An OASIS DITA Adoption Technical Committee Publication

DITA 1.3 Feature Article:
Using DITA 1.3 Troubleshooting

Author: Bob Thomas
On behalf of the DITA Adoption Technical Committee

Date: 30 May 2014

OASIS (Organization for the Advancement of Structured Information Standards) is a not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards. Members themselves set the OASIS technical agenda, using a lightweight, open process expressly designed to promote industry consensus and unite disparate efforts. The consortium produces open standards for Web services, security, e-business, and standardization efforts in the public sector and for application-specific markets. OASIS was founded in 1993. More information can be found on the OASIS website at http://www.oasis-open.org.

The OASIS DITA Adoption Technical Committee members collaborate to provide expertise and resources to educate the marketplace on the value of the DITA OASIS standard. By raising awareness of the benefits offered by DITA, the DITA Adoption Technical Committee expects the demand for, and availability of, DITA conforming products and services to increase, resulting in a greater choice of tools and platforms and an expanded DITA community of users, suppliers, and consultants.

DISCLAIMER: All examples presented in this article were produced using one or more tools chosen at the author's discretion and in no way reflect endorsement of the tools by the OASIS DITA Adoption Technical Committee.

This white paper was produced and approved by the OASIS DITA Adoption Technical Committee as a Committee Draft. It has not been reviewed and/or approved by the OASIS membership at-large.

Document History

Revision	Date	Author	Summary
First Draft	2 February 2014	Thomas	Initial draft
Second Draft	30 May 2014	Thomas	Reworked to be a tutorial

Table of Contents

Using DITA 1.3 Troubleshooting

> *This article briefly describes the troubleshooting information type plus the DITA 1.3 features that support it. Several examples follow that show how to use the new troubleshooting features.*

Troubleshooting information

> *This topic describes the troubleshooting information type.*

What is troubleshooting?

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 serve as successive fixes users can apply to eliminate an undesirable condition.

Embedded troubeshooting

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. Refer the user to 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 that may warrant a response.

DITA 1.3 troubleshooting features

This topic describes DITA 1.3 support for the troubleshooting information type.

What are the troubleshooting features?

DITA 1.3 introduced the following troubleshooting features:

<troubleshooting> topic
A new “troubleshooting” topic type that models troubleshooting information-type semantics.

<tasktroubleshooting>
In <task> topics, this section-like element is a place for specifying the corrective action to take when a task fails.

<steptroubleshooting>
In <task> topics, this step sub-element is a place for specifying the corrective action to take when a step fails.

troubleshooting note-type
In the <note> element, the “trouble” note-type is available alongside other <note> @type values such as “caution”, “note”, or “tip”. The “trouble” note-type contains an incidental corrective action that pertains to its surrounding content.

Why DITA 1.3 troubleshooting matters

With the new DITA troubleshooting markup, writers can create tightly focused information that helps readers resolve specific problems. Writers can now easily follow consistent patterns for troubleshooting information. This consistency expresses itself in predictable organization, titles, and icon graphics that lets readers quickly locate troubleshooting information.

How do you use it?

Learn more about how to use these features by reading through the examples that follow. Addtionally, an annotated template for the troubleshooting topic appears at the end of this article.

Troubleshooting-topic examples

Simple troubleshooting topic

This example is a basic troubleshooting topic.

Scenario

Name
Tripped circuit breaker

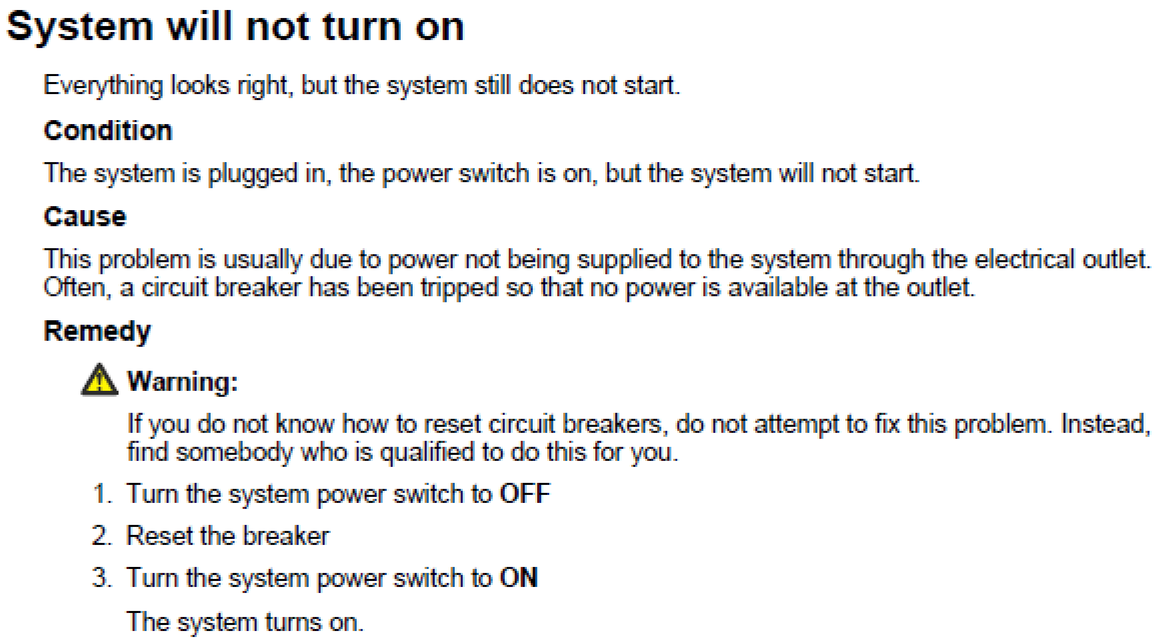
Description
The system is plugged in, the power switch is on, but the system will not start. This problem is external to the system, and it is almost always due to a tripped circuit breaker. The system is a low-power consumer product that runs on household electricity.

Discussion

We will use a troubleshooting topic with a single <troubleSolution> element to document the corrective action for this problem.

Output

Before looking at the tag view, look at the output first:



System will not turn on

Everything looks right, but the system still does not start.

Condition

The system is plugged in, the power switch is on, but the system will not start.

Cause

This problem is usually due to power not being supplied to the system through the electrical outlet. Often, a circuit breaker has been tripped so that no power is available at the the outlet.

Remedy

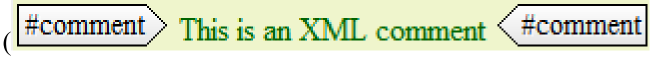
⚠ Warning:

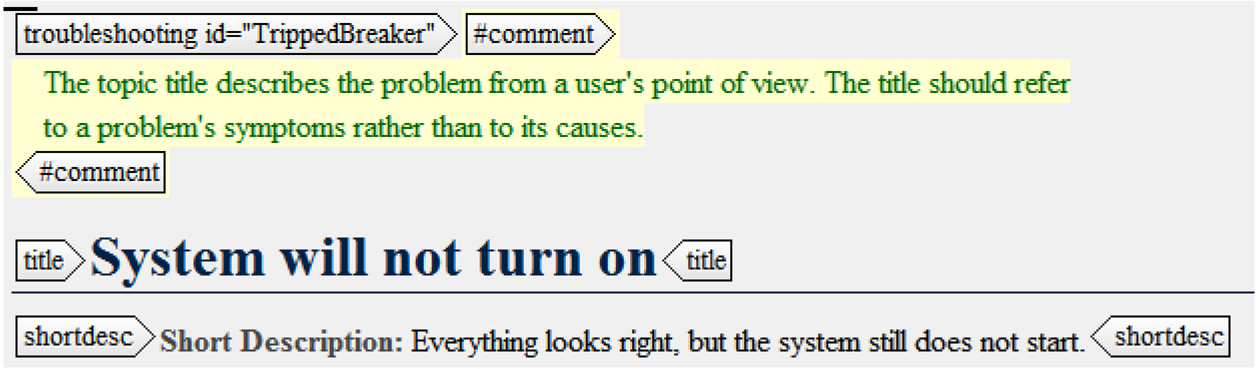
If you do not know how to reset circuit breakers, do not attempt to fix this problem. Instead, find somebody who is qualified to do this for you.

1. Turn the system power switch to **OFF**
2. Reset the breaker
3. Turn the system power switch to **ON**

   The system turns on.

Tag view

Collectively, the following series of XML editor screen-shots form a complete troubleshooting topic. They have been split into several segments for formatting reasons. Pay close attention to the XML comments (#comment This is an XML comment #comment)embedded within the topic markup. These comments contain usage advice.



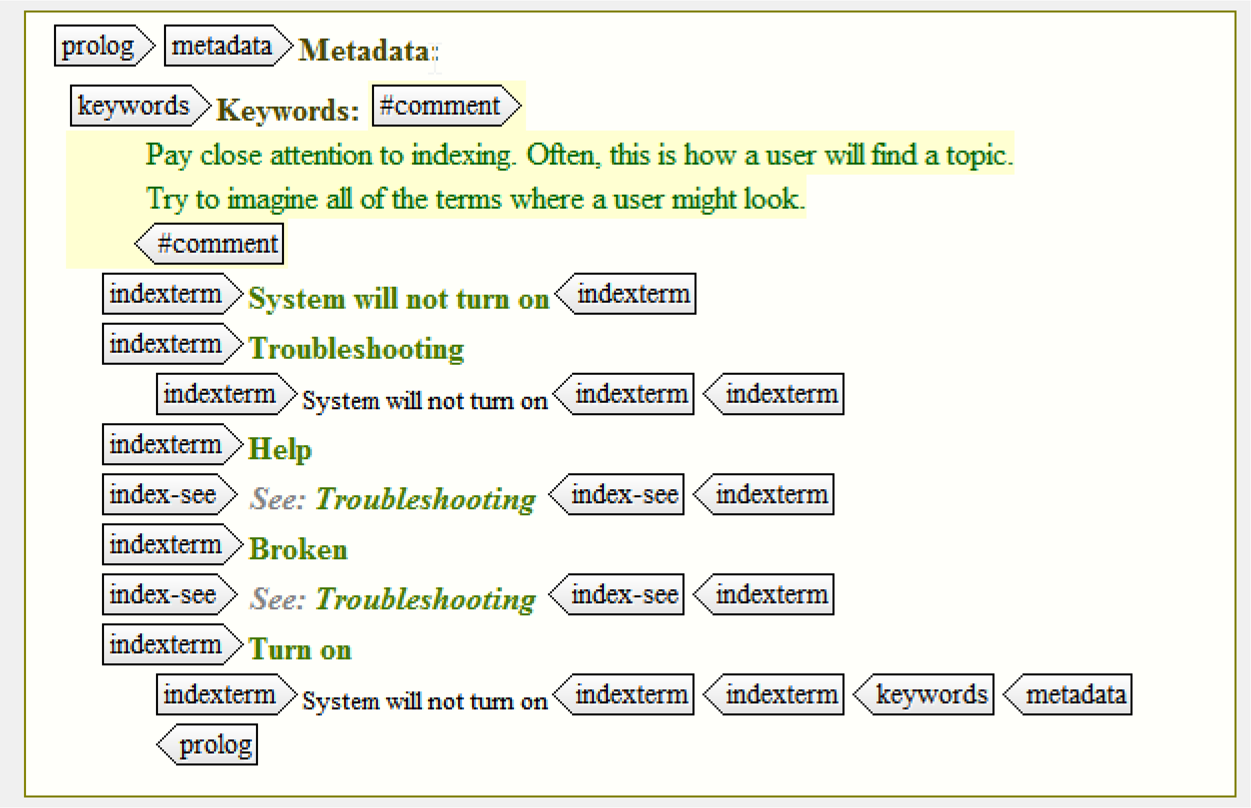
troubleshooting id="TrippedBreaker" #comment

The topic title describes the problem from a user's point of view. The title should refer to a problem's symptoms rather than to its causes.

#comment

title **System will not turn on** title

shortdesc **Short Description:** Everything looks right, but the system still does not start. shortdesc

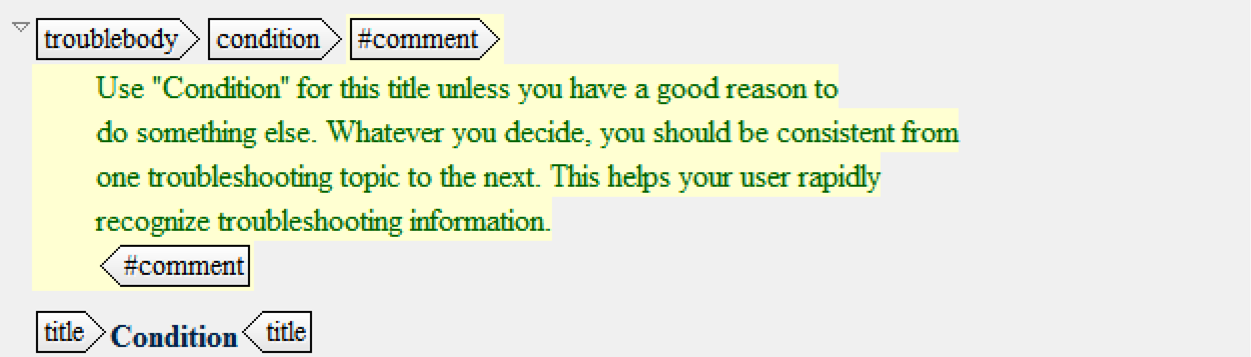


prolog metadata **Metadata:**

keywords **Keywords:** #comment

Pay close attention to indexing. Often, this is how a user will find a topic.
Try to imagine all of the terms where a user might look.
#comment

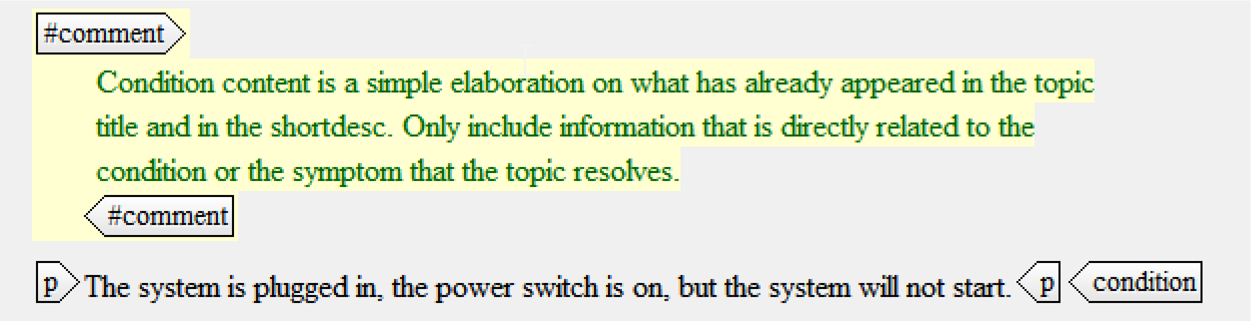
indexterm **System will not turn on** indexterm
indexterm **Troubleshooting**
indexterm System will not turn on indexterm indexterm
indexterm **Help**
index-see *See: **Troubleshooting*** index-see indexterm
indexterm **Broken**
index-see *See: **Troubleshooting*** index-see indexterm
indexterm **Turn on**
indexterm System will not turn on indexterm indexterm keywords metadata
prolog



troublebody condition #comment

Use "Condition" for this title unless you have a good reason to
do something else. Whatever you decide, you should be consistent from
one troubleshooting topic to the next. This helps your user rapidly
recognize troubleshooting information.
#comment

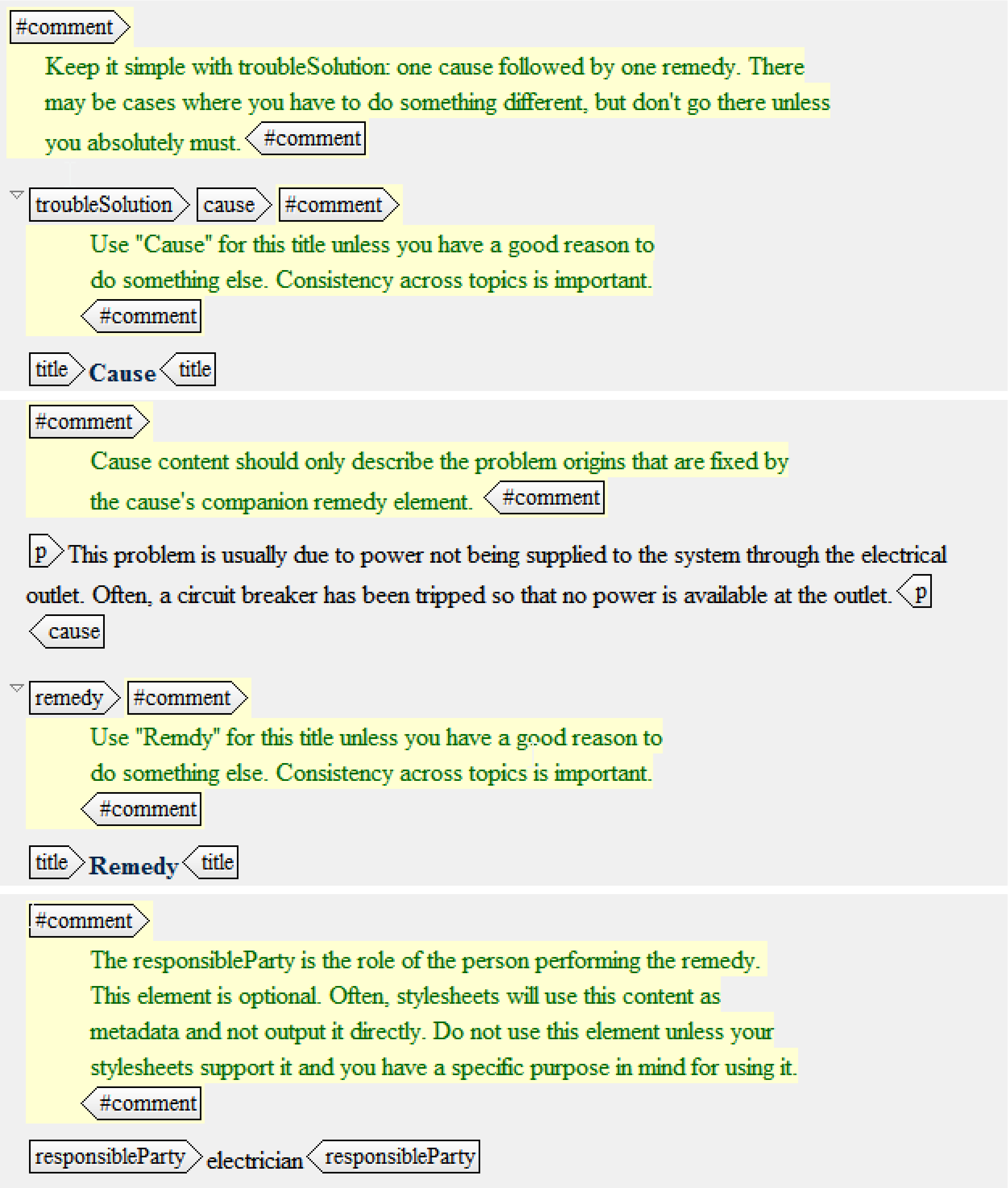
title **Condition** title



#comment

Condition content is a simple elaboration on what has already appeared in the topic
title and in the shortdesc. Only include information that is directly related to the
condition or the symptom that the topic resolves.
#comment

p The system is plugged in, the power switch is on, but the system will not start. p condition



#comment
Keep it simple with troubleSolution: one cause followed by one remedy. There may be cases where you have to do something different, but don't go there unless you absolutely must. #comment

troubleSolution cause #comment
Use "Cause" for this title unless you have a good reason to do something else. Consistency across topics is important.
#comment

title **Cause** title

#comment
Cause content should only describe the problem origins that are fixed by the cause's companion remedy element. #comment

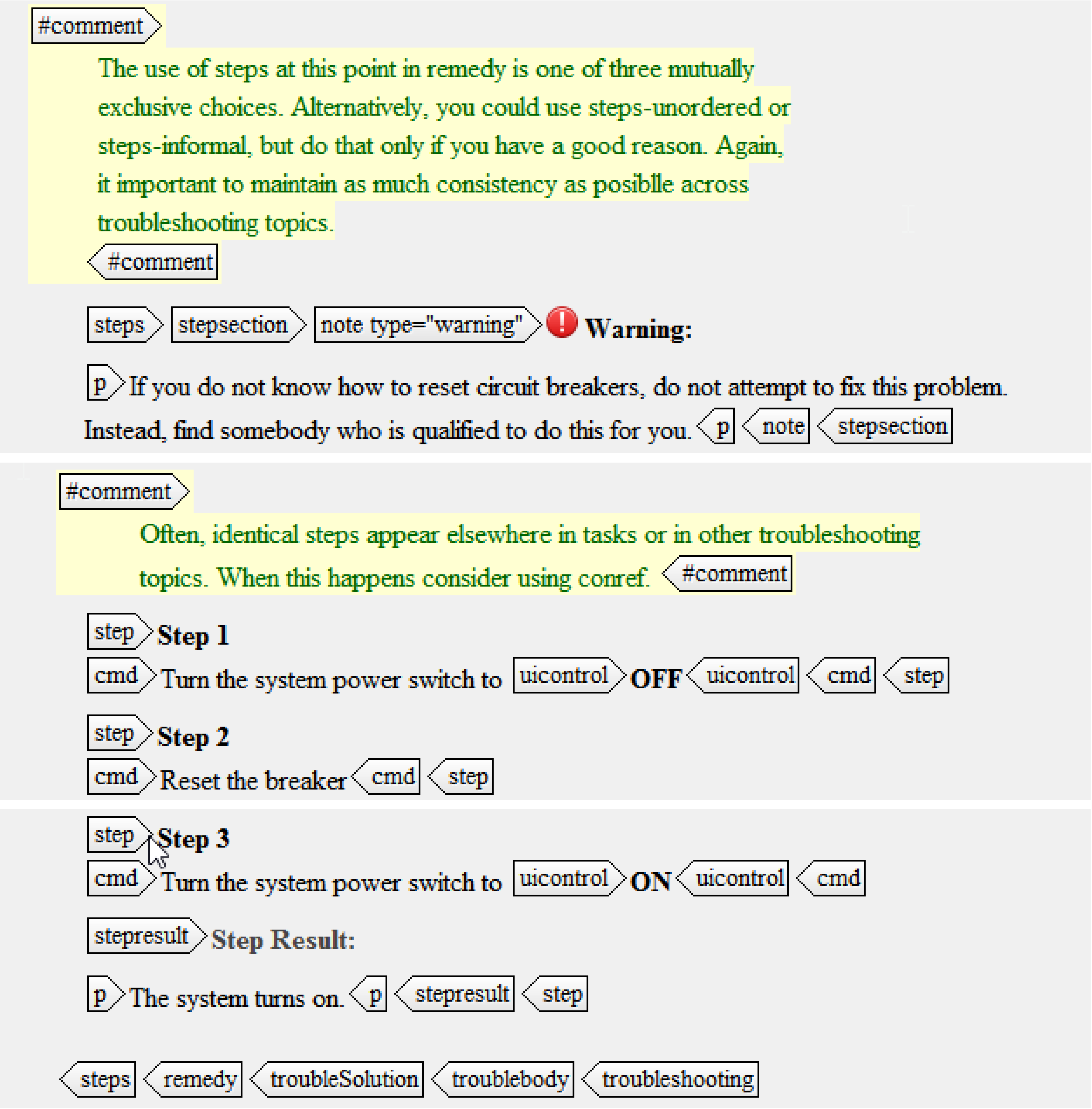
p This problem is usually due to power not being supplied to the system through the electrical outlet. Often, a circuit breaker has been tripped so that no power is available at the outlet. p
cause

remedy #comment
Use "Remdy" for this title unless you have a good reason to do something else. Consistency across topics is important.
#comment

title **Remedy** title

#comment
The responsibleParty is the role of the person performing the remedy. This element is optional. Often, stylesheets will use this content as metadata and not output it directly. Do not use this element unless your stylesheets support it and you have a specific purpose in mind for using it.
#comment

responsibleParty electrician responsibleParty



#comment

The use of steps at this point in remedy is one of three mutually exclusive choices. Alternatively, you could use steps-unordered or steps-informal, but do that only if you have a good reason. Again, it important to maintain as much consistency as posiblle across troubleshooting topics.

#comment

steps stepsection note type="warning" ❗ **Warning:**

p If you do not know how to reset circuit breakers, do not attempt to fix this problem. Instead, find somebody who is qualified to do this for you. p note stepsection

#comment

Often, identical steps appear elsewhere in tasks or in other troubleshooting topics. When this happens consider using conref. #comment

step **Step 1**

cmd Turn the system power switch to uicontrol **OFF** uicontrol cmd step

step **Step 2**

cmd Reset the breaker cmd step

step **Step 3**

cmd Turn the system power switch to uicontrol **ON** uicontrol cmd

stepresult **Step Result:**

p The system turns on. p stepresult step

steps remedy troubleSolution troublebody troubleshooting

Multiple solutions troubleshooting topic

> *This example is a troubleshooting topic with multiple remedies.*

Scenario

Name Cannot log in to the system.

Description A customer cannot log in to the system. The reasons for this could be: no account exists, the user forgot their user id, or the user forgot their password. The user needs to contact support if none of the remedies resolve this problem.

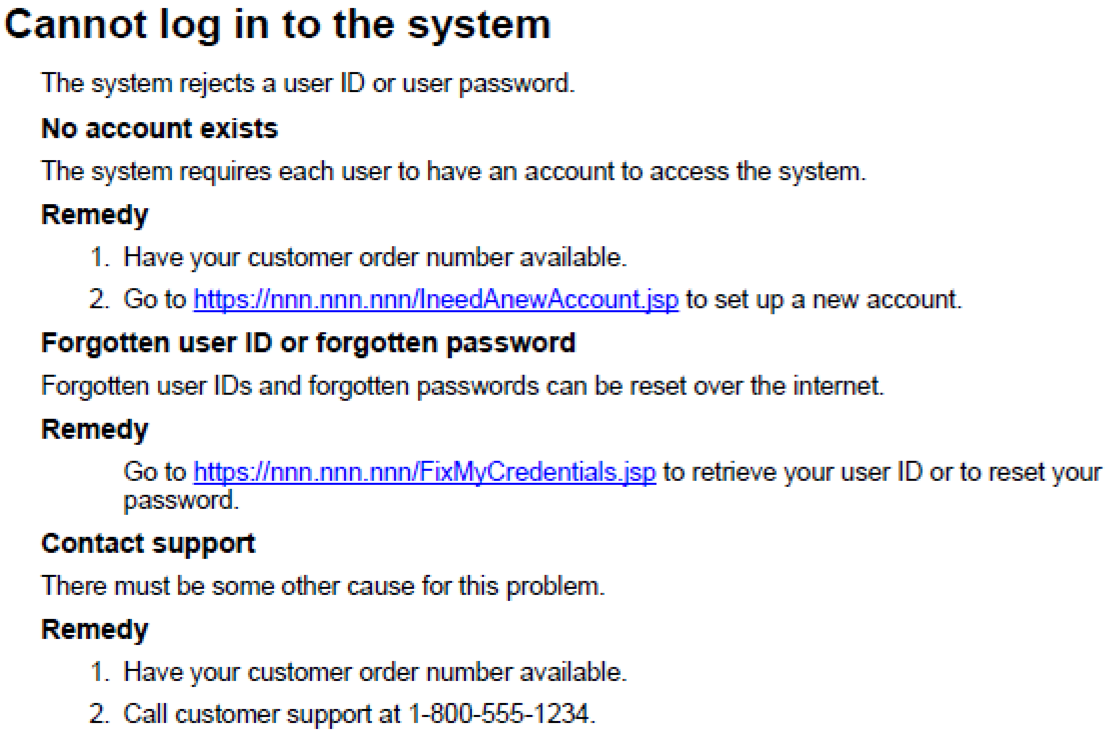
Discussion

We will use a troubleshooting topic with three <troubleSolution> elements to present a series of potential fixes (fallbacks) for this problem. The idea is that the user will try each <troubleSolution> until the problem is resolved. The first title in each <troubleSolution> is in the <cause> element, and it briefly describes the cause instead of using a consistent title such as “Cause.” In multiple <troubleSolution> scenarios, descriptive labeling helps users diagnose problems quicker. Therefore, a title that connotes the cause is more important than using a uniform title such as “Cause.”

No <condition> element is used here. That is because all pertinent information about the condition has already been given in the topic title and in the shortdesc.

The first <troubleSolution> contains the “no account exists” cause/remedy. This <troubleSolution> must be first because there is no point in a user trying to reset account credentials for a non-existing account. The second <troubleSolution> contains the "forgotten user id/forgotten password" cause/remedy. The third <troubleSolution> contains the final fallback: calling customer support. In practical applications, this third <troubleSolution> would be reused through conref.

Output



Cannot log in to the system

The system rejects a user ID or user password.

No account exists

The system requires each user to have an account to access the system.

Remedy

1. Have your customer order number available.
2. Go to https://nnn.nnn.nnn/IneedAnewAccount.jsp to set up a new account.

Forgotten user ID or forgotten password

Forgotten user IDs and forgotten passwords can be reset over the internet.

Remedy

Go to https://nnn.nnn.nnn/FixMyCredentials.jsp to retrieve your user ID or to reset your password.

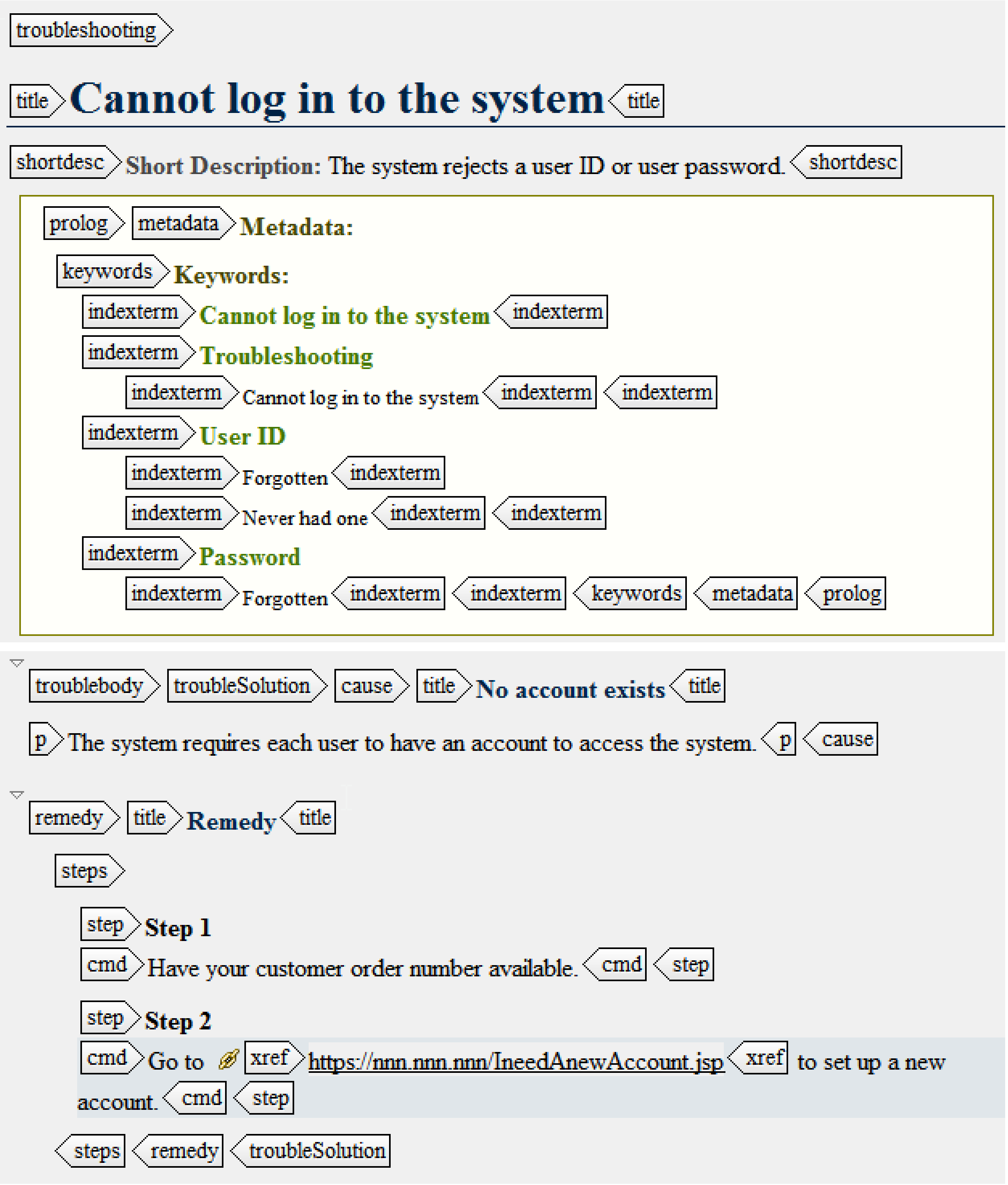
Contact support

There must be some other cause for this problem.

Remedy

1. Have your customer order number available.
2. Call customer support at 1-800-555-1234.

Tag view



troubleshooting

title Cannot log in to the system title

shortdesc **Short Description:** The system rejects a user ID or user password. shortdesc

prolog metadata **Metadata:**

keywords **Keywords:**

indexterm **Cannot log in to the system** indexterm

indexterm **Troubleshooting**

indexterm Cannot log in to the system indexterm indexterm

indexterm **User ID**

indexterm Forgotten indexterm

indexterm Never had one indexterm indexterm

indexterm **Password**

indexterm Forgotten indexterm indexterm keywords metadata prolog

troublebody troubleSolution cause title **No account exists** title

p The system requires each user to have an account to access the system. p cause

remedy title **Remedy** title

steps

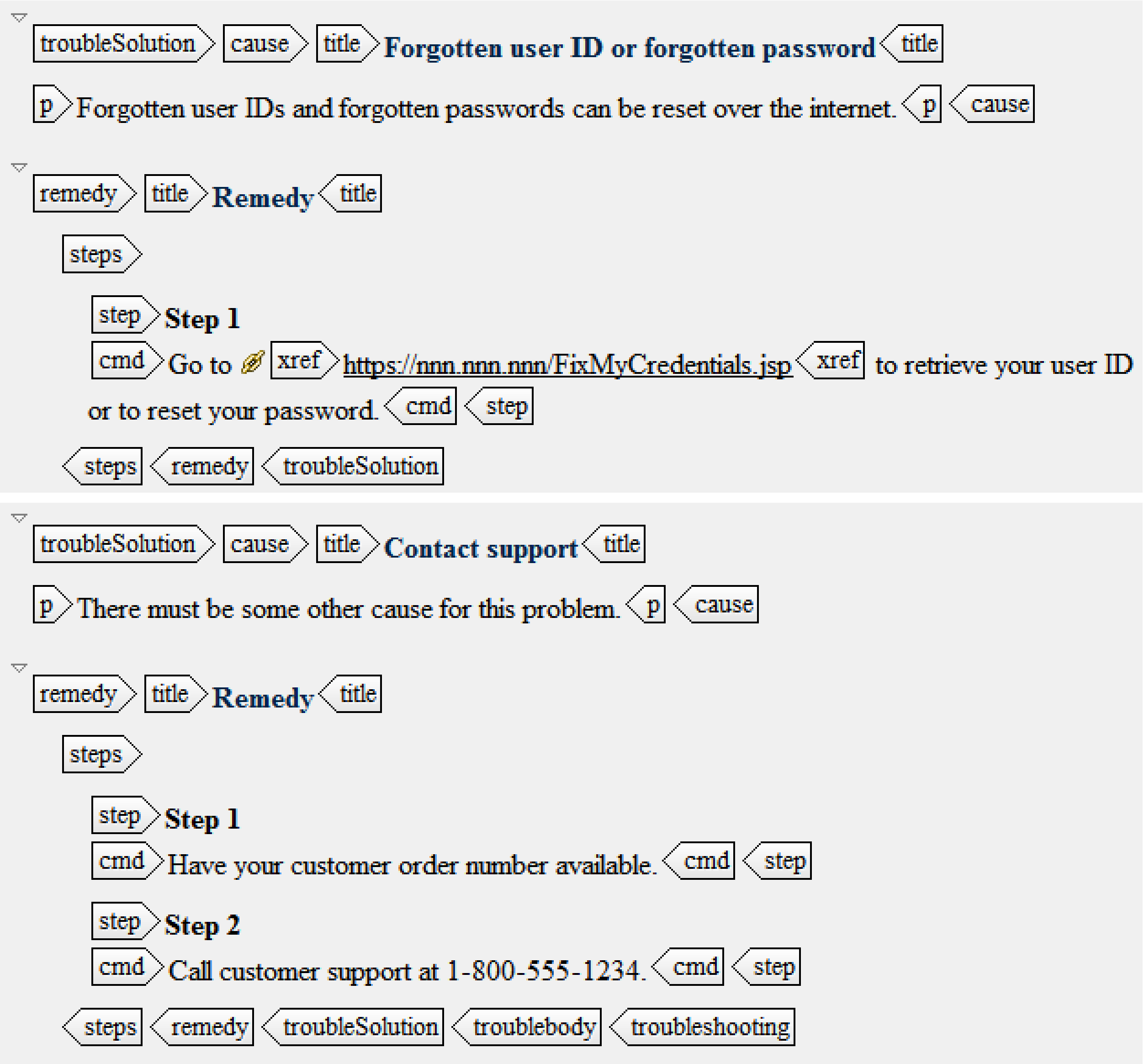
step **Step 1**

cmd Have your customer order number available. cmd step

step **Step 2**

cmd Go to xref https://nnn.nnn.nnn/IneedAnewAccount.jsp xref to set up a new account. cmd step

steps remedy troubleSolution



```
troubleSolution cause title Forgotten user ID or forgotten password title
p Forgotten user IDs and forgotten passwords can be reset over the internet. p cause

remedy title Remedy title
  steps
    step Step 1
    cmd Go to xref https://nnn.nnn.nnn/FixMyCredentials.jsp xref to retrieve your user ID
    or to reset your password. cmd step
  steps remedy troubleSolution

troubleSolution cause title Contact support title
p There must be some other cause for this problem. p cause

remedy title Remedy title
  steps
    step Step 1
    cmd Have your customer order number available. cmd step
    step Step 2
    cmd Call customer support at 1-800-555-1234. cmd step
  steps remedy troubleSolution troublebody troubleshooting
```

Managing complexity

> *Some troubleshooting information is inherently complex.*

Complexity

Software products are often complex. Fortunately, when things go wrong within a product, good software engineering shields users from most of this complexity. But, in some cases, there simply is not enough development budget to cover every eventuality with code. Unfortunately, if the code does not supply the intellegence for recovering from a problem, the responsbility falls to the product documentation. Failing that, the responsibilty cedes to technical support. And finally, failing that, the responsibility lands upon the user or upon a user community. For our purposes, we assume that the product documentation has accepted responsibility.

Resolving complex problems

Regardless of complexity, the basics of the troubleshooting information type are still valid: condition, cause, remedy. However, complexity manifests itself almost immediately. Typically, the condition the product reports is too general, making it unlikely that the user will be able to immediately identify the actual cause. This becomes problematic in troubleshooting scenarios where the user cannot simply try one remedy after the next. For some products, that method is too disruptive, and it may be unsafe. The cure for this is to use diagnostic steps to further explore the condition until it points to a single cause. Once the cause has been isolated, its corresponding remedy can be followed.

The troubleshooting topic and complexity

The best practice for complex scenarios is to use the first <troubleSolution> element in a troubleshooting topic to help users determine a specific cause. Subsequent <troubleSolution> elements contain a description and a remedy for each cause identified in the first <troubleSolution>.

Complex scenario

> *This example is a complex troubleshooting topic.*

Scenario

Name Rare alarm

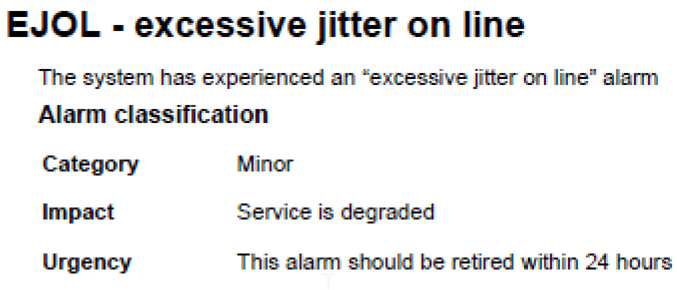
Description A fiber-optic switching system can issue any of 49 separate digital transmission alarms. 42 of these alarms are somewhat likely to happen during normal operation. Consequently, the software architecture contains logic to evaluate several system state variables and so that it can report a specific cause to the user along with the alarm code. But, the likelihood of the remaining 10 alrams occurring is small enough that the software architecture does little more than issue an alarm. Consequently, the system alarm manual must incorporate diagnostic steps into the troubleshooting topics for each those 10 alarms.

This scenario is about documenting the EJOL (Excessive Jitter On Line) alarm—one of the 10 rare alarms. The EJOL alarm can occur due to: trouble on the far-end system, excessive line noise, or a faulty LN243 circuit pack.

Discussion

This solution uses two conventions. First, it uses a <dl> element inside of the <condition> element to present alarm classification information. Second, this solution devotes the first <troubleSolution> element soley to identifying the alarm's cause. Details have been omitted from the remaining <troubleSolution> elements because they are not important for understanding how to organize complex troubleshooting information.

Output

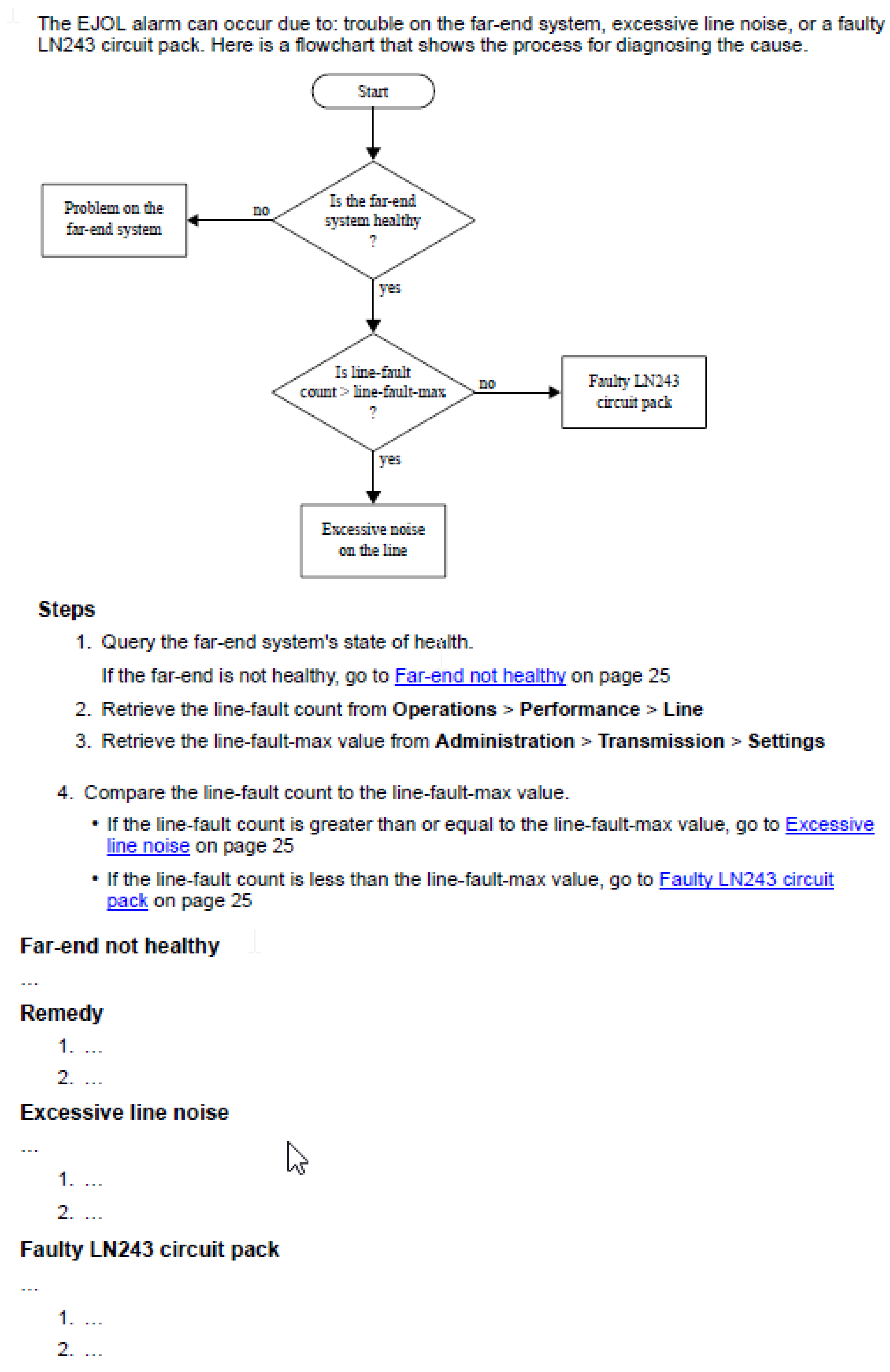


EJOL - excessive jitter on line

The system has experienced an "excessive jitter on line" alarm

Alarm classification

Category	Minor
Impact	Service is degraded
Urgency	This alarm should be retired within 24 hours



The EJOL alarm can occur due to: trouble on the far-end system, excessive line noise, or a faulty LN243 circuit pack. Here is a flowchart that shows the process for diagnosing the cause.

Start
Is the far-end system healthy ?
no
Problem on the far-end system
yes
Is line-fault count > line-fault-max ?
no
Faulty LN243 circuit pack
yes
Excessive noise on the line

Steps

1. Query the far-end system's state of health.

   If the far-end is not healthy, go to Far-end not healthy on page 25
2. Retrieve the line-fault count from **Operations** > **Performance** > **Line**
3. Retrieve the line-fault-max value from **Administration** > **Transmission** > **Settings**

4. Compare the line-fault count to the line-fault-max value.
   - If the line-fault count is greater than or equal to the line-fault-max value, go to Excessive line noise on page 25
   - If the line-fault count is less than the line-fault-max value, go to Faulty LN243 circuit pack on page 25

Far-end not healthy

…

Remedy

1. …
2. …

Excessive line noise

…

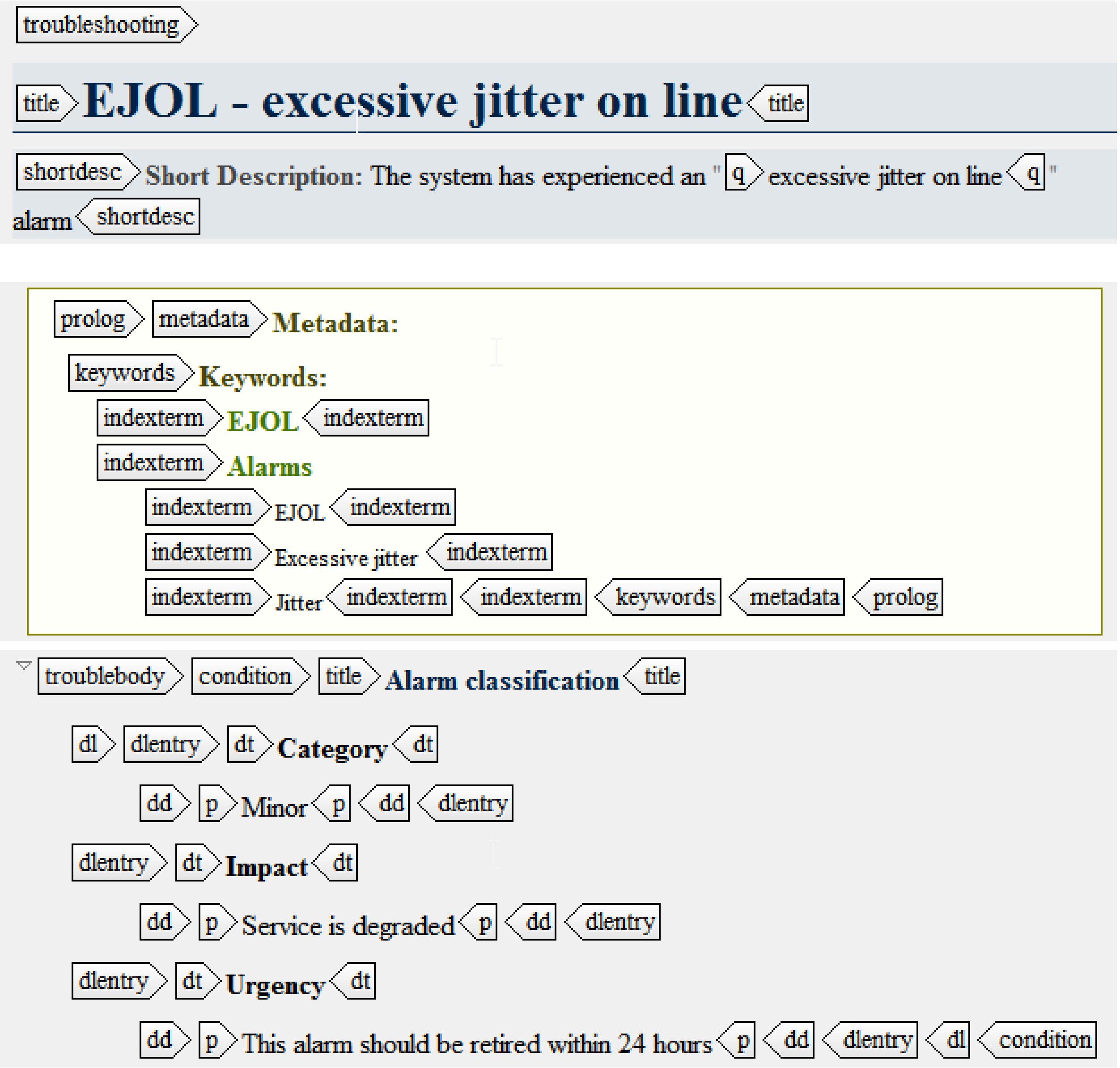
1. …
2. …

Faulty LN243 circuit pack

…

1. …
2. …

Tag view



troubleshooting

title EJOL - excessive jitter on line title

shortdesc Short Description: The system has experienced an " q excessive jitter on line q " alarm shortdesc

prolog metadata Metadata:

keywords Keywords:

indexterm EJOL indexterm

indexterm Alarms

indexterm EJOL indexterm

indexterm Excessive jitter indexterm

indexterm Jitter indexterm indexterm keywords metadata prolog

troublebody condition title Alarm classification title

dl dlentry dt Category dt

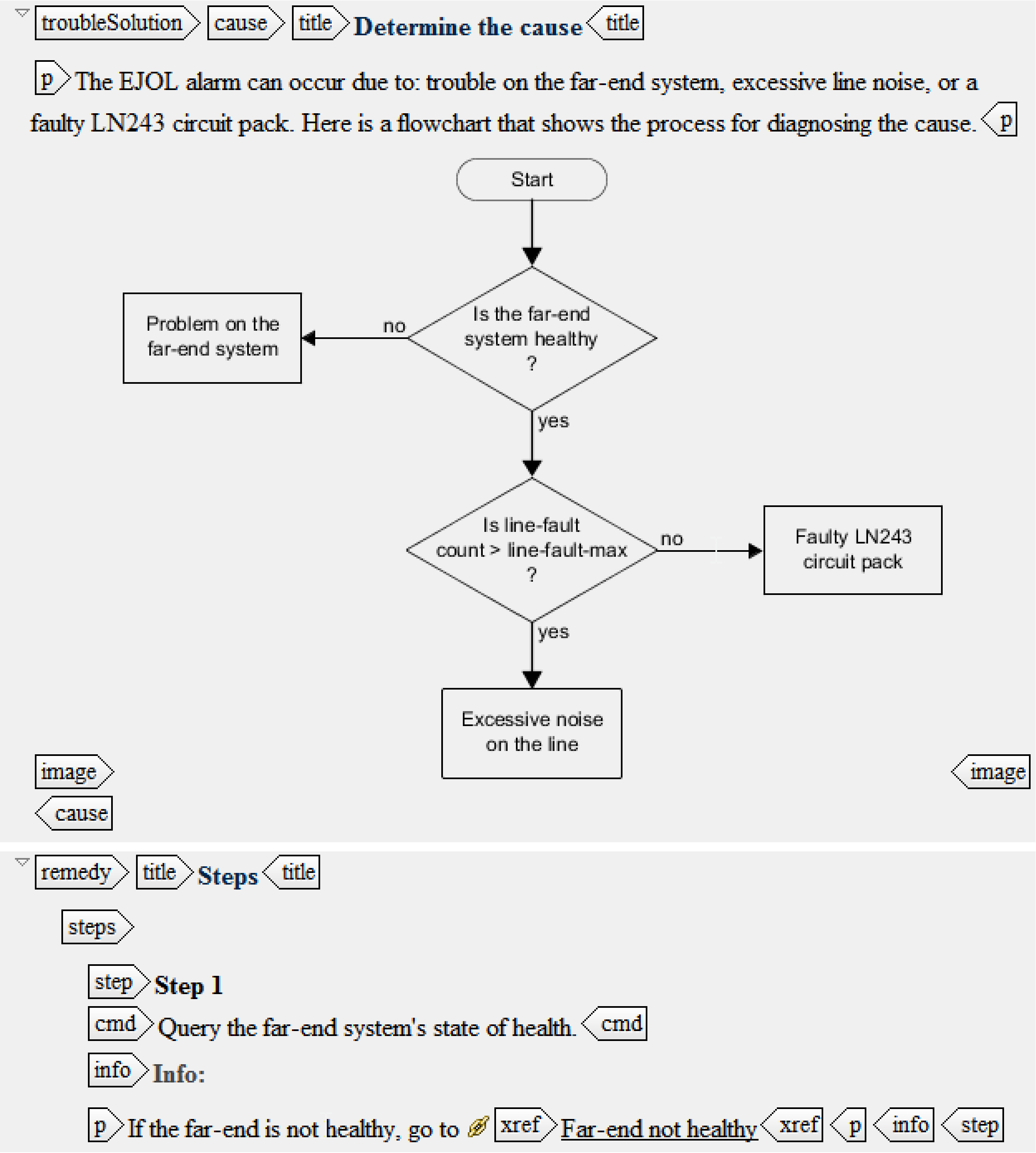
dd p Minor p dd dlentry

dlentry dt Impact dt

dd p Service is degraded p dd dlentry

dlentry dt Urgency dt

dd p This alarm should be retired within 24 hours p dd dlentry dl condition



troubleSolution cause title **Determine the cause** title

p The EJOL alarm can occur due to: trouble on the far-end system, excessive line noise, or a faulty LN243 circuit pack. Here is a flowchart that shows the process for diagnosing the cause. p

Start
Is the far-end system healthy ?
no
Problem on the far-end system
yes
Is line-fault count > line-fault-max ?
no
Faulty LN243 circuit pack
yes
Excessive noise on the line

image image

cause

remedy title **Steps** title

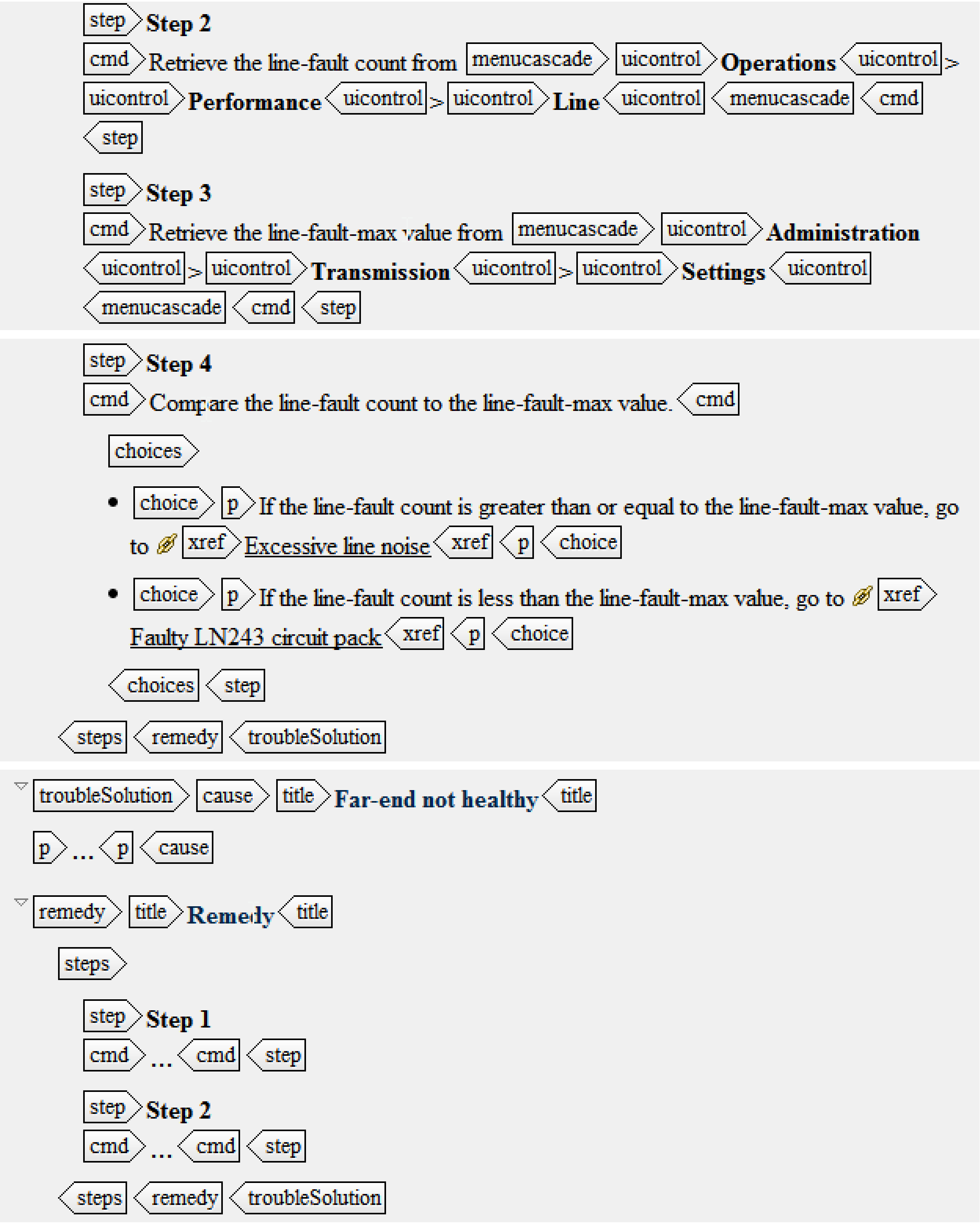
steps

step **Step 1**

cmd Query the far-end system's state of health. cmd

info **Info:**

p If the far-end is not healthy, go to xref Far-end not healthy xref p info step



step> **Step 2**
cmd> Retrieve the line-fault count from menucascade> uicontrol> **Operations** <uicontrol >
uicontrol> **Performance** <uicontrol > uicontrol> **Line** <uicontrol <menucascade <cmd
<step

step> **Step 3**
cmd> Retrieve the line-fault-max value from menucascade> uicontrol> **Administration**
<uicontrol > uicontrol> **Transmission** <uicontrol > uicontrol> **Settings** <uicontrol
<menucascade <cmd <step

step> **Step 4**
cmd> Compare the line-fault count to the line-fault-max value. <cmd

choices>

- choice> p> If the line-fault count is greater than or equal to the line-fault-max value, go to xref> Excessive line noise <xref <p <choice
- choice> p> If the line-fault count is less than the line-fault-max value, go to xref> Faulty LN243 circuit pack <xref <p <choice

<choices <step

<steps <remedy <troubleSolution

troubleSolution> cause> title> **Far-end not healthy** <title

p> … <p <cause

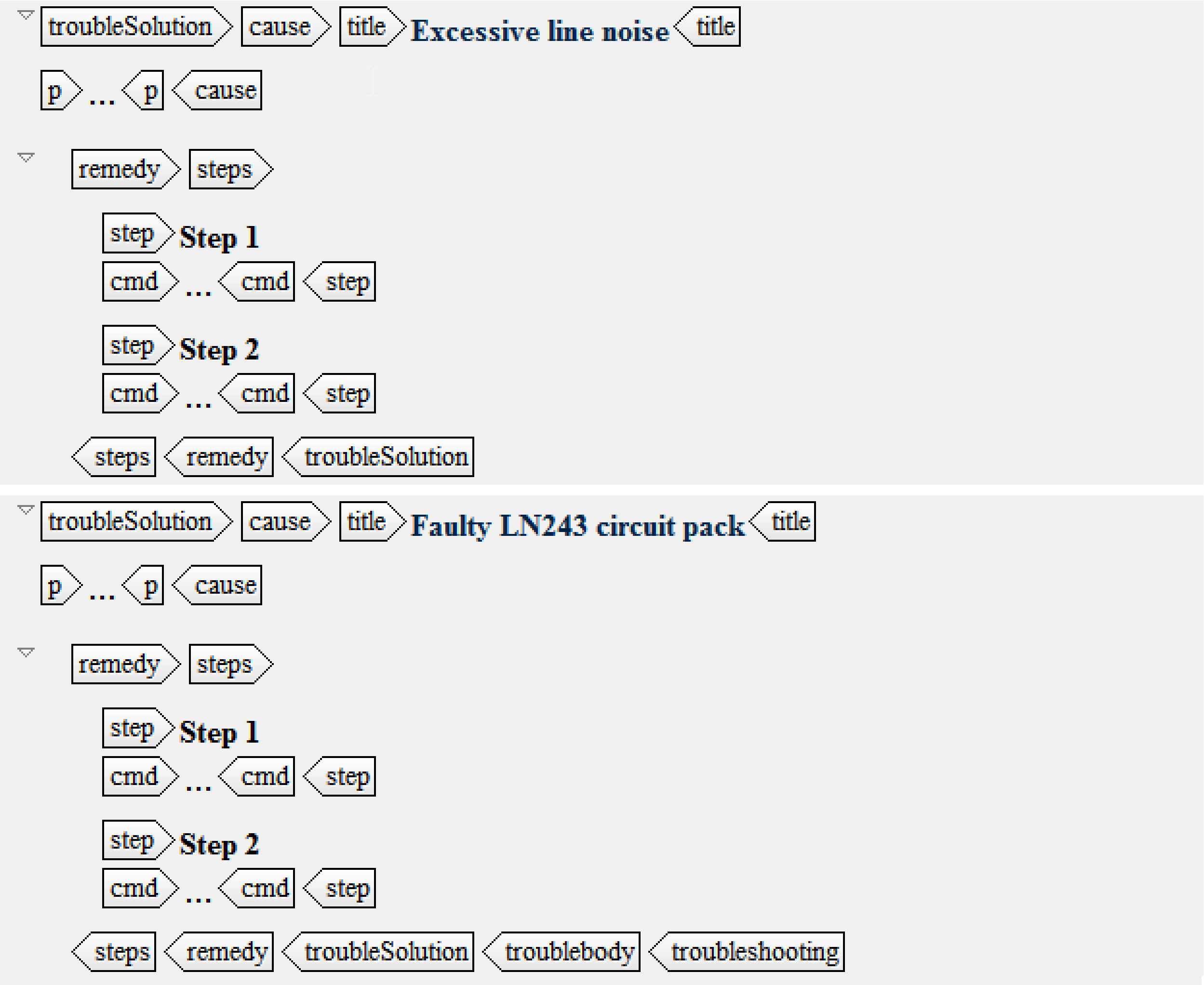
remedy> title> **Remedy** <title

steps>

step> **Step 1**
cmd> … <cmd <step

step> **Step 2**
cmd> … <cmd <step

<steps <remedy <troubleSolution



```
troubleSolution cause title Excessive line noise title
p ... p cause
    remedy steps
        step Step 1
        cmd ... cmd step
        step Step 2
        cmd ... cmd step
    steps remedy troubleSolution
troubleSolution cause title Faulty LN243 circuit pack title
p ... p cause
    remedy steps
        step Step 1
        cmd ... cmd step
        step Step 2
        cmd ... cmd step
    steps remedy troubleSolution troublebody troubleshooting
```

Embeded troubleshooting examples

Task-related

> *This section contains examples showing streptroubleshooting and tasktroubleshooting*

Discussion

The steptroubleshooting and tasktroubleshooting elements were designed to give users immediate, brief information about how to resolve problems when an undesirable result happens. This means writers should limit content to discussing condition, cause, and then perhaps a single step. If lists containing instructions are present in the content, that is too much. The writer should re-design the content so that the instructions are in a separate troubleshooting topic, and then refer the user to that topic.

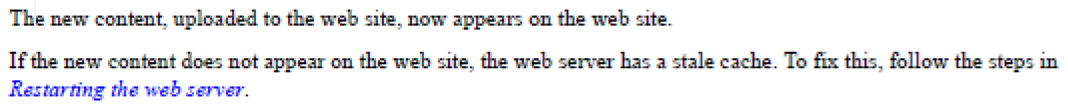
Task troubleshooting example

Scenario

Name	Web site update fails

Description A customer follows the steps in a task for updating web site content, but the web site is not showing the new content. This could be due to a stale web server cache. The customser should restart the web server.

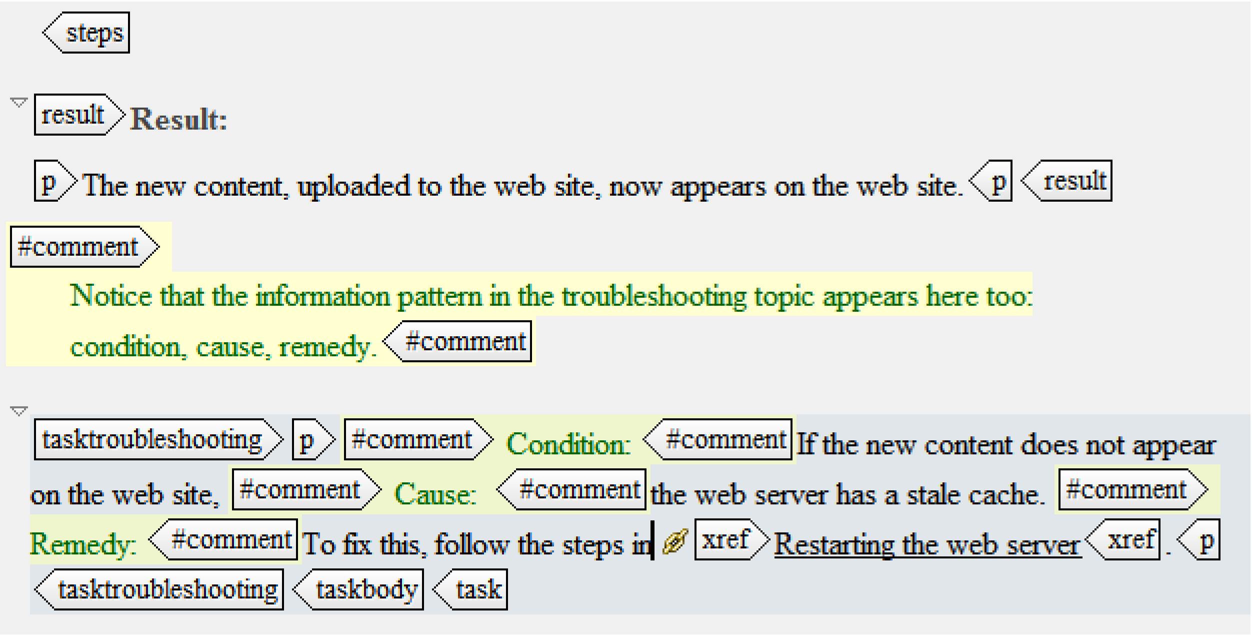
Output



The new content, uploaded to the web site, now appears on the web site.

If the new content does not appear on the web site, the web server has a stale cache. To fix this, follow the steps in *Restarting the web server*.

Tag view



steps

result **Result:**

p The new content, uploaded to the web site, now appears on the web site. p result

#comment Notice that the information pattern in the troubleshooting topic appears here too: condition, cause, remedy. #comment

tasktroubleshooting p #comment Condition: #comment If the new content does not appear on the web site, #comment Cause: #comment the web server has a stale cache. #comment Remedy: #comment To fix this, follow the steps in xref Restarting the web server xref . p tasktroubleshooting taskbody task

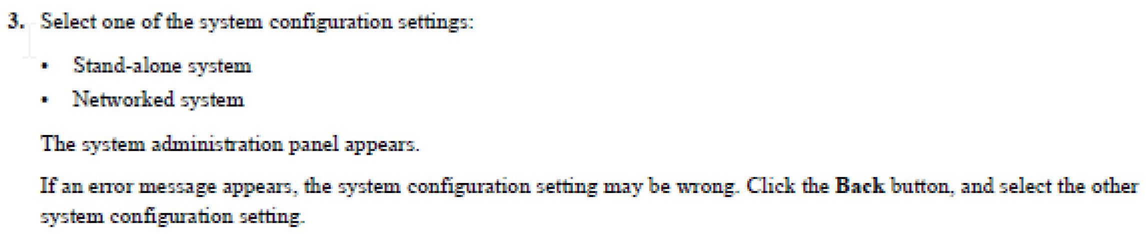
Step troubleshooting example

Scenario

Name Wrong option in step

Description A user has to select one of two system configuration settings in a step. If they select the wrong one, an error message displays. To recover, they need to click the **Back** button and select the other configuration setting.

Output



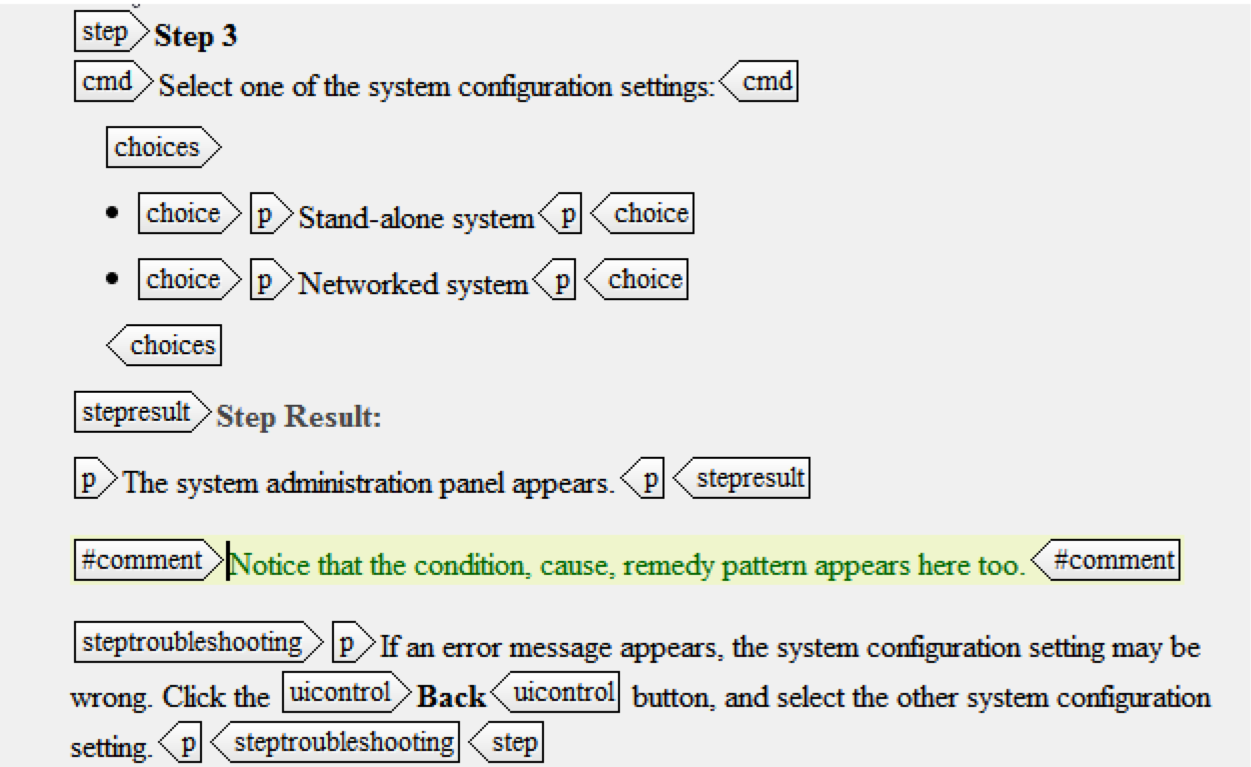
3. Select one of the system configuration settings:

    - Stand-alone system
    - Networked system

    The system administration panel appears.

    If an error message appears, the system configuration setting may be wrong. Click the **Back** button, and select the other system configuration setting.

Tag view



step **Step 3**

cmd Select one of the system configuration settings: cmd

choices

- choice p Stand-alone system p choice
- choice p Networked system p choice

choices

stepresult **Step Result:**

p The system administration panel appears. p stepresult

#comment Notice that the condition, cause, remedy pattern appears here too. #comment

steptroubleshooting p If an error message appears, the system configuration setting may be wrong. Click the uicontrol **Back** uicontrol button, and select the other system configuration setting. p steptroubleshooting step

Troubleshooting note-type

Scenario

Name Cannot see all menu items

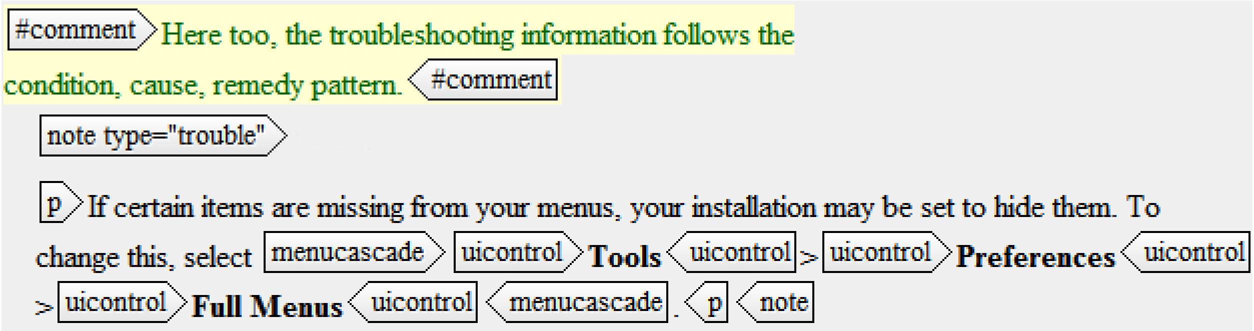
Description By default, a software product hides certain advanced menu items, however the product's documentation describes all of menus items. Customers following the documentation along on their systems have become confused when they did not see all of the menu items that were being described. The fix for this is to have the customer change their preferences to "Show Full Menus".

Output

Trouble?

If certain items are missing from your menus, your installation may be set to hide them. To change this, select **Tools** > **Preferences** > **Full Menus**.

Tag view



#comment Here too, the troubleshooting information follows the condition, cause, remedy pattern. #comment

note type="trouble"

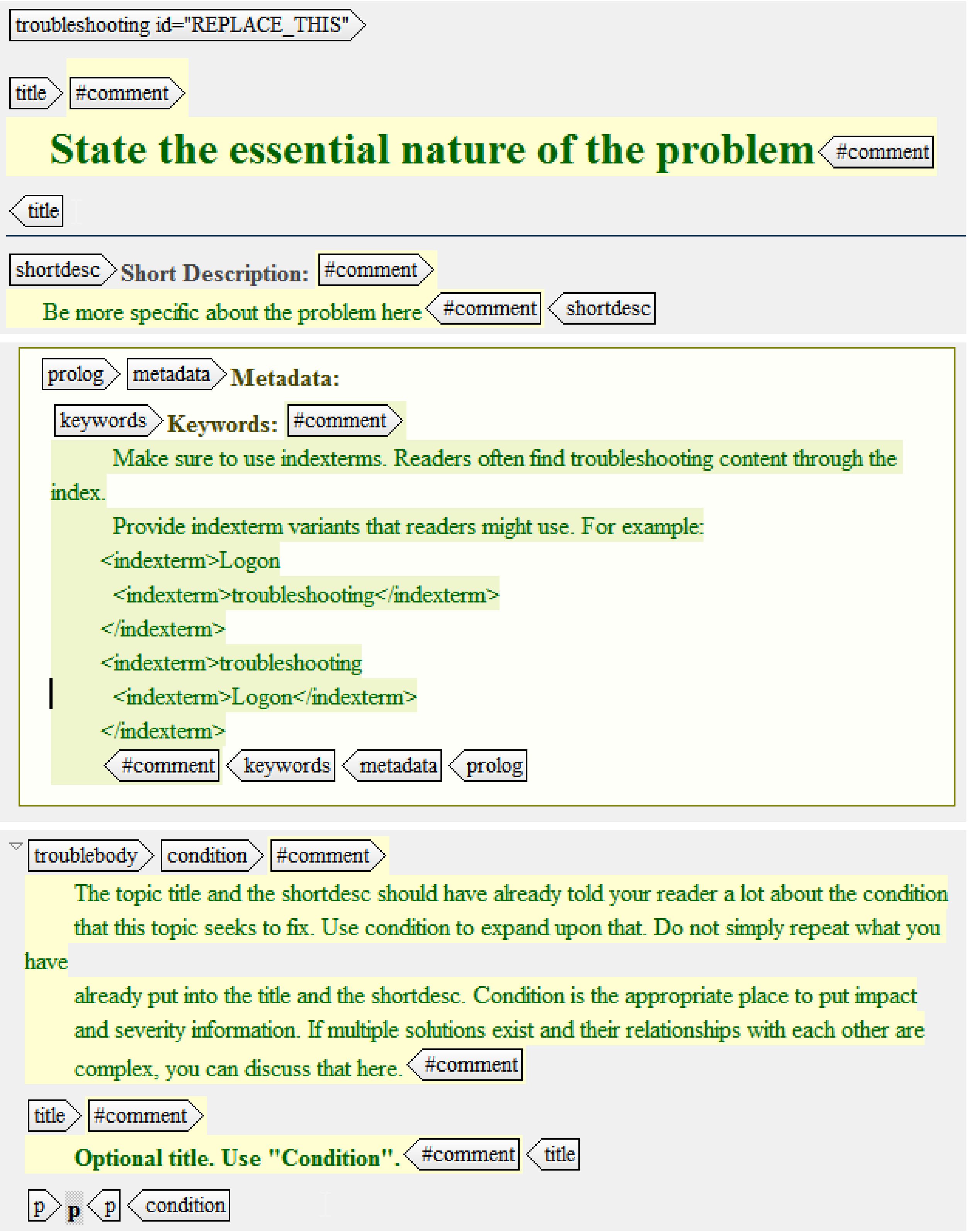
p If certain items are missing from your menus, your installation may be set to hide them. To change this, select menucascade uicontrol **Tools** uicontrol > uicontrol **Preferences** uicontrol > uicontrol **Full Menus** uicontrol menucascade . p note

Annotated troubleshooting template

> *This is a DITA troubleshooting topic template with usage comments.*

XML comments describe each element and how to use it. Formal descriptions for each troubleshooting element can be found in the *Darwin Information Typing Architecture (DITA) Version 1.3 OASIS Standard* Language Specification section.

Tag view



troubleshooting id="REPLACE_THIS"

title #comment

State the essential nature of the problem #comment

title

shortdesc **Short Description:** #comment

Be more specific about the problem here #comment shortdesc

prolog metadata **Metadata:**

keywords **Keywords:** #comment

Make sure to use indexterms. Readers often find troubleshooting content through the index.

Provide indexterm variants that readers might use. For example:

```
<indexterm>Logon
  <indexterm>troubleshooting</indexterm>
</indexterm>
<indexterm>troubleshooting
  <indexterm>Logon</indexterm>
</indexterm>
```

#comment keywords metadata prolog

troublebody condition #comment

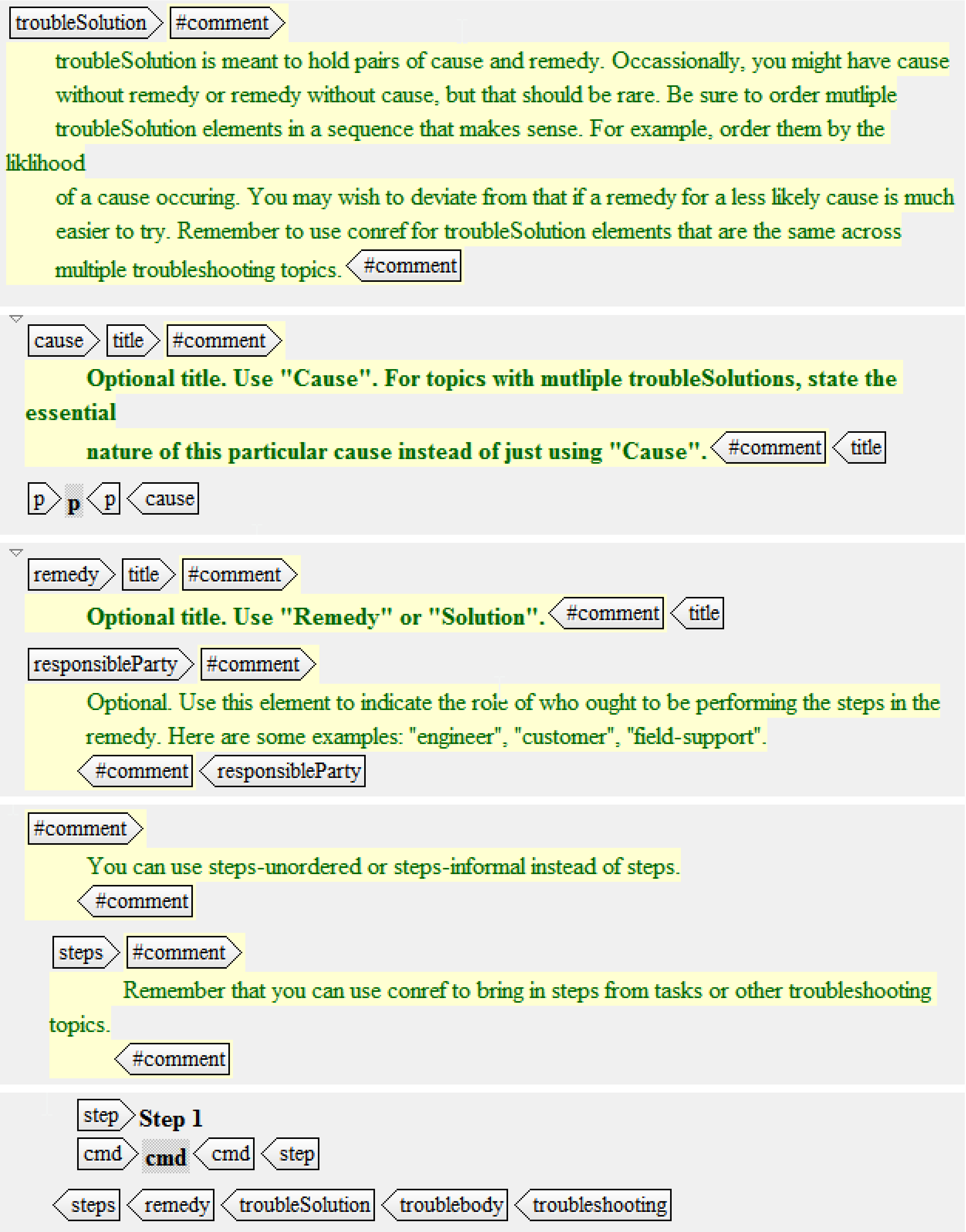
The topic title and the shortdesc should have already told your reader a lot about the condition that this topic seeks to fix. Use condition to expand upon that. Do not simply repeat what you have

already put into the title and the shortdesc. Condition is the appropriate place to put impact and severity information. If multiple solutions exist and their relationships with each other are complex, you can discuss that here. #comment

title #comment

Optional title. Use "Condition". #comment title

p **p** p condition



```
troubleSolution #comment
        troubleSolution is meant to hold pairs of cause and remedy. Occassionally, you might have cause
        without remedy or remedy without cause, but that should be rare. Be sure to order mutliple
        troubleSolution elements in a sequence that makes sense. For example, order them by the
liklihood
        of a cause occuring. You may wish to deviate from that if a remedy for a less likely cause is much
        easier to try. Remember to use conref for troubleSolution elements that are the same across
        multiple troubleshooting topics. #comment

cause title #comment
        Optional title. Use "Cause". For topics with mutliple troubleSolutions, state the
essential
        nature of this particular cause instead of just using "Cause". #comment title
p p p cause

remedy title #comment
        Optional title. Use "Remedy" or "Solution". #comment title
responsibleParty #comment
        Optional. Use this element to indicate the role of who ought to be performing the steps in the
        remedy. Here are some examples: "engineer", "customer", "field-support".
      #comment responsibleParty

#comment
        You can use steps-unordered or steps-informal instead of steps.
      #comment
  steps #comment
        Remember that you can use conref to bring in steps from tasks or other troubleshooting
topics.
      #comment

    step Step 1
    cmd cmd cmd step
  steps remedy troubleSolution troublebody troubleshooting
```

XML view

Here is the template's XML. It can be copied and pasted into an XML editor.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE troubleshooting
  PUBLIC "-//OASIS//DTD DITA Troubleshooting//EN"
  "troubleshooting.dtd">
<troubleshooting id="REPLACE_THIS">
  <title><!-- State the essential nature of the problem --></title>
  <shortdesc><!-- Be more specific about the problem here --></shortdesc>
  <prolog>
    <metadata>
      <keywords><!--
          Make sure to use indexterms. Readers often find troubleshooting content through the index.
          Provide indexterm variants that readers might use. For example:
        <indexterm>Logon
          <indexterm>troubleshooting</indexterm>
        </indexterm>
        <indexterm>troubleshooting
          <indexterm>Logon</indexterm>
        </indexterm>
        -->
      </keywords>
    </metadata>
  </prolog>
  <troublebody>
    <condition><!--
        The topic title and the shortdesc should have already told your reader a lot about the condition
        that this topic seeks to fix. Use condition to expand upon that. Do not simply repeat what you have
        already put into the title and the shortdesc. Condition is the appropriate place to put impact
        and severity information. If multiple solutions exist and their relationships with each other are
        complex, you can discuss that here.-->
      <title><!-- Optional title. Use "Condition". --></title>
      <p></p>
    </condition>
    <troubleSolution><!--
        troubleSolution is meant to hold pairs of cause and remedy. Occassionally, you might have cause
        without remedy or remedy without cause, but that should be rare. Be sure to order mutliple
        troubleSolution elements in a sequence that makes sense. For example, order them by the liklihood
        of a cause occuring. You may wish to deviate from that if a remedy for a less likely cause is much
        easier to try. Remember to use conref for troubleSolution elements that are the same across
        multiple troubleshooting topics.-->
      <cause>
        <title><!--
          Optional title. Use "Cause". For topics with mutliple troubleSolutions, state the essential
          nature of this particular cause instead of just using "Cause".-->
        </title>
        <p></p>
      </cause>
      <remedy>
        <title><!-- Optional title. Use "Remedy" or "Solution". -->
        </title>
        <responsibleParty><!--
          Optional. Use this element to indicate the role of who ought to be performing the steps in the
          remedy. Here are some examples: "engineer", "customer", "field-support".
        -->
        </responsibleParty>
        <!-- You can use steps-unordered or steps-informal instead of steps. -->
        <steps>
          <!-- Remember that you can use conref to bring in steps from tasks or other troubleshooting topics. -->
          <step>
            <cmd/>
          </step>
        </steps>
      </remedy>
    </troubleSolution>
  </troublebody>
</troubleshooting>
```