ABSTRACT
This paper will demonstrate how custom study documentation can easily be generated by using SAS to tap into a clinical trial database system such as Oracle Clinical. With an understanding of the Oracle Clinical database structure, what sometimes is a manually compiled (and labor intensive) document can be automated using SAS/Access to Oracle, Proc SQL and ODS. An example is given for a database structure document. Advantages over other reporting options or existing tools are mentioned.

INTRODUCTION
The need for proper documentation is ever-present in a regulated environment such as the pharmaceutical/biotechnology research industry. Part of the reliability and the controlled nature of our computer systems used in clinical research is the ability to document system specifications. One key document that is used is a database specification document, which contains information about the database tables, the variables and their types and lengths, format codes, and derivation algorithms that are part of a study database.

Clinical Data Management Systems (CDMS) such as Oracle Clinical are commonly used to support these study databases. These databases are usually exported to SAS datasets for analysis. Unfortunately, these systems do not provide adequate reports to support the documentation requirements. Custom reports are commonly developed to meet these requirements. Several reporting tools are available for developing custom reports. This paper will present how powerful SAS can be as a reporting tool in documenting a clinical database and the SAS extracts.

A simple report is desired that will document the study information, datasets used, and for each dataset the following information: Oracle table name, SAS dataset name, Oracle question name, Oracle extended attribute, SAS variable name, variable type and length, Oracle format codelist (Discrete Value Group or DVG), SAS format, and SAS label. Such a report would provide a concise document for mapping OC objects to SAS extracts as well as providing the data coding information.

CURRENT SHORTCOMINGS
It is expected that a CDMS would provide adequate reporting on its own database structures. While most systems do provide some reports, experience has shown that organizations need a highly customized report of database specifications that produces concise and meaningful documentation that integrates well into the business process.

Out-of-the-box reports for Oracle Clinical are verbose, difficult to read, unattractive and often lack the required information. In many cases the sought-after information is found on multiple reports among many other unwanted data. These reports provided by the vendor are often deemed undesirable for use in an organization’s business process.

The generated SAS code for Oracle Clinical Extract Views provides only partial extract data specifications, namely the SAS pieces of information. Most importantly, while one could refer to these files to find SAS dataset specifications, these files are not well-suited for documentation or a reporting solution. An example SAS extract code that is generated by Oracle Clinical is below.

```
libname extract 'd:\opapps\sas_view\oc4train\abc123\current\';
proc sql;
    connect to oracle(path='oc4train');
create view EXTRACT.DEM as select
    STUDY    as STUDY  label ="Clinical Study" format  $15.
    ,DCMNAME  as DCMNAME label="DCM Name" format  $16.
    ,INVSITE  as INVSITE label="Site" format  $10.
    ,PT      as PT     label="Patient"  format  $10.
    ,DCMDATE  as DCMDATE  label="DCM Date"  format  $8.
    ,VISIT    as VISIT  label="Visit"  format  10.
    ,BIRTHDT  as BIRTHDT label="Birth Date"  format  $8.
    ,SEX     as SEX   label ="Sex"  format $6.
from connection to oracle (select
    STUDY STUDY, DCMNAME DCMNAME, INVSITE INV SITE, INV INV, PT PT, DCMDATE DCMDATE,
    VISIT_VISIT, BIRTHDT BIRTHDT, SEX SEX
    from ABC123$CURRENT.DEM);
disconnect from oracle;
quit;
```
With all of the meta-data in Oracle Clinical, the preferred solution would be to query from these meta-data directly to generate the specifications document. The following example illustrates how this can be done.

**THE SOLUTION – SAS**

By using SAS/Access to Oracle, Proc SQL and ODS, it is reasonable that a SAS program could retrieve and format all of the desired information for the report from Oracle Clinical’s meta-data tables. The power of SAS easily handles the dynamic nature of the data to be included in the report and can be used to streamline the report code by using the SAS Macro utility and ODS.

Other reporting tools could also be used, but most lack the ability to handle the multi-step querying needed for this task. Stored procedures on the database side certainly could be used in conjunction with a generic reporting tool such as Crystal Reports, but that would add unneeded complexity. Also, SAS skills already found in a Clinical Data Management or Biostatistics group can be leveraged to avoid the need of learning a new reporting tool or techniques.

**ORACLE CLINICAL QUERIES**

We will first look at how to get the information out of Oracle Clinical that is needed for the report.

**ORACLE CLINICAL OBJECTS REQUIRED**

A study database in Oracle Clinical is constructed from several levels of database objects, namely Questions, Question Groups, Data Collection Modules, and Data Collection Instruments. Coded values use Discrete Values from Discrete Value Groups, similar to a SAS format. The database structure is exported to SAS through Extract View definitions. Since the objective of the desired report is to document the database in terms of SAS extracts while also including the source variables in the Oracle Clinical structure, the report will key off of the Extract View definition information in the Oracle Clinical system tables. The following objects in the Oracle Clinical database will be queried:

- Data_extract_views
- Template_columns
- View_template_questions
- View_question_mappings
- DCM_Questions
- Discrete_value_groups
- DCI_Modules
- DCIS

**SO, ORACLE CLINICAL, TELL ME ABOUT YOURSELF**

To begin, we would like to query Oracle Clinical for a list of all extract views defined for a particular study database. The SQL query to get this information is

```sql
select unique
    v.view_template_id, v.key_template_id, v.name, v.sas_name,
    decode(qg.description,Null,d.description,qg.description) description
from data_extract_views v, dcms d, dcm_question_groups dqg, question_groups qg
where v.clinical_study_id=8
    and v.view_definition_status_code='A'
    and v.dcm_id = d.dcm_id
    and v.dcm_question_group_id = dqg.dcm_question_grp_id (+)
    and dqg.question_group_id = qg.question_group_id (+)
where clinical_study_id is the Oracle Clinical internal code for the study. Once this list is obtained, a macro loop can be utilized to step through this list and query the extract definition information for each extract view.

An extract view is made up of the variables from a key template and the variables from the view template. Therefore the SQL for the view definition must query and UNION these variable lists together. The SQL query to do this is below where the macro variables &tid1 and &ktid1 hold the id values for the view template and key template respectively and &name1 and &sname1 hold the view name and SAS dataset name for a given extract view. In this example, the organization’s standard practice for database structure is to have a DCI represent one CRF page and the naming convention for DCI name is PAGExxxx, where xxxx is the zero-filled page number. With this information we can add to the specifications report the pages on which each individual variable is collected. Without such standard practices, one may need to remove the page field from these queries and the report.

```sql
select
    to_number(Null) dcm_id, to_number(Null) dcm_que_dcm_subset_sn,
    &tid1 template_id, &ktid1 key_template_id, &name1 view_name,
    &sname1 sas_ds_name, tc.display_sequence_no, tc.name oracle_name,
    tc.sas_name, tc.sas_label, 'ORACLE_VARIABLE' attribute_name,
    to_char(Null) pages, to_number(Null) pages_num,
```
Following the query from Oracle the following data step will concatenate the records of page numbers to one variable and provide some variable attributes. We also have to apply some length and format values that Oracle Clinical attributes to certain variable types during an extract job.

data extract2;
  set extracts;
  by template_id display_sequence_no pages_num dcm_que_dcm_subset_sn;
  length pages_all $1000;
  retain pages_all;
  select (attribute_name);
    when ('FULL_VALUE_TEXT','DVG_LONG_VALUE') do;
      data_type_length='CHAR(200)';
      sas_format='$$200.';
    end;
    when ('DVG_NUMBER') do;
      data_type_length='NUMBER';
      sas_format='10.';
    end;
    otherwise;
  end;
run;
if first.display_sequence_no then do;
  pages_all=pages;
end;
else if pages ne '' then pages_all=trim(pages_all)||', '||left(trim(pages));
if last.display_sequence_no then output;
drop pages pages_num;
label oracle_name='Variable Name'
sas_name='SAS Name'
sas_label='Label'
attribute_name='OC Data Location'
data_type_length='Field Type/Length'
dvg='OC Format'
sas_format='SAS Format'
pages_all='CRF Page'
view_name='OC Extract Name'
sas_ds_name='SAS Dataset Name'
;
run;

A MACRO TO AUTOMATE THE REPORT
A macro can easily be created that will perform the entire operation for a given study id. An example macro call is
%oc_extract_defs(db=oc4prod,user=test,pass=test,csid=12345,saveloc=studyABC\docs\)
The macro call includes the database connection information (OC instance name, username and password), clinical study id, and the location for the macro to save the resulting report.

The report macro uses ODS to generate a title page, a contents page, and the specifications pages in a nicely formatted RTF document. (Note: PDF can also be used, but since SAS8.2 does not embed fonts within the PDF documents, it may be preferred to use RTF. Subsequently, if desired, an RTF can be converted to PDF using the Acrobat Distiller printer driver).

EXAMPLE OUTPUT

<table>
<thead>
<tr>
<th>Protocol No.:</th>
<th>ABC123</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Name:</td>
<td>A double-blind, placebo controlled, randomized, etc etc etc</td>
</tr>
<tr>
<td>Date of Plan:</td>
<td>December 11, 2003</td>
</tr>
<tr>
<td>Database Application:</td>
<td>Oracle Clinical v4.0.3</td>
</tr>
<tr>
<td>Data Rights for Entry on Database:</td>
<td>See Attached List and Annotated CRF</td>
</tr>
<tr>
<td>Data Validation Methodology (double-entry, 100% visual check, etc):</td>
<td>2nd PASS ENTRY</td>
</tr>
</tbody>
</table>

DATA SETS:

| DEMO | Demographics |
| AX | Abuse Events |
| DQED | Prior/Concurrent Medications/Therapies |
| DSP | Study Completion or Entry Data dictionary |
| ROC | ROC |
| EXCL | Exclusion Criteria |
| ICL | Inclusion Criteria |
| LAB | Laboratory |
| MEDI | Medical History |
| PE | Physical Examination |
| VIT | Vital Signs |
### MACRO CODE

%macro oc_extract_defs(db=,user=,pass=,csid=,dbvers=4.0.3,dval_meth=2nd PASS ENTRY, deptroot=\server\users\database_operations, saveloc=);

%* OC_EXTRACT_DEFS macro to generate oc to SAS db specifications document,
%* PharmaSUG *;
%* Author: Kyle McBride, Instat Consulting Inc.
%* Date: 15-DEC-2003
%* This code may be modified and used free of charge provided a reference to the original
%* author is maintained.
%**query OC to get list of extract views for a given study**;
proc sql noprint;
connect to oracle(path=&db user=&user pass=&pass) ;
create table WORK.extract_views as select *
from connection to oracle (select unique
v.view_template_id, v.key_template_id, v.name, v.sas_name,
decode(qg.description,Null,d.description,qg.description) description
from data_extract_views v, dcms d, dcm_question_groups dqg, question_groups qg
where v.clinical_study_id=&csid
and v.view_definition_status_code='A'
and v.dcm_id = d.dcm_id
and d.dcm_question_group_id = dqg.dcm_question_grp_id (+)
and dqg.question_group_id = qg.question_group_id (+)
);
select study, title into :study, :title from
connection to oracle (select study, title from clinical_studies where clinical_study_id = &csid
);
disconnect from oracle;
quit;

%let study=%trim(&study);
%let title=%trim(&title);

%**put out title page **;
proc template;
define style Styles.mystyle;
pARENT = styles.rtf;
replace Body from Document
"Controls the Body file. " /
bottommargin = .25in
topmargin = 1in
rightmargin = .5in
leftmargin = .5in;
end;
run;
filename rtfout "&deptroot\&saveloc\&study._DB_SPECS.rtf"
options orientation=landscape;
ods listing close;
ods rtf file=rtfout style=mystyle;
title justify=L bold height=14pt "Database Structure Plan";
footnote justify=L height=8pt "&deptroot\&saveloc\&study._DB_SPECS.rtf" ;
data titlepg(keep=col1 col2);
  length col1 $100 col2 $1000;
  col1='Protocol No.:'; col2="&study";
  output;
  col1='Project Name:'; col2="&title";
  output;
  col1='Date of Plan:'; col2=left(put(date(),worddate20.));
  output;
  col1='Database Application:'; col2="Oracle Clinical v&dbvers";
  output;
  col1='Data fields for Entry on Database:';
  col2='See Attached List and Annotated CRF';
  output;
  col1='Data Validation Methodology (double-entry, 100% visual check, other):';
  col2="&dval_meth";
  output;
run;
proc report data=titlepg nowd noheader;
column col1 col2;
define col1 / display STYLE(COLUMN)=[font_size=12pt];
define col2 / display STYLE(COLUMN)=[font_size=12pt font_weight=bold];
run;
%**put out database contents page (dataset list)**;
title1 bold height=10pt justify=L "&study - DATABASE SPECIFICATIONS";
title3 bold height=12pt justify=C 'DATA SETS';
proc report data=extract_views nowd panels=2
  style(column)=[font_size=10pt];
column name description;
define name / display "Dataset/Name";
define description / display "Description";
run;
%**check if at least one extract view defined for the study before proceeding**;
%let dsid = %sysfunc(open(work.extract_views));
%if &dsid %then
  %do;
  %let nobs =%sysfunc(attrn(&dsid,NOBS));
  %let rc = %sysfunc(close(&dsid));
  %end;
%else %goto exit;
%if &nobs=0 %then %do;
  %put Macro exiting, no OC view templates found with clinical study id &csid..;
  %goto exit;
%end;
%**query OC for extract specifications**;
proc sql;
  connect to oracle(path=&db user=&user pass=&pass);
  create table WORK.EXTRACTS as select *
    from connection to oracle (select t1.name, t2.*
      from templates t1, (select to_number(NULL) dcm_id, to_number(NULL) dcm_que_dcm_subset_sn, &tid1 template_id, &ktid1 key_template_id, &name1 view_name, &sname1 sas_ds_name,
tc.display_sequence_no,
tc.name oracle_name, tc.sas_name, tc.sas_label,
'ORACLE VARIABLE' attribute_name, to_char(Null) pages,
to_number(Null) pages_num,
decode(em.data_type_code,'CHAR',em.data_type_code || '(' || em.length || ')',
em.data_type_code) data_type_length,
to_char(null) dvg,
em.sas_format sas_format
from template_columns tc, extract_macros em
where tc.template_id = &ktid1
and tc.key_extract_macro_id = em.extract_macro_id
UNION
select
dq.dcm_id, dq.dcm_que_dcm_subset_sn,
&tid1 template_id, &ktid1 key_template_id, &name view_name,
&name1 sas_ds_name,
100+tc.display_sequence_no display_sequence_no,
tc.name oracle_name, tc.sas_name, tc.sas_label,
tc.attribute_name,
decode(substr(i.name,1,4),'PAGE', to_char(to_number(substr(i.name,5))),
i.name) pages,
to_number(decode(substr(i.name,1,4),'PAGE',substr(i.name,5),9999)) pages_num,
decode(dq.question_data_type_code,
'CHAR',dq.question_data_type_code || '(' || dq.length || ')',
'DATE','CHAR(' || dq.length || ')',
dq.question_data_type_code) data_type_length,
dvg.name dvg,
case when tc.attribute_name='DVG NUMBER'
then substr(dvg.name,1,8)||'.'
when dq.question_data_type_code = 'NUMBER'
then dq.length+least(1,dq.decimal_places) || '.' || dq.decimal_places
when dq.question_data_type_code in ('CHAR','DATE')
then '$'||dq.length||'.'
else to_char(Null)
end as sas_format
from template_columns tc, view_template_questions vtq,
view_question_mappings vqm,
dcm_questions dq, discrete_value_groups dvg, dci_modules dm, dcis i
where tc.template_id = &tid1
and tc.template_question_id = vtq.view_template_question_id
and vtq.view_template_question_id = vqm.parent_question_id
and vqm.question_id = vtq.question_id
and vqm.dcm_question_id = dq.dcm_question_id
and vtq.occurrence_sn = dq.occurrence_sn
and dq.discrete_val_grp_id = dvg.discrete_value_grp_id (+)
and dq.discrete_val_grp_subset_nm = dvg.discrete_val_grp_subset_num (+)
and dq.dcm_id = dm.dcm_id
and dq.dcm_que_dcm_subset_sn = dm.dcm_subset_sn
and dm.dci_id = i.dci_id
%do i=2 %to &nobs;
UNION
select
to_number(Null) dcm_id, to_number(Null) dcm_que_dcm_subset_sn,
&&tid1 template_id, &&ktid1 key_template_id, &&name view_name,
&&name1 sas_ds_name,
tc.display_sequence_no,
tc.name oracle_name, tc.sas_name, tc.sas_label,
'ORACLE VARIABLE' attribute_name, to_char(Null) pages,
to_number(Null) pages_num,
decode(em.data_type_code,'CHAR',em.data_type_code || '(' || em.length || ')',
em.data_type_code) data_type_length,
to_char(null) dvg,
em.sas_format sas_format
from template_columns tc, extract_macros em
where tc.template_id = &&ktid&i
and tc.key_extract_macro_id = em.extract_macro_id
UNION
select
dq.dcm_id, dq.dcm_que_dcm_subset_sn,
&&tid1 template_id, &&ktid1 key_template_id, &&name view_name,
$&sname&i sas_ds_name,
100+tc.display_sequence_no display_sequence_no,
tc.name oracle_name, tc.sas_name, tc.sas_label,
tc.attribute_name,
decode(substr(i.name,1,4),'PAGE',to_char(to_number(substr(i.name,5))),
i.name) pages,
to_number(decode(substr(i.name,1,4),'PAGE',substr(i.name,5),9999))
pages_num,
decode(dq.question_data_type_code,
'CHAR',dq.question_data_type_code || '(' || dq.length || ')',
'DATE','CHAR(' || dq.length || ')',
dq.question_data_type_code) data_type_length,
dvg.name dvg,
case when tc.attribute_name='DVG_NUMBER'
then substr(dvg.name,1,8)||'.'
when dq.question_data_type_code = 'NUMBER'
then dq.length+least(1,dq.decimal_places)||'.'|| dq.decimal_places
when dq.question_data_type_code in ('CHAR','DATE')
then '$'||dq.length||'.'
else to_char(Null)
end as sas_format
from template_columns tc, view_template_questions vtq,
view_question_mappings vqm, dcm_questions dq,
 discrete_value_groups dvg, dci_modules dm, dcis i
where tc.template_id = &tid&i
and tc.template_question_id = vtq.view_template_question_id
and vtq.view_template_question_id = vqm.parent_question_id
and vtq.question_id = vqm.question_id
and vqm.dcm_question_id = dq.dcm_question_id
and vtq.occurrence_sn = dq.occurrence_sn
and dq.discrete_val_grp_id = dvg.discrete_value_grp_id (+)
and dq.discrete_val_grp_subset_nm = dvg.discrete_value_grp_subset_num (+)
and dq.dcm_id = dm.dcm_id
and dq.dcm_que_dcm_subset_sn = dm.dcm_subset_sn
and dm.dci_id = i.dci_id
%end;
) t2
where t1.template_id = t2.template_id
order by t1.template_id, display_sequence_no, pages_num, dcm_que_dcm_subset_sn
);
disconnect from oracle;
quit;

%**apply common attributes as OC does, and collapse page numbers**;
data extract2;
set extracts;
by template_id display_sequence_no pages_num dcm_que_dcm_subset_sn;
length pages_all $1000;
retain pages_all;
select (attribute_name);
when ('FULL_VALUE_TEXT','DVG_LONG_VALUE') do;
data_type_length='CHAR(200)';
sas_format='$200.';
end;
when ('DVG_NUMBER') do;
data_type_length='NUMBER';
sas_format='10.';
end;
otherwise;
end;
if first.display_sequence_no then do;
pages_all=pages;
end;
else if pages ne '' then pages_all=trim(pages_all)||', '||left(trim(pages));
if last.display_sequence_no then output;
drop pages pages_num;
label oracle_name='Variable Name'
sas_name='SAS Name'
sas_label='Label'
attribute_name='OC Data Location'
data_type_length='Field Type/Length'
CONCLUSION

Often in the Data Management process, documentation which is critical to the process is not easily produced and many organizations resort to manually compiled reports. Using SAS to query meta-data from a CDMS is a powerful tool that can alleviate much of the documentation and reporting needs.

Understanding the database schema for Oracle Clinical is critical to this solution. However, once an understanding is gained of how to join tables and pull the required information out of the system then the possibilities expand. For example, SAS has successfully been used as a metrics reporting tool that queries Data Management metrics from Oracle Clinical (page processing, query processing and resolution performance, status of studies, page tracking, etc.) using PROC SQL and ACCESS to Oracle and displays the results through HTML pages on an intranet or via email distribution. Another implementation of this involved producing customized SAS extracts based on the default extract views that would automatically create the SAS formats catalog from the DISCRETE_VALUE_GROUPS used and attaching these SAS format to the appropriate variables in the extracted SAS datasets.

With the right knowledge and tools, many documentation and reporting needs can be met with an automated report such as the example discussed in this paper.

CONTACT INFORMATION

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

Kyle McBride
Instat Consulting Inc.
227 Longwood Avenue
Chatham, New Jersey 07928
Work Phone: 973-997-1352
Email: kyle@instatconsulting.com
Web: http://www.instatconsulting.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.