Paper 265-2009

"Min Logistikk" – Using SAS® BI Web Services in a Service Oriented Architecture at Norway Post Delivering Information Securely to Customers on Extranet

Truls Andersen, Know IT Information Management AS, Oslo, Norway
Frode Langseth, SAS Institute AS, Oslo, Norway
Shiraz Ali Bhaiji, Avenir AS, Oslo, Norway
Stein Nyfløt, ErgoGroup AS, Oslo, Norway

ABSTRACT

This paper describes the systems architecture of one the reporting solutions that were developed through the program "IKT Plattform 2005-2006" at Norway Post. One important requirement for this solution was to deliver information securely from the internal data warehouse of Norway Post to business customers on extranet. Avenir AS got the assignment for the user interface using Microsoft .Net, while Know IT Information Management was responsible for delivering the report data using SAS®9 tools. The final decision was made to implement the innovative and unconventional technology SAS® BI Web Services for Java to deliver XMLA to the external Microsoft .Net application using a service oriented architecture. Ergo Group Systems Integration was responsible for the Web Access Control using eConnect/eTrust from Computer Associates.

INTRODUCTION

When the assignment of this reporting project started the customer was Norway Post Logistics. Their core business was collection, transportation and delivery of parcels primarily inside Norway. Norway Post was traditionally a governmental company, but during the last years the company has acquired several subsidiaries, e.g.:

- Box™ was the leading provider of express delivery and logistics services in the Nordic market, and Estonia. Box was the specialist within express logistics and delivery services, and Box Solutions was the specialist within warehousing, container and trailer haulage, and harbor services.
- Nor-Cargo AS was Norway's leading provider of transport and logistics systems by road, air and sea.
- Frigoscandia Distribution was the first-choice logistics provider to Europe's food processors, wholesalers and retailers.

Late 2008 Norway Post Logistics and its subsidiaries were consolidated into one big organization named Bring. Bring continues to be a part of Norway Post, with the six specialist areas Express, Citymail, Mail, Logistics, Frigoscandia and Dialogue. The owner of this project will be named Bring Logistics for the remaining part of the paper.

Significant parts of the program "IKT Plattform 2005-2006" were initiated to support the core business of Bring Logistics. The major tasks were to upgrade existing - and to develop new business intelligence (BI) applications and solutions regarding revenue, invoicing, transportation planning, production planning, production reporting, quality control, and internal accounting. The applications should be running on both Windows and Unix operating systems. Some of the BI-applications involved other innovative technologies like reading IBM MQ Series data from SAS®9, and data storage on SAS® Scalable Performance Data Server (SPDS).

The subject for this paper is the "Min Logistikk" reporting solution using SAS® BI Web Services to read SAS® data sets from two data marts, delivering report data through the Web Access Control layer using eConnect/eTrust (Computer Associates) - to a Microsoft .Net application on Extranet. The paper describes how SAS® is one of several important pieces to complete the puzzle and the roles of SAS® software components in a modern service oriented architecture with strict security requirements.

DW ARCHITECTURE + BI APPLICATIONS AND SOLUTIONS

Know IT Information Management Oslo AS was responsible for development and management of the applications listed in table 1. Figure 1 display how the applications fit into the Data Warehouse Architecture at Norway Post.

Table 1: Applications developed and managed by Know IT Information Management Oslo AS.

Application	Content
LM	”Logistics Engine” – simple reports based on detailed observations/events for the parcels. Used for monitoring deliveries and invoicing. Exports to other applications.
SFOR	“Standard Invoicing and Reporting” - customer centric reports describing customer activity in the matter of products, VAS (value added services), curves in consumption pattern etc.
Min Logistikk	“My Logistics” - important customers may check delivery status and make delivery reports of different kinds on extranet. 8.750 registered users as of January 20, 2009.
KOS & IR	“Quality, Monitoring and Statistics (KOS)” – the major application in the project. More than 1200 internal users. The most critical part keeps control over transportation. “Internal Accounting (IR)” – reports on costs and productivity in the parcel process, including some KPI’s for production, absence, net and gross productivity and unit costs etc.

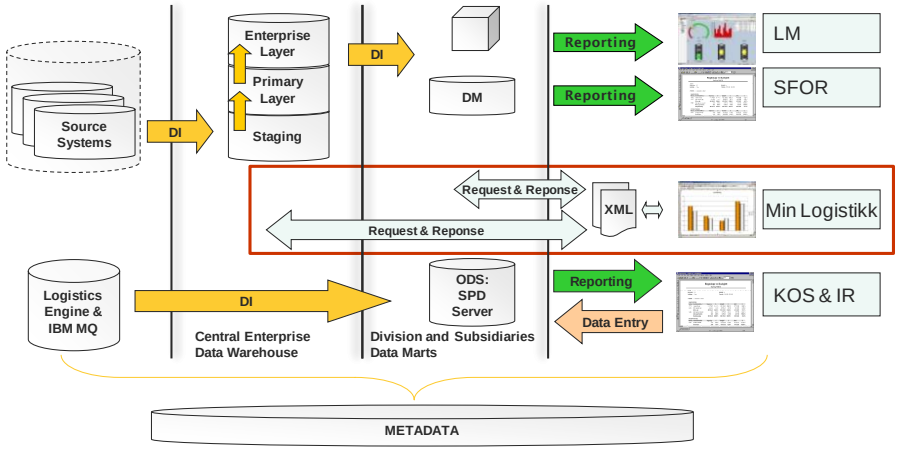

Figure 1: Norway Post – Bring Logistics’ BI applications and solutions in the DW Architecture

All reporting and analysis data should be present in the Primary Layer and the Enterprise Layer. The Enterprise Layer is the part of the common data warehouse which is prepared for reporting and analysis. Divisions and subsidiaries may develop their own Application/Solution Data Marts. Both the Enterprise Layer and the Data Marts may be exploited for reporting and analysis. No data should flow between Data Marts.

WEB SERVICES – IN GENERAL

W3C defines Web Services as “*a software system designed to support interoperable machine-to-machine interaction over a network*”. Web Services are independent of the underlying environment and programming language. A Web Service is one way of enabling synchronous communication between loosely coupled components and systems and commonly used in junction with Service Oriented Architecture (SOA). SOA does not require Web Services, nor does usage of Web Services imply that SOA is implemented.

In an architecture where Web Services are used, three parties are involved:

- Service Broker. Handles administration and a Web Service directory (Universal Description, Discovery, and Integration, UDDI)
- Service Requester. The client in a transaction, consumer of a service
- Service Provider. The server in a transaction, provider of a service

Many Web Service characteristics are comparable to Stored Processes in the SAS® world. A SAS® Stored Process is a SAS® program with an interface, authorization rights, and other properties defined in SAS® Metadata.

SAS® BI WEB SERVICES – BASIC ARCHITECTURE

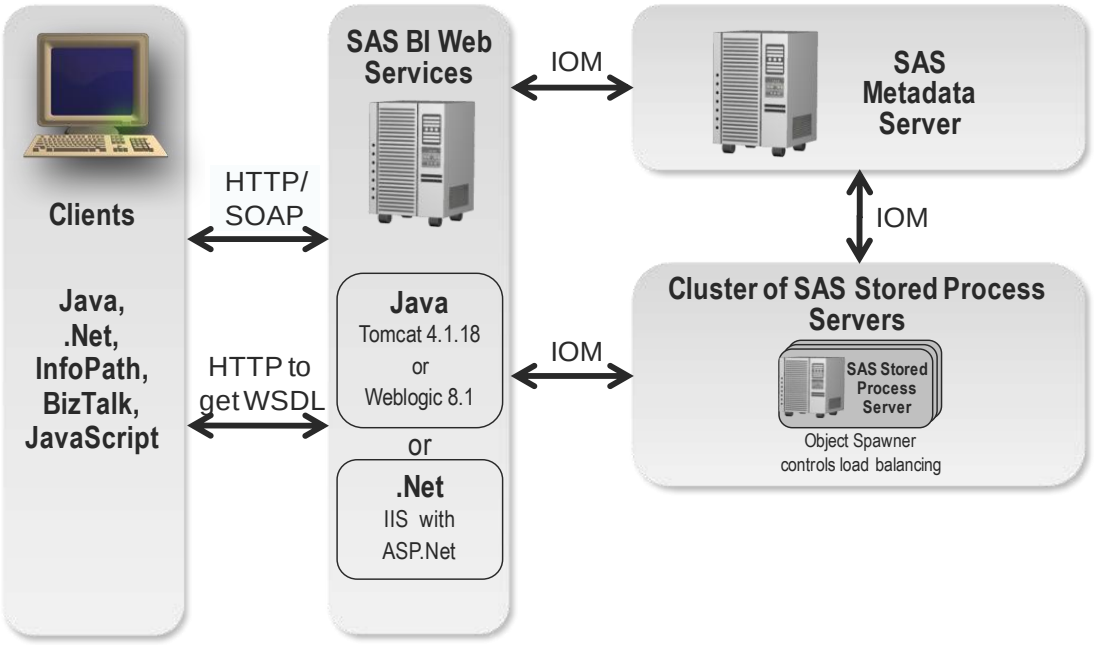

Figure 2: Basic Architecture of SAS® BI Web Services.

The URL http://support.sas.com/rnd/itech/doc9/dev_guide/websrvcs/index.html provides an overview of SAS® BI Web Services. This is the main reference for the contents of this chapter.

The SAS® Stored Process is the core component of the Web Services support provided by SAS®. The SAS® BI Web Services application surrounds the SAS® Stored Processes to support Web Services specific issues such as message transportation (HTTP is often used), provide Web Service Description Language (WSDL) interface, support for Simple Object Access Protocol (SOAP), etc.

There are two implementations of SAS® BI Web Services: one written in Java that requires a servlet container, and another written in C# using the .NET framework. For information about the differences between SAS® BI Web Services for .NET and SAS® BI Web Services for Java, see http://support.sas.com/rnd/itech/doc9/dev_guide/websrvcs/decide.html.

The SAS® BI Web Services interface is based on the XML for Analysis (XMLA) Version 1.1 specification. The figure above shows how Web services work. A client, such as a Web application or desktop application, starts by obtaining the Web Service Description Language (WSDL) from the Web service. The WSDL describes the available methods, endpoint (where to call the Web service), and the format of the XML required to call the Web service. Any client that can send and receive SOAP messages can access Web services. SAS® supports SOAP bindings over HTTP. The client sends XML requests and parameters in a SOAP envelope to the Web service, telling the Web service to either Discover or Execute stored processes. Discover() and Execute() are the two methods that are specified for XMLA. The Discover method consists of middle-tier code that calls the SAS® Metadata Server to get the requested metadata. The Execute method consists of middle-tier code that calls the SAS® Stored Process Server to invoke stored processes.

During execution, the requested stored process is executed by a SAS® Stored Process Server. Any number of simple string parameters are usually passed to the stored process, and a stream of XML data is returned. The input parameters to the stored process can be nothing or a combination of simple string parameters and any number of XML streams. The output from a stored process called by a SAS® BI Web Service is always an XML stream.

"MIN LOGISTIKK" REPORTING SOLUTION

TECHNICAL CHARACTERISTICS

Data Integration:
- Base SAS® (SAS Warehouse Administrator™ + SAS® Data Integration Studio) by Ergo Group-DWH

Distribution:
- SAS® BI Web Services for Java (i.e. XMLA)

Output:
- Streaming by means of XMLA enabled SAS® Stored Processes, running both on Unix and Windows

User Interface:
- Microsoft .Net application including interactive and subscription services developed by Avenir AS.

User Database:
- Responsibility of Avenir AS

SAS® Metadata Identities:
- Only one single SAS® Metadata Identity on each OS

Web Access Control:
- Systems Integration by Ergo Group-Systems Integration, using eConnect/eTrust from Computer Associates.

BUSINESS USAGE & REPORTS

The "Min Logistikk" Reporting Solution was established for the corporate market. It enables Bring Logistics' corporate customers to check delivery status and make delivery reports on extranet. More than 8750 external users were registered as of January 20, 2009. When Bring Logistics started to offer this reporting solution, customers considered it to be "nice-to-have". Today, many customers consider the solution to be "mission-critical".

The solution contains the following reports on Unix and Windows, primarily detail reports on Unix and summary reports on Windows.

Unix (FA Mart on Extranet)

- **Shipment control:**
  Report: Deviations between parcels registered for shipment and actual parcels shipped.
  Usage: Controlling theft from the conveyor belt, plus extra parcels shipped on the belt.

- **Preliminary freight calculations + Detailed freight statistics:**
  Report: Size, weight, cost of shipped parcels as measured by Bring Logistics.
  Usage: Bring Logistics' customers use this for billing their customers, and to control for deviations between their own measurements and Bring Logistics'.

- **Packages delivered + PoD report (Proof of Delivery):**
  Report: Which parcels have been delivered to who and when.
  Usage: Bring Logistics' customers get facts regarding deliveries of parcels to their customers, being able to perform proactive customer relationship management (CRM). "Case closed" after delivery.

- **Delays + Delay alerts:**
  Report: Parcels that should have been delivered to customers that did not reach the time limit.
  Usage: Facilitates Bring Logistics' customers to perform proactive CRM in order to alert delays.

- **Quality of deliveries:**
  Report: Detail and summary reports on miscellaneous statistics regarding quality of delivery.
  Usage: Bring Logistics' customers may use this report to analyze transportation problems in certain districts, regions, or parts of Norway.

- **Ad Hoc report:**
  Report: Facilitates reports on other events in the delivery chain than the other reports.
  Usage: In depth understanding of problems in the delivery chain.

Windows (SFOR on extranet)

- **No. of shipments and no. of value growth services:**
  Report: Summary report on freight costs/charges by product and month.
  Usage: Control on amounts according to transportation agreements with Bring Logistics.

- **Packages per price zone:**
  Report: Summary report on freight costs/charges by product, month and price zones.
  Usage: Compare amounts to last year and budget control.

- **Turnover per customer and product:**
  Report: Turnover per customer and product.
  Usage: Budget control.

DATA INTEGRATION & DATA STORAGE

Details regarding Data Integration are not considered to be an important subject in the context of Systems Architecture. Therefore, this chapter will partly be limited to describing data storage and data size - and partly how the XMLA enabled SAS® stored processes (web services) are consuming the data.

The data warehouse at Norway Post has been storing data in SAS® tables for many years. The initial Data Integration processes of this project were developed using Base SAS® programs as user written code - managed and documented by SAS Warehouse Administrator™ part of SAS® Foundation. Detail data are stored in SAS® tables and views on Unix, while both detail and summary tables are stored in SAS® Scalable Performance Data Server (SPDS) on Windows. Summary data on SPDS might initially sound strange, but the reason was that the critical "KOS & IR" application running on the same Windows server required data to be stored in SAS® SPD Server – leading to data for other applications to be stored in SPDS as well. By the time of writing the number of rows/columns in the SPDS tables ranged from 40.000/18 with 0 indexes - to 53.000.000/11 with 3 indexes.

Normally, each report request implies that the XMLA enabled stored process (web service) submit a lookup in 1 SAS® table/view. There are only 2 reports (PoD report & Ad Hoc report) looking up information in 2 tables, and only one report (Quality of deliveries) looking up information in 3 tables. 3 of the 9 detail reports on Unix are looking up information in multiple tables/views, where one report reads from 3 physical tables, an other one from 1 physical table and 1 SQL view, and the last one from 2 SQL views. Further 4 of the detail reports on Unix using only 1 table/view are looking up information in SQL views. By the time of writing the number of rows/columns in the physical SAS® tables on Unix ranged from 90.000/50 with 0 indexes - to 66.000.000/43 with 1 indexes.

SAS® BI WEB SERVICES – EXTENDED ARCHITECTURE FOR "MIN LOGISTIKK"

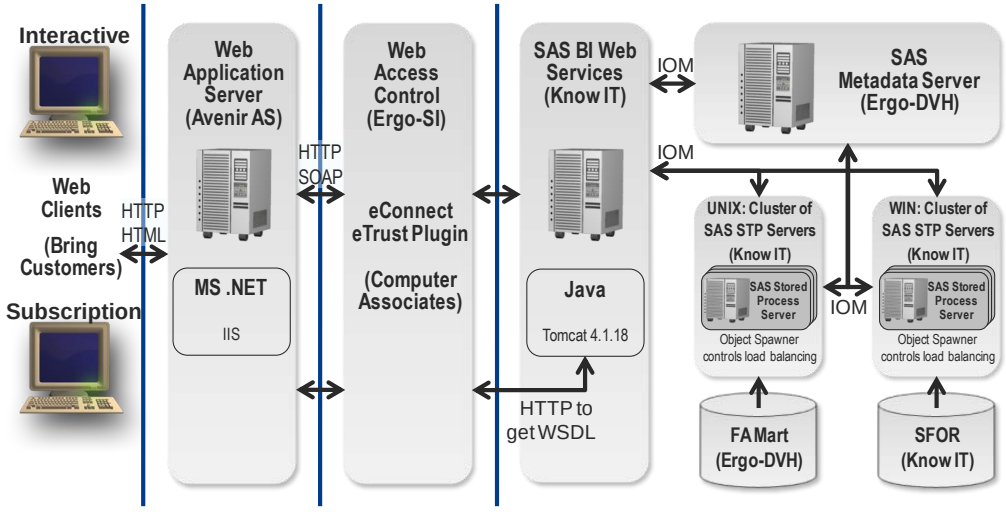

Figure 3: The extended architecture of SAS® BI Web Services for "Min Logistikk" Reporting Solution.

SAS® BI WEB SERVICES – DEVELOPMENT AND TESTING

The SAS® part of developing "Min Logistikk" reporting solution started when SAS® 9.1.3 service pack 2 was the latest release. Then, the components of SAS®9 were still pretty new for the major part of the SAS® developers that took part in the project, especially SAS® BI Services module which was brand new when the development started. The following methodology was used to develop and test XMLA enabled SAS® Stored Processes (web services):

1. Write an ordinary SAS® program that reads the required table/view and subsets the data by means of macro variables to emulate parameter transfers according the report specification – the "good old" way by means of the Enhanced Editor in SAS® Display Manager System (DMS).

2. Modify the SAS® program according to the specifications of a SAS® Stored Process (%ProcessBody; - %stebegin; and %stpend; were omitted) and register the SAS® program as an ordinary SAS® Stored Process, in order to ensure that Stored Process was able to deliver through the infrastructure of the SAS® Enterprise Intelligence Platform by means of the SAS® Stored Process Web Application.

3. Modify the SAS® Stored Process code and metadata registrations to XMLA enable it, i.e. modifications to the libname statements to use the XML libname engine and making the first keyword in the metadata registration to be exactly "XMLA Web Service".

4. Test the XMLA Web Service directly against the web application server running SAS® BI Web Services for Java on the inside of the network of Norway Post, i.e. without going though the Web Access Control layer. The key component that allowed that to happen was the runClient.java application that comes with the SAS® BI Web Services installation. This application was either run from a laptop computer with SAS® BI Web Services for Java installed for experimental purposes – or directly from the console on the web application server running the SAS® BI Web Services. Important elements for runClient are the following methods and files:

    a. **Discover()** – "Discover" successfully registered XMLA Web Services - and write a listing in the in HTML-file Discover.htm

    b. **Execute()** – "Execute" one successfully registered web service – and write the result of the response to the HTML-file Execute.htm

    c. **command.xml** – contains the URL of the web service to be run by the Execute() method, inclusive name/value-pairs of parameters

    d. **properties.xml** – contains e.g. authentication information

    e. **restrictions.xml** – never used by the SAS® developers in this project…

Example on the SAS® program of an XMLA Web Service, e.g. "WS_KM_delayed_parcels.sas":

```
*** 1. Start the stored process ***;
*ProcessBody;

*** 2. Declare input parameters ***;
%global anr produktkode fradato tildato;

*** 3. Declare output parameters ***;
%let  XMLSCHEMA = SchemaData;
%let outdata    = _WEBOUT.resultat ;

*** 4. Assign input and output XML librefs ***;
libname instream xml ;
libname _WEBOUT xml xmlmeta=&_XMLSCHEMA ;

*** 5.1 Macro program to build part of where clause*** ;
%macro build_where_part ;
%global prod where ;
  %if &produktkode NE %then %do ;
  %let prod_where = %str (and identprodukt = "&produktkode") ;
%end ;
%mend ;
```

```
*** 5.2 Run the macro program ***;
%build_where_part ;

*** 6. Subset data based on input parameters;
data delayed_parcels;
  set famart.FA KM DELAYED_PARCELS;
  where AKTORNR = "&anr"
        &prod where
    and DATOANKOMSTEST >= input ("&fradato", ddmmyy10.)
    and DATOANKOMSTEST <= input ("&tildato", ddmmyy10.);
run ;

*** 7. Translate illegal characters into blanks ***;
data delayed_parcels;
  set delayed parcels;
   aktornavn   = translate(aktornavn,
     ' ','0102030405060708B0C0E0F101112131415161718191A1B1C1D1E1F'x);
   produktnavn = translate(produktnavn,
     ' ','0102030405060708B0C0E0F101112131415161718191A1B1C1D1E1F'x);
run ;

*** 8. Send to output stream ***;
data &utdata. ;
  set delayed_parcels ;
run ;
```

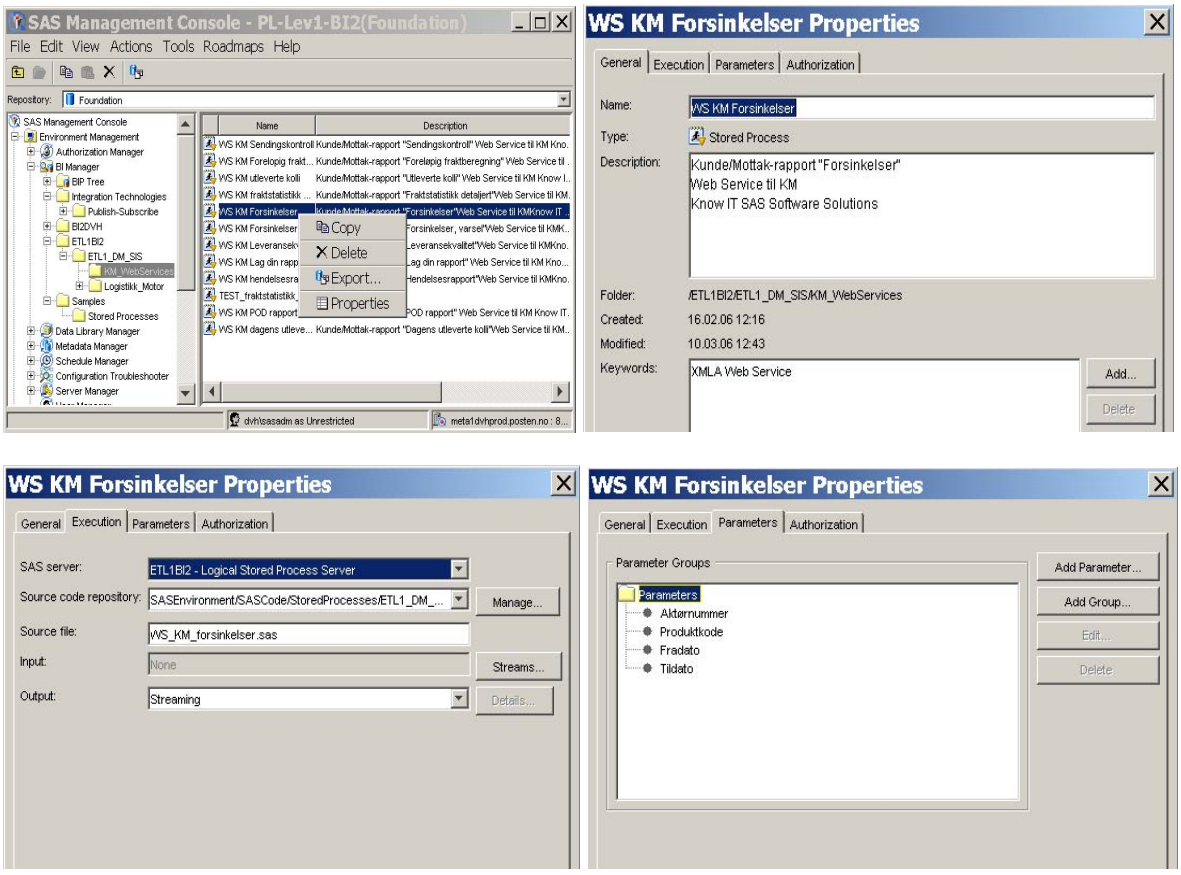

Figure 4: Register SAS® metadata for the XMLA Web Service, i.e.
a. Properties of a SAS® Stored Process, b. General, c. Execution, d. Parameters

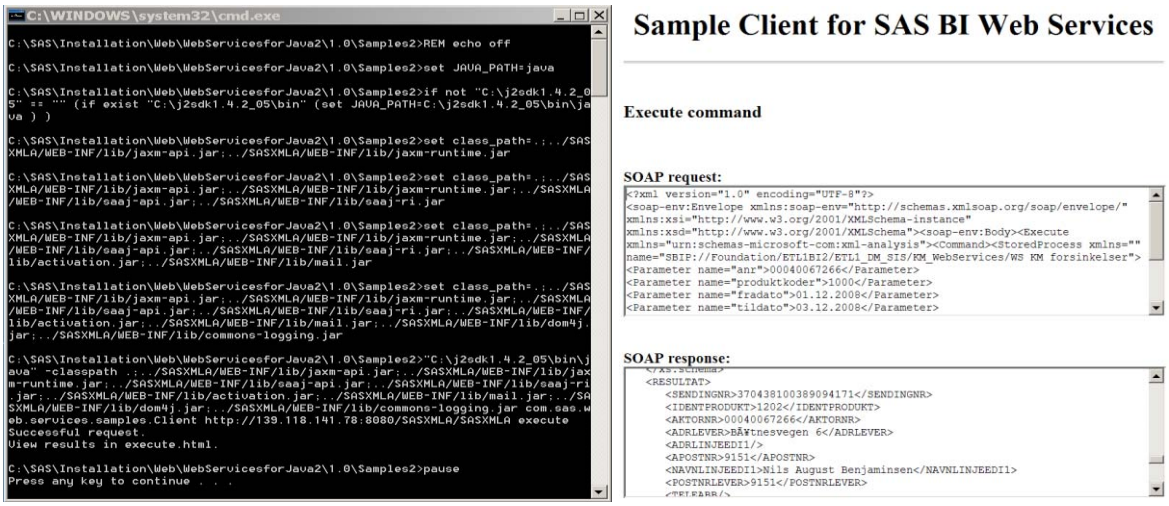

Figure 5: Testing the web service inside the Web Access Control using runClient.java, displaying a "Successful request" in "excute.html" residing in the same folder.

Working one or two parts of a pretty complex Service Oriented Architecture(SOA) reporting solution the runClient.java application was indeed very helpful to the SAS® developers while unit testing the XMLA Web Services, and at later stages in order to reject claims that the services were not work from outside the Web Access Control (referring to Figure 3). But, there was some initial trial and error to find out how runClient really was working.

WEB ACCESS CONTROL – ECONNECT/ETRUST

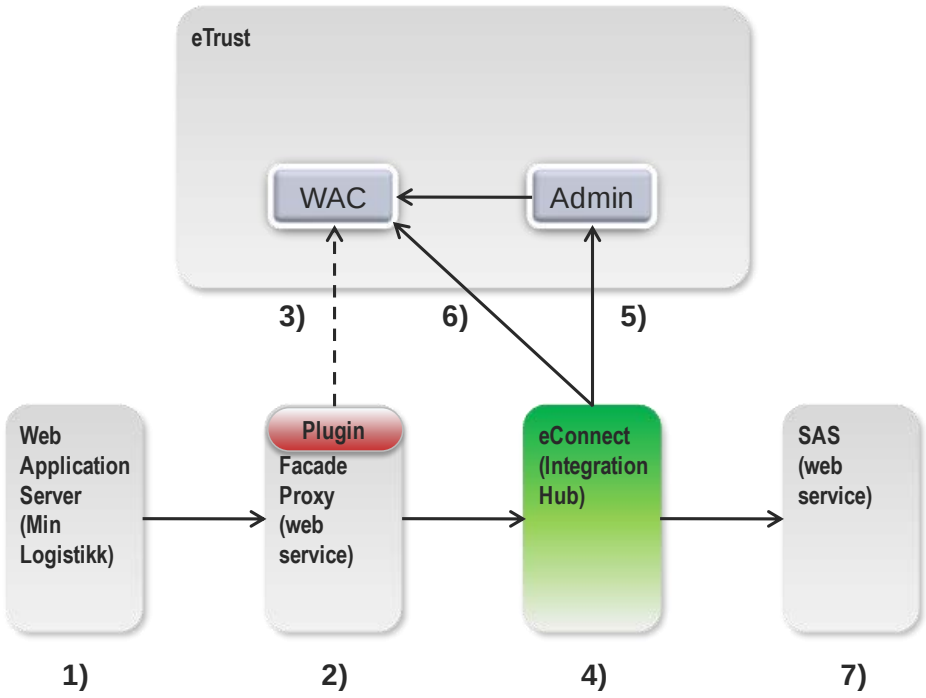

Figure 6: Web Access Control (WAC) – eConnect/eTrust, initial setup.

The eConnect Integration Hub administer the communication between Min Logistikk and SAS®.

The Web Application Server ("Min Logistikk") makes request via eConnect web services. "Min Logistikk" needs to call the web service through the open Internet. In order to do that the web service is exposed in a dedicated server (the Facade Proxy) which allows only specified URL's to be called.

1. The Facade Proxy routes the request to eConnect. In order to start a dialog, "Min Logistikk" first must call an authentication service (via eConnect). This service returns a token which is used in the following request.

2. An eTrust web agent is installed in the Facade Proxy. The web agent is connected to eTrust. It uses the incoming token to perform http-authorization of the requests.

3. If the request slips through the http-authorization it is routed further to eConnect. The request must supply information about the calling user (identity and token)

4. eConnect uses the user identity to ensure the user is registrated in eTrust (admin) and it checks consistency between the token and the user identity. eTrust admin is called by using a Java-API.

5. If everything is OK, eConnect analyze the SOAP request in order to determine which SAS® report is asked for. The Web Access Control (WAC) in eTrust stores information about which users have access to which reports. eConnect calls the eTrust WAC to check if the user has proper access. The communication between eConnect and eTrust is made though a Java API. In eTrust, the reports are stored as "Resources" (GURLs and URLs).

6. In the users have access to the report eConnect takes the original "Min Logistikk" request and uses it as input for calling the SAS® Web Service. The response from SAS® is returned back to "Min Logistikk".

Late 2008, when Norway Post Logistics and its subsidiaries reorganized into Bring, the Microsoft .Net web application server was moved inside the network of Ergo Group AS. This network is tightly integrated with Norway Post as Ergo Group AS is the primary suppliers of services regarding networking and systems integration for Norway Post and Bring. According to figure 7 this implied that the Façade Proxy could be removed without compromising the security of the solution.

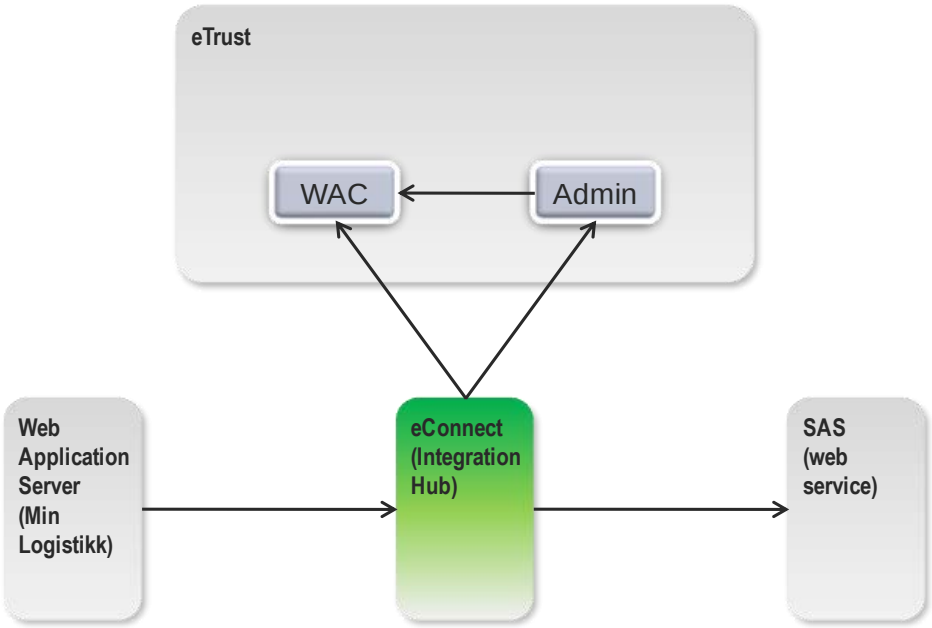

Figure 7: Web Access Control (WAC) – eConnect/eTrust, modified setup without the Façade Proxy.

This change has resulted in easier administration and management of the solution, and some performance improvement.

USER INTERFACE – MICROSOFT .NET APPLICATION

The user interface is a web based solution, accessed over Bring's extranet using a browser. The user interface is implemented by Avenir AS using Microsoft .Net. This application is the interface users have to utilize to access the logistic reports from extranet. Users are never exposed to the underlying complex security (only required to log in), the fact that SAS® is involved at all, how data is collected, transformed and summarized, etc. Figure 8-10 display the user interface running one of the reports interactively. Besides, the solution offers subscription services for reports.

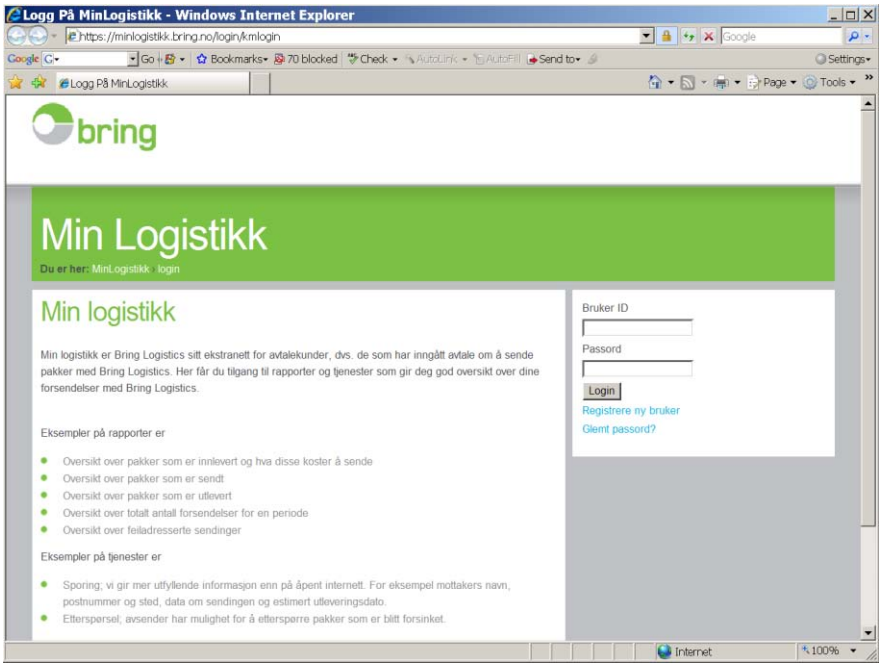

Figure 8: Entry point on extranet (WWW)

Figure 9: Select report "Forsinkelser - valgt periode" (Delays - selected period)

Figure 10: a. Make selections, b. Display results

Figure 11 provides an overview of the entire system as seen from the web server front end. The front end is programmed in Microsoft .Net, the eConnect service bus is programmed in Java. The core systems include SAS® data warehouse, IBM Mainframes, Oracle databases and LDAP catalogs. SAS® BI Web Services are limited to extract information from the "Data Warehouse SAS®9" in this large context.

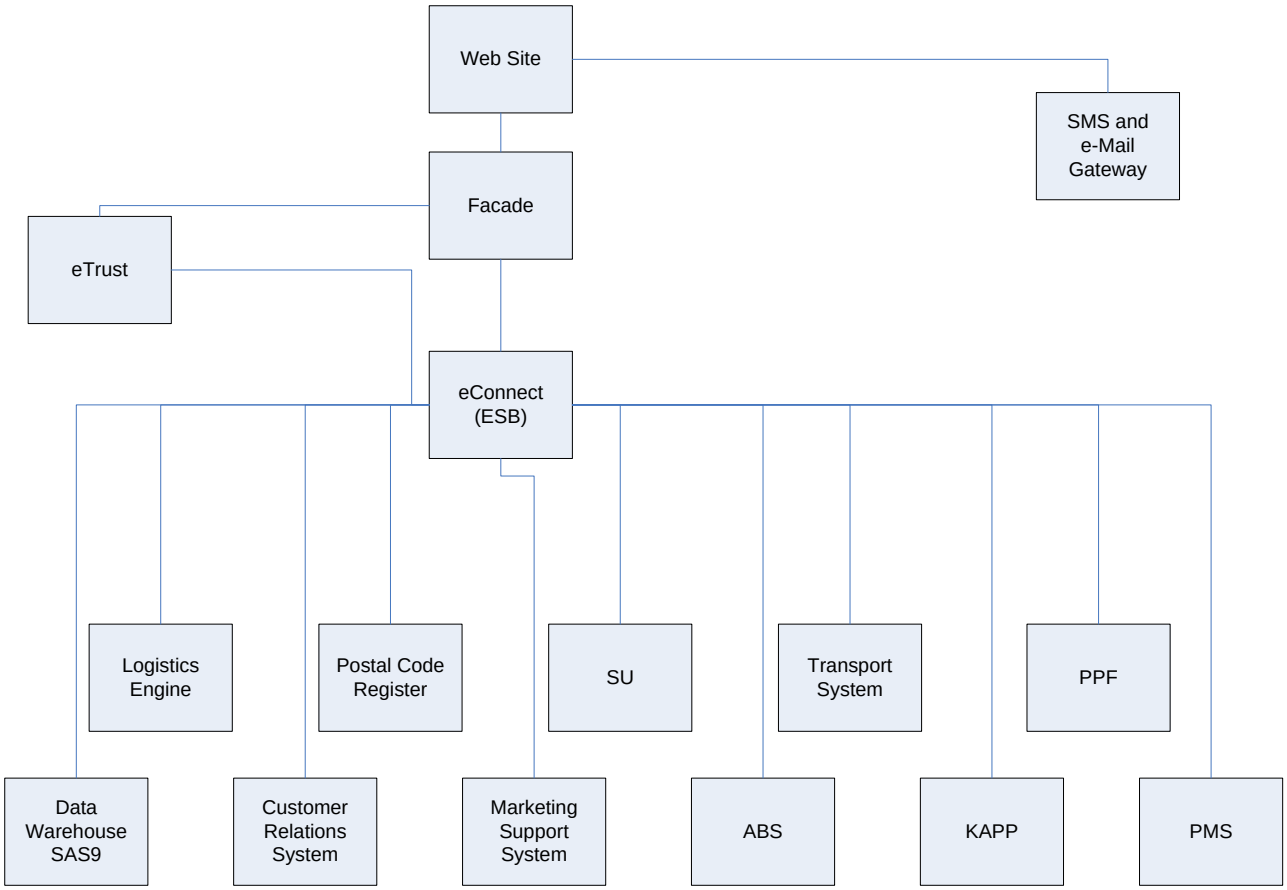

Figure 11: "Min Logistikk" - overview of the entire system

SECURITY

The main goals and requirements for security related to reports were:

- Unauthorized users should not be able to run any reports or to see any data.
- Users that have access rights to one company should not be able to see data from another company, without having necessary access rights.
- Users should only be able to run reports within the company that they are allowed to run. For example, all users should not be allowed to run management reports.
- The reports that can be run on a company's data should be limited to the reports that the company has subscribed to.
- As the web servers are running outside the main security context, all calls from the web servers must be validated.
- The system should use CA eTrust (LDAP directory) as an authentication and authorization database.

Challenges

The initial challenge was how to implement the access requirements within an LDAP database. One option was to have a node for each company, and a node for each role within each company. A role would give access to specific reports. The users would then be placed as members of this role.

There were several problems with this approach, mainly due to the large number of nodes that would be required, approximately 500,000. The problems were related to:

- Manageability
- Performance
- Pricing

Another challenge was where in the solution to check if the user has access to a report with specific data? The options were:

- At the web server
- At the security façade
- At the SOA layer (eConnect)
- At the data warehouse (SAS®9)

The layers above are listed in the order to in which they are called. If a lower layer does not perform an access control, it must trust one of the layers above to have performed that control.

SAS® has functionality to use an LDAP database for users, and SAS® metadata server was providing security on both report and table level in SAS® 9.1.3. service-pack 2 (sp2). This was the current release when the development of "Min Logistikk" started. Row-level security was added in sp4.

Solution

When the user logs in, a web service call with user name and password, is sent from the web server (.Net) to eConnect (Java), which in turn talks to eTrust (LDAP) via an API (C++). The web service call returns among other things a token. The token is placed in the local database and a key that can be used to access the token is sent to the user in a cookie. All pages in the solution check the user's key against the presence of a valid token.

We placed a detailed user and access rights database in an SQL server that was accessible by the web server layer. There is functionality that when a change is made to this database, then relevant changes are also made to the LDAP database.

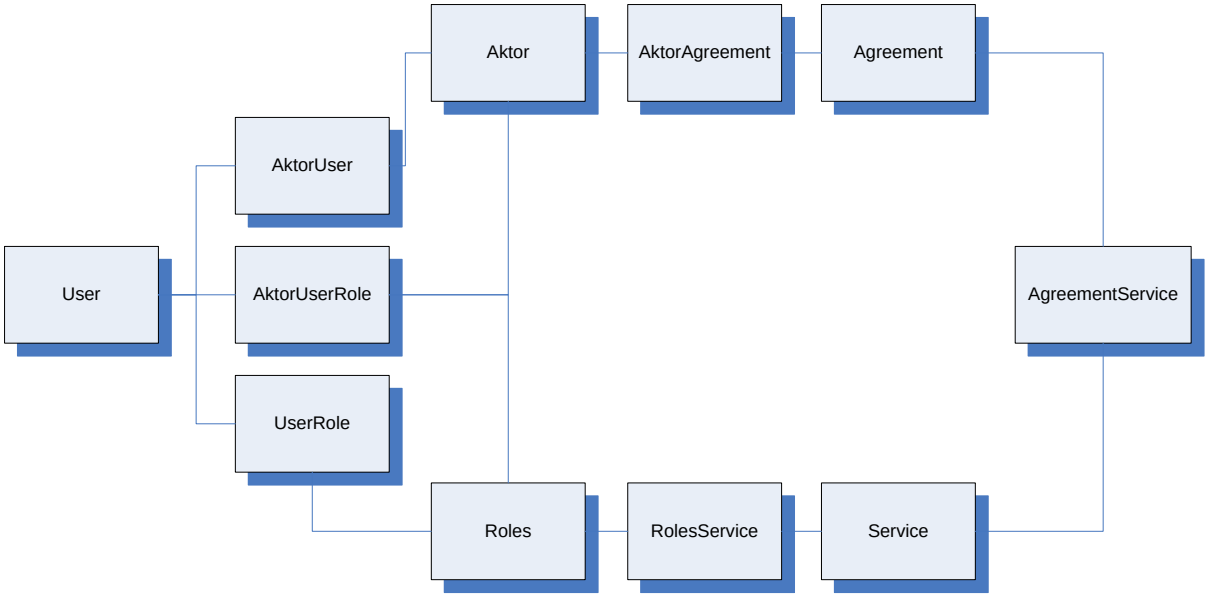

Figure 12: "Min Logistikk" - logical layout of the local security database in Microsoft .Net.
In this case a report is a "Service", and a company is an "Aktor".

To get access to a report, a user must have access to a company that has an agreement that covers that Service. At the same time the user must have access to a role for that company which grants access to the same Service.

On the eTrust side the security check is simpler since the role within the organization is not checked:

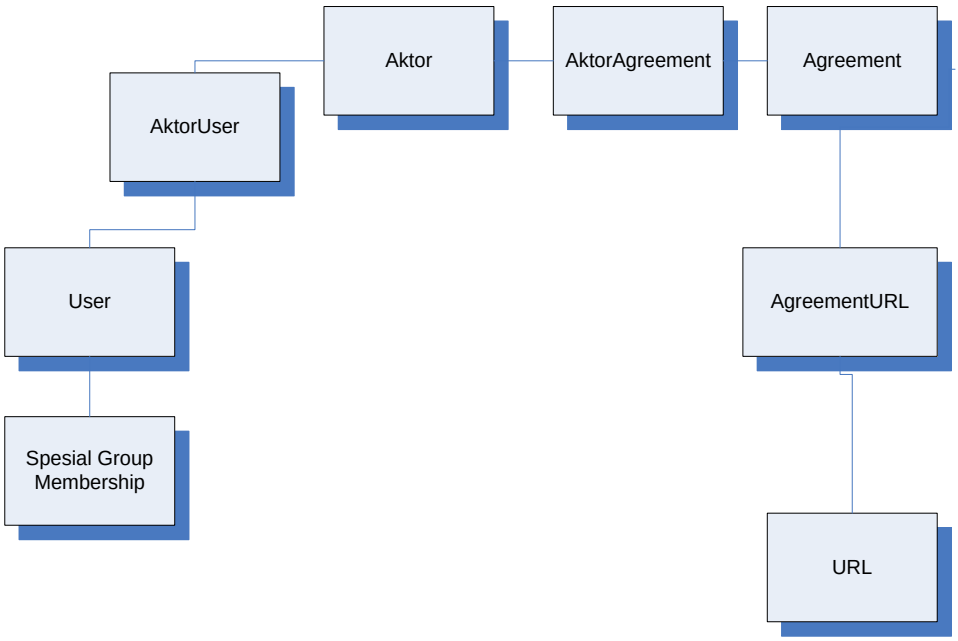

Figure 13: "Min Logistikk" – security model in eTrust.

REPORT DEFINITION

The goal while building the report definition was to provide a single general in memory report data format that could be used to hold report information independent of which report was running and how it was to be displayed. The chosen solution was the following:

Report
Contains Report Tables
Contains Report Lines
Contains Report Columns
Contains Report Cells

This structure does not only contain data, it also contains information about formatting and sorting.

In addition to standard reports the user has the ability to create custom reports. These report definitions must be saved in the database. The users can run these reports from the web site or subscribe to reports and have them sent at specific times.

Below is the data model used to connect a user to a subscription for a report, where the report is built up of specific columns and has a given format (Excel/Html):

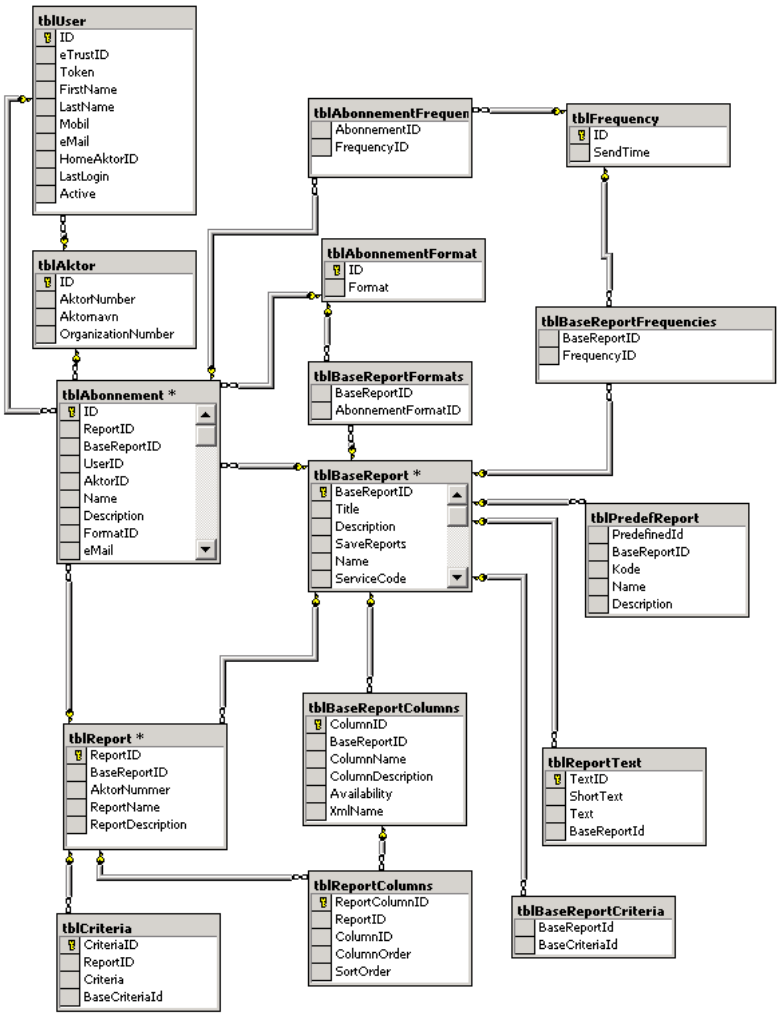

Figure 14: "Min Logistikk" – data model used to connect a user to a subscription report

REPORT EXECUTION

The example below shows some code snippets of the “HentDagensUtleverte” report. The logging, error handling and some of the complexity is removed, in order to highlight the communication with the SAS® BI Web Services.

The incoming call contains a standard internal type ReportRequestCargo and it returns a standard internal type ReportResponsecargo. This means that the code that is calling this method is unaware that the data is coming from SAS®.

```
public ReportResponseCargo HentDagensUtleverte(bool filtrerUtleverte, ReportRequestCargo request, ITransactionHelper transaction)
{
```

The first thing we must do is to build up a data request that will be sent to SAS®

```
DatavarehusRequestCargo dataRequest = new DatavarehusRequestCargo(Config.DatavarehusPathETL, SP_NAME);
         dataRequest.AddParameter("anr", request.Aktornummer);
         dataRequest.AddParameter("Hendelse",
                                  request.CriteriaCollection.FindByBaseCriteriaName("EnkeltHendelsesUtvalg").CriteriaString );

         CriteriaCargo produktCrit = request.CriteriaCollection.FindByBaseCriteriaName(BaseCriteriaCargo.Produkt);
         dataRequest.AddParameter("produktkode", produktCrit.CriteriaString);

         XmlElement dvResponse = null;
```

The system can run the data warehouse report directly against the SAS® web service or via eConnect. Which it uses is controlled via a configuration setting

```
         if (Config.WS0119_HentDatavarehusRapport != null)
         {
```

This is a call via eConnect

```
                  otd0119_HentDatavarehusRapport_Service webservice = new otd0119_HentDatavarehusRapport_Service();
                  webservice.Url = Config.WS0119_HentDatavarehusRapport;
                  webservice.HeaderMsg = BuildHeader(request.Token, transaction);
                  webservice.CookieContainer = BuildCookie(request.Token);
                  dvResponse = webservice.HentDatavarehusRapport(dataRequest.CommandElement, dataRequest.ProperitesElement);

         }
         else
         {
```

This is a call directly against SAS®

```
                  MsXmlAnalysis webservice = new MsXmlAnalysis();
                  webservice.Url = Config.WS_DvXmla;
                  webservice.CookieContainer = BuildCookie(request.Token);
                  dvResponse = webservice.Execute(dataRequest.CommandElement, dataRequest.ProperitesElement);

         }
```

The next step is to convert the response into a standard ReportResponseCargo

```
         ReportResponseCargo response = new ReportResponseCargo(request);
         ReportTableCargo data = response.DataTable;
         DatavarehusResponseEnumeration iter = new DatavarehusResponseEnumeration(dvResponse);
         while (iter.MoveNext())
         {
                  DatavarehusResponseCargo resultat = iter.CurrentRad;
                  ReportLineCargo line = new ReportLineCargo();
                  for (int i = 0; i < request.ColumnCollection.Count; i++)
                  {
                           string xmlname = request.ColumnCollection[i].XmlName;
                           if (xmlname.Equals(UTLEVERT_TIDSPUNKT))
                           {
                                    line.AddColumn(resultat.GetDateTime(UTLEVERT_TIDSPUNKT));
                           }
                           else if (xmlname.Equals(distribuert_tid))
                           {
                                    line.AddColumn(resultat.GetDateTime(distribuert_tid));
                           }
                           else if (xmlname.Equals(UTLEVERT_ENHET))
                           {
                                    line.AddColumn(resultat.GetString(UTLEVERT_ENHET, "{0} {1}"));
                           }
                           else
                           {
                                    line.AddColumn(resultat.GetString(xmlname));
                           }
                  }
                           data.Lines.Add(line);
                  }
         }
return response;
}
```

REPORT RENDERING

Based on the work above, the rendering of the reports is relatively simple.

The first stage is to get the report criteria:

```
ReportRequestCargo requestCargo=((BaseReportPage)Page).ReportRequest;
Criterias.ReportRequest = requestCargo;
ExtendedCriterias.SetReportVerdier(requestCargo.CriteriaCollection);
```

Then the report is requested and the response is sent back:;

```
ReportTableCollection responseTables =
                      GenerateResponseTables(((BaseReportPage)Page).ReportRequest);
```

Finally, the response is sent into the web control to be rendered:

```
Results.GenerateReport(responseTables);
```

The web control uses a repeater to render the HTML

```
<asp:Repeater ID="TableRepeater" runat="server"
              OnItemDataBound="TableRepeater_ItemDataBound">

<ItemTemplate>

  <table cellspacing="0" cellpadding="2" border="0"
         class="<%#DataBinder.Eval(Container.DataItem, "CssClass")%>">

    <asp:Repeater ID="HeadLineRepeater" Runat="server"
                  OnItemDataBound="HeadLineRepeater_ItemDataBound">
      <ItemTemplate>
        <tr>
         <asp:Repeater Runat="server" ID="HeadColumnRepeater">
           <ItemTemplate>
             <td class="<%#DataBinder.Eval(Container.DataItem, "CssClass")%>"
                   <%#DataBinder.Eval(Container.DataItem,"HtmlSpan")%>>
                   <%#DataBinder.Eval(Container.DataItem,"HtmlText")%>
              </td>
           </ItemTemplate>
         </asp:Repeater>
        </tr>
      </ItemTemplate>
    </asp:Repeater>

    <asp:Repeater ID="DataLineRepeater" Runat="server"
                  OnItemDataBound="DataLineRepeater_ItemDataBound">
      <ItemTemplate>
        <tr>
         <asp:Repeater Runat="server" ID="DataColumnRepeater">
           <ItemTemplate>
             <td class="<%#DataBinder.Eval(Container.DataItem, "CssClass")%>" nowrap
                   <%#DataBinder.Eval(Container.DataItem,"HtmlSpan")%>>
                   <%#DataBinder.Eval(Container.DataItem,"HtmlText")%>
             </td>
           </ItemTemplate>
         </asp:Repeater>
       </tr>
      </ItemTemplate>
    </asp:Repeater>

  </table>

</ItemTemplate>

</asp:Repeater>
```

UTILIZATION & PERFORMANCE (SAS® BI WEB SERVICES)

Figures 15-17 display selected statistics on utilization and performance regarding the SAS® BI Web Services (SAS® Stored Processes) running on Unix. The Web Services Provider is Tomcat 4.1.18 running on a Windows server.

Utilization is represented by the "No. of Runs" column displaying that the total number of requests to the selected reports constitutes approximately 25.000 for a period of two month inclusive Christmas (Figure 15). This averages to approximately 12.500 per month, 3.100 per week and 520 per day given six working days a week at Bring. Figure 16 and 17 indicates that the major part (70-80%) of the requests occur during the first half of the working day.

For the major part of the Web Services the Mean statistics of the core SAS® processing last less than 10 seconds, except "Delays" averaging to about 1 minute. The complete range is from 0 seconds up to 12 minutes. A comparison between the numbers in figure 16 and 17 displays the overhead caused by the Web Services Provider. The overhead is normally a matter of seconds regarding the different statistics. This is not a perfect 1-to-1 match comparison unveiling that the No. of Runs in SAS® actually is slightly higher than in Tomcat. A couple of the Max statistics regarding SAS® display higher numbers than Tomcat, implying that some of the heavy runs in SAS® were not caught by the analysis regarding Tomcat. Influential circumstances about performance will be discussed later.

	No. of Runs	Time			
		Mean	Min	Max	StdDev
Preliminary freight calculations	11.474	0:00:07.92	0:00:00.00	0:02:53.00	0:00:11.75
Delays	510	0:00:54.96	0:00:01.00	0:03:20.00	0:00:30.35
Delay alerts	34	0:00:02.06	0:00:00.00	0:00:08.00	0:00:02.15
Detailed freight statistics	5.922	0:00:08.44	0:00:00.00	0:04:23.00	0:00:14.04
Ad hoc report	76	0:00:06.57	0:00:00.00	0:00:16.00	0:00:03.95
Shipment control	4.600	0:00:02.64	0:00:00.00	0:01:24.00	0:00:02.82
Packages delivered	2.281	0:00:13.19	0:00:00.00	0:11:54.00	0:00:37.62
Grand Statistics	24.897	0:00:08.50	0:00:00.00	0:11:54.00	0:00:17.70

Figure 15: SAS® Log Statistics from Web Services running on Unix (FA Mart), from November 7, 2008 to January 6, 2009

	No. of Runs	Time			
		Mean	Min	Max	StdDev
Preliminary freight calculations	184	0:00:08.29	0:00:01.00	0:01:58.00	0:00:11.66
Delays	7	0:01:14.29	0:01:05.00	0:01:31.00	0:00:08.16
Delay alerts	1	0:00:05.00	0:00:05.00	0:00:05.00	.
Detailed freight statistics	82	0:00:10.18	0:00:00.00	0:02:57.00	0:00:21.08
Ad hoc report	1	0:00:07.00	0:00:07.00	0:00:07.00	.
Shipment control	72	0:00:03.43	0:00:00.00	0:00:15.00	0:00:02.62
Packages delivered	29	0:00:14.93	0:00:01.00	0:01:51.00	0:00:22.02
Grand Statistics	376	0:00:09.50	0:00:00.00	0:02:57.00	0:00:17.01

Figure 16: SAS® Log Statistics from Web Services running on Unix (FA Mart), from 00:00 to 13:00 on January 7, 2009

	No. of Runs	Time			
		Mean	Min	Max	StdDev
Preliminary freight calculations	178	0:00:08.62	0:00:01.00	0:01:32.00	0:00:08.68
Delays	7	0:01:15.86	0:01:06.00	0:01:31.00	0:00:07.86
Delay alerts	1	0:00:07.00	0:00:07.00	0:00:07.00	.
Detailed freight statistics	75	0:00:09.11	0:00:01.00	0:01:05.00	0:00:10.09
Ad hoc report	1	0:00:09.00	0:00:09.00	0:00:09.00	.
Shipment control	68	0:00:04.53	0:00:01.00	0:00:20.00	0:00:03.24
Packages delivered	28	0:00:17.00	0:00:03.00	0:01:51.00	0:00:23.55
Grand Statistics	358	0:00:09.91	0:00:01.00	0:01:51.00	0:00:14.11

Figure 17: Web Application Server Statistics (Tomcat on Windows) from Web Services running on Unix (FA Mart), from 00:00 to 13:00 on January 7, 2009

DISCUSSION

WHAT'S IN SAS® BI WEB SERVICES FOR YOU & ME?

As shown in Figure 2, SAS® Stored Processes is the core of the SAS® BI Web Services interface. Not only does this make it easy to create new SAS® jobs and models as services in a Service Oriented Architecture (SOA), it is often easy to service enable existing SAS® jobs and models as well. This makes it possible to make all the effort put into SAS® code, often during several years, available in a new and modern technology and architecture.

A nice example in this context is the SFOR reports which originally were manually submitted on a mainframe computer prior to the "IKT 2005-2006" program. The first development cycle was to make the reports Web-enabled on Windows to the internal users at Bring Logistics, while the second cycle was to make a subset of these reports available to customers on extranet through web services consumed by the Microsoft .Net application.

COMPLEX TECHNOLOGY…? UNDERSTANDING THE CHAIN – AND YOUR PART OF IT

Even though it is both possible and quite easy to make existing SAS® code available as Web Services, there are conditions that should be taken into consideration. In a SOA, a service is not just an isolated job; it is probably part of a chain of jobs, a business process. This does not imply that the SAS® developer has to know every detail about the entire process, but an understanding of the SAS® service's role and part of the process is required. A practical example of this is logging and following one transaction through the entire process. Since several systems and components are involved, finding an error might prove difficult.

In many aspects, including SAS® developers and users, practical experiences from utilizing Service Oriented Architectures are limited, at least compared to all the fuzz associated to it. This influences not only technical areas, but also administrative issues such as expectation management. For example the initial requirements may state that average execution time should be less than seven seconds. When the process is put into production, it turns out that average execution time is about two seconds. After a while, the customer might express disappointment since execution time never is less than one second - although this was never the original requirement.

DEMANDING CHALLENGES & LESSONS LEARNED

Compared to traditional reporting solutions solely using the SAS® system and perhaps some 3rd party middleware technologies, "Min Logistikk" is a pretty complex technological solution.

The main points of criticism from users at Bring Logistics and their customers are the following:

- Performance on
  - updating the data mart, which happens once through the night
  - running the reports
- The process of implementing new reports is considered to be lengthy and long-winded
- The desire for reports on close to "real time" data has increased substantially

Data Integration & Data Storage

There are certainly some "buttons-to-press" and "switches-to-toggle" regarding performance in such a compound solution. Before presenting the contents of a request for change that was proposed some time ago regarding the SAS® parts later in the paper, some other aspects need to be discussed.

The major part of the web services running on Unix query SQL views, and some are actually also querying multiple SQL views. Using physical tables properly indexed according to the requests from the reports would probably increase the performance significantly regarding selected the web services running on Unix. A couple of arguments against converting the SQL Views to physical tables are that the update process then would last longer, and more disk space would be required. Therefore, there is a trade off between reporting performance on one hand - and DWH-update performance and disk space on the other.

Microsoft .Net Application

As with all projects, improvement areas are discovered at the end of the project that could have been done better. Also, due to the fact that this project is four years old, new technologies are available that could have made the project implementation simpler.

An example area of an identified improvement is the synchronous pattern for sending out the subscription reports. It works like this:

- The windows service checks at set time intervals if there is any report subscriptions that need to be run.
- The windows service then loops through all of these subscriptions one by one
- For each subscription the report is run, formatted and sent

On a normal day this works fine, but what happens when the customer calls and says that the report has not arrived?
You need to check several systems to find out why:

- Is the subscription active, does it have the correct address?
- Is there a problem with the subscription service?
- Is there a problem with eTrust?
- Is there a problem with eConnect?
- Is there a problem with the Data warehouse?
- Is there a problem with the mail server?

There is no functionality to retry a subscription.

A better solution would be to have a queue between each system. When the subscription service runs it loads all the reports into the queue. At each stage the subscription is moved to the next queue and that movement is logged. It would then be possible to build a management page where the user could see the status of all subscriptions and have then possibility to resend / retry a subscription.

The solution does neither have any synchronization between the update of the data mart and when the subscription reports are being executed. A Microsoft .Net batch job executes the subscription reports at a fixed time of the day. A couple of options to obtain synchronization:

- Table logging – the Microsoft .Net application could e.g. iteratively call a web service querying SAS® dictionary information or dedicated SAS® application metadata for the last modification date and time - until a certain set of conditions occur - before executing the distribution the subscription reports.
- Report bursting - pushing the subscription reports from inside SAS® after completion of the data mart update. Probably not a very popular solution at Avenir AS which is the actual service provider for the “Min Logistikk” reporting solution, but SAS® would still have to know which subscribers to send the reports to – and that would imply some interaction with the Microsoft .Net application anyway …

Utilization & Performance (SAS® BI Web Services)

There were 8.750 registered users for “Min Logistikk” as of January 20, 2009. 2875 distinct users logged on to the application throughout the last calendar year. 940 subscription reports are being run and sent every day, implying that more sources than the SAS® BI Web Services are being are queried (ref. figure 11). The total Utilization of the SAS® BI Web Services is still considered to be pretty high, constituting of more than 500 reports per working day. It is important to be aware of the fact that the numbers presented are limited to the web services running on Unix. As there are 3 web services (SAS® Stored Processes) running Windows as well, the real number is higher. The Windows web services were omitted from the performance analyses as the performance problems primarily exist on Unix.

There are certainly some influential circumstances regarding Performance that need to be discussed. The figures 15-17 show the fact that the “Delays” web service has the poorest performance. This web service queries a physical SAS® table with 3 million rows and 44 column with a composite index on two column. A brief test querying the table using a typical set of input parameters, unveiled the fact that changing the order of the columns in the composite index increased the performance from e.g. 16 to 0.16 seconds.

The other most important circumstances influential to the performance are considered to be the utilization of the different servers. All the Business Intelligence applications and solutions of Bring Logistics that is mentioned in this paper are running on two servers, i.e. “KOS & IR”, “SFOR”, “LM Rapp.” additional to “Min Logistikk”. As the performance issues of this solution are mainly related to the Unix platform, where the workload apparently has exceeded the capacity of the server, a brand new and more powerful Unix-server will soon be available in production at Norway Post. The Windows server running both the Web Services Provider and some of the actual web services is the main Business Intelligence server for Bring. There are other demanding applications running on that server implying a pretty constant heavy load on both the Tomcat Servlet Container and the SAS® Stored Process Server using a cluster of 22 connections (i.e. SAS® sessions) in the current setup. Periodically heavy requests from the other application might explain the overhead by the Web Services Provider in some situations.

A request for change was proposed some time ago regarding the SAS® parts of the solution, but was rejected primarily because of the worst case scenario estimate of the implementation. The request for change included the following activities to boost the performance from SAS®:

- Optimize the data model of the data mart to better fit existing and future report requirements
- Develop new data integration processes from the Logistics Engine by using IBM MQ Series to the data mart to facilitate more frequent updates, e.g. every 5 min. like the update frequency for the KOS & IR application
- Change data storage from SAS® tables to SAS® Scalable Performance Data Server tables
- Review the web services' SAS® programs for possible modification to improve performance
- Modify the SAS® parts of the solution to facilitate a greater flexibility for some of the subscription reports by setting them up to run whenever wanted in a 24*7 SLA-framework.

Real life experiences from other BI-applications and BI-solutions in the same environment, e.g. the "KOS & IR" application, have proved that the changes above would have caused significant improvements on:

- Update frequency of data
- Response time on interactive reports
- Flexibility regarding the subscription reports

That means greater flexibility and scalability to meet future demands from customers.

WHAT ABOUT SECURITY…?

Did the solution really provide security …? No hackers have been engaged to check this out yet.
But, there are limitations regarding what might be extracted in order to limit unwanted coupling of information.

The security implementation does not exempt Bring Logistics' customers from a security mindset based on common sense. For example if one employee leaves to join a competitor, it is important to make sure that his/her user id is disabled from accessing the solution.

Finally, the security solution was initially approved both by ErgoGroup - Systems Integration secondly by the security department at Norway Post, implying that the residual security is acceptable.

CHANGES AND ENHANCEMENTS IN SAS® 9.2

SAS® 9.2 contains several significant changes regarding Web Services. These changes are both available in SAS® 9.2 phase 1 which was available by the time this paper was authored, and SAS® 9.2 phase 2 which is expected to be released quarter 1, 2009.

The enhancements in SAS® 9.2 phase 1 are mostly related to SAS® as a Web Service consumer:

- New PROCs have been added (e.g. PROC WSDL and PROC SOAP)
- The XML LIBNAME engine can interpret Web Services responses automatically
- More advanced and better support security standards (WS-Security)
- XML output formatting (using XMLMap)
- XSLT support, a.o.

SAS® 9.2 phase 2 enhancements are focused on improving abilities as service provider. The SAS® BI Web Services enables SAS® Stored Process to be interfaced as Web Service interface, not limited to XMLA like SAS® 9.1.3. Enhanced security and attachment support is added.

All improvements makes positioning SAS® in a SOA easier and adequate, for details see: *Jahn, D. 2008: Using SAS® BI Web Services and PROC SOAP in a Service-Oriented Architecture*

The solution could have benefited from at least one of the SAS® 9.2 improvements; the ability to expose a Stored Process interface as a Web Service interface. Since it was not desirable that the Microsoft .Net application should utilize the XMLA standard, the eConnect integration hub does not only enforce security, it also acts as a service proxy, enabling another Web Service interface than the XMLA interface SAS® provides as standard. If the enhanced interface capabilities in SAS® 9.2 had been available at the implementation time of the solution in question, eConnect should not have to provide the report service proxies.

The extended security capabilities in SAS® BI Web Services in SAS® 9.2 might also have impact on how security was implemented. This would require more detailed studies before any conclusions.

CONCLUSION

On the positive side, the following might be concluded regarding the “Min Logistikk” Reporting Solution:

- The solution delivers information securely to Bring Logistics and their customers on extranet, providing the ability to address and investigate problems and issues in order to be in control of the Supply Chain and to conduct proactive Customer Relation Management.
- The solution has moved from ”nice-to-have” to ”mission-critical” for Bring Logistics and corporate customers.
- The project team mastered the challenge of integration using unconventional approaches and innovative technologies in a complex Service Oriented Architecture
- The project was a successful co-operative effort, i.e. Bring Logistics + BearingPoint + Avenir AS + Know IT Information Management AS + Ergo Group AS + SAS Institute AS

The solution has contributed to the fulfillment of Bring & Norway Post targets regarding increased productivity, a more efficient organization, better reports and a better user interface.

On the negative side, there is still some potential for improvements regarding:

- Performance, Flexibility, Complexity

REFERENCES

Jahn, D. 2008:
Paper 310-2008: Using SAS® BI Web Services and PROC SOAP in a Service-Oriented Architecture, *SAS Global Forum 2008*

SAS Institute Inc. 2008:
SAS® Integration Technologies: Developers Guide
http://support.sas.com/rnd/itech/doc9/dev_guide/websrvcs/index.html
http://support.sas.com/rnd/itech/doc9/dev_guide/websrvcs/decide.html.

W3C (Wikipedia)
http://en.wikipedia.org/wiki/Interoperability
http://en.wikipedia.org/wiki/Machine_to_Machine
http://en.wikipedia.org/wiki/Computer_network

ACKNOWLEDGEMENTS

Thanks to Bring Logistics for choosing Avenir AS, Oslo and Know IT SAS Information Management AS, Oslo for the assignment “Min Logistikk”. Ergo Group – Systems Integration has a long term assignment of being in charge of the Security on behalf of Norway Post. Special thanks to the following key persons in this context:

Bring Logistics:
- Dagfinn Halvorsen, Business Requirements

BearingPoint:
- Martin Watne, Solution Coordinator/Architect

Avenir AS:
- Shiraz Ali Bhaiji, Solution & Technical Architect Microsoft .Net + Initial Dev.
- Arnfinn Løv-Mikkelsen, Project Manager Microsoft .Net
- John Hansen, Developer Microsoft .Net
- Lars Høiby, Daily Operations & Maintenance of Microsoft .Net

Know IT Information Management, Oslo:
- Truls Andersen, Solution & Technical Architect SAS® + Configuration SAS®
- Ketil Eidsaunet, Developer SAS® Stored Processes
- Dagfinn Larsen, Developer SFOR Data Integration Processes
- Lisbeth Laahne, Project Manager & Daily Operations & Maintenance of the SAS® parts

SAS Institute AS, Norway:
- Frode Langseth, Solution & Technical Architect SAS® Integration & Security

ErgoGroup AS - Systems Integration:
- Jon Ole Norleman, Solution Architect Web Access Control with eConnect, eTrust
- Stein Nyfløt, Developer Web Access Control with eConnect, eTrust
- Thomas Kjeldahl Nilsson, Developer Web Access Control with eConnect, eTrust
- Merete Ødegaard Thommesen, Project Manager Web Access Control

ErgoGroup AS - Data Warehousing:
- Bente Moe, Developer FA Mart Data Integration Processes
- Solveig Thue Johansen, Developer FA Mart Data Integration Processes
- Seamus McKenna, Developer FA Mart + Misc. Testing

Other consultants have also been involved in this project to a varying extend. Thanks to them all.

CONTACT INFORMATION

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

Truls Andersen, Dr. Scient. / Senior Advisor
Know IT SAS Software Solutions AS
Lille Grensen 5
0159 Oslo
Norway
+47 9055 6318
truls.andersen@knowit.se

Frode Langseth, Senior Advisor
SAS Institute Norway AS
Postboks 2666 Solli
0203 Oslo
NORWAY
+47 4145 1524
frode.langseth@nor.sas.com

Shiraz Ali Bhaiji, Manager
Avenir AS
Nedre Skøyen vei 26
0276 Oslo
NORWAY
+47 9010 8273
shiraz@bhaiji.com

Stein Nyfløt, Senior Advisor
ErgoGroup AS
Nydalsveien 28
Postboks 4364 Nydalen
0402 Oslo
NORWAY
+47 9154 7352
stein.nyflot@ergogroup.no