Chapter 17: The CRL Desktop Configuration File

In this chapter we illustrate the structure of the **CRL** desktop configuration file (in XML) as dictated by its embedded DTD file. We discuss each element type and its associated attributes as a unit.

We recommend that you read and understand the material in Chapter 16: *Introduction to XML and DTD Files* before delving into the material presented here. See Chapter 11: *Configuring and Launching CRL* for a task-oriented approach to configuring the application; it refers you to task-appropriate sections in the current chapter.

17.1 CRL Configuration DTD File Listing

The DTD file, embedded in the XML file as of **CRL** V1_7_06, defines the *element types* and the *attribute lists* that can be used in the corresponding **XML** file as elements and attributes, respectively, in markup declarations. Before the first element type declaration in the DTD file, there is an **XML** declaration which identifies the file to the parser as type **XML** and specifies the character encoding used. Immediately after this statement, the element type declarations and attribute lists begin.

Any changes to the DTD portion of the file could cause CRL to malfunction or become inoperable. This portion should not be edited!

```
[
<!ELEMENT LogBookConfig (Keyword*,InitialTabs)>
<!ELEMENT InitialTabs (EntryInputPage*,Page*,EntryInputPage*)>
<!ELEMENT Page (Keyword?,Menu+)>
<!ATTLIST Page
  class CDATA #REQUIRED
  title CDATA #REQUIRED
  tooltip CDATA #REQUIRED>
<!ELEMENT EntryInputPage (Keyword?,Menu+,ToolBar?)>
<!ATTLIST EntryInputPage
  class CDATA #REQUIRED
  title CDATA #REQUIRED
  tooltip CDATA #REQUIRED>
<!ELEMENT Menu (Keyword?,Topic*,SubMenu*,Topic*,SubMenu*)>
<!ATTLIST Menu name CDATA #REQUIRED>
<!ELEMENT SubMenu (Keyword?,Topic*,SubMenu*,Topic*,SubMenu*)>
<!ELEMENT Topic (Keyword?) >
<!ATTLIST Topic
  class CDATA #REQUIRED
```

```
  name CDATA #REQUIRED
  command CDATA #REQUIRED
  inquiryfile CDATA #IMPLIED
  double-click CDATA #IMPLIED
  double-click-parm1 CDATA #IMPLIED
  double-click-parm2 CDATA #IMPLIED
  size CDATA #IMPLIED
  offset CDATA #IMPLIED
  parm1 CDATA #IMPLIED
  parm2 CDATA #IMPLIED
  auto-entry CDATA #IMPLIED
  auto-entry-parm1 CDATA #IMPLIED
  auto-entry-parm2 CDATA #IMPLIED>
<!ATTLIST SubMenu
  name CDATA #REQUIRED>
<!ELEMENT ToolBar (ToolButton*)>
<!ELEMENT ToolButton EMPTY>
<!ATTLIST ToolButton
  class CDATA #REQUIRED
  tip CDATA #REQUIRED
  text CDATA #IMPLIED
  image CDATA #IMPLIED
  parm1 CDATA #IMPLIED
  parm2 CDATA #IMPLIED
  keyword CDATA #IMPLIED
  maillist CDATA #IMPLIED
  mailsubject CDATA #IMPLIED
  mailfrom CDATA #IMPLIED
  command1 CDATA #REQUIRED>
<!ELEMENT Keyword (#PCDATA)>
<!ATTLIST Keyword
  type CDATA #REQUIRED
  class CDATA #IMPLIED>

  ]>
```

17.2 Element Type and Attribute Listing

We don't list the entire default **XML** configuration file for the **CRL** application in the manual; it is quite long. We believe it's more instructive to describe each of the elements and attributes listed in it.

Note that for elements containing no nested elements, the following two constructions are equivalent:

```
<ElementName AttributeList> </ElementName>
```

and

```
<ElementName AttributeList />
```

XML is a case-sensitive language, however the code that uses this **CRL** configuration file ignores case. Therefore, you can too!

17.2.1 XML Declaration

An **XML** declaration always appears in the form `<?xml ... ?>`. In the **XML** file (as opposed to the DTD), the **XML** declaration takes the form:

```
<?xml version="1.0"?>
```

This is followed by a line that includes the DTD file on which the **XML** is based:

```
<!DOCTYPE LogBookConfig [
<!ELEMENT LogBookConfig (Keyword*,InitialTabs)
...(the rest of DTD file text)
]>
```

17.2.2 LogBookConfig

The `LogBookConfig` element type is defined in the DTD file as follows:

```
<!ELEMENT LogBookConfig (Keyword*,InitialTabs)>
```

This statement declares the element type `LogBookConfig`. `LogBookConfig` represents the overall **CRL** application, and is called the *root* element type.

In the **XML** configuration file, all the elements are included within it (i.e., the entire configuration file content is enclosed within `<LogBookConfig>` and `</LogBookConfig>`, with the exceptions of the `<?xml...?>` and `<!DOCTYPE...>` declarations at the top of the configuration file).

`LogBookConfig` allows content of the element types `Keyword` (optional and repeatable) and `InitialTabs` (required and non-repeatable). Therefore, all the other element types that may appear in the configuration file (listed in the DTD file) must be defined within one of these two element types, or within element types listed in their content, or further down the chain.

There is no attribute list for `LogBookConfig` and thus it has no attributes.

So far then, the **XML** configuration file is constrained by the DTD file to have a structure like the following (keeping in mind the keyword elements are optional, and must be of type “overall”; see section 17.2.3 *Keyword*):

```
<?xml...?>
<!DOCTYPE...>

<LogBookConfig>
   <Keyword type="overall">
     *KEYWORD1, ... ,KEYWORDN
   </Keyword>
   ... (more Keyword elements)

   <InitialTabs (no attributes)>

      ... (bulk of configuration) ...

   </InitialTabs>

</LogBookConfig>
```

17.2.3 Keyword

The `Keyword` element type is defined in the DTD file as follows:

```
<!ELEMENT Keyword (#PCDATA)>

   <!ATTLIST Keyword
   type CDATA #REQUIRED
   class CDATA #IMPLIED>
```

The `Keyword` element type is used to define the keywords in the application. **CRL** supports four types of keywords:

Type	Element Types which may contain Keyword Elements	Description
overall	LogBookConfig	These keywords are associated with the **CRL** application as a whole. They are available to be attached to any logbook entry.
entry	LogBookConfig	These keywords are associated with particular logbook data entry types (e.g., text, image, and so on). The `class` attribute on the `Keyword` element is used to specify the entry type. For a given entry, **CRL** reads in the class and makes available the keywords corresponding to the entry type. See section 17.3 *The CRL Java Class Libraries* for information on determining the `class` attribute value.

Type	Element Types which may contain Keyword Elements	Description
page	EntryInputPage, Page	These keywords are associated with particular desktop data entry pages. A keyword of this type is available to be attached to any entries made on its corresponding desktop page.
category	Menu, SubMenu, Topic	Keywords can be associated with particular menus, submenus and topics. A keyword of this type is available to be attached to any entries made in containers whose menu hierarchy includes the menu, submenu or topic for which the keyword was defined.

All keyword elements contain a text string (contents is listed as `#PCDATA`) which is a comma-separated list of keywords. Keywords should be entered in UPPERCASE only; **CRL** stores them as uppercase. Each keyword in the list may be preceeded by either one asterisk (`*`) or two (`**`):

- One asterisk (`*`) means the keyword is automatically inserted into associated entries, but the keyword is removable.
- Two asterisks (`**`) mean the keyword is automatically inserted into associated entries, and it is not removable.
- If no asterisks precede the keyword, it is not inserted automatically into any associated entry.

All keyword elements must have a `type` attribute, but the `class` attribute is used only for the `entry` type, as described in the box above.

The keyword elements listed above under the `LogBookConfig` element could be expanded as follows (note the uppercase used for keywords):

```
<Keyword type="overall">
     KEYWORD1, **KEYWORD2, ... ,KEYWORDN
</Keyword>

<Keyword type="entry" class="logbook.logentry.LogEntryText">
     *KEYWORD1, ... ,KEYWORDN
</Keyword>

<Keyword type="entry" class="logbook.logentry.LogEntryImage">
     KEYWORD1, ... ,*KEYWORDN
</Keyword>

<Keyword  type="page">
     **KEYWORD1, ... ,KEYWORDN
</Keyword>

<Keyword  type="category">
     KEYWORD1, ... ,**KEYWORDN
</Keyword>
```

17.2.4 InitialTabs

The `InitialTabs` element type is defined in the DTD file as follows:

```
<!ELEMENT InitialTabs (EntryInputPage*,Page*,EntryInputPage*)>
```

It is used to define the page tabs in the application. It has no attributes. It allows content of the element types `EntryInputPage`, `Page`, and again `EntryInputPage` (all of them optional and repeatable, within the sequence specified). So the `InitialTabs` element shown in section 17.2.2 *LogBookConfig* can be expanded to:

```
<InitialTabs>

  <EntryInputPage attribute1="value" ...>  ... </EntryInputPage>
  ... (more EntryInputPage elements)

  <Page attribute1="value" ...>  ... </Page>
  ... (more Page elements)

  <EntryInputPage attribute1="value" ...>  ... </EntryInputPage>
  ... (more EntryInputPage elements)

</InitialTabs>
```

17.2.5 Page

The `Page` element type is defined in the DTD file as follows:

```
<!ELEMENT Page (Keyword?,Menu+)>

    <!ATTLIST Page
            class CDATA #REQUIRED
            title CDATA #REQUIRED
            tooltip CDATA #REQUIRED>
```

The element type `Page` is used to define each of the non-data-input desktop pages in the application, i.e., report pages.

The element type `Page` has an associated attribute list:

- `class` should be set to the value `logbook.Page_General`
- `title` specifies the page title displayed at the top of the page.
- `tooltip` specifies the text that appears on mouse rollover of the page tab.

EntryInputPage is similar to **Page** but more versatile. We provide an example in section 17.2.6 *EntryInputPage*.

17.2.6 EntryInputPage

The `EntryInputPage` element type is defined in the DTD file as follows:

```
<!ELEMENT EntryInputPage (Keyword?,Menu+,ToolBar?)>
```

```
<!ATTLIST EntryInputPage
        class CDATA #REQUIRED
        title CDATA #REQUIRED
        tooltip CDATA #REQUIRED>
```

The element type `EntryInputPage` is used to define each of the desktop pages on which logbook entries can be made. Notice that it differs from `Page` (immediately above) only by the addition of the (optional) `ToolBar` content type. (The toolbar is used for entry input, and is discussed below.) Any keyword specified for an entry input page must be of the type "page" (see section 17.2.3 *Keyword*).

The `Page` and `EntryInputPage` elements (shown above in the box in section 17.2.4 *InitialTabs*) can be expanded to:

```
<InitialTabs>

<EntryInputPage class="logbook.Page_EntryInput" title="title1" tooltip="tooltip1">
   <Keyword type="page"> KEYWORD1, ... ,KEYWORDN </Keyword>
   <Menu attribute1="value" ...>     ... </Menu>
   ... (more Menu elements)
   <ToolBar (no attributes)>  ... </ToolBar>
</EntryInputPage>

... (more EntryInputPage elements)

<Page class="logbook.Page_General" title="title2" tooltip="tooltip2">
   <Keyword type="page"> KEYWORD1, ... ,KEYWORDN </Keyword>
   <Menu attribute1="value" ...>     ...   </Menu>
   ... (more Menu elements)
</Page>

... (more Page elements)

<EntryInputPage class="logbook.Page_EntryInput" title="title3" tooltip="tooltip3">
   <Keyword type="page"> KEYWORD1, ... ,KEYWORDN </Keyword>
   <Menu attribute1="value" ...>     ... </Menu>
   ... (more Menu elements)
   <ToolBar (no attributes)>  ... </ToolBar>
</EntryInputPage>

... (more EntryInputPage elements)

</InitialTabs>
```

17.2.7 Menu and SubMenu

The `Menu` and `Submenu` element types are defined in the DTD file as follows:

```
<!ELEMENT Menu (Keyword?,Topic*,SubMenu*,Topic*,SubMenu*)>
      <!ATTLIST Menu
              name CDATA #REQUIRED>

<!ELEMENT SubMenu (Keyword?,Topic*,SubMenu*,Topic*,SubMenu*)>
      <!ATTLIST SubMenu
              name CDATA #REQUIRED>
```

`Menu` appears in the content lists of the two page element types, described in sections 17.2.5 *Page* and 17.2.6 *EntryInputPage*. It is used to define entry-input menus on entry-input pages, and report menus on non-entry-input pages[1]. Notice that it can include submenus and/or topics (topics are the lowest-level menu items), and keywords. Any keyword specified for a menu or submenu must be of the type "category" (see section 17.2.3 *Keyword*). `Menu` requires a `name` attribute, which takes a text string giving the menu name. `SubMenu` is defined similarly; `Topic` is a little different, see section 17.2.8 *Topic*. Submenu and topic names appear as menu options on their next highest level menu.

The `Menu` element can be expanded to:

```
<Menu name="value1">

   <Keyword type="category"> KEYWORD1, ... ,KEYWORDN </Keyword>
   <Topic attribute1="value1" ...> ... </Topic>

   ... (more Topic elements)

   <SubMenu name="value2">
     <Keyword type="category"> KEYWORD1, ... ,KEYWORDN </Keyword>
     <SubMenu name="value3">
       <SubMenu name="value4">
         <Topic type="category" attribute2="value5" ...>
         ...
         </Topic>
         ... (more Topic elements)
       </SubMenu>
     </SubMenu>
   </SubMenu>

   ... (more SubMenu and/or Topic elements)

</Menu>
```

17.2.8 Topic

The `Topic` element type is defined in the DTD file as follows:

1. Note that you can add a report menu or menu item to an entry-input page as well. The class value for each topic (menu item) determines whether an entry-input container or a report container comes up when the user clicks on the menu item. See section 17.2.8 *Topic*.

```
<!ELEMENT Topic (Keyword?) >
      <!ATTLIST Topic
        class CDATA #REQUIRED
        name CDATA #REQUIRED
        command CDATA #REQUIRED
        inquiryfile CDATA #IMPLIED
        double-click CDATA #IMPLIED
        double-click-parm1 CDATA #IMPLIED
        double-click-parm2 CDATA #IMPLIED
        size CDATA #IMPLIED
        offset CDATA #IMPLIED
        parm1 CDATA #IMPLIED
        parm2 CDATA #IMPLIED
```

`Topic` is used with menu items as shown in section 17.2.7 *Menu and SubMenu*. There is a one-to-one correspondence between topics and containers. Any keyword specified for a topic must be of the type “category” (see section 17.2.3 *Keyword*).

The element type `Topic` has the attribute list as shown above, some attributes are required, others are optional (#IMPLIED). Here we provide descriptions of the attributes:

`class` determines what type of container pops up when the topic is selected. The values recognized by **CRL** are listed in section 17.3.1 *Container (Topic) Classes*.

`name` specifies the topic name; it is displayed as the title of the corresponding container

`command` All classes use either `command="add_to_Menu"` to add the topic to the menu, or `command="add_to_Menu_Page"` to add it to the menu and to pop up a container of this topic automatically on the page at run time

`inquiryfile` Currently not implemented.

`double-click` specifies the data entry type to create when a user double-clicks in container window. The `double-click` attribute takes the same values as the the `class` attribute of the ToolButton corresponding to the desired entry type; see section 17.2.9 *ToolBar and ToolButton*.

`double-click-parm1`

specifies the `parm1` (as defined in section 17.2.9 *ToolBar and ToolButton*) that corresponds to the logentry type specified by `double-click`, above. (Replaces the `parm1` attribute for **Topic**.)

`double-click-parm2`

Not currently used. Replaces the `parm2` attribute for **Topic**.

`size` specifies the default size of the container window for the corresponding topic. Specify the horizontal and vertical dimensions as `size="x,y"` in absolute number of pixels (e.g., `size="400,500"`) or as percent of **CRL** desktop size (e.g., `size="85%,90%"`). `x` and `y` can be specified in different units, e.g., `size="90%,500"` is acceptable.

`offset` specifies the default offset of the container window for the corresponding topic, measured from the top left corner. Specify the horizontal and vertical offsets as `offset="x,y"` in absolute number of pixels (e.g., `offset="4,5"`) or as percent of **CRL** desktop size

(e.g., `offset="5%,10%"`). `x` and `y` can be specified in different units, e.g., `offset="5%,10"` is acceptable.

`parm1` This will be phased out; it is replaced by `double-click-parm1`.

`parm2` This will be phased out; it is replaced by `double-click-parm2`.

`auto-entry` Not currently implemented.

`auto-entry-parm1` Not currently implemented.

`auto-entry-parm2` Not currently implemented.

So the `Topic` element shown above can be expanded to (notice, one keyword element only is allowed; it's optional):

```
<Topic class="logbook.MyInternalTopicFrame" name="Comments" command="add_to_Menu"
 double-click="logbook.logentry.LogEntryText" size="400,500" offset="10,10">

    <Keyword type="category"> KEYWORD1, ... ,KEYWORDN </Keyword>

</Topic>
```

This container is defined such that it does not automatically pop up on the window, but if user double-clicks in it a text entry will be created.

17.2.9 ToolBar and ToolButton

The `ToolBar` and `ToolButton` element types are defined in the DTD file as follows:

```
<!ELEMENT ToolBar (ToolButton*)>
```

<!ELEMENT ToolButton EMPTY>

```
<!ELEMENT ToolButton EMPTY>
<!ATTLIST ToolButton
        class CDATA #REQUIRED
        tip CDATA #REQUIRED
        text CDATA #IMPLIED
        image CDATA #IMPLIED
        parm1 CDATA #IMPLIED
        keyword CDATA #IMPLIED
        maillist CDATA #IMPLIED
        mailsubject CDATA #IMPLIED
        mailfrom CDATA #IMPLIED
        command1 CDATA #REQUIRED>
```

The `ToolBar` element type appears as content in the declaration of all `EntryInputPage` pages. It indicates that a toolbar for data entry is to appear on these pages. Its content consists of zero, one, or more `ToolButton` elements, described below. It has no attribute list.

The `ToolButton` element type appears as content in the declaration of `ToolBar`, described above. Each tool button represents a type or source of logbook data entry, definable via its attribute list:

`class`	specifies the data entry type. The allowed values are listed and described in section 17.3.2 *Entry type Classes*.
`tip`	defines the text that appears upon mouse rollover of the toolbutton.
`text`	specifies the text of the button (not used if image file specified)
`image`	specifies the file containing the image of the button (specified relative to the directory above `LogBook_admin`, unless full path is given). If image file is not specified, the text (above) is used to specify the button. It is generally preferred to use an image because it looks nicer.
`parm1`	specifies a parameter that depends on the entry type (which is specified by the `class` parameter). For a script, program, or OS command `parm1` is the command string. For a form it's the corresponding XML form definition file name.

For an output file from an external application, it's the file extension filter(s) (in **BROWSE** window, `parm1` provides the value for **FILES OF TYPE**). To include multiple file filters, separate them using plus signs (+), e.g., `parm1="doc+ppt+xls"`.

For a preset image `parm1` is either a URL or a path.

`keyword` specifies one or more keywords that get associated by default with each logbook entry of the corresponding type (`class` attribute, above) when the entry is created. Specify in a format like the following: `keyword="**KEYWORD1,*KEYWORD2,KEYWORD3"`, with or without asterisks, as desired. Keywords and asterisks are described in section 17.2.3 *Keyword*.

`maillist` If this attribute is set, then every time a form entry is archived, a message containing the form in html format gets sent to one or more recipients. `maillist` is used with form type entries only, and specifies the recipient(s) in one of the following ways:

a. An email address (which may be an individual address or a mail list address) in the format `<name>@<domain>`, e.g., `xyz-users@fnal.gov`.

b. The filename of a file containing a mail list (see section 11.5.2 *Enable Automatic Electronic Mailing of Form Entries*). If there is no `@` in the value specified, **CRL** looks for a mail list file. The file must be located in the directory specified by the parameter `Logbook.file_location.mail_list_directory` set in the properties file.

`mailfrom` is used with form type entries to specify the sender of the emailed form. It is a text string used as the "from" line in all email messages generated by the form, unless a `<FROM>...</FROM>` line is specified in the maillist file used, the latter taking precedence. `Mailfrom` must be a valid email address (which may be an individual address or a mail list address) in the format `<name>@<domain>`, e.g., `xyz-users@fnal.gov`.

`mailsubject` is used with form type entries. It is a text string used as the subject line in all email messages generated by the form, unless a subject is included in the maillist file used, the latter taking precedence. The `mailsubject` string may contain the substrings `$T`

and/or `$D` which get replaced by the time and date, respectively, in the formats HH:MM (24 hr clock) and MM.DD.YYYY, e.g., `mailsubject= “muon update $D $T”` generates the subject: `muon update 01.27.2002 17:32`.

`command1` is not currently used.

```
<EntryInputPage class="class1" title="title1" tooltip="tooltip1">
  ...
  <ToolBar>
     <ToolButton class="logbook.logentry.LogEntryText" tip="tip1"
                 image="image1.gif" keyword="KEYWORD1"
     />

     <ToolButton class="logbook.logentry.LogEntryForm" tip="test form"
                 text="test form" parm1="TestObjectsForm.xml"
                 maillist="mylistfile.txt" mailsubject="Test Form from CRL"
                 mailfrom="owlshifters@fnal.gov"
     />

     <ToolButton ... />
     ...

  </ToolBar>
  ...
</EntryInputPage>
```

17.3 The CRL Java Class Libraries

17.3.1 Container (Topic) Classes

Standard Input Container	logbook.MyInternalTopicFrame
Autoscheduled Input Container	logbook.MyInternalScheduledTopicFrame
Thread Container	logbook.logthreads.MyInternalThreadFrame
Report Container	logbook.MyInternalSearchFrame
Checkpoint Container	logbook.MyInternalCheckpointFrame

17.3.2 Entry type Classes

Text	`logbook.logentry.LogEntryText` Used for (formattable) text insertion via keyboard. Allows text formatting. Allows insertion of images, date and time, and/or text from a file.
Plain Text	`logbook.logentry.LogEntryPlainText` Used for text insertion via keyboard. No formatting, no insertion of images; allows insertion of date and time, and/or plain text from a file.
Output file from external application	`logbook.logentry.LogEntryRoot` Used to insert output file from virtually any external application, and having any virtually any file extension. Allows insertion of (formattable) text, date and time, and/or text from a file. (This class was originally created for ROOT output files, hence its name. It has been generalized since to allow all file types).
Program output (text)	`logbook.logentry.LogEntryExec` Used to insert output from virtually any OS command output (DOS or UNIX). Allows insertion of (formattable) text, date and time, and/or text from a file.
Form	`logbook.logentry.LogEntryForm` Used to insert a form. A separate toolbutton of this class is required for each form. Allows insertion into form text fields of plain text, date and time, and/or text from a file.
IPen® free-hand drawing	`logbook.logentry.LogEntryIpen` Used to insert a free-hand sketch or writing done on an Ipen® tablet and saved as a gif file.
Preconfigured image	`logbook.logentry.LogEntryImage` Inserts a particular image file of type gif or jpg into the entry it creates. Allows insertion of (formattable) text, date and time, and/or text from a file.

It is possible to define additional classes for data input; request information from *crl-dev@fnal.gov.*

17.3.3 Desktop Page Classes

Entry Input Page	logbook.Page_EntryInput
Report Page (non-entry-input)	logbook.Page_General