



## ODBC And SQL

V4.x

06/30/2005 Document v1.01

### Overview

The purpose of this document is to provide a basic understanding of how Kepware's ODBC driver works with Microsoft SQL. This is a quick reference document, and not a detailed instruction manual.

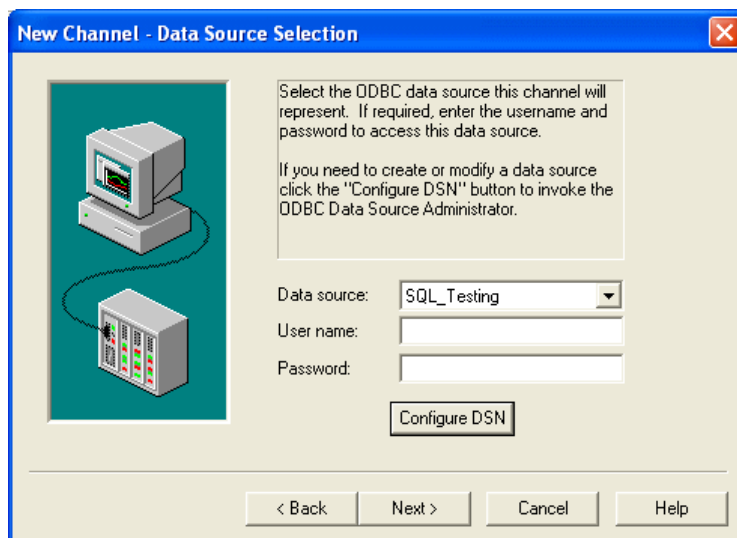
### Getting Started

The first topic we will cover is setting up a project within the ODBC driver. The ODBC driver uses the ODBC API, which requires the MDAC be installed. MDAC consists of several core components that provide various database technologies; including ODBC and its drivers.

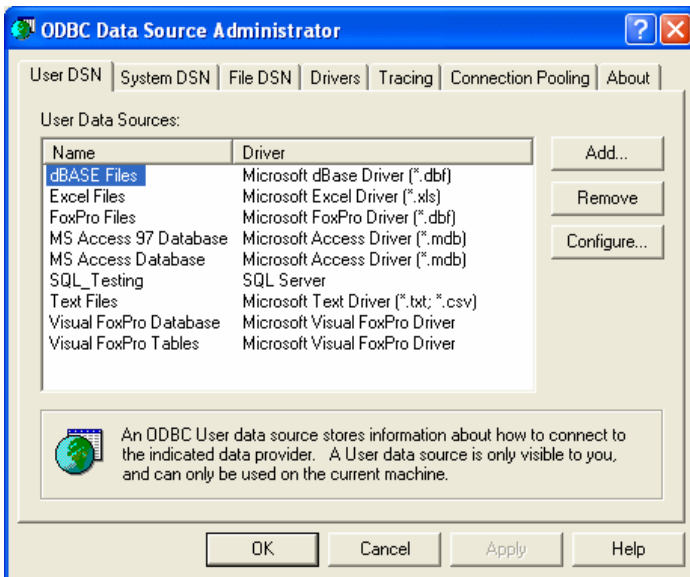
### Creating a Channel

We will start by creating a channel in the new project. The channel represents the connection to the data base.

1. Start by creating a new server project. To do this Click the New Project button in the main menu.
2. Click the "Add a New Channel" link to open the channel wizard. Assign a name for your channel then click next to select the ODBC Client Driver.
3. Next, select write optimization options. Accepting the defaults is fine.
4. Now select your Data Source. You must have a data source configured for the SQL database you are connecting to.



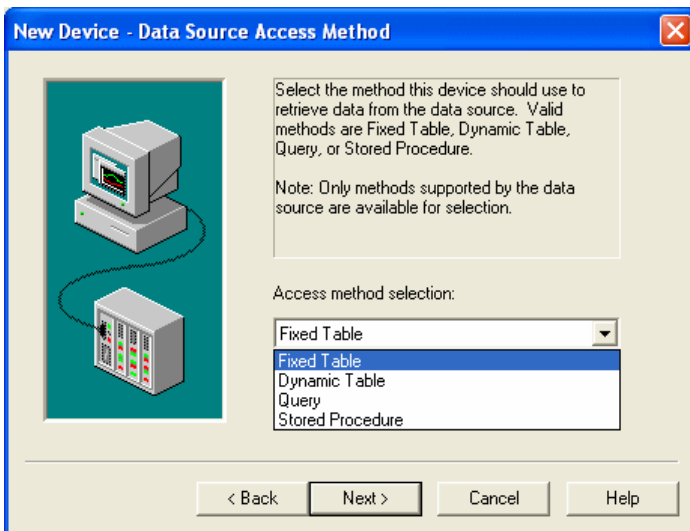
5. If you do not have a data source configured, click the "Configure DSN" button to open the Microsoft ODBC Data Source Administrator and do that now. See the Kepware ODBC driver Help file for information on creating a DSN for you database.



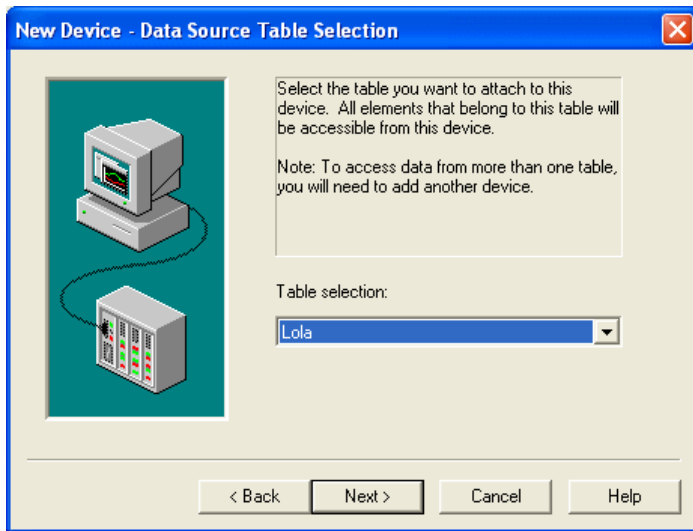
6. Finish the channel set up wizard, and click the “Add a New Device” link to open the device wizard.

### Creating a Device

7. The device created is a Fixed table, Dynamic table, Query, or Stored Procedure view of the SQL Database.

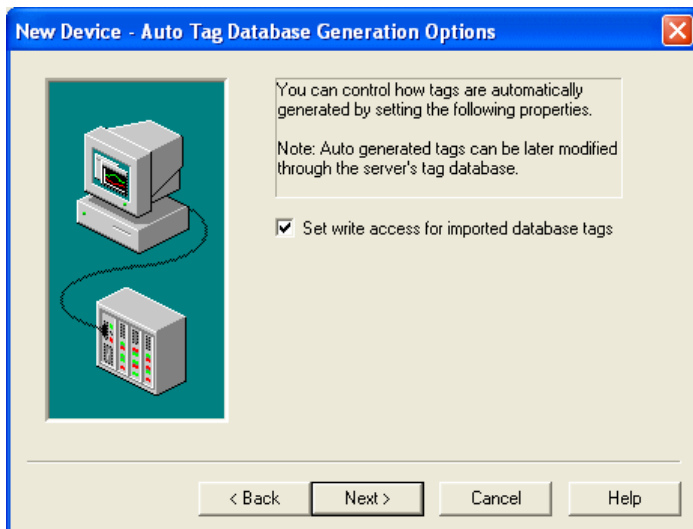


8. Next, select the Table or Stored Procedure that you want to use for your device. If you selected “Query” as an access method, enter the query you want to use here.

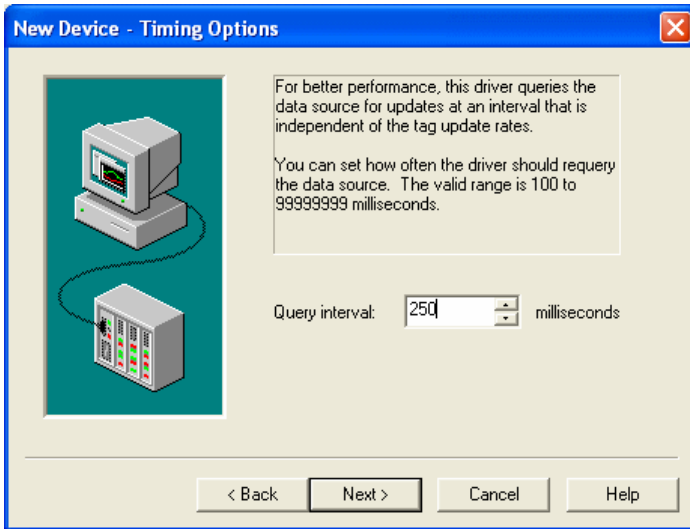


9. Next, decide if the connected OPC client should be allowed permission to write to the table by placing a check mark in the “Set write access for imported database tags” checkbox.

**Note: Enabling this option allows you to write, or modify the fields in the current record as long as the database allows it. This setting does not allow you to write, or append data to the database.**

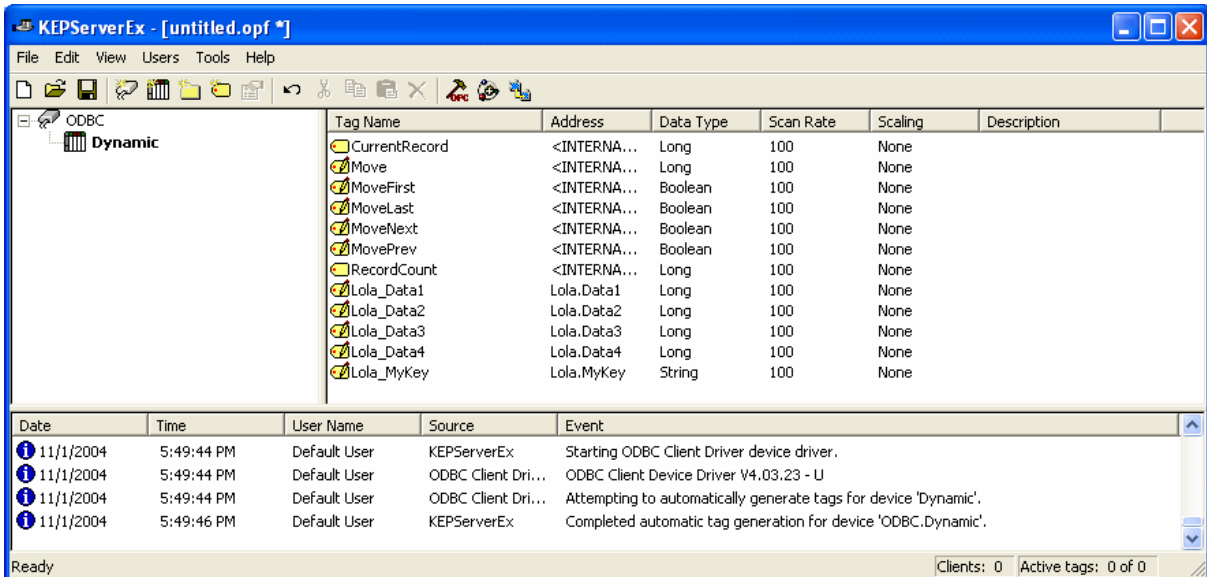


10. Lastly, set the rate at which you would like the Kepserver to refresh the dynaset of the table. Refreshing the dynaset too often may negatively affect the performance of the SQL data base.

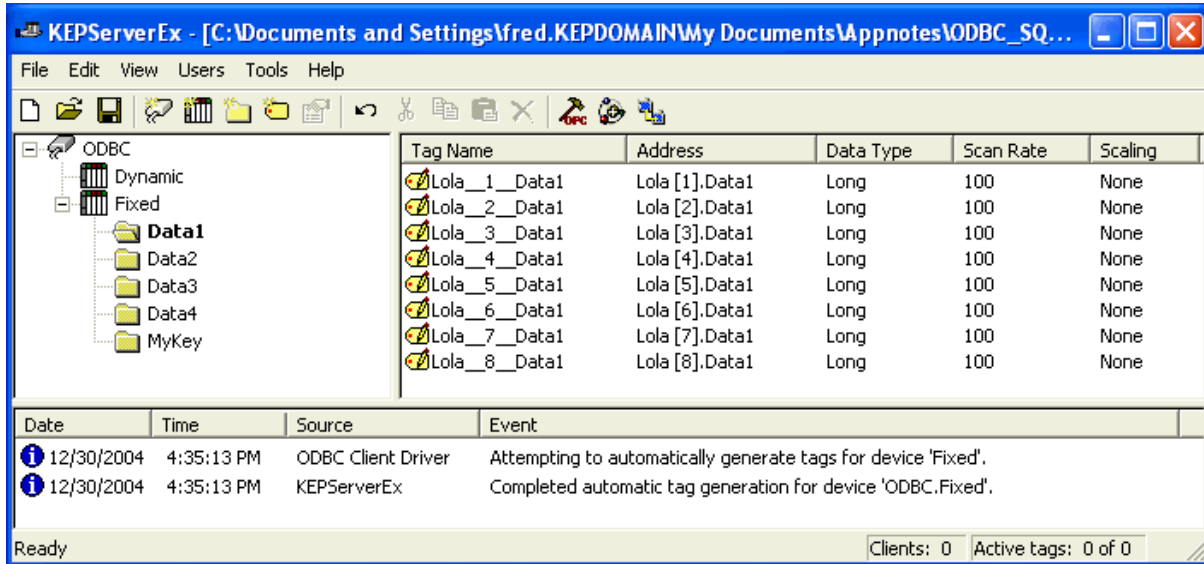


## Creating Tags

11. Once you have finished creating the device, the driver will automatically generate tags for the device. In this example we manually created tags to allow a user to navigate within the data base. We also created tags for the data fields located in the database table we wish to look at.



12. Notice the difference in the organizational method of tags in the example of a fixed table located below. A fixed table creates tag groups for each data field, and then generates a tag for every database record within the group.



**Note:** Notice the fixed table does not provide a means for navigating through the data in the table. In fact, true to its name, the number of tags created is fixed to the number of records in the table when it was created. Alternatively, when a dynamic table is refreshed the record count will increase if new records have been added to the database.

## Writing or Logging to SQL

One of the most common questions we get with the ODBC driver is, “How can I change, or log data, in the database?” There is no direct way to append data to a table without using a Stored Procedure. However, changing existing data in the table requires only the proper configuration. We will describe both methods briefly.

### Changing Data in a SQL Database Table

The ability to change existing data in a table has 4 requirements.

1. When you create a table in the SQL data base, set access permissions to allow user writes.
2. Make sure you assign a primary index. If you do not assign the primary index, your connection with the server project will default to a read only state. This occurs because we take a dynaset snapshot of the table when we connect to it at run time. Without an index there is no way to verify data write transactions are sent to the correct record. When there is a unique index we know exactly where new data is being sent.
3. When you create the device in the server place a check in the “Set write access for imported database tags” check box, to ensure the data tags generated are Write Enabled.

A good thing to remember when using the ODBC driver; writing data to tables may take longer than writing data to a PLC. That means if data is changing very rapidly, some changes may be lost if the database is overwhelmed.

## Logging Data to SQL Database Tables

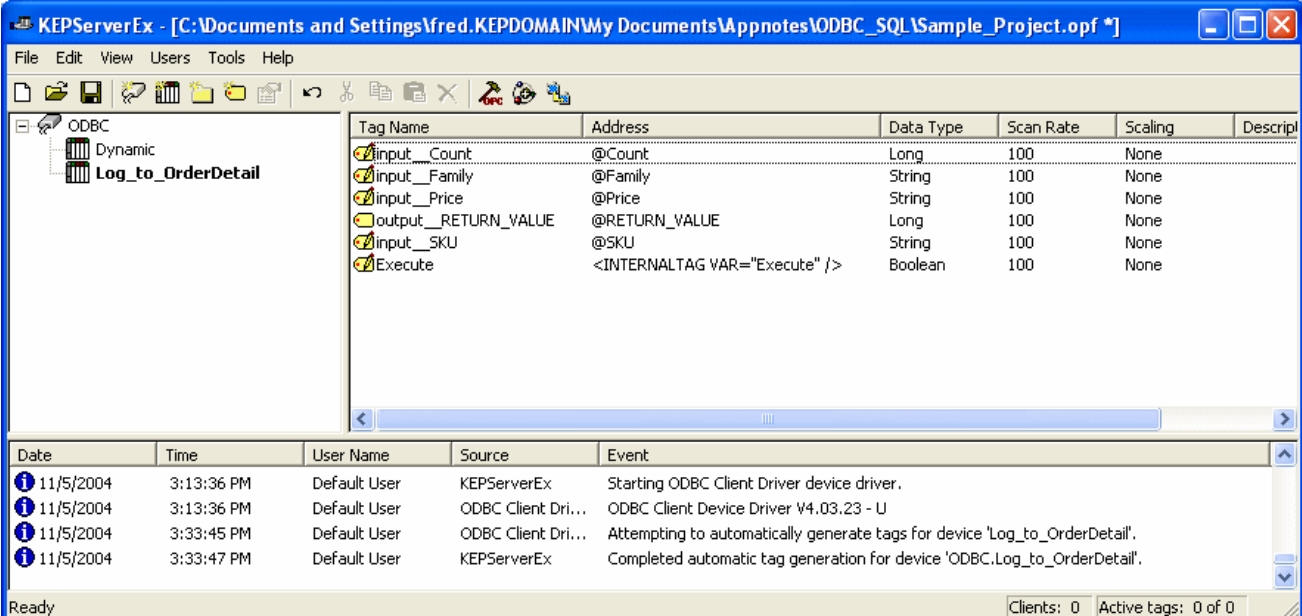
When logging data to a SQL table using the ODBC Client driver you will need to use a Stored Procedure. This is because data cannot be appended to a dynaset. The following content is from a Stored Procedure named "Log\_to\_OrderDetail". Stored Procedures can create dynamic queries by using user input. For instance, in the query below we are logging POS (Point of Sale) information to a SQL table.

```
CREATE PROCEDURE Log_to_OrderDetail @SKU Text, @Family Text, @Price Money,
@Count Int
AS Insert Into OrderDetail (Product_SKU, Family, Price, Quantity)
Values (@SKU, @Family, @Price, @Count)
GO
```

The way this works is as follows:

1. First we need to create a stored procedure like the one above. Notice we have assigned input parameter data types that match the data types of the fields they will be written to.
2. In our example the input parameters are SKU, Family, Price, and Count.
3. Specify which fields the data will be inserted into. This is important because we will not be starting with the first field in the table.
4. Determine which values will be entered into the fields, and specify the parameters.

When you create a table in the server you will notice several tags are generated. There should be one tag for each of the input parameters. There should be an output tag for the return value. This tag is not used for data logging. Lastly there should be an Execute tag. This tag is very important because it is the trigger that enables the Stored Procedure to log data.



The screenshot shows the KEPServerEx interface. The main window displays a list of ODBC tags for the 'Log\_to\_OrderDetail' device. The tags are:

Tag Name	Address	Data Type	Scan Rate	Scaling	Descript
input_Count	@Count	Long	100	None	
input_Family	@Family	String	100	None	
input_Price	@Price	String	100	None	
output_RETURN_VALUE	@RETURN_VALUE	Long	100	None	
input_SKU	@SKU	String	100	None	
Execute	<INTERNALTAG VAR="Execute" />	Boolean	100	None	

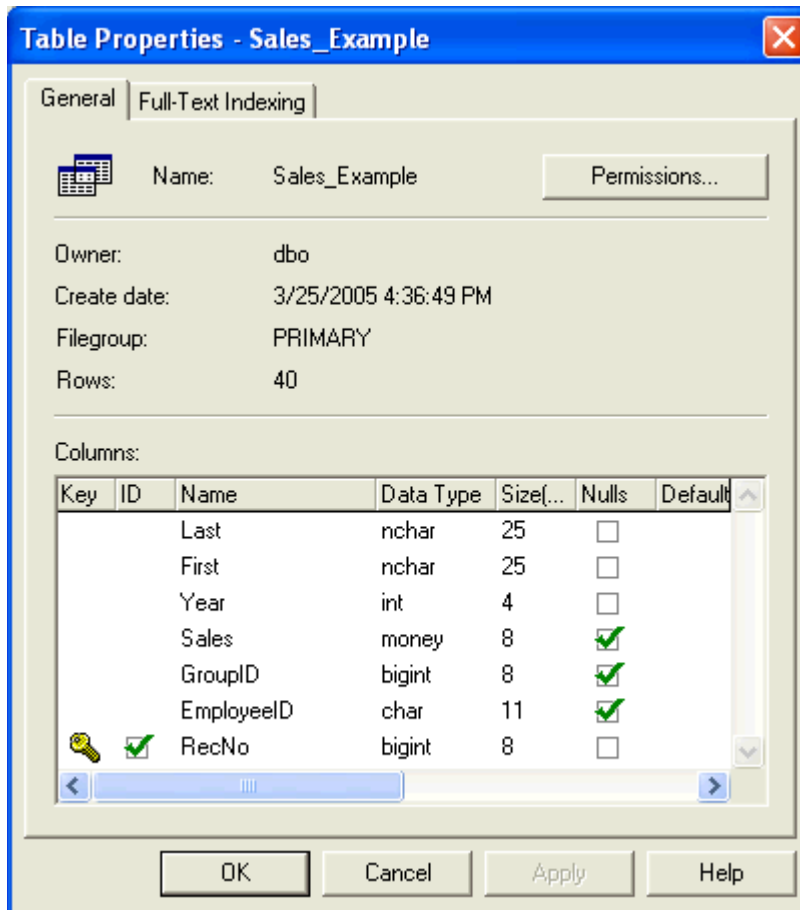
Below the tag list is a log window showing the following events:

Date	Time	User Name	Source	Event
11/5/2004	3:13:36 PM	Default User	KEPServerEx	Starting ODBC Client Driver device driver.
11/5/2004	3:13:36 PM	Default User	ODBC Client Dri...	ODBC Client Device Driver V4.03.23 - U
11/5/2004	3:33:45 PM	Default User	ODBC Client Dri...	Attempting to automatically generate tags for device 'Log_to_OrderDetail'.
11/5/2004	3:33:47 PM	Default User	KEPServerEx	Completed automatic tag generation for device 'ODBC.Log_to_OrderDetail'.

To log data write to the input tags using your client. Then write a 1 to the Execute tag to log the data. The procedure will only run when you execute it.

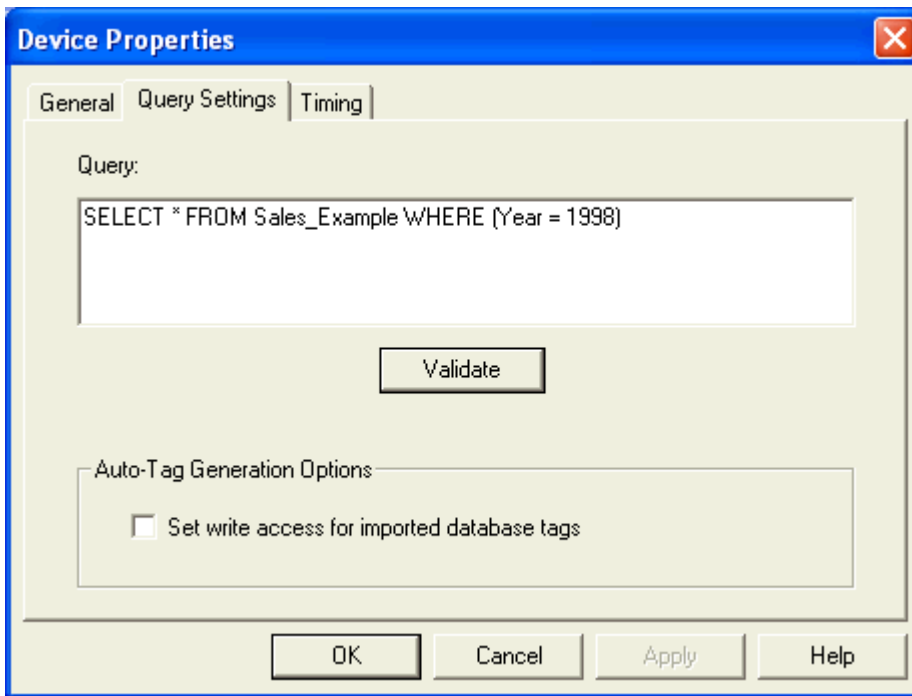
## Querying SQL Tables

There are 2 ways to query data when connecting to SQL Databases. The first is to use a stored procedure as we did when logging data. The second is to pass a query statement to the SQL database and receive the response. The query is evaluated at the rate of the query interval, which is set in the device's property page. This example will use a simple table with sales data in it. The following graphic shows the structure of the table.

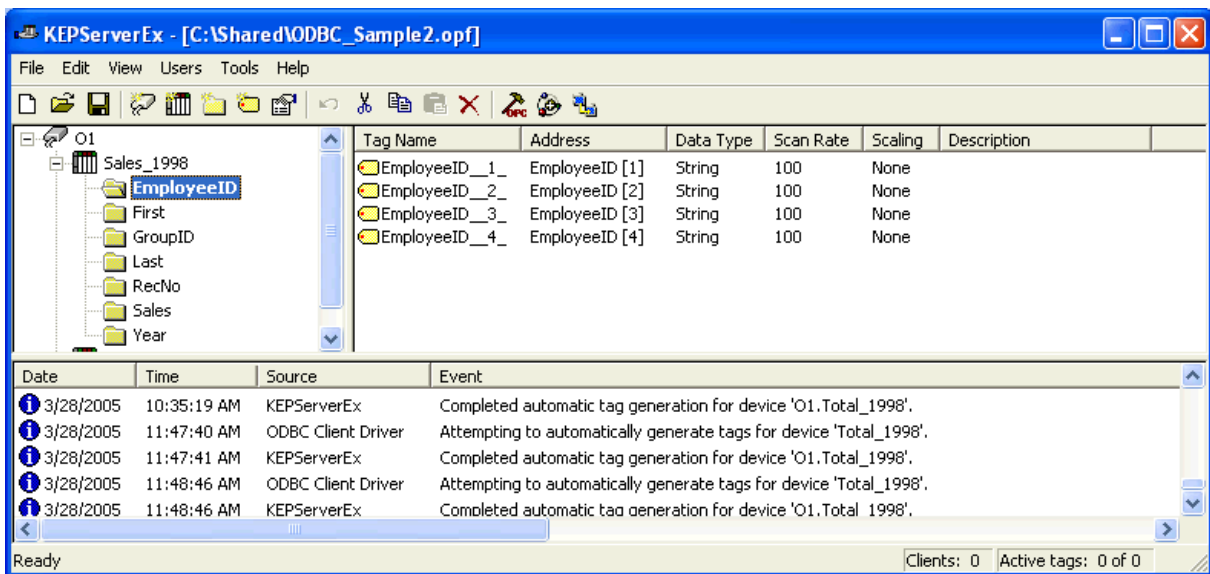


## Query Statement

The following is an example of how to create a simple query. In a KepserverEx ODBC project create a device. The Access Method will be Query. In this query we will ask for all Employee Sales records for the year 1998. The following example is how the query is entered, "SELECT \* FROM Sales\_Example WHERE (Year = 1998)".

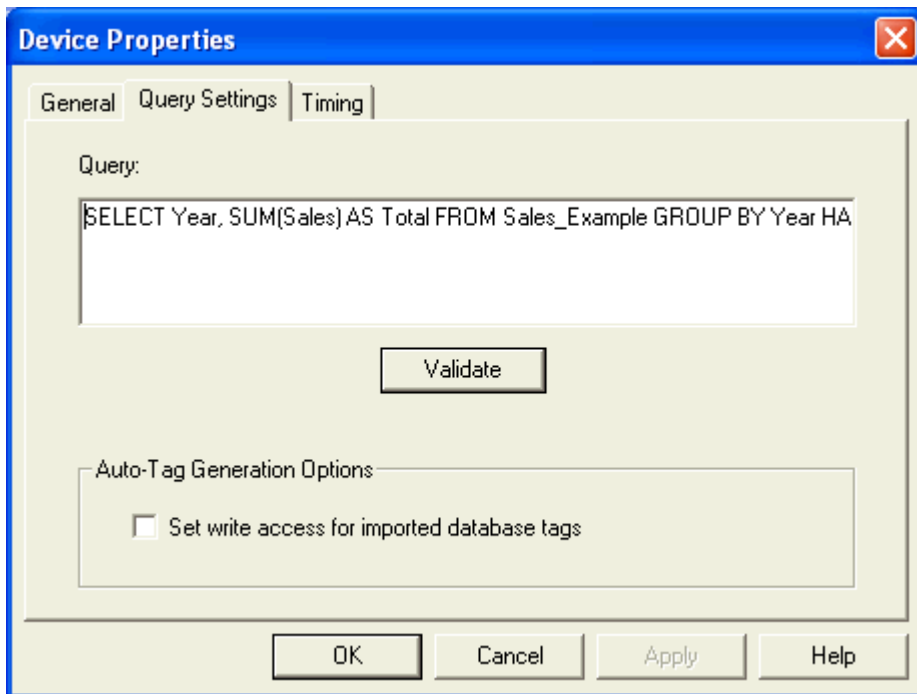


When you complete the wizard the server will automatically generate tags for items that are returned.

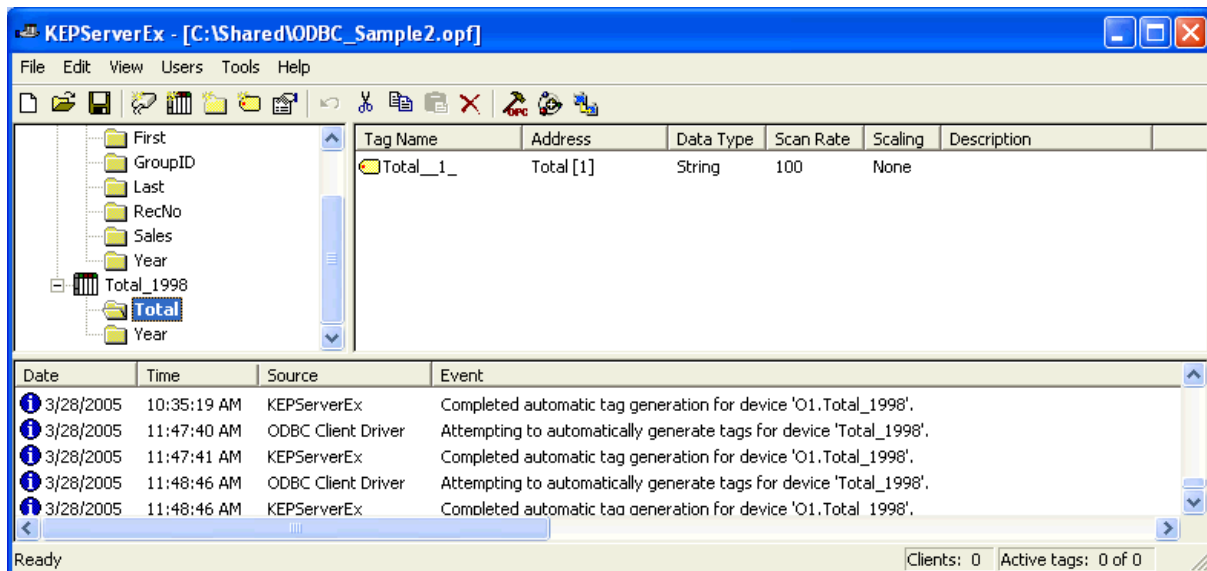


In our database there are 4 different Employees with Sales Total records for 1998.

Next, let's look at a more complex query. In this example we want to see a total sales amount for the year 1998. To obtain this information we must create another device with a Query Access mode, and enter the following query statement, "SELECT Year, SUM(Sales) AS Total FROM Sales\_Example GROUP BY Year HAVING (Year = 1998)".



This is referred to as a compound query. The server should contain two tag groups named Year, and Total, with one tag in each of them.



If you want to retrieve the yearly total for all sales data in the database table, simply modify the previous query as follows, “SELECT Year, SUM(Sales) AS Total FROM Sales\_Example GROUP BY Year”. With this query statement the server should contain two data groups with a tag for each year that contained data.