

KCS Solution Quality Quick Reference Guide (an example)

Guidelines for Creating Findable, Useable Solutions

Problem/Question

As a matter of usability, it's helpful to see clear, unique problems or questions when searching for solutions.

Problems - tell us what bad things are happening in the customers words.

1. Don't create "compound statements" - keep the environment terms out of the problem if you can.
No: Problem: 3Com NIC X1000 has the following error message: Comu.dll triggered an error in an invalid page in the module Comu.dll.
Yes: Problem: Error: "Comu.dll triggered an error in an invalid page"
 Environment: 3Com X1000
 Module - Comu.dll
2. Make the thoughts complete:
Yes: Problem: Program crashes on startup with an error.
 Problem: Error: "Program crashes due to insufficient memory"

Error Messages

- Error: "<exact error message text>"
- Error: "Cannot start program. Required application not recognized"

Ordering Problems

If your solution has multiple Problems, order them in the solution as follows:

- Less detailed first (generic)
- More detailed to follow (specific)

Example

- Cannot print a file
- Error printing file to network printer
- Error: "Invalid page layout for this printer driver. (24301)"

Things you don't need to say!

Certain phrases are unnecessary when writing statements:

"I want to", "The customer is trying to"
 "The customer is using..."
 "The customer is getting..."
 "It worked okay before I..."

Just get to the point!

Verb Tense

- Write in present tense.
- Don't tell us what you did, tell us what to do!

Use Explicit Subjects

- **Implicit Subject:** Won't print.
 (Unclear - What won't print?)
- **Explicit Subject:** Documents do not print.
 (Better)

Solution Types

Describe Intent of Solution

Problem!	Provides a corrective action for undesirable results. May include a fix and/or a work-a-round
How To?	Provides procedural info on how to do something specific
Information?	Provides general information, it may be about a product or product range.
Diagnostic	Describes a diagnostic process, how to run a diagnostic tool and/or the meaning of diagnostic results
Enhancement	Documents a product enhancement

Environment Information

Naming Products

Environment information should be formal and detailed, including as much information as necessary to uniquely identify the product/operating system being described.

- <Vendor> <Product>, version <Version Number>

Examples of Good Environment Information

- 3Com X1000 , version 03.02.01
- Windows XP Pro, SP1
- Siemens PLC S7
-

Environment Information helps Classify Problems

1. Do not put multiple Environments in a single statement.
2. Modify existing solutions to add new Environments as needed:
 - Windows 2000
 - MPI Protocol
 - Siemens PLC S7

Changes in the Environment

1. Think about what the user may have done:
 - Change: Installed the update to the software
 - Change: Reset the counter to zero
 2. Changes are not the cause - don't confuse the two.
 3. Do not jump to conclusions:
- No:** Change: *It worked before we replaced an XBTVA with an XBTVM*
- Yes:** Change: *Upgrade from Release 3.2 to 4.0*

The Most Reusable Error Structure

To structure error-statement-type Problems for the greatest opportunity for reuse, structure Problems by breaking them into two "modular" statements:

- a general statement that an error occurs, and the conditions during which it occurs
- the specific error statement, with no conditional modifiers

Example

If a caller reports getting the error "Post Office is now closed" when trying to send mail to other Post Offices, we suggest structuring the solution as:

Error: "Out of memory"

Error: "Error writing UDP packet 8101"

Error: "No document libraries available"

Status

Indicates progression of solution through its solution life cycle:

In Progress	The initial state of a solution, i.e., default state for a newly created solution; represents work in progress, no fix or resolution has been identified
Draft	A solution the author considers complete but they do not have high confidence in the resolution (not yet validated). Or, the author is a KCS I and not licensed to create <i>Approved</i> solutions. Draft solutions have limited visibility and can be validated through reuse.
Approved	A status assigned to a solution when a KCS II is confident in the resolution and the structure of the solution. KCS IIs and coaches can put solutions into the <i>Approved</i> state. <i>Approved</i> solutions have broad visibility in the system.
Published	A solution that is customer viewable
Obsolete	This solution is no longer relevant. It is a candidate for archiving

Resolution - Fixes and Answers

The resolution should address the problem or answer to the question

- The Fix statement clearly lists what steps to take to resolve the issue
- There can be multiple ways to resolve a problem, a formal fix or ways to workaround the situation, these can be documented in fix statements and should be labeled "workaround: "
- Fix statements should not include active hypertext links to uncontrolled Web sites
- Use a link when helpful to point to existing documents or more details.

Structure of a Fix Statement

- Keep the whole fix within one "statement". If several steps must be performed in order, number the steps.
- Use tabs for formatting and readability.
- Write everything as a present tense list of commands, as if you were reading them step by step to the customer.
- Do not include "if-then" statements in Fixes. This is an indication that you have two separate solutions.
- The solution may contain more than one fix statement - but all Fix statements must be applicable.

The Root-Cause (optional)

There should be only one cause per solution. If a solution has more than one cause, it is likely that it should be multiple solutions.

If you must decide between applying one fix statement or another (because only one will work for your customer), the solution should be split into two!

Add unique statements to differentiate solutions.