**Outsystems Security Migration - Guide for DBA**

**1. Database and Table Creation**

**1.1 Create the Database**

Use the following SQL script to create the database:

USE [master]; GO CREATE DATABASE [CBP\_Permissions\_Test] ON PRIMARY ( NAME = N'CBP\_Permissions\_Test', FILENAME = N'E:\SQLDEVDBs\SQL2019DBs\SQLDATA\CBP\_Permissions\_Test.mdf', SIZE = 8192KB, MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB ) LOG ON ( NAME = N'CBP\_Permissions\_Test\_log', FILENAME = N'E:\SQLDEVDBs\SQL2019DBs\SQLLOG\CBP\_Permissions\_Test\_log.ldf', SIZE = 8192KB, MAXSIZE = 2048GB, FILEGROWTH = 65536KB ); GO

**1.2 Create Required Tables**

USE CBP\_Permissions\_Test; GO

CREATE TABLE Customers ( id BIGINT PRIMARY KEY, external\_id NVARCHAR(255) NULL, name NVARCHAR(255) NOT NULL, tenant\_id INT NOT NULL UNIQUE, is\_active BIT NOT NULL, created\_on DATETIME NOT NULL, updated\_on DATETIME NULL );

CREATE TABLE Groups ( id BIGINT PRIMARY KEY, name NVARCHAR(512) NOT NULL, customer\_id BIGINT NOT NULL, created\_at DATETIME NOT NULL, updated\_at DATETIME NOT NULL, FOREIGN KEY (customer\_id) REFERENCES Customers(id) );

CREATE TABLE Users ( id BIGINT PRIMARY KEY, username NVARCHAR(255) UNIQUE NOT NULL, email NVARCHAR(255) NULL, mobile\_phone NVARCHAR(50) NULL, tenant\_id INT NOT NULL, is\_active BIT NOT NULL, creation\_date DATETIME NOT NULL, last\_login DATETIME NULL, external\_id NVARCHAR(255) NULL, customer\_id BIGINT NOT NULL, FOREIGN KEY (customer\_id) REFERENCES Customers(id) );

CREATE TABLE UserGroups ( user\_id BIGINT NOT NULL, group\_id BIGINT NOT NULL, PRIMARY KEY (user\_id, group\_id), FOREIGN KEY (user\_id) REFERENCES Users(id), FOREIGN KEY (group\_id) REFERENCES Groups(id) );

CREATE TABLE Roles ( id BIGINT PRIMARY KEY, name NVARCHAR(255) UNIQUE NOT NULL );

CREATE TABLE GroupRoles ( group\_id BIGINT NOT NULL, role\_id BIGINT NOT NULL, PRIMARY KEY (group\_id, role\_id), FOREIGN KEY (group\_id) REFERENCES Groups(id), FOREIGN KEY (role\_id) REFERENCES Roles(id) );

**2. Data Migration Instructions**

**2.1 Migrate Customers**

INSERT INTO Customers (id, external\_id, name, tenant\_id, is\_active, created\_on, updated\_on) SELECT ID, EXTERNALID, NAME, TENANT\_ID, CASE WHEN ISACTIVE = 1 AND ISREMOVED = 0 THEN 1 ELSE 0 END, CREATEDON, UPDATEDON FROM outsystems.dbo.OSUSR\_HBV\_CUSTOMER;

**2.2 Migrate Groups**

INSERT INTO Groups (id, name, customer\_id, created\_at, updated\_at) SELECT ID, NAME, CUSTOMER\_ID, GETDATE(), GETDATE() FROM outsystems.dbo.ossys\_group;

**2.3 Migrate Users**

INSERT INTO Users (id, username, email, mobile\_phone, tenant\_id, is\_active, creation\_date, last\_login, external\_id, customer\_id) SELECT ID, USERNAME, EMAIL, MobilePhone, TENANT\_ID, IS\_ACTIVE, CREATION\_DATE, LAST\_LOGIN, EXTERNAL\_ID, CUSTOMER\_ID FROM outsystems.dbo.ossys\_user;

**2.4 Migrate User-Group Assignments**

INSERT INTO UserGroups (user\_id, group\_id) SELECT gu.USER\_ID, gu.GROUP\_ID FROM outsystems.dbo.ossys\_group\_user gu;

**2.5 Migrate Roles**

INSERT INTO Roles (id, name) SELECT DISTINCT ID, NAME FROM outsystems.dbo.ossys\_role;

**2.6 Migrate Group Role Assignments**

INSERT INTO GroupRoles (group\_id, role\_id) SELECT DISTINCT gu.GROUP\_ID, gu.ROLE\_ID FROM outsystems.dbo.ossys\_group\_role gu;

**3. Handling Orphaned Records**

**3.1 Identify Orphaned Users**

SELECT u.ID, u.USERNAME, u.EMAIL FROM outsystems.dbo.ossys\_user u LEFT JOIN outsystems.dbo.ossys\_group\_user gu ON u.ID = gu.USER\_ID WHERE gu.USER\_ID IS NULL;

**3.2 Assign Orphaned Users to a Default Group**

INSERT INTO Groups (id, name, customer\_id, created\_at, updated\_at) VALUES ((SELECT MAX(id) + 1 FROM Groups), 'Default Users', (SELECT TOP 1 id FROM Customers ORDER BY id ASC), GETDATE(), GETDATE());

INSERT INTO UserGroups (user\_id, group\_id) SELECT ID, (SELECT id FROM Groups WHERE name = 'Default Users') FROM Users WHERE ID NOT IN (SELECT DISTINCT user\_id FROM UserGroups);

**4. Verification Steps**

**4.1 Check Data Counts**

SELECT COUNT(*) AS Customers\_Count FROM Customers; SELECT COUNT(*) AS Groups\_Count FROM Groups; SELECT COUNT(*) AS Users\_Count FROM Users; SELECT COUNT(*) AS UserGroups\_Count FROM UserGroups; SELECT COUNT(*) AS Roles\_Count FROM Roles; SELECT COUNT(*) AS GroupRoles\_Count FROM GroupRoles;

**4.2 Verify Data Integrity**

-- Ensure all users belong to at least one group SELECT u.ID, u.USERNAME FROM Users u LEFT JOIN UserGroups ug ON u.ID = ug.user\_id WHERE ug.user\_id IS NULL;

-- Ensure all groups are assigned to at least one role SELECT g.ID, g.NAME FROM Groups g LEFT JOIN GroupRoles gr ON g.ID = gr.group\_id WHERE gr.group\_id IS NULL;

-- Ensure all roles are assigned to at least one group SELECT r.ID, r.NAME FROM Roles r LEFT JOIN GroupRoles gr ON r.ID = gr.role\_id WHERE gr.role\_id IS NULL;

**5. Post-Migration Cleanup**

-- Remove any test data DELETE FROM UserGroups WHERE user\_id IN (SELECT id FROM Users WHERE username LIKE 'testuser%'); DELETE FROM Groups WHERE name LIKE 'TestGroup%'; DELETE FROM Roles WHERE name LIKE 'TestRole%';