OASIS Committee Note

Using DITA 2.0 Troubleshooting Version 1.0

Working Draft 01

13 September 2021

This version:
Not yet applicable.

Previous version:
Not applicable.

Latest version:
Not yet applicable.

Technical Committee:
OASIS Darwin Information Typing Architecture (DITA) TC

Chair:
Kristen James Eberlein (kris@eberleinconsulting.com), Eberlein Consulting LLC

Editors:
Kristen James Eberlein (kris@eberleinconsulting.com), Eberlein Consulting LLC
Dawn Stevens (dawn.stevens@comtech-serv.com), Comtech Services, Inc.
Nancy Harrison (nharrison@infobridge-solutions.com), Individual member

Additional artifacts:
This document is one component of a Work Product that also includes:

- ZIP file that contains the DITA source for this document.
- ZIP file that contains sample DITA troubleshooting topics (?)

Related work:
This document is related to:

- *Darwin Information Typing Architecture Version 2.0.*
- *Darwin Information Typing Architecture for Technical Content Version 2.0.*

Abstract:
This committee note provides information about using DITA 2.0 to document troubleshooting information.

Status:

Citation format:

When referencing this note, the following citation format should be used:

[dita-tbs]

Using DITA 2.0 Troubleshooting Version 1.0. Edited by Kristen James Eberlein, Nancy Harrison, and Dawn Stevens. 13 September 2021. Working Draft 01.

Notices:

Table of contents

1 Introduction.......... 4
1.1 Terminology.......... 4
2 What is troubleshooting?.......... 5
2.1 Components of troubleshooting information.......... 5
2.2 Corrective action information.......... 5
2.3 Types of DITA troubleshooting information.......... 5
3 Why troubleshooting information is important.......... 7
3.1 Providing targeted information for readers experiencing a problem.......... 7
3.2 Providing easy-to-find information to users experiencing problems.......... 7
4 DITA troubleshooting features.......... 8
4.1 Goals of troubleshooting features.......... 8
4.2 Troubleshooting note.......... 8
4.3 Troubleshooting elements in the task topic.......... 8
4.4 Troubleshooting topic.......... 8
5 Examples of embedded troubleshooting information.......... 10
5.1 The troubleshooting note.......... 10
5.2 Step troubleshooting information.......... 11
5.3 Task troubleshooting information.......... 12
6 Examples of troubleshooting topics.......... 13
6.1 Simple troubleshooting topic.......... 13
6.2 Troubleshooting topic with multiple solutions.......... 15
6.3 Troubleshooting topic with diagnostic information.......... 18
A Informative references.......... 24
B Acknowledgments.......... 25
C Revision history.......... 26
D Notices.......... 27

1 Introduction

Comment by Kristen J Eberlein on 02 September 2021

At a minimum, I think that this topic should contain the following information:

- Information that troubleshooting features were introduced in DITA 1.3 and enhanced in DITA 2.0
- Fact that this is an update of a white paper produced by the Adoption TC in 2014
- Thanks to Bob Thomas, for championing the DITA 1.3 features and authoring the white paper
- Acknowledgement of Dawn Steven's effort in picking up on work that Bob Thomas was unable to complete and championing the DITA 2.0 feature of a `<diagnostics>` element

1.1 Terminology

Comment by Kristen J Eberlein on 02 September 2021
Do we have terms that we want to define? If not, remove this topic.

This topic provides information about terminology and how it is used in this committee note.

term
definition

2 What is troubleshooting?

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

Troubleshooting information can be brief, consisting of just a few sentences. Brief troubleshooting information is embedded within tasks or descriptions. Alternatively, troubleshooting information might be extensive. In these cases, the information warrants an entire topic.

2.1 Components of troubleshooting information

Troubleshooting information has a standard pattern that includes sequential components: condition, cause, and remedy.

Troubleshooting components are defined in the following way:

Condition
: A symptom, typically an undesirable state in a product, system, or service that a user wants to correct

Cause
: The underlying cause for the undesirable state

Remedy
: The action that a user can perform to restore the product, system, or service to its normal state

2.2 Corrective action information

Types of corrective action information that follow this standard pattern are alarm clearing, error resolution, and event response.

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.

2.3 Types of DITA troubleshooting information

This committee note covers the following types of DITA troubleshooting information.

Embedded troubleshooting information
: Embedded troubleshooting information appears within tasks or descriptions. It is brief. Despite its reduced size, embedded troubleshooting also follows the condition, cause, remedy pattern. Often, the condition or cause is implied by the information that surrounds it. Remedies are usually conveyed with a single sentence.

Simple troubleshooting topic with a single cause
: A simple troubleshooting topic typically includes a condition or symptom, an optional cause, and a remedy.

Simple troubleshooting topic with multiple causes

In some cases, there might 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.

Complex troubleshooting information

Comment by Kristen J Eberlein on 02 September 2021
What do we want to say here?

3 Why troubleshooting information is important

Troubleshooting information can be targeted to users who are seeking information in order to solve a problem. Troubleshooting information can be optimized to be easy for readers to locate.

3.1 Providing targeted information for readers experiencing a problem

Robust troubleshooting information is targeted to users experiencing problems.

Information developers recognize that many users seek information only when they experience a problem. First, users become aware of the problem, often because they receive an error message or an outcome that they expected fails to occur (for example, “the machine should turn on but does not”). Second, users articulate or define the problem, putting it into words (for example, “I cannot get this font to change size”). At this point, the users begin to search for content that might help them solve the problem.

At this stage, the users might be reading the manual, searching for relevant information. However, most manuals “pose formidable obstacles to finding problem-solving information.”[1]

Troubleshooting information that is minimalistic, well-structured, and explicitly targeted to the users experiencing problems can overcome the obstacles posed by traditional manuals.

3.2 Providing easy-to-find information to users experiencing problems

When troubleshooting information is developed in discrete units (topics, sections, steps, or specific types of notes), it is easier for users to locate. It also can be marked with specific cues (verbal or visual) to make it easily recognizable.

Research by Hans van der Meij, minimalism guru at the University of Twente in the Netherlands, tells us that problem-solving information should be marked by a visual or verbal cue to make it easily accessible to users. We recommend that you use images and labels to make the troubleshooting elements in your content quickly recognizable and easy-to-find.

[1] “Does the Manual Help? An Examination of the Problem-Solving Support Offered by Manuals,” *IEEE Transactions on Professional Communication,* Vol. 39, No. 3, September 1996.

4 DITA troubleshooting features

DITA includes support for a troubleshooting note, troubleshooting-specific elements in the task topic, and a specialized troubleshooting topic.

4.1 Goals of troubleshooting features

The troubleshooting components of the DITA architecture are designed to provide a structure for troubleshooting information that supports both authors and readers.

Ease of authoring
The DITA troubleshooting components make it easier for writers to author well-structured, minimalistic information that supports error recognition and recovery. The components enables writers to easily follow consistent patterns for troubleshooting information and so create tightly-focused information that helps readers solve specific patterns.

Ease of reading
Consistent patterns of troubleshooting information enable predictable organization. In combination with robust stylesheets, these patterns can produce titles, icons, and labels that enable readers to quickly locate troubleshooting information.

4.2 Troubleshooting note

The troubleshooting note supports adding brief troubleshooting information; it is designed to contain brief corrective action that pertains to its immediate, surrounding context.

The troubleshooting note is a `<note>` element with the `@type` attribute set to "trouble." It is part of the core DITA vocabulary and so is available in all DITA topics.

4.3 Troubleshooting elements in the task topic

The troubleshooting elements available in the task topic enable authors to provide error recovery information directly at the point of need.

Step troubleshooting
The `<steptroubleshooting>` element contains information that specifies corrective action to take when the result of a step is not as expected.

Task troubleshooting
The `<tasktroubleshooting>` element contains information that specifies corrective action to take when the result of a task is not as expected.

4.4 Troubleshooting topic

The troubleshooting topic provides strategies for isolating and solving potential problems end users might encounter when using a product. A troubleshooting topic typically answers a "How Do I Identify and Resolve..." question.

In contrast to the embedded troubleshooting elements within task, the troubleshooting topic provides a place for authors to provide robust error diagnostic and recovery information. The troubleshooting topic models the specific structure of typical troubleshooting information, providing semantic elements to contain a description of the problem (`<condition>`), the reason the problem has occurred (`<cause>`), and how to fix the problem (`<remedy>`). Recognizing that some problems may have many causes and therefore many solutions, a single troubleshooting topic may include any number of `<cause>` and `<remedy>` elements grouped together within `<troubleSolution>` elements designed to associate causes with their related solutions. In such situations, users may not readily know which cause applies to their situation, so the troubleshooting topic also allows for a `<diagnostics>` section that assists users in determining the cause associated with their specific situations.

The troubleshooting topic enhances reusability of content across an organization:

- Many problems are common across different products. By isolating troubleshooting information into individual topics, authors can more easily select the specific problems for their product they are documenting. Further, the exact mixture of causes may vary across those products. By associating causes and remedies into `<troubleSolution>` elements, authors can easily reuse only those cause/remedy associations that apply.
- Many solutions often repeat the steps for performing the task – the problem occurred, perhaps, because the end user didn't follow the written instructions in the first place. However, users in this situation don't necessarily realize they should try again by referencing the task topic, but are looking for troubleshooting information, often found in another portion of a manual or even another document all together. Because `<remedy>` contains the same `<steps>`, `<steps-unordered>`, or `<steps-informal>` elements used in task topics, authors can easily reuse the same instructions in both places.

Troubleshooting elements

All elements within the troubleshooting topic are optional. However, if a `<troubleSolution>` is provided at least one of `<cause>` or `<remedy>` must be included within it. Similarly, if `<diagnostics>` is included, it must contain at least one of `<diagnostics-general>` or `<diagnostics-steps>`

<condition>
: Describes the problem. The condition should expand on the title and short description (if included) and provide information such as the observable symptoms of the problem and where and when the problem might occur.

<diagnostics>
: Container for related `<diagnostics-general>` and `<diagnostics-steps>` elements.

<diagnostics-general>
: Includes non-procedural information (such as a diagnostic table or flowchart) that can help determine which of multiple causes apply to a specific situation. Results of the diagnosis may link to the specific cause-remedy section or stand-alone topic that applies to each result.

<diagnostics-steps>
: Includes step-by-step information that can help determine which of multiple causes apply to a specific situation. Results of each diagnostic step may link to the specific cause-remedy section or stand-alone topic that applies to each result.

<troubleSolution>
: Container for related `<cause>` and `<remedy>` elements.

<cause>
: Explains the reason that a problem might occur.

<remedy>
: Container for the steps required to resolve the problem. Nest a `<steps>`, `<steps-unordered>`, or `<steps-informal>` element within the `<remedy>` element to provide step-by-step instructions and descriptions for diagnosing and solving the problem.

<responsibleParty>
: Indicates that the instructions within the `<remedy>` should be completed by a specific person.

5 Examples of embedded troubleshooting information

There are several types of embedded troubleshooting information: troubleshooting note, step troubleshooting, and task troubleshooting. This chapter contains examples of each.

5.1 The troubleshooting note

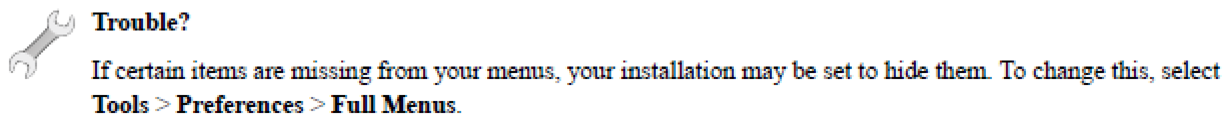
The troubleshooting note is contained in a `<note>` element with the `@type` attribute set to "trouble".

Scenario

In this scenario, a user is reading the online help for a software product. By default, the product does not display all of the advanced menu items. Users reading the documentation while viewing the software application have become confused when they did not see all of the menu items that are described in the documentation. The solution is for the users to change their preferences so that the application shows all menu items.

Rendered output

The following screen capture shows the rendered output for the troubleshooting note:



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**.

XML source

The following code block shows the markup for the troubleshooting note:

```
<note type="trouble">
  If certain items are missing from the menus, your installation might be
  configured to hide them. To change this, click
    <menucascade>
      <uicontrol>Tools</uicontrol>
      <uicontrol>Preferences</uicontrol>
      <uicontrol>Full Menus</uicontrol>
    </menucascade>.
</note>
```

Writing best practices

The following table shows how the information follows the sequential pattern of condition, cause, and remedy:

Type of information	Content
Condition	If certain items are missing from the menus ...
Cause	… your installation might be configured to hide them.
Remedy	To change this, click **Tools > Preferences > Full Menus**.

Writers should limit the content of the `<note>` element to discussing the condition, cause, and remedy expressed in a single step. If the content of the remedy requires a list, then the writer should redesign the content so that the instructions are in a separate troubleshooting topic.

5.2 Step troubleshooting information

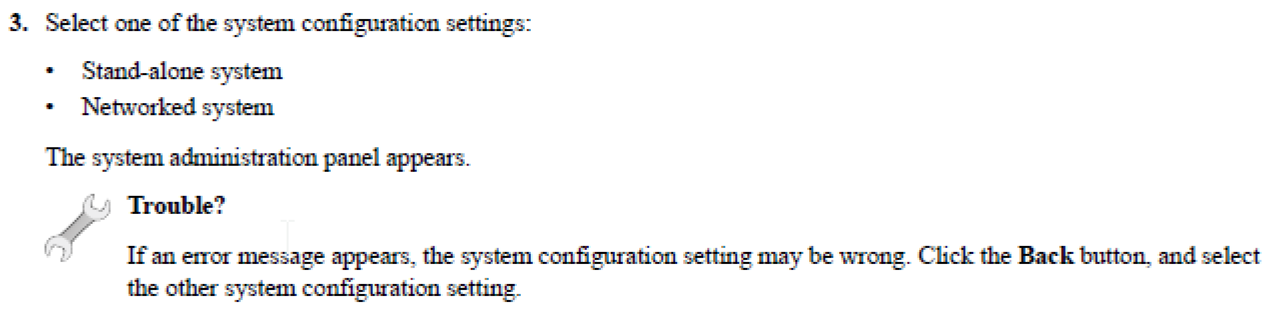
Step troubleshooting information is contained in the `<steptroubleshooting>` element. The element is located after the optional `<stepresult>` element.

Scenario

In this scenario, a user of a software application has to select a configuration setting as they perform a task. If they select the wrong setting, an error message is displayed. To recover, the user needs to click the **Back** button and select the other setting.

Rendered output

The following screen capture shows the step troubleshooting information as rendered:


3. Select one of the system configuration settings:
   - Stand-alone system
   - Networked system

   The system administration panel appears.

   Trouble?

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

XML source

The following code block shows the markup for the step troubleshooting information:

```
<step>
  <cmd>Select one of the following configuration settings:</cmd>
  <choices>
    <choice><uicontrol>Stand-alone system</uicontrol></choice>
    <choice><uicontrol>Networked system</uicontrol></choice>
  </choices>
  <steptroubleshooting>
      If an error message is displayed, the system configuration
      setting might be wrong. Click <uicontrol>Back</uicontrol>, and
      then select the other configuration setting.
  </steptroubleshooting>
</step>
```

Writing best practices

The following table shows how the information follows the sequential pattern of condition, cause, and remedy:

Type of information	Content
Condition	If an error message is displayed ...
Cause	… the system configuration setting might be wrong.

Type of information	Content
Remedy	Click **Back**, and then select the other configuration setting.

Writers should limit the content of the `<stepTroubleshooting>` element to discussing the condition, cause, and remedy expressed in a single step. If the content of the remedy requires a list, then the writer should redesign the content so that the instructions are in a separate troubleshooting topic.

5.3 Task troubleshooting information

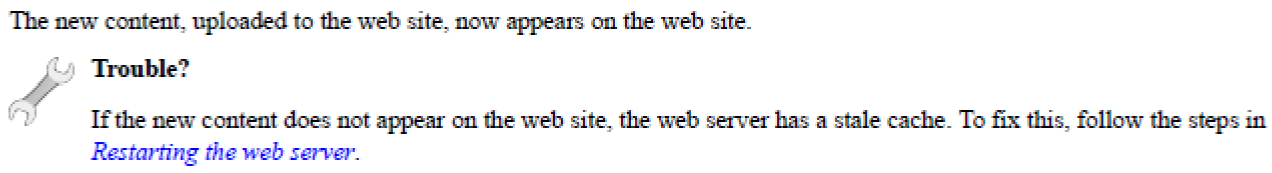
Task troubleshooting information is contained in the `<tasktroubleshooting>` element. The element is located after the optional `<results>` element in a task topic.

Scenario

In this scenario, a user follows the steps in a task for updating Web site content, but the new content is not displayed. This might be due to a stale cache on the Web server. The solution is for the user to restart the Web server.

Rendered output

The following screen capture shows the output of the task troubleshooting information:



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

Trouble?

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*.

XML source

The following code block shows the markup for the task troubleshooting information:

```
<tasktroubleshooting>If the new content is not displayed on the Web site, the Web server
          might have a stale cache. To fix this, follow the steps in <xref
keyref="restarting-web-server"/>.</tasktroubleshooting>
```

Writing best practices

The following table shows how the information follows the sequential pattern of condition, cause, and remedy:

Type of information	Content
Condition	If the new content is not displayed on the Web site ...
Cause	… the Web server might have a stale cache.
Remedy	To fix this, follow the steps in "Restarting the Web server."

Writers should limit the content of the `<tasktroubleshooting>` element to discussing the condition, cause, and remedy expressed in a single step. If the content requires a list, then the writer should redesign the content so that the instructions are in a separate troubleshooting topic.

6 Examples of troubleshooting topics

This chapter contains examples of a simple troubleshooting topic, a troubleshooting topic with multiple solutions, and a troubleshooting topic with diagnostic information.

6.1 Simple troubleshooting topic

We cover a simple scenario that can be handled in a troubleshooting topic with a single cause and remedy pair. We will show both the rendered output, XML source, and writing best practices.

Scenario

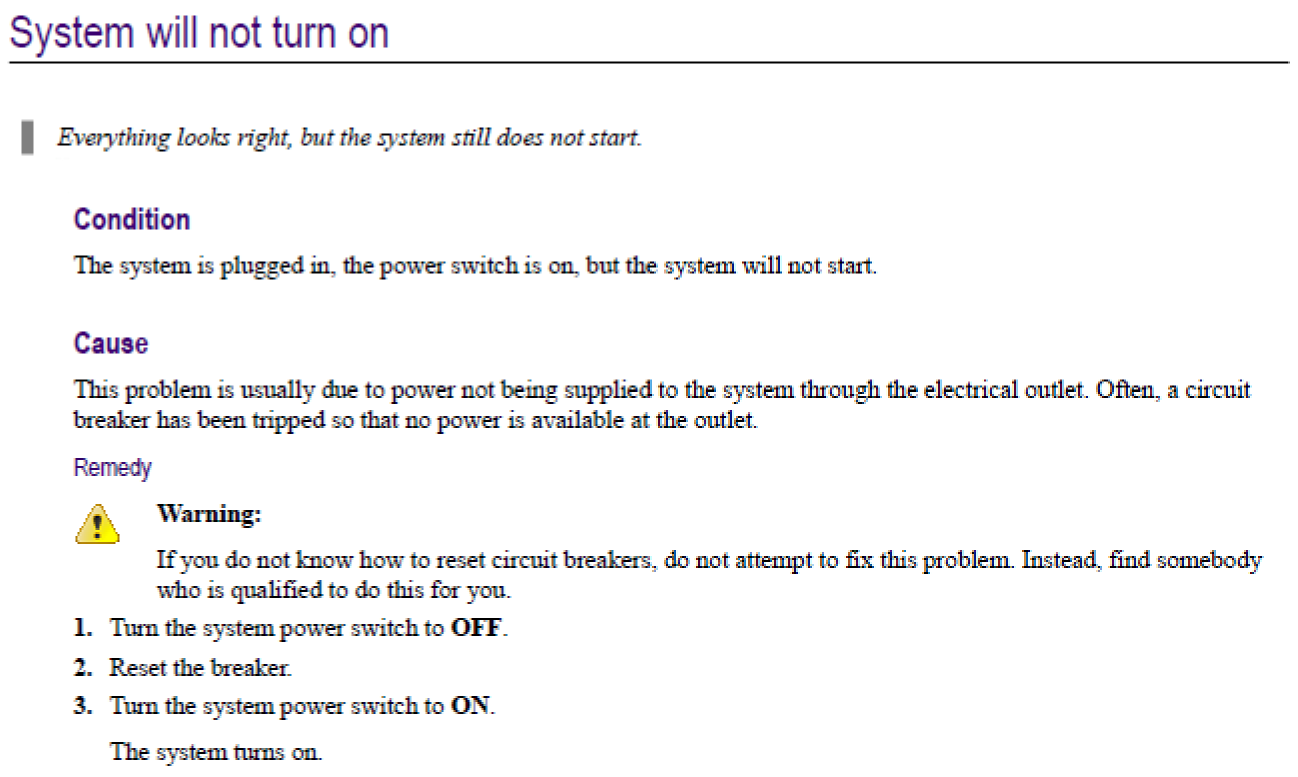
In this example, we cover simple troubleshooting for a tripped circuit breaker. A single `<troubleSolution>` can be used to document the corrective action.

The following information summarizes the problems that the user encounters:

- 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.

Rendered output

The following screen capture shows rendered output for the simple troubleshooting topic:



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 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.

XML source

The following code block shows the markup for the simple troubleshooting topic.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE troubleshooting PUBLIC "-//OASIS//DTD DITA Troubleshooting//EN"
"troubleshooting.dtd">
<troubleshooting id="code_simple_tbs_topic">
    <title>System will not turn on</title>
    <shortdesc>Everything looks right, but the system still does not start.</shortdesc>
    <troublebody>
        <condition>
            <title>Condition</title>
            <p>The system is plugged in, the power switch is on, but the system will not
start.</p>
        </condition>
        <troubleSolution>
            <cause>
                <title>Cause</title>
                <p>The problem is usually due to the 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>
                <title>Remedy</title>
                <steps>
                    <stepsection>
                        <note type="warning">If you do not know how to reset circuit
breakers, do
                            not attempt to fix this problem. Find someone qualified to
perform the
                            task.</note>
                    </stepsection>
                    <step>
                        <cmd>Turn the system power switch in <uicontrol>OFF</uicontrol>.</cmd>
                    </step>
                    <step>
                        <cmd>Reset the breaker.</cmd>
                    </step>
                    <step>
                        <cmd>Turn the system power switch to <uicontrol>ON</uicontrol>.</cmd>
                    </step>
                </steps>
            </remedy>
        </troubleSolution>
    </troublebody>
</troubleshooting>
```

Writing best practices

The following table provides best practice information for the elements used in the simple troubleshooting topic:

Element	Best practices	Example
`<title>`	Describe the problem from the user's point of view. Describe the symptoms rather than the cause.	System will not turn on
`<shortdesc>`	Provide additional information or context.	Everything looks right, but the system still does not start.
`<condition>`/`<title>`	Use "Condition" for the title unless you have good reason to do something else. Be consistent from	**Condition**

Element	Best practices	Example
	one troubleshooting topic to the next. Considering having your stylesheets add the title for the `<condition>` element.	
`<condition>`	Provide simple elaboration on what has already appeared in the topic title and short description. Only include information that is directly related to the condition or the symptom that the topic resolves.	The system is plugged in, the power switch is on, but the system will not start
`<cause>`/`<title>`	Use "Cause" for the title unless you have good reason to do otherwise.	**Cause**
`<cause>`	Describe only the origin of the problem that is resolved by the remedy.	The problem is usually due to the 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.

6.2 Troubleshooting topic with multiple solutions

This example covers a troubleshooting scenario that can be covered in a single troubleshooting topic, although it includes multiple solutions for a problem.

Scenario

In this example, we cover a scenario in which a user cannot log into the system.

The following list summarizes the causes for the user problem:

- No user account exists
- The user has forgotten their user ID or password
- An unidentified problem

Each cause has an associated remedy.

Assumptions and topic design

The assumption is that the user will try each cause and remedy pair in a sequential order until the problem is resolved. This design has implications for which elements are used and where titles are located.

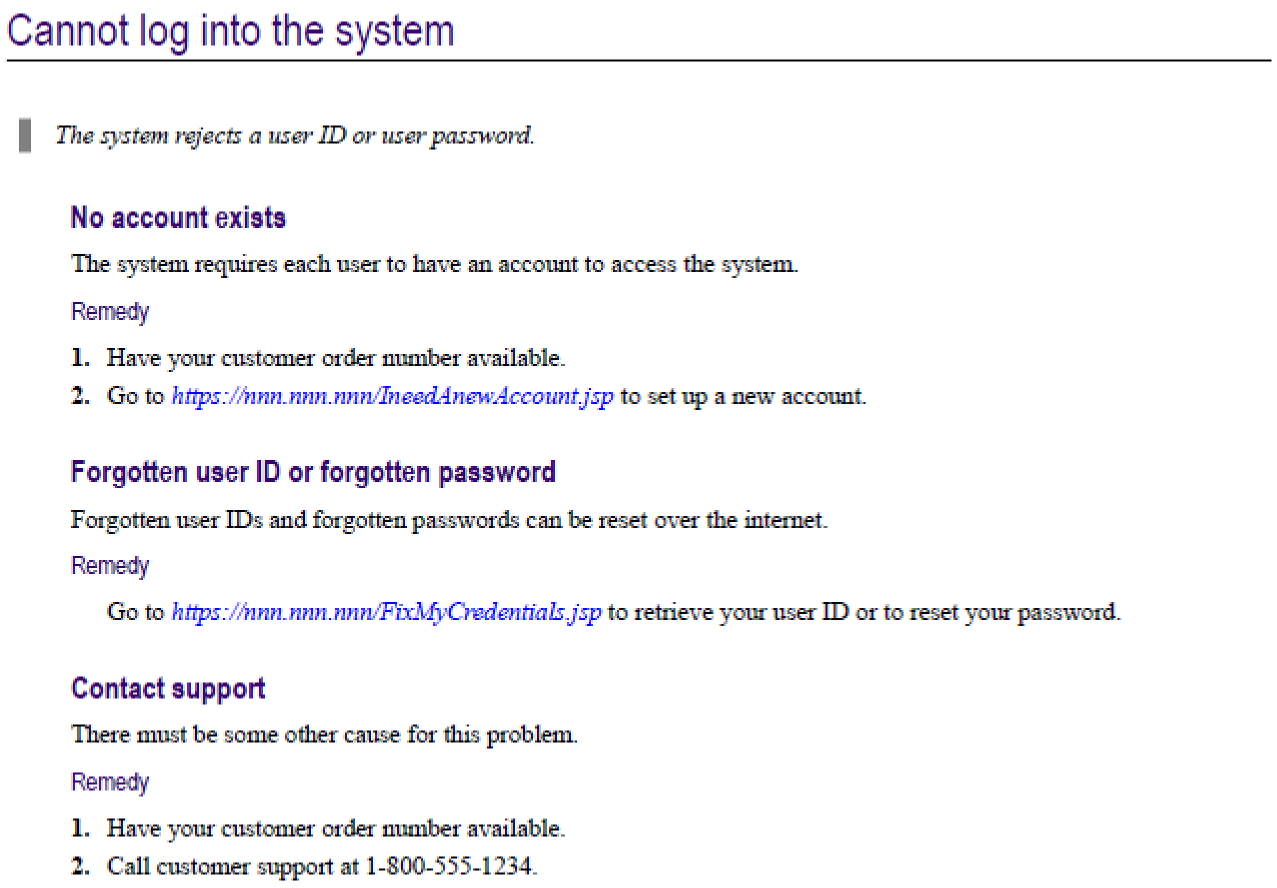
The first `<troubleSolution>` contains the “no account exists” cause and remedy. This `<troubleSolution>` must be first because there is no point in a user trying to reset account credentials for a non-existing account. The second `<troubleSolution>` contains the "forgotten user ID or password" cause and remedy. The third `<troubleSolution>` contains the final fallback: calling customer support

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 short description.

Rendered output

The following screen capture shows the rendered output for the troubleshooting topic that contains multiple solutions:



Cannot log into 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.

XML source

The following code block shows the markup for this troubleshooting topic with multiple solutions:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE troubleshooting PUBLIC "-//OASIS//DTD DITA Troubleshooting//EN"
"troubleshooting.dtd">
<troubleshooting id="code-cannot-log-into-the-system">
  <title>Cannot log into the system</title>
    <shortdesc>The user cannot log into the system. Multiple solutions
        exist for the problem, which the user should try in
        sequence.</shortdesc>
    <troublebody>
      <troubleSolution>
        <cause>
          <title>No user account exists</title>
          <p>The system requires that each user have an account to access the system.</p>
        </cause>
        <remedy>
          <title>Remedy</title>
            <steps>
```

```
              <step>
                <cmd>Have your customer account number available.</cmd>
              </step>
              <step>
                <cmd>Go to <xref keyref="create-new-account"/> to set up a new account.</cmd>
              </step>
            </steps>
          </remedy>
        </troubleSolution>
        <troubleSolution>
          <cause>
            <title>Forgotten user ID or password</title>
            <p>Forgotten user IDs and passwords can be reset.</p>
          </cause>
          <remedy>
            <title>Remedy</title>
            <steps>
              <step>
                <cmd>Go to <xref keyref="fix-credentials"/> to retrieve your user ID or reset
your
                     password.</cmd>
              </step>
            </steps>
          </remedy>
        </troubleSolution>
        <troubleSolution>
          <cause>
            <title>Contact support</title>
            <p>There is an unidentified cause for this solution.</p>
          </cause>
          <remedy>
            <title>Remedy</title>
            <steps>
              <step>
                <cmd>Have your customer order number available.</cmd>
              </step>
              <step>
                <cmd>Contact customer support at <keyword keyref="customer-support-phone-
number"/>.</cmd>
              </step>
            </steps>
          </remedy>
        </troubleSolution>
    </troublebody>
</troubleshooting>
```

Writing best practices

The following table provides best practice information for the elements that are used in the troubleshooting topic with multiple solutions.

Element	Best practices	Example
`<title>`	Describe the problem from the user's point of view. Describe the symptoms rather than the cause.	Cannot log into the system
`<shortdesc>`	Provide additional information or context.	The user cannot log into the system. Multiple solutions exist for the problem, which the user should try in sequence.
`<condition>`	Do not use. All pertinent information about the condition is provided in the topic title and short description.	

Element	Best practices	Example
	Comment by Kristen J Eberlein on 02 September 2021 Do we want instead to say "Only use if all pertinent information about the condition cannot be provided in the topic title and short description? If we do make that recommemdation, how would it affect stylesheets?	
`<cause>/<title>`	Briefly describe the cause.	**Forgotten user ID or password**
`<cause>`	Describe only the origin of the problem that is resolved by the remedy.	Forgotten user IDs or passwords can be reset.
`<remedy>`	Use `<steps>`, `<steps-informal>`, or `<steps-unordered>` to provide a solution.	Go to <xref keyref="fix-credentials"/> to retrieve your user ID or reset your password.

6.3 Troubleshooting topic with diagnostic information

This example covers a complex troubleshooting scenario that contains diagnostic information.

Scenario

In this example, we cover a complex scenario in which a fiber-optic switching system issues an alarm. The impact is that service is degraded, and the alarm should be retired within 24 hours. Unfortunately, the alarm does not provide users with any information that they could use to troubleshoot the information.

To provide more background: The fiber-optic switching system can issue any of 49 alarms. 42 of these alarms are likely to happen during normal operation. Consequently, the software architecture contains logic to evaluate several system state variables, and so it can report a specific cause to the user along with the alarm code. But, the likelihood of the remaining seven alarms occurring is small enough that the software architecture does little more than issue an alarm. Consequently, the product documentation must incorporate diagnostic steps into the troubleshooting topics for each of those seven alarms.

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

Assumptions and topic design

This scenario could be covered in a single troubleshooting topic, or it could be represented by a DITA map that referenced individual troubleshooting topics for each cause and remedy pair. In this example, we use a single troubleshooting topic.

Rendered output

The following screen captures show the rendered output for the troubleshooting topic that contains diagnostic information:

EJOL - excessive jitter on line

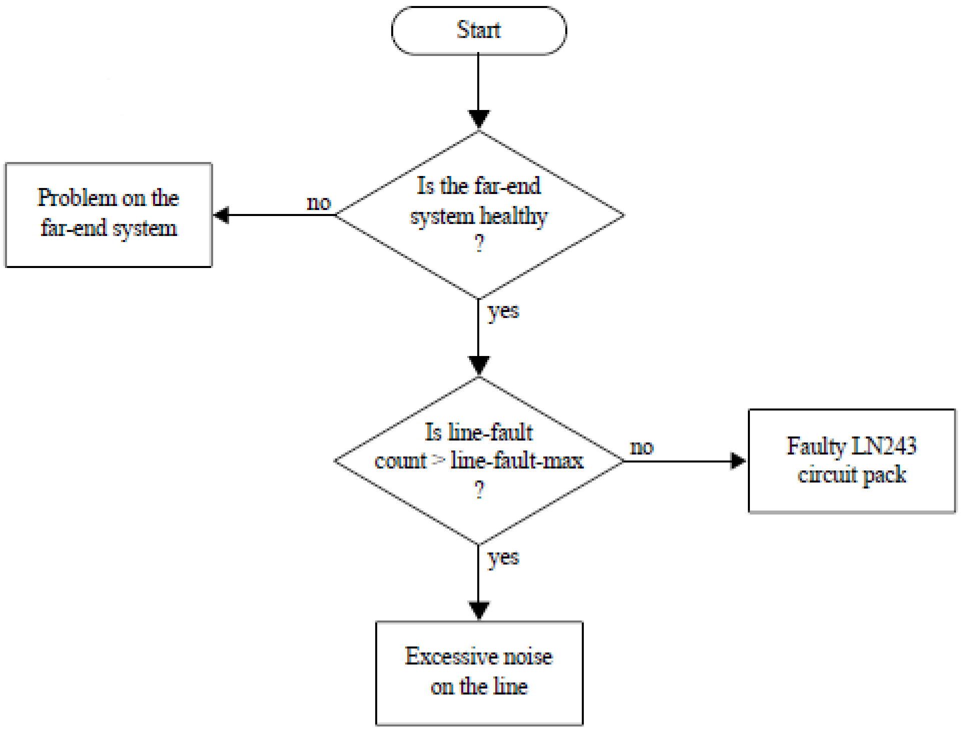
The system has experienced an "excessive jitter on line" alarm

Alarm classification

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

Determine the cause

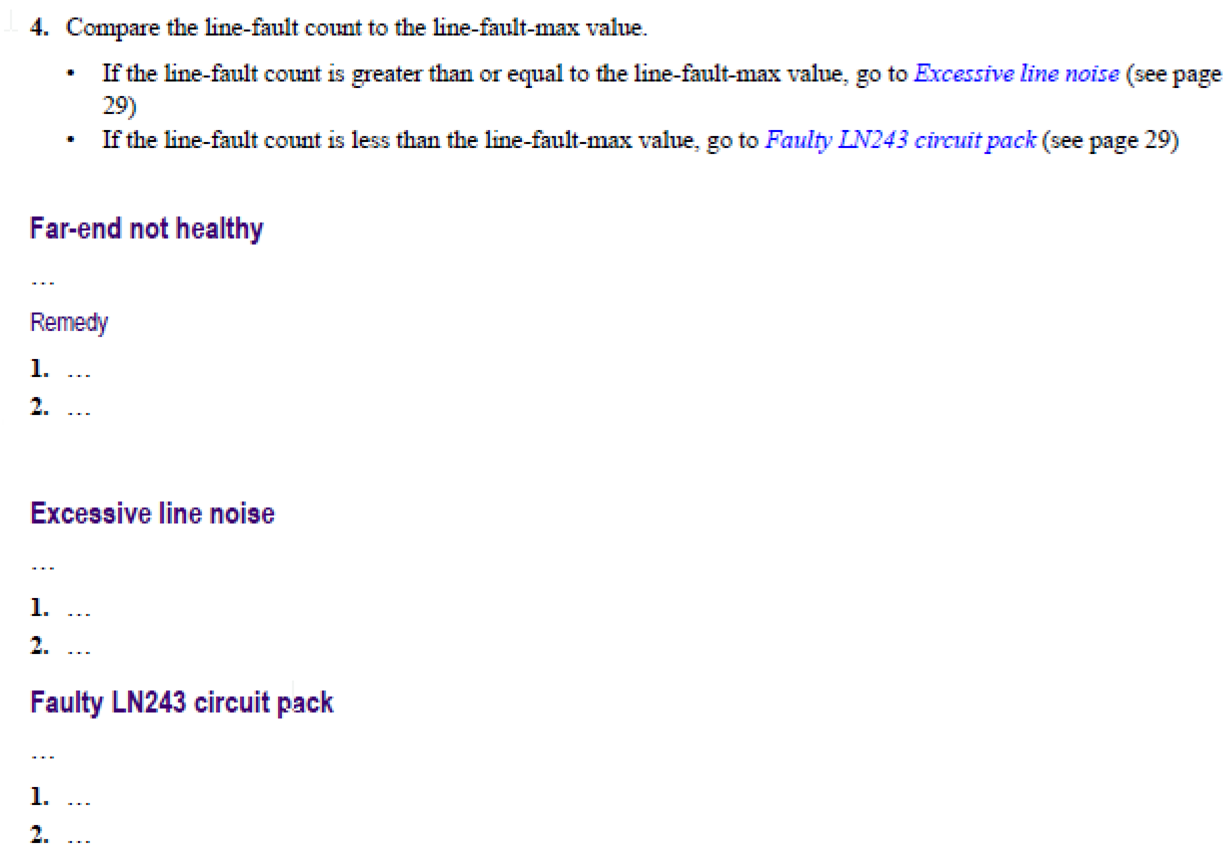
The EJOL alarm can occur due to: trouble on the far-end system, excessive line noise, or a faulty LN243 circuit pack. Here is a flowchart that shows the process for diagnosing the cause.


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* (see page 29)
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* (see page 29)
   - If the line-fault count is less than the line-fault-max value, go to *Faulty LN243 circuit pack* (see page 29)

Far-end not healthy

…

Remedy

1. …
2. …

Excessive line noise

…

1. …
2. …

Faulty LN243 circuit pack

…

1. …
2. …

XML source

The following code block shows the markup for the troubleshooting topic with diagnostic information:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE troubleshooting PUBLIC "-//OASIS//DTD DITA 2.0 Troubleshooting//EN"
"troubleshooting.dtd">
<troubleshooting id="ejol">
  <title>EJOL: excessive jitter on line</title>
  <shortdesc>The system experienced an <q>excessive jitter on line</q>
    alarm. Although the category is minor, the alarm should be cleared
    within 24 hours.</shortdesc>
  <troublebody>
    <diagnostics>
      <diagnostics-general>
        <title>Determine the cause</title>
        <p>The EJOL alarm occurs due to the following issues:</p>
        <ul>
          <li>Trouble on the far-end system</li>
          <li>Excessive line noise</li>
          <li>Faulty LN243 circuit pack</li>
        </ul>
        <p>Below is a flow chart that shows the process for diagnosing the
          cause:</p>
        <image href="images/ejol-diagnostics-flowchart.png"/>
      </diagnostics-general>
      <diagnostics-steps>
        <steps>
          <step>
            <cmd>Query the state of health of the far-end system.</cmd>
            <info>If the far-end system is not healthy, go to <xref
                href="#ejol/far-end-not-healthy"/>.</info>
          </step>
          <step>
            <cmd>To retrieve the line-fault count, click <menucascade>
                <uicontrol>Operations</uicontrol>
                <uicontrol>Performance</uicontrol>
                <uicontrol>line</uicontrol>
              </menucascade>.</cmd>
          </step>
          <step>
            <cmd>To retrieve the line-fault-max value, click <menucascade>
                <uicontrol>Administration</uicontrol>
                <uicontrol>Transmission</uicontrol>
                <uicontrol>Settings</uicontrol>
              </menucascade>.</cmd>
          </step>
          <step>
            <cmd>Compare the line-fault count to the line-fault-max
              value:</cmd>
            <choices>
              <choice>If the line-fault count is greater than or equal to
                the line-fault-max value, go to <xref
                  href="#ejol/excessive-line-noise"/>.</choice>
              <choice>If the line-fault count is less than the
                line-fault-max value, go to <xref
                  href="#ejol/faulty-LN243-line-pack"/>.</choice>
            </choices>
          </step>
        </steps>
      </diagnostics-steps>
    </diagnostics>
    <troubleSolution id="far-end-not-healthy">
      <cause>
        <title>Far-end not healthy</title>
        <p>...</p>
      </cause>
      <remedy>
        <steps>
          <step>
            <cmd>...</cmd>
          </step>
          <step>
            <cmd>...</cmd>
          </step>
          <step>
            <cmd>...</cmd>
          </step>
        </steps>
      </remedy>
    </troubleSolution>
    <troubleSolution id="excessive-line-noise">
      <cause>
        <title>Excessive line noise</title>
        <p>...</p>
      </cause>
      <remedy>
        <steps>
          <step>
            <cmd>...</cmd>
          </step>
          <step>
            <cmd>...</cmd>
          </step>
          <step>
            <cmd>...</cmd>
          </step>
        </steps>
      </remedy>
    </troubleSolution>
    <troubleSolution id="faulty-LN243-line-pack">
      <cause>
        <title>Faulty LN243 circuit pack</title>
        <p>...</p>
      </cause>
      <remedy>
        <steps>
```

```
           <step>
             <cmd>...</cmd>
           </step>
           <step>
             <cmd>...</cmd>
           </step>
           <step>
             <cmd>...</cmd>
           </step>
         </steps>
       </remedy>
     </troubleSolution>
   </troublebody>
 </troubleshooting>
```

Writing best practices

The following table provides best practice information for the elements that are used in the troubleshooting topic with multiple solutions.

Element	Best practices	Example
`<title>`	Describe the problem from the user's point of view. Describe the symptoms rather than the cause.	EJOL: excessive jitter online
`<shortdesc>`	Provide additional information or context.	The system experienced an "excessive jitter on line" alarm. Although the category is minor, the alarm should be cleared within 24 hours.
`<diagnostics-general>`/`<title>`		**Determine the cause**
`<diagnostics-general>`	Provide information that explains X	The EJOL alarm occurs due to the following issues: • Trouble on the far-end system • Excessive line noise • Faulty LN243 circuit pack Below is a flow chart that shows the process for diagnosing the cause: ...
`<diagnostics-steps>`	Provide the process for diagnosing the problem. Use stylesheets to insert a title.	**1.** Query the state of health of the far-end system ... **2.** ... **3.** ... **4.** ...
`<condition>`	Do not use. All pertinent information about the condition is provided in the topic title, short description, and diagnostics information.	
`<cause>`/`<title>`	Briefly describe the cause.	**Far-end not healthy**

Element	Best practices	Example
`<cause>`	Describe only the origin of the problem that is resolved by the remedy.	...
`<remedy>`	Use `<steps>`, `<steps-informal>`, or `<steps-unordered>` to provide a solution.	**1.** ... **2.** ... **3.** ...

A Informative references

This appendix contains the references that are used in this document.

While any hyperlinks included in this appendix were valid at the time of publication, OASIS cannot guarantee their long-term validity.

[dita-1.3-troubleshooting]
DITA 1.3 Feature Article: Using DITA 1.3 Troubleshooting. Authored by Bob Thomas on behalf of the OASIS DITA Adoption Committee. Published 03 July 2014. Principal URL: https://www.oasis-open.org/committees/download.php/53516/DITA13TroubleshootingArticle_FINAL_03jul14.pdf.

[van-der-Meij-1996]
Hans van der Meij. “Does the Manual Help? An Examination of the Problem-Solving Support Offered by Manuals”. *IEEE Transactions on Professional Communication*. Volume 39, number 3: September 1996.

B Acknowledgments

The following individuals were members of the technical committee during the creation of this document, and their contributions are gratefully acknowledged.

Kristen James Eberlein, Eberlein Consulting LLC
Nancy Harrison, Individual member
Dawn Stevens, Comtech Services, Inc.

Comment by Kristen J Eberlein on 02 September 2021
We will add the names of additional TC members who provide feedback, participate in a review, etc.

In addition, the OASIS DITA Technical Committee also would like to recognize the following people for their expertise, insights, and assistance:

<Name>
<Name>

C Revision history

The following table contains information about revisions to this document.

Revision	**Date**	**Editor**	**Description of changes**
01	02 September 2021	Kristen James Eberlein	First draft of committee note, based on reworked content from the *DITA 1.3 Feature Article: Using DITA 1.3 Troubleshooting*.
02	07 September 2021	Kristen James Eberlein	New content for "Troubleshooting topic" authored by Dawn Stevens.
03	12 September 2021	Kristen James Eberlein	Incorporated editorial corrections from Silke Achterfeld, Ericcson.

D Notices