Time Tracker Common 3.1 Requirements Specification

1. Scope

1.1 Overview

The Time Tracker Common Custom component is part of the Time Tracker application. It provides for some basic common classes used by many of the components in TimeTracker. This component is simply a couple of classes. It encapsulates the persistence for payment types. It is packaged as a component in order to make it commonly available to many other components.

This component previously existed as part of the existing Time Tracker User component.

1.2 Logic Requirements

1.2.1 TimeTracker Bean

This is the basis for every entity, which it persisted in to the database. It provides the common attributes, which are required when persisting. This is an abstract class that is to be used by many of the entities in Time Tracker

1.2.2 Payment Terms

1.2.2.1 Overview

Payment Terms are used on an invoice to determine the payment agreement for that invoice. Clients and projects have default payment terms associated with their records. The pamany term extends TimeTrackerBean and has the following attributes:

- PaymentTerm Id – unique identifier for the payment term
- Description – a descriptnio which describes the number of days allowed before payment is due
- Term – the actual number of days before a payment is due
- Active – used to delete terms which are no longer valid, a setting of false will remove it from the active list

1.2.2.2 Database Schema

The payment terms information will be stored in the following tables (refer to TimeTrackerCommon_ERD.jpg):

- payment_terms

1.2.2.3 Required Operations

- Create a new payment term
- Retrieve an payment term by ID
- Update an existing payment term
- Enumerate all existing payment terms
- Enumerate all active payment terms

1.2.3 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.

1.2.4 Component Interface Diagram

The new Component Interface Diagram depicts most of the interface. The CID is subject to change as it goes through the design phase.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The Time Tracker application will use this component to perform operations related to company and user authentication and authorization.

1.5 Future Component Direction

Other database systems maybe plugged in for some client environments. Multiple user stores may be used for the same client environment.

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.common

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

None.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

- JavaBeans (http://java.sun.com/products/javabeans/docs/spec.html)

3.2.2 TopCoder Software Component Dependencies:

**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:

- Solaris 7
- RedHat Linux 7.1
- 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.