Stage 3 proposal: Feature #13056

Contents

Chapter 1. Stage 3 proposal: Feature #13056

Add the ability to group information within conditional processing attributes

Champion

Robert D Anderson

Tracking information

Event	Date	Links
Stage 1 proposal accepted	Jan 3 2012	https://www.oasis-open.org/committees/download.php/44749/minutes20120103.txt
Stage 2 proposal submitted	Feb 28 (first draft), March 8, 2013 (second draft)	Second draft - HTML: https://www.oasis-open.org/committees/download.php/48508/13056-AttributeGroup-Stage2.html; DITA: https://www.oasis-open.org/committees/document.php?document_id=48507
Stage 2 proposal discussed	March 5, 2013	Merged: https://www.oasis-open.org/committees/download.php/49823/minutes2013-q12.txt
Stage 2 proposal approved	March 12, 2013	Merged: https://www.oasis-open.org/committees/download.php/49823/minutes2013-q12.txt
Stage 3 proposal submitted to reviewers		Deb Bissantz, Dick Hamilton, Kris Eberlein, Eliot Kimber
Stage 3 proposal (this document) submitted		

Approved technical requirements

Allow the four original property attributes (audience, platform, product, and otherprops) to accept grouped values, using the same syntax for generalized attribute values within @props and @base. Each group is to be treated as an independent attribute for the purposes of filtering. For eample, the groups "appserver" and "db" could be used within @product to create two sets of product conditions, one for related databases and another for related application servers. This allows content to be filtered out when all listed databases are filtered, without regard to specified application servers.

Dependencies or interrelated proposals

None.

Modified DTDs

No change to DTDs / Schemas.

Modified specification documentation

This update results in several changes to the topic http://docs.oasis-open.org/dita/v1.2/os/spec/archSpec/condproc.html. That topic is rendered below with revisions marked.

The following changes will be made to http://docs.oasis-open.org/dita/v1.2/os/spec/archSpec/conditional-processing-attributes.html:

DITA 1.2	Revised for DITA 1.3
In general, a conditional processing attribute provides a list of one or more values separated with whitespace. For instance, `audience="administrator programmer"` qualifies the content as applying to administrators and programmers.	In general, a conditional processing attribute provides a list of one or more values separated with whitespace, or it contains one or more groups that subcategorize the attribute. For instance, `audience="administrator programmer"` qualifies the content as applying to administrators and programmers. Using groups, `product="database(A B) appserver(C)"` qualifies the content as applying to databases A and B, as well as to application server C.

The following change will be made to the definition of the "att" attribute on the <prop> element inside of the DITAVAL markup (http://docs.oasis-open.org/dita/v1.2/os/spec/langref/ditaval-prop.html):

DITA 1.2	Revised for DITA 1.3
The attribute to be acted upon. Must be one of props, audience, platform, product, otherprops, or a specialization of props. If the att attribute is absent, then the prop element declares a default behavior for any conditional processing attribute.	The attribute to be acted upon. If using a literal attribute name, it must be one of props, audience, platform, product, otherprops, or a specialization of props. Otherwise, the value should be the name of a group used within the audience, platform, product, or otherprops attributes. If the att attribute is absent, then the prop element declares a default behavior for any conditional processing attribute.

The following new example should be added to the DITAVAL language spec topic that currently gives sample DITAVAL markup (http://docs.oasis-open.org/dita/v1.2/os/spec/langref/ditaval-val.html).

```
<val>
   <prop action="exclude" att="product" val="appserver"/>
   <prop action="include" att="product" val="mySERVER"/>
   <prop action="include" att="database" val="dbFIRST"/>
   <prop action="include" att="database" val="dbSECOND"/>
   <prop action="exclude" att="database" val="newDB"/>
</val>
```

Assume that "database" and "appserver" are used as group names within the product attribute. In that case, the sample DITAVAL above performs the following actions:

- The first prop element excludes the value "appserver" when used within the product attribute. It also sets a default of "exclude" for values within any appserver() group inside of the product attribute.
- The second prop element sets "mySERVER" to include; this applies whether mySERVER appears alone in the product attribute (`product="mySERVER"`) or inside of any group (`product="appserver(mySERVER)"` or `product="otherGroup(mySERVER)"`).
- The third and fourth prop elements set the database values "dbFIRST" and "dbSECOND" to include. If those values appear inside of a "database" group, they are explicitly set to "include". If they appear elsewhere in a conditional attribute (such as `product="dbFIRST"` or `platform="dbSECOND"`), this rule does not apply.
- The final prop element sets the database value "newDB" to exclude. If that value appears inside of a database group, it is explicitly set to "exclude". If it appears in any other group or attribute, this rule does not apply.

Remember that with groups, if all values inside of a single group evaluate to "exclude", that is equivalent to an entire attribute evaluating to "exclude", which results in the removal of the content. Using this sample DITAVAL:

- `<p product="appserver">` is filtered out, because this value is excluded.
- `<p product="appserver(A B)">` is filtered out, because there is no explicit rule for A or B, and values in the "appserver" group inside of @product default to exclude.
- `<p product="appserver(A B mySERVER)">` is included, because product="mySERVER" evaluates to "include", which means the entire group evaluates to "include".
- `<p product="newDB">` is included, because no rule in the DITAVAL applies, so the "newDB" token defaults to "include".
- `<p product="database(newDB)">` is filtered out, because the token "newDB" is excluded when found in the database group.
- `<p product="database(dbFIRST dbSECOND newDB)">` is included, because both "dbFIRST" and "dbSECOND" are included, so the group evaluates to include.
- `<p product="database(newDB) appserver(mySERVER)">` is filtered out, because the token "newDB" is excluded when found in the database group. The entire "database" group on this paragraph evaluates to "exclude", so the element is excluded, regardless of how the "appserver" group evaluates.

Figure 1. DITAVAL with conditions for groups

Chapter 2. Conditional processing (profiling)

Attribute-based profiling, also known as conditional processing or applicability, is the use of classifying metadata that enables the filtering, flagging, searching, indexing, and other processing based on the association of an element with one or more values in a specific classification domain.

DITA defines attributes that are specifically intended to enable filtering or flagging of individual elements. Those attributes are @audience, @platform, @product, @otherprops, @props, and @rev (flagging only). This enables the creation of topics and maps that can be dynamically configured at processing time to reflect a specific set of conditions, using the DITA-defined conditional processing profile (DITAVAL).

Processors should be able to perform filtering and flagging using the attributes listed above. Although metatdata elements exist with similar names, such as the <audience> element, processors are not required to perform conditional processing using metadata elements. The @props attribute can be specialized to create new attributes, and processors should be able to perform conditional processing on specializations of @props.

Conditional processing attributes

For a topic or topicref, the audience, platform, and product metadata can be expressed with attributes on the topic or topicref element or with elements within the topic prolog or topicmeta element. While the metadata elements are more expressive, the meaning of the values is the same, and can be used in coordination. For example, the prolog elements can fully define the audiences for a topic, and then metadata attributes can be used within the content to identify parts that apply to only some of those audiences.

audience
The values in the audience attribute may also be used to reference a more complete description of an audience in an audience element. Use the name of the audience in the audience element when referring to the same audience in an audience attribute.

The audience attribute takes a space-delimited list of values, which may or may not match the name value of any audience elements. The attribute may also include groups of values specified using the same syntax as generalized attributes within @props; see for details on grouping syntax.

platform
The platform might be the operating system, hardware, or other environment. This attribute is equivalent to the platform element for the topic metadata.

The platform attribute takes a space-delimited list of values, which may or may not match the content of a platform element in the prolog. The attribute may also include groups of values specified using the same syntax as generalized attributes within @props; see for details on grouping syntax.

product
The product or component name, version, brand, or internal code or number. This attribute is equivalent to the prodinfo element for the topic metadata.

The product attribute takes a space-delimited list of values, which may or may not match the value of the prodname element in the prolog. The attribute may also include groups of values specified using the same syntax as generalized attributes within @props; see for details on grouping syntax.

rev The identifier for the revision level. For example, if a paragraph was changed or added during revision 1.1, the rev attribute might contain the value "1.1".

otherprops

A catch-all for metadata qualification values about the content. This attribute is equivalent to the othermeta element for the topic metadata.

The attribute takes a space-delimited list of values, which may or may not match the values of othermeta elements in the prolog. The attribute may also include groups of values specified using the same syntax as generalized attributes within @props; see for details on grouping syntax. The grouping syntax matches the original @otherprops grouping syntax defined in DITA 1.0 and 1.1. That usage was reserved exclusively for @otherprops, and was deprecated in DITA 1.2.

props A generic attribute for conditional processing values. Starting with DITA 1.1, the props attribute can be specialized to create new conditional processing attributes. See .

Using conditional processing attributes

Each attribute takes zero or more space-delimited string values or grouped values. For example, you can use the product attribute to identify that an element applies to two particular products.

```
<p audience="administrator">Set the configuration options:
    <ul>
        <li product="extendedprod">Set foo to bar</li>
        <li product="basicprod extendedprod">Set your blink rate</li>
        <li>Do some other stuff</li>
        <li platform="Linux">Do a special thing for Linux</li>
    </ul>
</p>
```

Figure 2. Example source

Groups may be defined using the generalized attribute syntax found in . This is intended to provide subcategories of the attribute; for example, the following could be used to indicate that a step applies to one application server and two databases. Each group consists of one or more space-delimited string values; groups cannot be nested.

```
<steps>
  <step><cmd>Common step</cmd></step>
  <step product="appserver(mySERVER) database(ABC dbOtherName)">
    <cmd>Do something special for databases ABC or OtherName when installing on mySERVER</cmd>
  </step>
  <!-- additional steps -->
</steps>
```

Figure 3. Grouped values in @product

If both grouped values and ungrouped values are found in a single attribute, the ungrouped values must be treated as a single group. The following product attributes are equivalent:

```
product="a database(dbA dbB) b appserver(mySERVER) c"
product="product(a b c) database(dbA dbB) appserver(mySERVER)"
```

If two groups with the same name are found in a single attribute, they should be treated as if all values are specified in the same group. The following product attributes are equivalent:

```
product="database(a b) database(c) appserver(APPNAME)"
product="database(a b c) appserver(APPNAME)"
```

Evaluating conditional processing attributes

At processing time, a DITAVAL conditional processing profile may be used to specify values you want to include, exclude, or flag.

For example, a publisher producing information for a mixed audience using the basic product could choose to flag information that applies to administrators, and exclude information that applies to the extended product, by defining a conditional processing profile like this:

```
<val>
  <prop att="audience" val="administrator" action="flag">
    <startflag><alt-text>ADMIN</alt-text></startflag>
  </prop>
  <prop att="product" val="extendedprod" action="exclude"/>
</val>
```

At output time, the paragraph is flagged, and the first list item is excluded (since it applies to extendedprod), but the second list item is still included (even though it does apply to extendedprod, it also applies to basicprod, which was not excluded).

The result should look something like:

ADMIN Set the configuration options:

- Set your blink rate
- Do some other stuff
- Do a special thing for Linux

| If a conditional processing attribute is set to an empty value (`product=""`), this is equivalent to an
| unspecified attribute. An empty group within an attribute is equivalent to an unspecified group. For
| example, `product="database()"` is equivalent to `product=""`, and `product="database() A"` is equivalent to
| `product="A"`.

| If two groups of the same name exist on different attributes, each group will evaluate the same way.
| Subject schemes may be used to define the same group name differently for two different attributes, but
| for the purposes of DITAVAL filtering or flagging, there is no way to distinguish. If there is a need to
| distinguish between a similarly named group on different attributes, the best practice is to use more
| specific group names such as platformGroupname and productGroupname. Alternatively, DITAVAL rules
| can be specified based on the attribute name rather than the group name.

| Note that in the case where the same group name is used in different attributes, a complex scenario could
| be created where different defaults are specified for different attributes, with no rule set for a group or
| individual value. In this case a value may end up evaluating differently in the different attributes. For
| example, a DITAVAL may set a default of "exclude" for @platform and a default of "include" for
| @product. If no rules are specified for group oddgroup(), or for the value `oddgroup="edgecase"`, the
| attribute `platform="oddgroup(edgecase)"` will evaluate to "exclude" while `product="oddgroup(edgecase)"`
| will resolve to "include".

Filtering logic

By default, values in conditional processing attributes that are not defined in a DITAVAL profile evaluate to "include". For example, if the value audience="novice" is used on a paragraph, but this value is not defined in a DITAVAL profile, the attribute evaluates to "include". However, the DITAVAL profile may change this default to "exclude", so that any value not explicitly defined in the DITAVAL profile will evaluate to "exclude". The profile may also be used to change the default for a single attribute; for example, it may declare that values in the platform attribute default to exclude while those in the product attribute default to include. See **** MISSING FILE **** for information on how to set up a DITAVAL profile and how to change default behaviors.

When deciding whether to include or exclude a particular element, a processor should evaluate each attribute, and then evaluate the set of attributes.

- If **all** the values in a single attribute evaluate to "exclude", the attribute evaluates to "exclude".

- If **all** thje values in **any single group** within an attribute evaluate to "exclude", the attribute evaluates to "exclude".
- If **any single attribute** evaluates to exclude, the element is excluded.

For example, if a paragraph applies to three products and the publisher has chosen to exclude all of them, the processor should exclude the paragraph. This is true even if the paragraph applies to an audience or platform that is not excluded. But if the paragraph applies to an additional product that has not been excluded, then its content is still relevant for the intended output and should be preserved.

Similarly, with groups, a step may apply to one application server and two databases, as in the sample above:

```
<steps>
  <step><cmd>Common step</cmd></step>
  <step product="appserver(mySERVER) database(ABC dbOtherName)">
    <cmd>Do something special for databases ABC or OtherName when installing on mySERVER</cmd>
  </step>
  <!-- additional steps -->
</steps>
```

If a publisher decides to exclude the application server mySERVER, then the appserver() group evaluates to exclude. This may be done by setting product="mySERVER" to exclude *or* by setting appserver="mySERVER" to exclude. This means the step should be excluded, regardless of how the values "db2" or "oracle" evaluate. If a rule is specified for both product="mySERVER" and appserver="mySERVER", the rule for the more specific group name "appserver" takes precedence.

Similarly, if both "db2" and "oracle" evaluate to exclude, then the database() group evaluates to exclude and the element should be excluded regardless of how the "WAS" value is set.

In more advanced usage, a DITAVAL may be used to specify a rule for a group name. For example, an author could create a DITAVAL rule that sets `product="database"` to "exclude". This is equivalent to setting a default of "exclude" for any value in a database group (it would also exclude the more common usage of "database" as a single token within the product attribute). Thus when the token "db2" appears in a database() group within the product attribute, the full precedence for determining whether to include or exclude the value is as follows:

1. Check for a DITAVAL rule for database="db2"
2. Check for a DITAVAL rule for product="db2"
3. Check for a DITAVAL rule for product="database" (default for the database group)
4. Check for a DITAVAL rule for "product" (default for the product attribute)
5. Check for a default rule for all conditions
6. Otherwise, evaluate to "include"

Flagging logic

When deciding whether to flag a particular element, a processor should evaluate each value. Wherever a value that has been set as flagged appears in its attribute (for example, audience="administrator") the process should add the flag. When multiple flags apply to a single element, multiple flags should be rendered, typically in the order they are encountered.

Flagging could be done using text (for example, bold text against a colored background) or using images. When the same element evaluates as both flagged and filtered (for example, flagged because of an audience attribute value and filtered because of its product attribute values), the element should be filtered.