How to Create a Query

DEPRECATED as at v2.0.

Introduction

Open-AudIT comes with many queries inbuilt. Only a few are activated by default, but you are able to active more by going to menu -> Admin -> Queries -> Activate Query.

If you require a specific query and none of the pre-packaged queries fit your needs, it's quite easy to create a new one and load it into Open-AudIT for running.

Some background first will help you understand how it all hangs together....

Query definitions are stored as XML files which can then be "activated". By activating a query you import the XML into the database which will also make the query name appear in the menu when you view a Group. Anyone can run a query. Only Admin level users can activate or deactivate a query.

Definition Sections

The XML definition file has two main sections - "details" and "columns".

The first relates to extracting the information from the database and the other meta data about the query (it's name, etc).

The second section defines how we wish to view the query data on the screen.

For this wiki page I will use the "Workstation Hardware" query as an example is it's simple, but contains some interesting items.

First Section - Details

An example of the first section is below.

```
    <details>
        <report_name>Workstation Hardware</report_name>
        <report_display_in_menu>y</report_display_in_menu>
        <report_sql><![CDATA[SELECT system.man_icon, system.man_os_family, system.system_id, system.hostname,
                            system.man_ip_address, system.man_manufacturer, system.man_model, system.
man_serial,
                            floor(system.pc_memory / 1024) AS pc_memory, system.man_form_factor,
sys_hw_processor.processor_description
                            FROM system LEFT JOIN oa_group_sys ON system.system_id = oa_group_sys.system_id
LEFT JOIN sys_hw_processor
                            ON (sys_hw_processor.system_id = system.system_id AND sys_hw_processor.timestamp =
system.timestamp) LEFT JOIN
                            oa_location ON (system.man_location_id = oa_location.location_id) WHERE
oa_group_sys.group_id = @group AND
                            man_type = 'computer' AND man_class != 'server' GROUP BY system.hostname ORDER BY
system.hostname]]></report_sql>
        <report_view_file>v_report</report_view_file>
        <report_view_contents></report_view_contents>
        <report_processing></report_processing>
        <report_sort_column>3</report_sort_column>
        <report_display_sql></report_display_sql>
        <report_description>Device details - name, ip, manufacturer, model, serial, form factor, memory,
processor.</report_description>
    </details>
```

Working through this, most fields are self explanatory.

report_name is the name for the query as it appears on the screen and in the menu.

report_display_in_menu determines if the report should be shown in the menu. Certain queries are not designed to be displayed in the menu. Specific Software is one example. You need to provide a software_id to this report. It is designed to be called from the Installed Software report.

report_sql is the SQL used to query the database. More on this below.

report_view_file is normally set to v_report. Unless your view file needs to contain something special in terms of extra processing PHP code, just leave this blank.

report_view_contents is not used. This was originally designed to contain the view file contents (see report_view_file). Set the report_view_file instead of using this.

report_processing is not used. This was originally designed to contain extra PHP processing code.

report_display_sql is not used.

report_sort_column is the initial sort column id (starting at 0).

report_description is shown on the List Queries page (menu -> admin -> List Queries).

Second Section - Columns

An example of the second section of the query definition XML is below.

```
<columns>
    <column>
        <column_order>1</column_order>
        <column_name>Icon</column_name>
        <column_variable>man_icon</column_variable>
        <column_type>image</column_type>
        <column_link></column_link>
        <column_secondary>man_os_family</column_secondary>
        <column_ternary></column_ternary>
        <column_align>center</column_align>
    </column>
    <column>
        <column_order>2</column_order>
        <column_name>System Name</column_name>
        <column_variable>hostname</column_variable>
        <column_type>link</column_type>
        <column_link>/main/system_display/</column_link>
        <column_secondary>system_id</column_secondary>
        <column_ternary></column_ternary>
        <column_align>left</column_align>
    </column>
    <column>
        <column_order>3</column_order>
        <column_name>IP Address</column_name>
        <column_variable>man_ip_address</column_variable>
        <column_type>ip_address</column_type>
        <column_link></column_link>
        <column_secondary></column_secondary>
        <column_ternary></column_ternary>
        <column_align>left</column_align>
    </column>
</columns>
```

The column attributes are again reasonably self evident.

- column_id is the numeric order of the column on the displayed query page.
- column_name is the header value for the column.
- column_variable is the name of the variable returned from the SQL statement above.
- column_type can be set to 'text', 'link', 'image', 'ip_address', 'multi', 'url', 'timestamp'. Formatting for each type is below:
    - text - simply show the output.
    - timestamp - currently outputs as per 'text' type.
    - url - and external link is created that points to the value of column_link. Will be shown using an icon instead of the actual text.
    - multi - is not used.
    - ip_address - format the output to de-pad the ip address. IP Addresses are stored in the database padded, so an ip address thus 192.168.1.1 would be internally stored as 192.168.001.001.
    - image - If column_name == Icon, use an image in /var/www/open-audit/theme-tango/tango-images/16_*.png where * is the value from column_variable. If column_name == Picture, use an image in /var/www/open-audit/device_images/*.jpg where * is the value from column_variable.

    - link - use the column_link variable to create a hyperlink and append the column_secondary value. Typically used to create a link to a device.

- column_link is used to set the external link when column_type is 'url'.
- column_secondary is appended where required when column_type is 'link'.
- column_ternary is appended where required when column_type is 'link'.
- column_align is the alignment of the data in the displayed column.

Report SQL

So the main piece of code of interest is report_sql. This defines what data is extracted from the database.

When a device is queried, most of it's specific information is stored in the 'system' table. If data from another table (an attribute table, say hard_drive) is required a join must be created.

With the release of 1.10, the below has changed (with the exception of the sys_hw_network_card_ip table). See the release notes for 1.10 here - Release Notes for Open-AudIT v1.10

There is now only a single join required an attribute table to the system table. As well, most tables have been renamed. Using the old example below, we would select items like this:

```
processor.system_id = system.system_id and processor.current = "y"
```

~~Open-AudIT stores its data using two main keys. sys_hw_processor.system_id = system.system_id is pretty obvious. The second key should use the timestamps. This way you will return current attribute rows. If you omit this you will receive all attribute rows, whether they're current or not. So sys_hw_processor.timestamp = system.timestamp. FYI - Using the timestamps you can easily find out what's not current by specifying sys_hw_processor.timestamp != system.timestamp.~~

~~So your attribute joins should look like this:~~

```
LEFT JOIN sys_hw_processor ON (sys_hw_processor.system_id = system.system_id AND sys_hw_processor.timestamp = system.timestamp)
```

The storage of the data is also explained on this wiki page - https://community.opmantek.com/display/OA/Information+about+how+Open-AudIT+processes+and+stores+data.

The other join that should appear in a report_sql statement is the group join. In order to run a query on a particular group, the group id is passed for you by the front-end and you can reference it via @group. You need to create a join like thus:

```
LEFT JOIN oa_group_sys ON system.system_id = oa_group_sys.system_id
```

And in the WHERE section, specify

```
WHERE oa_group_sys.group_id = @group
```

Loading the Query

Once you have created your XML definition you can get it into Open-AudIT in one of two ways. You can save the file to /open-audit/code_igniter/application /controllers/reports/queryname.xml and then go to menu -> Admin -> Queries -> Activate Query. If all is correct, it should appear in the list and be able to be activated by clicking the 'tick' icon on the right side.

You can alternately copy the XML contents and insert it into the text box on the menu -> Admin -> Queries -> Import Query page.

Once you have added the query to Open-AudIT, it should appear in the menu under Queries when you view a Group.

You can upload, test the query and if it's not working as intended, delete it via menu -> Admin -> Queries -> List Queries. Make your changes and upload /activate the query again. Test, rinse, repeat 🙂

The web front end will also take care of exporting the query result to Excel, JSON, XML, CSV, HTML etc for you. Just look for the buttons on the right side of the menu bar (near the 'Search Attributes' box).

And don't forget - you can always post to the forums and ask for help!