An approach to sheet level security

A frequent question I get is can we do sheet level security and the answer is yes but with a but.....

My view is that showing and hiding sheets is little connected to security and is more about adapting the experience of one app to different roles. To achieve this there are different approached where one is to do sheet level security others are

- Creating multiple apps one for each developer
- Streamlining the app and use section access if the same visualisations is needed but with different data.

I would only recommend that the approach i describe in this post is used for a limited number of apps as it will quickly be hard to manage and understand who have access to what.

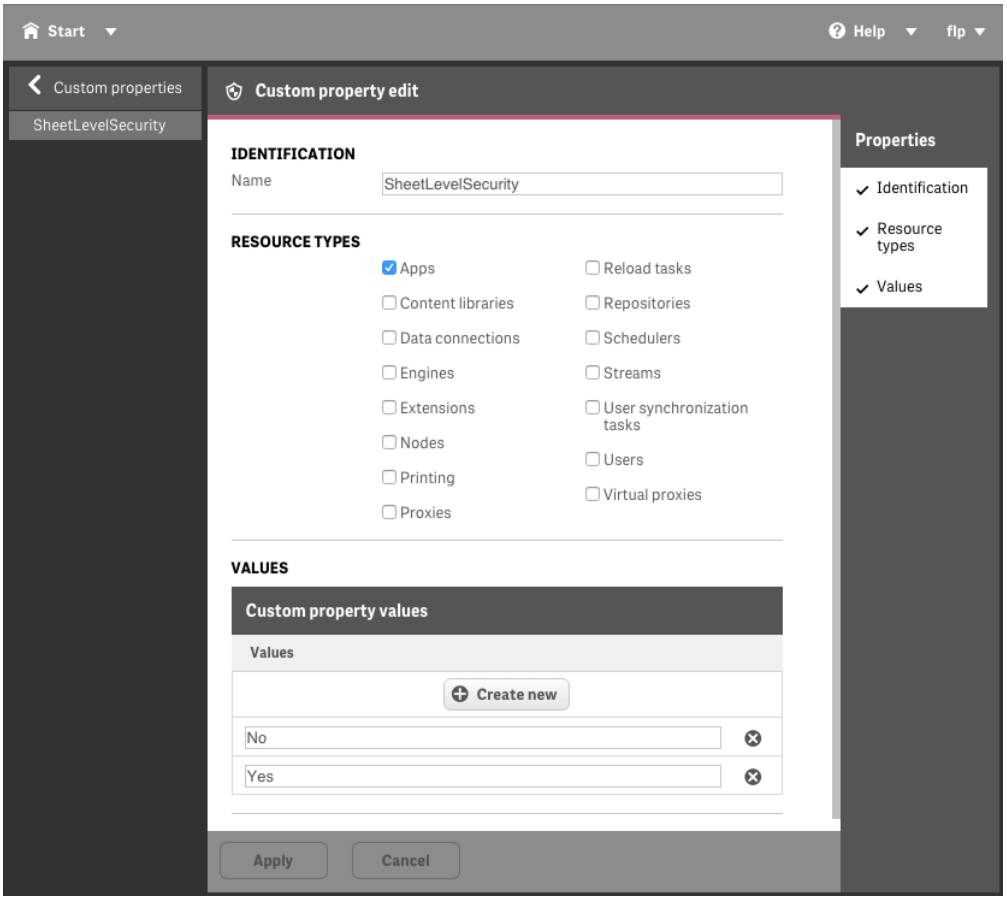
So how can it be achieved?

I will outline a little different approach here that I believe can help with manageability of a solution.

The basic idea with the approach is that we want to configure two things:

1. Which apps that need sheet level security turned on
2. Who should have access to the sheets

To configure which apps should use sheet security we will use a custom property called @SheetLevelSecurity



For the apps where we want to apply sheet level security we will set this custom property to Yes.

We will then use this custom property to in two rules.

Name	Stream
Resource filter	App*

Condition	(resource.resourcetype = "App" and resource.stream.HasPrivilege("read")) or ((resource.resourcetype = "App.Object" and resource.published ="true" **and resource.app.@SheetLevelSecurity!="Yes"**) and resource.app.stream.HasPrivilege("read"))
Actions	Read

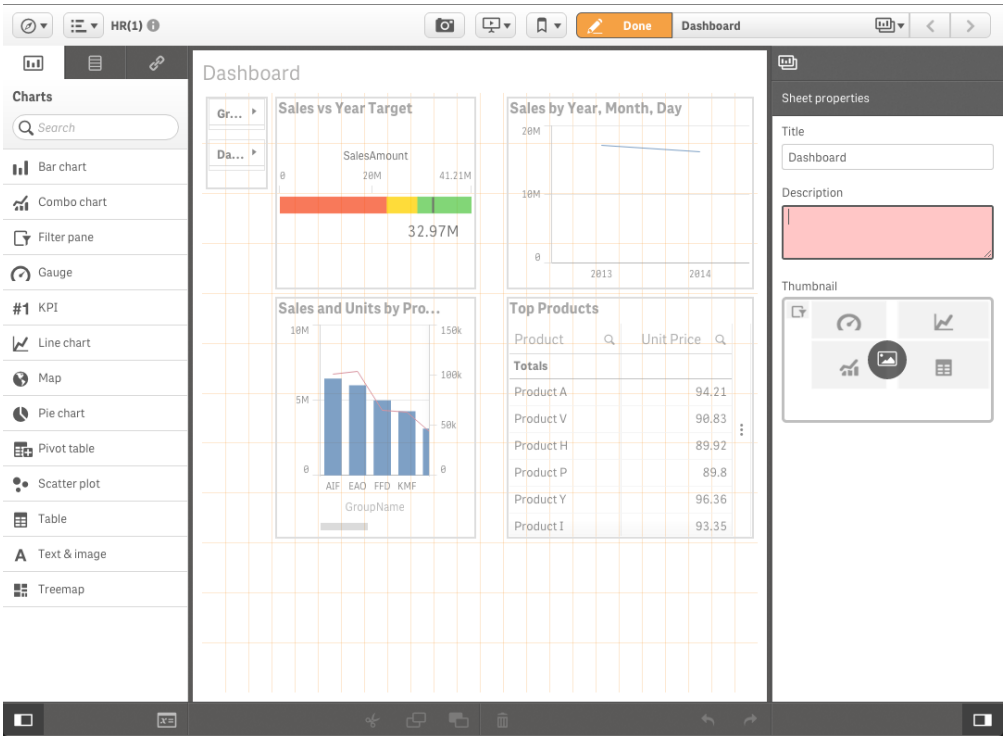
What this rule say is that for apps where we want to use sheet level security we should not automatically give access to the sheets just because we have access to the stream.

The second rule is about how we decide who should get access

Name	SheetLevelSecurity
Resource filter	App.Object_*
Condition	((resource.objectType="sheet" and resource.published ="true" and resource.app.@SheetLevelSecurity="Yes" and user.group=resource.description) or (resource.published ="true" and resource.objectType!="sheet")) and resource.app.stream.Has Privilege("read")
Actions	Read

What this rule describes is that the user will get access to published objects and for sheets they will only see sheets that contains a group in the description field of the sheet that they are member off.

If we now have an app with multiple sheets by adding a single group to each description of a sheet this group would get access to the sheet.



Summary

This approach shows how sheet level security can be achieved with reasonable manageability. Future enhancements would be to allow for custom properties on sheets which would allow centralised management in the QMC.

SheetLevelSecurity720p.mov