DITA 1.3 Feature Article
Documenting Troubleshooting Information
An OASIS DITA Adoption Technical Committee Publication

Contents

Part

I

Troubleshooting

Topics:

Troubleshooting semantics became part of DITA with DITA 1.3.

Troubleshooting information type

Troubleshooting information is corrective action that a user can follow to fix a problem.

Troubleshooting topic

DITA 1.3 introduced the troubleshooting topic to model troubleshooting semantics.

Embedded troubleshooting information

Occasionally, small amounts of troubleshooting information are embedded within a task or a description.

DITA 1.3 introduced the following markup to support embedded troubleshooting:

trouble note type	The <note> element now supports a type-value called "trouble" along with its other types, such as "caution", "tip", and "warning".
<steptroubleshooting>	The <step> element now supports an optional <steptroubleshooting> child element as its last child.
<tasktroubleshooting>	The <taskbody> element now supports an optional <tasktroubleshooting> element in between the <result> and <example> elements.

Troubleshooting information type

Troubleshooting as an information type

Troubleshooting is an information type that provides corrective actions for changing the state of a product or a system to a state that is more desirable.

Simple troubleshooting

In its simplest form, troubleshooting information provides corrective actions that follows this pattern:

1. A condition or symptom. Usually the condition or symptom is an undesirable state in a system, a product, or a service that a reader may wish to correct.
2. A cause for the condition or symptom
3. A remedy for the condition or symptom that restores the system, product, or service to its normal state.

Complex troubleshooting

In complex cases, there may be more than one possible cause for a condition or a symptom. When this happens, each cause can be presented along with its associated remedy. These cause-remedy pairs can serve as successive fallbacks (fixes) for users to eliminate an undesirable condition.

A common way to organize these fallbacks is by the likelihood of a cause occurring. However, it might be advantagous to deviate from this order if a less likely cause has a significantly simpler remedy.

Instructions for how to contact a support organization often appear as a final fallback. In those cases, the cause would be unspecified and the remedy would provide contact instructions.

Embedded

Embedded troubleshooting information appears within tasks or descriptions. It is brief. Often, the condition or cause is implied by the information that surrounds it. Remedies can usually be conveyed with a single sentence.

Tip:

Extensive amounts of troubleshooting information ought never be embedded. Use troubleshooting topics instead.

Other corrective action information

Other corrective action information that follows the troubleshooting information type pattern are:

- Alarm clearing. When something goes wrong, a system returns an alarm from a predefined set of alarms.
- Error resolution. When something goes wrong, a system returns an error code from a predefined set of error codes.
- Event response. When a significant event occurs, a system returns an event from a predefined set of events. Some of these events, while not errors, are nonetheless undesirable states.

Troubleshooting topic

The troubleshooting topic models the semantics of the troubleshooting information type.

The troubleshooting topic type provides semantics for simple and complex troubleshooting information. These same semantics can also be applied to other corrective action domains such as alarm clearing procedures.

Troubleshooting topic elements

The <troublebody> element contains the following elements:

<condition> This element is the first child of <troublebody>, and it describes the condition or symptom associated with some undesirable state in a system, a product, or a service. Though an option, this element is typically used. However, in cases where the topic title already connotes the condition, this element should not be used. The content model for <condition> is similar to the content model for the <section> element found in other topic types, such as concept.

<troubleSolution> One or more <troubleSolution> elements must appear in the <troublebody> element. When the <condition> element is used, the <troubleSolution elements follow it. <troubleSolution> is a wrapper element for <cause> and <remedy>, each of which are a cause-remedy pair.

<cause> This optional first-child of <troubleSolution> describes a possible cause for the <condition>. The content model for <cause> is similar to the content model for the <section> element found in other topic types such as concept.

<remedy> This optional last-child of <troubleSolution> describes a possible remedy for the <condition>. The <remedy> element begins with an optional <responsibleParty> element followed by either a <steps> element or a <steps-unordered> element. The content models for <steps> and <steps-unordered> are borrowed from <task>. This allows <remedy> to reuse steps from tasks through conref.

<responsibleParty> This optional first child of <remedy> indicates who is expected to perform the steps in the <remedy>. Involved troubleshooting scenarios can be prescriptive about who is supposed to do what. For instance, some remedies may be performed by low-level customer support agents, while other remedies are restricted to enginering staff.

Constrained section model used for <condition> and <cause>

This model begins with an optional <title> element. Unlike the <section> element, this is the only place in the model where <title> is available.

The remainder of the model consists of the same mix of elements available in section, except that text and phrase-based DITA inline elements, such as <b> and <uicontrol>, are not allowed at the top level. Instead, text is entered under a <p> element.

Simple troubleshooting example

Simple troubleshooting information consists of a condition with a single solution.

Example

```
<troubleshooting id="ApplicationNotResponding" xml:lang="en-us">
  <title>Applciation is not responding</title>
  <troublebody>
    <condition>
      <p>The application does not respond to input from the
keyboard or the mouse.</p>
    </condition>
    <troubleSolution>
      <cause>
        <title>Cause</title>
        <p>The application has entered into a state from which
```

```
it cannot recover.</p>
      </cause>
      <remedy><title>Remedy</title>
      <steps>
        <step>
          <cmd>Right-click on the <uicontrol>Windows task bar</
uicontrol> and select
          <uicontrol>Task Manager</uicontrol></cmd>
        </step>
        <step>
          <cmd>Right-click on the application and select
<uicontrol>End Task</uicontrol></cmd>
        </step>
      </steps></remedy>
    </troubleSolution>
  </troublebody>
</troubleshooting>
```

Complex troubleshooting example

Complex troubleshooting information consists of a condition with multiple potential solutions.

Example

```
<troubleshooting id="NoPower">
  <title>The system will not turn on</title>
  <troublebody>
    <condition>
      <p>The system has no power.</p>
    </condition>
    <troubleSolution>
      <cause>
        <title>Uplugged</title>
        <p>The system is uplugged.</p>
      </cause>
      <remedy>
        <steps-informal>
          <p>Plug the power cord into the wall outlet.</p>
        </steps-informal>
      </remedy>
    </troubleSolution>
    <troubleSolution>
      <cause>
        <title>Tripped circuit breaker</title>
        <p>The electrical circuit that services the wall outlet
has been tripped.</p>
      </cause>
      <remedy>
        <title>Reset the circuit breaker</title>
        <steps>
          <step>
            <cmd>Locate the circuit breaker box.</cmd>
          </step>
          <step>
            <cmd>Identify the circuit servicing the wall
outlet.</cmd>
          </step>
          <step>
```

```
            <cmd>Reset the breaker for that circuit.</cmd>
          </step>
        </steps>
      </remedy>
      <remedy/>
    </troubleSolution>
    <troubleSolution>
      <cause>
        <title>Contact support</title>
        <p>The system still does not turn on.</p>
      </cause>
      <remedy>
        <title>Contacting support</title>
        <steps
conref="ContactingSuppport.dita#ContactingSuppport/steps">
          <step>
            <cmd/>
          </step>
        </steps>
      </remedy>
    </troubleSolution>
  </troublebody>
</troubleshooting>
```

Alarm example

Alarm resolution information follows the same pattern, and uses the same semantics as troubleshooting information.

Example

```
<troubleshooting id="AlarmClearing" xml:lang="en-us">
  <title>FCOM-232 Communications card failure</title>
  <troublebody>
    <troubleSolution>
      <cause>
        <title>Circuit pack CM-292 has become unseated</title>
      </cause>
      <remedy>
        <title>Reseat the cicuit pack</title>
        <steps>
          <step id="RemovePanel">
            <cmd>Remove the system back panel</cmd>
          </step>
          <step id="RemoveCircuitPack">
            <cmd>Remove the CM-292 from the rack</cmd>
          </step>
          <step>
            <cmd>Reinsert the CM-292 into the rack</cmd>
          </step>
        </steps>
      </remedy>
    </troubleSolution>
    <troubleSolution>
      <cause>
        <title>Circuit pack CM-292 is defective</title>
      </cause>
      <remedy>
        <steps>
```

```
          <step conref="#AlarmClearing/RemovePanel">
            <cmd/>
          </step>
          <step conref="#AlarmClearing/RemoveCircuitPack">
            <cmd/>
          </step>
          <step>
            <cmd>Insert a new CM-292 circuit pack into the
rack</cmd>
          </step>
        </steps>
      </remedy>
    </troubleSolution>
  </troublebody>
</troubleshooting>
```

Embedded troubleshooting

Embedded troubleshooting is brief, corrective action information that appears within other information types. A “trouble” note type is available, plus <steptroubleshooting> and <task-troubleshooting> elements are available within tasks.

Important:

Limit the embedded troubleshooting content to no more than a sentence or two. When extensive troubleshooting information is needed, use embedded troubleshooting to refer the reader to a dedicated troubleshooting topic. A good inicator that a dedicated troubleshooting topic should be used instead is when a numbered list appears within embedded troubleshooting information.

Trouble note type

When incidental troubleshooting information appears within non-troubleshooting information types, one can present this information using a <note> element with its type attribute set to “trouble.”

Example

```
<note type="trouble">
  <p>If you cannot find certan menu choices, go into
<uicontrol>User Preferences</uicontrol> and
    make sure that <uicontrol>Show Full Menus</uicontrol> has
been selected.</p>
</note>
```

Task troubleshooting

Task topics may include a <tasktroubleshooting> element. While a <result> element need not be present for <tasktroubleshooting> to be used, it generally is. The combination of <result; followed by <tasktroubleshooting> is particularly useful for handling situations in which the user obtains unexpected results for a task.

Example

```
<result>
  <p>The system backup should be present on the backup
drive.</p>
</result>
<tasktroubleshooting>
  <p>If the system backup is not present on the backup drive
refer to <xref

href="SystemBackupFailure.dita#SystemBackupFailure">System
backup failed.</xref></p>
</tasktroubleshooting>
```

Step troubleshooting

Steps may include a <steptroubleshooting> element. While a <stepresult> element need not be present for <steptroubleshooting> to be used, it generally is. The combination of <stepresult; followed by <steptroubleshooting> is particularly useful for handling situations in which the user obtains unexpected results for a step.

Example

```
<steptroubleshooting>
  <p>If login fails make sure that the CAPS LOCK key is off.</p>
</steptroubleshooting>
```