Guide

Distributed Transaction Processing:
Reference Model, Version 3

TECHNICAL GUIDES

X ™

THE *Open* GROUP

[This page intentionally left blank]

X/Open Guide

Distributed Transaction Processing:

Reference Model, Version 3

X/Open Company Ltd.

Contents

List of Figures

Preface

X/Open

X/Open is an independent, worldwide, open systems organisation supported by most of the world's largest information systems suppliers, user organisations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems.

X/Open's strategy for achieving this goal is to combine existing and emerging standards into a comprehensive, integrated, high-value and usable open system environment, called the Common Applications Environment (CAE). This environment covers the standards, above the hardware level, that are needed to support open systems. It provides for portability and interoperability of applications, and so protects investment in existing software while enabling additions and enhancements. It also allows users to move between systems with a minimum of retraining.

X/Open defines this CAE in a set of specifications which include an evolving portfolio of application programming interfaces (APIs) which significantly enhance portability of application programs at the source code level, along with definitions of and references to protocols and protocol profiles which significantly enhance the interoperability of applications and systems.

The X/Open CAE is implemented in real products and recognised by a distinctive trade mark — the X/Open brand — that is licensed by X/Open and may be used on products which have demonstrated their conformance.

X/Open Technical Publications

X/Open publishes a wide range of technical literature, the main part of which is focussed on specification development, but which also includes Guides, Snapshots, Technical Studies, Branding/Testing documents, industry surveys, and business titles.

There are two types of X/Open specification:

- *CAE Specifications*

  CAE (Common Applications Environment) specifications are the stable specifications that form the basis for X/Open-branded products. These specifications are intended to be used widely within the industry for product development and procurement purposes.

  Anyone developing products that implement an X/Open CAE specification can enjoy the benefits of a single, widely supported standard. In addition, they can demonstrate compliance with the majority of X/Open CAE specifications once these specifications are referenced in an X/Open component or profile definition and included in the X/Open branding programme.

  CAE specifications are published as soon as they are developed, not published to coincide with the launch of a particular X/Open brand. By making its specifications available in this way, X/Open makes it possible for conformant products to be developed as soon as is practicable, so enhancing the value of the X/Open brand as a procurement aid to users.

- *Preliminary Specifications*

  These specifications, which often address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations, are released in a controlled manner for the purpose of validation through implementation of products. A Preliminary specification is not a draft specification. In fact, it is as stable as X/Open can make it, and on publication has gone through the same rigorous X/Open development and review procedures as a CAE specification.

  Preliminary specifications are analogous to the *trial-use* standards issued by formal standards organisations, and product development teams are encouraged to develop products on the basis of them. However, because of the nature of the technology that a Preliminary specification is addressing, it may be untried in multiple independent implementations, and may therefore change before being published as a CAE specification. There is always the intent to progress to a corresponding CAE specification, but the ability to do so depends on consensus among X/Open members. In all cases, any resulting CAE specification is made as upwards-compatible as possible. However, complete upwards-compatibility from the Preliminary to the CAE specification cannot be guaranteed.

In addition, X/Open publishes:

- *Guides*

  These provide information that X/Open believes is useful in the evaluation, procurement, development or management of open systems, particularly those that are X/Open-compliant. X/Open Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming X/Open conformance.

- *Technical Studies*

  X/Open Technical Studies present results of analyses performed by X/Open on subjects of interest in areas relevant to X/Open's Technical Programme. They are intended to communicate the findings to the outside world and, where appropriate, stimulate discussion and actions by other bodies and the industry in general.

- *Snapshots*

  These provide a mechanism for X/Open to disseminate information on its current direction and thinking, in advance of possible development of a Specification, Guide or Technical Study. The intention is to stimulate industry debate and prototyping, and solicit feedback. A Snapshot represents the interim results of an X/Open technical activity. Although at the time of its publication, there may be an intention to progress the activity towards publication of a Specification, Guide or Technical Study, X/Open is a consensus organisation, and makes no commitment regarding future development and further publication. Similarly, a Snapshot does not represent any commitment by X/Open members to develop any specific products.

Versions and Issues of Specifications

As with all *live* documents, CAE Specifications require revision, in this case as the subject technology develops and to align with emerging associated international standards. X/Open makes a distinction between revised specifications which are fully backward compatible and those which are not:

- a new *Version* indicates that this publication includes all the same (unchanged) definitive information from the previous publication of that title, but also includes extensions or additional information. As such, it *replaces* the previous publication.

- a new *Issue* does include changes to the definitive information contained in the previous publication of that title (and may also include extensions or additional information). As such, X/Open maintains *both* the previous and new issue as current publications.

Corrigenda

Most X/Open publications deal with technology at the leading edge of open systems development. Feedback from implementation experience gained from using these publications occasionally uncovers errors or inconsistencies. Significant errors or recommended solutions to reported problems are communicated by means of Corrigenda.

The reader of this document is advised to check periodically if any Corrigenda apply to this publication. This may be done in any one of the following ways:

- anonymous ftp to ftp.xopen.org

- ftpmail (see below)

- reference to the Corrigenda list in the latest X/Open Publications Price List.

To request Corrigenda information using ftpmail, send a message to ftpmail@xopen.org with the following four lines in the body of the message:

```
open
cd pub/Corrigenda
get index
quit
```

This will return the index of publications for which Corrigenda exist. Use the same email address to request a copy of the full corrigendum information following the email instructions.

This Document

This document is a Guide (see above). It provides a functional description of the X/Open Distributed Transaction Processing (DTP) model, a software architecture that allows multiple application programs to share resources provided by multiple resource managers, and allows their work to be coordinated into global transactions.

This guide describes the use of the X/Open DTP Model within the X/Open Common Applications Environment (CAE), and is a prerequisite to other present and emerging X/Open documents that address DTP. This guide gives an introduction to the interfaces those other documents specify; it defines terms that those documents use; and it gives a concise overview of the interrelation of those DTP interfaces and of the software components that use them.

X/Open DTP publications based on this model specify a portable high-level TP language (HTL), portable application programming interfaces (APIs) and system-level interfaces that facilitate:

- portability of application program source code to any X/Open environment that offers that HTL and/or those APIs

- interchangeability of transaction managers and resource managers from various sources

- interoperability of diverse transaction managers and resource managers in the same global transactions.

This guide is structured as follows:

- Chapter 1 is an introduction.

- Chapter 2 provides fundamental definitions for the remainder of the guide.

- Chapter 3 describes the X/Open DTP Model.

- Appendix A addresses some frequently asked questions about how existing product structures fit within the X/Open DTP Model.

Intended Audience

This guide is intended to introduce the X/Open DTP Model to application developers, system administrators and product builders and suppliers. It assumes prior knowledge of distributed transaction processing.

Typographical Conventions

The following typographical conventions are used throughout this document:

- **Bold** font is used in text for keywords and references to published standards and specifications.

- *Italic* strings are used for emphasis and to identify the first instance of a word requiring definition. Italics in text also denote C-language functions; these are shown as follows: *name*( ).

Superseded Documents

This Version 3 guide supersedes the X/Open **Distributed Transaction Processing: Reference Model, Version 2** Published in 1993. Since that version, the X/Open **CPI-C, Version 2** interface has superseded the X/Open **Peer-to-Peer** interface as one of the three interfaces between an Application Program (AP) and a Communication Resource Manager (CRM). Also, the X/Open High-level Transaction Language (HTL) has been introduced as an additional means of producing an AP.

As future developments may change the number or titles of relevant publications, readers should consult a current X/Open publications catalogue (http://www.xopen.org) before ordering copies of individual specifications.

Trademarks

X/Open$^{®}$ is a registered trade mark, and the ''X'' device is a trade mark, of X/Open Company Limited.

Referenced Documents

The following X/Open documents are referenced in this guide:

CPI-C, Version 2
> X/Open CAE Specification, November 1995, Distributed Transaction Processing: The CPI-C Specification, Version 2 (ISBN: 1-85912-135-7, C419).

DTP, Version 2
> X/Open Guide, November 1993, Distributed Transaction Processing: Reference Model, Version 2 (ISBN: 1-85912-019-9, G307).

ISAM
> X/Open Developers' Specification, August 1990, Indexed Sequential Access Method (ISAM) (ISBN: 1-872630-03-0, D010).

SQL
> X/Open CAE Specification, August 1992, Structured Query Language (SQL) (ISBN: 1-872630-58-8, C201).

STDL
> X/Open Preliminary Specification, December 1995, Structured Transaction Definition Language (STDL) (ISBN: 1-85912-120-9, P536).

TX
> X/Open CAE Specification, April 1995, Distributed Transaction Processing: The TX (Transaction Demarcation) Specification (ISBN: 1-85912-094-6, C504).

TxRPC
> X/Open CAE Specification, November 1995, Distributed Transaction Processing: The TxRPC Specification (ISBN: 1-85912-115-2, C505).

XA
> X/Open CAE Specification, December 1991, Distributed Transaction Processing: The XA Specification (ISBN: 1-872630-24-3, C193 or XO/CAE/91/300).

XA+
> X/Open Snapshot, July 1994, Distributed Transaction Processing: The XA+ Specification, Version 2 (ISBN: 1-85912-046-6, S423).

XAP-TP
> X/Open CAE Specification, April 1995, ACSE/Presentation: Transaction Processing API (XAP-TP) (ISBN: 1-85912-091-1, C409).

XATMI
> X/Open CAE Specification, November 1995, Distributed Transaction Processing: The XATMI Specification (ISBN: 1-85912-130-6, C506).

XDCS
> X/Open Guide, November 1992, Distributed Computing Services (XDCS) Framework (ISBN: 1-872630-64-2, G212).

The following non-X/Open documents are referenced in this guide:

APPC
> Advanced Peer-to-Peer Communications: Resource Reference, Order Number G325-0055), IBM Corporation.

CIW CPI-C
> Common Program Interface Communications Specification, Second Edition (June 1994), Order Number SC31-6180-01, IBM Corporation.

OSI CCR
> ISO/IEC 9804
> > ISO/IEC 9804:1994, Information Technology — Open Systems Interconnection — Service Definition for the Commitment, Concurrency, and Recovery Service Element.
>
> ISO/IEC 9805
> > ISO/IEC 9805:1994, Information Technology — Open Systems Interconnection — Protocol Specification for the Commitment, Concurrency, and Recovery Service Element.

OSI TP
> ISO/IEC 10026, Information Technology — Open Systems Interconnection — Distributed Transaction Processing, Parts 1 to 3:
>
> > Part 1 (1992): OSI TP Model
> > Part 2 (1992): OSI TP Service
> > Part 3 (1992): Protocol Specification.

Introduction

1.1   Overview

The X/Open Distributed Transaction Processing (DTP) model is a software architecture that allows multiple application programs to share resources provided by multiple resource managers, and allows their work to be coordinated into global transactions.

The X/Open DTP Model comprises five basic functional components:

- an Application Program (AP), which defines transaction boundaries and specifies actions that constitute a transaction

- Resource Managers (RMs) such as databases or file access systems, which provide access to resources

- a Transaction Manager (TM), which assigns identifiers to transactions, monitors their progress, and takes responsibility for transaction completion and for coordinating failure recovery.

- Communication Resource Managers (CRMs), which control communication between distributed applications within or across TM domains.

- a communication protocol, which provides the underlying communication services used by distributed applications and supported by CRMs.

These terms are defined more precisely in Section 3.2 on page 8.

X/Open DTP publications based on this model specify a portable high-level TP language (HTL), portable APIs and system-level interfaces that facilitate:

- portability of application program source code to any X/Open environment that offers that HTL and/or those APIs

- interchangeability of TMs, RMs and CRMs from various sources

- interoperability of diverse TMs, RMs and CRMs in the same global transaction.

1.2   Benefits of X/Open DTP

Distributed transaction processing provides the necessary mechanism to combine multiple software components into a cooperating unit that can maintain shared data, potentially spanning multiple physical processors or locations, or both. This enables construction of applications that manipulate data consistently using multiple products, that can easily incorporate additional components, and that can be scaled by adding additional hardware and software components.

Portability, interchangeability and interoperability let application writers benefit from a wider selection of transaction managers, resource managers and communication resource managers. Application writers also gain the freedom to select from alternative products, hardware and software platforms.

Published language and interface specifications simplify software design. For example, some software products that might once have been implemented using customised interfaces may

now be viewed as interchangeable DTP components. This approach shortens development time and extends the above benefits to a greater number of products.

All parties benefit from the X/Open method of achieving consensus and communication among open systems providers, with a significant voice for system vendors, independent software vendors (ISVs) and users.

1.3   Areas Not Addressed

The X/Open DTP Model does not address all issues of importance in transaction processing, for example:

- configuration, administration and monitoring of transaction processing systems

- forms management and other user interfaces

- specification of benchmarks

- security

- naming

- communication techniques that may be used within RM implementations.

1.4   Relationship to International Standards

The X/Open DTP Model shows that DTP software components use the communication protocol specified in the referenced OSI TP standards to facilitate interoperability of components in heterogeneous environments. The use of OSI TP is not mandatory where implementors require to use a product-internal communication provider.

Data structures from the referenced OSI CCR standards are a recommended component of the transaction identifier that X/Open specifies.

Definitions

This chapter gives fundamental definitions on which subsequent chapters depend. It is structured as follows:

- Transaction Definitions

- Model Definitions.

The terms Application Program (AP), Resource Manager (RM), Transaction Manager (TM) and Communication Resource Manager (CRM) are defined in Section 3.2 on page 8.

2.1   Transaction Definitions

Transaction

A *transaction* is a complete unit of work which, in general terms, can apply to many contexts. It may comprise many computational tasks including user interfaces, data retrieval and communications. A typical transaction modifies resources. The model described in the referenced OSI TP standards defines the term *transaction* more precisely.

This guide refers to specific types of transaction, such as *global* and *distributed*. These are defined below.

Transaction Properties

Transactions typically exhibit the following properties:

Atomicity	The results of the transaction's execution are either all committed or all rolled back.
Consistency	A completed transaction transforms a shared resource from one valid state to another valid state.
Isolation	Changes to shared resources that a transaction effects do not become visible outside the transaction until the transaction commits.
Durability	The changes that result from transaction commitment survive subsequent system or media failures.

These properties are known by their initials as the **ACID** properties. In the X/Open DTP Model, the TM coordinates Atomicity at global level whilst each RM is responsible for the Atomicity, Consistency, Isolation and Durability of its resources.

Global Transaction

The term *global transaction* collectively describes all the work done by participating RMs anywhere in the network for a single unit of work. An AP defines the start and end of a global transaction. A single, coordinating, TM manages its initiation and completion.

Distributed Transaction

This guide uses the term *global*, rather than *distributed*, transaction. Chapter 3 concentrates on global transaction processing whilst avoiding confusion about the actual location of the work and its distribution across physical locations, network nodes or TM Domains.

Commitment

Commitment is the act that ends a transaction and makes permanent all changes to resources specified during that transaction.

Rollback

Rollback is the act that ends a transaction and nullifies or undoes all changes to resources specified during that transaction.

Transaction Completion

Transaction completion means either *commitment* or *rollback*.

Commitment Protocol

A *commitment protocol* is the synchronisation that occurs at transaction completion. The X/Open DTP Model follows the *two-phase commit with presumed rollback*[1] protocol defined in the referenced OSI TP standards. A description of the basic protocol is given in Section 3.4.3 on page 13. In certain cases, a global transaction may be completed *heuristically*. Heuristic transaction completion is described in Section 3.4.5 on page 14.

RM Resource

In the X/Open DTP Model, an *RM resource* is the collection of data that an RM manages. A resource may be *shared* with other global transactions or *dedicated* to a single transaction.

RM Native Interface

The RM's native interface is the RM-defined API by which an AP operates on the RM's resource. The RM may support a standard interface, such as SQL or ISAM, in which case the AP may be portable to other RMs that use the same interface. The RM may, on the other hand, offer a proprietary interface specific to its services.

_____

1. To aid readability, some parts of this guide use the term *two-phase commit* as an abbreviation of *two-phase commit with presumed rollback*.

2.2   Model Definitions

Instance of the Model

An *instance of the model* (or *instance*)[2] is the set of computing entities that implement the functional components and interfaces of all or part of an application within the X/Open DTP Model. Each *instance* may support one AP, one TM and multiple RMs. A distributed application is represented by two or more *instances* and includes a CRM in each *instance.* An *instance* is part of a single TM Domain. A TM Domain can contain one or more instances.

The possible implementation of product structures that make up an instance may vary from simple to very complex. Appendix A addresses some frequently asked questions about how existing product structures fit within the X/Open DTP Model.

TM Domain

A *TM Domain* consists of one or more instances that use the same TM. This TM is common to all applications that run within that TM Domain. The common TM uses logically-shared data structures and logs for global-transaction management, such as when issuing global transaction identifiers or when coordinating global-transaction recovery.

The figure below illustrates four *instances* of applications within a single TM Domain. The TM is the same in each *instance* and is labelled **TM1**. Different RMs may participate in each instance.
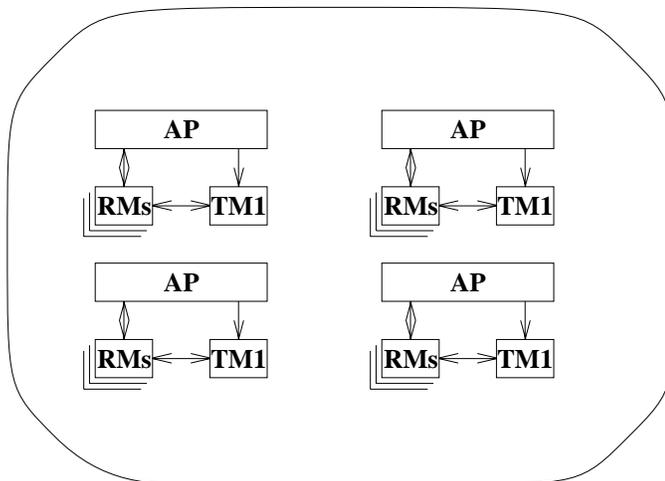
Figure 2-1 A TM Domain with Four Instances

_____

2. To aid readability, some parts of this guide use the term *instance* as an abbreviation of *instance of the model*

X/Open DTP Model

At a basic level, the model is a software architecture that allows a single AP to share resources provided by multiple RMs within a single TM Domain, and allows RM-internal work to be coordinated into a global transaction using a single TM. It uses the TX and XA interfaces.

The full model is a software architecture that allows multiple APs to share resources provided by multiple RMs located across one or more TM Domains, and allows RM-internal work to be coordinated into a global transaction using multiple TMs. The full model uses the STDL language, the TX, XA and XA+ interfaces, the interfaces provided by the Communication Resource Manager (CRM) specifications, namely TxRPC, XATMI and CPI-C, Version 2, and the XAP-TP interface to coordinate communication through OSI TP.

Threads

A thread is the entity, with all its context, that is currently in control of a processor. For portability reasons, the notion of thread must be common among the AP, TM and RM.

The thread concept is central to the TM's coordination of RMs. APs call RMs to request work, while TMs call RMs to delineate transaction branches. The way the RM knows that a given work request pertains to a given branch is that the AP and the TM both call it from *the same thread*. For example, an AP thread calls the TM to declare the start of a global transaction. The TM records this fact and informs RMs. After the AP regains control, it uses the native interface of one or more RMs to do work. The RM receives the calls from the AP and TM in the same thread.

The Model

This chapter gives an abstract description of the X/Open DTP Model of an application program environment for global transaction processing.

3.1 Functional Model

The boxes in the figure below are the functional components and the connecting lines are the interfaces between them. The arrows indicate the directions in which control may flow.

SUPERIOR NODE
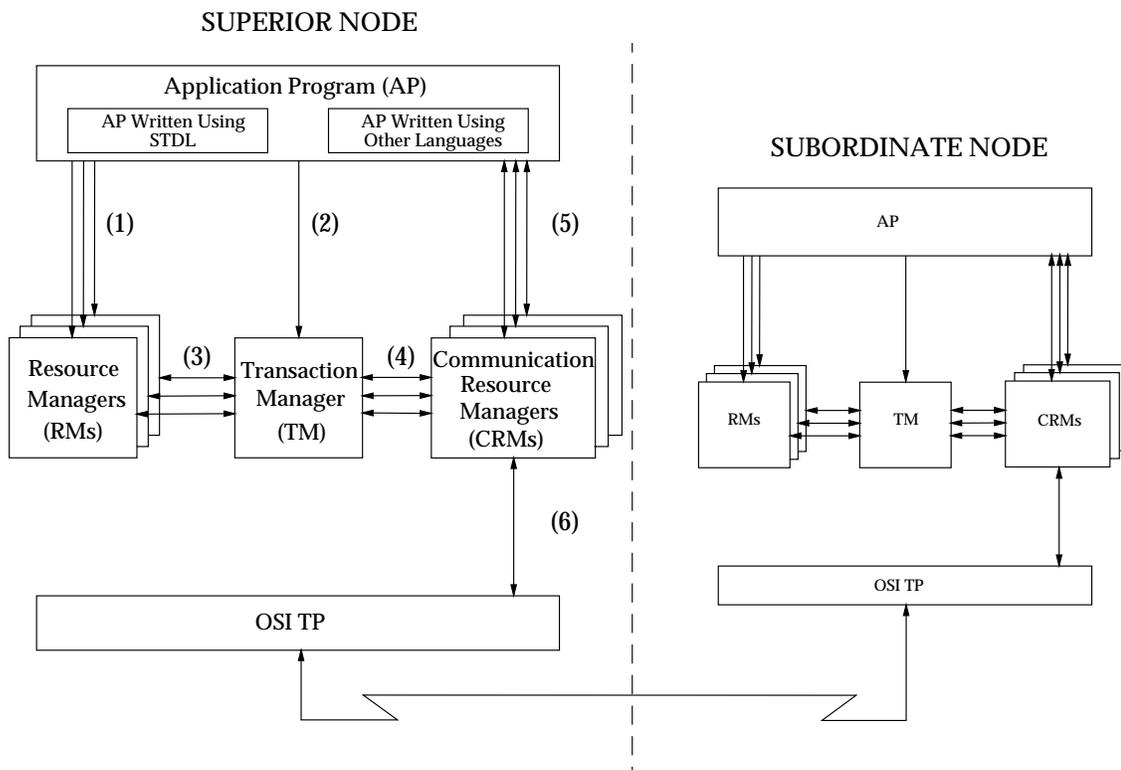
Figure 3-**1** Functional Components and Interfaces

Descriptions of the functional components shown can be found in Section 3.2 on page 8. The numbers in brackets in the above figure represent the different X/Open interfaces that are used in the model. They are described in Section 3.3.1 on page 10. The subsequent sections provide a description of how the model works.

3.2   Functional Components

3.2.1   Application Program (AP)

The application program (AP) implements the desired function of the end-user enterprise. Each AP specifies a sequence of operations that involves resources such as databases. An AP defines the start and end of global transactions, accesses resources within transaction boundaries, and normally makes the decision whether to commit or roll back each transaction.

Where two or more APs cooperate within a global transaction, the X/Open DTP Model supports three *paradigms* for AP-to-AP communication. These are the TxRPC, XATMI and CPI-C, Version 2 interfaces, described in Section 3.7 on page 21.

Where two or more APs support a global transaction, one AP is designated the superior AP and the other AP(s) the subordinate(s). The superior AP makes distributed requests for services to be provided by subordinates; for example, service requests using the XATMI interface. A superior AP can have many subordinate APs whereas each subordinate AP has only a single superior. A subordinate AP can also be superior to other APs working at a lower level. Hence a hierarchy of APs can be built up.

An AP acting as the superior typically exerts more control than any of its subordinates. For example, depending on the communication paradigm, only the superior AP may be allowed to start and commit a global transaction. Its subordinates however may roll back the global transaction.

APs can be written using the X/Open HTL, the X/Open APIs or a combination of the two. See Section 3.8 on page 23 for further information on the X/Open HTL.

3.2.2   Transaction Manager (TM)

The transaction manager (TM) manages global transactions and coordinates the decision to start them, and commit them or roll them back. This ensures atomic transaction completion. The TM also coordinates recovery activities of the resource managers when necessary, such as after a component fails.

In addition, the following information applies where two or more TMs cooperate within a global transaction.

The TM that works on behalf of the superior AP is designated the superior TM. Other TMs are designated subordinate TMs. The superior/subordinate relationship is the same for TMs as for their respective APs. Within a global transaction, each TM only controls those RMs within its own instance of the model. Through their CRMs (see Section 3.2.4 on page 9), TMs communicate global transaction information between each other by use of the XA+ interface.

3.2.3   Resource Manager (RM)

The resource manager (RM) manages a defined part of the computer's shared resources. These may be accessed using services that the RM provides. Examples for RMs are database management systems (DBMSs), a file access method such as X/Open ISAM, and a print server.

In the X/Open DTP Model, RMs structure all changes to the resources they manage as recoverable and atomic transactions. They let the TM coordinate completion of these transactions atomically with work done by other RMs.

In addition, the following information applies when RMs take part in a global transaction involving two or more TM Domains or instance of the model.

Within the context of the X/Open DTP Model, an RM receives global transaction information only from the TM in the instances of the model with which it is registered. This information is provided through the XA interface. An RM uses the *ax_*()* functions of the XA interface to communicate with its TM.

Outside the X/Open DTP Model context, an RM is free to access data at many physical locations. For example, a Relational Database Management System (RDBMS) may use a *gateway* technology to access a remote database. The RDBMS presents the AP with a single-image logical view of the database concerned, and the AP is not aware of either its physical location or the detailed structure of its components. In such a setup, the RM must be able to complete its own RM-internal work atomically, on behalf of the global transaction, no matter how physically diverse are its resources.

3.2.4   Communication Resource Manager (CRM)

A CRM allows an instance of the model to access another instance either inside or outside the current TM Domain. Within the X/Open DTP Model, CRMs use OSI TP services to provide a communication layer across TM Domains. CRMs aid global transactions by supporting the following interfaces:

- the communication paradigm (TxRPC, XATMI or CPI-C, Version 2) used between an AP and CRM

- XA+ communication between a TM and CRM

- XAP-TP communication between a CRM and OSI TP.

A CRM may support more than one type of communication paradigm, or a TM Domain may use different CRMs to support different paradigms. The XA+ interface provides global transaction information across different instances and TM Domains. The CRM allows a global transaction to extend to another TM Domain, and allows TMs to coordinate global transaction commit and abort requests from (usually) the superior AP. Using the above interfaces, information flows from superior to subordinate and *vice versa*.

The CRM that supports the superior TM is called the superior CRM. Other CRMs, that support other TMs, within the global transaction, are called subordinate CRMs. The superior/subordinate relationship is the same for CRMs as for their respective APs and TMs. This means that a CRM can only directly support the AP and TM of its own instance. It follows that a CRM in one instance cannot directly support an AP and TM in another instance, but must request the services of the CRM for that instance. This paired relationship between CRMs is shown in Figure 3-1 on page 7.

3.3   Interfaces between Functional Components

3.3.1   Functional Component Interfaces

There are six interfaces between software components in the X/Open DTP Model. The numbers correspond to the numbers in Figure 3-1 on page 7.

(1) **AP-RM.** The AP-RM interfaces give the AP access to resources. X/Open interfaces, such as SQL and ISAM, provide AP portability. The X/Open DTP Model imposes few constraints on native RM APIs. The constraints involve only those native RM interfaces that define transactions. (See the referenced **XA** Specification.)

(2) **AP-TM.** The AP-TM interface allows the AP to coordinate global transaction management with the TM. For example, when the AP calls *tx_begin*( ), the TM informs the participating RMs of the start of a global transaction. After each request is completed, the TM provides a return value to the AP reporting back the success or otherwise of the TX call.

For details of the AP-TM interface, see the referenced **STDL** Specification and the **TX** (Transaction Demarcation) Specification.[3]

In addition, the following information applies when two or more APs cooperate within a global transaction.

If the AP is designated as the superior, it uses the AP-TM interface to delimit global transactions. If the AP is designated as a subordinate, then depending on the communication paradigm employed, it may not be permitted to use all the facilities of the AP-TM interface. For example, the subordinate AP may not be allowed to request global transaction commitment.

A TM may receive global transaction information from another TM routed through remote and local CRMs. For example, a superior TM may inform subordinate TMs to commit a global transaction without the subordinate APs' knowledge. Subordinate APs can, however, access information about the current global transaction context by calling *tx_info*( ).

(3) **TM-RM.** The TM-RM interface (the XA interface) lets the TM structure the work of RMs into global transactions and coordinate completion or recovery. The XA interface is the bidirectional interface between the TM and RM.

The functions that each RM provides for the TM are called the *xa_**( ) functions.[4] For example, the TM calls *xa_start*( ) in each participating RM to start an RM-internal transaction as part of a new global transaction. Later, the TM may call in sequence *xa_end*( ), *xa_prepare*( ) and *xa_commit*( ) to coordinate a (successful in this case) two-phase commit protocol. The functions that the TM provides for each RM are called the *ax_**( ) functions. For example, an RM calls *ax_reg*( ) to register dynamically with the TM.

For details of the TM-RM interface, see the referenced **XA** Specification.

The TX and XA interfaces cooperate to provide global transaction management between the AP and TM and, at a lower level, the TM and participating RMs. The TX interface drives the

_____

3. The TX (transaction demarcation) interface provides both a C and a COBOL programming interface. TX functions are in lower-case for C — for example, *tx_commit*( )— and in upper-case for COBOL — for example, TXCOMMIT. To aid readability, this guide uses the C versions only.

4. The XA interface is a C interface only, between the TM and RM. It is not accessed directly by the AP.

XA interface when managing global transactions. The XA interface reports back to the TM the success (or otherwise) of global transaction work at RM level. The TM then collates the RM returned information and reports back to the AP the success (or otherwise) of global transaction management at TM level.

Since the XA interface is invisible to the AP, the TM and RM may use other methods to interconnect without affecting application portability.

(4) **TM-CRM.** The TM-CRM interface (the XA+ interface) supports global transaction information flow across TM Domains. In particular TMs can instruct CRMs by use of *xa_*()* function calls to suspend or complete transaction branches, and to propagate global transaction commitment protocols to other transaction branches. CRMs pass information to TMs in subordinate branches by use of *ax_*()* function calls. CRMs also use *ax_*()* function calls to request the TM to create subordinate transaction branches, to save and retrieve recovery information, and to inform the TM of the start and end of blocking conditions.

For details of the TM-CRM interface, see the referenced **XA+** Specification.

The XA+ interface is a superset of the XA interface and supersedes its purpose. Since the XA+ interface is invisible to the AP, the TM and CRM may use other methods to interconnect without affecting application portability.

(5) **AP-CRM.** X/Open provides portable APIs for DTP communication between APs within a global transaction. The API chosen can significantly influence (and may indeed be fundamental to) the whole architecture of the application. For this reason, these APIs are frequently referred to in this guide and elsewhere as *communication paradigms*. In practice, each paradigm has unique strengths, so X/Open offers the following popular paradigms:

- the TxRPC interface (see the **TxRPC** Specification)

- the XATMI interface (see the **XATMI** Specification)

- the CPI-C, Version 2 interface (see the **CPI-C, Version 2** Specification).

These paradigms are described in more detail in Section 3.7 on page 21.

X/Open interfaces, such as the three CRM APIs listed above, provide application portability across products offering the same CRM API. The X/Open DTP Model imposes few constraints on native CRM APIs.

(6) **CRM-OSI TP.** This interface (the XAP-TP interface) provides a programming interface between a CRM and Open Systems Interconnection Distributed Transaction Processing (OSI TP) services. XAP-TP interfaces with the OSI TP Service and the Presentation Layer of the seven-layer OSI model. X/Open has defined this interface to support portable implementations of application-specific OSI services. The use of OSI TP is mandatory for communication between heterogeneous TM domains. For details of this interface, see the referenced **XAP-TP** Specification and the OSI TP standards.

3.3.2   Data Interfaces

Transaction Identifier

A transaction identifier (XID) is a data structure that a TM assigns. It represents the unique relationship between an AP, the work it issues to RMs, and the global transaction which the TM manages on behalf of the AP.

The XID lets the TM track and coordinate all of the work associated with a global transaction. Each RM maps the XID to the RM-internal work it does for the global transaction. To ensure global uniqueness, the XID should contain *atomic action identifiers* as specified in the referenced OSI CCR standards. For more information on XIDs, see also the **XA** Specification.

XA Switch Structure

Each RM provides a set of pointers to the functions that the TM calls, in a data structure known as the XA Switch structure.

3.4   Activity between Functional Components Involving a Single AP

3.4.1   Transaction Initiation

When an AP instructs its TM to start a global transaction, the TM informs all participating RMs in order to associate with the global transaction any work the AP may request of them.

Some RMs are configured so that the TM does not inform them when a global transaction starts. Instead, the RM contacts the TM to become associated with a global transaction only after the AP calls it to request actual work. This is known as *dynamic registration*.

3.4.2   Transaction Association

A thread working on a transaction branch may suspend association with that transaction branch when blocking, awaiting a message from another application. The TM may resume the association in the same thread or optionally in a different one subject to RM control. The TM suspends an association by issuing *xa_end*( ) to the local RMs with the TMSUSPEND flag set. The TM also sets the TMMIGRATE flag if it might resume the association in a different thread. If all RMs permit migration, the TM may resume the association in the same or a different thread. Otherwise, the association must be resumed in the same thread. Each RM must retain transaction context (for example, locks and cursors) while the association is suspended.

3.4.3   Transaction Commitment

When an AP instructs its TM to commit a transaction, the TM and participating RMs use the two-phase commit protocol to ensure transaction atomicity. The AP requests that the TM initiate commitment.

In Phase 1, the TM issues *xa_prepare*( ) which requests each participating RM to *prepare to commit* its work and report whether it can guarantee its ability to commit the work it did on behalf of the global transaction. If an RM can commit its work, it replies affirmatively. A negative reply reports failure for any reason.

In Phase 2, the TM directs all RMs either to commit or roll back the work done on behalf of the global transaction. Whether the TM issues *xa_commit*( ) or *xa_rollback*( ) depends on the RMs' responses during Phase 1. All RMs commit or roll back changes to shared resources and then return status to the TM.

When an AP requests commitment of a global transaction, the TM reports back to the AP whether commitment or rollback was the outcome. This is based on reports the TM received from all participating RMs.

The **XA** Specification contains two optimisations in the calling sequence between TM and RM:

- An RM can withdraw from further participation in a global transaction during Phase 1 if it was not asked to update resources. This is known as *read-only optimisation*.

- A TM can use *one-phase commit* if it is dealing with only one RM that is making changes to resources.

The **XA** Specification discusses requirements for the stable recording of transaction data, and specifies when the TM and participating RMs are free to discard their knowledge of the global transaction. See the referenced **XA** Specification for further details.

3.4.4   Transaction Rollback

Transaction rollback can occur in three ways:

- **Explicit Rollback**

  The AP requests explicit rollback of the global transaction.

- **Implicit Rollback**

  The TM rolls back the global transaction if any RM responds negatively to the Phase 1 request.

- **Presumed Rollback**

  On restart, an RM rolls back any transaction that was active at the time of failure, provided that it has not already responded positively to a Phase 1 *prepare to commit* request from the TM.

In each case, the participating RMs must not allow any changes to resources to become permanent.

3.4.5   Heuristic Transaction Completion

In certain unusual cases, an RM may unilaterally commit or roll back changes to recoverable resources that it made within a global transaction. This may happen, for example, when the RM experiences too long a delay between Phases 1 and 2 of the two-phase commit protocol, or it is prompted by operator intervention to complete (commit or rollback) its RM-internal transaction. In such cases, the RM is said to make a *heuristic* decision. Such a decision is made by the RM without knowledge of the state of the global transaction of which it is part. The RM may then unlock *shared* resources and allow other global transactions to make further changes.

If a heuristic decision is taken, it is possible that at a later point the global transaction may complete with a decision contrary to that made by the RM. In such a circumstance, the global transaction fails to maintain global **ACID** properties. In other cases a TM may not be able to determine whether an RM's heuristic decision matched the TM's global transaction completion decision. If a contrary heuristic decision is reported by an RM, the TM reports to the AP that a heuristic decision has occurred. This decision is either contrary to the AP's direction or both complementary and contrary *(mixed)* depending on the combined results from all participating RMs. If the TM cannot determine whether an RM's heuristic decision matches the decision made by the TM, the TM reports to the AP that there is a possibility (*hazard*) of a contrary heuristic decision.

In the X/Open DTP Model, an RM that reports heuristic completion to the TM must not discard its own knowledge of the decision until authorised by the TM. Until this happens, a TM can request *xa_recover*( ) of an RM to collect a list of transaction identifiers for heuristically completed and prepared transactions.

The referenced OSI CCR standards define heuristics more precisely.

3.4.6   Recovery after Failure

Recovery is the process of restoring resources to a consistent state after various types of failure. Recovery processing in an X/Open DTP system is compatible with that described in the OSI TP standards, which define the presumed-rollback protocol. The X/Open DTP Model makes the following assumptions:

- that the TM and RMs have access to stable storage

- that the TM initiates and controls transaction recovery

- that RMs provide for their own restart and recovery as directed by the TM.

3.5   Distributed Communication Facilities

3.5.1   Communication within TM Domains

Distributed applications that support global transactions across two or more instances of the model may communicate with each other within a single TM Domain. Communication between two or more instances is managed at each end by a CRM. The RMs coordinate with the domain's TM used in the appropriate instance.

3.5.2   Communication across TM Domains

The X/Open DTP Model specifies that heterogeneous TM Domains support CRMs that use the OSI TP protocol as a common communication layer. Proprietary protocols or the OSI TP protocol may be used between homogeneous TM Domains.

3.5.3   Sharing Resources across TM Domains

TM Domains may share resources in ways other than by using CRMs. For example, RDBMSs in different TM Domains may share a common back-end database. In this case, an RM may receive XIDs from several TMs that have no knowledge of each other. Use of a common format of XID, and adherence to the rules for uniqueness of the global transaction identifier (see Section 3.3.2 on page 12) ensure that the RM can distinguish between different global transactions and their associated transaction branches.

3.5.4   Global Transaction Demarcation

The AP-TM interface is used in the X/Open DTP Model application program environment to define global transaction start, end, commitment or rollback and other facilities. Before an AP starts a global transaction, it is in non-global transaction status, and so any work requested of RMs by the AP during this period is not part of a global transaction started later.

The TX interface supports the concept of *unchained* and *chained* global transactions. In unchained mode (which is the default), when a global transaction completes, a new transaction does not automatically start until the AP calls *tx_begin*( ) again.

When in chained mode, after the initial global transaction starts and completes, the next global transaction is started automatically.

3.5.5   Global Transaction Tree Structure

Within the X/Open DTP Model, global transactions that operate across distributed APs are managed by use of a tree structure. This is shown in Figure 3-2 on page 17.
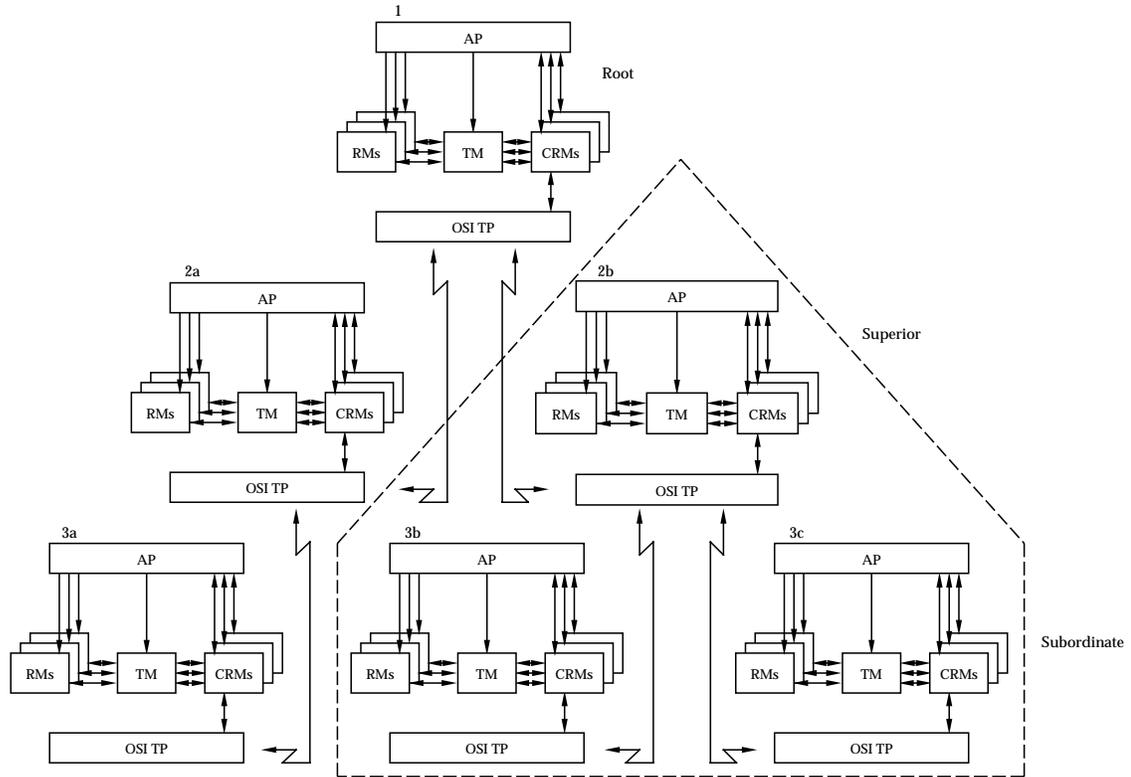
Figure 3-2 Global Transaction Tree Structure

When distributed communication occurs between two instances, they have a relationship of *Superior* and *Subordinate*. The instance that requests the participation of another instance in a global transaction is called a superior. The requested instance is called a subordinate. Especially, an instance that does not have a superior is called a root.

An example of this is shown in Figure 3-2, where 2b is a superior to both 3b and 3c; 2b, however, is a subordinate to 1.

3.5.6   Global Transactions and the Transaction Tree

A global transaction has one or more transaction branches. Each branch represents part of the work in support of a global transaction for which a TM and set of participating RMs engage.

During two-phase-commit processing, the superior TM manages commitment coordination with subordinate TMs and their participating RMs at branch level. This is achieved by the superior CRM reporting back to the superior TM the success or failure of one or more branches in response to prepare and commit requests.

3.5.7   Tightly- and Loosely-coupled Threads

Many application threads can participate in a single global transaction. All the work done in these threads is atomically completed. Within a single global transaction, the relationship between any pair of participating threads is either *tightly-coupled* or *loosely-coupled*:

- A tightly-coupled relationship is one where a pair of threads are designed to share resources. In addition, with respect to an RM's isolation policies, the pair are treated as a single entity. Thus, for a pair of tightly-coupled threads, the RM must guarantee that resource deadlock does not occur within the transaction branch.

- A loosely-coupled relationship provides no such guarantee. With respect to an RM's isolation policies, the pair may be treated as if they were in separate global transactions even though the work is atomically completed.

Within a single global transaction, a set of tightly-coupled threads may consist of more than just a pair. Moreover, many sets of tightly-coupled threads may exist within the same global transaction and each set is loosely coupled with respect to the others.

3.5.8   Commitment Coordination

The TM that manages global transaction completion is called the *commitment coordinator*. It is the AP of the commitment coordinator instance of the model that requests commitment.

3.6   Activity between Functional Components Involving Two or More APs

In addition to the description of transaction management given in Section 3.4 on page 13, the following information applies when two or more distributed APs cooperate within a global transaction.

3.6.1   Transaction Initiation

When an AP makes a request to a remote AP for the first time, a new transaction branch for a subordinate must be created. There are two methods of creating and managing branches. One is called *TM-managed transaction branches*, the other is called *CRM-managed transaction branches*.

With TM-managed transaction branches, the CRM local to its instance of the model requests the TM to build and log details of a new transaction branch prior to accessing the remote AP for the first time.

With CRM-managed transaction branches, the CRM builds the transaction branch and manages it without involvement from the TM. Such a CRM appears to the TM as an ordinary RM, but in effect is acting as the distributed manager of the TM in the global transaction.

In either case, subsequent accesses to the same remote AP are made through the same transaction branch. A CRM accesses different remote APs through different transaction branches.

The CRM uses transaction branches to target global transaction coordination information passed on to the (subordinate) branch via XA+ commands issued by the TM.

A CRM can be configured so that the TM does not inform it when a global transaction starts. Such a CRM contacts the TM to become associated with a global transaction only after the AP calls it to make a distributed request. This is known as *dynamic registration*.

3.6.2   Transaction Association

The CRM suspends the association of a thread with a transaction branch by issuing *ax_end*() to the TM with the TMSUSPEND flag set. The CRM also sets the TMMIGRATE flag if it might resume the association in a different thread. If the TM and all local RMs permit migration, the CRM may resume the association in the same or a different thread. Otherwise the association must be resumed in the same thread. (See Section 3.4.2 on page 13).

3.6.3   Transaction Commitment

When an AP in the root or superior instructs its TM to commit a global transaction, the TM follows the two-phase commit protocol. In addition to Phase 1 coordination of commitment of participating RMs, the superior TM uses its CRM to instruct all other transaction branches within the global transaction to prepare to commit. The CRM passes an *xa_prepare*() request to subordinate TMs in other branches. The CRM then waits for and returns to its own TM each transaction branch response.

If the TM receives a positive response from its participating RMs and from all associated transaction branches, the TM initiates Phase 2 by requesting *xa_commit*() of the global transaction. The TM again uses its CRM to instruct all other transaction branches and await their response.

3.6.4   Transaction Rollback

If the TM receives a negative response from a local RM or from any of the associated transaction branches, the TM initiates Phase 2 by requesting *xa_rollback*() of the global transaction. The TM again uses its CRM to instruct transaction branches and await their response.

3.6.5   Heuristic Transaction Completion

When a global transaction extends across two or more transaction branches through communication, RMs in any of the branches of subordinates might experience too long a delay between Phases 1 and 2 of the two-phase commit protocol. For example, a communication failure might have delayed the commit/rollback indication from the superior TM. As Section 3.4.5 on page 14 describes, a subordinate RM may choose to complete its work without the *xa_commit*() instruction from the TM.

During Phase 2 of the two-phase commit protocol, the superior TM receives a response from a transaction branch that heuristic completion of a subordinate RM has occurred. The superior TM logs the response and informs the superior AP.

In some cases — for example, after a communication failure — the superior TM may not be able to determine whether heuristic completion of a subordinate RM has occurred. The superior TM logs the fact and informs the superior AP.

3.6.6   Recovery after Failure

Recovery processing in an X/Open DTP system is described in Section 3.4.6 on page 15. In addition, when a global transaction extends across two or more transaction branches, a CRM may request *ax_recover*() of a TM to collect a list of transaction identifiers for prepared transactions for which the TM has information logged on behalf of the CRM.

3.7   CRM Communication Paradigms with APs

3.7.1   The TxRPC Interface

For distributed applications using the remote procedure call (RPC) mechanisms of the X/Open Distributed Computing Services (XDCS) Framework, a Transactional Remote Procedure Call, TxRPC, interface is offered. This allows application writers to invoke remote procedures in the same form as local procedures. Typically the calling environment is an AP referred to as the *client*. The environment where the call is processed is referred to as the *server*. The client calls a remote procedure, and waits for the results of the call. When the server finishes processing the procedure, it returns the results to the client which then resumes execution. The TxRPC interface optionally permits the AP to extend the scope of a global transaction from the client to the server.

Calls that have either *transaction-mandatory* or *transaction-optional* attributes are called *TxRPC operations*. If a TxRPC operation takes place within the scope of a global transaction, the call becomes a *transactional RPC*.

Descriptions of the types of parameters used in the call, plus additional global transaction information semantics, are available to the client and server APs via an *Interface Definition Language* (IDL).

For details of this interface, see the referenced **TxRPC** Specification.

3.7.2   The XATMI Interface

For distributed applications using service requests in a client-server paradigm, X/Open offers the XATMI interface. This interface supports the use of *client* APs requesting services to be performed. A *service* is an AP routine that performs a specific application function on behalf of the client. The structure of the client AP is defined entirely by the application writer and the structure of the service routine is defined by the XATMI interface.

XATMI supports two types of service:

- **Request/Response**

  These services receive a single request from a client AP and produce at most a single response to the request. Requests can be made in two ways:

  — The client AP can make a synchronous request by calling *tpcall*(). The client AP is then suspended until a response is received.

  — The client AP can make an asynchronous request by calling *tpacall*(). The client AP can then continue work while the service routine processes the request. The client AP may, in fact, make other requests and so exploit parallelism since multiple requests can be simultaneously processed by different service routines. The *tpacall*() function also returns a *call descriptor*. This is used later as a parameter by the client AP when it calls *tpgetrply*() to receive a service response to a specific earlier request.

- **Conversational Services**

  These services are invoked by a *tpconnect*() request from the client AP. Once the connection is established and the service invoked, the client and service can exchange data using *tpsend*() and *tprecv*() for an indefinite period of time in a conversational manner. Conversations take place in *half duplex* mode; that is, only one AP can send data at a time. XATMI does not allow the receiver AP to send data until the sender AP yields its control of the conversation. The connection request is completed when the service routine calls *tpreturn*().

For details of this interface, see the referenced **XATMI** Specification.

3.7.3 The CPI-C, Version 2 Interface

For distributed applications using a conversational paradigm, where communication takes place through an application-defined exchange of messages, X/Open offers the CPI Communications (CPI-C), Version 2 interface[5] in addition to the XATMI interface (described above).

The conversational model of program-to-program communication is commonly used in the industry today, and a wide variety of applications are based on this model. The model is historically thought of in terms of two applications *speaking* and *listening*, hence the term *conversation*. A conversation is simply a logical connection between two programs that allows the programs to communicate with each other. From an application's perspective, X/Open CPI-C, Version 2 provides the function necessary to enable this communication.

The CPI-C conversational model is implemented in two major communication protocols: Open Systems Interconnection Distributed Transaction Processing (OSI TP)[6] and Advanced Program-to-Program Communications (APPC). The APPC model is also referred to as logical unit type 6.2 (LU 6.2). X/Open CPI-C, Version 2 provides access to both communication protocols.

A primary benefit of this design is that CPI-C, Version 2 defines a single programming interface to the underlying network protocols across many different programming languages and environments. The interface's rich set of programming services shields the AP from details of system connectivity and eases the integration and porting of the application programs across the supported environments.

Note: The X/Open **CPI-C, Version 2** Specification derives from the CPI-C 2.0 specification[7] produced by the CPI-C Implementors' Workshop, but with the following major differences:

— X/Open CPI-C, Version 2 only supports the C and COBOL programming languages.

— X/Open CPI-C, Version 2 does not support the concept of a *distributed directory*.

3.7.4 Relationships between the Communication Paradigms

Where different applications use the *same CRM communication paradigm*, X/Open facilitates the following goals of paradigm consistency:

- Applications are portable from one TM domain to another.

- Different TM domains can interoperate within the same communication paradigm by using the OSI-TP protocol.

- If the XA+ interface is used, different CRM implementations are interchangeable (this feature is optional).

Relationships between *different CRM communication paradigms* are not determined by the model.

_____

5. This specification supersedes the X/Open **Peer-to-Peer** Snapshot, published in 1992, as one of the three X/Open interfaces between an AP and a CRM.

6. See the referenced OSI TP standards.

7. See the referenced CIW CPI-C Specification.

3.8   High-level TP Language

X/Open has adopted a high-level TP language (HTL) called the Structured Transaction Definition Language (STDL). The purpose of the XHTL is to allow an AP to be written at a higher, language-syntax, level than the relevant X/Open APIs (namely the TX and CRM APIs) which take the form of embedded services. The XHTL is intended to be mappable to all of the X/Open CRMs in order to provide CRM- and communication-paradigm independence. The **STDL** Preliminary Specification is mapped only to the TxRPC CRM.

An AP writer can use the XHTL instead of the relevant X/Open APIs to specify a sequence of operations that involves resources such as databases. The AP writer can use STDL syntax in place of the TX interface to coordinate global transaction management. An AP writer can also use STDL syntax in place of the TxRPC interface for AP-to-AP communications.

STDL implements transactional operations within higher-level syntax such as C or COBOL. AP writers can choose to use STDL when they want to work at the language-syntax level, which provides the benefit of syntax checking, potentially resulting in fewer runtime errors. Using STDL, the AP writer creates separate STDL procedures that contain the transactional operations and that call C, COBOL or possibly other language procedures for other operations. STDL procedure calls are transactional when the called procedures access RMs, and are optionally transactional when calling remote procedures that access RMs.

STDL supports both non-transactional and transactional modes (chained or unchained).

See the referenced **STDL** Specification for further information about STDL syntax and its relationship to the syntax of other languages such as SQL, C and COBOL, as well as the definition of RM access and mapping to the TxRPC protocol.

Frequently Asked Questions

How do existing products' structures compare with the model's structure?

See the questions below.

What elements in the model lead to a transaction processing monitor?

A Transaction Processing Monitor may often include both a TM component and a CRM component. It may also provide additional features outside the X/Open DTP Model such as task scheduling and access to security subsystems.

How does the model handle non-global transactions?

The X/Open DTP Model does not insist that an AP performs all work within the scope of a global transaction. Work performed when no global transaction is defined is termed a *non-global transaction*. In non-global transaction processing, one or more RMs are used by the AP, but are not coordinated by the TM. RM-specific work may be committed using the appropriate native interface commands, such as SQL COMMIT. As far as individual RMs are concerned, the work performed is transactional and the **ACID** transaction properties are obeyed; but when two or more RMs are involved, there is no coordinated commitment between them.

An AP may choose to run *non-global* and *global* transactions within the same TM Domain (see **TM Domain** on page 5 for a definition). The two approaches for such a combination are:

- The AP runs non-global and global transactions in parallel.
- The AP switches between non-global and global transactions, but never runs both types at the same time.

In either case, the coordination of the global and non-global transactions is an additional issue of application design. Most applications should not need to mix non-global and global transactions.

What are RM-internal transactions?

Many RM products structure their own work into transactions. An *RM-internal transaction* is a recoverable unit of work owned by a single RM. In the X/Open DTP Model, a global transaction consists of one or more RM-internal transactions. The TM coordinates the start and completion of the RM-internal transactions of each participating RM.

What elements in the model lead to a distributed database?

In a distributed database, the database RM manages the coordination of that part of the transaction it has internally distributed. Therefore, in the model it is a single RM component. When it receives the transaction instructions from the TM — for example, a request to *prepare*— it transparently prepares any subordinates before responding positively to the TM's *prepare* request.

Where do distributed file systems fit?

A distributed file system can access data at multiple sites, but in general it is not transactional. That is, changes made to such a file system are not coordinated atomically with other actions in the transaction. Such a file system is outside the model. However, it is certainly possible for a distributed file system to be implemented with transactional semantics and to support the XA interface, in which case it would be an RM.

Can I use the model for distributed systems management?

The X/Open DTP Model can be used as a transactional framework for distributed systems management. A CRM that implements CMIP or SNMP, for example, could, in conjunction with an X/Open TM, provide transactional systems management with atomicity across the various objects and sites. The systems management CRM could do the transaction management entirely on its own, or it could use the services of existing RMs to provide ACID storage properties.

Must *commit* occur in the same process that operated on the resources?

The XA specification specifically allows the TM to commit a transaction in a different thread of control from that which operated on the particular resource. In addition, the TM itself may have auxiliary processes to optimise the commit process, and so may the RM.

How does a client with just an AP and communication fit into the model?

Each instance of the X/Open DTP Model must include a TM to provide transaction semantics. A client, operating without a TM, may send requests into a system that implements the model, but such a client is not part of any global transaction. If such a client has local resources, updates to them are not coordinated by the system.

How does a database client-server architecture fit into the model?

A client-server database that supports the XA interface is an RM within the model. As the next question elaborates, the process structure is not identical to the model components. The client process, for example, may be the AP plus libraries from the RM and TM, with the server process just an internal part of the RM.

A client-server database that does not support the XA interface for transaction control is outside the scope of the model.

What is the relationship between model components and processes?

X/Open DTP Model components (APs, TMs, RMs and CRMs) in no way imply a particular process mapping. There are several reasons why a single component may be instantiated with multiple processes, or that multiple components may be collapsed into a single process:

- **Performance**

  A component may have separate processes to handle tasks somewhat asynchronous to the application logic; for example, system management and logging.

- **Internal distribution schemes**

  For example, an RM may be a database with a client-server process model. Such an RM would be expected to provide the correct distributed semantics for its non-visible subordinates.

- **Practicality of linking**

  The model component called the AP needs to communicate with the TM and each RM, and, as such, must have within its process at least some library entry points that give it access to the TM and RM functions. As described above, the TM and RM functions may result in actions being taken by a different process, but the AP needs at least a call-level linkage to them.

An example of how the model might be projected onto processes is shown in the following figure, with ellipses indicating processes and rectangles indicating model components.
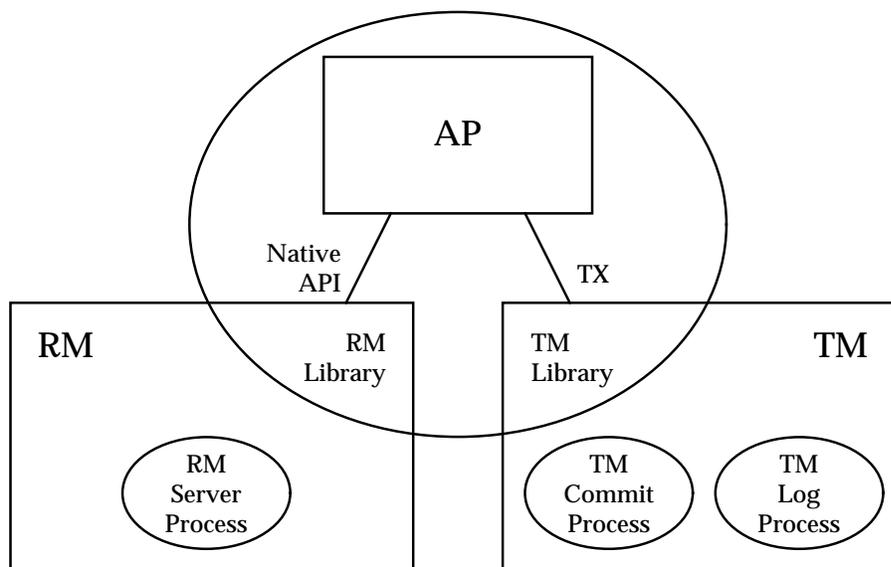
Figure A-1 Projection of Model onto Processes

Is an X/Open TM a transaction monitor?

An X/Open TM is a transaction manager, and as such coordinates the atomic completion of a transaction. A transaction monitor (or transaction processing monitor) typically supplies additional services such as distributed communication, task scheduling and other facilities. An X/Open TM should be thought of as a small, though important, subset of a transaction monitor.

How can I integrate existing products and services with products based on the X/Open DTP Model?

There are two groups of existing products to consider: the system-level components of the model (TMs, RMs and CRMs) and the products that provide services to those components (security subsystems, for example).

The X/Open DTP Model concentrates on transaction atomicity over distributed products and sites. The model does not preclude the incorporation of existing products. In fact, because the X/Open DTP Model allows for native APIs to the RMs, products with existing APIs can be included. A product being incorporated as an RM or TM must have a *two-phase commit* ability that is compatible with the X/Open DTP Model. Also, for an existing product to interoperate with X/Open components, it must provide gateway software between the existing system and the XTP system. If these requirements are met, then the existing product can participate in X/Open transactions.

There are also existing products that supply complementary functions such as security checking or user interface capability. These can also be incorporated. They do not directly support transaction atomicity, but they provide additional features to make a more comprehensive application environment.

Finally, it is possible to map both the X/Open HTL and APIs to existing products to provide additional portability in existing product environments.

How do products implementing the model interoperate?

In terms of system component integration, products that implement the XA or XA+ interfaces can interoperate. In terms of network interoperability, two products that implement the same communication paradigm can interoperate, but two with different communication paradigms cannot. For example, some requests generated by the CPI-C paradigm have no equivalent meaning in TxRPC.

Can I switch easily from one vendor using the model to another?

The model allows for different native interfaces to RMs, and in general you can only switch between RMs that provide the same interface. For example, program recoding would be necessary to move from an RDBMS RM to an ISAM RM, or to move from a TxRPC CRM to an XATMI CRM. However, it is the intent of X/Open that programs are portable between vendors supplying the same RM or CRM API.

Why are there three communication APIs?

The three communication APIs currently supported were developed in response to both user requirements and technology divergence. For example, many users may not require the precise control afforded by the CPI-C paradigm, but applications needing to interact directly with legacy systems may.

Can the HTL and CRM APIs be combined in the same AP?

Yes; for example, the combination of STDL and CPI-C in a single AP is possible, although the precise behaviour of such a combination is not defined.

Index