Building/defining Types for the three main Aspects described in ISO/IEC 81346.

General

**Aspect object Types** are used as a blueprint for creating instances of aspect objects when building an asset information model.  
Such a **Type** is defined by its intended **Purpose**, a given set of inputs/output **Terminals**, and a given set of **Attributes**. No two types can have the same combination of these.

The primary **Aspects** of the asset information model are: **Functional, Product, Location**. Aspect object Types are constructed differently for each of these aspects because the different aspects apply different perspectives. The difference in use of Attributes is that:

* The **Functional** Aspect object Types apply “required attributes” – meaning attributes that must be assigned values when instantiating Aspect objects.  
  E.g. a pumping function must be assigned a *required* *nominal flow* value.
* the **Product** Aspect object Types apply “specification attributes” – meaning attributes with values specified.  
  E.g. a pump product specification includes a *maximum design pressure* value.
* the **Location** Aspect object Types apply both “required attributes” and “specification attributes”.  
  E.g. a room location both requires the *box dimensions* to be assigned a value for that room, as well as specifies the *fire protection level* value for that type of rooms.
* All Aspect object Types may also apply “optional or administrative attributes”.  
  E.g. *version number, graphical symbol, etc.*

Defining a Type

1. An aspect object Type is first defined by its essential **Purpose**. The Purpose is selected from a library of Purpose types. In the Functional Aspect the term Purpose is to be understood as about what the function does, i.e. **Activity**, whereas in the Product Aspect the term Purpose is to be understood as about what the thing is, the **Artifact**.  
   E.g. a Purpose could be either of: ‘Pumping’, ‘Separating’, ‘Storing’, ‘Switching’, ‘Transforming’ or other.
2. An Aspect object Type is further defined by the set of- and types of **Attributes** that is part of the Type. The attributes are selected from a library of Attribute types.  
   E.g. the Attributes could be one or more of: ‘Flowrate, volumetric’, ‘Pressure, delta’, ‘Voltage RMS’, ‘Frequency’, etc.
3. An aspect object Type definition may also include inputs and outputs (including bi-directional). This is done by means of defining **Terminals**. The Terminal Types are selected from a library of Terminal Types.  
   E.g. a Terminal could be ‘Input Material/Fluid: Hydrocarbon Multiphase Flow’.
   1. The Type of Terminal defines the Type of **Media** that can flow through it, and in what **direction**
   2. Media Type is selected from a library of Media types.
   3. Terminal Types have their own set of **Attributes**
4. The main types of Media are: Material, Energy, Force, Information. These are again divided into subtypes, e.g. Material/Fluid, Energy/Electric – and may be further subdivided until the desired granularity of specification is reached, e.g. Material/Fluid/Cold Seawater or  
   Energy/Electric/High Voltage DC.

Product Aspect object Type

The **Product** Aspect differs from other Aspects in that a Product Aspect object Type needs to be a **Composite**, meaning that it must also point to a number of lower-level **Simple** object Types in order to achieve a full Type specification.  
A **Simple** object Type only serves to specify a particular feature or quality of the product, and therefore does not specify inputs/outputs or Purpose, but only specifies attributes. There can be any number of Simple object Types pointed to by a Composite type. This supports an incremental approach to the asset information modelling, providing for a stepwise increase in granularity of specification.   
Every Type of product will generally require a lot of detailed Type definition before the product is fully specified, and since Simple Types specify a particular feature or quality of the product, different main Types (**composite**) are likely to reuse many of the same Simple Types  
  
  
Usually a Simple Type is used to refer to an industry standard or a design code which provides the details of the feature or quality. E.g. to specify that a product shall have a certain level of water protection, a Simple object Type ‘Open weather protection’ could be pointed to by the Composite Type. To provide the details of such a specification, this Simple object Type in this instance could refer to the standard IEC 60509 and the level IP67 as specified by this standard.

In summary, a fully defined Product Aspect object Type will have the definition of its **Purpose**, its inputs/ouputs, **Terminals**, and its main **Attributes** contained in its **Composite** main section, and extending on this, it will point to a number of **Simple** Types to address specific Type qualities or features.

Fig. Product Aspect object Type and example of Variants: FSB, Transport and Interface.  
(Note: for clarity admin- and optional attributes are omitted).

A picture containing graphical user interface

Description automatically generated

Variants of Types

There are variants of aspect objects, given by their main Purpose, which means as well that there are variants of the aspect object Types that provide a blueprint for the aspect objects. The variants are:

In the **Functional** and the **Product** aspects:

1. **FSB**: Functional System Block – Purpose to perform some essential function on some type of Media(s).  
   An FSB can have any of a range of Purposes, such as ‘mixing’, ‘separating’, ‘transforming’, etc., and can have any number of inputs/outputs.
2. **Transport** – Purpose to transport some type of Media over some distance, from output of one FSB to input of a different FSB.  
   Transports have only one Purpose: to transport. Consequently, there can be only one input and one output (alternatively two bidirectional).
3. **Interface** – Purpose to connect directly between an FSB output and a different FSB input.  
   Interfaces have only one Purpose: to connect. Consequently, there can be only one input and one output, and they are identical (alternatively two bidirectional), although since an interface is only the definition of how two Terminals connect, it does not have any real inputs/outputs, but is only about the interface where the connection is.

In the **Location** aspect:

1. **Location** – a space box which is virtually or physically delimited in 3 dimensions.  
   A Location has nil Purpose and zero inputs/outputs.
2. **Trajectory –** a locational definition of a from-to (e.g. pipe/cable route)A Location has nil Purpose and zero inputs/outputs.
3. **Range** – all positions where some position-dependent condition is true (e.g. zone with WiFi coverage).  
   A Range has nil Purpose and zero inputs/outputs.

Instantiation of Types

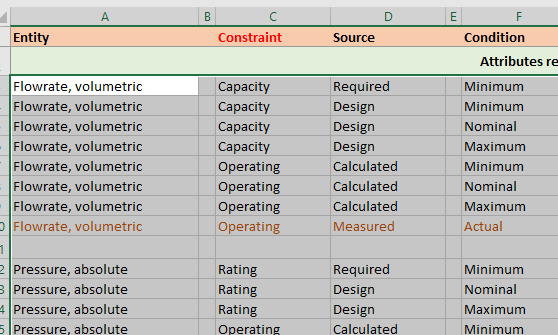
How Types are used for instantiating an Aspect Object (how the blueprint is used) depend on which Aspect is applied:

1. **Functional**: an object in this aspect is an instantiation of a Functional Aspect object Type.  
   Example of application: An ‘Electrical drive’ is represented by a Functional aspect object which is an instantiation of an ‘Electric Drive’ Type.  
   The aspect object Type defines the required attributes that must be given values when instantiating the aspect object, such as Output power, Input voltage, Rotational Speed, the inputs/outputs, and the Purpose.  
   Assigning values to the attributes is done for the aspect object as part of building the actual asset information model, e.g., assigning Output power = 75kW, Input voltage= 690V, and Rotational Speed= 3000rpm.
2. **Product**: an object in this aspect is an instantiation of a Product Aspect object Type, a **composite Type** which also refers to several sub-Types – **simple Types**. Each of the simple Types are defining details of specification. There is no principal limit to the number of such sub-Types that can be referred to by a given composite Type.  
   Example of application: An ‘Electric motor’ is represented by a Product aspect object which is an instantiation of an ‘Electric Motor’ composite Type. The Type defines required attributes with their values set, including by referring to several simple Types. These simple Types could include: Motor norm Type= IEC, Explosion protection Type= Ex ec, Cooling Type= IC411, Weather protection Type= IP65, etc., all defining required attributes and their values.
3. **Location**: an object in this aspect is an instantiation of a Location Aspect object Type.  
   Example of application: An ‘Electrical Equipment Room’ is represented by a Location aspect object which is an instantiation of an ‘Electrical Equipment Room’ aspect object Type. The aspect object Type defines required attributes that must be given values when defining the aspect object, such as length, Width, Height, Grid location, and Elevation. It also defines required attributes with values set, such as Hazardous classification= ‘Safe by ventilation’, Noise protection level= 70dB, etc..

Libraries and Lists

To support this structure there must be common libraries or list definitions for the following elements:

1. Type **variants**  
   These are relatively few but may grow as new Aspects are later introduced extending on the three basic Aspects. A library is not required since the list is not likely to extend beyond 20. At present the following are defined:
   * FSB – Functional System Block
   * Transport
   * Interface
   * Location
   * Trajectory
   * Range
2. **Purpose** Types  
   The types of purposes are likely to grow as more disciplines begin defining Aspect object Types and will further grow as more Aspects are introduced. A library will be needed to manage this common resource. Assuming in the order of 20 purposes for each discipline the total number of purposes is likely to be a few 100. For practical purposes the same term is used both for the Purpose as an Activity and as about an Artifact. E.g. ‘Pump’ can mean the purpose is ‘pumping’ or can mean the thing is a ‘pump’.  
   At present the following purposes are defined:
   * Process:
     + Manifold
     + Separate
     + Distribute
     + Pump
     + Compress
     + Mix
     + Exchange (heat)
     + Store
     + Control
     + Block
     + Condense
     + Heat
     + React
     + Transform
   * Electro:
     + Generate
     + Switch
     + Break
     + Distribute
     + Transform
     + Convert
     + Drive
     + Heat
     + Store
   * Automation:
     + Measure
     + Control
     + Actuate
   * Structural:
     + Support
     + Enclose
3. **Attribute** Types  
   There is a large library of attributes, and whenever possible these are aligned with industry standard definitions/ data dictionaries. This library is likely to grow significantly, probably into the 1000s.  
   At present the essential attributes for Process and Electro disciplines are defined (in separate spreadsheet):



1. **Terminal** types  
   Terminal types are defined by **direction** and **Media**. There are only 3 directions: Input, Output, Bidirectional, and there are 4 main Media: Material, Energy, Force, and Information. RDS category codes have been allocated for these different Terminal Types to support the need for identifying them as individual information objects.  
   However such a limited range of Terminal Types does not provide the required level of granularity. Therefore, Terminal Types refer to Media Types that are further sub-divided, e.g. allowing for a Terminal Type definition such as:
   * Input:Material/Fluid/Cooling medium  
     (where the Media Type referred to is Material/Fluid/Cooling medium).
2. **Media** Types  
   The library of Media Types is in principle open ended but is ordered in a hierarchy, beginning with only a limited number of upper-level media types. At present only a few Media Types for Process and Electro disciplines are defined (in separate spreadsheet), but below is an illustration of the hierarchical structure:
3. **Simple** Types  
   The library of Simple Types is likely to grow to a very large size, as these Types address the detailed specification of products and components. At present very few Simple Types are specified (will be in a separate spreadsheet), but her are a few examples, where both the Simple Type for weatherproofing and for noise limitation may also be pointed to by completely other main Types, e.g. a motor, a valve, etc.:
   * Main Type (composite): Centrifugal Pump
     + Simple Type: Centrifugal Pump, Type API 610
     + Simple Type: Weatherproof IEC 60529, IP67
     + Simple Type: Noise limitation 70dB, as per ISO 20361:2019

The libraries will at first be developed and be hosted by the developers of the pilot project, but are intended to become an industry shared resource, and will therefore be hosted by a suitable organisation. This hosting is intended to include the necessary tools, services and processes need to grow and maintain