Time Tracker Audit 3.2 Requirements Specification

1. Scope

1.1 Overview

The Time Tracker Audit custom component is part of the Time Tracker application. It provides the ability to provide audit records and search the existing audit records. This component handles the persistence and other business logic required by the application.

The design for this specification exists, but requires modification. The text in RED is new requirements. You are to make the additions to the existing design.

1.2 Logic Requirements

1.2.1 Audit Info

1.2.1.1 Audit Header Info

An audit record records the changes to the data in the system. The appropriate data will be stored in order to track changes. Once audit data is written to the database it will becomes read only. This will be enforced by the component itself. The information to be captured is as follows:

- Audit Id – a unique id for this audit entry
- Company Id – the company this audit is associated with
- Application Area – this is a valid id indicating the data being changed
- Table name – the name of the table where the data is being changed
- Entity Id – the id of the record in the table being changed
- User Name – the name of the user who changed the data
- Audit Created – the time stamp of the change
- Action type – this will be (UPDATE, DELETE or INSERT)
- Client Id – If the record is associated with a client this is the client id for the record
- Project Id – If the record is associated with a project this is the project id for the record
- Resource Id – If the record is associated with a resource this is the user id for the record

1.2.1.2 Audit Detail Info

This is the specific data in the row that changed. Once audit data is written to the database it will becomes read only. The information to be captured is as follows:

- Detail Id – the unique Id of the detail record
- Audit Id – the id of the audit header
- Column name – the name of the column which the data is being changed
- Old value - the existing value in the column.
- New value – the value being entered into the column. This will be null if the record was deleted.

1.2.1.3 Search Filters

This component will provide search functionalities based on a logical (AND, OR, NOT) combination of search filters. The following is a summary of the required filters:

- Return all entries with the given Application Area(s)

- Return all entries with the given Resource Id(s)
- Return all entries with the given Client Id(s)
- Return all entries with the given Project Id(s)
- Return all entries with Audit Created within a given inclusive date range (may be open-ended)

1.2.1.4 Database Schema

The time entry information will be stored in the following tables (refer to TimeTrackerAudit_ERD.jpg):

- audit_header
- audit_detail
- application_area

1.2.1.5 Required Operations

- Create new audit record
- Rollback audit record, in the event that the original transaction fails (the one that established the need for this audit record) then this will allow a record to be also rolled back, or deleted.
- Search for audit by filter

NOTE: If the transaction fails while creating or deleting audit record(s) the information, which was being logged, must be written to a log.

1.2.2 Application Area

The Application area defines the areas of application, which are to be audited. This attributes allows portioning of the audit information into logical groups. The current valid application areas are:

- TT_EXPENSE
- TT_FIXED_BILLING
- TT_TIME
- TT_CLIENT
- TT_COMPANY
- TT_PROJECT
- TT_USER
- TT_INVOICE
- TT_NOTIFICATION

1.2.3 Pluggable Persistence

All entities defined in previous sections will be backed by a database. The design will follow the DAO pattern to store, retrieve, and search data from the database. All ID numbers will be generated automatically using the ID Generator component when a new entity is created. All creation and modification dates will be taken as the current datetime.

For this version, the Informix database system will be used as persistence storage for this component and the Time Tracker application. Other database systems should be pluggable into the framework.

1.2.4 JavaBeans Conventions

For all the entities described in previous sections, the JavaBeans conventions will be followed (http://java.sun.com/products/javabeans/docs/spec.html):

- The class is serializable
- The class has a no-argument constructor
- The class properties are accessed through `get`, `set`, `is` methods. i.e. All properties will have get<PropertyName>() and set<PropertyName>(). Boolean properties will have the additional is<PropertyName>().

Note: Event-handling methods are not required.

1.2.5 Transaction Management

As a result of the fine grain components used in this design there needs to be a transaction management strategy, which allows a single transaction to exist that encompasses all components called for a single use case. Since this component will be deployed into an Enterprise Java Bean container, JBoss 4.0.x, a Stateless Session Bean will be used to manage the transaction. The container will start a transaction when a method is invoked if one is not already running. The method will then join the new or existing transaction. Transaction Management will be Container Managed.

1.2.5.1 User API for component

The user API for this component will exist in a Delegate object. This delegate will provide the contract for the component and interface with the EJB. The Delegate is not an EJB rather it will be a POJO. It will look up the EJB and call the related method, retrieve the results and return the results to the consumer. There will be no additional logic in the delegate.

1.2.5.2 Stateless Session Bean

The methods on the Stateless Session bean will have a transaction level of REQUIRED in the deployment descriptor. This will allow for either a new transaction to be created or for the method to join the existing transaction. For this release we will use a Local Bean and not a Remote Bean. There are a few obstacles, which will need to be addressed:

- No File IO from with in the EJB so ConfigurationManager cannot use a file. Values can however be stored in the Deployment Descriptor.
- All parameters passed to/from the Session bean must be Serializable, however the Filter Object in Search Builder 1.3.1 is not Serializable. This is being addressed and will be fixed.
- The Session Bean should not have any class level variables to store things like the DAO. If it does have a class level variable it must be transient, therefore after activation it will have a value of null. Any of the approaches outlined below are acceptable:
  - Have a class level dao attribute and only access it via a getDAO() method which checks for null and sets the dao attribute if it is null.
  - Like the first method have a class level attribute but on creation or activation load the DAO to the class dao attribute. You must then ensure that under all scenarios that the attribute will be not null.
  - Use a singleton to act as a DAO cache
  - There may be others, and you are not limited to one of these.
- No threads can be created with in the EJB.
- Review the Sun J2EE specification for any other limitations.

All Business logic for the componet will reside in the Stateless Session Bean. There will be no logic in the delegate or in the DAO. There is one exception to this, in that the Audit functionality will exist in the DAO.

1.2.5.3 DAO

The DAO's must retrieve the connection that it uses from the configured TXDatasource in JBoss. The configuration of the DataSource should be externalized so that is can be configured at deployment time.

All audit functionality will exist in the DAO.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The Time Tracker application will use this component to generate various time and expense reports for employees, projects, and clients.

1.5 Future Component Direction

Other database systems maybe plugged in for some client environments. Additional reports may be implemented in the future.

2. Interface Requirements

2.1.1 Graphical User Interface Requirement

None.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

- Development language: Java 1.4
- Compile target: Java 1.4, Java 1.5

2.1.4 Package Structure

com.topcoder.timetracker.audit

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Display for empty or null table cells

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
- DB Connection Factory
- Search Builder
- Time Tracker Common
- Time Tracker Base Entry
- Time Tracker Audit

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

Informix Database.

3.2.4 QA Environment:

- JBoss 4.0
- Windows 2000
- Windows Server 2003
- Informix

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 Poseidon.

