PDS4 Field Format Conventions

The <field_format> attribute found in table field definitions is equivalent to the PDS3 *FORMAT* keyword in *COLUMN* definitions. In PDS4 we are using a subset of the POSIX I/O conversion specifiers (as used in most modern programming languages with a *printf* statement or the equivalent), rather than the FORTRAN specifiers used in PDS3.

NOTE: As of this update, this is still a *proposal*, not a standard. The details have been discussed by the DDWG, and the following text edited accordingly for the next round of discussion. There is one outstanding question:

- If your binary field contains packed data, what should be used for a field format?

Do check for a final standard before submitting data for review or archiving.

Why bother?

The *field_format* attribute provides two potential benefits to users and the archive:

1. In character tables, the *field_format* specification provides width and precision information that can be used in validating individual values in the table data.
2. In binary tables, the *field_format* specification can be used as an output format for converting binary numeric values to a character form without losing or overstating significant digits.

The POSIX Standard

The latest edition of the relevant standard is IEEE Std 1003.1-2004. The subset we're selecting is compatible with the 2001 version of the same standard, which in turn defers to the ISO C standard for *printf* conventions. Specifically, I'm referencing section 5: "File Notation Conventions".

Formation Rule

The formation rule for a *field_format* value is:

```
%[+|-]width[.precision]specifier
```

where square brackets indicate an optional component, and:

width
is the total potential width of the field (i.e., the width of the widest value occurring in the field)
precision
is the number of digits following the decimal point for real numbers (but is otherwise ignored)
specifier
is exactly one of the characters in the set [doxfes]

Breaking this rule down into separate parts...

%
The format *must* begin with a percent sign ("%").
[+|-]

Either a "+" or "-", but never both. The "-" may be used for string fields, to indicate that the string is (or should be) left-justified in the field. This is actually the preferred way to present most string values in character tables, so the *field_format* value for fields with a data type of *ASCII_String* will nearly always begin with a "-". Similarly for any of the date/time type fields. In PDS4 labels, the "-" prefix is *forbidden* for all numeric fields (integers, floating point numbers, and numbers using scientific notation).

The "+" may be used with numeric fields to indicate that an explicit sign is included in the field for input, and should be displayed on output. In PDS4 labels, the "+" is *forbidden* for string fields.

width

The width is an integer value indicating the maximum number of characters needed for the complete representation of the largest (in terms of display bytes, not necessarily magnitude) value occurring, or potentially occurring, in the field. This should include bytes for signs, decimal points, and exponents. In the case of string values, it should be the maximum width from the first non-blank character to the last non-blank character. It should *not* include bytes for field delimiters, which are not considered part of the field. In character tables, it *must* be the same as the `<field_length>` for scalar fields.

The **width** is separated from the **precision** by a decimal point ("."). If there is no **precision** specified, the decimal point must be omitted.

precision

The precision value is used in three different ways:

1. In real numbers, it indicates the number of digits to the right of the decimal point.
2. In integers, it indicates that the integer will be zero-padded on the left out to the full field width. For example, the value "2" in "%3.3d" format is "002".
3. In strings, it signifies the maximum number of characters from the actual string value that should be printed. (It is possible in programming, for example, to print no more than the first 10 characters from a string, but require that the output field be left-justified and padded with at least 5 blanks by using a specifier of "%15.10s".) In PDS4 archive labels, if a *precision* value is included for a string format, it *must* be equal to the *field width*.

specifier

The specifier indicates the broad data type for display. It will be one of a subset of the conversion specifiers included in the IEEE standard:

d

A decimal integer

o

An unsigned octal integer

x

An unsigned hexidecimal number

f

A floating point number in the format *[-]ddd.ddd*, where the actual number of digits before and after the decimal point are determined by the preceding **width** and **precision** values (note that the **width** includes the decimal point and any sign).

e

A floating point number in the format *[-]d.ddde+/-dd* where "+/-" stands for exactly one character (either "+" or "-"), there is always exactly one digit to the left of the decimal point, and the number of digits to the right of the decimal point is determined by the preceding **precision** value (note that the **width** includes all digits, signs, and the decimal point).

s

A string value. Note that strings should generally be left-justified in fixed width character tables and on output from a binary table, so most *field_format* values ending in "**s**" should begin with "**-**"

Variations on the Theme

The proposal is intended to be a limited subset of the total universe of possible format conversion specifiers. Here are some things that were discussed by the DDWG but not included in the proposed standard, above. If you have an opinion about them, ***NOW*** would be an excellent time to tell me.

"i" conversion specifier

The "i" specifier is identical to the "d" specifier in every way. I chose "d" over "i" because it was mnemonic ("d" for decimal, "o" for octal, "x" for hexadecimal - OK, so it's not a perfect system). The DDWG also preferred it in that it makes it clear that we are using a different format specification system from the PDS3 standards.

"X" and "E" specifiers

These uppercase specifiers only matter on output - they cause the letters in their formats ("a"-"f" for hexadecimal, "e" for reals) to be uppercase rather than lowercase. Programmers outputting values from PDS4 label info can certainly convert the lowercase forms to uppercase easily and unambiguously, so these were omitted to keep the archiving requirements simple and to the point.

"-" for numbers

POSIX allows numbers to be left-justified, but this is problematic for documenting numeric fields in an archive. If you have a field that looks like a number that you want to left-justify, you should treat it as a simple string.

"g", "G" and "c" specifiers

The "g|G" specifier switches back and forth between floating point and exponential notation, depending on the magnitude of the output value. I'm not a big fan of that for archive tables, but perhaps it is useful. The "c" specifier is for I/O on a single character at a time - it cannot take a field width or precision, and the output is a single printable character, *not* the decimal equivalent. I think "%1s" is good enough for us, and the subtle distinction between "%c" and "%1s" is not something I want to spend the next 10 years explaining.