



And therefore, I advise you, much like Abbie Hoffman, to “Steal This Report.”

I’ve posted the SWQL code below, as well. A few notes about the specifics of what this report is doing/expecting:

- It is largely based on the default “All Active Alerts” report. I do not mean to imply in any way that I was the author of that wonderful resource
- It pulls a Node custom property named “importance,” which is intended for sorting of the actual report
- In addition to the “Acknowledge Alert” link, the name of the object that triggered the alert is clickable as well

## The SWQL Query

```
SELECT DISTINCT
AlertActive.AlertActiveID, AlertObjects.AlertObjectID, AlertConfigurations.Name,
AlertConfigurations.Severity, AlertConfigurations.ObjectType,
AlertObjects.EntityUri, AlertObjects.EntityType, AlertObjects.EntityDetailsURL,
AlertObjects.RelatedNodeID, NodeCP.Importance,

'<A HREF="' + AlertObjects.EntityDetailsURL + '" target="_blank">' + AlertObjects.EntityCaption + '</a>'
AS Caption,
'<A HREF="/Orion/Netperfmon/AckAlert.aspx?AlertDefID=' + toString(AlertObjects.AlertObjectID) + '"
target="_blank">ClickToAck</a>' AS AcknowledgeIt,

ToLocal(AlertActive.TriggeredDateTime) AS TriggeredDateTime, AlertObjects.LastTriggeredDateTime,
AlertActive.TriggeredMessage AS Message,
AlertActive.AcknowledgedDateTime, AlertActive.Acknowledged AS Acknowledged,
AlertActive.AcknowledgedBy, AlertActive.AcknowledgedNote,
Case
    When Floor((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) + 0.0)/86400) > 0 Then
        ToString(ToString(Floor((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate())
+ 0.0)/86400)) + 'd ' +
        ToString(Floor(((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) -
86400 * (Floor((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) + 0.0)/86400))) +
0.0)/3600)) + 'h ' +
        ToString(Floor(((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) -
3600 * (Floor((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) + 0.0)/3600))) + 0.0)/60)) + 'm
')
    When Floor(((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) -
86400 * (Floor((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) + 0.0)/86400))) +
0.0)/3600) > 0 Then
        ToString(ToString(ToString(Floor(((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) -
86400 * (Floor((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) + 0.0)/86400))) +
0.0)/3600)) + 'h ' +
        ToString(Floor(((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) -
3600 * (Floor((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) + 0.0)/3600))) + 0.0)/60)) + 'm
')
    When Floor(((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) -
3600 * (Floor((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) + 0.0)/3600))) + 0.0)/60) > 0
Then
        ToString(ToString(ToString(ToString(Floor(((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) -
3600 * (Floor((SecondDiff(AlertActive.TriggeredDateTime, GetUtcDate()) + 0.0)/3600))) + 0.0)/60)) + 'm
')
        Else ''
End AS ActiveTime
FROM Orion.AlertObjects (nolock=true) AlertObjects
INNER JOIN Orion.AlertActive (nolock=true) AlertActive ON
AlertObjects.AlertObjectID=AlertActive.AlertObjectID
INNER JOIN Orion.AlertConfigurations (nolock=true) AlertConfigurations ON
AlertConfigurations.AlertID=AlertObjects.AlertID
INNER JOIN Orion.Nodes (nolock=true) Nodes ON AlertObjects.RelatedNodeID = Nodes.NodeID
INNER JOIN Orion.NodesCustomProperties (nolock=true) NodeCP on AlertObjects.RelatedNodeID =
NodeCP.NodeID
Order By AlertConfigurations.Name, AlertObjects.EntityCaption
```

## Part 2: Making Web-Based Reports Do Your Bidding

I realized that the skills needed to address the customer request weren't particularly complex or specialized, but they relied on the monitoring engineer to be familiar with certain fundamental aspects of the Orion® Platform that not everyone has had a chance to explore. And so, in this chapter (and in the two that come after this), I'm going to dive into each one of those techniques. Just so you have a map, they are:

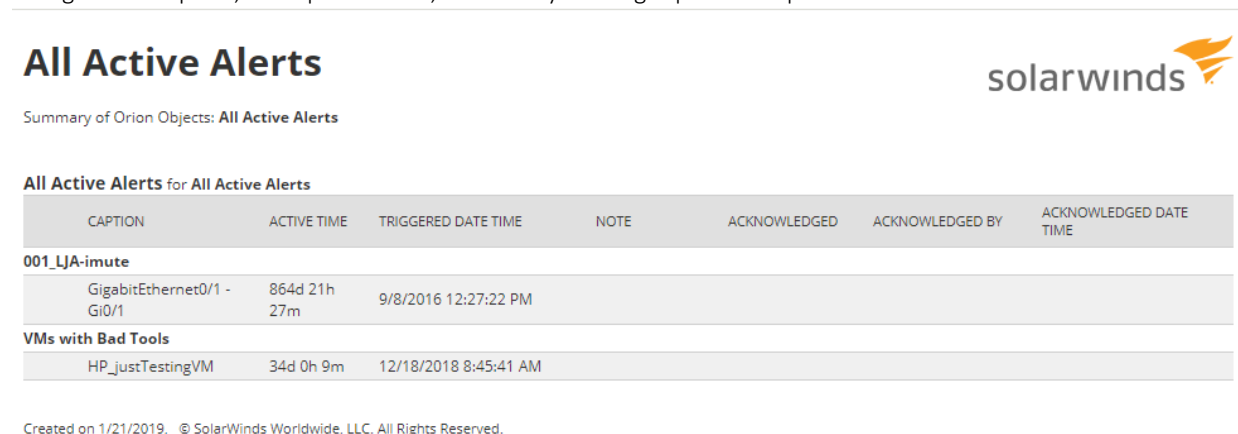
1. Making customizations to the data that feeds a web-based report (you are here)
2. Adding HTML links to the data returned in a report, so items are clickable
3. Creation an "action" column in a report that lets you perform a management task, such as acknowledging an alert

### The Magic Word is "SWQL"

When I speak of customizing reports, I'm not focusing on the cosmetic items like the title, colors, logos, etc. I want to *get at* the data. While there are a host of pre-existing views, tables, and Orion objects that you can pull into a report simply by selecting them (and I strongly recommend you have a look, via all the pre-built reports that you get upon installation of Orion and each SolarWinds module), the true power of the Orion Platform has always been the ability to pull information directly from the database and display it however you want. All it takes is a little know-how.

As I mentioned in the last post, I started with the "All Active Alerts" report, then customized it to my needs. I believe this is the best way to begin most projects because it usually gets you further ahead than you would if you'd started from scratch.

Going to the Reports, All Reports menu, I found my starting report and opened it to see what it looked like.



CAPTION	ACTIVE TIME	TRIGGERED DATE TIME	NOTE	ACKNOWLEDGED	ACKNOWLEDGED BY	ACKNOWLEDGED DATE TIME
<b>001_LJA-imute</b>						
GigabitEthernet0/1 - Gi0/1	864d 21h 27m	9/8/2016 12:27:22 PM				
<b>VMs with Bad Tools</b>						
HP_justTestingVM	34d 0h 9m	12/18/2018 8:45:41 AM				

Taking this as my starting point, I knew there were a few things I wanted to do:

1. Add my custom property for node importance
2. Sort the report by the importance value
3. Get rid of the grouping

Of course, I don't want to directly edit this report because I may want to fall back to the original at some point, so instead of clicking "Edit Report," I go back to the report list and click the "Manage Reports" link in the upper-right corner. Then I find my report again, select it, and click "Duplicate and Edit."

REPORT MANAGER		SCHEDULE MANAGER				
GROUP BY:	Report Category	Report Title	Description	Schedule assigned	Type	Category
All (352)	01_Leons_Custom_Reports (2)	001-LJA-All Active Alerts			Web-based	Custom
Agent Management (2)	APM: Current Application and ...	All Active Alerts			Web-based	Custom
APM: Daily Application Availa...	APM: Exchange Reports (4)	All Configured Alerts			Web-based	Custom
APM: Historical Application CP...	APM: Historical Reports (3)	All Configured Alerts Inventory Report			Web-based	Custom
		Triggered Alerts - Last 30 Days	Display all Alerts over the Past 30 Days		ReportWriter	Events
		Triggered Alerts - Last 30 Days web-b...	Display all Alerts over the Past 30 Days		Web-based	Events
		Triggered and Reset Alerts - Last 30 ...	Displays all Triggered and Reset Alert...		ReportWriter	Events

Now we're ready to get to work. Of course, you should change the title of the report first thing, along with other essentials like the logo, footer, etc. But the meat-and-potatoes of the work is the tiny little "edit" button that rests next to the "For" drop-down.

## Edit Report

REPORT MANAGER | SCHEDULE MANAGER

CREATE NEW REPORT | EDIT REPORT | **DUPLICATE & EDIT** | VIEW REPORT | SCHEDULE REPORT | EXPORT/IMPORT | DELETE

Report width: 960 px Fit to window width

**Header**

Copy of All Active Alerts

Enter Subtitle...

Logo  BROWSE FOR LOGO

**Content** Page Layout

Layout columns: 1

100%

All Active Alerts EDIT TABLE Duplicate

For All Active Alerts Edit

Clicking this will bring up the "Add Content" window, and you'll be automatically dropped into the query editor. There are other options—static device selection and a dynamic query builder that strongly resembles the system used to create alerts. But we're going to focus on SWQL because that's one of the most useful (and often least-appreciated) skills a monitoring specialist can develop.

From within that window, you can include additional tables, add fields, and more. There's only one problem: there's no explorer, let alone a "test" option, so you have no way of knowing if any of your edits are going to work.

Or do you?

## The Entrance to the Chamber of SWQL Secrets

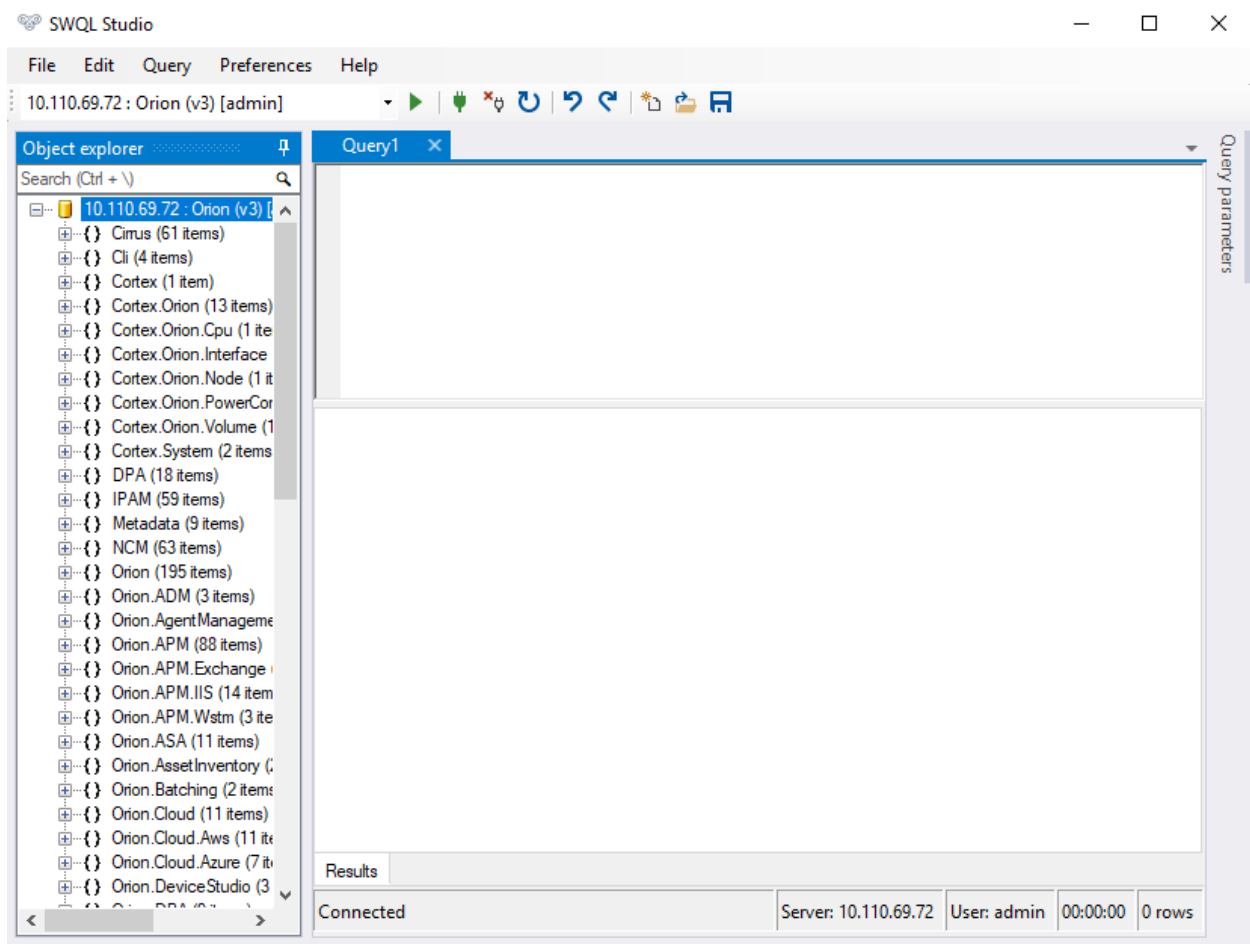
*...lies through SWQL Studio.*

Now, many will say that the dynamic query builder is the way to go, and I won't argue that it's a good start for many projects. But there are still limitations to the query builder, including the inability to pull in data from other tables, and that's precisely what we need here.

Without a web-based tool, where does a monitoring specialist turn? To the Orion Software Developer's Kit (SDK). This free set of tools can be downloaded from GitHub (<https://github.com/solarwinds/OrionSDK>) and has a large and

active forum on the THWACK® community (<https://thwack.solarwinds.com/community/resources/orion-sdk>) with plenty of how-to guides and tutorials to get you started.

Now if your first thought is that the SDK is meant for, well, “developers,” you wouldn’t be wrong. But the fact is that it’s a great tool for any SolarWinds aficionado to have in their toolkit because it includes the SWQL Studio, which lets you interactively (and safely) explore the entire Orion database and schema.



As long as you have a grasp of SQL query basics, you can use the SWQL Studio as a jumping-off point to create far more powerful alerts, reports, and resources.

In this example, we’re going to take the existing “All Alerts” query and add the “importance” custom property to the output.

Getting connected in SWQL Studio is just a matter of inputting the name or IP of your primary Orion polling engine along with the username and password you normally use to log in through the web portal.

Connect to Information Service

Server Name: MyOrionServer

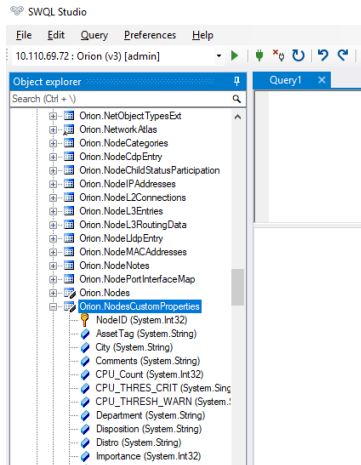
Server Type: Orion (v3)

User Name: username

Password: .....

Connect Cancel

Now the first thing I want to do is make sure I know where those darn custom properties are. If you've used SolarWinds tools for a few years, then you know they've moved from the main Nodes table into their own table.



I didn't use any magic tricks here. I just kept clicking and searching until I found something that said "custom properties" on it. To be certain it had the information I wanted (i.e., the "Importance" custom property"), I right-clicked the table and clicked "Generate Select Query," then the "Run" button to see the data.

SQL Studio Query1 window showing the following SQL query:

```
SELECT TOP 1000 NodeID, InstanceType, AssetTag, City, Comments, CPU_Count, CPU_THRES_CRIT, CPU_THRES_WARN, Department, Disposition, Distro, Importance, InServiceDate, n_mute, owner_email, polling_level, PONumber, PurchaseDate, PurchasePrice
FROM Orion.NodesCustomProperties
```

The query results are displayed in a table with the following columns: NodeID, Instance Type, Asset Tag, City, Comments, CPU\_Count, CPU\_THRES\_CRIT, CPU\_THRES\_WARN, Department, Disposition.

NodeID	Instance Type	Asset Tag	City	Comments	CPU_Count	CPU_THRES_CRIT	CPU_THRES_WARN	Department	Disposition
1	Orion.NodesCustomProperties	stomProperties	APAC	NULL	4	NULL	NULL	NULL	NULL
	stomProperties	NULL	APC	NULL	4	NULL	NULL	NULL	NULL
	stomProperties	NULL	Austin	NULL	2	NULL	NULL	NULL	NULL
	stomProperties	NULL	Austin	NULL	1	NULL	NULL	NULL	Decomm
	stomProperties	NULL	UK	NULL	2	NULL	NULL	NULL	NULL
	stomProperties	NULL	Chicago	NULL	2	NULL	NULL	NULL	NULL
	stomProperties	NULL	Chicago	NULL	4	NULL	NULL	NULL	NULL
13	Orion.NodesCustomProperties	NULL	Austin	NULL	4	NULL	NULL	NULL	NULL
15	Orion.NodesCustomProperties	NULL	APAC	NULL	4	NULL	NULL	NULL	NULL
20	Orion.NodesCustomProperties	NULL	Chicago	NULL	1	NULL	NULL	NULL	NULL
22	Orion.NodesCustomProperties	NULL	APAC	NULL	1	NULL	NULL	NULL	NULL
23	Orion.NodesCustomProperties	NULL	Chicago	NULL	4	NULL	NULL	NULL	NULL
24	Orion.NodesCustomProperties	NULL	UK	NULL	4	NULL	NULL	NULL	NULL
27	Orion.NodesCustomProperties	NULL	APAC	NULL	2	NULL	NULL	NULL	NULL
29	Orion.NodesCustomProperties	NULL	APAC	NULL	4	NULL	NULL	NULL	NULL

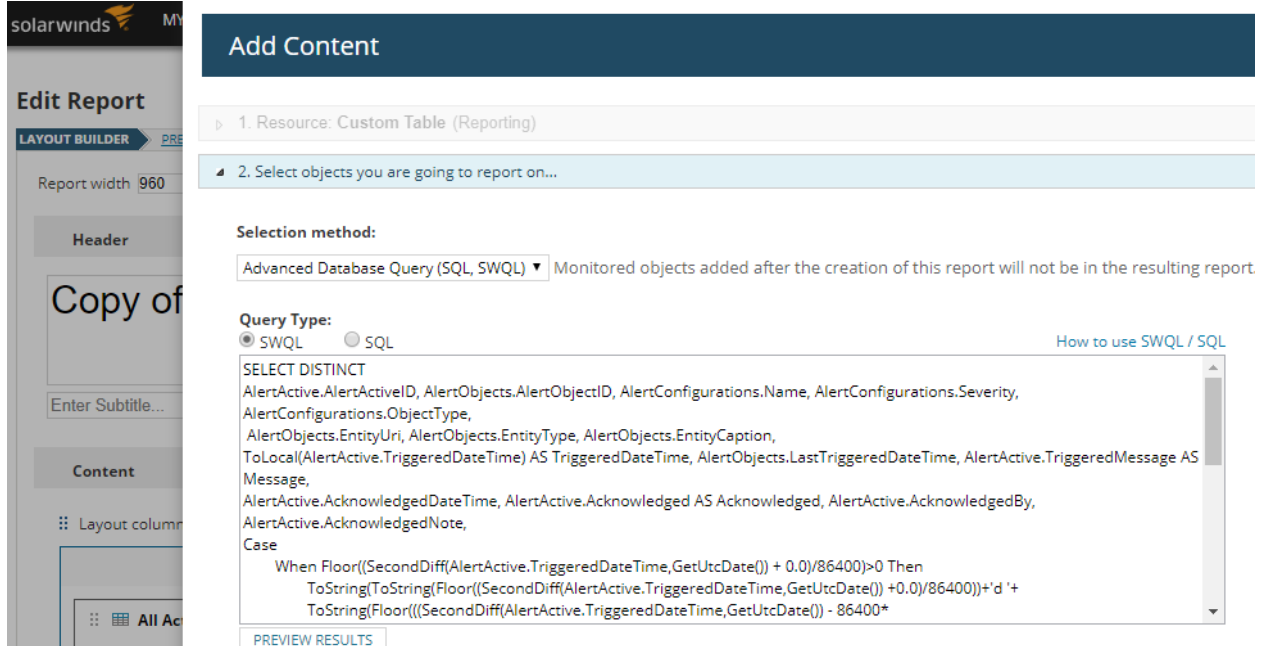
Results: 95 rows

Scrolling through the columns, I found the “Importance” column and verified the data, so I knew I had found what I needed.

Now... how to get *that* into the Alerts query?

## The SWQL Two-Step Tango

First things first, I needed to pull the query out of the existing All Alerts report. This didn’t prove difficult at all, as I just duplicated the existing report (because I’ve worked in IT for more than 15 minutes, so I know to make backup copies of everything), click the “Edit” link next to the data source drop-down, *et voila!* There’s my query.

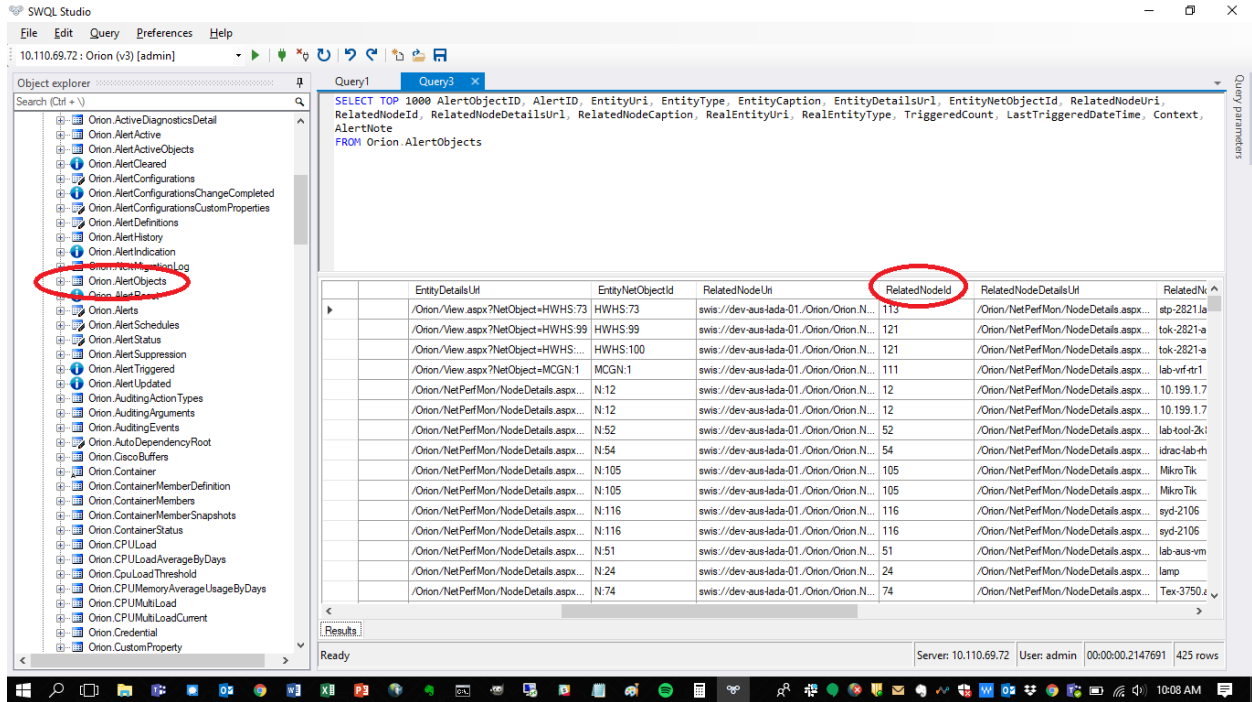


The screenshot shows the 'Add Content' dialog in SolarWinds Report Builder. It is set to 'Advanced Database Query (SQL, SWQL)' with the selection method 'Monitored objects added after the creation of this report will not be in the resulting report'. The 'Query Type' is set to 'SWQL'. The query text is as follows:

```
SELECT DISTINCT
AlertActive.AlertActiveID, AlertObjects.AlertObjectID, AlertConfigurations.Name, AlertConfigurations.Severity,
AlertConfigurations.ObjectType,
AlertObjects.EntityUri, AlertObjects.EntityType, AlertObjects.EntityCaption,
ToLocal(AlertActive.TriggeredDateTime) AS TriggeredDateTime, AlertObjects.LastTriggeredDateTime, AlertActive.TriggeredMessage AS
Message,
AlertActive.AcknowledgedDateTime, AlertActive.Acknowledged AS Acknowledged, AlertActive.AcknowledgedBy,
AlertActive.AcknowledgedNote,
Case
When Floor((SecondDiff(AlertActive.TriggeredDateTime,GetUtcDate())) + 0.0)/86400)>0 Then
ToString(ToString(Floor((SecondDiff(AlertActive.TriggeredDateTime,GetUtcDate())) +0.0)/86400))+ 'd ' +
ToString(Floor(((SecondDiff(AlertActive.TriggeredDateTime,GetUtcDate()) - 86400)*
```

I can cut and paste that into the SWQL Studio just to see how it runs.

Now, the next step really leans on your existing SQL knowledge. You should see that the Custom Properties table uses NodeID as its key field. The question is whether that data exists in any of the tables that the original All Alerts report uses. To figure that out, I found each of the tables in the All Alerts query and did that same “Generate Select Query” action to see what fields were there.



Wouldn't you know it, the Orion.AlertObjects table includes a "Related NodeID" field, which works perfectly for us. Now I can add the table source to the bottom of the All Alerts query with this line:

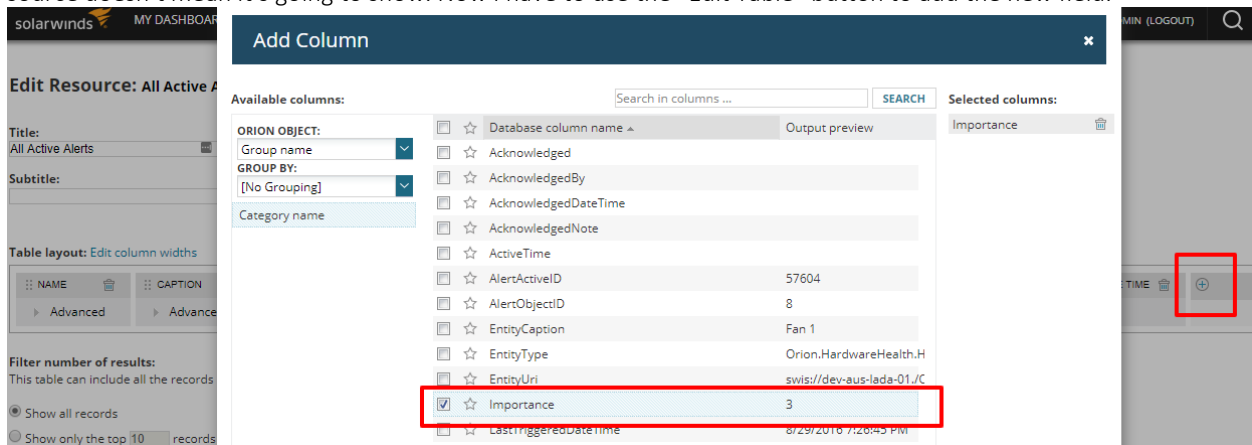
```
INNER JOIN Orion.NodesCustomProperties (nolock=true) NodeCP
ON AlertObjects.RelatedNodeID = NodeCP.NodeID
```

Then add the "Importance" field to the list of fields being collected:

```
SELECT DISTINCT
NodeCP.Importance, AlertActive.AlertActiveID,
AlertObjects.AlertObjectID, AlertConfigurations.Name,
```

Once I verify that everything works in SWQL Studio, I can copy and paste it back into the Customized All Alerts report in my Orion Portal.

But that's only half (OK, more like three-quarters) of the battle. Just because I'm pulling that field into the data source doesn't mean it's going to show. Now I have to use the "Edit Table" button to add the new field.





And move it to the first column position.

**Edit Resource: All Active Alerts for All Active Alerts** You can change this after submit.

Title:  
All Active Alerts

Subtitle:


Table layout: [Edit column widths](#)

IMPORTANCE	NAME	CAPTION	ACTIVE TIME	TRIGGERED DATE TIME
Advanced	Advanced	Advanced	Advanced	Advanced

Now I'm able to customize the report display by removing the grouping (that's just a personal preference of mine) and sort the results first by importance, then by alert name, and finally by node name.

When I run it, I have the report I wanted (at least so far).

## 001-LJA\_Importance\_All Active Alerts



Summary of Orion Objects: **All Active Alerts**

**All Active Alerts for All Active Alerts**  
 Ordered by: Importance - Ascending then by Name - Ascending then by Caption - Ascending

IMPORTANCE	NAME	CAPTION	ACTIVE TIME	TRIGGERED DATE TIME	NOTE	ACKNOWLEDGED	ACKNOWLEDGED BY	ACKNOWLEDGED DATE TIME
3	001_LJA-imute	GigabitEthernet0/1 - Gi0/1	864d 22h 54m	9/8/2016 12:27:22 PM				

## Part 3: Advanced Alert Reporting: Hyperlinks in Data Output

In the last chapter I covered the techniques you need to customize a report by adding new data sources (tables), and fields. Now it's time for me to show you how to add HTML code into your SWQL query so that the resulting data set includes clickable elements.

### What Are We Trying To Do?

While there are a lot of data points that could potentially use a clickable link in a report, I'm going to go with a fairly simple and obvious one for starters. Once you have the concept, you'll see how easily it can be applied in so many other situations. For this example, I'm going to make the Node Name clickable, so that clicking it takes you to the Node Details page for that device.

"You can be a millionaire and never pay taxes!  
 You say... "Steve, how can I be a millionaire and never pay taxes?"  
"First, get a million dollars..."  
 • from Steve Martin's "Cruel Shoes"  
 ([https://www.youtube.com/watch?v=zXmQW\\_aqBks](https://www.youtube.com/watch?v=zXmQW_aqBks))

As with my previous post, I'm going to be modifying the "All Alerts" report. Right now, the column called "caption" is actually the name of whatever element (interface, disk, node, application, etc.) that triggered the alert. So, the first part of this process, somewhat obviously, is to add the node name to the report.

As with the "Importance" field, the first thing I need to do is figure out which table has the field I want. Unlike the previous post, this one is a no-brainer: the Orion.Nodes table.

Once again, I'm going to test this out in SWQL Studio before copy/pasting the final version into the edited report.

I'm going to add a JOIN to include the Nodes table to the query.

```
End AS ActiveTime
FROM Orion.AlertObjects (nolock=true) AlertObjects
INNER JOIN Orion.AlertActive (nolock=true) AlertActive ON AlertObjects.AlertObjectID=AlertActive.AlertObjectID
INNER JOIN Orion.AlertConfigurations (nolock=true) AlertConfigurations ON AlertConfigurations.AlertID=AlertObjects.AlertID
INNER JOIN Orion.NodesCustomProperties (nolock=true) NodeCP ON AlertObjects.RelatedNodeID = NodeCP.NodeID
INNER JOIN Orion.Nodes (nolock=true) Nodes ON AlertObjects.RelatedNodeID = Nodes.NodeID
```

And add the node name (i.e., "Caption") field to the output. Note that I'm going to relabel it "NodeName" so it's differentiated from the existing "Caption" field.

```
SELECT DISTINCT
Nodes.Caption AS NodeName, Nodes.DetailsURL, NodeCP.Importance,
AlertActive.AlertActiveID, AlertObjects.AlertObjectID, AlertConfigurations.Name, AlertConfigu
ObjectType, AlertObjects.EntityUri, AlertObjects.EntityType, AlertObjects.EntityCaption,
ToLocal(AlertActive.TriggeredDateTime) AS TriggeredDateTime, AlertObjects.LastTriggeredDateTi
```

After adding the new column to my report, everything is working fine. Now let's dig in to how to make this clickable.

### Key Concepts

Before diving into the specific steps, I want to clarify the basic skills and knowledge I'm drawing on to make this happen:

1. HTML. You should understand how tags work, specifically the `<A HREF="">` tag ([HTML a href Attribute](#))
2. Literals in a SQL query ([How to Use Literal Character Strings in SQL SELECT Statement for MySQL](#))
3. Concatenating multiple elements in a SQL query ([+ \(String Concatenation\) \(Transact-SQL\) - SQL Server | Microsoft Docs](#))

With those prerequisites out of the way, we can move on to the Orion-specific aspects.

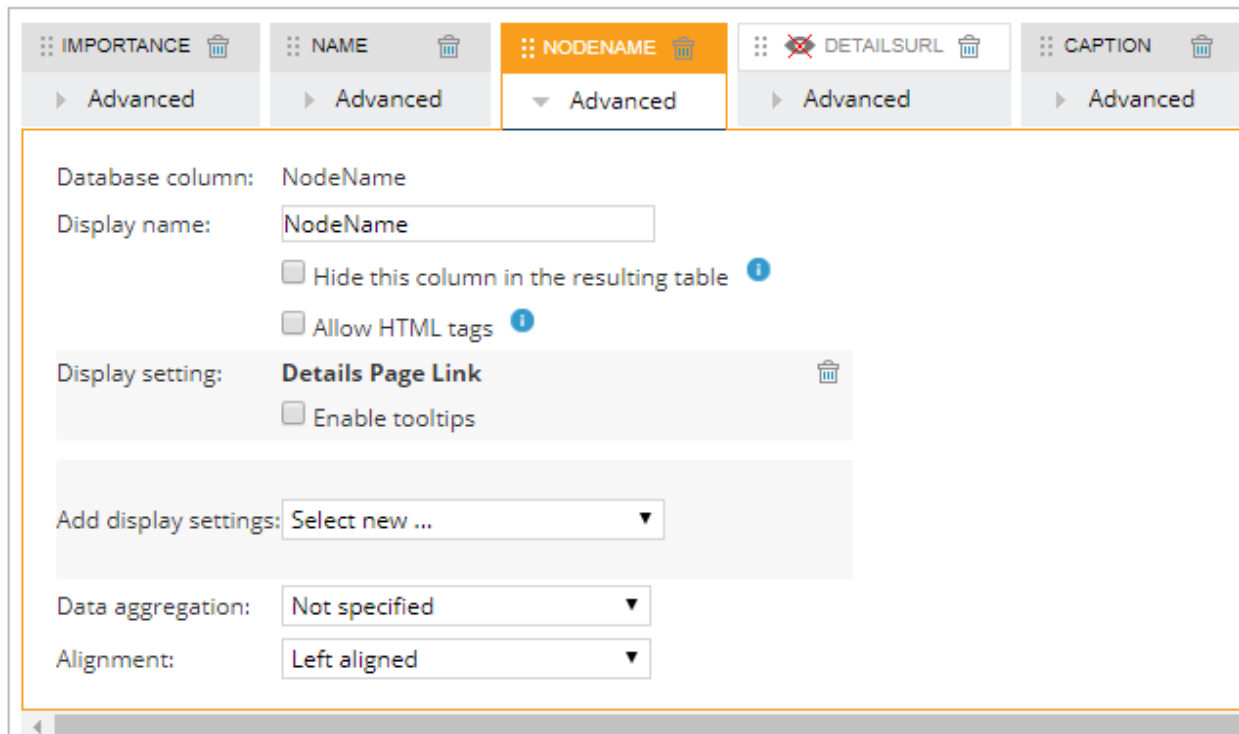
The first thing you need to know is that there's already a field ready-made for this in the Nodes table, called "DetailsURL." So, adding it to the existing field output is trivial.

### Slightly Off-Topic Detour

If all we actually wanted was to make the Node Name clickable, then this process would be VERY simple:

- Add the Caption and DetailsURL field from the Orion.Nodes table
- Add both of those fields to the report
- Click the "Advanced" arrow for the DetailsURL column and click the "Hide" checkbox
- Now, click the "Advanced" arrow for the NodeName column
- In the Display Settings drop-down, select "Details Page Link"

**Table layout:** [Edit column widths](#)



The screenshot shows the configuration panel for the 'NODENAME' column. The 'Database column' is 'nodeName'. The 'Display name' is 'nodeName'. There are checkboxes for 'Hide this column in the resulting table' and 'Allow HTML tags', both of which are currently unchecked. The 'Display setting' is set to 'Details Page Link', and there is an unchecked checkbox for 'Enable tooltips'. Below this, there is a dropdown for 'Add display settings' currently showing 'Select new ...'. At the bottom, there are dropdowns for 'Data aggregation' (set to 'Not specified') and 'Alignment' (set to 'Left aligned').

Because we only have one DetailsURL link, Orion Report Writer knows what we mean, and the Node names will be clickable.

And honestly, this is the right way to do things if all you want is a clickable column. I'm only showing you the more roundabout way because I want you to understand the techniques to add HTML to your query output, because we're going to use that in the next post.

### Meanwhile, Back at the SQL

OK, so just as a reminder, what we're doing here is creating a field that lists the node name and is clickable in the report.

- The node name field is "Nodes.Caption" (or more specifically, "Nodes.Caption AS nodeName")
- The field that has the Node url is Nodes.DetailsURL
- A web link uses the <A HTTP="URL">
- Surround literal elements in a SQL query with a single quote
- Combining elements together into a single output (i.e., "concatenation") in a SQL query uses the + symbol

With all that said, the following line should make sense:

```
'<A HREF="' +AlertObjects.EntityDetailsURL+' " target="_blank">' +Nodes.Caption+'</a>' AS NodeName,
```

Adding that to the All Alerts Report query we've been building. At the same time, you can remove the Nodes.Caption AS NodeName and Nodes.DetailsURL fields since we've effectively combined both of them into a single field output.

Adding the new column to the table will yield a report that looks like this:

## 001-LJA\_Importance-n-Nodes\_All Active Alerts



Summary of Orion Objects: **All Active Alerts**

### All Active Alerts for All Active Alerts

Ordered by:Importance - Ascending then by Name - Ascending then by NodeName - Ascending then by Caption - Ascending

IMPORTANCE	NAME	NODENAME	CAPTION	ACTIVE TIME	TRIGGERED DATE TIME	NOTE	ACKNOWLEDGE	ACKNOWI BY	ACKNOWLED DATE TIME
3	001_LJA-imute	<A HREF="/Orion/View.aspx?NetObject=l:76" target="_blank">cur-2851.lab.cur</a>	GigabitEthernet0/ - Gi0/1	867d 3h 9m	9/8/2016 12:27:22 PM				

Obviously, we have a problem with display. Click the "Advanced" arrow for the new NodeName column and check the "Allow HTML Tags" checkbox.

**Edit Resource: All Active Alerts for All Active Alerts** ⓘ You can change this after submit.

Title:

Subtitle:

Table layout: [Edit column widths](#)

Database column:

Display name:

Hide this column in the resulting table ⓘ

Allow HTML tags ⓘ

Add display settings:

Data aggregation:

Alignment:

This will cause Orion to interpret the HTML tags correctly and give you a report that looks more like this:

## 001-LJA\_Importance-n-Nodes\_All Active Alerts



Summary of Orion Objects: **All Active Alerts**

### All Active Alerts for All Active Alerts

Ordered by:Importance - Ascending then by Name - Ascending then by NodeName - Ascending then by Caption - Ascending

IMPORTANCE	NAME	NODENAME	CAPTION	ACTIVE TIME	TRIGGERED DATE TIME	NOTE	ACKNOWLEDGE	ACKNOWI BY	ACKNOWLED DATE TIME
3	001_LJA-imute	<a href="#">cur-2851.lab.cur</a>	GigabitEthernet0/ - Gi0/1	867d 3h 11m	9/8/2016 12:27:22 PM				

## Part 4: Creating an Acknowledge Link Where Once There Was None

In this last chapter I'm going to show how to use those two techniques to add an Acknowledge option, so that a report showing the current active alerts can be used by NOC staff to also manage those alerts, in addition to simply listing them out.

As a reminder, I've already built this report and posted it to the Content Exchange (<https://thwack.solarwinds.com/docs/DOC-204064>), in case you'd like to use it as a starting point for your own custom reporting adventure.

### Be Alert. The World Needs More Lerts.

When you look at the original "All Active Alerts" page in a standard Orion installation (here's an example from our live demo: <https://oriondemo.solarwinds.com/Orion/NetPerfMon/Alerts.aspx>), there's a few things that you may notice about the "Acknowledge" item:

1. It's modal—meaning it pops up on top of everything else
2. It's interactive—you can add notes, set a permanent option not to be prompted, cancel out, etc.

...and, if you look at the underlying code on the Alerts.aspx page, you'll note that

3. It's actually JavaScript.

Now I'm certain that some enterprising person could embed JavaScript into an Orion custom report. In fact, I challenge you readers to do so, and post your results to the THWACK® product forums and the Content Exchange to share with everyone. But I'm a man who knows his limitations—both in terms of programming skill and the time I have to devote to this side project—so I'm going to go with an easier option: I'm going to use the "Acknowledge" link you can find in an email alert.

What? You've never set up an email alert with an embedded "acknowledge" link? Well let's rectify that situation right now.

First, create a new alert.

#### Add New Alert



PROPERTIES > TRIGGER CONDITION > RESET CONDITION > TIME OF DAY > TRIGGER ACTIONS > RESET ACTIONS > SUMMARY

#### 1. Alert Properties

Name of alert definition (required)  
Stupid alert to see the acknowledge thingy

Description of alert definition  
Displayed on Manage alerts page.

Enabled (On/Off)  
ON

It doesn't really matter what we call it, and it doesn't even really matter what the alert triggers on, as long as it triggers at least once. The most important step is to set up an email alert and embed the acknowledge alert link variable.

Once it's done, I'll kick off that alert, receive the email, and see what that link looks like.

**Configure Action: Send An Email/Page**

Name of action  
email Leon

1. Recipients leon.adato@solarwinds.com

2. Message

Subject  
Error on \${N=SwisEntity;M=Caption} at \${N=Alerting;M=AlertTriggerTime;F=DateTime} INSERT VARIABLE

Message  
You want your acknowledge link? Here it is already!  
\${N=Alerting;M=AcknowledgeLink} INSERT VARIABLE

## Let's SWQL One More Time

Now that I know what the link looks like, I have to go back to SWQL studio and see if I can find that link or at least a portion of it embedded SOMEWHERE. After just a bit of searching, I found it in the AlertObjects table, where there's an AlertDefID field that matches up with the tail-end of the standard alert URL `"/Orion/Netperfmon/AckAlert.aspx?AlertDefID="`.

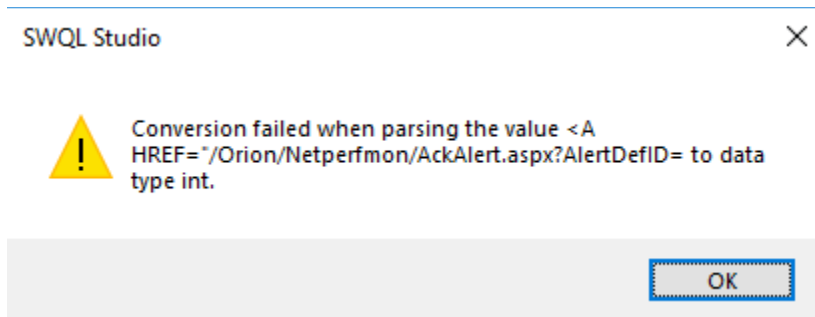
SELECT TOP 1000 AlertObjectID, AlertID, EntityUri, EntityType, EntityCaption, EntityDetailsUrl, EntityNetObjectID, RelatedNodeUri, RelatedNodeId, RelatedNodeDe...  
TriggeredCount, LastTriggeredDateTime, Conte...  
FROM Orion.AlertObjects

Mon 1/28/2019 4:17 PM  
Network Performance Monitor <noreply@solarwinds.com>  
Error on 10.199.1.74 at Monday, January 28, 2019 3:16 PM  
To: Adato, Leon  
<https://dev-aus-lada-01:443/orion/netperfmon/ack-alert.aspx?alertdefid=1753>  
You want your acknowledge link? Here it is already!  
Click or tap to follow link.  
[Click here to acknowledge this alert](#)

AlertObjectID	AlertID	EntityUri	EntityType	EntityCaption	Entity
724	116	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	dev-aus-lada-02	/Orion
725	116	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	DEV-AUS-BROW-S.swdev.local	/Orion
726	116	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	lab-tex-dc-01 lab tex	/Orion
727	116	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	LAB-EXC2003-DC	/Orion
728	116	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	xenapp711sec lab tex	/Orion
732	116	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	lab-wsus-01 lab tex	/Orion
735	116	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	lab-UTM-web.wptestdomain.local	/Orion
736	116	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	lab-websphere8	/Orion
1749	139	swis://dev-aus-lada-01./Orion/Orion.V...	Orion.VIM.VirtualMachines	HP_just_TestingVM	/Orion
1750	158	swis://dev-aus-lada-01./Orion/Orion.V...	Orion.VIM.VirtualMachines	HP_just_TestingVM	/Orion
1751	149	swis://dev-aus-lada-01./Orion/Orion.V...	Orion.VIM.Hosts	lab-esx-hp.lab.tex	/Orion
1753	187	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	LAB-HP-OPSMGR	/Orion
1754	187	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	10.199.1.74	/Orion
1755	187	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	lab-eccm2007.lab.tex	/Orion
1756	187	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	lab-sql-demo	/Orion
1757	187	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	lab-netflowrepl	/Orion
1758	187	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	lab-eccm2012R2b.lab.tex	/Orion
1759	187	swis://dev-aus-lada-01./Orion/Orion.N...	Orion.Nodes	dev-aus-pa-ua swdev.local	/Orion

By concatenating the AlertDefID onto the end of that URL, I'm in busine...

Hang on, it threw an error.



The issue here is that the URL is a word-based object (a "string" in technical parlance) but the AlertDefID is a numeric value. So, we need to convert them to match. Since you can't make words into numbers, we'll have to convert the AlertDefID number so that it's treated as a string:  
`toString(AlertObjects.AlertObjectID)`

The final field description would look like this:

```
'<A HREF="/Orion/Netperfmon/AckAlert.aspx?AlertDefID='+toString(AlertObjects.AlertObjectID)+'" target="_blank">ClickToAck</a>' AS AcknowledgeIt
```

I'll add that to my SWQL statement, update it in the report definition, add the "AcknowledgeIt" field to the report, and I have something that looks like this:

**All Active Alerts for All Active Alerts**

Ordered by: Importance - Ascending then by Name - Ascending then by NodeName - Ascending then by Caption - Ascending

IMP	NAME	NODENAME	CAPTION	ACTIVE TIME	ACKNOWLEDGE BY	ACKNOWLEDGE DATE TIME	ACKNOWLEDGEIT
3	001_LJA-imute	cur-2851.lab.cur	GigabitEthernet0/1 - Gi0/1	872d 4h 7m			<a href="#">ClickToAck</a>
3	Stupid alert to see the acknowledge thingy	dev-aus-pa-ua.swdev.local	dev-aus-pa-ua.swdev.local	17m			<a href="#">ClickToAck</a>
3	Stupid alert to see the acknowledge thingy	LAB-HP-OPSMGR	LAB-HP-OPSMGR	17m			<a href="#">ClickToAck</a>
3	Stupid alert to see the acknowledge thingy	10.199.1.74	10.199.1.74	17m			<a href="#">ClickToAck</a>
3	Stupid alert to see the acknowledge thingy	lab-sccm2007.lab.tex	lab-sccm2007.lab.tex	17m			<a href="#">ClickToAck</a>
3	Stupid alert to see the acknowledge thingy	lab-sql-demo	lab-sql-demo	17m			<a href="#">ClickToAck</a>
3	Stupid alert to see the acknowledge thingy	lab-netflowrepl	lab-netflowrepl	17m			<a href="#">ClickToAck</a>
3	Stupid alert to see the acknowledge thingy	lab-sccm2012R2b.lab.tex	lab-sccm2012R2b.lab.tex	17m			<a href="#">ClickToAck</a>

## The Mostly Un-Necessary Summary

Throughout this ebook, I hope I've shown that seemingly challenging, "advanced" tasks are nothing more than the combining of relatively simple, common skills or bits of knowledge in new and creative ways. While every IT practitioner cannot be expected to have every combination of these techniques and concepts, what IS true is that you can take what you have now and think about how it can be re-used; and as you grow in your skills, knowledge, and experience, that base of creativity only grows.

But that's not all!

In the next chapter I'll take an introspective turn, and talk about what I learned in all of this from a philosophical, rather than technical, point of view.

## Chapter 5: What Have We Learned?

At the start of this ebook, I wrote about building a custom report in Orion®. It all started when a customer reached out with an “unsolvable” problem. Just to be clear, they weren’t trying to play on my ego. They had followed all the other channels and really *did* think the problem had no solution. After describing the issue, they asked, “Do you know anyone on the development team who could make this happen?”

As a matter of fact, I *did* know someone who could make it happen: me.

That’s not because I’m a super-connected SolarWinds employee who knows the right people to bribe with baklava to get a tough job done. (I do, but that wasn’t needed here.) Nor was it because, as I said at the beginning of the week, “I’m some magical developer unicorn who flies in on my hovercraft, dumps sparkle-laden code upon a problem, and all is solved.”

Really, I’m more like a DevOps ferret than a unicorn—a creature that scrabbles around, seeking out hidden corners and openings, and delving into them to see what secret treasures they hold. Often, all you come out with is an old wine cork or a dead mouse. But every once in a while, you find a valuable gem, which I tuck away into my stash of shiny things. And that leads me to the first big observation I recognized as part of this process:

**Lesson #1: IT careers are more often built on a foundation of “found objects”—small tidbits of information or techniques we pick up along the way—which we string together in new and creative ways.**

And in this case, my past ferreting through the dark corners of the Orion Platform had left me with just the right stockpile of tricks and tools to provide a solution.

I’m not going to dig into the details of how the new report was built, because that’s what the other four posts in this series are all about. But I *\*do\** want to list out the techniques I used, to prove a point:

- Know how to edit a SolarWinds report
- Understand basic SQL queries (really just select and joins)
- Have a sense of the Orion schema
- Know some HTML fundamentals

Honestly, that was it. Just those four skills. Most of them are trivial. Half of them are skills that most IT practitioners may possess, regardless of their involvement with SolarWinds solutions.

Let’s face it, making a loaf of bread isn’t technically complicated. The ingredients aren’t esoteric or difficult to handle. The process of mixing and folding isn’t something that only trained hands can do. And yet it’s not easy to execute the first time unless you are comfortable with the associated parts. Each of the above techniques had some little nuance, some minor dependency, that would have made this solution difficult to suss out unless you’d been through it before.

Which takes me to the next observation:

**Lesson #2: None of those techniques is complicated. The trick was knowing the right combination and putting them together.**

I had the right mix of skills, and so I was able to pull them together. But this wasn’t a task my manager set for me. It’s not in my scope of work or role. This wasn’t part of a side-hustle that I do to pay for my kid’s braces or feed my \$50-a-week comic book habit. So why would I bother with this level of effort?

OK, I’ll admit—I figured it might make a good story. But besides that?



I'd never really dug into Orion's web-based reporting before. I knew it was there, I played with it here and there, but really dug into the guts of it and built something useful? Nah, there was no burning need. This gave me a reason to explore and a goal to help me know when I was "done." Better still, this goal wouldn't just be a thought experiment, it was actually helping someone. And that leads me to my last observation:

**Lesson #3: Doing for others usually helps you more.**

I am now a more accomplished Orion engineer than I was when I started, and in the process I've (hopefully) been able to help others on THWACK® become more accomplished as well.

And there's nothing complicated about knowing how that's a good thing.