SQL User Data Store Tables and Stored Procedures Configuration Guide

Introduction

Use this guide as a reference to create SQL tables and stored procedures that are needed for SecureAuth IdP to utilize the SQL database for membership and / or profile information.

Prerequisites

1. Have an on-premises SQL User Data Store

2. Integrate the SQL User Data Store with SecureAuth IdP (refer to **Data Tab Configuration** and **SQL Server Configuration Guide** for specific configuration steps)

3. Ensure that ports are open to enable connection to the SQL User Data Store

4. Have basic SQL knowledge to deploy the scripts below

SQL Table Samples

User Table

This single table contains the User ID, Password, and Profile Information

User Table

```
CREATE TABLE [dbo].[UserTable](
        [UserName] [varchar](60) NOT NULL,
        [Password] [varchar](60) NULL,
        [PasswordSalt] [varchar](128) NULL,
        [PasswordFormat] [int] NULL,
        [PwdLastSet] [datetime] NULL,
        [FirstName] [varchar](50) NULL,
        [LastName] [varchar](50) NULL,
        [Phone1] [varchar](40) NULL,
        [Phone2] [varchar](40) NULL,
        [Phone3] [varchar](40) NULL,
        [Phone4] [varchar](40) NULL,
        [Email1] [varchar](60) NULL,
        [Email2] [varchar](60) NULL,
        [Email3] [varchar](60) NULL,
        [Email4] [varchar](60) NULL,
        [AuxID1] [varchar](512) NULL,
        [AuxID2] [varchar](512) NULL,
        [AuxID3] [varchar](512) NULL,
        [AuxID4] [varchar](512) NULL,
        [AuxID5] [varchar](512) NULL,
        [AuxID6] [varchar](512) NULL,
        [AuxID7] [varchar](512) NULL,
        [AuxID8] [varchar](512) NULL,
        [AuxID9] [varchar](512) NULL,
        [AuxID10] [varchar](512) NULL,
        [pinHash] [varchar](120) NULL,
        [Questions] [varchar](1000) NULL,
        [Answers] [varchar](1000) NULL,
        [ChallengeQuestion] [varchar](1000) NULL,
        [ChallengeAnswer] [varchar](1000) NULL,
        [CertResetDate] [datetime] NULL,
        [CertCount] [int] NULL,
        [MobileResetDate] [datetime] NULL,
        [MobileCount] [int] NULL,
        [ExtSyncPwdDate] [datetime] NULL,
        [OATHSeed] [varchar](1000) NULL,
        [OneTimeOATHList] [varchar](1000) NULL,
        [iOSDevices] [varchar](50) NULL,
        [HardwareToken] [varchar](50) NULL,
        [CertSerialNumber] [varchar](1000) NULL,
        [GroupList] [varchar](1000) NULL
) ON [PRIMARY]
GO
```

Object Table Type

This creates the Object Table Type, which is used for Fingerprint and PUSH Notification token information

Object Table Type

```
CREATE TYPE [dbo].[ObjectTable] AS TABLE ([ObjectValue] [varbinary](max) NULL)
GO
```

Fingerprint Table

This table contains Browser / Device Fingerprinting information for each user

Fingerprint Table

```
CREATE TABLE [dbo].[UserFP] (
        [UserName] [varchar](60) NOT NULL
        ,[DigitalFP] [varbinary](max) NOT NULL
        ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

PUSH Notification Table

This table contains PUSH Notification token information for each user

PUSH Notification Table

```
CREATE TABLE [dbo].[UserPN] (
        [UserName] [varchar](60) NOT NULL
        ,[PNToken] [varbinary](max) NOT NULL
        ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

Access History Table

This table contains Access History information for each user

Access History Table

```
CREATE TABLE [dbo].[UserAccessHistory] (
        [UserName] [varchar](60) NOT NULL
        ,[AccessHistory] [varbinary](max) NOT NULL
        ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

Stored Procedure Samples

The following are sample stored procedures for use with the SecureAuth IdP SQL Membership and Profile providers

Membership Stored Procedures

Get User

This stored procedure checks if the username exists, and returns that same username in the case that it does

Get User Stored Procedure

```
CREATE PROC [dbo].[sp_GetUser] @UserName VARCHAR(60)
AS
BEGIN
        SELECT UserName
                ,ISNULL([GroupList], '')
                ,ISNULL([PwdLastSet],'1/1/1900')
        FROM UserTable
        WHERE UserName = @UserName
END
GO
```

Reset Password

This stored procedure resets the password for the given user

Reset Password Stored Procedure

```
CREATE PROC [dbo].[sp_ResetPassword]
@UserName VARCHAR(60),
@Password VARCHAR(60),
@PasswordSalt VARCHAR(128),
@PasswordFormat int
AS
BEGIN
        UPDATE UserTable SET [Password]=@Password, [PasswordSalt]=@PasswordSalt, [PasswordFormat]
=@PasswordFormat, [PwdLastSet]=GetDate() WHERE @UserName = [UserName]
        IF @@ROWCOUNT > 0
        BEGIN
                SELECT 1
        END
END
GO
```

Get Password

This stored procedure gets the password, password salt, and password format

GetPasswordWithFormat Stored Procedure

```
CREATE PROCEDURE [dbo].[sp_GetPasswordWithFormat]
    @UserName                       varchar(60)
AS
BEGIN
    SELECT  [Password], PasswordFormat, PasswordSalt
    FROM    dbo.UserTable
    WHERE   @UserName = UserName
END
GO
```

Create User

This stored procedure inserts the username and password into the user table, and returns a MembershipCreateStatus enumeration, with zero (0) signaling success

For more information on adding new enumeration values to the Create User stored procedure, please click **here**

Create User Stored Procedure

```
CREATE PROC [dbo].[sp_CreateUser]
@UserName VARCHAR(60),
@Password VARCHAR(60),
@PasswordSalt VARCHAR(128),
@PasswordFormat int,
@Status int OUTPUT
AS
BEGIN
INSERT INTO UserTable ([UserName], [Password], [PasswordSalt], [PasswordFormat], [PwdLastSet]) VALUES
(@UserName, @Password, @PasswordSalt, @PasswordFormat, GetDate())

IF @@ROWCOUNT > 0
SELECT @Status = 0
ELSE
SELECT @Status = 1
END
GO
```

Profile Stored Procedures

The sample stored procedures below include all profile information and they must be modified to match the profile information that maps to the SQL profile provider in the SecureAuth IdP Web Admin (Data tab)

For example, if only using SQL to store and retrieve Fingerprint information, then only the Fingerprint parameter needs to be sent by SecureAuth IdP

Get User Profile

This stored procedure retrieves the profile of the given username

Get User Profile Stored Procedure

```
CREATE PROC [dbo].[sp_GetUserProfile] @UserName VARCHAR(60)
AS
BEGIN
    SELECT UserName
        ,IsNull(FirstName, '') FirstName
        ,IsNull(LastName, '') LastName
        ,IsNull(Phone1, '') Phone1
        ,IsNull(Phone2, '') Phone2
        ,IsNull(Phone3, '') Phone3
        ,IsNull(Phone4, '') Phone4
        ,IsNull(Email1, '') Email1
        ,IsNull(Email2, '') Email2
        ,IsNull(Email3, '') Email3
        ,IsNull(Email4, '') Email4
        ,IsNull(AuxID1, '') AuxID1
        ,IsNull(AuxID2, '') AuxID2
        ,IsNull(AuxID3, '') AuxID3
        ,IsNull(AuxID4, '') AuxID4
        ,IsNull(AuxID5, '') AuxID5
        ,IsNull(AuxID6, '') AuxID6
        ,IsNull(AuxID7, '') AuxID7
        ,IsNull(AuxID8, '') AuxID8
        ,IsNull(AuxID9, '') AuxID9
        ,IsNull(AuxID10, '') AuxID10
        ,IsNull(pinHash, '') pinHash
        ,IsNull(Questions, '') Questions
        ,IsNull(Answers, '') Answers
        ,IsNull(ChallengeQuestion, '') ChallengeQuestion
        ,IsNull(ChallengeAnswer, '') ChallengeAnswer
        ,IsNull(CertResetDate, '1/1/1900') CertResetDate
        ,IsNull(CertCount, 0) CertCount
        ,IsNull(CertSerialNumber, '') CertSerialNumber
        ,IsNull(MobileResetDate, '1/1/1900') MobileResetDate
        ,IsNull(MobileCount, 0) MobileCount
        ,IsNull(ExtSyncPwdDate, '1/1/1900') ExtSyncPwdDate
        ,IsNull(HardwareToken, '') HardwareToken
        ,IsNull(iOSDevices, '') iOSDevices
        ,IsNull(OATHSeed, '') OATHSeed
        ,IsNull(OneTimeOATHList, '') OneTimeOATHList
                ,IsNull(GroupList, '') GroupList
    FROM UserTable
    WHERE UserName = @UserName
    SELECT DigitalFP
    FROM UserFP
    WHERE UserName = @UserName
    SELECT PNToken
    FROM UserPN
    WHERE UserName = @UserName
        SELECT AccessHistory
        FROM UserAccessHistory
        WHERE UserName = @UserName
END
GO
```

Update User Profile

This stored procedure updates the user profile with the given profile information

Update User Profile Stored Procedure

```
CREATE PROC [dbo].[sp_UpdateUserProfile] @UserName VARCHAR(60)
        ,@FirstName VARCHAR(50)
        ,@LastName VARCHAR(50)
        ,@Phone1 VARCHAR(60)
        ,@Phone2 VARCHAR(40)
        ,@Phone3 VARCHAR(40)
        ,@Phone4 VARCHAR(40)
        ,@Email1 VARCHAR(60)
        ,@Email2 VARCHAR(60)
        ,@Email3 VARCHAR(60)
        ,@Email4 VARCHAR(60)
        ,@AuxID1 VARCHAR(512)
        ,@AuxID2 VARCHAR(512)
        ,@AuxID3 VARCHAR(512)
        ,@AuxID4 VARCHAR(512)
        ,@AuxID5 VARCHAR(512)
        ,@AuxID6 VARCHAR(512)
        ,@AuxID7 VARCHAR(512)
        ,@AuxID8 VARCHAR(512)
        ,@AuxID9 VARCHAR(512)
        ,@AuxID10 VARCHAR(512)
        ,@pinHash VARCHAR(120)
        ,@Questions VARCHAR(1000)
        ,@Answers VARCHAR(1000)
        ,@ChallengeQuestion VARCHAR(1000)
        ,@ChallengeAnswer VARCHAR(1000)
        ,@CertResetDate DATETIME
        ,@CertSerialNumber VARCHAR(1000)
        ,@CertCount INTEGER
        ,@MobileResetDate DATETIME
        ,@MobileCount INTEGER
        ,@ExtSyncPwdDate DATETIME
        ,@HardwareToken VARCHAR(1000)
        ,@iOSDevices VARCHAR(1000)
        ,@OATHSeed VARCHAR(1000)
        ,@OneTimeOATHList VARCHAR(1000)
        ,@DigitalFP dbo.ObjectTable READONLY
        ,@PNToken dbo.ObjectTable READONLY
        ,@AccessHistory dbo.ObjectTable READONLY
AS
IF NOT EXISTS (
                SELECT UserName
                FROM UserTable
                WHERE UserName = @UserName
                )
BEGIN
 INSERT INTO UserTable (UserName) VALUES (@UserName)
END
        BEGIN TRY
                BEGIN TRANSACTION
                UPDATE UserTable
                SET FirstName = @FirstName
                        ,LastName = @LastName
                        ,Phone1 = @Phone1
                        ,Phone2 = @Phone2
                        ,Phone3 = @Phone3
                        ,Phone4 = @Phone4
                        ,Email1 = @Email1
                        ,Email2 = @Email2
                        ,Email3 = @Email3
                        ,Email4 = @Email4
                        ,AuxID1 = @AuxID1
                        ,AuxID2 = @AuxID2
                        ,AuxID3 = @AuxID3
                        ,AuxID4 = @AuxID4
                        ,AuxID5 = @AuxID5
                        ,AuxID6 = @AuxID6
                        ,AuxID7 = @AuxID7
```

```
                        ,AuxID8 = @AuxID8
                        ,AuxID9 = @AuxID9
                        ,AuxID10 = @AuxID10
                        ,pinHash = @pinHash
                        ,Questions = @Questions
                        ,Answers = @Answers
                        ,ChallengeQuestion = @ChallengeQuestion
                        ,ChallengeAnswer = @ChallengeAnswer
                        ,CertResetDate = @CertResetDate
                        ,CertCount = @CertCount
                        ,CertSerialNumber = @CertSerialNumber
                        ,MobileResetDate = @MobileResetDate
                        ,MobileCount = @MobileCount
                        ,ExtSyncPwdDate = @ExtSyncPwdDate
                        ,HardwareToken = @HardwareToken
                        ,iOSDevices = @iOSDevices
                        ,OATHSeed = @OATHSeed
                        ,OneTimeOATHList = @OneTimeOATHList
                WHERE UserName = @UserName
                --- Update Fingerprints ---
                DELETE
                FROM UserFP
                WHERE UserName = @UserName
                INSERT INTO UserFP
                SELECT @UserName
                        ,ObjectValue
                FROM @DigitalFP
                ---------------------------
                --- Update Push Notification ---
                DELETE
                FROM UserPN
                WHERE UserName = @UserName
                INSERT INTO UserPN
                SELECT @UserName
                        ,ObjectValue
                FROM @PNToken
                ---------------------------
                --- Update Access History ---
                DELETE
                FROM UserAccessHistory
                WHERE UserName = @UserName
                INSERT INTO UserAccessHistory
                SELECT @UserName
                        ,ObjectValue
                FROM @AccessHistory
                ---------------------------
                COMMIT
        END TRY
        BEGIN CATCH
                IF @@TRANCOUNT > 0
                        ROLLBACK
        END CATCH
GO
```

Roles and Permissions

This creates a service account user, service account role, and adds the user to the role

> Change the username (**SecureAuthSQLUser**) and password (**Password**) to the actual values

Create Role and User

```
CREATE LOGIN SecureAuthSQLUser WITH PASSWORD = 'Password';
CREATE USER SecureAuthSQLUser FOR LOGIN SecureAuthSQLUser;
CREATE ROLE db_serviceaccount
EXEC sp_addrolemember 'db_serviceaccount', 'SecureAuthSQLUser'
GRANT EXECUTE ON [UserStore].dbo.sp_CreateUser TO db_serviceaccount
GRANT EXECUTE ON [UserStore].dbo.sp_GetUser TO db_serviceaccount
GRANT EXECUTE ON [UserStore].dbo.sp_GetUserProfile TO db_serviceaccount
GRANT EXECUTE ON [UserStore].dbo.sp_UpdateUserProfile TO db_serviceaccount
GRANT EXECUTE ON TYPE::[UserStore].dbo.ObjectTable TO db_serviceaccount
GRANT EXECUTE ON [UserStore].dbo.sp_GetPasswordWithFormat TO db_serviceaccount
GRANT EXECUTE ON [UserStore].dbo.sp_ResetPassword TO db_serviceaccount
```

All-inclusive Create Database Script

The **CreateDatabase.sql** script creates a "UserStore" database and also creates all of the above tables, objects, and stored procedures within that database