TopCoder Cockpit Project Metadata Services Requirements Specification

1. Scope

1.1 Overview

In the TopCoder Cockpit application, we are going to allow user to associate metadata to the cockpit project. The project metadata consists of predefined metadata and custom metadata.

The project metadata are generally key / value pairs associated with the cockpit project. For predefined metadata, the key are predefined, the user only needs to set the value. For custom metadata, the user needs to set a key first, then set a value for that key. The metadata value can be specified by the user with any value or choose from a predefined list.

For project metadata types, we can only set one value, for example, for the metadata 'budget', we only about one value set for the budget of the project.

For other project metadata types, for example, the metadata 'project technologies', we allow multiple values to be set like 'Java', 'Web application', 'EJB3', 'struts2' for example.

This component is responsible for providing project metadata services.

1.2 Logic Requirements

1.2.1 Scope

This component is responsible for implementing the interfaces and exceptions defined in com.topcoder.direct.services.project.metadata package.

1.2.2 DirectProjectMetadataKeyService

An implementation should be provided for it to manage the project metadata key and predefined values in database.

The project metadata key can be common to all clients or specific to a certain client depending on the DirectProjectMetadataKey.clientId is null or not.

This service provides the following methods:

- createProjectMetadataKey – create new DirectProjectMetadataKey
- updateProjectMetadataKey – update an existing DirectProjectMetadataKey
- saveProjectMetadataKey – create or update the DirectProjectMetadataKey
- deleteProjectMetadataKey – delete an existing DirectProjectMetadataKey
- getProjectMetadataKey – get the DirectProjectMetadataKey by its id
- getCommonProjectMetadataKeys – get the common DirectProjectMetadataKeys (whose clientId = null)
- getClientProjectMetadataKeys – get client specific DirectProjectMetadataKeys (with the specified clientId). If the grouping argument is null, all client specific keys are returned, otherwise, only the client specific keys with the specified grouping value are returned.

1.2.2.1 DirectProjectMetadataKeyValidator

In DirectProjectMetadataKeyService, the DirectProjectMetadataKey must be validated first before being stored into the database.

An implementation must be provided for DirectProjectMetadataKeyValidator to perform the following validation:

- The metadata key name must be non-empty and unique
- Common metadata key must have null clientId and grouping attributes.
- For client specific metadata key
  - its key name must not exceed 20 characters
  - its grouping attribute must be non-null
  - its predefinedValues must be empty or null.

1.2.3 DirectProjectMetadataService

An implementation should be provided for it to manage the project metadata.

This service provides the following methods:

- createProjectMetadata – create new DirectProjectMetadata
- updateProjectMetadata – update an existing DirectProjectMetadata
- saveProjectMetadata – create or update the DirectProjectMetadata
- deleteProjectMetadata – delete an existing DirectProjectMetadata
- getProjectMetadata – get the DirectProjectMetadata by its id
- getProjectMetadataByProject – get the DirectProjectMetadata list associated with the given tcDirectProjectId
- addProjectMetadata
  - addProjectMetadata(tcDirectProjectId, projectMetadataList) – it adds a list of project metadata into the given tc direct project.
  - addProjectMetadata(tcDirectProjectIds, projectMetadata) – it adds the project metadata into a list of given tc direct projects.
- searchProjects – search tc direct projects by the given filter. The following filters need to be supported:
  - MetadataKeyIdValueFilter – the tc direct project has the metadata whose key matches the key in the filter and whose value should compare with the value in the filter using the metadataValueOperator.
  - MetadataKeyNameValueFilter – the tc direct project has the metadata whose key matches the key in the filter and whose value should compare with the value in the filter using the metadataValueOperator.
  - CompositeFilter – its inner projectFilters will be matched with the AND or OR compositeOperator.

Note that the TcDirectProject is stored in a different database (corporate_oltp).

1.2.3.1 DirectProjectMetadataValidator

In DirectProjectMetadataService, the project metadata must be validated first before being stored into the database.

An implementation must be provided for DirectProjectMetadataValidator to perform the following validation:

- The DirectProjectMetadataKey.single flag indicates the key allows single value or multiple values
- For client specific metadata key, no other validation is needed.
- Several common metadata keys are predefined, and their values should be validated as the table below.

Common Metadata Key	**Description**	**Value Format**
Budget	the budget of the project	positive integer value
Project Status	the status of the project	choose from the predefined value list

Project Technologies	The technologies the project uses	choose from the predefined value list
Planned Duration	The scheduled duration of the project	a positive integer value represents days like *120 days*
Client handles	The topcoder handles of client stakeholders of the project	N/A
TC PM handles	The topcoder handles of the topcoder project manager	N/A
Private Flag	whether the project is a private	true or false
Project Jira Key	The key of Jira project which is set up for this cockpit project	Non-empty string
Project Forum Id	The id of project forum where client / PM / Copilot discuss about the project	positive integer value
Project SVN address	The SVN URL used as the repository of the cockpit project	Valid SVN URL address

Designer should provide generic validators according to the value format and then configure them for specific common metadata key.

1.2.4 Exceptions

The following exceptions must be implemented:

- DirectProjectServiceException – the parent exception for all custom exceptions in this module
- ValidationException – thrown if the metadata fails validation
- EntityNotFoundException – thrown if the requested entity is not found in db for update and delete methods.
- PersistenceException – thrown if db error occurs.
- ConfigurationException – thrown if configuration error occurs.

Designers are allowed to add new custom exceptions by extending proper predefined exceptions.

1.2.5 Guideline

Note that for the get (or search) service method, if the method returns a single object and the requested object doesn't exist, null should be returned; if the method returns a list and the requested objects don't exist, an empty list should be returned.

Refer to ADS 1.3.1 for threading requirement.
Refer to ADS 1.3.2 for persistence requirement.
Refer to ADS 1.3.3 for transaction requirement.
Refer to ADS 1.3.4 for configuration requirement.
Refer to ADS 1.3.5 for logging requirement.
Refer to ADS 1.3.8 for auditing requirement. Note that the passed-in userId method argument is used for auditing purpose.

1.3 Required Algorithms

None

1.4 Example of the Software Usage

This component provides the project metadata services.

1.5 Future Component Direction

None

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None

2.1.2 External Interfaces

Refer to the provided architecture TCUML.

2.1.3 Environment Requirements

- Development language: Java
- Compile Target: Java 1.5

2.1.4 Package

com.topcoder.direct.services.project.metadata

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- The validator implementation in the service
- The auditActionTypeId for the audit
- The common metadata key ids and value format validator mappings
- Logger and JPA EntityManager

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

JPA

3.2.2 TopCoder Software Component Dependencies:

- Base Exception 2.0
- Logging Wrapper 2.0

Custom:

- Project Metadata Entities 1.0

3.2.3 Third Party Component, Library, or Product Dependencies:

- Spring 2.5.6: http://www.springsource.org/
- Hibernate 3.6.4: http://www.hibernate.org/

3.2.4 QA Environment:

- Java 1.5
- Informix 11.5
- JBoss 4.0.2

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification
- DLL scripts for all the new tables for this design suited for Informix 11.5 (check provided ERD.mwb for new tables defined)