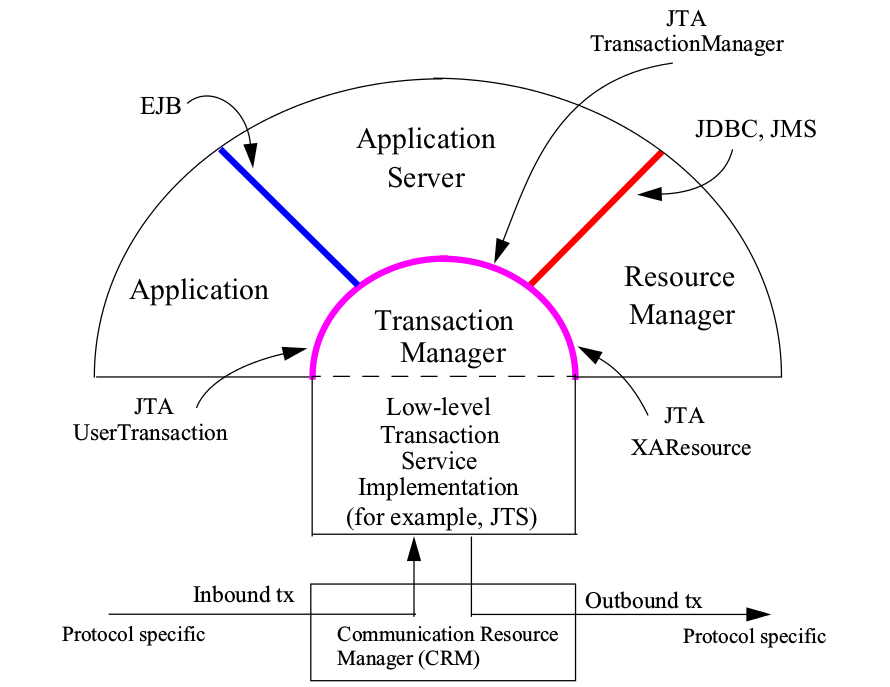
**Java Transaction API**

# **Preamble**

The Java Transaction API (JTA) specifies the local Java interfaces between a transaction manager and the parties involved in a distributed transaction system: the application, the resource manager, and the application server.

The JTA package consists of three parts:

1. **JTA UserTransaction**: A high-level application interface that allows a transactional application to demarcate transaction boundaries.
2. **X/Open XA Mapping**: A Java mapping of the industry standard X/Open XA protocol that allows a transactional resource manager to participate in a global transaction controlled by an external transaction manager.
3. **JTA TransactionManager**: A high-level transaction manager interface that allows an application server to control transaction boundary demarcation for an application being managed by the application server.



**X/Open OR The Open Group**

*X/Open Company, Ltd.* was a consortium founded by several European UNIX systems manufacturers in 1984 to identify and promote open standards in the field of information technology.

*The Open Group* is a vendor and technology-neutral industry consortium, currently with over four hundred member organizations. It was formed in 1996 when *X/Open* merged with the *Open Software Foundation*. Services provided include strategy, management, innovation and research, standards, certification, and test development.

*The Open Group* is most famous as the certifying body for the UNIX trademark, and its publication of the Single UNIX Specification technical standard, which extends the POSIX standards and is the official definition of a UNIX system.

**X/Open Distributed Transaction Processing: XA**

The XA standard is a specification, stands for “eXtended Architecture”, by The Open Group (formerly X/Open) for distributed transaction processing (DTP).

It describes the interface between the *global transaction manager* and the *local resource manager*. The goal of XA is to allow multiple resources to be accessed within the same transaction, thereby preserving the ACID properties across application.

It uses a Two-Phase commit to ensure that all resources either COMMIT or ROLLBACK any particular transaction consistently.

The eXtended Architecture (XA) is an X/Open standard for executing a *global transaction* that accesses more than one back-end resource. It specifies how a transaction manager will roll up the transactions against resources into an *atomic* transaction and execute this with 2PC for the transaction. Thus, it is a type of transaction coordination.

**XA** **Resource**

The XA specification describes what a resource manager must do to support transactional access. Resource managers that follow this specification are said to be XA-compliant.

**Java Transaction Service (JTS)**

The Java Transaction Service (JTS) is a specification for building a transaction manager which supports the JTA interfaces at the high-level and the standard Java mapping of the CORBA Object Transaction Service (OTS) specification at the low-level.

**Resource**

e.g. Database

**Resource Manager**

e.g. Database server

**Resource Adapter**

e.g. JDBC driver, JMS provider

**Communication Resource Manager (CRM)**

A communication resource manager (CRM) allows the transaction manager to participate in transactions initiated by other transaction managers.

It supports transaction context propagation and access to transaction service for incoming and outgoing requests. – Interoperability between transaction managers.

**Transaction Manager**

The transaction manager is the core component of this architecture and is an implementation of Java Transaction Service (JTS) specification.

It provides interfaces to create transactions (including transaction demarcation and propagation of transaction context), allows enlistment and delistment of resources, provides interfaces for registering components for application synchronization, implements the synchronization protocol, and initiates and directs the two phase commit and recovery protocol with the resource managers.

Allows application server to control transaction boundaries on behalf of the application being managed. The transaction manager maintains the transaction context association with threads as part of its internal data structure. Multiple threads may concurrently be associated with the same global transaction.

**Transaction**

A transaction is a related collection of operations on the application state, and has ACID properties.

In other words, a transaction is a unit of work done by multiple distributed components on shared data with ACID properties.

Transactions,

* allow us to modify the persistent state of an application in a controlled manner.
* provide a framework for enforcing data consistency and integrity in the face of system or component failures.

**Application State**: All of the in-memory and on-disk data items that affect the application's operation; everything the application knows.

**ACID Properties**

**Atomicity:** Either all of the transactions' operations are applied to the application state or none of them are applied (All-Or-Nothing nature). – the transaction is an indivisible unit of work.

**Consistency:** The transaction represents a correct transformation of the application state. – any integrity constraints implicit in the application are not violated by the transaction.

**Isolation:** The effects of one transaction do not affect other transactions that are executing concurrently.

**Durability:** Once a transaction successfully completes, changes to the application state will survive failures.

**Local Transaction Vs Global Transaction**

A transaction which deals with a single resource manager is called as local transaction in JTA. And a transaction which manages multiple resource managers is called a Global transaction.

**Anatomy of a transaction**

1. Application

A transaction application hosted on an application server container.

1. Transaction Processing Monitor (TPM)

An application server (TPM) provides the infrastructure required to support the application run-time environment. The TPM coordinates the activities of the RMs to ensure the all-or-nothing nature of a transaction. e.g. EJB Server

1. Resource Managers (RMs)

The RMs store the application state through a resource adapter. e.g. database server, message queue server, or other transactional resources.

**Steps:**

1. A transaction begins when the application asks the container or transaction monitor to start a new

transaction.

1. As the application accesses various RMs, they are enlisted in the transaction.
2. The RM must associate any changes to the application state with the transaction requesting the changes.
3. A transaction ends when one of two things happens: the transaction is committed by the application, or the transaction is rolled back either by the application or because one of the RMs failed.
4. If the transaction successfully commits, changes associated with that transaction will be

written to persistent storage and made visible to new transactions.

1. If it is rolled back, all changes made by that transaction will be discarded; it will be as if the transaction never happened at all.

# **Single Phase Commit(1PC) protocol**

# **Two Phase Commit (2PC) protocol**

# **Three Phase Commit (3PC) protocol**

# **Compensation Commit protocol**