Filling Out the DD Rule Class

The ability to define Schematron rules gives you another mechanism for validating labels through defined dependencies and contingencies over and above what the XSD Schema standard provides. For example, you can define a rule that requires that any label that contains *ClassA* also contains *ClassB*, or that all calibrated products contain a class with instrument calibration parameters.

Note: There are additional subclasses defined in the PDS4 schema for this class, but these are left-over from early prototyping efforts and are ignored by the process that analyzes this class content and produces the Schematron rules in the output. They will be removed in later releases and thus are not documented here.

In the following descriptions, the *sch:* prefix is used to identify elements from the Schematron namespace as they appear in the output Schematron file, and the *ex:* prefix is used to indicate the namespace being defined by the *Ingest_LDD* input file (which would be replaced by the actual *namespace_id* value in practice).

Before You Start

A few things to consider:

- Schematron does not recognize a default namespace, so make sure that *all* attributes and classes that appear in *anywhere* in the <*DD_Rule*> definitions have the appropriate namespace abbreviation prepended. This is where you will need to use the <*namespace_id*> value defined at the top of the input file to reference your own classes and attributes. Use the "pds:" prefix for anything you might be referencing from the core namespace.
- PDS4 Schematron files use *XPath 2.0* to define context paths and tests. If you are planning to do anything non-trivial with the <*DD_Rule*> class, a basic knowledge of *XPath 2.0* is both required. Some of the most common examples are included in the example file set (File:LDDTool 1E00 examples.zip) to get you started, but you should have at least some minimal knowledge of the syntax of **XPath** specifications and Schematron statements before proceeding, or the terminology can be rather opaque and any errors difficult to interpret.
- There is much more flexibility in Schematron itself than there is in these classes. Keep your rules very simple and very explicit, and this class will do the job. Anything more intricate will require hand-editing the output Schematron, or requesting an upgrade to LDDTool - which you can do through your PDS contact. (Please provide details and examples when you do.)
- Each <*DD_Rule*> class you include in your dictionary will result in one Schematron <sch:pattern> definition in the output file. All rules that need to be tested within that context *must* be defined in a single <*DD_Rule*> class, or the output schema will likely not perform as expected. *LDDTool* does not check for duplication of context, so author beware.

<local_identifier>

REQUIRED

This value provides a handle for this rule within the dictionary input file. It must be unique within the *Ingest_LDD* file.

<rule_context>

REQUIRED

The value of this attribute is the *Xpath* path defining the context (that is, the specific class or attribute you want to test) for the rule. Specifically, it provides the value of the context= XML attribute of the Schematron <sch:rule> element. It needs to be specific enough to identify the correct context for the test to be executed. For example, if you have multiple instances of the *<pds:Internal_Reference>* class including in your dictionary, you will need to provide different *<pds:reference_type>* values for each. To distinguish those different contexts, you will have to include enough path information to identify each occurrence of *<pds:Internal_Reference>* uniquely. Typically, this would mean including the class that contains the *<pds:Internal_Reference>* class in the path.

The best and highly recommended practice is to group all the tests you want to perform on a particular context under a single *<DD_Rule>* for that context. This grouping is required in the output for proper Schematron validation, but *LDDTool* will group rules with the same *<rule_context>* in the output Schematron file even if they are defined in separate *<DD_Rule>* classes. Grouping the rules in your *Ingest_LDD* file makes maintenance and trouble-shooting easier, and lessens the chance that you will define the same (or nearly the same) rule twice.

<rule_assign>

OPTIONAL

This attribute lets you create a variable assignment within your rule context. This variable will be available for use in your subsequent *<DD_Rule_Statement>* attributes as appropriate. The value of this attribute is the attribute definitions for a Schematron *sch:let* element, so do not include "let" in your value. Here's a very simple example:

> Including:
>
> <rule_assign>name="good_val" value="3"</rule_assign>
>
> in your *Ingest_LDD* file will result in this line:
>
> <sch:let name="good_val" value="3"/>
>
> in the corresponding context section of your output Schematron file.

You can then use the variable reference $good_val in any *<rule_test>* defined within this *<DD_Rule>* class. The *value* is not required to be a constant - it can be any valid *XPath 2.0* expression and can reference attributes from the schema file. You may repeat this attribute if you have more than one variable to define. Note, however, that variables are not translated in the message reported out - so using your variable in the *<rule_message>* text will *not* have the expected result.

Note that most dictionary writers will never have a reason to use this capability - standard values and constant values should be defined using the options provided by the *<DD_Attribute> and <DD_Class>* templates. But in some complex mission dictionary scenarios it might be a useful technique to have available, so here it is.

<DD_Rule_Statement>

REQUIRED

This class defines the various bits and pieces that go into a single <sch:assert> or <sch:report> element in the output Schematron file for the context defined by

the containing <*DD_Rule*> class. Repeat this class if you have more than one rule to define within the context defined by that <*rule_context*>.

<rule_type>

REQUIRED

This value indicates the type of test to be defined. There are only two defined values you should ever use:

Assert - Creates an <sch:assert> element within the <sch:rule>.

Report - Creates a <sch:report> element within the <sch:rule>

An *assert* statement does nothing if the associated <*rule_test*> evaluates to *true*. If the test is not true, the <*rule_message*> is displayed. *Assert* statements are usually used for error-detection.

A *report* statement does the opposite - it displays the <*rule_message*> text only if the <*rule_test*> is true. *Report* statements are more often used for information-gathering (to report things that are valid, but interesting).

<rule_test>

REQUIRED

This attribute contains the text that will go into the test= XML attribute of the <sch:assert> or <sch:report> element being defined. Schematron does not recognize a default namespace, so make sure that *all* attributes and classes that appear in your test (or anywhere in your rule) have the appropriate namespace abbreviation prepended. Use the <*namespace_id*> value defined at the top of the input file to reference your own classes and attributes; use the "pds:" prefix for anything you might be referencing from the core namespace.

The test string must use XPath 2.0 syntax. It's beyond the scope of this wiki to cover XPath syntax, but you can find some of the most common examples illustrated in the sample files File:LDDTool 1E00 examples.zip provided on this wiki.

<rule_message>

REQUIRED

This attribute provides the message that will be displayed when the associated sch:assert or sch:report is triggered by the test. This the message that your dictionary users - the ones creating and validating labels that use your dictionary classes - will see, so please try to make it as helpful as possible.

<rule_description>

OPTIONAL

Use this free-text field to provide a human-readable explanation of what is being tested and why, if it is not already clear from the *rule_message* text. This appears to be entirely internal documentation - the description is *not* transferred to the output Schematron file produced by *LDDTool*.