Contents
What is an account aware report?................................................................. 3
  Using report templates for multiple accounts ......................................... 3
Setting up the environment........................................................................ 4
  Set up iReport................................................................. 4
Token replacement................................................................................... 7
Creating reports....................................................................................... 8
  Creating the first test report............................................................ 8
  Creating a report using the stored procedure ...................................... 14
  Parameters for the stored procedure ................................................... 18
  Creating a report using dynamic SQL with token .............................. 19
Known problems..................................................................................... 31
Appendix ................................................................................................. 32
  XML for example report using SQL query language ......................... 32
  XML for example report using nimsoft query language .................... 35
What is an account aware report?

The Unified Reporter is a reports portlet that can be added to a page in the Nimsoft Unified Monitoring Portal (UMP). In Unified Reporter you can create reports that are what we call account aware. Account aware means you can create a report template, and the data displayed when the report is executed differs according to the account the user belongs to.

Typically when setting up users in a multi-tenancy environment, you will assign each account a unique origin. This way, users for the account see only data from that origin.

This guide assumes you are familiar with separating data using the origin field. It also assumes you know how to create queries in SQL and are familiar with how the iReport graphical report designer works. For a quick introduction, see the Nimsoft Unified Reporter iReport Quick Start Guide. For advanced topics on iReport, see the iReport Ultimate Guide.

Using report templates for multiple accounts

You can reuse reports across accounts and filter data based on who is viewing them.

To do this, we use a query executor. It allows us to determine the security context of the user(s) viewing a report in Unified Reporter. Based on this security context, we can determine the appropriate filter to use for the data.

Nimsoft has written its own query executor and plugged it into the Unified Reporter server. When a report is executed, the Query Executor modifies the SQL query for the report.
Setting up the environment

To design account aware reports, we recommend you design them using iReport and add support for our query executor. Then deploy the report to Unified Reporter and view the results.

First you need to set up iReport.

Set up iReport

1. Create a folder, for example C:\iReport_classes, on the machine where iReport is installed.
2. Navigate to <Nimsoft>/probes/service/wasp/webapps/jasperserver-pro/WEB-INF/classes and copy the folders com, net, and bundles to the folder you created in step 1.
3. Choose the Tools > Options menu.
4. Click the Classpath tab, then click Add Jar.
6. Choose spring-security-core-2.0.6.RELEASE.jar and click Open.
7. Click Add Folder.
8. Navigate to the folder you created in step 1, and click Open.
9. Click the **Query Executers** tab.
10. Click **Add**.

11. Add the following values (enter “nimsoft” in lowercase for the language):

<table>
<thead>
<tr>
<th>Language:</th>
<th>nimsoft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factory Class</td>
<td>net.sf.jasperreports.engine.query.NimsoftQueryExecutorFactory</td>
</tr>
<tr>
<td>Fields provider</td>
<td>com.jaspersoft.ps.nimsoft.NimsoftFieldsProvider</td>
</tr>
</tbody>
</table>

12. Click **OK**.
The nimsoft query executor should now be listed.
11. Click **OK**.
The query executor should now be ready for use.

**Token replacement**

The query executor examines the query of each report that is set up to use the nimsoft query executor. If a known token is found, it is replaced with runtime values of the user who is viewing the report on the Unified Reporter server.

The following table lists the tokens that can be used:

<table>
<thead>
<tr>
<th>token</th>
<th>If nimsoft user is executing</th>
<th>If account/contact user is executing</th>
</tr>
</thead>
<tbody>
<tr>
<td>nimsoft_account_id</td>
<td>null</td>
<td>&lt;account_id&gt;</td>
</tr>
<tr>
<td>nimsoft_account_name</td>
<td>null</td>
<td>&lt;account_name&gt;</td>
</tr>
<tr>
<td>nimsoft_acl</td>
<td>&lt;acl&gt;</td>
<td>Null</td>
</tr>
</tbody>
</table>
**Creating reports**

**Creating the first test report**

In this example, we will create a basic report that filters data depending on who is logged in to Unified Reporter.

2. In the Report Inspector windows, right-click and choose **Edit Query**.
3. Enter this query: `select myACL='ireport dummy'`
4. Make sure the **Report query** tab is selected, the **Query language** is SQL, and uncheck **Automatically Retrieve Fields**.
5. Click **Read Fields**.

6. In the bottom pane, the field information you entered is now listed.
7. Click **OK**.
8. Drag the **myACL** field onto the detail section of your report pane.
9. Click **File, Save**.
10. Connect to your repository using the Repository Navigator, then publish the report as **acl1** under **/public**.
11. Log in to Unified Reporter and view your acl1 report.

The report displays the value of **ireport dummy** that we typed in our query.

12. Go back to iReport and change your query to: select **myACL='nimsoft_acl'**
13. Change the query language from **SQL** to **nimsoft**, but do **not** click the **Read Fields** button.
14. Click **OK**.
15. If the **Save** button is not active, change your report pane to trigger the **Save** button.
16. Save the updated report.
17. Find your report unit in the Repository Navigator window. Expand it and right-click the main report definition file (usually main.jrxml).
18. Select **replace with current jrxml**.
19. Reopen your report unit in Unified Reporter to see how the content changed.

20. Log out.
22. In the Report Inspector window, expand the Fields node and delete the field called **myACL**.
23. Change the query to: `select myACL='nimsoft_acl',
account='nimsoft_account_name'
` 
24. Change the **Query language** to **SQL**, click **Read Fields**, then click **OK**.
25. Verify that your fields are up to date. If they are not, you may have to go back into the query and click Read Fields one more time.

26. Change the Query language to Nimsoft.
27. Drag the account field onto the report page.
28. Save the report and republish it to the repository.
29. Log in to Unified Reporter as both an account and a nimsoft user and see the results.
nimsoft user

Superuser

contact in account
developer_account

dtvloptr_account
In this example, we have demonstrated how tokens can be replaced at runtime, depending on who is viewing the report. To make this easier to use, we have created a stored procedure that can be executed to filter data based on who is logged in.

**Creating a report using the stored procedure**

In this example, you want to list your sources and targets with defined QOS for CPU Usage.

Your query might look like this:

```
SELECT qos.source, qos.target FROM S_QOS_DATA qos
WHERE qos.qos = 'QOS_CPU_USAGE'
ORDER BY qos.source ASC, qos.target ASC
```

Result example:

<table>
<thead>
<tr>
<th>source</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>193.71.55.101</td>
<td>193.71.55.101</td>
</tr>
<tr>
<td>193.71.55.101</td>
<td>Idle</td>
</tr>
<tr>
<td>193.71.55.101</td>
<td>System</td>
</tr>
<tr>
<td>193.71.55.101</td>
<td>User</td>
</tr>
<tr>
<td>193.71.55.101</td>
<td>Wait</td>
</tr>
<tr>
<td>193.71.55.102</td>
<td>193.71.55.102</td>
</tr>
<tr>
<td>193.71.55.102</td>
<td>Idle</td>
</tr>
<tr>
<td>193.71.55.102</td>
<td>System</td>
</tr>
<tr>
<td>193.71.55.102</td>
<td>User</td>
</tr>
<tr>
<td>193.71.55.102</td>
<td>Wait</td>
</tr>
<tr>
<td>193.71.55.156</td>
<td>CPU last 5min</td>
</tr>
<tr>
<td>c3po.nimsoft.no</td>
<td>c3po.nimsoft.no</td>
</tr>
<tr>
<td>c3po.nimsoft.no</td>
<td>Idle</td>
</tr>
<tr>
<td>c3po.nimsoft.no</td>
<td>System</td>
</tr>
<tr>
<td>c3po.nimsoft.no</td>
<td>User</td>
</tr>
<tr>
<td>c3po.nimsoft.no</td>
<td>Wait</td>
</tr>
<tr>
<td>cisco1710.nimsoft.no</td>
<td>CPU last 5min</td>
</tr>
<tr>
<td>cisco1721.nimsoft.no</td>
<td>CPU last 5min</td>
</tr>
<tr>
<td>everest.nimsoft.no</td>
<td>everest.nimsoft.no</td>
</tr>
<tr>
<td>kenobi.nimsoft.no</td>
<td>kenobi.nimsoft.no</td>
</tr>
<tr>
<td>nimrum.nimsoft.no</td>
<td>nimrum.nimsoft.no</td>
</tr>
</tbody>
</table>

If you were to filter by origin, you would have to filter data on the origin column in the s_qos_data table.

If we change the query to this:
SELECT qos.source, qos.target FROM S_QOS_DATA qos
WHERE qos.qos = 'QOS_CPU_USAGE'
AND qos.origin = 'xpgeha2'
ORDER BY qos.source ASC, qos.target ASC

We get a different result:

<table>
<thead>
<tr>
<th>source</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>193.71.55.156</td>
<td>CPU last 5min</td>
</tr>
<tr>
<td>cisco1710.nimsoft.no</td>
<td>CPU last 5min</td>
</tr>
<tr>
<td>cisco1721.nimsoft.no</td>
<td>CPU last 5min</td>
</tr>
<tr>
<td>rackswitch.nimsoft.no</td>
<td>CPU last 5min</td>
</tr>
<tr>
<td>ruth.nimsoft.no</td>
<td>CPU last 5min</td>
</tr>
<tr>
<td>switch1.nimsoft.no</td>
<td>CPU last 5min</td>
</tr>
</tbody>
</table>

We have created a stored procedure to help write these types of queries, for use with iReport and Unified Reporter, to make account aware queries.

The stored procedure is “spn_js_ExecuteAccountAwareReport”.

It has these arguments:

- @Select varchar(max) = null,
- @Orderby varchar(max) = null,
- @Restriction varchar(max) = null, /* and d.origin = $placeholder */ /* where d.origin = $placeholder */
- @AccountId varchar(max) = null, /* nimsoft_account_id */
- @ProcedureTrace int = 5

Always set the @AccountId parameter to: nimsoft_account_id.

Example:

EXEC [dbo].[spn_js_ExecuteAccountAwareReport]
    @Select = N'SELECT qos.source, qos.target FROM S_QOS_DATA qos ',
    @Restriction = N'WHERE qos.qos = 'QOS_CPU_USAGE' AND qos.origin=$origin_place_holder',
    @Orderby = N'ORDER BY qos.source ASC, qos.target ASC',
    @AccountId = nimsoft_account_id

The stored procedure produces the SQL based on the values provided. The token nimsoft_account_id is replaced by the query executor at runtime.

Create an account aware report:

1. Create a new empty report.
2. Set the **Query language** to **SQL**.
3. Copy the example query above into the window and click **Read Fields**.
4. Click **OK** and verify the fields are populated.

5. Drag the fields onto the report. Also drag target to the column header as an aggregate function.
6. Set the **Query language** to **nimsoft** and click **OK**.
7. Save the report and publish to Unified Reporter.
8. Log in as a nimsoft user and then as an account contact user to see the results.

For a nimsoft user:

```
<table>
<thead>
<tr>
<th>View</th>
<th>Manage</th>
<th>Create</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

There are 26 targets on this page, 30 on page 2, and so on.
An account contact user sees filtered data based on the origin.

You now have created a sample report, using the stored procedure and the nimsoft query executor to replace the token nimsoft_account_id at runtime with the actual account id. It will be used by the stored procedure to filter on the origin.

**Parameters for the stored procedure**

<table>
<thead>
<tr>
<th>parameter</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Select</td>
<td>'SELECT qos.source, qos.target FROM S_QOS_DATA qos '</td>
</tr>
<tr>
<td>@Restriction</td>
<td>'WHERE qos.qos = ''QOS_CPU_USAGE'' AND qos.origin=$origin_place_holder'</td>
</tr>
<tr>
<td>@Orderby</td>
<td>'ORDER BY qos.source ASC, qos.target ASC '</td>
</tr>
<tr>
<td>@AccountId</td>
<td>nimsoft_account_id</td>
</tr>
</tbody>
</table>

The restriction parameter should contain the origin column = $origin_place_holder in all the tables that you include that have the column. In the above example, the s_qos_data table is included in the query and we include the origin filter as shown above.
The stored procedure will create and run SQL statements like this:

(Assuming a contact user with account id 2 is running the report)

Before token replacement:

```sql
SELECT qos.source, qos.target FROM S_QOS_DATA qos WHERE qos.qos = 'QOS_CPU_USAGE' AND qos.origin=$origin_place_holder ORDER BY qos.source ASC, qos.target ASC
```

After token replacement:

```sql
SELECT qos.source, qos.target FROM S_QOS_DATA qos WHERE qos.qos = 'QOS_CPU_USAGE' AND qos.origin IN (SELECT origin FROM CM_ACCOUNT_OWNERSHIP WHERE account_id = 2) ORDER BY qos.source ASC, qos.target ASC
```

**Creating a report using dynamic SQL with token**

Because our special tokens are not recognized by the MS SQL server directly, we recommend you use dynamic SQL while designing your query, and leave the token as a comment in the SQL statements until you are ready to publish your report to Unified Reporter.

If you do not wish to use the stored procedure, you can create more complex queries where you have more control.

One of the last steps is to change the **Query language** from **SQL** to **nimsoft** and then publish the report again.

Here is an example of using dynamic SQL. Let’s assume you wish to create a report of your account users.

- The report should be filtered based on account
- The report should display average ldap response time last 30 minutes (assuming your account has any ldap response qos times)

For an overview of which QOS you have available, run this query on your SQL server:

```sql
select def.description, def.name from s_qos_definition def order by def.description asc
```
In this demo database we have 108 different objects. For this example we choose QOS_LDAP_RESPONSE_TIME.

Next, we pull data from both s_qos_data and the corresponding rn table, which holds the actual samples. We need to find which rn table to use.

You can use this query to find the rn table for a given QOS.

```sql
declare @qos1 varchar(max)
select @qos1 = 'QOS_LDAP_RESPONSE_TIME'

declare @rntable1 varchar(max)
select @rntable1 = 'RN_QOS_DATA_' + reverse(stuff('0000' , 1, len(cast(s.qos_def_id as varchar(max))), reverse(cast(s.qos_def_id as varchar(max))))) from s_qos_definition s where name = @qos1

select qos=@qos1, rn=@rntable1
```

Result in our demo database:

```
<table>
<thead>
<tr>
<th>qos</th>
<th>rn</th>
</tr>
</thead>
<tbody>
<tr>
<td>QOS_LDAP_RESPONSE_TIME</td>
<td>RN_QOS_DATA_C107</td>
</tr>
</tbody>
</table>
```

You can hard code the value for your report or use this dynamic approach. This guide will use the dynamic approach.

Now you have a QOS and the rn table. The SQL to get the average last 30 minutes of LDAP response times might have been this if you do not use dynamic SQL:

```sql
declare @qos1 varchar(max)
select @qos1 = 'QOS_LDAP_RESPONSE_TIME'
```
```
declare @dtFrom datetime
declare @dtTo datetime
select @dtFrom = dateadd(minute, -30, getdate())
select @dtTo = getdate()

select s.source, 'ldap profile'=s.target, s.host,
    avg=avg(rn.samplevalue) from s_qos_data s
inner join rn_qos_data_0107 rn
    on rn.table_id = s.table_id
where s.qos = @qos1
and rn.sampletime between @dtFrom and @dtTo
group by s.source, s.target, s.host
order by 'ldap profile' asc
```

Result:

<table>
<thead>
<tr>
<th>source</th>
<th>ldap profile</th>
<th>host</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>support</td>
<td>objects in testlab dc</td>
<td>193.71.55.115</td>
<td>NULL</td>
</tr>
<tr>
<td>js1</td>
<td>odin</td>
<td>193.71.55.103</td>
<td>14.133333</td>
</tr>
<tr>
<td>support</td>
<td>odin2</td>
<td>193.71.55.115</td>
<td>1629.233333</td>
</tr>
<tr>
<td>support</td>
<td>users that can login</td>
<td>193.71.55.115</td>
<td>NULL</td>
</tr>
</tbody>
</table>

In the example, two of the LDAP profiles return null data for all rows in the time span (30 minutes).

You can modify this SQL to be dynamic like this:

```
declare @qos1 varchar(max)
select @qos1 = 'QOS_LDAP_RESPONSE_TIME'

declare @rntable1 varchar(max)
select @rntable1 = 'RN_QOS_DATA_' + reverse(stuff('0000', 1, len(cast(s.qos_def_id as varchar(max))), reverse(cast(s.qos_def_id as varchar(max))))) from s_qos_definition s where name = @qos1

declare @sql varchar(max)
select @sql = '
declare @dtFrom datetime
declare @dtTo datetime
select @dtFrom = dateadd(minute, -30, getdate())
select @dtTo = getdate()

select s.source, ''ldap profile''=s.target, s.host,
    avg=avg(rn.samplevalue) from s_qos_data s
inner join ' + @rntable1 + ' rn
    on rn.table_id = s.table_id
where s.qos = ''' + @qos1 + '''
and rn.sampletime between @dtFrom and @dtTo
group by s.source, s.target, s.host
order by ''ldap profile'' asc
```
The result is the same as the previous example.

You can then add a placeholder for the SQL and token needed to make the report account aware (when we run the report through the query executor).

Change the SQL used in the previous example to this:

```sql
declare @qos1 varchar(max)
select @qos1 = 'QOS_LDAP_RESPONSE_TIME'

declare @rntable1 varchar(max)
select @rntable1 = 'RN_QOS_DATA_' + reverse(stuff('0000', 1, len(cast(s.qos_def_id as varchar(max))), reverse(cast(s.qos_def_id as varchar(max)))))) from s_qos_definition s where name = @qos1

declare @sql varchar(max)
select @sql = '

declare @dtFrom datetime
declare @dtTo datetime
select @dtFrom = dateadd(minute, -30, getdate())
select @dtTo = getdate()

select s.source, ''ldap profile''=s.target, s.host, avg=avg(rn.samplevalue) from s_qos_data s
inner join ' + @rntable1 + ' rn
on rn.table_id = s.table_id
where s.qos = ''' + @qos1 + '''
and rn.sampletime between @dtFrom and @dtTo

/* add room here for account aware token */

select @sql = @sql + '

group by s.source, s.target, s.host
order by ''ldap profile'' asc

exec (@sql)
```

You can now add the SQL and token needed to make the SQL account aware.

We suggest adding a single long line like this, which can be copied onto new reports you want to make account aware:
if isnull(nimsoft_account_id, -1) > 0 begin select @query = @query + ' and d.origin in (select origin from cm_account_ownership where account_id = nimsoft_account_id ) ' end

This line appends the filter needed if the token is replaced at runtime with a valid account id.

**NOTE:** This line must be left commented out during our session on the SQL server directly, because the SQL server does not understand the token nimsoft_account_id.

Our complete SQL is now:

```sql
declare @qos1 varchar(max)
select @qos1 = 'QOS_LDAP_RESPONSE_TIME'
declare @rntable1 varchar(max)
select @rntable1 = 'RN_QOS_DATA_' + reverse(stuff('0000' , 1, len(cast(s.qos_def_id as varchar(max))), reverse(cast(s.qos_def_id as varchar(max))))) from s_qos_definition s where name = @qos1
declare @sql varchar(max)
select @sql = 
        declare @dtFrom datetime
        declare @dtTo datetime
        select @dtFrom = dateadd(minute, -30, getdate())
        select @dtTo = getdate()
        select s.source, ''ldap profile''=s.target, s.host, avg=avg(rn.samplevalue) from s_qos_data s
        inner join ' + @rntable1 + ' rn
        on rn.table_id = s.table_id
        where s.qos = ''' + @qos1 + '''
        and rn.sampletime between @dtFrom and @dtTo
        group by s.source, s.target, s.host
        order by ''ldap profile'' asc

exec (@sql)
```

Now we will create a report in Unified Reporter that uses this SQL and the nimsoft query executor.

1. Create a new blank report in ireport.
2. In the report inspector, select the report name and name it **my first account aware report.**

3. Edit the query for the report by copying the query we prepared in the previous example.

4. Leave the **Query language** set to **SQL** and click **Read Fields.**
5. Design your report. Here is one example of how it might look:

If you click the XML button, you see an XML view of the report. You can copy this XML to your report for the purpose of this demo (assuming you used the same LDAP response QOS).

The example XML used for this report can be seen in the appendix to this document.
6. Publish the report to the Unified Reporter repository and view the sample report which currently uses normal SQL.

7. Go back to iReport and change the **Query language** to nimsoft.
8. Remove the two hyphens (--) that comment out the nimsoft_account_id token line.
9. Click **OK**.
10. Make a change in the report content to trigger the **Save** button, then click **Save**.
11. Update your report in the repository (right-click and replace `main.jrxml` with current).

Here is one example of what the report will look like for a contact user who does not have any QOS with the ldap_response probe.

In the screenshot above you see the word **acc3**. This is the same as you type when you publish the report to Unified Reporter repository:
It is the same as you see in Unified Reporter:

In the demo described here, we used acc3 as the display name of the report unit.
This is what the report will look like for a different contact user belonging to an account that has LDAP response records:
And the nimsoft users do not have any filters:

<table>
<thead>
<tr>
<th>Source</th>
<th>LDAP Profile</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>support</td>
<td>objects in testlab dc</td>
<td>null</td>
</tr>
<tr>
<td>support</td>
<td>odm2</td>
<td>1533.32</td>
</tr>
<tr>
<td>support</td>
<td>users that can login</td>
<td>null</td>
</tr>
</tbody>
</table>

Total pages: 1  December 01, 2009
Known problems

- Publishing a new report to the Unified Reporter repository using nimsoft as the query language displays an error message. Subsequent updates to a report unit that already exists in the Unified Reporter repository do not display this error. We recommend first publishing using SQL as the query language, then republishing using nimsoft as the query language in order to avoid this error message.
- The SQL language does not understand the nimsoft query executor tokens.
Appendix

XML for example report using SQL query language

```xml
<?xml version="1.0" encoding="UTF-8"?>
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports
http://jasperreports.sourceforge.net/xsd/jasperreport.xsd" name="my first account aware report" pageWidth="595" pageHeight="842" columnWidth="535"
leftMargin="20" rightMargin="20" topMargin="20" bottomMargin="20">
  <queryString>
    <![CDATA[
    declare @qos1 varchar(max)
    select @qos1 = 'QOS_LDAP_RESPONSE_TIME'
    
    declare @rntable1 varchar(max)
    select @rntable1 = 'RN_QOS_DATA_' + reverse(stuff('0000' , 1,
    len(cast(s.qos_def_id as varchar(max))), reverse(cast(s.qos_def_id as
    varchar(max)))))) from s_qos_definition s where name = @qos1
    
    declare @sql varchar(max)
    select @sql = '
    declare @dtFrom datetime
    declare @dtTo datetime
    select @dtFrom = dateadd(minute, -30, getdate())
    select @dtTo = getdate()
    select s.source, ''ldap profile''=s.target, s.host, avg=avg(rn.samplevalue) from
    s_qos_data s
    inner join ' + @rntable1 + ' rn
    on rn.table_id = s.table_id
    where s.qos = ''' + @qos1 + '''
    and rn.sampletime between @dtFrom and @dtTo
    ,
    /* add room here for account aware token */
    --if isnull(nimsoft_account_id, -1) > 0 begin select @sql = @sql + ' and s.origin in
    (select origin from cm_account_ownership where account_id = nimsoft_account_id )
    ' end
    select @sql = @sql + ' group by s.source, s.target, s.host
    order by ''ldap profile'' asc
    ,
    exec (@sql)
    ]]><queryString>
</jasperReport>
```
<table>
<thead>
<tr>
<th>Source</th>
<th>LDAP Profile</th>
<th>Host</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>![My First account aware report]]&gt;</td>
<td>![CDATA[My First account aware report]]&gt;</td>
<td>![CDATA[My First account aware report]]&gt;</td>
<td>![CDATA[My First account aware report]]&gt;</td>
</tr>
</tbody>
</table>
XML for example report using nimsoft query language

<?xml version="1.0" encoding="UTF-8"?>
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsiLocation="http://jasperreports.sourceforge.net/jasperreports
http://jasperreports.sourceforge.net/xsd/jasperreport.xsd" name="my first account aware report" pageWidth="595" pageHeight="842" columnWidth="535"
leftMargin="20" rightMargin="20" topMargin="20" bottomMargin="20">
  <queryString language="nimsoft">
    <![CDATA[
declare @qos1 varchar(max)
select @qos1 = 'QOS_LDAP_RESPONSE_TIME'

declare @rntable1 varchar(max)
select @rntable1 = 'RN_QOS_DATA_' + reverse(stuff('0000' , 1,
len(cast(s.qos_def_id as varchar(max))), reverse(cast(s.qos_def_id as
varchar(max)))))) from s_qos_definition s where name = @qos1

declare @sql varchar(max)
select @sql = '
declare @dtFrom datetime
declare @dtTo datetime
select @dtFrom = dateadd(minute, -30, getdate())
select @dtTo = getdate()

select s.source, "ldap profile"=s.target, s.host, avg(avg(rn.samplevalue) from
s_qos_data s
inner join ' + @rntable1 + ' rn
on rn.table_id = s.table_id
where s.qos = ''' + @qos1 + '''
and rn.sampletime between @dtFrom and @dtTo
,'

/* add room here for account aware token */
if isnull(nimsoft_account_id, -1) > 0 begin select @sql = @sql + ' and s.origin in
(select origin from cm_account_ownership where account_id = nimsoft_account_id )
' end

select @sql = @sql + ','
group by s.source, s.target, s.host
order by "ldap profile" asc
,'
exec (@sql)]]></queryString>
  <field name="source" class="java.lang.String">
    <fieldDescription><![CDATA[]]></fieldDescription>
  </field>
  <field name="ldap profile" class="java.lang.String">
    <fieldDescription><![CDATA[]]></fieldDescription>
  </field>
</jasperReport>
<table>
<thead>
<tr>
<th>Source</th>
<th>LDAP Profile</th>
<th>Value</th>
</tr>
</thead>
</table>

My First account aware

```xml
<fieldDescription><![CDATA[]]></fieldDescription>
</field>
<field name="host" class="java.lang.String">
  <fieldDescription><![CDATA[]]></fieldDescription>
</field>
<field name="avg" class="java.math.BigDecimal">
  <fieldDescription><![CDATA[]]></fieldDescription>
</field>
<background>
  <band/>
</background>
<title>
  <band height="79">
    <staticText>
      <reportElement x="64" y="14" width="395" height="48"/>
      <textElement textAlignment="Center">
        <font size="24"/>
      </textElement>
      <text><![CDATA[My First account aware report]]></text>
    </staticText>
  </band>
</title>
&pageHeader>
  <band height="35">
    <frame>
      <reportElement mode="Opaque" x="0" y="0" width="555" height="20" backcolor="#6666FF"/>
    </frame>
  </band>
</pageHeader>
<columnHeader>
  <band height="20">
    <staticText>
      <reportElement x="0" y="0" width="153" height="20"/>
      <textElement/>
      <text><![CDATA[Source]]></text>
      <text><![CDATA[LDAP Profile]]></text>
      <text><![CDATA[Value]]></text>
    </staticText>
  </band>
</columnHeader>
```

