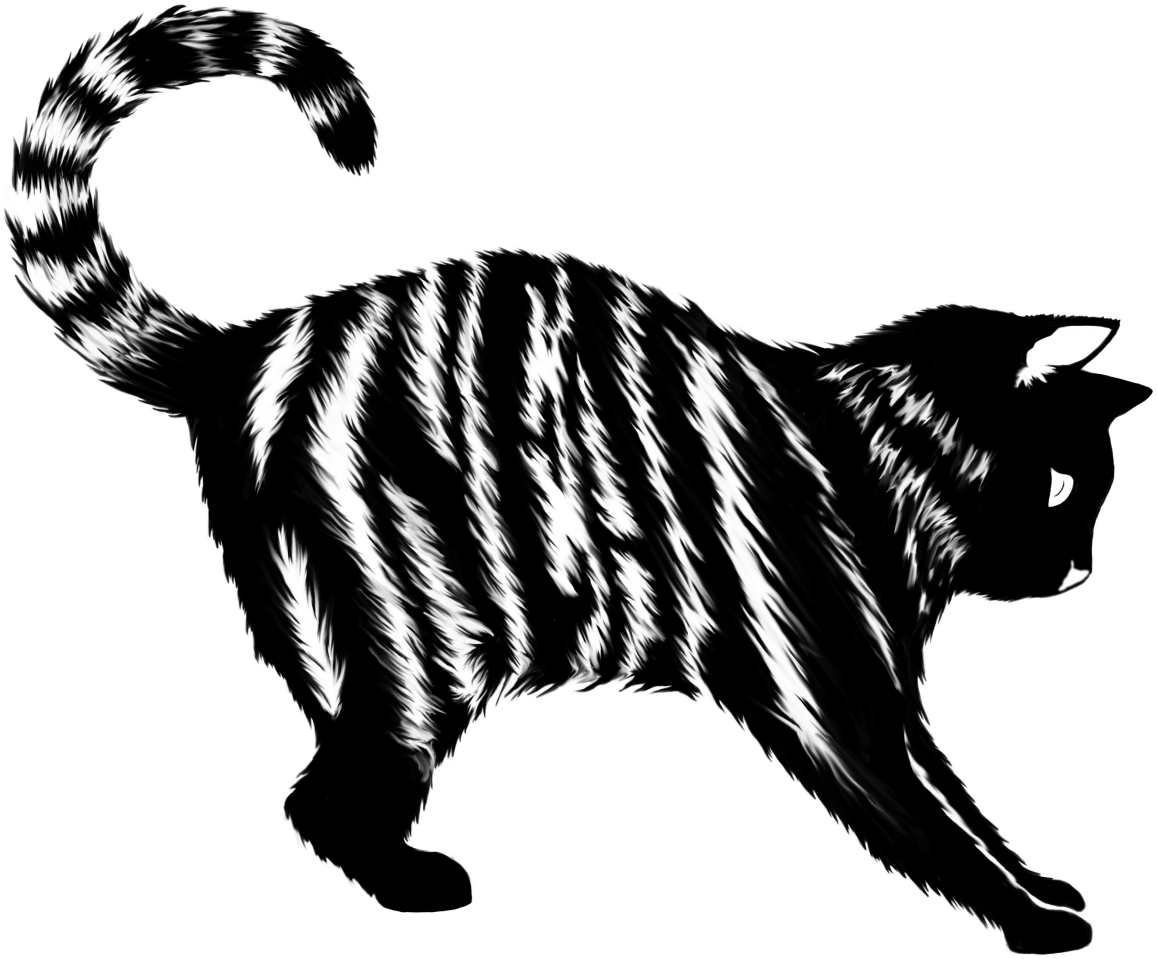
Kneadable Nimble Engineering Artifact Design



KNEAD-LaTeX-Manual

Revision: P7:10

Effective Date: 01 Mar 2022

Revision Date: 20 Apr, 2024

DCN: KNEAD-LaTeX-Manual-P7:10

Revision Date: 20 Apr, 2024

Prepared by:

Lewis Collier

DOCUMENT CHANGE HISTORY

The following table is a simple list of released revisions sent for review.

Change Record

Date	Version	Author(s)	Change Reference
20 Apr 2024	P7	Lewis Collier	Seventh Draft
02 Jan 2024	P6	Lewis Collier	Sixth Draft
12 Mar 2023	P5	Lewis Collier	Fifth Draft
28 May 2022	P4	Lewis Collier	Forth Draft
22 Apr 2022	P3	Lewis Collier	Third Draft
20 Dec 2021	P2	Lewis Collier	Second Draft
13 Dec 2021	P1	Lewis Collier	First Draft

Draft P7

1. Added *TeXnicCenter* loading and setup information in § A.

Draft P6

1. Added CUI information in § 3.3.

Draft P5

1. Added bibliography information in § 4.

Draft P4

1. Added appendix regarding Specifications, that includes newly defined specification macros.

Draft P3

1. Third draft version of this document for general review and discussion.

Draft P2

1. Second draft version of this document for general review and discussion.

Draft P1

1. Initial draft version of this document for general review and discussion.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

This chapter provides the purpose, scope, overview of the system, and a description of the contents of this document.

1.1 Purpose

The purpose of this manual is to discuss the Kneadable Nimble Engineering Artifact Design (**KNEAD**) methodology for designing systems and associated documentation. This manual itself was generated in the **KNEAD** methodology so it, and its underlying LaTeX source files, can be used as an example of how things work in **KNEAD**. There also are numerous examples of normal MIL-STD-498 artifacts to demonstrate the approach. This purpose of this document is to provide focused detailed on the mechanics of the **KNEAD** documentation itself.

This document also provides background as to why this methodology is called **KNEAD**. As with many computer-based systems, the name is whimsical yet with special meaning. The **KNEAD** project follows this paradigm.

The **KNEAD** project is so named because of the use of the design methodology itself. The methodology target is both the documentation process and the system that is being described by the **KNEAD** artifacts! Thus, **KNEAD** is an adjective acronym, whereby all the parts of the acronym describe the methodology.

The parts of the **KNEAD** moniker follow the Royal Order of Adjectives (`https://dictionary.cambridge.org/us/grammar/british-grammar/adjectives-order`). A quick summary of this naming process also discloses much about the goals of the **KNEAD** methodology itself. While this whimsical description helps the author rest easy, it is the underlying use that is important and not the exact adjective order relationships. And, of course, the double-entendre about kneading to form a product matches the intent of working and reworking the artifacts as living documentation of the system design. The reader is encouraged to concentrate on the flexible nature of the methodology and enjoy the name as it will be used to qualify all of the LaTeX commands within the methodology.

K stands for "Kneadable", which is is an opinion [order 1] that the resultant methodology is malleable but from a structure, much as the same dough recipe can be used to create multiple bread products. Likewise, the artifacts created within the **KNEAD** methodology also can be shaped to form a set of artifacts using the same ingredient set.

N stands for "Nimble", which is a physical quality [order 3] of the design process. Many practitioners of systems engineering and related development activities overload the word "agile" when they really mean "nimble" or a variant of this adjective. To avoid confusion with the software engineering agile manifesto, "nimble" is used to avoid confusion and to more truly reflect the adaptability and flexibility of this process.

E represents the "Engineering" aspect of the methodology, which is the origin [order 7] of the methodology. Normally this is considered to be a country when describing adjectives but engineering is the source of the methodology so this seems a logical placement for the adjective. Again, both the artifacts and the system are being engineering, but in the **KNEAD** moniker, it is the Artifacts that are being described by the adjectives.

A represents the artifact, which is the type of the methodology [order 9]. It often is said that if something is not recorded then it did not happen. This too is true when dealing with system design: if something is not written down then the resultant product cannot be held accountable to the thing. Artifacts are the key component of the **KNEAD** methodology. But by making them living tailorable documents, they can be used to guide and record the development process.

D represents the design, which is the purpose [order 10] of the methodology. The end goal of the **KNEAD** methodology is to create a design. The **KNEAD** artifacts direct the design process as well as record the resultant design. This two-for-one gain enables the **KNEAD** methodology to effectively and efficiently create better designs.

1.2 Scope

This manual documents the ins and outs of the **KNEAD** methodology of designing and documenting a system via LaTeX documentation. The contents of this manual are directed towards the technical aspects of the artifact generation and not the design process itself. Thus, following this manual will enable the reader to create a set of artifacts that capture the key elements of the overall **KNEAD** design process.

1.3 System Overview

KNEAD is a hierarchical system for generating a consistent set of documentation for a project. This layout is shown in Figure 1. The project hierarchy consists of three (3) levels:

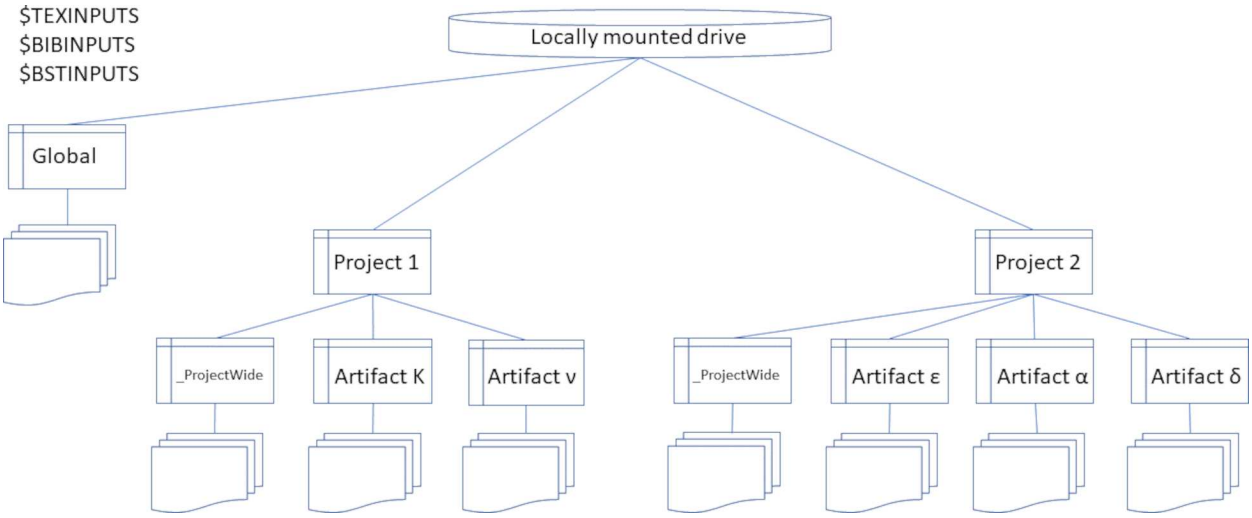


Figure 1: Overview of KNEAD Hierarchy

Global defines **KNEAD** items that are used across all projects. This includes definitions such as global LaTeX commands, acronyms, glossary definitions, and bibliography references. These items might include things such as company identifiers (e.g., name and logo), industry-wide references, and other common things used by all systems created by an agency.

Project defines LaTeX commands, acronyms, glossary definitions, and bibliography references that are reused across a given project. These items might include things such as the project name, distributions statements, and maybe customer information. A project is expected to generate multiple documents, so commonality and re-usability is a key tenet to nimble creation of engineering artifacts.

Document defines LaTeX commands, acronyms, glossary definitions, and bibliography references that are only used within a given artifact. This would, of course, include things such as the title, author, and document identification that varies from artifact to artifact.

By segregating features into these levels, reuse and commonality across the enterprise and project are enabled.

The hierarchy guides the file structure as well. All folders are expected to be located on a drive accessible to the machine locally. The LaTeX system cannot follow the URLs of cloud drives. Thus, if SharePoint™ is used, the files must be synchronized to the local machine. The global files are contained in a commonly available folder with read permissions to all and write permissions to LaTeX administrators. Each project is expected to have a single top-level folder that houses all **KNEAD** artifacts for the project. A folder called _**ProjectWide** is used to house the files common across all the project artifacts. Each artifact has its own folder that contains sub-folders and the files for the items specific to that artifact. In this manner, artifact-specific content is separable from the project-wide content that also is separable from the global content. Reuse is highly supported but not at the expense of customization.

KNEAD defines all items needed for artifact generation, but allows for customization as needed. For example, a title page layout is provided for quick start-up, but the format is separated into its own file so that a custom title page format can be realized at the global, project, or artifact level. Likewise, a workable style is defined, but this too can be altered should a different layout be desired. The goal of **KNEAD** is to provide everything that is needed to capture a design and generate a proper set of documentation, but to allow for customization as needed. This allows engineers to focus on design aspects rather than fighting traditional office suite tools to wrangle organizational presentation and formatting issues.

This "quick start" mentality includes providing a set of templates for all of the standard MIL-STD-498 [1] document templates. This manual, however, does not include information about the various artifact formats.

1.4 KNEAD Manual Overview

This section is ...TBD.... See the Table of Contents for now.

CHAPTER 2

References

This section provides a list of referenced items for this document.

2.1 Acronyms and Abbreviations

This section defines acronyms and abbreviations used in this and related documents.

Table 1: Acronym Definitions

Acronym	Definition
SPS	System Performance Specification
SSDD	System / Sub-system Design Description
SSS	System / Sub-system Specification
STD	System Test Description
SRS	Software Requirements Specification
STS	System Test Specification
STP	System Test Plan
STR	System Test Report
SUM	System Usage Manual
End of acronym definition table	

2.2 Glossary and Definitions

This section defines glossary terms used in this and related documents.

Table 2: Glossary Terms and Definitions

Glossary Term	Definition
Stakeholder	Any member of the project that has an interest in the project.
End of glossary terms table	

2.3 Referenced Documents

This section lists the referenced documents for this document. The references are categorized into two categories:

External Documents not directly associated with this project.

Project Documents that are directly associated with this project.

2.3.1 External Documents

[1] MIL-STD-498. *Military Standard Software Development and Documentation*. Dec. 31, 1994.

[2] Lewis Collier. *KNEAD Manual Pages*. The KNEAD Project. Mar. 20, 2022.

[3] DI-IPSC-81431. *Data Item Description for System Segmentation Specification (SSS)*. Dec. 31, 1994.

2.3.2 Project Specific Documents

[4] Lewis Collier. *Non-existent CDD for Example Specification Appendix System*. Specification. The KNEAD Project, May 28, 2022.

CHAPTER 3

Document Parts

This chapter describes the parts of the **KNEAD** document. These parts are stored in separate files so understanding the files and their contents is key to successful use of the **KNEAD** methodology. Each section describes a set of LaTeXcommands that the user can alter to easily change the content and format of the document.

3.1 Main File

This section defines the main file of a **KNEAD** artifact. The goal is to segregate items into separate files so that each file is limited in what it affects. Thus, the main file defines a minimal set of items, mainly by including other files that define information for the document itself. The reader is encouraged to follow along in an example file to see how these features function.

3.1.1 Class Options

The main file invokes the LaTeX file that defines the **KNEAD** style and passes options to that *.sty* file. This section describes those **KNEAD** class options. These are shown on the first non-comment line of the main file, as shown below, where the options are listed within the square brackets "[" and "]" while the **KNEAD** style file is at the end within the braces "{" and "}".

```
\documentclass[showtpg,...,makesinglespace]{KNEADdocument}
```

The defined **KNEAD** options are:

showtpg Show the title page. Omit this option when a simple document with no title page, table of contents, etc. is desired.

showtoc Show the table of contents. Omit this option when a simple document with no title page, table of contents, etc. is desired.

showlof Show the list of figures. Omit this option when a simple document with no title page, table of contents, etc. is desired.

showlot Show the list of tables. Omit this option when a simple document with no title page, table of contents, etc. is desired.

showlos Show the list of specifications. This is used for specification documents that contain specifications (requirements).

showlop Show the list of procedures. This is used for procedural documents that contain procedures (e.g. test procedures).

showlol Show the list of checklists. This is used for check list documents that contain operational checklists (e.g. check list documents).

makesinglespace Make the document single space. Default is double spacing, which is good for providing space for review comments.

makeoneptfivespace Make the document spacing a line and a half. Default is double spacing, which is good for providing space for review comments.

3.1.2 Definition Files

The first sub-section of the main file includes all the files needed to define commands and include packages for the current document. This is done as shown below.

```
%%% include acronyms and other definitions
% get KNEAD_GlobalDefs.tex from $TEXINPUTS folder
\input{KNEAD_GlobalDefs.tex}
% from project's zProjectWide common folder
\input{../_ProjectWide/KNEAD_ProjectDefs.tex}
% from local folder for this artifact
\input{./KNEAD_ArtifactDefs.tex}

%%% include stuff used in artifact layouts
% get KNEAD_GlobalFormatting.tex from $TEXINPUTS folder
\input{KNEAD_GlobalFormatting.tex}
% from project's zProjectWide common folder
\input{../_ProjectWide/KNEAD_ProjectFormatting.tex}
% from local folder for this artifact
\input{./KNEAD_ArtifactFormatting.tex}



%%% include stuff for multi-part bibliographies using Biber
% get KNEAD_GlobalBiber.tex from $TEXINPUTS folder
\input{KNEAD_GlobalBiber.tex}
% from project's zProjectWide common folder
\input{../_ProjectWide/KNEAD_ProjectBiber.tex}
% from local folder for this artifact
% this is rarely used; maybe for seprated appendicies?
```

```
\input{./KNEAD_ArtifactBiber.tex}
```

The defined **KNEAD** files to include are:

Defs contains accessible commands and definitions. This is where commands and variables are defined to create acronyms and similar items. The Global defines globally needed items that are expected to be overridden in Project and Artifact level file, such as document number and revision level. The global file also defines a set of well-known acronyms (see the file to get a complete list) while the project Defs file provides acronyms and such that need to be consistent across a project.

Formatting contains formatting definitions that cannot be defined in the *KNEAD.cls* file. This allows for a global set of definitions to be created and then to be overridden in the project and artifact levels. See the files themselves for a description of the formatting aides provided.

Biber contains information needed to support the split bibliography. As with the other support file types, this is divided into 3 areas but could be extended as needed.

Examples of extension possibilities should help illuminate why there are levels. One example is for QMS artifacts that might have levels of "Plan" instead of "Global", "Process" instead of "Project", "Activity" instead of "Artifact", and a 4th level of "Forms" that has no equivalence in typical **KNEAD** documentation. Or, a new top-level "Enterprise" level could be added that encompasses several "Global" files as divisions or sectors within an agency or company. By splitting the files into levels, reuse is maximized without loss of extendability.

3.1.3 Source Control Information

KNEAD supports both Subversion™ and Git™. In both cases, packages are invoked within the **KNEAD** style file to support the CM capabilities. The goal for all CM is to glean the CM information such as version number so that it can be displayed in processed files. Having this information allows for traditional CM techniques to be followed as are done for software.

3.1.3.1 Subversion

For the Subversion™ tool, the \svnidlong[5] command is automatically populated by Subversion™ when a file is committed.

```
\svnidlong
{$HeadURL: https://192.168.1.5/svn/...$}
{$LastChangedDate: 2024-01-02 22:16:24 -0500 (Tue, 02 Jan 2024) $}
```

```
{$LastChangedRevision: 6 $}
{$LastChangedBy: KneadProject $}
\svnid{$Id: KNEADmanual_Section_MainFile.tex 6 2024-01-03 03:16:24Z KneadProject $}
```

3.1.3.2 Git

This section is ...TBD....

3.1.4 Document Body

The main part of the document is described as shown below.

```
\begin{document}

% from $TEXINPUTS folder, copy locally and roll your own if needed
\KNEADtitlepage{KNEAD_DefaultTitlePage.tex}

% from $TEXINPUTS folder, copy to project or to each artifact
% to customize as needed
\dochistory{KNEADmanual_doc_approval.tex}

% make local copy from template for each artifact
\dochistory{./KNEADmanual_doc_history.tex}

%chapters from local artifact folder
% use common items for all artifacts such as System Overview from project folder
\newchapter{./KNEADmanual_Chapter_Intro.tex}
%same for all documents, but local common copy for placement of .aux files
\newchapter{./KNEADmanual_Chapter_Refs.tex}
\newchapter{./KNEADmanual_Chapter_DocumentParts.tex}

%% Include any appendices.
\newappendix{./KNEADmanual_DummyAppendix.tex}

%% Output the Index as the last item in the document.
\KNEADPrintTheIndexHere

\end{document}
```

The main part of the file only contains a minimal set of commands to reduce clutter and to promote reuse. All local definitions are contained, of course, in the `./KNEAD_ArtifactDefs.tex` file. The parts of the main body are:

Title Page The `\KNEADtitlepage` variable is used to define the name of the file used to format the title page. In general, the default title page format is used with the

`KNEAD_DefaultTitlePage.tex` file name. This file, which is found in the folder referred to by the `$TEXINPUTS` environment variable, can be copied to the project common folder, or to the local document folder, should a more customized title page be needed.

Document Approval The `\KNEADmanual_doc_approval` variable is used to define the file that contains the approval names and signature lines for the given document. This file is expected to live within the project document folder as each document will most likely have its own document approval chain. Obviously, all documents within a project should have similarly formatted approval pages, with names changed as necessary for each document.

Document History The `\KNEADmanual_doc_history` variable is used to define the file that contains the change history for the given document. This file is expected to live within the local document folder as each document should have its own document history. Obviously, all documents within a project should have similarly formatted change history files.

Chapter The `\newchapter` command is used to enter each chapter from a separate file. These files are expected to reside in the document folder. The Chapter 2 file, however, just points to a common file that is found in the folder referred to by the `$TEXINPUTS` environment variable. This file is defined in § 3.

Appendix The `\newappendix` command is used to enter each a from a separate file. These files are expected to reside in the document folder.

Index The `\KNEADPrintTheIndexHere` command is used to inject the index into the overall document. This command is defined to provide all the commands needed to inject the index and add it to the table of contents. This line may be commented out if no index is needed or desired in the current document.

3.2 Header Footer Items

This section describes the **KNEAD** header and footer items items. These variables are used to define the content of the headers and footer. *ALL* pages receive the same header and footer, with the exception of page number style.

The phantom

hspace commands are used to center the content with respect to the other line. The phantom

space in the center footer is needed to prevent error on title pg when page number is gobbled (not shown) on the title page. These values are set manually, by trial and error, since these variables do not use fixed dimension regions in the header and footer area.

```
\newcommand{\KNEADtopLeftHeading}
{\phantom{\hspace{8.0pt}}KNEAD TOP\\
LEFT HEADER\phantom{\hspace{0.0pt}}}

\newcommand{\KNEADtopCenterHeading}
{KNEAD TOP\phantom{\hspace{0.0pt}}\\
CENTER HEADER\phantom{\hspace{0.0pt}}}

\newcommand{\KNEADtopRightHeading}
{KNEAD TOP\phantom{\hspace{10.0pt}}\\
RIGHT HEADER\phantom{\hspace{0.0pt}}}

\newcommand{\KNEADbtmLeftHeading}
{\phantom{\hspace{8.0pt}}KNEAD BTM\\
LEFT FOOTER\phantom{\hspace{0.0pt}}}

\newcommand{\KNEADbtmCenterHeading}
{\thepage\phantom{\hspace{0.0pt}}\\
KNEAD BTM CENTER}

\newcommand{\KNEADbtmRightHeading}
{\phantom{\hspace{8.0pt}}KNEAD BTM\\
RIGHT FOOTER\phantom{\hspace{0.0pt}}}
```

A set of pre-defined headers are provided for traditional MIL-STD-498 [1] documents. These provide a two-line description of the artifact as shown.

```
\newcommand{\KNEADHDRSDP}
{System\phantom{\hspace{30.0pt}}\\
Development Plan\phantom{\hspace{0.0pt}}}

\newcommand{\KNEADRHDROCD}
{Operational Concept\phantom{\hspace{0.0pt}}\\
Description\phantom{\hspace{22pt}}}

\newcommand{\KNEADRHDRSPS}
{System Performance\phantom{\hspace{0.0pt}}\\
Specification\phantom{\hspace{18.0pt}}}

\newcommand{\KNEADRHDRSUM}
```

```
{System\phantom{\hspace{16.0pt}}\\
Usage Manual\phantom{\hspace{0.0pt}}}

\newcommand{\KNEADRHDRSSS}
{System / Subsystem\phantom{\hspace{0.0pt}}\\
Specification\phantom{\hspace{20.0pt}}}

\newcommand{\KNEADRHDRSSDD}
{System / Subsystem\phantom{\hspace{0.0pt}}\\
Design Description\phantom{\hspace{4.0pt}}}

\newcommand{\KNEADRHDRSRS}
{Software\phantom{\hspace{0.0pt}}\\
Test Plan\phantom{\hspace{0pt}}}

\newcommand{\KNEADRHDRHRS}
{Hardware\phantom{\hspace{0.0pt}}\\
Test Plan\phantom{\hspace{0pt}}}

\newcommand{\KNEADRHDRSTP}
{System\phantom{\hspace{6.0pt}}\\
Test Plan\phantom{\hspace{0pt}}}

\newcommand{\KNEADRHDRETP}
{Engineering\phantom{\hspace{0.0pt}}\\
Test Plan\phantom{\hspace{6pt}}}

\newcommand{\KNEADRHDREITP}
{Engineering and\phantom{\hspace{15.0pt}}\\
Integration Test Plan\phantom{\hspace{0pt}}}

\newcommand{\KNEADRHDRSTD}
{System\phantom{\hspace{27.0pt}}\\
Test Specification\phantom{\hspace{0pt}}}

\newcommand{\KNEADRHDRSTS}
{System\phantom{\hspace{27.0pt}}\\
Test Specification\phantom{\hspace{0pt}}}

\newcommand{\KNEADRHDRATP}
{System Acceptance\phantom{\hspace{0pt}}\\
Test Procedure\phantom{\hspace{9.5pt}}}

\newcommand{\KNEADRHDRSTR}
{System\phantom{\hspace{12.0pt}}\\
```

```
Test Report\phantom{\hspace{0pt}}}
```

3.3 Title Page Items

This section describes the **KNEAD** title page items.

The first items that are defined in the "KNEAD_GlobalDefs.tex" file set the CUI status of the artifact. These variables define the CUI status and define the information needed in the CUI front page information block is displayed on the title page. These variable can be renewed in the project or artifact definition file as needed. The CUI variables are

KNEADcuiStatus is used to designate the document as CUI or not. By setting this value to be "CUI", the center header and footer values are set to the value (e.g., CUI, or CUI//CTI, etc.).

KNEADcuiCOMPONENT

KNEADcuiOFFICE

KNEADcuiCATEGORIES

KNEADcuiDISTRIBUTION

KNEADcuiPOC

KNEADcuiTitlepageDistributionStatement

See `https://www.dodcui.mil/Home/Training/` for more information on the front page CUI dissemination block parts.

KNEADtitlepageLogoFileEPS

KNEADtitlepageLogoWidth

KNEADtitlepageTitle

KNEADtitlepageCompanyHeader

KNEADtitlepageDistributionStatement

CHAPTER 4

Generating Bibliographies

This chapter describes the methodology for generating bibliographies in **KNEAD** documents. The best way to understand this processing is to look at the bibliography section of Chapter 2 of this document. The following discussion explains what is happening in this file.

4.1 Overview

The basic idea behind how bibliographies are generated in **KNEAD** is based upon the use of *BibLaTeX/Biber* and categories for bibliography entries. The BibTeX entries are used, with a `keywords` entry, to drive the *BibLaTeX/Biber* processing. The keyword is then used to allow for segregated bibliographies based on the keywords. A traditional use is for general external references to be tagged with a keyword of "KNEAD_BiberGlobal" and for agency-specific and project-specific keywords to be used as applicable to provide for project or other delineations. The bibliography then uses multiple sections, each calling out a different keyword, to separate the bibliography entries as desired. See § 2.3 for an example.

An example for such a separation delineation can be found in quality management system documentation. Such systems generally provide artifacts in categories such as manuals, playbooks, guides, procedures, forms, etc. The **KNEAD** bibliography paradigm allows for each type of artifact to be tagged so that they can be separated as needed.

4.2 BIB File

The **KNEAD** bibliography processing begins in the bibliography files, known as ".bib" files. For each bibliography entry, the `keywords` entry is provided.

```
@report{ ref__KNEAD_Manual,
  author      = {Lewis Collier},
  title       = {KNEAD Manual Pages},
  type        = {manpages},
  institution = {The KNEAD Project},
  date        = {2022-03-20},
  keywords    = {KNEAD_BiberArtifact, QMS_Guide},
}
```

Note that this allows for more than a single keyword to be included. A comma-separated list provides for multiple delineations for each bibliography entry. See any of the myriad online BibTeX documentation for a full description of all applicable entry fields.

4.3 BIBER File

As noted in § 3.1.2, there are three (3) "Biber" files.

KNEAD_GlobalBiber.tex contains global *BibLaTeX/Biber* information (references used across multiple projects)

KNEAD_ProjectBiber.tex contains project-specific *BibLaTeX/Biber* information (references specific to a given project)

KNEAD_ArtifactBiber.tex contains artifact-specific *BibLaTeX/Biber* information (references only used within the given artifact)

The global file is similar to the following:

```
%%%
%%% stuff for bibliography in multiple parts
%%%
%defernumbers=true,backend=biber,
%\usepackage[backend=biber,natbib=true,style=numeric,sorting=none]{biblatex}
\usepackage[backend=biber,natbib=true,style=numeric,sorting=none]{biblatex}

\addbibresource{KNEAD_GlobalReferences.bib}

\newcommand{\citePMBOK}
{\citeauthor{ref__PMI_PMBOK},
  \citetitle{ref__PMI_PMBOK}~\cite{ref__PMI_PMBOK}\xspace}
```

The `usepackage` is the standard inclusion for the `biblatex` package, here configured to use *BibLaTeX/Biber*. The `addbibresource` command adds the global ".bib" file to the list of files from which the bibliography entries can be pulled. The project and artifact level files are similar, but need not include the `usepackage` command. The above lines are needed for general correct operation of the *BibLaTeX/Biber* processing.

The **KNEAD** paradigm adds the concept of providing a citation command for every reference. While not absolutely necessary, this provides a common framework for citing all references. This paradigm also provides for a common placement for all items for a reference. By using the `citeXYZ` command, the attributes from the ".bib" file for the references. As is seen, the author, title, and other entry fields can be referenced by just the `citeXYZ` commands. This allows for citation text to read in a customizable way. For example, this document can be cited as *KNEAD Manual Pages*, "The **KNEAD** Project", Collier [2]. The following code provides this capability.

```
\newcommand{\citeKneadLatexManual}
{\citetitle{ref__KNEAD_Manual}, ``The \KNEAD Project'',
\citeauthor{ref__KNEAD_Manual}~\cite{ref__KNEAD_Manual}\xspace}
```

Note that text can be inserted as desired to supply additional information that is not supported by the limited number of `citeXYZ` commands. The supported `citeXYZ` commands are:

citeauthor references the "author" entry

citetitle references the "title" entry

citeyear references the "year" entry

citedate references the "date" entry

citeurl references the "url" entry

4.4 Reference File

```
\printbibliography[heading=none,keyword=KNEAD_BiberProject]
```

APPENDIX A

Loading LaTeX on MS-Windows

This section describes how to load the TeXnicCenter and Miktex processing system on Windows™. The basic steps, which are detailed below, are:

LaTeX Inputs set up base template files.

MikTex the underlying implementation of LaTeXprocessing system.

GhostScript converts PostScript™ (.PS) files to .PDF files.

Adobe Reader displays .PDF files.

TeXnicCenter the integrated development environment (IDE).

Environment Variables path variables.

Test to make sure it all works.

A.1 LaTeX Inputs

A major step in setting up the **KNEAD** system is to set up a project structure. The overall **KNEAD** philosophy provides for three levels folder structures: global, project, and artifact. All the files for a given artifact are housed within a single folder and sub-folders to organize those contents. All such artifact folders, along with project-wide files, are contained within a single project folder. The global folder is the subject of this section. This folder contains all the style definition files and example artifacts from which new projects and artifacts can be derived.

This step makes a copy of the basic **KNEAD** templates that support the **KNEAD** process for artifact generation. These can go anywhere on your local computer, or can be on a common SharePoint™ site (preferred), since each computer user has environment variables that point to the location.

While the actual working folders for LaTeXfiles can be on a SharePoint™ site, they should not be. LaTeXutilizes temporary axillary (.AUX) files that update with each run of LaTeX. As such, they mainly clog up the SharePoint™ and OneDrive™ synchronization and for no good reason. They can be set to ignore by the SharePoint™ administrator, but this is done on a large scale. This means that should there be any files from other applications that need to use similar file extensions, these may not be properly archived by SharePoint™. Thus, a better mechanism is preferred for archival and configuration management of these files.

A source control system such as SVN or GIT is a better way to manage these files. Since the LaTeXfiles are text-based, these tools work for LaTeXdocumentation much the same as they o for general programming source code. SVN is preferred over GIT since the SVN server can be tied into underlying Windows™ authentication, thus eliminating the need for a second set of user profiles. Either system, or other source control mechanisms can be used, although SVN and GIT both have direct support in LaTeXfor gleaning repository information so it can be presented within the document for full visibility into the source control status.

A.2 Load MikTex

Install *MikTex* by running the installer program as noted within the installer program. Having *MikTex* installed first allows the IDE to recognize it and auto-configure some items based on the *MikTex* installation.

NOTE: Make sure that the *normal user account* is used for this and NOT an account with Administrator privileges. Installation with an account with Administrator rights can cause the installer, and *MikTex* Console (the maintenance program), to get confused about where to install things.

NOTE: When selecting *MikTex* make sure you configure to *LOAD FOR JUST the owner user*. This is an important STEP. This option will be a drop-down menu which you need to select this option (load for just the logged in user is the default option).

This is needed due to issues with keeping packages for a local user copy and the Administrator copy in sync. If the *MikTex* is loaded for ALL users then it gets installed in

```
C:\Program Files
```

, which normal users do not have access. Updates can then get the two folders out of synchronization, which means getting the IT group involved to install updates. Keeping it all in the local user space allows the user to get updates to LaTeX packages as needed.

Once *MikTex* is installed, select the following options:

Paper set up letter sized, (not A4, which is the default).

On-The-Fly set to YES so *MikTex* installs new packages when it finds them (this is why *MikTex* needs to be in a user accessible space).

After installing, run [StartMenu]MikTeK 2.9 -¿ MikTeK Console to check for updates. By default, the console starts in user mode, even if run from an Admin account (see note above about installing from an account with NO admin rights!). Select “Restart As User” so it will do the update in the non-privileged account.

A.3 Load GhostScript

Install GhostScript by running the installer program. NOTE: This seems to require an account with Administrator privileges. Generate cidfmap for windows CJIK TT fonts (I think this is just a check box or drop-down menu should say something about cidfmap). If steps are not mentioned here specifically in the install steps just keep clicking next.

A.4 Load Adobe Reader

Load adobe reader if not already installed. *MAKE SURE to UNCHECK the McAfee stuff in the download box BEFORE doing the download. The type and version of Adobe™ Reader will be needed later so check this and record so the* TeXnicCenter *settings can be set correctly.*

A.5 Load TeXnicCenter

Load TeXnicCenter*. Just click next in the steps, don't worry about fields if it's not specifically mentioned.*

TeXnic Center BUILD PARAMETERS SETUP The following items will be in the Build Tab (on menu bar) and in the Define output profiles (alt + f7). This is directly copied from the how to load latex.txt file without changing format. It seems like this stuff is the default, but if its not the same then make changes as necessary.

A.5.1 (La)Tex Tab

Check [v] Run (La)Tex in this profile

Do not check [] Stop compilation on LaTeX error
Path to the (La)Tex :

```
C:\Users\UserName\AppData\Local\Programs\MiKTeX\miktex\bin\x64\latex.exe}
```

Command line arguments : -max-print-line=254 -interaction=nonstopmode "%wm"

Do not check check [] Do not use BibTex in this profile
Path to BibTex :

```
C:\Users\UserName\AppData\Local\Programs\MiKTeX\miktex\bin\x64\bibtex.exe
```

Command line arguments : "%tm"

Do not check [] Do not use Makeindex in this profile
Path to MakeIndex:

```
C:\Users\UserName\AppData\Local\Programs\MiKTeX\miktex\bin\x64\makeindex.exe
```

Command line arguments : "%tm.idx" -r -t "%tm.ilg" -o "%tm.ind"

-r puts every index entry separately, i.e. 1,2,3 not 1-3 (useful for finding every page easily)

KNEAD *uses "-r option" for* –CHECK-ME– *and ...TBD... and ChangeBar macros to help locate pages with issues*

A.5.2 (Postprocessor Tab

Select Dvi2PS and set the executable path to the same location as other programs, e.g.

```
C:\Users\UserName\AppData\Local\Programs\MiKTeX\miktex\bin\x64\dvips.exe
```

Set arguments to -P pdf "%Bm.dvi"

Select ps2pdf *and set it up for GhostScript™*
Point executable to

```
C:\Program Files\gs\gs9.23\bin\gswin64.exe
```

or similar as defined by the version of GhostScript that is loaded. Use this version to get GS errors in the log file instead of its own window.

Point Arguments to (YOU WILL HAVE TO CHANGE THIS SETTING) but it all goes on one line.

```
-sPAPERSIZE=letter -dSAFER -dBATCH -dNOPAUSE -sDEVICE=pdfwrite
-sOutputFile=‘‘\%bm.pdf’’ -c save pop -f ‘‘\%bm.ps’’
```

Copy and paste the above lines to save typing.

A.5.3 (Viewer Tab

For Adobe reader, the executable path should be set to

```
C:\Program Files\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe
```

or similar based on your installation location.
For Acrobat, the executable path should be set to

```
C:\Program Files\Adobe\Acrobat 2017\Acrobat\Acrobat.exe
```

Click the button and find your AcroRd32.exe or Acrobat.exe file. Set up the following DDE commands so that TeXnicCenter *can send commands to the Adobe™ viewer you are using.*

Select DDE command for each one View projects output

[DocOpen("%bm.pdf")][FileOpen("%bm.pdf")]

Forward search

[DocOpen("%bm.pdf")][FileOpen("%bm.pdf")]

Close document

[DocClose("%bm.pdf")]

Each section

Topic field = control

Server field

The following options for the server option

acroviewR20 acroviewA17

Note: If your build and view output opens up the .PDF viewer and then does not load, this step is the most likely the problem. The solution is to change the server option to one of the 3.

A.5.4 (BIBER Profile

Setup a BIBER profile since **KNEAD** *uses this to handle split bibliographies.*

Copy above LaTeX-¿PS-¿PDF to (Biber) LaTeX-¿PS-¿PDF

[put biber first so it shows up in the skinny dropdown]
Make BibTeX path say

```
C:\Program Files\MiKTeX 2.9\miktex\bin\x64\biber.exe
```

A.6 Set Environment Variables

You may need system administration rights for this step, depending on your computer's configuration.

Search and open up Control Panel

Enter "environment" in the find a setting entry box

Select "Edit environment variables for your account"

Add/change the following environment variables in the field "User variables for username"

Variables
TEXINPUTS – Variable points to the

```
_LatexInputs
```

folder on your computer
Example:

```
C:\(your local folder)\_LatexInputs
```

BSTINPUTS – Variable points to the

```
_LatexInputs
```

folder on your computer
Example:

`C:\(your local folder)\_LatexInputs`

BIBINPUTS – Variable points to the

`_LatexInputs`

folder folder on your computer
Example:

`C:\(your local folder)\_LatexInputs`

`MIKTEX_ALLOWUNSAFEOUTPUTFILES`

– Variable needed for

Just enter 1 as the ENV variable value.

See https://tex.stackexchange.com/questions/12417/texify-and-output-directory for more details.

A.7 Test

APPENDIX B

Specification Macros

This section describes the macros provided for use in writing specification documents. These could be a System Performance Specification (SPS) or a System / Subsystem Specification (SSS) artifact, both of which generally follow the outline provided in the SSS Data Item Description (DID) [3]. These macros provide a table to encapsulate all features of a given specification in a single location. This differs from formatting used by many that splits the specification, verification expectations, and traceability into separate document sections. This approach also provides macros for single-part and multi-part specifications. Finally, these macros provide an argument that allows for a change-bar to be displayed when the contents of the specification are updated.

The table formats are described below. The first format expects just a single specification statement. This would be used for an "atomic" specification as shown where the overall identifier is provided in the table header. The second format provides support for multi-part specifications. These would be used to group tightly-coupled requirements such as those needed to fully specify a display's resolution, pixel size, brightness, contrasts, and such. Having these related requirements separated would not be prudent, thus the multi-part format macro. These descriptions illustrate the content that is expected for each argument of the macro. See the file `KNEAD_SpecificationCommands.tex` *for full details.*

B.1 General Description

The system specifications are listed in a common table format as shown below. There are two different macros that provide these capabilities: `ONERQMTV[9]` *and* `\MULTIRQMTV[10]`*. The* `\ONERQMTV[9]` *macro is defined this way so it takes nine (9) arguments. The* `\MULTIRQMTV[10]` *macro is defined as* `\MULTIRQMTV[1]` *since it saves the first argument than passes the remaining 9 into* `\MULTIRQMTVbase[9]`*. The* `\MULTIRQMTV` *macro is defined this way since LaTeX only allows 9 arguments in a macro. See the file* `KNEAD_SpecificationCommands.tex` *for full details.*

1. *The first row of a table provides a unique number and a title for the requirement or expectation. This row is generated from the first 3 arguments to the macro.*
2. *The second row of the table provides the specification text of the requirement. For a single specification, this is a sentence with a single testable requirement (shall) state-*

ment. For a multi-part specification, this is a sentence that identifies the requirement and associates it with the multiple parts listed in the next row.

3. *The next row, which is only included for multi-part specifications, provides the list of independent but tightly-knit requirements.*
4. *The next row of the table provides the status for the specification listed in the table. This includes the applicable phases or release versions in which the required feature is supported, where* $S \in \{$*(T)hreshold, (O)bjective, (I)nactive, (D)eleted*$\}$.
5. *The next row of the table provides the acceptance criteria. This row follows the form of "This requirement shall be verified by* $V \in \{$*inspection, demonstration, test, analysis*$\}$*."*
6. *The next row of the table provides the traceability of the requirement. The traceability connects the requirement to a higher level document that calls out the need for a requirement. The structure of traceability is expected to be of the form*
   `\item [number]` *Section in Document-Name*`~cite{ref__CitedDocument}`*.*
   where the source is expected to be listed in the reference documents section. This style allows for the traces to be sorted by an external program that sorts first by the right-hand-side of the list item and then by the original requirement number. This traceability provides an easily readable list for humans to verify that all requirements have been captured.
7. *The next row of the table provides, if applicable, notes for the specification. Notes are not a formal part of the requirement or expectation but provide supporting information regarding the feature. This support may be information such as citing a meeting where a decision was made about the requirement to details about the requirement that should not be included in the specification, but are helpful in conveying the intent of the requirement. These can be omitted by setting the variable* `\setboolean{boKNEADshowSpecNotes}{false}` *in the* `KNEAD_ArtifactFormatting.tex` *file.*
8. *The final row of the table is used to define the artifact version in which the specification was last changed. This lets the system automatically place a "change-bar" next to the entire table for easier review of new/changed items.*

B.2 ONERQMTV Example

The below code provides an example of how to use the `\ONERQMTV[9]` *macro. See the comments in the macro for more definition about what is happening.*

```
%%% \ONERQMTV[9] is macro for consistently formatted requirements
```

```
%%% See KNEAD_SpecificationCommands.tex for details
\ONERQMTV
% #1 is requirement Number
{\RqtNumberBase.1}%\RqtNumberBase is set in a higher-level file,
                  % may include section numbers for consistency.
% #2 is Title
{Omni-Directional Antenna}
% #3 is requirement label (expected to be of form rqt:XXX)
{rqt:Directional Antenna}
% #4 is Text
{The SYSTEM shall provide communications to an audience in a
omni-directional pattern from the transmitting source.}
% #5 is Status (S in {(T), (O), (I), (D)} listed by phase as \item [] S)
{
\item [Phase 1] Threshold
}
% #6 is Acceptance
{This requirement shall be verified by demonstration.}
% #7 is Traceability
{
\item [5.1.2] Section in CDD for SYSTEM~\cite{ref__SYSTEM_CDD}.
}
% #8 is Notes; listed as enumeration \item ...
{
\item This requirement is provided to guide selection of the power
      output of the SYSTEM.
\item It is well known that the range is limited more by the antenna height
      and power of the target device than the transmitter power.
        This is meant to guide overall tradeoff analysis between \SWaP,
amplifier power, and antenna beam width
rather than to be a ``hard requirement''.
\item A final tradeoff will be needed between power (which drives \SWaP)
      and obtainable range to determine a final solution.
}
% #9 is version in which this specification was last changed;
%    to auto generate change bars
{P1}
%%%%% end \ONERQMTV[9] macro
```

This formats into a table that looks like:

Specification 1.1.1.1 Omni-Directional Antenna	
Text	The SYSTEM shall provide communications to an audience in a omni-directional pattern from the transmitting source.
Status	**Phase 1** Threshold
Acceptance	This requirement shall be verified by demonstration.
Traceability	**5.1.2** Section in CDD for SYSTEM [4].
Notes	1. This requirement is provided to guide selection of the power output of the SYSTEM. 2. It is well known that the range is limited more by the antenna height and power of the target device than the transmitter power. This is meant to guide overall tradeoff analysis between SWaP, power, and antenna beam width rather than to be a "hard requirement". 3. A final tradeoff will be needed between power (which drives SWaP) and obtainable range to determine a final solution.

B.3 MULTIRQMTV Example

The below code provides an example of how to use the `\MULTIRQMTV[10]` *macro. See the comments in the macro for more definition about what is happening.*

```
%%% \MULTIRQMTV[10] is macro for consistently formatted requirements
%%% See KNEAD_SpecificationCommands.tex for details
\MULTIRQMTV
% #1 is requirement Number
{\RqtNumberBase.2}%\RqtNumberBase is set in a higher-level file,
                  % may include section numbers for consistency.
% #2 is Title
{Directional Antenna}
% #3 is requirement label (expected to be of form rqt:XXX)
{rqt:Directional Antenna}
% #4 is Text
{The SYSTEM shall provide communications to an audience in a
 directional pattern from the transmitting source.}%1dA
% #5 is Specifics
{
\item The range of the transmission shall reach at least
      5 miles at the \MRA of the beam pattern.
\item The -3 \dB width of the transmission beam shall be
      at least \TBD degrees.
}
% #6 is Status (S in {(T), (O), (I), (D)} listed by phase as \item [] S)
{
\item [Phase 1] Threshold
}
% #7 is Acceptance
{This requirement shall be verified by demonstration.}
% #8 is Traceability
{
\item [5.1.6] Section in CDD for SYSTEM~\cite{ref__SYSTEM_CDD}.
}
% #9 is Notes; listed as enumeration \item ...
{
\item This requirement is provided to guide selection of the power
      output of the SYSTEM.

\item It is well known that the range is limited more by the antenna height
      and power of the target device than the transmitter power.
        This is meant to guide overall tradeoff analysis between \SWaP,
amplifier power, and antenna beam width
rather than to be a ‘‘hard requirement’’.
```

```
\item A final tradeoff will be needed between power (which drives \SWaP)
      and obtainable range to determine a final solution.
}
% #10 is version in which this specification was last changed;
%    to auto generate change bars
{P2}
%%%%% end \MULTIRQMTV[10] macro
```

This formats into a table that looks like, including a change-bar since this specification was changed in the latest version of the overall specification:

Specification 1.1.1.1.2 Directional Antenna	
Text	The SYSTEM shall provide communications to an audience in a directional pattern from the transmitting source.
Specifics	1. The range of the transmission shall reach at least 5 miles at the MRA of the beam pattern. 2. The -3 dB width of the transmission beam shall be at least ...TBD... degrees.
Status	**Phase 1** Threshold
Acceptance	This requirement shall be verified by demonstration.
Traceability	**5.1.6** Section in CDD for SYSTEM [4].
Notes	1. This requirement is provided to guide selection of the power output of the SYSTEM. 2. It is well known that the range is limited more by the antenna height and power of the target device than the transmitter power. This is meant to guide overall tradeoff analysis between SWaP, power, and antenna beam width rather than to be a "hard requirement". 3. A final tradeoff will be needed between power (which drives SWaP) and obtainable range to determine a final solution.

B.4 Follow-up Discussion

The above examples should provide an overview of why two different types of specifications are needed. Singular specifications such as a paint type, operating temperature, etc. should be captured in the `\ONERQMTV` *style. Multi-part specifications are better captured in the* `\MULTIRQMTV` *style. These two specification types lead to nuances about how the specification itself is described.*

Specifications need to fully specify all *aspects of the requirement so that the sentence provided in the "Text" field, and in conjunction with the "Specifics" field, can be verified entirely. As is shown, the singular specification text includes the thing to provide the capability,*

as well as the capability itself, and to whom the capability is to be delivered. This allows the verification step to fully understand the intent of the specification. When the `\MULTIRQMTV` *style is used, the list of additional specifications add to the overall verification procedure. Together, these parts of the specification dictate what counts as successful implementation of the specification.*

A final note needs to be made about specifications regarding the subject of the specification. Specifications levy requirements on the system, not those who interact with the system. This, all parts of the specification must *be directed at the "system". User interface requirements must be phrased in terms of the system, not the operator. And, likewise, for those affected by the system.*

"The system shall accept an input X from the user in order to perform an action."
is an acceptable requirement whereas

"The user inputs X to get the system to perform an action."
is not since it directs the user to do something rather than the system. This is a semantic difference that is essential in ensuring proper specification syntax.

Index