BEA Tuxedo®

BEA Tuxedo CORBA University サンプル・アプリケーション

BEA Tuxedo 8.1
2003 年 1 月

Copyright

Restricted Rights Legend

Trademarks or Service Marks

目次

4. Security サンプル・アプリケーション

5. Transactions サンプル・アプリケーション

6. Wrapper サンプル・アプリケーション

7. Production サンプル・アプリケーション

このマニュアルについて

このマニュアルでは、BEA Tuxedo ソフトウェアに付属の University サンプル・アプリケーションについて説明します。

このマニュアルでは、以下の内容について説明します。

- 第 1 章 「Introduction」では、このサンプル・アプリケーションの概要について説明します。
- 第 2 章 「環境設定」では、システム要件、および UBBCONFIG ファイルでシステム環境の環境変数とパラメータを設定する方法について説明します。
- 第 3 章 「Basic サンプル・アプリケーション」では、Basic サンプル・アプリケーションについて説明します。
- 第 4 章 「The Security Sample Application」では、Security サンプル・アプリケーションについて説明します。
- 第 5 章 「The Transactions Sample Application」では、Transactions サンプル・アプリケーションについて説明します。
- 第 6 章 「The Wrapper Sample Application」では、Wrapper サンプル・アプリケーションについて説明します。
- 第 7 章 「Production サンプル・アプリケーション」では、Production サンプル・アプリケーションについて説明します。
- 付録 A 「Setting Up the Database」では、University サンプル・アプリケーションで使用するデータベースの設定方法について説明します。

対象読者

このマニュアルは、BEA Tuxedo 製品の CORBA 環境を理解するために役立つ段階的なサンプルを必要としている、アプリケーション設計者およびプログラマを対象としています。

e-docs Web サイト

BEA Tuxedo 製品のマニュアルは BEA 社の Web サイトで参照することができます。BEA ホーム・ページの [製品のドキュメント] をクリックするか、または http://edocs.beasys.co.jp/e-docs/index.html に直接アクセスしてください。

マニュアルの印刷方法

このマニュアルは、ご使用の Web ブラウザで一度に 1 ファイルずつ印刷できます。Web ブラウザの [ファイル] メニューにある [印刷] オプションを使用してください。

このマニュアルの PDF 版は、Web サイト上にあります。また、マニュアルの CD-ROM にも収められています。この PDF を Adobe Acrobat Reader で開くと、マニュアル全体または一部をブック形式で印刷できます。PDF 形式を利用するには、BEA Tuxedo マニュアルのホーム・ページにある [PDF 版] ボタンをクリックして、印刷するマニュアルを選択します。

Adobe Acrobat Reader をお持ちでない場合は、Adobe 社の Web サイト (http://www.adobe.co.jp) から無償でダウンロードできます。

関連情報

CORBA、分散オブジェクト・コンピューティング、トランザクション処理、C++ と Java によるクライアント・プログラミング、および C++ によるサーバ・プログラミングの詳細については、BEA Tuxedo オンライン・マニュアルの「Bibliography」を参照してください。

サポート情報

皆様の BEA Tuxedo マニュアルに対するフィードバックをお待ちしています。ご意見やご質問がありましたら、電子メールで docsupport-jp@bea.com までお送りください。お寄せいただきましたご意見は、BEA Tuxedo マニュアルの作成および改訂を担当する BEA 社のスタッフが直接検討いたします。

電子メール メッセージには、BEA Tuxedo 8.0 リリースのマニュアルを使用していることを明記してください。

BEA Tuxedo に関するご質問、または BEA Tuxedo のインストールや使用に際して問題が発生した場合は、www.bea.com の BEA WebSUPPORT を通して BEA カスタマ・サポートにお問い合わせください。カスタマ・サポートへの問い合わせ方法は、製品パッケージに同梱されているカスタマ・サポート・カードにも記載されています。

カスタマ・サポートへお問い合わせの際には、以下の情報をご用意ください。

- お客様のお名前、電子メール・アドレス、電話番号、Fax 番号
- お客様の会社名と会社の住所
- ご使用のマシンの機種と認証コード
- ご使用の製品名とバージョン
- 問題の説明と関連するエラー・メッセージの内容

表記上の規則

このマニュアルでは、以下の表記規則が使用されています。

規則	項目
太字	用語集に定義されている用語を示します。
Ctrl + Tab	2 つ以上のキーを同時に押す操作を示します。
イタリック体	強調またはマニュアルのタイトルを示します。

等幅テキスト	コード・サンプル、コマンドとオプション、データ構造とメンバ、データ型、ディレクトリ、およびファイル名と拡張子を示します。また、キーボードから入力するテキストも等幅テキストで表示します。
	例 :
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

太字の等幅テキスト	コード内の重要な語を示します。 例 : `void` **`commit`** `( )`
斜体の等幅テキスト	コード内の変数を示します。 例 : `String` *`expr`*

規則	項目
大文字のテキスト	デバイス名、環境変数、および論理演算子を示します。 例 : LPT1 SIGNON OR
{ }	構文の行で、選択肢の組み合わせを示します。かっこは入力しません。
[ ]	構文の行で、オプション項目を示します。かっこは入力しません。 例 : `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
|	構文の行で、相互に排他的な選択肢の区切りとして使います。記号は入力しません。
...	コマンド・ラインで、以下のいずれかの場合を示します。 ■ コマンド・ラインで、同じ引数を繰り返し使用できることを示します。 ■ 文で、追加のオプション引数が省略されていることを示します。 ■ 追加のパラメータ、値、またはその他の情報を入力できることを示します。 記号は入力しません。 例 : `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
. . .	コード例または構文の行で、項目が省略されていることを示します。記号は入力しません。

章

1 はじめに

ここでは、BEA Tuxedo 製品で CORBA 環境向けに用意された University サンプル・アプリケーションについて説明します。クライアントおよびサーバのプログラマは、このサンプル・アプリケーションを利用することで、CORBA 環境を使用した分散クライアント / サーバ・アプリケーションの開発に関する基本概念を理解できます。また、サンプル・アプリケーションには、BEA Tuxedo 製品の高度な CORBA 機能が数多く導入されています。

ここでは、以下の内容について説明します。

- University サンプル・アプリケーションの概要
- University サンプル・アプリケーションの命名規則

University サンプル・アプリケーションの概要

BEA Tuxedo ソフトウェア・キットには、CORBA サンプル・アプリケーション・スイートが付属しています。これは、University でインプリメントされたクライアント・アプリケーションとサーバ・アプリケーションをベースにしています。各 University サンプル・アプリケーションでは、順を追って例を体験しながら、CORBA の新機能について解説します。University サンプル・アプリケーションは、BEA Tuxedo 製品の特定の CORBA 機能に関連する手順やプロセスに焦点を絞って解説するために簡素化されています。

表 1-1 では、各 University サンプル・アプリケーションについて説明します。

表 1-1 University サンプル・アプリケーション

University **サンプル・アプリケーション**	**説明**
Basic	CORBA クライアント・アプリケーションとサーバ・アプリケーションの作成方法、CORBA アプリケーションのコンフィギュレーション方法、Basic サンプル・アプリケーションに含まれるクライアント・アプリケーションとサーバ・アプリケーションのビルドおよび実行方法について説明します。CORBA C++、CORBA Java、および ActiveX クライアント・アプリケーションと CORBA C++ サーバ・アプリケーションが提供されています。
Security	Basic サンプル・アプリケーションの CORBA クライアント・アプリケーション、および CORBA アプリケーションのコンフィギュレーションにアプリケーション・レベルのセキュリティを追加します。
Transactions	Basic サンプル・アプリケーションの CORBA クライアント・アプリケーションとサーバ・アプリケーションにトランザクション・オブジェクトを追加します。Transactions サンプル・アプリケーションでは、インプリメンテーション・コンフィギュレーション・ファイル (ICF) を使用して、CORBA オブジェクトのトランザクション方針を定義する方法を示します。
Wrapper	CORBA オブジェクトとして ATMI アプリケーションをラッピングする方法を示します。
Production	CORBA サーバ・アプリケーションの複製方法、状態を持たない CORBA オブジェクトの作成方法、および CORBA サーバ・アプリケーションでのファクトリ・ベース・ルーティングのインプリメント方法を示します。

University サンプル・アプリケーションを使用する際は、以下のマニュアルも併せて参照してください。

- *『BEA Tuxedo CORBA アプリケーション入門』*
- *『BEA Tuxedo CORBA クライアント・アプリケーションの開発方法』*
- *『BEA Tuxedo CORBA サーバ・アプリケーションの開発方法』*

University サンプル・アプリケーションの命名規則

表 1-2 では、University サンプル・アプリケーションのコードでの命名規則の一覧と説明を示します。

表 1-2 University サンプル・アプリケーションの命名規則

規則	説明
`crs`	course (コース) の省略形
`syn`	synopsis (概要) の省略形
`det`	details (詳細) の省略形
`lst`	list (リスト) の省略形
`enum`	enumerator (列挙子) の省略形
`stu`	student (学生) の省略形
`num`	number (数) の省略形
`cur`	current (カレント) の省略形
`_oref`	`CORBA::Object` リファレンス
`_ref`	型付きオブジェクト・リファレンス

表 1-2 University サンプル・アプリケーションの命名規則 (続き) (続き)

規則	説明
p_	ptr の省略形
v_	var の省略形
s_	file static data (ファイル静的データ) の省略形
m_	class member data (クラス・メンバ・データ) の省略形
メソッド名および変数名	名前にはすべて小文字を使用します。メソッド名内で文字を区切るには下線 (_) を使用します。たとえば、`m_v_crs_syn_list` はメンバ・データである var で、コースの概要のリストを格納しています。
型名	型名の先頭には小文字を使用します。また、型名で文字を区切る場合にも小文字を使用します。型名では省略形は使用しません。たとえば、`UniversityB::CourseSynopsisEnumerator_var` などです。

2 環境設定

ここでは、University サンプル・アプリケーションを実行できるように CORBA アプリケーションをコンフィギュレーションする方法について説明します。

ここでは、以下の内容について説明します。

- システムの前提条件
- setenv および UBBCONFIG ファイルの編集

各サンプル・アプリケーションのディレクトリには、トラブルシューティング情報が記載された `Readme.txt` のコピー、およびサンプル・アプリケーションの設定に関する最新情報が格納されています。

University サンプル・アプリケーションで使用できるようにデータベースを設定する方法については、「データベースの設定」を参照してください。

システムの前提条件

製品でサポートされているオペレーティング・システムのプラットフォームについては、『BEA Tuxedo システムのインストール』を参照してください。

University サンプル・アプリケーションでクライアント・アプリケーションを実行するには、以下の開発ツールが必要になります。

- Visual C++ Version 5.0 Service Pack 3
- ActiveX クライアント・アプリケーションを実行する場合、Visual Basic Version 5.0 Service Pack 3

- CORBA Java クライアント・アプリケーションを実行する場合、JDK (Java Development Kit) 1.3

setenv および UBBCONFIG ファイルの編集

University サンプル・アプリケーションが正常に動作するには、setenv および UBBCONFIG ファイルのパラメータを次のように設定する必要があります。

- setenv ファイルでは、サンプル・アプリケーションのビルドおよび実行に必要なシステム環境変数を設定します。各サンプル・アプリケーションのディレクトリには、一意な setenv ファイルが格納されています。setenv ファイルの名前は、そのファイルを使用するサンプル・アプリケーションを示します。たとえば、setenvb は、Basic サンプル・アプリケーション用のファイルです。各サンプル・アプリケーションのディレクトリには、Windows 2000 用と UNIX オペレーティング・システム用の setenv ファイルが格納されています。setenv ファイルに固有のファイル名については、表 0-1 を参照してください。
- UBBCONFIG ファイルは、サンプル・アプリケーションのコンフィギュレーション・ファイルです。UBBCONFIG ファイルでは、サンプル・アプリケーションでのクライアント・アプリケーションとサーバ・アプリケーションの動作に関するパラメータを定義します。各サンプル・アプリケーションのディレクトリには、一意な UBBCONFIG ファイルが格納されています。UBBCONFIG ファイルの名前は、そのファイルを使用するサンプル・アプリケーションを示します。たとえば、ubb_b は、Basic サンプル・アプリケーション用のファイルです。各サンプル・アプリケーションのディレクトリには、Windows 2000 用と UNIX オペレーティング・システム用の UBBCONFIG ファイルが格納されています。UBBCONFIG ファイルに固有のファイル名については、表 0-1 を参照してください。

setenv と UBBCONFIG ファイルの情報は一致している必要があります。以降の節では、setenv と UBBCONFIG ファイルの編集方法について説明します。

setenv および UBBCONFIG ファイルの命名規則

表 0-1 では、setenv および UBBCONFIG ファイルの命名規則について説明します。サンプル・アプリケーションを識別する文字は太字で示しています。

表 0-1 setenv および UBBCONFIG ファイルの命名規則

University **サンプル・アプリケーション**	**命名規則**
Basic	■ setenv**b**.cmd — Windows 2000 用 setenv ファイル ■ setenv**b**.sh — UNIX 用 setenv ファイル ■ ubb_**b**.nt — Windows 2000 用 UBBCONFIG ファイル ■ ubb_**b**.mk — UNIX 用 UBBCONFIG ファイル
Security	■ setenv**s**.cmd — Windows 2000 用 setenv ファイル ■ setenv**s**.sh — UNIX 用 setenv ファイル ■ ubb_**s**.nt — Windows 2000 用 UBBCONFIG ファイル ■ ubb_**s**.mk — UNIX 用 UBBCONFIG ファイル
Transactions	■ setenv**t**.cmd — Windows 2000 用 setenv ファイル ■ setenv**t**.sh — UNIX 用 setenv ファイル ■ ubb_**t**.nt — Windows 2000 用 UBBCONFIG ファイル ■ ubb_**t**.mk — UNIX 用 UBBCONFIG ファイル
Wrapper	■ setenv**w**.cmd — Windows 2000 用 setenv ファイル ■ setenv**w**.sh — UNIX 用 setenv ファイル ■ ubb_**w**.nt — Windows 2000 用 UBBCONFIG ファイル ■ ubb_**w**.mk — UNIX 用 UBBCONFIG ファイル

表 0-1 setenv および UBBCONFIG ファイルの命名規則 (続き)

University サンプル・アプリケーション	命名規則
Production	■ `setenv`**`p`**`.cmd` ― Windows 2000 用 `setenv` ファイル ■ `setenv`**`p`**`.sh` ― UNIX 用 `setenv` ファイル ■ `ubb_`**`p`**`.nt` ― Windows 2000 用 `UBBCONFIG` ファイル ■ `ubb_`**`p`**`.mk` ― UNIX 用 `UBBCONFIG` ファイル

setenv のパラメータの設定

表 0-2 では、setenv ファイルで変更する必要があるパラメータの一覧を示しています。

表 0-2 setenv ファイルのパラメータ

パラメータ	説明

APPDIR	サンプル・アプリケーション・ファイルをコピーしたディレクトリ・パス。たとえば、次のように入力します。
	Windows 2000
	`APPDIR=c:\`*`work`*`\university\basic`
	UNIX
	`APPDIR=/usr/`*`work`*`/university/basic`

TUXCONFIG	コンフィギュレーション・ファイルのディレクトリ・パスと名前。たとえば、次のように入力します。
	Windows 2000
	`TUXCONFIG=c:\work\university\basic\tuxconfig`
	UNIX
	`TUXCONFIG=/usr/`*`work`*`/university/basic/tuxconfig`

表 0-2 setenv ファイルのパラメータ (続き)

パラメータ	説明

TUXDIR	BEA Tuxedo ソフトウェアをインストールしたディレクトリ・パス。たとえば、次のように入力します。
	Windows 2000
	`TUXDIR=c:\Tux8`
	UNIX
	`TUXDIR=/usr/local/Tux8`

ORACLE_HOME	Oracle ソフトウェアをインストールしたディレクトリ・パス。たとえば、次のように入力します。
	Windows 2000
	`ORADIR=c:\Orant`
	UNIX
	`ORACLE_HOME=/usr/local/oracle`

NETSCAPE	Netscape Enterprise Server ソフトウェアをインストールしたディレクトリ・パス。たとえば、次のように入力します。
	Windows 2000
	`NETSCAPE=c:\Netscape\SuiteSpot`
	UNIX
	`NETSCAPE=/usr/local/netscape/suitespot`
	このパラメータは、CORBA Java クライアント・アプリケーションを使用する場合にのみ指定する必要があります。

JARTYPE	使用している JDK のバージョンを指定します。 Windows 2000 `JARTYPE = JDK` または `jdk` UNIX `JARTYPE = JDK` または `jdk` JARTYPE パラメータを指定しない場合は、JDK 1.1 を使用しているものと見なされます。このパラメータは、CORBA Java サンプル・アプリケーションを使用する場合にのみ指定する必要があります。

表 0-2 setenv ファイルのパラメータ (続き)

パラメータ	説明
JDKDIR	JDK ソフトウェアをインストールしたディレクトリ・パス。たとえば、次のように入力します。 Windows 2000 JDKDIR=c:\JDK1.2 UNIX JDKDIR=/usr/local/jdk1.1.6 このパラメータは、CORBA Java サンプル・アプリケーションを使用する場合にのみ指定する必要があります。
TOBJADDR	サーバ・アプリケーションとは異なるマシン上で CORBA C++ クライアント・アプリケーションを使用している場合、サーバ・アプリケーションが動作しているマシンのホストとポートを入力します。ホストとポートは、そのマシンの UBBCONFIG ファイルの記述と同じになるように、大文字と小文字を区別して正確に指定しなければなりません。たとえば、//BEANIE:2500 のように指定します。
USERID	Oracle データベースのリモート・インスタンスを使用している場合、次の形式で指定します。 USERID=username/password@*aliasname* これは、Oracle データベースのリモート・インスタンスの設定時に定義した情報と同じです。 Oracle データベースのローカル・インスタンスを使用している場合は、次の形式で指定します。 USERID=username/password
ORACLE_SID	Oracle データベースのインスタンス ID。Windows 2000 の場合、ORACLE_SID は自動的に ORCL にデフォルト設定されるので、このパラメータの指定は不要です。
CCMPL	C コンパイラのディレクトリの場所。このパラメータは、通常のインストール先のディレクトリに設定します。インストール先の場所と指定したディレクトリの場所が一致していることを確認し、必要に応じて場所を変更してください。このパラメータの適用対象は、UNIX オペレーティング・システムのみです。

表 0-2 setenv ファイルのパラメータ (続き)

パラメータ	説明
CPPCMPL	C++ コンパイラのディレクトリの場所。このパラメータは、通常のインストール先のディレクトリに設定します。インストール先の場所と指定したディレクトリの場所が一致していることを確認し、必要に応じて場所を変更してください。このパラメータの適用対象は、UNIX オペレーティング・システムのみです。
CPPINC	C++ インクルード・ディレクトリのディレクトリの場所。このパラメータは、通常のインストール先のディレクトリに設定します。インストール先の場所と指定したディレクトリの場所が一致していることを確認し、必要に応じて場所を変更してください。このパラメータの適用対象は、UNIX オペレーティング・システムのみです。
SHLIB_PATH、LD_LIBRARY_PATH、または LIBPATH	共有ライブラリのディレクトリの場所。このパラメータは、通常のインストール先のディレクトリに設定します。インストール先の場所と指定したディレクトリの場所が一致していることを確認し、必要に応じて場所を変更してください。このパラメータの適用対象は、UNIX オペレーティング・システムのみです。
PROC	Oracle Programmer C/C++ SQL Precompiler のディレクトリの場所。このパラメータを指定する必要があるのは、Windows 2000 オペレーティング・システムを使用している場合のみです。
PRODIR	Oracle Programmer C/C++ SQL Precompiler のディレクトリの場所。このパラメータを指定する必要があるのは、Windows 2000 オペレーティング・システムを使用している場合のみです。

UBBCONFIG のパラメータの設定

表 0-3 では、UBBCONFIG ファイルで変更する必要があるパラメータの一覧を示しています。

表 0-3 UBBCONFIG **ファイルのパラメータ**

パラメータ	説明

MY_SERVER_MACHINE	このパラメータを削除して、サーバ・マシンの名前に置き換えます。
	Windows 2000 の場合、サーバ・マシン名を取得するには、MS-DOS プロンプトで次のコマンドを入力します。
	`set COMPUTERNAME`
	UNIX の場合、サーバ・マシン名を取得するには、シェル・プロンプトで次のコマンドを入力します。
	`prompt>uname -n`
	サーバ・マシン名は、コマンドの出力表示と同じになるように、大文字と小文字を区別して正確に入力しなければなりません。
	表示されたサーバ・マシン名を指定します。たとえば、BEANIE のように指定します。
	マシンの完全名は、二重引用符で囲む必要があります。たとえば、次のように入力します。"beanie.bea.com".

APPDIR	サンプル・アプリケーション・ファイルをコピーしたディレクトリの絶対パス。ディレクトリ・パスは、二重引用符で囲む必要があります。たとえば、次のように入力します。
	Windows 2000
	`APPDIR="c:\work\university\basic"`
	UNIX
	`APPDIR="/usr/work/university/basic"`
	このパラメータは、setenv ファイルの APPDIR パラメータと一致している必要があります。

TUXCONFIG	コンフィギュレーション・ファイルのディレクトリの絶対パス。これは、サンプル・アプリケーションのサブディレクトリです。ディレクトリ・パスは、二重引用符で囲む必要があります。たとえば、次のように入力します。
	Windows 2000
	`TUXCONFIG="c:\work\university\basic\tuxconfig"`
	UNIX
	`TUXCONFIG="usr/work/university/basic/tuxconfig"`
	このパラメータは、setenv ファイルの TUXCONFIG パラメータと一致している必要があります。

表 0-3 UBBCONFIG ファイルのパラメータ (続き)

パラメータ	説明

TUXDIR	BEA Tuxedo ソフトウェアをインストールしたディレクトリの絶対パス。ディレクトリ・パスは、二重引用符で囲む必要があります。たとえば、次のように入力します。
	Windows 2000
	`TUXDIR="c:\Tux8"`
	UNIX
	`TUXDIR="/usr/local/Tux8"`
	このパラメータは、setenv ファイルの TUXDIR パラメータと一致している必要があります。

CLOPT (ISL プロセス用)	サーバ・アプリケーションがインストールされているマシンのホスト名とポート番号を入力します。たとえば、次のように入力します。
	`ISL`
	`SRVGRP = SYS_GRP`
	`SRVID =`
	`CLOPT = "-A --n //BEANIE:2500"`

OPENINFO	Transactions、Wrapper、または Production の各サンプル・アプリケーションを使用している場合、このパラメータを Oracle データベースに指定する必要があります。
	Oracle データベースのリモート・インスタンスを使用している場合、OPENINFO パラメータは次のように指定します。
	`OPENINFO = "Oracle_XA:Oracle_XA+SqlNet=`*`aliasname`*`+Acc=P/`*`account/password`*`+SesTM=100+LogDir=.+MaxCur=5"`
	たとえば、Windows 2000 の場合は次のように指定します。
	`OPENINFO = "Oracle_XA:Oracle_XA+SqlNet=ORCL+Acc=P/scott/tiger+SesTM=100+LogDir=.+MaxCur=5"`
	Oracle データベースのローカル・インスタンスを使用している場合、OPENINFO パラメータは次のように指定します。
	`OPENINFO = "Oracle_XA:Oracle_XA+Acc=P/`*`account/password`*`+SesTM=100+LogDir=.+MaxCur=5"`
	たとえば、Windows 2000 の場合は次のように指定します。
	`OPENINFO = "Oracle_XA:Oracle_XA+Acc=P/scott/tiger+SesTM=100+LogDir=.+MaxCur=5"`

setenv コマンドの実行

University サンプル・アプリケーションを使用する前に、setenv スクリプトを実行して、Oracle データベース設定で加えた変更、およびコンフィギュレーションがすべてシステム環境変数に反映されていることを確認しておく必要があります。setenv コマンドを実行する手順については、各サンプル・アプリケーションのビルドに関する説明の中で記載されています。

注記　University サンプル・アプリケーションの makefile では、Microsoft Visual C++ が Windows 2000 の次の場所にインストールされていることを前提としています。

```
c:\Progra~1\Devstu~1\VC
```

上記とは異なるディレクトリに Microsoft Visual C++ をインストールしている場合は、次のコマンドを実行して適切なシステム環境変数を設定します。

```
c:\Progra~1\Devstu~1\VC\Bin\VCVARS32.bat
```

Oracle Pro*C/C++ コンパイラでは短縮名が使用されます。したがって、システム変数を正しく設定するために、ディレクトリ・パスでは「~」を使用する必要があります。

章

3 Basic サンプル・アプリケーション

ここでは、以下の内容について説明します。

- Basic サンプル・アプリケーションのしくみ
- Basic サンプル・アプリケーションの OMG IDL
- クライアント・スタブとスケルトンの生成
- クライアント・アプリケーションの記述
- Basic サンプル・アプリケーションのコンフィギュレーション
- Basic サンプル・アプリケーションのビルド
- Basic サンプル・アプリケーションのコンパイル
- Basic サンプル・アプリケーションの実行
- Basic サンプル・アプリケーションのクライアント・アプリケーションの使用方法

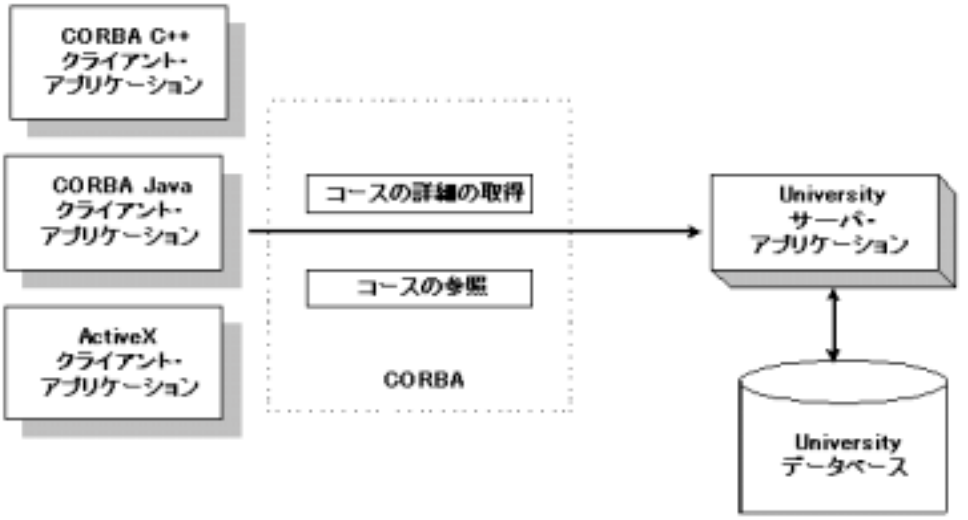
トラブルシューティング情報については、`\basic` ディレクトリにある `Readme.txt` を参照してください。また、Basic サンプル・アプリケーションの使用方法に関する最新の情報も参照してください。

CORBA アプリケーションに関連する概念、および CORBA アプリケーションの開発プロセスの説明については、『BEA Tuxedo CORBA アプリケーション入門』を参照してください。

Basic サンプル・アプリケーションのしくみ

Basic サンプル・アプリケーションでは、ユーザは利用可能なコースを参照したり、選択したコースの詳細を取得したりできます。図 3-1 に、Basic サンプル・アプリケーションのしくみを示します。

図 3-1Basic サンプル・アプリケーション



Basic サンプル・アプリケーションでは、次の機能について示します。

- CORBA クライアント・アプリケーションとサーバ・アプリケーションの作成
- CORBA アプリケーションのコンフィギュレーション情報の定義
- BEA Tuxedo 製品で用意されている CORBA コマンドとツールを使用した、クライアント・アプリケーションとサーバ・アプリケーションのビルド

Basic サンプル・アプリケーションの OMG IDL

クライアント・アプリケーションとサーバ・アプリケーションを作成する最初の手順は、OMG IDL を使用してすべての CORBA インターフェイスとそのメソッドを指定することです。Basic サンプル・アプリケーションでは、次の CORBA インターフェイスをインプリメントします。

インターフェイス	説明	オペレーション
`RegistrarFactory`	`Registrar` オブジェクトへのオブジェクト・リファレンスを作成します。	`find_registrar()`
`Registrar`	データベースからコースの情報を取得します。	`get_courses_synopsis()` `get_courses_details()`
`CourseSynopsisEnumerator`	コース・データベースから検索基準に一致するコースの概要を取得し、メモリに読み込みます。次に、概要の最初のサブセットを Registrar オブジェクトに返し、Registrar は概要をクライアント・アプリケーションに返します。最後に、クライアント・アプリケーションが残りの概要を取得できるようにします。	`get_next_n()` `destroy()`

リスト 3-1 では、Basic サンプル・アプリケーションの CORBA インターフェイスを定義した `univb.idl` ファイルを示します。このファイルのコピーは、Basic サンプル・アプリケーションのディレクトリ内にあります。

コード リスト 3-1 Basic **サンプル・アプリケーションの** OMG IDL

```
module UniversityB

{

   typedef unsigned long CourseNumber;
   typedef sequence<CourseNumber> CourseNumberList;

   struct CourseSynopsis
   {
        CourseNumber   course_number;
        string         title;
   };

   typedef sequence<CourseSynopsis> CourseSynopsisList;

   interface CourseSynopsisEnumerator
   {
        CourseSynopsisList get_next_n(
            in unsigned long number_to_get,
            out unsigned long number_remaining
        );
         void destroy();
 };

   typedef unsigned short Days;
   const Days MONDAY    =  1;
   const Days TUESDAY   =  2;
   const Days WEDNESDAY =  4;
   const Days THURSDAY  =  8;
   const Days FRIDAY    = 16;

   struct ClassSchedule
   {
       Days           class_days; // 日のビットマスク
       unsigned short start_hour; // 軍用時間による時間
       unsigned short duration;   // 分
   };

   struct CourseDetails
   {
       CourseNumber   course_number;
       double         cost;
       unsigned short number_of_credits;
       ClassSchedule  class_schedule;
       unsigned short number_of_seats;
       string         title;
```

```
        string          professor;
        string          description;
    };

    typedef sequence<CourseDetails> CourseDetailsList;

    interface Registrar
    {
        CourseSynopsisList
        get_courses_synopsis(
           in string                    search_criteria,
           in unsigned long             number_to_get, // 0 = すべて
           out unsigned long            number_remaining,
           out CourseSynopsisEnumerator rest
        );
    CourseDetailsList get_courses_details(in CourseNumberList
         courses);

    interface RegistrarFactory
    {
         Registrar find_registrar(
         );
    };
};
```

クライアント・スタブとスケルトンの生成

注記　University サンプル・アプリケーションの CORBA クライアント・アプリケーションでは、静的起動を使用しています。動的起動インターフェイスの使用例については、『BEA Tuxedo CORBA クライアント・アプリケーションの開発方法』を参照してください。CORBA Java クライアント・アプリケーションを作成する場合、クライアント・スタブを取得するための OMG IDL のコンパイルについては、Java ORB のマニュアルを参照してください。ActiveX クライアント・アプリケーションでは、クライアント・スタブは使用しません。

IDL コンパイラでは、OMG IDL で定義されたインターフェイス仕様を使用して、クライアント・アプリケーションのクライアント・スタブおよびサーバ・アプリケーションのスケルトンを生成します。クライアント・スタブは、すべてのオペレーション呼び出しでクライアント・アプリケーションによって使用されます。スケルトンは、CORBA オブジェクトをインプリメントするサーバ・アプリケーションを作成するために、記述したコードと共に使用します。クライアント・スタブとスケルトンの生成および使用方法については、『BEA Tuxedo CORBA アプリケーション入門』を参照してください。

開発プロセスでは、`idl` コマンドを使用して、OMG IDL ファイルをコンパイルしたり、クライアント・スタブおよびスケルトンを生成します。この作業は、Basic サンプル・アプリケーションの `makefile` で自動化されています。`idl` コマンドについては、『BEA Tuxedo コマンド・リファレンス』を参照してください。

クライアント・アプリケーションの記述

BEA Tuxedo の CORBA 環境では、次の 3 種類のクライアント・アプリケーションをサポートしています。

- CORBA C++
- CORBA Java
- ActiveX

注記　ActiveX クライアント・アプリケーションは Visual Basic で記述します。

開発プロセスでは、以下の処理を行うクライアント・アプリケーション・コードを記述します。

- ORB を初期化する。
- Bootstrap 環境オブジェクトまたは標準 CORBA メカニズムを使用して、BEA Tuxedo ドメインとの通信を確立する。

- FactoryFinder 環境オブジェクトへの初期リファレンスを解決する。
- ファクトリを使用して Registrar オブジェクトのオブジェクト・リファレンスを取得する。
- Registrar オブジェクトの get_courses_synopsis() および get_courses_details() メソッドを呼び出す。

Basic サンプル・アプリケーションには、C++、Java、および Visual Basic による各クライアント・アプリケーションが用意されています。クライアント・アプリケーションの記述方法については、『BEA Tuxedo CORBA アプリケーション入門』および『BEA Tuxedo CORBA クライアント・アプリケーションの開発方法』を参照してください。

サーバ・アプリケーションの記述

開発プロセスでは、次の内容を記述します。

- University サンプル・アプリケーションを初期化し、Registrar オブジェクトを BEA Tuxedo ドメインに登録する Server オブジェクト
- Registrar、RegistrarFactory、および CourseSynopsisEnumerator オブジェクトのオペレーションのメソッド・インプリメンテーション

サンプル・アプリケーションには、Server オブジェクトの C++ コード、および University サーバ・アプリケーションのメソッド・インプリメンテーションが用意されています。

開発プロセスでは、genicf コマンドを使用してインプリメンテーション・コンフィギュレーション・ファイル (ICF) を作成します。その際、ICF ファイルを編集して Registrar、RegistrarFactory、および CourseSynopsisEnumerator オブジェクトの活性化方針とトランザクション方針を定義します。Basic サンプル・アプリケーションの場合、Registrar、RegistrarFactory、および CourseSynopsisEnumerator オブジェクトの活性化方針は process、トランザクション方針は ignore です。Basic サンプル・アプリケーション用の ICF ファイルが用意されています。

サーバ・アプリケーションの記述方法については、『BEA Tuxedo CORBA サーバ・アプリケーションの開発方法』を参照してください。

Basic サンプル・アプリケーションのコンフィギュレーション

どの CORBA アプリケーションでも、重要な役割を果たすのが UBBCONFIG ファイルです。管理者が UBBCONFIG ファイルの作成を担当する場合でも、プログラマはこのファイルの存在と使用方法を把握しておくことが重要です。システム管理者がコンフィギュレーション・ファイルを作成するときは、一連のパラメータを使用して CORBA アプリケーションを記述します。これらのパラメータが BEA Tuxedo ソフトウェアによって解釈されると、実行可能なアプリケーションが生成されます。

コンフィギュレーション・ファイルには次の 2 種類があります。

- UBBCONFIG ファイル。ASCII 形式なので、任意のエディタで作成および変更できます。すべての University サンプル・アプリケーションで使用される UBBCONFIG ファイルの必須パラメータの設定については、「環境設定」を参照してください。
- TUXCONFIG ファイル。バイナリ形式の UBBCONFIG ファイルで、tmloadcf コマンドを使用して作成します。tmloadcf コマンドを実行する際は、TUXCONFIG 環境変数を TUXCONFIG ファイルの名前とディレクトリの場所に設定する必要があります。

UBBCONFIG ファイルおよび tmloadcf コマンドについては、『BEA Tuxedo アプリケーションの設定』および『BEA Tuxedo コマンド・リファレンス』を参照してください。

Basic サンプル・アプリケーションのビルド

Basic サンプル・アプリケーションをビルドするには、以下の手順に従います。

1. Basic サンプル・アプリケーションのファイルを作業ディレクトリにコピーします。
2. Basic サンプル・アプリケーションのファイルの保護を変更します。
3. 環境変数を設定します。
4. University データベースを初期化します。
5. `UBBCONFIG` ファイルをロードします。
6. クライアントおよびサーバ・サンプル・アプリケーションをビルドします。

以降の節では、上記の各手順について説明します。

注記　Basic サンプル・アプリケーションをビルドまたは実行する前に、「環境設定」の手順を実行しておく必要があります。

Basic サンプル・アプリケーションのファイルを作業ディレクトリにコピーする

Basic サンプル・アプリケーションの各ファイルは、次のディレクトリにあります。

Windows 2000

drive`:\TUXDIR\samples\corba\university\basic`

UNIX

```
/usr/TUXDIR/samples/corba/university/basic
```

また、utils ディレクトリも作業ディレクトリにコピーする必要があります。utils ディレクトリには、ログ、トレース、および University データベースへのアクセスを設定するファイルが格納されています。

表 3-1 では、Basic サンプル・アプリケーションの作成に使用するファイルについて説明します。

表 3-1 Basic サンプル・アプリケーションに含まれるファイル

ファイル	説明
univb.idl	CourseSynopsisEnumerator、Registrar、および RegistrarFactory インターフェイスを宣言する OMG IDL。
univbs.cpp	Basic サンプル・アプリケーションの University サーバ・アプリケーション用 C++ ソース・コード。
univb_i.h univb_i.cpp	CourseSynopsisEnumerator、Registrar、および RegistrarFactory インターフェイスのメソッド・インプリメンテーション用 C++ ソース・コード。
univbc.cpp	Basic サンプル・アプリケーションの CORBA C++ クライアント・アプリケーション用 C++ ソース・コード。
frmBrowser.frm frmBrowser.frx	Basic サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic ソース・コード。
modPublicDeclarations.bas	サンプル・アプリケーションで使用される変数の宣言が記述された Visual Basic ファイル。
frmTracing.frm frmTracing.frx	ActiveX クライアント・アプリケーションにトレース機能を提供するファイル。
University.vbp	Basic サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic プロジェクト・ファイル。

表 3-1 Basic サンプル・アプリケーションに含まれるファイル (続き)

ファイル	説明
University.vbw	Basic サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic ワークスペース・ファイル。
UnivBApplet.java	Basic サンプル・アプリケーションの CORBA Java クライアント・アプリケーション用 Java ソース・コード。
univb_utils.h univb_utils.cpp	CORBA C++ クライアント・アプリケーションのデータベース・アクセス関数を定義するファイル。
univb.icf	Basic サンプル・アプリケーションのインプリメンテーション・コンフィギュレーション・ファイル (ICF)。
setenvb.sh	Basic サンプル・アプリケーションのビルドおよび実行に必要な環境変数を設定する UNIX スクリプト。
setenvb.cmd	Basic サンプル・アプリケーションのビルドおよび実行に必要な環境変数を設定する MS-DOS コマンド。
ubb_b.mk	UNIX オペレーティング・システム用のコンフィギュレーション・ファイル。
ubb_b.nt	Windows 2000 オペレーティング・システム用のコンフィギュレーション・ファイル。
makefileb.mk	UNIX オペレーティング・システムでの Basic サンプル・アプリケーション用の makefile。
makefileb.nt	Windows 2000 オペレーティング・システムでの Basic サンプル・アプリケーション用の makefile。
log.cpp、log.h、 log_client.cpp、 log_server.cpp	サンプル・アプリケーションにログ機能とトレース機能を提供するクライアント・アプリケーションとサーバ・アプリケーション。これらのファイルは、\utils ディレクトリにあります。

表 3-1 Basic サンプル・アプリケーションに含まれるファイル (続き)

ファイル	説明
oradbconn.cpp、 oranoconn.cpp	Oracle SQL データベース・インスタンスへのアクセスを提供するファイル。これらのファイルは、\utils ディレクトリにあります。
samplesdb.cpp、 samplesdb.h	サンプル・アプリケーションでのデータベース例外に出力関数を提供するファイル。これらのファイルは、\utils ディレクトリにあります。
unique_id.cpp、 unique_id.h	サンプル・アプリケーションの C++ Unique ID クラスのルーチン。これらのファイルは、\utils ディレクトリにあります。
samplesdbsql.h、 samplesdbsql.pc	SQL データベースへのアクセスをインプリメントする C++ クラスのメソッド。これらのファイルは、\utils ディレクトリにあります。
university.sql	University データベース用の SQL。このファイルは、\utils ディレクトリにあります。

Basic サンプル・アプリケーションのファイル保護の属性を変更する

BEA Tuxedo ソフトウェアのインストール時には、サンプル・アプリケーションは読み取り専用に設定されています。Basic サンプル・アプリケーションのファイルを編集または作成するには、次のように作業ディレクトリにコピーしたファイルの保護を変更する必要があります。

Windows 2000

```
prompt>attrib -r drive:\workdirectory\*.*
```

UNIX

```
prompt>chmod u+rw /workdirectory/*.*
```

環境変数を設定する

次のコマンドを使用して、Basic サンプル・アプリケーションのクライアント・アプリケーションとサーバ・アプリケーションのビルドに使用する環境変数を設定します。

Windows 2000

```
prompt>setenvb
```

UNIX

```
prompt>/bin/ksh
prompt>. ./setenvb.sh
```

University データベースを初期化する

次のコマンドを使用して、Basic サンプル・アプリケーションで使用する University データベースを初期化します。

Windows 2000

```
prompt>nmake -f makefileb.nt initdb
```

UNIX

```
prompt>make -f makefileb.mk initdb
```

UBBCONFIG ファイルをロードする

次のコマンドを使用して、UBBCONFIG ファイルをロードします。

Windows 2000

```
prompt>tmloadcf -y ubb_b.nt
```

UNIX

```
prompt>tmloadcf -y ubb_b.mk
```

Basic サンプル・アプリケーションのコンパイル

開発プロセスでは、buildobjclient および buildobjserver コマンドを使用して、クライアント・アプリケーションとサーバ・アプリケーションをビルドします。ただし、Basic サンプル・アプリケーションの場合は、この手順は不要です。

Basic サンプル・アプリケーションのディレクトリには、makefile が格納されています。この makefile により、クライアントとサーバ・サンプル・アプリケーションがビルドされます。

Basic サンプル・アプリケーションの CORBA C++ クライアント・アプリケーションとサーバ・アプリケーションをビルドするには、次のコマンドを使用します。

Windows 2000

```
prompt>nmake -f makefileb.nt
```

UNIX

```
prompt>make -f makefileb.mk
```

CORBA Java クライアント・アプリケーションをビルドするには、次のコマンドを使用します。

Windows 2000

```
prompt>nmake -f makefileb.nt javaclient
```

UNIX

```
prompt>make -f makefileb.mk javaclient
```

ActiveX クライアント・アプリケーションのビルドおよび使用方法については、「ActiveX クライアント・アプリケーションの起動」を参照してください。

buildobjclient および buildobjserver コマンドの詳細については、『BEA Tuxedo コマンド・リファレンス』を参照してください。

Basic サンプル・アプリケーションの実行

Basic サンプル・アプリケーションを実行するには、次の手順に従います。

1. University サーバ・アプリケーションを起動します。
2. 1 つまたは複数のクライアント・アプリケーションを起動します。

サーバ・アプリケーションの起動

Basic サンプル・アプリケーションでシステムおよびサンプル・アプリケーションのサーバ・アプリケーションを起動するには、次のコマンドを入力します。

```
prompt>tmboot -y
```

このコマンドを入力すると、次のサーバ・プロセスが開始されます。

- TMSYSEVT

  BEA Tuxedo システムの EventBroker。

- TMFFNAME

  NameManager サービスや FactoryFinder サービスなどのトランザクション管理サービス。

- TMIFSRVR

  インターフェイス・リポジトリ・サーバ・プロセス。これは、ActiveX クライアント・アプリケーションでのみ使用されます。

- univb_server

University サーバ・プロセス。

- ISL

  IIOP リスナ / ハンドラ・プロセス。

ほかのサンプル・アプリケーションを使用するには、次のコマンドを入力して、システムおよびサンプル・アプリケーションのサーバ・プロセスを停止します。

```
prompt>tmshutdown
```

CORBA C++ クライアント・アプリケーションの起動

Basic サンプル・アプリケーションの CORBA C++ クライアント・アプリケーションを起動するには、次のコマンドを入力します。

```
prompt>univb_client
```

CORBA Java クライアント・アプリケーションの起動

Basic サンプル・アプリケーションの CORBA Java クライアント・アプリケーションを実行するには、以下の手順に従います。

1. UnivBApplet.html ファイル内の次の行を変更します。

   ```
   code="UnivBApplet.class"
   codebase=.
   ```

   上記の行を次のように変更します。

   ```
   code="UnivBApplet"
   archive="UnivBApplet.jar,m3envobj.jar"
   ```

2. 変更した UnivBApplet.html ファイルを Web サーバのソース・ディレクトリにコピーします。ディレクトリは、Web サーバ製品によって異なります。

3. makefile を実行して Basic サンプル・アプリケーションをビルドした後、次のように UnivBApplet.jar ファイルを作成します。

   a. サンプル・アプリケーションをビルドしたディレクトリの下に tmp ディレクトリを作成します。UniversityB サブディレクトリとその中に格納されているクラス・ファイルを tmp ディレクトリにコピーします。

      makefile によって生成された、Basic サンプル・アプリケーション・ディレクトリにあるクラス・ファイルを tmp ディレクトリにコピーします。ディレクトリを tmp ディレクトリに設定 (cd) し、次のいずれかのコマンドを入力して、すべての Basic サンプル・アプリケーション・クラスを含んだ .JAR ファイルを作成します。

      ```
      jar -cf ..\UnivBApplet.jar *.* (Microsoft Windows 2000 システム )
      jar -cf ../UnivBApplet.jar * (UNIX システム )
      ```

4. 前の手順で作成した UnivBApplet.jar ファイルを Web サーバのソース・ディレクトリにコピーします。ディレクトリ名は、Web サーバ製品によって異なります。

5. 適切なサブディレクトリ (Microsoft Windows 2000 システムの場合は %TUXDIR%\udataobj\java 、UNIX システムの場合は ${TUXDIR}/udataobj/java) から、m3envobj.jar ファイルを Web サーバのソース・ディレクトリにコピーします。

6. Basic サーバ・アプリケーションが動作していることを確認してから、Web ブラウザを起動し、Web サーバが動作しているノードを参照します。

注記　Microsoft Windows 2000 システムの場合、ノード名はすべて大文字にする必要があります。たとえば、UBBCONFIG ファイルおよび UnivBApplet.html ファイルでノードを SERVER に指定した場合、ブラウザは http://SERVER/UnivBApplet.html に設定します。

ActiveX クライアント・アプリケーションの起動

注記　University サンプル・アプリケーションでは、インターフェイス・リポジトリに CORBA インターフェイスの OMG IDL をロードする作業は `makefile` によって自動化されています。

ActiveX クライアント・アプリケーションを起動するには、Application Builder を使用して CORBA インターフェイスの ActiveX バインディングを作成する必要があります。

CORBA インターフェイスの ActiveX バインディングを作成するには、次の手順に従います。

1. BEA Tuxedo プログラム・グループで [BEA Application Builder] アイコンをクリックします。

   [Domain] ログオン・ウィンドウが表示されます。

2. [Domain] ログオン・ウィンドウに、`UBBCONFIG` ファイルの `ISL` パラメータで指定したホスト名とポート番号を入力します。ホスト名とポート番号は、`UBBCONFIG` ファイルで大文字で指定した内容と正確に一致するように入力します。たとえば、次のように入力します。`//BEANIE:2500`.

   Application Builder のログオン・ウィンドウが表示されます。

3. [Services] ウィンドウで UniversityB フォルダを強調表示して [Workstation Views] ウィンドウにドラッグします。または、[Services] ウィンドウから UniversityB フォルダをコピーして [Workstation Views] ウィンドウに貼り付けます。

   確認のウィンドウが表示されます。

4. [Create] をクリックすると、Basic サンプル・アプリケーションで CORBA インターフェイスの ActiveX バインディングが作成されます。

   Application Builder では、以下のものが作成されます。

   - CORBA インターフェイスに対するバインディング。バインディングの名前の形式は、`DI`*`modulename_interfacename`* です。たとえば、`Registrar` インターフェイスに対するバインディングの名前は、`DIUniversityB_Registrar` となります。

- 型ライブラリ。デフォルトでは、型ライブラリは \TUXDIR\TypeLibraries に格納されています。

  型ライブラリのファイル名の形式は、DI*modulename_interfacename*.tlb です。

- CORBA インターフェイスの Windows システム・レジストリ・エントリ。このエントリには、オブジェクト型ごとに一意のプログラム ID が入っています。

ActiveX クライアント・アプリケーションを開くには、次の手順に従います。

1. Visual Basic で University プロジェクトを開きます。
2. University プロジェクトを実行します。
3. [Run] メニューの [Start] をクリックします。

   ログオン・ウィンドウが表示されます。
4. [Logon] ウィンドウに、UBBCONFIG ファイルの ISL パラメータで指定したホスト名とポート番号を入力します。ホスト名とポート番号は、UBBCONFIG ファイルで大文字で指定した内容と正確に一致するように入力します。

Basic サンプル・アプリケーションのクライアント・アプリケーションの使用方法

以降の節では、Basic サンプル・アプリケーションに含まれているクライアント・アプリケーションについて簡単に説明します。

CORBA C++ クライアント・アプリケーション

CORBA C++ クライアント・アプリケーションを起動すると、メニューが表示されます。メニューには、次のオプションがあります。

```
<F>   Find courses

<A>   List all courses

<D>   Display course details

<E>   Exit
```

特定のカリキュラムのサブジェクトに一致するコースを検索するには、次の手順に従います。

1. Options プロンプトで「F」を入力します。
2. 画面に「Enter search string:」と表示されたら、「computer」などのテキスト文字列を入力します。入力する文字列は、大文字と小文字を任意で組み合わせてもかまいません。

   検索文字列に一致したコースがすべて一覧表示されます。

データベース内のすべてのコースを一覧表示するには、次の手順に従います。

1. Options プロンプトで「A」を入力します。

   10 コースの一覧が表示されます。
2.「y」を入力すると、次の 10 コースの一覧が表示されます。「n」を入力すると、Options メニューに戻ります。

特定のコースの詳細を表示するには、次の手順に従います。

1. Options プロンプトで「D」を入力します。
2. 画面に「Course Number」と表示されたら、コース番号とその後に続けて「-1」を入力します。たとえば、次のように入力します。

   ```
   100011
   100039
   -1
   ```

   指定したコースの要約が表示されます。

C++ CORBA クライアント・アプリケーションを終了するには、Options プロンプトで「E」を入力します。

CORBA Java クライアント・アプリケーション

特定のカリキュラムのサブジェクトに一致するコースを検索するには、次の手順に従います。

1. 画面に「search string?」と表示されたら、テキスト・ボックスにテキスト文字列を入力します。テキスト・ボックスには、コースのタイトル、教授の名前、またはコースの説明を入力できます。たとえば、「computer」などの文字列を入力します。
2. [Show] ボタンをクリックします。

   検索文字列に一致したコースがすべて一覧表示されます。

データベース内のすべてのコースを一覧表示するには、次の手順に従います。

1. [Course Name Search String] テキスト・ボックスの中にカーソルを置きます。
2. Enter キーを押します。

   コース・データベース内のすべてのコースが一覧表示されます。

特定のコースの詳細を表示するには、次の手順に従います。

1. [Course Number/Course Name] ウィンドウでコースを選択します。
2. [Details] ボタンをクリックします。

   選択したコースの詳細の要約が表示されます。

CORBA Java クライアント・アプリケーションを終了するには、[Applet] メニューから [Quit] を選択します。

ActiveX クライアント・アプリケーション

ActiveX クライアント・アプリケーションにログオンすると、[Course Browser] ウィンドウが表示されます。University で利用可能なコースの検索は、[Course Browser] ウィンドウから行います。

特定のカリキュラムのサブジェクトに一致するコースを検索するには、次の手順に従います。

1. [Find Courses] ボタンの横にあるテキスト・ボックスにテキスト文字列を入力するか、またはプル・ダウン・メニューからカリキュラムのサブジェクトを選択します。たとえば、「`computer`」などの文字列を入力します。
2. [Find Courses] ボタンをクリックします。

   検索文字列に一致したコースがすべて一覧表示されます。

特定のコースの詳細を表示するには、次の手順に従います。

1. [Get Details] ボタンの横にあるウィンドウでコースを選択します。
2. [Get Details] ボタンをクリックするか、またはコース名をダブルクリックします。

   選択したコースの詳細の要約が表示されます。
3. スケジュールにコースを入力するには、コース名をダブルクリックします。

ActiveX クライアント・アプリケーションを終了するには、[File] メニューから [Exit] を選択します。

4 Security サンプル・アプリケーション

ここでは、以下の内容について説明します。

- Security サンプル・アプリケーションのしくみ
- Security サンプル・アプリケーションの開発プロセス
- Security サンプル・アプリケーションのビルド
- Security サンプル・アプリケーションのコンパイル
- Security サンプル・アプリケーションの実行
- Security サンプル・アプリケーションのクライアント・アプリケーションの使用方法

トラブルシューティング情報については、`\security` ディレクトリにある `Readme.txt` を参照してください。また、Security サンプル・アプリケーションの使用方法に関する最新の情報も参照してください。CORBA アプリケーションでのセキュリティのインプリメントに関する詳細については、『BEA Tuxedo CORBA アプリケーションのセキュリティ機能』を参照してください。

Security サンプル・アプリケーションのしくみ

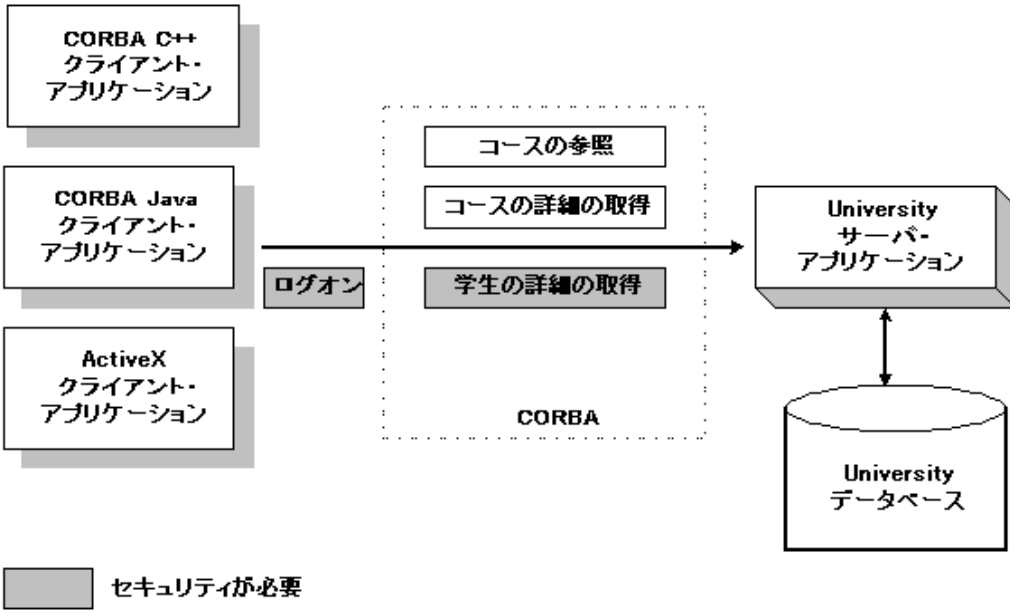
Security サンプル・アプリケーションでは、アプリケーション・レベルのセキュリティを CORBA アプリケーションに追加することにより、Basic サンプル・アプリケーションを拡張します。アプリケーション・レベルのセキュリティでは、学生ごとに ID とパスワードを付与する必要があります。そのため、Security サンプル・アプリケーションには Student という概念が追加されます。

以下の機能が Basic サンプル・アプリケーションに追加されます。

- クライアント・アプリケーションによって、ログオン用のオペレーションが追加されます。このオペレーションでは、SecurityCurrent 環境オブジェクトを使用して PrincipalAuthenticator オブジェクトのオペレーションを呼び出します。PrincipalAuthenticator オブジェクトは、ドメインにアクセスするためのログオン・プロセスの一部です。
- University サーバ・アプリケーションによって、追加オペレーションの `get_student_details()` が `Registrar` オブジェクトにインプリメントされ、学生に関する情報が返されます。CORBA ログオンが正常に完了した後、`get_student_details()` オペレーションはデータベース内の学生情報にアクセスして、クライアントのログオン・オペレーションに必要な学生情報を取得します。
- University データベースには、コースの情報とともに学生情報が格納されています。

図 4-1 では、Security サンプル・アプリケーションのしくみを示しています。

図 4-1 Security サンプル・アプリケーション



Security サンプル・アプリケーションの開発プロセス

ここでは、CORBA クライアント・アプリケーションとサーバ・アプリケーションにセキュリティを追加する際に必要な開発プロセスについて説明します。以下の手順は、第 3 章 「Basic サンプル・アプリケーション」で概説した開発手順の追加手順です。

注記　この節に記載されている手順はすでに完了しており、Security サンプル・アプリケーションに組み込まれています。

OMG IDL

開発プロセスでは、Object Management Group (OMG) インターフェイス定義言語 (IDL) で `StudentDetails` 構造体および `get_student_details()` オペレーションを定義します。

クライアント・アプリケーション

開発プロセスでは、クライアント・アプリケーションに次のコードを追加します。

- 指定した BEA Tuxedo ドメインの SecurityCurrent 環境オブジェクトへのリファレンスを取得するための Bootstrap 環境オブジェクト
- BEA Tuxedo ドメインで必要な認証の型を返すための、SecurityCurrent 環境オブジェクトの `Tobj::PrincipalAuthenticator` オペレーション
- 必要なセキュリティ情報を使用して BEA Tuxedo ドメインにログオンするためのオペレーション

Security サンプル・アプリケーションの場合、上記のコードはすでに追加されています。CORBA クライアント・アプリケーションへのセキュリティの追加については、『BEA Tuxedo CORBA アプリケーションのセキュリティ機能』を参照してください。

サーバ・アプリケーション

開発プロセスでは、`get_student_details()` オペレーションのメソッド・インプリメンテーションを記述します。メソッド・インプリメンテーションの記述方法については、『BEA Tuxedo CORBA サーバ・アプリケーションの開発方法』を参照してください。

UBBCONFIG ファイル

BEA Tuxedo ソフトウェアでは、システム管理者がコンフィギュレーションのセキュリティ・レベルを定義します。BEA Tuxedo ドメインのセキュリティを定義する際は、UBBCONFIG ファイルの SECURITY パラメータの RESOURSES セクションを目的のセキュリティ・レベルに設定します。Security サンプル・アプリケーションの場合、SECURITY パラメータをアプリケーション・レベルのセキュリティの APP_PW に設定します。BEA Tuxedo ドメインへのセキュリティの追加については、『BEA Tuxedo アプリケーションの設定』および『BEA Tuxedo CORBA アプリケーションのセキュリティ機能』を参照してください。

ICF ファイル

インプリメンテーション・コンフィギュレーション・ファイル (ICF) は変更不要です。

Security サンプル・アプリケーションのビルド

Security サンプル・アプリケーションをビルドするには、次の手順に従います。

1. Security サンプル・アプリケーションのファイルをコピーします。
2. Security サンプル・アプリケーションのファイル保護の属性を変更します。
3. 環境変数を設定します。
4. University データベースを初期化します。

5. UBBCONFIG ファイルをロードします。
6. クライアントおよびサーバ・サンプル・アプリケーションをビルドします。

以降の節では、上記の各手順について説明します。

注記　Security サンプル・アプリケーションをビルドまたは実行する前に、第 2 章 「環境設定」の手順を実行しておく必要があります。

Security サンプル・アプリケーションのファイルを作業ディレクトリにコピーする

Security サンプル・アプリケーションの各ファイルは、次のディレクトリにあります。

Windows 2000

drive:*TUXDIR*\samples\corba\university\security

UNIX

/usr/*TUXDIR*/samples/corba/university/security

また、utils ディレクトリも作業ディレクトリにコピーする必要があります。utils ディレクトリには、ログ、トレース、および University データベースへのアクセスを設定するファイルが格納されています。

表 4-1 に、Security サンプル・アプリケーションの作成に使用するファイルの一覧を示します。

表 4-1 Security サンプル・アプリケーションに含まれるファイル

ファイル	説明
univs.idl	CourseSynopsisEnumerator、Registrar、および RegistrarFactory インターフェイスを宣言する OMG IDL。

表 4-1 Security サンプル・アプリケーションに含まれるファイル (続き)

ファイル	説明
univss.cpp	Security サンプル・アプリケーションの University サーバ・アプリケーション用 C++ ソース・コード。
univs_i.h univs_i.cpp	CourseSynopsisEnumerator、Registrar、および RegistrarFactory インターフェイスのメソッド・インプリメンテーション用 C++ ソース・コード。
univsc.cpp	Security サンプル・アプリケーションの CORBA C++ クライアント・アプリケーション用 C++ ソース・コード。
frmBrowser.frm	Security サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic ソース・コード。
frmOpen.frm	Security サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic ソース・コード。
University.vbp	Security サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic プロジェクト・ファイル。
University.vbw	Security サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic ワークスペース・ファイル。
modPublicDeclarations.bas	サンプル・アプリケーションで使用される変数の宣言が記述された Visual Basic ファイル。
frmTracing.frm frmTracing.frx	ActiveX クライアント・アプリケーションにトレース機能を提供するファイル。
frmLogon.frm	ActiveX クライアント・アプリケーションに対してセキュリティ・ログオンを実行する Visual Basic ファイル。

表 4-1 Security サンプル・アプリケーションに含まれるファイル (続き)

ファイル	説明
UnivSApplet.java	Security サンプル・アプリケーションの CORBA Java クライアント・アプリケーション用 Java ソース・コード。
univs_utils.h univs_utils.cpp	CORBA C++ クライアント・アプリケーションのデータベース・アクセス関数を定義するファイル。
univs.icf	Security サンプル・アプリケーションのインプリメンテーション・コンフィギュレーション・ファイル (ICF)。
setenvs.sh	Security サンプル・アプリケーションのビルドおよび実行に必要な環境変数を設定する UNIX スクリプト。
setenvs.cmd	Security サンプル・アプリケーションのビルドおよび実行に必要な環境変数を設定する MS-DOS コマンド。
ubb_s.mk	UNIX オペレーティング・システム用の UBBCONFIG ファイル。
ubb_s.nt	Windows 2000 オペレーティング・システム用の UBBCONFIG ファイル。
makefiles.mk	UNIX オペレーティング・システムでの Security サンプル・アプリケーション用の makefile。
makefiles.nt	Windows 2000 オペレーティング・システムでの Security サンプル・アプリケーション用の makefile。
log.cpp、log.h、 log_client.cpp、 log_server.cpp	サンプル・アプリケーションにログ機能とトレース機能を提供するクライアント・アプリケーションとサーバ・アプリケーション。これらのファイルは、\utils ディレクトリにあります。

表 4-1 Security サンプル・アプリケーションに含まれるファイル (続き)

ファイル	説明
oradbconn.cpp、oranoconn.cpp	Oracle SQL データベース・インスタンスへのアクセスを提供するファイル。これらのファイルは、\utils ディレクトリにあります。
samplesdb.cpp、samplesdb.h	サンプル・アプリケーションでのデータベース例外に出力関数を提供するファイル。これらのファイルは、\utils ディレクトリにあります。
unique_id.cpp、unique_id.h	サンプル・アプリケーションの C++ Unique ID クラスのルーチン。これらのファイルは、\utils ディレクトリにあります。
samplesdbsql.h、samplesdbsql.pc	SQL データベースへのアクセスをインプリメントする C++ クラスのメソッド。これらのファイルは、\utils ディレクトリにあります。
university.sql	University データベース用の SQL。このファイルは、\utils ディレクトリにあります。

Security サンプル・アプリケーションのファイル保護の属性を変更する

BEA Tuxedo ソフトウェアのインストール時には、サンプル・アプリケーションは読み取り専用に設定されています。Security サンプル・アプリケーションのファイルを編集またはビルドするには、次のように作業ディレクトリにコピーしたファイル保護の属性を変更する必要があります。

Windows 2000

```
prompt>attrib -r drive:\workdirectory\*.*
```

UNIX

```
prompt>chmod u+rw /workdirectory/*.*
```

環境変数を設定する

次のコマンドを使用して、Security サンプル・アプリケーションのクライアント・アプリケーションとサーバ・アプリケーションのビルドに使用する環境変数を設定します。

Windows 2000

```
prompt>setenvs
```

UNIX

```
prompt>/bin/ksh
prompt>. ./setenvs.sh
```

University データベースを初期化する

次のコマンドを使用して、Security サンプル・アプリケーションで使用する University データベースを初期化します。

Windows 2000

```
prompt>nmake -f makefiles.nt initdb
```

UNIX

```
prompt>make -f makefiles.mk initdb
```

UBBCONFIG ファイルをロードする

次のコマンドを使用して、UBBCONFIG ファイルをロードします。

Windows 2000

```
prompt>tmloadcf -y ubb_s.nt
```

UNIX

```
prompt>tmloadcf -y ubb_s.mk
```

UBBCONFIG ファイルの作成プロセスでは、アプリケーション・パスワードの入力が求められます。このパスワードは、クライアント・アプリケーションへのログオンに使用されます。パスワードを入力して Enter キーを押します。その際、パスワードを再入力してパスワードの確認を求めるメッセージが表示されます。

Security サンプル・アプリケーションのコンパイル

開発プロセスでは、buildobjclient および buildobjserver コマンドを使用して、クライアント・アプリケーションとサーバ・アプリケーションをビルドします。ただし、Security サンプル・アプリケーションの場合は、この手順は不要です。

Security サンプル・アプリケーションのディレクトリには、makefile が格納されています。この makefile により、クライアントとサーバ・サンプル・アプリケーションがビルドされます。

Security サンプル・アプリケーションの CORBA C++ クライアント・アプリケーションとサーバ・アプリケーションをビルドするには、次のコマンドを使用します。

Windows 2000

```
prompt>nmake -f makefiles.nt
```

UNIX

```
prompt>make -f makefiles.mk
```

CORBA Java クライアント・アプリケーションをビルドするには、次のコマンドを使用します。

Windows 2000

```
prompt>nmake -f makefiles.nt javaclient
```

UNIX

```
prompt>make -f makefiles.mk javaclient
```

ActiveX クライアント・アプリケーションの起動については、「ActiveX クライアント・アプリケーションの起動」を参照してください。

`buildobjclient` および `buildobjserver` コマンドの詳細については、『BEA Tuxedo コマンド・リファレンス』を参照してください。

Security サンプル・アプリケーションの実行

Security サンプル・アプリケーションを実行するには、次の手順に従います。

1. University サーバ・アプリケーションを起動します。
2. 1 つまたは複数のクライアント・アプリケーションを起動します。

上記の手順については、以降の節を参照してください。

University サーバ・アプリケーションの起動

Security サンプル・アプリケーションでシステムおよびサンプル・アプリケーションのサーバ・アプリケーションを起動するには、次のコマンドを入力します。

```
prompt>tmboot -y
```

このコマンドを入力すると、次のサーバ・プロセスが開始されます。

- `TMSYSEVT`

  BEA Tuxedo システムの EventBroker。

- `TMFFNAME`

  NameManager サービスや FactoryFinder サービスなどのトランザクション管理サービス。

- TMIFSRVR

  インターフェイス・リポジトリ・サーバ・プロセス。これは、ActiveX クライアント・アプリケーションでのみ使用されます。

- univs_server

  University サーバ・プロセス。

- ISL

  IIOP リスナ / ハンドラ・プロセス。

ほかのサンプル・アプリケーションを使用するには、次のコマンドを入力して、システムおよびサンプル・アプリケーションのサーバ・プロセスを停止します。

```
prompt>tmshutdown
```

CORBA C++ クライアント・アプリケーションの起動

Security サンプル・アプリケーションの CORBA C++ クライアント・アプリケーションを起動するには、次の手順に従います。

1. MS-DOS プロンプトで次のコマンドを入力します。

   ```
   prompt>univs_client
   ```

2. 画面に「Enter student id:」と表示されたら、100001 ～ 100010 の任意の数値を入力します。
3. Enter キーを押します。
4. 画面に「Enter domain password:」と表示されたら、UBBCONFIG ファイルをロードしたときに定義したパスワードを入力します。
5. Enter キーを押します。

CORBA Java クライアント・アプリケーションの起動

Security サンプル・アプリケーションの CORBA Java クライアント・アプリケーションを起動するには、以下の手順に従います。

1. UnivSApplet.html ファイル内の次の行を変更します。

```
code="UnivSApplet.class"
codebase=.
```

上記の行を次のように変更します。

```
code="UnivSApplet"
archive="UnivSApplet.jar,m3envobj.jar"
```

2. 変更した UnivSApplet.html ファイルを Web サーバのソース・ディレクトリにコピーします。ディレクトリは、Web サーバ製品によって異なります。

3. makefile を実行して Security サンプル・アプリケーションをビルドした後、次のように UnivSApplet.jar ファイルを作成します。

   a. サンプル・アプリケーションをビルドしたディレクトリの下に tmp ディレクトリを作成します。UniversityS サブディレクトリとその中に格納されているクラス・ファイルを tmp ディレクトリにコピーします。

      makefile によって生成された、Security サンプル・アプリケーション・ディレクトリにあるクラス・ファイルを tmp ディレクトリにコピーします。ディレクトリを tmp ディレクトリに設定 (cd) し、次のいずれかのコマンドを入力して、すべての Security サンプル・アプリケーション・クラスを含んだ .JAR ファイルを作成します。

      jar -cf ..\UnivSApplet.jar *.* (Microsoft Windows 2000 システム)
      jar -cf ../UnivSApplet.jar * (UNIX システム)

4. 前の手順で作成した UnivSApplet.jar ファイルを Web サーバのソース・ディレクトリにコピーします。ディレクトリ名は、Web サーバ製品によって異なります。

5. 適切なサブディレクトリ (Microsoft Windows 2000 システムの場合は `%TUXDIR%\udataobj\java`、UNIX システムの場合は `${TUXDIR}/udataobj/java`) から、`m3envobj.jar` ファイルを Web サーバのソース・ディレクトリにコピーします。
6. Security サーバ・アプリケーションが動作していることを確認してから、Web ブラウザを起動し、Web サーバが動作しているノードを参照します。

注記　Microsoft Windows 2000 システムの場合、ノード名はすべて大文字にする必要があります。たとえば、`UBBCONFIG` ファイルおよび `UnivSApplet.html` ファイルでノードを `SERVER` に指定した場合、ブラウザは `http://SERVER/UnivSApplet.html` に設定します。

7. student ID フィールドに 100001 ～ 100010 の数値を入力します。
8. `Domain Password` フィールドに、`UBBCONFIG` ファイルをロードしたときに定義したパスワードを入力します。
9. [Logon] ボタンをクリックします。
10. コースを検索するための検索文字列を入力します。

ActiveX クライアント・アプリケーションの起動

注記　University サンプル・アプリケーションでは、インターフェイス・リポジトリに CORBA インターフェイスの OMG IDL をロードする作業は `makefile` によって自動化されています。

ActiveX クライアント・アプリケーションを起動するには、Application Builder を使用して CORBA インターフェイスの ActiveX バインディングを作成する必要があります。

CORBA インターフェイスの ActiveX バインディングを作成するには、次の手順に従います。

1. BEA Tuxedo プログラム・グループで [BEA Application Builder] アイコンをクリックします。

[IIOP Listener] ウィンドウが表示されます。

2. [IIOP Listener] ウィンドウに、UBBCONFIG ファイルの ISL パラメータで指定したホスト名とポート番号を入力します。ホスト名とポート番号は、UBBCONFIG ファイルで大文字で指定した内容と正確に一致するように入力します。

   [Logon] ウィンドウが表示されます。

3. UBBCONFIG ファイルをロードしたときに定義したユーザ名とパスワードについて、100001 ～ 100010 の学生 ID を [Logon] ウィンドウに入力します。

   [Application Builder] ウィンドウが表示されます。インターフェイス・リポジトリにロードされた CORBA インターフェイスがすべて、Application Builder の [Services] ウィンドウに表示されます。

4. [Services] ウィンドウで UniversityS フォルダを強調表示して [Workstation Views] ウィンドウにドラッグします。または、[Services] ウィンドウから UniversityS フォルダをコピーして [Workstation Views] ウィンドウに貼り付けます。

   確認のウィンドウが表示されます。

5. [Create] をクリックすると、Security サンプル・アプリケーションで CORBA インターフェイスの ActiveX バインディングが作成されます。

   Application Builder では、以下のものが作成されます。

   - CORBA インターフェイスに対するバインディング。バインディングの名前の形式は、DI*modulename_interfacename* です。たとえば、Registrar インターフェイスに対するバインディングの名前は、DIUniversityS_Registrar となります。
   - 型ライブラリ。デフォルトでは、型ライブラリは \TUXDIR\TypeLibraries. に格納されています。

     型ライブラリのファイル名の形式は、DI*modulename_interfacename*.tlb です。
   - CORBA インターフェイスの Windows システム・レジストリ・エントリ。このエントリには、オブジェクト型ごとに一意のプログラム ID が入っています。

ActiveX クライアント・アプリケーションを実行するには、以下の手順に従います。

1. Visual Basic で `University.vbw` ファイルを開きます。
2. [Run] メニューの [Start] をクリックします。

   [IIOP Listener] ウィンドウが表示されます。
3. [IIOP Listener] ウィンドウに、`UBBCONFIG` ファイルの `ISL` パラメータで指定したホスト名とポート番号を入力します。ホスト名とポート番号は、`UBBCONFIG` ファイルで大文字で指定した内容と正確に一致するように入力します。

   [Logon] ウィンドウが表示されます。
4. `UBBCONFIG` ファイルをロードしたときに定義したユーザ名とパスワードについて、100001 ～ 100010 の学生 ID を [Logon] ウィンドウに入力します。

Security サンプル・アプリケーションのクライアント・アプリケーションの使用方法

以降の節では、Security サンプル・アプリケーションのクライアント・アプリケーションの使用方法について簡単に説明します。

CORBA C++ クライアント・アプリケーション

Security サンプル・アプリケーションの CORBA C++ クライアント・アプリケーションには、次の追加オプションがあります。

```
<L>    List your registered courses
```

このオプションを使用すると、CORBA C++ クライアント・アプリケーションへのログオン用の学生 ID で登録したコースが一覧表示されます。

CORBA Java クライアント・アプリケーション

Security サンプル・アプリケーションでは、CORBA Java クライアント・アプリケーションの追加機能はありません。

ActiveX クライアント・アプリケーション

Security サンプル・アプリケーションでは、ActiveX クライアント・アプリケーションの追加機能はありません。

5 Transactions サンプル・アプリケーション

ここでは、以下の内容について説明します。

- Transactions サンプル・アプリケーションのしくみ
- Transactions サンプル・アプリケーションの開発プロセス
- Transactions サンプル・アプリケーションのビルド
- Transactions サンプル・アプリケーションのコンパイル
- Transactions サンプル・アプリケーションの実行
- Transactions サンプル・アプリケーションのクライアント・アプリケーションの使用方法

トラブルシューティング情報については、`\transactions` ディレクトリにある `Readme.txt` を参照してください。また、Transactions サンプル・アプリケーションの使用方法に関する最新の情報も参照してください。CORBA アプリケーションでのトランザクションの使用に関する詳細については、『BEA Tuxedo CORBA トランザクション』を参照してください。

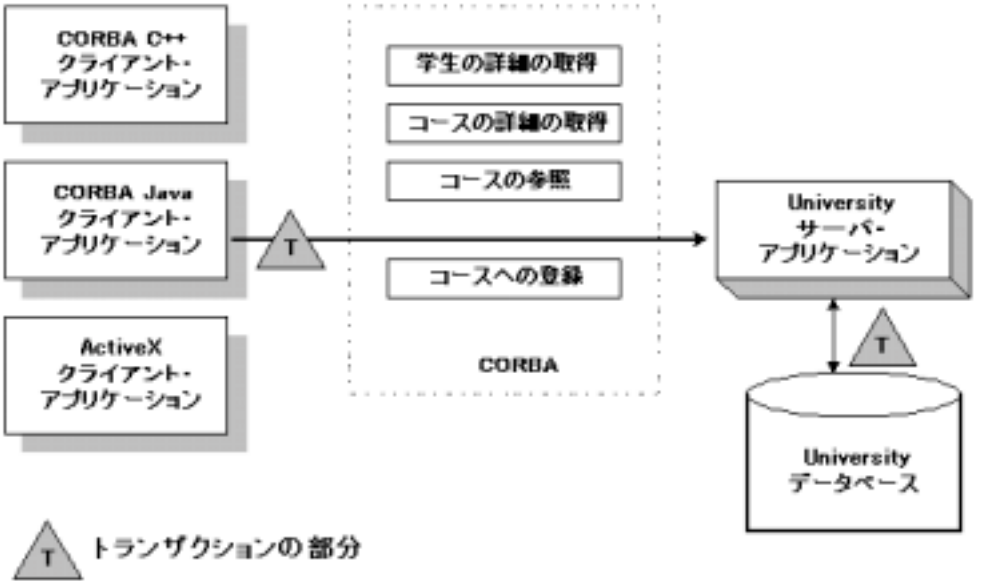
Transactions サンプル・アプリケーションのしくみ

Transactions サンプル・アプリケーションでは、学生はクラスの登録ができます。コースの登録は、トランザクションのスコープ内で実行されます。Transactions サンプル・アプリケーションは、以下のように機能します。

1. 学生は登録するコースのリストを送信します。
2. リスト内のコースごとに、University サーバ・アプリケーションは次の点をチェックします。
   - コースがデータベースにあるかどうか
   - 学生がコースに登録済みであるかどうか
   - 学生が履修できる単位の最大数を超えているかどうか
3. 次のいずれかの処理が行われます。
   - すべての基準をコースが満たしている場合、University サーバ・アプリケーションはコースに学生を登録します。
   - コースがデータベースにない場合、または学生がコースに登録済みの場合、University サーバ・アプリケーションは、その学生が未登録の登録コースのリストにコースを追加します。すべての登録要求の処理が完了したら、サーバ・アプリケーションは登録に失敗したコースのリストを返します。その際、クライアント・アプリケーションから学生に、トランザクションをコミットして登録要求が成功したコースに学生を登録するか、またはトランザクションをロールバックしてすべてのコースの登録を行わないように求めるメッセージが表示されます。
   - 学生が履修できる単位の最大数を超えている場合は、`TooManyCredits` ユーザ例外が University サーバ・アプリケーションからクライアント・アプリケーションに返されます。続いて、クライアント・アプリケーションによって、要求が拒否されたという内容の簡単なメッセージが表示されます。その際、クライアント・アプリケーションはトランザクションをロールバックします。

図 5-1 では、Transactions サンプル・アプリケーションのしくみを示しています。

図 5-1Transactions サンプル・アプリケーション



Transactions サンプル・アプリケーションの開発プロセス

ここでは、Transactions サンプル・アプリケーションにトランザクションを追加する手順について説明します。以下の手順は、第 3 章 「Basic サンプル・アプリケーション」で概説した開発手順の追加手順です。

注記　この節に記載されている手順はすでに完了しており、Transactions サンプル・アプリケーションに組み込まれています。

OMG IDL

開発プロセスでは、Object Management Group (OMG) インターフェイス定義言語 (IDL) で `Registrar` の `register_for_courses()` オペレーションを定義します。`register_for_courses()` オペレーションには、`NotRegisteredList` パラメータがあり、登録に失敗したコースのリストをクライアント・アプリケーションに返します。`NotRegisteredList` の値が空の場合、クライアント・アプリケーションはトランザクションをコミットします。

また、`TooManyCredits` ユーザ例外も定義する必要があります。

クライアント・アプリケーションの記述

開発プロセスでは、クライアント・アプリケーションに次のコードを追加します。

- 指定した BEA Tuxedo ドメインの TransactionCurrent 環境オブジェクトへのリファレンスを取得するための Bootstrap 環境オブジェクト
- トランザクションに CORBA オブジェクトを組み込むための TransactionCurrent 環境オブジェクトのオペレーション
- 学生がコースを登録できるようにするための `register_for_courses()` の呼び出し

クライアント・アプリケーションでのトランザクションの使用については、『BEA Tuxedo CORBA アプリケーション入門』および『BEA Tuxedo CORBA トランザクション』を参照してください。

University サーバ・アプリケーション

開発プロセスでは、University サーバ・アプリケーションに次のコードを追加します。

- Server オブジェクトの Server::initialize() および Server::release() オペレーションでの、TP::open_xa_rm() および TP::close_xa_rm() オペレーションの呼び出し
- register_for_courses() オペレーションのメソッド・インプリメンテーション

上記の作業については、『BEA Tuxedo CORBA サーバ・アプリケーションの開発方法』を参照してください。

UBBCONFIG ファイル

開発プロセスでは、UBBCONFIG ファイルで次のものが必要になります。

- University サーバ・アプリケーションおよびデータベースを管理するサーバ・アプリケーションの両方を含んだサーバ・グループ。このサーバ・グループは、トランザクションとして指定する必要があります。
- Oracle データベースの XA パラメータに従って定義された OPENINFO パラメータ。Oracle データベースの XA パラメータの説明については、『Oracle7 Server Distributed Systems』の「Developing and Installing Applications that Use the XA Libraries」を参照してください。

注記　Oracle 以外のデータベースを使用する場合、XA パラメータの定義については製品のマニュアルを参照してください。

- TLOGDEVICE パラメータでのトランザクション・ログ (TLOG) のパス名

トランザクション・ログおよび UBBCONFIG ファイルのパラメータの定義については、『BEA Tuxedo アプリケーションの設定』を参照してください。

ICF ファイル

開発プロセスでは、Registrar オブジェクトのトランザクション方針を optional から always に変更します。always トランザクション方針は、このオブジェクトが必ずトランザクションの一部であることを示します。CORBA オブジェクトのトランザクション方針の定義については、『BEA Tuxedo CORBA トランザクション』を参照してください。

Transactions サンプル・アプリケーションのビルド

Transactions サンプル・アプリケーションをビルドするには、次の手順に従います。

1. Transactions サンプル・アプリケーションのファイルをコピーします。
2. Transactions サンプル・アプリケーションのファイルの保護を変更します。
3. 環境変数を設定します。
4. University データベースを初期化します。
5. UBBCONFIG ファイルをロードします。
6. トランザクション・ログを作成します。
7. クライアントおよびサーバ・サンプル・アプリケーションをビルドします。

以降の節では、上記の各手順について説明します。

注記　Transactions サンプル・アプリケーションをビルドまたは実行する前に、第 2 章 「環境設定」の手順を実行しておく必要があります。

Transactions サンプル・アプリケーションのファイルを作業ディレクトリにコピーする

Transactions サンプル・アプリケーションの各ファイルは、次のディレクトリにあります。

Windows 2000

drive:*TUXDIR*\samples\corba\university\transaction

UNIX

/usr/*TUXDIR*/samples/corba/university/transaction

また、utils ディレクトリも作業ディレクトリにコピーする必要があります。utils ディレクトリには、ログ、トレース、および University データベースへのアクセスを設定するファイルが格納されています。

表 5-1 に、Transactions サンプル・アプリケーションの作成に使用するファイルの一覧を示します。

表 5-1 Transactions サンプル・アプリケーションに含まれるファイル

ファイル	説明
univt.idl	CourseSynopsisEnumerator、Registrar、および RegistrarFactory インターフェイスを宣言する OMG IDL。
univts.cpp	Transactions サンプル・アプリケーションの University サーバ・アプリケーション用 C++ ソース・コード。
univt_i.h univt_i.cpp	CourseSynopsisEnumerator、Registrar、および RegistrarFactory インターフェイスのメソッド・インプリメンテーション用 C++ ソース・コード。
univtc.cpp	Transactions サンプル・アプリケーションの CORBA C++ クライアント・アプリケーション用 C++ ソース・コード。

表 5-1 Transactions サンプル・アプリケーションに含まれるファイル (続き

ファイル	説明
frmBrowser.frm	Transactions サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic ソース・コード。
frmOpen.frm	Transactions サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic ソース・コード。
University.vbp	Transactions サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic プロジェクト・ファイル。
University.vbw	Transactions サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic ワークスペース・ファイル。
modPublicDeclarations.bas	サンプル・アプリケーションで使用される変数の宣言が記述された Visual Basic ファイル。
frmTracing.frm frmTracing.frx	ActiveX クライアント・アプリケーションにトレース機能を提供するファイル。
frmLogon.frm	ActiveX クライアント・アプリケーションに対してセキュリティ・ログオンを実行する Visual Basic ファイル。
UnivTApplet.java	Transactions サンプル・アプリケーションの CORBA Java クライアント・アプリケーション用 Java ソース・コード。
univt_utils.h univt_utils.cpp	CORBA C++ クライアント・アプリケーションのデータベース・アクセス関数を定義するファイル。
univt.icf	Transactions サンプル・アプリケーションの ICF ファイル。

表 5-1 Transactions サンプル・アプリケーションに含まれるファイル (続き

ファイル	説明
setenvt.sh	Transactions サンプル・アプリケーションのビルドおよび実行に必要な環境変数を設定する UNIX スクリプト。
setenvt.cmd	Transactions サンプル・アプリケーションのビルドおよび実行に必要な環境変数を設定する MS-DOS コマンド。
ubb_t.mk	UNIX オペレーティング・システム用の UBBCONFIG ファイル。
ubb_t.nt	Windows 2000 オペレーティング・システム用の UBBCONFIG ファイル。
makefilet.mk	UNIX オペレーティング・システムでの Transactions サンプル・アプリケーション用の makefile。
makefilet.nt	Windows 2000 オペレーティング・システムでの Transactions サンプル・アプリケーション用の makefile。
log.cpp、log.h、log_client.cpp、log_server.cpp	サンプル・アプリケーションにログ機能とトレース機能を提供するクライアント・アプリケーションとサーバ・アプリケーション。これらのファイルは、\utils ディレクトリにあります。
oradbconn.cpp、oranoconn.cpp	Oracle SQL データベース・インスタンスへのアクセスを提供するファイル。これらのファイルは、\utils ディレクトリにあります。
samplesdb.cpp、samplesdb.h	サンプル・アプリケーションでのデータベース例外に出力関数を提供するファイル。これらのファイルは、\utils ディレクトリにあります。

表 5-1 Transactions サンプル・アプリケーションに含まれるファイル (続き

ファイル	説明
unique_id.cpp、 unique_id.h	サンプル・アプリケーションの C++ Unique ID クラスのルーチン。これらのファイルは、\utils ディレクトリにあります。
samplesdbsql.h、 samplesdbsql.pc	SQL データベースへのアクセスをインプリメントする C++ クラスのメソッド。これらのファイルは、\utils ディレクトリにあります。
university.sql	University データベース用の SQL。このファイルは、\utils ディレクトリにあります。

Transactions サンプル・アプリケーションのファイル保護の属性を変更する

BEA Tuxedo ソフトウェアのインストール時には、サンプル・アプリケーションは読み取り専用に設定されています。Transactions サンプル・アプリケーションのファイルを編集または作成するには、次のように作業ディレクトリにコピーしたファイル保護の属性を変更する必要があります。

Windows 2000

```
prompt>attrib -r drive:\workdirectory\*.*
```

UNIX

```
prompt>chmod u+rw /workdirectory/*.*
```

環境変数を設定する

次のコマンドを使用して、Transactions サンプル・アプリケーションのクライアント・アプリケーションとサーバ・アプリケーションのビルドに使用する環境変数を設定します。

Windows 2000

```
prompt>setenvt
```

UNIX

```
prompt>/bin/ksh
prompt>. ./setenvt.sh
```

University データベースを初期化する

次のコマンドを使用して、Transactions サンプル・アプリケーションで使用する University データベースを初期化します。

Windows 2000

```
prompt>nmake -f makefilet.nt initdb
```

UNIX

```
prompt>make -f makefilet.mk initdb
```

UBBCONFIG ファイルをロードする

次のコマンドを使用して、UBBCONFIG ファイルをロードします。

Windows 2000

```
prompt>tmloadcf -y ubb_t.nt
```

UNIX

```
prompt>tmloadcf -y ubb_t.mk
```

UBBCONFIG ファイルの作成プロセスでは、アプリケーション・パスワードの入力が求められます。このパスワードは、クライアント・アプリケーションへのログオンに使用されます。パスワードを入力して Enter キーを押します。その際、パスワードを再入力してパスワードの確認を求めるメッセージが表示されます。

トランザクション・ログを作成する

トランザクション・ログには、CORBA アプリケーションでのトランザクション処理が記録されます。開発プロセスでは、UBBCONFIG ファイルの TLOGDEVICE パラメータでトランザクション・ログの場所を定義する必要があります。Transactions サンプル・アプリケーションの場合、トランザクション・ログは作業ディレクトリに格納されています。

Transactions サンプル・アプリケーションのトランザクション・ログを開くには、次の手順に従います。

1. 次のコマンドを入力して、会話型の管理インターフェイスを起動します。

   ```
   tmadmin
   ```

2. 次のコマンドを入力して、トランザクション・ログを作成します。

   ```
   crdl -b blocks -z directorypath
   clog -m SITE1
   ```

   この例では、

   blocks にトランザクション・ログに割り当てるブロック数を指定し、*directorypath* にトランザクション・ログの場所を指定します。*directorypath* オプションは、UBBCONFIG ファイルの TLOGDEVICE パラメータで指定した場所と一致しなければなりません。Windows 2000 でのコマンドの例を次に示します。

   ```
   crdl -b 500 -z c:\mysamples\university\Transaction\TLOG
   ```

3.「q」を入力して、会話型の管理インターフェイスを終了します。

Transactions サンプル・アプリケーションのコンパイル

開発プロセスでは、buildobjclient および buildobjserver コマンドを使用して、クライアント・アプリケーションとサーバ・アプリケーションをビルドします。また、クライアント・アプリケーションとサーバ・アプリケーションのトランザクション・イベントを調整するために、データベース固有のトランザクション・マネージャを作成します。ただし、Transactions サンプル・アプリケーションの場合は、この手順は不要です。Transactions サンプル・アプリケーションのディレクトリには、makefile が格納されています。この makefile により、クライアントとサーバ・サンプル・アプリケーションがビルドされ、TMS_ORA というトランザクション・マネージャが作成されます。

注記　makefile では、次のパラメータがハード・コードされ、Oracle データベース用のトランザクション・マネージャが作成されます。

```
RM=Oracle_XA
```

Oracle 以外のデータベースを使用する場合は、上記のパラメータを変更する必要があります。

Transactions サンプル・アプリケーションの CORBA C++ クライアント・アプリケーションとサーバ・アプリケーションをビルドするには、次のコマンドを使用します。

Windows 2000

```
prompt>nmake -f makefilet.nt
```

UNIX

```
prompt>make -f makefilet.mk
```

CORBA Java クライアント・アプリケーションをビルドするには、次のコマンドを使用します。

Windows 2000

```
prompt>nmake -f makefilet.nt javaclient
```

UNIX

```
prompt>make -f makefilet.mk javaclient
```

ActiveX クライアント・アプリケーションの起動については、「ActiveX クライアント・アプリケーションの起動」を参照してください。

`buildobjclient` および `buildobjserver` コマンドの詳細については、『BEA Tuxedo コマンド・リファレンス』を参照してください。

Transactions サンプル・アプリケーションの実行

Transactions サンプル・アプリケーションを実行するには、次の手順に従います。

1. サーバ・アプリケーションを起動します。
2. 1 つまたは複数のクライアント・アプリケーションを起動します。

上記の手順については、以下の節を参照してください。

サーバ・アプリケーションの起動

Transactions サンプル・アプリケーションでシステムおよびサンプル・アプリケーションのサーバ・アプリケーションを起動するには、次のコマンドを入力します。

```
prompt>tmboot -y
```

このコマンドを入力すると、次のサーバ・プロセスが開始されます。

- `TMSYSEVT`

  BEA Tuxedo システムの EventBroker。

- TMFFNAME

  NameManager サービスや FactoryFinder サービスなどのトランザクション管理サービス。

- TMIFSRVR

  インターフェイス・リポジトリ・サーバ・プロセス。これは、ActiveX クライアント・アプリケーションでのみ使用されます。

- univt_server

  University サーバ・プロセス。

- ISL

  IIOP リスナ / ハンドラ・プロセス。

ほかのサンプル・アプリケーションを使用するには、次のコマンドを入力して、システムおよびサンプル・アプリケーションのサーバ・プロセスを停止します。

```
prompt>tmshutdown
```

CORBA C++ クライアント・アプリケーションの起動

Transactions サンプル・アプリケーションの CORBA C++ クライアント・アプリケーションを起動するには、次の手順に従います。

1. MS-DOS プロンプトで次のコマンドを入力します。

   ```
   prompt>univt_client
   ```

2. 画面に「Enter student id:」と表示されたら、100001 ～ 100010 の任意の数値を入力します。
3. Enter キーを押します。
4. 画面に「Enter domain password:」と表示されたら、UBBCONFIG ファイルをロードしたときに定義したパスワードを入力します。
5. Enter キーを押します。

CORBA Java クライアント・アプリケーションの起動

Transactions サンプル・アプリケーションの CORBA Java クライアント・アプリケーションを実行するには、以下の手順に従います。

1. `UnivTApplet.html` ファイル内の次の行を変更します。

```
code="UnivTApplet.class"
codebase=.
```

上記の行を次のように変更します。

```
code="UnivTApplet"
archive="UnivTApplet.jar,m3envobj.jar"
```

2. 変更した `UnivTApplet.html` ファイルを Web サーバのソース・ディレクトリにコピーします。ディレクトリは、Web サーバ製品によって異なります。

3. `makefile` を実行して Transactions サンプル・アプリケーションをビルドした後、次のように `UnivTApplet.jar` ファイルを作成します。

   a. サンプル・アプリケーションをビルドしたディレクトリの下に `tmp` ディレクトリを作成します。`UniversityT` サブディレクトリとその中に格納されているクラス・ファイルを `tmp` ディレクトリにコピーします。

   `makefile` によって生成された、Transactions サンプル・アプリケーション・ディレクトリにあるクラス・ファイルを `tmp` ディレクトリにコピーします。ディレクトリを `tmp` ディレクトリに設定 (`cd`) し、次のいずれかのコマンドを入力して、すべての Transactions サンプル・アプリケーション・クラスを含んだ `.JAR` ファイルを作成します。

   `jar -cf ..\UnivTApplet.jar *.*` (Microsoft Windows 2000 システム)
   `jar -cf ../UnivTApplet.jar *` (UNIX システム)

4. 前の手順で作成した `UnivTApplet.jar` ファイルを Web サーバのソース・ディレクトリにコピーします。ディレクトリ名は、Web サーバ製品によって異なります。

5. 適切なサブディレクトリ (Microsoft Windows 2000 システムの場合は `%TUXDIR%\udataobj\java`、UNIX システムの場合は `${TUXDIR}/udataobj/java`) から、`m3envobj.jar` ファイルを Web サーバのソース・ディレクトリにコピーします。
6. Transactions サーバ・アプリケーションが動作していることを確認してから、Web ブラウザを起動し、Web サーバが動作しているノードを参照します。

注記　Microsoft Windows 2000 システムの場合、ノード名はすべて大文字にする必要があります。たとえば、`UBBCONFIG` ファイルおよび `UnivTApplet.html` ファイルでノードを `SERVER` に指定した場合、ブラウザは `http://SERVER/UnivTApplet.html` に設定します。

7. student ID フィールドに 100001 ～ 100010 の数値を入力します。
8. Domain Password フィールドに、`UBBCONFIG` ファイルをロードしたときに定義したパスワードを入力します。
9. [Logon] ボタンをダブルクリックします。

ActiveX クライアント・アプリケーションの起動

注記　University サンプル・アプリケーションでは、インターフェイス・リポジトリに CORBA インターフェイスの OMG IDL をロードする作業は `makefile` によって自動化されています。

ActiveX クライアント・アプリケーションを起動するには、Application Builder を使用して CORBA インターフェイスの ActiveX バインディングを作成する必要があります。

CORBA インターフェイスの ActiveX バインディングを作成するには、次の手順に従います。

1. BEA Tuxedo プログラム・グループで [BEA Application Builder] アイコンをクリックします。

   [IIOP Listener] ウィンドウが表示されます。

2. [IIOP Listener] ウィンドウに、UBBCONFIG ファイルの ISL パラメータで指定したホスト名とポート番号を入力します。ホスト名とポート番号は、UBBCONFIG ファイルで大文字で指定した内容と正確に一致するように入力します。

   [Logon] ウィンドウが表示されます。

3. UBBCONFIG ファイルをロードしたときに定義したユーザ名とパスワードについて、100001 ～ 100010 の学生 ID を [Logon] ウィンドウに入力します。

   [Application Builder] ウィンドウが表示されます。インターフェイス・リポジトリにロードされた CORBA インターフェイスがすべて、Application Builder の [Services] ウィンドウに表示されます。

4. [Services] ウィンドウで UniversityT フォルダを強調表示して [Workstation Views] ウィンドウにドラッグします。または、[Services] ウィンドウから UniversityT フォルダをコピーして [Workstation Views] ウィンドウに貼り付けます。

   確認のウィンドウが表示されます。

5. [Create] をクリックすると、Transactions サンプル・アプリケーションで CORBA インターフェイスの ActiveX バインディングが作成されます。

   Application Builder では、以下のものが作成されます。

   - CORBA インターフェイスに対するバインディング。バインディングの名前の形式は、DI*modulename_interfacename* です。たとえば、Registrar インターフェイスに対するバインディングの名前は、DIUniversityT_Registrar となります。
   - 型ライブラリ。デフォルトでは、型ライブラリは *TUXDIR*\TypeLibraries に格納されています。

     型ライブラリのファイル名の形式は、DI*modulename_interfacename*.tlb です。
   - CORBA インターフェイスの Windows システム・レジストリ・エントリ。このエントリには、オブジェクト型ごとに一意のプログラム ID が入っています。

ActiveX クライアント・アプリケーションを実行するには、以下の手順に従います。

1. Visual Basic で `University.vbw` ファイルを開きます。
2. [Run] メニューの [Start] をクリックします。

   [IIOP Listener] ウィンドウが表示されます。
3. [IIOP Listener] ウィンドウに、`UBBCONFIG` ファイルの `ISL` パラメータで指定したホスト名とポート番号を入力します。ホスト名とポート番号は、`UBBCONFIG` ファイルで大文字で指定した内容と正確に一致するように入力します。

   [Logon] ウィンドウが表示されます。
4. `UBBCONFIG` ファイルをロードしたときに定義したユーザ名とパスワードについて、100001 ～ 100010 の学生 ID を [Logon] ウィンドウに入力します。

Transactions サンプル・アプリケーションのクライアント・アプリケーションの使用方法

以降の節では、Transactions サンプル・アプリケーションのクライアント・アプリケーションの使用方法について簡単に説明します。

CORBA C++ クライアント・アプリケーション

Transactions サンプル・アプリケーションの CORBA C++ クライアント・アプリケーションには、次の追加オプションがあります。

```
<R>    Register for Courses
```

コースに登録するには、次の手順に従います。

1. Options プロンプトで「R」を入力します。
2. 画面に「Course Number」と表示されたら、コース番号とその後に続けて「-1」を入力します。たとえば、次のように入力します。

```
100011
100039
-1
```

3. Enter キーを押します。
4. Options プロンプトで「L」を入力すると、学生 ID が登録されているコースを一覧表示できます。

C++ CORBA クライアント・アプリケーションを終了するには、Options プロンプトで「E」を入力します。

CORBA Java クライアント・アプリケーション

CORBA Java クライアント・アプリケーションにログオンすると、[Student Account Summary] ウィンドウが表示されます。このウィンドウでは、コースへの登録ができます。

クラスへの登録を行うには、次の手順に従います。

1. [Search String] テキスト・ボックスにテキスト文字列を入力して、[Student Account Summary] ウィンドウから利用可能なコースのリストを取得します。たとえば、「computer」などの文字列を入力します。
2. [Search Catalog] ボタンをクリックします。

   検索文字列に一致したコースがウィンドウに一覧表示されます。
3. [Student Account Summary] ウィンドウの下部に表示されるコース名をクリックしてコースを選択します。

   コースが登録済みの場合は、[Student Account Summary] ウィンドウの Registered フィールドに「Yes」と表示されます。

4. コースに登録するには、[Student Account Summary] ウィンドウの [Register] ボタンをクリックします。

学生 ID が登録されているコースを一覧表示するには、[Show Registration] ボタンをダブルクリックします。

CORBA Java クライアント・アプリケーションを終了するには、[Student Account Summary] ウィンドウの [Logoff] ボタンをクリックするか、または [Applet] メニューから [Quit] を選択します。

ActiveX クライアント・アプリケーション

ActiveX クライアント・アプリケーションにログオンすると、[Course Browser] ウィンドウが表示されます。このウィンドウでは、コースへの登録ができます。

クラスへの登録を行うには、次の手順に従います。

1. [Find Courses] ボタンの横にあるテキスト・ボックスにテキスト文字列を入力するか、またはプル・ダウン・メニューからカリキュラムのサブジェクトを選択します。たとえば、「`computer`」などの文字列を入力します。

2. [Find Courses] ボタンをクリックします。

   検索文字列に一致したコースがすべて一覧表示されます。

3. [Get Details] ボタンの横にあるウィンドウに表示されるリストからコースを選択するか、またはコース名をダブルクリックします。

   選択したコースの詳細が表示されます。

4. [Register for Course] ボタンをクリックするか、またはコース名をダブルクリックしてスケジュールにコースを入力します。

   ウィンドウの下部に学生のスケジュールにあるコースが表示されます。登録済みのコースは、緑で表示されます。以前に登録したコースと競合するコースは、赤で表示されます。

スケジュールからコースを削除するには、スケジュールでコースをダブルクリックします。

学生 ID が登録されているコースを一覧表示するには、[Get Registered Courses] ボタンをクリックします。

コースに関する情報の表示を閉じるには、[Get Details] ボタンをクリックします。

ActiveX クライアント・アプリケーションを終了するには、[File] メニューから [Exit] を選択します。

章

6 Wrapper サンプル・アプリケーション

ここでは、次の内容について説明します。

- Wrapper サンプル・アプリケーションのしくみ
- Wrapper サンプル・アプリケーションの開発プロセス
- Wrapper サンプル・アプリケーションのビルド
- Wrapper サンプル・アプリケーションのコンパイル
- Wrapper サンプル・アプリケーションの実行
- Wrapper サンプル・アプリケーションのクライアント・アプリケーションの使用方法

トラブルシューティング情報については、`\wrapper` ディレクトリにある `Readme.txt` を参照してください。また、Wrapper サンプル・アプリケーションの使用方法に関する最新の情報も参照してください。

Wrapper サンプル・アプリケーションのしくみ

Wrapper サンプル・アプリケーションでは、クラスに登録した学生の口座にクラスの代金が請求され、その口座残高が更新されます。また、学生は自身の口座残高を照会できます。

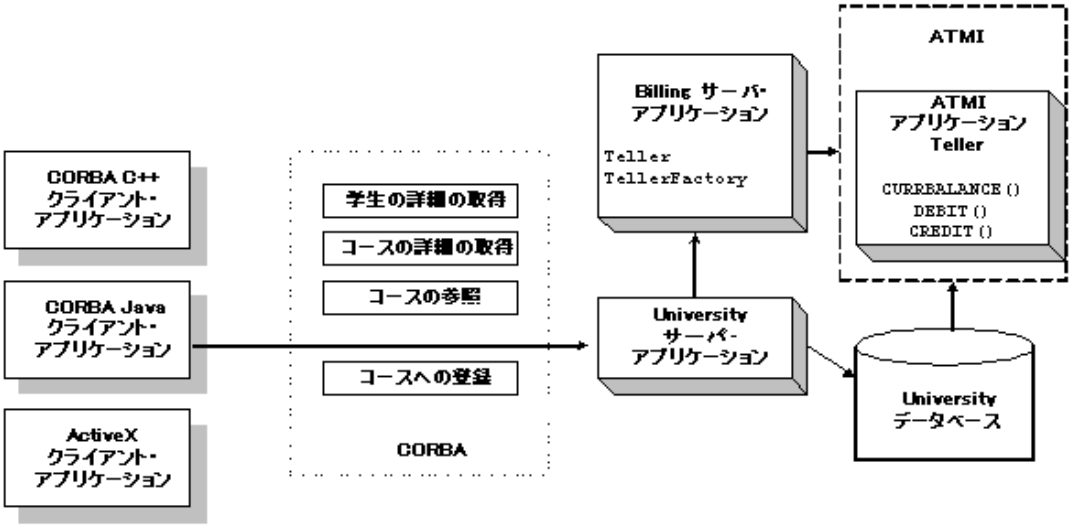
Wrapper サンプル・アプリケーションでは、ATMI サービスを使用します。Billing サーバ・アプリケーションには、`Teller` オブジェクトがあります。このオブジェクトは、ATMI アプリケーションの「Teller」を呼び出します。Teller アプリケーションは、次の課金用のオペレーションを実行します。

- クラスの代金を学生の口座の振替元に記入
- クラスの代金を学生の口座の振替先に記入
- 学生の現在の口座残高を照会

University データベースは、口座情報を含めるために変更されます。

図 6-1 では、Wrapper サンプル・アプリケーションのしくみを示しています。

図 6-1Wrapper サンプル・アプリケーション



Wrapper サンプル・アプリケーションの開発プロセス

ここでは、CORBA アプリケーションで ATMI サービスをラッピングする際に必要な開発プロセスについて説明します。以下の手順は、「第 3 章 Basic サンプル・アプリケーション」で概説した開発手順の追加手順です。

注記　この節に記載されている手順はすでに完了しており、Wrapper サンプル・アプリケーションに組み込まれています。

OMG IDL

ATMI サービスをラッピングする際は、ATMI サービスと相互運用するオブジェクト、およびそのオブジェクトを作成するファクトリを定義する必要があります。Wrapper サンプル・アプリケーションの場合、Teller と TellerFactory オブジェクトが ATMI サービスとの相互運用を行います。開発プロセスでは、Object Management Group (OMG) インターフェイス定義言語 (IDL) で Teller および TellerFactory オブジェクトのインターフェイスを次のように定義します。

オブジェクト	説明	オペレーション
TellerFactory	Teller オブジェクトへのオブジェクト・リファレンスを返します。	find_teller()
Teller	ATMI アプリケーションと相互運用して、課金用のオペレーションを実行します。	get_balance() credit() debit()

また、Balance フィールドを StudentDetails 構造体に追加する必要があります。クライアント・アプリケーションでは、Balance フィールドを使用して学生の口座残高を示します。さらに、ユーザ例外 DelinquentAccount も追加します。

クライアント・アプリケーション

開発プロセスでは、クライアント・アプリケーションにコードを追加して、register_for_courses() オペレーションによって生成されるユーザ例外 Delinquent Account を処理します。

サーバ・アプリケーション

開発プロセスでは、Billing サーバ・アプリケーションに以下を記述します。

- Teller オブジェクトの get_balance()、credit()、および debit() オペレーションのメソッド・インプリメンテーション。これらのメソッド・インプリメンテーションには、次の処理を行うコードを含める必要があります。
  - FML メッセージ・バッファを割り当てる。
  - ATMI アプリケーション Teller に送信するデータを FML メッセージ・バッファに格納する。
  - ATMI アプリケーション Teller を呼び出す。
  - ATMI アプリケーション Teller から返された FML メッセージ・バッファから情報を抽出する。
  - FML メッセージ・バッファから University サーバ・アプリケーションに情報を返す。
- TellerFactory オブジェクトの find_teller() オペレーションのメソッド・インプリメンテーション。
- TellerFactory オブジェクトを作成および登録し、open_XA_RM および close_XA_RM 関数を呼び出す Billing サーバ・オブジェクト。

開発プロセスでは、University サーバ・アプリケーションに次のコードを追加します。

- University サーバ・アプリケーションのコードのサーバ初期化部分に Bootstrap オブジェクトを組み込んで、TellerFactory オブジェクトの FactoryFinder オブジェクトを取得します。University サーバ・アプリケーションでは、クライアント・アプリケーションの場合と同様に Bootstrap および FactoryFinder オブジェクトを使用します。
- University サーバ・アプリケーションのコードには、Registrar オブジェクトのサーバントのコンストラクタに TellerFactory オブジェクトへのリファレンスを含めます。TellerFactory オブジェクトは、Teller オブジェクトの作成に使用します。

- Registrar オブジェクトの get_student_details() および register_for_courses() オペレーションのメソッド・インプリメンテーションでは、Teller オブジェクトの get_balance() および debit() オペレーションを呼び出します。

ATMI サービスをラッピングするサーバ・アプリケーションの記述方法については、『BEA Tuxedo CORBA サーバ・アプリケーションの開発方法』を参照してください。

UBBCONFIG ファイル

開発プロセスでは、UBBCONFIG ファイルに以下の変更を加える必要があります。

- UBBCONFIG ファイルの GROUPS セクションで次のサーバ・グループを定義します。
  - ORA_GRP。これには、University サーバ・アプリケーション、Teller アプリケーション、および University データベースのサーバ・アプリケーションが含まれます。このサーバ・グループによって、University サーバ・アプリケーションと Teller アプリケーションの両方から University データベースへのアクセスが可能になります。
  - APP_GRP。これには、Billing サーバ・アプリケーションが含まれます。
- Wrapper サンプル・アプリケーションのサーバ・アプリケーションを指定します。サーバ・アプリケーションは、UBBCONFIG ファイルの SERVERS セクションの順序で起動します。サーバ・アプリケーションの起動順序は次のとおりです。
  a. ATMI アプリケーション Teller
  b. Billing サーバ・アプリケーション
  c. University サーバ・アプリケーション

ICF ファイル

開発プロセスでは、Teller および TellerFactory オブジェクトの活性化方針とトランザクション方針を定義する必要があります。Teller および TellerFactory オブジェクトの各方針は次のように設定します。

- Teller オブジェクトの活性化方針は process に、トランザクション方針は optional に設定します。
- TellerFactory オブジェクトの活性化方針は process に、トランザクション方針は ignore に設定します。

CORBA オブジェクトの活性化方針とトランザクション方針の定義については、『BEA Tuxedo CORBA サーバ・アプリケーションの開発方法』を参照してください。

Wrapper サンプル・アプリケーションのビルド

Wrapper サンプル・アプリケーションをビルドするには、次の手順に従います。

1. Wrapper サンプル・アプリケーションのファイルをコピーします。
2. Wrapper サンプル・アプリケーションのファイルの保護を変更します。
3. 環境変数を設定します。
4. University データベースを初期化します。
5. UBBCONFIG ファイルをロードします。
6. トランザクション・ログを作成します。
7. クライアントおよびサーバ・サンプル・アプリケーションをビルドします。

以降の節では、上記の各手順について説明します。

注記　Wrapper サンプル・アプリケーションをビルドまたは実行する前に、「第 2 章 環境設定」の手順を実行しておく必要があります。

Wrapper サンプル・アプリケーションのファイルを作業ディレクトリにコピーする

Wrapper サンプル・アプリケーションの各ファイルは、次のディレクトリにあります。

Windows 2000

drive:*TUXDIR*\samples\corba\university\wrapper

UNIX

/usr/*TUXDIR*/samples/corba/university/wrapper

また、utils ディレクトリも作業ディレクトリにコピーする必要があります。utils ディレクトリには、ログ、トレース、および University データベースへのアクセスを設定するファイルが格納されています。

表 6-1 に、Wrapper サンプル・アプリケーションの作成に使用するファイルの一覧を示します。

表 6-1 Wrapper サンプル・アプリケーションに含まれるファイル

ファイル	説明
billw.idl	Teller および TellerFactory インターフェイスを宣言する OMG IDL。
univw.idl	CourseSynopsisEnumerator、Registrar、および RegistrarFactory インターフェイスを宣言する OMG IDL。

表 6-1 Wrapper サンプル・アプリケーションに含まれるファイル (続き)

ファイル	説明
billws.cpp	Wrapper サンプル・アプリケーションの Billing サーバ・アプリケーション用 C++ ソース・コード。
univws.cpp	Wrapper サンプル・アプリケーションの University サーバ・アプリケーション用 C++ ソース・コード。
billw__i.h billw_i.cpp	Teller および TellerFactory インターフェイスのメソッド・インプリメンテーション用 C++ ソース・コード。
univw_i.h univw_i.cpp	CourseSynopsisEnumerator、Registrar、および RegistrarFactory インターフェイスのメソッド・インプリメンテーション用 C++ ソース・コード。
univwc.cpp	Wrapper サンプル・アプリケーションの CORBA C++ クライアント・アプリケーション用 C++ ソース・コード。
frmBrowser.frm	Wrapper サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic ソース・コード。
university.vbp	Wrapper サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic プロジェクト・ファイル。
University.vbw	Wrapper サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic ワークスペース・ファイル。
modPublicDeclarations.bas	サンプル・アプリケーションで使用される変数の宣言が記述された Visual Basic ファイル。
frmTracing.frm frmTracing.frx	ActiveX クライアント・アプリケーションにトレース機能を提供するファイル。

表 6-1 Wrapper サンプル・アプリケーションに含まれるファイル (続き)

ファイル	説明
frmLogon.frm	ActiveX クライアント・アプリケーションに対してセキュリティ・ログオンを実行する Visual Basic ファイル。
univWApplet.java	Wrapper サンプル・アプリケーションの CORBA Java クライアント・アプリケーション用 Java ソース・コード。
univw_utils.h univw_utils.cpp	CORBA C++ クライアント・アプリケーションのデータベース・アクセス関数を定義するファイル。
univw.icf	Wrapper サンプル・アプリケーションの University サーバ・アプリケーション用 ICF ファイル。
billw.icf	Wrapper サンプル・アプリケーションの Billing サーバ・アプリケーション用 ICF ファイル。
setenvw.sh	Wrapper サンプル・アプリケーションのビルドおよび実行に必要な環境変数を設定する UNIX スクリプト。
tellw_flds、tellw_u.c、tellw_c.h、tellws.ec	ATMI アプリケーション Teller のファイル。
setenvw.cmd	Wrapper サンプル・アプリケーションのビルドおよび実行に必要な環境変数を設定する MS-DOS コマンド。
ubb_w.mk	UNIX オペレーティング・システム用の UBBCONFIG ファイル。
ubb_w.nt	Windows 2000 オペレーティング・システム用の UBBCONFIG ファイル。

表 6-1 Wrapper サンプル・アプリケーションに含まれるファイル (続き)

ファイル	説明
makefilew.mk	UNIX オペレーティング・システムでの Wrapper サンプル・アプリケーション用の makefile。
makefilew.nt	Windows 2000 オペレーティング・システムでの Wrapper サンプル・アプリケーション用の makefile。
log.cpp、log.h、log_client.cpp、log_server.cpp	サンプル・アプリケーションにログ機能とトレース機能を提供するクライアント・アプリケーションとサーバ・アプリケーションのファイル。これらのファイルは、\utils ディレクトリにあります。
oradbconn.cpp、oranoconn.cpp	Oracle SQL データベース・インスタンスへのアクセスを提供するファイル。これらのファイルは、\utils ディレクトリにあります。
samplesdb.cpp、samplesdb.h	サンプル・アプリケーションでのデータベース例外に出力関数を提供するファイル。これらのファイルは、\utils ディレクトリにあります。
unique_id.cpp、unique_id.h	サンプル・アプリケーションの C++ Unique ID クラスのルーチン。これらのファイルは、\utils ディレクトリにあります。
samplesdbsql.h、samplesdbsql.pc	SQL データベースへのアクセスをインプリメントする C++ クラスのメソッド。これらのファイルは、\utils ディレクトリにあります。
university.sql	University データベース用の SQL。このファイルは、\utils ディレクトリにあります。

Wrapper サンプル・アプリケーションのファイル保護の属性を変更する

BEA Tuxedo ソフトウェアのインストール時には、サンプル・アプリケーションは読み取り専用に設定されています。Wrapper サンプル・アプリケーションのファイルを編集または作成するには、次のように作業ディレクトリにコピーしたファイル保護の属性を変更する必要があります。

Windows 2000

```
prompt>attrib -r drive:\workdirectory\*.*
```

UNIX

```
prompt>chmod u+rw /workdirectory/*.*
```

環境変数を設定する

次のコマンドを使用して、Wrapper サンプル・アプリケーションのクライアント・アプリケーションとサーバ・アプリケーションのビルドに使用する環境変数を設定します。

Windows 2000

```
prompt>setenvw
```

UNIX

```
prompt>/bin/ksh
prompt>. ./setenvw.sh
```

University データベースを初期化する

次のコマンドを使用して、Wrapper サンプル・アプリケーションで使用する University データベースを初期化します。

Windows 2000

```
prompt>nmake -f makefilew.nt initdb
```

UNIX

```
prompt>make -f makefilew.mk initdb
```

UBBCONFIG ファイルをロードする

次のコマンドを使用して、UBBCONFIG ファイルをロードします。

Windows 2000

```
prompt>tmloadcf -y ubb_w.nt
```

UNIX

```
prompt>tmloadcf -y ubb_w.mk
```

UBBCONFIG ファイルの作成プロセスでは、アプリケーション・パスワードの入力が求められます。このパスワードは、クライアント・アプリケーションへのログオンに使用されます。パスワードを入力して Enter キーを押します。その際、パスワードを再入力してパスワードの確認を求めるメッセージが表示されます。

トランザクション・ログを作成する

トランザクション・ログには、CORBA アプリケーションでのトランザクション処理が記録されます。開発プロセスでは、UBBCONFIG ファイルの TLOGDEVICE パラメータでトランザクション・ログの場所を定義する必要があります。Wrapper サンプル・アプリケーションの場合、トランザクション・ログは作業ディレクトリに格納されています。

Wrapper サンプル・アプリケーションのトランザクション・ログを開くには、以下の手順に従います。

1. 次のコマンドを入力して、会話型の管理インターフェイスを起動します。
   ```
   tmadmin
   ```
2. 次のコマンドを入力して、トランザクション・ログを作成します。

```
crdl -b blocks -z directorypath
crlog -m SITE1
```

この例では、

トランザクション・ログに割り当てるブロック数を *blocks* に、トランザクション・ログの場所を *directorypath* に指定します。*directorypath* オプションは、UBBCONFIG ファイルの TLOGDEVICE パラメータで指定した場所と一致しなければなりません。Windows 2000 でのコマンドの例を次に示します。

```
crdl -b 500 -z c:\mysamples\university\wrapper\TLOG
```

3.「q」を入力して、会話型の管理インターフェイスを終了します。

Wrapper サンプル・アプリケーションのコンパイル

開発プロセスでは、buildobjclient および buildobjserver コマンドを使用して、クライアント・アプリケーションとサーバ・アプリケーションをビルドします。ただし、Wrapper サンプル・アプリケーションの場合は、この手順は不要です。Wrapper サンプル・アプリケーションのディレクトリには、makefile が格納されています。この makefile により、クライアントとサーバ・サンプル・アプリケーションがビルドされます。

Wrapper サンプル・アプリケーションの CORBA C++ クライアント・アプリケーションとサーバ・アプリケーションをビルドするには、次のコマンドを使用します。

Windows 2000

```
prompt>nmake -f makefilew.nt
```

UNIX

```
prompt>make -f makefilew.mk
```

CORBA Java クライアント・アプリケーションをビルドするには、次のコマンドを使用します。

Windows 2000

```
prompt>nmake -f makefilew.nt javaclient
```

UNIX

```
prompt>make -f makefilew.mk javaclient
```

ActiveX クライアント・アプリケーションの起動については、「ActiveX クライアント・アプリケーションの起動」を参照してください。

`buildobjclient` および `buildobjserver` コマンドの詳細については、『BEA Tuxedo コマンド・リファレンス』を参照してください。

Wrapper サンプル・アプリケーションの実行

Wrapper サンプル・アプリケーションを実行するには、次の手順に従います。

1. サーバ・アプリケーションを起動します。
2. 1 つまたは複数のクライアント・アプリケーションを起動します。

上記の手順については、以下の節を参照してください。

サーバ・アプリケーションの起動

Wrapper サンプル・アプリケーションでシステムおよびサンプル・アプリケーションのサーバ・プロセスを開始するには、次のコマンドを入力します。

```
prompt>tmboot -y
```

このコマンドを入力すると、次のサーバ・プロセスが開始されます。

- `TMSYSEVT`

  BEA Tuxedo システムの EventBroker。

- `TMFFNAME`

  NameManager サービスや FactoryFinder サービスなどのトランザクション管理サービス。

- `TMIFSRVR`

  インターフェイス・リポジトリ・サーバ・プロセス。これは、ActiveX クライアント・アプリケーションでのみ使用されます。

- `univw_server`

  University サーバ・プロセス。

- `tellw_server`

  ATMI アプリケーション Teller のアプリケーション・プロセス。

- `billw_server`

  Billing サーバ・アプリケーション・プロセス。

- `ISL`

  IIOP リスナ / ハンドラ・プロセス。

ほかのサンプル・アプリケーションを使用するには、次のコマンドを入力して、システムおよびサンプル・アプリケーションのサーバ・プロセスを停止します。

```
prompt>tmshutdown
```

CORBA C++ クライアント・アプリケーションの起動

Wrapper サンプル・アプリケーションの CORBA C++ クライアント・アプリケーションを起動するには、以下の手順に従います。

1. MS-DOS プロンプトで次のコマンドを入力します。

```
prompt>univw_client
```

2. 画面に「Enter student id:」と表示されたら、100001 ～ 100010 の任意の数値を入力します。
3. Enter キーを押します。
4. 画面に「Enter domain password:」と表示されたら、UBBCONFIG ファイルをロードしたときに定義したパスワードを入力します。
5. Enter キーを押します。

CORBA Java クライアント・アプリケーションの起動

Wrapper サンプル・アプリケーションの CORBA Java クライアント・アプリケーションを実行するには、以下の手順に従います。

1. UnivWApplet.html ファイル内の次の行を変更します。

```
 code="UnivWApplet.class"
codebase=.
```

上記の行を次のように変更します。

```
 code="UnivWApplet"
archive="UnivWApplet.jar,m3envobj.jar"
```

2. 変更した UnivWApplet.html ファイルを Web サーバのソース・ディレクトリにコピーします。ディレクトリは、Web サーバ製品によって異なります。
3. makefile を実行して Wrapper サンプル・アプリケーションをビルドした後、次のように UnivWApplet.jar ファイルを作成します。
   a. サンプル・アプリケーションをビルドしたディレクトリの下に tmp ディレクトリを作成します。UniversityW サブディレクトリとその中に格納されているクラス・ファイルを tmp ディレクトリにコピーします。

makefile によって生成された、Wrapper サンプル・アプリケーション・ディレクトリにあるクラス・ファイルを tmp ディレクトリにコピーします。ディレクトリを tmp ディレクトリに設定 (cd) し、次のいずれかのコマンドを入力して、すべての Wrapper サンプル・アプリケーション・クラスを含んだ .JAR ファイルを作成します。

```
jar -cf ..\UnivWApplet.jar *.* (Microsoft Windows 2000 システム )
jar -cf ../UnivPWpplet.jar * (UNIX システム )
```

4. 前の手順で作成した UnivWApplet.jar ファイルを Web サーバのソース・ディレクトリにコピーします。ディレクトリ名は、Web サーバ製品によって異なります。
5. 適切なサブディレクトリ (Microsoft Windows 2000 システムの場合は %TUXDIR%\udataobj\java、UNIX システムの場合は ${TUXDIR}/udataobj/java) から、m3envobj.jar ファイルを Web サーバのソース・ディレクトリにコピーします。
6. Wrapper サーバ・アプリケーションが動作していることを確認してから、Web ブラウザを起動し、Web サーバが動作しているノードを参照します。

注記　Microsoft Windows 2000 システムの場合、ノード名はすべて大文字にする必要があります。たとえば、UBBCONFIG ファイルおよび UnivWApplet.html ファイルでノードを SERVER に指定した場合、ブラウザは http://SERVER/UnivWApplet.html に設定します。

7. student ID フィールドに 100001 ～ 100010 の数値を入力します。
8. Domain Password フィールドに、UBBCONFIG ファイルをロードしたときに定義したパスワードを入力します。
9. [Logon] ボタンをダブルクリックします。

CORBA Java クライアント・アプリケーションを起動した、[Appletviewer] ウィンドウのステータス・バーまたは MS-DOS ウィンドウに例外が表示されます。

ActiveX クライアント・アプリケーションの起動

注記　University サンプル・アプリケーションでは、インターフェイス・リポジトリに CORBA インターフェイスの OMG IDL をロードする作業は `makefile` によって自動化されています。

ActiveX クライアント・アプリケーションを起動するには、Application Builder を使用して CORBA インターフェイスの ActiveX バインディングを作成する必要があります。

CORBA インターフェイスの ActiveX バインディングを作成するには、次の手順に従います。

1. BEA Tuxedo プログラム・グループで [BEA Application Builder] アイコンをクリックします。

   [IIOP Listener] ウィンドウが表示されます。

2. [IIOP Listener] ウィンドウに、`UBBCONFIG` ファイルの `ISL` パラメータで指定したホスト名とポート番号を入力します。ホスト名とポート番号は、`UBBCONFIG` ファイルで大文字で指定した内容と正確に一致するように入力します。

   [Logon] ウィンドウが表示されます。

3. `UBBCONFIG` ファイルをロードしたときに定義したユーザ名とパスワードについて、100001 ～ 100010 の学生 ID を [Logon] ウィンドウに入力します。

   [Application Builder] ウィンドウが表示されます。インターフェイス・リポジトリにロードされた CORBA インターフェイスがすべて、Application Builder の [Services] ウィンドウに表示されます。

4. [Services] ウィンドウで UniversityW フォルダを強調表示して [Workstation Views] ウィンドウにドラッグします。または、[Services] ウィンドウから UniversityW フォルダをコピーして [Workstation Views] ウィンドウに貼り付けます。

   確認のウィンドウが表示されます。

5. [Create] をクリックすると、Wrapper サンプル・アプリケーションで CORBA インターフェイスの ActiveX バインディングが作成されます。

   Application Builder では、以下のものが作成されます。

   - CORBA インターフェイスに対するバインディング。バインディングの名前の形式は、`DI`*`modulename_interfacename`* です。たとえば、`Registrar` インターフェイスに対するバインディングの名前は、`DIUniversityW_Registrar` となります。
   - 型ライブラリ。デフォルトでは、型ライブラリは `\`*`TUXDIR`*`\TypeLibraries` に格納されています。

     型ライブラリのファイル名の形式は、`DI`*`modulename_interfacename`*`.tlb` です。
   - CORBA インターフェイスの Windows システム・レジストリ・エントリ。このエントリには、オブジェクト型ごとに一意のプログラム ID が入っています。

ActiveX クライアント・アプリケーションを実行するには、以下の手順に従います。

1. Visual Basic で `University.vbw` ファイルを開きます。
2. [Run] メニューの [Start] をクリックします。

   [IIOP Listener] ウィンドウが表示されます。
3. [IIOP Listener] ウィンドウに、`UBBCONFIG` ファイルの `ISL` パラメータで指定したホスト名とポート番号を入力します。ホスト名とポート番号は、`UBBCONFIG` ファイルで大文字で指定した内容と正確に一致するように入力します。

   [Logon] ウィンドウが表示されます。
4. `UBBCONFIG` ファイルをロードしたときに定義したユーザ名とパスワードについて、100001 ～ 100010 の学生 ID を [Logon] ウィンドウに入力します。

Wrapper サンプル・アプリケーションのクライアント・アプリケーションの使用方法

以降の節では、Wrapper サンプル・アプリケーションのクライアント・アプリケーションの使用方法について説明します。

CORBA C++ クライアント・アプリケーション

Wrapper サンプル・アプリケーションの CORBA C++ クライアント・アプリケーションには、次の追加オプションがあります。

```
<B>    Display Your Balance
```

`Display Your Balance` オプションを使用すると、CORBA C++ クライアント・アプリケーションのログオンに使用する学生 ID に関連付けられた口座残高が表示されます。

C++ CORBA クライアント・アプリケーションを終了するには、Options プロンプトで「`E`」を入力します。

CORBA Java クライアント・アプリケーション

Wrapper サンプル・アプリケーションの CORBA Java クライアント・アプリケーションでは、口座残高を表示することができます。

CORBA Java クライアント・アプリケーションにログオンすると、[Student Account Summary] ウィンドウが表示されます。[Student Account Summary] ウィンドウには、CORBA Java クライアント・アプリケーションのログオン

に使用する学生 ID に関連付けられた口座残高が表示されます。コースを追加登録した場合、[Student Account Summary] ウィンドウの [Balance] テキスト・ボックスに表示される総額が増加します。

CORBA Java クライアント・アプリケーションを終了するには、[Student Account Summary] ウィンドウの [Logoff] ボタンをクリックするか、または [File] メニューから [Quit] を選択します。

ActiveX クライアント・アプリケーション

Wrapper サンプル・アプリケーションの ActiveX クライアント・アプリケーションでは、口座残高を表示することができます。

ActiveX クライアント・アプリケーションにログオンすると、[Course Browser] ウィンドウが表示されます。[Course Browser] ウィンドウには、登録したコース、および ActiveX クライアント・アプリケーションのログオンに使用する学生 ID の口座残高が表示されます。コースを追加登録した場合、[Course Browser] ウィンドウの [Balance] テキスト・ボックスに表示される総額が増加します。

ActiveX クライアント・アプリケーションを終了するには、[File] メニューから [Exit] を選択します。

章

7 Production サンプル・アプリケーション

ここでは、以下の内容について説明します。

- Production サンプル・アプリケーションのしくみ
- Production サンプル・アプリケーションの開発プロセス
- Production サンプル・アプリケーションのビルド
- Production サンプル・アプリケーションのコンパイル
- Production サンプル・アプリケーションの実行
- Production サンプル・アプリケーションをさらにスケーリングする方法

注記　Production サンプル・アプリケーションのクライアント・アプリケーションは、Wrapper サンプル・アプリケーションのクライアント・アプリケーションと同じ方法で動作します。

トラブルシューティング情報については、`\production` ディレクトリにある `Readme.txt` を参照してください。また、Production サンプル・アプリケーションの使用方法に関する最新の情報も参照してください。

Production サンプル・アプリケーションのしくみ

Production サンプル・アプリケーションには、Wrapper サンプル・アプリケーションと同じエンド・ユーザ向けの機能が用意されています。Production サンプル・アプリケーションでは、BEA Tuxedo ソフトウェアの CORBA 機能を利用して、CORBA アプリケーションをスケーリングする方法を示します。Production サンプル・アプリケーションでは、以下の処理を行います。

- UBBCONFIG ファイルに定義されている ORA_GRP および APP_GRP サーバ・グループ内で、University サーバ・アプリケーション、Billing サーバ・アプリケーション、および ATMI Teller アプリケーションを複製します。
- 追加のサーバ・マシン (Production マシン 2) 上で、ORA_GRP1 および APP_GRP1 サーバ・グループを ORA_GRP2 および APP_GRP2 として複製し、データベースを分割します。
- サーバ・アプリケーションが同時に管理できるクライアント・アプリケーションからの要求数を増やすために、状態を持たないオブジェクト・モデルをインプリメントします。
- 次のオブジェクトに一意なオブジェクト ID (OID) を割り当てて、対応するサーバ・グループでそれらのオブジェクトを同時に複数回インスタンス化できるようにします。これにより、次のオブジェクトをプロセス単位ではなく、クライアント・アプリケーション単位で利用できるようになります。
  - Registrar
  - RegistrarFactory
  - Teller
  - TellerFactory
- 一部の学生から 1 台のマシンへの要求、およびほかの学生から別のサーバ・マシンへの要求の代わりに、クライアント・アプリケーションからの直接の要求にファクトリ・ベース・ルーティングをインプリメントします。

注記　Production サンプル・アプリケーションの使用を簡単にするために、BEA Tuxedo ソフトウェア・キットでは、1 台のマシン上で 1 つのデータベースを使用して実行できるようにコンフィギュレーションされています。ただし、Production サンプル・アプリケーションでは、複数のマシンで複数のデータベースを使用して実行できるようにコンフィギュレーションされています。複数のマシンおよびデータベースへのコンフィギュレーションを変更する場合、`UBBCONFIG` ファイルを変更し、データベースを分割するだけで変更できます。

以下の節では、Production サンプル・アプリケーションで、複製されたサーバ・アプリケーション、複製されたサーバ・グループ、オブジェクト状態管理およびファクトリ・ベース・ルーティングを使用して、Production サンプル・アプリケーションをスケーリングする方法について説明します。

サーバ・アプリケーションの複製

サーバ・アプリケーションを複製すると、以下の処理が可能になります。

- クライアント・アプリケーションからそのサーバ・アプリケーションに着信する要求の負荷のバランシングを行う方法が得られます。要求がサーバ・グループの Tuxedo ドメインで受信されると、BEA Tuxedo システムは、グループ内で負荷が最も低いサーバ・アプリケーションに要求をルーティングします。
- サーバ・マシン上で実行する任意のサーバ・アプリケーション・プロセスのコピー数を指定できます。コピー数によって、クライアント・アプリケーションからの要求を Tuxedo ドメインが並列処理できるレベルが決まります。
- サーバ・アプリケーションの 1 つが処理を停止した場合でも、有効なフェイルオーバー保護機能が用意されます。

Production サンプル・アプリケーションでは、サーバ・アプリケーションは次の方法で複製されます。

- University サーバ・アプリケーション、ATMI Teller アプリケーション、および University データベースのサーバ・アプリケーションは、`ORA_GRP` グループ内で複製されます。

- Billing サーバ・アプリケーションは、APP_GRP グループ内で複製されます。

図 7-1 では、複製された ORA_GRP および APP_GRP サーバ・グループを示しています。

図 7-1Production サンプル・アプリケーションの複製されたサーバ・グループ

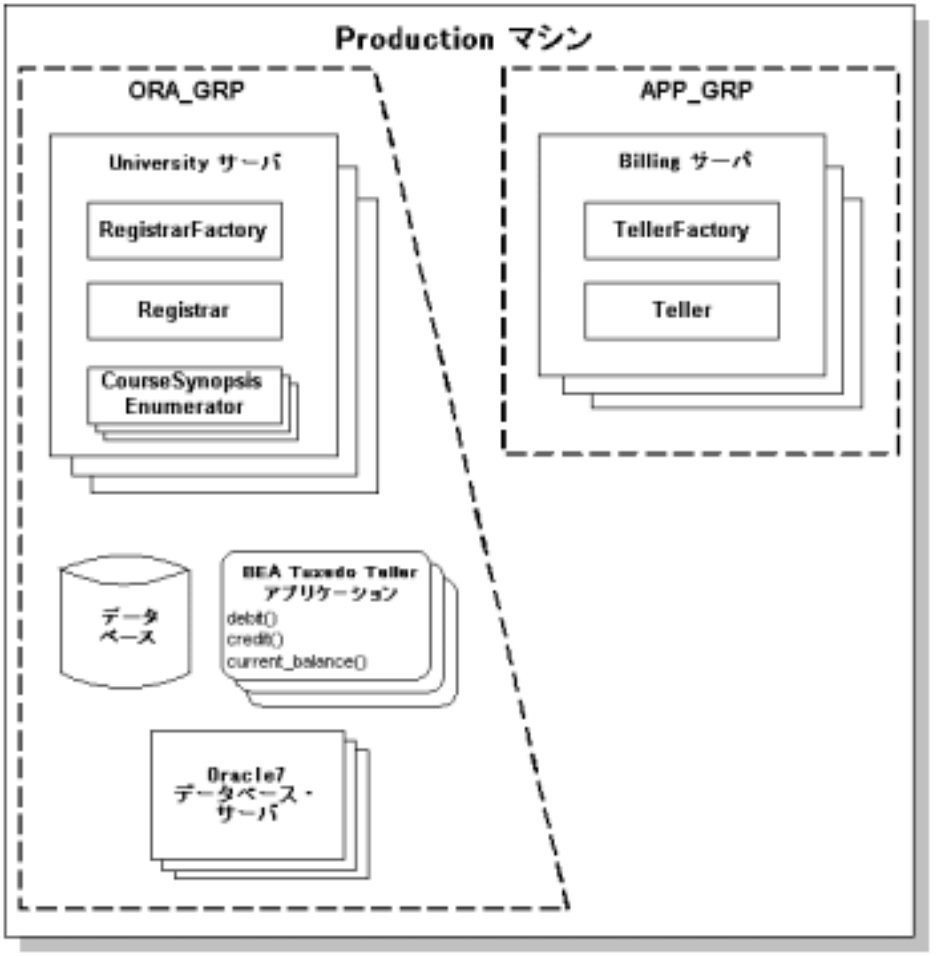


図 7-1 では、次の点に注意してください。

- RegistrarFactory、Registrar、TellerFactory、または Teller オブジェクトのインスタンスは、1 つのサーバ・アプリケーション・プロセス内に 1 つしか存在できません。
- CourseSynopsisEnumerator オブジェクトは、1 つのサーバ・アプリケーション・プロセス内に任意の数だけ存在できます。

サーバ・グループの複製

サーバ・グループは、既存の CORBA アプリケーションにサーバ・マシンを追加できる、BEA Tuxedo ソフトウェアの機能です。サーバ・グループを複製すると、次の処理が可能になります。

- CORBA アプリケーションの処理にかかる負荷を複数のサーバ・マシンに分散できます。
- ファクトリ・ベース・ルーティングを使用して、クライアント・アプリケーションからの要求を特定のサーバ・マシンに送信できます。

サーバ・グループのコンフィギュレーションおよび複製方法は、UBBCONFIG ファイルで指定します。

図 7-2 では、2 番目のサーバ・マシンで複製された Production サンプル・アプリケーションのサーバ・グループを示しています。Production サンプル・アプリケーションの UBBCONFIG ファイルでは、複製されたサーバ・グループは、ORA_GRP2 および APP_GRP2 として定義されています。

図 7-2 サーバ・マシン間でのサーバ・グループの複製

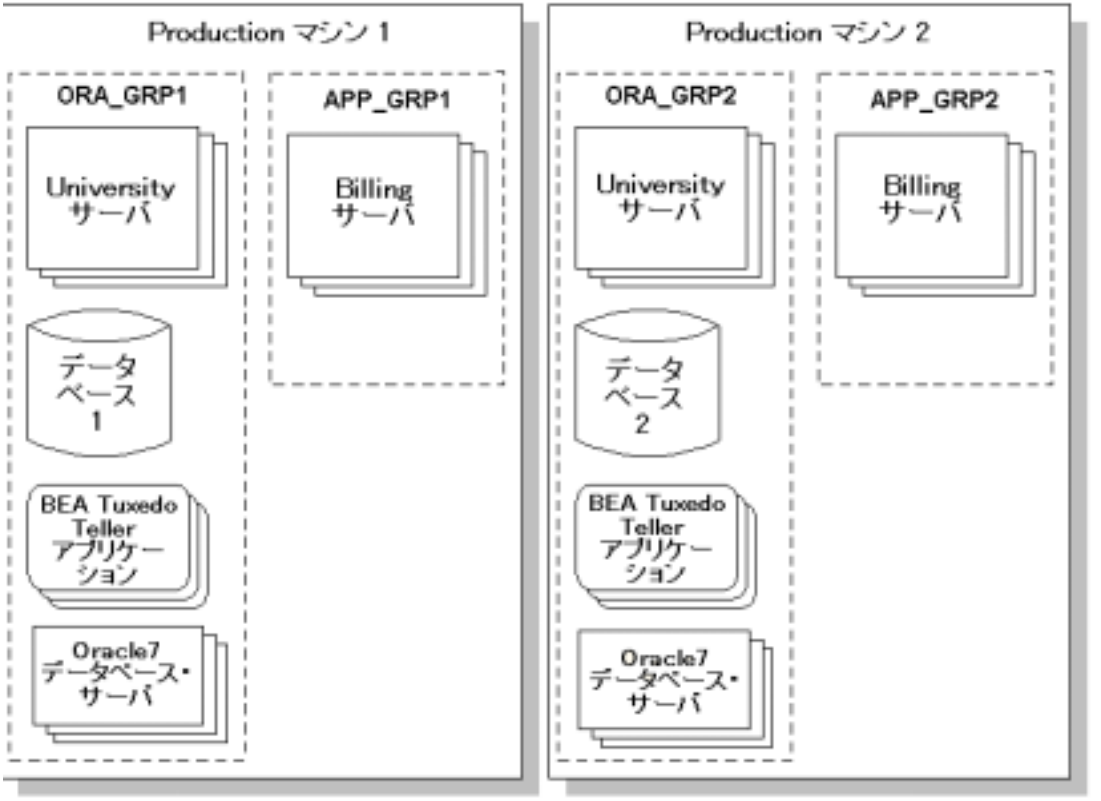


図 7-2 では、Production マシン 1 および 2 のサーバ・グループの内容で異なるのはデータベースのみです。University データベースは、2 つのデータベースに分割されています。Production マシン 1 のデータベースには、ID が 100001 ～ 100005 の学生の学生情報および口座情報が格納されています。Production マシン 2 のデータベースには、ID が 100006 ～ 100010 の学生の学生情報および口座情報が格納されています。

状態を持たないオブジェクト・モデルの使用方法

スケーラビリティを向上させるには、Production サンプル・アプリケーションで、Registrar および Teller オブジェクトが method 活性化方針を持つようにコンフィギュレーションします。method 活性化方針によって、オブジェクトの振る舞いは次のように変化します。

- オブジェクトが呼び出されると、常に適切なサーバ・グループで Tuxedo ドメインによってインスタンス化されます。
- 呼び出しが完了すると、オブジェクトは Tuxedo ドメインによって非活性化されます。

Basic から Production までのサンプル・アプリケーションの Registrar オブジェクトには、process という活性化方針があります。Registrar オブジェクトのクライアント・アプリケーションの要求は、すべてサーバ・マシンのメモリにある同じオブジェクトのインスタンスに送られます。この設計は小規模なデプロイメントに適しています。ただし、クライアント・アプリケーションからの要求が増加すると、Registrar オブジェクトの要求はキューに登録されるようになり、応答時間が遅くなります。

これに対し、Registrar および Teller オブジェクトに method 活性化方針があり、これらのオブジェクトを管理するサーバ・アプリケーションを複製することにより、Registrar および Teller オブジェクトはクライアント・アプリケーションからの複数の要求を並列処理できるようになります。唯一の制限は、Registrar および Teller オブジェクトのインスタンス化に利用可能なサーバ・アプリケーション・プロセスの数です。

CORBA アプリケーションでは、複製済みの各サーバ・アプリケーション・プロセスで Registrar および Teller オブジェクトのコピーをインスタンス化するために、Registrar および Teller オブジェクトには、一意なオブジェクト ID (OID) が割り当てられています。一意な OID は、これらのオブジェクトを作成するファクトリで割り当てられます。一意なオブジェクト ID の生成についての詳細は、『BEA Tuxedo CORBA サーバ・アプリケーションの開発方法』を参照してください。

ファクトリ・ベース・ルーティングの使用方法

ファクトリ・ベース・ルーティングは、クライアント・アプリケーションからの要求を特定のサーバ・グループに送信可能にする CORBA の機能です。ファクトリ・ベース・ルーティングを使用すると、CORBA アプリケーションの処理にかかる負荷を複数のサーバ・マシンに分散できます。Production サンプル・アプリケーションでは、ファクトリ・ベース・ルーティングを次のように使用します。

- クライアント・アプリケーションから `Registrar` オブジェクトへの要求は、学生 ID に基づいてルーティングされます。ID が 100001 ～ 100005 の学生からの要求は、Production マシン 1 にルーティングされます。ID が 100006 ～ 100010 の学生からの要求は、Production マシン 2 にルーティングされます。
- `Registrar` オブジェクトから `Teller` オブジェクトへの要求は、口座番号に基づいてルーティングされます。200010 から 200014 までの口座番号への支払い要求は、Production マシン 1 にルーティングされます。200015 から 200019 までの口座番号への支払い要求は、Production マシン 2 にルーティングされます。

ファクトリ・ベース・ルーティングの設定についての詳細は、『BEA Tuxedo CORBA サーバ・アプリケーションの開発方法』を参照してください。

Production サンプル・アプリケーションの開発プロセス

ここでは、CORBA アプリケーションのスケーリングに必要な開発プロセスについて説明します。以下の手順は、第 3 章 「Basic サンプル・アプリケーション」で概説した開発手順の追加手順です。

注記　この節に記載されている手順はすでに完了しており、Production サンプル・アプリケーションに組み込まれています。

OMG IDL

開発プロセスでは、ファクトリ・ベース・ルーティングをサポートするために、Object Management Group (OMG) インターフェイス定義言語 (IDL) の次のオペレーションの定義を変更します。

- 学生 ID を要求する、RegistrarFactory オブジェクトの find_registrar() オペレーション
- 口座番号を要求する、TellerFactory オブジェクトの find_teller() オペレーション

ファクトリ・ベース・ルーティングのインプリメントについての詳細は、『BEA Tuxedo CORBA サーバ・アプリケーションの開発方法』を参照してください。

クライアント・アプリケーション

開発プロセスでは、Registrar オブジェクトの作成時に STU_ID 値を指定します。STU_ID 値には、クライアント・アプリケーションからの要求のルーティング先となるサーバ・グループを定義します。

Production サンプル・アプリケーションでは、クライアント・アプリケーションの場合と同じ方法で、University サーバ・アプリケーションで Teller オブジェクトが作成されます。そのため、Teller オブジェクトの作成時に ACT_NUM 値を指定する必要があります。

サーバ・アプリケーション

開発プロセスでは、RegistrarFactory と TellerFactory オブジェクトの TP::create_object_reference() オペレーションの呼び出しを変更して、ルーティング基準を指定する NVlist を組み込む必要があります。TP::create_object_reference() オペレーションの criteria パラメータには、ファクトリ・ベース・ルーティングで使用する名前付き値のリストを次のように指定します。

- Production サンプル・アプリケーションの RegistrarFactory オブジェクトは、criteria の値を STD_ID として指定します。
- Production サンプル・アプリケーションの TellerFactory オブジェクトは、criteria の値を ACT_NUM として指定します。

criteria パラメータの値は、UBBCONFIG ファイルの ROUTING セクションで指定したルーティング基準名、フィールド、およびフィールド型と正確に一致しなければなりません。

ファクトリでのファクトリ・ベース・ルーティングのインプリメントについての詳細は、『BEA Tuxedo CORBA サーバ・アプリケーションの開発方法』を参照してください。

UBBCONFIG ファイル

CORBA アプリケーションにおけるスケーラビリティの向上には、UBBCONFIG ファイルが重要な役割を果たします。ここでは、Production サンプル・アプリケーションの UBBCONFIG ファイルを変更して次の処理を行う方法について説明します。

- サーバ・アプリケーション・プロセスおよびサーバ・グループの複製
- ファクトリ・ベース・ルーティングのインプリメント

サーバ・アプリケーション・プロセスおよびサーバ・グループの複製

開発プロセスでは、以下の方法で UBBCONFIG ファイルを変更して、サーバ・アプリケーション・プロセスおよびサーバ・グループの複製をコンフィギュレーションします。

1. UBBCONFIG ファイルの GROUPS セクションに、コンフィギュレーションするグループの名前を指定します。Production サンプル・アプリケーションには、APP_GRP1、APP_GRP2、ORA_GRP1、および ORA_GRP2 の 4 つのサーバ・グループがあります。
2. UBBCONFIG ファイルの SERVERS セクションに、複製するサーバ・アプリケーション・プロセスに関する次の情報を入力します。

- サーバ・アプリケーションの名前。
- GROUP パラメータ。サーバ・アプリケーション・プロセスが属するサーバ・グループの名前を指定します。複数のグループに関係するサーバ・プロセスを複製する場合は、各グループに 1 つずつサーバ・プロセスを指定します。
- SRVID パラメータ。サーバ・マシンの一意な管理 ID を指定します。
- MIN パラメータ。CORBA アプリケーションの起動時に開始する、サーバ・アプリケーション・プロセスのインスタンス数を指定します。開始するサーバ・アプリケーション・プロセスは最低 2 つ必要です。
- MAX パラメータ。同時に実行できるサーバ・アプリケーション・プロセスの最大数を指定します。指定可能なサーバ・アプリケーション・プロセスの数は最大 5 つです。

MIN および MAX パラメータによって、任意のオブジェクトで任意のサーバ・アプリケーションが要求をどの程度まで並列処理できるかを指定します。実行時には、必要に応じて、システム管理者がリソースのボトルネックを調べて、新しいサーバ・プロセスを起動することができます。このように、アプリケーションのスケーリングは、システム管理者が行います。

次に、Production サンプル・アプリケーションの UBBCONFIG ファイルの GROUPS および SERVERS セクションから引用した行を例示します。

```
*GROUPS
    APP_GRP1
        LMID      = SITE1
        GRPNO     = 2
        TMSNAME   = TMS
    APP_GRP2
        LMID      = SITE1
        GRPNO     = 3
        TMSNAME   = TMS
    ORA_GRP1
        LMID      = SITE1
        GRPNO     = 4
        OPENINFO  = "ORACLE_XA:Oracle_XA+Acc=P/scott/tiger+SesTm=100+LogDir
                     =.+MaxCur=5"
        CLOSEINFO = ""
        TMSNAME   = "TMS_ORA"
    ORA_GRP2
        LMID      = SITE1
```

```
        GRPNO     = 5
        OPENINFO  = "ORACLE_XA:Oracle_XA+Acc=P/scott/tiger+SesTm=100+LogDir
                    =.+MaxCur=5"
        CLOSEINFO = ""
        TMSNAME   = "TMS_ORA"


*SERVERS
    # デフォルトでは、各サーバのインスタンスを 2 つ活性化
    # して、管理者が各サーバのインスタンスを 5 つ
    # まで活性化できるようにしている
    DEFAULT:
        MIN     = 2
        MAX     = 5
    tellp_server
        SRVGRP  = ORA_GRP1
        SRVID   = 10
        RESTART = N
    tellp_server
        SRVGRP  = ORA_GRP2
        SRVID   = 10
        RESTART = N
    billp_server
        SRVGRP  = APP_GRP1
        SRVID   = 10
        RESTART = N
    billp_server
        SRVGRP  = APP_GRP2
        SRVID   = 10
        RESTART = N
    univp_server
        SRVGRP  = ORA_GRP1
        SRVID   = 20
        RESTART = N
    univp_server
        SRVGRP  = ORA_GRP2
        SRVID   = 20
        RESTART = N
```

ファクトリ・ベース・ルーティングのインプリメント

ファクトリ・ベース・ルーティングを使用できるようにするには、各インターフェイスについて、UBBCONFIG ファイルに次の情報を定義する必要があります。

- ルーティング基準のデータの詳細

■ 各基準ごとの、特定のサーバ・グループへルーティングするための値

開発プロセスでは、UBBCONFIG ファイルを以下のように変更します。

1. INTERFACES セクションには、ファクトリ・ベース・ルーティングを有効にするインターフェイスの名前の一覧が示されます。このセクションで、各インターフェイスのルーティング値を指定します。ルーティング値は、FACTORYROUTING 識別子で指定します。

   次に、Production サンプル・アプリケーションの Registrar および Teller オブジェクトの FACTORYROUTING 識別子の例を示します。

   ```
   INTERFACES
       "IDL:beasys.com/UniversityP/Registrar:1.0"
           FACTORYROUTING = STU_ID
       "IDL:beasys.com/BillingP/Teller:1.0"
           FACTORYROUTING = ACT_NUM
   ```

2. ROUTING セクションでは、ルーティング値ごとに以下のデータを指定します。

   - TYPE パラメータ。ルーティングの種類を指定します。Production サンプル・アプリケーションでは、ルーティングの種類はファクトリ・ベース・ルーティングです。したがって、このパラメータは、FACTORY と定義します。
   - FIELD パラメータ。ファクトリがルーティング値に挿入する名前を指定します。Production サンプル・アプリケーションでは、フィールドのパラメータは student_id および account_number です。
   - FIELDTYPE パラメータ。ルーティング値のデータ型を指定します。Production サンプル・アプリケーションでは、STU_ID および ACT_NUM のフィールド・タイプは long です。
   - RANGES パラメータ。各グループにルーティングされる値を指定します。

   次に、Production サンプル・アプリケーションで使用する UBBCONFIG ファイルの ROUTING セクションの例を示します。

```
*ROUTING
    STU_ID
        FIELD     = "student_id"
        TYPE      = FACTORY
        FIELDTYPE = LONG
```

```
    RANGES    = "100001-100005:ORA_GRP1,100006-100010:ORA_GRP2"
ACT_NUM
    FIELD     = "account_number"
    TYPE      = FACTORY
    FIELDTYPE = LONG
    RANGES    = "200010-200014:APP_GRP1,200015-200019:APP_GRP2"
```

上記の例では、ID が 100001 ～ 100005 の学生の Registrar オブジェクトは ORA_GRP1 でインスタンス化され、ID が 100006 ～ 100010 の学生のオブジェクトは ORA_GRP2 でインスタンス化されることを示しています。同様に、200010 ～ 200014 の口座番号の Teller オブジェクトは APP_GRP1 でインスタンス化され、200015 ～ 200019 の口座番号のオブジェクトは APP_GRP2 でインスタンス化されることを示しています。

3. UBBCONFIG ファイルの ROUTING セクションにある RANGES 識別子で指定されたグループを識別およびコンフィギュレーションする必要があります。たとえば、Production サンプル・アプリケーションでは、ORA_GRP1、ORA_GRP2、APP_GRP1、および APP_GRP2 の 4 つのグループが指定されています。これらのグループをコンフィギュレーションし、グループが実行されるマシンを識別する必要があります。

注記　GROUPS セクションのサーバ・グループの名前と ROUTING セクションで指定したグループの名前は、正しく一致している必要があります。

ICF ファイル

開発プロセスでは、Registrar、RegistrarFactory、Teller、および TellerFactory オブジェクトの活性化方針を process から method に変更する必要があります。CORBA オブジェクトの活性化方針およびトランザクション方針の定義についての詳細は、『BEA Tuxedo CORBA サーバ・アプリケーションの開発方法』を参照してください。

Production サンプル・アプリケーションのビルド

Production サンプル・アプリケーションをビルドするには、次の手順に従います。

1. Production サンプル・アプリケーションのファイルを作業ディレクトリにコピーします。
2. Production サンプル・アプリケーションのファイルの保護を変更します。
3. 環境変数を設定します。
4. University データベースを初期化します。
5. UBBCONFIG ファイルをロードします。
6. トランザクション・ログを作成します。
7. クライアントおよびサーバ・サンプル・アプリケーションをビルドします。

以降の節では、上記の各手順について説明します。

注記　Production サンプル・アプリケーションをビルドまたは実行する前に、第 2 章 「環境設定」の手順を実行しておく必要があります。

Production サンプル・アプリケーションのファイルを作業ディレクトリにコピーする

Production サンプル・アプリケーションの各ファイルは、次のディレクトリにあります。

Windows 2000

drive:*TUXDIR*\samples\corba\university\production

UNIX

```
/usr/TUXDIR/samples/corba/university/production
```

また、utils ディレクトリも作業ディレクトリにコピーする必要があります。utils ディレクトリには、ログ、トレース、および University データベースへのアクセスを設定するファイルが格納されています。

Production サンプル・アプリケーションを作成するには、表 7-1 のファイルを使用します。

表 7-1 Production サンプル・アプリケーションに含まれるファイル

ファイル	説明
billp.idl	Teller および TellerFactory インターフェイスを宣言する OMG IDL。
univp.idl	CourseSynopsisEnumerator、Registrar、および RegistrarFactory インターフェイスを宣言する OMG IDL。
billps.cpp	Production サンプル・アプリケーションの Billing サーバ・アプリケーション用 C++ ソース・コード。
univps.cpp	Production サンプル・アプリケーションの University サーバ・アプリケーション用 C++ ソース・コード。
billp__i.h billp_i.cpp	Teller および TellerFactory インターフェイスのメソッド・インプリメンテーション用 C++ ソース・コード。
univp_i.h univp_i.cpp	CourseSynopsisEnumerator、Registrar、および RegistrarFactory インターフェイスのメソッド・インプリメンテーション用 C++ ソース・コード。
univpc.cpp	Production サンプル・アプリケーションの CORBA C++ クライアント・アプリケーション用 C++ ソース・コード。

表 7-1 Production サンプル・アプリケーションに含まれるファイル (続き)

ファイル	説明
frmBrowser.frm	Production サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic ソース・コード。
university.vbp	Production サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic プロジェクト・ファイル。
University.vbw	Production サンプル・アプリケーションの ActiveX クライアント・アプリケーション用 Visual Basic ワークスペース・ファイル。
modPublicDeclarations.bas	サンプル・アプリケーションで使用される変数の宣言が記述された Visual Basic ファイル。
frmTracing.frm frmTracing.frx	ActiveX クライアント・アプリケーションにトレース機能を提供するファイル。
frmLogon.frm	ActiveX クライアント・アプリケーションに対してセキュリティ・ログオンを実行する Visual Basic ファイル。
univPApplet.java	Production サンプル・アプリケーションの CORBA Java クライアント・アプリケーション用 Java ソース・コード。
univp_utils.h univp_utils.cpp	CORBA C++ クライアント・アプリケーションのデータベース・アクセス関数を定義するファイル。
univp.icf	Production サンプル・アプリケーションの University サーバ・アプリケーション用インプリメンテーション・コンフィギュレーション・ファイル (ICF)。
billp.icf	Production サンプル・アプリケーションの Billing サーバ・アプリケーション用 ICF ファイル。

表 7-1 Production サンプル・アプリケーションに含まれるファイル (続き)

ファイル	説明
tellw_flds、tellw_u.c、tellw_c.h、tellws.ec	ATMI アプリケーション Teller のファイル。
setenvp.sh	Production サンプル・アプリケーションのビルドおよび実行に必要な環境変数を設定する UNIX スクリプト。
setenvp.cmd	Production サンプル・アプリケーションのビルドおよび実行に必要な環境変数を設定する MS-DOS コマンド。
ubb_p.mk	UNIX オペレーティング・システム用の UBBCONFIG ファイル。
ubb_p.nt	Windows 2000 オペレーティング・システム用の UBBCONFIG ファイル。
makefilep.mk	UNIX オペレーティング・システムでの Production サンプル・アプリケーション用の makefile。
makefilep.nt	Windows 2000 オペレーティング・システムでの Production サンプル・アプリケーション用の makefile。
log.cpp、log.h、log_client.cpp、log_server.cpp	サンプル・アプリケーションにログ機能とトレース機能を提供するクライアント・アプリケーションとサーバ・アプリケーションのファイル。これらのファイルは、\utils ディレクトリにあります。
oradbconn.cpp、oranoconn.cpp	Oracle SQL データベース・インスタンスへのアクセスを提供するファイル。これらのファイルは、\utils ディレクトリにあります。
samplesdb.cpp、samplesdb.h	サンプル・アプリケーションでのデータベース例外に出力関数を提供するファイル。これらのファイルは、\utils ディレクトリにあります。

表 7-1 Production サンプル・アプリケーションに含まれるファイル (続き)

ファイル	説明
unique_id.cpp、unique_id.h	サンプル・アプリケーションの C++ Unique ID クラスのルーチン。これらのファイルは、\utils ディレクトリにあります。
samplesdbsql.h、samplesdbsql.pc	SQL データベースへのアクセスをインプリメントする C++ クラスのメソッド。これらのファイルは、\utils ディレクトリにあります。
university.sql	University データベース用の SQL。このファイルは、\utils ディレクトリにあります。

Production サンプル・アプリケーションのファイル保護の属性を変更する

BEA Tuxedo ソフトウェアのインストール時には、サンプル・アプリケーションは読み取り専用に設定されています。Production サンプル・アプリケーションのファイルを編集または作成するには、次のように作業ディレクトリにコピーしたファイル保護の属性を変更する必要があります。

Windows 2000

```
prompt>attrib -r drive:\workdirectory\*.*
```

UNIX

```
prompt>chmod u+rw /workdirectory/*.*
```

環境変数を設定する

次のコマンドを使用して、Production サンプル・アプリケーションのクライアント・アプリケーションとサーバ・アプリケーションのビルドに使用する環境変数を設定します。

Windows 2000

```
prompt>setenvp
```

UNIX

```
prompt>/bin/ksh
prompt>. ./setenvp.sh
```

University データベースを初期化する

次のコマンドを使用して、Production サンプル・アプリケーションで使用する University データベースを初期化します。

Windows 2000

```
prompt>nmake -f makefilep.nt initdb
```

UNIX

```
prompt>make -f makefilep.mk initdb
```

UBBCONFIG ファイルをロードする

次のコマンドを使用して、UBBCONFIG ファイルをロードします。

Windows 2000

```
prompt>tmloadcf -y ubb_p.nt
```

UNIX

```
prompt>tmloadcf -y ubb_p.mk
```

UBBCONFIG ファイルの作成プロセスでは、アプリケーション・パスワードの入力が求められます。このパスワードは、クライアント・アプリケーションへのログオンに使用されます。パスワードを入力して Enter キーを押します。その際、パスワードを再入力してパスワードの確認を求めるメッセージが表示されます。

トランザクション・ログを作成する

トランザクション・ログには、CORBA アプリケーションでのトランザクション処理が記録されます。開発プロセスでは、UBBCONFIG ファイルの TLOGDEVICE パラメータでトランザクション・ログの場所を定義する必要があります。Production サンプル・アプリケーションの場合、トランザクション・ログは作業ディレクトリに格納されています。

Production サンプル・アプリケーションのトランザクション・ログを開くには、次の手順に従います。

1. 次のコマンドを入力して、会話型の管理インターフェイスを起動します。

   ```
   tmadmin
   ```

2. 次のコマンドを入力して、トランザクション・ログを作成します。

   ```
   crdl -b blocks -z directorypath
   crlog -m SITE1
   ```

   この例では、

   blocks にトランザクション・ログに割り当てるブロック数を指定し、*directorypath* にトランザクション・ログの場所を指定します。*directorypath* オプションは、UBBCONFIG ファイルの TLOGDEVICE パラメータで指定した場所と一致しなければなりません。Windows 2000 でのコマンドの例を次に示します。

   ```
   crdl -b 500 -z c:\mysamples\university\production\TLOG
   ```

3. 「q」を入力して、会話型の管理インターフェイスを終了します。

Production サンプル・アプリケーションのコンパイル

開発プロセスでは、buildobjclient および buildobjserver コマンドを使用して、クライアント・アプリケーションとサーバ・アプリケーションをビルドします。ただし、Production サンプル・アプリケーションの場合は、この手順は不要です。Production サンプル・アプリケーションのディレクトリには、makefile が格納されています。この makefile により、クライアントとサーバ・サンプル・アプリケーションがビルドされます。

Production サンプル・アプリケーションの CORBA C++ クライアント・アプリケーションとサーバ・アプリケーションをビルドするには、次のコマンドを使用します。

Windows 2000

```
prompt>nmake -f makefilep.nt
```

UNIX

```
prompt>make -f makefilep.mk
```

CORBA Java クライアント・アプリケーションをビルドするには、次のコマンドを使用します。

Windows 2000

```
prompt>nmake -f makefilep.nt javaclient
```

UNIX

```
prompt>make -f makefilep.mk javaclient
```

ActiveX クライアント・アプリケーションの起動については、「ActiveX クライアント・アプリケーションの起動」を参照してください。

buildobjclient および buildobjserver コマンドの詳細については、『BEA Tuxedo コマンド・リファレンス』を参照してください。

Production サンプル・アプリケーションの実行

Production サンプル・アプリケーションを実行するには、次の手順に従います。

1. サーバ・アプリケーションを起動します。
2. 1 つまたは複数のクライアント・アプリケーションを起動します。

以降の節では、上記の各手順の詳細について説明します。

サーバ・アプリケーションの起動

Production サンプル・アプリケーションでシステムおよびサンプル・アプリケーションのサーバ・アプリケーションを起動するには、次のコマンドを入力します。

```
prompt>tmboot -y
```

このコマンドを入力すると、次のサーバ・プロセスが開始されます。

- `TMSYSEVT`

  BEA Tuxedo システムの EventBroker。

- `TMFFNAME`

  NameManager サービスや FactoryFinder サービスなどのトランザクション管理サービス。

- `TMIFSRVR`

  インターフェイス・リポジトリ・サーバ・プロセス。これは、ActiveX クライアント・アプリケーションでのみ使用されます。

- `univp_server`

  University サーバ・アプリケーションの 4 つのプロセス。

- `tellp_server`

  ATMI アプリケーション Teller の 4 つのプロセス。

- `billp_server`

  Billing サーバ・アプリケーションの 4 つのプロセス。

- `ISL`

  IIOP リスナ / ハンドラ・プロセス。

ほかのサンプル・アプリケーションを使用するには、次のコマンドを入力して、システムおよびサンプル・アプリケーションのサーバ・プロセスを停止します。

```
prompt>tmshutdown
```

CORBA C++ クライアント・アプリケーションの起動

Production サンプル・アプリケーションの CORBA C++ クライアント・アプリケーションを起動するには、次の手順に従います。

1. MS-DOS プロンプトで次のコマンドを入力します。

   ```
   prompt>univp_client
   ```

2. 画面に「`Enter student id:`」と表示されたら、100001 ～ 100010 の任意の数値を入力します。
3. Enter キーを押します。
4. 画面に「`Enter domain password:`」と表示されたら、`UBBCONFIG` ファイルをロードしたときに定義したパスワードを入力します。
5. Enter キーを押します。

注記　Production サンプル・アプリケーションでは、Production サンプル・アプリケーションと Wrapper サンプル・アプリケーションの各 CORBA C++ クライアント・アプリケーションは同じ方法で動作します。

CORBA Java クライアント・アプリケーションの起動

Production サンプル・アプリケーションの CORBA Java クライアント・アプリケーションを実行するには、以下の手順に従います。

1. UnivPApplet.html ファイル内の次の行を変更します。

```
 code="UnivPApplet.class"
codebase=.
```

上記の行を次のように変更します。

```
 code="UnivPApplet"
archive="UnivPApplet.jar,m3envobj.jar"
```

2. 変更した UnivPApplet.html ファイルを Web サーバのソース・ディレクトリにコピーします。ディレクトリは、Web サーバ製品によって異なります。

3. makefile を実行して Production サンプル・アプリケーションをビルドした後、次のように UnivPApplet.jar ファイルを作成します。

   a. サンプル・アプリケーションをビルドしたディレクトリの下に tmp ディレクトリを作成します。UniversityP サブディレクトリとその中に格納されているクラス・ファイルを tmp ディレクトリにコピーします。

      makefile によって生成された、Production サンプル・アプリケーション・ディレクトリにあるクラス・ファイルを tmp ディレクトリにコピーします。ディレクトリを tmp ディレクトリに設定 (cd) し、次のどちらかのコマンドを入力して、すべての Production サンプル・アプリケーション・クラスを含んだ .JAR ファイルを作成します。

      jar -cf ..\UnivPApplet.jar *.* (Microsoft Windows 2000 システム)
      jar -cf ../UnivPApplet.jar * (UNIX システム)

4. 前の手順で作成した UnivPApplet.jar ファイルを Web サーバのソース・ディレクトリにコピーします。ディレクトリ名は、Web サーバ製品によって異なります。
5. 適切なサブディレクトリ (Microsoft Windows 2000 システムの場合は %TUXDIR%\udataobj\java 、UNIX システムの場合は ${TUXDIR}/udataobj/java) から、m3envobj.jar ファイルを Web サーバのソース・ディレクトリにコピーします。
6. Security サーバ・アプリケーションが動作していることを確認してから、Web ブラウザを起動し、Web サーバが動作しているノードを参照します。

注記　Microsoft Windows 2000 システムの場合、ノード名はすべて大文字にする必要があります。たとえば、UBBCONFIG ファイルおよび UnivPApplet.html ファイルでノードを SERVER に指定した場合、ブラウザは http://SERVER/UnivPApplet.html に設定します。

7. student ID フィールドに 100001 ～ 100010 の数値を入力します。
8. Domain Password フィールドに、UBBCONFIG ファイルをロードしたときに定義したパスワードを入力します。
9. [Logon] ボタンをダブルクリックします。

注記　Production サンプル・アプリケーションでは、Production サンプル・アプリケーションと Wrapper サンプル・アプリケーションの各 CORBA Java クライアント・アプリケーションは同じ方法で動作します。

ActiveX クライアント・アプリケーションの起動

注記　University サンプル・アプリケーションでは、インターフェイス・リポジトリに CORBA インターフェイスの OMG IDL をロードする作業は makefile によって自動化されています。

ActiveX クライアント・アプリケーションを起動するには、Application Builder を使用して CORBA インターフェイスの ActiveX バインディングを作成する必要があります。

CORBA インターフェイスの ActiveX バインディングを作成するには、次の手順に従います。

1. BEA Tuxedo プログラム・グループで [BEA Application Builder] アイコンをクリックします。

   [IIOP Listener] ウィンドウが表示されます。

2. [IIOP Listener] ウィンドウに、`UBBCONFIG` ファイルの `ISL` パラメータで指定したホスト名とポート番号を入力します。ホスト名とポート番号は、`UBBCONFIG` ファイルで大文字で指定した内容と正確に一致するように入力します。

   [Logon] ウィンドウが表示されます。

3. `UBBCONFIG` ファイルをロードしたときに定義したユーザ名とパスワードについて、100001 ～ 100010 の学生 ID を [Logon] ウィンドウに入力します。

   [Application Builder] ウィンドウが表示されます。インターフェイス・リポジトリにロードされた CORBA インターフェイスがすべて、Application Builder の [Services] ウィンドウに表示されます。

4. [Services] ウィンドウで UniversityP フォルダを強調表示して [Workstation Views] ウィンドウにドラッグします。または、[Services] ウィンドウから UniversityP folder フォルダをコピーして [Workstation Views] ウィンドウに貼り付けます。

   確認のウィンドウが表示されます。

5. [Create] をクリックすると、Production サンプル・アプリケーションで CORBA インターフェイスの ActiveX バインディングが作成されます。

   Application Builder では、以下のものが作成されます。

   - CORBA インターフェイスに対するバインディング。バインディングの名前の形式は、`DI`*`modulename_interfacename`* です。たとえば、`Registrar` インターフェイスに対するバインディングの名前は、`DIUniversityP_Registrar` となります。

- 型ライブラリ。デフォルトでは、型ライブラリは `\TUXDIR\TypeLibraries` に格納されています。

  型ライブラリのファイル名の形式は、`DImodulename_interfacename.tlb` です。

- CORBA インターフェイスの Windows システム・レジストリ・エントリ。このエントリには、オブジェクト型ごとに一意のプログラム ID が入っています。

ActiveX クライアント・アプリケーションを実行するには、以下の手順に従います。

1. Visual Basic で `University.vbw` ファイルを開きます。
2. [Run] メニューの [Start] をクリックします。

   [IIOP Listener] ウィンドウが表示されます。
3. [IIOP Listener] ウィンドウに、`UBBCONFIG` ファイルの `ISL` パラメータで指定したホスト名とポート番号を入力します。ホスト名とポート番号は、`UBBCONFIG` ファイルで大文字で指定した内容と正確に一致するように入力します。

   [Logon] ウィンドウが表示されます。
4. `UBBCONFIG` ファイルをロードしたときに定義したユーザ名とパスワードについて、100001 ～ 100010 の学生 ID を [Logon] ウィンドウに入力します。

注記　Production サンプル・アプリケーションでは、Production サンプル・アプリケーションと Wrapper サンプル・アプリケーションの各 ActiveX クライアント・アプリケーションは同じ方法で動作します。

Production サンプル・アプリケーションをさらにスケーリングする方法

Production サンプル・アプリケーションをさらにスケーリングするには、次の方法に従います。

- Production サンプル・アプリケーションのサーバ・グループを複数のマシンに複製します。

  新しいサーバ・グループ、そのサーバ・グループで実行するサーバ・アプリケーション・プロセス、およびサーバ・グループを実行するサーバ・マシンを指定するには、UBBCONFIG ファイルを変更する必要があります。

- ファクトリ・ベース・ルーティングのテーブルを修正します。

  たとえば、Production サンプル・アプリケーションの既存の 2 つのサーバ・グループにルーティングする代わりに、UBBCONFIG ファイルのルーティング規則を修正すると、アプリケーションを追加されたサーバ・グループの間でさらに分割できます。ルーティング・テーブルを修正する場合は、UBBCONFIG ファイルの情報と一致させる必要があります。

注記　データベースを使用する既存の CORBA アプリケーションに機能を追加する場合は、データベースがどのように設定されているかを考慮する必要があります。ファクトリ・ベース・ルーティングを使用している場合は、特に注意が必要です。たとえば、Production サンプル・アプリケーションが 6 台のマシン間に分散している場合は、UBBCONFIG ファイルのルーティング・テーブルに基づいて、適切に各マシンのデータベースを設定する必要があります。

章

A データベースの設定

University サンプル・アプリケーションでは、University データベースというデータベースを使用して、コース名やコース概要といった、サンプル・アプリケーションで使用されるデータをすべて格納します。University サンプル・アプリケーションをビルドおよび実行するには、このデータベースを事前にインストールおよび設定しておく必要があります。

この付録では、データベースの設定手順について説明します。手順の詳細については、ご使用のデータベースの製品マニュアルを参照してください。

データベースのサポート

BEA Tuxedo ソフトウェアのバージョン 8.0 に付属している University サンプル・アプリケーションは、Oracle のバージョン 7.3.3 以降で使用できます。表 A-1 に、各オペレーティング・システムでサポートされている Oracle のバージョンの一覧を示します。

表 A-1 データベースのサポート

オペレーティング・システム	サポートされている Oracle データベースのバージョン
Solaris SPARC バージョン 2.6	Sun SPARC 対応 Oracle バージョン 7.3.4 Enterprise Edition
Solaris SPARC バージョン 7,0	Solaris バージョン 2.7 対応 Oracle バージョン 7.3.4

表 A-1 データベースのサポート (続き)

オペレーティング・システム	サポートされている Oracle データベースのバージョン
HP-UX バージョン 10.20	HP-UX バージョン 10.20 対応 Oracle バージョン 7.3.3
HP-UX バージョン 11.0	HP-UX バージョン 11.0 対応 Oracle バージョン 8.0
IBM AIX バージョン 4.3.2	IBM AIX 対応 Oracle バージョン 8.0.4
Compaq Tru64	Compaq Tru64 対応 Oracle バージョン 7.3.3
Sequent バージョン 4.4.2	Sequent Dynix 対応 Oracle バージョン 8.0.4
SGI IRIX バージョン 6.5 IP27	SGI IRIX 対応 Oracle バージョン 8.0.4 Client
Windows NT/Intel バージョン 4.0	Windows NT 対応 Oracle バージョン 7.3.3. および Windows NT 対応 Oracle バージョン 8.0
Windows NT/Alpha バージョン 4.0	Windows NT 対応 Oracle バージョン 7.3.3.0 および Windows NT 対応 Oracle バージョン 8.0

Oracle データベースの設定手順

Oracle データベースは、次のように使用できます。

- ローカル・インスタンスとして
- リモート・インスタンスとして

ローカル・データベース・インスタンスの設定

Oracle データベースのローカル・インスタンスを使用している場合は、次の Oracle コンポーネントをインストールする必要があります。

- Programmer/2000 Pro*C/C++
- TCP/IP Adapter
- SQL*Net Client
- SQL*Plus
- Oracle 7 Server
- Oracle 7 Utilities
- SQL*Net Server

Oracle のインストール・プログラムによって作成されたデフォルトのデータベースを使用してください。Oracle データベース用の接続文字列およびデフォルトのユーザ ID とパスワードが必要です。この情報を取得する方法については、Oracle の製品マニュアルを参照してください。

Oracle ソフトウェアをインストールしたら、Oracle データベースのデーモンを起動します。通常、デーモンはマシンの起動プロセスの一部として起動します。また、XA リソース・マネージャを有効にして、Oracle データベースとサンプル・アプリケーション (`v$xatrans$`) との相互作用を XA リソース・マネージャで管理できるように権限を設定します。

次のコマンドを入力して、XA リソース・マネージャが University サンプル・アプリケーションと連携して機能するようにします。

```
SQL>grant select on v$xatrans$ to public;
```

注記　バージョン 8.0 の Oracle データベースを使用している場合にのみ、さらに次のコマンドを入力します。

```
SQL>grant select on dba_pending_transactions to user;
SQL>commit;
```

上記の中で、*user* には、Oracle データベースのデフォルトのユーザを指定します。

この手順についての詳細は、Oracle の製品マニュアルを参照してください。

Oracle データベースを初期化する手順については、各サンプル・アプリケーションのビルドに関する説明の中で記載されています。

リモート・データベース・インスタンスの設定

ほかのマシンで実行中のデータベースなど、Oracle データベースのリモート・インスタンスを使用している場合は、次の Oracle コンポーネントをインストールする必要があります。

- Programmer/2000 Pro*C/C++ (バージョン 2.2.3.0.0)
- TCP/IP Adapter (バージョン 2.3.3.0.0)
- SQL*Net Client (バージョン 2.3.3.0.0)
- SQL*Plus (バージョン 3.3.3.0.0)

Oracle データベースのリモート・インスタンスを使用するには、データベースのエイリアスを定義する必要があります。また、次の情報が必要になります。

- Oracle データベースのリモート・インスタンスを識別する文字列。
- Oracle データベースのリモート・インスタンスが置かれているマシンの名前。この名前は、マシンの `UBBCONFIG` ファイルで指定したホスト名と一致していなければなりません。
- Oracle データベースのリモート・インスタンスの `SID`。

Oracle データベースのリモート・インスタンスのエイリアスを定義するには、SQL *Net Easy Configuration Utility を使用します。この手順についての詳細は、Oracle の製品マニュアルを参照してください。

アクセスするデータベースでは、データベースとサンプル・アプリケーション (v$xatrans$) との相互作用を可能にする権限を持つ、有効な XA リソース・マネージャが必要です。

次のコマンドを入力して、XA リソース・マネージャが University サンプル・アプリケーションと連携して機能するようにします。

```
SQL>grant select on v$xatrans$ to public;
```

注記　バージョン 8.0 の Oracle データベースを使用している場合にのみ、さらに次のコマンドを入力します。

```
SQL>grant select on dba_pending_transactions to user;

SQL>commit;
```

上記の中で、*user* には、Oracle データベースのデフォルトのユーザを指定します。

Oracle データベースを初期化する手順については、各サンプル・アプリケーションのビルドに関する説明の中で記載されています。

索引

A

B

C

D

F

G

I

J

L

M

N

O

P

R

S

T

U

W

X

い

こ

さ

し

す

せ

そ

と

ひ

ふ

ま

め

ゆ

ら

ん