**Papofeed Parser User Guide and Documentation**

Paul Lee (paul.lee@nist.gov)

Michelle Steves (michelle.steves@nist.gov)

National Institute of Standards and Technology

Bldg. 225, Rm. A227

100 Bureau Drive, Mail Stop 8940 Gaithersburg, MD 20899-8940

Table of Contents

INTRODUCTION

1.1 Overview5

1.2 Installation6

1.3 Parser Usage7

FILES

2.1 docx2xml.py8

2.2 papo\_interface.py8

2.3 elements.py8

2.4 extractor.py8

2.5 toxml.py9

2.6 feed.py9

1. Concept Examples

3.1 Groups, Pages, Questions11

3.1.1 Groups11

3.1.2 Pages11

3.1.3 Questions11

3.2 Insert, Mod, Regular Categories13

3.2.1 Insert13

3.2.2 Mod13

3.2.3 Regular Categories13

1. Element Syntax Instructions and Examples

4.1 General Structure15

4.2 Misc. Formatting15

4.3 Group15

4.4 Page17

4.5 Question19

4.5.1 ID19

4.5.2 Title20

4.5.3 Instructions21

4.5.4 Text22

4.5.5 Note23

4.5.6 Comment24

4.5.7 CDATA25

4.5.8 Indent25

4.5.9 Display\_When26

4.5.10 Display\_Where27

4.5.11 Response\_Validation28

4.5.12 Radio Button29

4.5.13 Select Box Button30

4.5.14 Numerical32

4.5.15 Textbox32

4.5.16 Memo33

4.5.17 Cloze34

4.5.18 Clone35

4.5.19 Insert36

4.5.20 Mod37

4.5.21 BNF\_Mapping37

1. QUESTIONNAIRE TAG DESCRIPTIONS

4.1 <questionnaire>39

4.1.1 <xmlversion> 39

4.1.2 <qaversion>39

4.1.3 <BNFversion>39

4.1.4 <indent\_px>39

4.1.5 <default\_insert\_text >39

4.2 <index>39

4.2.1 <index> Attributes39

4.2.2 <group>39

4.2.2.1 <group> Attributes39

4.2.2.2 <group> <instructions>40

4.2.2.3 <group> <text>40

4.2.2.4 <group> <page>40

4.3 Questions40

4.3.1 Question Attributes40

4.3.2 Question <title>41

4.3.3 Questions <instructions>41

4.3.4 Questions <text>41

4.3.5 Questions <note>41

4.3.6 Questions <response>41

4.4 Responses41

4.4.1 Response Attributes41

4.4.2 Response <validation>41

4.4.3 Response <BNF\_mapping>42

4.4.4 Response <text>42

4.4.5 Response <option>42

4.5 Additional Comments43

4.5.1 Additional Comment <comments>43

4.6 Other Pages with Comments43

4.6.1 Other Pages with Comments <general comments>43

4.6.2 Other Pages with Comments <demographics\_survey>43

4.7 Static Text 43

4.7.1 Static Text <static\_text> 43

4.7.2 Static Text <sign\_in>44

4.7.3 Static Text <sign\_up>44

4.7.4 Static Text <policies\_for\_review>44

4.7.5 Static Text <questionnaire>45

4.7.6 Static Text <demographics\_survey>45

4.7.7 Static Text <admin\_configure>45

4.7.8 Static Text <admin\_testing\_setup>45

4.7.9 Static Text <admin\_reporting>45

4.7.10 Static Text <general\_comments>45

4.7.11 Static Text <completion\_page>46

4.7.12 Static Text <help>46

4.8 JSON46

4.8 JSON {“cloze”:{ …}} 46

4.8 JSON {“insert”:{ …}} 47

**1 Introduction**

**Overview**

The Papofeed parser is a tool developed for use with the Password Policy Taxonomy Software, and is responsible for page structure, question creation, populating instructions, transferring markup data, the inclusion of response forms, and more.

Each page of the browser-based questionnaire is made up of questions, responses, and comments. These elements can be customized to using various attributes and sub-elements such as text, instructions, validations, and more.

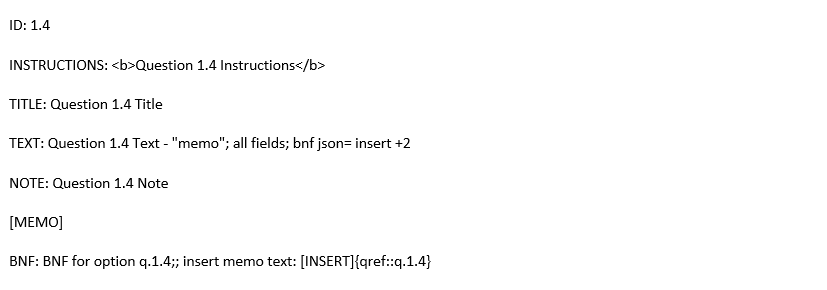


Figure - Input Document Syntax

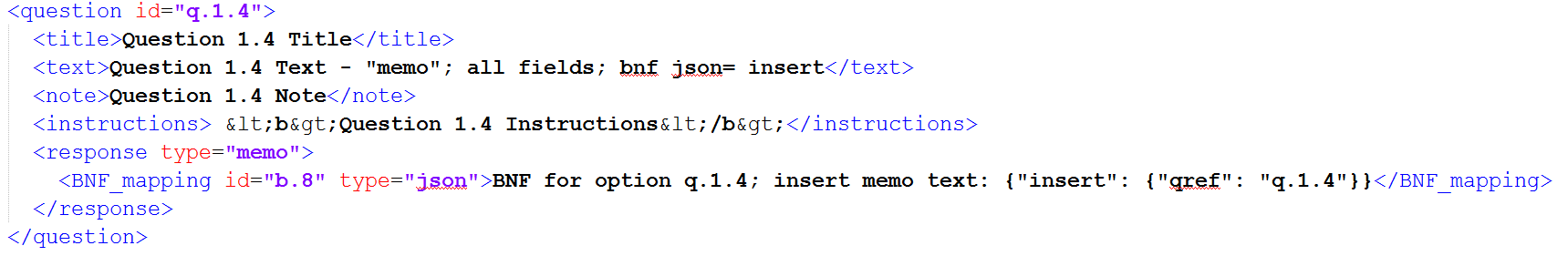


Figure - XML output

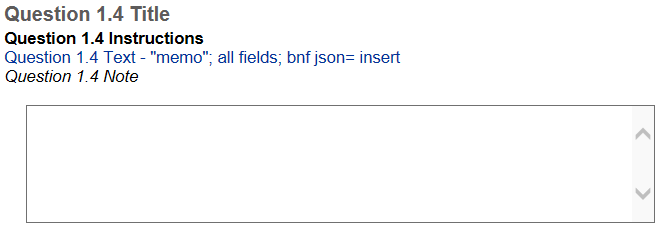


Figure - What is displayed in the browser

To generate the XML used as the contents of the questionnaire, a .doc(x) file with elements adhering to the Papofeed syntax is fed into the parser. The parser combines the contents of this file with additional element and formatting information from the template file, and outputs the complete XML file to the output directory. This file can then be used in conjunction with the server application code to update the contents of the questionnaire.

Figures 1-3 above are an example of an example question in the word document, the resulting XML, and how it looks in the questionnaire application.

**INSTALLATION**

1. Navigate to the “installation” folder of the unzipped folder and double click on “python-2.7.6” to install python.
2. Within the “installation” folder of the unzipped folder, double click on “ez\_setup.exe” to install the setuptools module.
3. Within the “installation” folder of the unzipped file, right click on “papofeed-0.2.linux-x86\_64.exe” and select “Run as Administrator” to install papofeed.
4. Within the “installation” folder of the unzipped file, double click on “PyQt4-4.11.3-gpl-Py2.7-Qt4.8.6-x32” to install PyQt4.
5. Add the respective folders to your environment variables.
   1. Windows 7:
      1. Start button -> Computer -> System Properties -> Advanced System Settings -> Environment variables
      2. Navigate to the System Variables window
      3. Scroll down to the Path variable, select it, then select edit.
      4. Within the Variable Value textbox, paste your Python install directory, ending with a semi-colon. Ex. C:\Python27;
      5. Paste the site packages folder of your Python installation into this textbox, ending with a semicolon. Ex. C:\Python27\Lib\site-packages;
      6. Paste the scripts folder of your Python installation into this box, ending with a semicolon. Ex. C:\Python27\Scripts;
6. Open the command prompt
   1. Type ‘easy\_install jinja2’ and press return to install Jinja2
   2. Type ‘easy\_install pip’ and press return to install pip
7. Download and install 32bit version of lxml.
   1. Navigate to <http://www.lfd.uci.edu/~gohlke/pythonlibs/#lxml>
   2. Click on “lxml-3.7.3-cp27-cp27m-win32.whl” to download the .whl file.
      1. Use the win32 version regardless of whether your machine is x64 or x84
   3. Open a command line window.
      1. Navigate to where the .whl file was saved.
      2. Type “pip install lxml-3.6.1-cp27-cp27m-win32.whl” and press return to install lxml.

**Usage**

Via the command line or windows explorer, navigate to the directory where papo\_interface.py is located (should be right inside main downloaded folder). Start papo\_interface.py and the GUI will appear.

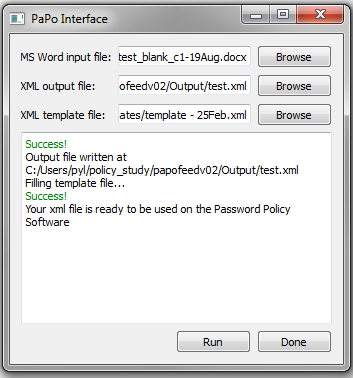


Figure - The Papofeed Parser Interface

Select the input .docx file as the “MS Word input file”, the desired path and name for the output XML file for “XML output file”, and the desired template for “XML template file”. The template file should be provided for you. If successful, the window will show a confirmation message and the output file will appear in the selected directory. If unsuccessful, the window will display which line in the input file the error is on.

**Note:**  Browse buttons currently default to the “Questionnaires”, “Templates”, and “Output” folders within the papofeed directory. If these folders do not exist, the browse buttons will default to the papofeed directory.

1. **FILES**
   1. **docx2xml**

Dependencies: os, re, extractor, feed, elements, toxml

Docx2xml serves as the main overhead module, triggering parts of extractor.py, feed.py, elements.py, and toxml.py. Main functions include saving the text from the input file as an XML string, extracting the plain text, and then parsing the plain text string with the parse function from feed.py. Once the string is parsed, toxml.populate() uses the elements module lists to feed an XML tree, then writes it to an output.xml file.

* 1. **papo\_interface.py**

Dependencies: sys, glob, os, subprocesses, QtGui + QtCore (from PyQt4), expanduser (from os.path)

Module responsible for setting up the interface shown in Figure 4.

* **\_\_init\_\_**
  + Basic gui setup, including buttons, text fields, and display boxes.
* **browseDocxFiles(), browseXMLFiles(), browseAndSetOutput()**
  + Sets default directories for the three browse buttons.
* **Run()**
  + Handles error checking for both the docx2xml output file and the template filling process. Displays messages for successes and failures.

**Note:**  Browse buttons currently default to the “Questionnaires”, “Templates”, and “Output” folders within the papofeed directory. If these folders do not exist, the browse buttons will default to the papofeed directory.

* 1. **elements.py**

Dependencies: re

Module with XML element class representations

* 1. **extractor.py**

Dependencies: lxml.etree, zipfile, re

Module to extract data from the input .docx file.

* **Extract\_xml\_string()**
  + Receives the path of a .docx file and returns a OOXML string
* **Extract\_plain\_text()**
  + Receives the OOXML string produced by extract\_xml\_string and returns a plain text string with the relevant data.
  + Accounts for bolding, underlining, italics, and strong style.
  + Replaces Microsoft word auto-conversion of particular character sequences into non-Unicode characters, back into their original states. Ex. Three periods meant for a Unicode representation of an ellipses becomes \xe2\x80\xa6 via Microsoft Word’s auto-conversion process.
  1. **toxml.py**

Dependencies: lxml.etree, elements

Module that takes the various lists of elements generated by feed.py, and generates the XML output file.

* **Populate()**
  + Main method that uses the elements module lists to feed an XML tree.
  + Loops through all the question elements in elements.\_\_questions\_\_ and checks for various attributes and sub-elements.
  + Once found, sets the sub-element or attribute of the current question or comment to the content of the sub-element or attribute of the question/comment in the element module lists.
* **Write()**
  + Receives a path and writes the XML to it.
* **Populate\_group()**
  + Receives a group object and a XML element and populates this element with the group object
  1. **feed.py**

Module that takes in the plain text generated by extractor.py, and creates objects for the element classes based on specific text occurrences.

**parse()**The plain text is turned into an array of lines, with each line being parsed for keywords such as PAGE, TITLE, NOTE, VALIDATION, etc. Once a keyword is found, the attribute/element mapped to the keyword is added to the current question, page, or group.

**replace\_with\_json()**For each line of the plain text string, replaces the special fields(Insert, Memo, Cloze, Numerical, Select one, Select multi) with JSON code described in utils/questionnaire\_tag\_description.docx

**deleteCDATAContructs()**  
Removes the CDATA formatting syntax found at the beginning and end of CDATA element lines - i.e ‘<![CDATA[ ' and ' ]]>’

**Find\_nth()**  
 Finds and returns the index of the nth occurrence of a character in a string.

**replaceRegCatDelinForMod()**  
 Replaces all delineators in all non-insert/mod qref fields – from ‘::’ to ‘:’

**convertRegCatToXML()**Adds double quotes to the beginning and end of all non-insert/mod qref fields

**replaceInsertDelinForMod()** Replaces all delineators in all insert qref fields – from ‘::’ to ‘:’

**convertInsertArrayToXML()**  
Based on what position a qref element within an insert statement is in, different formatting changes are needed to comply with XML conversion. This function accounts for the qrefs of starting insert, mods within insert statements, and non-insert/mod qrefs.

**replaceArrayWithXML()**  
The current line’s original docx qref syntax is replaced by the converted, XML compatible qref elements.

**cleanupFinalStatement()**  
Performs misc. formatting fixes to the converted XML statement including adding/removing brackets, character replacement, etc.

**linePreProcessing()**  
Gets rid of line whitespace after ‘::’ delineations found in mod statements.

**removeInsertWS()**  
Gets rid of white space after the :: and ;; delineators in insert statements, as well as the white-spaces between the end of an insert and non-white-space character.

**convertToXML()**The main insert/mod XML conversion method that takes in the unformatted line, splits them up into different qref types – inserts, mods, nested inserts/mods, non-insert/mods – and converts the syntax from the word doc syntax to XML compatible syntax.

**removeInputTypeSpaces()**Takes in the line, removes any whitespaces before or after [INSERT], [NUMERICAL], or any other input form type.

**set\_instructions()**  
Mutator for the instructions variable for the current object - Sets the instruction element of the current object with the current line (minus content tags)

**set\_bnf()**  
Sets BNF mapping object for the current object.

**set\_text()**  
Sets the instruction element of the current object with the current line (minus content tags)

1. **Concept Examples**
   1. **Groups, Pages, Questions**
      1. **Groups** - define a top-level collection of pages
      2. **Pages** – define a single page within a group, which contains a collection of questions, instructions, comments, and forms.
      3. **Questions** – defines a single question within a page, which can contain title, text, input, instruction, and other fields.
      4. **Group, Page, and Question Word Doc Example**:

Groups outlined in blue, Pages outlined in green, Questions outlined in red

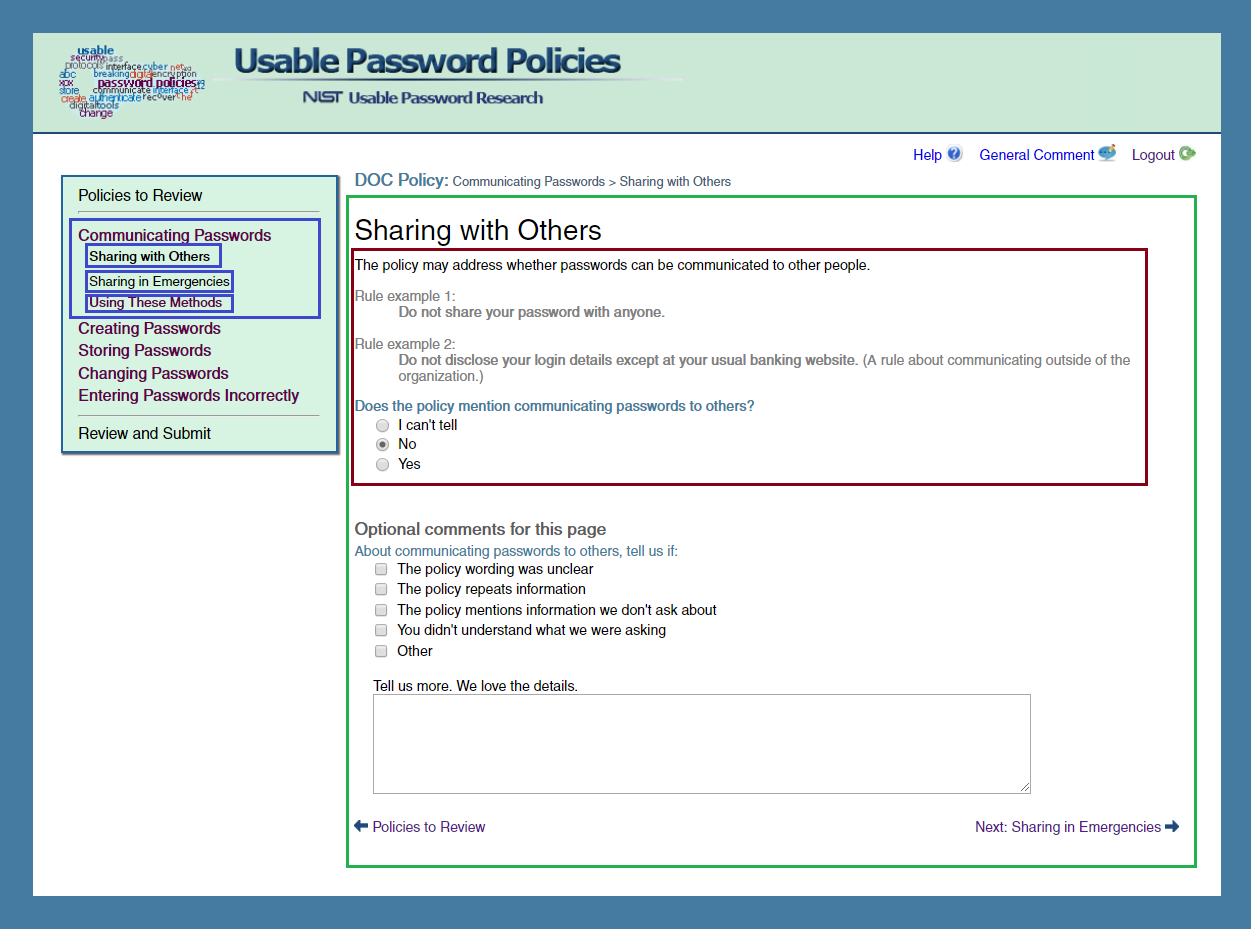


* + 1. **Group, Page, and Question XML Example:**

****

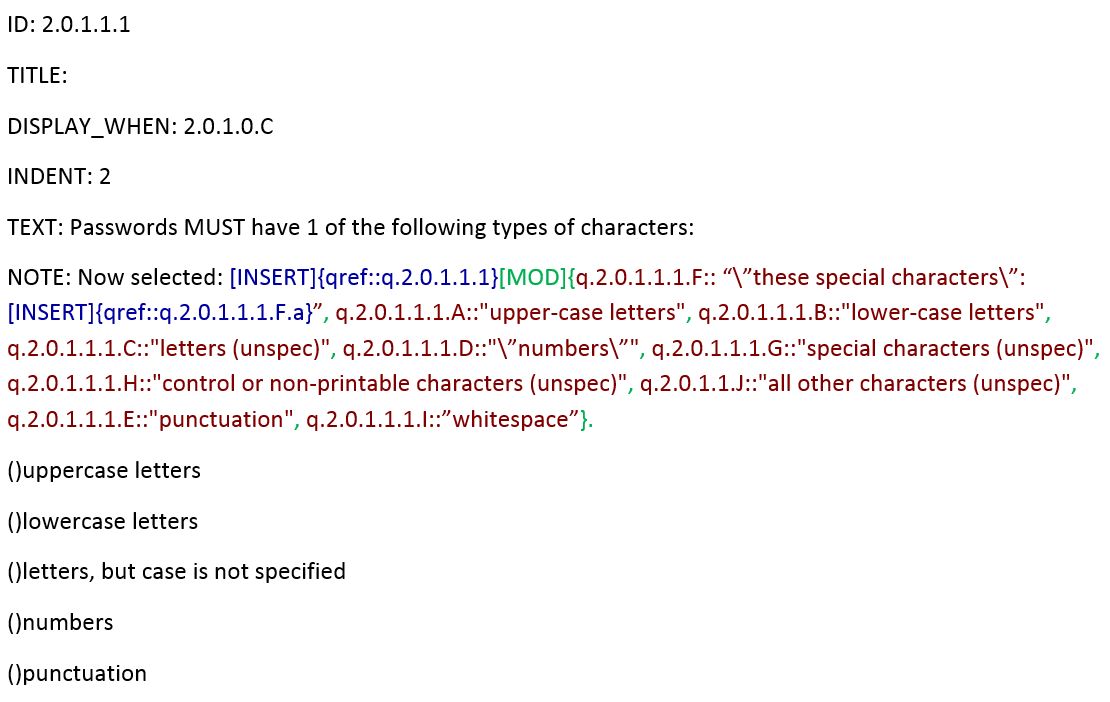
****

* + 1. **Web App Example:**



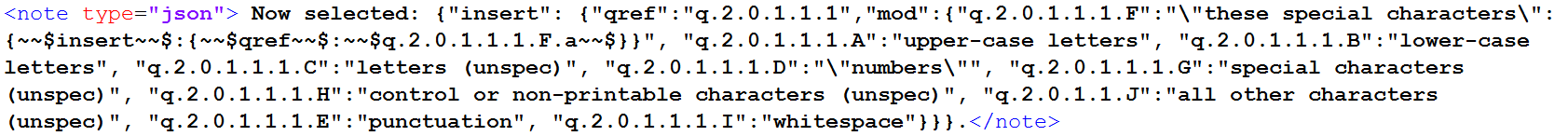
* 1. **Insert, Mod, Regular Categories**
     1. **Insert** – Displays the contents of an input form within the statement.
     2. **Mod** – Allows for the display of nested input forms. All mods must include an insert statement containing an input form within the mod syntax.
     3. **Regular Categories** – All non-insert input form options held within a mod.
     4. **Insert, Mod, Regular Category Word Doc Example:**

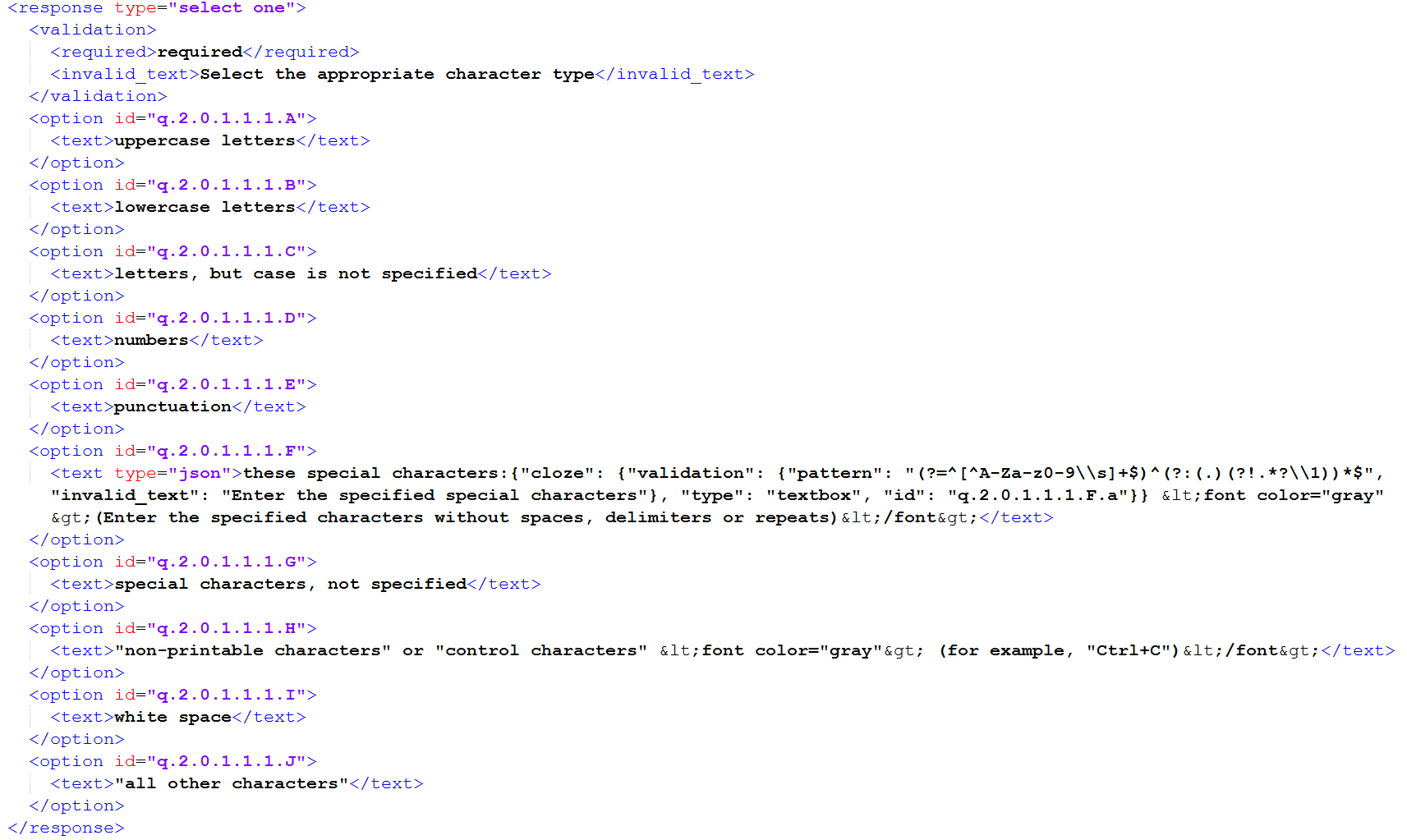
Inserts in blue, Mods in green, Regular Categories in red.

****

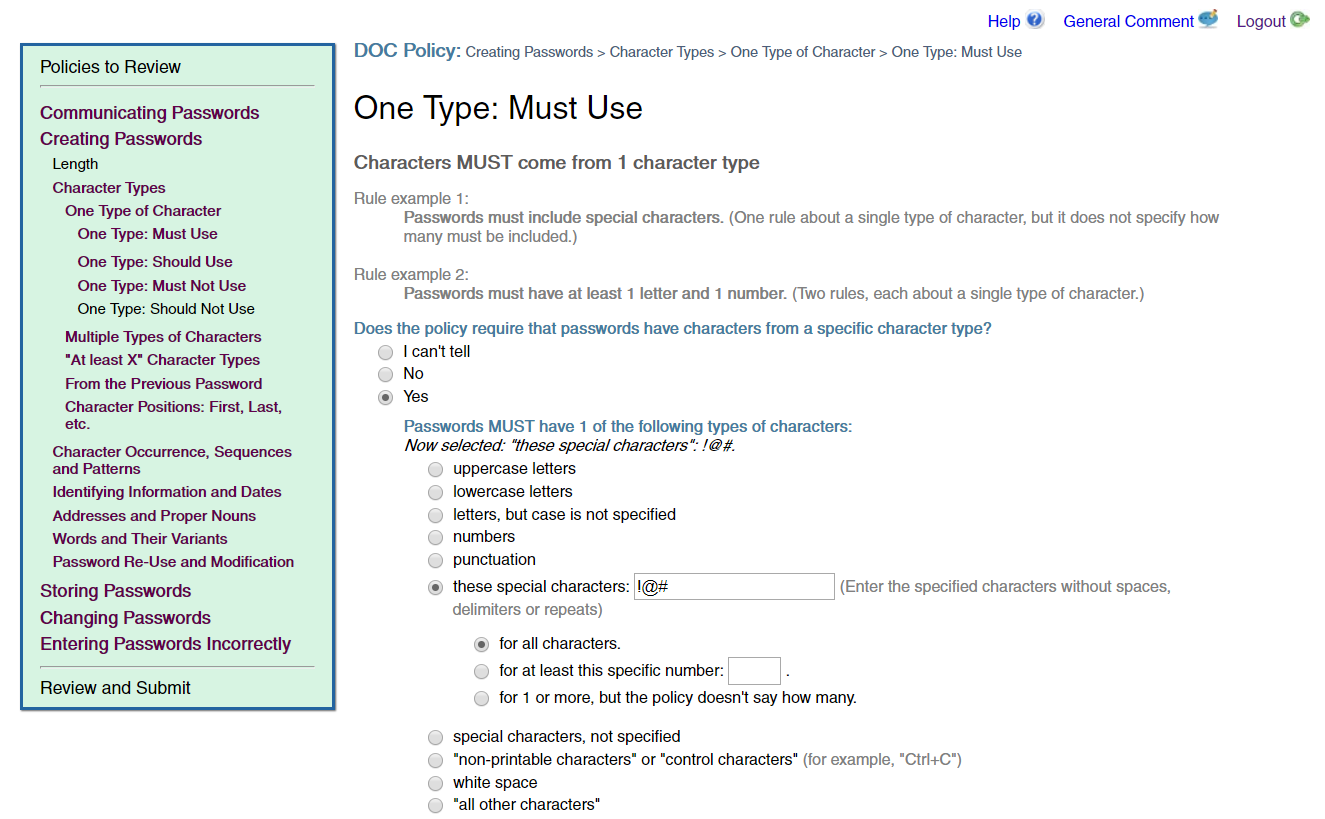
Note: The mod statement in Figure 9 extends from the beginning of [MOD]{ to the ending }. Mod statements are essentially made completely from inner insert statements and regular categories.

* + 1. **Insert, Mod, Regular Category XML Example:**

****

****

* + 1. **Insert, Mod, Regular Category Web App Example:**

****

1. **Element Syntax Instructions**
   1. **General Structure**

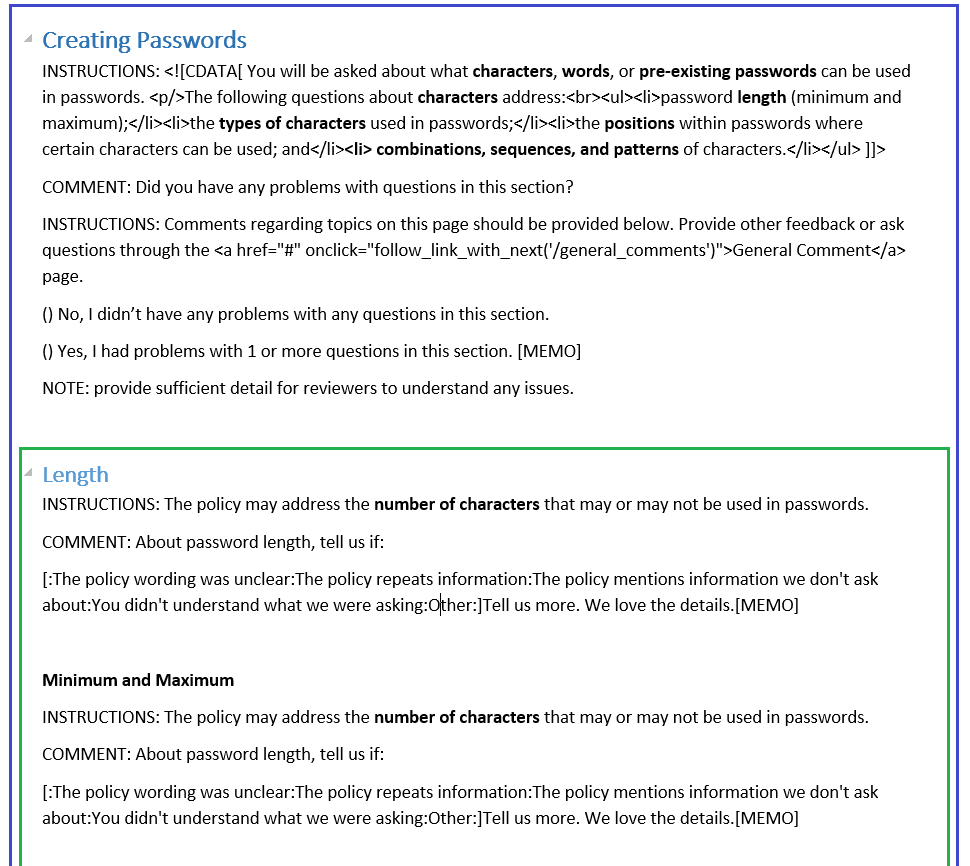
* Survey structure must always go groups->pages->questions
* There must always be at least one page in a group.
* Pages do not necessarily need to contain a question.
  1. **Misc. Formatting**
* Each element must be separated from the previous by a return character.
* Spaces and tabs in the beginning of a line will be ignored.
* Bolded, italicized, and underlined formatting will always be carried over into their HTML representations.
  + This may cause problems in some cases (e.g. page titles are translated into links localhost/<b>page\_1</b>.html).
* HTML tags can be written directly into element text.
  + Ex. <dl><dt>placeholder text</dt></dl> for description lists
  + Ex. <font color=”gray”> for font color assignment for a particular section.
* The parser has been updated to be less sensitive to errant white spaces within a line. However, there may still be some cases where this is the cause of runtime errors. Use properly formatted syntax whenever possible.
* Comments (#thisisacomment) should not affect parser functionality.
  1. **GROUP**
* Must use docx heading styles to differentiate groups.
* Different heading styles determine nested groups.
  + Ex. Heading level n will be translated into a level n group.
* Groups will differentiate themselves automatically from other groups when following a return character. For example, the following would be parsed into two different groups.

# Communicating Passwords 1

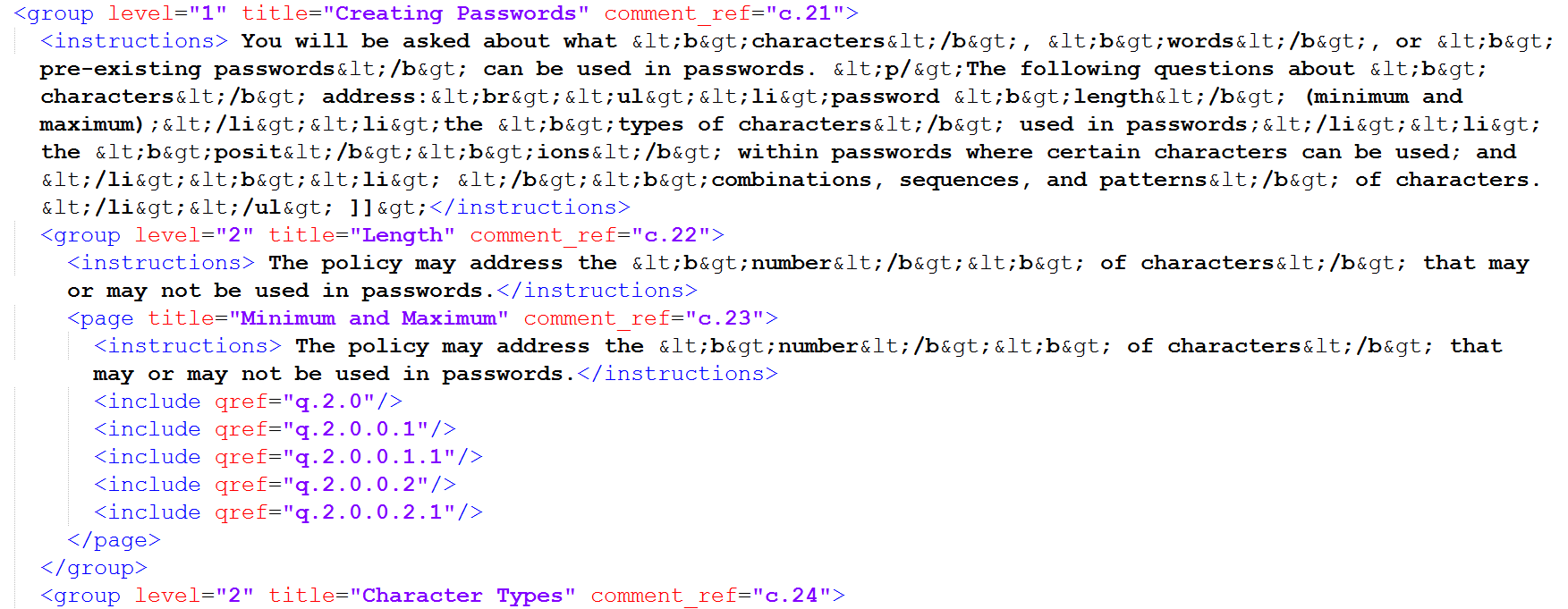
# Communicating Passwords 2

* Optionally, instruction and comment elements may be included immediately following group titles.
* There are unlimited levels of question groupings. No group can contain both a group and page - If the index violates this rule, an exception will be thrown when the application starts which prints information about which group is to blame.
* Group Word Doc Example

Blue lines encapsulate group level 1 (Creating Passwords). Green lines encapsulate group level 2 (Length) and all pages and questions held within. The shown example is only a small excerpt from the group contents, see the ‘Creating Passwords’ group in the example word doc file for full reference text.

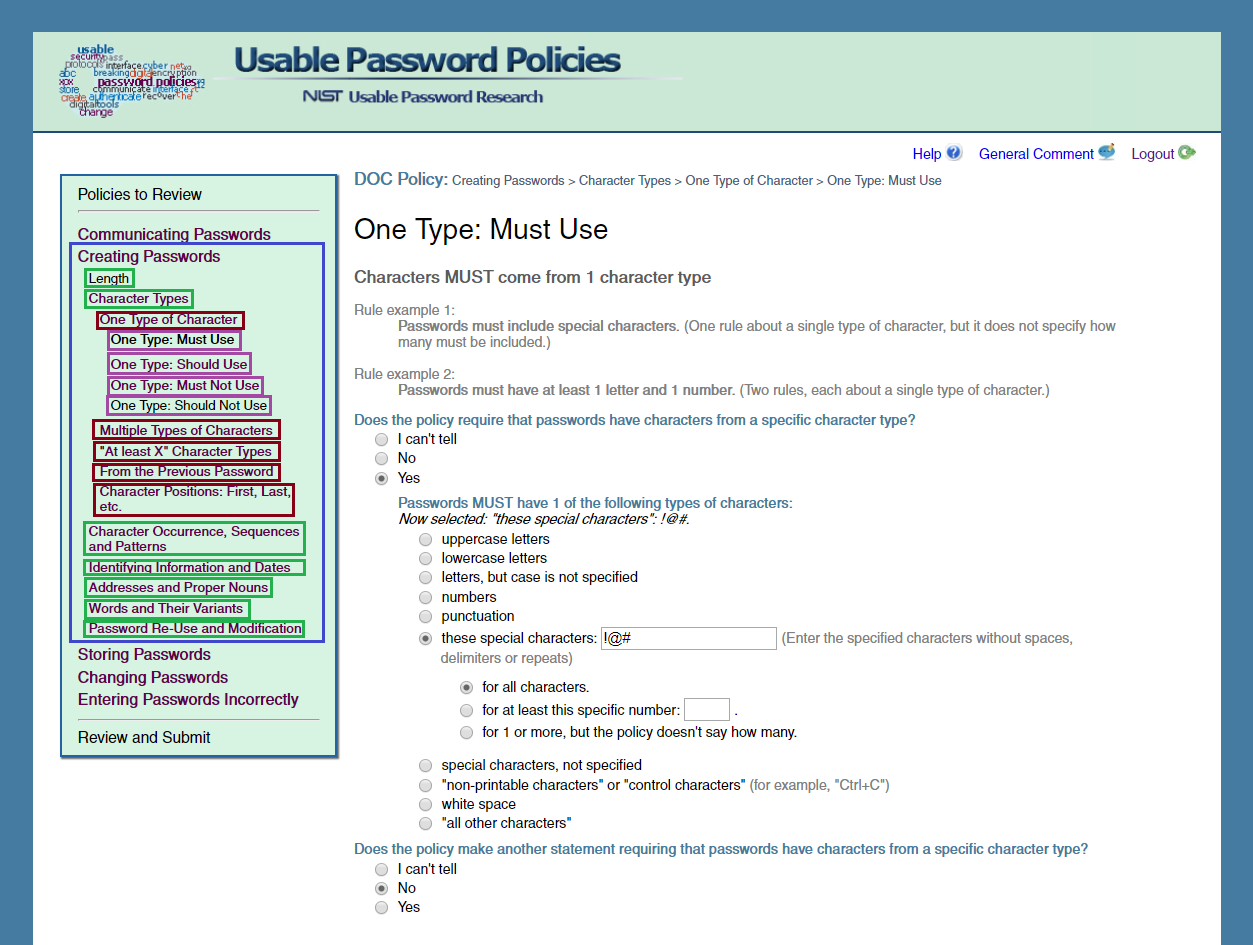


* Group XML Example



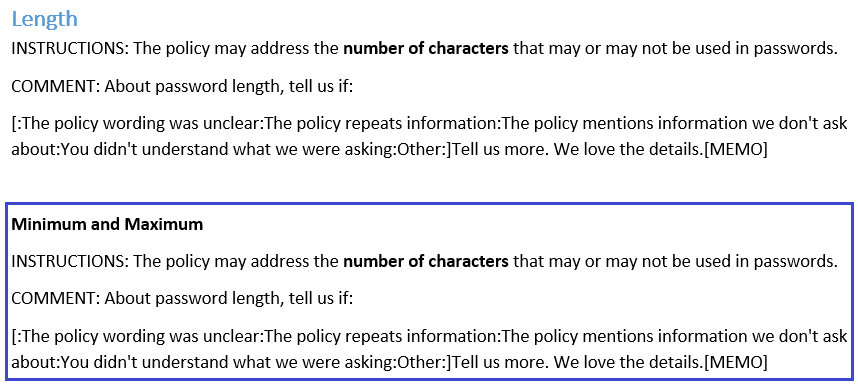
* Group Web App Example

Blue lines encapsulate group level 1 (Creating Passwords). Green lines encapsulate group level 2 (Length) and all pages and questions held within. This particular level 1 group extends down to group 3 and 4, which are included in this example using red and purple lines respectively.



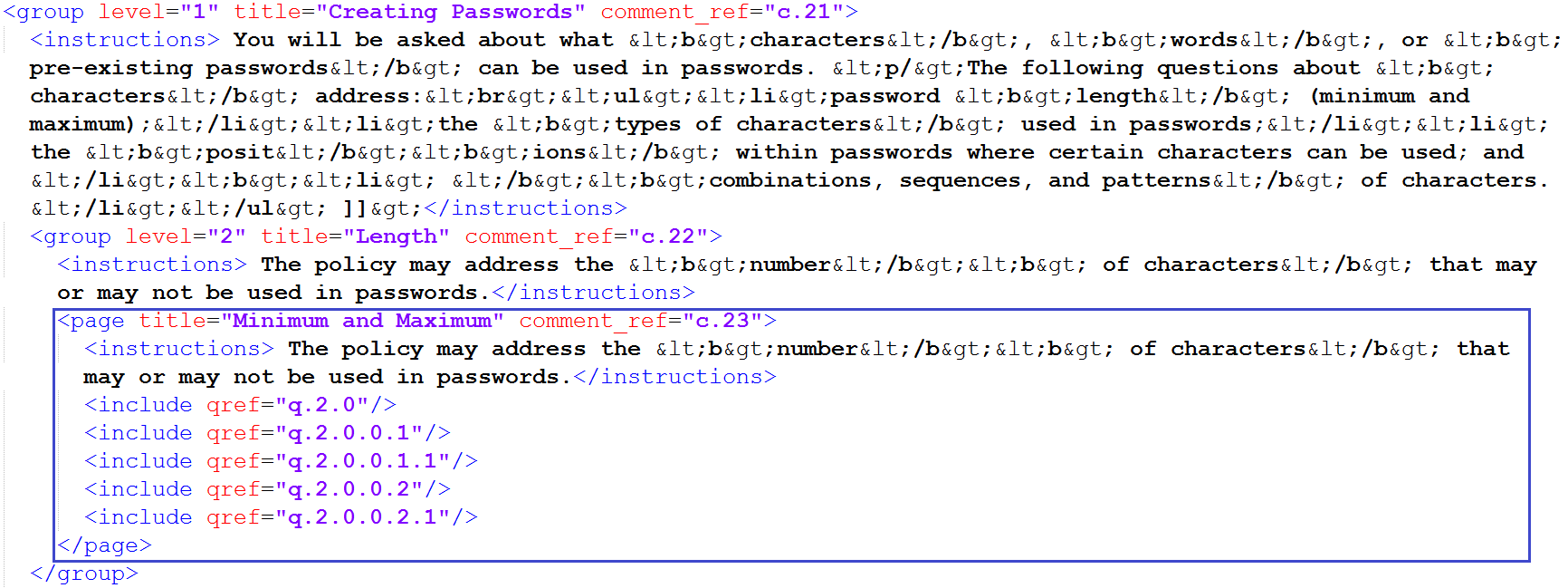
* 1. **PAGE**
* Uses the docx Strong style. Cannot use **bold** in lieu of **Strong** – will result in regular bold text rather than the start of a new page.
* Pages must be encapsulated within a group.
* Like groups, pages can also have page specific instructions and comments.
* Page Word Doc Example

Page syntax is in blue.



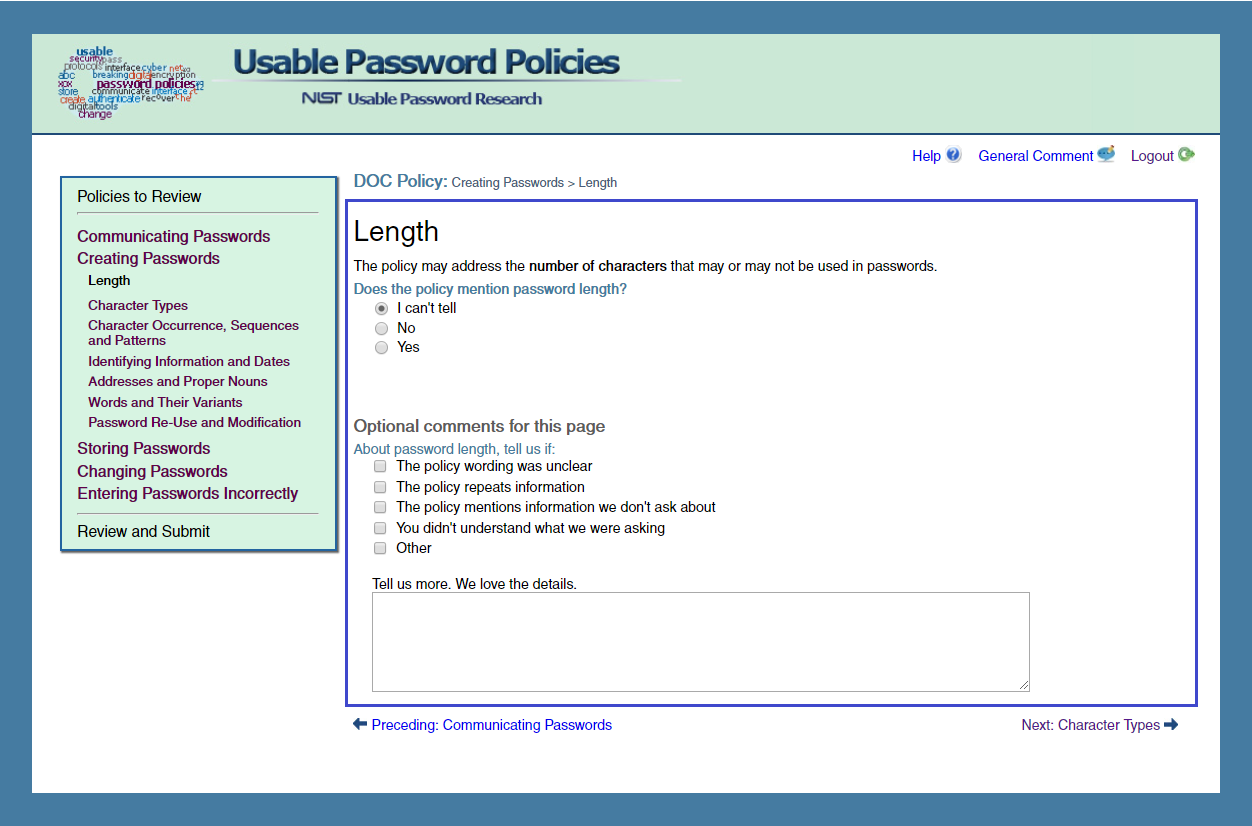
* Page XML Example

Page syntax is in blue.



* Page Web App Example

Pages are represented by the blue rectangle.



* 1. **QUESTION**
* Questions are always encapsulated within a page.
* A question starts with a line beginning with ‘ID:’ and ends with an empty line.
* Are only required to have an ID, all other question elements are optional.
* See syntax details for all question elements in 3.5.1-3.5.21
* Example Syntax:

ID: <question id>

TITLE: <question title>

TEXT: <question text>

NOTE: <question notes>

DISPLAY\_WHEN: <display\_when attribute>

DISPLAY\_WHERE: <display\_where attribute>

OPTION or TEXT (this line can be repeated as many times as wanted)

COMMENT: <additional comment question text>

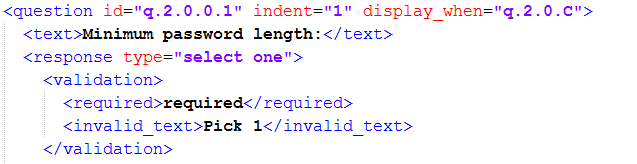
OPTION or TEXT (this line can be repeated as many times as wanted)

<Blank line>

* + 1. **ID**
* Must start with ‘ID:’
* IDs must be a number optionally followed by a point and another number and so on (‘2’, ‘2.1’, ‘2.1.1.1.1’, ‘100.123.1000.1’). These do not have to correspond with the page or group level number the question is encapsulated by.
* All question IDs should be unique. Having duplicate IDs will not cause a runtime error in the word doc parser, but may cause an error with the web parser.
* The parser is not sensitive to the number sequence, so IDs can appear in any order, as long as they are unique. For example, a page can include questions with the IDs 1.02, 5.2.2.1, 5.292.1, and 2.11 in that order.
* Question ID Word Doc Example



* Question ID XML Example



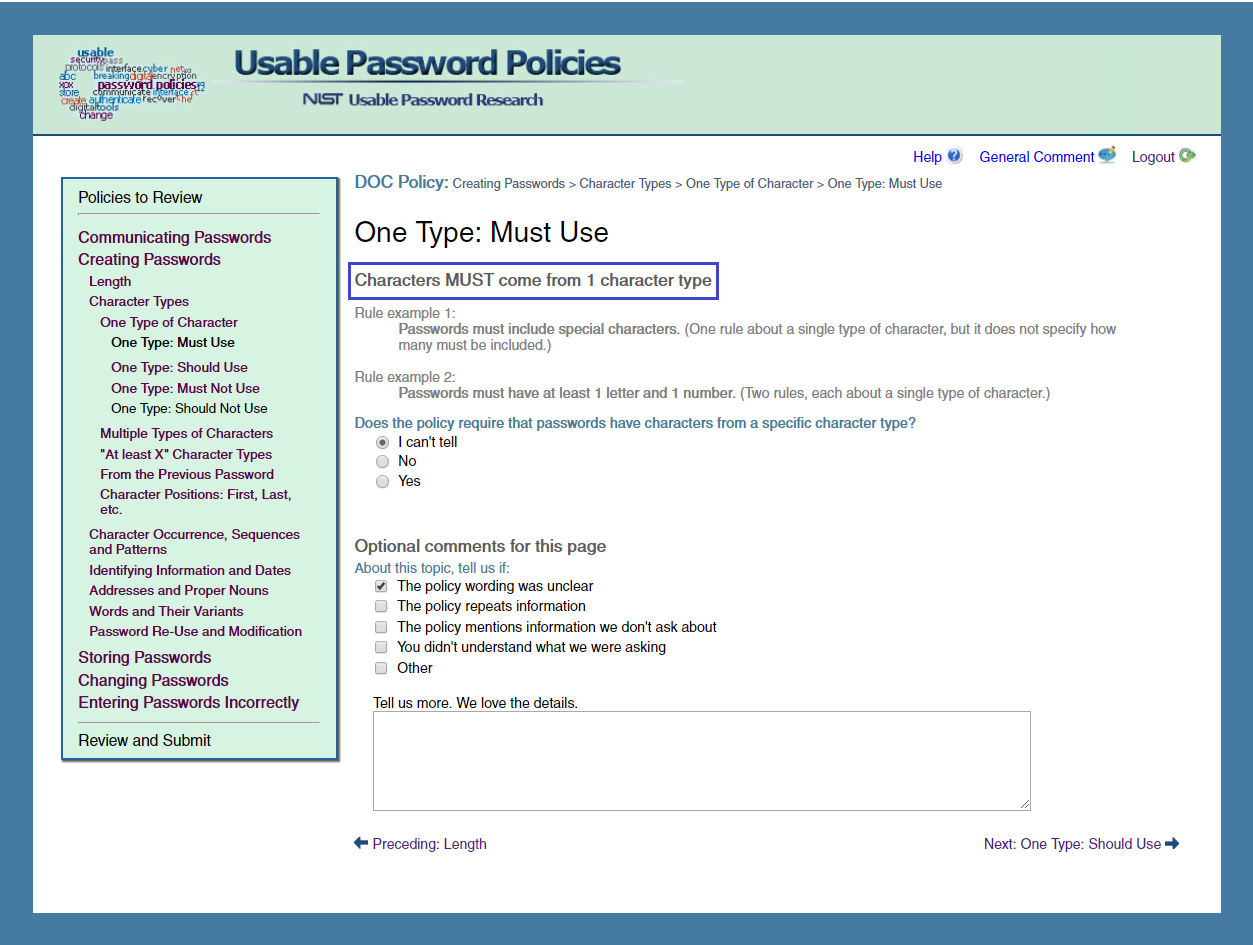
* + 1. **TITLE**
* Must start with ‘TITLE:’
* Page level titles currently cannot use formatting tags. Will result in plaintext tags displayed.
* Title Word Doc Example



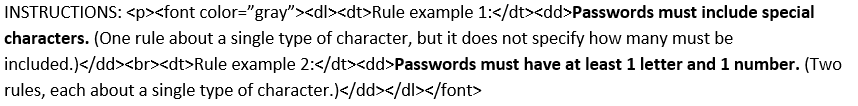
* Title XML Example



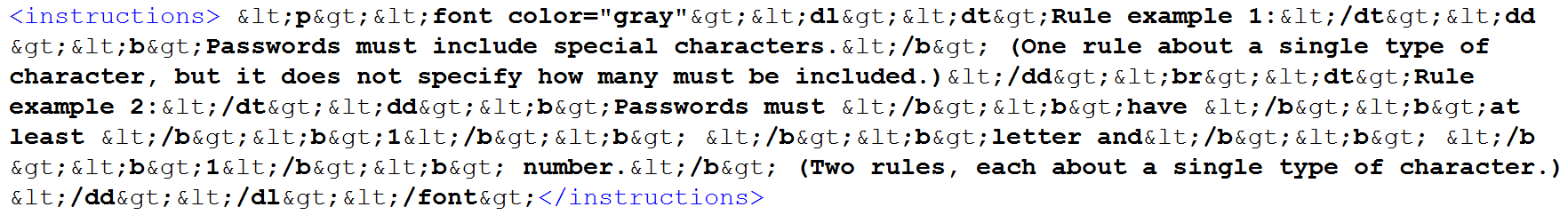
* Title Web App Example



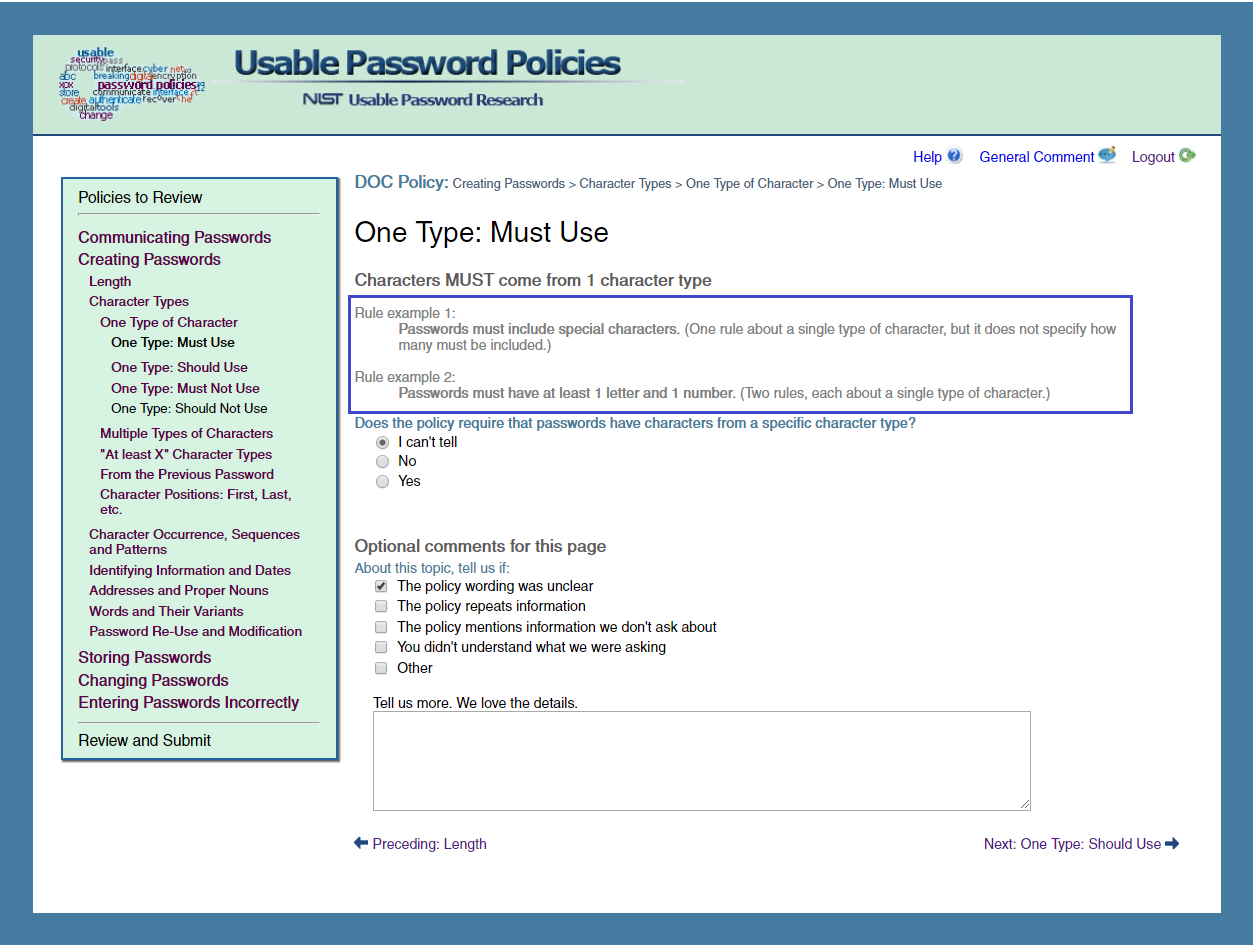
* + 1. **INSTRUCTIONS**
* Must be started with the ‘INSTRUCTIONS:’ marker.
* Instructions Word Doc Example



* Instructions XML Example



* Instructions Web App Example



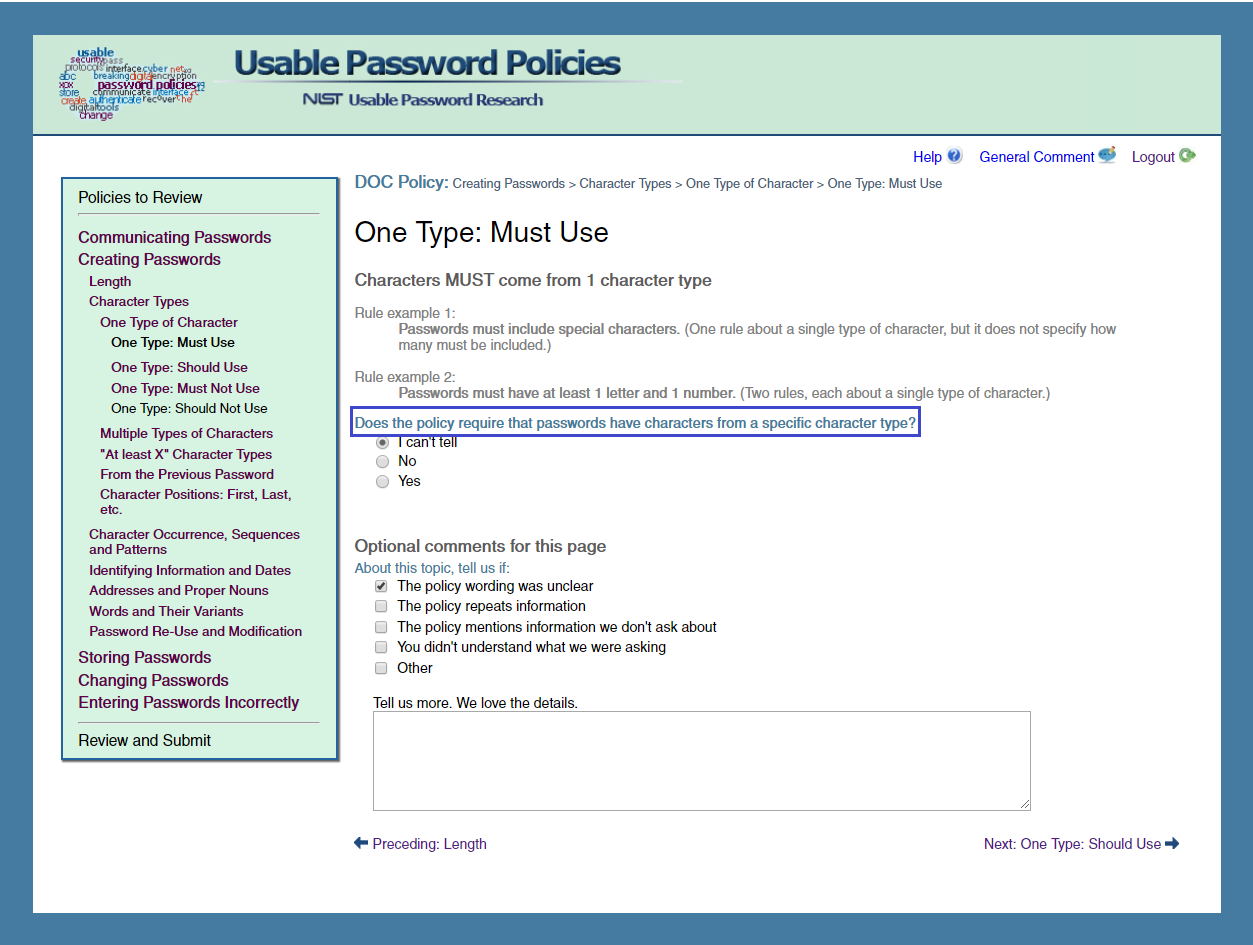
* + 1. **TEXT**
* Must be started with the ‘TEXT:’ marker
* Text Word Doc Example



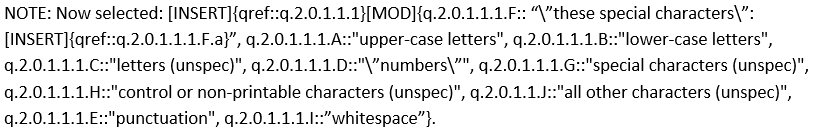
* Text XML Example



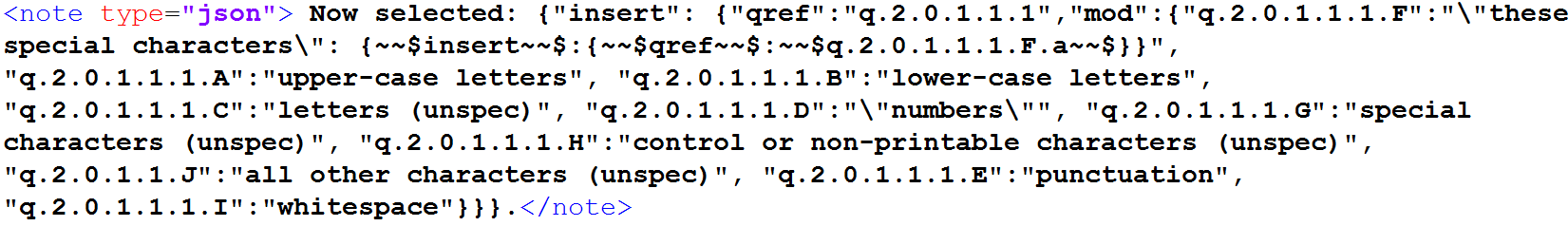
* Text Web App Example



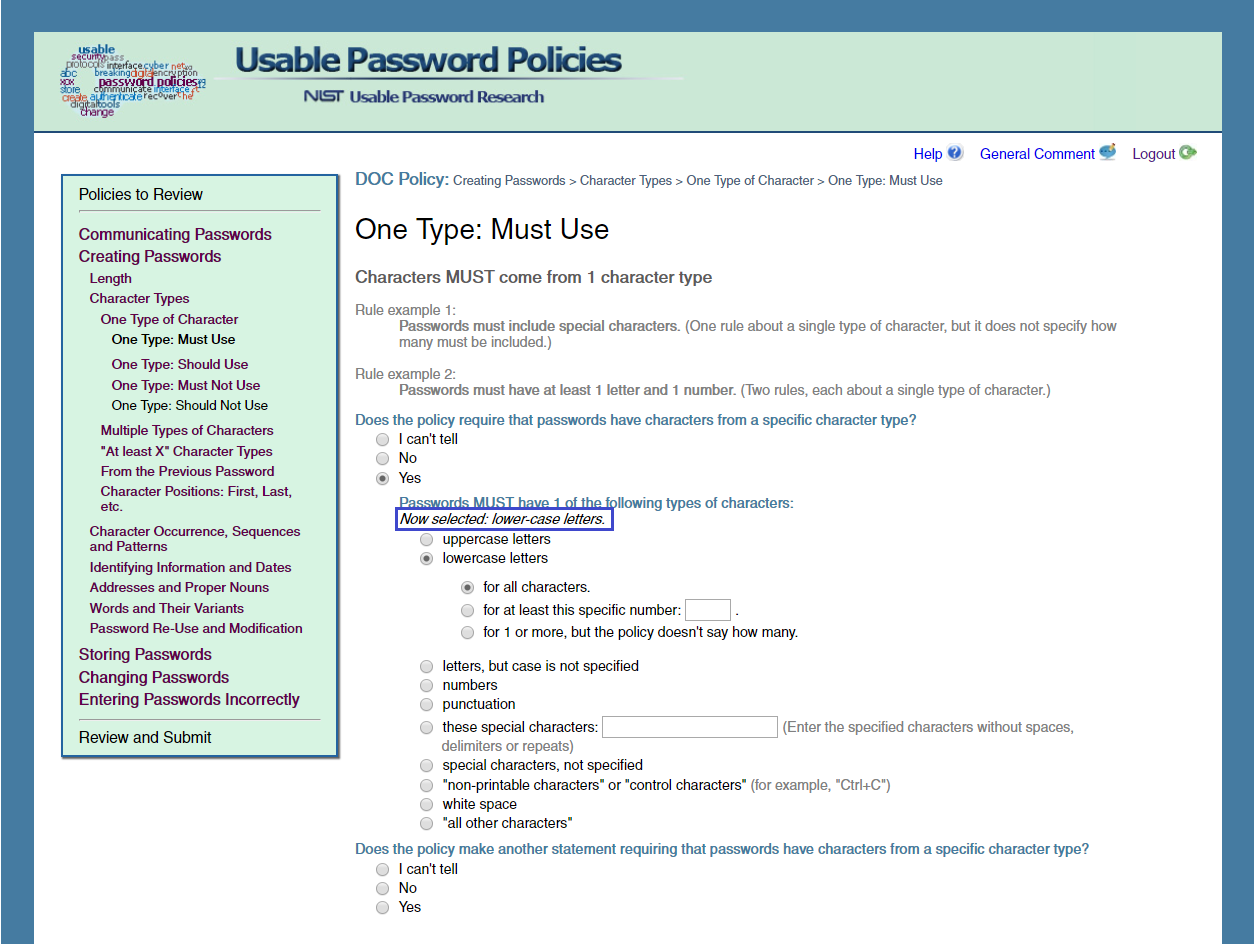
* + 1. **NOTE**
* Must be started with the ‘NOTE:’ marker
* Note Word Doc Example



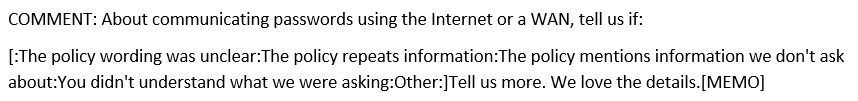
* Note XML Example



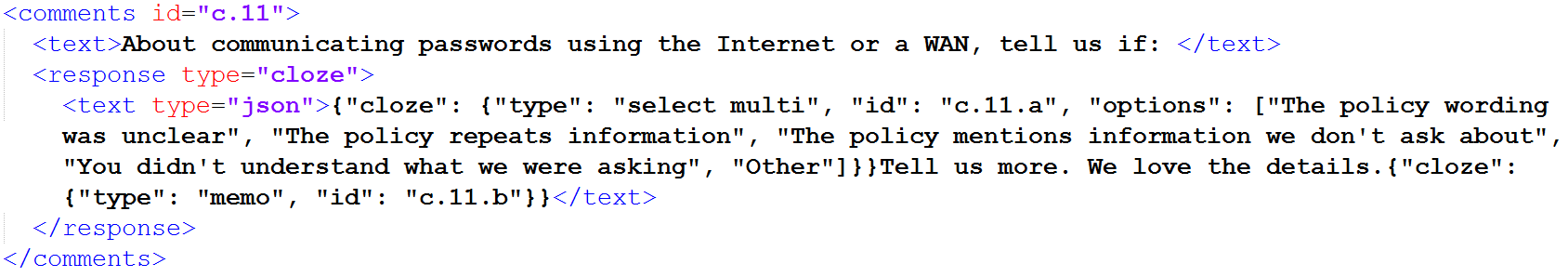
* Note Web App Example



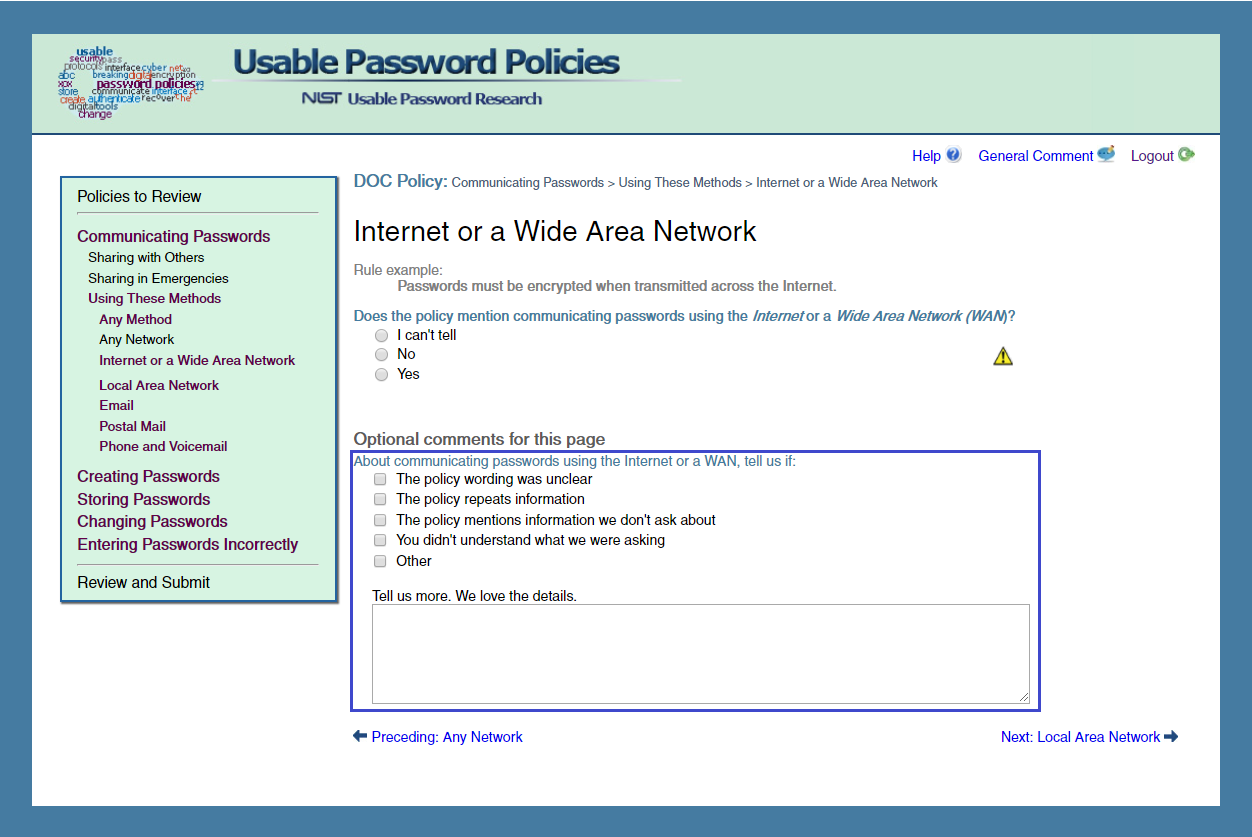
* + 1. **COMMENT**
* Must be started with the ‘COMMENT:’ marker
* Comments will be sectioned off into a separate section in the XML file.
* Comment Word Doc Example



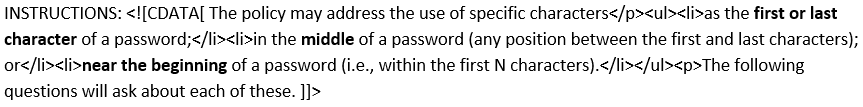
* Comment XML Example



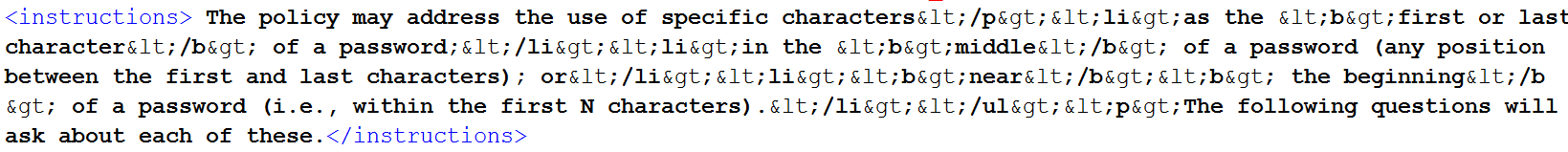
* Comment Web App Example



* + 1. **CDATA**
* For use when using formatting tags not covered in the CSS file on server (style.css in /policy\_study/static)
* Segment to be surrounded in CDATA should be prefaced by ‘<![CDATA[‘ and ended with ‘]]>’
* CDATA formatting tags are automatically removed by the parser as the input word file is processed.
* CDATA Word Doc Example



* CDATA XML Example



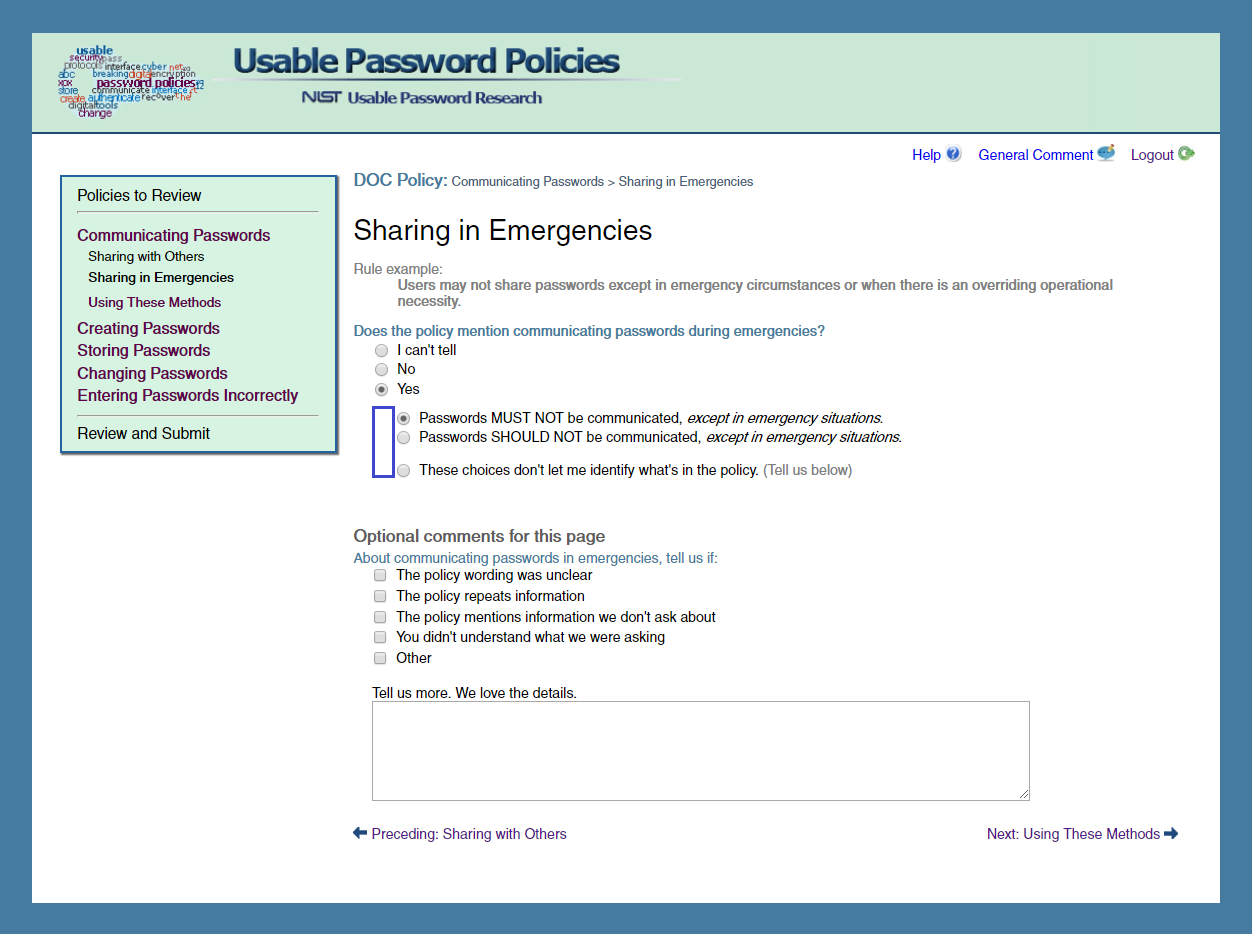
* + 1. **INDENT**
* Use when indenting an entire question is needed.
* Put syntax directly after question ID line
* Must be started with ‘INDENT:’ marker.
* Indent size is calculated by multiplying the value of <indent\_px> found in the template file by the value following the ‘INDENT:’ marker.
* Indent Word Doc Example



* Indent XML Example



* Indent Web App Example



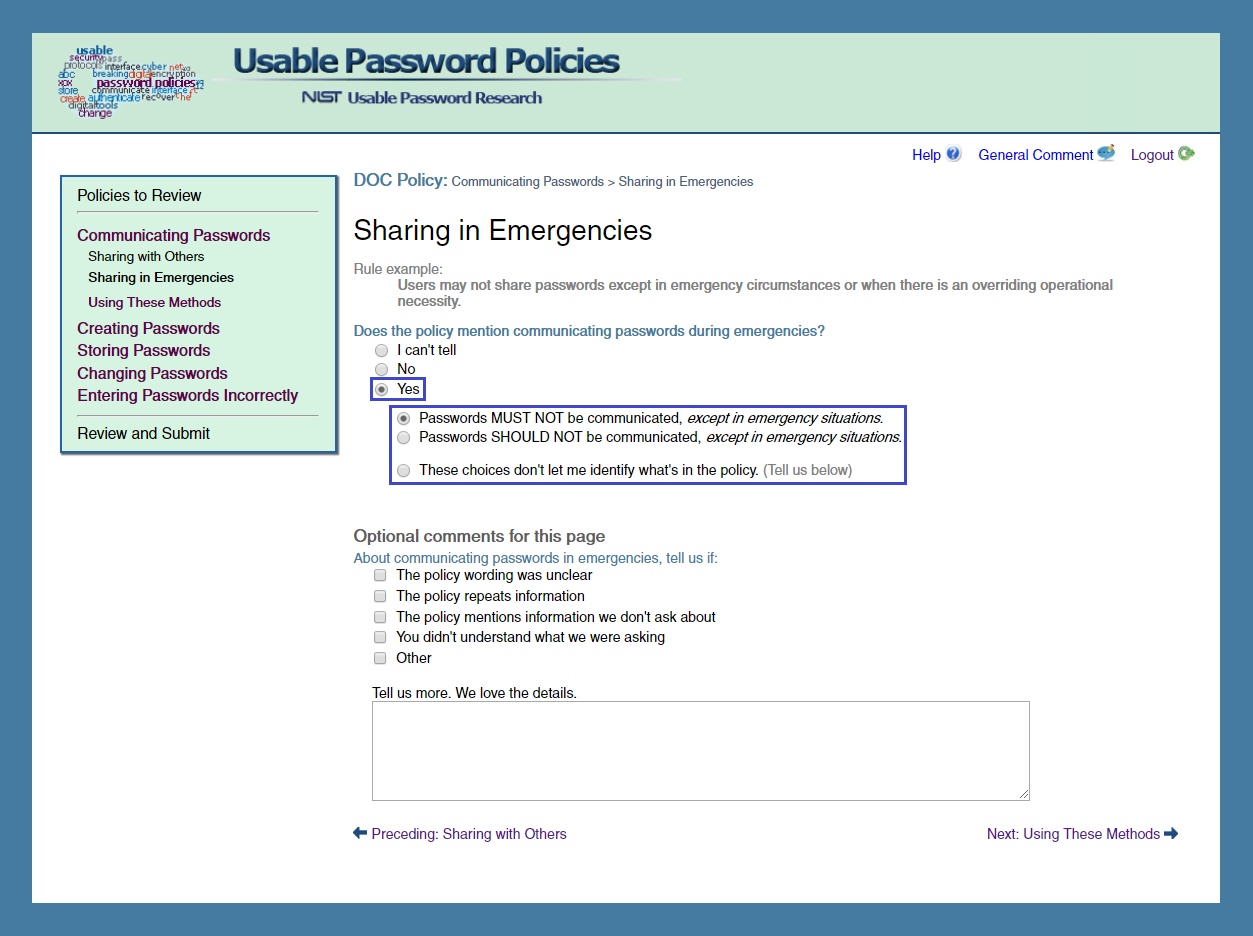
* + 1. **DISPLAY\_WHEN**
* DISPLAY\_WHEN is used if the current question is to be displayed only when a particular OPTION has been selected in another question.
* Must be started with ‘DISPLAY\_WHEN:’ marker.
* The OPTION in the below example is ‘C’, i.e. when the third option in question 1.2.1 has been selected.
* Display\_When Word Doc Example



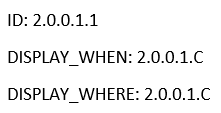
* Display\_When XML Example



* Display\_When Web App Example



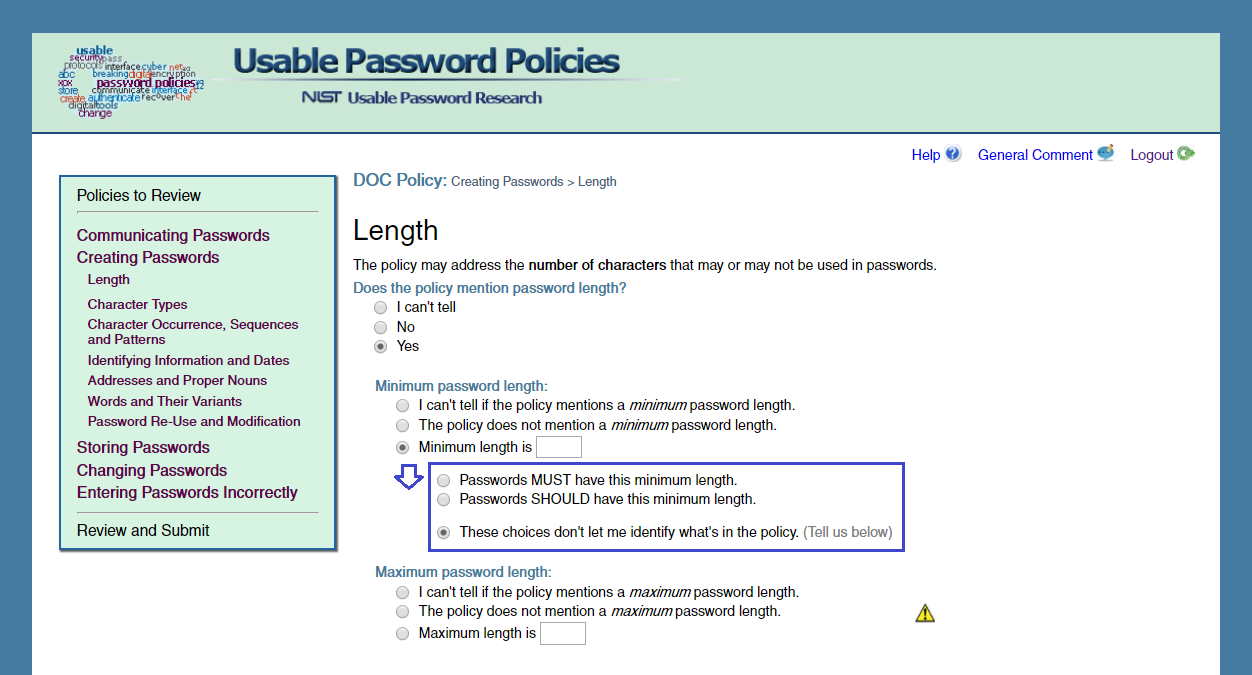
* + 1. **DISPLAY\_WHERE**
* Determines where the sub-question will appear assuming it is dependent on a response. Can be specified to display below the response or after the main question.
* Must be started with ‘DISPLAY\_WHERE:’ marker.
* Display\_Where Word Doc Example



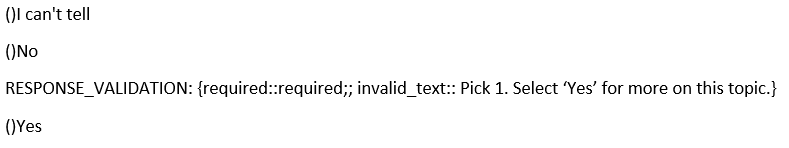
* Display\_Where XML Example



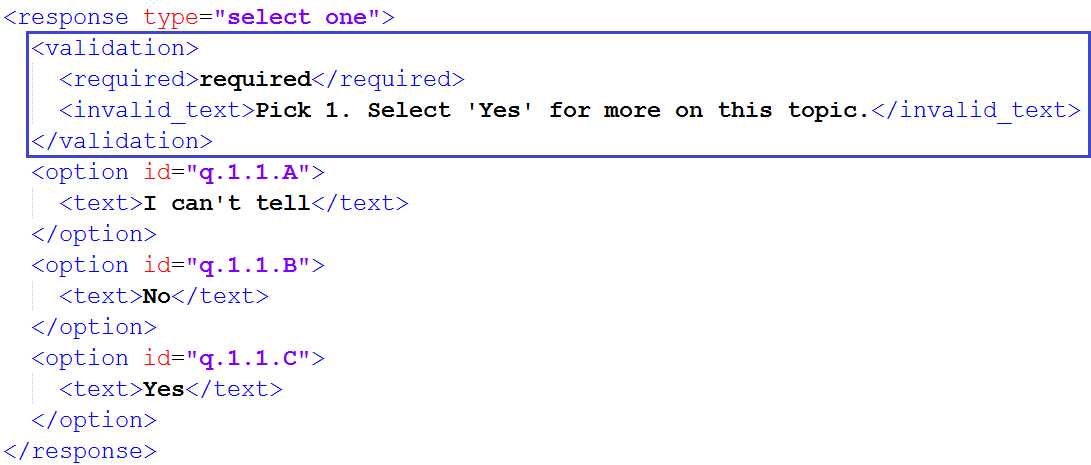
* Display\_Where Web App Example



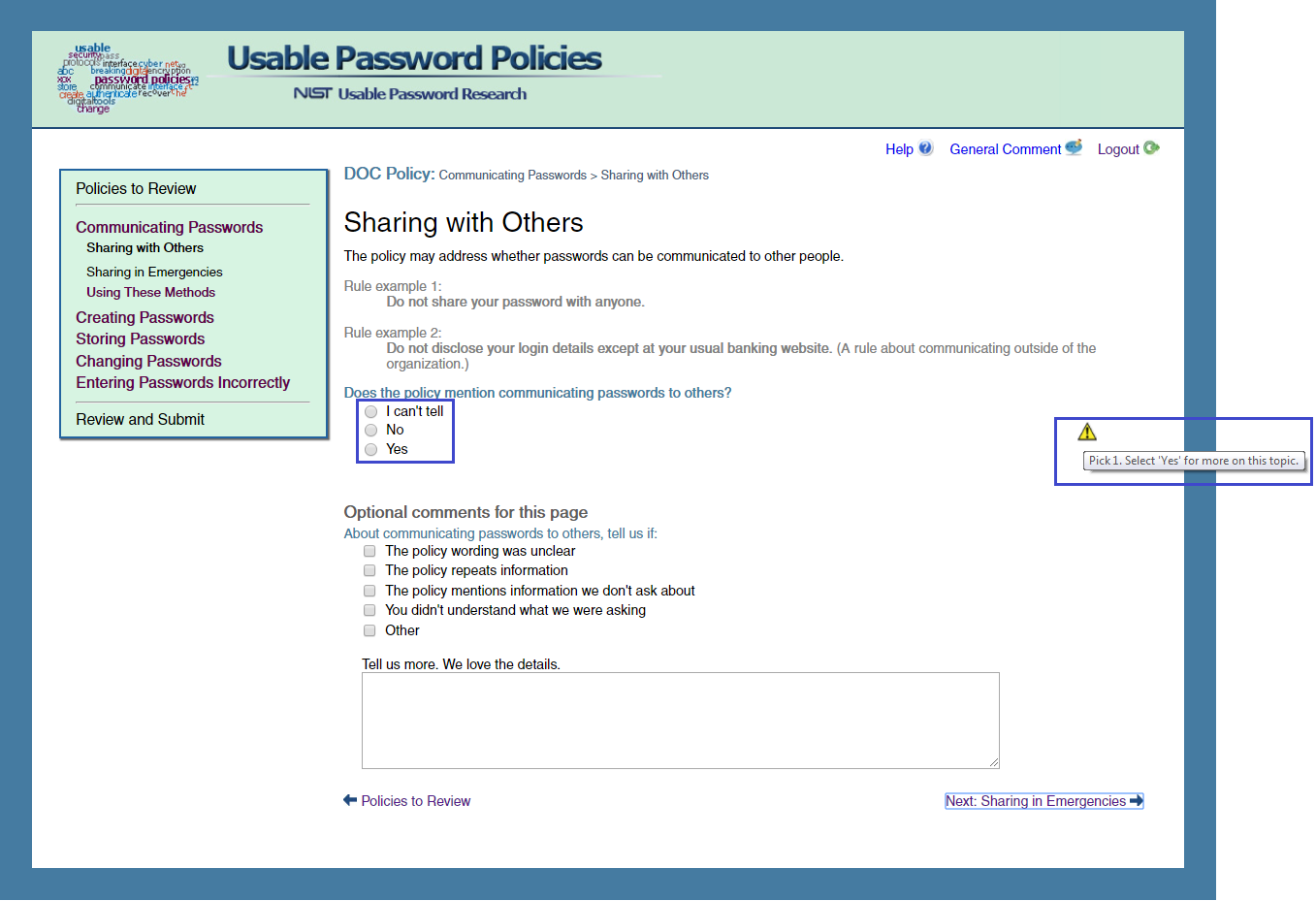
* + 1. **RESPONSE\_VALIDATION**
* Checks to see that a response element (i.e. select box option), has at least one item indicated.
* Must be started with ‘RESPONSE\_VALIDATION:’ marker.
* Dictionary based, with :: and ;; used as delimiters.
* By default, only select multi (checkbox) responses are not required. To make these required, use syntax: RESPONSE\_VALIDATION: {required::required}
* In order to mark an otherwise required by default response as optional, use syntax RESPONSE\_VALIDATION: {required::}.
* Response\_Validation Word Doc Example



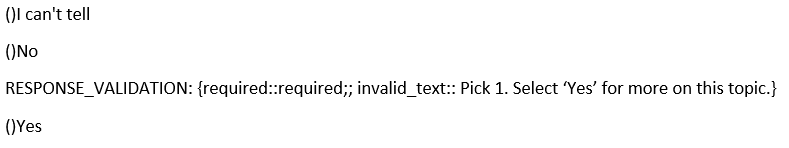
* Response\_Validation XML Example



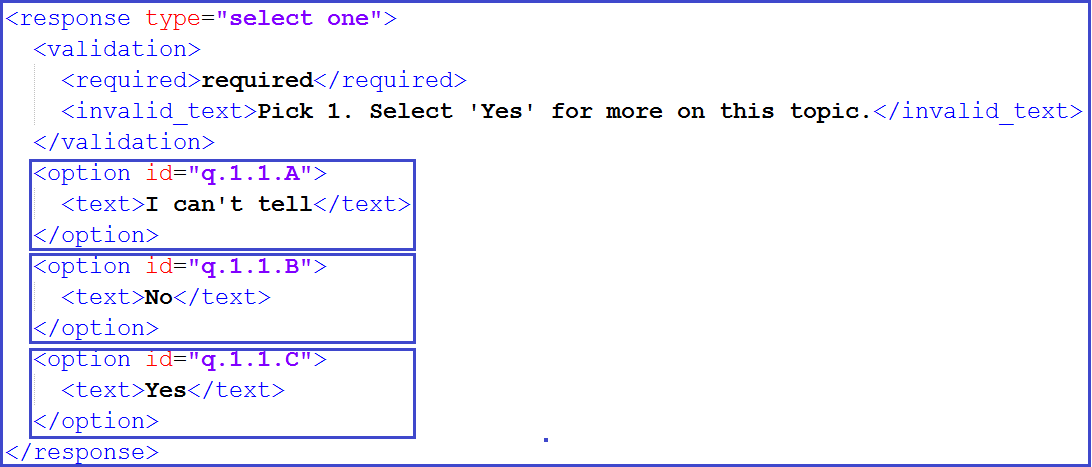
* Response\_Validation Web App Example



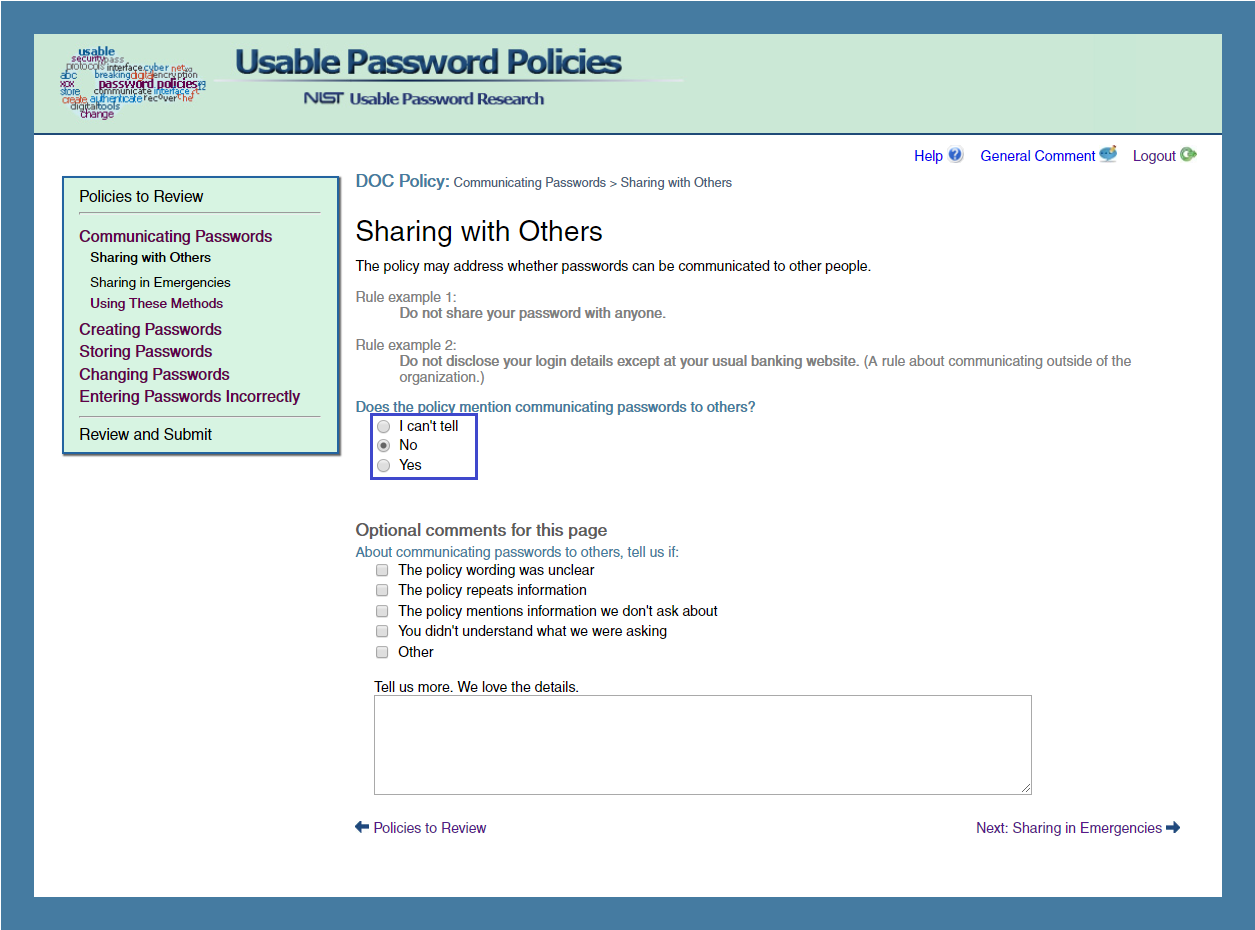
* + 1. **() – Radio Button**
* Allows for radio buttons in a question response.
* Must be started with ‘()’
* Text immediately following ‘()’ will be used as the radio button label.
* Can be used with the response validation element to customize required answers.
* Radio Button Word Doc Example



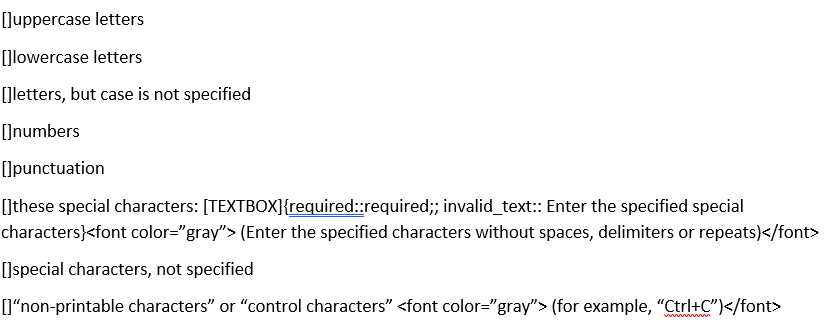
* Radio Button XML Example



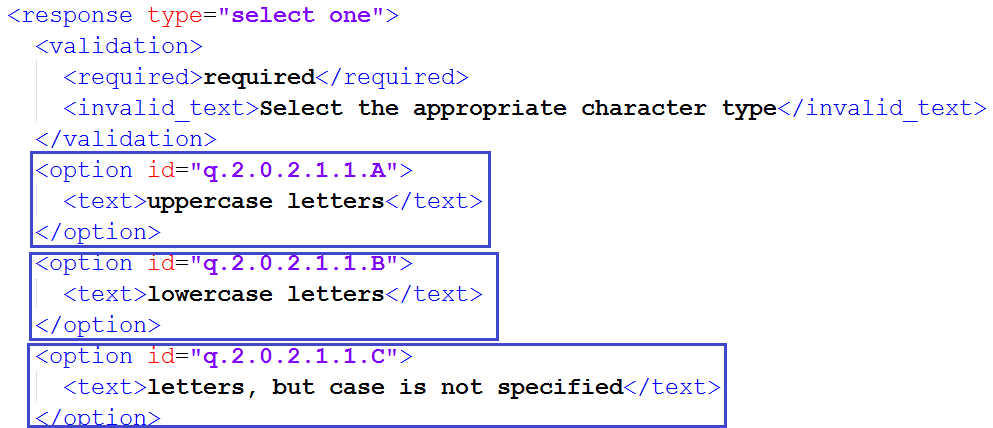
* Radio Button Web App Example



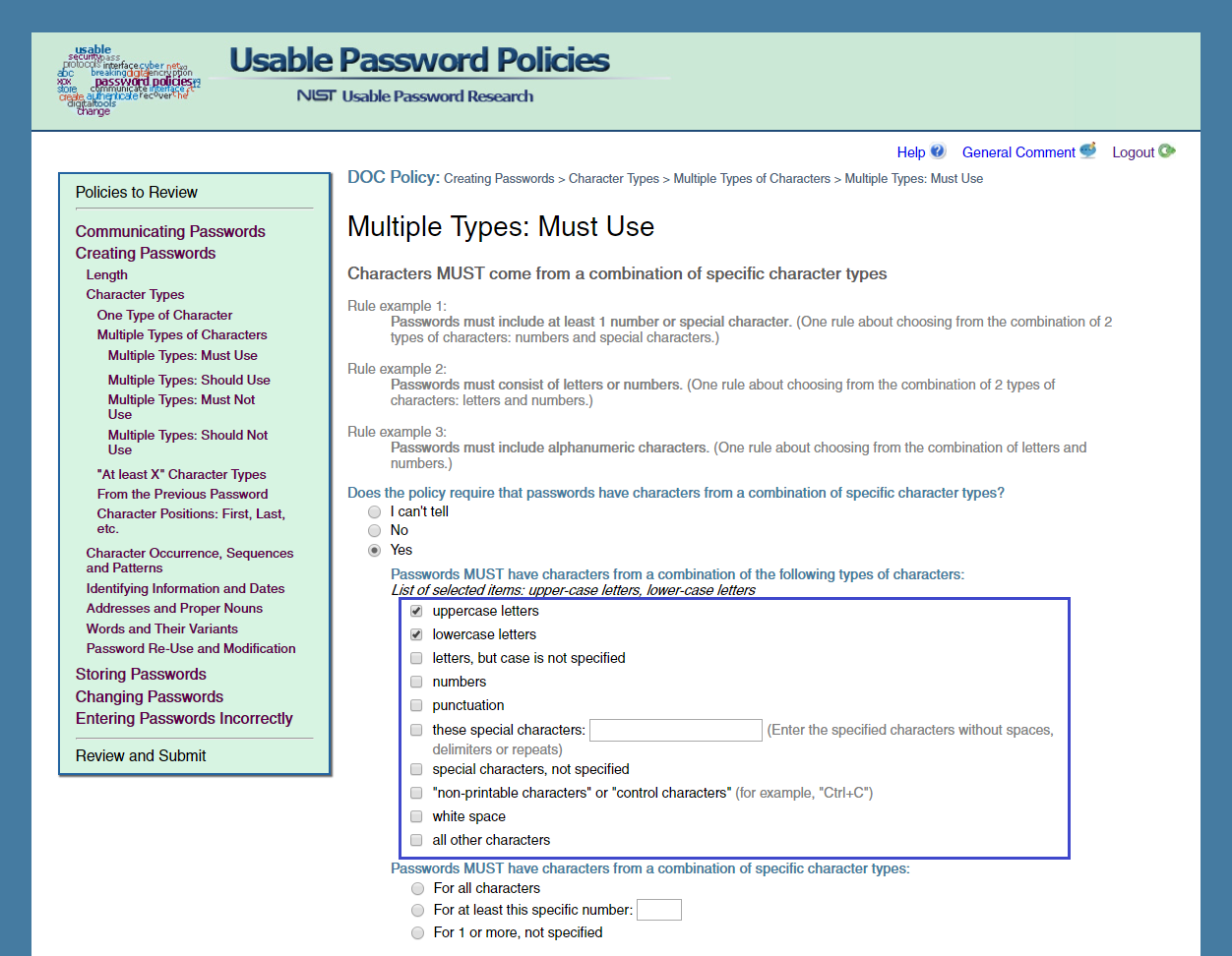
* + 1. **[] – Select Box Option**
* Allows for checkbox buttons in a question response.
* Must be started with ‘[]’
* Text immediately following ‘[]’ will be used as the radio button label.
* Can be used with the response validation element to customize required answers.
* Select Box Word Doc Example



* Select Box XML Example



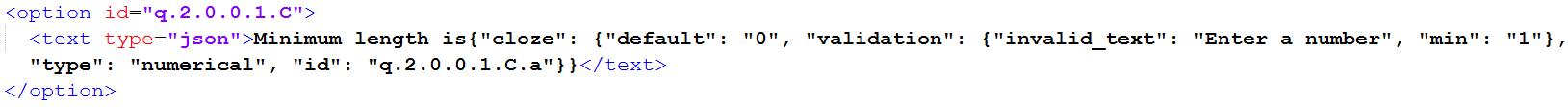
* Select Box Web App Example



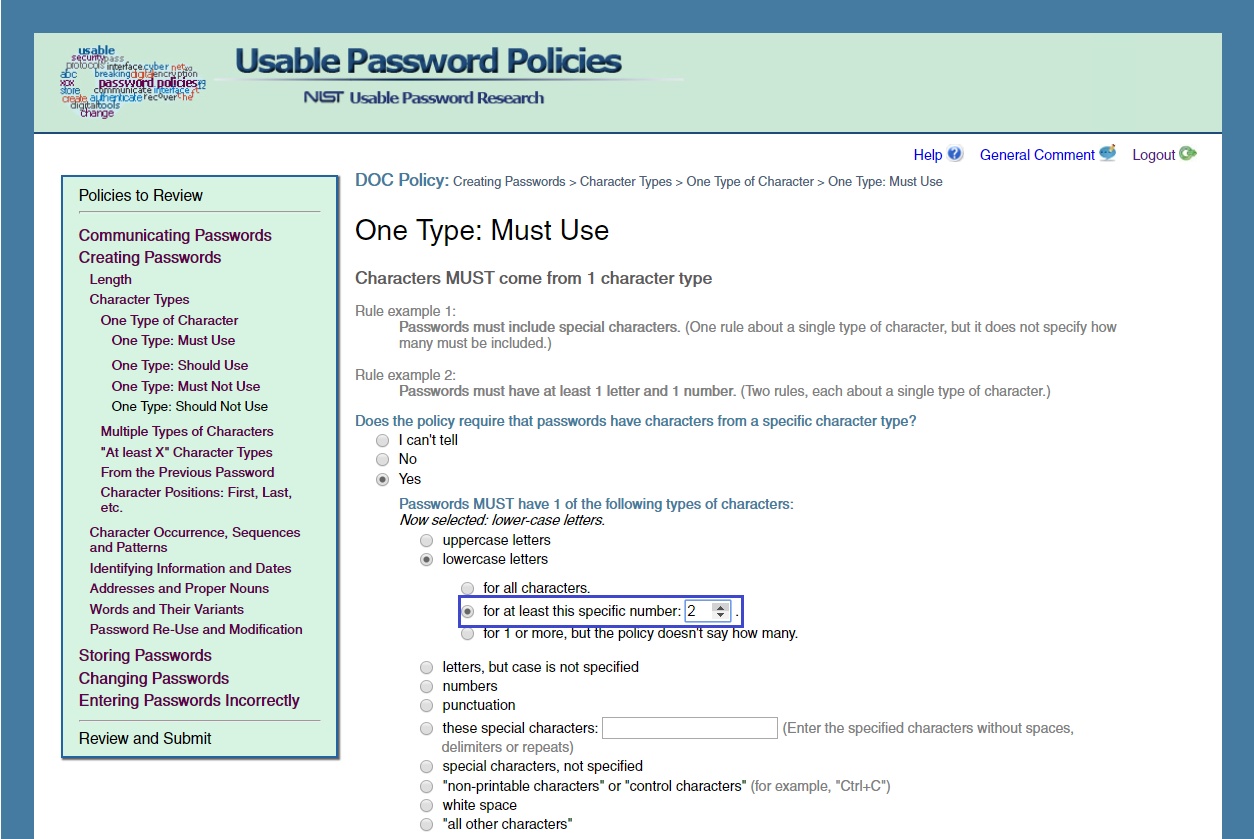
* + 1. **NUMERICAL**
* Input form limited to numbers.
* Must be started with the ‘[NUMERICAL]’ marker.
* Can be put anywhere within a line.
* Numerical Word Doc Example



* Numerical XML Example



* Numerical Web App Example

****

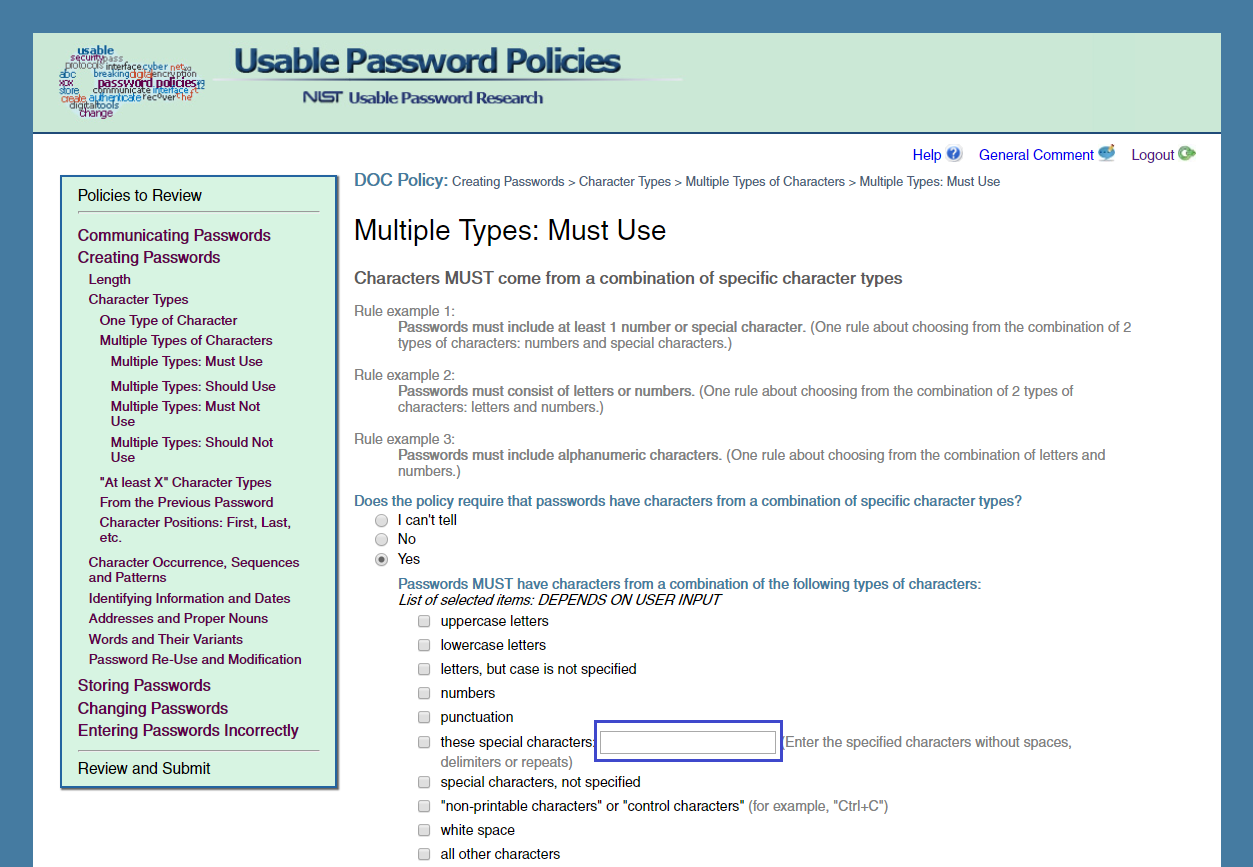
* + 1. **TEXTBOX**
* Single line response form
* Represented by ‘[TEXTBOX]’
* Can be put anywhere within a line.
* Textbox Word Doc Example



* Textbox XML Example



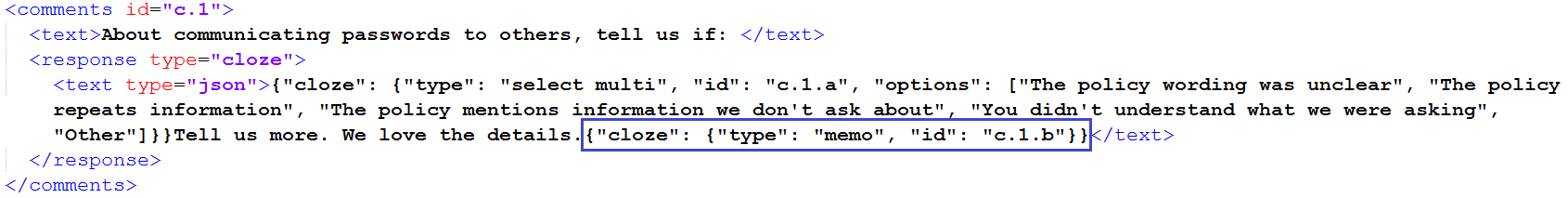
* Textbox Web App Example



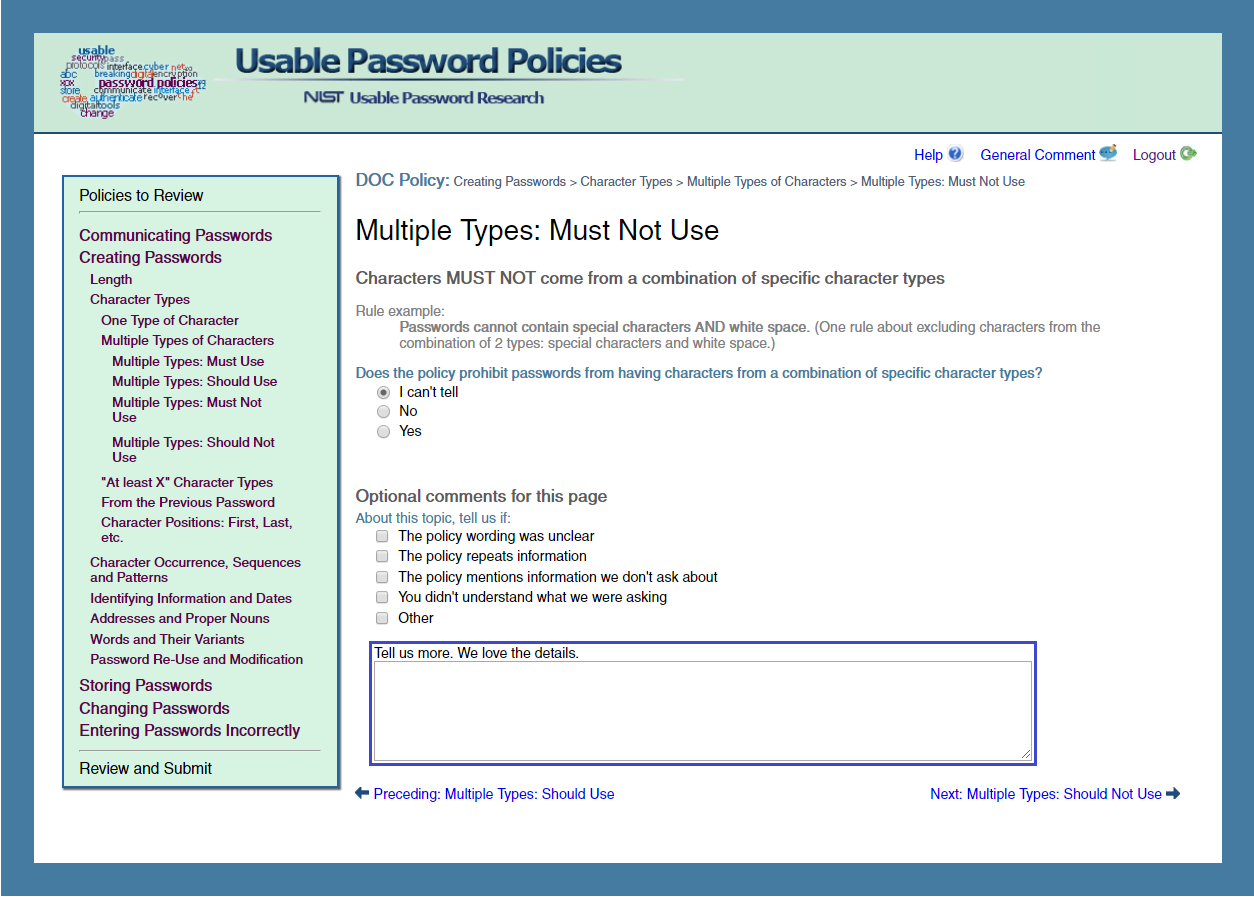
* + 1. **MEMO**
* Multi line response form used for comment forms.
* Represented by ‘[MEMO]’
* Can be put anywhere within a line.
* Memo Word Doc Example



* Memo XML Example



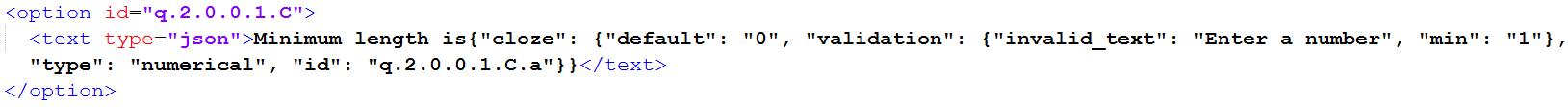
* Memo Web App Example



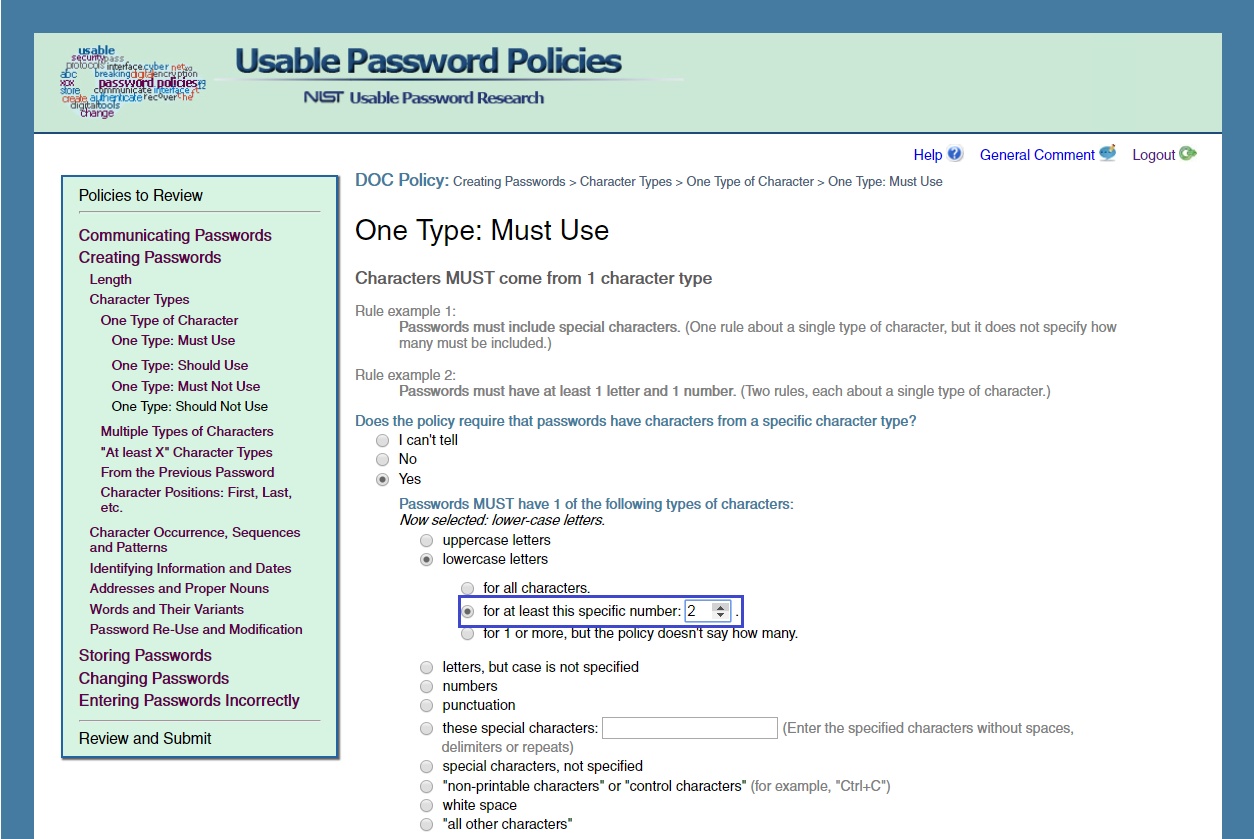
* + 1. **CLOZE**
* For Any combination of input forms, i.e. Having a radio button ‘()’ option with a required number ‘[NUMERICAL] field.
* See Questionnaire Tag Description for detailed syntax.
* Cloze Word Doc Example



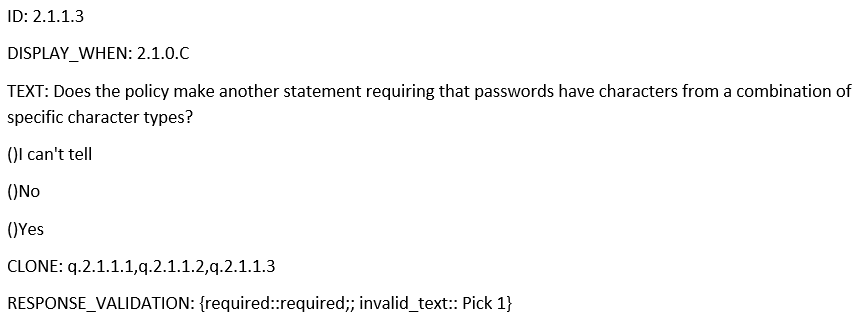
* Cloze XML Example



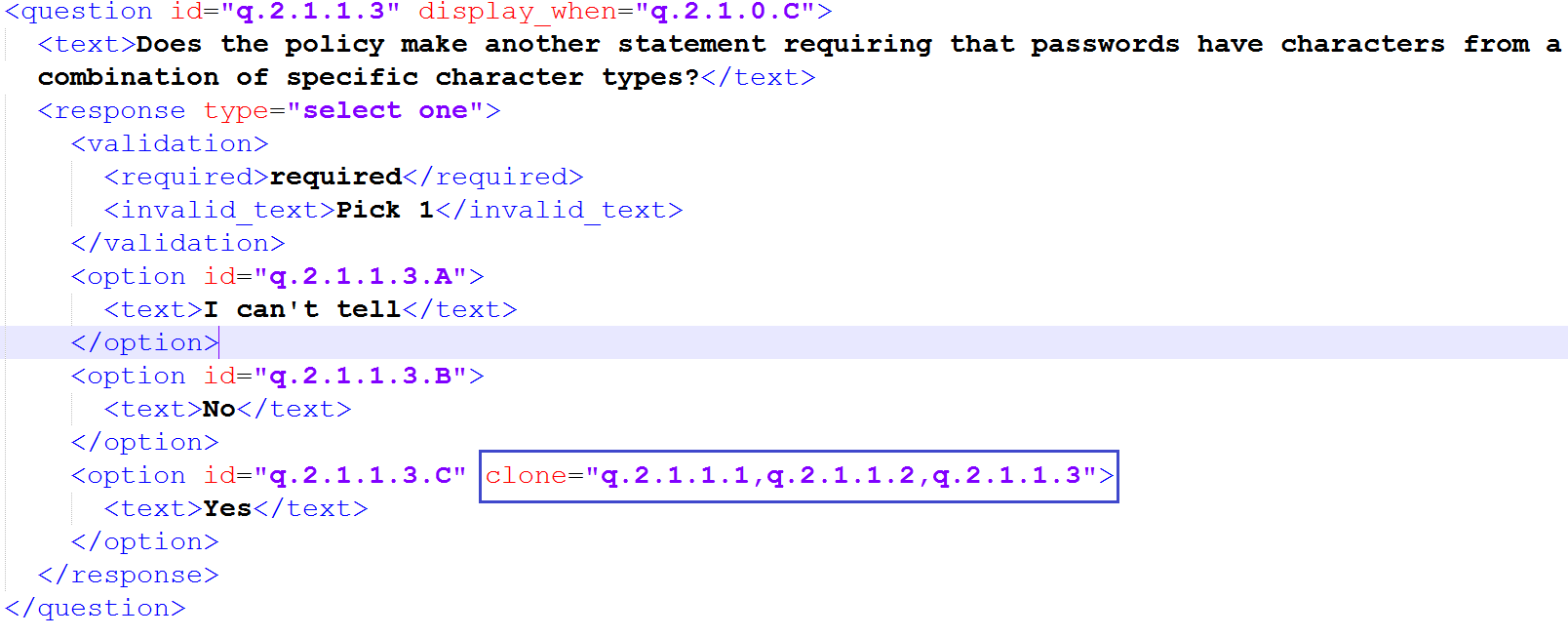
* Cloze Web App Example



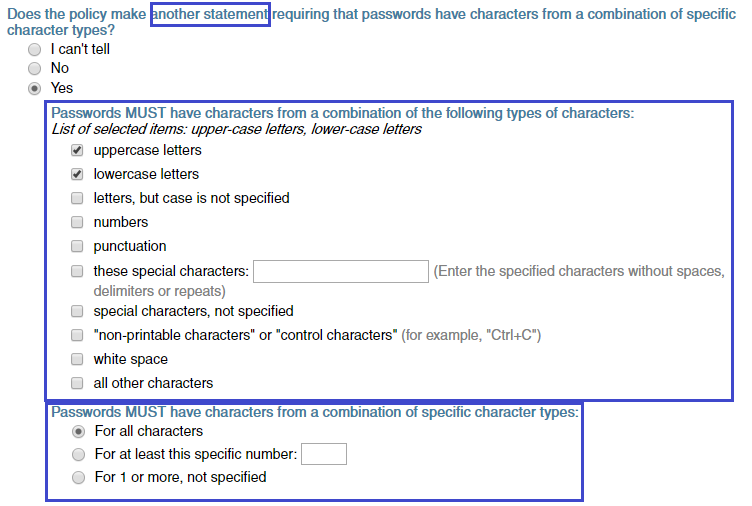
* + 1. **CLONE**
* Used when you want to display an exact copy of a question(s).
* Ex. When you want to ask the same set of questions for multiple items.
* Represented by ‘CLONE:’ , followed by all repeated question IDs.
* Word Doc Example



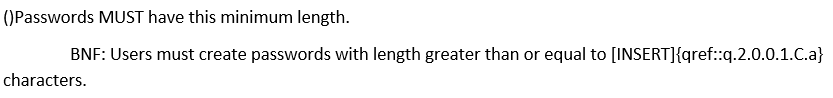
* XML Example



* Web App Example



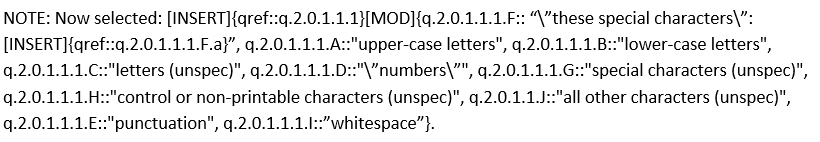
* + 1. **INSERT**
* For when user input needs to be inserted into the text of a question, option, or BNF mapping.
* Must be prefaced with ‘[INSERT]’ and have a ‘qref’/ID pairing within parenthesis.
* Delineated by ‘::’
* Word Doc Example



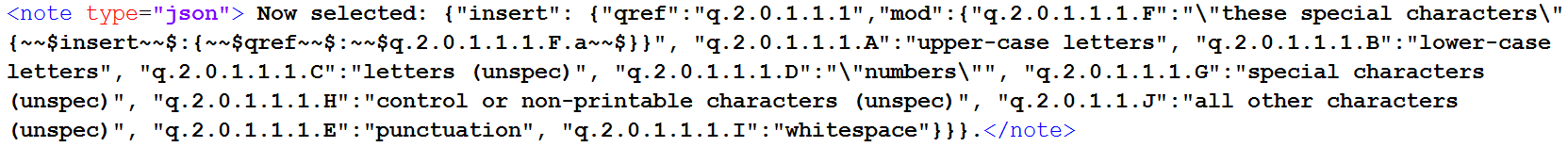
* XML Example



* + 1. **MOD**
* For when inserts need to be nested within another insert
* Represented by [MOD]
* Different question IDs are delineated using ‘::’
* Word Doc Example



* XML Example



* Web App Example



* + 1. **BNF\_MAPPING**
* Specifies the BNF Statement to map to.
* For TEXT, follow up the line with BNF: <bnf mapping statement>. For OPTION, follow up the line with the same. The BNF statement will map to the option or text line accordingly.
* See questionnaire tag descriptions for more detailed explanations.
* Word Doc Example



* XML Example



1. **Questionnaire Tag Descriptions**

**Note:** Many tags support type=”json”, which denotes embedded json objects within the text. E.G.

<text type=”json”> I like {“insert”: {“qref”: “q.1.2.a”}} on my ice cream. </text>

If it is desired to have html embedded in this text as well, one must use a CDATA tag like so:

<text type=”json”><![CDATA[ I like {“insert”: {“qref”: “q.1.2.a”}} <br/> on my ice cream. ]]></text>

**Note:** Some tags and attributes specify a default value. If the tag is omitted, it will be treated as though it were there with the default value. Otherwise the tag is required (unless ‘optional’ is specified).

## <questionnaire>: Root element

Questionnaire Information:

* **<xmlversion> =** the version of xml document / structure – format TBD.
* **<qaversion>** = the question/answer version name – format TBD
* **<BNFversion>** = version of the BNF used – format TBD
* **<indent\_px> =** questions will be indented by this many pixels multiplied by the number on their indent attribute (default 0)
* **<default\_insert\_text>** = text to be used if an “insert” statement on a page can’t be resolved. (i.e. the user has not answered the referenced question. For inserts in BNF, the statement is only generated if all contained inserts resolve.

## <index>: To display the order of questions and page breaks. Navigation should be based on this.

* Attributes:
  + **grouping** – can be “page” or “1”, “2”, “3”, etc. Selects what level of grouping to create pages from. If the level is an integer and that level of grouping is not present, the next level up (lower number) that is present will be used.
  + **interweave\_next\_level** – when “true”, and the grouping is an integer, pages will be rendered with the instructions element from the next group down interwoven among the questions.
* **<group>** - There are unlimited levels of question groupings. No group can contain both a group and page - If the index violates this rule, an exception will be thrown when the application starts which prints information about which group is to blame.
  + Attributes:
    - **level** –to keep track of what level in the hierarchy we are and useful for navigation. Always an integer; if this number is equal to the grouping attribute from <index>, then all questions contained within this group will be rendered on a single page of the survey.
    - **title** – unique identification to be used in navigation/breadcrumbs as well as at the top of the group webpage where the page title goes.
    - **comment\_ref** – refers to the id of a comment element defined later in the document. Only used if this level of grouping is used. All comments are numbered as “c.#”.
  + **<instructions>** - Instructions are rendered when this level of grouping is selected. Instructions will also be rendered when one level up (lower number) is selected, and the interweave\_next\_level attribute is true. (optional)
    - Attributes:
      * **type=”json”** – to indicate that the text should be searched for a JSON clause. Adding type=json is always optional. See json documentation for “insert”.
  + **<text>** - only shown when this level of grouping is selected. (optional)
    - Attributes:
      * **type=”json”** – to indicate that the text should be searched for a JSON clause. See json documentation for “insert”.
  + **<page>**
    - Attributes (Only used when “page” option is chosen):
      * **title**– unique identification to be used in navigation and potentially at top of page.
      * **comment\_ref** - refers to the id of a comment element defined later in the document. Only used if “page” option is used. All comments are numbered as “c.#”.
    - **<instructions>** - only shown if grouping is “page” (optional)
      * Attributes:
        + **type=”json” –** to indicate that the text should be searched for a JSON clause. See json documentation for “insert”. (optional)
    - **<text>** - only shown when grouping is “page”.
      * Attributes:
        + **type=”json” –** to indicate that the text should be searched for a JSON clause. See json documentation for “insert”.
    - **<include>** – if there are questions at this level, this is used to show which questions. (can be multiple)
      * Attributes:
        + **qref** – “question reference”

## Questions section

* **<questions>**
  + **<question>**
    - attributes:
      * **id** – always numerical, with periods denoting hierarchical level, starts with “q.”
      * **display\_when** – the question should only appear when the reference question/response option is true. (optional)
      * **display\_where** – if the question is dependent on a response, then this denotes whether the question should appear immediately below the response (i.e. option), or after the main question. If this attribute doesn’t exist, then whatever location is specified in the index should be used. (May be in a different group or page than the display\_when question) (optional)
      * **indent** – level of indentation for this whole question. Question will be indented by this number times “indent\_px” pixels. Defaults to 0.
    - **<title>** - the title of the question. If this is the first question on a page and the page title is exactly the same, this title will not be shown. (optional)
    - **<instructions>** - (optional)
      * Attributes:
        + **type=”json” –** to indicate that the text should be searched for a JSON clause. See json documentation for “insert”.
    - **<text>** - typically is the question itself (optional)
      * Attributes:
        + **type=”json” –** to indicate that the text should be searched for a JSON clause. See json documentation for “insert”.
    - **<note> -** information to be specially highlighted, regardless of hierarchy (optional)
      * Attributes:
        + **type=”json” –** to indicate that the text should be searched for a JSON clause. See json documentation for “insert”.
    - **<response>** - see below

## Responses

* **<response>** - denotes when user will be interacting. May have question text if the type is true/false or cloze. There must be only one response tag in any question.
  + Attributes:
    - **type** - the type of response
      * select one - This is a "select one" type response which means it contains a list of options to choose from. Rendered as radio buttons.
      * select multi - User can select multiple answers. Rendered as checkboxes.
      * textbox – A small text input area
      * Memo - A large text input area.
      * cloze - text with intermingled json objects which represent inline questions.
  + **<validation>** - (optional)
    - **<required>** - To be considered answered (for the completion page), at least one answer needs to be checked in this select multi Ex. <required>required</required>. All questions have the required tag added to them by default, except for select multi questions. When a select multi is required, at least one option must be selected for the question to validate. To make any question explicitly not required, add <required></required> to validation, and no validation warnings will be generated if there is no response to the question.
    - **<invalid\_text>** - This text will be shown as mouseover on the warning icon when this question is invalid
    - HTML 5 Validation options. Any other options will be passed directly to the element as key/value pairs. Ex. <pattern>[^\s]\*</pattern> would mean that the text box would only validate if there were no spaces.
    - **<lt> <lte> <gt> <gte> <eq>** - are all supported the same way as in json validation. i.e. <lte>4</lte> or <gt>q.3.2.4</gt>
  + **<BNF\_mapping>** = BNF statement to map to (by text for now, but could be ID field). (optional, may have multiple)
    - Attributes:
      * **id** – format = b.#
      * **type**=”json”
      * **when – references a comma separated list of option ids (e.g. when=”q.1.2.A,q.1.2.B”)** this mapping applies when ALL of the stated option ids are selected. If SOME, but not ALL of the option ids are selected, the bnf statement will still be generated, but with “INCOMPLETE BNF” prepended to it, along with the missing option ids in parentheses. E.g. if q.1.2.A was selected, and q.1.2.B was not, the BNF statement would start with “INCOMPLETE BNF (q.1.2.B):”
  + **<text>** 
    - Attributes:
      * **Type=”json”**
  + **<option>** - option for a selection type of response (may have multiple)
    - Attribute:
      * **id** – format = The question id plus a period and an upper-case letter.
      * **clone –** a question (given next to the “clone” attribute) may need to be replicated based on how a user responds to a question. List the clone id(s) separated by comma (no space). The clone will be given an automatically generated ID, incremental to the previous question. The clone includes all of the original question’s subquestions and each subquestion / option / cloze id as well as any other ids is updated. There may be many clones of a question, and clones may be cloned.
    - **<text>** - text in the option.
      * Attributes:
        + **type=”json”**– to indicate that the text should be searched for a JSON clause.
    - **<validation>** -same as above
    - **<BNF\_mapping>** = A BNF\_mapping under an option will only be generated when that option is selected. Otherwise it behaves the same way as described above: see the description for BNF\_mapping under <response>.
    - **<note>** - additional instructions or information (optional)
      * Supported on ‘select multi’ and ‘select one’ questions. The note will be rendered just above the option it is attached to, and indented to the same level as the checkbox or radio button.
      * Attributes:
        + **type**=”json”– to indicate that the text should be searched for a JSON clause.

## Additional Comments

* **<additional comments>**
  + **<comments>** = identical to questions. ID’s formatted as “c.#”

## Other pages with questions:

* **<general\_comments>** - Questions for the general comments page. Rendered on a single page in the order they appear – otherwise the same way as questionnaire questions are rendered. Any items that are not part of a question will have their contents rendered in the order they appear, regardless of the name of the tag.
* **<demographics\_survey>** - questions for the demographic survey page. Rendered on a single page in the order they appear – otherwise the same way as questionnaire questions are rendered. Any items that are not part of a question will have their contents rendered in the order they appear, regardless of the name of the tag.

## Static Text:

* **<static\_text>**
  + **< title>** - title at the very top of each page
  + **<help\_text>** - the text (or image reference?) to use for each “help” button/link
  + **<enable\_flashed\_messages> -** Whether or not to display status messages on the page on login/logout, wrong password, and other events. Also controls whether or not to show incomplete question ids on the completion page. Leave blank to disable
  + **<enable\_logged\_in\_message>** - Whether or not to display a "Logged in as ...." message at the top of each page. Leave blank to disable
  + **<enable\_question\_ids> -** Whether or not to display the id of each question under the title. Leave blank to disable
  + **<nojs\_message>** - Message to display if the user does not have javascript enabled
  + **<nopopups\_message>** - Message to display if the user is blocking popups for this page
  + **<email\_sent\_text>** - Message displayed in alert box after submitting an email from the general comments page (if it was sent successfully).
  + **<email\_error\_text>** - Message displayed in alert box after submitting an email from the general comments page (if there was an error).
  + **<email\_ok>** - Text for the ‘ok’ button on the alert box displayed after submitting an email from the general comments page.
  + **<email\_title>** - Title of in alert box after submitting an email from the general comments page.

### <sign\_in> - the sign in page

* + - **<pgtitle>** - title to be at top of the page
    - **<username>** - text for spot users enter their user name
    - **<no\_account>** - text for button / link if the user doesn’t have an account
    - **<intro\_text>** if additional information is wanted
    - **<instructions>** - if additional information is wanted
    - **<error\_text>** - text to show in a popup if the user-name doesn’t exist (e.g. “Invalid user name. Please try again or sign up for an account”)
    - **<error\_title\_text>** - title on the error popup if the user-name does not exist.

### <sign\_up> - the sign up for an account page

* + - **<pgtitle>** - title to be at top of the page
    - **<intro\_text>** if additional information is wanted
    - **<instructions>** - if additional information is wanted
    - **<name\_taken>** - text to show in a popup if the user-name already exists (e.g. “That user name is reserved. Please try a different user name”)
    - **<name\_taken\_title>** - tite for popup window when name is taken.
    - **<has\_account>** - text for button / link if the user already has an account
    - **<success>** - text for when creation is successful
    - **<user\_created\_text>** - text for alert when user id is created
    - **<user\_created\_ok>** - text for button 'created' pop-up
    - **<user\_created\_title>** - title for alert box wen user id is created

### <policies\_for\_review>

* + - **<pgtitle>** - Title for the policies for review page.
    - **<preinstructions>** - Additional instructions for the page
    - **<instructions>** - Instructions on the policies for review page.
    - **<new\_and\_draft\_text>** - Title for new and draft section.
    - **<new\_text>** - Title for ‘new’ subsection.
    - **<draft\_text>** - Title for ‘draft’ subsection.
    - **<completed\_text>** - Title for ‘completed’ section.

### <questionnaire>-

* + Text for policies for review page and completion page from the study.
    - **<back\_text>** - Text for the link back to the policies for review page.
    - **<finish\_text>** - Text for the link on the sidebar, and at the bottom of the last questionnaire page to go to completion page.
    - **<continue\_label>** - Label for continuing with the selected action for validation popups on questionnaire pages.
    - **<cancel\_label>** - Label for cancelling the selected action for validation popups on questionnaire pages.
    - **<invalid\_popup\_title>**  - title for javascript popup when not all questions are valid
    - **<invalid\_popup\_text>** - text for javascript popup when not all questions are valid
    - **<logout\_title>** - Title for logout popup window
    - **<logout\_text>** - Text for logout popup window
    - **<logout\_continue>** - Text on “continue” button for logout popup window.
    - **<logout\_cancel>** - Text on “cancel” button for logout popup window.
    - **<clone\_warning\_title>** - Title of clone warning popup window. (Shown when the user changes or removes an answer to a clone question which could cause other questions to be deleted.)
    - **<clone\_warning\_text>** - Text for clone warning popup window.
    - **<clone\_warning\_continue>** - Text on “continue” button for clone warning popup window.
    - **<clone\_warning\_cancel>** - Text on “cancel” button for clone warning popup window.
    - **<empty\_page\_message>** - Message to be displayed on pages which have no questions for the user to answer. (This could be due to questions that have “display\_when” attributes not being shown).

### <demographics\_survey>

* + - **<submit\_text>** - Text for popup upon clicking submit if there are validation issues.
    - **<submit\_text\_title>** - Title for the submit\_text popup box.
    - **<submit\_continue>** - Text for button that continues to submit the survey
    - **<submit\_cancel>** - Text for button that keeps the user at the survey.

### ADMIN Pages

* + **<admin\_configure>**
    - **<pgtitle> - Title for this page.**
  + **<admin\_testing\_setup>**
    - **<pgtitle> - Title for this page.**
  + **<admin\_reporting>**
    - **<pgtitle> - Title for this page.**

### <general\_comments>

* + - **<pgtitle> - Title for this page.**

### <completion\_page>

* + - **<pgtitle>** - Title for the study completion page.
    - **<description>** - Text that appears on the study completion page.
    - **~~<invalid\_text>~~** ~~- Text that appears on the study completion page, after the description if there are invalid questions.~~
    - **<complete\_text>** - Text for the link to confirm completion of the study.
    - **<go\_back\_text>** - Text for the link to return to the study.
    - **<review\_all\_completed>** - text to be inserted in the review section when all responses have been completed
    - **<submit\_all\_complete>** - text to be inserted in the submit section when all responses have been completed
    - **<review\_need\_responses>** - text to be inserted in the review section when there are missing responses
    - **<submit\_need\_responses>** - text to be inserted in the submit section when there are missing responses

### <help>

* + - **<pgtitle>** - Title on the help page
    - **<html>** - Define all content for the page in a CDATA block as html
    - Example:

<![CDATA[  
 <p> I <b> need help!</b> </p>  
 <br/>  
 <br/>  
 <ul>  
 <li> Help me 1</li>  
 <li> Help me 2</li>  
 <li> Link to comments <a href="/general\_comments"> Comment? </a> </li>  
 <a onclick="history.back(-1)" href="javascript:void(0);"> Go Back </a>  
 </ul>  
 ]]

## JSON:

### {“cloze”:{ …}} - For when there is a question within the text of an option. Uses JSON format

* + **id** – format = id of the question/option plus a lower-case letter period in between
  + **type** – the type of input
    - numerical – numerical input box with up/down arrows (in most browsers)
    - select one – rendered as a dropdown menu
    - select multi – rendered as checkboxes
    - textbox = small text box
    - memo = big text box
  + **default –** the default option or text to show (there may be text here which is not listed as an option), used for ‘select one’ cloze questions (drop down menus) (optional – default is blank)
  + **validation** – an object which contains validation information. Does not required the user to respond in order to move on, just warns. (optional)
    - **HTML 5 validation -** See this documentation <https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Forms_in_HTML> The following are legal:
      * “pattern” - html5 regular expressions
      * “min”
      * “max”
      * “step”
      * “maxlength”
    - **Extensions to HTML5 validation:**
      * gt, lt, gte, lte, eq: These options stand for "greater than", "less than", "greater than or equal", "less than or equal", and "equal" respectively. The value of any is the id of a question with an integer answer, or a select multi question (number of responses will be counted). This question will only validate if it satisfies the relationship with the referenced question. i.e. {"validation": { "gte": "q.3.2.a"}} means that this question will be valid only if its answer is greater than or equal to the answer to question 3.2.a. These parameters are only valid on select multi or numerical questions.
      * “invalid\_text” - the text to show when a validation error is tripped.
      * “required” - All questions have the required tag added to them by default, except for select multi questions. When a select multi is required, at least one option must be selected for the question to validate. To make any question explicitly not required, add “required”: “” to validation, and no validation warnings will be generated if there is no response to the question.
  + **options** – an array of options for the user to select from (for use with either “select one” or “select multiple”)

### {“insert”:{ …}} – For when user input needs to be inserted into the text of a question, option, or BNF mapping.

* + **qref –** id of the question whose response will be inserted. For historical reasons “cref” can also be used here.
  + **math -** value can be any positive or negative integer - gets added to the inserted number. Error if the referenced question is not numerical (or a select multi and ‘count’ is present) (optional)
  + **count -** If this key is present, qref must reference a select\_multi question, and the number inserted or used in any math statement will be the number of selected options in the select multi. (optional)
  + **“mod”:{… –** if an the option wording needs to be changed to match the bnf (optional)
    - The value of “mod” is a mapping from either cloze text or option id, to the text which should replace it. i.e. { “second (s)”: “second”, “minute (s)”: “minute”} or {“q.1.2.1.A”:”minutes”, “q.1.2.1.B”:”seconds”}
  + **separator –** when a select multi question is referenced, all responses to it are inserted separated by this string. Defaults to “,”