Team Services 1.0 Requirements Specification

1. Scope

1.1 Overview

This component implements all the business rules governing the process of building teams for competitions. It relies on a lower layer of components that manages teams, projects, resources and users. A team is a group of resources working for a project. It has basic info (a header), a set of filled or unfilled positions and a set of custom properties. A team can also be finalized. A finalized team cannot contain unfilled positions.
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 Interfaces

This component must define and implement the interfaces depicted in the Component Interface Diagram inside the com.topcoder.team.service package. The diagram will be provided.

1.2.2 Create or Update Team

This service receives a team header and “saves” it. It must decide whether the team is new or an existing one and call the proper method on the TeamManager interface.
It gives the ability for a client application to just “Save” a team header, ignoring whether it is a new team or an update to an existing one.

1.2.2.1 Team details cannot be edited once the team has been finalized.

1.2.2.2 Team names must be unique across all teams registered to currently active projects.

1.2.2.3 A Team name cannot be the same as an existing member’s handle, except if the member is the Team Captain for this team.

1.2.3 Retrieve Teams

This service returns all the teams registered for a given project Id.

1.2.4 Retrieve Team Members

This service returns all the Resources already assigned to positions for the given team Id, including the Team Captain.

1.2.5 Remove Team

This service eliminates an entire team and all its positions. It sends a notification message to all team members through the ContactMemberService.

1.2.5.1 Any team members of the dissolved team are changed to Free Agents.

1.2.5.2 Any pending offers from the Team Captain to Free Agents are expired

1.2.5.3 The text template of the notification message must be configurable.

1.2.6 Retrieve Team Position Details

This service returns, given a team Id, the team’s Positions along with the associated Resources if the position is filled.

1.2.7 Find Free Agents

This service returns all the registered members of a project with the “Free Agent” role. The Id of

this ResourceRole will be configurable.

1.2.8 Create or Update Position

This service receives a team Position and “saves” it. It must decide whether the Position is new or an existing one and call the proper method on the TeamManager interface.
It gives the ability for a client application to just “Save” a team Position, ignoring whether it is a new position or an update to an existing one.

1.2.8.1 Positions may not be created after the team has been finalized.

1.2.8.2 Position Name cannot be a member handle

1.2.8.3 Position Name cannot duplicate an existing position name within the team.

1.2.8.4 If the position is filled, the resource must be a member registered in the team’s project as Free Agent.

1.2.8.5 If the position is filled, the resource must not be a member of any other team for that project.

1.2.9 Retrieve Position

This service returns the Position given a position Id.

1.2.10 Remove Position

This service removes a team member from a team.

1.2.10.1 Only un-published and un-filled positions can be deleted.

1.2.10.2 Team must not be finalized to perform this operation.

1.2.11 Validate Finalization

This service performs all checking needed to make sure a finalization will not have business related problems.
If this validation is successful, any problem finalizing the team will result in a runtime error. The following are required validations to be performed. Additional validations may be added by the designer as needed to ensure the finalization will be successful.

1.2.11.1 The team must be associated with an active project

1.2.11.2 The calling user must be the team’s captain.

1.2.11.3 The project must be currently in the registration phase

1.2.11.4 All team members must be registered to the project as Free Agents

1.2.11.5 Team captain must be registered as a resource to the project with the Team Captain ResourceRole.

1.2.11.6 All positions must be already filled

1.2.11.7 Team must not be already finalized

1.2.12 Finalize Team

This service performs the actual finalization of the team and updates the resources appropriately following these rules.

1.2.12.1 It must first validate the finalization using the Validate Finalization feature

1.2.12.2 It must update the team header to reflect the finalized state

1.2.12.3 It must change the Team Captain's ResourceRole to "Submitter". The "Submitter" ResourceRole id must be configurable.

1.2.13 Send Offer

This service sends the given offer through the Offer Management component and sends a notification message to the destination member through the ContactMemberService.

1.2.13.1 The position must be published and unfilled

1.2.13.2 The project associated with the team must be in the registration phase

1.2.13.3 The project must be currently in the registration phase

1.2.13.4 If the sender is a team captain, the receivers must be free agents registered for the project.

1.2.13.5 If the sender is a free agent, the receiver must be the team's captain.

1.2.13.6 The text template of the notification message must be configurable.

1.2.14 Retrieve Offers

This service returns all the unanswered and not expired offers that the user has received.

1.2.15 Accept offer

This service updates the team position with the accepting user by setting the resource, the payment percentage and the filled status.
It also updates the offer to ACCEPTED status and sends a notification message to the offerer through the ContactMemberService.

1.2.15.1 The position must be unfilled.

1.2.15.2 The project associated with the team must be in the registration phase

1.2.15.3 The new payment percentage must not make the team total to exceed 100%

1.2.15.4 The text template of the notification message must be configurable.

1.2.16 Reject offer

This service updates the offer to "Free Agent Rejected" or "Team Captain Rejected" as appropriate.
It also sends a notification message to the destination member through the ContactMemberService.

1.2.17 Remove Member

This service removes a team member from a team and sends a notification message to the team captain through the ContactMemberService..

1.2.17.1 It only removes team members. Not team captains.

1.2.17.2 It must send a notification of the removal to the team captain.

1.2.17.3 If the team is finalized, the payment percentage of the removed member is distributed evenly along all members including the team captain.

1.2.17.4 The text template of the notification message must be configurable.

1.2.18 Force Finalization

This service forces the finalization of all not finalized teams. This service has fewer restrictions that the Finalize Team service so it must not call the Validate Finalization. Still, some restrictions apply for each finalization.

1.2.18.1 All unfilled position prize percentages will be split evenly among the registered team members.

1.2.18.2 Unfilled positions are deleted by the system.

1.2.18.3 All team members must be registered to the project as Free Agents

1.2.18.4 Team captain must be registered as a resource to the project with the Team Captain ResourceRole.

1.2.19 Authentication

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

1.2.20 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.

1.2.21 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.
The Entity Management layer comprises the following components:

- **Team Management**: Under development. See component interface diagram.
- **Resource Management**: Generic component. Available in the catalog.
- **Project Phases**: Generic component. Available in the catalog.
- **Project Management**: Generic component. Available in the catalog.
- **User Project Data Store**: Custom component. Provided

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

- Project Services
- Registration Services
- Contact Member Services

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

1.2.22 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.
See 2.1.2 External Interfaces on how to deal with exceptions from components currently under development.

1.2.23 Logging

This component must produce the appropriate logging information for tracing / debugging / production support.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

These services will be used by client applications to allow the registered users to build and manage the teams themselves.

1.5 Future Component Direction

Support for new team roles such as “Coach” will be added.

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.team.service.

OperationResult and ResourcePosition are Java Beans. The interfaces must include getters and setters for elements modeled as attributes. The instance variables of the implementations must be private. At least an empty constructor must be provided.

2.1.2.2 Under development interfaces

For using interfaces yet under development you must refer to the Component Interface Diagram provided and the following guidelines.

2.1.2.2.1 Offer Manager

All methods throw the runtime exception OfferManagerException or a subclass in case the operation cannot be performed. This component must not catch these exceptions.

2.1.2.2.2 Team Manager

All TeamManager methods will throw specific checked exceptions on any business related problem. Examples are trying to add a position to a nonexistent team or updating a payment percentage in such a way that the sum of all team payment is greater than 100%.
These exceptions must be all subclasses of TeamManagerException, subclass of the com.topcoder.util.errorhandling.BaseException.
If appropriate, this kind of problems must be added to the OperationResult objects as the end user might be able to fix them.
Runtime problems like component misuse, invalid arguments or database connection must be thrown as unchecked exceptions.

2.1.3 Environment Requirements

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

2.1.4 Package Structure,

com.topcoder.management.team – Main package

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- The actual implementation to be used of the following components must be configurable (if applicable).
  - Team Management
  - Resource Management
  - Project Phases
  - Project Management
  - Project Services
  - Registration Services
  - Contact Member Services
- ResourceRole Ids constants must be configurable.
- All notification messages text templates must be configurable.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

Configuration Manager
Team Management
Resource Management
Project Phases
Project Management
User Project Data Store
Project Services
Registration Services
Contact Member Services
Document Generator
Logging Wrapper

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).

3.3.2 Component Scalability

The component needs to be scalable. Running multiple instances in the same JVM or in multiple JVM's concurrently should not cause any problem.

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.