Filling Out the DD Class Class

<*DD_Class*> is used to define a group of related attributes and, possibly, sub-classes. Repeat it once for each class you want to define

<name>

REQUIRED

This is the name that will appear in the class tag in your product label. It must be unique within the label. ***This is not validated!*** If the value is not unique, only one of the classes defined with this name will appear in the output schema.

<version_id>

REQUIRED

This is a version specific to this *DD_Class* definition. Use it to track progress on specific class definitions independently of the dictionary as a whole. There are no specific format requirement on this version number, and it is not required to have any specific relationship to the dictionary *version_id*. On 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 within this dictionary. This is the ID you will use to include this class as a sub-class of other classes. You can use the same value here as you do for <*name*> if you like, and the same validation warning applies. Some dictionary writers append a prefix related to the containing class or some other circumstance to help with locating attributes and classes for maintenance.

<submitter_name>

REQUIRED

This is the name of the person responsible for this particular class 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 should be a short, human-readable explanation of the class contents, or why it exists. This will be reviewed in the peer review and available to posterity.

<abstract_flag>

OPTIONAL

Do not use this attribute.

If this attribute has a value of *true*, it is supposed to indicate that this class cannot itself be used in labels, but rather is used as a parent class for deriving other classes in other dictionaries. This would be a fairly advanced technique that most local dictionary preparers would never need to employ.

In fact this attribute seems to have no effect on the output class definition at all.

<element_flag>

OPTIONAL

A value of *true* indicates that this class should be defined as an XML element (as opposed to an XML type) in the output XSD schema. You should include this attribute with a value of **true** for the top-level classes you plan to include in your labels. If you give a value of **false**, or do not include this attribute in the *DD_Class*, then the class will only be defined as an XML type.

Here's what's going on: For various reasons based on modeling, inheritance, and validation requirements, PDS dictionary schemas define classes and attributes as XML types, and create XML elements only at the highest levels. This ensures uniformity of definition and tag names down through the entire structure. When you create your *Ingest_LDD* input file, you should consider how you want to include your local classes in your labels. Any class you define with <element_flag>true</element_flag> will have an XML element definition created for it in the output dictionary, and can be included directly in the <Mission_Area> of your labels. All other classes can only appear as subclasses of a class that does have an XML element definition.

By default, **none** of the classes defined in your *Ingest_LDD* will have XML element definitions in the output, so you will have to use this attribute in at least one of your classes. As a quick example, say you've defined a structure like this:

```
<ClassA>
   <attribute1>
   <attribute2>
   <ClassB>
      <ClassD>
      </ClassD>
      <ClassE>
      </ClassE>
   </ClassB>
</ClassA>
```

In order to include the entire *ClassA* in the label, you must set element_flag in the *ClassA* DD_Class definition to **true**. Setting *element_flag* to **true** when defining *ClassD*, however, would make it possible to include <*ClassD*> directly in your label without the wrapper classes that contain it.

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

<Internal_Reference>

OPTIONAL

As for <*DD_Attribute*>, if you have a class that would be much more understandable in conjunction with an explanatory graphic or supplementary document, you can use this to associate that product with this class definition. Use the same sort of criteria here as for the <DD_Attribute>/<Internal_Reference>, above.

Note: *This class cannot be used because there are no defined values for the* <reference_type> *attribute in this context. The issue has been reported.*

<DD_Association>

REQUIRED

This class associates a *<DD_Attribute>* or *<DD_Class>* definition with this class, and defines the parameters of that association. **You may not define recursive or circular associations, either directly or indirectly.** Trying to do so, accidentally or otherwise, can lead to some random error messages.

Repeat this class once for every attribute or sub-class you want to include as part of this class. In the output XSD schema file, the attributes and sub-classes will be required to be in the same order as the order in which they were mentioned in *DD_Association* classes.

<identifier_reference>

REQUIRED

The value here should be the *local_identifier* value from the *<DD_Attribute>* or *<DD_Class>* that you want to include in the current class. It must match exactly.

Note: *Prior to IM version 1.9.0.0, the attribute <local_identifier> was used for this purpose. The <identifier_reference> attribute is required for version 1.9.0.0 and later of the IM and its associated LDDTool.*

Typically, each *DD_Association* will have a single *<identifier_reference>*. There is an advanced technique for creating a **choice** list in your output schema that requires multiple *<identifier_reference>* attributes in a single *DD_Association*. See the Advanced_LDDTool_Techniques page for more information.

Similarly, if you want to reference a type, attribute, or class from another namespace, there is a syntactical trick to do this available. *But you should only do this if you have **seriously** good reasons to do so.* You can find the details for this on the Advanced_LDDTool_Techniques page.

<reference_type>

REQUIRED

This string describes the relationship between the class you are defining and the attributes or sub-classes identified by the *local_identifier* value(s) immediately preceding.

The valid values are:

attribute_of -	The *local_identifier* references an attribute to be included in this class.
component_of -	The *local_identifier* references a class to be included as a sub-class in this class.
parent_of -	This class is a parent class from which the class identified by *local_reference* is derived.

The **parent_of** value is used in conjunction with some advanced dictionary-creation features. Contact your PDS node consultant if you are planning on using this and related constructs.

<minimum_occurrences>

REQUIRED

This attribute provides the minimum number of times the referenced attribute or sub-class is required to be included in the current class. If you want the attribute/sub-class to be optional, then give a value of "0" (zero).

Note: If you included a series of, for example, attribute *local_identifiers* in the *DD_Association*, and have a value of *minimum_occurrences* that is greater than one, bear in mind that in the label the first attribute will must be repeated *minimum_occurrences* times, **then** the second attribute will be repeated, and so one. This does *not* create a repeating group.

<maximum_occurrences>

REQUIRED

This attribute provides the maximum number or times the referenced attribute or sub-class may be included in the current class. If you want the attribute/sub-class to appear in labels a specific number of times, then *<minimum_occurrences>* and *<maximum_occurences>* should have the same value.

If you do not want to limit the number of times the attribute/sub-class can be repeated, use a value of "unbounded":

```
<maximum_occurrences>unbounded</maximum_occurrences>
```

<constant_value>

OPTIONAL

If you want to require that a particular attribute contains a specific, fixed value whenever it occurs in this class, you can assign it a constant value using this parameter. You might want to do this, for example, if you have classes that include the same keyword, but that keyword value is not variable in some classes.

<DD_Attribute_Reference>

OPTIONAL

> *This class is undocumented. Avoid it.*

<DD_Class_Reference>

OPTIONAL

> *This class is undocumented. Avoid it.*