The cyber package*

Jared Jennings
jjennings@fastmail.fm

March 25, 2015

Contents

*This document corresponds to cyber v2.0, dated 2015/03/21.

1 What cyber is for

This package helps you construct documents that talk about compliance with named requirements.

For example, U.S. National Institute of Standards and Technology (NIST) Special Publication (SP) 800-53 sets out *security controls* for U.S. Federal agency information systems, one of which is AC-2c, "[The organization] establishes conditions for group and role membership." You may wish to write a document that says, "We comply with AC-2c, and here's how." This package is for you.

2 Document identifiers

Your document may talk about compliance with named requirements from multiple sources. For example, besides compliance with NIST SP 800-53 you might want to also talk about compliance with the U.S. Defense Information Systems Agency (DISA) UNIX Operating System (OS) Security Requirements Guide (SRG). Now to fully identify a requirement, you need to identify both the requirement and the document it comes from, for example "IA control IAAC-1," or "UNIX SRG Potential Discrepancy Item (PDI) GEN006480."

At the beginning of your document, you must identify all sources of named requirements; then you can talk about those sources when you are identifying the requirements. As examples here we use NIST SP 800-53, and assign it the document identifier `secctrl`; and we use the UNIX SRG, identifying it by `unixsrg`. See §7 for more.

3 Macros used in the body of your document

When you use the macros below to indicate requirements levied by policies handed down from above, or parts of code or documentation that relate to those requirements, you get some style for free, and an entry in an index. Also, other automated document features are made possible which are not strictly based on LaTeX; for example, a Puppet manifest annotated using cyber may have a table of compliance which a script constructs automatically by searching the manifest for uses of these macros.

3.1 Mentioning named requirements

Here are macros to use to indicate when you are talking about a specific requirement. These tags do not indicate any compliance posture, just that we're mentioning the requirement. So use these to refer to requirements in passing. Just like the names of celebrities are sometimes set in bold type in magazines, so that you can skip around and look for your favorite famous person, we want to make it easy to find all mentions of a given named requirement.

`\secctrl`

- `\secctrl {`⟨*control identifier*⟩`}`

\unixsrg

- `\unixsrg {`⟨*PDI*⟩`}`

The exact set of these macros depends on the set of requirements documents named at the beginning of your document; see §7.

Example:

```
Compliance with such and such requirement is similar to compliance with
\unixsrg{GEN002600}, \emph{q.v.}.
```

3.2 Making assertions about compliance posture

Most mentions of named requirements are going to be something like, "We comply with bla requirement," or "We don't comply with this one, and here's why." The following tags make it quick and simple for you to write such assertions.

\documents

Use `\documents` for notating some text in your document that says a fact about a system which supports a requirement. It takes two parameters: the type of requirement, and the name of the requirement. Example:

```
\documents{secctrl}{SC-18} We use only JavaScript, which is Category I
mobile code according to DoD guidelines.
```

Some requirements say something like, "All *mumble*s must be justified and documented with the Information Assurance Officer (IAO)." Use `\documents` to assert either that those things are documented, right here, or that you should look in *bla* document to find that documentation.

For example, the UNIX SRG requires that UNIX systems unable to require a password to enter single-user mode must be documented with the IAO.

```
\documents{unixsrg}{GEN000040} UNIX hosts unable to require a password
to enter single-user mode must be documented with the IAO. We administer
no such UNIX hosts.
```

(This is written in a document which is produced under the IAO's purview and approval; otherwise it would have to point to another document which is "with the IAO.")

\notapplicable

Just before some discussion about why a requirement is not applicable in a certain situation, put the `\notapplicable` tag. For example:

```
\notapplicable{secctrl}{SC-18(5)} We do not use mobile code at all.
```

\implements

The `\implements` tag notates a section of code or documentation which directly implements a requirement. It's usually used in automated policy. For example, the UNIX SRG says under GEN001362, "The `/etc/resolv.conf` file must be owned by root." A piece of Puppet policy that implements this requirement may go like:

```
# \implements{unixsrg}{GEN001362} Make sure resolv.conf
# is owned by root.
    file { '/etc/resolv.conf':
        owner => root,
    }
```

`\doneby` The `\doneby` tag is for requirements that are not fulfilled in an automated fashion: people are responsible for carrying them out on a day-by-day basis. It takes three parameters: who exactly does this thing, the kind of requirement, and the requirement's name. For example, the RHEL5 STIG requires that SELinux be turned on when installing RHEL5. Administrators have to make sure it's enabled during the install process.

```
\doneby{admins}{rhel5stig}{GEN000000-LNX00800} Make sure SELinux is
enabled when installing RHEL5.
```

`\bydefault` Use the `\bydefault` tag to talk about how a product we use implements a given requirement in its default configuration. It takes three parameters: the product we're talking about, the kind of requirement, and the requirement's name.—For example, the UNIX SRG says under GEN001300 that all library files must have mode `0755` or less permissive. Both RHEL 5 and RHEL 6 systems have this property by default.

```
\bydefault{RHEL5, RHEL6}{unixsrg}{GEN001300} All library files under
RHEL are mode \Verb!0755! or less permissive by default.
```

`\unimplemented` Use the `\unimplemented` tag to talk about a requirement we need to fulfill but haven't yet. (Don't use `\unimplemented` to talk about a requirement we don't intend ever to fulfill: it turns some things red to bring our attention to them; we don't want false alarms.)

Note carefully that this tag has three parameters: the kind of requirement, the exact requirement, and *the reason* the requirement is unimplemented. You can't use `\Verb` in the reason. Stick with plain text.

```
\unimplemented{unixsrg}{GEN008380}{Unless root kit checking
functionality is provided by the virus scanner, we would likely
want to deploy {\tt chkrootkit}. But this software is not yet
approved...}
```

4 Security labels

Some regulations require that documents be marked and labelled. These tags help you do that.

`\articlesecuritylabel` When you are writing an article (`\documentclass{article}`), this tag helps you label it. It will put the text you give at the top of every page. Put the tag right after `\usepackage{cyber}`, as close to the top of the document source as possible. That way it will be visible to viewers of the document source as well.

Most reStructuredText documents use the article class.

Example, in LaTeX:

```
\articlesecuritylabel{UNCLASSIFIED//FOUO}
```

\booksecuritylabel `\booksecuritylabel` acts the same as `\articlesecuritylabel`, but this is the one you should use if you are using the `book` document class.

Part of marking a document properly is writing the distribution statement and statement of disposition on the title page.

See `http://www.dtic.mil/dtic/submit/guidance/distribstatement.html` on distribution statements and on destruction of limited-distribution unclassified documents.

5 Distribution statements

Write the distribution statement right after the security label. Statements allowed by DoDI 5320.24 as of 23 August 2012 are A through F. Macros for these are provided. Dates are set to `\today`. For guidance on the reasons, see the DoD instruction.

Just like you need `\maketitle` to make your `\title` show up, you need to (instead) use `\makedodtitle` to make your distribution statement show up.

\distributionA To mark your document with Distro A (public release), write `\distributionA {`⟨*other information*⟩`}`. Local policies may require that you include a reference number, or the name of the organization which cleared the document, or some such. That's "other information."

\distributionB To U.S. Government agencies only: `\distributionB {`⟨*reason why*⟩`} {`⟨*controlling office*⟩`}`.

\distributionC To U.S. Government agencies and their contractors: `\distributionC {`⟨*reason why*⟩`} {`⟨*controlling office*⟩`}`.

\distributionD To DoD and DoD contractors only: `\distributionD {`⟨*reason why*⟩`} {`⟨*controlling office*⟩`}`.

\distributionE To DoD Components only: `\distributionE {`⟨*reason why*⟩`} {`⟨*controlling office*⟩`}`.

\distributionF Only as directed by the controlling office or higher DoD authority: `\distributionF {`⟨*controlling office*⟩`}`.

You are responsible for understanding what these statements mean and whether you're allowed to use them.

6 Document formatting aids

6.1 DoD title page

\makedodtitle In a book-class document, use `\makedodtitle` instead of `\maketitle` and you will get an organization logo on your title page. You'll also get the distribution statement. You must provide the organization logo, as a graphics file of a suitable

format (*e.g.*, PNG or PDF if you're using `pdflatex`, EPS if you're using `latex`, that sort of thing), entitled `org_logo`, *e.g.* `org_logo.png`.

6.2 Narrower margins

`\narrowermargins` Nobody appreciates the rules of typography these days. And making a bad impression on auditors by submitting a document that doesn't look how they expect is a needless risk.

This macro narrows the margins, while leaving a sizable margin in which to write text. It probably only works on books (not articles). It takes the security label into account.

In the top of your document, put `\narrowermargins` to make all of the margins narrower by a half inch, increasing the text area by one inch horizontally and one inch vertically. You have to put it above the security label.

This has not been tested much, and only on a book-class document.

6.3 Change log

Most documents submitted in support of cybersecurity goals must indicate what version they are, and what changes have happened between versions. In software development terminology, these correspond to *releases*, not *revisions* or *changesets*; that is to say, these versions are likely not to change more than once a month, and changes are likely to be, "Added a new chapter about foo," rather than "Corrected spelling on page 358."

Write a changelog and call it `changelog.tex`. Include it in your front matter just after the table of contents. Its format is like so:

```
\begin{changelog}

\change{3 Aug 2011}{Jared Jennings}{Added Backup Plan}

\change{2 Aug 2011}{Jared Jennings}{New server purchased; changed Hardware
Baseline to match}

\end{changelog}
```

Add new entries to the top of the changelog, not the bottom.

6.4 Executive summary

An "executive summary" is supported. This is just a section containing a table which lists the larger security controls this document is about. Its format is like so:

```
\begin{executivesummary}
\executivesummaryiacontrol{IAIA-1}{Individual Authentication}
\end{executivesummary}
```

Put it after the changelog, after the ToC.

7 Document identifiers

To identify a requirement you want to talk about without ambiguity, you need to give a name for both the requirement and the document it came from. Since document identifiers are written in potentially hundreds of compliance tags across a document, they need to be short, but unique. Here are some example document identifiers:

iacontrol Information Assurance Controls, laid out in DoDI 8500.2, e.g. DCMC-1.

secctrl Security Controls, laid out in NIST SP 800-53, e.g. SC-18c.

unixsrg UNIX Security Requirements Guide (SRG) Potential Discrepancy Identifiers (PDIs), e.g. GEN008520.

rhel5stig Red Hat Enterprise Linux (RHEL) 5 STIG PDIs.

7.1 Defining requirements documents

`\requirementsdocument`

In the preamble of your document, you must tell `cyber` from which documents your requirements will come. Use the `\requirementsdocument` macro to do this. For example,

```
\requirementsdocument{joesaidso}{Joe's Demands}{Demand }
```

The first parameter is the document identifier, which is used in the compliance tags. For example, after this `\requirementsdocument`, in the body of our document we could write

```
\documents{joesaidso}{Joe34} We close the door when going outside, because
Joe said to. Door-closing logs are kept just inside the door.
```

The second parameter is the header under which all requirements from this document will be placed in the Compliance index. Given the above code, the index might look like:

```
Compliance

 Joe's Demands
   Joe34
     documented, 3
```

The third parameter is the prefix with which requirements from this document will be named. For example, if we were to write `\joesaidso{Joe34}` in the body of a document, we would see "Demand Joe34" in the rendering of that document.

8 Adding cyber to a document

If you are editing or adding to an existing document which already uses cyber, you need not concern yourself with this section.

To use cyber in a new document: you should have the cyber package installed; see `http://en.wikibooks.org/wiki/LaTeX/Packages/Installing_Extra_Packages`.

At the top of your document, in its preamble, put the following lines:

```
\usepackage{cyber}
```

This makes the indices that the cyber tags put entries into.

Somewhere in the preamble of your document, after you use the cyber package, tell cyber about all your `\requirementsdocument`s.

Somewhere at the end of your document, put

```
\printindex{cyber_compliance}{Compliance}
```

This grabs the index made and makes it part of the final document.

8.1 Controls in contents

If you want the IA controls covered in a section to be summarized in the section's title written in the table of contents, use this additional package:

```
\usepackage{iacic}
```

Write this after the usepackage directive for cyber. If you use the hyperref package, write this statement after the usepackage directive for hyperref.

If you use iacic, your section titles cannot contain any formatting, and your document cannot be originally written in reStructuredText. Section titles written in the body text will not change: only the ones in the contents. Section titles written in page headers will vary. In general this is a buggy, wonky feature. See `http://tex.stackexchange.com/questions/29818`. Dive into the wonderful world of TEX and LATEX. Learn you a fix for great good!

8.2 Temporarily suppressing index entries

If some part of your document is an auto-summary of another part of your document, to index mentions of requirements in the summarized part may be redundant and undesirable.

`\Cyberindexingenabledfalse`
`\Cyberindexingenabledtrue`

Use these macros to turn off and on (respectively) indexing of compliance assertions for part of your document. For example:

```
\Cyberindexingenabledfalse
\include{auto_summary}
\Cyberindexingenabledtrue
```

9 Interactions with other packages

If you use hyperref, do so *before* you use cyber.

Because cyber requires longtable, you can't use two columns in a document that uses cyber. The requirement for longtable stems from the executive summary and changelog, which may have enough rows to span pages.

Because cyber requires index, any other package which automatically indexes things may not work in conjunction with cyber.

cyber also pulls in color, fancyhdr and graphicx.

10 Implementation

```
\makeatletter
```

We require longtable for the changelog and executive summary, which could span pages.

```
\RequirePackage{longtable}
```

We require color so that we can make red some text regarding unimplemented requirements.

```
\RequirePackage{color}
```

There are potentially hundreds of requirements with which to comply. So we want them in their own index.

```
\RequirePackage{index}
```

The fancyhdr package is how we add security labels.

```
\RequirePackage{fancyhdr}
```

And the graphicx package is needed so the organization's logo can be put on the front page.

```
\RequirePackage{graphicx}

```

Index entries are written into the compliance index.

```
\newindex{cybercompliance}{cybercompliance.idx}{cybercompliance.ind}{Compliance}
```

Notations of compliance throughout the document are written in compliance.aux and explanations.aux.

```
\AtBeginDocument{
\newwrite\complianceaux
\immediate\openout\complianceaux=cyber_compliance.aux
\newwrite\explanationsaux
\immediate\openout\explanationsaux=cyber_explanations.aux
}

\AtEndDocument{
\immediate\closeout\complianceaux
\immediate\closeout\explanationsaux
}
```

The foreach code below is from: `http://stackoverflow.com/questions/2402354/split-comma-separated-parameters-in-latex` and/or `http://maraist.`

org/comma-separated-lists-in-latex_08-2009/. Added two extra pass-through parameters.

`\foreach` Functional foreach construct. Usage: `\foreach {`⟨*macro*⟩`} {`⟨*firstarg*⟩`} {`⟨*secondarg*⟩`} {`⟨*third1,third2,third3,...*⟩`}`. Evaluates to `\macro{firstarg}{secondarg}{third1} \macro{firstarg}{secondarg}{third2}` etc.

Put another way: ⟨*#1*⟩ is a function to call once for each comma-separated item in ⟨*#4*⟩. ⟨*#2*⟩ and ⟨*#3*⟩ are parameters to pass to function in ⟨*#1*⟩ as the first parameters. ⟨*#4*⟩ is a comma-separated list of items, each of which to pass as the third parameter to an invocation of function ⟨*#1*⟩.

```
\def\foreach#1#2#3#4{%
  \@test@foreach{#1}{#2}{#3}#4,\@end@token%
}
```

Internal helper function - Eats one input.

```
\def\@swallow#1{}
```

Internal helper function - Checks the next character after #1 and #2 and continues loop iteration if `\@end@token` is not found.

```
\def\@test@foreach#1#2#3{%
  \@ifnextchar\@end@token%
    {\@swallow}%
    {\@foreach{#1}{#2}{#3}}%
}
```

Internal helper function - Calls `#1{#2}{#3}{#4}` and recurses. The magic of splitting the third parameter occurs in the pattern matching of the `\def`.

```
\def\@foreach#1#2#3#4,#5\@end@token{%
  #1{#2}{#3}{#4}%
  \@test@foreach{#1}{#2}{#3}#5\@end@token%
}
```

`\requirementsdocument` Usage: `\requirementsdocument {`⟨*tag*⟩`} {`⟨*index heading*⟩`} {`⟨*requirement title*⟩`}`. The tag is a one-word lowercase name, like iacontrol or unixsrg or rhel5stig, that you'll use in `\implements` and other tags to say from which document comes the requirement you're talking about. An index of compliance will be made; `index heading` is the title under which all compliance with this document will be listed in the compliance index, for example "NIST SP 800-53 Security Controls." The `requirement title` is how to name a requirement from that document. For example, the DISA UNIX SRG contains many requirements, called Potential Discrepancy Items (PDI). So you may give "UNIX SRG PDI " (note the space at the end) as the requirement title, and when a requirement is named using `\requirement` it will be prefixed with the title, e.g. "UNIX SRG PDI GEN006480."

Write all `\requirementsdocument`s in the preamble of the document.

```
\newcommand{\requirementsdocument}[3]{%
    % Say how to name requirements that come from this document.
    \expandafter\providecommand\csname #1\endcsname[1]{%
        \namedrequirement{#1}{#3}{##1}}
```

```
    \expandafter\newcommand\csname indexhead#1\endcsname{#2}
}
```

`\requirement` Usage: `\requirement {`⟨*document tag*⟩`} {`⟨*requirement name*⟩`}`. Example: `\requirement{unixsrg}{GEN006480}`. Use this macro to mention requirements that come from other documents, without implying a particular compliance posture. For example, "The check content of `\requirement{unixsrg}{GEN006480}` suggests the following solution."

```
\newcommand{\requirement}[2]{%
    \expandafter\csname requirement@#1\endcsname{#2}}
```

`\namedrequirement` Usage: `\namedrequirement {`⟨*index name*⟩`} {`⟨*official title prefix*⟩`} {`⟨*requirement name*⟩`}`. Example: `\namedrequirement{iacontrol}{IA Control }{IAAC-1}` will say "IA Control IAAC-1" in the body of the document, and make an index entry in the compliance index under the "IA Controls" heading. ("IA Controls" in this example is the value of a parameter given to `\requirementsdocument`, not to this macro.)

This macro exists so it will be possible to redefine it to change how such a thing looks. When referring to a named requirement in a document that you are writing, use `\requirement` above, not this macro.

The index macro is the same as in indexhelper below but without the last `!{#2}` for the subhead.

```
\newcommand{\namedrequirement}[3]{%
    \index[cybercompliance]{\csname indexhead#1\endcsname @\iaindexheadstyle{\csname indexhead#
    {\small #2\mbox{#3}}%
}
```

This will be redefined by iacic if that package is included.

```
\def\addtosectionname#1{}
```

This is here so you can turn off indexing for a portion of your document.

```
\newif\ifCyberindexingenabled
\Cyberindexingenabledtrue
```

Style the text for per-document heads in the compliance index.

```
\newcommand{\iaindexheadstyle}[1]{%
    \vspace{1em}{\large \textbf{#1}}\vspace{1em}}
```

parameters: index name, implementation status, requirement name

```
\newcommand{\indexhelper}[3]{%
\ifCyberindexingenabled\index[cybercompliance]{\csname indexhead#1\endcsname @\iaindexheadstyle
```

parameters: index name, prefix (e.g. RHEL5:), requirement name

```
\newcommand{\marginhelper}[3]{#2~\mbox{#3}\\ }
```

parameters: index name, compliance status, requirement name. Compliance status is not necessarily the same as implementation status.

```
\newcommand{\compliancehelper}[3]{%
\write\complianceaux{#1:#3:#2}}
```

parameters: index name, explanation, requirement name

```
\newcommand{\explanationshelper}[3]{%
\write\explanationsaux{#1:#3:}%
\write\explanationsaux{#2}%
\write\explanationsaux{:}%
}
```

parameters: index name, nothing, requirement name

```
\newcommand{\sectionnamehelper}[3]{%
    \def\@numberone{#1}%
    \def\@iacontrol{iacontrol}%
    \ifx\@numberone\@iacontrol\addtosectionname{#3}\fi}

\def\iamarginsize{\footnotesize}
```

These all take two parameters. The first parameter is the identifier of the document from which the requirement comes (e.g. iacontrol, unixstig, spanstig, apachestig, etc)—same names as the indices. The second parameter is which requirement(s) is/are in the given status (e.g. GEN000320). You may supply multiple values for this one, separated by commas.

Example: `\implements{unixsrg}{GEN000320,GEN000340}`

```
\newcommand{\implements}[2]{%
    \@bsphack
    \hspace{0pt}% this makes sure the marginpar doesn't go with the previous
                % paragraph
    \foreach{\indexhelper}{#1}{implemented}{#2}%
    \marginpar{\iamarginsize \raggedright%
        \foreach{\marginhelper}{#1}{auto:}{#2}%
    }%
    \foreach{\compliancehelper}{#1}{compliant}{#2}%
    \foreach{\sectionnamehelper}{#1}{}{#2}%
    \@esphack}
\newcommand{\unimplemented}[3]{%
    \@bsphack
    \hspace{0pt}% this makes sure the marginpar doesn't go with the previous
                % paragraph
    \foreach{\indexhelper}{#1}{\textcolor{red}{unimplemented}}{#2}%
    \marginpar{\iamarginsize \color{red} \raggedright%
        \foreach{\marginhelper}{#1}{}{#2}%
    }%
    \foreach{\compliancehelper}{#1}{non-compliant}{#2}%
    \foreach{\explanationshelper}{#1}{#3}{#2}%
    \foreach{\sectionnamehelper}{#1}{}{#2}%
    {#3}%
    \@esphack}
```

Except this one takes three parameters: operating system(s), identifier of requiring document, requirement(s).

```
\newcommand{\bydefault}[3]{%
    \@bsphack
```

```
    \hspace{0pt}% this makes sure the marginpar doesn't go with the previous
                % paragraph
    \foreach{\indexhelper}{#2}{default for {#1}}{#3}%
    \marginpar{\iamarginsize \raggedright%
        \foreach{\marginhelper}{#2}{#1: }{#3}%
    }%
    \foreach{\compliancehelper}{#2}{compliant}{#3}%
    \foreach{\sectionnamehelper}{#1}{}{#2}%
    \@esphack}
\newcommand{\notapplicable}[2]{%
    \@bsphack
    \hspace{0pt}% this makes sure the marginpar doesn't go with the previous
                % paragraph
    \foreach{\indexhelper}{#1}{N/A}{#2}%
    \marginpar{\iamarginsize \raggedright%
        \foreach{\marginhelper}{#1}{N/A: }{#2}%
    }%
    \foreach{\compliancehelper}{#1}{N/A}{#2}%
    \@esphack}
\newcommand{\doneby}[3]{%
    \@bsphack
    \hspace{0pt}% this makes sure the marginpar doesn't go with the previous
                % paragraph
    \foreach{\indexhelper}{#2}{prescribed}{#3}%
    \marginpar{\iamarginsize \raggedright%
        \foreach{\marginhelper}{#2}{#1 do }{#3}%
    }%
    \foreach{\compliancehelper}{#2}{compliant}{#3}%
    \foreach{\sectionnamehelper}{#1}{}{#2}%
    \@esphack}
```

deprecated

```
\newcommand{\prescribed}[2]{\doneby{admins}{#1}{#2}}
\newcommand{\documented}[2]{%
    \@bsphack
    \hspace{0pt}% this makes sure the marginpar doesn't go with the previous
                % paragraph
    \foreach{\indexhelper}{#1}{documented}{#2}%
    \marginpar{\iamarginsize \raggedright%
        \foreach{\marginhelper}{#1}{}{#2}%
    }%
    \foreach{\compliancehelper}{#1}{compliant}{#2}%
    \foreach{\sectionnamehelper}{#1}{}{#2}%
    \@esphack}
\newcommand{\documents}[2]{\documented{#1}{#2}}
```

`\articlesecuritylabel` Use this to label documents of class *article*. Use like `\articlesecuritylabel{`⟨*thing to write on each page*⟩`}`. The thing will be written as part of the page header.

```
\newcommand{\articlesecuritylabel}[1]{%
    \pagestyle{fancy}%
    \renewcommand{\headrulewidth}{0pt}%
    \fancyhf{}%
    \chead{#1}%
    \cfoot{\thepage}%
    \fancypagestyle{plain}{%
        \fancyhf{}%
        \fancyhead[C]{#1}%
    }%
}
```

\booksecuritylabel Use this one to label books. Syntax is the same as above.

```
\newcommand{\booksecuritylabel}[1]{%
    \pagestyle{fancy}%
    \renewcommand{\headrulewidth}{0.1pt}%
    \fancyhf{}%
    \fancyhead[CE,CO]{#1\vspace{2\baselineskip}}%
    \fancyhead[LE,RO]{\thepage}%
    \fancyhead[RE]{\nouppercase{\leftmark}}%
    \fancyhead[LO]{\nouppercase{\rightmark}}%
    %
    \fancypagestyle{empty}{%
        \renewcommand{\headrulewidth}{0pt}%
        \fancyhf{}%
        \fancyhead[C]{#1\vspace{2\baselineskip}}%
    }%
    \fancypagestyle{plain}{%
        \renewcommand{\headrulewidth}{0pt}%
        \fancyhf{}%
        \fancyhead[C]{#1\vspace{2\baselineskip}}%
    \fancyfoot[C]{\thepage}%
    }%
}
```

See `http://www.dtic.mil/dtic/submit/guidance/distribstatement.html` on distribution statements and on destruction of limited-distribution unclassified documents.

```
\def\@distribution{\@latex@warning@no@line{No distribution statement given}}
```

```
\def\@destruction{%
    \textbf{DESTRUCTION NOTICE}: Destroy by any method that will prevent disclosure or reconstr
```

\distributionA

```
\def\distributionA#1{%
    \gdef\@distribution{%
        \textbf{DISTRIBUTION STATEMENT A}: Approved for public release. #1}}
```

\distributionB

```
\def\distributionB#1#2{%
    \gdef\@distribution{%
        \textbf{DISTRIBUTION STATEMENT B}: Distribution authorized to U.S. Government agencies
```

\distributionC

```
\def\distributionC#1#2{%
    \gdef\@distribution{%
        \textbf{DISTRIBUTION STATEMENT C}: Distribution authorized to U.S. Government agencies
```

\distributionD Use this macro to indicate that your document uses Distribution Statement D, and why, like so: `\distributionD {`⟨*reason why distribution is limited*⟩`}`. Write this in the top of your document, before `\begin{document}`.

```
\def\distributionD#1#2{%
    \gdef\@distribution{%
        \textbf{DISTRIBUTION STATEMENT D}: Distribution authorized to the Department of Defense
```

\distributionE

```
\def\distributionE#1#2{%
    \gdef\@distribution{%
        \textbf{DISTRIBUTION STATEMENT E}: Distribution authorized to DoD Components only #1 \@
```

\distributionF

```
\def\distributionF#1{%
    \gdef\@distribution{%
        \textbf{DISTRIBUTION STATEMENT F}: Further dissemination only as directed by #1 \@date\
```

```
\def\@version{}
```

\version Use the `\version {`⟨*some indication of version*⟩`}` to indicate the version of the document.

```
\def\version#1{\gdef\@version{#1}}
```

\today This redefinition of the `\today` macro formats the date like "4 Oct 2011," as military folks are used to. Use it in the `\date` macro.

```
\def\today{\number\day\space\ifcase\month\or Jan\or Feb\or Mar\or Apr\or
  May\or Jun\or Jul\or Aug\or Sep\or Oct\or Nov\or Dec\fi
  \space\number\year}
```

\narrowermargins

```
\newcommand{\narrowermargins}{%
    \addtolength{\hoffset}{-0.5in}%
    \addtolength{\textwidth}{1in}%
    \addtolength{\marginparwidth}{-0.5in}%
    \addtolength{\voffset}{-0.5in}%
    \addtolength{\textheight}{1in}%
    % bring up bottom of text to match whitespace at top caused by security
    % label
    \addtolength{\textheight}{-2\baselineskip}%
}
```

`\makedodtitle` Use this macro instead of `\maketitle`, right after `\begin{document}`.

```
\newcommand{\makedodtitle}[0]{%
    \begin{titlepage}%
    \begin{center}%
    ~\vskip 1.5in%
    {\LARGE \@title \par}%
    \vskip 3em%
    {\large \lineskip .75em \begin{tabular}[t]{c}%
      \@author%
     \end{tabular}\par}%
    {\large \@date \par \large \@version}\vskip 0.5in%
    \includegraphics[width=0.4\textwidth]{org_logo}
    \end{center}\par%
    \@thanks\vskip 0.5in\par%
    \@distribution\par%
    \@destruction%
    \vfil\null
    \end{titlepage}}
```

`changelog` `\change` Usage: `\change {`⟨*when*⟩`} {`⟨*who*⟩`} {`⟨*what*⟩`}`. For example,

```
\change{3 Aug 2011}{Jared Jennings}{Added backup plan}
```

```
\newcommand{\change}[3]{%
    {#1} & {#2} & {#3}\\
}
```

```
\newenvironment{changelog}{
    \phantomsection
    \chapter*{Changelog}
    \addcontentsline{toc}{chapter}{Changelog}
    \begin{longtable}{@{\extracolsep{\fill}} l l p{4in}}
    \hline
    {\bf Date} & {\bf Person} & {\bf Change Description}\\
    \hline\hline
    \endhead
}{\end{longtable}}
```

`executivesummary` An "executive summary," in the context of documents submitted as information assurance artifacts, means specifically a table of IA controls to which this document applies. Executive summaries can be built automatically by the shaney script, but must be included in the main document by means of the `\include` command.

`\executivesummaryiacontrol` Usage: `\executivesummaryiacontrol {`⟨*IA control ID*⟩`} {`⟨*IA control name*⟩`}`. Use only inside the executivesummary environment.

```
\newcommand{\executivesummaryiacontrol}[2]{%
    {#1} & {#2}\\
}
```

```
\newenvironment{executivesummary}{
    \phantomsection
    \chapter*{Executive Summary}
    \addcontentsline{toc}{chapter}{Executive Summary}
    The following table lists the NIST SP 800-53 Security Controls that are satisfied
    through this artifact.

    \begin{longtable}{@{\extracolsep{\fill}} l p{4in}}
    \hline
    {\bf IA Control Number} & {\bf IA Control Name}\\
    \hline\hline
    \endhead
}{\end{longtable}}
```