Using Redgate SQLDoc with Data Catalog

One of the challenges that face any company offering products that collate and present information is how best their product should display that information to the intended audience. This challenge is particularly acute in a Data Catalog for two main reasons.

* The intended audience is [diverse with equally diverse needs](https://documentation.red-gate.com/sql-data-catalog/taxonomy/further-advice-on-defining-a-taxonomy).
* The corporate data asset can be rich and varied

Three approaches that can be taken to address the presentation challenge are shown below

* Presentation is handled by the product itself
* Data is exported from the product in a form that can be used in other facilities.
* Official documentation exists to support the use of business intelligence tools on the data catalog repository.

These options should not be considered mutually exclusive. Reject the tyranny of OR in favor of the genius of AND.

For example, [Redgate Data Catalog](https://documentation.red-gate.com/sql-data-catalog) provides both information dashboards and the facility to export the catalog contents in CSV format.

Although an information tool should always provide some visualizations it is not practical or even possible to provide for every need.

For this reason, explicit support for BI and reporting tools using the database underpinning a Data Catalog is necessary. To make this viable that database has to be well documented. For this [Redgate SQLDoc](https://www.red-gate.com/products/sql-development/sql-doc/index) is an excellent complement for the Redgate Data Catalog product.

SQLDoc as a metadata authoring tool

In an ideal world the supply of meaningful descriptions for database objects should be part of standard development practice when objects are created or updated. If you cannot explain the object you are creating/updating to the relevant audience then your understanding of why you are creating/updating it is insufficient. Are you developing software with a full understanding of requirements?

A software engineer’s enthusiasm for having high quality documentation available is diametrically opposed to their level of enthusiasm for being the person who provides it. SQL Server also has an awkward syntax for attaching descriptions to objects.

Compare SQL Server’s [sp\_addextendedproperty](https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/sp-addextendedproperty-transact-sql?view=sql-server-ver15) and [sp\_updateextendedproperty](https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/sp-updateextendedproperty-transact-sql?view=sql-server-ver15) to that provided by platforms such as PostGres, AWS Redshift or Vertica that have a simple syntax

COMMENT ON {object} IS ‘{meaningful description}’;

Reluctance combined with complexity means that documentation is less common that it should be.

This is where the use of Redgate SQLDoc as an authoring tool becomes invaluable. Data stewards, business analysts and data subject matter experts can carry out the necessary software archeology to attach meaningful descriptions to database objects. Such descriptions need to backfill into the source code repository so that they are not lost if objects are dropped and recreated.

It is here that we must remember that documentation should be written for the readers of that documentation. It is not a tick box exercise to close a JIRA ticket or an item to be descoped so that development time gains some contingency padding. A system becomes “legacy” when people avoid changing it for fear of consequences that are unpredictable in both scope and nature. Such unpredictability can be driven by poor coding hygiene and lack of easily accessible information about the system.

Software benefits from [clean coding](https://www.amazon.co.uk/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882/ref=sr_1_1?dchild=1&keywords=clean+code&qid=1604869341&sr=8-1) practices such as ruthless refactoring. Documentation is also an artefact that benefits from refactoring and continuous improvement.

Extending the capabilities of Redgate SQLDoc

Redgate SQLDoc can produce documentation in a number of formats

* MS Word
* CHM – Compiled HTML Help file
* PDF
* HTML
* Git Markdown

We are going to focus on HTML and Git Markdown as web pages and git compatible source control systems are almost ubiquitous.

To make documentation more effective there are techniques we can use to emphasize the relevant information.

* The use of bulleted/ numbered lists
* The use of hyperlinks

Redgate SQLDoc already creates hyperlinks between the database objects it is documenting. It would be useful to be able to hyperlink to content external to the database documentation. For example, if we were to use Redgate SQLDoc to document the Redgate Data Catalog it would be useful to hyperlink to the product documentation.

It is here that we begin to find challenges.

If we attempt to put HTML into object descriptions the product will html encode it. A hyperlink such as the following

<a href=”https://www.red-gate.com/products/dba/sql-data-catalog/”>Data catalog</a>

Will be converted to

&lt;a href=<https://www.red-gate.com/products/dba/sql-data-catalog/>&lt;Data catalog&lt;/a&gt;

If we were to use markdown format instead then we could use the following.

[Data catalog]( <https://www.red-gate.com/products/dba/sql-data-catalog/>)

This would render correctly when Redgate SQLDoc publishes markdown files but not if we asked for html files.

Similarly if we wished to use bullet points then having a description for the dbo.AssetTypes table of the Redgate Data Catalog DB as follows would render in a barely acceptable form in HTML but would break the markdown table structure completely.

Describes the type of object within the data catalog that can be categorized  
\* SQL Server Instance  
\* Database  
\* Schema  
\* Table  
\* Column

A final challenge is the default filenames used when you access the documentation for a directory differs depending on whether you use a web servers or git repository.

* Webservers use index.html
* Git repositories use readme.md

All these problems are solvable with a little code.

STEP ONE: Generating the initial documentation set

The foundations for the first step were laid in the article [Getting started with SQLDoc and Powershell](https://www.red-gate.com/hub/product-learning/sql-doc/getting-started-with-sql-doc-and-powershell) by Phil Factor.

The techniques in the article were used to create RunSQLDoc.ps1 file as shown below

Set-Alias sqldoc 'C:\Program Files (x86)\Red Gate\SQL Doc 5\SQLDoc.exe' -Scope Script

# base\_path is the folder containing this script

$base\_path = Split-Path -Parent $PSCommandPath

$documentation\_path = "$base\_path\SQLDoc\DADSPCJUNE2014\_documentation"

# Make sure we start with a clean folder

Get-ChildItem -Path $documentation\_path -Recurse|Remove-Item -Force -Recurse

# Generate both md and html documentation

sqldoc /server:DADSPCJUNE2014 /database:Redgate\_SqlDataCatalog /project:RedgateDataCatalog.sqldoc /fileType:md

sqldoc /server:DADSPCJUNE2014 /database:Redgate\_SqlDataCatalog /project:RedgateDataCatalog.sqldoc /fileType:html

The Get-ChildItem|Remove-Item construct is used instead of the SQLDoc /force option because we are going to rename the index.md files to readme.md. Doing so ensures that the documentation generation has the same starting state every time.

The sqldoc alias is called twice so that both a markdown and html documentation set is generated.

STEP TWO: index.md to readme.md

If the index.md files are renamed to readme.md then any links in the markdown documentation set that point to the original index.md files must also be updated. The approach taken was

* Create a ScriptHelpers subfolder
* Create an index\_to\_readme.ps1 file within the ScriptHelpers folder as follows

[cmdletbinding()]

param([string]$Foldername)

# Git repos automatically render readme.md files so change the index.md files to readme.md files

Get-ChildItem -Path $Foldername -Recurse -File -Filter "index.md"|Rename-Item -Force -NewName "readme.md"

# As all references point to index.md change ALL references across all md files to them to point to readme.md

Get-ChildItem -Path $Foldername -Recurse -File -Filter "\*.md"|ForEach-Object{

(Get-Content -Path $\_.FullName).Replace("index.md","readme.md")|Set-Content -Path $\_.FullName

}

A call to this script can be added to the original RunSQLDoc.ps1 file described in STEP ONE.

& ".\ScriptHelpers\index\_to\_readme.ps1" "$documentation\_path"

When RunSQLDoc.ps1 is run the output will be as follows

* HTML documentation set
* Markdown documentation set
* Default readme.md files
* Documentation breadcrumb trails and links correctly using readme.md as targets

STEP THREE: Rendering links embedded in descriptions