DITA 2.0 proposed feature #361: Split syntax and programming domains

Split syntax diagram from programming domain so that they can be used (and supported) independently.

Date and version information

Include the following information:

Date that this feature proposal was completed
June 30 2020

Champion of the proposal
Robert D Anderson

Links to any previous versions of the proposal
Original email to list: https://lists.oasis-open.org/archives/dita/202006/msg00038.html

Links to minutes where this proposal was discussed at stage 1 and moved to stage 2
June 30, 2020: link TBD

Reviewers for Stage 2 proposal
Kristen James Eberlein

Links to e-mail discussion that resulted in new versions of the proposal
N/A

Link to the GitHub issue
https://github.com/oasis-tcs/dita/issues/361

Original requirement or use case

Presented June 30, 2020: split syntax diagram domain from programming domain for easier maintenance and support, and to simplify the process for hiding syntax diagram elements by those who do not need them. The end result will be the existing programming domain (unchanged apart from removal of these elements), and a new syntax diagram domain (containing the 15 elements used for syntax diagram markup).

Use cases

Splitting the domains has the following advantages:

- Authors that use some of the programming domain elements but do not need the 15 syntax diagram elements can more easily remove them. With DITA 1.x this required the use of a constraint, but with this proposal the domain is simply not included in the shell.
- The grouping is more logical. The syntax diagram markup is made of the one diagram container `<syntaxdiagram>`, along with 14 nested elements that can only ever be used inside of the diagram. This is over half of the current programming domain for a set of elements that most authors do not use. Placing them in a separate domain clarifies the intent, that these are **only** related to syntax diagram markup.
- Few applications today support rendering of syntax diagram markup as an actual diagram. To my knowledge, the only tool that has been maintained within the last few years is this one, at github: svg-syntaxdiagrams. Separating the domains allows applications to accurately claim full support for the programming domain, even if they do not have any support for diagrams.

New terminology

N/A

Proposed solution

- The following 15 syntax diagram elements will be moved to a new "Syntax diagram domain": syntaxdiagram, groupseq, groupchoice, groupcomp, fragment, fragref, synblk, synnote, synnoteref, kwd, var, oper, delim, sep, repsep
- Content models and attribute sets are unmodified, apart from the `@class` attribute. The domain token for these elements will switch from `pr-d/NAME` to `syntaxdiagram-d/NAME`.
- File and entity names for these elements will match current patterns for domains.
- The 11 remaining elements in the programming domain are unmodified.

Benefits

Address the following questions:

Who will benefit from this feature?

Information architects, authors, and tool implementors.

What is the expected benefit?

- Information architects can more easily exclude this set of elements.
- Authors using those custom shells will not see these elements in insert-markup lists, reducing confusion about elements (such as `<kwd>` vs the base element `<keyword>`).
- Tool implementors can now claim support for the programming domain, without needing to develop support for rendering syntax diagrams.

How many people probably will make use of this feature?

Many.

How much of a positive impact is expected for the users who will make use of the feature?

Minor positive impact.

Technical requirements

Adding new elements or attributes

Adding a domain

In the OASIS shells, all shells that currently use the programming domain will also get the syntax diagram domain.

Adding an element

N/A

Adding an attribute

N/A

Renaming or refactoring elements and attributes

Renaming or refactoring an element

All syntax diagram elements will have their domain token modified to use `syntaxdiagram-d/` in place of `pr-d/`. All other information about the element (names, ancestry, content models, and attribute lists) are unaffected.

Renaming or refactoring an attribute

N/A

Removing elements or attributes

Removing a topic or map specialization

N/A

Removing a domain entirely, or from select document types

N/A

Removing an element

Elements will be removed from the programming domain, but re-integrated using the syntax diagram domain.

Removing an attribute

N/A

Processing impact

Tools with any support for the syntax diagram names today will need to be updated to recognize new class attribute tokens. The change has no other processing impact; tools that do not support syntax diagrams today are unlikely to add that support (with or without this proposed change).

Overall usability

No impact to authors.

Backwards compatibility

DITA 2.0 is the first DITA release that is open to changes affecting backwards compatibility. To help highlight any impact, does this proposal involve any of the following?

Was this change previously announced in an earlier version of DITA?

No.

Removing a document type that was shipped in DITA 1.3?

No.

Removing a domain that was shipped in DITA 1.3?

No.

Removing a domain from a document type shell that was shipped in DITA 1.3?

No, but it is splitting one domain into two, so some elements will be dropped without an update to shells.

Removing or renaming an element that was shipped in DITA 1.3?

No.

Removing or renaming an attribute that was shipped in DITA 1.3?

No.

Changing the meaning of an element or attribute in a way that would disallow existing usage?

No.

Changing a content model by removing something that was previously allowed, or by requiring something that was not?

No.

Changing specialization ancestry?

Yes; all syntax diagram elements will now use `syntaxdiagram-d/` where they previously used `pr-d/`

Removing or replacing a processing feature that was defined in DITA 1.3?

No.

Are element or attribute groups being renamed or shuffled?

The programming domain is being split into two groups.

Migration plan

If the answer to any question in the previous section is "yes":

Might any existing documents need to be migrated?

It is possible, though unlikely, that documents save class attributes in the content file. Those class attributes would need to be updated.

Might any existing processors or implementations need to change their expectations?

Yes, they need to add support for the new `@class` attribute token for these already-defined elements. Otherwise no change.

Might any existing specialization or constraint modules need to be migrated?

- If anyone has specialized syntax diagrams, the ancestry would need to change in those specialized elements. This is considered extremely unlikely.
- Constraint modules that remove the syntax diagram markup can be discarded in favor of simply not including the new domain. If the constraint also modifies other programming domain behavior, then it can be simplifed by removing any portion related to diagrams.
- Authors that use custom shells, and wish to continue using syntax diagrams, will need to add this domain to their shell along with other OASIS domain modules.

If no migration need is anticipated, why not?

N/A

Costs

Outline the impact (time and effort) of the feature on the following groups.

Maintainers of the grammar files

Small amount of time to split the domain files.

Editors of the DITA specification

- A new topic will be required to group the new domain elements.
- The syntax diagram element topics will need very minor edits to correct the domain inheritance.

Vendors of tools

Tools will need to be updated to recognize the new domain tokens alongside the old domain tokens.

DITA community-at-large

- Will this feature add to the perception that DITA is becoming too complex? No
- Will it be simple for end users to understand? Yes
- If the feature breaks backwards compatibility, how many documents are likely to be affected, and what is the cost of migration? Only impact is to grammar files, content files should be unaffected

Producing migration instructions or tools

- How extensive will migration instructions be, if it is integrated into an overall 1.3 # 2.0 migration publication or white paper? Minimal; search-and-replace for 15 class attribute tokens.
- Will this require an independent white paper or other publication to provide migration details? No
- Do migration tools need to be created before this change can be made? If so, how complex will those tools be to create and to use? No

Examples

The class attributes for these elements are currently:

```
<!ATTLIST  syntaxdiagram class CDATA "+ topic/fig pr-d/syntaxdiagram ">
<!ATTLIST  delim        class CDATA "+ topic/ph pr-d/delim ">
<!ATTLIST  fragment     class CDATA "+ topic/figgroup pr-d/fragment ">
<!ATTLIST  fragref      class CDATA "+ topic/xref pr-d/fragref ">
<!ATTLIST  groupchoice  class CDATA "+ topic/figgroup pr-d/groupchoice ">
<!ATTLIST  groupcomp    class CDATA "+ topic/figgroup pr-d/groupcomp ">
<!ATTLIST  groupseq     class CDATA "+ topic/figgroup pr-d/groupseq ">
<!ATTLIST  kwd          class CDATA "+ topic/keyword pr-d/kwd ">
<!ATTLIST  oper         class CDATA "+ topic/ph pr-d/oper ">
<!ATTLIST  repsep       class CDATA "+ topic/ph pr-d/repsep ">
```

```
<!ATTLIST  sep          class CDATA "+ topic/ph pr-d/sep ">
<!ATTLIST  synblk       class CDATA "+ topic/figgroup pr-d/synblk ">
<!ATTLIST  synnote      class CDATA "+ topic/fn pr-d/synnote ">
<!ATTLIST  synnoteref   class CDATA "+ topic/xref pr-d/synnoteref ">
<!ATTLIST  var          class CDATA "+ topic/ph pr-d/var ">
```

After the update they will be:

```
<!ATTLIST  syntaxdiagram class CDATA "+ topic/fig syntaxdiagram-d/syntaxdiagram ">
<!ATTLIST  delim        class CDATA "+ topic/ph syntaxdiagram-d/delim ">
<!ATTLIST  fragment     class CDATA "+ topic/figgroup syntaxdiagram-d/fragment ">
<!ATTLIST  fragref      class CDATA "+ topic/xref syntaxdiagram-d/fragref ">
<!ATTLIST  groupchoice  class CDATA "+ topic/figgroup syntaxdiagram-d/groupchoice ">
<!ATTLIST  groupcomp    class CDATA "+ topic/figgroup syntaxdiagram-d/groupcomp ">
<!ATTLIST  groupseq     class CDATA "+ topic/figgroup syntaxdiagram-d/groupseq ">
<!ATTLIST  kwd          class CDATA "+ topic/keyword syntaxdiagram-d/kwd ">
<!ATTLIST  oper         class CDATA "+ topic/ph syntaxdiagram-d/oper ">
<!ATTLIST  repsep       class CDATA "+ topic/ph syntaxdiagram-d/repsep ">
<!ATTLIST  sep          class CDATA "+ topic/ph syntaxdiagram-d/sep ">
<!ATTLIST  synblk       class CDATA "+ topic/figgroup syntaxdiagram-d/synblk ">
<!ATTLIST  synnote      class CDATA "+ topic/fn syntaxdiagram-d/synnote ">
<!ATTLIST  synnoteref   class CDATA "+ topic/xref syntaxdiagram-d/synnoteref ">
<!ATTLIST  var          class CDATA "+ topic/ph syntaxdiagram-d/var ">
```