Errata for Classes of Implementation-Specific Behavior

# Introduction

There are two classes of implementation-specific behavior in this specification.

[Definition: An **implementation-defined** feature is one where the implementation has discretion in how it is performed, and the implementation must document how it is performed.]

[Definition: An **implementation-dependent** feature is one where the implementation has discretion in how it is performed, but the implementation is not required to document how the feature is performed.]

This document lists all implementation-defined and implementation-dependent features in DFDL 1.0. Specific wording changes in the document are highlighted in yellow below.

## Implementation-defined

* "additional mechanisms for dealing with effective processing errors, such as the means of specifying retry points or the means of skipping some data so as to recover from the error in some way" (Section 2.3)
* Implementation Limitations that result in a Schema Definition Error
  + includes size/schema complexity limits
  + Implementation must specify what happens when it encounters DFDL schema constructs that it doesn't support.
  + References:
    - Section 2.6 Specific Errors Classified
* Implementation Limits that result in Processing Errors
  + References:
    - Section 3 Glossary, "Array"
    - Section 2.6 Specific Errors Classified
      * "Implementation Limit Errors - Implementations can have fixed or adjustable limits that some formats and some data may exceed at processing time. This specification does not further specify what these errors are, but some possible examples are:
        + Data longer than allowed for representation of a given data type

Example: exceed maximum length of representation of xs:decimal when dfdl:representation is "text".

* + - * + Expression references too far back into infoset (parsing)
        + Expression references too far forward into infoset (unparsing)
        + Number of array elements exceeds limit.
        + Regular expression exceeds time limit”
    - Section 12.3.7.2.1 Maximum length in bits of xs:nonNegativeInteger, xs:integer, and xs:decimal data types.
    - Section 12.3.7.2.4 Maximum supported length in bytes for calendar types when dfdl: binaryCalendarRep is 'packed', 'bcd', or 'ibm4690Packed'.
      * Note: See Action Item #252 on clarifying length of binary packed calendar types.
    - Section 12.3.7.2.5: “The maximum specified length of a packed decimal is implementation defined.”
      * Editorial change: Modify “implementation defined” to “implementation-defined” (make it hyphenated)
    - Section 13.6.1.1: “The term *maximum integer digits* is a limit that is implementation dependent, but must be at least 20 (which is the number of digits in a base 10 unsigned long).”
      * Minor change: Modify “implementation dependent” to “implementation-defined”.
    - Section 13.7: “Note that the maximum allowed value for twos-complement and unsigned base-2 binary integers is implementation dependent, but must be at least that of a xs:long type, which is the equivalent of an 8 byte/64-bit signed integer.”
      * Minor change: Modify “implementation dependent” to “implementation-defined”.
* Additional non-standard character sets supported by the implementation, along with the alignment for each. References:
  + Section 3, Glossary, "Character Set Encoding": "The DFDL standard allows for implementation-specific character set encodings to be supported"
  + Section 11, Properties Common to both Content and Framing, encoding property
  + Section 12.1.2, Mandatory Alignment for Textual Data
* Mechanism to specify external properties
  + References
    - Section 7.7: "The external property is optional. If not specified it takes the default value 'false'. If true the value may be provided by the DFDL processor and this external value will be used as the global default value (overriding any defaultValue specified on the dfdl:defineVariable). The mechanism by which the processor provides this value is unspecified and implementation specific."
    - Section 18
* “Implementations of DFDL may provide control mechanisms for limiting the speculative search behavior of DFDL parsers” (Section 9.1). Any such control mechanisms must be documented by the implementation and are thus implementation-defined.
* How to specify the distinguished root node of the schema
  + References
    - Section 9.1: “The logical parser recursively descends the DFDL schema beginning with the element declaration specified (in an implementation specific manner, see Section 18) of the *distinguished root node* of the schema passed to the DFDL processor.”
    - Section 18
* Behavior on an unsuppressed processing error
  + References
    - Section 9.3.3.1 “The behavior of a DFDL processor on an unsuppressed processing error is not specified, but it is allowable for implementations to abort further parsing.“
  + Reasoning
    - While aborting the parse is probably the most prevalent approach, if an implementation were to do something different, certainly that implementation should document how it handles unsuppressed processing errors, as it would have a huge impact on how errors are handled. For example, should a parser attempt to do error-correction on an unsuppressed processing error, that behavior should definitely be documented. Hence, I think this behavior on an unsuppressed processing error should be “implementation-defined”.
* Maximum specified length for binary calendar elements when dfdl:binaryCalendarRep is a packed decimal value.
  + Reference: Section 12.3.7.2.4 Length of Binary Calendar Elements.
  + See corresponding public comment and WG minutes for more details.
* Which optional features are supported by the implementation and which are not (section 21)
* Implementation-specific limitations on the use of forward and backward references in DFDL expressions.
  + Reference: Section 23.1 Expression Language Data Model
  + "Implementations may have specific limitations on the use of forward or backward reference, or may provide controls for bounding the reach of such references."

## Implementation-dependent

* Error messages used when a processing error or schema definition error occurs while evaluating a message expression in a dfdl:assert annotation, as explained in section 7.3.1, Properties for dfdl:assert:
  + "If a processing error or schema definition error occurs while evaluating the message expression, a recoverable error is issued to record this error containing implementation-defined content, then processing of the assert continues as if there was no problem and in a manner consistent with the failureType property, but using an implementation-defined substitute message."
  + Change: "implementation-defined" in above paragraph to "implementation-dependent".
* Error messages used when a processing error or schema definition error occurs while evaluating a message expression in a dfdl:discriminator annotation, as explained in section 7.4.1, Properties for dfdl:discriminator:
  + "If a processing error or schema definition error occurs while evaluating the message expression, a recoverable error is issued to record this error containing implementation-defined content, then processing of the discriminator continues as if there was no problem, but in the case of failure using an implementation-defined substitute message."
  + Change: "implementation-defined" in above paragraph to "implementation-dependent".
* section 2.7 - implementation-dependent checks on validity of properties that don't apply or that have invalid values
  + optional warning if property not being used has invalid value..... - result of WG meeting 11/26/2013.
* Number of Unicode Replacement Characters inserted when then data contains more than one adjacent decode error.
  + Reference: Section 11.2.1.2
  + No changes to document needed