Paper 2501-2015

Best Practice for Creation and Maintenance of a SAS® Infrastructure

Paul Thomas, ASUP Ltd.

ABSTRACT

The advantage of using metadata to control and maintain data and access to data on databases, marts and warehouses is well understood. SAS Metadata Server®, SAS Management Console® and other associated products provide a facility for hosting this metadata, however the metadata itself also needs managing.

This paper, aimed at individuals new to SAS administration or who are looking for rules to help tidy up an existing installation, provides some best practice methods and tips to ensure well controlled and managed metadata. This includes suggestions for folder structures, access rights, environments and promotion, SAS DI Studio® jobs and flows.

INTRODUCTION

SAS® is a great software product because it provides:

- Unparalleled ability to manipulate data, with the Data Step and numerous Procedures
- Many types of products for a wide range of tasks, e.g. Data quality, ETL, Forecasting, Business Intelligence, etc.

Unfortunately, SAS® is not so great to administrate for similar reasons:

- Unparalleled ability of users to manipulate data, meaning you lose control of where the data is, and what is represents.
- Many types of products, these can be very hard to co-ordinate.

SAS Metadata Server®, SAS Management Console® and other associated products aim to address these risks by creating a layer of metadata that records what data is where, and if you are using SAS DI Studio® how these data items are associated with each other. On top of this, it contains a structure for authorisation that contains information about who is allowed to see each individual data item. As the data on the estate that is being managed grows, the metadata that manages that estate can become unwieldy.

There are 169 types of metadata objects in SAS® version 9.4, however the ones this paper will focus on are:

- Directories
- Users and Groups
- Jobs and Schedules
- Libnames
- Data sets

There is also a short section about the movement of metadata between different levels on an estate.

This paper aims to promote some best practice solutions for managing metadata and is aimed at new administrators. It assumes that the reader has access to a correctly set up and fully functioning SAS Metadata Server®, has knowledge of the basic principles of how metadata works, and is able to comfortably use the provided tools to access that metadata.

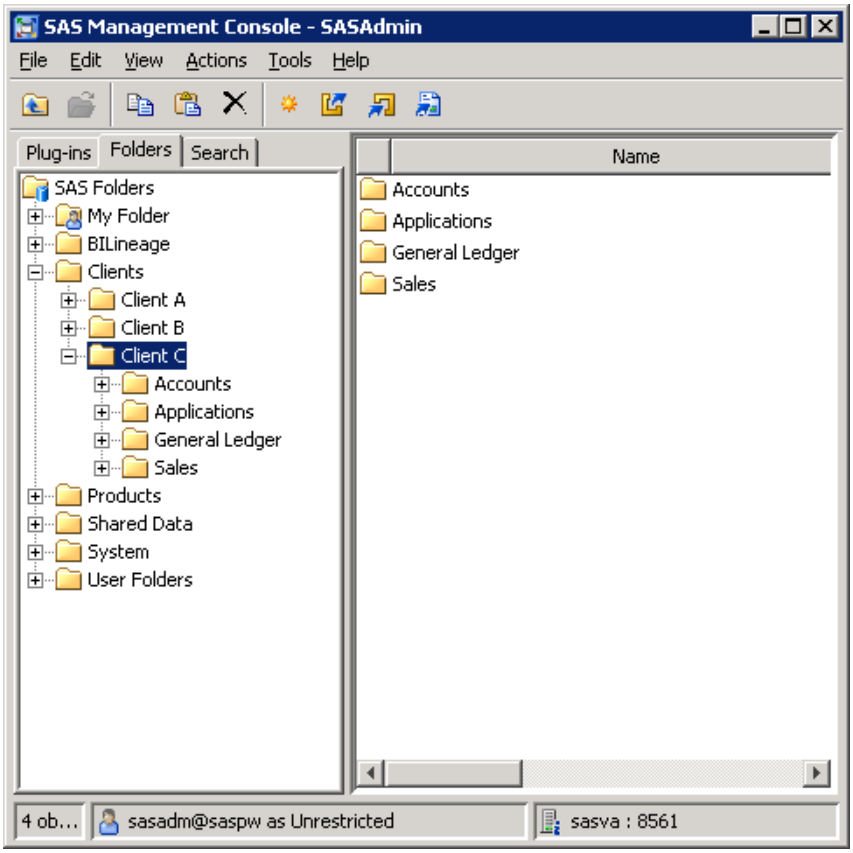
METADATA DIRECTORIES

Metadata directories behave in a very similar way to an operating system directory structure, i.e. directories can contain both other directories and other metadata objects, with the added ability to enforce a metadata permission system.

As such, directories should be organised in a considered, predefined and obvious manner. It is recommended that a template is chosen and published to the user community that accurately reflects the working practices of the business where SAS® estate is deployed. This template should also seek to avoid redundancy.

Where possible it is also recommended that the metadata folder structure and operation system folders are synchronised as much as possible. This could lead to folders being organised by project, data source, client, etc. or some combination of these. One aim of the structure should be to minimise clicks to find content.

Also of consideration, the structure should be designed to minimise the touch points for application of access rights. This consideration often conflicts with the previous aim to make content easily found, so some trade-off is required between splitting the folder structure between projects or types or users within that project.



Display 1. Example client/project based folder structure

Finally, whatever template is chosen for the structure, it should be actively and aggressively maintained.

USERS AND GROUPS

Once you have created your folder structure, access rights will need to be assigned to the previously mentioned touch points in the folder structure.

Properly assigned access rights will allow users of the estate to see, create or edit the data and metadata they need to perform their role in the appropriate manner; and stop them seeing, creating or editing data and metadata that they should not be able to.

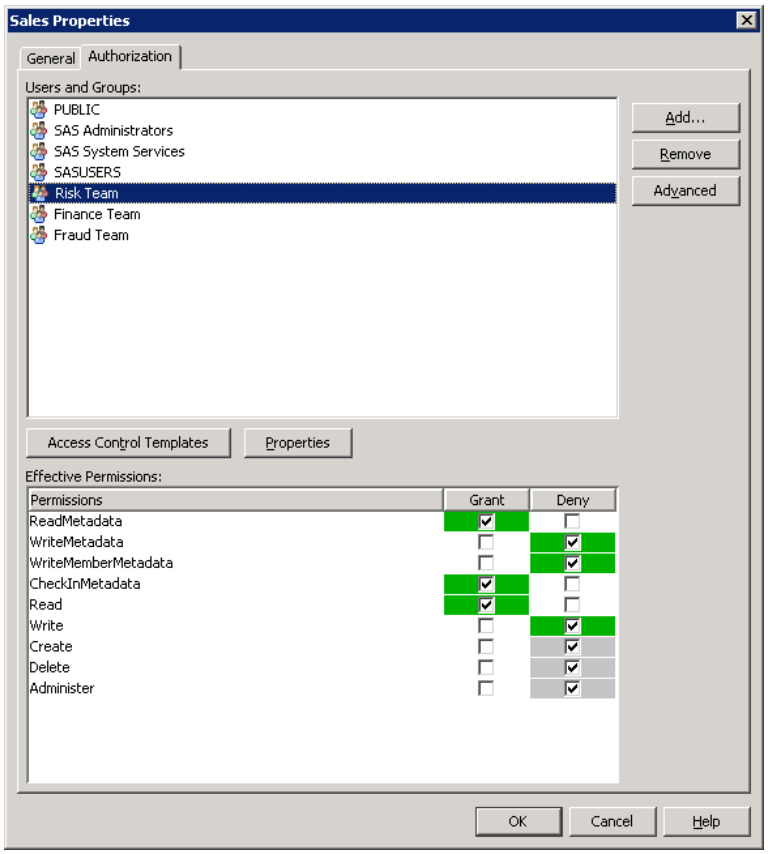
This is best achieved by following these rules:

- Only ever assign access rights to directories.

The purpose of this it to keep it simple. There is an exception to this - if you are running multiple application servers these usually lie outside the directory structure and authorisation will need to be applied to these individually.

- Never assign access rights to an individual user, always put users into groups then assign access rights to the group.

This can mean that groups need to be created for single users or purposes. This is quite acceptable, and preferable to directly assigning the access rights to that user – it is reasonable to assume that the user performing that role will change at some point in the future.



Display 2. Example of applied access rights

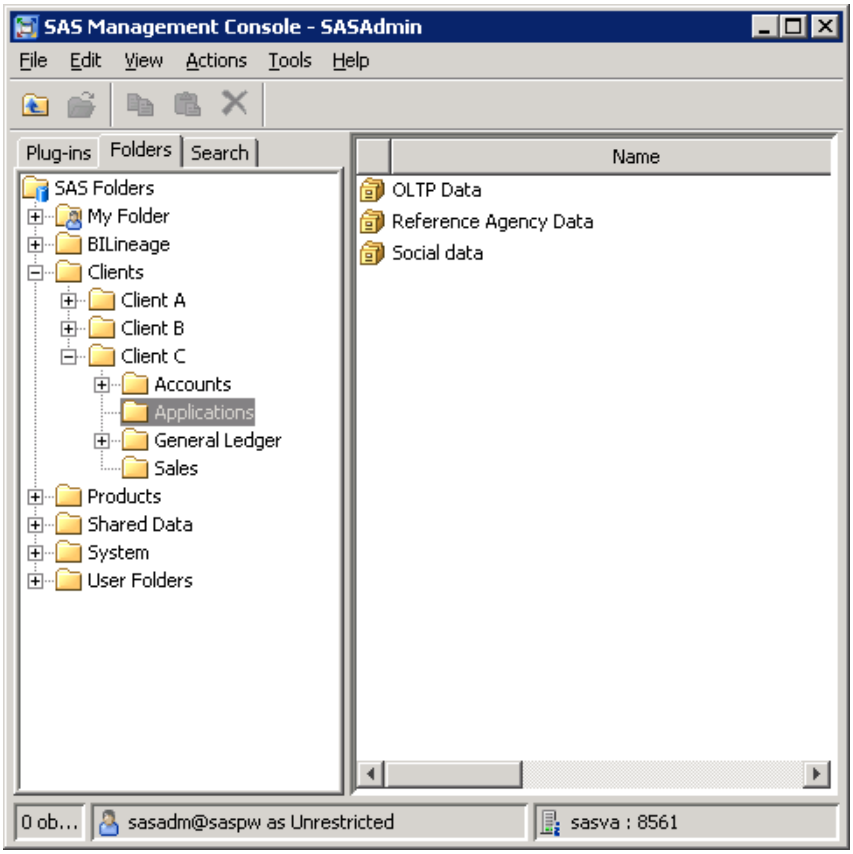
Also, there are often circumstances when it is preferable to have more than one layer of grouping, and to put groups within groups. One example of this is when a group is used to contain a team, and that team need access to various projects – each of which has its own group too.

ACCESS CONTROL TEMPLATES

Access control templates are pieces of metadata that enforce an authorisation pattern on a metadata object. These can be very useful when the same access rights will be repeated on different directories across the directory system, and they make it much easier to manage the problem alluded to earlier in the conflict between having few authorisation touch points and an intuitive metadata directory structure.

LIBRARIES

Libraries are often the starting point for most users on an estate, i.e. they control what data can and cannot be seen. As such, getting the authorisation right for libraries is very important – users should be able to see the libraries they require, and none of the ones that they don't, how well this is accomplished is often a good indicator of how well the estate is managed. Placing libraries (and their associated data sets) in the correct position in your metadata directory structure will facilitate this view.



Display 3. Example of library storage

In the example above, the correct authorisation profile would have been applied to the Applications directory, and the libraries inside this folder would inherit that authorisation profile.

Ideally, to get the full benefit of using metadata, all libraries should be assigned by the available metadata as metadata libraries, using the authorisation profile available to the user who is signing onto the SAS Metadata Server®. However, there are some circumstances where this is not practical, e.g. where users need to be able to store their own data; and some circumstances where this becomes more complex, e.g. for accessing external databases that require a user name/password combination as part of the LIBNAME statement.

EXTERNAL DATABASE LIBRARIES

When a user name/password combination is required to access an external data source, the AuthDomain property of the library is used to find the correct credentials to use in the libname statement. Ideally, each user should have their own user name/password combination, but sometimes this is not the case are shared credentials are used. If shared credentials are required, create a group specifically for them and add other groups into it – in a similar way to metadata directories.

PRE-ASSIGNED LIBRARIES

These are useful when users need to be able to create and write data into an area, but it is preferable to use metadata assigned libraries where possible, and avoid pre-assigned libraries. Making a library pre-assigned has the effect or removing any metadata authorisation for this library, giving all users who can see it read/write access to the data – subject to operating system permissions. There are circumstances where this is desirable, as users will often need to create and store permanent data, but it should definitely be avoided for production data.

METADATA DATA BOUND LIBRARIES

Unless you have stopped your users from being able to submit SAS code, all administrators will be aware that it is possible to submit a LIBNAME statement and get access to any SAS® data that the user can see on the operating system.

From SAS® version 9.4M2 onwards, there is a way to stop this happening using PROC AUTHLIB. Data sets can be forced to perform a metadata authorisation check before allowing data to be read.

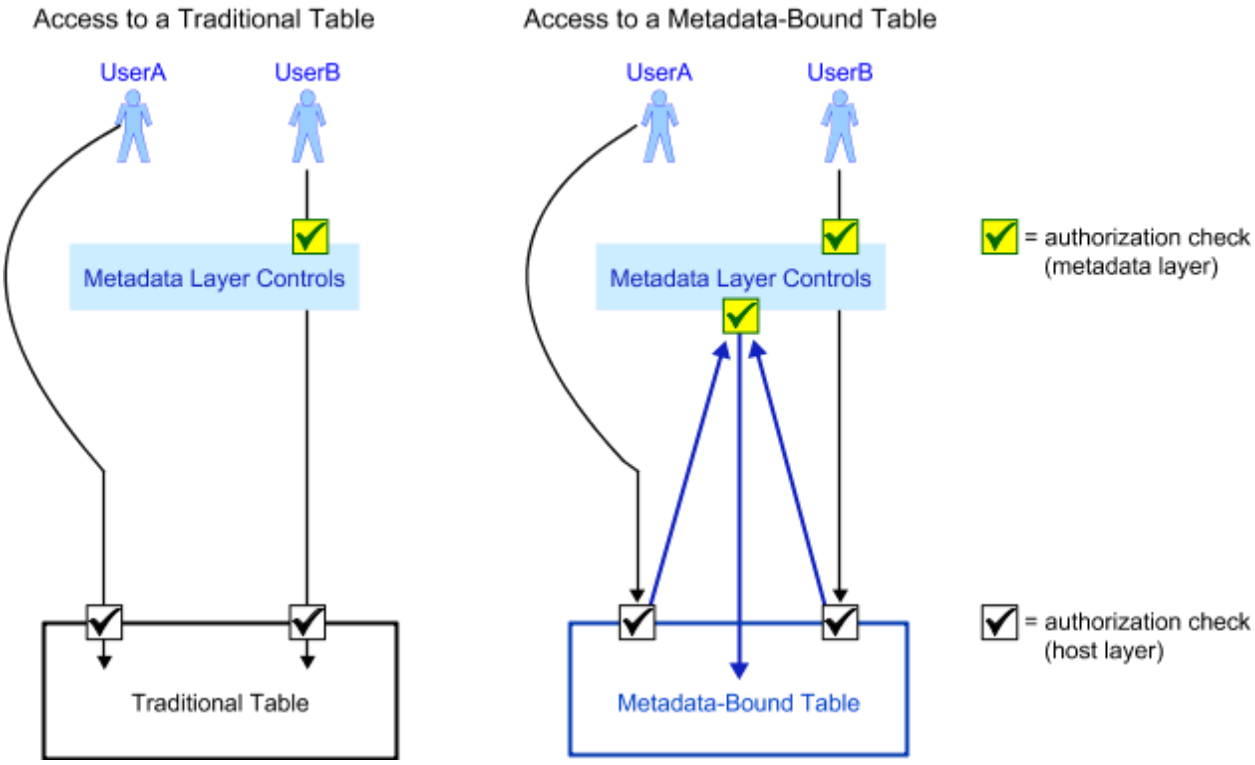


Figure 1. Authorization Checks (by Data Type and Access Method)

DATA SETS

Well managed data sets on a SAS® estate can save administrators much time and effort, especially on estates where the use of SAS Data Integration Studio® is prevalent. In general, some rules that should be followed to make life easier for both developers and analysts are:

- Make data set names unique across the metadata instance, even across different libraries
- Never have more than one metadata representation of a physical data set
- Ideally keep metadata and physical names the same for each data set

The three rules above make promotion across environments much less susceptible to error, and the management of the data set metadata simpler.

- Use the same variable names on different table where they join together, avoid using the same variable names where there should be no join

This makes using SQL joins in SAS Enterprise Guide® much simpler. This might not seem like a big deal, but it does make life significantly better for analysts how use the tool a lot.

- Push back on data sets that have not been through a data modelling process.

Data modelling is not a new phenomenon, but as SAS® has the ability to handle virtually any data that is thrown at it, SAS® programmers often do not consider thinking about how their data is structured. However, as the SAS® tool set in use becomes more advanced, and the data you look at becomes bigger, this becomes more important.

- Try to avoid applying different access rights within a library, e.g. having some data sets as read only and some as read-write.

It is even possible to assign individual authorisation patterns to columns within data sets, however all these scenarios become difficult to track. It is preferable to consider libraries of containers of not only different types of data, but different uses of data.

DATA INTEGRATION JOBS

If your estate is using SAS Data Integration Studio®, it is likely that this part of the estate will be the largest contributor of metadata. It is obviously important that this data is put into a sensible directory structure, but there are also rules that should be followed to ensure that the metadata the jobs include is properly structured too. Administrators should have an overview of the DI jobs on their estate, and should be able to enforce published rules about minimum expectations for job development.

- Keep all DI jobs as small as possible.

Large DI jobs are a problem – they take a long time to open, they become harder to understand and they are more likely to become corrupt. Ideally, each permanent data set should only be either an inputs or output to the job, and there should only be one stream of connected nodes in a job. If this is not the case then split the job up.

- Make sure all jobs are error aware and re-runnable.

Force errors to happen if the job is not processing ideal data – in a production environment, warnings or notes that indicate something needs to be checked are often not enough. Also, when errors do happen (or even if they do not happen), the job should be engineered in such a way that re-running it will give the same results, or correct results if the cause of the error has now been fixed.

- There are lots of spaces for descriptions and comments to be put in, make sure that they have been used.

It is a very good habit to fill in description fields, even when the comments are trivial.

- Have a system for controlling the date (and time) that the jobs is supposed to be running.

It is very rare that a system will work in entirely real time, and even then, that the associated development or testing will also be in real time. So, avoid using functions like Today() and instead use macro variables to represent date and time values. A controlling macro called in the job pre-code can then be used to supply these macro variables. This controlling macro would typically be fed by a reference data set.

- Don't have jobs within jobs, and avoid loops in jobs.

Having a job inside another job should be unnecessary if you have a mechanism for scheduling. Having a job within another job also raises the possibility of a partial job failure, which can be unnecessarily tricky to sort out. Similarly, it is best practice to expand loops into their individual components so that any failures can be dealt with easily. There are circumstances on SAS/Grid®

MULTIPLE ENVIRONMENTS

Most SAS® estates will contain more than one environment or level to enable development, testing and production to happen in a mutually exclusive manner. Ideally administrators will also have another separate sandbox environment in which they can perform/test upgrades, metadata changes or any other tasks they might desire without affecting the general user population.

Where this is the case, there will be a requirement to move metadata between these levels as a promotion, i.e. an export from one environment and import to another.

SAS Data Integration Studio® has had integration with open source tools CVS and SVN for source code management, and more recently SAS Enterprise Guide® has been integrated with GIT. Using these tools to manage packages is definitely a best practice for promotions as is gives you a detailed understanding of what has been deployed in each environment.

If the skills are available inside the organisation, it is also worth taking the time to utilise these tools to store deployed SAS code in the production environment, to facilitate comparing different versions of the

code that has been deployed. This is not usually effective between environments as differing metadata ids will be flagged up as code changes.

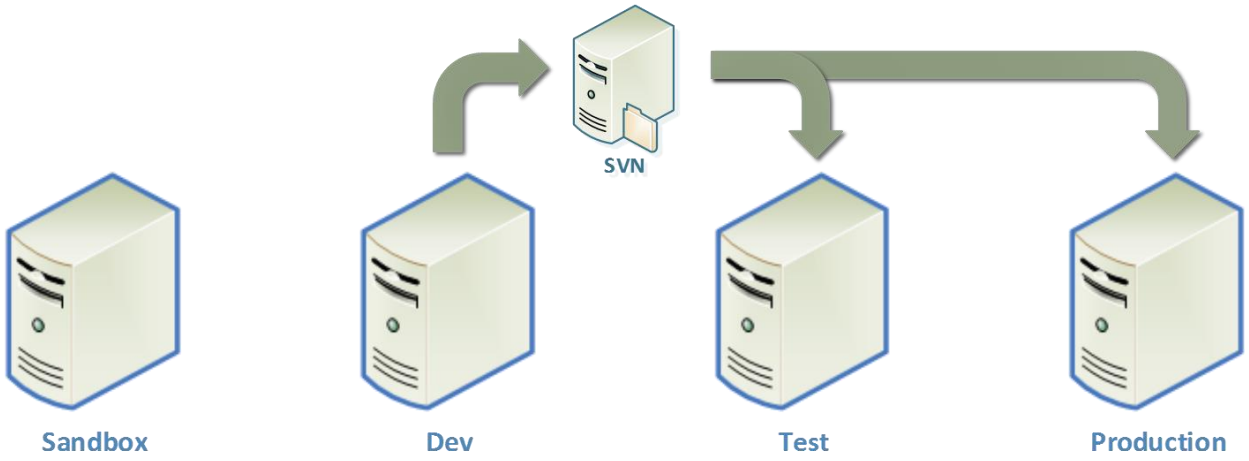


Figure 2. Promotion with SVN

PROMOTIONS

When you are importing metadata from a package as part of a promotion, it will be necessary to match the objects you are importing inside the package against objects that already exist in the target level. For this reason it is important to ensure that the source and target levels are as close as possible and certainly use the same naming conventions.

Metadata names have to match between levels for promotions to work as smoothly as possible, however the physical data behind that environment may differ, and will be specific to that particular level.

This could apply to file names, server names, and environment variables specific to each level. Where you can't use metadata, set automatic macro variables or have data that allows code to differentiate between levels.

Promotion packages should not need changing between levels, i.e. the package that is promoted into the test level should be the same package that is promoted into the production level.

SCHEDULED CODE & LOGS

The purpose of all the metadata is eventually to produce some Base SAS® code that runs and produces a log. This also needs monitoring and managing, although this more of an operating system management task than a metadata one.

For this, it is preferable to have one area for code, although this may be sub-divided into functionality directories, deployed code should definitely not be fragmented around the operating system. The areas where code is deployed should be kept clean, and as jobs are decommissioned, care should be taken to remove redundant code.

Similarly, have as few directories as possible for job logs. Archive log files frequently to keep directory listings small.

SCHEDULE MANAGEMENT

As the number of jobs increases on your estate, it becomes increasingly important to be able to track your jobs and how well they are behaving. There is no guarantee that what was once a perfectly viable piece of code will always behave well, so it is critical to be able to monitor how each job is performing every time it runs.

If you are using Platform LSF for scheduling, don't be afraid to change the default Platform (LSF) parameters. These are set to run one job per processor core. This isn't always ideal, especially if your jobs get data from a database on a different server, and in this circumstance jobs can often be waiting whilst data processing in happening elsewhere.

CONCLUSIONS

This paper provides an insight into some of the issues that will present themselves when managing metadata on a SAS® estate, and provides some ideas for how to approach them in a best practice manner.

The key learning points are:

- Plan everything before going ahead, even simple tasks need standards and will become more complex over time
- Have an obvious metadata directory system, preferably one that mirrors the operations system directories where appropriate
- Do not let anyone be an individual, use groups instead
- Be strict with the standards you create, monitor them, and be prepared to move/delete where these standards are not adhered to.
- Promote with care, promotions should be easy, if they are not then you are doing something wrong.
- Work hard at being lazy!

REFERENCES

"Authorization Checks (by Data Type and Access Method)" "Authorization Model for Metadata-Bound Tables" Accessed on 26th March 2015, available at http://support.sas.com/documentation/cdl/en/seclibag/66930/HTML/default/viewer.htm#n0mr3uqudlccdcn173a7b2khsu1z.htm.

RECOMMENDED READING

- *SAS(R) 9.4 Intelligence Platform: System Administration Guide, Third Edition*
- *SAS(R) 9.4 Metadata Model: Reference*
- *SAS(R) 9.4 Guide to Metadata-Bound Libraries, Second Edition*

CONTACT INFORMATION

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

Paul Thomas
ASUP Limited
paul@asup.co.uk
http://sascode.asup.uk