# Naming Conventions

**by Sudheer Sharma - Sunday, August 23, 2009**

http://dwhnotes.com/data-integrator/naming-conventions

What do you think of this post?

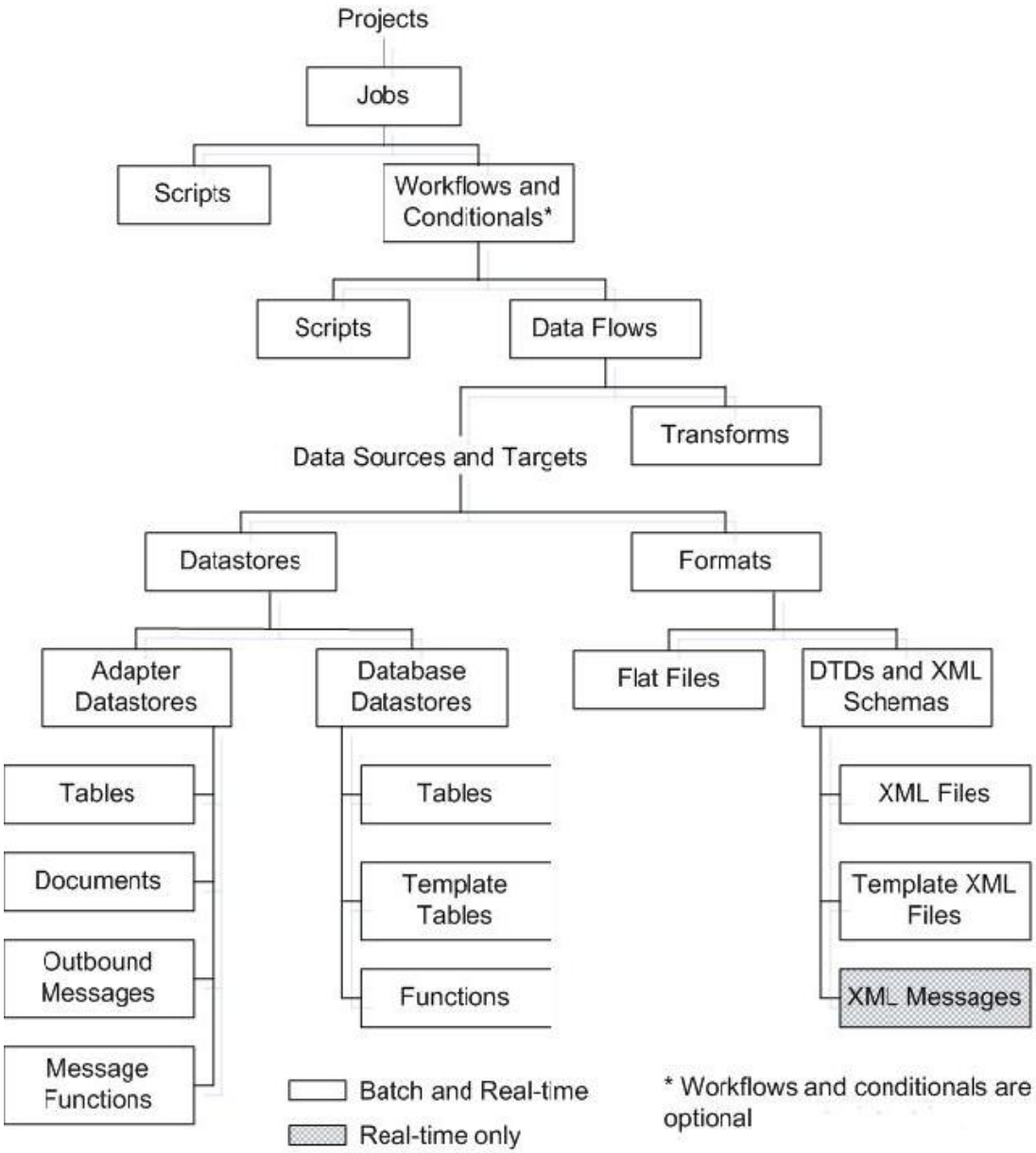
Awesome (9) Interesting (2) Useful (9)

When designing an ETL code using Data Integrator/Data Services its always a good idea to follow some sort of naming conventions. The following are some general guidelines that any DI/DS Developer should follow in naming the objects.

- The name of an Object should be short and intuitive.
- In some cases the naming conventions may not resolve into complete meaningful definitions. In such situations, we can provide descriptive information about the Object and its relationship to other Objects Description on the Object properties window.
- Do not use the common words like “load,” “get,” “check” in Object names.  While it seems appropriate to do so, these words are implied by the Object type itself – for example, all Dataflows get and load data.  Naming a Dataflow as “DF_GET_XXXXXX” takes up valuable space in real name.

Before naming the objects will see the objects hierarchy and their relationship.

**Technical Document says:** Any element or entity defined in Data Integrator environment is called an Object. The below picture depicts hierarchal relationship between major Objects types.




# Naming a Project Folder/Object

Everybody knows that ‘Projects’ are the highest level Objects in Data Integrator. They allow to group jobs that are dependent or belong to the same application. In general, the format “PRJ_<<Description/Subject Area>> can be followed. But in few cases multiple projects might have the same subject area or descriptions. In such cases “PRJ_<<Description/Subject Area>>_<<Project Code>> can be followed – where <<Project Code>> should be unique for the projects.

**Format:**

**PRJ_<<Description/Subject Area>>**

**Or**

**PRJ_<<Description/Subject Area>>_<<Project Code>>**

**Example:**

**PRJ_CUSTOMERS_IMPLEMENTATION**

**Or**

**PRJ_CUSTOMERS_IMPLEMENTATION_C0001**

**Naming Jobs**

Jobs are always specific to a Project. It is a good practice to include project code or abbreviation in its name. <<Project Code/Abbreviation>> in the format is to place project code or an abbreviation. <<Functionality Description>> in the format is to place load name or load type or load description.

**Format:**

**JOB_<<Project Code/Abbreviation>>_<<Functionality Description>>**

**Example:**

**JOB_C0001_SOURCE_TO_STAGE**

**JOB_C0001_STAGE_TO_TARGET**

## Naming Workflows

In general, Workflows are not specific to a Project. A project name may not be required to include in name. Workflows can have many types of names to reflect their usages. These are grouped as follows

- **Workflows with Dataflows that load a specific table:** These Workflows are single units that can load a specific table or file. They may in turn call one or more Dataflows, Scripts or even Workflows. Such workflows can be of the following format.

**Format:**

**WF_<<Target Table Name>>**

**Example:**

**WF_Customer_Dim**

<<Target Table Name>> is the final table name in the Workflow that is being loaded.

- **Workflows with Dataflows that load a group/category of tables:** These Workflows are several units that can load a subject area. An example would a Workflow that loads all the dimension tables of the Cutomers Implementation project. Such workflows can follow the following format.

**Format:**

**WF_<<Category or Subject Area>>**

**Example:**

**WF_Customers_SubjectArea**

<<Category or Subject Area>> is the category or Subject Area name of the tables that the Workflow loads.

- **Workflows that perform only a portion of a table load:** Often, particularly with aggregated fact tables and other complex transformations, it will take more than one work flow to load the table. This will be the case particularly if if-then logic needs to be employed during the load, or if parallel and sequential loading is to be mixed. An example of this would be a work flow that loads a table from three staging tables in parallel. Such Workflows can follow the following format.

**Format:**

**WF_<<Description with Target Table Name/s >>**

**Example:**

**WF_SalesFact**

<< Description with Target Table Name/s >> is the target table name with technical description of the Workflow.

## Naming Dataflows

Dataflows are the lowest level and are reusable objects in Data Integrator. Multiple Dataflows can reside in a Workflow. They are not specific to any other object, and as such, do not need to reference another object in their names. (NOTE: It is good practice to refer Project, Job, or Workflow that this data flow is called from in its object properties). Naming Dataflows follow the same rules as for Workflows. The following format can be followed in naming the Dataflows.

**Format:**

**DF_<<Target Table Name or Load Name/Description >>**

**Example:**

**DF_Customer_Dim**

**EDF_Customer_Dim** << Naming an embedded dataflow>>

<< Target Table Name or Load Name/Description >> is the complete name of target table or description of load.

## Naming Scripts

Scripts can be created in Jobs or Workflows. They can be used to declare and assign variables, call a SQL function or perform SQL statements. So the scripts should be descriptively named according to their function.

**Example:**

**SC_Initialize_Variables –** If the script is meant to initialize the variables at the beginning of a job.

**SC_Get_Latest_Date –** If the script is meant to execute a SQL statement to get latest date value.

## Naming Transforms

I would like to write more about the Query Component, since this is the common component and widely used in design.

**A Query transform is similar to a SQL Select statement that retrieves a data set that satisfies conditions that are specified.** Assigning meaningful names to Query transforms helps us to better understand the Dataflows and ETL process. So the Query transformations should be properly named according to their functionality.

**Example:**

**Qry_Calc_Avg_Periods –** If the Query transform is meant to calculate aggregate values.

**Qry_Filter_Dups –** If the Query transform is meant to distinct the input data set.

**Qry_Find_Dim_Keys –** If the Query transform is meant to do lookups for dimensional keys.

## Other Transforms

- **Case – Case Transform**
- **DT – Data_Transfer**
- **DG – Date_Generation**
- **ED – Effective_Date**
- **HF – Hierarchy_Flattening**
- **HP – History_Preserving**

- **KG – Key_Generation**
- **Map_CDC – Map_CDC_Operation**
- **Merge – Merge**
- **MO – Map_Operation**
- **Pivot – Pivot**
- **RPivot – Reverse_Pivot**
- **RG – Row_Generation**
- **SQL_<<Functionality>> – SQL Transform**
- **TC – Table_Compare**
- **Validation – Validation**
- **Xml_Pipeline**

**Naming Variables**

Variables are classified as local and global. Though variables are not reusable objects in Data Integrator environment, following a pattern in naming the variables – particularly global variables – improves the readability of the overall process. All global variables must start with "gbl" to distinguish from local variables. The format gbl_<<**Description**>> can be followed to name global variables. Local variable names can be identified and codified, as **$fiscal_year or $load_date etc.**

**Naming File Formats**

A file format is not specific to any Project. So it is not required to include Project name or other object name in its name. The best way to name a file format is to include original file name in the total format name. The following format can be followed to name the flat files.

**Format:**

**FF_<<Source File Name>>**

**Example:**

**FF_Customers**

<<Source File Name>> is the name of the original file name for which the format has to be created.

## Naming Custom Functions

Sometimes we need to build our own custom functions either for single use or for multiple uses. Proper naming conventions to custom functions would helps developer or the team to understand the functionality of that function easily.

**Format:**

**CuF_<<Technical/Functional Description>>**

**Example:**

**CuF_Date_Check**

<<Technical/Functional Description>> is description of the function.

## Naming Datastores

A Datastore provides connection to a particular database instance. Datastores may be created by an ETL developer. Use an intuitive name as a DataStore connection. The following format can be used in naming Datastores.

**Format: DS_<<Database Instance Name>>**

**Example:**

**DS_SOURCE**

**DS_STAGE**

**DS_TARGET**

Some developers would prefer to add(suffix) ‘DS’ at the end. But I would like to add at the begining.

What do you think of this post?

Awesome (9) Interesting (2) Useful (9)

---