Fixed Billing Entry 1.0 Requirements Specification

1. Scope

1.1 Overview

The Fixed Billing Entry custom component is part of the Time Tracker application. It provides an abstraction of a fixed billing entry that an manager enters into the system. This component handles the persistence and other business logic required by the application.

While this a new component, it is very similar to the Expense Entry component. The previous version of the Expense Entry component should be referenced when designing this, as it will provide a useful starting place.

1.2 Logic Requirements

1.2.1 Company Account

A company has a relation to everything in the context of the application, such as users, clients, projects, and entries. This relation provides a separation of data by company such that many separate companies can use the same Time Tracker application with a logical separation of data.

1.2.2 Fixed Billing Entry

1.2.2.1 Overview

A fixed billing entry represents the date and amount of money an project manager has spent for a particular project and client. It is normally used for billing purposes. This entity object FixedBillingEntry extends Base Entry and models the following fixed billing entry information:

- Entry ID – the unique fixed billing entry ID number (TimeTrackerBean id field)
- Amount – the amount of money the employee spent
- Fixed Billing Type – the type of fixed billing
- Status – the status of the fixed billing entry
- Invoice – this is the invoice that the entry is associated with. This may be null if it has not yet been invoiced.

1.2.2.2 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 a given company ID
- Return all entries with the given Invoice ID
- Return all entries with description that contains a given string
- Return all entries with entry date within a given inclusive date range (may be open-ended)
- Return all entries with amount within a given inclusive amount range (may be open-ended)
- Return all entries with a given fixed billing type
- Return all entries with a given fixed billing status
- Return all entries with a given billable flag
- Return all entries with a given reject reason ID
- Return all entries created within a given inclusive date range (may be open-ended)
- Return all entries modified within a given inclusive date range (may be open-ended)
- Return all entries created by a given username
- Return all entries modified by a given username

1.2.2.3 Database Schema

The fixed billing entry information will be stored in the following tables (refer to TimeTrackerFixedBilling_ERD.jpg):

- Fixed_billing_entry
- fb_reject_reason

1.2.2.4 Required Operations

- Create a new fixed billing entry
- Retrieve an existing fixed billing entry by ID
- Update an existing fixed billing entry information
- Delete an existing fixed billing entry
- Enumerate all existing fixed billing entries
- Batch versions of the CRUD operations
- Search fixed billing entries by filters
- Get/Set fixed billing type of an existing fixed billing entry (company ID must match)
- Get/Set fixed billing status of an existing fixed billing entry
- Link reject reason IDs to an existing fixed billing entry (company ID must match)
- Unlink reject reason IDs from an existing fixed billing entry
- Unlink all reject reason IDs (if any) from an existing fixed billing entry
- Get all linked reject reason IDs for an existing fixed billing entry

NOTE: See the older version of the Expense Entry Component Interface Diagram in the ZUML for more on the last six bullet pointed methods.

1.2.2.5 Audit Requirements

Each method, which is capable of modification of data, is required to allow the consumer to optionally require auditing. This allows the consumer to determine if the change of data will be audited or not. The Time Tracker Audit component will encapsulate the actual auditing of the data. Note that the audit information should not exist for a transaction, which rolled back. If you have a transaction that fails and you have already submitted audit information make sure to remove the audit information.

The Audit component required the consumer to identify the application area that the audit is for. The application area for the Fixed Billing Entry will be TT_FIXED_BILLING.

1.2.3 Fixed Billing Status

1.2.3.1 Overview

Each fixed billing entry has an assigned status. The entry status will change over the course of the application lifetime. This component extends TimeTrackerBean and models the following fixed billing status information:

- Fixed Billing Status ID – the unique fixed billing status ID number (TimeTrackerBean id field)
- Description – a brief description of the fixed billing status

1.2.3.2 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 statuses with description that contains a given string
- Return all statuses created within a given inclusive date range (may be open-ended)
- Return all statuses modified within a given inclusive date range (may be open-ended)
- Return all statuses created by a given username
- Return all statuses modified by a given username

1.2.3.3 Database Schema

The fixed billing status information will be stored in the following tables (refer to TimeTrackerFixedBilling_ERD.jpg):

- fixed_billing _status

1.2.3.4 Required Operations

- Create a new fixed billing status
- Retrieve an existing fixed billing status by ID
- Update an existing fixed billing status information
- Delete an existing fixed billing status
- Enumerate all existing fixed billing statuses
- Search existing fixed billing by filters

1.2.4 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 will be pluggable into the framework.

1.2.5 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.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The Time Tracker application will use this component to perform operations related to fixed billing entries.

1.5 Future Component Direction

Other database systems maybe plugged in for some client environments.

2. Interface Requirements

2.1.1 Graphical User Interface Requirement

None.

2.1.2 External Interfaces

- Time Tracker Common
  - TimeTrackerBean
- Time Tracker Audit
  - AuditManager
  - AuditHeader
  - AuditDetail
- Time Tracker Base Entry
  - BaseEntry
- Time Tracker Reject Reason
  - RejectReasonManager
  - RejectReason

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.entry.fixedbilling

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:

- Configuration Manager
- DB Connection Factory

- ID Generator
- Search Builder
- Time Tracker Common
- Time Tracker Reject Reason
- 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:

- 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. No use of Store procedures or triggers should exist.

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.

