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 contains errata to clarify between implementation-defined and implementation-dependent in DFDL 1.0. In addition, this errata document will contain a new appendix to be added to the specification that lists implementation-defined features. Specific wording changes in the document are highlighted in yellow below.

##  Implementation-defined

*(Note: The below list of implementation-defined features aims to be a complete list. From this list, a separate list of changes and a new appendix will be generated.)*

* "It is expected that DFDL implementations will provide and define 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 (Section 2.6 Specific Errors Classified)
	+ Implementation must specify what happens when it encounters DFDL schema constructs that it doesn't support.
	+ includes size/schema complexity limits
	+ Rename the corresponding bullet in the schema definition error section
		- “Implementation-defined Limitations”
* Implementation Limits that result in Processing Errors
	+ Section 3 Glossary, "Array"
		- “If XSD property maxOccurs is 'unbounded' then there is no constraint to the maximum number of occurrences, though implementations may ~~have~~ define maximum capabilities.”
	+ Section 12.3.7.2.4 Maximum supported length in bytes for calendar types when dfdl: binaryCalendarRep is 'packed', 'bcd', or 'ibm4690Packed'.
		- No changes needed to specification.
* 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~~defined character set encodings to be supported"
	+ Section 11, Properties Common to both Content and Framing, encoding property
		- “Implementations may ~~allow~~define additional implementation-specific encoding names only for character set encodings for which there is no IANA name standard nor CCSID standard. These implementation-~~specific~~defined encodings must have "x-" as a prefix to their name, and they are subject to being superceded with DFDL standard names in future versions of the specification.”
	+ Section 12.1.2, Mandatory Alignment for Textual Data
		- “~~Some i~~Implementations may ~~include~~define additional encodings which have other alignments. The DFDL standard specifies a name for one such character set encoding, although conforming implementations are not required to support this encoding:
			* US-ASCII-7bit-packed: the alignment is 1-bit (textual data in this encoding may appear on any bit boundary, i.e., no byte alignment is required).

~~Implementations may also provide identifiers for non-standard encodings, and these will have their own specific alignments as well.~~ Similarly, each implementation must define a name (see “encoding” property in Section 11) and the alignment for each non-standard encoding that it supports.”

* 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-defined ~~specific~~."
		- Section 18
* “Implementations of DFDL may provide and define 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-defined ~~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; any other behavior must be defined by the implementation.“
	+ 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”.
* Number of fractional second digits supported
	+ “Any number of fractional seconds "S" may by specified in the pattern and accepted by implementations, but an implementation is free to represent a limited number of fractional seconds internally. Excess fractional seconds are truncated, not rounded up. At least millisecond accuracy must be implemented.” (Section 13.11.1)
	+ “The number of fractional second digits supported is implementation dependent but must be at least one.” (Section 13.11.1)
	+ Corrections
		- “The number of fractional second digits supported is implementation-defined but must be at least three.”

## Implementation-dependent

* Implementation Limits that result in Processing Errors
	+ References:
		- Section 12.3.7.2.1 Maximum length in bits of xs:nonNegativeInteger, xs:integer, and xs:decimal data types.
			* Editorial change: Modify “Implementation defined” to “Implementation dependent”
		- Section 12.3.7.2.5: “The maximum specified length of a packed decimal is implementation defined.”
			* Editorial change: Modify “implementation defined” to “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".