# Filling Out the DD Attribute Class

The <*DD_Attribute*> class defines a single, specific attribute for use in labels. Attributes defined using this class are grouped into label classes via <*DD_Class*> definitions. Repeat this class for each specific attribute definition you need to create your local classes.

## <name>

*REQUIRED*

This is the tag name that you'll use in labels to identify this attribute. Tag names (as well as <*local_identifier*>, following) must be unique. ***This is not validated!*** Good organization and bookkeeping on this issue will be essential. If you accidentally duplicate a tag name, the output schemas will be affected in ways that are not entirely predictable, and these will likely not show up as validation errors in the schemas themselves.

*There are a couple of major issues with validation, or rather lack thereof, associated with this field:*

- *The character set is not restricted to ASCII. Unless you* ***know*** *the environment you're working in is configured to treat UTF-8 properly* ***and*** *you know how to get the proper code points into your* Ingest_LDD *file for UTF-8 characters, you should avoid UTF-8. You should probably avoid it anyway for this field.*
- *The character set is not restricted to letters, numbers, and underscores, or in fact even to printable characters. A fair number of those currently permitted characters are problematic for processing and should be aggressively avoided. Character entity substitutions are* not *a reasonable alternative here.*

## <version_id>

*REQUIRED*

This is a version specific to this attribute definition. Use it to track progress on specific definitions independently of the dictionary as a whole. There are no specific format requirements on this version, and it is not required to have any specific relationship to the dictionary *version_id*. In other words, you have a fair amount of freedom to define a versioning system appropriate to your development environment and circumstances.

## <local_identifier>

*REQUIRED*

The identifier *must* be unique (among <*local_identifier*> values) within this dictionary (as must <*name*> - the same validation warning applies here). This is the ID you will use to add this attribute to classes. This attribute can, in fact, have exactly the same value as the <*name*> attribute in the same class. Some dictionary writers append a prefix related to the containing class or some other circumstance to help with locating attributes and classes for maintenance.

*The character set in this case is restricted to ASCII, but there is still a major issue with lack of validation of the content of this field:*

- *The character set is not restricted to letters, numbers, and underscores, or in fact even to printable characters. A fair number of those currently permitted characters are problematic for processing and should be aggressively avoided. Character entity substitutions are* not *a reasonable alternative here.*

## <nillable_flag>

*REQUIRED*

The PDS shared namespaces follow this convention: If an attribute is required to appear in a class, but there is a reasonable case in which the attribute might not exist (lost data, or it is not a logically applicable concept in certain contexts, etc.), then the attribute may be *nillable*. *Nillable* means that the attribute is still present in the label, but it is explicitly declared to be "nil" and a reason for that is given.

For example, if you defined an attribute call *exposure_command* that would normally be required in all labels, but might be null because of a known data glitch, then you can declare it null in any particular label by doing this:

```
<exposure_command xsi:nil="true" nil_reason="missing"/>
```

or this, which is an alternate but equivalent syntax:

```
<exposure_command xsi:nil="true" nil_reason="missing"></exposure_command>
```

In PDS shared namespaces, an attribute can either be optional or nillable, but is never both in the same context. The list of *nil_reason* values is in the `pds:` namespace Schematron file.

Value values for *nillable_flag* are ***true*** and ***false***.

## <submitter_name>

*REQUIRED*

This is the name of the person responsible for this particular attribute definition. In large projects, dictionary development duties may be delegated to several individuals, and this field enables tracking that responsibility. In small projects, this will be the same as (or equivalent to) the *<full_name>* value in the *<Ingest_LDD>* parameters at the top of the file.

## <definition>

*REQUIRED*

This is the human-readable description of the attribute meaning. This will be reviewed by the external reviewers, and will also be available to user for the life of the data set (50 years, at least). As far as reasonable, the definition should not require intimate knowledge of the mission, instrument, or data to understand. So be careful using abbreviations, make cross-references fairly specific, and while you can assume the user looking up this definition is probably looking at a related data label, try not to assume he's read and remembered all the collection documentation.

## <comment>

*OPTIONAL*

This is an optional free-text field for additional comments. These are not transferred to the final dictionary files, so they should only be used for comments that are not relevant to end users. You may want to include notes to yourself, for example, or remarks about completion status or team members providing needed information.

## <Internal_Reference>

*OPTIONAL*

Sometimes a picture really is worth a thousand words. If you are defining an attribute that, for example, is related to temperature sensors located around the spacecraft, it could be particularly helpful to users if the definition linked to an image file that actually illustrated the physical locations referenced. To do that, label the image as a PDS product and include it in the documentation, then use this class to reference the image from this definition.

Use this method when there is a positive and unique benefit to linking to the cross-referenced product every time the definition is displayed, as in the case above. It would likely be much more distracting than helpful, for example, to link every calibration-related keyword to a calibration document.

This is filled out the same way as in Filling Out the Reference List Classes - <Internal_Reference>, except that the <*reference_type*> attribute must have the value **LDD_to_Source**.

***Note**: This value is not currently validated. Type carefully.*

## <Terminological_Entry>

*OPTIONAL*

The **Terminological_Entry** class may be used to provide a translation of this definition into a different language. It is intended to be used to support multi-lingual archives. It may *not* be used to provide alternate and conflicting definitions for the same attribute. Use multiple attributes with the same name and different <*DD_Attribute*> definitions for creating context-varying definitions for the same attribute name.

Note that you ***must*** use a UTF-8 character set for *Ingest_LDD* files that contain definitions with non-ASCII characters in them. Check your editor settings before you start!

### <name>

*REQUIRED*

This string contains the translation of the <*name*> attribute of *DD_Attribute* into the language specified by <*language*>. It must follow the same rules for names as the <*name*> field in the main *DD_Attribute* definition, except that rather than being constrained to using ASCII letters, it may use UTF-8 characters that have the Unicode "alphabetic" property.

### <definition>

*REQUIRED*

This is a free-format text field that contains the translation of the <*definition*> text of *DD_Attribute* into the language specified by <*language*>, following.

### <language>

*REQUIRED*

The English name of the language used for the *name* and *description* translations, above. So far, there are only two standard values defined: **English**, which may be used *only* if the primary language of the dictionary is *not* English; and **Russian**. If you want to see an additional value added to this list, contact your PDS node consultant and make the request.

#### <preferred_flag>

*REQUIRED*

If this flag is *true*, it indicates that this translation of the definition should be preferred over any others that might appear, including the main <*definition*>. This should never be true for data preparers creating new dictionaries to be archived primarily in the PDS, but as time goes on it may be useful for migrating dictionaries from the archives of the various IPDA partners to archives in other nations, and for international collaborative efforts.

Allowed values are **true** and **false**.

#### <instance_id>

*OPTIONAL*

This attribute, if it exists, must contain an identifier that is unique within the file. It exists as a handle for applications making use of *Terminological_Entry* information for advanced or location-specific processing. Routine dictionary operations can carry on without it, so you can omit it unless you have a specific use for it in mind.

#### <External_Reference_Extended>

*OPTIONAL*

This extended version of the standard <*External_Reference*> class includes two additional fields to hold a name and URL. It is not recommended to reference a URL in a critical archive document like a mission dictionary, so you should treat this class as if it were a standard <*External_Reference*>, and fill it out the same way as Filling Out the Reference_List Classes - <External_Reference>, but using the class name *External_Reference_Extended*.

### <DD_Value_Domain>

*REQUIRED*

The *DD_Value_Domain* class is used to provide constraints on what can appear as a value of the attribute being defined. The more specific you can be here, the more validation can be performed automatically for you in your labels, so take maximum advantage of the opportunity.

#### <enumeration_flag>

*REQUIRED*

If *true*, this flag indicates that the attribute has an enumerated value list, which will be defined subsequently. Enumerated value lists are an excellent way to catch and eliminate both human variation and typos in things like names and abbreviations used across instrument teams.

Valid values are **true** and **false**.

#### <value_data_type>

*REQUIRED*

This field defines the base data type of the attribute. It must be logically consistent with whatever additional constraints you add - so, for example, don't use a numeric data type that can't be less than zero and then try to extend that type by setting <*minimum_value*> to "-90". This sort of conflict is unlikely to be flagged by validators or *LDDTool*, and will likely lead to very confusing validation messages for your product labels.

Valid data types are:

- ASCII_AnyURI
- ASCII_Boolean
- ASCII_DOI
- ASCII_Date_DOY
- ASCII_Date_Time_DOY
- ASCII_Date_Time_DOY_UTC
- ASCII_Date_YMD
- ASCII_Date_Time_YMD
- ASCII_Date_Time_YMD_UTC
- ASCII_Directory_Path_Name
- ASCII_File_Name
- ASCII_File_Specification_Name
- ASCII_Integer
- ASCII_LID
- ASCII_LIDVID
- ASCII_LIDVID_LID
- ASCII_MD5_Checksum
- ASCII_NonNegative_Integer
- ASCII_Numeric_Base2
- ASCII_Numeric_Base8
- ASCII_Numeric_Base16
- ASCII_Real
- ASCII_Short_String_Collapsed
- ASCII_Short_String_Preserved
- ASCII_Text_Collapsed
- ASCII_Text_Preserved
- ASCII_Time
- ASCII_VID
- UTF8_Short_String_Collapsed
- UTF8_Short_String_Preserved
- UTF8_Text_Preserved

For descriptions of the various data types, see PDS4 Character Data Type Definitions.

## <formation_rule>

*OPTIONAL*

This is a human-readable description of how valid values for this attribute should be formulated. It might contain, for example, a description of the fields in a spacecraft clock string. It's a good idea to include this description if you are also using <*pattern*> to constrain the field values, to explain what a non-trivial *pattern* expression is actually doing.

## <minimum_characters>

*OPTIONAL*

Use this for any non-numeric data types that must have a minimum number of characters. For example, if you are defining a time field that must always be accurate at least to 0.01 seconds, then you can set <*minimum_characters*> to "11" as a sanity check.

## <maximum_characters>

*OPTIONAL*

Use this for any non-numeric data type that have a length restriction in the number of characters. For example, if want to make sure that a descriptive field you've defined for your labels will never overflow the 255 characters you've allocated for it in a mission database, you can set <*maximum_characters*> to "255" to catch potential overflow cases in validation.

## <minimum_value>

*OPTIONAL*

Use this to specify a minimum valid value for a numeric field. Typically a field described as a count can never be less that zero, for example, so you can specify that via <*minimum_value*>.

**If your attribute has a defined <*unit_of_measure_type*>,** you should include a <*specified_unit_id*> (see following) to define what unit of measure the *minimum_value* is in. You should do this for completeness and to avoid potential future editing errors even if your minimum is zero.

**N.B.:** *Use of* <specified_unit_id> *is not currently validated. Omitting it can lead to subtle or even undetected validation failures down the road.*

## <maximum_value>

*OPTIONAL*

Use this to specify a maximum valid value for a numeric field. This is useful, for example, if you're defining a keyword that corresponds to a measurement that is physically constrained. If your instrument can never rotate more than ten degrees off a particular axis, for example, you can specify this via <*maximum_value*>.

**If your attribute has a defined <*unit_of_measure_type*>,** you should include a <*specified_unit_id*> (see following) to define what unit of measure the *maximum_value* is in. You should do this for completeness and to avoid potential future editing errors even if your maximum value is zero.

**N.B.:** *Use of* <specified_unit_id> *is not currently validated. Omitting it can lead to subtle or even undetected validation failures down the road.*

## <pattern>

*OPTIONAL*

**Note:** LDDTool *cannot determine if your pattern will be syntactically valid in the context of the output XML Schema file. Therefore, it is essential that any dictionary preparer who is using the* <pattern> *attribute check the output XML Schema file for validity* before *attempting to use it to create or validate labels. You can validate the schema using a validating editor or any other XML schema-based validation tool you might have. (It references the PDS4 core schema, so you will need to feed that into the validator as well.) An invalid pattern in the .xsd file will cause validation failures when attempting to validate a label that will most likely be attributed to a missing or unreadable schema file.*

The *pattern* attribute contains a regular expression pattern (using a syntax defined in the XML Schema standard) that is applied during validation to require that values successfully match the pattern. A <*pattern*> to require that an *ASCII_Integer*-type field has exactly ten digits, for example, would look like this:

```
<pattern>[0-9]{10}</pattern>
```

Regular expressions are as powerful as they are tricky. You should plan to test any patterns in your data dictionary thoroughly for both spurious matches and unintended misses. For a quick summary of the key XML Schema regular expression features, see the XML Schema Regular Expressions - Basics page. In addition, PDS imposes the following constraints on this attribute:

- It may not be repeated. The work-around for this is to use a single pattern declaration with each desired pattern parenthesized and separated from the others by the alternation bar (' | ').
- Your pattern may not contain non-ASCII characters, even if you specify them by character entity reference. This effectively prevents patterns validation for any non-ASCII characters.

## <unit_of_measure_type>

*OPTIONAL*

For background information on units of measure of PDS4 label attributes, see the short page Units of Measure. Consistent with the unit handling in the PDS shared namespaces, when you define an attribute with units in your local dictionary you actually associate the attribute with a class of units via this attribute.

**N.B.:** If you are defining an attribute that has a natural unit associated with it, you *must* define a unit class for it, and you *must* always specify a unit when you use the attribute in a label. This is a basic and very hard requirement for properly documenting your data. No short-cuts, please.

The valid values for the attribute are the names of the unit classes defined in the *pds:* core namespace:

- Units_of_Acceleration
- Units_of_Amount_Of_Substance
- Units_of_Angle
- Units_of_Angular_Velocity
- Units_of_Area
- Units_of_Current
- Units_of_Frame_Rate
- Units_of_Frequency
- Units_of_Length
- Units_of_Map_Scale
- Units_of_Mass
- Units_of_Misc
- Units_of_None
- Units_of_Optical_Path_Length
- Units_of_Pressure
- Units_of_Radiance
- Units_of_Rates
- Units_of_Solid_Angle
- Units_of_Spectral_Irradiance
- Units_of_Spectral_Radiance
- Units_of_Storage
- Units_of_Temperature
- Units_of_Time
- Units_of_Velocity
- Units_of_Voltage
- Units_of_Volume
- Units_of_Wavenumber

Valid units for each of the types are enumerated on the Unit Classes Standard Values page.

***N.B.:*** Try not to define an attribute as having **Units_of_None**. If the value has no units, do not define it with a <*unit_of_measure_type*>. If a user might expect an attribute with this *name* to have a unit, state why it does not in the <*definition*>*.*

## <specified_unit_id>

*OPTIONAL*

Use this attribute if and only if these two things are both true:

1. You have defined a <*unit_of_measure_type*>.
2. You have also defined either a <*minimum_value*> or <*maximum_value*>, or both, for this attribute.

If this is the case, you should *always* include this attribute.

The unit indicated is only applied to the minimum or maximum value in the attribute definition, *not* to any label value. Specifically, it is *not* a default unit for the attribute.

It means that, for example, you can define a wavelength range using Ångstroms in your data dictionary, but the labels can use units of nanometers, and the PDS validation tool should be able to deal with the unit difference and make the appropriate check. (*But note that I haven't actually checked if this is true.*)

**N.B.:** *Use of* <specified_unit_id> *with* <unit_of_measure_id> *is not currently validated. Omitting it can lead to subtle or even undetected validation failures down the road.*

- *The actual unit of measure provided is not validated against the* <unit_of_measure> *unit class. Please check this manually and type carefully. You can find a list of valid units for each class on the* "Unit Classes Standard Values" *page.*

## <DD_Permissible_Value>

*OPTIONAL*

This class is used define the enumerated values that are valid for this attribute, with their meanings. You must repeat this class once for each enumerated value you wish to define.

*<value>*

*REQUIRED*

The value string that must be matched *exactly* in the labels. Both case and imbedded whitespace are significant. Do not include any sort of delimiter around the value you're defining; the *<value>* and *</value>* tags are the delimiters.

Enumerated values for things like instrument, phase, or target references can be a particularly effective way to a) make sure everyone is using uniform terminology for key parameters, and b) have the validation software hunt down typos in these critical fields. PDS encourages our data preparers to use enumerated values wherever they can be reasonably applied for these reasons.

*<value_meaning>*

*REQUIRED*

This should contain a brief, human-readable definition of the meaning of the related value. So, for example, if your required values are integers representing quality assessments, here's where you say that "O" corresponds to lost data, "1" to questionable data, and so one. If your enumerated values are mission-specific abbreviations, say for operational modes, here's where you can spell out the acronyms.

In general, your enumerated values should be reasonably self-explanatory if the attribute meaning is likely to be significant to a typical end-user who is trying to understand a product label. The additional explanation in *<value_meaning>* should augment the name for users who want a deeper understanding.

*<Terminological_Entry>*

*OPTIONAL*

As with the main attribute definition, in international collaborations it can be useful to have translations of the standard values and their meanings available in alternate languages. In those cases, the *Terminological_Entry* provides that capability. It is filled out the same way, and with the same considerations, here as for the *DD_Attribute* <Terminological_Entry> above.