# OPC Data Logger Case Study

Efficient Database Data Logging

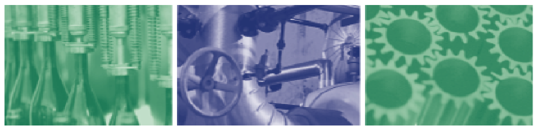
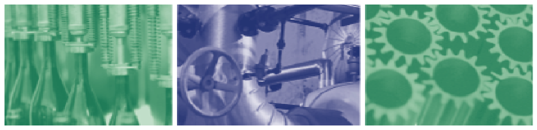


Table of Contents



# Introduction

## Intended Audience

This document is intended for people who need to log OPC data to a database. Plain and simple.

This document assumes no prior database experience.

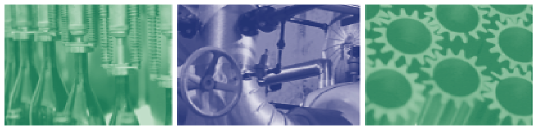
## Purpose of this document

This document is intended to provoke design decisions prior to implementation.

This document is intended to help facilitate the easiest possible OPC Data Logger configuration, while leveraging maximum database efficiency which will result in:

- Less data being logged
- More accurate data being logged
- Maximum reporting capabilities.

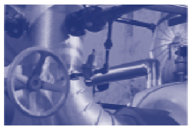
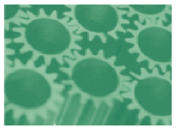
This document will outline a simple data logging scenario along with step-by-step instructions that will accomplish a solution.



## Data Logging Scenario

Let's begin by listing our requirements:

1. We have several items/tags that we want to log.
2. We want to analyze the logged data later (not in real time)
3. We want to see the data logged in a table format, showing the data between time-frames.
4. We want to see if any data was questionable/bad during a specific time-frame.

## Data Design

### Common Problem – The Quickest/Easiest Approach

Quite often, what seems like the quickest and easiest approach is the one that most people go with… let's see what this typically looks like for most people.

We will log our data to a single table. Pay close attention to the data within the table:

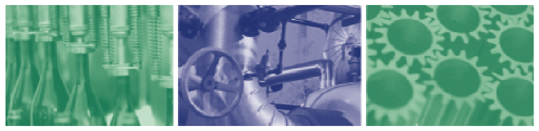
| Item_Name | Item_Value | Item_Quality | Item_Timestamp |
|---|---|---|---|
| Channel1.device.1.Tag1 | 10 | 192 | 1/1/2008 12:01:01 |
| Channel1.device.1.Tag1 | 20 | 192 | 1/1/2008 12:01:01 |
| Channel1.device.1.Tag2 | 30 | 192 | 1/1/2008 12:01:01 |
| Channel1.device.1.Tag1 | 40 | 192 | 1/1/2008 12:01:01 |
| Channel1.device.1.Tag1 | 50 | 192 | 1/1/2008 12:01:01 |
| Channel1.device.1.Tag3 | 60 | 192 | 1/1/2008 12:01:01 |

The name of each item that has been logged is approx. 22 characters wide. If we look at the above table we can see that *Channel1.Device1.Tag1* has been logged 4 times. That means that we have logged a total of:

22 x 4 characters = 88 characters

Obviously, as more data is logged, this consumption of space will increase. Obviously this is not the best use of space.

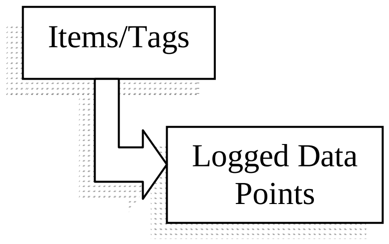
**Analysis capabilities will be hindered** if we want to search for all of the data points for a particular item, as **the database will have to query the entire database table** marking a huge performance penalty.



## Separating Data into Multiple Tables

Now that we have seen the most commonly seen problem, and that we also have our requirements, we can begin our data design.

The first step is to identify and normalize the data. Database normalization will out the scope of this document, but we will show you an example below:

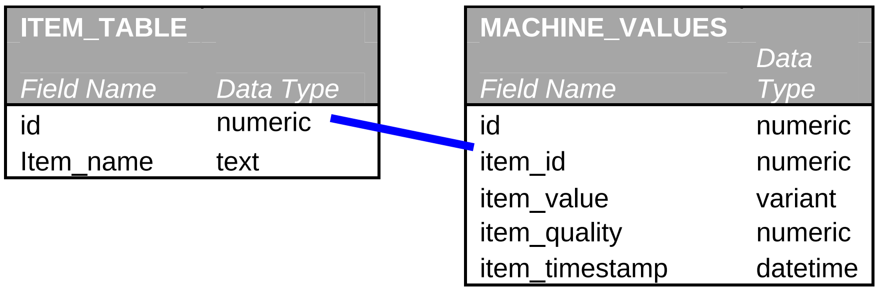


In this case, we have separated the data into 2 respective areas:

- The Items/Tags – and we will relate them to the Machines
- The logged data points – and we will relate them to the Items

## Data Table Design

In our simple design, we have opted to separate our logged values from the underlying items themselves. Here is a look at these simple tables:



| ITEM_TABLE | |
|---|---|
| *Field Name* | *Data Type* |
| id | numeric |
| Item_name | text |

| MACHINE_VALUES | |
|---|---|
| *Field Name* | *Data Type* |
| id | numeric |
| item_id | numeric |
| item_value | variant |
| item_quality | numeric |
| item_timestamp | datetime |

In the above case, we have defined a relationship between the values being logged against the known item in the ITEM_TABLE.

## *How will data be stored?*

In our first example we stored all of our data into a single table. Now we are opting to store the data in multiple tables. This does also mean that our ITEM_TABLE must be populated before we begin logging. For example:

| **ITEM_TABLE** | |
|---|---|
| *Id* | *Item_Name* |
| 1 | Channel1.Device1.Tag1 |
| 2 | Channel1.Device1.Tag2 |
| 3 | Channel1.Device1.Tag3 |

Now that our Items database table contains our data, we can log our values into the MACHINE_VALUES as follows:

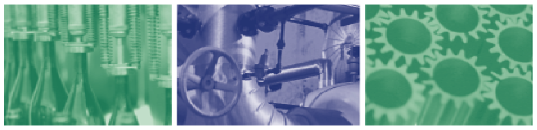
| **MACHINE_VALUES** | | | | |
|---|---|---|---|---|
| *Id* | *Item_Id* | *Item_Value* | *Item_Quality* | *Item_Timestamp* |
| 1 | 1 | 10 | 192 | 1/1/2008 12:01:01 |
| 2 | 1 | 20 | 192 | 1/1/2008 12:01:01 |
| 3 | 2 | 30 | 192 | 1/1/2008 12:01:01 |
| 4 | 1 | 40 | 192 | 1/1/2008 12:01:01 |
| 5 | 1 | 50 | 192 | 1/1/2008 12:01:01 |
| 6 | 3 | 60 | 192 | 1/1/2008 12:01:01 |

Note: this time the name of the item is NOT being logged, instead, its index within the other/related table is being logged instead.

## *What kind of reports can we obtain using this design?*

Here are some examples:

1. a list of values for any item between a specific time-frame
2. a list of values for a specific item or set of items
3. a list of items, grouped by their quality code, by item or group of items
4. total number of data points logged, by item or group of items
5. average of data points logged, by item or group of items

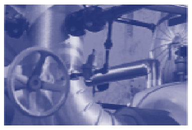
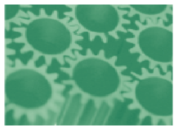


***Are there any other benefits to this design?***

Yes, several.

1. If you need to add more items to log, then add them into the ITEM_TABLE first, and then configure the OPC Data Logger accordingly. No other database schema changes are needed.

2. If you no longer need to log any particular item(s), then simply stop logging them from the OPC Data Logger. The data within the database can remain.

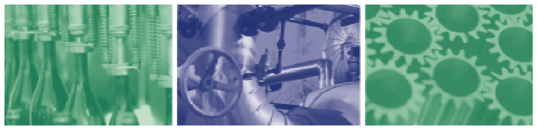
# Configuring the OPC Data Logger

## Configuration Process

Before configuring the OPC Data Logger, your database should be already configured with the previously documented tables, and the ITEM_TABLE already pre-populated with the names of the items you will be logging.

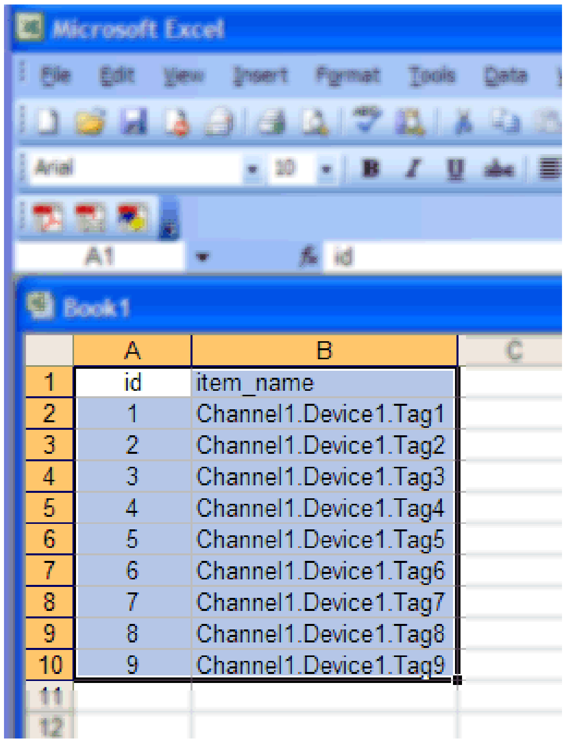
Assuming the items already exist within the database, simply export the contents of the ITEM_TABLE into a *.CSV file and open the file within Microsoft Excel (or any application that will allow you to modify the data).

We will now configure the data such that we will be able to import this *.CSV file straight into the OPC Data Logger, which will prevent us from having to manually configure our items.



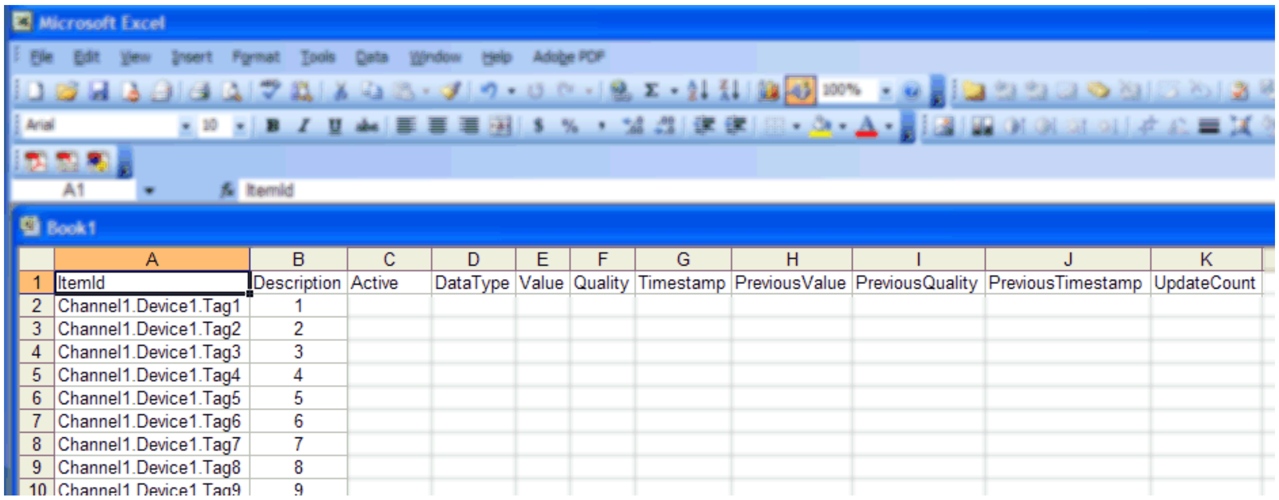
## Modifying the items data within Excel

The next step assumes that the items have been exported from the database and are now loaded within Excel.

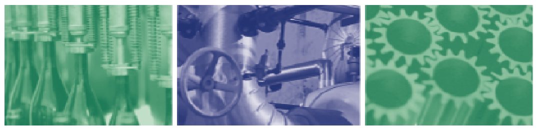


| | A | B |
|---|---|---|
| 1 | id | item_name |
| 2 | 1 | Channel1.Device1.Tag1 |
| 3 | 2 | Channel1.Device1.Tag2 |
| 4 | 3 | Channel1.Device1.Tag3 |
| 5 | 4 | Channel1.Device1.Tag4 |
| 6 | 5 | Channel1.Device1.Tag5 |
| 7 | 6 | Channel1.Device1.Tag6 |
| 8 | 7 | Channel1.Device1.Tag7 |
| 9 | 8 | Channel1.Device1.Tag8 |
| 10 | 9 | Channel1.Device1.Tag9 |

In order to import this data into the OPC Data Logger, we will need to modify the data by first adding the extra columns that are needed:


Microsoft Excel
File Edit View Insert Format Tools Data Window Help Adobe PDF
Arial 10
A1 ItemId
Book1

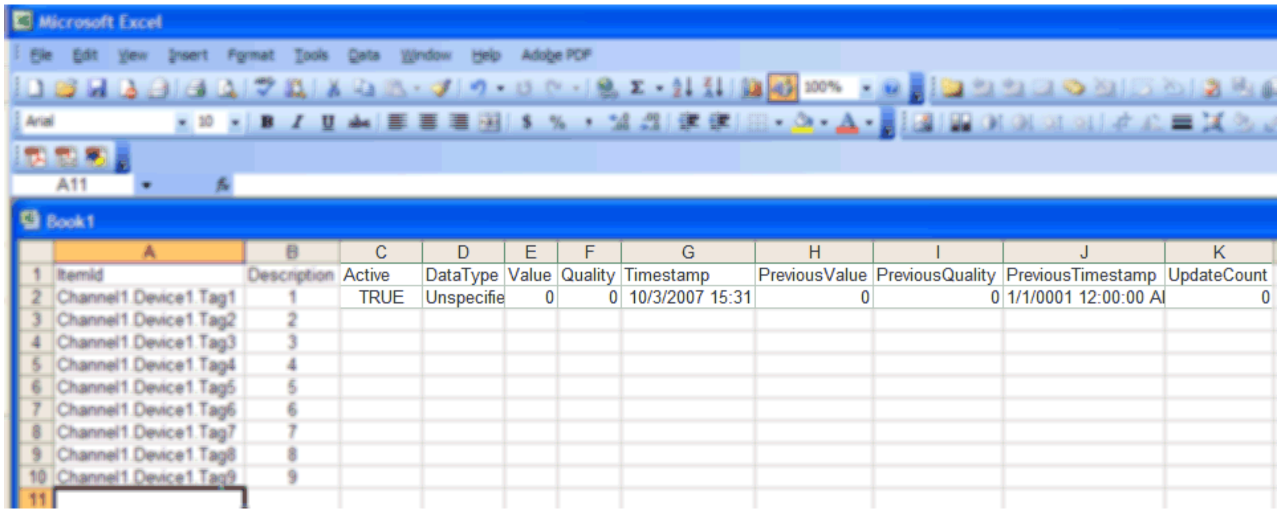
| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ItemId | Description | Active | DataType | Value | Quality | Timestamp | PreviousValue | PreviousQuality | PreviousTimestamp | UpdateCount |
| 2 | Channel1.Device1.Tag1 | 1 | | | | | | | | | |
| 3 | Channel1.Device1.Tag2 | 2 | | | | | | | | | |
| 4 | Channel1.Device1.Tag3 | 3 | | | | | | | | | |
| 5 | Channel1.Device1.Tag4 | 4 | | | | | | | | | |
| 6 | Channel1.Device1.Tag5 | 5 | | | | | | | | | |
| 7 | Channel1.Device1.Tag6 | 6 | | | | | | | | | |
| 8 | Channel1.Device1.Tag7 | 7 | | | | | | | | | |
| 9 | Channel1.Device1.Tag8 | 8 | | | | | | | | | |
| 10 | Channel1.Device1.Tag9 | 9 | | | | | | | | | |



Important:

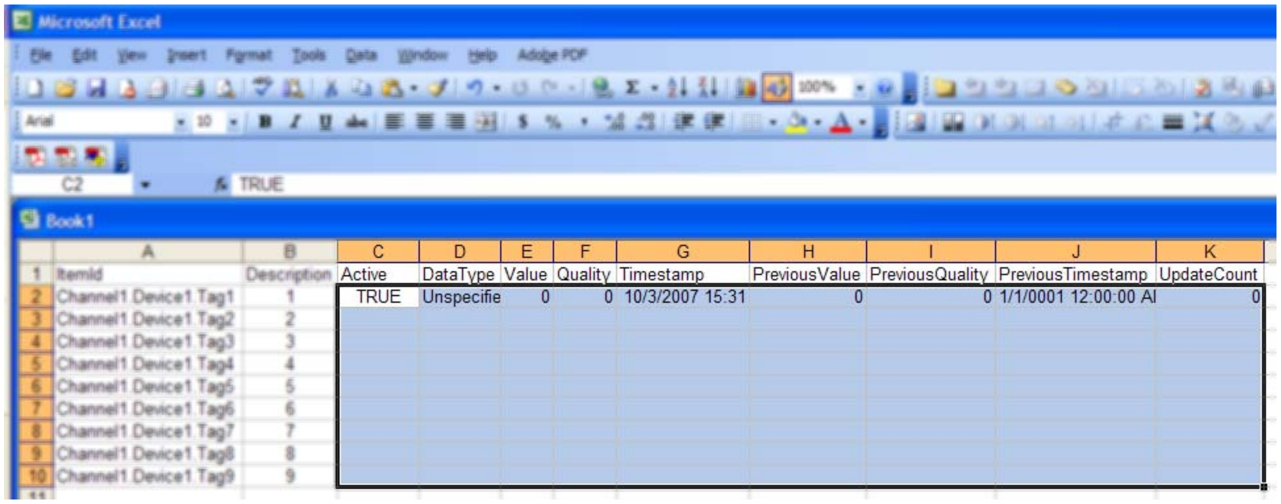
The DESCRIPTION column actually contains the index (id) for the record within the database.

Now we will fill-in the empty cell values for the first row:

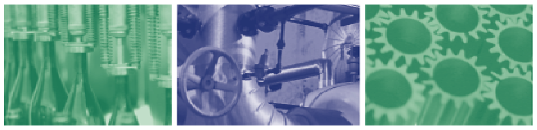

Microsoft Excel
File Edit View Insert Format Tools Data Window Help Adobe PDF
Arial 10
100%
A11
Book1

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ItemId | Description | Active | DataType | Value | Quality | Timestamp | PreviousValue | PreviousQuality | PreviousTimestamp | UpdateCount |
| 2 | Channel1.Device1.Tag1 | 1 | TRUE | Unspecifie | 0 | 0 | 10/3/2007 15:31 | 0 | 0 | 1/1/0001 12:00:00 AI | 0 |
| 3 | Channel1.Device1.Tag2 | 2 | | | | | | | | | |
| 4 | Channel1.Device1.Tag3 | 3 | | | | | | | | | |
| 5 | Channel1.Device1.Tag4 | 4 | | | | | | | | | |
| 6 | Channel1.Device1.Tag5 | 5 | | | | | | | | | |
| 7 | Channel1.Device1.Tag6 | 6 | | | | | | | | | |
| 8 | Channel1.Device1.Tag7 | 7 | | | | | | | | | |
| 9 | Channel1.Device1.Tag8 | 8 | | | | | | | | | |
| 10 | Channel1.Device1.Tag9 | 9 | | | | | | | | | |
| 11 | | | | | | | | | | | |

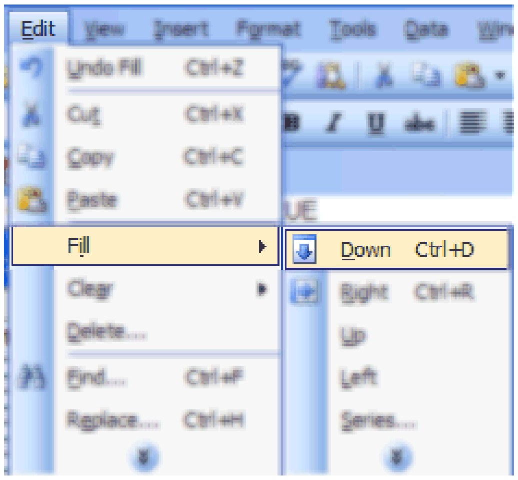
Now we will copy all of these values we have just added to the cells in the remaining empty rows. We can do this quickly and easily by using the **Fill-down** option within Excel. The first key is to select the entire range of data that we added, in this case cell **C2** to **K2**. But, we also need to expand this range so that it covers the blank cells beneath this range as shown below:


Microsoft Excel
File Edit View Insert Format Tools Data Window Help Adobe PDF
Arial 10
100%
C2
TRUE
Book1

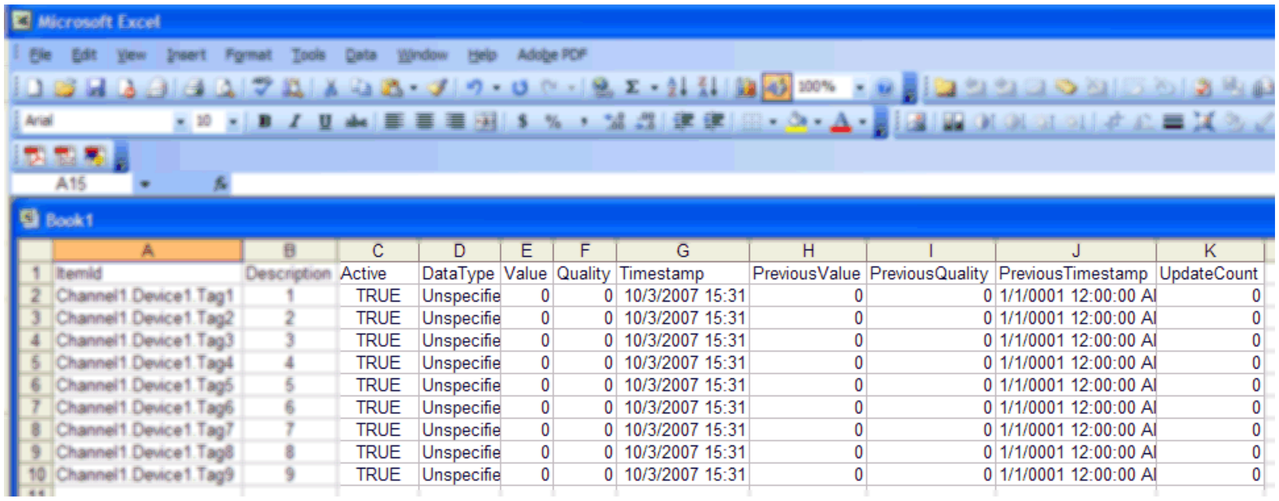
| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ItemId | Description | Active | DataType | Value | Quality | Timestamp | PreviousValue | PreviousQuality | PreviousTimestamp | UpdateCount |
| 2 | Channel1.Device1.Tag1 | 1 | TRUE | Unspecifie | 0 | 0 | 10/3/2007 15:31 | 0 | 0 | 1/1/0001 12:00:00 AI | 0 |
| 3 | Channel1.Device1.Tag2 | 2 | | | | | | | | | |
| 4 | Channel1.Device1.Tag3 | 3 | | | | | | | | | |
| 5 | Channel1.Device1.Tag4 | 4 | | | | | | | | | |
| 6 | Channel1.Device1.Tag5 | 5 | | | | | | | | | |
| 7 | Channel1.Device1.Tag6 | 6 | | | | | | | | | |
| 8 | Channel1.Device1.Tag7 | 7 | | | | | | | | | |
| 9 | Channel1.Device1.Tag8 | 8 | | | | | | | | | |
| 10 | Channel1.Device1.Tag9 | 9 | | | | | | | | | |



Now choose Fill-down:

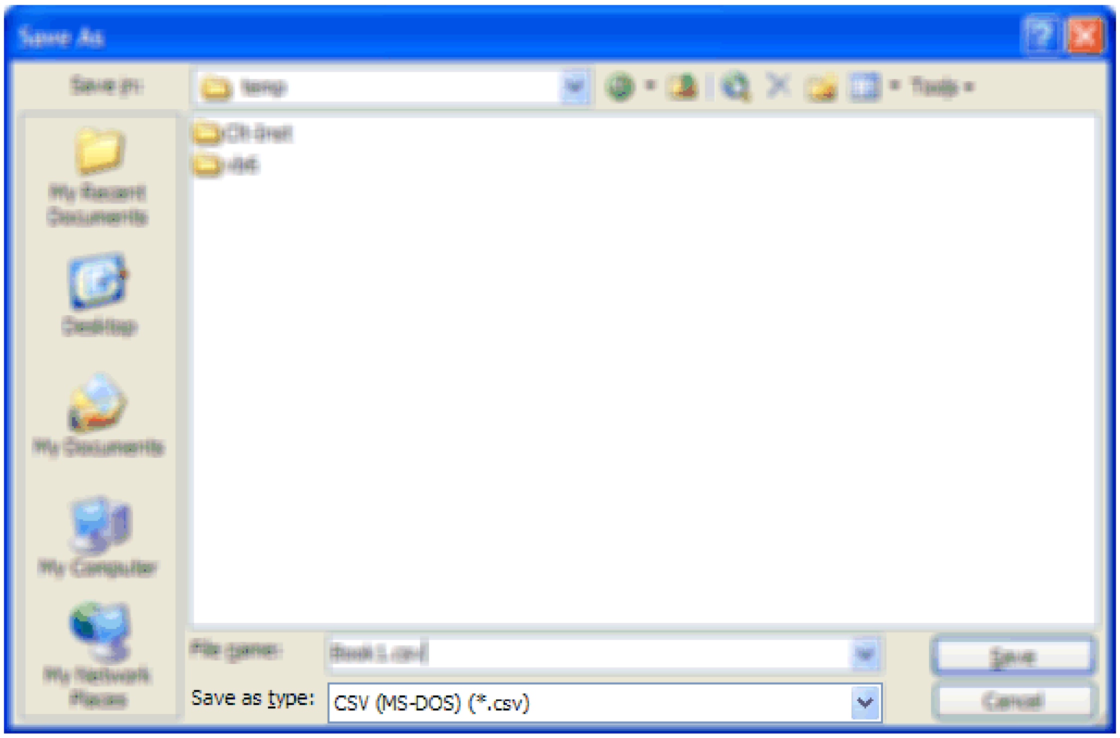


The values will now be copied onto each row as shown here:


Microsoft Excel
File Edit View Insert Format Tools Data Window Help Adobe PDF
Arial
10
A15
Book1

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ItemId | Description | Active | DataType | Value | Quality | Timestamp | PreviousValue | PreviousQuality | PreviousTimestamp | UpdateCount |
| 2 | Channel1.Device1.Tag1 | 1 | TRUE | Unspecifie | 0 | 0 | 10/3/2007 15:31 | 0 | 0 | 1/1/0001 12:00:00 AI | 0 |
| 3 | Channel1.Device1.Tag2 | 2 | TRUE | Unspecifie | 0 | 0 | 10/3/2007 15:31 | 0 | 0 | 1/1/0001 12:00:00 AI | 0 |
| 4 | Channel1.Device1.Tag3 | 3 | TRUE | Unspecifie | 0 | 0 | 10/3/2007 15:31 | 0 | 0 | 1/1/0001 12:00:00 AI | 0 |
| 5 | Channel1.Device1.Tag4 | 4 | TRUE | Unspecifie | 0 | 0 | 10/3/2007 15:31 | 0 | 0 | 1/1/0001 12:00:00 AI | 0 |
| 6 | Channel1.Device1.Tag5 | 5 | TRUE | Unspecifie | 0 | 0 | 10/3/2007 15:31 | 0 | 0 | 1/1/0001 12:00:00 AI | 0 |
| 7 | Channel1.Device1.Tag6 | 6 | TRUE | Unspecifie | 0 | 0 | 10/3/2007 15:31 | 0 | 0 | 1/1/0001 12:00:00 AI | 0 |
| 8 | Channel1.Device1.Tag7 | 7 | TRUE | Unspecifie | 0 | 0 | 10/3/2007 15:31 | 0 | 0 | 1/1/0001 12:00:00 AI | 0 |
| 9 | Channel1.Device1.Tag8 | 8 | TRUE | Unspecifie | 0 | 0 | 10/3/2007 15:31 | 0 | 0 | 1/1/0001 12:00:00 AI | 0 |
| 10 | Channel1.Device1.Tag9 | 9 | TRUE | Unspecifie | 0 | 0 | 10/3/2007 15:31 | 0 | 0 | 1/1/0001 12:00:00 AI | 0 |

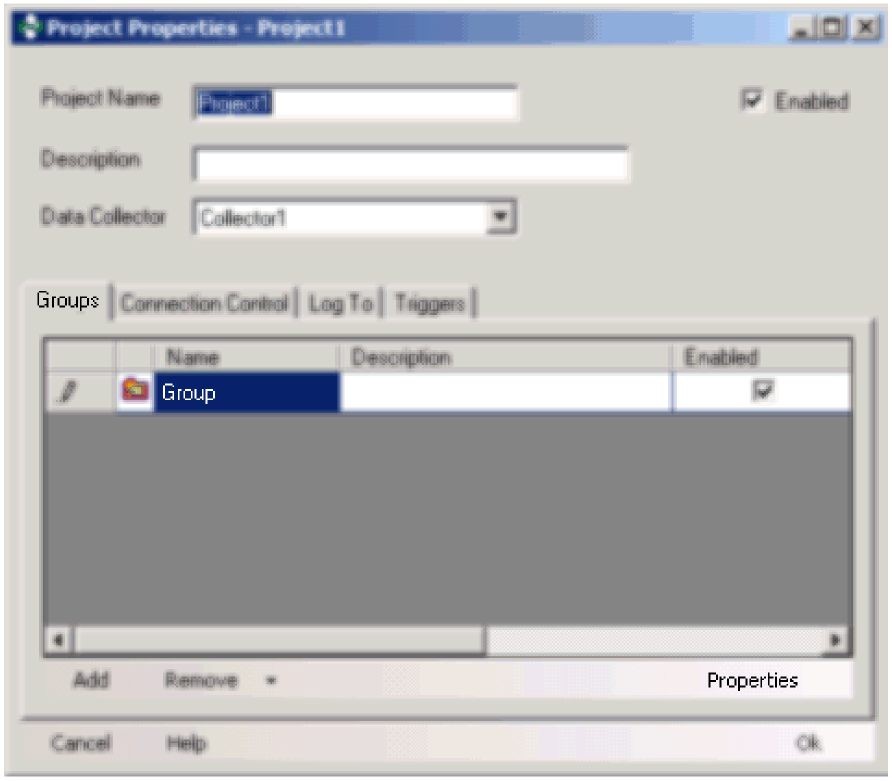
Now save this sheet as a *.CSV file by simply choosing FILE -> SAVE AS and then picking *.CSV as shown here:



Save as type: CSV (MS-DOS) (*.csv)

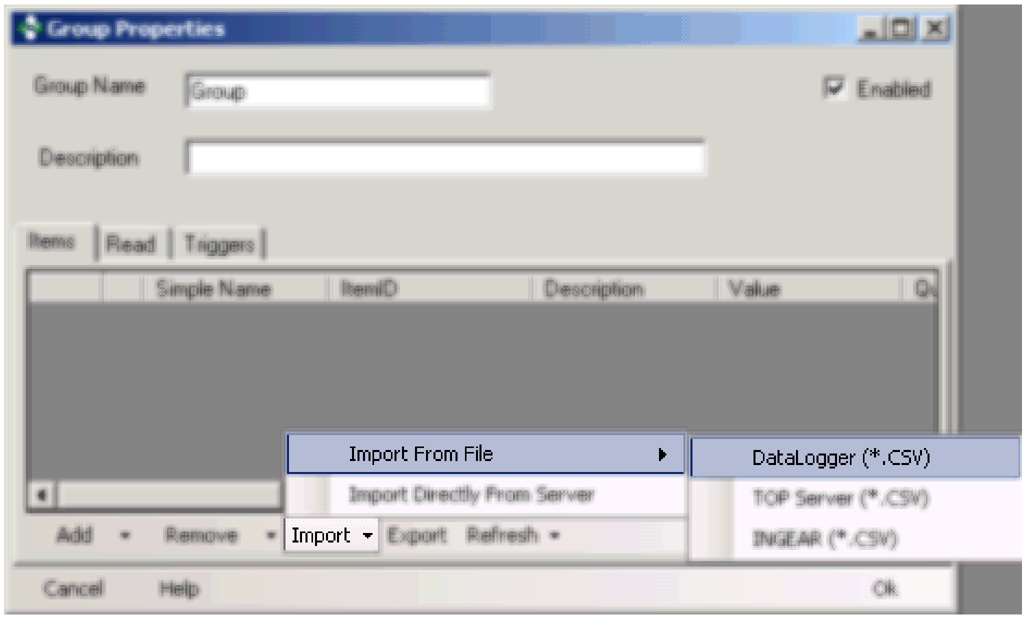
## Importing the items into the OPC Data Logger

1. Open the OPC Data Logger, and then open your Project containing the group we will be modifying.



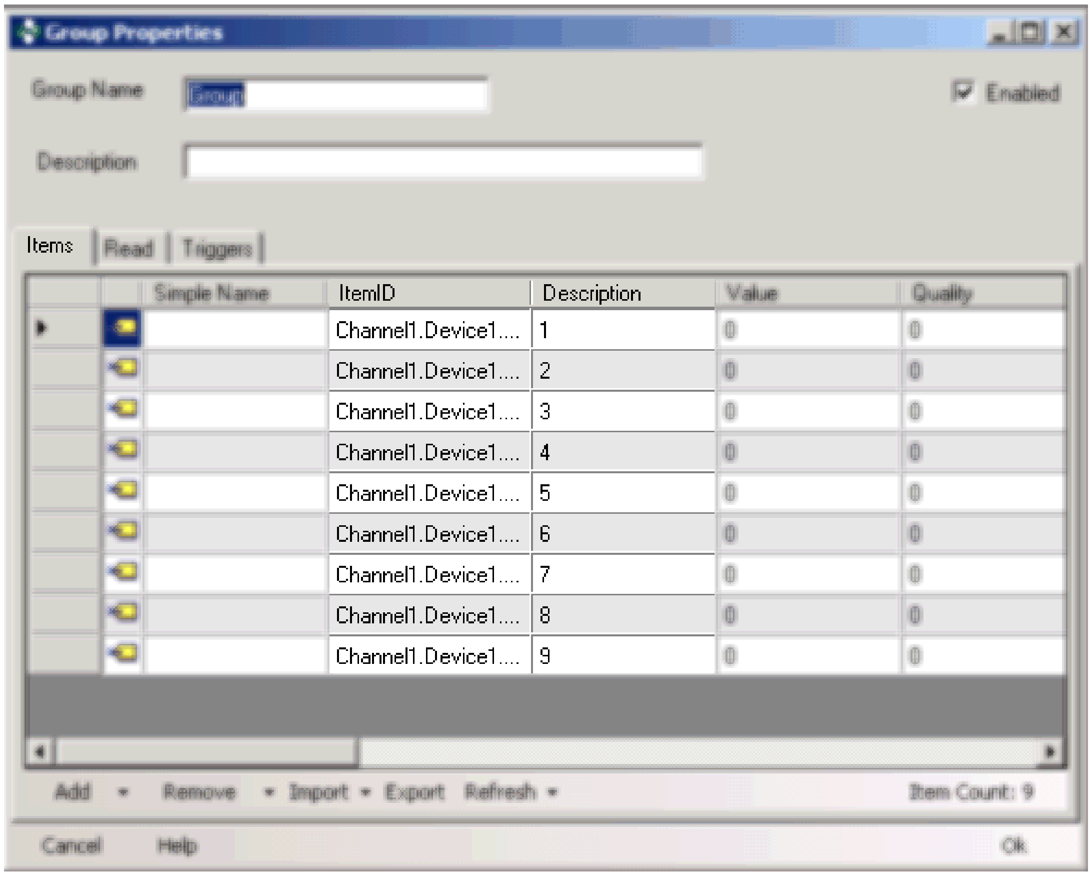
2. Highlight (or add) the group, and then open its **Properties**.

3. Click on the **Items** tab and then open the **Import** option at the bottom of the window, choosing the **Import From File** sub-menu finally clicking on the **DataLogger (*.CSV)** option:



4. A dialog will prompt you to choose the file to import. Locate your file and then click OK.

5. The items should now be imported:



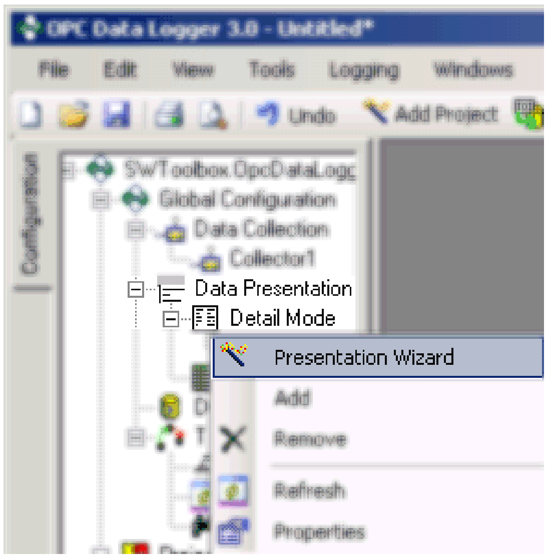
| Simple Name | ItemID | Description | Value | Quality |
|---|---|---|---|---|
| | Channel1.Device1.... | 1 | 0 | 0 |
| | Channel1.Device1.... | 2 | 0 | 0 |
| | Channel1.Device1.... | 3 | 0 | 0 |
| | Channel1.Device1.... | 4 | 0 | 0 |
| | Channel1.Device1.... | 5 | 0 | 0 |
| | Channel1.Device1.... | 6 | 0 | 0 |
| | Channel1.Device1.... | 7 | 0 | 0 |
| | Channel1.Device1.... | 8 | 0 | 0 |
| | Channel1.Device1.... | 9 | 0 | 0 |

Note that Item's id in the database is stored within the **Description** column.

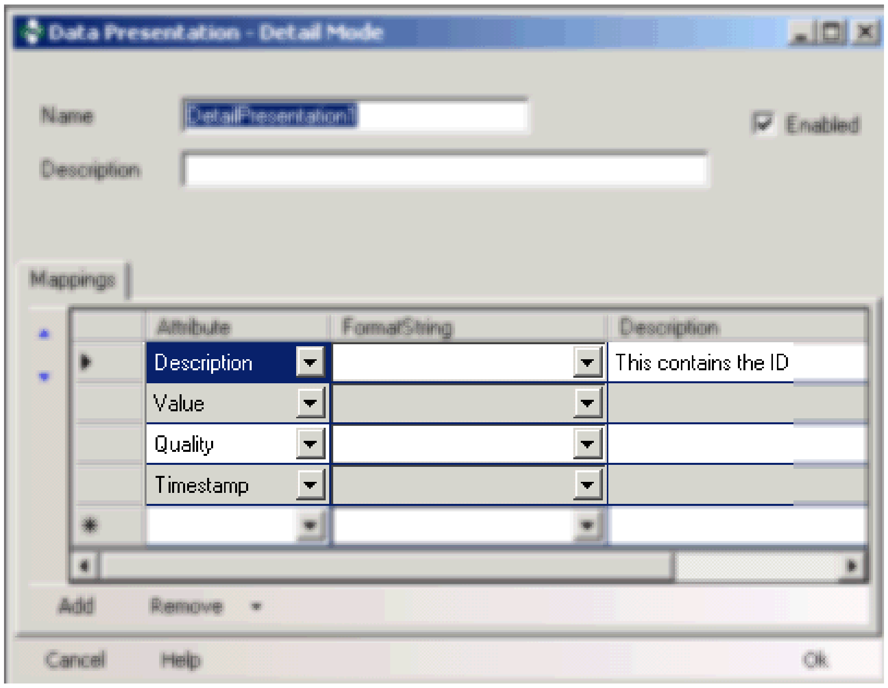
## Configuring a Detail mode presentation formatter

We need to configure just one detail mode formatter that will correctly log our items (including the reference to the Item ID).

1. Create a new Detail mode presentation by simply right-clicking on the **Detail Mode** icon in the main application tree-view:



2. Now configure the detail mode to resemble the following:



Name: DetailPresentation1

Description:

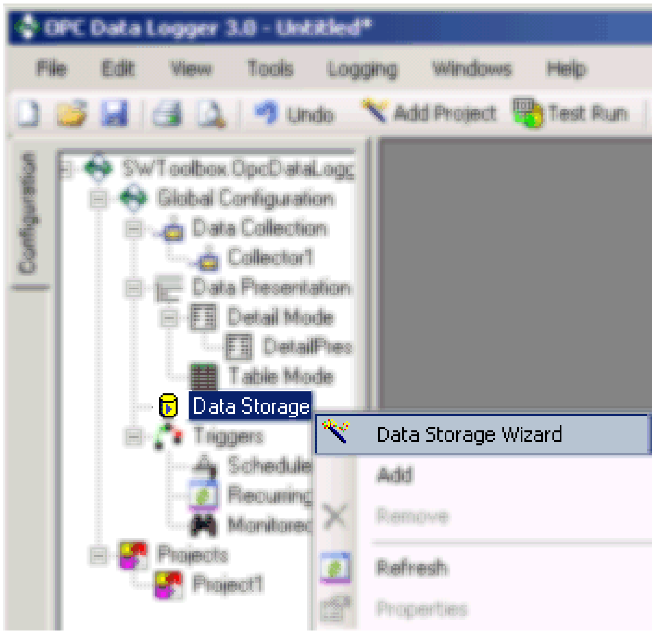
| | Attribute | FormatString | Description |
|---|---|---|---|
| ▶ | Description | | This contains the ID |
| | Value | | |
| | Quality | | |
| | Timestamp | | |
| * | | | |

3. Click OK to save and close this window.

## Connecting to your Database

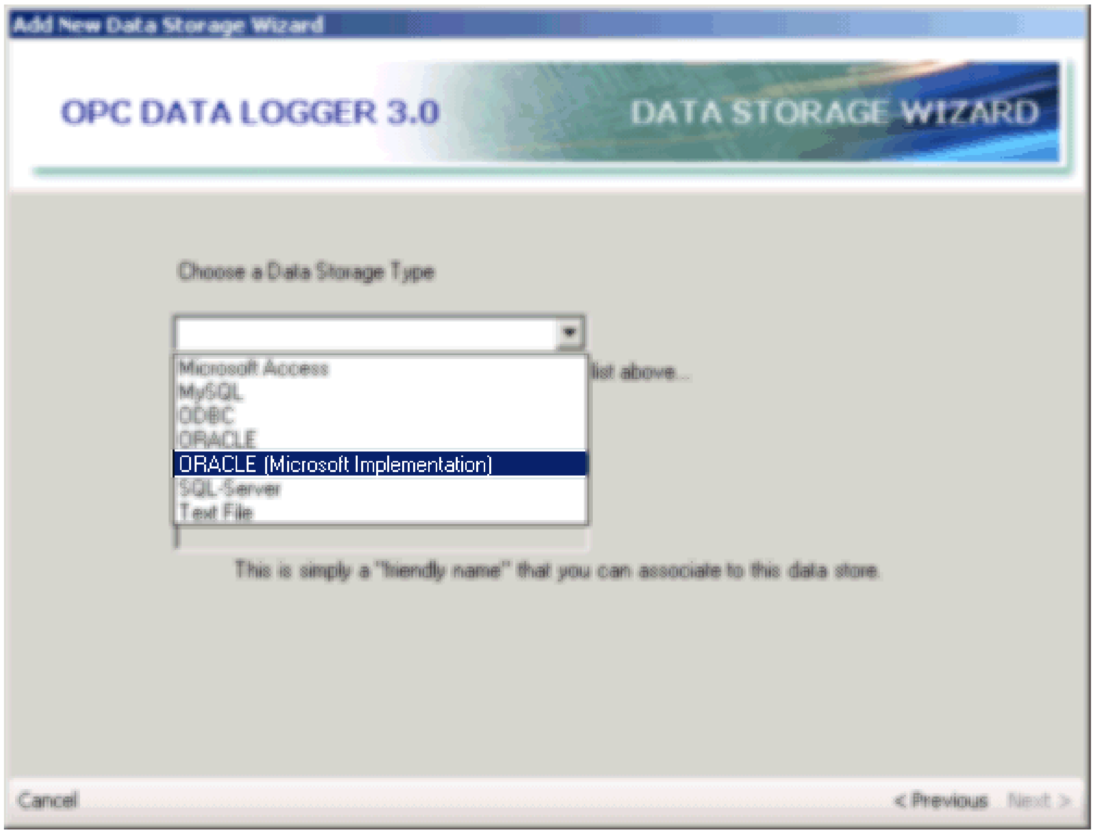
Now that the database has been configured, the items have been imported into the database and OPC Data Logger, and we have defined our Detail mode presentation. We can now complete the configuration process by binding our detail-mode formatter to the database table. Once this is done, we can begin logging data.

1. Right-click on the **Data Storage** node within the treeview and choose the **Data Storage Wizard**:



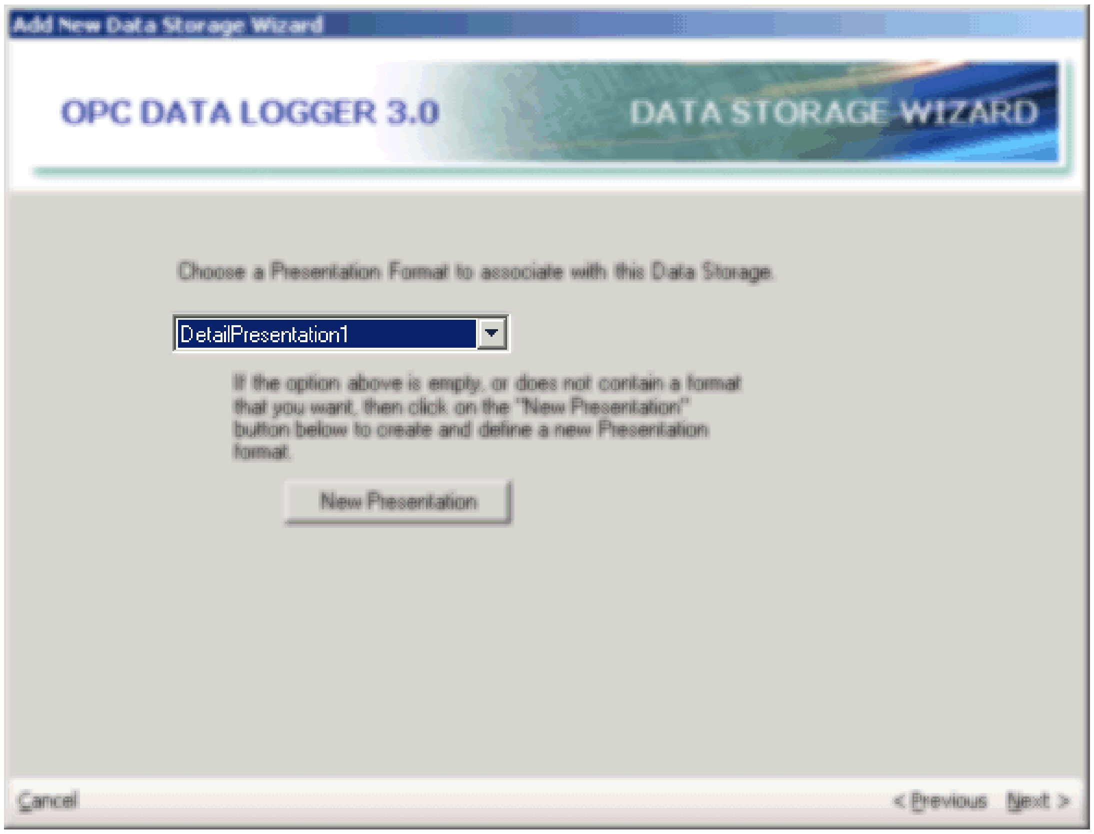
2. The wizard will begin. Press **Next** to bypass the welcome screen.

3. Next, choose your database type from the list. In this example we will be connecting to a remote ORACLE database:


Add New Data Storage Wizard
OPC DATA LOGGER 3.0
DATA STORAGE WIZARD
Choose a Data Storage Type
Microsoft Access
MySQL
ODBC
ORACLE
ORACLE (Microsoft Implementation)
SQL-Server
Text File
list above...
This is simply a "friendly name" that you can associate to this data store.
Cancel
< Previous
Next >

Then click the **Next** button.

4. Pick the Detail presentation that was previously created:



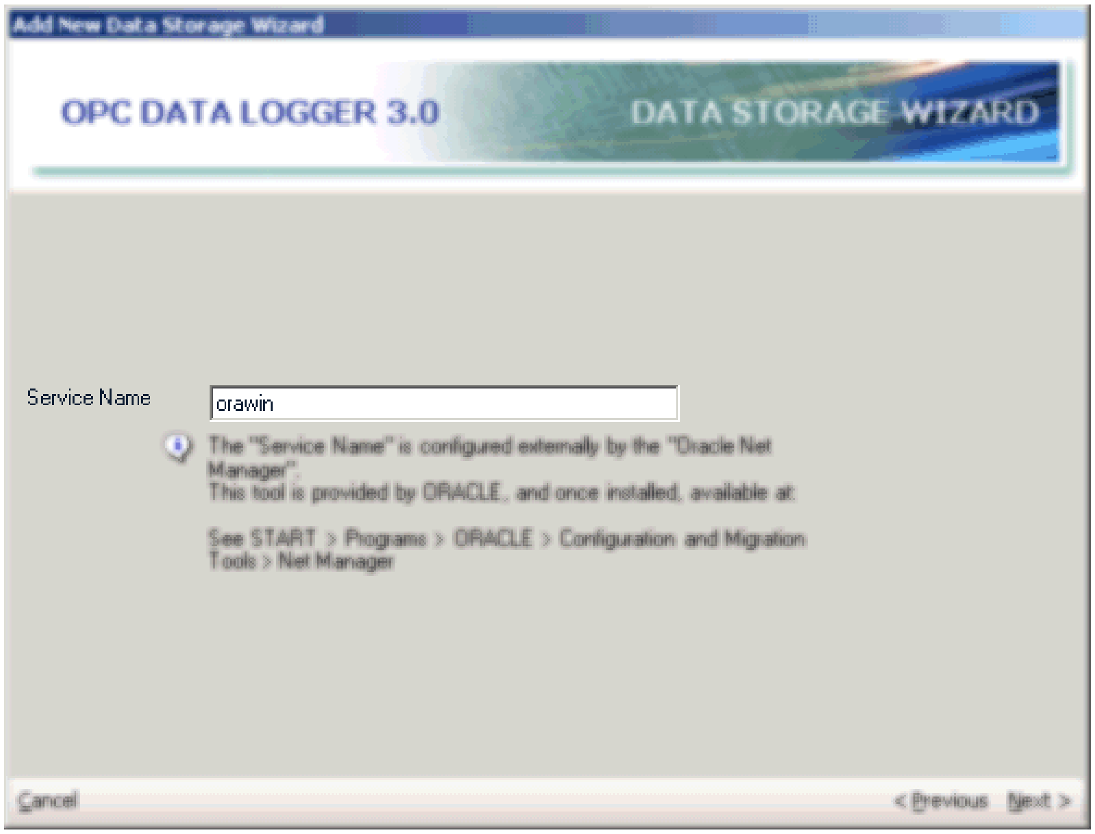
Choose a Presentation Format to associate with this Data Storage.

DetailPresentation1

If the option above is empty, or does not contain a format that you want, then click on the "New Presentation" button below to create and define a new Presentation format.

Then click the **Next** button.

5. In the case of connecting to an ORACLE database, the following window will be displayed requiring the entry of a valid Service Name. This does not apply to any other type of database:



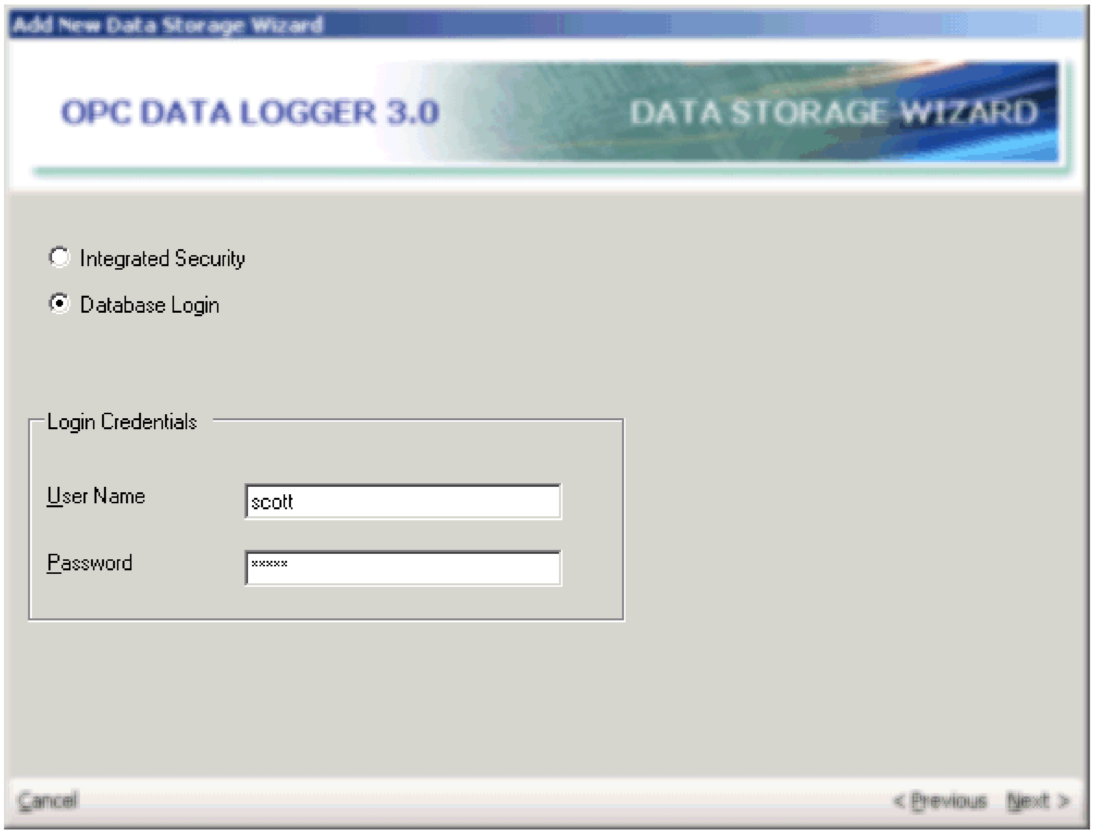
Service Name: orawin

The "Service Name" is configured externally by the "Oracle Net Manager".
This tool is provided by ORACLE, and once installed, available at:

See START > Programs > ORACLE > Configuration and Migration Tools > Net Manager

Simply click the **Next** button to proceed.

6. Now specify how you will log into the database:



( ) Integrated Security
(●) Database Login

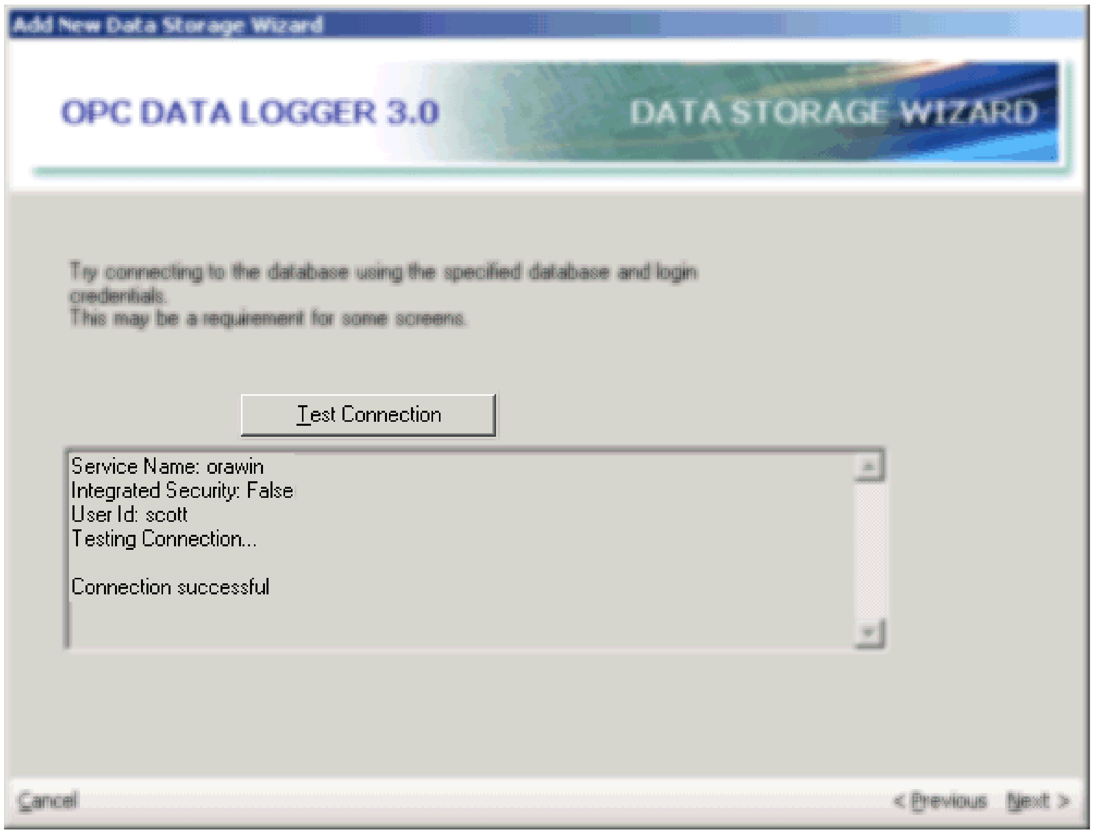
Login Credentials

| | |
|---|---|
| User Name | scott |
| Password | ***** |

In this case we will be using the **scott\tiger** default account installed by Oracle.

Click the **NEXT** button.

7. Now you will need to test your configuration:



Try connecting to the database using the specified database and login credentials.
This may be a requirement for some screens.

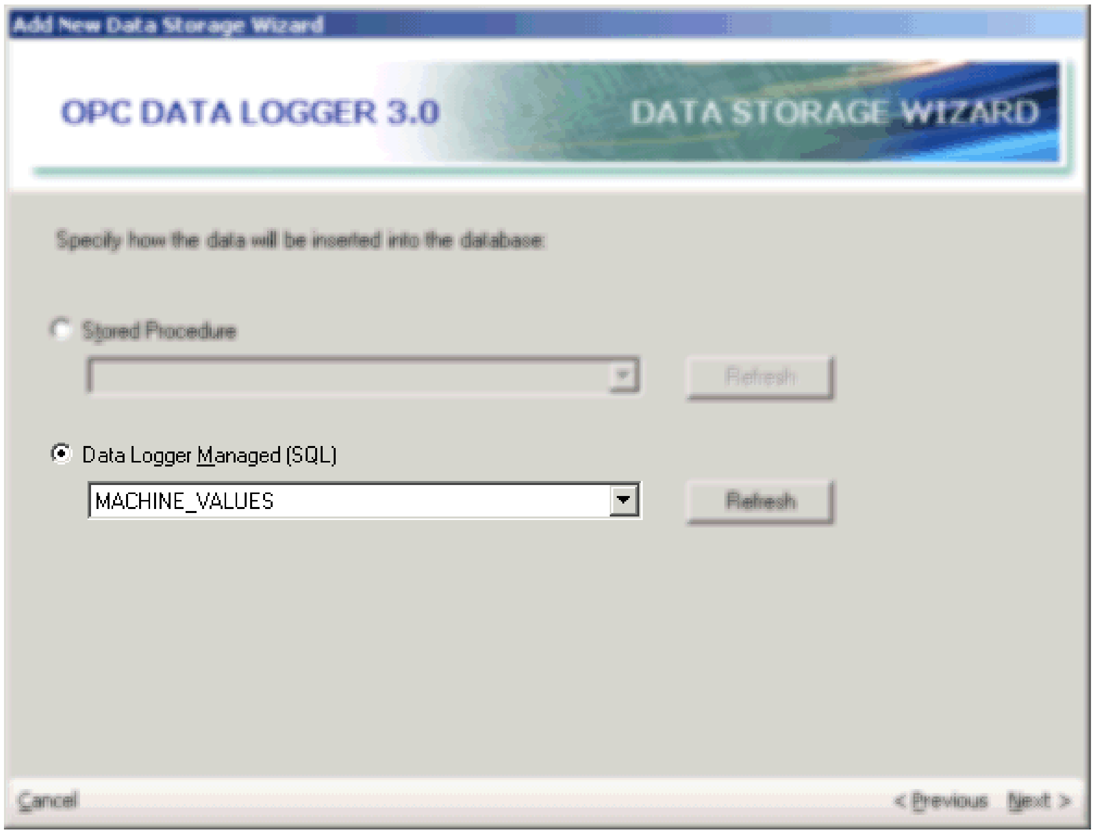
```
Service Name: orawin
Integrated Security: False
User Id: scott
Testing Connection...

Connection successful
```

It is important that the test is successful for the wizard to proceed.

Click the **Next** button to continue.

8. Now we will pick our MACHINE_VALUES table from the list of available tables, because this is the table where we will log our data to:



Specify how the data will be inserted into the database:

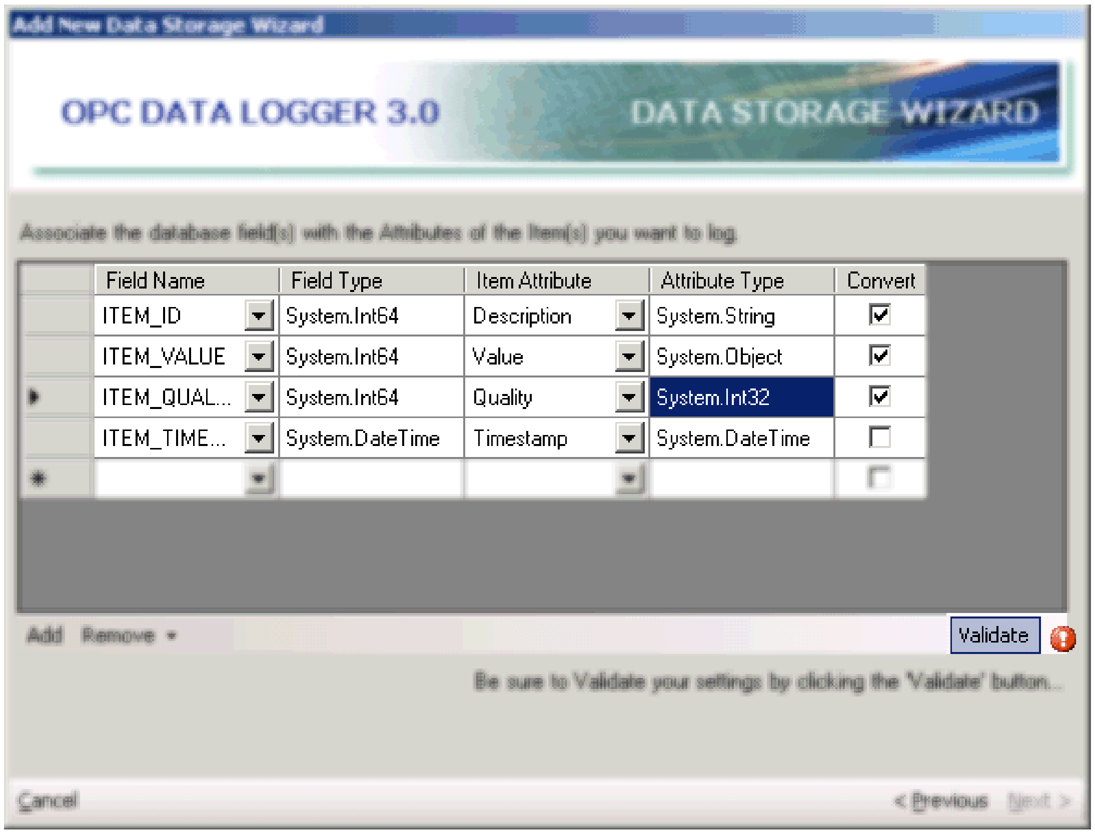
( ) Stored Procedure

(●) Data Logger Managed (SQL)

MACHINE_VALUES

Click the **Next** button to continue.

9. Now we must configure our bindings, that is, the relationship between the item values within the OPC Data Logger and the fields within our selected table:

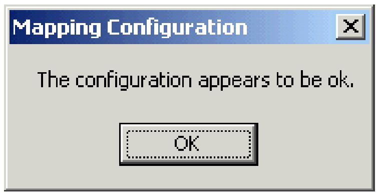


Associate the database field(s) with the Attributes of the Item(s) you want to log.

| Field Name | Field Type | Item Attribute | Attribute Type | Convert |
|---|---|---|---|---|
| ITEM_ID | System.Int64 | Description | System.String | ✓ |
| ITEM_VALUE | System.Int64 | Value | System.Object | ✓ |
| ITEM_QUAL... | System.Int64 | Quality | System.Int32 | ✓ |
| ITEM_TIME... | System.DateTime | Timestamp | System.DateTime | |

Be sure to Validate your settings by clicking the 'Validate' button...

You must click the **Validate** button to verify that your configuration is valid.



The configuration appears to be ok.

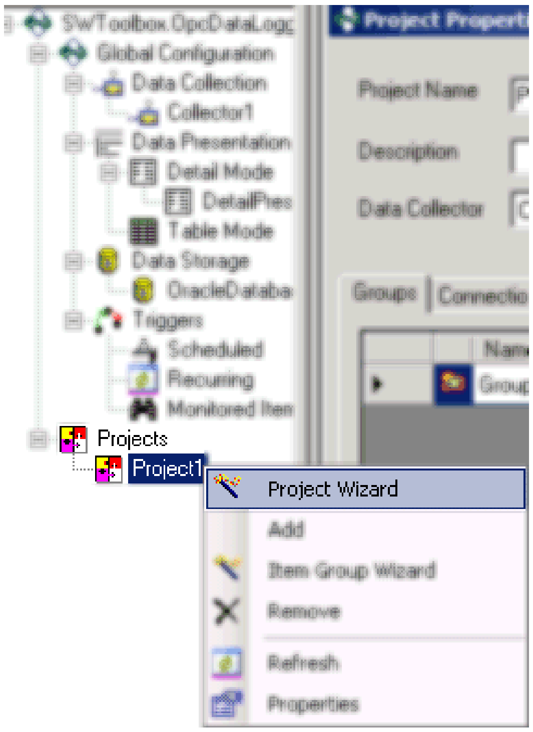
Press the **Next** button to proceed.

10. The wizard should now summarize what you have just done, simply click the **Finish** button and then save your OPC Data Logger configuration.

## Last Step – Connecting the Data Collection to the Data Storage

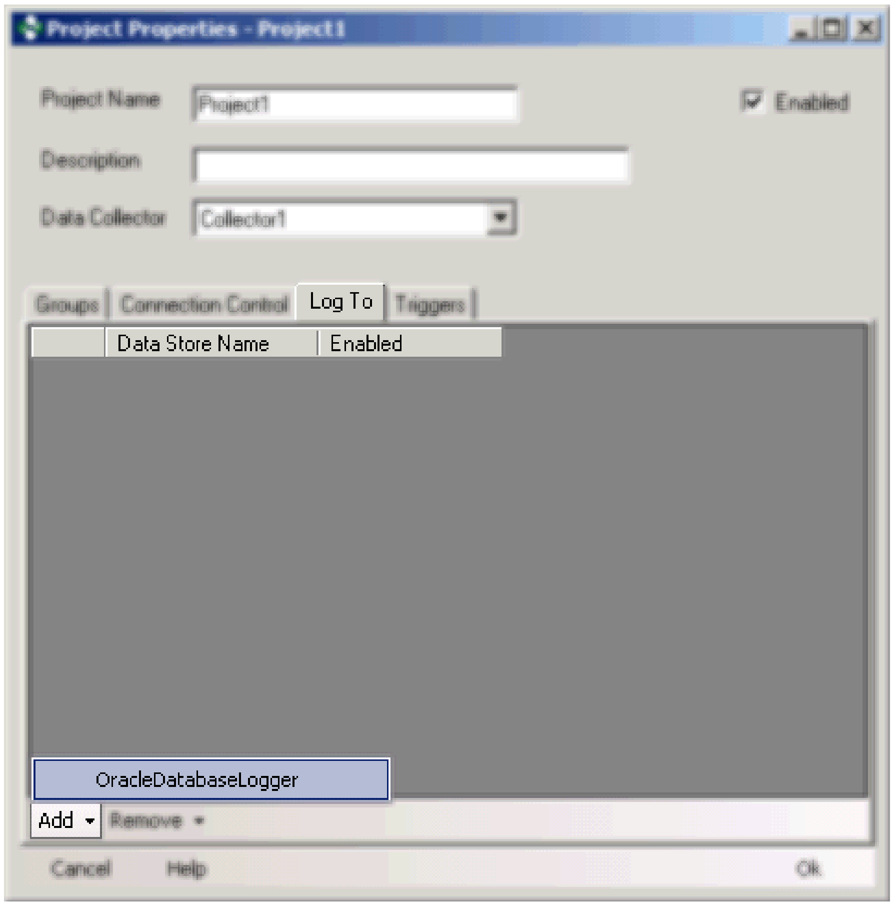
Now that the data collection has been defined, the items added, presentation format defined, and the database connection defined, we are now ready to bring it all together.

1. Open or create a new **Project** under the **Projects** node in the OPC Data Logger tree-view:



TIP: Whenever you are creating a new project, use the **Project Wizard** as it dramatically simplifies the configuration process and ensures the configuration is valid and correct.

2. Open the **Project** properties window and click on the **Log To** tab.



At the bottom of the window click on the **Add** button, and you will see the available data stores available, in this case we have our only Oracle database storage component called *OracleDatabaseLogger*.

Click **OK** to save and close the screen.

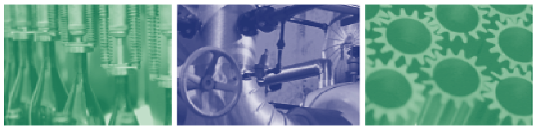
That's it! We're ready to log data.

## Further Optimization – Stored Procedures vs SQL Injection

In this example we have used direct SQL injection to log the data to the MACHINE_VALUES table. See step 8 on Page 26. This is not optimal. This is a slower method of logging data. The reason being is that we are sending a SQL statement to the database, which then must be parsed and validated by the database server prior to compilation and final execution. This must happen on every time we send data to the database.

The obvious step would to try to avoid the parsing/compilation needed by the database server each time we log data. This can be accomplished by creating a **Stored Procedure**.

Creating a Stored Procedure is outside the scope of this document as the procedure is similar (but not the same) between all of the databases that support them.



## Example Reports

To complete this case-study, we will take a look at some simple SQL Queries for reporting.

NOTE: The following SQL queries were creating against an ORACLE 10i database server; consequently, the exact syntax may not be compatible with other database engines.

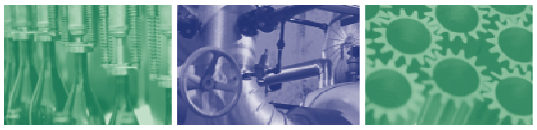
### Retrieving values logged for an item between a date-range

```
select
  scott.machine_tags.tagname,
  scott.machine_values.item_value,
  scott.machine_values.item_quality,
  scott.machine_values.item_timestamp
from
  scott.machine_values, scott.machine_tags
where
  scott.machine_values.item_id = scott.machine_tags.tag_id
  and
  scott.machine_tags.tagname like 'channel1.device1.tag1'

;
```

This produces the following output:

```
TAGNAME                                  ITEM_VALUE ITEM_QUALITY ITEM_TIME
---------------------------------------- ---------- ------------ ---------
channel1.device1.tag1                            10          192 01-JAN-08
channel1.device1.tag1                            20          192 01-JAN-08
channel1.device1.tag1                            30            0 01-JAN-08
```



## Retrieving a count of logged values for all items between a date-range

```
select
  scott.machine_tags.tagname,
  scott.machine_values.item_timestamp,
  count (*)
from
  scott.machine_values,
  scott.machine_tags
where
  scott.machine_values.item_id = scott.machine_tags.tag_id
  and
  scott.machine_values.item_timestamp
    between TO_DATE('1-JAN-2008 12:00:00 AM', 'dd-Mon-yyyy HH:MI:SS AM')
      and   TO_DATE('1-FEB-2008 12:00:00 AM', 'dd-Mon-yyyy HH:MI:SS AM')
group by
  scott.machine_tags.tagname,
  scott.machine_values.item_timestamp
;
```

This produces the following output:

```
TAGNAME                                  ITEM_TIME   COUNT(*)
---------------------------------------- --------- ----------
channel1.device1.tag1                    01-JAN-08          3
channel1.device1.tag2                    01-JAN-08          2
channel1.device1.tag3                    01-JAN-08          1
```