AVEVA™
Historian Scenarios Guide
formerly Wonderware

Publication date: Wednesday, November 11, 2020

Contact Information

AVEVA Group plc
High Cross
Madingley Road
Cambridge
CB3 0HB. UK

https://sw.aveva.com/

For information on how to contact sales and customer training, see https://sw.aveva.com/contact.

For information on how to contact technical support, see https://sw.aveva.com/support.

Contents

Welcome

In your plant, you work hard to manage large amounts of data. While you and other AVEVA Historian customers may have very different processes and products, you probably do share some key issues around managing your data and using it to better run your operations.

This book discusses ten common scenarios where AVEVA Historian can help. They are:

1. Using hourly and daily reports.
2. Tracking inventory and production with packing line counters
3. Tracking liquid and gas use with custody meters.
4. Analyzing cycle time for your process.
5. Calculating the total flow.
6. Calculating downtime statistics.
7. Pinpointing reasons for downtime.
8. Filling in the gaps in your data.
9. Understanding and using data.
10. Getting summary statistics when a certain condition applies

AVEVA Historian Documentation Set

The AVEVA Historian documentation set includes the following guides:

- *AVEVA System Platform Installation Guide*
  This guide provides information on installing the AVEVA Historian, including hardware and software requirements and migration instructions.
- *AVEVA Historian Concepts Guide*
  This guide provides an overview of the entire AVEVA Historian system and its key components.
- *AVEVA Historian Scenarios Guide*
  This guide discusses how to use AVEVA Historian to address some common customer scenarios.
- *AVEVA Historian Administration Guide*
  This guide describes how to administer and maintain an installed AVEVA Historian, such as configuring data acquisition and storage, managing security, and monitoring the system.
- *AVEVA Historian Retrieval Guide*
  This guide describes the retrieval modes and options that you can use to retrieve your data.
- *AVEVA Historian Database Reference*
  This guide provides documentation for all of the AVEVA Historian database entities, such as tables, views, and stored procedures.
- *AVEVA Historian Glossary*
  This guide provides definitions for terms used throughout the documentation set.

In addition, the *AVEVA License Manager Guide* describes the AVEVA License Manager and how to use it to install, maintain, and delete licenses and license servers on local and remote computers.

Scenario 1: Use hourly and daily reports

You can use the AnalogSummaryHistory view to regularly track your plant's productivity. For example, you can answer questions such as:

- What was the peak temperature each hour?
- How much product did we produce today?

What is Analog Summary?

For analog data (such as temperature of a boiler or volume within a tank), you can retrieve summary data using the AnalogSummaryHistory view. For even faster retrieval, you can configure the tags to be stored with specific summary data and retrieve that summary data later. You can use a variety of available summary statistics, including:

- Time-weighted average
- Standard deviation
- First, last, minimum, or maximum value for a timestamped period

About the AnalogSummaryHistory View

AVEVA Historian stores historical data in history blocks rather than in the database itself. The data is accessible via extension tables.

With AnalogSummaryHistory, you can query summary statistics for analog tags. The AnalogSummaryHistory view lets you return multiple statistics for a single tag within one query.

Example: Creating a Daily Report

The following query returns the minimum, maximum, and average values for the R31.ReactTemp tag for two days of data at 24-hour resolution.

```
SELECT StartDateTime, TagName, Minimum, Maximum, Average
   FROM AnalogSummaryHistory
      WHERE TagName='R31.ReactTemp'
      AND StartDateTime >= '2015-08-21'
      AND EndDateTime <= '2015-08-23'
      AND wwResolution=24*60*60*1000
```

The results are:

StartDateTime	TagName	Minimum	Maximum	Average
2015-08-21 01:00:00.0000000	R31.ReactTemp	181	211	201
2015-08-22 01:00:00.0000000	R31.ReactTemp	177	219	200

Scenario 2: Track inventory and production with packing line counters

You can use the Counter retrieval feature to calculate production rates and totals. For example, you could answer questions like:

- How many screw tops have we used in the last week?
- How many products have we produced since Monday?

What is Counter Retrieval?

With Counter retrieval, you can easily calculate the amount of increase in a tag value (and therefore the number of items produced) over time. For example, you might have an integer counter that keeps track of how many cartons were produced. The counter has an indicator like this:

```
[ 9 ][ 9 ][ 9 ][ 9 ]
```

Some counters are reset manually -- for example, at the end of a shift before restocking. Others may never be reset, and instead will simply rollover, for example from 9999 to 0. Calculating totals accurately depends on:

- Whether the counter has rolled over
- Whether the counter gets reset manually

About Rollover Values

The rollover value is defined in the AVEVA Historian for each tag.

The next value after the highest value that can be physically shown by the counter is called the rollover value. For example, the rollover value for a four-digit integer counter is 10,000. When the counter reaches the 9,999th value, the counter rolls back to 0. Therefore, a counter value of 9,900 at one time and a value of 100 at a later time means that you have produced 200 units during that period, even though the counter value has dropped by 9,800 (9,900 minus 100). Counter retrieval allows you to handle this situation and calculate the correct value.

Similarly, if you had a PLC register that is a 2-byte unsigned integer, the rollover value would be 65,536.

For each cycle, the counter retrieval mode shows the increase in that counter during the cycle, including rollovers.

Counter retrieval is a true cyclic mode. It returns one row for each tag in the query for each cycle. The number of cycles is based on the specified resolution or cycle count.

The counter algorithm is applied only to analog and discrete tags. For integer analog tags, the result will be an integer returned as a float data type. For a real analog tag, the rollover value and the result may be real values and can include fractional values. For discrete tags, the rollover value is assumed to be 2. If a query contains tags of other types, then no rows are returned for those tags.

Resetting a Counter

If you have a counter that normally gets reset manually before it rolls over, you must set the rollover value for the tag to 0 so that the count is simply how much change occurred since the manual reset.

For example, assume that you have the following data values for five consecutive cycle boundaries, and that the value 0 occurs as the first value within the last cycle:

```
100, 110, 117, 123, 3
```

If you set the rollover value to 0, the counter retrieval mode assumes that the 0 following the value 123 represents a manual reset, and returns a value of 3 for the last cycle, which is assumed to be the count after the manual reset. The value 0 itself does not contribute 1 to the counter value in this case.

If the rollover value is instead set to 200, then the counter retrieval mode assumes that the value 0 represents a normal rollover, and a count of 80 is calculated and returned (200 - 123 + 3). In this case, the value 0 contributes 1 to the counter value, and that is the change from the value 199 to the value 200.

Handling Exceptions

To handle reversals instead of treating them as rollovers, you can use a counter deadband. For more information, see Counter Retrieval - Using a Counter Deadband in the *AVEVA Historian Retrieval Guide.*

For details on how counter retrieval handles "gaps" in data and how to compensate for that in your queries, see Counter Retrieval and Counter Retrieval - How It Works in the *AVEVA Historian Retrieval Guide.*

Example: Counting Items with Counter Retrieval

To use the counter mode, set the following parameter in your query:

```
wwRetrievalMode = 'Counter'
```

In the following example, the rollover value for the TotalProduced.Counter tag is set to 0. In a time span of 3 hours, the tag increments from 0 to 9999 twice. The following query returns the an hourly total count within the queried time span. The QualityDetail of 212 indicates that a counter rollover occurred during the reported time range.

```
SELECT DateTime, TagName, Value, Quality, QualityDetail AS QD
   FROM History
      WHERE TagName like 'TotalProduced.Counter'
      AND DateTime >= '2015-08-17 13:00'
      AND DateTime < '2015-08-17 16:00'
      AND wwResolution=3600000
      AND wwRetrievalMode='counter'
      AND wwQualityRule='optimistic'
      AND wwTimeStampRule='start'
```

The results are:

DateTime	TagName	Value	Quality	QD
2015-08-17 13:00:00.0000000	TotalProduced.Counter	4460	0	212
2015-08-17 14:00:00.0000000	TotalProduced.Counter	4544	0	64
2015-08-17 15:00:00.0000000	TotalProduced.Counter	4481	0	212

Scenario 3: Track liquid and gas use with custody meters

You can use the Counter retrieval feature to keep track of how much liquid or gas a process uses. For example, you could answer questions like:

- How much syrup was dispensed from Tank 3?
- How much propane have we used so far this month?

Using Counter Retrieval with Floating Point Counters

Counter retrieval also works with floating point counters, which is useful for flow meter data. Similar to the carton counter, some flow meters report accumulated flow and "roll over" after a certain amount of flow accumulates. For both examples, the need is to convert the accumulating measure to a "delta change" value over a given period. (For an example of totalizing a flow rate, see *Scenario 5* (see "*Scenario 5: Calculate the total flow*" on page 15).)

Example: Tracking Liquid or Gas Volumes with Counter Retrieval

```
SELECT DateTime, TagName, Value, Quality, QualityDetail AS QD
   FROM History
      WHERE TagName like 'ReceivedVolume.Counter'
      AND DateTime >= '2015-09-01 13:00'
      AND DateTime < '2015-09-03 18:00'
      AND wwResolution=3600000
      AND wwRetrievalMode='counter'
      AND wwQualityRule='optimistic'
      AND wwTimeStampRule='start'
```

The results are:

DateTime	TagName	Value	Quality	QD
2015-09-01 13:00:00.0000000	ReceivedVolume.Counter	8965.23345496515	0	212
2015-09-01 14:00:00.0000000	ReceivedVolume.Counter	8977.65306903776	0	212
2015-09-02 15:00:00.0000000	ReceivedVolume.Counter	8959.81258651845	0	212
2015-09-02 16:00:00.0000000	ReceivedVolume.Counter	8990.74717176479	0	212
2015-09-03 17:00:00.0000000	ReceivedVolume.Counter	8992.89247964001	0	212

Scenario 4: Analyze cycle time for your process

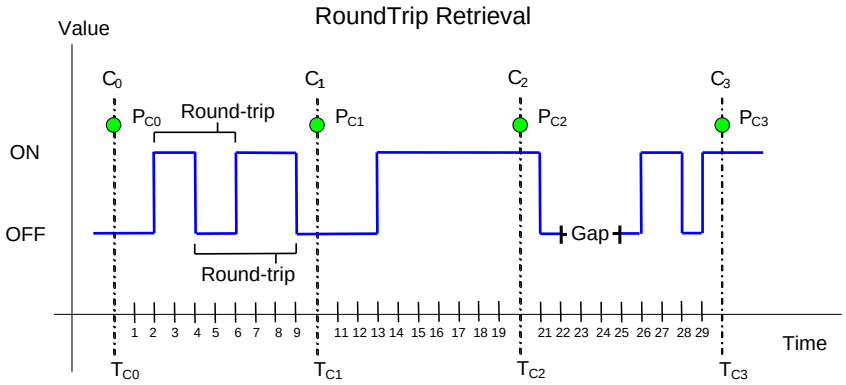
You can use the Roundtrip retrieval feature to analyze cycle time. For example, you could answer questions like:

- How long does it take between starting to fill one bag and starting to fill the next one?
- How long is the period between downtimes (or, the Mean Time Between Failures, MTBF)?
- How long between when a pump starts and the next time it starts?

What is RoundTrip Retrieval?

With RoundTrip retrieval, you can determine how long it take for the total cycle of a process to complete, through each "state" in the process. For example, a switch or a valve has two states -- on and off -- and therefore can be tracked using a discrete tag.

Here's an illustration of RoundTrip retrieval for a discrete tag that cycles through on and off states.



RoundTrip calculates the time a state begins until the time the same state begins again. For a discrete tag that simply tracks on and off, RoundTrip mode returns two rows per cycle. For example, when querying a discrete tag, RoundTrip can calculate the time between when a motor starts until the next time it starts, and when the motor stops until the next time it stops.

Example: Analyzing Cycle Time with RoundTrip Retrieval

To use the RoundTrip retrieval mode, set the following parameter in your query:

```
wwRetrievalMode = 'RoundTrip'
```

This query uses RoundTrip retrieval:

```
SELECT DateTime, TagName, vValue, StateTime
   FROM History
      WHERE TagName like 'Motor1.State'
      AND DateTime >= '2015-09-17 0:00'
      AND DateTime < '2015-09-17 2:00'
      AND wwResolution=3600000
      AND wwRetrievalMode='RoundTrip'
```

```
AND wwStateCalc='MinContained'
AND wwTimeStampRule='start'
```

The results are:

DateTime	TagName	vValue	StateTime
2015-09-17 00:00:00.0000000	Motor1.State	1	22001
2015-09-17 00:00:00.0000000	Motor1.State	0	21981
2015-09-17 01:00:00.0000000	Motor1.State	1	22001
2015-09-17 01:00:00.0000000	Motor1.State	0	24000

In these results, the value "0" shows when the valve was off and "1" when it was on.

The resulting rows show the average amount of time for each state and have a timestamp of the query end time (the default).

Scenario 5: Calculate the total flow

You can use the Integral retrieval feature to convert a rate into a quantity, such as totalizing a flow. For example, you can answer questions such as:

- How much vanilla extract have we used in the last month to produce ice cream?
- How much orange soda did we bottle yesterday?

What is Integral Retrieval?

Integral retrieval calculates the values at retrieval cycle boundaries by integrating the graph described by the points stored for the tag. It works much like average retrieval, but also applies a scaling factor.

Integral retrieval mode is ideal for calculating volume for a particular tag. For example, if a data tag represents product flow in liters per second, you can use integral retrieval to retrieve the total product flow in liters during a given time period.

Integral retrieval returns one row for each tag in the query for each cycle. You can specify cycles using resolution or cycle count.

Integral retrieval works only with analog tags.

Using Integral Instead of Counter

Integral retrieval and counter retrieval both deal in rates and quantities, but in different ways:

- **Counter retrieval** converts a total quantity measurement into a rate.

  For example, counter retrieval can take liters and a rate of 60000 to determine a rate in liters per minute.

- **Integral retrieval** converts a rate measurement into a quantity.

  For example, a rate in liters per minute can be translated into total liters for a given period.

Example: Calculating the Total Flow with Integral Retrieval

To use the integral retrieval mode, set the following parameter in your query:

```
wwRetrievalMode = 'Integral'
```

In this example, the value is computed for each of five one-minute cycles. The wwQualityRule parameter ensures that only points with good quality are used in the computation; data points with doubtful quality are discarded.

```
SELECT DateTime, TagName, Value, OPCQuality, PercentGood
   FROM History
       WHERE TagName like 'Flow1.PV'
       AND DateTime >= '2015-09-24 12:00'
       AND DateTime < '2015-09-25 8:00'
       AND wwRetrievalMode = 'Integral'
       AND wwCycleCount = 6
       AND wwQualityRule = 'Good'
```

The results of this query are:

DateTime	TagName	Value	OPCQuality	PercentGood
2015-09-24 12:00:00.000	Flow1.PV	15820.7228325108	192	100.0
2015-09-24 16:00:00.000	Flow1.PV	7776.62767927986	192	100.0
2015-09-24 20:00:00.000	Flow1.PV	14535.3472187875	192	79.0
2015-09-25 00:00:00.000	Flow1.PV	17058.0302965352	192	100.0
2015-09-25 04:00:00.000	Flow1.PV	18737.2427425968	192	100.0

The Value column shows the total flow for each 4-hour cycle. To calculate the total flow for the entire period, add the flow values together.

Scenario 6: Calculate down time statistics

You can use the RoundTrip retrieval feature to calculate periodic downtime, including:

- What is the mean time between failures for my operation?
- What is the mean time to repair (or replace) for a certain component?

What are Mean Time Between Failures and Mean Time to Repair?

Both mean time between failures (MTBF) and mean time to repair or replace (MTTR) view the total cycle of a process.

- MTBF describes how long a process will likely run before the system goes down. To calculate this number, divide the total time processing by the number of breakdowns.
- MTTR describes how long it will probably take for a particular part to need repair or replacement. (For example, a pump would be repaired, but a paper filter would be replaced.) To calculate this number, divide the total downtime by the number of breakdowns.

Tracking States with RoundTrip Retrieval

Some parts in your process have just two states -- on and off -- while others have more. For example, an engine might have four states -- switch on, warm up, run, switch off. RoundTrip retrieval works with integer analog tags, discrete tags, and string tags to track a variety of states. You can use this type of retrieval with History and StateWideHistory tables.

RoundTrip retrieval performs calculations on state occurrences in the within a specified cycle period. RoundTrip retrieval uses the time between consecutive leading edges of the same state for its calculations.

The RoundTrip mode returns a row for each state in any given cycle. For example, if your engine has four states in a cycle, RoundTrip retrieval will return four rows per cycle.

Points on the boundary of the end cycle are considered part of the next cycle and are not counted in the calculation.

Example: Tracking MTBF with RoundTrip Retrieval

If the tag used is "1" when the equipment is down, this returns the MTBF over the period queried:

```
SELECT DateTime, TagName, StateTime, OPCQuality, PercentGood
   FROM History
   WHERE TagName like 'PLCSim.Boolean6'
      AND DateTime >= '2016-03-15 06:00'
      AND DateTime < '2016-03-15 08:00'
      AND wwRetrievalMode = 'RoundTrip'
      AND wwTimeStampRule='start'
      AND wwStateCalc='avgcontained'
      AND wwCycleCount = 1
      AND Value=1
```

The results of this query are:

DateTime	TagName	StateTime	OPCQuality	PercentGood
2016-03-15 06:00:00.0000000	PLCSim.Boolean6	12201.5602716469	192	100

Example: Tracking MTTR with RoundTrip Retrieval

This example shows how to track MTTR.

```
SELECT DateTime, TagName, StateTime, OPCQuality, PercentGood
    FROM History
    WHERE TagName like 'PLCSim.Boolean6'
        AND DateTime >= '2016-03-15 06:00'
        AND DateTime < '2016-03-15 08:00'AND wwRetrievalMode = 'ValueState'
        AND wwTimeStampRule='start'
        AND wwStateCalc='avgcontained'
        AND wwCycleCount = 1
        AND Value=1
```

The results of this query are:

DateTime	TagName	StateTime	OPCQuality	PercentGood
2016-03-15 06:00:00.0000000	PLCSim.Boolean6	6100.29661016949	192	100

Scenario 7: Pinpoint reasons for downtime

You can use State Summary information to help you answer questions like:

- How much downtime was due to feeder jams?
- What else is causing downtime?

What is State Summary Data?

State summary data summarizes the states of a tag value. You can use this to analyze process variables with a limited number of states, such as a machine's state of running/starting/stopping/off or a string that represents a downtime reason.

For each distinct state within a cycle, state summary replication also provides:

- Total time
- Percent of the cycle
- Shortest time
- Longest time
- Average time

Using the StateSummaryHistory View

You can use the StateSummaryHistory view to retrieve state summary data.

A state summary results in a series of values, each representing a different state, for the same tag and time period. You configure the maximum states when you create the state summary tag. For more information, see State Summary Replication in the *AVEVA Historian Administration Guide.*

Example: Querying to Pinpoint Downtime

Suppose you know ahead of time what the likely causes for downtime might be in your plant. You could create a string or integer tag that identifies those causes. Then you can use a query like this one to track actual reasons for downtime in your facility:

```
SELECT DateTime, TagName, vValue, StateTime, OPCQuality AS 'OPC', PercentGood
AS '%Good'
   FROM History
      WHERE TagName like 'Downtime.Status'
      AND DateTime >= '2015-10-07 0:00'
      AND DateTime < '2015-10-08 0:00'
      AND wwRetrievalMode = 'ValueState'
      AND wwTimeStampRule='start'
      AND wwStateCalc='percentcontained'
      AND wwCycleCount = 1
      ORDER BY StateTime DESC
```

The results of this query are:

DateTime	TagName	vValue	StateTime	OPC	%Good
2015-10-07 00:00:00.000	Downtime.Status	Running	55.173125	0	0

DateTime	TagName	vValue	StateTime	OPC	%Good
2015-10-07 00:00:00.000	Downtime.Status	Idle	15.4923715277778	0	0
2015-10-07 00:00:00.000	Downtime.Status	Material Jam	9.52366666666667	0	0
2015-10-07 00:00:00.000	Downtime.Status	Electrical	6.60920138888889	0	0
2015-10-07 00:00:00.000	Downtime.Status	Backed Up	5.18525925925926	0	0
2015-10-07 00:00:00.000	Downtime.Status	Equipment Jam	4.21300347222222	0	0
2015-10-07 00:00:00.000	Downtime.Status	Starved	3.33917083333333	0	0
2015-10-07 00:00:00.000	Downtime.Status	NULL	0.446497685185185	0	0

Example: Finding the Minimum Time in a State

The following query finds the shortest closed/open time for the SteamValve discrete tag.

```
SELECT DateTime, TagName, vValue, StateTime, wwStateCalc
   FROM History
      WHERE TagName IN ('SteamValve')
      AND DateTime > '2015-10-04 0:00:00'
      AND DateTime <= '2015-10-08 0:00:00'
      AND wwCycleCount = 1
      AND wwRetrievalMode = 'ValueState'
      AND wwStateCalc = 'MinContained'
```

The results of this query are:

DateTime	TagName	vValue	StateTime	wwStateCalc
2015-10-08 00:00:00.000	SteamValve	0	20688	MinContained
2015-10-08 00:00:00.000	SteamValve	1	19105	MinContained

Example: Finding the Percentage of Time in a State

The following query returns the percentage of time in state for a discrete tag for multiple retrieval cycles. The TimeStampRule system parameter is set to "End" (the default setting), so the returned values are timestamped at the end of the cycle. Note that the first row returned represents the results for the period starting after 2015-10-05 0:00 and ending at 2015-10-08 0:00.

The "Percent" time-in-state retrieval mode is the only mode in which the StateTime column does not return time data. Instead, it returns percentage data (in the range of 0 to 100 percent) representing the percentage of time in state.

```
SELECT DateTime, TagName, vValue, StateTime, wwStateCalc
   FROM History
      WHERE TagName IN ('PLCSim.Boolean6')
      AND DateTime >= '2016-03-15 06:00'
      AND DateTime < '2016-03-15 08:00'
```

```
AND Value = 1
AND wwRetrievalMode = 'ValueState'
AND wwStateCalc = 'Percent'
AND wwCycleCount = 6
```

The results of this query are:

DateTime	TagName	vValue	StateTime	wwStateCalc
2016-03-15 06:00:00.0000000	PLCSim.Boolean6	1	50.18475	Percent
2016-03-15 06:20:00.0000000	PLCSim.Boolean6	1	49.8515	Percent
2016-03-15 06:40:00.0000000	PLCSim.Boolean6	1	49.9315833333333	Percent
2016-03-15 07:00:00.0000000	PLCSim.Boolean6	1	50.1493333333333	Percent
2016-03-15 07:20:00.0000000	PLCSim.Boolean6	1	49.9735	Percent
2016-03-15 07:40:00.0000000	PLCSim.Boolean6	1	49.8709166666667	Percent

Scenario 8: Fill in data gaps

In a perfect world, instruments and data acquisition always capture measurements reliably, but the real world is full of interruptions. In some environments, these interruptions are so routine that they interfere with operational reporting. You can use the optimistic retrieval feature to help fill data gaps. For example, you could answer:

- If there is no data available at midnight when my report runs, how can find the most recent value before that?
- How can I fill in the gaps in my trend caused by unreliable communications?
- How do I make a "best guess" for information lost because of a communication failure between my equipment and the application server?

What is Optimistic Retrieval?

In addition to capturing data values, AVEVA Historian records the data quality (OPC quality) of a value. AVEVA Historian also marks any disconnects with NULL values and an associated QualityDetail of 10 and 24. By default, these appear in trends as gaps and in reports as NULL values .

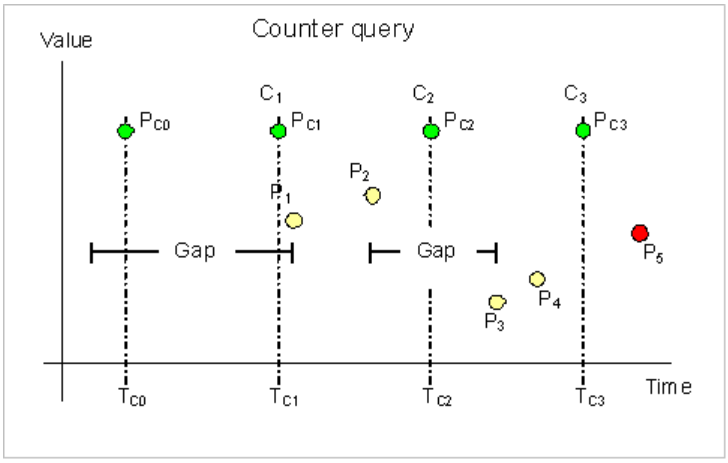
Details about data gaps and less-than-optimal data quality are important for diagnosing communications problems and for supervisory control. However, there may be gaps in the good quality data, making it hard to create the reports you need to have. For example, it would be difficult to create a report with a total for yesterday's production or the raw material consumption at midnight if there were communications problems.

Using optimistic retrieval allows you to make a best guess without tampering with the underlying data.

Using the Optimistic quality rule lets you retrieve information that is possibly incomplete but likely to help fill in some gaps in the good quality data. This setting calculates using the last known good value before the gap (if possible). The logic for determining the percent good, value count, and OPC quality remains unchanged.

How Optimistic Retrieval Works

The following illustration shows a Counter retrieval situation where three of the four shown cycle boundaries are located in data gaps. Without using the Optimistic quality rule, Counter queries return NULL at all cycle boundaries because the mode needs valid good values at each end of the cycle calculate a precise difference.



If the query specifies Optimistic, the results include a PercentGood column indicating whether all the data was "good" (100) or included some combination of "bad" and "uncertain" data (anything less than 100).

For the illustration above, there is no previous data and no available data in the first cycle, so it is skipped. At the second cycle boundary, the value 0 is returned, because there was a gap in the data for the entire first cycle. In the second cycle, there are two points, P1 and P2. The query uses P2 as the end value of the cycle and infers a start value of the cycle from P1. At the third cycle boundary, Tc2, the query returns P2 – P1. Similarly, at the last cycle boundary, the query returns P4 – P3.

Example: Masking Gaps in Your Data

At times, the results of a query might return null values, as in the following example:

```
SELECT TagName, DateTime, Value, PercentGood, QualityDetail
   FROM History
      WHERE TagName = 'BatchPercentConc'
      AND DateTime >= '2015-09-16 08:30'
      AND DateTime <= '2015-09-17 08:30'
      AND wwRetrievalMode = 'Delta'
```

The results are:

TagName	DateTime	Value	PercentGood	QualityDetail
BatchPercentConc	2015-09-16 08:30:00	(null)	0	65536
BatchPercentConc	2015-09-16 14:54:33	50	100	44
BatchPercentConc	2015-09-16 15:25:39	(null)	0	24
BatchPercentConc	2015-09-17 07:55:59	50	100	252
BatchPercentConc	2015-09-17 08:01:25	30	100	192
BatchPercentConc	2015-09-17 08:15:50	(null)	0	24
BatchPercentConc	2015-09-17 08:16:05	50	100	252

You can improve the usefulness of your data by adding this statement:

```
wwQualityRule = 'Optimistic'
```

For example:

```
SELECT TagName, DateTime, Value, QualityDetail
   FROM History
      WHERE TagName = 'BatchPercentConc'
      AND DateTime >= '2015-09-16 08:30'
      AND DateTime <= '2015-09-17 08:30'
      AND wwRetrievalMode = 'Delta'
      AND wwQualityRule = 'Optimistic'
```

The results are:

TagName	DateTime	Value	QualityDetail
BatchPercentConc	2015-09-16 08:30:00	(null)	65536
BatchPercentConc	2015-09-16 14:54:33	50	44

BatchPercentConc	2015-09-17 07:55:59	50	8444
BatchPercentConc	2015-09-17 08:01:25	30	192
BatchPercentConc	2015-09-17 08:16:05	50	8444

Scenario 9: Understand and use data

Data that is not 100% reliable can still be of value. Using data quality parameters, you can find out why data quality was marked uncertain, and when you find it useful, add data marked "uncertain" to your results.

You can use data quality indicators to answer questions like:

- What data did we lose because of a device failure?
- What was the volume readout before the power outage?

Data Quality in AVEVA Historian

When AVEVA Historian receives data from a data source, it adds a timestamp and quality indicator to the data value. Then, it stores the combined value, timestamp, and quality indicator (VTQ) as a data tag. The quality indicator has two parts, exposed by two separate columns:

- **OPCQuality:** What was the quality of this data as reported by the source?
- **QualityDetail:** What special processing has AVEVA Historian performed on the data?

OPC Data Quality: The Good, the Bad, and the Uncertain

OPC Data Access is an industry standard for real-time data exchange between hardware and software components. This standard defines a consistent way of exchanging live values by appending a data value with a timestamp and a quality value. In AVEVA Historian, this quality is exposed in the OPCQuality column. OPC defines quality as a 16-bit integer. The most significant bits for Historian are bits 6 and 7. They indicate whether the data's overall quality is "good", "bad" or "uncertain".

Historian stores and retrieves the timestamps and qualities for "good", "bad", and "uncertain" data. Historian always converts values with "bad" quality to NULL to protect users and applications from unwittingly propagating these values known to be bad.

The OPC standard allows status of "bad" to include a substatus. For example, a substatus of "5" means "Bad, Last Known Value" and "3" means "Bad, Device Failure". In a strict "current value only" protocol (like OPC DA), the best you can do is provide the quality and that previous value. However, AVEVA Historian will also store that previous value (with its quality and timestamp).

Note: Although in many cases OPCQuality and QualityDetail use the same codes to indicate similar concepts (for example, "192" indicates normal "good" values), they will often be different and should not be used interchangeably.

QualityDetail: More to the Story

QualityDetail tells what happened with data while it was processed by various AVEVA Historian subsystems. For example, it can indicate that some data was stored as the first value after the connection to the historian was restored after a temporary disconnect, or it can tell that the retrieved value is not the actual value generated by an I/O Server or DAServer, but a manually corrected one.

Quality detail codes are metadata used to describe data values stored or retrieved using the AVEVA Historian. Quality details are 2 byte values that use the low order byte for the basic quality detail and the high order byte to store quality detail flags. The basic quality detail and quality detail flags are OR'd together.

Viewing Quality Values and Details

Descriptive labels for the many OPC quality codes are stored in the OPCQualityMap table of the Runtime database. To view OPCQuality values and descriptions, execute the following query:

```
SELECT OPCQuality, Description FROM OPCQualityMap
```

Similarly, descriptive labels for the many quality detail codes are stored in the QualityMap table of the Runtime database. To view the complete list of QualityDetail values and descriptions, execute the following query:

```
SELECT QualityDetail, QualityString FROM QualityMap
```

Note: Although a quality detail of 65536 is used to indicate block gaps, NULL values are not produced for block gaps.

Using WWQualityRule

AVEVA Historian provides three different options for interpreting the stored OPCQuality:

- **Good:** Filters out "uncertain" values, as if they did not exist, and converts "bad" values to NULL.
- **Extended**: Includes "uncertain" values, but takes them out of the "PercentGood" calculations and makes the OPCQuality "uncertain" for calculated results that include included them. Extended also converts any "bad" valuesto NULL.
- **Optimistic**: Filters out "bad", but is otherwise similar to "extended".

Examples: Using OPCQuality to Refine Results

Suppose you wanted a report to tell you the average production level for a certain storage tank for each 8-hour shift. You could run this query:

```
select DateTime, Value, OPCQuality, PercentGood
   from History
      where TagName like 'Flow273.PV'
      and DateTime > '2016-03-02 0:00'
      and DateTime <= '2016-03-02 3:00'
      and wwRetrievalMode = 'integral'
      and wwResolution = 60*60*1000
      and wwInterpolationType = 'stairstep'
      --and wwQualityRule = 'extended'
```

Note that the last line is commented out because "QualityRule = 'extended'" is the default.

The results are:

DateTime	Value	OPCQuality	PercentGood
2016-03-02 01:00:00.0000000	255932.995605469	192	100
2016-03-02 02:00:00.0000000	263511.000823975	64	100
2016-03-02 03:00:00.0000000	190458.002471924	64	75

To force a query to exclude points with uncertain OPC quality, set wwQualityRule to Good. For example, you can change the previous example to this:

```
select DateTime, Value, OPCQuality, PercentGood
   from History
      where TagName like 'Flow273.PV'
      and DateTime > '2016-03-02 0:00'
```

```
and DateTime <= '2016-03-02 3:00'
and wwRetrievalMode = 'integral'
and wwResolution = 60*60*1000
and wwInterpolationType = 'stairstep'
and wwQualityRule = 'good'
```

The results are:

DateTime	Value	OPCQuality	PercentGood
2016-03-02 01:00:00.0000000	255932.995605469	192	100
2016-03-02 02:00:00.0000000	261971.994781494	192	100
2016-03-02 03:00:00.0000000	190458.002471924	64	75

A third option is to change the quality rule to Optimistic:

```
select DateTime, Value, OPCQuality, PercentGood
   from History
      where TagName like 'Flow273.PV'
      and DateTime > '2016-03-02 0:00'
      and DateTime <= '2016-03-02 3:00'
      and wwRetrievalMode = 'integral'
      and wwResolution = 60*60*1000
      and wwInterpolationType = 'stairstep'
      and wwQualityRule = 'optimistic'
```

The results are:

DateTime	Value	OPCQuality	PercentGood
2016-03-02 01:00:00.0000000	255932.995605469	192	100
2016-03-02 02:00:00.0000000	263511.000823975	64	100
2016-03-02 03:00:00.0000000	255474.000549316	192	75

Finally, queries to AnalogSummaryHistory use the Optimistic quality rule exclusively . For example:

```
select StartDateTime, Value, OPCQuality, PercentGood
   from AnalogSummaryHistory
      where TagName like 'Flow273.PV'
      and DateTime > '2016-03-02 0:00'
      and DateTime <= '2016-03-02 3:00'
      and wwRetrievalMode = 'integral'
      and wwResolution = 60*60*1000
      and wwInterpolationType = 'stairstep'
      and wwQualityRule = 'optimistic'
```

The results are:

StartDateTime	Value	OPCQuality	PercentGood
2016-03-02 01:00:00.0000000	263511.000823975	64	100
2016-03-02 02:00:00.0000000	255474.000549316	192	100

Scenario 10: Get summary statistics when a certain condition applies

You can use SliceBy to answer questions like:

- How much energy was consumed per batch?
- What was the average flow rate when a valve was open?
- What was lowest pressure today while a pump was running?

Using the SliceBy Parameter

SliceBy lets you define batches to be used with an Analog Summary query to get averages by batch. When you also use SliceByValue, you can get summary information per batch when a certain condition exists.

Example: Finding Total Energy Consumed per Batch

This query finds the total energy consumed per batch.

```
select SliceBy, SliceByValue, TagName, MIN(StartDateTime) as PeriodStart,
   MAX(EndDateTime) as PeriodEnd, SUM(wwResolution)/1000 as
[Duration(Sec)],
   MIN(Minimum) as [Min], MAX(Maximum) as [Max], SUM(Integral) as [Total]
from AnalogSummaryHistory
   where TagName='PowerMeter'
     and SliceBy='R31.BatchNum'
     and StartDateTime>='2019-04-17 0:00'
     and EndDateTime<='2019-04-17 2:00'
group by TagName, SliceBy, SliceByValue
```

The results are:

SliceBy	SliceBy Value	TagName	PeriodStart	PeriodEnd	Duration(Sec)	Min	Max	Total
R31.BatchNum	844	PowerMeter	4/17/2019 0:02:42	4/17/2019 0:11:57	554	8.4944	15.1857	610.6690
R31.BatchNum	845	PowerMeter	4/17/2019 0:11:57	4/17/2019 0:25:30	812	6.7169	14.8926	1267.8577
R31.BatchNum	846	PowerMeter	4/17/2019 0:25:30	4/17/2019 0:34:45	554	5.3918	17.2337	431.7148

Example: Finding Average Flow Rate When Valve is Open

This query retrieves statistics on the flow rate, but only while the valve is open. The query finds an average of weighted averages.

First, the query defines a batch by instance that the valve is open. Then, it calculates the total volume of the flow for that batch and calculates a weighted average by duration. Finally, it calculates an "average of averages" for the batch.

```
select SliceBy, SliceByValue, TagName, MIN(StartDateTime) as PeriodStart,
   MAX(EndDateTime) as PeriodEnd, SUM(wwResolution)/1000 as
[Duration(Sec)], MAX(Maximum) as [Max],
   SUM(Integral) as [Total],
AVG=SUM(Average*wwResolution)/SUM(wwResolution)
from AnalogSummaryHistory
   where TagName='WaterFlow'
     and SliceBy='R31_PLC.WaterValve'
     and SliceByValue = 1 -- Valve Open
     and StartDateTime>='2019-04-17 0:00'
     and EndDateTime<='2019-04-17 2:00'
group by TagName, SliceBy, SliceByValue
```

The results are:

SliceBy	SliceBy Value	TagName	PeriodStart	PeriodEnd	Duration (Sec)	Max	Total	Avg
R31_PLC.WaterValve	1	WaterFlow	4/17/2019 0:02:45	4/17/2019 1:54:42	1199	1000	614995.85	512.51234

Note: This example correctly calculates the overall average for each state in the "Average" column by weighting the duration of each state. As explained by Simpson's Paradox, the simpler, "AvgOfAvg" calculation is not statistically accurate and can differ significantly with some data sets.

Example: Finding the Lowest Pressure Today While a Pump Was Running

This example finds today's lowest pressure reading for a pump while it was running.

```
declare @date datetime
set @date = convert(date, getdate())

select SliceBy, SliceByValue, TagName, MIN(StartDateTime) as PeriodStart,
   MAX(EndDateTime) as PeriodEnd, SUM(wwResolution)/1000 as
[Duration(Sec)], MIN(Minimum) [Min],
   MAX(Maximum) as [Max]
from AnalogSummaryHistory
where TagName='LinePressure'
and SliceBy='StorageTank_001.OutletValve'
and SliceByValue = 1 -- Valve Open
and StartDateTime>= @date
group by TagName, SliceBy, SliceByValue
```

Where results might be:

SliceBy	SliceBy Value	TagName	PeriodStart	PeriodEnd	Duration (Sec)	Min	Max
StorageTank_001.OutletValve	1	LinePressure	4/19/2019 0:25:10	4/19/2019 8:05:07	3094	9.0150	15.4113