Paper 2818-2018

BI Centralization: The benefits of implementing a Report Encyclopedia for your organization

John Mattox, Magellan Health

ABSTRACT

In the world of Business Intelligence and Reporting, the standardization, maintenance/upkeep, accuracy, and reuse of reports across an organization are all key aspects to a well-groomed BI Solution - regardless of the technology. In a medium to large sized organization, it's highly likely that SAS is not the only reporting tool available to BI developers. As the sources of reports grow in number, it can become increasingly difficult to keep track of all of the reports and metadata associated with them. One solution to help solve this problem is to create a centralized catalog, or "Encyclopedia" to house all of this data. The report encyclopedia should serve as a method to accomplish the key aspects discussed above. By having a method to centralize report metadata in an organized manner for developers, clients, and users, the tool will help to educate users on the extent of reports offered by your team, and can promote the reuse or modification of reports and will alleviate much of the stress associated with requesting a report. This paper will explore how a central, connected tool such as a report encyclopedia can help the organization with all of these aspects, as well as how the metadata can be used to generate SAS report indexes after report creation.

INTRODUCTION

Within your organization, it can be a challenge to keep track of the full breadth of reports that a Business Intelligence or reporting team offers to the end-users. Many organizations go day-to-day with a large inventory of available reports that have not been organized for scale or re-use, and a result, resources have to be inappropriately utilized and assigned to develop a report that may have been previously created. If the reports were cataloged and presented in a user-friendly manner, then you could enable the end-user to search for, select and request a report essentially ‘by name’ that will satisfy their current need. In a modern company, it is very common to use multiple technologies in addition to SAS to develop reports such as Cognos, QlikView, Tableau, etc. With such a variety in reporting platforms, it is difficult to tell what already exists, and what should actually be “new development”. This paper explores the process you would need to follow and the benefits that you would receive by implementing a report encyclopedia in your organization.

GETTING STARTED

Developing a report encyclopedia within you organization is not an easy task. It will require hard work and determination, but there are many options for how to implement a tool such as this. This paper will break it down into steps that will make it easier to get your foot in the door. The sections below highlight important tasks and explain what you need to keep in mind to be successful.

DEFINE YOUR REQUIREMENTS

As with any well planned project or task, establishing a set of requirements is essential. First you must ask yourself what it is exactly that you want the desired state/outcome to be. What problem do you need to solve? Within our organization, the Business Intelligence team sought to be more that just a *reporting* team, but rather a *solutions* team. While it is still our job to develop reports to support our end-users, we needed to alleviate some of the pressure from the influx of report requests so we could focus on new development projects. As a greater number of customers and end users make requests of the BI team, the focus of BI developers has shifted to deliver results as quickly as possible, which did not align with our new strategy and goal. At the time, no mechanism existed with which to provide users with the full range of reports and capabilities available, and we needed to get out of that cycle of inefficiency.

In this particular use case, the requirements have been defined as follows:

- Provide end-users with a rich set of tools and capabilities with which to be self-sufficient, by making BI capabilities available in an organized taxonomy, easily navigable, and with an intuitive interface and back-end automation for the delivery of the desired output or results
- Define report categorization (currently inconsistent) through a thorough analysis of report inventories and create Metadata for existing reports
- Design a process for future reports to be categorized such that the Report Catalog can be maintained as it grows.
- Develop the Report Catalog, facilitating new BI capabilities through the frequency of re-use of the existing business logic, leading to the creation of new opportunities for logical improvement and standardization.
- Support and drive BI team's goal to transition from a reporting service team into a BI solutions team, shifting the output of the team from individual report production to scalable and reusable information delivery solutions.
- Users should have the ability, in a central location, to see a report description and details, including a sample image of the report and its contents.
- If and when users encounter a report that would satisfy their need, they should be able to 'click' to request the report, with an established delivery mechanism then making the results available to them.
- If a user navigating the Report Catalog passes certain checks indicating their information need cannot be satisfied through self-service, they should be able to 'launch' into the creation of a new report request to trigger the creation of a request in JIRA Task Management tool.
- The ability to quickly and simply add new entries to the tool, which will immediately become searchable to the tool's end users
- This catalog of reports shall be searchable and be broken out by both client and functional area. For example an eligibility report for Idaho should be contained in both an eligibility section, and an Idaho section.
- Each report definition in the report catalog shall contain, Report Name/ID, Business definition of report, report technology (Cognos/SAS), input variables, input source, return fields, location of output, frequency/schedule of run, location of code, and sample outputs

SELECTING THE TECHNOLOGY

Choosing the right technology platform for your Report Encyclopedia is a vital step in the endeavor, and can be root cause of why a project succeeds or fails. Based on the requirements you have identified in the section above, you need to ensure that the technology of choice is able to support your business needs, is scalable, and is maintainable. Do you have the means to develop this solution in house or does it need to be presented to vendors as an RFP? Perhaps there is an off the shelf product that will serve your needs. Do you have staff on hand that is trained and able to support the tool? The fanciest and newest tool might not be right for your organization's needs.

In this use case, many options were considered before a decision was made. After multiple RFP responses were reviewed and none of them seemed to deliver the solution we were looking for, we ultimately decided to build the solution in house. The final technology of choice was an ASP.NET MVC web application, with a SQL Server database serving as the backend repository for the report catalog. ASP.NET MVC web applications are versatile and dynamic, and fit into our organization's environment well since it has the ability to integrate with the existing authentication using Active Directory and we had web development resources on-hand that we could allocate to this solution.

However, this was easier said than done, and we had to prove the idea by creating a proof of concept before we were given the resources. Using a macro embedded within Microsoft Excel, we created a functional and searchable prototype of the Report Encyclopedia. In the following example, we have

already designed the backend database structure and populated the tables with sample data. The following VBA module sample searches the database and returns the results all within the excel worksheet:

```
Sub report()

 If Worksheets("Sheet1").Shapes("btnReportID").ControlFormat.Value = xlOn
Then
     searchcol = "ReportID"
 ElseIf Worksheets("Sheet1").Shapes("btnDescription").ControlFormat.Value =
xlOn Then
     searchcol = "Description"
 ElseIf Worksheets("Sheet1").Shapes("btnCategory").ControlFormat.Value =
xlOn Then
     searchcol = "Category"
 ElseIf Worksheets("Sheet1").Shapes("btnSubCategory").ControlFormat.Value =
xlOn Then
     searchcol = "SubCategory"
 ElseIf
Worksheets("Sheet1").Shapes("btnSourceApplication").ControlFormat.Value =
xlOn Then
     searchcol = "SourceApplication"
 ElseIf Worksheets("Sheet1").Shapes("btnReportName").ControlFormat.Value =
xlOn Then
     searchcol = "ReportName"
 ElseIf Worksheets("Sheet1").Shapes("btnClientName").ControlFormat.Value =
xlOn Then
     searchcol = "ClientName"
 End If

searchtext = Range("txtSearch").Value
    With ActiveWorkbook.Connections("server_name BI_Reports
REPORT").OLEDBConnection
        .CommandText = "select a.ReportName, a.ReportID, a.Category,
a.SubCategory, a.Description, a.ReportOutput, d.SourceApplication,
b.clientname, a.repository, c.Frequency  from dbo.REPORT a inner join
client b on a.reportkey = b.reportkey inner join scheduling c on
a.reportkey = c.reportkey inner join [database] d on a.reportkey =
d.reportkey where " & searchcol & " like '%" & searchtext & "%'"
    End With
    ActiveWorkbook.Connections("server_name BI_Reports REPORT").Refresh

    Range("txtSearch").Activate

End Sub
```

In the code above, the series of IF statements determines the field in the database to search against based on the selection of a radio button on the excel worksheet. It combines this with the text entered in the search box to build out a SQL statement, and query the database. This VBA module is fairly simple, and while it does not quite meet the requirements outlined earlier, it did however prove the concept to the stakeholders to push the project forward, which was the goal of this exercise.

Figure 1 below shows the main worksheet of the prototype encyclopedia using excel:

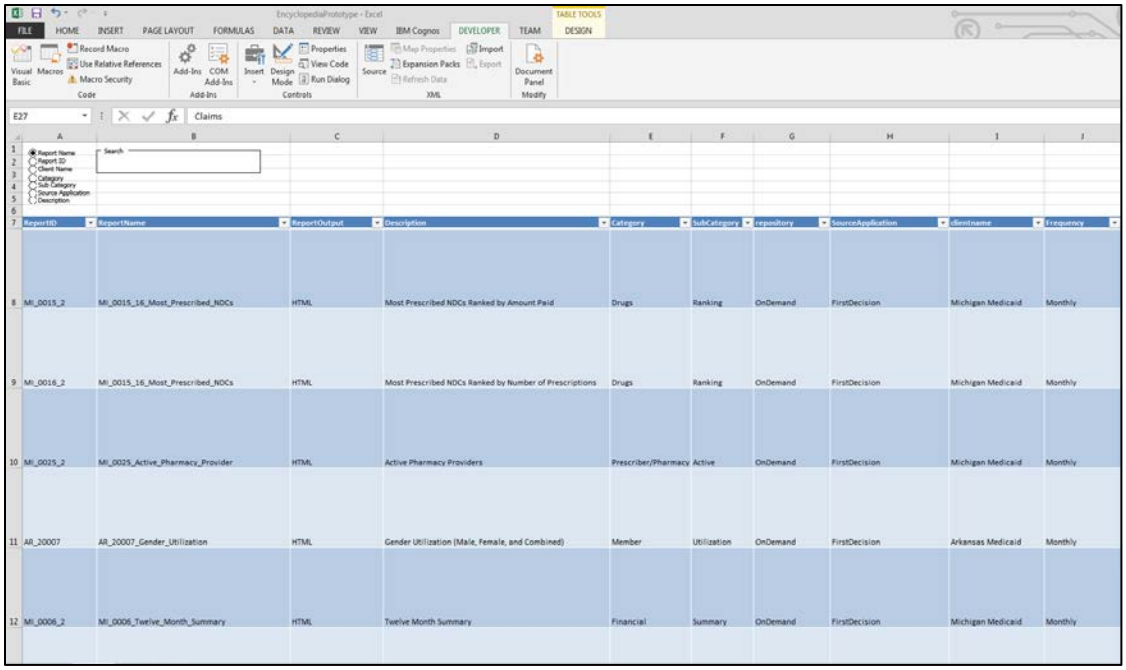

Figure 1. Excel Report Encyclopedia Prototype

This example also goes to show that there are many possibilities for designing a solution such as a report encyclopedia. Depending on your requirements and available resources, your team can be creative when it comes to the design of the solution, and the excel prototype above could potentially solve for your needs.

DESIGNING THE SOLUTION

The sections below describe the high level design processes for the report encyclopedia solution.

DATABASE DESIGN

A web application needs to have a well structured database to use as part of the backend architecture. Depending on your solution, the particular database vendor you use will likely be tied to the design of the application and the technology used. For example, ASP.NET applications perform best when using SQL Server as the database platform. The database design is centralized around the core of the solution: a report. Following a star schema model, the report table acts as the FACT table, recording the lowest level of detail for a single report entity. Surrounding the Report FACT are dimension tables that house the descriptive attributes associated with each report. Figure 2 below shows the ERD (Entity Relationship Diagram) for the report encyclopedia database:

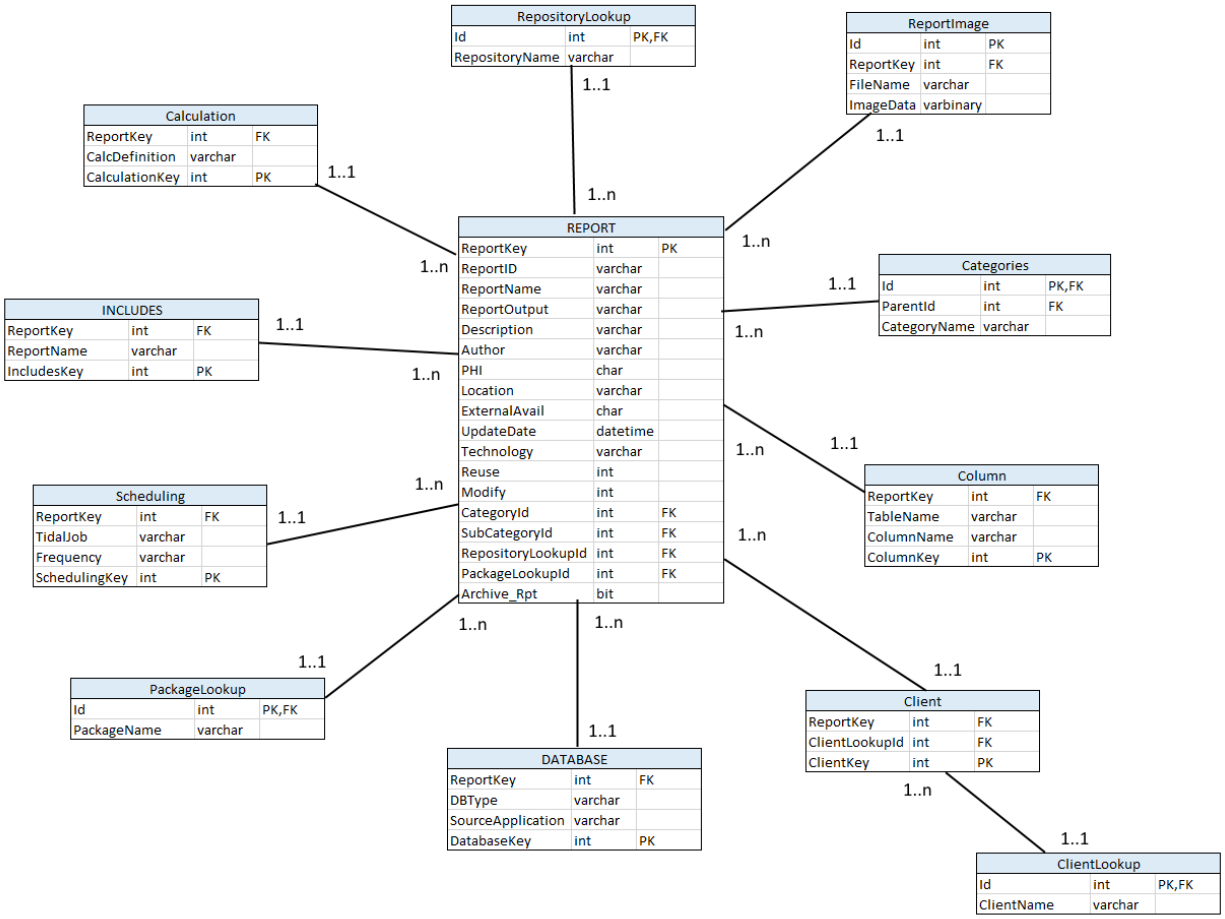

Figure 2. Database ERD

The tables in the database were designed in a way to incorporate all types and sources of reports, not only those produced from SAS programs. While we attempted to use naming conventions that could serve multiple purposes, there were some tables and fields that only apply to one BI reporting tool. For example, the table INCLUDES in the diagram above was designed to store the list of programs that are called from within the parent SAS program with a %include reference, and therefore is only applicable to SAS reports. The lookup tables in the diagram serve as a means to store a distinct list for that dimension, and are linked to an interactive control such as a drop-down box or radio button on the front-end of the web application. The values from the table populate the control when inserting and updating records. For further details, Table 1 below highlights the purpose of each table in the database:

Table Name	**Purpose**
Report	The central table in the schema. Contains general descriptive metadata for each record in the report encyclopedia.
Client	Contains the keys associated with each report and client.

Table Name	Purpose
ClientLookup	Distinct lists of clients and the associated look-up ID linked to the client table.
Database	Contains the database type and source application that is associated with each report
Column	Contains the name of the tables and columns that the data is pulled from to produce each report
Calculation	Contains the definition of calculations used within each report
Includes	Contains information associated with ***%include*** statements found within SAS programs
Scheduling	Contains information associated with the scheduling and frequency of each report
PackageLookup	Contains a distinct list of report packages.
Categories	A distinct list of categories and subcategories, stored with a parent ID and sub IDs to form the hierarchy.
ReportImage	Contains the .JPG sample image of the report stored as VARBINARY
RepositoryLookup	Distinct list of report repositories, for example in our organization, Cognos reports are stored in the Cognos portal, and SAS reports are routed to another internal portal named OnDemand.

Table 1. Descriptions of the tables in the Reports Database

USER INTERFACE

The user interface plays a major role in how well the solution is adopted amongst the organization. The interface must be easy to navigate and understand, both for the end-users who will be searching for and requesting reports, and for the developers, who will be maintaining and entering records for report entries. With an ASP.NET web application, the entire front-end interface is highly customizable with HTML, CSS, and MVC Views. The user interface for this solution is broken down into a few main views. The Home page, the advanced search, the report list view, the report detail view, and the report modification view. The first view is the home page and can be seen in Figure 3 below:

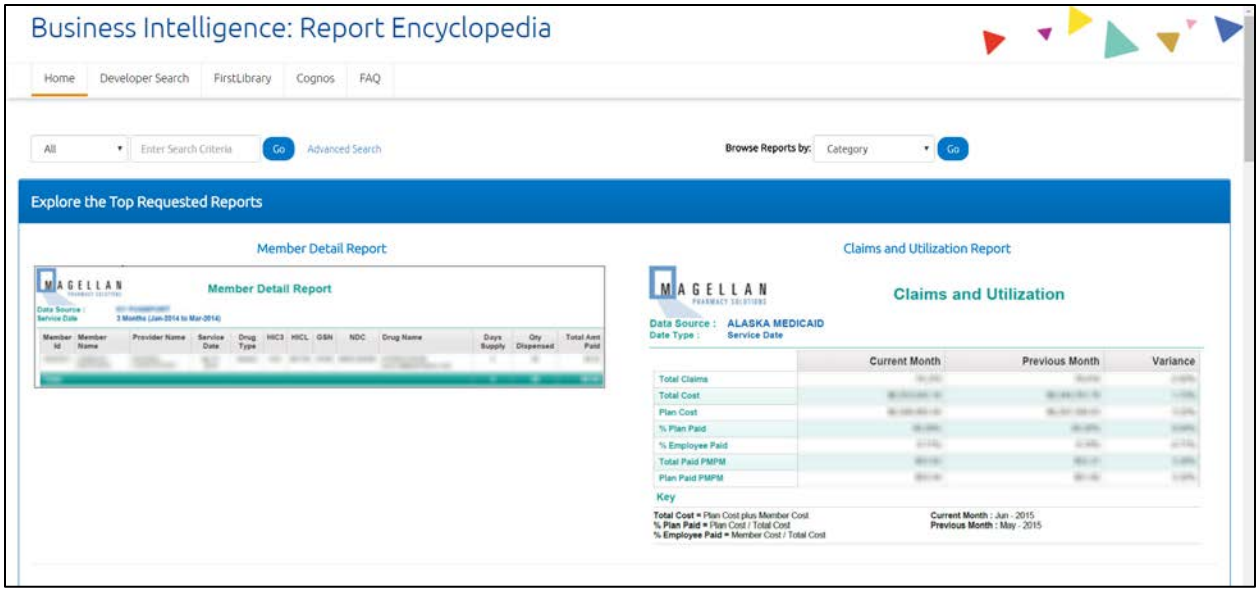

Figure 3. Home Page View

The home page of the website sets the first impression to the end user. In a well designed website, the user should almost instantly be able to digest what they are viewing on the screen, with quick access to site navigation and links to important details you would like to highlight, in this case the top requested reports. In our example above, you can see the navigation bar at the top, which will appear on all pages, gives immediate access to the Home page, and the advanced search page. The remaining links direct the user to the report repositories where you actually access the physical reports (FirstLibrary, for SAS reports, and the Cognos Portal). The FAQ link downloads a document to the user's computer with helpful tips. Below the top navigation bar is a search feature, where users can search for reports using the Report Name, Report ID, Description, or all fields. User search criteria can also be modified afterwards on the results page. From here you will be redirected to the results page, with the first 10 entries being displayed on page 1 in the results list. Figure 4 below shows the report list view:

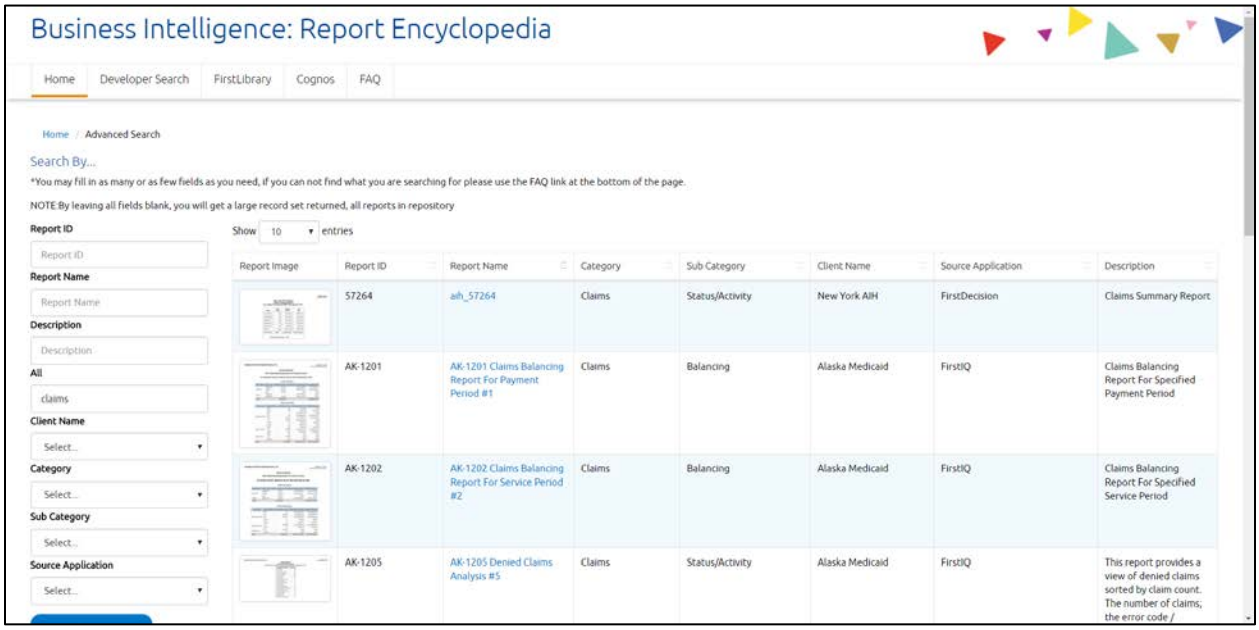

Figure 4. Report List View

The report list view displays the search results based on the criteria entered from the home screen Users can further refine the results by utilizing the features of the Advanced Search. The advanced search can be used to find more specific results, with more options available on the left side of the screen to narrow the search. Once the changes are made to the criteria, simply hit apply and the results list will be updated with the selected filters/options. To view more information on a report, click on the report name and you will be redirected to the detail report information. Figure 5 below shows the report detail view:

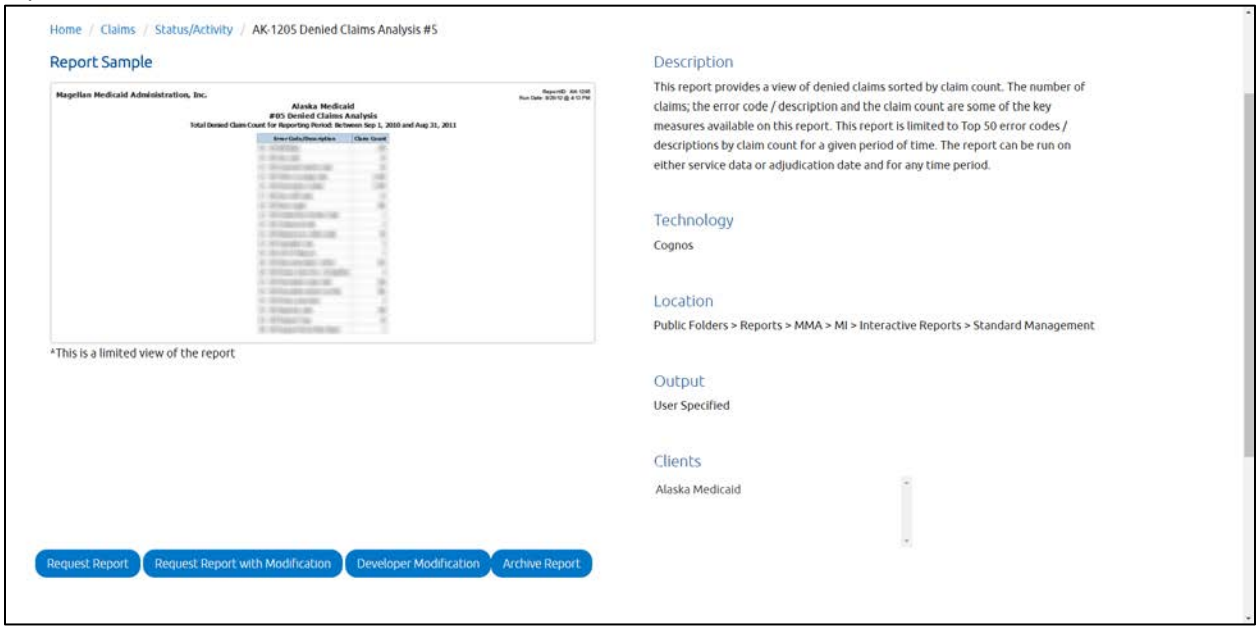

Figure 5. Report Detail View

The report list view shows a selection of important metadata fields associated with the chosen report, along with a sample image of the report, with any protected and private data obscured. From this screen the end-user will get a good representation of what the report offers, and can then decide if they need a copy of this report, or if the report is similar to what they need, except with the addition/removal of a few columns or a different client's data. For either of these options, the user can click the button at the bottom of the screen for "Request Report" and "Request Report with Modification". Once clicked, a prepopulated report request form is downloaded to the user's computer with the information from the selected report. This feature is key to alleviating work from the report request. By already having a defined set of requirements and a report to reference, less time can be spent on analysis and development and the turn-around time for a request is greatly reduced. The remaining two buttons at the bottom of the screen only appear if the user has "developer" permissions, and are used for making modifications to a report, and for archiving or decommissioning a report if the BI team no longer offers it. Figure 6 below displays the report modification view:

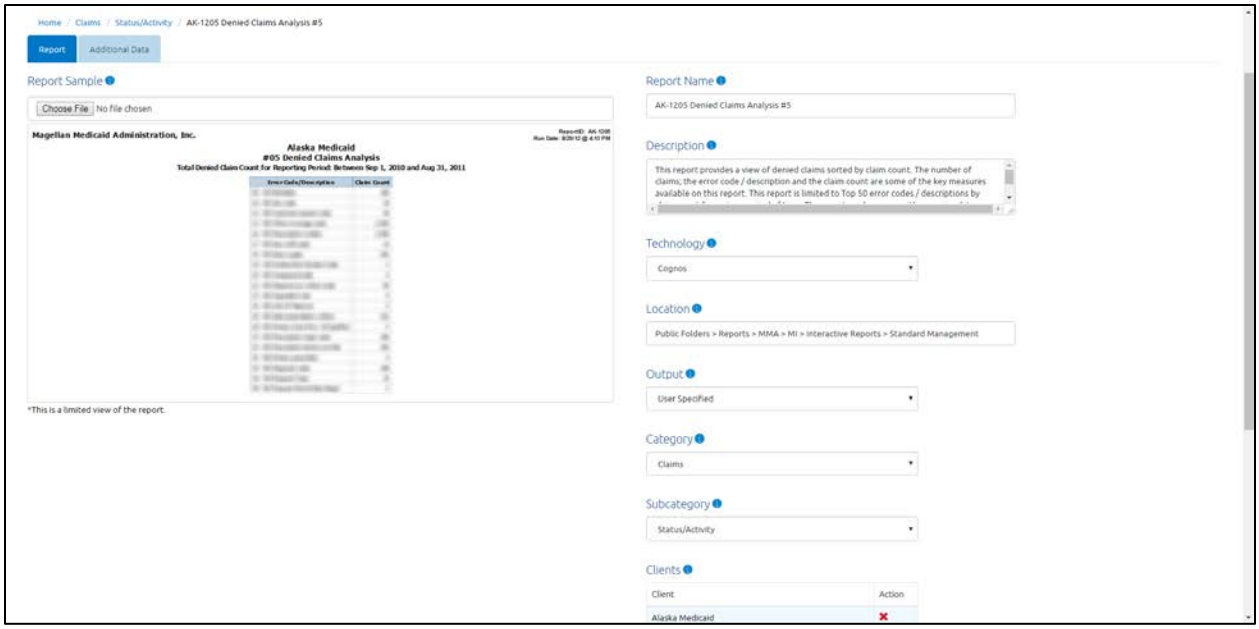

Figure 6. Report Modification View

The report modification view is where the report developer will come to make changes to all of the metadata fields available for a report. These fields are connected to the backend SQL Server database, and are linked with functions in the application that build out INSERT or MODIFY SQL statements behind the scenes to update the record. Once all modifications are completed, click on “Submit Modifications” to save the changes. You will receive a notification if the changes have been saved successfully.

INTEGRATION WITH SAS

Base SAS® is a powerful analytical and reporting tool with an endless number of uses. Throughout this project, Base SAS® was used on multiple occasions on both the report metadata collection process, as well as after the solution was implemented as a source for creating report indexes. The following sections explain and show examples of these two processes.

USING BASE SAS® TO INVENTORY THE METADATA

One of the most essential parts of this project is the actual collection and cataloging of metadata associated with all of your reports. The metadata is the key to making this solution work, and the collection of it can be one of the most time consuming efforts. While much of the metadata is not predefined and will require some manual manipulation and analysis, you can utilize a SAS macro to scan through and collect much of this information from your SAS programs. The following SAS code can be run using Base SAS and is used to collect text located around key words that you identify in the macro:

```
%macro client_search(client_name);

  %macro text_loop(search_text);
      /*    SAS Program Directory   */
      FILENAME myfiles ("\\server_name\prod\code\&client_name\*.sas");

      /*    Results are written to this dataset */
      DATA  results;
            length client $15 tmp fname $1000;
            INFILE myfiles truncover filename = tmp;
            client = "&client_name";
```

```
          INPUT @1 line $200.;
          line = left(translate(line," ","09"x));   * Convert tabs into
spaces;

          fname = tmp;

          /*    Uses the index function to search for the text parameter
from the text_loop macro    */
                Results are written to this dataset;
                if index(upcase(line),upcase(&search_text));
    run;

    /*    append each iteration onto the search_results dataset */
    proc datasets;
          append            base = search_results
                            data = results
                            force;
          delete results;
    quit;

%mend;

/*  Call the macro to look for each instance of the search text */
%text_loop('%let rptid=');
%text_loop('author');
%text_loop('%let type=');
%text_loop('%let cat=');
%text_loop('%let sub_cat=');
%text_loop('%let desc=');

%mend;

/*  Call the macro to look in multiple client directories */
%client_search(dc);
%client_search(ak);

/*  sort by client and report     */
proc sort
    data=search_results;
    by client fname;
run;

/*  transpose all results onto one row per report   */
proc transpose
    data=search_results
    out = search_results_trans;
    by client fname;
    var line;
run;
```

In the code sample above, there are two macros being used. The first is an overall macro to loop through the specified SAS program file directory. This is a location on our server that contains all of the SAS code broken out into separate folders based on the client. The second macro is looping through the SAS programs within the directory, scanning for the specified search text passed via the %text_loop with the index function. All of the results are then appended to the data set named “search_results”. After the conclusion of the macros, the data set is sorted and then transposed, to place all of the results on one

line per report entry. As with any solution, standardization is key, and the code above is dependent on some standardization within the code base. All of the programs within the directory should have the standard header that identifies the report name, author, scheduling, etc., as well as the standard index routine that identifies the categorization, file type, and other variables. However, standards have changed over the years, and there will inevitably be code in production that does not conform to these guidelines. If the programs are structured according to standards, then the results will look similar to the Figure 7 below, with each of the search results broken out into columns:

COL1	COL2	COL3	COL4	COL5	COL6
%let rptid=&rptid;	| AUTHOR : Carlos Petrinovich,	%let type=xls;	%let cat=CLAIMS MANAGEMENT;	%let sub_cat=WEEKLY REPORTS;	%let desc=PAID CLAIMS WITH OTHER COVERAGE CODE 3 and 4;
%let rptid=AK_211943;	| AUTHOR : Scott Smith	%let type=xls;	%let cat=PRIOR AUTHORIZATION;	%let sub_cat=WEEKLY REPORTS;	%let desc=Alaska Hep C Weekly Report;
%let rptid=AK_217342;	| AUTHOR : Scott Smith	%let type=xls;	%let cat=PRIOR AUTHORIZATION;	%let sub_cat=WEEKLY REPORTS;	%let desc=Alaska ERLA Weekly Report;
%let rptid=130698;	| AUTHOR : Robert Woodley	%let type=html;	%let cat=CLAIMS MANAGEMENT;	%let sub_cat=MONTHLY REPORTS;	%let desc=Twelve Month Summary;
%let rptid=DC_227341;	AUTHOR :	%let type=pdf;	%let cat=PHARMACY MANAGEMENT REPORTS;	%let sub_cat=WEEKLY REPORTS;	%let desc=Prior Authorization Type Code 4 Claims;

Figure 7. Output from a SAS macro code

Once the data has been extracted and transformed to meet the requirements of the database structure in the report encyclopedia, it is ready to be loaded. There are many options for completing this task, ranging from a proc sql statement within Base SAS, or a SQL Server bulkload command importing the data from a CSV file.

USING THE REPORT ENCYCLOPEDIA WITH REPORT INDEX GENERATION

There are endless methods to export and consume the output from a SAS program, and each organization will have their own process, standards, and expectations. One common practice to associate a set of metadata attributes with a report is to pair it with an index file. When a report is “published”, there are two files created from the SAS program: the report file itself (PDF, Excel, HTML, etc.), and the index file (.IND file). These two files will typically have the exact same name, so that they can be linked together when pushed to a webserver for publishing. In our organization, we have a centralized program that is called upon at the end of a program in a %include statement when exporting the report to the desired location. This index uses a set of variables that are defined manually within the report itself, which is not ideal, but the solution has never run into any real issues. However, now that the Report Encyclopedia has been created and serves as the central location for all report Metadata, we can now create the index in a more dynamic way by querying the database. The following macro can be used to accomplish this task:

```
%macro report_index(report_id);


proc sql;
    connect to oledb (provider = "SQLOLEDB" properties = ("Integrated
Security"=SSPI"data source"=server_name"Initial Catalog"=BI_Reports));
    create table rpt_index as select * from connection to oledb
    (
       SELECT     rep.reportid as rptid,
                        rep.reportOutput as type,
                        sch.Frequency as freq,
                        cat1.CategoryName as cat,
                        cat2.CategoryName as sub_cat,
                        rep.description as rep_desc,
                        cl.clientname as client
      FROM report rep
            join categories cat1
```

```
                on rep.categoryId = cat1.id
          join categories cat2
                on rep.subcategoryId = cat2.id
          join Scheduling sch
                on rep.reportkey = sch.reportkey
          join client
                on rep.reportkey = client.reportkey
          join clientlookup cl
                on client.clientlookupid = cl.id
    WHERE rep.reportid = (%str(%')&report_id%str(%'))
  );
disconnect from oledb;
quit;
%mend;

%report_index(MI_0015_2);
```

In the example above, a proc sql statement is wrapped in a macro and is passed the report_id from the report. The SQL statement then connects to the Report Encyclopedia database and queries the tables to return data for the passed report id. By altering the %include program that creates the report index file in this manner, there is no longer room for error from manual text entry, the data will be more consistent and standardized, and this new process enforces that the developer has created a record in the Report Encyclopedia and keeps it up to date. If any of the metadata changes, we can simply update the entry using the Report Encyclopedia web application and avoid having to touch the code in production.

THE CONCLUSION

When the project concludes and the Report Encyclopedia is fully populated, you should have very few instances where a user would need to request a new report, and should only be allowed to request a new report once they have confirmed that no existing and readily available report will meet their needs. By adding this additional step in the workflow for a report request, you can ensure that the duplication of work is at a minimum. In summary, we have reviewed some of the essential things to keep in mind when defining requirements and selecting the right technology. It is important that the solution is tailored to your team goals and the business needs, and not just another product that will fall into the background, unused and outdated. It is the responsibility of the team to make sure records stay up to date and valid. In our organization, we have added the “Report Encyclopedia Check” as part of the report approval process to ensure that this occurs. We have also discussed how the database design and the user interface interact with each other and how a developer can maintain records within the application. You don’t want the act of searching for or updating a record to be cumbersome and time consuming. It is more likely to be adopted if the user interface is clean, intuitive, and responsive. Finally, we have discussed how Base SAS was involved in the collection of the report metadata, and how the overall SAS development workflow has improved through the use of the Report Encyclopedia database with the report index generation process. The role of Business Intelligence in your organization can be more than just “reporting”, and the Report Encyclopedia can be a stepping stone in the journey of transforming the BI team into creating reusable information delivery solutions.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

John Mattox
Magellan Health
1-804-548-0131
jmattox@magellanhealth.com