Project Payment Management 1.0 Component Specification

1. Design

The Project Payment Management component will define new entities for project payments and provide the managers.

This component provides ProjectPaymentManager and ProjectPaymentAdjustmentManager interfaces together with their implementations that use pluggable persistence instances and Search Builder component for searching for project payments.

Also this component provides a static helper class ProjectPaymentFilterBuilder that defines method for creating filters that can be used when searching for project payments.

1.1 Design Patterns

Strategy pattern – manager and their implementations can be used in some external strategy context; managers use pluggable persistence instance.

Delegate pattern – CRUD methods of managers simply delegates execution to the namesake method of the pluggable persistence implementation instance.

DAO/DTO pattern – managers and persistence implementations are DAOs for ProjectPayment and partially for ProjectPaymentAdjustment DTOs.

Composite pattern – when building the filters in ProjectPaymentFilterBuider the Composite pattern is used.

1.2 Industry Standards

SQL, JDBC, XML (for configuration)

1.3 Required Algorithms

1.3.1 Logging

This component must perform logging in all public business methods of managers and persistence implementations.

All information described below must be logged using log:Log attribute. If log attribute is null, then logging is not required to be performed.

In all mentioned methods method entrance with input argument, 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 thrown exceptions and errors must be logged at ERROR level.

1.4 Component Class Overview

ProjectPayment

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

ProjectPaymentType

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

ProjectPaymentManager

This interface represents a project payment manager. It defines CRUD and search methods.

ProjectPaymentAdjustment

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

ProjectPaymentAdjustmentManager

This interface represents a project payment adjustment manager. It defines create/update and retrieve by project id methods.

ProjectPaymentFilterBuilder

This is a static helper class that provides factory methods for creating filters that can be used when searching for project payments.

ProjectPaymentManagerImpl

This class is an implementation of ProjectPaymentManager that uses Search Builder component to search for project payments in persistence and pluggable ProjectPaymentPersistence instance for CRUD operations in persistence. This class uses Logging Wrapper component to log errors and debug information.

ProjectPaymentAdjustmentManagerImpl

This class is an implementation of ProjectPaymentAdjustmentManager that uses pluggable ProjectPaymentAdjustmentPersistence instance for all operations in persistence. This class uses Logging Wrapper component to log errors and debug information.

ProjectPaymentPersistence

This interface represents project payment persistence. Currently it defines CRUD methods for project payment.

ProjectPaymentAdjustmentPersistence

This interface represents project payment adjustment persistence. Currently it defines create/update and retrieve by project id methods for project payment adjustment.

BaseProjectPaymentManagementPersistence

This class is base class for persistence classes of this component that uses JDBC and DB Connection Factory component. This class aggregates logger from Logging Wrapper component to be used by subclasses to log errors and debug information.

DatabaseProjectPaymentPersistence

This class is an implementation of ProjectPaymentPersistence that uses JDBC and DB Connection Factory component. This class uses Logging Wrapper component to log errors and debug information.

DatabaseProjectPaymentAdjustmentPersistence

This class is an implementation of ProjectPaymentAdjustmentPersistence that uses JDBC and DB Connection Factory component. This class uses Logging Wrapper component to log errors and debug information.

1.5 Component Exception Definitions

ProjetPaymentManagementConfigurationException

This exception is thrown by managers and persistence implementation when some error occurs while initializing an instance using the given configuration.

ProjectPaymentManagementException

This exception is thrown by implementations of managers when some not expected error occurred. Also this exception is used as a base class for other specific checked custom exceptions.

ProjectPaymentNotFoundException

This exception is thrown by implementations of managers and persistence interfaces if project payment with given id was not found in persistence.

ProjectPaymentValidationException

This exception is thrown if project payment is not valid.

ProjectPaymentAdjustmentValidationException
This exception is thrown if project payment adjustment is not valid.

ProjectPaymentManagementPersistenceException
This exception is thrown by implementations of persistence interfaces when error occurs while accessing the persistence.

ProjectPaymentManagementDataIntegrityException
This exception is thrown if project payment integrity is broken, for example if project payment is being persisted with payment type that is not present in DB.

1.6 Thread Safety

This component is thread safe.

Implementations of managers are required to be thread safe when entities passed to them are used by the caller in thread safe manner. Additionally it's assumed that configure() method of persistence implementations will be called just once right after instantiation.

Manager implementations provided in this component are immutable and thread safe when entities passed to them are used by the caller in thread safe manner. Thread safe SearchBundle, persistence and Log instances are used.

Persistence implementations are mutable, but thread safe when configure() method is called just once right after instantiation and entities passed to it are used by the caller in thread safe manner. It uses thread safe DBConnectionFactory and Log instances.

DTO entities are mutable and not thread safe entities.

ProjectPaymentFilterBuilder is immutable and thread safe static utility class.

2. Environment Requirements

2.1 Environment

Development language: Java 1.5
Compile target: Java 1.5, Java 1.6
QA Environment: Java 1.5, RedHat Linux 4, Windows 2000, Windows 2003

2.2 TopCoder Software Components

Base Exception 2.0 – is used by custom exceptions defined in this component.

Configuration API 1.1.0 – is used for initializing classes from this component.

Configuration Persistence 1.0.2 – is used for reading configuration from file.

Search Builder 1.3.3 – is used for searching.

Database Abstraction 2.0 – defines CustomResultSet class used in this component.

DB Connection Factory 1.1.1 – is used for creating database connections.

Logging Wrapper 2.0 – is used for logging errors and debug information.

Object Factory 2.0.1 – is used for creating pluggable object instances.

Object Factory Configuration API Plugin 1.0 – allows to use Configuration API for creating Object Factory.

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

None

3. Installation and Configuration

3.1 Package Name

com.topcoder.management.payment
com.topcoder.management.payment.impl
com.topcoder.management.payment.impl.persistence
com.topcoder.management.payment.search

3.2 Configuration Parameters

3.2.1 Configuration of ProjectPaymentManagerImpl

The following table describes the structure of ConfigurationObject passed to the constructor of manager class (angle brackets are used for identifying child configuration objects). This ConfigurationObject can be optionally read from a configuration file using Configuration Persistence component.

Parameter	**Description**	**Values**
loggerName	The name of Logging Wrapper logger to be used for logging errors and debug information. When not provided, logging is not performed.	String. Not empty. Optional.
searchBundleManagerNamespace	The namespace used when creating an instance of SearchBundleManager.	String. Not empty. Required.
projectPaymentSearchBundleName	The name of the search bundle used by this class when searching for project payments.	String. Not empty. Required.
<objectFactoryConfig>	This section contains configuration of Object Factory used by this class for creating pluggable object instances.	ConfigurationObject. Required.
projectPaymentPersistenceKey	The Object Factory key that is used for creating an instance of persistence to be used by this manager.	String. Not empty. Required.
<projectPaymentPersistenceConfig>	The configuration for persistence instance.	ConfigurationObject. Required.

3.2.2 Configuration of ProjectPaymentAdjustmentManagerImpl

The following table describes the structure of ConfigurationObject passed to the constructor of manager class (angle brackets are used for identifying child configuration objects). This ConfigurationObject can be optionally read from a configuration file using Configuration Persistence component.

Parameter	**Description**	**Values**
loggerName	The name of Logging Wrapper logger to be used for logging errors and debug information. When not provided, logging is not performed.	String. Not empty. Optional.
<objectFactoryConfig>	This section contains configuration of Object Factory used by this class for creating pluggable object instances.	ConfigurationObject. Required.

projectPaymentAdjustmentPersistenceKey	The Object Factory key that is used for creating an instance of persistence to be used by this manager.	String. Not empty. Required.
<projectPaymentAdjustmentPersistenceConfig>	The configuration for persistence instance.	ConfigurationObject. Required.

3.2.3 Configuration of DatabaseProjectPaymentPersistence and DatabaseProjectPaymentAdjustmentPersistence

The following table describes the structure of ConfigurationObject passed to the constructor of both persistence classes (angle brackets are used for identifying child configuration objects).

Parameter	**Description**	**Values**
loggerName	The name of Logging Wrapper logger to be used for logging errors and debug information. When not provided, logging is not performed.	String. Not empty. Optional.
dbConnectionFactoryConfig	The configuration to be used for creating DBConnectionFactoryImpl instance.	ConfigurationObject. Required.
connectionName	The connection name to be passed to the connection factory when establishing a database connection. If not specified, a default connection is used.	String. Not empty. Optional.

3.3 Dependencies Configuration

Please see docs of Logging Wrapper, DB Connection Factory and Object Factory components to configure them properly.

3.3.1 Configuration of Search Builder component

The proposed configuration for Search Builder component is provided below:

```
<?xml version="1.0"?>
<CMConfig>
  <!-- Namespace for DBConnectionFactory component -->
  <Config name="com.topcoder.db.connectionfactory.DBConnectionFactoryImpl">
    <Property name="connections">
      <Property name="default">
        <Value>dbconnection</Value>
      </Property>
      <Property name="dbconnection">
        <Property name="producer">
          <Value>com.topcoder.db.connectionfactory.producers.JDBCConnectionProducer</Value>
        </Property>
        <Property name="parameters">
          <Property name="jdbc_driver">
            <Value>com.informix.jdbc.IfxDriver</Value>
          </Property>
          <Property name="jdbc_url">
         <Value>jdbc:informix-sqli://localhost:1526/test:informixserver=ol_topcoder</Value>
          </Property>
          <Property name="SelectMethod">
            <Value>cursor</Value>
          </Property>
          <Property name="user">
            <Value>informix</Value>
          </Property>
          <Property name="password">
            <Value>123456</Value>
          </Property>
        </Property>
      </Property>
    </Property>
  </Config>

 <!-- Namespace for SearchBuilder component -->
<Config name="ProjectPaymentManagerImpl.SearchBuilderManager">
  <Property name="searchStrategyFactoryNamespace">
   <Value>com.topcoder.search.builder.strategy.factory</Value>
  </Property>
  <Property name="fieldValidatorFactoryNamespace">
   <Value>com.topcoder.search.builder.validator.factory</Value>
  </Property>
  <Property name="searchBundles">
   <Property name="Project Payment Search Bundle">
    <Property name="searchStrategy">
     <Property name="class">
      <Value>dbStrategy</Value>
     </Property>
    </Property>
    <Property name="context">
     <Value>
      SELECT project_payment.project_payment_id,
       project_payment.resource_id,
       project_payment.submission_id,
       project_payment.amount,
       project_payment.pacts_payment_id,
       project_payment.create_date,
       project_payment_type_lu.project_payment_type_id,
       project_payment_type_lu.name,
       project_payment_type_lu.mergeable,
       resource.project_id
      FROM project_payment
      INNER JOIN project_payment_type_lu
       ON project_payment.project_payment_type_id = project_payment_type_lu.project_payment_type_id
      INNER JOIN resource
       ON project_payment.resource_id = resource.resource_id
      WHERE
     </Value>
    </Property>
    <Property name="searchableFields">
     <Property name="id">
      <Property name="validator">
       <Property name="class">
        <Value>validator</Value>
       </Property>
       <Property name="identifier">
        <Value>default</Value>
       </Property>
      </Property>
     </Property>
     <Property name="projectPaymentId">
      <Property name="validator">
       <Property name="class">
        <Value>validator</Value>
       </Property>
       <Property name="identifier">
        <Value>default</Value>
       </Property>
      </Property>
     </Property>
     <Property name="resourceId">
      <Property name="validator">
       <Property name="class">
        <Value>validator</Value>
       </Property>
       <Property name="identifier">
        <Value>default</Value>
       </Property>
  </Property>
</Property>
<Property name="submissionId">
  <Property name="validator">
    <Property name="class">
      <Value>validator</Value>
    </Property>
    <Property name="identifier">
      <Value>default</Value>
    </Property>
  </Property>
</Property>
<Property name="amount">
  <Property name="validator">
    <Property name="class">
      <Value>validator</Value>
    </Property>
    <Property name="identifier">
      <Value>default</Value>
    </Property>
  </Property>
</Property>
<Property name="projectId">
  <Property name="validator">
    <Property name="class">
      <Value>validator</Value>
    </Property>
    <Property name="identifier">
      <Value>default</Value>
    </Property>
  </Property>
</Property>
<Property name="createDate">
  <Property name="validator">
    <Property name="class">
      <Value>validator</Value>
    </Property>
    <Property name="identifier">
      <Value>default</Value>
    </Property>
  </Property>
</Property>
<Property name="pactsPaymentId">
  <Property name="validator">
    <Property name="class">
      <Value>validator</Value>
    </Property>
    <Property name="identifier">
      <Value>default</Value>
    </Property>
  </Property>
</Property>
<Property name="projectPaymentTypeId">
  <Property name="validator">
    <Property name="class">
      <Value>validator</Value>
    </Property>
    <Property name="identifier">
      <Value>default</Value>
    </Property>
  </Property>
</Property>
<Property name="projectPaymentTypeName">
  <Property name="validator">
    <Property name="class">
      <Value>validator</Value>
    </Property>
    <Property name="identifier">
      <Value>default</Value>
    </Property>
```

```
          </Property>
        </Property>
        <Property name="projectPaymentTypeMergeable">
          <Property name="validator">
            <Property name="class">
              <Value>validator</Value>
            </Property>
            <Property name="identifier">
              <Value>default</Value>
            </Property>
          </Property>
        </Property>
      </Property>
      <Property name="alias">
        <Property name="projectPaymentId">
          <Value>project_payment.project_payment_id</Value>
        </Property>
        <Property name="resourceId">
          <Value>project_payment.resource_id</Value>
        </Property>
        <Property name="projectId">
          <Value>resource.project_id</Value>
        </Property>
        <Property name="projectPaymentTypeId">
          <Value>project_payment.project_payment_type_id</Value>
        </Property>
        <Property name="submissionId">
          <Value>project_payment.submission_id</Value>
        </Property>
        <Property name="pactsPaymentId">
          <Value>project_payment.pacts_payment_id</Value>
        </Property>
        <Property name="createDate">
          <Value> project_payment.create_date</Value>
        </Property>
      </Property>
    </Property>
  </Property>
</Config>

<Config name="com.topcoder.search.builder.validator.factory">
  <Property name="validator:null">
    <Property name="type">
      <Value>com.topcoder.util.datavalidator.NullValidator</Value>
    </Property>
  </Property>
  <Property name="validator:default">
    <Property name="type">
      <Value>com.topcoder.management.payment.MockValidator</Value>
    </Property>
  </Property>
</Config>
<Config name="com.topcoder.search.builder.strategy.factory">
  <Property name="dbStrategy">
    <Property name="type">
      <Value>com.topcoder.search.builder.database.DatabaseSearchStrategy</Value>
    </Property>
    <Property name="params">
      <Property name="param1">
        <Property name="type">
          <Value>String</Value>
        </Property>
        <Property name="value">
          <Value>DBSearchStrategy</Value>
        </Property>
      </Property>
    </Property>
  </Property>
</Config>
<Config name="DBSearchStrategy">
```

```
<!-- Property defining a specification for constructing the dbConnectionFactory to use. -->
<Property name="connectionFactory">

  <!-- The namespace of the ConnectionFactory -->
  <Property name="name">
    <Value>com.topcoder.db.connectionfactory.DBConnectionFactoryImpl</Value>
  </Property>
  <Property name="class">
    <Value>com.topcoder.db.connectionfactory.DBConnectionFactoryImpl</Value>
  </Property>
</Property>


<!-- The name to request from the connection factory when acquiring a connection. If not present, then the default connection is used. -->
<Property name="connectionName">
  <Value>dbconnection</Value>
</Property>


<Property name="searchFragmentFactoryNamespace">
  <Value>com.topcoder.search.builder.database.factory</Value>
</Property>

<Property name="searchFragmentBuilders">
  <Property name="first">
    <Property name="targetFilter">
      <Value>com.topcoder.search.builder.filter.AndFilter</Value>
    </Property>
    <Property name="className">
      <Value>com.topcoder.search.builder.database.AndFragmentBuilder</Value>
    </Property>
  </Property>

  <Property name="second">
    <Property name="targetFilter">
      <Value>com.topcoder.search.builder.filter.OrFilter</Value>
    </Property>
    <Property name="className">
      <Value>com.topcoder.search.builder.database.OrFragmentBuilder</Value>
    </Property>
  </Property>

  <Property name="third">
    <Property name="targetFilter">
      <Value>com.topcoder.search.builder.filter.LikeFilter</Value>
    </Property>
    <Property name="className">
      <Value>com.topcoder.search.builder.database.LikeFragmentBuilder</Value>
    </Property>
  </Property>

  <Property name="fourth">
    <Property name="targetFilter">
      <Value>com.topcoder.search.builder.filter.NotFilter</Value>
    </Property>
    <Property name="className">
      <Value>com.topcoder.search.builder.database.NotFragmentBuilder</Value>
    </Property>
  </Property>

  <Property name="fifth">
    <Property name="targetFilter">
      <Value>com.topcoder.search.builder.filter.EqualToFilter</Value>
    </Property>
    <Property name="className">
      <Value>com.topcoder.search.builder.database.EqualsFragmentBuilder</Value>
    </Property>
  </Property>
```

```
      <Property name="sixth">
        <Property name="targetFilter">
          <Value>com.topcoder.search.builder.filter.InFilter</Value>
        </Property>
        <Property name="className">
          <Value>com.topcoder.search.builder.database.InFragmentBuilder</Value>
        </Property>
      </Property>

      <Property name="eighth">
        <Property name="targetFilter">
          <Value>com.topcoder.search.builder.filter.NullFilter</Value>
        </Property>
        <Property name="className">
          <Value>com.topcoder.search.builder.database.NullFragmentBuilder</Value>
        </Property>
      </Property>

      <Property name="ninth">
        <Property name="targetFilter">
          <Value>com.topcoder.search.builder.filter.GreaterThanFilter</Value>
        </Property>
        <Property name="className">
          <Value>com.topcoder.search.builder.database.RangeFragmentBuilder</Value>
        </Property>
      </Property>

      <Property name="tenth">
        <Property name="targetFilter">
          <Value>com.topcoder.search.builder.filter.GreaterThanOrEqualToFilter</Value>
        </Property>
        <Property name="className">
          <Value>com.topcoder.search.builder.database.RangeFragmentBuilder</Value>
        </Property>
      </Property>

      <Property name="eleventh">
        <Property name="targetFilter">
          <Value>com.topcoder.search.builder.filter.BetweenFilter</Value>
        </Property>
        <Property name="className">
          <Value>com.topcoder.search.builder.database.RangeFragmentBuilder</Value>
        </Property>
      </Property>

      <Property name="twelvth">
        <Property name="targetFilter">
          <Value>com.topcoder.search.builder.filter.LessThanOrEqualToFilter</Value>
        </Property>
        <Property name="className">
          <Value>com.topcoder.search.builder.database.RangeFragmentBuilder</Value>
        </Property>
      </Property>

      <Property name="thirteenth">
        <Property name="targetFilter">
          <Value>com.topcoder.search.builder.filter.LessThanFilter</Value>
        </Property>
        <Property name="className">
          <Value>com.topcoder.search.builder.database.RangeFragmentBuilder</Value>
        </Property>
      </Property>
    </Property>
  </Config>
  <Config name="com.topcoder.search.builder.database.factory">
  </Config>
</CMConfig>
```

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Setup Informix database test:

  a. Run test_files/create.sql to create the tables.

  test_files/drop.sql can be used to drop tables.

  b. Update "jdbc_url", “user” and “password” properties in test_files/ ProjectPaymentManager.xml and test_files/SearchBundleManager.xml if needed.

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

4.2 Required steps to use the component

Please see the demo.

4.3 Demo

4.3.1 Sample ProjectPaymentManagerImpl configuration file

```
<?xml version="1.0"?>
<CMConfig>
 <Config name="com.topcoder.management.payment.impl.ProjectPaymentManagerImpl">
  <Property name="loggerName">
   <Value>myLogger</Value>
  </Property>
  <Property name="objectFactoryConfig">
   <Property name="DatabaseProjectPaymentPersistence">
    <Property name="type">
     <Value>com.topcoder.management.payment.impl.persistence.DatabaseProjectPaymentPersistence</Value>
    </Property>
   </Property>
  </Property>
  <Property name="searchBundleManagerNamespace">
   <Value>ProjectPaymentManagerImpl.SearchBuilderManager</Value>
  </Property>
  <Property name="projectPaymentSearchBundleName">
   <Value>Project Payment Search Bundle</Value>
  </Property>
  <Property name="projectPaymentPersistenceKey">
   <Value>DatabaseProjectPaymentPersistence</Value>
  </Property>
  <Property name="projectPaymentPersistenceConfig">
   <Property name="loggerName">
    <Value>myLogger</Value>
   </Property>
   <Property name="dbConnectionFactoryConfig">
    <Property name="com.topcoder.db.connectionfactory.DBConnectionFactoryImpl">
     <Property name="connections">
      <Property name="default">
       <Value>myConnection</Value>
      </Property>
      <Property name="myConnection">
       <Property name="producer">
         <Value>com.topcoder.db.connectionfactory.producers.JDBCConnectionProducer</Value>
       </Property>
       <Property name="parameters">
        <Property name="jdbc_driver">
         <Value>com.informix.jdbc.IfxDriver</Value>
        </Property>
        <Property name="jdbc_url">
         <Value>jdbc:informix-sqli://localhost:1526/test:informixserver=ol_topcoder</Value>
        </Property>
        <Property name="SelectMethod">
```

```
                <Value>cursor</Value>
              </Property>
              <Property name="user">
                <Value>informix</Value>
              </Property>
              <Property name="password">
                <Value>123456</Value>
              </Property>
            </Property>
          </Property>
        </Property>
      </Property>
    </Property>
    <Property name="connectionName">
      <Value>myConnection</Value>
    </Property>
  </Property>
 </Config>
</CMConfig>
```

4.3.2 Sample ProjectPaymentAdjustmentManagerImpl configuration file

```
<?xml version="1.0"?>
<CMConfig>
 <Config name="com.topcoder.management.payment.impl.ProjectPaymentAdjustmentManagerImpl">
  <Property name="loggerName">
    <Value>myLogger</Value>
  </Property>
  <Property name="objectFactoryConfig">
    <Property name="DatabaseProjectPaymentAdjustmentPersistence">
      <Property name="type">

<Value>com.topcoder.management.payment.impl.persistence.DatabaseProjectPaymentAdjustmentPersistence</Value>
      </Property>
    </Property>
  </Property>
  <Property name="projectPaymentAdjustmentPersistenceKey">
    <Value>DatabaseProjectPaymentAdjustmentPersistence</Value>
  </Property>
  <Property name="projectPaymentAdjustmentPersistenceConfig">
    <Property name="loggerName">
      <Value>myLogger</Value>
    </Property>
    <Property name="dbConnectionFactoryConfig">
      <Property name="com.topcoder.db.connectionfactory.DBConnectionFactoryImpl">
        <Property name="connections">
          <Property name="default">
            <Value>myConnection</Value>
          </Property>
          <Property name="myConnection">
            <Property name="producer">
                <Value>com.topcoder.db.connectionfactory.producers.JDBCConnectionProducer</Value>
            </Property>
            <Property name="parameters">
              <Property name="jdbc_driver">
                <Value>com.informix.jdbc.IfxDriver</Value>
              </Property>
              <Property name="jdbc_url">
                <Value>jdbc:informix-sqli://localhost:1526/test:informixserver=ol_topcoder</Value>
              </Property>
              <Property name="SelectMethod">
                <Value>cursor</Value>
              </Property>
              <Property name="user">
                <Value>informix</Value>
              </Property>
              <Property name="password">
                <Value>123456</Value>
              </Property>
            </Property>
          </Property>
        </Property>
      </Property>
    </Property>
    <Property name="connectionName">
      <Value>myConnection</Value>
    </Property>
  </Property>
</Config>
</CMConfig>
```

4.3.3 Sample input and output

Assume that initially database contains the following data:

project_payment_type_lu

project_payment_type_id	name	mergeable
1	Contest Payment	false
3	Review Payment	true

project_payment table
(not important columns are skipped)

project_payment_id	submission_id	resource_id	pacts_payment_id	amount	project_payment_type_id
1	1011111	1001	4	500.0	1
2	1022222	1002	3	800.0	1

project_payment_adjustment table
(not important columns are skipped)

project_id	resource_role_id	fixed_amount	multiplier
1001	4 (Reviewer)	50.0	NULL
1002	3 (Screener)	0.0	NULL
1003	13 (Manager)	100.0	NULL

Then the following code can be executed:

```
 // Create an instance of ProjectPaymentManagerImpl using configuration
ConfigurationObject configuration = getConfig(ProjectPaymentManagerImpl.DEFAULT_CONFIG_NAMESPACE);
ProjectPaymentManagerImpl projectPaymentManager = new ProjectPaymentManagerImpl(configuration);

// Create an instance of ProjectPaymentManagerImpl using config file and namespace
projectPaymentManager = new ProjectPaymentManagerImpl(ProjectPaymentManagerImpl.DEFAULT_CONFIG_FILENAME,
    ProjectPaymentManagerImpl.DEFAULT_CONFIG_NAMESPACE);

// Create an instance of ProjectPaymentManagerImpl using default config file
projectPaymentManager = new ProjectPaymentManagerImpl();

// Retrieve the project payment with ID=1
ProjectPayment projectPayment = projectPaymentManager.retrieve(1);
// id must be 1
// submission id must be 1011111
// resource id must be 1001
// PACTS payment id must be 4
// amount must be 500.0
// etc.

// Update the project payment by changing amount
projectPayment.setAmount(BigDecimal.valueOf(600));
projectPaymentManager.update(projectPayment);
```

After this step the database must be updated respectively:

project_payment_id	submission_id	resource_id	pacts_payment_id	amount	project_payment_type_id

1	1011111	1001	4	600.0	1
2	1022222	1002	3	800.0	1

```
// Search for all project payments with submission ID=1011111
Filter submissionIdFilter = ProjectPaymentFilterBuilder.createSubmissionIdFilter(1011111);

List<ProjectPayment> projectPayments = projectPaymentManager.search(submissionIdFilter);
// projectPayments.size() must be 1
// id must be 1
// submission id must be 1011111
// resource id must be 1001
// PACTS payment id must be 4
// amount must be 600.0
// etc.

// Delete the project payment by id
projectPaymentManager.delete(2);
```

After this step the database must be updated respectively:

project_payment_id	submission_id	resource_id	pacts_payment_id	amount	project_payment_type_id
1	1011111	1001	4	600.0	1

```
// Create an instance of ProjectPaymentAdjustmentManagerImpl using configuration
configuration = getConfig(ProjectPaymentAdjustmentManagerImpl.DEFAULT_CONFIG_NAMESPACE);
ProjectPaymentAdjustmentManagerImpl projectPaymentAdjustmentManager = new ProjectPaymentAdjustmentManagerImpl(
    configuration);

// Retrieve the project payment adjustments with project ID=1001
List<ProjectPaymentAdjustment> projectPaymentAdjustments = projectPaymentAdjustmentManager
    .retrieveByProjectId(1001);
// projectPaymentAdjustmentss.size() must be 1
// project id must be 1001
// resource role id must be 4
// fixed amount must be 50
// etc.
```

5. Future Enhancements

None