Filling Out the Packed Data Fields Sub-Structure

The ***<Packed_Data_Fields>*** class is used to define multiple sub-fields within a single *<Field_Binary>* value. These values do not have to observe byte boundaries, and so are defined in terms of bits.

> ***Note:*** *The whole* Packed_Data_Field *scheme is poorly defined. The class is not mentioned in the Standards Reference at all. The Data Dictionary does not (and can't, really) put constraints - including a byte order constraint - on the containing* <Field_Binary>*, nor does it specify bit order (although "most significant bit first" is almost certainly correct).*

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

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

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

<bit_fields>

REQUIRED

This attribute contains the number of *<Field_Bit>* classes in this *<Packed Data Field>*.

<description>

OPTIONAL

This attribute provides space for a free-format text description of the packed data.

<Field_Bit>

REQUIRED

This class, analogous to the *<Field_Binary>* class, defines a binary field in terms of bits rather than bytes. There must be at least one of these in the *Packed_Data_Fields* class, and there may be many.

<name>

REQUIRED

The name of the bit field.

<field_number>

OPTIONAL

> *The definition for this attribute is not sufficiently specific for it to be used reliably. It is not clear, for example, whether the* field_numbers *within a* Packed_Data_Field *always start at one, or continue numbering started at a higher level. If you must label packed data before its various issues are resolved, just do something reasonably with this and document what you've done in the* description *attribute of the* <Packed_Data_Fields> *class.*

<start_bit>

REQUIRED

The first bit comprising the value in the bit field. Numbering begins at 1.

It does not appear to be explicitly stated, but the only rational way to approach the bit-parsing problem is to require that the containing field be treated as character data - that is, the bytes are left in their file storage order for parsing. Then bit 1 is the most significant bit in the sequentially first byte in the field, and the bits are numbered sequentially across byte boundaries for the length of the field.

<stop_bit>

REQUIRED

The last bit comprising the value in the bit field.

<data_type>

REQUIRED

This must be one of two standard values: **SignedBitString** or **UnsignedBitString**.

***Note:** These standard values do not follow the syntax pattern of most standard values, and in particular of other* data_type *values.*

<field_format>

OPTIONAL

This attribute should correspond with the *data_type* field.

<unit>

OPTIONAL

If there is a unit of measurement associated with the extracted value, here's where it goes.

<scaling_factor>

OPTIONAL

If the original value was scaled to fit into the bit field, this attribute is the place to put the scaling factor to be applied when parsing the data into program memory. The value extracted from the packed data field is multiplied by this value before any *value_offset* is applied.

<value_offset>

OPTIONAL

If a value was applied to the original value to fit it into the bit field, this attribute is the place to put that offset value for use when parsing the data into program memory. The value extracted from the packed data field is first multiplied by the *scaling_factor*, then this *value_offset* is added.

<description>

OPTIONAL

This attribute holds free-format text containing a description of the extracted bit field value.

<Special_Constants>

OPTIONAL

This class defines flag values used to indicate that a particular field value is unknown. 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 available in this class:

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