# Contest Service Facade 1.1 Component Specification

Changes to 1.0 version will be coded in **blue** font and new additions in version 1.1 will be coded using the **red** font.

## 1. Design

This component provides the façade for the users to manage the contest, submission, and process the payment. ContestServiceFacade is the main classes of this component, its implementation will act as a façade and delegate the call to the studio service component to handle the user invocation. For the payment process, currently, the design provides the paypal payment implementation

In the version 1.1, the ContestServiceFacade have four more methods added:
setSubmissionMilestonePrize(submissionId:long,milestonePrizeId:long):void

getUserContests(userName:String):List<StudioCompetition>

getMilestoneSubmissionsForContest(contestId:long):List<SubmissionData>

getFinalSubmissionsForContest(contestId:long):List<SubmissionData>

### 1.1 Design Patterns

**Strategy Pattern** – ContestServiceFacade and its implementation implement the strategy pattern.
**Facade Pattern –** ContestServiceFacade implement the façade pattern for the user service, catalog service, studio service, etc.
**DTO Pattern –** The entities in this component implement the DTO pattern.
**DAO Pattern** – ContestServiceFacade and its implementation also implement the DAO pattern to manage the entities.

### 1.2 Industry Standards

JPA
EJB 3.0
Web Service

### 1.3 Required Algorithms

#### *1.3.1 Logging in ContestManagerBean*

Logging should be done in all public methods of ContestManagerBean using the JBOSS logger. The method entry/exit (including the arguments and return value) should be logged, the exception error message should be logged, the time should also be logged. PayflowProPaymentProcessor will also perform the logging in the public methods.

### 1.4 Component Class Overview

#### *1.4.1 Package com.topcoder.service.facade.contest*

**CommonProjectPermissionData**

Bean class to hold permissions for project and their contests.

Annotations:

@XmlAccessorType(XmlAccessType.FIELD)

@XmlType(name = "commonProjectPermissionData", propOrder = { "contestId",

@XmlType(name = "commonProjectPermissionData", propOrder = { "contestId",

ContestPaymentResult

This class contains payment result and contest data. Its instances are created in reply to processing contest payment.

Annotations:

@XmlAccessorType(XmlAccessType.FIELD)

@XmlType(name = "contestPaymentResult", propOrder = { "paymentResult", "contestData" })

@XmlType(name = "contestPaymentResult", propOrder = { "paymentResult", "contestData" })

**SoftwareContestPaymentResult**

This class contains payment result and software competition data. Its

instances are created in reply to processing contest payment.

Annotations:

@XmlAccessorType(XmlAccessType.FIELD)

@XmlType(name = "softwareContestPaymentResult", propOrder =
{ "paymentResult",

@XmlType(name = "softwareContestPaymentResult", propOrder =
{ "paymentResult",

**ContestServiceFilter**

A custom filter used as a wrapper aroung the actual filter which is delegated to process the calls to the methods.

**ContestServiceFacade**

An interface for the web service implementing unified Facade interface to various service components

provided by global TopCoder application.As of this version a facade to Studio Service is provided only.

Module Cockpit Contest Service Enhancement Assembly change: Several new methods related to the permission

and permission type are added.

Module Cockpit Share Submission Integration Assembly change: Added method to retrieve all permissions by projectId.

Annotations:

@WebService(name = "ContestServiceFacade")

----------changed in the version 1.1----------

Four methods are added

setSubmissionMilestonePrize(submissionId:long,milestonePrizeId:long):void

getUserContests(userName:String):List<StudioCompetition>

getMilestoneSubmissionsForContest(contestId:long):List<SubmissionData>

getFinalSubmissionsForContest(contestId:long):List<SubmissionData>

---

Thread safety requirement is not changed.

**CommonProjectContestData**

Represents the common entity class for contest info for SimpleProjectContestData.

Thread Safety: This entity is not thread safe since it is mutable.

### *1.4.2 Package com.topcoder.service.facade.contest.ejb*

**ContestServiceFacadeBean**

This is an implementation of Contest Service Facade web service in form of stateless session EJB. It

holds a reference to {@link StudioService} which is delegated the fulfillment of requests.

Module Cockpit Contest Service Enhancement Assembly change: Several new methods related to the permission

and permission type are added.

Module Cockpit Share Submission Integration Assembly change: Added method to retrieve all permissions by projectId.

Annotations:

@Stateless

@WebService

@EndpointConfig(configName = "Standard WSSecurity Endpoint")

@DeclareRoles({"Cockpit User" })

@RolesAllowed({"Cockpit User" })

@TransactionManagement(TransactionManagementType.CONTAINER)

@TransactionAttribute(TransactionAttributeType.REQUIRED)

----------changed in the version 1.1----------

Four methods are added

setSubmissionMilestonePrize(submissionId:long,milestonePrizeId:long):void

getUserContests(userName:String):List<StudioCompetition>

getMilestoneSubmissionsForContest(contestId:long):List<SubmissionData>

getFinalSubmissionsForContest(contestId:long):List<SubmissionData>

---

thread safety requirement is not changed.

**ContestServiceFacadeLocal**

An interface providing local access to {@link ContestServiceFacade} implementation available within the same

application instance or EJB container.Thread safety: The implementations of this interface must operate in a thread-safe manner to be used inside

the EJB container.

Annotations:

@Local

**ContestServiceFacadeRemote**

An interface providing remote access to {@link ContestServiceFacade} implementation available within the remote

application instance or outside of the EJB container.Thread safety: The implementations of this interface must operate in a thread-safe manner to be used inside

the EJB container.

Annotations:

@Remote

### *1.4.3 Package com.topcoder.service.payment*

## CreditCardPaymentData

This is a bean class to hold the credit card details like number, expiry date, payer details etc. This class simply

captures the information from client. Client would have done the validation of various fields. It doesn't perform any

validation over that.

Annotations:

@XmlAccessorType(XmlAccessType.FIELD)

@XmlType(name = "creditCardPaymentData", propOrder = { "cardNumber", "cardType", "cardExpiryMonth", "cardExpiryYear",

@XmlType(name = "creditCardPaymentData", propOrder = { "cardNumber", "cardType", "cardExpiryMonth", "cardExpiryYear",

TCPurhcaseOrderPaymentData

This class captures the Purchase Order payment data. Purchase order in the current implementation is recognized by

po-number only.

Updated for: Cockpit Release Assembly for Receipts

Added 4 new properties to hold client and project details for the given po.

Annotations:

@XmlAccessorType(XmlAccessType.FIELD)

@XmlType(name = "tcPurhcaseOrderPaymentData", propOrder = { "poNumber", "clientId", "clientName", "projectId", "projectName"})

@XmlType(name = "tcPurhcaseOrderPaymentData", propOrder = { "poNumber", "clientId", "clientName", "projectId", "projectName"})

**PaymentResult**

This class captures the payment result after successful processing of the payment data. The current implementation

just captures the payment reference number as the result.

Annotations:

@XmlAccessorType(XmlAccessType.FIELD)

@XmlType(name = "paymentResult", propOrder = { "referenceNumber" })

@XmlType(name = "paymentResult", propOrder = { "referenceNumber" })

**PaymentProcessor**

This interface defines the contract for processing a payment request. Payment can be processed either through PayPal,

Purchase order or other mechanism. It solely depends on the implementing class.

**PaymentType**

An ENUM which captures different type of payments to be processed. Current implementation just supports two payment types viz: a) PayPal Credit Card. b) TC Purchase order.

**PaymentData**

This is base class for capturing the payment data.

Annotations:

@XmlSeeAlso( { CreditCardPaymentData.class,
TCPurhcaseOrderPaymentData.class })

@XmlSeeAlso( { CreditCardPaymentData.class,
TCPurhcaseOrderPaymentData.class })

*1.4.4 Package com.topcoder.service.payment.paypal*

**PayPalPaymentProcessor**

This implementation class processes the payment (PaymentData) through PayPal SOAP APIs

**PayflowProPaymentProcessor**

This class uses PayFlow payment system to process Paypal payments.

### 1.5 Component Exception Definitions

*1.5.1 System Exceptions*

- **IllegalArgumentException**: It is thrown when the passed-in argument is illegal. NOTE: A string is empty if its length is 0 after being trimmed.
- **IllegalStateException**: It is thrown if configuration errors occur

*1.5.2 Custom Exceptions*

- **PaymentException**: This exception will be thrown if any error occurs during the payment process.
- **ContestServiceException**: This exception will be thrown by StudioFacade and its implementation.

### 1.6 Thread Safety

This component will be deployed as EJB or web service. The interface ContestServiceFacade requires thread safety. The ContestServiceFacadeBean is a stateless bean, it will be initialized only once in the initialize method immediately after it is created, and their values will never be changed afterwards, so they won't affect the thread-safety of this bean.

The interface PaymentProcessor requires thread safety. Its two implementations are immutable once created, and they are thread safe.

The entities in this component are mutable and not thread safe. But they will not be modified during the operations of ContestServiceFacade, hence, they do not affect the thread safety of this component.

## 2. Environment Requirements

### 2.1 Environment

- Java 5.0+
- JBoss 4.2
- Hibernate 3.2 and higher

### 2.2 TopCoder Software Components

- **Studio Service 1.3**: This components mainly wraps the studio service component.
- **Contest and Submission Entities 1.2**: It provides the data entities used in this design.

- **Project Service 1.0.1**: The used Project data entity is from this component.
- **Search Builder 1.3.2**: This is used for the Filter definition which we utilize in the component.
- **Configuration Persistence 1.0.1** This is used to load the configuration values in the stateless bean
- **Document Generator 2.0** This is used to generate the content from the templatet in the stateless bean
- **Configuration API 1.0** This is used to load the configuration object
- **Email Engine 3.0** Used to send email
- **TopCoder Service Integration v2 assembly lib** The services lib is from this assembly
- **JBoss Login Module 2.0** Used to handle the use login
- **Base Exception 2.0** not used
- **Logging Wrapper 2.0** not used

### 2.3 Third Party Components

Paypal payment process API.

## 3. Installation and Configuration

### 3.1 Package Name

com.topcoder.service.facade.contest
com.topcoder.service.facade.contest.ejb
com.topcoder.service.payment

### 3.2 Configuration Parameters

Configuration for StudioFacadeBean

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/ejb-jar_3_0.xsd"
        version="3.0">
    <description>Contest Services Facade EJB</description>
    <display-name>Contest Services Facade EJB</display-name>
    <enterprise-beans>
        <session>
            <ejb-name>ContestServiceFacadeBean</ejb-name>
            <remote>com.topcoder.service.facade.contest.ejb.ContestServiceFacadeRemote</remote>
            <local>com.topcoder.service.facade.contest.ejb.ContestServiceFacadeLocal</local>
            <service-endpoint>com.topcoder.service.facade.contest.ContestServiceFacade</service-endpoint>
            <ejb-class>com.topcoder.service.facade.contest.ejb.ContestServiceFacadeBean</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
            <env-entry>
```

```
    <env-entry-name>logName</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>contest_service_facade_log</env-entry-value>
</env-entry>
<env-entry>
    <env-entry-name>payPalApiUserName</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>@build-rep.contest_service_facade.payPalApiUserName@</env-entry-value>
</env-entry>
<env-entry>
    <env-entry-name>payPalApiPassword</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>@build-rep.contest_service_facade.payPalApiPassword@</env-entry-value>
</env-entry>
<env-entry>
    <env-entry-name>payPalApiSignature</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>@build-rep.contest_service_facade.payPalApiSignature@</env-entry-value>
</env-entry>
<env-entry>
    <env-entry-name>payPalApiEnvironment</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>@build-rep.contest_service_facade.payPalApiEnvironment@</env-entry-value>
</env-entry>

<!--  BUGR-1239 -->
<env-entry>
    <env-entry-name>payFlowHostAddress</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>@build-rep.contest_service_facade.payFlowHostAddress@</env-entry-value>
</env-entry>

<env-entry>
    <env-entry-name>payFlowUser</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>@build-rep.contest_service_facade.payFlowUser@</env-entry-value>
</env-entry>
<env-entry>
    <env-entry-name>payFlowPartner</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>@build-rep.contest_service_facade.payFlowPartner@</env-entry-value>
</env-entry>
<env-entry>
```

```
    <env-entry-name>payFlowVendor</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>@build-rep.contest_service_facade.payFlowVendor@</env-entry-value>
</env-entry>
<env-entry>
    <env-entry-name>payFlowPassword</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>@build-rep.contest_service_facade.payFlowPassword@</env-entry-value>
</env-entry>
                        <env-entry>
    <env-entry-name>forumBeanProviderUrl</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>@build-rep.contest_service_facade.forumBeanProviderUrl@</env-entry-value>
</env-entry>
<env-entry>
    <env-entry-name>createForum</env-entry-name>
    <env-entry-type>java.lang.Boolean</env-entry-type>
    <env-entry-value>@build-rep.contest_service_facade.createForum@</env-entry-value>
</env-entry>

                        <!-- Cockpit Release Assembly for Receipts -->
                        <env-entry>
    <env-entry-name>documentManagerConfigFile</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>DocumentManager.xml</env-entry-value>
</env-entry>

<!-- Cockpit Release Assembly for Receipts -->
                        <env-entry>
    <env-entry-name>activateContestReceiptEmailTemplatePath</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>email_templates/activate_contest_receipt.txt</env-entry-value>
</env-entry>

                        <!-- Cockpit Release Assembly for Receipts -->
                        <env-entry>
    <env-entry-name>activateContestReceiptEmailFromAddr</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>directassist@topcoder.com</env-entry-value>
</env-entry>

<!-- Cockpit Release Assembly for Receipts -->
                        <env-entry>
```

```
    <env-entry-name>activateContestReceiptEmailBCCAddr</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>directassist@topcoder.com</env-entry-value>
</env-entry>

<!-- Cockpit Release Assembly for Receipts -->
                        <env-entry>
    <env-entry-name>activateContestReceiptEmailSubject</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>Your TopCoder Direct Order (%ORDER_NUMBER%)</env-entry-value>
</env-entry>

                        <!-- Cockpit Release Assembly for Receipts -->
                        <env-entry>
    <env-entry-name>purchaseSubmissionReceiptEmailTemplatePath</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>email_templates/purchase_submission_receipt.txt</env-entry-value>
</env-entry>

                        <!-- Cockpit Release Assembly for Receipts -->
                        <env-entry>
    <env-entry-name>purchaseSubmissionReceiptEmailFromAddr</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>directassist@topcoder.com</env-entry-value>
</env-entry>

<!-- Cockpit Release Assembly for Receipts -->
                        <env-entry>
    <env-entry-name>purchaseSubmissionReceiptEmailBCCAddr</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>directassist@topcoder.com</env-entry-value>
</env-entry>

<!-- Cockpit Release Assembly for Receipts -->
                        <env-entry>
    <env-entry-name>purchaseSubmissionReceiptEmailSubject</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>Your TopCoder Direct Order (%ORDER_NUMBER%)</env-entry-value>
</env-entry>

<ejb-ref>
    <ejb-ref-name>ejb/StudioService</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
```

```
            <remote>com.topcoder.service.studio.StudioService</remote>
        </ejb-ref>
                                <ejb-ref>
            <ejb-ref-name>ejb/PermissionService</ejb-ref-name>
            <ejb-ref-type>Session</ejb-ref-type>
            <remote>com.topcoder.service.permission.PermissionService</remote>
        </ejb-ref>
                                <ejb-ref>
            <ejb-ref-name>ejb/UserService</ejb-ref-name>
            <ejb-ref-type>Session</ejb-ref-type>
            <remote>com.topcoder.service.user.UserService</remote>
        </ejb-ref>
      </session>
   </enterprise-beans>
</ejb-jar>
```

### 3.3 Dependencies Configuration

The dependent data entities should be deployed properly.

## 4. Usage Notes

### 4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Execute 'ant test' within the directory that the distribution was extracted to.

### 4.2 Required steps to use the component

Deploy this component in an EJB 3.0 container, and then follow the demo to call the beans. This component will deployed in the jboss server, the jboss xml file is:

```
<?xml version="1.0"?>
<jboss>
    <enterprise-beans>
        <session>
            <ejb-name>ContestServiceFacadeBean</ejb-name>
            <jndi-name>remote/ContestServiceFacadeBean</jndi-name>


            <ejb-ref>
                <ejb-ref-name>ejb/StudioService</ejb-ref-name>
                <jndi-name>StudioServiceBean/remote</jndi-name>
            </ejb-ref>
            <ejb-ref>
                <ejb-ref-name>ejb/CatalogService</ejb-ref-name>
                <jndi-name>CatalogService/remote</jndi-name>
            </ejb-ref>
            <ejb-ref>
                <ejb-ref-name>ejb/ProjectServicesBean</ejb-ref-name>
                <jndi-name>ProjectServicesBean/remote</jndi-name>
            </ejb-ref>
            <ejb-ref>
                <ejb-ref-name>ejb/PermissionService</ejb-ref-name>
                <jndi-name>PermissionServiceBean/remote</jndi-name>
```

```
            </ejb-ref>

                    <ejb-ref>
                <ejb-ref-name>ejb/UserService</ejb-ref-name>
                <jndi-name>UserServiceBean/remote</jndi-name>
            </ejb-ref>

                    <port-component>
                            <port-component-
name>ContestServiceFacadeBean</port-component-name>
                            <port-component-
uri>/ContestServiceFacadeBean</port-component-uri>
                            <transport-guarantee>CONFIDENTIAL</transport-
guarantee>
                    </port-component>
        </session>
    </enterprise-beans>

    <security-domain>java:/jaas/JBossLoginModule</security-domain>
       <webservices>
              <context-root>/contestfacade</context-root>
              <webservice-description>
              <webservice-description-name />
              <wsdl-publish-location />
    </webservice-description>
 </webservices>
</jboss>
```

## 4.3 Demo

```
// the demo show the bean part, the web service usage is similar.
// lookup the bean from JNDI
ContestServiceFacade contestServiceFacadeBean = (ContestServiceFacade)
        new InitialContext().lookup("contestServiceBean/remote");

// assume that the submission id and milestone prize id has been created in the
persistence.
// set submission milestone prize
contestServiceFacadeBean.setSubmissionMilestonePrize(1, 1);

// get user contests
List<StudioCompetition> contestsForUser =
contestServiceFacadeBean.getUserContests("tom");

// get milestone submissions for contest
List<SubmissionData> milestoneSubmissions =
contestServiceFacadeBean.getMilestoneSubmissionsForContest(1);

// get final submissions for contest
List<SubmissionData> finalSubmissions =
contestServiceFacadeBean.getFinalSubmissionsForContest(1);

// check the final submission
for (SubmissionData data : finalSubmissions) {
    // do some action here
}

// the usage of other methods are similar
// create contest
```

```
StudioCompetition studioCompetition =
contestServiceFacadeBean.createContest(contest, tcDirectProjectId);

// get contests for project
List<StudioCompetition> contestsForProject =
contestServiceFacadeBean.getContestsForProject(tcDirectProjectId);

// update contest, assume that the contest was created somewhere
contestServiceFacadeBean.updateContest(contest);

// update contest status
contestServiceFacadeBean.updateContestStatus(contestId, newStatusId);

// add document to contest
contestServiceFacadeBean.addDocumentToContest(documentId, contestId);

// remove document from contest
contestServiceFacadeBean.removeDocumentFromContest(document);

// get status list
List<ContestStatusData> csd = contestServiceFacadeBean.getStatusList();

// get simple contest data
List<SimpleContestData> scd = contestServiceFacadeBean.getSimpleContestData();

// get simple contest data bypid
List<SimpleContestData> csdByPid =
contestServiceFacadeBean.getSimpleContestDataByPID(pid);

// search contests
List<StudioCompetition> comps = contestServiceFacadeBean.searchContests(filter);

// get all contest types
List<ContestTypeData> contestTypeDatas =
contestServiceFacadeBean.getAllContestTypes();

// get all mediums
List<MediumData> mediumDatas = contestServiceFacadeBean.getAllMediums();

// add change history
contestServiceFacadeBean.addChangeHistory(history);
// rank submissions
boolean flag = contestServiceFacadeBean.rankSubmissions(submissionIdsInRankOrder);

// process contest credit card payment
ContestPaymentResult contestPaymentResult =
contestServiceFacadeBean.processContestCreditCardPayment(competition,  paymentData);
// check the result here

// process contest purchase order payment
contestPaymentResult =
contestServiceFacadeBean.processContestPurchaseOrderPayment(competition,
paymentData);

// process contest credit card sale
SoftwareContestPaymentResult softwareContestPaymentResult =
contestServiceFacadeBean.processContestCreditCardSale(competition, paymentData);
```

```
// process contest purchase order sale
softwareContestPaymentResult =
contestServiceFacadeBean.processContestPurchaseOrderSale(competition,  paymentData);

// process submission credit card payment
PaymentResult paymentResult =
contestServiceFacadeBean.processSubmissionCreditCardPayment(completedContestData,
paymentData);

// process submission purchase order payment
paymentResult =
contestServiceFacadeBean.processSubmissionPurchaseOrderPayment(completedContestDat
a, paymentData);
// check the result here
```

## 5 Future Enhancements

None.