DR Points Manager 1.0 Component Specification

1. Design

This component provides operations on digital run points entities like create/update/remove/get/search for DigitalRunPoints and create/update/remove/get/get all for DigitalRunPointsType, DigitalRunPointsOperation, DigitalRunPointsStatus and DigitalRunPointsReferenceType. Component runs as a stateless EJB. The ejb will make use of five daos to interact with the persistence. This component is used by Digital Run Service component.

1.1 Design Patterns

Strategy: Different implementations of DAO interfaces can be plugged into the DigitalRunPointsManagerBean. Application can also used different implementations of DigitalRunPointsManager interface.
DAO: DigitalRunPointsDAO, DigitalRunPointsTypeDAO, DigitalRunPointsReferenceTypeDAO, DigitalRunPointsStatusDAO and DigitalRunPointsOperationDAO are dao interfaces.
Factory: DigitalRunPointsFilterFactory implements this pattern.

1.2 Industry Standards

- EJB 3.0

1.3 Required Algorithms

1.3.1 How to perform logging

Logging will be performed in all public methods and in AbstractDAO.getEntityManager protected method. Following are the rules for logging:

- Entrance and exit of methods should be logged at the INFO level
- Exception should be logged at the ERROR level

1.4 Component Class Overview

DigitalRunPointsManagerBean
This class is a stateless sesion bean that manages digital run points entities. It implements the DigitalRunPointsManagerLocal and DigitalRunPointsManagerRemote interfaces. All public methods from this class perform logging using LoggingWrapper component. This bean will not access the persistence directly; it will use five daos to access the persistence. These daos will be creates in initialize post construct method using Object Factory. The logger will also be initialized in initialize method.

This class will be annotated with:
@Stateless
@TransactionManagement(TransactionManagementType.CONTAINER)

@TransactionAttribute(TransactionAttributeType.REQUIRED)

Since the container will manage the transactions this class can be considered as thread safe. This bean is also immutable.

DigitalRunPointsManager
This interface defines the contract for classes that will manage digital run points entities into persistence. The implementing classes will provide the following functionality:
-create a digital run points/ digital run points type /digital run points status/digital run points operation/digital run points reference type into persistence
-update a digital run points/digital run points type/digital run points status/digital run points operation/digital run points reference type into persistence
-remove a digital run points/digital run points type/digital run points status/digital run points operation/digital run points reference type identified by its id from persistence
-get a digital run points/digital run points type/digital run points status/digital run points operation/digital run points reference type identified by its id from persistence
-search the digital run points entities in persistence that match a given filter
-get all digital run points type/digital run points status/digital run points operation/digital run points reference type from persistence
Implementations are required to be thread safe.

DigitalRunPointsManagerLocal
This EJB Local interface extends the DigitalRunPointsManager, and it is marked with @Local annotation to indicate it's an EJB Local interface.
Implementation will run in an EJB container and thread safety depends on the container. Generally the implementation is thread safe when it's running inside the container.

DigitalRunPointsManagerRemote
This EJB Remote interface extends the DigitalRunPointsManager, and it is marked with @Remote annotation to indicate it's an EJB Remote interface.
Implementation will run in an EJB container and thread safety depends on the container. Generally the implementation is thread safe when it's running inside the container.

ConfigurationProvider
This class holds retrieves and stores the configuration for the current bean (and possibly for future beans and different implementations of DigitalRunPointsManager interface). It uses Configuration Persistence component to retrieve the configuration from a file. The user can also provide the configuration object by calling setConfiguration method. The configuration can be retrieved by calling getConfiguration method. This class was created because file access in ejb is forbidden by ejb specification.

This class is thread safe because all the methods are synchronized.

DigitalRunPointsFilterFactory
This class provides static methods that create filters that can be used by the DigitalRunPointsManager implementations to retrieve DigitalRunPoints entities that match the filter criteria, from the persistence. The methods provided return filters with:
-equals/in operation for points id/track id/points status id/points type id/user id
-equals/like operation points description
-equals operation for is potential
-equals+equals/equals+in operations for reference type id+reference id
-equals/greater or equal than/lower or equal than for amount, points application date and points award date
This class also provides and/or/not filters so that the user does not need to interact with Search Builder component to create such filters.
This class is thread safe since it has no state.

DigitalRunPointsDAO
This interface defines the contract for classes that will manage digital run points entities into persistence. The implementing classes will provide the following functionality:
-create a digital run points entity into persistence
-update a digital run points entity into persistence
-remove a digital run points entity identified by its id from persistence
-get a digital run points entity identified by its id from persistence
-search the digital run points entities in persistence that match a given filter
Implementations are not required to be thread safe.

DigitalRunPointsTypeDAO
This interface defines the contract for classes that will manage digital run points type entities into persistence. The implementing classes will provide the following functionality:
-create a digital run points type entity into persistence
-update a digital run points type entity into persistence
-remove a digital run points type entity identified by its id from persistence
-get a digital run points type entity identified by its id from persistence
-get all digital run points type entities from persistence
Implementations are not required to be thread safe.

DigitalRunPointsReferenceTypeDAO
This interface defines the contract for classes that will manage digital run points reference type entities into persistence. The implementing classes will provide the following functionality:
-create a digital run points reference type entity into persistence
-update a digital run points reference type entity into persistence

-remove a digital run points reference type entity identified by its id from persistence
-get a digital run points reference type entity identified by its id from persistence
-get all digital run points reference type entities from persistence
Implementations are not required to be thread safe.

DigitalRunPointsStatusDAO
This interface defines the contract for classes that will manage digital run points status entities into persistence. The implementing classes will provide the following functionality:
-create a digital run points status entity into persistence
-update a digital run points status entity into persistence
-remove a digital run points status entity identified by its id from persistence
-get a digital run points status entity identified by its id from persistence
-get all digital run points status entities from persistence
Implementations are not required to be thread safe.

DigitalRunPointsOperationDAO
This interface defines the contract for classes that will manage digital run points operation entities into persistence. The implementing classes will provide the following functionality:
-create a digital run points operation entity into persistence
-update a digital run points operation entity into persistence
-remove a digital run points operation entity identified by its id from persistence
-get a digital run points operation entity identified by its id from persistence
-get all digital run points operation entities from persistence
Implementations are not required to be thread safe.

AbstractDAO
This abstract class holds the common field of all the five concrete dao implementations. It has a SessionContext and an unit name used to obtain an EntityManager instance and a Log instance used to perform logging if logging is desired; if logging is not desired this field will remain null.It provides a method that obtains an EntityManager instance.
It is not thread safe because it is mutable. Anyway, setters are meant to be called only once so there will be no thraed safety issues.

JPADigitalRunPointsTypeDAO
This class implements DigitalRunPointsTypeDAO interface. It also extends the AbstractDAO class. This class manages DigitalRunPointsType entities in a JPA persistence (currently the JPA persistence will use Hibernate as a provider but any provider can be used). Each public method performs logging.
This class is not completely thread safe because it doesn't manage transactions and it is also mutable. Anyway, the intent is to use this implementation in a stateless session bean so there will be no thread safety issues generated by this class since the container will ensure thread safety.

JPADigitalRunPointsStatusDAO
This class implements DigitalRunPointsStatusDAO interface. It also extends the AbstractDAO class. This class manages DigitalRunPointsStatus entities in a JPA persistence (currently the JPA persistence will use Hibernate as a provider but any provider can be used). Each public method performs logging.
This class is not completely thread safe because it doesn't manage transactions and it is also mutable. Anyway, the intent is to use this implementation in a stateless session bean so there will be no thread safety issues generated by this class since the container will ensure thread safety.

JPADigitalRunPointsDAO
This class implements DigitalRunPointsDAO interface. It also extends the AbstractDAO class. This class manages DigitalRunPoints entities in a JPA persistence (currently the JPA persistence will use Hibernate as a provider but any provider can be used). It uses Search Builder component to search for DigitalRunPoints entities. Each public method performs logging.
This class is not completely thread safe because it doesn't manage transactions and it is also mutable. Anyway, the intent is to use this implementation in a stateless session bean so there will be no thread safety issues generated by this class since the container will ensure thread safety.

JPADigitalRunPointsOperationDAO
This class implements DigitalRunPointsOperationDAO interface. It also extends the AbstractDAO class. This class manages DigitalRunPointsOperation entities in a JPA persistence (currently the JPA persistence will use Hibernate as a provider but any provider can be used). Each public method performs logging.
This class is not completely thread safe because it doesn't manage transactions and it is also mutable. Anyway, the intent is to use this implementation in a stateless session bean so there will be no thread safety issues generated by this class since the container will ensure thread safety.

JPADigitalRunPointsReferenceTypeDAO
This class implements DigitalRunPointsReferenceTypeDAO interface. It also extends the AbstractDAO class. This class manages DigitalRunPointsType entities in a JPA persistence (currently the JPA persistence will use Hibernate as a provider but any provider can be used). Each public method performs logging.
This class is not completely thread safe because it doesn't manage transactions and it is also mutable. Anyway, the intent is to use this implementation in a stateless session bean so there will be no thread safety issues generated by this class since the container will ensure thread safety.

1.5 Component Exception Definitions

DigitalRunPointsManagerException

This is the base custom exception of all checked custom exceptions defined in this component. It is not thrown directly by any class.

EntityNotFoundException
This is a custom exception that will be thrown if an entity is not found in the persistence.

DigitalRunPointsManagerPersistenceException
This is a custom exception that will be thrown to indicate that errors occurred when accessing the persistence storage.

DigitalRunPointsManagerBeanConfigurationException
This is a custom exception that will be thrown to indicate that configuration related errors occurred (missing required property or errors that occur when getting the configuration from file).

1.6 Thread Safety

The component is not completely thread safe. The DigitalRunPointsManagerBean is thread safe because it is immutable and it will run in an ejb container which ensures thread safety. DigitalRunPointsFilterFactory is thread safe because it has no state. ConfigurationProvider is thread safe because its methods are synchronized. The dao implementations are not completely thread safe because they don’t manage transactions. The dao implementations are intended to be plugged into the bean and in this case there are no thread safety issues; also, considering this usage, it is not necessary to make the dao implementations thread safe.

2. Environment Requirements

2.1 Environment

- Development language: Java 1.5
- Compile Target: Java 1.5
- RedHat Linux 9
- JBoss 4.2 GA
- Informix 10.00.UC 5

2.2 TopCoder Software Components

- **Configuration API 1.0** - used for configuration purposes
- **Configuration Persistence 1.0** - used to get the configuration from file
- **Object Factory 2.0.1** - used to create objects
- **Object Factory Configuration API Plugin 1.0** - used with Object Factory component to create objects from ConfigurationObject.
- **Base Exception 2.0** - custom exceptions extend from BaseCriticalException and BaseRuntimeException

- **Search Builder 1.3.2 -** used to perform searches with filters
- **Digital Run Entities 1.0 -** entities are used from this component
- **Logging Wrapper 2.0 -** used to perform logging

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.service.digitalrun.points
com.topcoder.service.digitalrun.points.manager.bean
com.topcoder.service.digitalrun.points.dao.implementations

3.2 Configuration Parameters

Configuration parameters for the bean:

Parameter Name	Parameter Description	Parameter Value
logName	Used to create a Log instance. If not present it means that logging is disabled. **Optional.**	non empty string
pointsDAOKey	The key used with ObjectFactory to create a DigitalRunPointsDAO instance. **Required.**	non empty string
pointsTypeDAOKey	The key used with ObjectFactory to create a DigitalRunPointsTypeDAO instance. **Required.**	non empty string
pointsOperationDAOKey	The key used with ObjectFactory to create a DigitalRunPointsOperationDAO instance. **Required.**	non empty string
pointsReferenceTypeDAOKey	The key used with ObjectFactory to create a DigitalRunPointsReferenceTypeDAO instance. **Required.**	non empty string
pointsStatusDAOKey	The key used with ObjectFactory to create a DigitalRunPointsStatusDAO instance. **Required.**	non empty string

The configuration will also contain properties necessary to create a SearchBundle instance. It will have three properties :

Configuration parameters for the bean:

Parameter Name	**Parameter Description**	**Parameter Value**
name	The name of the SearchBundle instance.. **Required.**	non empty string
search_strategy_key	The key used with ObjectFactory to create a SearchStrategy instance. **Required.**	non empty string
context	The context. **Required.**	non empty string

It will also contain a child that contains the searchable fields of the SearchBundle. This child will be registered under “fields” name. This child contains attributes where the attribute name represents the field and the attribute value is a key used with ObjectFactory to create an ObjectValidator instance.

It will also contain a child that contains the column aliases. This child will be registered under “aliases” name. The attribute name is the column alias while the value is the column name,

3.3 Dependencies Configuration

None.

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 bean. The bean will be configured in ejb-jar.xml (see ejb-jar.xml from /docs directory)

4.3 Demo

1.3.1 Customer scenario that shows how to work with DigitalRunPoints entities

```
-it is necessary for the application to place the configuration
in the ConfigurationProvider class so that the bean can create
the necessary daos
```

```
-ConfigurationObject config = …;
ConfigurationProvider.setConfiguration(config);
//or
ConfigurationProvider.retrieveConfigurationFromFile(filename,
namespace);

// lookup the bean from JNDI
InitialContext ctx = …;
DigitalRunPointsManagerRemote bean =
(DigitalRunPointsManagerRemote)
ctx.lookup("DigitalRunPointsManagerBean/remote");

//create a DigitalRunPoints instance and set its fields
DigitalRunPoints drp = new DigitalRunPoints();
//ceate a DigitalRunPointsStatus, DigitalRunPointsType,
//DigitalRunPointsReferenceType and DigitalRunPointsOperation
//entities and set them on DigitalRunPoints instance then set the
//rest of DigitalRunPoints fields; among others set amount to 100
…
drp.setAmount(100);

//create DigitalRunPoints entity into persistence
drp = bean.createDigitalRunPoints(drp);

//the drp entity has now an id; the amount is 100
//modify drp.amount field and update
drp.setAmount(200);
bean.updateDigitalRunPoints(drp);


//get a DigitalRunPoints entity by specifying an id
long id = drp.getId();
DigitalRunPoints drp1 = bean.getDigitalRunPoints(id);
//the DigitalRunPoints created above was returned

//search for DigitalRunPoints entities using a filter that
//returns entities with amount <= 200
Filter filter = DigitalRunPointsFilterFactory.
createAmountLowerOrEqualFilter(200);
List<DigitalRunPoints> drpList =
bean.searchDigitalRunPoints(filter);
//the list contains the DigitalRunPoints entity created above

//remove the created DigitalRunPoints from persistence
bean.removeDigitalRunPoints(id);
```

1.3.2 How to manage DigitalRunPointsType instances

```
//create digital run points type
DigitalRunPointsType type = bean.createDigitalRunPointsType
                                                     (type);

//update digital run points type
bean.updateDigitalRunPointsType(type);

//get digital run points type
```

```
DigitalRunPointsType type1 = bean.get DigitalRunPointsType(id);

//get all digital run points types
List<DigitalRunPointsType> types = bean.getAll
                                        DigitalRunPointsTypes();

//remove digital run points type
bean.removeDigitalRunPointsType(id);
```

1.3.3 How to manage DigitalRunPointsReferenceType instances

```
//create digital run points reference type
DigitalRunPointsReferenceType referenceType =

bean.createDigitalRunPointsReferenceType(referenceType);

//update digital run points reference type
bean.updateDigitalRunPointsReferenceType(referenceType);

//get digital run points reference type
DigitalRunPointsReferenceType referenceType1 = bean.get
                             DigitalRunPointsReferenceType(id);

//get all digital run points reference types
List<DigitalRunPointsReferenceType> referenceTypes = bean.getAll
                               DigitalRunPointsReferenceTypes();

//remove digital run points reference type
bean.removeDigitalRunPointsReferenceType(id);
```

1.3.4 How to manage DigitalRunPointsStatus instances

```
//create digital run points status
DigitalRunPointsStatus status = bean.create
                              DigitalRunPointsStatus(status);

//update digital run points status
bean.updateDigitalRunPointsStatus(status);

//get digital run points status
DigitalRunPointsStatus status1 = bean.get
                                    DigitalRunPointsStatus(id);

//get all digital run points statuses
List<DigitalRunPointsStatus> statuses = bean.getAll
                                     DigitalRunPointsStatuses();

//remove digital run points status
bean.removeDigitalRunPointsStatus(id);
```

1.3.5 How to manage DigitalRunPointsOperation instances

```
//create digital run points operation
DigitalRunPointsOperation operation = bean.create
                           DigitalRunPointsOperation(operation);
```

```
//update digital run points operation
bean.updateDigitalRunPointsOperation(operation);

//get digital run points operation
DigitalRunPointsOperation operation1 = bean.get
                               DigitalRunPointsOperation(id);

//get all digital run points operation
List<DigitalRunPointsOperation> operations = bean.getAll
                                DigitalRunPointsOperations();

//remove digital run points operation
bean.removeDigitalRunPointsOperation(id);
```

5. Future Enhancements

Auditing can be added in the future for the create/update/remove methods.