[TopCoder]

Registration Services 1.0 Requirements Specification

1. Scope

1.1 Overview

This component implements the business logic for managing registrations to contests. The main interface of this component is fine grained enough to provide a useful API to client applications, but coarse grained enough to offer transactional atomic services and allow the presentation layer to minimize the calls to this layer.

1.2 Logic Requirements

1.2.1 Register For Project

This Service captures the registration process for any type of competition, such as assembly, testing, and component. The focus is on the type of registration as opposed to the type of competition registering for.

In this version, registration as Free Agents and Team captains will be supported.

A user can already be registered for the selected project; this allows a user to alter their desired role for this project. For example, a user registered as a free agent may re-register for the project as a Team Captain. Doing so removes their free agent role and grants them the Team Captain role.

A user may only have one role at any time for a particular project; but the user may, subject to eligibility and rules, alter their role in a project.

This service returns whether the validation was successful, the previous registration if there was one and the error messages in case of a validation failure.

1.2.1.1 Validation

This Service should call the Validate Registration before performing the registration. If the validation failed, the registration process should finish and return the validation result. In that case, the previous registration return value will be null, even if there was one.

1.2.1.2 Perform the registration

The registration process consists in adding a new Resource to the selected Project with the specified Role.

These are the Resource fields that are mandatory

- **Project**: from the RegistrationInfo argument
- **Resource Role**: Team member or Team Captain (note that these are new Roles)
- **Custom Properties**: See below.

The following are the custom properties of the Resource that must be set and where to get their values from.

Map key	Value
External Reference ID	RegistrationInfo.userId
Handle	ExternalUser.handle
Email	ExternalUser.email
Registration Date	Now

If the member was already registered for the project, the old registration should be removed and returned as part of the return value.

1.2.2 Validate Registration

This service relies on the Registration Validation interface to validate the registration. The implementation to be used is a pluggable object and should be retrieved via the Configuration Manager component.

This service should gather all required objects to pass them to the RegistrationValidation but there's no registration validation logic in this component.

1.2.3 Retrieve Registration Info

This is a simple service that returns the registration info given a user ID and a project ID.

1.2.4 Find Available Registration Positions

This service retrieves, given a project category, all active contests in the registration phase along with the available roles for each project.

Project categories are Design, Development, Assembly, etc.

1.2.5 Remove Registration

Administrators must have the ability to remove registered members from a competition. This includes free agents, team members, team captains, reviewers, and coaches.

The authorization to perform this operation will be granted in an upper layer.

This operation should remove the registration using the ResourceManager and perform the removal of the team related data through the TeamServices removeMember() method.

1.2.6 Retrieve Registered Resources

This service retrieves, given a project id, all members registered to the competition.

1.2.7 Retrieve Registered Projects

This service retrieves, given a user id, all the projects the member is registered to.

1.2.8 Authentication

No authentication should be performed in this component. Authentication will be done in an upper layer.

1.2.9 Authorization

No authorizations should be performed in this component. Authorizations will be done in an upper layer.

1.2.10 Return values of Array type

For all methods/services returning arrays, in case there are no objects to fill the array it must return an empty one, never a null.

1.2.11 Access to the data model

This component uses existing lower level Entity Management components to execute the requests whenever necessary. It's required not to use the Data Access Objects directly since low level validations are performed in Entity Management layer. This layer comprises the following components:

- Resource Management 1.1
- Project Phases 1.1
- Project Management 1.1
- User Project Data Store 1.1

It's also allowed to use services form the Services Layer if necessary. The components in this layer are:

- Project Services
- Team Services
- Contact Member Services

The actual implementation of each Manager/Service must be pluggable.

1.2.12 Required Interface: RegistrationServices

In the Component Interface Diagram is defined the RegistrationServices interface. All registration services are provided through that interface. This component will contain it along with an implementation based on lower level components enumerated below.

1.2.13 Required Interface: RegistrationInfo

Is a DTO to move around the minimum data needed to perform a registration,

1.2.14 Required Interface: RegistrationPosition

Represent a project in the registration phase along with all available roles a member could register for.

1.2.15 Required Interface: RegistrationValidator

Implementations of this interface will perform all necessary validations required to perform a registration.

1.2.16 Required Interface: RegistrationResult

It's the result of a registration request.

1.2.17 Required Interface: RemovalResult

It's the result of a registration removal request.

1.2.18 Interface Implementations

This component must provide implementations for all required interfaces.

1.2.19 Exception Management

Services returning a Result object will not throw checked exceptions. All business related errors must be reported in the returning object for such methods.
The remaining services must add checked exceptions for business related errors.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

These services will be used by client applications to manage registration to competitions.

1.5 Future Component Direction

Registration to non-team competitions will be added, along with new services like Unregister to Project.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

2.1.2.1 Interfaces defined

This component must define and implement the interfaces depicted in the Component Interface Diagram inside the com.topcoder.registration.service package.

Value objects are Java Beans. Attributes must be private and they must provide getters and setters every of them. At least an empty constructor must be provided.

2.1.2.2 Interfaces used

For all non existing components, such as Project Services, the component design must follow the Component Interface Diagram as a reference and assume they will be available for development.

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.4

2.1.4 Package Structure,

com.topcoder.registration.services

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

Registration validations must be configurable. So are the implementations to be used of the following components, in case they are actually used.

- Resource Management
- Project Phases
- Project Management
- User Project Data Store
- Project Services
- Team Services
- Contact Member Services

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

Resource Management: 1.1
Project Phases: 1.1
Project Management: 1.1
Configuration Management: 2.1.5

3.2.3 Third Party Component, Library, or Product Dependencies:

None

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003

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.

3.3.1 EJB compliant

This component will be used from EJB objects, thus it must comply with EJB development restrictions (http://java.sun.com/blueprints/qanda/ejb_tier/restrictions.html).

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of Poseidon.