Migrating from SQL CE to Zumero

Migrating mobile applications from SQL Compact Edition and Merge Replication to Zumero for SQL Server

---

ZUMERO

great minds sync alike.

Table of Contents

Introduction

In 2010, when businesses needed to extend enterprise data out to early generation mobile devices, developers in the Microsoft eco-system often deployed .NET Compact Framework based applications targeting Windows Mobile. With the help of SQL Server Compact Edition (SQL CE) and SQL Server Merge Replication, these applications could accommodate the "occasionally-connected" or "offline" nature of mobile computing and still integrate with corporate data stored in SQL Server. These mobile solutions targeted Windows CE, Pocket PC, and the Windows Mobile family of mobile devices allowing developers to leverage a familiar Microsoft-based technology stack: VB.NET, C#, and SQL Server through ADO.NET.

To further assist developers, Hood Canal Press published Rob Tiffany's *Enterprise Data Synchronization with Microsoft SQL Server 2008 and SQL Server Compact 3.5 Mobile Merge Replication*[1]. Rob's book provided the technical details for configuring and administering a replicated SQL Server database solution for mobile applications. The book also included a sample application used by "Contoso Bottling"[2], a division of Microsoft's renowned Contoso Ltd[3], selling soft drinks through a geographically distributed network of distribution centers.

Then in 2011 Microsoft released SQL CE 4.0, and things began to change. SQL CE 4.0 removed support for data replication with SQL Server using Merge Replication and Sync Framework[4]. Later, the release of SQL Server 2014 dropped support for Merge Replication with older versions of SQL CE entirely[5]. To make matters worse, while Microsoft was reducing their SQL CE support on mobile devices, most enterprise developers were faced with a growing need to more easily support data replication and synchronization on platforms such as iOS and Android.

Zumero for SQL Server[6] was developed to provide a migration path for developers stranded on SQL CE and needing a data replication and sync solution for SQL Server that could easily support all mobile platforms, including iOS and Android. This paper outlines the steps necessary to migrate from SQL CE / Merge Replication to Zumero for SQL Server, focusing on the changes needed to port Rob's sample app to Zumero.

What is Zumero for SQL Server?

Zumero for SQL Server (ZSS) is a data replication and synchronization solution for mobile apps and SQL Server.

The concept of ZSS is similar to SQL CE with Merge Replication:

- Data, or a subset of data, is retrieved from SQL Server to an occasionally connected mobile device and stored in a local database (SQLite).
- Mobile applications running on the device access the data in the local database using native client libraries.
- New, modified, or deleted data is synchronized with SQL Server using the Zumero client library's `sync()` method.

ZSS consists of three major components, the ZSS Server, ZSS Manager, and ZSS Client SDK.

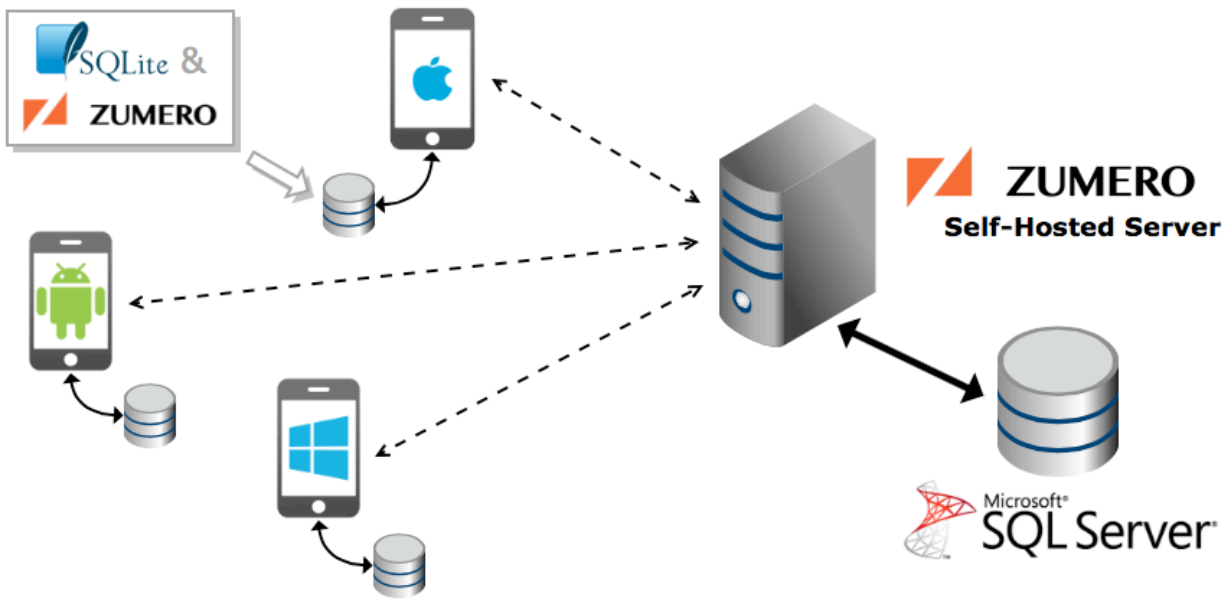
The ZSS Server is an Internet Information Server (IIS) module (version 7.5 or higher) which handles synchronization between ZSS clients and SQL Server 2008 R2 or later. The ZSS Server communicates with SQL Server using ODBC and can run on a different machine from SQL Server. For security, SSL should be enabled on the IIS site running Zumero.

The ZSS Manager is a tool for configuring SQL Server databases for synchronization with ZSS clients. It is designed to look familiar to users of SQL Server Management Studio. ZSS Manager allows you to prepare specific database tables for synchronization, setup user permissions and authentication, and create filters for controlling the amount of data synchronized to mobile devices.

The ZSS Client SDK consists of documentation, headers, and native, client side libraries for building applications for Android, iOS, Xamarin (iOS/Android), PhoneGap (iOS/Android), and Windows Phone 8.x. The ZSS Client SDK also includes libraries for building desktop applications on Windows, Windows RT, Mac OS, and Linux. On each client platform, ZSS relies on SQLite as its underlying data storage layer.

In essence, mobile apps are built with a native SQLite client library in tandem with the platform's Zumero library. Often upon start-up, the application will make a call to Zumero's `sync()` method. The initial call will contact the ZSS Server, download the SQL Server database schema, and create the equivalent schema within SQLite for use on the device. After the local SQLite database has been initialized, the application can create additional tables, indices or other SQLite database objects for its own use.

As applications make modifications to the local SQLite database or require updated data be retrieved from SQL Server, they again call `sync()` to synchronize the local SQLite database with SQL Server. Upon successful completion of `sync()`, changes in the local SQLite database will have been sent and updated in SQL Server, and any changes made within SQL Server will be retrieved and applied to the mobile application's SQLite database. ZSS handles any conflicts that may have occurred during the `sync()` call.



Contoso Server Side Implementation

In Rob Tiffany's book, *Enterprise Data Synchronization with Microsoft SQL Server 2008 and SQL Server Compact 3.5 Mobile Merge Replication*, the Contoso Bottling example provides detailed instructions for configuring SQL Server Merge Replication through the implementation of a private network. Similarly, the migrated sample also uses a private network. The network will participate in a domain, ZDOMAIN, and consist of three machines, a Domain Controller named ZDC, an IIS Server called ZWEB, and a SQL Server, ZSQL.

Each machine should be running Windows Server 2008 R2 or Windows Server 2012, and be members of the ZDOMAIN domain. As with Rob's sample, this design is provided as a model for existing corporate networks. If availability of hardware for this setup is an issue, the configuration can be created with virtual machines. Ensure ZWEB has been assigned the Web Server Role for IIS and IIS Management, and install SQL Server's database engine and the SQL Server management tools onto ZSQL. Again, it is important ZWEB and ZSQL both be joined to the ZDOMAIN domain.

Domain Computers and Users

The ZSS Server's IIS module will run under an IIS Application pool and make the connection to the SQL Server database. ZSS supports database connections with either SQL Server authentication or Windows authentication; the migrated sample will use Windows authentication. With the SQL Server instance and ZSS Server installed on different machines, using a domain user account simplifies the configuration. This requires running ZSS Server's Application Pool under the context of a known, trusted domain user account. The domain account will also need permission to connect to SQL Server.

To verify the Server domain configuration:

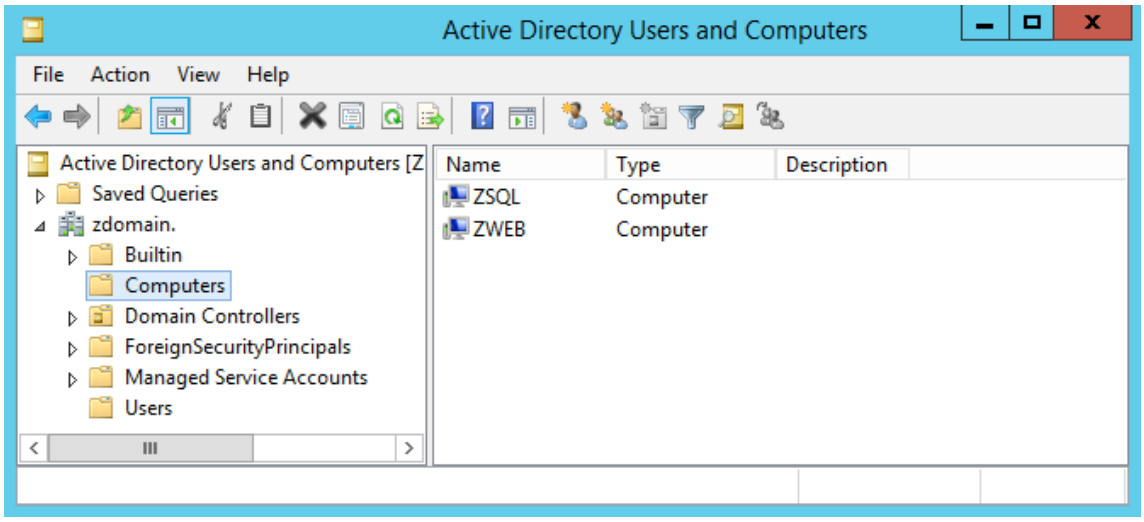
1. Log on to the ZDC computer

2. From the Start menu, type “dsa.msc” to start the “Active Directory Users and Computers” utility

3. In the Active Directory Users and Computers dialog, expand the ZDOMAIN node

4. On the left hand side tree view, select the **Computers** node

5. Verify ZWEB and ZSQL are listed as members of ZDOMAIN



To create the domain user account:

1. Under the ZDOMAIN node, highlight the **Users** node

2. Through the context menu, choose **New → User**

3. In the **New Object – User** dialog:

   a. Enter "ZumeroSync" as the **First name**
   b. Enter "User" for **Last name**
   c. Enter "zumerosyncuser" for **User logon name**

4. Click **Next**

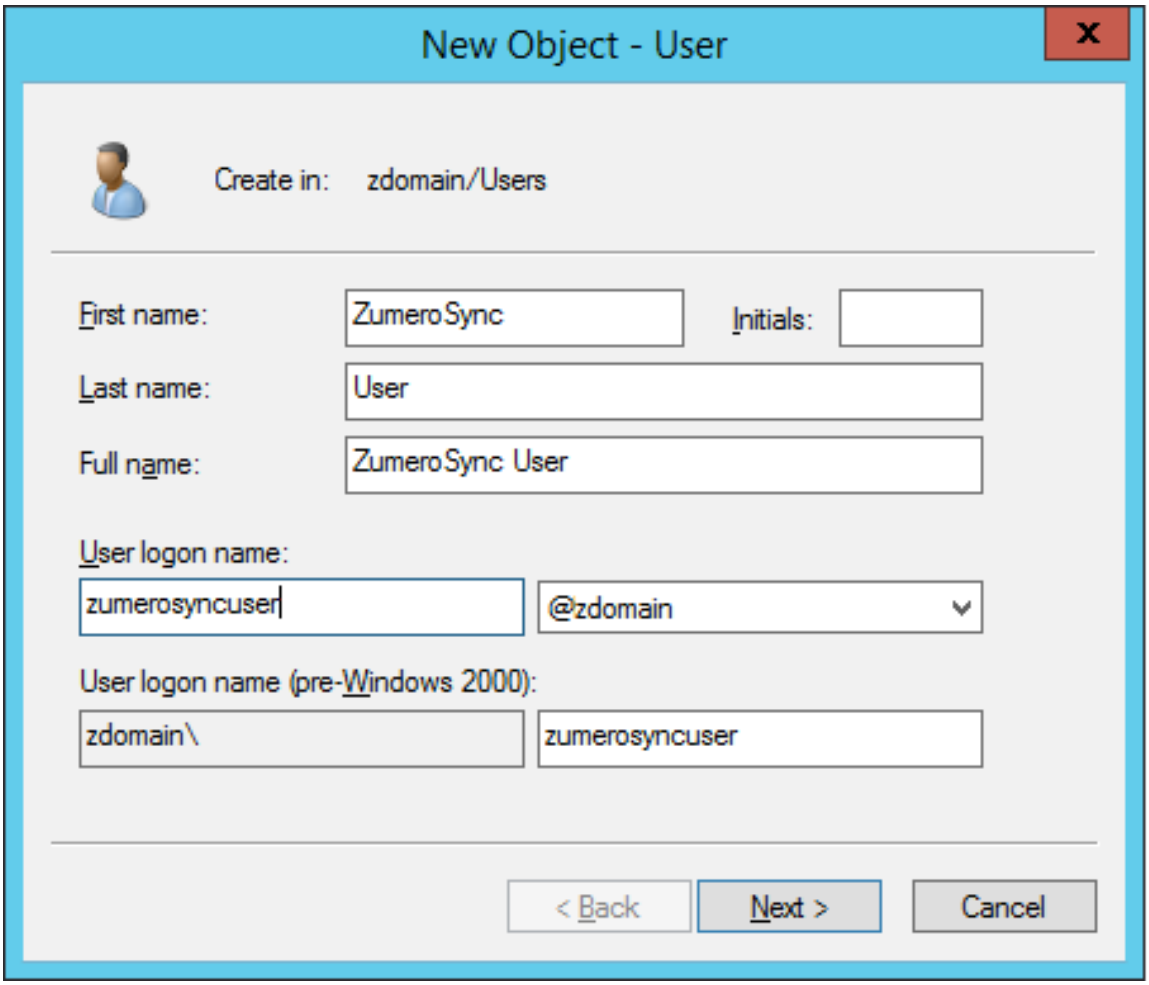
5. For this page:

   a. Enter "P@assw0rd" in both the **Password** and **Confirm password** text boxes
   b. User must change password at next logon checkbox should be **unchecked**
   c. **Password never expires** check box is **checked**

6. Click **Next**

7. Confirm the settings, and click **Finish**

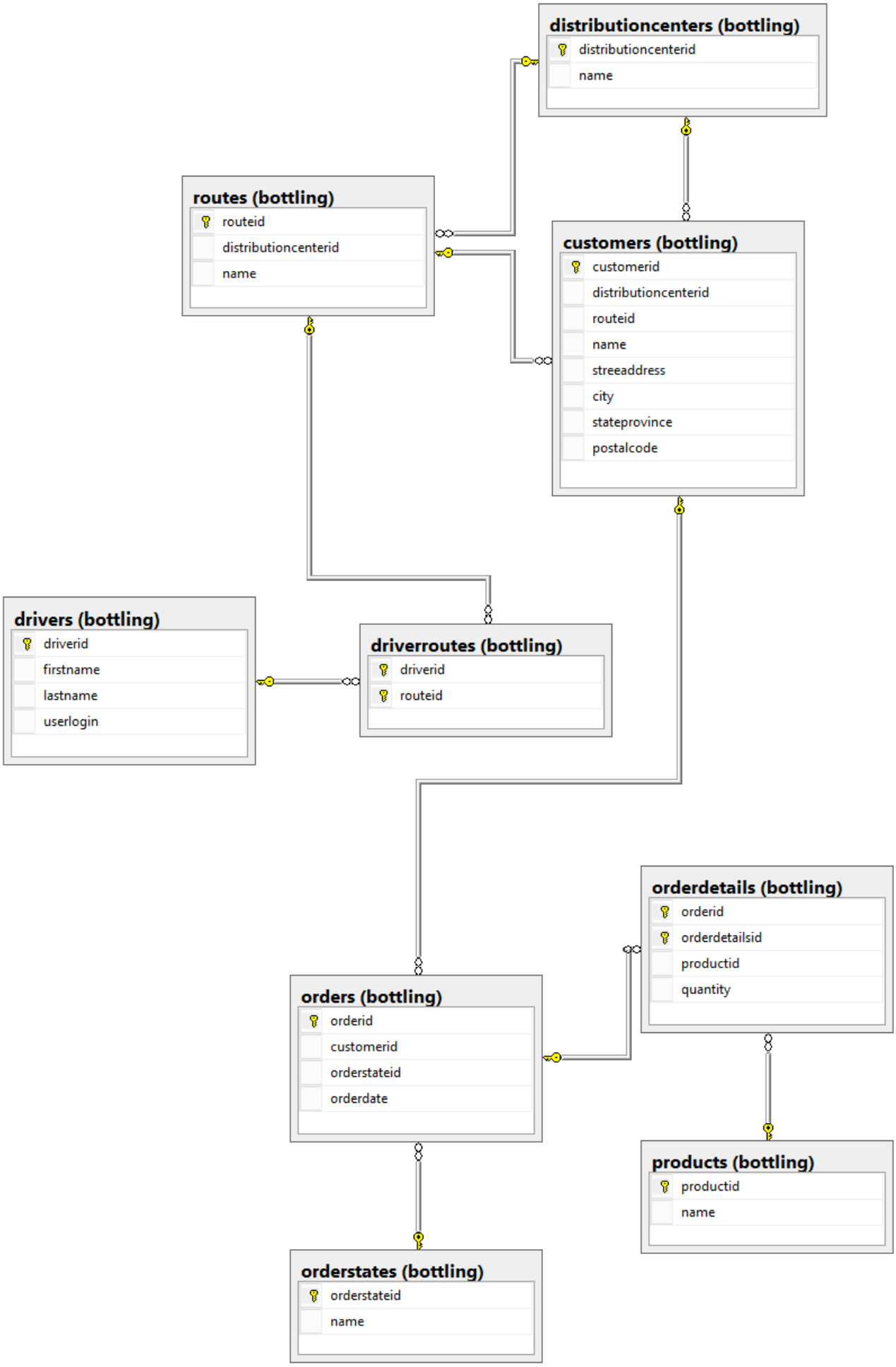
8. Close the **Active Directory Users and Computers** dialog



SQL Server and the Contoso Bottling Database

Our migrated sample will slightly modify the supply-chain database for Contoso Bottling, *contosodb*, used in the original sample application. Zumero allows mobile devices to filter synchronized data by "user", which in this case is a *driver* delivering the bottled products. While the migrated sample application does not update data within the SQL Server database, the *orders* and *orderdetails* tables are configured to allow for data modification.

The Contoso Bottling database model has a set of *distribution centers* that has a relationship to *routes* and *customers*. Customers have *orders* with *order details*. A deviation from the original sample is the removal of the route's ID from the *drivers* table, and the addition of a *driverroutes* table that allows a *driver* to run multiple *routes*. *Products* contain all the drinks available for sale, and *orderstate* contains information regarding the state of an order's fulfillment. The following graphic displays the data model and relationships:


distributioncenters (bottling)
distributioncenterid
name
routes (bottling)
routeid
distributioncenterid
name
customers (bottling)
customerid
distributioncenterid
routeid
name
streeaddress
city
stateprovince
postalcode
drivers (bottling)
driverid
firstname
lastname
userlogin
driverroutes (bottling)
driverid
routeid
orderdetails (bottling)
orderid
orderdetailsid
productid
quantity
orders (bottling)
orderid
customerid
orderstateid
orderdate
products (bottling)
productid
name
orderstates (bottling)
orderstateid
name

Notice the tables contain IDENTITY columns instead of uniqueidentifier / ROWGUIDCOL. Zumero works just fine with tables containing whole number IDENTITY columns[8]. There's no need to replace a table's IDENTITY columns with uniqueidentifiers for primary keys.

The database files used in the migrated sample can be extracted from the *contoso_bottling_sql.zip* file which can be downloaded at https://drive.google.com/file/d/0B-KMSqi99DHJMXNDNUpxY3E1dDQ/view?usp=sharing

To create the database:

1. Download and extract *contoso_bottling_sql.zip*

2. From the Start menu, type "ssms.exe" to start the SQL Server Management Studio

3. Log on to SQL Server Management Studio using an account with privileges to create databases

4. Select menu, **File → Open → File…**, browse to the *contoso_ddl.sql* file, open it, and execute the statements in the file

5. Once complete, follow the same steps for the *contoso_data.sql* file

As previously mentioned, the ZDOMAIN\zumerosyncuser will need to be granted access to SQL Server and the contosodb database so that the ZSS Server can access the database. For brevity's sake, this is done in one simple step.

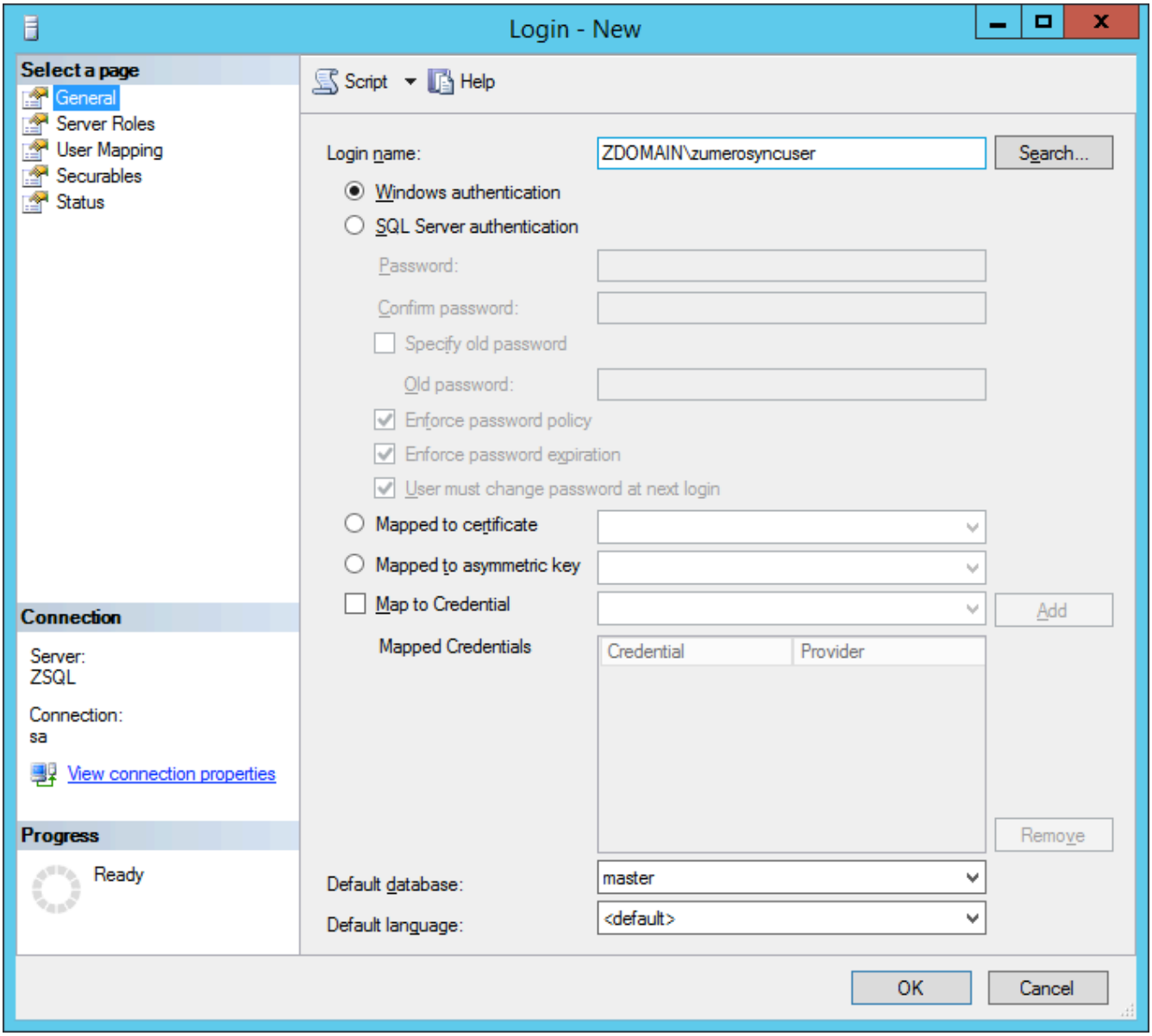
To configure the ZDOMAIN\zumerosyncuser account:

1. Within SQL Server Management Studio's Object Explorer, expand the Server (ZSQL) and **Security** nodes

2. Invoke the context menu on **Logins**, and then "**New Login...**"

3. On the **Login – New** dialog:

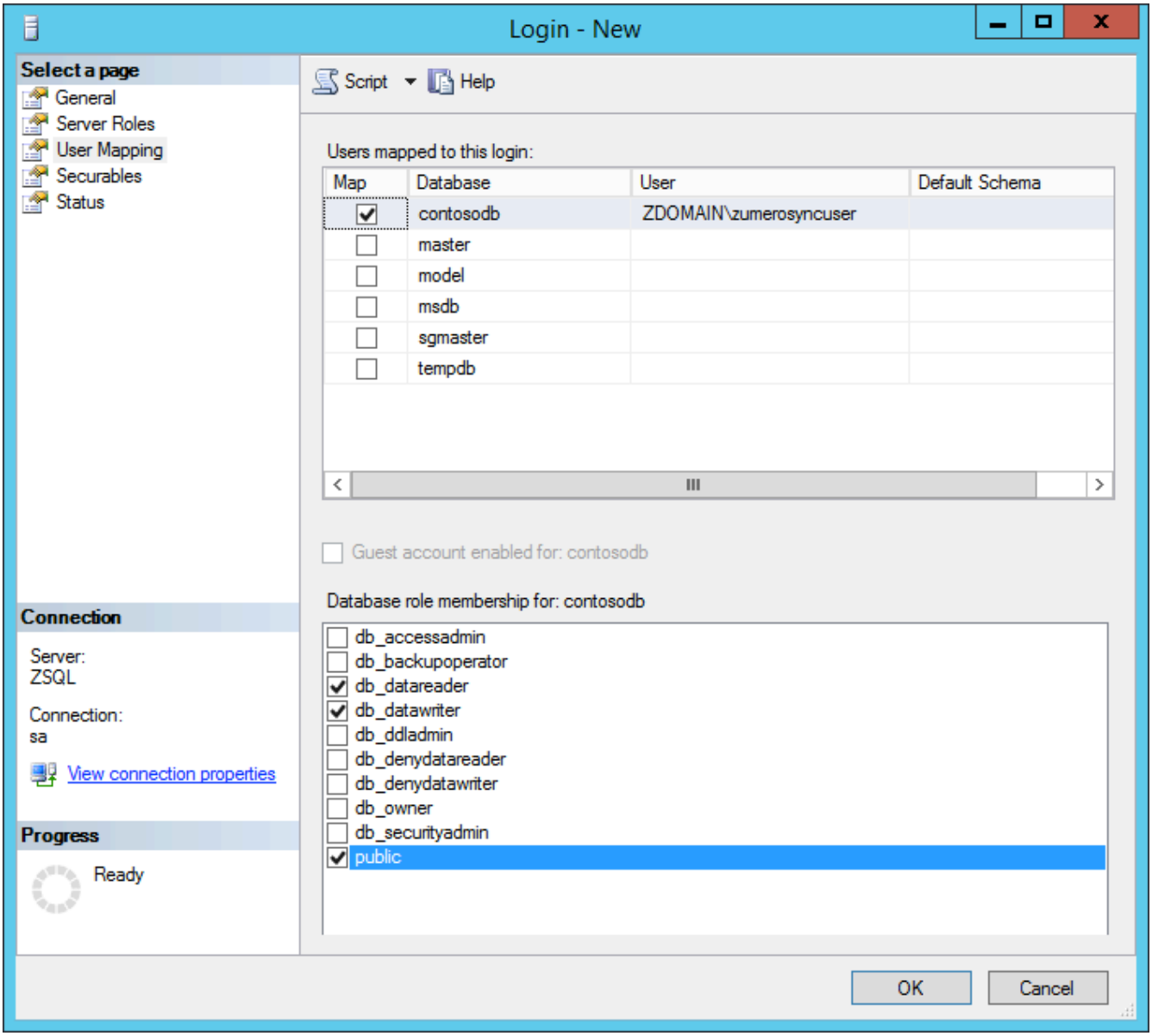
a. Enter "ZDOMAIN\zumerosyncuser" for the **Login name**, and ensure the **Windows authentication** radio button is selected



b. Next select **User Mapping** on the **Select a page** list on the left hand side

c. On the right hand side grid, check the box **Map** for *contosodb*. The user column should be filled with the text "ZDOMAIN\zumerosyncuser"
d. For the **Database role membership for : contosodb**, check the boxes for **db_datareader**, **db_datawriter**, and **public**

4. Click **OK**



Finally, in order for the ZSS Server to view changed data, additional permissions will need to be granted to "ZDOMAIN\zumerosyncuser"[9].

To grant these permissions:

- Within SQL Server Management Studio, open a new **Query with the Current Connection**
- Execute the following statements:

```
USE master
GO
GRANT VIEW SERVER STATE TO "ZDOMAIN\zumerosyncuser"
GO
```

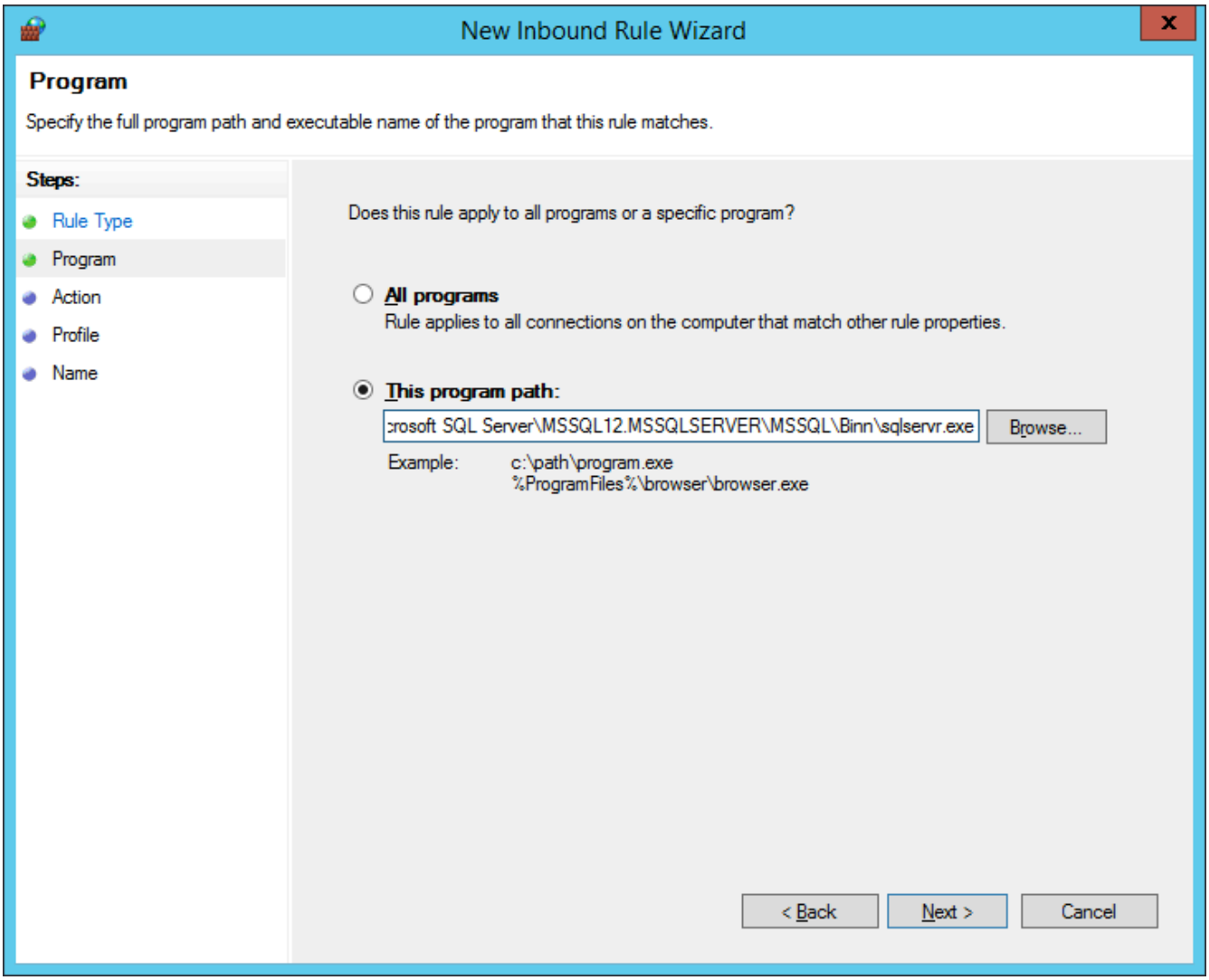
Also note, if any Zumero clients will be inserting rows into a table containing IDENTITY columns, ALTER permissions will need to be granted to the ZDOMAIN\zumerosyncuser user as well (ex: `GRANT ALTER ON contosodb.bottling.products TO "ZDOMAIN\zumerosyncuser"`). Since the tables used in the migrated sample application will be read only, this step is omitted.

After SQL Server has been installed and configured, ZSQL's firewall must be configured to allow SQL Server to accept incoming requests from the ZSS Server[10].

To configure ZSQL's firewall:

1. Open the **Windows Firewall** control panel item
2. In the left pane, click **Advanced Settings**
3. In the **Windows Firewall with Advanced Security** dialog box, in the left pane, click **Inbound Rules**, and then, in the right pane, click **New Rule**
4. In the **New Inbound Rule Wizard** dialog, select the radio button to create a **Program** rule, then click **Next**

5. Enter: "<PATH\TO\SQLSERVER\INSTALLATION\DIRECTORY>\MSSQL\Binn\sqlservr.exe" in the **This program path** field, then click **Next**

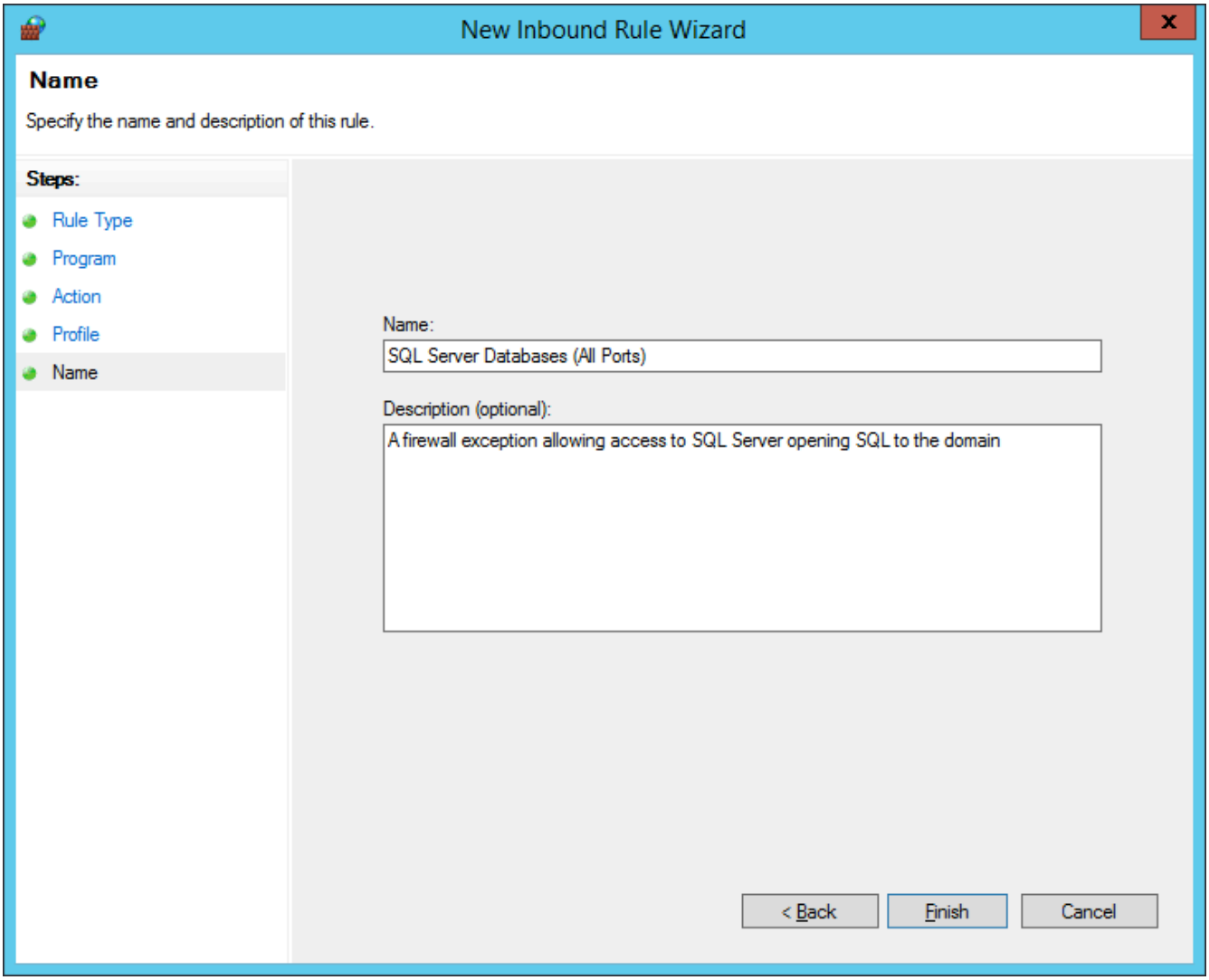


6. On the next page, ensure **Allow the connection** is selected, then click **Next**

7. Ensure **Domain** is checked. **Public** and **Private** should be unchecked, then click **Next**

8. Enter the new rule's **name** of "SQL Server Databases (All Ports)"

9. Click **Finish**



ZSS Server Installation and Configuration

The ZSS Server is an IIS module requiring a Windows 2008 R2 Server running IIS. It is not required that the ZSS Server be installed on the same machine as SQL Server. This is an important point, as best practice would be to deploy and secure the IIS server on a different physical machine in a DMZ and leave the SQL Server behind the DMZ's firewall. This will require some additional server configuration to connect ZSS Server to SQL Server.

IIS needs to be configured with firewall and security settings that allow access to mobile devices needing to synchronize with the ZSS Server. For most installations, the ZSS Server will be installed and configured as a public web site, allowing Anonymous access. However, it can also be restricted with a virtual private network or other network device in front of the server as long as devices can access IIS through HTTP/HTTPS. When configured to use SSL, network traffic will be encrypted between Zumero clients and the ZSS Server. It is recommended to acquire and install an SSL certificate from a trusted Certificate Authority.

To install ZSS Server, first verify the ZWEB Server has added the **Web Server (IIS)** and **IIS Management Tools** roles. Once installed, test the ZDOMAIN\zumerosyncuser account by logging on to ZWEB with the zumerosyncuser's credentials. If the log on succeeds, log off, and log back on using a domain administrative account.

Using a browser, download the ZSS Server installer[13]. While on the download page, request a free Zumero evaluation license by providing an email address and clicking the **Request Key** button.

Once the evaluation license key has arrived via email and the installer downloaded to ZWEB**:**

1. Start the ZSS Server installation

2. Review and agree to the license terms and conditions

3. The installation process will then create a “zumero” website on port 8080

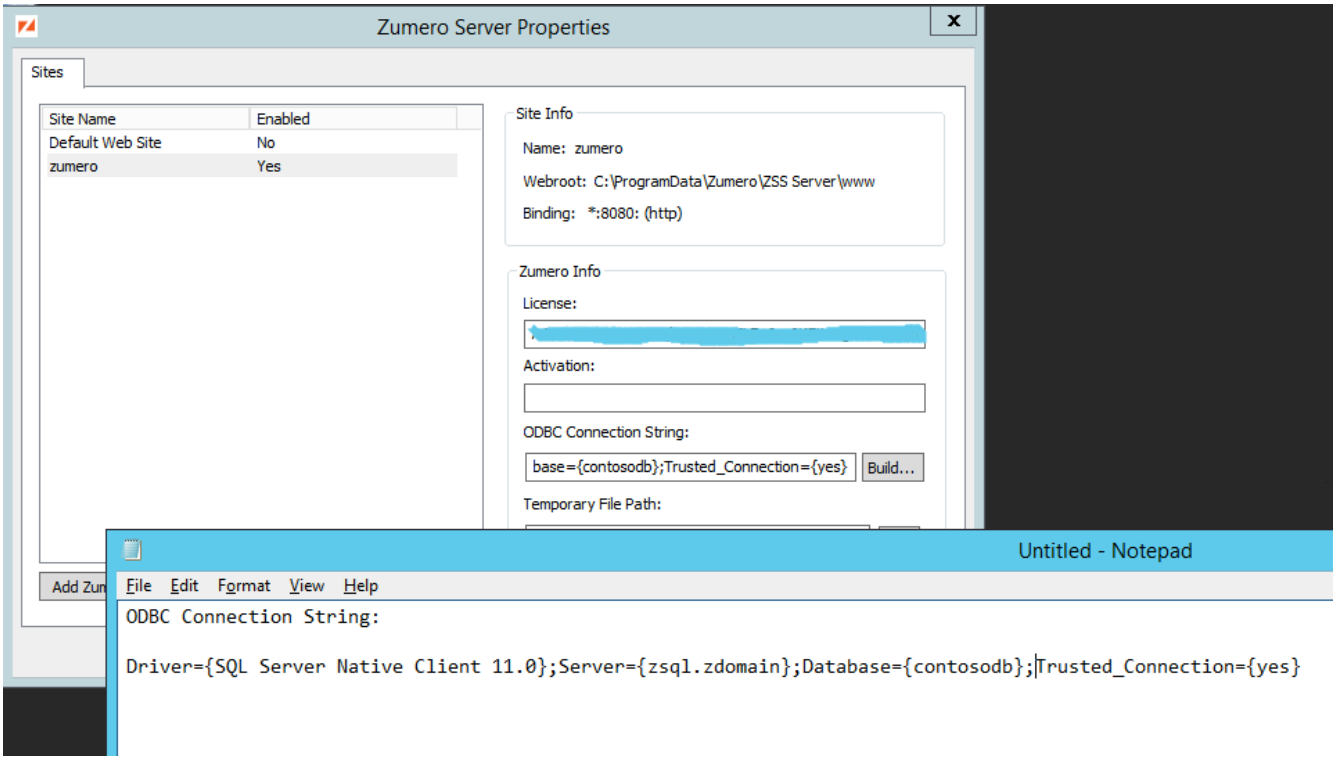
4. Once the install completes, click the **Configure Server** button

5. Within the **Zumero Server Properties** dialog:

    a. Select the "zumero" site, and enter the evaluation license into the **License** text box
    b. Leave the **Activation** field blank
    c. To create an **ODBC Connection String** for the "zumero" site:

        i. Click the **Build** button
        ii. On the first wizard page, enter the domain name for ZSQL, which **hosts the target database**
        iii. Enter "contosodb" for the **name of the target database**
        iv. Click **Next**
        v. Select "Use Windows authentication" for **How should ZSS be authenticated by SQL Server?**
        vi. Click **Finish**

6. Click **OK**



The **Zumero Server Properties** dialog displays the IIS website where the ZSS Server will be running. Each site displays the physical disk location and port. The **Enabled** column signifies which websites are capable of ZSS synchronization. The installation's default ZSS site is named “zumero” located on port 8080.

The **Temporary File Path** can be set to a disk location on the file system. The ZSS Server uses the location to temporarily store data when communicating with Zumero clients as well as SQL Server.

The next set of steps will change the ZSS Server's Application pool to run under the “zumerosyncuser” account created in the previous section:

1. From the Start menu, type “inetmgr.exe” to start the IIS Manager utility

2. Expand the ZWEB Connection

3. Select **Application Pools**

4. In the Application Pools listing, select the “zumero” Application Pool

5. Click the **Advanced Settings** link on the right hand side

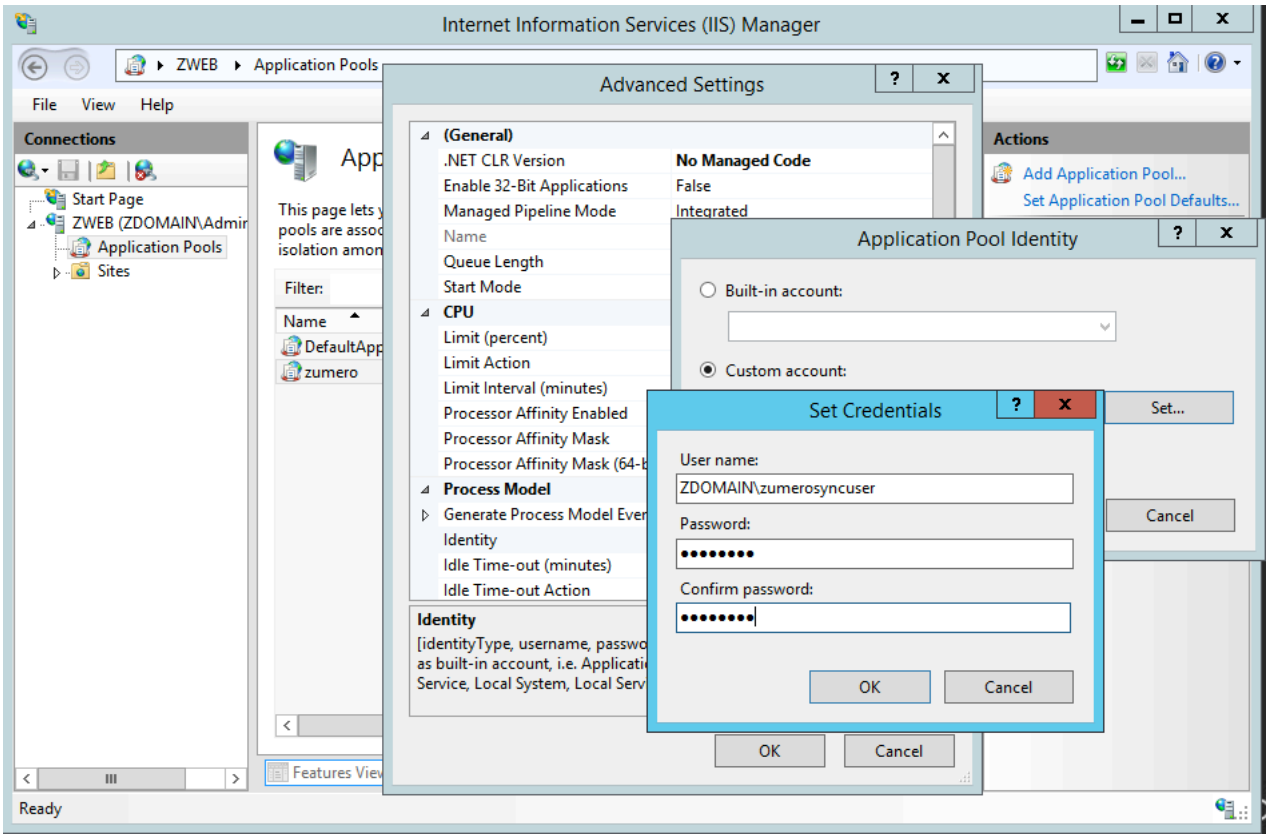
6. Find and select the **Identity** property on the **Advanced Settings** dialog

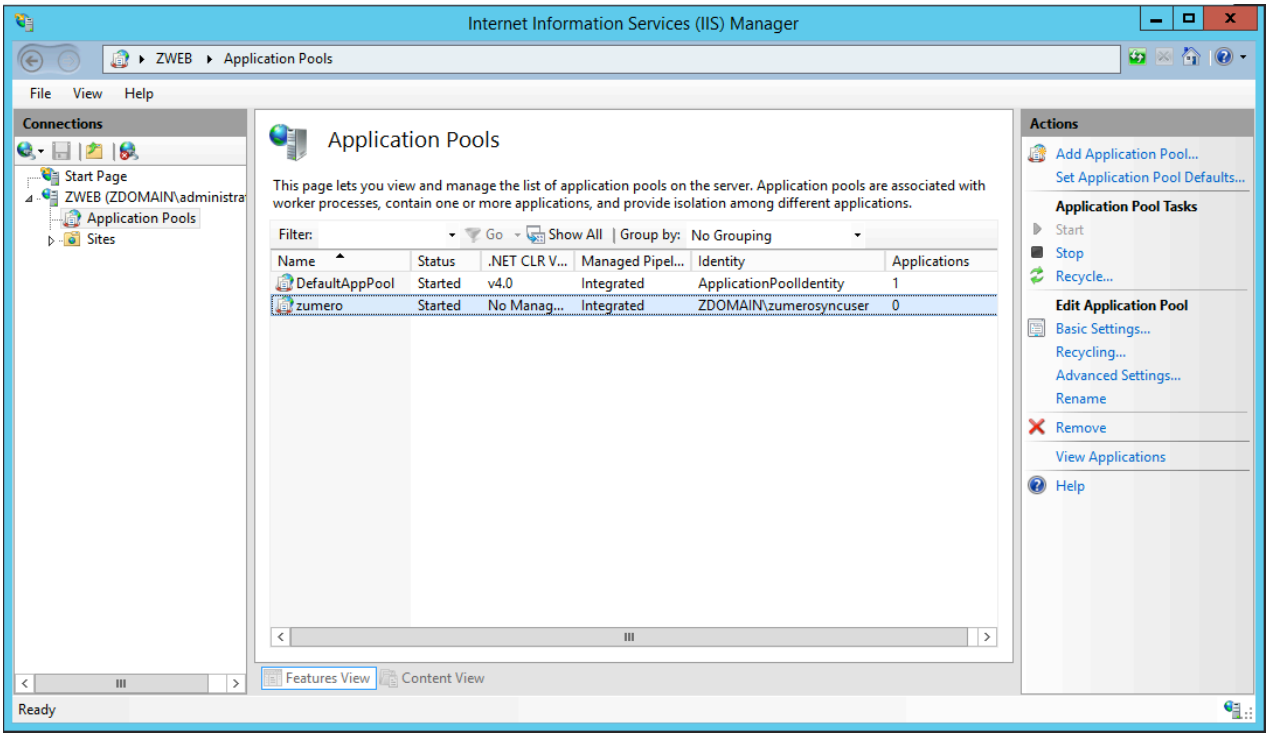
7. Click the ellipsis button

8. On the **Application Pool Identity** dialog, select the **Custom account** radio button, and then click the **Set...** button

9. In the **Set Credentials** dialog enter “ZDOMAIN\zumerosyncuser” for the **User name**, “P@ssw0rd” for the password and confirmation

10. Select **OK** on all of the dialogs: **Set Credentials**, **Application Pool Identity**, and **Advanced Settings**





If a firewall is being used, the last step is to open the TCP/IP port for Zumero network traffic.

To configure ZWEB's Firewall:

1. Open the **Windows Firewall** control panel item

2. In the left pane, click **Advanced Settings**

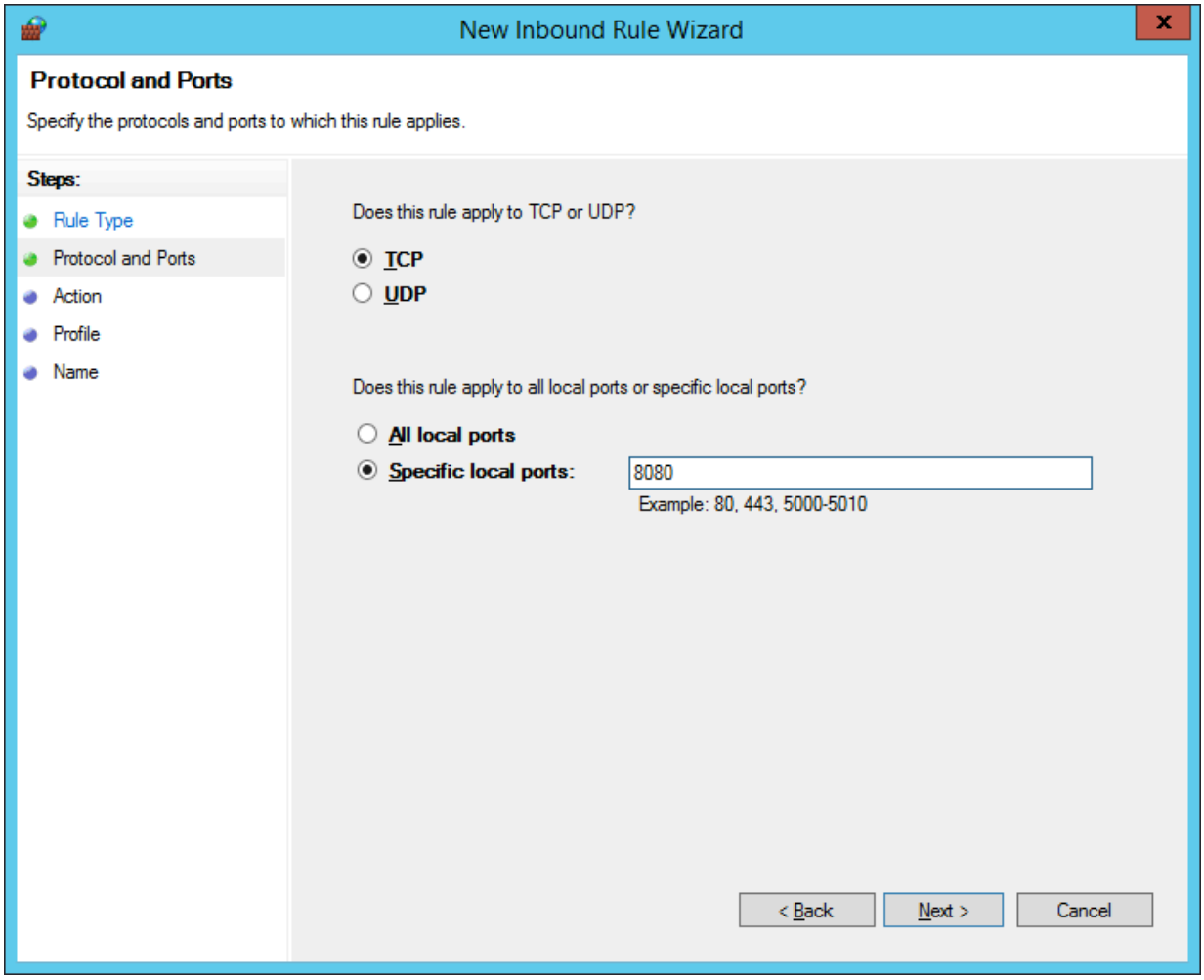
3. In the **Windows Firewall with Advanced Security** dialog box, in the left pane, click **Inbound Rules**, and then, in the right pane, click **New Rule…**

4. In the **New Inbound Rule Wizard** dialog, select the radio button to create a **Port** rule, then click **Next**

5. Ensure **TCP** is selected, then enter **8080** in the **Specific local ports** field, then click **Next**

6. On the next page, ensure **Allow the connection** is selected, then click **Next**

7. Ensure **Domain**, **Public**, and **Private** are all checked, then click **Next**
8. Enter “Zumero Traffic” as the new rule's **Name**
9. Click **Finish**



To test the configuration, from a browser running on ZWEB, try browsing to http://localhost:8080 and http://ZWEB:8080. The resulting webpage should display a quick message:

Zumero for SQL Server

Version <installed version number>

Configuring the Database with ZSS Manager

Unlike SQL Server's Publisher / Distributor model using Snapshots on disk, ZSS maintains all pertinent information regarding synchronization within the database. In this way, a full database backup will also back up the meta-data used by ZSS Server to synchronize with Zumero clients. Hence, there are no Snapshot folders to create or Snapshot folder configuration required by ZSS Server.

SQL Server Merge Replication uses a publishing industry metaphor to represent the components in its replication topology, which includes *Publisher, Distributor, Publications, Articles, Subscribers, and Subscriptions*. While Zumero has similar constructs, the nomenclature for these components is a bit different.

Zumero's smallest "unit" of synchronization is the *DBFile*. A Zumero DBFile is analogous to SQL Server Merge Replication's *Publication*. The DBFile represents a logical grouping of database tables that will sync as a unit with Zumero clients. A DBFile, created with ZSS Manager or a SQL Script, also corresponds to a SQLite database file on each ZSS client device. You can create one or more DBFiles per SQL Server database, but a DBFile cannot span multiple SQL Server databases.

Once a DBFile has been created, the next step is to configure tables for synchronization. This is accomplished by using Zumero's **Prepare Table**. A table can only be prepared for at most one DBFile. A Zumero Prepared Table is analogous to a SQL Server Merge Replication *Article* within a *Publication*. Just as *Articles* are defined to minimize what is replicated from SQL Server, tables required by Zumero client applications are the only tables that should be prepared. This will save bandwidth and device resources, as well as streamline processing during synchronization.

When a table is first prepared for synchronization, all compatible columns (including newer SQL Server data types such as *datetime2* and *geography*) are included and made ready for synchronization. Incompatible data types are *datetimeoffset*, *decimal*/*numeric* having precision greater than 18, *image*, *ntext*, *sql_variant*, *text*, *timestamp*, and *xml*. After a table has been prepared, it can also be configured to synchronize a smaller subset of the table's columns.

A table must have a primary key in order to be eligible for synchronization. Table columns, default values (using literals and *GETUTCDATE*), primary keys, and foreign keys will all be replicated and synchronized by Zumero. Indices are not replicated by Zumero but should be created externally by client applications for performance reasons.

When preparing a table for synchronization, ZSS Manager will:

- Save the table's current schema.
- Create several housekeeping tables for tracking changes to the selected table.
- Create insert, update, and delete triggers on the table allowing ZSS Server to keep track of changes made by other applications.
- Create a view (with triggers) that the ZSS Server uses to query and modify the selected table when a Zumero client syncs.
- Save transaction data, effectively pretending that all existing rows in the selected table were inserted in a Zumero transaction.

Using a browser, download ZSS Manager from http://zumero.com/dev-center/zss/ and install it on a machine (such as ZWEB) with access to ZSQL.

The following steps will create the DBFile for our sample application:

1. From the Start menu, type "ZssManager.exe" to start ZSS Manager

2. ZSS Manager will prompt for credentials for SQL Server. Verify the **Server Name** is the name of the SQL Server instance, in this case, ZSQL. Enter in the proper credentials, and click **Connect**

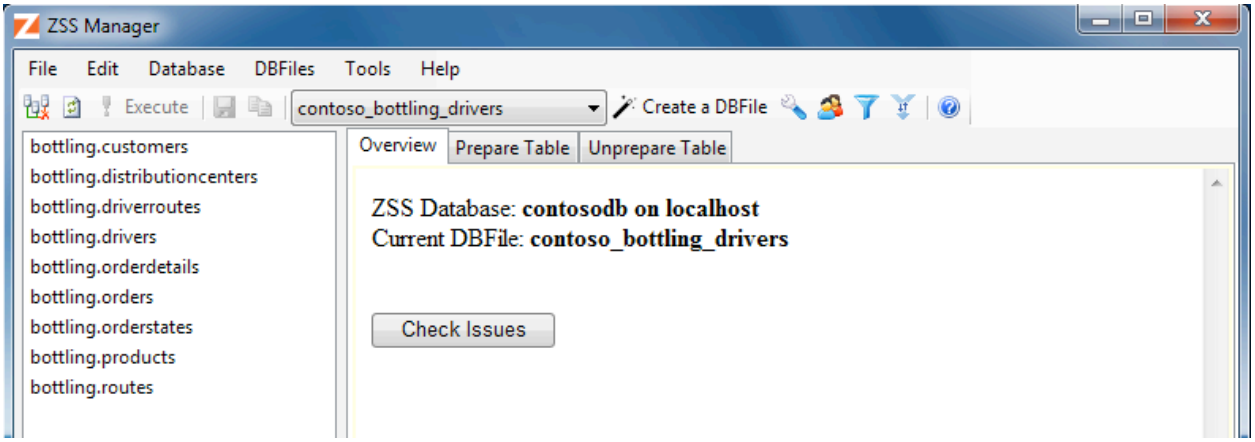
3. After ZSS Manager has connected, select the *contosodb* database. This ensures any commands invoked in ZSS Manager will take place in the *contosodb* database

4. From the toolbar, select **Create a DBFile**, which launches the **Create a DBFile** wizard dialog

    a. Enter "contoso_bottling_drivers" in the **DBFile Name** text box and select **Next**

b. Select **The "contosodb" database** and select **Next**
c. Review the steps on the final wizard page and select **Next**
d. If the DBFile was created successfully, select **Done**

5. The **Overview** tab of ZSS Manager should now display *contoso_bottling_drivers* as the **Current DBFile** and the **User and Permissions** toolbar button will become enabled



After the DBFile has been created, prepare the tables for synchronization:

1. Select the **Prepare Table** tab

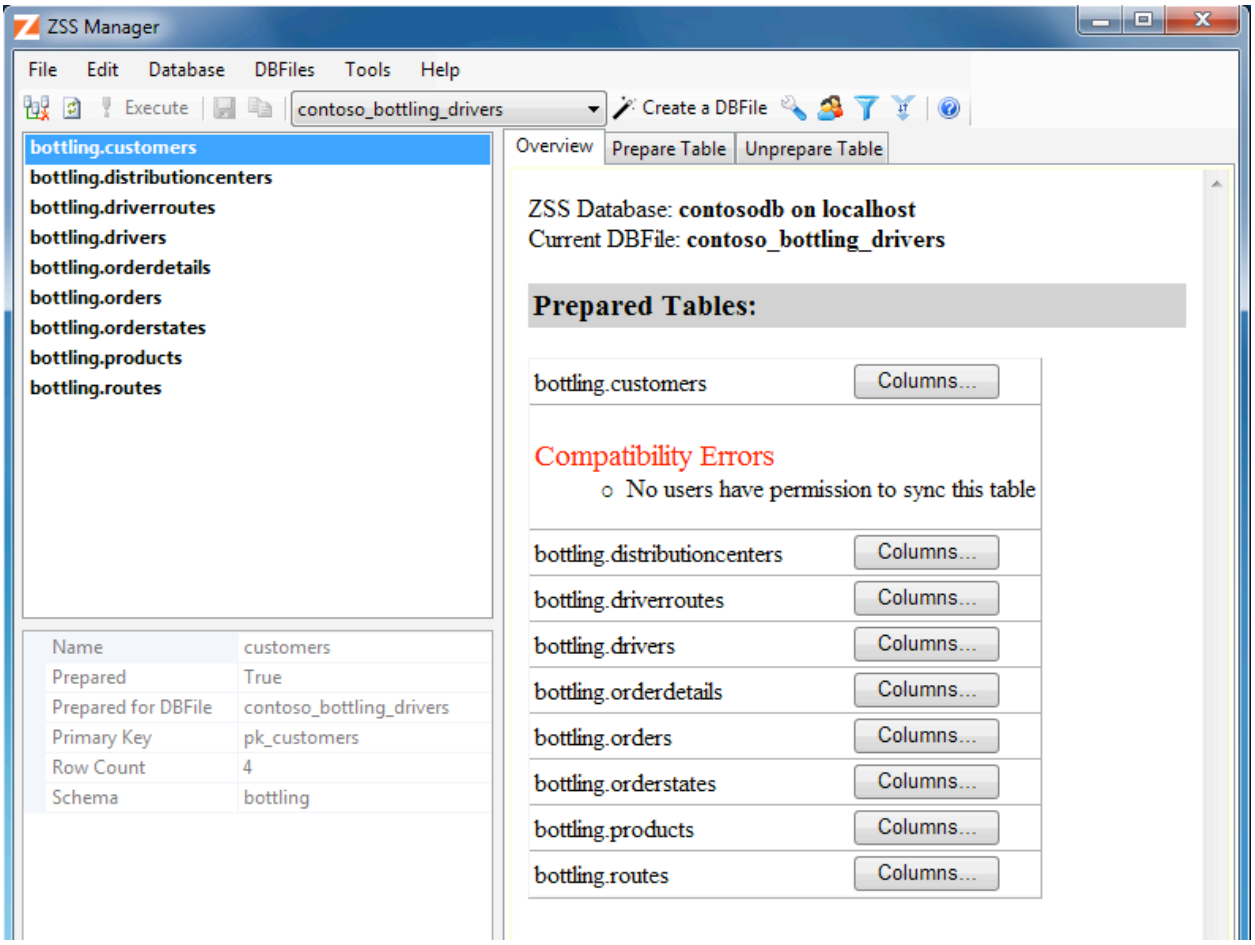
2. On the left, select the *bottling.distributioncenters* table

3. Click the **Execute** button, and **OK** to execute the script

4. The *bottling.distributioncenters* list item text should now be bold, and the status message should state the Table Preparation process was successful

Now using steps 2 & 3 from above, prepare the rest of the tables for replication in the following order:

- *bottling.products*
- *bottling.orderstates*
- *bottling.routes*
- *bottling.customers*
- *bottling.drivers*
- *bottling.driverroutes*
- *bottling.orders*
- *bottling.orderdetails*

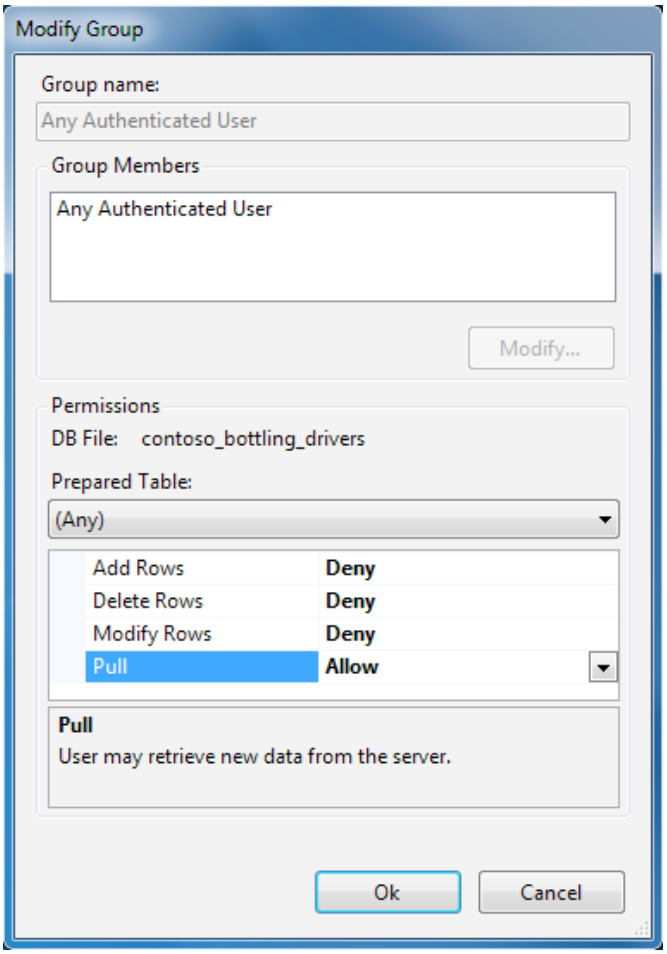


To configure user permissions, click on **Users and Groups** from the tool bar, or from the menu select **Tools → Users and Groups**. If not already created, a dialog will ask **Create database authentication tables now?** Click **Yes**.

Next, a dialog will display the list of named users and groups that can access the SQL Server database. For our sample, users will need to read data from all tables in the database:

1. Select **Any Authenticated User** in the **Groups** list box, and click **Modify...**

2. In the **Modify Group** dialog, a list of permissions is displayed for any user authenticated with the ZSS Server. Ensure the **Prepared Table** is set to *(Any)* and change the values for **Add Rows**, **Delete Rows**, and **Modify Rows** to **Deny**, and change the **Pull** permission to **Allow**



At this point, all prepared tables have been set to read-only, and no changes will be synchronized back into SQL Server.

Next:

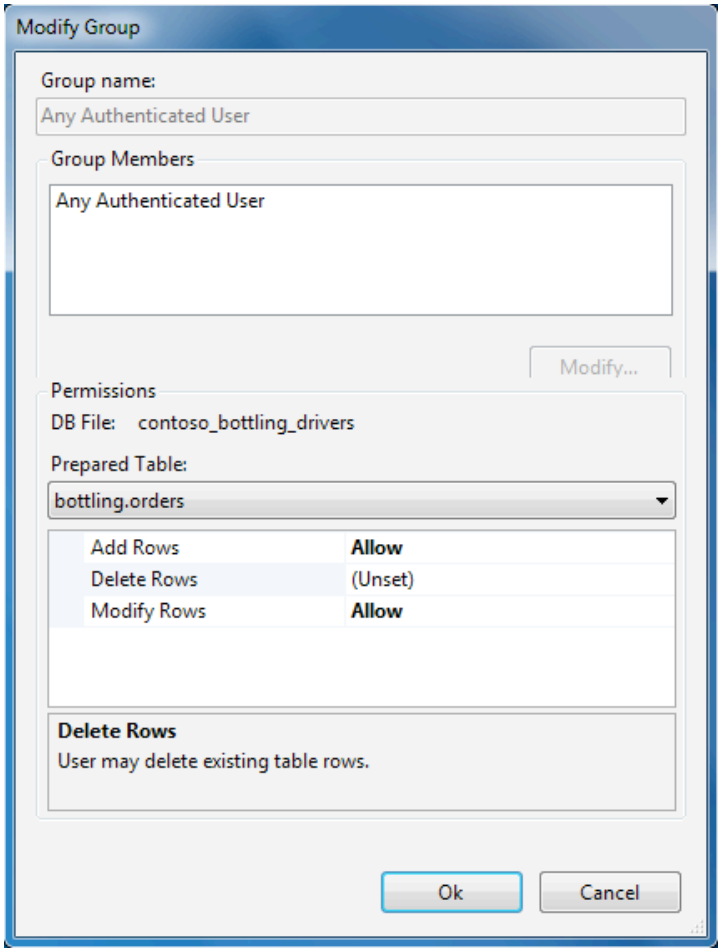
1. From the **Prepared Table** drop down combo box, choose *bottling.orders*, and set the permission for **Add Rows** and **Modify Rows** to **Allow**

2. Leave **Delete Rows** as **(Unset)**

3. From the **Prepared Table** drop down combo box, choose *bottling.orderdetails*, and set the permission for **Add Rows**, **Delete Rows** and **Modify Rows** to **Allow**

4. Click **OK** to save the changes

5. Click **Done**



Zumero provides options for controlling how conflicts are handled between clients and server during synchronization. From the toolbar, click on **Conflict Rules** or from the menu choose **Tools → Conflict Rules…**. Within the **Conflict Rules** dialog is an interface for controlling what happens when a conflict is encountered during a synchronization operation. Our sample will use the default values.

For more information on Zumero's conflict resolution features, see:

http://zumero.com/docs/zumero_for_sql_server_manager.html#conflict-resolution

Filtering User Data

One of the most powerful features of Zumero for SQL Server is the ability to filter the data being synchronized from the server to client devices.

Benefits of filtering include:

- Preventing users from gaining access to data from which they should be restricted
- Sending different sets of data to mobile devices based on user credentials
- Reducing the amount of synchronized data to save bandwidth and limit resource requirements on the device

When a filter is created, it automatically applies the filter to any dependent tables via foreign key. So a filter does not necessarily need to be created on each and every table.

In the sample, contoso drivers will only receive a subset of the information filtered by the driver's login. The mobile application's SQLite database should only contain information relevant to that driver: the driver's routes, customers, and orders.

Since the contosodb sample database contains foreign keys, only three filters need to be created:

1. Click **Filters** on the toolbar, or from the menu choose **Tools → Filters**

2. Within the dialog, click the **Add New Filter** button to start the **Add Filter** wizard.

3. Select **Any Authenticated User Filter** for the kind of filter to create and then select the button **Edit New Filter**

4. Select *bottling.drivers* from the **Prepared Tables** list. For the **Row Inclusion WHERE Clause** enter:

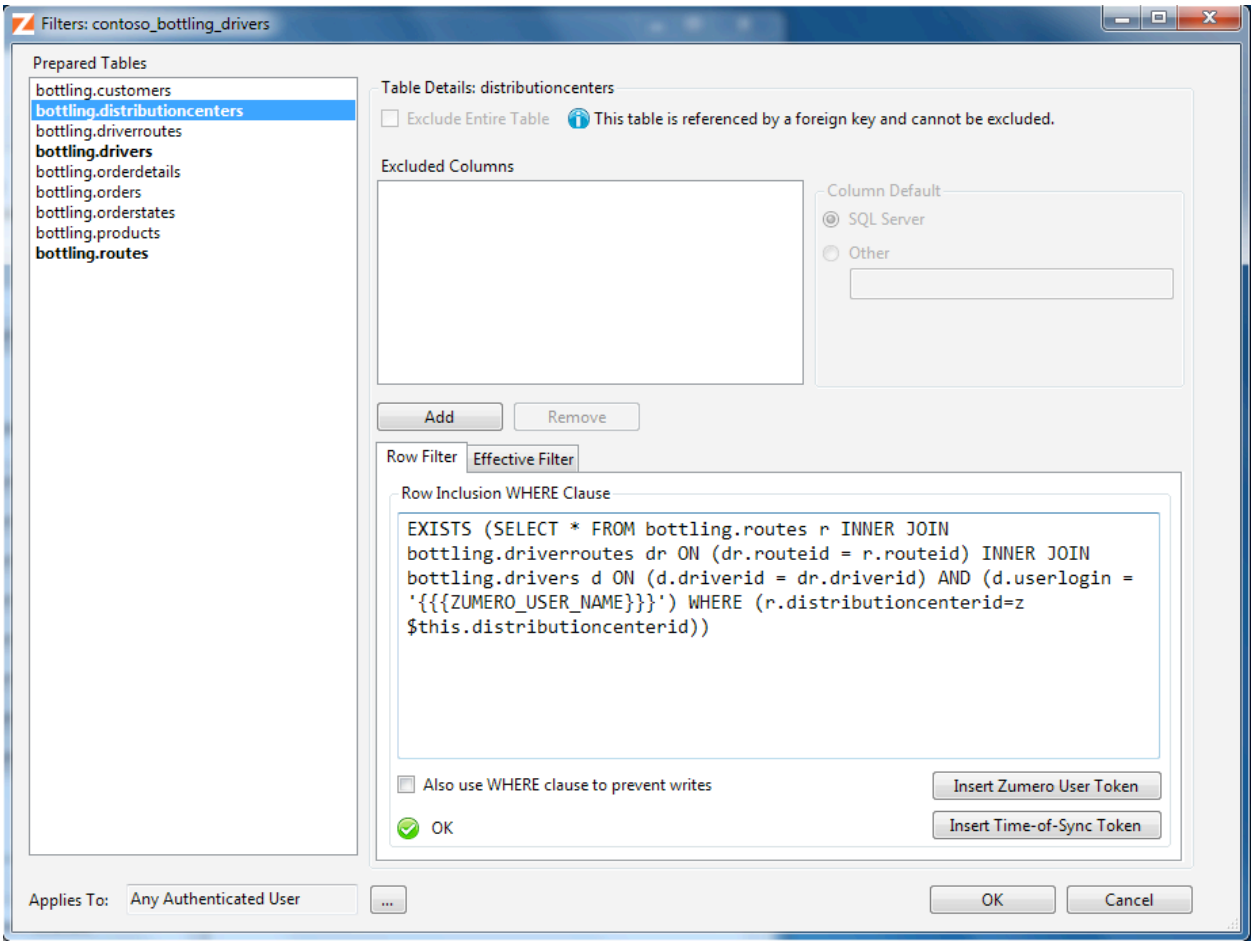
```
(userlogin = N'{{{ZUMERO_USER_NAME}}}')
```

5. Select *bottling.routes* from the **Prepared Tables** list. For the **Row Inclusion WHERE Clause** enter:

```
EXISTS (SELECT * FROM bottling.drivers d INNER JOIN
bottling.driverroutes dr ON (dr.driverid =
d.driverid) AND (dr.routeid = z$this.routeid) WHERE
(d.userlogin = N'{{{ZUMERO_USER_NAME}}}'))
```

6. Select *bottling.distributioncenters* from the **Prepared Tables** list. For the **Row Inclusion WHERE Clause** enter:

```
EXISTS (SELECT * FROM bottling.routes r INNER JOIN
bottling.driverroutes dr ON (dr.routeid =
r.routeid) INNER JOIN bottling.drivers d ON
(d.driverid = dr.driverid) AND (d.userlogin =
'{{{ZUMERO_USER_NAME}}}') WHERE
(r.distributioncenterid=z$this.distributioncenterid
))
```



These filters will limit synchronized data to a subset of all table information based on the distribution center, routes, customers, and orders for a particular driver. The filters need only to be placed on three different tables. Due to the definitions of the foreign keys, the remaining tables will automatically have any table rows restricted by the filter of the dependent tables.

One particular item of interest is the filter directly placed on *bottling.routes*. By default, the filter on *bottling.distributioncenters* will automatically filter the route information to those distribution centers in which the driver has a route. However, by itself, the synchronized route table will contain additional route information for other drivers assigned to routes of the distribution centers. To further limit the table row information for a particular driver, the additional filter is required.

User Authentication

Zumero provides features[16] for authenticating users before allowing them to sync with the ZSS Server. In the migrated sample, in order to support driver authentication, an extra column, `userlogin nvarchar(100) NOT NULL`, has been added to the *bottling.drivers* table. The login scheme uses a driver's first initial concatenated to the last name. So, “Rob Tiffany” becomes “RTiffany”, “Liam Cavanaugh” becomes, “LCavanaugh”, and so on.

Using ZSS Manager, perform the following actions:

1. From the toolbar, click on **Users & Groups**, or from the menu choose **Tools → Users & Groups...**

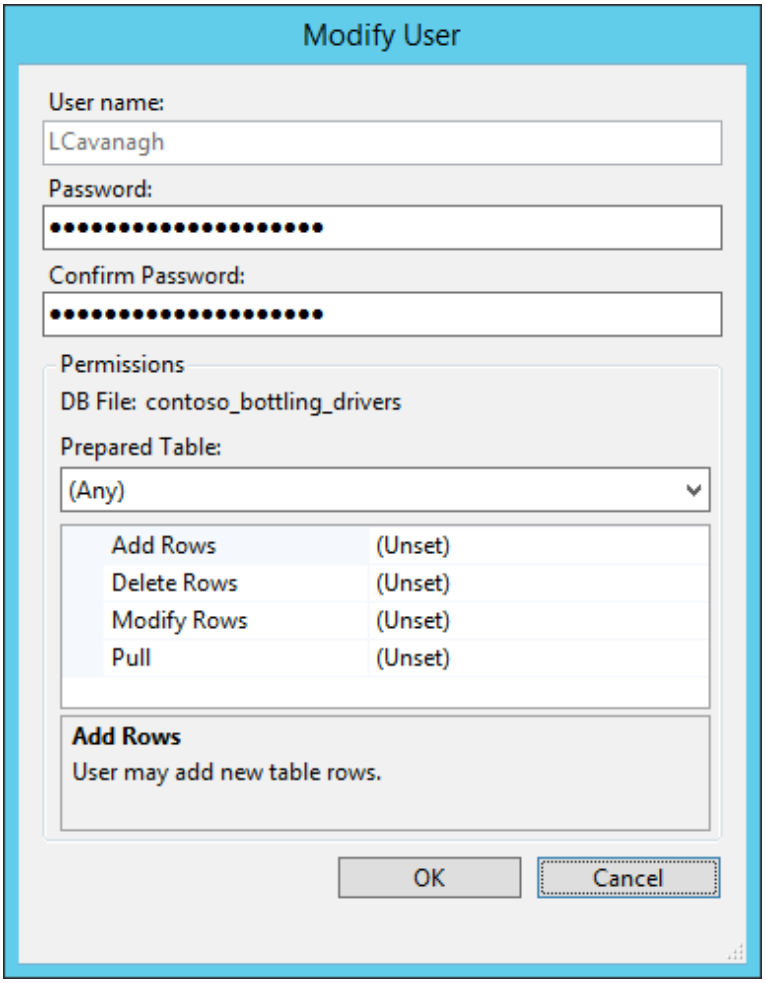
2. Within the **Users** dialog, click the **Add...** button

3. On the subsequent dialog, enter *RTiffany* for the **User name**, and create a password for the user. Do not change any settings for Permissions

4. Click **OK**

The **Users & Groups** dialog allows for fine grain control of security permissions. However, since permissions were assigned previously using **Any Authenticated User**, no extra steps are required.

Use the previous steps to create a Zumero User for *LCavanagh*, and the other drivers found in **contoso.bottling.drivers**. All created users should have the same Zumero User name as found in the **userlogin** column in that table. The Zumero User name requires an exact match as the login passed in with the Zumero synchronization request and will also be applied to any filtering using the '{{{ZUMERO_USER_NAME}}}' placeholder.



Migrating the Client-Side Mobile App

The Contoso Bottling sample application[2] referenced in *Enterprise Data Synchronization with Microsoft SQL Server 2008 and SQL Server Compact 3.5 Mobile Merge Replication* contains a Visual Studio 2008 SP1 solution with a Smart Device Project in C#. This project targets Windows Mobile 6 Professional.

The app starts by launching *frmMain*, a Form displaying some information about the local SQL Server Compact database and a **Sync Now** button. The project also includes *DatabaseManager*, a singleton class used to handle the implementation for the SQL Server Compact database, and *DAL*, a class containing static methods to gather meta-data regarding synchronization operations.

When the **Sync Now** button is clicked, the event handler displays a second form, *Synchronize*. Within this form, `Synchronize._doSynch()` initiates a sync operation between the local SQL Server Compact database and SQL Server using an instantiated *SqlCeReplication* object. Information required by the synchronization process such as IIS Server URL, Replication Distributor, Publication, Publisher, Subscriber, etc. is stored in the project's config.xml file. Other SQL Server Compact database credentials can be found in the *Credentials* class with some variables referenced in Constants.cs.

In contrast, Contoso Bottling for ZSS, *ContosoMobile_Z*, is a Visual Studio 2013 solution consisting of multiple Xamarin.Forms projects in C#. The solution includes projects for targeting iOS and Android, as well as Windows Phone 8.x. The sample code can be downloaded from http://zumero.com/dl/ContosoMobile_Z.zip

Conceptually, the two application samples are very similar. ContosoMobile_Z consists of a form with a **Sync Now** button, and a dialog that displays information during sync operations. However, the ContosoMobile_Z project is designed using the *ModelView-View-Model* architectural pattern (MVVM) and includes the following:

- The ContosoMobile_Z/Pages/ folder contains MainPage.cs and SynchronizeNowPage.cs that comprise the main UI for the app. (Xamarin.Forms displays UI in *Pages*.)

- ContosoMobile_Z/ViewModels/ contains the Pages' View-Models.

MainPageViewModel.cs and SynchronizeNowPageViewModel.cs contain classes to bind the data components, and commands to interact with their UI pages. ContosoMobileViewModelNavigator.cs contains a singleton class used to display and close UI pages based on user interaction.

- ContosoMobile_Z/DataServices/ contains a base class and two distinct database components. LocalContosoMobileDB.cs is a base class for interacting with the local SQLite database, and LocalContosoMobileDB_Z1.cs is a subclass that implements calls to the local database through the *SQLite-net* ORM library. Likewise, LocalContosoMobileDB_Z2.cs also inherits from *LocalContosoMobileDB*, but interacts with the database using the *SQLite-pretty* library – an ADO.NET stylized library.

- ContosoMobile_Z/Services/ contains classes used to create and provide access to the local database in LocalContosoMobileDBService.cs, and to provide background data synchronization through BaseSyncService.cs

ContosoMobile_Z's **SynchronizeNowPage** allows the user to enter the ZSS Server URL and login credentials. Once the ZSS Server and user credentials have been entered, the **Synchronize** button's click event will trigger the app to synchronize the SQLite database with the Zumero Server.

Synchronization with the ZSS Server occurs on a background thread. This is represented in the `BaseSyncService.ZumeroSync()` method. However, on the Android platform, background operations are tied to a construct called an *Activity*. The application uses Xamarin.Forms' *DependencyServices* to create the Synchronize Form's *Activity* in order to perform the sync operation in the background. See ContosoMobile_Z.Droid/Services/SyncService.cs for additional details.

Zumero Application Generator

Another great resource for developers is the Zumero Application Generator (ZAG), a desktop application with support for OS X, Ubuntu and Windows. ZAG can generate sample mobile apps for varying languages and platforms based on database tables prepared by the ZSS Manager. The generated code is a fantastic reference for developers to use when starting with Zumero.

A video tutorial for ZAG can be found at http://zumero.com/howto/zag/xamarin/

To use ZAG with the Contoso Drivers database, follow these steps:

1. Download and extract the ZAG utility for the correct platform and start the *zag* application.

2. If a local SQLite database had previously been created, the database can be opened via the menu **File → Open Database**. However, as this is a first run, choose **File → New Database**. This will bring up the **Sync from ZSS** dialog.

3. Within the **Sync from ZSS** dialog enter the following information:

   - Server URL: "http://ZWEB:8080"
   - DBFile: "contoso_bottling_drivers"
   - Use Authentication: <checked>
   - Scheme: "{"scheme_type":"table","table":"users"}"
   - Username: "RTiffany"
   - Password: *<Rob Tiffany's password within ZSS>*
   - Click **OK**

4. Save the database to a known directory. For example, %USERPROFILE%\Documents\ZAG\ContosoBottlingDriver\ContosoBottlingDriver.zssdb

5. Next click the **Database Schema** tab. After a database has been loaded into ZAG, the **Database Schema** tab will list the tables that were configured with ZSS Manager. Once a table within the tree has been selected, ZAG

will display the column name as well as other additional SQLite information including data type, precision, scale, and width.

6. Select the table *bottling_orderdetails*, and examine the *quantity* column. The SQL Server column *orderdetails.quantity* is of type *decimal (10, 2)*. However, the SQLite column has a data type of *integer* with precision of 10 and scale of 2. Because SQLite has no exact floating point datatype, columns in ZSS synchronized tables need a slight conversion. This is true for any SQL Server column of type *decimal*, *numeric*, *money* and *smallmoney*. For additional information, see: http://zumero.com/docs/zumero_for_sql_server_manager.html#decimal-numeric-money-and-smallmoney-types

7. From the menu, select **Generate → Demo1 → Xamarin.Forms C#**

8. Enter "ZagContosoBottlingDriver" for the Project Name and click **OK**

9. On the **Server Settings as Seen from Device** dialog enter the following information and click **OK**:

    - Server URL: "http://ZWEB:8080"
    - DBFile: "contoso_bottling_drivers"
    - Use Authentication: <checked>
    - Scheme: "{"scheme_type":"table","table":"users"}"

10. Save the project to a known directory. For example, %USERPROFILE%\Documents\ZAG\ContosoBottlingDriver\ZagContosoBottlingDriverApp

At this point, a Visual Studio / Xamarin Studio solution has been saved to disk. The solution is ready to be loaded within the IDE. Here are the steps for loading the solution within Visual Studio 2013:

1. Start Visual Studio 2013

2. From the menu, choose **File → Open → Project/Solution...**

3. Browse to and open ...\ZagContosoBottlingDriverApp\demo1__xamarin.forms\ZagContosoBottlingDriver.sln

4. Next, select the *ZagContosoBottlingDriver (Solution)* node in Solution Explorer

5. From the menu, select **Tools → NuGet Package Manager → Manage NuGet Packages for Solution...**

6. On the **ZagContosoBottlingDriver.sln - Manage NuGet Packages** dialog:

    a. Select **Installed packages** on the left hand side list view, and then click the **Restore** button found on the upper right of the dialog. This will download any missing library packages for the solution.
    b. Click **OK**

Once loaded, developers can build and debug the project on a device or device emulator. When the generated app is started, its main page displays some instructions. When the **Sync** button is clicked, a new page is displayed asking for credentials. For the purposes of this tutorial, enter "RTiffany" / "rtiffany" for the authentication credentials, and click **Sync Now**. The resulting page should list the synchronized tables:

- *bottling_distributioncenters*
- *bottling_routes*
- *bottling_customers*
- *bottling_drivers*
- *bottling_driverroutes*
- *bottling_products*
- *bottling_orderstates*
- *bottling_orders*
- *bottling_orderdetails*

As each table is selected the app should display the filtered data for the RTiffany driver. Make note of the rows in *bottling_drivers*, *bottling_driverroutes*, and *bottling_routes*. Shut down the device/emulator. [Note: Uninstall the sample application if the actual device or emulator persists data between runs. This will remove the SQLite database file (data.db3) from the device.]

Now run the app again, but this time use different credentials for "LCavanagh" / "lcavanagh". Again, examine the tables after the synchronization operation, especially *bottling_drivers, bottling_driverroutes*, and *bottling_routes*. Notice the table rows will contain a different set of filtered data, this time for LCavanagh.

Conclusion

Businesses that were left stranded when SQL Server Compact eliminated support for mobile platforms and merge replication now have a migration path for their mobile apps using Zumero for SQL Server. Zumero provides a replacement replication and synchronization solution for SQL Server, and supports all of the latest mobile platforms such as iOS, Android, and Windows 8 Phone.

Zumero replicates SQL Server data to mobile devices where the data is stored locally in SQLite. Mobile apps then work directly with the data in SQLite and Zumero handles data synchronization, merging, and confliction resolution, when the devices are online. Zumero also provides features for filtering data and ensuring security through encryption, authentication, and permissions.

Developers familiar with SQL CE and Merge Replication can easily migrate their mobile apps to Zumero by following the steps outlined in the previous chapters. While some details and terminology between the two solutions are different, many of the core concepts and design principles are the same. Zumero has also provided the source code for a sample mobile app that was migrated from SQL CE and Merge Replication to Zumero.

Zumero provides many other technical resources, including video tutorials and documentation, online at the ZSS dev center, http://zumero.com/dev-center/zss/. Technical questions related to this document or Zumero development in general can be sent to support@zumero.com. In addition, Zumero also offers development services to help businesses migrate their mobile apps from SQL CE to Zumero. Please contact sales@zumero.com or visit http://zumero.com/services/ for more information.

1 - http://www.amazon.com/Enterprise-Synchronization-Microsoft-Compact-Replication/dp/0979891213
2 - http://cid-8b9c82da88af61fc.skydrive.live.com/self.aspx/Public/ContosoBottling.zip or https://onedrive.live.com/?id=8B9C82DA88AF61FC!298&cid=8B9C82DA88AF61FC&group=0&parId=8B9C82DA88AF61FC!218&o=OneUp
3 - http://en.wikipedia.org/wiki/Contoso
4 - http://blogs.msdn.com/b/sqlservercompact/archive/2011/01/12/microsoft-sql-server-compact-4-0-is-available-for-download.aspx
5 - https://msdn.microsoft.com/en-US/library/bb500342(v=sql.120).aspx
6 - http://www.zumero.com
7 - http://zumero.com/dev-center/zss/
8 - http://zumero.com/faq/#identitycolumns
9 - http://zumero.com/docs/zumero_for_sql_server_manager.html#permissions
10 - https://msdn.microsoft.com/en-us/library/cc646023.aspx
11 - https://support.microsoft.com/en-us/kb/322385
12 - http://blogs.msdn.com/b/joesack/archive/2009/01/28/sql-server-and-hba-queue-depth-mashup.aspx
13 - http://zumero.com/dev-center/zss/
14 - http://zumero.com/docs/zumero_for_sql_server_manager.html#default-values
15 - http://zumero.com/docs/zumero_for_sql_server_manager.html#warnings
16 - http://zumero.com/docs/zumero_for_sql_server_manager.html#create-a-user-and-set-permissions
17 - http://zumero.com/dev-center/zss/#zag