PDS4 Character Data Type Definitions

Following is a glossary of data type definitions for values expressed as strings of characters, extracted from the PDS4 Information Model and master schema. They are used to describe fields defined in local and discipline dictionaries as well as values included in data objects (tables and arrays, for example).

ASCII Representations

ASCII_AnyURI
Use this for fields that are intended to be interpreted as Uniform Resource Identifiers (URIs). PDS restricts these strings to the ASCII character set, so you should URL-encode any non-ASCII characters in your URIs.

ASCII_Boolean
This corresponds exactly to the XML Schema data type of "boolean". Valid values are "true", "false", "1" (one), and "0" (zero).

ASCII_Date_DOY
This data type is identical to the **ASCII_Date** type except that the date *must* be in the day-of-year format.
Usage Note: The date itself is not validated beyond simple numerical ranges, so PDS schema validation will not tell you, for example, that "1999-366" is not a valid date.

ASCII_Date_Time_DOY
This data type is identical to the **ASCII_Date_Time** type, except that the date portion *must* be in the day-of-year format.

ASCII_Date_Time_DOY_UTC
This data type is identical to the **ASCII_Date_Time_DOY** type, except that the value *must* have the **Z** appended to the end to indicate that the value is a UTC date and time.

ASCII_Date_Time_YMD
This data type is identical to the **ASCII_Date_Time** type, except that the date portion *must* be in the year-month-day format.

ASCII_Date_Time_YMD_UTC
This data type is identical to the **ASCII_Date_Time_YMD** type, except that the value *must* have the **Z** appended to the end to indicate that the value is a UTC date/time.

ASCII_Date_YMD
This data type is identical to the **ASCII_Date_YMD** type, except that the date *must* be in the year-month-day format.

ASCII_Directory_Path_Name
Use this data type for path information. It is constrained to use only the ASCII character set.
Usage Note: All paths in PDS4 labels should be specified using Unix-style notation, and should *never* be absolute (so they should never begin with either a device identifier or a slash character). This will also typically be true for paths that appear in archival tables, but check with your PDS node if this presents a problem. The schema validation does *not* enforce these constraints. You should also not assume that fields with this data type include a trailing slash character.

ASCII_DOI
This is a string corresponding to a DOI of the form "10.*string*/*string*", where *string* can be any sequence of one or more non-whitespace characters.

ASCII_File_Name
This data type is a string representing a file name without path information. The characters are constrained to be in the ASCII subset.

Usage Note: Do not assume that any validator will check for file existence unless it specifically claims to do so. Schema validation is simple and will not, for example, tell you that you have included path information (as indicated by the presence of a slash character), or included values that would be problematic for some or all operating systems (like the asterisk or question mark characters).

ASCII_File_Specification_Name
This data type is for file names with path information. It is effectively the concatenation of the **ASCII_Directory_Path_Name** and **ASCII_File_Name**, with an additional slash character as needed. The **Usage Notes** for those data types apply here as well.

ASCII_Integer
This data type is a direct synonym for the XML Schema *xs:integer* data type, so values are constrained to be unbounded integers. You may include a "+" or '-' sign.

ASCII_LID
This data type is intended to hold PDS4 Logical Identifier (LID) values, without version numbers. It is constrained to be at least 14 characters long and to use only ASCII characters.

Usage Note: No format checking is done on these values, so schema validation cannot warn you, for example, that "URN:NASA:PDS:MYBUNDLE" is invalid (because it violates the PDS4 Standards lowercase requirements). Do not assume a data object validator is doing format checking of LID values unless it explicitly claims to.

ASCII_LIDVID
This data type represents the concatenation of a PDS4 Logical Identifier (LID) with a Version Identifier (VID), with a double colon ("::") between them. It is constrained to be at least 19 characters long and to use only ASCII characters.

Usage Note: No format checking is done on these values, so schema validation cannot warn you, for example, that "URN:NASA:PDS:MYBUNDLE::1.0" is invalid (because it violates the PDS4 Standards lowercase requirements). Do not assume a data object validator is doing format checking of LIDVID values unless it explicitly claims to.

ASCII_LIDVID_LID
This data type accepts either **ASCII_LID** or **ASCII_LIDVID** values. See the **Usage Notes** for those data types.

ASCII_MD5_Checksum
Values of this data type must contain exactly 32 hexadecimal digits.

Usage Note: Do not assume that validators will do a checksum check with this value unless they specifically claim to do so.

ASCII_NonNegative_Integer
This data type includes integers in the range 0 and up. You may include a "+" sign, if so moved.

ASCII_Numeric_Base16
This data type is a synonym for the XML Schema type xs:hexBinary. Hex digits above 9 may be upper or lower case.

ASCII_Numeric_Base2
This data type is constrained to contain only the digits '1' and '0'.

Usage Note: There is no base indicator allowed in the value, so there is no way for a user who sees the value to know whether the string "101" is supposed to represent the value 5 in binary, or the value 65 in octal, or the decimal value 101.

ASCII_Numeric_Base8
This data type is constrained to contain only the digits '0' through '7'.
Usage Note: There is no base indicator allowed in the value, so there is no way for a user who sees the value to know whether the string "101" is supposed to represent the value 5 in binary, or the value 65 in octal, or the decimal value 101.

ASCII_Real
This data type is a synonym for the XML Schema type *xs:double*. It accepts values representable in a 64-bit IEEE754 floating point format. It includes simple floating point values as well as exponential notation (i.e., powers of 10), as well as the special constants *INF* for positive infinity, *-INF* for negative infinity, and *NaN* for "Not a Number". Case counts for these special values.
Usage Note: The special constants for +/- infinity and NaN *should not appear* in archival data - either in labels or in data tables. In labels, declare attributes as nil or omit them entirely; in data tables, define a numeric constant to use as a flag for missing data.

ASCII_String
This data type is based on the XML Schema type *xs:token* constrained to the ASCII character set, and corresponds to a non-empty string of ASCII characters (which may include whitespace) of unlimited length. Whitespace should be collapsed on input. This data type is used for describing fields in character tables.

ASCII_Time
This data type is for values that hold a 24-hour clock time in the standard *hh:mm:ss.ssss* format. The string may optionally end in a **Z** to indicate a UTC time. The string may be truncated at the appropriate point for the actual precision; omit the ':' separator when there is no value to the right of it. Both *00:00* and *24:00* are valid values.

ASCII_VID
This data type corresponds to a PDS4 Version Identifier (VID). It is a two-part version number of the form *N.n*, where both *N* and *n* are present and non-negative. The major version number (*N*) may be zero, but may not contain leading zeroes for values greater than zero. So "0.1" is valid, but "01.1" is not.

UTF-8 Representations

UTF8_String
This data type is based on the XML Schema type *xs:token* and corresponds to a non-empty string of UTF-8 characters (which may include whitespace) of unlimited length. Whitespace should be collapsed on input. This data type is used for describing fields in character tables. Note that UTF-8 characters may be more than one byte long. Care should be taken when dealing with UTF-8 data in fixed-width tables to ensure that "bytes" and not "characters" are used to calculate locations and value lengths.