SYBASE®

XA インタフェース統合ガイド for CICS、Encina、TUXEDO

Adaptive Server® Enterprise

15.5

ドキュメント ID：DC30086-01-1550-01
改訂：2009 年 10 月

マニュアルの注文

マニュアルの注文を承ります。ご希望の方は、サイベース株式会社営業部または代理店までご連絡ください。マニュアルの変更は、弊社の定期的なソフトウェア・リリース時にのみ提供されます。

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

目次

はじめに

対象読者

このマニュアルは、次の担当者を対象としたリファレンス・マニュアルです。

- 分散トランザクション管理機能が備わった 1 つまたは複数の Adaptive Server が含まれる分散トランザクション処理 (DTP) 環境を設定するシステム管理者。トランザクションは、CICS、Encina、または TUXEDO のトランザクション・マネージャ (TM) システム内から DTP 環境にアクセスします。
- Embedded SQL™ または Client-Library™ を使用して 1 つまたは複数の Adaptive Servers に格納されているデータにアクセスするアプリケーション・プログラマ。

このマニュアルでは、以下のものに精通していることを前提としています。

- TM 操作環境
- Embedded SQL™
- Open Client™ Client-Library
- Adaptive Server® の管理

このマニュアルの内容

CICS、Encina、または TUXEDO の TM 内から 1 つまたは複数の Adaptive Server に格納されているデータにアクセスできるようにするために、環境の設定や、アプリケーションのコーディングを行うときにこのマニュアルを参照してください。

「第 1 章 概要」では、DTM XA インタフェースを既存の環境に完全に統合するために必要な手順について説明します。

「第 2 章 Sybase XA 環境」では、分散トランザクション処理とトランザクション管理のより大きなコンテキストに Sybase XA 環境を組み込むときに役立つ、基本的な情報を提供します。この章では、X/Open DTP モデルでの分散トランザクション処理について考察し、Sybase DTM XA インタフェースをこのモデルに適合させます。さらに、アプリケーションが TM から Adaptive Server のデータにアクセスできるようにするために、Sybase® XA 環境の各コンポーネントが相互に機能する仕組みについても説明してします。

「第 3 章 XA 環境の設定」では、アプリケーション、Sybase DTM XA インタフェース、1 つまたは複数の Adaptive Server、TM ソフトウェアを完全に統合するための、環境の設定方法について説明しています。

「第 4 章 アプリケーション・プログラミングのガイドライン」では、Embedded SQL アプリケーションや Client-Library アプリケーションを、Sybase XA 環境で定められている特定のコーディング制約に準拠させる方法について説明しています。

関連マニュアル

Adaptive Server® Enterprise には次のマニュアルが用意されています。必要に応じて参照してください。

- 使用しているプラットフォームの『リリース・ノート』– マニュアルには記載できなかった最新の情報が記載されています。

  このリリース・ノートの最新バージョン (英語版) を入手できます。製品の CD がリリースされた後で、製品またはマニュアルに関する重要な情報が追加されているかを確認するには、Sybase Product Manuals Web サイトを使用してください。

- 使用しているプラットフォームの『インストール・ガイド』 – すべての Adaptive Server および関連する Sybase 製品のインストール、アップグレード、設定の手順について説明しています。

- 『新機能ガイド』– Adaptive Server の新しい機能について説明しています。また、新しい機能をサポートするためのシステム変更や、既存のアプリケーションに影響を与える可能性がある変更についても説明しています。

- 『Active Messaging ユーザーズ・ガイド』– Active Messaging を使用して、Adaptive Server Enterprise データベースでトランザクション (データ変更) を取得し、外部アプリケーションにイベントとしてリアルタイムで渡す方法について説明しています。

- 『コンポーネント統合サービス・ユーザーズ・ガイド』– コンポーネント統合サービスを使用して、リモートの Sybase データベースおよび Sybase 以外のデータベースに接続する方法について説明しています。

- 使用しているプラットフォームの『設定ガイド』– 特定の設定作業の手順について説明しています。

- 『用語解説』– Adaptive Server マニュアルで使用されている技術用語について説明しています。

- 『Historical Server ユーザーズ・ガイド』– Historical Server を使用して、Adaptive Server のパフォーマンス情報を入手する方法について説明しています。

- 『Adaptive Server Enterprise における Java』– Adaptive Server データベースで Java クラスをデータ型、関数、ストアド・プロシージャとしてインストールして使用する方法について説明しています。

- 『Job Scheduler ユーザーズ・ガイド』– コマンド・ラインまたはグラフィカル・ユーザ・インタフェース (GUI) を使用して、ローカルまたはリモートの Adaptive Server でジョブのインストール、設定、作成、スケジュールを行う方法について説明しています。

- 『マイグレーション技術ガイド』− 別のバージョンの Adaptive Server にマイグレートするための方法とツールについて説明しています。
- 『Monitor Client Library プログラマーズ・ガイド』− Adaptive Server のパフォーマンス・データにアクセスする Monitor Client Library アプリケーションの記述方法について説明しています。
- 『Monitor Server ユーザーズ・ガイド』− Monitor Server を使用して、Adaptive Server のパフォーマンス統計を取得する方法について説明しています。
- 『モニタリング・テーブル・ダイヤグラム』− モニタリング・テーブルと、そのエンティティの関係をポスター形式で図解しています。フル・サイズのダイアグラムは印刷版だけで参照できます。コンパクト版は PDF 形式で参照できます。
- 『パフォーマンス&チューニング・シリーズ』− Adaptive Server で最高のパフォーマンスを実現するためのチューニング方法について説明しています。
  - 『基本』− Adaptive Server のパフォーマンスに関する問題の理解と調査の基本について説明しています。
  - 『統計的分析によるパフォーマンスの向上』− Adaptive Server で統計情報がどのように保存され、表示されるかについて説明しています。また、set statistics コマンドを使用して、サーバの統計情報を分析する方法について説明しています。
  - 『ロックと同時実行制御』− ロック・スキームを使用してパフォーマンスを向上させる方法と、同時実行性を最小限に抑えるようにインデックスを選択する方法について説明しています。
  - 『sp_sysmon による Adaptive Server の監視』− sp_sysmon を使用してパフォーマンスをモニタリングする方法について説明しています。
  - 『モニタリング・テーブル』− Adaptive Server のモニタリング・テーブルに統計情報や診断情報を問い合わせる方法について説明しています。
  - 『物理データベースのチューニング』− データの物理的配置、データに割り付けられた領域、テンポラリ・データベースの管理方法について説明しています。
  - 『クエリ処理と抽象プラン』− オプティマイザがクエリを処理する方法と、抽象プランを使用してオプティマイザのプランの一部を変更する方法について説明しています。
- 『クイック・リファレンス・ガイド』– コマンド、関数、システム・プロシージャ、拡張システム・プロシージャ、データ型、ユーティリティの名前と構文の包括的な一覧表を記載したポケット版 (PDF 版は通常サイズ) のマニュアルです。
- 『リファレンス・マニュアル』− 詳細な Transact-SQL 情報を記載しています。

- 『ビルディング・ブロック』− データ型、関数、グローバル変数、式、識別子とワイルドカード、予約語について説明しています。
- 『コマンド』− コマンドについて説明しています。
- 『プロシージャ』− システム・プロシージャ、カタログ・ストアド・プロシージャ、システム拡張ストアド・プロシージャ、dbcc ストアド・プロシージャについて説明しています。
- 『テーブル』− システム・テーブル、モニタリング・テーブル、dbcc テーブルについて説明しています。

- 『システム管理ガイド』でさらに詳しく説明しています。
  - 『第 1 巻』− 設定パラメータ、リソースの問題、文字セット、ソート順、システムの問題の診断方法に関する説明を含め、システム管理の基本の概要について説明しています。『第 1 巻』の後半は、セキュリティ管理に関する詳細な説明です。
  - 『第 2 巻』− 物理的なリソースの管理、デバイスのミラーリング、メモリとデータ・キャッシュの設定、マルチプロセッサ・サーバとユーザ・データベースの管理、データベースのマウントとマウント解除、セグメントの作成と使用、reorg コマンドの使用、データベース一貫性の検査方法についての手順とガイドラインを説明しています。『第 2 巻』の後半では、システムとユーザ・データベースをバックアップおよびリストアする方法について説明しています。
- 『システム・テーブル・ダイアグラム』− システム・テーブルと、そのエンティティとの関係をポスター形式で図解しています。フル・サイズのダイアグラムは印刷版だけで参照できます。コンパクト版は PDF 形式で参照できます。
- 『Transact-SQL ユーザーズ・ガイド』− リレーショナル・データベース言語の拡張版である Sybase の Transact-SQL について説明しています。まだ経験の浅いデータベース管理システムのユーザは、このマニュアルをガイドブックとして使用してください。pubs2 および pubs3 サンプル・データベースの詳細も説明しています。
- 『トラブルシューティング・シリーズ』−
  - 『トラブルシューティング：エラー・メッセージと詳細な解決方法』− 発生する可能性のある問題について、トラブルシューティング手順を説明しています。このマニュアルで取り上げられている問題は、Sybase 製品の保守契約を結んでいるサポート・センタに最も頻繁に寄せられるものです。
  - 『トラブルシューティング＆エラー・メッセージ・ガイド』− 発生頻度が高い Adaptive Server のエラー・メッセージの解決方法について詳しい手順を説明しています。
- 『暗号化カラム・ユーザーズ・ガイド』− Adaptive Server を使用して暗号化カラムを設定し、使用する方法について説明しています。

- 『インメモリ・データベース・ユーザーズ・ガイド』－ インメモリ・データベースの設定および使用方法について説明しています。
- 『Adaptive Server 分散トランザクション管理機能の使用』－ 分散トランザクション処理環境での Adaptive Server DTM 機能の設定、使用、トラブルシューティングについて説明しています。
- 『IBM® Tivoli® Storage Manager と Backup Server の使用』－ IBM Tivoli Storage Manager を設定および使用して Adaptive Server のバックアップを作成する方法について説明しています。
- 『高可用性システムにおける Sybase フェールオーバの使用』－ Sybase のフェールオーバ機能を使用して、Adaptive Server を高可用性システムのコンパニオン・サーバとして設定する方法について説明しています。
- 『Unified Agent および Agent Management Console』－ Unified Agent について説明しています。Unified Agent は、分散 Sybase リソースを管理、モニタ、制御するためのランタイム・サービスを提供します。
- 『ユーティリティ・ガイド』－ オペレーティング・システム・レベルで実行される isql および bcp などの、Adaptive Server のユーティリティ・プログラムについて説明しています。
- 『Web Services ユーザーズ・ガイド』－ Adaptive Server 用の Web サービスの設定、使用、トラブルシューティング方法について説明しています。
- 『XA インタフェース統合ガイド for CICS、Encina、TUXEDO』－ X/Open XA トランザクション・マネージャを備えた Sybase DTM XA インタフェースを使用する方法について説明しています。
- 『Adaptive Server Enterprise における XML サービス』－ データベースに XML 機能を導入する、Sybase ネイティブの XML プロセッサと Sybase Java ベースの XML のサポートについて、また XML サービスに準拠したクエリとマッピング用の関数について説明しています。

その他の情報

Sybase Getting Started CD、SyBooks™ CD、Sybase Product Manuals Web サイトを利用すると、製品について詳しく知ることができます。

- Getting Started CD には、PDF 形式のリリース・ノートとインストール・ガイド、SyBooks CD に含まれていないその他のマニュアルや更新情報が収録されています。この CD は製品のソフトウェアに同梱されています。Getting Started CD に収録されているマニュアルを参照または印刷するには、Adobe Acrobat Reader が必要です (CD 内のリンクを使用して Adobe の Web サイトから無料でダウンロードできます)。
- SyBooks CD には製品マニュアルが収録されています。この CD は製品のソフトウェアに同梱されています。Eclipse ベースの SyBooks ブラウザを使用すれば、使いやすい HTML 形式のマニュアルにアクセスできます。

  一部のマニュアルは PDF 形式で提供されています。これらのマニュアルは SyBooks CD の PDF ディレクトリに収録されています。PDF ファイルを開いたり印刷したりするには、Adobe Acrobat Reader が必要です。

SyBooks をインストールして起動するまでの手順については、Getting Started CD の『SyBooks インストール・ガイド』、または SyBooks CD の『*README.txt*』ファイルを参照してください。

- Sybase Product Manuals Web サイトは、SyBooks CD のオンライン版であり、標準の Web ブラウザを使用してアクセスできます。また、製品マニュアルのほか、EBFs/Updates、Technical Documents、Case Management、Solved Cases、ニュース・グループ、Sybase Developer Network へのリンクもあります。

  Technical Library Product Manuals Web サイトにアクセスするには、Product Manuals (http://www.sybase.com/support/manuals/) にアクセスしてください。

Web 上の Sybase 製品の動作確認情報

Sybase Web サイトの技術的な資料は頻繁に更新されます。

❖ **製品認定の最新情報にアクセスする**

1 Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)

2 [Certification Report] をクリックします。

3 [Certification Report] フィルタで製品、プラットフォーム、時間枠を指定して [Go] をクリックします。

4 [Certification Report] のタイトルをクリックして、レポートを表示します。

❖ **コンポーネント認定の最新情報にアクセスする**

1 Web ブラウザで Availability and Certification Reports を指定します。(http://certification.sybase.com/)

2 [Search By Base Product] で製品ファミリとベース製品を選択するか、[Search by Platform] でプラットフォームとベース製品を選択します。

3 [Search] をクリックして、入手状況と認定レポートを表示します。

❖ **Sybase Web サイト (サポート・ページを含む) の自分専用のビューを作成する**

MySybase プロファイルを設定します。MySybase は無料サービスです。このサービスを使用すると、Sybase Web ページの表示方法を自分専用にカスタマイズできます。

1 Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)

2 [MySybase] をクリックし、MySybase プロファイルを作成します。

Sybase EBF とソフトウェア・メンテナンス

❖ EBF とソフトウェア・メンテナンスの最新情報にアクセスする

1 Web ブラウザで Sybase Support Page (http://www.sybase.com/support) を指定します。

2 [EBFs/Maintenance] を選択します。MySybase のユーザ名とパスワードを入力します。

3 製品を選択します。

4 時間枠を指定して [Go] をクリックします。EBF/Maintenance リリースの一覧が表示されます。

鍵のアイコンは、「Technical Support Contact」として登録されていないため、一部の EBF/Maintenance リリースをダウンロードする権限がないことを示しています。未登録でも、Sybase 担当者またはサポート・コンタクトから有効な情報を得ている場合は、[Edit Roles] をクリックして、「Technical Support Contact」の役割を MySybase プロファイルに追加します。

5 EBF/Maintenance レポートを表示するには [Info] アイコンをクリックします。ソフトウェアをダウンロードするには製品の説明をクリックします。

表記規則

次の項では、このマニュアルで使用されている表記について説明します。

SQL は自由な形式の言語で、1 行内のワード数や、改行の仕方に規則はありません。このマニュアルでは、読みやすくするため、例や構文を文の句ごとに改行しています。複数の部分からなり、2 行以上にわたる場合は、字下げしています。複雑なコマンドの書式には、修正された BNF (Backus Naur Form) 記法が使用されています。

表 1 に構文の規則を示します。

表 *1:* このマニュアルでのフォントと構文規則

要素	例
コマンド名、プロシージャ名、ユーティリティ名、その他のキーワードは sans serif フォントで表記する。	select sp_configure
データベース名とデータ型は sans serif フォントで表記する。	master データベース
ファイル名、変数、パス名は斜体で表記する。	システム管理ガイド *sql.ini* ファイル *column_name* *$SYBASE/ASE* ディレクトリ

変数 (ユーザが入力する値を表す語) がクエリまたは文の一部である場合は Courier フォントの斜体で表記する。	`select column_name`
	`from table_name`
	`where search_conditions`

カッコはコマンドの一部として入力する。	compute *row_aggregate* (*column_name*)

要素	例
2 つのコロンと等号は、構文が BNF 表記で記述されていることを示す。この記号は入力しない。「～と定義されている」ことを意味する。	`::=`
中カッコで囲まれたオプションの中から必ず 1 つ以上を選択する。コマンドには中カッコは入力しない。	`{cash, check, credit}`
角カッコは、オプションを選択しても省略してもよいことを意味する。コマンドには角カッコは入力しない。	`[cash | check | credit]`
中カッコまたは角カッコの中のカンマで区切られたオプションをいくつでも選択できることを意味する。複数のオプションを選択する場合には、オプションをカンマで区切る。	`cash, check, credit`
パイプまたは縦線は複数のオプションのうち 1 つだけを選択できることを意味する。	`cash | check | credit`
省略記号 (...) は、直前の要素を必要な回数だけ繰り返し指定できることを意味する。	`buy thing = price [cash | check | credit] [, thing = price [cash | check | credit]]...` この例では、製品 (thing) を少なくとも 1 つ購入 (buy) し、価格 (price) を指定する必要があります。支払方法を選択できる。角カッコで囲まれた項目の 1 つを選択する。追加品目を、必要な数だけ購入することもできる。各 buy に対して、購入した製品 (thing)、価格 (price)、オプションで支払方法 (cash、check、credit のいずれか) を指定します。

- 次は、オプション句のあるコマンドの構文の例です。

  sp_dropdevice [*device_name*]

  複数のオプションを持つコマンドの例を示します。

  select *column_name*
  from *table_name*
  where *search_conditions*

  構文では、キーワード (コマンド) は通常のフォントで表記し、識別子は小文字で表記します。ユーザが提供するワードは斜体で表記します。

- Transact-SQL コマンドの使用例は次のように表記します。

  ```
  select * from publishers
  ```

- 次は、コンピュータからの出力例です。

```
pub_id    pub_name                city          state
-------   ---------------------   -----------   -----
0736      New Age Books           Boston        MA
0877      Binnet & Hardley        Washington    DC
1389      Algodata Infosystems    Berkeley      CA

(3 rows affected)
```

このマニュアルでは、例に使用する文字はほとんどが小文字ですが、Transact-SQL のキーワードを入力するときは、大文字と小文字は区別されません。たとえば、SELECT、Select、select はすべて同じです。

テーブル名などのデータベース・オブジェクトの大文字と小文字を Adaptive Server が区別するかどうかは、Adaptive Server にインストールされたソート順によって決まります。シングルバイト文字セットを使用している場合は、Adaptive Server のソート順を再設定することによって、大文字と小文字の区別の取り扱い方を変更できます。詳細については、『システム管理ガイド』を参照してください。

アクセシビリティ機能

このマニュアルには、アクセシビリティを重視した HTML 版もあります。この HTML 版マニュアルは、スクリーン・リーダーで読み上げる、または画面を拡大表示するなどの方法により、その内容を理解できるよう配慮されています。

Adaptive Server HTML マニュアルは、連邦リハビリテーション法第 508 条のアクセシビリティ規定に準拠していることがテストにより確認されています。第 508 条に準拠しているマニュアルは通常、World Wide Web Consortium (W3C) の Web サイト用ガイドラインなど、米国以外のアクセシビリティ・ガイドラインにも準拠しています。

注意 アクセシビリティ・ツールを効率的に使用するには、設定が必要な場合もあります。一部のスクリーン・リーダーは、テキストの大文字と小文字を区別して発音します。たとえば、すべて大文字のテキスト (ALL UPPERCASE TEXT など) はイニシャルで発音し、大文字と小文字の混在したテキスト (Mixed Case Text など) は単語として発音します。構文規則を発音するようにツールを設定すると便利かもしれません。詳細については、ツールのマニュアルを参照してください。

Sybase のアクセシビリティに対する取り組みについては、Sybase Accessibility (http://www.sybase.com/accessibility) を参照してください。Sybase Accessibility サイトには、第 508 条と W3C 標準に関する情報へのリンクもあります。

不明な点があるときは

Sybase ソフトウェアがインストールされているサイトには、Sybase 製品の保守契約を結んでいるサポート・センタとの連絡担当の方 (コンタクト・パーソン) を決めてあります。マニュアルだけでは解決できない問題があった場合には、担当の方を通して Sybase のサポート・センタまでご連絡ください。

第 1 章

概要

概要

DTM XA インタフェースは、XA インタフェース標準の Sybase 版で、X/Open 分散トランザクション処理 (DTP) モデルの 1 つの要素となるものです。X/Open DTP モデルは、分散トランザクション処理アプリケーションを開発するための業界標準です。

XA インタフェースを使用すると、Adaptive Server に格納されているデータに、CICS、Encina、または TUXEDO の TM からアクセスできます。ネイティブの Adaptive Server 分散トランザクション管理 (DTM) 機能を TM とともに、または TM なしで使用する方法については、『Adaptive Server 分散トランザクション管理機能の使用』を参照してください。

Microsoft Transaction Server (MTS) でも、トランザクション・コンポーネントで XA インタフェースを使用します。MTS XA 設定は、Sybase ODBC ドライバの設定時に行います。詳細については、ODBC ドライバのマニュアルを参照してください。

このほかにも IBM MQ サービスなどの TM で XA インタフェースを使用します。TM の設定方法の詳細については、トランザクション・マネージャのマニュアルを参照してください。

❖ Adaptive Server に格納されているデータに TM トランザクションからアクセスできるようにする

1 Adaptive Server と分散トランザクション管理機能をインストールします。ソフトウェアのインストールと機能ライセンスについては、使用しているプラットフォームに応じた Adaptive Server の『インストール・ガイド』を参照してください。

> **注意** 分散トランザクション管理は Adaptive Server の機能ですが、ライセンスが別途必要です。DTM の有効なライセンスを購入してインストールしてから使用してください。

2 すべてのクライアント・マシンに Sybase Open Client をインストールします。DTM XA インタフェースは Open Client に含まれています。

3 分散トランザクション管理機能をサポートしている Adaptive Server を起動します。詳細については、『Adaptive Server 分散トランザクション管理機能の使用』を参照してください。

4 「第 3 章 XA 環境の設定」の説明に従って、Embedded SQL または Client-Library アプリケーションと Adaptive Server で使用できるように、TM ソフトウェアを設定します。

5 「第 4 章 アプリケーション・プログラミングのガイドライン」の説明に従って、Embedded SQL または Client-Library アプリケーションをコーディング制約に準拠させます。

6 CICS、Encina、または TUXEDO の TM を起動します。

注意 Sybase XA 環境でグローバル・リカバリを手動で管理するには、XA 固有の dbcc コマンドを呼び出す必要があります。『Adaptive Server 分散トランザクション管理機能の使用』を参照してください。

稼働条件

Adaptive Server バージョン 12.5 用の XA インタフェースは、以下の製品と互換性があります。

- Open Client 12.5
- Embedded SQL 12.0 以降
- Adaptive Server 12.0 以降
- CICS/6000 2.1.1.6
- Encina 2.5/TX Series 4.2
- TUXEDO 6.4 (IBM プラットフォームでは 6.3/6.4)

第 2 章

Sybase XA 環境

この章では、X/Open DTP モデルについて説明し、このモデルに Sybase XA 環境のコンポーネント (特に DTM XA インタフェース、アプリケーション・プログラム、Adaptive Server) を適合させる方法についても示します。また、Sybase XA 環境での接続の確立方法と管理方法についても説明します。

定義

X/Open DTP モデルを使用するには、以下の用語について理解する必要があります。

- **トランザクション** − 1 つまたは複数の計算タスクから構成される 1 作業単位。ほとんどの場合、1 つのトランザクションに属する複数のタスクは共有リソースを処理する。
- **コミットされたトランザクション** − 共有リソースに対して永続的な変更を加えた、完了済みトランザクション。
- **ロールバックされたトランザクション** − 共有リソースに対する変更を無効にする、完全なトランザクション。
- **ACID テスト** − 正しいトランザクションであるかを調べるテスト。このテストにパスするには、トランザクションが次の特性を持っている必要がある。
  - **原子性 (atomicity)** − トランザクションの実行結果がすべて有効またはすべて無効。
  - **一貫性 (consistency)** − トランザクションがロールバックされる場合は、トランザクション実行結果を反映したすべてのリソースが、トランザクション実行前の状態に戻される。
  - **独立性 (isolation)** − トランザクションの実行結果は、そのトランザクションがコミットするまで他のトランザクションから参照できない。

    - **持続性 (durability)** – コミットの結果、リソースへの変更は永続的なものになり、この変更内容は今後障害が発生しても変わらない。
- **トランザクション処理** – 共有化および集中化したリソース上で複数のユーザが実行するトランザクションを調整するシステム。
- **分散トランザクション処理** – 各共有リソースをコンピュータ・ネットワーク上の物理的に異なる場所に配置する、トランザクション処理モデル。
- **ローカル・トランザクション** – 単一のデータベースに格納されているデータを処理の対象とするトランザクションであり、その内部に存在するタスクは単一のリソース・マネージャによって実行される。リソース・マネージャの定義については、「X/Open DTP モデルの概要」(4 ページ) を参照。
- **グローバル・トランザクション** – 複数のデータベースおよび複数のリソース・マネージャに影響を与えるトランザクション。
- **トランザクション・ブランチ** – グローバル・トランザクションを構成する処理の一部。
- **トランザクション識別子** – TM がトランザクションに割り当てる識別子。トランザクション・モニタの場合、トランザクション識別子を使用して、グローバル・トランザクションに関連するすべてのアクティビティを調整する。リソース・マネージャの場合、グローバル識別子を使用して、トランザクションに対して実行したタスクとリカバリ可能なタスクを一致させる。
- **リカバリ** – 障害が発生した後にトランザクション処理システムの一貫性を回復するための処理。特に、コミット前の状態で残っているトランザクションを解決する。

X/Open DTP モデルの概要

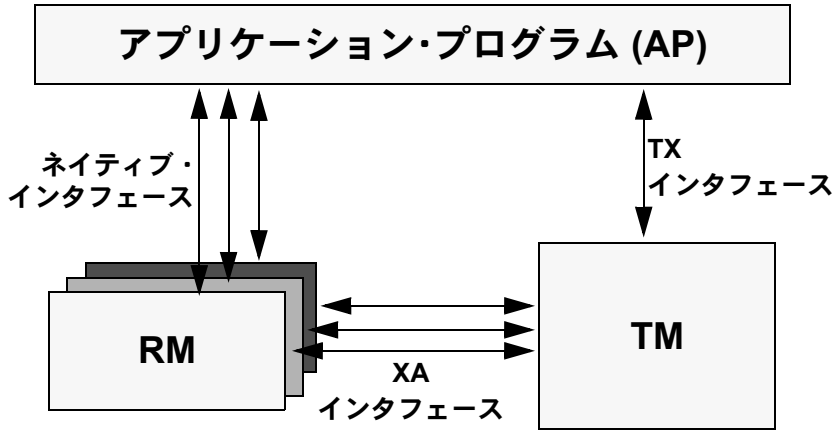
X/Open DTP モデルは、ソフトウェア・アーキテクチャ用のモデルです。このモデルを使用すると、複数のリソース・マネージャから提供されたリソースを複数のアプリケーション・プログラムで共有できます。また、そのプログラムの処理をグローバル・トランザクション内のタスクとして調整することもできます。

X/Open DTP モデルでは、分散トランザクション処理環境でキーのエンティティを決定し、その役割と対話を標準化します。エンティティは次のとおりです。

- トランザクション処理モニタ (TM)
- リソース・マネージャ (RM)
- アプリケーション・プログラム (AP)

この項では、X/Open DTP 機能モデルと、その主要なコンポーネントおよびインタフェースについて説明します。図 2-1 は、X/Open DTP モデルを示します。

図 *2-1: X/Open DTP* モデルの概念



上記のコンポーネントは、「コンポーネント間の通信方法」(6 ページ) で説明されているように、ネイティブ・インタフェース、XA インタフェース、TX インタフェースを介して通信します。

このモデルのコンポーネント

X/Open DTP 機能モデルは、次のコンポーネントで構成されています。

- アプリケーション・プログラム (AP)
- リソース・マネージャ (RM)
- トランザクション処理モニタ (TM)

AP には、特定のトランザクションまたはその一部を実行するコードが記述されています。そのコードには、グローバル・トランザクションの開始と終了が示されています。

RM は共有リソースへのアクセスを提供します。RM には、データベース・サーバ、ファイル・サーバ、プリント・サーバがあります。通常の X/Open DTP 環境では、1 つの AP が複数の RM と通信します。Sybase XA 環境では、Adaptive Server データベースが RM として機能します。

TM は、トランザクションに関連するすべてのコンポーネント間の通信を調整します。TM を使用すると、AP が行った作業をグローバル・トランザクションのタスクの 1 つとして、アトミックにコミットしたりアボートしたりできます。

特に、TM には次のタスクが含まれます。

- トランザクションにグローバル識別子を割り当てる。
- グローバル・トランザクションの進行状況をモニタリングする。
- AP と RM の間のトランザクション情報の流れを調整する。
- トランザクションのコミットメント・プロトコルと障害リカバリを管理する。詳細については、「コンポーネント間の対話方法」(6 ページ) を参照。

コンポーネント間の通信方法

AP、RM、TM は、3 種類の異なるインタフェース (ネイティブ、TX、XA) を介して通信します。

ネイティブ・インタフェースは、AP が RM に要求を直接送信するときの通信手段です。これは RM に固有のインタフェースです。Sybase XA 環境では、ネイティブ・インタフェースは Embedded SQL または Client-Library のいずれかになります。

TX インタフェースは、AP と TM 間の通信手段です。AP では TX 呼び出しを使用してトランザクション境界を示します。つまり、AP は TX インタフェースを介して、TM の開始、TM によるグローバル・トランザクションのコミットやロールバックを要求します。これは TM に固有のインタフェースです。

XA インタフェースは、RM と TM 間の通信手段です。DTM XA インタフェースは、Sybase 版の Adaptive Server 用インタフェースです。TM は XA 呼び出しを使用して、トランザクションの開始、コミット、ロールバックの時期を RM に通知します。TM はリカバリも実行します。

コンポーネント間の対話方法

トランザクションの開始から終了まで、複数のコンポーネントが一緒に機能します。

AP によって、トランザクション境界が区切られます。AP は TX 呼び出しを介して、グローバル・トランザクションが開始されていることを TM に通知します。その後、TM が XA 呼び出しを介して有効なすべての RM と通信し、グローバル・トランザクション内で RM が AP のために実行するすべての処理に 1 つのトランザクション識別子を関連付けます。

AP が TM によるグローバル・トランザクションのコミットを要求すると、TM と RM では 2 フェーズ・コミット・プロトコルを使用してトランザクションの原子性を保証します。

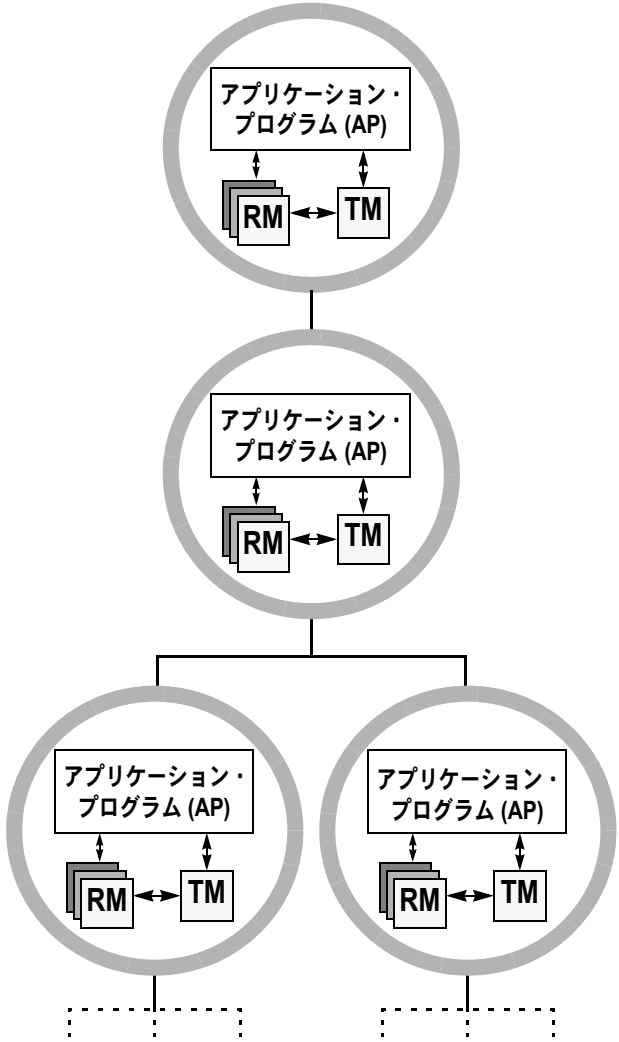
トランザクションの完了処理は、準備フェーズとコミット・フェーズで実行されます。2 フェーズ・コミット・プロトコルの詳細については、『Open Client DB-Library/C リファレンス・マニュアル』を参照してください。

準備フェーズでは、TM が各 RM に対して、グローバル・トランザクションのそれぞれの部分のコミットを準備するよう要求します。この部分は、「トランザクション・ブランチ」とも呼ばれます。

コミット・フェーズでは、TM が各 RM に対して、トランザクションのそれぞれのブランチをコミットまたはアボートするように指示します。すべての RM から、それぞれのトランザクション・ブランチの準備が整っているというレポートを受けると、TM はトランザクション全体をコミットします。RM のいずれかから、準備が整っていない、または応答できないというレポートを受けると、TM はトランザクション全体をロールバックします。

図 2-2 に一般的なトランザクション・ブランチの構造を示します。

図 2-2: トランザクション・ブランチ



リカバリ

TM はグローバル・リカバリを管理します。状況によっては、管理者は、TM とは別にトランザクション・ブランチを実行できます。このような管理者の決定を「ヒューリスティック決定」といいます。

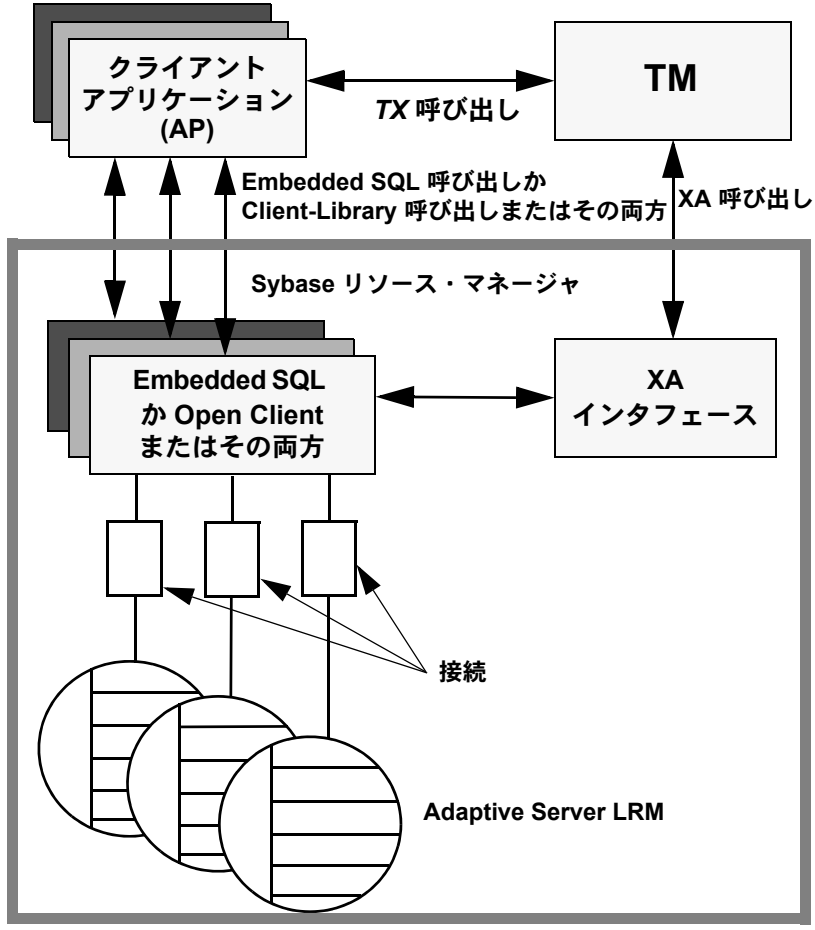
ヒューリスティック決定が TM の決定と競合することがあります。たとえば、管理者がトランザクション・ブランチをコミットしているのに、TM がそのアボートを要求することがあります。

このような競合が発生した場合、システム管理者が手動介入する必要があります。Sybase XA 環境のヒューリスティック決定については、『Adaptive Server 分散トランザクション管理機能の使用』を参照してください。

Sybase XA 環境

DTM XA インタフェースは、X/Open の DTP モデルを実装する Sybase のトランザクション処理モデルによって異なります。図 2-3 の例では、Adaptive Server を RM として使用します。

図 *2-3: Sybase XA DTP* モデル



Sybase XA 環境のコンポーネント

Sybase XA 環境のコンポーネントは次のとおりです。

- Sybase DTM XA インタフェース － Sybase に実装されている Adaptive Server 用の XA インタフェース。「コンポーネント間の通信方法」(6 ページ) の説明を参照。
- Open Client ライブラリ － Client-Library 呼び出しを、アプリケーションとリソース・マネージャ間のネイティブ・インタフェースの一部にすることができる。

  注意 XA Client ライブラリは、Solaris 64、AIX 64、HP-UX 64、Windows プラットフォームでは 32 ビット・モードで実行され、Solaris 64、AIX 64、HP-UX 64 プラットフォームでは 64 ビット・モードで実行されます。

- Embedded SQL/C および Embedded SQL/COBOL － Embedded SQL 呼び出しを、アプリケーションとリソース・マネージャ間のネイティブ・インタフェースの一部にすることができる。
- 1 つまたは複数の Adaptive Server － RM として機能する。
- XA 設定ファイル － このファイルには、XA とともに使用するクライアント／サーバ接続の定義エントリが含まれている。
- XA 固有の dbcc コマンドのセット － システム管理者は、これらのコマンドを使用してヒューリスティック・トランザクションを管理する。
- TM 固有の設定ファイルとコマンド。

「第 3 章 XA 環境の設定」では、Adaptive Server に格納されているデータにトランザクションが DTM XA インタフェースを使ってアクセスできるようにするための、上記のコンポーネントの設定方法について説明します。

Sybase XA 環境での接続

X/Open DTP モデルには接続という概念はありませんが、Sybase クライアント／サーバ・アーキテクチャでは接続が最も重要な機能です。Sybase XA 環境ではこの矛盾を解決する必要があります。

解決策として、Sybase XA 環境には論理リソース・マネージャ (LRM) という概念が用意されています。

LRM を介した接続の特定

Sybase RM の各インスタンスは、1 つまたは複数の LRM として TM に表示されます。

LRM は、クライアント／サーバ接続に記号名を割り当てます。AP はその名前を使用して、1 つまたは複数の Adaptive Server に対する物理的な接続を特定します。TM はその名前を使用して、AP に代わって接続をオープンします。

接続情報の格納場所

以下に挙げた Sybase XA 環境のコンポーネントには、LRM に関する情報が格納されています。システム管理者は、これらのファイルを設定してから TM を起動します。完全な設定処理の情報については、「第 3 章 XA 環境の設定」を参照してください。

Sybase XA 設定ファイル

Sybase XA 設定ファイルでは、LRM ごとにエントリが 1 つ含まれています。このエントリにより、LRM と物理的な Adaptive Server 名が関連付けられ、事前接続 Client-Library 機能とプロパティが LRM に割り当てられます。XA 設定ファイルの詳細については、「DTM XA インタフェース用の XA 設定ファイル」(18 ページ) を参照してください。

CICS XA 製品の定義 (XAD)

CICS XAD には、各 LRM にスタンザが 1 つ含まれています。このスタンザは、各 LRM にオープン文字列形式のユーザ名とパスワードを割り当てます。このユーザ名とパスワードによって、特定の接続を介した Adaptive Server リソースへのアクセスを Sybase XA 環境で管理できます。CICS XAD ファイルの詳細については、「CICS 領域 XAD 定義への Sybase スタンザの追加」(27 ページ) を参照してください。

Encina の *monadmin create rm* コマンド

monadmin create rm コマンドを使用して、各 LRM に「オープン文字列」形式のユーザ名とパスワードを割り当てます。このユーザ名とパスワードによって、特定の接続を介した Adaptive Server リソースへのアクセスを Sybase XA 環境で管理できます。Encina の monadmin コマンドの詳細については、「monadmin create rm によるオープン文字列の割り当て」(28 ページ) を参照してください。使用しているバージョンの Encina に、RM を指定するコマンドが追加されていることがあります。

注意　シェル・コマンド monadmin の代わりに、Encina 対話型コマンド enconsole を使用できます。

詳細については、『Encina Monitor System Administrator's Guide and Reference』を参照してください。

TUXEDO の *UBBCONFIG* ファイル

TUXEDO を統合するには、Sybase 設定ファイルを変更するだけでなく、*UBBCONFIG* という TUXEDO 設定ファイルをカスタマイズする必要があります。*UBBCONFIG* ファイルで変更が必要な部分は、オープン文字列のみです。この部分には、ユーザ名とパスワードが含まれています。ユーザ名とパスワードによって、接続を介した Adaptive Server リソースへのアクセスを Sybase XA 環境で管理できます。詳細については、「UBBCONFIG ファイルの設定」(32 ページ) を参照してください。

接続の確立

TM では XA インタフェースを併用し、いくつかの手順に従ってアプリケーションと RM 間の接続を確立します。

❖ CICS 環境で接続を確立する

1 CICS 領域を利用する準備ができたら、各オープン文字列に含まれた情報を使用して、XAD 内に設定された各 LRM に XA オープン呼び出しを発行します。

2 CICS 領域は、各スタンザに関連付けられているオープン文字列を XA インタフェース・ライブラリに渡します。オープン文字列には、LRM 名、ユーザ名、パスワードが含まれています。

3 XA インタフェースは Sybase XA 設定ファイルから LRM 名を検索し、この LRM 名を実際の RM 名 (つまり、実際の物理的な Adaptive Server) と一致させます。RM 名は、Adaptive Server の interfaces ファイルにあるエントリと一致します。

4 XA インタフェースは、Adaptive Server への接続を各 LRM エントリに 1 つ確立します。そして、LRM 用に設定された事前接続のプロパティと機能を各接続に与えます。

❖ Encina 環境で接続を確立する

1 アプリケーションは、mon_RegisterRmi 関数を発行して、LRM の使用を要求します。

2 オープン文字列に含まれている情報を使用して、TM から LRM に XA オープン呼び出しが発行されます。この LRM は、monadmin create rm コマンドに設定されていて、その名前は前述の手順 1 で発行したものと一致します。

3 TM は、各 monadmin create rm コマンドに関連付けられているオープン文字列を XA インタフェースに渡します。オープン文字列には LRM 名が含まれています。

4 XA インタフェースは Sybase XA 設定ファイルから LRM 名を検索し、この LRM 名を実際の RM 名 (つまり、実際の物理的な Adaptive Server) と一致させます。RM 名は、Adaptive Server の *interfaces* ファイルにあるエントリと一致します。

5　XA インタフェースは、Adaptive Server への論理接続を各 LRM エントリに 1 つ確立します。そして、LRM 用に設定された事前接続のプロパティと機能を各接続に与えます。

❖ **TUXEDO 環境で接続を確立する**

1　アプリケーションは、*UBBCONFIG* ファイルに指定されている LRM を使用して、グローバル・トランザクションのブランチ用の論理接続を参照します。LRM 名を使用すると、アプリケーションは LRM を暗黙的に要求して確立します。

2　トランザクション・マネージャは、手順 1 で発行した LRM 名と一致する LRM を介して、適切なオープン文字列を XA インタフェースに渡します。XA インタフェースは、LRM 名、ユーザ名、パスワードを使用します。

3　XA インタフェースは、*xa_config* ファイルを使って LRM 名と Adaptive Server との関係を調べます。Adaptive Server 名は、ネットワーク情報が格納されている *interfaces* ファイルのエントリと一致します。

4　XA インタフェースは、Adaptive Server への論理接続を各 LRM エントリに 1 つ確立します。そして、LRM 用に設定された事前接続のプロパティと機能を各接続に与えます。

LRM 間での作業の分散

システム管理者とアプリケーション・プログラマは、Sybase XA 環境内に含める LRM の数、および LRM の名前について合意しておく必要があります。

システム管理者が、この同意に基づき TM と Sybase XA の設定ファイルを設定します。アプリケーション・プログラマは、アプリケーション・コード内で特定の LRM 名を呼び出し、接続を介してグローバル・トランザクションの一部を送信します。TM がこの分散を調整します。

実際に使用する接続よりも多くの接続に対して、Sybase XA 環境を設定できます。したがって、XA 設定ファイルにはアクティブではないエントリが含まれることがあります。

たとえば、Adaptive Server SYBXA_1 および LRM 接続 connection_1 が存在する CICS 環境では、次のようになります。

- Adaptive Server の *interfaces* ファイルには、サーバ情報が含まれています。

```
sybXA_1 query tcp ether groucho 6161
```

- XA 設定ファイルには、接続情報とサーバ情報が含まれています。

```
; lrm – Names the logical connection as seen by the
application and the TP monitor.
; server – Names the physical server as found in the Sybase
interfaces file.
lrm=connection_1
server=SYBXA_1
```

- LRM の XAD ファイル・スタンザには、接続情報が含まれています。

```
XAOpen="-Uuser1 -Ppassword1 -Nconnection_1"
```

- アプリケーション・プログラムには、connection_1 を使用するトランザクションが含まれています。

第 3 章

XA 環境の設定

この章では、CICS、Encina、TUXEDO の各 TM で XA 環境を使用するための設定方法を説明します。

Adaptive Server をリソース・マネージャとして使用するには、DTM XA インタフェース・ライブラリを、使用している X/Open XA 準拠のトランザクション・マネージャにリンクさせる必要があります。

注意 使用しているシステムに合わせて DTM XA インタフェースを設定する方法の詳細については、*$SYBASE/$SYBASE_OCS/sample* 内のサブディレクトリにある *README* ファイルを参照してください。

Adaptive Server の設定

Adaptive Server を Sybase XA 環境で使用するには、分散トランザクション管理機能を使用するためのライセンスと設定が必要です。詳細については、Adaptive Server のインストール・ガイドと『Adaptive Server 分散トランザクション管理機能の使用』を参照してください。

分散トランザクション管理機能の使用ライセンスを持っている場合は、enable dtm 設定パラメータでその機能を有効にできます。

```
sp_configure 'enable dtm', 1
```

このパラメータを有効にするには、Adaptive Server の再起動が必要です。

DTM XA インタフェース用のオープン文字列パラメータ

X/Open XA の仕様では、各リソース・マネージャ・ベンダがオープン文字列とクローズ文字列を定義できるようになっています。DTM XA インタフェースでは、クローズ文字列は必須ではなく、使用されていません。

DTM XA インタフェースでは、以下のような必須またはオプションのオープン文字列パラメータを使用します。

オープン文字列パラメータ

DTM XA インタフェースでは、オープン文字列のパラメータを以下の形式で指定します。

-N*lrm_name* -U*username* -P*password* [-L*logfile_name*]
[-T*traceflags*] [-V11] [-O1] [-O-1]

オープン文字列の各パラメータを表 3-1 に示します。

表 3-1: Sybase X/Open XA オープン文字列パラメータ

パラメータ	意味
lrm_name	XA 設定ファイルで定義されている LRM の名前。
username	Adaptive Server へのログイン時に使用するユーザ名。詳細については、「username に必要な dtm_tm_role」(17 ページ) を参照してください。
password	ユーザ名に対応するパスワード。
logfile_name	XA インタフェースがトレース情報を書き込む、完全に修飾されたファイル名 (オプション)。 XA インタフェースは、最初の xa_open() の呼び出しによってログ・ファイルとトレース・フラグの設定を初期化する。*logfile_name* が指定されていない場合、DTM XA インタフェースは、現在のディレクトリで *syb_xa_log* という名前のファイルに情報を書き込む。
traceflags	トレース・フラグは、ログ・ファイルに書き込まれる出力を制御する (オプション)。有効なトレース・フラグのリストについては、「[all] セクションのパラメータ定義」(19 ページ) を参照してください。
-V11	下位互換性を保つために、Open Client バージョン 11 の動作を指定する (オプション)。
-O1 または -O-1	トランザクション動作のオプションを指定する。現時点では、-O1 (密結合トランザクション・ブランチ) のみがサポートされている。-O1 はオプションを設定し、-O-1 はオプション設定を解除する。デフォルトは -O-1。 **警告！** -O1 を設定する場合は、「密結合トランザクション」(45 ページ) を参照してください。

username に必要な *dtm_tm_role*

リソース・マネージャのオープン文字列では、指定されている *username* に、対応する Adaptive Server での dtm_tm_role が必要です。システム・セキュリティ管理者は、sp_role または grant コマンドを使用してこの役割を割り当てることができます。次に例を示します。

```
sp_role "grant", dtm_tm_role, user_name
```

ログ・ファイルとトレース・フラグのパラメータ

Adaptive Server に対して DTM XA インタフェースを使用している場合、X/Open XA オープン文字列ではなく XA 設定ファイルの [all] セクションで、ログ・ファイルとトレース・フラグのパラメータを定義できます。ログ・ファイルとトレース・フラグのコンポーネントの詳細については、「共通 LRM パラメータを定義する [all] セクション」(18 ページ) を参照してください。

ログ・ファイルのエントリのラベル

DTM XA インタフェースでは、ログ・ファイル内の各エントリに、メッセージの重要度または原因を示すラベルが付きます。各ラベルを表 3-2 に示します。

表 *3-2:* ログ・ファイルのメッセージ・ラベル

ラベル	エントリのタイプ
エラー	トランザクション・マネージャに返されるエラー。
Fatal Error	DTM XA インタフェースで発生した深刻な障害。アプリケーションまたはトランザクション・マネージャでもエラーが発生した可能性がある。
Message	以前のエラーに関する追加情報、または動作環境の説明。
Warning	トランザクション・システムに問題が発生している可能性がある。
Note	問題は起きていないが、エラー発生時に役に立つ情報。
XA trace	xa トレース・フラグ設定の結果として記録された情報。
RM trace	xl トレース・フラグ設定の結果として記録された情報。
Connection trace	xc トレース・フラグ設定の結果として記録された情報。
ASE I/F trace	xs トレース・フラグ設定の結果として記録された情報。
Misc trace	misc トレース・フラグ設定の結果として記録された情報。
Event trace	event トレース・フラグ設定の結果として記録された情報。
Verbose trace	v トレース・フラグ設定の結果として記録された情報。
Function trace	cmn トレース・フラグ設定の結果として記録された情報。
Open Client trace	ct トレース・フラグ設定の結果として記録された情報。

xa_open() 関数の動作

X/Open XA 関数の xa_open() は、Adaptive Server に対して接続を 1 つ開始します。オープン文字列で定義された *username* と *password* は、サーバで dtm_tm_role を持っている必要があります。詳細については、「username に必要な dtm_tm_role」(17 ページ) を参照してください。

DTM XA インタフェース用の XA 設定ファイル

Adaptive Server への DTM XA インタフェースは、XA 設定ファイルを使用して Open Client 接続を設定するためのメカニズムを提供します。接続機能、プロパティ、オプションは、すべて XA 設定ファイルを使用して設定してください。

設定ファイルを指定する環境変数

DTM XA インタフェースでは、環境変数 XACONFIGFILE を使用して、XA 設定ファイルのフル・パスとファイル名を探します。この環境変数で、必要に応じて異なるロケーションと名前を設定情報として指定できます。

以下は UNIX プラットフォームで指定する場合の例です。

```
setenv XACONFIGFILE /usr/u/sybase/xaconfig1.txt
```

XACONFIGFILE が定義されていない場合や、この変数で指定した設定ファイルが有効でない場合、DTM XA インタフェースは以下のディレクトリで *xa_config* という名前のファイルを探します。

- *$SYBASE/$SYBASE_OCS/config*
- *$SYBASE/$SYBASE_OCS*
- *$SYBASE/config*
- *$SYBASE*

最初に見つかった *xa_config* ファイルが DTM XA インタフェースで使用されます。

共通 LRM パラメータを定義する [all] セクション

DTM XA インタフェースでは、[all] セクションですべての LRM に適用するパラメータを定義します。

[all] セクションで定義される一部のパラメータ (ログ・ファイルとトレース・フラグの定義) は、X/Open XA トランザクション・マネージャ用のオープン文字列でも定義できます。

[all] セクションのパラメータ定義

XA 設定ファイルの [all] セクションで指定するエントリは、次のとおりです。

```
[all]
logfile="logfile_name"
traceflags="[xa | xl | xc | cm| event | misc | os | ct | all]"
[properties=name=value] [...]
```

注意　*xa_config* ファイルにある値 "0x" 値は、一部の文字の処理方法が原因で、失われたり、ASCII 以外の文字に変換されたりする場合があります。この問題を回避するには、すべての文字列値を引用符で囲む必要があります。引用符で囲まれていない場合、文字列中に予期しない文字が混入する可能性があります。

表 3-3 は、各コンポーネントを示します。

表 *3-3: XA* 設定ファイルの *[all]* セクションで指定するパラメータ

パラメータ	意味
logfile_name	DTM XA インタフェースがトレース情報を書き込む、完全に修飾されたファイル名。 DTM XA インタフェースは、最初の xa_open() の呼び出しによってログ・ファイルとトレース・フラグの設定を初期化する。
traceflags	トレース・フラグは、ログ・ファイルに書き込まれる出力を制御する。以下のフラグを 1 つ以上指定する必要がある。 • all − すべてのトレース。 • ct − CS_DBG_ERROR フラグを指定した ct_debug オプション (ct_debug 機能は Client-Library のデバッグ・バージョンからのみ使用可能)。 • cmn − 内部 XA インタフェース関数のエントリ・ポイントと終了ポイントのトレース。 • event − 重要な内部イベントのトレース。 • misc − 問題の解決についてのアクティビティと情報のトレース。 • xa − xa_* レベルでのエントリ・ポイントと終了ポイントのトレース。 • xc − xc_* レベルでのエントリ・ポイントと終了ポイントのトレース。 • xl − xl_* レベルでのエントリ・ポイントと終了ポイントのトレース。 **注意**　xc_*、xl_*、event、misc、cmn レベルでのトレースは、Sybase の開発でのみ使用します。Sybase 製品の保守契約を結んでいるサポート・センタから指示があった場合にのみ、これらのトレース・レベルを指定してください。

パラメータ	意味
プロパティ	**注意** 以下のプロパティは、[all] スタンザで設定してください。[xa] スタンザでは設定できません。 CS_LOGIN_TIMEOUT=*timeout* XA 設定ファイルの [all] セクションで、以下のオプション・プロパティを指定できる。 • PROPERTIES=CS_DISABLE_POLL=[CS_TRUE | CS_FALSE] • PROPERTIES=CS_EXTRA_INF=[CS_TRUE | CS_FALSE] • PROPERTIES=CS_HIDDEN_KEYS=[CS_TRUE | CS_FALSE] • PROPERTIES=CS_MAX_CONNECT=*number_of_connections* • PROPERTIES=CS_NOINTERRUPT=[CS_TRUE | CS_FALSE] • PROPERTIES=CS_TEXTLIMIT=*textlimit* • PROPERTIES=CS_TIMEOUT=*timeout*

XA 設定ファイルの編集

XA 設定ファイルは、アプリケーション環境に合わせてカスタマイズする必要があります。任意のテキスト・エディタを使用して XA 設定ファイルを開き、編集してください。以下は、XA 設定ファイルのサンプルです。

```
; Comment line as first line of file REQUIRED!
;
; xa_config - sample xa_config file.
;
; Note that the Adaptive Server names may need
; to be customized for your environment.

; simprpc.ct sample application entry.

[all]
       logfile="logfile_name"
       traceflags="traceflags"
       properties="name"="value" [, "name"="value"] [...]

 [xa]
       lrm="connection1"
       server="sqlserver"

 ; Rentapp sample xa_config entries.

 [xa]
       lrm="FLEET_CON"
       server="fleetsrv"

 [xa]
       lrm="RESERVE_CON"
```

```
server="rsrvsrv"
```

注意　*xa_config* ファイルの 1 行目は必ずセミコロン (;) で始まるコメント行にしてください。

このほかに指定する LRM についても、それぞれ以下の形式でエントリを作成します。connection1 エントリは、インストール環境の検証用に残しておいてください。

```
[xa]
<tab> lrm="connection_name"
<tab> server="adaptive_server_name"
<tab> capabilities="name"="value" [, "name"="value"] [...]
<tab> properties="name"="value" [, "name"="value"] [...]
<tab> options="name"="value" [, "name"="value"] [...]
```

connection_name は、アプリケーションと SQL 間の接続を表す記号名です。*adaptive_server_name* は、接続に関連付けられている Adaptive Server の名前です。*adaptive_server_name* は、*interfaces* ファイルで定義されているサーバ名と対応する必要があります。

DTM XA インタフェースで使用できる機能、プロパティ、オプションのリストについては、「その他の機能、プロパティ、オプション」(21 ページ) を参照してください。

その他の機能、プロパティ、オプション

XA 設定ファイルで機能、プロパティ、オプションを指定する場合、エントリの一般的な形式は以下のとおりです。

```
<tab> capabilities="name"="value" [, "name"="value"] [...]
<tab> properties="name"="value" [, "name"="value"] [...]
<tab> options="name"="value" [, "name"="value"] [...]
```

以下の表に、DTM XA インタフェースの XA 設定ファイルで定義できる機能の名前を示します。これらの表で特に指定がない場合、それぞれの機能で有効な値は、CS_TRUE と CS_FALSE です。

注意　これらの機能、プロパティ、オプションの名前と値はすべて、CS-Library の「キーワード」と対応します。詳細については、『Open Client Client-Library/C リファレンス・マニュアル』を参照してください。

CS_CON_NOINBAND	CS_DATA_NODATETIMEN	CS_DATA_NOMONEYN
CS_CON_NOOOB	CS_DATA_NODEC	CS_DATA_NONUM
CS_DATA_NOBIN	CS_DATA_NOFLT4	CS_DATA_NOSENSITIVITY
CS_DATA_NOVBIN	CS_DATA_NOFLT8	CS_DATA_NOTEXT
CS_DATA_NOLBIN	CS_DATA_NOIMAGE	CS_PROTO_NOBULK
CS_DATA_NOBIT	CS_DATA_NOINT1	CS_PROTO_NOTEXT
CS_DATA_NOBOUNDARY	CS_DATA_NOINT2	CS_RES_NOEED
CS_DATA_NOCHAR	CS_DATA_NOINT4	CS_RES_NOMSG
CS_DATA_NOVCHAR	CS_DATA_NOINT8	CS_RES_NOPARAM
CS_DATA_NOLCHAR	CS_DATA_NOINTN	CS_RES_NOTDSDEBUG
CS_DATA_NODATE4	CS_DATA_NOMNY4	CS_RES_NOSTRIPBLANKS
CS_DATA_NODATE8	CS_DATA_NOMNY8	

以下の表に、DTM XA インタフェースの XA 設定ファイルで定義できるプロパティの名前を示します。これらの表で特に指定がない場合、それぞれのプロパティで有効な値は、CS_TRUE と CS_FALSE です。

CS_ASYNC_NOTIFS	CS_SEC_NEGOTIATE
CS_DIAG_TIMEOUT	CS_TDS_VERSION= [CS_TDS_40 | CS_TDS_42 |CS_TDS_46 |CS_TDS_50]
CS_DISABLE_POLL	CS_TEXTLIMIT=*textlimit*
CS_HIDDEN_KEYS	CS_EXTRA_INF
CS_PACKETSIZE=*packetsize*	CS_MAX_CONNECT=*connections*
CS_SEC_APPDEFINED	CS_NOINTERRUPT
CS_SEC_CHALLENGE	CS_TIMEOUT=*timeout*
CS_SEC_ENCRYPTION	

以下の表に、DTM XA インタフェースの XA 設定ファイルで定義できるオプションの名前を示します。これらの表で特に指定がない場合、それぞれのオプションで有効な値は、CS_TRUE と CS_FALSE です。

CS_OPT_ANSINULL
CS_OPT_ANSIPERM
CS_OPT_ARITHABORT
CS_OPT_ARITHIGNORE
CS_OPT_DATEFIRST=[CS_OPT_SUNDAY |CS_OPT_MONDAY |CS_OPT_TUESDAY |CS_OPT_WEDNESDAY |CS_OPT_THURSDAY |CS_OPT_FRIDAY |CS_OPT_SATURDAY]
CS_OPT_DATEFORMAT=[CS_OPT_FMTMDY |CS_OPT_FMTDMY |CS_OPT_FMTYMD |CS_OPT_FMTYDM |CS_OPT_FMTMYD |CS_OPT_FMTDYM]
CS_OPT_FIPSFLAG
CS_OPT_FORCEPLAN
CS_OPT_FORMATONLY

CS_OPT_GETDATA
CS_OPT_ISOLATION=[CS_OPT_LEVEL1 |CS_OPT_LEVEL3]
CS_OPT_NOCOUNT
CS_OPT_NOEXEC
CS_OPT_PARSEONLY
CS_OPT_QUOTED_IDENT
CS_OPT_RESTREES
CS_OPT_ROWCOUNT=*rowcount*
CS_OPT_SHOWPLAN
CS_OPT_STATS_IO
CS_OPT_STATS_TIME
CS_OPT_STR_RTRUNC
CS_OPT_TEXTSIZE=*textsize*

CICS での DTM XA インタフェースの使用

この項では、CICS 環境で DTM XA インタフェースを使用するための設定方法を説明します。XA 設定ファイルの作成については、「DTM XA インタフェース用の XA 設定ファイル」(18 ページ) を参照してください。

スイッチロード・ファイルの構築

CICS 環境で定義された各 RM は、XA スイッチロード・ファイルを提供する必要があります。スイッチロード・ファイルは CICS 設定のコンポーネントで、XAD 内で参照されます。これには、DTM XA インタフェースにより提供される、RM の名前、フラグ、バージョン番号、RM のエントリ・ポイントを指す一連の null 以外のポインタが含まれています。

Sybase XAD はすべて、1 つのスイッチロード・ファイルを共有しています。Sybase スイッチロード・ファイルは、以下のロケーションにある *sybasexa.c* ファイルを使用して構築できます。

$SYBASE/$SYBASE_OCS/sample/xa-dtm/cics/switch

次は、*sybasexa.c* の内容です。

```
/*
 **
 ** sybasexa.c
 **
 ** The sybasexa routine references the Sybase xa
 ** switch structure named "sybase_TXS_xa_switch".
 ** The switch structure is part of the
 ** XA product library "libdtmxa.a".
```

```
**
** See your CICS documentation for details on the
** switch-load file.
*/

#include <stdio.h>
#include <tmxa/xa.h>

extern struct xa_switch_t sybase_TXS_xa_switch;
extern struct xa_switch_t RegXA_xa_switch;
extern struct xa_switch_t *cics_xa_switch;

struct xa_switch_t *sybasexa(void)
{
    cics_xa_switch = &sybase_TXS_xa_switch;

    cics_xa_init();

    return(&RegXA_xa_switch);
}
```

このソース・コードは Sybase XA スイッチ構造体を参照しています。Sybase XA スイッチ構造体は DTM XA インタフェース全体で使用されているもので、次のように定義されています。

```
struct xa_switch_t sybase_TXS_xa_switch =
 {
 "SYBASE_SQL_SERVER",
 TMNOFLAGS,
 0,
 xa_open,
 xa_close,
 xa_start,
 xa_end,
 xa_rollback,
 xa_prepare,
 xa_commit,
 xa_recover,
 xa_forget,
 xa_complete
 };
```

TMNOFLAGS は、DTM XA インタフェースがスレッド・マイグレーションをサポートするが、動的な登録や同期オペレーションはサポートしないことを示しています。これらの機能の詳細については、『X/Open CAE Specification (December 1991) Distributed Transaction Processing: The XA Specification』を参照してください。

IBM RISC System/6000 AIX でのスイッチロード・ファイルのコンパイル

makefile *sybasexa.mk* を使用して、*$SYBASE/$SYBASE_OCS/sample/xa-dtm/cics/switch* にある *sybasexa.c* をコンパイルします。

次は、*sybasexa.mk* の内容です。使用している設定に合わせて編集してください。

```
SYB_LIBDIR = $(SYBASE)/$(SYBASE_OCS)/lib
SYBLIBS = -lxadtm -lct_r.so -lcs_r.so -ltcl_r.so -lcomn_r.so -lintl_r
          -lxdsxom

all :sybasexa.c xlc_r4  -bnoquiet -v -D_THREAD_SAFE ¥
 -I/usr/lpp/encina/include sybasexa.c  ¥
 -o sybasexa ¥
 -esybasexa ¥
 -L/usr/lpp/cics/lib ¥
 -L$(SYB_LIBDIR) ¥
 $(SYBLIBS) ¥
 -lcicsrt -ldce -lm ¥
 /usr/lpp/cics/lib/regxa_swxa.o
```

注意　共有可能なバージョンの CS-Library (*libcs_r.so*) と Common Library (*libcom_r.so*) を使用してください。

HP9000 Series 800 HP-UX でのスイッチロード・ファイルのコンパイル

makefile *sybasexa.mk.hp800* を使用して、*$SYBASE/$SYBASE_OCS/sample/xa-dtm/cics/switch* にある *sybasexa.c* をコンパイルします。

次は、*sybasexa.mk.hp800* の内容です。使用している設定に合わせて編集してください。

```
#
# Makefile to compile the LoadSwitchTable
# This makefile should be run with the command
# "make -f sybasexa.mk.hp800"
#

CC=/opt/ansic/bin/cc
CCOPTS= -Aa +z -Dsybasexa=CICS_XA_Init
ENCINA=/opt/encina
CICS=/opt/cics
LD=/usr/ccs/bin/ld

SYB_LIBDIR = $(SYBASE)/$(SYBASE_OCS)/lib
CICS_LIBDIR = $(CICS)/lib

all:sybasexa

sybasexa:sybasexa.o
```

```
$(LD) -b ¥
    +e CICS_XA_Init ¥
    -o sybasexa ¥
    sybasexa.o ¥
    $(CICS_LIBDIR)/regxa_swxa.o ¥
    -Bimmediate -Bnonfatal +s +b/opt/cics/lib ¥
    $(SYB_LIBDIR)/libxadtm.a ¥
    $(SYB_LIBDIR)/libct_r.a ¥
    $(SYB_LIBDIR)/libcs_r.sl ¥
    $(SYB_LIBDIR)/libtcl_r.a ¥
    $(SYB_LIBDIR)/libcomn_r.sl ¥
    $(SYB_LIBDIR)/libintl_r.sl ¥
    -lm ¥
    $(CICS_LIBDIR)/libcicsrt.sl ¥
    -lc

sybasexa.o:sybasexa.c
    $(CC) -c $(CCOPTS)¥
    -I$(ENCINA)/include sybasexa.c
```

注意 共有可能なバージョンの CS-Library (*libcs_r.sl*) と Common Library (*libcomn_dce.sl*) を使用してください。

スイッチ・ロード・テーブルの構築には、ANSI C コンパイラが必要です。

Sun Solaris 2.x (SPARC) でのスイッチロード・ファイルのコンパイル

$SYBASE/$SYBASE_OCS/sample_dtm/cics/switch にある makefile *sybasexa_sol.mk* を使用して、*sybasexa.c* をコンパイルします。

次は、*sybasexa_sol.mk* の内容です。使用している設定に合わせて編集してください。

```
#Makefile to compile the LoadSwitchTable
#This makefile should be run with the command "make -f
sybasexa_sol.mk"

SYB_LIBDIR = $(SYBASE)/lib
SYBLIBS = lxadtm -lct_r -lcs_r.so -ltcl_r -lcomn_r.so
-lintl_r -lxdsxom -lm

all:sybasexa.c
/bin/xlc_r -v -D_THREAD_SAFE ¥
-I /usr/lpp/encina/include sybasexa.c ¥
-o sybasexa ¥
-esybasexa ¥
-L/usr/lpp/cics/lib ¥
-L$(SYBLIBS) ¥
-lcicsrt ¥
/usr/lpp/cics/lib/regxa_swxa.o -ldce
```

CICS 領域 XAD 定義への Sybase スタンザの追加

CICS TM は、CICS XAD 情報を使用して他の RM と通信します。XAD 定義には、LRM ごとに Sybase スタンザが 1 つ含まれています。XAD スタンザの属性については、CICS のマニュアルを参照してください。

以下に、Sybase XAD スタンザのサンプルを 2 つ示します。CICS 領域にスタンザを追加するには、SMIT ユーティリティを使用してください。

```
betaOne:
GroupName=""
ActivateOnStartup=yes
ResourceDescription="XA Product Definition"
AmendCounter=2
Permanent=no
SwitchLoadFile="/usr/lpp/sybase/sample/xa_library/
                cics/switch/sybasexa"
XAOpen="-Uuser_1 -Ppassword_1 -Nconnection_1"
XAClose="ignored"
XASerialize=all_operations


betaTwo:
GroupName=""
ActivateOnStartup=yes
ResourceDescription="XA Product Definition"
AmendCounter=2
Permanent=no
SwitchLoadFile="/usr/lpp/sybase/sample/xa_library/
                cics/switch/sybasexa"
XAOpen="-Uuser_2 -Ppassword_2 -Nconnection_2"
XAClose="ignored"
XASerialize=all_operations
```

次のフィールドは設定に合わせて変更する必要があります。

- SwitchLoadFile
- XAOpen
- XAClose
- XASerialize

注意　すべての Sybase スタンザで同じスイッチロード・ファイルを使用できます。

XAD 定義の XAOpen 文字列で指定される内容については、「DTM XA インタフェース用のオープン文字列パラメータ」(16 ページ) を参照してください。

Encina での DTM XA インタフェースの使用

この項では、Encina で使用する場合のオープン文字列の割り当てと RM の初期化を行う方法を説明します。XA 設定ファイルの作成については、「DTM XA インタフェース用の XA 設定ファイル」(18 ページ) を参照してください。

monadmin create rm によるオープン文字列の割り当て

monadmin create rm コマンドを使用して、各 LRM に「オープン文字列」形式のユーザ名とパスワードを割り当てます。このユーザ名とパスワードによって、特定の接続を介した Adaptive Server リソースへのアクセスを DTM XA インタフェースで管理できます。オープン文字列の内容の詳細については、「DTM XA インタフェース用のオープン文字列パラメータ」(16 ページ) を参照してください。

以下は、monadmin create rm セッションのサンプル画面の内容です。

```
echo "Creating connection_1 resource manager record"
monadmin delete rm connection_1 >>& demo_conf.log
monadmin create rm connection_1¥
-open "-Usa -Psecret -Nconnection_1" ¥
-close "not used" >>& ¥
demo_conf.log
if ($status) then
echo "Failed to create lrm_1 resource mgr.";
exit 1;
endif
```

使用しているバージョンの Encina に、RM を指定するコマンドが追加されていることがあります。詳細については、『Encina Monitor System Administrator's Guide and Reference』を参照してください。

注意 シェル・コマンド monadmin の代わりに、Encina 対話型コマンド enconsole を使用できます。

---

mon_RegisterRmi による LRM の初期化

Encina Monitor アプリケーション・サーバ内から、mon_RegisterRmi の呼び出しによって各 LRM を登録する必要があります。次に例を示します。

```
status =
mon_RegisterRmi(&sybase_TXS_xa_switch,"connection_1",
&rmiID);
if (status != MON_SUCCESS)
{
fprintf(stderr, "mon_RegisterRmi failed (%s).¥n",
mon_StatusToString(status));
```

```
bde_Exit(1);
}
fprintf(stderr, "mon_RegisterRmi complete¥n");
```

monadmin create rm コマンドによって登録する各 LRM には、LRM を初期化する mon_RegisterRmi コマンドが必要です。monadmin create rm コマンド内で指定する *rmname* は、mon_RegisterRmi コマンド内の *rmname* と一致していなければなりません。

以下については、『Encina Monitor Programmer's Guide』を参照してください。

- 登録関数によって実行されるタスクと、その実行順序
- mon_RegisterRmi コマンドの完全な構文

アプリケーションと DTM XA インタフェース・ライブラリとのリンク

アプリケーションを DTM XA インタフェース・ライブラリ *libxadtm.a* とリンクします。

接続の確立

TM は DTM XA インタフェース・ライブラリとともに、いくつかの手順に従ってアプリケーションと RM 間の接続を確立します。

1　アプリケーションは、mon_RegisterRmi 関数を発行して、LRM の使用を要求します。

2　オープン文字列に含まれている情報を使用して、TM から LRM に XA オープン呼び出しが発行されます。この LRM は、monadmin create rm コマンドに設定されていて、その名前は前述の手順 1 で発行したものと一致します。

3　TM は、各 monadmin create rm コマンドに関連付けられているオープン文字列を DTM XA インタフェースに渡します。オープン文字列には LRM 名が含まれています。

4　DTM XA インタフェースは XA 設定ファイルから LRM 名を検索し、この LRM 名を実際の RM 名 (つまり、実際の物理的な Adaptive Server) と一致させます。RM 名は、Adaptive Server の *interfaces* ファイルにあるエントリと一致します。

5　DTM XA インタフェースは、Adaptive Server への論理接続を各 LRM エントリに 1 つ確立します。そして、LRM 用に設定された事前接続のプロパティと機能を各接続に与えます。

TUXEDO での DTM XA インタフェースの使用

この項では、XA インタフェースを TUXEDO に統合する場合に必要となる、アプリケーションに特有の手順を説明します。

統合の手順の中で、アプリケーションによって異なるのは次の部分です。

- アプリケーションからアプリケーション・サーバへのリンク
- *UBBCONFIG* ファイルの設定
- トランザクション・モニタ・サーバ (TMS) の構築
- リソース・マネージャへのアプリケーション・サーバの統合

ここでは、TUXEDO が *$TUXDIR* ディレクトリにインストールされ、何らかのリソース・マネージャもシステムにインストールされていると想定します。

注意 次の説明では、環境変数を TUXEDO の実際のパスに置き換えて読んでください。*$TUXDIR* は実際のルート・ディレクトリ、*$SYBASE* は DTM XA インタフェースのインストール・ディレクトリになります。

表 3-4 は、TUXEDO の統合を行うために必要な、Sybase に特有の情報の一覧です。この情報については、『TUXEDO Installation Guide』の「Integrating a Resource Manager With System/T」に記載されています。

表 *3-4: TUXEDO* システムの統合に必要な情報

情報のタイプ	Sybase の場合	説明
RM 名	SYBASE_XA_SERVER	xa_switch_t 構造体の *name* 要素内にあるリソース・マネージャの名前。
XA 構造体名	sybase_TUX_xa_switch	リソース・マネージャ識別子、リソース・マネージャの機能のフラグ、XA 関数の関数ポインタが含まれる xa_switch_t 構造体の名前。
ライブラリ名	*$SYBASE/$SYBASE_OCS/lib* にある、ライブラリ・ファイル *ct_r*、*cs_r*、*comn_r*、*tcl_r*、*intl_r*。	DTM XA インタフェースをサポートするために必要なファイルの一覧とフル・パス名。
オープン文字列の内容	詳細については、このマニュアルの「DTM XA インタフェース用のオープン文字列パラメータ」を参照。	関数に渡される情報文字列の形式。

注意 DTM XA でのリエントラント・ライブラリ (*ct_r*、*cs_r*、*comn_r*、*tcl_r*、*intl_r* など) 使用の必要性は、スレッド・アプリケーションを開発しているかどうかによります。スレッド・アプリケーションには、リエントラント・ライブラリを使用する必要があります。非マルチスレッド・アプリケーションの場合は、非リエントラント・ライブラリ (*ct*、*cs*、*comn*、*tcl*、*intl* など) を使用できます。

XA 設定ファイルの作成については、「DTM XA インタフェース用の XA 設定ファイル」(18 ページ) を参照してください。

リンク

TUXEDO RM ファイルには、TUXEDO ユーティリティが TUXEDO サーバにリンクするために使用する情報が含まれます。RM ファイルに、Sybase アプリケーションにリンクするための適切な指定が含まれているかを確認してください。

1　任意のテキスト・エディタを使用して *$TUXDIR/udataobj/RM* ファイルを開き、編集してください。

2　Sybase リソース・マネージャ用のエントリを追加／検証し、XA 情報でファイルを更新します。simprpc.ct サンプル・アプリケーションを含め、ほとんどの Sybase アプリケーションでは、SYBASE_XA_SERVER 用のエントリが 1 つあるだけで十分です。rentapp サンプルを構築して実行する場合は、rentapp に必要な、SCRAP_XA_SERVER 用のエントリも追加する必要があります。

$SYBASE/$SYBASE_OCS を、XA インタフェースが含まれている Sybase インストール・ディレクトリへの完全に修飾されたパスで次のように置き換えてください。

```
SYBASE_XA_SERVER:sybase_TUX_xa_switch:-t -Bstatic -L$SYBASE/$SYBASE_OCS/lib
-lcobct -lxadtm -lct_r -lcs_r -lcomn_r -ltcl_r -lintl_r -Bdynamic -ldl
SCRAP_XA_SERVER:sybase_TUX_xa_switch:-t -Bstatic -L$SYBASE/$SYBASE_OCS/lib
-lcobct -lxadtm -lct_r -lcs_r -lcomn_r -ltcl_r -lintl_r -Bdynamic -ldl
```

注意　各エントリは、続けて 1 行に入力してください。

cobct ライブラリが必要なのは、ESQL/COBOL アプリケーション・サーバを構築する場合のみです。ESQL/COBOL を使用しない場合、-lcobct の指定は省略できます。

TUXEDO サーバですべての Sybase ライブラリを動的にロードして実行するには、次のようにエントリを指定します。ダイナミック・ライブラリを使用すると、TUXEDO サーバ実行時の CPU 負荷が増加することがあります。

```
 SYBASE_XA_SERVER:sybase_TUX_xa_switch:-L$SYBASE/$SYBASE_OCS/lib -lxadtm
-lct_r -lcobct -lcs_r -lcomn_r -ltcl_r -lintl_r
 SCRAP_XA_SERVER:sybase_TUX_xa_switch:-L$SYBASE/$SYBASE_OCS/lib -lxadtm
-lct_r -lcobct -lcs_r -lcomn_r -ltcl_r -lintl_r
```

注意　各エントリは、続けて 1 行に入力してください。

行頭にシャープ記号 (#) を付けて、コメント行にすることもできます。

UBBCONFIG ファイルの設定

この項では、XA インタフェースで TUXEDO *UBBCONFIG* ファイルを設定する場合の具体的な例を示します。

simprpc.ct サンプル・アプリケーションでは、Adaptive Server に pubs2 データベースがインストールされている必要があります。Adaptive Server ディレクトリの *scripts/installpubs2* にあるインストール・スクリプトを使用してください。

「DTM XA インタフェース用のオープン文字列パラメータ」(16 ページ) では、*UBBCONFIG* ファイル内のオープン文字列について説明します。

1 任意の ASCII テキスト・エディタを使用して *$SYBASE/$SYBASE_OCS/sample/xa-dtm/tuxedo/simprpc.ct/ubbsimpct* を開き、編集します。ここでは説明の便宜上、各行に番号を付けて示します。

```
1  *RESOURCES
2  IPCKEY          123456
3
4  MASTER          sybsite
5  MAXACCESSERS    5
6  MAXSERVERS      5
7  MAXSERVICES     10
8  MODEL           SHM
9
10 MAXGTT          5
11
12 *MACHINES
13 yourmachine     LMID=sybsite
14 TUXDIR="$TUXDIR"
15 APPDIR="$SYBASE/$SYBASE_OCS/sample/xa-dtm/tuxedo/simprpc.ct"
16 TLOGDEVICE="$SYBASE/$SYBASE_OCS/sample/xadtm/tuxedo/
   simprpc.ct/tuxlog"
17 TLOGNAME=TLOG
18 TUXCONFIG="$SYBASE/$SYBASE_OCS/sample/xa-dtm/tuxedo
   /simprpc.ct/tuxconfig"
19 ULOGPFX="$SYBASE/$SYBASE_OCS/sample/xa-dtm/tuxedo/simprpc.ct/ULOG"
20
21 *GROUPS
22 DEFAULT:        TMSNAME=simprpccttms TMSCOUNT=2
23
24 GROUP1          LMID=sybsite    GRPNO=1
25 OPENINFO="SYBASE_XA_SERVER:-Uuserid1 -Ppassword1 -Nconnection1"
26
27 *SERVERS
28 simpsrv         SRVGRP=GROUP1   SRVID=1
29
30 *SERVICES
```

2　表 3-5 に従って、ファイル内のエントリを環境に応じたエントリで置き換えます。

表 3-5: 環境固有のファイル・エントリ

行番号	エントリ	置換後のエントリ
13	*yourmachine*	XA インタフェースがインストールされているマシンの名前。マシン名では大文字と小文字が区別される。
14	*$TUXDIR*	実際の TUXEDO ルート・ディレクトリのパス。
15, 16, 18, 19	*$SYBASE/$SYBASE_OCS*	XA インタフェースのインストール・ディレクトリ。
22	simprpcttms	このパラメータは simprpc.ct サンプルの場合のもの。通常このパラメータは「TMS の構築」(33 ページ) で説明する buildtms コマンドの -o パラメータで指定された値と関連させる必要がある。
25	オープン文字列パラメータ	詳細については、「DTM XA インタフェース用のオープン文字列パラメータ」(16 ページ) を参照してください。

注意　*UBBCONFIG* ファイルの詳細については、『TUXEDO Installation Guide』を参照してください。

TUXEDO 設定ファイルの作成

次のコマンドにより、*$TUXCONFIG* 環境変数の値を、*ubbsimpct* 内のエントリと一致する値に設定します。

```
setenv TUXCONFIG $SYBASE/$SYBASE_OCS/sample/xa-dtm/tuxedo/simprpc.ct/tuxconfig
```

次のコマンドを実行して、*UBBCONFIG* ファイルから TUXEDO 設定ファイルを作成します。

```
$TUXDIR/bin/tmloadcf -y ubbconfig_file_name
```

simprpc.ct サンプルによる検証では、*ubbconfig_file_name* を *ubbsimpct* で置き換えてください。

TMS の構築

トランザクション・モニタ・サーバ (TMS) を構築するには、次のコマンドを実行します。*output_file* は、トランザクション・モニタ・サーバ・プログラムに付ける名前です。

```
$TUXDIR/bin/buildtms -r SYBASE_XA_SERVER -o
$TUXDIR/bin/output_filetms
```

上記の例のように名前に tms を含めると、識別しやすくなります。ほかのリソース・マネージャの TMS プログラムと重複しないように、ユニークな名前を付けてください (TMS、TMS_D、TMS_SQL は予約されています)。

simprpc.ct サンプルによる検証の場合、*UBBCONFIG* ファイルは、表 3-5 (33 ページ) で説明している simprpccttms を使用します。

このプログラムは *$TUXDIR/bin* に格納されており、TUXEDO System/T 起動プログラムで検出できます。

COBOL 実行環境の構築

CICS トランザクションの中で、COBOL トランザクションは COBOL ランタイムを使用します。Sybase XA 環境と通信するためには、COBOL ランタイムに修正を加える必要があります。

❖ **Sybase XA COBOL トランザクションをサポートするように CICS を設定する**

1 root としてログインします。

2 *COBDIR* 環境変数を、MicroFocus COBOL 環境のディレクトリ・パスに設定します。

3 *PATH* 環境変数を、MicroFocus COBOL バイナリ・ディレクトリを含むように設定します。

4 ディレクトリを *$SYBASE/$SYBASE_OCS/sample/xa-dtm/cics* に変更します。

5 *xa_make_cobol_runtime* を実行します。

警告！ このスクリプトは、CICS COBOL ランタイム・ファイルが */usr/lpp/cics/v1.1/bin* にインストールされていることを前提としています。CICS のインストール先が異なる場合は、それに応じてスクリプトを編集する必要があります。

このスクリプトは、CICS と Sybase XA をサポートする MicroFocus COBOL ランタイム環境を構築します。COBOL で書かれた CICS トランザクションで、ランタイムに XA インタフェースや Open Client の関数を参照することができます。このスクリプトの実行には数分かかります。詳細については、CICS のマニュアルを参照してください。

注意 MicroFocus COBOL 3.1 以降を使用する必要があります。

第 4 章

アプリケーション・プログラミングのガイドライン

Embedded SQL アプリケーションと Client-Library アプリケーションを Sybase XA 環境内で実行できるようにするには、アプリケーションのコードを特定のコーディング制約に準拠させる必要があります。この章では、前述の制約について説明するとともに、Client-Library でのコーディング例を 1 つ、Embedded SQL でのコーディング例を 2 つ示します。

X/Open DTP と従来の Sybase でのトランザクション処理の違い

X/Open DTP モデルでのトランザクション処理は、従来の Sybase モデルの場合とは大きく異なります。従来の Sybase トランザクション処理 (TP) 環境は、接続指向型でした。プログラムは、接続管理用の SQL 文を使って、アプリケーション・プログラムと Adaptive Server 間の接続を直接設定していました。しかし XA インタフェース環境では、XA インタフェースが LRM を使ってアプリケーションの接続を設定します。

表 4-1 に、従来の TP モデルと X/Open DTP モデルの相違点を示します。

表 4-1: 従来の *TP* モデルと *X/Open DTP* モデルとの相違点

従来の TP モデル	X/Open DTP モデル
1 つのクライアント／サーバ接続について 1 つ以上のトランザクションがある。	接続という概念がない。コンポーネントはインタフェースを介して通信する。
トランザクションは通常はローカルなものであり、トランザクションと Adaptive Server の関係は 1 対 1 に限定される。	トランザクションはグローバルなものである。トランザクションは複数のリソース・マネージャに影響を与える。1 つのトランザクション内で実施した作業を完了するときに、複数のリソース・マネージャを使用する。
各 Adaptive Server は、それぞれのサーバに存在するデータのリカバリを行う。	トランザクション・マネージャは、すべてのリソース・マネージャに格納されているデータのリカバリを行う。

トランザクションと接続の管理

アプリケーションは、以下に関係するコマンドに対して特に注意を払う必要があります。

- トランザクションの管理
- 接続の管理
- 現在の接続

注意 XA インタフェースでは、ANSI のデフォルト独立性レベル 3 を使用します。読み込み専用ロックを最小限に抑えるために、プログラムでは、トランザクションの独立性レベルを XA 設定ファイルに設定できます。また、個々の SQL オペレーションで select xxx from table noholdlock を使用できます。トランザクションの独立性レベルの詳細については、『Transact-SQL ユーザーズ・ガイド』を参照してください。

トランザクションの管理

CICS、Encina、または TUXEDO の TM はトランザクション管理を行います。トランザクション管理では、グローバル・トランザクションを作成して、その中でアプリケーションの作業のすべてをコミットまたはロールバックします。したがって、アプリケーションは、トランザクションを管理する SQL 文は発行できません。

たとえば、次の Embedded SQL コマンドは、アプリケーションから発行できません。

- begin transaction
- commit
- rollback

Client-Library アプリケーションは、(ct_command、ct_dynamic、または ct_cursor を使用して) 以下の Transact-SQL コマンドを実行することはできません。

- begin transaction
- commit transaction
- rollback transaction
- set (chained、noexec、isolation、parseonly、statistics io、statistics time)
- save transaction

注意　アプリケーションは、Adaptive Server で検出されたエラーを認識したり、TM を使ってトランザクションをアボートまたはロールバックしたりする必要があります。この動作は、Adaptive Server でデッドロックが検出された場合に特に重要です。

接続の管理

アプリケーションは、クライアント／サーバ接続の管理については Sybase XA 環境に依存しています。接続管理はアプリケーションに対して透過的に発生します。したがって、Embedded SQL アプリケーションは、XA が管理する接続については次のコマンドを呼び出すことができません。

- connect
- disconnect

Client-Library アプリケーションは、XA が管理する接続を使用して次の Client-Library コマンドを呼び出すことができません。

- ct_close
- ct_con_alloc
- ct_con_drop
- ct_con_props

- ct_config (次のパラメータを指定したもの)
  - CS_ENDPOINT
  - CS_EXPOSE_FMTS
  - CS_HIDDENKEYS
  - CS_MAX_CONNECT
  - CS_NETIO
  - CS_TRANSACTION_NAME
- ct_connect
- ct_exit
- ct_getloginfo
- ct_init
- ct_options (次のパラメータを指定したもの)
  - CS_OPT_CHAINXACTS
  - CS_OPT_FORCEPLAN
  - CS_OPT_FORMATONLY
  - CS_OPT_NOEXEC
  - CS_OPT_PARSEONLY
  - CS_OPT_STATS_IO
- ct_remote_pwd
- ct_setloginfo
- CS_OPT_STATS_TIME

さらに、Client-Library アプリケーションでは、次の CS-Library コマンドも呼び出すことができません。

- cs_ctx_drop (グローバル・コンテキスト・ハンドルを指定)
- cs_objects (CS_CLEAR、CS_SET)

現在の接続

Open Client Embedded SQL のマニュアルで説明していますが、デフォルト接続という概念は Sybase XA 環境には存在しません。したがって、アプリケーションは、常に明示的に現在の接続を指定する必要があります。

Embedded SQL で現在の接続を指定する方法には、次の 2 つがあります。

- set connection コマンド
- at *connection name* 句

現在の接続は、複数のトランザクションには影響しません。たとえば、CICS の SYNCPOINT コマンドや Encina の onCommit コマンドを実行する場合は、コマンド実行後に毎回、アプリケーションは現在の接続を再設定する必要があります。現在の接続の範囲を明確にするために、Sybase ではすべての Embedded SQL 文に at *connection_name* 句を使用することをおすすめします。

非トランザクション指向の接続

アプリケーションは、Open Client または Embedded SQL の標準インタフェースを介して、非トランザクション指向の接続を開いたり使用したりできます。非トランザクション指向の接続上で行われるオペレーションはトランザクションとは関係ないので、コミットもロールバックも行われません。非トランザクション指向の接続が適しているのは、変化しないデータベースへのクエリを行う場合と、正確さが疑わしいデータを更新する場合です。

Client-Library でのカーソルの割り付け解除

アプリケーション・プログラムでは、そのプログラム用に XA インタフェースを介して割り付けられた接続を使用／再使用します。SQL Server バージョン 10.1 をはじめとする Sybase のカーソル実装を使用する場合は、クライアント側 (TM/RM プログラム) および Adaptive Server 側の両方にカーソル構造体が必要です。

クライアントが明示的にカーソルの割り付けを解除した場合、またはクライアントの接続が閉じた場合、Adaptive Server はサーバのカーソル構造体の割り付けを解除します。

プログラムを最初に実行するときにカーソルを開くか閉じるかしたが、(XA-Library の場合と同じように) 接続が割り付けられたままの場合は、次に同じプログラムを実行すると、同じ名前のカーソルを開こうとしてエラーになります。Adaptive Server は、開こうとしたカーソルと同じネスト・レベルにこの名前のカーソルがすでに存在することを通知します。

アプリケーション・プログラムがトランザクションをコミットまたはアボートする場合は、あらかじめ、明示的にカーソルを閉じてカーソルの割り付けを解除しておく必要があります。カーソルの割り付け解除は、カーソルを割り付けたトランザクション・プログラムで行ってください。Embedded SQL は、XA インタフェースを使って割り付け解除を実行するカーソルについて、情報を記録します。

Client-Library を使用する場合は、TM アボート・コードの呼び出し前にカーソルの割り付けが解除されるように、エラー・パスを処理する必要があります。つまり、開いたカーソルが動作している場合は、そのカーソルの割り付けを解除してください。

ct_cursor( ) を使用する場合、タイプには CS_CURSOR_CLOSE、オプションには CS_DEALLOC を指定します。

動的 SQL

動的 SQL 文の使用法には、カーソルの使用法と多くの共通点があります。ただし、テンポラリ・ストアド・プロシージャが Adaptive Server に配置されることがあるという点で、動的 SQL 文の方が複雑です。トランザクション指向のアプリケーションに動的 SQL を使用することはおすすめできません。使用する場合は、次のガイドラインに従ってください。

- Embedded SQL の場合は、「メソッド 3：カーソルによる Prepare と Fetch」(ESQL のマニュアル、またはメソッド 3 の説明を参照) をなるべく使用してください。メソッド 3 を使用すると、Embedded SQL によって情報がシステムに格納されます。このシステムでは、XA インタフェースを使用して、動的 SQL とカーソルのすべての割り付け／割り付け解除を行います。
- 上記以外の場合は、他のトランザクションのパフォーマンスが低下しないように、動的 SQL 文とすべての関連するカーソルを閉じ、割り付けを解除してください。メモリ・リークを回避するため、関連する Client-Library コマンド構造体をすべて削除する必要があります。これらのコマンド構造体の削除方法については、Open Client および ESQL のマニュアルを参照してください。

Client-Library 接続ハンドルの取得

接続ハンドルの取得は、Client-Library アプリケーション特有の問題です。

TM によって Adaptive Server への接続が開かれると、XA インタフェースは専用で使用する CS_CONNECTION 構造体を割り付けます。アプリケーションに制御が移った場合、そのアプリケーションは、上記の構造体に含まれている接続ハンドルを使用する必要があります。

接続ハンドルを取得するには、cs_object ルーチンの action パラメータに CS_GET を指定します。オブジェクト・タイプは CS_CONNECTION です。cs_object の objdata パラメータは、connection フィールドを含んだ構造体を返します。このフィールドには、CS_CONNECTION 構造体へのハンドルが含まれています。

警告！ XA インタフェースでは CS_COMMAND 構造体も割り付けます。この構造体のハンドルは、構造体の中にある command フィールドに返されます。構造体へのポインタは、objdata パラメータに含まれています。XA インタフェース自体がこのハンドルを継続して使用するので、アプリケーションはこのコマンド・ハンドルを使用できません。

CS_CONNECTION 構造体へのハンドルを取得する場合は、次のコーディング例を参考にしてください。

```
/*
 ** Arguments:
 **connection null-terminated name of the connection
 **(ESQL) or LRM connHloaded with the CS_CONNECTION
 ** handle if the lookup is successful
 **
 ** Returns:
 ** CS_SUCCEED connection handle found successfully
 ** CS_FAIL unable to find connection handle for given
 ** connection /#include <stdio.h> #include <strings.h>
 ** #include <cspublic.h>CS_RETCODE getConn(connection,
 ** connH)CS_CHAR connection[128];CS_CONNECTION connH;
 {
 CS_INT retcode;
 CS_CONTEXT *ctx;
 CS_OBJNAME name;
 CS_OBJDATA data;
 CS_THREAD thread_functions;
 CS_INT outlen;
 #define THREADID_SIZE 8
 CS_BYTE thread_id[THREADID_SIZE];
 /* Check arguments */
 if (strlen(connection) >= 128)
 {
 /* Connection name is too long */
 return(CS_FAIL);
```

```
}
/* Get the global context handle */
retcode = cs_ctx_global(CS_VERSION_100, &ctx);
if (retcode != CS_SUCCEED)
{
/* Major environment problems!*/
return(CS_FAIL)
}
/*
** Initialize the CS_OBJNAME structure to look
** for the specified connection name.
*/
name.thinkexists = CS_FALSE;
name.object_type = CS_CONNECTNAME;
strcpy(name.last_name, connection);
name.fnlen = CS_UNUSED;
name.lnlen = CS_NULLTERM;
name.scopelen = CS_UNUSED;
/*
** Set the current thread-id so we get the instance of
** this connection that this thread should be using.
*/
retcode = cs_config(ctx, CS_GET,
CS_THREAD_RESOURCE, &thread_functions,
CS_UNUSED, &outlen);
if (retcode != CS_SUCCEED)
{
/*
** Even in an non-threaded environment,this should be
** successful.
*/
return(CS_FAIL);
}
name.thread = (CS_VOID *) thread_id;
retcode = (*thread_functions.thread_id_fn)(
name.thread, THREADID_SIZE,
&name.threadlen);
if (retcode != CS_SUCCEED)
{
return(CS_FAIL);
}
/*
** Initialize the CS_OBJNAME structure to look
** connection handle for this connection name
*/
data.actuallyexists = CS_FALSE;
data.connection = (CS_CONNECTION *) NULL;
data.command = (CS_COMMAND *) NULL;
data.buffer = (CS_VOID *) NULL;
data.buflen = CS_UNUSED;
/* Retrieve the connection information */
```

```
 retcode = cs_objects(ctx, CS_GET, &name,

 &data);
 if (retcode == CS_SUCCEED)
 {
 if (data.actuallyexists == CS_TRUE)
 {
*connH = data.connection;
return(CS_SUCCEED);
}
else
{
/* No connection by that name exists */
return(CS_FAIL);
}
}
else
{
/*
** The global CS_CONTEXT handle is probably not
** initialized with connection information yet
*/
return(CS_FAIL);
}
}
```

マルチスレッド環境の問題

スレッドは、1 つのオペレーティング・システム・プロセスにおいて、複数かつ同時に発生する実行のパスです。そのプロセスに割り付けられているリソースへのアクセスは、スレッド間で共有されます。

アプリケーション・プログラミング・インタフェース (API) の中には、アプリケーション開発者がトランザクション環境で効率的にスレッドを使用できるものがあります。Sybase では、XA インタフェースによって最大レベルの同時実行性をサポートして、マルチスレッド環境を有効活用します。

ただし、アプリケーション開発者にとっては問題もあります。マルチスレッドの問題の基本的な情報や詳細については、OSF の『DCE Application Developer's Guide』を参照してください。

Open Client のリファレンス・マニュアルには、スレッドセーフ・プログラミングの項があります。XA インタフェースは、TM の要求があったときにスレッドへの接続を割り当てます。この割り当てによって、接続上で同時に機能するスレッドが確実に 1 つだけに限定されます。「Client-Library 接続ハンドルの取得」(41 ページ) で説明されている cs_object 要求にスレッド ID が含まれているのは、このためです。XA インタフェースが割り当てた接続を、接続が割り当てられているスレッドで使用していて、かつスレッドの使用制限に従っているかぎりは、Open Client や ESQL のスレッド関連の問題は起こりません。

スレッドの使用に関する注意事項

Client-Library は、接続状態のマシンを使用して、アプリケーションが Client-Library ルーチンを論理的な順序で呼び出すことを確かめます。Client-Library アプリケーションの構築に関係する手順については、『Open Client Client-Library/C プログラマーズ・ガイド』を参照してください。

スレッドの使用に関して想定されていることは、スレッドとトランザクション・ブランチとの関連付けが解除されると、現在のマシンは非アクティブな状態のままになるということです。デフォルトでは、すべての Embedded SQL 文が接続をクワイス状態にします。Client-Library では、これは以下の場合にのみ当てはまります。

- ct_results が CS_END_RESULTS を返す場合、または、結果タイプ CS_SUCCEED を伴った CS_CURSOR_RESULT を返す場合。
- アプリケーションが *type* に CS_CANCEL_ALL を指定して ct_cancel を呼び出したあと。
- アプリケーションが CS_CANCELED を返す場合。CS_CANCELED を返す API には、ct_send()、ct_results()、ct_get_data() が含まれる。

警告！ 接続が非アクティブ状態になっていない場合は、その結果として、トランザクション・ロールバック、XA インタフェースによる接続クリーンアップ時のオーバヘッドの増大 (この場合、すべての接続を閉じてから、もう一度接続を開く必要があります)、後続トランザクションのエラーが発生する可能性があります。こうした状況下で、XA インタフェースはアプリケーションのオペレーションを維持しようとしながら、エラーを最小限に抑えます。

Embedded SQL のスレッドセーフ・コード

スレッドセーフ・コードは、ミューテックス (mutex: MUTual EXclusion semaphore (相互排他セマフォ) の略) を使った共有リソースの使用を保護します。ミューテックスは、複数スレッドからの同時アクセスを防止することによって、ファイルやグローバル変数などの共有リソースを保護します。

スレッドセーフ・コード作成時のプリコンパイラ・オプションには、-h (UNIX の場合)、または /threadsafe (VMS の場合) を使用します。

密結合トランザクション

XA 環境では、1 つのトランザクションにおいて機能する各スレッドまたは各プロセスを、トランザクション・ブランチとして処理します。トランザクション・ブランチにはそれぞれ異なる xid が割り当てられ、他のブランチとは無関係に機能します。ただし、すべてのブランチは、1 つの単位としてコミットまたはロールバックされます。これは MTS/COM+ 環境において、バージョン 3.6 よりも古い ctlib ベースのドライバを使用している場合のみに当てはまります。

TM には、ブランチを密結合させるものがあります。複数の密結合ブランチには同じ xid が割り当てられ、トランザクションではともに動作します。この場合、オープン文字列には -O1 を指定できます。このオプションによって、Adaptive Server は、必要に応じて複数の接続間で動作を切り替えたり、複数の接続間でロックが発生しそうな場合はそれを防止したりします。TM を密結合オペレーション用に設定する方法については、TM のマニュアルを参照してください。

警告！ 更新の競合が起こらないようなアプリケーション設計が保証されている場合にかぎり、-O1 オプションを設定してください。通常、このことが当てはまるのは、アプリケーション・ブランチが完全に直列化されている (たとえば、ブランチ A が完了するまでブランチ B が実行されない) 場合だけです。密結合ブランチ間の対話がうまく設計されていないと、データの矛盾が発生することがあります。

-O1 オプションを指定しないと、複数のブランチがデータベース内の同じローを更新しようとした場合に、トランザクション内でデッドロック状態になります。TM を使ってブランチが密結合されていない状態で、各ブランチに別々の xids が割り当てられている場合は、-O1 オプションを指定しても実際には効果がありません。

警告！ トランザクションを別の接続に割り当てた場合、カーソルと動的 SQL は保持されません。したがって、トランザクション上で他のブランチを一切実行しないときにカーソルと動的 SQL を開閉するようなアプリケーション構造になっている場合を除き、カーソルと動的 SQL は使用しないでください。

注意 トランザクションは、SQL バッチ間でのみ、別の接続に再割り当てされます。密結合アプリケーションは、すべてのオペレーションを 1 つのバッチとして実行することによって、競合することなく一連のオペレーションを完了させることができます。これは、1 つのストアド・プロシージャ内にあるオペレーションもまた、競合なしで完了することを意味します。

たとえば、テーブル B のロー z には、テーブル A のロー x とロー y の合計が入るとします。次の例では、ロー z に入る値が正しくありません。

```
Branch 1:                          Branch 2
    Updates Row x -> 5
    Reads Row y (= 4)
                                   Updates Row y -> 5
                                   Reads Row x (= 5)
                                   Updates Row z -> 10
    Updates Row z -> 9 (wrong value)
```

ブランチを直列で実行していれば、エラーにはなりません。

```
Branch 1:                          Branch 2
    Updates Row x -> 5
    Reads Row y (= 4)
    Updates Row z -> 9
                                   Updates Row y -> 5
                                   Reads Row x (= 5)
                                   Updates Row z -> 10
```

また、エラー対策として、制御ブランチを使用することもできます。

```
Branch 0:            Branch 1:      Branch 2
(controller)
Starts Branches 1 and 2
Waits for both to complete
                     Updates Row x  Updates Row y -> 5
                     Terminates     Terminates
    Reads Row y
    Reads Row x
    Updates Row z
```

TM 固有のブランチ制御メカニズムは、こうした直列化メカニズムを実装するために使用してください。

CT-Library とのリンク

XA インタフェースでは、アプリケーションがスレッド・バージョンの Open Client ライブラリにリンクしていることが必要です。指定する必要があるライブラリを確認するには、使用しているプラットフォームの『Open Client/Server プログラマーズ・ガイド補足』を参照してください。適切なスレッドセーフ・ライブラリをリンクしていない場合は、さまざまな Open Client エラーが発生します。

Embedded SQL/COBOL のサンプル・コード

このコーディング例では、現在の接続を設定し、Adaptive Server データベースにデータを挿入します。

```
*REMARKS.TRANSACTION-ID IS 'POPS'.
*THIS TRANSACTION POPULATES A DATABASE'S DATA TABLE *WITH STOCK
DATA ENTRIES.


ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
COPY DFHBMSCA.
COPY DFHAID.
COPY AIXCSET.
EXEC SQL INCLUDE SQLCA END-EXEC.
77  RESPONSE        PIC 9(8) COMP.
01  MSG-LIST.
02  MSG-1           PIC X(70) VALUE
'Transaction Failed:Unable To Prime Stock'
```

```
-'Table.'.
02  MSG-2           PIC X(70) VALUE
'Stock Records Added Successfully.'.
01  TRANSFAIL       PIC X(70).

EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01  STOCK-RECORD.
02  STOCK-NUM       PIC X(5).
02  ITEM-DESC       PIC X(30).
02  STOCK-QTY       PIC X(7).
02  UNIT-PRICE PIC S9(4)V99 VALUE ZEROES.
EXEC SQL END DECLARE SECTION END-EXEC.

PROCEDURE DIVISION.
* CHECK BASIC REQUEST TYPE
*
IF EIBAID = DFHCLEAR
EXEC CICS SEND CONTROL FREEKB
END-EXEC
EXEC CICS RETURN
END-EXEC
END-IF.


* MAIN PROCESSING
*SET UP STOCK RECORD DETAILS AND THEN WRITE OUT
*STOCK RECORD.
*
MOVE '31421'TO STOCK-NUM.
MOVE 'Widget (No.7)'TO ITEM-DESC.
MOVE '0050035'TO STOCK-QTY.
MOVE 25.55 TO UNIT-PRICE.
PERFORM WRITE-STOCKREC.

MOVE '43567'TO STOCK-NUM.
MOVE 'Splunkett ZR-1'TO ITEM-DESC.
MOVE '0005782'TO STOCK-QTY.
MOVE 143.79 TO UNIT-PRICE.
PERFORM WRITE-STOCKREC.

EXEC CICS SYNCPOINT
RESP(RESPONSE)
END-EXEC.

IF RESPONSE NOT = DFHRESP(NORMAL)
MOVE MSG-1 TO TRANSFAIL
PERFORM FAIL-TRANS
END-IF.

MOVE MSG-2 TO MSGOUTO.
EXEC CICS SEND MAP('MSGLINE')
```

```
MAPSET('AIXCSET')
FREEKB
END-EXEC.
EXEC CICS RETURN
END-EXEC.
GOBACK.


* ATTEMPT TO WRITE OUT NEW STOCK RECORD.


*
WRITE-STOCKREC.
EXEC SQL SET CONNECTION connection_2
END-EXEC

IF SQLCODE NOT = 0
MOVE MSG-1 TO TRANSFAIL
PERFORM FAIL-TRANS
END-IF.

EXEC SQL INSERT INTO STOCK VALUES (:STOCK-RECORD)
END-EXEC

IF SQLCODE NOT = 0
MOVE MSG-1 TO TRANSFAIL
PERFORM FAIL-TRANS
END-IF.


* IF UNABLE TO APPLY CREATE, END TRANSACTION
* AND DISPLAY REASON FOR FAILURE.
*
FAIL-TRANS.
MOVE TRANSFAIL TO MSGOUTO
EXEC CICS SEND MAP('MSGLINE')
MAPSET('AIXCSET')
FREEKB
END-EXEC
EXEC CICS RETURN
END-EXEC.
```

Embedded SQL/C のサンプル・コード

このコーディング例では、現在の接続を設定し、Adaptive Server に格納されているデータにアクセスします。

```
EXEC SQL INCLUDE sqlca;
int rcode;

EXEC SQL BEGIN DECLARE SECTION;
char name[15];
char supplier[30];
char supplier_address[30];
int order_quantity;

EXEC SQL END DECLARE SECTION;
main()
{
char errmsg[400];
char qmsg[400];
short mlen;

EXEC SQL WHENEVER SQLERROR GOTO :errexit;
EXEC SQL WHENEVER SQLWARNING GOTO :errexit
EXEC SQL WHENEVER NOT FOUND GOTO :errexit

/* Get addressability for EIB... */

/*
** Write record to CICS temporary storage queue...
*/

/* Send the first map */

EXEC CICS SEND MAP("PANEL1") MAPSET("UXA1")
FREEKB ERASE RESP(rcode);
if (rcode != DFHRESP(NORMAL))
EXEC CICS ABEND ABCODE("X001");

/* Receive the response */

EXEC CICS RECEIVE MAP("PANEL1") MAPSET("UXA1")
RESP(rcode);
if (rcode != DFHRESP(NORMAL))
EXEC CICS ABEND ABCODE("X002");

/* Select a record from the table based on user input. */

sprintf(name, "%s", panel1.panel1i.newnamei);
EXEC SQL SET CONNECTION connection_1;
EXEC SQL SELECT name, supplier, supplier_address,order_quantity
```

```
INTO
:name, :supplier, :supplier_address, :order_quantity
FROM cheese
WHERE name = :name;

/* Handle "no rows returned" from SELECT */



if (sqlca.sqlcode == 100)
{
sprintf(panel4.panel4o.messageo, "%s",
NOCHEESE);
EXEC CICS SEND MAP("PANEL4") MAPSET("UXA1") FREEKB ERASE RESP(rcode);
if (rcode != DFHRESP(NORMAL))
EXEC CICS ABEND ABCODE("X009");

EXEC CICS SEND CONTROL FREEKB;
EXEC CICS RETURN;

}
/* Fill in and send the second map */

memset ( &panel2.panel2o, '0',
sizeof(panel2.panel2o));
sprintf(panel2.panel2o.nameo, "%s", name);
sprintf(panel2.panel2o.supplo, "%s",supplier);
sprintf(panel2.panel2o.addresso, "%s",
supplier_address);
sprintf(panel2.panel2o.ordero, "%d", order_quantity);

EXEC CICS SEND MAP("PANEL2") MAPSET("UXA1")
FREEKB ERASE RESP(rcode);
if (rcode != DFHRESP(NORMAL))
EXEC CICS ABEND ABCODE("X003");

/* Receive the response */

if (panel2.panel2i.questi == 'y')

{

/* Send the third map...*/
/* Receive the response...*/
/* Update the database */

order_quantity = atoi(panel3.panel3i.newordi);

EXEC SQL UPDATE cheese
set order_quantity = :order_quantity
where name = :name;
```

```
/* Write a record to the temporary queue */

sprintf(qmsg, "%s", "The cheese table was updated");


mlen = strlen(qmsg);


EXEC CICS WRITEQ TS QUEUE("TEMPXAQ1")
FROM(qmsg) LENGTH(mlen) RESP(rcode);
if (rcode != DFHRESP(NORMAL))
EXEC CICS ABEND ABCODE("X010");
}
else
{

/*
** The user does not wish to update so free the keyboard and return...
*/
}
/* Commit the update */

EXEC CICS SYNCPOINT RESP(rcode);
if (rcode != DFHRESP(NORMAL))
EXEC CICS ABEND ABCODE("X011");

/*
** Send the fourth map confirming successful update...
*/

EXEC CICS RETURN;
errexit:
fprintf(stderr,"error in cheeseland %d¥n",sqlca.sqlcode);

/* Handle general errors */

sprintf(errmsg, "%.60s¥n",sqlca.sqlerrm.sqlerrmc);
strncpy(panel4.panel4o.messageo, errmsg, 60);
sprintf(panel4.panel4o.codeo, "%d", sqlca.sqlcode);

/*
** Send the fourth map with appropriate message...
*/

/* Rollback the transaction */

EXEC CICS SYNCPOINT ROLLBACK;
EXEC CICS SEND CONTROL FREEKB;
EXEC CICS RETURN;
 }
```

索引

記号

数字

A

B

C

D

E

G

I

L

M

O

P

R

S

T

U

V

X

あ

い

お

か

き

く

け

こ

さ

し

す

せ

ち

つ

て

と

ね

は

ひ

ふ