15

Transactions

Objectives

After completing this lesson, you should be able to:

- Describe WebLogic Server’s role in managing transactions
- Configure a database persistent store for transactions

Transactions and ACID

- A transaction is a mechanism to handle a group of operations as if they were one. It is a unit of work.
- Transactions have four key properties (ACID).
  - **A**tomic: The entire sequence of operations must either be completed successfully or be as if none of them occurred at all. The transaction cannot be partially successful.
  - **C**onsistent: A transaction transforms a system from one valid state to another valid state.
  - **I**solated: Each transaction occurs independently. Its effect is not visible until it has completed.
  - **D**urable: Completed transactions remain permanent, even during system failure.

Global Transactions, 2PC, and XA

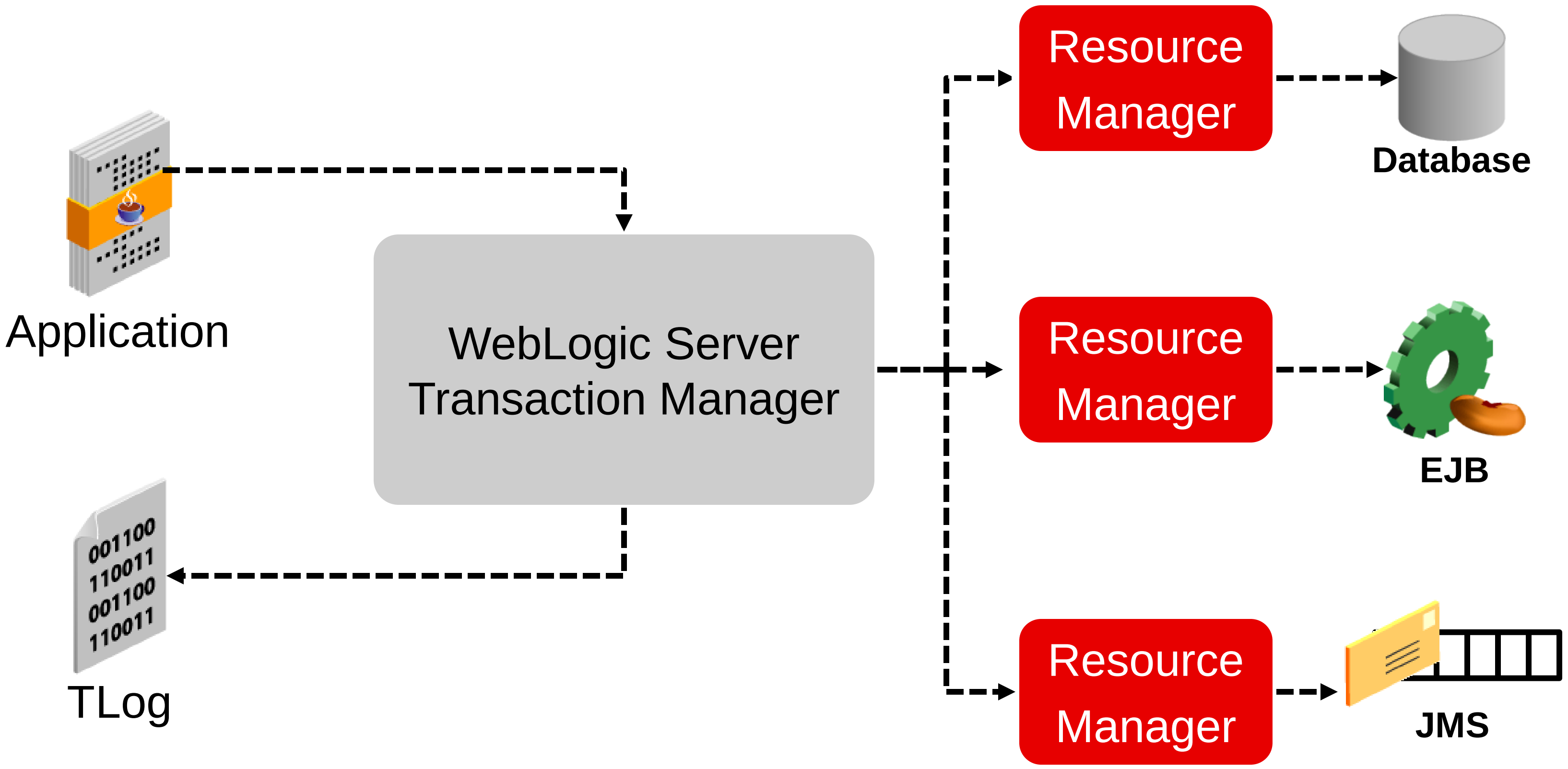
- A global (distributed) transaction involves more than one transactional resource.

WebLogic Server can act as a TM.

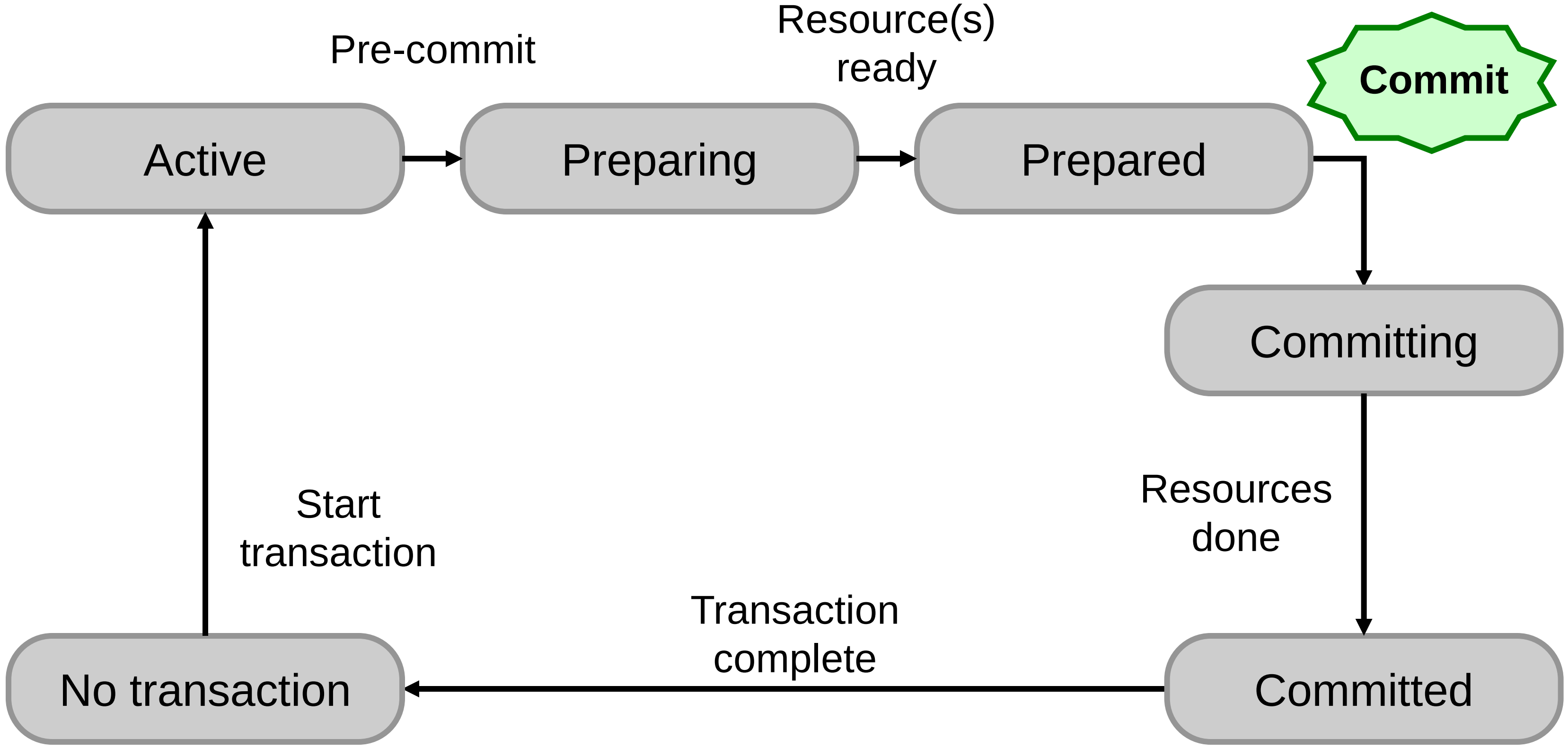
  - A transaction manager (TM) deals with each resource manager (RM).
- The Two-Phase Commit (2PC) protocol uses two steps to commit changes within a global transaction:
  - Phase 1: TM asks RMs to prepare to make the changes.
  - Phase 2: If all RMs report that they are ready to commit, TM tells the RMs to commit, which makes the changes permanent. If any RM is not ready to commit, TM tells all RMs to roll back (undo any changes).
- The Extended Architecture (XA) specification implements the 2PC protocol.

WebLogic Server as a Transaction Manager

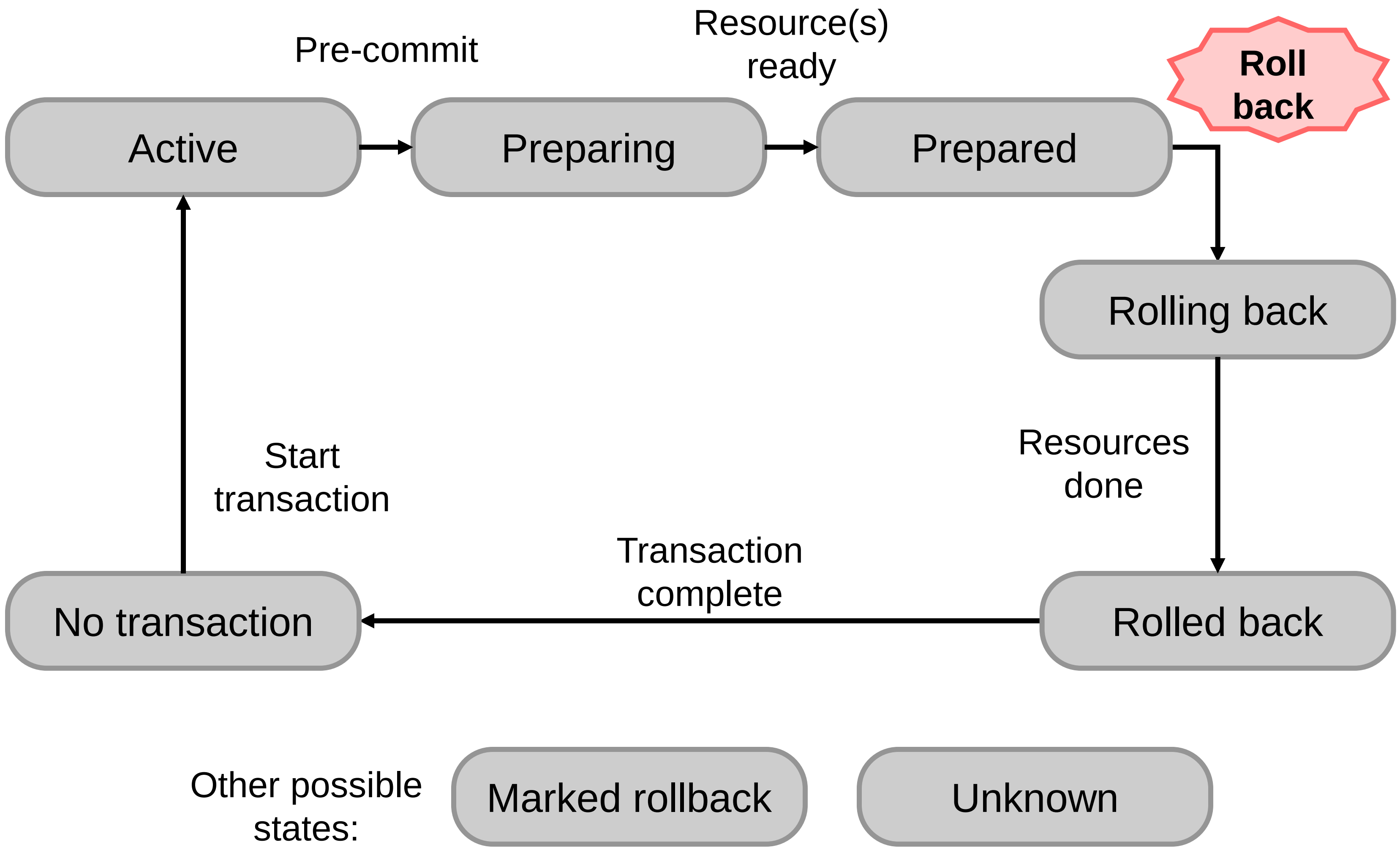
WebLogic Server coordinates a global transaction with the various transactional resource managers involved.



Transaction States when Committing



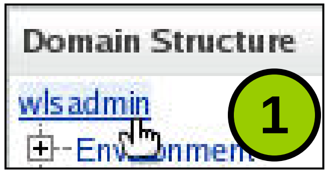
Transaction States when Rolling Back

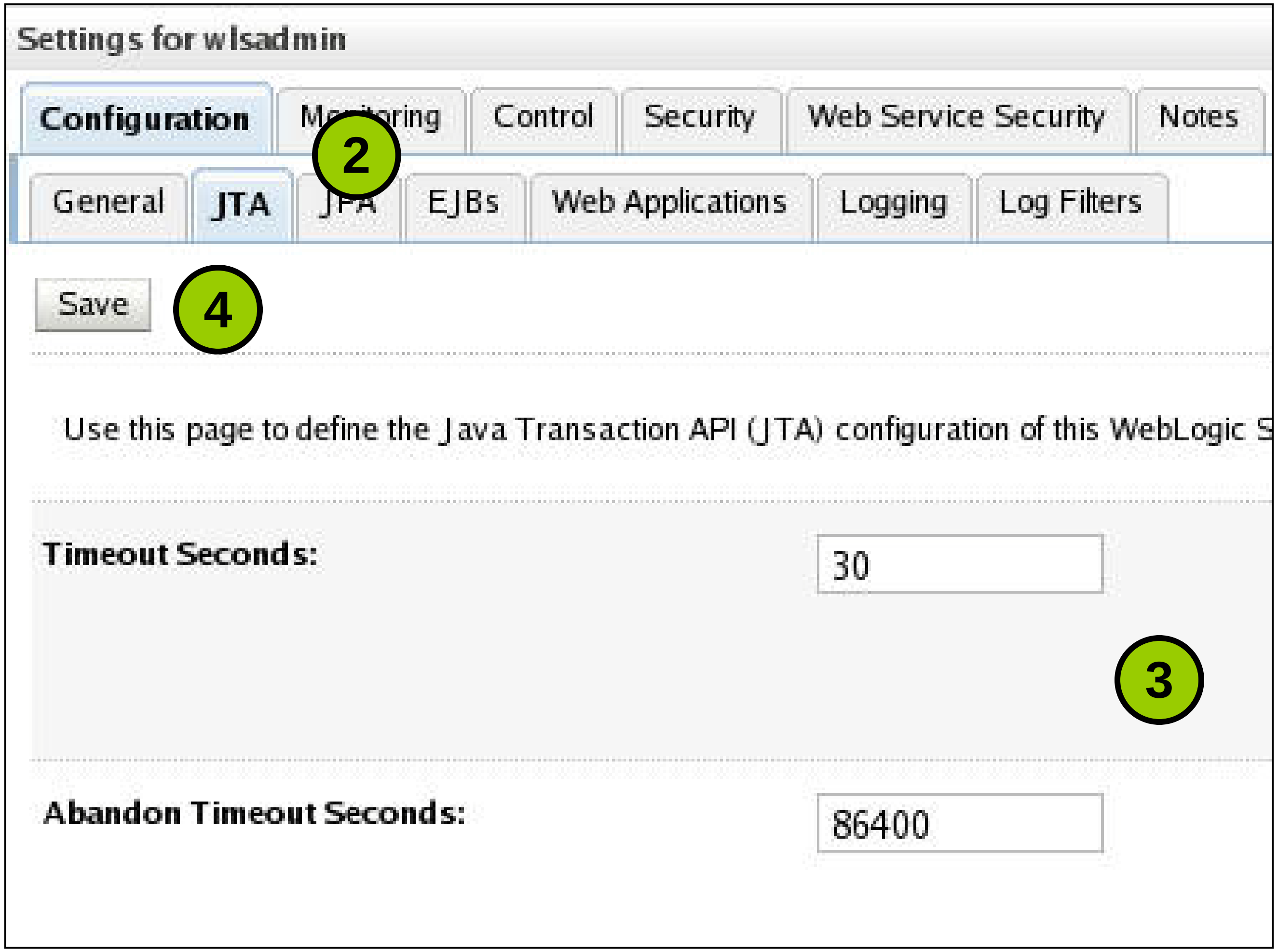


Java Transaction API (JTA)

- WebLogic Server uses JTA to implement and manage global transactions.
- WebLogic Server's JTA implementation:
  - Creates a unique transaction identifier (`XID`)
  - Supports an optional transaction name
  - Tracks objects involved in transactions
  - Notifies databases of transactions
  - Orchestrates 2PC using XA
  - Executes rollbacks
  - Executes automatic recovery procedures in the event of failure
  - Manages timeouts

Configuring Transactions





Use this page to define the Java Transaction API (JTA) configuration of this WebLogic S

Timeout Seconds: 30

Abandon Timeout Seconds: 86400

JTA Configuration Options

Field	Description
Timeout Seconds	The time the TM waits for a new transaction to go into the prepared state
Abandon Timeout Seconds	The time the TM waits for a transaction to go from prepared to committed
Before Completion Iteration Limit	The maximum number of times the TM will call the `beforeCompletion()` method. This allows interested objects to take part in notifications as the transaction progresses.
Max Transactions	The maximum number of simultaneous in-progress transactions allowed on a server
Max Unique Name Statistics	The maximum number of unique transaction names for which statistics will be maintained
Checkpoint Interval Seconds	How often the TM creates a transaction log and checks to see if old logs can be deleted

JTA Configuration Options

Field	Description
Forget Heuristics	Whether or not the TM will automatically perform a "forget" operation for resources reporting a heuristic decision. The default is true. Disable this only if you know what to do with resources reporting heuristic decisions.
Unregister Resource Grace Period	The seconds the TM waits for transactions to complete before unregistering a resource (for example, when a data source is undeployed. it is unregistered). If at that time transactions are still outstanding, a log message is written.
Execute XA Calls in Parallel	XA calls are executed in parallel if threads are available.
Enable Two Phase Commit	Use 2PC for global transactions.

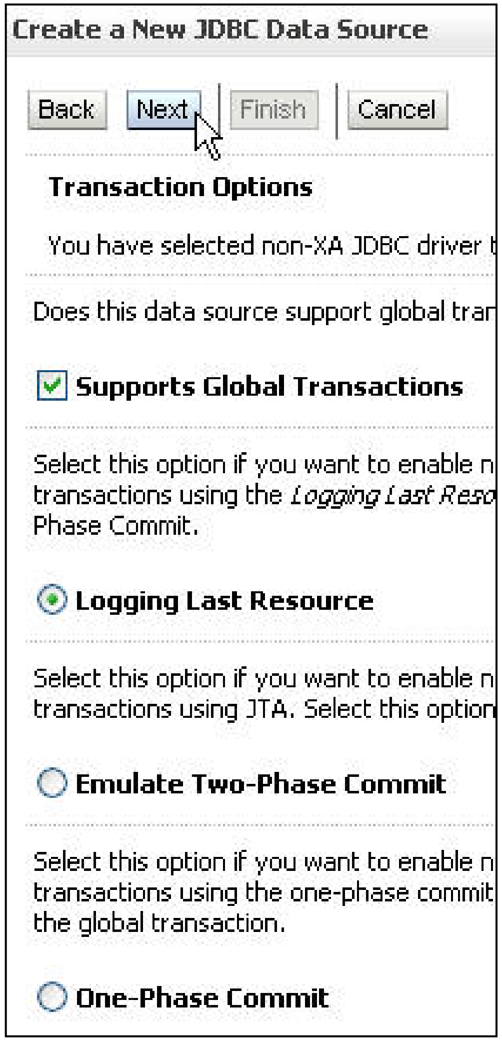
WebLogic Extension of JTA

Developers writing transactional code to run on WebLogic Server have available WebLogic-specific extensions to standard JTA.

- The WebLogic JTA transaction object supports the `weblogic.transaction.Transaction` interface (which extends `javax.transaction.Transaction`).
  - This adds various capabilities, the most important of which, to an administrator, is the ability to name transactions.
    - If developers write their code well, transactions can have business names (not just transaction IDs), which makes statistics and error messages more meaningful.

JDBC Reminder

- For your database to participate in global transactions, choose an XA driver when creating the data source.
- If you must choose a non-XA driver, select **Supports Global Transactions**, and then select how the driver will support them. The recommendation is **Logging Last Resource**.
  - The resource is processed last. If it succeeds, the other resources are told to commit; if it fails, they are told to roll back.



Transaction Options

You have selected non-XA JDBC driver t

Does this data source support global tran

[✓] **Supports Global Transactions**

Select this option if you want to enable n
transactions using the *Logging Last Reso*
Phase Commit.

[✓ **Logging Last Resource**]

Select this option if you want to enable n
transactions using JTA. Select this option

Emulate Two-Phase Commit

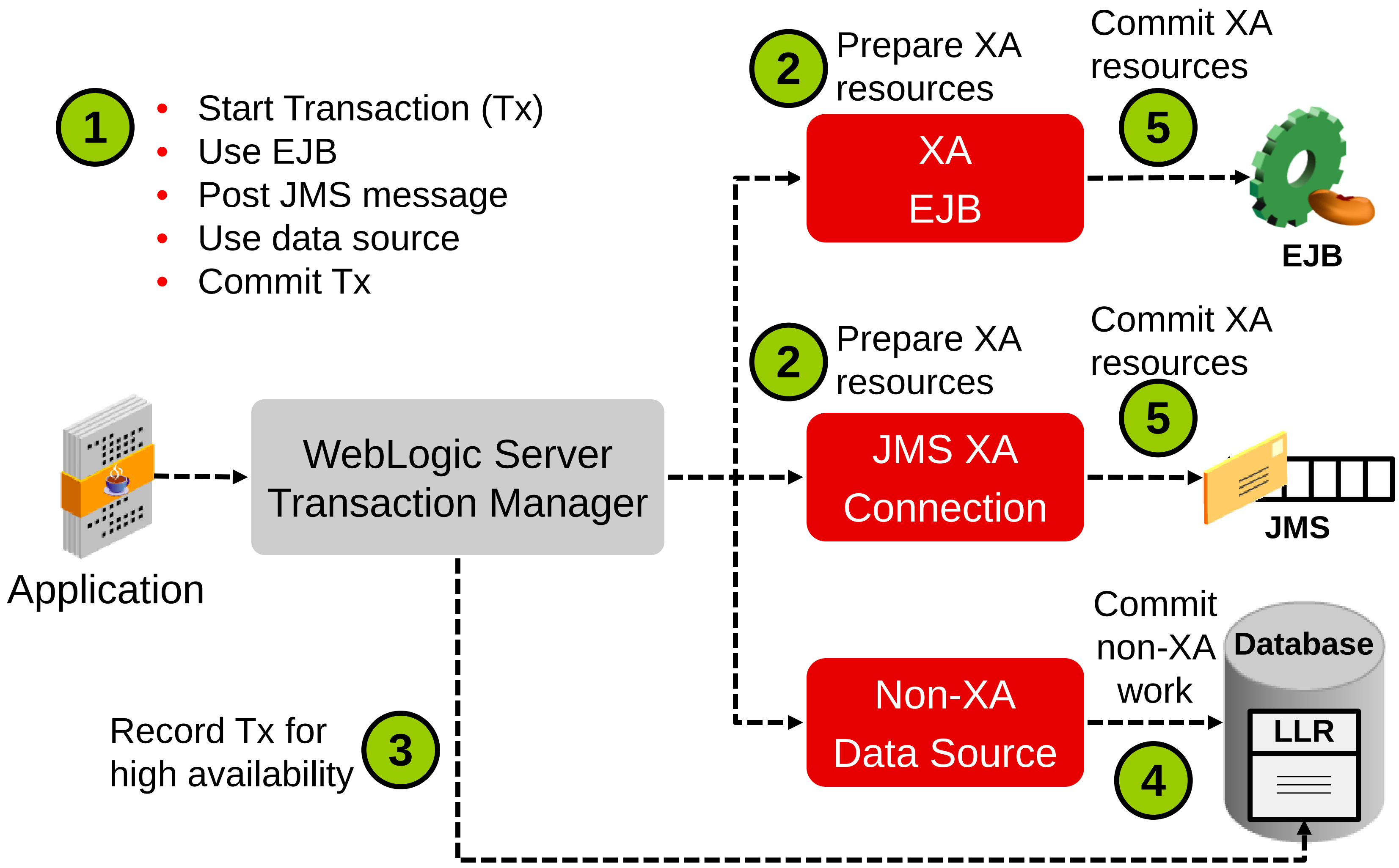
Select this option if you want to enable n
transactions using the one-phase commit
the global transaction.

One-Phase Commit

Logging Last Resource and Performance

- Even if XA drivers are available, you may want to configure your data source with a non-XA driver and select Logging Last Resource (LLR), if this data source represents the only database participant in your global transactions. This will improve performance and has the same ACID guarantee as XA.
- Non-XA drivers with LLR improves performance by:
  - Removing the need for an XA JDBC driver to connect to the database. XA JDBC drivers are typically less efficient than non-XA JDBC drivers.
  - Reducing the number of processing steps to complete the transaction, which also reduces network traffic and I/O.
  - Removing the need for XA processing at the database level (if the database is the only non-XA resource).

LLR: Example


1
Start Transaction (Tx)
Use EJB
Post JMS message
Use data source
Commit Tx
Application
WebLogic Server Transaction Manager
2
Prepare XA resources
XA EJB
5
Commit XA resources
EJB
2
Prepare XA resources
JMS XA Connection
5
Commit XA resources
JMS
Non-XA Data Source
Commit non-XA work
4
Database
LLR
3
Record Tx for high availability

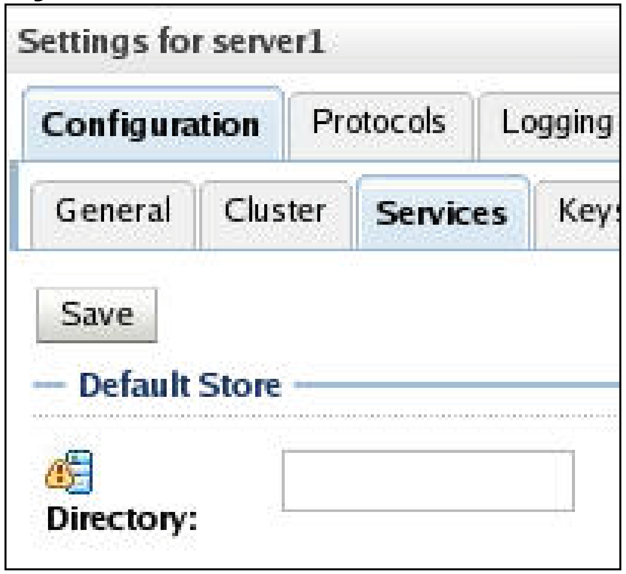
Transaction Log (TLog)

- During a transaction, the server writes to a binary transaction log (TLog).
  - The transaction log is not like other WebLogic Server logs. You do not view it (it is binary).
- If the server fails, when it is restarted, it reads its TLog to be able to recover transactions.
- If the server cannot be brought back up on the same machine due to a hardware failure, it can be started on a new hardware.

The TLog must be available.

Configuring the Default Store

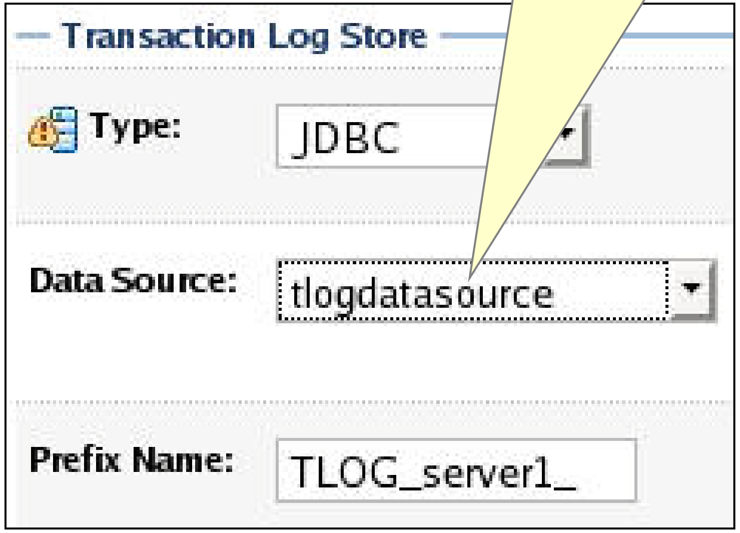
- The transaction log can be file based or in a database.
  - By default, the transaction log uses the default store.
    - The default store is file based.
    - If the default store directory is not set, it is here by default: `<domain>/servers/<server>/data/store/default`
- To change the default store directory:
  1. Select a server from the Servers table and then select **Configuration** > **Services**.
  2. Under Default Store, change the Directory field. For transaction recovery, change it to some reliable shared storage directory.



Configuring a JDBC Transaction Log

- To use a database transaction log:
  1. Select a server from the Servers table and then select **Configuration** > **Services**.
  2. Under Transaction Log Store, use the Type drop-down list and select **JDBC**.
  3. Use the Data Source drop-down list to select a data source that has already been created and configured.
  4. Enter a Prefix Name for the table.

This data source cannot use an XA driver nor support global transactions.


Transaction Log Store

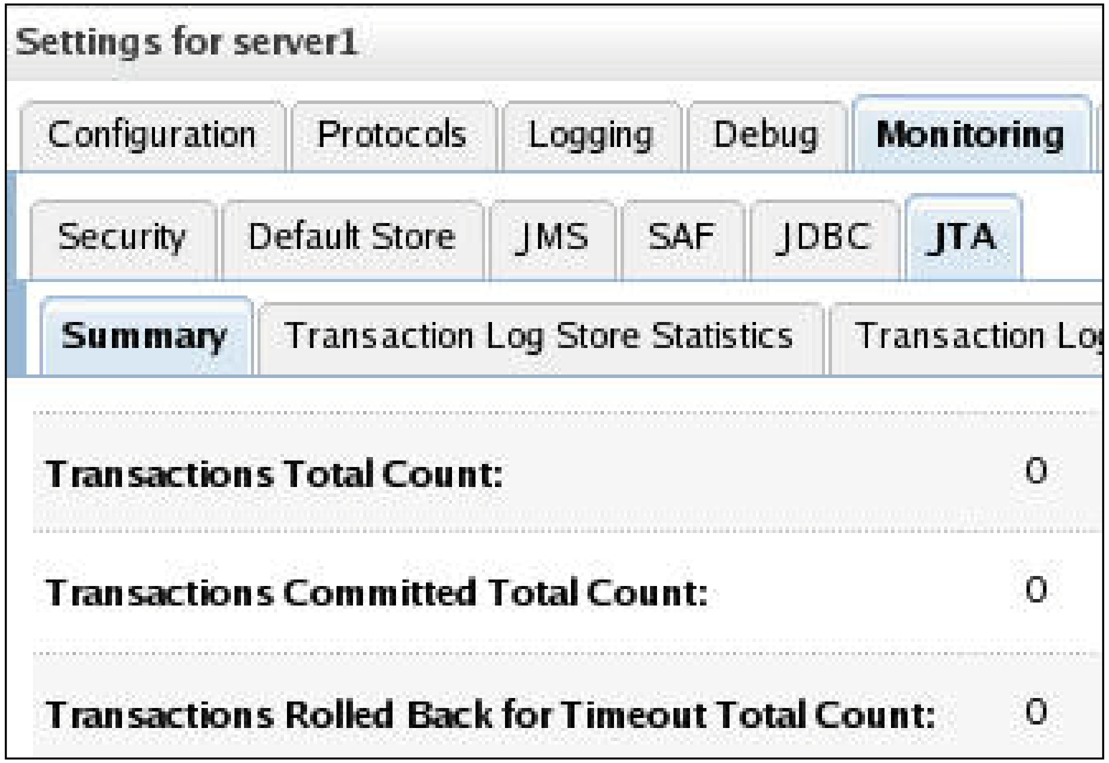
Type:	JDBC
Data Source:	tlogdatasource
Prefix Name:	TLOG_server1_

Comparing File Store to JDBC Store

	File Store	JDBC Store
Default store	It is the default	Cannot be used for the default
Transactions	Both have the same transaction guarantees and semantics	
Interface	The application interface is the same	
Throughput	Better	Worse
Configuration	Easier	Harder
Failure recovery	The file store must be configured to reside on shared storage	Made easier as all servers can use JDBC (data sources) to access the store
Backup and recovery	Shared storage another item to back up	Simplifies it, as database backup and recovery can include the TLog

Monitoring Transactions

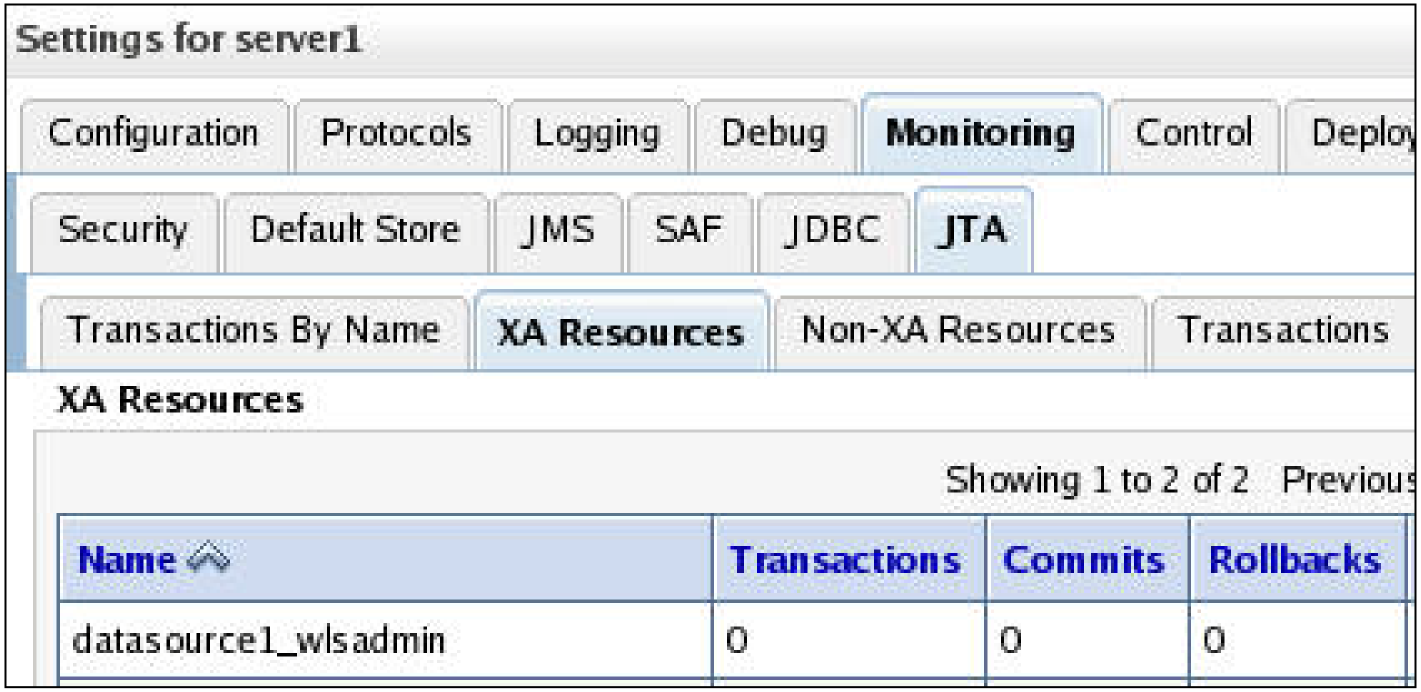
- Select a server from the Servers table and then select **Monitoring** > **JTA**. Then select one of the many subtabs under **JTA**.



Transactions Total Count:	0
Transactions Committed Total Count:	0
Transactions Rolled Back for Timeout Total Count:	0

Viewing Transaction Statistics for a Resource

To view transactional outcomes for a particular resource: Select the server, then **Monitoring** > **JTA** and either **XA Resources** or **Non-XA Resources**.


Settings for server1

Configuration | Protocols | Logging | Debug | **Monitoring** | Control | Deploy

Security | Default Store | JMS | SAF | JDBC | **JTA**

Transactions By Name | **XA Resources** | Non-XA Resources | Transactions

XA Resources

Showing 1 to 2 of 2 Previous

Name	Transactions	Commits	Rollbacks
datasource1_wlsadmin	0	0	0

Forcing a Commit or Rollback

Under a server’s **Monitoring** > **JTA** > **Transactions** tab, current transactions are listed, with information about them, including their status.

- If a transaction is “stuck,” due to some system or network failure, eventually the Abandon Timeout period will elapse and the transaction will be removed (with a heuristic error written to the server log). Before then, the transaction can be selected and a button pressed to force the transaction to completion. The buttons are:
  - Force Local Commit
  - Force Global Commit
  - Force Local Rollback
  - Force Global Rollback

Forcing a Commit or Rollback

Transaction Status	Can Use Force Commit?	Can Use Force Rollback?
Active		Yes
Preparing		Yes
Prepared	Yes	Yes
Committing	Yes	
Committed	Yes	
Rolling Back		Yes
Rolled Back		Yes
Marked Roll Back		Yes
Unknown	Yes	Yes

Troubleshooting Transactions

- Use the monitoring capabilities of the administration console.
  - With all the possible states of a transaction, you can see how far along in the process an active transaction is.
- Use the server logs.
  - If a message is logged during a transaction, the transaction ID is part of that log message.
  - Look for exceptions in the logs. WebLogic JTA supports all the standard JTA exceptions.
    - It extends the `RollbackException` class to preserve the original reason for the rollback.
  - Turn on transaction debug flags for more detailed log messages.

Quiz

A global (distributed) transaction involves more than one ________________________.

a. WebLogic Server
b. Cluster
c. Transactional resource
d. Domain
e. Continent

Quiz

JTA stands for:

a. Just in Time Architecture
b. Java Transaction Architecture
c. Job Transaction API
d. Java Transaction API

Summary

In this lesson, you should have learned how to:

- Describe WebLogic Server’s role in managing transactions
- Configure a database persistent store for transactions

Practice 15-1 Overview: Configuring Transaction Persistence

This practice covers configuring a database as the persistent store for transaction logs.