Appendix 3: Using the Exsys CORVID Servlet Runtime

The Exsys CORVID Ver 2 provides a powerful new way to deliver CORVID expert systems to end users. The new Exsys CORVID Servlet Runtime is a Java servlet that can be installed on a server. The Servlet Runtime incorporates the proven CORVID Inference Engine allowing it to be run on the server, with all user interface screens to ask questions or present data generated dynamically in HTML. This opens many new design options. Since all processing is done on the server, with no applet to download, the amount of material that needs to be downloaded to the client is greatly reduced. System security is enhanced, since all system files reside only on the server.

The new CORVID Servlet Runtime is in addition to the earlier CORVID Runtime applet for Web delivery, and stand-alone Java applications. The same CORVID system can be run via the applet, or servlet runtime programs or run as a stand-alone application. This provides a wide range of flexibility for Web or non-Web delivery of systems, depending on what is required.

Running a system with the Exsys CORVID Servlet Runtime simply requires adding HTML templates that will be used to ask questions and display results - the logic of the system remains the same. The system can then be moved to a server and run. To simplify the process, a variety of sample templates are provided for different styles of user interface. These templates are HTML files and can be edited using any standard HTML editor.

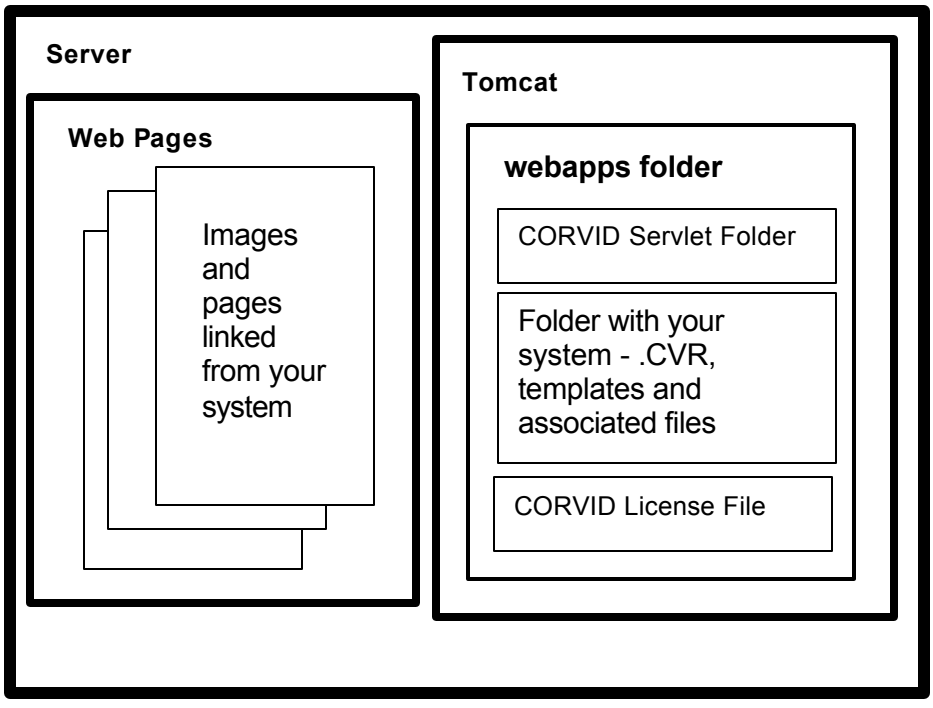
1. Installing and Starting the CORVID Servlet Runtime

The CORVID Servlet Runtime is a Java Servlet. It is provided as a .war file. Copy this file to the appropriate location on your server. In the case of Tomcat, this is the webapps folder under the Tomcat folder. If you are not sure where or how to install the file, check with your system administrator or servlet engine (e.g. Tomcat) documentation.

You will then need to have the server install the servlet. In Tomcat, this simply requires stopping and restarting Tomcat. Again, if you are unsure, check with your system administrator for how this is done on your server. This will create a folder named CORVID. In Tomcat this is in the webapps folder. (On some systems, it is necessary to delete the old CORVID folder before installing a new one.)

Your knowledge base files will need to be referenced relative to this CORVID folder. It is recommended that you not put your knowledge base files in the CORVID folder, since in at least some cases, this will cause problems when reinstalling a new .war file.

A convenient place for the knowledge bases is in a folder at the same level as the CORVID folder in the webapps folder. Create a folder named CORVID_apps and move the knowledge base .CVR file, the HTML screens and templates used by that system and any other files that the system may need (e.g. MetaBlock spreadsheets, data files). If your system policies prevent you from creating a folder at this level, it can be anywhere on the server that can be referenced from the CORVID folder.



Any files needed by the system that are referenced as URL links (e.g. images, linked pages) need to be put in a section of the server (or another server) that is able to serve Web pages. This is generally NOT the same section of the server that holds servlet files. It should be the portion of the server that would be used if a normal Web page was to be added to your site. (Check with your system administrator if unsure about the location.) The reason for this is that CORIVD will be building HTML pages using templates. These HTML pages will be sent to the end user's browser. It is that browser which will convert the links for images, etc. into the page displayed on the screen, and that browser needs to be able to obtain those images from a server. To do this, the images must be in a section of the server that can serve the image files to the browser.

It is not necessary to move the CVRU file to the server since any version of Java that supports servlets will be able to read the compressed .CVR file. It is also not necessary to move the .CVD file to the server. Unless the folder is in a location on the server that is secure and can not be accessed externally, the CVD file should not be placed on the server.

1.1 The Servlet License File

The CORVID Runtime Servlet is licensed for a specific server IP address . This is implemented with a file that holds license codes that activate the servlet on a particular server IP address. The license file can hold multiple license codes, but one must match the IP of the server that is calling the servlet. If a server is referenced by multiple IP addresses, such as an external IP and a different internal LAN IP address, each should have a license code in the file. Each license code is a numeric string and each should be on a different line in the file.

The default name for the license file is Corvid LicenseCodes.txt, and normally it should be in the same folder as the .war file is placed.

The license codes are obtained from EXSYS Inc. They are provided based on the license(s) purchased. If you have purchased the CORVID Servlet Runtime, you will be sent a code when the CORVID Servlet Runtime license agreement is returned to EXSYS Inc. All that should be needed is to place the code sent by EXSYS Inc in the CorvidLicenseCodes.txt file in the same folder as the .war file.

Temporary licenses can be provided in some cases allowing the CORVID Servlet Runtime to be tested and evaluated.

Technical Note: The CORVID Servlet Runtime checks in two folders for the license file. The Java code the Java code getServletContext().getRealPath is used to find the location of the servlet on the server. Different servlet engines return different relative addresses for this Java command. To compensate for this, CORIVD looks in the returned folder and the folder above it. One of these should be the location where the .war file is located, and a convenient place for the license file.

If your system can not find the license file, there is another option. The license file can be place anywhere on the server and given any name. It then must be called once when the servlet runtime is installed using the normal call to start the servlet runtime, but with **LICENSE=fileID** added:

```
www.myServer:8080?LICENSE=fileID&KBNAME=….
```

The **fileID** is the full URL to the license file and should by URL encoded if needed. The **fileID** can start with file: or http: depending on where the license file is stored. If this approach is used, it only has to be called **ONCE** after the servlet is installed for each IP address that is used. The license info will carry over for that IP until the servlet runtime is reinstalled.

1.2 Starting the Servlet

The call to start a CORVID system is done with a simple URL. The syntax is:

```
servlet_engine_path/CORVID/corvidsr?KBNAME=cvr_file
```

The servlet_engine_path depends on how your servlet engine references the servlets it has installed. For Tomcat "servlet_engine_path" is something like http://myServer:8080. The actual CORVID servlet will be called by "CORVID/corvidsr". On most servers this is case sensitive.

Using the folder arrangement recommended above, the link to start a system would be:

```
http://myServer:8080/CORVID/corvidsr?KBNAME=../MyApps/My
System.cvr
```

The "../" is a reference off the CORVID folder in Tomcat. In this case it moves down one level and into the MyApps folder which is where the application files are stored. If your folder is in a different location, use as many "../" as needed to move to the correct level and then up into the correct folder.

NOTE: The application files MUST be referenced off the location of the CORVID folder. The application files (.cvr) can NOT be referenced by a URL address or a full path. This assures that only CORVID applications on your server are run using your CORVID Runtime and server resources.

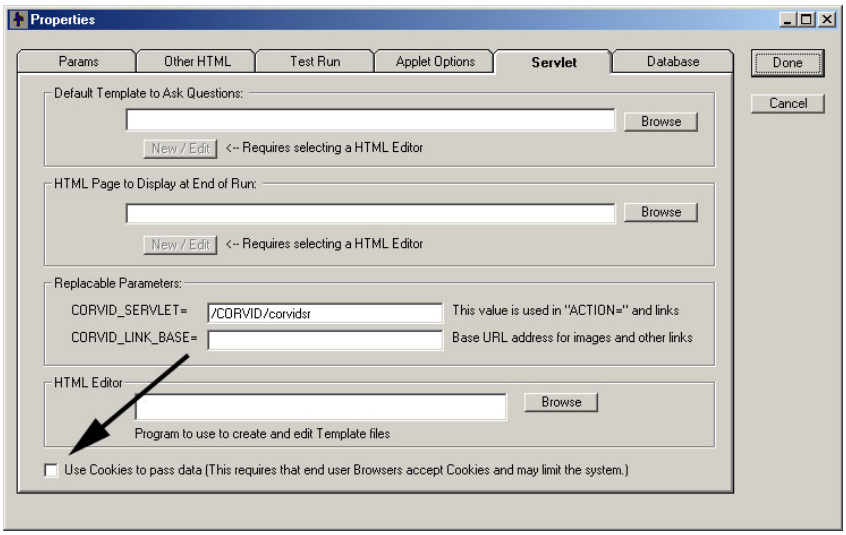
The link that is used to start the CORVID system can be placed anywhere a URL link is legal - off text, image, image maps, etc on other pages.

1.3 Cookies

Cookies are small packets of data that are passed invisibly between a servlet and the end user's browser. They are a very useful way to pass data, b ut can also be used to pass sensitive or confidential user data back to a server. Because cookies can be used in undesirable ways by some sites, most browsers have an option to prevent cookies. You can not be sure that all users will have their browser configured to accept cookies, unless your application will be distributed only in a know environment (e.g. intranet or standard company configuration). Most systems that will allow access to anyone on the Web should not use cookies to provide maximum compatibility.

The CORVID Servlet Runtime permits cookies, but does not require them. If a CORVID application is part of a larger system that requires cookies, cookies can be used with CORVID and they will remain active.

All Java servlets that are interactive need to send session information back to the servlet to let it know which user is returning data. In addition, CORVID has other session data that it needs to send back. The CORVID Servlet Runtime can do this via cookies, or by adding information to t he URL. The default is to NOT use cookies. This will result in some extra text in the URL, but will not effect the operation of the system. If you would rather use cookies to send this data, select the "Use Cookies" option on the Properties window Servlet tab for the system.



If this option is checked, cookies will be used to pass data. If a user has their browser set to not accept cookies, the system will not work on that browser. Unless there are specific reasons why you need to use cookies, it is strongly recommended that this option not be checked.

1.4 Passing Data at Startup

Initial data can be passed to the CORVID Servlet Runtime when it is started using the URL.

The normal link to start the CORVID Servlet Runtime is:

```
http://myServer:8080/CORVID/corvidsr?KBNAME=../MyApps/My
System.cvr
```

Data can be passed to CORVID by adding it to the end of the URL. The format is:

[VarName]=value

where “VarName” is the name of the CORVID variable to assign the value to and “value” is the value to assign to it. If the value includes any characters that need URL encoding, they must be encoded (This includes spaces and characters used in filenames). The data assignment is separated from the KBNAME string by "&". If there are multiple items of data to assi gn, they should be separated by "&", without any spaces.

For example, to pass the value of variables [X] and [Y] at startup:

```
http://myServer:8080/CORVID/corvidsr?KBNAME=../MyApps/sys.cvr&
[X]=1&[Y]=5
```

Static list variables can be assigned the number of a value or the short text of the value.

This ability to pass data at startup allows integration of the CORVID Servlet Runtime with other pages on a site. Those pages could be used to ask questions with a form, or interface to content management or persona lization software, which could provide the information to the expert system.

If a form external to CORVID is used to start the system, it should use the GET method to pass the data as part of the URL. When such a system is done, the results page should l ink back to the initial form to start another system, giving the user a chance to change their initial input data on the form.

1.5 Browser BACK button

When running the servlet version, the user interacts with the system via their browser, with the individual HTML pages generated dynamically from templates. The BACK button on the user's browser can be used to go back to an earlier page.

With the applet version, the CORVID applet is part of a particular page and the BACK button will take you to the prev ious page, which does not have the applet. When running with an applet, the effect of the browser BACK button is provided by the UNDO button within the applet. In the servlet, the browser BACK button will take you back to a previous question in the system. The input on that screen can be changed and then the system will continue from that point. All input that came after that screen will be "forgotten". Servlets also support an UNDO button on the page that can be used in place of the browser Back button.

In the CORVID Servlet Runtime, the BACK button can only be used while still in the system. If a user finishes a session, goes on to other web pages and then trys to go back to the system, the session may no longer be active. It depends on the server a nd how long it maintains the particular session as active. Once the server terminates a particular session, a user can not go back into it and will have to start the session over to rerun it.

1.6 Emulating the Servlet Interface from the Development Environment

Actually running with the CORVID Servlet Runtime requires that the system be run from a server that supports Java servlets (e.g. Apache Tomcat), with the CORVID Servlet Runtime installed, and the appropriate system files in the correct location. This is the way the final system will be delivered, but is not the way most development computers are configured.

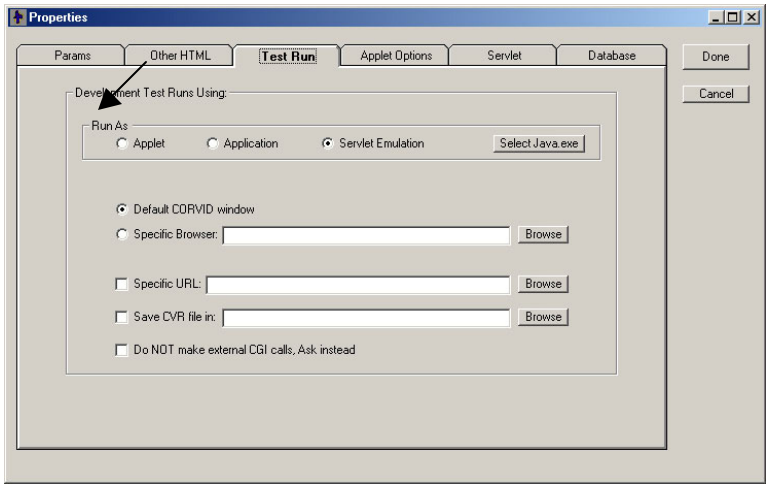
Normally when a CORVID system is run from the CORVID editor, it is via the applet Runtime. During development of the system logic, it should still be run as an applet, even if the intended end delivery is via the servlet. The first step in any expert system development project is to get the logic correct. For this stage, it is much better to not be concerned about details of user interface and concentrate on getting the logic complete and correct. If the system will be delivered only via servlet, just use the default applet screens to ask questions and display results.

Once the logic is correct the next issue when delivering the system via the CORVID Servlet Runtime is the templates that will define the user interface. This aspect of the system can be tested from the development environment. Version 2 of the CORVID editor can run the system in a mode that asks the questions using the same templates and HTML screens that will be used by the servlet. These screens are built using the same templates as will be used by the servlet. CORVID builds the screen and displays it using the CORVID browser. The CORVID Editor then waits until the user has provided some input, then closes the CORVID browser and continues processing. This results in the questions window opening and closing with each question - this will not be seen when run with the servlet on a server. This is only to let you test and refine the servlet question and result screens.

Any system that is intended for servlet delivery should be tested on the actual server. The emulation mode that CORVID uses, will work identically for most systems. However, systems that do special server-side processing, which are affected by the way a servlet saves state, or which use a different server operating system that may be case sensitive, could operate differently. Also, the emulation mode uses local values for replaceable parameters such as CORVID_SERVLET and those should be carefully tested on the server.

Note: The servlet emulation requires running as a Java application. You will need to have java.exe installed. If you have not already done so, click the "Select Java.exe" button and browse to where you have installed Java.exe. If you have not installed Java, you will first need to do that.

To run in the emulation mode, open the Properties window, click the "Test Run" tab and select the "Servlet Emulation" radio button.



Once the "Servlet Emulation" is selected, just click the usual blue RUN triangle icon to run. When you run in the servlet emulation mode, there will be a black "MSDOS" window displayed which will display trace messages if trace is turned on. There will also be a white Java application window where CORVID is actually running as an application. All user interaction will be via HTML screens built dynamically and displayed via the CORVID Browser window. The browser window will open and close as each question or report is displayed. (When running with the actual CORVID Servlet Runtime, the browser window will stay open and only the content will change.)

The screens displayed are the same HTML as would be displayed from the servlet. However, since the emulation mode can not process database or other server-side processing calls, these will have to be asked directly of the user.

*NOTE: **To cancel a session, you must close the MSDOS window**. This is usually a black "DOS" window. Just click on the "X" in the upper right of the window to close it. If you close the HTML window that is asking the question, it will just pop back and reask the same question. Closing the MSDOS window will cancel the run. After you close the MSDOS window, you may need to also close the window asking the question if it is still open. Once the MSDOS window is closed, the question window will not pop back up.*

2. Moving to the CORVID Servlet Runtime

In CORVID Ver 1.x, all screens were designed using the CORVID Screen Commands for questions, results and other informational screens displayed by the system. The CORVID Screen Commands are still supported. Any applet screens/commands that have been added to a system will remain in the system and be used when the system is run via the Applet Runtime . However, the Screen Commands apply only when running a system with the CORVID Applet Runtime.

When running a CORVID system via the Servlet Runtime, each screen that the system displays to the users is built using a "Template" screen that is designed using HTML and special CORVID commands and parameters. Modifying a system to run via the Servlet Runtime involves simply providing a template screen to be used for each screen the user could see. Since most templates are designed to be generic with replaceable parameters, a single template can apply a standard design to all the questions in a system. Typically a system can be delivered with the CORVID Servlet Runtime by just adding 2 templates - a template that can is used to ask questions and a template that is used to display results. Many more templates can be added if the user interface design requires it, but most systems can be run using only 2 templates.

The template screens are designed using a combination of HTML and special CORVID commands and parameters to build an HTML form. Sample templates are provided with various styles of user interface. These can be edited with any HTML editor. The templates can also include JavaScript, XML or any other commands that are supported by your browser.

The Exsys CORVID Servlet Runtime program needs to be installed on a server that supports Java servlets, such as Apache or IIS with Jakarta Tomcat and any other comparable servers that support servlets. Since the servlet is in Java, it does not matter if the server is UNIX, LINIX, MS Windows or OSX. All that is required is that the server support Java servlets.

2.1 Servlets

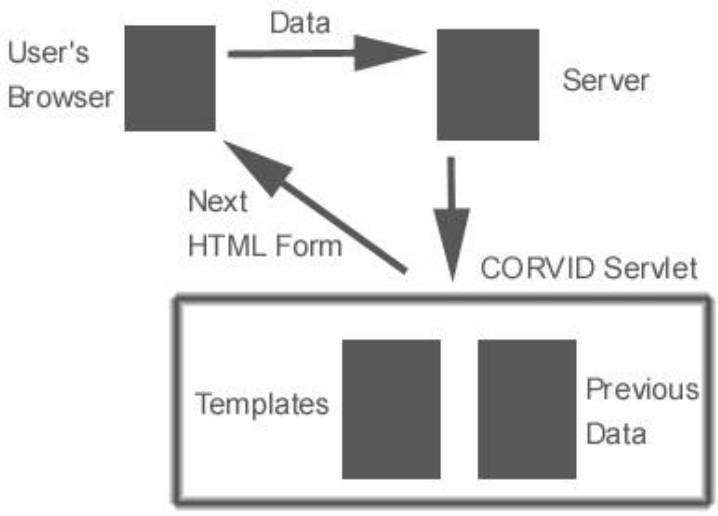
Servlets are Java programs that run on the server. In many ways they are similar to CGI programs, but remove many of the disadvantages presented by CGI. Servlets are run using a servlet engine such as Tomcat, though there are many other servlet engines. The servlet engine installs the servlet and makes it available to be called. The user calls the servlet using a special URL address and passes it information. The servlet processes the information and typically sends back an HTML screen to the user. In the case of the CORVID Servlet Runtime delivering an expert system, there may

be a succession of screens asking questions or displaying results. Each screen adds information and continues the session.

Since there may be many users simultaneously running sessions, the servlet is responsible for keeping track of each user and their data. Java servlets have built-in capabilities to simplify this task. However, each screen presented to the end user must contain certain information that allows the data to be sent back to the CORVID servlet with an identification of the individual user. This information is automatically added to the template.

2.2 Templates

The CORVID Servlet Runtime communicates with the end user via HTML forms. These forms are built using template files that define the page design and tell the system what HTML controls to use to ask questions or display results. The CORVID Servlet Runtime processes the data it has and, using the inference engine, determines what variable to ask next. To ask the user a question, the system must display an HTML page on the user's browser window. This HTML page is built by the CORV ID Servlet Runtime using a template file. The template has CORVID commands and replaceable parameters that are processed by the CORVID Servlet Runtime to produce the HTML screen sent to the user. When the user responds to the question presented, the data is sent back to the servlet. This adds to the information in the session, which then determines the next question to ask. This process continues until the results are displayed and the session is completed.



2.3 Building Systems for Servlet Delivery

Building a system in CORVID for delivery via servlet or applet is essentially the same except for the user interface. The logic of the system is designed exactly the same way using the Logic and Command Blocks. Regardless of delivery mode, the first step is always to get the logic of the system working correctly. This can be done with the simple default applet screens. Make sure the system asks the appropriate questions and arrives at the correct conclusion. If there are errors in the logic, correct them before spending time on the user interface - there is no point in delivering a system that is not giving the correct results.

Once the system is arriving at the correct results, it is time to add the user interface. This is where the steps for servlets are quite different from those for applets and stand-alone systems. Servlet delivery requires that template files be added that define how questions will be asked and how results will be displayed. Sample templates are provided and custom templates can easily be created. Applet delivery requires that CORVID Screen Commands be added to the system to define the look and controls used to ask questions within the applet. A system can be designed to run in both servlet and applet modes by adding both Custom Screen commands for applet delivery and templates for the Servlet. These are added in different ways and will not conflict with each other.

3. Templates

To make a system work with the CORVID Servlet Runtime requires adding the appropriate templates. Templates are a very powerful tool and have many options. The first step is to learn how to add a template to a system. It is easiest to start with the predefined sample templates. The next step is to understand the syntax for building your own templates. To get the most out of templates requires some knowledge of HTML and CORVID commands, however many systems can be delivered by simply editing one of the sample templates with a standard HTML editor.

Whenever the CORVID Servlet Runtime needs to ask the user for the value of a variable, it uses a template to build the screen. In most cases the template will be a generic template that can be used for many questions, giving them all the same look-and-feel and making maintenance easy. In special cases, such as a question asked with an image map unique to that question, there can be a special template made for that specific question.

A generic template is one that will work for most (or all) of the variables asked of the user. This template is defined to be the system default template and will be used to ask the end user for the value of any variable that does not have an associated template that overrides it. Any variables that require a different template can have a template associated them. A specific template associated with a variable will take precedence over the system default template.

In addition there is an overall CORVID default template that will be used for any system where no template is specified as the system default and no individual templates are associated with variables. This overall default provides a very simple and generic interface and is intended to be used only to test a system before other templates are designed, or if you are having trouble running a system and want to use the simplest template available.

Order of priority of template use:

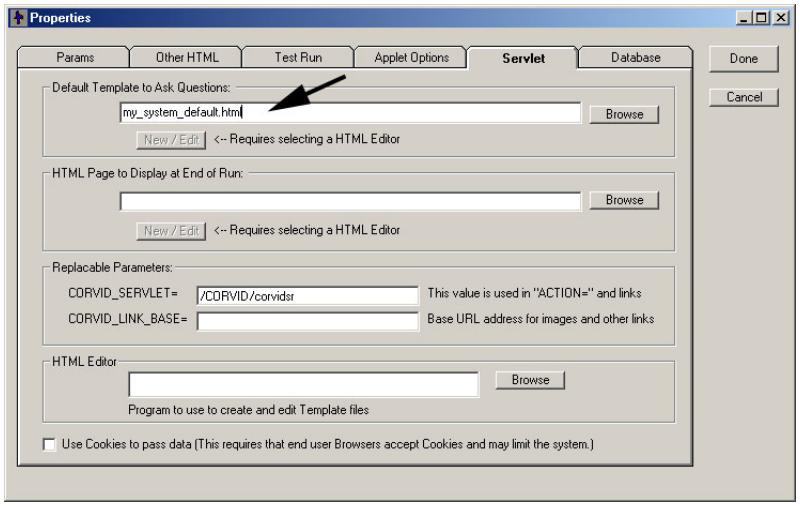
1. Template associated with the variable being asked
2. Knowledge base system default template
3. The overall CORVID default template screen

The overall CORIVD default template is installed by the WAR file that installs the CORVID servlet Runtime. It will be in the same directory as the CORVID servlet. This file should not be modified - it is the last option for asking questions. To give a system a different look-and-feel, create or select a template and make it the system default template.

A set of sample templates is provided with CORVID v2. These illustrate different template styles and approaches. A quick way to build an attractive template for a system is to select one of the sample templates that you like. Then open that template in an HTML editor and modify it for your system by adding text or images specific to your system. This modified template can then be used as the system default for your system.

3.1 Specifying the System Default Template to Ask Questions

The knowledge base system default template file is set from the Properties window. Click on the "Servlet" tab. This will display a window for setting the knowledge base defaults.



In this window you can select a system default template. If a template already exists, click on the Browse button in the "Default Template to Ask Questions" box and select the template to use.

If you wish to edit the template file or create a new one from within CORVID, you will need to use an external HTML editor. To do this, first you must select the HTML editor you prefer. This can be anything from a sophisticated HTML editor such as Adobe GoLive to something as simple as Notepad. (Microsoft FrontPage can also be used, but since it tends to add many Microsoft specific commands, it is not recommended unless you will be fielding the system only on a Microsoft server.)

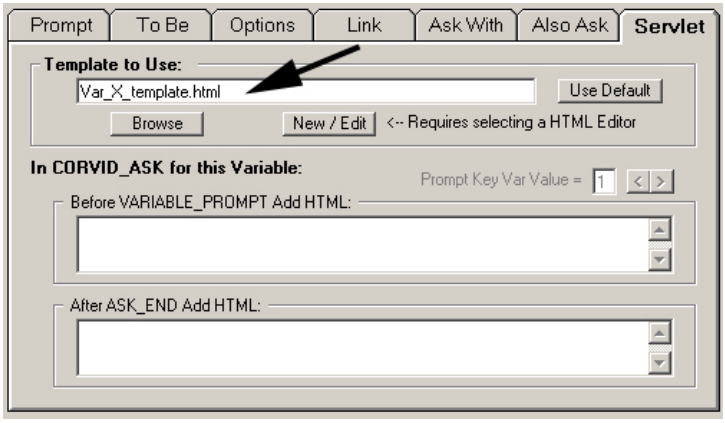
Click the Browse button in the HTML Editor box and locate the editor you wish to use. Once an HTML editor is selected, you will be able to click the New/Edit button for a template and it will open a new file with your selected HTML editor, or if a template file has already been selected, it will be opened for editing.

NOTE: The template files can have any name and extension - including .HTML. Your templates do not need to have a .HTML extension but some HTML editors will only work correctly if you give the template a .HTM or .HTML extension.

3.2 Specifying the Individual Variable Template to Ask Questions

In addition to the system default template, any variable can have an individual template specified to use when asking the user to input the value for that variable.

When a variable is selected in the Variable window, there is a new tab labeled "Servlet".



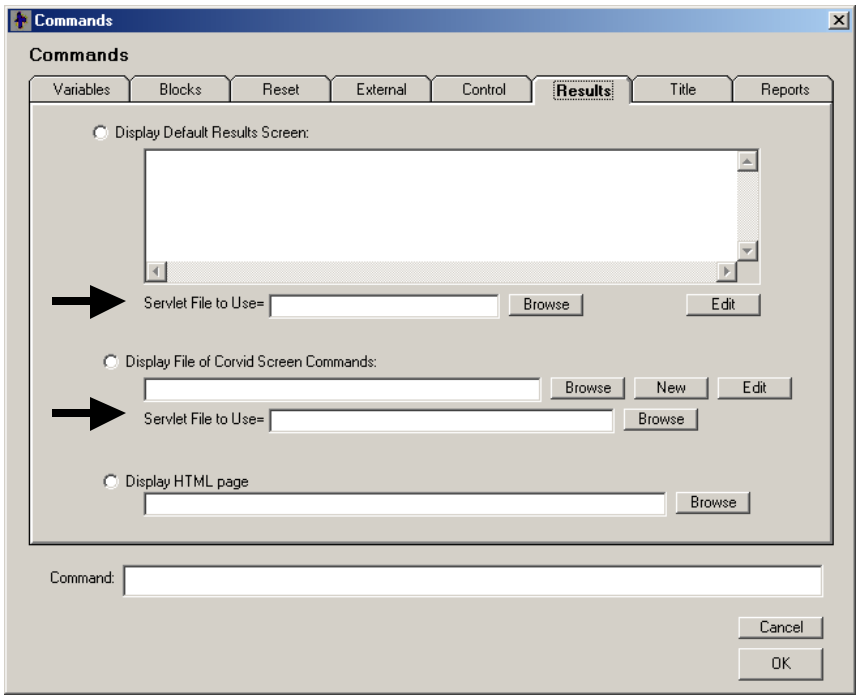
The Servlet tab allows assigning a specific template to be used when asking the user for the value of the variable. Enter the template for the variable in the "Template to Use" edit box. This can be selected by browsing to a file. If an HTML editor has been specified in the Properties page, you can use it to edit the template or create a new one by clicking the "New/Edit" button. If you had a template selected and then decide to just use the default, click the "Use Default" button , which will clear the edit window and change some internal settings. (Note: The Use Default button only needs to be used to clear an existing template. If the edit box is empty, the system default template will be used.)

Other controls on this tab allow you to specify text that is used with CORVID replaceable parameters in the template screens. This will be covered later in creating and designing templates.

3.3 Templates to Display Results

In addition to the templates used to ask questions, most syst ems will have at least one template that is used to display the system results. This is typically associated with the CORVID Command Block command "RESULTS" or "DISPLAY". The commands may have CORVID Screen Commands associated with them, but these apply only when running with the applet runtime. The Servlet runtime requires a template, which is specified by "SERVLET=template_file" following the RESULTS or DISPLAY command, where template_file is the name of the template to use.

This command can be built from the command builder for the RESULTS command:



When building a command, enter the name of the Servlet Template file to use for the RESULTS and DISPLAY commands.

When running with the CORVID Applet Runtime or as a standalone application, the "SERVLET=" part is optional and will be ignored. The system will use the CORVID Screen Commands associated with the command and run as it did in CORVID ver1.x.

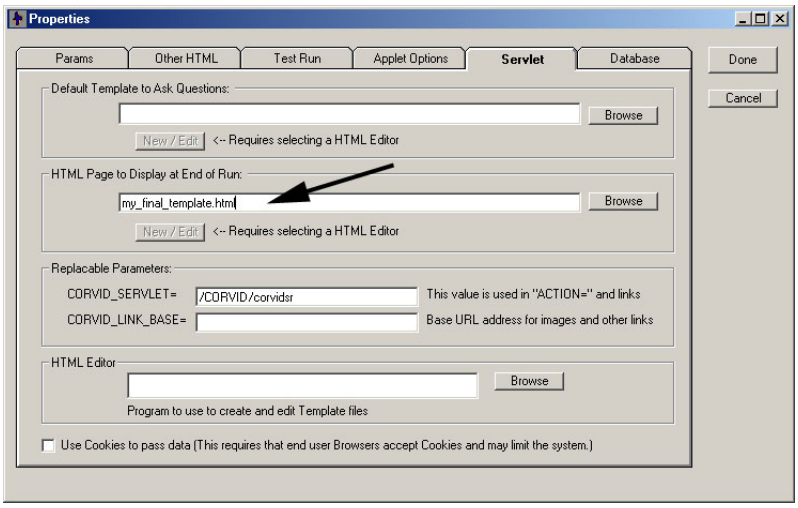
When running with the Servlet runtime, the CORVID Screen Commands will be ignored and the template will be used. If there is no "SERVLET=template_file" added to a RESULTS command, CORVID will

automatically use a very generic template that displays the values of all variables in the system. This can be used for testing a system before developing the RESULTS template for the system. The default template is installed with the CORVID Servlet Runtime and should NOT be modified since it is the template that the system will use when no other can be found.

3.4 Final Screen Template

In addition to the RESULTS template, a Final screen template can be added to a system. This is a screen that will be displayed when the system has completed the run. In most systems this will not be necessary since a RESULTS screen will be displayed at the end of the run. If it allows only a "Restart" or exit from the system, there will be no way to reach the Final screen. However, if you wish to have a screen after the results or if there are multiple ways to exit a system and the results might not be displayed, then the Final screen can be used.

To add a final screen to a system, open the Properties window and select the Servlet tab.



In the "HTML Page to Display at End of Run" box enter the name of the template to use. The Browse button allows selecting an exist ing file. If a HTML editor has been specified, the "New/Edit" button allows creating a new template, or editing the one that is selected.

If the system requires a Final screen, and none is specified, a very generic screen will be displayed. This is an ov erall CORVID default and should not be modified.

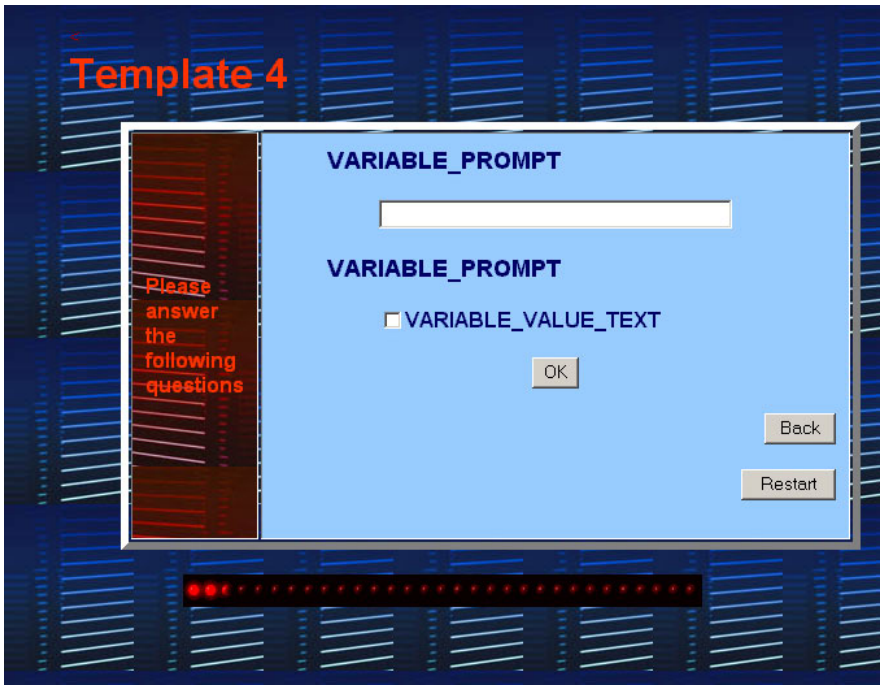
4. Sample Templates

As a first step, a system can be run with one of the sample templates provided with CORVID ver 2. These provide a variety of user interfaces. The intention of the samples is to provide a template that works well at providing a particular look-and-feel and which can be easily edited with a standard HTML editor. It is expected that you will want to modify the templates to include text or graphics for the system subject, your company, contact links etc. The template is intended to be a place to start.

Templates can include images just like any other HTML page, and like any other page, the image files must be located on a Web server. There are various ways to handle the issue of the location of image files, but for the sample templates the easiest approach is to use the copies of the templates and images that are on the Exsys Web site. This requires that you be on-line, but is a fast and easy way to try out sample templates that have all links in the correct location.

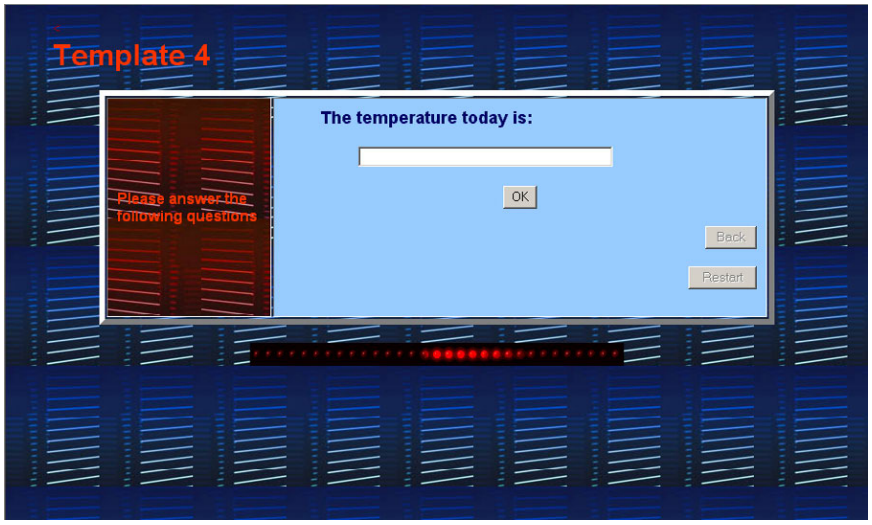
Currently there are 9 sample templates, but more will be added in the future. The easiest way to look at the templates is to go to http://www.exys.com/servlettemplates and click on the various sample styles displayed. The individual templates can be viewed as normal HTML pages. Find a template that you want to use. Copy the URL for that template and paste it in your system as the system default template (as described below).

Remember that the replaceable parameters are not converted to their associated variable text until runtime. For example, a template may look like:

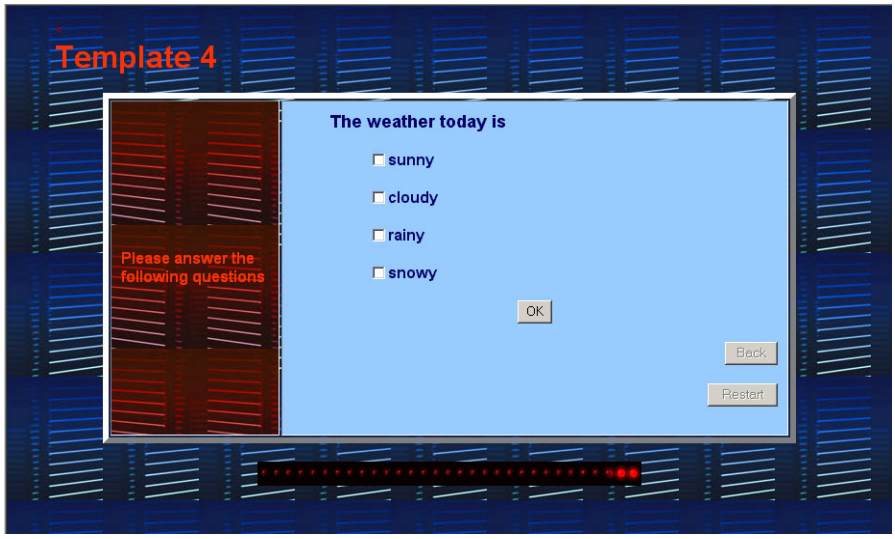


When this is run, the upper VARIABLE_PROMPT and the edit box below will be used to ask for numeric or string variables. VARIABLE_PROMPT will be replaced by the prompt text for the variable and the user will be able to enter their input in the edit box. The second VARIABLE_PROMPT and the VARIABLE_VALUE_TEXT below it will be used for static and Dynamic List variables. That VARIABLE_PROMPT will be replaced by the prompt for the variable and the VARIABLE_VALUE_TEXT will be replaced by a set of lines with check boxes, each with the text of one of the possible values.

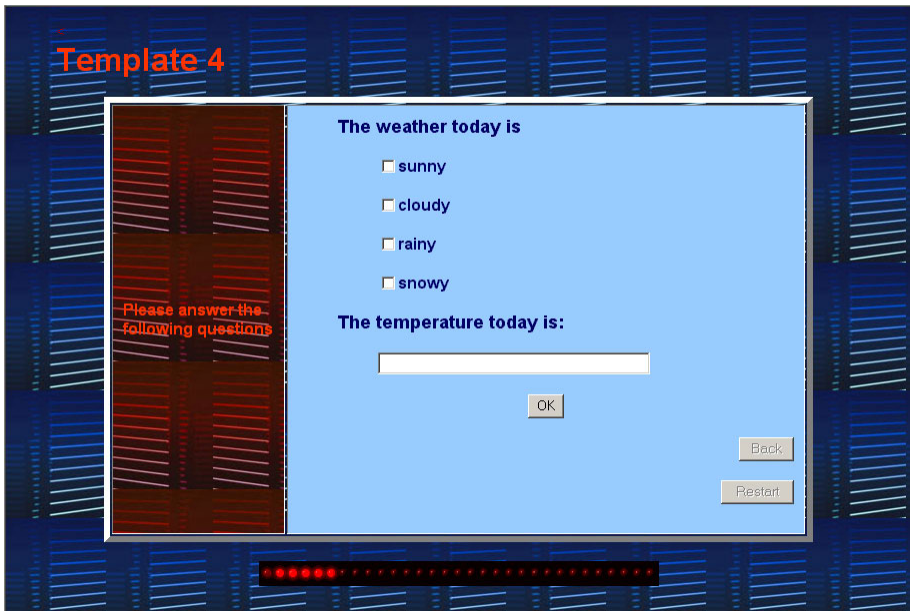
For example, if the above template was used to ask the numeric variable [TEMP] with the prompt "The temperature today is:", it would look like:



If a Static List variable [WEATHER] was asked with the same template, where the variable prompt was "The weather today is" with values "sunny", "cloudy", "rainy", and "snowy", it woul d look like:



When "Also Ask" is used to ask multiple questions on a single screen, CORVID automatically uses the appropriate section of the template to ask each individual question. Any number of questions can be asked on a single page, and they will be asked in the order specified by the Also Ask list. If the above template is used to ask both questions on the same screen, it would look like:



New sample templates will be added to the ones on the www.exsys.com Web site, so check occasionally for th e latest.

To modify a template for your individual system, download the template or save the HTML source for the page as a file. Open the page with an HTML editor and make any changes that are desired. In the above example, the "Template 4" text could be changed to the name of your application. Company information could be added to the page, etc. Then save the page and move it to your server. Move any image files that are needed, and change the BASE address for the images (or use CORVID_LINK_BASE and set it from within the system). Once the modified template is on your server, and can be viewed in a Browser using the new URL, just change the system question default template to the address of the new template on your server.

5. Templates to Ask Questions

Template files are a combination of HTML, special CORVID commands in HTML comments and CORVID replaceable parameters. A template can be a pure HTML form built with no CORVID commands or replaceable parameters, but that would be limited to a spec ific variable and server configuration. The CORVID commands and replaceable parameters allow a template to be generic. This enables the single template to work for many variables, and on any server. A system using generic templates often only requires only a single template for all questions, and it is easy to maintain.

The CORVID commands are added to the template as HTML comments - text between "<!--" and "-->". This makes it easy to add them with HTML editors. Most of the commands mark a section o f the HTML code that is only included in certain cases (e.g. only for certain variables or only if a Boolean test is true) or mark a block that is to be used repeatedly (e.g. repeated for each value in a Static List variable's value list).

The replaceable parameters are used to automatically assign text from the system. For example, a template to ask the user for the value of a variable can use the replaceable parameter "VARIABLE_PROMPT". This will automatically be replaced with the actual prompt text fo r the variable. The "VARIABLE_PROMPT" string can be formatted in the HTML page to set its style, color, size, etc. and that formatting will apply to the actual text of the prompt when it is replaced. If the prompt is changed in the system, the template screen will automatically use the new text.

There are many replaceable parameters that can be used in templates to handle the prompts and values of variables, information on the server and trace information.

5.1 The Servlet Template Form

The template files can have any name and extension. The sample templates use an .HTML file extension. Your templates are not required to have an .HTML extension but some HTML editors will only work correctly if you give the file an .HTM or .HTML extension. A .TPT ext ension can be used with some editors to make it easier to recognize a template.

The syntax of templates is designed to allow many types of questions to be asked using the same template. For most systems, it is recommended that a single template fi le, specified from the Parameters page, be used for all questions. Specific command options for a particular variable(s) can be

included in the single template using CORVID_ASK and CORVID_IF commands. This allows the look and feel of the system to be def ined in a single HTML page, assuring consistency and making it easier to create, maintain and update. Most of the HTML code in a template always applies to all variables, with only a portion that varies with a specific variable type. A separate template associated with a specific variable would only be used if that question needed to have a very different look from the default template, such as an image map or special JavaScript commands for particular effects.

Most templates include an HTML form. (Other types of templates that are not forms, but which use links to return data are discussed later.) The template is a normal HTML page and can use any HTML design or layout that is desired. The CORVID commands in the template are added as HTML comments - text between "<!--" and "-->".

The form section of the template is used to ask the user a question using radio buttons, check boxes, lists, edit boxes etc. The user's input is sent back to the CORVID Servlet Runtime which will process the data and c ontinue the run. The HTML outside of the form section usually does not use CORVID commands and can be any HTML design.

There are 5 main sections to the form:

1. First, the section needs the standard HTML code that indicates the section of the page is a FORM. The first line of the form should be:

```
<FORM METHOD="POST" ACTION="CORVID_SERVLET">
```

The text "CORVID_SERVLET" is a replaceable parameter that is set by the system. This is explained below.

2. The body of the form will usually have one or more sections marked with CORVID_ASK commands that indicate sections of HTML code that should be used to ask certain variables. The command "CORVID_ASK" allows a section of the form to be specified for use by particular types of variables (e.g. numeric), spe cific named variables (e.g. all variables that start with "XYZ", an individual variable, or all variables not covered by one of the other CORVID_ASK sections.

Within the CORVID_ASK section, there are usually replaceable parameters that are entered as no rmal HTML text, but which will be replaced with the Prompt and Value, etc. from the variable.

3. Optionally, there can also be sections of the form that are included with CORVID_IF commands. This allows a Boolean test expression to be used to determine if a section of the template should be included. For example, if a certain Confidence variable has been given a value, you may wish to notify the user immediately or ask a question in a different way.

4. A SUBMIT button(s) must be part of the form. Thi s submits the data or sends other data to indicate a BACK or RESTART button has been pressed. There must be at least one submit button on the form.

5. The closing </FORM> tag indicating the end of the form.

If you are familiar with HTML forms, the temp late can be built using a simple text editor such as Notepad, but it is generally easier to use an HTML editor.

5.1.1 <FORM…> Tag

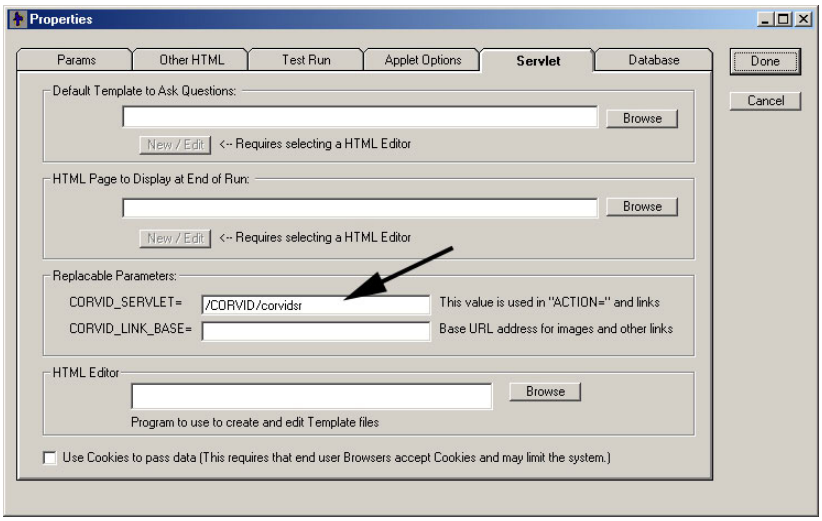
The <FORM…> tag indicates the start of the form. The <FORM> tag must include a METHOD="POST" and "ACTION=CORVID_SERVLET".

```
<FORM METHOD="POST" ACTION="CORVID_SERVLET">
```

All data should be sent back from CORVID forms using POST. This assures that any amount of user input can be sent, even if large blocks of text are entered in an edit box, or there are many questions on a scr een.

The ACTION="CORVID_SERVLET" tells the page where the CORVID servlet is located so that it can send the data back. CORVID_SERVLET is a replaceable parameter that is set for the system on Servlet tab of the Properties page. This will be the address where the servlet is installed on the server. It will end in "/CORVID/corvidsr", and will be something similar to:

```
"http://www.my_server:8080/CORVID/corvidsr"
```



When a system is moved to a server, the address of the CORVID Runtime Servlet for that machine is entered on the Properties page and will automatically be used to replace the value of the text "CORVID_SERVLET" when it is found in any template. Using CORVID_SERVLET, rather than hard coding the address in the template, allows the system to be moved to another server simply by changing one entry on the Properties page.

The <FORM> tag must be in the BODY section of the page. For example, a simple page might start:

```
<html>
<head>
<title>My CORVID System</title>
</head>
<body bgcolor="#ffffff">
<form method="post" action="CORVID_SERVLET">
```

There could also be any HTML code for site look and feel between the <BODY> tag and the <FORM> tag. There could be code to display text, images, setup tables, links, standard colors, styles or other content not specific to the running of the expert system. The form itself can also be an entry in a table to provide a particular arrangement or layout.

5.2 Referencing Image Files

The template file often will incorporate image files or other files from the server. Normally an HTML page is a physical page on a server, and images can be put in the same folder or a subfolder. This allows the images to be referenced using the location of the page as a base address. For example, if a page uses an image named "my_image.jpg" and that image is in the same folder (directory) as the page, it can just use "my_image.jpg" and the Browser will automatically look in the same folder as the page. However, a page displayed from the CORVID Runtime Servlet is created dynamically and does not really have a physical location on the server. Consequently, image files can not be specified by relative location to a page and a more detailed URL address is required. In addition, the template file does not need to be in a location on the server that allows Web browsing (though it can be), however, the image files MUST be in a section that is accessible via Web browsers.

The image files in a template can be referred to in several ways.

Full URL

If you know where the image is located on a server, just include the full URL to that image. This requires that the image be in a location that can be referred to by a URL - entering the URL in a Web browser must display the image.

For example:

```
<IMG SRC="http://www.mysite.com/images/myImage.jpg">
```

Use BASE to set a Base URL for Images

If all the images are in one folder (or subfolders of a folder) you can use the

```
<BASE href="…">
```

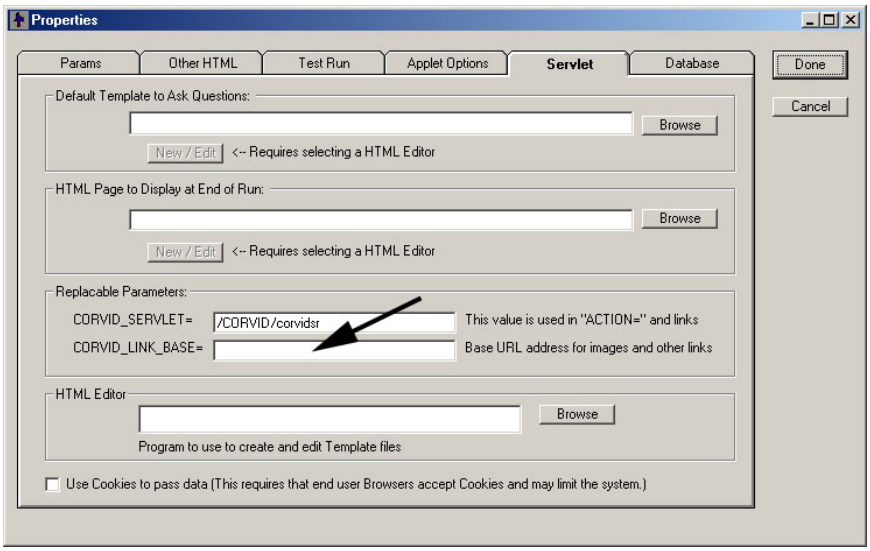
tag in the template. The <BASE> tag indicates the location to use for all images and links. Any image file or link that is not specified by a full URL starting with "http://" will use this base address as the starting location. The BASE tag must occur in the HEAD section of the template - that is between the <HEAD> and </HEAD> tags. The base URL can be hard coded such as :

```
<BASE href="http://www.mysite.com/images/">
```

Alternatively, the replaceable parameter CORVID_LINK_BASE can be used.

```
<BASE href="CORVID_LINK_BASE">
```

This makes it easier to move systems to other servers or change the location of the image files. The CORVID_LINK_BASE parameter is set from the Servlet tab in the Properties window.



For example, if all the images in the template are in "http://www.mysite.com/images/", just use <BASE href="CORVID_LINK_BASE"> in each template and in the Properties page enter "http://www.mysite.com/images/" for the value of CORVID_LINK_BASE. This value will be used to replace CORVID_LINK_BASE where it is used in the templates. If the images are then moved to a different location, just changing the CORVID _LINK_BASE value will change the value in all templates when they are used.

5.3 Section to Ask Questions

Templates can be designed to ask various types of questions - multiple choice questions for Static and Dynamic Lists, and edit fields for numeric or string variables. In addition, the formatting for some variables may be different for others and multiple variables may be asked on the same screen. The template syntax makes it easy to incorporate all of these into a single template.

Within the template's FORM section, there can be sections of code that are used for specific CORVID variable(s).

The section of code for a particular variable is marked with CORVID commands.

```
<!-- CORVID_ASK  VarID  -->  …  <!-- ASK_END -->
```

Note that these are HTML comments. The command must start with <! -- and end with -->. HTML editors allow adding comments to the HTML, and the CORVID commands can be added using that feature of the editors.

The CORVID_ASK command through the associated ASK_END command, MUST be in the FORM portion of the page - that is between the <FORM> and </FORM> tags.

The VarID indicates which variable(s) the CORVID_ASK code applies to. The allowed values for VarID are:

VARIABLE	All variables
STATIC_LIST	All Static List variables
DYNAMIC_LIST	All Dynamic List variables
CONTINUOUS	All numeric, string and date variables
NUMERIC	All numeric variables
STRING	All string variables
DATE	All date variable
[VARNAME]	The specific variable varname
[VARMASK]	All variables fitting the mask pattern

If a mask pattern is used, the standard CORVID mask characters are used:

Character	Matches
?	Matches any character
*	Matches the rest of the string
character	Matches itself
#	Matches any digit 0-9
{abc}	Matches any single characterin the {}
{X-Z}	Matches any single character between X and Z

A template file typically will have multiple CORVID_ASK sections, and MUST have a section that applies to each variable that will be asked using the template. For each variable that is asked using the template, **the first CORVID_ASK section which has a matching VarID will be used and all other CORVID_ASK sections will be ignored for that variable - even if they would also have matched** For example, if there were 3 sections in this order:

<!-- CORVID_ASK [COLOR] --> *Section 1* <!-- ASK_END -->

<!-- CORVID_ASK [C*] --> *Section 2* <!- ASK_END -->

<!-- CORVID_ASK STATIC LIST--> *Section 3* <!-- ASK_END -->

The variable [COLOR] would use section 1 code, and ignore sections 2 and 3. A variable starting with "C", but not [COLOR], would use section 2 and ignore section 1 and 3. All static list variables that did not start with "C" would use section 3 and ignore section 1 and 2.

One convenient way to make sure a template will work for all variables is to end with a <!-- CORVID_ASK VARIABLE --> section. This will apply to all variables that have not already matched a CORVID_ASK command in the page.

Care must be taken since Static List and Numeric variables typically need very different formats and controls for the questions. If a system has numeric, string, date, Static List and Dynamic List questions, it can be handled with 2 sections:

```
<!-- CORVID_ASK  CONTINUOUS -->  Section 1  <!-- ASK_END -->
<!-- CORVID_ASK  VARIABLE -->  Section 2    <!-- ASK_END -->
```

Section 1 will be used for all numeric, string and date questions. Section 2 will be used for all variables that are not Continuous, so it will be used for all Static and Dynamic List variables. Naturally, you might want to have m ore variation and have a separate section for each type.

5.3.1 Also Ask

The Servlet Runtime supports the "Also Ask" feature of CORVID, which allows multiple questions to be easily asked on one screen. As with the Applet Runtime, just go to the Also Ask tab on the variable's properties and select the other variables to ask at the same time. When the selected variable is asked, each variable in the Also Ask list will be asked on the same screen. In the servlet version, all questions will be asked with th e same template associated with the initial variable being asked. That template MUST have sections appropriate for each variable that will be asked in the Also Ask group. That means there must be a CORVID_ASK section that will match each variable in the Also Ask list.

The CORVID_ASK sections for each Also Ask variable in the template screen can be in any order. CORVID will check all the CORVID_ASK sections for each variable and use the first matching section. The initial variable will be asked with it' s appropriate CORVID_ASK section, followed in order by each of the Also Ask variables asked with their matching CORVID_ASK sections.

For example, using the CORVID_ASK section:

```
<!-- CORVID_ASK  [COLOR] -->  Section 1  <!-- ASK_END -->
<!-- CORVID_ASK  [C*] -->  Section 2    <!-- ASK_END -->
<!-- CORVID_ASK  STATIC LIST-->  Section 3    <!-- ASK_END -->
```

If a system asks the variable [COLOR] which has an Also Ask list of [INPUTS], [OUTPUTS] and [COST]. Where [INPUTS] and [OUTPUTS] are static list variables, the screen would have:

[COLOR] asked Section 1 used to ask [COLOR]

[INPUTS] asked with Section 3

[OUTPUTS] asked with Section 3

[COST] asked with Section 2

Also Ask uses the template for the initial variable to ask all variables in the Also Ask list - even if that is not the template associated with those individual Also Ask variables. This allows a variable to be asked in different ways. If the variable [COST] is asked as an Also Ask from [COLOR], it will use the section of the template associated with [COLOR] that matches [COST]. If no input is provided for [COST], and the value is needed by the system, it would be reasked using the template associated with [COST]. The second time [COST] is asked, you could use a different template that reminds the user that this input was not previously provided. (This ability to ask a question different ways can easily be done in the servlet version, but is not easy to do using applets.)

5.3.2 Controls

Within a CORVID_ASK section is the definition of how to ask the user for that specific type of variable. This can include the type of control to use, formats, colors, fonts, etc. Various HTML commands can be used for radio buttons, check boxes, lists, edit boxes, etc. This is done using the standard HTML tags such as <INPUT...>, <SELECT ...>, <TEXTAREA…>. These can be built and formatted using an HTML editor, but MUST be within the associated <!-- CORVID_ASK --> … <!-- ASK_END --> commands.

The exact syntax of the various HTML control tags is discussed below. In all cases except buttons, the **NAME=** parameter must be the name of the variable being asked, in brackets-[ ]. This is the identifier that will be used for the user's input when the data is sent back to the CORVID Servlet Runtime.

For example, to ask for a string value that will be returned for variable [X], you could use:

```
<input type="text" name="[X]">
```

The value input would be assigned to variable "[X]".

For a template associated with a single variable, the name can be hard coded in the temp late. This works for a screen associated only with [X], but to allow a more generic screen, CORVID will automatically replace the string **VARIABLE_NAME** with the name of the variable being asked. This can be done anywhere in a CORVID_ASK section, but is mo st useful associated with name= parameter. The command:

```
<input type="text" name="[VARIABLE_NAME]">
```

could be used for any numeric, string or date variable. If it were used for the variable [TEMP], the parameter VARIABLE_NAME would be replaced with "TEMP " to produce:

```
<input type="text" name="[TEMP]">
```

If used for variable [COST], it would produce:

```
<input type="text" name="[COST]">
```

NOTE: The variable name MUST be in [ ]. The parameter VARIABLE_NAME will be replaced by just the name of the variable, so be sure to put square brackets, [ ], around the VARIABLE_NAME.

The name= parameter identifies the data that is returned to CORVID. In addition, the control needs a label to indicate what input is to be entered.

A template for a single variable can "hard code" this into the screen. For example:

```
The weight of the item is <input type="text"
name="[WEIGHT]">
```

However, it is better to make a CORVID_ASK section that can be used for multiple variables. To allow this, the replaceable parameter **VARIABLE_PROMPT** will be replaced with the prompt text for the variable being asked. If there are multiple prompts (e.g. different languages), CORVID will select the appropriate one based on the key variable.

So, if the CORVID_ASK section has:

```
VARIABLE_PROMPT <input type="text"
name="[VARIABLE_NAME]">
```

it can be used to ask any numeric, string or date variable. The prompt for the variable will be followed by an edit box. The value entered will be assigned to the appropriate variable.

Any of the HTML options for formatting text can be used for the prompt. Whatever formatting (size, color, font, etc.) is applied to the text "VARIABLE_PROMPT" will be applied to the actual prompt text when it is replaced. Any additional options for the <INPUT tag can also be used to format the edit field.

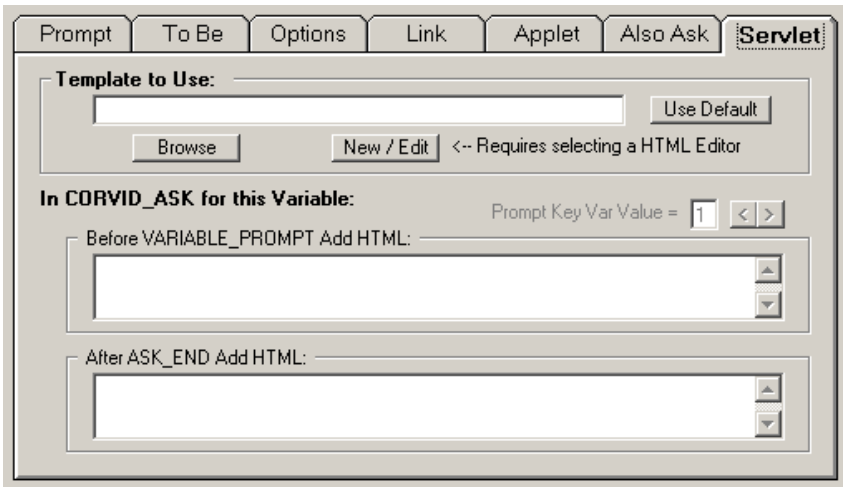
To build the command with an HTML editor, just use the text VARIABLE_PROMPT and VARIABLE_NAME when building the control and format them with the HTML editor. Whatever format you set for the text VARIABLE_PROMPT is the format th at will be used when "VARIABLE_PROMPT" is replaced by the actual prompt.

Note: When formatting replaceable parameters such as "VARIABLE_PROMPT", be sure to do it in a way that leaves "VARIABLE_PROMPT" as a single text string. If "VARIABLE_" was made red and "PROMPT" made blue, the string "VARIABLE_PROMPT" would be broken and not replaced.

The replaceable parameters VARIABLE_NAME and VARIABLE_PROMPT should only be used within a CORVID_ASK section. They must have the context of a specific variable to have meaning. The CORVID_ASK section can be quite general, such as CORVID_ASK VARIABLE, but the variable being asked by that section will still set the context.

5.3.3 Extra HTML Code for VARIABLE_PROMPT

The replaceable parameter VARIABLE_PROMPT is normally just replaced by the prompt for the variable. However, each variable can have additional HTML code added before the prompt.



This is done from the Servlet tab for the variable. Enter any HTML code into the "Before VARIABLE_PROMPT add HTML" edit box. This HTML will be added before the prompt text when "VARIABLE_PROMPT" is replaced.

If there are multiple prompts, multiple "Before VARIABLE_PROMPT" texts can be entered and they will be selected with the same key variable as used to select the prompt. Just click the < and > arrows above the "Before VARIABLE_PROMPT" box to select the appropriate key variable value.

Additional HTML code can also be added after the ASK_END in the CORVID_ASK section to ask the variable. Like the "Before VARIABLE_PROMPT" text, there can be multiple "After ASK_END" texts if there are multiple prompts.

For example, if you have several variables that should display a JPG image, and then ask the question below it. This could be done by having an "<IMG SRC=.." tag as part of t he text of the prompt. That would work, but adds extra text to all the places where the prompt is used in the systems. Instead, just have the prompt be the descriptive text and add the "<IMG SRC=..." in the "Before VARIABLE_PROMPT add HTML". This tag w ill be added before the prompt without it being part of the prompt. Since each variable can have a different image file, it will still work generically in the template.

The text in the "Before" and "After" should be simple HTML code, and not include CORVID commands such as CORVID_IF.

5.3.4 Numeric, String and Date Variables

When asking for numeric, string or date variables, the user's input will be a text string. This may be a numeric value, a date or just text. HTML provides controls allowing the user to input text in edit boxes. (If the variable only has specific values, you could use a list or set of radio buttons, but if that is the case, the variable probably should be a Static List.)

Within the <FORM section of the template, there must be a CORVID_ASK command that matches the variable. This can be:

<!-- CORVID_ASK VARIABLE -->…<!-- ASK_END -->

Section will be applied to all variables. Make sure to handle any Static List or Dynamic List variables in a CORVID_ASK section above this.

<!-- CORVID_ASK CONTINUOUS -->…<!-- ASK_END -->

Section will be applied to all numeric, string and date variables.

<!-- CORVID_ASK NUMERIC -->…<!-- ASK_END -->

Section will be applied to all numeric variables.

<!-- CORVID_ASK STRING -->…<!-- ASK_END -->

Section will be applied to all string variables.

<!-- CORVID_ASK DATE -->…<!-- ASK_END -->

Section will be applied to all date variables.

<!-- CORVID_ASK [VarName] -->…<!-- ASK_END -->

Section will be applied to the specific variable VarName.

<!-- CORVID_ASK [VarMask] -->…<!-- ASK_END -->

Section will be applied all variables matching the mask. Make sure to handle any Static List or Dynamic List variables that match the mask in a CORVID_ASK section above this.

Within the CORVID_ASK section for Continuous variables, there are 3 standard HTML controls that can be used which allow the user to type in text. (These are standard HTML commands, not a unique feature of CORVID, and more details on these commands can be fond in a manual on HTML.)

Simple Text Edit Box

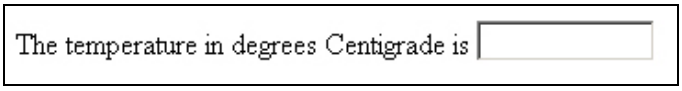
<input type="text" name="[VARIABLE_NAME]">

This allows a single line of text, which will be assigned to the variable. The tag allows an optional size="#" parameter that sets the size of the edit field, where # is a numeric value. The positio ning of the edit field is done using HTML formatting commands outside of the tag. The prompt for the variable can be displayed to the left of this tag or above it and should use the VARIABLE_PROMPT replaceable parameter.

For example, a template section t hat would work for all numeric variables:

```
<!-- CORVID_ASK NUMERIC -->
VARIABLE_PROMPT <input type="text"
name="[VARIABLE_NAME] size="12">
<!-- ASK_END -->
```

Using this to ask for the temperature would look like:



Password Edit Box

<input type="password" name="[VARIABLE_NAME]">

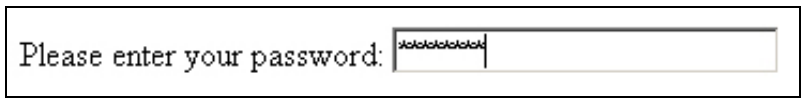
This also allows a single line of text, which will be assigned to the variable. It is similar to the simple edit box above, but the edit box will echo back "*" or some meaningless symbol instead of the user's input. This hi des the text that is typed in and should be used for passwords. The tag allows an optional size="#" parameter that sets the size of the edit field, where # is a numeric value. The positioning of the edit field is done using HTML formatting commands outside of the tag. The prompt for the variable can be displayed to the left of this tag or above it and should use the VARIABLE_PROMPT replaceable parameter.

Variable values are sent back to the servlet using POST, which is relatively secure compared to GET which makes the value part of the URL. However, CORVID does not encrypt the information itself and if highly sensitive password information is being provided by the user, a secure server connection should be used for added protection.

For example, to ask for the value of all variables starting with "PASSWORD":

```
<!-- CORVID_ASK [PASSWORD*] -->
VARIABLE_PROMPT <input type="password"
name="[VARIABLE_NAME]>
<!-- ASK_END -->
```

Using this to ask for a password would look like:



Larger, Multiline Edit Box

<textarea name="[VARIABLE_NAME]" cols="#" rows="#">
</textarea>

This allows inputting multiple lines of text, and handles scrolling etc. The value entered will be assigned to the variable. The cols="#" parameter that sets the number of columns in the edit box (width) and rows="#" sets the number of rows, where # is a numeric value. The positioning of the edit field is done using HTML formatting commands outside of the tag. The prompt for the variable can be displayed to the left of this tag or above it and should use the VARIABLE_PROMPT replaceable parameter.

Note that the tag requires the closing </textarea> tag to mark the end. If you wish to have text already in the edit box when it is displayed, put it before the </textarea> marker. In most ca ses you will want, the edit field will be blank and there will be no text between <textarea…> and </textarea>.

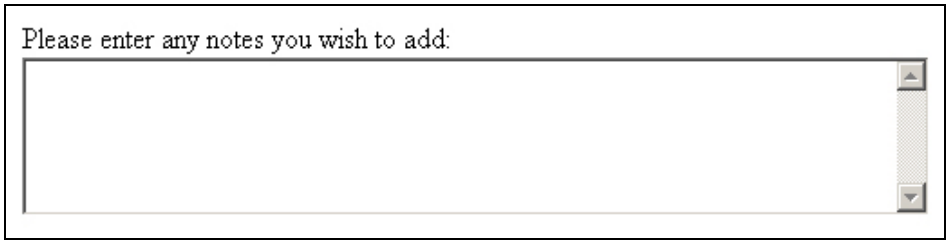
For example you could display a comment generated by the system to see if the user wishes to add to it. The system generated note is in the var iable [SysComment] which is included with double square bracket, [[ ]], embedding. The user's modified note will be sent to the variable [UserNote] (In this case hard coded into the template):

Please enter any notes you wish to add: <textarea name="[Use rNote]" cols="60" rows="5"> [[SysComment]]</textarea>

To ask for the value of all string variables:

```
<!-- CORVID_ASK STRING -->
VARIABLE_PROMPT <BR>
<textarea name="[VARIABLE_NAME]" cols="60"
rows="5">  </textarea>
<!-- ASK_END -->
```

Using this to ask for a note would look like:



5.3.5 Static and Dynamic List Variables

Asking for the value of a numeric or string variable is relatively simple since there is a single control (edit box) and the user just enters text which is returned to CORVID. Static and Dynamic List variables have a set of defined possible values that the user must select among. This requires multiple controls (check boxes or radio buttons) or a list with multiple entries. HTML code must be generated for each of the possible values. CORVID templates have special commands to allow generic CORVID_ASK sections to be used to handle any number of values associated with a variable.

When asking the user for input on a Static List or Dynamic List variable that has multiple values, the INPUT and SELECT tags can be used to build controls.

As with all controls except buttons, the "name=" parameter for the control must be the name of the variable being asked, in brackets, [ ]. In addition each control (check box, radio button or value in a list) has a "value=" parameter. This must be associated with the matching value for the variable. The "value=" parameter for a control MUST be the number of the associated value or the short text of the associated value. Since Dynamic List variables do not have a short text for the values, they can only use the value number.

For example, if there is a variable [WEATHER] and the 3rd value is "Rain or sleet", with short text for the value "rain", the following check box controls could be used:

```
<input type="checkbox" value="rain"
name="[COLOR]">Rain or sleet
```

or

```
<input type="checkbox" value="3" name="[COLOR]">Rain
or sleet
```

Note that even if the numeric value is used, it is a string in quotes.

To ask a question that provides the user with a group of va lues to select among, requires a set of check boxes or radio buttons, each with the same name, but different values. The template for a Static List variable can be "hard coded" into the template screen. To ask for [COLOR] which could have short value text of "Red" or "Blue", you could use:

```
The color is
<input type="checkbox" value="red"
name="[COLOR]">Reddish
<input type="checkbox" value="blue"
name="[COLOR]">Bluish
```

These checkboxes would return the value "red" or "blue" for the variable [COLOR]. The labels for the checkboxes would be "Reddish" and "Bluish".

In practice, having to build a screen for each variable, and keep it up -to-date would be very tedious. Instead generic templates can be designed for all Static and Dynamic List variables.

The replaceable parameters VARIABLE_PROMPT and VARIABLE_NAME can be used to make generic templates. To handle Static List variables, there are 3 additional replaceable parameters:

VARIABLE_VALUE_TEXT	The full text of a value
VARIABLE_VALUE_NUM	The number of a value
VARIABLE_VALUE_SHORT	The short text of a value (Static List only)

With these a generic form of the tag can be written:

```
<input type="checkbox" name="[VARIABLE_NAME]"
value="VARIABLE_VALUE_SHORT"> VARIABLE_VALUE_TEXT
                    or
<input type="checkbox" name="[VARIABLE_NAME]"
value="VARIABLE_VALUE_NUM"> VARIABLE_VALUE_TEXT
```

All that is missing is a way to apply this generic tag once for each value in the value list of a Static or Dynamic List variable. CORVID provides 2 ways to do this.

CORVID_REPEAT

The first is the CORVID_REPEAT command. This is added to the CORVID_ASK section of HTML code for a Static or Dynamic List variable with:

```
<!-- CORVID_REPEAT --> … <!-- REPEAT_END -->
```

The code between the CORVID_REPEAT and REPEAT_END commands will be repeated for each value in the variable's value list. The parameters VARIABLE_VALUE_TEXT, VARIABLE_VALUE_SHORT and VARIABLE_VALUE_NUM can be used in the CORVID_REPEAT section. They will be replaced with the associated value data. The first time through the CORVID_REPEAT, they will be given data from the first value. The second time through, the second value, etc for each value in the variable's value list.

So if [COLOR] has a prompt "The color is" and values:

Value Short Text	Full Value Text
red	Reddish
blue	Bluish
green	Greenish

Using:

```
<!-- CORVID_ASK STATIC_LIST -->
     VARIABLE_PROMPT <BR>
<!-- CORVID_REPEAT -->
     <input type="checkbox"
     value="VARIABLE_VALUE_SHORT"
     name="[VARIABLE_NAME]">VARIABLE_VALUE_TEXT
     <BR>
<!-- REPEAT_END -->
<!-- ASK_END -->
```

would produce:

```
The color is <BR>
<input type="checkbox" value="red"
name="[COLOR]">Reddish <BR>
<input type="checkbox" value="blue"
name="[COLOR]">Bluish <BR>
<input type="checkbox" value="green"
name="[COLOR]">Greenish <BR>
```

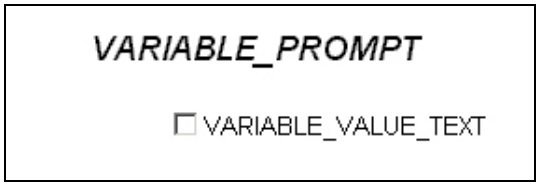
This will look like:

This same section could be used to ask any Static List variable with any number of values since all parameters will be obtained from the values in the system. This has rather simplistic formatting, but has all the elements to ask any Static List. If one Static List variable has 2 values and another 20, it does not matter, the appropriate number of check boxes will be created for each.

If we look at the template with an HTML editor, it will look like:

There is only a single check box bec ause the HTML editor does not understand the CORVID_REPEAT command, which in HTML is just a comment. The text VARIABLE_PROMPT or VARIABLE_VALUE_TEXT could be formatted for color, size, font, style, etc and that formatting would apply when the variable's values were filled in.

If the template was formatted in the HTML editor to look like:



When it was applied to the above variable, the formatting would be applied to the actual text and it would look like:

VALUE

The CORVID_REPEAT command makes it easy to design templates that include a control for each value in the list and is very useful when all the values are to be asked the same way. This is normally the case, but if you wish to apply some different formatting or add HTML code to some value s and not others, there is a second way to build screens for Static and Dynamic List variables.

The code for each specific value can be specified using:

```
<!-- VALUE # -->
```

where # is the number of the value.

All code between `<!-- VALUE 1 -->` and `<!-- VALUE 2 -->` will be to display value number 1. All replaceable parameters will be set according to the variable's first value. Then all code between `<!-- VALUE 2 -->` and `<!-- VALUE 3 -->` will be built according to the second value, etc. until the next HTML comment is reached.

Unlike a CORVID_REPEAT, you must know the maximum number of possible values and design the template accordingly. The template MUST have as many value # sections as the MAXIMUM number that any variable asked with that template will have. If a variable does not have as many values as there are value # sections in the template, the code in the sections that are higher than the number of values for that variable will be ignored.

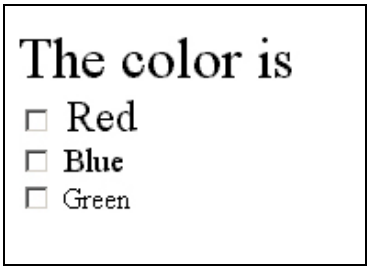
For example to decrease the font size of the text for each of 3 values value:

```
<!-- CORVID_ASK STATIC_LIST    -->
     <font size="6"> VARIABLE_PROMPT<BR></font>
<!-- VALUE 1 -->
<input type="checkbox" value="VARIABLE_VALUE_SHORT"
name="[VARIABLE_NAME]"><font size="5">
VARIABLE_VALUE_TEXT </font><BR>
<!-- VALUE 2 -->
<input type="checkbox" value="VARIABLE_VALUE_SHORT"
name="[VARIABLE_NAME]"><font size="4">
VARIABLE_VALUE_TEXT </font><BR>
<!-- VALUE 3 -->
<input type="checkbox" value="VARIABLE_VALUE_SHORT"
name="[VARIABLE_NAME]"><font size="3">
VARIABLE_VALUE_TEXT </font><BR>
<!-- ASK_END -->
```

This will work for variables that have 3 or less values. If the system had variables with more than 3 values, more VALUE # sections would need to be added.

If you look at this in an HTML editor, or browser, it will look like:

Unlike a CORVID_REPEAT template, there is a line for each possible value. If this were applied to a Static List variable, it would look like:



In most cases, this type of formatting is not needed and a CORVID_REPEAT is easier to use. However for special design situations, a VALUE # approach can be very useful. It is often used for putting values into a table, for example to have multiple rows with 2 values each. (There is also a way to do this with CORVID_REPEAT and CORVID_IF)

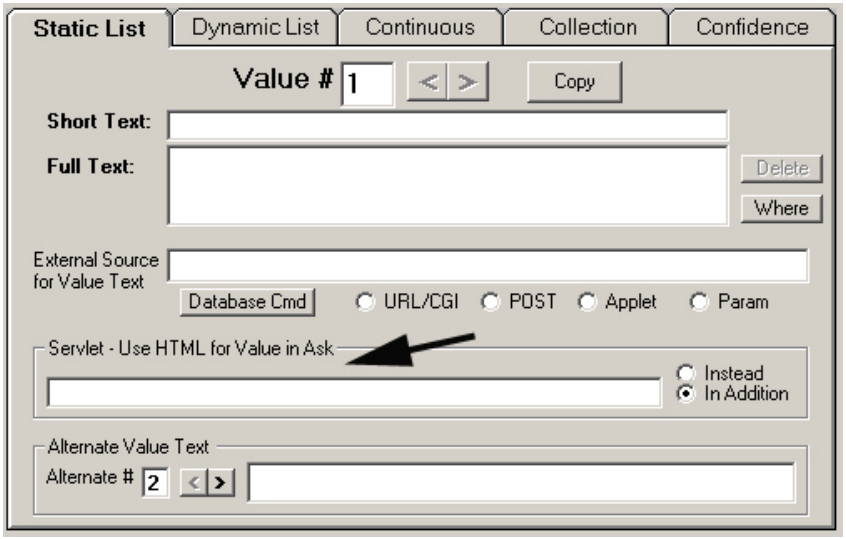
A CORVID_ASK section can use either CORVID_REPEAT or VALUE #, but not both in the same section. Different CORVID_ASK sections in the same template can use either CORVID_REPEAT or VALUE #.

The parameters VARIABLE_VALUE_TEXT, VARIABLE_VALUE_NUM, and VARIABLE_VALUE_SHORT should only be used within a CORVID_REPEAT or VALUE # section. They do not have any context or meaning outside of these sections.

5.3.6 Alternate Static List Value Text

When using Static List variables with a servlet template, you can have HTML code that is associated with individual values and used only when the variable is asked by the servlet runtime. This text can be used in place of, or in addition to, the value text in replacement of VARIABLE_VALUE_TEXT parameters.

The text is set from the Static List tab:



Text can be entered in the "Servlet - Use HTML for Value in Ask" edit field. This way graphics or other HTML commands can be added, without having to embed the code into the value text. This makes the text easier to read, while still allowing more complex screens.

Text can be added for each value individually. This text can then be marked to be used in addition to the value text, or instead of the value text when asking a question by selecting the appropriate radio button to t he right of the edit box. When the "In addition" radio button is selected, the text will be added before the normal value text. When "Instead" is selected, the value will be used instead of the value text. The text should be normal HTML and should not include any CORVID commands or replaceable parameters.

For example:

To add a graphical element, such as an arrow, before each value in the list:

Add an IMG SRC tag, such as <IMG SRC="arrow.jpg"> in the edit box and click the "In Addition" radio button. This will need to be done for each value in the value list.

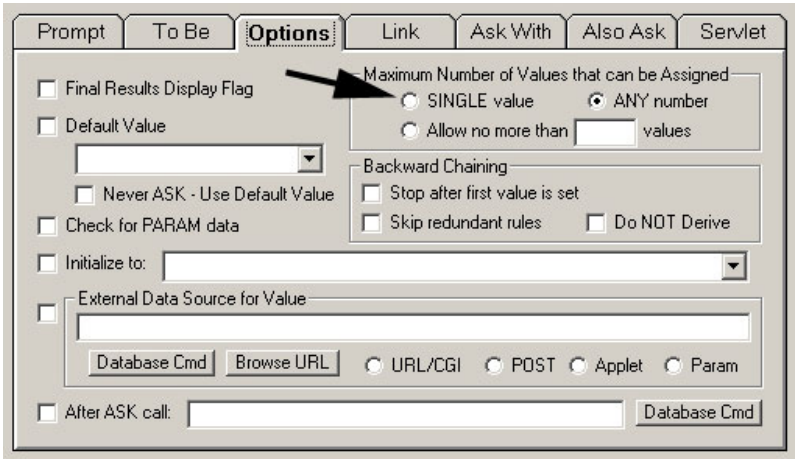
To keep the value text easy to read, use a jpg image in place of the text when asking the question from the servlet:

For value 1, add an IMG SRC tag, such as <IMG SRC="img1.jpg"> in the edit box and click the "Instead" radio button. Repeat for each value in the value list with the image file appropriate to that value.

5.3.7 Checkbox / Radio Button

In many systems, some variables are only allowed to have a single value and should use radio buttons (or lists), and should not use checkboxes that may allow more than one value to be selected at the same time. Other variables can accept multiple values and should use checkboxes. To allow the template to be generic, **always design with checkboxes unless radio buttons should always be used**. For those variables that are limited to only a single value, CORVID will automatically convert the checkboxes to radio buttons for that variable. Variables that can have multiple values will use the checkboxes.

A variable is set to allow only a single value from the Variable window by clicking the Options tab. Set the "Maximum Number of Value that be Assigned" to "Single Value". Then any CORVID_ASK section that is specified for "checkbox" will automatically have it converted to "radio".



5.3.8 CORVID_ASK and [.PROPERTY]

Any text in the CORVID_ASK section can include a property of the variable associated with that section. Any property can be added generically by using **[.property]** This will be converte d to [varname.property] for the variable being asked. This can be used as text to display or in expressions.

For example to generically have the CORVID_ASK section tell the user how many values they can choose from, you could use the [varname.COUNT] property that returns the number of values that the variable has.

```
  <!-- CORVID_ASK STATIC_LIST -->
       Select among the [[.COUNT]] values below.  <BR><BR>
       VARIABLE_PROMPT <BR>
<!-- CORVID_REPEAT -->
       <input type="checkbox" value="VARIABLE_VALUE_SHORT"
       name="[VARIABLE_NAME]">VARIABLE_VALUE_TEXT <BR>
<!-- REPEAT_END -->
<!-- ASK_END -->
```

NOTE: Only properties that are defined when the question is asked should be used. A property such as .COUNT is defined during development, however a property such as .VALUE may only be known after the question is asked and should NOT be used in the screen to ask the question. This could result in an infinite loop.

5.3.9 Static and Dynamic List Control Types

There are really only 2 HTML tags that are used for Static and Dynamic List variables, but they have optional parameters to produce more controls.

Simple Set of Checkboxes or Radio Buttons

**<input type="checkbox" value="VARIABLE_VALUE_SHORT" name="[VARIABLE_NAME]">VARIABLE_VALUE_TEXT
**

This adds a single checkbox for a value. It should be used in a CORVID_REPEAT or VALUE # section to handle the multiple values that a variable may have. In this case the
 is used to have each value on a new line. If all values should be on the same line, the
 should be omitted.

The template can always be designed with type="checkbox". This will automatically be converted to type="radio" if the variable only allows a single value to be set.

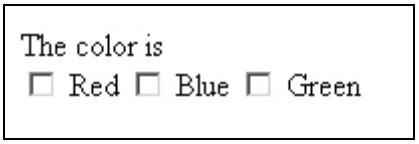
If the system has Dynamic List variables that may be asked using the template, the replaceable parameter VARIABLE_VALUE_SHORT should be changed to VARIABLE_VALUE_NUM since Dynamic List variables do not have short value text and must be referenced by number.

For example, to ask for the value of all Static List variables:

```
<!-- CORVID_ASK STATIC_LIST -->
VARIABLE_PROMPT <BR>
<!-- CORVID_REPEAT -->
<input type="checkbox" value="VARIABLE_VALUE_SHORT"
name="[VARIABLE_NAME]">VARIABLE_VALUE_TEXT <BR>
 <!-- REPEAT_END -->
<!-- ASK_END -->
```

Using this to ask for the variable [COLOR] would look like:

If the
 in the CORVID_REPEAT section was removed to put all values on one line, it would look like:



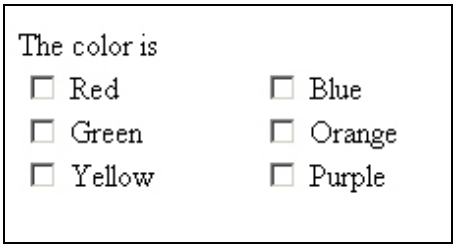
For more complex formatting of the values, such as multiple rows with 2 values each, use VALUE # in a table, such as:

```
VARIABLE_PROMPT<BR>
<table border="0" cellpadding="0" cellspacing="2"
width="462">
<tr>
<!-- VALUE 1 --->
<td><input type="checkbox" value="VARIABLE_VALUE_SHORT"
name="[VARIABLE_NAME]"> VARIABLE_VALUE_TEXT
</td>
<!-- VALUE 2 --->
```

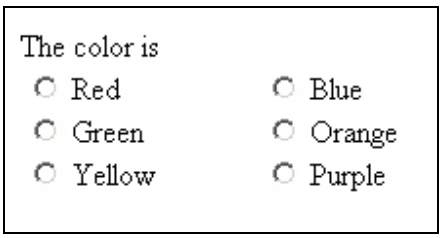
```
<td><input type="checkbox" value="VARIABLE_VALUE_SHORT"
name="[VARIABLE_NAME]"> VARIABLE_VALUE_TEXT

</td>

</tr>

<tr>

<!-- VALUE 3 --->

<td><input type="checkbox" value="VARIABLE_VALUE_SHORT"
name="[VARIABLE_NAME]"> VARIABLE_VALUE_TEXT

</td>

<!-- VALUE 4 --->

<td><input type="checkbox" value="VARIABLE_VALUE_SHORT"
name="[VARIABLE_NAME]"> VARIABLE_VALUE_TEXT

</td>

</tr>

<tr>

<!-- VALUE 5 --->

<td><input type="checkbox" value="VARIABLE_VALUE_SHORT"
name="[VARIABLE_NAME]"> VARIABLE_VALUE_TEXT

</td>

<!-- VALUE 6 --->

<td><input type="checkbox" value="VARIABLE_VALUE_SHORT"
name="[VARIABLE_NAME]"> VARIABLE_VALUE_TEXT

</td>

</tr>

</table>
```

In this case, the VALUE # approach was used since the <TR> </TR> that define the rows in the table occur with every second value. When using CORVID_REPEAT, the same code must occur with every variable in the same way. Using VALUE # allows formatting into the table easily.

When applied to a variable [COLOR] with 6 values, it would look like:



If [COLOR] was set to only allow a single value, the "checkbox" would automatically be converted to "radio" and it would look like:



Value Lists and Dropdown Lists

<select name="[VARIABLE_NAME]" size="#">

<!-- CORVID_REPEAT –>

<option value="VARIABLE_VALUE_SHORT">
VARIABLE_VALUE_TEXT

</option>

<!-- REPEAT_END -->

</select>

This will create a list of values. If the size is set to 1 (size="1"), then the control will be a dropdown list with only the selected value displayed. If the size is set to a higher value, the control will be a list with the number of rows displayed equal to the size value and a scroll bar if needed.

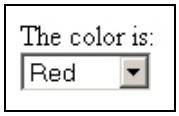
The OPTION tag defines the values in the list. It should be used within the CORVID_REPEAT command to add all the values into the list.

If the system has Dynamic List variables that may be asked us ing the template, the replaceable parameter VARIABLE_VALUE_SHORT should be changed to VARIABLE_VALUE_NUM since Dynamic List variables do not have short value text and must be referenced by number.

For example, to ask for the value of all Static List varia bles:

```
<!-- CORVID_ASK STATIC_LIST -->
VARIABLE_PROMPT <BR>
<select name="[VARIABLE_NAME]" size="1">
<!-- CORVID_REPEAT -->
<option value="VARIABLE_VALUE_SHORT">
VARIABLE_VALUE_TEXT
</option>
<!-- REPEAT_END -->
</select>
<!-- ASK_END -->
```

Using this to ask for the variable [COLOR] would look like:



Clicking on the down arrow next to "Red" would drop down to display the other values that could be selected.

If the same code had the size= parameter changed to size="4", a list control would be produced:

When a list is used, and the size= parameter is greater than 1, "multiple" should be added after the size= to indicate that multiple values can be selected from the list.

```
<select name="[VARIABLE_NAME]" size="1" multiple>
```

If the variable only allows a s ingle value to be selected, CORVID will automatically remove the "multiple" and build a list control that only allows a single value. Adding "multiple" makes a more generic control that will work correctly for all Static and Dynamic List variables. If al l the variables in a system only allow a single value the "multiple" option can be left off.

Simple Buttons

**VARIABLE_PROMPT
**

<!-- CORVID_REPEAT –>

<input type="submit"

value="[VARIABLE_NAME]=VARIABLE_VALUE_NUM"
name="VARIABLE_VALUE_TEXT">

<!-- REPEAT_END -->

Most of the question controls require that the user select a value from a list or by checking an item, and then click the OK button. This approach has the advantage that the user can review their selection and make changes before submitting it. However, for some systems, a better end user interface is one that just allows the user to click on a button and have the system immediately take the input. This is especially true if the system asks multiple Yes/No or True/False questions.

This type of interface can be implemented by using buttons created by <input type="submit". The syntax for the buttons is somewhat different than any of the other controls. The value associated with the control must be the string:

```
"[VarName]=value"
```

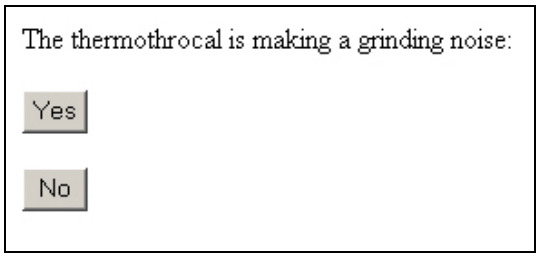
where VarName is the name of the CORVID variable and value is the value to assign. For Static List variables, value can be the value number or the short text of the value. A generic way to do this is to use:

```
value="[VARIABLE_NAME]=VARIABLE_VALUE_NUM"
```

The name= parameter of the control is what defines the label for the button. If the variables have relatively short value text, this text can just be put on the button with:

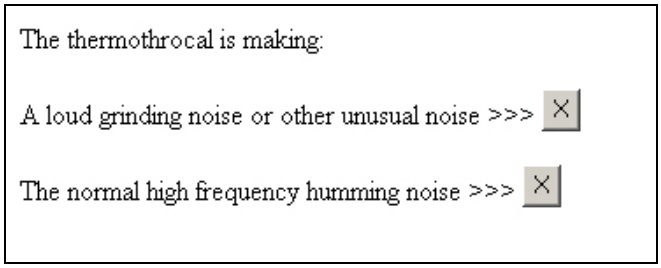
```
name="VARIABLE_VALUE_TEXT"
```

For example:



If the text is too long to put on the butto n, it can be put next to the button, with the button having an "*", image or some other generic label. This can be done with:

```
VARIABLE_PROMPT<BR>
<!-- CORVID_REPEAT -->
VARIABLE_VALUE_TEXT
<input type="submit"
value="[VARIABLE_NAME]=VARIABLE_VALUE_NUM" name="X">
<!-- REPEAT_END -->
```



Note, when using buttons to ask a question the OK button is not needed and should not be used. The UNDO and RESTART buttons can be used.

5.4 The Submit Button

Each template that uses a form (except those using butt ons to ask the question) must have a submit button to send the data back to the servlet engine and continue processing of the system. The typical submit button is:

```
<input type="submit" name="OK" value="OK">
```

The NAME= parameter for the button should be " OK". The VALUE= parameter will be the label on the button. It can be text such a "Enter", "Submit", "Continue" or any other text that you would like to use.

If the system needs to work in multiple languages, a double square bracket, [[ ]], replacement can be used for the name of the button. This variable would be set to a value appropriate to the language. Such as:

```
<input type="submit" name="OK"
value="[[OK_Btn_label.VALUE]]">
```

All submit buttons must be part of the form. They must occur in the HTML between the <form…> and </form> tags.

The last screen in a system (typically a result screen) may have no "OK" button, and instead have only a RESTART button or a link back into your overall web site. This is legal in any case where there is no further processing to be done by the system, and the only action the user can take is to restart another session, or go to another page.

5.4.1 UNDO and RESTART buttons

There are 2 special submit buttons - UNDO and RESTART. A template that uses a form can optionally contain both UNDO and RESTART buttons. If the UNDO button is clicked, CORVID will move back to the previous question in the system and allow the user to change their answer. RESTART will return to the start of the system and, in effect, start a ne w session.

Both buttons are SUBMIT buttons with the following syntax:

```
<input type="submit" name="…" value="…">
```

and **MUST** have

name="~UNDO"	for the UNDO button
name="~RESTART"	for the RESTART button

The VALUE= parameter that defines the la bel on the button can be anything you wish. For example:

```
<input type="submit" name="~UNDO" value="Step Back">
<input type="submit" name="~RESTART" value="Start Over">
```

As with the OK button, these buttons must be part of the form and occur in the HTML be tween the <form…> and </form> tags. Often the question templates will have OK, UNDO and RESTART buttons, but only a RESTART on the results template.

If the final screen in the system (typically the results) has an OK button, it will take the user to the Final Screen set in the properties window (or the CORVID default if none is set.) The Final Screen can display any other information, but should not contain an OK. At most it should contain a RESTART button, or just a link to other parts of the site.

When a template has an UNDO or RESTART button, CORVID checks that it is a valid option at the time. CORVID will automatically disable the UNDO button when there is no UNDO available, and will disable RESTART on the first screen of the system. This allows a generic question template to include OK, UNDO and RESTART even though there are times when some may not be active.

5.5 The Browser BACK button

The Browser BACK button can be used to move back to a previous question. CORVID embeds a hidden value in ea ch page to identify if the user pressed the BACK button one or more times. This allows them to step back to an earlier question in the system, change answers and continue the session.

To do this, the CORVID servlet stores certain data on the server abo ut the session. The amount of time that the server keeps this data varies with the installation of the servlet engine. If a user waits a long time (typically over an hour) and then tries to go back to an earlier CORVID session, the needed data may no lon ger be available and BACK will not work. Likewise, bookmarks for a particular question will not work and will instead result in a restarting the system.

If it necessary for the user to return to a session days later and have access to their earlier data, that data will need to be stored to a database for permanent storage and recovered when the user returns.

5.6 TRACE

There are 2 special replaceable parameters that can be used to display the CORVID system trace. These should be used only when devel oping the system and removed before final release.

The parameter CORVID_TRACE will be replaced by the full trace information. This is the same trace that would be echoed to the trace window in the applet version.

1. In the knowledge base, select the "Add Trace Applet / Enable Trace in Servlet" checkbox in the Properties window. This turns trace on.

2. In the templates that you wish to include a trace, add the replaceable parameter "CORVID_TRACE" where the trace is to be displayed. This can be anywhere in the page, but normally this is at the bottom of the page after the end of the form (</FORM> tag) but before the end of the body (</BODY> tag)

The trace will be displayed in the same font and color as specified for the CORVID_TRACE parameter, how ever if an error is detected, the trace color will be changed to red. The background selected for the trace should allow it to be readable if it changes to red.

CORVID_TRACE will put the entire trace up to that point in the screen. This can get rather long for some systems (especially metablock systems). To see a shorted version of the trace, use the replaceable parameter CORVID_TRACE_CLEAR. This will display the trace, but will skip earlier portions. This can be easier to read, while still including the main information for the last step.

Another way to do trace is to trace to the servlet log file. This can be done by selecting "Trace to Java Console" in the Properties window for the system. Since often there is no Java console for the servlet, t he trace is echoed to the log and can be read there. **This should ONLY be used in development and ONLY when there is only a single user running the system.** If there are multiple users, the traces will intermingle producing a very difficult read trace.

If the trace only needs to be seen only occasionally, another approach can be used. A SUBMIT button similar to the UNDO or RESTART button can be added to the template as part of the form. This button should have:

```
<input type="submit" name="…" value="Display Trace">
```

where

name="~TRACE"	displays the entire trace
name="~TRACE_CLEAR"	displays the trace up to that point and then clears it

When the TRACE button is hit, the next screen will display the trace up to that point and then return to the s ystem. The ~TRACE_CLEAR will display the trace and then erase earlier portions.

5.7 Conditional Inclusion

Sections of a template file can be included based on a conditional test. This can be done in both templates to ask questions and those to dis play results.

This is done using:

```
<!-- CORVID_IF  expr -->  …  <!-- END_IF -->
```

The expression *expr* can be any CORVID expression that will evaluate to true or false. (e.g. [X] > 0). If the expression is true, all the code up to the associated END_IF will be included. If the expression is false, all code up to the associated END_IF will be ignored and not included in the page produced.

CORVID_IF blocks can be nested, but each must have an associated END_IF. (In place of END_IF, IF_END can also be used for consistency with other commands.)

The CORVID_IF can be either in a CORVID_ASK section, or outside of it, but must be either entirely in or outside the section. The following are legal:

```
<!-- CORVID_IF  expr -->
<!-- CORVID_ASK  var -->
      ...
<!-- ASK_END -->
<!-- END_IF -->
```

```
<!-- CORVID_ASK  var -->
      ...
     <!-- CORVID_IF  expr -->
      …..
<!-- END_IF  -->
...
<!-- ASK_END  -->
```

```
<!-- CORVID_IF  expr -->
...
<!-- END_IF -->
<!-- CORVID_ASK  var -->
...
<!-- ASK_END -->
```

The followin g are NOT legal because they cross CORVID_ASK sections:

```
<!-- CORVID_IF  expr -->
<!-- CORVID_ASK  var -->
                  <!-- END_IF -->   …..
<!-- ASK_END -->
```

```
<!-- CORVID_ASK  var -->
<!-- CORVID_IF  expr -->
            …
<!-- ASK_END -->
<!-- END_IF -->
```

The expression in the CORVID_IF can be any Boolean expression using CORVID variables. The HTML code within the CORVID_IF section will be included if the expression is true and ignored if the expression is false.

The expression in the CORVID_IF should NOT con tain variables that do not have values and which would need to be asked of the user. Since the expression will be evaluated midway through the generation of the page, CORVID can not simply change and ask another page, which would mix the 2 pages.

The variables used in the expression should be asked or derived at the start of the system before they are needed. For example, the system might initially ask if the user is running from a palmtop computer. If the answer is YES, simple screens formatted for a smaller size might be used. If the answer was no, full screens could be used.

5.8 CORVID Commands

The same CORVID commands that are executed in a Command Block can be executed from within the template. Just include:

```
<!-- CORVID_CMD  command -->
```

where *command* is any single command that is legal in a Command Block. The Command Block commands FOR, WHILE, FOR_EACH and IF are not allowed since they can only be used in the context of the Command Block. The commands can use any CORVID variables and th eir properties. To assign a value, use the SET command.

For example, if you want to keep track of how many times a question is asked, a specific template could be associated with the variable. It could include:

```
<!-- CORVID_CMD  SET [COUNT]  ([COUNT] + 1) -->
```

You could then initialize [COUNT] to 0 at the start of the system and use the value of [COUNT] in a CORVID_IF statement to control how the question is asked. For example, if a question was not answered the first time, the system could automatically ask it in more detail the next. This could also be used to provide more details if the user input was not in a valid format to be accepted. For example, if a date variable had to be reasked, it could ask with more detail on the expected input format.

You could even build a system with an attitude by adding something like:

```
<!-- CORVID_IF  [COUNT] > 5 -->
I've asked this [[COUNT]] times - how about an
answer!<BR>
<!-- END_IF -->
```

NOTE: Any CORVID command can be included in the template, including ones that control the logical functioning of the system. However, ONLY commands related to the appearance of the user interface SHOULD be used. Including commands that control the logic will cause a system to run differently in the servlet mode than when run as an applet (which would not use the template). It would also be much more difficult to understand why the system was doing something, which might be dependent on a particular screen being displayed, complicating development and maintenance.

5.9 Using Image Maps to Ask Questions

In most cases, the HTML screens that ask a user to answer a question are forms that use POST to return the data to the servlet. However, a system can use simple HTML links to return data to the servlet. This includes image maps and individual images or text that have associated HTML links.

The key in doing this a link that calls back to the CORVID Servlet Runtime, with information that identifies the session and the data to be returned. CORVID provides a simple replaceable parameter that allows you to do this easily:

CORVID_SERVLET_GET

The parameter CORVID_SERVLET_GET will be replaced by a call to the CORVID Servlet Runtime along with information that will identify the place in the session. The link back to the CORVID Ser vlet Runtime must NOT be hard coded, since the additional information will not be know until the system is run.

Since the GET approach will be used to send the data, the parameter has the _GET extension. This approach adds the data to the URL that is u sed. When returning data via GET, CORVID adds some additional information, which is normally passed as hidden fields in the POST data sent by the forms. The URL back to the CORVID servlet is specified by the value set for CORVID_SERVLET in the Properties window user the Servlet tab.

To use CORVID_SERVLET_GET, just add a normal link to an image map, image or text that returns a value to CORVID. Then make the link "CORVID_SERVLET_GET " followed by the name of the variable in [ ], an "=", and the value to assign. If there are multiple values, separate them with "&". There should not be any spaces between data. If there are spaces or other characters in the value that require URL encoding, they should be encoded.

For example, the link off an image (or s ection of an image map) that sets the value of [X] to 5 might be:

```
href="CORVID_SERVLET_GET [X]=5"
```

CORVID will replace the CORVID_SERVLET parameter with the associated value, plus a "?", the session identification, an "&", and the data you provide.

For Static List variables, the value assigned can be a numeric value that is the number of the value, or the short text of the value. For Numeric variables, the value would be numeric. For string variables, the value would be a string. It would not need to be in quotes, but might need URL encoding.

For a generic screen that asks a question and assigns the value 5, you could alternatively use:

```
href="CORVID_SERVLET_GET [VARIABLE_NAME]=5"
```

This would be unusual since almost all cases that would use this tec hnique will require a specific page or image map. Normally image map screens are not used generically and are associated with only a single variable, so using the actual name is not a problem. However, using VARIABLE_NAME in templates does allow the variable's name to be changed in the system without having to edit the template.

If you wished to return data for more than one variable, you can make a list of values separated by "&", for example to set [X] to 5 and [Y] to 2:

```
href="CORVID_SERVLET_GET [X]=5&[Y]=2"
```

Remember not to add any spaces and to URL encode any characters that require it.

An example of a page that uses an image map to ask a question is:

```
<html>
      <head>
            <title>Image Map Demo</title>
      </head>

      <body>
        <BASE href="CORVID_LINK_BASE">

         <div align="center">
```

```
<img src="MyImage.gif"
 width="620" height="120" border="0"
 usemap="#map1162e6c">
 <map name="map1162e6c">
 <area shape="rect" coords="497,29,613,115"
     href="CORVID_SERVLET_GET [X]=1">
 <area shape="rect" coords="377,31,488,114"
     href="CORVID_SERVLET_GET [X]=2">
 <area shape="rect" coords="255,29,368,114"
     href="CORVID_SERVLET_GET [X]=3">
 <area shape="rect" coords="138,27,249,111"
     href="CORVID_SERVLET_GET [X]=4">
 <area shape="rect" coords="12,27,126,114"
     href="CORVID_SERVLET_GET  [X]=5">
 </map>


  </div>
</body>
</html>
```

This image map has 5 regions with links. A cli ck on any of the regions will set the value of the variable [X].

Using image maps is quite easy with an HTML editor and allows very advanced user interfaces to be created. An alternative to an image map is to have several image files (jpg or gif) on a pa ge with the same type of CORVID_SERVLET_GET link off them. A click on any of the images would set the associated value. This provides another way to ask questions with images.

5.9.1 Eliminating the OK button

Another way that links can be used is to build a screen that has no OK button. In the forms approach to templates, the user selects an answer and then must take the second step of clicking the OK button to submit the data. There are advantages to this since the user has a chance to review and c orrect any incorrect selections. However, in some systems the desired interface is to have the system immediately take the user's input and continue the session. This happens automatically with image maps and approaches that use the link approach to return data.

For example, if a system asked a serried of "Yes/No" questions, graphics could be used with links that would return that data. This can even be used generically, provided that the variables in the system are designed to use the values of "Yes" and "No" consistently.

A simple screen that would use this approach is:

```
<html>
<head>
<title>Link Demo</title>
<BASE href="CORVID_LINK_BASE">
</head>
<body >
<!-- CORVID_ASK STATIC_LIST -->
   VARIABLE_PROMT<BR><BR>
    <a href="CORVID_SERVLET_GET [VARIBLE_NAME]=YES">
       <img src="yes.jpg">
    </a>
    <a href="CORVID_SERVLET_GET [VARIBLE_NAME]=NO">
       <img src="no.jpg">
    </a>
<!-- ASK_END -->
</body>
</html>
```

This template could be associated with any Static List variable that had "Yes" and "No" as the possible values. The system will display the text of the prompt and 2 images for "Yes" and "No". A click on either image would immediately return the data to the CORVID Servlet Runtime.

6. Templates to Display Results

CORVID templates provide a flexible way to display the system results and recommendations in a variety of ways. It is easy to build very attractive Results screens, and with the powerful commands available even very complex screens can be built. Templates to display results are in many respects very similar to the templates to ask questions. The main difference is that instead of presenting multiple options for the user to select among, the template must display the appropriate items of information as a result or recommendation.

6.1 Forms

As with question templates, most results screens and reports should be HTML forms. This means that they have a

```
<form method="post" action="CORVID_SERVLET">
content
<input type="submit" name="OK" value="OK">
</form>
```

where the content is the actual report or results. The

```
method="post" action="CORVID_SERVLET"
```

is required to return to the CORVID servlet engine so it can continue processing the system.

Any report or information screen that is displayed before the system is finished MUST be a form and MUST include an OK button. A screen displayed at the end of a run often will not have an OK button, but will generally have a RESTART button or a link to some other page.

The only exception to using a form is a screen displaye d at the very end of a run, where the end user is not going to be given an option to restart the system. In that case, the screen does not need to be a form, but can still use all of the CORVID servlet commands for building the report. Such a screen should have some way for the user to navigate through links to other parts of the site.

6.2 Displaying Variables in Results

The intent of a results screen is to display the values of certain variables that had values set during a run, and to format this dat a in specific ways. Results and reports typically display Confidence or Collection variables set during a session, though other types of variables can also be included.

The easiest way to display results is to embed the individual variables, with [[ ]] embedded in the text. This allows any data in the system to be displayed.

For example, if the goal of a system was to set the value for [AMOUNT] based on various factors and logic, and that value was all the user needed to know, a Results screen could just contain the text :

"The amount to add is [[AMOUNT.VALUE]]"

If this was part of a Result template, the derived value of [AMOUNT] would be embedded. Note that the .VALUE property is used to have only the value added. Otherwise, the prompt and valu e would be displayed. If the prompt for [AMOUNT] was the string you wish to use in the results, just [[AMOUNT]] could be used with no additional text.

Using double square bracket, [[ ]], embedding is sometimes a very effective way to build reports. However, most of the time there are a large number of possible variables to display, and only certain ones should be displayed. For example, all the confidence variables that received a value of greater than 50.

CORVID provides a way to have a section of th e template that is repeated for multiple variables. This is done using:

```
<!-- FOR_EACH  VarID -->  … <!-- EACH_END -->
```

This allows a section of the template to be applied to each variable specified by the VarID. The values of VarID are the same as those used in CORVID_ASK when asking a question, plus additional values to display Collection and Confidence variables

The VarID indicates which variable(s) the code applies to. The allowed values are:

VARIABLE	All variables
STATIC_LIST	All Static List variables
DYNAMIC_LIST	All Dynamic List variables
CONTINUOUS	All numeric, string and date variables
NUMERIC	All numeric variables
STRING	All string variables
DATE	All date variable
CONFIDENCE	**All Confidence Variables**
CONFIDENCE_ASCENDING	**All Confidence Variables, sorted in order of value, with lowest values first**
CONFIDENCE_DECENDING	**All Confidence Variables, sorted in order of value, with highest values first**
COLLECTION	**All Collection variables**
[VARNAME]	The specific variable varname
[VARMASK]	All variables fitting the mask pattern

Replaceable parameters can be used for FOR_EACH. The most useful parameters are:

VARIABLE_PROMPT	The prompt of the variable.
[.PROPERTY]	Evaluated as [Var.PROPERTY] for all the legal properties of the variable.

Within the FOR_EACH, a CORVID_IF test can be used to select only certain variables.

For example, to display all the Confidence variables that received a value of greater than 20:

```
<!-- FOR_EACH CONFIDENCE -->
<!-- CORVID_IF ([.VALUE] > 20) -->
[[.FULL]]<BR>
<!-- END_IF -->
<!-- EACH_END -->
```

When this runs, the code between FOR_EACH and EACH_END will be applied to each Confidence variable in turn. The Confidence variables will be tested in the order that they are defined in the system. The first Confidence variable will have [.VALUE] replaced by [ConfVar1.VALUE] and evaluated, where ConfVar1 is the name of the first Confidence variable. If the value is greater than 20, the [[.FULL]] will be replaced by [[ConfVar1.FULL]]. This will be repla ced by the text of the confidence variable and its assigned value. The same lines of code will be repeated for each Confidence variable. Each one with a value greater than 20 will be included in the report. To have the Confidence values sorted with ones with the highest confidence values displayed at the top, just change "CONFIDENCE" in the FOR_EACH to "CONFIDENCE_DECENDING".

To display the values of all variables in the system, just use:

```
<!-- FOR_EACH VARIABLE -->
[[.FULL]]<BR>
<!-- EACH_END -->
```

6.2.1 Collection Variable Values

Collection variables are a powerful tool for handling expert system results. The Collection variable provides a freeform way to combine and sort recommendations and build reports.

One very powerful technique is to assign s tring values to a Collection variable that are themselves HTML commands. This allows the value of the Collection variable to be a block of HTML that can just be double square bracket, [[ ]], embedded in the report. This technique can be used to create complex interfaces such as building a table that displays the results. Just build the table in a collection variable as the system runs and then embed the value in the report. For some complex formatting, this can be much easier than using FOR_EACH commands.

The individual values in a Collection variable can be displayed using commands very similar to the way Static List values are handled in CORVID_ASK screens.

The CORVID_REPEAT command can be used to step through each value in a Collection variable's va lue list. The syntax is the same as in screens to ask questions:

```
<!-- CORVID_REPEAT --> … <!-- REPEAT_END -->
```

however, here it is used with a FOR_EACH command to define the variable it applies to. (NOTE: The FOR_EACH must be used even if there is only a single Collection variable being displayed. The FOR_EACH defines the context for the CORVID_REPEAT)

Within the CORVID_REPEAT for Collection variables, replaceable parameters can be used to obtain the text of the individual value, the number of the value and the "score" the value was given if the Collection variable had values added with .ADDSORTED.

VARIABLE_VALUE_TEXT	Text of the specific individual value
VARIABLE_VALUE_NUM	Number of the individual value
VARIABLE_VALUE_SORT	Sort value (Value must have been added to the Collection with ADDSORTED)

For example, to display all the values in a Collection variable [Comment] as a numbered list:

```
<!-- FOR_EACH [Comment] -->
<!-- CORVID_REPEAT -->
      VARIABLE_VALUE_NUM:  VARIABLE_VALUE_TEXT <BR>
<!-- REPEAT_END -->
<!-- EACH_END -->
```

If [Comment] had values "aaa", "bbb", and "ccc", this would display:

```
1: aaa
2: bbb
3: ccc
```

The "FOR_EACH [Comment]" is required even though there is only a single variable to set the context for the CORVID_REPEAT.

If the values in a Collection variable [Notes] had been added sorted, the values with a sort value of greater than 25 could be displayed with:

```
<!-- FOR_EACH [Notes] -->
<!-- CORVID_REPEAT -->
<!-- CORVID_IF  (VARIABLE_VALUE_SORT > 25) -->
      VARIABLE_VALUE_TEXT <BR>
<!-- END_IF -->
<!-- REPEAT_END -->
<!-- EACH_END -->
```

A second way to display specific values in a Collection variable is with the VALUE # command. This is very similar to the way specific values are reference when asking for Static List values with CORVID_ASK. The syntax is:

```
<!-- VALUE # -->
```

In a FOR_EACH section for a Collection variable, all code between `<!-- VALUE 1 -->` and `<!-- VALUE 2 -->` will apply to the first value in the Collection variable's value list. All code between `<!-- VALUE 2 -->` and `<!-- VALUE 3 -->` will apply to the second value in the list, etc. If there is no matching value for the one specified, the section will be skipped.

For example, to display the first 3 values of Collection variable [Notes] in decreasing font size:

```
<!-- FOR_EACH [Notes] -->
<!-- Value 1 -->
```

```
<font size="5"> VARIABLE_VALUE_TEXT <BR>
<!-- Value 2 -->
<font size="4"> VARIABLE_VALUE_TEXT <BR>
<!-- Value 3 -->
<font size="3"> VARIABLE_VALUE_TEXT <BR>
<!-- EACH_END -->
```

If [Notes] had values "aaa", "b bb", "ccc", this would display:

aaa

bbb

ccc

6.3 Title and Information Screens

Title and information screens can be displayed in a system.

The command to display a title screen is the TITLE command in a Command Block, and other HTML screens can be displayed with the DISPLAY_HTML command. While these commands are similar in intent, they are handled quite differently in CORVID.

The TITLE command is intended to display a title screen. This is defined with CORVID Screen Commands that are used only in the applet runtime. The command can have a "SERVLET=" parameter added which provides the name of the template to use for the title when running with the servlet runtime. This template can use all of the commands that are supported for templates, including conditional inclusion of sections, display of variables, etc. The TITLE command should be used with the appropriate template. This template should be a form with "action=CORVID_SERVLET" and an OK submit button.

For the servlet version, there could be multiple TITLE commands, since each has a "servlet=" parameter that could specify a different template. However, when run with the Applet Runtime, there is only one title screen specified with the CORVID screen commands, and it would be used each time the TITLE command was used, regardless of "servlet=" parameter. If you wish to have a system that will run the same with either applet or servlet runtime, only use a single TITLE command.

The DISPLAY_HTML command can also be used to display an HTML tit le or other informational screen. Unlike the TITLE or RESULTS command, screens displayed with the DISPLAY_HTML command are not parsed for CORVID commands. The screen is just displayed. This is to allow compatibility between the applet and servlet runtim es. In the applet runtime a DISPLAY_HTML command simply opens the specified page in a new browser window. In the servlet version, the page is simply displayed in the user's browser. However, to allow the displayed page to return to the CORVID servlet, there must be a form and action command. If the page has no form, the CORVID servlet will automatically add an OK button at the bottom of the page. This will have the link to return to the servlet.

If placing the OK button at the bottom of the page is not desired, the page MUST have a form in the page. If a <FORM> tag is found, CORVID will not add it's own OK button. To add your own form, the displayed page should have:

```
<form method="post" action="CORVID_SERVLET">
      content
      <input type="submit" name="OK" value="OK">
</form>
```

The "value="OK" sets the label for the button and can be any name. It is also possible to use an image map or link to return. (See the section on image maps for details.) In that case, include a <FORM> and </FORM> tag, but no content in the form. If this is done, CORVID will not add it's own OK button.

Remember that if a form is added to an informational page, it will not work correctly when run with the applet, so for most cases it is better to allow CORVID to add the OK button when using the servlet. This allows the same page to be used with both applet and servlet runtimes.

6.4 Links to Other Pages

Any of the screens displayed by the CORVID servlet can include links to other HTML pages. However, only the page displayed by the CORVID Servlet Runtime will have the embedded information needed to return to the servlet engine. If a link takes the user to another page using the same browser window, they will need to use their Back button on their browser to return to the page displayed by the servlet to continue the session. Consequently, it is best to have links open a new page in a separate browser window. This will allow the links to be viewed, but keeps the page that will continue the session available.

It is easy to have a link open the page in a new window by using:

```
target="_blank"
```

in the link. For example:

```
<a href="page_to_display.html"
target="_blank">link text</a>
```

6.5 JavaScript, XML, Etc.

Any of the templates that are used by CORVID can contain JavaScrip t, XML or any other code that is supported by the browser. The script can include any of the replaceable CORVID parameters. It can appear in CORVID_REPEAT or other blocks. CORVID only processes the specific CORVID commands, such as CORVID_ASK, CORVID_IF, CORVID_REPEAT and the replaceable parameters. It does not parse or "understand" any of the code that may be between these commands. It simply echoes it back out, with any replacements. To CORVID, it does not matter if the code between commands is HTML , XML, JavaScript or any other code that may appear in the future.

The use of JavaScript in the templates allows some very complex effects. Likewise the ability to integrate XML with the replaceable parameters allows system results and data to be conver ted to, and passed as, XML.

Remember that the processing is being done on your server, so do not include any server-side script that could cause security problems.

6.6 Final Screen

A screen that will be displayed at the end of the run can be specified in the Properties window, Servlet tab. This provides a simple way to display results or other information. In most cases, it is better to do this in the Results screen, but if there are several ways to terminate a session, the overall Final Screen may be desirable. This can be a simple "Thank you for running …" screen with no link back into the system, or one that includes a RESTART button to start a new session.

The screen can use any of the CORVID commands that are supported in templates. Replaceable parameters can be assigned. This screen can be used as a results screen, but if that is done, the system will not have a result screen when run with the applet runtime. If you wish to run the system in both modes, always use the RESULT command that is supported in both applet and servlet environments.

6.6.1 Default Final Screen Template

If there is no final screen template specified, a system default template will be used. This is a generic template that displays all the variables in the system. This file is automatically installed by the war file that installs the CORVID servlet. It should not be modified or deleted since it is the last option if another Final Screen is not specified. In most systems, this screen would not be used once the sys tem is fielded, but can be useful for development.

To have the system not display the Final Screen, end the controlling command block with a RESULTS or other command that displays a screen which does not allow the session to continuing processing. This is typically a screen with only a RESTART button or one with no form, and only a link to another part of your web site.

6.7 Using Report Commands when Asking Questions

Any of the report commands can be included in templates which are used to ask questions. For example, to display all the variables that currently have a value on a screen that asks a question use:

```
The data in the system so far is:<BR>
<!-- FOR_EACH VARIABLES  -->
      [[.FULL]]<BR>
<!-- EACH_END -->
<BR><BR>
Now tell me:<BR>
<!-- CORVID_ASK STATIC_LIST    -->
      .....
<!-- ASK_END -->
```

This can be a way to provide the user with intermediate feedback.