TDS Admin Design Documentation

Smarter Balanced Assessment Consortium

Components: Test Delivery System

Smarter Balanced Task Order 17

# 

Table of Contents

[1 Overall Application Design](#h.2vf7xr8wgel6)

[2 Authentication with SSO](#h.f4mxtpkchqv0)

[3 Authorization with Permission](#h.1ztb2e1ezn9x)

[4 Integration with Program Management](#h.5b7fy94dd7y9)

[5 Integration with MNA](#h.9fybayex03vg)

[6 Standard Smarter Balanced Look](#h.v8ykyo7bgnwq)

[7 Fetch Opportunity](#h.626516rml21y)

[7.1 Sequence Diagram](#h.okr2qpusxjbr)

[7.2 Class Diagram](#h.kajtcyfhuylc)

[8 Opportunities in table with pagination, sorting and multi-select](#h.66kak8exf6uc)

[9 Set Opportunity Segment Permeability](#h.spc3vcpdg4wr)

[10 Restore Test Opportunity](#h.ccb8v3szvh04)

[11 Reopen a Test Opportunity](#h.26dfk9nhsu4w)

[12 Invalidate a Test Opportunity](#h.ls0qunlj1nyk)

[13 Extend an Opportunity's Grace Period](#h.yb2c5wn0dmcn)

[14 Alter an Opportunity's Expiration](#h.keciz9ocmywr)

[15 Reset a Test Opportunity](#h.wyh3q0x8or13)

[16 Show Result of Procedure Execution](#h.i2yz5zvzxqwl)

[17 Logout of Current Session for Security](#h.yu8w6a4xrpzm)

[18 Expire Session after 15 mins of Inactivity](#h.gad0ypz9qfzc)

[19 Deployment Instructions](#h.w77vtmvacu3v)

# 

# 

# 

# 

# 1 Overall Application Design

The application has two architectures for the user interface and the REST APIs separated by user authentication methodology. The user interface uses SAML while the REST API uses OAuth 2.0. The following two diagrams depicts the how the application works:

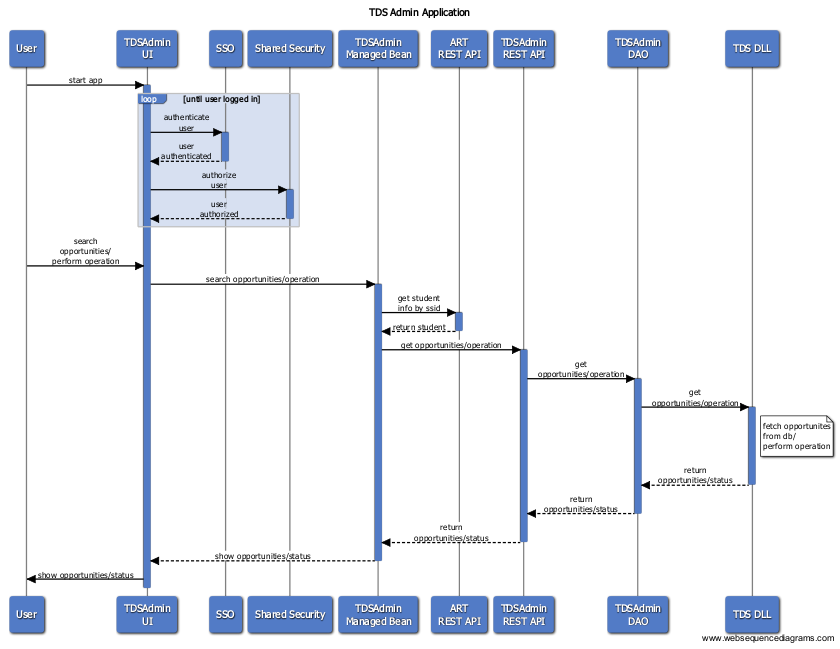
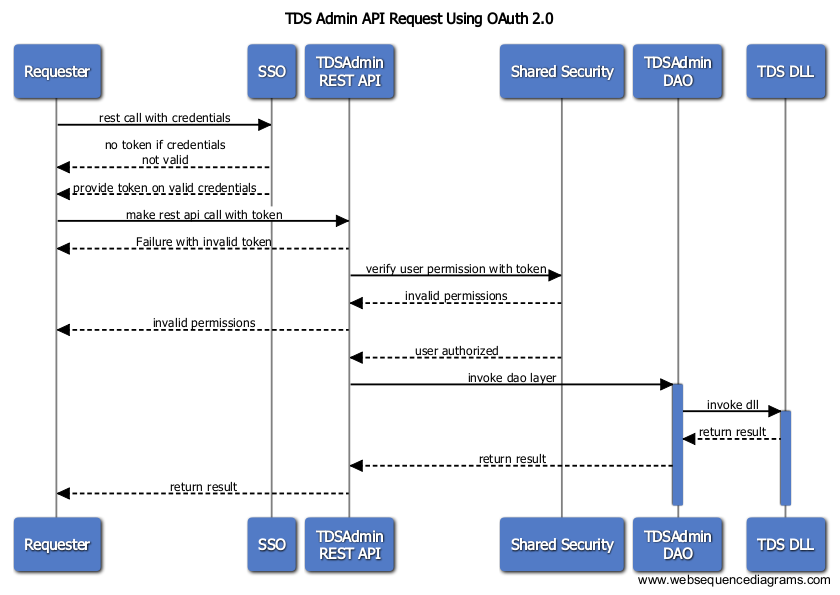


Figure 1: High level design for TDS Admin application with User Authentication

Figure 2: High level design for TDS Admin API Request with OAuth 2.0

# 2 Integration with Program Management

The application needs to be integrated with program management as it contains configuration with following parameter and their values:

Following are the steps to integrate Program Management with TDSAdmin:

1. Add following dependencies to pom.xml

<properties>

<prog.mgmnt.client.version>0.0.3-SNAPSHOT</prog.mgmnt.client.version>

<sb11-mna-client.version>0.0.4-SNAPSHOT</sb11-mna-client.version>

<sb11-shared-code.version>0.0.6-SNAPSHOT</sb11-shared-code.version>

</properties>

<!-- Start: Program management integration dependency -->

<dependency>

<groupId>org.opentestsystem.shared</groupId>

<artifactId>prog-mgmnt-client</artifactId>

<version>${prog.mgmnt.client.version}</version>

</dependency>

<dependency>

<groupId>org.opentestsystem.shared</groupId>

<artifactId>prog-mgmnt-null-client</artifactId>

<version>${prog.mgmnt.client.version}</version>

</dependency>

<!-- End: Program management integration dependency -->

<!-- Start: Monitoring and Alerting integration dependency -->

<dependency>

<groupId>org.opentestsystem.shared</groupId>

<artifactId>monitoring-alerting.client-null-impl</artifactId>

<version>${sb11-mna-client.version}</version>

</dependency>

<dependency>

<groupId>org.opentestsystem.shared</groupId>

<artifactId>monitoring-alerting.client</artifactId>

<version> ${sb11-mna-client.version}</version>

</dependency>

<dependency>

<groupId>org.opentestsystem.shared</groupId>

<artifactId>sb11-shared-code</artifactId>

<version>${sb11-shared-code.version}</version>

</dependency>

<!-- End: Monitoring and Alerting integration dependency -->

2. Add following entries in web.xml

<!--Start: Added for Program Management and Monitoring and Alerting dependency -->

<context-param>

<param-name>contextInitializerClasses</param-name> <param-value>org.opentestsystem.shared.mna.client.listener.ClientSpringConfigurator,org.opentestsystem.shared.progman.init.InitSpringPropertyConfigLoad</param-value>

</context-param>

<!--End: Added for Program Management and Monitoring and Alerting dependency -->

3. Create new configuration in prog-man. use the config name and enviornment in step 4 in property -Dprogman.locator="tdsadmin,{env}" (i.e. {end} is local in my computer)

4. Add following to your run configuration

-DSB11\_CONFIG\_DIR={directory having progman} -Dspring.profiles.active="progman.client.impl.integration,mna.client.integration" -Dprogman.baseUri=http://pm-dev.opentestsystem.org:8080/programmanagement.rest/ -Dprogman.locator="tdsadmin,local" -Djavax.net.ssl.trustStore="c:\Development\security\samlKeystore.jks" -Djavax.net.ssl.trustStorePassword="nalle123"

Here, DSB11\_CONFIG\_DIR is the directory which has the progman folder, it contains certificates ( \*\_local\_sp.xml, \*.properties etc.). Change truststore and baseUrl for respective systems.

5. To map variable to prog-man properties use following annotation

@Value(value="${mna.logger.level:}")

6. Add progman-loader-config-props-context.xml file to the project's resource/spring folder. You can copy the file from Permission or any other project already configured with prog-man

# 3 Authorization with Permissions

User authorization is done through a REST API call to permissions, the URL of the API is provided in the program management configuration. This is done by Shared Security module.

# 4 Integration with MNA

The following steps are required for integration with monitoring and alerting application:

1) Follow all the steps as shown above for program management.

2) Add logback.xml file inside your project's resource folder and add following. Change the value of app.base.package.name and app.context.name inside logback.xml file

<configuration scan="true">

<property scope="local" name="app.context.name" value="tdsadmin" />

<property scope="local" name="app.base.package.name" value="tds.tdsadmin" />

<property scope="local" name="mna.appender.active" value="true" />

<include resource="logback-included-common-config.xml" />

</configuration>

# 5 Authentication with Single Sign On

Integration with program management and monitoring & alerting modules is required before single sign can work. Assuming they are already integrated, following are the steps to add single sign on capability with TDSAdmin:

1. Copy security folder from src/main/resource of Permission application and paste it into the project's src/main/resource

2. Open samlmetadata-context.xml and change the properties starting with ${} to your project's configuration in progman. i.e. if we are copying from permission to tdsadmin, then replace ${permission.security.saml.keystore.cert} with ${tdsadmin.security.saml.keystore.cert} (This is something should be mapped to your prog-man config)

3. Open securityContext.xml and follow the same steps as above(2).

4. Inside pom.xml add following dependency.

<properties> <org.springframework.security-version>3.2.9.RELEASE</org.springframework.security-version>

</properties>

<!-- Start: SSO integration dependency -->

<dependency>

<groupId>org.opentestsystem.shared</groupId>

<artifactId>sb11-shared-security</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-core</artifactId>

<version>${org.springframework.security-version}</version>

</dependency>

<dependency>

<groupId>xerces</groupId>

<artifactId>xercesImpl</artifactId>

<version>2.10.0</version>

</dependency>

<!-- End: SSO integration dependency →

5. Add following filter mapping into web.xml

<!--Start: Following filter is added for integrating SSO -->

<filter>

<filter-name>springSecurityFilterChain</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

</filter>

<filter-mapping>

<filter-name>springSecurityFilterChain</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!--End: Following filter is added for integrating SSO →

6. Add following entries in the beginning of the spring context xml file of your project

<!-- Start : Added for SSO Integration -->

<beans:import resource="classpath:security/securityContext.xml"/>

<beans:import resource="classpath:security/samlmetadata-context.xml"/>

<!-- End : Added for SSO Integration -->

# 

# 6 Standard Smarter Balanced Look

This is done by the design team with additional inline CSS in Default.xhtml and two external CSS file tdsadmin.css and core.css. Used same images as other Smarter apps.

# 7 Fetch Opportunity

## 7.1 Sequence Diagram

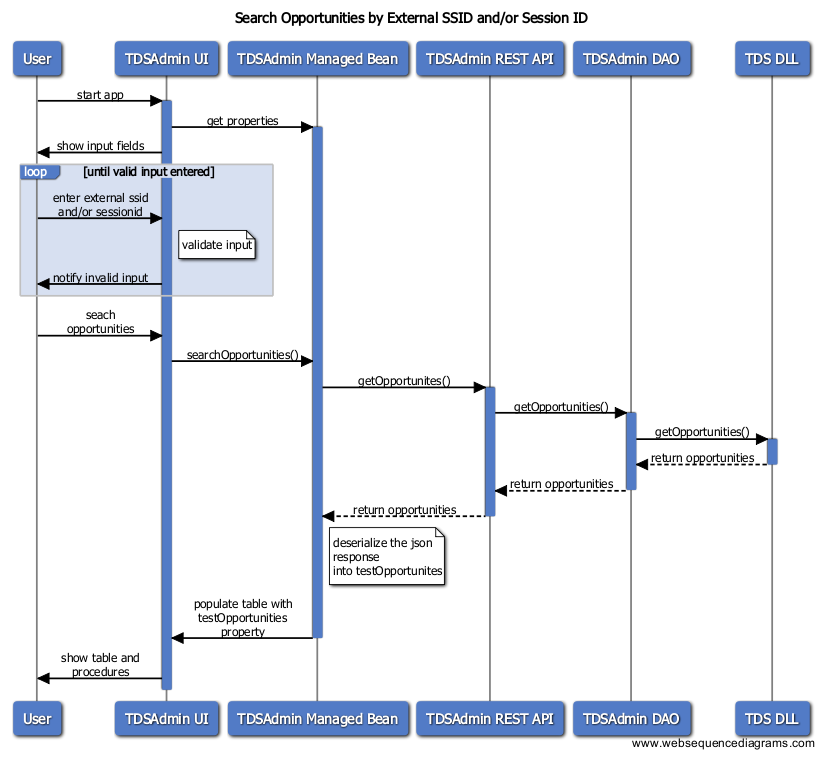


Figure 2: Search Opportunities by External SSID and/or Session ID

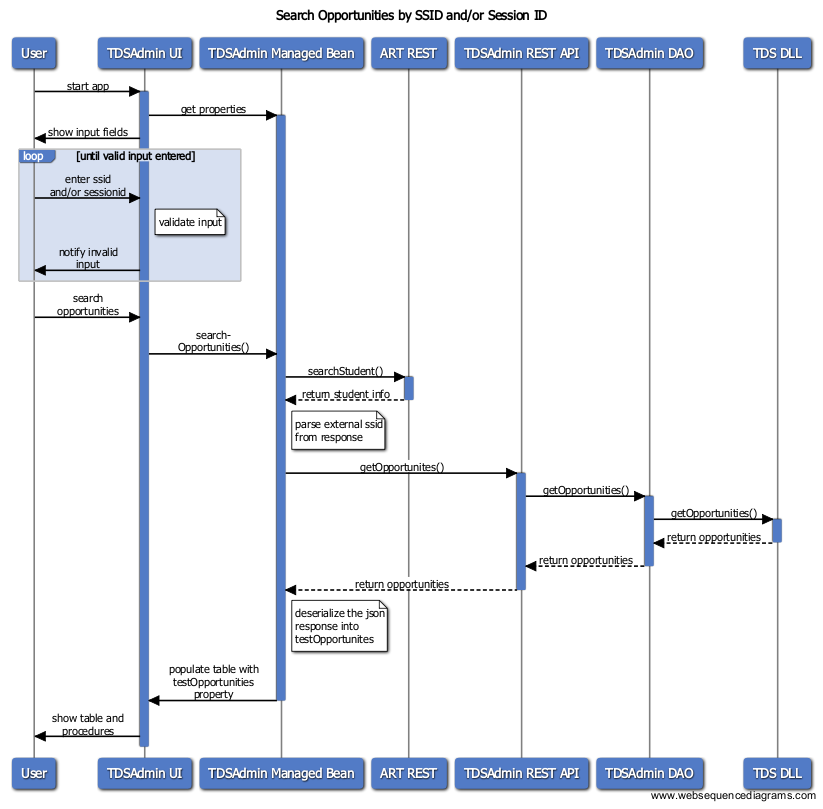


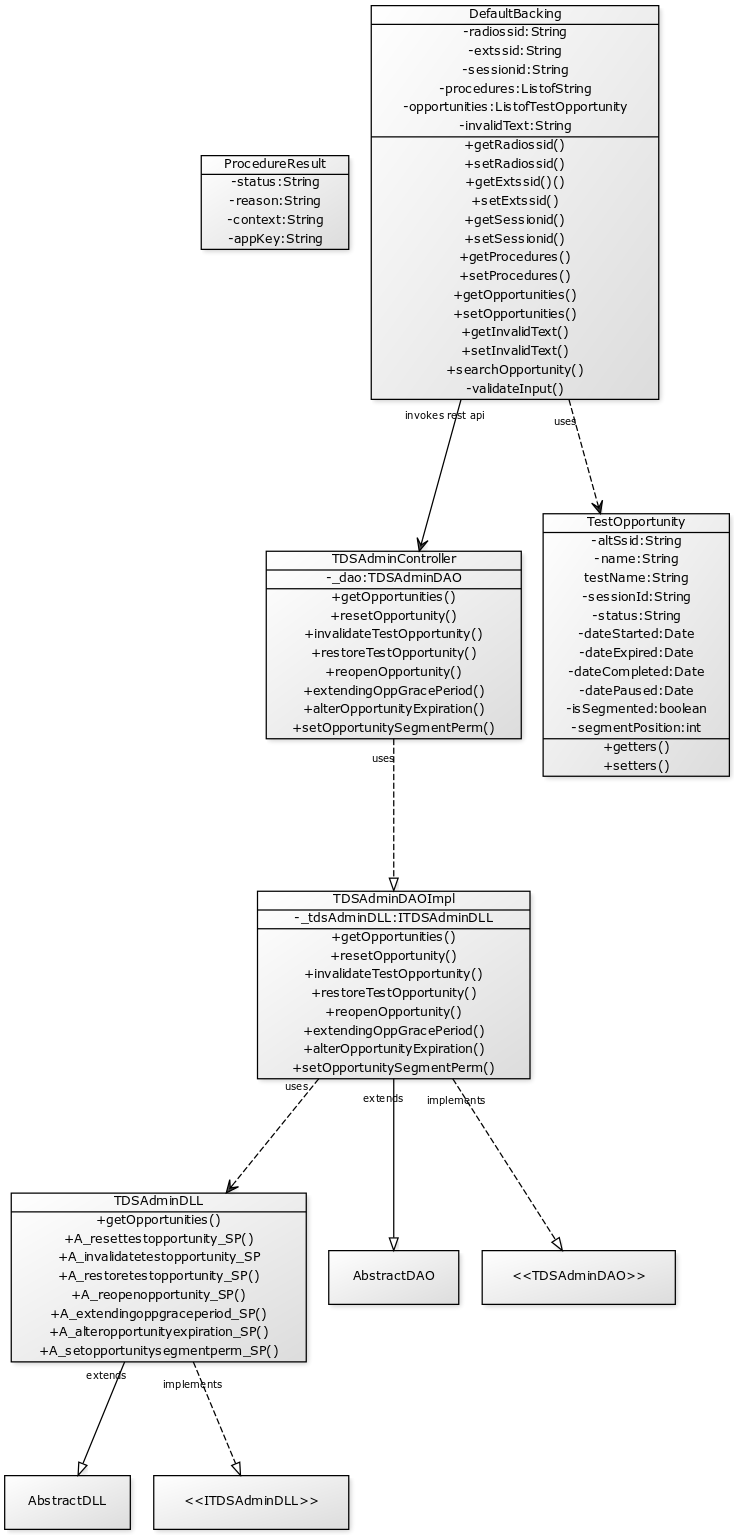
Figure 3: Search Opportunities by SSID and/or session ID

Components:

1. TDS Admin UI : This will be one jsf page providing the interface to user for fetching opportunities, select database operations and modify opportunities.
2. TDS Admin Backing Bean: This is the java code working behind the UI, the class is called DefaultBacking.
3. TDS REST API: These are the REST web service called from TDS Admin Backing Bean to interact with the database layer. TDS Admin can’t call the database layer directly.
4. TDS DLL: This component provides the API to execute database operation in raw level.
5. SSO: This manages user authentication (Not shown in diagram, for clarity).
6. MNA: This manages logging of all sorts of activities within the TDS Admin application (Not shown in diagram, for clarity).

### 

## 7.2 Class Diagram



This class diagram show the core classes and methods in the application, there are additional helper methods in some of the classes.

**DefaultBacking**: The backing bean for the default page, the primary and only user interface.

**TestOpportunity**: This is the model class storing information about an opportunity.

**TDSAdminController**: This is the application’s controller class which defines the REST endpoints and handles client requests for getting and updating information about opportunities.

**TDSAdminDAO**: Interface for the data access object (DAO) used by controller.

**TDSAdminDAOImpl**: Implementation for TDSAdminDAO.

**AbstractDAO**: Resides in shared-db module inside sharedmultijar. It’s extended by TDSAdminDAOImpl.

**ITDSAdminDLL**: Interface providing the skeleton for accessing database and calling stored procedures. It is the tds-dll-api project.

**TDSAdminDLL**: Implementation of ITDSAdminDLL. It is in tds-dll-mysql project.

**AbstractDLL**: Resides in shared-db module inside sharedmultijar and extended by TDSAdminDLL.

**ProcedureResult**: It’s used to return the result of procedure execution to the user.

# 8 Managing Table of Opportunities

Used JSF primefaces library ( version 4.0 under Apache license 2.0) for datatable which works with java object of type LazyDataModel<T> (It has generic type of list within it, TestOpportunity in our case) and used custom column sorting using java reflection.

# 9 Set Opportunity Segment Permeability

This option is shown only for logged in user with appropriate authorization. The post -authentication and -authorization workflow is depicted in the following sequence diagram:

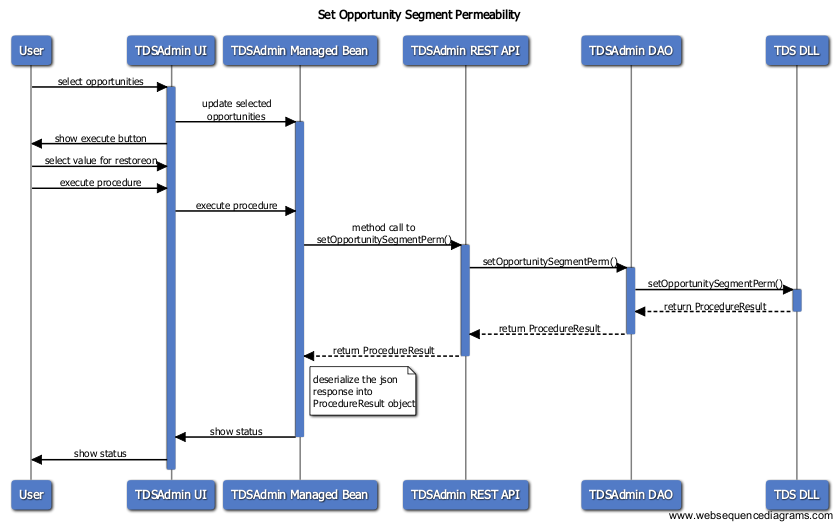
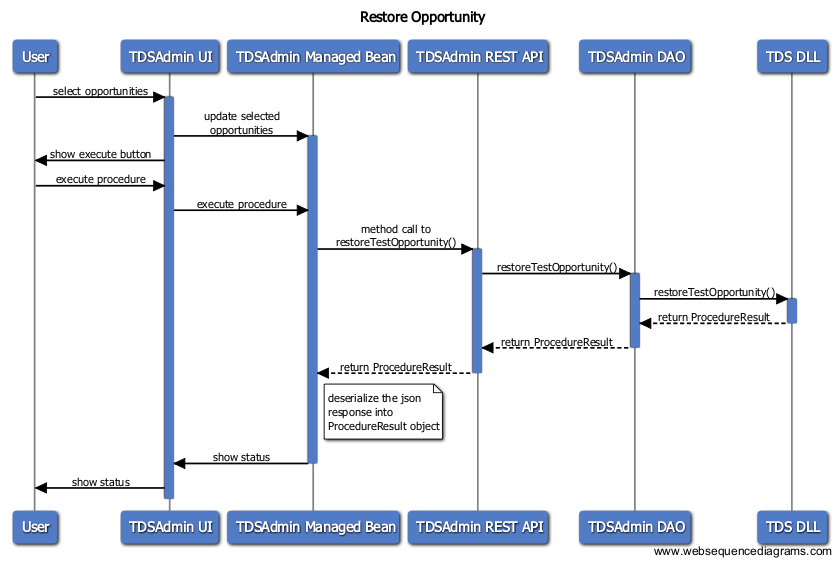
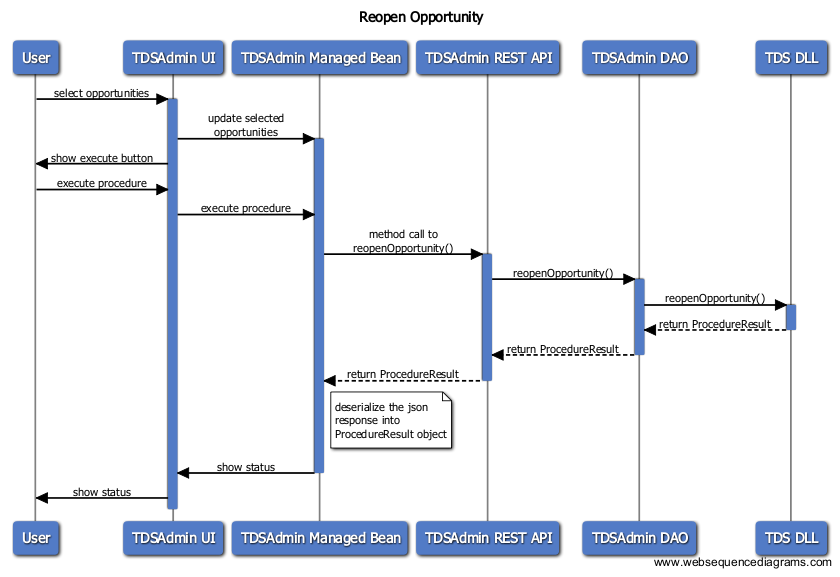


Figure 4: Set opportunity segment permeability

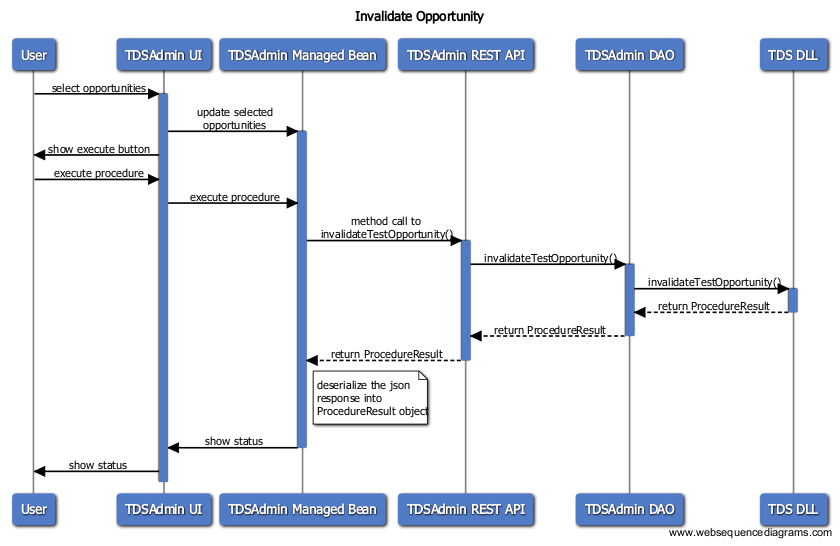
# 10 Restore Test Opportunity



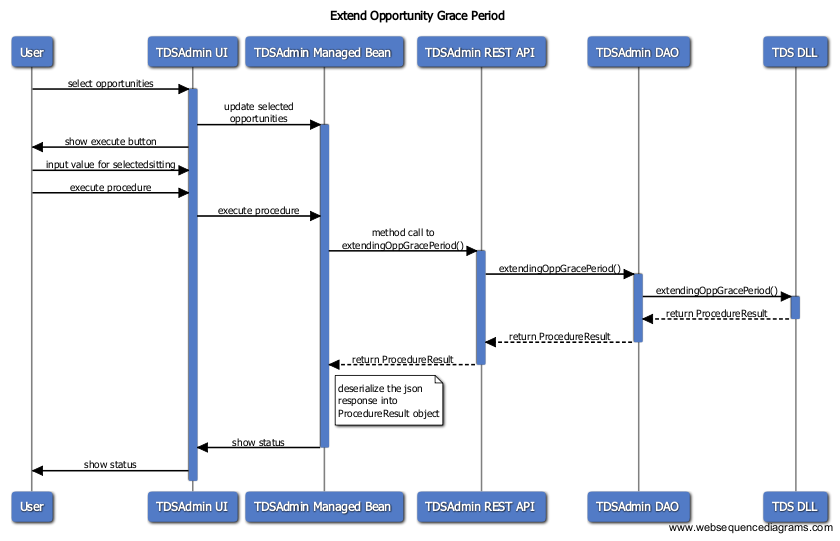
# 11 Reopen a Test Opportunity



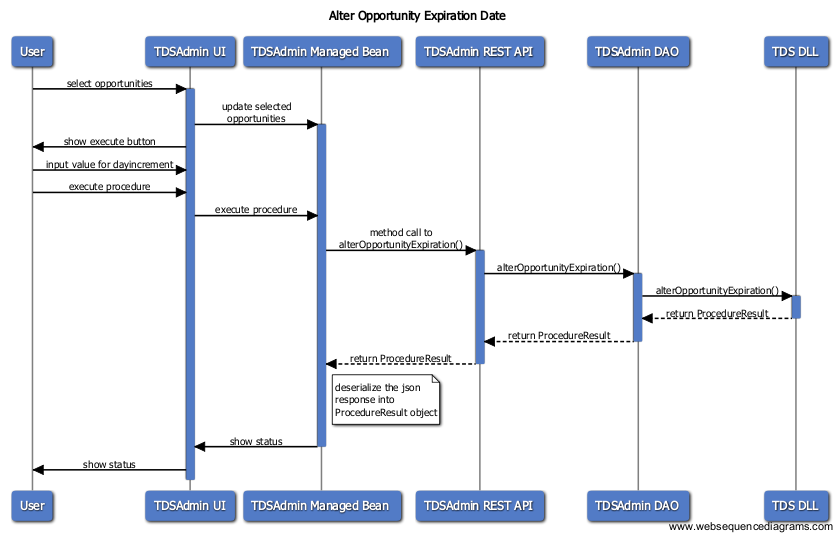
# 12 Invalidate a Test Opportunity



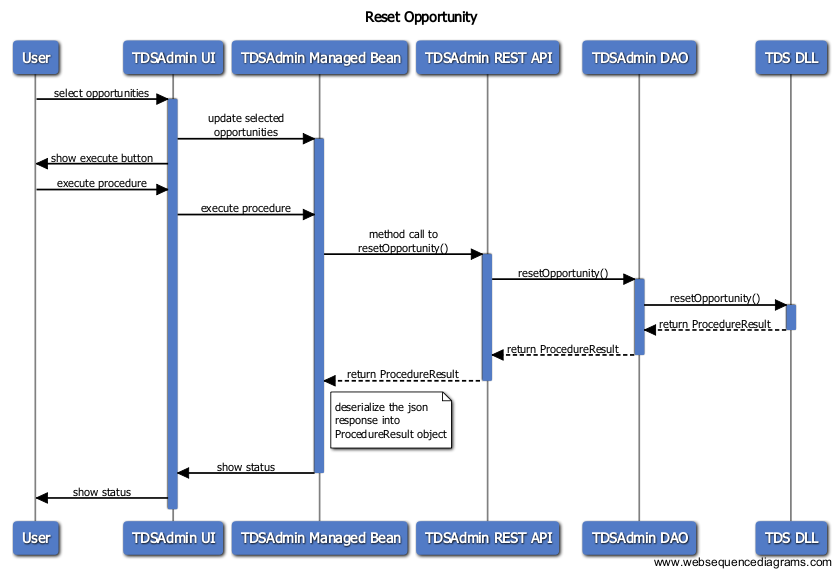
# 13 Extend an Opportunity's Grace Period



# 14 Alter an Opportunity's Expiration



# 15 Reset a Test Opportunity



# 16 Show Result of Procedure Execution

This information is extracted from ProcedureResult object returned after API call for each procedure and shown by the managed bean (DefaultBacking).

# 17 Logout of Current Session

The user interface provides a logout button which enables the user to logout.

# 18 Expire Session after 15 mins of Inactivity

A class called TdsAdminSessionListener is registered in the deployment descriptor which handles the session expiration time due to inactivity.

# 19 Deployment Instructions

When the application is deployed to the server, it reads the following database connection properties from the server’s context.xml file.

* <Parameter name="jdbc.url" value="jdbc:mysql://tds-db.dev.opentestsystem.org:3306?useUnicode=true&amp;useFastDateParsing=false&amp;characterEncoding=UTF-8" type="java.lang.String" override="true" />
* <Parameter name="username" value="dbsbac" type="java.lang.String” override="true" />
* <Parameter name="password" value="*db\_password\_here*" type="java.lang.String" override="true" />

The values have to be changed accordingly in respective environments. Note that the database URL is followed by extra parameters: *“useUnicode=true&amp;useFastDateParsing=false&amp;characterEncoding=UTF-8”*. Without this, the column values of DateTime type cannot be read, given the underlying database is MySQL.

TDSAdmin application has dependency on following projects:

1. shared-db
2. tds-dll-api
3. tds-dll-mysql