

# TopCoder Contribution Tracking Services Component - Dev Only

## Dev Only!

Note that this is a development only component, there are no design documents provided in this competition. The provided TCUML diagram only provides the services interface and entities models diagram with no implementation notes.

## 1. Overview

A few years ago, TopCoder made the great decision to open up the platform so that clients can interact directly with the PMs and copilots. They created an easy to use application (TopCoder Direct) which provides all the contest management functionality needed to develop almost any software project.

Using this and other tools which represent together the TopCoder software development platform, any client can leverage copilot's skills and build pretty much any kind of software they want. The next step in the evolution of the platform is to adapt in order to meet new market needs and to address opportunities.

One part of the adaptation process is the TopCoder Labs project which will make the platform and the community expertise accessible to a wider range of clients and by allowing different types of contributors to be part in the development process.

The idea behind TopCoder Labs project is to provide an alternative model for developing software through the Topcoder platform. Instead of the clients coming with a lump sum of money and provide it upfront to start the development process, in Topcoder Labs part of the development cost can be awarded in the form of shares from the final application generated revenue in a way similar to how a normal royalties system works.

After the successful completion of the project, Topcoder will be involved in the application's lifecycle to the degree necessary for tracking the generated revenue or any other forms of value created.

Out of the generated revenue, TopCoder will then pay members who worked on that application proportionally to their contribution

analogous to how TopCoder already does with royalties on generic components, with the difference that for TopCoder Labs project the revenue gathering process will be performed for the entire application.

In the process of managing the development process, tracking and revenue generation processes, Topcoder will receive a portion of the revenue sharing.

This system provided the functionalities of the Tracking part of the process, which provides the services required to manage the awarding of prizes and the tracking of contribution points during the development life of a project as well as updating the platform applications necessary to support the new processes like introducing special icon markers on the contests being part of a Labs project or by adding widgets that will explain to members what the conditions and gains are to be expected from participating in a given contest.

This is an initial module of the whole system, which only contains a small set of functionalities.

This component defines the models and backend services for the contribution tracking system.

## 1.1 Design Patterns

- **Strategy pattern** - service interfaces together with their implementation provided in this component can be possibly used in some external strategy context.
- **DAO/DTO pattern** - MembersContributionPointsService is a DAO for MemberContributionPoints DTO; ProjectContestCPConfigService is a DAO for ProjectContestCPConfig, DirectProjectCPConfigService is a DAO for DirectProjectCPConfig DTO.

## 1.2 Industry Standards

SQL, HQL, IoC, Java, JavaBean

## 1.3 Required Algorithms

### 1.3.1 Logging

This component must perform logging in all public methods of MembersContributionPointsServices, ProjectContestCPConfigService and DirectProjectCPConfigService. (except getters, setters methods).

All information must be logged using an (optional) Log instance field (you should define getter and setter for this field). If this getter returns null, then logging is not required to be performed - the getter should be injected via spring into the service implementation.

In all mentioned methods method entrance with input arguments, method exit with return value and call duration time must be logged at DEBUG level. It's not required to log method exit when method throws an exception.

All errors (for all thrown exceptions) must be logged at ERROR level with exception message and stack trace.

### 1.3.2 Hibernate entity mapping

You must define and provide the hibernate mapping file in your submission.

## 1.4 Component Class Overview

### **New classes!**

You are encouraged to add generic classes that provide common functionality (e.g. Logging or SessionFactory ..et) for the services of this component, note that any new class **MUST** have either private or protected visibility.

### **MemberContributionPoints [Serializable]**

This class is a container for information about member contribution points. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

### **DirectProjectCPConfig [Serializable]**

This class is a container for information about the contribution points associated with the given TopCoder Direct Project. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

### **ProjectContestCPConfig [Serializable]**

This class is a container for information about the contribution points associated with the given contest. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

### **MemberContributionPointsService [Interface]**

This interface represents a member contribution points DAO. It provides CRUD methods for manipulating member contribution points.

### **MemberContributionPointsServiceImpl [Not in TCUML]**

- This class is an implementation of *MemberContributionPointsService*.
- It should use Hibernate session to access entities in persistence.
- This class should use Logging Wrapper logger to log errors and debug information.
- Argument validation and initialization validation must take place in this implementation.
- The method names are self-explanatorily, this is a straightforward DAO implementation.

### **ProjectContestCPConfigService [Interface]**

This interface represents the DAO of the contribution points associated with the provided contest. It provides CRUD methods for manipulating contribution points that are associated with the provided contest.

### **ProjectContestCPConfigServiceImpl [Not in TCUML]**

- This class is an implementation of *ProjectContestCPConfigService*.
- It should use Hibernate session to access entities in persistence.
- This class should use Logging Wrapper logger to log errors and debug information.
- Argument validation and initialization validation must take place in this implementation.
- The method names are self-explanatorily, this is a straightforward DAO implementation.

### **DirectProjectCPConfigService [Interface]**

This interface represents the DAO of the contribution points associated with the provided contribution project. It provides CRUD methods for manipulating contribution points that are associated with the provided contribution project.

### **DirectProjectCPConfigServiceImpl [Not in TCUML]**

- This class is an implementation of *DirectProjectCPConfigService*.

- It should use Hibernate session to access entities in persistence.
- This class should use Logging Wrapper logger to log errors and debug information.
- Argument validation and initialization validation must take place in this implementation.
- The method names are self-explanatorily, this is a straightforward DAO implementation.

## 1.5 Component Exception Definitions

### **ContributionServiceException [The base class of other exceptions, Serializable, Not in TCUML]**

This exception is thrown by implementations of the services when some unexpected error occurred. Also this exception is used as a base class for other specific custom exceptions.

#### **Methods Signature!**

```
public ContributionServiceException(String message);
public ContributionServiceException(String message, Throwable cause);
public ContributionServiceException(String message, ExceptionData
data);
public ContributionServiceException(String message, Throwable cause,
ExceptionData data)
```

### **ContributionServiceInitializationException [Serializable, Not in TCUML]**

This exception is thrown by implementation of the services when the class was not initialized properly (e.g. while required property is not specified or property value has invalid format).

#### **Methods Signature!**

```
public ContributionServiceInitializationException(String message);
public ContributionServiceInitializationException(String message,
Throwable cause);
public ContributionServiceInitializationException(String message,
ExceptionData data);
public ContributionServiceInitializationException(String message,
Throwable cause, ExceptionData data)
```

### **ContributionServiceEntityNotFoundException [Serializable, Not in TCUML]**

This exception is thrown by implementations of the services when entity with the given ID does not exist in the persistence.

#### **Class Signature!**

```
private final String entityTypeNames;
```

```

private final long entityId;
public ContributionServiceEntityNotFoundException(String message,
String entityTypeName, long entityId);
public ContributionServiceEntityNotFoundException(String message,
Throwable cause, String entityTypeName, long entityId);
public ContributionServiceEntityNotFoundException(String message,
ExceptionData data, String entityTypeName, long entityId);
public ContributionServiceEntityNotFoundException(String message,
Throwable cause, ExceptionData data, String entityTypeName, long
entityId);
public String getEntityTypeName();
public long getEntityId();

```

## 1.6 Thread Safety

This component is thread safe (when used correctly).

## 1.7 Unit Testing

You must provide unit tests for all non-private classes in this component.

# 2.0 EnvironmentRequirements

## 2.1 Environment

- Development language: Java1.5+
- Compile target: Java1.5+
- QA Environment: Java 1.5+, JBoss 4.0.2, Informix 11.0

## 2.2 TopCoder Software Components

- **Logging Wrapper 2.0** - is used for logging errors and debug information.
  - **Base Exception 2.0** - is used by custom exception defined in this component.
  - **TopCoder Commons Utility 1.0** - defines ParameterCheckUtility, ValidationUtility and LoggingWrapperUtility used in this component.

*NOTE: The default location for TopCoder Software component jars is ../lib/tcs/COMPONENT\_NAME/COMPONENT\_VERSION relative to the component installation. Setting the tcs\_libdir property in topcoder\_global.properties will overwrite this default location.*

## 2.3 Third Party Components

- Spring 3.1 (<http://www.springsource.org/>)
  - Hibernate 3.6 (<http://www.hibernate.org/>)
- \_NOTE: The default location for 3rd party packages is ../lib relative to this component installation.  
Setting the ext\_libdir property in topcoder\_global.properties will overwrite this default location.\_


## 3. Installation and Configuration

### 3.1 Package Name

com.topcoder.utilities.cp.entities  
com.topcoder.utilities.cp.services  
com.topcoder.utilities.cp.services.impl

### 3.2 Configuration Parameters

The following configuration parameters are common parameters between the three services, it's assumed that all services implementations will be initialized via Spring IoC :

Property	Description	Values
loggerName	The name of the logger to be used for logging errors and debug information. If not specified, logging is not performed.	String. Not empty. Optional.
sessionFactory	The Hibernate session factory to be used when retrieving session for accessing entities stored in database.	 SessionFactory. Required.

### 3.3 DependenciesConfiguration

Please see docs of Logging Wrapper component to configure it properly.

## 4. UsageNotes

### 4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Setup Informix database "test":
  - a. Run `test_files/DBSetup.sql` to create the tables.

b. Update "dataSource" bean in test\_files/beans.xml and test\_files/beans\_invalid.xml if needed.

- Execute 'ant test' within the directory that the distribution was extracted to.

## 4.2 Required steps to use the component

See demo.

## 4.3 Demo

### 4.3.1 Sample Spring beans configuration

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xmlns:util="http://www.springframework.org/schema/util"

    xmlns:aop="http://www.springframework.org/schema/aop"

    xmlns:tx="http://www.springframework.org/schema/tx"

    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
        http://www.springframework.org/schema/util
        http://www.springframework.org/schema/util/spring-util-3.1.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-3.1.xsd">

    <bean id="dataSource"

        class="org.apache.commons.dbcp.BasicDataSource">

        <property name="driverClassName" value="com.informix.jdbc.IfxDriver" />

        <property name="url" value="jdbc:informix-sqli://localhost:1526/test:informixserver=ol_topcoder" />

        <property name="username" value="informix" />

        <property name="password" value="123456" />

    </bean>
```



```
<bean id="sessionFactory"

    class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

    <property name="dataSource" ref="dataSource"/>

    <property name="mappingResources">

    <list>

        <value>mapping/DirectProjectCPConfig.hbm.xml</value>

        <value>mapping/MemberContributionPoints.hbm.xml</value>

        <value>mapping/ProjectContestCPConfig.hbm.xml</value>

        <value>mapping/OriginalPayment.hbm.xml</value>

    </list>

    </property>

    <property name="hibernateProperties">

    <props>

        <prop key="hibernate.dialect">org.hibernate.dialect.InformixDialect</prop>

        <prop key="show_sql">true</prop>

    </props>

    </property>

</bean>


<bean id="directProjectCPConfigService"

    class="com.topcoder.utilities.cp.services.impl.DirectProjectCPConfigServiceImpl">

    <property name="sessionFactory" ref="sessionFactory"/>

    <property name="loggerName" value="com.topcoder.utilities.cp.services.imp.logger"/>

</bean>


<bean id="projectContestCPConfigService"

    class="com.topcoder.utilities.cp.services.impl.ProjectContestCPConfigServiceImpl">

    <property name="sessionFactory" ref="sessionFactory"/>

    <property name="loggerName" value="com.topcoder.utilities.cp.services.imp.logger"/>
```

```

</bean>

<bean id="memberContributionPointsService"

    class="com.topcoder.utilities.cp.services.impl.MemberContributionPointsServiceImpl">

    <property name="sessionFactory" ref="sessionFactory"/>

    <property name="loggerName" value="com.topcoder.utilities.cp.services.imp.logger"/>

</bean>

<!-- transaction management configuration -->

<!-- enable the configuration of transactional behavior based on annotations -->

<tx:annotation-driven transaction-manager="transactionManager"/>

<bean id="transactionManager"

    class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

    <property name="dataSource" ref="dataSource"/>

</bean>

</beans>

```

## 4.3.2 API usage

### 4.3.2.1 DirectProjectCPConfigService

```

// Load Spring XML file
ClassPathXmlApplicationContext applicationContext = new ClassPathXmlApplicationContext("beans.xml");

// Get DirectProjectCPConfigService from Spring context
DirectProjectCPConfigService directProjectCPConfigService =
    (DirectProjectCPConfigService) applicationContext.getBean("directProjectCPConfigService");

DirectProjectCPConfig directProjectCPConfig = new DirectProjectCPConfig();
directProjectCPConfig.setDirectProjectId(1);
directProjectCPConfig.setUseCP(true);
directProjectCPConfig.setAvailableImmediateBudget(1);

// Create the DirectProjectCPConfig entity
long directProjectCPConfigId = directProjectCPConfigService.create(directProjectCPConfig);

// Get the DirectProjectCPConfig entity
directProjectCPConfig = directProjectCPConfigService.get(directProjectCPConfigId);

// Update the DirectProjectCPConfig entity
directProjectCPConfig.setAvailableImmediateBudget(10);
directProjectCPConfigService.update(directProjectCPConfig);

// Get the available initial payments
double availableInitialPayments = directProjectCPConfigService

```

```

        .getAvailableInitialPayments(directProjectCPConfigId);

// Delete the DirectProjectCPConfig entity

directProjectCPConfigService.delete(directProjectCPConfigId);

```

#### 4.3.2.2 ProjectContestCPConfigService

```

// Load Spring XML file
ClassPathXmlApplicationContext applicationContext = new ClassPathXmlApplicationContext("beans.xml");

// Get ProjectContestCPConfigService from Spring context
ProjectContestCPConfigService projectContestCPConfigService =
    (ProjectContestCPConfigService) applicationContext.getBean("projectContestCPConfigService");

ProjectContestCPConfig projectContestCPConfig = new ProjectContestCPConfig();
projectContestCPConfig.setContestId(1);
projectContestCPConfig.setCpRate(2);

OriginalPayment originalPayment = new OriginalPayment();
originalPayment.setContestId(1);
originalPayment.setPrize1st(500);
originalPayment.setPrize2nd(250);
originalPayment.setPrize3rd(100);
originalPayment.setPrize4th(100);
originalPayment.setPrize5th(100);
originalPayment.setPrizeCopilot(150);
originalPayment.setPrizeMilestone(50);
originalPayment.setSpecReviewCost(50);
originalPayment.setReviewCost(200);
projectContestCPConfig.setOriginalPayment(originalPayment);

// Create the ProjectContestCPConfig entity
long projectContestCPConfigId = projectContestCPConfigService.create(projectContestCPConfig);

// Get the ProjectContestCPConfig entity
projectContestCPConfig = projectContestCPConfigService.get(projectContestCPConfigId);

// Update the ProjectContestCPConfig entity
projectContestCPConfig.setCpRate(3);
projectContestCPConfig.getOriginalPayment().setPrize1st(700);
projectContestCPConfig.getOriginalPayment().setPrize2nd(350);
projectContestCPConfigService.update(projectContestCPConfig);

// Delete the ProjectContestCPConfig entity

projectContestCPConfigService.delete(projectContestCPConfigId);

```

#### 4.3.2.3 MemberContributionPointsService

```

// Load Spring XML file
ClassPathXmlApplicationContext applicationContext = new ClassPathXmlApplicationContext("beans.xml");

// Get MemberContributionPointsService from Spring context
MemberContributionPointsService memberContributionPointsService =
    (MemberContributionPointsService) applicationContext.getBean("memberContributionPointsService");

MemberContributionPoints memberContributionPoints1 = new MemberContributionPoints();
memberContributionPoints1.setUserId(1);
memberContributionPoints1.setContestId(1);
memberContributionPoints1.setContributionPoints(10);
memberContributionPoints1.setContributionType("ct1");

// Create the MemberContributionPoints entity
long memberContributionPointsId1 =
memberContributionPointsService.create(memberContributionPoints1);

MemberContributionPoints memberContributionPoints2 = new MemberContributionPoints();
memberContributionPoints2.setUserId(1);
memberContributionPoints2.setContestId(2);
memberContributionPoints2.setContributionPoints(20);
memberContributionPoints2.setContributionType("ct1");

```

```

// Create the MemberContributionPoints entity
long memberContributionPointsId2 =
memberContributionPointsService.create(memberContributionPoints2);

MemberContributionPoints memberContributionPoints3 = new MemberContributionPoints();
memberContributionPoints3.setUserId(2);
memberContributionPoints3.setContestId(1);
memberContributionPoints3.setContributionPoints(30);
memberContributionPoints3.setContributionType("ct1");

// Create the MemberContributionPoints entity
long memberContributionPointsId3 =
memberContributionPointsService.create(memberContributionPoints3);

// Get the MemberContributionPoints entities by user and contest
List<MemberContributionPoints> memberContributionPointsList1 = memberContributionPointsService
    .getByUserAndContest(1, 1);

// Get the MemberContributionPoints entities by user
List<MemberContributionPoints> memberContributionPointsList2 =
memberContributionPointsService.getByUserId(1);

// Get the MemberContributionPoints entities by direct project
List<MemberContributionPoints> memberContributionPointsList3 = memberContributionPointsService
    .getByProjectId(1);

// Get the MemberContributionPoints entities by contest
List<MemberContributionPoints> memberContributionPointsList4 = memberContributionPointsService
    .getByContestId(1);

memberContributionPoints1 = memberContributionPointsList1.get(0);

// Update the MemberContributionPoints entity
memberContributionPoints1.setContributionPoints(15);
memberContributionPointsService.update(memberContributionPoints1);

// Delete the MemberContributionPoints entity

memberContributionPointsService.delete(memberContributionPointsId1);

```

## 5. Future Enhancements

None.

## 6. Provided Documentation

 <a href="#">Name</a>	<a href="#">Size</a>	<a href="#">Creator</a>	<a href="#">Date</a>	<a href="#">Comments</a>
 <a href="#">ERD.mwb</a>	7 kb	elkhawajah	Dec 06, 2011	ERD Diagram
 <a href="#">TC_Contribution_Tracking_Initial.tcuml</a>	19 kb	elkhawajah	Dec 06, 2011	Architectural TCUML Tables so
 <a href="#">tables.sql</a>	2 kb	elkhawajah	Dec 06, 2011	from ERD Diagram
 <a href="#">Application_Design_Specification.doc</a>	172 kb	elkhawajah	Dec 05, 2011	Application Design Specification

## 7. Support

### **Standard 30 days support.**

After this competition finishes we will be running small assembly competitions to integrate the component into Direct and into web\_module.

**The winner must monitor the forums for those assemblies and answer any questions related to the usage of the component.**