

Information and specifications published here are current as of the date of publication of this document. Tridium, Inc. reserves the right to change or modify specifications without prior notice. The latest product specifications can be found by contacting our corporate headquarters, Richmond, Virginia. Products or features contained herein may be covered by one or more U.S. or foreign patents. © 2007 Tridium, Inc.

Sample Reports Using BQL and Bound Tables

Customized report graphics are easily created using both a standard Px page or the newer report Px page which was introduced in build 3.2.16. Using the report service and the report pane it is possible to produce more robust interfaces that are easily exported. The examples presented in this document can be created in older builds with standard Px pages.

Creating The Report Px Page

If using the 3.2.x build, there is a new file type which may be added to the station called ReportPxFile.px. This is the preferred default template to use when creating a report graphic. When a standard Px file is created it contains a scroll pane at the root of the widget tree with a canvas pane inside of the scroll pane. The default report Px file only includes a report pane at the root of the widget tree. There is no functional difference between the two pages other than the default panes which are present when the file is created. If using any previous builds older than 3.2.x, then the report pane component is not available.

Report Pane Versus Canvas Pane

The report pane provides several benefits over using a standard Px page with scroll and canvas panes.

- There are no settings for the report pane size. The report pane is the root of the Px file and auto sizes based on the content to display.
- The report pane has properties which allow the inclusion of a logo image, the page number and a date time stamp on each page of an exported report.
- Bound tables in a report pane auto size to display all rows without the need of a scroll bar, whereas a bound table in a standard canvas pane must be sized manually.
- When exporting the report Px to a pdf file the required number of pages automatically generate.

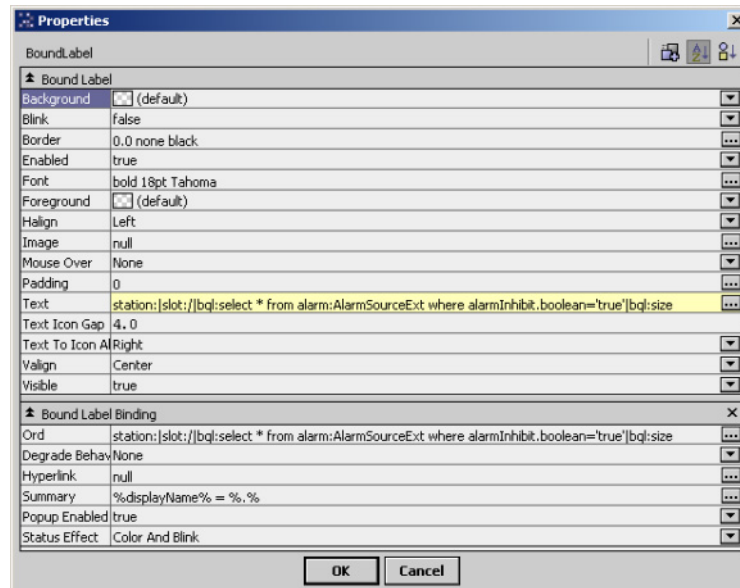
Note: *It is important not to put the report pane inside of a scroll pane or the auto sizing and page generation will not function correctly.*

Using Bound Tables and Bound Labels

You can create a report Px page using bound labels and bound tables to display real time data. Bound tables can also be used to display historical data on the reports as well.

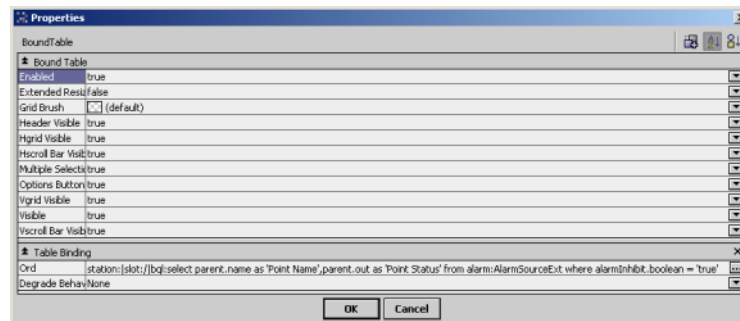
A bql query can be entered in the ord field of the bound label binding (see [1-1](#)), as opposed to referencing a component in the station. Animating the text field allows displaying the results of the bql query. If the bformat text is configured to '%.%' then it will simply display the value. Additional standard text may be used as well such as, 'Points With Alarms Disabled (Alarm Inhibit = True): %.%'.

Figure 1-1 Bound label property sheet using bql query in the Ord field



The bql query is entered in the ord field of the bound table to display the list of points or historical data (see 1-2).

Figure 1-2 Bound table property sheet using bql query in Ord field



Example Reports

The following sections provide examples of how to create specific types of reports. You can find complete versions of these examples on “www.Niagara-Central.com” under the KnowledgeBase area. The report examples may be downloaded in both the Px and PDF file formats.

- [Point Status Report](#)
- [Schedule Report](#)
- [Tenant Override Report](#)
- [Weekly Electrical Demand Report](#)
- [Point Status Report](#)

Point Status Report

Operators often desire a snap shot report which displays the status of points. In particular emphasis is placed on points which are in an abnormal state such as overridden, alarm, fault, disabled or which have alarm functionality disabled.

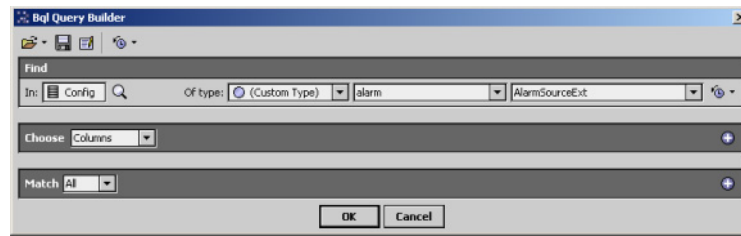
- **Displaying Points with Alarm Functionality Disabled**

Alarm extensions have a boolean property called 'Alarm Inhibit'. When the property value is 'true', processing of alarm events is prevented. The alarm inhibit status is not displayed in the status of the parent point because it is not actually a property of the BStatus type.

Using bql the station can be searched for components which are alarm source extensions. The bql query builder (see 1-3) can be used to create the query or you can simply type the syntax, as shown below.

```
station:|slot:|/|bql:select * from alarm:AlarmSourceExt
```

Figure 1-3 Bql query builder constructing query for alarm source extensions

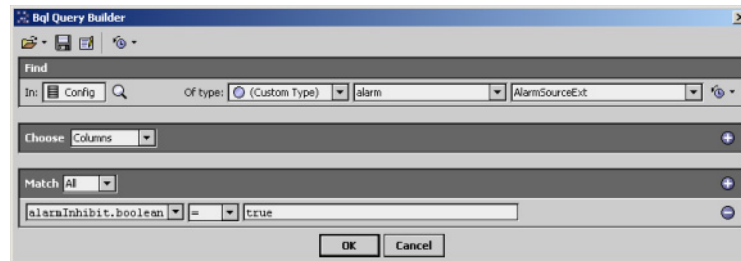


- **Filtering by Alarm Inhibit Property**

This query returns a list of all alarm source extensions in the station regardless of whether the extension is inhibited or not. To limit the return based on the alarm inhibit property, the query needs to be modified as below (see 1-4).

```
station:|slot:/|bql:select * from alarm:AlarmSourceExt where alarmInhibit.boolean = 'true'
```

Figure 1-4 Bql query builder constructing query filtered by alarm inhibit property

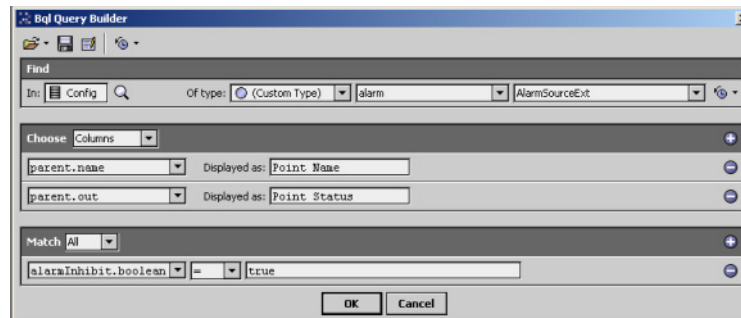


- **Displaying Specific Columns**

The filtered query returns a list of alarm source extensions whose alarm inhibit property is set to true, but the table displays all of the properties for the extension. It is preferable to further filter the results so that only the desired properties are displayed. The columns which are displayed in the table may be limited by further modifying the query as below (see 1-5).

```
station:|slot:/|bql:select parent.name as 'Point Name',parent.out as 'Point Status' from alarm:AlarmSourceExt where alarmInhibit.boolean = 'true'
```

Figure 1-5 Bql query builder constructing query to filter the displayed columns



Note: In the example above, bformat text is used to display information from the parent component of the alarm source extension. The name or displayName of the parent point and the out slot would likely be useful information to display. Other columns could be added to display information from the alarm source extension as well.

- **Displaying the Number of Records in a Query**

It is also possible to use a bql query to calculate and display the number of records returned in the query. The original query can be modified as below.

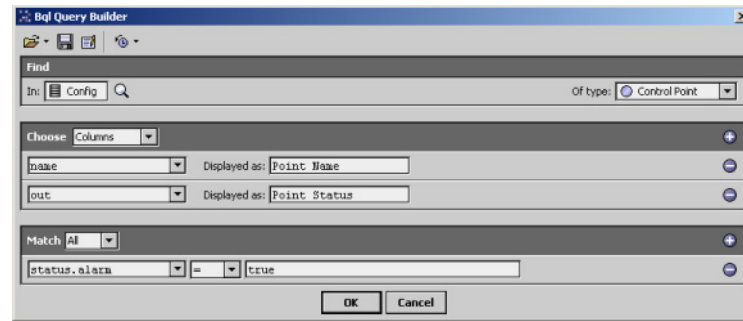
```
station:|slot:/|bql:select * from alarm:AlarmSourceExt where alarmInhibit.boolean = 'true'|bql:size
```

Note: The bql query builder does not support the 'bql:size' syntax, although it is a valid query. It is necessary to delete this syntax from the query prior to opening the bql query builder.

- Displaying Points Currently In Alarm**
 Each control point in the station has both a value and a status. The status reflects the current condition or reliability of the point value. Available statuses are down, alarm, unacknowledged alarm, overridden, disabled, fault and stale.
 Using bql the station can be searched for control points where the alarm status flag is true. The bql query builder (see 1-6) can be used to create the query or you can simply type the syntax below.

```
station:|slot:|/bql:select name as 'Point Name',out as 'Point Status' from
control:ControlPoint where status.alarm = 'true'
```

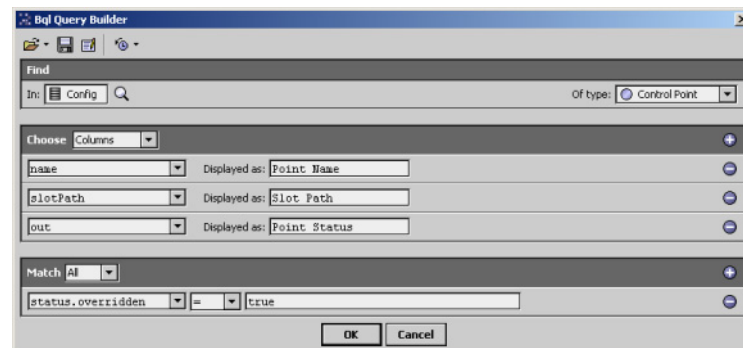
Figure 1-6 Bql query builder constructing query for control points in alarm



- Displaying Points Currently Overridden**
 Using bql the station can be searched for control points where the overridden status flag is true. The bql query builder (see 1-7) can be used to create the query or you can simply type the syntax below.

```
station:|slot:|/bql:select name as 'Point Name',out as 'Point Status' from
control:ControlPoint where status.overridden = 'true'
```

Figure 1-7 Bql query builder constructing query for control points currently overridden



Schedule Report

Operators often request a print out of all scheduled events including the normal weekly schedules and special events. A schedule report can be generated for all of the schedules in a given station by using a bql query.

- Examining a Typical Schedule**
 By default the composite schedule component of a standard schedule is hidden. Accessing the slot sheet of a schedule component allows removing the hidden flag from the composite schedule. Viewing the composite schedule property sheet (see 1-8) helps to understand how the composite schedule is organized and how to construct the bql query.

Figure 1-8 Composite schedule property sheet

The screenshot shows a hierarchical property sheet for a 'Composite Schedule'. The main 'Schedule' object has two properties: 'Always Effective' (set to false) and 'Union' (set to true). It contains two child objects: 'specialEvents' and 'week'. 'specialEvents' is itself a 'Composite Schedule' with 'Always Effective' set to false and 'Union' set to true. 'week' is a 'Week Schedule' that lists the days of the week (Sunday through Saturday), each associated with a 'Daily Schedule' component.

The main composite schedule consists of a child composite schedule called specialEvents and a week schedule called week. Special events which are added to the schedule and each specific day of the week schedule are all daily schedule type objects.

Using bql the station can be searched for the daily schedule components. The bql query builder (see 1-9) can be used to create the query or you can simply type the syntax below.

```
station:|slot:/|bql:select * from schedule:DailySchedule
```

Figure 1-9 Bql query builder creating query for daily schedule components

The screenshot shows the 'Bql Query Builder' dialog box. In the 'Find' section, 'In:' is set to 'Config', 'Of type:' is set to '(Custom Type)', and 'schedule' is selected. The 'Choose' section has 'Columns' selected. The 'Match' section has 'All' selected. 'OK' and 'Cancel' buttons are at the bottom.

This basic query for the daily schedule component returns the slot path, property values and names of the sub schedule components. This is not very useful information for an operator, so the query will likely need to be modified to display more specific pertinent information. The property sheet of the daily schedule can be used to get a better idea of what data might be useful to display in the columns (see 1-10).

Figure 1-10 DailySchedule property sheet with children components expanded

Each daily schedule consists of two child components. The first component is called 'day' which is a day schedule and the second component is called 'days' which is an abstract schedule. The abstract schedule defines the event date and the day schedule defines the event times and event value.

```
Weekly Schedule (Boolean Schedule) %parent.parent.parent.name%
|-- Schedule (Composite Schedule) %parent.parent.name%
|   |-- Special Events (Composite Schedule) %parent.name%
|   |   |-- Board Meeting (Daily Schedule)
|   |       |-- Day (Day Schedule)
|   |           |-- time (Time Schedule)
|   |           |-- Days (Abstract Schedule)
|   |-- week (week schedule)
|       |-- Sunday (Daily Schedule)
|           |-- Day (Day Schedule)
|               |-- time (Time Schedule)
|               |-- Days (Abstract Schedule)
```

- **Resolving the Schedule Name**

Since the bql query is for the daily schedule components, it is necessary to use BFormat text to derive the schedule name using the below syntax. The actual schedule component is three levels up the navigation tree from the daily schedule components.

```
%parent.parent.parent.displayName%
```

- **Displaying the Event Date**

The days component returns the actual event date if specified or if it is a weekly schedule then it displays 'weekday schedule'.

- **Displaying the Event Name**

Since the query returns the daily schedules, use 'displayName'.

- **Displaying the Event Times**

The event times are slots of the time schedule which is a child of the daily schedule components.

There are individual slots for the start and finish times, which may be displayed using the below syntax.

```
day.time.start (displays the time of the first start event)
day.time.finish (displays the time of the first stop event)
day.time1.start (displays the time of the second start event)
day.time1.finsih (displays the time of the second stop event)
```

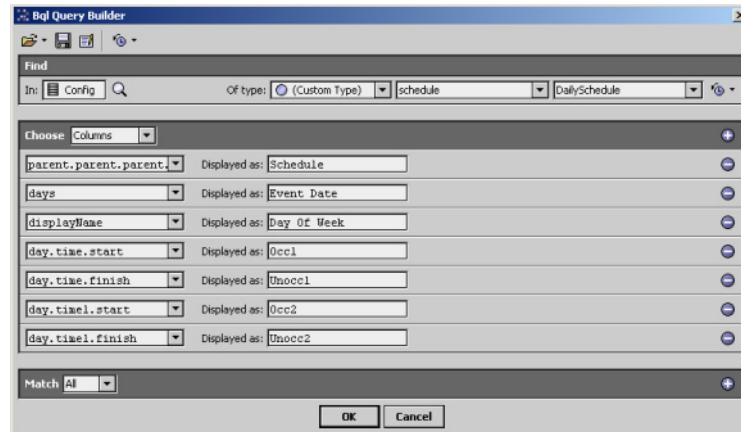
- **Filtered Query for Daily Schedules**

Using the above concepts a filtered query can be created to display the desired results. The bql query builder (see 1-11) can be used to create the query or you can simply type the syntax below.

```
station:|slot:|bql:select parent.parent.parent.displayName as
'Schedule',days as 'Event Date',displayName as 'Day Of Week',day.time.start as
'Occ1',day.time.finish as 'Unoccl',day.time1.start as 'Occ2',day.time1.finish
as 'Unocc2' from schedule:DailySchedule
```

Each daily schedule consists of two child components. The first component is called 'day' which is a day schedule and the second component is called 'days' which is an abstract schedule. The abstract schedule defines the event date and the day schedule defines the event times and event value.

Figure 1-11 Bql query builder creating query for daily schedule components

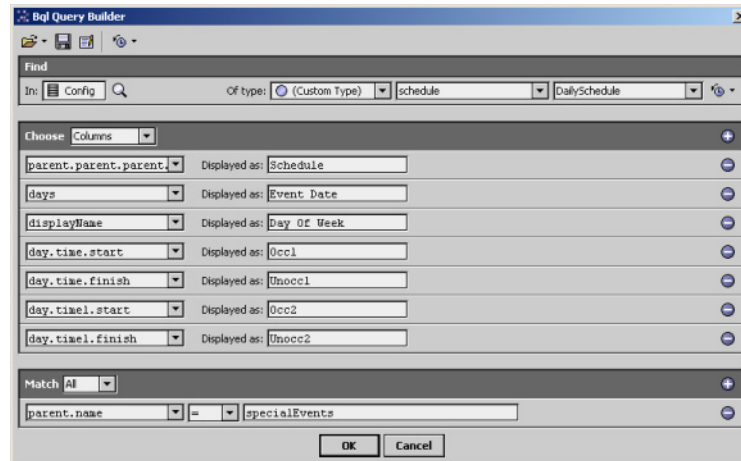


- **Filtering by Special Events**

It may also be helpful to filter the results to only display special events, after all people typically know what the standard schedule events are. The bql query builder (see 1-12) can be used to modify the query or simply type the syntax below.

```
station:/slot:/|bql:select parent.parent.parent.displayName as
'Schedule',days as 'Event Date',displayName as 'Day Of Week',day.time.start as
'Occ1',day.time.finish as 'Unocc1',day.time1.start as 'Occ2',day.time1.finish
as 'Unocc2' from schedule:DailySchedule where parent.name = 'specialEvents'
```

Figure 1-12 Figure 12: bql query builder creating query for special events



Tenant Override Report

It is often times necessary to generate reports on historical data, which can be equally as important as reporting on real time data. Histories in the station may be configured to record information at specified intervals, a change of value or state, or only when certain other conditions exist. There are also standard logs in the station like the audit log and log history which record operator actions and system errors.

If a tenant has access to override set points or scheduled periods of operation, then it may be a requirement to document these actions. The standard audit log can be used to track the actions, or additional histories could be configured to track the actions as well.

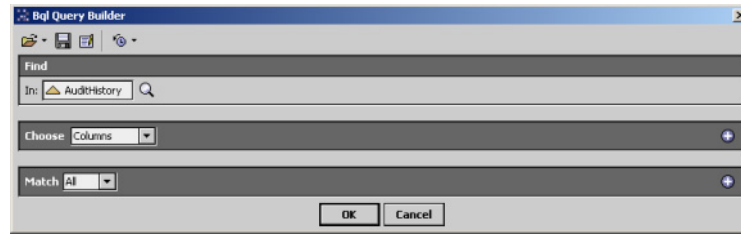
- **Using the Audit Log**

The audit log can be searched for records where a specific set point or component has been overridden. The audit log record displays the timestamp, operation (added, removed, invoked, changed, etc), target (path to component), slot name, old value (occupied, unoccupied, 78 deg F, etc), value (the new value), and the user who performed the operation.

The bql query builder (see 1-13) can be used to create the query or you can simply type the syntax below.

```
history:/demo/AuditHistory|bql:select *
```

Figure 1-13 Bql query builder creating query for the audit log

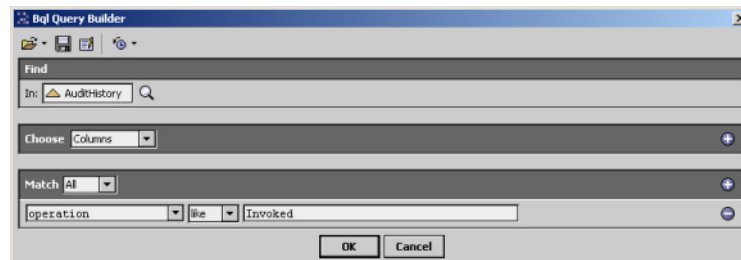


- **Filtering by Operation**

This basic query returns a list of every record in the audit log and every available data column. Since the objective is to only display records relating to an overridden point, the query can be modified as below (see 1-14) to limit the return based on the operation

```
history:/demo/AuditHistory|bql:select * where operation like 'Invoked'
```

Figure 1-14 Figure 14: bql query builder creating query for the audit log filtered by operation

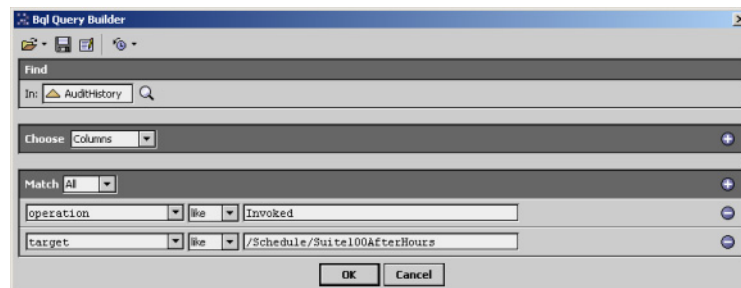


- **Filtering by Target Component**

The modified query only returns records where the operation is 'invoked'; however, the return is for every component in the station. The query can be further modified as below to limit the return based on the (target) component. (see 1-15)

```
history:/demo/AuditHistory|bql:select * where operation like 'Invoked' and target like '/Schedule/Suite100AfterHours'
```

Figure 1-15 Bql query builder creating query for the audit log filtered by operation and target

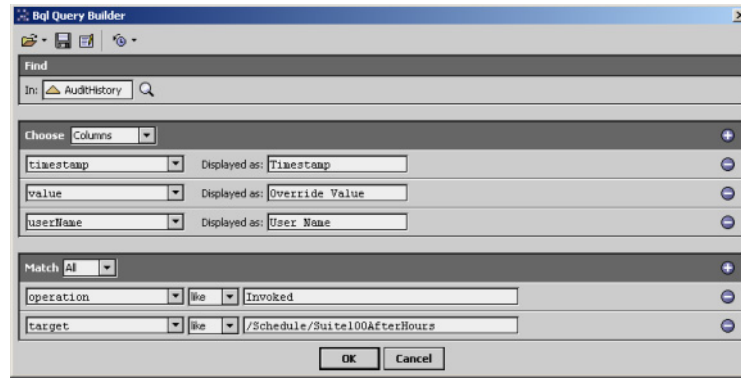


- **Configuring the Displayed Columns**

The return could be cleaned up by further modifying the query as below (see 1-16) to limit the displayed columns.

```
history:/demo/AuditHistory|bql:select timestamp as 'Timestamp',value as 'Override Value',userName as 'User Name' where operation like 'Invoked' and target like '/Schedule/Suite100AfterHours'
```


Figure 1-16 Creating query for the audit log filtered by operation, target and columns

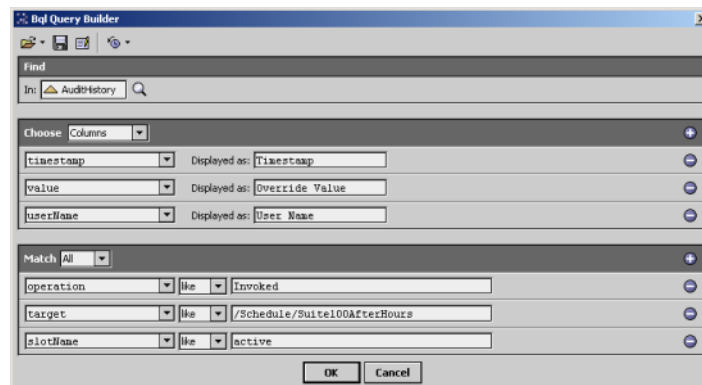


- **Filtering by Specific Actions**

The return is now limited to actions which have been invoked on a specific component in the station by any operator. It may be necessary to further filter the results to a single operator or a specific action. The bql query can be modified as below (see 1-17) to limit the return to the specific operation of operator override to the active state.

```
history:/demo/AuditHistory|bql:select timestamp as 'Timestamp',value as 'Override Value',userName as 'User Name' where operation like 'Invoked' and target like '/Schedule/Suite100AfterHours' and slotName like 'active'
```

Figure 1-17 Creating query for the audit log filtered by operation, target, columns and slot



- **Filtering by Timestamps**

The bql query now produces a fairly presentable table which displays the timestamp, value and operator for each record. The return contains records from any time period in the audit log. Depending on how many records are maintained in the audit log, this could be a fairly large list. It is likely that the query will need to be modified as below to limit the return based on desired date ranges.

```
history:/demo/AuditHistory?period=lastMonth|bql:select timestamp as 'Timestamp',value as 'Override Value',userName as 'User Name' where operation like 'Invoked' and target like '/Schedule/Suite100AfterHours'
```

Note: The bql query builder does not support the '?period=' syntax, although it is a valid query. It is necessary to delete this syntax from the query prior to opening the bql query builder.

Weekly Electrical Demand Report

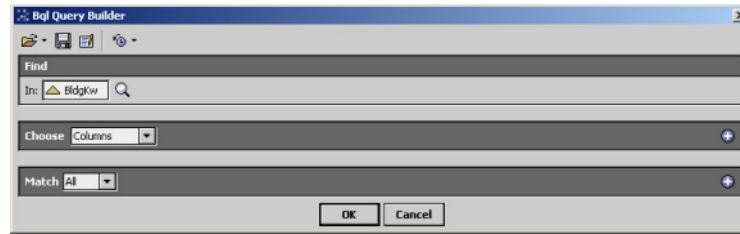
Bound tables and charts can be used in a report Px page to display historical information. An example might be to create a report which displays electrical demand readings for the previous week.

- **Creating the History Query**

Any history in the station can be queried using bql. The bql query builder (see 1-18) can be used to create the query or you can simply type the syntax below.

```
history:/demo/BldgKw|bql:select *
```

Figure 1-18 Creating query for the BldgKw history



- **Limiting the Query**

The above bql query will return a table which contains all of the records available in the referenced history. If the export source object is executed once a week, then it would probably be safe to limit the query to the previous weeks worth of data. The period syntax can be used to limit the query as below.

```
history:/demo/BldgKw?period=lastWeek|bql:select *
```

This will still return a significant amount of data. Assuming that the data is recorded in fifteen minute intervals, this would return 672 records and the pdf file would span seventeen pages. Bql includes a special function called history rollup. This function can be applied to the previous query to display the same results using fewer records. Each record in the rollup indicates the number of samples, minimum value, maximum value, average value and a sum of all samples. The below example uses the history rollup function with a daily interval which results in only seven records (one for each day of the week) displayed in the table.

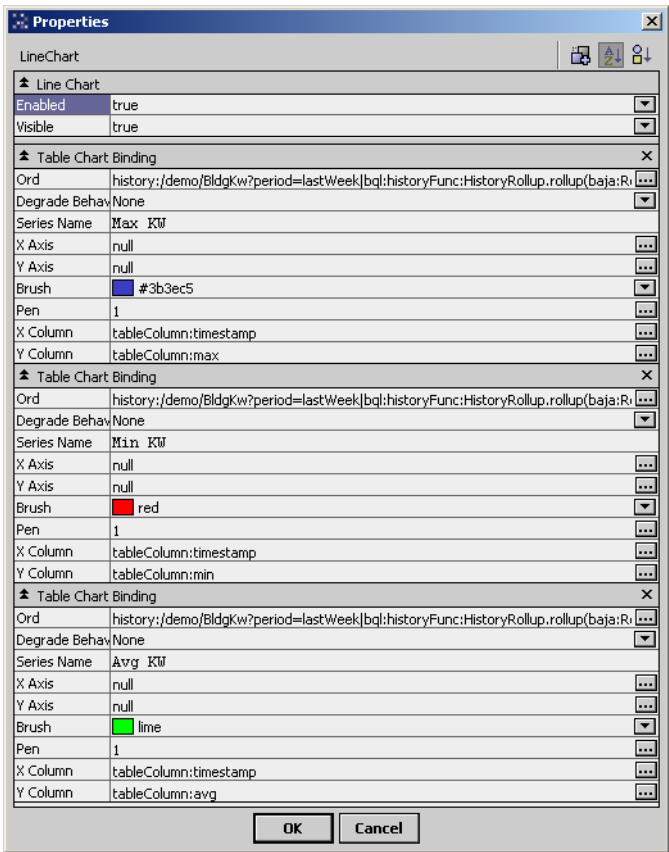
```
history:/demo/BldgKw?period=lastWeek|bql:historyFunc:History-  
Rollup.rollup(baja:RelTime '86400000')
```

Note: The history rollup function does not allow specifying which columns are to be displayed. By default all columns are displayed in the table.

- **Using a Chart**

In some cases it may be a requirement to display the data in chart form as opposed to using a table. The chart palette includes two types of chart widgets, the line and bar chart. After adding the line chart widget to the Px file, table chart bindings may be added and configured. The previous query can be used in the ord field of the table chart binding. Each table chart binding represents one trace on the chart and must specify which column of the query to display on the Y axis. Multiple table chart bindings can reference the same query but display different columns such as min, max, avg or sum.

Figure 1-19 Creating table chart queries for the BldgKw history



Note: The current export mechanism does support using history chart tables or history chart views directly in the report Px file. Both the history table and chart views allow exporting directly from the specific view.

Document change log

Updates (changes/additions) to this *Engineering Note: Sample Reports Using SQL and Bound Tables* document are listed below.

- Publication: December 10, 2007
Initial publication.

