Global Enablement & Learning

Recommended SAS® 9.4 Security Model Design: Data Integration

First Edition

Contact Information

Name: David Stern

Title: Principal Technical Architect, SAS Global Enablement & Learning

Phone Number: +44 1628 490851 Cell: +44 7775 754259

E-mail address: David.Stern@sas.com

THE POWER TO KNOW.

Revision History

Version	By	Date	Description
0.1	David Stern	August 2016	Initial creation
0.2	David Stern	23 August 2016	Minor changes for consistency with paper for SAS Visual Analytics
0.3	David Stern	September 2016	Added content on libraries and config files
0.4	David Stern	November 2016	Updates in response to review feedback
1.0	David Stern	December 2016	Published
1.1	David Stern	December 2016	Granted permission to share with customers

References

Ref	Document Title	By	Date	Description and source
Ref 1	Metadata Security in SAS® 9.4 – Step-by-Step	Johannes Jørgensen, Cecily Hoffritz	September 2013	Fourth version of Metadata Security in SAS® – Step-by-Step, for SAS® 9.4. Contains a best practice for security implementation and [at the time of publication] has been the de facto standard [for implementation of SAS metadata security] for more than 7 years in Denmark. Source: http://misksapm.na.sas.com/KnowledgeSharingApplication/AdvSearchDisplayArtifact.jsp?Entry_ID=4156

Table of Contents

1 Introduction

1.1 Series Overview

Figure 1 below is an overview of this series of papers, which together represent the GEL recommended practices for security model design in SAS 9.4.

GEL is an abbreviation used widely in these papers, to refer to the SAS Global Enablement and Learning team, part of SAS Global Consulting Services.

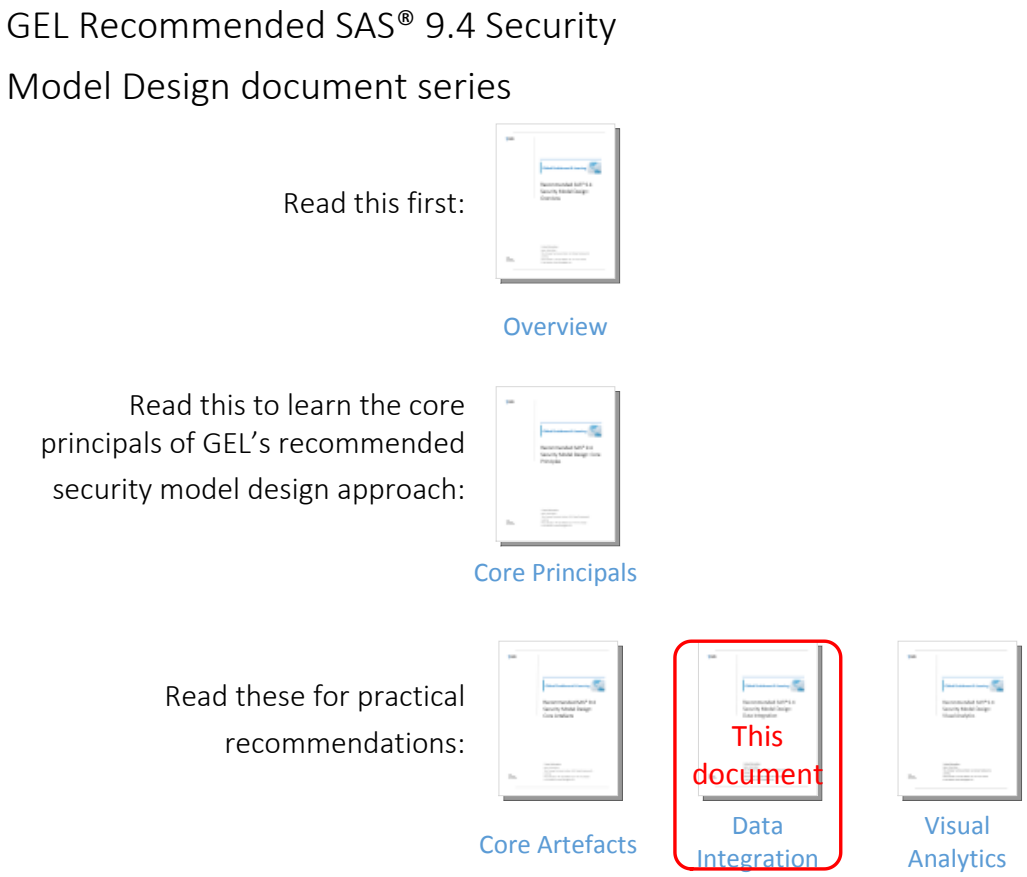

Figure 1 - Overview of papers in this series

1.2 Purpose of this paper

This paper is the first edition of the GEL Recommended SAS 9.4 Security Model Design for **Data Integration**.

It is one of a series of papers discussing recommended practices for security model design in SAS 9.4. This paper covers SAS 9.4 security model concepts that apply specifically to SAS platforms where SAS Data Integration Studio is present. If you have not done so already, you should start by reading

the Recommended SAS® 9.4 Security Model Design: **Core Principles**. Then, follow the guidance in the GEL Recommended SAS 9.4 Security Model Design for **Core Artefacts** paper. We intend that this paper be used as an extension of the guidance in those two documents, rather than as an alternative, or as a standalone document.

In this paper, we recommend a template pattern of metadata folders and filesystem directories, group, ACTs and permissions, which you should implement on all SAS deployments with Data Integration.

See other papers in the "Recommended SAS® 9.4 Security Model Design" series for solution-specific recommendations for other solutions.

You may find this process quicker and easier if you use the set of scripts that we have developed to 'turbocharge' (i.e. speed up) your security model setup. The GEL Turbo scripts and accompanying resources are discussed in the **Core Principles** paper.

1.3 Assumptions

This paper assumes you have read the Recommended SAS® 9.4 Security Model Design: Core Principles, and have some awareness of SAS Data Integration Studio.

This paper also assumes that you can make a consistent system-wide backup, from which you know you can successfully restore, before you make any changes recommended in this paper. We further assume that you have done so immediately before you begin making changes, and may make additional backups at sensible points throughout the process of making these changes.

1.4 Permission to share this document

2 Naming conventions for multitenancy and multi-environment ecosystems

The Core Principles paper discusses naming conventions for objects in security models in general terms, and in particular, for security models in SAS deployments that currently (or will, in the future) support some form of multitenancy. You should adopt and follow clear naming convention for objects implementing your security model, because each tenant of the platform will have its own set of similar objects, and you must be able to identify them easily and distinguish between the ones belonging to each tenant.

The naming convention below allows for consistent naming throughout an *ecosystem* of related SAS deployments: for example, a Development, Test and Production environment which are all related to each other, and form a 'route to live' for application content. In such an example arrangement of environments, application content is developed in Dev, promoted to Test and then promoted again to a live Production environment. Of course, other arrangements of environments (and we are specifically interested in the SAS deployments in those environments) in an *ecosystem* are easily catered for in these naming conventions, so long as each environment has a distinct name.

We recommend names in this document that take follow patterns illustrated in the following examples. This is not an exhaustive list, but the following examples are enough to illustrate the pattern:

1. LDAP-synchronised dynamic group named "**SAS Dev Rigel DI Developers**" (where 'Rigel' is the name of an example tenant organisation)
2. Static shadow group named "**Rigel DI Developers**"
3. ACT named "**Rigel DI Developers ACT**"
4. Libref "**RigelSRC**"

In the table which follows, we break down the components of each of those example names, to explain what each component is, and when and why they are needed.

Ex.	Object Type	Full Object Name	Component	Explanation
1	Group (LDAP-synch-ronised dynamic group)	SAS Dev Rigel DI Developers	SAS	A group of this name should exist in the customer's Active Directory system, and a group of this name should also exist in SAS metadata. The 'SAS' prefix helps AD administrators identify the group as being important for SAS, and means that it sorts alphabetically together with other SAS groups in Active Directory. The 'SAS' prefix also serves the purpose of identifying the LDAP-synchronised corresponding group in SAS Metadata as having come from Active Directory synchronisation; there is no other reason to prefix a group name in SAS metadata with 'SAS'! If you are using shadow groups, leave this prefix on the dynamic group name. If you are not using shadow groups, you can remove it from the group name in SAS Metadata.
			Dev	The second component of this name signifies that this group should contains users of the SAS Development environment, in an organisation where there is an ecosystem of related Dev, Test and Prod environments. Allows users to be put in a group like DI Developers in different environments, so that the permissions assigned to the static *group* can be consistent across all environments in the ecosystem, but one *user* does not necessarily get the same permissions in all environments. If you are using shadow groups, you can leave this prefix on the dynamic group name (so that it is the same in metadata as in LDAP). If you are not using shadow groups, you MUST remove it from the group name in SAS Metadata – there is no need to include

				the environment name in a group in the environment! And all your environments in an ecosystem should have groups of the same name in their ACTs (and this group will be used in ACTs if you are not using shadow groups).
			Rigel	Important for SAS ecosystems which support multitenancy, the long name of the 'tenant' organisation. Distinguishes this group from another tenant's DI Developers. **Optional if your SAS deployment is always going to be strictly single-tenant.**
			DI Developers	Indicates what members of this group do.
2	Group (static shadow group)	Rigel DI Developers	Rigel	Important for SAS ecosystems which support multitenancy, the long name of the 'tenant' organisation. Distinguishes this group from another tenant's DI Developers. **Optional if your SAS deployment is always going to be strictly single-tenant.** The shadow group is static, in the sense that it can't be accidentally deleted if the Active Directory group 'SAS Dev Rigel DI Developers' is deleted or if synchronisation with LDAP fails. This group is therefore safe to use in ACTs. No 'SAS' prefix: this group only exists inside SAS metadata, so does not need to be labelled as 'SAS'. No 'Dev' prefix: within this metadata repository, everything belongs to the same SAS deployment, so no qualification is necessary. This also keeps the name of this group consistent across all deployments in the ecosystem: it exists with the exact same name in each one.

				If you wish, you can also choose a naming standard so that shadow groups must have a suffix of '_SG' after the group name. We have not shown this suffix here in the table, but it is perfectly acceptable and sensible. Please either use or do not use such a suffix *consistently*, in the same way for all shadow groups – all must have it nor none must have it. It is not optional *per group*.
			DI Developers	Indicates what members of this group do.
3	ACT	Rigel DI Developers ACT	Rigel	Important for SAS ecosystems which support multitenancy, the long name of the 'tenant' organisation. Distinguishes this tenant's DI Developers ACT from another tenants' DI Developers ACTs. **Optional if your SAS deployment is always going to be strictly single-tenant.**
			DI Developers	Indicates which group this ACT features.
			ACT	This suffix is redundant when discussing ACTs only, but in a broader context when we may discuss ACTs together with groups and folders, helps identify this object as an ACT.
4	Libref	RigelSRC	Rigel	Librefs are limited to 8 characters. We allow up to 5 characters for a short version of the tenant or project's name. If there are multiple tenants, and each has multiple projects/teams/data topics etc, this part of a libref name can become very condensed: you may need to design a naming convention where e.g. the first three characters indicate the tenant organisation, and the next two indicate the project/team/data topic.

			SRC	Librefs are limited to 8 characters. We allow up to 3 characters for an abbreviation of the library name: “Source” or “Source Data” in this case.

Table 1 – Example object names illustrating our recommended naming convention

You should use the structure above if it will work for your customer. Apart from providing a regular naming convention that allows you to name a large number of objects in a multi-tenant ecosystem of related SAS deployments, another benefit of following this naming convention in particular is that SAS staff who are familiar with these recommendations will immediately understand it, and you will waste less time explaining your naming convention to them.

If the structure described above will not work for your customer (e.g. because your customer has some more complex internal organisation which must be represented in more complex object names), you may absolutely deviate from the naming convention above. However, you should then document your naming convention clearly, and use it consistently across all the SAS deployments that belong together in an ecosystem, for this customer.

3 Static metadata groups

'Static metadata groups' in SAS metadata are static in the sense that the process of synchronising groups with Active Directory or another LDAP provider does not create or destroy them.

See the discussion of dynamic groups synchronised from LDAP, and their corresponding Static Groups in the Recommended SAS® 9.4 Security Model Design: **Core Principles** section titled "Identify groups of users who will use the assets".

For SAS deployments that include Data Integration Studio, create one of each of the following groups for each 'tenant' organisation in each SAS deployment in your ecosystem. If this SAS deployment will **never** need to support multitenancy, omit the 'tenant' part of each group name.

Metadata Group	Description	Uses DI Folder Colour[1]
Optional group – only create this group if you need it: **[Tenant] Analysts**	Contains expert users working with model-based statistical analysis, forecasting, or data mining. Preferred applications: • SAS® Enterprise Miner™ • SAS® Visual Analytics Explorer • SAS® Forecast Server • SAS® Enterprise Guide® • SAS/STAT® • SAS® Programming in an editor • …	**Light-Blue: Specialist Folders.**
Optional group – only create this group if you need it: **[Tenant] BI Developers**	Contains expert users automating, integrating, and distributing reports and analyses to the organization. Preferred applications: • SAS® Enterprise Guide® • SAS® Stored Processes • SAS® Information Map Studio • SAS® Web Report Studio • SAS® Add-In for Microsoft Office • SAS® BI Dashboard	**Dark Blue: Business User Folders**

[1] See colour-coding of Metadata Folders for Data Integration Studio below

	• SAS® Visual Analytics • SAS® Programming in an editor	
[Tenant] DI Developers	Contains expert users creating jobs to extract, transform, and load data and other data management tasks. Preferred applications: • SAS® Data Integration Studio • SAS® Data Management Studio • SAS® Programming in an editor	**Red: Data Management Folders**
Optional group – only create this group if you need it: **[Tenant] Report Consumers**	Contains end users viewing information/reports, typically from the company's intranet. Preferred applications: • SAS® Portal • SharePoint • SAS® Visual Analytics Report Viewer • Mobile devices (iPad) • …	**Dark Blue: Business User Folders**
Optional group – only create this group if you need it: **[Tenant] Report Creators**	Contains super users designing and building reports, typically for own department. Preferred applications: • SAS® Web Report Studio • SAS® Add-In for Microsoft Office • SAS® Visual Analytics Report Designer	**Dark Blue: Business User Folders**
[Tenant] SASBatch	Contains batch user. For deploying and scheduling jobs, especially in a change-managed environment where your user is not authorized to do this. Also used for export of metadata to SPK file in batch. • Preferred applications: • SAS® Data Integration Studio • SAS® Management Console • Platform Suite for SAS	**Red: Data Management Folders**

Table 2 – Metadata Groups for SAS Data Integration Studio, describing the users belonging to each group and the 'colour' code of folders the group uses

Readers familiar with the SAS Denmark Metadata Security Standards (**Ref 1**) may recognise the colour coding in the third column of Table 2 above. We have adopted the same colour coding here.

Since the time when **Ref 1** was written and published, SAS Web Report Studio has become much less widely deployed, and SAS Visual Analytics has become available as its replacement, to some extent. Some customers still plan to have users who use Stored Processes for significant parts of their work, and for those customers, the light and dark blue groups in the table above are still useful. But many other customers would have little use for the dark blue groups in this design. They remain part of the recommended design at present, for customers who may still find a use them. You may omit them if you know you will not use them.

In contrast to the static metadata groups recommended for SAS Visual Analytics in the paper for that solution, there are no Data Integration-specific groups for more than one separate Department (or Project, Team, Subject Area, Data Topic, or sub-organisation etc.). There is no obvious departmental placeholder such as [Dept1], in this design. This is because it is relatively unusual to have Data Integration users in multiple departments in the way that is so much more common for SAS Visual Analytics users. However, if a tenant organisation does have DI developers, analysts etc. in separate departments, you can of course design multiple groups of each type, one per department, project, team etc. as required.

4 Metadata folders

For SAS deployments that include Data Integration Studio, create one of each of the following metadata folder structures for each tenant organisation in your SAS deployment.

Some deployments may have several separate 'tenants' who may wish to share some data, or other resources (jobs, formats etc.) with one another. If this applies to a customer, in that customer's multitenant structure, consider creating an additional 'global', 'shared' or 'common' tenant metadata folder structure, in which common resources (data, formats, jobs etc.) can be placed, and made accessible to each of the other tenants. Of course, some deployments which multiple tenants will not require this, especially if the tenants are truly separate customer organisations.

If this SAS deployment will **never** need to support multitenancy, create one copy of this metadata folder structure, but we recommend that you *still group these folders inside a common top-level folder* named after the organisation or project. Doing this keeps the more 'technical' Data Integration folder structure neatly contained within a folder so that you can more easily secure it, and keep it hidden from non-technical end users who access SAS metadata via reporting tools such as SAS Visual Analytics.

Libraries registered in metadata are stored beneath these folders, together with their tables, and not separately in some dedicated 'Libraries' folder. The Formats folder contains a metadata registration of a Formats library named e.g. Rigel Formats_RigelFMT.

Note the colour coding: 'red folders' are always hidden from all 'blue users' (both light blue or dark blue). The Data Integration folders in red can be secured very simply if you are careful NOT to create any folders which 'blue users' will access inside the 'red folder' structure. You will have to perform tedious application of additional ACTs in order to secure 'blue folders' below 'red folders' correctly, and may cause difficult-to diagnose permissions conflicts if you do not secure those 'blue folders' correctly. We strongly recommend you avoid doing this.

Table 3 – Metadata Folders for SAS Data Integration Studio

In contrast with the folder structure recommended for SAS deployments that include Visual Analytics, this folder structure is deeper, and has numeric prefixes before most of the folder names. Users of SAS Data Integration Studio tend to be highly technical, and involved deeply in building the application functionality for which the SAS deployment has been commissioned. They work with a large number of inter-related metadata objects, and require a highly organised metadata folder structure that allows them to keep all these objects in their right places.

[2] Depending on your needs, create either 05_Data_Marts or 05_Analytical_Marts, but not both.

Please remember that **the folder structure above is only a template**, a starting point that you should adapt to your customer's needs:

- Some customers will have data, or requirements that make some of the folders in the template folder structure shown above unnecessary. For example, a customer's source data may be already clean, so that a staging area is not required. Or they may have only one source system, rendering a System2 folder unnecessary in each of several parent folders. (You would of course *rename* all of the System*n* folders to have the name of their actual source system e.g. Oracle, Teradata etc. – 'System*n*' is a placeholder name.)
- Other customers may need additional folders not shown here. For example, it is not unusual for data originating in separate source systems to be brought through the layers of the ETL process as far as the 03_Detail_Data_Store still in separate subfolders for System1, System2 etc. Please do create such subfolders if your project requires them.

You should not use (and in fact, you should delete) any folders you do not require in a particular customer deployment. It is unnecessary and actively counterproductive to have empty folders; empty folders are confusing to the users. Do not create unnecessary folders in your metadata and filesystem folder structures simply because this series of papers shows them or recommends could create them.

The next section describes a similar structure of filesystem folders.

5 Filesystem folders

For SAS deployments that include Data Integration Studio, create one of each of the following filesystem folder structures for each 'tenant' organisation in your SAS deployment. If this SAS deployment will **never** need to support multitenancy, create one copy of this metadata folder structure, but we recommend that you group these folders inside a common top-level folder named after the organisation or project. Doing this keeps the more 'technical' Data Integration folder structure neatly contained within a folder that mirrors the top level folder in the metadata folder structure.

As with the metadata folder structure, the table shown below is only a template. If your SAS deployment stores most of its data in an third party database (Oracle, Hadoop, Teradata), you may not need some of the folders shown, since some or all of the physical data you are working with will be stored in that database, not in SAS datasets on the filesystem. Only create (or keep) the folders you need.

Similarly, if you need additional folders, create them as required, following your chosen Data Integration development standards. The folder structure here is intended as a starting point, and absolutely not a rigid structure which you are prohibited from changing.

We recommend that filesystem folders are named in all lowercase, on both Windows and Unix.

In the table below, the top level folder is shown as …\sasfolders. This is intended to mean that you should create a folder called 'sasfolders' in an appropriate place on a filesystem accessible from all the SAS servers (Workspace Server, Stored Process Server) in your SAS deployment's main Application Server Contexts (e.g. SASApp). For example, on Unix you may place this folder at /opt/sas/data/sasfolders, or on Windows you may place it at D:\sasfolders.

You may choose another name for the 'sasfolders' folder if you prefer. The suggested name is meant to mirror the "SASFolders" top-level folder in metadata.

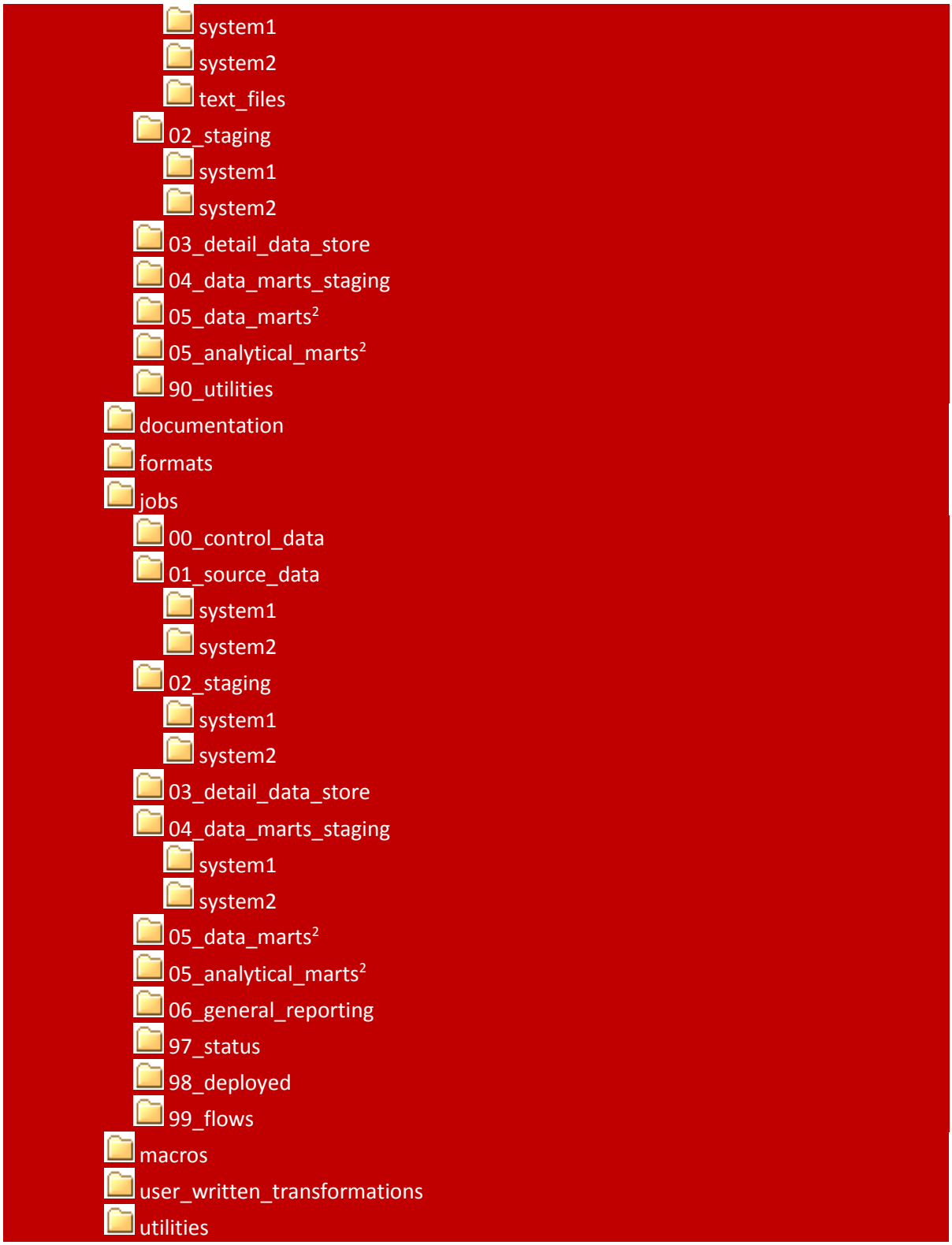

Table 4 – Filesystem Folders for SAS Data Integration Studio

6 Libraries

6.1 Base Libraries

For SAS deployments that include Data Integration Studio, create one of each of the following BASE libraries for each 'tenant' organisation in your SAS deployment. In this table:

- The placeholder string [TEN] should be replaced with a maximum 5-letter abbreviation for the tenant organisation name in the libref.
- The placeholder string [Tenant] should be replaced with the longer name of the tenant organisation.
- Create one copy of each of the libraries containing the placeholder string [Dept1] for each department (etc.) that a given tenant has.

To learn which filesystem folder each Filesystem Alias refers to, see section 7 below.

Librefs are constrained to a maximum length of 8 characters. In our naming convention, five of these characters are reserved to contain an abbreviation of the name for the tenant organisation. The remaining 3 characters in each libref (e.g. CD, SRC) are used for a somewhat cryptic-looking 3 letter code identifying the library. While they are not necessarily intuitive to an uninformed reader, we intend them to make the library somewhat recognisable to an informed reader who knows what these 3-letter codes are, by the libref alone.

Libref	Library Name	Metadata Folder	Filesystem Alias	Description
[TEN]CD	[Tenant] Control Data_([TEN]CD)[3]	/[Tenant]/Data/00_Control_Data	![Tenant]_cd	For tables used in controlling and monitoring

[3] There are two reasons for having libref in parentheses at the end of the text library name. First, convenience: in SAS Enterprise Guide, when you see the library, you can also see the libref without having to keep looking at the properties. This makes writing (or reading) code which references the library a little easier and more intuitive. Second, when you create a library, Metadata Server does not check for uniqueness for libref, it only checks for uniqueness of the text library name. Having the libref in each of the text library names makes it easier for the administrator or DI developer to check by eye that a new library he is creating does not use a libref which is already in use.

				the behavior of DI job flows.
[TEN][SC1..*n*]SC[4]	[Tenant][SC1] Source Data_([TEN][SC1..*n*]SC)	/[Tenant]/Data/01_Source_Data/[System1..*n*]	![Tenant]_src/[System1..*n*]	Source Data from source system 1..*n*[5]. Additional libraries should be defined for each additional source system.
[TEN][SC1..*n*]ST	[Tenant] Staging Data_([TEN][SC1..*n*]ST)	/[Tenant]/Data/02_Staging/[System1..*n*]	![Tenant]_stg/[System1..*n*]	Staging data for data extracted from source system 1..*n*. Additional libraries should

[4] Choosing good librefs for Source and Staging (and sometimes Detail Data Store) libraries in SAS deployments which have both multitenancy AND multiple source systems is difficult. You are limited to 8 characters in a libref. In that 8-character string, we would like to convey: 1. the tenant's name, 2. the data source's origin and 3. that this is the Source, Staging or DDS library for that tenant for that source system. It is tough to do that in only 8 characters, while choosing a libref which makes intuitive sense to your DI developers. It is slightly easier if you only have one tenant, or one source system, since you can then omit one component of the compound string.

[5] Instead of the string 'TENSC1SC', which is intended here as a compound of three example name components, make this string something more meaningful in your SAS deployment. E.g. in place of 'TEN' (the tenant), use the client tenant's name, e.g. 'Rigel Enterprises' abbreviated as 'RGL'. Instead of 'SC1' (source system 1), use an abbreviation of the source system's name, OR1 or OR2 (Oracle 1 or Oracle 2), or TD or HDP (Teradata or Hadoop), or if it makes better sense in your deployment, something like CRD (Credit Risk) or LAB (Lab Results). Keep the last two letters: SC=Source data, ST=Staging, DD=Detail Data Store. If you don't require any one of these three elements, e.g. because there is only one tenant, or only one source system, don't include it, and use the 8 characters more freely for the remaining elements you do need. For example, if you don't keep need DDS libraries for your different source systems because they are already combined by the DDS layer, you can name the DDS library simply as [TEN]DDS, which is much simpler.

				be defined for each additional source system.
[TEN]DDS	[Tenant] Detail Data Store_([TEN]DDS)	/[Tenant]/Data/03_Detail_Data_Store	![Tenant]_dds	Detail Data Store[6] for this tenant.
[TEN]DMS	[Tenant] Data Marts Staging_([TEN]DMS)	/[Tenant]/Data/03_Data_Marts_Staging	![Tenant]_dms	Staging area for the Reporting or Analytics (or other) Data Mart for this tenant.
[TEN]DM	[Tenant] Data Marts_([TEN]DM)	/[Tenant]/Data/03_Data_Marts	![Tenant]_dm	Reporting or Analytics (or other) Data Mart for this tenant.
[TEN]FMT	[Tenant] Formats_([TEN]FMT)	/[Tenant]/Formats	![Tenant]_sasdwfmt	DI Formats library for this tenant

Table 5 – GEL Recommended 'DI' Base and Format Libraries to include in security model designs for deployments which include SAS Data Integration Studio

[6] In some implementations, you may have multiple Detail Data Stores, one per source system. If so, name as per Source and Staging libraries.

7 Modifications to AppServer Config Files

In this section, example entries in the main sasv9_usermods.cfg file, and in supplemental configuration files, are shown with placeholder strings as follows:

Placeholder string	Substitute this with
`[tenant]`	The tenant's name written without spaces
`sasfolders_dir`	The full path to that tenant's sasfolders directory
`[DQ LOCALES]`	The name of any Data Quality Locales you wish to set
`[QKB PATH]`	The last part of the folder path to the Quality Knowledge Base for Data Management Studio for this tenant

Following the suggested template for each set of configuration lines, examples are shown in which the placeholder strings have been substituted for a tenant called "Rigel" whose sasfolders directory is at D:\sasfolders on Windows, or /opt/sas/data/sasfolders on Unix.

7.1 SASApp/sasv9_usermods.cfg

For each DI tenant using the SAS deployment, add one copy of a set of lines similar to those in the following examples to the sasv9_usermods.cfg file in each AppServer Context which users belonging to that tenant will use.

Note that the GEL Recommended SAS 9.4 Security Model Design: **Core Artefacts** paper explains how an alias for `![tenant]_sasfolders` is defined in the sasv9_usermods.cfg file.

Notice also how the last '-config' line added to the SASApp/sasv9_usermods.cfg file causes SAS to also read the tenant-specific DI config file in the tenant's sasfolders directory. That file is described in section 7.2 below.

Take care when editing sasv9_usermods.cfg to not specify invalid values: an error reading the config file can prevent SAS from starting a workspace, batch or other SAS session.

7.1.1 Windows

Note that on Windows the filesystem paths must be in double quotes, even when there are no spaces in the path.

```
-DQLOCALE ([DQ LOCALES])
-DQSETUPLOC "sasfolders_dir\[tenant]\qkb\[QKB PATH]"
-NOFMTERR
-EOC=no
```

```
-SET [tenant]_difolders "sasfolders_dir\[tenant]"

-config "![tenant]_sasfolders/[tenant]_di_usermods.cfg"
```

Here is what this might look like with the placeholder strings replaced with example values:

```
-DQLOCALE (DADNK ENUSA)
-DQSETUPLOC "D:\sasfolders\Rigel\qkb\ci_2013a_course_qkb"
-NOFMTERR
-EOC=no

-SET Rigel_difolders "D:\sasfolders\Rigel"

-config "!Rigel_sasfolders/Rigel_di_usermods.cfg"
```

7.1.2 Unix and Linux

Note that on Unix and Linux, the filesystem paths do not need to be quoted unless they contains spaces.

```
-DQLOCALE ([DQ LOCALES HERE])
-DQSETUPLOC sasfolders_dir\[tenant]\qkb\[QKB PATH]
-NOFMTERR
-EOC=no

-SET [tenant]_difolders sasfolders_dir\[tenant]

-config ![tenant]_sasfolders/[tenant]_di_usermods.cfg
```

Here is what this might look like with the placeholder strings replaced with example values:

```
-DQLOCALE (DADNK ENUSA)
-DQSETUPLOC /opt/sas/data/sasfolders/Rigel/qkb/ci_2013a_course_qkb
-NOFMTERR
-EOC=no

-SET Rigel_difolders /opt/sas/data/sasfolders/Rigel

-config !Rigel_sasfolders/Rigel_di_usermods.cfg
```

7.2 …/sasfolders/[tenant]_di_usermods.cfg

For each DI tenant using the SAS deployment, create a separate SAS config file in the tenant's sasfolders directory, named [tenant]_di_usermods.cfg, containing the lines below. These define the libref filepath aliases listed in section 6 above.

Notice that there are aliases and paths defined for the *parent folders* of the source and staging libraries. This does not mean that they limit you to only one source and only one staging library per tenant. See section 6.1 above, Table 5 which shows how libref, library name, metadata folder paths

and filesystem aliases might be named for more complex Data Integration designs, having multiple source systems as well as multiple tenants. In that table, characters indicating the source system are included as part of the libref and library name, but are *appended on to the end* of the filesystem path and path alias, preceded by forward slash, like this: ![Tenant]_src/[System1..*n*].

The aliases in the …/sasfolders/[tenant]_di_usermods.cfg file described above are thus able to support deployments with multiple source systems.

7.2.1 Windows

Note that on Windows the filesystem paths must be in double quotes, even when there are no spaces in the path.

Notice how on Windows, you can use another filesystem alias (such as !Rigel_difolders) in the second argument to a –SET line.

The template content to put in this file is:

```
-SET [tenant]_sasdw     "![tenant]_difolders/sasdw"
-SET [tenant]_cd        "![tenant]_difolders/sasdw/data/00_control_data"
-SET [tenant]_src       "![tenant]_difolders/sasdw/data/01_source_data"
-SET [tenant]_stg       "![tenant]_difolders/sasdw/data/02_staging"
-SET [tenant]_dds       "![tenant]_difolders/sasdw/data/03_detail_data_store"
-SET [tenant]_dms       "![tenant]_difolders/sasdw/data/04_data_marts_staging"
-SET [tenant]_dm        "![tenant]_difolders/sasdw/data/05_data_marts"
-SET [tenant]_datamarts "![tenant]_difolders/data_marts/data"

-SET [tenant]_sasdwfmt  "!Rigel_difolders/sasdw/formats"

-insert fmtsearch [tenant]_sasdwfmt
```

Here is what this might look like with the placeholder strings replaced with example values:

```
-SET Rigel_sasdw     "!Rigel_difolders/sasdw"
-SET Rigel_cd        "!Rigel_difolders/sasdw/data/00_control_data"
-SET Rigel_src       "!Rigel_difolders/sasdw/data/01_source_data"
-SET Rigel_stg       "!Rigel_difolders/sasdw/data/02_staging"
-SET Rigel_dds       "!Rigel_difolders/sasdw/data/03_detail_data_store"
-SET Rigel_dms       "!Rigel_difolders/sasdw/data/04_data_marts_staging"
-SET Rigel_dm        "!Rigel_difolders/sasdw/data/05_data_marts"
-SET Rigel_datamarts "!Rigel_difolders/data_marts/data"

-SET Rigel_sasdwfmt  "!Rigel_difolders/sasdw/formats"

-insert fmtsearch Rigel_sasdwfmt
```

7.2.2 Unix and Linux

Note that on Unix and Linux, the filesystem paths do not need to be quoted unless they contains spaces.

Notice now on Unix, you cannot use another filesystem alias in the second argument to a –SET line. You have to give the target directory in full.

The template content to put in this file is:

```
-SET [tenant]_sasdw      sasfolders_dir/[tenant]/sasdw
-SET [tenant]_cd         sasfolders_dir/[tenant]/sasdw/data/00_control_data
-SET [tenant]_src        sasfolders_dir/[tenant]/sasdw/data/01_source_data
-SET [tenant]_stg        sasfolders_dir/[tenant]/sasdw/data/02_staging
-SET [tenant]_dds        sasfolders_dir/[tenant]/sasdw/data/03_detail_data_store
-SET [tenant]_dms        sasfolders_dir/[tenant]/sasdw/data/04_data_marts_staging
-SET [tenant]_dm         sasfolders_dir/[tenant]/Rigel/sasdw/data/05_data_marts
-SET [tenant]_datamarts sasfolders_dir/[tenant]/Rigel/data_marts/data

-SET [tenant]_sasdwfmt  /opt/sas/data/sasfolders/Rigel/sasdw/formats

-insert fmtsearch [tenant]_sasdwfmt
```

Here is what this might look like with the placeholder strings replaced with example values:

```
-SET Rigel_sasdw     /opt/sas/data/sasfolders/Rigel/sasdw
-SET Rigel_cd        /opt/sas/data/sasfolders/Rigel/sasdw/data/00_control_data
-SET Rigel_src       /opt/sas/data/sasfolders/Rigel/sasdw/data/01_source_data
-SET Rigel_stg       /opt/sas/data/sasfolders/Rigel/sasdw/data/02_staging
-SET Rigel_dds       /opt/sas/data/sasfolders/Rigel/sasdw/data/03_detail_data_store
-SET Rigel_dms       /opt/sas/data/sasfolders/Rigel/sasdw/data/04_data_marts_staging
-SET Rigel_dm        /opt/sas/data/sasfolders/Rigel/sasdw/data/05_data_marts
-SET Rigel_datamarts /opt/sas/data/sasfolders/Rigel/data_marts/data

-SET Rigel_sasdwfmt  /opt/sas/data/sasfolders/Rigel/sasdw/formats

-insert fmtsearch Rigel_sasdwfmt
```

8 Access Control Templates

In this section, the permissions set in access control templates are given using abbreviations, which are explained in the section titled "Abbreviations of Permissions in Access Control Templates" in the Recommended SAS® 9.4 Security Model Design: Core Principles paper.

For SAS deployments that include Data Integration Studio, create one of each of the following Access Control Templates for each 'tenant' organisation in your SAS deployment. These should be created *in addition* to the ACTs described in the Recommended SAS® 9.4 Security Model Design: Core Principles paper.

Create or Modify: ACT Name	Group	Permissions Granted (G) or Denied (D)
Create: [Tenant] DI Developers ACT	[Tenant] DI Developers	G: RM WM WMM CM R W C D I U S CT DT AT A E
Create: [Tenant] DI Developers CheckIn_CheckOut ACT	[Tenant] DI Developers	G: RM CM R W C D I U S CT DT AT A E
Create: [Tenant] Analysts ACT	[Tenant] Analysts	G: RM WMM R W C D S
Create: [Tenant] Analysts Server ACT	[Tenant] Analysts	G: RM WM
Create: [Tenant] SASBatch ACT	[Tenant] SASBatch	G: RM WM WMM CM R W C D I U S CT DT AT A E

Table 6 – GEL Recommended 'DI' Access Control Templates to include in security model designs for deployments which include SAS Data Integration Studio

9 Apply Access Control Templates

9.1.1 Apply ACTs to Metadata Folders

Table 7 below shows which DI ACTs to apply to which metadata folders.

Location in Folders Tab	DI ACTs to apply
SAS Folders	
[Tenant]	[Tenant] PUBLIC and SASUSERS Denied ACT SAS Administrator Settings [Tenant] SASBatch ACT *If DI Developers should be allowed to work (on jobs and other artefacts) directly in this metadata repository, apply:* [Tenant] DI Developers ACT *But if DI Developers should work in a project repository, and will only check in metadata into this metadata repository, you should instead apply:* [Tenant] DI Developers CheckIn CheckOut ACT *Apply one or the other of the two ACTs above, not both.*
Reason: Securing the DI folder structure is very simple. SAS Administrators, DI Developers and SASBatch need to work with DI 'red' folders in this security model design, and get access to the entire DI folder structure at the top level folder. Other groups do not need access to this folder structure. There is no differentiation between which 'red' folders DI Developers and SASBatch can see in this simple design. All of them see everything. Of course, you may need a more differentiated design for some customer sites.	
Shared Data	
SASApp – OLAP Schema	[Tenant] DI Developers ACT
Reason: In a standard installation, DI Developers (and BI Developers, and anyone else) cannot create cubes because the SASApp - OLAP Schema, which you find in the folder SASApp – OLAP Schema below the folder Shared Data is locked down with an inherited denial of WM, which originates from a system applied ACE, denying WM for PUBLIC on SAS Folders. *If you want to allow DI Developers (or BI Developers or anyone else) to create cubes, you must apply need the DI Developers ACT to this folder. Similarly, if you have other groups such as BI Developers in your design who will create cubes, you must apply your e.g. BI Developers ACT (granting at least WMM, if not WM) to the folder where they will save the cube metadata object.* *Note: Familiarize yourself with other standard folders and their metadata objects to assess whether they need special consideration for selected groups. We cannot sensibly design for every possible DI and BI usage pattern for the standard folders in these recommendations.*	

Table 7 – Metadata folders which should have GEL recommended 'DI' ACTs applied to them

9.1.2 Apply ACTs to the DI ACTs and to SASApp

Table 8 below shows how you should apply ACTs to a range of metadata object types, so that only SAS Administrators can modify the ACTs, and DI developers can deploy jobs for scheduling.

Objects and their location in the Plug-in Tab	**DI ACTs to apply**
SAS Management Console	
Environment Management	
Authorization Manager	
Access Control Templates	
[Tenant] DI Developers ACT **[Tenant] DI Developers CheckIn CheckOut ACT** **[Tenant] Analysts ACT (if it exists)** **[Tenant] Analysts Server ACT (if it exists)** **[Tenant] SASBatch ACT**	*To* each one *of these ACTs, apply:* SAS Administrator Settings [Tenant] SASUSERS Read Only ACT
Reason: Only SAS Administrators should have permission to modify ACTs in your security model. Everyone (i.e. SASUSERS) has RM on every ACT: anyone can see the design of any ACTs, but read-only.	
SAS Management Console	
Environment Management	
Server Manager	
SASApp	[Tenant] DI Developers ACT [Tenant] SASBatch ACT *…in addition to other ACTs recommended in the other papers in this series*
Reason: DI Developers need Write Metadata access to SASApp in order to create/modify data libraries, save stored processes in source code repositories, deploy jobs in deployment directories etc. SASBatch accounts need Write Metadata permission on SASApp in order to deploy jobs.	
SAS Management Console	
System	
Secured Libraries	[Tenant] DI Developers ACT
Reason: Allow DI Developers full control of metadata-bound libraries and content. *Note: In a standard installation, SAS Administrator Settings has already been applied to /System. The WMM permission on /System is sufficient for SAS Administrators to create metadata-bound libraries. The WMM in combination with WM on the secured library object allows administrators to remove libraries. If your design requires that*	

	SAS Administrators work at the secured table level and not just at the secured library level, then you will need to supply SAS Administrator Settings with the same permissions as in the SMC DI Developers ACT. *SASUSERS inherit RM to /System/Secured Libraries and content. In order to allow other groups to access metadata-bound tables, it is necessary to apply an appropriate ACT. Our practice is to apply them to the metadata-bound library object below /Secured Libraries because we do not maintain folders here. For example, to allow SASBatch access to a particular metadata-bound library, apply the [Tenant] SASBatch ACT directly to the that metadata-bound library object.*

Table 8 – Metadata objects which should have GEL recommended 'DI' ACTs applied to them

10 Filesystem folder permissions and ownership

10.1.1 Windows Security

On Windows, three levels of permission are used in security windows directories:

Abbreviation	Permission	Controls
F	Full Control	Users with this permission can do anything with the object
C	Change	Read-write. Users with Change permission can edit the object, but cannot change *permissions* on the object.
R	Read	Read only.

Secure the DI-specific directories on the filesystem like this:

Folder	Windows Security Settings
…\sasfolders\[Tenant project or organization]	(Set on "This folder, subfolders and files") SYSTEM: F Administrators: F SAS [Tenant] [Level] [Project] DI Developers: C SAS [Tenant] [Level] [Project] Batch: C

10.1.2 Unix and Linux Security

Unix has basic posix filesystem security capabilities, and file mounts can optionally be configured to use a more advanced capability called Access Control Lists (ACLs). These recommendations use the basic posix capabilities. More advanced filesystem security designs are only possible with ACLs.

The general principle for filesystem access in this design is that if a user gets access to a directory, they have write access to it.

Only one group can be set as the owner of a directory, which means that we create groups differently in Unix than we do in Windows. The following are examples only: you will need to design these as required for your specific needs.

Unix Group	Members
sas	SAS Installation User (sas or sasinst) SAS General Servers user (sassrv)
sasproj	SAS DI Developers Users SAS Batch Users
sasana	SAS DI Developers Users

	SAS Batch Users SAS Analyst Users
sasanabi	SAS DI Developers Users SAS Batch Users SAS Analyst Users SAS BI Developers*
sass2 (short for 'SAS subject area 2')	SAS DI Developers Users SAS Batch Users SAS SubjectArea2*

Then secure the sasfolders directory on the filesystem like this:

Folder	**Unix Security Settings**	
	Owner (user:group)	Permission pattern
…/sasfolders	**sasbatch**[7]:sas	2775
…/sasfolders/[Tenant project or organization]	sasbatch:sasproj	2770

[7] Note that the sasfolders directory on the Unix or Linux filesystem was defined in the GEL Recommended SAS 9.4 Security Model: Core Principles document as being user-owned by 'sas', whereas here we say it should be owned by 'sasbatch'. This inconsistency is **deliberate**. For solutions which do not include SAS Data Integration Studio, the 'core' document still applies, and we recommend the sasfolders folder should exist, and should be owned by the SAS Installation User (sas or sasinst). However, when your SAS deployment does include SAS Data Integration Studio, the sasfolders folder is better owned by the user sasbatch, as recommended here, since the user sasbatch will be more likely to be used to work with the folder and its content.

SAS INSTITUTE INC. WORLD HEADQUARTERS SAS CAMPUS DRIVE CARY, NC 27513
TEL: 919 677 8000 FAX: 919 677 4444 U.S. SALES: 800 727 0025 WWW.SAS.COM