Dita_Syntax_Markup

Contents

Chapter 1. Introduction to DITA....3

Chapter 2. Organization of DITA elements....4

Topic elements....4

Map elements....5

Body elements....6

Prolog elements....6

Domain elements....7

Programming domain elements....8

User interface domain elements....14

Software domain elements....16

Utilities domain elements....18

Metadata domain elements....19

Typographic elements....20

Chapter 1. Introduction to DITA

The Darwin Information Typing Architecture (DITA) is an XML-based architecture for authoring, producing, and delivering topic-oriented, information-typed content that can be reused and single-sourced in a variety of ways. While DITA historically has been driven by the requirements of large-scale technical documentation authoring, management, and delivery, it is a standard that is applicable to any kind of publication or information that might be presented to readers, including interactive training and educational materials, standards, reports, business documents, trade books, travel and nature guides, and more.

DITA is designed for creating new document types and describing new information domains based on existing types and domains. The process for creating new types and domains is called specialization. Specialization enables the creation of specific, targeted XML grammars that can still use tools and design rules that were developed for more general types and domains; this is similar to how classes in an object-oriented system can inherit the methods of ancestor classes.

Because DITA topics are conforming XML documents, they can be readily viewed, edited, and validated using standard XML tools, although realizing the full potential of DITA requires using DITA-aware tools.

Related information

Organization of DITA elements *(on page 4)*

Chapter 2. Organization of DITA elements

Elements in DITA are grouped together into a number of categories for organizational and comprehension purposes.

DITA elements can be broadly categorized as follows:

- topic elements
- map elements
- body elements
- prolog elements
- domain elements
- specialisation elements

In fact, this is how the *DITA Language Reference* groups DITA elements. This categorization of the various elements is partly to aid understanding, and partly to make it technically easier for the schema files (DTD and XSD) to be managed.

Elements used within topics can also be differently categorized as block or phrase elements.

Related information

Introduction to DITA *(on page 3)*
Topic elements *(on page 4)*
Map elements *(on page 5)*
Body elements *(on page 6)*
Prolog elements *(on page 6)*
Domain elements *(on page 7)*

Topic elements

The topic elements are the basic structural building blocks of all information types.

The topic elements are the main structural elements of topics. Some topic elements are generic (i.e., inherited from the *topic* proto information type), while others are specific to the concept, task or reference information types.

Examples of topic elements include:

- topic (and concept, task, reference)
- titlealts
- shortdesc
- body (and refbody, conbody, taskbody)
- section
- example
- related-links

Related information

Organization of DITA elements *(on page 4)*

Map elements *(on page 5)*

Body elements *(on page 6)*

Prolog elements *(on page 6)*

Domain elements *(on page 7)*

Map elements

The map elements are the elements used in ditamaps and bookmaps.

The map elements are a small set of elements, some of which have been specialised into other elements for use in bookmaps.

The map elements include:

- map
- topicref
- topicmeta
- topicgroup
- topichead
- reltable

Related information

Organization of DITA elements *(on page 4)*

Topic elements *(on page 4)*

Body elements *(on page 6)*
Prolog elements *(on page 6)*
Domain elements *(on page 7)*

Body elements

The simple block structures within the body of topics are categorized as the body elements.

Body elements are the most common content authoring block elements, and include:

- paragraph
- list
- phrase
- figure

Related information
Organization of DITA elements *(on page 4)*
Topic elements *(on page 4)*
Map elements *(on page 5)*
Prolog elements *(on page 6)*
Domain elements *(on page 7)*

Prolog elements

A topic's metadata is stored in a range of prolog elements.

The DITA prolog elements contain the main metadata for a topic or collection.

The types of information recorded in the prolog include:

- author
- copyright information
- critical tracking dates
- permissions for use/management of the content

- extensive metadata about the content of the document

Related information

Organization of DITA elements *(on page 4)*
Topic elements *(on page 4)*
Map elements *(on page 5)*
Body elements *(on page 6)*
Domain elements *(on page 7)*

Domain elements

The domain elements are comprised of a number of separate sets of elements that relate to specific documentation fields.

Remembering that DITA started life within IBM as a tool for creating software and hardware documentation, it shouldn't be a surprise to discover that DITA's base elements reflect that background.

Elements that relate to a particular field (such as software) are called *domain elements*. The domain elements within DITA are grouped into:

typographical elements

generic word-processor like elements used to highlight text

programming elements

terms and structures related to programming environments

software elements

terms and structures related to the operation of a software program

table elements

elements that relate to table structures

user interface elements

terms and structures related to a software user interface

utilities elements

elements that don't fit anywhere else!

If you are writing a programmer's reference, you will mainly use elements in the programming domain.

If you are writing a mobile phone user guide, you should avoid using programming domain elements, and mainly use user interface domain elements.

The typographical domain elements are designed to be used only when **no semantically-appropriate elements are available** and a formatting effect is required. These elements should therefore only be used as a last resort.

Related information

Organization of DITA elements *(on page 4)*
Topic elements *(on page 4)*
Map elements *(on page 5)*
Body elements *(on page 6)*
Prolog elements *(on page 6)*
Programming domain elements *(on page 8)*
User interface domain elements *(on page 14)*
Software domain elements *(on page 16)*
Utilities domain elements *(on page 18)*
Metadata domain elements *(on page 19)*
Typographic elements *(on page 20)*

Programming domain elements

The elements in the programming domain each have a specific semantic purpose.

Here is a list of elements, their semantic purposes, and an example for each.

apiname

API name

Example: Google Maps

codeblock

code block

Example:

```
dir /s/w/o/p/a:-d
```

codeph

code phrase

Example: `dir read *.*`lists all files in the current directory that begin with read with any extension.

option

one of a set of options

Example: Topic can be a concept, or a task, or a reference. Let's prefer task option here.

parmname

parameter or argument

Example: Use the **env** parameter of the "config" command to update to update a field value.

parml

parameter list

Example:

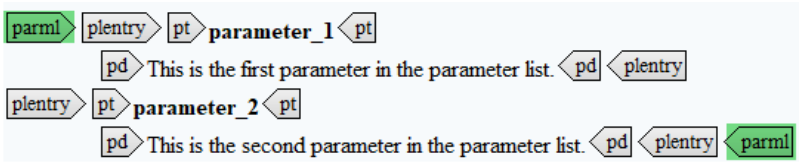
parameter_1

This is the first parameter in the parameter list.

parameter_2

This is the second parameter in the parameter list.

Here's how you can define a parameter list using this tag:



plentry

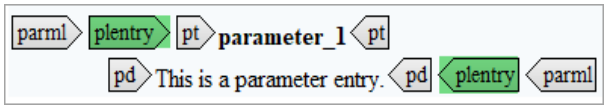
parameter list entry (within parml)

Example:

parameter_1

This is a parameter entry.

Here's how you can use this tag to enter a parameter:



pt

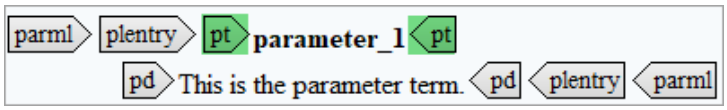
parameter term (within plentry)

Example:

parameter_1

This is the parameter term.

Here's how this tag is used:



pd

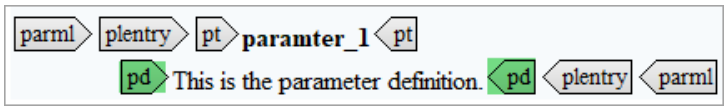
parameter definition

Example:

paramter_1

This is the parameter definition.

Here's how this tag is used:



synph

syntax phrase. It is used when a complete syntax diagram is not required, but some of the syntax elements, such as `<kwd><oper><delim>` are used within the text flow of the topic content.

Example:**format***volumename* Here's how this tag is used:

syntaxdiagram

syntax diagram

Example:

This is the element container for syntax elements.

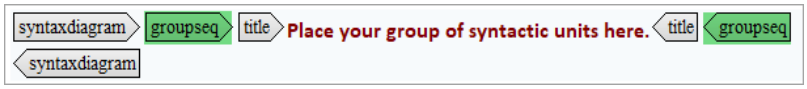
Here's how this tag is used:

groupseq

group of syntactic units (used only for syntax diagrams)

Example:

Here's how this tag is used:



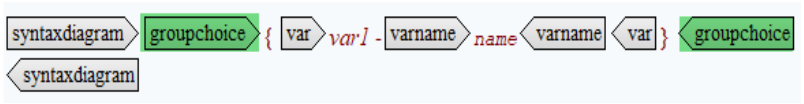
groupchoice

choice of a group of syntactic units (used only for syntax diagrams)

Example:

{ *var1 -name* }

Here's how this tag is used:



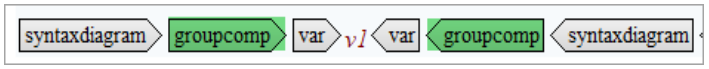
groupcomp

group of composite syntactic units (used only for syntax diagrams)

Example:

v1

Here's how this tag is used:



fragment

fragment of syntax (used only for syntax diagrams)

Example:

(explicit id)

Here's how this tag is used:

fragref

cross-reference to a fragment of syntax diagram

Example:

```
<xyz>
```

Here's how this tag is used:

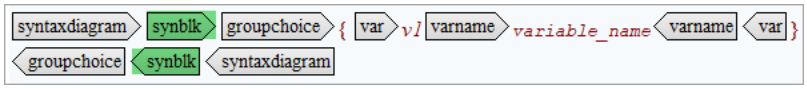
synblk

block of small pieces of a syntax definition into a larger piece.

Example:

```
v1variable_name
```

Here's how this tag is used:



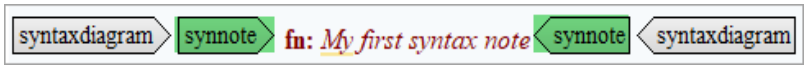
synnote

footnote within syntax (syntax note). Use it to explain aspects of the syntax that cannot be expressed in the markup itself.

Example:

```
(explicit id )
```

Here's how this tag can be used:



synnoteref

cross-reference to a syntax note

Example:

```
[]
```

Here's how this tag can be used:

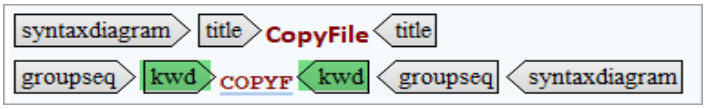
kwd

syntax keyword (used only for syntax diagrams)

Example:

CopyFile

COPYF



var

variable that a user must supply, such as their user name or password (used only for syntax diagrams)

Example:

{ *input-filename* }

Here's how this tag can be used:

oper

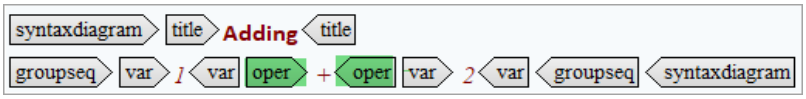
operator character (such as +, -, and =) within syntax

Example:

Adding

1 + 2

Here's how this tag can be used:

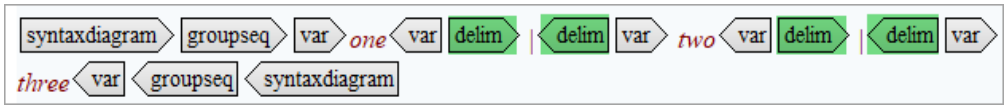


delim

delimiter characters (such as /, |, and ;) to mark the beginning or end of a section within a syntax diagram.

Example:

one | two | three

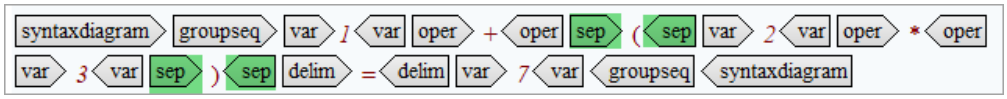


sep

seperator character within syntax

Example:

```
1 + ( 2 * 3 ) = 7
```

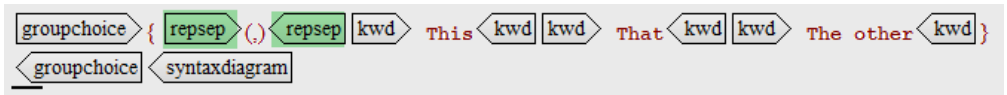


repsep

seperator character for repeated syntax elements

Example:

```
{ | This | That | The other }
```



Related information

Domain elements *(on page 7)*

User interface domain elements *(on page 14)*

Software domain elements *(on page 16)*

Utilities domain elements *(on page 18)*

Metadata domain elements *(on page 19)*

Typographic elements *(on page 20)*

User interface domain elements

The elements in the user interface domain each have a specific semantic purpose.

uicontrol

user interface control element is used to mark up UI controls, such as button names, menu items, and other objects.

Example:

Press the **Submit** (<uicontrol>)button to submit your form.

wintitle

window title

Example: The **Configure Options** window opens with your last set of selections highighted.

menucascade

menu cascade

Example: To open the Notepad, navigate to **Start > Programs > Accessories > Notepad**

shortcut

shortcut

Example: Navigate to **Start > Settings** to open the system settings.

screen

character (text only) screen

```
   File  Edit  Search  View  Options  Help
+-------------------------------- UNTITLED1
 ---------------------------------+
¦ C:\Users\Administrator>screen
'screen' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Administrator>cmd window
Microsoft Windows [Version 10.0.19045.4170]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd..

C:\Users>dir
 Volume in drive C has no label.
 Volume Serial Number is 8888-9999

 Directory of C:\Users

03-08-2023  10:53    <DIR>          .
03-08-2023  10:53    <DIR>          ..
```

```
04-03-2023  15:46                 0 abd
30-03-2024  12:42    <DIR>          Administrator
24-02-2023  18:48    <DIR>          admin_86e45
04-03-2023  16:15    <DIR>          Public
               1 File(s)              0 bytes
               5 Dir(s)  23,563,603,968 bytes free
                                                 ¦
¦
    ¦
¦
    ¦
¦
    ¦
¦  Line:1    Col:1  F1=Help
    ¦
+-------------------------------------------------------------------------
----+
```

Related information

Domain elements *(on page 7)*
Programming domain elements *(on page 8)*
Software domain elements *(on page 16)*
Utilities domain elements *(on page 18)*
Metadata domain elements *(on page 19)*
Typographic elements *(on page 20)*

Software domain elements

The elements in the software domain each have a specific semantic purpose.

Below are the software domain elements, their semantic purposes, and an example for each.

msgph

message phrase

Example: Double-click `Setup.exe` to begin the installation process.

msgblock

message block

Example: <msgblock>

```
<title>Warning</title> <msgph>This action cannot be undone. Are you sure you
 want to proceed?
```

msgnum

message number

Example: 401: `Unauthorized Access`.You don't have permission to access this resource.

cmdname

command name

Example: To compile the source code, use the javac following by the name of the Java source file.

varname

variable (to be provided by user) name

Example: Use *md* command following the directory name to create a new directory.

filepath

file name or path, or URI

Example: Uncompress the `gbbrsh.gz` file to the `/usr` directory.

userinput

user input

Example: Instructions to install a software:

1. Open the terminal window.
2. Type the following command: `sudo apt-get install software-name`.
3. Press Enter to execute the command.
4. Follow the on-screen instructions to complete the installation.

systemoutput

system output

Example: To access the "sri" directory available in the root directory, first type the comamnd `cd..`, to go to the root directory. Use this command until you see this path: `C:\Users>`. Then, type the command cd sri, and hit **Enter** key. The system takes you to the directory: `C:\Users\sri>`.

Related information

Domain elements *(on page 7)*

Programming domain elements *(on page 8)*

User interface domain elements *(on page 14)*

Utilities domain elements *(on page 18)*

Metadata domain elements *(on page 19)*

Typographic elements *(on page 20)*

Utilities domain elements

The elements in the utilities domain each have a specific purpose in defining image map properties.

imagemap

client-side image map

area

hotspot area within an image map

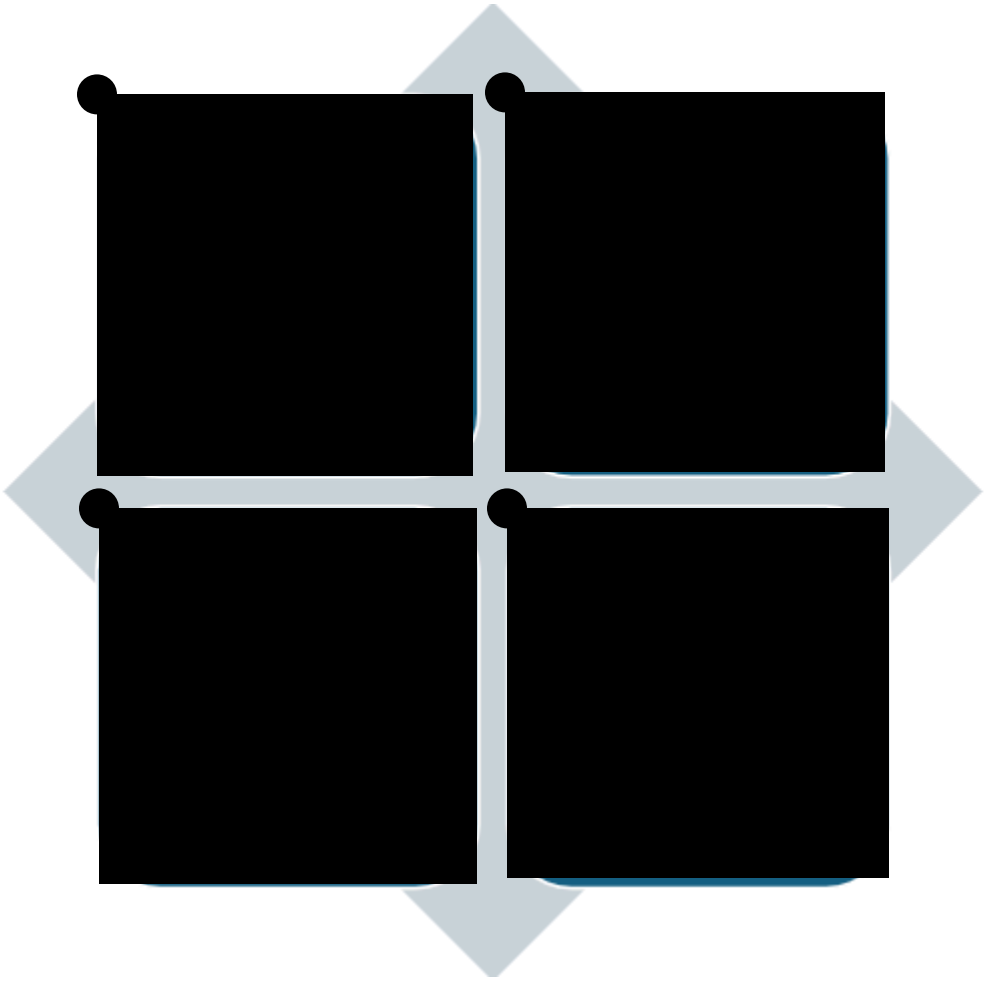
coords

co-ordinates of a hotspot area within an image map

shape

shape of a hotspot area within an image map

Below is the image inserted using the "imagemap" element:



1. Overview *(on page 3)*
2. Click here for "Topic Elements" *(on page 4)*
3. Click here to understand map elements *(on page 5)*
4. Click here to understand prolog elements *(on page 6)*

Related information

Domain elements *(on page 7)*
Programming domain elements *(on page 8)*
User interface domain elements *(on page 14)*
Software domain elements *(on page 16)*
Metadata domain elements *(on page 19)*
Typographic elements *(on page 20)*

Metadata domain elements

Related information

Domain elements *(on page 7)*
Programming domain elements *(on page 8)*
User interface domain elements *(on page 14)*
Software domain elements *(on page 16)*
Utilities domain elements *(on page 18)*
Typographic elements *(on page 20)*

Typographic elements

Related information

Domain elements *(on page 7)*
Programming domain elements *(on page 8)*
User interface domain elements *(on page 14)*
Software domain elements *(on page 16)*
Utilities domain elements *(on page 18)*
Metadata domain elements *(on page 19)*