

# **Oracle® Insurance Policy Administration**

## **Cycle**

Version 9.4.1.0

Documentation Part Number: E23637\_01

October 2011

Copyright © 2009, 2011, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

#### **U.S. GOVERNMENT RIGHTS**

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Where an Oracle offering includes third party content or software, we may be required to include related notices. For information on third party notices and the software and related documentation in connection with which they need to be included, please contact the attorney from the Development and Strategic Initiatives Legal Group that supports the development team for the Oracle offering. Contact information can be found on the Attorney Contact Chart.

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your beta trial agreement only. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Software License and Service Agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

# Table of Contents

<b>INTRODUCTION.....</b>	<b>4</b>
<b>OVERVIEW.....</b>	<b>5</b>
CYCLE SYSTEM LEVELS .....	5
ARCHITECTURE.....	5
<i>Cycle Agents</i> .....	5
<i>Cycle Client</i> .....	6
DATABASE TABLES .....	7
CONTAINER DEPLOYMENT .....	8
CONFIGURATION FILES .....	8
<i>Scheduled Valuation</i> .....	10
<i>Coherence</i> .....	11
DATA SOURCES.....	15
DATABASE DRIVERS.....	15
<b>INSTALLATION .....</b>	<b>16</b>
AGENT.....	16
<i>Web Container</i> .....	16
CLIENT .....	18
<b>RUNNING CYCLE.....</b>	<b>19</b>
CLIENT .....	19
<i>Windows</i> .....	19
<i>Unix</i> .....	20
COMMANDS.....	21
<b>TROUBLESHOOTING.....</b>	<b>23</b>
CHECK LIST .....	23
COMMON ERRORS.....	23

## Introduction

Oracle Insurance Policy Administration (OIPA) provides a subsystem for batch processing of insurance transactions called Cycle. Cycle is a high-performance distributed subsystem designed to process as many pending transactions as possible in the shortest amount of time. By using concurrency techniques, multiple threads, automatic failover, automatic scaling and real time configuration changes, OIPA provides a robust Cycle solution. Various available commands allow you to run and view Cycle(s) according to your business needs.

The Cycle Agent is also used to process scheduled valuation work. Scheduled valuation calculates the value of a group of policies at a specific point in time and stores that value for subsequent use. The time interval for running the scheduled valuation and the frequency at which policies would be selected can be selected by the client. Typically, these intervals are quarterly, semi-annually or annually. In order for scheduled valuation to function, two additional options must be included in the Agent's configuration file.

In addition, Cycle is used to advance the system date stored in the `AsSystemDate` table.

## Overview

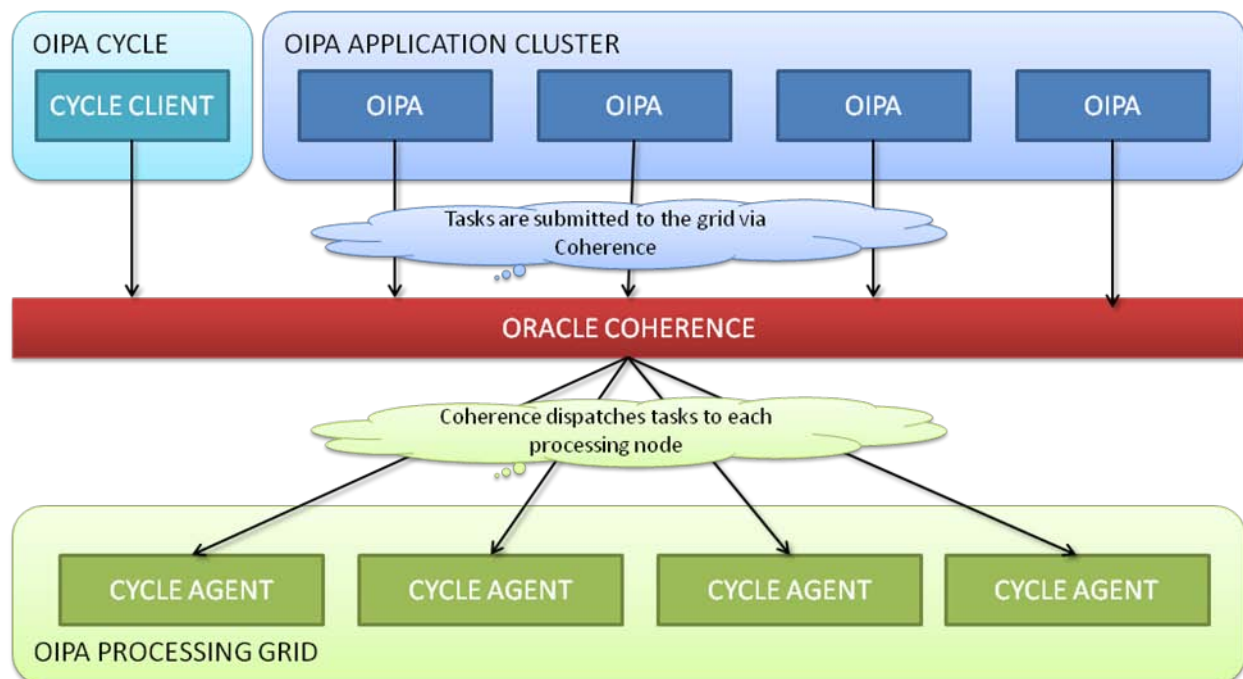
### Cycle System Levels

When using OIPA Cycle for pending activity processing, activities may be configured to process according to OIPA system levels. Applicable levels for processing batch activities are:

- Company
- Client
- Plan
- Policy

### Architecture

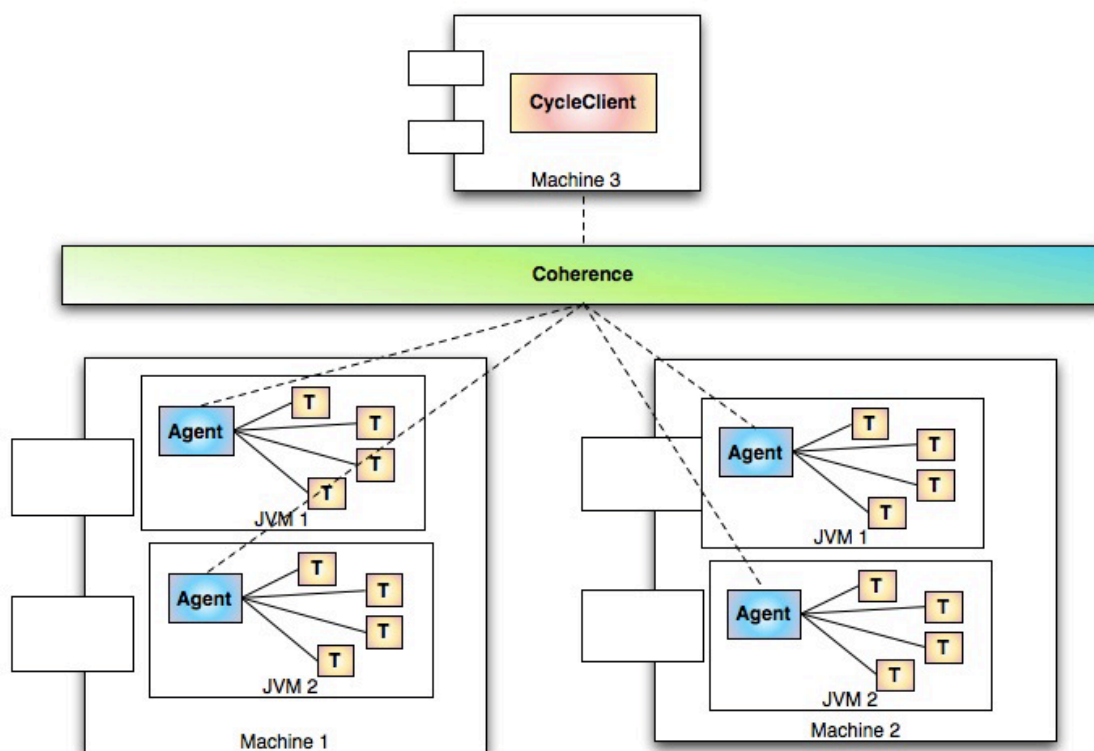
The Cycle subsystem drives processing of transactions through a Cycle Grid, which is comprised of a set of Cycle agents. Each Cycle Agent is a separate JVM instance, which acts as a processing node in the Grid. A special application called a Cycle client submits tasks to the Cycle Grid to direct what type of Cycle processing the group is to work on. The following is a high-level diagram showing how the different collaborators work together to start processing a Cycle run.



### Cycle Agents

Cycle Agents are the programs that execute the tasks that comprise the Cycle batch process. Each Cycle Agent runs in its own Java Virtual Machine (JVM). Depending on the number of tasks that must be executed during Cycle processing, any number of JVM's may be used to run the desired number of Cycle

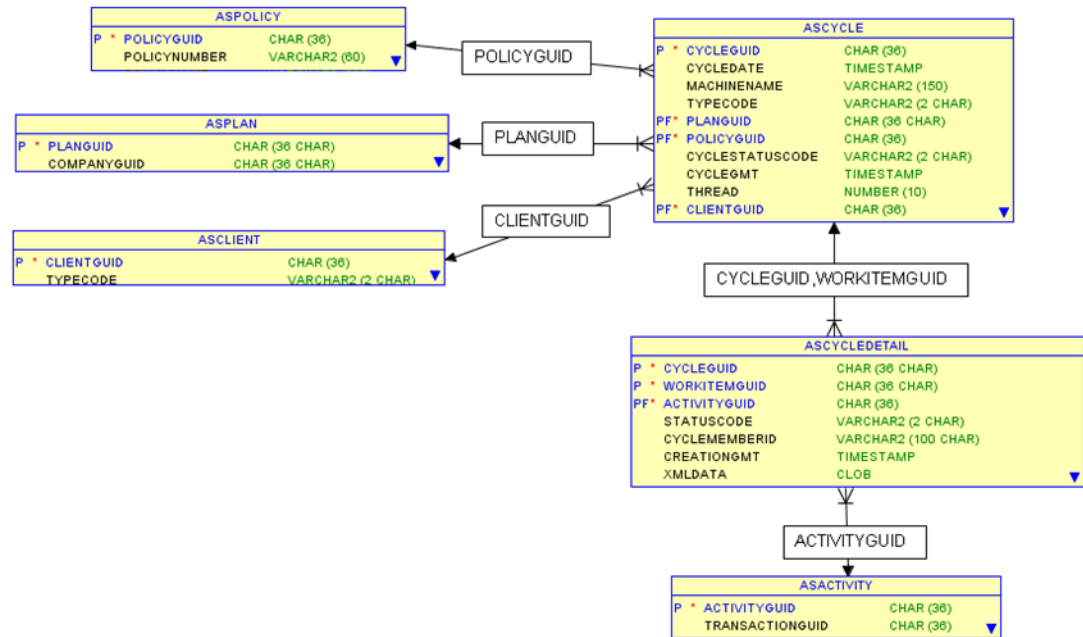
Agents. The JVM's may be started on one or more physical machines. If more than one machine is used, they will be effectively clustered via the Coherence clustered caching and messaging system. The following conceptual deployment diagram shows how the Cycle Agents may be distributed between multiple machines and JVM's. The boxes labeled "T" represent the threads controlled by the agents to perform the units of work.



## Cycle Client

Cycle Client is a standalone command line application used to control the Cycle process. It allows you to issue commands to all the Cycle Agents in order to start and stop the process

## Database Tables



## ***Container Deployment***

In a Container deployment, a Cycle agent operates inside of a web container, like WebLogic or WebSphere. The Cycle client still operates as a standalone application.

## ***Configuration Files***

**Cycle.PROPERTIES** – The first three properties to set in this file relate to Cycle and control the number of threads, the time intervals Cycle will check for work, and the maximum number of Cycle tasks that can be run simultaneously. The rest of the properties in this file should match what are in the OIPA properties file. There are descriptions of each property in the actual **Cycle.PROPERTIES** file.

**Cycle-coherence-config.xml** – Controls the clusters found in coherence. Do not alter this file unless you have an in-depth understanding of coherence. The information here is based on how the installation was done. The well known address is how you filter out what machines are allowed to join the Cycle cluster. You will receive an error if you try to access and you are not on the cluster.

**Cycle-coherence-cache-config.xml** – Default file defines how coherence caches are configured in terms of best practices.

**log4j.xml** – This file controls the level of detail that reports to the log files. Do not alter this file unless you have an in-depth understanding of this technology.

**client-appContext.xml** – Only used for Cycle Client.



## Cycle.properties

There are three properties at the top of the properties file that affect how Cycle processing will run. **cycle.period** is the number of seconds that the Cycle agent will wait before checking for additional work.

- Make it higher in order to avoid pinging the database too frequently for work. Frequent trips to the database, especially across multiple Cycle agents, can impact performance.
- Make it low enough so you do not starve the Cycle agent. If the threads in the Cycle agent run out of work, they will not do anything until the Cycle.period expires and work again is checked.

**cycle.batchSize** is the maximum number of Cycle tasks that will process in the Cycle Grid. Consider the following:

- The batchSize should minimally be the sum total of all of the threads available across all Cycle agents in the grid. For example, if there are four Cycle agents, each with 10 threads, the minimum batch size should be 40 threads.
- For performance reasons, you may want to make sure that the threads are always running. To ensure that the threads are always running, set the batchSize to 2x the number of threads in the Cycle Grid. For example, if there are four Cycle agents running 10 threads each, set the batchSize to 80.
- If you have a cycle.groupSize greater than 1, multiply the previous result by the groupSize to determine the batchSize. For example, if there are four Cycle agents running 10 threads each, and the groupSize is 10, set the batchSize to 800.
- The limitation on the number of tasks running in the cluster is dependent on the available memory in the cluster, not the thread pool sizes. The number of threads in the grid is only a theoretical estimate of how many tasks will be running at one time in the grid. Increasing the batchSize has the added benefit of reducing locking in the database when retrieving new tasks, which will increase performance. Setting the batchSize to 1000, 10000 or more is possible, depending on the memory in the cluster.

**cycle.groupSize** determines the number of cycle work items to group together and have execute on a single thread in the cluster.

- The groupSize has an impact on the number of tasks that are submitted to the grid for processing. If the batchSize is set to 10000, and the groupSize is set to 10, then 1000 actual tasks will be submitted to the grid for processing, each task containing 10 work items to be completed. Consider the groupSize value when determining the batchSize.
- The cycle.groupSize property is only required to be set on the cycle client.
- Increase this value in order to group together tasks and speed up task submission and processing. The grid will process 10000 work items faster if they are grouped in sets of 10, then it would processing 10000 individual work items separately.

**grid.taskSubmissionThreadPoolSize** is the number of threads dedicated to submit tasks to the grid for processing. Increasing this number can speed up how quickly tasks are distributed to the grid.

- Required to be set on every cycle agent.
- Increase this value in relation to the batchSize. The larger the batchSize, the more threads that should be dedicated to task submission.

## Scheduled Valuation

In order for the Cycle agent to be used in conjunction with Scheduled Valuation, two other options must be set in the agent's Cycle.properties file.

- **scheduledValuation.batchSize** – This is the batch size for processing scheduled valuation.
- **scheduledValuation.period** – This is the number of seconds that the scheduled valuation monitor task will sleep before waking up and checking on the status of queued tasks.

## Coherence

The Coherence cluster configuration, located in `Cycle-coherence-config.xml`, must be modified to include each Cycle client and agent that will participate in the Cycle group.

Since each Cycle member exists in its own JVM process and each member may be spread across multiple machines, the Cycle members in the same Cycle group must communicate. The messaging functionality of Coherence is used to accomplish this communication.

It is required that all Cycle members in the same Cycle group be configured in the same Coherence cluster.

The top of the file contains a section that allows for the addition of each machine that will participate in the Cycle group. Each Cycle client and agent that should participate in a single Cycle group must be contained in this section. Also, each OIPA instance should also be listed in this section, as OIPA has the ability to submit work to the Cycle group. In the case that a machine is not included in the Coherence configuration, the machine will not be able to join the Cycle group and will encounter issues starting up.

When configuring the well-known-address list, keep the following in mind:

- Every Cycle Client, Cycle Agent, and OIPA instance must be listed in the well known address list
- The well-known-address list must be **exactly the same** across all members (Cycle Clients, Cycle Agents, and OIPA Instances)
- If you do not have the same well-known-address list, some members may not start up properly or run at all.
- Incorrect configuration of the well-known-address list is the most common source of Cycle and OIPA problems. Ensure the list is configured correctly and shared by all members.
- If you use a member-identity section like the example below, make sure that the member-name is unique in the cluster, and that the cluster-name is the same for all members of the cluster. If the cluster-name is not the same, members may not be allowed to join the Coherence cluster, causing startup issues for the OIPA instance, Cycle Agent, or Cycle Client.

```
<coherence>
  <cluster-config>
    <unicast-listener>
      <well-known-addresses>
        <socket-address id="1">
          <address>localhost</address>
          <port>8088</port>
        </socket-address>
      </well-known-addresses>
    </unicast-listener>
  </cluster-config>
</coherence>
```

An example configuration follows:

```
<cluster-config>
  <member-identity>
```

```

        <cluster-name>PRODCLUSTER</cluster-name>
        <member-name>CYCLEAGENT1</member-name>
    </member-identity>
    <unicast-listener>
        <address>123.456.78.10</address>
        <port>42222</port>
        <port-auto-adjust>false</port-auto-adjust>
    <well-known-addresses>
        <socket-address id="1">
            <address>123.456.78.10</address>
            <port>42222</port>
        </socket-address>
        <socket-address id="2">
            <address>123.456.78.11</address>
            <port>42222</port>
        </socket-address>
        <socket-address id="3">
            <address>123.456.78.12</address>
            <port>42222</port>
        </socket-address>
    </well-known-addresses>
</unicast-listener>
</cluster-config>

```

The Coherence cache configuration, located in Cycle-coherence-cache-config.xml, configures the processing configuration on each Cycle Agent. This file must be modified only for each Cycle Agent, in order to uniquely identify the task processors that will be working in the Cycle Cluster. If the identifiers are not unique across all agents in the cluster, the agents with duplicate identifiers will not process any work.

The cache configuration only needs to be configured for Cycle Agents. It does not need to be modified for OIPA instances or Cycle Clients.

The following section highlights the most significant change for the cache config file. This is an example of the cache config file for a Grid Processing node (a Cycle Agent).

```

<cache-config
xmlns:processing="class:com.oracle.coherence.patterns.processing.configuration.ProcessingPatternNamespaceHandler">

    <processing:cluster-config pof="true">
        <processing:dispatchers>
            <processing:task-dispatcher displayname="Task Dispatcher" priority="1">
                <processing:composite-policy>
                    <processing:attribute-match-policy />
                    <processing:round-robin-policy />
                </processing:composite-policy>
            </processing:task-dispatcher>

```

```

</processing:dispatchers>

<!-- MAKE SURE THAT ALL IDs ARE UNIQUE ACROSS THE CLUSTER, OR THE MEMBER WILL
NOT PARTICIPATE IN GRID PROCESSING -->

<processing:taskprocessors>
  <processing:taskprocessordefinition id="RunnableTaskProcessor"
displayname="Runnable Task Processor"
    type="SINGLE" taskpattern="SingleTask">
      <processing:default-taskprocessor id="Runnable Task Processor"
        threadpoolsize="10">
      </processing:default-taskprocessor>
      <processing:attribute name="type">runnable</processing:attribute>
    </processing:taskprocessordefinition>
    <processing:taskprocessordefinition id="ResumableTaskProcessor"
displayname="Resumable Task Processor"
      type="SINGLE" taskpattern="SingleTask">
        <processing:default-taskprocessor id="Resumable Task Processor"
          threadpoolsize="5">
        </processing:default-taskprocessor>
        <processing:attribute name="type">resumable</processing:attribute>
      </processing:taskprocessordefinition>
    </processing:taskprocessors>
</processing:cluster-config>

```

**NOTE:** You cannot share a single coherence-cache-config file. Each Cycle Agent / Cycle Web JVM must have its own separate coherence-cache-config.xml file.

1. Make sure that each Cycle Agent / Cycle Web instance is using its own cycle-coherence-cache-config.xml file.
2. Make sure the ID attributes in the processing:taskprocessors xml section are unique across the entire cluster. In the example above, you must have a unique identifier for all items that are underlined and highlighted. There are six identifiers in total.

**NOTE:** If your identifiers are not unique across ALL Cycle Agents in the cluster, those agents with duplicate identifiers will not process work. There will be no obvious exception in this instance.

The following list mentions the changes in this file:

1. Remove the Logging Dispatcher. The logging dispatcher only logs task processing in the grid. It is valuable from a diagnostics standpoint, but does introduce overhead of logging into processing. You can remove the logging-dispatcher element from the configuration file if you do not require or want the additional logging.

2. Change the **threadpoolsize** attribute on the **processing:default-taskprocessor** element with the ID "RunnableTaskProcessor". This includes processing activities on a policy or a plan, or performing scheduled valuation on a single policy. Since **Runnable** tasks tend to be more processing intensive, it is recommended that more threads are given to this thread pool.
3. Change the **threadpoolsize** attribute on the **processing:default-taskprocessor** element with the ID "ResumableTaskProcessor". This thread pool executes **Resumeable** tasks, which are long running tasks in the grid that maintain intermediate state and report progress. Since **Resumeable** tasks tend to be less processing intensive, it is recommended that less threads are given to this thread pool.

## **Data Sources**

### **Database Drivers**

The necessary database drivers must be obtained and copied to the `lib` folder prior to starting Cycle.

#### **Oracle**

The drivers required for Oracle databases are packaged with the application and no intervention is required.

#### **IBM DB2**

The driver for IBM DB2 includes three jar files that must be obtained:

1. `db2jcc`
2. `db2jcc_license_cisuz`
3. `db2jcc_license_cu`

**Note:** `db2jcc`, `db2jcc_license_cisuz`, and `db2jcc_license_cu` are included with the purchase of the DB2 software. These files are not available for download. Contact your IT department if you need assistance locating these files.

**Note:** Make sure that the version of the DB2 drivers used is the correct version for your specific DB2 database.

#### **Microsoft SQLServer**

The JTDS driver for Microsoft SQL Server must be obtained.

Download the driver from the following site: <http://sourceforge.net/projects/jtds/>

1. Select the **Download jtds - SQL Server and Sybase JDBC driver** link.
2. Scroll down to the area that lists the previous File Releases and download the distribution file for version 1.2 (`jtds-1.2-dist.zip`).
3. Open the downloaded archive and extract it.

# Installation

## Agent

### Web Container

1. Upload all of the files to a location on your server's file system where they can be accessed from your application server.
2. Upload the jdbc driver for your database to a location on your server's file system where it can be accessed from your application server.
3. Modify any necessary configuration files as outlined in the Configuration section.
4. Add the location of the files you uploaded to the classpath of the application server so it can find the files. This is different depending on the application server you use.
  - For JBOSS, put them into the **conf** folder of the server that you are using.
  - For Weblogic 11g (10.3x), put them into the root folder of the domain (server) that you are using. For example, in Windows, this might be under:  
E:\Oracle\Middleware\user\_projects\domains\base\_domain
  - For Websphere, upload them onto the application server. Then go to your application server -> Java and Process Management -> Process Definition -> Java Virtual Machine, and you will be able to set the classpath there.
5. The JVM startup arguments must be modified in order to run Cycle. The changes are highlighted as follows:
  - The cache config file must be specified as a JVM argument. Previously, the cache configuration file was specified in the coherence config file. This is no longer the case, and must be specified as a JVM argument. The argument must contain the path to the coherence cache config file. The following is an example of the cache config argument.

```
-Dtangosol.coherence.cacheconfig=./conf/cycle-coherence-cache-  
config.xml
```

- The application Portable Object config file must be specified as a JVM argument. The Portable Object config file is packaged with the application, and describes the portable object setup. The argument should **not** include a path to the Portable Object config file, as it is packaged in an application library and available on the class path. The following is an example of the Portable Object config argument for a Cycle agent:

```
-Dtangosol.pof.config=com-adminserver-cycle-agent-pof-config.xml
```

- The complete JVM startup arguments for the container might look like:



```
-javaagent:/opt/Environments/spring-agent.jar -
Dtangosol.coherence.cacheconfig=/opt/Environments/TEST/cycle-
coherence-cache-config.xml -Dtangosol.pof.config=com-adminserver-
cycle-agent-pof-config.xml -
Dtangosol.coherence.override=/opt/Environments/TEST/cycle-
coherence-config.xml
```

- For JBOSS, set these in the /jboss/home/bin/run.bat or run.sh file.
- For Websphere, go to the application server -> Java and Process Management -> Process Definition -> Java Virtual Machine, and set this in Generic JVM Arguments.

6. Create data sources named **ADMINSERVERDS**, **ADMINSERVERRESOURCEDS**, **ADMINSERVERSEARCHDS**, and **ADMINSERVERREADONLYDS** on the application server. It is important that the data sources are named correctly as they are used by the application.

- **ADMINSERVERDS** – data source for OIPA database.
- **ADMINSERVERSEARCHDS** – data source for OIPA database.
- **ADMINSERVERRESOURCEDS** – data source for multi-language definitions. The OIPA database should be used by default unless multi-language definitions are being stored separately.
- **ADMINSERVERREADONLYDS** – Read-only access to the OIPA database.

## *Client*

1. Ensure that the `JAVA_HOME` environmental variable is set to a valid JDK 1.5 installation.  
Alternatively, you can modify the startup scripts to use a specific JVM.
  2. Extract the `Cycle.client` archive file into the desired location on your file system. The following directories are included:
    - bin** - contains the executables for the Cycle agent to run.
    - conf** - contains the configuration files for the Cycle agent (more details provided later).
    - lib** - contains the java libraries required for the Cycle agent.
  3. Modify the necessary configuration files as outlined in the configuration section. In most cases, the following files will need to be configured:
    - `Cycle.PROPERTIES`
    - `Cycle-coherence-config.xml`
  4. If you are using a database other than Oracle, the correct database drivers must be copied to the `lib` directory, as outlined in the data source section of this document.
- 
1. If you are using a database other than Oracle, the correct database drivers must be copied to the `lib` directory, as outlined in the data source section of this document.

# Running Cycle

## Client

## Windows

Running Cycle client allows you to set how you want a set of pending activities to process. You must use the `run.bat` file for a Microsoft® Windows environment.

1. Open a command line.
2. Navigate to the path where the application is installed and then to the Cycle.client folder. Type 'run'.
3. Type in the letter or number associated with the command you want to run. Explanations of the commands are below.
4. Refer to the log files to see messages and errors.

```

C:\WINDOWS\system32\cmd.exe - run
2009-06-09 11:01:35.116/17.315 Oracle Coherence 3.4.2/411 <Info> <thread=main, member=n/a>: Loaded o
perational overrides from file "C:\Development\test\cycle.client\conf\cycle-coherence-config.xml"
2009-06-09 11:01:35.116/17.315 Oracle Coherence 3.4.2/411 <D5> <thread=main, member=n/a>: Optional c
onfiguration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.4.2/411
Grid Edition: Development mode
Copyright <c> 2000-2009 Oracle. All rights reserved.

2009-06-09 11:01:35.647/17.846 Oracle Coherence GE 3.4.2/411 <Info> <thread=main, member=n/a>: Loade
d cache configuration from resource "file:/C:/Development/test/cycle.client/conf/cycle-coherence-cac
he-config.xml"
2009-06-09 11:01:37.049/19.248 Oracle Coherence GE 3.4.2/411 <D5> <thread=Cluster, member=n/a>: Serv
>: Asking member 1 for 128 out of 128 primary partitions
2009-06-09 11:01:38.721/20.920 Oracle Coherence GE 3.4.2/411 <D5> <thread=DistributedCache, member=2
>: Deferring the distribution due to 49 pending configuration updates
2009-06-09 11:01:39.292/21.491 Oracle Coherence GE 3.4.2/411 <D5> <thread=TcpRingListener, member=2>
: TcpRing: connecting to member 1 using TcpSocket<State=STATE_OPEN, Socket=Socket[addr=/10.1.100.98,
port=1685, localport=8089]>
2009-06-09 11:01:43.788/25.987 Oracle Coherence GE 3.4.2/411 <D5> <thread=Invocation:InvocationServi
ce, member=2>: Service InvocationService joined the cluster with senior service member 1

*****
OIPA Cycle Command Interface
Menu:
  P - Pause
  R - Resume
  D - Advance System Date
  A - Abort Current Cycle
  Q - Exit
  01 - Begin Pre-Company
  02 - Begin Pre-Plan
  03 - Begin Policy
  04 - Begin Post-Plan
  05 - Begin Post-Company
  06 - Begin Plan Status
  07 - Begin Pre-Client
  08 - Begin Post-Client
*****
ENTER COMMAND \>03
  
```

After running the menu options will appear

Enter the applicable command

## Unix

The shell script file, `run.sh`, is used to start the Cycle agent application is located in the `client/bin` subdirectory of the Cycle client installation directory. It may be run manually or scheduled using any standard scheduling utility.

1. Open a shell
2. Navigate to the path where the application is installed and then to the `Cycle.client` folder.
3. Type `./run.sh`.
4. Type in the letter or number associated with the command you want to run.
5. Refer to the log files to see messages and errors.

## Commands

- **P Paused** – Cycle will not process, although it will still check to see if activities should be processed. Then it sees that the status is still in pause and returns to sleep status.
- **R Resume** – Takes Cycle out of pause status.
- **D Advance System Date** - Advances date in the `AsSystemDate` table. Should be run after command(s) 01 thru 08 are run.
- **A Abort Current Cycle** – This should only be used when there is a serious error. This is a non recoverable scenario and a shutdown will occur. This will mark all records in the database as 99.
- **Q Exit** – Exits Cycle.
- **01 Pre-Company** – Runs only the pre-company portion of Cycle.
  - i. Cycle will attempt to process all company level activities with an effective date less than or equal to the current system date with one of the pending statuses and a processing order of one-thousand or less.
  - ii. A non-automatically overridden business error or a system error will cause Cycle to abort further processing at this level and move onto the next task.
- **02 Pre-Plan** – Runs only the pre-plan portion of Cycle.
  - i. Cycle will attempt to process all plan level activities with an effective date less than or equal to the current system date with one of the pending statuses and a processing order of one-thousand or less.
  - ii. Plans are executed in parallel using JMS threading.
  - iii. A non-automatically overridden business error or a system error will cause Cycle to move onto the next plan in the queue or the next processing level task if any.
- **03 Policy** – Runs the policy level portion of Cycle processing.
  - i. All policy level activities with an effective date less than or equal to the current system date will be processed in the following order: effective date (ascending), processing order (ascending), entry gmt (ascending).
  - ii. Policies are processed in parallel using JMS threading.
  - iii. A non-automatically overridden business error or a system error will cause Cycle to abort further processing on the policy and move onto the next policy or processing task (if any).

- **04 Post-Plan** – Runs only the post-plan portion of Cycle.
  - i. Cycle will attempt to process all plan level activities with an effective date less than or equal to the current system date with one of the pending statuses and a processing order greater than one-thousand.
  - ii. Plans are executed in parallel using JMS threading.
  - iii. A non-automatically overridden business error or a system error will cause Cycle to move onto the next plan in the queue or the next processing level task if any.
- **05 Post-Company** – Runs only the post-company portion of Cycle.
  - i. Cycle will attempt to process all company level activities with an effective date less than or equal to the current system date with one of the pending statuses and a processing order **greater** than one-thousand.
  - ii. A non-automatically overridden business error or a system error will cause Cycle to abort further processing at this level and move onto the next task.
- **06 Plan Status** – Only used for V7 Cycle.
- **07 Pre-Client** – Runs only the pre-client portion of Cycle.
  - i. Cycle will attempt to process all client level activities with an effective date less than or equal to the current system date with one of the pending statuses and a processing order of one-thousand or less.
  - ii. Clients are executed in parallel using JMS threading.
  - iii. A non-automatically overridden business error or a system error will cause Cycle to move onto the next client in the queue or the next processing level task if any.
- **08 Post-Client** – Runs only the post-client portion of Cycle.
  - i. Cycle will attempt to process all client level activities with an effective date less than or equal to the current system date with one of the pending statuses and a processing order greater than one-thousand.
  - ii. Clients are executed in parallel using JMS threading.
  - iii. A non-automatically overridden business error or a system error will cause Cycle to move onto the next client in the queue or the next processing level task if any.

# Troubleshooting

## Check List

1. Make sure that you change both the `jpa.databasePlatform` and `application.databaseType` properties in the `Cycle.properties` file to match **your** database. Often one or both of these can be overlooked, causing exceptions to be thrown.
2. Make sure to change the `datasource.type` property in the `Cycle.properties` file to `jndi` when running in a container.
3. Make sure `Cycle.properties` is on the classpath and loaded by the container. You may see a message like "cannot find bundle Cycle" if it cannot find the properties file.
4. Make sure that the `coherence-config.xml` files are on the classpath. It is possible that the `coherence-cache-config.xml` file that is referenced in the `coherence-config.xml` file has to be fully qualified. This is the case with WebSphere, where you need to put the full path to the `Cycle-coherence-cache-config.xml`.
5. Make sure that the well-known-address list in the `Cycle-coherence-config.xml` and the OIPA `coherence-config.xml` match *exactly*.
6. Make sure that the `Cycle-coherence-cache-config.xml` file is not shared by Cycle agents.
7. Make sure that the `Cycle-coherence-cache-config.xml` file has unique identifiers as mentioned in the Coherence configuration section of this document.

## Common Errors

### The Cycle agent doesn't do anything, even though I ran the Cycle Client.

The most common problem is that the Cycle agent is not sharing the same Coherence Cluster as the other members of the Cycle group, or the Cycle client. If the Cycle agent is not sharing the same cluster, then it will not be able to receiving messages, and will not do anything. There are a few reasons why the Cycle agent is not part of the same Coherence Cluster:

- Check the startup parameters for the application server to ensure that the system property is passed in, it is `-Dtangosol.coherence.override=` and should point to the `Cycle-coherence-config.xml` file. If the instance cannot find the `Cycle-coherence-config.xml` file, it may start its own, and will be outside of the shared cluster.
- Check the well-known-addresses section in the `Cycle-coherence-config.xml` file and make sure that the Cycle agent and/or Cycle client have exactly the same addresses listed. If you are denied access to the cluster, the instance may start its own, and therefore be outside of the shared cluster.
- Make sure the `Cycle-coherence-cache-config.xml` file is loaded. You can check this in the log file or the console output. Sometimes it will indicate that it could not be loaded, and load the default. If the default cache config file is loaded, it will not be in the shared coherence cluster. You may need to update the `Cycle-coherence-config.xml` file to include the fully qualified path to the `Cycle-coherence-cache-config.xml` file. This change needs to be made to run inside of WebSphere on Linux.