Paper 885-2017

Implementing Role Based Access Control / DSoD authorization schema on SAS® Platform

Perttu Muurimäki, Statistics Finland; Heikki Rouhuvirta, Statistics Finland (retired)

ABSTRACT

Traditionally role based access control is implemented as group memberships. Access to SAS® datasets or metadata libraries requires membership in the group that "owns" the resources. From the point of view of a SAS process these authorizations are additive.

If a user is member in two distinct groups, user's SAS processes have access to the data resources of both groups simultaneously. This happens every time the user runs a SAS process - even when the code in question is meant to be used with only one group's resources. As a consequence, having master data source defining dataflows between groups becomes futile as any SAS process of the user can bypass said definitions. Additionally, as it is not possible to reduce user's authorizations to match those of only the relevant group, it becomes challenging to determine whether other members of the group have sufficient authorization. Furthermore, it becomes difficult to audit statistics production as it can not be automatically determined which of the groups owns a certain log file.

All these problems can be avoided by using role based access control with dynamic separation of duties (RBAC DSoD). In DSoD user is able to activate only one group membership at a time. However, SAS does not support this: metadata identities use one-to-one mapping to host/AD/LDAP -accounts. It is not possible to select authorizing identity after authentication. This paper describes one way to implement RBAC/DSoD -schema in UNIX -server environment.

INTRODUCTION

Statistics Finland produces annually nearly 200 distinct statistics in around 800 publications. Statistics production employes approximately 400 people running around 500.000 lines of SAS code developed in-house. Every year over 500 GB raw data on Finnish population, households and enterprises are provided from different official registeries. Additionally, thousands of individuals, households and enterprises are interviewed in depth. The amount of observation data grows relentlessly year after year.

At the same time the statisticians and researches want to make good use of all this data. Datasets grow in size as new observations are added, new variables created and timeseries grow in length. Simultaneously complexity increases as different datasets are combined in new and interesting ways. These demands place considerable stress on the underlying technological platform as well as on the people that keep it operating.

Statistics Finland has used SAS for statistics production for several decades both on servers and on workstations. In 2011 started work on a new incarnation of SAS server environment (named TKSAS). At that time top level directory structure on the old SAS server was based on organizational units of 2003. The directory layout had no relation to how the production of statistics was conducted at the time. Subdirectories had no standard structure – or structure of any kind. Several terabytes of data was stored in around 170.000 files in hundreds of subdirectories of different depths. Situation in databases was similar. The SAS programs and smaller datasets were placed on network fileshares and bigger datasets on SAS server. SAS RSUBMIT procedure was used to submit code from workstations to server.

In 2011 it was decided that something more than just a new SAS version was needed. A master data model was developed to conceptualize statistics production. Then the model was operationalized around SAS 9.3 and SAS metadata in particular. In this paper the master data model is presented followed by a detailed description of the implementation. Finally, real world experiences on using the new TKSAS platform and some conclusions are provided.

DESIGN PHASE

PROBLEMS THAT NEEDED TO BE ADDRESSED

In TKSAS design phase in 2011-2012 several problems and shortcomings were identified:

- The proliferation of datasets, programs, applications and their respective permissions would need to end. Previous decade had shown that keeping record of each and every resource and it's supposed and actual permission set was an impossible task.
- Common namespace for resources was needed. Granting permissions on files, directories, programs and databases is impossible if permissions are asked to unknown logical or application spesific resources.
- Programs would need to be managed. Almost all programs deal (and work) only with particular datasets and conditions. Yet the programs were stored separately from data. There was no way of knowing
    1. whether any particular dataset was used somewhere.
    2. whether modifying or removing a particular data set would break a program somewhere.

Also several desired properties were identified:

- The knowledge on what permissions are granted and why, would have to be stored *outside* of any particular server or database. Otherwise there would be no way of knowing whether a particular authorization on a server or database was valid or not.
- All employees assigned to work on an a statistic or a datacollection in a distinct role should be equal. All of them should be able to run same the programs and access same the resources with identical results.
- There should be no need to apply permission for an employee individually. If an employee is assinged to work on a certain statistics, it is known that the employee needs permissions also to a certain set of data collections. Those additional permissions should be granted automatically.
- Data owners should have (near) real time view of granted roles and permissions. They also should have the ability to administer them without the aid of IT personnel.
- Statistics, data collections and other resource groups should have proper lifecycle management. Projects come and go, new data collections are launched and old retired. Occasionally production of a statistic comes to an end as information needs develop. The relations between objects should be known as there may be interdependencies. While groups, directories, databases and tables can not exist indefinetily, they can not just vanish either.
- System should provide a basic workflow model for all statistical work in Statistics Finland.

CONCEPT OF GENERIC STATISTICAL UNIT

Producing statistics involves data, programs and people. Same data can be used in several statistics, same programs can be used in producing different statistics, same people can take part in production of several statistics in different roles. Managing statistics production has to be able to cope with all those factors.

A concept of Generic Statistical Unit (GSU) was developed as a way to describe statistical production work at a general level. Real world objects such as code, datasets, persons and processes would all be subordinated to them.

Several properties are attached to GSUs.

1. Each GSU has a purpose: there is an information requirement that needs to be addressed. From this purpose a name and a short description can be readily generated. Only after creating this logical container object, can any tangible resources be meaningfully grouped together.

2. In production of a statistics, there is a known set of generic phases the data go through: data collection, data preparation, data analysis and publication of results. In more generic terms: GSU input, GSU processing and GSU output.
3. Programs manipulating data have different levels of maturity: usually development, testing and productions levels are distinguished.

Finally, human resources are attached to GSUs to do the actual work.

While GSUs are independent and self-contained, they would need to communicate with each other, as two or more GSUs can be chained to produce a published statistical end product. Hence, relations between GSUs need to be recorded also.

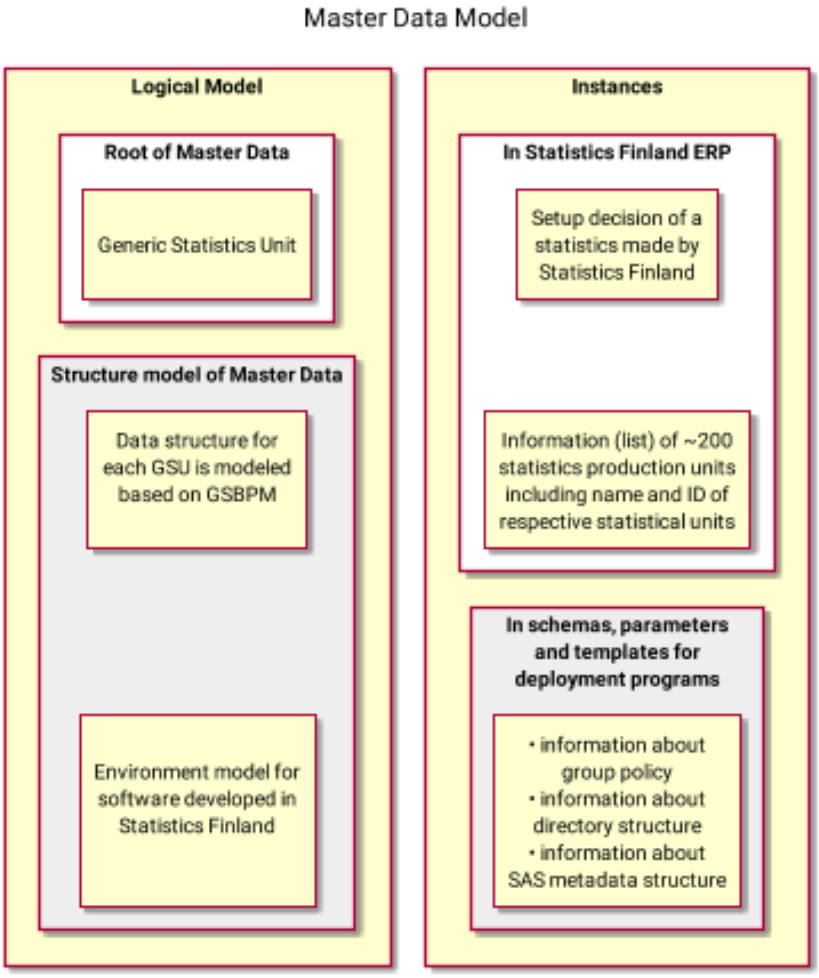

Figure 1: Master Data Model in Statistics Finland

MODELLING MASTER DATA

Accepting GSU as a starting point Master Data Model was conceptually formalized (see figure 1).

The developed Master Data System should comply with following demands:

- An authorization management **system** was needed. Authorizations made on paper were insufficient.

- An authorization management system should form a authorization master data – the only valid source of authorization information.
- Authorization master data should be master data for real. When master data and a deployment platform disagree master data must override.
- Authorization master data should be separate from any particular deployment platform. It should be used only to contain authorization metadata.
- Master data should create one shared namespace for all different deployment platforms.
- Master data should be deployed automatically and instantly on deployment platforms. It should also be continuously enforced to prevent discrepancies. On deployment platforms all permissions to resources (files, directories, databases, tables, …) should be granted to groups – not individual employees.
- Relations between GSUs in master data should be managed via GSUs themselves, not their respective members. For example, when a statistics needs data from a data collection, then the statistics is given authorization to access the needed data. The access is granted to the GSU itself – not to it's members.

NEED FOR DYNAMIC SEPARATION OF DUTIES

Many of the design goals would be achievable if regular RBAC was applied rigorously. However, standard solutions have certain properties that undermine the long-term viability of a regular RBAC system.

1. After a while very few employees have the same set of permissions (group memberships). If the permissions of employees are additive, then the SAS processes of an employee are run with employees full permission set everytime. It becomes non-trivial to make sure that new program works for everyone who has been given access to it in master data as program developers and users may have different active permission sets.
2. Relationships between objects (statistics, data collections) are managed at object level. These relationships are implemented as read and/or write permissions in standard directories. If the permissions of employee are additive employee can disregard both existing and non-existing relationships in metadata
    - by writing code that would run succesfully only by the employee or someone with same (sub)set of permissions.
    - by creating new datasets that combine data from different objects in ways not defined in metadata.
    - by placing datasets in locations not defined in metadata.

In long-term use these effects would accumulate and result in unpredictability in deployment of authorizations and difficulties in administering the system as a whole.

IMPLEMENTATION

HOW DOES IT WORK?

Starting point

There are two separate tasks: first 1) creation of GSU and then 2) managing access to it.

In the beginning, a decision to produce a new statistics is made. It is given a name and an ID and a corresponding object is created to ERP. From ERP the name and ID are copied to authorization master data system, and required elements are created. First level of Master Data is organised as a tree structure and a technical solution is LDAP.

Access control happens by adding and removing members to role groups in GSU's authorization master data. Members are Person -objects or other GSU role groups.

Changes made to authorization master data are propagated automatically to deployment platforms.

Automatic phases

All master data is deployed algorithmically. Creating new GSU branches to LDAP -tree launches processes on SAS servers that create respective OS and metadata groups and OS and metadata directory/folder -structures. Adding new members to a groups in LDAP launches processes on SAS servers that create new OS logins and metadata identities and add them to right groups. Users are sent their new logins via email. Deleting members from group in LDAP makes them disappear from SAS servers during the off-hours.

Additionally, filesystem permissions are enforced when new files or directories are created (NFSv4 property). They are also enforced nightly to the whole SASDATA filesystem hierarchy.

Metadata folder permissions are not enforced. Permissions are inherited from the ACTs of the group's root folder and there should be no need to modify individual object permissions. Automatic enforcing is always possible, however, and will be used if need arises.

DETAILED DESCRIPTIONS

Master data

There was no satisfactory data sources enumerating GSUs readily available. ERP was the most accurate but its use is not generally accepted as "being a good idea" as its purpose lies in accounting. All the same, in the end ERP was chosen as the source and it has worked adequetly.

The actual master data was placed in LDAP -tree on directory server. LDAP was chosen as the implementation technology for master data repository as it had several benefits:

4. it (protocol) is standard technology universally available on different platforms
5. it (protocol) was already available and in use in Statistics Finland
6. it (server) was universally visible within internal network of Statistics Finland
7. master data is modelled as a tree.

Dedicated branch was obtained for storing GSU structures. Employees were already present elsewhere in the tree as Person -objects. Figure Kuva Kuva presents the general structure of an GSU..

Basic data model is simple:

- GSUs are represented as organizationalUnit (ou) -objects.
- Roles (permission sets) are represented as Group -objects.
- Employees and employee groups are represented as Person -objects or Group -objects.

Granting an employee a permission to work on a statistic is reduced to adding the employee to the right group. Granting a statistic permission to read from an Export -directory of a data collection is reduced to adding statistics group to Export group of the data collection.

SAS server environment specific master data is organized as a subtree in LDAP directory. All object names and codes are the same as in ERP. If a code is missing from ERP it will not be added to TKSAS. At this stage this is only a policy as no technical barriers have been implemented.

At the root level there are branches for development, test and production environments. Under each brach are sub braches for each GSU type (statistic, data collection, project, …). Under them are branches for each individual GSU. There is a one-to-one mapping between GSU and an ou -object in LDAP. Under each such ou are the groups that roles are mapped to. All ou -objects have same set of role groups.

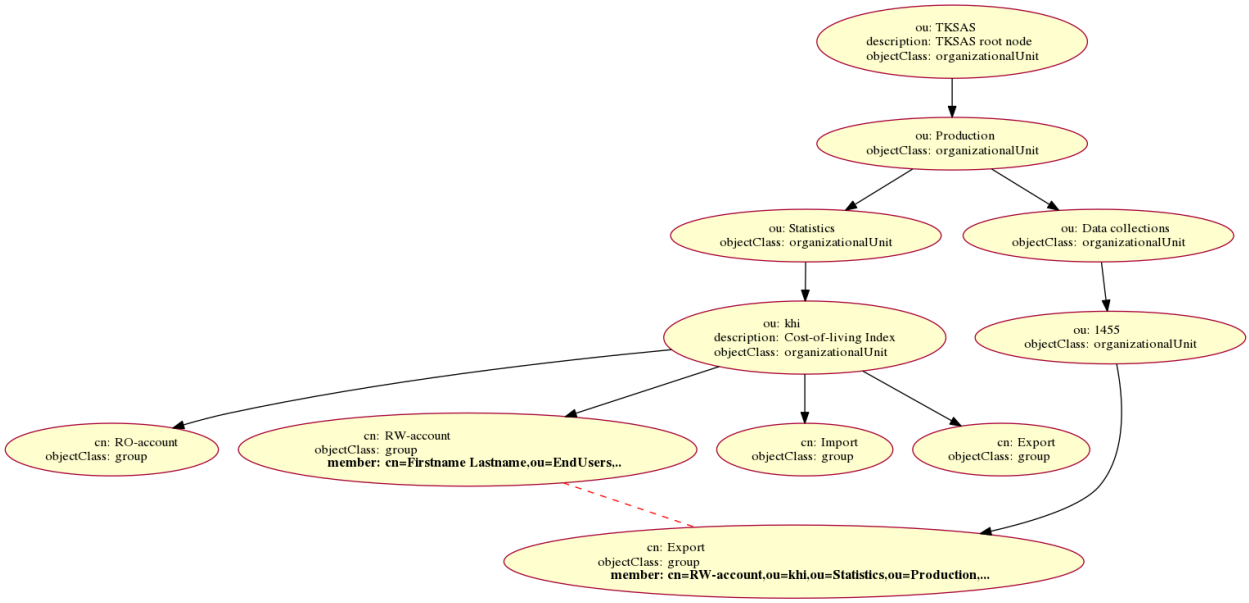

Figure 2: Example of GSU structure in LDAP -tree. Red dashed line depicts read permission given to Consumer price index -statistics (ID khi) to Export -directory of 1455 data collection.

Authentication

Implementing DSoD requires distinct authentication and authorization phases. At first the user is authenticated. After successful authentication the user chooses **one** of the available authorized GSUs as the user is not authorized to access them simultaneously. This clean separation turned up to be technically hard to achieve.

The main obstacle was one-to-one mapping of logins to metadata identities in SAS metadata. As permissions in metadata are tied to a metadata identity, each separate and distinct permission set requires a separate and distinct metadata identity and hence a separate and distinct login.

Some alternative arrangements were discussed.

8. Giving users access to all their metadata at the same time but enforcing DSoD on operating system level (SAS processes).
    - This was considered too confusing for the user.
    - Additionally, it appeared impossible to know *which* permission set to activate for any given process. Object Spawner did not place enough information in SAS processes launch environment.
9. Use of group identities.
    - This was seen undesired by database administrators. Ability to identify the owner of each SQL query was deemed necessery.

A satisfactory solution to the need of having several metadata identities available for a single operating system or eD/AD account has not been found.

In the end distinct OS user accounts and metadata identities were chosen as the policy to implement.

Technically all OS accounts are equal. However, by creating a naming and grouping convention the desired functionality was achieved. Accounts are organized as mainlogins and sublogins. Required permissions are granted through group permissions.

10. Mainlogin is employees personal OS account. Its ID is the same the employee uses to authenticate to workstations and other IT resources in Statistics Finland. Mainlogin is not used for statistics work and it doesn't belong to any GSU resource group.

    For example: muurimak

11. Sublogin it the actual GSU spesific account that employee logins with to work on a GSU. It is a member of exactly one GSU resource group. Connection between sublogin and GSU is implemented simply by naming convention.

    Sublogins are named *mainlogin_GSU type_GSU code*.

    For example: muurimak_til_xxtest

Creating all mainlogins and sublogins as regural Person objects in eD/AD was deemed unfeasible for maintainance and licensing reasons. Dedicated authentication realm that spanned to all hosts in respective TKSAS environment was needed. NIS and LDAP were considered of which NIS was chosen.

SAS metadata

Each environment (production, testing, development) has its own metadata server.

All group memberships in TKSAS master data map to separate SAS Metadata identities.

Automatically created resources:

Metadata identities

For each GSU role a metadata identity is created.

Logins

For each metadata identity a DefaultAuth Login is created that maps to operating system sublogin account with the same id.

Groups

Two kinds of groups are created: GSU groups which is used in access control to GSU folders and personal groups for each distinct employee to hold all metadata identities of the employee.

Personal groups are used to administer employee's login information to external databases. Employees personal group has the OS mainlogin as DefaultAuth Login and SAS Personal Login Manager is the client application.

GSU groups do not have Logins defined. That can change in future if database administering policies change.

ACT

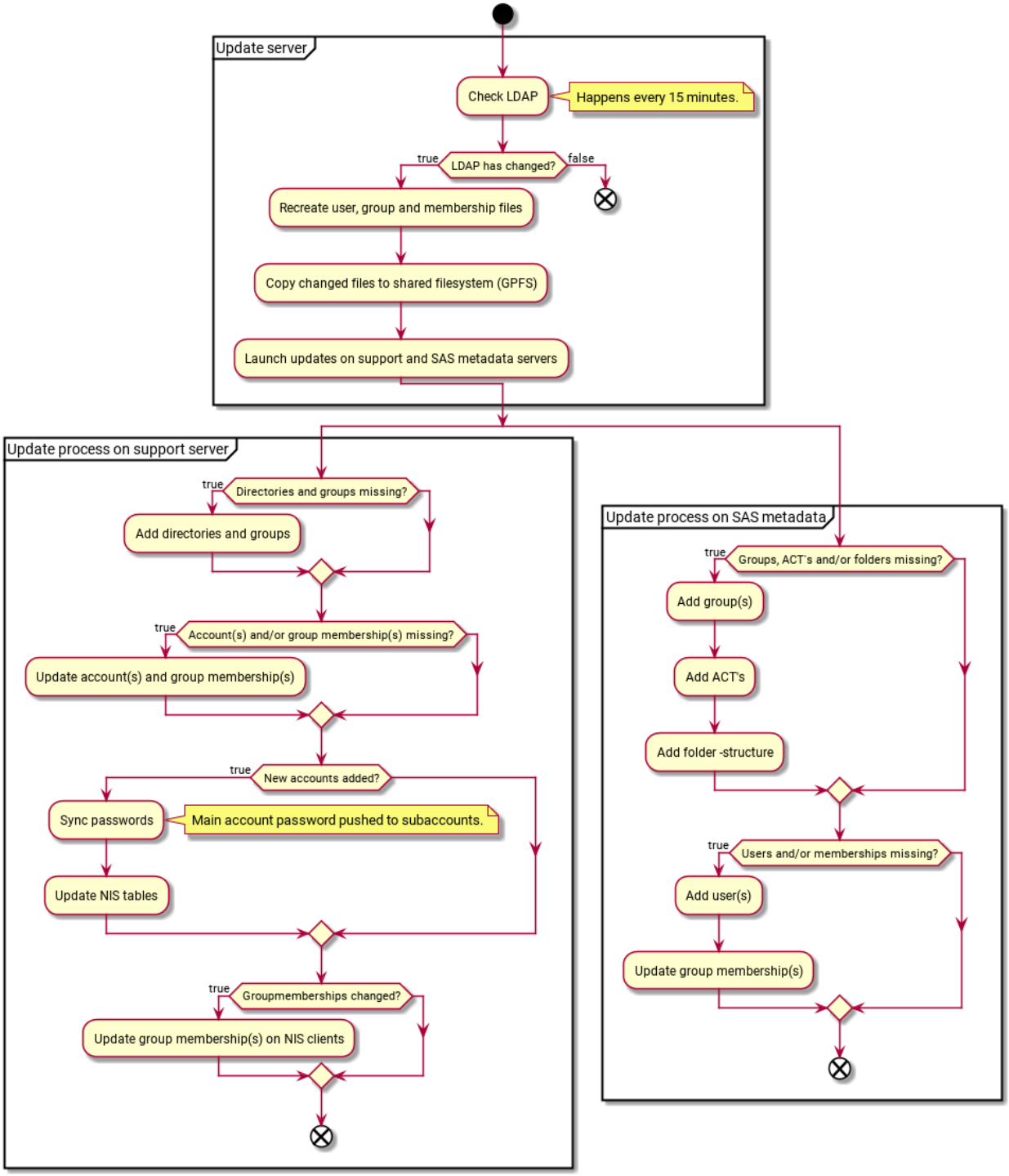

Figure 3: Creating/Updating user accounts and standard resources

Each GSU has several ACTs created for metadata folders -structure access control. GSUs are considered distinct and with any metadata identity only the folder structure of the GSU is tied to is visible.

Folders

Each GSU type has its own top level folder structure. The GSU spesific folder structure is created automatically as a new GSU is created. ACTs are deployed to folders at creation.

Operating system services

SAS users authenticate with SAS client applications using operating system accounts. Additionally access to datasets is controlled on OS level as dataset reside on OS filesystem. Therefore operating system and its facilities play a major role in implementation and running TKSAS SAS server platform.

User accounts

All user accounts are technically regular accounts. However, DSoD is implemented by naming convention as described in .

Employees mainlogin and uid are unique inside Statistics Finland. Special care is needed to ensure that mainlogin's uid (an integer) is correct in all TKSAS environments. Sublogins are internal to TKSAS so there is no need to enforce any particular numbering scheme.

User groups

GSU's user group is the mechanism to implement access control to GSU's SAS resources. Each GSU has currently defined several role groups: For example maingroup for read-write -access to the GSU's all resources, Import group to give write access to the GSU's Import directory to other GSUs and Export group to give read access to the GSU's Export directory to other GSUs.

Sublogin may be a member of only one main group (it's own) but any number of other GSUs' Import and Export groups.

Directories

GSU's standard directory structure is created in SASDATA. Standard dictates two top level hierarchies but underneath them GSU members are free to create what ever hierarchies they see fit. Access to them can't be changed, however.

Filesystem access

Traditional DAC filesystem permission model is not adequate as permissions can only be set to *owner*, *group* and *others*. It is not possible to achieve enough granularity with such limited possibilies and therefore NFSv4 type ACL is used. With ACL:s several different access profiles are created for different groups.

Newly created files or directories always inherit permissions of their context. They are always available to all members of the owning GSU group. No employee can set access permissions of any file or directory – all permissions are strictly enforced by the system.

Servers

Statistics Finland had one Power5 and one Power6 server when designing of TKSAS started in 2011. During the years more servers has been acquired as servers have aged and especially when memory requirements have increased. Currently Statistics Finland has a total of 4 Power7 and Power8 servers of which Power7 -servers are being phased out in 2017.

TKSAS SAS server platform is divided into three environments: production, testing and development. Each environment consists of 5 AIX LPARs of which 2 are support LPARs and 3 are for only running SAS software. SAS is divided into one SAS Metadata server, one SAS Application server and one SAS Middletier server. Support LPARs run TKSAS -support scripts (eg. user, group and directory management), NIS -servers, GPFS -servers and TSM -client (for backup purposes). All in all 15 LPARs are in use.

CPU and memory allocations are different on each server depending on environment and role (metadata, application server or middle tier). They can also be dynamically adjusted with PowerVM.

Each environment has it's own distributed cluster filesystems (GPFS) for both SASDATA and SASWORK and users' home directories. Distributed filesystem means that all filesystem resources are common and accessible from all LPARs in the cluster equally. SAS installations are local to each LPAR.

GPFS was chosen as the filesystem because of it's perfomance as all LPARs can access data through SAN independently from any central server. NFS is ethernet only, requires central server and also suffers from group size limit. These shortcomings made it unsuitable for use.

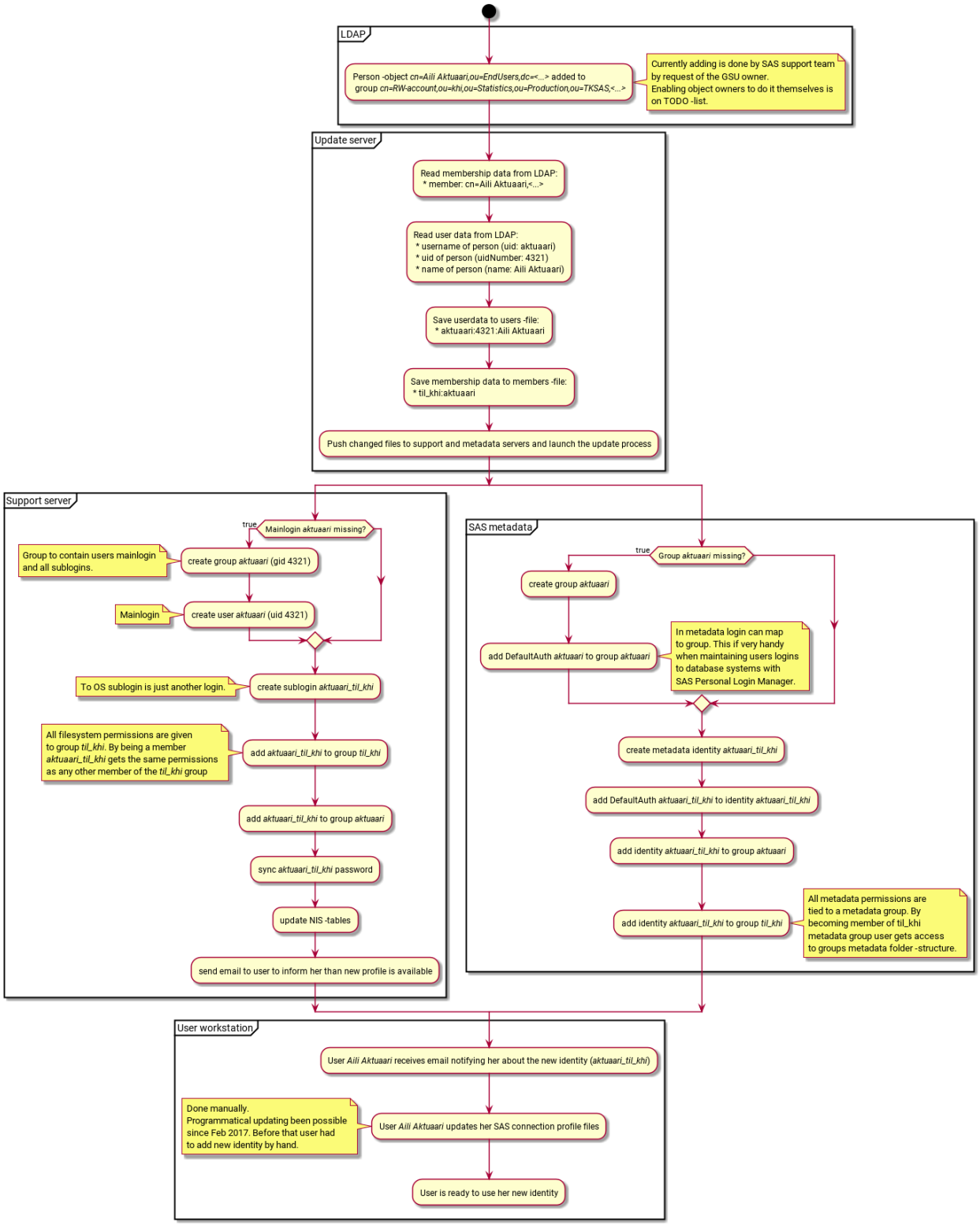

Figure 4 Example of granting permission to a person procedure.

Automation

All automation on TKSAS servers is implemented as shell scripts. Even metadata modification scripts are shell scripts that generate SAS IOM xml that is then fed to SAS metadata procedure. Shell scripts are sufficient since most tasks only concern users, groups and directories for which operating system provides adequete tools and only orchestation is needed.

Reading LDAP and creating input files for SAS servers is another matter. For that a dedicated update server (Linux) is used. Also a real programming language (Python) is used since reading and manipulating LDAP data is more natural that way. Initially LDAP integration was done with shell scripts also but it was deemed "unreadable" and re-implemented in Python when transition from eDirectory to Active Directory was made.

Snippet of "workhorse" shell script that takes filename as an argument and then feeds as input xml to proc metadata. If apropriate command is not available (eg. make-act) required modification to metadata is done with proc metadata and correct xml input file.

```
...
. sas_metaconfig
. sas_credentials

cat  $INFILE > $INXMLFILE

# Mikäli Repositorya ei ole määritelty käytetään
# Foundation-repositorya.

METAREPOS=${METAREPOS=-Foundation}

cat <<EOF > ${SASFILE}
filename myinput "${INXMLFILE}" lrecl=256;
filename myoutput "${OUTXMLFILE}" lrecl=256;

proc metadata
     server="${METASERVER}"
     port=${METAPORT}
     repos="${METAREPOS}"
     userid="${METAUSER}"
     password="${METAPASSWORD}"
     in=myinput
     out=myoutput
     header=full;
run;
EOF

if ${SAS} -LOG ${LOGFILE} ${SASFILE}
...
```

Usage example of SAS provided command to a specific action (create/update ACT). Script enforces ACT naming convention.

```
if is_metaact "Til ${TIL_ID_UPPER} ACT"
then
   log "Meta ACT \"Til ${TIL_ID_UPPER} ACT\" on jo olemassa."
   log "Päivitetään ACT:n tiedot ..."
   ${MAKEACT} -update "Til ${TIL_ID_UPPER} ACT" \
      -grant "SAS General Servers(UserGroup)":ReadMetadata,Read \
      -deny  "SAS General Servers(UserGroup)":WriteMetadata \
      -grant "SAS System Services(UserGroup)":ReadMetadata \
```

```
        -deny  "SAS System Services(UserGroup)":WriteMetadata \
        -grant
"SASAdministrators(UserGroup)":ReadMetadata,WriteMetadata,WriteMemberMetadata,CheckInMeta
data,Administer \
        -grant "til_${TIL_ID}(UserGroup)":ReadMetadata,Read,Write,Create,Delete \
        -deny  "SASUSERS(UserGroup)":ReadMetadata,WriteMetadata,Read \
             -profile BatchProfile
    log "... ACT päivitetty."
else
    log "Luodaan ACT \"Til ${TIL_ID_UPPER} ACT\" ..."
    ${MAKEACT} -create "Til ${TIL_ID_UPPER} ACT" \
```

FINDINGS AND OBSERVATIONS

New SAS server platform (TKSAS) has been in use since 2013. Transition from old SAS server will last until the end of 2018. Long transition period (2014-2018) is required because production has to be reorganized around GSU master data model. That is needed in order to solve the problems described in chapter . Currently over 50% of the volume of statistics productions has moved over.

RESOURCES

In both testing and production following GSUs have been founded: 73 statistics, 105 data collections, 23 statistical subject areas and 12 projects. In production SASDATA has 5.5 TB and in testing 3.5 TB.

Each GSU has standard role groups defined.

Table 1: Number of members in GSU groups

Environment	N GSU groups	Avg. mem.	Min	1. qrtle	2. qrtle	3. qrtle	Max
Production	213	8.6	1	4	8	10	79
Testing	213	9.2	1	5	8	10	83

SUBLOGINS

Currently TKSAS production and testing environments have 350 distinct users each. In production there are 1980 and in testing 2158 sublogins.

Table 2: Logins

Environment	N mainl.	N subl.	Avg. subl.s	Min	1. qrtle	2.qrtle	3. qrtle	Max
Production	347	1980	5.7	2	2	3	4	96
Testing	357	2158	6.0	1	2	3	4	96

SUBLOGIN USAGE

Most interest is focused on how much inconvenience DSoD creates – if any. To find out that exactly, a detailed SAS log file analysis has been conducted. All SAS EG/DI logfiles created on the TKSAS servers from 27.12.2016 to 28.2.2017 were studied.

Log analysis has revealed that while some employees do have very many sublogins, and incidentally they use several of them on a daily basis, this is a rare occurence. Most employees do not have to change their SAS profiles at all because they only have one. Those who do have several profiles, do not need to change between them daily.

Table 3: Log analysis population

Monitored days	Unique employees	SAS EG/DI processes
60	180	25206

Table 4: SAS EG/DI -usage during the period

Number of days SAS was used during the period	Number of employees
1	19
2-5	42
6-10	32
11-15	15
16-25	28
26-35	29
36-	15
	180

Table 5: Sublogins during the period

Number of active sublogins during the period	Number of employees
1	109
2	43
3	15
4	2
5	4
6	2
7-	5
	180

As can be seen from table Taulukko the majority of employees (109 of 180) used only one sublogin during the whole 2 month period. Vast majority (167 of 180) used at most 3 sublogins during the whole 2 month period but not necessarily during the same working day.

Of the employees who used 2 or more sublogins during the 2 month period (71) only 60 used them during the same working day. That means that 2/3 of employees did not encounter the effects of DSoD implementation in any way.

Interestingly, even those employees who have several sublogins, didn't need to change them often during a working day.

Taulukko 6: Maximum active sublogins during a day

Number of maximum active sublogins during a day	Number of employees
2	41
3	13
4	3
5	2
6	1
	60

Most intriguingly, even those employees who have several sublogins and had changed them during a working day, didn't do it most days. In that group EG or DI was used on average during 22.8 working days. Only in 5.67 days (on average) they used 2 or more sublogins during the same working day.

ADMINISTRATION RESOURCE USAGE

TKSAS and its strict adherence GSU and RBAC/DSoD policies and the old SAS server environment are not readily comparable because TKSAS encompasses much wider set of functionality and policies. Hence measuring impact of TKSAS is not straightforward. Some system services are common, however, such as deploying authorizations.

In the old system deploying authorization decisions on each deployment platform is done by IT personnel in charge of the platform in question. More often than not there is a lack of clear understanding between statistics producers and IT personnel of what any particular requested authorization change means. Consequently results and time used can vary considerably.

On TKSAS platform no ongoing IT personnel involment is required. Requested authorizations are activated by SAS-team by modifying group memberships in LDAP -tree. Direct access to deployment platform is not needed. As there are clear unambiguous policies in place, all the required operations can be (and have been) automated. The IT personnel involvement have focused on upfront automation instead of continuous operation.

CONCLUSION

Transition to the new TKSAS SAS server platform has not yet been finalized so it is unfortunately too early to say anything definitive about the usefulnes of the approach. However, some preliminary observations can be made.

Using and enforcing GSU policy hasn't halted statistics production. Mandating GSU creation have caused at most some delays as old habits and methods have suddenly become obsolete. Mostly problems have arisen when a large software system is being newly developed and there is no clear understanding on data needs and requirements or data flows between GSUs. After systems stabilize problems tend to dissappear or workarounds can be developed.

Mandating sublogin (DSoD) usage hasn't made work unbearable for employees. The results in subchapter ”Sublogin usage” on page 12 seem to imply that the use of sublogins is an non-issue altogether. At most it has effects on power users and it can be argued that some of those effects are actually positive as system predictability, stability and transparency have increased.

While the problems seem to be smaller than first feared all the benefits have materialized. Resource and authorization deployments are automated, systems usage has expanded without system complexity increasing simultanously and a predictable and rational statistics production model for common use has been introduced. Implementing architecture of master data model have as a consequence increased system rationality, lessened error frequency and improved resource findability considerably. System and application specific guides and rules for authorization management have become unnecessary as they have been converted to use the generic master data model. Remaining problems are mainly in databases.

Transition is not over, however. Only at the end of 2018 can the final word be given. Major challenges need to be tackled with especially in database management as master data model has only recently started to have impact in that space. Nonetheless, the combined GSU and RBAC/DSoD approach as a whole has been validated.

REFERENCES

Sandhu, R., Ferraiolo, D.F. and Kuhn, D.R. "The NIST Model for Role Based Access Control: Toward a Unified Standard". Accessed March 5 2017 http://csrc.nist.gov/rbac/sandhu-ferraiolo-kuhn-00.pdf

Ninghui Li, Ji-Won Byun, Elisa Bertino. "A Critique of the ANSI Standard on Role Based Access Control". Accessed March 5 2017 https://www.cs.purdue.edu/homes/ninghui/papers/aboutRBACStandard.pdf

Kuhn, D.R. "Mutual Exclusion of Roles as a Means of Implementing Separation of Duty in Role-Based Access Control Systems"

UNECE: General Statistical Business Process Model (GSBPM). Accessed March 5 2017 http://www1.unece.org/stat/platform/display/GSBPM/GSBPM+v5.0

UNECE: General Statistical Information Model (GSIM). Accessed March 5 2017 http://www1.unece.org/stat/platform/display/gsim/GSIM+Specification

Statistics Finland. Common Structure of Statistical Information (CoSSI). Accessed March 5 2017 http://www.stat.fi/org/tut/dthemes/drafts/cossi_en.html