IM Persistence 1.0 Requirements Specification

1. Scope

1.1 Overview

The IM Persistence component provides database implementations of the persistence interfaces for a chat application, including user profiles, roles, categories and statuses. For user profiles, two implementations will be provided. One is for registered users, the other for unregistered users. Their database schemas are different.

1.2 Logic Requirements

1.2.1 Chat User Profile

For this customized version, the user profile will consist of the following attributes. All attributes have single value.

- First Name
- Last Name
- Company
- Title
- Email

1.2.2 Profile Key Manager

The component will provide an implementation of the `ProfileKeyManager` interface. The manager deals with two types of users: “Registered” and “Unregistered”. ID Generator will be used to generate an id for each such user. Records are persisted in the `all_user` table of the database.

```
CREATE TABLE all_user (
  user_id INTEGER NOT NULL,
  registered_flag VARCHAR(1) NOT NULL,
  username VARCHAR(64) NOT NULL,
  create_date DATETIME YEAR TO FRACTION(3) NOT NULL,
  create_user VARCHAR(64) NOT NULL,
  modify_date DATETIME YEAR TO FRACTION(3) NOT NULL,
  modify_user VARCHAR(64) NOT NULL,
  PRIMARY KEY(user_id)
);
```

For registered users, the username is specified in the `ProfileKey`. For unregistered users, the username is not determined yet. The `ProfileKeyManager` should simply put the string form of the newly assigned id to the username field.

1.2.3 Profile Persistence

The component will provide two implementations of the `ChatUserProfilePersistence` interface. One is for registered users, the other for unregistered users. The persistence should work with the user profiles containing the attributes mentioned in 1.2.1.

1.2.3.1 For Registered Users

Create, update and delete operations are disabled. Only retrieve and search functionalities need to be implemented. The database schema is a truncated version.

```
create table 'informix'.user (
    user_id DECIMAL(10,0) not null,
```

```
    first_name VARCHAR(64),
    last_name VARCHAR(64),
    handle VARCHAR(50) not null
);

alter table 'informix'.user add constraint primary key
      (user_id)
      constraint u124_45;

create table 'informix'.email (
    user_id DECIMAL(10,0),
    address VARCHAR(100)
);

create table 'informix'.company (
    company_id DECIMAL(10,0),
    company_name VARCHAR(100)
);

alter table 'informix'.company add constraint primary key
        (company_id)
        constraint u171_139;

create table 'informix'.contact (
    contact_id DECIMAL(10,0) not null,
    company_id DECIMAL(10,0) not null,
    title VARCHAR(100)
);

alter table 'informix'.contact add constraint foreign key
        (company_id)
        references 'informix'.company
        (company_id)
        constraint contact_company_fk;

alter table 'informix'.contact add constraint foreign key
        (contact_id)
        references 'informix'.user
        (user_id)
        constraint contact_user_fk;
```

The username passed in the `ProfileKey` corresponds to handle. If a user is associated with more than one companies (and thus titles), only one of them is selected and returned.

1.2.3.2 For Unregisterd Users

Update and delete operations are disabled. Only create, retrieve and search functionalities need to be implemented. The database schema is shown below:

```
CREATE TABLE client (
  client_id INTEGER NOT NULL,
  first_name VARCHAR(64) NOT NULL,
  last_name VARCHAR(64) NOT NULL,
  company VARCHAR(256) NULL,
  title VARCHAR(256) NULL,
```

```
  email VARCHAR(256) NOT NULL,
  create_date DATETIME YEAR TO FRACTION(3) NOT NULL,
  create_user VARCHAR(64) NOT NULL,
  modify_date DATETIME YEAR TO FRACTION(3) NOT NULL,
  modify_user VARCHAR(64) NOT NULL,
  PRIMARY KEY(client_id)
);
```

The username passed in the `ProfileKey` corresponds to client_id after casting it to an integer.

1.2.3.3 Additional Attribute

An additional “Name” attribute is put into the profile before it is returned. If the user is of registered type, the name is simply the username. If the user is of unregistered type, the name is the concatenation of “First Name” and “Last Name” attributes contained in the profile.

1.2.4 Roles and Categories

These operations must be provided for roles and categories:

- Get users assigned to a specified role
- Get all categories
- Get all categories of a specific type (chattable or non-chattable)
- Get / Update the categories associated with a manager

The categories returned must contain these attributes: id, name, description and chattable type. The users returned must contain these attributes: id and name. The database schema will be as follows. Pluggability is not required.

```
CREATE TABLE principal(
  principal_id integer NOT NULL,
  principal_name varchar(255),
  PRIMARY KEY (principal_id)
);

CREATE TABLE role(
  role_id integer NOT NULL,
  role_name varchar(255),
  PRIMARY KEY (role_id)
);

CREATE TABLE principal_role(
  principal_id integer NOT NULL,
  role_id integer NOT NULL,
  PRIMARY KEY (principal_id, role_id),
  FOREIGN KEY (principal_id)
    REFERENCES principal(principal_id),
  FOREIGN KEY (role_id)
    REFERENCES role(role_id)
);

CREATE TABLE category (
  category_id INTEGER NOT NULL,
  name VARCHAR(64) NOT NULL,
  description VARCHAR(256) NOT NULL,
  chattable_flag VARCHAR(1) NOT NULL,
  create_date DATETIME YEAR TO FRACTION(3) NOT NULL,
```

```
  create_user VARCHAR(64) NOT NULL,
  modify_date DATETIME YEAR TO FRACTION(3) NOT NULL,
  modify_user VARCHAR(64) NOT NULL,
  PRIMARY KEY(category_id)
);

CREATE TABLE manager_category (
  manager_id INTEGER NOT NULL,
  category_id INTEGER NOT NULL,
  create_date DATETIME YEAR TO FRACTION(3) NOT NULL,
  create_user VARCHAR(64) NOT NULL,
  modify_date DATETIME YEAR TO FRACTION(3) NOT NULL,
  modify_user VARCHAR(64) NOT NULL,
  PRIMARY KEY(manager_id, category_id),
  FOREIGN KEY(category_id)
    REFERENCES category(category_id)
);
```

1.2.5 Status Persistence

The component will also provide an implementation of the `EntityStatusTracker` interface. For the entity of name XXX, the status will be stored in XXX_status table; the status history will be stored in XXX_status_history table.

```
CREATE TABLE XXX_status (
  XXX_status_id INTEGER NOT NULL,
  name VARCHAR(64) NOT NULL,
  description VARCHAR(256) NOT NULL,
  create_date DATETIME YEAR TO FRACTION(3) NOT NULL,
  create_user VARCHAR(64) NOT NULL,
  modify_date DATETIME YEAR TO FRACTION(3) NOT NULL,
  modify_user VARCHAR(64) NOT NULL,
  PRIMARY KEY(XXX_status_id)
);

CREATE TABLE XXX_status_history (
  XXX_id INTEGER NOT NULL,
  start_date DATETIME YEAR TO FRACTION(3) NOT NULL,
  XXX_status_id INTEGER NOT NULL,
  end_date DATETIME YEAR TO FRACTION(3) NULL,
  create_date DATETIME YEAR TO FRACTION(3) NOT NULL,
  create_user VARCHAR(64) NOT NULL,
  modify_date DATETIME YEAR TO FRACTION(3) NOT NULL,
  modify_user VARCHAR(64) NOT NULL,
  PRIMARY KEY(XXX_id, start_date),
  FOREIGN KEY(XXX_status_id)
    REFERENCES XXX_status(XXX_status_id)
);
```

1.3 Required Algorithms

None

1.4 Example of the Software Usage

The assembler will plug in the persistence implementations in the appropriate places to make the IM application work as a whole. For user profiles, this component will be utilized to separate the retrieval of registered and unregistered user details from different data sources. The user details will be used in notification emails and chat requests.

1.5 Future Component Direction

None

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None

2.1.2 External Interfaces

- Please refer to the interfaces defined in the Chat User Profile and Chat Status Tracker components.

2.1.3 Environment Requirements

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

2.1.4 Package Structure

- com.cronos.im.persistence

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Database connection information

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None

3.2.2 TopCoder Software Component Dependencies:

- DB Connection Factory 1.0
- Chat User Profile 2.0
- Chat Status Tracker 1.0

**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 10

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000

- Windows 2003

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.