User Management

VXML Server includes a user management system for basic personalization and user-activity tracking. The primary reason for a user management system is to facilitate the customization of voice applications depending on user preferences, demographics, and prior user activity. It is not meant to be a replacement for fully featured commercial user management systems and can be used in conjunction with those systems. Additionally, Unified CVP voice applications do not require the presence of a user management system, it is provided as an aid to application designers.

While the bulk of the user management system is designed to track individual users, its most basic form can prove useful for certain applications. This bulk help those applications that do not need to track individual users, but still want to provide very simple personalization, such as playing *Welcome back* when a call is received from a phone number that has called before. When turned on, the user management system automatically keeps track of information based on the phone numbers of callers. This is available automatically; the developer does to need to do any additional work.

The user management system is fully integrated into VXML Server. An API is included to provide two different interfaces to the user management system. The first interface manages the user database, allowing separate, external processes to populate, maintain, and query the system. The second interface is provided for dynamic components of a voice application to allow runtime updates and queries to the system. This second interface allows a voice application to perform tasks such as playing a customized message to registered users, making decisions based on user demographics or history, and even adding new users after the caller completes a successful registration process. The API has both Java and XML versions. These APIs are fully detailed in Programming Guide for Cisco Unified CVP VXML Server and Unified Call Studio.

Deployment

The user management system is basically a database accessed by VXML Server. Each hosted voice application may refer to a separate user management database or may share databases if users are to be shared across applications. The user management system can be activated by providing a JNDI name for the relational database where the user data is to be stored. This activation is done in the settings pane for the application in the Builder for Call Studio. Currently, the databases supported are MySQL and SQLServer.

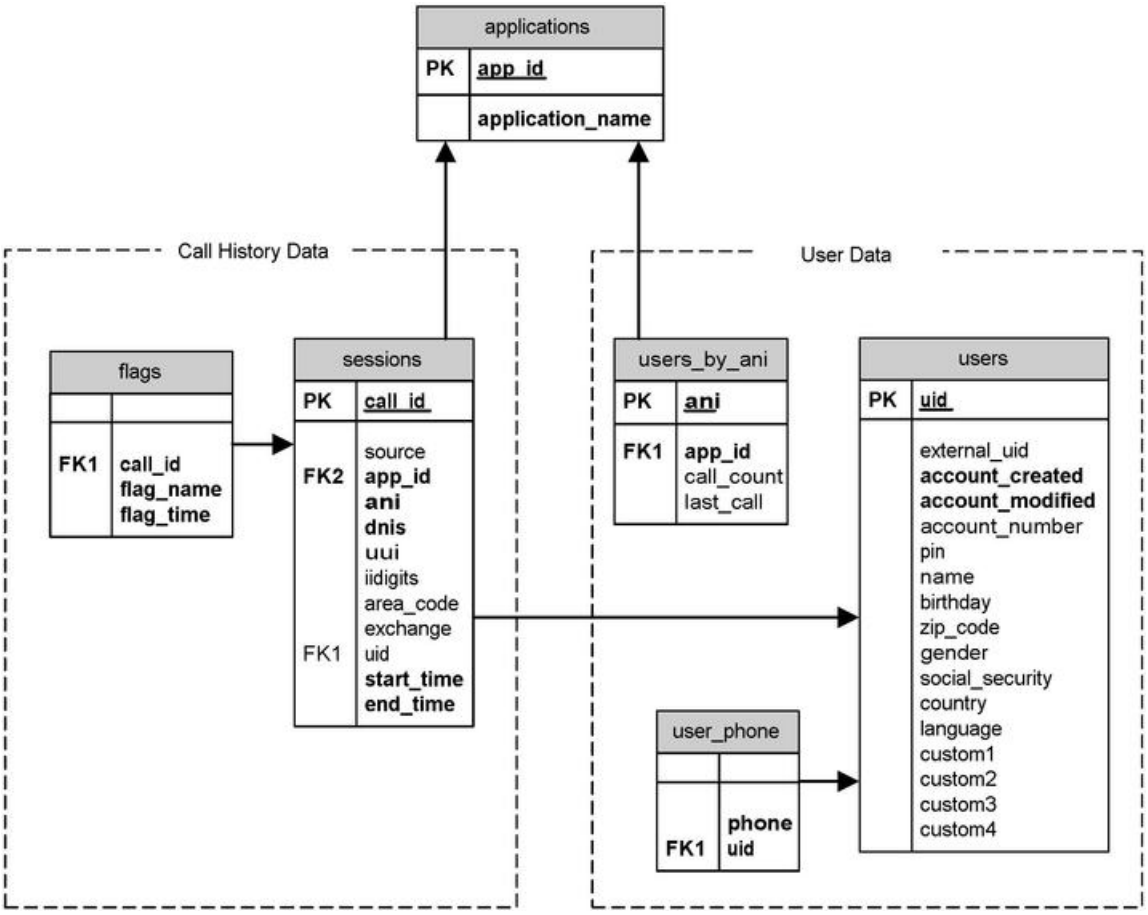
Note The application server must be set up to manage connections to this database.

Once the database is set up, VXML Server automatically handles the process of creating the database tables.

Database Design

The following figure displays an ER diagram of the database tables comprising the user management system. The following sections describe each table individually and its purpose.

Figure 1: Database ER Diagram



Applications

This table is used to provide a primary key for the voice applications utilizing this user management database. Most voice applications use their own user management system in which case this table will have only one entry in it. For those applications that share a common user management system, this table's key is used to keep track of the activities of users visiting each application, should that separation be necessary.

Column	Type	Description
app_id	integer (primary key)	Automatically generated application ID.

application_name	varchar(50)	The name of the application with the specified application ID.

User Data

The tables under this category are used to store information about the users in the system.

users

This table is the main user table. Each row contains the information for a single user. Both demographic and account information are stored here. The table specification is as follows:

Column	Type	Description
uid	integer (primary key)	This is a user ID automatically generated by the system to identify a particular user. Once a call is associated with a UID, the system knows the caller's identity. The user management system relies on this UID throughout.
external_uid	varchar(50)	If an external user management system is used in conjunction with this one, there must be a way to link a user on the Unified CVP system with one in the external system. This column stores the ID for this user on the external system to provide that link. Can be null if the Unified CVP user management system is used exclusively.
account_created	datetime	This stores the time the user was added to the system. It will always have a value.
account_modified	datetime	This stores the time of the last update to this user in the system. It will always have a value.
account_number	varchar(50)	Some voice applications identify users by account numbers. If so, the account number should be stored here, otherwise, it can be null.
pin	varchar(20)	If the voice application uses a PIN to verify the user, the PIN is stored here. Null if no PIN is used or required.
name	varchar(50)	The user's name. Can be null.
birthday	varchar(50)	The user's birthday. Can be null.
zip_code	varchar(10)	The user's zip code. Can be null.
gender	varchar(10)	The user's gender: male, female, or null if not stored.
social_security	varchar(10)	The social security number of the user. Can be null.
country	varchar(50)	The user's full country name. Can be null.
language	varchar(50)	The language the user speaks or prefers. This can be used to provide audio content in different languages. Can be null.

custom1-custom4	varchar(200)	These columns are provided to allow the developer to place custom user-related data in the system. It can be used for such data as e-mail addresses, financial account balances, proprietary IDs, and so on. Can be null.

user_phone

This table is an adjunct to the main user table. It is used to store the phone numbers associated with the user. The reason this data is placed in a separate table is to allow an application to associate more than one phone number with a user. For example, a voice application allowing a user to associate with their account both their home and work numbers can automatically recognize who the caller is when calls are received from either number, rather than requiring them to log in. If multiple phone numbers are not required or necessary, this table can contain one entry per account or remain empty. Because there may be multiple rows in the system with the same UID, there is no primary key to this table. The table specification is as follows:

Column	Type	Description
phone	varchar(10)	A phone number to associate with this account.
uid	integer (foreign key)	The UID identifying the user.

users_by_ani

This table is used to track calls made from specific phone numbers (ANIs). This table is automatically updated by VXML Server and need only be queried by the developer when information about a caller is desired. The table contains information about the number of calls and the last call made from a phone number. This information can be used to welcome a caller back to the application or warn that menu options have changed since their last call even if the application itself is not set up to track individual users through logins. The table specification is as follows:

Column	Type	Description
ani	varchar(10)	The phone number of the caller.
app_id	integer	The application the caller called into. This exists in case multiple applications share a common user management system.
call_count	integer	The number of calls received by this phone number to this application.
last_call	datetime	The last time a call was received by this phone number to this application.

Historical Data

Tracking user information is only part of a user management system. Many applications benefit from knowing information about the past history of a user's interaction with the phone system. This component of the user management system is automatically updated by VXML Server and only needs to be queried by the developer when information about users is desired.

sessions

This table contains records of every call made to the system. It stores telephony information about the call as well as when the call was made. The table specification is as follows:

Column	Type	Description
call_id	integer	This is an automatically incremented ID for the call. It is used exclusively within the user management system.
source	varchar(50)	This column contains the name of the application which transferred to this one or is null if the application was called directly.
app_id	integer	The application ID of the application called. If the user management system is not shared across multiple applications, this ID would be the same for all calls.
ani	varchar(10)	The ANI of the originating caller. Is NA if the ANI was not sent by the telephony provider.
dnis	varchar(10)	The DNIS of the originating caller. Is NA if the DNIS was not sent by the telephony provider.
uui	varchar(100)	The UUI of the originating caller. Is NA if the UUI was not sent by the telephony provider.
iidigits	varchar(100)	The IIDIGITS of the originating caller. Is NA if the IIDIGITS was not sent by the telephony provider.
area_code	varchar(10)	The area code of the originating caller. Is null if the ANI is NA.
exchange	varchar(10)	The exchange of the originating caller. Is null if the ANI is NA.
uid	integer	The UID of the caller if the call was associated with a user. If not, it will appear as null.
start_time	datetime	The date and time the visit to the application began. If no other application can transfer to this one, this will be the time the call was made.
end_time	datetime	The date and time the visit to the application ended. If this application cannot transfer to any other application, this will be the time the call ended in a hang-up or disconnect.

flags

This table contains records of the flags triggered by every call made to the system. Because flags are used to indicate important parts of the voice application, knowing what areas of the voice application people visited in the past can be very useful. The table specification is as follows:

Columns	Type	Description
call_id	integer	This refers to the call ID of the call.
flag_name	varchar(100)	This is the name of the flag that was triggered.
flag_time	datetime	This is the date and time the flag was triggered.