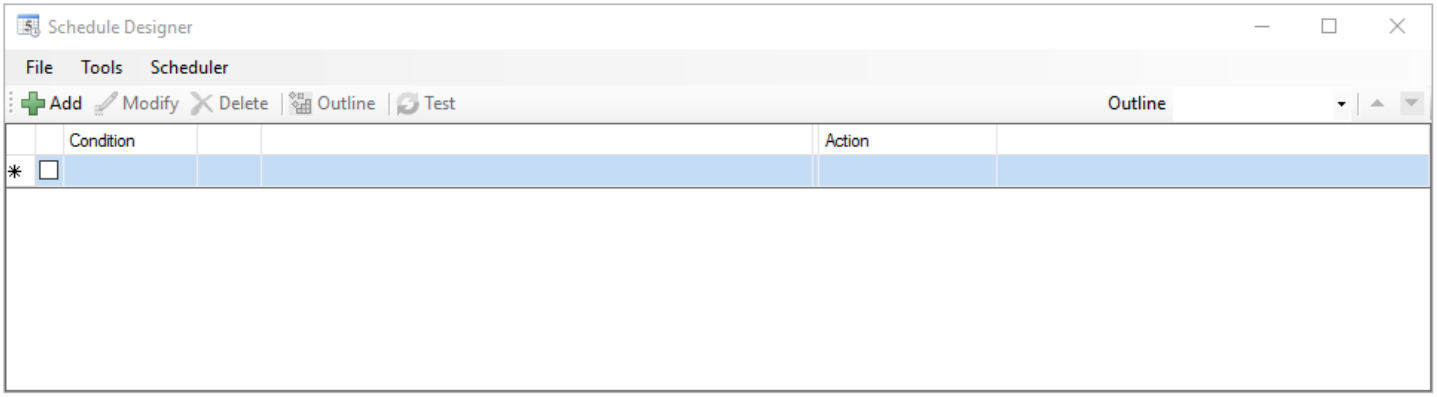
Event-Based Reporting

Overview

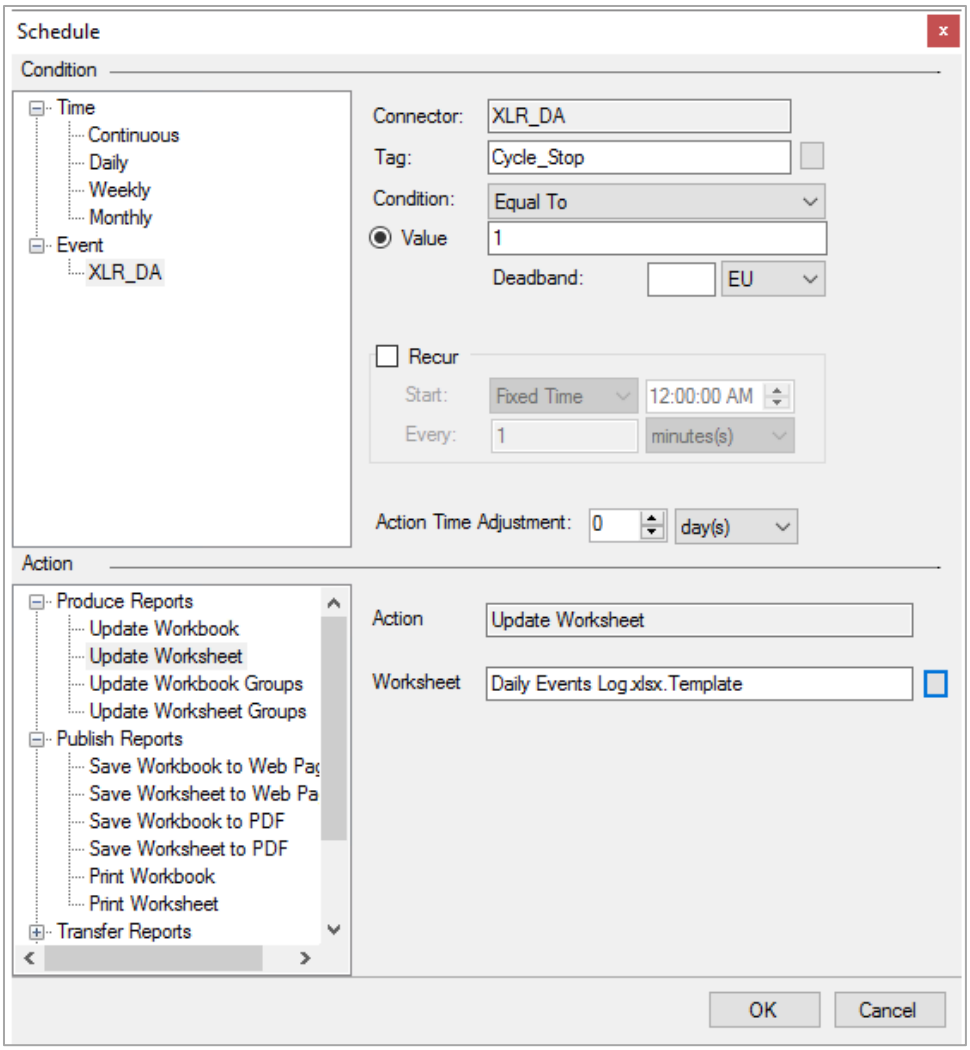
In many reporting applications, the most accurate way to collect data is based on a “trigger” condition in the process. In other words, the live KPI in a PLC may be most relevant at the exact moment in time when a machine has completed a job. Or, a trend graph must display data only for the duration of that job. Other times, there are multiple events occurring in a process that must be included, in time-series order, on a summary report. All of these scenarios, and many more, can be handled in **XLReporter** by implementing an **Event-Based** schedule.

Schedule

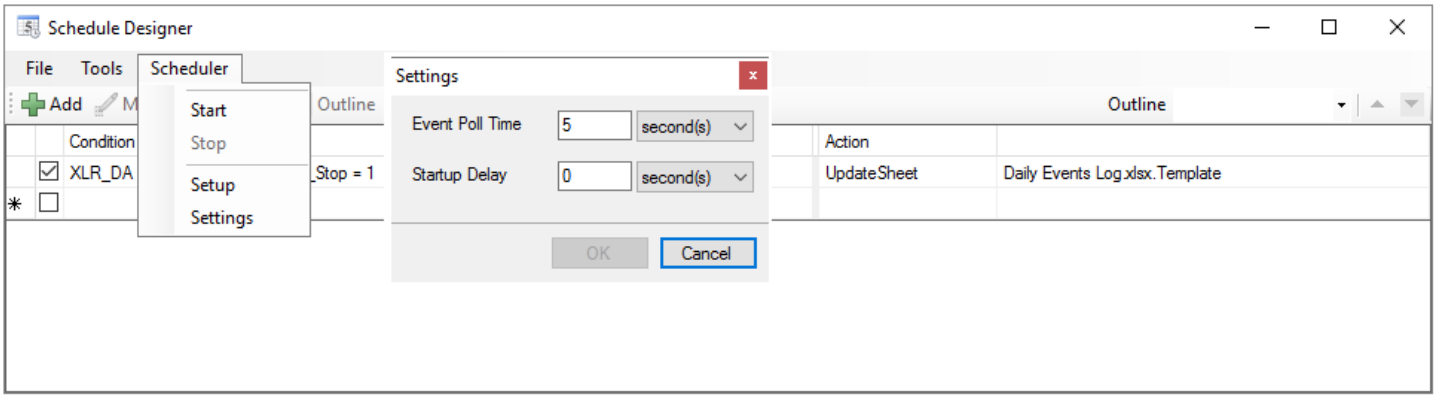
At the highest level, event-based reports are driven from the **XLReporter Project Schedule**. The schedule organizes automated reports for the scope of the entire reporting application and can be configured from the **Project Explorer** under the **Project** tab by selecting **Schedule**, **Designer.**



The **Add** button is used to add a new **Schedule Action.** Each **Action** has an associated **Condition**, which can be based on the system time, or based on a tag value read through a **Connector** defined for a real time source, like a PLC or HMI.



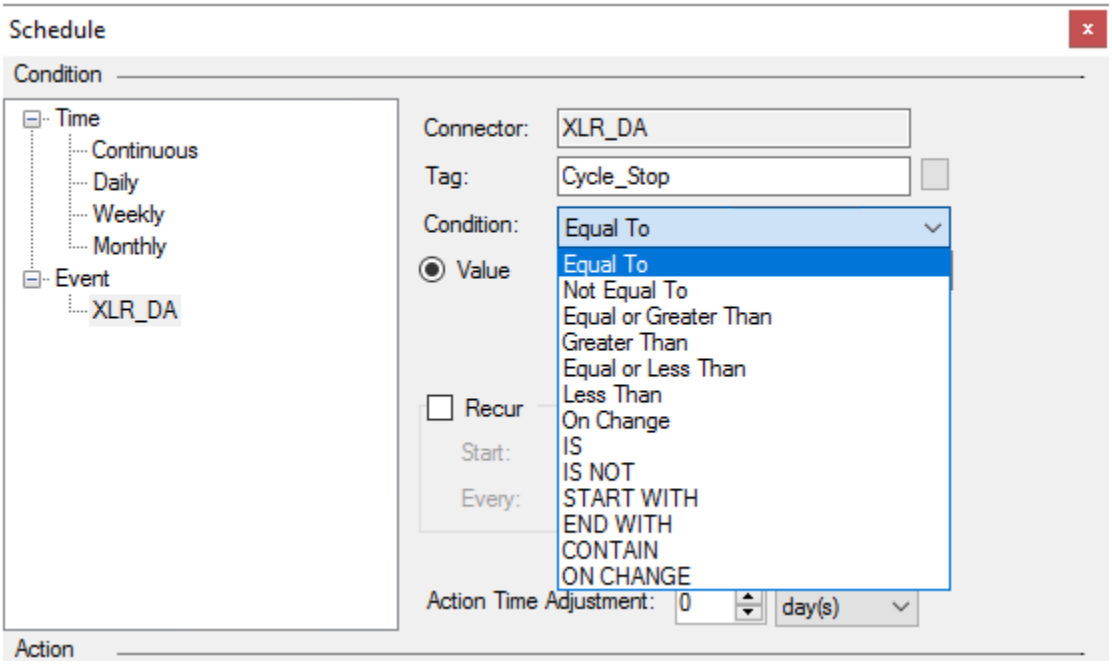
In the example image above, a **Daily Event Log** report is scheduled to update whenever the value of the **Cycle_Stop** tag transitions from any other value to the value of *1*.



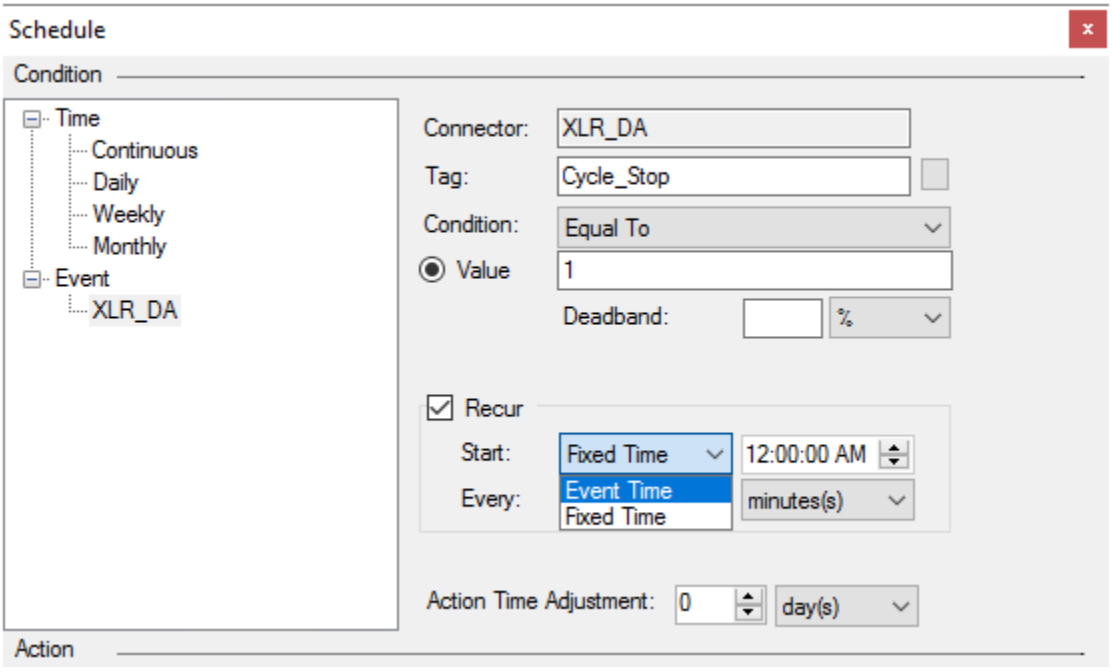
With Event-Based scheduling, the **XLReporter Scheduler** polls the tag(s) set as conditions every **5 seconds** by default. This can be lowered to 1 second if necessary. If a transition into the **Condition** value from another value (such as a transition *from* 0 *to* 1) is detected from one polling cycle to the next, the **Action** is processed.

Event-Based Conditions

There are a variety of configurable **Event Conditions.**



The first half of conditions, which are spelled out with mixed casing, e.g. “Equal To”, operate on numeric tags such as floating point, integer, or Boolean values. The second half, spelled out in ALL CAPS, operate on string-based tags.



The **Recur** parameter acts similarly to a *while* loop in that once the **Condition** is detected, as long as it remains true, the **Action** executes at the specified interval until the **Condition** is no longer true. The recurrence can be based on the time of the **Event Condition** or on the **Fixed** system time.

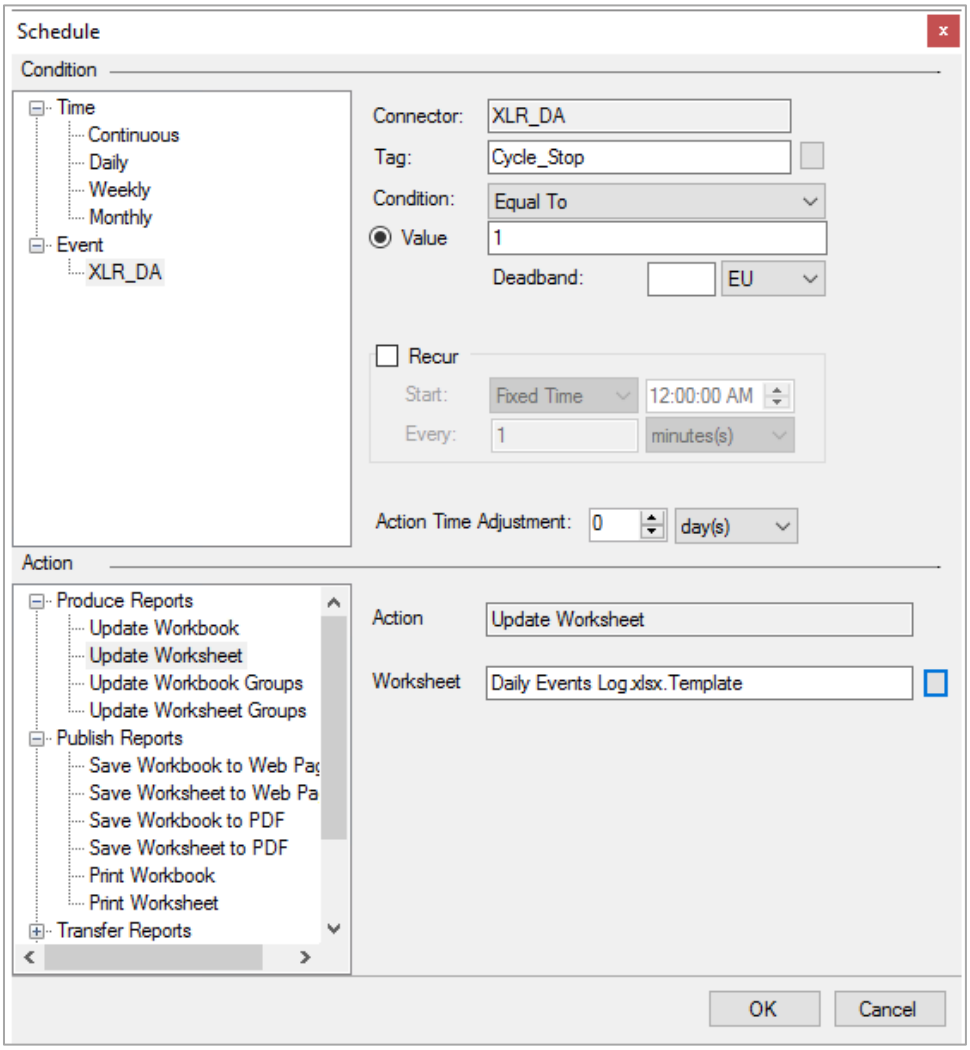
Event-Based Report Scenarios

Below are a few scenarios where event based reporting is used.

Daily Cycle Report

In this scenario a report must be generated for the day displaying values from the process at the end of every cycle ran for the day.

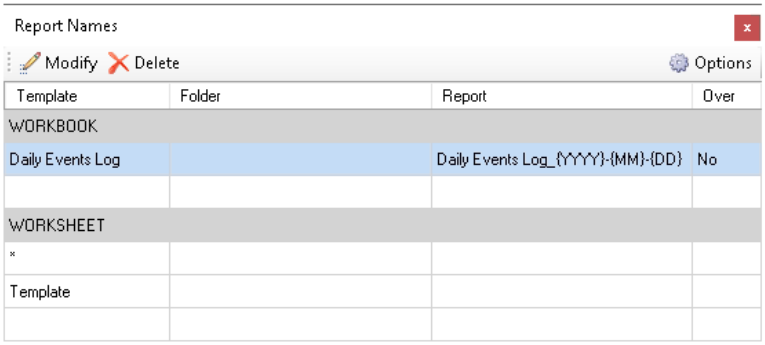
Schedule



The **Condition** used on the schedule represents the end of the process event being monitored, e.g. *Cycle_Stop =1*.

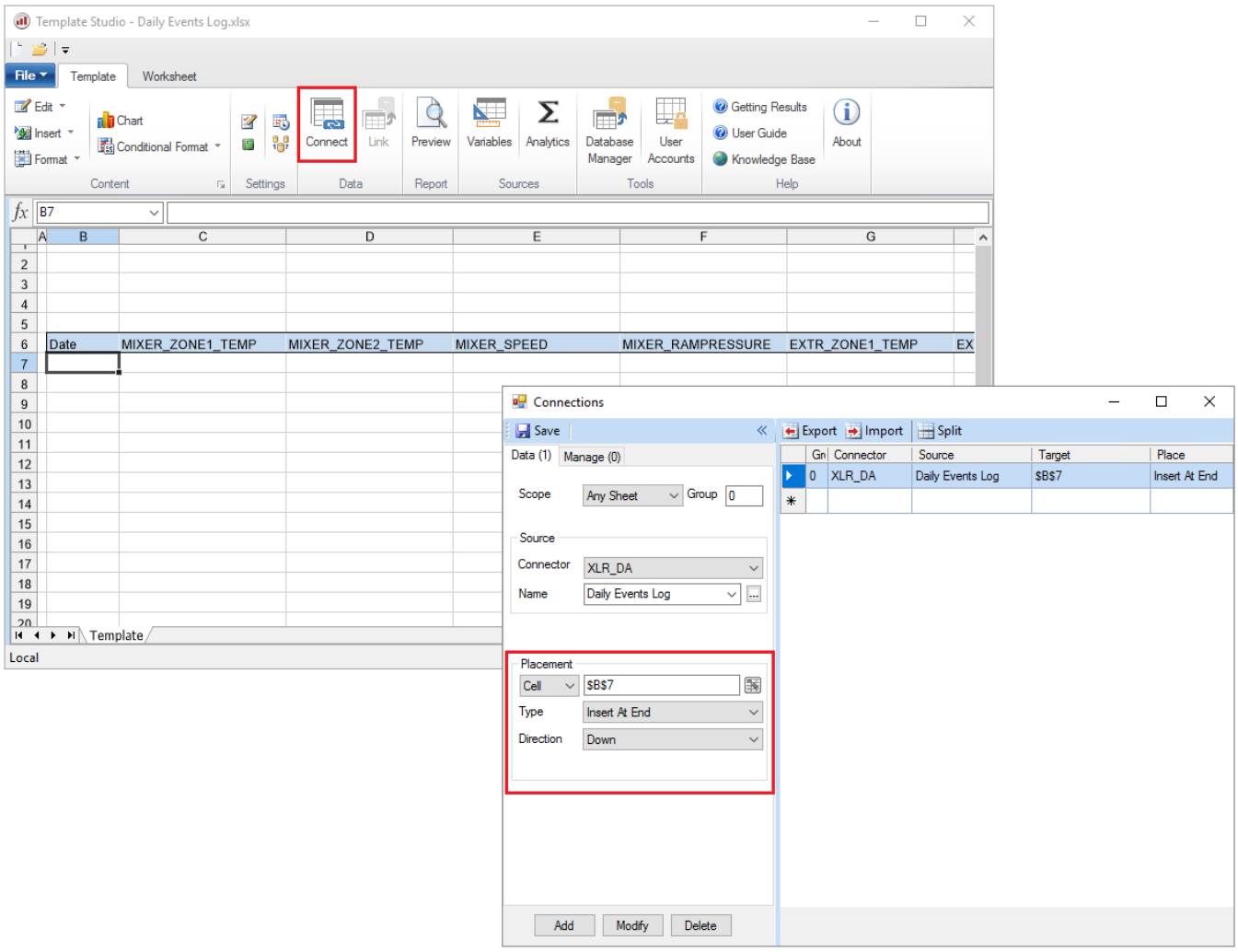
Naming Convention

Because the events are collated into separate worksheets based on the date, the **Workbook Name** uses the **Calendar Variables** *{YYYY}, {MM}, and {DD}*.

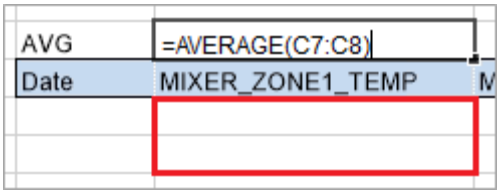


Data Placement

When the report is updated, the real time data must be placed onto the next available row below the existing data. For that reason, the **Placement Type** used is *Insert at End, Down.*



Charts, Formulas and Formatting



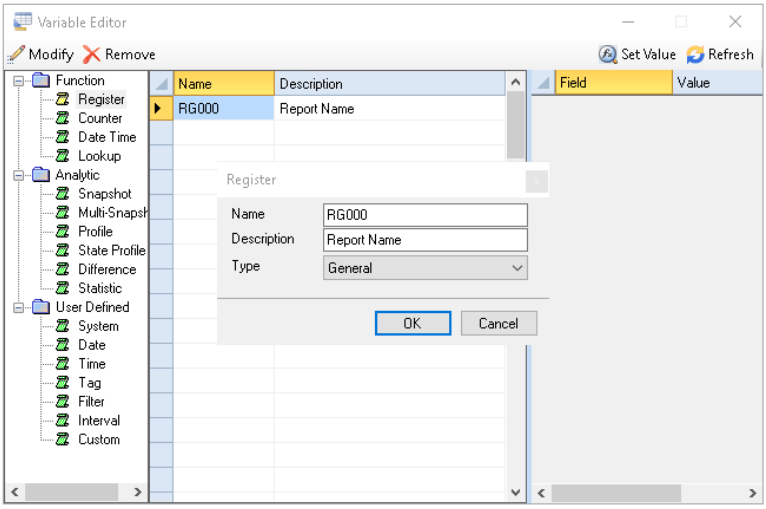
The data is inserted in *B7*. If any formulas, chart series, or formatting (including conditional formatting) are configured for rows 7 and 8, the formula or series ranges will automatically expand as each event row is collected and inserted into the report. In other words, configuring these functions to the first two data rows allows them to adapt dynamically to variable row counts. This is important because we don't know how many cycles will run each day.

Cycle Report

In this scenario a report is required for each cycle that captures the cycle ID at the beginning of the cycle and adds data to the report continuously throughout the cycle.

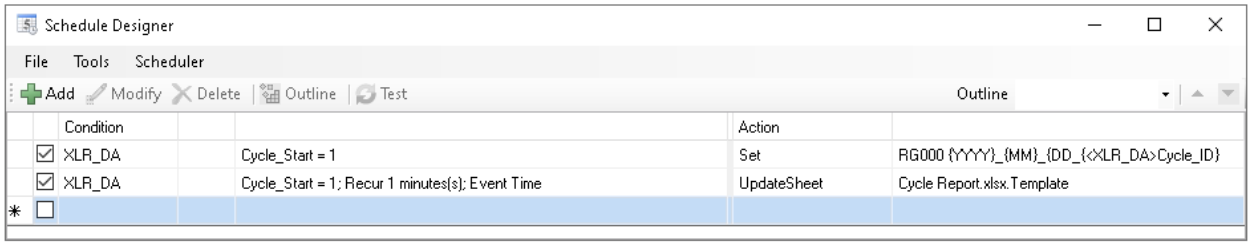
Variable

This configuration requires a **Variable** to be added to the project to manage the naming convention of the report. A variable can be added from the **Data** tab of **Project Explorer** by clicking **Variables** to open the **Variable Editor**.

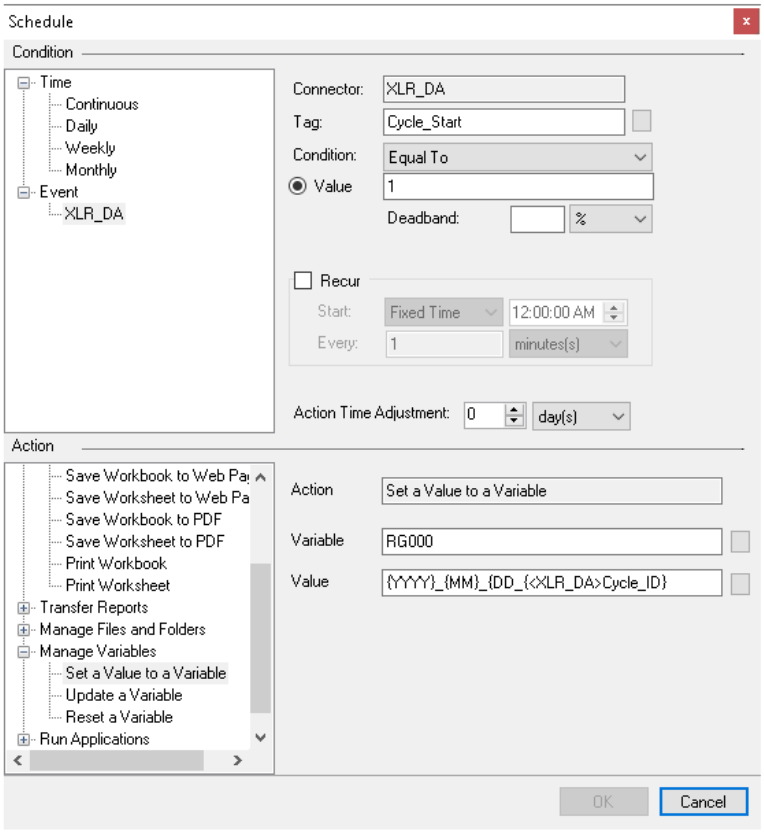


Register variables are provided to hold information like this. To configure a Register under **Function** select **Register**. Select an empty line and click **Modify** to add a new **Register**.

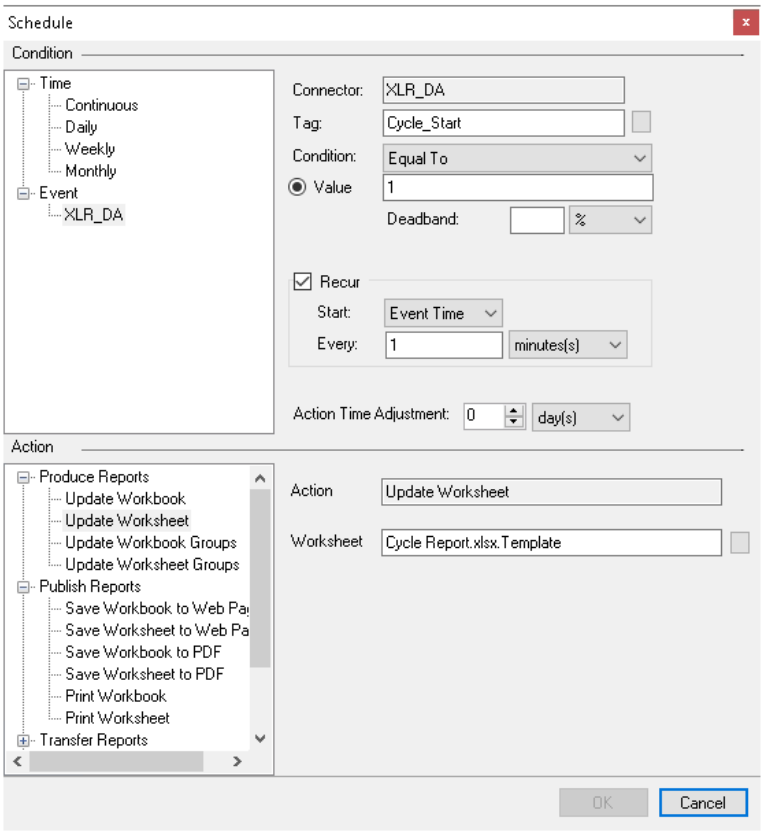
Schedule



At the cycle start **Condition,** a **Set a Value to a Variable** action is used to set the value of the **Variable** *RG000* to a concatenation of the date and the value of the *Cycle_ID* tag.

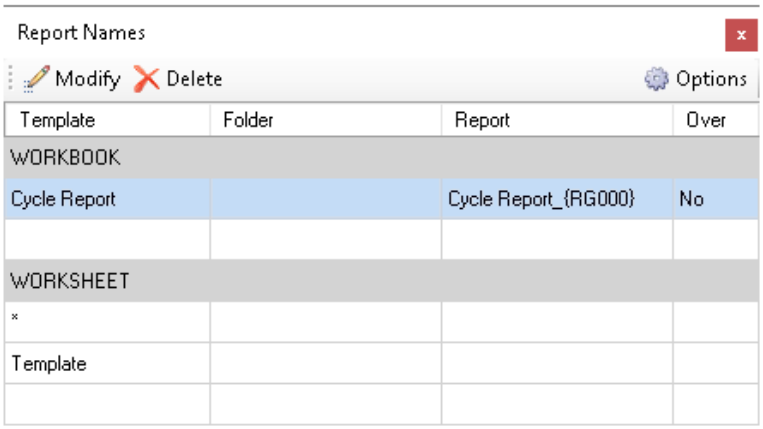


This can be built by clicking the browse button […] for Value. The Calendar Item list provides keywords for year, month and day. The Connector Item branch opens a tag browser to select tags from real time connectors defined in the project.



The second action is configured with the same **Condition** as the first, but the **Recur** setting is enabled so that report is updated continuously while the cycle is running (Cycle_Start = 1).

Naming Convention



The naming convention for this configuration sets the **WORKBOOK Name** to *{RG000}*. This allows the scheduler to generate a discrete report for each unique value of the variable, or in other words, each unique cycle ID combined with each unique date.

Data Placement

Like the previous scenario, the real time **Data Connection** is placed with the type *Insert at End, Down.*

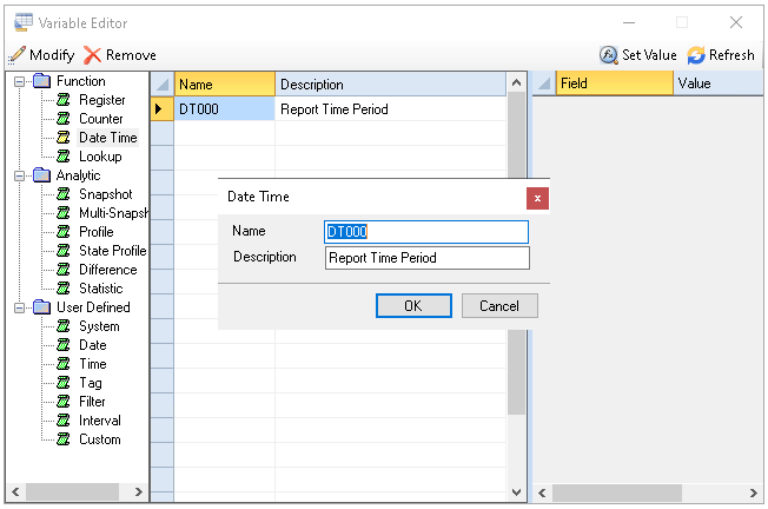
Batch History Report

In this scenario a report is required for each batch. The data for the batch report is stored in a continuous historian.

Variable

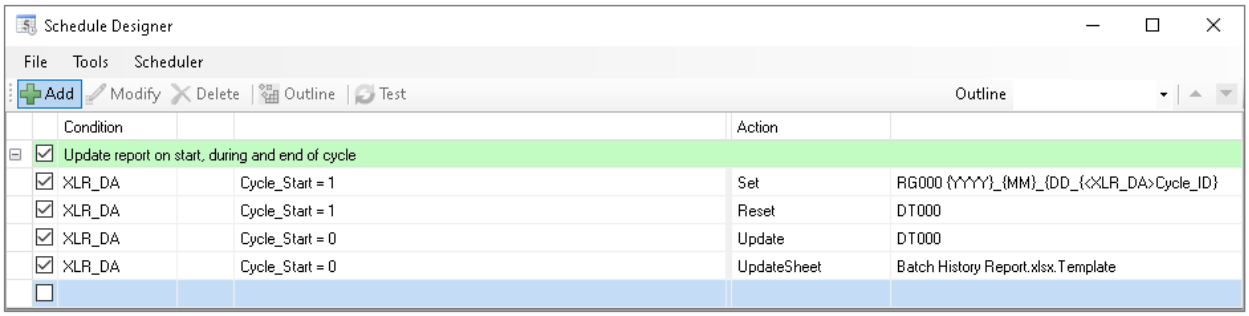
Like the previous example, this configuration uses a **Function, Register** variable in the naming convention, and it is set at the start of the cycle event.

This configuration also includes a **Function, DateTime** variable to track the start and stop of the event cycle for the purposes of querying the historical data.

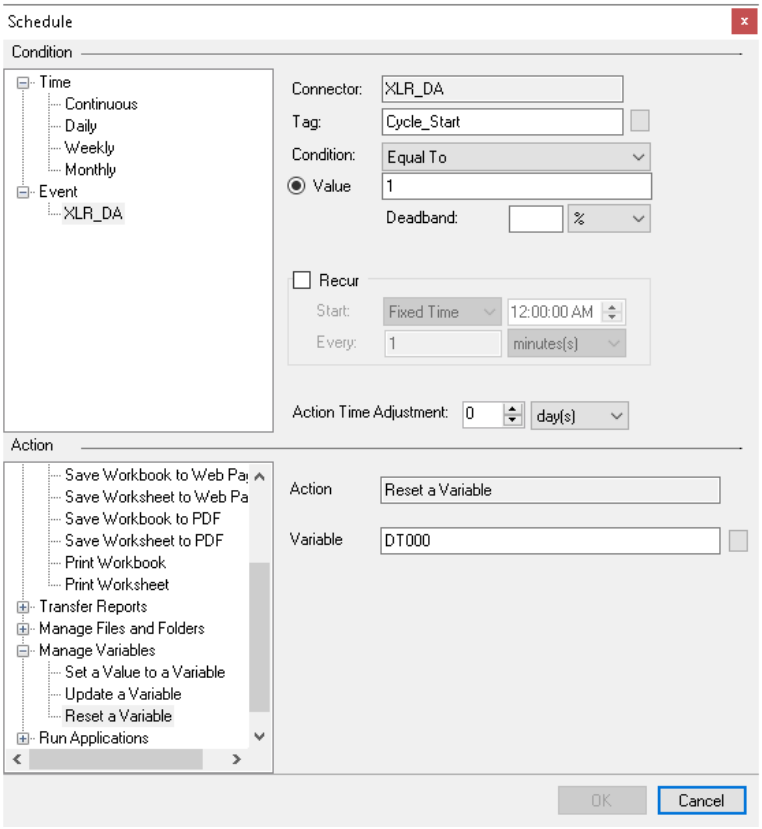


Schedule

The schedule includes four **Actions** in total. The first action, the value of *RG000* (the report name variable) is set to the value of the *Cycle ID* tag.



In the second action, the *Reset* action is scheduled for the *DT000* variable. This sets the **Start Date** and **Start Time** fields of the variable to the time the cycle started.

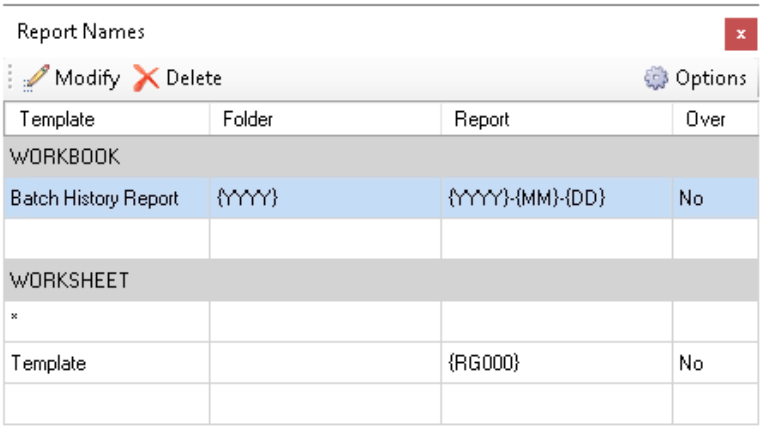


The third action is processed at the end of the cycle event, in this case when *Cycle_ID* transitions to a value of *0.* This action runs the *Update* action on the *DT000* variable, which sets the **End Date** and **End Time** fields of the variable to the time the cycle ended.

Finally, the fourth action, also processed at the end of the cycle event, updates the *Batch History Report* template. Since the cycle ID, and its start and stop times have been established at this point, all of the necessary information is available and the report can be generated.

Naming Convention

This configuration expands on the naming convention discussed in the previous example. In that example, the *{RG000}* variable was set on the schedule to a concatenation of the cycle ID and the date. That way, a unique file would be generated based on a given cycle ID occurring on a given day.

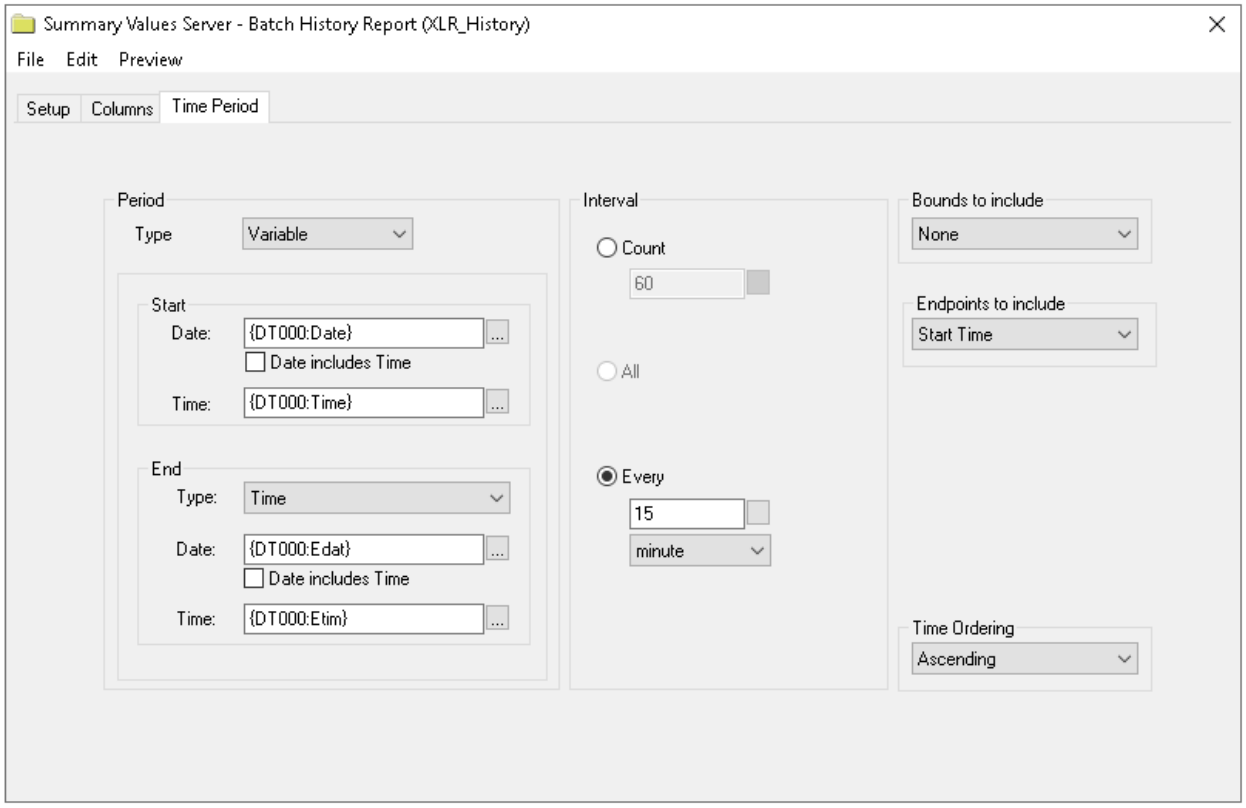


Template	Folder	Report	Over
WORKBOOK			
Batch History Report	{YYYY}	{YYYY}-{MM}-{DD}	No
WORKSHEET			
*			
Template		{RG000}	No

This configuration uses the **Folder** parameter to create a subfolder in the output based on the year. The **Workbook Name** is set to the year, month, and day. The **Template Sheet Name** is set to the variable *{RG000}.* So the report for cycle *ABC123* which occurred on 3/20/2020 will be stored in [Project Report Path]\Batch History Report\2020\2020-03-20.xlsx. In a worksheet called *ABC123*. If multiple cycles occurred that day, they would be placed in additional sheets in the same file. To instead store each cycle report in a different file, use a naming convention similar to the previous example.

Data Connection

Similarly to both of the previous examples, the **Data Connection** in this template is placed using the *Insert at End, Down* **Placement Type**. However, because this template uses a **Historical Data Group** as the data source, the **Time Period** is defined based on the *DT000* variable.



At the runtime of the report, which is triggered at the end of the cycle event, these variable fields resolve to the times that the event started and stopped respectively. The samples are collected every 15 minutes throughout the cycle based on the **Interval** parameter.