Data Definition Quick Reference Guide

“You cannot effectively exchange what you cannot understand.”

The goal of this document is to summarize guidelines for creating clear, concise, and unambiguous definitions of data components.

Pro Tips:

- A data definition is a representation of a concept, via a descriptive phrase, which distinguishes the concept from related concepts.
- Defining data components by consistently applying guidelines helps Data Consumers with limited exposure to more easily understand, use or re-use, and benefit from said components.
- Widely-know terminology is usually ordinary and well understood. However, to elevate the value of definitions, develop descriptions for about the application or system in simple plain language for non-technical or first-time Users.
- Define a data component before naming it. Reversing this sequence may yield a definition that precisely describes the name but does not adequately describe the concept the data component represents.
- Refer to the Data Naming Quick Reference Guide.

Note: *The examples used below are not meant to be actual definitions for these specific terms but rather serve as examples to better explain each guiding principle.*

1. Define a data component without using self-referencing or circular definitions

A self-referencing definition is one in which the same term, or terms used in the name are used in the definition. Perhaps in a different order or by adding simple connectors, yet no more information is given to provide context to the definition.

Incorrect Example: **Requisition_Number:** The number for a Requisition.

Correct Example: **Requisition_Number:** The unique alpha-numeric identifier used to reference a request for products or services.

2. Define outliers based on particular business practices

A good definition not only represents a concept, but it also distinguishes any differences or nuances to the concept. This allows Users to distinguish data that exists in multiple applications.

Basic Example: **Vendor:** The unique identifier that represents a company that provides services or products.

Better Example: **Vendor:** The unique identifier representing a company that provides **medical** services or **medical monitoring** products to **Medicare and Medicaid** patients.

3. Define a data component in the singular tense unless the underlying concept itself is plural

In the following incorrect example, it is unclear whether a reference number applies to one article or several. Avoid confusion by consistently defining a data component in terms of a single instance of said component.

Incorrect Example: **Article_Number:** Reference number identifying articles.

Correct Example: **Article_Number:** The reference number that identifies a piece of writing in a newspaper, magazine, or other publication.

4. Define a data component in terms of what it is, not only what it is not

In the following incorrect example, the ‘negative’ definition leaves unclear what the data component actually represents.

Incorrect Example: **Freight_Cost_Amount:** Costs which are not related to packaging, documentation, loading, unloading, and insurance.

Correct Example: **Freight_Cost_Amount:** Cost amount incurred by a shipper in moving goods from one place to another.

5. Define a data component in a descriptive phrase or sentence(s)

A clearly written explanation is almost always necessary to precisely define a concept and avoid ambiguity.

Incorrect Example: **Agent_Name:** Representative.

Correct Example: **Agent_Name:** The name of a party authorized to act on behalf of another party.

6. Expand uncommon abbreviations on their first occurrence

Many abbreviations are not commonly known outside of specific contexts. Use the full term of an abbreviation to enhance understanding.

> **Incorrect Example: Tide_Height:** The vertical distance from MSL to a specific tide level.
>
> **Correct Example: Tide_Height:** The vertical distance from mean sea level (MSL) to a specific tide level.

7. Define a data component concisely with only the level of detail needed to state the essential meaning of the underlying concept

In the following incorrect example, the inclusion of extraneous material renders the definition less clear than the correct example. The additional verbiage does not enhance understanding.

> **Incorrect Example: Invoice_Amount:** The total sum of all chargeable items mentioned on an invoice, taking account of deductions on one hand, such as allowances and discounts, and additions on the other, such as charges for insurance, transport, handling, etc.
>
> **Correct Example: Invoice_Amount:** The total sum charged on an invoice.

In the following incorrect example, the additional language does not enhance understanding of the underlying concept in the present context.

> **Incorrect Example: Character_Set_Name:** The name given to the set of phonetic or ideographic symbols in which data is encoded, for the purpose of this metadata registry, or, as used elsewhere, the capability of systems hardware and software to process data encoded in one or more scripts.
>
> **Correct Example: Character_Set_Name:** The name for a set of phonetic or ideographic symbols in which data is encoded.

8. A data component's definition should be precise, unambiguous, and allow only one possible interpretation

In the following incorrect example, it is unclear what is meant by 'delivered'. A definition should make explicit what the underlying concept is.

> **Incorrect Example: Shipment_Receipt_Date:** The date on which a specific shipment is delivered.
>
> **Correct Example: Shipment_Receipt_Date:** The date on which the receiving party acknowledges the quantity, date, and time that ordered goods arrived.

9. A data component’s definition should stand alone

In the following incorrect example, the definition unnecessarily requires the aid of a second definition to explain the meaning of the first.

> **Incorrect Example:** **School_Location_City_Name:** See ‘School Site’.
>
> **Correct Example:** **School_Location_City_Name:** The official name of the city where the school is situated.

10. Define a data component without embedding other definitions

Definitions should only describe the data component at hand. If a term within a definition requires its own definition, it should be defined separately.

> **Incorrect Example:** **Sample_Type_Code:** The alphabetic code identifying the kind of sample. (e.g., G = Ground, A = Air, S = Structure, etc.). **A sample is a small specimen taken for testing**.
>
> **Correct Example:** **Sample_Type_Code:** The alphabetic code identifying the kind of sample. (e.g. G = Ground, A = Air, S = Structure, etc.)

11. Use consistent phrasing and logical structure for related definitions

Using common phrasing for similar or associated concepts enhances the association(s) for readers.

> **Consistent Example, Pt 1:** **GoodsDispatchDate:** The date on which goods were **sent off to their destination** by a given party.
>
> **Consistent Example, Pt 2:** **GoodsReceiptDate:** The date on which goods were **taken into possession** by a given party.

12. Define a data component using example data

When defining a data component, the definition may not convey the specific properties of the data component. It is acceptable to use example data to enhance clarity. When writing examples, use the same format for every definition. Every value is not necessary or required in the list.

> **Example, Pt 1:** **Vendor Status Code:** The alphabetic character used to indicate approval conditions for companies that provide services or equipment.
>
> **Example, Pt 2:** **Vendor Status Code:** The alphabetic character used to indicate approval conditions for companies that provide services or equipment. (e.g., A = Approved, C = Cancelled, P = Pending, etc.).

References

The Definitions Books: How to Write Definitions - https://www.unifiedcompliance.com/education/how-to-write-definitions/

A Quick Guide to Data Dictionary

What is a Data Dictionary?

A data dictionary[1] is a collection of descriptions of data objects or terms, definitions, and properties in a data asset. Data dictionaries provide information about the data.

Data dictionaries are an essential communications tool for data modeling, curation, governance, and analytics, especially when dealing with datasets that have been collected, compiled, categorized, used, and reused by different internal and external data Consumers across the organization.

Why do you need a data dictionary?

Data dictionaries provide valuable definitions for the data as well as help data Consumers understand any data asset before delving into the details within that asset. In order for data to be trusted and appropriately used, it must be understood and supported by clear definitions.

An established data dictionary can provide organizations many benefits, including:

- Greater data standardization and consistency across the organization or domain
- Improved analysis and decision-making based on better understanding of data
- Improved documentation
- Increased reuse of data

What is in a Data Dictionary?

A data dictionary consists of several data components, which contains multiple levels: **data asset**, **entity**, **attribute**, and **value domain**. Each level includes different components, but each component should be defined with the following properties:

Data Component Name	Data Component Name that represents a class of real-world entities or characteristics of those entities
Description	A short description for the data component name
Type	Logical or physical data component
Required	Required or optional data component
Sample	A sample of the data component

The data asset level is composed of one line that contains a data asset profile, which includes the data asset name, description, type, version, and create and last update date.

[1] Refer to the CMS Data Description Guidelines webpage to download the Data Dictionary Template.

The entity level is composed of one line per entity/table that contains all required properties for this element, such as logical and physical data element name, requirement ID (justification for having this entity in the system), security category (Consumers that can access this entity) and so forth.

The attribute level is composed of one line per attribute/column that includes logical and physical data element name (as well as parent element, i.e. entity/table name) and some logical and physical properties (data type and size, null option, is attribute/column a primary key, a foreign key, an unique/alternative key, PII field, etc.).

Finally, the value domain level is composed of a list of groups/collections of similar values (sex codes, account types, credit card types, enrollment plans, etc.) defined as business rules, look-up tables or list of valid values, and each available value within each collection.

Logical vs. Physical Data Dictionary

A logical data dictionary describes information in business terms and focuses on the meaning of terms and their relationship with other terms. A physical data dictionary represents data in a specific database and includes actual tables and columns in the data asset's schema.

- **Logical:** Business-related names, entities/attributes, logical naming convention (mixed case). Platform-agnostic.
- **Physical:** Technical names, tables/fields, platform-driven naming convention, field length and data types. Platform-specific.

How do you create and maintain a Data Dictionary?

Most data modeling tools and database management systems (DBMS) have built-in, active data dictionaries the capable of generating and maintaining data dictionaries. Data stewards may also utilize the CMS Data Dictionary Template[2] and guide for manually creating a simple data dictionary in Excel.

Best Practices:

- Start building a data dictionary during the gathering business requirements phase.
- The data dictionary is a living document that must be regularly maintained.
- If utilizing erwin, then Consumers should build the data dictionary from their data model using Report Designer.
- Reference the CMS Data Naming Quick Reference Guide[3] for data naming guidance.
- Reference the CMS Data Definition Quick Reference Guide[4] for data definition guidance.

[2] Refer to the CMS Data Description Guidelines webpage to download the Data Dictionary Template.
[3] For additional data naming guidance, refer to the CMS Data Naming Quick Reference Guide.
[4] For additional data definition guidance, refer to the CMS Data Definition Quick Reference Guide.

Data Naming Quick Reference Guide

This document summarizes guidelines for naming data components with syntactic consistency, semantic precision, and simplicity.

Naming data components by applying guidelines consistently makes it easier for data consumers who are not familiar with said components to identify, understand, use or re-use, and benefit from them.

1. Form of a Data Component Name

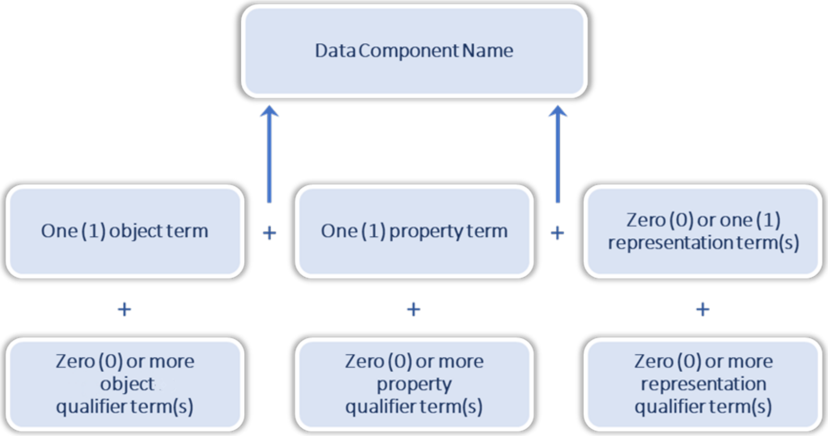
Generally, the name of a data component should take the form:

1. An object term,
2. a property term, and
3. if necessary, a representation term.

If necessary, qualifier terms may precede or follow an object, property, or representation term.

A **term** is a meaningful word, a commonly used abbreviation for a word, or an acronym.

Figure 1 - Form of a Data Component Name


Data Component Name
One (1) object term
+
One (1) property term
+
Zero (0) or one (1) representation term(s)
+
+
+
Zero (0) or more object qualifier term(s)
Zero (0) or more property qualifier term(s)
Zero (0) or more representation qualifier term(s)

Object Term

An **object term** represents a class of real-world entities or concepts.

In the name ‘**Person Street Address Text**’, ‘**Person**’ is the object term.

Property Term

A **property term** represents a characteristic of an object. A property has meaning but no inherent structure.

In the name ‘**Person Street Address Text**’, ‘**Address**’ is the property term.

Representation Term

A **representation term** describes the data type or value set of a data component.

Don’t use representation terms if they are redundant with property terms. For example, use ‘**Person First Name**’ instead of ‘**Person First Name Name**’.

The name of a data component that doesn't have child components should always use an appropriate representation term.

In the name '**Person Street Address Text**', '**Text**' is the representation term.

Qualifier Term

A qualifier term modifies another term to increase semantic precision and reduce ambiguity.

Limit the number of qualifier terms to the minimum required to make a data component's name unique and understandable within the component's context.

In the name '**Person Street Address Text**', '**Street**' is a qualifier term modifying the term '**Address**'.

2. General Naming Guidelines

Guideline	Explanation
A name should be composed of English words.	A name should be composed of words from the English language, using the prevalent U.S. spelling.
A name should only have specific characters.	Only use the following characters in name: • **Upper-case letters:** A B C D E F G H I J K L M N O P Q R S T U V W X Y Z • **Lower-case letters:** a b c d e f g h I j k l m n o p q r s t u v w x y z • **Digits:** 0 1 2 3 4 5 6 7 8 9 • **Punctuation characters (underscore, hyphen, period):** _ - . • **Spaces** The allowable characters may be further limited in other contexts. For example, SQL names may not contain spaces, hyphens, or periods and might be interpreted with or without case sensitivity.
Names should use consistent capitalization.	Within a context, names should follow a consistent capitalization convention (e.g., **camelCase**, **PascalCase**, **infix_underscores**).
A name should use common abbreviations.	A name should use commonly used abbreviations instead of full meanings. For example, use '**ID**' instead of '**Identifier**.'
A name should use singular forms instead of plural.	A noun used as a term in a name should be in **singular** form (e.g., '**Vehicle Depreciation Rate**') unless the concept itself is plural (e.g., '**Tool Suite Total License Cost**').
A name should use the present tense.	A verb used as a term in a name should be used in the **present tense** unless the concept itself is past tense (e.g., '**Petition Signatures Collected Amount**').
A name should only use essential words.	Avoid **articles**, **conjunctions**, and **prepositions** in a name except where required for clarity or by convention (e.g., '**Power of Attorney Code**').

The Layman's Guide to Reading Data Models

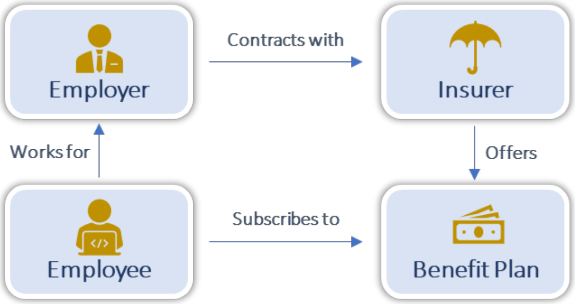
A data model shows a data asset's structure, including the relationships and constraints that determine how data will be stored and accessed.

1. Common Types of Data Models

Conceptual Data Model

A **conceptual data model** defines high-level relationships between real-world entities in a particular domain. Entities are typically depicted in boxes, while lines or arrows map the relationships between entities (as shown in Figure 1).

Figure 1: Conceptual Data Model



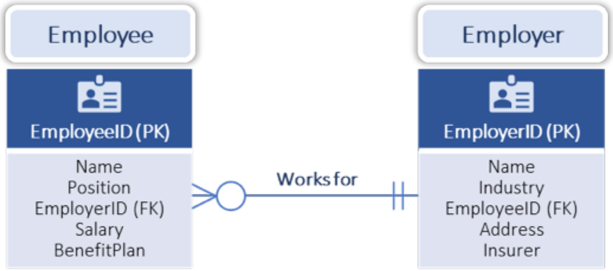
Logical Data Model

A **logical data model** defines how a data model should be implemented, with as much detail as possible, without regard for its physical implementation in a database. Within a logical data model, an entity's box contains a list of the entity's **attributes**.

One or more attributes is designated as a primary key, whose value uniquely specifies an instance of that entity. A primary key may be referred to in another entity as a **foreign key**.

In the Figure 2 example, each Employee works for only one Employer. Each Employer may have zero or more Employees. This is indicated via the model's line notation (refer to the Describing Relationships section).

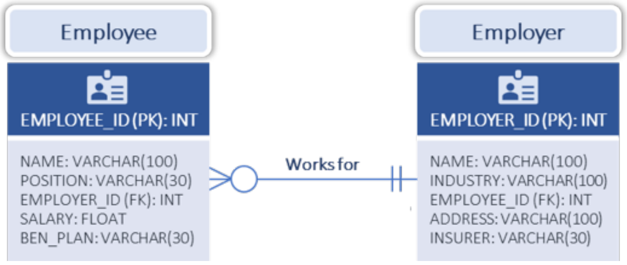
Figure 2: Logical Data Model



Physical Data Model

A physical data model describes the implementation of a data model in a database (as shown in Figure 3). Entities are described as tables, Attributes are translated to table column, and Each column's data type is specified.

Figure 3: Physical Data Model



2. Describing Relationships

Ordinality and Cardinality

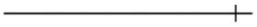
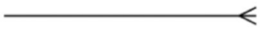
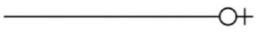
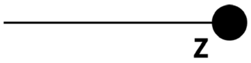
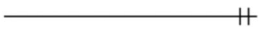
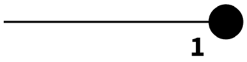
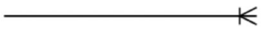
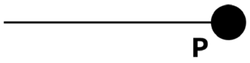
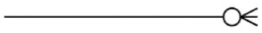
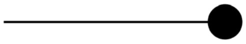
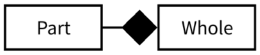
Logical and physical data models describe two entities' **ordinality** and **cardinality**, or the minimum and maximum number of times an instance of one entity can relate to instances of another entity.

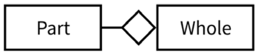
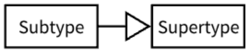
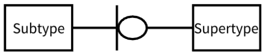
Line Notation Style

Different data models use different styles of line notation to indicate ordinality, cardinality, and other types of relationships between entities. In the examples above, ordinality and cardinality are described using crow's foot notation (the symbols at the end of each line).

Common notations in **Unified Modeling Language (UML)**, crow's foot, and **Integration DEFinition for Information Modeling (IDEF1X)** notation are described in the following table:

Table 1: Syntax in Common Data Modeling Notation Styles

Notation	Crow's Foot	UML	IDEF1X
One		N/A	N/A
Many		N/A	N/A
Zero or one		0..1	
One only		1	
One or more		1..*	
Zero or more		0..*	
(Specific range)	N/A	3..7	N/A
Composition*	N/A		"Is part of"

Notation	Crow's Foot	UML	IDEF1X
Aggregation*	N/A		"Is part of"
Subtype**	N/A		

***Aggregation** and **composition** are specific kinds of relationships. Aggregation means one entity can exist independently of another entity (i.e., an Employee and a Benefit Plan). Composition means one entity can't exist independently of another entity (i.e., an Employee must have an Employer).

A **subtype is an entity that has a parent-child relationship with another entity, a **supertype**. A supertype has attributes that are common to all of its subtypes.