Paper 300-2007

Case Study in Synchronizing Identities in the SAS®9 Metadata Server with an Enterprise Security Provider

Forrest Boozer, SAS Institute Inc. and Christoper Zogby, Zogby Enterprises, Inc.

ABSTRACT

The SAS®9 Business Intelligence (BI) Server architecture allows a level of SAS® resource administration that was not possible in prior software releases. Access rights to SAS resources can be granted or revoked at the group or individual user level via the SAS® Metadata Server. To leverage these administrative features, user identities must be stored in the metadata repository. Hand entering and maintaining a site's user identities is a tedious, error-prone exercise. This case study highlights the advantage of importing user and group information from an enterprise security provider, such as the Microsoft Active Directory. SAS provides macros that can be integrated with scheduling and other tools to synchronize SAS metadata repository identities with the enterprise Lightweight Directory Access Protocol (LDAP) environment.

INTRODUCTION

The SAS®9 BI Server software provides distinct advantages to both SAS developers and administrators in enterprise environments. Its centralized metadata architecture allows application developers to distribute a single, authoritative source of business intelligence to a vast array of users within their enterprise. This is a powerful feature of the SAS BI Server architecture. To use this power effectively, administrators need to define access rights to metadata objects at the individual user and group levels. The SAS Metadata Server allows this type of administrative control, but only for groups and user identities that are stored within a metadata repository. User identities and groups can be entered manually and maintained in a metadata repository via the User Manager plug-in for the SAS® Management Console. However, such manual entry is impractical at sites with a large, dynamic, SAS user population. For these sites, a more attractive alternative is a maintenance solution that ties metadata repository identities to their state in the enterprise security provider. This type of solution not only leverages identity maintenance work that is performed by the site's system administrators, but also lends itself very well to process automation. The conditions at our study site justified the need for such an identity maintenance application.

Our site employs approximately 800 staff members, and more than 175 analysts who depend on SAS software for their analytic and reporting needs. Analysts are either research staff or human resources specialists, exclusively. Both of these groups make frequent use of non-permanent staff, such as interns and contractors, which contributes to the dynamic nature of our SAS user population. To address the diverse needs of this user community, we created a business intelligence solution using SAS® Enterprise BI Server software. Project-specific metadata such as information maps, stored processes, and library and data set definitions are stored in a metadata repository that is accessible from a centralized SAS metadata server. SAS® Enterprise Guide® 4.1 and the SAS® Add-In for Microsoft Office client tools are used regularly in conjunction with the metadata server to produce intelligence-driven results. The site's willingness to adopt this SAS application architecture depended on our ability to demonstrate that data confidentiality rules would be enforced. We needed to prove that we could prevent research analysts from accessing human resources metadata objects. To prevent unauthorized access to confidential metadata objects in an efficient manner we needed to do the following:

1. Load SAS user identities into the metadata repository.
2. Assign identities to groups based on their job function (for example, human resources or research duties).
3. Maintain the identity metadata to reflect the current state of the site's SAS user community.

Fortunately, we discovered that the site's enterprise security provider, Microsoft Active Directory, contained the data needed to produce our metadata repository identity management solution.

The remainder of this paper discusses the following:

- Major features of the application that synchronizes SAS metadata repository identities and group data to their current state in Active Directory.
- How Active Directory data is extracted, transformed, and initially loaded into a metadata repository are provided.

- How to use bulk load macros that SAS supplies
- How to use scheduling software to automate metadata repository identity data synchronization events.

INITIAL METADATA REPOSITORY IDENTITY DATA LOAD

ACTIVE DIRECTORY IDENTITY EXTRACTION

The first step in the process involved extracting the site's SAS user identities and their associated groups from the Active Directory. SAS® Integration Technologies, which is bundled with the SAS Enterprise BI Server, provided an interface to LDAP data sources via a series of data step, LDAP call routines. The code used in this part of the application was modeled after the SAS sample program, IMPORTAD.SAS. Adaptation of the sample LDAP extraction code to conditions at the study site required relatively minor modifications.

Once a connection was established with our Active Directory server, two separate data step queries were executed to extract user identities and Active Directory group names. The results from these two queries were combined on LDAP common name values. The results were filtered on an Active Directory group name to restrict our identity list to only those people who were SAS software users. The elements stored in the resulting data set were user first name, user last name, e-mail address, organization name, identity, and Active Directory group name.

LOAD MASTER CANONICAL TABLES

SAS recommends that you store the extracted identity data in specially formatted, normalized data sets called *master canonical tables*. *Canonical tables* define the standard attributes and associations for identity metadata objects. Because we are populating the SAS metadata repository with externally-supplied user identity data, we want to ensure that we pass this data in the format that is required by the metadata repository. Also, the autocall macros supplied by SAS that are used to load and maintain metadata repository identities were written specifically to process data stored in this special format. We used the SAS autocall macro, %MDUIMPC, to create empty versions of the eight available canonical tables. To enforce the canonical table's logical data model, primary and foreign key constraints were applied using SQL procedure code.

Canonical tables contained elements that were not available in our Active Directory schema. To load all canonical table elements into the metadata repository, we needed to supplement our extracted Active Directory data with personnel items from our site's Oracle HR database. We used SAS/ACCESS® to Oracle with SQL procedure code to load combined Oracle and Active Directory elements into our relational, master canonical tables. Data that violated the model's integrity constraints was automatically rejected during insert operations performed during the canonical table data load step.

POPULATE METADATA REPOSITORY WITH CANONICAL TABLE DATA

The application used the autocall macro, %MDUIMPL, to load canonical table data into our target metadata repository. This specific data load was performed only once, to populate, initially, our metadata server with external identity data. During this step, %MDUIMPL converts the SAS® canonical data sets into representative XML data files, which are imported into the repository by the METADATA procedure.

At this stage, our metadata repository contained the identity and group metadata necessary to apply security controls to enforce the site's data confidentiality requirements. To ensure that these controls remained in force when our SAS user population changed, our solution also needed to update identities in the metadata repository to reflect their current state in Active Directory. The following section describes how we achieved this end.

IDENTITY SYNCHRONIZATION WITH BULK LOAD MACROS

Metadata repository identity data was kept current by adding bulk load macros supplied by SAS to our identity management solution. We continued to created current master canonical table data using the process described earlier. However, we no longer loaded these values directly into our target metadata data repository. Instead, the current canonical table data were compared with target metadata repository identities to determine which operations were required to keep metadata repository identity data current. This process can be executed repeatedly, but only after the target metadata repository has had its initial external identity data load performed.

The following list contains the steps used to synchronize our repository identities with Active Directory. Steps 2 through 5 used bulk load macros that SAS supplies to perform their described function. Each step that is dependent on bulk load macros is described below.

1. Rebuild master canonical tables.
2. Create target canonical tables from the metadata repository.

3. Compare master and target canonical tables and generate update operations.
4. Validate update operations stored in the change tables.
5. Update target metadata repository with change information.

CREATE TARGET CANONICAL TABLES FROM THE METADATA REPOSITORY

In this step, the identity metadata for the target metadata repository was extracted into a separate set of canonical tables. The SAS bulk load macro, %MDUEXTR, performed the repository identity data extraction. The macro's LIBREF= parameter was set to the desired output library reference to make sure that the resulting tables were stored in their own SAS library. The SAS system options METASERVER= and METAREPOSITORY= defined the target metadata server's name and repository. The specified options were defined prior to calling the %MDUEXTR macro.

COMPARE MASTER AND TARGET CANONICAL TABLES AND GENERATE UPDATE OPERATIONS

The SAS bulk load macro, %MDUCMP, was used to compare master and extracted canonical table data to determine if discrepancies existed. The comparison generated output data sets that contained information on any outstanding insert, update, and delete operations that are required to keep the target metadata repository's identity data current. The macro's CHANGE parameter was set to the desired library reference to ensure that these output data sets were written to their own library. For this application, we were only interested in synchronizing metadata repository identities that originated from this bulk load process. To avoid comparison of manually entered identities, we left the macro's EXTERNALONLY parameter set to its default value of 1.

VALIDATE UPDATE OPERATIONS STORED IN THE CHANGE TABLES

The contents of the change data sets were evaluated for potential integrity constraint violations by the SAS bulk load macro, %MDUCHGV. This macro requires library references for the extracted target canonical tables and change data sets. If the validation step determines that the target repository's integrity constraints will be violated, an errors data set that contains details on the constraint violations is created. The name and location of the errors data set can be specified by setting the macro's ERRORSDS parameter to the desired one- or two-level data set name.

UPDATE TARGET METADATA REPOSITORY WITH CHANGE INFORMATION

If the change data sets were successfully validated, the target metadata repository's identity data is updated by the SAS bulk load macro, %MDUCHGL. The change data location is defined by setting the CHANGE parameter in the macro to the change data's library reference. The target metadata repository is defined by the SAS system options METASERVER= and METAREPOSITORY=.

The addition of the SAS bulk load macros enabled us to modularize the identity synchronization process. Once we were confident that each step produced accurate results, we divided the application code into five separate SAS programs, each of which executed their respectively numbered step as described earlier. The five programs were loaded into a Platform LSF job flow that was scheduled to run just after midnight on business days only. After each job in the flow completed without errors, the next job was started. The exception to this rule was the fourth job in the flow, which also depended on the absence of a problems data set. The jobs were not resource-intensive, and generally, completed within two minutes of initiation. The status of the nightly identity synchronization jobs was broadcast to SAS administrators via customized e-mail messages.

CONCLUSIONS

The biggest obstacle to producing an effective identity synchronization application was to correctly modify the sample LDAP call routines to allow accurate extraction of the required identity data from our site's Active Directory installation. The IMPORTAD.SAS sample program, in its original state, provided most of the LDAP extraction code required. However, prior to making effective modifications, we needed to study the attributes of our site's Active Directory installation with a GUI-based LDAP browser. Once we had a visual representation of our Active Directory data, modifications become much more intuitive.

Once stabilized, the resulting application provided current, accurate metadata identity data to the site's SAS administrators. Within a short time, it became clear that our application was tracking accurately all changes to our SAS user population directly in metadata repository. This boosted confidence that defined metadata server security controls would remain in effect without a great deal of administrative overhead. Ultimately, the ability to maintain metadata repository identities automatically paved the way for the study site's adoption of a metadata-centric, SAS business intelligence application architecture.

RESOURCES

SAS Institute Inc. 2006. "Appendix 2. Bulk Load Processes for Identity Management". *SAS® 9.1.3 Intelligence Platform: Security Administration Guide, Second Edition*. Cary, NC: SAS Institute Inc. Available support.sas.com/documentation/configuration/bisecag.pdf.

ACKNOWLEDGMENTS

I would like to extend a special “thank you” to Bubba Talley of SAS Technical Support for his patience, encouragement, sense of humor, and superior knowledge of SAS Enterprise BI Server software. In addition, I am grateful to Diane Hatcher at SAS Institute for her interest in this topic. Finally, I would like to thank my co-author, Forrest Boozer, also at SAS Institute, for his thoughtful contributions and ideas on this topic and paper.

CONTACT INFORMATION

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

Chris Zogby
Zogby Enterprises, Inc.
P.O Box 11245
Alexandria, VA 22312-9998
E-mail: info@zogbyenterprises.com
Web: www.zogbyenterprises.com