Invoice 1.0 Requirements Specification

1. Scope

1.1 Overview

The Invoice custom component is part of the Time Tracker application. It provides an abstraction of an Invoice used to bill a client once time, expense and fixed billing entries are entered for a project. This component handles the persistence and other business logic required by the application.

1.2 Logic Requirements

1.2.1 Company Account

A company has a relation to everything in the context of the application, such as invoices, 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 Invoice

1.2.2.1 Overview

An Invoice represents the aggregation of Time, Expense and Fixed Billing entries for a given period of time and Project. The invoice will be used as a device to send printed billing to clients and for tracking of revenue. The entity object Invoice extends TimeTrackerBean and models the following expense entry information:

- Invoice Id – the unique Invoice Id number (TimeTrackerBean id field)
- Company Id – the Company that this invoice belongs to
- Project Id – the project for which this invoice will be billed to
- Invoice Number – The Accounting Invoice number which ties this invoice back to the account system
- Purchase Order number – Entered by admin to tie the invoice to a customers purchase order
- Invoice Date – the date that this invoice was created
- Due date – the Invoice date plus the number of days in the terms
- Service Sub Total – the total of all the Service Details (will be dynamically calculated)
- Expense Sub Total – the total of all the expense (will be dynamically calculated)
- Invoice Status – The current status of the invoice
- Payment terms – the terms for this invoice
- Sales tax – the Sales tax for this invoice, from the project record
- Expense Entries – the list of Expense being billed on this invoice
- Service Details – the list of Time and Fixed Billing Entries being billed on this Invoice
- Paid – has this invoice been paid

1.2.2.2 Search Filters

This component will provide search functionalities based on a logical (AND, OR, NOT) combination of search filters. The return will be a collection of Invoice objects. If no results are found then the result set will be empty. An exception will be thrown in the event of an error condition being raised. The following is a summary of the required filters:

- Return all entries with a given company Id
- Return all entries with a given project Id
- Return all entries with a given client Id
- Return all entries with a given Invoice status
- Return all entries with a given Invoice number
- Return all entries with invoice date within a given inclusive date range (may be open-ended)
- Return all entries with due date within a given inclusive date range (may be open-ended)
- Return all entries with the boolean paid set / unset
- 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 Invoice information will be stored in the following tables (refer to TimeTrackerInvoice_ERD.jpg):

- invoice

1.2.2.4 Required Operations

- Create a new Invoice
- Retrieve an existing Invoice by ID
- Update an existing Invoice
- Enumerate all existing Invoices
- Search Invoices by filter and depth. The Depth will indicate if only the Invoices are returned or if the entries, which make up the details, are returned as well.
    - INVOICE_ONLY – will return only invoices and no entries
    - INVOICE_ALL – will return the invoice and the Entries associated with the Invoice
- Can Create Invoice – this method looks at a particular project and checks all entries (Time, Expense and Fixed billing) if there are any that are PENDING then it will return false, else it will return true.

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 Invoice will be TT_INVOICE. Modifications to the Time, Expense or Fixed billing Entry should be done using the Time Tracker Time, Time Tracker Expense or Time Tracker Fixed billing Component, with the audit function turned on.

1.2.3 Service Detail

1.2.3.1 Overview

The Service Detail encapsulates the data for both Time and rate of the resource. Once a Time entry is associated with an Invoice the Rate is stored with it. We use a Service Detail to create a composite view of the two pieces of data. This allows the service detail rate to be edited apart from original project_workers rate entry. ProjectWorker exists in the Project Component and defines system users as workers on a project. The entity object ServiceDetail extends TimeTrackerBean and models the following expense entry information:

- Service Detail ID – the unique Service Detail Id number (TimeTrackerBean id field)
- Invoice Id – the parent invoice that this record is assigned to
- Time Entry Id – the id of the associated Time Entry
- Rate – The resources Rate if this is a time entry
- Amount - the amount of the hours times the rate.
-

1.2.3.2 Database Schema

The service detail information will be stored in the following tables (refer to TimeTrackerInvoice_ERD.jpg):

- service_detail

1.2.3.3 Required Operations

- Create a new Service Detail
- Retrieve an existing Service Detail by ID
- Update an existing Service Detail information
- Delete an existing Service Detail
- Batch versions of the CRUD operations
- Enumerate all existing Service Detail for a given Invoice

1.2.3.4 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 Service detail will be TT_INVOICE. Modifications to the Time Entry should be done using the Time Tracker Time, with the audit function turned on.

1.2.4 Invoice Status

1.2.4.1 Overview

Each Invoice has an assigned status. The status will change over the course of the application lifetime. This entity object InvoiceStatus extends TimeTrackerBean and models the following expense status information:

- Invoice Status ID – the unique Invoice status ID number (TimeTrackerBean id field)
- Description – a brief description of the Status

1.2.4.2 Database Schema

The invoice status information will be stored in the following tables (refer to TimeTrackerInvoice_ERD.jpg):

- invoice_status

1.2.4.3 Required Operations

- Create a new Invoice status
- Retrieve an existing invoice status by ID
- Update an existing Invoice status information
- Delete an existing Invoice status
- Enumerate all existing Invoice statuses

1.2.5 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.

1.2.6 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 Invoice processing. Invoices will be created after the Projects Manager approves all Time, Expense and Fixed Billing entries. Once the Invoice has been created the Invoice will be managed through a short workflow, which allows account the ability to modify the invoice and purchase order number as well as indicate when the invoice has been paid.

1.5 Future Component Direction

Invoice reporting, aging of invoices and additional steps I the workflow process will likely evolve.

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 Time Entry
    - TimeManager
    - TimeEntry
- Time Tracker Expense Entry
    - ExpenseManager
    - ExpenseEntry
- Time Tracker FixedBilling Entry
    - FixedBillingManager
    - FixedBillingEntry
- Time Tracker Project
    - Project Manager
    - Project
    - ProejctWorker

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.invoice

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 Time Entry
- Time Tracker Expense Entry
- Time Tracker Fixed Billing Entry
- Time Tracker Common
- Time Tracker Audit
- Time Tracker Project

**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.