Working With CCCApply
Supplemental Questions V.2016.1

Contents

Getting Started with Supplemental Questions..... 3

Supplemental Questions: Layout And User Experience.....6

Supplemental Questions: Response Elements.....11

Getting Started with Supplemental Questions

You can create one or more sets of supplemental questions to be added to your college's student application, and you can control which set currently displays. On the *Supplemental Questions* tab in the Administrator you import your XML supplemental questions and activate and deactivate question set(s). The instructions below provide details on activating/deactivating previously-imported XML file sets, and then provide an introduction to creating supplemental questions in XML.

Activating And Deactivating Question Sets

After you have uploaded a set of supplemental questions, there are three factors that determine whether that set will show up in your college's student application:

- The active/inactive status
- The effective date
- The statuses and effective dates of other question sets you have previously uploaded

If you marked a previously-uploaded set of supplemental questions as *Not Active* in the Administrator, that set will not show up on your college's application, regardless of any other factors. Similarly, a future-dated set of supplemental questions will not display until the future date becomes the present date, regardless of other factors. If you have only a single set of questions, then marking it *Active* and setting the effective date to today or earlier will cause that set to show up in your college's application.

Multiple supplemental question sets require some management. For example, if your supplemental question sets have different effective dates, the set with the most recent date (that is not in the future) takes precedence over any others. If you have multiple supplemental question sets with the same effective date, then the set with the lowest *Page ID* number will take precedence.

Consider the following example that shows questions that have been uploaded to the *Supplemental Questions* tab. Note that the default order in which the question sets display is by most recent *Page ID*.

	Page ID	Status	Effective Date
☐	486	Active	03/24/2015
☐	485	Not Active	03/22/2015
☐	484	Not Active	03/22/2015
☐	479	Active	03/22/2015
☐	478	Active	03/21/2015
☐	477	Not Active	03/21/2015
☐	476	Not Active	03/21/2015

If today were 23 March 2015, the questions for *Page ID* 479 would show up on your college's student application. *Page ID* 486 would not show up, because although it is marked *Active* and has the latest effective date, that date is in the future. "Tomorrow", 24 March 2015, *Page ID* 486 will take precedence and the questions on *Page ID* 479 will no longer appear on the application.

	Page ID	Status	Effective Date
☐	486	Active	03/24/2015
☐	485	Not Active	03/22/2015
☐	484	Active	03/22/2015
☐	483	Active	03/22/2015
☐	482	Not Active	03/22/2015

In this case, the questions for *Page ID* 483 take precedence. The effective date is the same for both pages 483 and 484, so the set with the lower page number is used.

Note:

If you have used supplemental questions in the past, you may notice that the <SupplementalQuestions> element contains an *EffectiveDate* attribute. This attribute has no effect on your supplemental questions. The system simply ignores it.

Note:

When you upload a set of questions, there is a two- to three-minute delay before they appear on the student application. In the live environment, you won't notice this delay, as you will have uploaded your questions well in advance. But it will be noticeable during your testing phase. After you upload your test questions, wait a few minutes before attempting to see them on your student application.

Working with XML

Although working with computer languages can be daunting, you don't need to be an expert to create supplemental questions for your college's student application, and you don't need any sophisticated tools. A simple text editor and a rudimentary understanding of XML hierarchy will be helpful.

If you are comfortable writing computer software, you can of course use the tools that best suit you. Otherwise, create your XML files using your operating system's built-in text editor. On Windows, the text editor is called Notepad, and on OS X, it is called TextEdit.

Note:

These applications can work with file formats other than plain text. Be sure to save your files in plain text format.

Important:

Unless you have experience working with text editors and XML, do not use XMLNotepad to edit your files. It adds extra characters to the beginning of the file that will prevent your XML from uploading. If you do have experience in this area, use XMLNotepad to edit your XML (to take advantage of the schema), then use a plain text editor or some other suitable tool to remove the extra characters (the BOM) before uploading.

Most of the contents of your XML file will depend on the questions you want to ask and the responses you expect to receive. For details on creating questions and guiding the student through the application process see*Supplemental Questions: Layout And User Experience* and *Supplemental Questions: Response Elements*.

Setting up Your XML File

1. Copy and paste the example starter XML file structure below into your XML editor before you begin creating questions. This text will serve as your XML base and you can use it over and over again for each new supplemental question XML file set.

```
<SupplementalQuestions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://xmlns.cccnext.org/apply ../../code/ccc-admin/
lib/sites/admin/misc/cccSuppQuesTypes.xsd"
 xmlns="http://xmlns.cccnext.org/apply" CollegeId="nnn" CollegeName="My
 College">



</SupplementalQuestions>
```

2. Set *CollegeId* to your college's three-digit MIS code. If you are creating questions for multiple colleges in a district, then indicate the MIS codes of all of the applicable colleges, separated by commas.
3. Set *CollegeName* to the name of your college.

You will define your questions by inserting XML code in the blank space in the above example. The example below is a complete and legal definition of a supplemental questions page:

```
<SupplementalQuestions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://xmlns.cccnext.org/apply ../../code/ccc-admin/lib/
sites/admin/misc/cccSuppQuesTypes.xsd"
```

```
xmlns="http://xmlns.cccnext.org/apply" CollegeId="611" CollegeName="Allan
Hancock College">

<Header>Tell us more about yourself!</Header>
<Section>
 <YesNo id="1">
  <Label>While attending classes, do you intend to work more than 40 hours
per week?</Label>
 </YesNo>
</Section>

</SupplementalQuestions>
```

Important:

It is very important that your uploaded XML file be error-free. If there are errors in your supplemental questions XML file, you will likely see a difficult-to-understand error message in the Administrator when you attempt to upload it. If you are especially unlucky, your XML file will upload without error, but when the student attempts to navigate to the Supplemental Questions tab, an unfriendly error message may display and they will not see any of your supplemental questions.

It is therefore strongly recommended that before uploading, you validate your XML with a free, online validator tool such as *http://www.xmlvalidation.com/*. This site allows you to copy and paste your XML for syntax checking. After you paste your XML and click **validate**, the site will ask you to copy and paste a second file. In industry jargon, the file it is requesting is called a *schema*. You can copy the schema from here and paste it into the validator. The validator then tells you if there are problems with your XML. Although the error messages may still be somewhat intimidating, this particular validator will at least show you in a friendly way *where* the error is in your XML.

Supplemental Questions: Layout And User Experience

For supplemental questions, there are a number of elements available that relate to the layout of the application and the applicant's user experience, as opposed to managing the raw data of the applicant's responses. This section provides detailed information about the layout-related elements with example pairs of XML and the resulting display in the application.

These elements fit into two groups: Layout Elements: *Header*, *Indent*, and *Section*; and Prompt Elements: *HoverHelp* and *Label*.

Additionally, there are several attributes available for response elements that relate to guiding the student through the application process. These are the *default*, *required*, and *title* attributes. The following descriptions and examples show how to use these items.

Layout Elements: *<Header>*, *<Indent>*, *<Section>*

The <Header> element creates a heading for the whole page, as well as for any sections you define for the page.

The <Indent> element offsets portions of the page for readability or hierarchical display.

The <Section> element groups other elements visually on the page. See the examples below for details.

User Prompt Elements: *<HoverHelp>*, *<Label>*

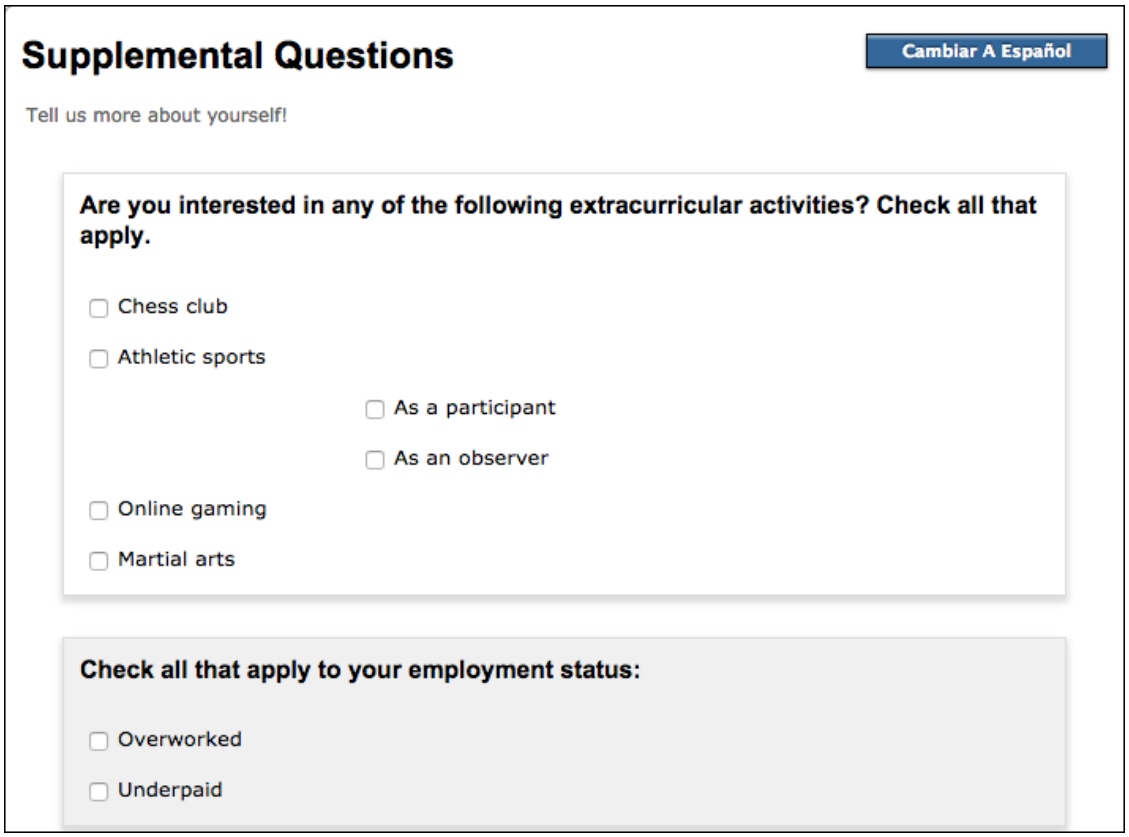
Both <Label> and <HoverHelp> are used only as child elements of the response elements. The <Label> element can be thought of as the question being asked for a response element, while the <HoverHelp> element can be used to provide further explanatory text when the applicant hovers the mouse over the related input field. Multiple <HoverHelp> children are allowed, one for each language your student application supports.

The following XML will create:

- A page header
- Two sections of check boxes, each section with its own header
- Two check boxes indented to the right from those above and below
- One check box with hover help in both English and Spanish

```
<Header>Tell us more about yourself!</Header>

<Section>
  <Header>Are you interested in any of the following extracurricular
 activities? Check all that apply.</Header>

  <Checkbox id="1"><Label>Chess club</Label></Checkbox>
  <Checkbox id="2"><Label>Athletic sports</Label></Checkbox>

  <Indent>
    <Checkbox id="3"><Label>As a participant</Label></Checkbox>

    <Checkbox id="4">
      <Label>As an observer</Label>
      <HoverHelp lang="en">Check this box if you enjoy watching sporting
 events</HoverHelp>
      <HoverHelp lang="es">Marque si te gusta ver los eventos deportivos</
HoverHelp>
    </Checkbox>
  </Indent>

  <Checkbox id="5"><Label>Online gaming</Label></Checkbox>
  <Checkbox id="6"><Label>Martial arts</Label></Checkbox>
</Section>
```

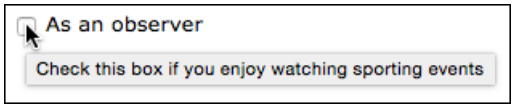
```
<Section>
  <Header>Check all that apply to your employment status:</Header>
  <Checkbox id="7"><Label>Overworked</Label></Checkbox>
  <Checkbox id="8"><Label>Underpaid</Label></Checkbox>
</Section>
```



The application will use the <HoverHelp> tag that applies to the currently-selected language. Only English (*lang"=e"n*) and Spanish (*lang"=e"s*, for *Español*) are supported. The hover help appear when the applicant hovers the mouse over the check box.

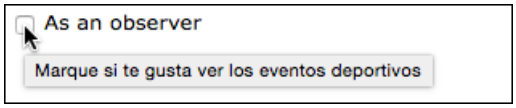
This hover help from the example above will appear when the page is in English mode:

```
<HoverHelp lang="en">Check this box if you enjoy watching sporting events</
HoverHelp>
```



This hover help from the example above will appear when the page is in Spanish mode:

```
<HoverHelp lang="es">Marque si te gusta ver los eventos deportivos</HoverHelp>
```



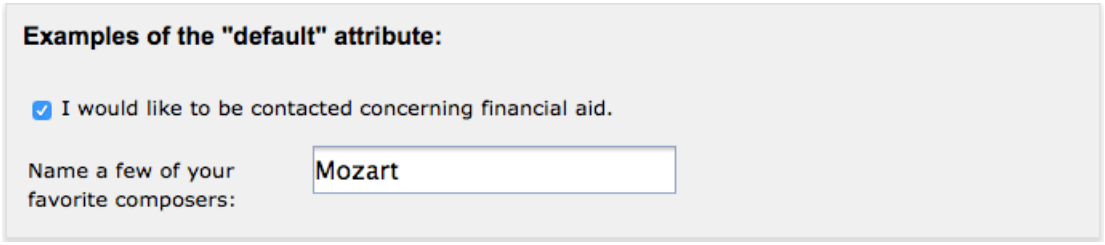
Warning:

You *must* specify a language (using the *lang* attribute as in the examples above) for your hover help, or your *Supplemental Questions* tab will fail, and the applicant will see an error message rather than your supplemental questions.

Response Element: The *default* Attribute

Depending on your specific circumstances, you can allow the applicant to ignore a given question, or you can require the applicant to provide a response. If you allow a question to be ignored, the system will store a system-default value for the question. The legal default values depend on the response element you specified. When you add a *default* attribute to a response element, the system auto-answers the question with the default value you specify. When the page appears, the question displays as already answered:

```
<Section>
 <Header>Examples with the "default" attribute:</Header>

 <Checkbox id="29" default="checked"><Label>I would like to be contacted
 concerning financial aid.</Label></Checkbox>

 <Text id="20" default="Mozart"><Label>Name a few of your favorite
 composers:</Label></Text>
</Section>
```



The following table shows the legal defaults for each response element.

Element	Legal Defaults
Checkbox	"checked" or "unchecked"
CountryList	Any of the two-character country codes in the Data Dictionary (Standard or International)
Date	See note below.
Menu	The contents of the *value* attribute in any of the <MenuItem> elements you create for the menu. See *The <Menu> Element* for details.
PhoneNumber	
StatesList	Any of the two-character state codes in the Data Dictionary for the application (Standard or International).
Text	Any free-form text less than the maximum length of 250 characters, or less than the maximum you set using the *maxLength* attribute.
YesNo	"yes" or "no"

Note:

It is recommended that you do not set a default for a <Date> element, as it will malfunction on the student application page.

Note:

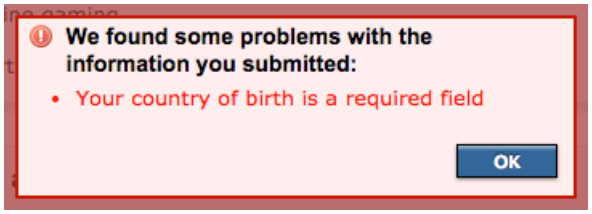
Because a response is automatically filled in when you set a default, setting *required"=tru"e* will have no effect. The applicant can simply ignore the question, in which case the default you have supplied will be stored in the database when the student submits the application.

Response Element: The *required* Attribute

The *required* attribute allows you to require that the applicant respond to selected response elements, or to groups of response elements. To require responses to all the elements in a section, use *required"=tru"e* in the <Section> element. For an example, see the discussion of the *title* attribute below. To require responses to an individual element, use *required"=tru"e* in that element. The following is an example of a <CountryList> element that requires the applicant to select a country:

```
<CountryList id="3" required="true" title="Your country of birth">
 <Label>Select the country in which you were born.</Label>
</CountryList>
```

The system will not allow the applicant to continue the application process if no country is selected. Instead, an error message will be displayed:



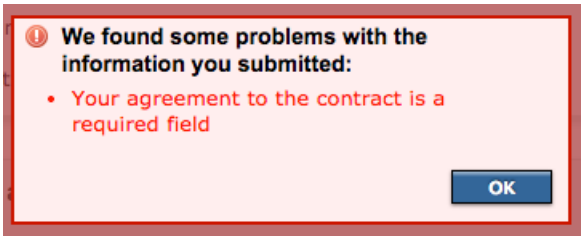
Note:

The *title* attribute allows you to customize the error message somewhat, to make it easier for the applicant to understand the nature of the error. It is best to supply a title for every response node; see *The title Attribute* for details.

Note:

For <Checkbox> elements, *required"=tru"e* means that the applicant *must* mark the check box. This applies to <Checkbox> elements within a <Section> that has *required"=tru"e*, as well as individual <Checkbox> elements that have *required"=tru"e*. This might be useful for requiring the applicant to read the application carefully or agree to some specific terms:

```
<Checkbox id="11" required="true" title="Your agreement to the contract">
 <Label>I agree to the terms and conditions of the contract.</Label>
</Checkbox>
```



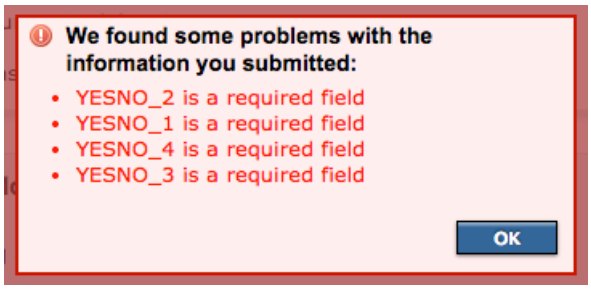
Response Element Attribute: The *title* Attribute

The *title* attribute allows you to give names to response elements to make certain error messages more intelligible to the applicant. Consider the following example, in which the <Section> element contains *required"=tru"e*. This means all of the yes/no questions in the section must be answered.

```
<Section required="true">
 <Header>While you attend classes, do you intend to...</Header>

 <YesNo id="1"><Label>...work more than 40 hours per week?</Label></YesNo>
```

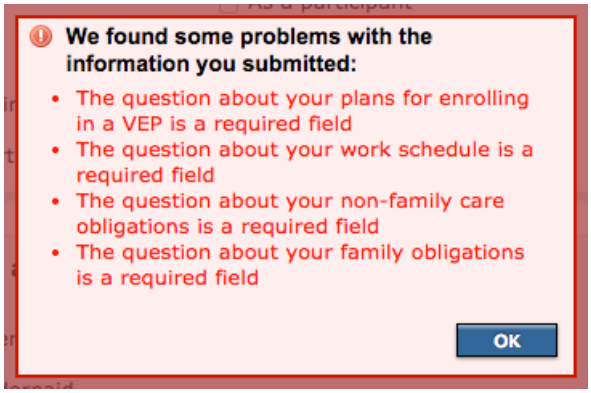
```
 <YesNo id="2"><Label>...enroll in a vocational education program (VEP)?</
Label></YesNo>
 <YesNo id="3"><Label>...care for children or elderly persons in your family?
</Label></YesNo>
 <YesNo id="4"><Label>...care for children or elderly persons outside your
 family?</Label></YesNo>
</Section>
```

The system will not allow the applicant to continue the application process if any of these questions are left unanswered. Instead, an error message will be displayed:



It is unlikely that the applicant will understand these errors. It is recommended that you always use the *title* attribute to make the error message friendlier wherever you require a response.

```
<Section required="true">
 <Header>While you attend classes, do you intend to...</Header>

 <YesNo id="1" title="The question about your work schedule">
  <Label>...work more than 40 hours per week?</Label>
 </YesNo>

 <YesNo id="2" title="The question about your plans for enrolling in a VEP">
  <Label>...enroll in a vocational education program (VEP)?</Label>
 </YesNo>

 <YesNo id="3" title="The question about your family obligations">
  <Label>...care for children or elderly persons in your family?</Label>
 </YesNo>

 <YesNo id="4" title="The question about your non-family care obligations">
  <Label>...care for children or elderly persons outside your family?</Label>
 </YesNo>
</Section>
```



Supplemental Questions: Response Elements

The response elements are the XML elements that allow you to manage the raw data of supplemental question applicant responses, which are stored in the CCC Tech Center's supplemental questions database.

A response element serves three primary functions:

- It determines how the question is to be presented to the applicant.
- It determines the form in which the response data is to be stored.
- It indicates to the system where in the database to store the response.

Response elements can contain both <Label> and <HoverHelp> child elements to help guide the user through the application process. Certain attributes such as *default* and *required* can also be assigned to response elements for guiding the user. See *Supplemental Questions: Layout And User Experience* for details.

Each type of response element is associated with a group of fields in the database. Responses to <Checkbox> elements, for example, are stored in a group of fields whose names follow the pattern of *supp_check_* followed by a two-digit number. There are 50 *supp_check* fields in the database, from *supp_check_01* to *supp_check_50*. The *id* attribute you assign to each response element will determine which particular field the response is stored in. For example, the response to <Checkbox id="42"> will be stored in the field called *supp_check_42*. When you are ready to download student data to your college's Student Information System, you will use these field names to tell the Download Client which data you want.

The following table shows the association between a response element and the supplemental questions database.

Element Name	Database Field Name Prefix	Number of Fields
Checkbox	supp_check	50
CountryList	supp_country	5
Date	supp_date	5
EncryptedText	supp_secret	5
Menu	supp_menu	30
PhoneNumber	supp_phonenumber	5
StatesList	supp_state	5
Text	supp_text	20
YesNo	supp_yesno	30

The database fields for most of these elements have a fixed size. For example, because a <YesNo> element accepts only a *yes* or a *no*, the corresponding database field is only one character long, allowing either a "1" for *yes* or a "0" for *no*. Some of the response elements correlate to fields that can contain more information. It is not required that you fill a database field, but it is legal to do so.

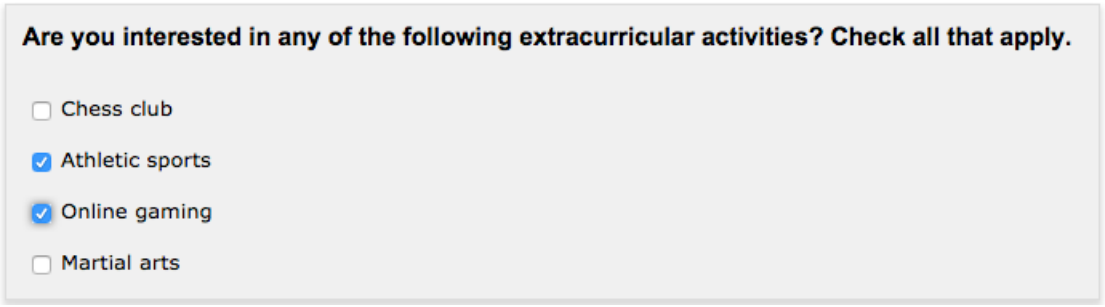
Element Name	Capacity of Database Field
Menu	60 characters
PhoneNumber	25 characters
Text	250 characters

Below are some details on how to use each type of response element.

The <Checkbox> Element

This element creates a simple check box that the applicant can either select or leave untouched, for a simple checked/unchecked result. Check boxes are typically used in groups, to allow the applicant to provide multiple answers to a single question. For example:

```
<Section>
 <Header>Are you interested in any of the following extracurricular
 activities? Check all that apply.</Header>
 <Checkbox id="1"><Label>Chess club</Label></Checkbox>
 <Checkbox id="2"><Label>Athletic sports</Label></Checkbox>
 <Checkbox id="3"><Label>Online gaming</Label></Checkbox>
 <Checkbox id="4"><Label>Martial arts</Label></Checkbox>
</Section>
```

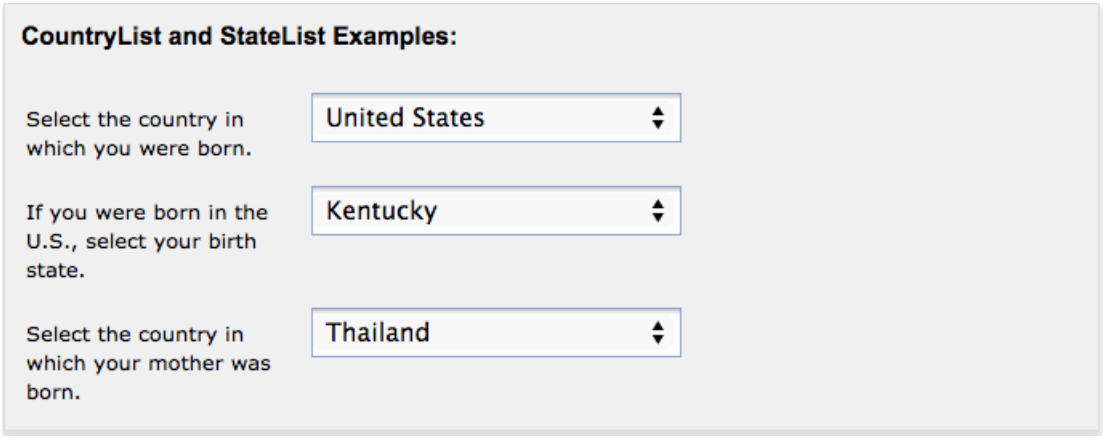


In the database (and Download Client results, if downloaded), the example answers in the image above would result in a "0" being stored in *supp_check_01* and *supp_check_04*, while a "1" would be stored in *supp_check_02* and *supp_check_03*.

The <CountryList> and <StatesList> Elements

These two elements provide similar functionality but differ slightly in their details. The <CountryList> element creates a drop-down menu that lists the countries recognized by the CCC Tech Center. The <StatesList> element creates a drop-down menu that lists U.S. states and various associated territories. For example:

```
<Section>
 <Header>CountryList and StateList Examples:</Header>

 <CountryList id="1"><Label>Select the country in which you were born.</
Label></CountryList>

 <StatesList id="1"><Label>If you were born in the U.S., select your birth
 state.</Label></StatesList>

 <CountryList id="2"><Label>Select the country in which your mother was
 born.</Label></CountryList>
</Section>
```



This would result in "US" being stored in *supp_country_01*, "KY" in *supp_state_01*, and "TH" in *supp_country_02*. For details on the two-character codes for countries and states, see the data dictionaries for either the Standard or International applications.

The <Date> Element

This element creates a text input field that allows for a date in MM/DD/YYYY format. We recommend that you do not use this element; it does not properly validate user input, and can result in invalid values being stored in the database.

The <EncryptedText> Element

The EncryptedText element creates a text input field that is a hidden password validator field, displaying asterisks (*) for the characters entered.

Note:

Though this element is called EncryptedText, it is not really handling encryption, but is only validating the password the user enters to ensure it matches your regular expression entered in the regex attribute. The data entered by the user in this supplemental question field will be encrypted in the database after they submit their application, and that is where the encryption takes place.

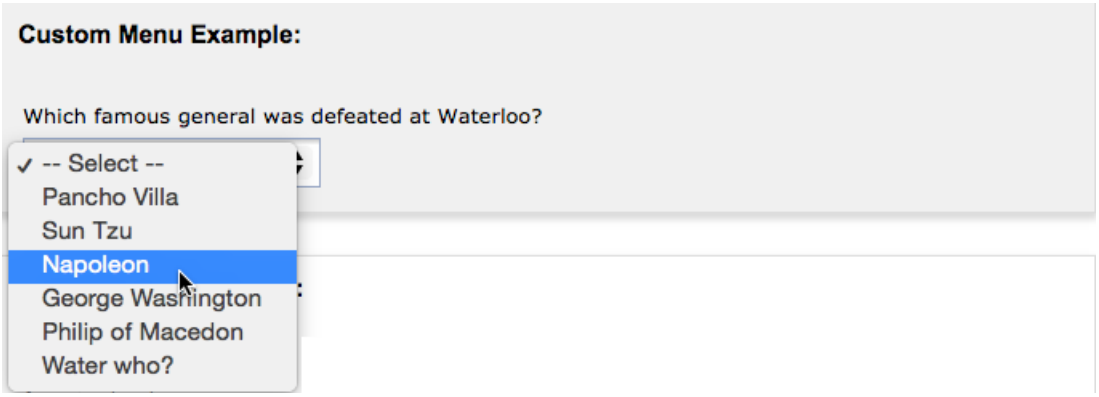
- The id attribute sets the id for the field. You can have up to five EncryptedText elements.
- The required attribute can be either true or false. When true, the user is required to enter values in the fields before submission.
- The reenter attribute can be either true or false. When true, the user must reenter the value in the second field that displays.
- The maxLength attribute sets the maximum length of the text entry.
- The regex attribute is where you define the regular expression to be applied to the fields, and validates the user entry to your requirements. The example below specifies a 6-to-20 character string with at least one digit, one upper case letter, one lower case letter and one special symbol (“@#$%”). You may provide your own regular expression that suits your needs here.

```
<Section>
    <Header>Hidden Password Example:</Header>

            <EncryptedText id="1" required="true" reenter="true"
 maxLength="20" regex="((?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%]).{6,20})">
                <Label>Temporary Password</Label>
            </EncryptedText>
</Section>
```

The <Menu> Element

This element allows you to create a custom drop-down menu that allows the applicant to select a single item from a list. Use one <MenuItem> child element to define each item in your drop-down menu. The *label* attribute of <MenuItem> is the text that will display to the user in the drop-down menu. The *value* attribute indicates the data that will be stored for the selected <MenuItem> element. For example:

```
<Section>
 <Header>Custom Menu Example:</Header>
 <Menu id="27">
  <Label>Which famous general was defeated at Waterloo?</Label>
  <MenuItem value="0" label="Pancho Villa"/>
  <MenuItem value="1" label="Sun Tzu"/>
  <MenuItem value="2" label="Napoleon"/>
  <MenuItem value="3" label="George Washington"/>
  <MenuItem value="4" label="Philip of Macedon"/>
  <MenuItem value="5" label="Water who?"/>
 </Menu>
</Section>
```



The example above would result in the value "2" being stored in *supp_menu_27* in the database. It is best practice, for the sake of simplicity, to use numeric values to represent your menu selections. However, you can use any alphanumeric string up to 60 characters long. Also, although you would typically assign a unique value to each item, it is legal to assign the same value to multiple items. The obvious drawback is that you would not later be able to distinguish between the selections made by applicants, but the system allows you to make this choice if your situation calls for it.

The <PhoneNumber> Element

This element allows you to create a box that accepts a phone number, very much like the <Text> element described below. The maximum length for the phone number field is 25 characters and the system will prevent the applicant from typing more than 25 characters into the field.

Beside the system-imposed maximum length, the difference between <PhoneNumber> and <Text> is that <PhoneNumber> allows you to use the <Format> child element to restrict the applicant to a particular phone number format, or set of formats. The applicant can type anything into the field, but the system will halt the application process and display an error if the user attempts to submit the page with a phone number that doesn't match at least one of the specified formats. To allow free-form entry into the phone number field, create a <PhoneNumber> element without any <Format> child elements.

Note:

The phone number text field that is rendered in the application has default help text below the text field indicating that the format of (xxx) xxx-xxxx is the acceptable format. If you want to allow for other formats than this hard-coded UI display, it is recommended that you show the applicant the acceptable format(s) in your <Label> element. Naturally, this will be difficult if you allow many different formats. You could minimize wordiness by leaving out the <Format> element altogether, but that will allow the applicant to enter something that isn't a phone number at all. Balance the complexity of your phone number prompts against

your college's willingness to accept ill-formatted phone numbers, including input that may not be a phone number at all.

Note:

If you use the <Format> child element for a <PhoneNumber>, the application will behave as though you had set *required"=tru"e*. That is, the applicant will not be allowed to leave the field blank, even if you set *required"=fals"e*. If you wish to allow the field to be left blank, you must leave out the <Format> child element altogether. It is legal, although of limited usefulness, to set *required"=tru"e* for a <Format>-less <PhoneNumber> element, because although input would be required, any input, no matter how useless, would be accepted by the application.

```
<Section>
 <Header>Phone Number Examples:</Header>

 <PhoneNumber id="1">
  <Label>Enter your home phone number:</Label>
  <Format>(999) 999-9999</Format>
 </PhoneNumber>

 <PhoneNumber id="2">
  <Label>Enter your cell phone number:</Label>
  <Format>(999) 999-9999</Format>
  <Format>999-999-9999</Format>
  <Format>(99) 99 9999 9999</Format>
  <Format>99 99 99 99</Format>
 </PhoneNumber>

 <PhoneNumber id="3">
  <Label>Enter your emergency contact's cell phone number:</Label>
 </PhoneNumber>
</Section>
```

In the example above, the <PhoneNumber> element with the *id"="1* would allow a phone number in the one format indicated. The <PhoneNumber> with *id"="2* would accept a phone number in any one of the four formats indicated. The <PhoneNumber> with *id"="3* would accept any alphanumeric input or keyboard symbols, up to 25 characters long because no child format element is specified. The data entered into the three resulting phone number fields, including spaces and punctuation marks as accepted by their <Format> child elements, would be stored in *supp_phonenumber_01*, *supp_phonenumber_02*, *supp_phonenumber_03*.

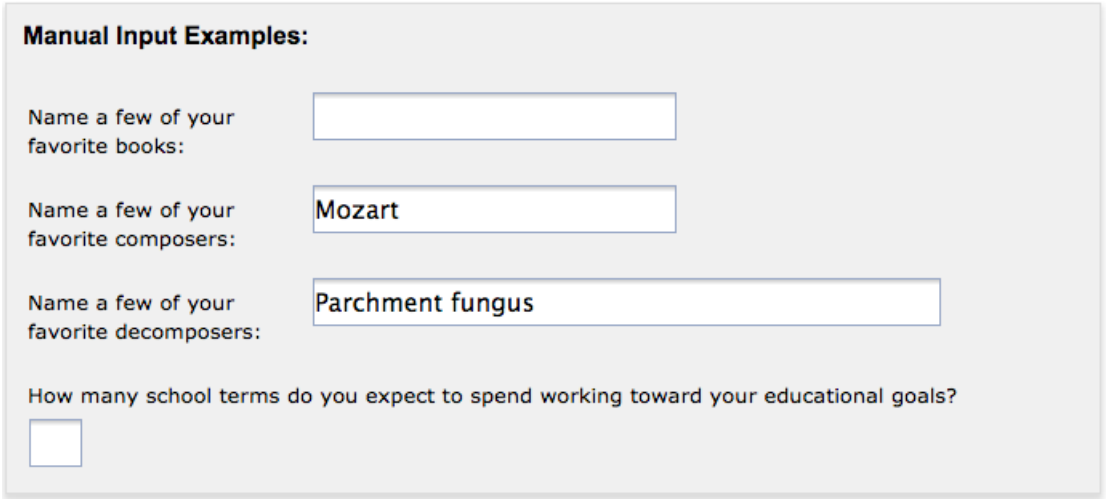
The <Text> Element

This element allows you to create a box that accepts non-specific keyboard input. While the maximum length for the field is 250 characters, you can further limit the length using the *maxLength* attribute. Setting this value will prevent the applicant from typing beyond the character limit you have set.

You can also require numeric-only input by using *numeric"=tru"e*. Setting this attribute will prevent the applicant from typing anything other than digits in the field. It is recommended that you use *maxLength* whenever you use *numeric"=tru"e*, to impose an approximately reasonable maximum number that can be entered into the field. For example:

```
<Section>
 <Header>Manual Input Examples:</Header>

 <Text id="16"><Label>Name a few of your favorite books:</Label></Text>
 <Text id="17" default="Mozart"><Label>Name a few of your favorite
 composers:</Label></Text>

 <Text id="18" default="Parchment fungus" maxLength="50">
  <Label>Name a few of your favorite decomposers:</Label>
 </Text>

 <Text id="19" numeric="true" maxLength="2">
```

```
  <Label>How many school terms do you expect to spend working toward your
 educational goals?</Label>
 </Text>
</Section>
```



If the student were to submit the application without adding to these input fields, the effect in the database would be as follows:

- An empty string would be stored in *supp_text_16*. (The field would be left blank.)
- "Mozart" would be stored in *supp_text_17*.
- "Parchment fungus" would be stored in *supp_text_18*.
- And *supp_text_19* would be left blank.

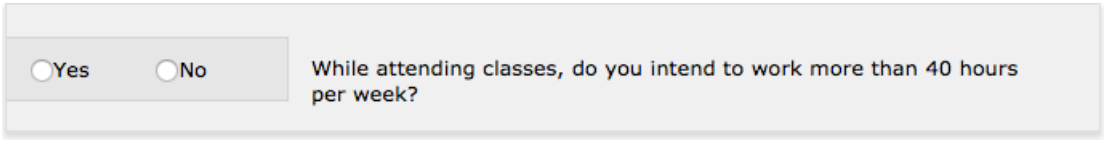
Note:

Setting *numeric"=tru"e* affects only the applicant's interaction with the input field, not the content in the database. If, for example, the applicant leaves the field blank, then the corresponding database field (*supp_text_19* in the example above) will also be left blank, as opposed to containing a zero.

The <YesNo> Element

This element allows you to create a simple yes/no question. For example:

```
<Section>
 <YesNo id="1">
  <Label>While attending classes, do you intend to work more than 40 hours per
 week?</Label>
 </YesNo>
</Section>
```



In the database, a "1" would be stored into *supp_yesno_01* if the applicant selected "Yes", or a "0" if the applicant selected "No".