

**Using SAS/IntrNet® to Generate Data Products from a Database:
The SAO Online USAS Guide**

Kirby Cossey, Olin Davis, and Tom Winn
State Auditor's Office, Austin, Texas

Abstract

This paper describes how SAS® programmers at the Texas State Auditor's Office developed an intranet application to provide auditors and investigators with information derived from a large database containing accounting data for the State of Texas.

The Texas State Auditor's Office – Our Context

Operating within the legislative branch, the State Auditor's Office (SAO) provides information about the operations of state agencies and universities to legislators, agency management, and the citizens of the State. The goal of the SAO is to actively assist government leaders to create and maintain strong accountability systems that ensure efficient, effective operation of state agencies and universities. At the direction of the Legislative Audit Committee, the State Auditor would conduct an audit or investigation of any entity receiving funds from the state.

Unlike many other organizations, the SAO does not "own" the data that are examined using our analytical procedures. The State of Texas has 209 agencies, boards, commissions, schools, and higher-education institutions. Many of these entities have their own internal record-keeping systems, which they maintain on their own computing platforms. Systems analysts at the SAO spend a great amount of time extracting data from agency sources and converting them into something usable by our auditors.

At the SAO, SAS® is used in both the mainframe and Windows environments. The systems analysts that support auditors use SAS on MVS mainframe platforms at the Texas Legislative Council, the State Comptroller's Office, and the Texas Education Agency. On those computers, we mostly use SAS in batch mode via TSO/ISPF. Unfortunately, none of these installations have licensed SAS Integration Technologies, so we are unable to take advantage of the interoperability and information distribution that would be available using that SAS System component. Using DATA step programming, PROC SQL, and other SAS procedures, we extract various types of data from production database tables and extract data files. Often, we obtain data from various agencies on tape cartridges, CD-ROMs, or via FTP, from which we retrieve the information using SAS. The data are extracted, combined, summarized, processed, and written out to flat files, and then they are downloaded via FTP to our PC/LAN environment, where the data files are imported into a data format which will be usable by the auditors (Excel, Access, or ACL). We also use SAS for Windows, locally installed on our PCs, to do many of the same kinds of things that we do on the mainframes, but using data that comes to us in various formats.

Many SAO auditing activities involve financial data which come from the Uniform Statewide Accounting System (USAS), which resides on a mainframe computer at the State Comptroller's Office. USAS is the central accounting system for the State of

Texas. Some of the USAS data files are very large, and are stored on magnetic tape. Data extraction routines running against those files usually are very long-running batch jobs, and these jobs are vulnerable to cancellation by impatient computer operators with different priorities than helping the SAO.

The Decisions – Our Challenge

In July 2002, the SAO made the following decisions:

- (1) Instead of executing a separate query against the statewide accounting system database each time a data request is received, the SAO decided to create its own version of the statewide accounting transactions data every month, containing frequently-requested columns, and to use those data for data requests instead, as needed.
- (2) The SAO also decided to develop an application which auditors could use, without having to solicit assistance from programmers, to create standardized reports or data files containing the financial data they need.

What Was Done – A Chronology of Our Efforts Toward the Solutions

In November 2002, the SAO established a SAS® Application Server on a Windows Terminal Server. An important part of the justification for our SAS Server was the need for our state agency to improve our ability to work with some very large data files including, but not limited to, the USAS financial data.

We decided which data files, and which columns within them, would be downloaded from USAS on the Comptroller's Office mainframe computer. We wanted to include information pertaining to expenditures, revenues, encumbrances, budgets, and vendor payments, so we decided to include data from the detail transactions history, General Ledger, and budget/appropriations files. Starting with September 1999, historical data from the desired files were extracted, downloaded, and stored in SAS data sets on our SAS server. This made it possible for us to use SAS to create data products quickly for various auditor requests, without having to deal with the inconvenience and delays of repeatedly running queries against the production data. However, since auditors couldn't access the SAS data themselves, it still was necessary for them to contact a SAS programmer for help, whenever they had a request.

In August 2003, the SAO began planning the development of a web-based application for distributing the USAS financial data to auditors. There were several meetings with representatives from SAS Institute, and some of those discussions focused on SAS Institute developing a web application Pilot Project at the SAO, using the USAS data. During October 2003 and February 2004, Kirby Cossey, Olin Davis, and Tom Winn took training classes about SAS web tools. On October 14, 2003, the SAS/IntrNet software was installed on our SAS Server, and Olin Davis began experimenting with this software. The conversations about a possible SAS Pilot Project continued for several months without the attainment of a mutually acceptable agreement. Throughout the negotiations, Olin continued working with SAS/IntrNet, and he assembled the various pieces of an intranet application according to the specifications in the original plan. Olin occasionally consulted with Kirby Cossey and Tom Winn about the work that he was

doing. Olin's application creates either printed reports or data files containing comma-separated-values, from the USAS data, according to various user-specified selections. Finally, the discussions between SAS Institute and the SAO regarding a Pilot Project for a USAS web application became moot. The intended system already had been created, and it was developed completely "in-house". During March 2004, the SAO's online "Uniform Statewide Accounting System (USAS) Guide" was moved from the test web server into production, and it was publicized to the auditors.

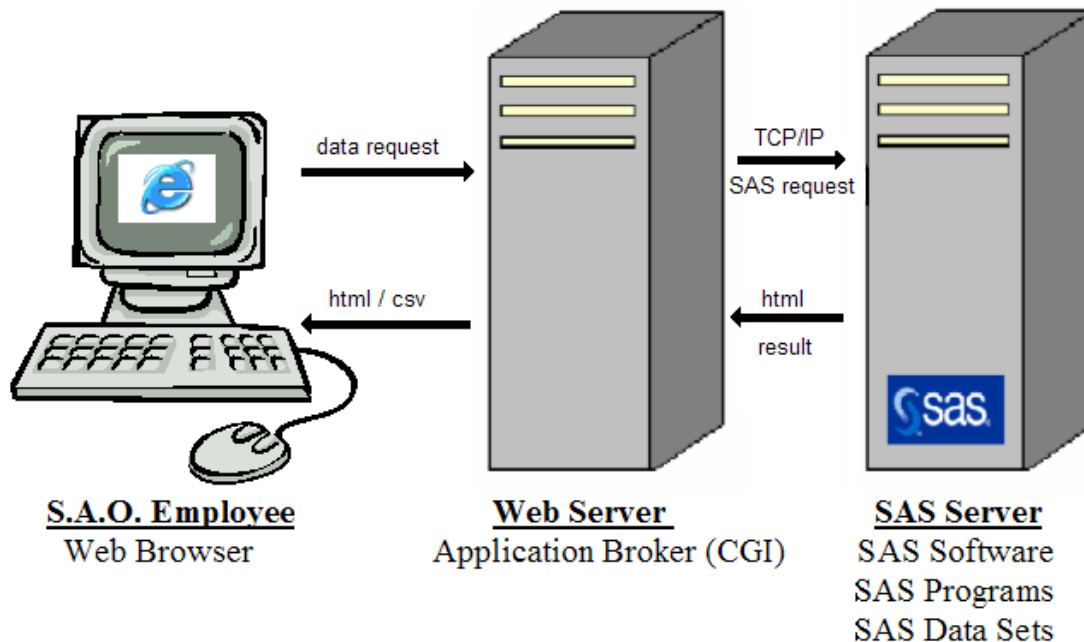
How It Was Done – Some of the Details Concerning Our Solutions

The State Auditor's Office SAS server system consists of a Dell PowerEdge™ 2650 server (with two Pentium® III Xeon™ processors) connected to a Dell PowerVault™ storage system. The storage system consists of one PowerVault 660F 14-bay fiber channel RAID array and one PowerVault 220S 14-bay SCSI RAID array, both in RAID 5 configuration. The server is configured with Microsoft® Windows® 2000 Server. In November 2002, when the SAS Application Server was initially set up, the software installed on our server was limited to Base SAS®, SAS/SHARE®, SAS/STAT®, SAS/GRAPH®, SAS/CONNECT®, and SAS/ACCESS® Interface to PC File Formats. In October 2003, SAS/IntrNet®, and SAS/ACCESS® Interface to Sybase® were added.

The SAO's Web server has the same basic specifications as the SAS Server, except that there is no external storage system attached to it. It is a Dell PowerEdge 2650 server with dual Pentium III Xeon processors. The internal hard drives are configured in RAID 5 configuration. It is running Windows 2000 Server with Internet Information Services.

The USAS detail transaction history files are stored on magnetic tape on the Comptroller's mainframe computer. Annually, this file contains about 40 million records, each of which is 1300 bytes long. SAS programs are run to extract the 60 most frequently-requested columns into SAS data sets, and also to create a few additional variables, which are used to facilitate manipulation and interpretation of the data. PROC CPORT is run to create transport files from the SAS data sets on the mainframe, and then the transport files are sent via the FTP process to the SAS Server. Then, at the SAS Server, PROC CIMPORT is run to translate the transport files into SAS data sets which can be used in the SAS Server environment. PROC DATASETS is used to create four simple indexes (agency number, vendor number, document number, and object number) for the data sets. These indexes will expedite the querying and reporting process later. Various check programs are run, to ensure the validity of the extracted information. Procedural steps similar to those described above are also used for the General Ledger files, and the Appropriations/Budgets summary files. Around the beginning of each month, these steps are executed to capture data pertaining to accounting transactions during the preceding month.

Here is a conceptual diagram which describes the interaction of the operational components that are used by the online USAS Guide at the SAO:



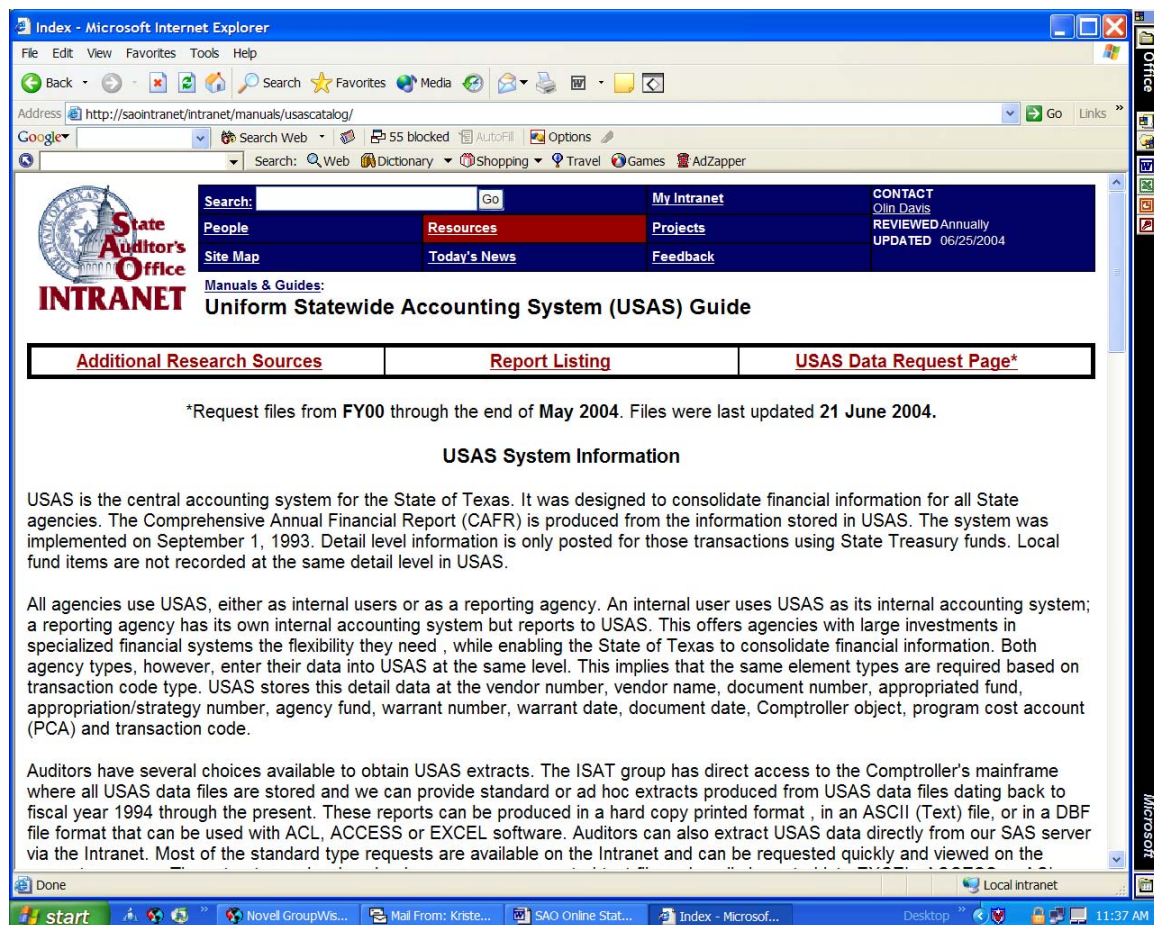
The USAS Guide intranet application is made possible by SAS/IntrNet software, using Compute Services with the Application Dispatcher. Application Dispatcher is composed of two pieces: (1) the Application Broker, which is a Common Gateway Interface (CGI) program that resides on the web server, and communicates between the browser and the SAS Application Server, and (2) the SAS/IntrNet compute server, which is a SAS program on the SAS Server. Since the SAO's has SAS/SHARE installed on the SAS Server, it also would be possible for the SAO to develop SAS/IntrNet applications that make use of Data Services through execution of the htmSQL CGI program on the Web server; however, the USAS Guide does not use htmSQL.

The USAS Guide is available to all SAO employees by following a straightforward path from the intranet home page. In the application, requestors are prompted to make selections from a sequence of linked web pages, and ultimately from one of several HTML forms. The parameters from the HTML form are passed to a SAS program as values of macro variables, and the program executes a query and generates HTML output, which is sent back to the requestor. All of the HTML documents, plus some static web pages containing reference information and examples, are kept in a single folder on the web server. They are retrieved from the web server by referencing their URLs. The SAS programs and data are stored on the SAS Server.

The web pages pertaining to the data requests were constructed using Macromedia Dreamweaver MX software. Like all SAO internet/intranet applications, the USAS Guide conforms to agency standards for web applications. For example, a particular style sheet is used repeatedly by all of the HTML files which generate data requests. It provides a standardized header, as well as some links to certain other SAO web pages. There also are standards which pertain to the coding and formatting of web content. For example, we are prohibited from using frames, and also from using certain colors.

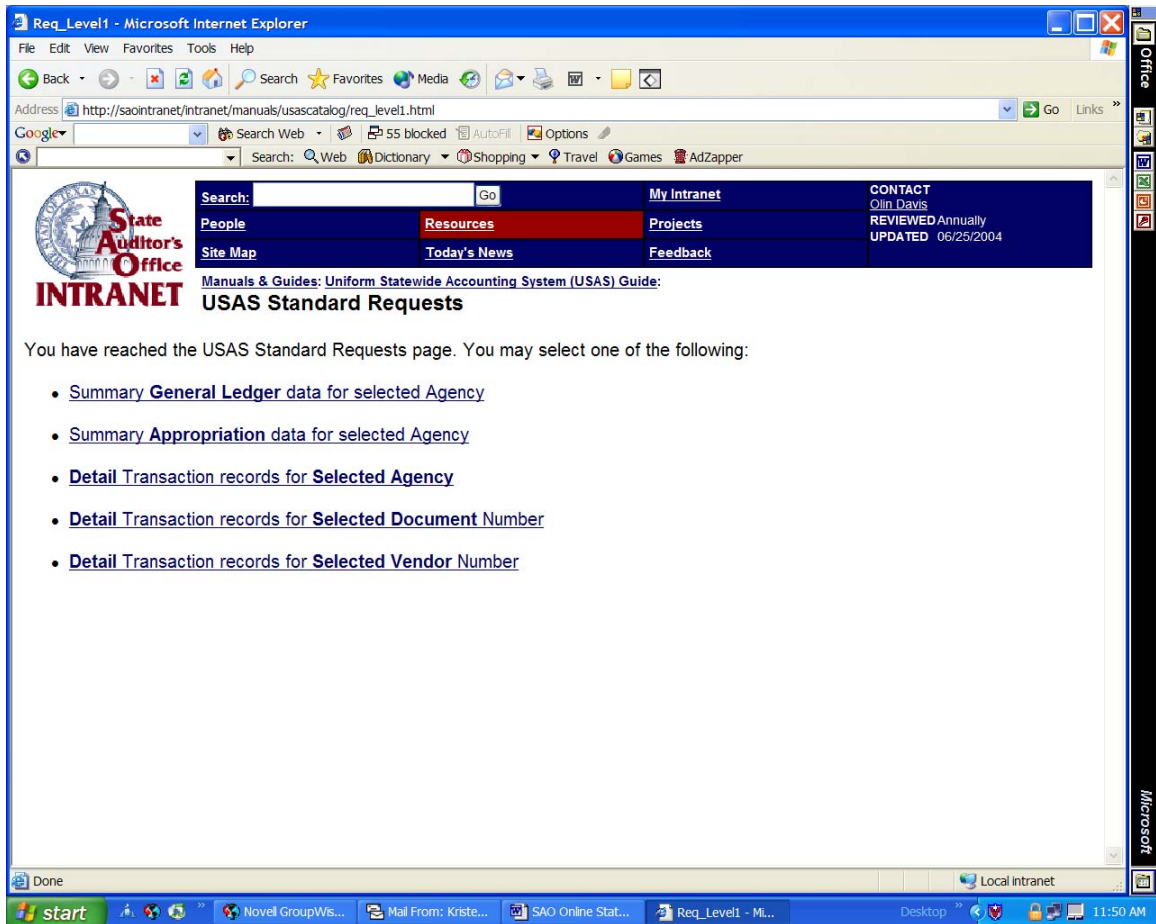
How the USAS Guide Works – The Results From Our Efforts

Here is a portion of the main page for the USAS Guide:

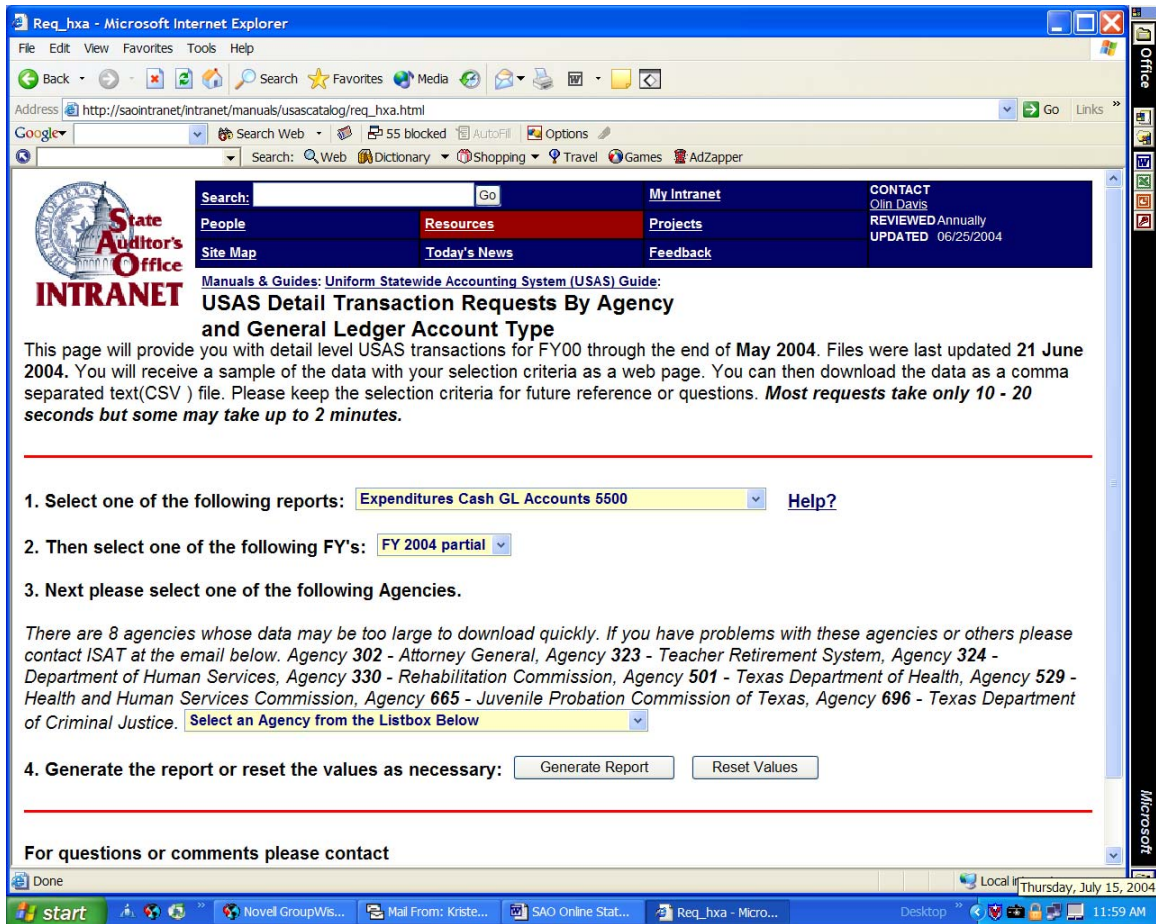


The main page presents the user with some general information about USAS, some links to descriptive information pertaining to the columns, to additional reference materials, to some sample reports, and to the USAS Data Request Page. The sample reports are static web pages that provide the users with examples of typical information for the kinds of data which are available in the USAS Guide. The column headers in the sample reports contain clickable links to their corresponding descriptions in the column definitions table.

By clicking on the link for “USAS Data Request Page”, the user is presented with a listing of the available selection categories for the standard reports.



Let us suppose that the user is interested in certain reports containing transactions details for a specific agency and accounting period. So, if he/she clicked on the link for “Detail Transaction Records for Selected Agency”, then the following HTML form would be returned:



The user would make his/her selections from the drop-down boxes, and then would click on the “Generate Report” button. At this point, the important details for the specific data request would be sent from the Web browser to the SAS Application Broker CGI program, which runs on the Web server. Then, the Application Broker would access the SAS Application Server, and would pass the parameters for the user’s particular selections to the appropriate SAS programs.

In the USAS Guide, all data requests are handled dynamically, which means that the application does not include a collection of pre-prepared reports, covering all possible requests, stored as HTML documents, which would be displayed whenever one of them was requested. The application could have been designed to work that way, but it wasn’t -- because we didn’t want to have to manage an enormous collection of separate, static reports. Instead, each data request results in the execution of SAS programs which extract the data which are pertinent to the request from appropriate SAS data sets, and then create HTML web content using the SAS Output Delivery System (ODS).

If we had decided to build this application using static web pages, then accommodating just five fiscal years of data for the USAS Guide would have required more than 18 million static web pages!

Let us suppose that the user made the following selections for his/her data request: “Expenditures Cash GL Account 5500”, “FY 2004 partial” (when this request was sent on July 15, 2004, Fiscal Year 2004 hadn’t been completed), and “Agency 308”. The next display that the user would see would be the following (partial) web page:

You have requested Detailed USAS HX Expenditure Control Cash GL 5500 data for Agency 308 in FY 04. Here is a sample of up to 15 transactions from the 3,169 transactions in the data set. The total of Amount_WS for this data is \$14,131,252 . The General Ledger total is \$14,131,252 and the difference is \$0 . Any non zero differences can be attributed to several factors including differences in the way HX and GL data are stored and the timing between accruals, encumbrances and cash transactions. If the non zero total is relevant to your project then ISAT can assist with additional data research but contacting the comptroller Appropriations Control Officer or the Agency may still be necessary to determine the reasons. You can download the complete dataset as a comma seperated(CSV) file or go back to the request form and modify your request. Today is 15JUL2004.

Save Selection Criteria as Rich Text File

Download as a CSV File

If you want to include the Title columns such as Agency name or fund name then select this button to download a CSV file. Depending on the number of records it may take an extra 60 seconds to download.

Download as a CSV File with Title Columns

***FY 04 USAS Detail Transaction Data
Expenditure Control Cash GL 5500
For Agency: 308***

Obs	Eff_DATE	TCode	Rev_IND	Apro_NO	PCA	Object_NO	Ven_NO	MailCode	Doc_DATE	Doc_NO
1	03MAY2004	390		13001	00010	7002			30APR2004	YP044FR

Notice the URL in the “Address” line near the top of the Internet Explorer window.

The first SAS program launched on the SAS Server is ad_dispatch.sas. This program identifies which specific SAS program to execute for the query, based upon the parameters for “major”, and “minor”, which identify report families. For this request, “Expenditures Cash GL Account 5500”, the data source was the HX detail history extract file (major=hx), and the user wanted information pertaining to an agency (minor=a), and so ad_dispatch.sas determined that the next program to be launched would be hxaw.sas. The SAS Server includes separate programs for each available combination of major and minor report families. The URL also contains the values of parameters that pertain to the user’s specific selections for this query, “seltyp”, “selfy”, and “agy”. In the request, seltyp=ec identifies “expenditures – cash basis”, selfy=04 identifies “Fiscal Year 2004”, and agy=308 identifies Agency #308, which is the State Auditor’s Office. Those parameters were inserted into the particular SAS program, hxaw.sas, as values of macro

variables, before it was executed. The program generated a standardized report containing data items have been found to be useful for most data requests of the same type. The program used ODS to write the output to “_webout “, which specifies that the results are to be sent directly to the Web browser.

Besides figuring out which program to launch next, ad_dispatch.sas also does some parameter validation, using SAS. In many other web applications, Javascript is commonly used for error-checking at the browser-level, but Olin’s approach illustrates that this can be done on the server side of the application, and without using other web scripting languages.

Observe that there is a message at the top of the report which describes the query, and also which summarizes the results. Following this descriptive information, there is a button which allows the user to download the displayed query description and summary, for documentation purposes. (Capturing this information is very important to our auditors.) Scrolling farther down the page, the user is able to examine 15 rows from the result set. The purpose of this listing is to give the user an idea of what to expect when he/she downloads the entire result set as a CSV file. If a result set happened to contain 15 rows or fewer, then the application would display all of the rows.

The screenshot shows a web browser window titled "SAS Output - Microsoft Internet Explorer". The address bar contains the URL: `http://saointranet/sas/sas-cgi/broker.exe?_service=SAO&format=W&_program=saoprogram.ad_dispatch.sas&major=hx&minor=a&seltyp=ec&selfy=04&agy=308`. The browser displays a table with 15 rows of data. The table has columns for row number, date, and various numerical and alphanumeric values. A status bar at the bottom indicates the request took 2.01 seconds of real time.

1	03MAY2004	390		13001	00010	7002			30APR2004	YP044FR
2	03MAY2004	390		13001	00010	7003			30APR2004	YP044FR
3	03MAY2004	390		13001	00010	7022			30APR2004	YP044FR
4	03MAY2004	390		13001	00010	7050			30APR2004	YP044FR
5	03MAY2004	390		90327	90327	7032			30APR2004	YP044FR
6	03MAY2004	390		99327	99327	7041			30APR2004	YP044FR
7	03MAY2004	390		13001	00010	7002			30APR2004	YP044FS
8	03MAY2004	390		13001	00010	7022			30APR2004	YP044FS
9	03MAY2004	390		13001	00010	7050			30APR2004	YP044FS
10	03MAY2004	390		90327	90327	7032			30APR2004	YP044FS
11	03MAY2004	390		99327	99327	7041			30APR2004	YP044FS
12	03MAY2004	368		00632	34308	7996			30APR2004	YLR0412
13	04MAY2004	390		13001	00010	7002			30APR2004	YP044FR
14	04MAY2004	390		13001	00010	7003			30APR2004	YP044FR
15	04MAY2004	390		13001	00010	7022			30APR2004	YP044FR

This request took 2.01 seconds of real time (v8.2 build 1391).

You might be wondering how the application handles both the 15-row data example and the complete result set, since CGI is essentially a batch-type process. In the SAS program, there is a line which specifies “%let rc=%sysfunc(appsrv_session(create));”. So, the program starts a temporary session, and creates a SAS data set that meets the selection criteria. This session remains active (for a specified time) while the program returns an HTML page that provides some information about the data set and an example of a few observations. The user can evaluate the results and determine if the output will be suitable, or if different criteria should be used. Selection buttons are provided on the HTML page that permits the data set and/or the selection criteria to be downloaded. Clicking on any of the three download buttons would result in the execution of other SAS programs, which would generate the appropriate CSV or RTF files for the application’s outputs. Clicking these buttons also will return the user to the active session using the session id. In addition, whenever a user makes a download request, a separate file of the selection criteria is sent to another directory, for tracking purposes.

During development and testing of the application, a very useful technique for debugging SAS programs invoked through the Application Dispatcher was to append the debug flag and associated parameter value “&_debug=131” to the URL. This returns all values passed to the SAS Server, the SAS Log, and the total elapsed time.

Despite our efforts to maximize the efficiency of processing data extracted from our SAS data library, we experienced various kinds of difficulties involving timeouts, whenever we needed to process a particularly large data request – either the program wouldn’t run long enough, or it took too long to download, or the temporary session wouldn’t stay open long enough. Solution of these problems involved resetting several default timeout settings. A helpful reference was SN-005892, available from SAS Technical Support (see **Suggestions for Further Reading**).

If the user doesn't know what the column-headings in the reports represent, there also is a useful alphabetical listing of the column-headings, together with their associated definitions.

The screenshot shows a Microsoft Internet Explorer window displaying the State Auditor's Office Intranet. The page title is "Definitions for Column Headers in USAS Data Requests, Alphabetical Listing". The page contains a table with two columns: the column header and its definition.

Column Header	Definition
Agency_No	Displays the three-digit agency number established in the D02 Agency Profile of the agency that received financial impact.
AGY_CODE1_FLAG	The D26 Agency Code 1 Profile is used for any coding purposes. This information is used for an agency's internal use only.
AGY_CODE2_FLAG	The D27 Agency Code 2 Profile is used for any coding purposes. This information is used for an agency's internal use only.
AGY_CODE3_FLAG	The D36 Agency Code 3 Profile may be used for any coding purposes
AGY_CON_FLAG	The 25 Agency Control Profile allows each agency to perform selected actions independently of other agencies. This profile must be established before an agency can process transactions in USAS. The 25 Profile turns the cost allocation cycle on and off and also defines the agency allocation indicators used to govern the cost allocation cycle.
AGY_GLACT_FLAG	The D32 Agency General Ledger Account Profile contains the Agency General Ledger, which can be a more detailed account than the Comptroller General Ledger Account.
AGY_OBJ_FLAG	The D11 Agency Object Profile contains Agency Object Codes, which are established at the same level of detail, at a more detailed level, or at a higher level with less detail than the Comptroller Object Code. If established at a higher level, both the Agency and Comptroller Object Codes are required when entering a transaction.
AGY_OBJGP_FLAG	The D25 Agency Object Group Profile establishes the Agency Object Group and the associated title for agency budgeting and reporting purposes
Amount_WS	The actual financial dollar amount of a transaction with a sign determined by the normal General Ledger Account credit or debit indicator.

Miscellaneous Remarks.

The USAS Guide application includes many selections besides the illustrations described in this paper, but the operational flow for them is similar to the examples described.

The USAS Guide does not include data analysis capabilities, and no customizable details pertaining to the data products themselves. Providing those functionalities would have required a more complicated application than the one that was developed by the SAO. However, our auditors prefer performing their own data analysis using other software tools (currently Microsoft Excel, Microsoft Access, and ACL). Therefore, the USAS Guide seems to meet their needs very well. It is easy to use, and it includes most of the financial information that auditors need (vendor payments, cash expenditures, revenues, encumbrances, appropriations, and budget transfers). Of course, not all of our audits involve financial data from the statewide accounting system, but auditors have reported to us that they have found the application to be very useful for those audits which do.

Developing the USAS Guide has provided us with some important experience, and also has provided us with a template for future development of some other web-based applications. The USAS Guide is a “work in progress”, and we anticipate that making improvements to it will continue to be an ongoing task.

Conclusion

- SAS/IntrNet software was used to create an intranet application for auditors and investigators (non-SAS programmers) to use for creating standardized data products from a large database of financial information.
- In this paper, we have provided descriptions of how the application was developed, and also how it works.

Suggestions for Further Reading

- Teresia Arthur & Mary Jafri, “Web Enable Your SAS Applications,”
 - Proceedings of the 28th Annual SAS Users Group International Conference (2003), Paper 35-28, and
 - Proceedings of SCSUG 2003, pp. 19-25 (sponsored by South-Central SAS Users’ Group, on October 26-28, 2003).
- Keith Cranford & Dan Hammarstrom, “Tricks and Tips with SAS/IntrNet,”
 - Proceedings of the Texas Conference for Government SAS Users (sponsored by the Texas State Auditor’s Office and SAS Institute Inc., on July 29, 2003), pp.57-70, and
 - Proceedings of SCSUG 2003, pp. 70-83 (sponsored by South-Central SAS Users’ Group, on October 26-28, 2003).
- Kevin Davidson, “Using SAS/IntrNet Software,” SSU 2001 Proceedings, pp. 204-210 (sponsored by the SouthEast SAS Users Group, and the South-Central SAS Users’ Group, on August 19-22, 2001)
- Matthew Grover, “SAS Solutions for the Web: Static and Dynamic Alternatives,” Proceedings of SCSUG 2003, pp. 167-178 (sponsored by South-Central SAS Users’ Group, on October 26-28, 2003).
- Lauren Haworth, “HTML for the SAS Programmer,”
 - Proceedings of the 26th Annual SAS Users Group International Conference (2000), pp. 235-255, and
 - SSU 2001 Proceedings, pp. 193-201 (sponsored by the SouthEast SAS Users Group, and the South-Central SAS Users’ Group, on August 19-22, 2001).
- Lauren E. Haworth, “HTML Output,” Chapter 3 in Output Delivery System: The Basics, Cary, NC: SAS Institute Inc., 2001.

- Doyle McDonald & Chas Webb, “SAS/IntrNet Configuration: Deploying a Web-Enabled SAS Environment,”
 - Proceedings of the New Mexico SAS Users Conference, pp. 80-138 (sponsored by South-Central SAS Users’ Group, on August 5, 2002), and
 - Proceedings of the Louisiana SAS Users Conference, pp. 77-140 (sponsored by South-Central SAS Users’ Group, on June 16, 2003).
- Frederick E Pratter, Web Development with SAS by Example, Cary, NC: SAS Institute Inc., 2003.
- SN-005892, “Timeout settings to consider when using the IntrNet Application Dispatcher,” available from SAS Technical Support web site, <http://support.sas.com/techsup/unotes/SN/005/005892.html>

Author Information

Kirby Cossey
Senior Systems Analyst
Information Systems Team
Texas State Auditor’s Office
P.O. Box 12067
Austin, TX 78711-2067

512 / 936-9739
kcossey@sao.state.tx.us

Olin Davis
Information Systems Audit Analyst
Information Systems Audit Team
Texas State Auditor’s Office
P.O. Box 12067
Austin, TX 78711-2067

512 / 936-9660
odavis@sao.state.tx.us

Tom Winn
Senior Systems Analyst
Information Systems Team
Texas State Auditor’s Office
P.O. Box 12067
Austin, TX 78711-2067

512 / 936-9735
twinn@sao.state.tx.us