BASIC FORM DEVELOPMENT CONCEPTS

The following is a brief summary of concepts to consider. Further information is available from the references noted at the end of this handout.

1. What information do you need?

Before starting to design any form it is important to decide on the information that you are trying to gather (and what information the user will likely want to enter). This allows you to design a form that minimizes the need for free text entry and maximizes efficiency of data entry.

2. How do you want to store data?

It is also important to decide whether the data that is being entered should be stored for later retrieval or will only be used temporarily for generation of the note. Data that is used only to generate a note can be stored in temporary variables. Data to be stored for later use (e.g. reporting) needs to be linked to observation terms. In this case a decision needs to be made regarding the observation terms to use to store data. It is important to make sure that one observation term is not used to store multiple different types of values. (e.g. You would not want to store current, previous, never in "Tobacco Use" in one form and 1 ppd, 2 ppd, etc… in this observation term in another form.)

3. How can you best gather that information (item types)?

A full description of item configuration is not included here, but the following are the types of items and their general uses to consider.

1. **Heading:** Usually a bold character string which describes some attributes of the items to follow it.
2. **Text:** Character string which displays either static text or instructions for the user.
3. **Data display:** Display a larger string of text, which can be multi-line and may also be used to display information dynamically.
4. **Edit field:** Allows the user to enter data about the patient and store it in a temporary variable or observation term. Data is entered by typing and the field must be on a single line.
5. **Multi-line edit field:** Allows the user to enter data about the patient and store it in a temporary variable or observation term. Data is entered by typing, but this allows the user to enter multiple lines of data and also allows scrolling.
6. **Dropdown:** Allows the user to enter data about the patient and store it in a temporary variable or observation term. Data can be entered either by typing or selection from a list of predefined options.
7. **Radio button:** Allows the user to select from several options for storage in a temporary variable or observation term.
8. **Check box:** Allows the user to enter data by checking a box for storage in a temporary variable or observation term. This is useful for Boolean type expressions
9. **List box:** Allows the user to select multiple items from a scrollable list for storage in a temporary variable or observation term.
10. **Action button:** Allows the user to press a button in order to trigger an action. This could be a predefined action or a user defined MEL expression.
11. **Flow sheet view:** A table which is included in the form and displays clinical data as defined for the selected flow sheet.
12. **Page:** Allows the user to view multiple "pages" of data from within the same form.
13. **Section:** Formatting region that allows placement of multiple items within a rectangular area. This also allows bounding of the area with a line and inclusion of a heading for the section.
14. **Visibility region:** Allows the form to display / hide items based on evaluation of a logical expression (written in MEL).

In addition to the above item types some item types will allow the design of the form to place restrictions on the data to be entered. This can be especially important for observation terms that will be used later for reporting. For example edit fields and dropdowns can be restricted to only expect

certain numeric data formats. List boxes, check boxes and radio buttons by design restrict user input to the options offered. Dropdowns can also be used to restrict data in this way, but it is not automatic.

4. How should the form layout be designed?
It is helpful to add all anticipated items (or at least all related items) to the form prior to work on final layout. It is important to group related items together. For example in a physical exam form you would want to group items by organ system, in a problem specific form you would probably want to group all elements of the history together. These groupings can be implemented with the section or page items. Because the standard chart note is formed by translating items in the order that they appear on the page it is important to keep this in mind when you are ordering/grouping items. If there are several small items they can be placed on a single line to save space.

5. How is the final note generated?
The note is based on information that is stored in the chart note translation for each item. The translation for all items is created and concatenated to form the final note. The chart note translation can be based on simple strings or formatted strings. The following two functions are used to format data for display.

fmt: The function fmt() is for formatting text in the translation. The function accepts 2 parameters. The first is the text to be displayed. The second is formatting parameters to use. Formatting options are B (bold), I (italic), U (underline) and an integer number to increase or decrease the size of the text. For example fmt("Vital Signs", "B,2") will place in the chart note translation the text:

Vital Signs

cfmt: The function cfmt() is for conditional formatting of text. It can accept up to 6 parameters. The first 2 parameters are similar to fmt(), but usually the text parameter does not always have a value. The other four parameters are optional. The third and fourth parameters are a text expression to appear before the first text expression and the formatting for this text (pre-text). The fifth and sixth parameters are the text and formatting for a string to appear after the first string (post-text). The pre-text and post-text are only displayed in the form if the first parameter has a value.

For example cfmt(obsnow("pulse rate"), "B,2", "Pulse: ", "2", "
", "") will not place anything in the chart note if there is no value entered in the current update for the observation term pulse rate "pulse rate". If the pulse has been entered and is 86 the following text would be displayed (with the next item on the next line of the note):

Pulse: **86**

As another example cfmt(obsnow("pulse rate"), "", "The patient’s pulse is ", "", "beats per minute. ", "") will not place anything in the chart note if there is no value entered in the current update for the observation term pulse rate "pulse rate". If the pulse has been entered and is 86 the following text would be displayed (with the next item on the same line of the note):

The patient’s pulse is 86 beats per minute.

ADVANCED FORM DEVELOPMENT CONCEPTS

1. Opposing list boxes allow configuration of two list boxes that contain the identical options, but are mutually exclusive. This can be used to document positive/negative symptoms or findings (e.g. ROS form). As a result if the user checks a box in one side the same option is unchecked in the "opposing" list box.

2. Adding power to forms using buttons

Action buttons can allow a means for quickly performing a set of predefined functions. These can roughly be broken into 4 groups.

> **Data Entry Buttons**: CALCULATION and SETVALUES buttons allow for setting the value of an item or group of items. CALCULATION buttons are often used to perform a set of actions and place the resulting value in a specified item (either observation term or document variable). SETVALUES buttons are often used to set the values of one or more items to a set "normal" value. However, they can also perform more complex calculations in order to determine a value.
>
> **Navigational buttons**: Buttons can be used to add additional form components, move between form components or move to another page within the current form component.
>
> **Clinical operation buttons**: There are a number of buttons that can open access to other clinical lists such as the medications, problems or orders entry.
>
> **RUNPROCESS**: The RUNPROCESS action button can be used to trigger any number of defined functions or execute MEL operations. They can even be used to launch other programs outside of Logician such as web browsers. There are two functions that can easily be configured using RUNPROCESS buttons that can help make forms more powerful. These are adding specific medications or problems to the clinical lists without requiring further user input. This can be done by setting the connection type for the button to RUNPROCESS and then pressing the MEL Medication or MEL Problem buttons. This will step you through the process of adding the desired problem or medication to the clinical lists without having to worry about configuring the specific MEL expression.

3. Advanced translation concept

Suppression blocks can be used to create narrative translation for chart notes. A suppression block is a group of items for which no translation will be created if none of the items in the group has a translation. This can be used for displaying a label for a section of items only if some data is entered in the section. In addition, it can be used for items such as a line feed at the end of the translation of a group of items.

Container/server configuration allows forms to work together to produce a more standard note when multiple forms are used. By configuring one form (e.g. Vital signs) as a container it will display the translation for multiple other notes within a SOAP note format. In order to do this each form will need to be configured as a server and then each page within the note will need to be configured as part of the subjective, objective, assessment or plan section of the note. See the Visual Form Editor Reference Manual for more information.

4. Functions/MEL

Functions. Functions are defined in the format:

```
fn fnFunctionName(parameterA, parameterB, …)
{
Instructions
.
.
}
```

Where parameters are the parameters to pass and instructions are lines of commands to be executed based on these parameters. These commands must be written in Medicalogic Expression Language (MEL). For a complete reference of MEL functions and syntax see the document Using Data Symbols. MEL can contain command mathematic functions (+, -, /, *). It can also contain string manipulation routines (strA = strB + strC will create string A as a concatenation of string B and C). In addition there are a number of comparison operators and logical expressions that can be used. For example, if … then … else … endif.

Some things to keep in mind when writing MEL.

1. Always make sure that all appropriate keywords or syntax markers are included. When using an "if" statement both the "then" and the "endif" are required or an error will be generated.
2. Always make sure that parentheses ( )and braces { } are matched.
3. If a value is being returned (or when creating translations) always make sure to include an "else" statement (otherwise "FALSE" will be returned).
4. Be careful about calling one function from another function if you have generically named variables (e.g. strTemp). If the function being called uses a variable with the same name it will be changed in both functions.
5. "Standardized" formatting of functions at the onset may take slightly more time initially, but will save a tremendous amount of time later when reading the function again.

MEL Watcher Expressions

These are MEL statements that are triggered if any variable/observation term referenced by them changes. These are used to automatically re-evaluate a value when there is a change in another value. (e.g. When a patient's weight is changed it would be appropriate to re-evaluate Body Mass Index). MEL expressions do not have any header information, but consist of an expression enclosed in braces.

```
{ OBSNOW("BMI", fnCalcBMI(OBSNOW("WEIGHT"), OBSANY("HEIGHT")) }
```

Function Libraries

Funcion libraries can be used in Logician 5.5 service pack 6 or later to share functions between multiple forms. They can be created from the function view window of Visual Form Editor and consist of a group of "related" functions. Choose Function from the Insert pulldown and then select User Defined Library. This creates text components that will be automatically loaded when the form is opened in Logician.

For further information:

Visual Form Editor© Reference Manual (included with help system of Visual Form Editor)
Visual Form Editor© Tutorial (included with help system of Visual Form Editor)
Using Data Symbols and Creating Encounter Forms (available at support.medicalogic.com/logician/55/documentation.html)