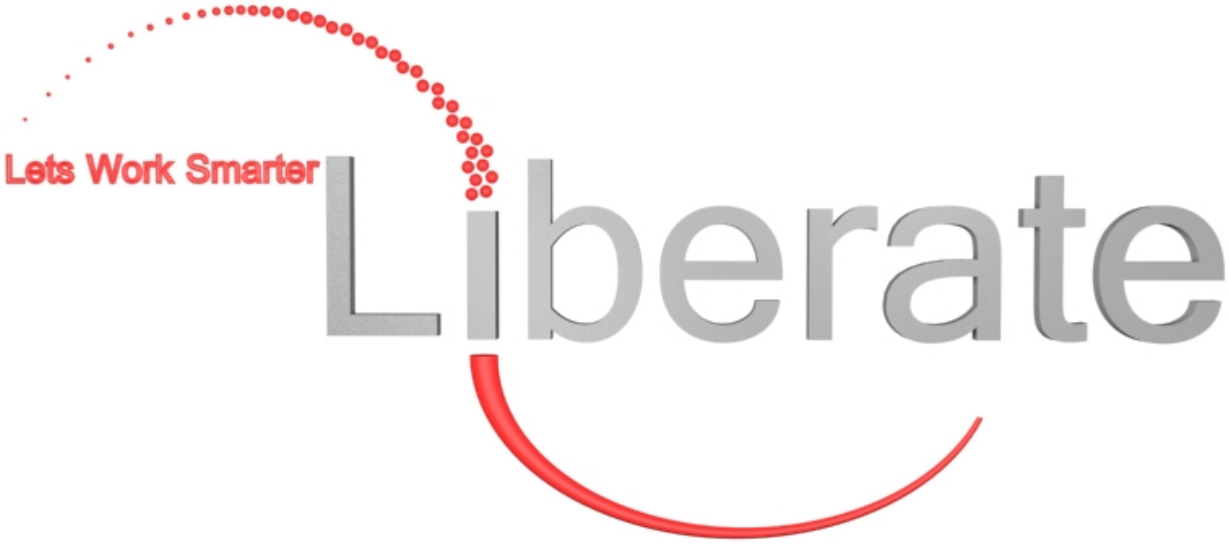
Paper AD06 – PHUSE 2007

Liberate, a SAS reporting system



David Royle
I.T Director

Huntingdon Life Sciences Ltd
Huntingdon U.K.

Contents

1. Abstract

This paper describes a reporting architecture designed and implemented at Huntingdon Life Sciences. The architecture called **Liberate** has been designed to build, store, manage and execute XML scripted SAS data-reports. The execution environment is unique and provides a distributed set of background services that operate across a network.

2. Introduction

The purpose of the paper is to examine the design of the Liberate architecture and describe how SAS®, Microsoft Office®, SQL and Adobe® have been used as background distributed services. The paper discusses how the system has utilized technologies like Oracle®, DCOM, XML, Multi-tiered architecture and networking into the reporting framework.
Particular attention is given to the technique used to populate Microsoft Word template documents from XML data that has been generated using BASE SAS scripts.

The Liberate system is distributed, multi-layered and complex; the paper concentrates on key elements of the architecture rather than describing all the modules in detail.
The motivation behind the system, design approach, detailed examination of key components and operation of the framework is also covered in the paper.

3. Objectives

"***Liberate the data, store in a central repository and provide flexible reporting and analyses from this repository***"

The system was split into three high level projects:-

1. **ETL** (Extraction, Transformation and Loading) Links to major systems like Xybion®, ClinAxys®, WATSON-LIMS®, DEBRA®, EMKA®, Po-Ne-Mah®, & others.
2. **Oracle Data-warehouse** (Oracle database repository to store and manage the loaded data)
3. **The reporting architecture called Liberate**; to connect to databases and produce formatted data tables.

This paper concentrates on the third project the reporting architecture called **Liberate**, the high level objectives for this project were:-

- Use SAS as the statistical and data-processing engine without requiring SAS on the desktop ~ create a SAS background processing service
- Provide flexible report formats from SAS into Word, Excel, Adobe-PDF, HTML and others
- Make use of Word templates for data table design, rather than SAS-ODS
- Background processing and scheduling capabilities
- Load balancing and distribution across a network
- Provide an interface for report developers to create, store and maintain reports as SAS-XML scripts
- Provide a mechanism to include a variety of individual report options and run-time parameters
- Provide a mechanism to include data exclusions and automated report footnotes
- Provide an easy interface for users to run, schedule and group individual data-tables under a study or project

4. Modular Approach

The Liberate reporting project had high expectations for automation, efficiency and was technically complex.

Building a solution to meet these requirements meant breaking the project down into sub-systems that provide a set of related services, then create a framework to manage and control these sub-systems.

The Liberate reporting sub-systems:-

- **User Interface Layer** (Graphical thin-client screens to manage data reports)
- **Distributed Component Layer** (DCOM application services in the middle-tier)
- **Background Service Layer** (Remote Worker programs running SAS, Office, Adobe & more)
- **Persistence Layer** (Oracle Database Schemas)

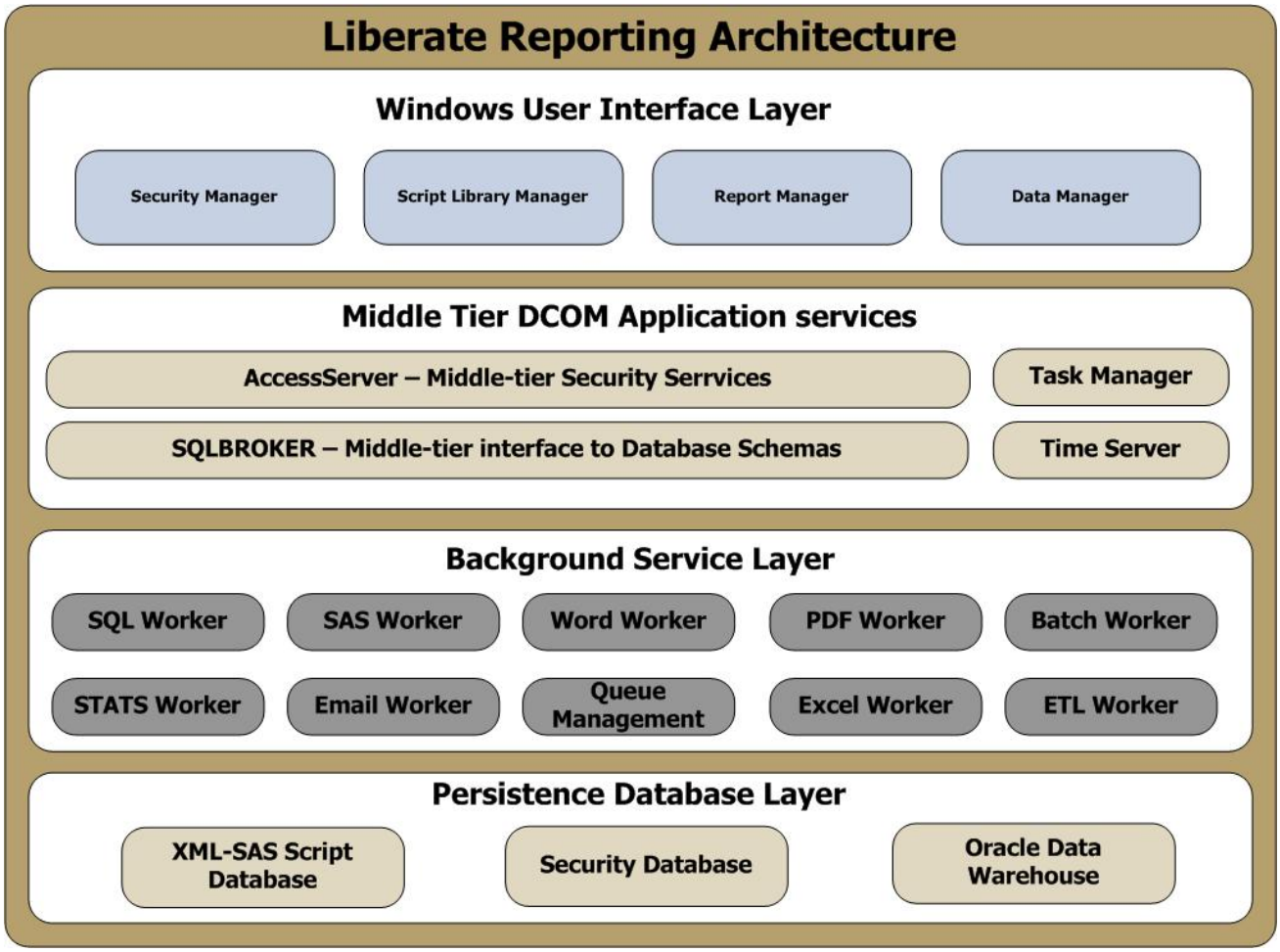


Figure 1

The user interface and DCOM layers form the development foundation for the system but are not part of the actual reporting process. The key layer in the system is the **Background Service Layer**. This layer contains components that collaborate to produce data-analysis and eventually formatted output. The user interface and DCOM layers are briefly described, but the key concepts are the SAS and Word workers; these are described in more depth. These two components collaborate to produce a highly customizable and flexible data reporting framework.
The key design goals of the system are loose coupling and high cohesion, each layer is loosely connected and each component has a very specific role.

5. User Interface Layer

This layer was developed using the Borland® Delphi Developer tools. The client side programs are all thin-client forms that require no database connectivity software or application software like Adobe®, SAS or Microsoft Office®.
User interface forms connect to the DCOM (Distributed Component Object Model Layer) servers across the network and the DCOM servers act as a conduit between the background services and persistence data layer (Oracle Schemas).
The location of the DCOM servers is held in a local configuration file and is usually an IP address or network name.
The benefits of this model include:-

- Small footprint deployment to the desktop.
- Reusability, the Security DCOM server encapsulates all authentication methods and is used by all interface programs.
- Access to Oracle database schemas uses the **SQLBroker** DCOM server, which manages connections and execution of stored procedures and dataset retrieval.
- All SQL statements are held in the DCOM layer outside of the client programs in XML files.
- Licenses for SAS and Adobe are not required on the desktop, reducing costs.

The user interfaces contain very little business logic and are loosely coupled to the background service layer and persistence database schemas. The code to connect and call remote COM methods is straightforward and utilizes early binding through client side type libraries.
The location of the remote DCOM servers on the network is transparent to the client code. One or more physical servers can be setup to host the DCOM servers; this can give resilience and load balancing to the system.

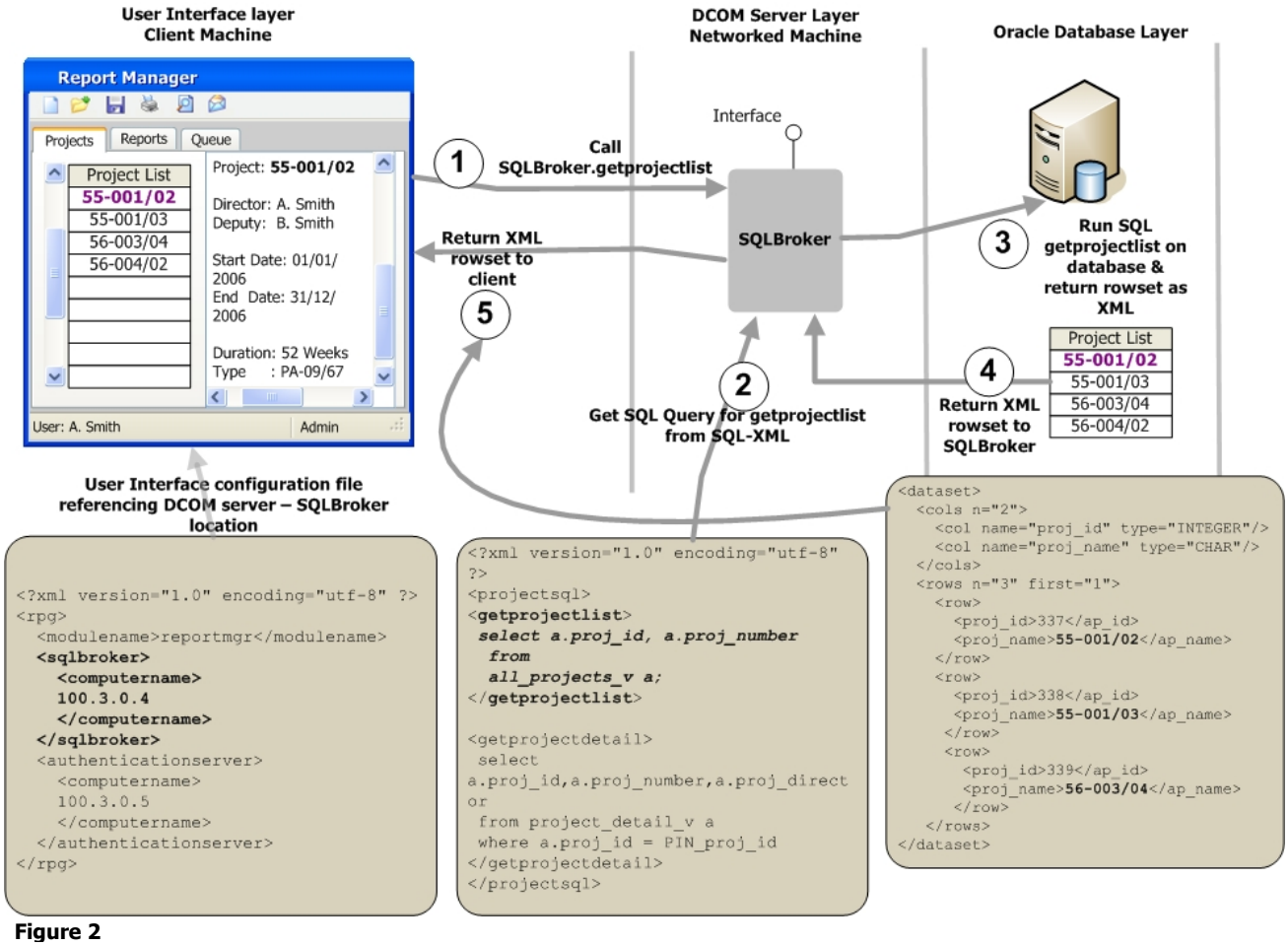


Figure 2

6. Distributed Component Object Model Layer

COM is a Microsoft technology that provides a framework for integrating components. These components must adhere to a standard but can be developed in any programming language implementing the standard, which allows interoperability. DCOM is an extension to COM that allows components to interact over a network. COM components run on the same physical machine but in a different address space to the executing program.

In Liberate the DCOM layer acts as an interface to database schemas, security and task management. Client code uses these COM server interfaces in this layer to access remote methods. The COM servers provide methods to connect and manage databases, authenticate and manage users, and manage Liberate report tasks in the database.

6.1. SQLBroker - Database Services

This DCOM server acts as an interface to all Oracle (or other) database schemas.
All SQL statements are stored in this layer in XML documents.
When a client program needs to retrieve data or execute a stored procedure on the database the SQL statements are fetched from the XML document, parsed, parameters substituted and the SQL executed. The resulting rowset is then returned as an XML dataset.

An example of a typical **SQL-XML** file:-

```
<?xml version="1.0" encoding="utf-8" ?>

<accessgate displayname="AccessGate">

<changeuserpassword displayname="changeuserpassword">
begin
  ChangeUserPassword(:PNIn_accid, :PCIn15_newpw,
  :PNIn_peid, :PNIn_immediate, :PNOut_result);
end;
</changeuserpassword>

<validateuser displayname="validateuser">
begin
ValidateUser( :PNIn_sysid, :PCIn50_name, :PCIn15_pw,
:PNOut_result );
end;
</validateuser>

</accessgate>
```

Figure 3

Database connections are expensive in terms of server-side resource, the SQLBroker DCOM server is designed to pool database connections and maintain state between current clients and the database. All access to database resource is through this server interface.
The client initiates a connection with SQLBroker and requests a database connection. If a connection already exists in the pool a reference to this connection is returned to the client to use for future methods, otherwise a new database connection is established with the database.

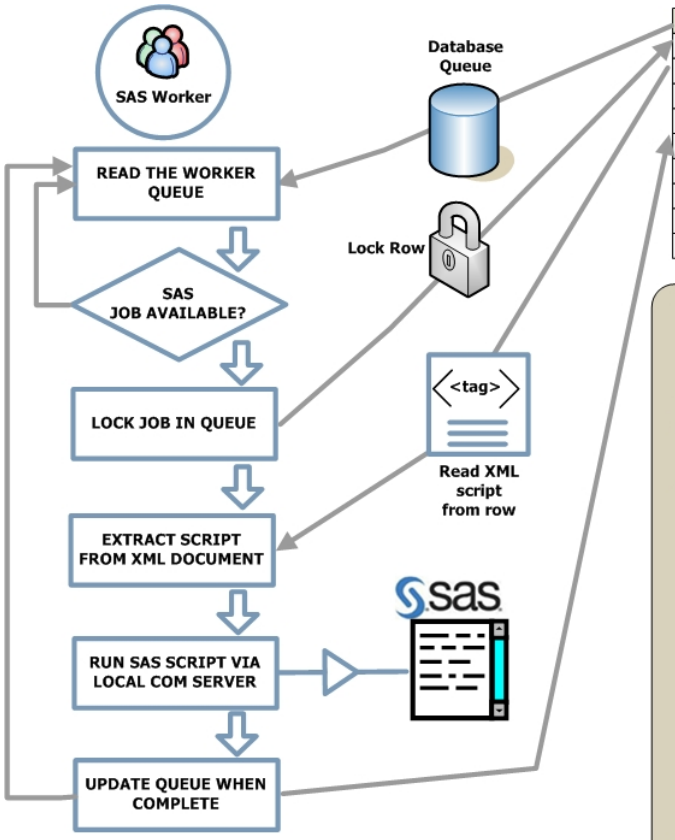
6.2. AccessServer - Security services

The AccessServer DCOM server provides an interface to the Oracle security schema. This server in conjunction with the Security Manager UI provides user, role, action and system security management. All user interfaces security services are catered for by this DCOM Server. Client applications makes security requests without knowing anything about the underlying database schema.

7. Background service layer

This layer contains the key components of the Liberate reporting architecture, **Worker** services. Worker services are background programs that reside on remote networked computers.
A worker program is a Windows executable or service that continually loops, and scans a database queue for jobs. Each worker is programmed to carry out a specific task. These tasks vary from running SAS programs, extracting data using SQL, converting documents to PDF, creating Word or Excel documents and sending e-mails.

Once a job of the correct type is found in the queue the worker will extract the job details held in an XML document and carry out the instructions contained in the embedded script.



Worker Queue
Request:456211 SAS Job
Request:456723 Word Job
Request:457810 PDF Job
Request:458712 SAS Job
Request:458713 SAS Job
Request:458734 Email Job
Request:458755 Unix Job
Request:458798 CMD Job
...

Typical XML SAS Script. ***Note the SAS parameters in { }***

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
 <task id="5306" version="1.0.1" name="Food consumption">
 <package_template id="2936" name="Food consumption">
  <script="SAS" version="8.2">
/* Connect to database and retrieve
   food, housing & mortality data by study */
proc sql;
  connect to oracle as myconn
  (user='{dbusername}' password='{dbpassword}'
path='{dbname}');
  %put &sqlxmsg;
  create table food_consumption_data  as
  select * from connection to myconn
  (select * from toxdata.food_v
     where study = '{study}');

  create table housing_data as
  select * from connection to myconn
  (select * from toxdata.housing_v
  where study = '{study}');

  create table mortality_data as
  select * from connection to myconn
  (select * from toxdata.mortality_v
     where study = '{study}');

  %put &sqlxmsg;
  disconnect from myconn;
quit;
</script>
</package_template id>
</task id>
```

Figure 4

To use a worker, simply place a job (XML document) into the worker queue. The first worker of the correct type to find the job in the queue will carry out that job as instructed.
This ensures that the worker programs are loosely coupled and not tied into any software protocols.
Worker programs are distributed onto the network and hosted on separate physical machines. When one worker is busy another idle worker will take over. The more workers running on the network, the more distributed the workload, the quicker the jobs are completed.

Sending individual jobs to worker programs via the database queue will get that particular script executed autonomously, but if the workers could combine and work together, then we would have a much more powerful framework to achieve more complex tasks.

A Liberate task is a series of steps (jobs) that accomplish a large piece of work. Each step is carried out by a worker (SAS, SQL, PDF, Word, and Excel). When all the steps are complete the task is finished.

7.1. Liberate tasks – combining workers

The real power of the Liberate architecture is realized when worker programs are combined to accomplish complex tasks. These tasks range from extracting data for reporting, rendering, updating databases overnight, loading data and many more processes.

These tasks are designed by developers and stored in the task database. The task itself is an XML document, with references to XML worker scripts. The output from each worker script in the task can be used as input to the next worker script to make a sequence. To produce a typical data table the task would comprise of:-

- **An extraction script in SQL** (this would extract the bodyweight data against a set of parameters)
- **A SAS processing script** (this would process the extracted data and produce an XML file)
- **A Word script** (this would use the SAS XML file and merge with a defined Word-XML template)
- **An optional PDF script** (this would render the Word document into PDF)
- **An optional email script** (this would send a message on completion, either error or success)

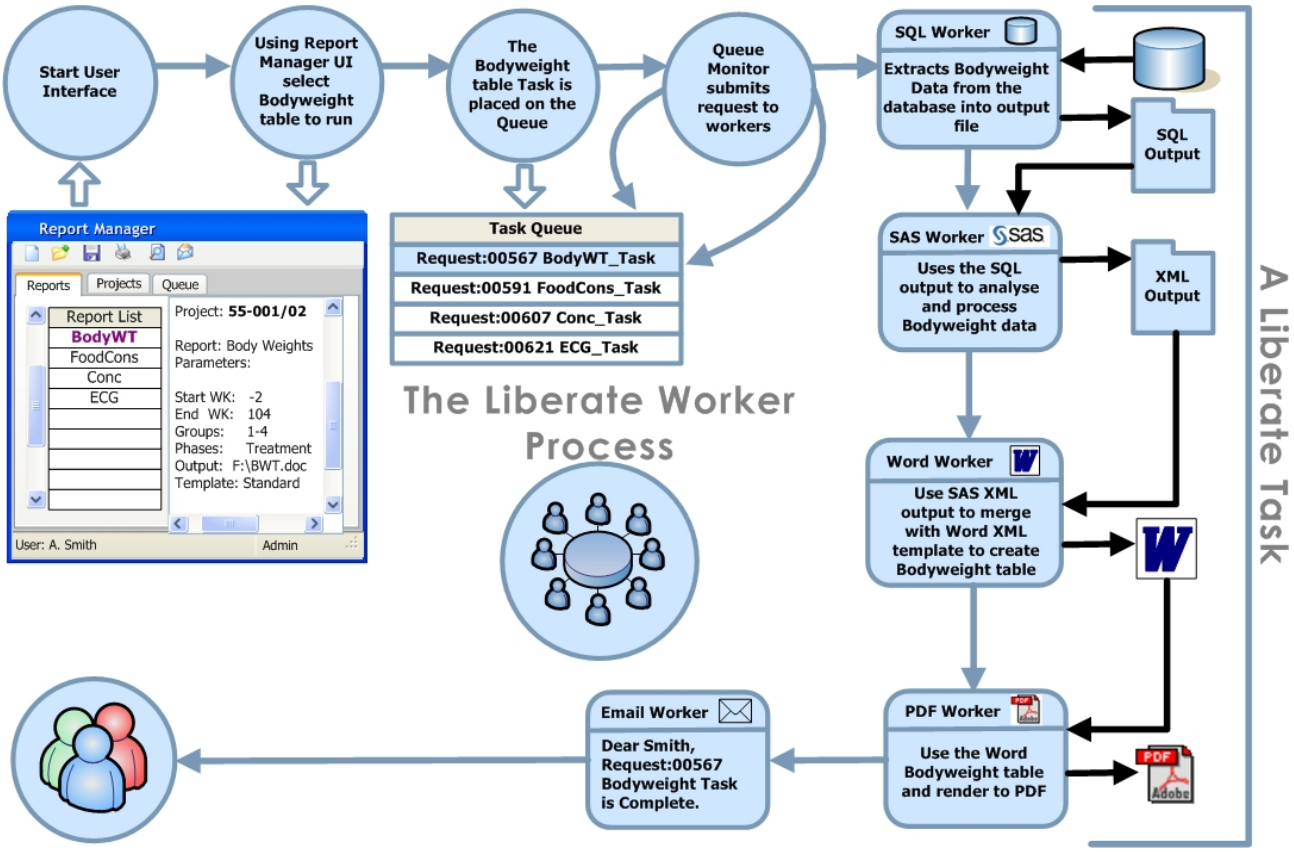


Figure 5

The task execution process occurs in the background on remote networked machines, usually in the server room. The task can be scheduled to run immediately, at a preset date/time in the future or repeatedly between two dates. All the data tables for a project/study can be setup in advance of the start date and scheduled to run at convenient points during the project.

As the load on the system increases, the wait times for the user will also increase. By introducing more worker programs onto the network the load is distributed and the wait times will decrease. Multiple workers can exist on the same physical machine, to maximize hardware resources.

8. The SAS Worker

The SAS worker interfaces with the SAS local COM server on the same physical machine.
SAS worker scripts contain more than just SAS code, they identify the job, requestor, and importantly contain parameters.
The parameters in all Worker scripts are identified by a pair of curly braces { }. Parameters are not SAS macro variables; they are text place holders in braces that require substitution by actual text values.
Parameter values are obtained by the User Interface when the job is submitted by prompting for each parameter tag in the script.
This makes the scripts very flexible, parameters in the SAS code can be placed anywhere and do not have to follow the SAS syntax rules.
When the SAS worker retrieves an XML script, it replaces all the parameters with values before submission. The SAS code is submitted to the local SAS COM server for execution.

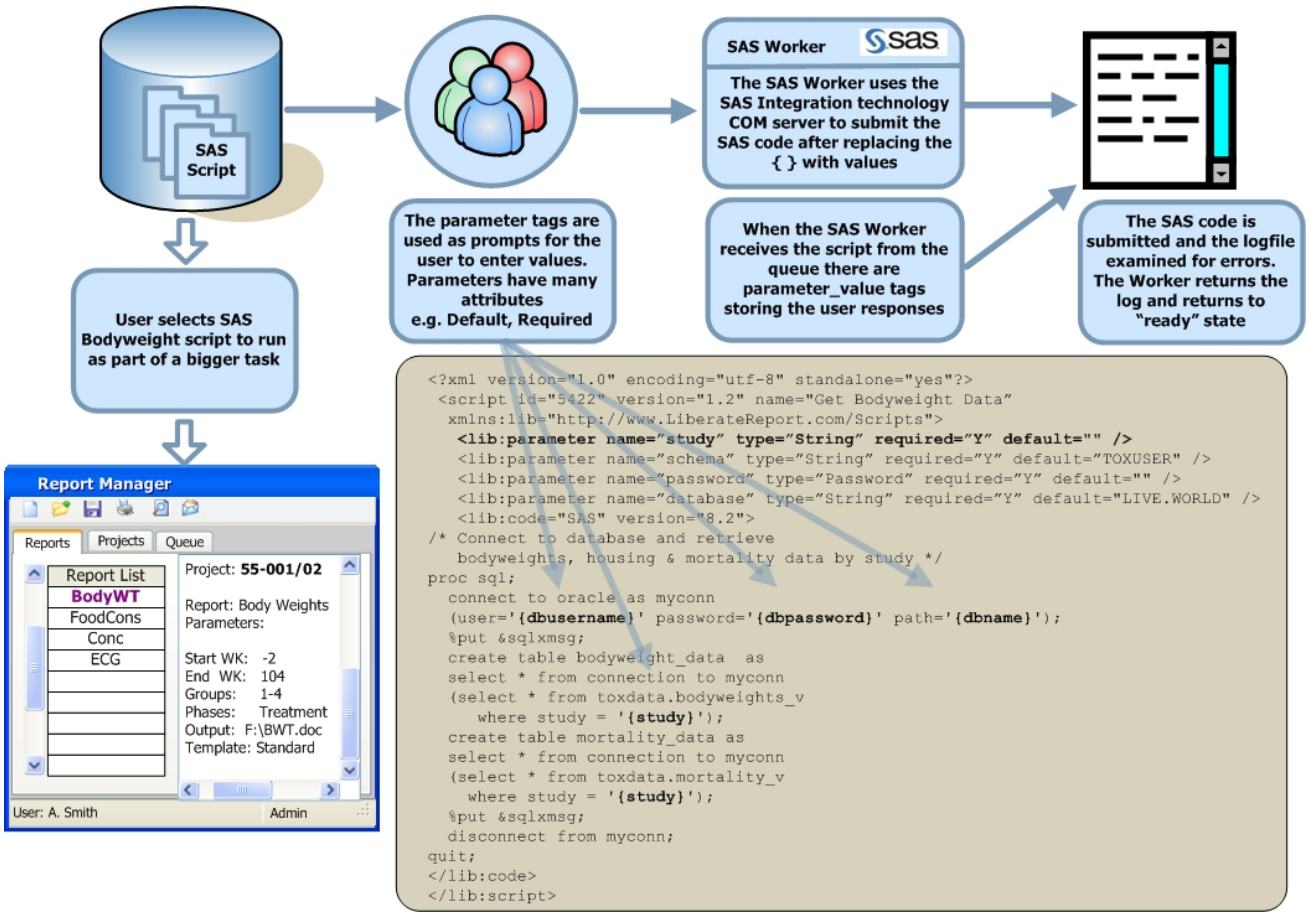


Figure 6

Liberate parameters include basic and complex types and have many attributes. If a given task involves many scripts then parameters for all the scripts are scanned at submission and duplicate names are only prompted for once, e.g. Project or Study Number.

The SAS worker only uses COM to connect to the SAS system, therefore the Worker and SAS have to be installed on the same physical machine. However to communicate with the SAS Worker in Liberate you only need to place a script in the worker queue and the first idle worker will respond and execute the code contained in the script. This effectively means that a single SAS worker instance on one machine could service the needs of many users, reducing SAS licensing costs.
If the system starts to respond poorly, introduce more SAS Workers onto the network and distribute the workload. **One SAS Worker only requires one SAS license**.

9. SAS to Word

Creating Word documents using SAS has usually involved using SAS ODS (Output Delivery System).
However this has many limitations, most organizations have standard Word templates that incorporate formatting styles, fonts, table layouts, headers, footers and many more Word objects.
SAS ODS creates reports dynamically in code and has no concept of Word or Excel templates. The degree of control with ODS is not granular enough for complex data tables in Word, that may require merged cells, split cells and specific cell, row, column and text formatting.
Liberate incorporates the concept of an empty Word template that has a pre-defined table structure in the document. The Word template is then populated with the data that has been previously processed by the SAS Worker. The Word templates are stored in the database and given names. These templates then form part of the task (see Figure 5) when processing a data table.
The population of these templates is the job of the Word Worker. The Word Worker merges the SAS output file, which is in XML format and the empty Word template from the database to produce a new Word populated document. This new document has all the correct formatting, styles and layout. The document template in the database can also be updated, (within limitations) without needing to modify the SAS script.

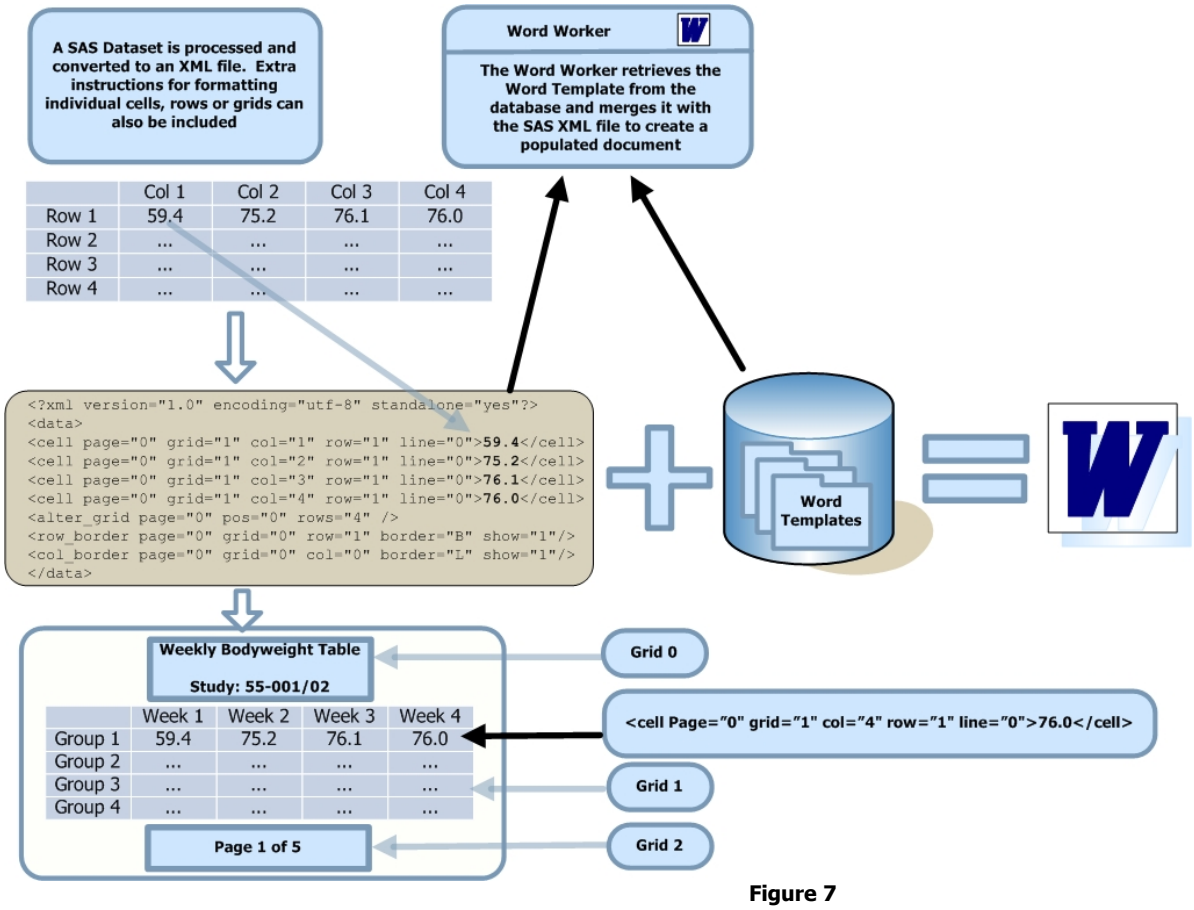


Figure 7

The XML file that SAS produces can be laid out in any order, each value from the dataset has a corresponding <cell> tag. The Word Worker reads the SAS XML file and enters the values into the correct cells as identified by the value of the attributes. The Word templates are only one page in length normally. If the SAS XML file makes reference to a cell on a page greater than zero, then a new page is cloned from the template and the value entered into the correct cell. A Word page template can contain many grids and these are referenced in order starting at zero. Extra formatting on cells, rows, columns and grids can be included in the SAS XML to control the appearance of the document.

10. Managing Worker scripts & tasks

Scripts for SAS, SQL, Word, Excel, PDF and other workers all require managing in the database. A DCOM server and User Interface are used to maintain the scripts and the tasks that combine scripts.
The Task Manager DCOM server resides in the DCOM layer (see Figure 1) and the Script Library Manager (**SLM**) is the graphical interface to this server. The SLM manages all aspects of script and task management:-

Function	Description
Script Development	Creation, Update, Reference, Parameters for all Worker types
Versioning	Version control, History, Auditing
Deployment	Testing, Submission to Workers, Production
Task Development	Creation, Update, reference to scripts in sequence
Security	Access, Privileges, Roles

The SLM provides developers with the ability to store and maintain scripts and Word/Excel templates in one environment. SAS scripts are stored and maintained in the database; version controlled and can be submitted for execution to remote SAS Workers.
Scripts for all workers can be joined together to form a Liberate task and will execute in sequence. There are no limitations to the number of scripts or type that can be used to make a task. Common routines can be designed in scripts and referenced by other scripts. This opens up the SAS development environment and adds in the powerful feature of collaborating with other Worker scripts.
When a task is submitted in the Liberate system, it receives a unique request ID and a permanent log-file is kept recording all the script code that was used and the SAS log. All the requests and log-files remain on the database and can be examined along with the parameter values, requestor and status.

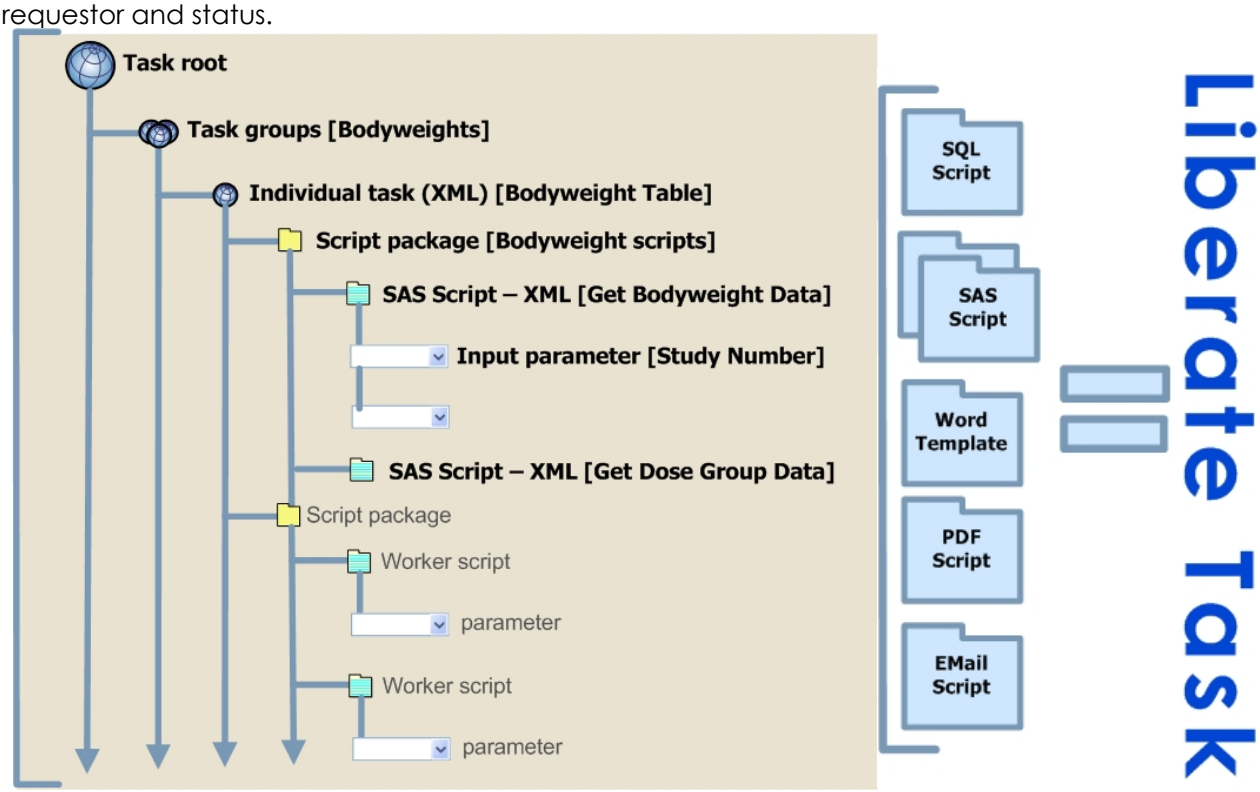


Figure 8 Scripts combined to make a task

The concept of a task being a sequence of steps and each step is represented by a Worker script plays an important role in the Liberate architecture.

11. Summary

11.1. Conclusion

The Liberate architecture embodies two main design principles, loose coupling and high cohesion. All the Worker programs are very loosely coupled into the framework; their only connection to the system is via a database queue. They also importantly only perform one function (SAS, SQL, Word, Excel, PDF etc), this allows them to be used as building blocks in complex task orientated business processes.
Splitting the system into layers and components within those layers, allows the system to be distributed across a network, utilizing available hardware.
This modular approach to development benefits maintenance, testing and deployment.
The system started out life as a reporting application, but because the framework is so flexible it can be used for many other tasks.
We currently use the system as part of the ETL for the data warehouse, producing spreadsheets from the Finance systems, performing routine database administration and many other functions.

11.2. Ongoing & Future Developments

The current system is based on the Microsoft DCOM technology, this has reached end of life and we are currently planning to upgrade the framework to use the Microsoft .Net technology platform.
The .Net platform will also allow us to make use of Web services and open up the Liberate architecture across the Internet. Essentially the Worker programs will have a Web service layer that will allow them to be used via HTTP on port 80. This will effectively mean that tasks and Worker jobs could be submitted via a website and Worker programs like SAS could be distributed across the Internet. Web service orientated architecture is based on XML which is the foundation for all Liberate scripts.

11.3. Other modules

An important part of the reporting system is the Oracle data warehouse. The database has evolved to include data from different source applications and provides the reporting system with a consistent view of data at all times. Some pre-processing of data takes place during the ETL process, but a large part of the data processing for reports takes place outside of the database using the worker programs.

The design and development of the warehouse will form the basis of a future paper.

11.4. Contact

Author: David Royle

Company: Huntingdon Life Sciences Ltd

Position: IT Director

Tel: 01480 892000 **Mob: 07738 135 886**

email: royled@ukorg.huntingdon.com

Lets Work Smarter Liberate