# Planning the Implementation Phases for Daffodil

2012-11-07

Mike Beckerle

This document describes a breakdown of features for DFDL to try to rationalize the phasing of development activities in the Daffodil open-source project.

Tier 1 is anything needed for *parsing* most legacy and contemporary textual and binary data formats. It also includes

* Good diagnostic behavior so that real models can be created and debugged/tested.
* Sufficient (though initial) attention to performance to enable realistic formats to be developed, tested, and used.
* XML convertibility extensions (allow DFDL-described data to be converted to legal XML)

The Tier 1 functionality would be complete enough to be released as an initial usable and supportable product.

Tier 2 is large things we think we can get away with deferring temporarily. By temporarily we mean for many months at least while we focus on things we deem higher priority. Examples:

* Unparsing
* Broader library of character set encodings (beyond the bare minimum)
* Easily worked-around features: unordered sequences, floating elements are examples.

We've done enough research to know that unparsing is completely achieveable, so it is simply a matter of priority and a rather large QA investment. There are also some features which add no new expressive power to DFDL. They are simply there to provide convenience in expressing certain formats.

Tier 3 is things we believe will not be needed unless very specific applications/formats are undertaken. Basically anything to do with

* Legacy IBM Mainframe data types and Cobol-oriented data.
* packed/zoned decimal and bcd
* certain Nil-value idioms (nilKind='literalCharacter')
* subtle mixing of binary data with delimiters (scanning for delimiters over packed decimal data)
* Rich internationalization such as bi-directional text (needed for many middle eastern, south and central asian languages), and rich legacy character encodings such as the older pre-Unicode Asian language encodings.

Eventually, all the functionality at all 3 tiers is needed. The Tier structure is simply a set of guidelines to control the complexity and keep a team effort focused and making progress, and it also stages our need for certain specialized knowledge/skills out to Tier 3.

The largest body of work and the one requiring far more detail in planning is everything associated with Tier 1. So we break that down into much greater detail below.

## Tier 1: Major Functionality Areas & Timeframe Targets

Within Tier 1, there are a number of feature areas still to be implemented (as of this writing 2012-11-01). This table is a rough breakdown and a target timeframe for implementation. This is one set of inputs to the organization of our work into target spins, the specific goals/themes of which are beyond the scope of this document, and subject to more rapid evolution.

These timeframe targets of course depend on other factors – time spent fixing bugs, time spent on performance work, diagnostic-message quality, and other areas of investment of time. (i.e., they're goals/targets, not estimates.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Topic | Description | Timeframe  30-60 days | Timeframe  60-120 days | Timeframe 120-180 days |
| Infrastructure |  |  |  |  |
| * Multiple-file Schemas | The ability to divide a DFDL schema into multiple schema files that include/import each other | x |  |  |
| * Improved diagnostic messages | Diagnostic messages need to identify the location of the error, and relevant part of the schema (by file, line, character ideally) | x |  |  |
| Data Types | The implmentation of each data type is split into a basic part, which introduces the type, and additional topics which address the larger effort of introducing the rich properties for complex format control. |  |  |  |
| * Date, Time, DateTime |  |  | x |  |
| * Boolean |  |  | x |  |
| Textual Data |  |  |  |  |
| * LengthKind Pattern | Partially implemented now |  | Complete |  |
| * LengthKind EndOfData |  |  |  | x |
| * Defaulting |  |  | x |  |
| * Trimming (aka Padding removal) |  | x |  |  |
| * Separator Suppression |  |  | x |  |
| * Nil values – logicalValue | Applies to binary data also. Logical values don't care if they're from text or binary. |  |  | x |
| Binary Data |  |  |  |  |
| * LengthKind Explicit(in bits) |  | x |  |  |
| * LengthKind EndOfData |  |  |  | x |
| * Bit alignment |  | x |  |  |
| * Expressions involving dfdl:occursIndex() | Mostly already implemented. | x |  |  |
| * Defaulting |  |  | x |  |
| * bit-packed ASCII 7-bit |  | x |  |  |
| Expression Language |  |  |  |  |
| * Functions beyond dfdl:position |  | A few | More | All (except those intended for unparsing) |
| * Expression compiler | Enforces DFDL semantics on the Xpath expressions and the JDOM/Saxon engine used inside Daffodil. |  | Partial | Complete |
| Rich Text Data Types | A large number of properties are very specific to text representations of certain data types. Implemenatation of all these properties for the corresponding types, along with testing for them. |  |  |  |
| * Text Calendar | (YYYYmmmDD HH:mm:SS.ssss that sort of thing.) |  |  | x |
| * Text Numbers | “+##99V99E+99”, textNumberZeroRep, textNumberGroupingSeparator, etc. |  |  | x |
| * Boolean text properties | TextBooleanTrueRep etc. |  |  | x |
| Rich Binary Data Types |  |  |  |  |
| * Boolean type |  |  |  | x |
| * Calendar types |  |  |  | x |
| Complex Types |  |  |  |  |
| * Nils |  |  |  | x |
| * Choice – remove restrictions | Current choice implementation has some restrictions it enforces. (All children must be elements for example.) |  | x |  |
| * Choice - length |  |  |  | x |
| Infoset |  |  |  |  |
| * Validation: Pattern and Enumeration Facets, max/min Occurs |  | x |  |  |
| * Validation: other Facets |  |  |  | x |
| * Conversion/Adaptation for XML Infoset |  |  | x |  |
| * Hidden Groups |  |  | x |  |
| Other |  |  |  |  |
| * Dynamic Delimiters |  |  |  | x |
| * Dynamic Byte Order |  |  |  | x |
| * Dynamic Number Format Properties |  |  |  | x |
| * OccursCountKind Stop Value |  |  |  | x |
| * Prefixed Element Length |  |  |  | x |
| * Choices with Dispatch |  |  |  | x |
| * Raw-bytes entities |  |  |  | x |

## Proposed Interfaces and Extensions to DFDL

As part of the Daffodil effort we anticipate a few extensions to DFDL.

Any given extension is to address a real need, most would be experiments which would be submitted for consideration to include in a future version of the DFDL standard.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Detection** | Tier 1 | Tier 2 |
| XML Infoset Strings | XML interchange is a critical use case. A feature allowing compatible translation of strings from the DFDL Infoset (which allows XML-illegal characters) into the XML Infoset is needed.  The special daffodil extension property daffodil:encodingModifiers=”translateIllegalXMLCharstoPUA” specifies a safe transformation of characters that are not allowed in XML into characters that are allowed, without loss of information content. | x |  |
| Debug Annotations | Ability to annotate a schema with control annotations that help turn on/off debugging assistance on narrower regions of the data and associated schema. | x |  |
| Error Recovery Annotations | Many applications need a way to specify what to do with non-conforming data. This would be to develop a feature that could then be proposed as an addition to DFDL for a future revision of the spec. (DFDL v2.0) | x |  |
| Layered Translation | Ability to create a multi-layer schema where strings or byte-arrays from one DFDL schema are indirectly used as the representation layer for elements that are also described in the same schema |  | x |
| Transformation Hooks | Ability to embed some richer data transformations directly into the same schema so as to create a parse output Infoset that is more closely aligned with application needs. |  | x |
| Recursive definitions | Extension to the DFDL spec (which would be proposed as a DFDL v2.0 feature). This is desriable to properly model some more document-centric formats. |  | x |

## Appendix: DFDL Specification 'Optional' Features Priority

This table addresses the list of features deemed 'optional' by the DFDL Specification.

The point of this appendix is to put an end to using that part of the DFDL specification as some sort of priority guideline. We show below how what the specification deems 'optional' cuts across our Tier1, Tier2, Tier3 priority organization.

The quick summary of this table: The vast majority of feature areas are important and are “Tier 1”.

Very few features can be postponed in their entirety which is also to say that this table is much too coarse to really be using for planning purposes as far as the approach to organizing the building out of DFDL functionality.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **Detection/Description** | Tier 1 | Tier 2 | Tier 3 |
| Validation | External switch. Warnings are generated about checks against XSD Schema facets, and minOccurs/maxOccurs constraints | x |  |  |
| Named Formats | dfdl:defineFormat or dfdl:ref | x |  |  |
| Choices | xs:choice in xsd | x |  |  |
| Arrays where size not known in advance | dfdl:occursCountKind 'implicit', 'parsed', 'stopValue' | X (all but stopValue) | X (stopValue) |  |
| Expressions | Use of a DFDL expression in any property or attribute value | x |  |  |
| End of parent | dfdl:lengthKind = "endOfParent" | x |  |  |
| Simple type restrictions | xs:simpleType in xsd | x |  |  |
| Text representation for types other than String | dfdl:representation="text" for Number, Calendar or Boolean types | x |  |  |
| Delimiters | dfdl:separator <> "" or dfdl:initiator <> "" or dfdl:terminator <> "" or dfdl:lengthKind="delimited" | x |  |  |
| Nils | xs:nillable='true' in xsd | X (except nilKind literalCharacter) |  | X (nilKind literalCharacter) |
| Defaults | xs:default or xs:fixed in xsd | x |  |  |
| Bi-Directional text | dfdl:textBiDi='yes' |  |  | x |
| Lengths in Bits | dfdl:alignmentUnits='bits' or dfdl:lengthUnits='bits' | x |  |  |
| Delimited lengths and representation binary element | dfdl:representation='binary' (or implied binary) and dfdl:lengthKind='delimited' |  | x |  |
| Regular expressions | dfdl:lengthKind='pattern',  dfdl:assert with dfdl:testkind 'pattern' ,  dfdl:discriminator with dfdl:testkind 'pattern' | x |  |  |
| Zoned numbers | dfdl:textNumberRep='zoned' |  |  | x |
| Packed numbers | dfdl:binaryNumberRep='packed' |  |  | x |
| Packed calendars | dfdl:binaryCalendarRep='packed' |  |  | x |
| S/390 floats | dfdl:binaryFloatRep='ibm390Hex' |  |  | x |
| Unordered sequences | dfdl:sequenceKind='unordered' |  | x |  |
| Floating elements | dfdl:floating='yes' |  | x |  |
| dfdl functions in expression language | dfdl:functions appear in expression | x |  |  |
| Hidden groups | dfdl:hiddenGroupRef <> '' | x |  |  |
| Calculated values | dfdl:inputValueCalc <> '' or dfdl:outputValueCalc <> '' | x |  |  |
| Escape schemes | dfd:defineEscapeScheme in xsd | x |  |  |
| Extended encodings | Any dfdl:encoding value beyond the core list | X (will need ascii-7-bit packed encoding) | X (others) |  |
| Asserts | dfdl:assert in xsd | x |  |  |
| Discriminators | dfdl:discriminator in xsd | x |  |  |
| Prefixed lengths | dfdl:lengthKind='prefixed' |  | x |  |
| Variables | dfdl:defineVariable,  dfdl:newVariableInstances,  dfdl:setVariable  Variables in DFDL expression language  Note that variables as a feature is dependent on the Expressions feature. | X (all but newVariableInstance) | X (newVariableInstance) |  |
| BCD calendars | dfdl:binaryCalendarRep="bcd" |  |  | x |
| BCD numbers | dfdl:binaryNumberRep=”bcd” |  |  | x |
| Multiple schemas | xs:include or xs:import in xsd | x |  |  |