TopCoder Security Groups Frontend Part 1 1.0 Component Specification

1. Design

TopCoder is a company which administers contests in computer programming and provides software development services to other companies using a component based development approach. The development process is managed using a series of tools, known as the Topcoder competition platform. The tools empower customers and client employees to directly take advantage of the competition platform to develop quickly and efficiently their software projects.

From time to time TopCoder updates different applications composing the competition platform in order to meet client needs or to make the development process more efficient.

This project addresses one of these enhancement requests, namely management of client overall account in the entire platform. The client account will include all client assets as well as arbitrary groups of users. For example, a client has many projects that are created in Cockpit and each of those projects has multiple contests. Each customer, project and contest also has a set of users that need to be aware of them.

This front-end module will provide all the necessary pages and corresponding front-end actions for all the new user interfaces.
This component implements JSPs and Struts 2 Actions related to ARS 2.10 – 2.14.

1.1 Design Patterns

1.1.1 Strategy

The action classes use implementations of group services, authorization service and so which are pluggable.
In addition, the actions of this component are also used strategically by the Struts framework.

1.1.2 DTO

The entities like Group are the DTOs used by this component.

1.1.3 MVC

Actions can be treated like the Controller part of the MVC pattern.

1.1.4 IoC

The configuration is done through Spring injection. Therefore the Inversion of Control (IoC) pattern is used.

1.2 Industry Standards

XML
JSP
HTML
HTTP

1.3 Required Algorithms

Please refer to TCUML method doc for details not listed below.

1.3.1 Logging

The component will log activity and exceptions using the Logging Wrapper in struts actions(com.topcoder.shared.util.logging.Logger).

It will log errors at Error level, and method entry/exit information at DEBUG level.

Specifically, logging will be performed as follows, if logging is turned on.

- Method entrance and exit will be logged with DEBUG level.
  - Entrance format: [Entering method {*className*.*methodName*}]
  - Exit format: [Exiting method {*className*.*methodName*}]. Only do this if there are no exceptions.
- Method request and response parameters will be logged with DEBUG level
  - Format for request parameters: [Input parameters[{request_parameter_name_1}:{ request_parameter_value_1}, {request_parameter_name_2}:{ request_parameter_value_2}, etc.)]]
  - Format for the response: [Output parameter {response_value}] . Only do this if there are no exceptions and the return value is not void.
  - If a request or response parameter is complex, use its toString() method. If that is not implemented, then print its value using the same kind of name:value notation as above. If a child parameter is also complex, repeat this process recursively.
- All exceptions will be logged at ERROR level, and automatically log inner exceptions as well.
  - Format: Simply log the text of exception: [Error in method {*className*.*methodName*}: Details {*error details*}]

In general, the order of the logging in a method should be as follows:

1. Method entry
2. Log method entry
3. Log method input parameters
4. If error occurs, log it and skip to step7
5. Log method exit
6. If not void, log method output value
7. Method exit

1.3.2 Error Handling

All the action methods will throw java.lang.Exception if any error caught. And then the front end module should make an error page, that is shown when there is any error. The error page should show error message and exception stack trace.

1.3.3 Validation

XML validation is used for validations. Note that the XML validation should only apply to the the methods that take input from the front end page.. Therefore the validation rule should be specified in the files with name “ActionName-alias-validation.xml” where “Action alias” refers to each action name with method names such as"createGroup", rather than the action class name.

Because there are a lot of action aliases, this design only gives instructions on how to write the validation XML according to the rules in ARS (Application Requirement Specification), and the developers should write the actual validation XML based on instructions given here (the full tutorial of the Struts2 XML validation can be found here).

For input properties that has “Y” in “R?” column in ARS, use “requiredstring” validator like the following example:

```
<field-validator type="requiredstring">
    <param name="trim">true</param>
    <message>${getText("username.missing")}</message>
</field-validator>
```

For checking the maximal length of a string property, use “stringlength” validator like the following example:

```
<field-validator type="stringlength">
    <param name="maxLength">20</param>
    <message>${getText("username.length")}</message>
</field-validator>
```

The validation rule of the primitive fields is specified in the variable document of the struts actions. For the bean fields of the actions, they should be validated programmatically. See the validation rule in the below:

SearchGroupAction will validate the GroupSearchCriteria in the validate method against the following table:

Data Element	Description	Format	R?
Group Name	The name of the group.	String, max 45 chars, can be empty.	N
Customer Name	The name of the customer.	String, max 45 chars, can be empty.	N
Project Name	The project Name of the group.	String, max 45 chars, can be empty.	N

Access Rights	The access right of the user to the group.	Multiple selections.
		The values can be:
		Full;
		Read;
		Write;
			N

Account Name	The billing account name of the group.	String, max 45 chars, can be empty.	N
Member Name	The member name of the group.	String, max 45 chars, can be empty.	N

CreateGroupAction and UpdateGroupAction will validate the Group in the validate method against the following table:

Data Element	Description	Format	R?
Group Name	The name of the group.	String, max 45 chars, non empty.	Y

Group Authorization Attributes	The access rights allowed for the group.	Single Selection.
		The values can be:
		Full;
		Read;
		Write;
			Y

Customer Name	The name of the customer.	String, max 45 chars, non empty. Single Selection.	Y
Billing Accounts	The billing accounts of the group.	Multiple sections.	Y
Projects	The projects of the group.	Multiple sections.	Y

Data Element	Description	Format	R?
Resource Restrictions	The restricted resources of the group.	Multiple sections. The options can be: Billing Accounts; Projects	Y

CreateGroupAction and UpdateGroupAction will validate the GroupMembers in the validate method against the following table(please be noted that only if the member handle or email address is not empty, the GroupMember will be validated against the following table):

Data Element	Description	Format	R?
Member Handle/Email Address	The handle or email address of the member.	String, max 45 chars, non empty. The administrator can choose member by search. Please refer chapter 2.10.2 for details.	Y
Group User Authorization Attributes	The Authorization Attributes for the user.	Single Selection. The values can be: Group Default; User Specific.	Y
Access Level	The access level of the member.	Single Selection. This element is required if the “Group User Authorization Attributes” option is “User Specific” The values can be: Full; Read; Write;	N

1.3.4 Message Key and Resource Bundle

Since the developers are responsible for writing the actual XML validation files, they are also responsible for defining the message key for each validation error message, as well as the resource bundle that provides the message text.

1.3.5 @PostConstruct

The method that is marked with <<@PostConstruct>> in TCUML should be set as the value of the “init-method” attribute in the bean declaration of the Spring application context file. These methods don’t need to be added with the javax.annotation.PostConstruct.

1.3.6 Audit

The createGroup, updateGroup and deleteGroup methods should perform the auditing. The previousValue, newValue and handle fields of the audit record DTO will be retrieved as following:

Check the com.topcoder.direct.services.view.action.contest.launch.DirectStrutsActionsHelper.getTCSubjectFromSession(). And get the the TCPrincial from the return TCSubject. The handle will be TCPrincipal.getUserName().

For the createGroup, the previousValue will be null, and the newValue will be the groupId + groupName.
For the updateGroup, the previousValue will be groupId+groupName of the old group, and the newValue will be groupId+groupName of the new group.

For the deleteGroup, the previousValue will be groupId+groupName, and the newValue will be null.

1.3.7 Authorization

Authorization is done in various action methods.
First current user id is retrieved using existing authentication related code, then rendered content is filtered according to user role. For this component, the authorization has nothing to do with the action methods of this component.

1.4 Component Class Overview

BaseAction
BaseAction is used to hold the common fields such as logger for the sub classes to use.
It's mutable and not thread safe. However the struts framework will guarantee that it will be used in the thread safe model.

CreateUpdateGroupAction
CreateUpdateGroupAction is used to provide common fields for the UpdateGroupAction and CreateGroupAction, it also provide common action methods such as enter group details for the update/create group action.
It will log events and errors.
It will audit the methods.
It's mutable and not thread safe. However the struts framework will guarantee that it will be used in the thread safe model.

SearchUserAction
SearchUserAction is used to perform the search user function
It will log events and errors.
It will audit the methods.
It's mutable and not thread safe. However the struts framework will guarantee that it will be used in the thread safe model.

DeleteGroupAction
DeleteGroupAction is used to perform the delete group function.
It will log events and errors.
It will audit the methods.
It's mutable and not thread safe. However the struts framework will guarantee that it will be used in the thread safe model.

SearchGroupAction
SearchGroupAction is used to perform the search group function.
It will log events and errors.
It will audit the methods.
It's mutable and not thread safe. However the struts framework will guarantee that it will be used in the thread safe model.

ViewGroupAction
ViewGroupAction is used to perform the view group function.
It will log events and errors.
It will audit the methods.
It's mutable and not thread safe. However the struts framework will guarantee that it will be used in the thread safe model.

SessionAwareBaseAction
SessionAwareBaseAction is used to represents the session aware action.
It's mutable and not thread safe. However the struts framework will guarantee that it will be used in the thread safe model.

UpdateGroupAction
UpdateGroupAction is used to perform the update group function.
It will log events and errors.
It will audit the methods.
It's mutable and not thread safe. However the struts framework will guarantee that it will be used in the thread safe model.

CreateGroupAction
CreateGroupAction is used to perform the create group function.
It will log events and errors.
It will audit the methods.
It's mutable and not thread safe. However the struts framework will guarantee that it will be used in the thread safe model.

1.5 Component Exception Definitions

1.5.1 Custom exceptions

SecurityGroupsActionConfigurationException
SecurityGroupsActionConfigurationException is thrown by the action methods if any configuration error is caught.
This class is not thread safe because its base class is not thread safe.

SecurityGroupsActionException
SecurityGroupsActionException is thrown by struts actions if any error is caught during the action operation.
This class is not thread safe because its base class is not thread safe.

SecurityGroupsActionValidationException
SecurityGroupsActionValidationException is thrown by action validate methods if any error is caught during the validation
This class is not thread safe because its base class is not thread safe.

1.6 Thread Safety

Although individual classes in this component are not thread-safe as per 1.4, this component is effectively thread-safe to use under Struts framework because dedicated instances of struts actions will be created by the Struts framework to process each user request, and because the struts action instances will not be changed after initialization, and finally the services interfaces used by the actions are all thread-safe to use.

2. Environment Requirements

2.1 Environment

Development language: Java1.6 J2EE 1.5

Compile target: Java1.6 J2EE1.5

2.2 *TopCoder Software Components*

- Logging Wrapper 2.0 – used for logging.
- Security Manager 1.1 – provides TC security entities, used to get user handle for auditing purposes.
- DirectStrutsActionsHelper class from TopCoder cockpit code base:

  https://coder.topcoder.com/tcs/clients/cronos/applications/direct/trunk/src/java/main/com/topcoder/direct/services/view/action

2.3 *Existing Module*

The existing TopCoder website code.

2.4 *Third Party Components*

Spring 2.5.6 (http://www.springsource.org/download)

Struts 2.1.8 (http://struts.apache.org/download.cgi)

3. Installation and Configuration

3.1 *Configuration Parameters*

Configuration for the BaseAction

Parameter	Description	Value
logger	Logger used to perform logging as per CS 1.3.1. If null, logging will not be performed.	Logger Optional.
auditService	Audit service used to perform auditing for all modifying operations.	GroupAuditService Required.
authorizationService	authorizationService is used to represents the authorization service.	AuthorizationService Required.

Configuration for SessionAwareAction(it inherits the configuration from BaseAction):

Parameter	Description	Value
session	session is used to represents the http session map injected.	Map Required.

Configuration for CreateUpdateAction(It inherits the configuration from SessionSessionAwareBaseAction).

Parameter	Description	Value
groupSessionKey	groupSessionKey is used to represents the group session key.	String Required. Not empty.
clientService	clientService is used to represents the client service.	ClientService Required.
billingAccountService	billingAccountService is used to represents the billing account service.	BillingAccountService. Required.
directProjectService	Represents the direct project service	DirectProjectService Required.
groupInvitationService	GroupInvitationService used to create group invitations.	GroupInvitationService

		Required.
groupService	GroupService used to manage group information.	GroupService Required.
customerAdministratorService	CustomerAdministratorService used to get manage customer administrators.	CustomerAdministratorService Required.
registrationUrl	Registration URL which will be added to invitations.	String Required. Not empty.
acceptRejectUrlBase	Base part of URL for accepting/rejecting invitations. Hyperlinks to such URL's will be added to invitations.	String Required. Not empty.
groupMemberService	It's used to get/update the group members.	GroupMemberService. Required
oldGroupMembersSessionKey	oldGroupMemebersSessionKey is used to represents the old group memebers session key.	String, Not empty. Required

The CreateGroupAction and UpdateGroupAction inherit the configuration from the CreateUpdateGroupAction.

Configurations for ViewGroupAction:

Parameter	Description	Value
groupService	GroupService used to manage group information.	GroupService Required.

The **SearchGroupAction** and **DeleteGroupAction** have the same configuration as ViewGroupAction and all of them inherit the configuration for BaseAction.

Configuration for SearchUserAction:
It inherits the configuration from the BaseAction.

Parameter	Description	Value
userService	userService is used to represents the user service.	UserService Required.

3.2 *Dependencies Configuration*

None.

3.3 *Package Structure*

com.topcoder.security.groups.actions

4. Usage Notes

4.1 *Required steps to test the component*

- Extract the component distribution.
- Follow Dependencies Configuration.
- Execute 'ant test' within the directory that the distribution was extracted to.

4.2 *Required steps to use the component*

Please see the demo.

4.3 *Demo*

The spring configuration is:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

    <!-- dependency beans must be defined in applicationContext.xml -->

  <bean id="updateGroupAction"
        class="com.topcoder.security.groups.actions.UpdateGroupAction">
        <property name="groupService" ref="groupService"/>
        <property name="groupSessionKey" value="groupSessionKey"/>
         <property name="groupMemberService" ref="groupMemberService"/>
        <property name="oldGroupMemebersSessionKey" value="oldGroupMemebersSessionKey"/>
        <property name="clientService" ref="clientService"/>
        <property name="customerAdministratorService" ref="customerAdministratorService"/>
        <property name="directProjectService" ref="directProjectService"/>
        <property name="groupInvitationService" ref="groupInvitationService"/>
        <property name="acceptRejectUrlBase" value="acceptRejectUrlBase"/>
        <property name="registrationUrl" value="registrationUrl"/>
        <property name="billingAccountService" ref="billingAccountService"/>
        <property name="logger" ref="logger"/>
        <property name="auditService" ref="auditService"/>
        <property name="authorizationService" ref="authorizationService"/>

        <!-- other properties here -->
  </bean>

  <bean id="createGroupAction"
        <property name="groupService" ref="groupService"/>
        <property name="groupSessionKey" value="groupSessionKey"/>
        <property name="clientService" ref="clientService"/>
        <property name="customerAdministratorService" ref="customerAdministratorService"/>
        <property name="directProjectService" ref="directProjectService"/>
        <property name="groupInvitationService" ref="groupInvitationService"/>
        <property name="acceptRejectUrlBase" value="acceptRejectUrlBase"/>
        <property name="registrationUrl" value="registrationUrl"/>
        <property name="billingAccountService" ref="billingAccountService"/>
        <property name="logger" ref="logger"/>
        <property name="auditService" ref="auditService"/>
        <property name="authorizationService" ref="authorizationService"/>

        <!-- other properties here -->
  </bean>

  <bean id="deleteGroupAction"
        class="com.topcoder.security.groups.actions.DeleteGroupAction">
        <property name="groupService" ref="groupService"/>
        <property name="logger" ref="logger"/>
        <property name="auditService" ref="auditService"/>
        <property name="authorizationService" ref="authorizationService"/>

        <!-- other properties here -->
  </bean>
  <bean id="searchGroupAction"
        class="com.topcoder.security.groups.actions.SearchGroupAction">
        <property name="groupService" ref="groupService"/>
        <property name="logger" ref="logger"/>
        <property name="auditService" ref="auditService"/>
        <property name="authorizationService" ref="authorizationService"/>

        <!-- other properties here -->
  </bean>

  <bean id="searchUserAction"
        class="com.topcoder.security.groups.actions.SearchUserAction">
        <property name="userService" ref="userService"/>
        <property name="logger" ref="logger"/>
        <property name="auditService" ref="auditService"/>
        <property name="authorizationService" ref="authorizationService"/>
        <!-- other properties here -->
  </bean>

  <bean id="viewGroupAction"
        class="com.topcoder.security.groups.actions.ViewGroupAction">
        <property name="groupService" ref="groupService"/>
        <property name="logger" ref="logger"/>
        <property name="auditService" ref="auditService"/>
        <property name="authorizationService" ref="authorizationService"/>

        <!-- other properties here -->
  </bean>
    <!-- put other configurations here -->

</beans>
```

The struts configuration is:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

<package name="security" namespace="/security" extends="struts-default">
  <!-- the test.jsp is just a place holder, developers should replace it with the real page name defined by the prototype -->
  <action name="createGroup" class="com.topcoder.security.groups.actions.CreateGroupAction" method="createGroup">
                <result name="SUCCESS">test.jsp</result>
  </action>
  <action name="enterGroupDetails" class="com.topcoder.security.groups.actions.CreateGroupAction"
method="enterGroupDetails">
                <result name="SUCCESS">test.jsp</result>
  </action>
  <action name="addMembers" class="com.topcoder.security.groups.actions.CreateGroupAction" method="addMembers">
                <result name="SUCCESS">test.jsp</result>
  </action>
    <action name="updateMember" class="com.topcoder.security.groups.actions.CreateGroupAction"
method="updateMember">
                <result name="SUCCESS">test.jsp</result>
  </action>

  <action name="deleteMember" class="com.topcoder.security.groups.actions.UpdateGroupAction" method="deleteMember">
                <result name="SUCCESS">test.jsp</result>
  </action>
  <!-- the config forUpdateGroupAction is similar to the CreateGroupAction -->


  <action name="viewGroup" class="com.topcoder.security.groups.actions.ViewGroupAction" method="execute">
                 <result name="SUCCESS">test.jsp</result>
  </action>

  <action name="deleteGroup" class="com.topcoder.security.groups.actions.DeleteGroupAction" method="execute">
                 <result name="SUCCESS">test.jsp</result>
  </action>

  <action name="searchUsers" class="com.topcoder.security.groups.actions.SearchUserAction" method="execute">
                 <result name="SUCCESS">test.jsp</result>
  </action>

  <action name="searchGroups" class="com.topcoder.security.groups.actions.SearchGroupAction" method="execute">
                 <result name="SUCCESS">test.jsp</result>
  </action>
</package>

</struts>
```

Prototype is provided for this component. The pages related to ARS 2.10-2.14 are in scope. These pages should be converted to JSP pages, and properly bound to the above actions.

SearchUserAction -> administrator-create-new-group.html, administrator-update-user-group.html, administrator-create-new-administrator.html, administrator-send-group-invitation.html
In those pages, when the users click search the users by handle, a window will popup to require the handle to input. Once the search button is clicked, this action will be invoked to search users.

DeleteGroupAction -> administrator-view-user-group-details.html, administrator-view-user-groups.html
In those groups, when the delete group button is clicked, this action will be invoked.

SearchGroupAction -> administrator-view-user-groups.html
In this page, when the search button is clicked, the search criteria will be populated and this action will be invoked to search the matching users.

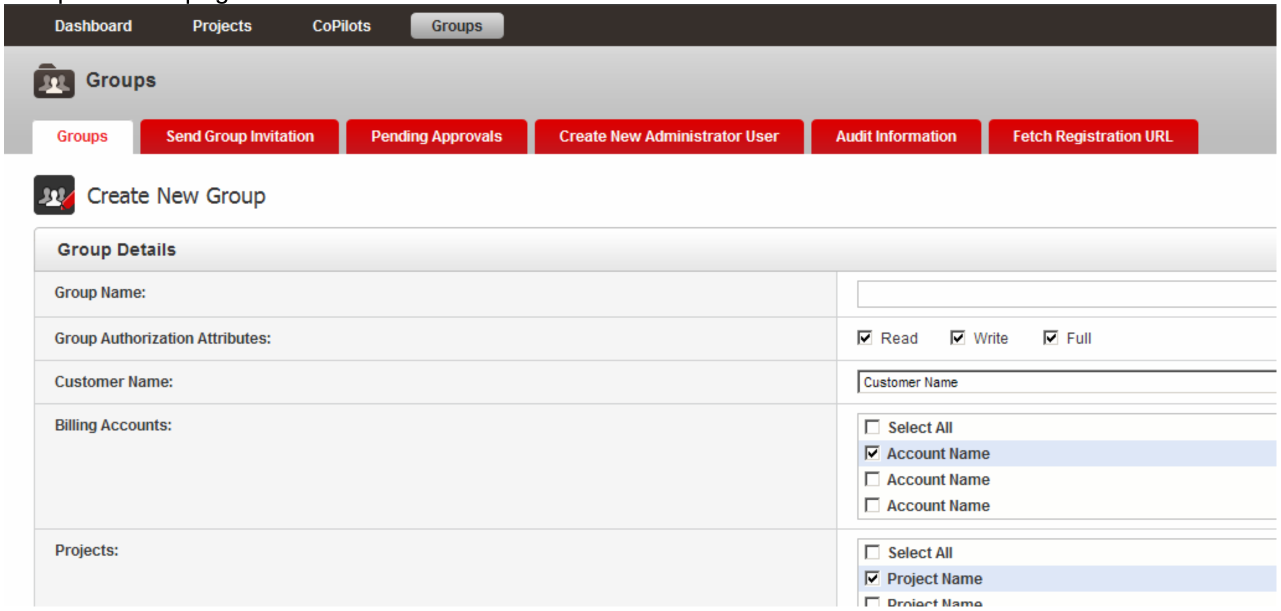
ViewGroupAction -> administrator-view-user-group-details.html
This action will be invoked to get the group details.

CreateGroupAction -> administrator-create-new-group.html
When the create group button is clicked, this action's createGroup method will be invoked to create the group. In the group members section of this page, when the remove button is clicked, the removeMember method of the action will be invoked. When the add more members button is clicked, addMemebers method will be invoked. When the update group member button is clicked, the updateGroupMemeber will be invoked. enterGroupDetails() method will be invoked when the users enter this page.

UpdateGroupAction -> administrator-update-user-group.html
This is similar to the CreateGroupAction discussion except that enterGroupDetails() of this action will provide a non-empty group for the users to update.

Here an example of create group.

The part of the page will be:



Group Name:	
Group Authorization Attributes:	☑ Read ☑ Write ☑ Full
Customer Name:	Customer Name
Billing Accounts:	☐ Select All ☑ Account Name ☐ Account Name ☐ Account Name
Projects:	☐ Select All ☑ Project Name ☐ Project Name

Please refer to the prototype for detail.

Suppose the user access http://localhost/createGroup, users input the necessary data to create the group, the struts framework will first validate the user input data for the group. And then CreateGroupAction will be called to save the group via GroupService.
.

5. Future Enhancements

None