Filling Out the Table Character Data Structure

The ***<Table_Character>*** class contains the data structure definition for a character table. Each row in the table has the same structure, defined in a *Record_Character* class. Records are themselves composed of scalar fields, or sub-records called "grouped fields".

For additional explanation, see the PDS4 *Standards Reference*, or contact your PDS node consultant.

For a video walkthrough of filling out the Character Table class, watch this video:
Filling Out the Character Table Class Video

Following are the attributes and subclasses you will find in <*Table_Character*>, in label order.

Note that in the PDS4 master schema, all classes have capitalized names; attributes never do.

<name>

OPTIONAL

If you'd like to give this table a descriptive name, here's the place to do it.

<local_identifier>

OPTIONAL

If you want to reference this *Table_Character* from somewhere else in this label, give it a formal label here. Use an identifier that would make a valid variable name in a typical programming language, and you should be OK syntactically.

<md5_checksum>

OPTIONAL

Use this attribute to provide the MD5 checksum of the object *only*. If the object occupies the entire file, then the checksum should be given as an attribute of the <*File*> object. This checksum should be calculated using *only* the bytes defined as being part of this table.

<offset>

REQUIRED

The offset, in bytes, from the beginning of the file holding the table data to the first character in the table. You must specify the unit for this keyword, like this:

```
<offset unit="byte">0</offset>
```

<records>

REQUIRED

This is the total number of records in the table. Note that in a *Table_Character* table, each record including the last must have a carriage-return/linefeed record delimiter at the end.

<description>

OPTIONAL

This attribute provides a place for additional free-format text comments.

<record_delimiter>

REQUIRED

This attribute must have the value **Carriage-Return Line-Feed or Line-Feed**. The data file must also have carriage-return/linefeed record delimiters, of course.

<Uniformly_Sampled>

OPTIONAL

If this *Table_Character* contains records which are uniformly spaced in some dimension (time, wavelength, distance, etc.), you can use this class to define that dimension and interval rather than including an additional field in each row to hold the value explicitly. The details are on the Filling Out the Uniformly Sampled Class page.

<Record_Character>

REQUIRED

This class describes the structure of one complete record in the table.

<fields>

REQUIRED

The number of *Field_Character* classes directly under (that is, in the first nesting level of) the *Record_Character* class. Do ***not*** count *Field* classes nested under *Group_Field* classes.

If your *Record_Character* contains only one or more *Group_Field_Character* classes, this will have a value of zero.

<groups>

REQUIRED

The number of *Group_Field_Character* classes directly under (that is, in the first nesting level of) the *Record_Character* class. Do ***not*** count *Group_Field_Character* classes nested under other *Group_Field_Character* classes.

If your *Record_Character* contains only one or more *Field_Character* classes, this will have a value of zero.

<record_length>

REQUIRED

The total length of the record, including all fields, all repetitions of group fields, any space between fields, and the record delimiters. You must specify a unit of bytes for this value:

```
<record_length unit="byte">1234</record_length>
```

A Note about Fields and Group Fields

Records are composed of *Fields* and *Group Fields*. A *Record* must have at least one of those (either will do), and can have an arbitrary number of them, in any order (that is, you can have *Fields* and *Group Fields* interspersed). Note, however, that *Group Fields* are ***never*** necessary - they are a notational convenience to save writing out large numbers of essentially identical *Field* definitions.

<Field_Character>

OPTIONAL

This class defines a single, non-repeating scalar field.

<name>
REQUIRED

The name of the field. SBN recommends that this be something fairly human-readable that can be easily turned into a variable name for use in applications, or displayed as a meaningful column heading.

<field_number>
OPTIONAL

This is the sequential number of the *Field_Character* definition. For SBN data products, the *field_number* is intended to be a help to human readers trying to map field definitions to columns in a print-out of the *Table*.

> The *Standards Reference* lays out rules for using the *field_number* in cases where there are *Group_Field_Character*s present which can be useful in programmatic contexts, but not so much in the visual-inspection case

<field_location>
REQUIRED

This is the location, in bytes, of the first character in the field relative to the start of the enclosing *Record_Character* or *Group_Field_Character*. (Note that locations begin with one, rather than zero.) You must indicate a unit of bytes for this field:

```
<field_location unit="byte">1</offset>
```

This must be the location of the first byte that actually constitutes the field data. It must *not* correspond to any field delimiter or gutter space included in the data table.

> *As a rule and as good practice,* table fields should not overlap*. That sort of thing can hide subtle errors from users, reviewers, and software. If you have a good reason to consider defining overlapping fields, please make sure you get that OKed by your consulting PDS node early on in the development process.*

<data_type>
REQUIRED

The type of the values in the field. This must be one of the values listed in the Standard Values Quick Reference.

<field_length>
REQUIRED

The length of the field, in bytes. You must specify the unit:

```
<field_length unit="byte">12</field_length>
```

As with *field_location*, this must include only the bytes containing field data. Do not include any delimiters or gutter space that might be included in the data table.

<field_format>
OPTIONAL

The value of this attribute is a string representing the read/print format for the data in the field, using a subset of the POSIX print conventions defined in the *Standards Reference*, and also described on the PDS4 Field Format Conventions page.

<validation_format>
OPTIONAL

The validation_format attribute gives the magnitude and precision of the data value with the expectation that both will be validated exactly. A subset of the standard POSIX string formats is allowed.

<unit>
OPTIONAL

If the value in this field has an associated unit, this is where it goes. This value is case sensitive, and you may use characters from the UTF-8 character set (like the Angstrom symbol) where appropriate.

Note: If a field contains a unitless value, then there should be no <*unit*> attribute. **NEVER** include a null *unit* value, or even worse, this: <unit>N/A</unit>.

<scaling_factor>
OPTIONAL

If the observational values of this field were scaled prior to writing, this attribute should contain the value the data must be multiplied by to get back to the original value. Scaling factors are applied prior to adding any offset.

<value_offset>
OPTIONAL

If the observational values of the field were shifted by an offset prior to writing, this attribute should contain the value that must be added to each field value to get back to the original value. Offsets and added *after* the scaling factor, if any, is applied.

<description>

OPTIONAL

Free-format text describing the content of the field.

<Special_Constants>
OPTIONAL

This class defines flag values used in the data table to indicate why a particular field value is not available, and to document limiting values for the field. It is identical to the <*Special_Constants*> class used in the *Array* classes. For details, check the Filling Out the Array 2D Data Structure - <Special_Constants> page. Here is a quick list of the special constants included in this class:

- *saturated_constant*
- *missing_constant*
- *error_constant*
- *invalid_constant*
- *unknown_constant*
- *not_applicable_constant*
- *valid_maximum*

- *high_instrument_saturation*
- *high_representation_saturation*
- *valid_minimum*
- *low_instrument_saturation*
- *low_representation_saturation*

You may use as many of these within the *Special_Constants* class as are applicable to the field. Say, for example, you have a table of compiled properties, and in one particular integer field there are cases where either the value is unknown, or in some cases the field itself is just not applicable. If you wanted to document that the valid range of values is 100-500, and you are using "-999" to indicate "Unknown", and "-888" to indicate "Not Applicable", you would adding this to your *<Field_Character>* class:

```
<Special_Constants>
  <unknown_constant>-999</unknown_constant>
  <not_applicable_constant>-888</not_applicable_constant>
  <valid_maximum>500</valid_maximum>
  <valid_minimum>100</valid_minimum>
</Special_Constants>
```

<Field_Statistics>
OPTIONAL

If you want to include things like extrema, mean value, and such for all the values that occur in this field through all the records in the table, this is the place to do it. This class is identical for all *Field* types. For details, see Filling Out the Field Statistics Class. Here is a quick list of the field statistics available in this class:

- *local_identifier*
- *maximum*
- *minimum*
- *mean*
- *standard_deviation*
- *median*
- *description*

<Group_Field_Character>

OPTIONAL

This class defines a set of *Field_Character*s and *Group_Field_Character*s that repeats a given number of times in each record. *Group_Field_Character*s may be nested.

<name>
OPTIONAL

You can use this field to specify a name for the group, if you like.

<group_number>
OPTIONAL

Analogous to *field_number* for scalar fields, this is a sequential number useful for referencing *Group_Field_Character* classes at a single nesting level of a complex *Record_Character* definition.

<repetitions>

REQUIRED

The number of times the complete set of *Field_Character*s and *Group_Field_Character*s comprising this <*Group_Field_Character*> repeats.

<fields>
REQUIRED

The count of *Field_Character* classes directly under (i.e., at the first nesting level below) the *Group_Field_Character* definition. This will be zero if the group contains no *Field_Character* classes.

<groups>
REQUIRED

The count of *Group_Field_Character* classes directly under (i.e., at the first nesting level below) the present *Group_Field_Character* definition. This will be zero if the group contains no nested *Group_Field_Character* classes.

<description>
OPTIONAL

Here's a place to describe what this grouping of fields represents.

<group_location>
REQUIRED

This is the location of the first byte of the first field of the first repetition of this group relative to the containing *Record_Character* or *Group_Field_Character* location. If the group starts at the beginning of the containing *Record_Character* or *Group_Field_Character*, this has a value of one. You must specify a unit of "byte" for this value. For example:

```
<group_location unit="byte">1</group_location>
```

This value should be set bearing in mind that it will be used as the base offset for locating each repetition of the group. In other words, the location of the start of the first repetition of the group is *group_location* + 0*(*group_length*/*repetitions*); the location of the start of the second repetition of the group is *group_location* + 1*(*group_length*/*repetitions*), and so on.

<group_length>
REQUIRED

This is the length of the entire group - that is, ***the length of one repetition of the group, multiplied by the value in the* <repetitions> *attribute***, above. You must indicate a *unit* of "bytes" for this length. It must be evenly divisible by the value of *repetitions*.

Note: *This value is **not** validated. Calculate carefully.*

Fields and Nested Groups

As in the *Record_Character*, the *Group_Field_Character* may contain either *Field_Character* classes, or *Group_Field_Character* classes, or both intermixed. *Group_Field_Character* classes may be nested arbitrarily deeply. The requirements for these data structure classes inside a <*Group_Field_Character*> are identical to those above.