



— Lightning Round —
DITA Specialization
Support:
It Should Just Work

Specialization Is A Key Value DITA Brings

- ◆ Enables blind interchange knowing only that partners use DITA
- ◆ Enables satisfaction of local requirements
- ◆ Use of specialization by itself does not require change to existing processing
- ◆ NOTE: There is *no* standard or reliable public or system ID that indicates a DITA document: Every user should have their own local shell DTDs with own their identifiers

All DITA Users Will Use Specialization

- ◆ It's easy to create specializations
- ◆ Specializations will flower over time
 - ◆ Local specializations
 - ◆ Community-specific specializations
 - ◆ New core specializations
- ◆ Realizes the full value of an investment in DITA

All DITA-Aware Tools Should Just Work

- ◆ My opinion: All DITA-supporting tools must support specialization to be considered valid DITA implementations
- ◆ Use of specializations should not impose any additional cost
- ◆ No “specialization tax”

How Is This Possible?

- ◆ DITA documents are unambiguously self describing
- ◆ DITA architecture ensures base level of consistency
 - ◆ In documents
 - ◆ In schemas and DTD declarations
- ◆ DITA class= attribute defines type hierarchy for all element types
- ◆ For valid DITA documents, all element types must be mapped to a base type

DITAArchVersion= Attribute is the Key

- ◆ Occurs on all map and topic element types
- ◆ Is in an OASIS-defined namespace
- ◆ If attribute is present, must be a DITA document
- ◆ If attribute is not present, no obligation to assume document is a DITA document
- ◆ Value and occurrence independent of use of specialization

Thus: You Can Always Know It's DITA

- ◆ Tools can reliably recognize DITA documents
- ◆ Without prior exposure to or knowledge of the documents' DTDs or schemas

So Do the Right Thing

- ◆ Therefore tools should recognize DITA documents...
- ◆ ...and apply appropriate processing automatically:
 - ◆ Presentation styles
 - ◆ Import/export processing
 - ◆ Query
 - ◆ etc.

No Required Schema-Specific Configurations

- ◆ No requirement for per-DTD or per-schema configurations
- ◆ Can apply some sort of default configuration automatically
- ◆ However, need to allow for schema-specific configurations when needed

Obligations of Users

- ◆ Valid DITA doctypes
 - ◆ Must reflect DITA architecture for declaration organization and structure
 - ◆ Must be valid specializations
- ◆ Documents must be valid to their declarations
- ◆ This should be the normal case

Only Bootstrap Requirement: Catalogs

- ◆ For tools to “just work” they must have a mapping from public IDs and absolute system IDs to local files

Catalogs Are Required

- ◆ DITA Open Toolkit effectively requires the use of public or absolute system IDs...
- ◆ ...Because of the way it generates and processes intermediate files
- ◆ It cannot process documents with relative pointers to their declarations
- ◆ But that's OK because you should use absolute system IDs anyway

Thus: Catalogs Required

- ◆ Therefore, will always need a catalog
- ◆ No general convention that allows finding of catalogs

Thus: Tools must be given appropriate catalog

- ◆ One-time configuration for any DITA-aware tool: which catalog to use to resolve DTDs and schemas
- ◆ Ideally should be able to use master catalog for DITA Open Toolkit

Optional Setup: DITA OT Location

- ◆ Tools may also make use of the DITA Open Toolkit (e.g., processing from inside editors or CMS systems)
- ◆ May need to tell tool where the Toolkit is installed
- ◆ Tools should allow use of pre-existing Toolkit installations
- ◆ Not a hard requirement for “just working”

Adding Support for Specialized Types Should be Incremental

- ◆ Some specializations will require custom processing
- ◆ Adding such processing should be incremental on top of base DITA features
- ◆ Implies appropriate extension points and modularity of base features

Message to Tool Implementors

- ◆ All processing must be based on class= attribute values
- ◆ Use DITAArchVersion= attribute to automatically identify DITA documents
- ◆ Must design configuration and customization mechanisms carefully to allow incremental extension from base functions

Expected CMS Features

- ◆ Given a DITA DTD or schema:
 - ◆ Import and configuration of the schema (if required at all) should be a one-step process.
 - ◆ Configuration of DITA-specific metadata handling should be automatic
- ◆ Given a DITA map:
 - ◆ Import of map and all topics should be a one-step process
 - ◆ Map should be accurately reflected within CMS

Expected CMS Features (cont.)

- ◆ Be able to find DITA elements by any type in class hierarchy
- ◆ Be able to query efficiently and easily on DITA-defined metadata (e.g., all specializations of `<data>`)
- ◆ Export map and all topics as single-step action
- ◆ Optional:
 - ◆ Create or modify maps using CMS UIs
 - ◆ Set map context for operations

To Sum Up

- ◆ DITA makes it possible to unambiguously identify DITA documents
- ◆ DITA makes it possible to unambiguously map specialized elements to DITA-defined base types
- ◆ Therefore can always apply default processing to DITA documents
- ◆ Therefore DITA should just work in all tools claiming DITA support
- ◆ If a given tool doesn't just work, you have to ask “why not”?

