



Building Dynamic Web Content Based on Visitor Requirements

WINK (What I Need to Know)™ Virtual Concierge

Web sites are wonderful resources, but visitors to complex sites often have difficulty finding the specific information they need. This is especially true for companies that have a wide range of products and services to choose from, with visitors that have diverse experience and background levels.



A site with appropriate content for novices may be boring for experienced or repeat customers. A site aimed at top-level knowledgeable clientele would confuse or overwhelm novices. Site visitors can become confused and frustrated drilling down through many layers of information to find the answers they need. WINK provides an effective way to provide a dynamically constructed page with content customized for individual visitors based on logical rules.

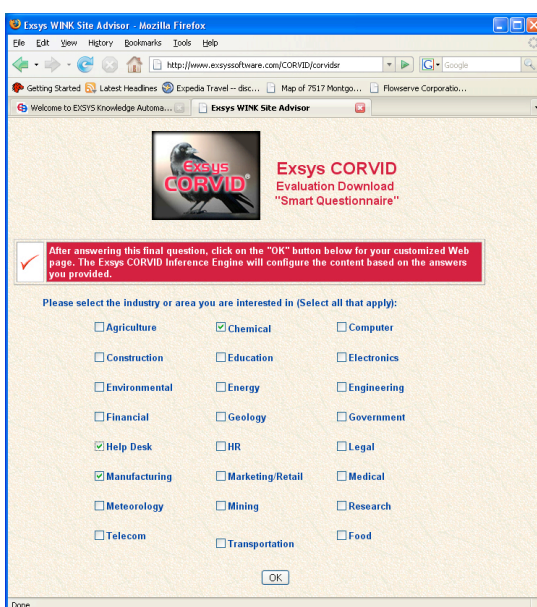
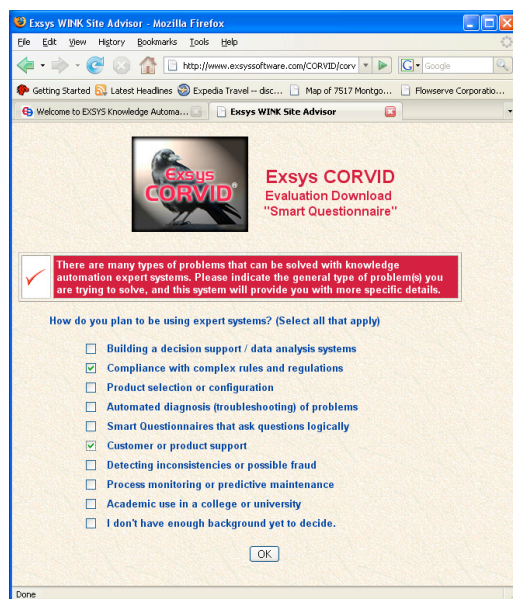
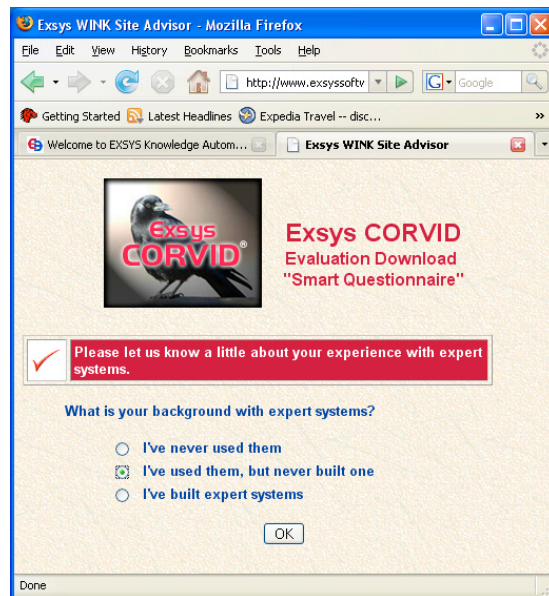
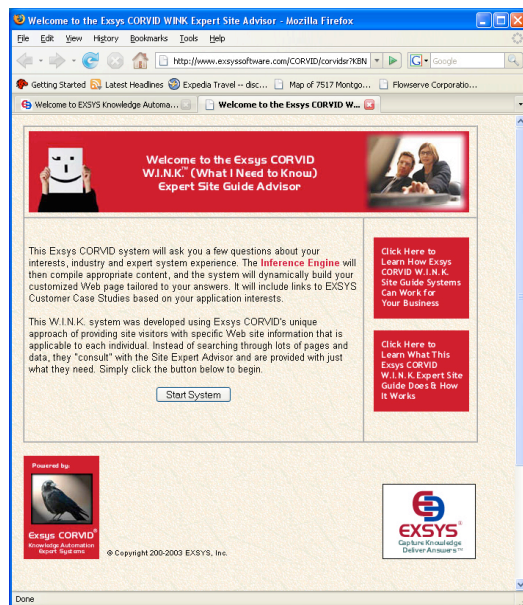
WINK systems are a new approach to dynamically building custom Web pages for each site visitor based on their specific interests and requirements. The WINK page presents the content that will be most relevant to that visitor, generally with links to other parts of the site for more details or to take action on purchases. WINK systems are ideal for large web sites aimed at visitors with very different interests, backgrounds and goals. A WINK system serves as a "virtual concierge" that asks the visitor some simple questions. Then using the logic of the Corvid knowledge automation system, selects the content that is most appropriate for that visitor and builds a custom page for them. This immediately provides the content that the visitor wants, without them having to navigate a potentially confusing site and not finding the information that they need. WINK provides a way to immediately provide recommendations and advice comparable to the interaction with a concierge or top-level representative in a brick-and-mortar store.

WINK systems use Exsys Corvid's powerful Inference Engine to provide Web-based expert "consultations" that ask questions and determine what content is relevant to a visitor. A custom Web page is compiled that includes all of the applicable information specific to their interests and requirements, based on logical criteria built into the system.

WINK is ideal for web sites for companies that support customers in many different categories, or any large site that has problems with visitors getting "lost" without finding the content that they is relevant to them. WINK saves time, and provides compelling content without visitors having to browse through information they don't need. The more complex a site, the more it needs WINK to organize and present information.

For example, consider Exsys Inc. We have customers with widely different backgrounds in expert systems – they want to build systems to solve a large variety of problems in many different industries. The Exsys web site has content for all the different customers. However, it is easy for someone that wants to build a simple diagnostic system, to navigate into a section of the web site that deals with complex product selection problems and think that Corvid does not do diagnostic systems. This can result in a confused customer that leaves the site. Instead, a WINK system asks the visitor a few questions about their interests and experience, and builds a custom page with content relevant to their interests and case studies that illustrate how Exsys tools have been used to solve their type of problem. WINK makes it possible in a few questions to provide information that might take considerable site navigation to find – and which the visitor might miss completely.

With WINK the system just asks a few questions:



When Web site visitors run WINK systems, the Corvid interface asks focused questions and provides an interaction that is "conversational", rather than search-based. If the visitor input indicates more detail is needed in order to select the best page content, the system will automatically ask more detailed questions to determine what additional information to provide. The interaction emulates talking to a knowledgeable expert who can determine what is needed and then provides it. It is like the difference between talking to the doctor and reading the medical textbook. Visitors get the information they need, and are confident that all relevant information has been provided.

Corvid WINK is also ideal for wireless web devices. Corvid allows building interfaces that can be very small and efficient. CSS can be used to format the content for multiple devices.

The goal of a WINK system is to build a customized HTML page that includes the content that is most appropriate for a specific site visitor.

- Creating a general layout and design for the page that will be created.
- Writing rules to determine the pieces of content to be provided based on the visitor's input.
- Incorporating the selected content into the page design using HTML, templates and special Corvid commands.
- Displaying the page to the site visitor.



Thank you for remailing the Exsys CORVID WINN Site Guide. Based on the answers you provided, this system has selected specific information on Exsys CORVID related to your industry, interests and background. The results shown below will provide you with details on our facilities, information on our products and services. Exsys CORVID software is used by many different types of applications, like yours or part of bookmark this page for future reference. If you have any additional questions or would like to contact us online, please [click here](#).




Your Exsys CORVID Knowledge Automation Expert System Information

Since you have used expert systems, you are probably aware of how they can assist in the acquisition and dissemination of needed knowledge and expertise. Exsys CORVID experts built around the EXSYS® software are as common use every day by large and small companies around the world.

Rule-based expert systems are simply a way to systematically describe the steps of a decision-making process so that they can be implemented on a computer. An expert system is based on "rules" which are the individual small steps in a decision-making process. These rules are basically the same as **If/Then** statements you would use to explain how or why a decision was made. The second part of an expert system is the **Inference Engine**. This is a program that uses the rules to determine what information is needed to reach a decision, if that information can be derived from other rules and lower level information, when and how to draw data from the user and how to probabilistically rank the possible decision options in order of likelihood.

Building an expert system is a process of stating the rules that should be used to make the decision. This is often easier than it seems. The inference engine will automatically find those relationships, and use them when the system first runs, to combine the rules in the most effective manner. Rules can be defined in almost any **WinTcl** statement or in **win-structured logic diagrams** that help organize the decision making process.



Our newest product, **Exsys CORVID™**, comes from a fresh re-examination of what is needed to build expert systems in today's Internet-oriented world. CORVID is a whole new approach to expert system development based on almost 20 years of working with system developers. CORVID's development interface is based on a new concept for building rules and trees using **EXSYS Logic-Block structure**, to organize the logical elements of a system.

Command Blocks provide a visual way to see how the procedural flow of the system will proceed. A new screen design allows Web pages and links to be easily incorporated into the web interface. CORVID provides system delivery via a small Java applet or, for more complex operations, the **Serial Router**.

The serial router is unique because it very easy to add a system to its Web page just by adding a few lines of code that CORVID produces. The system is designed to be added to existing rather than the server, so systems are highly scalable regardless of number of users. The serial router allows the interface to be built in HTML, using templates screens. Simply adding 2 templates is enough to run most systems with the Serial Router!

Just take a look at the **CORVID 30 Day Evaluation Program** and **"Quick Start"** tutorial. You will be building your test small system in a few hours. You will be emailed links to download these programs.

Your Exsys CORVID Knowledge Automation Expert System Industry Information

EXSYS customers are located worldwide - with thousands of systems built and deployed. Chances are well above odds that you know someone who uses expert systems. Industries include almost all. Based on 25 years of customer experience, we'll be happy to help you find the best application for your knowledge automation expert system projects. Industries include:

- Aerospace/Astronautics
- Architectural
- Biochemical/Biotechnology
- Business
- Chemical
- Civil Engineering
- Computer
- CRM
- Defense
- E-commerce
- Energy
- Environmental
- Financial Services
- Food & Beverage
- Healthcare
- Hospitality
- Human Resources
- Industrial
- International Trade
- Legal
- Manufacturing
- Marketing
- Sales & Marketing
- Security
- Telecommunications
- Transportation & Traveling
- Web Development
- Writing
- Utilities

Regulatory Compliance
Retail
Sales & Customer Service
Sales & Marketing
Security
Telecommunications
Transportation & Traveling
Web Development
Writing
Utilities

Exsys Customer Case Studies Based On Your Industry and Interests (Click red buttons for details)

[**Mail Sorting Machine Diagnostics and Repair - United States Postal Service**](#)

[**Regulatory Requirements for Confined Spaces - US Dept of Labor/OSHA**](#)

[**Outfitting Manufacturing Knowledge - Texas Eastman**](#)

[**Computer Hardware and Network Configuration - Hewlett Packard**](#)

[**National Fire Code Advisor - Dynalene Corporation**](#)

[**Material and Process Design Advisors - Rockwell International**](#)

[**Asbestos Advisor - US Dept of Labor / OSHA**](#)

[**Electronic Load Assistant - State of Georgia Dept of Industrial Systems**](#)

Here's Information on our Application Area - You're Interested In Remailing

REGULATORY COMPLIANCE - Every company is subject to a wide range of regulatory requirements ranging from Federal, state and local government requirements to internal corporate policies. The regulations are often extensive, complex and difficult to understand. Just providing copies of the regulations to employees is not enough.

EXSYS CORVID Knowledge Automation Regulatory Compliance Systems are one of the best uses of expert system technology. The regulatory information is typically well documented and defined in manuals, but big legal books that are hard to acquire, page long and deliver the information. An expert system allows all the subtleties, nuances and special cases of the regulations to be delivered automatically.

Employees only need to answer simple questions to obtain the detailed compliance information they need. When regulations change, a massive education program is not required - instead changes can rapidly be made in the expert system and everyone gets the most current answers available.

Whether distributed "stand-alone" or online, compliance expert systems help their employees up-to-date in meeting regulatory requirements, and are especially useful in cases where non-compliance can carry heavy fines. Exsys CORVID software is in daily use by hundreds of companies in systems for **Financial Transactions, Environmental Regulations, Judicial Decisions, Quality Control, and OSHA standards**.

Click here to run the Environmental Compliance Self-Assessment System. This system was built by one of the largest food manufacturing companies in the world to meet environmental regulations as outlined by **ISO 14001**.

TECHNICAL SUPPORT - According to a recent **Foresight Research** survey, over 88% of companies consider self-service technology important to their overall customer service strategy. Expert systems are the most effective way to put a virtual support person on your Web site that can effectively deal with questions in a way that keep your customers happy and reduce your support load.

Technical and customer support is a major expense for most companies. The more successful a product, the more technical support staff is needed. Support staff costs money. They understand all the special cases and how to answer virtually every question. That means support help desk takes 2 times to get to a support question answered. There is 3 phone calls, 3 discussions about the problem, and 2 times the customer received the wrong answer. This is typically due to the fact that support staff don't have fully experienced. They allow their best effort, but they just are not experienced enough to reliably know how to answer certain questions.

Exsys CORVID Knowledge Automation Systems provide a very effective alternative. They allow the expertise of the best and most experienced staff to be automatically delivered in a self-service mode. So these top members of the support staff can answer virtually anything, but are not available every call. Employees and business partners can find the answers they need without human assistance. [Click here to run an online Technical Support System](#).

Typically there are a relatively small number of questions that occur frequently. These are ideal for an expert system. The most interesting technical support staff know which items apply to them, and can come forward to the customer and not an ineffective way to answer questions with any degree of complexity. Expert systems provide a way to interact with each customer and gives them the answers they need. The next time they call, they will be able to solve the problem themselves, and makes a huge recommendation. It is like making your last best support staff available at the level of support staff. Being them to work in one unique place cannot be put on the expert system.

FAD files are usually a first attempt on-line support, but they only provide pages of information to the customer so far through it. It may lead to the customer to decide which item applies to them, and then come forward to the customer and not an effective way to answer questions with any degree of complexity. Expert systems provide a way to interact with each customer and gives them the answers they need. The next time they call, they will be able to solve the problem themselves, and makes a huge recommendation. It is like making your last best support staff available at the level of support staff. Being them to work in one unique place cannot be put on the expert system.

[Return to the EXSYS Web Site](http://www.exsys.com)

First, consider what content and information is to be provided to the visitor.

- If the site covers many different products/capabilities/markets, the page generated should only show information on the ones the visitor is interested in. In some cases it may be possible to determine which are appropriate with a few questions and a simple tree diagram. Others may require more complex probabilistic logic to find the ones that should be displayed.
- If the site is technical and site visitors may have varying levels of experience and background, content for different appropriate technical levels should be provided. This can also apply to presenting content in various languages.
- Typically WINK only builds a single page, so try to keep the content relatively short, using links to other pages for more details. The goal of WINK is to select what is relevant and present it in an organized way. Often this requires only a brief description with links for more details. Once the visitor is interested and has links to the relevant parts of the site, they can find the details there.
- Some information and links may be generic and would be provided to all visitors. This is normally made part of the page design rather than being handled in the logic that selects content.

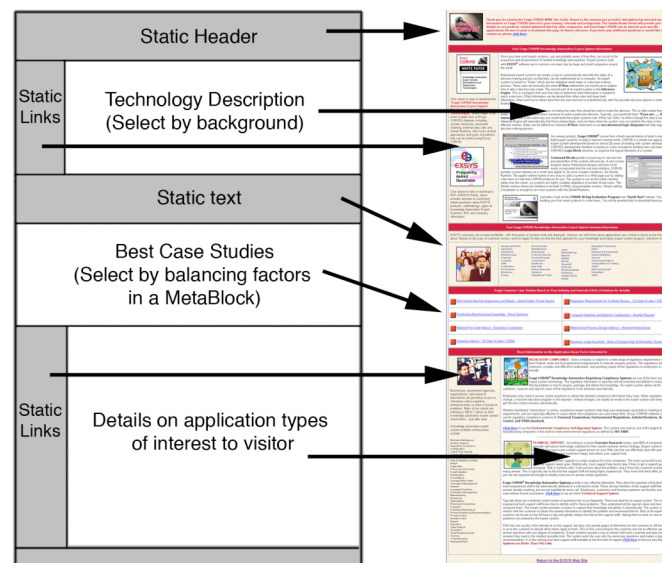
General Design

Create a general design for the page that will be generated. This can simply be sketched out on paper to indicate the overall appearance. This general layout can be mostly fixed or quite dynamic, depending on the complexity of the WINK system. Typically there will be various regions on the page, which in the final design may be cells in a table.

- Regions may be static, with the same content included for all visitors.
- Regions may indicate an area of the page that will always include content on the same general subject, though the specific details may vary with each visitor. For example, a block of a text may always contain a description of company capability and experience, though the specific text might vary to reflect the visitor's requirements.
- Some regions may be highly dynamic with the entire layout of that section changing significantly and set within the WINK system.

For a first system, it is best to limit the design to the first 2 items since this will be easier. The third, allows considerably more dynamic pages, but is actually just a matter of applying the first 2 items recursively to build a section of the page.

For the Exsys page shown earlier, the general design is:



The gray areas are static and have the same content for all visitors. The 3 white regions are the content that the WINK system will build. Each is independent of the others and will be constructed based on the visitor input and system logic.

For more complex systems, some of the content regions might have different layouts for specific situations. In that case, layout designs can be created for the various possibilities for that region.

Templates

The page that will be built by WINK is a combination of the content selected for that visitor, presented in a layout design created in HTML. To the extent possible, the WINK system should try to separate content and layout – though some overlap can occur.

The dynamic content for each of the non-static regions will be built in a Corvid Collection variable. These collection variables can be embedded into an HTML template using double square bracket replacement `[[]]`. This allows a template to be easily designed using standard HTML design tools and WYSIWYG editors.

The template should layout the regions, typically using tables. The page should match the look-and-feel desired for the page, or designed to match other site content. Cascading Style Sheets (CSS) can be used to match site standards and layout, and provide options for how the page will be displayed on various devices.

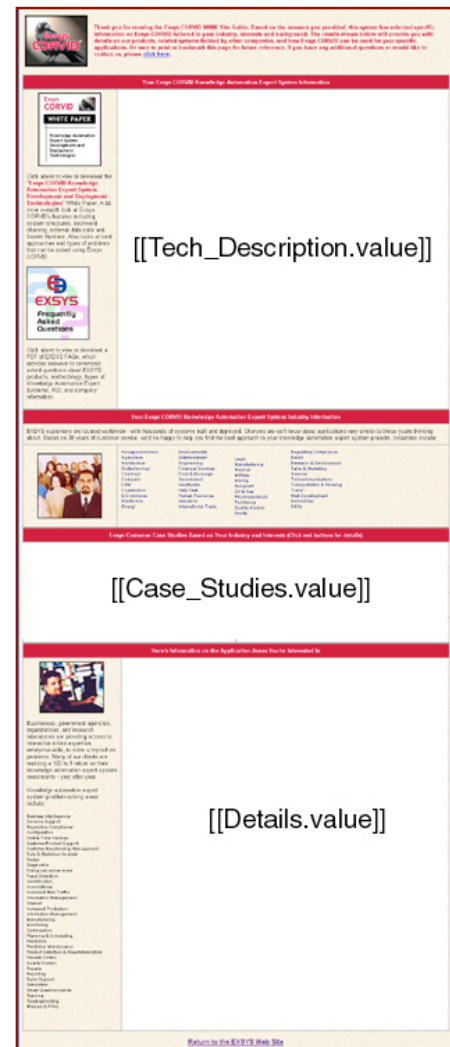
The static text regions can be designed like content on any other HTML page with text, images, links or any other content supported in a browser. However the individual content regions that will change should just be an embedded Collection variable.

In the example the entire region marked “Technology Description” could be just an embedded variable named `[Tech_Description]`. This single variable would be embedded in the template in place of the content for the technology description. It can be added to the template as text using an HTML editor. The variable name should be placed in double square brackets, which will later cause it to be replaced by the content set by the logic in the WINK system. The “.value” property is used since only the value that will be set should be embedded – not the normal Prompt plus Value.

The same can be done for other content regions.

The template to the right has 3 collection variables embedded to provide the dynamic content that will be set by the WINK system. The other parts of the template have content that is static and will be presented to all visitors. This includes headings, logos, and other content that is not visitor-specific.

This provides a quick way to create a template that WINK can work with. The next step is to create the content to be used to replace the variables.



Content

When the result page is displayed, Corvid will replace the embedded collection variables with their values. Since collection variables were used, the value is a text string. The logic in the WINK system can add one or more text strings to the collection, depending on the complexity of the content.

For the first variable, `[Tech_Description]`, the logic is very simple. The content will be one of several descriptions that are at different levels of technical detail. The appropriate one will be selected based on asking the visitor their level of background with the technology. Here one rule will set the full content for the variable.

The [Case_Studies] variable will have much more complex logic, using a MetaBlock to weight and select among many options.

The [Details] variable can cover multiple areas, and will have multiple blocks of text added to the same variable by different rules.

Adding Content to a Collection Variable

There are many methods to add content to a Collection variable, but for WINK systems 2 ways are the most common.

- **Adding content directly**

This uses the .ADD method in the THEN part of a rule:

```
[var.ADD] content
```

With this method the “content” is the text and HTML codes to be included in the content. This approach requires “hard coding” the text and HTML codes into the system. If the text is likely to change frequently, this approach will make it somewhat more difficult to edit the text since it is part of the logic and rules in the system.

- **Adding content from a file**

Corvid allows adding a section of text from a file using the .ADDFILE method.

```
[var.ADDFILE] file, key
```

With this approach, a section of another HTML file can be included. The “key” is a marker in the file that indicates the start and end of the text to include. Since it is very useful for WINK systems to be able to include a section of another HTML page, the “key” markers that must be placed in the file are HTML comments, allowing them to be included but not displayed. The syntax is:

```
text
<!-- CORVID_KEY=key -->
text to add
<!-- CORVID_KEY_END=key -->
text
```

All text between the marker <!-- CORVID_KEY=key_str --> and the closing <!-- CORVID_KEY_END=key_str --> will be added to the file. Since the start and end markers are associated with a specific key, they can overlap, and can also include CORVID_IF conditional inclusion sections. If no “key” is specified, the entire contents of a file will be added to the Collection variable.

Adding content from a file is a very convenient way to select content in a way that:

- Allows it to be built with HTML design tools
- Can be viewed in a browser to see how the HTML appears
- Is edited and maintained separate from the logic of the WINK system

In this case, there could be a new HTML page created called “Description.html” (or any other name). It would have the various descriptions that the system would select among to use as the [Tech_Description] content. These can have relatively simple formatting since they will in the end be dropped into the overall design template.

However, if more complex formatting is needed, it can be used. CSS can be used to format the content of pages being automatically generated. Once the various descriptions are created, open the page source and put the

```
<!-- CORVID_KEY=key
<!-- CORVID_KEY_END=key -->
```

markers in the file. These should be placed so that the content between them can be dropped into the template in place of the "[[Tech_Description.value]]" embedded variable. The text from the new page between the "CORVID_KEY" markers can be copied and pasted into a copy of the template to test this.

Use "key" text that will make it easy to use the appropriate section of the page.

Now add rules in the system that adds the appropriate block of HTML code from the new page based on the input from the site visitor.

```
IF
    Visitor experience with expert systems is limited
THEN
    [Tech_Description.ADDFILE] Description.html, basic

IF
    Visitor experience with expert systems is extensive
THEN
    [Tech_Description.ADDFILE] Description.html, advanced
```

In the Description.html page, there would be sections:

```
Optional text
<!-- CORVID_KEY=basic -->
    Basic description of how systems work
<!-- CORVID_KEY_END=basic -->
Optional text
<!-- CORVID_KEY=advanced -->
    Advanced description of how systems work
<!-- CORVID_KEY_END=advanced -->
Optional text
```

The actual descriptions can involve as much text, HTML code and other content as needed – provided it can be dropped into the template in place of "[[Tech_Description.value]]"

Once the system is finished, if there needs to be a change in the descriptions, it can be edited in the Description.html file, without having to edit the actual WINK logic or content.

Adding Multiple Items of Content

To add multiple items of content to the variable, as in the [Details] section, just add content more than once. The [Details] section may have one or many descriptions based on items in a list that the site visitor selected. Collection variables can have any number of items added to them.

A series of rules could add content to the Collection variable. A Details.html page could be created with the descriptions of the various areas that could be included. These would be marked using the "CORVID_KEY" markers. The rules could then add the appropriate pieces to the collection variable. Some of these might be:

```
IF
    Area of interest is product selection
THEN
    [Details.ADDFILE] Details.html, Product_description

IF
    Area of interest is diagnostics
THEN
    [Details.ADDFILE] Details.html, diagnostics
```

These rules would be run from a Logic Block in forward chaining. One or more rules could fire, adding content to the [Details] collection. The content would come from the separate HTML page(s) with the various possible descriptions, and "CORVID_KEY" tag markers.

When selecting the sections of the Details.html page to add, make sure that the total HTML code for all the sections can be concatenated and embedded in place of the "[[Details.value]]" in the template.

Building Content from Another Template

Another very useful way to build more complex content to put into the main template is to use other templates to build sections. This can be a file, or a section of a HTML page indicated with "CORVID_KEY" markers.

As with the main template, this can include embedded Corvid variables in double square brackets. This second level template can include formatting, embedded variables and other content.

For example, suppose a system will recommend the best products for the customer. There are enough products that there will always be 4 to recommend, but the system should display the top 4 items in a table.

To select the products, a MetaBlock will be used to sort the top products into the Collection variable [Products]. (See the MetaBlock section of the manual for details on product selections systems).

Create a small HTML page with just the table that you want to use. This can have cells with the various items from the selected Products list.

[[Product.ITEM 1]]	[[Product.ITEM 2]]
[[Product.ITEM 3]]	[[Product.ITEM 4]]

Notice that these are each double square bracket expressions. When the template content is added to another Corvid variable, these double square bracket expressions will automatically be evaluated when the content is added.

Looking at the code for the page in the file Product.html:

```
<html>

  <head>
    <title>Product Template</title>
  </head>

  <body bgcolor="#ffffff">
    <!-- CORVID_KEY=Product_table -->
      <table border="1" cellspacing="2" cellpadding="0">
        <tr>
          <td>[[Product.ITEM 1]]</td>
          <td>[[Product.ITEM 2]]</td>
        </tr>
        <tr>
          <td>[[Product.ITEM 3]]</td>
          <td>[[Product.ITEM 4]]</td>
        </tr>
      </table>
    <!-- CORVID_KEY_END=Product_table -->

  </body>

</html>
```

A Corvid variable [Product_table] would get the content for the full table, with all double square bracket embeds replaced with:

```
[Product_table.ADDFILE] Product.html, Product_table
```

The Corvid variable [Product_table] would now have the formatted content for the recommended products, which could then be embedded into the top-level template.

Using templates for specific sections that are then embedded into a higher-level template can be repeated recursively as many levels as is needed for a layout.

It can also be used as a way to select details of the layout. In this example, 4 products are displayed in a 2x2 table. If only 3 products were suitable, a separate ADDFILE command could call a different table layout for the top 3 products and use that as the value for [Product_table]. This could then simply be embedded into the top-level template to change that detail of the layout.

Embedding the Values of Variables

In addition to the main Collection variables embedded in the templates, any of the text added to the collections either directly or from files can include the embedded values of other variables. This is a convenient way to include user input, values calculated by the system or short text strings.

To do this, just put the name of the variable in double square brackets in the text that will be added to the collection variable. In many cases it will be useful to use the properties of the variable to format it.

For example, if the visitor has specified a spending limit which is stored in the variable [Limit], part of the content added to a collection variable could be:

Since the spending limit is \$[[Limit.format ###]], the products recommended are ...

Any of the properties for variables can be used when they are embedded, though the .VALUE and .FORMAT are most commonly used.

When a variable is embedded, Corvid will attempt to derive the value for the variable, this can lead to backward chaining and the system asking questions to attempt to derive the value. If it is certain that the value will be fully known when the embedded variable is used, Corvid can be told to not attempt to derive a value, but use the current value. This is done by putting an * between the [[and the variable name. For example:

[[*price.value]]

would embed the current value of [Price]] without trying to derive it.

Using CORVID_IF

In addition to all the other ways to select and conditionally include text, CORVID_IF commands can be used in the text added to the Collection variables from a file or template.

The syntax is:

#CORVID_IF (expression)

lines of text to add

#CORVID_ENDIF

the expression can be any Boolean expression, including ones with Corvid variables. Any expression that could appear in an IF node of a Logic Block can be used. See the manual section on CORVID_IF for the details of using this option.

For example if the system needed to select between 2 table forms to display either 3 or 4 products:

#CORVID_IF ([Num_products_to_display] >= 4)

```
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>[[Product.ITEM 1]]</td>
    <td>[[Product.ITEM 2]]</td>
  </tr>
  <tr>
    <td>[[Product.ITEM 3]]</td>
    <td>[[Product.ITEM 4]]</td>
  </tr>
</table>
```

#CORVID_ENDIF

#CORVID_IF ([Num_products_to_display] < 4)

```
<table border="1" cellpadding="2" cellspacing="2">
  <tr><td>[[Product.ITEM 1]]</td></tr>
  <tr><td>[[Product.ITEM 2]]</td></tr>
  <tr><td>[[Product.ITEM 3]]</td></tr>
</table>
```

#CORVID_ENDIF

Depending on the value of [Num_products_to_display] one of the tables would be selected. Notice that these include other embedded values that would be replaced by the values set by the system.

Displaying the Finished Page

Now that there is a top-level template (TopTemplate.html) and all of the Collection variables embedded into that page have values, the last step is to display it.

Before displaying the final WINK page, make sure all the embedded variables have their final value. This can be done from the Command Block. It is a good idea to have a Logic or Action Block to derive the value for each embedded variable. These can be run directly from the Command Block with the FORWARD command. (The DERIVE command can also be used if backward chaining is used to set the value, but remember that backward chaining uses rules as they are needed, so the order that content will be added may not be as predictable.)

So the first part of the Command Block will look something like:

```
FORWARD BLOCK=Product_Block
FORWARD BLOCK=Detail_Block
FORWARD BLOCK=Description_Block
```

Once all of the top-level template embedded variables have values, the page can be displayed. There are several ways to do this depending on if the system is run with the Corvid Applet or Servlet Runtime, and if the resulting page should be saved and able to be bookmarked for later reference.

The easiest way to display the page is when running with the Servlet Runtime, and without the need for bookmark support. Just specify the TopTemplate.html as the results screen template and do the RESULTS command.

```
RESULTS Servlet= TopTemplate.html
```

The RESULTS command will automatically replace all the embedded variables with their value and display the resulting page. This will have all the content that was generated for the site visitor, but since it is just a page from the Servlet, it does not have a unique URL and cannot be bookmarked. The visitor can view or print the page, but once the session is done, or once they browse to another page the content will be lost. (Because you cannot leave the page and return, any links on the page should open new pages in a separate browser window, leaving the current display window unchanged.)

Display from an Applet or Supporting Bookmarks

The final WINK page can also be displayed from the Applet Runtime, or from the Servlet Runtime in a way that allows bookmarking the page. This is done by setting the value of a final Collection variable [WINK_page] by adding the top level template with ADDFILE:

```
[[WINK_page.ADDFILE] TopTemplate.html
```

Notice that there is no KEY value since you want to include the full content of the entire page.

As above, before adding the content to WINK_Page, make sure all the embedded variables have their final value. So the first part of the Command Block will look something like:

```
FORWARD BLOCK=Product_Block
FORWARD BLOCK=Detail_Block
FORWARD BLOCK=Description_Block
...
SET [[WINK_page.ADDFILE] TopTemplate.html
```

There is now a collection variable [WINK_Page] with the full HTML content that is to be displayed. Java security does not allow this to be done directly. Instead, the content must be saved as a Corvid report and then the report displayed.

Saving the Page

Java security prevents an applet from simply displaying an HTML page, or from writing the page to the local hard disk. The applet can only write the page back to the server, and then call another server program that displays the page in a new browser window.

Call the SaveReport command to save the report to the server. First create a new string variable [Page_ID] to store the page ID on the server. When the page is saved, it will return an identifier that will be stored in [Page_ID] and later used to specify which file to display.

The content that you send to the server for the page is the Collection variable [WINK_Page]. Add the SaveReport command to the Command file after the command that assigned the TopTemplate to [WINK_Page]. The command will be something similar to:

```
SaveReport [Page_ID] [WINK_Page] IP=0 SERVLET="SaveRptServletURL"
```

Where the "SaveReportServlet" URL is the URL for the Corvid save Report Servlet on your server. (See the Corvid manual for more details on the SaveReport command and its various options.)

Note: WINK normally builds an HTML page and the SaveReport/DisplayReport commands assume that the content-type is text/html. It is also possible to build WINK style pages using other content types such as RTF or plain text. In that case, change the "Content-type" to the appropriate type. See the SaveReport part of the Corvid manual for details.

Displaying the Page

Now all that remains is to display the file you created with the SaveReport command in a new browser window.

Use the Corvid DisplayReport command, which includes the String variable [PAGE_ID] that was set in the SaveReport command. This variable has the information that identifies the page on the server to display. The command will be similar to:

```
DisplayReport [PAGE_ID] SERVLET="DispRptServletURL"
```

This will display the page in a new browser window. The site visitor can then print, save or bookmark the page from their browser.

The page will have a unique identifier that allows it to be bookmarked. The SaveReport command has options that determine how long the page will be kept on the server and who can access the page. Access may be limited to the originator only, or allowed to be viewed by anyone. If some limits are applied, the visitor should be warned that the page will be available only for x days, etc. (See the SaveReport section of the Corvid manual for the details and options of the SaveReport command).

Ending the System

The DisplayReport command will display the custom WINK page for the visitor in a new browser window. However, the window running the system should have some simple closing screen to display some information – "thank the user for running the system", etc. This can be done with a standard Results command.

Conclusion

WINK systems are a very powerful way to build content tailored to individual site visitors. Corvid provides many ways to maintain items of content as HTML pages that are parsed and conditionally included into an overall template for the final page. The various ways to dynamically modify content and layout, which can be used recursively, allows highly complex designs to be implemented.