This page last changed on Aug 16, 2010 by hohosky.

# Copilot Pool and Profile Services Requirements Specification

## Scope

### Overview

TopCoder is revamping the current copilot selection process to build a more structural one. One big change is that we will bring in the idea of the copilot pool which is consisted of authorized pre-approved members; only members in the copilot pool will have access to the new copilot opportunities (the new copilot opportunities will be posted as copilot selection contests by client/PM). And each copilot will have copilot profile pages which show the statistics of his past copilot performance, such as number of projects managed, number of contests managed, the number of current working on projects and past project histories etc. The copilot profile will be used by the client/PM as a reference when choosing a copilot.
This component is responsible for defining the services for the copilot pool and profile module.

#### Version

1.0

### Logic Requirements

The architecture TCUML will be provided, and this component is responsible for the classes/interfaces defined in the com.topcoder.direct.services.copilot and com.topcoder.direct.services.copilot.dto packages.
The services in this component will be implemented as POJOs (plain old java objects).

#### GenericService

This interface defines the generic CRUD methods to manage the entity into the persistence. It contains the following methods:

- create - create a new record into persistence for the entity and return the generated entity id.
- update - update the entity back to the persistence.
- delete - delete the entity from the persistence by entity id.
- retrieve - retrieve the entity from the persistence by entity id.
- retrieveAll - retrieve all the entities from the persistence.

In the concrete service implementation, all the methods above should simply delegate to the corresponding method in the GenericDAO<T>.

#### CopilotProfileService

This interface simply extends the GenericService interface to manage the copilot profile.
All methods from GenericService should be implemented.
The additional getCopilotProfile(userId) method simply delegate to the corresponding method in CopilotProfileDAO.
It also contains the following methods.

##### *getCopilotPoolMembers*

This method retrieves all the members in the copilot pool.

It should call CopilotProfileDAO.retrieveAll method to get a list of CopilotProfile objects first. Then for each CopilotProfile object, a CopilotPoolMember object should be populated.
The CopilotPoolMember object's attributes should be populated as below:

- copilotProfile attribute - The retrieved CopilotProfile object

```
// get all copilot projects for the copilot
copilotProjects = CopilotProjectDAO.getCopilotProjects(CopilotProfile.id)
```

- totalProjects - copilotProjects.size()
- currentProjects - count the number of CopilotProject in copilotProjects where its status is Active

```
// get the contests for the copilot project
Foreach copilotProject in copilotProjects
contestIds = UtilityDAO.getCopilotProjectContests(CopilotProfile.userId, CopilotProject.tcDirectProjectId)
EndFor
Collect the contestIds for each copilotProject above into a contestIdList list.
// get contests' details
contests = ProjectManager.getProjects(contestIdList)
```

- currentContests - count the number contest in contests where Project.projectStatus is 1 Active
- totalFailedContests - Count the number of contest in contests where Project.projectStatus is one of the values below:
  4 Cancelled - Failed Review
  5 Cancelled - Failed Screening
  6 Cancelled - Zero Submissions
  10 Cancelled - Requirements Infeasible
  11 Cancelled - Zero Registrations

```
totalReposted = 0;
For each contestId in contestIdList
projectLinks = ProjectLinkManager.getSourceProjectLinks(contestId)
If the returned projectLinks contains a ProjectLink with ProjectLinkType = ProjectLinkType.REPOST_FOR,
// this contest is reposted
totalReposted ++
EndIf
EndFor
```

- totalRepostedContests - totalReposted
- totalContests - contests.size() - totalReposted (the reposted contests are not counted)

```
totalBugs = 0
For each contestId in contestIdList
totalBugs += UtilityDAO.getContestBugCount(contestId)
EndFor
```

- totalBugraces - totalBugs

### *getCopilotProfileDTO*

This method retrieves the copilot profile DTO.
It calls CopilotProfileDAO.getCopilotProfile(userId) to get the CopilotProfile object first, and then creates the CopilotProfileDTO object and populates it. The attributes extended from the CopilotPoolMember will be populated in the same way as above, and the other attributes should be populated as below:

- earnings - UtilityDAO.getCopilotEarnings(CopilotProfile.userId)

Group the contests (the same contests retrieved in last requirement) by the Project.projectCategory, and each group will contain a list of contests with the same ProjectCategory value. The key of the contestTypeStats Map is the projectCategory's name.
Foreach group
Create a ContestTypeStat object, and its attributes should be populated as below

- 
    - projectCategoryId and projectCategoryName will be retrieved from the projectCategory of this group.
    - realContests - the number of contests in group
    - repostedContests - the number of reposted contests in this group (use the same algorithm above)
    - failedContests - the number of failed contests in this group (use the same algorithm above)

EndFor
// get planned contests
Foreach copilotProject in copilotProjects
projectPlan = CopilotProjectPlanDAO.getCopilotProjectPlan(copilotProject.id)
Collect projectPlan.plannedContests into a plannedContests list
EndFor
Group the planned contests by the projectCategoryId, and each group will contain a list of PlannedContest objects with the same projectCategoryId value.
Foreach group
Get the corresponding ContestTypeStat object created above.

- 
    - plannedContests - the number of planned contests in the group.

EndFor

### CopilotProjectService

This interface simply extends the GenericService interface to manage the CopilotProject entity.
All methods from GenericService should be implemented.
It also contains the following methods.

#### *getCopilotProjects*

This method retrieves all the copilot projects for a copilot.
It should call CopilotProjectDAO.getCopilotProjects(copilotProfileId) method to get a list of CopilotProject objects first. Then for each CopilotProject object, a CopilotProjectDTO object should be populated.
The CopilotProjectDTO object's attributes should be populated as below:

- copilotProject attribute - The retrieved CopilotProject object
- totalPlannedContests - Get the corresponding CopilotProjectPlan from CopilotProjectPlanDAO, and then count all the planned contests.
- totalRealContests - if the CopilotProject's wholeProject is true, all the contests (excluding the reposted ones) in the corresponding TC Direct project should be counted, otherwise, only the associated contests are counted.
- totalRepostedContests - the number of reposted contests.
- totalFailedContests - the number of failed contests.
- totalRealBugraces - the number of bug races for all the contests in this copilot project
- plannedDuration (in days) - corresponding CopilotProjectPlan.plannedDuration
- plannedBugraces - corresponding CopilotProjectPlan.plannedBugraces
- realDuration - the number of days between startDate & endDate
    - startDate - retrieve the min start date of all the contests in this copilot project
    - endDate - if the copilot project is active, retrieve the max end date of all the contests in this copilot project; otherwise, simply use the CopilotProject.completeDate.
    - The PhaseManager should be used to retrieve the startDate & endDate of the contest, and the UtilityDAO.getContestLastestBugResolutionDate is used to get the latest resolution date for the bugs in each contest.
- contestTypeStats - the contest statistics for each project category.

Most fields here will be populated in the same way as those mentioned in 1.2.2.1.

#### *getCopilotProjectDTO*

This method retrieves the copilot project DTO.
It calls CopilotProjectDAO.retrieve(copilotProjectId) to get the CopilotProject object first, and then creates the CopilotProjectDTO object and populates it in the same way as described above.

### CopilotProjectPlanService

This interface simply extends the GenericService interface to manage the copilot project plan entity.
All methods from GenericService should be implemented.
It contains the following additional method:

- getCopilotProjectPlan - get copilot project plan associated with the given copilot-project-id. Simply delegate to the corresponding method in the corresponding DAO.

### LookupService

This interface defines the following 3 methods:

- getAllCopilotProfileStatuses - get all the copilot profile statuses.
- getAllCopilotProjectStatuses - get all the copilot project statuses.
- getAllCopilotTypes - get all the copilot types.

All methods will simply delegate to the corresponding method in the corresponding DAO.

### Configuration

Spring framework should be used to load configuration.

### Transaction

Any method that modifies (creates, deletes or updates) the persistence data should be transactional.
Spring declarative transaction should be used, and the designer should use the @Transactional annotation for the transactional methods.

### Logging

The method entry and exit need to be logged with DEBUG level, and the exception needs to be logged with ERROR level.
The TCS Logging Wrapper should be used to do the logging.

### Exception

A base exception (CopilotServiceException) is already provided for this component. Designers are free to add any custom exception extending the base exception.

### Thread-safety

This component should work thread-safely.

## Required Algorithms

None

## Example of the Software Usage

This component provides the services for the copilot pool and profile module.

## Future Component Direction

None

# Interface Requirements

### Graphical User Interface Requirements

None

### External Interfaces

None

### Environment Requirements

- Development language: Java1.5
- Compile target: Java1.5
- JBoss 4.0.2
- Informix 11.0

### Package Structure

com.topcoder.direct.services.copilot
com.topcoder.direct.services.copilot.dto

# Software Requirements

## Administration Requirements

### What elements of the application need to be configurable?

- Active Contest Status Id
- Failed Contest Status Ids
- Active Copilot Project Status Id

## Technical Constraints

### Are there particular frameworks or standards that are required?

None

### TopCoder Software Component Dependencies:

Logging Wrapper 2.0
Base Exception 2.0
Project Management 1.0.1
Project Phases 2.0.1
Phase Management 1.0.4
Copilot Pool and Profile DAO 1.0

### Third Party Component, Library, or Product Dependencies:

*Spring 2.5.6

**QA Environment:**

- Java 1.5
- JBoss 4.0.2
- Informix 11.0

## Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines.

## Required Documentation

### Design Documentation

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

### Help / User Documentation

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