Review C: Programming, software, and UI domains

Table of contents

1 Programming domain

The programming domain elements are used to define the syntax for programming languages. They can also be used to provide examples.

Comment by Eliot Kimber
I would say "define and describe"
Disposition: Unassigned

1.1 <apiname>

The `<apiname>` element identifies the name of an application programming interface (API), such as a Java class name or method name.

Specialization hierarchy

The `<apiname>` is specialized from `<keyword>`. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and `@keyref` (26).

Example

The following code sample shows how the `<apiname>` element can be used to identify the `document.write` method:

```
<p>Use the <apiname>document.write</apiname> method to create text
output in the dynamically constructed view.</p>
```

1.2 <codeblock>

The `<codeblock>` element identifies lines of program code.

Rendering expectations

001 (17)

Processors **SHOULD** preserve line the breaks and spaces that are present in the content of a `<codeblock>` element.

Comment by Eliot Kimber
Content SHOULD be rendered in a monospaced font.

Kris Eberlein, 17 January 2023

We can't make a normative statement about font choices. Font choices are just **not** germane to interoperability. In the "Formatting conventions"topic, we do suggest monospace for the contents of the <codeblock> element.

Disposition: Unassigned

Specialization hierarchy

The <codeblock> is specialized from <pre>. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: display attributes (22), universal attributes (18), and @xml:space (30).

Example

The following code sample shows how the <codeblock> element can be used to tage an excerpt from the code for a program:

```
<codeblock>
/* a long sample program */
Do forever
  Say "Hello, World"
End
</codeblock>
```

1.3 <codeph>

The <codeph> element identifies a code snippet.

Comment by Eliot Kimber
Should there be a Rendering expectations as for codeblock to indicate monospace font?
Disposition: Unassigned

Specialization hierarchy

The <codeph> is specialized from <ph>. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and @keyref (26).

Example

In the following code sample, the <codeph> element identifies a code snippet. The code snippet will be rendered in-line in the paragraph.

```
<p>The second line of the sample program code, <codeph>Do forever</codeph>,
represents the start of a loop construct.</p>
```

1.4 <coderef>

The <coderef> element references an external file that contains literal code.

Rendering expectations

When evaluated, the <coderef> element causes the target code to be displayed inline. If the target code contains non-XML characters such as '<' or '&', those characters need to be handled so that they can be displayed correctly by the final rendering engine.

Comment by Eliot Kimber
Content should be rendered in monospaced font.
Disposition: Unassigned

Specialization hierarchy

The <coderef> is specialized from <include>. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: inclusion attributes (22), link-relationship attributes (22), universal attributes (18), and @keyref (26).

For this element, the @parse attribute has a default value of "text".

Example

In the following code sample, the <codref> element references the content of the process-dita.xsl file. In the rendered output, the XSL code will be presented in a code block.

```
<example>
  <title>Processing DITA</title>
  <p>This code is an example of how to process DITA.</p>
  <codeblock>
    <coderef href="process-dita.xsl"/>
  </codeblock>
</example>
```

1.5 <option>

The <option> element describes an option that can modify a command or a configuration.

Comment by Eliot Kimber
What is the semantic distinction between <option> and <parmname>?

When documenting how to use command-line commands I always use <parmname> but reading this I just realized I didn't even remember that <option> was available.

In my mind, a "parameter" is anything you specify after the command name when invoking a command-line command, but the example for <option> shows using it to document a command-line command. But so does the example for <parmname>.

So when would you use one in preference to the other and why?

Kris Eberlein, 17 January 2022

This is a good point. I suspect that there was not enough rigor originally in making a distinction between <option> and <parmname> (but I haven't looked at the DITA 1.0 spec). If we can clarify between uses for the two elements now, it would be good.

Here's what the 1.0 spec contained:

<option>
The <option> element describes an option that can be used to modify a command (or something else, like a configuration).
Example: "something <option>/modifier</option>"

<parmname>

When referencing the name of an application programming interface parameter within the text flow of your topic, use the parameter name (<parmname>) element to markup the parameter.
Example: "Use the <parmname>/env</parmname> parameter of the <cmdname>config</cmdname> command to update the field value."

Disposition: Unassigned

Specialization hierarchy

The `<option>` is specialized from `<keyword>`. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and `@keyref` (26).

Example

The following code sample shows how the command-line options for a tool are defined in a list:

```
<p>The most common command line options include:</p>
<ul>
  <li><option>-compress</option> will generate data in compressed form.</li>
  <li><option>-debug</option> will generate debug information while running.</li>
  <li><option>-help</option> will print extended help information.</li>
</ul>
```

1.6 <parmname>

The `<parmname>` element identifies the name of a parameter.

Comment by Eliot Kimber
See my question on `<option>` and how it differs from `<parmname>`
Disposition: Unassigned

Specialization hierarchy

The `<parmname>` is specialized from `<keyword>`. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and `@keyref` (26).

Example

The following code sample shows how the `<parmname>` element can be used to identify a parameter that is used with the `config` command:

```
<p>Use the <parmname>/env</parmname> parameter of the <cmdname>config</cmdname>
command to update the field value.</p>
```

1.7 <parml>

The `<parml>` element identifies a specialized definition list that is designed for documenting parameters.

Comment by Kristen James Eberlein on 14 January 2023
I suggest using natural language: "A parameter list is a specialized definition list that is …"

Eliot 2023-01-17: I agree.

Disposition: Unassigned

Specialization hierarchy

The <parml> is specialized from <dl>. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and @compact (24).

Example

The following code sample shows how a set of sample code is followed by a parameter list that defines those parameters:

```
<p>This code example is a basic method signature:</p>
<codeblock>returnType methodName(pList1, pList2)</codeblock>
<p>The method requires the following parameters:</p>
<parml>
  <plentry>
    <pt>pList1</pt>
    <pd>The first variable declaration that is passed to methodName</pd>
  </plentry>
  <plentry>
    <pt>pList2</pt>
    <pd>The second variable declaration that is passed to methodName</pd>
  </plentry>
</parml>
```

1.8 <plentry>

The <plentry> element contains one or more parameter terms and definitions,

Comment by Kristen James Eberlein on 14 January 2023
I suggest using natural language: "A parameter-list entry contains …"
Disposition: Unassigned

Specialization hierarchy

The <plentry> is specialized from <dlentry>. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (18).

Example

See <parml> (6).

1.9 <pt>

The <pt> element specifies a parameter term within a parameter list entry.

Specialization hierarchy

The <pt> is specialized from <dt>. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and @keyref (26).

Example

See <parml> (6).

1.10 <pd>

The <pd> element specifies a parameter definition within a parameter list entry.

Specialization hierarchy

The <pd> is specialized from <dd>. It is defined in the programming domain module.

Attributes

The following attributes are available on this element: universal attributes (18).

Example

See <parml> (6).

2 Software domain

The software domain elements are used to describe the operation of a software program.

Comment by Kristen James Eberlein on 13 January 2023
The elements need to be listed in alphabetical order.
Disposition: Unassigned

Comment by Eliot Kimber
Not suggesting we change it, but just observing that I've always found the distinction between the programming domain and the software domain somewhat dubious. In my normal how-to documentation work I use `<cmdname>` with `<parmname>` to document command-line command operation and of course `<filepath>` is just generally useful everywhere.

So there's few situations I can think of where I would ever need one domain and not the other.

Disposition: Unassigned

2.1 <msgph>

The `<msgph>` element identifies the text of a message that is produced by an application or program.

Specialization hierarchy

The `<msgph>` element is specialized from `<ph>`. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and `@keyref` (26).

Example

The following code sample shows how the `<msgph>` element can be used to tag a message that is returned by the server:

```
<p>A server log entry of <msgnum>I:0</msgnum> is equivalent to the
text message, <msgph>informational: successful</msgph>.</p>
```

2.2 <msgblock>

The `<msgblock>` element contains a multi-line message or set of messages.

Usage information

The `<msgblock>` element can contain multiple message numbers and message descriptions, each enclosed in `<msgnum>` and `<msgph>` elements. It can also contain the message content directly.

Rendering expectations

002 (17)

Processors **SHOULD** preserve the line breaks and spaces that are present in the content of a `<msgblock>` element.

Comment by Eliot Kimber
The content should be rendered in a monospaced font.
Disposition: Unassigned

Specialization hierarchy

The <msgblock> element is specialized from <pre>. It is defined in the software domain module.

Attributes

The following attributes are available on this element: display attributes (22), universal attributes (18), and @xml:space (30).

Example

The following code sample shows a <msgblock> element that contains a multi-line message that is returned by an application:

```
<p>A sequence of failed password attempts generates the following message stream:</p>
<msgblock>
I:0
S:3
I:1
S:3
I:1
S:4
S:99 (lockup)
</msgblock>
```

2.3 <msgnum>

The <msgnum> element identifies the number of a message that is produced by an application or program.

Comment by Eliot Kimber
c/number of a message/identifier of a message/ Messages may not necessarily have numbers, strictly speaking, that identify them. In the online game Destiny, all the message identifiers are animal names (I.e, message "badger" indicates a network connection issue or whatever).
Disposition: Unassigned

Specialization hierarchy

The <msgnum> element is specialized from <keyword>. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and @keyref (26).

Example

The following code sample shows a <msgnum> element that identifies the number of the message that is returned by an application:

```
<p>A server log entry of <msgnum>I:0</msgnum> is equivalent to the
text message <msgph>informational: successful</msgph>.</p>
```

2.4 <cmdname>

The <cmdname> element identifies the name of a software command.

Specialization hierarchy

The <cmdname> element is specialized from <keyword>. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and @keyref (26).

Example

The following code sample shows a <cmdname> element that identifies the name of the rm command.

```
<p>Use the <cmdname>rm</cmdname> command to permanently delete an object.</p>
```

2.5 <varname>

The <varname> element identifies a variable that is supplied to a software application.

Specialization hierarchy

The <varname> element is specialized from <keyword>. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and @keyref (26).

Example

The following code sample shows how the <varname> element is used to identify variables that represent the "installation directory," "project directory," and "file name":

```
<filepath>
  <varname>install-dir</varname>\projects\working\<varname>project-dir</varname>
      \source\<varname>filename</varname>.java
</filepath>'
```

Comment by
Eliot Kimber
Can we change "\" to "/"? Even Windows recognizes forward slashes these days.

Disposition: Unassigned

2.6 <filepath>

The <filepath> element identifies file names and system paths.

Specialization hierarchy

The <filepath> element is specialized from <ph>. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and `@keyref` (26).

Example

In the following code sample, the `<filepath>` element is used to tag both file names and system paths:

```
<p>Uncompress the <filepath>gbbrsh.gz</filepath> file to the
<filepath>/usr</filepath> directory. Ensure that the
<filepath>/usr/tools/data.cfg</filepath> path is listed in
the execution path system variable.</p>
```

2.7 <userinput>

The `<userinput>` element identifies text that a user types in response to an application or system prompt.

Specialization hierarchy

The `<userinput>` element is specialized from `<ph>`. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and `@keyref` (26).

Example

In the following code sample, the `<userinput>` element identifies text that a user should type at the command prompt:

Comment by Eliot Kimber
I would consider this example incorrect markup as "dir" is a command name and should be tagged with `<cmdname>`. I think a better example would show `<userinput>` being used for a response to a prompt, i.e.:

```
% <cmdname>copy-files</cmdname>
<systemoutput>Specify source directory:</systemoutput> <userinput>~/Downloads</userinput>
<systemoutput>Specify target directory:</systemoutput> <userinput>~/workspace/data</
userinput>
...
%
```

Kris Eberlein, 17 January 2023.

Eliot, I respectfully disagree. I'd of course tag "dir" with the <cmdname> element in an expository text flow, but not when it was part of a procedure, where you are simply directing a user to type a particular string. In that situation (which is what is shown in the current example), you simply want to tag the string with <userinput>.

Disposition: Unassigned

```
<p>From a DOS command prompt, type <userinput>dir</userimput> to view a list
of files in the current directory.</p>
```

2.8 <systemoutput>

The <systemoutput> element identifies computer output or responses to a command or situation.

Specialization hierarchy

The <systemoutput> element is specialized from <ph>. It is defined in the software domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and @keyref (26).

Example

In the following code sample, the <systemoutput> element identifies an application response to user input:

```
<p>After you type <userinput>mealplan dinner</userinput>, the meal planning program
will print <systemoutput>For what day?</systemoutput>.
Reply by typing the day of the week for which you want a meal plan,
for example, <userinput>Thursday</userinput>.</p>
```

3 User interface domain

The user-interface domain elements are used to describe the user interface of a software program.

Comment by Kristen James Eberlein on 13 January 2023
The elements need to be listed in alphabetical order.
Disposition: Unassigned

3.1 <uicontrol>

The `<uicontrol>` element identifies user interface controls, such as names of buttons, fields, menu items, and other objects that enable users to control an interface.

Usage information

The `<uicontrol>` element is also used inside a `<menucascade>` element to identify a sequence of menu choices in a nested menu, such as **File** > **New**.

Specialization hierarchy

The `<uicontrol>` element is specialized from `<ph>`. It is defined in the user-interface domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and `@keyref` (26).

Example

The following code sample shows how the `<uicontrol>` element can be used to identify a button that a user is directed to press:

```
<p>Press <uicontrol>OK</uicontrol> to continue.</p>
```

3.2 <wintitle>

The `<wintitle>` element identifies named windows and dialogs.

Specialization hierarchy

The `<wintitle>` element is specialized from `<keyword>`. It is defined in the user-interface domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and `@keyref` (26).

Example

The following code sample shows how the `<wintitle>` element can be used to tag the name of the "Configuration Options" window:

```
<step>
  <cmd>Click <uicontrol>Configure</uicontrol>.</cmd>
```

```
  <stepresult>The <wintitle>Configuration Options</wintitle> window
  opens with your last set of selections highlighted.</stepresult>
</step>
```

3.3 <menucascade>

The <menucascade> element identifies a sequence of menu choices in a nested menu, such as **File** > **New**.

Specialization hierarchy

The <menucascade> element is specialized from <ph>. It is defined in the user-interface domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and @keyref (26).

Example

The following code sample shows how the <menucascade> element can be used to identify a series of menu choices that enable users to launch the Notepad application:

```
<menucascade>
 <uicontrol>Start</uicontrol>
 <uicontrol>Programs</uicontrol>
 <uicontrol>Accessories</uicontrol>
 <uicontrol>Notepad</uicontrol>
</menucascade>
```

3.4 <shortcut>

The <shortcut> element identifies a keyboard shortcut for a menu or window action.

Specialization hierarchy

The <shortcut> element is specialized from <keyword>. It is defined in the user-interface domain module.

Attributes

The following attributes are available on this element: universal attributes (18) and @keyref (26).

Example

In the following code sample, the <shortcut> element identifies the keyboard shortcut for the "Start Programs" menu action:

```
<menucascade>
 <uicontrol>Start</uicontrol>
 <uicontrol><shortcut>P</shortcut>rograms</uicontrol>
</menucascade>
```

3.5 <screen>

The <screen> element contains a textual representation of a terminal console or other text-based computer interface.

Rendering expectations

003 (17)

Processors **SHOULD** preserve the line breaks and spaces that are present in the content of a <screen> element.

Comment by Eliot Kimber
Content should be rendered in monospaced font.
Disposition: Unassigned

Specialization hierarchy

The <screen> element is specialized from <pre>. It is defined in the user-interface domain module.

Attributes

The following attributes are available on this element: universal attributes (18), display attributes (22), and @xml:space (30).

Example

In the following code sample, the <screen> element is used to provide a representation of a DOS window:

```
<screen>
File  Edit  Search  View  Options  Help
+--------------------------------- UNTITLED1 ----------------------------------+
¦                                                                              ¦
¦                                                                              ¦
¦                                                                              ¦
¦                                                                              ¦
¦  Line:1    Col:1  F1=Help                                                    ¦
+------------------------------------------------------------------------------+
</screen>
```

A Aggregated RFC-2119 statements

This appendix contains all the normative statements from the DITA for Technical Content 2.0 specification. They are aggregated here for convenience in this non-normative appendix.

Item	Conformance statement
001 (3)	Processors **SHOULD** preserve line the breaks and spaces that are present in the content of a `<codeblock>` element. **Comment by Eliot Kimber** Content SHOULD be rendered in a monospaced font. --- Kris Eberlein, 17 January 2023 We can't make a normative statement about font choices. Font choices are just **not** germane to interoperability. In the "Formatting conventions"topic, we do suggest monospace for the contents of the <codeblock> element. **Disposition: Unassigned**
002 (9)	Processors **SHOULD** preserve the line breaks and spaces that are present in the content of a `<msgblock>` element. **Comment by Eliot Kimber** The content should be rendered in a monospaced font. **Disposition: Unassigned**
003 (16)	Processors **SHOULD** preserve the line breaks and spaces that are present in the content of a `<screen>` element. **Comment by Eliot Kimber** Content should be rendered in monospaced font. **Disposition: Unassigned**
004 (23)	If no value is set and no value cascades from a containing element, processors **SHOULD** assume a default of "merge". Processors can define custom values for the `@cascade` attribute.

B Attributes

This section collects commonly used attributes, with common definitions. If an element uses a different definition, or narrows the scope of, an otherwise common attribute, it will be called out in the topic that defines the element. This section contains definitions for commonly-used attributes. If an attribute is defined differently on a specific element, that information is covered in the topic for the specific element.

Comment by Kristen J Eberlein on 29 December 2021

Add a brief overview of the fact that some specific attributes are overloaded – and have different meanings depending on what element they are specified upon.

Disposition: Unassigned

B.1 Universal attribute group

The universal attribute group defines a set of common attributes that are available on almost every DITA element. The universal attribute group includes all attributes from the ID, localization, and metadata attribute groups, plus the `@class` and `@outputclass` attributes.

Comment by Kristen J Eberlein on 29 December 2021

This is something wrong with the organizational structure of this topic ... Look at it in outline form, and check that the sections, titles, and content all make logical sense with the topic title of "Universal attribute group".

Disposition: Unassigned

Common attribute groups

The following attribute groups are referenced in this specification. They are also used in the grammar files when the element attributes are defined.

Universal attributes

Includes `@class` and `@outputclass`, along with every attribute in the ID,localization, and metadata attribute groups.

ID attributes

This group includes the attributes that enable the naming and referencing of elements: `@conaction`, `@conkeyref`, `@conref`, `@conrefend`, and `@id`.

Localization attributes

This group includes attributes that are related to translation and localization: `@dir`, `@translate`, and `@xml:lang`.

Metadata attributes

Comment by Kristen J Eberlein on 31 December 2021

Why do we need to mention that two attributes are available for specialization here? I think it makes the paragraph hard to read.

Disposition: Unassigned

This group includes common metadata attributes, two of which are available for specialization: `@base`, `@importance`, `@props`, `@rev`, and `@status`.

The base DITA vocabulary from OASIS includes several specializations of `@props`: `@audience`, `@deliveryTarget`, `@otherprops`, `@platform`, and `@product`. These attributes are defined as attribute-extension domains. By default, they are integrated into all OASIS-provided document-type shells, but they can be made unavailable by implementing custom document-type shells.

Comment by Kristen J Eberlein on 29 December 2021

Why do we provide information about specialization and custom document-type shells here? I think that information could be removed.

Disposition: Unassigned

Universal attribute definitions

The universal attributes for OASIS DITA elements are defined below. Specialized attributes, which are part of the OASIS distribution but are only available when explicitly included in a shell, are noted in the list.

@audience *(specialized attribute)*
Indicates the intended audience for the element. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value will cascade from the closest containing element.

@base
Specifies metadata about the current element. It is often used as a base for specialized attributes that have a simple syntax for values but are not filtering or flagging attributes.

The `@base` attribute takes a space-delimited set of values. However, when serving as a container for generalized attributes, the attribute values will be more complex. See Attribute generalization for more details.

@class *(not for use by authors)*
This attribute is not for use by authors. If an editor displays `@class` *attribute values, do not edit them.* Specifies a default value that defines the specialization ancestry of the element. Its predefined values allow DITA and XDITA tools to work correctly with specialized elements. In a generalized DITA document the `@class` attribute value in the generalized instance might differ from the default value for the `@class` attribute for the element as given in the DTD or schema. See class attribute rules and syntax for more information. This attribute is specified on every element except for the `<dita>` container element. It is always specified with a default value, which varies for each element.

@conaction
Specifies how the element content will be pushed into a new location. The following values are valid:

mark
The element acts as a marker when pushing content before or after the target, to help ensure that the push action is valid. The element with `conaction="mark"` also specifies the target of the push action with `@conref`. Content inside of the element with `conaction="mark"` is not pushed to the new location.

pushafter
Content from this element is pushed after the location specified by `@conref` on the element with `conaction="mark"`. The element with `conaction="pushafter"` is the first sibling element after the element with `conaction="mark"`.

pushbefore
Content from this element is pushed before the location specified by `@conref` on the element with `conaction="mark"`. The element with `conaction="pushbefore"` is the first sibling element before the element with `conaction="mark"`.

pushreplace
Content from this element replaces any content from the element referenced by the `@conref` attribute. A second element with `conaction="mark"` is not used when using `conaction="pushreplace"`.

-dita-use-conref-target
See STUB CONTENT for more information.

See STUB CONTENT for examples and details about the syntax.

@conkeyref
Specifies a key name or a key name with an element ID that acts as an indirect reference to reusable content. The referenced content is used in place of the content of the current element. See STUB CONTENT for more details about the syntax and behaviors.

@conref
Specifies a URI that references a DITA element. The referenced content is used in place of the content of the current element. See STUB CONTENT for examples and details about the syntax.

@conrefend
Specifies a URI that references the last element in a sequence of elements, with the first element of the sequence specified by `@conref`. The referenced sequence of elements is used in place of the content of the current element. See STUB CONTENT for examples and details about the syntax.

@deliveryTarget ***(specialized attribute)***
Specifies the intended delivery target of the content, for example, "html", "pdf", or "epub". If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value will cascade from the closest containing element.

@dir
Specifies the directionality of text. The following values are valid:

lro
Indicates an override of normal bidirectional text presentation, forcing the element into left-to-right mode.

ltr
Left to right (the processing default).

rlo
Indicates an override of normal bidirectional text presentation, forcing the element into right-to-left mode.

rtl
Right to left.

-dita-use-conref-target
See STUB CONTENT for more information.

See The dir attribute for more information.

@id
Specifies an identifier for the current element. This ID is the target for references by `@href` and `@conref` **href** and **conref** attributes and for external applications that refer to DITA or LwDITA content. This attribute is defined with the XML data type NMTOKEN, except where noted for specific elements within the language reference. See ID attribute for more details.

@importance
Specifies the importance or priority that is assigned to an element. The following values are valid: "default", "deprecated", "high", "low", "normal", "obsolete", "optional", "recommended", "required", "urgent", and "-dita-use-conref-target". This attribute is not used for DITAVAL-based filtering or flagging, although applications might use the importance value to highlight elements. For example, in steps of a task, the value of the `@importance` attribute indicates whether a step is optional or required.

@otherprops ***(specialized attribute)***
Specifies a property or properties that provide selection criteria for the element. Alternatively, the `@props` attribute can be specialized to provide a new metadata attribute instead of using the general `@otherprops`

attribute. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value will cascade from the closest containing element.

@outputclass

Specifies a role that the element is playing. The role must be consistent with the basic semantic and expectations for the element. In particular, the `@outputclass` attribute can be used for styling during output processing; HTML output will typically preserve `@outputclass` for CSS processing.

> **Comment by robander**
> I don't like "The role must be consistent...", that seems like best practice that cannot be normative – and I could easily say outputclass="flashy" which makes my element show up with sparkles, and has nothing to do with "the basic semantic and expectations for the element".
> **Disposition: Unassigned**

@platform ***(specialized attribute)***

Indicates operating system and hardware. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value will cascade from the closest containing element.

> **Comment by robander**
> I think this could specify a platform that is not an operating system or hardware, right? The current definition explicitly limits platform to those two … maybe "Specifies a platform or platforms to which the element applies, such as the operating system or hardware relevant to a task."
> **Disposition: Unassigned**

@product ***(specialized attribute)***

Specifies the name of the product to which the element applies. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value will cascade from the closest containing element.

@props

Specifies metadata about the current element. New attributes can be specialized from the `@props` attribute. This is an attribute that supports conditional processing for filtering or flagging. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value will cascade from the closest containing element.

The `@props` attribute takes a space-delimited set of values. However, when acting as a container for generalized attributes, the attribute values will be more complex; see Attribute generalization for more details.

@rev

Specifies a revision level of an element that identifies when the element was added or modified. It can be used to flag outputs when it matches a run-time parameter. It cannot be used for filtering nor is it sufficient to be used for version control. If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value will cascade from the closest containing element.

@status

Specifies modification status of the current element. The following values are valid: "new", "changed", "deleted", "unchanged", and "-dita-use-conref-target".

@translate

> **Comment by Kristen J Eberlein on 31 December 2021**
>
> Does Element-by-element recommendations for translators really provide suggested processing defaults for each element? I thought it covered whether an element was block or in-line and whether there were considerations that translators needed to be aware of.
>
> **Disposition: Unassigned**

Specifies whether the content of the element should be translated. The following values are valid: "yes", "no", and "-dita-use-conref-target". See Element-by-element recommendations for translators for suggested processing defaults for each element.

@xml:lang

Specifies the language of the content contained in an element. The following values are valid: language tokens or the null string. The `@xml:lang` attribute and its values are described in the Extensible Markup Language 1.0 specification, fifth edition.

B.2 Common attributes

The common attributes topic collects defines most of the attributes that are used on more than one base element.

Common attribute groups

The following groups are referenced in this specification, and they are also used in grammar files when defining attributes for elements.

Architectural attributes

This group includes a set of attributes that are defined for document-level elements such as `<topic>` and `<map>`: `@DITAArchVersion`, `@specializations`, and `@xmlns:ditaarch`.

Common map attributes

This group includes attributes that are frequently used on map elements: `@cascade`, `@chunk`, `@collection-type`, `@keyscope`, `@linking`, `@processing-role`, `@search`, and `@toc`.

Complex table attributes

This group includes attributes that are defined on table elements but not simple table elements. These attributes are part of the OASIS Exchange Table Model, unless otherwise noted. Table elements generally use only a subset of the attributes that are defined in this group. This group contains the following attributes: `@align`, `@char`, `@charoff`, `@colsep`, `@rowheader`, `@rowsep`, and `@valign`.

Data-element attributes

Includes attributes defined on `<data>` and its many specializations: `@datatype`, `@name`, and `@value`

Date attributes

Includes attributes that take date values, and are defined on metadata elements that work with date information: `@expiry` and `@golive`

Display attributes

This group includes attributes that affect the rendering of many elements: `@expanse`, `@frame`, and `@scale`.

Inclusion attributes

Includes attributes defined on `<include>` and its specializations: `@encoding` and `@parse`.

Link-relationship attributes

This group includes attributes whose values can be used for representing navigational relationships: `@format`, `@href`, `@type`, and `@scope`.

Simple table attributes

This group includes attributes that are defined only on the `<simpletable>` element: `@keycol` and `@relcolwidth`. These attributes are listed in a group because the `<simpletable>` element is frequently used as a specialization base.

Other attributes (not in a group)

These are attributes that are used in the same way on more than one base element, but they are not formally grouped together: `@compact`, `@duplicates`, `@otherrole`, `@role`, and `@title-role`.

Common attribute definitions

Common attributes, including those in the groups listed above, are defined as follows.

@align (complex table attributes)

Specifies the horizontal alignment of text in table entries. The following values are valid:

left
Indicates left alignment of the text.

right
Indicates right alignment of the text.

center
Indicates center alignment of the text.

justify
Justifies the contents to both the left and the right.

char
Indicates character alignment. The text is aligned with the first occurrence of the character specified by the `@char` attribute.

-dita-use-conref-target
See STUB CONTENT for more information.

The `@align` attribute is available on the following table elements: `<colspec>`, `<entry>`, and `<tgroup>`.

@cascade (common map attributes)

> **Comment by Kristen J Eberlein on 31 December 2021**
>
> We need to move the normative statement elsewere. Should it go in Cascading of metadata attributes in a DITA map?
>
> **Disposition: Unassigned**

Specifies how metadata attributes cascade within a map. The specification defines the following values:

merge
Explanation needed

nomerge
Explanation needed

004 (17)	If no value is set and no value cascades from a containing element, processors **SHOULD** assume a default of "merge". Processors can define custom values for the `@cascade` attribute.

See Cascading of metadata attributes in a DITA map for more information about how this attribute interacts with metadata attributes.

@char (complex table attributes)

Specifies the alignment character, which is the character that is used for aligning the text in table entries. This attribute applies when `align="char"`. A value of "" (the null string) means there is no aligning character.

For example, if `align="char"` and `char="."` are specified, then text in the table entry aligns with the first occurrence of the period within the entry. This might be useful if decimal alignment is required.

The `@char` attribute is available on the following table elements: `<colspec>` and `<entry>`.

@charoff (complex table attributes)

Specifies the horizontal offset of the alignment character that is specified by the `@char` attribute. The value is a greater-than-zero number that is less than or equal to 100. It represents the percentage of the current column width by which the text is offset to the left of the alignment character.

For example, if `align="char"`, `char="."`, and `charoff="50"` are all specified, then text in the table entry is aligned 50% of the distance to the left of the first occurrence of the period character within the table entry.

The `@charoff` attribute is available on the following table elements: `<colspec>` and `<entry>`.

@chunk (common map attributes)

Specifies how a processor should render a map or branch of a map. For example, it can be used to specify that individual topic documents should be rendered as a single document, or that a single document with multiple topics should be rendered as multiple documents.

For a detailed description of the `@chunk` attribute and its usage, see Chunking.

@collection-type (common map attributes)

Specifies how topics or links relate to each other. The processing default is "unordered", although no default is specified in the DTD or Schema. The following values are valid:

unordered
Indicates that the order of the child topics is not significant.

sequence
Indicates that the order of the child topics is significant; output processors will typically link between them in order.

choice
Indicates that one of the children should be selected.

family
Indicates a tight grouping in which each of the referenced topics not only relates to the current topic but also relate to each other.

@colsep (complex table attributes)

Specifies whether to render column separators between table entries. The following values are valid: "0" (no separators) and "1" (separators).

The `@colsep` attribute is available on the following table elements: `<colspec>`, `<entry>`, `<table>`, and `<tgroup>`.

@compact

Specifies whether the vertical spacing between list items is tightened. The following values are valid: "yes", "no", and "-dita-use-conref-target". Some DITA processors or output formats might not support the `@compact` attribute.

@datatype (data-element attributes)

Specifies the type of data contained in the `@value` attribute or within the `<data>` element. A typical use of `@datatype` will be the identifying URI for an XML Schema datatype.

@DITAArchVersion (architectural attributes)

Specifies the version of the DITA architecture that is in use. The default value increases with each release of DITA. This attribute is in the namespace `http://dita.oasis-open.org/architecture/2005/`. This attribute is defined with the XML data type CDATA, but it uses a default value of the current version of DITA. The current default is "2.0".

@duplicates

Specifies whether duplicate links are removed from a group of links. Duplicate links are links that address the same resource using the same properties, such as link text and link role. How duplicate links are determined is processor-specific. The following values are valid:

yes
Specifies that duplicate links are retained.

no
Specifies that duplicate links are removed.

-dita-use-conref-target
See STUB CONTENT for more information.

The suggested processing default is "yes" within `<linklist>` elements and "no" for other links.

> **Comment by robander on Dec 28 2021**
> "How duplicate links are determined is processor-specific" ==> this should be included in any updates to standardize language around "implementation dependent".
> **Disposition: Unassigned**

@encoding (inclusion attributes)

> **Comment by Kristen J Eberlein on 29 April 2019**
>
> Can we replace "should" in the following definition?
>
> **Disposition: Unassigned**

Specifies the character encoding to use when translating the character data from the referenced content. The value should be a valid encoding name. If not specified, processors may make attempts to automatically determine the correct encoding, for example using HTTP headers, through analysis of the binary structure of the referenced data, or the `<?xml?>` processing instruction when including XML as text. The resource should be treated as UTF-8 if no other encoding information can be determined.

When `parse="xml"`, standard XML parsing rules apply for the detection of character encoding. The necessity and uses of `@encoding` for non-standard values of `@parse` are implementation-dependent.

@expanse (display attributes)
Specifies the horizontal placement of the element. The following values are valid:

column
Indicates that the element is aligned with the current column margin.

page
Indicates that the element is placed on the left page margin for left-to-right presentation or the right page margin for right-to-left presentation.

spread
Indicates that the object is rendered across a multi-page spread. If the output format does not have anything that corresponds to spreads, then "spread" has the same meaning as "page".

textline
Indicates that the element is aligned with the left (for left-to-right presentation) or right (for right-to-left presentation) margin of the current text line and takes indentation into account.

-dita-use-conref-target
See STUB CONTENT for more information.

For `<table>`, in place of the `@expanse` attribute that is used by other DITA elements, the `@pgwide` attribute is used in order to conform to the OASIS Exchange Table Model.

Some processors or output formats might not support all values.

@expiry (date attributes)
Specifies the date when the information should be retired or refreshed. The date is specified using the ISO 8601 format: *YYYY-MM-DD*, where *YYYY* is the year, *MM* is the month (01 to 12), and *DD* is the day (01-31).

@format (link-relationship attributes)
Specifies the format of the resource that is referenced. See STUB CONTENT for detailed information on supported values and processing implications.

@frame (display attributes)
Specifies which portion of a border surrounds the element. The following values are valid:

all
Indicates that a line is rendered at the top, bottom, left, and right of the containing element.

bottom
Indicates that a line is rendered at the bottom of the containing element.

none
Indicates that no lines are rendered.

sides
Indicates that a line is rendered at the left and right of the containing element.

top
Indicates that a line is rendered at the top of the containing element.

topbot
Indicates that a line is rendered at the top and bottom of the containing element.

-dita-use-conref-target
See STUB CONTENT for more information.

Some processors or output formats might not support all values.

@golive (date attributes)
Specifies the publication or general availability (GA) date. The date is specified using the ISO 8601 format: *YYYY-MM-DD*, where *YYYY* is the year, *MM* is the month (01 to 12), and *DD* is the day (01-31).

@href (link-relationship attributes)
Specifies a reference to a resource. See STUB CONTENT for detailed information on supported values and processing implications.

@keycol (simpletable attributes)
Specifies the column that contains the content that represents the key to the tabular structure. If `@keycol` is present and assigned a numerical value, the specified column is treated as a vertical header.

@keyref
Specifies a key name that acts as a redirectable reference based on a key definition within a map. See STUB CONTENT for information on using this attribute.

> **Comment by robander**
> The definiton above for @keyref should be synchronized with the definition in the linked section on keys.
> **Disposition: Unassigned**

@keys
Specifies one or more names for a resource. See STUB CONTENT for information on using this attribute.

@keyscope (common map attributes)
Specifies that the element marks the boundaries of a key scope. See STUB CONTENT for information on using this attribute.

@linking (common map attributes)
Specifies linking characteristics of a topic specific to the location of this reference in a map. If the value is not specified locally, the value might cascade from another element in the map (for cascade rules, see Cascading of metadata attributes in a DITA map).

> **Comment by robander on Dec 28 2021**
> The text below matches 1.3 spec text but I'm nervous about "cannot link" type definition. It's describing how to generate links based on the current context in the map - it's not describing what the topic itself is allowed to link to, which is how I interpret "can".
> **Disposition: Unassigned**

The following values are valid:

targetonly
A topic can only be linked to and cannot link to other topics.

sourceonly
A topic cannot be linked to but can link to other topics.

normal
A topic can be linked to and can link to other topics. Use this to override the linking value of a parent topic.

none
A topic cannot be linked to or link to other topics.

-dita-use-conref-target
See STUB CONTENT for more information.

@name (data-element attributes)
Defines a unique name for the object.

> **Comment by robander**
> Do we need to specify the scope of "unique" here?
> **Disposition: Unassigned**

@otherrole
Specifies an alternate role for a link relationship when the `@role` attribute is set to "other".

@parse (inclusion attributes)
Specifies the processing expectations for the referenced resource. Processors must support the following values:

text

The contents should be treated as plain text. Reserved XML characters should be displayed, and not interpreted as XML markup.

xml

The contents of the referenced resource should be treated as an XML document, and the referenced element should be inserted at the location of the `<include>` element. If a fragment identifier is included in the address of the content, processors must select the element with the specified ID. If no fragment identifier is included, the root element of the referenced XML document is selected. Any grammar processing should be performed during resolution, such that default attribute values are explicitly populated. Prolog content must be discarded.

It is an error to use `parse="xml"` anywhere other than within `<foreign>` or a specialization thereof.

Processors may support other values for the `@parse` attribute with proprietary processing semantics. Processors should issue warnings and use `<fallback>` when they encounter unsupported `@parse` values. Non-standard `@parse` instructions should be expressed as URIs.

Note Proprietary `@parse` values will likely limit the portability and interoperability of DITA content, so should be used with care.

@processing-role (common map attributes)
Describes the processing role of the referenced topic. The processing default is "normal". If no value is specified but the attribute is specified on a containing element within a map or within the related-links section, the value will cascade from the closest containing element. The following values are valid:

normal
Normal topic that is a readable part of the information.

resource-only
The topic is used as a resource for processing purposes. This topic should not be included in a rendered table of contents, and the topic should not be rendered on its own.

-dita-use-conref-target
See STUB CONTENT for more information.

@relcolwidth (simpletable attributes)
Specifies the width of each column in relationship to the width of the other columns. The value is a space-separated list of relative column widths. Each column width is specified as a positive integer or decimal number followed by an asterisk character.

For example, the value `relcolwidth="1* 2* 3*"` gives a total of 6 units across three columns. The relative widths are 1/6, 2/6, and 3/6 (16.7%, 33.3%, and 50%). Similarly, the value `relcolwidth="90* 150*"` causes relative widths of 90/240 and 150/240 (37.5% and 62.5%).

@role
Specifies the role that a linked topic plays in relationship with the current topic.

For example, in a parent/child relationship, the role would be "parent" when the target is the parent of the current topic, and "child" when the target is the child of the current topic. This can be used to sort and classify links when rendering.

The following values are valid:

ancestor
Indicates a link to a topic above the parent topic.

child
Indicates a link to a direct child such as a directly nested or dependent topic.

cousin
Indicates a link to another topic in the same hierarchy that is not a parent, child, sibling, next, or previous.

descendant
Indicates a link to a topic below a child topic.

friend
Indicates a link to a similar topic that is not necessarily part of the same hierarchy.

next
Indicates a link to the next topic in a sequence.

other
Indicates any other kind of relationship or role. The type of role is specified as the value for the `@otherrole` attribute.

parent
Indicates a link to a topic that is a parent of the current topic.

previous
Indicates a link to the previous topic in a sequence.

sibling
Indicates a link between two children of the same parent topic.

-dita-use-conref-target
See STUB CONTENT for more information.

@rowheader (complex table attributes)
Specifies whether the entries in the respective column are row headers. The following values are valid:

firstcol
Indicates that entries in the first column of the table are row headers. This applies when the `@rowheader` attribute is specified on the `<table>` element.

headers
Indicates that entries of the column that is described using the `<colspec>` element are row headers. This applies when the `@rowheader` attribute is specified on the `<colspec>` element.

norowheader
Indicates that entries in the first column are not row headers. This applies when the `@rowheader` attribute is specified on the `<table>` element.

-dita-use-conref-target
See STUB CONTENT for more information.

Note This attribute is not part of the OASIS Exchange Table Model upon which DITA tables are based. Some processors or output formats might not support all values.

The `@rowheader` attribute is available on the following table elements: `<table>` and `<colspec>`.

@rowsep (complex table attributes)
Specifies whether to render row separators between table entries. The following values are valid: "0" (no separators) and "1" (separators).

The `@rowsep` attribute is available on the following table elements: `<colspec>`, `<entry>`, `<row>`, `<table>`, and `<tgroup>`.

@scale (display attributes)
Specifies the percentage by which fonts are resized in relation to the normal text size. The following values are valid: "50", "60", "70", "80", "90", "100", "110", "120", "140", "160", "180", "200", and -dita-use-conref-target.

This attribute is primarily useful for print-oriented display. Some processors might not support all values.

If the `@scale` attribute is specified on an element that contains an image, the image is not scaled. The image is scaled **only** if a scaling property is explicitly specified for the `<image>` element.

@scope (link-relationship attributes)
Specifies the closeness of the relationship between the current document and the referenced resource. The following values are valid: "local", "peer", "external", and "-dita-use-conref-target". See STUB CONTENT for detailed information on supported values and processing implications.

@search (common map attributes)
Specifies whether the target is available for searching. If the value is not specified locally, the value might cascade from another element in the map (for cascade rules, see Cascading of metadata attributes in a DITA map). The following values are valid: "yes", "no", and "-dita-use-conref-target".

@specializations (architectural attributes)
Specifies the attribute domain specializations that are included in the document-type shell. This attribute is set as a default within the document-type shell. The value varies depending on what domains are integrated into the document-type shell. For example, a grammar file that includes the specialized attributes `@audience`, `@deliveryTarget`, and `@newBaseAtt` would set the value to `@props/audience @props/deliveryTarget @base/newBaseAtt`.

@title-role (REQUIRED)
Specifies the role that the alternative title serves. Multiple roles are separated by white space. The following roles are defined in the specification: "linking", "navigation", "search", "subtitle", and "hint".

Processors can define custom values for the `@title-role` attribute.

@toc (common map attributes)
Specifies whether a topic appears in the table of contents (TOC) based on the current map context. If the value is not specified locally, the value might cascade from another element in the map (for cascade rules, see Cascading of metadata attributes in a DITA map). The following values are valid:

yes
The topic appears in a generated TOC.

no
The topic does not appear in a generated TOC.

-dita-use-conref-target
See STUB CONTENT for more information.

@type (link-relationship attributes)
Describes the target of a reference. See STUB CONTENT for detailed information on supported values and processing implications.

@value (data-element attributes)
Specifies a value associated with the current property or element.

@valign (complex table attributes)
Specifies the vertical alignment of text in table entries. The following values are valid:

bottom
Indicates that text is aligned with the bottom of the table entry.

middle
Indicates that text is aligned with the middle of the table entry.

top
Indicates that text is aligned with the top of the table entry.

-dita-use-conref-target
See STUB CONTENT for more information.

The `@valign` attribute is available on the following table elements: `<entry>`, `<tbody>`, `<thead>`, and `<row>`.

@xml:space
Specifies how to handle white space in the current element. This attribute is provided on `<pre>`, `<lines>`, and on elements specialized from those. It ensures that parsers respect white space that is part of the data in those elements, including line-end characters. When defined, it has a fixed value of "preserve", making it a default property of the element that cannot be changed or deleted by authors.

@xmlns:ditaarch (architectural attributes)
Declares the default DITA namespace. Although this is technically a namespace rather than an attribute, it is specified as an attribute in the DTD-based grammar files that are distributed by OASIS. The value is fixed to "http://dita.oasis-open.org/architecture/2005/".

C Formatting conventions

Although how DITA elements are formatted is ultimately implementation-specific, certain conventions are common.

Element	Suggested formatting
<chdeschd>	Apply bold highlighting to the contents of the <chdeschd> element.
<choicetable>	Unless the @keycol attribute is set to "0", processors typically apply bold highlighting to the contents of the "Option" column.
<choptionhd>	Apply bold highlighting to the contents of the <choptionhd> element.
<codeblock>	Use a monospaced font for the contents of the <codeblock> element.
<codeph>	Use a monospaced font for the contents of the <codeph> element.
<menucascade>	Separate <uicontrol> elements with a character to represent the menu cascade.
<numcharref>	Surround the contents of the <numcharref> element with a leading ampersand (&) and a trailing semi-colon (;).
<parameterentity>	Surround the contents of the <numcharref> element with a leading percentage sign (%) and a trailing semi-colon (;).
<screen>	Enclose the contents of the <screen> element with a box to suggest a computer display screen.
<shortcut>	Highlight the keyboard shortcut with underlining.
<syntaxdiagram>	Traditionally, the syntax diagram is formatted with "railroad tracks" that connect the units of the syntax together, but the presentation might differ depending on the output media.
<textentity>	Surround the contents of the <textentity> element with a leading ampersand (&) and a trailing semi-colon (;).
<var>	Apply italic highlighting to the contents of the <var> element.
<xmlatt>	Precede the contents of the <xmlatt> element with a commercial at symbol (@).
<xmlelement>	Surround the contents of the <xmlelement> element with leading (<) and trailing (>) angle brackets.
<>	

Index

T

U

V