SHYFT ANALYTICS

Analytic Platform Developer Guide

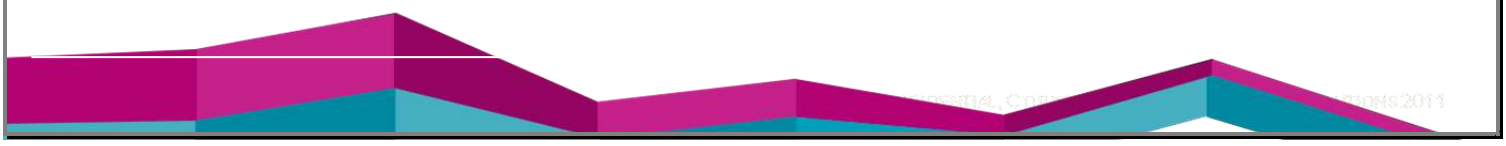


TABLE OF CONTENTS

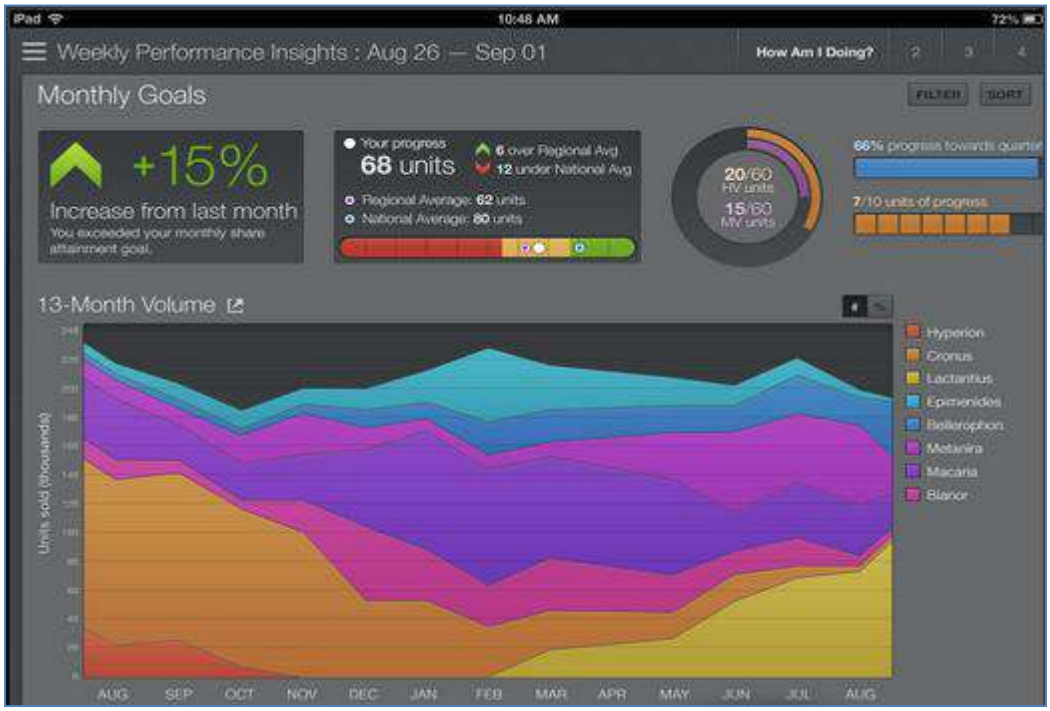
OVERVIEW

Introduction

The Shyft Analytic Platform mobile product is iPad based software that provides pharmaceutical sales analytics.

Beginning in Spring 2011 Shyft began offering a mobile iPad solution in addition to its long standing Excel and Cognos based products. v1 of a mobile offering brought Shyft into the mobile world, significantly enhancing our product line.

Analytic Platform was introduced in 2012 with a much improved user interface. Mobile v2 brings Shyft into a market leadership role in presenting cloud based, mobile, pharmaceutical sales analytics. Following is a screen snap shot.



The purpose of this manual is to give Shyft BI personnel the information necessary to develop v2 mobile reports. It should be studied in its entirety by new Shyft BI personnel and can be used as a reference manual by experienced BI personnel to understand specific development items.

Object Model

The Mobile v2 Object Model provides the foundation upon which the v2 product rests. Understanding the model is fundamental to developing v2 reports.

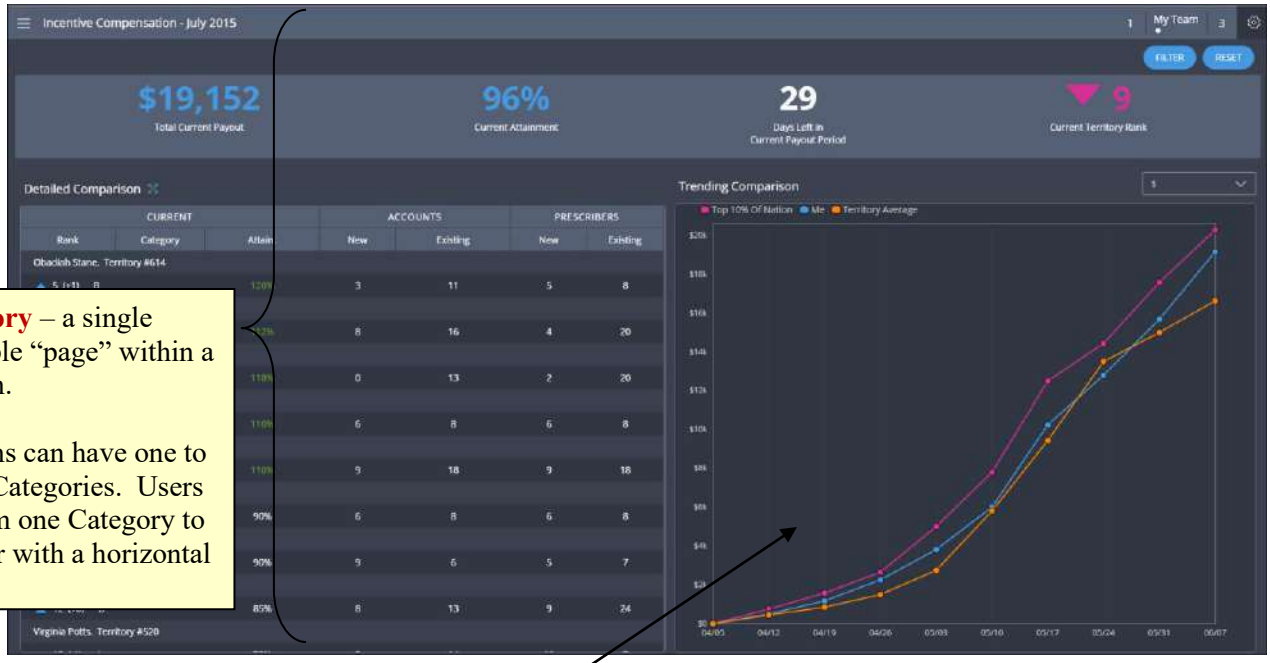
The primary objects within the model are:

1) **Reports**. The overarching object that encompasses the entire product.
2) **Sections**. The primary divisions within a Report.
3) **Categories**. Sub division within a Section. Each "page" a user views on the iPad is a Category.
4) **Containers**. A rectangular box in which the final end user Components are displayed.
5) **Components**. The actual viewable content, for example: Grids, Charts and Freeform objects.
6) **Column Groups**. A grouping of multiple adjacent columns in a Grid. For example, City, State and Zip might be columns in the address Column Group.
7) **Columns**. The individual columns in a Grid Component.
8) **Filters**. Objects that limit the data displayed in Components. Filters are the "where clause" in a SQL statement. In most cases Filters are children of a Category object and thus affect all the Components of a Category. (In certain cases, Filters can be children of Sections).

The following page provides a pictorial representation of the objects.

Report – the root object from which all other objects flow

Section – major divisions within a Report



Category – a single viewable "page" within a Section.

Sections can have one to many Categories. Users go from one Category to another with a horizontal swipe.

Container – that object that holds the final viewable Components. This Container has two Components which are accessed via the drop down button.

Component – Examples include Charts and Freeform Components.

Filter – Objects that support user control over the sub-set of data to be displayed.

Development Stages

The overall development process consists of the following steps:

1) **Design and Mockup creation**. Business Analysts work closely with clients to design the final functionality. The culmination of this work will be the creation of a mockup (usually in the form of PowerPoint slides). BI's are encouraged to comment on the mockup prior to client sign off, paying particular attention to any proposed feature that Mobile v2 does not support.

2) **RPT Table Design**. Those SQL tables that will directly populate the final user interface need to be designed. Such tables are referred to as 'Rpt tables'. The design effort is frequently done by the BI, (though occasionally might also be done by the DB), and must be designed to perfectly match the client accepted final mockup. Once the BI designs a set of proposed Rpt tables the DB should be consulted and tables finalized.

3) **Develop the Mobile v2 user interface**. With the Rpt tables created, work may begin on the actual development of the iPad interface. (Note that the Rpt tables do not need to be populated with final data to begin this effort). This effort is divided into two primary steps: 1) Writing of the stored procedures and 2) Population of configuration tables.

4) **Population of the Rpt Tables**. It is the DB's responsibility to both create and populate the Rpt tables. The DB uses data supplied by both third party vendors and the client to populate the tables.

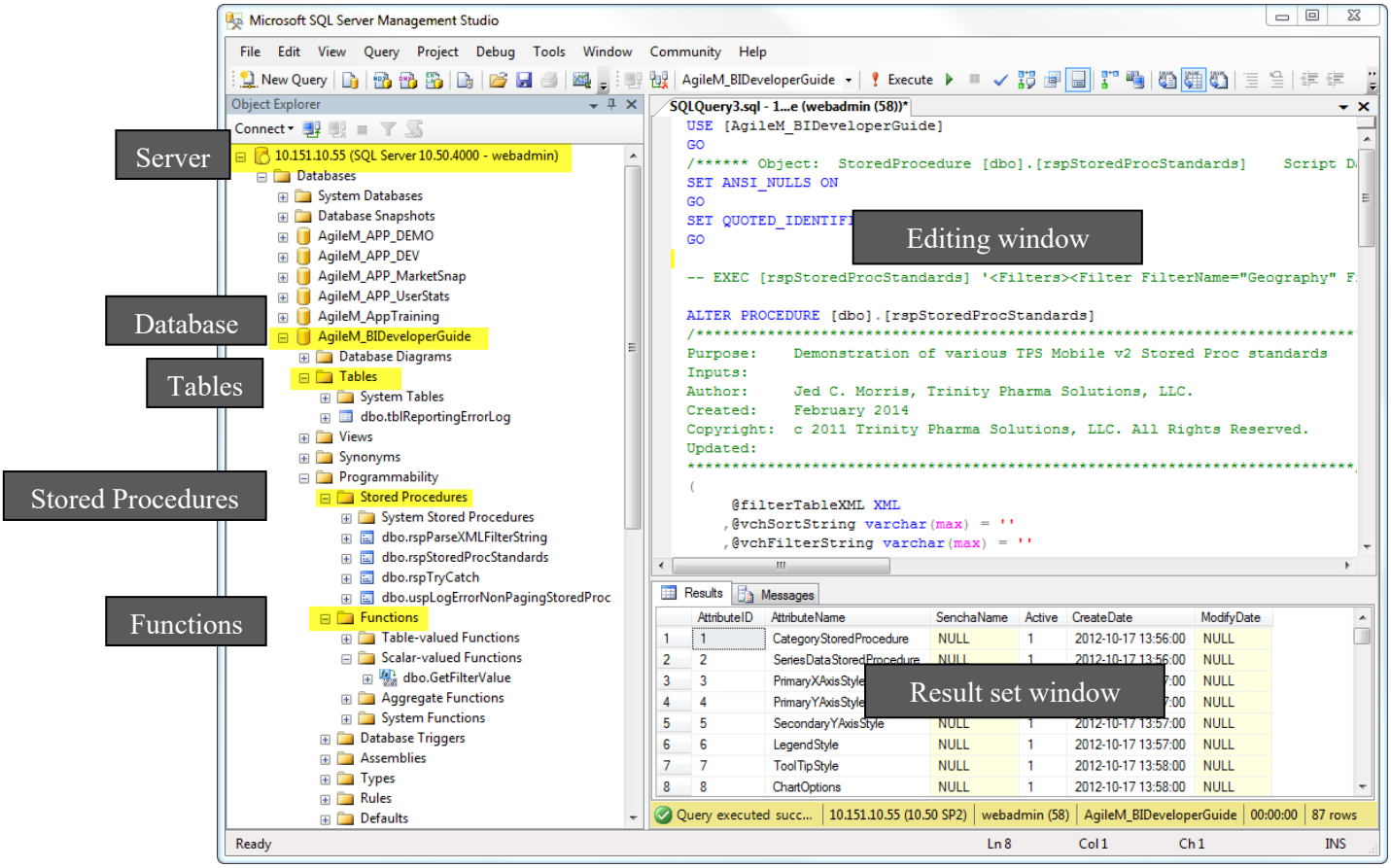
   The remainder of this manual concentrates primarily on the efforts involved in step three – Develop the Mobile v2 user interface.

DEVELOPMENT TOOLS

SQL Server Management Studio

SQL Server Management Studio (SSMS) is a Microsoft Integrated Development Environment (IDE) that provides, in a single interface, a wide range tools for working with SQL databases.

SSMS provides an interface into all objects of the database hierarchy including the primary objects used during Mobile v2 development: Servers, Databases, Tables, Stored Procedures and Functions. Following is a screen snap shot of the interface:



Mobile v2 developers primarily use:

1) **Explorer Window**. The left hand tree structure that presents the database object model.
2) **Editing Window**. This is where stored procedures and queries are written and executed.
3) **Result Set Window**. This is where messages and the results of queries are presented.

SSMS is launched through the Windows Start menu → Microsoft SQL Server program group → SQL Server Management Studio.

AgileM Studio

AgileM Studio is a graphical web based Mobile v2 software development tool designed to populate the two v2 configuration tables - tblCfgReportLayout and tblCfgReportConfiguration.

There are two primary tasks involved in developing v2 reports:

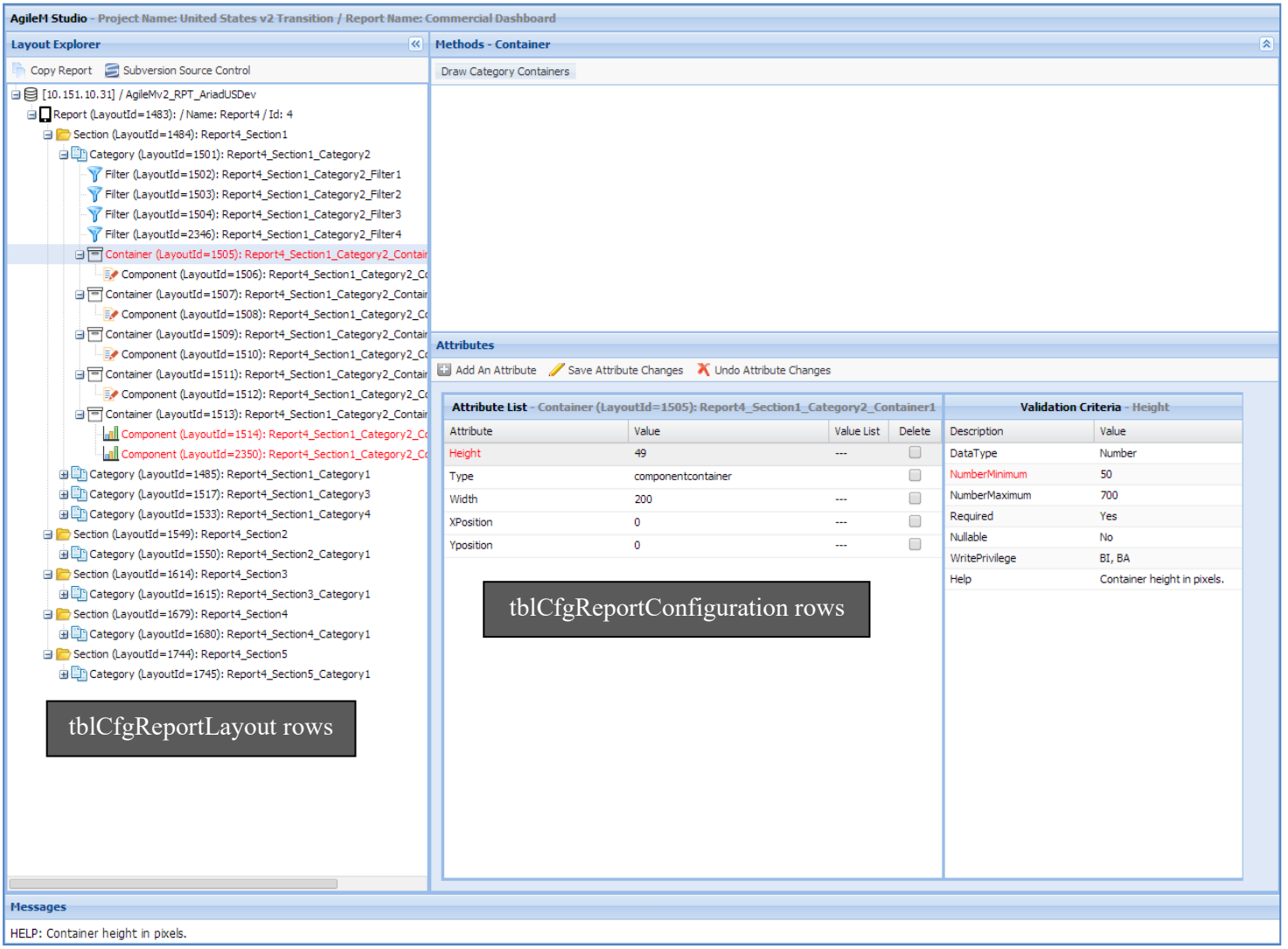
1) Writing the back end stored procs that pull reporting data from rpt tables, and
2) Populating the configuration tables that define the user interface.

AgileM Studio deals strictly with task two – populating the configuration tables.

Access AgileM Studio at: http://10.151.10.239/AgileMStudio/.

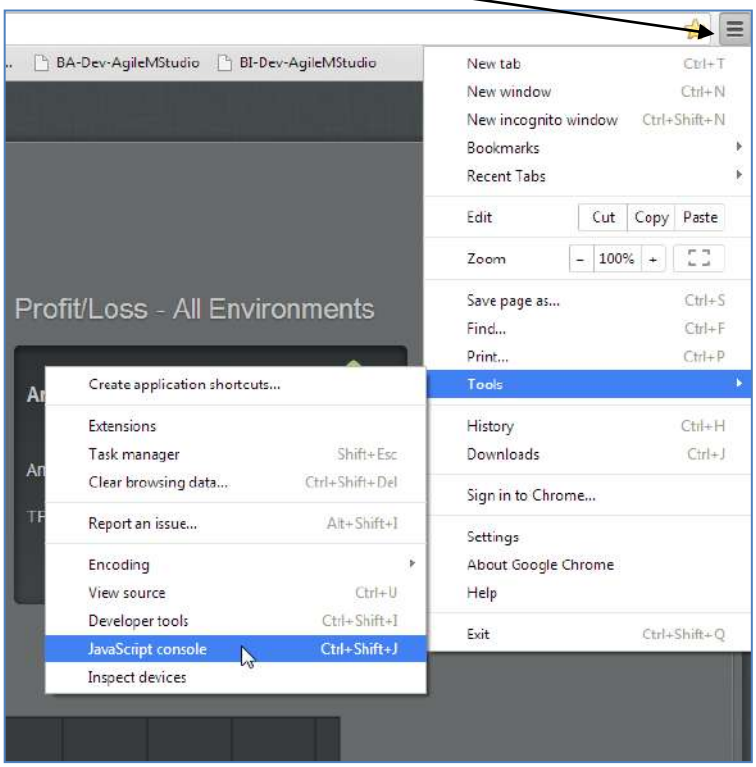
AgileM Studio is fully documented in its own separate user manual. The initial AgileM Studio dialog box presents a button that will download the manual to your desktop. The manual should be reviewed in its entirety.

Following is screen shot of the user interface.



Chrome JavaScript Console

Google Chrome provides a debugger that can offer troubleshooting assistance. From within a v2 Report, right click and select Tools → JavaScript console.

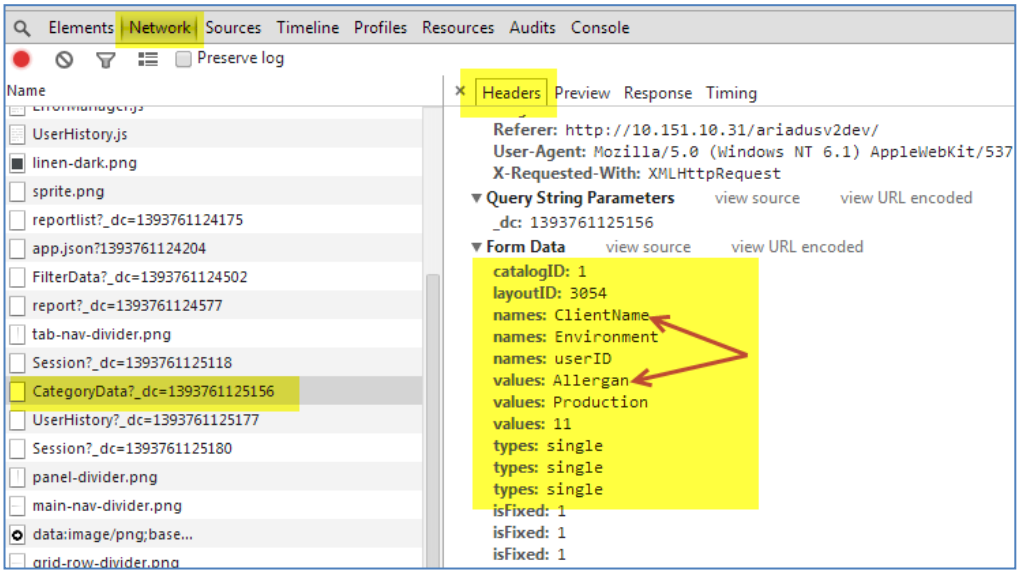


The debugger is quite powerful and most of its functionality is not of particular troubleshooting use to the v2 developer, but some functions that can provide assistance are detailed below.

However, please be aware that troubleshooting, by its very nature, is challenging to document and while the notes below provide some overview it will take using the tools to get insight into their usage.

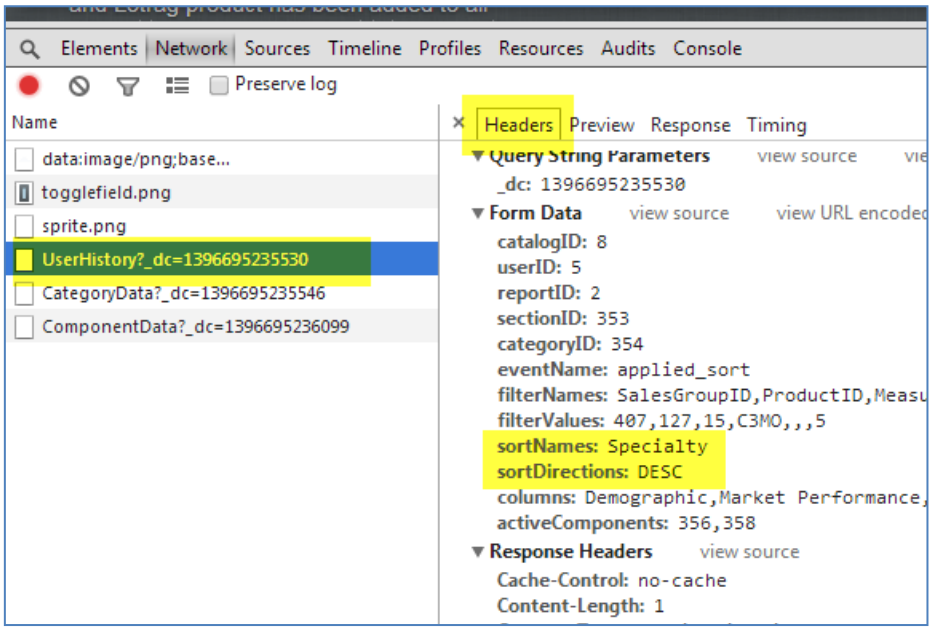
1) Observing filter values:

From within the debugger, select the Network panel, locate CategoryData on the left hand side, the Header tab on the right hand side and then scroll down to Form Data. Note the Name/Value pairs that provide filter information to the stored procs. This will allow you to identify possible Filter issues.



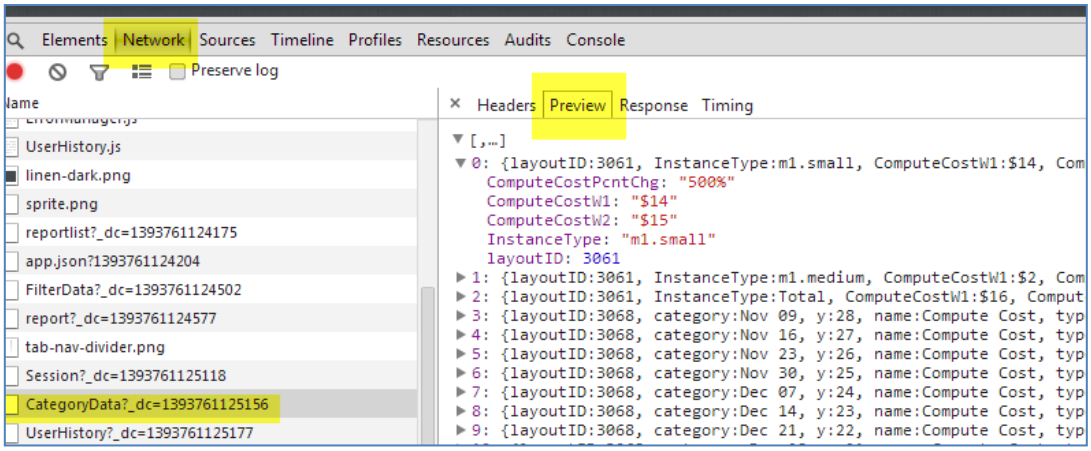
2) Observing sort values:

From within the debugger, select the Network panel, locate UserHistory on the left hand side, the Header tab on the right hand side and then scroll down to Form Data. Note the sortNames/sortDirections pairs that provide filter information to the stored procs. This will allow you to identify possible Sort issues.



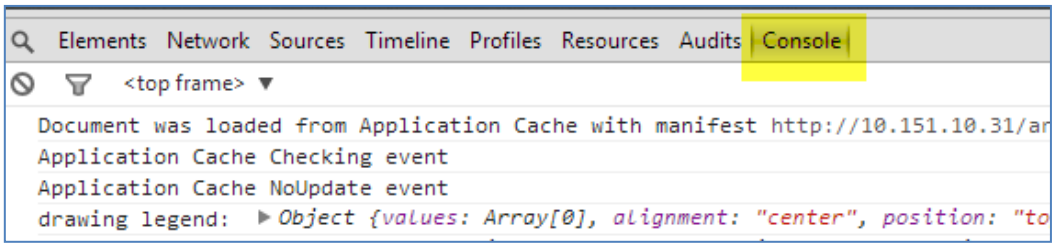
3) Observing stored proc return data result sets:

Select the Preview tab to observe the data that has been returned from a stored procedure.



4) Console error messages:

Select the Console tab to see general purpose error messages which occasionally provide insight.



Amazon EC2

Shyft uses Amazon Web Services (EC2) as the provider of its cloud based disk services. As of December 2013 Shyft transitioned all its internal work and client facing products to EC2. Mobile v2 developers are insulated from needing to know how such serves are established or maintained.

However, it is helpful to understand that:

1) The URL for all our disks have the same IP address prefix of: '10.151.10'. Typical full addresses will then be: 10.151.10.32 or 10.151.10.44. So, when logging into a SQL server or making a remote desktop connection you would enter the IP address as:

9) Clients will have up to three servers dedicated to their own use with each server will have its own 10.151.10.x IP address:

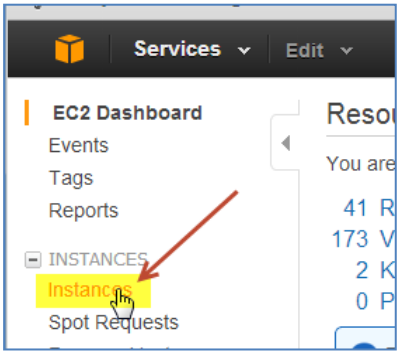
   a. Development (where all the Mobile v2 implementation work is done by BI personnel)

   b. Processing

   c. Production (where clients actually access their iPad reports).

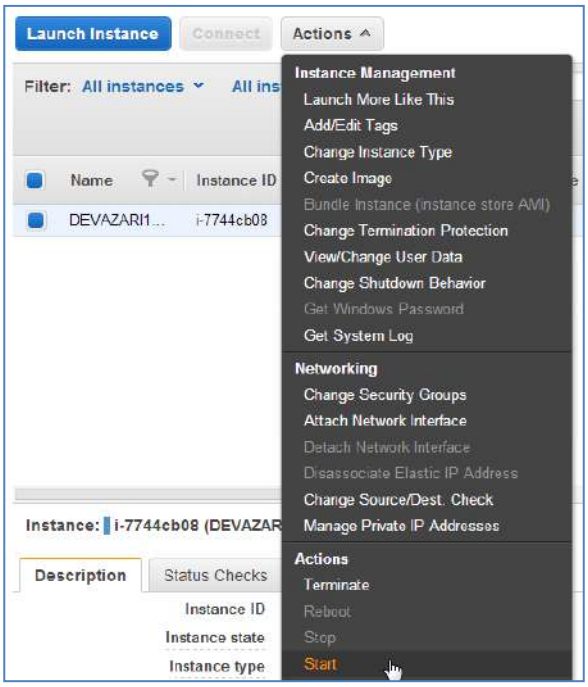
10) Production servers are available 24 * 7 every day. To minimize the cost of these servers, development machines will generally be turned off in the evening. When you find that a server is not responding, (for example, first thing in the morning), you may need to start the service as follows:

- Logon to Amazon Web Services (you'll be provided the URL and a username and password).
- Select Instances.



- Enter the IP address and then select the Instance.

- In the Actions drop down menu, select Start (being careful to **not** select Terminate). The service generally takes a few minutes before it is fully available.



SVN

Apache Subversion (SVN) is an open source version control system which was adopted by Shyft in late 2013. Business needs met by SVN are:

1) **Historical Versioning**. To provide a mechanism for tracking historically changes in software.

2) **Backup**. To provide a repository where software that is actively being worked on and selected data can be restored from if lost.

3) **Software Release**. To provide a single point from which software can be released from. For example, rather than copy a Development environment to a Production environment, software is first placed in SVN and then released from SVN to Production. (This objective has not been fully realized yet).

This manual will limit the discussion of SVN usage to placing SQL objects (stored procedures, functions, etc) into SVN. The general process is:

- From within SSMS, generate scripts that create objects (stored procs, functions, etc.) and then write the scripts to your local hard disk SVN folder.
- Using SVN, commit the scripts to their central repository.

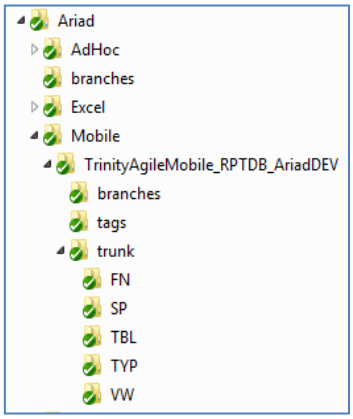
Installation:
Each machine must install SVN which is located here: \\filesrv01\TPSData\TPS_Softwares\SVN

Execute the file: TortoiseSVN-1.8.0.24401-win32-svn-1.8.0.msi

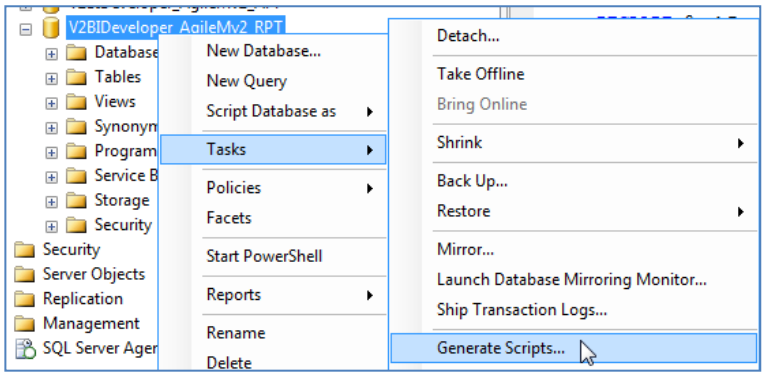
One time set-up:
Following are one time set-up steps:

1. Get a user name and password from the SVN administrator.
2. Create a folder on your machine which will act as your local copy of SVN (generally, c:\Projects\SVN_BITeam).
3. Right click in that folder and select "SVN Checkout …"
4. Under URL of Repository enter: https://10.151.10.53/svn/BITeam
5. Files will be downloaded to your hard drive.

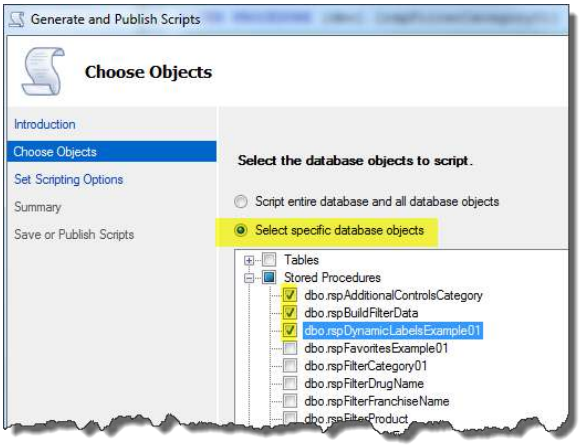
Background:
The SVN folder hierarchy must be consistent across all BI members. The following folder structure has been established. For each client, please create folders as below (SP = Stored Procedures, FN = Functions, etc):



Generate Scripts:
From within SSMS, right click a database → Tasks → Generate Scripts…

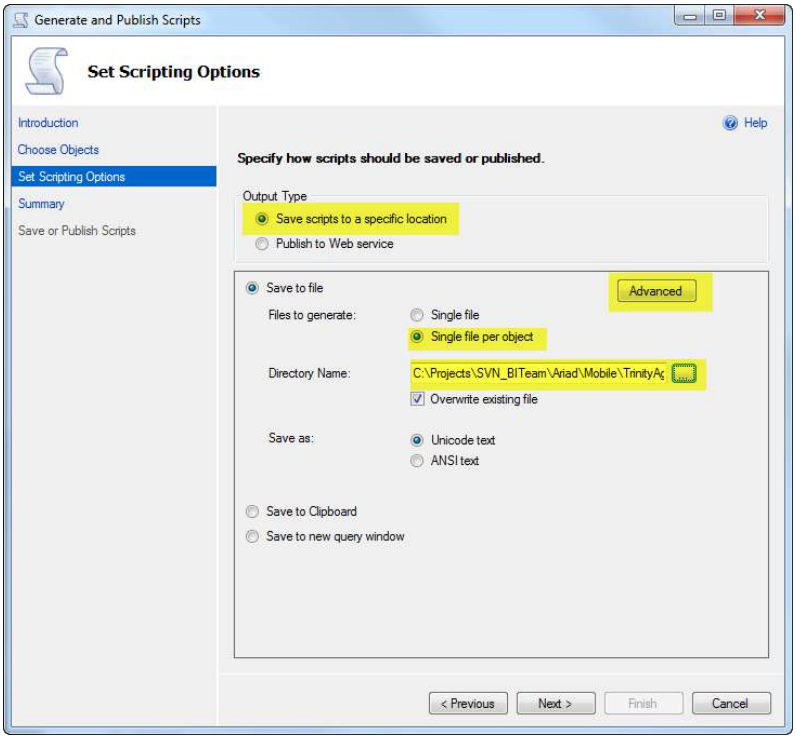


Select the button: Select specific database objects. Then chose a single category, such as Stored Procedures - do not select more than one object type. Select specific items or all the items in an object type:

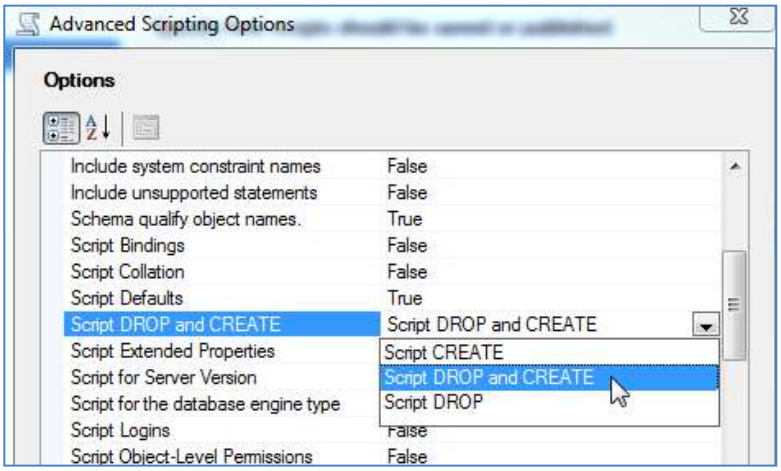


In the following dialog box select:

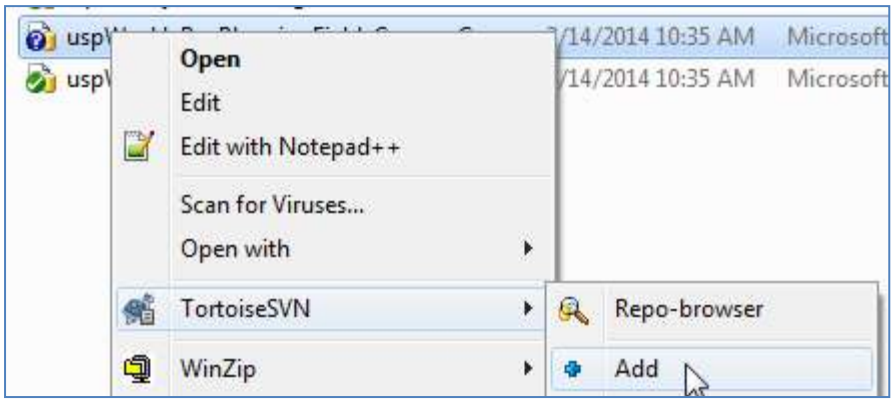
- Save scripts to a specific location
- Single file per object
- The correct directory name (following the folder hierarchy as noted above)



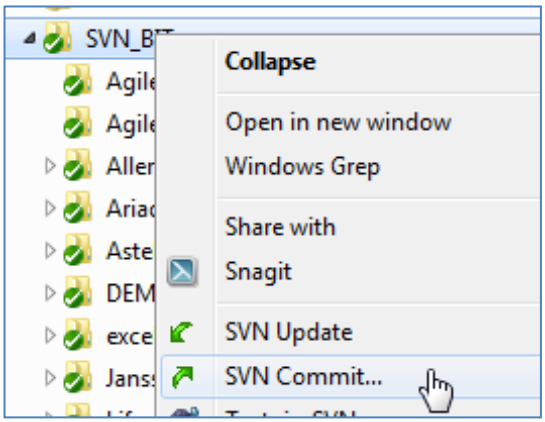
Also, click Advanced and take SCRIPT DROP and CREATE (this will add a SQL Drop statement prior to the Create statement). Then continue with Next> (twice) and the scripts will be saved in your local SVN directory.



Commit scripts to SVN:
Add the scripts by right clicking the scripts TortoiseSVN → Add.

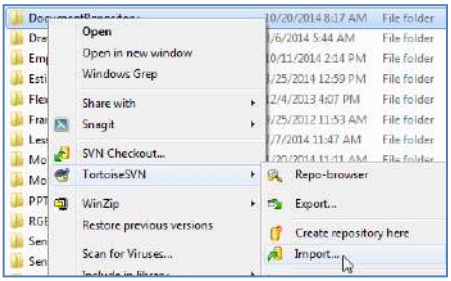


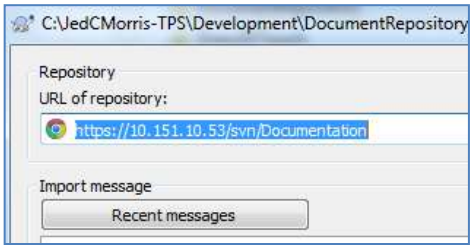
After the scripts are added, then commit the changes. Right click the folder SVN_BITeam → SVN Commit.



Accessing departmental documents:
Departmental documentation is stored on SVN and is available for download. To access these documents:

1) Create a new folder in a location and with a name of your choosing.
2) Right click the folder and select TortoiseSVN→ SVN Checkout….
3) Select the URL of repository as: https://10.151.10.53/svn/Documentation. Click OK and the folder structure will be downloaded.





SQL Compare and SQL Data Compare

SQL Compare and SQL Data Compare, both by Redgate, are tools designed to copy SQL database objects from one database to another. This is commonly done, for example, to copy changes in development to production. For some projects DB's will perform this responsibility, but it is not uncommon for the BI to also perform the task of updating Processing or Production with the Development environment.

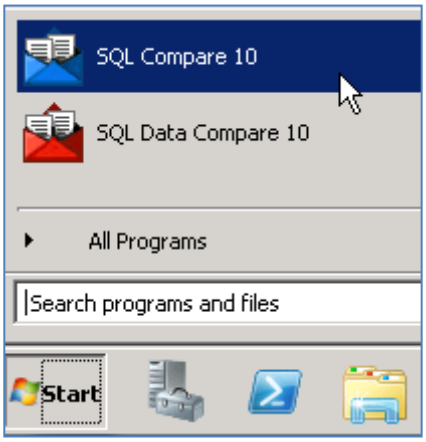
SQL Data Compare deals only with the data that is in SQL tables. SQL Compare deals with all other database objects (stored procedures, functions, user defined types, etc.).

Note that when updating a database, SQL Compare must be run prior to SQL Data Compare. This is because table structure must be equivalent before data can be copied. For example, if a new column has been added to a table, (i.e. the table structure has changed), then the tables structure must be identical before data can be moved.

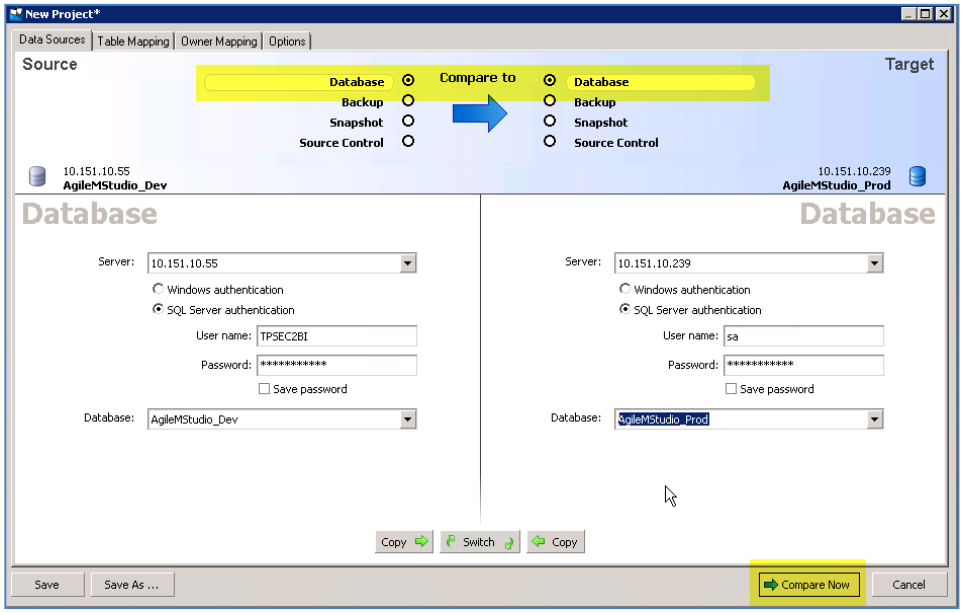
Following are steps to run SQL Compare. (Steps to run SQL Data Compare are virtually identical).

Log on to tpstermserver:
Using Remote Desktop Connection, log on to tpstermserver with your personal username.

Launch SQL Compare 10:
Launch SQL Compare 10 from the Start menu.



Select the Source and Target databases:
Select the Source and Target server and database. Click Compare Now.



Select the objects to update:

SQL Compare compares objects in the Source and Target databases to determine differences. In most cases, the SQL objects the BI Developer will be primarily interested in are stored procedures and tables. A display like the following will appear:

Items to pay particular attention to are:

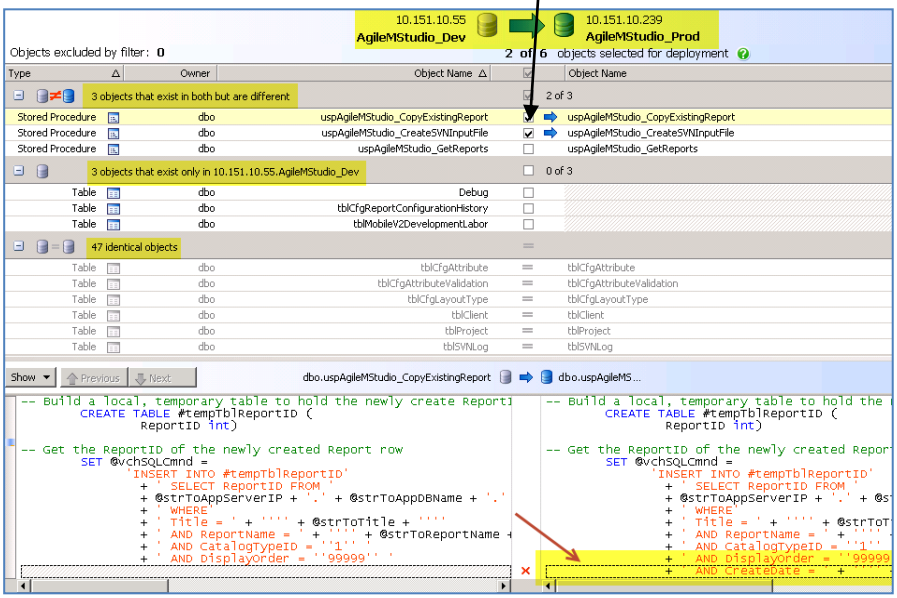
1) **Correct Source and Target database**. At the top of the display, confirm that the correct databases are being compared.

2) **Objects that exist in both but are different**. This section will list objects that have differences in the databases. Note that when selecting an object, the window at the bottom of the screen will provide a comparison of the two stored procs, noting the exact lines of code where there are differences.

   You will need to carefully study this list and select those objects that should be moved. *Note that you should **not** move objects that are the primary responsibility of the DB. The DB will move such objects*.

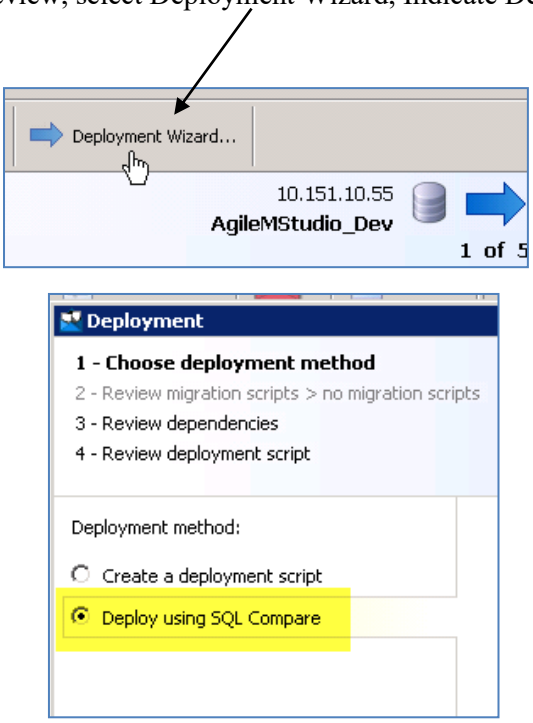
3) **Objects that exist only in the Source database**. Those objects that are unique to the Source database are listed here. In some cases you will want to select these objects to be copied, but in other cases it is appropriate to have an object only in the Source.

4) **Identical objects**. There will be not be any need to copy these objects from the Source to the Target.



Deployment Wizard:

After careful review, select Deployment Wizard, Indicate Deploy using SQL Compare and click Next.

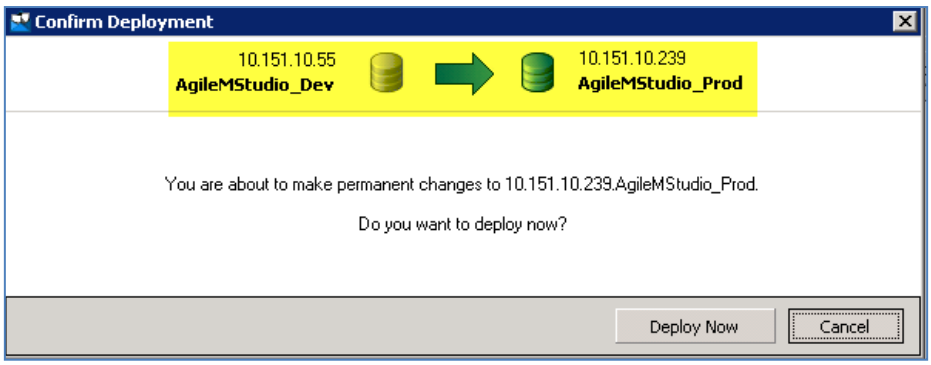


Deployment:

On the next screen, click Deploy Now.

Confirm Deployment:

On the final prompt, carefully review which data base will be permanently changed. **Take a deep breath. An update to the incorrect database is irrecoverable**.



Final Note

Note that if you push over tblCfgReportConfiguration and/or tblCfgReportLayout that you will need to manually execute uspCreatReportConfiguratinHoz. If you push over filter data value changes you will need to manually execute rspBuildFilterData.

TECHNOLOGIES

SQL

SQL is a programming language designed for managing data held in relational databases. The scope of SQL includes the ability to create and manage database objects (schemas, tables, views, functions, etc.) and the data in the database (insert, update, delete, etc.).

BI Developers will primarily be concerned with writing SQL statements and combining them into a sequentially executed series of statements called stored procedures.

This manual will not make any attempt to provide SQL training. Full SQL training is beyond the scope of this manual and has been well documented in scores of printed and on-line books. Also, Shyft makes SQL on-line training available to BI personnel here: L:\TPS_BI\TrainingVideos.

Dot Liquid

Dot Liquid is a technology that assists in the formatting of data and is particularly useful in formatting the contents of a Free Form component.

This Dot Liquid discussion will be divided into two sections. The first section provides a detailed example of one template. The second section will discuss how to work with templates.

Dot Liquid Template Example

Think of Dot Liquid as a "mini-programming language" that supports variables, if statements, looping and formatting masks. The following example demonstrates these features.

APDeveloper Report 01:

Section: Free Form
Container: Free Form Example 02
Stored Proc: rspFreeFormExamplesCategory01
TemplateID: 3

```
{% assign growthColor = 'rgb(139, 185, 79)' %}
{% assign declineColor = 'rgb(207, 78, 61)' %}
{% for d in this.array %}

<table class="freeform-tbl" style="width: 100%">
  <tr style="height: 70px;">
    <td style="vertical-align: top; text-align: left; padding-right: 0px;">
      <p style="font-size: 22px; font-weight: bold">Percent Chg</p>
    </td>
    <td style="{% if d.ComputeHoursPcntChg > 0 %} color: {{ growthColor }};
               {% elsif d.ComputeHoursPcntChg < 0 %}  color: {{ declineColor }};
               {% endif %}
               vertical-align: top; text-align: right; font-size: 33px; font-weight: bold; padding-right: 0px;">
               {{ d.ComputeHoursPcntChg | Number: '0.#%;-0.#%;0%' }}

               <span class="icn main-progress
               {% if d.ComputeHoursPcntChg > 0 %} growth
               {% elsif d.ComputeHoursPcntChg < 0 %} decline
               {% endif %}">
               </span>
    </td>
  </tr>

  <tr style="height: 50px;">
    <td style="vertical-align: middle; text-align: left; font-size: 14px;">Current Week:</td>
    <td style="vertical-align: middle; text-align: center; font-size: 14px;">{{ d.ComputeHoursW1 | Number: 'N1' }} hours</td>
  </tr>

  <tr style="height: 50px;">
    <td style="text-align: left; font-size: 14px;">Prior Week:</td>
    <td style="text-align: center; font-size: 14px;">{{d.ComputeHoursW2 | Number: 'N1'}} hours</td>
  </tr>

</table>
{% endfor %}
```

The above Dot Liquid template will output the Free Form object to the left.

The white borders for the rows and columns are displayed for demonstration purposes – the actual Free Form will not have these borders. Add the borders by temporarily specifying your '<table' as '<table border=1'.

```
{% assign growthColor = 'rgb(139, 185, 79)' %}
{% assign declineColor = 'rgb(207, 78, 61)' %}
```

In this code section variables are being assigned rgb color values. The variables will be used later in the template.

```
<table class="freeform-tbl" style="width: 100%">
  <tr style="height: 70px;">
    <td style="vertical-align: top; text-align: left; padding-right: 0px;">
      <p style="font-size: 22px; font-weight: bold">Percent Chg</p>
    </td>
```

Free Forms are displayed as HTML tables which consist of the HTML tags: <table> (the entire Free Form will be one table), <tr> (a table consists of rows), and <td> (rows consist of cells).

Here the table is being established, the first row is created with styling information that says the full row will be 70px in height, and the first cell, which will contain the text Percent Chg is created.

</td> ends the cell.

```
<td style="{% if d.ComputeHoursPcntChg > 0 %} color: {{ growthColor }};
           {% elsif d.ComputeHoursPcntChg < 0 %}  color: {{ declineColor }};
           {% endif %}
```

This code section begins to use the actual data that is supplied by the stored proc (which is shown below):

	layoutID	HTMLTemplate	ComputeHoursW1	ComputeHoursW2	ComputeHoursPcntChg
1	502	{% assign growthColor = 'rgb(139, 185, 79)' %}{%...	6250.0000	6463.0000	-0.0330

Note the stored proc returns a column named ComputeHoursPcntChg and the template makes reference to d.ComputeHoursPcntChg.

Also note the use of a traditional programming IF statement. The template is saying: “if d.ComputeHoursPcntChg > 0, then style = color growthColor else style = color declineColor”.

```
{{ d.ComputeHoursPcntChg | Number: '0.#%;-0.#%;0%' }}
```

This code section uses the concept of individual masks separated by semi-colons (for example, 0.#% is one mask). d.ComputeHoursPcntChg will be formatted with the mask ‘0.#%’ if greater than zero, with the mask’-0.#%’ if less than zero and with the mask ‘0%’ if equal to zero.

```
<span class="icn main-progress
{% if d.ComputeHoursPcntChg > 0 %} growth
{% elsif d.ComputeHoursPcntChg < 0 %} decline
{% endif %}">
</span>
```

This code section uses the concept of CSS classes and is responsible for the output of the large red Chevron (down facing arrow). That is, a class = "icn main-progress decline" will be a large red down facing Chevron, while a class = "icn main-progress growth" will be a large green up facing Chevron.

```
  </td>
</tr>
```

This section of code ends the first row.

The remainder of the template repeats the concepts noted above.

Also, looping is available in Dot Liquid and supports the processing of multiple rows of data in the record set. While not used in this particular example, (since there is only one record in the record set), the following lines of code are what implements looping.

```
{% for d in this.array %}
    <full template code>
{% endfor %}
```

Working with Templates

Background

Agile Mv2 uses the following tblCfgDLHtmlTemplate table to store the actual templates:

```
SELECT TOP 1000 [TemplateID]
      ,[Template]
  FROM [V2BIDeveloper_AgileMv2_RPT].[dbo].[tblCfgDLHtmlTemplate]
```

Results | Messages

TemplateID	Template
1	{% for d in this.array %} <table class="freeform-tbl" style="width: 100%;"> <tbody style="text-align: cen...
2	{% for d in this.array %} <table class="freeform-tbl" style="width: 100%;"> <tbody style="text-align: left;...
3	{% assign growthColor = 'rgb(139, 185, 79)' %}{% assign declineColor = 'rgb(207, 78, 61)' %}{% for d in t...

Note how the stored procedure reads in the template using TemplateID.

```
--Component: Free Form Example 02
    SELECT
          layoutID = 502
         ,HTMLTemplate = (Select Template FROM tblCfgDLHtmlTemplate WHERE TemplateID = 3)
         ,ComputeHoursW1
         ,ComputeHoursW2
         ,ComputeHoursPcntChg
    FROM  tblRptComputeCostHistory
```

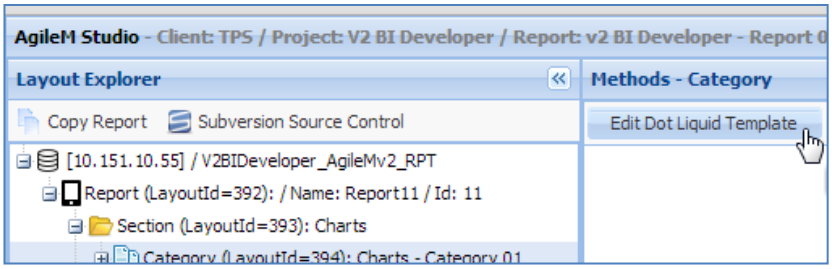
There are two methods for working with Dot Liquid templates. Method 1 below is far easier and should always be used by default. Method 2 is presented below in the event Method 1 has an issue with a particular template.

Additional Dot Liquid Resources

- https://github.com/Shopify/liquid/wiki/Liquid-for-Designers
- http://havanacommerce.net/en/help/templates.syntax
- http://msdn.microsoft.com/en-us/library/0c899ak8(v=vs.100).aspx
- http://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.100).aspx
- Freeform Configuration.docx (located in the SVN Document Repository at: Framework → AgileMv2).

Working With Templates – Method 1

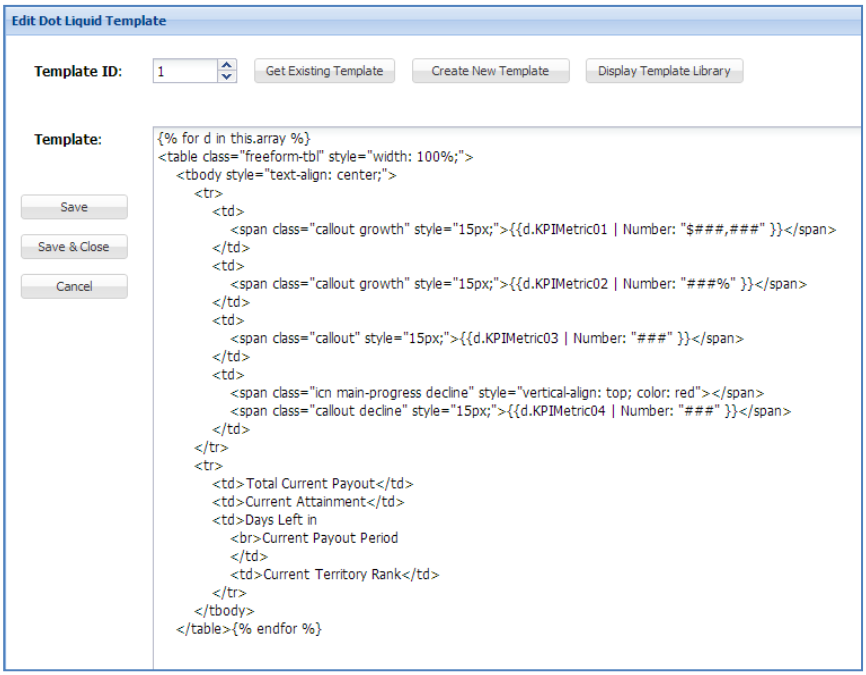
From within AgileM Studio elect the Edit Dot Liquid Template button.

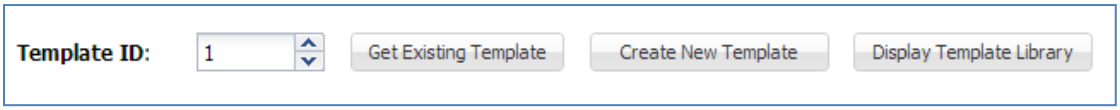


The following dialog box appears. Enter a TemplateID and click Get Existing Template. Edit the template and Save.

Note that after saving a template, and then observing that the changes were not successful, Ctrl-Z can undo changes.

Also note that dot liquid templates must not use a single tick mark for string delimitation – use a double quote instead. If a single tick mark is used, then AgileM Studio will not be able to save the template.



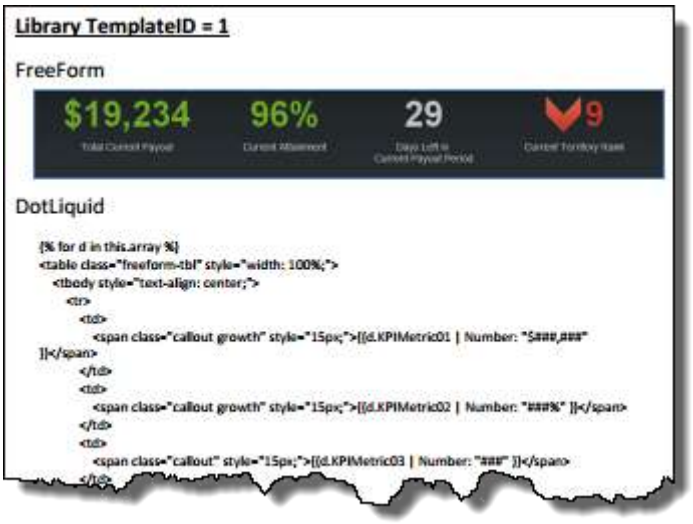


Also note the additional buttons Create New Template and Display Template Library.

Display Template Library

Note that a library is maintained with a variety of Dot Liquid templates used to display standard Free Form objects. In many cases an existing template, in the library, will be similar or identical to your current needs.

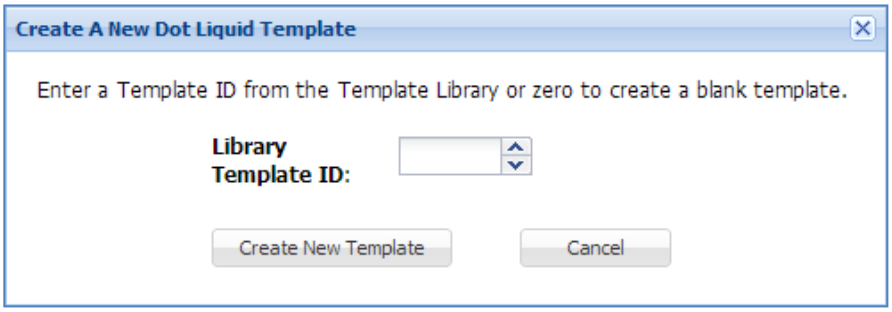
The Display Template Library button will show a .pdf file with many examples of standard Free Form objects.



Scroll through the file to determine if an existing library template is similar to your needs.

Create New Template

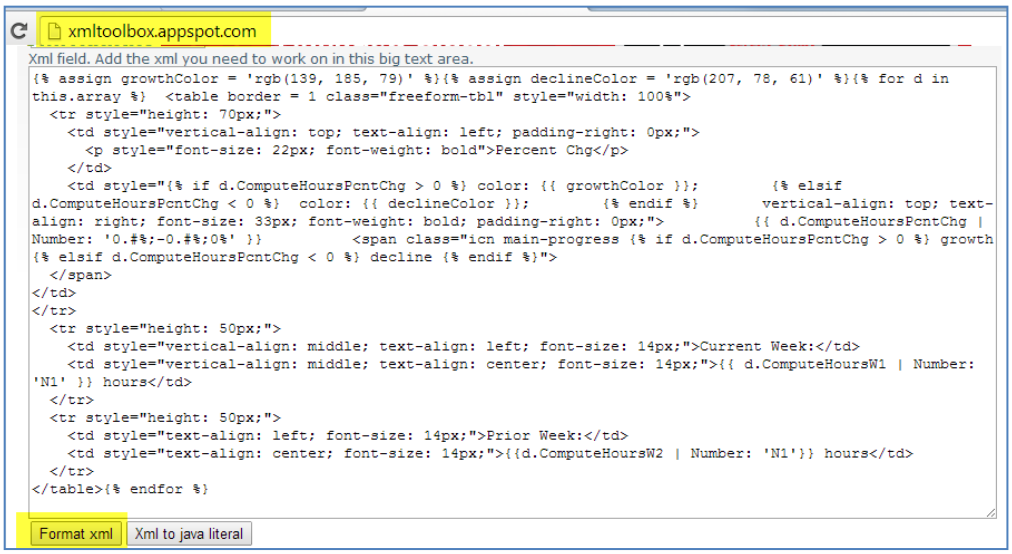
This buttons allows you to create a new Dot Liquid template. The new template can be a copy of a template from the template library (explained above) or a blank template.



Working With Templates – Method 2 (used only if Method 1 above fails)

Step 1:

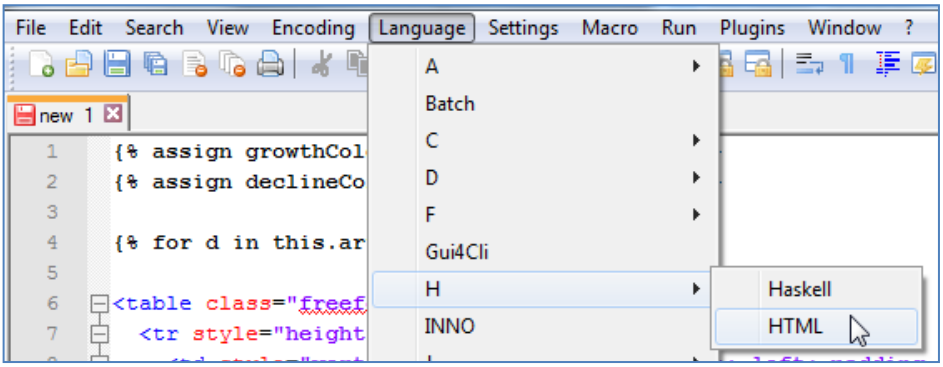
Copy the single long string into a XML formatter. There are many on the web including the following: http://xmltoolbox.appspot.com/. Paste the string into this site, click FormatXML and the single long string will be presented into a well formatted, easy to read, display.



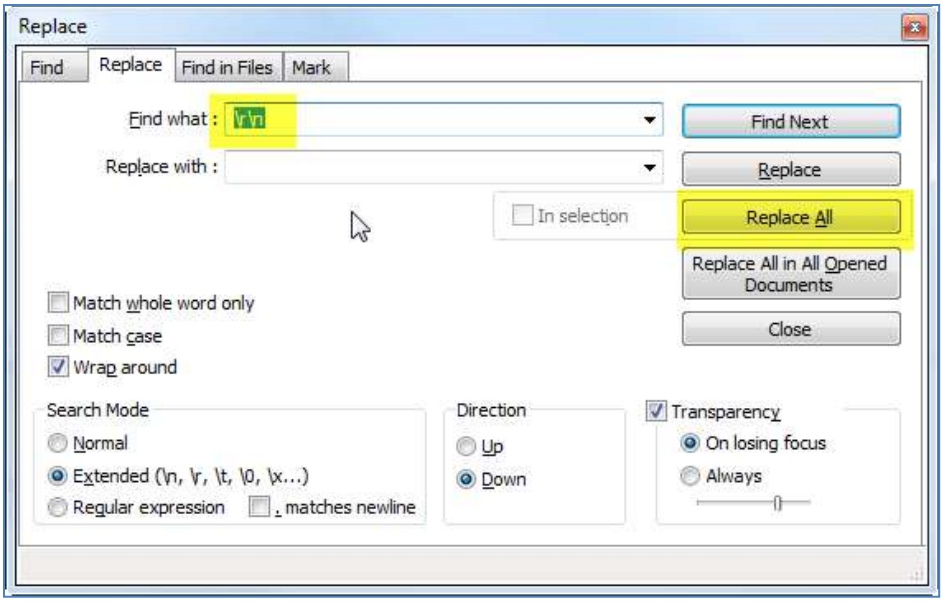
Step 2:

Copy the well formatted result into Notepad++ (available here: \\tpsfile01\TPSData\TPS_Softwares\ npp.6.5.4.Installer.exe).

Within Notepad++ select Language → H → HTML which will add tag colors for easier reading.



Step 3:
Work with the template. When you are ready to return the template back to SQL, Search → Replace, and then Replace All '\r\n' with nothing. Copy/Paste the result back to SQL.



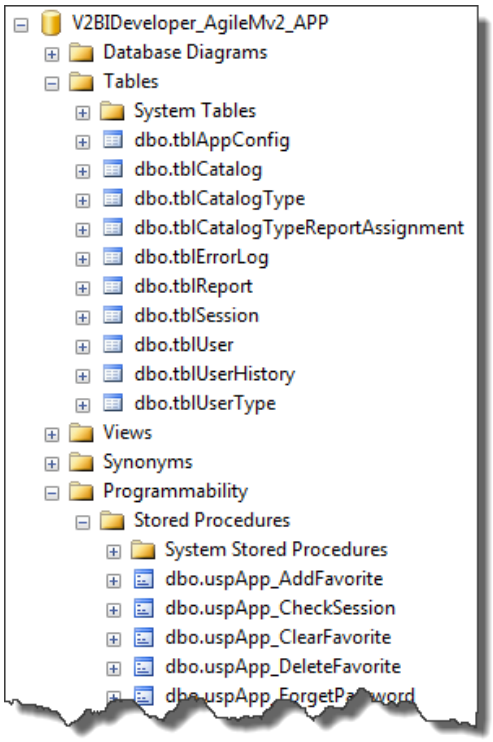
ARCHITECTURE

The Analytic Platform mobile product has database and application components. Each is described below.

Databases

Two databases, each residing on the same server, support the v2 mobile product. The application database contains generic tables that are common to all v2 databases, (which the BI Developer will have limited responsibility for), and common stored procs, (which the BI Developer will have no responsibility for).

Application Database



This is the application database whose tables and stored procedures are common to all v2 application databases. Note that the database name contains the text 'APP'.

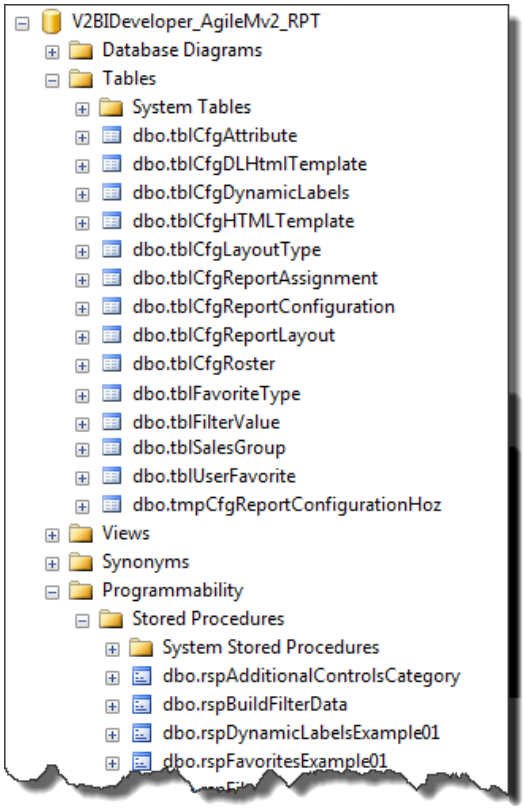
The BI Developer will establish the existence of users and reports in the APP tables, but will not create any tables.

The BI Developer will not have any responsibility for the stored procs in the App database.

Following are brief comments on selected App tables.

Table Name	Usage
tblCatalog	Provides a mapping of a Project to its reporting database name located on the same server.
tblErrorLog	A logging, by the v2 application, of runtime errors. Occasionally used by the BI Developer.
tblReport	One row for each report. The primary purpose is to establish a unique ReportID and ReportName.
tblSession	Used by the v2 application to establish a unique GUID. Not used by the BI Developer.
tblUser	Contains user information: a unique UserID, Name, Username, Password and e-mail address.
tblUserHistory	Contains user usage history. Occasionally used by the BI Developer during debugging.

Reporting Database



This is the reporting database. Note that the database name contains the text 'RPT'.

The tables in the reporting database fall under the following two categories:

1) The tables pictured here that are common to all v2 reporting databases – some of which the BI Developer will be concerned with.

2) A large number of 'rpt' tables which are: not pictured here, whose name begins with 'tblRpt', are originally populated by the database developer, and are the primary source of data for the front end. The BI Developer will do most of their work with these so called 'rpt' tables.

Following are brief comments on the common tables that BI Developer will have responsibility for

Table Name	Usage
tblCfgAttribute	Provides a listing of all Attributes.
tblCfgDLHtmlTemplate	Provides formatting information for Free Form objects.
tblCfgDynamicLabels	Provides the ability for dynamic Grid headers.
tblCfgLayoutType	Provides a listing of all the Layout types.
tblCfgReportAssignment	A mapping of a report to a user.
tblCfgReportConfiguration	A fundamental table that establishes the behavior of front end objects. The content of this table is maintained primarily by AgileM Studio, but the BI Developer should be very aware of this table's data.
tblCfgReportLayout	A fundamental table that establishes the existence of front end objects. The content of this table is maintained primarily by AgileM Studio, but the BI Developer should be very aware of this table's data.
tlbFilterValue	Controls the content of front end filters.
tblUserFavorite	Supports the front end functionality that allows a user to indicate that, for example, a particular physician should be listed in a Grid.

V2 Mobile Application

The v2 Mobile application consists of three primary parts:

1) **IIS web site**. Users are provided a URL to this web site which provides full control over the user experience. This site is written in Sencha Touch (an advanced JavaScript framework particularly well suited to multi-device development), JavaScript, HTML5 and CSS. The BI Developer has no responsibility for the original development or maintenance of this site.

2) **Server based web API**. Residing on the same server as the IIS web site, this API sits between the IIS web site and the mobile databases. The IIS web site makes calls to this API which then instructs the database to execute stored procedures. This API is written in C#. The BI Developer has no responsibility for the original development or maintenance of this API.

3) **Mobile databases**. Actually, as noted above, two SQL databases – an APP and RPT database. The web API issues commands, along with required input parameters, to execute a stored procedure. The stored procedure returns result set(s) to the API which in turn passes the data to the IIS web site. The primary responsibility of the BI Developer is to write the RPT stored procedures.

INSTALLATION

There are three primary tasks to perform when setting up a new Analytic Platform system:

1) Create the SQL application (APP) and reporting (RPT) databases.
2) Install the actual mobile v2 application.
3) Establish the mobile v2 web site.

For new installations you will need to perform the primary steps 1-3 below. For upgrades, perform parts of step 1 below, and then skip to the section titled: Application Upgrades.

Special Note: If upgrading from a pre-2.4.0 release then also follow the instructions in step 2.

1-Create/Update the SQL App and Rpt databases

The mobile v2 application requires two databases. They are referred to as the APP database and the RPT database.

When initially creating the databases perform the following all steps 1-7 below. When updating an existing database follow only steps 4-6 below.

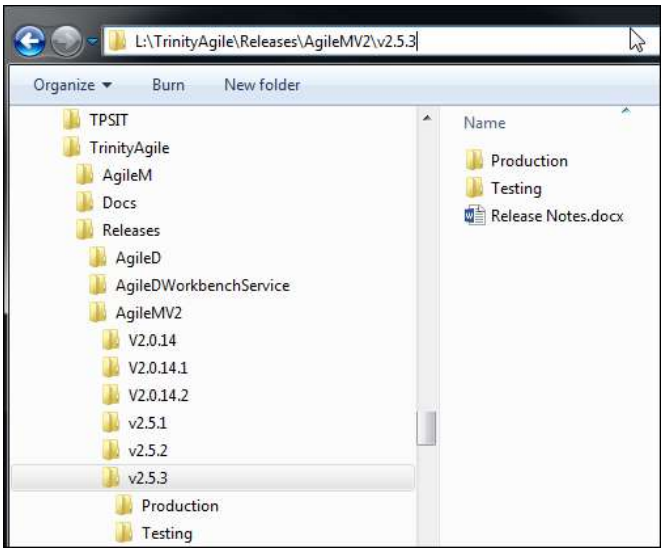
1) Select a project name. This name should be selected very carefully since it will be used in multiple steps during the installation, multiple directories will carry this name and it will be used by AgileM Studio. Items to consider:
    a. There is no need to include the client name since the project will be restricted to a single client server.
    b. Projects consist of one to many reports so do not limit the name to a specific report.
    c. The name should have a "client centric" view point. That is, determine a project name the client can relate to and encompasses the overall client business value of a collection of reports.
    d. The BA should be consulted.
    e. Changing a name is time consuming and error prone. Get it correct the first time.

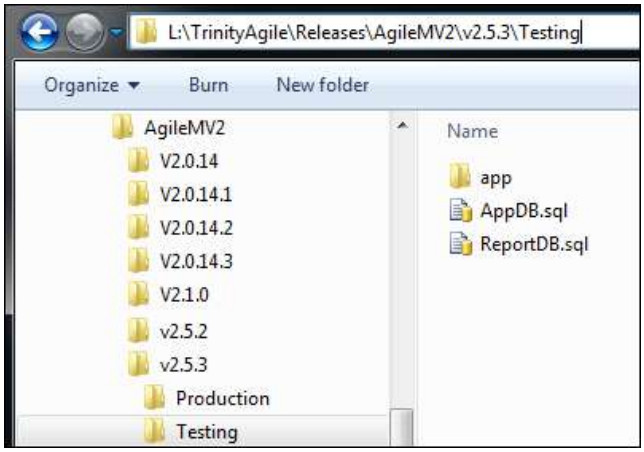
2) Using SQL Server Management Studio, logon to the selected client server environment. (Note that we always place the v2 data bases on 10.151.10.x servers).

3) Create two new databases. Name one <Project name>_AgileMv2_APP and the other database <Project name>_AgileMv2_RPT. On occasions when multiple RPT databases point to one APP database this naming convention is slightly modified, but otherwise do not stray from this standard.

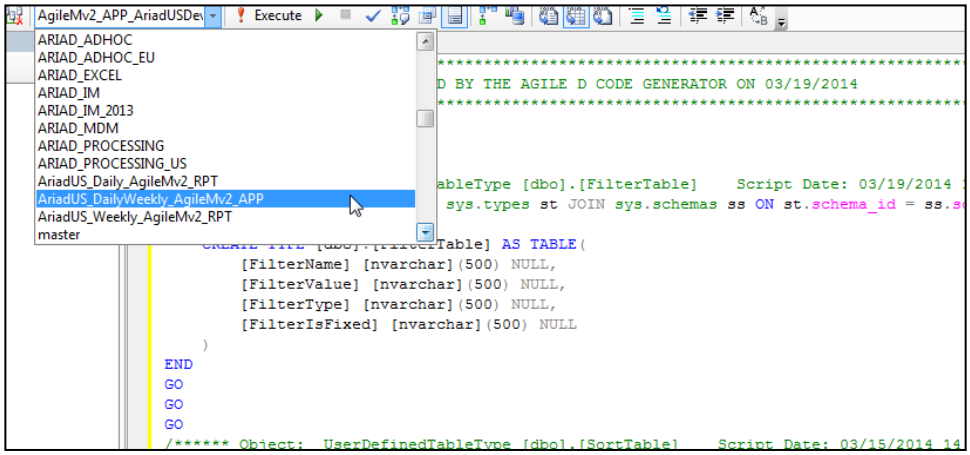
4) DB build scripts, named AppDB.sql and ReportDB.sql, have been created for both the APP and the RPT databases and are located in the following folder:

   L:\TrinityAgile\Releases\AgileMV2\<version number>



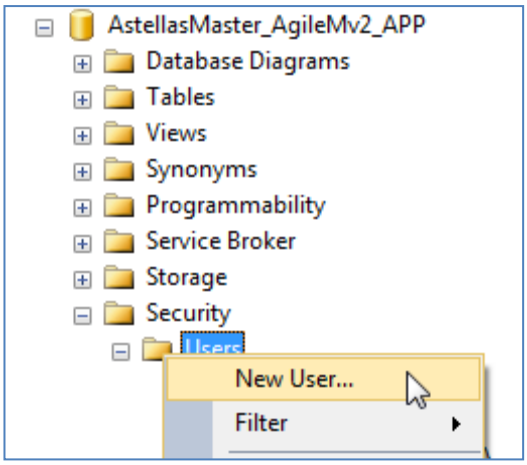


5) Open the AppDB.sql file and execute the App version against the APP database.

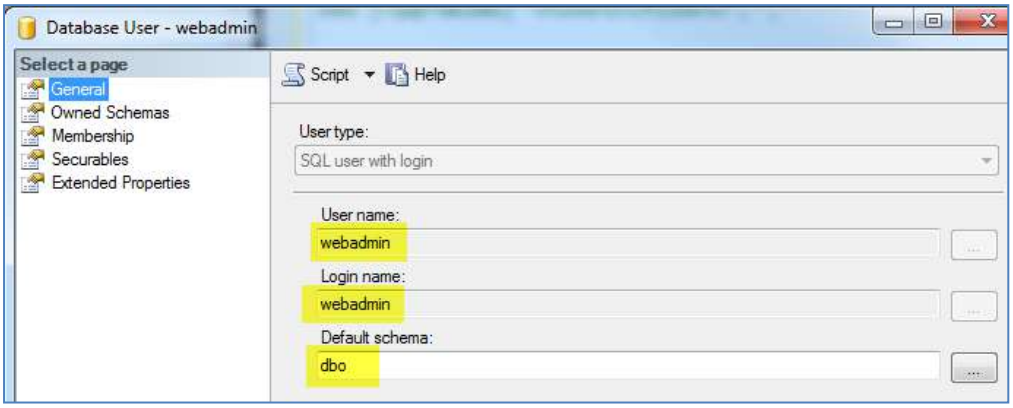


6) Repeat the process for the RPT database using the ReportDB.sql.

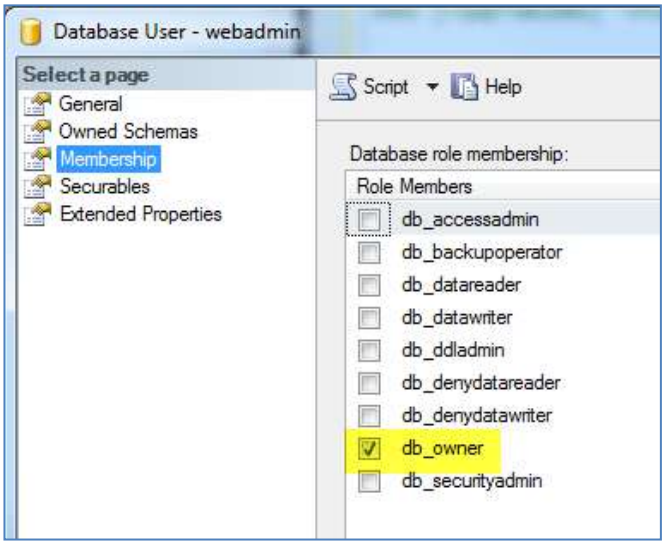
7) Create **two users** in both the APP and RPT databases – webadmin and TPSEC2BI. Right click Users as below:



On the General tab, enter the following:

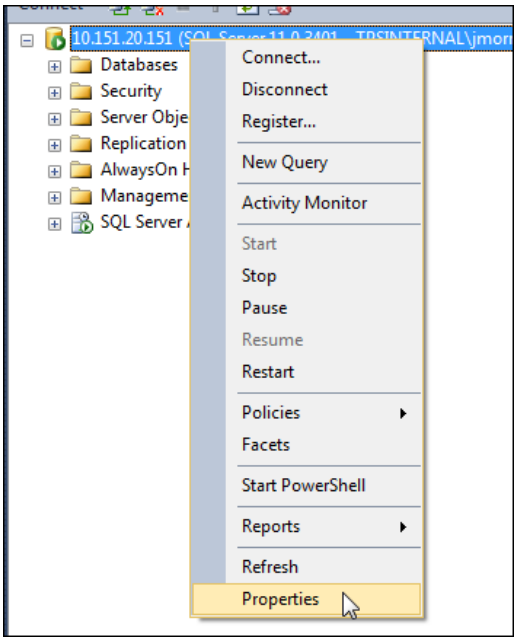


On the Membership tab, select db_owner:

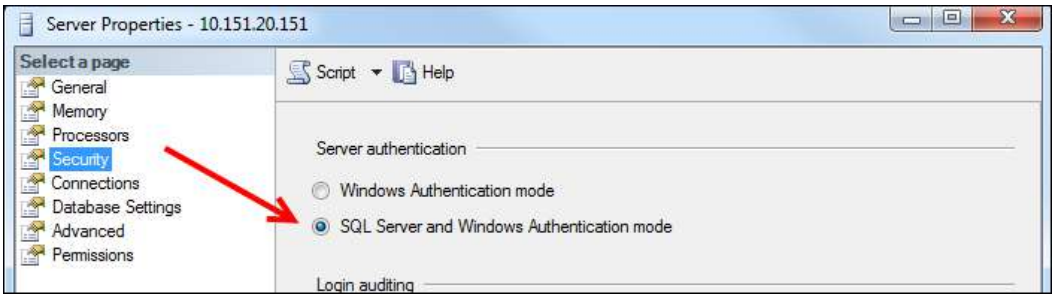


8) For servers that support SSMS logins using Windows Authentication, assure the server also supports SQL Server Authentication.

Right click the server and select Properties:



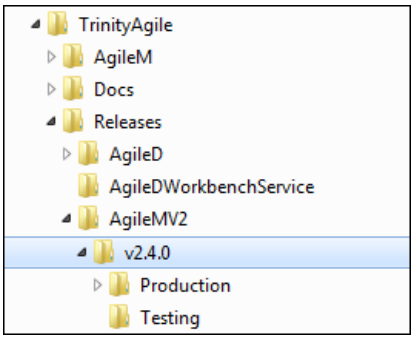
On the Security tab, select SQL Server and Windows Authentication.



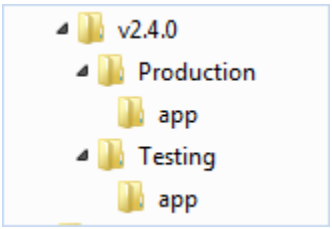
2- Install the mobile v2 application

Perform the following to install the actual mobile v2 application:

1) Locate the latest version at: L:\TrinityAgile\Releases\AgileMV2.



2) Notice within the version folder there is a Production and a Testing folder. The Testing folder contains code that provides more descriptive error messages and is installed in the Development environment. The Production folder contains code for client facing deployments.

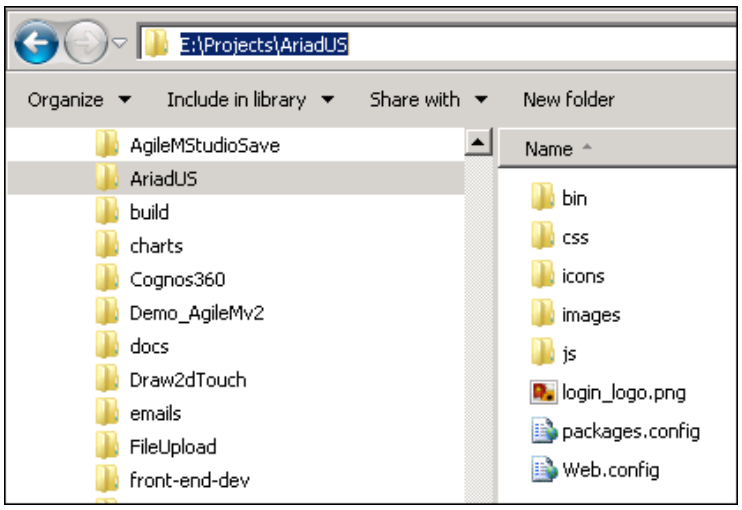


3) Create a remote desktop connection to the client development or production environment server.

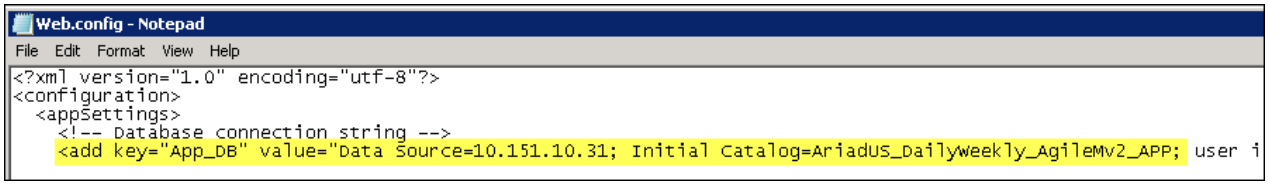
5) Create one new folder in the e:\Projects folder with the naming convention: <Project Name>. For example e:\Projects\AriadUS

Do not vary from this naming standard.

6) Copy files from the appropriate Production or Testing folder. The result will be **approximately** like the following (specific file/folder names may change slightly with new versions).



7) Modify Web.config. Set the key App_DB with the IP address and the Catalog of the APP database created above.



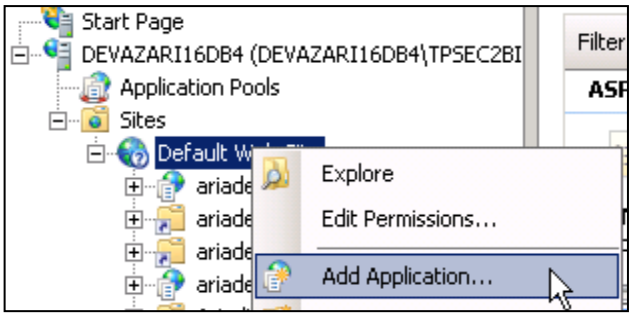
3-Establish the mobile v2 web site

The mobile v2 web site is established using the Internet Information Services Manager (IIS) as follows:

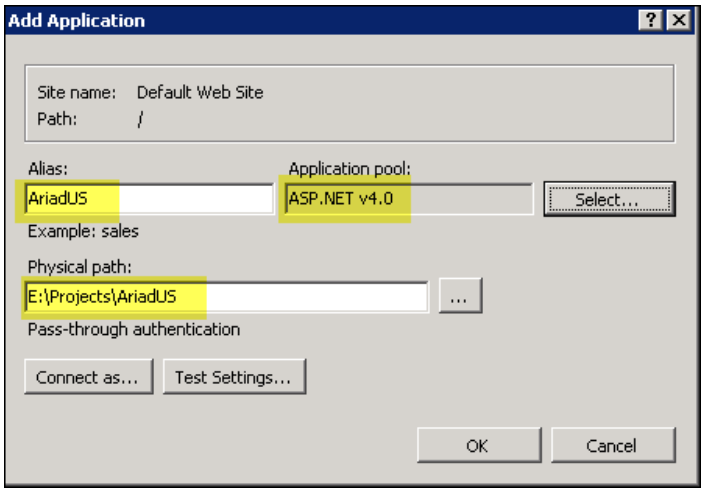
1) Create a remote desktop connection to the client development or production environment server.

2) Launch the Internet Information Services (IIS) Manager from the Start menu.

11) Right click Default Web Site and select Add Application…

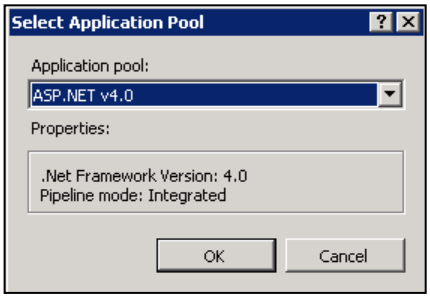


12) Set the Alias as <ProjectName>.

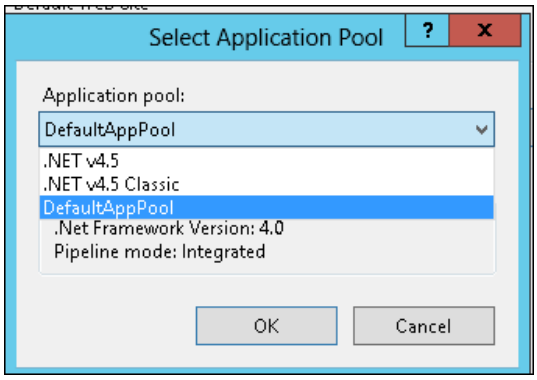


Set the physical path to the E:\Projects\<ProjectName> directory established above.

Also, set the Applicaion pool to ASP.NET v4.0 or DefaultAppPool.

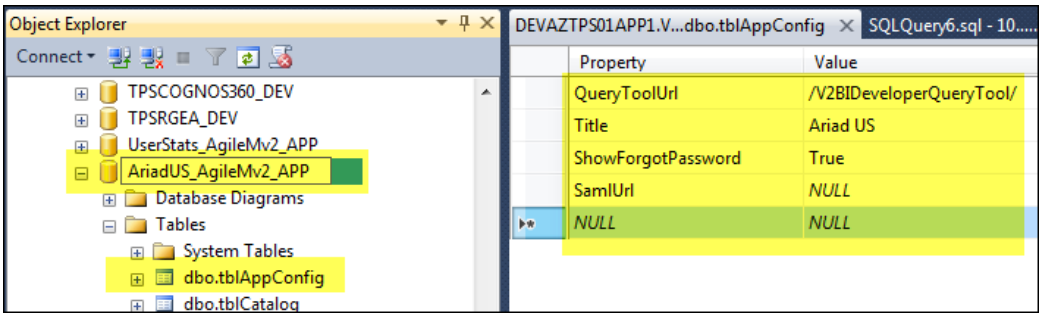


13)



14) Edit the application configuration settings in the <ProjectName>_AgileMv2_APP database, tblAppConfig table. (Note below that the SamlUrl entry below of *NULL* is not a valid entry).

    At this point the installation is complete and a full test should be performed.



Property	Value
AnalyticsSuite	Deprecated.
DisableExcelExport	**Optional**. Set to true to turn off the ability to export Component data to Excel. Note that this affects all Reports in the Project. Excel export will be supported if this property does not exist or the value for this property is anything other than true.
EnableUserTracking	**Optional**. Set to true to capture Google Analytics usage data. Any other value (including false), or the lack of this property will result in usage data not being captured.
FilterInfoAlwaysDisplay	**Optional**. Set to true to force the Applied Filters to always display, to prevent the Sprocket menu from offering the Hide Filter Info selection and to remove the 'X' close button on the right side of the Applied Filters display. Any value other than true, (including false), has no effect.
FilterInfoDefault	**Optional**. Set to: 'on' to have the Applied Filters panel display by default. Set to 'off' or do not enter this row to have the Applied Filters panel to not display by default.
InactiveMinutes	**Optional**. Set the number of minutes for which a user's session may be inactive without the user being informed the session has timed out and thus requiring a login to continue. If this property is missing, then the default value is 1440 minutes (i.e. 24 hours).
LogoutURL	**Optional**. An arbitrary URL that AP will redirect to upon selecting Logout from the Sprocket menu. See Logout and Return To Portal for additional required set-up information.
PortalText	**Optional**. Used in conjunction with PortalURL and limited to 26 characters long. The text that will appear in the Sprocket menu and when selected AP will redirect to. See Logout and Return To Portal for details.
PortalURL	**Optional**. Used in conjunction with PortalText. The URL that AP will redirect to when PortalText is selected in the Sprocket menu. See Logout and Return To Portal for details.
QueryTooUrl	URL to the Query Tool.
QueryToolUserTypeIDAccess	**Optional**. A comma separated list of userTypeID's that will have access to the Query Tool. If not present, and QueryToolUrl exists, than all users will have Query Tool access.
SamlUrl	**Optional**. URL for Auth0 login to AP.
ShowForgotPassword	Boolean that controls the existence of the "Forgot Password" prompt on the user login page.
StandardColorOrder	**Optional**. A comma separated list that establishes the order the system will present the standard colors in. That is, a set of standard colors listed in their order of usage. When Components are drawn, and their data set has not provided a color, the system will draw upon this color list and its order to color a Component, (for example, the slice of a pie). Click here for the colors and their order. This property allows the order of the colors to be changed. The list must contain only the standard colors, but the order may be adjusted. Example list (note the '#' in front of each color):

	#459AE3,#FF8602,#F9C213,#71B619,#EC4759,#5AC8E7,#AC92EC, #25C8B4,#F28FE5,#6577E9,#84D83C,#FD9D0A,#B41F51,#B4C1C8, #D73098
TermsOfUseDisplayDefault	Boolean that determines if the Terms of Use text displays by default.
TermsOfUseText	Text to display in the Terms of Use box in the user login page. Required and limited to a maximum of 400 characters.
ThemeDefault	**Optional**. Set to 'light' to have the light theme display by default. Set to 'dark' or do not enter this row to have the application display the dark theme.
Title	Title to appear on the user login page.

Installation in Production

When installed in a non-production environment, the APP database, the RPT database and the web site are all installed on the same 10.151.10.x server.

But, when installed in a production environment, we do not want clients having direct access to the databases on 10.151.10.x servers. Rather, set up the v2 mobile application and IIS on a 10.151.20.x server.

IT will establish a DNS name server entry so that clients will enter a friendly name, like: https://ariad.trinityagile.com/PhysicianSegmentation/.

The databases will remain on a 10.151.10.x server. The application will know where to find the 10.151.10.x databases by using the entry in web.config.

Note that an additional security step is required when installing the v2 mobile application on a 10.151.20.x server. The server must be given SQL access to the 10.151.10.x database servers. Request from IT that "SQL on port 1433 on the 10.151.20.x web server be allowed to access the 10.151.10.x database server".

Application Upgrades

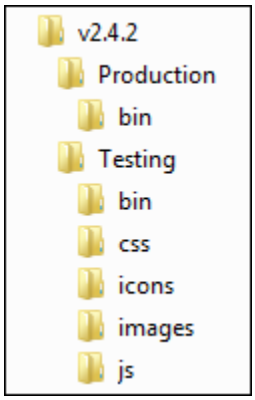
Application enhancements and bug fixes will require that the v2 application occasionally be upgraded. Following is a check list of the required steps:

1) **Backup Development**. On the mobile development server, create a backup copy of the application folder. Following the naming conventions above, create a copy of the: <Project Name> folder**.**

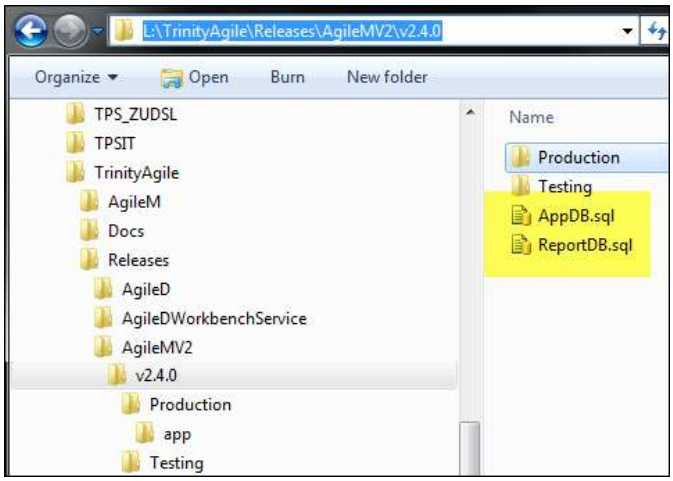
2) **Delete folder contents**. Delete the contents of the original <Project Name> folder.

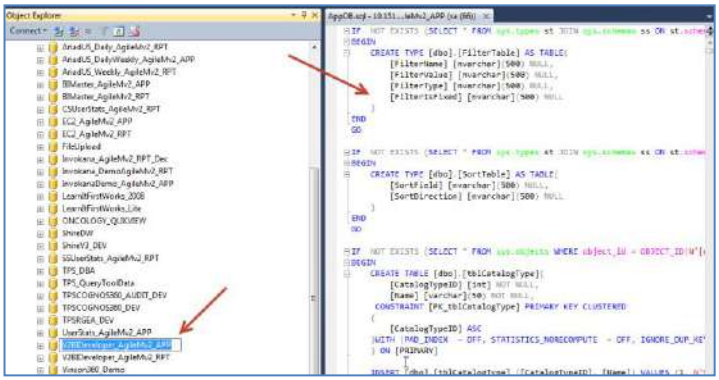
3) **Get current application files**. Locate the latest v2 release in: L:\TrinityAgile\Releases\AgileMV2. The structure of this folder will be the following. Copy the contents of the Testing folder to the e:\Projects\<ProjectName> folder.

V2 Application Files



4) **Database Update**. The AppDB.sql and ReportDB.sql will update the v2 _APP and _RPT databases. Each is a large build script that will create/update the database objects. Execute each build script in a query window.





5) **Test**. Test the new release. Read the release notes and pay particular attention to items that were changed or new features. Require that the BA also test and approve the new system.

6) **Repeat in Production**. Once confidence is gained in the new system and the EM has approved the upgrade, repeat the process in production with the following additional comments:

   - Updates in production must be performed at a time agreed upon with the EM/BA.
   - Use the release files located in the Production folder (not the Testing folder).

7) **Delete backup copies**. After some period of time delete the backup copies in step one above. Or optionally, one or two versions of the backup folders may remain – but don't make many backup copies.

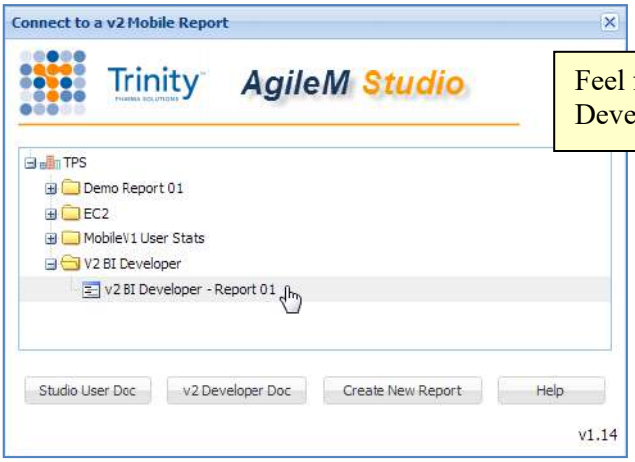
LAYOUT TYPES

Developer Database

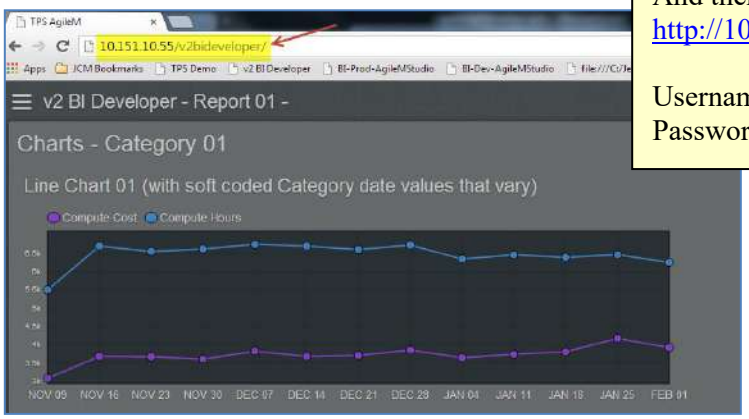
As a convenience to the v2 Developer, there is a single v2 mobile database that demonstrates many of the concepts throughout this manual. Following are location and access details:

- URL: http://10.151.10.187/APDeveloper/
- Username: a (Note: v2 Usernames are not case sensitive)
- Password: A0000000 (Note: v2 Passwords are case sensitive)
- Server: 10.151.10.187
- App database: APDeveloper_AgileMv2_APP
- Rpt database: APDeveloper_AgileMv2_RPT

You may use the following area to practice with AgileM Studio and then actually observe how your changes affect the user interface. Note that this area can easily be restored, so feel free to "play" with the Attribute setting in AgileM Studio.



Feel free to play with the v2 BI Developer – Report 01 report.



And then observe your changes here:
http://10.151.10.187/APDeveloper/

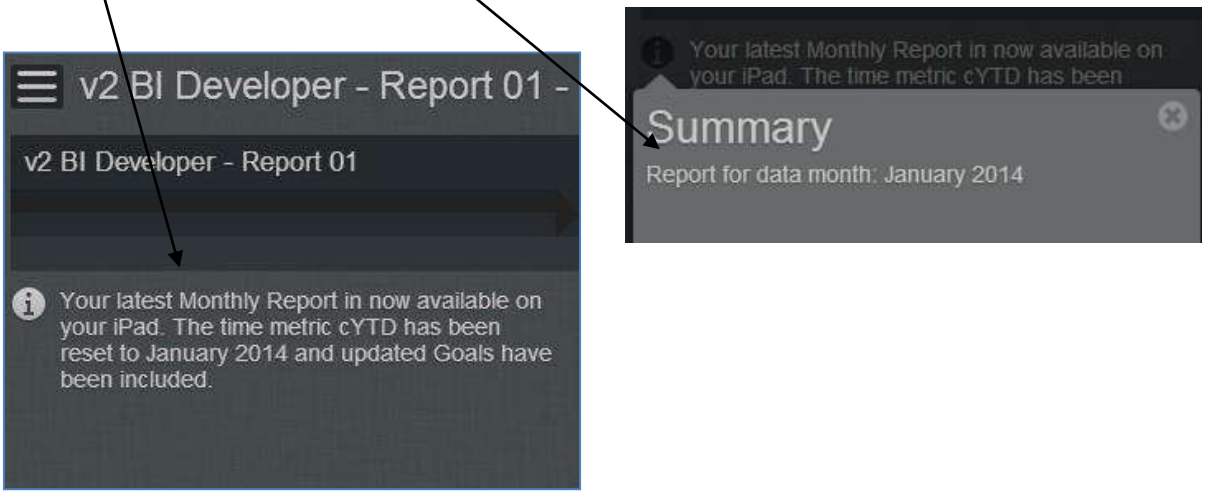
Username: a
Password: A0000000

Report

Report is the overarching object from which all other objects flow.

Attributes:

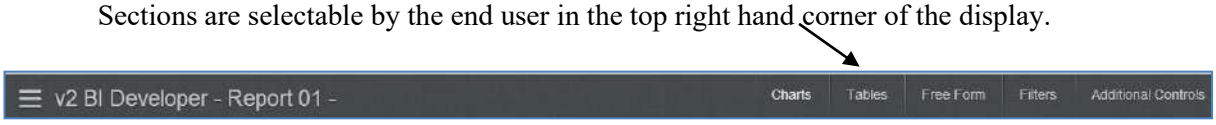
Attribute	Required	Function
Bulletin	Yes	Text that displays in the bottom of the Insights panel.
Summary	Yes	Text to display when the information button, located next to the Bulletin of the Insights panel, is selected.
Title	Yes	Unused.
Enable Global Filters	No	Flag to enable global filtering for this report.
Require Global Filters	No	Force global filtering on at all times.
DisableExcelExport	No	A comma separated list of those UserTypeIDs for whom Component Excel Export will be disabled.
GlobalFilterDefault	No	Establishes the initial state of global Filters as either on or off (in those Reports that support global Filters).



Section

Section Carousel

Section is the highest level object below the Report. A Report can have one to many Sections. Sections are selectable by the end user in the top right hand corner of the display.



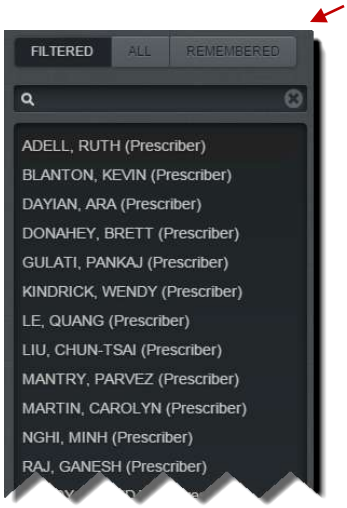
Section Carousel Attributes:

Attribute	Required	Function
DisplayOrder	Yes	Technically, not required, but **it is best practice to use DisplayOrder for all objects that have DisplayOrder** (Section, Category, Component, ColumnGroup, Column and Filter). If DisplayOrder is not used, then objects are ordered by LayoutID.
ShortTitle	Yes	The text to display in the navigation bar when a Section is not the active Section. (Note that the v2 BI Developer report sets ShortTitle = LongTitle, but for all client reports ShortTitle must be a number).
LongTitle	Yes	The text to display in the navigation bar when a Section is the active Section.

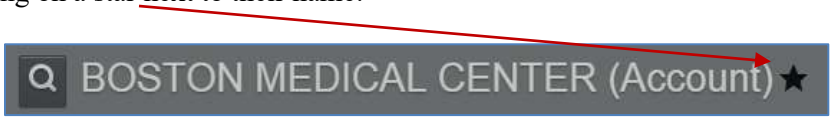
List View Section Carousel

Setting a section type to listviewsectioncarousel within AgileM Studio enables the front end search functionality on the section level, providing end users with the capability to search for a particular entity and view report components filtered with data belonging to the selected entity.

By clicking on the search button, located on the top left of the category, the user can view a list of entities broken out into three categories: 1) filtered, 2) all, 3) remembered.



'Filtered' includes a list of entities that have been filtered by the filters specified in the report. For instance, sales group and segment type filters specified in the report, only entities that have the specified values in their sales group and segment type fields will be presented in the filtered list.
In contrast, the 'All list includes all distinct entities that are not filtered for a particular sales group code and entity type and that exist in the table from which the Entity List is extracted from.
Finally, the 'Remembered' list includes those entities that have been specified as a favorite by clicking on a star next to their name:

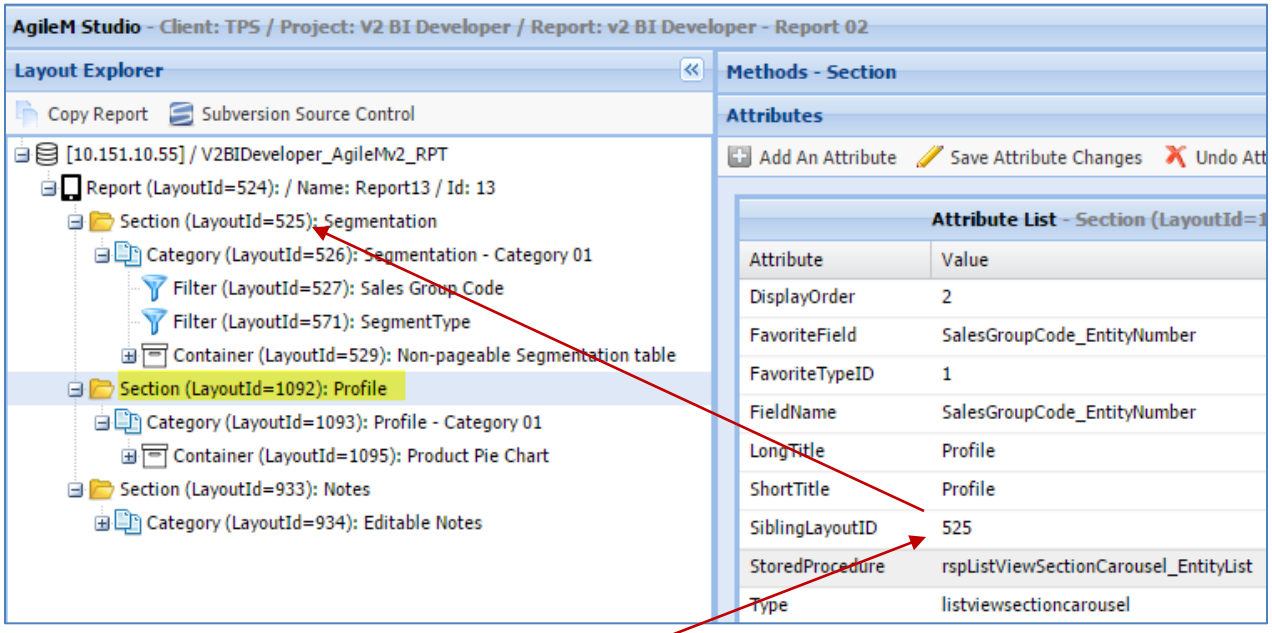


List View Section Carousel Attributes:

Attribute	Required	Function
DisplayOrder	Yes	Technically, not required, but **it is best practice to use DisplayOrder for all objects that have DisplayOrder** (Section, Category, Component, ColumnGroup, Column and Filter). If DisplayOrder is not used, then objects are ordered by LayoutID.
FavoriteField	Yes	FavoriteField is the value that gets populated into tblUserFavorite table as its Value field. The value of FavoriteField is defined and specified in the stored procedure code for the listviewsectioncarousel section.
FavoriteTypeID	Yes	Usually this is set to 1.
FieldName	Yes	FieldName in a listviewsectioncarousel is used to specify the value that will be used to track which selection the user has made. Within the stored procedure this has to be returned in the 'value' field in the SELECT statements. The app treats this value in a similar way it treats values for filters, storing it in the same place where filter values get stored. For this reason, the specified FieldName also needs to be defined and treated like a filter in the category stored procedure code.
ShortTitle	Yes	The text to display in the navigation bar when a Section is not the active Section. (Note that the v2 BI Developer report sets ShortTitle = LongTitle, but for all client reports ShortTitle must be a number).
LongTitle	Yes	The text to display in the navigation bar when a Section is the active Section.
SiblingLayoutID	Yes	SiblingLayoutID is the layoutID of the sibling section. Usually the sibling section is the section in the report that precedes the listviewsectioncarousel section.
StoredProcedure	Yes	The StoredProcedure attribute needs to point to the stored procedure that is written to handle the 3 list cases: filtered, all, and remembered.
Type	Yes	listviewsectioncarousel

APDeveloper Report 02:

Section: Profile
Container: /
Stored Proc: rspListViewSectionCarousel_EntityList



In this example, Profile is a listviewsectioncarousel type section, and its sibling section is Segmentation with LayoutID = 525. This means that the filters defined in the last category of the Segmentation section will be passed to the Profile section. In this particular example, the last category in Segmentation has two filters: Sales Group Code and Segment Type. Although, these filters are not defined anywhere in the configuration of the Profile section or its category, they need to be defined within the listviewsectioncarousel stored procedure code:

```
DECLARE @vchSalesGroupCode  VARCHAR(15) = (SELECT dbo.GetFilterValue(@tblFilterNameValue, 'SalesGroupCode'))
DECLARE @vchSegmentType     VARCHAR(15) = (SELECT dbo.GetFilterValue(@tblFilterNameValue, 'SegmentType'))
```

The FieldName attribute, set to SalesGroupCode_EntityNumber, on the other hand is not directly defined as a field in the select statement within the listviewsectioncarousel stored procedure. Its value is defined in the 'value' field in the select statements within the listviewsectioncarousel stored procedure and this is the value that the app stores in SalesGroupCode_EntityNumber FieldName attribute. Then, similar to how the filter values are retrieved in the stored procedure, the SalesGroupCode_EntityNumber needs to be called within the stored procedure for the category, which is in this case rspListViewSectionCarousel_Profile_Category01. That declaration might look like this:

```
DECLARE @vchSalesGroupCode_EntityNumber  VARCHAR(100) = (SELECT dbo.Getfiltervalue(@tblFilterNameValue, 'SalesGroupCode_EntityNumber'))
DECLARE @vchSalesGroupCode               VARCHAR(100) = LEFT(@vchSalesGroupCode_EntityNumber, Charindex('_', @vchSalesGroupCode_EntityNumber) - 1)
DECLARE @vchEntityNumber                 VARCHAR(100) = Substring(@vchSalesGroupCode_EntityNumber, Charindex('_', @vchSalesGroupCode_EntityNumber) + 1, 50)
```

Notice how the SalesGroupCode and EntityNumber are extracted from SalesGroupCode_EntityNumber and broken out into two separate fields that can later be used in the where clause to filter the result set for components within the category stored proc code.

Since the list view has three tabs on the front end (filtered, all, remembered), the stored procedure code needs to include the logic to handle these three cases; this logic can be observed in rspListViewSectionCarousel_EntityList. The three cases will be discussed here.

The code for the 'ALL' tab should look something like this:

```
--------------------------------All Tab-------------------------------
IF @vchSalesGroupCode IS NULL AND @showFavorites IS NULL
    BEGIN

        SELECT
             name = Name + ' ('+ SegmentType +')'
            ,value = SalesGroupCode + '_' + EntityNumber
            ,FavoriteID = SalesGroupCode + '_' + EntityNumber
            ,Fav = 0
        FROM tblRptSegmentation_Profile seg
        GROUP BY SalesGroupCode, Name, EntityNumber, SegmentType
        ORDER BY SegmentType

    END
```

Notice the capitalization of 'name' and 'value' fields within the select statement. It is important to pass these fields exactly in the format presented here. Their values (whatever is to the right of the = sign), on the other hand, vary from project to project. For real life client projects this query should also include a join to tblUser, tblCfgReportAssignment, and tblSalesGroup tables to ensure that reps only see entities from their geographies. In addition, the result set should be filtered in a way that shows rows from only the highest level geography, eliminating duplicates from showing up on the list.

The code for the 'REMEMBERED' tab should look something like this:

```
-----------------------------Remembered Tab-----------------------------------------------
ELSE IF @showFavorites = 1
    BEGIN

        SELECT
             name = Name + ' ('+ SegmentType +')'
            ,value = SalesGroupCode + '_' + EntityNumber
            ,FavoriteID = SalesGroupCode + '_' + EntityNumber
            ,Fav = 1
        FROM tblRptSegmentation_Profile seg
            LEFT JOIN tblUserFavorite f
                ON LEFT(f.Value, CHARINDEX('_', f.Value) - 1) = seg.SalesGroupCode
                AND SUBSTRING(f.value, CHARINDEX('_', f.Value) + 1, 50) = seg.EntityNumber
        WHERE f.UserID = @userID
            AND CHARINDEX('_', f.value) > 0
        GROUP BY SalesGroupCode, Name, EntityNumber, SegmentType

    END
```

Notice that the Fav field is set to 1 in this case, indicating that favorites should be shown. Also, notice that the FavoriteID, which matches the FavoriteField attribute from configuration, determines what is going to be stored in the tblUserFavorite as the Value field. In this case it is

important to concatenate sales group code and entity number because there might be instances in which one entity number has multiple sales group codes. This concatenation ensures proper identification. Knowing this information, the tblUserFavorite table can then be joined to the main table like it is in the code example above.

The code for 'FILTERED' tab should look something like this:

```
------------------------------Filtered Tab-------------------------
ELSE
    BEGIN

        SELECT DISTINCT
            name = Name + ' ('+ SegmentType +')'
            ,value = SalesGroupCode + '_' + EntityNumber
            ,FavoriteID = SalesGroupCode + '_' + EntityNumber
            ,Fav = 0
        FROM tblRptSegmentation_Profile
        WHERE SalesGroupCode = @vchSalesGroupCode
            AND SegmentType = @vchSegmentType
        GROUP BY SalesGroupCode, Name, EntityNumber, SegmentType

END
```

Notice the where clause and how the result set it filtered with the filters passed from the front end.

Limitations:
It is important that the developer be aware of known current limitations while implementing the listviewsectioncarousel section.

One of the limitations is that a listviewsectioncarousel must have a sibling section that precedes it. This means that a listviewsectioncarousel cannot be implemented in a report as a first section, or as its own separate section.

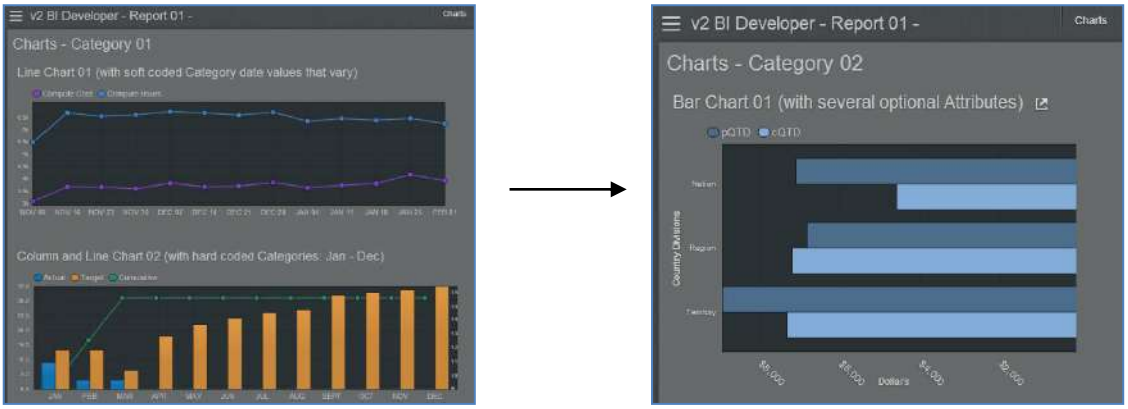
Another limitation is that the listviewsectioncarousel section gets filter values passed from the last category of its sibling section. This means that if a sibling section has multiple categories, only the filter selections that the user has made at the last category of the sibling section will be passed to the listviewsectioncarousel section.

Category

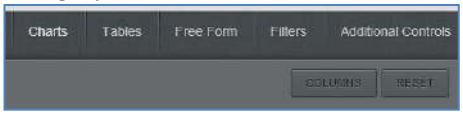
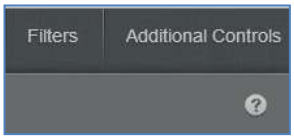
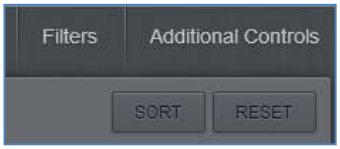
Category Panel

Sections are divided into Categories and can be thought of as similar to “pages”. A Section may have one to many Categories. When a user does a horizontal swipe they will go to the next Category (which may or may not be in the current Section).

This horizontal swipe goes from one Category to the next (in the same Charts Section).



Category Attributes:

Attribute	Required	Function
DisplayOrder	Yes	Technically, not required, but **it is best practice to use DisplayOrder for all objects that have DisplayOrder** (Section, Category, Component, ColumnGroup, Column and Filter). If DisplayOrder is not used, then objects are ordered by LayoutID.
StoredProcedure	Yes	A single stored procedure that provides data to non-pageable Components in the Category.
Title	Yes	The text to display at the top, left hand side, of a Category. While this Attribute is required it is also nullable. 
Type	Yes	Currently limited to categorypanel.
FavoriteField	No	Result set field name containing data to put in tblUserFavorite. See the Additional Controls - Favorites for more details.
Hideable	No	True or False to control the presence of the Columns button in a Category. 
ShowHelp	No	True or False to control the presence of the '?' in a Category. 
SiblingLayoutID	No	The LayoutID of a different Category to link to.
Sortable	No	Controls the displaying of the sort button on Category. Defaults to false. 

The primary points to understand about Categories is that they:

- Have a StoredProcedure Attribute
- The stored procedure returns one to many record sets
- Each record set contains a column titled: layoutID (spelled **exactly** as here)
- The layoutID maps each record set to one Component in the Category
- The record set then populates the Component with data

For example, in the following, these record sets are returned. Note the tie between layoutID in the record sets and LayoutId displayed in AgileM Studio. Also, note that it is acceptable to have multiple record sets with the same layoutID.

APDeveloper Report 01:

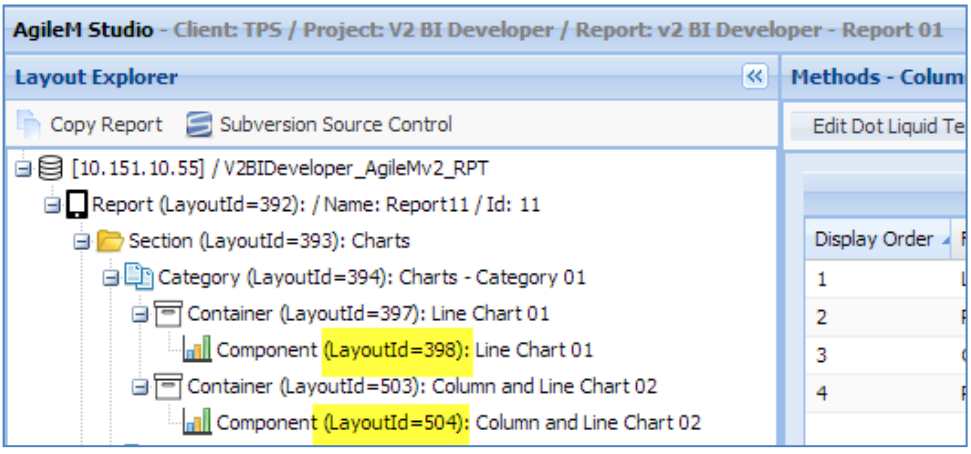
Section: Charts
Container: Line Chart 01
Stored Proc: rspChartsCategory01

Results | Messages

	layoutID	category	y	name	type	band1Color
1	398	Nov 09	3083.5400	Compute Cost	line	#8143bf
2	398	Nov 16	3679.3080	Compute Cost	line	#8143bf
3	398	Nov 23	3674.0850	Compute Cost	line	#8143bf

	layoutID	category	y	name	type	band1Color
1	398	Nov 09	5496.0000	Compute Hours	line	#4381bf
2	398	Nov 16	6695.0000	Compute Hours	line	#4381bf
3	398	Nov 23	6548.0000	Compute Hours	line	#4381bf

	layoutID	format	metric1Format	category	categoryorder	name	yAxis	y	type	band1Color
1	504	decimal	decimal	Jan	1	Actual	0	9.0000	column	#0e73b3
2	504	decimal	decimal	Feb	2	Actual	0	3.0000	column	#0e73b3
3	504	decimal	decimal	Mar	3	Actual	0	3.0000	column	#0e73b3
4	504	decimal	decimal	Apr	4	Actual	0	0.0000	column	#0e73b3
5	504	decimal	decimal	May	5	Actual	0	0.0000	column	#0e73b3
6	504	decimal	decimal	Jun	6	Actual	0	0.0000	column	#0e73b3



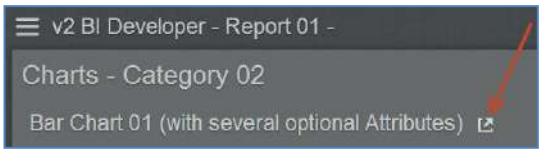
Container

Component Container

Containers define a rectangular area within the Category that will hold the Components users actually see. Containers are defined primarily by their size, (width and height), and position, (x and y co-ordinates).

Note that Containers can have multiple Components and when a Container has more than one Component there will be one button for each Component allowing the user to select the Component to display.

Container Attributes:

Attribute	Required	Function
Height	Yes	Container height in pixels
Title	Yes	Text to display above the Container. 
Type	Yes	Currently must be: componentcontainer
Width	Yes	Container width in pixels.
XPosition	Yes	The Container x position within the Category.
YPosition	Yes	The Container y position within the Category.
Expandable	No	Controls the ability to expand a Container to full screen size. 
SiblingLayoutID	No	Layout Id to use for linking between Containers
SiblingParentLayout	No	Layout Id to use for linking between Containers.

Component

Components are those objects the end user actually "sees" and form the real content of the mobile product. There are three primary types of Components: Charts, Tables, Free Form. The following, along with the V2BI Developer Report 01, provides examples of each.

Component Attributes - all Component types:

Components have many Attributes, most of which are Component type specific. However, the following Attributes apply to all Component types.

Attribute	Required	Function
Title	No	Text to display in a button if there is more than one Component in a Container.
Type	Yes	Defines the major functionality of the Component and will be one of the following: - d3Chart - editable note - freeform - gridpanelcollapse - hcplist - journeytree - multibullet - simplegrid - treemaplist
DisplayOrder	Yes	Technically, not required, but **it is best practice to use DisplayOrder for all objects that have DisplayOrder** (Section, Category, Component, ColumnGroup, Column and Filter). If DisplayOrder is not used, then objects are ordered by LayoutID.

Grid Panel Collapse

APDeveloper Report 01:
Section: Tables
Container: Table Example 01
Stored Proc: rspTablesCategory01

This example presents the simplest case of a Grid. It does not include any of the more complex Grid features (paging, a total row, sorting, rank or favorites).

One item of interest, in this example, is the coloring of the Pcnt Patients at Goal text.

Metric	Current Values	Goal	Pcnt Patients at Goal
BMI	23 to 50	25	9%
Diastolic	60 to 84 mmHg	80	82%

This is accomplished with the following SQL:

```
,PctOfPatientsGoal = '<span style="' +
        CASE WHEN PctofPatientGoal > .60 THEN 'color: #70A926'                  --green
             WHEN PctofPatientGoal between .30 and .60 THEN 'color: #DEB157'    --yellow
             WHEN PctofPatientGoal <.30 THEN 'color: #BA3C2F'                   --red
        ELSE '' END + '">' + PctofPatientGoalTitle + '</span>'
```

Which produces the following:

PctOfPatientsGoal
9%
82%

Another item of interest, in this example, is the use of the UNICODE characters (Circle and Arrow).

Metric	Current Values	Goal	Pcnt Patients at Goal	Circle	Arrow
BMI	23 to 50	25	9%	●	▼
Diastolic	60 to 84 mmHg	80	82%	●	▲

This is accomplished with the following SQL:

```
,Circle = CASE
            WHEN PctofPatientGoal > .60 THEN '<p style="font-size:20px;color:#70A926">&#x25CF</p>' --green dot
            WHEN PctofPatientGoal between .30 and .60 THEN '<p style="font-size:20px;color:#DEB157">&#x25CF</p>' --yellow dot
            WHEN PctofPatientGoal <.30 THEN '<p style="font-size:20px;color:#BA3C2F">&#x25CF</p>' --red dot
            ELSE ''
            END
,Arrow = CASE
            WHEN PctofPatientGoal > .60 THEN '<p style="font-size:20px;color:#70A926">&#9650</p>' --green up triangle
            WHEN PctofPatientGoal between .30 and .60 THEN '<p style="font-size:20px;color:#DEB157">&#9654</p>' --yellow right triangle
            WHEN PctofPatientGoal <.30 THEN '<p style="font-size:20px;color:#BA3C2F">&#9660</p>' --red down triangle
            ELSE ''
            END
```

Which produces the following:

Circle	Arrow
&#x25CF	&#9660
&#x25CF	&#9650

The UNICODE HTML number for a circle is &#x25CF and for an arrow is &#9660. Any UNICODE character can be used in this format as long as the HTML number is used.

It is important to note that the column names in a grid come from two locations – one, tblCfgReportConfiguration and two, the stored proc that generates the grids result set. These two lists of column names should be identical (including case).

When a column in a grid is not populated, be sure to check that the column name, (as shown in AgileM Studio), and the column name in the stored proc are identical. Also, observe the browser console for error messages like the following:

```
⚠ [WARN][TP.view.grid.GridCollapse#compareColumnLists] The following column(s) exists in tblCfgReportConfiguration for
  layoutID 421 but do not exist in this components record set. Remember column names are case sensative: labTest
                                                                    Console.js? dc=1409234110716:35
⚠ [WARN][TP.view.grid.GridCollapse#compareColumnLists] The following column(s) exists in this components record set but do
  not exist in tblCfgReportConfiguration for layoutID 421. This means the stored procedure is providing columns which are
  unused. Remember column names are case sensative: LabTest          Console.js? dc=1409234110716:35
```

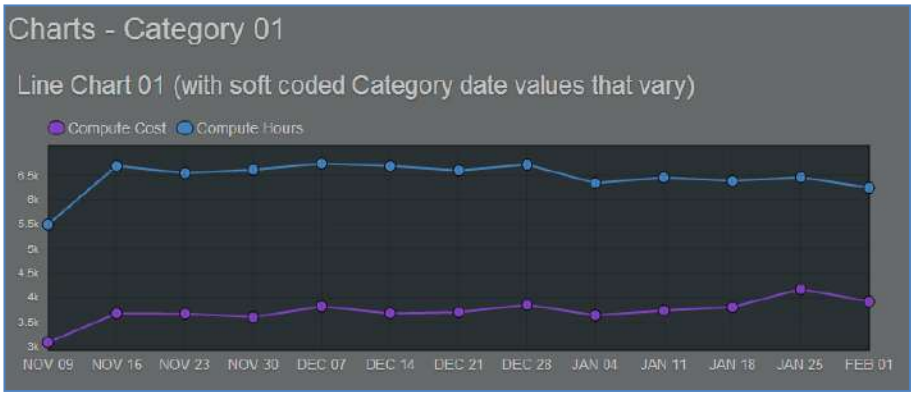
These messages can be a very helpful debugging aid. Note they include the layoutID for which the situation is occurring.

Simple Grid and Enhanced Tooltip

D3 Chart and Enhanced Tooltip – Category 01

APDeveloper Report 01:

Section: Charts
Container: Line Chart 01 and Enhanced Tooltip
Stored Proc: rspChartsCategory01



The stored proc for some Charts can be a little challenging. The following will break this stored proc down into manageable pieces.

Step 1: The inner Select statement will return the following. Note where the column names come from.

```
SELECT
    ComputeCostW13 [1],  ComputeCostW12 [2],  ComputeCostW11 [3],
    ComputeCostW10 [4],  ComputeCostW9  [5],  ComputeCostW8  [6],
    ComputeCostW7  [7],  ComputeCostW6  [8],  ComputeCostW5  [9],
    ComputeCostW4  [10], ComputeCostW3  [11], ComputeCostW2  [12],
    ComputeCostW1  [13]
FROM tblRptComputeCostHistory
WHERE ClientName = 'All Clients'
    AND Environment = 'Total'
    AND InstanceType = 'Total'
```

Results | Messages

1	2	3	4	5	6	7	8	9	10	11	12	13
3083.5400	3679.3080	3674.0850	3604.7190	3822.1920	3680.7040	3704.8340	3855.5350	3642.5840	3739.4720	3807.3980	4174.3020	3923.7380

Step 2: Adding the outer Select and the UNPIVOT, will result in the following. Note how the orientation of the data has changed from horizontal to vertical (i.e. pivoted).

```
SELECT
    ComputeCost
    ,DATEADD(week, CAST(WeekNum AS INT), @dteDataDate ) AS DataWeek
FROM (
    SELECT
        ComputeCostW13 [1],  ComputeCostW12 [2],  ComputeCostW11 [3],
        ComputeCostW10 [4],  ComputeCostW9  [5],  ComputeCostW8  [6],
        ComputeCostW7  [7],  ComputeCostW6  [8],  ComputeCostW5  [9],
        ComputeCostW4  [10], ComputeCostW3  [11], ComputeCostW2  [12],
        ComputeCostW1  [13]
    FROM tblRptComputeCostHistory
    WHERE ClientName = 'All Clients'
        AND Environment = 'Total'
        AND InstanceType = 'Total') s
    UNPIVOT ([ComputeCost] FOR WeekNum IN ([1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13])) upvt
```

Results | Messages

	ComputeCost	DataWeek
1	3083.5400	2013-11-09 00:00:00.000
2	3679.3080	2013-11-16 00:00:00.000
3	3674.0850	2013-11-23 00:00:00.000
4	3604.7190	2013-11-30 00:00:00.000
5	3822.1920	2013-12-07 00:00:00.000

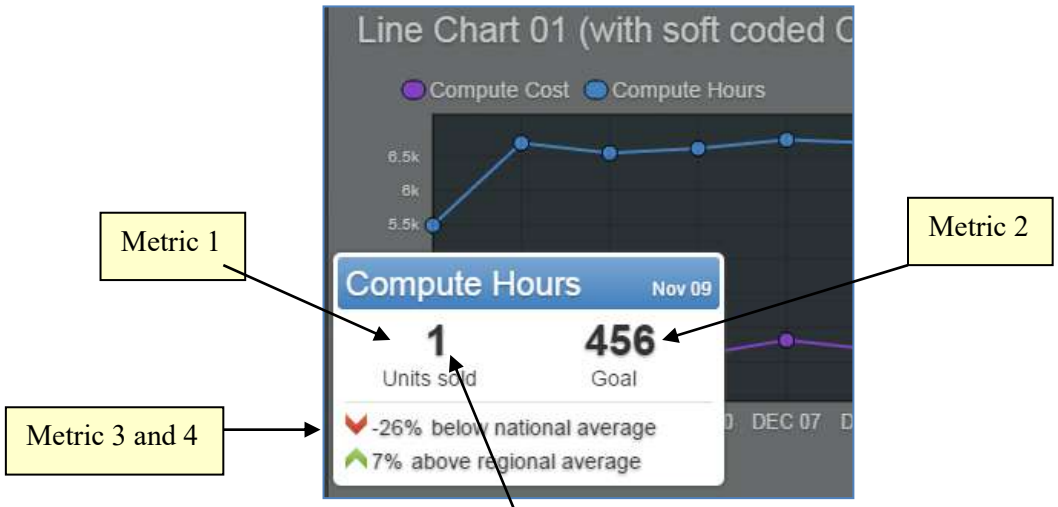
Step 3: Finally, the full Select statement returns the following. Note the With ComputeCost statement that creates a temporary named result set that is used in the final From statement.

```
WITH ComputeCost AS (
    SELECT
        ComputeCost
        ,DATEADD(week, CAST(WeekNum AS INT), @dteDataDate ) AS DataWeek
    FROM (
        SELECT
            ComputeCostW13 [1],  ComputeCostW12 [2],  ComputeCostW11 [3],
            ComputeCostW10 [4],  ComputeCostW9  [5],  ComputeCostW8  [6],
            ComputeCostW7  [7],  ComputeCostW6  [8],  ComputeCostW5  [9],
            ComputeCostW4  [10], ComputeCostW3  [11], ComputeCostW2  [12],
            ComputeCostW1  [13]
        FROM tblRptComputeCostHistory
        WHERE ClientName = 'All Clients'
            AND Environment = 'Total'
            AND InstanceType = 'Total') s
        UNPIVOT ([ComputeCost] FOR WeekNum IN ([1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13])) upvt )

SELECT
    layoutID = 398
    ,category = LEFT(CONVERT(VARCHAR, DataWeek, 7), 6)
    ,y = [ComputeCost]
    ,name = 'Compute Cost'
    ,[type] = 'line'
    ,band1Color = '#8143bf'
FROM ComputeCost
ORDER BY DataWeek ASC;
```

Results | Messages

	layoutID	category	y	name	type	band1Color
1	398	Nov 09	3083.5400	Compute Cost	line	#8143bf
2	398	Nov 16	3679.3080	Compute Cost	line	#8143bf
3	398	Nov 23	3674.0850	Compute Cost	line	#8143bf
4	398	Nov 30	3604.7190	Compute Cost	line	#8143bf
5	398	Dec 07	3822.1920	Compute Cost	line	#8143bf
6	398	Dec 14	3680.7040	Compute Cost	line	#8143bf
7	398	Dec 21	3704.8340	Compute Cost	line	#8143bf

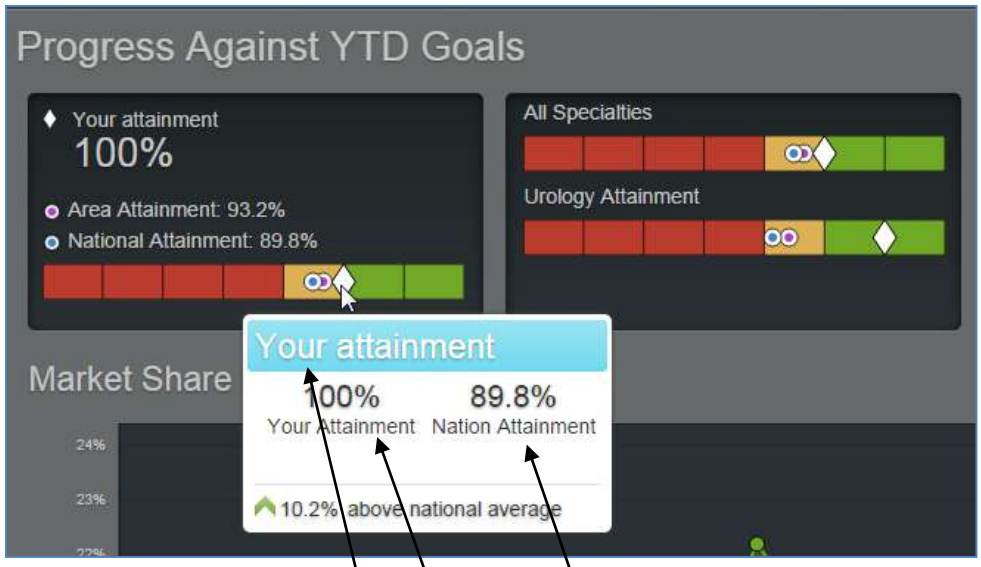
Line Chart 01 also demonstrates the enhanced tool tip which includes four metrics.



category	name	metric1Label	metric1Value	metric2Label	metric2Value	metirc2Format	metric3Label	metric3Value	metric3Format	metric3Chevron	metric4Label	metric4Value	metric4Format	metric4Chevron
Nov 09	Compute Hours	Units sold	1	Goal	456	decimal	below national average	-0.26	percent1	-1	above regional average	0.07	percent1	1
Nov 16	Compute Hours	Units sold	1	Goal	456	decimal	below national average	-0.26	percent1	-1	above regional average	0.07	percent1	1
Nov 23	Compute Hours	Units sold	1	Goal	456	decimal	below national average	-0.26	percent1	-1	above regional average	0.07	percent1	1
Nov 30	Compute Hours	Units sold	1	Goal	456	decimal	below national average	-0.26	percent1	-1	above regional average	0.07	percent1	1

Items to note for component types: componentcontainer, heatmap and multi-bullet

- Only metric1 is required for a tooltip to function. Metric2-4 are optional.
- Category is in the top right corner of the tool tip (for multi-bullet charts the column name is linkCategory).
- If metric1Value does not exist, then y is used here.
- If metric2 does not exist, then metric1 is centered in the tooltip
- All four metrics support metricXValue, metricXFormat and metricXLabel.
- Values for metricXFormat are:
  - percent
  - USD (for dollars)
  - decimal
  - all other values will return metricXValue
- metric3 and metric4 also support metricXChevron as follows:
  - < 0 is a small red down Chevron
  - >0 is a small green up Chevron
  - =0 then no Chevron

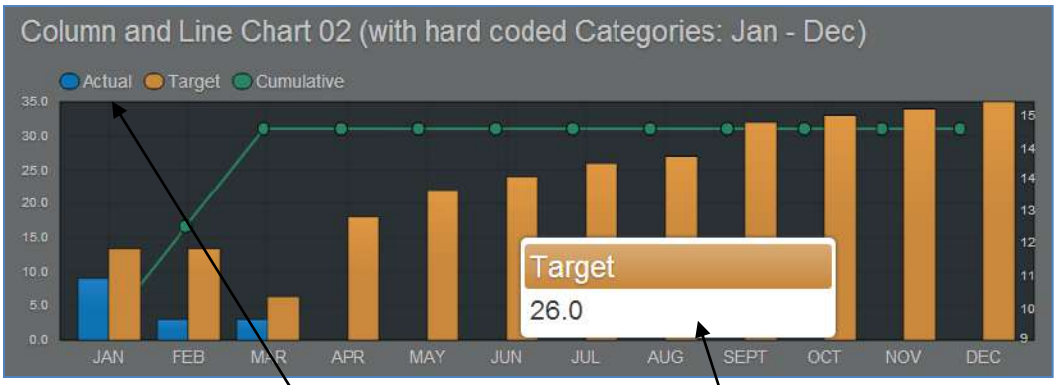


Items to note for component types: monthly-summary-progress, multiple-attainment and progresspercent

- There is no value in the top right corner.
- The label is actualPctLabel
- These values are actualLabel, actual and actualFormat
- These values are goalLabel, goal and goalFormat
- actual and goal are both required

APDeveloper Report 01:

Section: Charts
Container: Bar and Line Chart 02
Stored Proc: rspChartsCategory01



This Chart builds on Line Chart 01 above by showing: Columns, multiple series types (columns and lines), and hard coded Categories.

Items to note in the record set:

	layoutID	format	metric1Format	category	categoryorder	name	yAxis	y	type	band1Color
1	504	decimal	decimal	Jan	1	Actual	0	9.0000	column	#0e73b3
2	504	decimal	decimal	Feb	2	Actual	0	3.0000	column	#0e73b3
3	504	decimal	decimal	Mar	3	Actual	0	3.0000	column	#0e73b3
4	504	decimal	decimal	Apr	4	Actual	0	0.0000	column	#0e73b3
5	504	decimal	decimal	May	5	Actual	0	0.0000	column	#0e73b3
6	504	decimal	decimal	Jun	6	Actual	0	0.0000	column	#0e73b3

- **format**. Controls the mask used for the y-axis numbers.
- **metric1Format**. Controls the mask used for values in the pop-up that appears when clicking on a series.
- **name**. Name of the series that appears in the legend
- **type**. Controls the series type and may be: column, line or bar
- **band1Color**. Controls both the series and pop-up color
- **secondaryYAxisLabel**. Optional column that controls the existence and text to appear in a secondary Y axis label

Additional features of Charts:

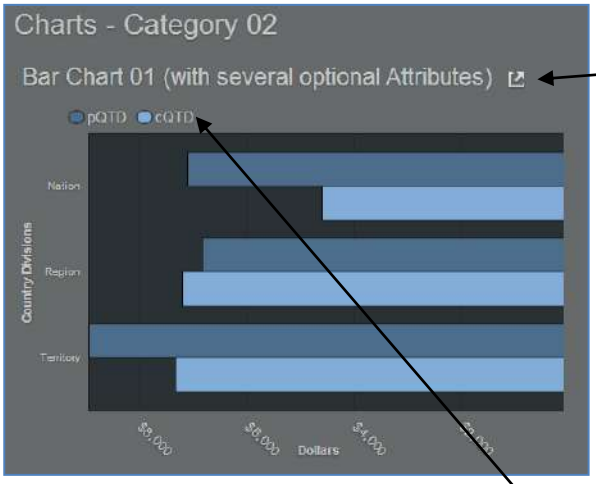
- The y-minimum and y-maximum values, on the vertical scale, may be system determined or controlled through data with yMax and yMin columns.

```
SELECT
    layoutID = 398
    ,category = LEFT(CONVERT(VARCHAR, DataWeek, 7), 6)
    ,y = [ComputeHours]
    ,yMax = 8000
    ,yMin= 0
    ,name = 'Compute Hours'
    ,[type] = 'line'
```

D3 Chart – Category 02

APDeveloper Report 01:

Section: Charts
Container: Bar Chart 01
Stored Proc: rspChartsCategory02



Container level Attribute Expandable that controls the presence of the expand button.

Pushing the button will expand the Container to full screen size.

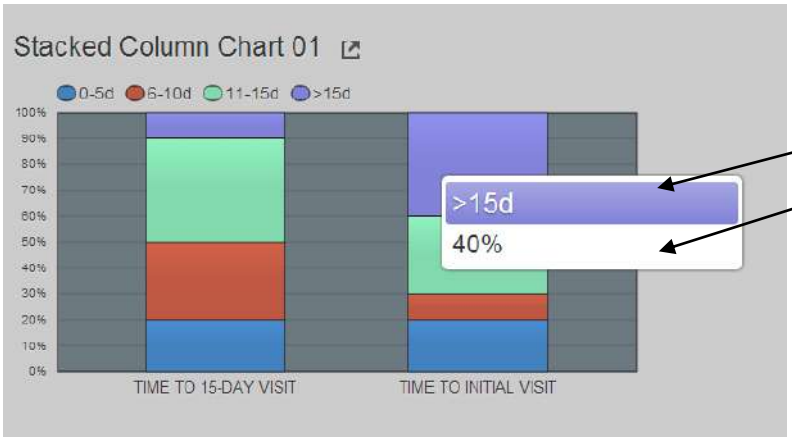
This Chart demonstrates a bar Chart and several optional Component Attributes as noted below.

Some of the Attributes have default values and in those cases the Attribute does not need to be listed. For example, ShowYAxis really only needs to be explicitly listed when its value is False, because if the Attribute is not listed, the axis will display.

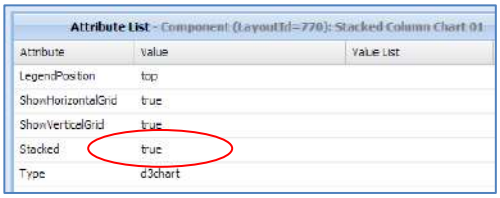
Attribute	Required	Function
HideLegend	No	True or False to control the display of the legend.
InvertChart	No	When False, as above, the bars will "start" from the right hand side.
LegendPosition	No	When Top, legend will display in the position as noted above. When Right, the legend is on the top/right side.
ShowHorizontalGrid	No	True to show horizontal lines. Note that the black line on a dark background is a little hard to actually see.
ShowVerticalGrid	No	True to show vertical lines. Note that the black line on a dark background is a little hard to actually see.
ShowXAxis	No	True to show the x-axis values.
ShowYAxis	No	True to show the y-axis values.
XAxisTitle	No	Text of x-axis labeling ("Dollars" above).
YAxisTitle	No	Text of y-axis labeling ("Country Divisions" above)

APDeveloper Report 01:

Section: Charts
Container: Stacked Column Chart 01
Stored Proc: rspChartsCategory02



This Chart demonstrates a Stacked Column Chart, which is defined by setting the Stacked attribute equal to 'true' in the attribute list in AgileM Studio and achieving the stored proc result set below.



Attribute List - Component (LayoutId=770): Stacked Column Chart 01

Attribute	Value	Value List
LegendPosition	top	
ShowHorizontalGrid	true	
ShowVerticalGrid	true	
Stacked	true	
Type	d3chart	

	layoutID	type	name	category	categoryor...	y	format	metric1For...	band1Co...
1	770	column	0-5d	Time to 15-day visit	1	0.2000	percent	percent	#4381bf
2	770	column	6-10d	Time to 15-day visit	1	0.3000	percent	percent	#bf5843
3	770	column	11-15d	Time to 15-day visit	1	0.4000	percent	percent	#82d9ad
4	770	column	>15d	Time to 15-day visit	1	0.1000	percent	percent	#8282d9
5	770	column	>15d	Time to initial visit	2	0.4000	percent	percent	#8282d9
6	770	column	11-15d	Time to initial visit	2	0.3000	percent	percent	#82d9ad
7	770	column	6-10d	Time to initial visit	2	0.1000	percent	percent	#bf5843
8	770	column	0-5d	Time to initial visit	2	0.2000	percent	percent	#4381bf

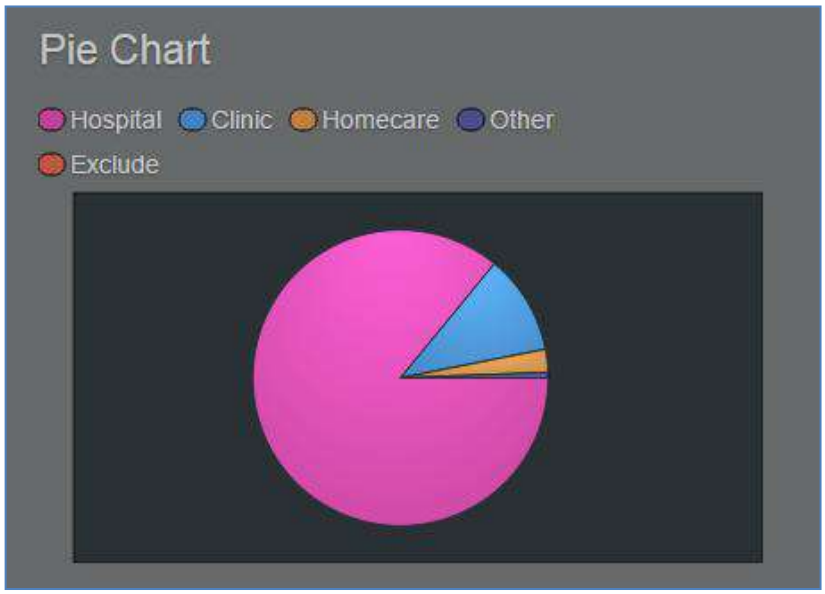
Looking at the stored procedure more carefully, the developer can notice a LEFT JOIN to the view, which was created to connect specific colors with series names ('name' in the result set). This is particularly useful when developing reports for clients who want specific colors in the report, as opposed to default colors determined by the app code.

Demonstrating 'light/dark theme' functionality of the app, the screenshot displays the chart in the 'light theme' (note the clearly defined horizontal grid). To access the light theme, the user can simply click on the report button on the top left corner of the app.

After, the user can click on the 'light theme' button located on the bottom left corner of the app. To go back to dark theme, the user would simply click 'dark theme'.

APDeveloper Report 01:

Section: Charts
Container: Pie Chart 01
Stored Proc: rspChartsCategory02



This Chart and record set below demonstrates a Pie Chart. Note that the columns that begin with 'metric' are optional and affect the tooltip display not the actual pie chart.

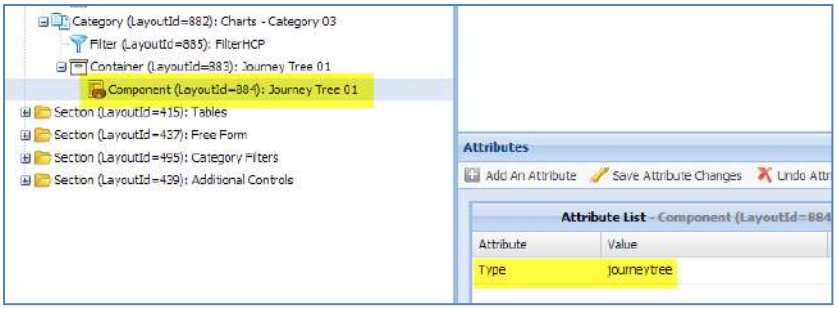
	layoutID	metric1Format	metric1Value	metric2Label	metric2Value	name	category	type	y
1	949	percent	0.8597	Vol	575.9998	Hospital	breakout	pie	575.9998
2	949	percent	0.1092	Vol	73.1429	Clinic	breakout	pie	73.1429
3	949	percent	0.0252	Vol	16.8572	Home...	breakout	pie	16.8572
4	949	percent	0.0060	Vol	4.0000	Other	breakout	pie	4.0000
5	949	percent	0.0000	Vol	0.0000	Exclude	breakout	pie	0.0000

Journey Tree – Category 03

APDeveloper Report 01:

Section: Charts
Container: Journey Tree 01
Stored Proc: rspChartsCategory03

This Chart demonstrates a Journey Tree. JourneyTrees are supported beginning with Mobile version 2.1.0. It is established by setting its Component Type attribute to 'journeytree'.



Note that the Expandable attribute of the Container should be set to true, since the display of this component may particularly benefit from being made larger.

The Journey Tree is a set of one or more tree structures. Each structure starts from a single node and then branches out from that node. A node is represented by a text label and a circle. The size of the circle is based on the value column in the Journey Tree record set and is relative to all other nodes in the record set. Nodes in the same level, i.e. nodes with the same parent, are sorted in descending value order.

The record set for this Journey Tree is the following. (Note that the column names must be capitalized exactly as below).

	layoutID	nodeID	name	category	value	parentNodeID	metric1Value	metric1Label	metric2Value	metric2Label
1	884	1	Gleevec	1st Line	6	NULL	1521	Days on Therapy	6	Number of Patients
2	884	2	Tasigna	2nd Line	3	1	154	Days on Therapy	3	Number of Patients
3	884	3	Sprycel	2nd Line	2	1	188	Days on Therapy	2	Number of Patients
4	884	4	Sprycel	3rd Line	3	2	749	Days on Therapy	3	Number of Patients
5	884	5	Bosulif	4th Line	1	4	6	Days on Therapy	1	Number of Patients

The following columns are required:

- **nodeID** A value that uniquely identifies this node. This cannot be null and is something that should be generated as part of the data process. This field, along with parentNodeID, allow the application to determine the hierarchy of the rows returned by the query.
- **parentNodeID** Should be set to the nodeID value of this row's parent. If this row is a root node this value must be NULL.
- **name** The value of name will be displayed as the node's label next to its circle and in the header of its tooltip. See the optional column fullName below for more details.
- **value** This is the metric that is associated with each node. It must be a number. This determines how large the circle for this node will be. By default this will also be shown as the metric1Value in the tooltip
- **category** This field determines what color the node will be. Nodes with the same category will be given the same color. The categories will be displayed with their associated colors at the top of the chart in the legend.

The following columns are optional:

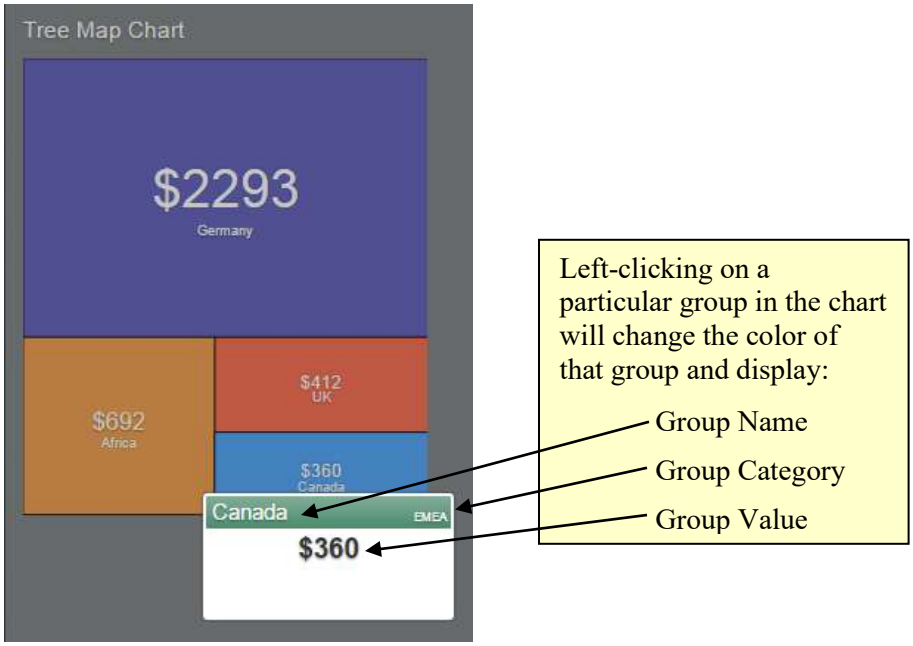
- **fullName** The title in the tooltip can come from either the column named 'fullName' (which is used first), or the column named 'name' (used when fullName does not exist). This use of two possible columns for the tooltip title allows the chart to contain an abbreviated name, but the tooltip to contain longer text.
- **metric1Label** The label that will be displayed next to the first metric in the tooltip.
- **metric1Value** The value that will be displayed next to the first metric in the tooltip. If this is not set the value field will be used.
- **Metric2Label** The label for the second tooltip metric.
- **Metric2Value** The value for the second tooltip metric.

Note that by default the Journey Tree will initially display with all nodes whose parentNodeID value is NULL. Mouse clicks on a node will expand the node.

Tree Map – Category 04

APDeveloper Report 01:

Section: Charts
Container: Example 01 – Tree Map Chart
Stored Proc: rspChartsCategory04



This Chart demonstrates a Tree Map Chart, which is defined by the following attributes in AgileM Studio.

Attribute List - Component (LayoutId=1054): Tree Map Chart Component

Attribute	Value	Value List
HideLegend	true	
Type	d3chart	
ShowSelected	#4381bf	

Note that the ShowSelected attribute can contain a HTML color code, which will be the color that the group will change to when it is selected.

The stored proc result set is shown below.

	layoutID	name	size	type	band1Color	label	metric1Format	metric1Value	category	subLabel
1	1054	Germany	2293	tree	#4d4d8c	$2293	USD	2293	EMEA	Germany
2	1054	UK	412	tree	#bf5843	$412	USD	412	EMEA	UK
3	1054	Africa	692	tree	#bf8143	$692	USD	692	EMEA	Africa
4	1054	Canada	360	tree	#4d8c6d	$360	USD	360	EMEA	Canada

The following columns are required:

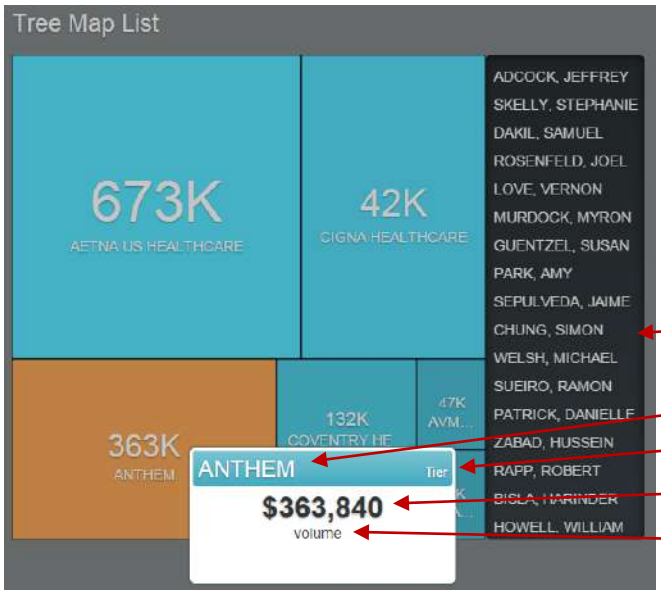
- **name** The value of name will be displayed in the group it represents and in the header of its tooltip.
- **size** This value determines the size of each group in the chart.
- **type** The type must be set to tree.

The following columns are optional:

- **band1Color** Controls the color of each group.
- **label** The value that is displayed in each group.
- **metric1Format** The format of the metric 1 value in the tooltip.
- **metric1Value** The value that will be displayed next to the first metric in the tooltip. If this is not set the size field will be used.
- **category** This value is seen in the top right corner of the tool tip.
- **subLabel** This value is displayed under the label in each group.

APDeveloper Report 01:

Section: Charts
Container: Example 02 – Tree Map List
Stored Proc: rspChartsCategory04
Component Stored Proc: rspChartsTreeMapList



Left-clicking on a particular group in the chart will change the color of that group and display:

- Names associated with that groupID
- Group Name
- Group Category
- Group Value
- Group Value Label

This Chart demonstrates a Tree Map List, which is defined by setting the Type attribute to 'treemaplist'. The following attributes are set in AgileM Studio.

Attribute List - Component (LayoutId=1056): Tree Map List Com

Attribute	Value	Value List
StoredProcedure	rspChartsTreeMapList	
Type	treemaplist	
ShowSelected	#bf8143	

Note that there is a stored proc at the component and the category level. The category stored procedure result set is shown below.

	layoutID	type	name	size	groupID	label	subLabel	band1Color	metric1Value	metric1Label	metric1Format	category
1	1056	tree	AETNA US HEA...	673164.38	1	673K	AETNA US HEA...	#43abbf	673164.38	volume	USD	Tier
2	1056	tree	CIGNA HEALTH...	427664.45	6	42K	CIGNA HEALTH...	#43abbf	427664.45	volume	USD	Tier
3	1056	tree	ANTHEM	363840.38	2	363K	ANTHEM	#43abbf	363840.38	volume	USD	Tier
4	1056	tree	COVENTRY HE...	132539.10	7	132K	COVENTRY HE...	#43abbf	132539.10	volume	USD	Tier
5	1056	tree	CAPITAL DISTR...	60900.52	4	60K	CAPITAL DISTR...	#43abbf	60900.52	volume	USD	Tier
6	1056	tree	AVMED HEALT...	47581.28	3	47K	AVMED HEALT...	#43abbf	47581.28	volume	USD	Tier
7	1056	tree	CHAMPVA DEP...	46358.41	5	46K	CHAMPVA DEP...	#43abbf	46358.41	volume	USD	Tier

The following columns are required:

- **type** The type must be set to tree.
- **name** The value of name will be displayed in the group it represents and in the header of its tooltip.
- **size** This value determines the size of each group in the chart.
- **groupID** This value uniquely identifies this group. This value cannot be null and is something that should be generated as part of the data process. This value determines what should be returned in the name list when a group is selected.

The following columns are optional:

- **label** The value that is displayed in each group.
- **subLabel** This value is displayed under the label in each group.
- **band1Color** Controls the color of each group.
- **metric1Value** The value that will be displayed next to the first metric in the tooltip. If this is not set the size field will be used.
- **metric1Label** The label of the metric 1 value in the tooltip.
- **metric1Format** The format of the metric 1 value in the tooltip.
- **category** This value is seen in the top right corner of the tool tip.

The component stored procedure result set is shown below.

	name	groupid
1	SAAVEDRA, OSWALD	1
2	FLATT, JAMES	1
3	SOTELO, TIFFANY	1
4	LEWIS, DONALD	1
5	PENCE, JACK	1
6	POBI, KWABENA	1
7	WRIGHT, ALISON	1

Looking at the stored procedure more carefully, the developer can notice an additional variable @vchGroupID is declared.

```
--  Populate @tblFilterNameValue with the Filter Name/Value pairs
    DECLARE @tblFilterNameValue as tblFilterNameValue
    INSERT INTO @tblFilterNameValue EXEC rspParseXMLFilterString @filterTableXML
    DECLARE @vchGroupID VARCHAR(10) = (SELECT dbo.GetFilterValue(@tblFilterNameValue, 'groupID'))
```

This filter is used to display the correct names when a group is selected by the user. It is a system generated filter that is passed to the component stored procedure by the system at click time.

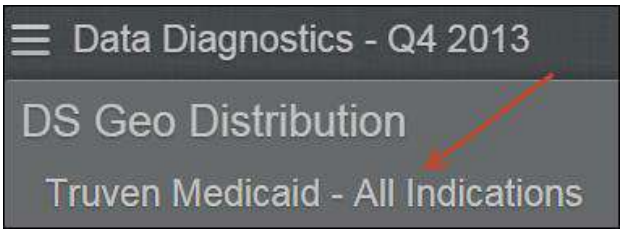
Free Form

Free Form Components, as their name suggests, support a very flexible layout of text and graphics. While the presentation of data in Grids and Charts is well defined and very controlled, Free Forms offer the full formatting power of HTML - so data can be sized, formatted, supplemented with graphics, etc. to meet most any need.

Mobile v2 Free Forms also support Dot Liquid technology – a templating system that has powerful formatting capabilities.

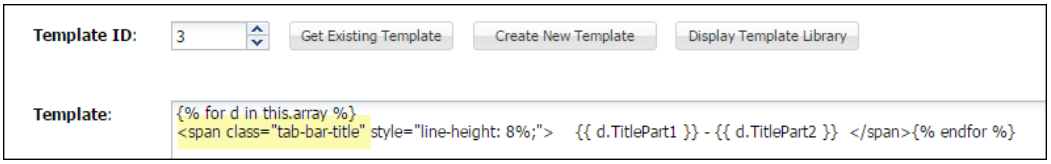
Free Forms as Labels

Note that Free Forms can be used to look like labels. This is accomplished by setting the BackgroundCls attribute = 'charttabpanel-title'. Following is an example of a Free Form that looks like a Category label.



The advantage to using a Free Form as a label is that now a Filter can adjust the value of the Free Form. In this case a Category title will now change with the Filter selections.

Note that for the Free Form to have the same appearance as a Category title, it's class should be set to 'tab-bar-title' as seen in the following Dot Liquid Template:



If the need was to have the Free Form have the appearance of a Section title, than use the tag <h2>.

APDeveloper Report 01:

Section: Free Form
Container: Free Form Example 01
Stored Proc: rspFreeFormExamples_Category01

Background:

All the formatting work for this Free Form is actually done by Dot Liquid. Note the table tblCfgDLHtmlTemplate which consists of the primary key TemplateID column and a Template column. The Template column contains the free text string that provides formatting information.

```
SELECT TOP 1000 [TemplateID]
      ,[Template]
  FROM [V2BIDeveloper_AgileMv2_RPT].[dbo].[tblCfgDLHtmlTemplate]
```

Results | Messages

TemplateID	Template
1	{% for d in this.array %} <table class="freeform-tbl" style="width: 100%;"> <tbody style="text-align: center;"

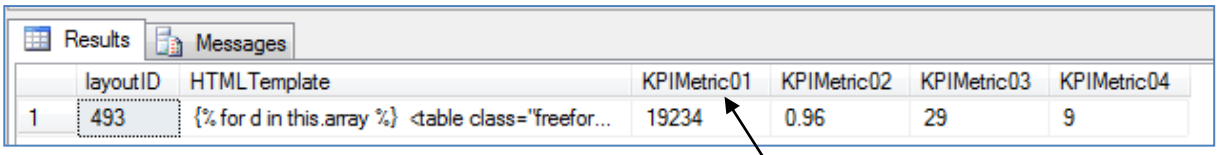
Following is the contents of the template column (formatted for ease of viewing):

Row 1

Row 2

```
{% for d in this.array %}  <table class="freeform-tbl" style="width: 100%;">
  <tbody style="text-align: center;">
    <tr>
      <td>
        <span class="callout growth" style="15px;">{{d.KPIMetric01 | Number: '$###,###' }}</span>
      </td>
      <td>
        <span class="callout growth" style="15px;">{{d.KPIMetric02 | Number: '###%' }}</span>
      </td>
      <td>
        <span class="callout" style="15px;">{{d.KPIMetric03 | Number: '###' }}</span>
      </td>
      <td>
        <span class="icn main-progress decline" style="vertical-align: top; color: red">
      </span>
        <span class="callout decline" style="15px;"> {{d.KPIMetric04 | Number: '###' }}</span>
      </td>
    </tr>
    <tr>
      <td>Total Current Payout</td>
      <td>Current Attainment</td>
      <td>Days Left in<br>Current Payout Period</td>
      <td>Current Territory Rank</td>
    </tr>
  </tbody>
</table>{% endfor %}
```

Note that the stored procedure returns the following record set:



	layoutID	HTMLTemplate	KPIMetric01	KPIMetric02	KPIMetric03	KPIMetric04
1	493	{% for d in this.array %} <table class="freefor...	19234	0.96	29	9

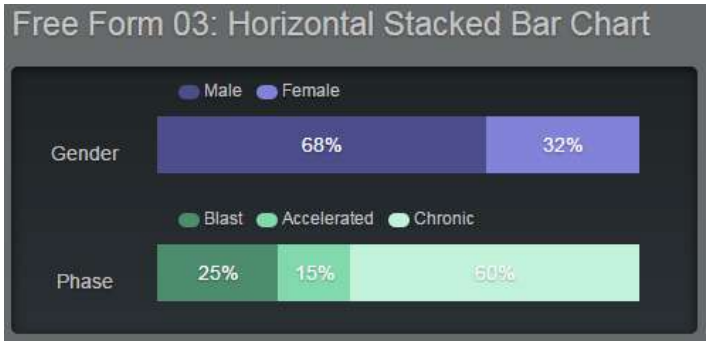
Reviewing both the template and the record set, note the following:

- The template provides formatting information and consists of HTML, particularly the <table>, <tr>, and <td> tags.
- The template also provides a mapping of columns in the record set with formatting instructions. For example, the column KPIMetric01 has a formatting mask of '$###,###'.
- The record set provides the data.

See the Dot Liquid section for more discussion of Dot Liquid.

APDeveloper Report 01:

Section: Free Form
Container: Free Form Example 01
Stored Proc: rspFreeFormExamples_Category01



Background:

Free forms can be utilized to build a limited number of bar charts, though these tend to be employed less frequently. In general, they are highly customizable and their construction is a bit more involved than a conventional text and number type display. Since these are free forms, Tool Tips and other attributes of Charts and Grids are not compatible features.

Results | Messages

	layoutID	HTMLTemplate	Attribute	AttributeName1	AttributeName2	AttributeName3	AttributeValue1	AttributeValue2	AttributeValue3	AttributeColor1	AttributeColor2	AttributeColor3
1	1167	{% for item in...	Gender	Male	Female	NULL	0.6818	0.3182	NULL	#4d4d8c	#8282d9	NULL
2	1167	{% for item in...	Phase	Blast	Accelerated	Chronic	0.2500	0.1500	0.6000	#4d8c6d	#82d9ad	#c2f2da

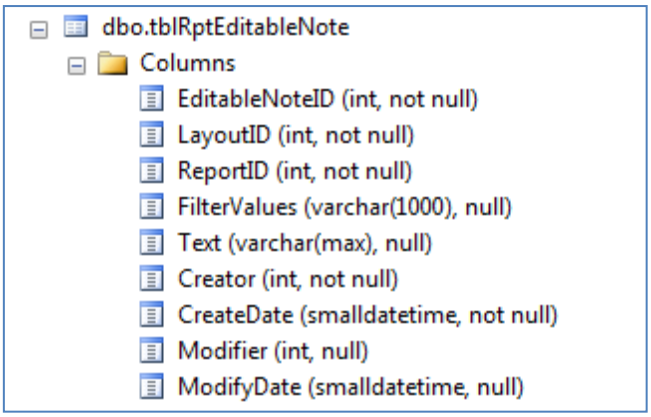
Note the following differences in the record set:

- Each horizontal stacked line generally has three different corresponding Attribute Values – a Name, Value, and Color
- The number of segments in the stacked bar relates to the number of attributes
- Each extra bar returns an additional record

Editable Note

Editable Notes support the clients ability, at run time, to add text to a Component. When the user clicks the editableNote they may then type text in the editableNote. This text is then automatically saved and displayed the next time the Component is presented to any user. Like all Components, the editableNote Component exists inside a Container.

The user entered text is saved in the following RPT table.



Note that the BI developer does not have to write a stored proc for this Component. The v2 application has full responsibility for both reading from and writing to the tblRptEditableNote table. However, note that if a Category consists of only editableNotes, then the StoredProcedure attribute for the Category must be populated with an actual stored procedure even though that stored procedure need not return any result set.

Comments regarding the columns in tblRptEditableNote:

- **EditableNoteID** is an identity column.
- **LayoutID** refers to the layoutID of the editableNote Component.
- **ReportID** is a bit redundant with the presence of LayoutID but is provided as a convenience column for possible future use.
- **FilterValues** is a concatenation of the Category Filter Name and Filter Value pairs. For example, if a Category has a Geography Filter and a Time Period Filter, FilterValues might equal 'GeographyNE12345TimePeriodC3mo'.

  The purpose of this value is to allow the text of an editableNote for one set of user selected Filters to be different than the text for a different set of Filter values.

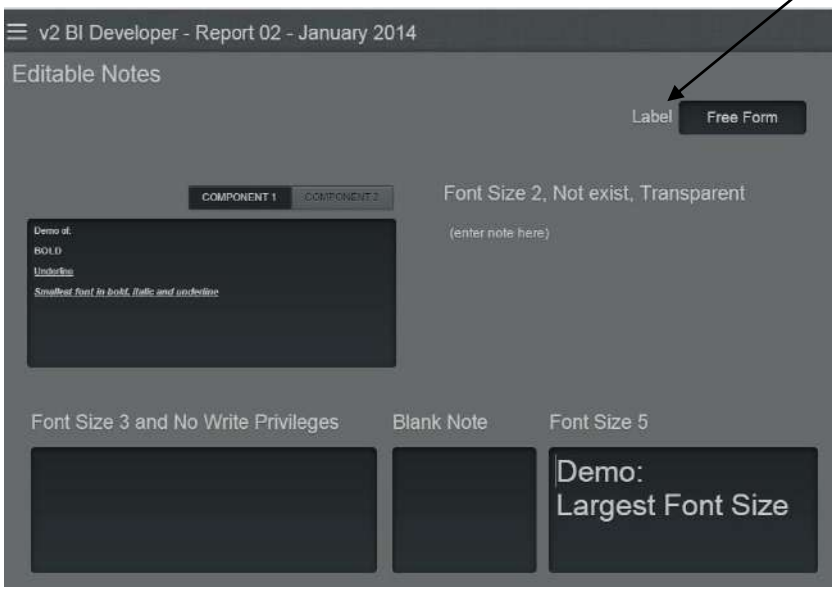
- **Text** is the actual user entered text. Note that limited rich text formatting is supported.

When starting a new reporting period clients may optionally have the tblRptEditableNote table refreshed so that there are no tblRptEditableNote rows (in which case new notes would need to be entered), or they may retain the previous reporting period tblRptEditableNote rows (in which case notes are "re-used").

Note that this decision can vary from one report to another within the project (and this is why tblRptEditableNote has a ReportID column). The DB will need to be informed about which action is appropriate on a report by report basis.

APDeveloper Report 02:

Section: Editable Notes
Container: (several Containers in the Section)
Stored Proc: (no stored procedure is required)



Trick - use an editableNote as a label
Since Container labels always appear above the Container, you could place an editableNote to the side of a Container, set its attributes BackgroundCls to 'charttabpanel-title' and WritePrivilages to blank. Now it will display like a label but can't be edited.

Caveat – if the table tblRptEditableNotes is refreshed every reporting period the text will have to be re-entered (unless the db restores selected rows).

EditableNotes exist inside Containers just like all Components. First create a Container which may then have one too many editableNote Components.

Besides the common Component Attributes of: DisplayOrder, Title and Type, editableNotes are supported by the following Attributes.

Attribute	Required	Function
ApplyFilter	Yes	When reading/writing Editable Notes, False will not consider the values of a Filter so that the same Note will appear regardless of the Filter selections. True will key Note values using all Filter values so the Note will vary with the Filter values.
BackgroundCls	Yes	'Free-form inset' will apply a background to the Note identical to the background of a Free Form. 'charttabpanel-title' will give the Note a transparent background.
FontSize	Yes	Size of the font in the Editable Note. 1=very small, 5=very large
WritePrivilages	Yes	A comma separated list of the UserType IDs that should have Editable Note write privileges.

Note that text is entered using a physical keyboard (which may be on attached to a desktop computer or an iPad with a physical keyboard). Text is entered using the soft keyboard of an iPad.

Also note that if text is first entered into Word and some formatting is applied there: font selection, size, bold, italic, underline the text can be copied/pasted into the editableNote and the formatting will be properly saved and re-displayed.

Each editableNote Component in the screen shot above is set up to display a certain feature of editableNotes.

Component 1: This shows the use of the smallest font and that there is support for some rich text formatting capability. With a physical keyboard use Ctrl-B to bold text, Ctrl-I to italicize text and Ctrl-U to underline text. Using the soft keyboard of an iPad use standard iPhone editing procedures to enter rich text. Also note that changing the Filter will change the text of this editableNote.

Component 2: The ApplyFilter value of this editableNote is set to False, so when changing the Filter the text remains constant.

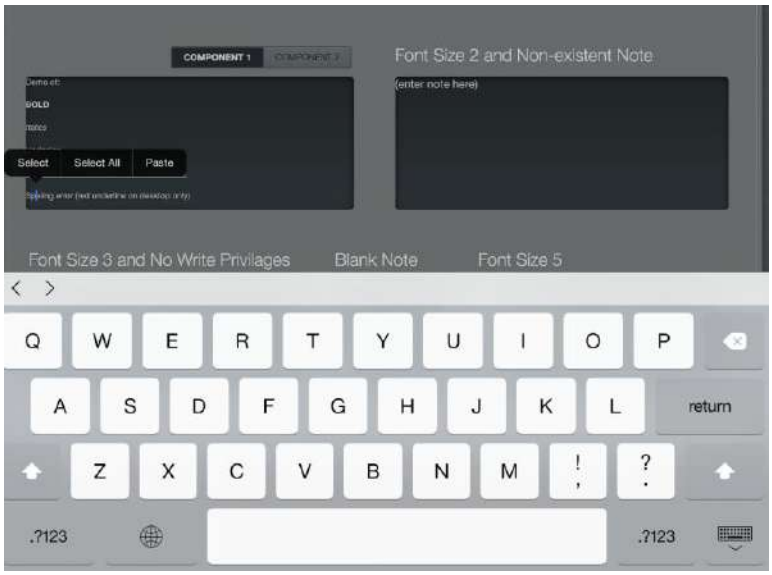
Font Size 2, Not Exist, Transparent: This demonstrates the FontSize of 2. This also demonstrates that when an editableNote has no tblRptEditableNote row the text '(enter note here)' displays by default. Also, since the Background attribute = False, the text will display without any background and might be more appropriate for a label.

Font Size 3 and No Write Privileges: When the User Type ID of the active user is not in the WritePrivileges Attribute, then it is not possible to edit this Component. This might be appropriate, for example, when home office users are give write privileges, but field users are not.

Blank Note. A demonstration that, in contrast to a non-existent tblRptEditableNote row which will have the default text of '(enter note here)', a Component may also have a tblRptEditableNote row whose text is blank. In this case a note will display as blank.

Font Size 5. A demonstration of the largest size font.

Following is a screen shot of the editing process on an iPad using a soft keyboard. Note that there is a known limitation when editing on the iPad without a physical keyboard – vertical scrolling does not work. A finger tap on the editableNote brings up the soft keyboard before any vertical scrolling occurs. Vertical scrolling does however work when using a physical keyboard (on either a desktop or iPad).



Additional comments regarding editing :

1) Notes can be copy/pasted from Word to an editableNote. This will then support additional formatting, (beyond just bold, italic and underline), such as: font type change and font size change.

2) The soft keyboard can obscure some of the editable note area.

3) A hard keyboard, on either the iPad or a full computer, is generally recommended for actually editing notes.

4) Full screen mode generally makes editableNotes more pleasing to work with.

Controlling Multiple Components with Tabs

AgileM now has the ability to allow developers to control multiple components at the same time with a single set of tabs. There are inputs that are required within AgileM Studio as well as within the category stored procedure itself that makes this possible. The following is a simple example that will be controlling a chart and a gridpanelcollapse, but, you can control other combinations of components as needed.

APDeveloper Report 01:

Section: Tables
Container: Table Example 02-A and Table Example 02-B
Stored Proc: Rsptablescategory02

The following Attributes are required:

- **Type:** The type will be whatever kind of component is being displayed
- **Title:** When used on the component level, it is the text that is displayed in a button. Required since there is more than one component that is displayed in this container. This value will also be referenced in the category stored procedure.
- **SiblingLayoutID and SiblingParentLayoutID:** These IDs are the container LayoutIDs that are linked together.
    - It is possible for a container to have multiple SiblingLayoutIDs. This can be achieved by separating each ID in a comma delimited format.
    - Containers may only be linked to one SiblingParentLayoutID. The SiblingParentLayoutID ultimately determines where the tabs will be placed.

When creating the stored procedure for multiple components being controlled by tabs, you must input "linkCategory". It will correspond to the title that is defined at the component level for the other linked container. In this particular example there is one result set that is assigned to a particular linkCategory as seen below:

```
Component: Linked Table
      SELECT
            layoutID = 1411,
            linkCategory = 'Linked Chart 1',
            DataMonth,
            Example01TRx
      FROM    tblRptLinkedChart
      WHERE   DataMonth IN ('JAN','FEB','MAR','APR','MAY','JUN')

      SELECT
            layoutID = 1411,
            linkCategory = 'Linked Chart 2',
            DataMonth,
            Example01TRx
      FROM    tblRptLinkedChart
      WHERE   DataMonth IN ('JUL','AUG','SEP','OCT','NOV','DEC')
```

	layoutID	linkCategory	DataMonth	Example01TRx
1	1411	Linked Chart 1	JAN	10
2	1411	Linked Chart 1	FEB	5
3	1411	Linked Chart 1	MAR	2
4	1411	Linked Chart 1	APR	1
5	1411	Linked Chart 1	MAY	15
6	1411	Linked Chart 1	JUN	20

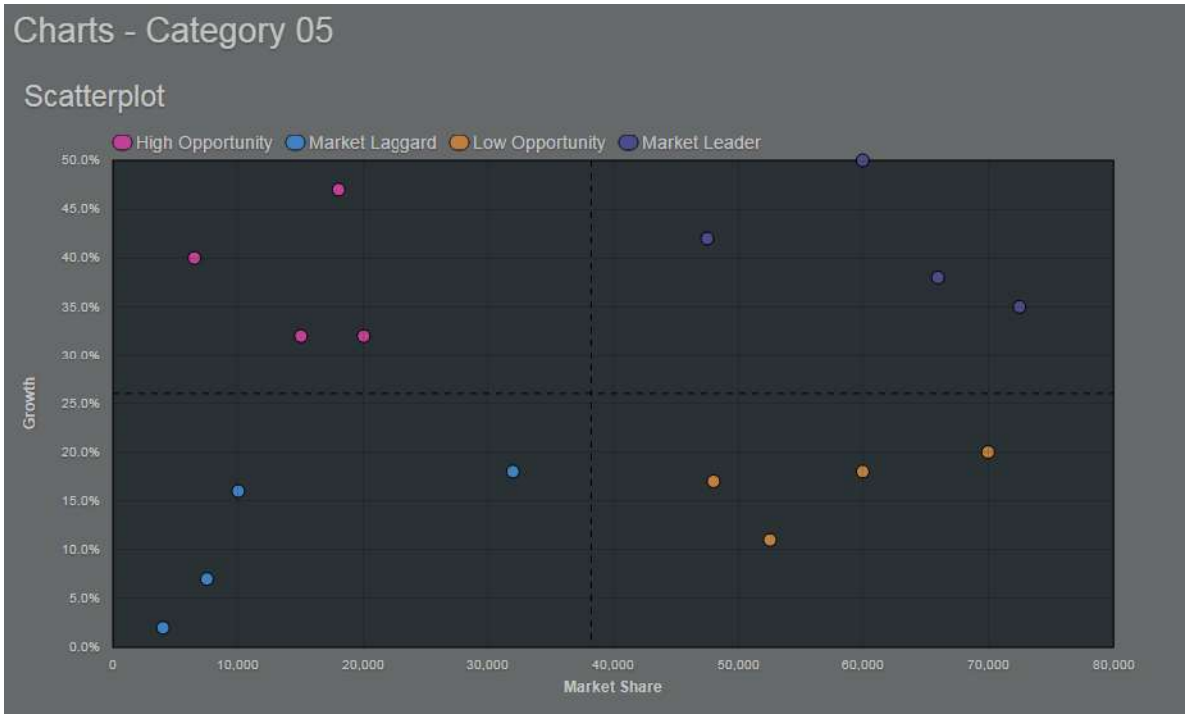
	layoutID	linkCategory	DataMonth	Example01TRx
1	1411	Linked Chart 2	JUL	15
2	1411	Linked Chart 2	AUG	4
3	1411	Linked Chart 2	SEP	10
4	1411	Linked Chart 2	OCT	7
5	1411	Linked Chart 2	NOV	2
6	1411	Linked Chart 2	DEC	9

Scatterplot

Scatterplots are designed to separate data into four quadrants. These quadrants determine how data points compare to each other in two different metrics. Two common metrics that are visualized on a scatterplot are growth and market share. Data points that score well on both of these metrics would usually be the ones that should be focused on most heavily by the end user.

APDeveloper Report 01:

Section: Charts 2
Container: Scatterplot
Stored Proc: Rspchartscategory05



The following Attributes are required:

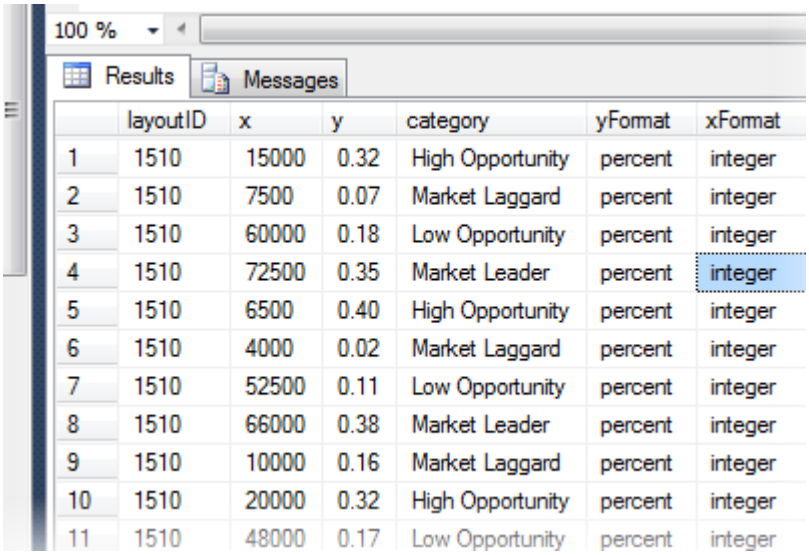
- **Type:** For scatterplots, it will be "scatter"
- **Quadrants:** This must be set to "true" in order to separate the data points into their four quadrants by weighted average.
- **XAxisTitle and YAxisTitle:** While not technically required, they are recommended so the end user know which metrics are being measured.

Notes:

- The Scatter plot does not support removal of a series by clicking the series in the legend.

When creating the stored procedure for scatterplots, the necessary inputs specific to the particular component are "x' and "y." A simple example and its output can be seen below:

```
        ------------------- START OF AUTO GENERATED NON-PAGING GRIDS
        SELECT layoutID = 1510,
               x = MarketShare,
               y = MarketGrowth,
               category = Category,
               yFormat = 'percent',
               xFormat = 'integer'
        --select *
        FROM   dbo.tblRptScatterplot
    -- End TRY - across entire stored proc
    END TRY
```



100 %

Results | Messages

	layoutID	x	y	category	yFormat	xFormat
1	1510	15000	0.32	High Opportunity	percent	integer
2	1510	7500	0.07	Market Laggard	percent	integer
3	1510	60000	0.18	Low Opportunity	percent	integer
4	1510	72500	0.35	Market Leader	percent	integer
5	1510	6500	0.40	High Opportunity	percent	integer
6	1510	4000	0.02	Market Laggard	percent	integer
7	1510	52500	0.11	Low Opportunity	percent	integer
8	1510	66000	0.38	Market Leader	percent	integer
9	1510	10000	0.16	Market Laggard	percent	integer
10	1510	20000	0.32	High Opportunity	percent	integer
11	1510	48000	0.17	Low Opportunity	percent	integer

Bar Chart

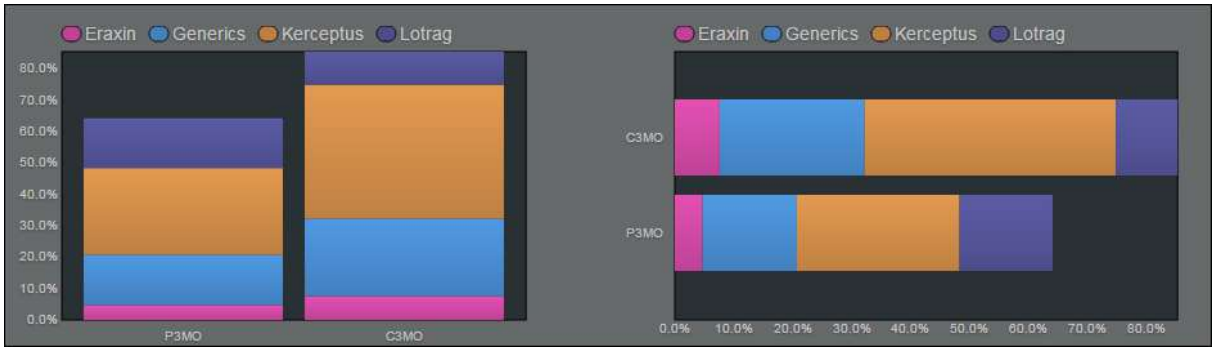
Charts of type 'bar' are designed to offer four variations:

1) Vertical and stacked.
2) Vertical and not stacked.
3) Horizontal and stacked.
4) Horizontal and not stacked.

Note that the dataset for all four Charts is identical and that Orientation (vertical or horizontal) and Stacked (true or false) are set as attributes and not in the data.

APDeveloper Report 01:

Section: Charts 2
Category: Charts2 – Category1
Stored Proc: rspCharts2Category01



Note that the record sets for both Charts above is identical and that Attributes control Orientation and Stacked.

	layoutID	category	y	format	name	band1Color	metric1Value	metric1Format	metric1Label
1	1513	P3MO	0.0470	percent	Eraxin	#D05F49	0.0470	percent	Market Share
2	1513	P3MO	0.1600	percent	Generics	#515194	0.1600	percent	Market Share
3	1513	P3MO	0.2760	percent	Kerceptus	#C68645	0.2760	percent	Market Share
4	1513	P3MO	0.1580	percent	Lotrag	#5C7084	0.1580	percent	Market Share
5	1513	C3MO	0.0760	percent	Eraxin	#D05F49	0.0760	percent	Market Share
6	1513	C3MO	0.2460	percent	Generics	#515194	0.2460	percent	Market Share
7	1513	C3MO	0.4260	percent	Kerceptus	#C68645	0.4260	percent	Market Share
8	1513	C3MO	0.1050	percent	Lotrag	#5C7084	0.1050	percent	Market Share

	layoutID	category	y	format	name	band1Color	metric1Value	metric1Format	metric1Label
1	1583	P3MO	0.0470	percent	Eraxin	#D05F49	0.0470	percent	Market Share
2	1583	P3MO	0.1600	percent	Generics	#515194	0.1600	percent	Market Share
3	1583	P3MO	0.2760	percent	Kercep...	#C68645	0.2760	percent	Market Share
4	1583	P3MO	0.1580	percent	Lotrag	#5C7084	0.1580	percent	Market Share
5	1583	C3MO	0.0760	percent	Eraxin	#D05F49	0.0760	percent	Market Share
6	1583	C3MO	0.2460	percent	Generics	#515194	0.2460	percent	Market Share
7	1583	C3MO	0.4260	percent	Kercep...	#C68645	0.4260	percent	Market Share
8	1583	C3MO	0.1050	percent	Lotrag	#5C7084	0.1050	percent	Market Share

Attribute List - Component (L

Attribute	Value
Orientation	vertical
Stacked	true

Heat Map

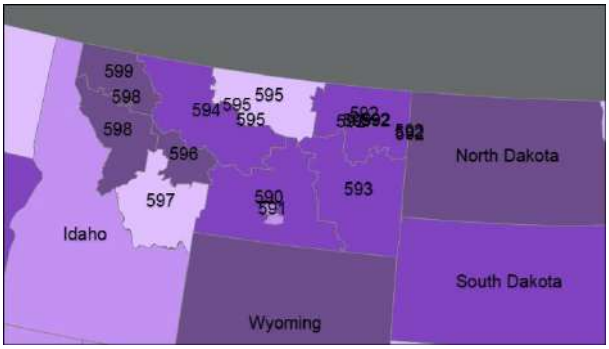
Heat Maps provide a visual representation of data on a map.

The data that is represented could be, for example: market share, sales volume or TRx. There are no restrictions on what datum of information is displayed. The maps that are used can be any country, state, zip code, etc. for which a shape file is available. (More on shape files below). Commonly supported maps include: US zip codes, US states, the United Kingdom and England Regions (the equivalent of US States).

The user interface features supported by Shyft Heat Maps are:

- **Data driven color:** The colors of a maps divisions, for example states, is data driven.
- **Multiple Levels**: A Map Component can have multiple levels. For example the top most level map could be a heat map of US Areas. A double click would then display a heat map for Regions and a second double click would then show Territories.
- **Drill down**: Similar to multiple levels – in that a double click presents a different map. However, with drill down, there is a required data link from the upper map to the lower map. For example, a double click on Montana would then "drill down" to a detailed zip code heat map, but with only Montana detailed at the zip code level – all other states would continue to be represented as full states. Note that the system will automatically zoom and pan to the selected division.

  In this case a double click on Montana, "drilled down" to a zip3 detail on just Montana. The other states are not detailed.



- **Merging**: Merging supports the ability to "merge" smaller shapes into one larger shape. For example, using the standard ZipTerr table, a set of Regional zip codes can be merged to form the "Great Lakes Territory".
- **Custom labeling**: Heat maps come with a default label for each division. However the system supports the optional ability to override the default label including the ability to have data/filter driven labels. This would allow, for example, labeling a Region as: "Northeast-5% Market Share".
- **Tool tips**: The standard AgileM tool tips are fully supported.
- **Legends**: Legends automatically display.
- **Scaling**: Maps will automatically scale to their configured Container size.
- **Country agnostic**: Nothing in the architecture limits map functionality to only US maps. Maps are limited only by the availability of shape files (as discussed below).

Background

To implement Shyft Heat Maps it is helpful to understand the following:

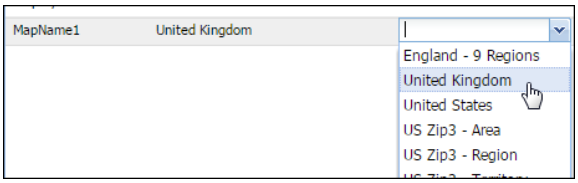
Shape Files

Shape files form the foundation for the display of heat maps. They define a polygon, for example a nation, state or zip code area with latitude and longitude data. Since the resulting display is a vector, and not a bit map, it can be scaled.

Shape files are available from a variety of sources including, governments, public domain sites and for profit companies. Large divisions, like countries and their first level division, may be readily available for free. However, zip code shape files are only available for a rather significant fee.

An important concept to understand about shape files is that they form the foundation of v2 Heat Maps. ***Shyft can only draw those shapes for which shapes files exist****. v2 Heat Maps will never be able to draw arbitrary shapes, for example, "the northern half of California", unless a shape file with such a shape exists.*

Shape files require considerable one-time pre-processing that will be the responsibility of the Product group. The pre-processed version of the shape file will exist in the v2 RPT/tblMaps table and will be selected in AgileM Studio at map configuration time.



ZipTerr table

As noted above the ZipTerr table defines a mapping between each zip code and which Area, Region, Territory, etc. it lives in. One of the shape file pre-processing steps is to perform the "merge", for example taking zip code 12345, 12346, 12347, etc. and merging them into the "Central Connecticut" Territory for presentation as its own map division.

Since the ZipTerr table can change regularly, this merging process will occur with each data run. While it will be the responsibility of the person doing the regular data run to execute the stored proc that performs the merge, it's insightful for the implementer to understand such a process exists.

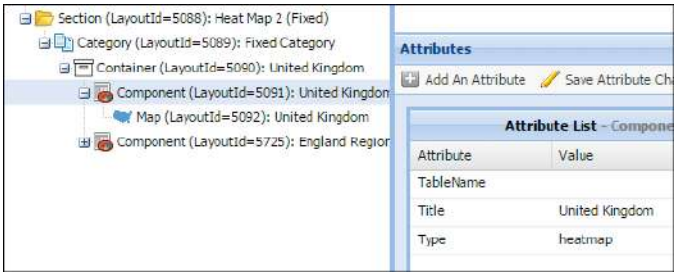
AgileM Studio Configuration

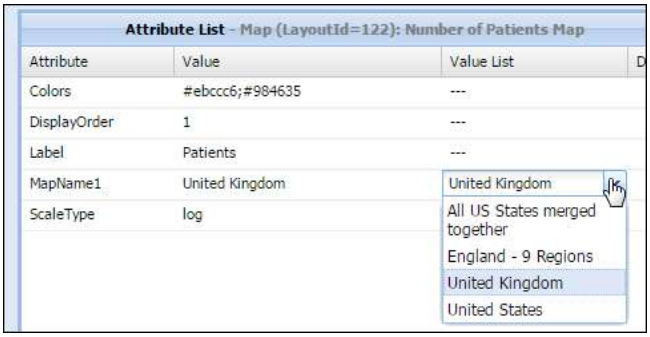
APDeveloper Report 02:

Section: Heat Map 2
Container: United Kingdom
Stored Proc: rspMapCategorySP_06

Heat Map – basic configuration

The most basic map has just a single level (i.e. double clicking does not present another map), and is not a drill-down map (double clicking does not zoom into the selected division).





Configuration requirements:

- A Component layout type whose Type attribute = heatmap
- A Map layout type that is a child of the Component.
- For the Map layout type, set the attributes:
  - **Colors**. A range of colors that will be used to color the map divisions. Details below.
  - **DisplayOrder**. DisplayOrder is required and refers to the maps "level". In the basic case of just a single map, set DisplayOrder=1.
  - **Label**. An optional free text string that will appear at the top of the legend.
  - **MapName1**. Use the Value list to choose from the list of shape files.
  - **ScaleType**. Controls how colors are automatically determined and mapped to the data. Select either 'log' or 'linear'. Details below.

APDeveloper Report 02:

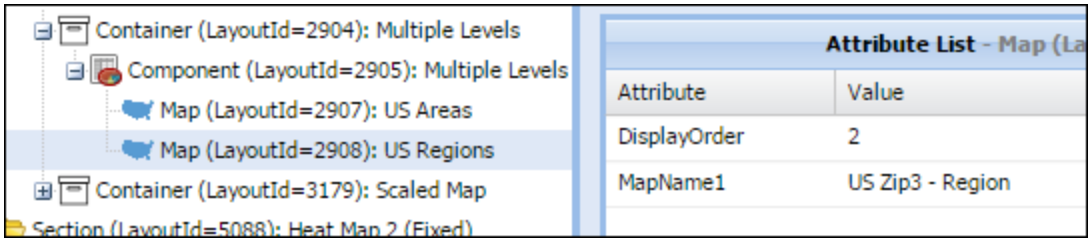
Section: Heat Map 1
Container: Multiple Levels
Stored Proc: rspMapCategorySP_05

Heat Maps – multiple levels

Maps with multiple levels support double clicking one map to go to the next map. A double click on the last map takes the user back to the first map.

Configuration is identical to the basic configuration as noted above, except additional Map layouts are added to the heatmap Component.

The Map layout attribute DisplayOrder controls the order maps are presented in.



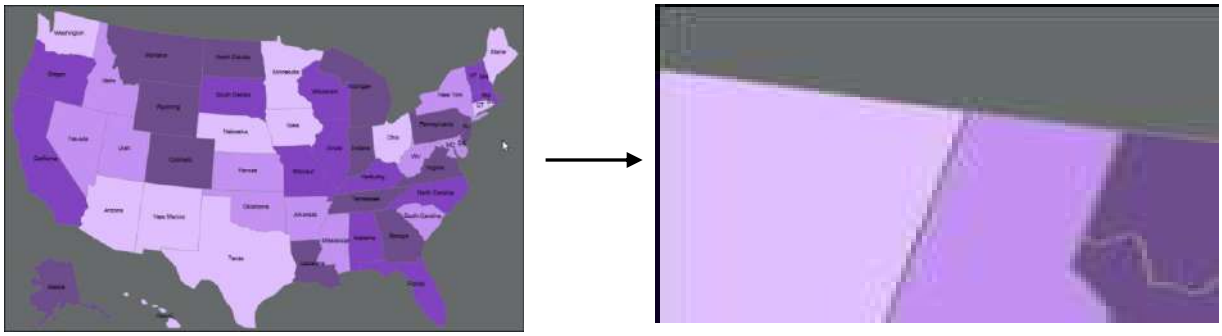
APDeveloper Report 02:

Section: Heat Map 1
Container: Drill Down
Stored Proc: rspMapCategorySP_04

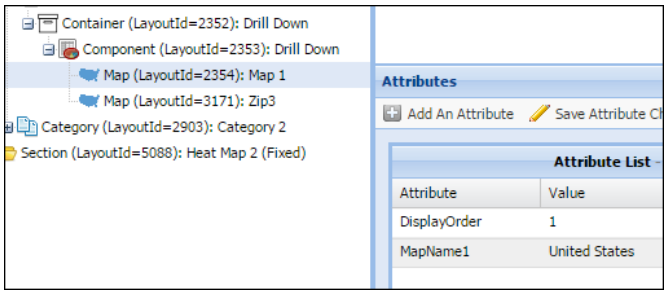
Heat Maps – drill down

Drill down maps support the ability to zoom into a specific division within a map.

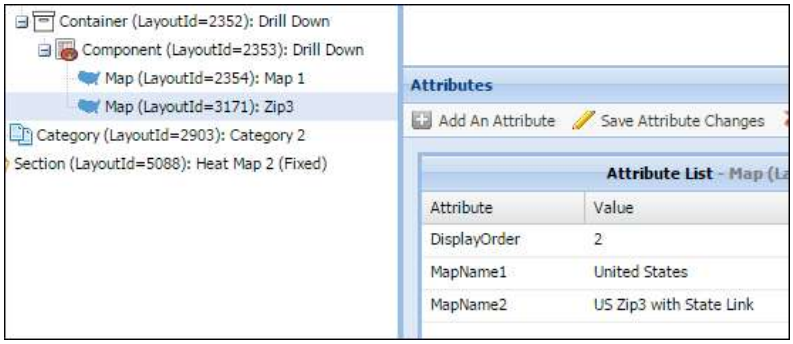
In the following case, double clicking on Montana "drills down" by zooming in on Montana and providing detail for only the state that was double clicked.



Note the configuration for the US Map. It's DisplayOrder=1 and there is the attribute MapName1.



Note the configuration for the "drill down" map. It's DisplayOrder=2 and it contains attributes MapName1 and MapName2. The value of MapName1 must be identical to the value of MapName1 above. The value of MapName2 will point to a shape file that has been specially developed to support drill down. Implementers will need to define the requested drill down functionality and then the Product group will be responsible for developing the modified shape file.

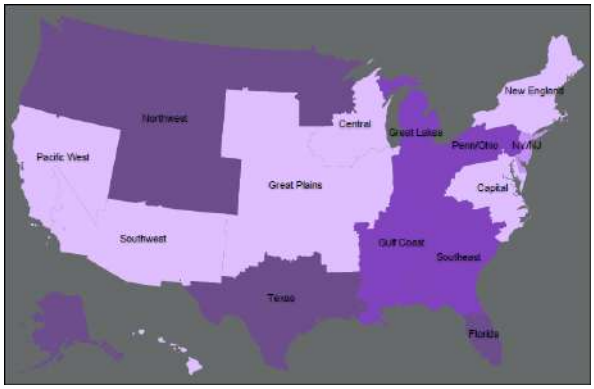


APDeveloper Report 02:

Section: Heat Map 1
Container: Multiple Levels
Stored Proc: rspMapCategorySP_04

Heat Maps – Merged Maps

Merged maps support the merging of many smaller shapes into one larger shape. For example a group of zip codes could be merged into one Territory. Or the New England states (Maine, New Hampshire, Vermont, Massachusetts, Connecticut and Rhode Island) could be merged into one New England Region.



The "definition" of a merge must be defined in a table. For example the following ZipTerr table defines, for each zip code, which Territory, Region and Area the zip code resides in.

	ZIPCode	TerritoryName	RegionName	AreaName
99	58790	Minneapolis, MN	Northwest	West
100	58792	Minneapolis, MN	Northwest	West
101	58793	Minneapolis, MN	Northwest	West
102	58794	Minneapolis, MN	Northwest	West
103	63951	Saint Louis West, MO	Great Plains	Central
104	63952	Saint Louis West, MO	Great Plains	Central
105	63953	Saint Louis West, MO	Great Plains	Central

When configuring non-merged maps the attribute MapName1 is set to a standard map – for example, US States or United Kingdom. But merged maps require that a custom map definition be created. This customization effort is a responsibility that will primarily be undertaken by the DBA and is fully documented in the AgileD Developers manual.

To get the customized map created, v2 Implementers should:

- Identify any merged maps in the report design.
- Determine which table defines the custom merged map-for example a ZipTerr table.
- Discuss with the DBA your need for merged map, who will then follow the set-up steps detailed in the AgileD Developers manual.

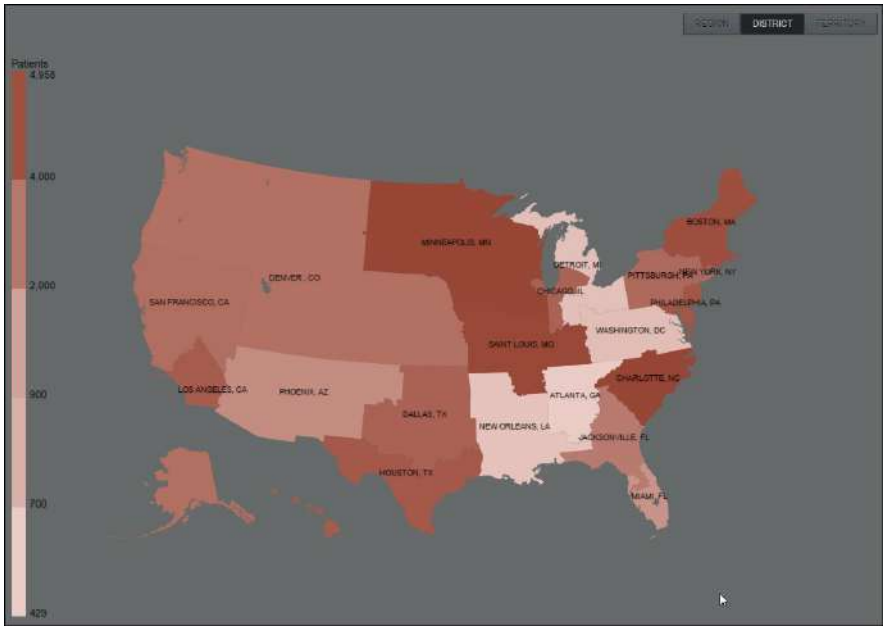
APDeveloper Report 02:

Section: Heat Map 3
Container: Region, District, Territory
Stored Proc: rspMapCategorySP_05

Another example of merged maps is based on US Zip codes.

Heat Maps – US Zip Code super groups

Heat map divisions may be "super groups" of US Zip codes. That is a client may define any set of zip codes to be in a division and there may be any reasonable number of divisions. (If the number of divisions gets over 100, then display times may be a bit slow or the labels of those divisions will overlap).



These zip code divisions can be defined in any table with a unique column of all zip codes associated with the desired divisions. This is typically found in a zipTerr table.

See the previous page, **Heat Maps – Merged Map,** for the additional responsibilities in implementing merged maps.

Stored Proc record sets

Following are the required and optional columns to be supplied to the heatmap Component.

- **layoutID.** Required. LayoutID of the Component whose type is 'heatmap'. Note this is not the layoutID of the actual map itself.
- **displayOrder.** Required. Maps back to the configuration attribute DisplayOrder.
- **mapNumber.** Required. For data associated with MapName1, then 1. For data associated with the optional MapName2, then 2.
- **Id.** Required. Used to label the divisions within the map. Values for each supported map are detailed below.
- **heatValue.** Required. Used to control color of the maps division.
- **shapeColor**. Required/Optional. If present will be used to color code the map division.

Required when the heatValue is zero. shapeColor must be set when the heatValue is zero. shapeColor may optionally be supplied for all non-zero heatValues as discussed below.

See the Colors configuration attribute for details on how the system can automatically determine a divisions color. But note that if shapeColor is present in the data setting it will override the system determined color.

Note that it is possible to provide shapeColor in the data set for a subset of the maps divisions and let the system assign color for the other divisions. This could be helpful if, for example, a selected state needed a special color, in which case code like the following could be used:

```
,shapeColor = CASE
                WHEN geo.GeographyName = 'MT' THEN '#ff0000'
                ELSE ''
              END
```

- **mapName.** Optional. Optionally provide mapName. mapName points to the shape file that will be used to actually draw the map. This value can also be supplied as the MapName1 or MapName2 configuration attribute value. The list of possible values is displayed in the AgileM Studio Value List for MapName1 and MapName2. If mapName is supplied as a column in the result set, then it will override the AgileM Studio configuration attribute value.

The advantage to being able to supply the map name in two locations is that if supplied as an attribute value it is hard coded and need not be supplied in the result set, but if supplied in the result set it will then be possible for the Heat Map to change based on the value of a Filter.

- **label.** Optional. If present will be used to label the divisions within a map (otherwise Id will be used). This will allow for labels to be data driven, for example: 'California - $123,456'. Also note that some rows may populate label (in which case it will be used to label the division), and other rows will leave label blank (in which case Id will be used to label the division).

- **legendLabelFormat.** Optional. Used to control the formatting of the values in the legend. Supported values are:
  - 'thousands' (for example, 12.345 would display as '12.34k')
  - 'millions' (for example, 12,345,678 would display as '12.34m')
  - 'decimal' (for two digits of precision)
  - 'comma' (for example, '123456' would display as '123,456')
- **Optional Tooltip columns.** The standard tooltip values are supported: MetricXValue, metricXFormat, metricXLabel and metricXChevron.

Additional Configuration Details

Following are details on selected configuration attributes.

Scale Type. Required. Set to either 'log' or 'linear'. Controls how the colors in the Color attribute are mapped to the heatValues in the maps record set.

When set to "log", then the Colors attribute must contain only two colors that determine the "high" and "low" color range. The application will then automatically map each heatValue, using a log scale, to a color that falls between the high and low color range. This can result in each maps division, (for example, state) possibly having a different color. Use of a log scale is helpful when the heatValues have a very wide range and/or there are outliers with particularly high or low values.

When set to "linear", then the Colors attribute can contain many colors with each color determining a "bucket" for the heatValues. This results in a "linear" mapping of each maps division falling into one of the specific color buckets. Use of a linear scale is appropriate when the heatValues are more tightly grouped.

Colors. Required. A semi-colon separated list of colors to be used in drawing the map.

The colors can be in either hex or rgb notation and the values must be semi-colon separated. For example: "rgb(12,34,56); rgb(78,90,12); rgb(34,56,78)", or "#bf5843; #c97260; #d38c7e". Note that the first color will be assigned to the lowest values, the last color will be assigned to the highest values and the colors should be in gradiant order. Note the color values are not comma separated, but semi-colon separated.

Note that when the ScaleType is set to "log", then only supply two colors which will set the "high" and "low" color values in the range. When the ScaleType is set to "linear", then several colors can be provided, with each determining a discrete bucket to place heatValues into.

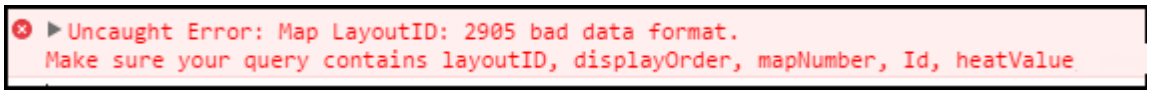
Also note that the data may optionally contain a 'shapeColor' column and if such a column is present and contains a value, that the data supplied shapeColor value will override the colors defined in the Colors attribute. See the ScaleType attribute Help for more details.

Additional Map information

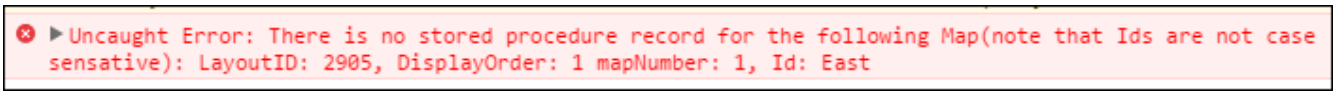
Console error messages

If a map fails to display, check the debugger Console window for the following messages:

1) Displayed if the record set is missing any required columns:



2) Displayed if the record set is missing a required ID. For example, if an Area map is expecting an Id of 'East', but the record set does not have an Id with the value 'East', the following will display:



ID's per Shape File

Each shape file has a set of required Id's. For example, the US State shape file requires that the record set have one row for each state. The following table presents the list of required Id's by shape file.

Note the special consideration when dealing with Merged Maps whose shapes are determined by merging, for example, multiple zip codes into a Territory shape. In this case, do a SELECT DISTINCT on the Territory column in the ZipTerr table. For each value returned, supply an Id with that value.

Shape File Name	Required Id's
US Zip Codes	Shyft supports merging US postal zip codes into super-groups of, for example, Region, District and Territory.
England - 9 Regions	East Midlands East of England London North East North West South East South West West Midlands Yorkshire and The Humber
United Kingdom	England Northern Ireland Scotland Wales
All US States merged together (useful for displaying the full United States as a single map without state divisions)	United States
Guam, Puerto Rico, Virgin Islands	Guam Puerto Rico United States Virgin Islands
UK CPRD	East Midlands East of England London North East North West South East Coast South Central South West West Midlands Yorkshire and The Humber

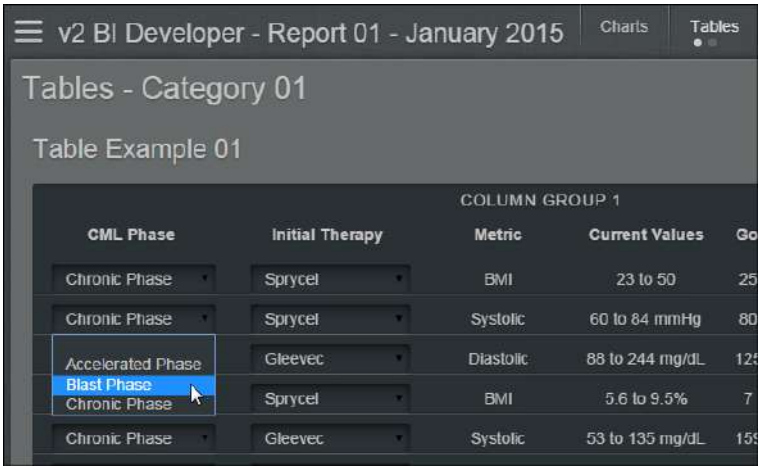
United States	Alabama Alaska Arizona Arkansas California Colorado Connecticut Delaware District of Columbia Florida Georgia Hawaii Idaho Illinois Indiana Iowa Kansas Kentucky Louisiana Maine Maryland Massachusetts Michigan Minnesota Mississippi Missouri Montana Nebraska Nevada New Hampshire New Jersey New Mexico New York North Carolina North Dakota Ohio Oklahoma Oregon Pennsylvania Rhode Island South Carolina South Dakota Tennessee Texas Utah Vermont Virginia Washington West Virginia Wisconsin Wyoming

In Line Editing

In line editing supports the ability of the user to modify data in real time through the use of a non-editable drop down list in a grid. Once the user makes a selection, the system automatically writes the change directly to the rpt table that populates the gridpanelcollapse Component and to a log table located in the app database.

See a fully functioning example here:

APDeveloper Report 01:
Section: Tables
Container: Table Example 01
Stored Proc: rspTablesCategory01



Notes regarding features and implementation:

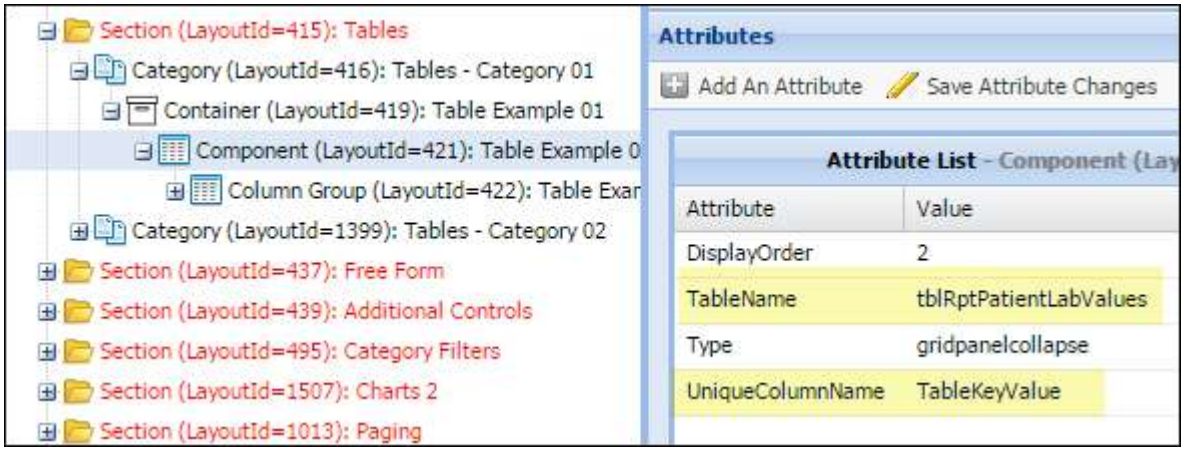
- **Non-editable list.** This is not a combo box (which also supports free text entry). Entries must come from the drop down list.
- **Drop down list.** The rpt table dbo.tblListOfValues fully controls the list and the order of possible selections.
- **User Type ID's.** A list of User Type ID's, set in AgileM Studio, controls which users will be presented with the drop down list. For users whose User Type ID are not allowed to make changes, the grid will present the current value in the rpt table without the drop down list functionality.
- **Log table.** User changes are written to the rpt table, (which varies with the specific implementation), and the app table tblUserEditLog (whose name does not vary). Entries made to the tblUserEditLog are primarily of interest to the dba and the implementer has no responsibility for processing this table.
- **ShowToolTip**. Note that the attribute ShowToolTip and Type='edit-dropdown' are mutually exclusive. A Column may have one or the other, but not both. This is because the single click event activates both.

Configuration

Configuration of a drop down list is set in both the gridpanelcollapse Component and the individual Columns.

At the gridpanelcollapse Component level, set the attributes:

- TableName: The name of the rpt database table name which, when the user makes a drop down list selection, the change will be written to.
- UniqueColumnName. Set to a column name in the table whose data uniquely identifies the row. This could, but is not required, to be a Primary Key column or an Identity column. Note also that this column must exist somewhere in the table either as a visible column or as a column in a column group whose Hidden attribute is set to true. Also note that this unique column may not itself be a column that supports a drop down list.

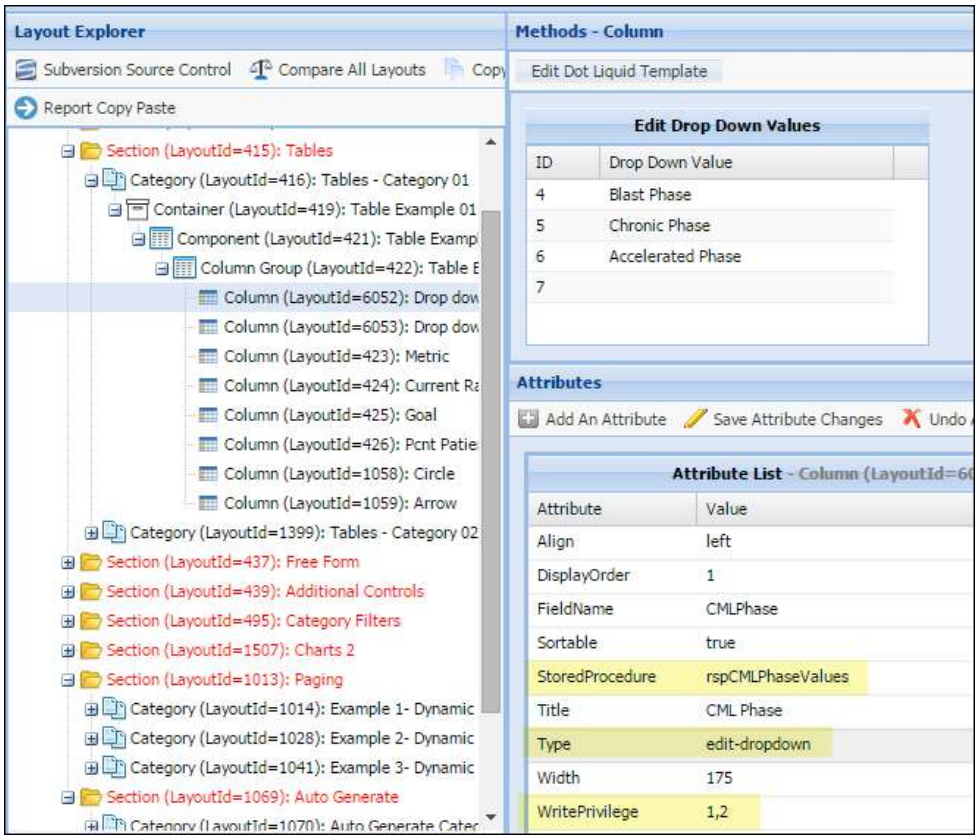


At the Column level, set the attributes:

- StoredProcedure: The name of a Stored Procedure that will be executed by RPT_rspBuildListOfValues to populate RPT_tblListOfValues. This ListOfValues table will contain the list of values presented for drop down columns.

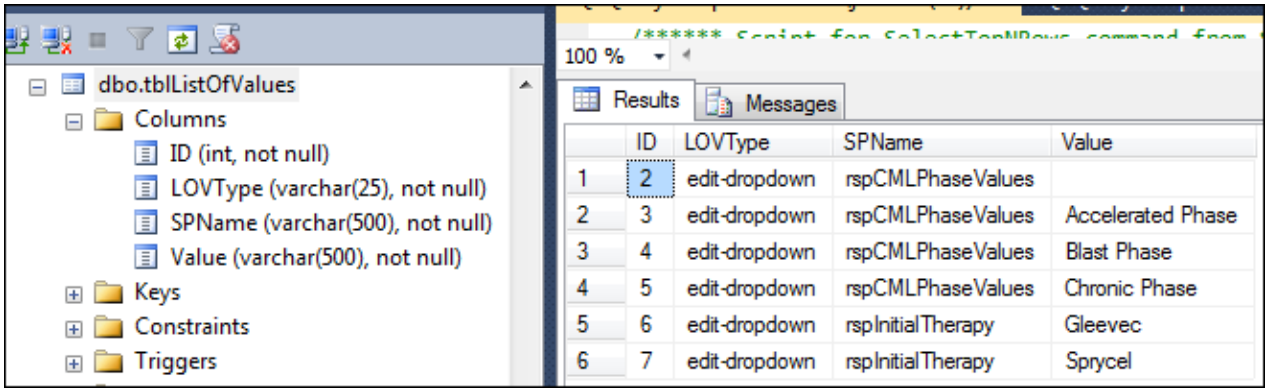
Note this architecture is identical to the way Filter are populated – that is, a stored procedure name is supplied, a master stored procedure is executed (in this case, RPT_rspBuildListOfValues), and a master table (in this case, RPT_tblListOfValues), drives the actual drop down list.

- Type: Set to 'edit-dropdown'.
- WritePrivilege; A comma separated list of UserType IDs that are allowed to edit this Column. If all users are to see the drop down list, then include all User Type ID's.



Additional notes:

- tblListOfValues. Following is the definition of tblListOfValues along with example rows. For drop down lists, the LOVType column must be 'edit-dropdown'.



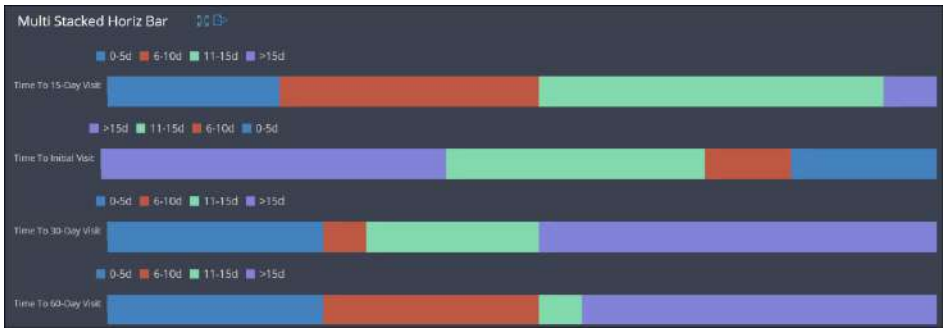
	ID	LOVType	SPName	Value
1	2	edit-dropdown	rspCMLPhaseValues	
2	3	edit-dropdown	rspCMLPhaseValues	Accelerated Phase
3	4	edit-dropdown	rspCMLPhaseValues	Blast Phase
4	5	edit-dropdown	rspCMLPhaseValues	Chronic Phase
5	6	edit-dropdown	rspInitialTherapy	Gleevec
6	7	edit-dropdown	rspInitialTherapy	Sprycel

- Matching tblListOfValues with the actual data. Note it is critical that the list of values include every value in the reporting tables. Say for example tlbListOfValues has only the entries: Product1, Product2 and Product3, but the data has Product4. When the drop down displays for a row that has Product4, the user will instead see Product1 even though the data has Product4. Bottom line, assure that tblListOfValues has every entry in the actual data.

Multi Horizontal Stacked Bar Chart

See a fully functioning example here:

APDeveloper Report 01:
Section: Charts3 (Flex)
Container: Multi Horiz Stacked Bar Chart
Stored Proc: rspChartsCategory06



This Chart is supported by the following record set:

	layoutID	type	name	category	categoryorder	y	format	metric1Format	band1Color
1	7930	bar	0-5d	Time to 15-day visit	1	0.2000	percent	percent	#4381bf
2	7930	bar	6-10d	Time to 15-day visit	1	0.3000	percent	percent	#bf5843
3	7930	bar	11-15d	Time to 15-day visit	1	0.4000	percent	percent	#82d9ad
4	7930	bar	>15d	Time to 15-day visit	1	0.1000	percent	percent	#8282d9
5	7930	bar	>15d	Time to initial visit	2	0.4000	percent	percent	#8282d9
6	7930	bar	11-15d	Time to initial visit	2	0.3000	percent	percent	#82d9ad
7	7930	bar	6-10d	Time to initial visit	2	0.1000	percent	percent	#bf5843

Items to note about the Multi Horizontal Stacked Bar Chart:

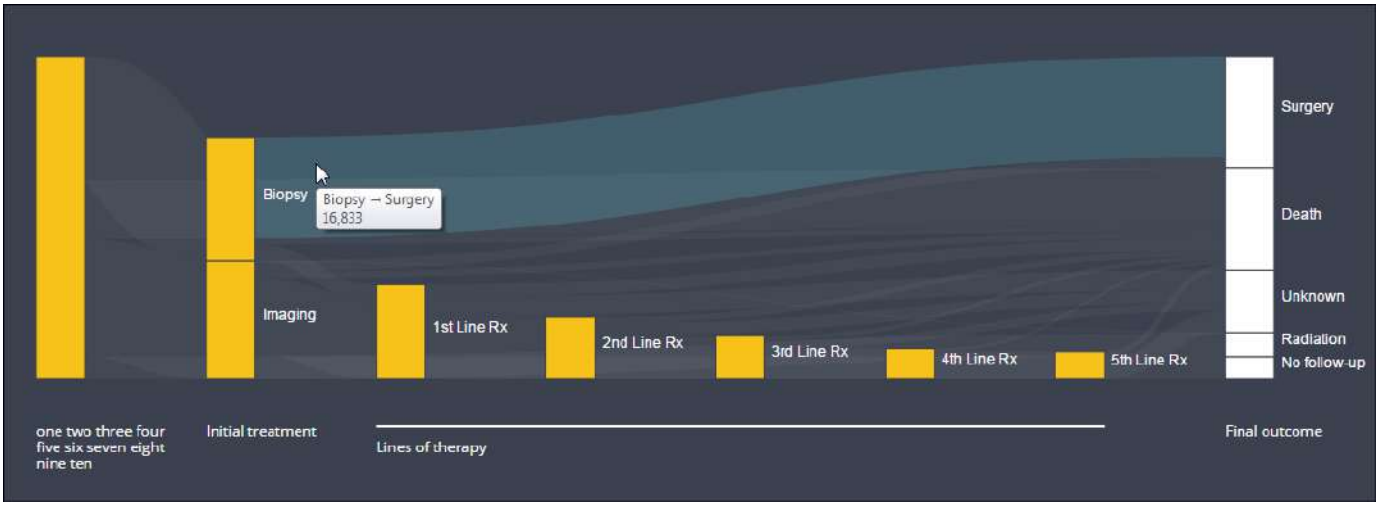
- In AgileM Studio, set the Component Type attribute to ‘multihorizstackbar’.
- The order of each category (Age Group, Gender, etc.), is controlled by the order of the rows in the Components record set.
- The order of each series within a category is also controlled by the order of the data in the Components record set.
- The color for a series will default to the system colors or may be set with the column band1Color.
- If a series has a y value of zero, that series will display in the legend but will not show as a stacked bar.
- The number of bars can vary within the same Container. The system will automatically spread the bars equally across the Container height
- The above example shows each category with different series and thus different legends - in this case the type attribute should be set to ‘multihorizstackbarD3’. If each category has the same series and thus the need for only one legend, then set the type attribute to ‘bar’ and the stacked attribute to ‘true’.

Process Flow Diagram

See a fully functioning example here:

APDeveloper Report 01:

Section: Charts3 (Flex)
Container: Process Flow
Stored Proc: rspChartsCategory06



In AgileM Studio, set the Component Type attribute to 'processflow'. This Chart is supported by two record sets – Nodes and Links:

Nodes

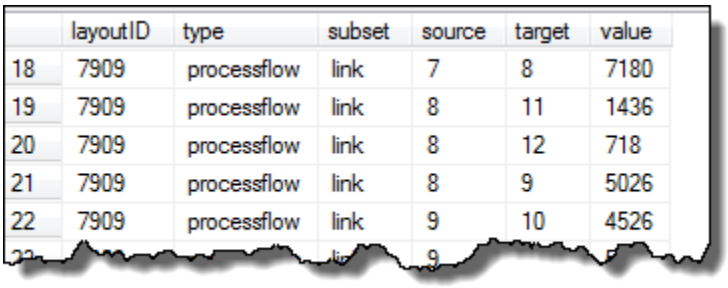
	layoutID	type	subset	name	colorType	stackLabel
6	7909	processflow	node	Surgery	neutral	Final outcome
7	7909	processflow	node	1st Line Rx	focus	Lines of therapy
8	7909	processflow	node	2nd Line Rx	focus	Lines of therapy
9	7909	processflow	node	3rd Line Rx	focus	Lines of therapy

The Nodes dataset describes the boxes displayed in the chart. The required columns are:

- layoutID: this is the layoutID of the container in AgileM Studio.
- type: this must be set to 'processflow'
- subset: this must be set to 'node'
- name: this is the label that appears to the right of the box (can be null)
- colorType: this defines which color the box is. Setting it to 'neutral' will make the box white, setting it to 'focus' will make the box yellow.

- stackLabel: this defines the column label that appears below the column stack the box is in. If you set this to be the same for boxes that appear in different columns, it will create a label group automatically; as seen in the screen shot for 'Lines of therapy'.

Links



	layoutID	type	subset	source	target	value
18	7909	processflow	link	7	8	7180
19	7909	processflow	link	8	11	1436
20	7909	processflow	link	8	12	718
21	7909	processflow	link	8	9	5026
22	7909	processflow	link	9	10	4526

The links dataset describes the wavy lines that go from box to box. The columns are:

- layoutID: this is the layoutID of the container in AgileM studio. Must be identical to the layoutID set in the NODES dataset
- type: this must be set to 'processflow'
- subset: this must be set to 'link'
- source: this is the index (zero based counting) of the NODE this link begins at
- target: this is the index of the NODE this link ends at
- value: the number of units moving from the source NODE to the target NODE; this sets the thickness of the line.

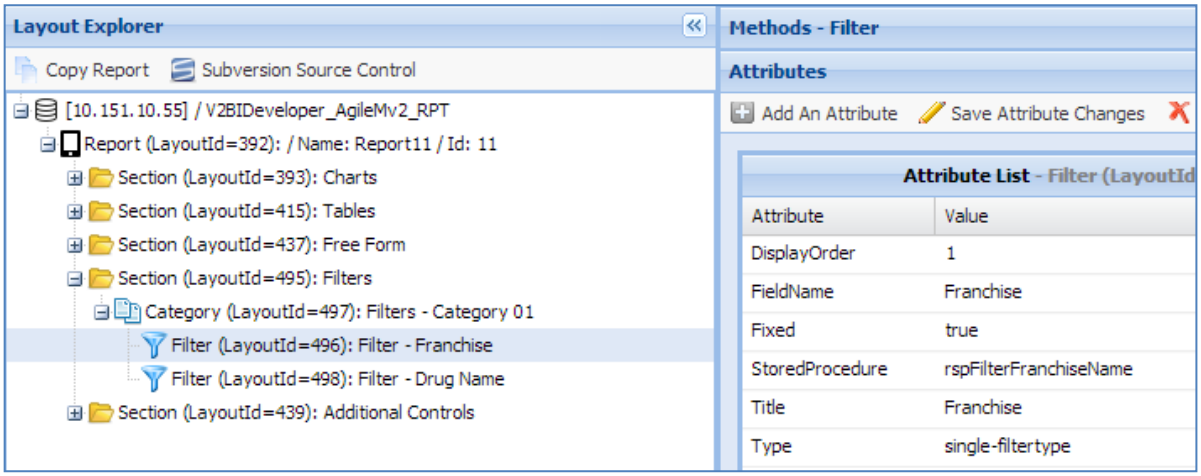
Filter

APDeveloper Report 01:

Section: Filters
Container: Selected Filter Values
Stored Proc: rspFilterCategory01

Overview

Following is a typical set of Attributes for a Filter.

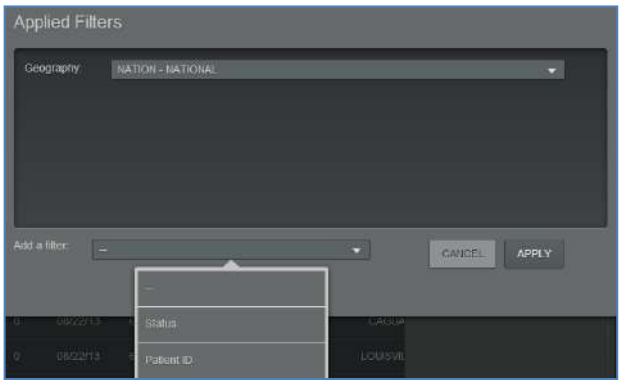


Selected Attribute comments:

- **FieldName**. Maps to the value of FilterName in the stored procedure XML string. Access the Filters value with code like the following;

```
-- Get Filter values
    DECLARE @vchFranchise VARCHAR(10) = (SELECT dbo.GetFilterValue(@tblFilterNameValue, 'Franchise'))
```

- **Fixed**. Fixed = true indicates the Filter will exist in the upper half of the Applied Filters dialog. Fixed = false indicates the Filter will be listed in the Add a Filter list of Filters.



- **Stored Procedure**. The name of a stored procedure that, when executed, will populate rows in tblFilterValue. (See below for more details).

All Filters derive their data from the table tblFilterValue.

tblFilterValue

	FilterValueID	SPName	Name	Value	ParentValue
1	1	rspFilterDrugName	Adravil	Adravil	Oncology
2	2	rspFilterDrugName	Biofoam	Biofoam	Oncology
3	3	rspFilterDrugName	Catana	Catana	Oncology
4	4	rspFilterDrugName	Doloxan	Doloxan	Immunology
5	5	rspFilterDrugName	Hibernol	Hibernol	Immunology
6	6	rspFilterDrugName	Jamitol	Jamitol	Immunology
7	7	rspFilterDrugName	Novril	Novril	Immunology
8	8	rspFilterDrugName	Zoronidril	Zoronidril	Respiratory
9	9	rspFilterFranchiseName	Oncology	Oncology	
10	10	rspFilterFranchiseName	Immunology	Immunology	

Notes regarding tblFilterValue:

- The stored procedure in the Stored Procedure Attribute will actually write the rows to tblFilterValue. That is, associated with each Filter is a Stored Procedure Attribute that, when run, will populate tblFilterValue with rows for that specific Filter.
- The stored procedure rspBuildFilterData will run all stored procedures associated with Filters. So, for this reason, do not manually enter values into tblFilterValue. BI's may run this as needed when Filter values change.
- The Name column presents the text that the user will see in the filter. The Value column is that text that will be passed to the Components stored proc. The values can be different, however **it is best practice to set Name and Value to identical text since it makes development and maintenance significantly easier**. Only if there is a compelling reason, (which occasionally there is), should Name and Value be different.

Filter Stored Procedure Code

APDeveloper Report 01:

Section: Filters
Container: Selected Filter Values
Stored Proc: rspFilterCategory01

The BI group has established a standard way of coding for the filters and all stored procs are expected to follow the standard which is detailed below.

Background

All v2 stored procs fall under one of two categories:

- Non-paging. Such stored procs populate components that do not "page". That is, all data for the component is supplied by the stored proc in exactly one record set.

- Paging. Such stored procs populate components that "page". Components that page have too much data to supply in a single record set – for example physician segmentation tables might have hundreds of rows. Supplying all the data in one record set would result in poor performance. Paging stored procs supply a "page" or a set of records with each vertical swipe by the user.

The stored proc input parameters for each type of stored proc are different.

Non-paging stored proc input parameters:

```
(
     @filterTableXML XML
    ,@vchSortString varchar(max) = ''
    ,@vchFilterString varchar(max) = ''
)
```

Paging stored proc input parameters:

```
(
     @filterTableXML XML
    ,@vchSortString varchar(max) = ''
    ,@vchFilterString varchar(max) = ''
    ,@vchRankBy varchar(500) = ''
    ,@intPageNumber int = null
    ,@intPageSize int = null
)
```

Calling a Non-Paging stored proc

When rspFiltercategory01 is called by the v2 application it does so with a command like the following:

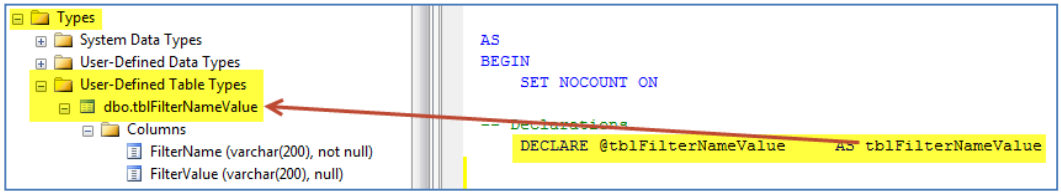
```
EXEC [rspFilterCategory01]
    '<Filters>
        <Filter FilterName="Franchise" FilterValue="Oncology" />
        <Filter FilterName="DrugName" FilterValue="Adravil" />
    </Filters>',
    '','
```

Note that the yellow highlighted section above, (which is an XML string), will then be placed in the input parameter @filterTableXML.

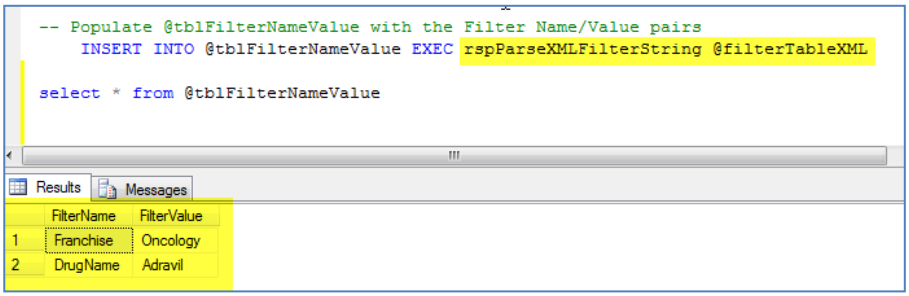
Also note above that there are two <Filter> (singular) tags surrounded by a single <Filters> (plural) tag. Each <Filter> tag is comprised of a FilterName and FilterValue attribute.

Processing @filterTableXML

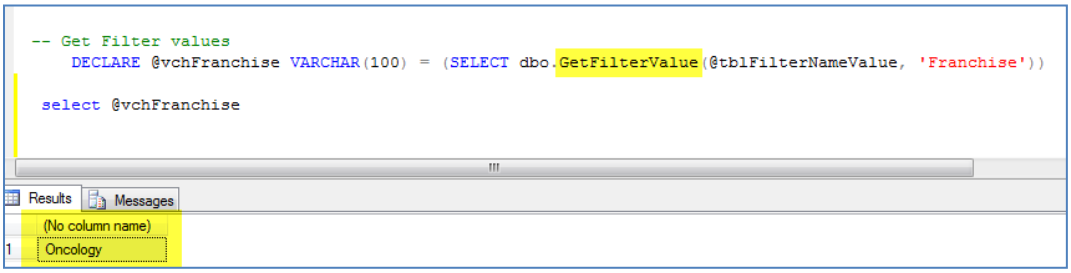
Just as there are SQL native data types (int, varchar, decimal, etc), there are types that are Tables. tlbFilterNameValue is a type.



The stored proc rspParseXMLFilterString will place the XML filter string into a table named @tblFilterNameValue. The select statement demonstrates the table contents.



Finally the function GetFilterValue will retrieve individual filter values.



Fixed vs. Non-Fixed Filters

Filters have an Attribute named Fixed which can take the value true or false. There are two primary differences between fixed and non-fixed filters – one, where they appear for the user and two, how their values are passed to the controlling stored procedure.

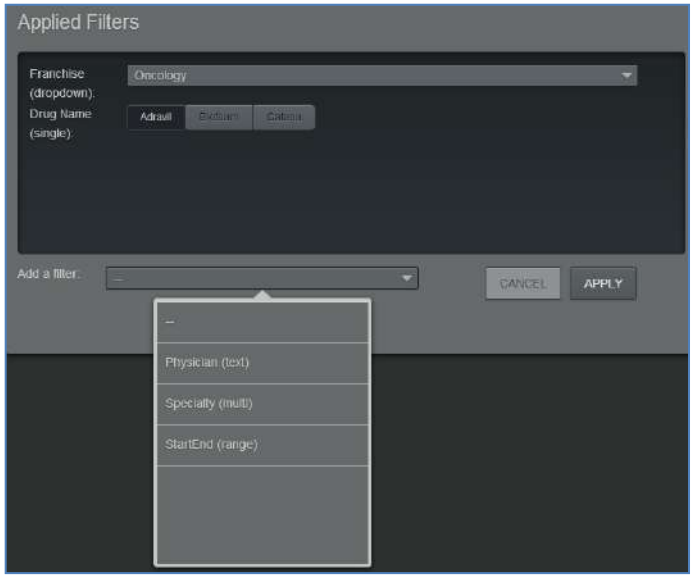
This discussion refers to the following example:

APDeveloper Report 01:

Section:	Filters
Container:	Selected Filter Values
Stored Proc:	rspFilterCategory01

Where Filters Appear. Note that fixed Filters will always appear in the Applied Filters dialog box. Non-fixed filters appear in the Add a filter drop down. The Franchise and Drug Name Filter are "fixed" in the Applied Filters dialog. Physician, Specialty and StartEnd Filters are "non-fixed" and thus appear in the Add a filter drop down where the user can optional select them.

Note that when a filter is "non-fixed" that the attribute Fixed should be removed. That is, do NOT set Fixed=false, but just remove the attribute entirely.



How Filter values are passed. Filter values are assigned to both the @filterTableXML and @vchFilterString stored procedure input parameters.

Fixed Filter values are assigned to @filterTableXML in XML format. In the example, Franchise and Drug Name will look like the following in @filterTableXML:

```
<Filters>
   <Filter FilterName="Franchise" FilterValue="Oncology" />
   <Filter FilterName="DrugName" FilterValue="Adravil" />
</Filters>
```

Non-Fixed Filter values are assigned to @vchFilterString in the form of a Where clause. In the example, Physician, Specialty and StartEnd will look like the following in @vchFilterString:

```
WHERE Physician LIKE('%Dr. Jones%') AND Specialty
IN('OBG','OTHER','UROLOGY') AND StartEnd Between 10 And 20
```

As an aside, note that if all the Filters in this case were non-fixed the @vchFilterString would look like the following:

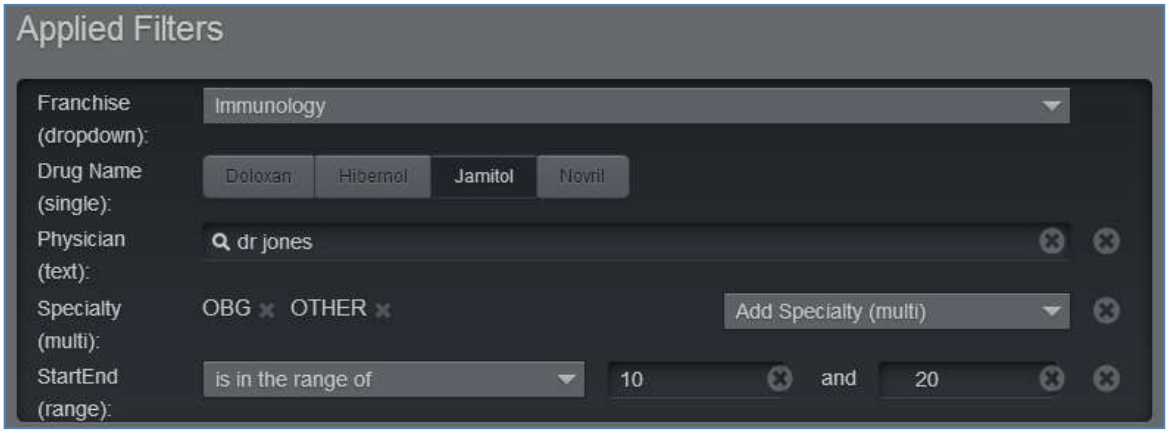
```
WHERE Franchise = 'Oncology' AND DrugName = 'Adravil' AND Physician
LIKE('%Dr. Jones%') AND Specialty IN('OBG','OTHER','UROLOGY') AND
StartEnd Between 10 And 20
```

The section Stored Procedure Standards discusses the next steps in actually using the Filter selections.

Filter Types

Mobile v2 supports several Filter types and the Filter section of APDeveloper Report 01 demonstrates one example of each.

The table below makes brief comments for each Filter type.



Filter Type	Comments
dropdown-filtertype	Typical drop down control whose list of possible values comes from tblFilterValue.
single-filtertype	List of values, from tblFilterValue, presented as shown above.
text-filtertype	Free text string entered by the user. Does not use a stored proc Attribute or get values from tblFilterValue. Also see the special considerations here when using text filters.
multi-filtertype	List of values, from tblFilterValue, where the user may select zero to all values.
range-filtertype	Supports the entry of a numeric range, greater than, or a less than value. Does not use a stored proc Attribute or get values from tblFilterValue.

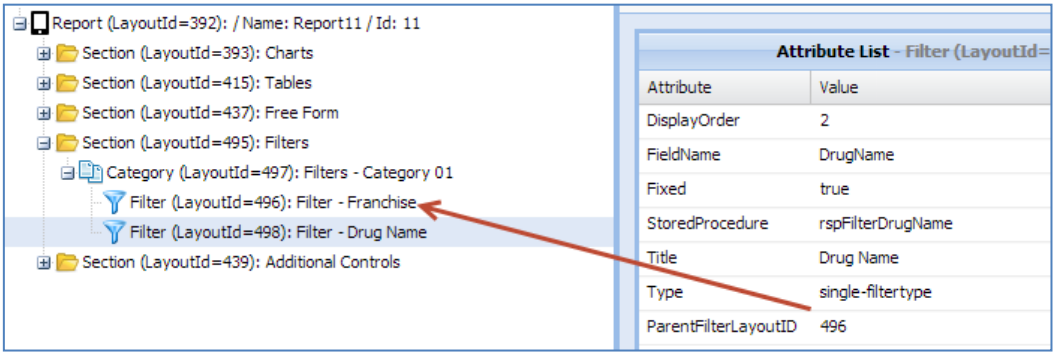
Cascading Filters

APDeveloper Report 01:

Section: Filters
Container: Selected Filter Values
Stored Proc: rspFilterCategory01

Mobile v2 supports linking of Filters, through the ParentFilterLayoutID Attribute, that allows a selection in one Filter to affect the options in a second filter. For example, if Filter1 is Franchise and Filter2 is Drug Name, and each Franchise sells different Drugs, then v2 supports the Drug Name selections changing based on the selected Franchise.

In this case, the ParentFilterLayoutID of the Drug Name Filter points back to the Franchise.



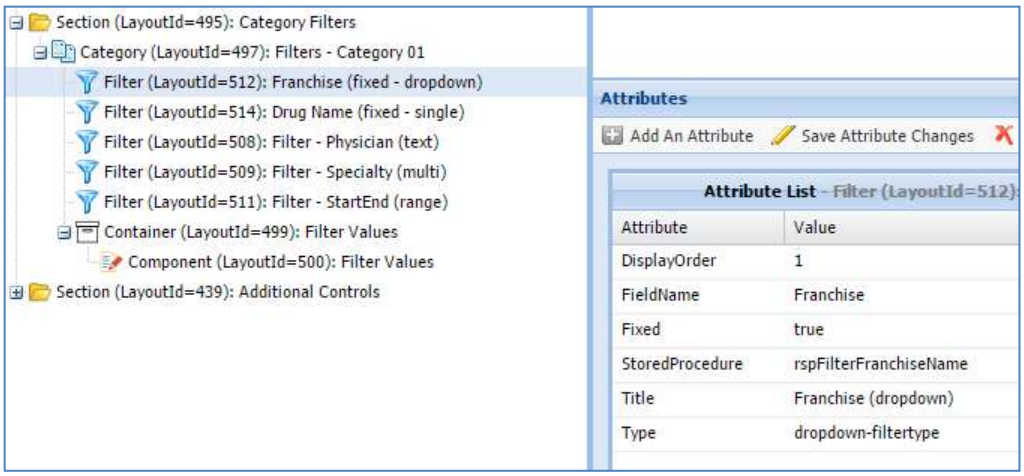
Also, the ParentValue column of tblFilterValue must be populated to link the Franchise to the Drug Name filter as noted below.

	FilterValueID	SPName	Name	Value	ParentValue
1	1	rspFilterDrugName	Adravil	Adravil	Oncology
2	2	rspFilterDrugName	Biofoam	Biofoam	Oncology
3	3	rspFilterDrugName	Catana	Catana	Oncology
4	4	rspFilterDrugName	Doloxan	Doloxan	Immunology
5	5	rspFilterDrugName	Hibernol	Hibernol	Immunology
6	6	rspFilterDrugName	Jamitol	Jamitol	Immunology
7	7	rspFilterDrugName	Novril	Novril	Immunology
8	8	rspFilterDrugName	Zoronidril	Zoronidril	Respiratory
9	9	rspFilterFranchiseName	Oncology	Oncology	
10	10	rspFilterFranchiseName	Immunology	Immunology	

Filter Creation Check List

Following is check list for Filter creation.

1) **Create the Filter in AgileM Studio**. For example see the following attribute selections and review the notes on selected filter attributes here. Determine the type of filter here.



2) **Determine the filter values**. Working with the BA, determine the values that will be presented to the user. For example, in this case, determine that the filter values will be: Oncology, Immunology and Respiratory.

Expect the list of value might be one of the following:

- A short hard coded list that will likely never vary, for example quarter: Qtr1, Qtr2, Qtr3, Qtr4. (In this case the filters stored proc might hard code these values).
- A short list that has potential to vary, for example a product list. Such a list may never be long but is likely to change over time. (In this case a tblRptProduct table might hold this list).
- A long list with regular variability. (In this case a distinct list of values might come from one of the tblRpt tables).

3) **Write the filters stored proc**. Create a stored procedure whose only purpose is to insert rows into tblFilterValue. The name of this stored procedure will used in the Filters StoredProcedure attribute. For details on tblFilterValue, click here. Also note that this stored procedure must first delete all of its rows in tblFilterValue.

```
AS
BEGIN

-- first, remove all current listings for rspFilterFranchiseName
    DELETE FROM tblFilterValue WHERE SPName = 'rspFilterFranchiseName'

-- add all Franchises from tblFranchise
    INSERT INTO tblFilterValue(SPName, Name, Value, ParentValue)
        (SELECT 'rspFilterFranchiseName',
                 FranchiseName
                ,FranchiseName
                ,''
         FROM tblFranchise)

END
```

```
FROM [V2BIDeveloper_AgileMv2_RPT].[dbo].[tblFilterValue]
```

100 %

Results | Messages

	FilterValueID	SPName	Name	Value	ParentValue
11	11	rspFilterFranchiseName	Oncology	Oncology	
12	12	rspFilterFranchiseName	Immunology	Immunology	
13	13	rspFilterFranchiseName	Respiratory	Respiratory	

4) **Execute rspBuildFilterData**. Execute the stored procedure rspBuildFilterData. The job of this stored proc is to execute the StoredProcedure associated with all Filters. Do not manually enter rows in tblFilterValue.

5) **Write the Category stored proc**. Write the Category stored proc which will now have access to the values a user has selected. Use statements like the following to access the filter value the user selected. Note that the value 'Franchise' comes from the Filters FieldName attribute.

```
-- Get Filter values
    DECLARE @vchFranchise VARCHAR(10) = (SELECT dbo.GetFilterValue(@tblFilterNameValue, 'Franchise'))
```

Rsp[X]ByUserEmail

Overview & Explanation:
By User Email is a type of stored procedure used within filters to determine what a user has permission to see during their session.
(Note: For this Filter to work the SP name must contain "ByUserEmail")

Its purpose, in this example, is to:

- Allow "Home Office" users to view all data
- Allow "Country Managers" to view only their country's / territories' data
- Allow "Field" users to view only their territory.

Below is an example of a "By User Email" stored procedure. It creates "By User Email" geography filters for the "Commercial Dashboard" report.

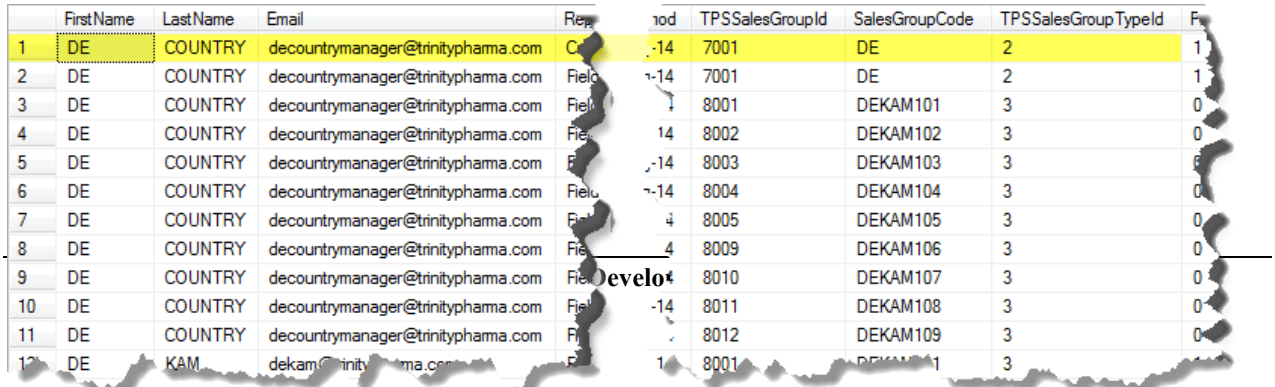
```
(
    @vchUserEmail VARCHAR(255)
   ,@intReportID INT
)
```

```
SELECT
    'rspCommercialDashboard_FilterGeographyByUserEmail'
    ,SalesGroupCode
    ,SalesGroupCode
    ,ParentValue = ''
FROM tblSalesGroupWeekly a
WHERE SalesGroupTypeID IN (1, 2)
    AND FieldForce = 'SALES'
    AND [SalesGroupID] IN (
            SELECT  [TPSSalesGroupId]
              FROM  [tblCfgRoster]
              WHERE Email = @vchUserEmail AND ReportName = 'Commercial Dashboard' )
ORDER BY SalesGroupTypeID ASC, a.SalesGroupCode ASC;
```

The stored procedure uses the **"@vchUserEmail**" (It is necessary to begin with "(vchUserEmail VARCHAR(255) ,@intReportID INT)"). The V2 application finds the email associated with a username in tblUser at log-in. The script above uses that email and tblCfgRoster to create the set of TPSSalesGroupIDs that the user will be able to access. This is then converted into SalesGroupCodes

This stored procedure does not make entries into tblFilterValue.
It is also necessary to make the FieldName: SalesGroupCode.

Below you can see that one Email can match to several TPSSalesGroupIDs. This is how permission to see multiple territories is given.

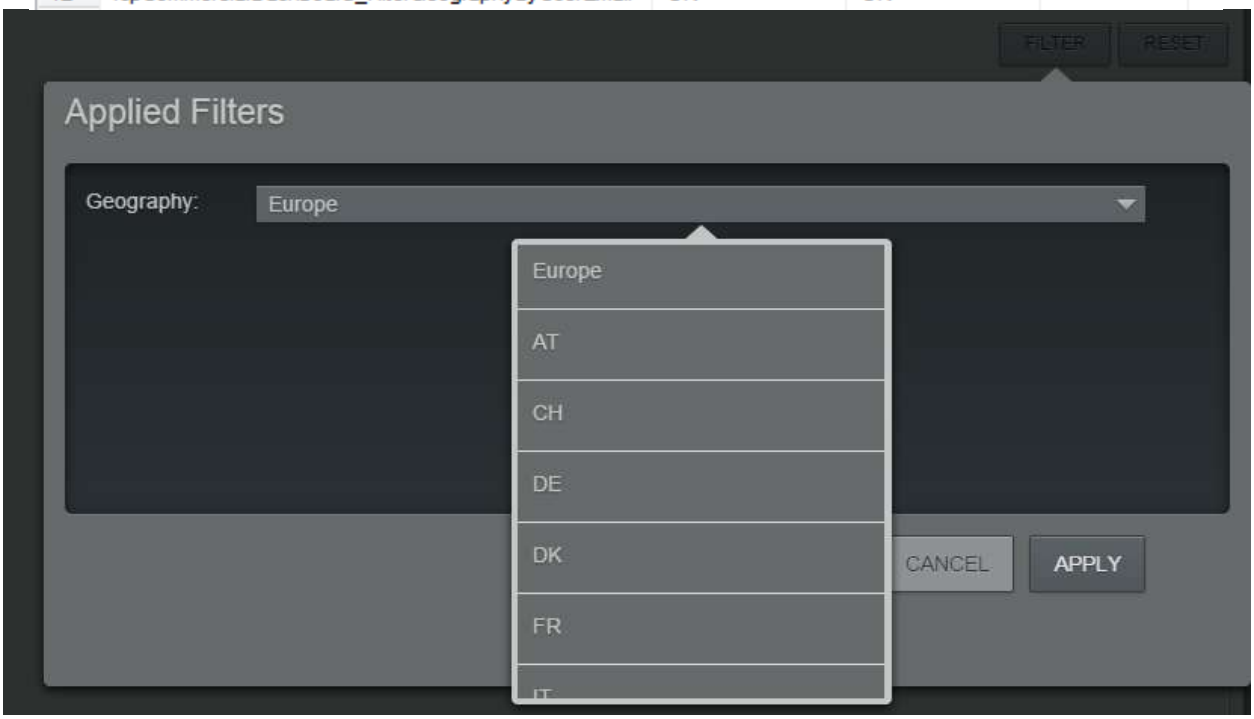


	FirstName	LastName	Email	Re…	…od	TPSSalesGroupId	SalesGroupCode	TPSSalesGroupTypeId	F…
1	DE	COUNTRY	decountrymanager@trinitypharma.com	C…	…-14	7001	DE	2	1
2	DE	COUNTRY	decountrymanager@trinitypharma.com	Field…	…-14	7001	DE	2	1
3	DE	COUNTRY	decountrymanager@trinitypharma.com	Field…	…	8001	DEKAM101	3	0
4	DE	COUNTRY	decountrymanager@trinitypharma.com	Fie…	14	8002	DEKAM102	3	0
5	DE	COUNTRY	decountrymanager@trinitypharma.com	F…	…-14	8003	DEKAM103	3	0
6	DE	COUNTRY	decountrymanager@trinitypharma.com	Field…	…-14	8004	DEKAM104	3	0
7	DE	COUNTRY	decountrymanager@trinitypharma.com	F…	…	8005	DEKAM105	3	0
8	DE	COUNTRY	decountrymanager@trinitypharma.com	Fie…	…4	8009	DEKAM106	3	0
9	DE	COUNTRY	decountrymanager@trinitypharma.com	Fie…	Develo…	8010	DEKAM107	3	0
10	DE	COUNTRY	decountrymanager@trinitypharma.com	Fie…	…-14	8011	DEKAM108	3	0
11	DE	COUNTRY	decountrymanager@trinitypharma.com	Fi…	…	8012	DEKAM109	3	0
12	DE	KAM	dekam@trinity…ma.com	F…	1…	8001	DEKAM…1	3	…

Results for "Home Office":
This stored procedure returns this result set for "Home Office" users. Note they can see all the countries.

Results | Messages

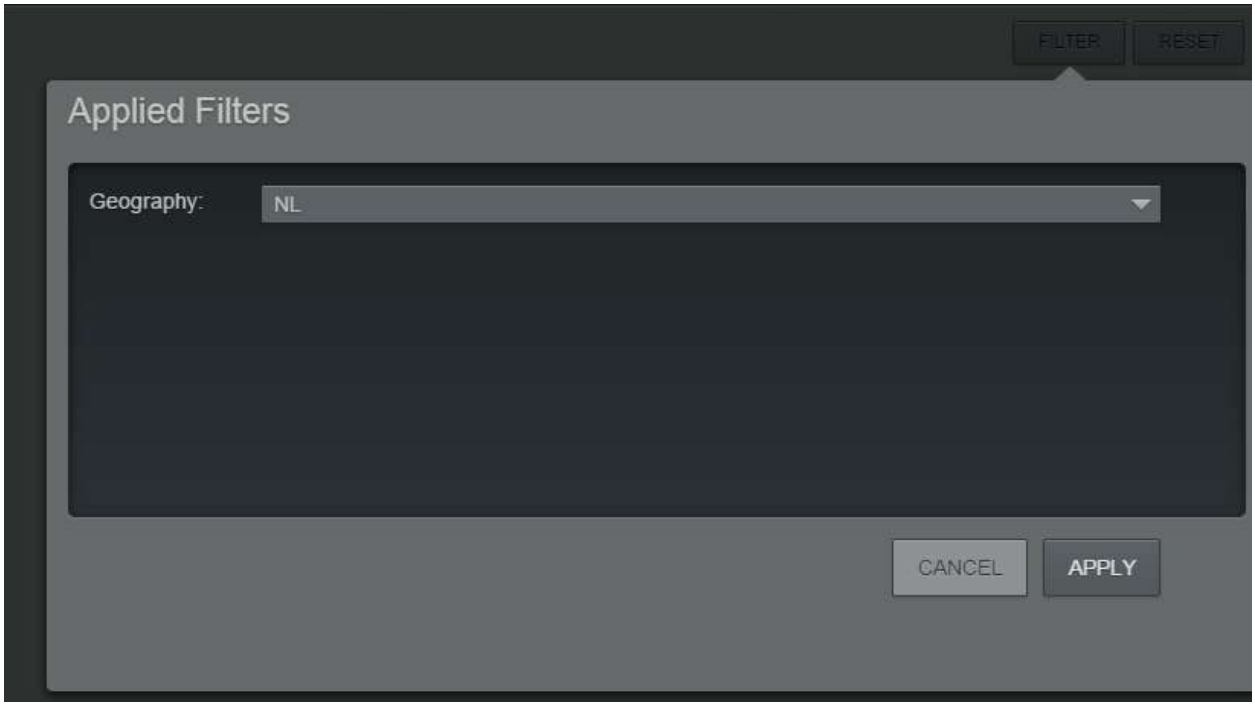
	(No column name)	SalesGroupCode	SalesGroupCode	ParentValue
1	rspCommercialDashboard_FilterGeographyByUserEmail	Europe	Europe	
2	rspCommercialDashboard_FilterGeographyByUserEmail	AT	AT	
3	rspCommercialDashboard_FilterGeographyByUserEmail	CH	CH	
4	rspCommercialDashboard_FilterGeographyByUserEmail	DE	DE	
5	rspCommercialDashboard_FilterGeographyByUserEmail	DK	DK	
6	rspCommercialDashboard_FilterGeographyByUserEmail	FR	FR	
7	rspCommercialDashboard_FilterGeographyByUserEmail	IT	IT	
8	rspCommercialDashboard_FilterGeographyByUserEmail	NL	NL	
9	rspCommercialDashboard_FilterGeographyByUserEmail	NO	NO	
10	rspCommercialDashboard_FilterGeographyByUserEmail	OT	OT	
11	rspCommercialDashboard_FilterGeographyByUserEmail	SE	SE	
12	rspCommercialDashboard_FilterGeographyByUserEmail	UK	UK	



Results for "Country Managers":
This is the result set for the "Country Managers." Note they can only see their country.

Results | Messages

	(No column name)	SalesGroupCode	SalesGroupCode	ParentValue
1	rspCommercialDashboard_FilterGeographyByUserEmail	NL	NL	



Note:
"Field" users (SalesGroupTypeID = 3) would not be able to see anything in this example because they would fail the first WHERE clause condition.

Interactive Example:
http://10.151.10.187/APDeveloper/
Here you can log in as any of these users:

User Name	Password	Associated Email	User Type
A	A0000000	ametro@trinitypharma.com	Internal
Home.Office	A0000000	UserType2@trinitypharama.com	Home Office
Field	A0000000	UserType3@trinitypharama.com	Field

The example is located in Report 1 in the 'Category Filters' section as the second category. The associated Stored Procedures are:

1. rspFilter_GeographyByUserEmail
2. rspFilterCategory02


By User Email
Viewable By:
ametro@trinitypharma.com
Nation
New York
Gotham
By User Email
Viewable By:
ametro@trinitypharma.com, UserType2@trinitypharama.com
New York
Gotham
By User Email
Viewable By:
ametro@trinitypharma.com, UserType2@trinitypharama.com,
UserType3@trinitypharama.com
Gotham

Responsive Layouts

Applications are said to be "responsive" when the presentation layer, as viewed by the end user, automatically adjusts based on the size of the display device. That is, if an application is "responsive" it will be equally viewable on a desktop, iPad and iPad mini.

When a v2 Report is configured to be responsive Containers will automatically adjust their size based on the display size. A Container will be physically smaller on an iPad mini than on a desk top display, but the Container size will be proportionally identical on all devices. For example, a KPI whose width is 25% of the full screen width on a desktop will still be 25% of the full screen width on an iPad mini (even though the KPI would be physically smaller on an iPad mini).

Up until July 2015 all v2 Reports were non responsive. When configuring reports, Container size was always established with width and height attributes and Container position was always established with x and y attributes. This is often referred to as "fixed positioning".

To make reports responsive "flex positioning" was introduced with the following primary configuration enhancements:

1. Containers can have child Containers.
2. Containers are sized with the Flex attribute.
3. Containers are positioned with the Layout attribute.

Each of these configuration enhancements is discussed below.

Note if wishing to convert a Report to Flex a stored procedure exists that can assist in this effort. Please note the following:

- The stored procedure is located in the BI departmental document repository at: SVN_BITeam. The file name is: Insert Flex Layout with instructions.sql.
- Depending on the complexity of the Report, expect this stored procedure will do a high percentage of the conversion effort. But realize, this is not a fully automated process.
- A full set of usage instructions are located at the top of the stored procedure.

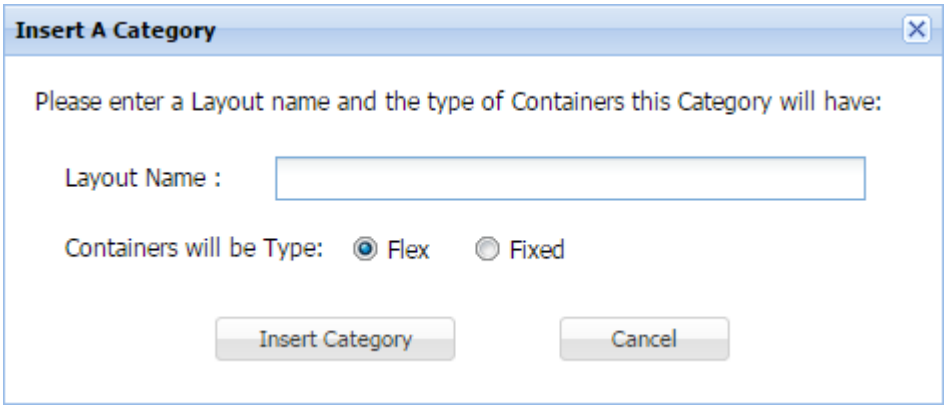
Containers can have child Containers

Containers that use flex positioning can themselves have Containers. That is, Containers can have child Containers. And furthermore, this nesting can be deep. That is, a Container can have a child Container, which can in turn have a child Container, etc. As a practical matter, the nesting will not likely go beyond 3-4 deep. The purpose of allowing Containers within Containers is discussed below.

The following screen shot shows how AgileM Studio supports nested Containers:

When using fixed positioning, Containers may not have child Containers.

Note that when creating a Category, the following prompt will display. At Category creation time AgileM Studio needs to establish if the Containers in the Category will be using flex or fixed positioning.



Note that flex Containers will have their Type attribute automatically set to: group_container.

Fixed Containers will automatically have their Type set to componentcontainer.

Once the Category is established as having flex or fixed Containers then all Containers in that Category will automatically be of the same type. It is not possible to have some Containers in a Category be flex while others are fixed.

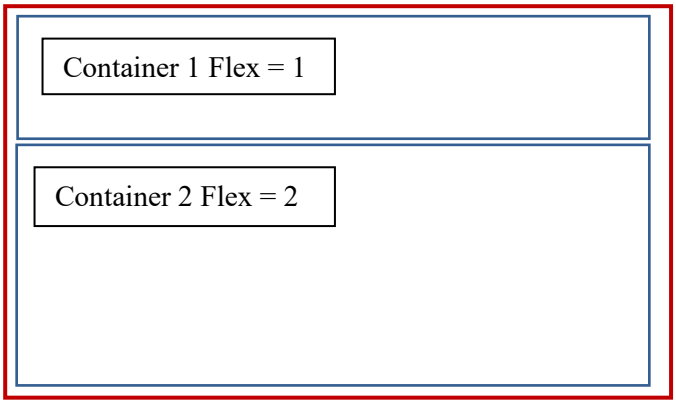
Containers are sized with the Flex attribute

With fixed positioning, Containers use the width and height attributes to indicate their absolute size. But with flex positioning Containers use the Flex attribute to indicate their relative size.

In the example below, consider a Category that has two Containers as follows:

1) Container 1 with a Flex attribute = 1
2) Container 2 with a Flex attribute = 2

In the example below, consider a Category that has two Containers as follows. (As an aside, this Category has Layout = 'vbox' – more on that later).



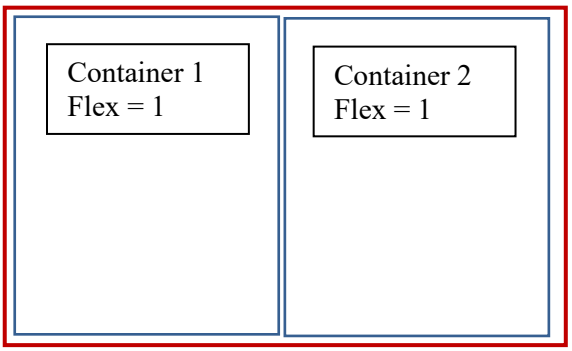
To determine a Containers relative size add up all the Flex values for each child Container of the Category. In this case, Flex =1 + Flex = 2 is a total of 3. Container 1 will take up a ratio of 1/3rd of the total Category space, while Container 2 will take up a ratio of 2/3rd of the total Category space.

The general rule to determine a Containers size is to calculate the following ratio:

$$\frac{\text{The Flex value for the given Container}}{\text{Sum of the Flex values for all sibling Containers}}$$

Note that you don't actually add up the Flex values for ALL Containers in the Category, but the Flex values for all the sibling Containers. This concept will be detailed in upcoming examples.

In the example below, consider a Category that has two Containers, each with a Flex=1. (As an aside, this Category has Layout = 'hbox' – more on that later). With each Container having Flex=1, the Containers will be identical in size.



Containers are positioned with the Layout attribute

Containers and Categories each have a Layout attribute. The Layout attribute controls how Containers are laid out.

For example, consider a Category whose Layout attribute = 'hbox' with three Containers (each with their Flex attribute = 1). The Containers are "laid out" horizontally across the full Category width.

Now consider a more complex example that starts with the example above where a Category has Layout = 'hbox'. And now, give the left most Container a Layout = 'vbox' (remember, the Layout attribute can be applied to both Categories and Containers).

In this example:

1) The red Category has Layout = 'hbox' (thus the blue Containers are laid out horizontally).

2) The Blue Containers all have Flex=1 (so they are all the same size).

15) The left most Blue Container has Layout='vbox' (so it's green Containers are laid out vertically).

16) The Green Containers have Flex=1 and Flex=3. So the smaller of the two green Containers fills 1/4 of its parent Container and the larger of the two green Containers fills 3/4 of its parent Container.

Finally, note that the Layout attribute must not exist on Containers when the children of Containers are Components. That is, Containers have Layout when their children are Containers, but not when their children are Components.

Miscellaneous Notes

While Flex Containers will generally have their size controlled with the Flex attribute, they may optionally be sized with the width and/or height attributes. That is, Containers may have either:

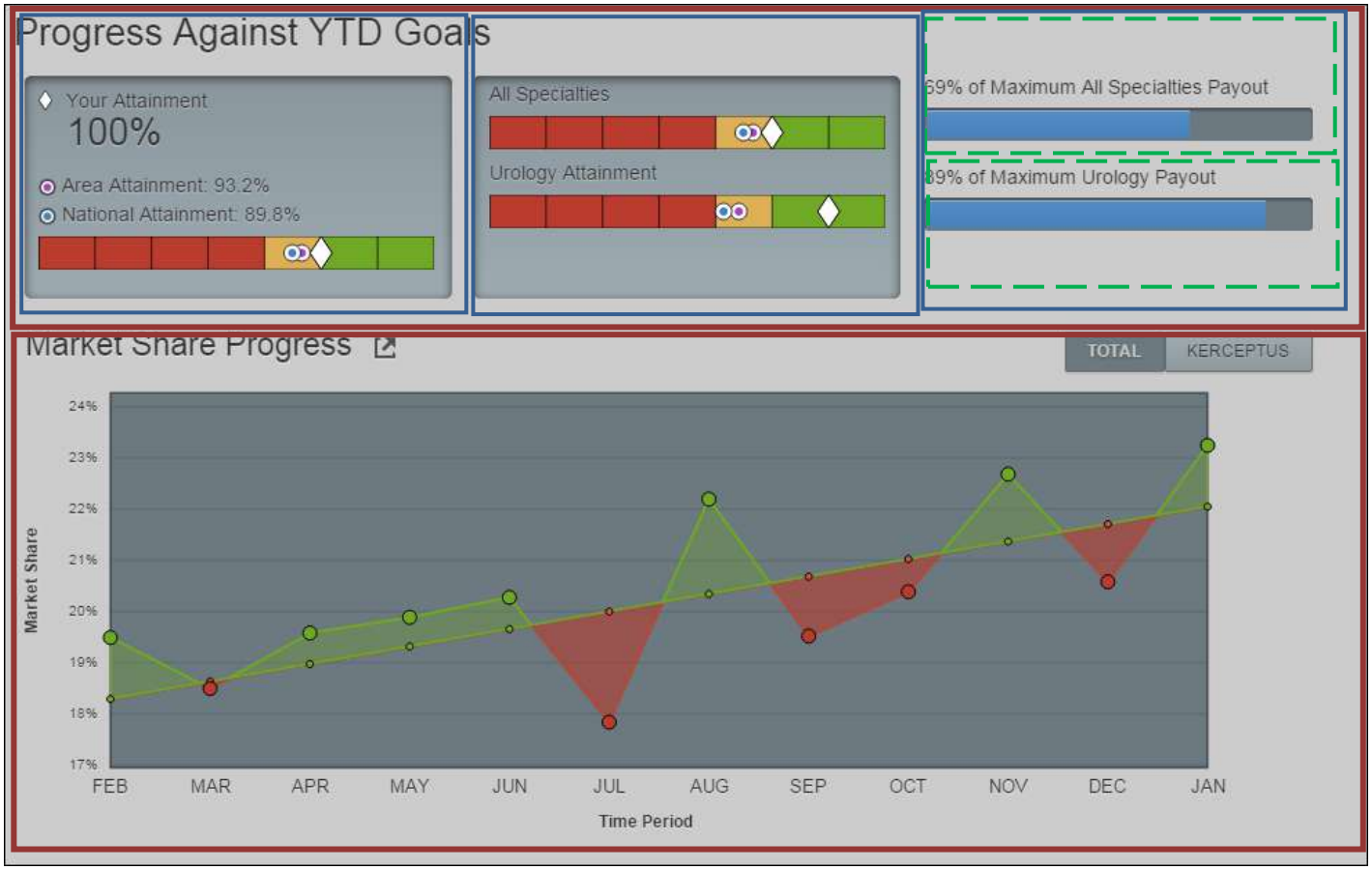
1) the Flex attribute alone populated, or

2) the width and/or height attributes populated.

This might be appropriate, for example, when there is a need to "hard code" the height of the KPI so their containing free form components always display correctly.

Use Case 1

APDeveloper Report 04:
Section: Flex Use Cases
Category: Use Case 1



Category: Layout = vbox
groupContainer: Layout=hbox, Height=150, DisplayOrder=1
groupContainer: Flex=1, DisplayOrder=1
component:
groupContainer: Flex=1, DisplayOrder=2
component:
groupContainer: Flex=1, Layout = vbox, DisplayOrder=3
groupContainer: Flex=1, DisplayOrder=1
component:
groupContainer: Flex=1, DisplayOrder=2
component:
groupContainer: Flex=1, DisplayOrder=2
component:
component:

Use Case 2

APDeveloper Report 04:

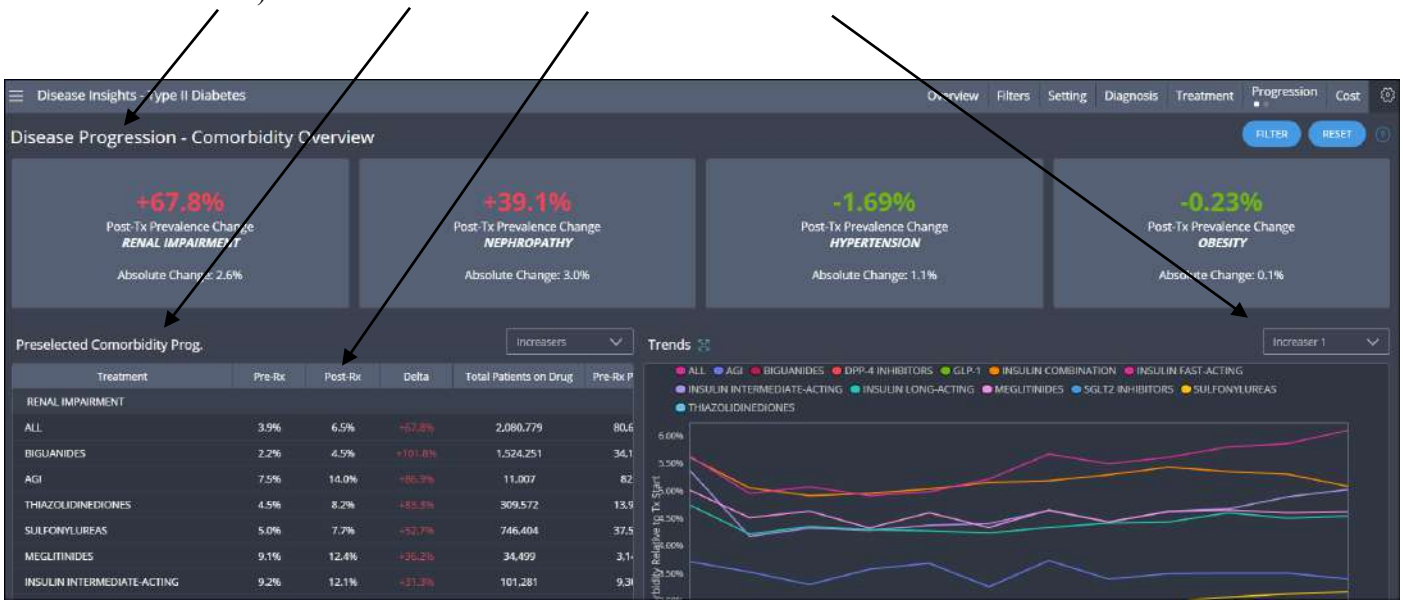
Section: Flex Use Cases
Category: Use Case 2

Category: Layout = vbox
groupContainer: Layout=hbox, Height=150, DisplayOrder=1
groupContainer: Flex=1, DisplayOrder=1
component:
groupContainer: Flex=1, DisplayOrder=2
component:
groupContainer: Flex=1, DisplayOrder=3
component:
groupContainer: Flex=1, DisplayOrder=4
component:
groupContainer: Flex=1, Layout=hbox, DisplayOrder=2
groupContainer: Flex=2, DisplayOrder=1
component:
groupContainer: Flex=5, DisplayOrder=2
component:
component:
component:
component:

Data Driven Titles

Titles exist for Categories, Containers, Grid column headers, x-axis and y-axis. For example:

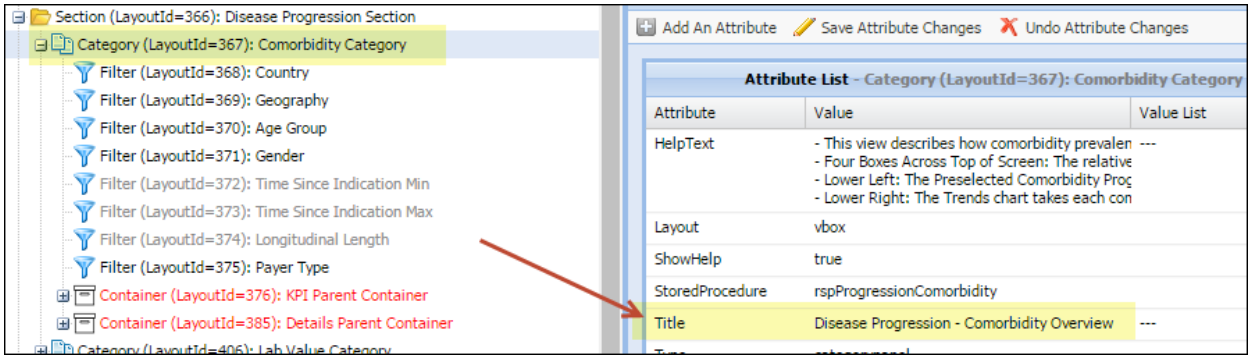
Category Title, Container Title, Column header, tab panel drop down text (x-axis and y-axis not shown).



The actual text to be displayed for all these titles is supplied from one of two possible sources:

1) **Title attribute value**: The layout types Category, Container, Column and Component each support a Title attribute and Plotly Components also support an XAxisTitle and YAxisTitle attribute.

   Here, the attribute Title has a hardcoded value of 'Disease Progression – Comorbidity Overview'. This value will always display as the Category title. Neither the data nor a change in Filter values will affect this title.



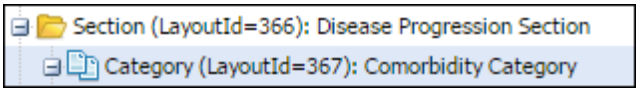
3) **Data driven titles**: Categories, Containers, Columns and Components also support data driven titles. That is, the stored procedures that provide a reports data may optionally contain the text that is to be displayed for a Category, Container, grid headers, tab panel dropdowns, xAxis and yAxis.

   Note the following:
   - Supplying data driven titles is optional.
   - A common use of data driven titles will be to adjust a title based on Filter value(s).
   - If Titles are supplied by the layouts Title attribute and in data, the data title will take precedence.
   - Categories and Containers are not required to have any title and if there is neither an attribute nor a data driven title, more screen real estate will be available for the object. In contrast, Column headers may also be blank but the Column header will still exist even though it is blank.

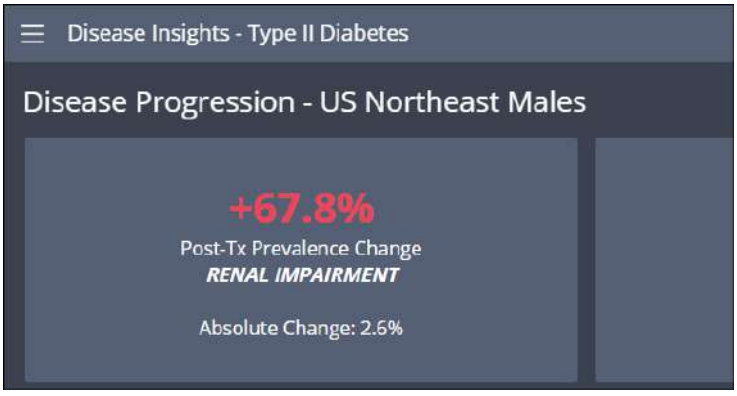
The remainder of this section reviews how to supply data driven titles.

Data driven titles for Categories:
Generally, Categories themselves do not have record sets. However, an exception occurs when supplying a data driven Category title. Using the Category LayoutID, supply a record set as follows with the LayoutID column and a column whose name must be exactly 'categoryTitle'.

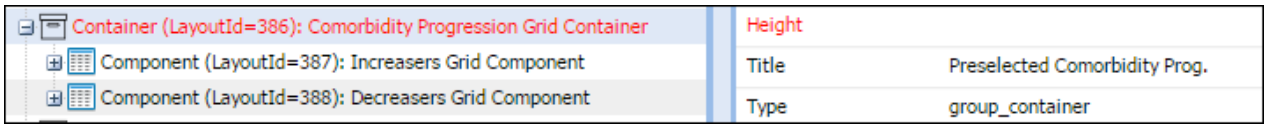
Note that the order of events in the v2 application first paints the Category, (including the Category title), and only then reads in the data. Because of this, if you are using a data driven Category title, it is best to leave the Category attribute Title blank to prevent a brief display of the attribute title before the data driven title.

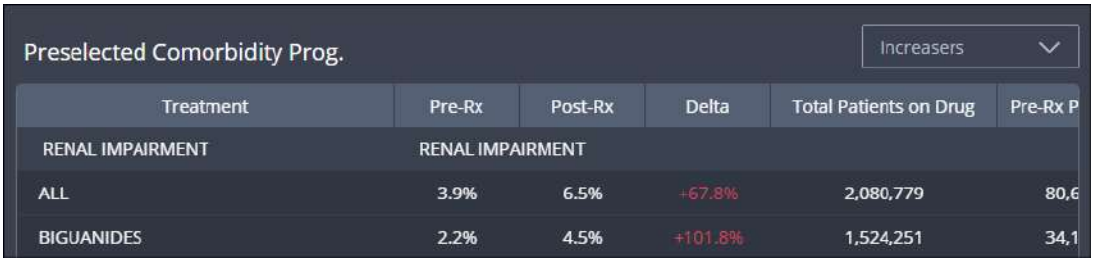



```
SELECT
     layoutID = 367
    ,categoryTitle = 'Disease Progression - US Northeast Males'
```



Data driven titles for Containers:
Note that when supplying Container Titles through a configuration attribute, the Title is supplied at the Container level. For example, the Container with LayoutID 386 has the hard coded Title 'Preselected Comorbidity Prog.'. This single title will display when Component 387 and Component 388 are displayed. This forces a design of using a generic title which is possibly less helpful or descriptive.





Data driven Container titles support a column in the Components record set named: 'containerTitle'. If a Components record set has a containerTitle column, it will be used in place of the Container Title attribute. By placing the containerTitle on the Component record set this allows each Component to have its own data driven Title.

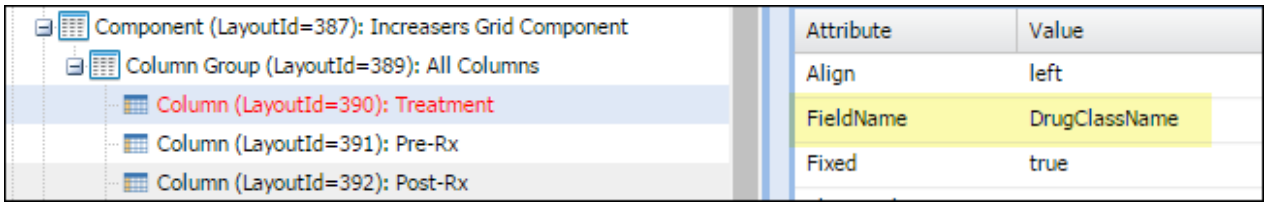
For example, this record set for the Component itself, will adjust the Container Title.

```
SELECT
     layoutID                = 387
    ,containerTitle = 'Increasing Comorbidity Progression'
    ,ComorbidityName         = UPPER(ComorbidityName)
    ,Summ[illegible]ary      = ''
```

When the Decliners option is taken in the drop down, presenting LayoutID=388, yet a different Container Title could be presented.

Data driven titles for Grid headers:
Grid column headers can also be data driven. Note the following example using a pageable gridpanelcollapse.

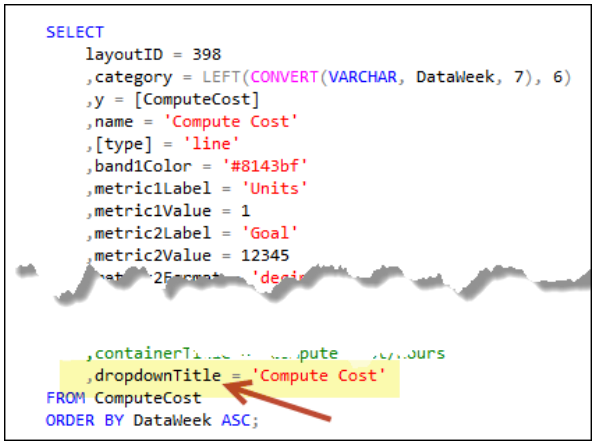
In order to uniquely identify, in the Components record set, which column is to have its title be data driven, use the concatenation of FieldName + 'ColumnTitle'.




```
SELECT
      layoutID                = 387
     ,containerTitle = 'Increasing Comorbidity Progression'
     ,DrugClassNameColumnTitle = 'Treatment - Increasers'
     ,ComorbidityName         = UPPER(ComorbidityName)
     ,DummySecondary          = ''
```

Note that for non-pageable grids, to uniquely identify the column whose title should be data driven, use the concatenation of Title + 'ColumnTitle'.

Data driven titles for tab panel drop down text:
The text that appears in a tab panel drop down may be data driven. For the Components data, include a column named: 'dropdownTitle'. For example:



Data driven titles for xAxis and yAxis:
The text that appears in the x and y axis may be data driven. For the Plotly or D3 charts data, include a column named: 'xAxisDataTitle' or 'yAxisDataTitle'. For example:

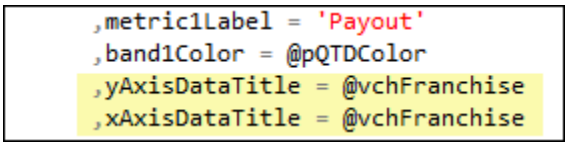


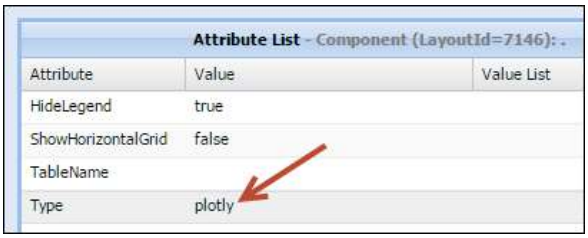
Chart Types – Plotly and d3

There are two primary frameworks used to display Analytic Platform Charts: Plotly and d3. The d3 engine was the primary underlying technology for many of the Chart visualizations prior to v2.8.9. Beginning with v2.8.9 the following Chart visualizations are also available using Plotly technology:

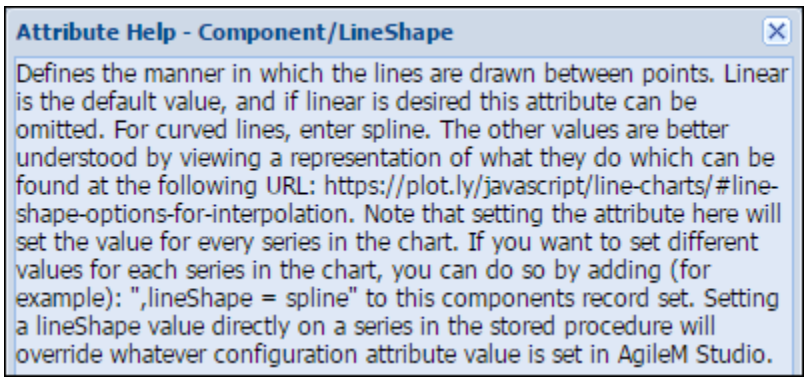
Chart types available using Plotly:

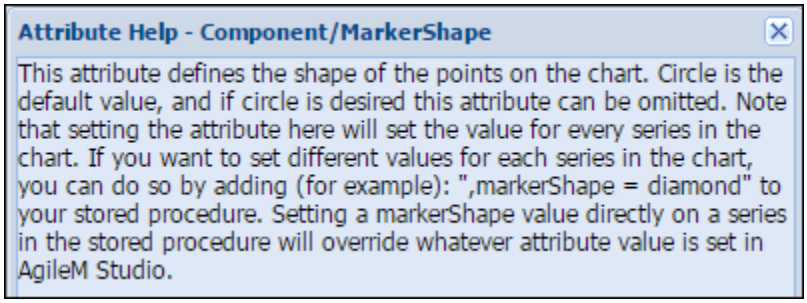
- Pie
- Line
- Bar
- Column
- Stacked Bar
- Area

When configuring a Chart in AgileM Studio the Type attribute will control which technology is used. As a general rule use the Plotly technology when available, since the visualizations are usually better and eventually d3 will be deprecated. Also in some cases there are specific features that are only available in Plotly.



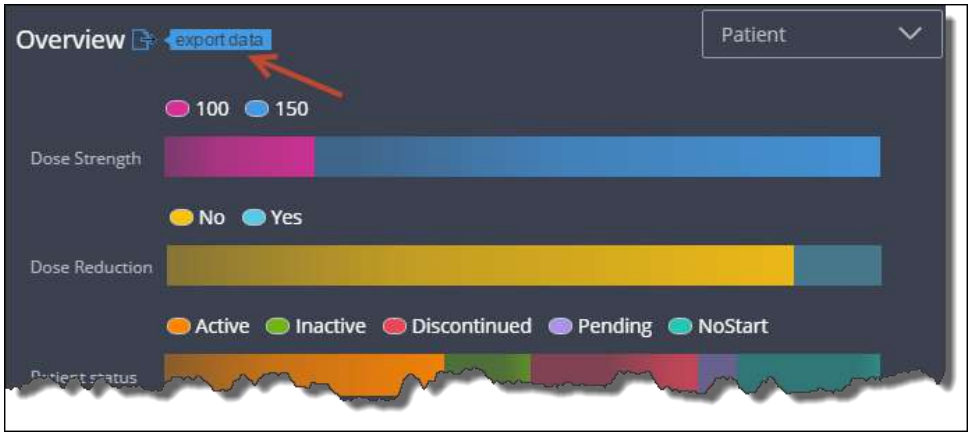
Features only available in Plotly:
Plotly Line Charts support the attributes MarkerShape and LineShape. Following is the AgileM Studio help associated with each attribute.





Excel Export Data

Excel Export Data supports the downloading of Component data to a local Excel file (or in the case of pageable grids, the output may optionally be sent through e-mail). Note the following Component with the "Excel worksheet like" icon to the right of the Component title. When the user mouses-over the icon the "export data" tip displays. Upon clicking the icon, an Excel workbook is downloaded to the users Downloads folder.



The Excel workbook contains two worksheets – the first sheet named "data" and the second sheet named "meta data". Following is the data worksheet. Note the columns contained in the data worksheet will vary by the specific Component whose data was output.

	A	B	C
1	**Category**	**Series**	**Value**
2	Dose Strength	100	21.0%
3	Dose Strength	150	79.0%
4	Dose Reduction	No	87.8%
5	Dose Reduction	Yes	12.2%
6	Patient status	Active	39.0%
7	Patient status	Inactive	12.2%
8	Patient status	Discontinued	23.2%
9	Patient status	Pending	5.5%
10	Patient status	NoStart	20.1%
11	Gender	Male	61.0%
12	Gender	Female	39.0%
13	Patient Age	<25	0.0%
14	Patient Age	25-39	0.2%
15	Patient Age	40-59	8.0%
16	Patient Age	60+	91.7%

data | meta data

Following is the meta data worksheet whose data contains information about when the data was captured, which Section and Category the data came from and the values of the Filters at the time the data was output.

Date:	01/15/2017
Time:	3:23 PM
Username:	tpsdemo
Report:	Patient Profile
Section:	Territory Overview
Category Number:	1
Component Type:	multihorizstackbar
Drop-Down Value:	Patient
Filter Value(s):	
Zone	NATION
Region	All
Territory	All

Following is an example output from a Grid. Note column G which contains the values from a Sparkline

	A	B	C	D	E	F	G	H	I	J	K
1	Grouping		HCP	Script Info					HCP Info		
2	Primary	Secondary	Rank	Segment	Total Referrals	TRx	6mo Product TRx Trend	Total Patients	City	State	Territory
3	RAGHU, GANESH	0162409	1	Defend	86	513	6,30,39,41,39,38	72	Seattle	WA	SEATTLE, WA
4	FLAHERTY, KEVIN	1574158	2	Build	45	271	2,15,20,21,19,25	37	Ann Arbor	MI	DETROIT, MI
5	VEGA OLIVO, MICHELLE	6122709	3	Defend	51	246	4,13,16,13,12,14	41	Philadelphia	PA	PHILADELPHIA
6	PASCUAL, JOSE	1585794	4	Defend	48	240	4,20,24,13,13,21	35	Hudson	FL	SAINT PETERSBURG, FL
7	HAMBLIN, MARK	2402718	5	Build	36	225	1,9,12,12,15,14	30	Kansas City	KS	KANSAS CITY, MO
8	RAY, DANIEL	0775844	6	Defend	22	208	2,13,10,12,9,14	20	Evanston	IL	CHICAGO, IL
9	DEANDRADE, JOAO	1836905	7	Build	28	199	0,8,12,14,14,15	27	Birmingham	AL	BIRMINGHAM, AL
											READING, PA

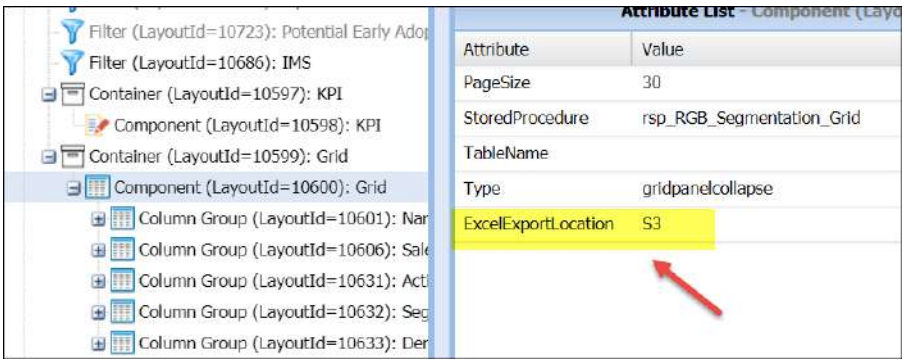
Following is a list of all Component types that support Excel data export: Area, Bar, Column/Line, Grids (both pageable and non-pageable Grids), Heat Map, Multi-stacked Horizontal Bar, Pie, Scatter, Tree and Waterfall.

Additional items to note about Excel Export Data:

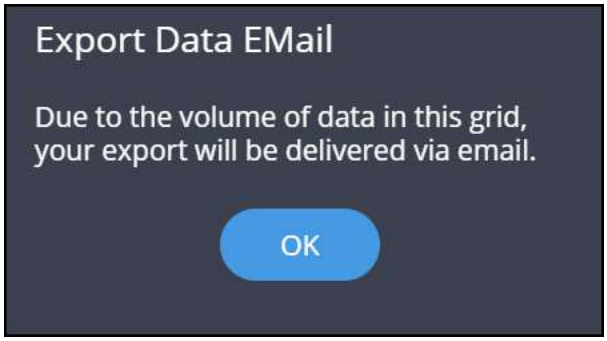
- Export is supported only on desktop computers and specifically not iPads where the operating system does not support file downloads.
- For pageable Grids all rows, (and not just those currently displayed), will be output.
- Regarding icons in grids:
  - The list of supported icons is located here.
  - Columns that contain just icons are supported. Columns that contain a mix of icons and text (for example, a red circle and the text '-12') are not supported.
- The file naming convention is: DataExport_ReportName_SectionLongTitle_CategoryDisplayOrder_ComponentType_YYYYMMDD.xlsx. For example: DataExport_ExecutiveInsights_NationalTrends_1_Bar_20170120.xlsx.

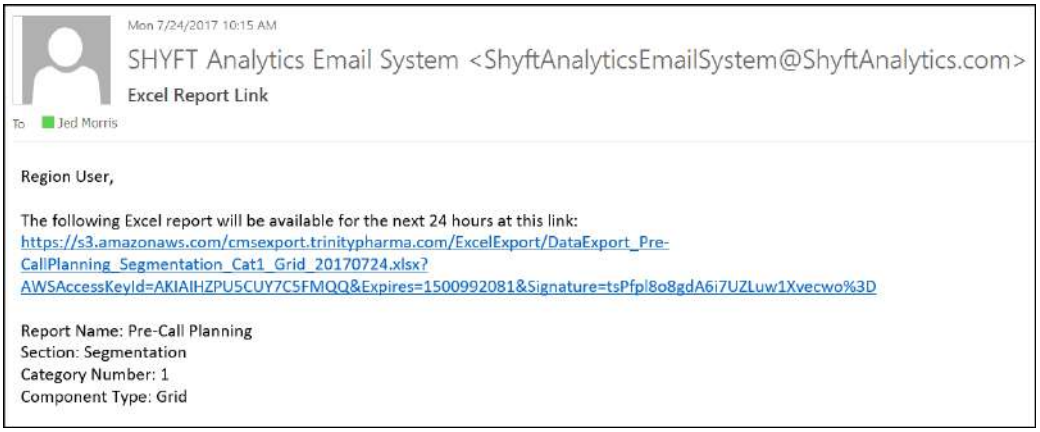
Note that large pageable grids can take several minutes to download locally and during the download a "preparing data…" message appears. Most users naturally think the system can't be used while the Excel data is being prepared. In fact the system can be fully used, however it is understandable users would think otherwise.

To create a better user experience, pageable grids may optionally be sent through e-mail using the Component attribute ExcelExportLocation. The only possible value for this attribute is 'S3'. Note all pageable grids will require this option – test the performance on a per Component basis.



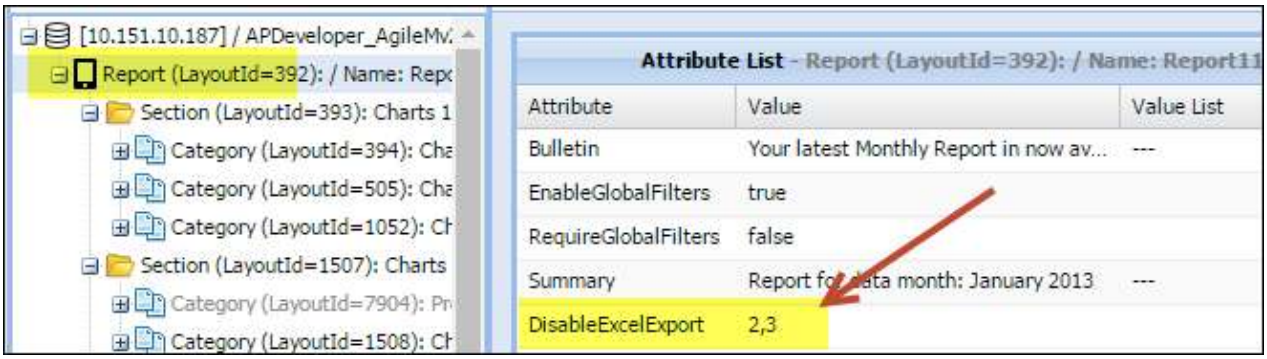
When exporting the component, the user will get the following message and e-mail. Note that the link will expire in 24 hours.





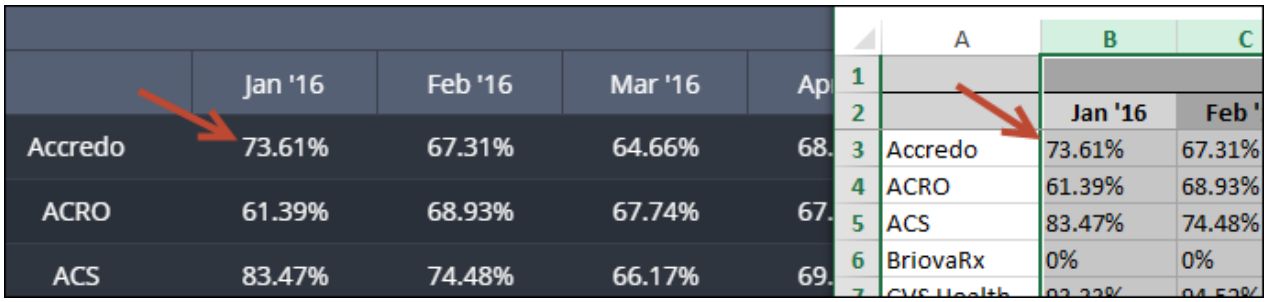
Implementation is automatic – the system is fully functional without doing anything. However, the system may be turned off as follows:

- Export may be turned off for the entire Project using the tblAppConfig property DisableExcelExport. For details click here.
- Export can be disabled at the user group level using the Report configuration attribute DisableExcelExport. In the following example, user groups 2 and 3 will not have access to Excel Export.
- To turn off export selectively for some Report(s) in a Project, do not make an entry in tblAppConfig and then list all user groups in the Report configuration attribute.
- Individual components may be turned off with the Component attribute: DisableExcelExport.



Note that AP supports grid columns whose values are numeric, yet presented to the UI as strings. This is often done, for example, to display a percent sign, a dollar sign or an up/down Chevron. While this works well for the UI, it can result in grid Excel export columns whose values are strings and thus can't be worked with as numbers.

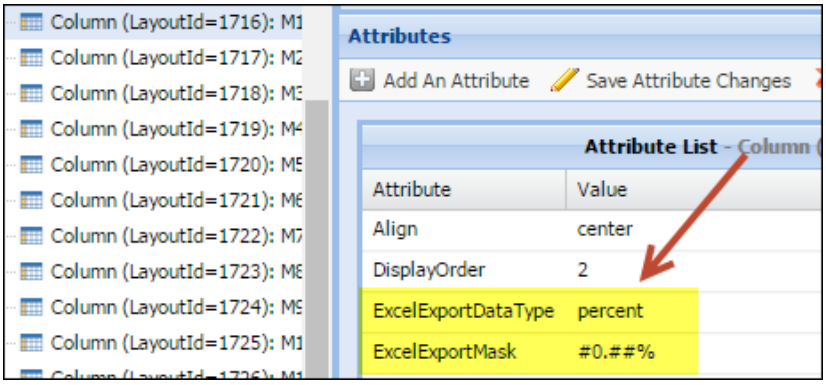
Note this grid and its associated Excel export file:



	Jan '16	Feb '16	Mar '16	Ap
Accredo	73.61%	67.31%	64.66%	68.
ACRO	61.39%	68.93%	67.74%	67.
ACS	83.47%	74.48%	66.17%	69.

	A	B	C
1			
2		Jan '16	Feb '
3	Accredo	73.61%	67.31%
4	ACRO	61.39%	68.93%
5	ACS	83.47%	74.48%
6	BriovaRx	0%	0%

The values look correct in the UI, but in Excel the values are strings (as noted by the fact that the values are left aligned). In cases where this occurs, use the configuration attributes ExcelExportDataType and ExcelExportMask to control the values in Excel.



ADDITIONAL CONTROLS

Favorites

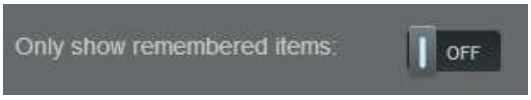
APDeveloper Report 01:

Section: Additional Controls
Container: Favorites Example 01
Stored Proc: rsp_FavoritesExample01

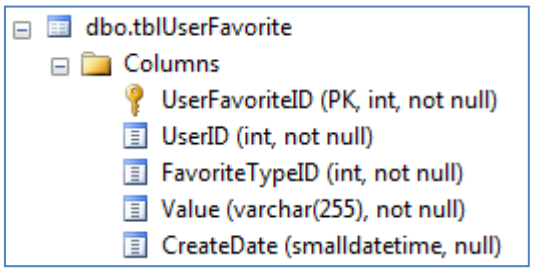
Background:
Favorites provide the end user with functionality to control, for example in client listings, which clients will be displayed.

In the following example, the user has indicated that Dr. Jones is not a favorite (by de-selecting the star in the most left hand column).

The Filter dialog box then supports the ability to turn the Favorites display on or off. (This means a Category that supports Favorites will have to have at least one Filter).

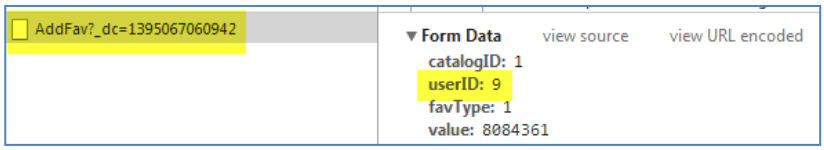


Tables:
tblUserFavorite records a users Favorite selections:

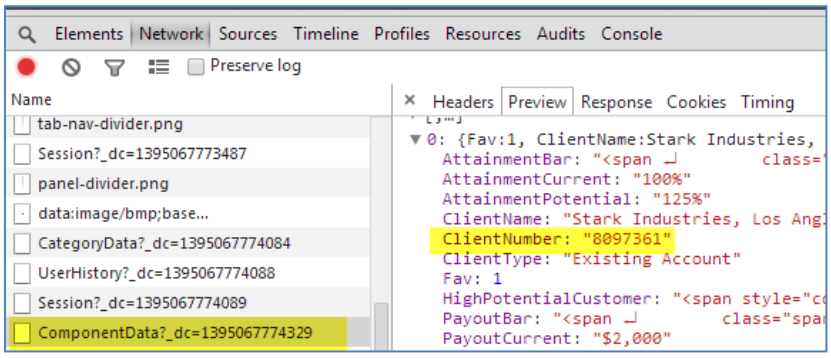


Note that Favorites are recorded by UserID so different users will have different Favorites.

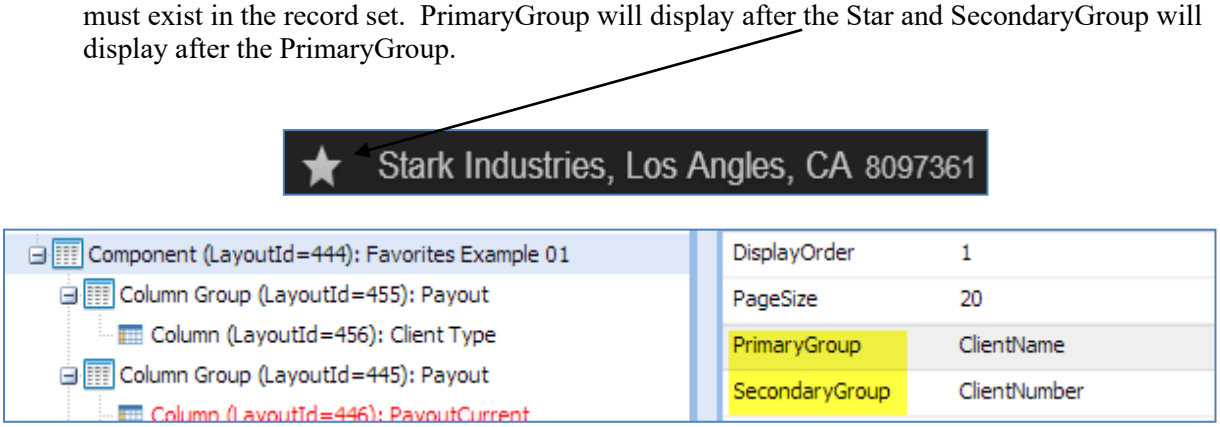
It is instructive to look at the debugger upon adding a Favorite. Note that userID is passed into the AddFav stored procedure and a new row will be added to tblUserFavorite.



AgileM Studio setup:
The Category must contain a FavoriteField attribute, in this case ClientNumber, which also must be a column in the Component recordset.



The Component must contain the Attributes PrimaryGroup and SecondaryGroup, each of which must exist in the record set. PrimaryGroup will display after the Star and SecondaryGroup will display after the PrimaryGroup.



Stored Procedure:
Get the Favorites setting as follows (expect either NULL for off, or 1 for on) and code something like the following in the WHERE clause.

```
-- Get filter values
    DECLARE @vchFavoriteFlag VARCHAR(30) = (SELECT dbo.GetFilterValue(@tblFilterNameValue, 'Fav'))
```

To return only the favorites, use code like the following:

```
WHERE
      @vchFavoriteFlag IS NULL
  OR (@vchFavoriteFlag = '1' AND ClientNumber in (SELECT value FROM tblUserFavorite WHERE UserID = @vchUserID))
```

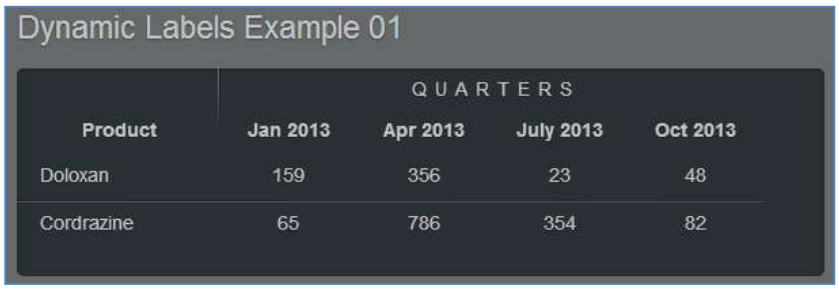
Dynamic Labels

APDeveloper Report 01:

Section: Additional Controls
Container: Dynamic Labels Example 01
Stored Proc: rsp_DynamicLabelsExample01

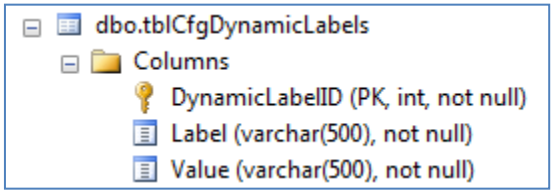
Background:
Dynamic Labels provide a way for the column headers of a Grid to change, from one reporting period to another, without BI Developer intervention.

Consider the following case:

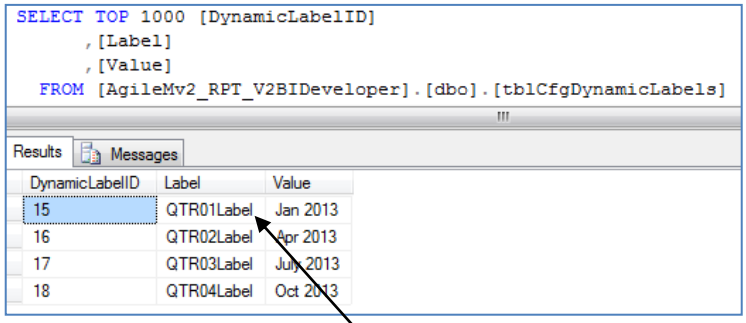


The column headers Jan 2013 → Oct 2013 will need to change regularly. Normally, grid titles are hard coded in the Title Attribute of a Column. That is, whatever text value is in the Title Attribute will appear as the Grids header text.

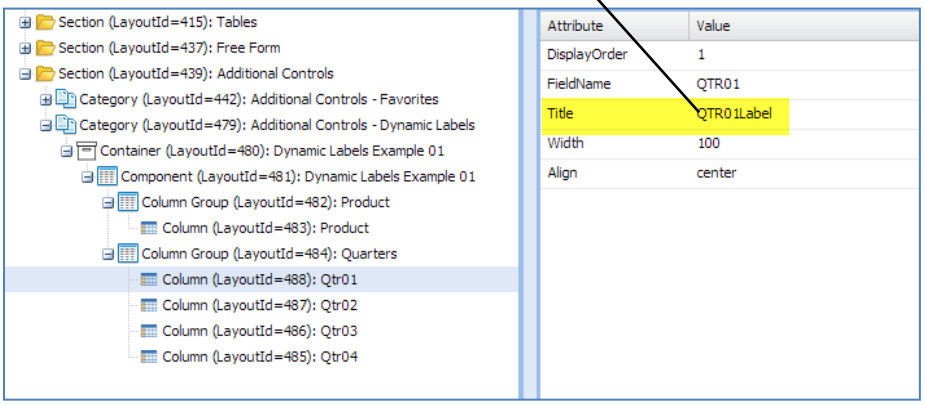
Tables:
tblCfgDynamicLabels is used to establish column header that can change.



For this example, note the records in tblCfgDynamicLabels:



AgileM Studio setup:
Now, in AgileM Studio, rather than entering a constant hard coded value for the Title Attribute, enter a value that will tie back to tblCfgDynamicLabels.



When called to display the report, the mobile application will now tie the Attribute Title 'QTR01Label' back to the tblCfgDynamicLabels.Label = 'QTR01Label' and return the Value column (in this case the text: Jan 2013).

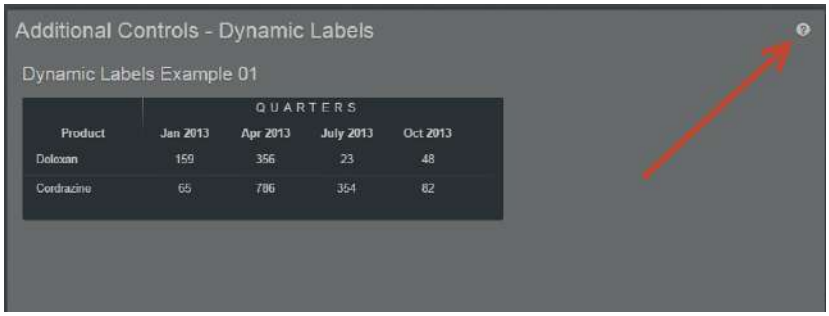
It is now the DB's responsibility, in each reporting period, to update the rows in tblCfgDynamicLabels. As a BI Developer, you should provide details to the DB regarding which tblCfgDynamicLabels rows the*y* will need to adjust

Note that dynamic labels may be used for Columns, Column Groups and Container Titles.

Category Level Help Button

APDeveloper Report 01:
Section: Additional Controls

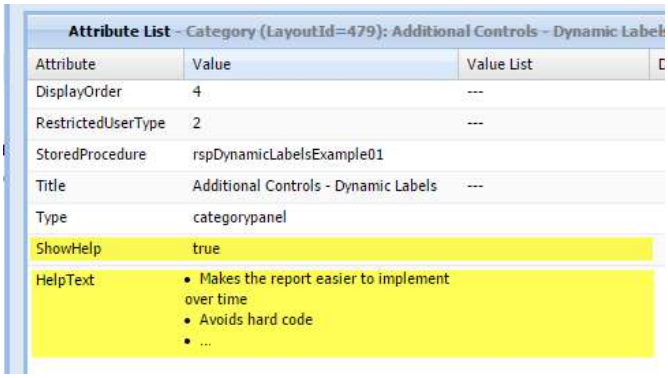
Background:
A help button can be implemented at the category level. It can contain any text written in HTML format. It appears in the upper right corner of the category page.



Once this button is clicked on a full screen information box pops up with the title of the category followed by "Help" and the designated text.

Note: A link to a website or email can be included in the help text.

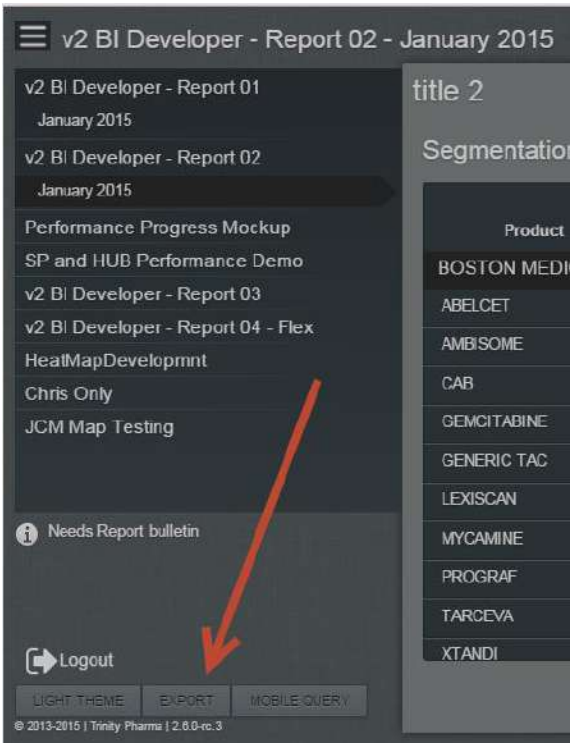
AgileM Studio setup:
To get this feature add the attribute "ShowHelp" and set the value equal to "true". Also, add the "HelpText" attribute and insert the desired text in HTML format.



Export to Excel Button

APDeveloper Report 02:
Section: All

Exporting directly from the V2 mobile application to Excel is now available. The Export button is located in the left panel in the mobile app. Please note the Export button is at the report level.



The Mobile V2 BI only needs to make one update to have the Export to Excel button available. In the APP database in [tblReport] an ExcelReportID must be populated. This ID is determined by the Excel BI. There are no updates to V2 stored procedures or Agile M Studio by the Mobile V2 BI.

	ReportID	Title	ReportName	CatalogTypeID	ExcelReportID	DisplayOrder	Active
1	11	v2 BI Developer - Report 01	v2 BI Developer - Report 01	1	NULL	1	1
2	13	v2 BI Developer - Report 02	v2 BI Developer - Report 02	1	123	2	1
3	22	Performance Progress Mockup	Performance Progress Mockup	1	NULL	3	1
4	23	SP and HUB Performance Demo	SP and HUB Performance Demo	1	NULL	23	1

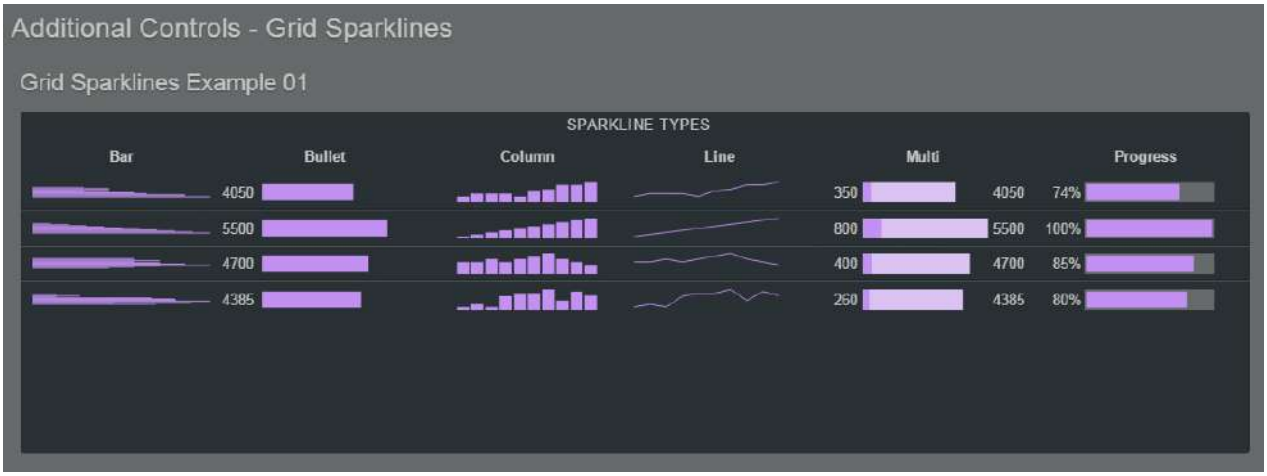
Grid Sparklines

APDeveloper Report 01:

Section: Additional Controls
Category: Additional Controls – Grid Sparklines
Container: Sparklines Example 01
Stored Proc: rspAdditionalControlsSparklines

Background:

Grid sparklines provide a visual representation of data within a grid. Currently, there are six sparkline types to choose from (bar, bullet, column, line, multi, progress), covering a wide variety of use cases. For example, bar, column and line sparklines could be a good choice to track trends – how did Volume trend over the past 13 months for instance. Multi sparkline on the other hand could be used for NRX & TRX relationship, and a progress sparkline could be used to track goal attainment or market share for instance.

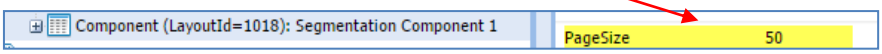


Stored Procedure:

Within the select statement for the grid component, a FieldName, which matches the FieldName attribute of the associated column layout ID within the configuration, needs to be defined and set equal to a function call that returns the html string, which creates an appropriate sparkline on the front end. Refer to rspAdditionalControlsSparklines, which contains function calls to all of the above sparkline types.

PAGING STORED PROCEDURES

Background:
When dealing with front-end result sets that return anywhere from a few thousand to a few million rows, keeping performance in mind is crucial. For this reason, the Analytic Platform application has the ability to split the result set across many pages and return a specified number of rows per page within the grid component. On the initial load, the grid loads the first page with the number of rows specified in the PageSize attribute of the component level within the configuration. Then, when the user scrolls or swipes to the bottom of the grid (the last row of the page) the grid reloads the next page with a new set of rows, and this process is repeated every time the user reaches the last row of any current page.



Often, because of the complexity of the result sets returned with paging stored procedures, paging stored procedures include logic for non-fixed filters, sorting, and ranking functionalities. A template stored procedure, rspTemplate_PagingSP, has been created to ensure that BI developers have a good starting point for these types of efforts.

In addition to standard input parameters, rspTemplate_PagingSP includes 3 additional input parameters:

```
(
      @filterTableXML XML
    ,@vchSortString varchar(max) = ''
    ,@vchFilterString varchar(max) = ''
    ,@vchRankBy varchar(500) = ''
    ,@intPageNumber int = 1
    ,@intPageSize int = 50
)
```

Comments regarding the input parameters:

- **@vchSortString** – sorting string that includes the ORDER BY clause
- **@vchFilterString** – optional filter string that includes the WHERE clause
- **@vchRankBy** – rank string or a 'sort within a sort'
- **@intPageNumber** – inputs the page number that the front-end grid last loaded
- **@intPageSize** – inputs the number of rows per page

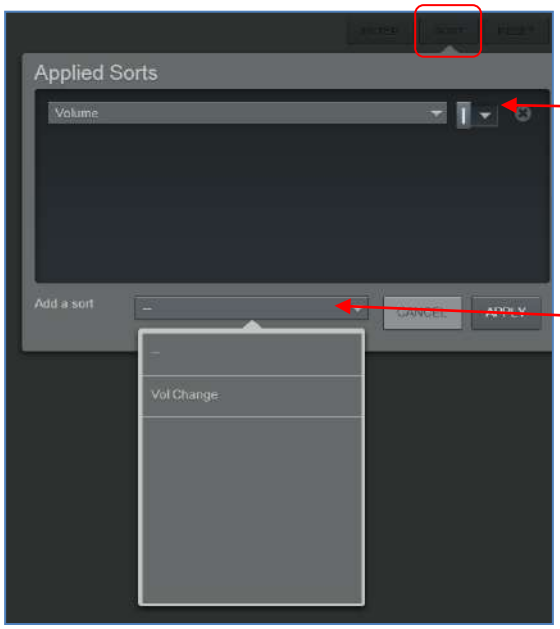
****NOTE:** All of the above mentioned input parameters will be used in the following examples, which point to stored procedures that are written according to the Analytic Platform implementation standards. Since T-SQL functions in a way that does not allow for variables to be passed in the order by (@vchSortString) and the where clause (@vchFilterString), these example procs are written using dynamic SQL.

Sorting

APDeveloper Report 01:

Section: Paging
Container: Example 01 – Non-Fixed Filters and Sorting
Stored Proc: rsp_Paging_NonFixedFiltersSorting

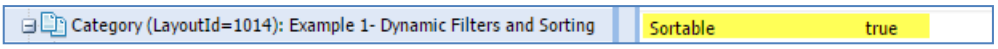
Background:
Sorting provides the end user with the functionality to sort the data based on the selected column or columns. In order to specify sorting, the user needs to select a column within the sort drop-down on the front end.



Specifies whether the sort should be ascending or descending. In this case it is descending.

User can click this button to view and add columns to sort on by clicking columns displayed in the drop-down.

AgileM Studio setup:
The Category must contain the Sortable attribute, which needs to be set as 'true'

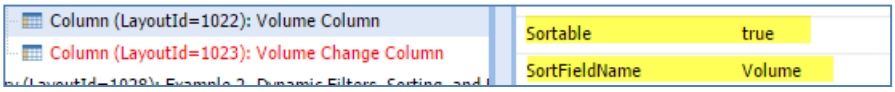


The component must contain a StoredProcedure attribute that points to the stored procedure, which returns the result set for the grid. In this case it is rsp_Paging_NonFixedFiltersSorting. Note this means the component must be a paging grid.

****NOTE:** in AgileM configuration for this example you can notice that the stored procedure attribute on the category level points to the dummy stored procedure rsp_Paging_DummySP. Usually, when developing for a client, the stored procedure attribute on the component level would point to the stored proc that returns the result set for the grid, and the stored procedure attribute on the category level would point to the stored proc that contains the code for the KPI freeform box for example.

The column may optionally contain the Sortable attribute which needs to be set as 'false' if the column should not be sortable. There is no need to set Sortable='true' since this is the default value.

The SortFieldName attribute needs to point to the column that is returned from the result set within the SQL stored procedure (usually SortFieldName and FieldName point to the same column)



Stored Procedure:
Stored procedure rsp_Paging_NonFixedFiltersSorting demonstrates the agreed standards that were developed in order to optimize performance and improve implementation and maintenance efforts. This approach should be followed while writing code that uses dynamic SQL to return a result set with sorting enabled on the front end.

First, a temporary table should be created that has all of the columns that will be returned on the front end. In addition, this table should be populated with a filtered result set based on fixed filters from the front end. The code might look like this:

```
SELECT
     Name
    ,EntityNumber
    ,Product
    ,Volume
    ,VolGr
INTO #tempMainTbl
FROM tblRptSegmentation_03 main
WHERE Market LIKE @vchMarket
```

Next, the dynamic SQL code is written as a common table expression and follows the example below with everything being the same from one implementation to another except for the columns in the select statement:

```
SET @sqlString = N'
    WITH gridResults AS (
        SELECT
            Name
            ,EntityNumber
            ,Product
            ,Volume = dbo.udf_FormatGridDecimal(Volume, 0)
            ,VolGr = dbo.udf_FormatGridChangeInteger(VolGr)
            ,RowNum = ROW_NUMBER() OVER (' + CASE @vchSortString WHEN '' THEN N'ORDER BY ' + @vchDefaultSortString ELSE @vchSortString END + N' )
        FROM #tempMainTbl '
        +
        CASE @vchFilterString
            WHEN '' THEN ''
            ELSE @vchFilterString
        END
        +
        N')

    SELECT
         *
        ,totalCount = (SELECT COUNT(*) FROM gridResults )
    FROM gridResults
    WHERE RowNum BETWEEN @intPageSize * (@intPageNumber - 1) AND @intPageSize * @intPageNumber
    ORDER BY RowNum ASC;'
```

RowNum, is a user defined column that establishes the row number with a built in ROW_NUMBER function and uses the OVER clause to get the proper sorting. Within the OVER clause, there is a CASE statement on the @vchSortString that gets passed from the front end. When this string is empty, in other words when the user has not specified the sort on the front end, the default sort, which is passed through @vchDefaultSortString kicks in. Otherwise, if the user has specified sorting, the result set is sorted based on that specification.

```
RowNum = ROW_NUMBER() OVER (' + CASE @vchSortString WHEN '' THEN N'ORDER BY ' + @vchDefaultSortString ELSE @vchSortString END + N' )
```

Variable @vchDefaultSortString needs to be defined and set within the stored procedure:

```
DECLARE  @vchDefaultSortColumn      VARCHAR(25)  = 'Volume'
        ,@vchDefaultSortDirection   VARCHAR(10)  = 'DESC'
        ,@vchDefaultSortString      VARCHAR(50);

SET @vchDefaultSortString = @vchDefaultSortColumn + ' ' + @vchDefaultSortDirection;
```

Note that this code can be reused, however the developer needs to specify values for @vchDefaultSortColumn and @vchDefaultSortDirection, which are 'Volume' and 'DESC', respectfully, in this case.

RowNum is also used for paging purposes within the where clause of the select statement that runs the common table expression:

```
WHERE RowNum BETWEEN @intPageSize * (@intPageNumber - 1) AND @intPageSize * @intPageNumber
```

The developer should also note the placement of @vchFilterString, used to filter the result set with non-fixed filters, within the code; this can remain consistent from one project to another:

```
+
CASE @vchFilterString
    WHEN '' THEN ''
    ELSE @vchFilterString
END
```

Favorites

APDeveloper Report 01:

Section: Paging
Container: Example 02 – Non-Fixed Filters, Sorting and Favorites
Stored Proc: rsp_Paging_NonFixedFiltersSortingFavorites

Background:
Frequently, the favorites functionality is incorporated into paging stored procedures. Below, the developer can find a reference on how to do so. For reference regarding configuration setup, refer to Favorites section within this guide.

Stored Procedure:
Stored procedure rsp_Paging_DynamicFiltersSortingFavorites is an expansion of rsp_Paging_DynamicFiltersSorting with the favorites functionality added to it. This section highlights the edits, which are the differences between the two stored procs.

Note that @userID and @showFavorites have to be declared:

```
DECLARE @userID INT = (SELECT dbo.GetFilterValue(@tblFilterNameValue,'userid'))
       ,@showFavorites BIT = (SELECT dbo.GetFilterValue(@tblFilterNameValue,'Fav'));
```

Next, the temporary table needs to include favoriteValue column, and it needs to be joined to tblUserFavorite.

```
SELECT
     Name
    ,EntityNumber
    ,Product
    ,Volume
    ,VolGr
    ,favoriteValue = fav.Value
INTO #tempMainTbl
FROM tblRptSegmentation_03 main
    LEFT JOIN tblUserFavorite fav
        ON main.EntityNumber = fav.Value AND fav.UserID = @userID
WHERE Market LIKE @vchMarket
```

The dynamic SQL part of the stored procedure needs to be modified as well:

```
SET @sqlString = N'
    WITH gridResults AS (
        SELECT
            Name
           ,EntityNumber
           ,Product
           ,Volume = dbo.udf_FormatGridDecimal(Volume, 0)
           ,VolGr = dbo.udf_FormatGridChangeInteger(VolGr)
           ,RowNum = ROW_NUMBER() OVER (' + CASE @vchSortString WHEN '' THEN N'ORDER BY ' + @vchDefaultSortString ELSE @vchSortString END + N' )
           ,Fav = CASE WHEN favoriteValue IS NOT NULL THEN 1 ELSE 0 END
        FROM #tempMainTbl main '
        +
        CASE @vchFilterString
            WHEN '' THEN 'WHERE '
            ELSE @vchFilterString + N' AND '
        END
        +
        N'
        1 = CASE
                WHEN @showFavorites = 1 THEN CASE WHEN favoriteValue IS NOT NULL THEN 1 ELSE 0 END
                ELSE 1
            END )

    SELECT
         *
        ,totalCount = (SELECT COUNT(*) FROM gridResults )
    FROM gridResults
    WHERE RowNum BETWEEN @intPageSize * (@intPageNumber - 1) AND @intPageSize * @intPageNumber
    ORDER BY RowNum ASC;'
```

Note that this column is named Fav, which matches the @showFavorites declaration:

```
@showFavorites BIT = (SELECT dbo.GetFilterValue(@tblFilterNameValue,'Fav'))
```

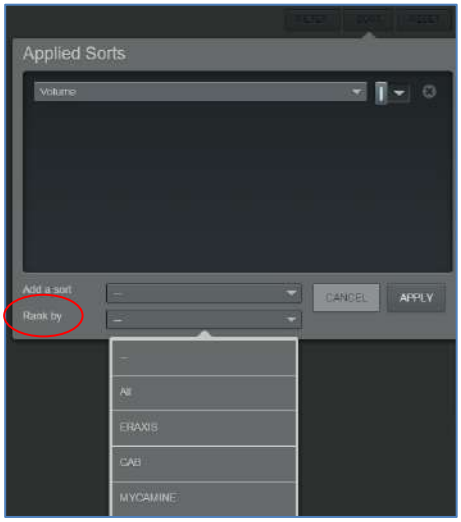
All aspects of incorporating the favorites functionality to the code can be replicated from this example from one project to another.

Ranking

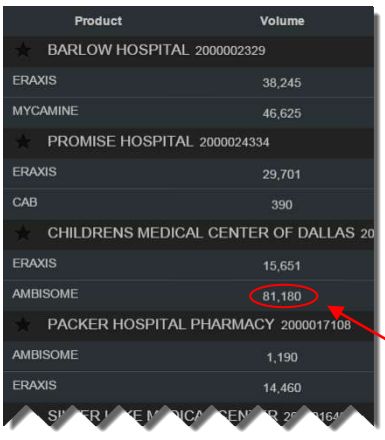
APDeveloper Report 01:

Section: Paging
Container: Example 03 – Non-Fixed Filters, Sorting, Favorites, and Ranking
Stored Proc: rsp_Paging_NonFixedFiltersSortingFavoritesRanking

Background:
Accessed by the 'Sort' button on the front end, ranking allows end users to determine how their result sets will be sorted; it can be viewed as a sort within a sort. An example is ranking by product, allowing the end user to specify a product in the 'Rank by' and then to apply a sort, such as sort by volume in a descending order based on that rank:

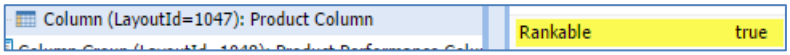


The following screenshot demonstrates a result set that is ranked by product named 'Eraxis' and sorted by Volume in a descending order:



Note that there is an instance in which Volume of another product is higher than any of Eraxis' volumes, but that has no influence on sorting because the field used in the 'Rank by' is Eraxis.

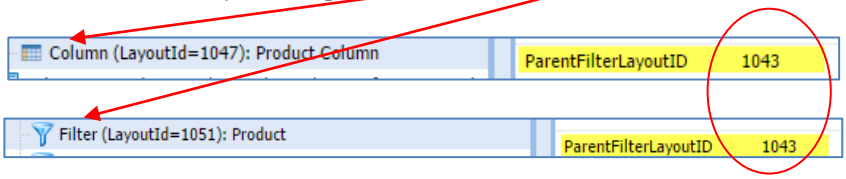
AgileM Studio setup:
The column must contain a Rankable attribute, which needs to be set as 'true'



The column also needs to link to a filter that is defined on the category level. This is done by setting the SiblingParentLayoutID's Value to the layout ID of the filter that corresponds to the rankable column. In the following example, layout ID of the filter that corresponds to the rankable column is 1051:

****NOTE:** due to the app limitation, it is not possible to implement a rankable column that does not correspond to a filter. A filter that matches the rankable column must be defined on the category level.

If the filter to which the rankable column is connected has a parent filter (refer to cascading filters), then ParentFilterLayoutID attribute needs to be set at the column level as well. Its value should match the ParentFilterLayoutID specified on the filter level:



Stored Procedure:
Stored procedure rsp_Paging_NonFixedFiltersSortingFavoritesRanking is an expansion of rsp_Paging_NonFixedFiltersSortingFavorites and demonstrates how to incorporate ranking into the stored procedure code. Required edits to the code that enable ranking are highlighted below.

Ranking is a sort within a sort. Therefore, RowNum field needs to be set up so that it is ordered by Rank, which is defined in a subquery and is the exact string used to enable sorting.

```
SET @sqlString = N'
    WITH gridResults AS (
        SELECT
            RowNum = ROW_NUMBER() OVER (ORDER BY CASE WHEN [Rank] IS NULL THEN 0 ELSE 1 END DESC, [Rank] ASC)
            ,Name
            ,seg.EntityNumber
            ,Product
            ,Volume = dbo.udf_FormatGridDecimal(Volume, 0)
            ,VolGr = dbo.udf_FormatGridChangeInteger(VolGr)
            ,Fav = CASE WHEN favoriteValue IS NOT NULL THEN 1 ELSE 0 END
        FROM #tempSegmentation seg
            LEFT JOIN (
                    SELECT EntityNumber, [Rank] = ROW_NUMBER() OVER (' + CASE @vchSortString WHEN '' THEN N'ORDER BY ' + @vchDefaultSortString ELSE @vchSortString END + N')
                    FROM #tempSegmentation
                    WHERE Product = @vchRankByProduct
                    ) r
                ON seg.EntityNumber = r.EntityNumber '
        +
        CASE @vchFilterString
            WHEN '' THEN 'WHERE '
            ELSE @vchFilterString + N' AND '
        END
        +
        N'
        1 = CASE
                WHEN @showFavorites = 1 THEN CASE WHEN favoriteValue IS NOT NULL THEN 1 ELSE 0 END
                ELSE 1
            END )

    SELECT
        *
        ,totalCount = (SELECT COUNT(*) FROM gridResults )
    FROM gridResults
    WHERE RowNum BETWEEN @intPageSize * (@intPageNumber - 1) AND @intPageSize * @intPageNumber
    ORDER BY RowNum ASC;'
```

Note that the subquery that defines Rank is left joined to the temporary table. This part of code can be reused from implementation to implementation and stays consistent for the most part. However, the following change:

1) The column that is used to join on + the column that is used in the select statement with Rank in order to retrieve a proper result set within the rank subquery

```
FROM #tempSegmentation seg
    LEFT JOIN (
            SELECT EntityNumber, [Rank] = ROW_NUMBER () OVER (' + CASE @vchSortString WHEN '' THEN N'ORDER BY ' + @vchDefaultSortString  ELSE @vchSortString END + N')
            FROM #tempSegmentation
            WHERE Product = @vchRankByProduct
              ) r
        ON seg.EntityNumber = r.EntityNumber '
```

2) The rankable column that is used in the where clause:

```
FROM #tempSegmentation seg
    LEFT JOIN (
            SELECT EntityNumber, [Rank] = ROW_NUMBER () OVER (' + CASE @vchSortString WHEN '' THEN N'ORDER BY ' + @vchDefaultSortString  ELSE @vchSortString END + N')
            FROM #tempSegmentation
            WHERE Product = @vchRankByProduct
              ) r
        ON seg.EntityNumber = r.EntityNumber '
```

Note the @vchRankByProduct user defined variable that is used in the where clause above. This variable is defined and set by the developer in the beginning of the stored procedure. It is set to the value based on the @vchRankBy input parameter, which is passed from the front end by the app and is populated with the value that should be passed in the rank subquery's where clause. Thus, @vchRankByProduct is used in the rank subquery's where clause to determine the sub-sort.

AUTOGENERATION OF STORED PROCEDURES

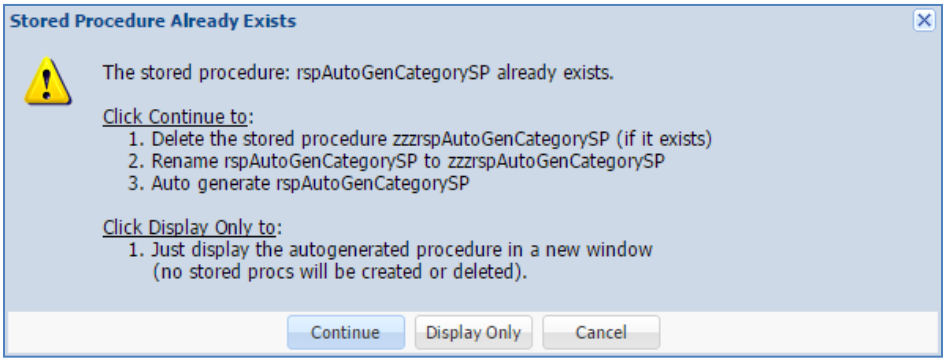
Background

AgileM Studio supports the automatic generation of stored procedures for selected Component types. By entering required attribute information for selected Layouts, AgileM Studio is able to actually generate the stored procedure that will supply the record set for the Component.

For example, the Grids in a Category stored procedure can be automatically generated by supplying the TableName attribute of a gridpanelcollapse Component and the FieldName attribute of each Column in the gridpanelcollapse Component.

Note that in all cases the stored procedure that is automatically generated will be nearly complete, but may need some manual adjustment.

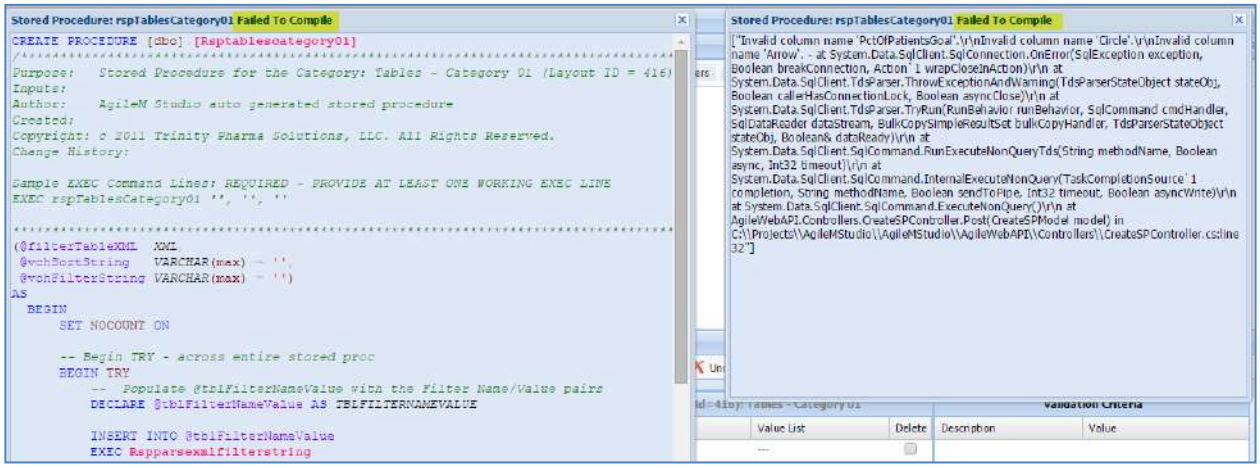
Also note that AgileM Studio, when automatically creating a new stored procedure, will not overwrite an existing stored procedure. When the stored procedure already exists, the following dialog box displays indicating the options. Note that one option is to just display the auto generated stored proc without creating or deleting any stored procs.



After generating the dynamic SQL AgileM Studio will attempt to compile the code, and upon success or failure, will then display the code in a separate window. If the code does compile successfully, then a new stored procedure will have been created. If the code fails to compile then two new windows are presented – one window containing the failed code and one window containing the compile error message.

For example, a failure could look like the following. The window on the left is the dynamic SQL that failed to compile (and thus did not create a new stored procedure), and the window on the right displays the compile error message.

In most cases, even though the code failed to compile, you will still find this a time saver, because the code can be copied/pasted into SQL Server Management Studio where only minor adjustments will be sufficient to compile successfully.



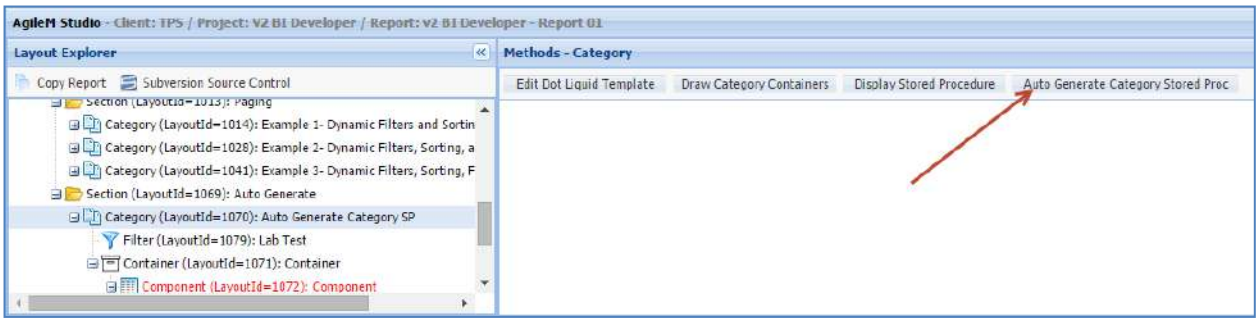
The following reviews the stored procedures that can be automatically created and details the attributes are required for the auto generation process.

Auto generation of Category Stored Procedures

APDeveloper Report 01:

Section: Auto Generate
Category: Auto Generated Category SP
Stored Proc: rspAutoGenCategorySP

Generate a Category Stored Procedure by selecting the Auto Generate Category Stored Proc button when a Category Layout is selected.



The automatically generated stored procedure will do the following:

- Create the basic structure of the stored proc using the standards established by the template: rspTemplate_NonPagingSP. (See here for more details on the non-paging stored procedure standard).
- Process all fixed Filters associated with the Category and generate the standard dynamic SQL required for all fixed filters. No code is generated for non-fixed filters.
- Generate the code associated with all Components that are of type gridpanelcollapse that do not have their own StoredProcedure attribute, (i.e. those grids that are non-pageable), including the WHERE clause for all fixed filters.

The Layouts and Attributes required are:

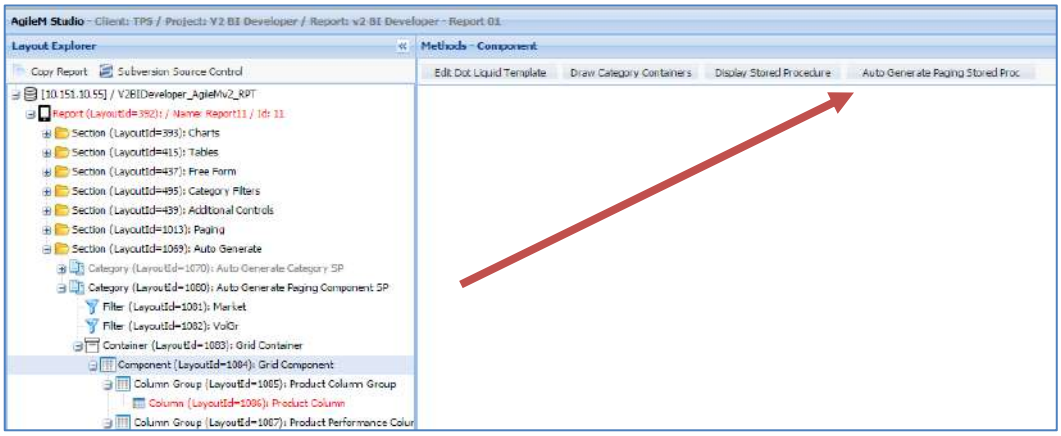
- Category layout must have the StoredProcedure attribute.
- Fixed Filter layouts are optional, but if present must include the FieldName attribute.
- Components of type gridpanelcollapse are optional, but if present must include the TableName attribute.

Auto generation of Paging Component Stored Procedures

APDeveloper Report 01:

Section: Auto Generate
Category: Auto Generated Component SP
Stored Proc: rspAutoGenPagingGridComponent

Generate a Paging Component Stored Procedure by selecting the Auto Generate Paging Stored Proc button when a Component Layout is selected.



The automatically generated stored procedure will create a paging component stored procedure based on the attributes defined in the layouts associated with the component. There are three templates that can be generated; refer to Sorting, Favorites, or Ranking sections in the guide for the attributes that need to be set in order for each functionality to work in the app. Following these guidelines should ensure that the automated sp is generated properly.

The necessary attributes that should be set in AgileM studio prior to hitting the Auto Generate Paging Stored Proc button are:

- Component layout:
  - StoredProcedure
  - PageSize
  - PrimaryGroup
  - SecondaryGroup
  - TableName
- Component type must be gridpanelcollapse
- Fixed Filter layouts are optional, but if present must include the FieldName attribute
- If the paging grid should have favorites enabled
  - FavoriteField is required on the category level
- If the paging grid should have ranking enabled
  - Column should have Rankable and SiblingParentLayoutID attributes

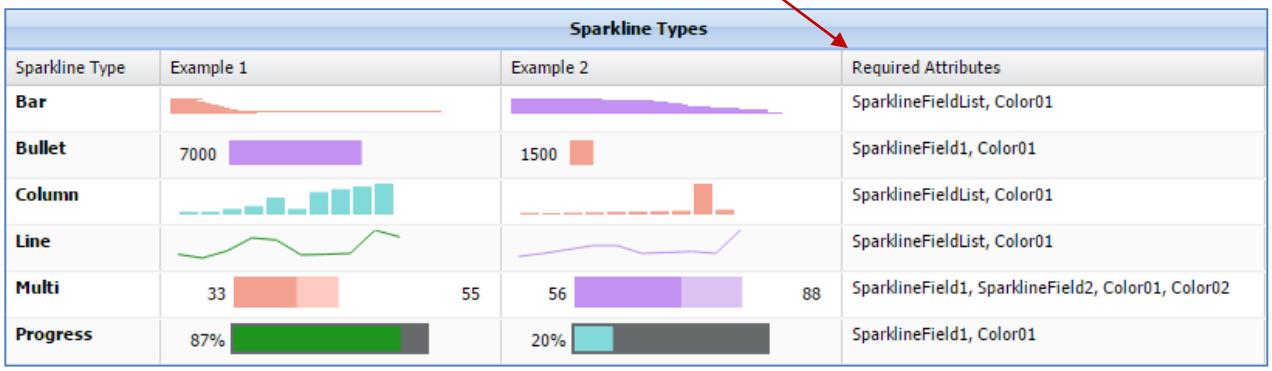
Including Grid Sparklines in the Auto Generation Process

APDeveloper Report 01:

Section: Auto Generate
Category: Auto Generated Category SP; Auto Generated Component SP
Stored Proc: rspAutoGenCategorySP; rspAutoGenPagingGridComponent

Grid Sparklines creation is supported by the auto generation process of both paging and non-paging stored procedures.

Sparklines are defined at the column level within AgileM Studio. First, SparklineType attribute needs to be defined and specified using the dropdown attribute value list. Then, depending on the sparkline type a certain set of attributes needs to be defined as suggested by the Sparkline Types chart within AgileM Studio.



Sparkline Types

Sparkline Type	Example 1	Example 2	Required Attributes
Bar			SparklineFieldList, Color01
Bullet	7000	1500	SparklineField1, Color01
Column			SparklineFieldList, Color01
Line			SparklineFieldList, Color01
Multi	33 55	56 88	SparklineField1, SparklineField2, Color01, Color02
Progress	87%	20%	SparklineField1, Color01

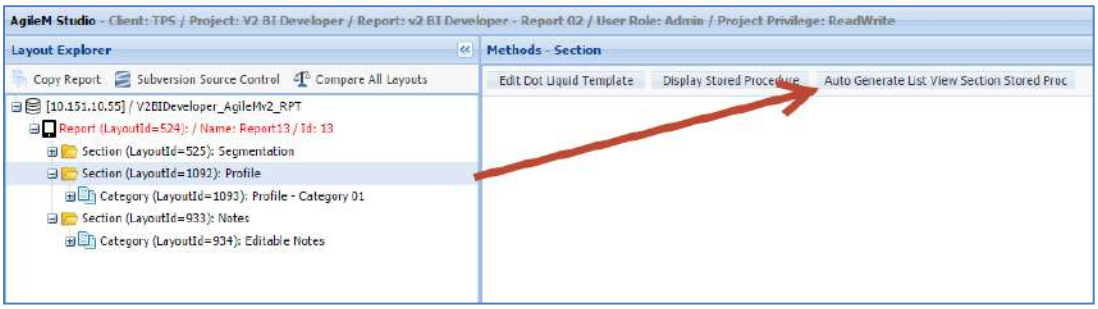
Refer to Attribute Help within AgileM Studio for information about a particular attribute or see here for more information on grid sparklines.

Auto generation of List View Section Carousel Stored Procedures

APDeveloper Report 03:

Section: Auto Generate List View
Category: /
Stored Proc: rspAutoGenListViewSectionCarousel

Generate a List View Section Carousel stored procedure by selecting the Auto Generate List View Section Stored Proc button when a Section with the type of 'listviewsectioncarousel' is selected.



The automatically generated stored procedure will create a list view stored procedure based on the attributes defined in the Section layout.

The necessary attributes that should be set in AgileM studio prior to hitting the Auto Generate List View Stored Proc button are:

- Section layout:
  - FavoriteField
  - FavoriteTypeID
  - FieldName
  - ListDisplayFieldName
  - OrderByFieldName
  - SiblingLayoutID
  - StoredProcedure
  - TableName
- Section type must be listviewsectioncarousel

See the Help text within AgileM Studio for a specific attribute description, or click here for more details regarding listviewsectioncarousel typed sections.

BEST PRACTICES

Maximizing Performance

Performance is a critical criteria in evaluating software. The highest quality data presented using the most insightful visualizations will not be appreciated by end users if performance is frustratingly poor.

It's easy for developers to overlook performance for several reasons:

- Overly focused on functionality
- Work with small data sets
- Work exclusively on high performance desktops with high speed data connections forgetting that many SHYFT users will be on iPads with slower connections
- Not consulted during design and only have input after client sign off
- Fail to test on tablets
- Over accommodate during the development phase in an effort to please the business analyst/client. It's acceptable to advise against a design, however developers are obligated to provide a valid business/database/performance reason for advising against a particular requested feature.

Items that can negatively affect performance include:

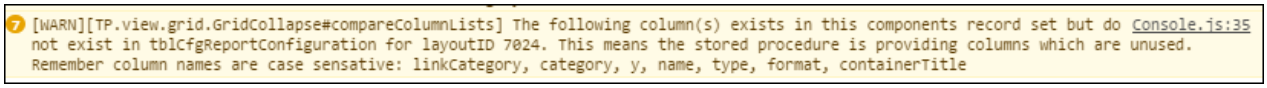
- The number of Categories in a Section. Remember, that all Categories for a single Section are loaded at one time.
- The number Components in a Category and the number of Components in a Container.
- The complexity of Components. For example, it takes a lot of latitude and longitude data to draw Heat Maps.
- The amount of data in tables. Tables with large numbers of rows or columns may need to be paging tables. There is no hard number of rows that determine if a table should be pageable, since the total amount of data, (affected by the number of columns and the amount of data in the columns), determines performance.
- The number of Filters will affect the performance of SQL Where clauses.

Ways to improve performance during design:

- Rarely should there be more than four Categories per Section.
- Keep the number and complexity of Components reasonable across the entire Section. For example, do not put more than several Components into a single Container. Do not overuse tab panels.
- Keep the number of Filters reasonable.
- Insist on a design review prior to client presentation and sign off.

Ways to improve performance during development:

- Consider if it is better to pre-aggregate or aggregate at run time. For details, click here.
- Consider using execution plans and indexes. For details, click here.
- Consider using tblRptFilterCombo to optimize Filters. For details, click here.
- Monitor network performance through the Chrome Developer Tools Network panel. For details, click here.
- When populating grids, supply only the required columns that are actually defined and visible to the end user. Note that the browser console details when a column is unused. For example, the following message indicates that LayoutID 7024 is supplying several unused columns, making the data download time and storage both larger than necessary:



- In the table thlFilterValue, populate the Value column with integer code values, not text.
- In setting the PageSzie attribute for pageable tables, experiment to find the right balance between performance and paging that occurs too frequently.
- Routinely perform iPad testing during development, not as an afterthought only late in the process when it is too late to change.

Preventing SQL Injection with Text Filters and ListView Section Carousels

SQL injection is a hacking technique that allows inappropriate and possibly malicious SQL commands to be issued against a database by non-authorized users. Such commands could, for example, delete tables.

V2 text filters and the ListView Section Carousel, which allow users to enter free text, are at risk for SQL injection. For example, if a text filter prompts for Physician name, rather than enter a name, a v2 user with sufficient knowledge, could enter a SQL command that performs malicious activity.

Only v2 activity where free text is entered open the possibility for SQL injection and there is a technique for preventing such an attempt from doing harm.

When text filters are used note that v2 supplies the value to the associated stored procedure in the @vchFilterString variable and that this is true for both paging and non-paging stored procedures. So if a user enters the Physician name as 'Smith', then @ vchFilterString will have the value 'WHERE Physician LIKE('%Smith%'). To process @vchFilterString, dynamic SQL must be used.

There are two SQL commands that can be used to actually execute dynamic SQL. EXECUTE and sp_executesql. The EXECUTE command exposes SQL injection. Sp_executesql does not and should be used whenever a text filter are in use.

Using sp_executesql might best be explained through an example which can be seen in all auto generated paging stored procedures.

```
SET @sqlParams = N'@intPageSize INT, @intPageNumber INT, @showFavorites BIT';

EXEC sp_executesql  @sqlString, @sqlParams
                    ,@intPageSize = @intPageSize
                    ,@intPageNumber = @intPageNumber
                    ,@showFavorites = @showFavorites;
```

Note:

- @sqlString will contain a built-up SQL command that would have used the stored procedure input parameter @ vchFilterString
- @sqlParams contains a list of parameters and their associated data type
- The parameters to EXEC sp_executesql are @sqlString, @sqlParams followed by each of the parameters in @sqlParams and their assigned values

The result of this syntax is that SQL injection attempts will fail.

Pre-Aggregation & Aggregation At Run-Time

Generally, best practice for query optimization in v2 is to **pre-aggregate** data within a reporting table. However, there may be times when **aggregation at run-time** may have be advantageous. Developers should minimize the time it takes to return a result set to the end-user using whichever method or combination of methods is appropriate. To provide guidance on this topic, this section will discuss the current best practice point of view, advantages and disadvantages of each, and what to do when a developer needs to resort to aggregation at run-time.

Pre-aggregation is the preferred method of designing reporting tables. This method relies on creating the result set for every filter combination that the end-user may choose. All of the calculations have been done and SQL need only return the result set. This method returns result sets very quickly and can take advantage of other best practices such as using integer identifiers instead of varchar names in the where clause or in the join conditions, using combination filters to identify a filter combination in a single integer column, and indexing a table upon key column(s). In this method, the business team can easily QA all possible combinations that the end-user might see. This method requires that all filter possibilities be predefined. Pre-aggregation should be used if the filter types are limited to: dropdown, single, and, in some cases, multi. These filter types have a discrete and limited number of permutations, which makes creating a full list of permutations in a single table feasible. Exceptions can be made in extreme cases of very large number of possible combinations, but should be discussed before implementing.

Notes on combination calculations:

- When multi filter types are used the potential combinations increase exponentially

  Example: For 3 non-cascading fixed filters: 1 drop down filter with 10 choices, 1 single filter with 3 choices, and 1 multi filter with 3 choices there are $10 * 3 * (2^3 - 1) = 210$ combinations, but this increases exponentially when the multi filter choices is raised: 4 choices, $10 * 3 * (2^4 - 1) = 450$ combinations; 5 choices, $10 * 3 * (2^5 - 1) = 930$ combinations; etc.
- To determine number of filters combinations on non-cascading fixed filters, multiply the possible states of each filter by the possible states of every other filter: The number of states for drop down and single filter are the number of values that are in the filter value table for that filter. The number of states for a multi filter is (2n-1) where n is the number of values in the filter value table for that filter (minus 1 because there is no empty set option). The calculation is more complicated, but not necessarily greater for scenarios including cascading filters. Non-fixed filters always create more combinations because it is the number of combinations for each filter plus the additional combinations when some filters are not used.
- Text and range filter types are type-in fields for the end user, so they could put in any value. This makes the number of theoretical combinations infinite.

Aggregation at run-time should only be used only when necessary. It is usually outperformed by pre-aggregation. The only time this should be used is when the number of combinations creates a table that is so large that searching the table takes more computation time than doing the calculations on a smaller set at run-time or the full set of combinations is not attainable because of filter type (text filters or range filters).

- Example: Consider a view with 2 multi type filters each with 10 choices and a territory filter with 25 territories. This results in $25*(2^{10} - 1)*(2^{10} - 1) =$

26,163,225 combinations. In this scenario, it may be more efficient to do some pre-aggregation and then aggregation at run-time (making sure to only use simple, built-in calculations i.e. SUM(), MAX(), etc.)

- It may be faster to create an independent record for each value in each multi filter for each territory (10*10*25 = 2,500 combinations) and aggregate at run time on 2,500 rows for whatever combination the end-user chooses.

Additionally, avoid complex or multi-step calculations at run-time. When the user is allowed to enter any filter value and it is necessary to return an aggregation, then aggregation is acceptable at run-time.

Overall, it is important to aggregate as much as possible, if not entirely, in reporting tables. For the few exceptions that do exist, tables must be well designed and well tested. As always, if there are concerns about the feasibility of requirements or usability of the end-product bring them to your manager's attention as soon as possible. Let the business team know when you are aggregating at run time, so they can QA appropriately. Further research should be done on what thresholds need to be met before aggregation at run time is acceptable. For now, it should only be done when pre-aggregation is too slow or impossible.

Logic for returning filtered data based on cascading Sales Groups in Hierarchical format

Filters based on sales group levels is commonly required in mobile reports. There are two needs here- the filters to cascade correctly based on which sales group is selected, and for the data to be correctly filtered based on the filter selection. For data that is assigned to a specific entity (prescriber or account) the data is often in a table at the territory level. Depending on filters selected, a user may want to see all the data from one territory, or many territories that belong to multiple regions or areas. For example, a "West" area rep will want to see all data in the territories that belong to the West area. Therefore, a system needs to be in place with the filters and data to select all the data in that sales group, and be dynamic enough to change at run time based on a user's filter selections.

The process for achieving this is done by setting four sales group levels Level1= territory, Level2=region, Level3=area, Level4=nation (assuming there is only four levels in the hierarchy). The trick here is to use negative sales group codes to roll up to the next level sales group. For example Level1 through Level4 is positive whenever a user is selected on 1 distinct territory. If a user wants to see all territories rolled up to a region, the Level1 sales group code will be a negative of the Level2 sales group code, and Level2 through Level4 will be positive. This pattern continues for area and nation "All" selections.

The two following images show an example of using the where clause to filter the data. It is written so that if a negative sales group code is seen then all the data remaining in the table is returned, otherwise the data is filtered down to that level. The variables will be defined from the filter values selected by the user, and the where clause will be used for the component in which the user wants to see data filtered based on their selections.

```
--Territory Level (NORTH VALLEY)
DECLARE @intLevel3SalesGroupID INT = 1532
DECLARE @intLevel2SalesGroupID INT = 1542
DECLARE @intLevel1SalesGroupID INT = 1543
        SELECT distinct
           Level4SalesGroupID,
           Level3SalesGroupID,
           Level2SalesGroupID,
           Level1SalesGroupID,
           SG.SalesGroupTypeName,
           SG.SalesGroupName
           FROM
            tblRptDailyMedical_OrdersList main
            INNER JOIN  dbo.tblSalesGroup SG    ON
                    main.Level1SalesGroupId = SG.SalesGroupID
       WHERE
              1 = CASE  WHEN Level4SalesGroupID = @intLevel3SalesGroupID THEN 1
                        WHEN Level3SalesGroupID = @intLevel3SalesGroupID THEN 1
                   END
          AND 1 = CASE  WHEN @intLevel2SalesGroupID < 0                  THEN 1
                        WHEN Level2SalesGroupID = @intLevel2SalesGroupID THEN 1
                   END
          AND 1 = CASE  WHEN @intLevel1SalesGroupID < 0                  THEN 1
                        WHEN Level1SalesGroupID = @intLevel1SalesGroupID THEN 1
                   END
```

Results | Messages

Level4SalesGroupID	Level3SalesGroupID	Level2SalesGroupID	Level1SalesGroupID	SalesGroupTypeName	SalesGroupName
1530	1532	1542	1543	TERRITORY	NORTH VALLEY, CA

```
-- (-) filter value when 'ALL' is selected
--NORTHERN, CA region, ALL Territories selected
DECLARE @intLevel3SalesGroupID INT = 1532         --area
DECLARE @intLevel2SalesGroupID INT = 1542         --region
DECLARE @intLevel1SalesGroupID INT = -1542           --terr
        SELECT distinct
           Level4SalesGroupID,
           Level3SalesGroupID,
           Level2SalesGroupID,
           Level1SalesGroupID,
           SG.SalesGroupTypeName,
           SG.SalesGroupName
           FROM
            tblRptDailyMedical_OrdersList main
            INNER JOIN  dbo.tblSalesGroup SG    ON
                    main.Level1SalesGroupId = SG.SalesGroupID
       WHERE
              1 = CASE  WHEN Level4SalesGroupID = @intLevel3SalesGroupID THEN 1
                        WHEN Level3SalesGroupID = @intLevel3SalesGroupID THEN 1
                   END
          AND 1 = CASE  WHEN @intLevel2SalesGroupID < 0                  THEN 1
                        WHEN Level2SalesGroupID = @intLevel2SalesGroupID THEN 1
                   END
          AND 1 = CASE  WHEN @intLevel1SalesGroupID < 0                  THEN 1
                        WHEN Level1SalesGroupID = @intLevel1SalesGroupID THEN 1
                   END
```

Results | Messages

Level4SalesGroupID	Level3SalesGroupID	Level2SalesGroupID	Level1SalesGroupID	SalesGroupTypeName	SalesGroupName
1530	1532	1542	1543	TERRITORY	NORTH VALLEY, CA
1530	1532	1542	1544	TERRITORY	MARIN COUNTY, CA
1530	1532	1542	1545	TERRITORY	WALNUT CREEK, CA
1530	1532	1542	1546	TERRITORY	SACRAMENTO, CA
1530	1532	1542	1547	TERRITORY	ROSEVILLE, CA
1530	1532	1542	1548	TERRITORY	RENO, NV
1530	1532	1542	1549	TERRITORY	MODESTO, CA
1530	1532	1542	1550	TERRITORY	FRESNO, CA

Optimizing Front End performance through Execution Plans and Indexes

The majority of standard stored procedures that are written for V2 can perform quickly and efficiently without the need for additional performance tuning. These procedures read from pre-computed reporting tables that require little to no data manipulation from the V2 developer to display the data on the front end.

However, there are a few types of tables that have a larger than normal amount of rows which can add a seconds to the load times of the front end. A few of the charts that typically take longer to load are Journey Trees, Entity Segmentations (Prescribers, Account etc.), Prescriber List/Lookup, and Nearby Prescriber. These charts typically have tables with row counts that can get into the millions depending. The easiest way to optimize these charts is to add an index to the reporting table. A good way to determine if your query needs an index, and what the index should be, is to utilize query execution plans.

A query execution plan is the set of steps that SQL will perform while running your queries. At its core a query plan shows you what method SQL will use to access the data (such as a table scan, index scan, or index seek) and the time cost that step has relative to the rest of the steps in the process. Query plans can also be useful for finding out more detailed information for each step such as the number of rows scanned, I/O and CPU cost. For most V2 developers, you will only need to use a query execution plan for determining if your queries could use an index for a performance boost.

MS SQL Server comes with 2 built in tools to display query execution plans:

- Estimated Execution Plan
- Actual Execution Plan

Estimated Execution Plan – Displays the "estimated" query plan for a set of code before you run the actual query.

Pros:

- Does not require running the full set of code to get a query plan
- Easier to run individual sets of code and look at query plans one by one

Cons:

- Selected code needs to be "runnable". If code has variables, the variables need to be declared and populated or replaced with hardcoded values, otherwise there will be a syntax error
- Query Plan is only an estimate, it could differ from the actual plan

Actual Execution Plan – After running a set of code, or stored procedure, the actual query plan that was used is included in the results pane.

Pros:

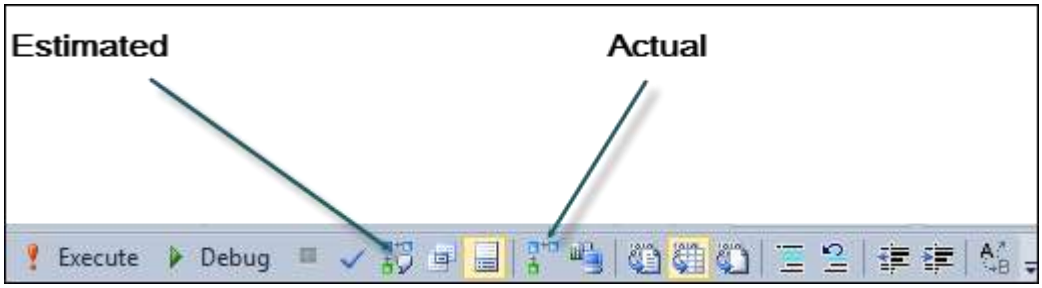
- Shows the actual plan that was used which can differ from estimation
- Easier to run complicated sets of queries because you can run an entire stored procedure and look at the entire plan afterwards

Cons:

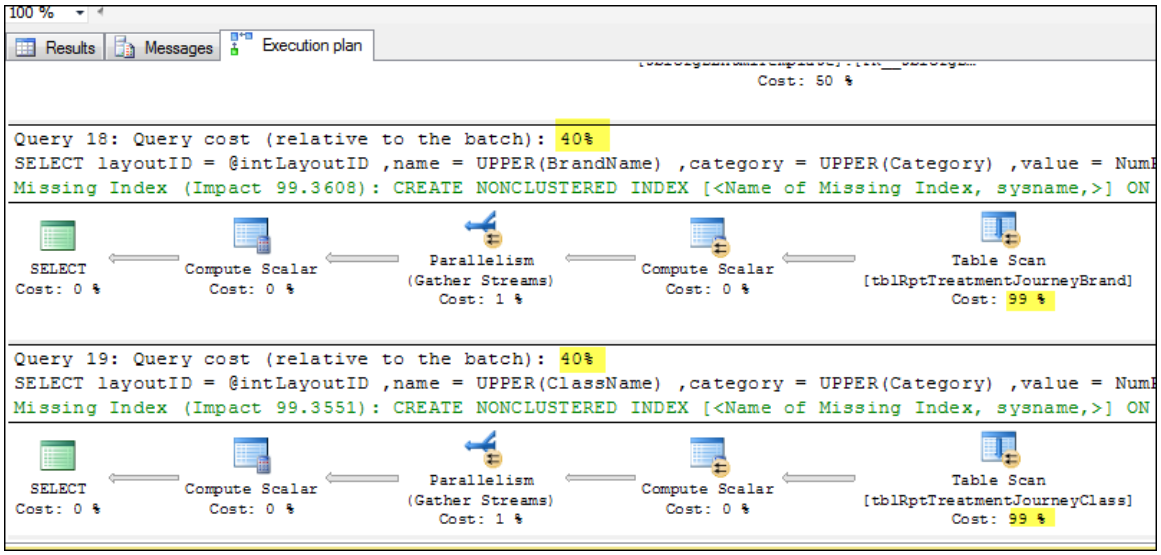
- If used to run with an entire stored procedure, you can get dozens of query plans in the batch which makes it harder to read the plans you care about.
- Requires waiting for SQL query to complete which can take longer if it is a time consuming process

How to use Execution Plans:

1. Make sure the SQL Editor toolbar is enabled (View → Toolbars → SQL Editor)

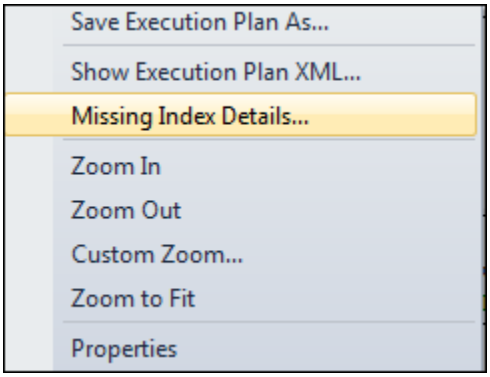


2. For Actual query plan, click the Actual toolbar button. When you execute your code the execution plan will be displayed afterwards. For Estimated query plan, highlight the desired code and click the Estimate toolbar button. The query plan will be displayed if your code has no syntax errors.



Finding Missing Indexes - The above image shows an example of an execution plan for an entire reporting stored procedure that populates a Journey Tree chart. Looking at either query plan, you will notice there is a Table Scan on tblRptTreatmentJourney that is costing the query 99% of its time. Additionally you will notice that these queries are numbered 18 and 19. That means these were the 18th and 19th query that needed to be executed in the stored procedure. They both have a cost of 40% which means they take up 80% of the entire stored procedure process.

SQL has determined that the query can be optimized if an index is added to the reporting tables. In the green font, SQL has provide a "Missing Index" message which means SQL has determined that the query can be optimized by adding a specific index, in this case the missing index is causing a 99% impact to query. To open the missing index script, right click the message and select "Missing Index Details…" this will open up a new query window with the index creation script.



How to add an index to your reporting table:

It is VERY important to note that you should not blindly add a suggested index to your table. If you want to add the index to test performance make sure you drop the index after you are done testing.

The database developer will need to add the index creation to their data run process. The database developer will need to drop and recreate the index every time they populate the table. Adding an index to the table without letting the DB developer drop and recreate the index in their data run can slow down the process of populating the reporting table. If the index is not dropped SQL will constantly be updating the index when a table is populated. Every time a new row is inserted the entire index needs to be updated. It is faster for SQL to populate a table and then add an index to the table afterwards, than it is for SQL to populate the table and index row by row.

Please follow these steps for adding an Index to your reporting table:

- Open the missing index script
- Copy the index script code
- Email (or create a Jira Ticket) the index to your DB developer asking for them to add this index to your reporting table.

Optimizing Filters through tblRptFilterCombo

End user Category Filters provide critical business value to Analytic Platform reports. Filters often become numerous and complex enough that considerations should be given to their development with the goals of:

- optimizing end user performance
- standardizing initial development
- easing ongoing maintenance

This section discusses how these development goals can be met through the use of the SQL table tblRptFilterCombo.

Background

Typical Reporting Table. Reporting tables generally contain both data and identifying information about that data. For example, in the following table there is a single data point (Metric) and six columns that identify that data (SalesGroupID through ProductName):

SalesGroupID	SalesGroupName	TimePeriodID	Penname	ProductID	ProductName	Metric
1	Boston	1	MTD	1	Product1	200
1	Boston	1	MTD	2	Product2	360
…		…		…		
50	San Fran	3	LTD	4	Product4	15000

A SQL SELECT statement for this table would be:

```
SELECT Metric FROM tblSalesInfo WHERE SalesGroupID=1 AND TimePeriodID=1 and ProductID=1
```

tblReportFilterCombo. tblReportFilterCombo is designed to assign a unique Id to each possible combination of all filters. For example, in a reporting system that supports 50 sales groups, 3 time periods, and 4 products, tblReportFilterCombo would have 600 rows (50 * 3 * 4) and would look like:

FilterComboID	SalesGroupID	TimePeriodID	ProductID
1	1	1	1
2	1	1	2
…	…	…	…
600	50	3	4

Note that every permutation of SalesGroupID, TimePeriodID and ProductID is represented and assigned a FilterComboID.

Reporting Table with FilterComboID. Using tblRptFilterCombo, the reporting table now looks like:

FilterComboID	Metric
1	200
2	360
...	
600	15000

And the SQL SELECT statement would now be:

```
SELECT Metric FROM tblSalesInfo WHERE FilterComboID = 1
```

Development

Building tblRptFilterCombo. Given SalesGroup, TimePeriod and Product tables that look like:

SalesGroupID	SalesGroupName
1	Boston
2	New York
...	

TimePeriodID	TimePeriodName
1	MTD
2	QTD
3	LTD

ProductID	ProductName
1	Product1
2	Product2
3	Product3

tblRptFilterCombo can be built with the following:

```
INSERT INTO tblRptFilterCombo
SELECT a.SalesGroupID, a.SalesgroupName, b.TimePeriodID, b.TPName, c.ProductID,
c.ProductName
 FROM tblstgSalesGroup a
        CROSS JOIN tblStgTimePeriod b
        CROSS JOIN tblStgProduct c
```

Note that filterComboID should be defined as an INT IDENTITY column. Reporting tables can now use FilterComboID.

Lookup function. Create a lookup function that, given each filter value, returns filterComboID. For example:

```
DECLARE @FilterComboID INT
SELECT @FilterComboID = FilterComboID
FROM tblRptFilterCombo
WHERE
    SalesGroupID        = @SalesGroupID
AND TimePeriodID        = @TimePeriodID
AND ProductID           = @ProductID
RETURN @FilterComboID
```

Converting Filter Values to @intFilterComboID. A typical Category stored procedure would now look like the following:

```
--Filter IDs declared here
DECLARE @SalesgroupID = (SELECT dbo.GetFilterValue(@tblFilterNameValue,
SalesGroupID');
DECLARE @ProductID = (SELECT dbo.GetFilterValue(@tblFilterNameValue, ProductID');
DECLARE @TimePeriodID = (SELECT dbo.GetFilterValue(@tblFilterNameValue,
TimePeriodID');

--Single FilterComboID is declared here
DECLARE @intFilterComboID = (SELECT dbo.udfGetFilterComboID(@SalesgroupID,
@ProductID, @TimePeriodID ));
```

Using @intFilterComboID. A typical SELECT statement would look like:

```
SELECT
 layoutID = 100
 ,type          = 'column'
 ,category      = 'percent'
 ,name   = CategoryName
 ,band1Color = #F00000
 ,y             = metric
 ,metric1Format = 'percent'
 ,metric1Label  = 'Insert Label Here'
FROM [Insert reporting table here] (Could also be your CTE/tmptable)
WHERE FilterComboId = @intFilterComboID
```

Additional Notes

Additional considerations:

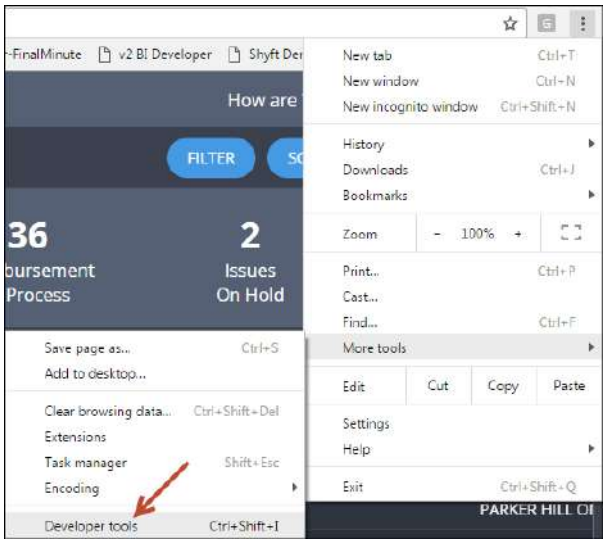
- For convenience, for example when performing QA, it may be helpful to create a View using both the reporting table and tblRptFilterCombo to combine the data and data description columns into easy to read rows.
- tblRptFilterCombo is not practical for text filters that support user entered free text.
- Instances where tblRptFilterCombo may be of less value:
    - If a report is so simple and small that tblRptFilterCombo is not worth the additional effort.
    - In reports where Filters vary widely between Categories.

Monitor performance through the Chrome Network Panel

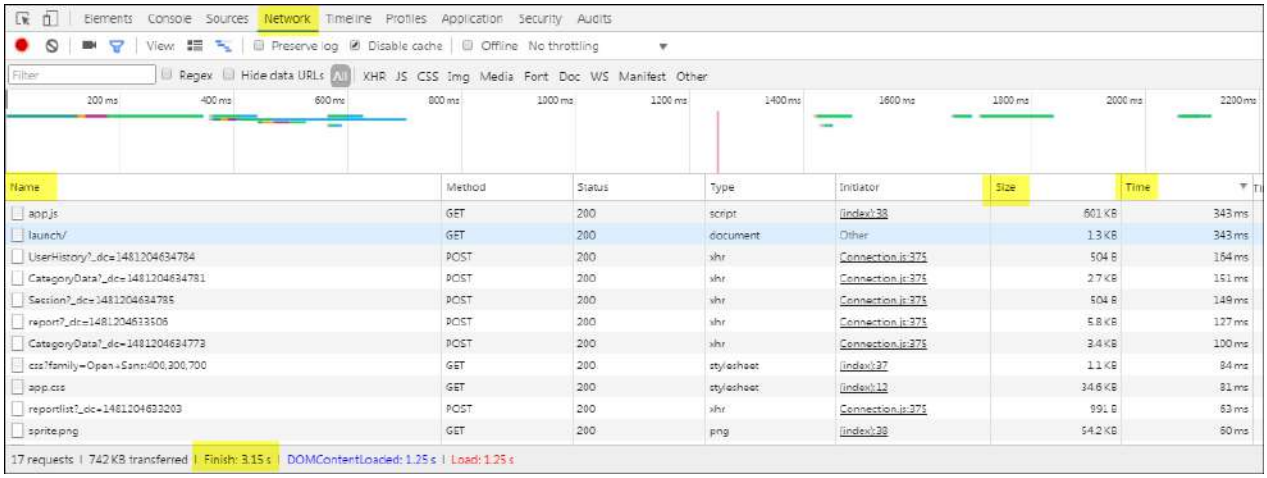
The Google Chrome Developer Tools Network panel provides a wide range of information regarding Analytic Platform network traffic.

Two primary data points provided are the time required to execute Category and Component stored procedures and the amount of data provided. Monitoring this information may identify stored procedures whose performance or volume of data are negatively affecting performance with the hope that some optimization may be possible.

Access the Chrome developer tools as follows:



Then access the Network panel:

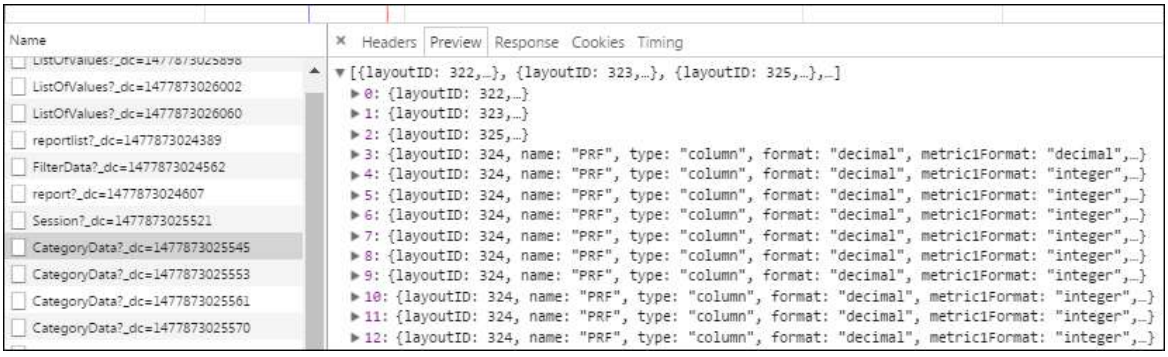


Primary columns of interest are:

- **Name**: CategoryData will be the execution of the stored procedure associated with a Category. ComponentData will be the execution of the stored procedure associated with a pageable grid.
- **Size**: Size indicates the number of bytes returned by the stored procedure. Click the column header to sort by Size.
- **Time**: Time indicates the number of milli-seconds from initial request until the full data set has been received. Click the column header to sort by Time.
- **Finish**: The length of time the request for a Section to load and all Components in the Section being fully rendered.

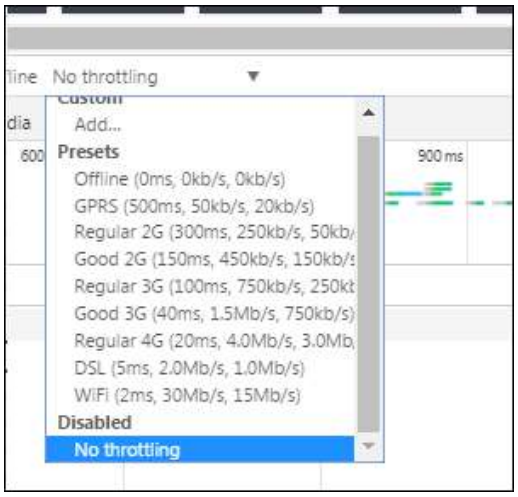
Regarding CategoryData:

- Click a CategoryData name and the Preview tab, to observe its returned data.

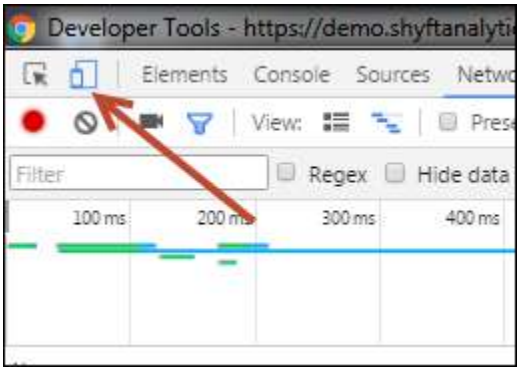


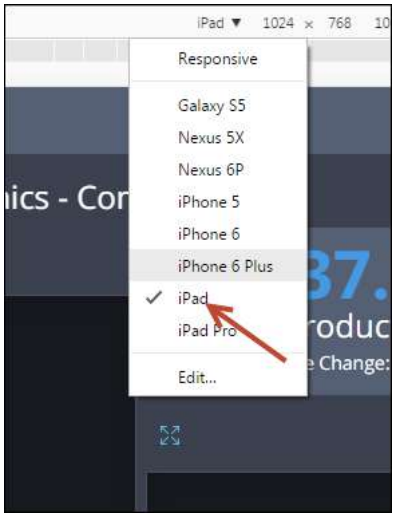
- Click a CategoryData name and the Timing tab, to observe its timings. Waiting (TTFB = time (un)til first byte), is the execution time of the associated stored procedure.

- Test performance having software throttled your network connection speed:



- Simulate the iPad display size:





Following is a comprehensive, easy to read guide, to using the Network panel of the Chrome developers tools: https://developers.google.com/web/tools/chrome-devtools/network-performance/resource-loading

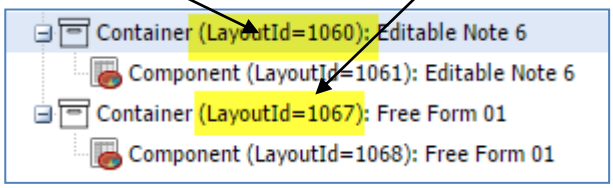
TIPS & TRICKS

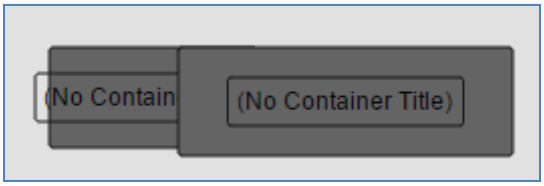
Using an editableNote as a Container Title

Since the system always places Container Titles above and to the left of the Container such Title placement may not be optimal. By setting the attributes BackgroundCls = 'charttabpanel-title' and WritePrivildge to empty, an editableNote can provide more flexibility in providing a Container Title.

An example of using an editableNote as a Container Title can be found here.

However, be aware of a quirk in using an editableNote as a Title. Be sure to have the LayoutID value of the editableNote be less than the LayoutID value of the Container it is to label. This will allow the editableNote to be much closer physically on the screen. If the LayoutID value of the Container is smaller, the system will create a larger border resulting in the editableNote likely being too far from the Container it is to label.





Now the Title can "overlap" the Container it is to label.

Tying Grid column titles to Filter changes

Dynamic Labels provide a way for the column headers of a Grid to change from one report to another. A developer might wish to have these column headers change with the selection of a filter, requiring them to become more agile. While we currently do not have the capability to make such changes truly 'on the fly', with a little bit of thought and ingenuity the developer can pre-configure certain labels that they would like to have change.

This can be achieved by creating a separate 'labels' table that will account for all of the desired columns headers that, like true Dynamic Labels, are updated from report to report, but can be changed as a user filters the report. This method has been put into practice by overlaying a FreeForm in place of the column header of a grid. The actual column headers will have blank Title attributes (be cognizant when using the 'expandable' attribute of a container). The FreeForm will pull columns from the new 'label' table with proper filters applied to display the desired values.

Below is an example where we would like to change the 12 month column headers of a grid for Fiscal YTD and Rolling Year:

Fiscal YTD:

	Apr-14	May-14	Jun-14	Jul-14	Aug-14	Sep-14	Oct-14	Nov-14	Dec-14	Jan-15	Feb-15	Mar-15
Source1	3	4	4	5	5	6	6	5	4	5	5	5
Source2	5	5	5	5	5	5	8	8	9	3	3	3
Source3	3	4	5	6	6	6	6	6	6	5	5	5

Rolling Year:

	Jan-14	Feb-14	Mar-14	Apr-14	May-14	Jun-14	Jul-14	Aug-14	Sep-14	Oct-14	Nov-14	Dec-14
Source1	3	4	4	5	5	6	6	5	4	5	5	5
Source2	5	5	5	5	5	5	8	8	9	3	3	3
Source3	3	4	5	6	6	6	6	6	6	5	5	5

tblDateLabel:

	TimePeriod	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
1	Rolling Year	Apr-14	May-14	Jun-14	Jul-14	Aug-14	Sep-14	Oct-14	Nov-14	Dec-14	Jan-15	Feb-15	Mar-15
2	Fiscal YTD	Jan-14	Feb-14	Mar-14	Apr-14	May-14	Jun-14	Jul-14	Aug-14	Sep-14	Oct-14	Nov-14	Dec-14

Category Dots

These dots provide the user an intuitive method of orientating themselves within a section of a report. Type in the Unicode text for a dot or fisheye in the title attribute of a category.

&#x25C9 = ◉

&#x25CF = ●

Attribute List - Category (LayoutId=3):

Attribute	Value
DisplayOrder	1
StoredProcedure	rsp_MSL_SEC1_CAT1
Title	◉●●●●
Type	categorypanel

1 | 2 | 3 | 4 | 5 = 1 2 3 4 5

&#x25C9&#x25CF&#x25CF&#x25CF&#x25CF = ◉●●●●

&#x25CF&#x25C9&#x25CF&#x25CF&#x25CF = ●◉●●●

Etc.

@@ROWCOUNT

Summary:
@@ROWCOUNT is an important variable in SQL. It returns the "rowcount" of the previously executed statements result set. This is useful when combined with an IF clause. The next select statement can be anything SQL supports. In the example below, if the filters return no rows, then a separate select will return the possible categories and all values set to NULL.

Business Uses:

1. Stops the endless loading when the query returns no rows. (Keeps the front end looking good and useful for bug fixing)
2. This is useful for data sets with many possible mathematical combinations, but fewer real world combinations as the DB does not have to manufacture rows with zeros. For example, in a world with 2 countries and a set of products only for that country:

Geography	Product	Possible Combinations	Real world Combinations
A	X,Y,Z	(A,X) (A,Y) (A,Z) (A,P) (A,Q)	(A,X) (A,Y) (A,Z)
B	P,Q	(B,X) (B,Y) (B,Z) (B,P) (B,Q)	(B,P) (B,Q)

```
        SELECT      layoutID        = 177,
                    name            = Type,
                    type            = 'column',
                    format          = 'decimal',
                    category        = CategoryType,
                    --categoryorder = lbl.DynamicLabelID,
                    y               = Value
        --SELECT *
        FROM        tblRptActivityOverview
        WHERE       Time            = 'Month'
        AND         MyTargets       = @vchMyTargets
        AND         GeographyFilter = @vchGeography
        AND         Product         = @vchProduct
        AND         Behavior        = @vchBehavior
        AND         EconomicValue   = @vchEconomicValue
        AND         AdoptionStatus  = @vchAdoptionStatus
        ORDER BY    name ASC

IF @@ROWCOUNT = 0
        SELECT      layoutID        = 177,
                    name            = NULL,
                    type            = 'column',
                    format          = 'decimal',
                    category        = CategoryType,
                    --categoryorder = lbl.DynamicLabelID,
                    y               = NULL
        --SELECT *
        FROM        tblRptActivityOverview
        WHERE       Time            = 'Month'
        GROUP BY    CategoryType
```

Control Default Filter Order

Problem:

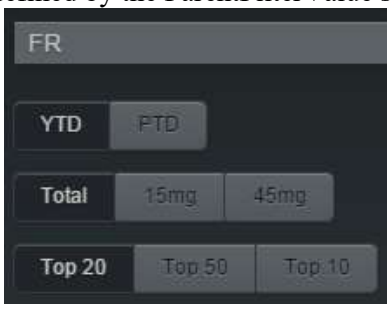
There is a need for the default order of dynamic filters to change depending on the value of another filter.

Solution:

The order of the filter values in a filter is based on the order they are inserted into tblFilterValue. By using the cascading filter functionality, the filter values can be put in a different order by ParentFilterValue

Resolution:

The filter order is defined by the ParentFilterValue selected in the parent filter.




```
DELETE FROM tblFilterValue WHERE SPName = 'rspFSB_AccountSegmentation_TopAccountsFilter'

INSERT INTO tblFilterValue(SPName, Name, Value, ParentValue)
SELECT
    SPName
    ,Name
    ,Value
    ,ParentValue
FROM
    (

-- Cascading in order to make default 10 for Europe, 20 for countries and 50 for KAMs

    SELECT
        spname = 'rspFSB_AccountSegmentation_TopAccountsFilter'
        ,Name = 'Top 50'
        ,Value = 'IsTop50'
        ,ParentValue = SalesGroupCode
        ,sortorder = 1
    FROM tblSalesGroup
    WHERE SalesGroupTypeID NOT IN (2,3)
    UNION ALL
    SELECT
        spname = 'rspFSB_AccountSegmentation_TopAccountsFilter'
        ,Name = 'Top 20'
        ,Value = 'IsTop20'
        ,ParentValue = SalesGroupCode
        ,sortorder = 2
    FROM tblSalesGroup
    WHERE SalesGroupTypeID NOT IN (2,3)
    UNION ALL
    SELECT
        spname = 'rspFSB_AccountSegmentation_TopAccountsFilter'
        ,Name = 'Top 10'
        ,Value = 'IsTop10'
        ,ParentValue = SalesGroupCode
        ,sortorder = 3
    FROM tblSalesGroup
    WHERE SalesGroupTypeID NOT IN (2,3)
```

```
UNION ALL
    SELECT
        spname = 'rspFSB_AccountSegmentation_TopAccountsFilter'
        ,Name = 'Top 20'
        ,Value = 'IsTop20'
        ,ParentValue = SalesGroupCode
        ,sortorder = 4
    FROM tblSalesGroup
    WHERE SalesGroupTypeID NOT IN (1,3)
    UNION ALL
    SELECT
        spname = 'rspFSB_AccountSegmentation_TopAccountsFilter'
        ,Name = 'Top 50'
        ,Value = 'IsTop50'
        ,ParentValue = SalesGroupCode
        ,sortorder = 5
    FROM tblSalesGroup
    WHERE SalesGroupTypeID NOT IN (1,3)
    UNION ALL
    SELECT
        spname = 'rspFSB_AccountSegmentation_TopAccountsFilter'
        ,Name = 'Top 10'
        ,Value = 'IsTop10'
        ,ParentValue = SalesGroupCode
        ,sortorder = 6
    FROM tblSalesGroup
    WHERE SalesGroupTypeID NOT IN (1,3)
```

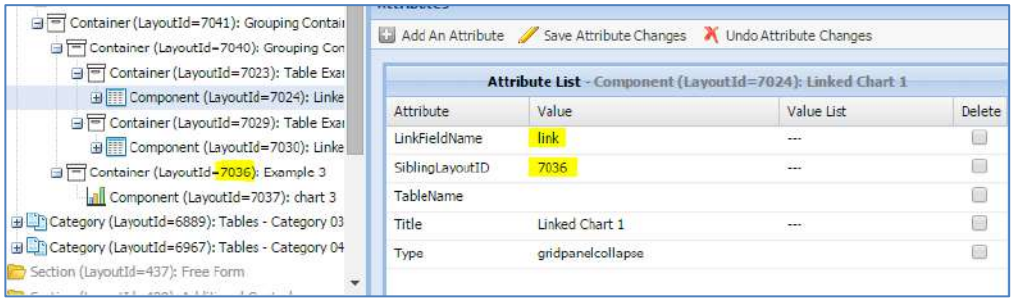
```
UNION ALL
    SELECT
        spname = 'rspFSB_AccountSegmentation_TopAccountsFilter'
        ,Name = 'Top 10'
        ,Value = 'IsTop10'
        ,ParentValue = SalesGroupCode
        ,sortorder = 7
    FROM tblSalesGroup
    WHERE SalesGroupTypeID NOT IN (1,2)
    UNION ALL
    SELECT
        spname = 'rspFSB_AccountSegmentation_TopAccountsFilter'
        ,Name = 'Top 20'
        ,Value = 'IsTop20'
        ,ParentValue = SalesGroupCode
        ,sortorder = 8
    FROM tblSalesGroup
    WHERE SalesGroupTypeID NOT IN (1,2)
    UNION ALL
    SELECT
        spname = 'rspFSB_AccountSegmentation_TopAccountsFilter'
        ,Name = 'Top 50'
        ,Value = 'IsTop50'
        ,ParentValue = SalesGroupCode
        ,sortorder = 9
    FROM tblSalesGroup
    WHERE SalesGroupTypeID NOT IN (1,2)

    ) t
    ORDER BY sortorder ASC
```

Linking: Grids & Charts

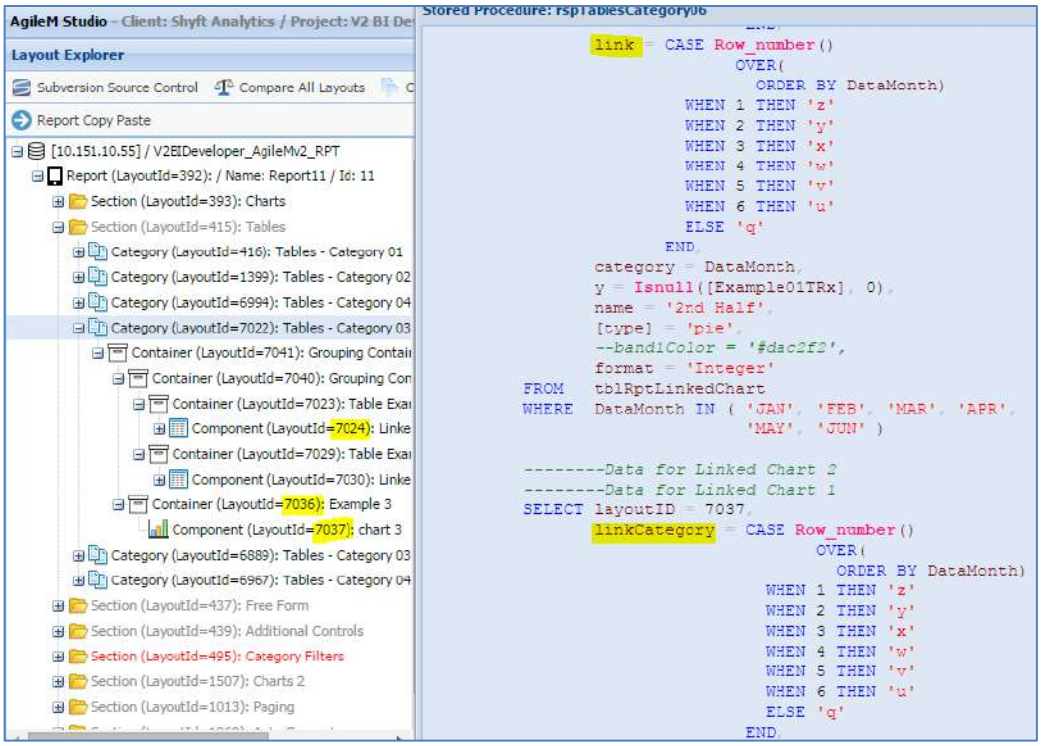
Linking is a great way to provide drill-down/detailed analysis in a report. They work as a filter only returning the result set from the row in a grid that the user clicks on. Below is a tutorial on how to build links and known limitations.

How To:
Linking requires a minimum of two objects in different containers. One that is an independent grid, which will work as the filter selection and one or more that will work as the dependent object. The dependent object may be a grid or a chart.

The data that needs to be provided to the app for this functionality to work is an attribute in the component level of the independent grid called “SiblingLayoutId” this attribute must contain a comma separated list of component containers that will be affected by the filter. The second attribute is called “LinkFieldName.” This attribute contains the field in the grid that will filter the dependent grids or charts.



The LinkFieldName, in this case a field called “link” should have a specific result set for a given filter combination to the table for the dependent grid or chart. The field “link” in the independent table must be equal to the field in the dependent table(s) and must be aliased “linkCategory.”



Known Advanced Functionalities:

- This feature allows linking from one independent to one dependent, one independent to many dependent, many independent to one dependent, and many independent to many dependent.
- This feature allows “chaining” the links, so one independent may affect a dependent, which then affect another dependent.

Known Limitations:

- This feature can only be used with category level procedures. So natural sorting, column hiding, ranking, and all features that need to be in a component level stored procedure are not supported.
- There must be a container separate from the independent grid for the dependent component to point to.
- Note that the entire result set of the dependent component is pulled into the application at once and then filtered upon a row-click.

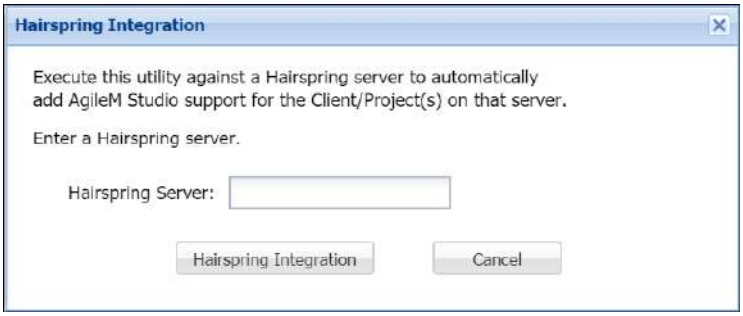
MISC

Hairspring Integration

Clients and Projects created through Hairspring may be easily added to the initial AgileM Studio prompt through the use of the Hairspring Integration button.

That is, having created an Analytics Platform project using the Hairspring tool the new Client/Project will not initially appear in the AgileM Studio start up prompt. To add the Client/Project select the button Hairspring Integration.

The following prompt will appear. Enter the server where the Hairspring project was created. When the Hairspring Integration button is clicked, the system will perform all necessary steps to bring new Client/Projects into AgileM Studio.

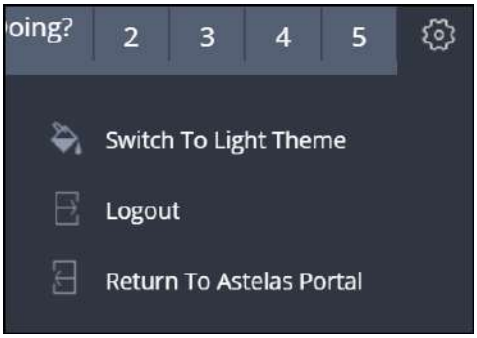


Note that this selection is "re-runable". That is, if this functionality was initially executed after Projects 1, 2 and 3 were created and then Project 4 is added later, there is no harm in running this functionality a second time.

Logout and Return to Portal

tblAppConfig, as detailed here, supports three properties that provide flexibility in exiting Analytics Platform.

Note that the Sprocket menu always offers the Logout selection. A second optional selection, whose text is configurable, is designed to redirect to a company portal. Following is an example:



Property 'LogoutURL'

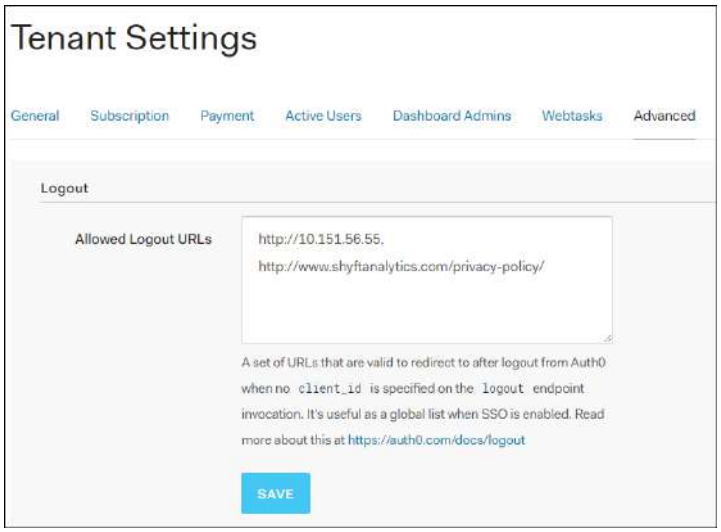
An optional property, that when present, will control where AP redirects to. While the Logout option will always appear, this optional parameter will control where the system will redirect to.

If the property is present, control will be directed to that URL. If the property is not present than control will be directed back to the AP logon point (which will be either the SHYFT default authentication or a single sign on system).

In all cases however, Logout will clear the system cache, thus losing for example: the user, their last selected Report, Filter selections, etc.

For those clients using both Auth0 and this LogoutURL feature, there is an additional one time set-up process in Auth0. The same LogoutURL established in tblAppConfig, must be also be made known to Auth0 as follows:

- Go to: https://manage.auth0.com/#/account/advanced
- Log into Auth0.
- Add the LogoutURL to the Allowed Logout URL's:



Properties 'PortalURL' and 'PortalText'
Optional properties, that when present, control the existence of a menu selection designed to return the user to a company portal page.

'PortalURL' should be the URL of the clients single sign on authentication. 'PortalText' is free text, up to 26 characters long, that will display in the Sprocket menu.

This option will not invalidate the system cache.

Google Analytics

Google Analytic (GA) supports the capture of AP usage data by logging the viewing of each Category. That is, each time the user swipes to display a Category, AP will send a log entry to GA. Following is an example report:

	Page	Pageviews	Avg. Time on Page	Time on Page	% Exit
		232	00:00:44	02:27:27	12.50%
23.	/APQAUnitTest_02/CommercialDashboard/HowareWeDoing/1	2	00:00:06	00:00:11	0.00%
24.	/APQAUnitTest_02/CommercialDashboard/HowareWeDoing/2	1	00:00:02	00:00:02	0.00%
29.	/APQAUnitTest_02/CommercialDashboard/Performance/1	1	00:00:02	00:00:02	0.00%
30.	/APQAUnitTest_02/CommercialDashboard/Performance/2	1	00:00:01	00:00:01	0.00%
31.	/APQAUnitTest_02/CommercialDashboard/Performance/3	1	00:00:03	00:00:03	0.00%

To implement this feature, add the EnableUserTracking property to tblAppConfig, as detailed here. There is nothing else for the implementer to do.

This feature is fully documented in the Analytic Platform Manual available in Confluence.

Navigation Menu Setup – pre v2.9.0

The initial set-up of a report requires entries in both APP and RPT tables. Most entries are made once by the BI. One entry is made by the DB for each reporting period. Following are the tables of concern.

Note that most of these tables have an Identity column. Identity columns are primary keys whose value is automatically entered only by the database and increments automatically. You can be assured the value is unique and will never be re-used.

The look-up process, from a report title to an RPT database is the following:

- Given a report title and data date, find tblReport.ReportID
- Using tblReport.ReportID, find tblCatalogTypeReportAssignment.CatalogTypeID
- Using tblCatalogTypeReportAssignment.CatalogTypeID and DataDate, find tblCatalogTypeReportAssignment.CatalogKey (i.e. the RPT database name)

APP.tblCatalog

When a user enters a URL in the browser, that URL is tied to an APP database by the IIS web server. Entries are made in tblCatalog that indicate the name of the RPT database along with other identifying information. As a routine manner, the DB will enter one new row for each reporting period. CatalogID is an Identity column.

```
,[Active]
FROM [AgileMv2_APP_EC2].[dbo].[tblCatalog]
```

Results | Messages

	CatalogID	CatalogTypeID	DataDate	CatalogKey	Description	PublishDate	Active
1	1	1	201312	AgileMv2_RPT_EC2	December 2013	2012-09-20 16:14:21.597	1

The above row would present the following to the user.

APP.tblCatalogType

Catalogs have a CatalogTypeID which comes from tblCatalogType.

```
,[Name]
FROM [AgileMv2_APP_EC2].[dbo].[tblCatalogType]
```

Results | Messages

	CatalogTypeID	Name
1	1	SalesForce Reporting
2	2	Incentive Compensation
3	3	Geisinger Patient
4	4	Geisinger Physician

APP.tblReport

tblReport rows are manually entered by the BI developer or automatically by AgileM Studio. Note the rows carry a CatalogTypeID (which is **NOT** used). ReportID is an Identity column.

```
,[ModifyDate]
FROM [AgileMv2_APP_EC2].[dbo].[tblReport]
```

Results | Messages

	ReportID	Title	ReportName	CatalogTypeID	ExcelReportID	DisplayOrder	Active
1	17	EC2 Usage-Cost Monitor	EC2 Usage-Cost Monitor	1	NULL	17	1

APP.tblCatalogTypeReportAssignment

Maps a ReportID (from tblReport) to a CatalogTypeID (in tblCatalog).

```
,[ReportID]
FROM [AgileMv2_APP_EC2].[dbo].[tblCatalogTypeReportAssignment]
```

Results | Messages

	CatalogReportID	CatalogTypeID	ReportID
1	1	1	17

APP.tblUser

Users are established in the tlbUser table. On a routine bases DB's will make these entries. BI's make entries during initial development. UserID is an Identity column.

```
FROM [AgileMv2_APP_EC2].[dbo].[tblUser]
```

Results | Messages

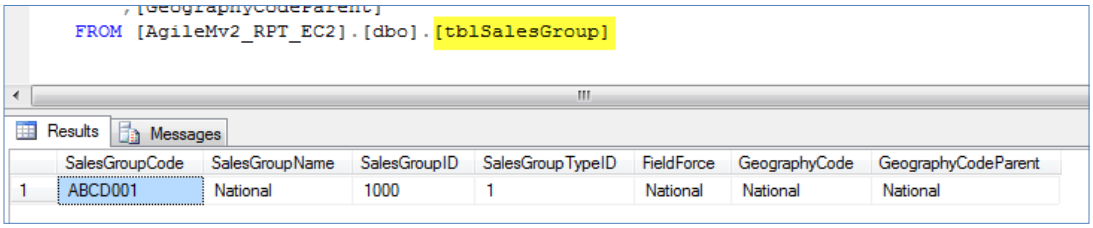
	UserID	UserTypeID	FirstName	LastName	Username	Password	Email
1	8	1	System	Admin	TPSDemo	0x7d763dc5d6112436b7b5133753818621	dnguyen@trinitypharma.com

Note that the Password column is an encrypted value – you can NOT just enter free text. As a general rule, paste a value in the Password column from a known working password.

Also, note that there is a tblUser.ResetPassword column (not pictured above), that can tell the application to prompt for a new password. Set ResetPassword = 1, log on with the old password and then receive the following prompt:

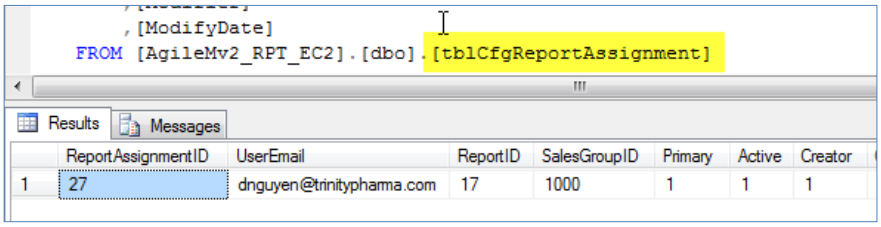
RPT.tblSalesGroup

Sales Groups are established in tblSalesGroup.



RPT.tblCfgReportAssignment

Finally, an individual user is given permission to access a report through tblCfgReportAssignment. Note that uniqueness is established through the UserEmail, not UserID. ReportAssignmentID is an Identity column. Set Primary to TRUE.



When trouble shooting a logon, the following Select statements conveniently gather, in one SSMS result window, helpful data:

```
select * FROM [MSLEU_AgileMv2_APP].[dbo].[tblCatalog]
select * FROM [MSLEU_AgileMv2_APP].[dbo].[tblCatalogType]
select * FROM [MSLEU_AgileMv2_APP].[dbo].[tblReport]
select * FROM [MSLEU_AgileMv2_APP].[dbo].[tblCatalogTypeReportAssignment]
select * FROM [MSLEU_AgileMv2_APP].[dbo].[tblUserType]
select * FROM [MSLEU_AgileMv2_APP].[dbo].[tblUser]
select * FROM [MSLEU_AgileMv2_RPT].[dbo].[tblSalesGroup]
select * FROM [MSLEU_AgileMv2_RPT].[dbo].[tblCfgReportAssignment]
```

Navigation Menu Setup – beginning with v2.9.0

Beginning with v2.9.0 two changes were made in establishing Reports and the left hand navigation menu:

1) The left hand navigation panel now supports three "levels".
2) The tables required to establish a Report were simplified.

Three Levels of Navigation

The left hand navigation panel has up to three "levels":

1) **Report Group**. A free text string that groups reports by function. For example, if a client has multiple reports whose purpose is to present sales information, than those reports could be grouped under the report group of "Sales".
2) **Report Name**. All Reports are given a name.
3) **Data Date**. All reports are associated with a data date. For example, a monthly report could be associated with the January 2017 data date.

The structure of the Report selection tree is configurable to the client's needs and consists of report group, report name and data period. There are four possible combinations:

1) Report group, report name, data date.
2) Report group, report name.
3) Report name, data date.
4) Report name.

Note that for a give project there is no ability to mix options. For example, if Report1 uses option 2 below ("Report Group, Report name"), than all reports in the project must use "Report Group, Report name".

The following presents an example of each possible combination:

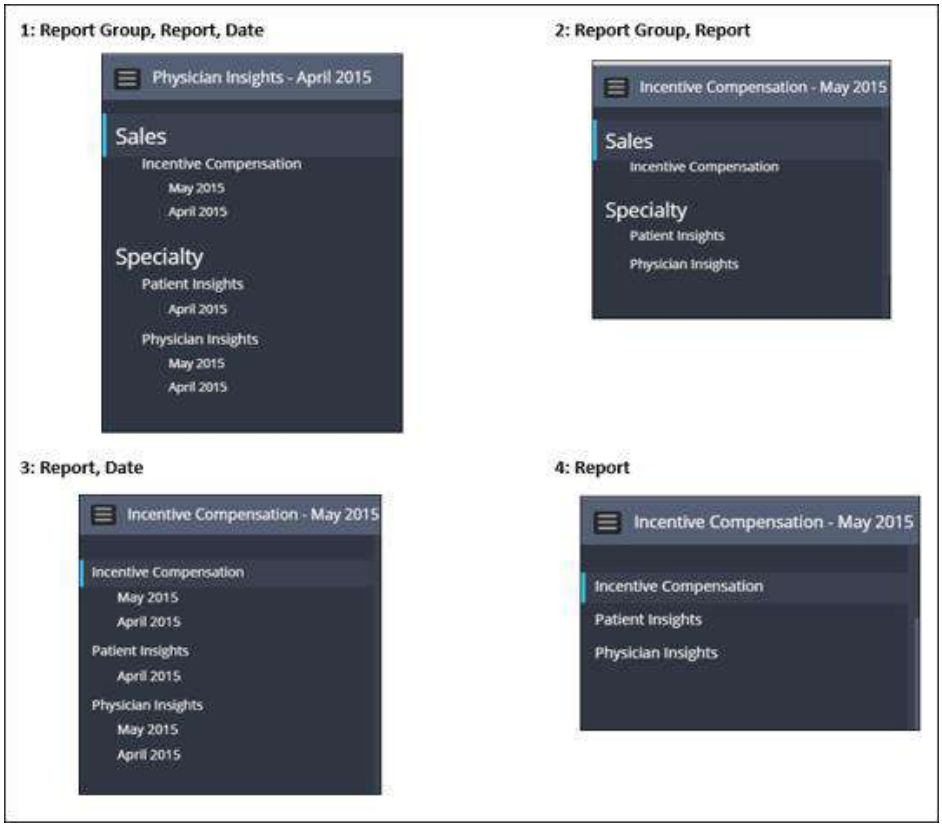


Table Simplification

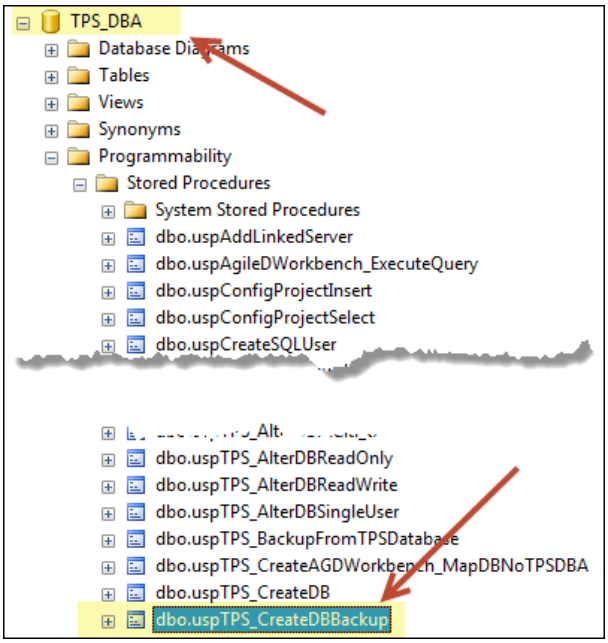
The following tables and columns, used in pre-v2.9.0, are no longer used:

1) **APP.tblCatalogType**. No longer used at all.
2) **APP.tblCatalogTypeReportAssignment**. No longer used at all.
3) **RPT.tblSalesGroup**. The table remains, but is no longer used by the Report set-up or navigation system.
4) **APP.tblCatalog**. Removed the column CatalogTypeID.
5) **APP.tblReport**. Removed the column CatalogTypeID.
6) **RPT.tblCfgReportAssignment**. The column Primary remains, but is no longer used for the navigation panel.

Converting from pre-v2.9.0 to v2.9.0

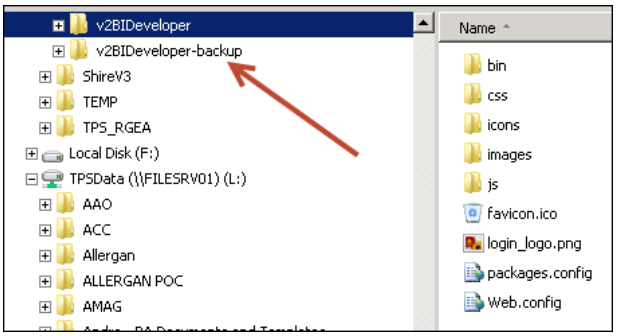
When first converting from pre-v2.9.0 to v2.9.0, perform the following steps in preparation for establishing the reports. Note that backups are particularly necessary since there are several table changes.

1) **Backup the APP and RPT databases**.



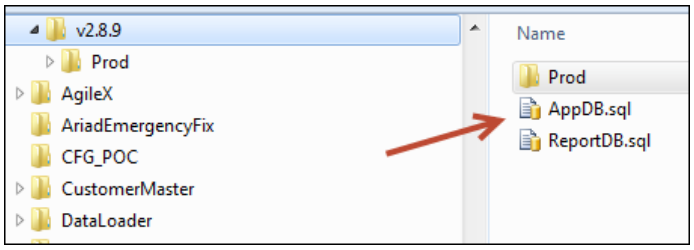
2) **Backup the Analytic Platform web site**.

Make a copy of the web site in a backup folder.



3) **Execute the AppDB.sql and ReportDB.sql scripts and update the AP web site**.

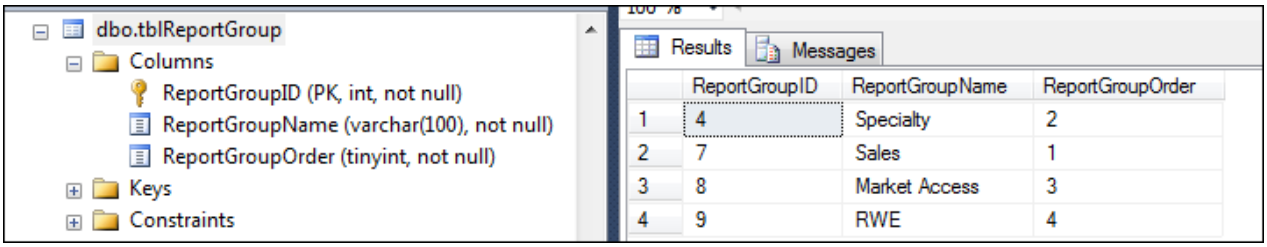
As always update the APP and RPT databases and the AP web site.



Establishing a Report and the Navigation Panel

Following are the steps necessary to establish a report and the navigation panel hierarchy:

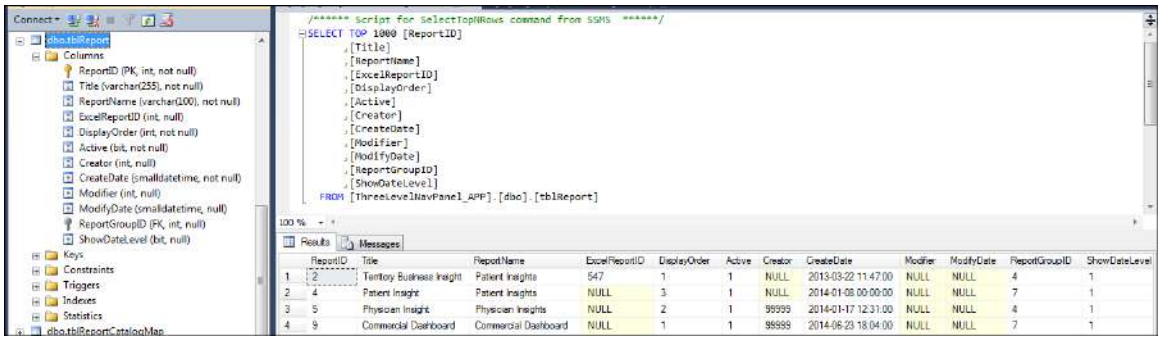
1) **APP.tblReportGroup**. If Report Group is to be used in the navigation hierarchy, make an entry in APP.tblReportGroup. Note the ability to control the Report Group order.



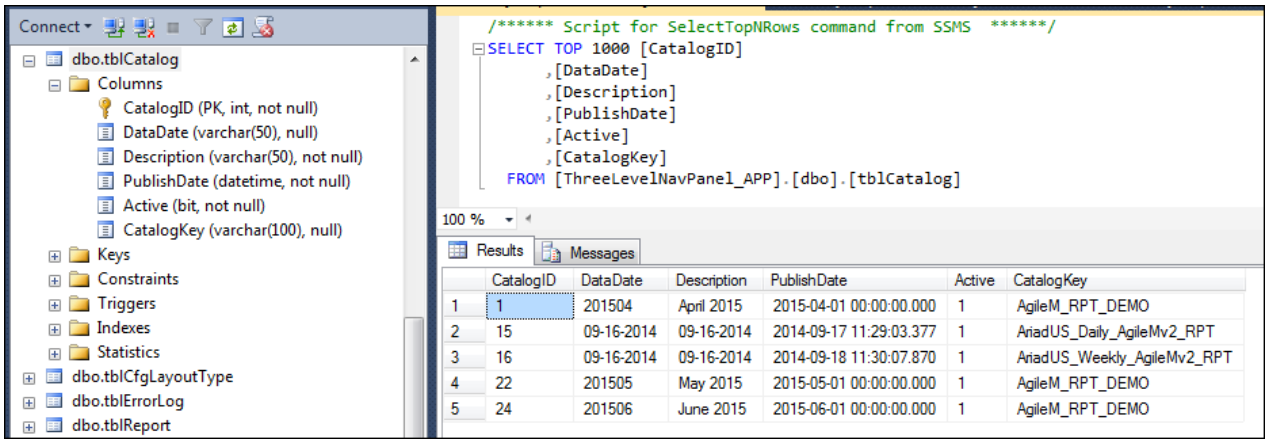
2) **APP.tblReport**. Always make an entry in APP.tblReport.

   Report Group ID: If Report Group is to be used in the navigation hierarchy then populate ReportGroupID from tblReportGroup, else set to NULL. Note that a non-null value in ReportGroupID controls the presence of the report group level in the navigation panel.

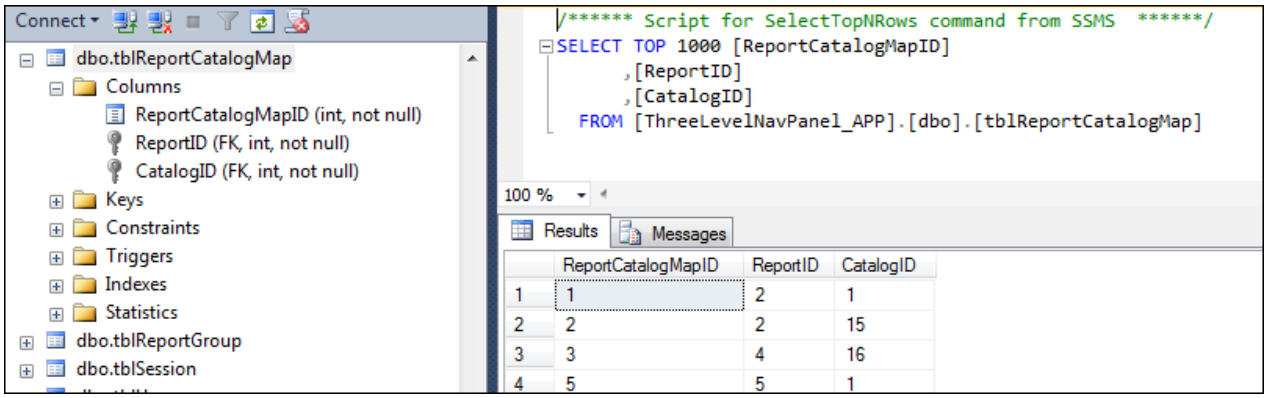
   ShowDateLevel: If data date is to be used, then set ShowDateLevel=1, else set ShowDateLevel=0. Note that ShowDateLevel controls the presence of the date level in the navigation panel.



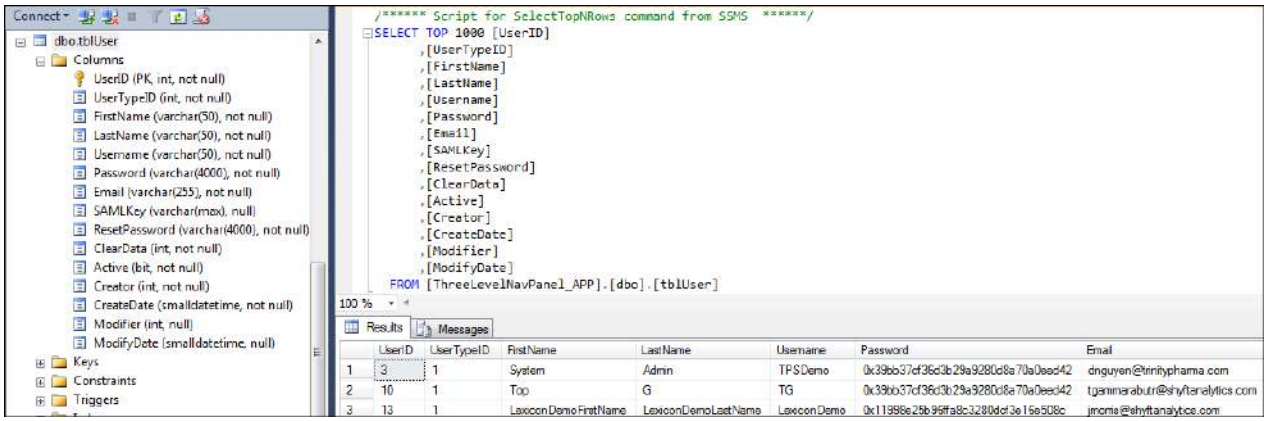
3) **APP.tblCatalog**. Always make an entry inAPP.tblCatalog. CatalogKey is a reference to an RPT database located on the same server.



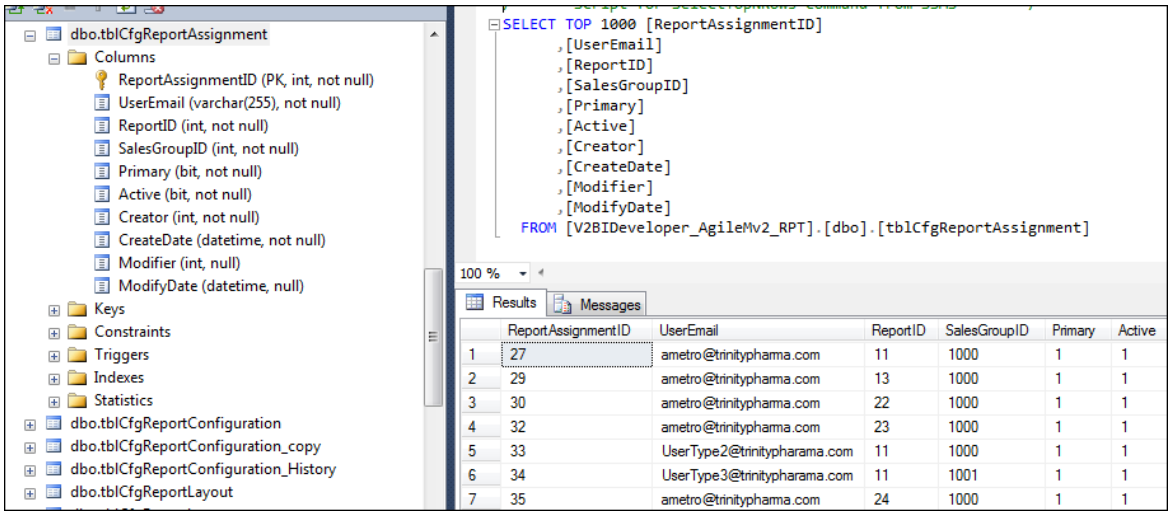
4) **APP.tblReportCatalogMap**. Always make an entry inAPP.tblReportCatalogMap to join a report to its catalog (that is, its RPT database).



5) **APP.tblUser**. Insert one row for each user that is to have access to any report.



6) **RPT.tblCfgReportAssignment**. For each user/report combination enter one row.



For those environments where one APP database services multiple RPT databases, the following sql statement is helpful for mapping a Report to its Catalog.

```
select c.catalogKey, r.reportName, r.reportID
  from tblReport r
  join tblReportCatalogMap rcm on rcm.reportID = r.reportID
  join tblCatalog c on c.catalogID = rcm.catalogID
  order by c.catalogKey, r.reportName
```

catalogKey	reportName	reportID
AgileM_RPT_DEMO	Field Insights	2
AgileM_RPT_DEMO	Incentive Compensation	10
AgileM_RPT_DEMO	Payer Insights	17
AgileM_RPT_DEMO	Payer Operational Alerts	4
AgileM_RPT_DEMO	Physician Insights	5
AriadUS_Daily_AgileMv2_RPT	Field Alerts	13
AriadUS_Daily_AgileMv2_RPT	Payer Alerts	14
AriadUS_Weekly_AgileMv2_RPT	Executive Insights	9
AriadUS_Weekly_AgileMv2_RPT	Physician Profile	15
AZ_MOB_RPT_DI_T2D_AgileM_DEMO	Disease Insights	20

Integrated Security in Web.config

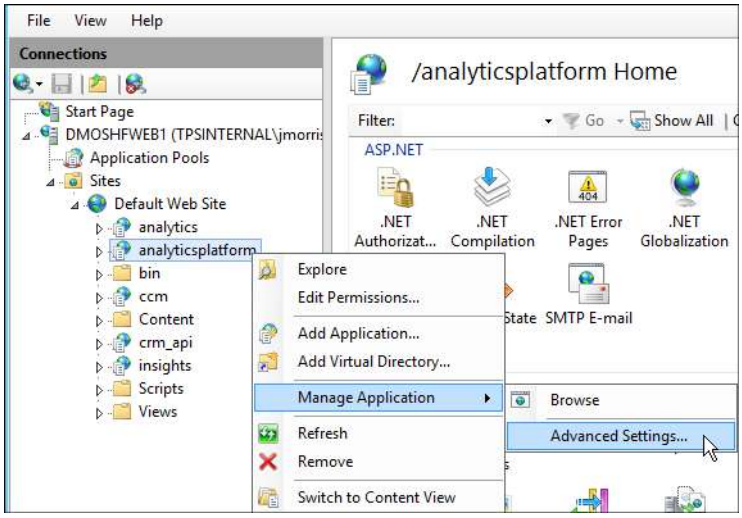
For security purposes, the web.config file for an Analytic Platform web site must use integrated security in its connection string. That is, a connection string with a hard coded user id and password is unacceptable.

Following is a proper example:

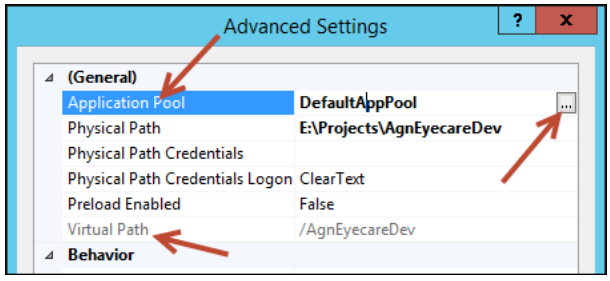
```
<appSettings>
   <add key="App_DB" value="Data Source=localhost; Initial
   Catalog=AgileM_APP_AGN; Integrated Security= true;
   Timeout=600; pooling='true'; Max Pool Size=100" />
</appSettings>
```

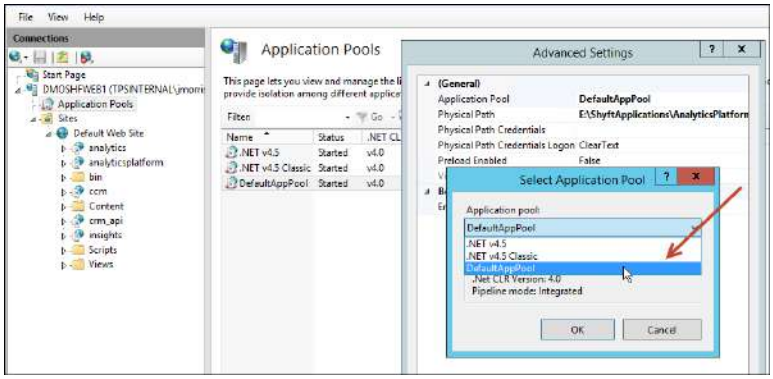
The following reviews how to set up support for integrated security.

1) From within Internet Information Server (IIS) on your Analytic Platform server, select the AP application, Manage Application, Advanced Settings…

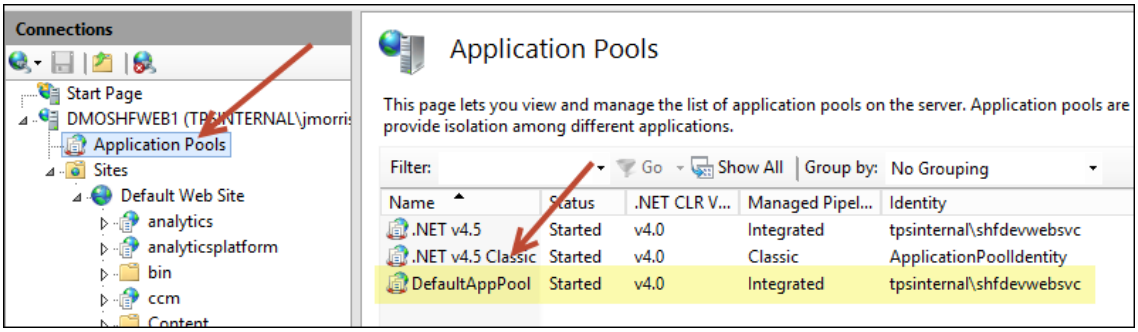


Note the name of the Application Pool. If needed, select the ellipses to change. Also note the virtual path (for use in step #3).

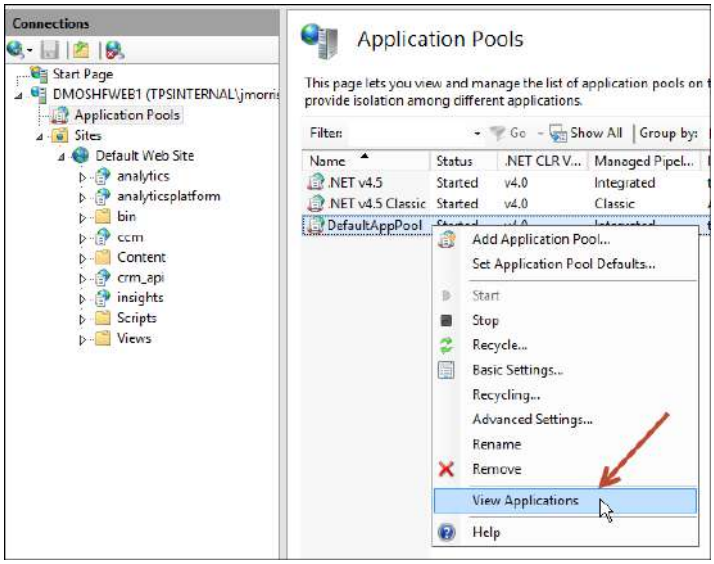


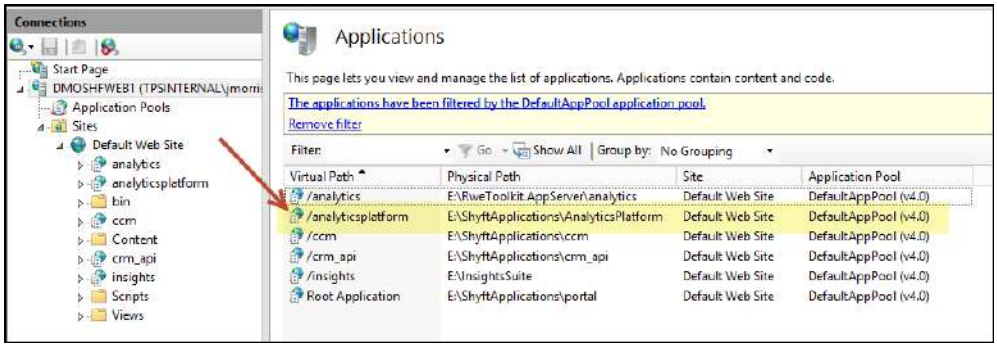


2) Double click Application Pools. Observe the application pool you wish to use – in this case DefaultAppPool. Note also that the identity of application pool must be: a Service Account. The Service Account is created by IT and if not available will require a help desk ticket. The names follow the pattern of: <client><environment: dev, qa, prod>websvc.

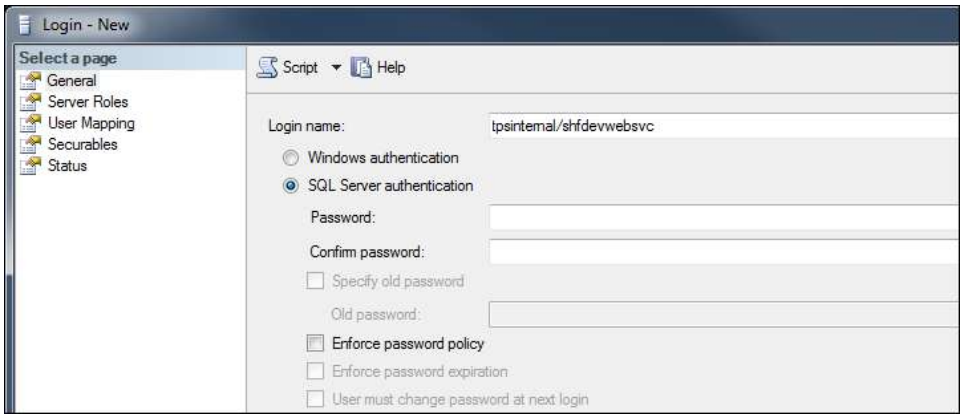


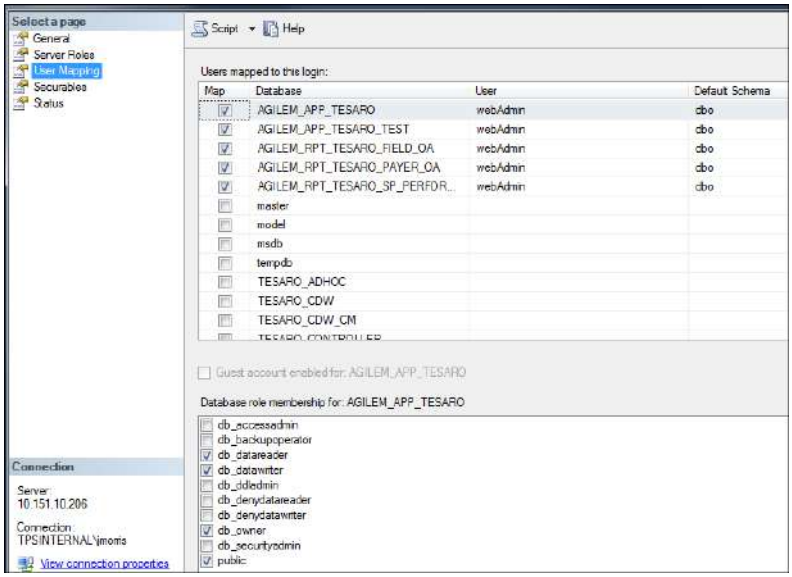
3) To confirm, right click the selected application pool and take View Applications. Assure that your virtual path, as noted in step 1, is using the desired application pool.





4) On the SQL server, using the same Service Account in step #2, map the APP and RPT databases to the Service Account login.





Stored Procedure Standards

This section reviews standards that have been established when writing mobile v2 stored procedures. Standards make stored procedures:

1) Consistent across all reports.
2) Easier to write originally.
3) Easier for others to maintain.
4) Less error prone.

Background

Most Components have a relatively small amount of data. For example, the following Component requires only a small record set to populate the Free Form.

However, some Grids could contain hundreds of rows with scores of columns. In such cases, their full record sets are so large that sending all rows to the iPad in one shot would result in un-acceptable performance.

So, rather than sending all data in one record set, a limited number of rows are sent in a "page". Then, as the user scrolls down the Grid, another "page" is sent. Such Grid Components are called "paging" Grids. Note that only Grids use this "paging" concept and that a Grid with a small amount of data need not support paging.

Al l BI personnel are expected to implement the following Stored Procedure Standards:

1) **Stored Procedure Names**. Category stored procedures are expected to be named as follows:

rsp<Report Name Abbreviation>_<SectionName>_<CategoryName>

where:

- **<Report Name Abbreviation>** is a short abbreviation of the report name (hopefully 2-5 characters in length) in all capital letters
- **<SectionName>** is a very close, or identical, match with the Section LongTitle attribute. All characters are lower case except the first letter of each word which is capitalized. No spaces.
- **<CategoryName>** is a very close, or identical, match with the Category Title attribute (if it exists), otherwise a description of the Category content. All characters are lower case except the first letter of each word which is capitalized. No spaces.

Note that Section and Category numbers are not to be used. For example:

- rspPLAN_Perf_Reach
- rspPLAN_Target_Summary
- rspFIELD_OperAlerts_Scripts

2) **dbo.rspTemplate_NonPagingSP**. The starting point for all non-paging Components. There are two primary features in this stored procedure.

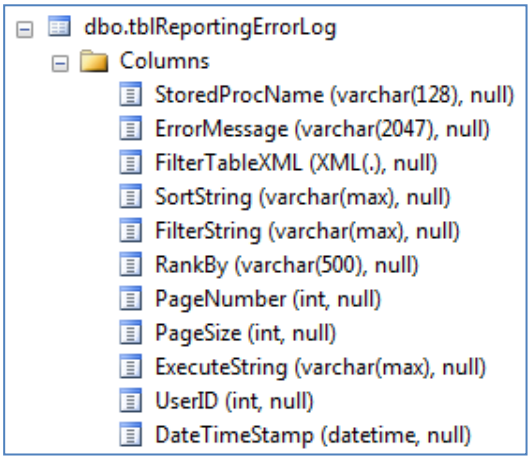
One, the SQL that is used to process the Filter XML.

```
--  Populate @tblFilterNameValue with the Filter Name/Value pairs
    DECLARE @tblFilterNameValue as tblFilterNameValue
    INSERT INTO @tblFilterNameValue EXEC rspParseXMLFilterString @filterTableXML

--  Get filter values with a function call
    DECLARE @vchTimePeriod VARCHAR(10) = (SELECT dbo.GetFilterValue(@tblFilterNameValue, 'TimePeriod'))
    IF @vchTimePeriod IS NULL RAISERROR ('Time Period is null',16,255)
```

Two, the fact that there is a Begin/End Try across the entire stored proc with a standardized Catch.

```
----------
----Catch for the full stored proc
----------
BEGIN CATCH
    DECLARE @vchStoredProcName VARCHAR(128) = OBJECT_NAME(@@PROCID)
    DECLARE @vchErrorMessage VARCHAR(2047) = ERROR_MESSAGE()
    DECLARE @intUserID INT = (SELECT dbo.GetFilterValue(@tblFilterNameValue, 'userID'))

    EXEC [uspLogStoredProcError]
         'NonPaging', @vchStoredProcName, @vchErrorMessage,
         @filterTableXML, @vchSortString, @vchFilterString,
         @intUserID

    RAISERROR (@vchErrorMessage,16,255)

END CATCH
```

Note that you should not adjust the Catch – if executed, it will automatically write one row to the tblReportingErrorLog table.



Pay particular attention to the ExecuteString column. This column will be populated with the same SQL code used by the mobile application to execute the stored proc. Using this string you should be able to debug a stored procedure.

3) **dbo.rspTemplate_PagingSP**. The starting point for all paging Components. The same standards exist in this paging template as in the non-paging template – standard XML Filter processing, a single Try Block and a pre-written Catch block that needs no adjustment.

The primary difference in the paging stored proc is the presence of three more input variables: @vchRankBy, @intPageNumber and @intPageSize. When a paging stored proc is called, these variables in effect say: "What page number should be returned and how many rows should be in the page".

```
(
      @filterTableXML XML
     ,@vchSortString varchar(max) = ''
     ,@vchFilterString varchar(max) = ''
     ,@vchRankBy varchar(500) = ''
     ,@intPageNumber int = null
     ,@intPageSize int = null
)
```

AgileM Studio

AgileM Studio Privileges and Historical Logging

The ability to change reports through AgileM Studio is controlled through the security concepts of Roles and Privileges. The purpose of controlling access is to implement the following over-arching policies:

- All employees should be able to view any Report in AgileM Studio.
- BI's responsible for a Report will have the privilege to change all aspects of a Report.
- BA's responsible for Report could be given the privilege to change the values of selected attributes. For example, a BA could change a column title and column width, but not a container type. (Note that while AgileM Studio fully supports this functionality, as of December 2014, such privileges have not actually been given).
- Security is controlled at the Project level (and not the Report level). That is, whatever privilege an individual has for a Project will be assigned to all Reports in that Project.

Access is implemented in AgileM Studio through the concepts of Roles and Privileges:

- **Role**. An employee is assigned a Role of either: Admin, BI, or BA. This Role is in effect for an employee during their use of AgileM Studio regardless of which Project they are working in.
- **Privilege**. Each employee is assigned, per Project, the Privilege of either: ReadWrite or ReadOnly.

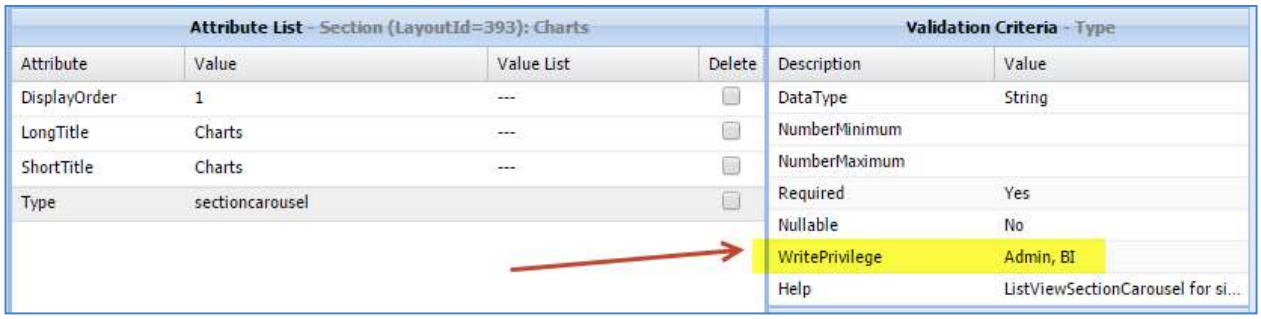
For example, an employee, when working in AgileM Studio, on Project1/Report1 might have the Role of BI (which remains true regardless of which Project they are on), and the Privilege of ReadWrite. In this case they could change all aspects of a Report. But the same employee on Project2/Report 1 would still be a BI but might have the Privilege of ReadOnly.

The following chart summarizes how all this is implemented within AgileM Studio.

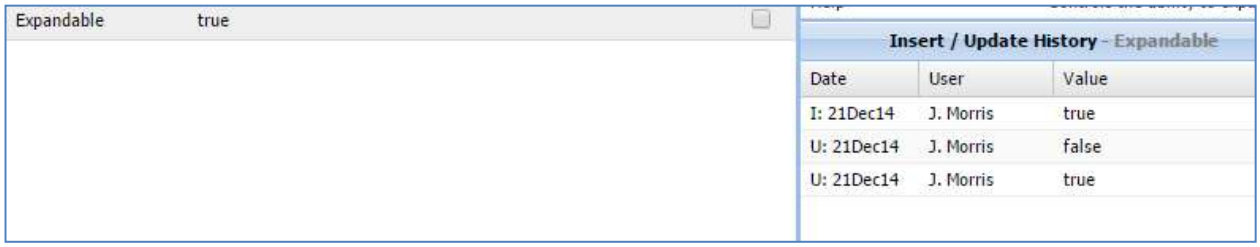
AgileM Studio Role:	**Admin**		**BI**		**BA**	
Project Privilege:	ReadOnly	ReadWrite	ReadOnly	ReadWrite	ReadOnly	ReadWrite
Change Attributes	y	y	n	Y	n	Y
Draw Containers	y	y	n	Y	n	y
Modify Layouts	y	y	n	Y	n	n
Edit Dot Liquid	y	y	n	Y	n	n
Copy Reports	y	y	n	Y	n	n
Autogenerate SP	y	y	n	Y	n	n
Update Layout & Children	y	y	n	y	n	n

Note that when opening a report the title indicates the user Role and Project Privilege.

Also note that when working on an Attribute, a comma separated list is presented with the list of Roles that have write privilege.



AgileM Studio also supports historical logging of inserts and updates in attribute values. The system will maintain up to five events – the single Insert event, (noted as "I" under the Date column), and up to a maximum of the four most recent update events (noted as "U" under the Date column).



Configuration Comparison and Update Functionality

AgileM Studio supports the ability to compare and update the configuration tables in the Development environment with the QA and Processing environments.

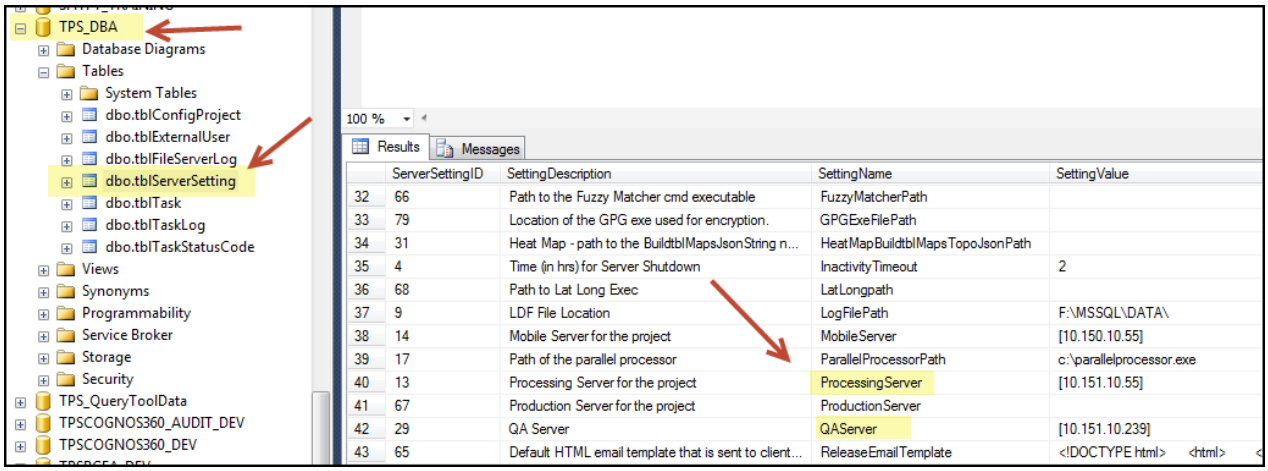
The system can:

- Compare the rows in RPT.tblCfgReportLayout and RPT.tblCfgReportConfiguration, in the Development environment, with the same rows in either the QA or Processing environments.
- Update the rows in Development to QA or Processing.
- Support comparison/update functionality between Development and QA or Processing. When both QA and Processing exist, the system will prompt for which environment you would like to compare/update against.

Usage Note:

- AgileM Studio locates the QA and Processing server addresses in the TPS_DBA.tblServerSetting table (with SettingName(s) of ProcessingServer and QAServer). Note that QAServer is optional.

These values must be properly populated for the following feature to work:



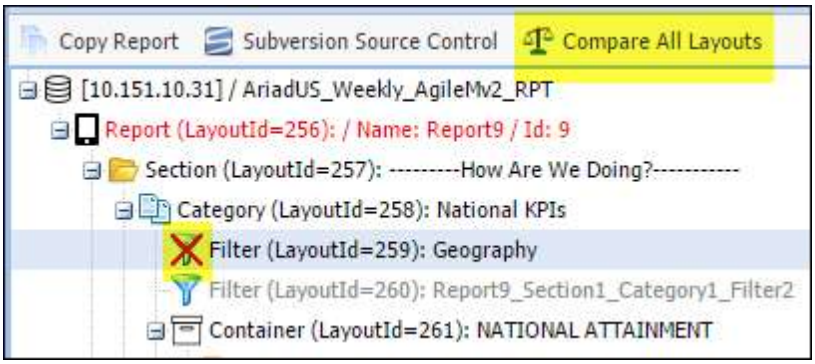
	ServerSettingID	SettingDescription	SettingName	SettingValue
32	66	Path to the Fuzzy Matcher cmd executable	FuzzyMatcherPath	
33	79	Location of the GPG exe used for encryption.	GPGExeFilePath	
34	31	Heat Map - path to the BuildtblMapsJsonString n...	HeatMapBuildtblMapsTopoJsonPath	
35	4	Time (in hrs) for Server Shutdown	InactivityTimeout	2
36	68	Path to Lat Long Exec	LatLongpath	
37	9	LDF File Location	LogFilePath	F:\MSSQL\DATA\
38	14	Mobile Server for the project	MobileServer	[10.150.10.55]
39	17	Path of the parallel processor	ParallelProcessorPath	c:\parallelprocessor.exe
40	13	Processing Server for the project	ProcessingServer	[10.151.10.55]
41	67	Production Server for the project	ProductionServer	
42	29	QA Server	QAServer	[10.151.10.239]
43	65	Default HTML email template that is sent to client...	ReleaseEmailTemplate	<!DOCTYPE html> <html>

Comparison

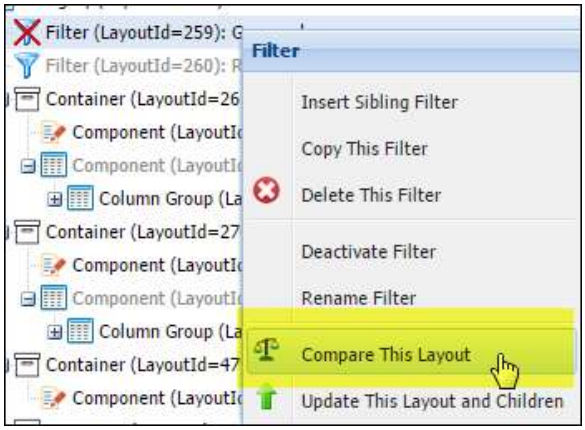
The system supports comparing Layouts between Development and either Processing or QA. When a comparison is performed, AgileM Studio sequentially processes each LayoutID in RPT.tblCfgReportLayout and, for each LayoutID, finds the Attributes and associated Values in RPT.tblCfgReportConfiguration. A comparison is then made against either Processing or QA.

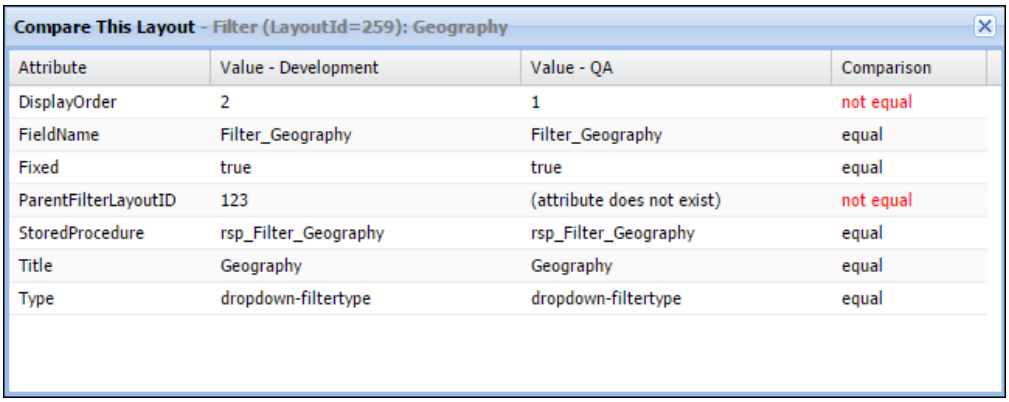
There are two ways to compare the layouts across environments: one, compare all LayoutID's (and provide a simple equal or not equal indication) or compare a single LayoutID (and provide a detailed comparison of each Attribute and its Value).

Compare All Layouts. Click the button Compare All Layouts. When a Layout's associated Attributes and Values are not identical the Layout's icon will be highlighted with a red "X". Note there is no indication what the differences are, just that there are some differences.



Compare One Layout. Right click a single LayoutID, select Compare This Layout. A table will be presented that lists each Attribute and its value in both environments.



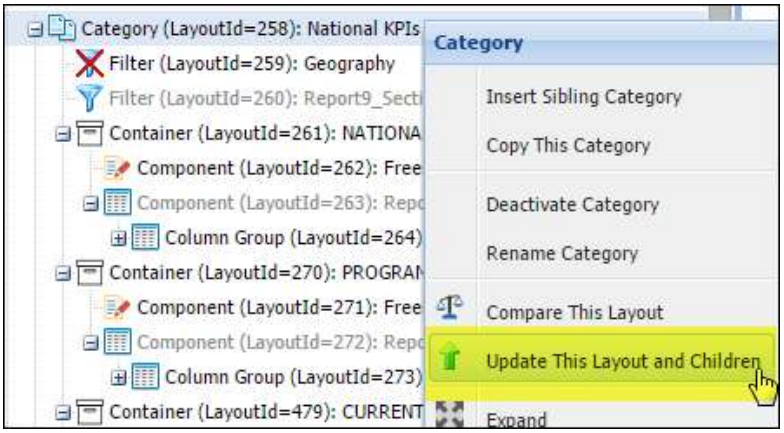


Compare This Layout - Filter (LayoutId=259): Geography

Attribute	Value - Development	Value - QA	Comparison
DisplayOrder	2	1	not equal
FieldName	Filter_Geography	Filter_Geography	equal
Fixed	true	true	equal
ParentFilterLayoutID	123	(attribute does not exist)	not equal
StoredProcedure	rsp_Filter_Geography	rsp_Filter_Geography	equal
Title	Geography	Geography	equal
Type	dropdown-filtertype	dropdown-filtertype	equal

Update

The system supports updating Layouts between Development and either Processing or QA. To update the Attributes and Values of a LayoutID, **and all its children**, right click the LayoutID and select Update This Layout and Children. Note that the values of the Identity columns are preserved during an update.



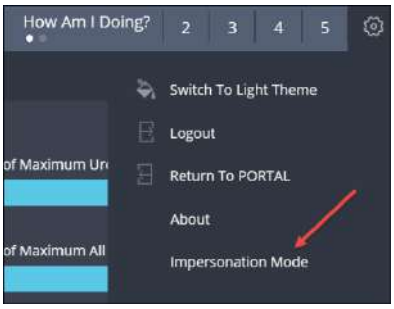
Misc

Support for Impersonation Mode

The system supports 'Impersonation Mode' – that is, the ability to log into the system as a different user. When debugging an issue or to fully and accurately simulate a user's experience, it can be useful for support personnel to be able to log into the system as that user.

To enter Impersonation Mode:

1) Log in with an account whose e-mail address has the domain '@shyftanalytics.com'.
2) Select Impersonation Mode from the Sprocket menu. (Note, this option will only be available when the current users e-mail domain is '@shyftanalytics.com').



3) Enter the Username of the person to be impersonated. Note that Password is not required.

4) Note the title bar indicates the session is under Impersonation Mode.

Manually moving a Report from One Project to Another Existing Project

The following details the steps required to move a Report to another Project:

1) Create a copy of the RPT database on the To server. Moving the RPT database can vary from trivial to complex and is beyond the scope of this manual.
2) Add a row to the APP.tblCatalog table. Note the new CatalogID value.
3) Observe APP.tblReportGroup. Note the ReportGroupID that the new report will be added to or create a new APP.tblReportGroup row.
4) Add a row to APP.tblReport. Note the new ReportID value.
5) Add a row to APP.tblReportCatalogMap.
6) Note Email values from APP.tblUser that are to have access to the Report.
7) Add rows to RPT.tblReportAssignment.
8) Note the new ReportID value will likely not match the ReportID value in RPT.tblCfgReportLayout and RPT.tblCfgReportConfiguration. Manually adjust with statements like the following:

```
update [dbo].[tblCfgReportLayout]
  set reportid=<new reportID] where reportid=<old reportID>
update [dbo].[tblCfgReportConfiguration]
  set reportid=<new reportID] where reportid=<old reportID>
```

9) Map the required Logins to the new RPT db. You can often know which Logins should be mapped by selecting a catalog from APP.tblCatalog and applying the same mappings.

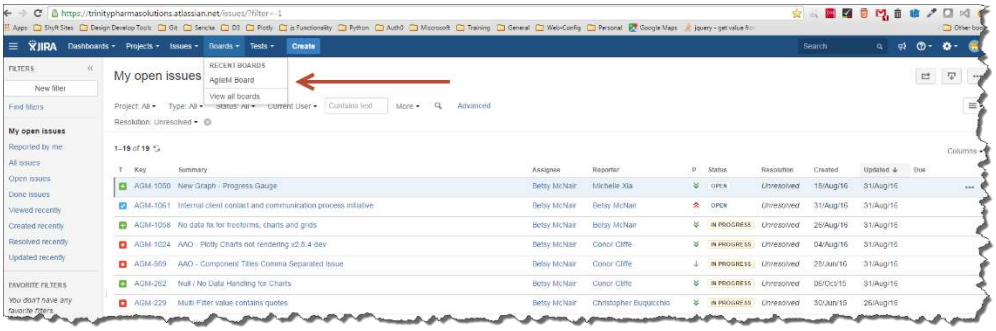
Deleting a Report

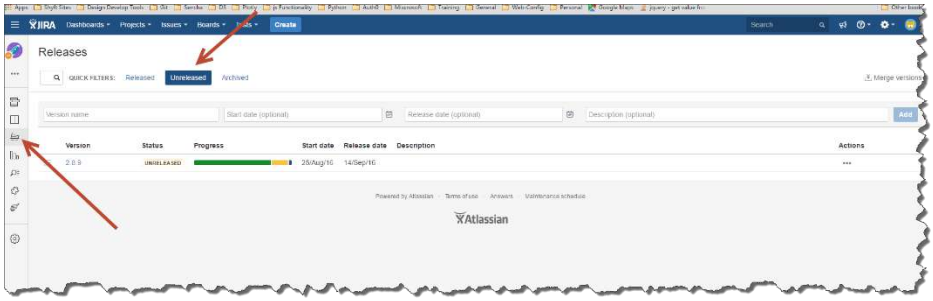
The following details the steps required to delete a Report from a Project:

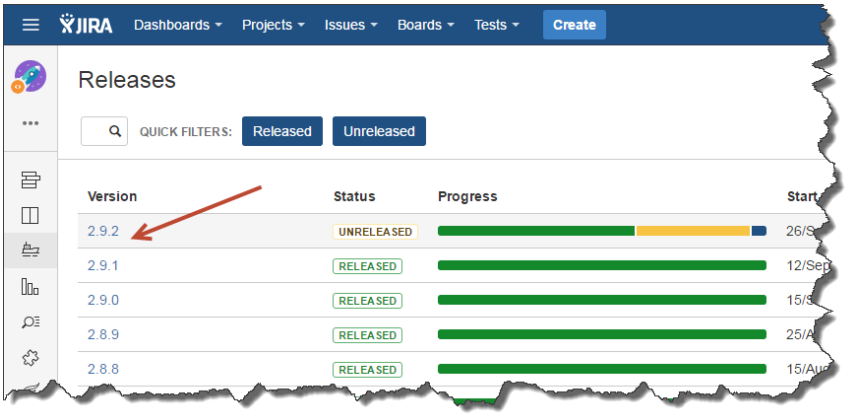
1) Delete the row in APP.tblReport.
2) Possibly delete a row in APP.tblCatalog if the catalog entry is only used for the deleted report.
3) Delete the row in APP.tblReportCatalogMap.
4) Delete rows in RPT.tblReportLayout (which will cascade delete to RPT.tblReportConfiguration).
5) Delete rows in RPT.tblReportAssignment.

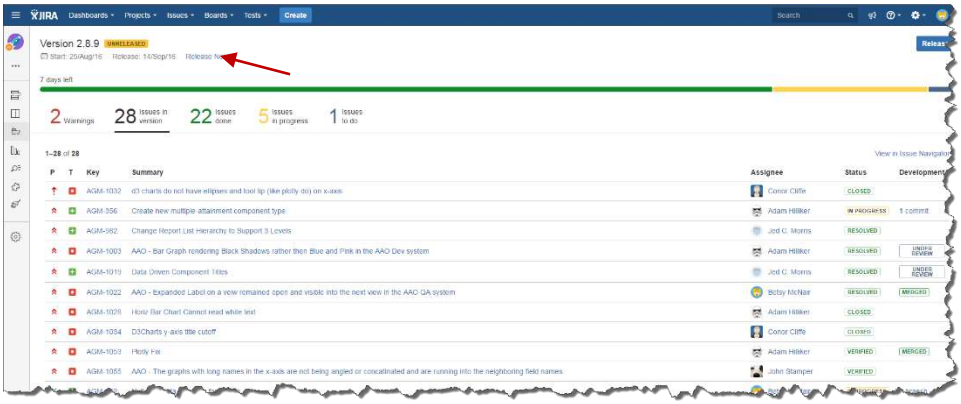
Monitoring Releases in JIRA

Information on previous and current releases can be monitored in JIRA, (at: https://trinitypharmasolutions.atlassian.net/login), as follows.







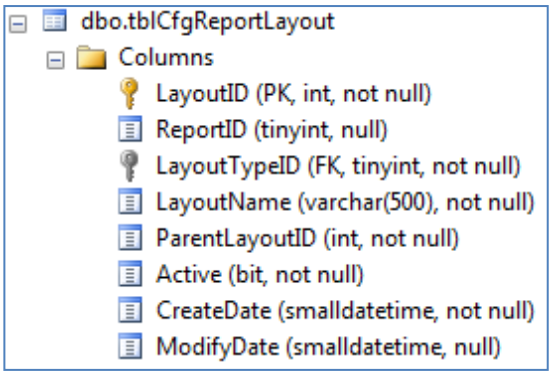


ReportLayout and ReportConfiguration

The two reporting tables, tblCfgReportLayout and tblCfgReportConfiguration, form the heart of the mobile reporting system. The primary work of AgileM Studio is to maintain these tables and BI developers should be well versed in their content.

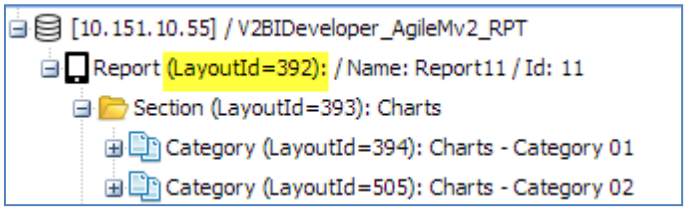
tblCfgReportLayout

This table contains one row for each object (Report, Section, Category, Container, etc.).



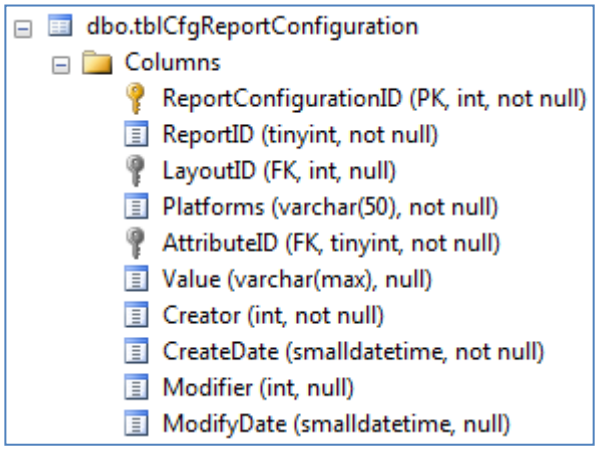
LayoutID is an identity column (i.e. a unique number assigned by SQL). LayoutTypeID refers to one of the object types. The primary purpose of this table is merely to establish the existence of an object.

Note that AgileM Studio displays the LayoutId of each tblCfgReportLayout row.



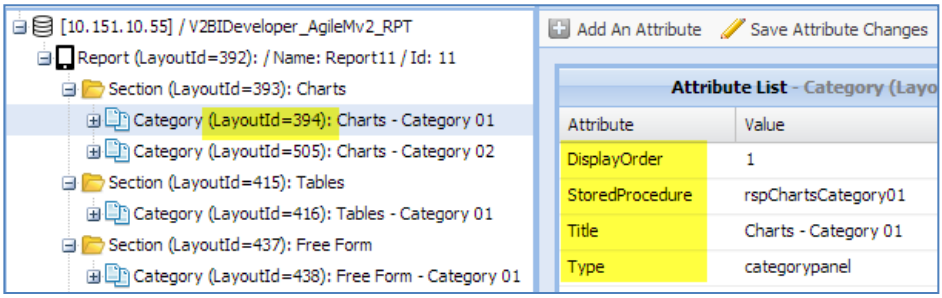
tblCfgReportConfiguration

This table contains one to many attribute rows for each tblCfgReportLayout row.

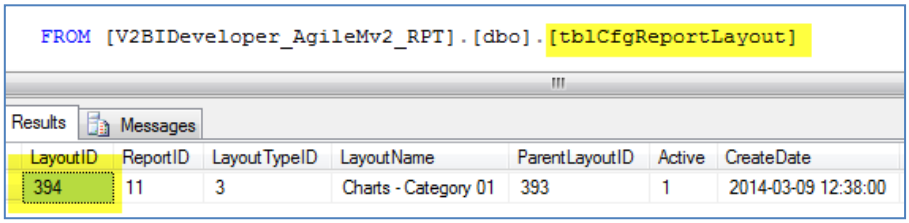


Note that LayoutID is a foreign key back to tblCfgReportLayout.LayoutID. Also note the AttributeID column that identifies specific Attributes associated with the LayoutID.

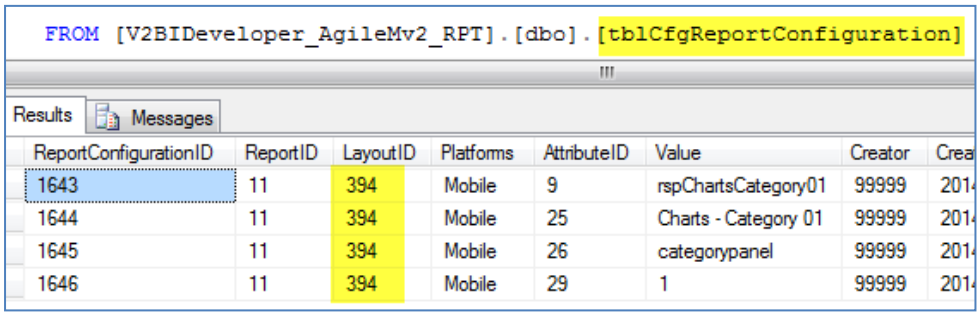
So, for example in the following, the one tblCfgReportLayout row (with LayoutID = 394), will have four tblCfgReportConfiguraiton rows.



All of this can be seen in the following. One Layout row:



Mapping to the four Configuration rows:



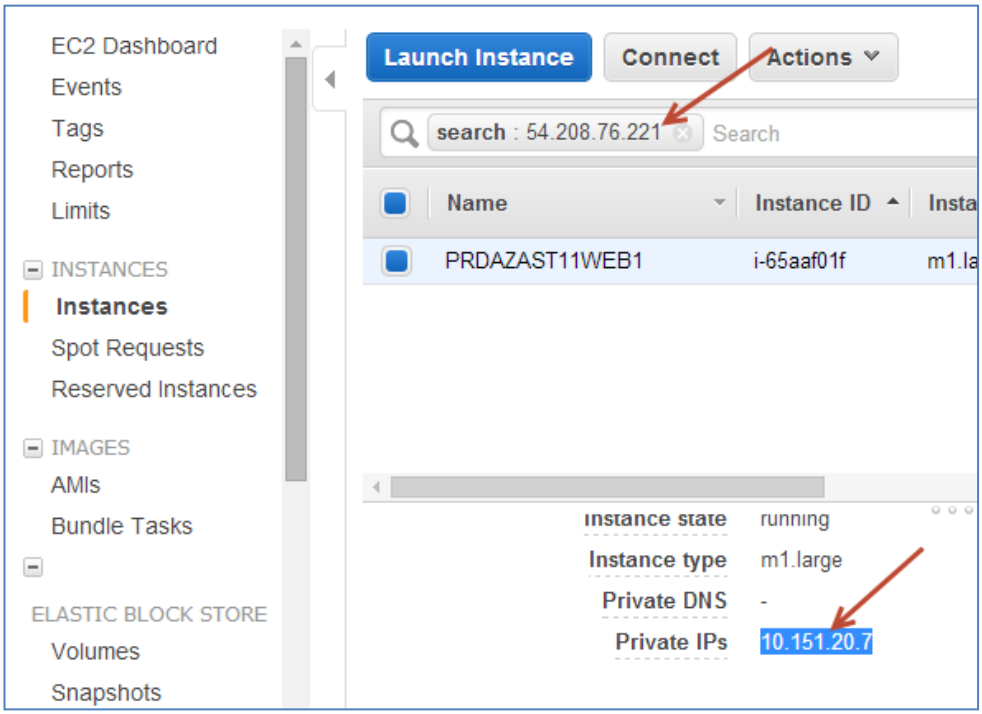
Mapping a public URL to its private IP address

When given a public URL to Analytic Platform, (i.e. the URL address clients will use to access mobile v2), you can convert it to its private IP address by doing the following:

For example, given the public URL https://astellas.trinityagile.com/agilemv2/, execute ping as follows:

In this case note the public IP address of [54.208.76.221].

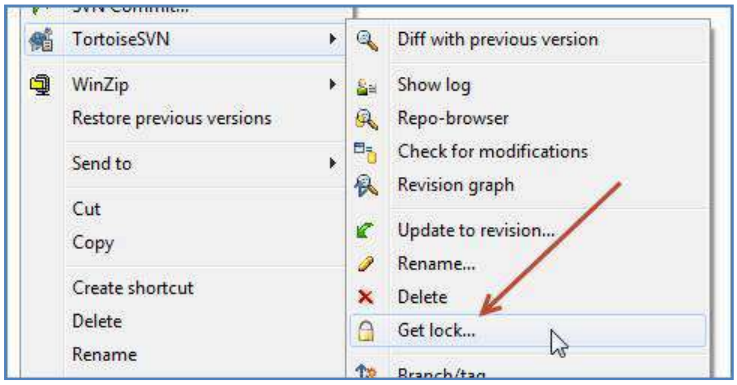
Then from within EC2, search on [54.208.76.221], which will then present the private IP address of, in this case, 10.151.20.7.



Procedure for editing documents in SVN

When you wish to edit any document in the Solution Services shard SVN document repository please follow these steps:

1) Do an SVN Update on the document to get the latest.
2) Get Lock… This will prevent someone else from committing the document until after you have committed and unlocked the document.
3) Make your edits. (Please don't hold onto the document for too long – a day or two is ok).
4) Do a Commit and release the Lock .
5) Some general communication can also help.



Client Custom logos

AP supports two client custom logos – one, a login logo that appears on the User/Password page and two, a branding logo that appears on the report navigation bar.

User/Password Page

Clients may optionally request their own logo be used on the initial login screen. For example:

To implement this, replace the following file: <root level of the web site>/login_logo.png. Additional notes:

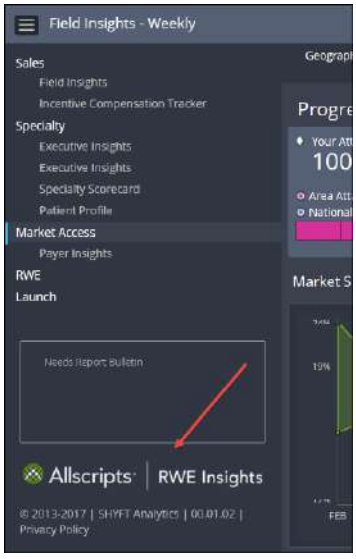
- This file must be in .png format.
- The size of the image must look appropriate.
- login_logo.png is at risk for being over written with the standard SHYFT logo on upgrades.

Branding logo – navigation bar

Clients may optionally request that a .png file appear between the report bulletin and the Privacy Policy line of the navigation panel.

To implement this, create a .png file here: <root level of the web site>/ReportNavLogo.png.

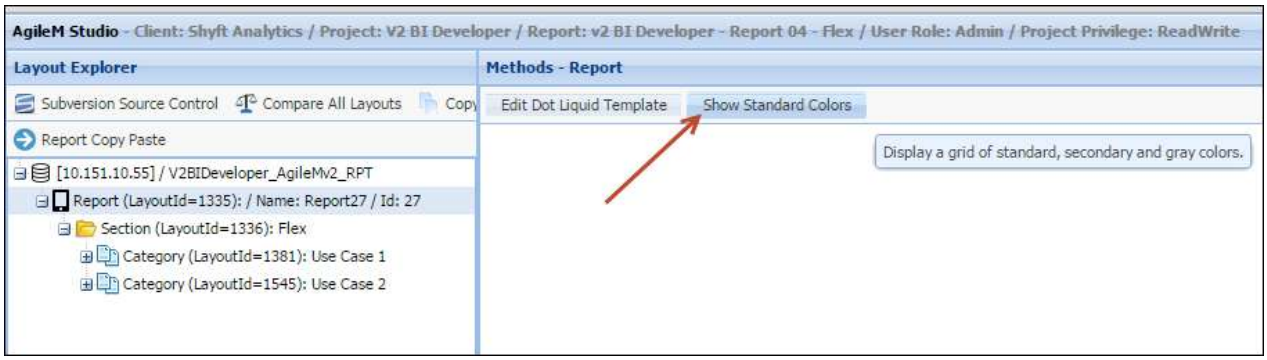
- This file must be in .png format.
- The size of the image should be no greater than 310px x 130px.
- The background should be transparent

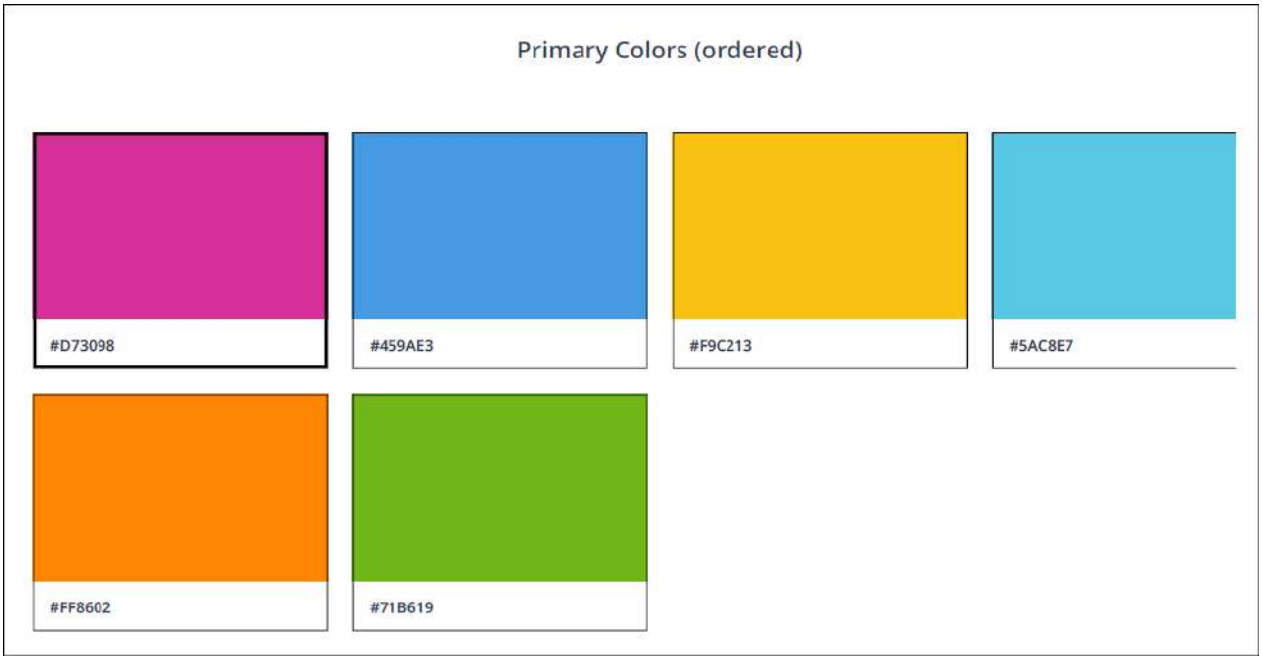


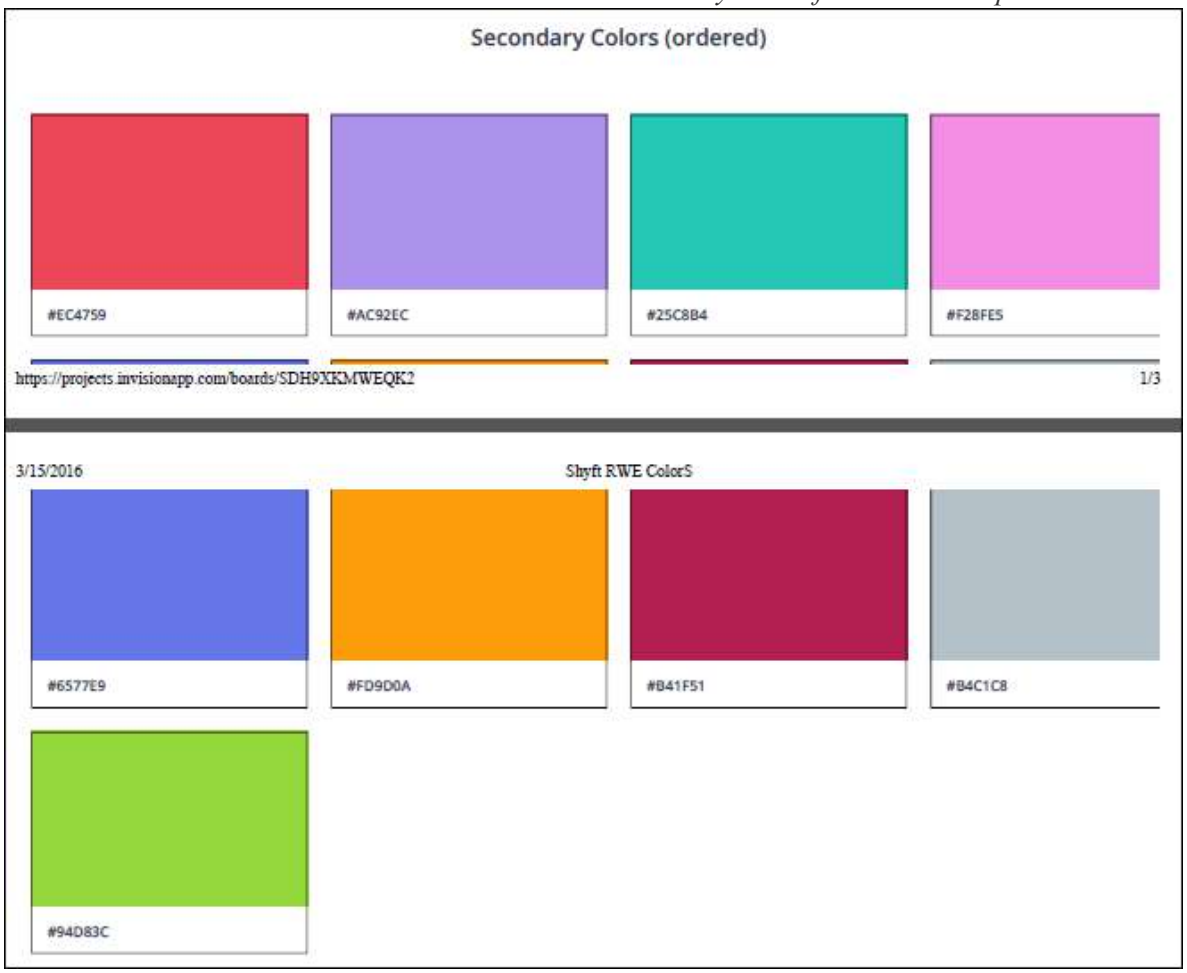
Recommended Colors

When selecting colors for display (for example, chart lines, bars, check marks, etc) it is best to use selections from the following color chart. This color chart can be found in the departmental documents folder as follows: Framework→AgileMv2→Analytic Platform Color Codes.pptx. (See a discussion regarding departmental documents here).

Also note that these colors and their hex values are selectable from within AgileM Studio.



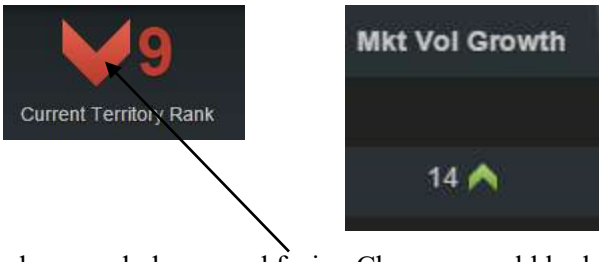



Secondary Colors (ordered)
#EC4759
#AC92EC
#25C8B4
#F28FE5
https://projects.invisionapp.com/boards/SDH9XKMWEQK2
1/3
3/15/2016
Shyft RWE ColorS
#6577E9
#FD9D0A
#B41F51
#B4C1C8
#94D83C


Gray / Dark Colors
#F3F4F6
#E7E9ED
#929BAD
#656D7D
#4F5769
#39404E
#313742

Icon Classes

The v2 application supports a wide range of icons for display. These icons are frequently used in KPI's and tables. For example:



Code for the large, red, downward facing Chevron would look something like the following:

```
SELECT
        layoutID = 1629
       ,HTMLTemplate = (SELECT Template
                        FROM   tblCfgDLHtmlTemplate
                        WHERE  TemplateID = 29)
       ,Value = '<font color="#FF0000">' +
       [dbo].[udfFormatDECtoVARCHAR](AverageReversalRate,0,0) +
       '  <span class="icn detail-progress decline"></span></font>'
```

The code that actually displays the Chevron is:

```
class="icn detail-progress decline"
```

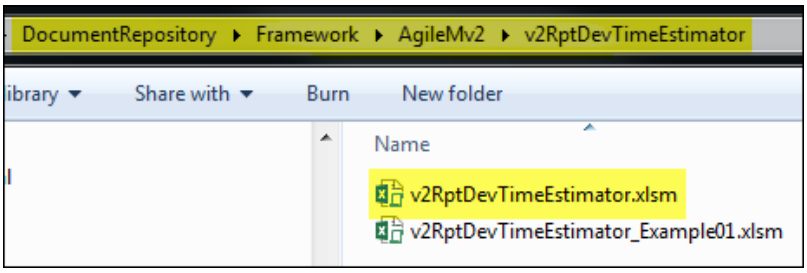
The AP Developer web site demonstrates each icon class – Report 01, Tables Category 1:

Icon Description	Icon Value
Red Flag	
Yellow Flag	
Green Flag	
Blue Flag	
Chevron growth	
Chevron decline	
Red circle	
Yellow circle	
Green circle	

v2 Report Development Time Estimator

There are a variety of business benefits in being able to estimate the amount of time required to implement a v2 report. Accurate estimates assist in personnel planning, company profitability, setting accurate deliverable dates and developing trust with BA's.

An Excel workbook has been developed to assist the Solutions Developer in providing reasonably accurate estimates. The file is named: v2RptDevTimeEstimator.xlsm and is located in the centralized departmental repository as noted below:



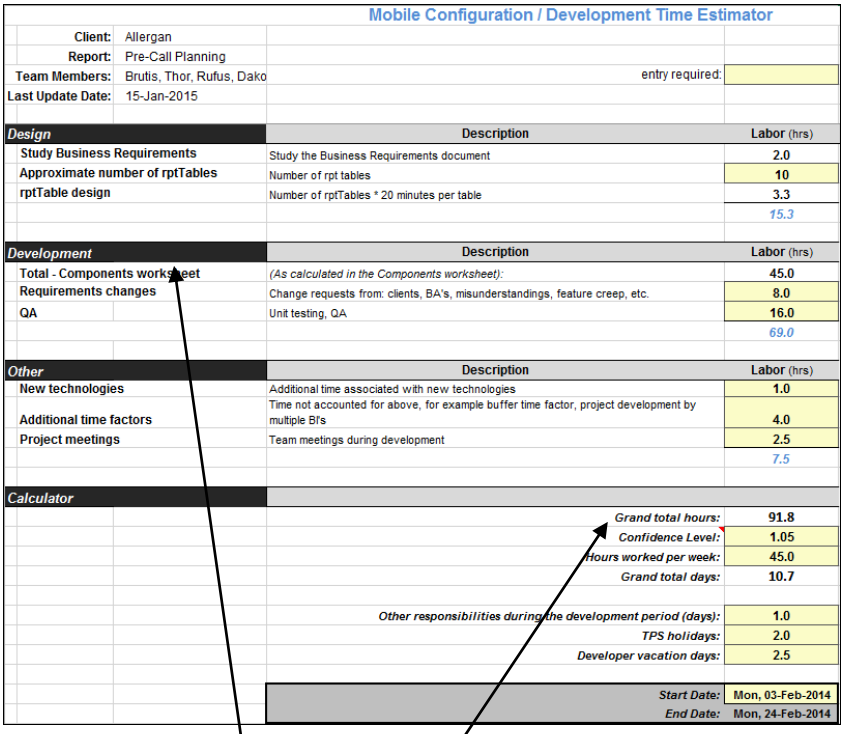
The workbook has two worksheets.

1) **Report worksheet**. This worksheet records some overall time factors in the development process – for example, studying the business requirements document and designing the rpt tables.

17) **Components worksheet**. This worksheet records the time associated with the development of each chart, table, freeform, etc.

Following are some brief notes on using the estimator.

Report Worksheet



Mobile Configuration / Development Time Estimator

Client:	Allergan
Report:	Pre-Call Planning
Team Members:	Brutis, Thor, Rufus, Dako
Last Update Date:	15-Jan-2015

entry required:

Design	Description	Labor (hrs)
Study Business Requirements	Study the Business Requirements document	2.0
Approximate number of rptTables	Number of rpt tables	10
rptTable design	Number of rptTables * 20 minutes per table	3.3
		15.3

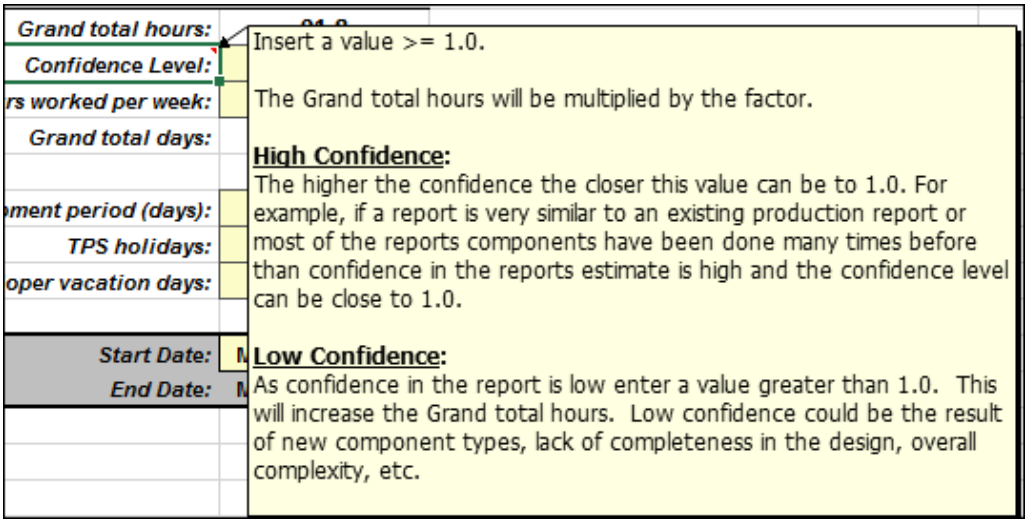
Development	Description	Labor (hrs)
Total - Components worksheet	*(As calculated in the Components worksheet):*	45.0
Requirements changes	Change requests from: clients, BA's, misunderstandings, feature creep, etc.	8.0
QA	Unit testing, QA	16.0
		69.0

Other	Description	Labor (hrs)
New technologies	Additional time associated with new technologies	1.0
Additional time factors	Time not accounted for above, for example buffer time factor, project development by multiple BI's	4.0
Project meetings	Team meetings during development	2.5
		7.5

Calculator	
Grand total hours:	91.8
Confidence Level:	1.05
Hours worked per week:	45.0
Grand total days:	10.7
Other responsibilities during the development period (days):	1.0
TPS holidays:	2.0
Developer vacation days:	2.5
Start Date:	Mon, 03-Feb-2014
End Date:	Mon, 24-Feb-2014

Entries are required for all cells with a yellow background. Most entries are self-explanatory. The value located in the **Total – Components worksheet** row is a calculated value that is automatically entered from the Components worksheet.

Note the row titled Confidence Level. This provides an optional ability to put a "risk" factor on the estimate. Mouse over the cell to observe notes associated with this entry.

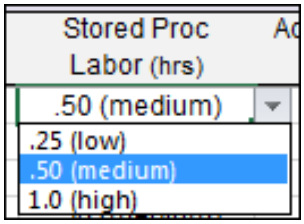


Components Worksheet

A	B	C	D	E	F	G	H	I
			Mobile Configuration / Development Time Estimator					
Client:	Allergan							
Report:	Pre-Call Planning							Calculate
Team Members:	Brutis, Thor, Rufus, Dakota							
Last Update Date:	15-Jan-2015							
Section	*Configuration - AgileM Studio Entry*		*Development - Stored Procedure Development Time*					
	Category Name	Config Labor (hrs)	**Component Name**	Layout Type	xType	Stored Proc Labor (hrs)	Additional Stored Proc Labor (hrs)	Justification / Notes
How Am I Doing	**Product Overview**	.50 (medium)	**Geography Filter**	Filter	all filter types	.25 (medium)		
			Product Filter	Filter	all filter types	.25 (medium)		
			KPI 1	Component	freeform/dot liquid template	.50 (medium)		
			KPI 2	Component	freeform/dot liquid template	1.0 (high)		
			KPI 3	Component	freeform/dot liquid template	.25 (low)		
			Market Share	Component	d3chart	1.0 (high)		
			Product Trend	Component	d3chart	1.0 (high)	2.0	complex data

The Components worksheet is designed to assign a time estimate to each Component in the report. Time estimates are made for configuring the report in AgileM Studio and writing the stored procedure for each Component.

Note that when entering time estimates for AgileM Studio configuration here and when entering Component stored procedure efforts here that a drop down combo box will appear where you may indicate a low, medium or high effort level, for example:



AgileM Studio Configuration

Enter each Category here, and provide a time estimate for how long it will take to make AgileM Studio entries for the single Category.

Stored Procedure

For each Component, estimate the amount of labor required to write the stored procedure. Note that pre-determined low, medium and high values are built into the estimator tool based on the type of Component.

Additional Stored Proc Labor

If you believe the pre-established labor hours are not sufficient, then enter more time. However, you should be prepared to defend the additional time.

Calculate button

When done, click Calculate. The hours will be totaled automatically and transferred to the Report worksheet.

Safari Set-up

Analytics Platform is supported on the iPad browser Safari. For AP to work, "Block Cookies" should always be set to "Always Allow". This setting can be found in Settings → Safari and scroll down to Privacy & Security.

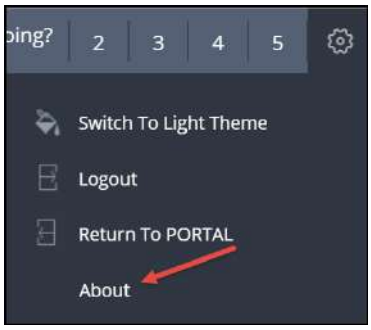
Auth0 Set-up

Auth0 is a tool that supports users accessing AP via single sign-on. Details regarding its installation can be found here. Note you will need to copy/paste this URL into your browser, directly clicking will not work.

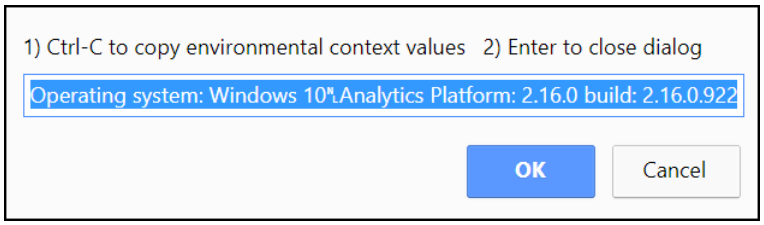
https://trinitypharmasolutions.atlassian.net/wiki/spaces/AP/pages/102763160/Auth0+Client+Setup

Environmental Context Information

When the domain for a user's e-mail is: @shyftanalytics.com, the Sprocket Menu supports an 'About' menu selection which provides a wide range of information about the context under which the application is being executed.



When the 'About' selection is taken the following dialog displays:



Enter Ctrl-C to copy the information in the dialog box. This information may now be pasted into an editor and will appear as:

SHYFT MOBILE QUERY TOOL

Query Tool Background

The Shyft Mobile Query Tool (QT) is an add-on to the Shyft Analytic Platform product. While QT is fully integrated into the Analytic Platform user interface it is in fact a separate product with its own set of SQL tables and web application.

QT requires a minimum of AgileMv2 v2.1.0.

QT is an ad-hoc reporting tool that displays data from a single table. The user can select which columns to display and then both filter and sort the rows. The flow of the user interface is the following:

1) Select a single reporting table (referred to as a Profile).
2) Select the specific columns, from the single table, to display.
3) Filter the rows.
4) Sort the rows.
5) A grid of data is presented along with selected summary values. The user can optionally have the grid e-mailed as an Excel file.

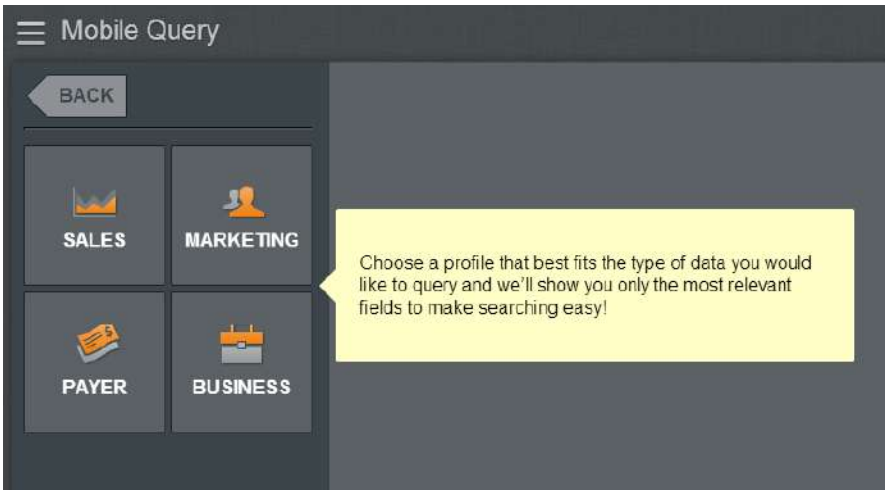
QT is not, yet, located in the APDeveloper mobile web site. Please see an example of QT in the Shyft Demo v2 mobile site:

- http://10.151.10.55/agilemdemo/
- Username: tpsdemo
- Password: D3m0u$er

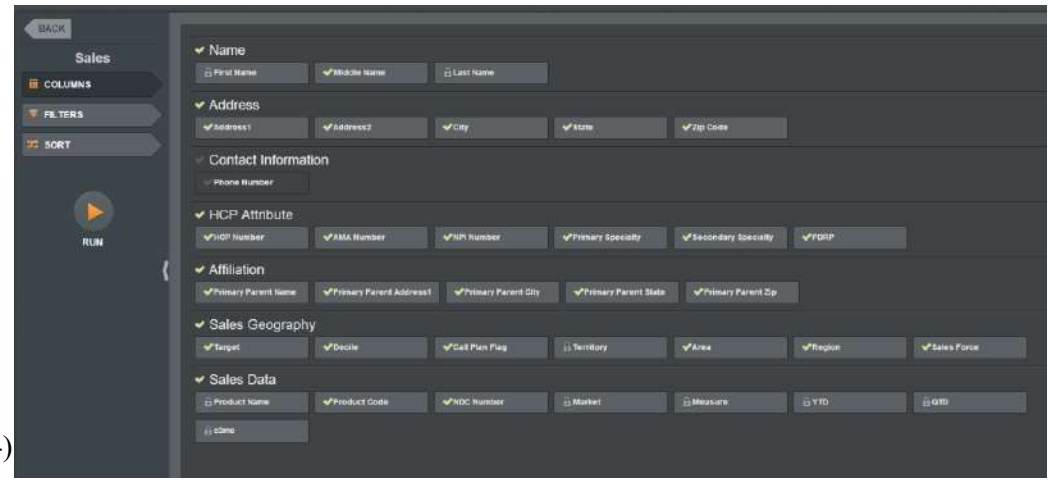
Following is a set of screen snap shots of the user interface:

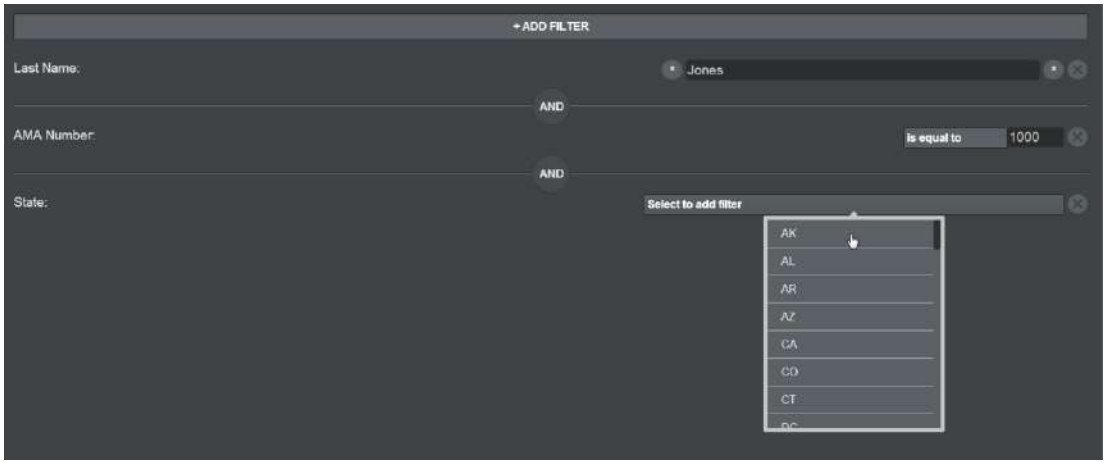
1) **Activate QT from within Analytic Platform.**

2) **Select a QT Profile** (which will equate to a single reporting table or view).

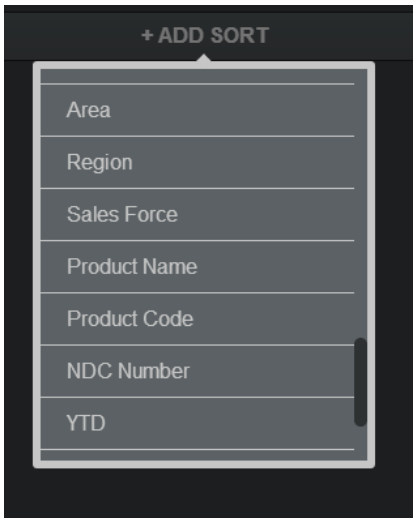


3) **From the Profile, select the columns to display**. (Note that the columns are in "Column Groups").

4) 

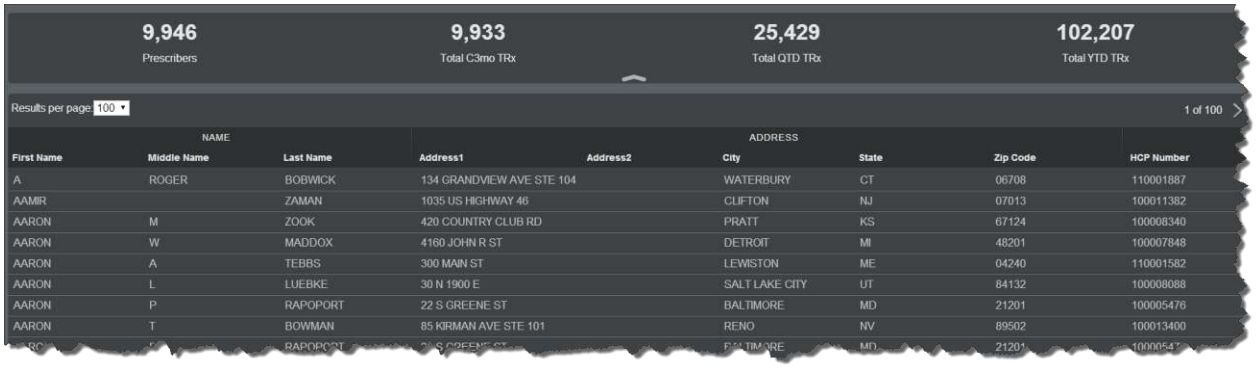


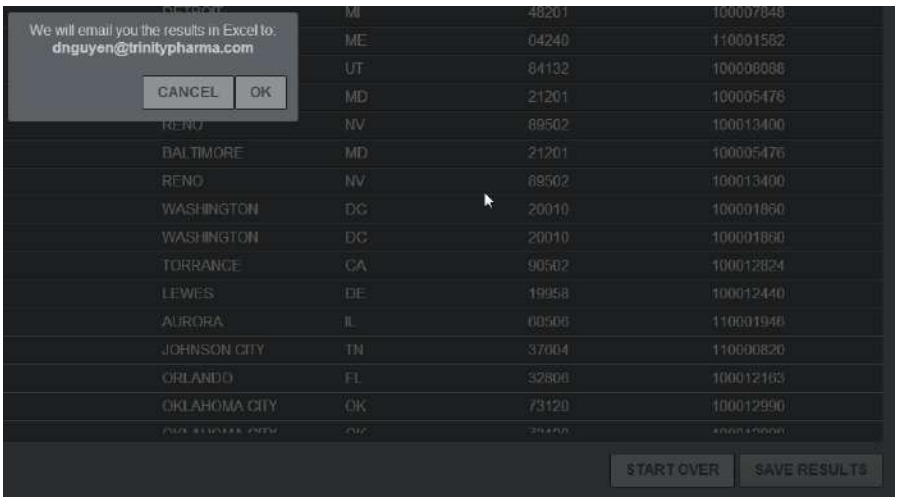
5) **Select the Sorting to apply.**



6) **Run the selections**.

7) **Observe the results and optionally e-mail the grid in an Excel file.**



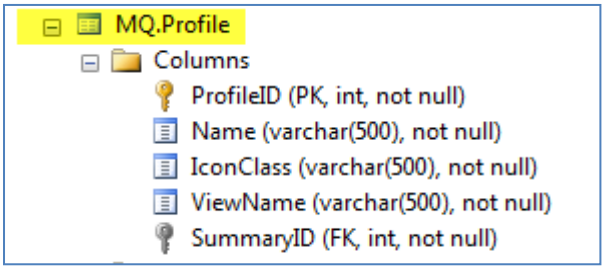


Query Tool Tables

All QT tables exist in the MQ schema of Agile Mv2 APP database. Following are the tables:

MQ.Profile

Contains one row for each Profile.

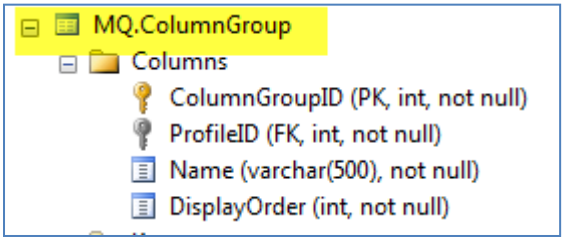


- IconClass: controls the picture on the initial user dialog. The values can be sales, market, payer, business which result in the following buttons.

- ViewName: the single table or view associated with the Profile

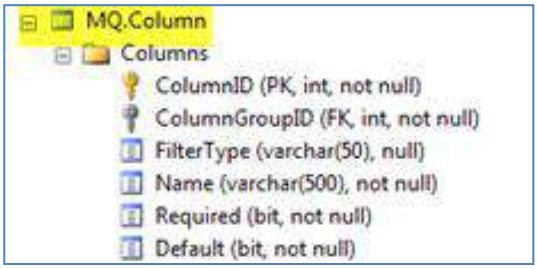
MQ.ColumnGroup

Contains one row for each grouping of columns. Below Name is a group.



MQ.Column

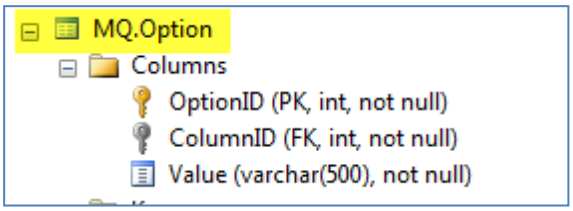
Contains one row for each column in the Profile.

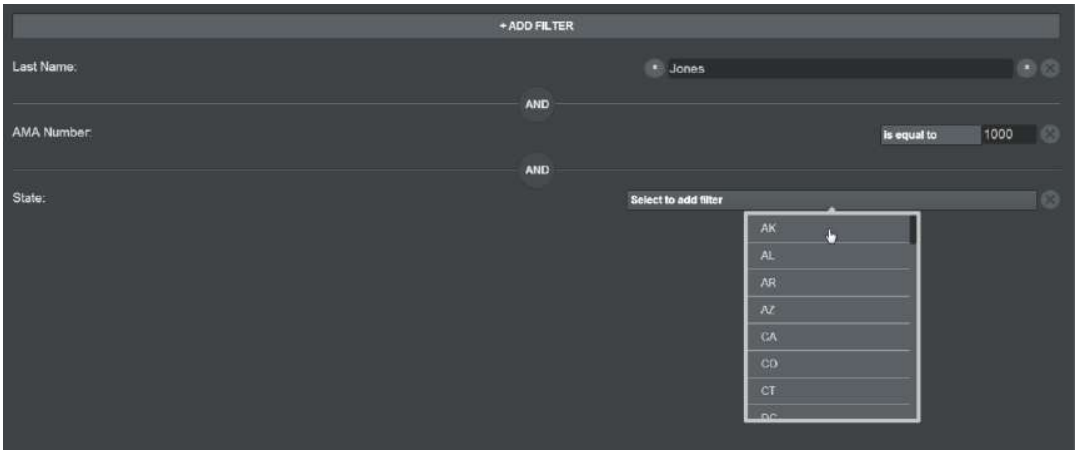


- FilterType: supports the following three filter types: 'text', 'numeric' and 'multi'.
- Name: the name of the column in the view and the name as the user will see it
- Required: controls if the column name must be selected for display in the final grid and when set to true will present a "lock" next to the column name (see above)
- Default: controls if a column will initially be selected for display in the final grid and when set to true the column will have a green check mark (see above)

MQ.Option

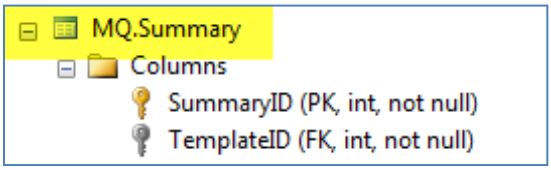
For columns of type 'multi' MQ.Option will have one row for each option available in a drop down filter query. For example, a State column could have 50 MQ.Option rows, one for each state.





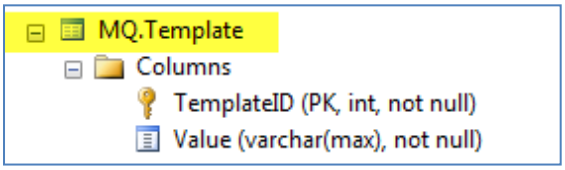
MQ.Summary

One row per Profile and provides a mapping from MQ.Profile.SummaryID to MQ.Template.



MQ.Template

One row per Profile.



- Value: contains a Dot Liquid template that controls the display of the summary panel

MQ.Template.Value for the above summary is:

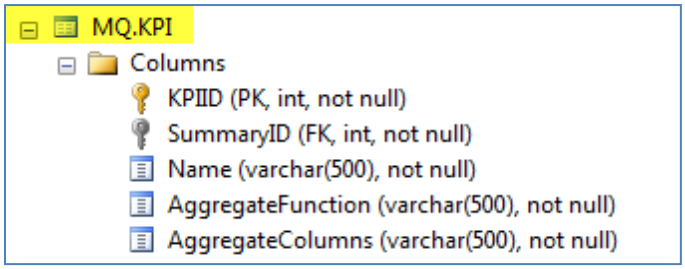
```
<DIV CLASS="pure-g">
<DIV CLASS="pure-u-1-4"><P CLASS="kpi-callout">{{PRESCRIBERS| NUMBER: 'N0'}}</P> <P CLASS="kpi-title">Prescribers</P>  </DIV>
<DIV CLASS="pure-u-1-4"><P CLASS="kpi-callout"> {{TOTALC3MO | NUMBER: 'N0'}}</P>  <P CLASS="kpi-title">Total C3mo</P> </DIV>
<DIV CLASS="pure-u-1-4"><P CLASS="kpi-callout">{{TOTALQTD | NUMBER: 'N0'}}</P>  <P CLASS="kpi-title">Total QTD</P> </DIV>
<DIV CLASS="pure-u-1-4"><P CLASS="kpi-callout">{{TOTALYTD | NUMBER: 'N0'}}</P>  <P CLASS="kpi-title">Total YTD</P> </DIV>
</DIV>
```

Note that the code: <DIV CLASS="pure-u-1-4"> controls the number of data points in the KPI header. In this case the "4" supports the four data points above. If a fifth data point was needed than all cases of <DIV CLASS="pure-u-1-4"> would need to change to <DIV CLASS="pure-u-1-5">. Also note that class values must be in all lower case (as they are above, for example: 'pure-u-1-4').

Query Tool templates can be edited in any text editor including the Dot Liquid Template editor window in AgileM Studio.

MQ.KPI

One row for each summary value in the summary panel.



- Name: A free text string as it will appear in the Dot Liquid template in MQ.Template.Value.
- AggregateFunction: QT supports the following functions: count, sum, average, marketshare and marketsharechange.
- AggregateColumns: The names of the columns required to calculate the function. When more than one column is required enter the column names as comma separated values:
  - For count, sum and average, supply one column name.
  - For marketshare supply two column names (comma separated):
    - Product volume column
    - Market volume column
  - For marketsharechange supply four column names (comma separated):
    - Current product volume column
    - Current market volume column
    - Prior product volume column
    - Prior market volume column

For example:

	KPIID	SummaryID	Name	AggregateFunction	AggregateColumns
1	3	1	prescriberCount	count	First Name
2	4	1	sum3Mo	sum	c3mo
3	5	1	sumQTD	sum	QTD
4	6	1	sumYTD	sum	YTD

Note that a column can be used in the KPI summary row even if it is not a column in detail row.

Query Tool Implementation

There are four primary implementation tasks:

1) Create a View from the actual reporting table
2) Create the MQ schema table
3) Create a webadmin user
4) Populate the MQ schema tables
5) Install the QT web site

One additional task may be to create users that can access the system. Note that for Query Tool only access the following tables must be populated: tblUser and tblCfgReportAssignment. Entries in these two tables will allow the user to logon into the v2 system and access the Query Tool even if they do not have access to any reports.

1-Create a View from the reporting table

The required reporting table may have two issues: one, there may not in fact be a single table with all the client requested columns and two, the column names may not be user friendly (for example, qtrSales vs. Quarterly Sales). Both problems may be solved by creating a View. The View could be a straight Select from an rpt table or could use Joins to get all required columns. Also, the column names should be made "user friendly".

Also note in creating the View that columns may be included in the View that are not placed in the MQ.Columns table. These columns, which would not be visible to the user for inclusion in the final grid could be used in MQ.KPI for inclusion in the summary.

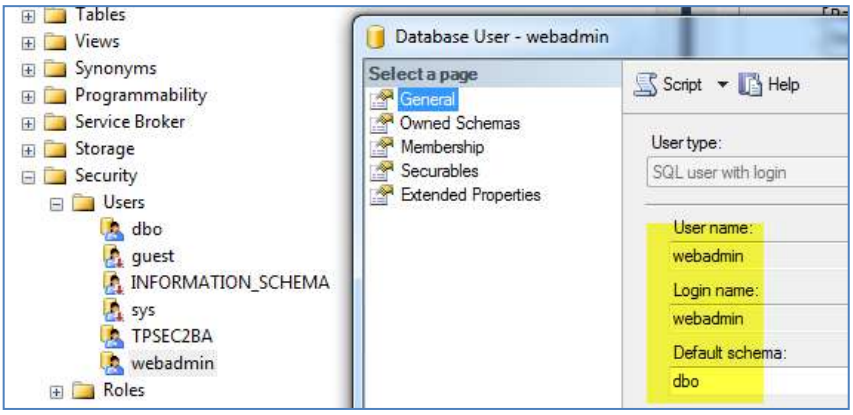
2-Create the MQ schema tables

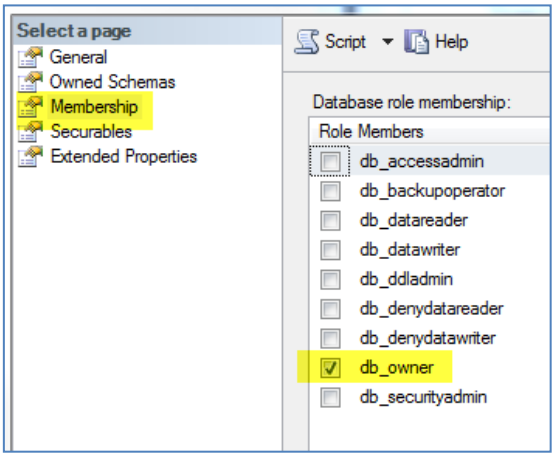
Located in the QT release folder will be a file named: database.sql. This file will create the MQ schema, all required QT tables and their constraints. Execute this file against the database that will contain the MQ tables.

Note that the selected database would generally not be the v2 RPT database or any database whose name changes with each reporting period. It may be appropriate to establish an entirely separate database.

3-Create a webadmin user

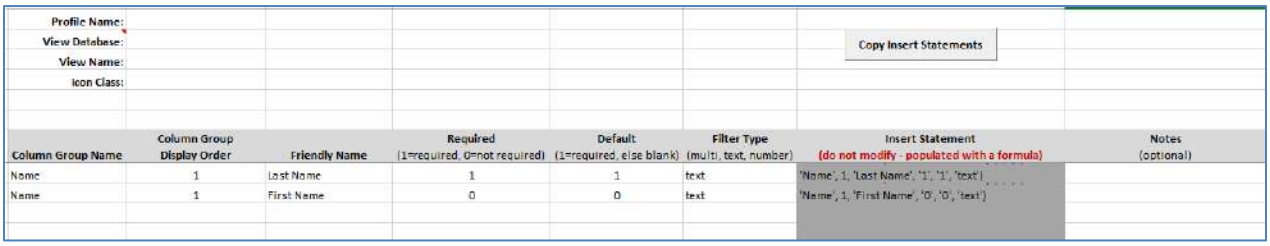
Create a webadmin user on the database that contains the MQ tables. If webadmin exists, verify it has the following Membership Role.





4-Populate the MQ schema tables

The following Excel worksheet is designed to ease the effort of populating the normalized MQ schema tables. (The sheet is in the Excel workbook named PopulateQTTables.xlsm and is located in the SVN Document Repository → Framework AgileMv2 → QueryTool).



Working in conjunction with the BA's populate this worksheet. **Do not add any new columns or significantly alter this worksheet – just add one row of data for each column and populate the values in the header**.

Note that columns that are set as Required=1 must also be set as Default=1.

Once populated, click the Copy Insert Statements button and paste the pre-built SQL Insert statements into rspPopulateQTTables.sql. Locate the comment that reads: "Place Insert statements for MQ.TableData here".

```
----------------------------  Place Insert statements for #qtTableData here:
        INSERT INTO #qtTableData VALUES ('Payer Mstr','ASTELLAS_PAYER_MASTER','vwRptSharamCAD_QT','payer', 'Plan Information', 1, 'Plan Number', '1', '1', 'text')
        INSERT INTO #qtTableData VALUES ('Payer Mstr','ASTELLAS_PAYER_MASTER','vwRptSharamCAD_QT','payer', 'Plan Information', 1, 'Plan Name', '1', '1', 'text')
        INSERT INTO #qtTableData VALUES ('Payer Mstr','ASTELLAS_PAYER_MASTER','vwRptSharamCAD_QT','payer', 'Plan Information', 1, 'Plan Type', '', '1', 'multi')
        INSERT INTO #qtTableData VALUES ('Payer Mstr','ASTELLAS_PAYER_MASTER','vwRptSharamCAD_QT','payer', 'Plan Information', 1, 'Plan Sub Type', '', '1', 'multi')
        INSERT INTO #qtTableData VALUES ('Payer Mstr','ASTELLAS_PAYER_MASTER','vwRptSharamCAD_QT','payer', 'Plan Information', 1, 'Plan Sub Type Code', '', '', 'multi')
        INSERT INTO #qtTableData VALUES ('Payer Mstr','ASTELLAS_PAYER_MASTER','vwRptSharamCAD_QT','payer', 'Plan Information', 1, 'State of Operation', '', '1', 'text')
        INSERT INTO #qtTableData VALUES ('Payer Mstr','ASTELLAS_PAYER_MASTER','vwRptSharamCAD_QT','payer', 'Plan Information', 1, 'Blues Affiliation Flag', '', '', 'multi')
        INSERT INTO #qtTableData VALUES ('Payer Mstr','ASTELLAS_PAYER_MASTER','vwRptSharamCAD_QT','payer', 'Plan Information', 1, 'CBM Number', '', '', 'text')
```

Execute the stored procedure. All QT tables, except MQ.KPI and MQ.Template. will be populated automatically. The KPI and Template tables will need to be manually populated.

MQ.Option is also populated for all multi filters. This is done by locating distinct values in the data table. **Note this requires that the data table must already be populated for this stored proc to be fully functional**. Also note that the data base developer may need to perform similar functionality each reporting period.

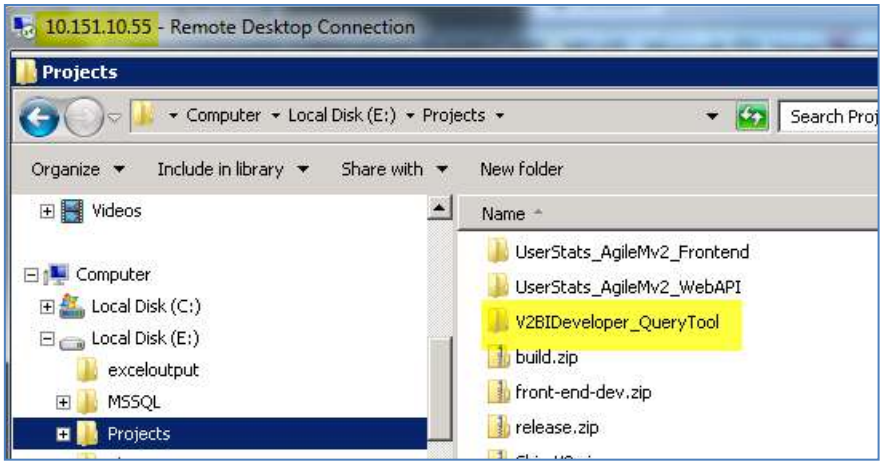
For example, if [Product Name] has a Filter Type of 'multi', and the data table contains [Product Names] of: 'Product1', 'Product2', and 'Product3' than MQ.Options would contain three rows that would be listed when the users filters on [Product Name]. If the list of [Product Names] varies each reporting period than the rows for [Product Name] in MQ.Options will need to be modified.

5-Install the QT Web site

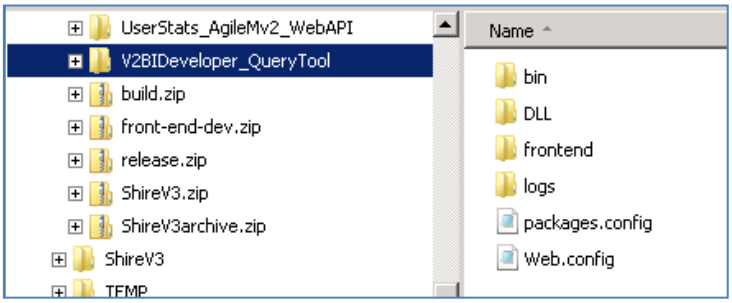
1) **Create the Query Tool Web Site**

First, create a query tool directory on the same server as the Analytic Platform site from which the query tool will be run. For example, since the v2 BIDeveloper v2 site is on 10.151.10.55, than create a query tool directory as follows, also on 10.151.10.55:

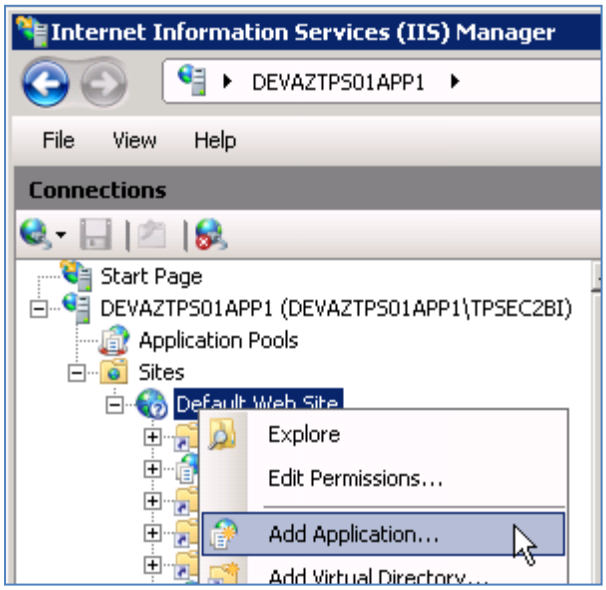
Follow the naming convention of <Project Name>_QueryTool.

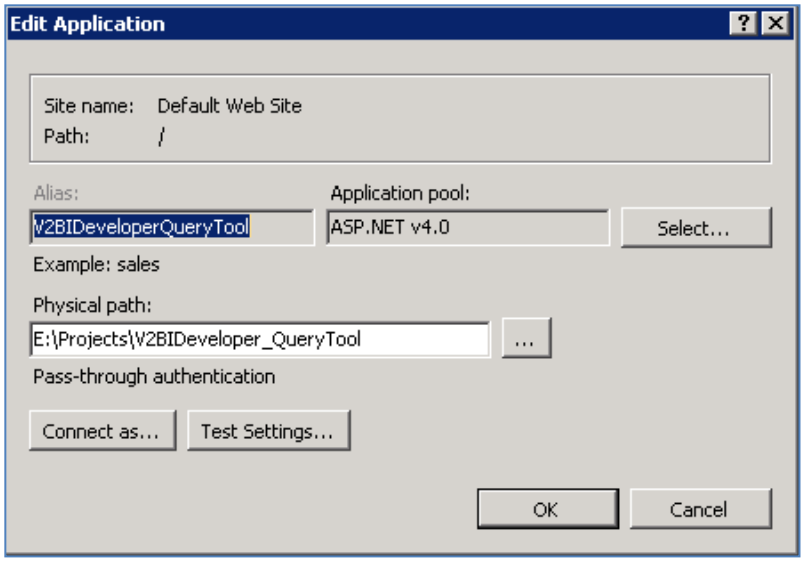


Second, get the query tool files from the Releases folder, and copy to the newly created directory.



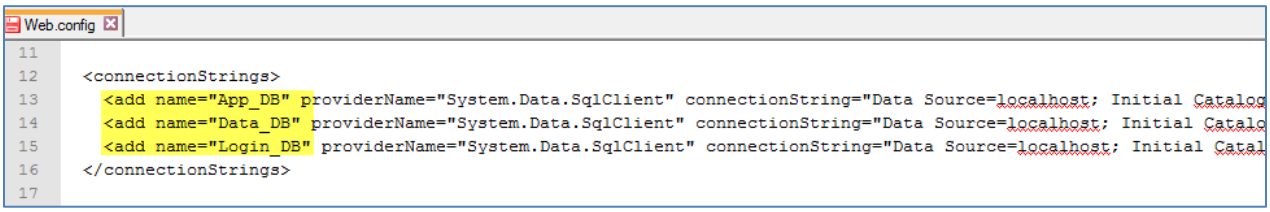
Third, add a new application through IIS.





2) **Adjust Web.config**

In Web.config of the QT web site, set the connection strings for App_DB, Data_DB and Login_DB.



App_DB = that database that contains the MQ schema data tables (frequently the v2 APP database).

Data_DB = that database that contains the MQ.Profile.ViewName Views

Login_DB = the APP database of the AgileMv2 system

18) **Adjust the APP table tblAppConfig**

See details here for adjustments to the APP tblAppConfig table that support the Query tool.

Query Tool Troubleshooting

A log file exists in the QT web site directory that provides information that can be helpful in debugging an issue. Among other information captured is the exact SQL command that is executed when producing the final output. This command could be run interactively to help resolve an issue.

The file is located here: