Data Handover

Guidelines for sustainable and scalable development  
By Nick Craft  
1/24/2023

# Purpose:

Throughout this document, term “solution” is used to broadly refer to a wide range of items that are created by developers and analysts such as analytical reports, data models, Excel documents, software tools, programing code chunks and other items that are designed to solve a specific problem or a achieve a specific goal. For the most part though, this article refers to data models and analytics used in development of interactive reporting tools.

This document aims to provide practical and actionable advice that readers can use to improve their solution development and solution management processes. The focus is on preparing analytical reports for sharing with others as well as note-keeping for the original developer. As strategies and overall methodology are presented, an explanation of the benefit to the author and the receiver of the document will also be included.

# Author’s Note:

Odds are, as a developer / analyst, at some point, you will need to share the things you make. If at the very least, you are required to explain how something works and the reasoning behind your decisions during presentation, these techniques I outline will be beneficial to you. More than likely, you will need to prepare something for handover so that others can iterate upon it or even just maintain it. Also, unless your goal is to lose your sanity, it is very likely that you will need to understand your own thinking when picking up a project after leaving it sitting on the shelf for a while.

Much of what is discussed is related to documentation, but there are other important considerations to improving understandability which I will cover. These handover components should be integrated into the development process - not put off until the end of development. Being cognizant of those who will inherit the project (including yourself) is an essential part of meeting quality standards. By not following these guidelines, the risk to a shared solution is that it won't be supported correctly or, even worse, it won't be adopted at all. In essence, the goal is to minimize the effort required to understand the things you make, guard against adulteration, increase the usefulness, and elevate the overall quality.

Consider the following data maxim:

*“Slow is smooth and smooth is fast”*

The phrase "slow is smooth, smooth is fast" originates from the SEAL teams but the purpose behind it can be applied to all situations. There are many situations when we want to react quickly and start moving, but in your efforts to move so quickly, you end up slowing down the operation. Investing the time to incorporate handover components into development will ultimately speed up final delivery by reducing opportunities for error or miscommunication.

# Table of Contents

1. Include documentation for the data model and report.

2. Set up version control

3. Assess the need for stability and push transformations upstream when appropriate

4. Create organized model diagrams

5. Develop unit tests and document test

6. Format code for readability

7. Comment complex code

8. Eliminate unnecessary data

9. Check data types & decimal precision

10. Organize expressions & fields in display folders

11. Hide non-reporting fields

12. Security: manage access and permissions

# 1. Include documentation for the data model and report

Having documentation completed upfront is optimal but also expected to be an ongoing part of development. Other outlined steps in reporting development include additions to the initial documentation. For consistency, every data set / report should include the same documentation structure, but some projects will start off with less direction than others.

Consider several possible audiences and try to strike a balance when choosing language that gives the necessary level of depth but could also be understood by non-developers. Documenting and commenting complex code is an important step that I will address elsewhere in this article. For this portion of documentation, end users are not necessarily expected to read through the entire contents, but those who inherent what you created with a lower skill level should still be able to glean a decent understanding of what you were trying to achieve and where to investigate if they have further questions.

Here are some guidelines on documentation that could be included:

## Report purpose / Business context:

This includes a narrative description and contextual information. It should provide background on the business need that gave direction to the report development. It can include the scope of the project for reference, but it may not be necessary to detail all strategic elements of the business perspective - a general overview should be sufficient while more detailed business outlines are established elsewhere. You can consider including information on how the report development process was managed and coordinated, including any tools or methodologies that were used to support collaboration and communication. By including this kind of information, you can help readers understand the broader context of the report and how it fits into the overall organization.

## Future options:

Depending on the level of conception, it may be possible to document anticipated changes. These can range from additional features, process improvements that there was no capacity for, or inclusion of additional data sources. It could be relevant to include the limitations and assumptions of the report, such as factors that could affect the accuracy or reliability of the results.

## Intended audience and platform for delivery:

Is this an internal report or a product? Does the report need to be formatted for mobile or print? Tailoring the content and format of the report ensures that it is easily understood and accessible to the intended audience. For example, if the report is intended for an internal audience, it may include more technical details and jargon than if it were intended for a broader, external audience.

## Data Dictionary:

Include a definition for all field and object names in the data set and in the report itself. For field names, include data types and size. It will likely be necessary to update or iterate upon the data dictionary as the report evolves, reflecting decisions made by the developer.

## Naming Conventions:

Ensuring that object / field names are concise, logical, and most importantly consistent helps future developers understand a model and avoid confusion for end users who have questions or need further explanation about the information being presented.

Being distinct is just as important as being specific. Something as simple as [Total Sales] vs [Sum Sales] can mean two different things. We want to call attention to these differences where they exist while also minimizing the similarity in naming between things that are not the same to avoid confusion. Label attributes such as categorical or prefix hierarchy are also important to keep consistent. Whenever possible, be consistent with units and aggregations. Even if these attributes are well-defined, they can cause confusion when displayed together due to likeness.

Avoid overuse of abbreviations and acronyms as a general rule. Try to find the right balance between limiting the number of characters in an object name and implementing abbreviations and acronyms. Too many abbreviations and acronyms can lead to confusion by assuming that everyone knows what you mean.

## Data set / Report support contact:

Who owns the data set specific to the report? Who makes decisions about it? After initial development, who will receive the handoff? This is an important question to answer from the onset. If the solution support person will be someone who was not involved in all stages of development, they will need to be oriented to the artifacts of the solution. It is important to consider the complexity of the solution and the skill level of the person taking over.

If possible, it is a good idea to set up a communication structure that will take place alongside development to include the support contact early and often. This way you can avoid a painful data dump that can be inefficient and unconstructive. By breaking up the handover to the support contact in chunks, the volume of information is easier to consume and gives the opportunity for questions along the way.

Depending on who the solution is for, they may have their own development and deployment process in place. Even within an organization, there can be different processes among teams. It can be helpful to match your development efforts to the existing process.

What methods were used to manage the performance and scalability of the report? Including these methods in your documentation can help future developers maintain your solution or adapt it to future business needs as the data landscape changes. This could include:

### Optimizing query performance with techniques:

Such as indexing, partitioning, and query optimization to improve the speed and efficiency of data queries.

### Reducing data redundancy:

This can involve using techniques such as normalization to reduce the amount of duplicated data, which can improve performance and reduce the risk of data inconsistencies.

### Caching data:

This can involve storing frequently accessed data in memory or on disk, which can improve performance by reducing the need to query the data source.

### Sampling data:

This can involve selecting a smaller, representative sample of the data to analyze, rather than analyzing the entire dataset. This can reduce the amount of data that needs to be processed and improve performance.

### Partitioning data:

This can involve dividing the data into smaller, more manageable chunks, which can improve performance by allowing data queries to be executed in parallel.

## Describe the analytical methods & techniques used in the report:

Sometimes the methods used to examine data to gain insights and draw conclusions, are not immediately visible to the end user. There are many methods used to help to provide a more comprehensive and accurate understanding of the data, and some can be more sophisticated than others. Again, it is important to understand how the handover will take place and for whom in order to properly assess the level of explanation needed here. Some common analytical methods and techniques used in analytical reports include:

### Descriptive statistics:

Summarizing and presenting data in a way that is easy to understand and interpret. This can include measures such as mean, median, mode, and standard deviation.

### Inferential statistics:

Using sample data to make inferences or predictions about a larger population. This can include techniques such as hypothesis testing, regression analysis, and analysis of variance.

### Machine learning:

Using algorithms and models to automatically analyze and learn from data. This can include techniques such as classification, clustering, and regression.

### Data mining:

Using techniques such as association rules and clustering to uncover hidden patterns and relationships in data.

## Data Source:

What data sources feed into the data set? Detailed descriptions of the data sources used, including information on how the data was collected, cleaned, and validated can be useful for a handover, but at a minimum, you should include the connection string / path to where the source data can be found.

How often are they updated?

Who is the Subject Matter Expert?

Instructions relevant to staging (i.e., Test data vs Production data)

Manual interventions & justifications (if any)

When developing reports, ideally the model for the data should not need to be touched. If part of the model requires the intervention of the report developer, it is important to identify those interventions and justify their existence. For example, an excel data source may contain spelling mistakes or require some additional data-cleaning steps. Because these interventions can reduce flexibility, a change in the source data can cause disruptions in the future that would be difficult to identify without documentation. Another example might be data sources excluded from a scheduled refresh to be refreshed asynchronously.

## Atypical properties warranting special attention:

Any atypical features of the model or report that warrant special attention or care, for example:

* Bookmarks
* Visual Level Filters
* Workarounds for visuals
* Code Visuals (R, Python)
* Appsource and Custom Visuals
* Bi-Directional Relationships
* Role-Playing Dimensions / Inactive Relationships
* Many-to-Many Relationships
* Composite Modeling

# 2. Set up version control

Is there a data governance solution in place? It is not necessary for all aspects of an organization's data governance to be documented in each report produced for the organization. The specific data governance solutions that should be documented in a report will depend on the specific context and purpose of the report. At a minimum, it is important to define how your report is versioned and stored (i.e., SharePoint or other cloud storage that tracks changes, GitHub, etc.).

It is essential to have version control in place because it helps to ensure the integrity, reliability, and traceability of the data and models used in analytical reports. With version control, each version of the model is saved in a central repository, which tracks changes and provides a history of the model's development. This allows the person supporting the solution to easily access the latest version, and to roll back to previous versions if necessary. In addition, version control provides visibility on changes to individual objects, such as code chunks, measures, or tables, which can be useful for understanding the logic and calculations being used. This can help to ensure that the asset is accurate and consistent and can support better decision-making and business outcomes.

# 3. Assess the need for stability and push transformations upstream when appropriate

Pushing transformations upstream can help to improve the performance, stability, reliability, and maintainability of reports, and can support better decision-making and business outcomes.

Stability or instability typically refers to the ability of a model to produce consistent and accurate results over time, even when the underlying data or environment changes. This is important because it ensures that the model continues to provide accurate and meaningful insights, even when the data or environment changes. A very stable model can also be described as rigid or inflexible, while an unstable model may be less predictable, it also provides for a more open-ended environment which can be desirable in a state of data exploration.

Typically, a developer on the path of discovery will want to leverage a model that is less stable because they need the freedom to explore the data in a raw environment to uncover insights that might be missed otherwise. Once those insights are drawn, or in another context where a very specific data point is called for, then an increase in stability will be required.

Pushing transformations upstream refers to the practice of performing data transformations and calculations as early as possible in the data pipeline, rather than waiting until the data is loaded into the model. This can improve performance by reducing the amount of data that needs to be processed and can also improve maintainability by making it easier to update and manage the transformations. This can sometimes become a collaborative decision that involves the report developer and a database administrator.

# 4. Create organized model diagrams

By default, most tools used to construct a diagram will display a single view with no logic or organization. To someone trying to comprehend a complex model, having tables and relationships scattered on a canvas is not helpful and needs to be adjusted. Having a clear, consistent, and understandable diagram can support better communication and collaboration. When creating organized diagrams of data models, there are several important considerations to keep in mind, including:

## Clarity and simplicity:

The diagram should be easy to read and understand and should clearly convey the key elements and relationships in the data model. Avoid using complex or unnecessary details and focus on the most important information.

## Consistency and standardization:

The diagram should use a consistent visual language and style and should follow established standards and conventions for representing data models. This can help to ensure that the diagram is clear, consistent, and easy to interpret.

## Labeling and naming:

The diagram should use clear, concise labels and names for the elements and relationships in the data model. This can help to improve the readability and understandability of the diagram and can support better communication and collaboration.

## Grouping and hierarchy:

The diagram should organize the elements and relationships in the data model into logical groups and hierarchies, using techniques such as indentation and nesting. This can help to improve the clarity and structure of the diagram and can make it easier to navigate and understand.

## Separate views:

You can create additional tabs for specific views, such as:

* Separate views for each Fact table, showing all tables that filter it
* A separate view for calculation groups
* A separate view for disconnected tables

# 5. Develop unit tests and document test

In the context of developing data analytics reports, a "unit test" is a small, focused test that is used to verify the behavior of a specific component or feature of the report. A unit test typically focuses on a single aspect of the report, such as a calculation, transformation, or visualization, and is designed to test whether the component or feature is working as expected. Best practice is to write a test up front, anticipating the expected outcome of the feature being tested. It is not ideal to develop features and wait until the end to begin testing. This can lead to an overwhelming amount of double work as a result of failed tests or worse - the tests are never written, and a dysfunctional report is produced.

Unit tests are an important part of the development process for data analytics reports because they help to ensure the reliability and accuracy of the report. By running unit tests during development, you can quickly identify and fix any issues with the report and can ensure that the report is working as intended. Testing is a large topic with enough industry discussion to warrant its own paper but here are some key takeaways regarding testing:

## Documenting tests:

Unit tests could be documented in a test plan or specification, which outlines the steps and expected outcomes of the test. The test plan or specification can be used to guide the development and execution of the unit test and can help to ensure that the test is comprehensive and effective.

* Who did the testing?
* When was testing done?
* What testing method was used, how was the test conducted and where is the result stored?

# 6. Format code for readability

Formatting code for readability has several benefits. For one, it is more efficient, but also, it can make the code easier to understand, especially for other people who may need to work on the code in the future. This can save time and reduce the likelihood of introducing errors when making changes to the code. Additionally, properly formatted code can make it easier to spot errors and bugs, since the code will be organized in a way that makes it easier to scan and identify potential problem areas. Formatting code for readability can help to improve the maintainability and longevity of the codebase.

## Formatting tools:

* For DAX code: <https://www.daxformatter.com/>
* For M code: <https://www.powerqueryformatter.com/> (For Power Query within Power Bi, it can be particularly useful to format your code since comments will show up in tooltips on top of the applied step)
* For SQL: [https://poorsql.com](https://poorsql.com/)

# 7. Comment complex code

Code comments should exist as a separate mechanism for documentation that are specific to the code as it functions. Code comments are used to guide a developer modifying the code. Adding comments inside code chunks can help a reader understand the original intent of the code and more quickly assess the need for changes when the code is no longer working as needed. The information given in a code comment shouldn't be excessive or comment on what’s obvious. A comment on every line of code isn’t necessary. Use comments to translate the code into plain language where it would otherwise be challenging or time-consuming for the developer to determine the code’s function. Too many comments can make the code harder to read, so it's a good idea to use them wisely.

# 8. Eliminate unnecessary data

Sometimes, when we begin development, we frontload our reports with data objects but when the development nears completion, some of these objects are no longer needed and can be removed. This applies to the data set, the data model, and the visualization. Minimizing the data in the data set not only improves performance but is also an important part of data security. Removing unnecessary objects from the visual not only improves load time but declutters the interface for the reader interacting with the report. It is important to remember to consider the potential risks of removing data objects, such as the potential for breaking existing dependencies or losing valuable information.

* Could you consolidate or combine similar data sets, tables, or visualizations to reduce redundancy and improve clarity?
* Can static objects & text be moved to part of the background image?
* Can slicers be moved to the filter pane if they are intended to be interactive?
* Where applicable, use data sampling and sampling methods, such as stratified sampling, to analyze a representative subset of the data rather than the entire dataset
* Are there redundant views / pages / visuals / bookmarks that aren't being used?

## As part of improving the overall UI UX:

Formatting and design for the purpose of user interface and user experience could be expanded on greatly. For now, I will keep this topic within the context of decluttering and simplifying the presentation as part of elements that subtract value rather than add to it.

* Use data filtering and aggregation techniques to focus on the most relevant information and reduce the amount of data displayed
* Use data hierarchies and drill-down functionality to allow users to explore the data in more detail without overwhelming them with too much information at once
* Use data visualization techniques, such as sparklines and small multiples, to display data in a more compact and visually appealing way

# 9. Check data types & decimal precision

Are data types appropriately set for all fields? There are two primary reasons to be concerned with the data type constraints of your model:

## Calculation expectations:

Having the correct data types ensures that the data is being analyzed and processed in the correct way. Calculations, filters, relationships, and other aspects of the report risk having errors if the data types for the inputs are not appropriately constrained.

## Performance:

Having the correct data types can help to ensure the efficiency and performance of the analytical process. If data is stored in the wrong format, it may take longer to analyze and process. In addition, incorrect data types can take up more space in the database, which can lead to slower access times and other performance issues.

### Set minimal decimal precision:

This is a simple hygienic check which can have a significant performance impact by reducing the size of your model. Fields that have too high a decimal precision will have a much higher cardinality.

Cardinality refers to the number of unique values that are present in a particular data field. In general, having a low cardinality in a data field means that there are fewer unique values while having a high cardinality means that there are many unique values. Minimizing decimal precision, and by extension cardinality, will improve the performance of analytical processes.

# 10. Organize expressions & fields in display folders

Many analytics tools offer options for compartmentalizing contents of your build. Display folders can be used both in Power Query and in Power Bi data modeling to organize objects. Using a nested folder structure with consistent naming / grouping rules will make navigating the model a lot easier, both for developers and users. Ensure that the display folder structure consistent and ordered logically. This is a quality-of-life step that will save a lot of time when performing maintenance, versioning and testing.

Data Model:

In Power BI Desktop, columns and measures can be added to columns from the model diagram view in the “Properties” pane, under “General”. Other properties that can be adjusted here are field names, descriptions (mentioned above), and synonyms (for Q&A).

Power Query:

In Power BI Desktop, Display folders in Power Query are called “query groups”. They can be created by right-clicking a query and selecting “new group…”. Queries and groups can be dragged & dropped into other groups, making organizing queries easy.

# 11. Hide non-reporting fields

Hiding non-reporting fields can make the model simpler and easier to understand for business users and report designers. By only showing fields that are relevant for reporting and analysis, you can reduce clutter and help people focus on the data that is most important to them.

This can also improve performance. Fields that are not used in reports and visuals do not need to be loaded into memory, which can help reduce the overall size of the model and improve query performance.

Hiding non-reporting fields can improve the security and privacy of your data. By limiting the number of fields that are visible, you can reduce the risk of sensitive data being accessed by unauthorized users.

To hide a field in a data model, you can use the "Hide in report view" option in Power BI, or the "Visible" property in Excel. This will make the field invisible to users when they are creating reports and visuals, but it will still be available for use in measures and calculated columns.

## Examples of technical fields to hide:

* Relationship keys such as: Customer key, Date fields in fact table(s)
* Technical fields used for specific DAX or visual functionalities such as: A measure that dynamically provides a color hex code for a Power BI report

# 12. Security: manage access and permissions

Steps for data security will be largely specific to the tool you are using, any connections to other files / data and the type of connection, as well as how the report will be disseminated.

Some techniques are more universal than others. Data masking, for example, allows you to obscure sensitive data in a report or dashboard, while still allowing users to see the general trends and patterns in the data. You can use data masking to hide specific values, or to replace them with placeholder values like "XXX" or "\*\*\*".

There are several considerations for security when working with Power BI, including managing access and permissions. Here are a few examples of how you can manage security in Power BI:

## Manage access to Power BI workspaces and content:

You can control who has access to your Power BI workspaces and the content within them by adding users and groups to the workspace and setting their access permissions. For example, you can give a group of users "Edit" permissions, which allows them to create and modify reports and dashboards within the workspace.

## Use row-level security:

Row-level security allows you to control which rows of data a user can see in a report or dashboard. This can be useful for protecting sensitive data, or for ensuring that users only see data that is relevant to them. You can set up row-level security using either Power BI's built-in security roles or by using a dynamic security model that uses a user's Active Directory groups to determine which rows they can see.

Power BI allows you to encrypt data at rest and in transit, using Azure Key Vault. This can help protect your data from unauthorized access and is especially important for sensitive or regulated data.

## Use Azure AD to manage access and permissions:

You can use Azure AD to manage access to Power BI and other Azure resources, including setting up multifactor authentication and conditional access policies. This can help ensure that only authorized users have access to your Power BI content and can provide an additional layer of security.