Liquid Portal Service Requirements Specification

1. Scope

1.1 Overview

This component will provide web service related to user and competition provisioning. The web service will be deployed as part of TopCoder's platform software (as part of the TopCoder Direct, i.e. cockpit application) and will leverage the same web services that Cockpit uses. The client stubs will be integrated into the clients Liquid Portal, which is a program management and support application for running a strategic initiative with TopCoder. The module will also provide a strategy for securing system to system communications and allowing the client system to act on behalf of a remote user.

1.2 Logic Requirements

The provided interface diagram should be followed, and designer should use stateless session bean (with both local and remote interfaces) to implement the provided web service.

The entities will be plain Java Beans with corresponding getters and setters.

An extra document containing the pseudo code for each requirement below is also provided to help understand how this component can be implemented. Detailed sequence diagrams are provided too.

1.2.1 Register User

Create the user in TopCoder site with the passed-in information.

1.2.1.1 Input

Data Element	Description	Required?
First Name	The first name of the user	Y
Last Name	The last name of the user	Y
Email address	A email address in the standard valid format.	Y
Password	Password for the user.	Y
Handle	TopCoder Handle user wants.	Y
Phone	User's phone number	Y
Terms Agreement	Time that user agreed to TopCoder terms of use	Y

1.2.1.2 Process

- The service will delegate to existing TopCoder API for creating handles.
- Add user to the Notus Observer Terms and Notus Eligibility groups.

1.2.1.3 Output

- An exception will be thrown if the handle was not created.
- A warning condition will be returned if the new user was not added to the Terms or Eligibility groups. This should also be logged to Jira for system admin support.
- If success, return the result with userId.

1.2.2 Validate User

Ensure the passed-in handle is valid, and it's in the correct groups.

1.2.2.1 Input

Data Element	Description	Required?
First Name	The first name of the user	Y

Last Name	The last name of the user	Y
Email address	A email address in the standard valid format.	Y
Handle	TopCoder Handle user wants.	Y
Force	Ignore warnings and add user to terms & eligibility groups.	Y

1.2.2.2 Process

- The service will use existing TopCoder APIs to retrieve the handle and certain other user information. The service will check that the First Name, Last Name, and Email Address all match the database information.
- Add user to the Notus Observer Terms and Eligibility groups. In the following situations.
  - All demographic information matches the request
  - Force is set to true

1.2.2.3 Output

- An exception will be thrown if the handle does not exist.
- A warning condition will be returned if any demographic information doesn't match (Email, First or Last Name)
- A warning condition will be returned if the user failed to be added to the Terms or Eligibility group
- If success, return an empty result (with no warnings).

1.2.3 Provision User

A provisioned user will be setup with the correct permissions to be able to create competitions as needed.

1.2.3.1 Input

Data Element	**Description**	**Required?**
Requestor Handle	The TopCoder handle of the user that requested the update	Y
User Handle	The handle of the user getting setup	Y
Has Account Access	If the user has account level access	Y
List of Cockpit Projects	All cockpit projects that the user has access to	Y (can be empty)
List of Billing Projects	List of (Time Tracker) billing projects that user can charge against when creating competitions	Y (can be empty)

1.2.3.2 Process

- Validate Handle and User
  - The service will confirm the Requestor and User Handle are valid
  - The service will confirm that both users are part of the Notus Eligibility group.
  - Requestor and User Handle may be the same.
- Add User to Cockpit Projects
  - For each cockpit project tied to the Notus Account – if the project is in the submitted list

add the user, if not remove the user.
    - If "Has Account Access" is true then add user to all projects.
- Add User to Billing Projects
    - For each Billing Project associated to the Notus Account – if the project is in the submitted list add the user, if not remove the user.
    - If "Has Account Access" is true then add user to all projects.

1.2.3.3 Output

- An exception will be thrown if either handle didn't validate or the changes couldn't be completed
- A warning condition will be returned if any cockpit or billing project submitted was not found or not part of the Notus account.
- If success, return the result with the retrieved billing projects and cockpit projects.

1.2.4 Create Competition

Users who have registered with the Liquid PMO, including creating or validating their TopCoder handle, and been given permissions to access billing projects can launch competitions.

1.2.4.1 Input

Data Element	Description	Required?
Requestor Handle	The TopCoder handle of the user that requested the update	Y
Contest Type	The contest type name. If its value is "studio", it means the contest is a studio contest, otherwise, the contest is a software competition, and this value means the project category name.	Y
Contest Subtype	The studio contest type name if it's a studio contest.	Required if the contest Type is "studio".
Contest Name	Name of Contest	Y
Cockpit Project	Name of Cockpit Project to associate contest to	Y
Requested Start Date	Earliest date contest should launch	Y
Automatically Reschedule	Is it okay to find an alternative date and time if capacity is full?	Y
Billing Project	The billing projects that contest will be associated with	Y
CCA	Should the contest be created with CCA terms	Y
List of Support Handles	List of handles that will perform contest support, such as forum monitoring	Y (can be empty)

1.2.4.2 Process

- Validate Permissions

- Validate Requestor Handle exists
- Validate Requestor is a member of Notus Eligibility Groups
- Validate Requestor is a member of the Billing Project
- Validate that if Cockpit Project exists, the requestor has permissions. To find the cockpit project lookup by project name and association with Notus Account

- Create Cockpit Project
  - If the cockpit project doesn't exist create the project using Cockpit Project as name and associating it with Notus Account
- Create Contest
  - Validate contest metadata (Type, Subtype, Name)
  - Create contest on requested date
  - If date is at capacity and Automatically Reschedule is true, then use the next available date
- Add Support Handles
  - For each support handle
    - Check handle exists
    - Check user is part of Notus Eligibility groups
    - Add handle to OR as observer (including forum permissions, file upload, and forum watch)

1.2.4.3 Output

- An exception will be thrown if
  - the permissions are invalid
  - Cockpit project creation failed
  - The contest cannot be created (including because date was unavailable and reschedule was not allowed)
- Warning should be returned if
  - Some or all of the support handles could not be added as observers in OR
  - Project had to be created
- If successful, return the result with created contest.

1.2.5 ***Provision Project***

When a new contest is created and it results in the creation of a cockpit project, the project may need to be provisioned with a list of users that can access the project.

1.2.5.1 Input

Data Element	Description	Required?
Requestor Handle	The TopCoder handle of the user that requested the update	Y
Cockpit Project	The project to provision	Y
List of Handles	User to be given full control of the cockpit project	Y

1.2.5.2 Process

- Validate Permissions
  - Validate Requestor Handle exists
  - Validate Requestor is a member of Notus Eligibility Groups
  - Validate that if Cockpit Project exists, the requestor has permissions. To find the cockpit project lookup by project name and association with Notus Account
  - Validate that the Requestor had permissions to the cockpit project
- Add Handles

- For each handle
  - Check handle exists
  - Check user is part of Notus Eligibility groups
  - Add handle to cockpit project with full control

1.2.5.3 Output

- An exception will be thrown if
  - the permissions are invalid
  - Cockpit project creation failed
  - The contest cannot be created (including because date was unavailable and reschedule was not allowed)
- Warning should be returned if
  - Some or all of the support handles could not be added as observers in OR
  - Project had to be created
- If successful, return an empty result (with no warnings).

1.2.6 Delete Competition

From time to time a user will realize that a competition will not (ever) be run and should be deleted.

1.2.6.1 Input

Data Element	Description	Required?
Requestor Handle	The TopCoder handle of the user that requested the update	Y
Contest ID	Contest to delete	Y
Is Studio Conteset	The contest is a studio contest or not.	Y
Reason	User provided reason for deletion	Y

1.2.6.2 Process

- Validate Permissions
  - Validate Requestor Handle exists
  - Validate Requestor is a member of Notus Eligibility Groups
  - Validate that contest exist
  - Validate that the user has permissions for the billing account associated to the competition
  - Validate that the user has permissions for the cockpit project associated to the competition
- Delete Contest
  - Set the contest OR status to deleted, and use <Requestor Handle>: <Reason> as the explanation.

1.2.6.3 Output

- An exception will be thrown if any validation fails or contest delete fails

1.2.7 Decommission User

From time to time a user may need to be removed from all Notus related settings.

1.2.7.1 Input

Data Element	Description	Required?

Requestor Handle	The TopCoder handle of the user that requested the update	Y
User Handle	User to decommission	Y

1.2.7.2 Process

- Validate Permissions
  - Validate Requestor Handle exists
  - Validate Requestor is a member of Notus Eligibility Groups
  - Validate User Handle exists
  - Validate User Handle is a member of Notus Eligibility Groups
- Remove User from Cockpit Projects
  - For each cockpit project that is associated with Notus Account remove user if the user has permissions
- Remove User from Billing Accounts
  - For each billing account that is associated with Notus Account remove user if the user has permissions
- Remove User from Eligibility Groups and Terms
  - Remove user from Notus Eligibility groups
  - Remove user from Notus Observer Terms

1.2.7.3 Output

- An exception will be thrown if any validation fails or any removal fails

1.2.8 Logging

The method entry and exit should be logged with INFO level, and the exception should be logged with ERROR level.

1.2.9 Transaction

Bean managed transaction should be used.
If the EJB method changes the data in persistence, it should use a transaction, otherwise no transaction is necessary.

1.2.10 Security

The stateless session bean should be decorated with the following notations:
@DeclareRoles({"Cockpit User", "Cockpit Administrator" })
@RolesAllowed({"Cockpit User", "Cockpit Administrator" })

1.2.11 Configuration

The Configuration API and Configuration Persistence components should be used to load the configuration parameters.

1.2.12 Thread Safety

This component should work thread-safely after being deployed in the container, as it's implemented as a stateless session bean.

1.2.13 JIRA Logging

If the newly created user cannot be added into the terms and groups, the error needs to be logged to JIRA. Generally an email about the error should be sent to the JIRA. And then the email will be processed by the handle in the following URL:
http://confluence.atlassian.com/display/JIRA/Services#Services-BuiltinServices

1.2.14 Database Information

ip: 174.129.64.230

Servername: informixoltp_tcp
Port: 2021
Username: informix
Password: 1nf0rm1x

You can connect it by http://www-01.ibm.com/software/data/informix/serverstudio/
or any tools which support Informix (11.5)

1.3 Required Algorithms

None

1.4 Example of the Software Usage

1.5 Future Component Direction

None

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None

2.1.2 External Interfaces

See Interface diagram. Any change to the interface should be approved by the PM in the forum.
Liquid_Portal_Service_pseudo_code.rtf

2.1.3 Environment Requirements

- Development language: Java1.5
- Compile target: Java1.5, Java 1.6

2.1.4 Package Structure

com.liquid.portal.service
com.liquid.portal.service.bean

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

Notus Observer Terms Id
Notus Eligibility Group Ids
Full Control Permission Type Id
Mappings from contest type name to project category id (for software competition)
Mappings from sub contest type name to contest type id (for studio contest)
Inactive Status Id for newly created studio contest
Inactive Status Id for newly created software competition
Deleted Status Id for deleted software competition
Notus Client Id (for Notus Account)
Billing Account Property Key for created software competition
Delete Reason Property Key for deleted software competition

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

EJB 3.0

JPA 1.0

Hibernate 3.2 and higher

3.2.2 TopCoder Software Component Dependencies:

Base Exception 2.0
Logging Wrapper 2.0
Email Engine 3.1.0
Configuration API 1.0
Configuration Persistence 1.0
Client Project Entities DAO 1.1
Project Service 1.1
Contest Service Façade 1.0
Pipeline Service Façade 1.0
Contest and Submission Entities 1.0

3.2.3 Third Party Component, Library, or Product Dependencies:

None

3.2.4 QA Environment:

- RedHat Linux 9
- Informix 10
- JBoss 4.2

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

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the ‘Documentation’ tab of the TopCoder UML Tool.