

Cloud*Tran*

CloudTran-Coherence v0.2

For Windows and Eclipse -

CloudTran-Coherence v0.2 Reviewer's Guide

This document walks you through the steps to run and evaluate CloudTran for Coherence.

It covers version 0.2, using Windows and Eclipse as the development environment.

CloudTran integrates into a typical TopLink Grid application using Coherence in Entity Grid mode (with all transactional data in memory).

The goal of this distribution is to demonstrate how to use a TopLink Grid application with Coherence and CloudTran to do high-speed transactions coordinated between the Coherence Grid and a database.

Download Eclipse and CloudTran-Coherence

To get started, you need to download both Eclipse and the CloudTran software.

Eclipse

For Eclipse, Indigo (3.7) is best, of which any of the multiple variants should work: we have tested with the 'Java' variant downloaded from <http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/indigo>. If you have an existing Helios (3.6) installation, that should work too.

We'll call the **Eclipse directory** `<eclipse>` (by default this is C:\eclipse).

CloudTran

Download the CloudTran-Coherence v0.2 distribution from

<http://www.cloudtran.com/downloads/CloudTran-Coherence/v0.2/CtCoh.zip>

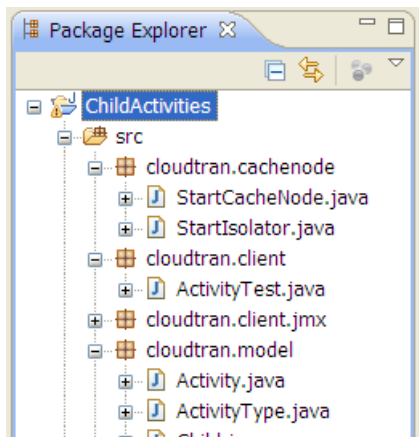
and unzip this. The top-level directory created by unzipping will be 'CloudTranCoherence'; we'll refer to this directory as `<CtCoh>`. Within CloudTranCoherence, the shipment contains

- doc/ documentation – this document and the manual
- examples/ the examples directory, holding the ChildActivities project
- lib/ the CloudTran and Coherence jars and other required jars
- MySQL the MySQL database binaries and data for the example project

The Application

The examples directory doubles as an Eclipse workspace; the ChildActivities directory is an Eclipse project.

So to access the application, run Eclipse and specify the workspace `<CtCoh>\examples`. You should see something like this:



CloudTran is additional functionality on top of Oracle TopLink with Coherence, which is described here:

http://download.oracle.com/docs/cd/E17904_01/doc.1111/e16596/configipa.htm

In particular, CloudTran adds functionality to the Grid Entity configuration (see section 2.5.4, “Grid Entity Configuration Examples”).

Defining Entities and Generated Primary Keys

To define an entity in CloudTran, you must use CloudTran customizers on the class rather than the @Customizer annotations specified in the TopLink Grid. For example, this is at the start of the “Parent” entity class:

```
import com.cloudtran.coherence.intercept.CTCoherenceReadWriteCustomizer;

@Entity(name = "Parent")
@Table(name = "PARENT")
@Customizer(CTCoherenceReadWriteCustomizer.class)
public class Parent implements Serializable, PortableObject
```

The @Customizer line tells CloudTran to send this to the database via CloudTran’s Transaction Manager.

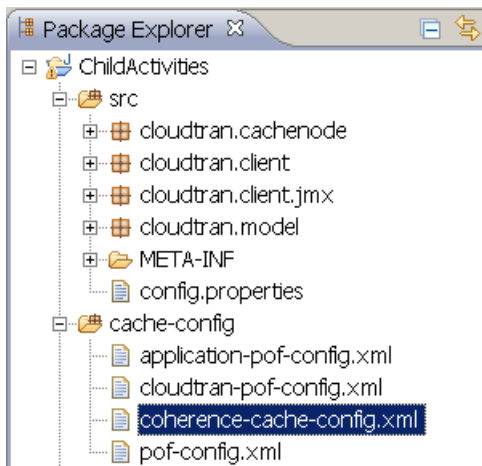
To get CloudTran to allocate primary keys for entities that are unique across the grid, you add a specialized generator annotation to the Id field:

```
@Id
@GeneratedValue(generator = "$CT.Parent")
private Long id;
```

The generator must begin with “\$CT.” and followed by a sequence name that defines the “namespace” for Id allocations; this name is normally the entity name like “Parent”, except in certain inheritance situations where more than one entity uses the allocation sequence.

Cache Configuration

In addition to the persistence.xml file you would expect to find in a JPA application, we need to define the Coherence caches for the domain objects; each object type needs its own cache. This definition is in the coherence-cache-config.xml file within the cache-config directory:



The `cache-config` directory also holds the pof config files: `pof-config.xml` and `application-pof-config.xml`, to efficiently serialise the CloudTran internal objects. This makes a dramatic difference to the “on-the-wire” efficiency of the application.

All entities are mapped to Coherence `NamedCache`'s, named for the entities; these caches are configured by CloudTran.

Running The Application

Note that these steps **must** be done in order shown.

1. Start MySQL via

```
<ctcoh>\MySQL\startMySQL.bat
```

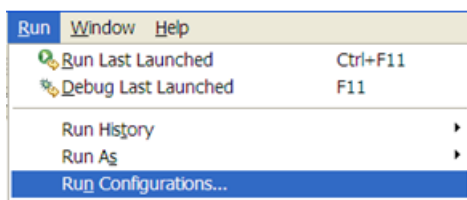
We use `username=cloudtran password=cloudtran`. This has admin privileges.

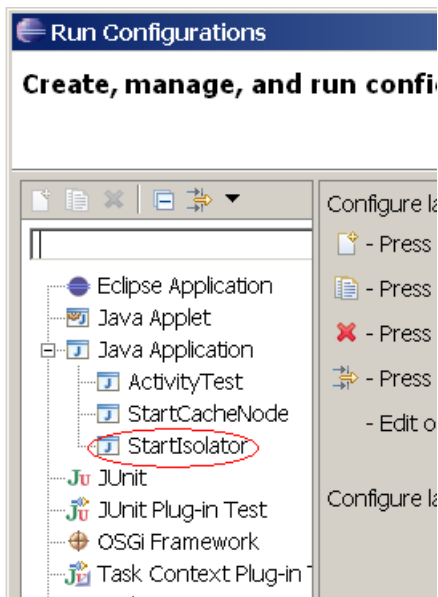
There is a `stopMySQL.bat` in the same directory.

The database is named `cloudtranTest`, for which the schema file is in

```
<ctcoh>\examples\ChildActivities\cloudtrantest.sql
```

2. Start the Isolator in Eclipse via Run Configurations.





Alternatively, there is a Windows batch job “startIsolator.bat”.

The isolator guarantees isolation of transactions committed across the grid en route to the database. The isolator is a singleton node: although it should be deployed with a backup node, while running only the primary is used. The isolator does not store information in a cache: its state is all in-memory and is reconstituted if there is a failover to the backup. For the demonstration described, you only need to start one isolator – but it doesn’t matter if you start more than one.

After starting the isolator, wait till you see these lines:

```
... PrimaryKeyGeneratorCacheService joined the cluster with senior
service member 1
CloudTran ISOLATOR NODE STARTED
```

3. Start a cache Node – via “StartCacheNode” in Eclipse, or via “startCacheNode.bat”. Note that this does take a little time – it is initialising the services, caches and quite a few threads.

Wait till you see these lines:

```
Started DefaultCacheServer...
CloudTran CACHE NODE STARTED on Member 2
```

4. Finally, start the Client – via the “ActivityTest” launch in Eclipse, or “startClient.bat”.

This creates a few thousand rows in the database and populates the grid, then runs a series of 10-row transaction tests. Each row consists of six reads, two updates, and one insert and one delete.

After the test has finished, the client console will show an output like

```
Number of threads: 5
Number of iterations: 1000
TotalTransactionCount 5000
Number of transaction failed: 0
Run finished in 39641ms
StartToCommitTime: 196625ms
StartToCommitTimeTxOnly: 115417ms
```

```
Tx/sec: 236.13204  
Average start and commit time      : 39ms  
Average start and commit time Tx Only: 23ms
```

As you can see this shows a transaction rate of around 236 transactions a second (this test was run on a Dell Latitude).