DITA Technical Committee

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"

Robert Anderson 23 January 2023

How about just "describe"?

Tammy Crowley 23 January 2023

I agree with "define and describe".

Is it important to note that they can be used for examples?

Robert Anderson 21 March 2023

Updating to "describe". Reasoning -

- Keeping it simpler / fewer words when we can
- In the normal case – the syntax for a programming domain is *defined* somewhere else, while the DITA markup is used to document or describe it.

Per Tammy's additional comment - dropping the line about examples, I agree that's not particularly important here.

Disposition: Completed

1.1 <apiname>

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

Comment by Bill Burns
Suggest "The API name identifies…"

Robert Anderson 23 January 2023

"An API name is the name...

Tammy Crowley 23 January 2023

"The API name identifies an application programming interface, such as..."

Robert Anderson 23 January 2023

Per our current style, we are trying to update short descriptions to say what the component is rather than what the XML element does ("This is *something...*" rather than "This element *does thing...*")

Frank Wegmann 24 January 2023

"...such as the name of a class or method in Java."

Disposition: Completed

Specialization hierarchy

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

Comment by tammy-PCAS
The <apiname> element is specialized...

Note: the word "element" is missing from all the programming domain specialization hierarchy sections.

January 23, 2023

Thanks, updating them all

Disposition: Completed

Attributes

The following attributes are available on this element: universal attributes and `@keyref`.

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.

Comment by Bill Burns
"Code block identifies…"

Robert Anderson 23 January 2023

"A code block is a set of lines from a program"

Tammy Crowley 23 January 2023

"A code block is used to list lines of program code."

Robert Anderson 23 January 2023

As with apiname - going with the current style that says what the content is, rather than what the element does

Disposition: Completed

Rendering expectations

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.

Tammy Crowley, 17 January 2023

If monospace is the recommended font, I would note it here with a reference to the "Formatting conventions" topic.

Robert Anderson, 18 January 2023

Per discussion at the TC on 17 January, the formatting information will be moved here, and included by reference in an appendix collecting formatting/rendering information.

Stan Doherty, Jan 23

If we specify processing behavior for spaces and line breaks, we ought to specify it for tabs.

Robert Anderson 23 January 2023

I would expect "tabs" to come under spaces, when preserving spaces - it covers all types of space

---

Robert Anderson 21 March 2023

Marking closed, the rendering section now states that content is typically monospace.

Disposition: Completed

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, universal attributes, and `@xml:space`.

Example

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

Comment by Zoë Lawson on 22 January 2023
Fix *tage* to *tag*.

This might be a bit circular, but is it worth mentioning that the code samples in this specification are formatted in codeblocks? That the spec itself is an example?

Robert Anderson 23 January 2023

Thanks for the typo catch!

For the other bit - I don't think so? Anyone reading is likely to guess this already, and I shy away from it for the same reason I don't generally like to put in DITA markup and then have the spec say "Here is an example of formatting" – we don't actually know how future tools will render it.

Disposition: Completed

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

Comment by Stan Doherty
Given the amazing things that DITA-OT does with <coderef>, I suggest adding and example of <codeblock> with an embedded <coderef>. That's how I manage code samples and examples. They are separately-managed, independently updated text files.

Robert Anderson 23 January 2023

Sounds good

Robert Anderson, June 1 2023

We already have a sample of coderef in that element topic, but I do think it's worth highlighting for anyone in this topic; rather than create a second one, I've added a paragraph below this code sample that suggests viewing the coderef element to see how it can be used to embed samples.

Disposition: Completed

Comment by Zoë Lawson on 22 January 2023
Per request, instead of this generic code snippet, would it be worth showing an XSLT example? (I think this would work, but I'm probably terribly wrong.)

```
<codeblock>
<xsl:template match="*[contains(@outputclass,'green')"]>
    <xsl:attribute name="color">#006400;</xsl:attribute>
</xsl:template>
</codeblock>
```

Robert Anderson 23 January 2023

Combining with Stan's comment, we could use an XSLT example with coderef

Robert Anderson, June 1 2023

Used the suggeested XSLT template as the example

Disposition: Completed

1.3 <codeph>

The `<codeph>` element identifies a code snippet.

Comment by tammy-PCAS
"A code phrase is used to identify a snippet of code." OR if considered acceptable, "A code phrase (<codeph>) is used to identify a snippet of code."

Nancy Harrison 23 January 2023

'snippet' is a pretty colloquial term. What about 'a small piece' rather than 'snippet'? If so, we'd want to change the instances of 'snippet' in the Example section as well.

Robert Anderson, January 23 2023

We need to follow the current style of saying what the component is - "A code phrase is a ...". Not sure about the word "snippet", we've been using that in this definition for a long time.

For now, going with the version Kris sent in email to the TC last week - "A code phrase is a small portion of source code, machine code, or text that is displayed in-line."

Disposition: Completed

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

Robert Anderson, 17 January 2023

It is already listed in the Formatting Expectations in the appendix.

Robert Anderson, 18 January 2023

Per discussion at the TC on 17 January, the formatting information will be moved here, and included by reference in an appendix collecting formatting/rendering information.

Robert Anderson 23 January 2023

Will update to include

Robert Anderson 21 March 2023

Formatting info now added to a rendering section here

Disposition: Completed

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 and `@keyref`.

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.

Comment by Stan Doherty
What does "literal code" mean here? What someone *really* needs to know is that the referenced document needs to be a text file. Binary files could contain code samples, but they would not be processed correctly with <coderef>.

Robert Anderson 23 January 2023

Agree that "literal" is a bit odd; for now, going with "...references an external text file that contains programming code", which should cover the comment above.

Frank Wegmann 24 January 2023

I would use the same wording as for codeblock: "...that contains [lines of] program code."

Robert Anderson 24 January 2023

Nice catch on the consistency there, switching from "programming code" to "program code"

Disposition: Completed

Usage information

Comment by tammy-PCAS
Should we note that <coderef> is contained by <codeblock>? Or is "contained by" info captured elsewhere?

Robert Anderson, 23 January 2023

That will be described in an appendix (and the actual definition is in the grammar file), so should not be restated here.

Disposition: Closed

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.

Robert Anderson, 17 January 2023

This element is only allowed inside of codeblock, which already specifies normative rules for spacing and includes the monospace formatting convention - I don't think it is necessary to double-up here.

Robert Anderson 23 January 2023

Will leave this out - it is already inside of codeblock which means it has to be monospace

Disposition: Closed

Comment by Zoë Lawson on 22 January 2023
I am concerned about the phrase "displayed inline" because I want to confirm the behavior. To me that means the content will be inserted where ever that is, with no line breaks or blocks around it. If that's correct, that's fine.

Do we need to mention that target code line breaks should be handled? In the rendering expectations, should the handling of non-XML characters be an official "must"? (since otherwise the DITA would break an XML processor)?

Robert Anderson 23 January 2023

This element can only ever be used inside of a codeblock element, so how about replacing "displayed inline" with "displayed in the codeblock"?

We should not need MUST here. As you noted, tools will break without it - which means it's not an interoperability thing, it's just a "this is how it works" thing, which does not require a normative rule.

Zoë Lawson 23 January 2023

I didn't realize this could only work inside of a codeblock. Makes sense.

Disposition: Closed

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, link-relationship attributes, universal attributes, and `@keyref`.

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.

Comment by Robert D Anderson
The element name above uses <codref> which should be <coderef>
Disposition: Completed

```
<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."

Tammy Crowley, 17 January 2022

I would use <parname> to reference an api parameter. I would use <option> when listing command line options only.

Nancy Harrison 23 January 2023

My expectation would be the same as Tammy's; to me, a parameter goes with an API; and option goes with a command. That being the case, the list of keywords below seems a bit weird to me, including, as it does, programming options (which I would use <parmname> for rather than <option>.

Robert Anderson, 23 January 2023

I don't think we have consensus on this one, and I suspect there is not uniform usage. I personally use parmname to hold the name of API parameters and CLI parameters, while I use option to hold the values that are specified with the parameter.

Kris Eberlein, 24 January 2023

Here's what Robert and I have been discussing:

<cmdname>
: A command name is the name of a software command.
: Example: Use the `dita` command to ...

<parmname>
: A parameter name is the name of a parameter that is passed to an API or a command-line interface (CLI).
: Example: Use the args.bookmap-order parameter to specify if the frontmatter and backmatter content order is retained.

<option>
: The <option> element specifies a permitted value for a parameter (or configuration).
: Example: The allowed values are retain and discard; the default value is discard.

The DITA-OT documentation provides solid and consistent usage of these elements that matches the meaning that I outlined.

Kris Eberlein, 01 February 2023

We discussed this at the DITA TC call yesterday. Folks were comfortable with implementing the changes that we outlined in this e-mail. I've made the changes to the short descriptions for `<parmname>` and `<option>`, as well as completely reworked the examples for `<option>`. @Robert, please check the changes, make edits if necessary, and then mark this comment as "Completed".

Robert Anderson 21 March 2023

Done

Disposition: Completed

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 and `@keyref`.

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>`

Stan Doherty 23 January 2023

Do we need to make a distinction for users between the types of "programming" elements here? Some like <parmname> are really CLI parameters whilst others like <apiname> are REST or Java. If my product has both CLI and REST interfaces, the distinction is meaningful.

Robert Anderson, 1 June 2023

Based on TC discussion around this, updated to: "A parameter name is the name of a parameter that is passed to an API or a command-line interface (CLI)."

Disposition: Closed

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 and `@keyref`.

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>
```

```
 Comment by
               Zoë Lawson on
                   22 January 2023
If we're following the distinction that <parmname> goes with <apiname>, shouldn't config in
 this example be using <apiname> instead?
```

```
Robert Anderson 21 March 2023

After considerable discussion, we are not making that distinction; parmname can have a
 parameter for an API or a command, both are in wide use today.


             Disposition: Closed
```

```
 Comment by
               Robert D Anderson
Should change /env to -env to be more platform agnostic, as suggested in other examples.
```

```
Robert Anderson 21 March 2023

Done


             Disposition: Completed
```

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.

Bill 2023-01-20: But maybe "specialized" isn't really necessary.

Robert Anderson 23 January 2023

I think "specialized" is more useful here than usual because we already have another element (dl) that says it "is a definition list" - so without that distinction here we'd be saying that parml is the same thing.

Frank Wegmann 24 January 2023

I agree with Robert here.

Disposition: Completed

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 and `@compact`.

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 …"

Zoë Lawson 22 January 2023

Fix the comma to a period at the end of the sentence.

Robert Anderson 23 January 2023

Thanks for both of those

Tammy Crowley 23 January 2023

"...contains one or more parameter terms and definitions." Can a parameter list entry have more than one parameter term and definition?

"A parameter list entry contains one parameter term and one definition." OR "A parameter list entry (<plentry>)..."

Robert Anderson 23 January 2023

Yes - it can have more than one term / definition. Using the original suggestion above, and fixing the typo.

Disposition: Completed

Specialization hierarchy

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

Comment by tammy-PCAS
The <plentry> element is specialized...
Disposition: Completed

Attributes

The following attributes are available on this element: universal attributes.

Example

See `<parml>` (12).

1.9 <pt>

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

Comment by tammy-PCAS
"The parameter term specifies the name of a parameter in an application programming interface. The parameter term is part of a parameter list entry." OR if acceptable, "The parameter term (<pt>) specifies the name of a parameter in an application programming interface. The parameter term is part of a parameter list entry (<plentry>)."

Robert Anderson, 23 January 2023

As noted in the option topic, I don't think we have consensus that this is only for APIs - I think that would render many existing usages invalid. Using the smaller update Kris suggested last week in TC email for now, can reassess later as we solidify on the option/parmname distinctions.

Disposition: Closed

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 and `@keyref`.

Example

See `<parml>` (12).

1.10 <pd>

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

Comment by tammy-PCAS
"The parameter definition specifies the definition of a parameter term. The parameter definition is part of a parameter list entry." OR if acceptable, "The parameter definition (<pd>) specifies the definition of a parameter term (<pt>). The parameter definition is part of a parameter list entry (<plentry>).

Robert Anderson, 23 January 2023

Updating to use our existing pattern, "A parameter definition is a definition of a term that is defined in a parameter-list entry."

Disposition: Closed

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.

Example

See `<parml>` (12).

2 Software domain

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

Comment by Stan Doherty
Recommend "used to describe the *presentation and* operation". Some of these have nothing to do with functional operation.

Robert Anderson 23 January 2023

Sounds good

Disposition: Completed

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

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.

Robert Anderson, 18 January 2023

Probably not something to change at this point, as you noted. I don't remember all of the discussion about how these were split, but I do remember that messages were considered to be something wider than just software, so in that ideal world where many configured shells exist and domains flow freely, you might need to describe messages that appear when operating some hardware but not be doing anything related specifically to software.

Disposition: Closed

2.1 <msgph>

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

Comment by tammy-PCAS
"The message phrase (<msgph>) identifies the text of a message that is..."

Robert Anderson 23 January 2023

As discussed at the TC last week, current style is to not repeat the element name with brackets in the short description.

Robert Anderson 21 March 2023

Updated to use our current style, "A message phrase is the tet of a message that is produced by an application or program."

Disposition: Completed

Rendering expectations

Comment by Kristen James Eberlein on 18 January 2023
Should this topic have a "Rendering expectations" section that states that the content of this element typically is rendered in a monospaced font? That would be parallel with the <codeph> topic.

Robert Anderson, 18 January 2023

Is rendering the message itself in monospace the normal expecation? Not sure about that. I'm more likely to expect the message number as monospace, so that you can clearly read every character of the number

---

Zoë Lawson 22 January 2023

I can see people wanting msgph to render in a font similar to whatever the application represents the message. I wouldn't think it's always monospace.

Robert Anderson 23 January 2023

Will not add this, I think Zoe's explanation is a good one

Disposition: Closed

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 and `@keyref`.

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.

Comment by Bill Burns
This lacks the context that `<megph>` has. Suggest adding "that is produced by an application or program."

Zoë Lawson 22 January 2023

For these messages, do we want to limit to application or program? I can envision hardware systems using this as well. Microwaves and printers can give messages these days. Use "application or device" instead? I realize we're in the software domain, but no reason to limit to just software.

Robert Anderson 23 January 2023

I like "that is produced by an application or device"

Tammy Crowley 23 January 2023

"A message block (<msgblock>) contains a multi-line message or a set of messages that is produced by an application or device."

Robert Anderson 23 January 2023

As noted at the TC and in other comments, current style is to not include the element name with brackets in the short description.

Disposition: Completed

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

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.

Robert Anderson, 18 January 2023

Per discussion at the TC on 17 January, the formatting information will be moved here, and included by reference in an appendix collecting formatting/rendering information.

In this case we don't say anything about msgblock in the formatting topic, but given the ancestry as <pre> this sounds reasonable to state.

Robert Anderson 21 March 2023

Rendering section now includes the "typically monospaced" statement.

Disposition: Closed

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, universal attributes, and `@xml:space`.

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).

Robert Anderson, 17 January

I agree with the distinction, but do not think we can use "identifies the identifier of a message". Need to reword it a bit.

Zoë Lawson 22 January 2023

"demarcates the identifier of a message that is produced by an application or device"?

Stan Doherty, Jan 23

"The <msgnum> element provides an alpha-numeric identifier . . . "?

Robert Anderson 23 January 2023

Rephrasing to use natural language handles some of the problem I had earlier, "A message number is the identifier of a message that is produced by an application or device."

Frank Wegmann 24 January 2023

"A message number identifies a message [that is] produced by an application or device." Whether or not that number takes the form of a number or an alphanumeric string or a sequence of Unicode code points is not the concern of the spec, I think. What do others think of "produced" here? It is not so much an act of production, but of utterance. "emit" comes to mind, but is not used elsewhere, I'm afraid.

Robert Anderson 24 January 2023

@Frank - agree that we do not want to limit it to the characters allowed. We have not done so before, and it's possible that something could use a glyph as the identifier. Perhaps the element would have been better named msgid for that case, but we don't want to change that at this point.

Robert Anderson 21 March 2023

Closing, with: "A message number is the identifier of a message that is produced by an application or program."

Disposition: Completed

Rendering expectations

Comment by Kristen James Eberlein on 18 January 2023
Should this topic have a "Rendering expectations" section that states that the content of this element typically is rendered in a monospaced font? That would be parallel with the <codeph> topic.

Robert Anderson, 18 January 2023

I think we should double check if this is the TC consensus before updating. I do think this is my normal expectation but not sure if it is universal/expected, I think I've seen it rendered as bold as well.

Robert Anderson 23 January 2023

Per discussion with Kris, deciding not to update this one as monospace

Disposition: Closed

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 and `@keyref`.

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.

Comment by tammy-PCAS
"A command name (<cmdname>) identifies a software command."

Robert Anderson 23 January 2023

Using our set pattern: "A command name is the name of a software command."

Disposition: Completed

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 and `@keyref`.

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.

Comment by Zoë Lawson on 22 January 2023
I realize we are in the software domain, but do we need to limit to software applications? I often use varname whenever I need to identify something that changes, such as in an example path `C:\Users\`*`UserName`*`\Documents`.

Stan Doherty, 23 January 2023

Ditto.

Robert Anderson 23 January 2023

I've also seen it used in file paths, I know others have too.

Tammy Crowley 23 January 2023

"Variable name (<varname>) identifies a variable to be supplied to a software application or a file path."

Zoë Lawson 23 January 2023

I'm still concerned about limiting it at all. I could see myself using varname around a code I had to enter in to a copy machine to be able to use it. "Variable name (<varname>) identifies content that changes such as variables supplied to a software application or a user-defined path." ?

Frank Wegmann 24 January 2023

As with <parmname>, you could have a broader understanding of variable in the sense of a value of varying nature. In software, think of a variable as a simple placeholder for value. There are programming languages with a variable concept where a variable, once a value has been assigned to it, no longer changes, behaving as what others would describe as a constant. That is probably not the way, most people would think about it

So your mileage may vary here (sic!). But then... I would also use <varname> inside a file path to indicate the *replaceable* portions. But DITA doesn't have <replaceable> (DocBook has).

Robert Anderson 21 March 2023

Using a combination of the suggestions above: "A variable name is a placeholder for content that might change based on how something is used, such as a variable supplied to a software application or a user-defined path in a command."

Disposition: Completed

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 and `@keyref`.

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.

Robert Anderson, 17 January

Agree, good idea

              Disposition: Completed
```

2.6 <filepath>

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

Comment by Stan Doherty
FWIW – <filepath> can also capture hierarchies to programming endpoints or database-driven structures. We do not have a generic element for hierarchy-thingie-path, so <filepath> may suffice for those use cases.

Robert Anderson 23 January 2023

I don't have a problem with using it that way, but I think it would be odd to define an element called "filepath" as something that is not a file path. Definitely needs reworking as natural language.

Robert Anderson 21 March 2023

Made it more general with the natural language rewrite: "A file path is the location of a resource, such as the system path and file name of a file on a storage device."

Disposition: Completed

Rendering expectations

Comment by Kristen James Eberlein on 18 January 2023
Should this topic have a "Rendering expectations" section that states that the content of this element typically is rendered in a monospaced font? And rendered inline?

Robert Anderson, 18 January 2023

This seems like a good update to me (the monospace bit). It's specialized from phrase, so I do not think we would normally need to call out that it is an inline element?

Robert Anderson, 21 March 2023

Added

Disposition: Completed

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 and `@keyref`.

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>
```

Comment by
Zoë Lawson
check the placing of the bold tag around the </filepath> for the second filepath element.
The leading < is excluded by accident.

Robert Anderson 23 January 2023

Thanks

Robert Anderson, 21 March 2023

Fixed

Disposition: Completed

2.7 <userinput>

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

Comment by Zoë Lawson on 22 January 2023
Can we just end with "a user enters."? Or "application or device"? I can see using this when explaining how to interact with a printer.

Robert Anderson 23 January 2023

Also needs reworking to use natural language. User input is text that a user enters.

I like the sound of "in response to" but being quite pedantic, you could be entering the text to initiate something (not in response to something)

Robert Anderson 21 March 2023

Going with: "User input is text that a user enters, such as a response to an application or system prompt."

While reworking for that, it occurs to me that interacting with a printer is interacting with an application of some sort (just not one that has a GUI on a monitor). Also, changing from the current "types in response to..." to "such as a response to..." means that those are only examples, and we no longer exclude other types of interactions that might not as clearly fit.

Disposition: Completed

Comment by Robert D Anderson on 23 January 2023
Need to remove extra space in the title
Disposition: Closed

Rendering expectations

Comment by Kristen James Eberlein on 18 January 2023
Should this topic have a "Rendering expectations" section that states that the content of this element typically is rendered in a monospaced font? And rendered inline?

Robert Anderson, 18 January 2023

Not sure, is the monospace formatting what everyone expects here? I don't have a strong association one way or another.

It's specialized from phrase, so I do not think we would normally need to call out that it is an inline element?

Zoë Lawson 22 January 2023

I don't think this has to be monospaced. I'm using a template where things the user enters are formatted bold and purple, not monospaced.

Robert Anderson 23 January 2023

No update needed

Disposition: Closed

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 and `@keyref`.

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>.

Robert Anderson, 21 March 2023

Marking closed for the same reasoning. I think you could make a case for either one here, but I'd be more inclined to use <cmdname> if it was "Use the [dir] command to …" or even "Type the [dir] command to..." but in this case it is literally an instruction to the reader to enter text at a prompt.

Disposition: Closed

```
<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.

Comment by Zoë Lawson on 22 January 2023
Similar to the option/paramname question, what is the difference between systemoutput and msgph? (Somehow I missed that this element existed, and I've always used some flavor of msgph/block.)

Robert Anderson 23 January 2023

I've always considered msgph to be specifically related to an error or info message, but this one is not. You could say that all of it comes out under systemoutput, so maybe msgph is a subset of that? But per the description above I think it's expected to be more - expected system response to something you are doing.

Robert Anderson 21 March 2023

Updated: "System output is content that a computer or device generates in response to a command or situation."

Disposition: Completed

Rendering expectations

Comment by Kristen James Eberlein on 18 January 2023

Should this topic have a "Rendering expectations" section that states that the content of this element typically is rendered in a monospaced font? And rendered inline?

Robert Anderson 23 January 2023

I don't think so - it's a phrase-based element so default to inline, and display would probably want to mimic whatever is displaying (which may not be monospace)

Disposition: Closed

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 and `@keyref`.

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 Stan Doherty
Should qualify this to "graphical user interface". CLI and API user interfaces are handled in separate sections.

Robert Anderson 23 January 2023

Sounds good

Nancy Harrison 23 January 2023

That may have been true when this was first created, but I would be inclined to expand it to '… of a software program or a device'. These days, many authors will not be thinking of a software program per se.

Bill Burns 24 January 2023: Agree with Nancy.

Kris Eberlein, 25 January 2023

I disagree. Look at the five elements in this domain; four of the five explicitly apply to software: <wintitle>, <menucascade>, <shortcut>, and <screen>. Beginning with DITA 2.0, we have the hardware control domain, and <hwcontrol> is the equivalent of <uicontrol> for hardware.

Software is software, regardless of what platform it runs on: Desktop, laptop, tablet, or phone. Yes, the lines are blurring with the introduction of viewing ports integrated into appliances (TV on the fridge or embedded in a hotel room mirror), but I don't think we do the spec any favors by blurring the lines.

Disposition: Completed

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

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.

Comment by nancylph

The UI Control denotes user interface controls, such as names of buttons, fields, menu items, and other objects that enable users to control a graphical user interface.

Robert Anderson, 22 March 2023

Updated to "A user interface control is a label for an item that allows a person or tool to control an interface, such as a button, field, menu item, or other object."

Disposition: Completed

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 and `@keyref`.

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.

Comment by Bill Burns
A window title highlights the name of a window or dialog.

Robert Anderson, 22 March 2023

Updated as suggested, except using "window title is" rather than "window title highlights"

Disposition: Completed

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 and `@keyref`.

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**.

Rendering expectations

Comment by Kristen James Eberlein on 18 January 2023
I think we need to use a normative "SHOULD" around how processors need to separate the contents of the <uicontrol> elements with a character. Without a character, the contents of <menucascade> become a run on.

Robert Anderson, 18 January 2023

Disagree slightly - it would be wrong to say "SHOULD" render with a > character like we started with in early DITA-OT, but could say something more generic about SHOULD provide some form of separation? That could be a multi-line rendering, could be arrows or graphics, or could be something more creative - we don't want to describe a specific type of separator.

Kris Eberlein, 18 January 2023

That's a good point. While I certainly did not specify WHICH character, I did specify "character". An image is certainly an option and who knows what else. I'm not sure we want to go in the direction of a "multi-line rendering", as I think <menucascade> is an inline element.

Robert Anderson 23 January 2023

Did not mean to suggest multi-line was correct. I'm fine with making a SHOULD statement about separation, just not sure yet how to phrase it. "Visual separation" is also not quite right because the rendering could be audible.

Robert Anderson 22 March 2023

Updated to say "...SHOULD separate the contents of the uicontrol elements *in some manner* to represent the menu cascade. For example, a visual rendering could separate each UI control with an arrow character."

Disposition: Completed

Usage information

Comment by Bill Burns on 24 January 2023

Do we include usage notes? I ask because menucascade frequently causes problems with translation memory tools and segmenting. We discourage our authors from using it.

Kris Eberlein, 25 January 2023

I have not seen this. What TM tools are you using? Is it possible that they have not been properly configured?

Robert Anderson, 22 March 2023

Marking closed

Disposition: Closed

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 and `@keyref`.

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>
```

```
 Comment by
               Zoë Lawson on
                   22 January 2023
We're dating ourselves using "Start > Programs". I'm using Windows 10, and it's now Start >
 Windows Accessories > Notepad. I don't know what Windows 11 does.

Stan Doherty 23 January 2023

How s'bout: [Apple icon] > System Settings > General > Softare Update. These will no change
 btween releases of MacOS.
```

```
Robert Anderson 23 January 2023

Agree it should be updated, would like something general that isn't OS specific if we can
```

```
Kris Eberlein, 25 January 2023

How about a menu cascade from Oxygen? I think all the menu options work the same way on
 MacOS and Windows.


"To perform operations against multiple files, click Find > Find/Replace in files."
```

```
Robert Anderson, 22 March 2023

Updated to use a generic editor switching to dark mode, View -> Editor layout -> Display
 mode -> Dark


              Disposition: Completed
```

3.4 <shortcut>

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

Comment by Bill Burns
A shortcut highlights two or more keys that together perform a menu or window action.

Robert Anderson 22 March 2023

It does not need to be two or more keys (I've almost always seen it used as just one, like <uicontrol><shortcut>F</shortcut>ile</uicontrol> where F is the shortcut.

Updating to: A shortcut is a keyboard shortcut that can perform a menu or window action.

Disposition: Completed

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 and @keyref.

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>
```

Comment by Zoë Lawson on 22 January 2023
Again, this shortcut no longer exists. Better to use <menucascade><uicontrol><shortcut>F</shortcut>ile</uicontrol><uicontrol><shortcut>S</shortcut>ave</uicontrol></menucascade>? Even though that's the example we're using everywhere?

Robert Anderson, 23 January 2023

Agreed, better to update; switched to use the simple File->Save prompt with F and S as shortcuts.

Disposition: Completed

3.5 <screen>

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

Comment by Robert D Anderson on 22 March 2023
Leaving this with the "screen element" construction because "screen" in natural language has many definitions, and none of them are "a container for screen stuff". Using natural language here would make the description too awkward.
Disposition: Closed

Rendering expectations

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.

Robert Anderson 23 January 2023

Done

Disposition: Completed

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, display attributes, and `@xml:space`.

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>
```

Comment by Zoë Lawson
Should we update this to a gitbash or maybe a flip-phone or odometer screen? (Or Zork if we could avoid copyright infringement...)

Stan Doherty, 23 January 2023

Agreed. I can provide a PC BIOS settings screen if you want to change this. I can't think of many 21st-century examples.

Robert Anderson, 23 January 2023

Agree, should update - I like the idea of a git sequence

Robert Anderson, 22 March 2023

Updated to use a git sequence

Disposition: Completed

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.

B 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>` **Comment by Zoë Lawson** fix the xml element name — Robert Anderson, 1 June 2023 This was fixed while moving the content into the rendering section of the element reference topic **Disposition: Completed** 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 (@).

Element	Suggested formatting
`<xmlelement>`	Surround the contents of the `<xmlelement>` element with leading (<) and trailing (>) angle brackets.
`<>`	

Index