User Persistence Requirements Specification

1. Scope

1.1 Overview

The User Persistence component provides the Orpheus application with an interface to persistent storage of user data. The central persistence functionality is handled by a stateless session EJB, but for convenient interoperation with a variety of TopCoder components, this is supplemented by an adapter for use of the bean as a persistence plug-in for the User Profile Manager component. The rest of the Orpheus application generally interoperates with this component via the User Profile Manager.

Additionally, this component provides support for pending confirmation storage for the E-mail Validation component. This takes a form similar to the main persistence interface, with a session bean filling the key role behind an adapter that fits it for use with the generic component.

1.2 Logic Requirements

1.2.1 Stateless Session Bean for User Data Persistence

The component will provide a stateless session EJB through which it will access user data stored in an SQL Server database on behalf of clients. The component will not rely on collocation of the client and EJB container. It is not required to use entity beans, and if it does use them it will not expose them to component clients. An ER diagram for the underlying database is provided in section 2.1.2.

1.2.2 User Profile Manager Persistence Plug-in

The component will provide an OrpheusUserProfilePersistence class, an implementation of the User Profile Manager's UserProfilePersistence interface that operates as a client of the persistence EJB (section 1.2.1) to adapt it to the User Profile Manager API. In addition to the base profile type, it will provide the profile types described in the following table (these profile types will not all apply to all users).

Profile Type Name	Property	Description
"player"	"player-payment-pref"	The player's preferred method of receiving prize payments, as a String
"sponsor"	"sponsor-fax-number"	The sponsor's FAX number, as a String
	"sponsor-payment-pref"	The sponsor's preferred method of paying sponsorship fees, as a String
	"sponsor-is-approved"	"Y" or "N" depending on whether the sponsor has been approved by game administration
"admin"	(no properties)	
"address"	"address-street1"	The first line of the user's street address, as a String
	"address-street2"	The second line of the user's street address, as a String
	"address-city"	The name of the city to which the street address applies, as a String
	"address-state"	The name of the U.S. state containing the city, as a String
	"address-postal-code"	The U.S. postal code containing the address, as a String
	"address-phone-number"	A voice telephone number at the address, as a String
"credentials"	"credentials-handle"	A handle or nickname for the user, as a String
	"credentials-password"	The user's password, as a String
	"credentials-is-active"	"Y" or "N" depending on whether the user account is active

All profile types will be implemented via User Profile's `ConfigProfileType`; appropriate User Profile configuration for this purpose will be provided among this component's deliverables.

1.2.3 Symbolic Names

The component will provide an interface defining symbolic constants for the profile type and profile property names specified in requirement 1.2.2:

```
package com.orpheus.user.persistence;

/**
 * The object responsible for managing persistent data about players
 * within the Orpheus application.
 */
public interface UserConstants {

      /**
       * The name of the player user profile type
       */
      public static final String PLAYER_TYPE_NAME = "player";

      /**
       * The name of the sponsor user profile type
       */
      public static final String SPONSOR_TYPE_NAME = "sponsor";

      /**
       * The name of the admin user profile type
       */
      public static final String ADMIN_TYPE_NAME = "admin";

      /**
       * The name of the credentials user profile type
       */
      public static final String CREDENTIALS_TYPE_NAME = "credentials";

      /**
       * The name of the address user profile type
       */
      public static final String ADDRESS_TYPE_NAME = "address";

      /**
       * The name of the payment preference user profile property of
       * the player profile type
       */
      public static final String PLAYER_PAYMENT_PREF =
            "player-payment-pref";

      /**
       * The name of the fax number property of the sponsor profile
       * type
       */
      public static final String SPONSOR_FAX_NUMBER =
            "sponsor-fax-number";
```

```
/**
 * The name of thepayment preference property of the sponsor
 * profile type
 */
public static final String SPONSOR_PAYMENT_PREF =
      "sponsor-payment-pref";

/**
 * The name of the approval property of the sponsor profile type
 */
public static final String SPONSOR_APPROVED =
      "sponsor-is-approved";

/**
 * The name of the street address 1 property of the address
 * profile type
 */
public static final String ADDRESS_STREET_1 = "address-street1";

/**
 * The name of the street address 2 property of the address
 * profile type
 */
public static final String ADDRESS_STREET_2 = "address-street2";

/**
 * The name of the city property of the address profile type
 */
public static final String ADDRESS_CITY = "address-city";

/**
 * The name of the state property of the address profile type
 */
public static final String ADDRESS_STATE = "address-state";

/**
 * The name of the postal code property of the address profile
 * type
 */
public static final String ADDRESS_POSTAL_CODE =
      "address-postal-code";

/**
 * The name of the telephone number property of the address
 * profile type
 */
public static final String ADDRESS_PHONE_NUMBER =
      "address-phone-number";

/**
 * The name of the handle property of the credentials profile
 * type
 */
public static final String CREDENTIALS_HANDLE =
```

```
            "credentials-handle";

        /**
         * The name of the password property of the credentials profile
         * type
         */
        public static final String CREDENTIALS_PASSWORD =
            "credentials-password";

        /**
         * The name of the &amp;quot;is active&amp;quot; property of the
         * credentials profile type
         */
        public static final String CREDENTIALS_IS_ACTIVE =
            "credentials-is-active";
         }
```

1.2.4 Stateless Session Bean for Pending Confirmation Persistence

The component will provide a stateless session EJB through which it will pending e-mail confirmation data stored in an SQL Server database on behalf of clients. The component will not rely on collocation of the client and EJB container. It is not required to use entity beans, and if it does use them it will not expose them to component clients. An ER diagram for the underlying database is provided in section 2.1.2.

1.2.5 E-mail Confirmation Persistence Plug-in

The component will provide an OrpheusPendingConfirmationStorage class, an implementation of the E-mail Confirmation component's PendingConfirmationStorage interface that operates as a client of the persistence EJB (section 1.2.3) to adapt it to the Email Confirmation API.

1.2.6 E-mail Address Case-sensitivity

The component will not be sensitive to the alphabetic case in which the characters of the e-mail address are provided. In particular, searches on e-mail addresses should not require matching case. The component ideally would preserve case when an e-mail address is stored, but it may instead convert to a canonical form if necessary.

1.2.7 Caching

The User Profile Persistence plug-in and associated bean will employ a caching strategy that reduces the number of read requests made across layers (plug-in to bean and bean to DBMS). Writes will not be delayed, but their data may be cached.

1.2.8 Local and Remote Interfaces

The component will provide both local and remote interfaces to both EJBs. The specifications will account for the differences between remote and local method invocation semantics.

1.2.9 Thread Safety

The component will be used in a multithreaded application server environment, and therefore must be thread safe.

1.2.10 Deployment Descriptors

The component deliverables will include sample deployment descriptors for the EJBs.

1.3 Required Algorithms

None

1.4 Example of the Software Usage

The component will be used to provide access to user data for the Orpheus application.

1.5 Future Component Direction

If additional user properties are devised for a future version of the application then this component will be upgraded to handle them. It may also be upgraded to remove direct access to and storage of plain-text user passwords in favor of encrypted passwords.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

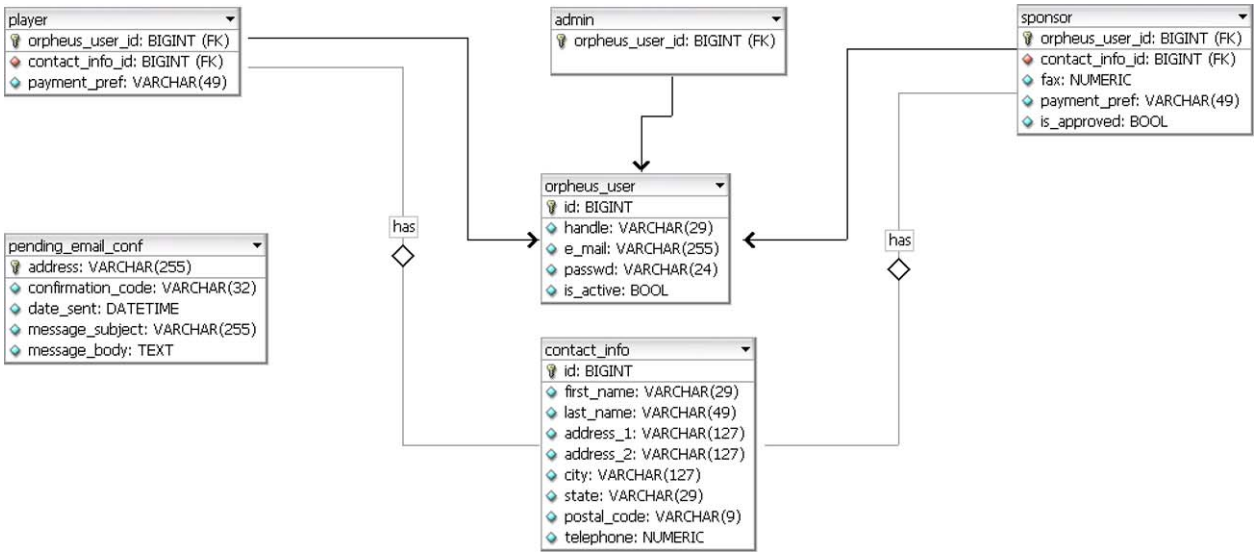
None

2.1.2 External Interfaces

2.1.2.1 Generic Component Interfaces

Refer to the User Profile Manager and Email Confirmation specifications for the `UserProfilePersistence` and `PendingConfirmationStorage` interfaces, respectively.

2.1.2.2 Database ER Diagram



2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.4
- J2EE 1.4

2.1.4 Package Structure

com.orpheus.user.persistence

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?

EJB 2.1

SQL 92

3.2.2 TopCoder Software Component Dependencies:

User Profile Manager 1.0

Email Confirmation 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:

None

3.2.4 QA Environment:

- RedHat Enterprise Linux 4
- JBoss Application Server 4.0.4
- Microsoft SQL Server 2005

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
- Sample Deployment Descriptors

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of Poseidon.