COE Form Processor V3 Documentation

Table of Contents

Overview

Styles

`XML and XSLT` for XML, XHTML and XSLT (its all XML)
`SQL` for SQL

Hyperlinks outside of this document
Bookmarks within this document
`File Names`
FORM MANAGER INTERFACE NAME
This is a tick. It represents a painful, but difficult to find problem.

What is the Form Processor

The form processor is a web application for generating forms which manipulate database records. The forms are edited via the form manager, an Access interface. This document is not written well enough to explain the form processor to someone without some familiarity with it.

How does the application work?

1. *Gets state data from URL and Intranet Session class.*
2. *Executes stored procedures to get configuration and target data.*
3. *Populates large objFormParameters data structure.*
4. *Populates target data class.*
5. Generates "form_data" xml document with the following elements. (see Appendix D)
   5.1. `<form_data>` is the root element with the following children:
   5.2. `<configuration>…</configuration>`. Contains most all the data from the frm_* tables except custom xslts.
   5.3. `<target_data>…</target_data>`. Contains all the target data and permissions.

5.4. `<index_data>…</index_data>`.
5.5. `<response>…</response>.` Error, instructions, warings, etx.
5.6. `<state>…</state>`. State variables. Generally the name value pairs in URL
5.7. `<supp_data>…</supp_data>` Supplemental data brought in for display purposes.
5.8. A schema or DTD is not available for this document at this time.

6. Generate first pass xslt . All the custom xslts for the frm_* tables are added to the "first_pass_default.xslt" They ar in the appropriate context (i.e. question xslts will only be applied to that particular question).
7. Execute first pass xslt on "form_data"
8. Execute 2nd pass xslt on the resulting xml document. The second pass is simply to transpose child record tables, but additional functionality may be added in the future.

To debug the form processor, you may look at the formParameters object, form_data xml document, first pass xslt, and s pass xslt.

Cascading Style Sheets in Form Processor

Cascading Style Sheets drive much of the layout and appearance of form processor forms. There is no ability to add add external CSS sheets without using a custom page_page_xslt.

- "form.css" Defines style CSS for form processor
- "positioning_menu.css" Defines positioning when FRM_PAGES.menu_xslt exists. Positioning divs are class="banner", class="menu", and class="page"
- "positioning_no_menu.css" Defines positioning when FRM_PAGES.menu_xslt is NULL (no menu). Positioning divs are class="banner", class="menu", and class="page"
- "widgets.css" contains styles for widgets such as pagination widgets

XSLTs in Form Processor

Form-Record Adds

Adds are allowable when all the one-to-one tables that have add_when_parent_added set to true have:

1. add permissions given in stored procedures
2. 'add' in the mask

If any of the one-to-one tables that have add_when_parent_added don't meet both criteria, adds will not be allowed and link won't show up on the index. The actual link is inserted with the "index_xslt_default" xslt template. If you have a c index xslt, you will need to add your own add link. Add links One-to-many child tables should never have add_when_parent_added set to true.

Derivation and Execution of Add SQL

1. The record id of the new record is derived:
A. If the primary table has an auto increment id field (record_id_auto = true), its insert is done. The auto generated i field becomes the record id. The insert SQL is derived as in A-C of step 2.

B. If a“record_id_insert_param”, this param is used as the record id.
C. Othewise the value in the “record_id_field” is used as the record id.

2. For each record of each relation that has an add, generate SQL. All 1-1 relations will will have 1 add if add_when_parent_added is true and add is in mask. Otherwise 1-1 relations will not have an add, regardless of whether has entered data in questions from that relation. One to many relations will have as many inserts as the user has created, regardless of the value of add_when_parent_added, provided add is in the maks. In a 1-1-* form that has three child rec being added, there will be 5 inserts derived.

A. derive insert columns and values from questions.

Insert columns example: lname,fname, age

Insert rows example: smith, joan, 12

B. add primary_key_insert_columns and primary_key_insert_values to these after state tags ([record_id],[userna are replaced.

C. INSERT INTO [TableName] ([insert columnes], created_by) VALUES ([insert values], ‘[username]')

3. Execute all the inserts at once.

When do adds show up in the index page?

- permissions must allow it
- masks must allow it

Form-Record Deletes

For a given table, how a delete is performed depends on permissions and `the soft_delete` and `delete_when_parent_deleted` attributes of the relation/table.

Soft deletes.

Soft deletes are when a record is deleted by setting a bit field named `delete_requested` to 1. Soft deletes may be turned on for one or more tables in a form. They work in both one-to-one and one-to-many relationships and both soft and hard deletes may be used in the same form. To enable soft deletes on a table:

- make sure there is a bit field **`delete_requested`** in the table.
- The **`delete_requested`** field should NOT be the data_location in any questions in the form.
- set **`ED.FRM_RELATIONS.soft_delete`** = 1
- make sure delete is in set **`ED.FRM_RELATIONS.permissions_mask`**
- make sure the security on the table allows for deletes

Soft deletes will appear identical to standard deletes to the user.

Hard Deletes

These are performed when permissions allow and `soft_delete` is not set to True in a relation. Sibling and child records are only deleted when `delete_when_parent_deleted` is set to True.

Delete Action Summary

When a user deletes a form-record, The following occurs:

Record Type	soft_delete		delete_when_parent_deleted	Action Performed
parent	1		~~Will always be True for parent record~~	Set delete_request to 1

			T	
parent	1		F	No action
child or sibling	1		T	Set delete_request to 1 for all child records
child or sibling	1		F	no action
parent	0		~~Will always be True for parent record~~ T	delete record
parent	0		F	No action
child or sibling	0		T	Attempt to delete. Cascading delete may take care of this when parent deleted
child or sibling	0		F	No action

When the user deletes a child record of a one-to-many while updating the form-record, a soft-delete or hard delete is performed to that indidual record.

When do deletes show up in the index page?

Security

Form processor permissions are driven by DB groups maintained by the DBA. There are record level permissions for update, delete, and view and form level permissions for adds. Form masks may be applied to override the DB group security. Masks are specific to individual tables and are stored in `FRM_relations.permissions_mask` and whatever values present are the actions allowed… provided the DB group security also allows it.

Public forms are also available that bypass all security. These should have a mask of only add and, possibly, view to them since updates and deletes would be impossible without a security layer.

Customizing without XSLTs

Customizing with XSLTs

General

All XSLT customizations are implemented in the first pass XSLT. They are integrated into the XSLT itself before the first pass xslt is applied to the form_data xml document. The custom XSLTs that are inserted into the FRM_* tables, are not complete templates, but fragments that are inserted into templates. This is necessary because the template match and mode attributes would be difficult for the user to determine; thus the root element and params for the template are generated by the form processor while the ‘guts’ of the template are drawn from the frm_* fields.

```
<xsl:template ….>
      <xsl:param name="…
      <xsl:param name="…
      … xslt code from FRM_* tables
</xsl:template>
```

The nesting of templates is best documented at \\Ed-nts\Ed Admin\OET\ITGroup\Projects\FormApps\Processor\User Documentation\xslt_hierarchy.xls , though this document often lags behind development.

You may see the first pass xslt by turning on the debug cookie and adding **`&xslt=true`** to the URL

All xslt templates must produce valid xhtml. **`<br/> <img src="…." />,`** etc. The best way to find examples of custom XSLTs is to look at other forms or look for the alternative default XSLT template in the first pass XSLT. **\\ed-webdev\wwwroot\intranet\formapp\v3\first_pass_default.xslt**

XSLT Resources

TopXML

XML.Com

ZVON XSLT Reference

MSDN XSLT in MSXML4.0 Context

Global Variables Available in XSLTs

The following global variables are available within custom xslts since they are defined at the root level of the first pass x
Use `<xsl:value-of select="$<variable_name>"/>` to display their value in the xslt)

anchors

required_fields_marker

required_fields_sentence

appropriate_record_name_article
(a or an) (for parent record)

record_name_plural
(for parent record)

record_name
(for parent record)

friendly_action_name

username

uid

form_id

record_id

action

index_id

order_by_field

page

param1…param10

post_url
attempt at current url

post_actions
attempt at get part of url

html_head_additions_xslt

This is used to add content within the html head element such as javascript or css. It does not replace the form generato content. It is used to add javascript, css, and other head elements such as meta tags.

html_head_xslt

This is used to replace the enitre default html head element. It is used when you want all your own javascript, css, etc to in a form. If left empty it will not be used. This is a complicated xslt to write and the html_head_additions_xslt is prefe adding content to the head element.

Callable XLST Templates (Widgets)

These widgets can be called from custom xslts. They are often used in the default xslts also.

Index Links XSLTs

These widgets will create a menu of all the indexes for a given form. It is called by:

```
<xsl:call-template name="index_links_list" /> or
<xsl:call-template name="index_links_select" />
```

index_links_list result will look something like:

```
<ul class="index_links_list">
<li><a href="default.asp?form_id=3&index_id=56">By People</a><li>
<li><a href="default.asp?form_id=3&index_id=57">By Pets</a><li>
</ul>
```

index_links_select result will look something like:

```
<select name="index_links" class="index_links_select"
onChange="location.href=(this.value);">
      <option name="Select An Index">Select An Index</option>
      <option name="default.asp?form_id=7&index_id=5">By Name</option>
      <option name="default.asp?form_id=7&index_id=5">By Zip Code</option>
</select>
```

Index Filter

This is the xslt that shows the filter input box for the index.
`<xsl:call-template name=”index_filter”  />` The filter includes the directions, submit button, and input for text. It is only available on the index page, but may be anywhere on the page. The html of the index filter depends on the `question_html_structure` attribute of the section, which may be “`div`”,”`col`”,”`td`”, or “`span`” .

“div”

“td”

“span”

“col”

Add_link

This is the xslt that shows the link to adding a new record. It takes the link name as an argument.

Applied with:

```
<xsl:call-template name="addlink">
<xsl:with-param name="link_name" select=”’add an inventory item’" />
</xsl:call-template>
```

pagination_page_links

a list of up to 5 pages such as 2 3 4 5 6 which links to pages around the current page
Applied with: `<xsl:call-template name="pagination_page_links" />`

pagination_jump_form

A form used to jump to a given page.
Applied with: `<xsl:call-template name="pagination_jump_form" />`

pagination_arrows

|< < > >| style buttons for pagination
`Applied with: <xsl:call-template name="pagination_arrows" />`

pagination_summary

Produces the following HTML:

```
<span class=”pagination_summary”>page X of N</span>

Applied with: <xsl:call-template name="pagination_summary" />
```

pagination_total_records

Produces the following HTML:

```
<span class=”pagination_total_records”>X total records</span>
```

Applied With: `<xsl:call-template name="pagination_total_records" />`

pagination

This is the xslt that shows all the pagination widgets It contains all the preceding pagination_* xslts, which may be called individually to get the ala cart pagination widgets.

Applied With: `<xsl:call-template name="pagination" />`

section_header_xslt_custom

To replace the text in a section header, you may populate the field section_header_xslt_custom

The format of the header, will depend on the “question_html_structure” you choose and whether the section is one to many or not. Below are the default section header xslts which can be used as a starting point for custom xslts.

One-to-one, @question_html_structure='table'

```
<tr class="section_header" name="section_header" id="section_header" >
	<td colspan="2">
		<h2 class="section_title"><xsl:value-of select="@section_title" /></h2>
	</td>
</tr>
```

One-to-one, question_html_structure='div' or @question_html_structure='col'

```
<div class="section_header" name="section_header" id="section_header" >
	<h2 class="section_title"><xsl:value-of select="@section_title" /></h2>
</div>
```

One-to-one, question_html_structure='span'

```
<span class="section_header" name="section_header" id="section_header" >
	<h2 class="section_title"><xsl:value-of select="@section_title" /></h2>
</span>
```

One-to-one, otherwise:

```
<tr class="section_header" name="section_header" id="section_header" >
	<td colspan="2">
		<h2 class="section_title"><xsl:value-of select="@section_title" /></h2>
	</td>
</tr>
```

One-To-Many

```
<h2 class="section_title"><xsl:value-of select="@section_title" /></h2>
```

Configuration Data available to Custom XSLTs.

Contains all the data in the frm_* tables.

Target Data available to Custom XSLTs

Response Data available to Custom XSLTs

State Data available to Custom XSLTs

Supplementary Data available to Custom XSLTs

Extra data may be brought into a FP form and access through custom XSLT. It will be contained in the <supp_data> element which is outlined in Appendix D: Form_data elements.

Intro Pages

Intro pages are simply a splash page for a form. The advantage to them over simply making an html page is that the intro page XSLT will always have the look and feel of the form processor and may call certain XSLT templates such as the index links. To create an intro page, simply add a form page wrapper (FRM_PAGE_WRAPPERS) with page_action = 'intro'. Populate the directions_xslt and the form_xslt with any valid xslt fragment (should be well formed xml but doesn't need a root element).

Index Pages

Records SQL

The sql used to query the index is stored in 4 fields: **`record_index_select`**, **`record_index_from`**, **`record_index_where`**, and **`record_index_order_by`**. All of these fields may have state tags in them. The references to tables and fields must be explicit in these expressions. <DB>.dbo.<TABLE>.<FIELD_NAME>. They are edited in the ***INDEXES*** interface.

Default Index. True or False to designate the index will show up when no index is specified. If no default index exists, none will be show.

Defautlt to Empty on Load. True or False. Important for index which require a long time to process. See https://www-s1.ed.uiuc.edu/intranet/formapp/v3/default.asp?form_id=135

A Note From the DBA

The index queries tie to the frm_relations table in a few critical ways. Below are some notes on this topic. I'll formalize these notes later (i.e. put in documentation).

- the index query must join on the primary table[s] in the FRM_Relations table (The "One" sides of the one-to-many)
- the value in the first column in the index query must either be the key from the Primary table or match to it in a foreign/primary key relationship
- tables referenced in the index query do not need to be in the frm_relations table unless a question references it.
- permissions on tables only referenced in the index query need only be on ed-mssql for ed-charlotte/intranet_role (user simply has to have permission on the joined-to primary table)
- tables referenced in filter questions must be in the index query
- tables referenced in filter questions do not need to be in frm_relations

Here's a sample scenario (see form 114):
An sql view is used in the Index query so that more powerful sql can be run and logic can be shared with other applications. However, the sql view is not updated or referenced by any target questions. Therefore, no relation exists for this table in frm_relations.

Several "filter" questions reference the view.
The index query joins the view to the primary table in the from clause.

Directions

Directions for index pages may be kept in two places:

- FRM_PAGE_WRAPPERS where page_action=index in the directions_xslt field. If directions are stored here they apply to all indexes for the form, unless directions exist in FRM_INDEX.index_directions
- FRM_INDEX.index_directions If directions are stored here, they only apply to this index.

Both are applied as XSLTs and may contain any XSLT elements.

Pagination

The number of pages on an index is that same for all the indexes of a form. It is edited in the ***FORM*** interface in the page_length field (FRM_FORM.page_length). The default pagination is 25 records per index page. Pagination widgets may be added to the menu or header or other custom xslts as outlined in the widgets section.

Order By and Sorting

The **`record_index_order_by`** is the default order by used if the user has not sorted the index.

The **`record_index_select_fields`** are the order by clauses that map to each of the select fields. They are delimited by "|". This field makes user selected sorting possible. An example best illustrates how this field is used. There must be the same number of fields in Record_index_select and Record_index_select_ fields! To make a field not sortable, put "NULL" in that position instead of sort fields. That column header in the index will not have a sort link in it.

Record_index_select (an SQL fragment)

```
ED.dbo.INV_ITEMS.item_id as record_id, ED.dbo.INV_ITEMS.TYPE, ED.dbo.INV_ITEMS.fname +
ED.dbo.INV_ITEMS.lname as name
```

Record_index_select_fields (Not an SQL fragment)
NULL|NULL| ED.dbo.INV_ITEMS.fname, ED.dbo.INV_ITEMS.lname

When the user sorts on column 3, ORDER BY ED.dbo.INV_ITEMS.fname, ED.dbo.INV_ITEMS.lname
Is passed into the stored procedure to make the sort.

The record_index_select field must be used to accomplish the sort, since passing `ORDER BY ED.dbo.INV_ITEMS.fname + ED.dbo.INV_ITEMS.lname as name` or `ORDER BY 2` would make the stored procedure fail.

- Don't forget to have the same number of fields in the `record_index_select_fields` as `record_index_select`. Even though the first column of `record_index_select` is the record id and is generally not displayed, its counterpart must be included in `record_index_select_fields` as in the example above.
- Don't forget to use "|" as the delimiter in `record_index_select_fields` and "," in `record_index_select`.

Filters

Each index may have a filter section in it. The filter section must have one or more filter questions in it. This is accomplished by creating a section with the class attribute set to "filter" and adding as many questions as you like to it. For a given session, the filter values remain with the user when they go back and forth from index to index or index to detail record. The user can only reset the filter with the reset filter button or by resubmitting the filte form.

Add a question for each filter you want available, paying particular attention to the following question attributes:

- **input_type** … may be of any type that is useful in a filter. It will likely be radio, select, or text. drilldown questions may be desirable.
- **data_location …** The field to be filtered on. It must be an actual, explicit field in that will work in the where_clause of the FRM_GetIndex_rs stored procedure.
- **value_list_id** The input values in value lists are values to be filtered on. The values "IS NULL" and "IS NOT NULL" will filter on IS NOT and IS NOT NULL. These could be typed directly into a text field or set in a special value list for filtering. For radio and select lists, it is best to create a new list such as the example below for responsible person in the inventory table that includes values for IS NULL, IN NOT NULL, and NULL (no filter) :
    - No Filter => NULL
    - No One Entered => IS NULL
    - Somebody Entered => IS NOT NULL
    - Jim Jones => 123
    - John Barclay => 645
    - …

- **input_default** … The default filter will be no filtering, unless a filter question has an input_default value. The input_default field may take state tags. So if you wanted a filter to default to the user's people id, your would use [people_id] as the input_default for the question.

After the filter where clause is derived from the user submitted values and the FRM_index..record_index_select field, it is passed into the FRM_GetIndex_rs stored procedure as the @where_clause parameter.

Supplemental Data

Supplemental data may be added to the <form_data> element to be parsed by custom XSLTs on the index page. The function executed to get this data is entered in the supp_data_function_id field on each index.

Customizing the index rows

Index rows are customized with the index_item_xslt in the indexes form. This xslt template is applied to each record in the index recordset. The default index item xslt looks like the following:

```
<xsl:template match="/form_data/index_data/rs:data/z:row" mode="index_item_xslt_default">
  <xsl:variable name="rowID" select="@RecordID" />
  <tr>
     <xsl:for-each select="/form_data/index_data/s:Schema/s:ElementType/s:AttributeType[position() &gt; 1 a
             position() &lt; (last() -4)]">
        <td class="data">
            <xsl:value-of select="/form_data/index_data/rs:data/z:row[@RecordID=$rowID]/@*[local-name()=
                    current()/@name]" />
        </td>
    </xsl:for-each>
    <td class="action_links">
     <xsl:call-template name="index_action_links">
        <xsl:with-param name="Row" select="."/>
      </xsl:call-template>
    </td>
  </tr>
</xsl:template>
```

When using a custom index_item_xslt the text in brown below is added around it. The text in green (below) is the text t
form designer would need to emulate or deviate from. The green text is the same as in the default index_item_xslt abov
first `for-each` loop generates a table cell for each field with the exception of the record_id and action fields. To work w
table header which names the columns, the same number of table cells should be spanned by the index_item_xslt_custo
xslt that generates the index header. The `call-template` near the end generates the action links.

```
<xsl:template match="/form_data/index_data/rs:data/z:row" mode="index_item_xslt_custom">
  <xsl:variable name="rowID" select="@RecordID" />
  <tr>

     <xsl:for-each select="/form_data/index_data/s:Schema/s:ElementType/s:AttributeType[position() &gt; 1 a
             position() &lt; (last() -4)]">
        <td class="data">
            <xsl:value-of select="/form_data/index_data/rs:data/z:row[@RecordID=$rowID]/@*[local-name()=
                    current()/@name]" />
        </td>
    </xsl:for-each>

    <td class="action_links">
     <xsl:call-template name="index_action_links">
        <xsl:with-param name="Row" select="."/>
      </xsl:call-template>
    </td>

  </tr>
</xsl:template>
```

To understand custom index_item_xslts, one should look at the `index_data` element in the configuration xml document. logging is turned on, it is located in \\ed-webdev\l$\[username]\[form_id]\index_data.xml The `rs:data/z:row` are what the xslt is being applied to.

An example of a custom index_item_xslt from form #153 follows. Note that the xsl:template element is not in the custo it is added when the form processor generates the first pass xslt. The purple text is the deviation from the default index_item_xslt.

```
<tr>
  <td class=”data”>
    <a href="default.asp?form_id=148&amp;action=view&amp;record_id={@RecordID}">
      <xsl:value-of select="@c1"/>
    </a>
   </td>
   <td class="action_links">
     <xsl:call-template name="index_action_links">
       <xsl:with-param name="Row" select="."/>
     </xsl:call-template>
   </td>
</tr>
```

Triggers

Triggers are actions that are performed after the SQL for a page is executed. When they are performed depends on the action and type of trigger. Several triggers may be associated with the same action and they will be executed in the order that the ‘order’ field dictates. They are executed after the Target SQL is executed.

State tags may be used in the trigger source field. i.e. [username] or [record_id]

Triggers for updates will be executed regardless of whether the user actually changes any data. Thus their may be no sql executed on the target data, but the update trigger will be performed.

If errors are encountered before the trigger is executed, processing will be halted and the triggers will not be executed. If errors are encountered during an sql trigger, processing will be halted. httprequest errors should be returned in the xml document itself to be displayed…otherwise they will be ignored. Redirect and terminal page errors will show up in the user’s browser window.

- Redirect triggers should always come last and there should only be one. Even if they are not ordered to be last they will be executed last, though only the first redirect will be executed.

Types

- **`httprequest. NOT IMPLEMENTED.`** This trigger request a web page which may return an xml document. It must be passed over SSL if the query requires ed-charlotte to authenticate. State tags may be used in this. The tags [admin-pwd] and [admin-username] may also be used, provided the request is over https to one of our servers https://www-s1... Or https://www-s.ed..... , substitution in url. i.e. https://www-s1.ed.uiuc.edu/intranet/createExtUser.asp?username=[param1]&pwd=[admin-pwd]&usrname=[admin-username] The

- **`sql`**. An sql statement to be executed.
- **`redirect`**. A redirect to another page. The first redirect will be used if more than one redirects exists for a given action. This redirect will be performed after all other triggers are executed and email notification and SQL logging has occurred.

- **terminal**. The browser window becomes a terminal page with a message and a 'close window' button. The message is the html is stored in the 'trigger_source' field of the trigger. Could be a thank you message etc. The results of the add, update, etc are also displayed. An example of this is form 132 which adds an organization to the OAR.

Question Attributes

Data_Location

The location the target data is stored. Should include full, explicit reference such as ED.DBO.INV_AGREEMENTS.DATE_START

xpath: /form_data/configuration/section/question/@data_location

Data_Area

xpath: /form_data/configuration/section/question/@data_area

Public_Use

Question_Name

The name of the question. ie. Favorite Color. Up to 60 characters

Question

The text of the question. ie. Please select your favorite color. If additional help is needed it can go in the help field.

Help

The text that comes up in the help box for that question. If help text exists, an information icon will be present that has a javascript link to the information pop-up window.

Data_type

This determines the type of validation to apply by (if Validation = True). Each type may have an associated server and/or side validation built into the form processor as well as filtering functionality. In cases such as UIUCadusername that can client side validation, validation will only occur server side. The following chart describes the effect of a data_type on validation and filtering. See also built in validation and filters.

Chronological Order of processing:

```
Client on change filter -> Client On Submit Filter -> Client Validation -> Server Validation –> Filter
```

Implementing new data types:

- add to form manager value list
- document here

- if data type needs client side validation
  - if data type needs a new fvalidate function, add it to fValidate.COECustomValidators and add its error message to fValidate.lanf-enUS.js
  - add case to getfcCode switch statement
- add type to question parameters validation in COEFPFormParameters.wsc on the line after arrQuestions(3,FIELDNAME) = "data_type"
- if data type needs client side filtering, integrate into filterOnChange functioon in fpfunctions.js
- implement in built in data types in validation COM object
- don't forget to test by applying it to a question by selecting it as the data type and selecting validation

Data Types and Associated Behaviors (when fully implemented)

Data Types In Detail

Client side validation is done with a modified version of Peter Bailey's favlidate.

Client side validation errors for text data type.

p=field is submitted. NA=not applicable. Please Enter = message will say something like "Please enter nickname". Ma
Smaller = message will say something like "nickname must be less than 100 characters, your nickname is 121 characters

Perm utation	Size limit	Null able	Valid ation	Fvalidate code	When Left Empy	When Left Oversized	Leav correc	Comments
1	x	x	x	length||n|bok	p	Make smaller	p	
2		x	x	null	p	NA	p	
3	x		x	lengthcoe|-1|n	Please enter	Make smaller	p	
4			x	blank	Please enter	NA	p	
5	x			blank	Please enter	p	p	
6		x		null	p	NA	p	
7	x	x		null	p	p	p	
8				blank	Please enter	NA	p	

alnum

data type	**alnum**
client filter	none
validation	A-Z a-z 0-9 only are allowed
client validation	yes
fvalidate params	`alnum|*|*|1|0|none[|bok] wher bok is included when nullable`
server validation	Yes
comments	

alpha

data type	**alpha**
client filter	none
validation	A-Z a-z only are allowed
client validation	yes
fvalidate params	`number|1|none[|bok] wher bok is included when nullable`
server validation	Yes – not implemented.

comments	

areacode

data type	**areacode**
client filter	Remove all spaces
validation	Must be a 3 digit integer
client validation	yes
fvalidate params	`areacode[|bok] wher bok is included when nullable`
server validation	Yes
comments	

boolean

`data type`	**boolean**
client filter	none
validation	Only values that could populate a bit field. 0,1, or NULL
client validation	no
fvalidate params	
server validation	yes
comments	

cdata

`data type`	**cdata**
client filter	none
validation	None
client validation	no
fvalidate params	
server validation	no
comments	Data type is only to make sure data in xml documents are encloded in CDATA elements.

currency

`data type`	**currency**
client filter	Remove all spaces, $, and commas
validation	NNN[.NN] after spaces, commas, and dollar signs are removed should be an integer or an integer followed by exactly 2 digits.
client validation	yes
fvalidate params	`decimaldollar[|bok] wher bok is included when nullable. Decimaldollar is our own custom type and is in the file fvalidate.special.js`
server validation	yes
comments	

date

data type	**date**
client filter	Remove all spaces. Replace – with / If year is two digits make 20XX if XX <=60 othewise 19XX
validation	A date with either – or / may be used as a delimiter. Must be a valid date also, not Feb 30th
client validation	yes
fvalidate params	`date|mm-dd-yyyy|/-|0|0[|bok] wher bok is included when nullable`
server validation	yes
comments	

datetime

data type	**datetime**

client filter	Remove all spaces. Replace – with / If year is two digits make 20XX if XX <6=60 othewise 19XX
validation	A date with either – or / may be used as a delimiter. Must be a valid date also, Time may be in the following format: 6[:30][ ]pm|PM|am|AM time part may also be completely left off
client validation	yes
fvalidate params	
server validation	yes
comments	Needs to go to the server as: dd/mm/yyyy NN:NN PM|AM

email

`data type`	**email**
client filter	Remove all spaces
validation	Must be valid email address in format: /^\w[\w\d]+(\.[\w\d]+)*@\w[\w\d]+(\.[\w\d]+)*\.[a-z]{2,7}$/i
client validation	yes
fvalidate params	`email|3[|bok] wher bok is included when nullable`
server validation	Yes
comments	

emaillist

`data type`	**emaillist**
client filter	Remove all spaces
validation	Must be one or more valid email addresses delimited by commas in format: /^\w[\w\d]+(\.[\w\d]+)*@\w[\w\z]{2,7}$/i
client validation	yes
fvalidate params	`emaillist[|bok] wher bok is included when nullable`
server validation	Yes
comments	

file

`data type`	**file**
client filter	none
validation	none
client validation	
fvalidate params	
server validation	
comments	Complete path to a file, including filename i.e. \\www.ed.uiuc.edu\libray\my.html

filename

`data type`	**filename**
client filter	none
validation	none
client validation	no
fvalidate params	
server validation	no
comments	Name of a file, excluding path. i.e. my.txt

fourdigityear

data type	**fourdigityear**
client filter	Remove all spaces

validation	Between 1900 and 9999
client validation	yes
fvalidate params	`fourdigityear .  bok is included when nullable`
server validation	Yes
comments	Only applicable to inputs of type text, textarea and password

html

`data type`	**html**
client filter	none
validation	None. Html has too many flavors to validate.
client validation	no
fvalidate params	
server validation	no
comments	This datatype is to designate that text may contain html elements that are not valid, but should not be escaped In final output . This datatype should be avoided as it creates great problems in xslts attempting to create html, Pdf, word, etc formats.

integer

`data type`	**integer**
client filter	Remove all spaces and commas
validation	A positive or negative integer
client validation	yes
fvalidate params	`number|none|none[|bok] wher bok is included when nullable`
server validation	Yes
comments	

natural

`data type`	**natural**
client filter	Remove all spaces and commas
validation	Integer > 0 (0 is not valid)
client validation	yes
fvalidate params	`number|1|none [|bok] wher bok is included when nullable`
server validation	Yes
comments	

number

`data type`	**number**
client filter	Remove spaces and commas
validation	Must be a valid integer or decimal
client validation	yes
fvalidate params	`number[|bok] wher bok is included when nullable`
server validation	Yes
comments	

palindrome

`data type`	**palindrome**
client filter	none
validation	Must be the same spelled backwards

client validation	no
fvalidate params	
server validation	Yes
comments	

percent

`data type`	**percent**
client filter	Remove spaces and commas and percent signs
validation	Must be a valid integer from 0 to 999
client validation	yes
fvalidate params	`percent[|bok] wher bok is included when nullable`
server validation	Yes
comments	

state2char

data type	**state2char**
client filter	Remove all spaces
validation	Must be U.S. state abbreviation as defined in fpfunctions.js
client validation	yes
fvalidate params	`State2char[|bok] wher bok is included when nullable`
server validation	Yes – not implemented.
comments	

text

`data type`	**text**
client filter	none
validation	None
client validation	no
fvalidate params	
server validation	no
comments	Deprecated. Should be replaced by cdata, xhtml, xml, xmltext, etc. applications may treat this as Cdata, html, etc as they did when the datatype became deprecated.

UIUCADusername

`data type`	**UIUCADusername**
client filter	none
validation	An existing AD username.
client validation	no
fvalidate params	
server validation	Yes . Must be in UIUC AD
comments	

url

`data type`	**url**
client filter	none
validation	Must be a valid URL syntax starting with http or https. Must also be an existing web page.
client validation	yes
fvalidate params	`url|http,https|1|1[|bok] wher bok is included when nullable`

server validation	yes
comments	Users don't like to put the http:// part in urls so its not a great datatype to require validation on

xhtml

data type	**xhtml**
client filter	none
validation	well formed xhtml
client validation	no
fvalidate params	
server validation	yes . may be a fragment without a root element. Its process consuming to validate xhtml, so it is validated as xml
comments	Fields of this data type should have validation turned on if xslts are to be applied to the data. Otherwise bad data will cause processing failures.

xhtmltext

data type	**xhtmltext**
client filter	none
validation	well formed xhtml
client validation	no
fvalidate params	
server validation	yes
comments	Text that is xml compliant. Should not include elements, just characters: A-Z, a-z, 0-9, _ and defined xml Character entities: " < > & . Basically all the valid text that could be in an xml attribute value Or within an xml element. See http://www.webreference.com/xml/reference/xhtmlchars.html for encoding additional characters

xml

data type	**xml**
client filter	none
validation	well formed xml
client validation	no
fvalidate params	
server validation	yes . may be a fragment without a root element.
comments	

zip

data type	**zip**
client filter	Remove all spaces
validation	Must be valid five or nine digit zip in format 12345 or 12345-6789
client validation	yes
fvalidate params	`zip[|bok] wher bok is included when nullable`
server validation	Yes
comments	

xml - escape bad characters
http://www.w3schools.com/xml/xml_parser.asp
-http://xml.coverpages.org/xml.html

- http://www.v2studio.com/k/code/xmlchecker/
http://msdn.microsoft.com/downloads/samples/internet/default.asp?url=/downloads/samples/internet/xml/xml_validator
sp
http://perso.wanadoo.fr/ablavier/TidyCOM/
http://users.rcn.com/creitzel/tidy.html#dotnet

Nullable

Determines if the the question is required.

Validate

determines if validation should be attempted on both server and client side. If no datype or function is available for val
none will occur.
.

Input_type

Type of html and or javascript widget used to input value.

Input Types and Behaviors

Input_type	***Html input type***	***Addt'l elements***	
• text	text		x
• textarea	textarea		x
• hidden	hidden		x
• display	none		x
• select	select		x
• date	text	Date selector javascript widg	x
• radio	radio		x
• checkbox	checkbox		x
• htmltextarea	Textarea or GUIon IE 4+	Not supported yet	
• fileuploadtoserver		Not supported yet	

• Button	button	Not supported yet	
• Password	Password		

Input_Size

Only used with input_type of ‘text’ and ‘password’. This is the size attribute of the html input element.

Input_maxlength

Specifies the maximum length of the input. This is only applicable for questions with input_types of: text, password, te and value lists with "other" option. It sets the ‘maxlength’ attribute of the html input element. It is required for these input_types.

Input_rows and Input_cols

Only relevant to input_types of textarea and htmltextarea. Rows and columns of box.

Input_default

Specifies value of questions. The form processor will only populate this for Adds. Updates will never have a default val This is not implemented because it is a bad idea. If a field must be populated, make it not nullable and give the user son picking the default value. This avoids the user accidentally selecting a default value in adds.

Value_list_id

Id of value list associated with question.

Value_list_allow_other

Should the user be able to hand type in a value that is not on the value list? If an other is allowed, it will not have a data validation associated with it other than the input_maxlength.

Value_list_allow_multiple

For checkboxes and multiple selects, should each value go in a separate record of a table. If this is false, data will go int comma separated format in a single field. This field is not used because we haven’t had a use for it. All checkboxes are assumed to be single Boolean values. Multiple selects are not supported.

Multivalues_in_related_table

For checkboxes and multiple selects, should each value go in a separate record of a table. If this is false, data will go int comma separated format in a single field. This field is not used because we haven’t had a use for it. All checkboxes are assumed to be single Boolean values. Multiple selects are not supported.

Value_list_standard_search

Should the user be able to search on this field using the standard search widget? This only applies to inputs with value l input_type select. Prerequisite #1: value_list_standard_search

Both List-Based and SQL-Based Value Lists can be searched. SQL-Based lists that are searchable need the tag [search_str] put in the sql expression such that a replace of the string and execution of sql would yield a "found set." The sql expression is stored in FRM_FUNCTIONS.expression.

Example:

```
SELECT ED.dbo.ED_ORGANIZATIONS.organization_id, organization_name FROM
ED.dbo.ED_ORGANIZATIONS WHERE organization_name_sort like '%[search_str]%'
ORDER BY 2
```

This function and value-list would only work with Searchable value lists.

A corresponding decode of this value list is also needed. The FRM_FUNCTIONS.expression_decode field needs to contain sql such that a replace of the string [input_value] and execution of sql would yield one human readable input_name value.

Example:

```
SELECT ED.dbo.ED_ORGANIZATIONS.organization_id, organization_name FROM
ED.dbo.ED_ORGANIZATIONS WHERE ED.dbo.ED_ORGANIZATIONS.organization_id = [input_value]
```

This decode is needed so that on an update or viewing of a form, the form processor can present the decoded value.

limiting_question_id

To make a one questions value list dependent on the value of another question, do the following:

limiting question

- Definition: the question whose value affects which values are available in the dependent question
- can of any input type, though radio and selects behave best.
- i.e. limiting question emp class (question id = 199) is a limiting question with the following value list:

input_value	input_name	drilldown_parent_value
A	Academic	NULL
C	Civil	NULL
F	Faculty	NULL
S	Student	NJULL

dependent question

- Definition: the question whose value list is changed depending on the limiting qestion
- should be of input type ‘select’ or another input type that uses the HTML select input.
- limiting_question_id points to the question(s) whose value lists depend on this question
- Third column of value list should be populated with values that match potential values in the limiting question.
- i.e. emp subclass (question id = 257) has the following value list and has `limiting_question_id` set to 199.

-

input_value	input_name	drilldown_parent_value
Academic Hourly	Academic Hourly	A
Fello	Fellow	S
GA	GA	S
GH	GH	S
NonTenure	NonTenure	A
NonTenure	NonTenure	F

Desired Behavior of limiting/dependent questions

The limiting question may be in any table (one-to-one or one-to-many). The dependent question may also be in any tabl
the following exception:

Limiting Question Location	Dependent Question	Dependent records affected by changing limiting question value
one-to-one table	one-to-one table	affects single dependent question
one-to-one table	one-to-many table	affects dependent question in all child records
one-to-many table	same one-to-many table	Affects only dependent question in the same child record
One-to-many table	Another one-to-many table	NOT ACCEPTABLE
one-to-many table	one-to-one table	NOT ACCEPTABLE

The initial value list of a dependent question will be based on the initial value in the limiting question. If it is an unpopu
question, the value list will contain all the values which have a `drilldown_parent_value` of NULL.

Value_list_allow_other

When set to true, this allows an other text box to appear next to value lists. This only applies to questions of type radio a
select.

Value_list_standard_search

When set to true, this allows a search button to show up next to questions. It only works for questions which use the HT
'select' element such as when input_type = 'select'

List_item_xslt

Only applies to questions with input lists. Xslt to be performed on each item in input list. That is, each select option or
button of the list. This xslt will have the following parameters passed to it: `value_list_id, label, value, drilldown_parent_value, selected, id,` and `input_name`. It is not a full xslt template, just a fragment. Note that the N
value_list_id is the id of the question in the configuration xml document. It is not the id of the value list in the sql tables

Examples might be:

Customized select option:

```
<option value="{@value}" >
      <xsl:if test="@selected='true'">                                                    <xsl:attribute
name="selected">true</xsl:attribute>                                                  </xsl:if>
      <xsl:value-of select="@label" />
</option>
```

Customized radio list item:

```
<input type="radio" name="{$input_name}"  value="{$value}" >
      <xsl:if test="@selected='true'">
            <xsl:attribute name="checked">true</xsl:attribute>
      </xsl:if>
</input>
<xsl:value-of select="@label"/>
```

question_xslt

This is the xslt for the entire question. It encompasses the question_name, question_question, input_xslt and all other question features including javascript, help button, etc. The question xslt is not a complete xslt template, just its innerds. It has the following parameters passed into it: record_id, stored_value, submitted_value, and question_id. From these parameters, it needs to construct the html for the question and input. This construct is best avoided.

question (should be called question_text_xslt)

This is the xslt for the question text. That is the text between the question_name and the question_input.

input_xslt

question_trailer_xslt

This is the xslt that follows the input of a question. The example below shows how the question_trailer_xslt is used. The contents of the question_trailer_xslt are in green below. Question trailer xslts have the following params and variables available to them: question_id, record_id, relation_id, and input_name. In the example below, a button is added to the question which when clicked executes a funtion. The input_name is passed to that function so that the function may affect the input value on the original form.

```
<!-- start  ## 1382 question trailer xslt ## -->
      <xsl:template match="/form_data/configuration/section[@section_id=1]/question[@question_id=1382]"
mode="question_trailer_xslt_custom">
            <xsl:param name="question_id"/>
            <xsl:param name="record_id"/>
            <xsl:param name="relation_id"/>
            <xsl:variable name="input_name">
                  <xsl:text>input</xsl:text><xsl:value-of select="$input_delimiter"/><xsl:value-of
select="$relation_id"/>
                  <xsl:value-of select="$input_delimiter"/>
                  <xsl:value-of select="$record_id"/>
                  <xsl:value-of select="$input_delimiter"/>
                  <xsl:value-of select="$question_id"/>
            </xsl:variable>
            <xsl:element name="input">
                  <xsl:attribute name="type">button</xsl:attribute>
                  <xsl:attribute name="value">Help Me Decide</xsl:attribute>
                  <xsl:attribute name="onclick">chooseTransport('form1.<xsl:value-of
select="$input_name"/>');</xsl:attribute>
```

```
            </xsl:element>
        </xsl:template>
<!-- end  ## 1382  question trailer xslt ## -->
```

Question Input Types in Detail

Checkbox.

A checkbox can only be used for boolean values. A checkbox field will either populate the field with 0 or 1 when an update is performed, regardless of the value list associated with the question or the initial data in the field.

To implemenent a checkbox make the data_type boolean and the input type to checkbox.

Value lists may be associated with checkboxes, but they serve no purpose.

Checkboxes have some user interface problems associated with them. If the user leaves a checkbox unchecked does that assume they have explicitly set the boolean value to '0'? In our implementation we assume this is the case. The alternative is to assume that an uncheck checkbox means they never really thought about it or "left it alone" and its value should be NULL.

Rather than have NULL, 0, and 1 in a field using a checkbox, it may be better to have 0 and 1 and make 0 the default initial value in the field. This avoids the problem of differentiating between NULL and 0 when the form processor doesn't.

Display.

Select

Date

Htmltextarea

Future implementation. The idea of this input is to use a textarea to edit HTML that will go into the target field. Depending on the browser, this may be implemented as a wysiwyg box. A help box with html tags is also available. See
(PageCreator - WYSIWYG (What you see is what you get) HTML Editor)
http://www.ed.uiuc.edu/dev/pagecreator/

Fileuploadtoserver

Future implementation. The specifics of this are unclear at this point. The design should be as simple as possible to avoid creating a document management solution. It may involve a new configuration table (FRM_files). Some parameters might include:

- overwrite existing files = T | F

- create directories if needed = T | F
- filepath = filepath with possible tags such as [username],[month],[day],[year],[userdefinedfilename] i.e. \\ed-nts\ed admin\oet\vacationpictures\[username]\myvacation_[year]_[month]_[day]_[userdefinedfilename].[filextension]
- filesizelimit = in kb
- http://www.asp101.com/articles/jacob/scriptupload.asp
- http://www.dundas.com/index.aspx?Section=DundasUpload&Body=Features.htm
- http://asp.aspobjects.com/ASP_Components/File_Management/Upload_Download/

Validation

Required Fields. (Validation type #1)

When question.nullable = 'Y' then javacript will be added to require a field be populated, regardless of whether question.validation is Y or N, the data type, or input_type. It will also be checked server side. If a left null and is nullable, it will not be validated in #2 and #3.

Built In Data Types and Filters. (Validation type#2)

This type of validation is executed if:

- question.validation is true
- AND a data type that has a built in validation is selected for the question. For example email or integer.
- AND the form-record or child record is being updated or added (not if no action is selected or it is being deleted

The form designer has no ability to change this behavior beyond turning validation off. Details of behaviors for specifi data_types are listed in Data Types and Associatied Behaviors table.

Not Implemented: Custom Single Question Validations and filtering. (Validation type #4)

These are signified by FRM_FUNCTIONS.medium = 'sql' | 'eval'
and FRM_FUNCTIONS.type='datatype' If question.validation = 'Y', server side validation will occur using the expression. If FRM_FUNCTION.javascript exists, it will be used for client side validation and filtering.

FRM_QUESTION_x_FUNCTION.
function_id
error_wrapper is displayed if result = false

Not Implemented: Composite validation. (Validation type #5)

Composite validations look at the entire form. Javascript is executed when the submit is hit, and server side is executed validations 1-4.

FRM_COMPOSITE_VALIDATIONS (aka FRM_FORM_x_FUNCTION)
- .id
- .form_id
- .function_id
- .error_wrapper

FRM_FUNCTIONS.medium = 'sql' | 'eval'
FRM_FUNCTIONS.type='boolean'
FRM_FUNCTION.javascript if exists, it will be used for client side validation when submitting.
- javascript will take form object as input and return error message in popup window, such that a the form onSubmit(.....) can be called if FRM_FUNCTION.javascript exists

FRM_FUNCTIONS.expression can have data tags such at:
- <tag context="parent-record|child-record" class="targetdata" db="ED" table="COMMITTEES" field="n
- tags for repeating child record field data in this context will not be implemented

Here are some new fields that may belong in FRM_FORM_x_QUESTIONS, FRM_QUESTIONS, or FRM_VALUELIS

VL_Add_Value_From_Standard_Search_JS_Call. The JS that opens a new form for adding a value to a value list. This be executed from a button on the standard search widget.

VL_Custom_JS_Buttons. i.e.JS button that calls the new window used to search and adding.
VL_Add_Values. 0|1. Should the user be able to add values to a value list.
VL_Drill_Down_Question_ID. The dependent child question ID for drill down questions.

Display_Only = 0 | 1. Is the question for display only? It may be of any type of input (text, radio, select, etc), but data i target data tables. This may be a select used in a drilldown or a text field that just needs to be viewed. If the type of inpu "none" or NULL then it will behave as display only fields currently do.

Relevant, existing fields:

question.input_type. The HTML input type to be used. Select, Radio, Checkbox,....

New Tables:
FRM_QUESTION_JAVASCRIPT
id
form_x_question_id. the form_x_question this js needs to be applied to
event. The event that will execute the JS. ie onClick if null then a button will be added to the question to execute it, wit button labelled with button_label field.,
button_Label.
rhs. The right hand side of the execution i.e onClick="validateme(form1.[question_name])";
headJS. the JS that goes in the head of the document between the script tags.
execution_order. the order that this particular script should be executed, relative to other JS with the same event
headerJS. addtional JS that needs to be added to <script></script> block with html head block.

CSS

The form processor relies heavily on CSS for formatting.

To apply CSS, you need to understand the model for the HTML structure for form pages. It is at: \\ed-nts\Ed Admin\oet\ITGroup\Projects\FormApps\Processor\User Documentation\form_dom.xml and can be opened with a text editor.

The default CSS is at https://www-s.ed.uiuc.edu/intranet/formapp/v3/css/form.css

Generally, to override a default CSS rule, you:

1. add a link to a style sheet in the with a link such as: <link rel="stylesheet" href="https://www-s.ed.uiuc.edu/intranet/formapp/v3/forms/134/form134.css type="text/css"/> in the html_head_additions_xslt or html_head_xslt in FRM_PAGES
2. In that style sheet add a rule with higher specificity than in the at https://www-s.ed.uiuc.edu/intranet/formapp/v3/css/form.css Specificitiy is explained on page 62 of eric meyer's cascading style sheets the definitive guide and on the w3c site: http://www.w3.org/TR/REC-CSS2/cascade.html#specificity. Generally the trick is to use the same selector, but add body#<form_id> to the front of it.

   For example in form.css, the default css, the table attribtutes for a one to many veritcal section are set by the rule:

```
table.one_to_many_h_section, table.one_to_many_v_section, table.one_to_one_section,
table.one_to_one_horizontal_section {
            margin-top:     5px;
            margin-left:    0px;
            margin-right:   20px;
            margin-bottom:  5px;
            padding-top:    <%= RegionTablePadding/2 %>px;
            padding-left:   20px;
            padding-right:  10px;
            padding-bottom: <%= RegionTablePadding/2 %>px;
            font-family: arial;
            font-size: x-small;
            font-weight: light;
            background-color: #<%= SecondaryBGColor %>;
            border: 1px solid #<%= SecondaryBGColor %>;
}
```

   To override this and make the background color of one to many vertical sections teal, use the following rule in your own style sheet.

```
body#134 table.one_to_many_v_section  { background-color:  Teal  }
```

   additional examples can be found in: https://www-s.ed.uiuc.edu/intranet/formapp/v3/forms/134/form134.css though this page is not actually used by form 134 anymore.

Appendix A: Migration and Documents Summary

Migrating the same version from Dev (ed-webdev) to Production (ed-w3)

If the version is the same, the following needs to be done to move the entire application to production. This does not include the shared dependent files such as COEValidationFunctionsV1.wsc, DBFunctions, etc. When these files are modified on the dev server, they should be moved to production as soon as they are shown to

work with all the applications that use them. If this can not be done, then multiple versions of these classes should be used.

- on production server, make a backup copy of all files in \\ed-w3\w$\scripts\formprocessor that have a v<version> in the name such as v3.
- Copy all the files in \\ed-webdev\w$\scripts\formprocessor to \\ed-w3\w$\scripts\formprocessor Don't delete the existing files on w3 or you will need to reregister the COM objects
- on production server, make a backup copy of the folder \\ed-w3\w$\wwwroot\intranet\formapp\v<version>.
- Copy the folder in \\ed-webdev\w$\wwwroot\intranet\formapp\v<version>. to \\ed-w3\w$\scripts\formprocessor
- This should take care of everything.

Migrating code from version B on Dev (ed-webdev) to version A on Dev

- make backups of files in \\ed-w3\w$\scripts\formprocessor that have a "vA" in the name and a backup copy of \\ed-webdev\w$\wwwroot\intranet\formapp\vA
- delete \\ed-webdev\w$\wwwroot\intranet\formapp\vA
- rename \\ed-webdev\w$\wwwroot\intranet\formapp\vB to …vA
- delete files in \\ed-w3\w$\scripts\formprocessor that have a "vA" in the name
- rename files in of files in \\ed-w3\w$\scripts\formprocessor that have a "vB" in the name to "Va". Then open these files and remove all references to vB that refer to migrated classes.
- Register the wsc objects. This must be done while logged onto the server.

Documentation

General_Documentation.doc

Word document overviewing application.

Fields.xsl

XL spreadsheet defining fields in frm_* tables

Xslt_hierarchy.xls

An explanation of the xslt templates. Instructions on reading the document are at: xlt_heirarchy_notes

Code

first_pass_default.xslt

XSLT that has customizations appended to it and is applied first to "form_data" xml document.

Second_pass.xslt

XSLT that is applied after first pass. No customizations are applied in this.

COEFPFormParameters.wsc

COEFormProcessorGlobalFunctionsV3.wsc

COEFPTargetDataV3.wsc

COEValidationFunctionsV1.wsc

Default.asp ASP page for form processor

Appendix B: Glossary

Target Database.

Database that the COE Form Processing Application will add, update, delete data from.

Target Table(s).

Table(s) that the FP will add, update, delete data from.

Form Configuration. (Aka FRM_* tables)

Data used to configure the form. Stored in tables such as:
FRM_FORM, FRM_FORM_PERMISSIONS,FRM_FUNCTIONS, FRM_PAGES, FRM_QUESTION_X_FUNCTION, FRM_RELATIONS, FRM_RESPONSES, FRM_VALUELISTS, FRM_VALUES

Form

is a single instance of the COE Form Processing Application (i.e. courses, ed committees,..). All database data which a Form populates is assumed to have the following shape: Single table single record with none or more one-to-one relationship with other table(s) and none or more one-to-many relationship. The form_id in the URL designates the form.

Parent/Primary Table

The target database table which has a single record associated with a given form. When a one-to-one relationship exists, one of the tables which has a single record associated with a form must be selected to be the primary table. The record_id in the URL indicates which Primary/Parent Table record is associated with the form by having the value of unique_identifier field in the Primary/Parent Table.

Related Table

All tables used in form with the exception of the **Parent/Primary Table**.

Child Table.

A table that is not the primary/parent table and is in a one-to-many relationship with the primary table.

Record ID Field

All target db tables used in this application must have a single field which uniquely identifies a record.

Form-Record

All the target table-records associated with a given primary table record. For example, in the committees form, a form-record would consist of the individual committee's record in the ED_COMMITTEES and all the associated committee member records in ED_COMMITTEE_MEMBERS.

Table-Record

A row in a database.

Table-Record-Action and Form-Record-Action.

When a specified action (add, delete, update, view) is performed or proposed for either a Table-Record or Form-Record.

Appendix C: Steps to Creating a Form Processor Form

1. Have a good understanding of the form, who will use it, etc.
2. Make a rough HTML or word markup of form.
3. Create Microsoft SQL Server Table that data will be stored in
    - Always include AutoIncrement field
    - Always make sure to include form processor fields:
    - created_date datetime
    - created_by varchar30
    - modified_date datetime
    - modified_by varchar30
    - [timestamp] timestamp

4. Have DBA create appropriate permissions
5. Open up Form Manager Access Interface and create new form. Populate forms in more or less the following order without any customization first. Then after testing and debugging form, add the customizations in the same order. Lastly add triggers.
    - Add New Form
    - Create Relationships
    - Create Functions for Value Lists
    - Create Value Lists
    - Add Values for explicit Value Lists
    - Create Questions
    - Add Questions to the Form

Test the form at https://www-s.ed.uiuc.edu/intranet/formapp/VC/default.asp?form_id=Form_ID

Appendix D: form_data XML Document

Form_data.xml Overview

The form_data xml document contains all target, configuration, state, and response data in it. First pass and second pass XSLTs are applied to this to produce the final web pages. A sample document is available at form_data.xml. This is best viewed in Internet Explorer or an XML editor.

The document looks like this:

```
<form_data>
   <state>…</state>
   <response>… instructions, errors, summaries of actions taken, etc...</response>
   <configuration>… most all data from the frm_* tables </configuration>
   <target_data>…data from target databases</target_data>
</form_data>
```

Each root element is outlined below. You may see the form_data xml document by turning on the debug cookie and adding **`&form_data=true`** to the URL. You will need to view it in the source code as it will be hidden from the browser**.**

<state> element

Each element is one of the state values. With the exception of the username and people_id, these state element names and values are in the get part of the URL. i.e. default.asp?form_id=7.

- username. Authenticated usename from COE User class stored in session object
- people_id. Authenticated people_id from COE User class stored in session object
- form_id.
- action = intro | index | add | update | delete
- record_id = id of parent record being edited
- newform deprecated?
- fkvalue deprecated?
- fkfield deprecated?

- filter_field = field being filtered on
- filter_value = value bing filtered on
- param1…param10 = params
- page = current page
- index_id = index being viewed.

<response> element

<configuration> element

The configuration element contains the following elements:

<indexes> contains child element <index>

One index is present for each index the form may have.

attributes:

- **index_id**
- **default_index = true or false**
- **index_menu_link_name**
- **default to empty on load = true or false**

<page>

attributes:

- **display_anchors** = true | false
- **index_is_add_update_response** = true | false
- **asterisks_in_required_questions** = true | false
- **sensitive_data** = true | false
- **service_id** = assocatied service id from ED.ED_USER_SERVICES.service_id
- **service_name** = assocatied service name from ED.ED_USER_SERVICES.name

The following attributes merely indicate the presence of custom xslts. The content of these custom XSLTs are integrated into the first pass xslt and need not be stored in the form_data xml document.

- **index_item_xslt** = true | false
- **page_page_xslt** = true | false

- **html_head_xslt** = true | false
- **menu_xslt** = true | false
- **header_xslt** = true | false
- **footer_xslt** = true | false

child elements:

- **form_name** and **form_title** name and title of form
- **record_name** and **record_name_plural** name of a single form_record.
- **associated_site_name**
- **primary_table**
- **manager_name, manager_phone, manager_email**
- **URL_Stem**
- **documentation_url**
- **error_email_notify, error_displayhtml_contact**
- **logo_image**

<relations> contains a <relation> child element for each relation in the form.

Attributes

- **one_to_many** = true | false
- **relation_id**
- **db_name**
- **table_name**
- **entity_name** and **entity_name_plural**
- **entity_max_add**
- **parent** = true | false

a <section> element exists for each section of the form

Attributes

- **one_to_many** = true | false
- **section_order** = ordinal integer
- **section_id** = FRM_SECTION.section_id
- **section_title**

The following attributes merely indicate the presence of custom xslts.

- **section_xslt** = true | false
- **section_footer_xslt** = true | false
- **section_header_xslt** = true | false
- **section_xslt_2nd_pass** = true | false

<question> elements are child elements of a <section> element
More information about questions may be found in question section.

Attributes

- **one_to_many** = true | false
- **question value_list_allow_other** = true | false
- **value_list_allow_multiple** = true | false
- **value_list_standard_search** = true | false
- **multivalues_in_related_table** = true | false
- **question_id** = FRM_Question.question_ID
- **section_id**
- **index_column_order**
- **question_order**
- **question_name**
- **data_area**
- **data_type**
- **data_location**
- nullable
- **validation**

- input_default
- **input_type** = name of input type such as “text”. See input_type list
- **value_list_id**
- **value_list_limiting_question_id**
- input_size
- input_maxlength
- input_rows
- input_cols
- relation_id
- function_id
- question_xslt = true | false
- input_xslt = true | false
- list_item_xslt = true | false
- error_xslt = true | false
- validation_error_xslt = true | false

Child Elements

<index_data> element

The index_data element is only present on index pages. The <index_data> element has the following attributes:

attributes:

- **add** = true or false if adds are permitted
- **record_count =** total number of records in index
- **total_pages**
- **current_page**
- **previous_page**
- **next_page**

<s:Schema > and <rs:data> are the two child elements of <index_data>.

They are both from the Microsoft ADO schema used when an ADO recordset is converted to XML. One thing to watch out for when using the rs:data elements is that fields/value pairs are represented as attribute/value pairs. Fields that are NULL simply are not represented in the XML row.

<supp_data> element

The purpose of the supp_data element is simply to bring in extraneous data to display on a form. This data is created by executing an SQL statement of getting XML from an HTTP request. A function is created for supp_data and stored in t frm_functions table. It may be associated with an index or with the form itself. If a form has 3 indexes it may have up t supp_data functions. 1 for each index and one for all the detail pages. The function should be of medium sql or http an be of type supp_data. These functions may contain state tags.

SQL expressions should return XML using the ADO's FOR XML clause. The structure of the XML will depend on the and the parameters of the FOR XML clause, but it will all be contained within the <supp_data> element beneath the roo form_data document. For example the expression
SELECT committee_role_id,role_name FROM ED.dbo.ED_COMMITTEE_ROLES FOR XML AUTO
Returns the element. Below:

```
<form_data>
…
<supp_data>
      <data>
      <ED.dbo.ED_COMMITTEE_ROLES committee_role_id="1" role_name="Chair">
      </ED.dbo.ED_COMMITTEE_ROLES>
      <ED.dbo.ED_COMMITTEE_ROLES committee_role_id="2" role_name="Co-Chair">
      </ED.dbo.ED_COMMITTEE_ROLES>
      <ED.dbo.ED_COMMITTEE_ROLES committee_role_id="3" role_name="Member">
      </ED.dbo.ED_COMMITTEE_ROLES>
      </data>
</supp_data>
…
</form_data>
```

URLs results must be valid xml to avoid breaking the form. And there is a performance penalty for loading via http. A example URL might be http://www.ed.uiuc.edu/facstaff/detail.asp?netid=[username] and it would return the following d

```
<form_data>
…
<supp_data>
      <data>
      <html>…the entire html document ends up here…</html>
      </data>
</supp_data>
…
</form_data>
```

More complete information on the SQL needed to persist recordsets in XML is at:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ado270/htm/pg_ado_update_data_14.asp

All the state tags may be used in the expression, such as [username],[form_id],[record_id] etc. Be careful not to use record specific state tags such as [record_id] on expressions used on index pages…they have no meaning here and will break your SQL.

Appendix E: State Tags

State tags are used in some fields edited within the form_manager. They may have any of the state attributes that are in element. They are replaced by the current value of the state variable before the data is made available to the form proce

Tags look like [record_id],[username], [param1], etc

For example the record_index_where field in the INDEXES form could be populated with:

DEVELOPMENT_2.dbo.INV_ITEMS .STATUS = 'I' AND CATEGORY LIKE '%[param2]%' AND DELETE_REQUESTED IS NULL

and before the form processor executed this index query, it would put the value of param2 in place of [param2]

Appendix F: Debugging Forms and Common Mistakes

Viewing Intermediary Documents

The following intermediary documents are written to the log files when the fpv3 log cookie is set to "true" The cookie can be set at the url https://www-s.ed.uiuc.edu/intranet/switches.asp . These files are in the form app debug folder at **`\\(ed-webdev|ed-w3)\l$\formapp\[username]\[form_id]`** Only one set of logs is available for a given form and user at a time as the files are written over each time. Additional processing instructions (PI) are added to these debugging files when they are output to files. These PI are only to make the documents work outside of the form processor.

File	Document
`log.html`	Stored procedure sql and general logging of errors
`parameters.html`	objParam data. This is the internal data structure for the form parameters. State data will also be shown here and in the form_data xml document.
`form_data.xml`	Form_data xml document. Contains all the top level elements (state, configuration,target_data, etc that are available in the other files below.
`firstpass.xslt`	First pass XSL document. This the document after custom xslts are applied. The stub document is at wwwroot\intranet\formapp\v3\first_pass_xslt.xslt
`firstpass.html`	Results of firstpass.xslt being applied to form_data.xml
`secondpass.xslt`	Second Pass XSLT document
`secondpass.html`	Results of second pass xslt being applied to firstpass.html This is the final document.
	CSS and Javascript. These documents are static and their locations can be found by looking at the references in the html head element.
`state.xml`	
`configuration.xml`	configuration element of **`form_data.xml`** document
`target_data.xml`	supp_data element of **`form_data.xml`** document

response.xml	response element of **form_data.xml** document
index_data.xml	index_data element of **form_data.xml** document
supp_data.xml	supp_data element of **form_data.xml** document

These files can be used in a number of ways including:

- checking if the correct data is available in the .xml files
- checking to see if custom xslts are correctly integrated into the first pass xslt
- testing implications of modifying configuration and custom xslts. By opening the **form_data.xml** log file in an xml aware application such as IE 6+, the first and second pass xslts will be applied to it. This will produce identical results to executing the form processor. To see the display effects of custom xslts, you can edit the first pass xslt and reload the **form_data.xml** document. However, you will not be able to see the resultant html source code…just its rendering. View source will show the xml document. To work with the final document…
- **secondpass.html** is the final document. You may want to use this to debug and develop javascript, css, etc. It is the same document you would get if you did a save from a web browser.

- Watch out that you don't develop code using these files and then have them overwritten by the form processor. Make a copy of the folder before starting to use them for development.
- Don't forget to turn the custom xslt switch back on or you will never be able to find out why your custom xslts aren't appearing.

Turning off Custom XSLTs

Customization can lead to all kinds of troubles from xslts that don't work to migration headaches to inconsistent user interfaces throughout the college. They can also lead to large amounts of time to produce a little functionality.

If you must use custom xslts, you can disable them for debugging purposes. To disable custom xslts in a document turn on the "disable custom xslt" cookie in the switches interface.. This will disable all custom XSLTs including the menu. Note that the custom xslts are still brought into the xslts, they are flagged as not existing in the form_data.xml document.

Appendix G: Adding Attributes to the Form Processor

Data in the FRM_* tables is used to configure the form processor. Adding an addional field in one of these tables should be done as follows:

1. Choose what table the attribute should be associated with (from, relation, section, question)
2. Choose a name, field type, and the set of values associated with it.
3. Have Kim add it to the appropriate stored procedure(s). *Probably FRM_GetParamData_rs or GetValueListData_rs*
4. Have Kim add it to the form manager
5. Change the appropriate db-to-xml transformation array in the appropriate section of COEFPFormParametersV3.wsc There are 4 types and you must choose one to determine how the attribute ends up in the form_data.xml document:
    a. **Boolean**. Will take form of <attribute_name>="true|false" based on bit field value in frm_* table. i.e. <section … soft_delete="true" … >…
    b. **Other**. Will take form of <attribute_name>="value frm_* database" i.e. <section … relation_id="56" …>

c. **BooleanAttributesFromXSLTExistance** <attribute_name> = “true|false” based on existence of data in field. This is simply to produce flags in form_data.xml document to help xslt decide if default or custom xslts should be applied. i.e. <section … section_xslt=”true” …> where FRM_SECTIONS.section_xslt contains an xslt.

d. **ChildElements**. Attribute will take the form of a child element. <<attribute_name>><attribute value from form_* table</<attribute_name> i.e. the help attribute in frm_questions.help <question><help>What is your pets name</help>…</question>

Appendix H: Form Processor HTML Structure

The form processor follows a particular html structure with particular id and div attributes of html elements. This must be maintained or avoided in customizations if CSS and Javascripts are expected to work properly on customized elements. If you are trying to avoid confilicting with ids and names your customization, use frmX_ as a prefix for (1) your attribute names, (2) values of ids, classes, and names.

i.e. `<p class=”frm3_specialformatting” id=”frm3_12” name=”frm3_special_paragraph” frm3_my_attribute=”blah blah”>`

```
<html>
<head id=”head”>
</head>
<body id=”form_NNN” class=”form_processor”>
  <div class=”header”>
  </div>
  <div class=”menu” id=”menu”>
  </div>
  <div class=”page” id=”main_content”>
      <h2><form_title>Inventory</form_title></h2>
      <div class="page_status">
            <div class="error_log" id="error_log" name="error_log" ><ul id="errors"> </ul><div>
            <div class="action_log" name="action_log" id="action_log">…<div>
      </div>
      <div class="directions"></div>
      <div class="pathbar"></div>
      <div class=”index”></div>
  </div>
  <div class=”footer” id=”footer” id=”footer”>…
  </div>
</body>
</html>
```

Optional elements are in green. elements only exist when data is present for them.

Footer HTML Structure

```
<div class="footer" id="footer" name="footer">
      <div class="credits" name="credits" id="credits">
            The form "COE Employee Data" is managed by Kim Nystrom
                  (<a href="mailto:knystrom@uiuc.edu">knystrom@uiuc.edu</a>)
                   265-0504
      </div>
</div>
```

Index Filters HTML Structure

Index filter forms are dependent on the question_html_structure attribute of the section. The filter questions within the filter form will have the same HTML structure as any questions within a section with the same question_html_structure value. The form which encompasses the questions has the following structure:

“table”

Use when the filter questions are intended to fit into a table structure.

“div”

“span”

When a horizontal effect is desired.

“col”

When a column effect is desired on the filter questions. Such as when there are many filter questions and the form is put in the left margin. This yields the following HTML structure:

```
<div class="filter" name="filter" id="filter">

   <form name="filter_53" action="default.asp?form_id=3&amp;action=index&amp;order_by_field=&amp;filter_fo
   method="post">

   <div class="section_header" name="section_header" id="section_header">
      <h2 class="section_title">Filter</h2>
   </div>
```

…one or more filter questions here that follow the structure outlined in questions…

```
   <div class="filter_submit" id="filter_submit" name="filter_submit">
      <input type="hidden" name="clear_filter" value="false">
      <input type="button" name="filter" value="filter" onclick="document.filter_53.submit()">
      <input type="button" name="reset" value="reset filter"
            onclick="document.filter_53.clear_filter.value = 'true'; document.filter_53.submit()">
   </div>
   </form>
</div>
```

Questions HTML Structure

The HTML structure of questions depends on the question_html_structure attribute of the section if the section is not a one to many section. If a section is a one-to-many section, the questions will depend on the value of the second_pass_xslt.

One to one section where question_html_structure =“td”

Use when the filter questions are intended to fit into a table structure.

```
<tr class="question_and_input" id="question_and_input" name="question_and_input">
      <td class="question" question_id="frm_x_question_147">
            <span class="required_symbol">* </span>
            <span class="question_name">University I-Card Number</span>
            <span class="question_question">What is the I-Card Number?</span>
```

```
      </td>
      <td class="question_input">
            <label for="input__259__536__147" friendlyname="University I-Card Number"></label>
            <input type="text" name="input__259__536__147" friendlyname="University I-Card Number"
      one_to_many="false" value="659352697" maxlength="9" size="15" alt="blank">
      </td>
</tr>
```

One to one section where question_html_structure =“div”

One to one section where question_html_structure =“span”

When a horizontal effect is desired.

One to one section where question_html_structure =“col”

When a column effect is desired. The structure of the HTML is as in the following examples:

```
   <div class="question_and_input" id="question_and_input" name="question_and_input">

      <div class="question" question_id="1410">
            <span class="question_name">Account Charged</span>
            <span class="question_question">What account was charged</span>
      </div>

      <div class="question_input"><label for="input__filter__filter__1410" friendlyname="Account
      Charged"></label><input type="text" name="input__filter__filter__1410" friendlyname="Account Charged
      one_to_many="false" value="" maxlength="25" size="15" alt=""></div>
   </div>
```

One to many section where second_pass_xslt = “v”

When a vertical one-to-many child record is desired.

One to many section where second_pass_xslt = “h”

When a horizontal one-to-many child record is desired.

Appendix J: Javascript

HEAD element and linked Javascript

Javascripts are brought in through the head element by the

- html_head_xslt_default template
- html_head_additions_xslt
- or by being added to the head_JS column of FRM_QUESTION_JAVASCRIPT .

If the javascript is used by more than one question, it is better to put it in the html_head_additions_xslt or html_head_xslt in the FRM_PAGES interfacse than in the question javascript data. This makes the code easier to read.

While it is easier edit the javascript in a separate file and link to it as an external file, the html_head_additions_xslt field has the added advantage to be processed as xsl. This allows the javascript code to be generated on the fly.

See also: html_head_xslt and html_head_additions_xslt

Body onload() content.

Whatever content is in javascript_onload in the FRM_PAGES interface will be added to the onload attribute of the HTML Body tag. For example if the FRM_PAGES.javascript_onload field is "calchours();" then the body tag will read: `<body id="195" onLoad="initialize();calchours();">`

Built in Javascripts

If you are using a custom head template, you need to bring them in to use them. You may view them in the fpfunctions.js file. The javascripts will not work if your html structure does not follow the form processor html structure. The following functions are in the fpfunctions.js file and are available for use.

- **`toggleMenu()`** turns menu on and off.
- **`removeMenu()`** removes menu, if it exists
- **`returnMenu()`** returns menu, if it has been removed
- **`pfOn()`** removes menu, header, and footer and turns element with id="offpf" to a style of display=block.
- **`pfOff()`** returns menu, header, and footer and turns element with id="offpf" to a style of display=none.
- **`removeHeader(), returnHeader(), removeFooter(), returnFooter()`** remove and return header and footers
- **`explain(msg)`** *Displays a message in kiosk window taking text as an argument*
- **`explain2(url)`** *Displays a url in a kiosk window taking a URL as an argument*
- **`confirmexit(newurl,msg)`** when called asks for confirmation of msg before exiting to new url.
- **`trim(string) removes leading and trailing spaces from passed string`**

Global Variables

`objConfiguration` is a large global object that represents the basic configuration data. Its constuctor function is generated by the first pass xslt being applied to the conf element. Empty values are set to javacsript null. 'true' and 'false' values are set to javascript True or False. To see what attributes are available, view source of html code.

- **`objConfiguration[form][< page or form level attribute>] = attribute value`**
- **`objConfiguration[relations][<relation_id>][<attribute or child element name>] = attribute or child element value`**
- **`objConfiguration[sections][<section_id>][<attribute or child element name>] = attribute or child element value`**

- **objConfiguration[quesitons][<question_id>][<attribute or child element name>] = attribute or child element value**

- **objConfiguration[value_lists][<question_id>][<value_id>][value|display|drilldownparentvalue] = value, display value, or drill down parent value**

- **objConfiguration[question_indexes][input_name][< question level attribute>] = attribute value. Input_name is the actual input name in the form (i.e. input_12_265_102) so there is one item for every question instance on the form. If a question is in a one to many relation in may have several input_names.**

Associating Custom Javascript with a Question

Additional javascript may be associated with a question input with the QUESTION_JAVASCIPT in the form manager. record in the FRM_QUESTION_JAVASCRIPT table represents a javascript event handler for that input.

- **question_id** is the question that the javascript should be applied to
- **event** may be "**onclick**", "**onfocus**", "**onchange**", or "**onblur**" Make sure the event is applicable to the input used. For example don't use an onclick event handler on a text area.
- **RHS**. The right hand side value is the javascript to be executed in the event. Don't forget the semicolon at the en is important for stringing mulitple calls into the same event handler.
- **head_JS**. This is the javascript that needs to be added to the head element of the HTML. It is only needed if a q RHS javascript needs javascript functions or global variables that are not included in the standard javascript libr processed in the xslt, so should either have valid xml characters (which is a pain to do) or have CDATA sections unprocessed data as in the example below.
- **execution_order**. This is the order the js is executed and should be unique for each event type and begin with should never be left empty.

The javascript in the RHS column is executed as an xslt of the form:

```
<xsl:template match="/form_data/configuration/javascript[@question_id=<
    current_question_id>]" mode="RHS_onchange">
  <xsl:param name="question_data" />
  <xsl:param name="relation_data" />
  <xsl:param name="section_data" />
  <xsl:param name="record_id" />
  <xsl:param name="input_name_stem" />
    [your RHS code goes here]
</xsl:template>
```

The data you put in the RHS section is inserted where the **[your RHS code goes here]** is.

The xsl params you have access to in the RHS xslt are:

- `$question_data`: the node of the question configuration data. i.e. /form_data/configuration/section/question[@question_id=<current_question_id>]
- `$relation_data`: the node of the relation configuration data. i.e. /form_data/relation[@relation_id=<current_relation_id>]
- `$section_data`: the node of the section configuration data. i.e. /form_data/configuration/section [@section_id=<current_section_id>]
- `$record_id` the record id of the current question. May be NEG1, NEG2, etc for adds

- `$input_name_stem`. This is the beginning stem of the input name such as input___<relation_id>__<record_id>__ This is very useful when the javascript in one question needs to access another question in the same record and relation, especially in one to many relationships. The input name of another question in a different relation may be more difficult to derive, but is quite doable by navigating the form_data xml document.

Example 1: Form #195 Question ID #1535

- **`question_id`** = 1535.
- **`event`** = onchange
- **`button_label`** = NULL
- **`RHS`** = calchours();
- **`execution_order`** = 1
- **`head_JS`** =

```
<script language="javascript">
<![CDATA[
function calchours() {
  var recordID = ]]><xsl:value-of
select="/form_data/target_data/relation[@relation_id='313']/record/@record_id" /><![CDATA[;
  var objQuestion;
  var intWeekValue;
  var totalHours = 0;
  for (var questionID in objConfiguration['questions']) {
    objQuestion = objConfiguration['questions'][questionID];
    if (objQuestion['section_id'] ==  '158' && objQuestion['question_id'] != '1537') {
      intWeekValue = parseInt(form1['input__313__'+ recordID + '__' + objQuestion['question_id'] ]
      if (intWeekValue) {totalHours = totalHours + parseInt(intWeekValue);};
    }
  }
  form1['input__313__' + recordID +'__1537'].value = totalHours;
}
]]>
</script>
```

This creates the following input with the above onnchange event hander. These event handlers will always follow the ev
handlers built into the form processor such as filterOnChange in the example below.

```
<input type="text" name="input__313__2__1535" friendlyname="Week 1" one_to_many="false" value="1"
maxlength="" size="4" onChange="filterOnChange(this);calchours();">
```

In the head section of the HTML document, the following is created:

```
<script language="javascript">
function calchours() {
  var recordID = 2;
  var objQuestion;
  var intWeekValue;
  var totalHours = 0;
  for (var questionID in objConfiguration['questions']) {
    objQuestion = objConfiguration['questions'][questionID];
    if (objQuestion['section_id'] ==  '158' && objQuestion['question_id'] != '1537') {
      intWeekValue = parseInt(form1['input__313__'+ recordID + '__' + objQuestion['question_id'] ].value);
      if (intWeekValue) {totalHours = totalHours + parseInt(intWeekValue);};
    }
  }
  form1['input__313__' + recordID +'__1537'].value = totalHours;
}
 </script>
```