。

始める前に: Windows NT および OS/390 UNIX システムを構成しなければなりません。

アプリケーション開発環境をセットアップするには、以下のステップに従ってください。

1. Windows NT の場合

a. アプリケーション開発者ごとに、Component Broker for Windows NT ランタイムおよび開発環境をインストールします。インストールの手順については、*Component Broker Quick Beginnings*, G04L-2375 を参照してください。

b. IBM では、各アプリケーション開発者のワークステーションに、NFS クライアントまたはそれと同等のものをインストールすることをお勧めします。代わりに、MKS の SAMBA クライアントを使用することもできます。

c. パーソナル・コミュニケーションズ /3270 (または同等のホスト・エミュレーター・ソフトウェア) をインストールします。

---

2. z/OS または OS/390 の場合

   a. Java JDK が構成されていることを確認してください。

   b. C++ コンパイラーが使用可能であることを確認してください。

   c. NFS サーバーまたはそれと同等のものをインストールします。代替サーバーは、MKS からの SAMBA サーバーです。

   d. アプリケーション開発者ごとに、少なくとも 100 シリンダーの HFS スペースをホーム・ディレクトリーに割り振ります。z/OS または OS/390 UNIX HFS ストレージを保守するのと同様に、アプリケーション開発ストレージを保守することができます。

   e. アプリケーション開発者ごとに、z/OS または OS/390 UNIX の使用が許可される TSO/E のユーザー ID があることを確認してください。

   f. 定義されているファイル記述子ファイルの数を検査してください (BPXPRMxx parmlib メンバーの MAXFILEPROC ステートメント)。プログラムをコンパイルするときに、追加のファイル記述子ファイルが必要となる場合があります。

   g. 許可されているプロセスの最大数を検査してください (BPXPRMxx parmlib メンバーの MAXPROCUSER ステートメント)。make の実行時に限度まで追加する必要がある場合があります。

   h. 各アプリケーション開発者の領域サイズを検査してください (BPXPRMxx の MAXASSIZE、または RACF ADDUSER または ALTUSER コマンドの ASSIZEMAX)。経験的に言えるのは、可能な限り最大の領域サイズで実行することです。しかし、最初は最小サイズ (256MB) で開始します。サイズは IEFUSI 出口、JES2 EXIT06、JES3 IATUX03、または TSO セグメントのデフォルトで制限される可能性が

あります。コンパイラーのメモリーが不足している場合、アプリケーション開発者の領域サイズを増やす必要がある場合があります。

BPXPRMxx について、詳しくは、*z/OS UNIX システム・サービス 計画*, GA88-8639 を参照してください。

---

3. z/OS または OS/390 の場合は、`/usr/lpp/WebSphere390/CB390/samples` の CB390make.env ファイルを検査してください。以下の表をもとに、処理内容を判断してください。

条件	処理	注
指定されたディレクトリーに WebSphere for z/OS をインストールしている。	CB390make.env に対する変更はありません。	
指定されたディレクトリー以外のディレクトリーに WebSphere for z/OS をインストールしている。	以下のいずれかを行います。 a. CB390make.env をコピーして、カスタマイズする。 b. .profile ファイルで CB390_ENVFILE 変数により新規位置を識別する。たとえば、以下のように指定する。 `export CB390_ENVFILE=/etc/CB390make.env`	IBM が CB390make.env に対してメンテナンスを実施した場合、ユーザーのコピー・バージョンに対してメンテナンス変更が必要な場合があります。
	あるいは、以下のことを行います。 .profile ファイルにより、51ページの表4 で説明している環境変数をオーバーライドする。	IBM が CB390make.env に対してメンテナンスを実施した場合、ユーザーの .profile ファイルの export ステートメントに対して変更が必要な場合があります。

---

4. z/OS または OS/390 の場合、WebSphere for z/OS ファイルの位置としてシステム全体のデフォルト・プロファイル (/etc/profile) を設定します。

51ページの表4 は、WebSphere for z/OS 用に開発したすべての CORBA アプリケーションで使用する環境変数について説明しています。

表 4. MOFW コンポーネント開発者用の汎用環境変数

変数	注
CB390_ENVFILE=*path*	CB390make.env ファイルの位置。 デフォルト: `/usr/lpp/WebSphere390/CB390/samples/CB390make.env`
CB390_STDINC=*path*	include ファイルの位置。 デフォルト: `/usr/include //'CBC.SCLBH.+'`
CLASSPATH=*path*	**MOFW C++ クライアントについては、**何も指定する必要はありません。
	MOFW Java クライアントについては、`ws390crt.jar` を指定します。
	サーバーの Java ビジネス・オブジェクトについては、`ws390srt.jar` を指定します。
LIBPATH=*path*	`/usr/lpp/WebSphere390/CB390/lib` を組み込むように LIBPATH 環境変数を変更します。
IVB_DRIVER_PATH=*path*	WebSphere for z/OS 製品ファイルの位置。 デフォルト: `/usr/lpp/WebSphere390/CB390/`
PATH=*path*	`/usr/lpp/WebSphere390/CB390/bin` を組み込むように PATH 環境変数を変更します。
SBBOEXEC_DSN=*DATA_SET_NAME*	SBBOEXEC の REXX EXEC の位置。 デフォルト: `BBO.SBBOEXEC`

表4 は、WebSphere for z/OS ファイルを指し示す環境変数のデフォルト値をリストしています。ユーザーのサイトのシステム・プログラマーが WebSphere for z/OS を IBM の指定以外のディレクトリーにインストールした場合、/etc/profile をすべての z/OS または OS/390 シェル・ユーザーに WebSphere for z/OS ファイルの位置が識別できるように変更している可能性があります。このシステム・プログラマーが、WebSphere for z/OS ファイルの位置を設定するために、このシステム全体のデフォルト・プロファイルを更新しなかった場合、製品ファイルの検索位置を検出する必要があり、$HOME/.profile ファイルまたはシェル・スクリプトを使用して、それに応じて、これらの環境変数の値を変更します。

推奨: 表4 にある環境変数が /etc/profile に設定されていない場合は、$HOME/.profile ファイルに設定してください。これらの環境変数は、WebSphere for z/OS 用に開発したすべての CORBA アプリケーションでも同じ設定でなければなりません。

例: 53ページの図13 のサンプル例は、/etc/profile にコピーされる可能性のあるシステム全体の変数を定義しています。このサンプルは、51ページの表4 の環境変数を設定するだけでなく、C++ コンパイラーを正しく操作するのに必要な設定を含んでいます。このサンプル /etc/profile は、IBM 提供の /samples/profile のバリエーションです。これについては、*z/OS UNIX システム・サービス 計画*, GA88-8639 で詳しく説明しています。

C++ カスタマイズ・セクションに関する注記:

1. C++ カスタマイズ・セクションの環境変数は、動的に割り振られるデータ・セットの名前の一部など、c89/cc/c++ ユーティリティーに対する情報を提供します。
2. コンパイラー、ランタイム・ライブラリー製品、またはその両方のインストールで異なる値を使用する場合は、適切なエクスポート行を正しい値に設定します。c89/cc/c++ ユーティリティーでは VOL=SER= パラメーターをサポートしていないので、c89/cc/c++ が使用する名前付きデータ・セットは、すべてカタログ化しなければならないことに注意してください。
3. c89/cc/c++ のデフォルト (SYSDA) がインストール済みシステムに対して定義されていない場合は、名前なし作業データ・セットに対するデフォルトの選出ユニットをオーバーライドしなければならない可能性があります。NULL ("") 値を指定して、c89/cc/c++ によりインストール定義されたデフォルトが使用できるようになります。
4. c89 コマンド変数だけを明示的に示します。cc および c++ 変数は、このカスタマイズ・セクションの最後において `eval` で始まるコマンド行により設定されます。
5. これは、c89/cc/c++ の振る舞いに影響を与える環境変数の完全なリストではありません。ただし、これは、一般的に、システム・プログラマーによるカスタマイズを必要とする可能性があるすべての環境変数です。

   推奨: マイグレーションを簡単にするには、c89/cc/c++ ユーティリティーの正しい操作のための変数だけを設定します。

```
# To enable and disable lines in this profile you may remove or add  '#'
# to uncomment or comment the desired lines.
#
# Export the values so the system will have access to them.
#
# Improve the shell's performance for users from ISPF or with
# STEPLIB data sets allocated.  This performance improvement is not
# applicable to non-interactive shells, for example those started with
# the BPXBATCH and OSHELL utilities.
if [ -z "$STEPLIB" ] && tty -s;
then
    export STEPLIB=none
    exec sh -L
fi
#
# Set the time zone as appropriate.
TZ=EST5EDT
export TZ
#
# Set the language
LANG=C
export LANG
#
# Set a default command path, including your current working
# directory.
# PATH=/bin:.:/usr/lpp/java/J1.3/bin
PATH=/bin:.:/usr/lpp/java/J1.3/bin:/usr/lpp/WebSphere390/CB390/bin
export PATH
#
# Specify the directory to search for a DLL (Dynamic Link Library)
# filename. If not set, the working directory is searched. In the
# sample below, db2_install_path is the HFS where you installed DB2 for OS/390.
LIBPATH=/db2_install_path/lib:/usr/lpp/java/J1.3/bin:/usr/lpp/java/J1.3/bin/classic:
    /usr/lpp/WebSphere390/CB390/lib
export LIBPATH
#
# Set the path for NLS files (message catalogs).
NLSPATH=/usr/lib/nls/msg/%L/%N
export NLSPATH
#
```

図 13. サンプル /etc/profile

```
# Set the path for man pages (help files).
MANPATH=/usr/man/%L
export MANPATH
#
# Set the name of the system mail file and enable mail notification.
MAIL=/usr/mail/$LOGNAME
export MAIL
#
# Set the default file creation mask
umask 022
#
# Set the LOGNAME variable readonly so it is not accidentally modified.
readonly LOGNAME
#
# ====================================================================
# Start of c89/cc/c++ customization section
#
# High-Level Qualifier "prefixes" for data sets used by c89/cc/c++:
#
#   C/C++ Compiler:
#   ---------------------------------------
export _C89_CLIB_PREFIX="SYS1.CPP"
#
#
#   Prelinker and run-time library:
#   ---------------------------------------
export _C89_PLIB_PREFIX="SYS1.LEMVS"
#
#
#   OS/390 system data sets:
#   ---------------------------------------
export _C89_SLIB_PREFIX="SYS1"
#
#
# Esoteric unit for data sets:
#
#
#   Unit for (unnamed) work data sets:
#   ---------------------------------------
export _C89_WORK_UNIT="SYSALLDA"
#
#
# Commands to propogate c89 environment variables for cc and c++:
# ================================================================
#
eval "export $(typeset -x | grep "^_C89_" | awk '{sub("_C89_","_CC_");printf
   "%s ",$0}')"
eval "export $(typeset -x | grep "^_C89_" | awk '{sub("_C89_","_CXX_");printf
   "%s ",$0}')"
#
# End of c89/cc/c++ customization section
# ====================================================================
#
```

図 14. サンプル /etc/profile の続き

```
# ====================================================================
# Start of WebSphere for z/OS customization section
#
# Provide the name of the WebSphere for z/OS environmental file
export CB390_ENVFILE=/usr/lpp/WebSphere390/CB390/samples/CB390make.env
#
# Provide system names for the include libraries, override taken.
export CB390_STDINC="/usr/include //'SYS1.CPP.SCLBH.+'"
#
# Provide the root structure for WebSphere for z/OS tree, default taken,
export CB390_ROOT=
#
# Provide the name for the REXX exec, override taken.
export SBBOEXEC_DSN=CB390.CB11002.SBBOEXEC
#
# Provide the path to the bin/obmdll20.mk, default taken
export IVB_DRIVER_PATH=/usr/lpp/WebSphere390/CB390/
#
#
# End of CB390 customization section
# ====================================================================
```

図 15. サンプル /etc/profile の続き

ワークステーションからサーバー・アプリケーション・ファイルの HFS ディレクトリー構造を作成するためのステップ

IBM がアプリケーション開発環境に推奨する製品の 1 つが NFS です。これを使用すると、ワークステーション上のローカル・ドライブとして HFS にアクセスすることができます。NFS またはそれと同等の製品を使用している場合、ワークステーション・ツールで作成するファイルを自動的に OS/390 に配置することができ、そのディレクトリー構造は、ワークステーションで使用されている構造とまったく同じになります。このような製品を使用していない場合は、ディレクトリー構造を判別して、ワークステーションにファイルを作成した後、それらのファイルを手動で転送しなければなりません。手動転送の手順は、57ページの『ワークステーションから z/OS または OS/390 にファイルを転送するためのステップ』にあります。

おそらく、最も簡単な方法は、ワークステーション上の作業ディレクトリーの構造と一致する作業ディレクトリー構造を使用することです。たとえば、Windows NT でオブジェクト・ビルダーを使用している場合、NT 上の作業ディレクトリーには、オブジェクト・ビルダーにより生成された DDL ファイルのサブディレクトリーと共に、オブジェクト・ビルダーにより生成されたソー

ス・ファイルが含まれます。ソース・ファイルおよび DDL のサブディレクトリーを含むように、HFS ディレクトリーをパターン化することもできます。サーバー・アプリケーション・ファイルの HFS ディレクトリー構造を作成するには、以下のステップを実行します。

1. サーバー・アプリケーションの複数のファイルに対して 1 つの HFS ディレクトリー構造を選択します。これらのファイルには、以下のものが含まれます。
   - 開発ツールによりワークステーションに作成する、アプリケーション (コンポーネント) オブジェクトのソース・ファイル。
   - サーバー・アプリケーション・ファミリーの構造ならびにそのホームおよびコンテナーを定義する複数のデータ定義言語 (DDL) ファイル。

---

2. 選択する構造に関係なく、以下の規則を守ってください。
   - ディレクトリー構造には、`PRODUCTION` サブディレクトリーがなければなりません。z/OS または OS/390 上のコードをコンパイルするときに、make プロセスは、make コマンドを入力するディレクトリーに `PRODUCTION` サブディレクトリーがあることを想定しています。このサブディレクトリーがないと、make プロセスは生成した出力内容を配置する場所が分かりません。
   - DDL ファイルのディレクトリー名は、アプリケーション・ファミリーの名前と正確に一致しなければなりません。たとえば、サーバー・アプリケーション・ファミリーに `CashAcctAppFam` という名前を使用すると、以下の例のように完全修飾ディレクトリー名を使用することができます。

     `/u/userid/CashAcctAppFam` または
     `/u/userid/Working390/CashAcctAppFam`
   - 手続き型アプリケーション・アダプターと共に使用する、Enterprise Java Bean またはサーバー・アプリケーションを開発している場合、z/OS または OS/390 に配置する必要がある JAR ファイルがあります。これらの JAR ファイルを、その他のサーバー・アプリケーションのソース・ファイルで使用しているディレクトリーと同じディレクトリーに配置しても、これらのファイルのために、別にディレクトリーまたはサブディレクトリーを作成しても構いません。

---

3. u/userid/ ディレクトリーから、必要に応じて mkdir コマンドを入力し、使用したいディレクトリー構造を作成します。mkdir コマンドについて詳しくは、*z/OS UNIX システム・サービス コマンド解説書*, SA88-8641 を参照してください。

ディレクトリー構造の作成が完了すると、ワークステーションから z/OS または OS/390 にファイル転送することができます。手順については、『ワークステーションから z/OS または OS/390 にファイルを転送するためのステップ』を参照してください。

ワークステーションから z/OS または OS/390 にファイルを転送するためのステップ

ワークステーション上のローカル・ドライブとして、z/OS または OS/390 階層ファイル・システム (HFS) にアクセスできる製品を使用していない場合は、ワークステーションから z/OS または OS/390 にサーバー・アプリケーション・ファイルを手動で転送する必要があります。ソフトウェア・プログラムには、ワークステーションから z/OS または OS/390 に、さらに簡単にファイルを送信する方法を提供するものもあり、ASCII から EBCDIC への変換に注意すれば、それらのプログラムをどれでも使用することができます。ここに提示されている手順は、ftp を使用して、z/OS または OS/390 上の HFS にファイル転送することを前提としています。

始める前に: z/OS または OS/390 上でファイルを配置する位置を決めます。それについては、55ページの『ワークステーションからサーバー・アプリケーション・ファイルの HFS ディレクトリー構造を作成するためのステップ』を参照してください。ファイルを使用して作業する場合の詳細なバックグラウンド情報については、*z/OS UNIX システム・サービス ユーザーズ・ガイド*, SA88-8640 のファイル・システムのトピックを参照してください。

MS-DOS で ftp を使用して、Windows NT から z/OS または OS/390 上の HFS にファイルを手動で転送するには、以下の手順を行います。これらの手順は、1 つの ftp セッションですべてのファイルを転送することを前提としていますが、そうしなくてもかまいません。ftp セッションを確立してから、ftp プロンプト (ftp>) から以下のコマンドを入力してください。

1. オブジェクト・ビルダー出力用に使用したディレクトリー (たとえば、x:) から、ftp -i *target-machine* コマンドを使用して、ターゲット HFS を稼動しているマシンとの間に ftp セッションを開始します。

注:

a. 引き数 `-i` は ftp のファイル・プロンプトをオフにするので、一度に多くのファイルを容易に移動することができます。これは、オブジェクト・ビルダーのソース・ファイルの転送には適しています。

b. 転送のデフォルト・モードは ASCII です。このモードは、オブジェクト・ビルダーのソース・ファイルと DDL ファイルの両方に適したモードです。JAR ファイルの転送にはいずれも、バイナリー・モードを使用します。モードを切り替えるには、ftp プロンプトで `ascii` または `bin` のいずれかを入力します。

---

2. オブジェクト・ビルダーのソース・ファイルを転送するには、以下のようにします。

   a. `cd` *`directory-path`* コマンドを使用して、オブジェクト・ビルダーのソース・ファイルを配置しようとする HFS 作業ディレクトリーに移動します。

   b. `lcd` *`directory-path`* コマンドを使用して、オブジェクト・ビルダーのソース・ファイルを含む NT サブディレクトリーに移動します。

   c. コマンド `mput *` を使用して、すべてのファイルを転送します。

---

3. DDL ファイルを転送するには、以下のようにします。

   a. `cd` *`directory-path`* コマンドを使用して、DDL ファイルを含む HFS ディレクトリーに移動します。

   b. `lcd` *`directory-path`* コマンドを使用して、オブジェクト・ビルダー生成の DDL ファイルを含む NT サブディレクトリーに移動します。

   c. `mput *` コマンドを使用して、NT サブディレクトリー内のすべてのファイルを転送します。

---

4. 手続き型アプリケーション・アダプターと共に使用する CORBA アプリケーションを開発している場合、JAR ファイルも転送することになります。複数の JAR ファイルがあるか、JAR ファイルに対して複数の位置 (サブディレクトリー) がある場合、以下のステップを繰り返す必要がある場合があります。JAR ファイルを転送するには、以下のようにします。

   a. `bin` コマンドを使用して、JAR ファイルの転送の適切なモードを設定します。

b. cd *directory-path* コマンドを使用して、JAR ファイルを配置しようとする HFS 作業ディレクトリーに移動します。

c. lcd *directory-path* コマンドを使用して、JAR ファイルを含む NT サブディレクトリーに移動します。

d. put *filename*.jar コマンドを使用して、JAR ファイルを転送します。

e. ファイル転送後、完全修飾ファイル名 (ディレクトリー・パスおよびファイル名。たとえば、/u/*userid*/jarfiles/CICSEXCI/BeCash.jar) を、環境変数で使用しているデータ・セットのメンバーの CLASSPATH ステートメントに追加します。そのデータ・セットの論理レコード・サイズでは、CLASSPATH ステートメントに対して完全名を追加できない場合、CLASSPATH ステートメントにすでにリストされているディレクトリーに JAR ファイルをコピーして、そこで JAR ファイルを配置してください。JAR ファイルを配置するには、jar -xvf *filename*.jar コマンドを使用します。

---

5. quit または bye コマンドを使用して、ftp セッションを終了します。

サーバー・アプリケーションに対して実行可能コードを配置する場所の決定に関するバックグラウンド

WebSphere for z/OS MOFW サーバーに配置するために、サーバー・アプリケーションをアセンブルしている場合、以下のように、実行可能コードを配置するために make プロセスを指示する場所だけでなく、最も効率的なランタイム・パフォーマンスが得られるように、実行可能ファイルを配置する場所も考慮する必要があります。

- CORBA アプリケーションをコンパイルするときに、区分データ・セット (PDS)、拡張区分データ・セット (PDSE)、または階層ファイル・システム (HFS) に実行可能コードを配置するために、make プロセスを指示することができます。PDS は 16 MB 以下のロード・モジュールだけを含むことができるので、選択は、たいてい PDSE または HFS の 2 つのオプションの間で行われます。
- サーバーにアプリケーションを配置するときに、実行可能コードの位置を識別するために、以下のいずれかの方法を使用することができます。
  - HFS におけるコードの場合、LIBPATH 環境変数を使用するか、リンク・パック域 (LPA) に HFS ファイルをロードすることができます。

– PDS または PDSE におけるコードの場合、WebSphere for z/OS MOFW サーバーに対して JCL プロシージャーで STEPLIB DD ステートメントを使用するか、LPA にデータ・セットをロードするか、リンク・リストにデータ・セットを追加することができます。

ランタイム用にコードを配置する場所は、システムおよびアプリケーションのパフォーマンス、および仮想記憶域の使用に影響を与えます。たとえば、サーバーがアプリケーションを実行できるようにするには、システムはアプリケーションの実行可能コードを検出し、記憶域にロードしなければなりません。システムは、以下のコード検索順序を使用します。コードが検索順序の上に行くに従って、システム・パフォーマンスは良くなります。

1. LIBPATH 変数
2. STEPLIB DD ステートメント
3. LPA
4. リンク・リスト

アプリケーションのコードの最適な場所を決定するには、コードが使用される頻度、およびインストール時にその製品レベルの WebSphere for z/OS MOFW サーバーを使用する方法を知っておく必要があります。

コードをコンパイルするだけならば、必ずしもランタイムの配置を考慮する必要はありませんが、追加の作業 (HFS ファイルを PDSE にコピーしたり、その逆を行ったりするなど) を避けることができるように、ランタイム・オプションについて知っている必要があります。以下の推奨と 62ページの表5 を参照してください。この表は、各配置オプションの利点と欠点を要約しています。必要に応じて、システムの検索順序にモジュールを配置したり、システムおよびアプリケーションのパフォーマンスについての配置の効果に関する詳細な説明は、*z/OS MVS 初期設定およびチューニング ガイド*, SA88-8563 を参照してください。

推奨:

- インストールで共用 HFS 機能 (z/OS バージョン 1 リリース 1、および OS/390 バージョン 2 リリース 9 以降で使用可能) を使用する場合は、HFS に実行可能コードを配置してください。この選択は、サーバー・アプリケーションが、シスプレックス内で複数システム間に分散されている場合、または、すべてのコードが 1 個所にあるため、サーバー・アプリケーションに Java コードが含まれている場合に特に便利です。Java クラス・ファイルは、HFS になければなりません。

  HFS に実行可能コードを配置することにした場合、コードを作業ディレクトリーではなく、HFS ディレクトリーに配置することができます。そのように選択した場合、`make` ユーティリティーを使用する前に、コンパイラー・オプ

ションで出力ファイル名を指定しなければなりません。そうしなければ、コンパイラーは作業ディレクトリーに出力を配置します。

- インストール内容に z/OS または OS/390 共用 HFS 機能が含まれていない場合、実行可能コードに HFS を使用することを引き続き考える必要があります。この選択は、すべてのコードが 1 個所にあるため、サーバー・アプリケーションに Java コードが含まれている場合に特に便利です。Java クラス・ファイルは、HFS になければなりません。

  z/OS または OS/390 共用 HFS 機能がなくても、すべてのユーザー・アプリケーション・コードを HFS に配置して、WebSphere for z/OS MOFW サーバーが稼動しているシステム間でファイル・システムを共用するほかの方法を使用することができます。たとえば、以下の方法を取ることができます。

  1. サーバー・アプリケーション用の Java クラス・ファイルおよび実行可能コードをすべて保持するファイル・システムを作成します。
  2. 1 つのシステムのサーバー・アプリケーションをコンパイルおよびテストします。
  3. サーバー・アプリケーションが実動状態になったら、WebSphere for z/OS MOFW サーバーが稼動するすべてのシステムのファイル・システムをマウントします。すべてのファイル・システムを読み取り専用モードでマウントするか、書き込みモードで 1 つ、読み取り専用モードでその他すべてをマウントします。
  4. ファイル・システムの DLL をリフレッシュするには、以下のようにします。
     a. マウントされたすべてのシステムからファイル・システムを取り外します。
     b. アプリケーションを使用する WebSphere for z/OS MOFW サーバーを静止します。
     c. DLL をリフレッシュします。
     d. 更新ファイル・システムを再マウントします。

  サーバー・アプリケーションのサイズおよび必要となる更新の頻度によっては、この方法は、z/OS または OS/390 共用 HFS 機能を使用するよりも管理が容易でパフォーマンスが向上する可能性があります。

表 5. CORBA アプリケーションの実行可能コードのコンパイルおよび実行時配置の要約

配置オプション	コンパイル時について	実行時について
HFS	HFS を使用すると、サーバー・アプリケーション・ファイルをすべて 1 個所に集めることができるので、特に便利です。アプリケーションに Java コードが含まれている場合、Java クラス・ファイルは HFS になければならないので注意してください。	**LIBPATH:** LIBPATH 環境変数を使用する場合、システムは、サーバー・インスタンス別に、各サーバー領域に対してアプリケーションの実行可能コードの 1 つのコピーを作成します。作成されたサーバー領域および複製されたサーバー・インスタンスの数によって、このオプションは、記憶域に関する制約を引き起こす場合があります。 また、アプリケーションのコピーを更新するには、更新を適用する前にアプリケーションを使用して、すべてのサーバーを静止しなければなりません。
		LPA: アプリケーションを LPA にロードする場合、システムは、アプリケーションの 1 つのコピーを共用仮想記憶域にロードします。この単一コピーは、同じシステムで稼動しているすべての WebSphere for z/OS MOFW サーバーで使用可能です。 LPA を使用すると、サーバー・インスタンスのレプリカを生成することが分かっているとき、またはアプリケーションに大きなワークロードが予測される場合に有利です。

表 5. CORBA アプリケーションの実行可能コードのコンパイルおよび実行時配置の要約 (続き)

配置オプション	コンパイル時について	実行時について
PDSE または **PDS**	PDS と比べると PDSE にはいくつかの利点があります。PDSE は、16 MB 以上のロード・モジュールを含むことができるだけでなく、モジュールの追加または削除の際に再編成または圧縮の必要がないため、管理が簡単です。 PDSE に実行可能コードを配置することにした場合、サーバー・アプリケーションをコンパイルする前に、データ・セットを割り振る必要があります。詳しくは、『CORBA アプリケーションの実行可能コードに対するデータ・セットの割り振りに関するバックグラウンド』を参照してください。	**STEPLIB:** STEPLIB DD ステートメントを使用する場合、システムは、サーバー・インスタンス別に、各サーバー領域に対してアプリケーションの実行可能コードの 1 つのコピーを作成します。これは、HFS と LIBPATH 変数を使用した場合の振る舞いと同じです。したがって、STEPLIB を使用することは、HFS および LIBPATH を使用することと同じです。ただし、PDSE は、シスプレックスにおいて複数システム間で共用できるため (共用 HFS サポートについて、z/OS バージョン 1 リリース 1、または OS/390 バージョン 2 リリース 9 以降がインストールされない限り)、複数システム間の分散は、代替オプションを使用する場合よりも簡単です。
		LPA: この場合、HFS および LPA の使用に関しても同じ考慮事項が適用されます。
		リンク・リスト: この場合、HFS および LIBPATH 変数の使用に関しても同じ考慮事項が適用されます。

CORBA アプリケーションの実行可能コードに対するデータ・セットの割り振りに関するバックグラウンド

データ・セットでサーバー・アプリケーションに対して実行可能コードをデータ・セットに配置することにした場合、または、サーバー・アプリケーションで DB2 リソースを使用する場合は、サーバー・アプリケーションをコンパイルする前に、複数の区分データ・セット (PDS) または PDSE を割り振らなければなりません。 64ページの表6では、コンパイル時環境変数設定など、これらのデータ・セットについての WebSphere for z/OS 固有の要件を一覧で示しています。コンパイル時の環境変数設定については、65ページの『make 処理に関するバックグラウンド』を参照してください。

データ・セットを割り振るには、バッチ・ジョブでの TSO/E 割り振りコマンド、ISPF、または JCL ステートメントなど、いくつかの方法を使用できます。追加情報については、必要に応じて以下の 1 つ以上の資料を参照してください。

- *z/OS TSO/E ユーザーズ・ガイド*, SA88-8638 では、割り振りコマンドまたは ISPF の使用方法について、データ・セットの割り振りに関する章で説明しています。
- *z/OS MVS JCL ユーザーズ・ガイド*, SA88-8570 では、データ・セット・リソースの割り振りに関する章で各種データ・セットの割り振り例を示しています。
- *z/OS UNIX システム・サービス ユーザーズ・ガイド*, SA88-8640 では、TSO/E コマンドまたは JCL を USS 環境で使用する方法を説明しています。
- *IBM DB2 適用業務プログラミングおよび SQL の手引き*, SC88-7377 では、DB2 リソースを使用するプログラムに関するバックグラウンド情報を提供しています。アプリケーション・プログラムの実行の準備に関する章を参照してください。ただし、プログラムのコンパイルに使用する make プロセスによって、自動的に DB2 プリコンパイラーが実行されることを考慮に入れてください。

表 6. サーバー・アプリケーション・データ・セットに対する要件

条件	必要な割り振り	注
HFS ではなくデータ・セットにサーバー・アプリケーションの実行可能コードを配置したい場合	以下の特性を備えたロード・ライブラリー・データ・セットが必要です。 LRECL=0 BLKSIZE=6144 RECFM=U DSORG=PO	make ユーティリティーを使用してアプリケーションをコンパイルする場合、以下の環境変数が設定されていることを確認してください。 NOHFSLNKOUT=1 LOADLIB=*data_set_name*

表 6. サーバー・アプリケーション・データ・セットに対する要件 (続き)

条件	必要な割り振り	注
サーバー・アプリケーションが DB2 データにアクセスする場合	DB2 支援オブジェクトごとに、以下の特性を備えた 1 つの DBRMLIB データ・セットが必要です。 LRECL=80 BLKSIZE=6160 RECFM=FB DSORG=PO	こうしたサーバー・アプリケーションの場合、make プロセスは、SQL ステートメントおよびソース・プログラムから抽出されたホスト変数情報、およびプログラムを識別して、変換されたソース・ステートメントに DBRM を結び付ける情報に対して、データベース要求モジュール (DBRM) を作成します。DBRM は、バインド・プロセスに対して入力になります。 DBRMLIB データ・セットに必要なスペースの決定に関する指針が必要な場合は、DBRM マッピング・マクロ DSNXDBRM を参照してください。これは、ライブラリー *prefix*.SDSNMACS にあります。ここで *prefix* は、インストール・システムが DB2 ライブラリー・データ・セットの識別に使用する名前を表します。 make ユーティリティーを使用してアプリケーションをコンパイルする場合、割り振り済みのデータ・セットを識別するように以下の環境変数を設定してください。 DBRMHLQ DBRMQUAL

make 処理に関するバックグラウンド

CORBA サーバー・アプリケーションをコンパイルするには、make ユーティリティーを使用します。これは、UNIX システム・サービス (USS) シェルを介して使用可能です。サーバー・アプリケーションの開発時に、オブジェクト・ビルダーは、いくつかの環境変数設定と、z/OS または OS/390 で C++ または Java コードをコンパイルおよびリンクするのに必要なオプションを含む make ファイルを生成します。ほとんどの場合、オブジェクト・ビルダー生成の make ファイルは、変更しなくても使用できるはずです。ただし、USS シェル環境で使用可能なさまざまなメソッドにより、特定のサーバー・アプリケーションの必要性に応じて、環境変数またはオプションを追加またはオーバーライドすることができます。

make ユーティリティーを使用する前に、以下のことを行う必要があります。

- z/OS または OS/390 でサーバー・アプリケーションをコンパイルするのに必要な設定を理解してください。アプリケーションのコンパイルのために設定できる環境変数のリストについては、68ページの表7 を参照してください。多くの変数には、WebSphere for z/OS 製品と共に出荷されたファイルにより提供されているデフォルト設定があるので、すべての変数を手動で設定する必要はないことに注意してください。ただし、インストール時のアプリケーション開発環境および開発中のサーバー・アプリケーションによっては、オーバーライドしたい設定がある場合があります。
- オブジェクト・ビルダーによるいくつかの環境変数およびオプションの設定方法を理解してください。これらのオプションを変更する必要はありませんが、変更しても構いません。たとえば、コンパイラー・リストを制御するオプションを変更したい場合があります。

  サーバー・アプリケーションの場合、オブジェクト・ビルダーは、`all.mak`、*xxx*`C.mak` および *xxx*`S.mak` ファイルという 3 つの make ファイルを生成します。これら 3 つの `.mak` ファイルは、`prjdefs.mk` ファイルを組み込んでいます。このファイルには、オブジェクト・ビルダーにより指定された作成オプションが含まれています。オブジェクト・ビルダーが使用するデフォルト設定およびオプションは、*Component Broker アプリケーション開発ツールの手引き* の構成トピックにリストされています。`prjdefs.mk` ファイルには、その他の z/OS または OS/390 以外のプラットフォームのコンパイラーに設定されているコンパイラー・オプションも含まれている可能性があります。これらの設定は無視することができます。

  xxx`C.mak` および *xxx*`S.mak` ファイルも `obmdll30.mk` ファイルを組み込んでいます。これは WebSphere for z/OS と共に出荷されます (make プロセスは、IVB_DRIVER_PATH 変数を介してこのファイルにアクセスします)。`obmdll30.mk` ファイルは、`CB390make.rules` および `CB390make.env` ファイルを使用します。`CB390make.help` ファイルには、上記のファイルの内容についての詳しい情報があります。
- システムが環境変数に設定する値を決定する方法を理解してください。シェル環境の設定は、以下の場所の 1 個所または複数個所に設定することができます。システムは、この場所をリストされている順に検索します。
  1. インストールで使用するセキュリティー製品のユーザー・プロファイル。
  2. /etc/profile ファイル。これは、すべての z/OS または OS/390 シェル・ユーザー用のシステム全体のファイルです。
  3. $HOME/.profile ファイル。これは、個別ユーザー用の個別設定ファイルです。
  4. ENV 環境変数で名前が付けられたファイル。
  5. シェル・コマンドまたはシェル・スクリプト。

1 つの環境変数が、以下のうちの 1 個所または複数個所で現れる場合、システムは、この検索順序で検出された最後の設定を使用します。たとえば、ある変数が /etc/profile とシェル・スクリプトに現れると、システムは、シェル・スクリプトの設定を使用します。

- ユーザーのサイトのシステム・プログラマーが、/etc/profile をすべての z/OS または OS/390 シェル・ユーザーに WebSphere for z/OS ファイルの位置が識別できるように変更しているかどうかを認識しておいてください。このシステム・プログラマーが、WebSphere for z/OS ファイルの位置を設定するために、このシステム全体のデフォルト・プロファイルを更新しなかった場合、製品ファイルの検索位置を検出する必要があり、$HOME/.profile ファイルまたはシェル・スクリプトを使用して、それに応じて、これらの環境変数の値を変更します。システム・プログラマーが、すべてのシェル・ユーザーに対する設定で /etc/profile を更新するために使用している可能性があるサンプル・プロファイルについては、48ページの『アプリケーション開発環境をセットアップするためのステップ』を参照してください。

ユーティリティーを実行するときに、make は、以下にリストされているリソース、ファイル、および環境設定を使用して、以下の処理を実行します。

1. インストール時にシステム・プログラマーによってセットアップされた、startup.mk ファイルを使用して、処理のためのデフォルト・ルールを検出します。
2. all.mak ファイルを使用して、完了させる処理を決定します。結果的に、make は、オブジェクト・ビルダー生成の *xxx*C.mak および *xxx*S.mak ファイルを使用して、それぞれ、クライアント側およびサーバー側のバインディングとヘッダーを作成します。
3. z/OS または OS/390 で IDL コンパイラーを使用して、インターフェース定義言語 (IDL) ファイルをコンパイルして C++ のパーツ (.cpp ファイル) にします。これらのパーツには、クライアント側とサーバー側のバインディングおよびヘッダーが含まれます。
4. サーバー・アプリケーションが DB2 データにアクセスする場合、DB2 プリコンパイラーを使用して、データベース要求モジュール (DBRM) を作成します。
5. z/OS または OS/390 で C++ コンパイラーを使用して、オブジェクト・デックを生成します。
6. サーバー・アプリケーションで Java クラスが必要な場合、z/OS または OS/390 で Java コンパイラーを使用して、Java クラス・ファイルを生成します。

7. バインダーとリンカーを使用して、エクスポート・ファイルならびにクライアントおよびサーバーのダイナミック・リンク・ライブラリー (DLL) を作成します。
8. サーバー・アプリケーションの一部に Java で作成されたコードが含まれている場合は、アーカイブ (JAR) ファイルを作成します。

表 7. z/OS または OS/390 で CORBA アプリケーションをコンパイルする場合に設定する環境変数

変数	CORBA アプリケーションのタイプ別の必須またはオプション		注
	C++	**Java**	
_CEE_PREFIX	オプション	オプション	z/OS または OS/390 言語環境プログラム・ファイルを含むデータ・セットのプレフィックスを指定します。この変数を設定するのは、システム・プログラマーが指定されたデータ・セットに z/OS または OS/390 言語環境プログラムをインストールしなかった場合です。
_CEE_CBC	オプション	オプション	C++ コンパイラーおよびクラス・ライブラリー・ファイルを含むデータ・セットを指定します。この変数を設定するのは、システム・プログラマーが指定されたデータ・セットに z/OS または OS/390 C++ コンパイラーをインストールしなかった場合です。
CB390_ENVFILE	オプション	オプション	CB390make.env ファイルの位置を指定します。**デフォルト:** /usr/lpp/WebSphere390/CB390/samples/CB390make.env. CB390make.env ファイルには、この表の多くの変数のデフォルト設定が含まれています。この変数を変更するのは、独自の CB390make.env の編集済みコピーを使用している場合のみです。

表 7. *z/OS* または *OS/390* で *CORBA* アプリケーションをコンパイルする場合に設定する環境変数 *(続き)*

変数	CORBA アプリケーションのタイプ別の必須またはオプション		注
	C++	Java	
CB390_ROOT	オプション	オプション	IVB_DRIVER_PATH が指定されていないときに、IVB_DRIVER_PATH を構成するためにシステムで使用する、プレフィックス・パスを指定します。**デフォルト:** デフォルトは設定されていません。IVB_DRIVER_PATH も CB390_ROOT も指定されていない場合、CB390_ROOT の値は ヌルで、IVB_DRIVER_PATH は /usr/lpp/WebSphere390/CB390 に設定されています。 この変数を変更するのは、システム・プログラマーが指定されたディレクトリーに WebSphere for z/OS をインストールしなかった場合です。
CB390_USR_CLASSPATH	適用不可	オプション	サーバー・アプリケーションにインクルードされるように、WebSphere for z/OS 以外の JAR ファイルを指定します。たとえば、手続き型アプリケーション・アダプターと共に使用するサーバー・アプリケーションを開発している場合、CLASSPATH 変数の前にこの変数を付加して、VisualAge for Java により作成されたサーバー・アプリケーション JAR ファイルをインクルードすることができます。**デフォルト:** WebSphere for z/OS は、この変数にデフォルト値を設定していません。
CB390_USR_CPPFLAGS	オプション	オプション	C++ コンパイラーに追加パラメーターを指定します。**デフォルト:** WebSphere for z/OS は、この変数にデフォルト値を設定していません。
CB390_USR_CPPSHELLFLAGS	オプション	オプション	C++ シェル・コマンドに追加スイッチを指定します。**デフォルト:** WebSphere for z/OS は、この変数にデフォルト値を設定していません。
CB390_USR_CXX_INCDIRS	オプション	オプション	C++ ヘッダーの検索に追加ディレクトリーを指定します。**デフォルト:** WebSphere for z/OS は、この変数にデフォルト値を設定していません。

表 7. z/OS または OS/390 で CORBA アプリケーションをコンパイルする場合に設定する環境変数 (続き)

変数	CORBA アプリケーションのタイプ別の必須またはオプション		注
	C++	Java	
CB390_USR_DLLFLAGS	オプション	オプション	DLL のリンクに追加パラメーターを指定します。**デフォルト:** WebSphere for z/OS は、この変数にデフォルト値を設定していません。
CB390_USR_EXEFLAGS	オプション	オプション	実行可能ファイル (つまり、メインプログラム) をリンクするために追加パラメーターを指定します。**デフォルト:** WebSphere for z/OS は、この変数にデフォルト値を設定していません。
CB390_USR_IDLC_INCLUDE	オプション	オプション	IDL ファイルの検索に追加ディレクトリーを指定します。**デフォルト:** WebSphere for z/OS は、この変数にデフォルト値を設定していません。
CB390_USR_PATH	オプション	オプション	追加の実行可能プログラムの検索パスを指定します。**デフォルト:** WebSphere for z/OS は、この変数にデフォルト値を設定していません。CB390_USR_PATH に値を指定する場合、WebSphere for z/OS は、この値を PATH 変数設定の前に付加します。
CB390_USR_PRLNKFLAGS	オプション	オプション	プリリンクに追加パラメーターを指定します (つまり、エクスポート・ファイルの作成)。**デフォルト:** WebSphere for z/OS は、この変数にデフォルト値を設定していません。
CB390_STDINC	オプション	オプション	WebSphere for z/OS に C++ ランタイム・ヘッダーの位置を指定します。**デフォルト:** /usr/include//'CBC.SCLBH.+' この変数を変更するのは、システム・プログラマーが指定されたディレクトリーに WebSphere for z/OS をインストールしなかった場合です。

表 7. *z/OS または OS/390 で CORBA アプリケーションをコンパイルする場合に設定する環境変数 (続き)*

変数	CORBA アプリケーションのタイプ別の必須またはオプション		注
	C++	Java	
CLASSPATH	オプション	必須	Java ビジネス・オブジェクトで使用するために、Java クラス・ファイルを指定します。**デフォルト:** デフォルト値は CB390make.env ファイルに設定され、すべての Java サーバー・アプリケーションに必要な以下のファイルをインクルードします。wszOSsrt.jar サーバー・アプリケーション用に作成した JAR ファイルをインクルードするには、この変数を使用するか、CB390_USR_CLASSPATH 変数を前に付加することもできます。
DBRMHLQ	オプション	オプション	DB2 プリコンパイラーがサーバー・アプリケーション用の DBRMLIB を配置する、MVS データ・セットの名前の上位修飾子を指定します。**デフォルト:** デフォルト値はユーザーのユーザー ID です。 この変数を指定するのは、サーバー・アプリケーションが、データを保管するために DB2 を使用している場合のみです。
DBRMQUAL	オプション	オプション	DB2 プリコンパイラーがサーバー・アプリケーション用の DBRMLIB を配置する、z/OS または OS/390 データ・セットの名前の中位修飾子を指定します。**デフォルト:** コンパイルする静的 SQL を含むサーバー DLL の名前。 この変数を指定するのは、サーバー・アプリケーションが、データを保管するために DB2 を使用している場合のみです。
IVB_BATCH_INCREMENTAL	オプション	オプション	**推奨:** 増分作成 (小さな変更を行ったあとの再作成) のパフォーマンスを向上させるために、1 に設定します。

表 7. z/OS または OS/390 で CORBA アプリケーションをコンパイルする場合に設定する環境変数 (続き)

変数	CORBA アプリケーションのタイプ別の必須またはオプション		注
	C++	Java	
IVB_BATCH_ PROCESS_ FACTOR	オプション	オプション	同時に (並行して) 作成するターゲットの最大数を指定します。**デフォルト:** ユーザーのプロジェクトの IDL ファイルの数。 **ヒント:** 大規模なプロジェクトでは多数の並行処理が必要となる場合があるため、使用されるプロセスの数を制御するために定数値を指定する必要が生じることがあります (システム・プログラマーは、BPXPRMxx parmlib メンバーの MAXPROCUSER ステートメントによりプロセスの最大数を設定します)。 **推奨:** 最適値はアプリケーションの平均サイズによって異なります。値 10 から開始して、必要に応じてこれを変更してください。
IVB_BUILD_ UNOPTIMIZE	オプション	オプション	最適化せずに C++ ソースをコンパイルするかどうかを指定します。この変数を 1 に設定すると、コンパイル時間が短縮されます。**デフォルト:** WebSphere for z/OS は、この変数にデフォルト値を設定していません。
IVB_BUILD_ VERBOSE	オプション	オプション	コンパイル時に生成するメッセージ量を指定します。**デフォルト:** オブジェクト・ビルダーは、デフォルト値を 1 に設定します。これは、詳細コンパイラー・メッセージの量を最大にします。
IVB_COMBINE_ SOURCE	オプション	オプション	ヘッダー・ファイルを複数回処理するかどうかを指定します。**デフォルト:** オブジェクト・ビルダーは、デフォルト値を 1 に設定します。これは、ヘッダー・ファイルの処理回数を 1 回だけにします。その結果、作成処理にかかる時間が短縮されます。
IVB_DRIVER_ PATH	必須	必須	WebSphere for z/OS 製品ファイルの位置を指定します。**デフォルト:** デフォルト値、`/usr/lpp/WebSphere390/CB390` が CB390make.env ファイルに設定されます。

表 7. z/OS または OS/390 で CORBA アプリケーションをコンパイルする場合に設定する環境変数 (続き)

変数	CORBA アプリケーションのタイプ別の必須またはオプション		注
	C++	Java	
IVB_OPTIMIZE	オプション	オプション	コードのインライン配置など、DLL を最適化してコンパイルするかどうかを指定します。**デフォルト:** オブジェクト・ビルダーは、デフォルト値を 1 に設定します (IVB_UNOPTIMIZE=0 と設定するのと同等)。 **推奨:** 0 に設定してください。
IVB_PAX_LIST	オプション	オプション	スペースを節約するために C++ リストを圧縮するかどうかを指定します。**例:** `xyz.cpp` 部分については、圧縮されたリストは `xyz.clst.Z` です。`xyz.clst.Z` リストを解凍するには以下のコマンドを使用します。 `pax -rf xyz.cpp.Z` 解凍の結果はリスト `xyz.clst` です。 **推奨:** リストを圧縮するには、1 に設定してください。
IVB_TRACE	オプション	オプション	オブジェクト・レベル・トレース・ツールにトレース・データを送信する機能を使用して DLL をコンパイルするかどうかを指定します。**デフォルト:** オブジェクト・ビルダーは、デフォルト値を 1 に設定します。
IVB_TRACE_ DEBUG	オプション	オプション	DLL をオブジェクト・レベル・トレース・ライブラリーにリンクするかどうかを指定します。**デフォルト:** オブジェクト・ビルダーは、デフォルト値を 1 に設定します。
IVB_UNOPTIMIZE	オプション	オプション	コードのインライン配置など、DLL を最適化してコンパイルするかどうかを指定します。**デフォルト:** オブジェクト・ビルダーは、デフォルト値を 1 に設定します (IVB_OPTIMIZE=0 と設定するのと同等)。 **推奨:** 0 に設定してください。

表 7. z/OS または OS/390 で CORBA アプリケーションをコンパイルする場合に設定する環境変数 (続き)

変数	CORBA アプリケーションのタイプ別の必須またはオプション		注
	C++	**Java**	
JAVA_COMPILER	適用不可	オプション	make プロセスで使用する必要がある Java コンパイラーを指定します。**デフォルト:** WebSphere for z/OS は、この変数にデフォルト値を設定していません。
JAVA_HOME	適用不可	オプション	Java 2 スタンダード版 (J2SE) ソフトウェア開発キット (SDK) をインストールするディレクトリーを指定します。**デフォルト:** WebSphere for z/OS は、この変数にデフォルト値を設定していません。

LIBPATH	必須 *	必須	HFS における DLL の検索パスを指定します。**デフォルト:** デフォルト値は /etc/profile ファイルに設定されているはずです。WebSphere for z/OS は、この変数にデフォルト値を設定していません。
			
			* 手続き型アプリケーション・アダプターを使用している場合、C++ アプリケーションに必須。この場合、Java アプリケーションについては、システム、WebSphere for z/OS、および Java DLL を指定します。たとえば、次のように入力します。
			
			`LIBPATH=/db2_install_path/lib`
			`:/usr/lpp/java/J1.3/bin`
			`:/usr/lpp/java/J1.3/bin/classic`
			`:/usr/lpp/WebSphere390/CB390/lib`
			
			この場合、*db2_install_path* は DB2 for OS/390 をインストールした HFS です。

表 7. *z/OS または OS/390 で CORBA アプリケーションをコンパイルする場合に設定する環境変数 (続き)*

変数	CORBA アプリケーションのタイプ別の必須またはオプション		注
	C++	Java	
LOADLIB	必須 *	必須 *	make プロセスがアプリケーション用の DLL を配置する z/OS または OS/390 データ・セットの名前を指定します (Java アプリケーションをコンパイルする場合、make は DLL をこのデータ・セットに配置し、JAR ファイルを HFS に配置します)。**デフォルト:** WebSphere for z/OS は、この変数にデフォルト値を設定していません。 * NOHFSLNKOUT 変数を 1 に設定している場合、つまり、DLL を HFS ではなくデータ・セットに配置したい場合のみ必須。
NOHFSLNKOUT	オプション	オプション	システムが DLL を HFS または z/OS または OS/390 データ・セットのいずれに配置するかを指定します。以下の値のいずれかを設定することができます。 • 0 は HFS にリンク・エディット出力を送信する。 • 1 は LOADLIB 変数により指定されたデータ・セットにリンク・エディット出力を送信する。 **デフォルト:** 0
PATH	オプション	オプション	実行したいコマンドを含むファイルに対する検索パスを指定します。**デフォルト:** デフォルト値は /etc/profile ファイルに設定されているはずです。WebSphere for z/OS は、その変数を指定する場合、まず IVB_DRIVER_PATH/bin を PATH ステートメントの前に付加してから、CB390_USR_PATH を前に付加して、/etc/profile 設定を変更します。
SBBOEXEC_DSN	オプション	オプション	WebSphere for z/OS REXX EXEC のあるデータ・セットを指定します。**デフォルト:** 'BBO.SBBOEXEC' この変数を変更するのは、システム・プログラマーが指定されたディレクトリーに WebSphere for z/OS をインストールしなかった場合です。

表 7. z/OS または OS/390 で CORBA アプリケーションをコンパイルする場合に設定する環境変数 (続き)

変数	CORBA アプリケーションのタイプ別の必須またはオプション		注
	C++	Java	
STEPLIB	必須 *	必須 *	アプリケーションで必要な追加データ・セットを指定します。**デフォルト:** デフォルト値は /etc/profile ファイルに設定されているはずです。WebSphere for z/OS は、この変数にデフォルト値を設定していません。 * 以下の条件が両方とも真である場合に必須。 • アプリケーションで DB2 を使用している。 • DB2 プリコンパイラー・データ・セット SDSNLOAD および SDSNEXIT が、システム・リンク・リストにない。

必要であれば、詳細について、以下のソースを参照してください。

トピック	参照する資料
USS シェル環境	*z/OS UNIX システム・サービス ユーザーズ・ガイド*, SA88-8640 は、USS シェル環境および環境変数を使用した作業について説明しています。
make ユーティリティー	• make の使用に関するチュートリアルについては、*z/OS UNIX システム・サービス プログラミング・ツール*, SA88-8643。 • オブジェクト・ビルダーが使用するデフォルト・オプションについては、*Component Broker アプリケーション開発ツールの手引き*。 • `make` コマンド・フォーマットおよびオプション、make ファイルの使用上の注意、およびその他の参照情報については、*z/OS UNIX システム・サービス コマンド解説書*, SA88-8641。
IDL	• インターフェース定義言語およびその構文についての説明は、*Component Broker プログラミングの手引き*。 • `idlc` コマンド、および WebSphere for z/OS IDL コンパイラーのオプションおよび構文については、193ページの『付録C. インターフェース定義言語 (IDL) コンパイラー』。

トピック	参照する資料
DB2 プリコンパイラー	*IBM DB2 適用業務プログラミングおよび SQL の手引き*, SC88-7377 には、特定のサーバー・アプリケーションに対して変更が考えられるプリコンパイラー・オプションおよびデフォルト値がリストされています。
C++	• 特定のサーバー・アプリケーションに対して変更が考えられる (c89 コマンド記述の下の) コンパイラー・オプションについては、*z/OS UNIX システム・サービス コマンド解説書*, SA88-8641。 • z/OS または OS/390 C++ コンパイラーおよびそのオプションについての概要は、*z/OS C/C++ ユーザーズ・ガイド*, SC88-8850。
z/OS または OS/390 における Java	http://www.s390.ibm.com/java/

z/OS または OS/390 で CORBA アプリケーションのソース・ファイルをコンパイルするためのステップ

CORBA サーバー・アプリケーション用のワークステーション・ファイルが z/OS または OS/390 の作業ディレクトリーにある場合、make ファイルを実行して、WebSphere for z/OS MOFW アプリケーション・サーバーでサーバー・アプリケーションを実行するのに必要な実行可能コードおよびバインディングを作成することができます。

ワークステーションでサーバー・アプリケーションの成果物を生成するときに、オブジェクト・ビルダーは、クライアント DLL make ファイル、サーバー DLL make ファイル、および `all.mak` ファイルの 3 つの make ファイルを作成しています。個別の DLL make ファイルではなく、`all.mak` ファイルを使用して、サーバー・アプリケーション用のコードをコンパイルします。`all.mak` ファイルを使用して、DLL が正しい順序で作成されていることを確認してください。`all.mak` ファイルには、クライアントおよびサーバー DLL 用にオブジェクト・ビルダーで指定した、任意の IDL コンパイル、Java コンパイル、CPP コンパイル、およびリンク・オプションも組み込まれます。

始める前に:

- サーバー・アプリケーションの実行可能コードを配置したい場所を決定します。推奨内容については、59ページの『サーバー・アプリケーションに対して実行可能コードを配置する場所の決定に関するバックグラウンド』を参照してください。
- サーバー・アプリケーションで必要なデータ・セットをすべて割り振っていることを確認してください。データ・セットの割り振りの手順については、63ページの『CORBA アプリケーションの実行可能コードに対するデータ・セットの割り振りに関するバックグラウンド』を参照してください。
- make 処理の機能、および環境変数の設定が make 処理に与える影響を必ず理解してください。make 処理および環境変数の設定についての詳細は、65ページの『make 処理に関するバックグラウンド』を参照してください。

サーバー・アプリケーションのソース・ファイルをコンパイルするには、以下のステップを実行します。

1. コンパイル・コードに適切な環境変数が設定されていることを確認してください。設定が必要な環境変数には、WebSphere for z/OS 製品ファイルの位置を識別し、サーバー・アプリケーションの実行可能コードを配置したい場所を識別する環境変数が含まれています。

   環境変数を設定するには、以下のことを行います。

   a. `set` または `export` コマンドを入力して、シェル環境の現行の設定を表示します。
   b. 現行のシェルの設定を 68ページの表7 と比較します。この表には、コンパイルしようとするサーバー・アプリケーションのタイプに従って、make 処理の設定に必要であると考えられる環境変数を記載しています。使用すべき環境変数のいくつかは、インストールおよび個別の開発環境によって決まります。
   c. 特定のサーバー・アプリケーションごとに、適宜、$HOME/.profile ファイルを編集するか、シェル・スクリプトを作成して、環境変数の設定を追加またはオーバーライドします。

      推奨:

      - 51ページの表4 にある環境変数が /etc/profile に設定されていない場合は、$HOME/.profile ファイルに設定してください。これらの環境変数は、WebSphere for z/OS 用に開発したすべてのサーバー・アプリケーションでも同じ設定でなければなりません。シェル・セッションで実行されるシェル・コマンドおよびスクリプトに受け渡すために、必ず、これらの変数をエクスポートしてください。

- シェル・スクリプトの make 処理に対する環境変数を設定します。アセンブルしているサーバー・アプリケーションごとに 1 つのシェル・スクリプトを使用することを考慮し、スクリプトごとに、以下の命名規則を使用します。

  *serverappname*_make_setup.sh

  アプリケーションごとに 1 つのスクリプトを使用して、個別アプリケーションごとに適宜、シェル設定およびコンパイラー・オプションを変更することができます。

  シェル・スクリプトを使用して、サーバー・アプリケーション用のオブジェクト・ビルダー生成 prjdefs.mk ファイルにも現れるいくつかのオプションを指定する場合、必ず prjdefs.mk ファイルを編集して、重複オプションをコメント化してください。

d. (オプション) WebSphere for z/OS と共に出荷された CB390make.env ファイルの任意のデフォルト設定を変更したい場合は、以下のようにします。

   1) CB390make.env ファイルの独自のコピーを作成します。CB390make.env は IVB_DRIVER_PATH/samples にあります。
   2) CB390make.env のコピーを編集して、変更内容を保管します。
   3) CB390make.env のコピーを指し示すように、CB390_ENVFILE 環境変数の設定を変更します。

---

2. シェル・スクリプトを使用して、変数を設定する場合、make コマンドを使用して make 処理を始動する前に、このスクリプトを実行する必要があります。シェル・スクリプトを実行するには、以下のコマンドを入力します。

   *serverappname*_make_setup.sh

---

3. z/OS または OS/390 上の作業ディレクトリーから、make -f all.mak と入力します。

   ヒント:

   - バックグラウンドで作成プロセスを実行し、必要に応じて、後で見直しするためにすべてのメッセージをファイルに書き込むには、以下の make コマンドを使用します。たとえば、./build_results.txt ファイルには、make プロセスで生成されたすべてのメッセージが含まれています。

     ```
     make -f all.mak 1>build_results.txt 2>&1 &
     ```

- ファイルを使用して、作成プロセスで生成されたメッセージを収集する場合、*z/OS UNIX システム・サービス コマンド解説書*, SA88-8641 で説明している、`head` および `tail` UNIX コマンドを使用して、make 処理が完了する前にメッセージを見直すことができます。
- ロード・ライブラリーに実行可能コードを置いている場合、make プロセスは、システム完了 (ABEND) コード x37 で失敗する可能性があります。これらの ABEND コードの意味は、データ・セットが小さ過ぎるなどのエラーです。問題点を判別して、それを修正するには、*z/OS MVS システム・コード*, SA88-8592 で特定の ABEND コードを検索してください。次に作成プロセスを再始動します。
- make プロセスが失敗した場合は、エラーを修正して、作成プロセスを再始動します。もう一度、最初から始動する必要がある場合は、`make -f all.mak clean` というコマンドを発行して、最初の作成試行時に HFS で生成されたすべてのファイルを消去します。

サーバー・アプリケーションの実行可能ファイルが、指定したデータ・セットまたは HFS 上の作業ディレクトリーにあることが確認できます。

CORBA アプリケーションをシステム検索パスに追加するためのステップ

サーバーにアプリケーションを配置する前に、実行可能コードを以下のいずれかの位置に移動することができます。

- HFS におけるコードの場合、リンク・パック域 (LPA) に HFS ファイルをロードすることができます。
- PDS または PDSE におけるコードの場合、LPA にデータ・セットをロードするか、リンク・リストにデータ・セットを追加することができます。

実行時用にコードを配置する場所は、システムおよびアプリケーションのパフォーマンス、および仮想記憶域の使用に影響を与えます。アプリケーション・コードに最適な配置を決定するには、59ページの『サーバー・アプリケーションに対して実行可能コードを配置する場所の決定に関するバックグラウンド』の推奨と情報を参照してください。

サーバー・アプリケーション用の実行可能コードをリンク・パック域 (LPA) またはリンク・リストにロードするには、以下の手順のいずれかを完了してください。

- HFS の DLL を LPA に移動するには、*z/OS UNIX システム・サービス 計画*, GA88-8639 の手順を参照してください。

- PDS または PDSE の DLL を LPA に移動するには、以下のことを行ってください。
  1. LPA ステートメントを使用する PROGxx parmlib メンバーを作成して、クライアントおよびサーバーの DLL を LPA に追加します。たとえば、WebSphere for z/OS インストール検査プログラムを LPA に追加するには、以下の LPA ステートメントが必要になります。

     ```
     LPA ADD MODNAME=(JPOLICYS) DSNAME(high-level_qualifier.SBBOLOAD)
     LPA ADD MODNAME=(JPOLSQMO) DSNAME(high-level_qualifier.SBBOLOAD)
     LPA ADD MODNAME=(JPOLTRIR) DSNAME(high-level_qualifier.SBBOLOAD)
     LPA ADD MODNAME=(POLICYC) DSNAME(high-level_qualifier.SBBOLOAD)
     LPA ADD MODNAME=(POLICYS) DSNAME(high-level_qualifier.SBBOLOAD)
     LPA ADD MODNAME=(POLSQIR) DSNAME(high-level_qualifier.SBBOLOAD)
     LPA ADD MODNAME=(POLTRIR) DSNAME(high-level_qualifier.SBBOLOAD)
     LPA ADD MODNAME=(POLHMIR) DSNAME(high-level_qualifier.SBBOLOAD)
     ```

     PROGxx parmlib メンバーに関する追加情報が必要な場合は、*z/OS MVS 初期設定およびチューニング 解説書*, SA88-8564 を参照してください。
  2. MVS コンソールまたは TSO/E セッションから、SET PROG=xx コマンドを入力して、アプリケーションを LPA にロードします。SET PROG=xx コマンドに関する追加情報が必要な場合は、*z/OS MVS システム・コマンド*, SA88-8593 を参照してください。
- PDS または PDSE の DLL をリンク・リストに移動するには、以下のことを行ってください。
  1. サーバー・アプリケーションの実行可能コードを含む PDS または PDSE を指定するために、LNKLST ADD ステートメントを使用する PROGxx parmlib メンバーを作成します。LNKLST ADD ステートメントまたは PROGxx parmlib メンバーに関する追加情報が必要な場合は、*z/OS MVS 初期設定およびチューニング 解説書*, SA88-8564 を参照してください。
  2. MVS コンソールまたは TSO/E セッションから、SET PROG=xx コマンドを入力して、アプリケーションをリンク・リストに追加します。SET PROG=xx コマンドに関する追加情報が必要な場合は、*z/OS MVS システム・コマンド*, SA88-8593 を参照してください。

CORBA アプリケーションにデータ・オブジェクトをバインドするためのステップ

サーバー・アプリケーションに、DB2 を使用してパーシスタント・データを保管するビジネス・オブジェクトおよびデータ・オブジェクトが含まれている場合、これらのオブジェクトが使用する DB2 表をセットアップし、各データ・オブジェクトを DB2 for OS/390 計画の独自のパッケージにバインドすることができます。DB2 表のセットアップについて、詳しくは、97ページの『DB2 を準備するためのステップ』を参照してください。

データ・オブジェクトをパッケージにバインドするには、サーバー・アプリケーションのコンパイル時に、make プロセスにより生成されたデータベース要求モジュール (DBRM) を使用する必要があります。make プロセスは、データ・オブジェクトごとに 1 つの DBRM ライブラリーを生成します。

始める前に: 必要に応じて以下の解説書を検討したり、入手したりしてください。

- DB2 パッケージのバックグラウンド情報については、*IBM DB2 適用業務プログラミングおよび SQL の手引き*, SC88-7377。
- BIND PACKAGE サブコマンド構文およびオプションについての詳細は、*IBM DB2 コマンド解説書*, SC88-7378。
- DB2 for OS/390 SYSADM 権限のある ID のセットアップについては、*IBM DB2 管理の手引き*, SC88-7376。

各データ・オブジェクトを独自のパッケージにバインドするには、以下のステップを実行します。

1. 作業 JCL データ・セットのメンバーを作成し、そこに以下の JCL サンプルをコピーします。

```
//jobname JOB
//*
//BIND1    EXEC  PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DBRMLIB  DD DISP=SHR,DSN=project_dbrmlib
//SYSTSIN  DD *
 DSN SYSTEM(DB2_subsystem_name)
 BIND PACKAGE(package_name) MEMBER(dbrm_name) ACTION(REPLACE) +
   bind_options
END
/*
//BIND2    EXEC  PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DBRMLIB  DD DISP=SHR,DSN=CB390_dbrmlib
//SYSTSIN  DD *
 DSN SYSTEM(DB2_subsystem_name)
   BIND PACKAGE(package_name) MEMBER(BBOSQDB ) ACTION(REPLACE) +
     ISOLATION(CS) CURRENTDATA(NO) RELEASE(COMMIT)             )
END
/*
```

---

2. 以下のように、メンバーを編集して、インストール用の JCL をカスタマイズします。

- インストール特定パラメーターを指定して JOB カードを更新します。
- アプリケーションおよびそのクライアントがクエリー・サービスを使用しない場合は、BIND2 ステップを削除またはコメント化します。DB2 を使用するすべてのサーバー・アプリケーションで BIND1 ステップを使用しなければなりませんが、BIND2 ステップが必要なのは、アプリケーションでクエリー・サービスを使用している場合のみです。
- 以下のように、変数を適切な値に置き換えてください。

表 8. DB2 パッケージをバインドするために JCL で置き換える変数

置き換えられる変数	置き換える値
project_dbrmlib	make 生成 DBRM ライブラリーを指定します。 アプリケーションには DB2 を使用する複数のデータ・オブジェクトがあるので、make プロセスで複数の DBRM を生成した場合は、必要に応じて、make 生成 DBRMLIB データ・セットをある区分データ・セットにコピーするか、別の DBRMLIB DD ステートメントを BIND1 ステップに追加してください。
DB2_subsystem_name	データ・オブジェクトが使用する表を管理する DB2 サブシステムを指定します。
package_name	この DB2 パッケージに使用したい名前を指定します。
dbrm_name	データベース要求を行うロード・モジュール名を指定します。つまり、*project_dbrmlib* に指定したライブラリーのメンバー名を使用します。各メンバーは、サーバー・アプリケーションのデータ・オブジェクトに対応しています。 複数の *project_dbrmlib* を指定するか、*project_dbrmlib* に複数のメンバーがある場合は、必ず、メンバーごとに 1 つの BIND PACKAGE サブコマンドを指定してください。
bind_options	パッケージをバインドするのに使用したいオプションを指定します。

- BIND2 ステップを使用している場合、DBRMLIB DD ステートメントの *CB390_dbrmlib* を置き換えます。表8 に示しているように、他の変数を置き換えます。

---

3. DB2 for OS/390 SYSADM 権限のあるユーザー ID から、ジョブを実行依頼します。

第4章 WebSphere for z/OS MOFW における CORBA アプリケーションの配置

実行可能 CORBA サーバー・アプリケーションが完成すると、そのランタイム環境 (WebSphere for z/OS アプリケーション・サーバー) をセットアップする必要が生じます。WebSphere for z/OS には、2 種類のアプリケーション・サーバーがあります。1 つは Java 2 エンタープライズ版 (J2EE) アプリケーション、もう 1 つは CORBA アプリケーション用のアプリケーション・サーバーです。CORBA アプリケーション用のサーバーは、マネージド・オブジェクト・フレームワーク (MOFW) サーバーと呼ばれています。WebSphere for z/OS 管理アプリケーションを使用して CORBA アプリケーションにサーバーを定義する際、管理アプリケーションではサーバーに対して `J2EE サーバー`および `Server` (MOFW サーバー・タイプの場合) の 2 つのラベルが使用されます。CORBA アプリケーションの場合は、`Server` を使用するようにしてください。

規則:

- CORBA アプリケーションを J2EE サーバー上に配置することはできません。
- J2EE アプリケーションを MOFW サーバー上に配置することはできません。

z/OS または OS/390 上にサーバー・アプリケーションを**配置する**には、アプリケーションを配置するための知識と、CORBA アプリケーションに必要な z/OS または OS/390 およびサブシステムの知識が必要です。z/OS または OS/390 システム・プログラマーとデータベース管理者は、アプリケーションの配置に必要な技術を有する可能性の最も高い人たちです。本書では、サーバー・アプリケーションのアセンブルおよび配置担当者を**アプリケーション・アセンブラー**と呼びます。これらの人たちは、配置するサーバー・アプリケーションについての詳細の他に、以下についての知識が必要です。

- USS 環境、および z/OS または OS/390 コンポーネントおよびデータ・セットにアクセスするための TSO/E での作業方法。
- アプリケーションに必要な z/OS または OS/390 サブシステム・リソースの設定方法。たとえば、これらのリソースにはデータベース、トランザクション・マネージャー、およびセキュリティー・プロダクトが含まれていることがあります。

各分野における精通の度合いは、配置するサーバー・アプリケーションのタイプによって異なります。たとえば、CORBA アプリケーションが IMS トランザクションを介してアクセスしないで直接 DB2 データにアクセスする場合には、IMS についての知識は必要ありません。

- WebSphere for z/OS MOFW アプリケーション・サーバーの定義と活動化の方法 (つまり、配置するアプリケーションのためのランタイム環境)。

ランタイム環境のエレメントに対する命名規則に関するバックグラウンド

WebSphere for z/OS 管理アプリケーションにより、サーバー構成を定義するときに、特定のタイプのアプリケーションについてランタイム環境のモデルを作成します。モデルを作成するときは、以下のものについて名前を指定しなければなりません。

- 汎用アプリケーション環境。これは、**サーバー**と呼ばれます。
- ある種の作業の役割を担うエンティティー。このエンティティーは、**サーバー・インスタンス**と呼ばれます。これは、1 つの制御領域と複数のサーバー領域から成ります。
- 論理リソース・マッピング (LRM)。これは、アプリケーションで使用する、特定の**タイプ**のリソース・マネージャー (DB2、IMS、または CICS など) です。
- LRM インスタンス。これは、特定のリソース・マネージャー・サブシステム (DB2SYSA または IMS2 など) を識別します。

サーバーおよび LRM は、いわば、一般的な定義です。それに対し、サーバー・インスタンスおよび LRM インスタンスは、z/OS または OS/390 に存在する特定のエンティティーに名前を与えたものです。

規則: ユーザーのモデルが z/OS または OS/390 でアクティブなランタイム環境となるには、以下のように、管理アプリケーションで使用される名前が、z/OS または OS/390 の関連定義で使用する名前と同じであることを確認しなければなりません。

- 管理アプリケーションで定義する**サーバー**、およびワークロード管理 IWMARIN0 ダイアログで定義するアプリケーション環境には、同じ総称名 (たとえば、`PAAIMSV`) を使用します。
- 管理アプリケーションだけでなく、以下の場所でも、**サーバー・インスタンス**には、同じ名前 (たとえば、`PAAIMSV1`) を使用しなければなりません。
  1. 制御領域を始動するために作成する JCL プロシージャー。この JCL proc は、通常、BBO.SBBOJCL の BBOASR1 メンバーのコピーです。独自使用のためにこのメンバーのコピーを作成するときに、PROC ステー

トメントを変更して、サーバー・インスタンスを識別しなければなりません。たとえば、以下のように指定します。

```
//label    PROC  SRVNAME='PAAIMSV1'
```

2. サーバー領域を始動するために作成する JCL プロシージャー。この JCL proc は、通常、BBO.SBBOJCL の BBOASR1S メンバーのコピーです。独自使用のためにこのメンバーのコピーを作成するときに、IWMSSNM パラメーターを変更して、サーバー・インスタンスを識別しなければなりません。たとえば、以下のように指定します。

```
// label    PROC  IWMSSNM='PAAIMSV1', PARMS='-ORBsrvname ',
//     CBCONFIG='/u/cb390'
```

MOFW サーバーのエレメントの命名規則についての推奨内容は、*WebSphere Application Server V4.0 for z/OS and OS/390: 操作および管理*, SA88-8653 を参照してください。

WebSphere for z/OS MOFW サーバーの環境変数の設定に関するバックグラウンド

管理アプリケーションを使用して新規の WebSphere for z/OS MOFW サーバーを作成したり、既存の WebSphere for z/OS MOFW サーバーを変更するときには、環境変数の設定値を変更することもできます。

149ページの『付録A. 環境ファイル』は、サーバー・アプリケーションが稼動する WebSphere for z/OS MOFW サーバーに使用する必要がある環境変数をリストしています。このリストを使用して、設定する必要がある変数を決定します。特に重要な変数の 1 つに、CLASSPATH 変数があります。この変数には、サーバー・アプリケーション用の JAR ファイルの完全修飾名 (ある場合) を手動で追加する必要があります。

WebSphere for z/OS MOFW サーバーを始動する JCL プロシージャーのコーディング

アプリケーション制御領域およびサーバー領域用の JCL プロシージャーを作成するには、TSO で以下のステップを実行します。

1. 作業用 PROCLIB データ・セットに、汎用サーバー名と一致する名前の新規メンバーを作成します。BBOASR1 サンプル・メンバーを BBO.SBBOJCL からこの新規メンバーにコピーして、ファイル内のコメントに従って、適切な更新を行います。WebSphere for z/OS 管理アプリケーションで指定するサーバー・インスタンス名を識別するように PROC ステートメントを変更します。この新規メンバーは、これで、アプリケーション制御領域を始動するのに使用できる JCL プロシージャーです。

2. PROCLIB には、アプリケーション・サーバー領域を始動する JCL プロシージャーを含む新規メンバーも作成します。BBOASR1S サンプル・メンバーを BBO.SBBOJCL からこの新規メンバーにコピーして、ファイル内のコメントに従って、以下のように、適切な更新を行います。
   - WebSphere for z/OS 管理アプリケーションで指定するサーバー・インスタンス名を使用するように IWMSSNM パラメーターを編集します。
   - サーバー・アプリケーションの実行可能コードの位置 (PDS、PDSE、または HFS) を識別できるようにメンバーを編集します。

WebSphere for z/OS MOFW サーバーの定義

9ページの『配置するサーバー・アプリケーションのバックグラウンド』は、サーバー・アプリケーション用にアプリケーション・サーバーまたはランタイム環境の定義を完了するステップを示しています。このプロセスのステップは、アプリケーション・サーバーをテスト用に定義しているか、または実動可能なアプリケーション用に定義しているかどうかにかかわらず同じです。
WebSphere for z/OS 管理アプリケーションにより、妥当性検査、コミット、および活動化を行うモデル構成を定義します。その構成を活動化する場合、WebSphere for z/OS システム・マネージャーは z/OS または OS/390 にランタイム環境を作成します。12ページの図8 は、モデルを活動化した結果のサンプル・モデルとランタイム環境を示しています。

この構成は極めて単純です。1 つのサーバー・アプリケーションだけがコンテナーに対して定義されます。これは、1 つのリソース・マネージャー・サブシステムにのみ接続されます。さらに複雑な構成も可能です。たとえば、複数のマネージャーを特定のコンテナーに接続することもできます。これは、WebSphere for z/OS に固有の長所です。定義する構成は、サーバー・アプリケーションが必要とするサービスの質によって決まります。

サーバー構成を定義する場合に、ユーザーの指定するオプションまたは選択は、その環境で稼動するサーバー・アプリケーションの振る舞いに直接大きな影響を与えます。

注: WebSphere for z/OS には、2 種類のアプリケーション・サーバーがあります。1 つは Java 2 エンタープライズ版 (J2EE) アプリケーション、もう 1 つは CORBA アプリケーション用のアプリケーション・サーバーです。CORBA アプリケーション用のサーバーは、マネージド・オブジェクト・

フレームワーク (MOFW) サーバーと呼ばれています。WebSphere for z/OS 管理アプリケーションを使用して CORBA アプリケーションにサーバーを定義する際、管理アプリケーションではサーバーに対して J2EE サーバーおよび Server (MOFW サーバー・タイプの場合) の 2 つのラベルが使用されます。CORBA アプリケーションの場合は、Server を使用するようにしてください。

WebSphere for z/OS 管理アプリケーションの使用に関するバックグラウンド

WebSphere for z/OS 製品のインストール後に初めて WebSphere for z/OS 管理アプリケーションを使用するときは、CBADMIN アドミニストレーター・ユーザー ID を使用してログオンします。後で追加の管理者用ユーザー ID を定義して、いくつかのワークステーションまたはセッションからアドミニストレーション・アプリケーションにアクセスできるようにすることができます。ただし、追加のユーザー ID を使用するときは、以下の規則に留意してください。

- 同一の管理者 ID を使用して、単一のワークステーションまたは複数のワークステーションから、アプリケーションの複数並行セッションにログオンすることはできません。たとえば、ユーザー ID として CBADMIN を使用して、ワークステーションでアドミニストレーション・アプリケーションを始動する場合、独自のワークステーションまたは別のワークステーションから別のセッションを始動することはできません。ただし、別の管理者用ユーザー ID を使用すれば、どのワークステーションからでも別のセッションを始動することができます。
- いくつかの管理者用ユーザー ID を定義した場合、これらをすべて同時にログオンできますが、一度に会話の更新と活動化ができるのは 1 つだけです。ある管理者が会話を活動化すると、その他の管理者は、読み取り専用または表示機能のためにアドミニストレーション・アプリケーションを使用する必要があります。

テスト・システムのサーバー・プロパティーの選択

MOFW サーバー・プロパティーの完全なリストおよび説明については、管理アプリケーションにより使用可能なヘルプを使用するか、*WebSphere Application Server V4.0 for z/OS and OS/390: システム管理ユーザー・インターフェース*, SA88-8656 を参照してください。

MOFW サーバーに対するコンテナーの定義

管理アプリケーションを使用して CORBA アプリケーションにコンテナーを追加するときは、以下に関する情報が必要です。

- アプリケーション開発におけるパッケージ化または構成段階でオブジェクト・ビルダーによって指定されたコンテナー名。

  規則: 所定のサーバー・アプリケーションについては、管理アプリケーションに指定されたコンテナー名とオブジェクト・ビルダーによって指定されたコンテナー名は一致する必要があります。
- サーバー・アプリケーションが必要とする quality of service (QoS)。これによって、コンテナー・プロパティーに指定する値が決まります。

注: オブジェクト・ビルダーを使用して z/OS または OS/390 プラットフォームに既存のアプリケーションを構成する場合、さらにそのアプリケーションがデフォルト・コンテナーに関連している場合は、WebSphere for z/OS によってそのアプリケーションを正常に配置することが許可されます。デフォルト・コンテナーはワークステーション・プラットフォーム上の Component Broker サーバーでの使用のみを目的として設計されているため、作成されたランタイム環境は、ワークステーション・プラットフォームのものとは異なる場合があります。WebSphere for z/OS MOFW サーバーでは、これらのデフォルト・コンテナーには、一時オブジェクトを管理するコンテナーに対して OS/390 推奨特性と一致する事前定義特性があります。

コンテナーを定義するときに、コンテナーによるオブジェクトの管理方法、およびこれらのオブジェクトの振る舞いを決定する各種ポリシーを選択します。これらのポリシーの 1 つは、コンテナー・トランザクション・ポリシーで、コンテナーが管理するオブジェクトのトランザクションの有効範囲を定義します。ほとんどのサーバー・アプリケーション・オブジェクトの場合、TX 必須トランザクション・ポリシーを選択する必要があります。その他のどのトランザクション・ポリシーを使用する前にも、必ず、19ページの『WebSphere for z/OS トランザクション環境に関するバックグラウンド』を読んで理解してください。

制限: 照会可能なホームを含むコンテナーを定義するときは、コンテナーに対して複数の論理リソース・マッピング接続を定義することはできません。

WebSphere for z/OS MOFW サーバーをバックエンドのリソース・マネージャーに接続

IMS-OTMA LRM インスタンスの接続データを提供する場合のガイドライン

WebSphere for z/OS MOFW サーバーおよび IMS サブシステムが OTMA を介して通信するには、IMS および OTMA 構成に関連した接続データを提供する必要があります。

以下のリストにより、IMS-OTMA LRM インスタンスに提供する必要がある接続データ、およびそのデータについて適切な値を決定する方法が確認できます。

XCF グループ名
IMS 初期化で使用する DFSPBxxx proclib メンバーの GRNAME パラメーターに指定された名前を埋めます。

XCF パートナー名
IMS 初期化で使用する DFSPBxxx proclib メンバーの OTMANM パラメーターに指定された名前を埋めます。そうしない場合は、DFSPBxxx メンバーの APPLID1 パラメーターで指定された名前を使用します。これは、OTMANM パラメーターが定義されていない場合、デフォルトの XCF パートナー名になります。

セッション数
1 と指定します。

TPIPE プレフィックス
システムはこの LRM で必要なすべてのトランザクション・パイプで使用するために、プレフィックスを指定します。プレフィックスは、4 文字以下でなければなりません。この LRM のトランザクション・パイプを作成するときに、システムは、このプレフィックスを使用して、4 文字のセッション関連情報を付加して、固有トランザクション・パイプ名を生成します。

IMS サブシステムにアクセスする場合の OTMA の使用についての詳細は、1 冊または複数冊の以下の資料を参照してください。

- IMS-OTMA 手続き型アダプター・サポートを使用する WebSphere for z/OS MOFW サーバーのセットアップに関する手順については、*WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652。
- OTMA の使用可能な IMS については、*IMS/ESA Open Transaction Manager Access Guide*, SC26-8743。

IMS-APPC LRM インスタンスの接続データを提供する場合のガイドライン

WebSphere for z/OS MOFW サーバーおよび IMS サブシステムが APPC/MVS を介して通信するには、サーバーに関連した特定の APPC/MVS 論理装置 (LU)、および IMS に関連した LU に関する接続データを提供する必要があります。

以下のリストにより、IMS-APPC LRM インスタンスに提供する必要がある接続データ、およびそのデータについて適切な値を決定する方法が確認できます。

ローカル LU 名

WebSphere for z/OS に関連付けられた論理装置 (LU) 名を埋めます。このローカル LU 名は、WebSphere for z/OS が稼動するシステムのための APPCPMxx parmlib メンバーの LUADD ステートメントで定義されます。

WebSphere for z/OS に関連付けられた LU に対する LUADD ステートメントを探します。**ローカル** LU 名として ACBNAME パラメーターに指定された名前を使用します。

規則: ACBNAME パラメーターに指定された値のみ使用してください。これはネットワーク LU 名です。ローカル LU にネットワーク修飾 (または完全修飾) 名を指定すると、エラー・メッセージ BBOU0106E を受け取ります。これは、ローカル LU 名が無効であることを意味します。

パートナー LU 名

WebSphere for z/OS サーバーが APPC 会話を開始するのに使用する LU 名を埋めます。このパートナー LU 名は、IMS が稼動するシステムのための APPCPMxx parmlib メンバーの LUADD ステートメントで定義されます。WebSphere for z/OS サーバーが稼動するシステム以外のシステムに、IMS サブシステムが存在する可能性がありますが、存在しなければならない訳ではありません。

IMS に関連付けられる LU についての LUADD ステートメントを探します (IMS に関連付けられる LU には、LUADD ステートメントの SCHED パラメーターに対して指定された IMS サブシステム名が付いています)。**パートナー** LU 名として ACBNAME パラメーターに指定された値を使用します。

ヒント: パートナー LU 名を指定すると、以下のフォームのいずれかを使用することができます。

- ACBNAME パラメーターに指定された値のみ (つまり、ネットワーク LU 名)。
- ネットワーク修飾名 (*networkID.networkLUname* フォームによる)。

  networkID は、VTAM 始動オプション NETID に指定された値です。*networkLUname* は、ACBNAME パラメーターに指定された値です。
- 汎用リソースを使用するようにインストール内容が構成されている場合は、VTAM 汎用リソース名。

VTAM ログモード名

このローカル LU とそのパートナー LU の間で任意の APPC 会話と関連付けられるようにネットワーク特性を指定する VTAM ログモード名を埋めます。ログモード名は、VTAM ログオン・モード・テーブルに現れます。これは、インストールした VTAMLIB データ・セットにあります。

APPC 会話タイムアウト値

WebSphere for z/OS サーバーが、IMS との会話の間に発行する、Allocate 呼び出しおよび以降のあらゆる呼び出しへの応答を待つ時間の長さを分単位で指定します。有効なタイムアウト値の範囲は、0 〜 1440 までの 24 時間です。

OTS_DEFAULT_TIMEOUT 環境変数に設定されている値よりも小さい値を指定すると、APPC 会話タイムアウト値は効果がありません。アプリケーション・サーバーの制御領域およびサーバー領域で使用する、OTS_DEFAULT_TIMEOUT 環境変数の設定を検索します。

APPC 同期レベル

以下の表にリストされている値のいずれかを埋めます。この値は、WebSphere for z/OS サーバーが IMS と通信するのに使用する APPC/MVS 会話のタイプを制御します。このサーバー構成でコンテナーに対して選択するトランザクション・ポリシー、およびこのサーバーに配置するアプリケーションの特性に基づきます。

推奨: サーバーが現在処理している要求のトランザクションの内容に対応する同期レベルを使用してください。同期レベルと内容を一致させる最も簡単な方法は、**Autotran** を選択することです。これにより、システムは一致する同期レベルを判別します。

LRM の接続先	指定する同期レベル	注
すべてが、TX 必須トランザクション・ポリシーを使用する 1 つまたは複数のコンテナー。	**Syncpt** (場合によっては、**None** も受け入れ可能)	このトランザクション・ポリシーは、グローバル・トランザクションの使用を強制するので、APPC 同期レベルの最も論理的な値は **Syncpt** です。**Syncpt** を指定すると、サーバーは、保護会話を割り振ります。これは、サーバーと IMS サブシステムの相互作用についてグローバル・トランザクションのコンテキストを保存し、会話のエラーまたは障害が発生した場合に、システムがすべてのリソースを回復できるようにします。 ただし、場合によっては、アプリケーションの処理がその処理のこの点でリソースを回復する能力に依存しない場合は、**None** を使用することも考えられます。**None** を指定すると、APPC/MVS、WebSphere for z/OS、および IMS は、分散アプリケーションによって実行された処理をまったく調整しないので、調整にかかるオーバーヘッドが発生しないことから、アプリケーションのパフォーマンスは向上します。 **推奨:** • サーバー・アプリケーションが、IMS サブシステムが稼動するのと同じ z/OS または OS/390 システムで常に稼動するということが保証できない場合は、**Syncpt** を使用してください。 • **None** は、慎重に使用してください。この場合、アプリケーションが使用するリソースは、会話エラーまたは障害が発生した場合、不整合な状態になる可能性があります。
TX 必須以外のトランザクション・ポリシーを使用する 1 つまたは複数のコンテナー。	**Autotran**	3 つのポリシーがある **Autotran** を使用します。それで、システムは、どちらの会話タイプ (Syncpt または None) が、現行スレッドの実行に関連付けられるトランザクションのコンテキストに適切であるか判断することができます。つまり、現行スレッドにローカル・トランザクションのコンテキストがある場合、サーバーは、同期レベル None を使用し、グローバル・トランザクションのコンテキストの場合、サーバーは Syncpt を使用します。

IMS サブシステムにアクセスする場合の APPC/MVS の使用についての詳細が必要な場合は、1 冊または複数冊の以下の資料を参照してください。

- ローカル LU およびパートナー LU として、それぞれ WebSphere for z/OS MOFW サーバーおよび IMS サブシステムを構成する手順については、*WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652。IMS-APPC 手続き型アプリケーション・アダプター・サポートの使用についてのトピックを参照してください。
- 通常の APPC/MVS 構成の詳細については、*z/OS MVS 計画: APPC/MVS 管理*, SA88-8571。
- VTAM 定義ステートメントについては、*OS/390 eNetwork Communications Server SNA リソース定義解説書*, SD88-7855。

CORBA アプリケーション DDL をインポートするためのガイドライン

サーバー・アプリケーションを正しく実行および管理するためには、WebSphere for z/OS サーバーは、CORBA サーバー・アプリケーションの内容を認識しておく必要があります。サーバー・アプリケーションをパッケージ化したときに、オブジェクト・ビルダーにより生成された DDL ファイルをインポートする場合に、サーバーはこの情報を取得します。

推奨: インポート・プロセスの場合の入力および出力ファイルについては、z/OS または OS/390 データ・セットではなく、HFS ファイルを使用することを考えてください。入力ファイルは、すでに HFS の作業ディレクトリーにあります。同じディレクトリーに出力ファイルを定義することができます。必ず、入力ファイルと異なる名前を使用してください。

始める前に:

- インポート・プロセスに z/OS または OS/390 データ・セットを使用したい場合は、DDL ファイルの内容をデータ・セットにコピーしなければなりません。HFS ファイルをデータ・セットにコピーする手順については、*z/OS UNIX システム・サービス ユーザーズ・ガイド*, SA88-8640 を参照してください。
- インポート DDL プロセスに対して、適切なセキュリティー定義をセットアップしていることを確認してください。このプロセスのための入力および出力ファイルは、BBOSMSS アドレス・スペースのユーザー ID に関連しているため、このユーザー ID は、以下のファイルにアクセスできなければなりません。

- インポートする DDL にデータ・セットを使用する場合、ユーザー ID は、入力データ・セットに対する読み取りアクセスが可能でなければならず、出力データ・セットに対する更新アクセスが可能でなければなりません。
- DDL に HFS ファイルを使用する場合、ユーザー ID には、入力ファイルを検索するためのディレクトリー検索機能、入力ファイルの読み取り機能、および出力ファイルへの書き込み機能が備わっていなければなりません。

- インポートする DDL にデータ・セットを使用する場合、これらのデータ・セットが別のプロセスで使用中でないことを確認してください。たとえば、インポートの開始と同時に、ISPF を使用して、データ・セットやデータ・セット・メンバーを編集したり、表示したりすることはできません。

注:

規則:

- 1 つの会話において、同じアプリケーション・ファミリーを削除して、再インポートすることはできません。アプリケーション・ファミリーを再インポートするには、以下のことを行う必要があります。
  1. 1 つの会話において、同じアプリケーション・ファミリーを削除します。
  2. その会話を活動化します。
  3. 別の会話を作成して、DDL を再インポートします。
- コンテナーを参照する任意の CORBA アプリケーション DDL をインポートできるようにするには、WebSphere for z/OS 管理アプリケーションでコンテナーを定義する必要があります。
- 特定の DDL をインポートする場合は、1 つの論理リソース・マッピングだけを DDL で参照されるコンテナーと関連付けることができます。
- アクティブな会話にすでに存在するオブジェクトを含む DDL をインポートすることはできません。

- DDL インポートに使用する入力および出力ファイルが階層ファイル・システム (HFS) にある場合、これらのファイルは、その上でシステム管理が稼動しているシスプレックスのすべてのシステムで (NFS マウントまたはレプリカ生成、または共用 HFS により) 使用可能でなければなりません。
- DDL インポートに使用される入力および出力ファイルが z/OS または OS/390 データ・セット (順次または区分) である場合、その特性は RECFM=VB、LRECL=1020、BLKSIZE=1024 となっていなければなりません。

- インポート・プロセス中に、インポートするデータ・セットを別のプロセスで使用することはできません。たとえば、インポートの開始と同時に、ISPFを使用して、データ・セットやデータ・セット・メンバーを編集したり、表示したりすることはできません。
- インポートする DDL を保管するために区分データ・セットを使用する場合は、インポート・プロセスで入力ファイル名を指定するときに、メンバー名を指定しなければなりません。
- インポート・プロセス用の出力ファイルとして、事前割り当てしたデータ・セットを指定する場合は、メンバー名を指定しなければなりません。
- 基本および特定の DDL を両方ともインポートする必要がある場合は、最初に、基本 DDL をインポートしてから、特定 DDL をインポートします。

アプリケーション処理のためのリソース・マネージャーの準備

インストール時に WebSphere for z/OS と共に使用するリソース・マネージャーが正しくセットアップされていることを確認するには、特定の要件について、*WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652 を参照してください。このセクションにおける手順は、インストール処理により、インストールが正しく実行され、アプリケーションで使用する必要があるすべてのリソース・マネージャーが構成されていることを前提としています。

DB2 を準備するためのステップ

サーバー・アプリケーションがデータの保管に DB2 を使用する場合、アプリケーション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。

- 各サーバー・アプリケーション・データ・オブジェクトを DB2 for OS/390 計画の独自のパッケージにバインドします。手順については、81ページの『CORBA アプリケーションにデータ・オブジェクトをバインドするためのステップ』を参照してください。
- サーバー・アプリケーションで使用する新規 DB2 表を作成するか、既存の DB2 表を検査します。

  推奨:

  – オブジェクト・ビルダーは、データベース・スキーマ・ファイルを生成できますが、ファイル入力 (SPUFI) 機能を使用して、SQL プロセッサーにより新規データベース・テーブルを作成してください。データベース・テーブルの作成について、詳しくは、*IBM DB2 適用業務プログラミングおよび SQL の手引き*, SC88-7377 を参照してください。

– ページ固定ではなく、行固定を使用してください。行固定の利点については、*IBM DB2 管理の手引き*, SC88-7376 を参照してください。

IMS を準備するためのステップ

z/OS または OS/390 で CORBA アプリケーションを駆動する任意のクライアント・アプリケーションを実行する前に、以下のステップを完了して、IMS サブシステムを準備します。

1. IMS 並列スケジューリング限度を 0 (ゼロ) に設定します。これは、任意のメンバーのトランザクションがスケジュールできるということです。
   - 現行設定を決定するには、`/DISPLAY TRANSACTION` コマンドを発行して、表示結果で `PARLM` というヘッダーの下の値を探すことができます。
   - 並列スケジューリング限度に新規の値を設定するには、IMS `/ASSIGN PARLIM` コマンドを発行することができます。

   IMS コマンドの使用について、詳しくは、*IMS/ESA Operations Guide*, SC26-8741 または *IMS/ESA オペレーター用解説書*, SD88-7127 を参照してください。

   ---

2. 十分な IMS メッセージ処理領域 (MPR) を始動すれば、多数のクライアント・アプリケーション要求を処理することができます。必要な MPR の数は、アプリケーションの処理およびそれが稼動する IMS 環境によって決まります。
   a. サーバー・アプリケーションおよびそのクライアントの情報と以下の図表のガイドラインを使用して、必要な MPR の数を決定します。

論理リソース・マッピング (LRM) のタイプ	MPR の数の決定のための参考情報
同期レベルが Syncpt の IMS-OTMA または IMS-APPC	この環境では、WebSphere for z/OS MOFW サーバー、IMS、およびその他のシステム・コンポーネントは、一緒に稼動して、単一のクライアント・トランザクション内でのすべての処理を調整するので、すべての更新は、正常に完了するか、ロールバックされるかのいずれかです。1 つの単一クライアント・トランザクションは、いくつかの IMS トランザクションを生成し、そのためにいくつかの MPR を使用するので、遅延が発生する可能性があります。これは、クライアント・トランザクションが完了した後に他の作業を処理できる状態になるまで、MPR が待機しなければならないためです。処理中のこのような遅延を避けるために、IMS には、少なくとも、クライアント・アプリケーションが生成する IMS トランザクションの数と同じ数の始動済み MPR が必要です。 たとえば、`begin` 命令と `commit/rollback` 命令を使用して、あるトランザクションの有効範囲を定義する CORBA アプリケーションについて考えます。その 1 つのトランザクション内に、クライアント・アプリケーションは、別に 3 つの IMS トランザクションを生成します。IMS トランザクションは、作業に対する要求の受け取り、作業するプログラムの呼び出し、および要求発行者への応答の送信に必要な処理で構成されます。この特定の場合には、単一クライアント・アプリケーションからの要求を処理するのに、少なくとも 3 つの MPR が必要です。
同期レベルが None の IMS-APPC	この環境では、WebSphere for z/OS MOFW サーバー、IMS、およびその他のシステム・コンポーネントは、単一クライアント・トランザクション内での処理をすべて調整するわけではありません。そのため、MPR は、単一 IMS トランザクションが完了するとすぐに、新しい作業で使用可能な状態になります。この場合、クライアント・アプリケーションが生成する IMS トランザクションの数よりも少ない数の MPR を使用して、受け入れ可能なアプリケーション・パフォーマンスを達成することができます。
同期レベルが Autotran の IMS-APPC	この環境で、システムは、IMS との通信を開始する必要がある、任意の APPC/MVS 会話で使用するために、同期レベルが Syncpt であるか None であるかを判断します。システムが同期レベル Syncpt を使用するときに発生する可能性がある遅延を避けるには、同期レベルが Syncpt の IMS-OTMA または IMS-APPC に対して、上記リストと同じガイドラインを使用してください。

b. 現在アクティブな MPR の数を判別します。判別するには、IMS /DISPLAY ACTIVE REGION コマンドを発行することができます。

c. 現在アクティブな領域数よりも多くの MPR が必要な場合は、IMSMSG という名前の IMS サンプル・ジョブを使用して、新規領域を始動します。このサンプル・ジョブは、IMS DFSMPR プロシージャーを起動して、新規領域を始動します。それぞれの MPR ごとに、固有の名前を指定するようにしてください。

   IMSMSG ジョブは、インストールの実施状況によって、IMS.JOBS または IMS.PROCLIB データ・セットのいずれかに入っています。このジョブについての詳細が必要であれば、*IMS/ESA 導入 第 2 巻*, GD88-7125 の『システム定義および調整』を参照してください。

WebSphere for z/OS インターフェース・リポジトリーへの CORBA アプリケーション・インターフェースの追加

クライアント・アプリケーションは、CORBA アプリケーションを使用するために、静的バインディングか動的インターフェース呼び出しを使用することができます。クライアントが動的インターフェース呼び出しを使用できるようにするには、z/OS または OS/390 プラットフォームに配置される CORBA アプリケーションごとに、WebSphere for z/OS インターフェース・リポジトリーを移植する必要があります。

オブジェクト・ビルダーは、サーバー・アプリケーションごとの作成プロセスの一部として、インターフェース・リポジトリー・ローダー・プログラム用のソース・ファイルを作成します。all.mak ファイルの実行は、サーバー・アプリケーション・コードだけでなくローダー・プログラムにも適合しています。結果的に作成された実行可能プログラム、*application-name*IR.exe は、サーバー・アプリケーション DLL と同じディレクトリーに配置されます。

ローダー・プログラムを実行して、サーバー・アプリケーション・インターフェースをインターフェース・リポジトリーに追加するには、以下のステップを実行します。

1. BBO.SBBOEXEC から作業 JCL データ・セットに BBOIRC3 メンバーをコピーします。

   ---

2. 以下のようにして、BBOIRC3 メンバーのコピーを編集します。
   - PGM 名を *application-name*IR に変更します。

- SET ARGV に以下のオプションの 1 つを単一引用符で囲んで指定します。

  IRdelete
  IR データベースから現行インターフェース情報のみを削除します。

  IRforce
  継承インターフェースおよび現行インターフェースの定義を強制して、それらをすべて再移植します。

  (デフォルト)
  現行インターフェース情報のみを強制削除して、このインターフェース情報を再移植します。

---

3. z/OS または OS/390 UNIX 環境または TSO/E からローダー・プログラムを実行します。
   - z/OS または OS/390 UNIX 環境から、実行可能プログラム名を入力します (デフォルトを使用したくない場合は引き数も入力します)。たとえば、以下のように入力します。

     ```
     executable-name,argv=IRforce
     ```

   - TSO/E から JCL プロシージャー名および実行可能プログラム名を指定します (デフォルトを使用したくない場合は引き数も入力します)。たとえば、以下のように入力します。

     ```
     s procname,exe=executable-name,argv=IRdelete
     ```

第5章 z/OS または OS/390 でのクライアント・アプリケーションの開発、アセンブル、および配置

z/OS または OS/390 クライアント・アプリケーションを開発する最も一般的な方法は、おそらく次のようなものになります。

1. Visual Age コンポーネント開発ツールを使ってワークステーションでクライアント・アプリケーションと CORBA アプリケーションの両方をコーディングし、テストする。
2. WebSphere for z/OS (MOFW) サーバーでサーバー・アプリケーションをアセンブル、配置、およびテストする。
3. 次のようにして、クライアント・アプリケーションを z/OS または OS/390 に移植する。
   a. ワークステーションまたは z/OS または OS/390 のいずれかでソース・コードを適宜編集する
   b. z/OS または OS/390 でコードをコンパイルしリンク・エディットする

ただし、最初にワークステーション上で稼動するクライアント・アプリケーションのバージョンを作成しないで、z/OS または OS/390 上で直接クライアント・アプリケーションをコード化し、テストすることができます。

しかし、方法に関係なく、アプリケーション・プログラマーには、以下のような、z/OS または OS/390 上で稼動するクライアント・アプリケーションを開発するためのスキルが必要です。

- C++ または Java プログラム言語
- Component Broker 共通クライアント・プログラミング・モデル。*WebSphere Application Server エンタープライズ版 Component Broker: プログラミングの手引き* に詳細な説明があります。
- 17ページの『第2章 WebSphere for z/OS における CORBA アプリケーションの開発方法の学習』におけるクライアント・アプリケーションとサーバー・アプリケーションのための設計上の考慮事項、および z/OS または OS/390 固有のガイドライン
- z/OS または OS/390 UNIX システム・サービス。この中には、シェルでの作業、環境変数の設定、`make` コマンドを使ったコードのコンパイル、およびその他のプログラミング作業が含まれます。
- インストールのセキュリティー・プロダクトを介して使用可能なセキュリティー・サービス

次の表は、z/OS または OS/390 でのクライアント・アプリケーションの準備と実行のためのサブタスク、および関連情報を示しています。

サブタスク	関連情報 (参照項目)
クライアント・アプリケーションに対して WebSphere for z/OS がサポートする環境についての学習	105ページの『サポートされるクライアントのランタイム環境に関するバックグラウンド』
z/OS または OS/390 でのアプリケーション開発環境についての学習	108ページの『z/OS または OS/390 におけるアプリケーション開発環境のセットアップに関するバックグラウンド』
クライアント・アプリケーションの開発	• 107ページの『サーバー・アプリケーションのためのクライアントの設計およびコーディングに関するバックグラウンド』 • 143ページの『Component Broker for Windows NT からのアプリケーションのマイグレーションに関するバックグラウンド』 • 19ページの『WebSphere for z/OS トランザクション環境に関するバックグラウンド』 • 30ページの『CORBA アプリケーションおよびクライアント・アプリケーションの制約事項』
z/OS または OS/390 でのクライアント・アプリケーションのアセンブル	• 108ページの『クライアント・アプリケーションに対して実行可能コードを配置する場所の決定に関するバックグラウンド』 • 109ページの『クライアント・アプリケーションの実行可能コードに対するデータ・セットの割り振りに関するバックグラウンド』 • 109ページの『make 処理のための環境変数の設定に関するステップ』 • 109ページの『z/OS または OS/390 でクライアント・アプリケーションをコンパイルするためのステップ』

サブタスク	関連情報 (参照項目)
z/OS または OS/390 でのクライアント・アプリケーションの開発	• 112ページの『サーバーおよび z/OS または OS/390 クライアントのセキュリティーのセットアップに関するバックグラウンド』 • 112ページの『z/OS または OS/390 でクライアント・アプリケーションを実行するためのステップ』

サポートされるクライアントのランタイム環境に関するバックグラウンド

MOFW サーバーで実行される CORBA アプリケーションの場合、WebSphere for z/OS は、106ページの図16 で示すように、z/OS または OS/390 およびその他のプラットフォームで実行されるクライアント・アプリケーションをサポートします。z/OS または OS/390 で実行されるクライアント・アプリケーションの場合、ランタイム環境には、以下のように、z/OS または OS/390 言語環境プログラムおよび UNIX システム・サービス (USS) が含まれます。

- z/OS または OS/390 言語環境プログラムは、C++、Java、およびその他の単一ランタイム環境で、メッセージ処理や記憶域管理などの共通サービスと言語固有のルーチンを提供します。

  z/OS または OS/390 の各種エレメントで言語環境プログラムが必要になるため、ユーザーのサイトのシステム・プログラマーは、z/OS または OS/390 のインストールの一環として、このランタイム環境をすでにセットアップしています。サポート言語用の共通ランタイム・ライブラリー (複数の場合もある) は、リンク・リスト、STEPLIB、またはランタイム・ライブラリー・サービス (RTLS) を介して、クライアント・アプリケーションですでに使用可能です。

- UNIX システム・サービスは、シェル、ユーザーが z/OS または OS/390 と対話できるようにする標準 UNIX コマンド行インターフェース、UNIX ユーザーがその作業環境で期待しているユーティリティーを提供します。UNIX システム・サービスを介して、ユーザーは、アプリケーション・プログラムの開発および移植、ファイルの管理、プロセス制御などを行うことができます。

  言語環境プログラムの場合と同じく、ユーザーのサイトのシステム・プログラマーは、すべてのユーザーのための USS 環境をすでにセットアップしています。z/OS または OS/390 における WebSphere for z/OS クライアント・アプリケーションの作成および実行の一環として、この章で説明しているように、環境を変更したり調整したりしなければなりません。

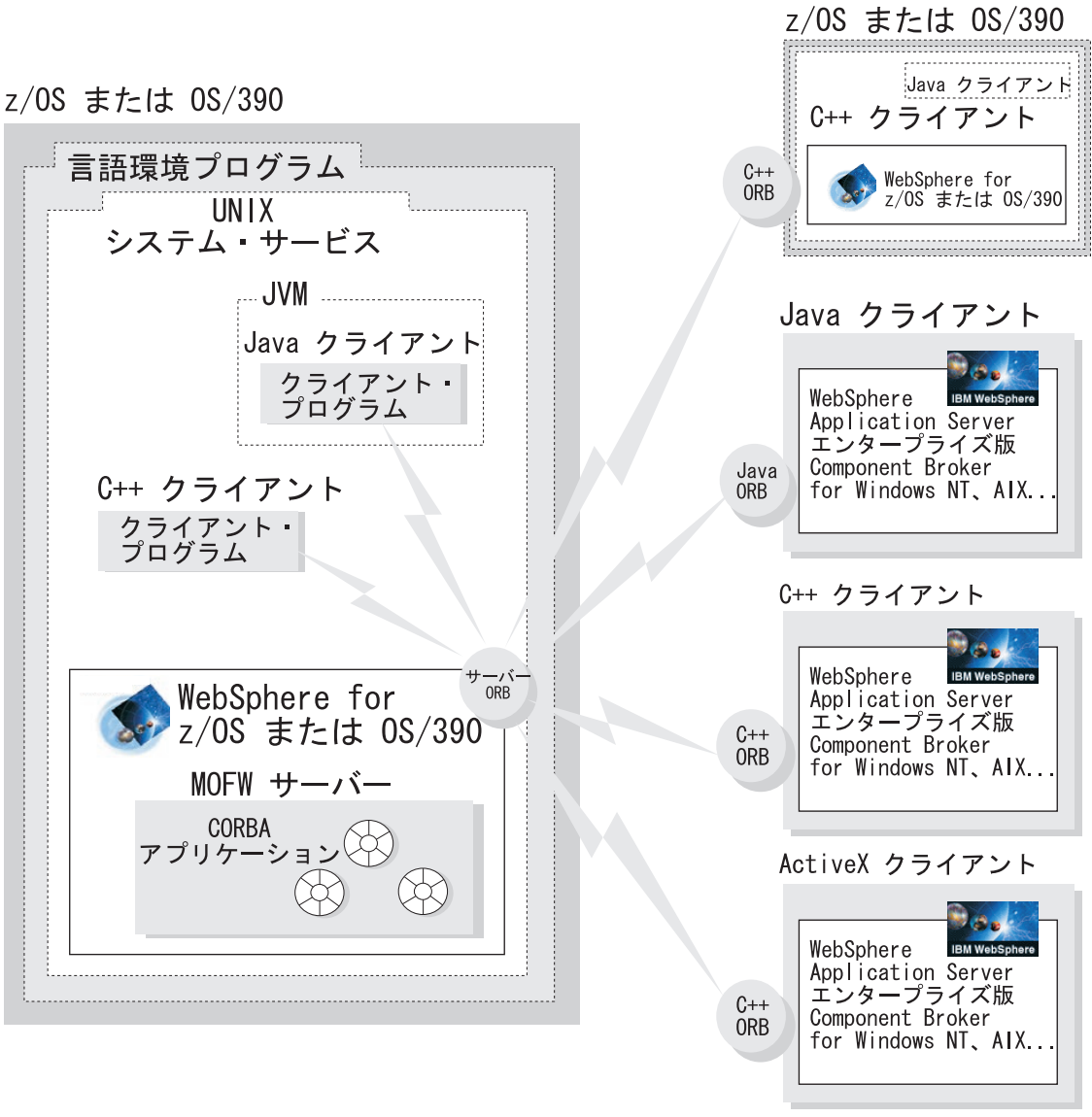

図 16. MOFW サーバーの CORBA アプリケーションついて WebSphere for z/OS がサポートするクライアント

この章で説明するトピックは、z/OS または OS/390 で稼動するクライアントにのみ適用されます。これらのクライアントは、以下のいずれかで稼動することができます。

1. クライアントが使用するオブジェクトを管理する WebSphere for z/OS MOFW サーバーと同じシステム。

2. WebSphere for z/OS が稼動している別のシステム。

1 番目の場合、106ページの図16 の左側の図にあるように、クライアント・アプリケーションは WebSphere for z/OS MOFW サーバーに対してローカルです。2 番目の場合、106ページの図16 の右側の図にあるように、クライアント・アプリケーションは WebSphere for z/OS MOFW サーバーに対してリモートです。

必要であれば、詳細について、以下のソースを参照してください。

トピック	参照する資料
z/OS または OS/390 以外のプラットフォームでのクライアント・アプリケーションの構成および実行	• *WebSphere Application Server エンタープライズ版 Component Broker プログラミングの手引き* • *WebSphere Application Server エンタープライズ版 Component Broker システム管理の手引き*
WebSphere for z/OS クライアントとしての IMS の構成および実行	179ページの『付録B. WebSphere for z/OS クライアントとしての IMS アプリケーション』
z/OS または OS/390 言語環境プログラム	• プログラム・モデル、呼び出し可能サービス、および用語集の説明など、言語環境プログラムの概念的な概要については、*z/OS 言語環境プログラム 概念*, SA88-8555 • 言語環境プログラムの下でアプリケーション・プログラムを作成および実行する手順については、*z/OS 言語環境プログラム プログラミング・ガイド*, SA88-8549
USS シェル環境	USS シェル環境の説明および各種システム・サービスを使用するための手順については、*z/OS UNIX システム・サービス ユーザーズ・ガイド*, SA88-8640

サーバー・アプリケーションのためのクライアントの設計およびコーディングに関するバックグラウンド

WebSphere Application Server Component Broker 製品は、共通プログラミング・モデルを定義しているので、クライアント・アプリケーションの設計およびコーディングに関する情報のほとんどが以下の資料に出てきます。

- *Component Broker プログラミングの手引き*
- *Component Broker 上級プログラミングの手引き*
- *Component Broker プログラミング解説書*
- *Component Broker アプリケーション開発ツールの手引き*

上記の資料では、Component Broker クライアント・アプリケーションを設計およびコード化するために理解しておく必要がある概念、コーディング演習、プログラミング・インターフェース、およびツールについて定義しています。上記の資料には、z/OS または OS/390 プラットフォームに適用されない概念またはインターフェースについても記載していますが、これらの制約事項が適用されるのは、WebSphere for z/OS MOFW サーバーで実行される CORBA アプリケーションに対してのみです。 Component Broker クライアント・プログラミング・モデルは、Component Broker 製品が実行されるすべてのプラットフォームで同一です。

z/OS または OS/390 におけるアプリケーション開発環境のセットアップに関するバックグラウンド

ユーザーのサイトのシステム・プログラマーが WebSphere for z/OS をインストールする場合、z/OS または OS/390 の UNIX アプリケーション開発環境を調整するオプションもあります。*WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652 で説明されている手順については、48ページの『アプリケーション開発環境をセットアップするためのステップ』にリストされています。通常、HFS スペースなどのアプリケーション・アセンブラーに対して行う調整は、z/OS または OS/390 でクライアント・アプリケーションを開発および実行するプログラマーが行ないます。

クライアント環境に対する特定のソフトウェア製品およびリリース・レベルについて、必要であれば、*WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652 を参照してください。

クライアント・アプリケーションに対して実行可能コードを配置する場所の決定に関するバックグラウンド

クライアント・アプリケーションの実行可能コードは、区分データ・セット (PDS)、拡張区分データ・セット (PDSE)、または階層ファイル・システム (HFS) に置くことができます。PDS は 16 MB 以下のロード・モジュールだけを含むことができるので、選択は、たいてい PDSE または HFS の 2 つのオプションの間で行われます。クライアント・アプリケーション・コードおよびサーバー・アプリケーション・コードに適用するオプションの推奨については、59ページの『サーバー・アプリケーションに対して実行可能コードを配置する場所の決定に関するバックグラウンド』を参照してください。

クライアント・アプリケーションの実行可能コードに対するデータ・セットの割り振りに関するバックグラウンド

クライアント・アプリケーションに対して実行可能コードをデータ・セットに配置することにした場合、または、クライアント・アプリケーションで DB2 リソースを直接使用する場合は、クライアント・アプリケーションをコンパイルする前の、複数の区分データ・セット (PDS) または PDSE の割り振りについて、63ページの『CORBA アプリケーションの実行可能コードに対するデータ・セットの割り振りに関するバックグラウンド』を参照してください。

make 処理のための環境変数の設定に関するステップ

クライアント・アプリケーションをコンパイルするには、make ユーティリティーを使用するのが簡単です。これは、UNIX システム・サービス (USS) シェルを介して使用可能です。『z/OS または OS/390 でクライアント・アプリケーションをコンパイルするためのステップ』には、使用可能なサンプル make ファイルが含まれています。このサンプル make ファイルは、WebSphere for z/OS MOFW サーバーに CORBA アプリケーションを配置するために、オブジェクト・ビルダーが生成する make ファイルを基にしています。ほとんどの場合、make 処理の要件、環境変数の設定、およびコードのコンパイル手順は、サーバー・アプリケーションでもそれらの z/OS または OS/390 クライアントでも同じです。make 処理および環境変数の設定については、65ページの『make 処理に関するバックグラウンド』で説明している内容を使用してください。

z/OS または OS/390 でクライアント・アプリケーションをコンパイルするためのステップ

クライアント・アプリケーションをコンパイルするには、make ユーティリティーを使用するのが簡単です。これは、UNIX システム・サービス (USS) シェルを介して使用可能です。以下の手順には、使用可能なサンプル make ファイルを示しています。このサンプル make ファイルは、WebSphere for z/OS MOFW サーバーに CORBA アプリケーションを配置するために、オブジェクト・ビルダーが生成する make ファイルを基にしています。ほとんどの場合、make 処理の要件、環境変数の設定、およびコードのコンパイル手順は、サーバー・アプリケーションでもそれらの z/OS または OS/390 クライアントでも同じです。

クライアント・アプリケーションをコンパイルするには、以下のステップを実行します。

1. コンパイル・コードに適切な環境変数が設定されていることを確認してください。設定する必要がある環境変数には、以下のことを行うものがあります。
   - WebSphere for z/OS 製品ファイルの位置の識別
   - サーバー・アプリケーションの実行可能コードを配置したい場所の識別

   make 処理の変数の設定に関する手順については、65ページの『make 処理に関するバックグラウンド』を参照してください。

---

2. クライアント・アプリケーションで必要なデータ・セットをすべて割り振っていることを確認してください。データ・セットの割り振りの手順については、63ページの『CORBA アプリケーションの実行可能コードに対するデータ・セットの割り振りに関するバックグラウンド』を参照してください。

---

3. ファイル命名規則 *filename*.mak を使用して、以下のサンプル make ファイルを作業 HFS ディレクトリーにコピーします。

```
all: pass3 pass7
OBJ_FILES=client.o
EXE_FILES=client.exe
.INCLUDE: obmdll30.mk
client.o.cppflags=NOEXPO
client.o : client.cpp
client.exe: client.o
```

---

4. z/OS または OS/390 の作業ディレクトリーから、make -f *filename*.mak と入力します。

   ヒント:
   - バックグラウンドで作成プロセスを実行し、必要に応じて、後で見直しするためにすべてのメッセージをファイルに書き込むには、以下の make コマンドを使用します。たとえば、./build_results.txt ファイルには、make プロセスで生成されたすべてのメッセージが含まれています。

     ```
     make -f filename.mak 1>build_results.txt 2>&1 &
     ```
   - ファイルを使用して、作成プロセスで生成されたメッセージを収集する場合、*z/OS UNIX システム・サービス コマンド解説書*, SA88-8641 で説明している、head および tail UNIX コマンドを使用して、make 処理が完了する前にメッセージを見直すことができます。
   - ロード・ライブラリーに実行可能コードを置いてあるときに、割り振ったデータ・セットが小さ過ぎる場合、make プロセスは、システム完了

(ABEND) コード x37 で失敗します。この問題を修正するには、さらに大きいデータ・セットを割り振り、作成プロセスを再始動します。

- make プロセスが失敗した場合は、エラーを修正して、作成プロセスを再始動します。もう一度、最初から始動する必要がある場合は、`make -f filename.mak clean` というコマンドを発行して、最初の作成試行時に HFS で生成されたすべてのファイルを消去します。

必要であれば、詳細について、以下のソースを参照してください。

トピック	参照する資料
USS シェル環境	*z/OS UNIX システム・サービス ユーザーズ・ガイド*, SA88-8640 は、USS シェル環境および環境変数を使用した作業について説明しています。
make ユーティリティー	• make の使用に関するチュートリアルについては、*z/OS UNIX システム・サービス プログラミング・ツール*, SA88-8643。 • `make` コマンド・フォーマットおよびオプション、make ファイルの使用上の注意、およびその他の参照情報については、*z/OS UNIX システム・サービス コマンド解説書*, SA88-8641。
IDL	• インターフェース定義言語およびその構文についての説明は、*Component Broker プログラミングの手引き*。 • `idlc` コマンド、および WebSphere for z/OS IDL コンパイラーのオプションおよび構文については、193ページの『付録C. インターフェース定義言語 (IDL) コンパイラー』。
DB2 プリコンパイラー	*IBM DB2 適用業務プログラミングおよび SQL の手引き*, SC88-7377 には、特定のサーバー・アプリケーションに対して変更が考えられるプリコンパイラー・オプションおよびデフォルト値がリストされています。
C++	• 特定のサーバー・アプリケーションに対して変更が考えられる (c89 コマンド記述の下の) コンパイラー・オプションについては、*z/OS UNIX システム・サービス コマンド解説書*, SA88-8641。 • z/OS または OS/390 C++ コンパイラーおよびそのオプションについての概要は、*z/OS C/C++ ユーザーズ・ガイド*, SC88-8850。
OS/390 における Java	http://www.s390.ibm.com/java/

サーバーおよび z/OS または OS/390 クライアントのセキュリティーのセットアップに関するバックグラウンド

ユーザー識別および認証を含むいくつかのセキュリティー課題が、z/OS または OS/390 で稼動するクライアント・プログラムに適用されます。クライアント・アプリケーションに対するセキュリティーのセットアップ方法は、主に、クライアント・アプリケーションで使用する WebSphere for z/OS MOFW サーバーの種類、クライアントがこれらのサーバーに対してローカルかリモートか、これらの通信で必要なセキュリティーの程度によって決まります。

クライアント・ランタイム環境のセキュリティーの設定について、詳しくは、*WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652 を参照してください。

z/OS または OS/390 でクライアント・アプリケーションを実行するためのステップ

クライアント・アプリケーションの実行可能コードがあれば、クライアント・アプリケーションを実行して、WebSphere for z/OS MOFW アプリケーション・サーバーに配置されている CORBA アプリケーション (つまり、オブジェクト) を使用することができます。

クライアント・アプリケーションを実行するには、以下のステップを実行します。

1. クライアントおよびそれが使用する WebSphere for z/OS MOFW サーバーに適切なセキュリティー機構をセットアップします。クライアント・ランタイム環境のセキュリティーの設定について、詳しくは、*WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652 を参照してください。
2. クライアント・ランタイム環境に適切な環境変数を設定します。コンパイル時変数の設定に使用したものと同じ機構および推奨を使用することができます。必要であれば、77ページの『z/OS または OS/390 で CORBA アプリケーションのソース・ファイルをコンパイルするためのステップ』を参照してください。ただし、設定する実際の変数値は、149ページの『付録A. 環境ファイル』にある表および定義を使用します。

---

3. シェル・スクリプトを使用して、変数を設定する場合、クライアント・アプリケーションを実行する前に、このスクリプトを実行する必要があります。シェル・スクリプトを実行するには、以下のコマンドを入力します。

   ```
   . clientappname_setup.sh
   ```

4. クライアント・アプリケーションを始動します。z/OS または OS/390 UNIX 環境により使用可能なメソッドをどれでも使用することができます。プログラムの実行についての詳細が必要であれば、*z/OS UNIX システム・サービス ユーザーズ・ガイド*, SA88-8640 または *z/OS 言語環境プログラム プログラミング・ガイド*, SA88-8549 を参照してください。

第6章 実動システムにおける CORBA アプリケーションに関する作業

WebSphere for z/OS 管理アプリケーションを使用したステップバイステップのインストール以外に、以下の代替メソッドを使用してアプリケーションを WebSphere for z/OS にインストールすることができます。

情報	参照項目
管理アプリケーションのエクスポート / インポート機能の使用	『管理アプリケーションでエクスポート / インポート・プロセスを使用するためのステップ』
システム管理スクリプト API の使用	117ページの『スクリプトを使用したアプリケーションのインストール』

管理アプリケーションでエクスポート / インポート・プロセスを使用するためのステップ

CORBA アプリケーションのテストを終了した後で、WebSphere for z/OS 管理アプリケーションを使って、テスト・システムで使用した (MOFW) サーバー構成をエクスポートし、実動システムにそのモデル構成をインポートすることができます。このエクスポート・プロセスおよびインポート・プロセスを介して、サーバー定義を含む HFS ファイルを構成し、実動システムに転送します。このプロセスは、スクラッチからのサーバー構成の定義より高速で、エラーが発生しにくくなります。

エクスポート / インポート・プロセスを使用するには、以下のステップを実行します。

1. 管理アプリケーションで、アプリケーションを配置する MOFW サーバーのサーバー・モデルをエクスポートします。
   a. アクティブ・イメージでサーバーを選択する。
   b. 「選択 (Selected)」メニュー・バー選択項目の「**サーバーのエクスポート ... (export server...)**」アクションを選択する。「**サーバーのエクスポート (Export server)**」ダイアログ・ボックスが表示されます。
   c. エクスポート・プロセスの出力を含める HFS ファイルの完全修飾名を入力する。

d. 「**OK**」をクリックする。

結果: WebSphere for z/OS は指定された出力 HFS ファイルを使用して、以下を含め、サーバーに関連した情報をすべて保管します。

- すべてのサーバー・プロパティー
- コンテナー
- アプリケーション・ファミリー
- 論理リソース・マッピング
- アプリケーション
- クライアント・インターフェース
- クラス
- DLL
- ホーム
- 照会およびメタデータ・オブジェクト

---

2. サーバーを稼動させたい z/OS または OS/390 実動システムに、出力 HFS ファイルをコピーまたは移動します。ファイルを移動またはコピーするための方法と説明については、*z/OS UNIX システム・サービス ユーザーズ・ガイド*, SA88-8640 を参照してください。

   警告: 出力 HFS ファイルは編集しないでください。

---

3. 管理アプリケーションで、以下のステップに従って MOFW サーバー・モデルをインポートします。
   a. 必要に応じて、会話を追加する。
   b. 「**サーバー (Servers)**」フォルダーを選択する。
   c. 「選択 (Selected)」メニュー・バー選択項目の「**サーバーのインポート ... (import server...)**」アクションを選択する。「**サーバーのインポート (Import server)**」ダイアログ・ボックスが表示されます。
   d. 「**サーバー名 (Server name)**」に、この WebSphere for z/OS 構成に対する固有の名前を入力する。
   e. 「**入力ファイル (Input file)**」に、実動システムに移動またはコピーした HFS ファイルの完全修飾名を入力する。
   f. 「**OK**」をクリックする。
   g. 「**制御領域 proc 名 (Control region proc name)**」および「**デバッガー許可 (Debugger allowed)**」を含め、サーバーのプロパティーを変更する。
   h. 実動システムに対するサーバー・インスタンスを適宜追加する。

i. 実動システムに対する論理リソースを適宜追加する。

---

4. さらに、管理アプリケーションで、以下のことを実行します。
   a. 会話を選択してから「**検査 (Validate)**」を選択して、インポートされたモデルの妥当性検査を行う。ステータス・バーにメッセージ BBON0442I が表示されると、新規会話は有効です。
   b. 会話を選択してから「**コミット (Commit)**」を選択して、会話をコミットする。「Do you still want to commit? (本当にコミットしますか。)」という質問に、「はい (Yes)」と答えます。ステータス・バーにメッセージ BBON0444I が表示されると、新規会話はコミットされています。
   c. OS/390 作業を適宜完了する。
   d. 会話を選択してから「**活動化 (Activate)**」を選択して、会話を活動化する。「Do you want to activate conversation? (会話を活動化しますか。)」という質問に、「はい (Yes)」と答えます。サーバー定義が活動化されると、ダイアログの下部にメッセージが表示されます。

スクリプトを使用したアプリケーションのインストール

WebSphere for z/OS 管理アプリケーションを使用せずにアプリケーションを MOFW サーバーにインストールする場合は、管理アプリケーションとまったく同じ機能を提供するシステム管理スクリプト API を使用できます。スクリプトを使用することにより、アプリケーションを実動サーバーにインストールするメソッドなど、より高速でエラーの可能性が低いメソッドが提供される場合があります。システム管理スクリプト API の使用については、*WebSphere Application Server V4.0 for z/OS and OS/390: システム管理スクリプト API*, SA88-8657 を参照してください。

第7章 CORBA アプリケーションの活動に関するデータの収集

WebSphere for z/OS では、MOFW サーバーで実行中の CORBA アプリケーションに関する情報を収集するためのさまざまな方式を提供しています。

情報	参照項目
SMF レコードを使用したアカウンティング情報の収集	『SMF レコードを使用した CORBA アプリケーション情報の収集』
JRas サポートを使用した、アプリケーションによるメッセージ発行および項目のトレース機能の使用可能化	『Java アプリケーションのロギング・メッセージおよびトレース・データ』

SMF レコードを使用した CORBA アプリケーション情報の収集

サーバー・アプリケーションに関連した統計を収集および記録する場合は、MOFW サーバーで z/OS または OS/390 システム管理機能 (SMF) を使用することを定義します。SMF 活動およびインターバル・レコードを使用して、MOFW サーバーはアプリケーションのプロファイル作成に使用できるアプリケーション詳細を記録します。SMF 記録機能を使用可能にするには、MOFW サーバー が SMF レコードを作成するように定義し、その他の管理作業を実行します。詳細については、まず *WebSphere Application Server V4.0 for z/OS and OS/390: 操作および管理*, SA88-8653 の SMF のトピックを参照してください。

Java アプリケーションのロギング・メッセージおよびトレース・データ

WebSphere for z/OS ランタイムは、Ras Toolkit for Java をサポートします。これを使用すると、WebSphere for z/OS J2EE または MOFW サーバーで稼動する Java サーバー・アプリケーションからメッセージを発行し、このサーバー・アプリケーションのトレース・データを収集することができます。JRas サポートと呼ばれるツールキットへの WebSphere for z/OS 拡張機能により、Java アプリケーションのメッセージを、MVS マスター・コンソール、あるいは WebSphere for z/OS のエラー・ログ・ストリームまたはコンポーネント・トレース (CTRACE) データ・セット内に表示できます。また、ユーザーのアプリケーションのトレース項目も同じ CTRACE データ・セットに表示できます。

MVS マスター・コンソールにメッセージを発行して、主幹業務アプリケーションの重大なエラー状態を報告することができます。マスター・コンソールを介して、オペレーターはメッセージを受け取り、ユーザーのアプリケーションの状態を示すメッセージに応答して処置を講じます。さらに、メッセージをマスター・コンソールに送信することによって、自動化パッケージを起動して、ユーザーのアプリケーションの処理に関連した特定の条件またはイベントに対して処置を講じることができます。

ユーザーのアプリケーションがコンソールに対して発行するメッセージは、そのメッセージのタイプによって、WebSphere for z/OS の CTRACE データ・セットまたはエラー・ログ・ストリームにも表示されます。これらのシステム・リソース内のメッセージをログに記録すると、ユーザーのアプリケーションの処理に関連したエラーを容易に診断することができます。

同様に、CTRACE データ・セット内のログ・トレース・データに要求を発行することは、エラー状態の記録、すなわち診断目的でのアプリケーション・データの収集のもう 1 つのメソッドです。収集するトレース・データの量とタイプは、選択することができます。したがって、パフォーマンスが優先される場合には最少のトレースでアプリケーションを実行したり、あるいは、問題を再作成して追加の診断情報を収集する場合は詳細にトレースしてアプリケーションを実行することができます。

推奨: エラー・ログ・ストリームである WebSphere for z/OS の CTRACE データ・セットとマスター・コンソールの主な目的は、システム・コンポーネントと重要なアプリケーションの診断データをモニターするか記録することです。インストールの構成によって、アプリケーション・メッセージとデータをこれらのリソースに送信することは、システム・パフォーマンスに悪影響を及ぼす可能性があります。たとえば、アプリケーション・データを CTRACE データ・セットに送信すると、そのデータ・セット内のトレース項目がさらに迅速にラップされる可能性があります。これは、ラップが発生したときにシステムが既存の項目に新規のものを書き込むために、重要な診断データを失う可能性があります。このロギング・サポートの使用は慎重に行ってください。

注:

1. ロギング・メッセージとトレース・データの WebSphere for z/OS サポートを、Java アプリケーション (Java アプレットではなく) のみに使用することができます。
2. Ras Toolkit に対する WebSphere for z/OS サポートは、エンタープライズ版 V3.02 で提供される JRas サポートとは異なります。新しい JRas サポートは以下のとおりです。

- ユーザーのアプリケーションが発行するメッセージを常にログに記録します。この変更は、メッセージを発行してアプリケーションを実行するようにアプリケーションをいったんコーディングすると、そのメッセージが常に収集されログに記録されることを意味します。エンタープライズ版 V3.02 では、メッセージ収集機能をオフにすることができました。
- トレース・データの収集を可能にするために異なる機構が必要です。エンタープライズ版 V3.02 では、MOFW アプリケーション・サーバーの環境変数がトレース・データの収集を制御していました。WebSphere for z/OS V4.0 では、カスタマー提供のトレース設定ファイルによってトレース・データの収集機能を使用可能または使用不可にすることができます。
- メッセージやトレース・ロガーを取得するためにさまざまなクラスを使用しますが、同じメソッド (createRASTraceLogger および createRASMessageLogger メソッド) を使用します。ただし、WebSphere for z/OS V4.0 のメソッドのシグニチャーは、エンタープライズ版 V3.02 のものとは多少異なります。

  エンタープライズ版 V3.02の createRASTraceLogger メソッドと createRASMessageLogger メソッドは使用すべきではありませんが、これらのメソッドを使用するためにコード化したプログラムを変更する必要はありません。ただし、これらのプログラムを OS/390 と同様に他のプラットフォームでも実行する必要がある場合を除きます。WebSphere for z/OS V4.0 では、createRASTraceLogger または createRASMessageLogger への呼び出しは新しい WebSphere for z/OS V4.0 のクラスの同じメソッドに委任されます。Windows NT などの別のプラットフォームでアプリケーションを実行するには、新規のメソッドを使用するためにプログラムを再コード化する必要があります。

  メッセージを発行したりあるいはトレース項目をログに記録したりするために、ユーザーのサーバー・アプリケーションから発行できるメソッドの説明については、*Using the WebSphere JRas Message Logging and Trace Facility* を参照してください。これには `com.ibm.ras` パッケージのメソッドに関する説明が記載されており、z/OS または OS/390 を含めたサポートされるすべてのプラットフォームに適用されます。

以下の表は、Java アプリケーション用のメッセージとトレース・データをログに記録するためのサブタスクおよび関連した手順を示しています。

サブタスク	関連した手順 (参照項目)
発行または収集するメッセージおよびトレース・データのタイプの決定	• 『アプリケーション・メッセージを MVS マスター・コンソールに発行するときのバックグラウンド』 • 125ページの『ユーザーのアプリケーションに対してトレース要求を発行するときのバックグラウンド』
メッセージとトレース要求を発行するための Java サーバー・アプリケーションの作成	127ページの『Java アプリケーションをコード化してメッセージとトレース要求を発行するためのステップ』
メッセージをログに記録しトレース・データを収集するための z/OS または OS/390 ランタイム環境の作成	134ページの『Java アプリケーションのメッセージとトレース要求をログに記録するための z/OS または OS/390 環境を作成するステップ』
Java サーバー・アプリケーション用に収集されたメッセージまたはトレース・データの表示	• 138ページの『メッセージとトレース・データを表示するときのバックグラウンド』 • 139ページの『IPCS をバッチ・モードで使用して、アプリケーション・トレース・データをフォーマットするためのステップ』

アプリケーション・メッセージを MVS マスター・コンソールに発行するときのバックグラウンド

Ras Toolkit のための WebSphere for z/OS ランタイム・サポート (JRas サポート) によって、ユーザーのアプリケーションから MVS マスター・コンソールにメッセージを発行することができます。MVS マスター・コンソールにメッセージを発行して、主幹業務アプリケーションの重大なエラー状態を報告したり、自動化パッケージを起動することができます。ユーザーのアプリケーションが発行するメッセージも、WebSphere for z/OS で使用するコンポーネント・トレース (CTRACE) データ・セット、およびエラー・ログ・ストリーム (メッセージがエラー・メッセージに分類された場合) に表示されます。メッセージをログに記録することは、エラー状態の記録、すなわち、診断目的でのアプリケーション・データの収集のもう 1 つのメソッドです。

WebSphere for z/OS は、メッセージ・ロガーを作成し管理するコードを提供します。メッセージ・ロガーは、アプリケーションのメッセージを処理します。メッセージ・ロガーは、ユーザーの Java アプリケーションが稼動する

WebSphere for z/OS J2EE または MOFW サーバーの Java 仮想マシン (JVM) で稼動します。メッセージ・ロガーを使用するには、Java アプリケーションで以下のことを行う必要があります。

1. メッセージ・ロガーを定義する。
2. メッセージ・ロガーを作成するように WebSphere for z/OS に指示するメソッドを駆動する。
3. アプリケーションの適切なポイントでメッセージをコード化する。

アプリケーションを更新して JRAS サポートを使用するための特別な指示は、127ページの『Java アプリケーションをコード化してメッセージとトレース要求を発行するためのステップ』にあります。ただし、これらの指示を使用してメッセージを適切にコード化できるようになるには、以下のトピックにある概念について理解しておく必要があります。

- 『インライン・メソッドの呼び出しまたはメッセージ・プロパティー・ファイルによるメッセージの定義』
- 124ページの『メッセージ・タイプがメッセージ宛先に及ぼす影響について』

インライン・メソッドの呼び出しまたはメッセージ・プロパティー・ファイルによるメッセージの定義

ユーザーの Java アプリケーションからメッセージを発行する場合は、メッセージをインラインで定義するか、あるいは別々のファイルを使用してメッセージを含めることができます。一般的には、メッセージをインラインで定義する方が速く、完了に必要なステップは少なくなります。さまざまな言語でメッセージ・テキストを提供する予定の場合は、使用可能度とテキスト変換のためには、別々のメッセージ・プロパティー・ファイルを使用する方が適しています。メッセージの定義にファイルを使用するかあるいはインライン方法を使用するかに関係なく、Java アプリケーション内のメソッドをコード化して、処理中の適切なポイントでメッセージを発行しなければなりません。適切なポイントで、`RASIMessageLogger` インターフェースに定義されたメソッドを使用してメッセージを発行します。

メッセージをインラインで定義する場合は、`textMessage` メソッドを使用して、ユーザーのアプリケーションからメッセージを発行します。メソッドの呼び出しで指定する文字列は、メッセージ・ロガーがマスター・コンソール、エラー・ログ・ストリーム、または CTRACE データ・セットに送信する内容です。

メッセージ・プロパティー・ファイルを使用する予定の場合は、以下のことを行う必要があります。

1. メッセージ・プロパティー・ファイルを作成する。

2. キーとテキストの組みを使用してすべてのメッセージを定義する。

   キーを使用すると、メッセージ・ロガーは適切なメッセージをメッセージ・ファイルで見つけることができます。テキストは、メッセージ・ロガーがマスター・コンソール、エラー・ログ・ストリーム、または CTRACE データ・セットに送信する内容です。

3. 適切なメソッドを使用して、アプリケーションのメッセージ用のメッセージ・テキストがある場所をメッセージ・ロガーに知らせます。

   以下の 2 つの機構によって、メッセージ・ロガーのためのメッセージ・ファイルを識別することができます。

   - `setMessageFile` メソッド。このメソッドで、メッセージ・テキストを検索するためのデフォルト・ファイルとして機能する 1 つのメッセージ・プロパティー・ファイルを登録します。
   - `message` メソッドまたは `msg` メソッド。これらのメソッドでは、メッセージ・プロパティー・ファイルの名前を指定することができます。

メッセージ・ファイルを作成するための特別な指示、このファイルでメッセージを定義するための規則、および例については、127ページの『Java アプリケーションをコード化してメッセージとトレース要求を発行するためのステップ』を参照してください。

メッセージ・タイプがメッセージ宛先に及ぼす影響について

メッセージを発行するためのメソッドをコード化するときに、メッセージにタイプを割り当てて、エラー・メッセージ、警告メッセージ、または情報メッセージとしてメッセージの特性を示します。`RASIMessageEvent` インターフェースはメッセージのタイプを定義します。 メッセージの 3 つのタイプすべてがマスター・コンソールと CTRACE データ・セットの両方に送信されます。エラー・メッセージもエラー・ログ・ストリームに表示されます。

注: メッセージは常にログに記録されます。つまり、メッセージを発行してアプリケーションを z/OS または OS/390 で実行するようにアプリケーションをいったんコーディングすると、そのメッセージは常に収集されログに記録されます。

処理時に、メッセージ・ロガーは、アプリケーションが発行する各メッセージに記述子コードと宛先コードを割り当てます。ロガーは、以下の表に示すように、メッセージ・タイプを基にして特定のコードを割り当てます。

ガイドライン: ユーザーのアプリケーション用にメッセージを開発する場合、メッセージの内容とタイプは、以下の表の記述子と宛先コードの説明に従います。各メッセージには 1 つのメッセージ・タイプだけを割り当てます。また、

インストール時に、マスター・コンソールに送信されるメッセージのガイドラインを定義しているかどうか調べてください。

RASIMessageEvent メッセージ・タイプ	記述子コード	宛先コード
TYPE_ERROR (または TYPE_ERR)	**12**: 重要な情報 メッセージには、コンソールには表示しなければならないが、応答のアクションを必要としない重要な情報が含まれています。	**2**: オペレーター情報 このメッセージは、システム状態の変化を示します。アクションを要求しません。逆に、アクションを必要とする可能性がある条件に応じて、マスター・コンソールのオペレーターに警告します。 この宛先コードは、状況がオペレーターの問い合わせによって特に要求されていないときに、ジョブ状況を示す任意のメッセージに使用します。
TYPE_WARNING (または TYPE_WARN)	**12**: 重要な情報 (上記の説明のとおり)	
TYPE_INFORMATION (または TYPE_INFO)	**4**: システム状況 メッセージは、システム・タスクの状況またはハードウェア装置の状況を示します。	

ユーザーのアプリケーションに対してトレース要求を発行するときのバックグラウンド

トレース・データ収集の目的は、十分な情報を提供して、アプリケーションでの問題を診断することです。Ras Toolkit のための WebSphere for z/OS ランタイム・サポート (JRas サポート) によって、ユーザーの Java アプリケーションからトレース要求を発行し、その結果のトレース・データを、WebSphere for z/OS が使用するコンポーネント・トレース (CTRACE) データ・セットに記録させることができます。ユーザーのアプリケーションが稼動する WebSphere for z/OS J2EE サーバーまたは MOFW サーバーの CTRACE データ・セットにアプリケーションのトレース・データが表示されます。

WebSphere for z/OS は、トレース・ロガーを作成し管理するコードを提供します。トレース・ロガーは、アプリケーションのトレース要求を処理します。 トレース・ロガーは、ユーザーの Java アプリケーションが稼動する WebSphere for z/OS J2EE または MOFW サーバーの Java 仮想マシン (JVM) で稼動します。トレース・ロガーを使用するには、Java アプリケーションで以下のことを行う必要があります。

1. トレース・ロガーを定義する。

2. トレース・ロガーを作成するように WebSphere for z/OS に指示するメソッドを駆動する。
3. アプリケーションの適当なトレース・ポイントでトレース要求をコード化する。

アプリケーションを更新して JRAS サポートを使用するための特別な指示は、127ページの『Java アプリケーションをコード化してメッセージとトレース要求を発行するためのステップ』にあります。ただし、これらの指示を使用してトレース要求を適切にコード化できるようになるには、以下のトピックにある概念について理解しておく必要があります。

- 『トレース・ポイントを配置する場所と要求するデータの決定』
- 127ページの『トレース・ポイントへのトレース・タイプの割り当て』

トレース・ポイントを配置する場所と要求するデータの決定

WebSphere for z/OS J2EE または MOFW サーバーで稼動中の Java アプリケーション用のトレース・データを収集するには、アプリケーションのコード内でトレース・ポイントを配置する場所を決定しなければなりません。このトレース・ポイントでは、`RASTraceLogger` クラス・インターフェースを使用してトレース項目を要求することができます。一般的なトレース・ポイントには、以下が含まれます。

- メソッド入り口
- メソッド出口
- 機能要求の開始
- 要求完了のプロセスの主なチェックポイント
- 機能要求の完了
- 他のシステム機能へのインターフェース
- 検出された入出力エラーまたは予期しない例外などの、異常なイベント

トレース項目に記録する情報の内容も決定しなければなりません。このトレース項目は可変データ量を保持できます。WebSphere for z/OS は、作業単位またはトランザクションのアドレス・スペース ID (ASID) とタスク制御ブロック (TCB)、またはスレッドの Java 名を自動的に収集します。以下は、WebSphere for z/OS J2EE または MOFW サーバーで稼動中の Java アプリケーションのトレース項目に入れる可能性がある追加のデータ・タイプを示しています。

- アプリケーションがサービスを提供している作業単位またはトランザクションの識別。これは、JOBNAME、USERID、またはトランザクション ID でもかまいません。
- 機能要求の開始をトレースする項目の場合は、入力パラメーター。
- 内部チェックポイントの場合は、このトレース項目を元の要求に結合する識別、およびプロセスの現在の状況に関する情報。

- 異常なイベントの場合は、問題の原因と任意の追加のデータ。たとえば、あらゆる例外とスタック・トレースを記録することができます。
- サービスからの戻りでは、戻りコードと理由コード。
- デバッグ補助機能ではなく、分析に使用するトレース項目の場合は、アプリケーションのユーザーが必要とするあらゆる情報。

トレース・ポイントへのトレース・タイプの割り当て

ユーザーの Java アプリケーションで定義するそれぞれのトレース・ポイントごとに、RASITraceLogger インターフェースに定義されたメソッドを使用して、トレース項目を要求します。トレース要求の一部として、この特定の要求にトレース・タイプを割り当てる必要があります。RASITraceEvent インターフェースはユーザーが使用できるタイプを定義します。

注: エンタープライズ版 V3.02 JRas サポートでは、ユーザーのアプリケーションでトレース・ポイントにトレース・レベルを割り当てる必要がありました。このような割り当ては現在でもサポートされているため、トレース・レベルを使用するアプリケーションを再コード化する必要はありません。

トレース要求をコード化すると、Java アプリケーションは稼動中にトレース要求を発行することができるようになります。しかし、要求されたトレース・データを実際に記録するには、ユーザーのアプリケーションが稼動する WebSphere for z/OS J2EE サーバーまたは MOFW サーバーでトレースが使用可能でなければなりません。134ページの『Java アプリケーションのメッセージとトレース要求をログに記録するための z/OS または OS/390 環境を作成するステップ』 では、特定のトレース・タイプについてトレース機能を使用可能にする方法について詳しく説明しています。

Java アプリケーションをコード化してメッセージとトレース要求を発行するためのステップ

メッセージを発行するためとトレース項目をログに記録するための指示をコード化することによって、ユーザーの Java サーバー・アプリケーションの信頼性、可用性、保守性 (RAS) を改善することができます。ユーザーの Javaアプリケーションが WebSphere for z/OS J2EE サーバーまたは MOFW サーバーで稼動する場合、そのメッセージは MVS マスター・コンソールと、WebSphere for z/OS で使用するコンポーネント・トレース (CTRACE) データ・セットに表示されます。アプリケーションのトレース項目は、同じ CTRACE データ・セットに表示されます。エラー・メッセージも WebSphere for z/OS で使用するエラー・ログ・ストリームに表示されます。

始める前に:

- ユーザーの Java アプリケーションからメッセージを発行したい場合は、メッセージをインラインで定義するか、あるいは別々のファイルを使用してメッセージを含めることができます。コード化を開始する前に、使用したい方法を決定します。必要な場合は、これらの 2 つの方法に関する 123ページの『インライン・メソッドの呼び出しまたはメッセージ・プロパティー・ファイルによるメッセージの定義』を参照してください。
- メッセージを発行したりあるいはトレース項目をログに記録したりするために使用できる JRas インターフェースおよびメソッドの説明については、*Using the WebSphere JRas Message Logging and Trace Facility* を参照してください。これには `com.ibm.ras` パッケージのメソッドに関する説明が記載されており、z/OS または OS/390 を含むサポートされるすべてのプラットフォームに適用されます。

以下のステップを実行して、ユーザーのサーバー・アプリケーションにコードを追加し、メッセージとトレース項目要求を z/OS または OS/390 メッセージ・ロギング機能とトレース・データ・ロギング機能に送信します。

1. (オプション) ユーザーのアプリケーションからのメッセージをログに記録する場合、およびメッセージをインラインで定義していない場合は、メッセージ・プロパティー・ファイルを作成します。Java アプリケーションが発行するそれぞれのメッセージごとに、キーとテキストを対にしてメッセージを定義します。
   - マスター・コンソールまたはエラー・ログ・ストリームに表示される内容を示すには、テキスト部分を使用します。
   - メッセージ・プロパティー・ファイルと Java アプリケーション・コードの両方に表示される内容を示すには、キーを使用します。これによって、ランタイム・コードで適切なメッセージ・テキストを検出することができます。

   規則:

   - キーをテキストから分離するには、いつでも等号を使用します。例:

     ```
     BBOJ0001=BBOJ0001 Java BO created.
     BBOJ0002=BBOJ0002 Policy number {0} obtained.
     ```

   - 変数データを含むメッセージ・テキストには、配置と内容を示すための特別なコード化が必要です。変数テキストをもつメッセージを正しく定義するには、中括弧 {} を使用して、変数はテキスト内の特別の位置に表示されることを示します。中括弧内では、数字を使用して、この位置に属する変数を示します。

たとえば、ユーザーのコードには、以下の指示が含まれていると想定します。

```
String day = "Monday";
Integer temp = new Integer(75);
msgLogger.message(RASIMessageEvent.TYPE_INFO,
                  this,
                  "methodName",
                  "APPL061I",
                  day,
                  temp);
```

このメッセージを正しく定義するには、メッセージ・プロパティー・ファイルで以下をコード化します。

```
APPL061I=APPL061I On {0}, it is {1} degrees.
```

---

2. ユーザーのアプリケーションに応じた適切なアプリケーション開発ツールを使用して、ユーザーの Java アプリケーションのソース・コードを以下のように編集します。
   - com.ibm.ras パッケージと com.ibm.WebSphere パッケージのインポート・ステートメントを追加する。たとえば、以下を入力します。

     ```
     import com.ibm.ras.*;
     import com.ibm.websphere.ras.*;
     ```

   - メッセージ・ロガーとトレース・ロガーの定義ステートメントを追加する。たとえば、以下を入力します。

     ```
     private RASIMessageLogger msgLogger = null;
     private RASITraceLogger trcLogger = null;
     ```

---

3. Java アプリケーションのコンストラクターを編集して、メッセージ・ロガー、トレース・ロガー、またはその両方を作成します。

ロガーのタイプ	完了するステップ
メッセージ	• createRASMessageLogger メソッドを使用して、メッセージ・ロガーを要求する。 • (オプション) アプリケーションからメッセージを発行するための方法に、インラインではなくファイルを使用している場合には、メッセージ・プロパティー・ファイルを定義する。
トレース	createRASTraceLogger メソッドを使用して、トレース・ロガーを要求する。

規則:

- アプリケーションは、RASIMessageLogger オブジェクト型として createRASMessageLogger メソッドが戻すオブジェクトを参照する必要があります。
- アプリケーションは、RASITraceLogger オブジェクト型として createRASTraceLogger メソッドが戻すオブジェクトを参照する必要があります。

ヒント: 接頭部 com.ibm. で始まるロガー名は使用しないでください。これは WebSphere for z/OS で使用するために予約されています。

---

4. ユーザーの Java アプリケーションからメッセージを発行する場合は、アプリケーションのソース・コードの適切なポイントでメッセージを追加します。

   規則:

   - メッセージをインラインで定義している場合は、RASIMessageLogger インターフェースで textMessage メソッドを使用して、メソッドの呼び出し時に完全なメッセージを文字列で指定する。
   - メッセージ・プロパティー・ファイルを使用している場合は、RASIMessageLogger インターフェースで message メソッドまたは msg メソッドを使用して、メソッドの呼び出し時にメッセージ・キーを指定する。
     例:

     ```
     msgLogger.message(RASIMessageEvent.TYPE_INFO,
                         "com.myCompany.JRasSample",
                         "doSomething",
                          "BBOJ0001");
     ```

   - それぞれのメッセージごとに、124ページの『メッセージ・タイプがメッセージ宛先に及ぼす影響について』のガイドラインの説明に従って適切なタイプを割り当てる。

     注:

     a. 各メッセージには 1 つのメッセージ・タイプだけを割り当てる。
     b. メッセージのタイプを割り当てないか、あるいはタイプに ″null″ を指定すると、Java コンパイラーはエラー・メッセージを発行する。
     c. 無効なタイプを割り当てると、メッセージ・ロガーは監査タイプとしてメッセージを処理する。

- メッセージに使用する各文字は、EBCDIC 文字にマップしなければならない。
- メッセージをマスター・コンソールに経路指定する場合、WebSphere for z/OS はメッセージ・テキストの最初の 700 文字しか送信しない。

制限: エラー・ログ・ストリームにメッセージを書き込む場合、WebSphere for z/OS は、識別のためにメッセージ・テキストに追加する情報を含めて、データの 512 文字しか使用しません (この追加情報は、日付、時刻、組織名、などで構成されます)。アプリケーション・メッセージ用のエラー・ログ・ストリーム項目の形式と内容については、*WebSphere Application Server V4.0 for z/OS and OS/390: メッセージおよび診断*, GA88-8655 を参照してください。

---

5. ユーザーの Java アプリケーション用のトレース・データを収集する場合は、アプリケーションのソース・コードの適切なポイントでトレース要求を追加します。

   規則:

   - それぞれのトレース要求ごとに、RASITraceEvent インターフェースで定義したとおりに適切なタイプを割り当てる。

     注: トレース要求にタイプを割り当てないと、トレース・ロガーはそのトレース要求を無視します。
   - トレース・データに使用する各文字は、EBCDIC 文字にマップしなければならない。

   制限: トレース・データの処理時に、WebSphere for z/OS は 16 進データまたは文字データの限定量しか使用しません。

   - 16 進のトレース・データ (Java バイト配列からの) の場合、WebSphere for z/OS は 1024 バイトより後のデータを切り捨てます。
   - 文字のトレース・データの場合は、そのトレース・データが 16384 文字を超えると、WebSphere for z/OS はリテラル `***BUFFER OVERFLOW***` で置き換えます。この累積制限により、それぞれの文字ストリングごとに 1 バイトのストリング終了文字が組み込まれます。

   ヒント: アプリケーションのパフォーマンスを改善するために、以下の代替のいずれかを使用することができます。

- RASTraceLogger.isLogging 変数のテストでのトレース呼び出しをラップする。トレース・ロギングがアクティブでない場合、この変数は false に設定されます。
- if 文で isLogging メソッドを使用して、トレース・ロギングがトレースのどのレベルでもアクティブであるかどうかをテストする。
- isLoggable メソッドを使用して、指定されたトレース・タイプでロギングがアクティブであるかどうかを判別する。

最初の 2 つの方法を使用すると、トレース・ロギングがアクティブでなくても、トレース項目の作成がオーバーヘッドになることはありません。対照的に、isLoggable メソッドでは、オーバーヘッドを多く必要としますが、特にトレースの一部のレベルがいつでもアクティブである場合に適したオプションとなる場合があります。

---

6. ユーザーの Java アプリケーションに応じた適切なアプリケーション開発ツールを使用して、アプリケーションのコードを生成し、コンパイルします。

Java アプリケーション用の実行可能コードが準備できたら、134ページの『Java アプリケーションのメッセージとトレース要求をログに記録するための z/OS または OS/390 環境を作成するステップ』 にリストされたステップを実行することができます。

例: 次の例は、上述の指示で説明したコーディング要件を示したものです。この例では、com/myCompany/JRasSample.properties という名前のメッセージ・プロパティー・ファイルの使用を想定しています。このファイルには以下のメッセージ定義が含まれています。

```
BBOJ0001=BBOJ0001 Java BO created.
BBOJ0002=BBOJ0002 Policy number {0} obtained.
BBOJ0003=BBOJ0003 Java BO destroyed.
```

```
package com.myCompany;
// Import JRas and Websphere packages
import com.ibm.ras.*;
import com.ibm.websphere.ras.*;
public class JRasSample
{
  // Loggers
  private RASIMessageLogger msgLogger = null;
  private RASITraceLogger trcLogger = null;
  // Message file
  private static final String MSG_FILE = "com.myCompany.JRasSample";
  // Array of trace objects
  Object[] objs = new Object[3];
```

```
// Constructor
public JRasSample()
{
  // Get logger manager object
  Manager manager = Manager.getManager();
  // Get logger
  trcLogger = manager.createRASTraceLogger("com.myCompany","myProduct",
                                           "myComponent","myLogger.COM");
  msgLogger = manager.createRASMessageLogger("com.myCompany","myProduct",
                                           "myComponent","myLogger.COM");
  msgLogger.setMessageFile(MSG_FILE);
}
// Example of JRas trace events and messages
public int doSomething(String parm1,String parm2,String parm3)
{
  int returnValue = 0;
  byte[] byteArray = {1,2,3,4,5};
  // Trace input parameters
  objs[0] = parm1;
  objs[1] = parm2;
  objs[2] = parm3;
  if (trcLogger.isLoggable(RASITraceEvent.TYPE_ENTRY_EXIT))
    trcLogger.entry(RASITraceEvent.TYPE_ENTRY_EXIT,
                    "com.myCompany.JRasSample",
                    "doSomething",
                     objs);
  if (trcLogger.isLoggable(RASITraceEvent.TYPE_MISC_DATA))
  {
    // Trace a text string
    trcLogger.trace(RASITraceEvent.TYPE_MISC_DATA,
                    "com.myCompany.JRasSample",
                    "doSomething",
                     "Text data to be traced");
    // Trace binary data
    trcLogger.trace(RASITraceEvent.TYPE_MISC_DATA,
                    "com.myCompany.JRasSample",
                    "doSomething",
                     byteArray);
    // Trace the current stack
    trcLogger.stackTrace(RASITraceEvent.TYPE_MISC_DATA,
                    "com.myCompany.JRasSample",
                          "doSomething");
  }
  // Issue message to WTO
  msgLogger.message(RASIMessageEvent.TYPE_INFO,
                    "com.myCompany.JRasSample",
                    "doSomething",
                     "BBOJ0001");
  // Issue warning message to WTO
  msgLogger.message(RASIMessageEvent.TYPE_WARN,
                    "com.myCompany.JRasSample",
                    "doSomething",
                     "BBOJ0002",
                     "123");
  // Issue error message to WTO and error log
```

```
    msgLogger.message(RASIMessageEvent.TYPE_ERR,
                      "com.myCompany.JRasSample",
                      "doSomething",
                       "BBOJ0003");
    // Trace return value
    if (trcLogger.isLoggable(RASITraceEvent.TYPE_ENTRY_EXIT))
      trcLogger.exit(RASITraceEvent.TYPE_ENTRY_EXIT,
                      "com.myCompany.JRasSample",
                      "doSomething",
                      returnValue);
    return returnValue;
  }
  // This method is invoked when a JRasSample object is traced
  public String toString()
  {
    String traceString = "This is the JRasSample object trace data";
    return traceString;
  }
  public static void main(String[] args)
  {
    JRasSample sample = new JRasSample();
    sample.doSomething("parm1","parm2","parm3");
  }
}
```

Java アプリケーションのメッセージとトレース要求をログに記録するための z/OS または OS/390 環境を作成するステップ

始める前に:

- WebSphere for z/OS のインストール過程で、エラー・ログ・ストリームとコンポーネント・トレース・データ・セットがセットアップされたかどうか、適切なインストール担当者に問い合わせます。エラー・ログと CTRACE データ・セットがすでに使用可能になっている場合がありますが、その一方で、インストール担当者は、他の WebSphere for z/OS サーバーとアプリケーションからの現行データだけでなく、ユーザーのアプリケーション・データを処理するために、変更する必要があると決定する場合があります。たとえば、インストール時に、すべての WebSphere for z/OS サーバーに共通のエラー・ログ・ストリームか、またはそれぞれの個々のサーバーごとに別々のログ・ストリームのいずれかをセットアップすることができます。インストール時に、共通のログの使用から別々のログの使用に切り替え、ユーザーの Java アプリケーションからの追加の診断データに適用した場合があります。
- J2EE または MOFW サーバーで稼動するアプリケーションに対してトレースをオンにするには、JVM プロパティー・ファイルを編集または作成する必要があります。この作業を実行するには、このファイルを編集したり適切なディレクトリーに保管するための特殊な権限が必要になる場合があります。

ユーザーのシステムに WebSphere for z/OS をインストールしたシステム・プログラマーに問い合わせてください。

注:

1. エラー・ログ・ストリームのセットアップに関する説明は、*WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652 にあります。
2. CTRACE のセットアップと実行に関する説明は、*WebSphere Application Server V4.0 for z/OS and OS/390: メッセージおよび診断*, GA88-8655 にあります。

以下のステップを実行して、ロギング・サポートのための z/OS または OS/390 環境をセットアップします。

1. z/OS または OS/390 で、階層ファイル・システム (HFS) にトレース設定ファイルを作成します。この作業は、WebSphere for z/OS J2EE サーバーまたは MOFW サーバーでアプリケーションのトレース・データを収集およびログに記録できるようにしたい場合に行います。このファイルで、使用するトレース設定を次の形式で入力します。

   logger_name=*type*=[enabled|disabled]

   例:

   ```
   myLogger.COM=all=enabled
   ```

   logger_name は、トレース・ロガーを取得するために create メソッドをコード化したときに、アプリケーションのソース・コードに指定したロガー名と一致させます。複数のロガー名に対してロギング・サポートを使用可能にするには、各ロガー名の全体を略さずに指定する代わりに、共通する接頭部にアスタリスクを付けて指定することができます (たとえば `a.b.c.*`)。`a.b.c.*` のように指定すると、`a.b.c.d` および `a.b.c.e` という名前のロガーに対してロギングが可能になります。

   ヒント: 接頭部 `com.ibm.` で始まるロガー名は使用しないでください。これは WebSphere for z/OS で使用するために予約されています。

   type は以下の表のプロパティー値のいずれかに対応します。プロパティー・タイプでは大文字と小文字が区別されます。

表 9. トレース設定プロパティー・タイプおよびそれに対応する JRas トレース・タイプ

指定する プロパティー・タイプ	トレースを使用可能にする JRas トレース・タイプ
all	サポートされるすべての RASITraceEvent タイプ
event	• RASITraceEvent.TYPE_ERROR_EXC • RASITraceEvent.TYPE_SVC • RASITraceEvent.TYPE_OBJ_CREATE • RASITraceEvent.TYPE_OBJ_DELETE • RASITraceEvent.TYPE_LEVEL1
entryExit	• RASITraceEvent.TYPE_ENTRY_EXIT • RASITraceEvent.TYPE_API • RASITraceEvent.TYPE_CALLBACK • RASITraceEvent.TYPE_PRIVATE • RASITraceEvent.TYPE_PUBLIC • RASITraceEvent.TYPE_STATIC • RASITraceEvent.TYPE_LEVEL1 • RASITraceEvent.TYPE_LEVEL2
debug	• RASITraceEvent.TYPE_MISC_DATA • RASITraceEvent.TYPE_LEVEL1 • RASITraceEvent.TYPE_LEVEL2 • RASITraceEvent.TYPE_LEVEL3

規則:

- 所定のロガーに対して、同じトレース・プロパティー・ファイルを使用して異なるトレース・タイプを使用可能にすることも可能です。ロガーのトレース・タイプごとに別々の行を使用して定義しない場合は、単一のコロン (:) を使用して各ロガーのトレース設定を区別する必要があります。

  例 (ロガーごとに別々の行を使用する場合):

  ```
  com.aCompany.*=all=enabled
  com.anotherCompany.*=event=enabled
  ```

  例 (各ロガーを同じ行で定義する場合):

  ```
  com.aCompany.*=all=enabled:com.anotherCompany.*=event=enabled
  ```

- 複数のトレース・タイプをロガーに指定するには、それぞれのトレース・タイプをコンマ (,) で区切ってください。

  例:

  ```
  com.aCompany.aComponent=debug=enabled,event=enabled
  ```

2. 作成したトレース設定ファイルを指すように、新規の Java 仮想マシン (JVM) プロパティー・ファイルを作成するか、または既存の Java 仮想マシン (JVM) プロパティー・ファイルを編集します。jvm.properties という名前のこのプロパティー・ファイルは、WebSphere for z/OS J2EE サーバーまたは MOFW サーバーで稼動する JVM のデフォルト設定を変更します。

   規則:

   - com.ibm.ws390.trace.settings システム・プロパティーは、トレース設定ファイルの完全修飾ディレクトリー・パスおよびファイル名に設定する必要があります。このシステム・プロパティーを指定しない場合、または誤ったパスおよびファイル名を指定した場合は、すべてのトレース・タイプが使用不可になります (デフォルト設定)。
   - サーバーの起動時にプロパティー設定を検索して使用できるようにするため、WebSphere for z/OS から jvm.properties ファイルにアクセスできるようにしておく必要があります。jvm.properties ファイルは、Java アプリケーションが実行されるサーバーの環境変数設定値を含む current.env ファイルが WebSphere for z/OS によって入れられた HFS ディレクトリーに入れてください。このディレクトリーの詳細については、149ページの『付録A. 環境ファイル』 を参照してください。
   - トレース・ロギングを動的に開始したり、停止したりすることはできません。

3. J2EE サーバーまたは MOFW サーバーによるコンポーネント・トレースの使用に関連した環境変数の設定値を検査してください。値の一部を変更して、CTRACE データ・セット内の追加のトレース項目に適用することができます。特に、以下の環境変数の設定値を検査してください。
   - TRACEBUFFCOUNT
   - TRACEBUFFSIZE

4. アプリケーションが実行される WebSphere for z/OS J2EE サーバーまたは MOFW サーバーを始動します。
   - すでにサーバーにインストール済みの既存のアプリケーションに対して JRas サポートをセットアップした場合は、次のことを行ってください。
     a. 新規にコンパイルしたコードによって既存のコードが置き換えられていることを確認する。

b. jvm.properties ファイルまたは環境変数に対して行った変更を WebSphere for z/OS サーバーがピックアップしていることを確認する。サーバーを一度停止してから再始動して、これらの変更内容をピックアップする必要があります。

- 新規のアプリケーションに対して JRas サポートをセットアップした場合は、適切なプロセスに従って Java アプリケーションをアセンブルし、WebSphere for z/OS サーバーにインストールします。Java アプリケーションを MOFW サーバーにインストールする場合は、85ページの『第4章 WebSphere for z/OS MOFW における CORBA アプリケーションの配置』を参照してください。

メッセージとトレース・データを表示するときのバックグラウンド

Java アプリケーションが稼動を開始すると、以下のように、そのメッセージとトレース・データを表示することができます。

表示に使用する出力のタイプ	使用する指示
MVS マスター・コンソール上のメッセージ	メッセージ・ロガーは、メッセージを読み取り可能な形式で MVS マスター・コンソールに自動的に経路を定めます。メッセージの外観と期間は、インストール時のコンソール構成のセットアップの仕方によって異なります。必要な場合には、メッセージ表示の制御、スクロール、および削除を含めた、コンソールの構成方法に関する説明について、*z/OS MVS 計画: 操作*, SA88-8573 を参照してください。
エラー・ログ・ストリーム内のメッセージ	エラー・ログ・ストリームにメッセージを表示するには、ログ・ブラウズ・ユーティリティー (BBORBLOG) を使用します。ログ・ブラウズ・ユーティリティーの使用方法およびメッセージ出力の例については、*WebSphere Application Server V4.0 for z/OS and OS/390: メッセージおよび診断*, GA88-8655 を参照してください。

表示に使用する出力のタイプ	使用する指示
コンポーネント・トレース内のメッセージまたはトレース・データ	コンポーネント・トレースにメッセージまたはアプリケーション・トレース・データを表示するには、以下の方法のいずれかで、対話式問題制御システム (IPCS) を使用しなければなりません。 • 端末でのライン・モード (IPCS CTRACE コマンド) • 端末でのフルスクリーン・モード (IPCS ダイアログ) • 端末モニター・プログラムを使用したバッチ・モード **推奨:** IPCS、TSO/E および ISPF に精通していない場合は、『IPCS をバッチ・モードで使用して、アプリケーション・トレース・データをフォーマットするためのステップ』で説明するように、バッチ・モードの IPCS を使用して、トレース・データのフォーマットと表示を行います。 IPCS ダイアログの使用方法およびメッセージとトレース・データ出力の例については、*WebSphere Application Server V4.0 for z/OS and OS/390: メッセージおよび診断*, GA88-8655 を参照してください。

注: ユーザーの Java アプリケーションのトレース・データを表示する場合、メッセージと CTRACE レコードは、ユーザーのアプリケーションがメッセージとトレース要求を発行した順序で表示されない可能性があります。すべてのメッセージ要求は、お互いに関連した順次配列で表示されます。同様に、すべての CTRACE レコードも、お互いに関連した順序で表示されます。ただし、トレース・データのタイプが異なると、順次に表示されない可能性があります。たとえば、トレース要求の後に発行されたメッセージが、トレース要求の前のトレース出力に表示される可能性があります。

IPCS をバッチ・モードで使用して、アプリケーション・トレース・データをフォーマットするためのステップ

コンポーネント・トレースからのメッセージまたはアプリケーション・トレース・データを表示するには、対話式問題制御システム (IPCS) を使用してデータをフォーマットしなければなりません。IPCS、TSO/E、および ISPF の使用経験が少ない場合には特に、バッチ・モードで IPCS を使用することは、データをフォーマットする最も簡単な方法です。バッチ・モードによって、IPCS を使用してトレース・データをフォーマットしたり、このトレース・データを

MVS データ・セットに書き込むことができます。オプションで、そのデータ・セットの内容を HFS ファイルにコピーして、表示することができます。

始める前に: IPCS をバッチ・モードで使用できるようにする前に、IPCS ダンプ・ディレクトリーを作成しなければなりません。IPCS のセットアップ時に、インストールにより、IPCS をユーザーのためにカスタマイズすることができます。このカスタマイズには、IPCS ダンプ・ディレクトリーを作成するために、IBM 提供の BLSCDDIR CLIST のデフォルト値を変更することを組み込むことができます。

インストールにより BLSCDDIR CLIST が変更されている場合は、以下のステップを実行して、IPCS ダンプ・ディレクトリーを作成します。

1. このディレクトリー用の完全修飾データ・セット名を決定する。
2. TSO/E コマンド・プロンプトから、BLSCDDIR コマンドを入力して、データ・セット名を指定する。たとえば、IBMUSER.DDIR という名前のダンプ・ディレクトリーを作成するには、以下を入力します。

   ```
   %blscddir dsn('ibmuser.ddir')
   ```

インストール時に、IPCS をカスタマイズしていない場合は、他の BLSCDDIR CLIST パラメーターを更新する必要がある場合があります。 BLSCDDIR CLIST を使用してダンプ・ディレクトリーを作成することの詳細については、*z/OS MVS IPCS ユーザーズ・ガイド*, SA88-8568 および *z/OS MVS IPCS コマンド*, SA88-8566 を参照してください。

IPCS をバッチ・モードで使用して、アプリケーション・トレース・データをフォーマットするには、以下のステップを実行します。

1. ファイルを作成し、以下のサンプル JCL をそのファイルにコピーします。この JCL が IPCS を呼び出し、JRAS トレース・データを抽出してフォーマットし、そのトレース・データを MVS データ・セットに書き込んだあと、TSO/E OPUT コマンドを使用してフォーマットされたデータを MVS データ・セットから HFS ファイルに書き込みます。

   ```
   //IBMUSERX   JOB ,
   // CLASS=J,NOTIFY=&SYSUID,MSGCLASS=H
   //IPCS       EXEC PGM=IKJEFT01,REGION=4096K,DYNAMNBR=50
   //IPCSDDIR   DD DSN=IBMUSER.DDIR,DISP=SHR
   //IPCSDOC    DD SYSOUT=H
   //JRASTRC    DD DSN=IBMUSER.CB390.CTRACE,DISP=SHR
   //IPCSPRNT   DD DSN=IBMUSER.IPCS.OUT,DISP=OLD
   //SYSTSPRT   DD SYSOUT=*
   //SYSTSIN    DD *
   IPCS
   DROPDUMP DDNAME(JRASTRC)
   PROFILE LINESIZE(80)PAGESIZE(99999999)
   ```

```
SETDEF NOCONFIRM
CTRACE COMP(SYSBBOSS) DDNAME(JRASTRC) FULL PRINT +
       NOTERMINAL
DROPDUMP DDNAME(JRASTRC)
END
/*
//OPUT       EXEC PGM=IKJEFT01,REGION=4096K,DYNAMNBR=50
//SYSTSPRT   DD SYSOUT=*
//SYSTSIN    DD *
oput 'ibmuser.ipcs.out' '/u/ibmuser/ipcs/jrastrace.txt' TEXT
/*
```

---

2. サンプル JCL を編集し、IBMUSER.DDIR を、作成した IPCS ダンプ・ディレクトリーに使用したデータ・セット名と置き換えます。

   注:

   a. 出力データ・セットを印刷しない場合にのみ、PROFILE ステートメントで PAGESIZE パラメーターを使用します。

   b. HFS ファイル名を既存の HFS ファイルの名前と置き換えることができますが、そのようにする必要はありません。指定したファイルがない場合には、OPUT コマンド処理により新規 HFS ファイルが作成され、読み取りアクセスと書き込みアクセスがユーザーのユーザー ID にのみ認可されます。

      既存の HFS ファイルを指定する場合は、OPUT コマンド処理により、そのファイル内にすでにあるあらゆるデータが上書きされます。OPUT コマンドについて詳しく知りたい場合は、*z/OS UNIX システム・サービス コマンド解説書*, SA88-8641 を参照してください。

   c. 例の JRASTRC DD で指定されたデータ・セット名を、CTRACE データが入っているデータ・セットの名前に変更してください。

   d. 必要な場合には、SYSTSIN ストリーム内の JRASTRC DD ステートメントと OPUT コマンドの両方にある MVS データ・セットの名前を変更してください。まず、JRAS CTRACE データのフォーマット済み出力が IPCSPRNT DD ステートメントで指定された MVS データ・セットに書き込まれたあと、(オプションで) HFS データ・セットにコピーされます。このデータ・セットを事前に割り当てるか、あるいは、サンプル JCL を変更してデータ・セットを割り当てるか、いずれかを行わなければなりません。このデータ・セットのレコード・フォーマットは VBA であり、レコード長は 133 でなければなりません。

---

3. JCL に IPCS バッチ・ジョブを開始するよう実行依頼します。

完了すると、vi などの UNIX エディターを使用して、HFS ファイル内のトレース・データを表示することができるようになります。UNIX エディターについて詳しく知りたい場合は、*z/OS UNIX システム・サービス ユーザーズ・ガイド*, SA88-8640 を参照してください。

第8章 CORBA アプリケーションの WebSphere for z/OS へのマイグレーション

以下のセクションには、アプリケーションの WebSphere for z/OS V4.0 MOFW サーバーへのマイグレーションに関する情報が記載されています。既存のアプリケーションの WebSphere for z/OS J2EE サーバーへのマイグレーションについては、*WebSphere Application Server V4.0 for z/OS and OS/390: J2EE アプリケーションのアセンブル*, SA88-8654 を参照してください。

Component Broker for Windows NT からのアプリケーションのマイグレーションに関するバックグラウンド

WebSphere Application Server web サイトのライブラリー・セクションでは、Component Broker for Windows NT と共に出荷されるプログラミング・サンプルを、WebSphere for z/OS サーバーで実行されるように更新するための手順を説明しています。WebSphere for z/OS ランタイム環境にマイグレーションしたいアプリケーションの場合は、これらのプログラミング・サンプルをモデルとして使用してください。

WebSphere for z/OS サーバーに既存のサーバー・アプリケーションを配置することにした場合は、区分データ・セット (PDS) メンバーの z/OS または OS/390 命名規則に従うように、コンポーネント・オブジェクト、クラス、またはスキーマのいくつかを名前変更しなければならない場合があります。これらの項目によっては、アプリケーション開発ツールにより、プロンプトが出されて名前を変更するか、名前を短くします。以下の場合は、z/OS または OS/390 PDS メンバーの命名規則を使用しなければなりません。

- パーシスタント・オブジェクト・クラス名
- データベース・テーブル名
- データベース・ユーザー名
- マネージド・オブジェクトの IR 実行可能名
- サーバー側 DLL ファイルの名前。z/OS または OS/390 にも配置される場合は、クライアント側 DLL ファイルの名前。

z/OS または OS/390 PDS メンバー命名規則は、以下のとおりです。

- 名前の長さは 8 文字を超えてはいけない。
- 名前は 英字で始めなければならない。
- 残りの文字は英数字だけにすることができる。

WebSphere Application Server for OS/390 エンタープライズ版 V3.02 で稼動するアプリケーションのマイグレーション・シナリオ

145ページの表10 に WebSphere Application Server for OS/390 エンタープライズ版 V3.02 で現在実行されているアプリケーションについて可能なマイグレーション例をリストします。さらに詳細な情報が必要な場合は、以下の情報源をご利用ください。

- Sun Microsystems Web サイト (`java.sun.com`)。ソフトウェア開発キット (SDK: Software Development Kit) の各レベル間の相違点、および Enterprise bean の仕様間の相違点について詳しく説明しています。
- *WebSphere Application Server V4.0 for z/OS and OS/390: J2EE アプリケーションのアセンブル*, SA88-8654。WebSphere for z/OS J2EE サーバーのアプリケーションの作成、アセンブル、インストールに関する作業の説明やガイドラインが必要な場合に参照してください。

注: 現在 WebSphere Application Server エンタープライズ版 for OS/390 V3.02 を実行しているのと同じ z/OS または OS/390 システム・イメージ、または同じシスプレックス内に WebSphere for z/OS J2EE サーバーをインストールすることは**できません**。

表 10. WebSphere Application Server for OS/390 エンタープライズ版 V3.02 アプリケーションのためのマイグレーション作業の要約

マイグレーションのシナリオ	説明
エンタープライズ版 V3.02 Enterprise bean の V4.0 へのマイグレーション	**推奨:** Enterprise bean の V3.02 サポートは制限されていたため、V4.0 J2EE サーバーによって提供される新規のサポート機能に対応するように、新規の Enterprise bean を設計およびコーディングするよう考慮してください。 **規則:** V4.0 の WebSphere for z/OS は、2 種類のサーバー、すなわち新規の J2EE サーバー、および CORBA アプリケーション用の V3.02 マネージド・オブジェクト・フレームワーク (MOFW: managed-object framework) に基づくサーバーをサポートします。Enterprise bean を V4.0 (MOFW) サーバーで実行することはできません。 V3.02 の bean を保持し、それを V4.0 J2EE サーバー・サポートに対応できるようにアップグレードする必要がある場合は、次の作業を実行します。 1. bean ソースを適切な開発ツールにインポートします。 **注:** ソース・コードにアクセスできない場合は、bean の JAR ファイルをインポートして使用すべきでないインターフェースを調べることができます。該当するものがない場合は、ステップ 4 に進みます。 2. **推奨:** 新規の SDK レベルまたは EJB 仕様において使用すべきでない関数 (ある場合) を除去するようコードを再作成してください。 3. エンティティー bean の場合は、コンテナー管理または bean 管理のパーシスタンスに関する要件を検討します。 4. bean コードを再生成し、作成された JAR ファイルをエクスポートします。 5. WebSphere for z/OS アプリケーション組み立てツールを使用して JAR ファイルをインポートし、bean を Enterprise archive (EAR) ファイルにパッケージします。 6. WebSphere for z/OS 管理アプリケーションを使用して EAR ファイルをインストールします。

表 10. WebSphere Application Server for OS/390 エンタープライズ版 V3.02 アプリケーションのためのマイグレーション作業の要約 (続き)

マイグレーションのシナリオ	説明
エンタープライズ版 V3.02 Java ビジネス・オブジェクトの V4.0 Enterprise bean へのマイグレーション	**推奨:** このマイグレーション・パスは、Java BO を後に Enterprise bean に変換する意図で、*WebSphere Application Server エンタープライズ版 for OS/390 Component Broker アプリケーション・アセンブル・ガイド バージョン 3.02*, GA88-7007 に記載されたガイドラインに従うように設計およびコーディングする場合にのみ使用してください。それ以外の場合、移行作業は新規の Enterprise bean を最初からコーディングするよりも複雑で困難になる可能性があります。 Java BO を Enterprise bean に変換するには、以下を行います。 1. CORBA および EJB のプログラミング・モデルと、新規の SDK レベルの違いを理解します。 2. 新規の Enterprise bean を適切な開発ツールに作成します。Java BO のビジネス・ロジックをコピーし、必要に応じてそれを変更します。 • **推奨:** 新規の SDK レベルまたは EJB 仕様において使用すべきでない関数 (ある場合) を除去するようコードを再作成してください。 • エンティティー bean の場合は、コンテナー管理または bean 管理のパーシスタンスに関する要件を検討します。 • 必要に応じてその他のコードを再作成します。その際、以下の点に特に注意してください。 – インプリメンテーションの継承 – 他のオブジェクトまたは CORBA オブジェクトを扱うヘルパー・クラス – 環境によって変化する可能性のあるプロパティーへの参照を含んでいるコード – 他のオブジェクトとの関連を持つオブジェクトの関連コード 3. bean コードを生成し、作成された JAR ファイルをエクスポートします。 4. WebSphere for z/OS アプリケーション組み立てツールを使用して JAR ファイルをインポートし、bean を Enterprise archive (EAR) ファイルにパッケージします。 5. WebSphere for z/OS 管理アプリケーションを使用して EAR ファイルをインストールします。

表 10. WebSphere Application Server for OS/390 エンタープライズ版 V3.02 アプリケーションのためのマイグレーション作業の要約 (続き)

マイグレーションのシナリオ	説明
エンタープライズ版 V3.02 Java ビジネス・オブジェクトをマイグレーションして V4.0 (MOFW) サーバーで実行できるようにする。	Java BO をアップグレードして V4.0 サーバーで実行するには、以下を実行します。 1. bean ソースを適切な開発ツールにインポートします。 **規則:** Java BO にはオブジェクト・ビルダー 3.5 を使用してください。 2. **推奨:** 新規の SDK レベルにおいて使用すべきでない関数 (ある場合) を除去するようコードを再作成してください。 3. エンタープライズ版 V3.02 MOFW サーバーの Enterprise bean の駆動に使用されたコードがこのオブジェクトに含まれている場合は、以下の機能を提供するコードを変更する必要があります。Enterprise bean は現在 V4.0 J2EE サーバーにある必要があるため、bean を使用するためのプログラミング・モデルは変更されています。 • 初期コンテキストの設定 • ホームの検索と使用 • オブジェクト参照子の限定 4. コードを再生成し、作成された JAR ファイルをエクスポートします。 5. アプリケーション・ファイルを z/OS または OS/390 に転送し、`all.mak` ファイルを実行してコードを再コンパイルします。 6. WebSphere for z/OS 管理アプリケーションを使用して、アプリケーションを V4.0 (MOFW) にインストールします。
エンタープライズ版 V3.02 C++ ビジネス・オブジェクトをマイグレーションして V4.0 (MOFW) サーバーで実行できるようにする。	既存の V3.02 C++ ビジネス・オブジェクトのソース・コードを変更する必要はまったくありませんが、コンパイラーまたはオブジェクト・ビルダーに対するサービス更新や拡張を利用するために、アプリケーション開発ツールを使用してこのソース・コードを再度利用することを検討してください。 **規則:** • C++ ビジネス・オブジェクトにはオブジェクト・ビルダー 3.5 を使用してください。 • C++ ビジネス・オブジェクトを z/OS または OS/390 で再コンパイルする必要があります。 C++ ビジネス・オブジェクトをアップグレードして V4.0 サーバーで実行できるようにするには、以下を実行します。 • `all.mak` ファイルを再実行して、コードを z/OS または OS/390 で再コンパイルします。 • 管理アプリケーションを使用して、アプリケーションを V4.0 (MOFW) サーバーにインストールします。

表 10. WebSphere Application Server for OS/390 エンタープライズ版 V3.02 アプリケーションのためのマイグレーション作業の要約 (続き)

マイグレーションのシナリオ	説明
CORBA サーバー・アプリケーションを駆動する「ネイティブ」 z/OS または OS/390 クライアント・アプリケーションのマイグレーション	CORBA サーバー・アプリケーションを使用するために、CORBA プログラミング・モデルを使用する既存の C++ または Java クライアントを変更する必要はありません。ただし、コンパイラーやツールに対するサービス更新や拡張を利用するために、アプリケーション開発ツールを使用してこれらのクライアントを再度利用することを検討してください。
Enterprise bean を駆動する「ネイティブ」z/OS または OS/390 クライアント・アプリケーションのマイグレーション	Enterprise bean は J2EE サーバー にある必要があるため、bean を使用するためのプログラミング・モデルは変更されています。「ネイティブ」z/OS または OS/390 クライアント・アプリケーションが CORBA プログラミング・モデルを使用して bean にアクセスしている場合は、ソース・コードを変更する必要があります。そのためには、以下の手順に従ってください。 1. bean ソースを適切な開発ツールにインポートします。 2. **推奨:** 新規の SDK レベルにおいて使用すべきでない関数 (ある場合) を除去するようコードを再作成してください。 3. 次の機能を提供するコードを変更します。 • 初期コンテキストの設定 • ホームの検索と使用 • オブジェクト参照子の限定 4. コードを再生成し、作成された JAR ファイルをエクスポートします。 5. コードを z/OS または OS/390 で再コンパイルします。
分散クライアント・アプリケーションのマイグレーション	以下の基準の両方に該当するクライアントを変更する必要はありません。 • MOFW サーバーがインストールされている C++ または Java ビジネス・オブジェクトを使用している。 • ワークステーション・プラットフォーム (Windows NT など) 上の、WebSphere Application Server エンタープライズ版 Component Broker バージョン 3.0、V3.5、または V3.6 で実行されている。

付録A. 環境ファイル

この付録には、環境ファイルと環境変数に関する参照情報を記載しています。

環境ファイルおよび環境変数

このセクションでは、以下のトピックについて説明します。

- WebSphere for z/OS による環境変数と環境ファイルの管理方法
- ランタイム・サーバー開始プロシージャーによる環境ファイルの指示方法
- OS/390 クライアント向け環境変数
- ランタイム環境変数の構文および意味

注: アプリケーション開発環境に追加の環境変数を設定することが必要になることがあります。

WebSphere for z/OS による環境変数と環境ファイルの管理方法

インストールおよびカスタマイズ中のブートストラップ・プロセスのあとで、WebSphere for z/OS は管理アプリケーションを介して環境データを管理し、システム管理データベースにその環境データを書き込みます。環境変数を追加または変更するには、シスプレックス、サーバー、またはインスタンス・プロパティー形式に環境データのペア (環境変数名とその値) を入力する必要があります。会話を活動化したり、コールド・スタートのための準備を行うと、環境変数データが HFS ファイルに書き込まれます。WebSphere for z/OS によって、環境ファイルを最も明確に指定する値が決定されます。たとえば、サーバー・インスタンスに対する設定は、そのサーバーに対する同じ変数の設定に優先し、サーバーに対する設定は、そのシスプレックスに対する同じ変数の設定に優先します。

管理アプリケーションを介さずに環境ファイルを直接変更すると、行われた変更は、会話を活動化したり、コールド・スタートのために準備したりするときに、すべて上書きされます。

会話を活動化したり、コールド・スタートのための準備を行うと、WebSphere for z/OS は環境データを各サーバー・インスタンスの HFS ファイルに書き込みます。各環境ファイルのパスと名前は、次のようになります。

CBCONFIG/controlinfo/envfile/*SYSPLEX*/*SRVNAME*/current.env

この場合、次のようになります。

CBCONFIG
インストール時に、WebSphere for z/OS による構成データと環境ファイルの書き込み先ディレクトリーとして指定する、読み取りおよび書き込みディレクトリーです。デフォルトは、`/WebSphere390/CB390` です。

推奨: システム管理サーバー領域のユーザー ID (この BBOCBRAC サンプルでは CBSYMSR1) は、`/WebSphere390/CB390` ディレクトリーの所有者でなければなりません。システム管理サーバー領域によって、このディレクトリーにファイルが書き込まれます。他のサーバー領域のユーザー ID がそのディレクトリーへの読み取りアクセス権を持つように、許可ビットを 775 にする必要があります。

SYSPLEX
ユーザーのシスプレックスの名前です。WebSphere for z/OS は、事前定義された &SYSPLEX JCL 変数からこの名前を派生させます。

SRVNAME
サーバー・インスタンス名です。

環境変数は、WebSphere for z/OS の初期インストールの場合を除き、管理アプリケーションを介して管理する必要があります。初期インストール時には、ブートストラップ・ジョブが使用する初期環境ファイルを変更する必要があります。環境変数ファイルを直接変更しなければならないのは、このときだけです。

したがって、ユーザーのサーバー向けに環境データを定義する明確な状態は、2 つあります。これらの状態に一致するのは、以下のようにして環境データを作成する、2 つの明確な方法です。

1. ブートストラップ・プロセスより前に環境変数をコーディングすることによって、環境データを定義する。この状態では、ここで与えられるサンプルを変更します。ブートストラップ・ジョブによって、ファイルが読み取られ、環境データがシステム管理データベース内に置かれます。HFS で環境ファイルを直接変更するのは、このときだけです。

   環境変数の構文については、152ページの『環境変数構文』を参照してください。
2. 管理アプリケーションを介して環境データを定義し、管理する。この状態では、管理アプリケーション内のパネルを介して環境データのペア (環境名とその値 － no ″=″) を入力します。

ランタイム・サーバー開始プロシージャーによる環境ファイルの指示方法

WebSphere for z/OS ランタイム・サーバー開始プロシージャーは、構成情報に対する環境ファイルを示す必要があります。開始プロシージャーは、HFS ファイルを示す PATH パラメーターを持つ BBOENV DD ステートメントを使用します。BBOENV DD ステートメントは、次のようになります。

```
//BBOENV    DD   PATH='&CBCONFIG/&RELPATH/&SYSPLEX/&SRVNAME/current.env'
```

この場合、次のようになります。

&CBCONFIG
開始プロシージャーの中に設定する変数です。インストール時に、WebSphere for z/OS による構成データと環境ファイルの書き込み先ディレクトリーとして指定する、読み取りおよび書き込みディレクトリーと一致していなければなりません。デフォルトは、CBINTFRPWebSphere390/CB390 です。

&RELPATH
サブディレクトリー (controlinfo/envfile) です。値は変更しないでください。

&SYSPLEX
ユーザーのシスプレックスの名前です。事前定義された JCL 変数であるため、開始プロシージャーの中で設定する必要はありません。

&SRVNAME
サーバー・インスタンス名です。プロシージャーの開始時にサーバー・インスタンス名を指定することによって、他のサーバー・インスタンスに対しても同じ開始プロシージャーを使用することができます。

例: サーバー・インスタンス名 BBOASRIA をその開始プロシージャーに渡すには、次のように指定します。

```
s bboasr1.bboasr1a,srvname='BBOASR1A'
```

サーバー・インスタンス BBOASR1B に対して同じ開始プロシージャーを使用するには、次のように指定します。

```
s bboasr1.bboasr1b,srvname='BBOASR1B'
```

OS/390 クライアント向け環境変数

管理アプリケーションは、OS/390 クライアント向けの環境変数を管理しません。OS/390 クライアント環境ファイルを作成、管理し、クライアント・プログラムからそれらを示す必要があります。154ページの表11 は、OS/390 クライアントに対する必須、またはオプションの環境変数を示しています。

置換変数使用上の注意事項

環境ステートメント内で変数置換 ($ 変数) を使用することはできません。UNIX シェル環境で使用される変数置換は、言語環境プログラム (LE) に実装されていません。WebSphere for z/OS は、言語環境プログラム内で環境変数を処理するため、パス環境変数内で $PATH などの変数を使用すると、失敗します。

例:

UNIX シェル環境は、多くの場合、次のように、既存のパスに新しいパスを追加することによってパスを設定します。

```
PATH=yourdir
PATH=$PATH/mydir
```

$PATH 変数を置き換えた後、結果として設定されたパスは、PATH=yourdir/mydir です。しかし、WebSphere for z/OS は言語環境プログラムで環境変数を処理するため、変数が割り当てられていないと、結果として設定されるパスは PATH=$PATH/mydir になります。

環境変数構文

この構文に従うのは、ブートストラップ・プロセスの前に初期環境ファイルを定義する場合だけです。

規則: 構文規則を以下に示します。

- 環境変数の構文は、次のようなパターンに従います。

  ```
  VARIABLE=VALUE
  ```

  この場合、次のようになります。

  VARIABLE
  環境変数です。

  VALUE
  変数に対する設定です。説明によって、変数ごとの可能な値が定義されています。

- 変数と値の両方について、先行および末尾の空白文字 (ブランクまたはタブ) は無視されます。**例:** 以下の 2 つの行は同じ結果になります。

  ```
  VARIABLE1=VALUE1
  ```

  および

  ```
        VARIABLE1      =      VALUE1
  ```

- 変数を空にすることはできません。また、最初の文字は英字にしてください。
- 『=』 が必要です。
- 値を空にすることは**できません**。空白以外の文字を最低でも 1 つ指定する必要があります。そうでない場合は、環境変数は無視されます。
- ブランク行は無視されます。
- このトピックの説明に従って、大文字と小文字をコーディングします。
- 環境変数をコメント化するには、変数に '#' などの文字を追加するだけです。たとえば、TRACEALL=0 を #TRACEALL=0 に変更できます。変数が英字で始まっていないため、システムはこのようなコーディングを無視します。

環境変数の使用

サーバーまたはクライアントごとに、すべての環境変数を使用する必要があるというわけではありません。154ページの表11 は、与えられた環境変数をどこで使用するかを示しています。以下は、各列に表示される内容の意味です。

- 『R』 は必須であることを意味します。
- 『O』 はオプショナルであることを意味します。
- 『F』 は今後のリリースにおいて必須であることを意味します。
- 「デフォルト」列のブランクは、変数が設定されていないことを意味します。
- その他の列のブランクは、変数が使用されていないことを意味します。

脚注は、表の終わりに示します。

注: デフォルト設定および例では、標準 _CEE_ENVFILE 構文を使用します。この構文は、管理アプリケーションで環境データを定義するときには使用しません。

表 11. 環境変数を使用する場所

環境変数 = <デフォルト>	デーモン・サーバー・インスタンス	システム管理サーバー・インスタンス		ネーミング・サーバー・インスタンス		インターフェース・リポジトリー・インスタンス		ビジネス・アプリケーション・サーバー・インスタンス		OS/390 クライアント
	制御領域	制御領域	サーバー領域	制御領域	サーバー領域	制御領域	サーバー領域	制御領域	サーバー領域	
BBOLANG=ENUS	O	O	O	O	O	O	O	O	O	O
CBCONFIG= /WebSphere390/CB390	R	R	R	R	R	R	R	R	R	
CLASSPATH=			O		O		O		O[1]	
CLIENT_DCE_QOP= NO_PROTECTION										O
CLIENT_HOSTNAME=										O
CLIENTLOGSTREAMNAME=										O
CLIENT_RESOLVE_IPNAME= <RESOLVE_IPNAME の値>			O		O		O		O	O
CLIENT_TIMEOUT=										
com.ibm.ws.naming.ldap.containerdn= ibm-wsnTree=t1,o=WASNaming,c=us				O	O					
com.ibm.ws.naming.ldap. domainname=*sysplex*				O	O					
com.ibm.ws.naming.ldap.masterurl= ldap://<localhost>:1389				O	O					
DAEMON_IPNAME=	R	O								
DAEMON_PORT=5555	O[2]	O[2]								
DATA.CTRLHOST=										O
DATA.CTRLPORT=5000										O
DEFAULT_CLIENT_XML_PATH=										O[3]
DM_GENERIC_SERVER_NAME= CBDAEMON	O[2]	O[2]								
DM_SPECIFIC_SERVER_NAME= DAEMON01	O[4]	O[4]		O[4]		O[4]		O[4]		
HOME=										O

表 11. 環境変数を使用する場所 (続き)

環境変数 = <デフォルト>	デーモン・サーバー・インスタンス	システム管理サーバー・インスタンス		ネーミング・サーバー・インスタンス		インターフェース・リポジトリー・インスタンス		ビジネス・アプリケーション・サーバー・インスタンス		OS/390 クライアント
	制御領域	制御領域	サーバー領域	制御領域	サーバー領域	制御領域	サーバー領域	制御領域	サーバー領域	
IBM_OMGSSL=0								O	O	
ICU_DATA= /usr/lpp/WebSphere/bin/		R	R							
IR_GENERIC_SERVER_NAME= CBINTFRP		O								
IR_SPECIFIC_SERVER_NAME= INTFRP01	O[4]	O[4]		O[4]		O[4]		O[4]		
IRPROC=BBOIR	O	O								
IVB_DEBUG_ENABLED=									O	O
IVB_DRIVER_PATH= /usr/lpp/WebSphere		R	R							
IVB_HOME=									O	O
java.naming.factory.initial= com.ibm.ws.naming.ldap. WsnLdapInitialContextFactory				O	O					
java.naming.security.credentials= secret				O	O					
java.naming.security.principal= cn=WASAdmin,o=WASNaming,c=us				O	O					
JAVA_COMPILER=									O	O
JAVA_IEEE754=										O[11]
JVM_DEBUG=			O		O		O		O	
JVM_HEAPSIZE=256									O	
JVM_LOGFILE=									O	O
LDAPBINDPW=		F			R[5]					
LDAPCONF=		F			R[5]					
LDAPHOSTNAME=		F			R[5]					
LDAPIRBINDPW=		F					R[6]			

表 11. 環境変数を使用する場所 (続き)

環境変数 = <デフォルト>	デーモン・サーバー・インスタンス	システム管理サーバー・インスタンス		ネーミング・サーバー・インスタンス		インターフェース・リポジトリー・インスタンス		ビジネス・アプリケーション・サーバー・インスタンス		OS/390 クライアント
	制御領域	制御領域	サーバー領域	制御領域	サーバー領域	制御領域	サーバー領域	制御領域	サーバー領域	
LDAPIRCONF=		F					R[6]			
LDAPIRHOSTNAME=		F					R[6]			
LDAPIRNAME=		F					R[6]			
LDAPIRROOT=		F					R			
LDAPNAME=		F			R[5]					
LDAPROOT=		F			R					
LIBPATH=			O		O		O		O[1]	
LOGSTREAMNAME=	O	O								
MIN_SRS=[MOFW の場合は 0、J2EE の場合は 1]								O		
NM_GENERIC_SERVER_NAME= CBNAMING		O								
NM_SPECIFIC_SERVER_NAME= NAMING01	O[4]	O[4]		O[4]		O[4]		O[4]		
NMPROC=BBONM	O	O								
OTS_DEFAULT_TIMEOUT=30	O	O	O	O	O	O	O	O	O	
OTS_MAXIMUM_TIMEOUT=60	O	O	O	O	O	O	O	O	O	
PATH=									O	O
RAS_MINORCODEDEFAULT= NODIAGNOSTICDATA										
REM_DCEPASSWORD=										O
REM_DCEPRINCIPAL=										O
REM_PASSWORD=		O[7]		O[7]		O[7]		O[7]		O

表 11. 環境変数を使用する場所 (続き)

環境変数 = <デフォルト>	デーモン・サーバー・インスタンス	システム管理サーバー・インスタンス		ネーミング・サーバー・インスタンス		インターフェース・リポジトリー・インスタンス		ビジネス・アプリケーション・サーバー・インスタンス		OS/390 クライアント
	制御領域	制御領域	サーバー領域	制御領域	サーバー領域	制御領域	サーバー領域	制御領域	サーバー領域	
REM_USERID=		O[7]		O[7]		O[7]		O[7]		O
RESOLVE_IPNAME=		O[8]	O[9]	O[9]	O[9]	O[9]	O[9]	O[9]	O[9]	R[10]
RESOLVE_PORT=900		O	O	O	O	O	O	O	O	O
SM_DEFAULT_ADMIN= CBADMIN		O								
SM_GENERIC_SERVER_NAME= CBSYSMGT		O								
SM_SPECIFIC_SERVER_NAME= SYSMGT01	O[4]	O[4]		O[4]		O[4]		O[4]		
SMPROC=BBOSMS	O	O								
SOMOOSQL=									O	
SRVIPADDR=	O	O		O		O		O		
SSL_KEYRING=										O
SYS_DB2_SUB_SYSTEM_NAME=DB2	R	R		R		R		R		
TRACEALL=1	O	O	O	O	O	O	O	O	O	O
TRACEBASIC=	O	O	O	O	O	O	O	O	O	O
TRACEBUFFCOUNT=4	O	O	O	O	O	O	O	O	O	
TRACEBUFFLOC=(サーバー: BUFFER クライアント: SYSPRINT)	O	O	O	O	O	O	O	O	O	O
TRACEBUFFSIZE=1M	O	O	O	O	O	O	O	O	O	
TRACEDETAIL=	O	O	O	O	O	O	O	O	O	O
TRACEMINORCODE=										
TRACEPARM=00	O									

表 11. 環境変数を使用する場所 (続き)

環境変数 = <デフォルト>	デーモン・サーバー・インスタンス	システム管理サーバー・インスタンス		ネーミング・サーバー・インスタンス		インターフェース・リポジトリー・インスタンス		ビジネス・アプリケーション・サーバー・インスタンス		OS/390 クライアント
	制御領域	制御領域	サーバー領域	制御領域	サーバー領域	制御領域	サーバー領域	制御領域	サーバー領域	

注:

1. IMS PAA および CICS PAA を含め、Java を使用するサーバー領域に必要です。
2. デーモン・サーバーに対する値を指定する場合は、システム管理サーバー制御領域に対するのと同じ値を指定する必要があります。
3. クライアントがシステム管理スクリプト API を使用する際に必要です。
4. この値は、シスプレックス内の 2 番目およびそれ以降のシステムに対して指定する必要があります。
5. LDAPCONF は、LDAPBINDPW、LDAPHOSTNAME、および LDAPNAME と一緒に使用することはできません。LDAPCONF か、あるいは LDAPBINDPW、LDAPHOSTNAME、および LDAPNAME が必要です。
6. LDAPIRCONF は、LDAPIRBINDPW、LDAPIRHOSTNAME、および LDAPIRNAME と一緒に使用することはできません。LDAPIRCONF か、あるいは LDAPIRBINDPW、LDAPIRHOSTNAME、および LDAPIRNAME が必要です。
7. サーバーが別のサーバーのリモート・クライアントになるときに使用されます。
8. デフォルトは、ブートストラップ中の DAEMON_IPNAME の値です。
9. デフォルトは、ローカル・システム IP 名です。通常は、コーディングしないでください。
10. デーモン・サーバーがクライアントとして同じシステム上にある場合は、オプションです。この場合、デフォルトはローカル・システム IP 名です。
11. OS/390 で実行される Java クライアントに必要です。

環境変数の説明

BBOLANG=*LANGUAGE*

使用される WebSphere for z/OS メッセージ・カタログの名前。デフォルトは、ENUS です。

CBCONFIG=*path*

会話が活動化されるときに WebSphere for z/OS による構成および環境ファイルの書き込み先となる、HFS ファイルの読み取りおよび書き込みディレクトリーを指定します。制御およびサーバー領域開始プロシージャーの &CBCONFIG 変数は、この値と一致していなければなりません。この方法で、WebSphere for z/OS は、これらの開始プロシージャーが実行されたときに、サーバーに対する適切な環境ファイルを検出することができます。デフォルトは、/WebSphere390/CB390 です。

例: `CBCONFIG=/WebSphere390/CB390`

CLASSPATH=*path1*:*[path2]*:...

サーバー領域内の Java ビジネス・オブジェクトが使用するための、Java クラス・ファイル (.jar ファイルおよび classes.zip ファイル) を指定します。ユーザーが Java ビジネス・オブジェクトを使用するときに、Java ビジネス・オブジェクトの .jar ファイルを指定します。CLASSPATH ステートメント全体は、1 行にしなければなりません。

例:

```
CLASSPATH=/usr/lpp/WebSphere/lib/xerces.jar:...
```

CLIENT_DCE_QOP= *value*

ローカル OS/390 クライアントが現行のトランザクション・フローに適用するために使用する、DCE メッセージ・オプションのレベル。通常は、リモート・システム上のサーバーにアクセスする OS/390 クライアントに対する DCE セキュリティーを設定します。サーバーに対する DCE レベルは、管理アプリケーションを介して設定されることに注意してください。

クライアントおよびサーバー上で DCE 認証が使用可能になると、DCE の第三者認証方式を使用したハンドシェーク・メッセージ交換が行われ、それによって、双方の正当性が証明されます。いったん上記の交換が行われると、3 つの保護レベルのうちの 1 つにメッセージが割り振られます。これらのメッセージは、以下に示すように、この環境変数の値です。

NO_PROTECTION

DCE は、メッセージとその応答が正当な送信側からのものであることだけを保証します。これはデフォルトです。

INTEGRITY
DCE は、メッセージが正当な送信側からのものであり、送信側による送信後、一切変更されていないことを保証します。

CONFIDENTIALITY
DCE は、正当な受信側だけが読めるように、メッセージを暗号化します。

CLIENT_HOSTNAME=
同じシステム上でデーモンが実行されていない場合に、OS/390 クライアントが自身のホスト IP 名を決定できるようにします。クライアント・プログラムが CBSeriesGlobal::hostName() メソッドを発行すると、システムは最初に CLIENT_HOSTNAME 環境変数を調べ、値が設定されている場合はそれを戻します。値が設定されていない場合は、システムはそのシステムで実行中のデーモン (ある場合) の IP 名を戻します。デフォルト値は、ヌルです。

例: `CLIENT_HOSTNAME=MYSYS.SYS.COM`

CLIENTLOGSTREAMNAME=*LOG_STREAM_NAME*
OS/390 クライアント ORB がエラー情報を書き込む WebSphere for z/OS エラー・ログ・ストリーム。

例: `CLIENTLOGSTREAMNAME=MY.CLIENT.ERROR.LOG`

CLIENT_RESOLVE_IPNAME=*IP_NAME*
OS/390 クライアント、またはクライアントとして機能しているサーバー領域が、ブートストラップ・サーバーにアクセスするため (つまり、クライアントまたはサーバー領域が解決初期参照メソッドを呼び出すとき) に使用する、インターネット・プロトコル名。デフォルトは、RESOLVE_IPNAME 環境変数によって指定された値です。この値は、システム管理サーバー (デフォルト・ブートストラップ・サーバー) に関連したインターネット・プロトコル名です。RESOLVE_IPNAME が設定されていない場合、値はクライアントまたはサーバー領域が実行されているシステムになります。

CLIENT_RESOLVE_IPNAME 環境変数を使用すると、リモート・システム上で稼動中のブートストラップ・サーバーを指定することができます。これに対して、その他のクライアントは、RESOLVE_IPNAME 環境変数で定義されたローカル・ブートストラップ・サーバーを使用します。

注: CLIENT_RESOLVE_IPNAME に対する TCP/IP ポート番号は、RESOLVE_PORT 環境変数で定義されます。

CLIENT_RESOLVE_IPNAME の値は、最大 255 文字までです。

例: CLIENT_RESOLVE_IPNAME=REMHOST

CLIENT_TIMEOUT=*n*

クライアント・メソッドの呼び出しからの応答に対するタイムアウト値を設定します。値は整数で、秒数は 10 倍 (したがって、値 10 は 1 秒) に表示されます。デフォルト値は 0 ですが、これは、タイムアウト値が設定されていないことを表します。

例: CLIENT_TIMEOUT=20

com.ibm.ws.naming.ldap.containerdn=*dn*

WsnName ツリーの開始点です。ネーミング・サーバーのみがこの環境変数を使用します。デフォルトは以下のとおりです。

```
ibm-wsnTree=t1,o=WASNaming,c=us
```

この値は、LDAP 初期設定ファイル (サンプルは bboldif.cb) に指定されている値と一致する必要があります。環境変数では大文字と小文字が区別されますが、LDAP ではどちらでもかまいません。″o=,c=″ の部分は、bboslapd.conf で接尾部として指定する必要もあります。例:

```
suffix   "o=WASNaming,c=US"
```

ヒント: 接尾部のステートメントは以下のようになります。

```
suffix         "<ws_rdn>"
```

(弊社が提供するサンプル bboslapd.conf の場合)

例:

```
com.ibm.ws.naming.ldap.containerdn=ibm-wsnTree=t1,o=WASNaming,c=us
```

com.ibm.ws.naming.ldap.domainname=*sysplex*

ホスト・ルートを一意的に識別し、JNDI グローバル・ネーム・スペースの区分化の基礎となります。ネーミング・サーバーのみがこの環境変数を使用します。デフォルトは、シスプレックス名です。

例:

```
com.ibm.ws.naming.ldap.domainname=plex1
```

com.ibm.ws.naming.ldap.masterurl=ldap://*IP_name*:*port*

LDAP サーバーの IP 名とポート番号。ネーミング・サーバーのみがこの環境変数を使用します。デフォルトは、ldap://<localhost>:1389 です。

例:

```
com.ibm.ws.naming.ldap.masterurl=ldap://wsldap:1389
```

DAEMON_IPNAME=*IP_NAME*

デーモン・サーバーがドメイン・ネーム・サービス (DNS) に登録するイン

ターネット・プロトコル名。WebSphere for z/OS と通信するどの CORBA クライアントにも、この IP 名が必要です。

インストール時には、デーモン・ブートストラップ・プロセスを開始する前に、必ず DAEMON_IPNAME 環境変数を定義する必要があります。定義をしないと、WebSphere for z/OS はエラー・メッセージを発行し、デーモンを終了してしまいます。

ブートストラップ・プロセスは、処理中にも、システム管理データベース内のデーモン IP 名を設定します。ブートストラップの後で、WebSphere for z/OS はシステム管理データベース内の値を使用します。ブートストラップの後で、DAEMON_IPNAME 環境変数の値を、システム管理データベース内にある値以外の値に変更することもできます。このような変更を行うと、エラー・メッセージが出されますが、デーモンはシステム管理データベースからのデーモン IP 名を使って初期化を行います。

デーモン・サーバー・インスタンスを同じホスト・クラスターに配置するには、サーバー・インスタンスごとに同じ DAEMON_IPNAME 値をコーディングする必要があります。

規則:

- DAEMON_IPNAME の値は、完全修飾された長い名前にする必要があります。
- 第 1 レベルの修飾子は、1 ～ 18 文字です。
- いったん選択したら、デーモンに対するポートおよび IP 名は変更しないでください。各オブジェクト参照にポートと IP 名が含まれているため、それらを変更すると、既存のオブジェクトにアクセスできなくなってしまうからです。

例: `DAEMON_IPNAME=CBQ091.PDL.POK.IBM.COM`

DAEMON_PORT=*n*

デーモン・サーバーが要求に対して listen するポート番号。デフォルトは、5555 です。値を指定する場合は、システム管理サーバー制御領域に対する値と同じにする必要があります。

例: `DAEMON_PORT=5555`

DATA.CTRLHOST=*IP_ADDRESS*

オブジェクト・レベル・トレース・クライアント・コントローラーが実行されているワークステーションの IP アドレスを指定します。IBM 分散デバッガーを使ってクライアントおよびサーバー・コンポーネントをデバッグしているときは、このアドレスを使用します。

例: DATA.CTRLHOST=MYHOST.IBM.COM

DATA.CTRLPORT=*n*

オブジェクト・レベル・トレース・クライアント・コントローラーが listen するポートを指定します。IBM 分散デバッガーを使ってクライアントおよびサーバー・コンポーネントをデバッグしているときは、このアドレスを使用します。デフォルトは、5000 です。

例: DATA.CTRLPORT=5000

DEFAULT_CLIENT_XML_PATH=*path*

システム管理スクリプト API が使用するデフォルト・パラメーター・リストを保持する XML ファイル・セットの位置を指定します。この環境変数は、システム管理スクリプト API を使用するクライアントに応じて設定する必要があります。

IBM は、デフォルトのパラメーター・リストが含まれたサンプルの XML ファイル・セットを提供しています。インストールが完了すると、これらのサンプルが /usr/lpp/WebSphere/samples/smapi に入っています。XML ファイルとパラメーター・リストの詳細は、*WebSphere Application Server V4.0 for z/OS and OS/390: システム管理スクリプト API*, SA88-8657 を参照してください。

システム管理スクリプト API のデフォルトの振る舞いは、次の 2 つの方法で変更できます。

1. システム管理スクリプト API を呼び出す REXX スクリプトにパラメーターを明示的に指定する。パラメーターを明示的に指定することにより、IBM が提供する XML サンプルを変更する必要がなくなります。クライアント環境ファイルで、以下をコーディングするだけで済みます。

   ```
   DEFAULT_CLIENT_XML_PATH=/usr/lpp/WebSphere/samples/smapi
   ```

2. XML ファイルを別のディレクトリーにコピーし (IBM 提供のサンプルは読み取り専用です)、パラメーター・リストを変更してから、新規のディレクトリーを指すように DEFAULT_CLIENT_XML_PATH を変更する。これらの変更は、システム管理スクリプト API のデフォルトの振る舞いをパーシスタント的に変更する場合にのみ必要になります。

例: DEFAULT_CLIENT_XML_PATH=/usr/lpp/WebSphere/samples/smapi

DM_GENERIC_SERVER_NAME=*SERVER_NAME*

デーモン・サーバーのサーバー名。デフォルトは、CBDAEMON です。値を指定する場合は、システム管理サーバー制御領域に対する値と同じにする必要があります。

例: DM_GENERIC_SERVER_NAME=CBDAEMON

DM_SPECIFIC_SERVER_NAME=*SERVER_INSTANCE_NAME*
デーモン・サーバーのサーバー・インスタンス名。デフォルトは、DAEMON01 です。この環境変数は、シスプレックス内の 2 番目およびそれ以降のすべてのシステムに対して指定する必要があります。

例: DM_SPECIFIC_SERVER_NAME=DAEMON01

IBM_OMGSSL=[0 | 1]
CORBA 準拠のセキュリティー・タグのみをサーバーによってエクスポートするかどうかを指定します。値 1 は、CORBA 準拠のタグのみをエクスポートすることを指定します。値 0 (デフォルト) は、CORBA 準拠のタグと CORBA に準拠しないタグの両方をエクスポートすることを指定します。

値 1 を指定するのは、サーバーが SSL (secure sockets layer) 基本認証のみをセキュリティーに使用し、クライアント (CICS またはその他の OEM ORB) が CORBA 準拠のタグを使用する場合です。これは、サーバーが SSL 基本認証を使用する場合に限ります。サーバーが SSL クライアント証明もサポートする場合は、この変数を設定する必要はありません。

値 0 を指定する (またはデフォルトを採用する) のは、サーバーが SSL 基本認証を使用し、分散プラットフォーム上または WebSphere Application Server エンタープライズ版 for OS/390 V3.02 の WebSphere クライアントと相互協調する場合です。

例: IBM_OMGSSL=1

HOME=*path*
ホーム・ディレクトリーを指定します。この変数は、ユーザーが UNIX シェルにログインすると、セキュリティー・プロダクト・ユーザー・プロファイルから自動的に設定されます。OS/390 上で実行されている C++ または Java クライアントの場合は、IBM 分散デバッガーを使ってビジネス・オブジェクトをデバッグするときに、この変数を /tmp に設定します。

例: HOME=/tmp

ICU_DATA=*path*
ブートストラップおよびインポート・サーバー・プロセッシング中にシステム管理サーバーが使用する、XML パーサーに必要なバイナリー・ファイルへのパス。デフォルト・ディレクトリーに WebSphere for z/OS コードをインストールした場合は、このパスを変更する必要はありません。デフォルトは、/usr/lpp/WebSphere/bin/ です

例: ICU_DATA=/usr/lpp/WebSphere/bin/

IR_GENERIC_SERVER_NAME=*SERVER_NAME*

インターフェース・リポジトリー・サーバーのサーバー名。デフォルトは、CBINTFRP です。インターフェース・リポジトリー・サーバーのサーバー領域を機能させるためには、この名前を使ってワークロード管理 (WLM) アプリケーション環境を定義する必要があります。

IR_SPECIFIC_SERVER_NAME=*SERVER_INSTANCE_NAME*

インターフェース・リポジトリー・サーバーのサーバー・インスタンス名。デフォルトは、INTFRP01 です。この環境変数は、シスプレックス内の 2 番目およびそれ以降のすべてのシステムに対して指定する必要があります。

IRPROC=*PROC_NAME*

インターフェース・リポジトリー・サーバーを開始するためにデーモン・サーバーが使用する開始プロシージャー。デフォルトは、BBOIR です。ユーザーは、独自の開始プロシージャーの名前を指定することができます。その場合は、デフォルトの開始プロシージャーから新規開始プロシージャーに情報をコピーします。

例: `IRPROC=BBOIR`

IVB_DEBUG_ENABLED=1

この OS/390 クライアントがオブジェクト・レベル・トレース・ランタイムをロードし、オブジェクト・レベル・トレースを使用することを指定します。IBM 分散デバッガーを使ってビジネス・オブジェクトをデバッグしている場合、アプリケーション・サーバーと、OS/390 上で実行されている C++ または Java クライアントに対しては、値 1 を指定する必要があります。

IVB_DRIVER_PATH=*path*

SMP/E のインストール後に WebSphere for z/OS ファイルが常駐するディレクトリーの名前。デフォルトは、`/usr/lpp/WebSphere` です。

例: `IVB_DRIVER_PATH=/usr/lpp/WebSphere`

IVB_HOME=*path*

IBM 分散デバッガーがアプリケーション・ソース・コードを検索できる位置を指定します。この環境変数は、オプションです。

JAVA_COMPILER=

ジャストインタイム (JIT) コンパイラーを使用することを指定します。

環境変数を使用する場合は、ヌル値 (`JAVA_COMPILER=`) を指定すると JIT コンパイラーがオンになります。その他の値を指定すると、JIT コンパイラーがオフになります。

デフォルトでは、OS/390 で稼動している Java 仮想マシン (JVM) は JIT コンパイラーを使用するので、この環境変数を明示的に設定する必要はありません。Java ビジネス・オブジェクトをデバッグしている場合は、ヌル以外の値を指定して JIT コンパイラーをオフにしてください。

例: `JAVA_COMPILER=`

JAVA_IEEE754=EMULATION

OS/390 上の Java クライアントが実行される Java 仮想マシン (JVM: Java virtual machine) 用にシステムをロードできるよう、正しい実行可能コードを指定します。この環境変数の設定は、OS/390 で実行される Java クライアントの場合のみ必要です。

java.naming.factory.initial=*context*

クライアントが使用する初期ネーミング・ファクトリー・コンテキスト。デフォルト値は、`com.ibm.ws.naming.ldap.WsnLdapInitialContextFactory` です。**例:**

```
java.naming.factory.initial=com.ibm.ws.naming.ldap.WsnLdapInitialContextFactory
```

java.naming.security.credentials=*password*

java.naming.security.principal によって指定された識別名が使用するパスワード。このパスワードは、初期システム・カスタマイズの過程で LDAP 初期設定ファイルによってアドミニストレーター・アクセス ID (デフォルトは WASAdmin) に定義されたパスワードと一致する必要があります。IBM は、WASAdmin アクセス ID を bboldif.cb というサンプル LDIF ファイルで提供しています。デフォルト値は `secret` です。

例:

```
java.naming.security.credentials=secret
```

推奨: IBM 提供のパスワードを変更する必要があります。

java.naming.security.principal=*distinguished_name*

WsnName ディレクトリーへの書き込みアクセスを保持するために定義される識別名 (ユーザー ID)。すべての JNDI ユーザーに読み取り / 書き込みアクセスを提供する場合にのみ指定します。この識別名は、初期システム・カスタマイズの過程で LDAP LDIF ファイルによってアドミニストレーター・アクセス ID (デフォルトは WASAdmin) に定義された名前と一致する必要があります。IBM は、WASAdmin アクセス ID を bboldif.cb というサンプル LDAP 初期設定ファイルで提供しています。デフォルト値は、`cn=WASAdmin,o=WASNaming,c=us` です。

例:

```
java.naming.security.principal=cn=WASAdmin,o=WASNaming,c=us
```

推奨: WASAdmin アクセス ID を保存しておくことをお勧めします。

JVM_DEBUG=1
OLT for Java オブジェクトをオンまたはオフにして、デバッグを設定します。IBM 分散デバッガーを使って Java オブジェクトをデバッグしている場合、OS/390 上で実行されているアプリケーション・サーバーおよび Java クライアントに対しては、値 1 が必要です。

この変数は、デバッグのために JVM メッセージを SYSOUT に転送する場合にも必要です。JVM_DEBUG=1 に設定して、JVM メッセージングを呼び出します。

JVM_HEAPSIZE=*n*
JVM ヒープの最大サイズ (メガバイト) を設定します。デフォルトは、256 MB です。

例: `JVM_HEAPSIZE=256 # specifies a 256 MB heap`

JVM_LOGFILE=*filename*
JVM からのメッセージを記録する HFS ファイルを指定します。

推奨: この変数は単一サーバー環境の場合のみ使用します。複数サーバー環境で `JVM_LOGFILE` を使用する場合は、すべてのサーバーが同じファイルに書き込みを行うため、ファイルを診断の目的で使用することは困難な場合があります。複数サーバー環境では、`JVM_DEBUG=1` を使用して JVM メッセージを特定のサーバーの SYSOUT に送信してください。

LDAPBINDPW=*password*
LDAP サーバーにバインドするためにネーミング・サーバーが使用するパスワード。LDAPNAME と結合して使用されます。

LDAPCONF=*filename*
WebSphere for z/OS が使用する LDAP 構成ファイル。HFS 内のファイルを指定する場合は、引用符を使用しないでください。MVS データ・セットを指定する場合は、そのデータ・セットを単一引用符で囲んでください。

例: `LDAPCONF='bbo.s21slapd.conf'`

LDAPHOSTNAME=*name:port*
インターフェース・リポジトリー・サーバーがそのデータ・ストアとして使用する LDAP サーバーのホスト名。

LDAPIRBINDPW=*password*
LDAP サーバーにバインドするためにインターフェース・リポジトリー・サーバーが使用するパスワード。LDAPIRNAME と結合して使用されます。

LDAPIRCONF=*filename*

インターフェース・リポジトリー・サーバーがそのデータ・ストアとして使用する LDAP サーバーによって使用される LDAP 構成ファイル。HFS 内のファイルを指定する場合は、引用符を使用しないでください。MVS データ・セットを指定する場合は、そのデータ・セットを単一引用符で囲んでください。

LDAPIRHOSTNAME=*name:port*

インターフェース・リポジトリー・サーバーがそのデータ・ストアとして使用する LDAP サーバーのホスト名。

LDAPIRNAME

インターフェース・リポジトリー・サーバーが使用する LDAP 入り口名。インターフェース・リポジトリー・サーバーは、この入り口名を使って、データ・ストアとして使用する LDAP サーバーに対してそれ自体を認証します。

LDAPIRROOT=*root*

インターフェース・リポジトリーがそのデータをアンカーする LDAP 入り口名。

例: `LDAPIRROOT=o=BOSS,c=U`

LDAPNAME

ネーミング・サーバーが使用する LDAP 入り口名。ネーミング・サーバーは、この入り口名を使って、データ・ストアとして使用する LDAP サーバーに対してそれ自体を認証します。

LDAPROOT=*root*

ネーミング・サーバーがそのデータをアンカーする LDAP 入り口名。

例: `LDAPROOT=o=BOSS,c=US`

LIBPATH=*path1*:*[path2]*:...

階層ファイルシステム (HFS) 内の Java に対する DLL 検索パスを指定します。システム、WebSphere for z/OS、および Java DLL を指定します。

例:

```
LIBPATH=/db2_install_path/lib:/usr/lpp/java/J1.3/bin:/usr/lpp/java/J1.3/bin/classic:/usr/lpp/WebSphere/lib
```

ここで、*db2_install_path* は DB2 for OS/390 をインストールした HFS です。

LOGSTREAMNAME=*LOG_STREAM_NAME*

ブートストラップ中にデーモンおよびシステム管理サーバーが使用する WebSphere for z/OS エラー・ログ・ストリーム名。システムは、ブートス

トラップ中に、デーモンおよびシステム管理サーバーに対する環境ファイル内に指定されていないと、次のアルゴリズムを使用してエラー・ログ・ストリーム名を作成します。

1. デーモン・サーバー内の IP 名の最初の修飾子を使用します。
2. 最初の修飾子が 9 文字以上である場合は、修飾子を 8 文字のストリングに分割し、ピリオドで区切ります。
3. 上位修飾子「BBO」を追加します。

たとえば、デーモン IP 名が MYDAEMONSERVER.IBM.COM である場合、アルゴリズムによって、エラー・ストリーム名 BBO.MYDAEMON.SERVER が作成されます。

ブートストラップの後で、管理アプリケーションを介して、シスプレックス全体、サーバー、またはサーバー・インスタンスに対するエラー・ログ・ストリーム名を作成または変更することができます。サーバー・エラー・ログ設定によって一般的な WebSphere for z/OS 設定が、サーバー・インスタンス設定によってサーバー設定が、それぞれ上書きされます。このようにして、サーバーやサーバー・インスタンスに対する一般的なエラー・ロギングを、特定のログ・ストリームに設定することができます。

処理中、指定されたログ・ストリームが見つからなかったり、アクセス不能な場合は、メッセージが表示され、サーバーのジョブ・ログにエラーが書き込まれます。

例: `LOGSTREAMNAME=MY.CB.ERROR.LOG`

ヒント: ログ・ストリーム名は引用符で囲まないでください。ログ・ストリーム名はデータ・セット名ではありません。

MIN_SRS=*nn*

初期化されたあとに実行を継続させるサーバー領域の数。つまり、ワークロード管理では、サーバー領域は非アクティブになってもシャットダウンされません。この環境変数を使用するのは、ワークロードの応答時間が、複数のサーバー領域が常に作業を処理できる状態にあることを必要とする場合です。

J2EE サーバーの場合のデフォルトは 1 です。MOFW サーバーの場合、デフォルトは 0 です。最大値は 20 です。20 を超える値を指定すると、変数は 20 に設定されます。

WebSphere for z/OS ガーベッジ・コレクションはサーバー領域をリフレッシュする場合がありますが、サーバー領域の最小数がこの環境変数に指定された値を下回ることはありません。

例: `MIN_SRS=2`

NM_GENERIC_SERVER_NAME=*SERVER_NAME*

ネーミング・サーバーのサーバー名。デフォルトは、CBNAMING です。ネーミング・サーバーのサーバー領域を機能させるためには、この名前を使ってワークロード管理 (WLM) アプリケーション環境を定義する必要があります。

例: `NM_GENERIC_SERVER_NAME=CBNAMING`

NM_SPECIFIC_SERVER_NAME=*SERVER_INSTANCE_NAME*

ネーミング・サーバーのサーバー・インスタンス名。デフォルトは、NAMING01 です。この環境変数は、シスプレックス内の 2 番目およびそれ以降のシステムのすべてのサーバー・インスタンスに対して指定する必要があります。

例: `NM_SPECIFIC_SERVER_NAME=NAMING01`

NMPROC=*PROC_NAME*

ネーミング・サーバーを開始するためにデーモン・サーバーが使用する開始プロシージャー。デフォルトは、BBONM です。ユーザーは、独自の開始プロシージャーの名前を指定することができます。その場合は、デフォルトの開始プロシージャーから新規開始プロシージャーに情報をコピーします。

例: `NMPROC=BBONM`

OTS_DEFAULT_TIMEOUT=*n*

デフォルトで、アプリケーション・トランザクションが完了するまでの所要時間 (秒数)。この時間は、 `current → set_timeout` メソッドを介してアプリケーション・トランザクションに独自のタイムアウト値が指定されていない場合に、アプリケーション・トランザクションに対して設定されます。

デフォルトは 30 秒で、最大値は 2147483 秒 (24.85 日) です。ヌルや 0 値は使用しないでください。

注: 会話がアクティブになると、システムはシステム管理サーバー・インスタンスについて**のみ**特殊処理を実行します。

- OTS_DEFAULT_TIMEOUT 変数が設定されていない場合は追加されます。

- OTS_DEFAULT_TIMEOUT の値が 3600 (秒) より小さい場合は、3600 に設定されます。

システム管理サーバー・インスタンスに対するこの特殊処理は、システム管理サーバー・インスタンスが実行時間の長いトランザクションを実行する場合があるために実行されます。その他のサーバー・インスタンスはこのような冗長なトランザクション・デフォルトを必要としません。

例: `OTS_DEFAULT_TIMEOUT=30`

OTS_MAXIMUM_TIMEOUT=*n*

アプリケーション・トランザクションが完了するまでの最大許容時間 (秒数)。アプリケーションによってより長い時間が割り当てられると、システムはそれを OTS_MAXIMUM_TIMEOUT 値に制限します。

デフォルトは 60 秒で、最大値は 2147483 秒 (24.85 日) です。ヌルや 0 値は使用しないでください。

注: 会話がアクティブになると、システムはシステム管理サーバー・インスタンスについて**のみ**特殊処理を実行します。

- OTS_MAXIMUM_TIMEOUT 変数が設定されていない場合は追加されます。
- OTS_MAXIMUM_TIMEOUT の値が 3600 (秒) より小さい場合は、3600 に設定されます。

システム管理サーバー・インスタンスに対するこの特殊処理は、システム管理サーバー・インスタンスが長期実行トランザクションを実行する場合があるために実行されます。その他のサーバー・インスタンスはこのような長時間のトランザクション・デフォルトを必要としません。

例: `OTS_MAXIMUM_TIMEOUT=60`

PATH=*path*

パスを指定します。OS/390 上で Java をトレースしたり、デバッグするときは、アプリケーション・サーバーの場合だけ、irmtdbgj という実行可能プログラムをパスに含めてください。

RAS_MINORCODEDEFAULT=*value*

システム例外マイナー・コードについてのドキュメンテーションを集めるためのデフォルトの動作を決定します。IBM サービスを受けている場合にだけ使用してください。

CEEDUMP
コールバックおよびオフセットを取り込みます。

ヒント: システムが CEEDUMP を取るには時間がかかるため、トランザクション・タイムアウトが起こることがあります。たとえば、OTS_DEFAULT_TIMEOUT が 30 秒に設定されていても、CEEDUMP を取るときに 30 秒より長くかかるため、アプリケーション・トランザクションがタイムアウトになることがあります。これを回避するためには、以下のいずれかを行います。

- トランザクション・タイムアウト値を増やす。
- RAS_MINORCODEDEFAULT=NODIAGNOSTICDATA をコーディングする。TRACEMINORCODE が環境ファイル内に**ない**ことを確認してください。

TRACEBACK
言語環境プログラムおよび OS/390 UNIX トレースバック・データを取り込みます。

SVCDUMP
MVS ダンプを取り込みます (クライアント内にダンプは作成しません)。

NODIAGNOSTICDATA
デフォルト。CEEDUMP、TRACEBACK、SVCDUMP の設定は収集されません。

注: 結果は、別の環境変数、TRACEMINORCODE の設定によって異なることがあります。TRACEMINORCODE=(ヌル値) および RAS_MINORCODEDEFAULT=TRACEBACK をコーディングすると、トレースバックが得られます。しかし、RAS_MINORCODEDEFAULT=NODIAGNOSTICDATA および TRACEMINORCODE=ALL をコーディングしても、トレースバックが得られます。したがって、RAS_MINORCODEDEFAULT=NODIAGNOSTICDATA を指定しても、TRACEBACK はキャンセルされず、TRACEBACK が収集されないだけです。

REM_DCEPASSWORD=*password*
OS/390 クライアントがシスプレックス外のシステムに要求を出し、SSL タイプ 1 認証が使用されているときに、セキュリティー・コンテキスト内で渡されるリモート DCE プリンシパルのパスワード。パスワードは、パスワードに対する DCE 要件を満たしていなければなりません。

例: REM_DCEPASSWORD=mydcePW

REM_DCEPRINCIPAL=*principal*
クライアントがシスプレックス外のシステムに要求を出し、SSL タイプ 1 認証が使用されているときに、セキュリティー・コンテキスト内で渡されるプリンシパル。このプリンシパルは、ターゲット・サーバー上で定義されていなければなりません。値は、プリンシパルに対する DCE 要件を満たしていなければなりません。

例: REM_DCEPRINCIPAL=myDCEprin

REM_PASSWORD=*password*
クライアントがリモート OS/390 システムに要求を出し、ユーザー ID / パスワード・セキュリティーまたは SSL セキュリティーが使用されているときに、セキュリティー・コンテキスト内で使用されるパスワード。

例: REM_PASSWORD=MYPASSW

REM_USERID=*USER_ID*
クライアントがリモート OS/390 システムに要求を出し、ユーザー ID / パスワード・セキュリティーまたは SSL セキュリティーが使用されているときに、セキュリティー・コンテキスト内で使用されるユーザー ID。

例: REM_USERID=MCOX

RESOLVE_IPNAME=*IP_NAME*
システム管理サーバーがドメイン・ネーム・サービス (DNS) に登録するインターネット・プロトコル。WebSphere for z/OS と通信するどの CORBA クライアントにも、この IP 名が必要です。未設定の場合、解決 IP 名は、プログラムが実行されているシステムです。

規則: RESOLVE_IPNAME の値は完全修飾名にする必要がありますが、255 文字を超えてはいけません。

例: RESOLVE_IPNAME=CBQ091.COMPANY.NY.COM

RESOLVE_PORT=*n*
システム管理サーバーが要求に対して listen するポート番号。デフォルトは、900 です。これは、オブジェクト・リクエスト・ブローカーに対する予約済ポートなので、弊社は、この値を変更しないことをお勧めします。すでにこのポートを使用しているアプリケーションがある場合は、TCP/IP バインド固有サポートと SRVIPADDR 環境変数を使用することを検討してください。

例: RESOLVE_PORT=900

SM_DEFAULT_ADMIN=*USER_ID*

管理および操作アプリケーションを使用する管理者のユーザー ID。この環境変数は、インストール中にシステム管理ブートストラップによって使用されます。システム管理ブートストラップの実行後にこの環境変数を設定しても、有効になりません。この環境変数を定義しない場合、デフォルトのユーザー ID は CBADMIN になります。このユーザー ID を OS/390 に定義して、適切なセキュリティー権限 (たとえば、RACF 許可および LDAP 許可) を与える必要があります。

注: ブートストラップの実行後は、管理アプリケーションを介してのみ追加の管理者ユーザー ID を定義することができます。これらのユーザー ID は、SM_DEFAULT_ADMIN によって定義されたユーザー ID には置き換わりません。

例: `SM_DEFAULT_ADMIN=DUDE`

SM_GENERIC_SERVER_NAME=*SERVER_NAME*

システム管理サーバーのサーバー名。デフォルトは、CBSYSMGT です。システム管理サーバーのサーバー領域を機能させるためには、この名前を使ってワークロード管理 (WLM) アプリケーション環境を定義する必要があります。

例: `SM_GENERIC_SERVER_NAME=CBSYSMGT`

SM_SPECIFIC_SERVER_NAME=*SERVER_INSTANCE_NAME*

システム管理サーバーのサーバー・インスタンス名。デフォルトは、SYSMGT01 です。この環境変数は、シスプレックス内の 2 番目およびそれ以降のシステムのすべてのサーバー・インスタンスに対して指定する必要があります。

例: `SM_SPECIFIC_SERVER_NAME=SYSMGT01`

SMPROC=*PROC_NAME*

システム管理サーバーを開始するためにデーモン・サーバーが使用する開始プロシージャー。デフォルトは、BBOSMS です。ユーザーは、独自の開始プロシージャーの名前を指定することができます。その場合は、デフォルトの開始プロシージャーから新規開始プロシージャーに情報をコピーします。

例: `SMPROC=BBOSMS`

SOMOOSQL=*値*

文字列属性についてオブジェクト指向 SQL 照会を使用するクライアン

ト・アプリケーションのパフォーマンスを向上させます。SOMOOSQL=1 を使用することによって、文字列比較がデータベースにプッシュダウンされます。

デフォルト値は、ヌル (SOMOOSQL=) です。

規則: SOMOOSQL=1 を使用できるのは、データベースとサーバー領域アドレス・スペースが同じロケール内で実行するように宣言されている場合だけです。

SRVIPADDR=*IP_ADDRESS*

クライアント接続要求を listen するために WebSphere for z/OS サーバーが使用する、ドット付き 10 進形式の IP アドレス。

この IP アドレスは、サーバーが TCP/IP にバインドするときに使用します。通常、サーバーは、ローカル TCP/IP スタックに構成されたすべての IP アドレスで listen します。しかし、作業を分離したり、同じポート上で複数のサーバーが listen できるようにしたいときは、SRVIPADDR を使用することができます。指定された IP アドレスは、WebSphere for z/OS がインバウンド要求を受信するときに使用する唯一の IP アドレスになります。通常、デーモン IP 名、解決 IP 名、またはユーザーが使用しているサーバーのホスト名も、この特定の SRVIPADDR にマップする必要があります。

SSL_KEYRING=*keyring*

SSL 処理において使用される OS/390 クライアントの鍵リングの名前。鍵リングは、RACF 内に置く必要があります。

例: `SSL_KEYRING=IVPRING`

SYS_DB2_SUB_SYSTEM_NAME=*NAME*

データベースに接続するために、デーモンおよびシステム管理サーバーが使用する DB2 for OS/390 名。DB2 for OS/390 サブシステム名かグループ接続名のいずれかを使用してください。デフォルトは、DB2 です。デフォルトがユーザーのインストールに適さない場合は、正しい値に合わせて環境変数を変更してください。

例: `SYS_DB2_SUB_SYSTEM_NAME=DB21`

TRACEALL=*n*

WebSphere for z/OS に対するトレース・レベルを指定します。有効な値とその意味は、以下のとおりです。

0 トレースなし。

1 例外トレース。デフォルトです。

2 基本および例外トレース。

3 基本および例外トレースを含めた、詳細トレース。

WebSphere for z/OS サブコンポーネントに対するトレース・レベルを設定するために、この変数を TRACEBASIC および TRACEDETAIL 環境変数と結合して使用します。この変数は、IBM サービス技術員の指示があるまで変更しないでください。

例: `TRACEALL=1`

TRACEBASIC=*n* | (*n*,...)
特定の WebSphere for z/OS サブコンポーネントに対するトレース・オーバーライドを指定します。番号で指定されたサブコンポーネントは、基本および例外トレースを受け取ります。複数のサブコンポーネントを指定する場合は、括弧を使って番号とコンマを区切ってください。サブコンポーネント番号とその意味については、IBM サービスにお問い合わせください。WebSphere for z/OS のその他のパーツは、TRACEALL 環境変数で指定されたとおりにトレースを受け取ります。TRACEBASIC は、IBM サービス技術員の指示があるまで変更しないでください。

例: `TRACEBASIC=3`

TRACEBUFFCOUNT=*n*
割り振るトレース・バッファーの番号を指定します。有効な値は、4 〜 8 です。デフォルトは 4 です。

TRACEBUFFLOC=SYSPRINT | BUFFER
トレース・レコードの移動先を、sysprint (SYSPRINT) またはメモリー・バッファー (BUFFER) のいずれかに指定し、その後、CTRACE データ・セットに指定します。デフォルトは、トレース・レコードをクライアントの sysprint に送信し、さらに他のすべての WebSphere for z/OS プロセスのためのバッファーに送信します。サーバーの場合は、片方または両方の値を、スペースで区切って指定することができます。クライアントの場合は、TRACEBUFFLOC=SYSPRINT しか指定できません。

例: `TRACEBUFFLOC=SYSPRINT BUFFER`

TRACEBUFFSIZE=*n*
単一トレース・バッファーのサイズ (バイト) を指定します。文字「K」(K バイト) または「M」(M バイト) を使用することができます。有効な値は、128K〜4M です。デフォルトは、1M です。

TRACEDETAIL=*n* | (*n*,...)
特定の WebSphere for z/OS サブコンポーネントに対するトレース・オー

バーライドを指定します。番号で指定されたサブコンポーネントは、詳細トレースを受け取ります。複数のサブコンポーネントを指定する場合は、括弧を使って番号とコンマを区切ってください。サブコンポーネント番号とその意味については、IBM サービスにお問い合わせください。
WebSphere for z/OS のその他のパーツは、TRACEALL 環境変数で指定されたとおりにトレースを受け取ります。TRACEDETAIL は、IBM サービス技術員の指示があるまで変更しないでください。

例:

```
TRACEDETAIL=3
TRACEDETAIL=(3,4)
```

TRACEMINORCODE=*value*

システム例外マイナー・コードのトレースバックを使用可能にします。IBM サービスから指示があった場合にだけ使用してください。値は、以下のとおりです。

ALL|all
: すべてのシステム例外マイナー・コードに対するトレースバックを使用可能にします。

minor_code
: 特定のマイナー・コードに対するトレースバックを使用可能にします。コードは、X'C9C21234' などの 16 進数で指定してください。

(ヌル値)
: デフォルト。トレースバックは収集されません。

注: 結果は、別の環境変数 RAS_MINORCODEDEFAULT の設定によって異なることがあります。TRACEMINORCODE=ALL と RAS_MINORCODEDEFAULT=NODIAGNOSTICDATA をコーディングすると、トレースバックが得られます。しかし、TRACEMINORCODE=(ヌル値) と RAS_MINORCODEFAULT=TRACEBACK をコーディングしても、トレースバックが得られます。したがって、TRACEMINORCODE=(ヌル値) を指定しても、TRACEBACK はキャンセルされず、TRACEBACK が収集されないだけです。

TRACEPARM=*SUFFIX* | *MEMBER_NAME*

CTRACE PARMLIB メンバーを識別します。値は、2 文字の接尾部 (ストリング CTIBBO に追加されて、PARMLIB メンバーの名前を作成する) であるか、PARMLIB メンバーの完全指定名のいずれかです。たとえば、接尾部「01」を使用することができますが、システムはこれを「CTIBBO01」に解決されます。完全指定名は、CTRACE PARMLIB メンバーに対する命

名要件を満たしていなければなりません。詳細については、*z/OS MVS 診断: ツールと保守援助プログラム*, GA88-8561 を参照してください。

デフォルト値は、00 です。

この環境変数が指定されていて、PARMLIB メンバーが見つからないときは、デフォルトの PARMLIB メンバーである CTIBBO00 が使用されます。指定された変数も、デフォルトの PARMLIB メンバーも見つからないときは、トレースが CTRACE に定義されますが、CTRACE 外部書き出しプログラムには接続されません。PARMLIB メンバーと CTRACE 外部書き出しプログラムの詳細は、*WebSphere Application Server V4.0 for z/OS and OS/390: メッセージおよび診断*, GA88-8655 を参照してください。

デーモン・サーバーは、この環境変数を認識する唯一のサーバーであることに注意してください。

例: `TRACEPARM=01`

付録B. WebSphere for z/OS クライアントとしての IMS アプリケーション

WebSphere for z/OS を使用すると、IMS トランザクション処理アプリケーションをクライアント・アプリケーションとして機能させることができるようになります。つまり、IMS メッセージ処理領域内で実行されている IMS アプリケーションは、WebSphere for z/OS MOFW サーバーで実行されている CORBA ビジネス・オブジェクトを作成、検索、使用、および削除することができます。180ページの図17は、IMS アプリケーションを WebSphere for z/OS クライアントにするための環境および処理を示しています。

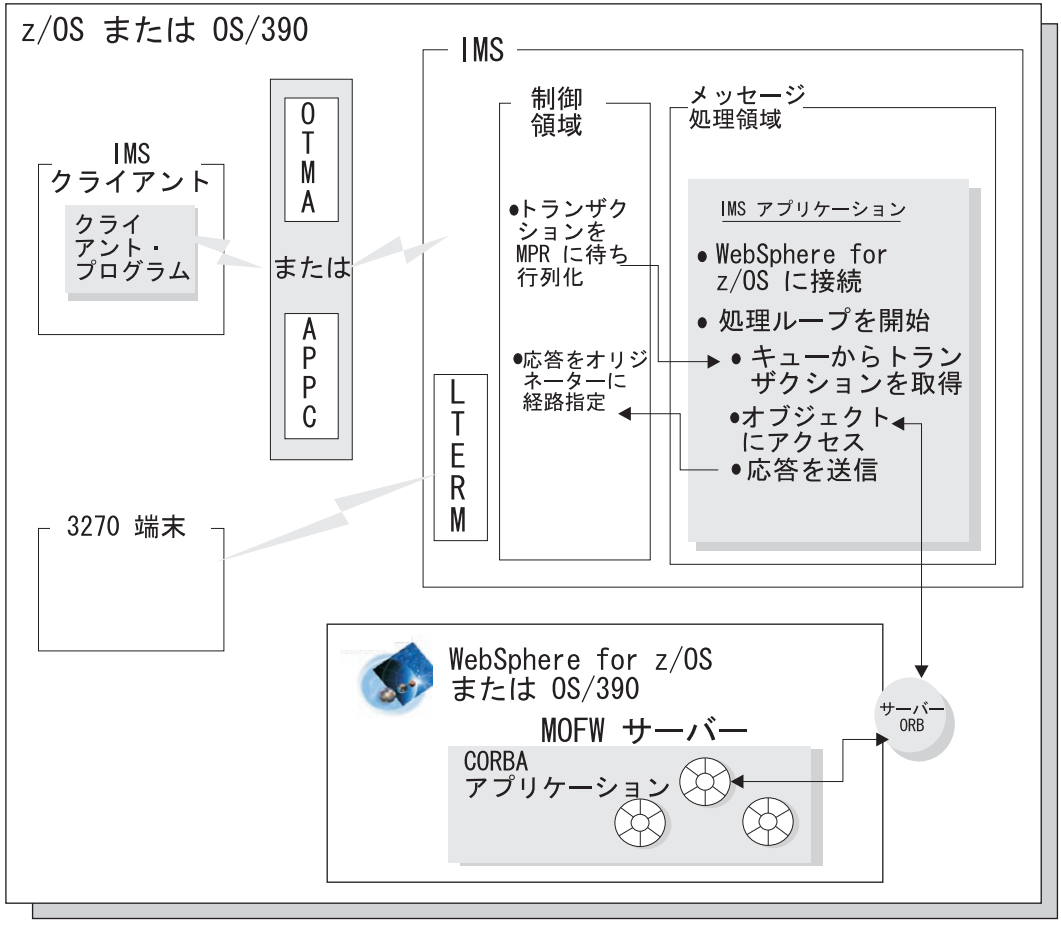

図 17. WebSphere for z/OS MOFW サーバーのクライアントとして実行されている IMS アプリケーション

この図では、次のような処理が行われます。

- IMS クライアントが IMS にトランザクションを実行依頼する。このトランザクションは、要求を受け取り、以下のうち 1 つを介して応答するように構成することができます。
  - オープン・トランザクション・マネージャー・アクセス (OTMA)、または
  - MVS の拡張プログラム間通信コンポーネント (APPC/MVS)

  別の方法として、IMS 定義論理端末 (LTERM) である 3270 端末からトランザクション要求を送信することもできます。
- IMS 制御領域は、WebSphere for z/OS クライアントとして実行されている IMS アプリケーションに対するトランザクション要求を待ち行列化する。

- IMS アプリケーションは、WebSphere Application Server Component Broker クライアント・プログラミング・モデルに従って WebSphere for z/OS MOFW サーバーに接続し、そのサーバー内のオブジェクトに対するメソッドを駆動して元の IMS クライアント・トランザクション要求を満たす。IMS アプリケーションは、その後 IMS クライアントに要求を送信します。

次の表は、IMS アプリケーションを WebSphere for z/OS クライアントとして作成し、実行するためのサブタスク、および関連手順を示しています。

サブタスク	関連した手順 (参照項目)
IMS アプリケーションを WebSphere for z/OS クライアントとして実行するための開発およびアセンブル	• 『IMS アプリケーションの設計に関するバックグラウンド』 • 186ページの『IMS アプリケーションを開発およびコンパイルするためのステップ』
IMS アプリケーションのためのランタイム環境の準備	• 184ページの『IMS アプリケーションに対するセキュリティーに関するバックグラウンド』 • 188ページの『IMS アプリケーションのランタイム環境を設定するためのステップ』

IMS アプリケーションの設計に関するバックグラウンド

WebSphere for z/OS クライアントとして作成された IMS トランザクション処理アプリケーションの場合は、以下のような、他の IMS アプリケーションにはないプログラミング特性を考慮する必要があります。

- Component Broker クライアント・プログラミング・モデルの使用と説明。
- IMS 処理のトランザクション有効範囲。

IMS アプリケーションを WebSphere for z/OS MOFW サーバーのクライアントにするには、Component Broker クライアント・プログラミング・モデルに従ってコーディングする必要があります。コーディングでは、次のような処理シーケンスを指定します。

1. `CBSeriesGlobal::Initialize` メソッドを使用して、WebSphere for z/OS との接続を確立する。
2. `CBSeriesGlobal::orb` メソッドを使用して、WebSphere for z/OS へのポインターを取得する。

3. resolve_initial_references メソッドを使用して、WebSphere for z/OS ネーミング・サーバーへの参照を取得する。
4. ファクトリー・ファインダーの位置を指定して、クライアントが使用する WebSphere for z/OS オブジェクトのホームを見つける。
5. IMS クライアントからのトランザクション処理の一部として、WebSphere for z/OS MOFW サーバー内でオブジェクトを検索または作成したり、使用、削除する。

推奨: WebSphere for z/OS のクライアントとして最高のパフォーマンスを得るために、アプリケーションを最初にロードするときは、WebSphere for z/OS を使って一度のみ初期化して IMS アプリケーションを設計してください。そうすると、アプリケーションは、再ロードや再初期化を行わなくても、アプリケーションに対して待ち行列化されたトランザクションを引き続き処理することができます。つまり、次のような処理を行う入力待ち (WFI) アプリケーションとしてアプリケーションを設計するということです。

1. CBSeriesGlobal::Initialize メソッドを発行する。
2. CBSeriesGlobal::orb メソッドを発行する。
3. resolve_initial_references メソッドを発行する。
4. トランザクション処理に必要なオブジェクト・ファクトリーとホームの位置を指定する。
5. IMS アプリケーションが次の処理を行うループを開始する。
   a. 次のトランザクションを得るための Get Unique 要求を発行する。
   b. WebSphere for z/OS MOFW サーバー内のオブジェクトに対するメソッドを発行して、トランザクションを処理する。
   c. IMS トランザクションのオリジネーターに応答する。

この設計により、WebSphere for z/OS のクライアントとしての IMS アプリケーションに対する最高のパフォーマンスが得られますが、アプリケーションに対して選択したセキュリティーのレベルによってその処理が上書きされることがあるため、各トランザクション要求を処理するために再ロードが必要になります。184ページの『IMS アプリケーションに対するセキュリティーに関するバックグラウンド』に、セキュリティー・オプションとそれがアプリケーション処理に及ぼす影響についての説明があります。

トランザクション有効範囲は、Component Broker クライアント・プログラミング・モデルと同様に、その他の IMS アプリケーションにはない、ユーザーの IMS アプリケーションの特徴です。たとえば、この IMS アプリケーションがトランザクションを要求すると、システムは特定のトランザクション・コンテキストを設定し、その単一のトランザクション要求を処理しながら、そのコンテキストの下でユーザーのアプリケーションが実行されるようにします。しか

し、ユーザーの IMS アプリケーションが WebSphere for z/OS MOFW サーバー内の CORBA オブジェクトに対してメソッドを発行すると、そのサーバーは IMS アプリケーションが実行されているトランザクション・コンテキストを使用しない場合があります。WebSphere for z/OS MOFW サーバーは、オブジェクトが置かれているコンテナーに対して有効なトランザクション・ポリシーによって、どのトランザクション・コンテキストを使用するかを決定します。一部のコンテナー・トランザクション・ポリシーに対しては、WebSphere for z/OS MOFW サーバーは別々のトランザクション・コンテキストを初期化して、オブジェクトに対するメソッド要求を処理し、呼び出し元 (すなわち、IMS アプリケーション) に戻る前にすべての変更をコミットします。このような動作のため、以下のリカバリー状態を考慮する必要があります。

- IMS トランザクションの処理中に IMS の下でエラーが発生した場合、およびオブジェクト更新要求が WebSphere for z/OS に送信される前に終了エラーが発生した場合、IMS トランザクションに対するリカバリーは、WebSphere for z/OS クライアント以外の IMS アプリケーションの場合と同じになります。
- IMS トランザクションに対するオブジェクト要求と、IMS トランザクション・コンテキストを伝搬したオブジェクト・コンテナー・ポリシーの処理中に WebSphere for z/OS の下で終了エラーが発生すると、WebSphere for z/OS は、オブジェクト要求に対して完了した処理をいずれもロールバックします。この場合、IMS アプリケーションは、WebSphere for z/OS からエラー通知をキャッチし、処理を続行するか、IMS トランザクションを終了するかの選択を行います (そのように設計されている場合)。
- WebSphere for z/OS が IMS アプリケーションの代わりにオブジェクト更新を正常に処理し、コミットした後 に、IMS トランザクションの処理中に IMS の下で終了エラーが発生した場合、ユーザーのアプリケーションは WebSphere for z/OS が完了した更新をロールバックすることはできません。しかし、ユーザーの IMS アプリケーションは、その独自の処理をロールバックすることがあります。この場合、終了エラーばかりでなく、一部の更新が完了し、それらがロールバックできないという事実も正しく伝えるように、アプリケーションを設計する必要があります。

必要に応じて、以下の 1 つまたは複数の資料で、詳しい情報を参照してください。

このトピックについての詳細	参照する資料
Component Broker クライアント・プログラミング・モデルおよびクライアントが使用するメソッドについての詳細	• *Component Broker プログラミングの手引き* • *Component Broker プログラミング解説書*
TRANSACT マクロを使用した、入力待ち (WFI) アプリケーションとしての IMS アプリケーションの定義についての詳細	*IMS/ESA 導入 第 2 巻*, GD88-7125 の『システム定義および調整』
WebSphere for z/OS MOFW サーバーのコンテナー・トランザクション・ポリシーおよびアプリケーション・プロセスのためのその含意についての詳細	19ページの『WebSphere for z/OS トランザクション環境に関するバックグラウンド』
WebSphere for z/OS のクライアントとしての IMS アプリケーションのコーディングに対する特定の規則および制限	186ページの『IMS アプリケーションを開発およびコンパイルするためのステップ』

IMS アプリケーションに対するセキュリティーに関するバックグラウンド

IMS アプリケーションをインストールする前に、アプリケーションそのものとインストールのセキュリティー要件についての情報に基づいて、そのアプリケーションに対してどのレベルのセキュリティーが必要であるかを決定する必要があります。セキュリティー管理は、メッセージ処理領域 (MPR) のユーザー ID か、あるいは IMS トランザクションのオリジネーターのユーザー ID のいずれかのレベルに設定することができます。 セキュリティーを MPR ユーザー ID のレベルに設定すると、ユーザーの IMS アプリケーションが処理するすべてのトランザクションは、アプリケーションが実行されている MPR のユーザー ID で実行されます。このセキュリティー設定は、IMS 定義論理端末 (LTERM) から受け取ったトランザクションに対するデフォルト・セキュリティー設定です。この場合、システムは、トランザクションまたはオリジネーターが、トランザクションを発行するための許可を得ているかどうかを検査しますが、その後、MPR ユーザー ID を使用して以下のことを行うために許可を検査します。

- データベースの更新
- UNIX システム・サービスの使用
- WebSphere for z/OS MOFW サーバーおよびそれらのサーバー内の CORBA オブジェクトの使用

セキュリティーを MPR ユーザー ID のレベルにすると、IMS アプリケーションが最初にロードされたときにすべての許可検査が行われます。トランザクシ

ョン処理が発生するユーザー ID は変更されないため、アプリケーションは、その後、追加の許可検査を行わずにトランザクションを処理します。

推奨: できれば、WebSphere for z/OS クライアントとしての IMS アプリケーションに対する可能な限り最高のパフォーマンスを得るために、181ページの『IMS アプリケーションの設計に関するバックグラウンド』で推奨されている設計と共に、このレベルのセキュリティーを使用してください。このセキュリティー設定を入力待ち (WFI) 設計と共に使用すると、アプリケーションが継続的にループして、初期プロセスを繰り返さずに追加のトランザクションを処理できるようになります。

ヒント: OTMA または APPC から受け取ったトランザクションの場合、セキュリティーのレベルを MPR ユーザー ID に設定するための 1 つの方法は、それぞれ IMS `/SECURE OTMA PROFILE` または `/SECURE APPC` コマンドを使用することです。

セキュリティーをトランザクション・オリジネーターのユーザー ID のレベルに設定すると、ユーザーの IMS アプリケーションが処理するすべてのトランザクションは、トランザクション・オリジネーターのユーザー ID で実行されます。このセキュリティー設定は、OTMA または APPC を介して IMS クライアントから受け取ったトランザクションに対するデフォルト・セキュリティー設定です。この場合、システムは、トランザクションを発行するためにオリジネーターが許可されているかどうかを見るときだけでなく、すべての許可検査に際しても、トランザクション・オリジネーターのユーザー ID を使用します。システムはまた、IMS アプリケーションがトランザクションを処理する前にアプリケーションを再ロードします。このレベルのセキュリティーは、「完全セキュリティー」として知られます。

完全セキュリティーでは、処理するトランザクションごとに IMS アプリケーションを再ロードしなければなりません。UNIX システム・サービスを使用する許可用に、IMS トランザクションを発行したユーザー ID に基づいて適切な安全保護環境を設定するためには、このような再ロードが必要です。この再初期設定は、IMS が 181ページの『IMS アプリケーションの設計に関するバックグラウンド』で推奨されるように設計されている場合であっても、行われます。

追加情報については、以下の 1 つまたは複数の資料を参照してください。

このトピックについての詳細	参照する資料
特に、LTERM、OTMA、または APPC から受け取ったトランザクションに対するデフォルト・セキュリティーを変更するための IMS セキュリティー設定の詳細	• *IMS/ESA 管理の手引き: システム*, SD88-7122 • *IMS/ESA Open Transaction Manager Access Guide*, SC26-8743
WebSphere for z/OS のクライアントとしての IMS アプリケーションに対するセキュリティーを設定するための特定の説明	188ページの『IMS アプリケーションのランタイム環境を設定するためのステップ』

IMS アプリケーションを開発およびコンパイルするためのステップ

WebSphere for z/OS クライアントである IMS アプリケーションを設計し、コーディングするには、IMS メッセージ処理領域で実行されるアプリケーションの設計とコーディングに必要な通常のステップの他に、いくつかのステップを完了する必要があります。

始める前に: 181ページの『IMS アプリケーションの設計に関するバックグラウンド』に記載されている、最適なパフォーマンスのための設計の推奨についての情報を必ず理解しておいてください。

以下のステップを実行して、IMS アプリケーションを WebSphere for z/OS クライアントとして作成およびコンパイルします。

1. IMS アプリケーション用の新規ソース・コードを作成するか、既存のソース・コードを編集します。

   規則:

   - このアプリケーションには、C++ プログラム言語を使用する。
   - Component Broker プログラミング・モデルを使用して、WebSphere for z/OS MOFW サーバーのクライアントとしてアプリケーションを設計し、コーディングする。簡単にいえば、IMS アプリケーションは、Component Broker プログラミング・クライアント・モデルに従って、以下を行う必要があります。
     - `CBSeriesGlobal::Initialize` メソッドを使用して、WebSphere for z/OS との接続を確立する。
     - `CBSeriesGlobal::orb` メソッドを使用して、WebSphere for z/OS へのポインターを取得する。

- resolve_initial_references メソッドを使用して、WebSphere for z/OS ネーミング・サーバーへの参照を取得する。
- ファクトリー・ファインダーの位置を指定して、クライアントが使用する WebSphere for z/OS CORBA オブジェクトのホームを見つける。
- IMS クライアントからのトランザクション処理の一部として、WebSphere for z/OS MOFW サーバー内でオブジェクトを検索または作成したり、使用、削除する。

Component Broker クライアント・プログラミング・モデルおよびクライアントが使用するメソッドの詳細については、*WebSphere Application Server エンタープライズ版 Component Broker: プログラミングの手引き* および *WebSphere Application Server エンタープライズ版 Component Broker: プログラミング解説書* を参照してください。

- 以下の z/OS または OS/390 言語環境プログラム・インターフェースを使用して、IMS に要求を渡す。使用するインターフェースによっては、以下のような適切なヘッダー・ファイルを含めます。

該当するインターフェースの場合	含めるヘッダー・ファイル
CTDLI	ims.h
CEETDLI	leawi.h

これらのインターフェースの追加情報については、以下の 1 つまたは複数の資料を参照してください。

- 使用法の詳細については、*z/OS 言語環境プログラム プログラミング・ガイド*, SA88-8549 と *IMS/ESA アプリケーション・プログラミング: トランザクション管理プログラム*, SD88-7121。
- インターフェースの構文については、*z/OS 言語環境プログラム プログラミング・リファレンス*, SA88-8550。

- ソース・コード内の #pragma runopts(...) ステートメントを使用して、IMS アプリケーションのためのランタイム・オプションを指定する。runopts ステートメントには、以下のオプションを含める必要があります。
  - POSIX(ON)。UNIX システム・サービス環境が必須であることを示します。
  - ENVAR("_CEE_ENVFILE=DD:BBOENV")。WebSphere for z/OS 固有の環境変数設定を含むファイルにリンクします。このファイルの作成につい

ては、『IMS アプリケーションのランタイム環境を設定するためのステップ』を参照してください。

必要に応じて、ランタイム・オプションの一般情報について記載されている *z/OS 言語環境プログラム プログラミング・ガイド*, SA88-8549 を参照してください。

制限: WebSphere for z/OS MOFW サーバー内の CORBA オブジェクトに対して駆動されるメソッドを含むトランザクションを定義するときは、`begin` 要求や `commit` 要求を発行しないでください。WebSphere for z/OS は、IMS 環境で実行されているクライアント・アプリケーションに対してこの機能をサポートしていません。

---

2. コンパイル・パラメーターである `TARGET(IMS)` と `PLIST(OS)` を使用して、IMS アプリケーション・ソース・コードをコンパイルします。

   必要に応じて、IMS でのアプリケーションの実行の詳細について記載された *z/OS 言語環境プログラム プログラミング・ガイド*, SA88-8549 を参照してください。

上記のステップを完了して IMS アプリケーションをコーディングし、コンパイルしたら、WebSphere for z/OS として実行される IMS トランザクションに必要な、WebSphere for z/OS 固有のインストール設定作業を実行することができます。これらの作業は、『IMS アプリケーションのランタイム環境を設定するためのステップ』にリストされています。

IMS アプリケーションのランタイム環境を設定するためのステップ

WebSphere for z/OS クライアントである IMS アプリケーションの環境を設定するには、IMS メッセージ処理領域でのアプリケーションのインストールと実行に必要な通常のステップの他に、いくつかのステップを完了することが必要です。

始めに: 184ページの『IMS アプリケーションに対するセキュリティーに関するバックグラウンド』に記載されている情報を必ず理解しておいてください。

以下のステップを実行して、WebSphere for z/OS クライアントとしての IMS アプリケーションの環境を設定します。

1. IMS アプリケーションに対する WebSphere for z/OS 固有の環境変数設定を含めるための HFS ファイルを作成します。149ページの『付録A. 環境フ

ァイル』に記載されている表と環境変数の説明を使用して、このクライアントに対してどのランタイム環境変数を設定する必要があるかを決定します。

注: 設定しているのは、WebSphere for z/OS サーバー環境ではなく、**クライアント**・ランタイム環境に対する変数です。

---

2. 新規 JCL プロシージャーを作成するか、既存の JCL プロシージャーを編集して、次のように、IMS アプリケーションが実行される IMS メッセージ処理領域を開始します。
   - IMSMSG という名前の IMS サンプル・ジョブを使用して、メッセージ処理領域を開始する。このサンプル・ジョブは、IMS DFSMPR プロシージャーを呼び出して新規領域を開始します。それぞれの MPR ごとに、固有の名前を指定するようにしてください。

     IMSMSG ジョブは、インストールの実施状況によって、IMS.JOBS または IMS.PROCLIB データ・セットのいずれかに入っています。
   - JCL プロシージャーに BBOENV DD ステートメントを追加して、クライアント・ランタイム環境用の WebSphere for z/OS 固有の環境変数設定を定義するために作成した HFS ファイルを識別する。この DD ステートメントには、151ページの『ランタイム・サーバー開始プロシージャーによる環境ファイルの指示方法』に記載されているものと同じ構文を使用します。

   IMSMSG という名前の IMS サンプル・ジョブまたは DFSMPR プロシージャーの詳細については、*IMS/ESA 導入 第 2 巻*, GD88-7125 の『システム定義および調整』を参照してください。

---

3. 184ページの『IMS アプリケーションに対するセキュリティーに関するバックグラウンド』に記載されている情報とインストールのセキュリティー要件についての知識に基づいて、IMS アプリケーションに対してどのレベルのセキュリティーが必要であるかを決定します。セキュリティー管理は、メッセージ処理領域 (MPR) のユーザー ID か、あるいは IMS トランザクションのオリジネーターのユーザー ID のいずれかのレベルに設定することができます。

   続いて、以下のような適切なセキュリティー作業を完了します。
   - 選択したセキュリティー・レベルによって、MPR のユーザー ID か、IMS トランザクションのオリジネーターのユーザー ID のいずれかに対する OMVS セグメントを定義する。UNIX システム・サービスは IMS

サービス要求を処理するための情報を抽出しなければならないため、ユーザー ID には、この許可が必要です。

必要に応じて、RACF への UNIX ユーザーの定義、またはユーザー OMVS セグメントの検査の詳細について記載された *z/OS UNIX システム・サービス 計画*, GA88-8639 を参照してください。

- RACF 内の CBIND クラスを使用して、WebSphere for z/OS MOFW サーバーへのアクセス許可を設定し、これらのサーバーに要求を渡す (すなわち、これらサーバー内のマネージド・オブジェクトを使用する)。WebSphere for z/OS は、CBIND クラス内の以下の 2 つのプロファイルを使用します。
  - CB.BIND.**server_name**。WebSphere for z/OS サーバーへのクライアント・アクセスを制御します。
  - CB.**server_name**。これらの WebSphere for z/OS サーバー内にあるマネージド・オブジェクトの使用を制御します。

  選択したセキュリティー・レベルによって、次のように許可を設定します。

該当するセキュリティーのレベル	使用する CBIND プロファイル

MPR のユーザー ID	MPR のユーザー ID に対して、IMS アプリケーションがアクセスする必要のある各サーバーと、これらの各サーバー内にあるマネージド・オブジェクトへの読み取りアクセスを許可します。例:
	`PERMIT CB.server1 CLASS(CBIND) ID(mpr_user_id) ACCESS(READ)`
	`PERMIT CB.BIND.server1 CLASS(CBIND) ID(mpr_user_id) ACCESS(READ)`

IMS トランザクション・オリジネーターのユーザー ID	MPR のユーザー ID とトランザクション・オリジネーターのユーザー ID は、次のように許可します。 – MPR のユーザー ID には、IMS アプリケーションがアクセスしなければならない各サーバーへの制御アクセスが必要です (CB.BIND.**server_name**)。 – すべてのオリジネーターのユーザー ID には、各サーバー内のマネージド・オブジェクトへの読み取りアクセス許可が必要です (CB.**server_name**)。

必要に応じて、*WebSphere Application Server V4.0 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652 に記載されているシステム・セキュリティーの設定についての情報を参照してください。

- 次のように、この IMS アプリケーションに対する TRANSACT マクロ・ステートメントに、適切なパラメーターを指定します。

該当するセキュリティーのレベル	指定するパラメーター
MPR のユーザー ID	– WFI パラメーター WFI パラメーターは、アプリケーションが別のトランザクションを要求するときに、IMS メッセージ・キューが空である場合に IMS アプリケーションが入力を待機することを指定します。 **ヒント:** 別の方法として、PWFI=Y を MPR のための DFSMPR プロシージャーに関する入力パラメーターとして指定することができます。 – PROCLIM=65535 この PROCLIM 設定によって、IMS アプリケーションを再ロードする前に、IMS アプリケーションが最大数のトランザクションを処理できるようになります。 このパラメーターの組み合わせによって、IMS アプリケーションは、再初期化しなくてもトランザクションを継続的に処理することができます。これによって、WebSphere for z/OS クライアントとして実行されている IMS アプリケーションに対して、可能な限り最高のパフォーマンスが得られます。
IMS トランザクション・オリジネーターのユーザー ID	PROCLIM=0 この PROCLIM 設定は、IMS アプリケーションが各トランザクションを処理した後で、そのアプリケーションを再ロードしなければならないことを示します。この設定は、ユーザー ID がトランザクション間で変更されるため、UNIX システム・サービス許可環境を正しく再設定するために必要です。

TRANSACT マクロの詳細については、必要に応じて、*IMS/ESA 導入 第 2 巻*, GD88-7125 の『システム定義および調整』に記載されている IMS トランザクションの定義についての情報を参照してください。

上記のステップを完了して IMS アプリケーションを WebSphere for z/OS のクライアントとして設定したら、通常、IMS トランザクションに必要なインストール作業をさらに進めることができます。詳細については、必要に応じて、*IMS/ESA 導入 第 2 巻*, GD88-7125 を参照してください。

付録C. インターフェース定義言語 (IDL) コンパイラー

IDL コンパイラーを使用して、CORBA 2.0 準拠 IDL ステートメントを含む 1 つまたは複数のファイルで説明されている、インターフェースの使用法とインプリメンテーション・バインディングを作成します。コンパイラーを使用するには、idlc コマンドに、有効なオプション値とコンパイルする IDL ファイルの名前を指定して入力します。

idlc コマンド構文

idlc *[-options] <file_name>*

[options]

IDL コンパイラーがどのようにユーザーの要求を処理するかを決定する、1 つまたは複数のオプション値。最初のオプション値の前にダッシュ (-) を付けることに注意してください。複数のオプションを指定したり、引き数の付いたオプションを指定するときは、追加の構文規則が適用されます。オプション値、関連した引き数、および残りの構文規則の説明については、194ページの『idlc コマンド・オプション構文および値』を参照してください。

<file_name>

コンパイルする 1 つまたは複数の IDL ファイル。IDL コンパイラーは、拡張子「.idl」の付いたファイルを処理します。ワイルドカード文字 (*) は、ファイル名のパス以外の部分で一度だけ使用することができます。ファイル名の前にパスが指定されていないと、IDL コンパイラーは現行ディレクトリー内でだけ、指定されたファイルを探します。

以下は、ディレクトリー “E:¥idl¥src” (次の例では、現行ディレクトリー) 内の IDL ファイルを指定する、受け入れ可能なメソッドの例です。

```
E:¥idl¥src¥xyz.idl
E:¥idl¥src¥xyz
E:¥idl¥src¥*.idl
E:¥idl¥src¥x*.idl
xyz.idl
xyz
x*
```

同じオプションか、あるいは類似するオプションのセットを使用する必要がある場合は、コンパイラーを実行するたびに、長いコマンドを再入力する代わりに IDLC_OPTIONS 環境変数を使用することができます。IDLC_OPTIONS 環

境変数には、任意の idlc コマンド・オプションを追加することができます。これによって、次に idlc コマンドを入力するときに、IDL コンパイラーは idlc コマンド自体に指定したオプションの前にこれらのオプションを処理します。たとえば、IDLC_OPTIONS 環境変数に以下のオプションを追加したとします。

```
-m cpponly -mdllname=mydll
```

その後、コマンド `idlc -ehh idlfile` を入力すると、要求の結果は以下のコマンドを入力した場合と同じになります。

```
idlc -m cpponly -mdllname=mydll -ehh idlfile
```

同じ技法を使用して、エミッターを指定することもできます。idlc コマンドそのものにエミット・リストを指定する代わりに、IDLC_EMIT 環境変数にエミッター名のリストを追加することができます。IDLC_OPTIONS 環境変数内のオプションの場合と同様に、IDL コンパイラーは、idlc コマンドに直接指定したオプションと共に、IDLC_EMIT 環境変数内のリストを処理します。

idlc コマンド・オプション構文および値

複数のオプション値を指定したり、引き数の付いたオプションを指定するときは、次の構文規則が適用されます。

- それぞれの値をブランクで区切ったり、オプション値を一緒に実行することができます。ブランクを使用する場合は、各オプション値をダッシュで始めるようにしてください。

  ```
  idlc [-p -V -v]
  ```

  オプション値を一緒に実行する場合は、最初のダッシュ以外はすべて省略してください。

  ```
  idlc [-pVv]
  ```

- 一部のオプションには、引き数を指定することができます。引き数を指定する場合は、必ずその引き数が適用されるオプションに指定してください。たとえば、以下の行のいずれかを指定することができます。

  ```
  idlc [-p -m tie]
  idlc [-p -mtie]
  idlc [-pm tie]
  idlc [-pmtie]
  ```

  関連した引き数の付いた複数のオプションを指定したい場合は、例の最後の 2 行のように、オプションを一緒に実行することはできません。(オプション値を一緒に実行するときは、最後のオプションの後にだけ引き数を指定する

ことができます。) 複数のオプションと引き数の場合、オプションと引き数の組み合わせを区切る必要があります。

- すべてのオプション値は、ファイル名に大文字と小文字の区別がないプラットフォーム上であっても、大文字と小文字の区別があります。それぞれのオプション値は、表12 で示すとおりに正確に入力するようにしてください。

表12 には、すべてのオプション値と引き数についてのリストと説明があります。

表 12. idlc コマンド・オプション

コマンド・オプション	説明
-?	idlc コマンド構文の要旨を標準出力に書き込む。
-D<*define-expression*>	IDL コンパイラーに対するプリプロセッサー変数を事前定義する。
-d<*directory-name*>	コンパイラーがエミット出力ファイルを配置するディレクトリーを指定する。デフォルトは、現行ディレクトリー。
-e<*emit-list*>	実行するエミッターのリストを指定する。エミット・リストは、短いエミッター名から構成される。リスト内の各エミッター名は、コロンまたはセミコロンで区切る。このオプションを介して指定できるエミッターのリストについては、196 ページの『サポートされているエミッター』を参照。
-h	idlc コマンド構文の要旨を標準出力に書き込む。
-I<*include-directory*>	IDL コンパイラーが # 組み込みファイルを検索するために使用するディレクトリーのリストに、現行ディレクトリーを追加する。-I オプションの他に、IDLC_INCLUDE 環境変数を使用してリストを指定することもできる。この場合、*include-directory* 名は、パス区切り文字 (¥) で区切る。
-i<*file-name*>	コンパイルするファイルの名前を指定する。このオプションは、IDL が含まれているが、ファイル名に「.idl」以外の拡張子が付いているファイルをコンパイルしたい場合にだけ使用する。
-J	IDL コンパイラーが使用する Java インタープリターにオプションを渡す。たとえば、`-J"-mx32m"` のようにして、インタープリターのヒープ・サイズを 32M に指定することができる。
-m<*name[=value]*>	エミッターが作成するバインディングに影響を及ぼす出力修飾子を指定する。このオプションを介して指定できる出力修飾子のリストについては、197ページの『サポートされている出力修飾子』を参照。
-p	IDL コンパイラーに対する -D_PRIVATE_ プリプロセッサー変数を指定する。
-s<*emit-list*>	-e<*emit-list*> と同じ。
-V	idlc コマンドのバージョン番号を示す。
-v	冗長モードを指定する。IDL コンパイラーが発行するすべての内部コマンド (およびその引き数) を参照できる。

サポートされているエミッター

表13 は、IDL コンパイラーを呼び出すための idlc コマンドに e オプションを付けて、指定できるエミッターのリストです。

表 13. IDL コンパイラーの呼び出し用コマンドに指定できる、サポートされているエミッター

エミッターのタイプ	エミッター名および説明
リポジトリー・エミッター	**ir** CORBA インターフェース・リポジトリーを、このコンパイル単位内のインターフェースで更新する。
C++ ファイル・エミッター	**hh** C++ 使用バインディングを作成する。idlc コマンドに -m オプションを使って修飾子を指定しないと、エミッターは、リモートの異言語間オペレーションに対するサポートによって、バインディングを作成する。**cpponly**、**localonly**、または **somthis** 修飾子を指定すると、エミッターは特殊なバインディングを作成する。詳しくは、-m オプションについての説明を参照。 **ic** Component Broker C++ マネージド・オブジェクト・インプリメンテーション・テンプレートを作成する。idlc コマンドに -m オプションを使って修飾子を指定しないと、エミッターは、独立型 ORB での使用に適した純粋な CORBA C++ バインディングを作成する。**mo** 修飾子を指定すると、エミッターは Component Broker マネージド・オブジェクトをサポートするバインディングを作成する。詳しくは、-m オプションについての説明を参照。 **ih** C++ インプリメンテーション・ヘッダーを作成する。idlc コマンドに -m オプションを使って修飾子を指定しないと、エミッターは、独立型 ORB での使用に適した純粋な CORBA C++ バインディングを作成する。**mo** 修飾子を指定すると、エミッターは Component Broker マネージド・オブジェクトをサポートするバインディングを作成する。詳しくは、-m オプションについての説明を参照。 **sc** C++ サーバー・インプリメンテーション・バインディングを作成する。一般に、このファイルは <class_name>_S.cpp ファイルである。idlc コマンドに -m オプションを使って修飾子を指定しないと、エミッターは、リモートの異言語間オペレーションに対するサポートによって、バインディングを作成する。**cpponly**、**localonly**、または **somthis** 修飾子を指定すると、エミッターは特殊なバインディングを作成する。詳しくは、-m オプションについての説明を参照。 **uc** クライアント側のインプリメンテーション・バインディングを作成する。これは、<class_name>_C.cpp ファイルである。idlc コマンドに -m オプションを使って修飾子を指定しないと、エミッターは、リモートの異言語間オペレーションに対するサポートによって、バインディングを作成する。**cpponly**、**localonly**、または **somthis** 修飾子を指定すると、エミッターは特殊なバインディングを作成する。詳しくは、-m オプションについての説明を参照。

表 13. IDL コンパイラーの呼び出し用コマンドに指定できる、サポートされているエミッター (続き)

エミッターのタイプ	エミッター名および説明
Java ファイル・エミッター	**bj** Java で作成されたビジネス・オブジェクトをサポートするファイルを作成する。以下のようなファイルがある。 • _<interface_name>Skeleton.java を置き換える _<interface_name>Wrapper.java • Java ビジネス・オブジェクトに関連した C++ マネージド・オブジェクトに対するインプリメンテーション側のプロキシーである、_<interface_name>Impl.java **sj** Java インプリメンテーション・スケルトンを作成する。 **uj** 異言語間 Java 使用バインディングを作成する。

サポートされている出力修飾子

表14 は、IDL コンパイラーを呼び出すための `idlc` コマンドに、`m` オプションを付けて、指定できる出力修飾子のリストです。これらの出力修飾子は、エミッターが作成するバインディングに影響を及ぼします。

表 14. IDL コンパイラーの呼び出し用に指定できる、サポートされている出力修飾子

名前 / 値	説明
cpponly	異言語間バインディングの作成を抑制し、標準 ORB での使用に適した標準 CORBA C++ バインディングを作成する。この修飾子は、hh、sc、および uc エミッターが作成するバインディングに影響を及ぼす。
dllname=<value>	エミッターが Windows NT インポート / エクスポート仕様を配置するクラスを含む、IDL の名前を指定する。
IRforce	インターフェース・リポジトリーを既存のオブジェクトと同じ名前のオブジェクトで更新するときに、ir エミッターが既存のオブジェクトを強制的に置き換えるようにする。
LINKAGE=<value>	カスタマイズされた C++ リンケージ修飾子を生成されたバインディング内に挿入する。
localonly	IDL ファイル内のすべての頻繁に派生するインターフェースに対するローカル・オブジェクトにアクセスするためにだけ使用できるバインディングを生成する。この修飾子は、IDL ファイルそのものの #pragma に **localonly** キーワードを使用する代わりになる。

表 14. IDL コンパイラーの呼び出し用に指定できる、サポートされている出力修飾子 (続き)

mo	Component Broker マネージド・オブジェクトをサポートするバインディングを生成する。この修飾子を指定しないと、エミッターは独立型 ORB での使用に適した純粋な CORBA C++ バインディングを作成する。この修飾子は、ih および ic エミッターにだけ影響を及ぼす。
nohelper	uj エミッターが <interface_name>Helper.java ファイルを生成しないようにする。
noholder	uj エミッターが <interface_name>Holder.java ファイルを生成しないようにする。
noimpl	bj エミッターが <interface_name>Impl.java ファイルを生成しないようにする。
noimplbase	sj エミッターが <interface_name>ImplBase.java ファイルを生成しないようにする。
nointerface	uj エミッターが <interface_name>.java ファイルを生成しないようにする。
noskeleton	sj エミッターが <interface_name>Skeleton.java ファイルを生成しないようにする。
nostub	uj エミッターが <interface_name>Stub.java ファイルを生成しないようにする。
notcconsts	C++ TypeCode 定数と多重定義された Any 演算子の生成を除去する。
nowrapper	bj エミッターが <interface_name>Wrapper.java ファイルを生成しないようにする。
orbadapter	C++ ORB による Java インプリメンテーションのディスパッチを可能にする C++ バインディングを生成する。
postInclude= <file-name>	ファイル `#include <file-name>` の直前で、次の行を使用バインディング (.hh) ファイルに追加する。
preInclude= <file-name>	corba.h: `#include <file-name>` に対する include ステートメントの直前で、次の行を使用バインディング (.hh) ファイルに追加する。
tie	継承ではなく代行を想定したバインディングを生成する。

idlc コマンドの結果

idlc コマンドを入力しても、コマンドそのもののエミット・リストまたは IDLC_EMIT 環境変数のいずれかを介してエミッターを指定しないと、IDL コンパイラーは、入力ファイル内の構文エラーを検査するだけです。それ以外の場合、IDL コンパイラーは指定されたエミッターに基づいて出力ファイルを作成します。これらの出力ファイルには、言語固有の使用法と IDL インターフェースに対するインプリメンテーション・バインディングが含まれています。出力ファイル名は、次のように、選択した言語によって異なります。

- C++ 向けエミッターを指定すると、各エミッターは 1 種類の出力ファイルを作成します。このファイルの名前は IDL ソース・ファイルと同じ名前で始まります。たとえば、Policy.idl という名前の IDL ソース・ファイルの場合、各エミッターは以下のような出力ファイルを作成します。

  エミッター
  出力ファイル名

  ic Policy_I.cpp

  ih Policy.ih

  hh Policy.hh

  sc Policy_S.cpp

  uc Policy_C.cpp

  IDL コンパイラーは、これらの出力ファイルを idlc コマンドで指定されたディレクトリーか、あるいは、ディレクトリーが指定されなかった場合は現行ディレクトリーの中に置きます。

- Java 向けのエミッターを指定すると、各エミッターは 1 つまたは複数のタイプの出力ファイルを作成することができます。たとえば、Policy.idl という名前の IDL ソース・ファイルの場合、各エミッターは以下のような出力ファイルを作成します。

  エミッター
  出力ファイル名

  bj
  - _PolicyWrapper.java
  - _PolicyImpl.java

  sj
  - _PolicySkeleton.java
  - _PolicyImplBase.java

uj

- Policy.java
- PolicyHelper.java
- PolicyHolder.java
- _PolicyStub.java
- ポリシー・インターフェースのエレメントに対する追加の .java ファイル。たとえば、ポリシー向けの IDL によって InvalidAmount という名前の例外が定義されると、uj エミッターは InvalidAmount.java という名前の出力ファイルを作成します。

Java パッケージの名前と Java ソース・ファイルが置かれているディレクトリー構造の間で整合性を保つために、IDL コンパイラーは出力ファイルを置くサブディレクトリーを作成します。サブディレクトリーは、idlc コマンドで指定したディレクトリーのサブディレクトリーか、ディレクトリーが指定されなかった場合は現行ディレクトリーのサブディレクトリーです。

付録D. 特記事項

本書において、日本では発表されていない IBM 製品 (機械およびプログラム)、プログラミングまたはサービスについて言及または説明する場合があります。しかし、このことは、弊社がこのような IBM 製品、プログラミングまたはサービスを、日本で発表する意図があることを必ずしも示すものではありません。本書でIBM ライセンス・プログラムまたは他の IBM 製品に言及している部分があっても、このことは当該プログラムまたは製品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。IBM 製品、プログラム、またはサービスに代えて、IBM の有効な知的所有権またはその他の法的に保護された権利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 によって明示的に指定されたものを除き、他社の製品と組み合わせた場合の操作の評価と検証はお客様の責任で行っていただきます。

IBM は、本書で解説されている主題について特許権 (特許出願を含む)、商標権、または著作権を所有している場合があります。本書の提供は、これらの特許権、商標権、および著作権について、本書で明示されている場合を除き、実施権、使用権等を許諾することを意味するものではありません。

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

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

本書に対して、周期的に変更が行われ、これらの変更は、文書の次版に組み込まれます。IBM は、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

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

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

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

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 Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

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

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書で使用している例について

本書で使用している例は、IBM が作成した単なるサンプルです。これらの例は、標準または IBM 製品の一部ではなく、単に、お客様のアプリケーション開発を支援することを目的として提供されています。例は、「現存するままの」状態で提供されています。IBM は、これらの例の機能またはパフォーマンスに関して、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明示もしくは黙示の保証責任を負わないものとします。IBM は、これらの例の使用によって生じたいかなる損害に対しても、たとえ、そのような損害の可能性を通知している場合であっても、その責任を負いません。

これらの例は、上記の免責条項をそのまま適用することを条件として、無料で配布し、複製し、改変し、他のソフトウェアに取り込むことができます。

プログラミング・インターフェース情報

本書には、お客様が WebSphere for z/OS のサービスを受けるためのプログラムを作成することを目的とした、プログラミング・インターフェースについての情報が含まれています。

商標

以下は、米国およびその他の国における IBM Corporation の商標または登録商標です。

AIX
CICS
DB2IBM
IMS
IMS/ESA
Language Environment
Open Class
OS/390
RACF
VisualAge
VTAM
WebSphere
z/OS

Java およびすべての Java 関連の商標は、Sun Microsystems, Inc. の米国およびその他の国における商標です。

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 がライセンスしている米国およびその他の国における登録商標です。

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

用語集

本書で使用されている用語の詳細については、以下のいずれかの情報源をご利用ください。

- *WebSphere Application Server エンタープライズ版 Component Broker 用語集*, SD88-7380 は、次のインターネット・サイトにあります。

  `http://www.ibm.com/jp/software/websphere/appserv/`

- Sun Microsystems Glossary of Java Technology-Related Terms は、次のインターネット・サイトにあります。

  `http://java.sun.com/docs/glossary.html`

探している用語が見つからない場合は、次のインターネット・サイトにある、*IBM Glossary of Computing Terms* を参照してください。

`http://www.ibm.com/ibm/terminology/`

あるいは、以下のサイトにある Sun の Web サイトを参照してください。

`http://www.sun.com/`

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]

[ラ行]

[ワ行]

A

C

D

H

I

J

L

M

N

O

P

S

T

V

［特殊文字］

プログラム番号: 5655-F31

Printed in Japan

SA88-8658-00

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