ArjunaTS support for XA recovery

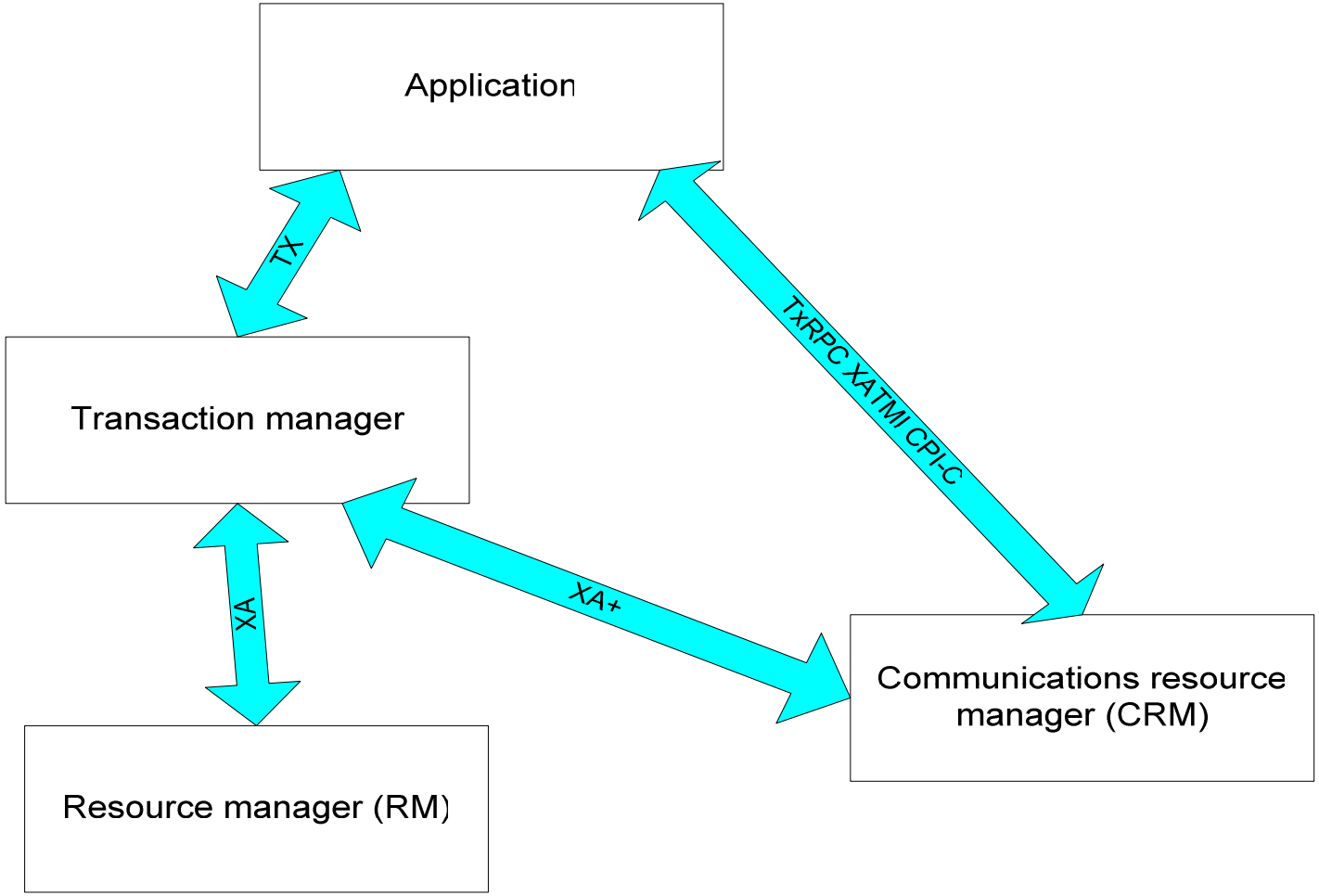
arjuna
middleware for reliability

XA recovery basics

- XA defines “presumed nothing” protocol
  - Recovery driven from the coordinator
  - Participants don’t maintain coordinator reference
- Administrator management for participants is default
  - Can lead to heuristic outcomes

XA architecture

arjuna
middleware for reliability



Disadvantages

- Slower recovery time
  - Resources remain blocked until coordinator recovers
- Heuristic outcomes can be common
- RMs may be shared between many different transaction sources
  - Resolution becomes even more difficult/risky
- Administrator cannot tell the difference between slow coordinator and failed coordinator

Advantages

- Quicker 2PC
  - Participant doesn't have to make coordinator reference durable
    - Doesn't make a different for one-phase commit

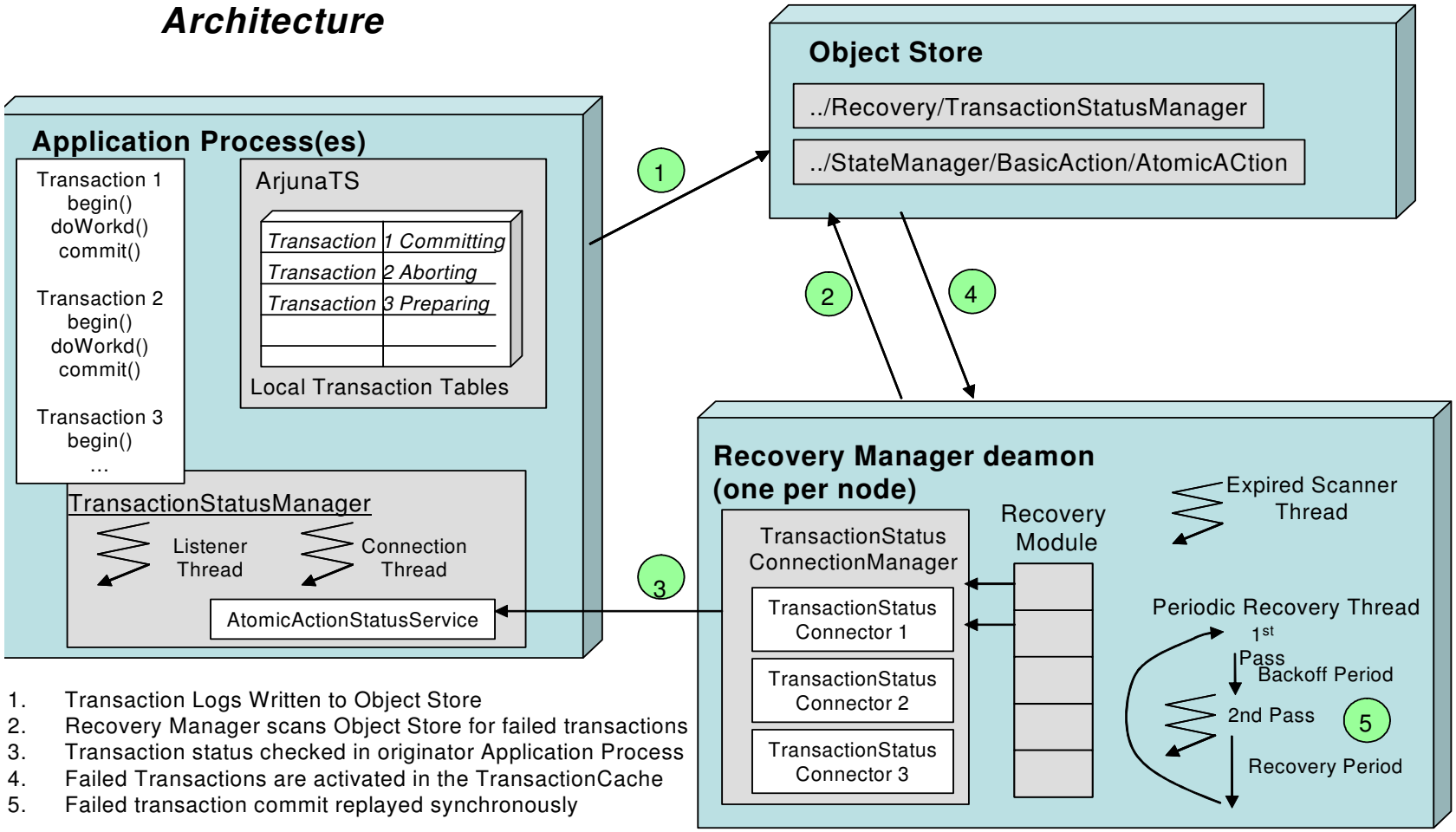
ArjunaTS approach

- "Presumed nothing" protocol supported
  - Nothing we can do about this (see previous issues)
- "Presumed abort" protocol provided
  - Participants transparently log information about coordinator
    - Allows participant driven recovery to occur automatically

Recovery architecture

arjuna
middleware for reliability

Crash Recovery Architecture



1. Transaction Logs Written to Object Store
2. Recovery Manager scans Object Store for failed transactions
3. Transaction status checked in originator Application Process
4. Failed Transactions are activated in the TransactionCache
5. Failed transaction commit replayed synchronously

RecoveryManager

- Recovery manager drives RecoveryModules
  - Typically one module per type of recoverable resource
    - E.g., file system, XAResources
- Runs periodically but can be driven directly
  - com.arjuna.ats.arjuna.tools.RecoveryMonitor

XA specifics

- XAResources are the participants that represent the backend RMs
- ATS supports two types of implementation
  - Serializable XAResources
  - Non-serializable XAResources

Serializable XAResources

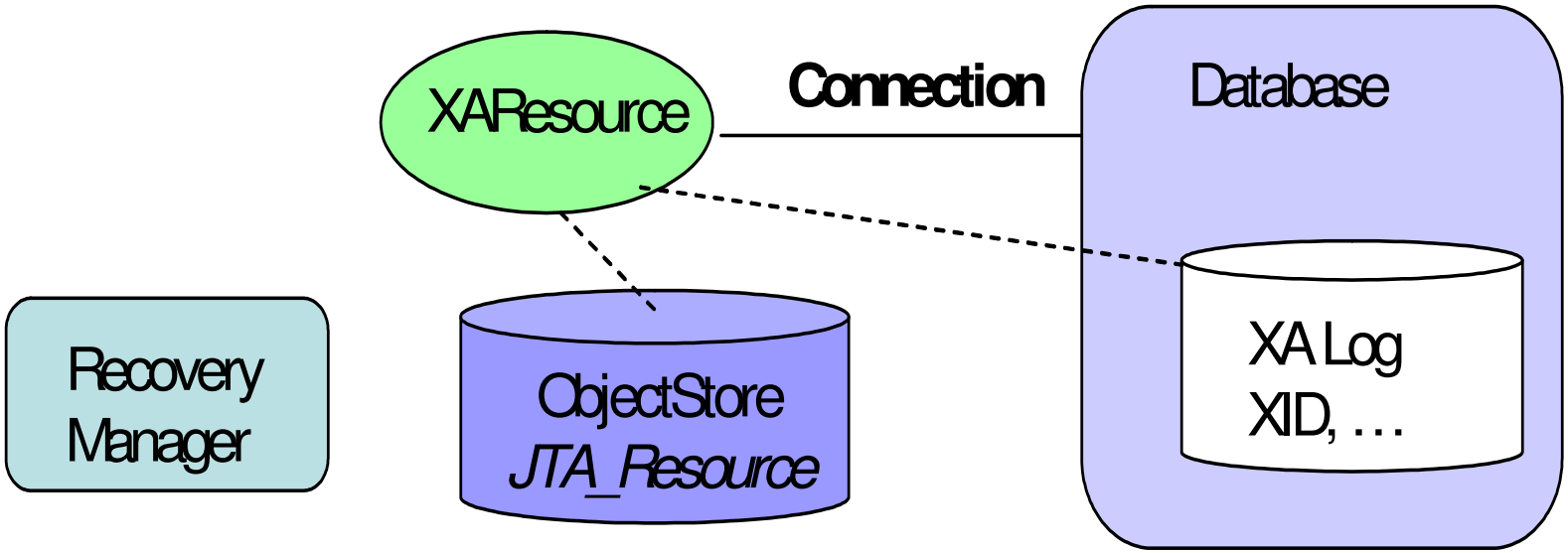
- Coordinator serializes reference to participant
- ATS XAResource wrapper records
  - Serializes state after successful prepare
  - Information on coordinator too

Non-serializable XAResources

- Coordinator serializes reference to participant
- ATS XAResource wrapper records
  - The fact that it needs a new XAResource instance for recovery
  - information on coordinator

Illustration

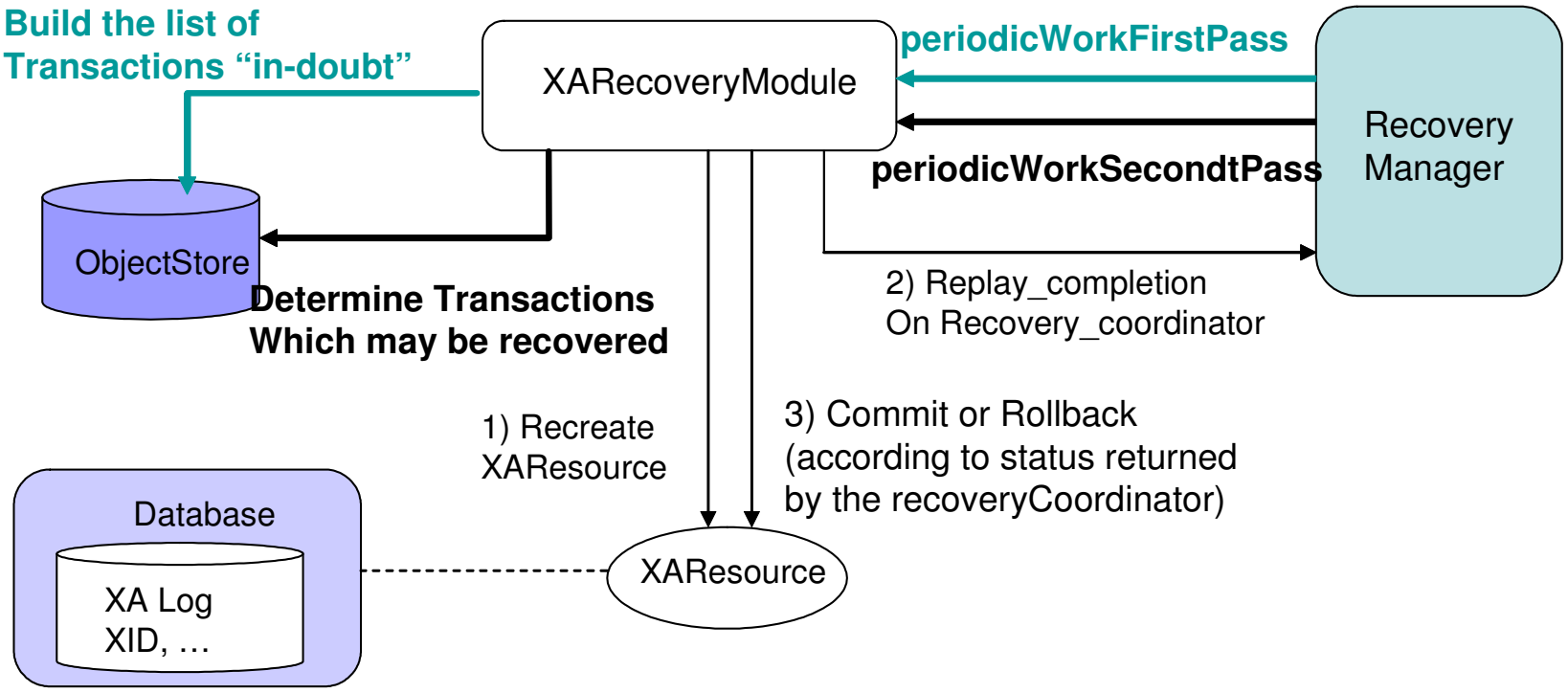
arjuna
middleware for reliability



Recovery from the coordinator

- Coordinator driven recovery identical for both types of XAResource
  - contacts remote Recovery Manager
    - Recovers specific XAResource wrapper state
    - Drives recovery on that instance
  - Has same issues as participant-driven recovery (next)

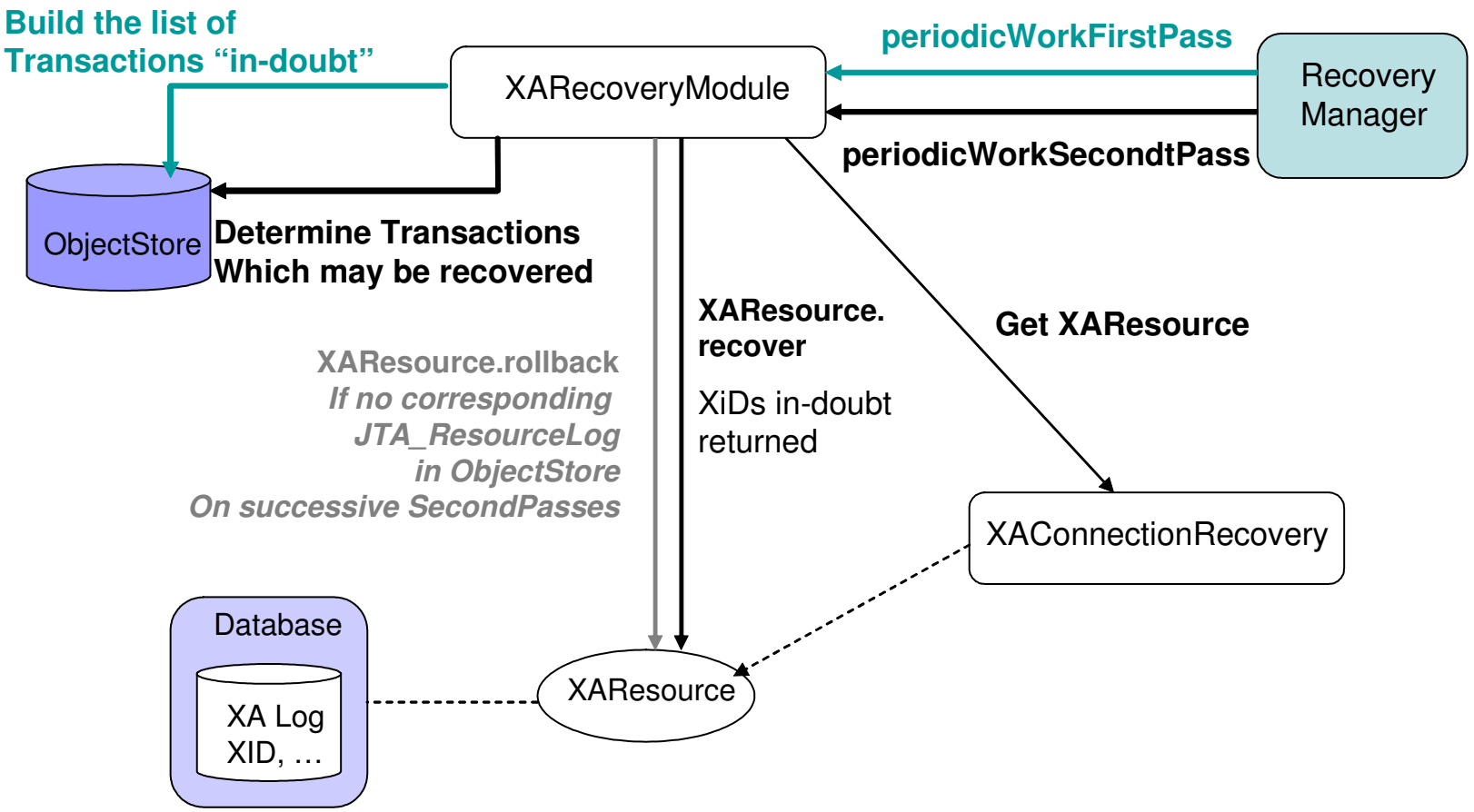
Example


Build the list of
Transactions “in-doubt”
XARecoveryModule
periodicWorkFirstPass
Recovery
Manager
periodicWorkSecondtPass
ObjectStore
Determine Transactions
Which may be recovered
2) Replay_completion
On Recovery_coordinator
1) Recreate
XAResource
3) Commit or Rollback
(according to status returned
by the recoveryCoordinator)
Database
XA Log
XID, …
XAResource

Recovery from the participant

- Participant-driven recovery complex
  - XA Recovery Module recreates XA wrapper
  - If XAResource was serializable then
    - Assume recreated instance can drive RM through recovery
    - End of story
  - If it wasn't serializable then need a new instance
    - XAConnectionRecovery

XAConnection Recovery



How it works

- Remember, ATS records information about resources after prepare in its log
  - XAResource wrapper does the same
- If we can't recover a wrapper instance because it needs a new XAResource
  - Call each XAConnectionRecovery instance until we get useable XAResource
    - Drive it through recovery

Final recovery phase

- Use XAResource.recover (from recorded instances as well as XAConnectionRecovery)
  - All Xids returned that are in our (coordinator/wrapper) log cannot be recovered yet
    - Cannot contact coordinator
  - All others must represent rolled back transactions
    - Roll them back

A word of warning

- XAConnectionRecovery can return many XAConnection instances
  - Manage recovery for multiple RMs
- But recovery module assumes RMs are only used by ATS
  - It will eventually roll back *all* transactions for which it does not have information in the log

Example


```
/**
 * This class implements the XAConnectionRecovery interface for XAResources.
 * The parameter supplied in setParameters can contain arbitrary information
 * necessary to initialise the class once created. In this instance it contains
 * the name of the property file in which the db connection information is
 * specified, as well as the number of connections that this file contains
 * information on (separated by ;).
 *
 * IMPORTANT: this is only an *example* of the sorts of things an
 * XAConnectionRecovery implementor could do. This implementation uses
 * a property file which is assumed to contain sufficient information to
 * recreate connections used during the normal run of an application so that
 * we can perform recovery on them. It is not recommended that information such
 * as user name and password appear in such a raw text format as it opens up
 * a potential security hole.
 *
 * The db parameters specified in the property file are assumed to be
 * in the format:
 *
 * DB_x_DatabaseURL=
 * DB_x_DatabaseUser=
 * DB_x_DatabasePassword=
 * DB_x_DatabaseDynamicClass=
 *
 * DB_JNDI_x_DatabaseURL=
 * DB_JNDI_x_DatabaseUser=
 * DB_JNDI_x_DatabasePassword=
 *
 * where x is the number of the connection information.
*/
```

Initialise the instance

- Locate the property file in which the connection information is located
  - Passed in via the initialise method parameter
  - Can be any arbitrary String

Example file


```
/*
 * Example:
 *
 * DB2_DatabaseURL=jdbc\:arjuna\:sequelink\://qa02\:20001
 * DB2_DatabaseUser=tester2
 * DB2_DatabasePassword=tester
 * DB2_DatabaseDynamicClass=com.arjuna.ats.internal.jdbc.drivers.sequelink_5_1
 *
 * DB_JNDI_DatabaseURL=jdbc\:arjuna\:jndi
 * DB_JNDI_DatabaseUser=tester1
 * DB_JNDI_DatabasePassword=tester
 * DB_JNDI_DatabaseName=empay
 * DB_JNDI_Host=qa02
 * DB_JNDI_Port=20000
 */
```

initialise method


```
public boolean initialise (String parameter) throws SQLException
   {
        int breakPosition = parameter.indexOf(BREAKCHARACTER);
        String fileName = parameter;

        if (breakPosition != -1)
        {
           fileName = parameter.substring(0, breakPosition -1);

           try
           {
                    numberOfConnections = Integer.parseInt(parameter.substring(breakPosition +1));
           }
           catch (NumberFormatException e)
           {
                    return false;
           }
        }

        try
        {
           String uri = com.arjuna.common.util.FileLocator.locateFile(fileName);
           jdbcPropertyManager.propertyManager.load(XMLFilePlugin.class.getName(), uri);

           props = jdbcPropertyManager.propertyManager.getProperties();
        }
        catch (Exception e)
        {
            return false;
        }

        return true;
   }
```

getConnection method


```
public synchronized XAConnection getConnection () throws SQLException
  {
      JDBC2RecoveryConnection conn = null;  // specific to this example

      if (hasMoreConnections())
      {
          connectionIndex++;

            String number = new String(""+connectionIndex);
            String url = new String(dbTag+jndiTag+number+urlTag);
            String password = new String(dbTag+jndiTag+number+passwordTag);
            String user = new String(dbTag+jndiTag+number+userTag);

            Properties dbProperties = new Properties();

            String theUser = props.getProperty(user);
            String thePassword = props.getProperty(password);

            if (theUser != null)
            {
                dbProperties.put(TransactionalDriver.userName, theUser);
                dbProperties.put(TransactionalDriver.password, thePassword);

                return new JDBC2RecoveryConnection(url, dbProperties).recoveryConnection().getConnection(); // again, specific only to this example
            }
            else
                return null;  // error
        }

        return null;
}
```