Exsys Corvid® Core for Apple Macintosh
Application How-To

Converting Dichotomous Keys into Expert Systems

Maryland Bats

Dichotomous keys are a standard and widely used way to describe a methodology to identify or select among similar items. They use a series of question each of which has 2 possible answers, typically Yes/No, true/false, present/not present, etc. Each question leads to either an identification or jumping to another question lower down in the question lists.

Dichotomous keys are standard in biology where they are the most common way to document the steps to separate different species, but they are also used in many other areas. They are popular since they somewhat simplify and automate the identification steps by breaking them into specific precise questions. This is helpful, but can be done much better by converting them to a Corvid system.

This system is a demonstration of converting a standard dichotomous key into a Corvid Core expert system. This is very easy to do, since dichotomous keys are fundamentally equivalent to tree diagrams, and can be easily built in Logic Blocks.

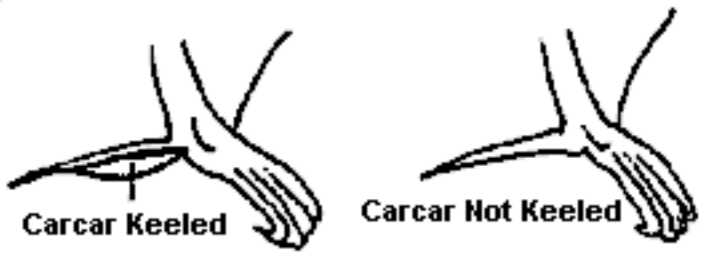
This system is based on a dichotomous key for the identification of bat species in Maryland. This can be found at:

http://www.dnr.state.md.us/wildlife/Plants_Wildlife/bats/bat_key.asp

This key was chosen because it is a typical key and covers an area that most people are not familiar with. Use of this system does not indicate any endorsement of Exsys or Exsys Corvid by the Maryland Dept. of Natural Resources.

Maryland Bat Identification Dichotomous Key		
Dichotomous means having two equal parts. A key is a tool used to identify organisms. A dichotomous key will give you a series of choices between two related characteristics. Choices based on these two characteristics that best fit the organism in question will take you on different paths to ultimately find the name of the organism. Choices typically begin with broad characteristics and become narrower as more choices are required. Begin with number 1 (a).		
1(a).	If dorsal (upper) surface of tail membrane furred	Go to 2
1(b).	If dorsal (upper) surface of tail membrane naked or sparsely furred	Go to 4
2 (a).	If fur black, many hairs silver-tipped	Silver-haired Bat (Lasionycteris noctivagans)
2(b).	If fur never uniformly black, hairs silver-tipped or not	Go to 3

3(a).	If ears conspicuously black edged with patches of yellowish hair scattered inside; fur yellowish to dark brown, distinctly frosted; tan throat patch	Hoary Bat (Lasiuruscinereus)
3(b).	If ears not conspicuously black-edged and bare; fur red-orange (male) or yellowish-brown (female), tips of hairs often frosted	Eastern Red Bat (Lasiurus borealis)
4(a).	If bats with brown fur only	Go to 5
4(b).	If small bat with yellowish fur and black wings; forearms 31-35mm; fur with three color bands, darkest band/base, lightest band/middle and darker tips	Eastern Pipistrelle Bat (Pipistrellus subflavus)
5(a).	If muzzle naked and black, tragus rounded	Go to 6
5(b).	If muzzle furred; tragus pointed (Genus Myotis)	Go to 7
6(a).	If large bat. Forearm more than 40mm	Big Brown Bat (Eptesicus fuscus)
6(b).	If small bat. Forearm less than 40mm	Evening Bat (Nycticeius humeralis)
7(a).	If ears more than 16mm; when laid forward, ears extend 2mm past end of nose	Northern Long-eared Bat (Myotis septentrionalis)
7 (b).	If ears less than 16mm	Go to 8

		
8(a).	If calcar keeled	Go to 9
8(b).	If calcar not keeled	Go to 10
9(a).	If foot is greater than 8.5mm long; forearm usually greater than 35mm long; hair on toes short and sparse, nose pinkish, no black face mask	<u>Indiana Bat</u> (Myotis sodalis)
9(b).	If foot is less than 8.5mm long; forearm usually less than 35mm long; light colored fur with black face mask and lips	<u>Small-footed Bat</u> (Myotis leibii)
I0(a).	If a few long hairs extend to tips of claws on foot or beyond	Go to 11
I0(b).	If no long hairs extend to tips of claws on foot	<u>Indiana Bat</u> (Myotis sodalis)
11	If fur glossy with sheen; not dense and woolly	<u>Little Brown Bat</u> (Myotis lucifugus)

There are 2 ways to convert dichotomous keys into a Corvid system and both are demonstrated here.

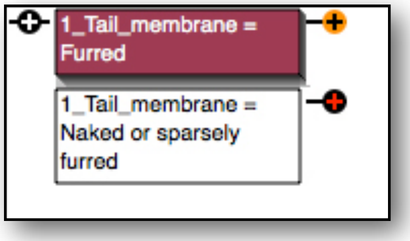
Conversion to a Single Tree

The first approach is to convert the key into its equivalent tree.

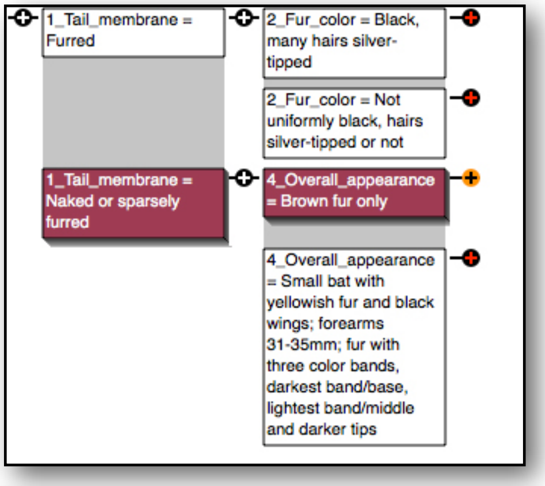
1. Build a List variable for each of the questions in the system, with 2 possible values that match the options for that question in the key. It is easier to keep track of questions, if the name includes the number of the question, though this does not need to be in the prompt. For example, the first question in the system relates to the tail membrane, to it could be named “1_Tail_Membrane”. This makes handling the many “goto” lines in the system easier.

2. Build a Confidence variables for each of the possible identifications. This can include HTML links to pages with more information,

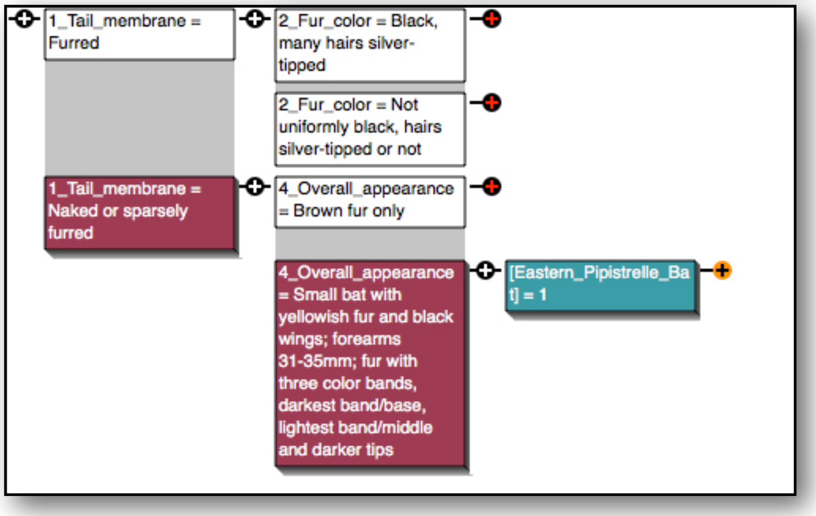
3. Start a Logic Block and add the first question in the key with its 2 values.



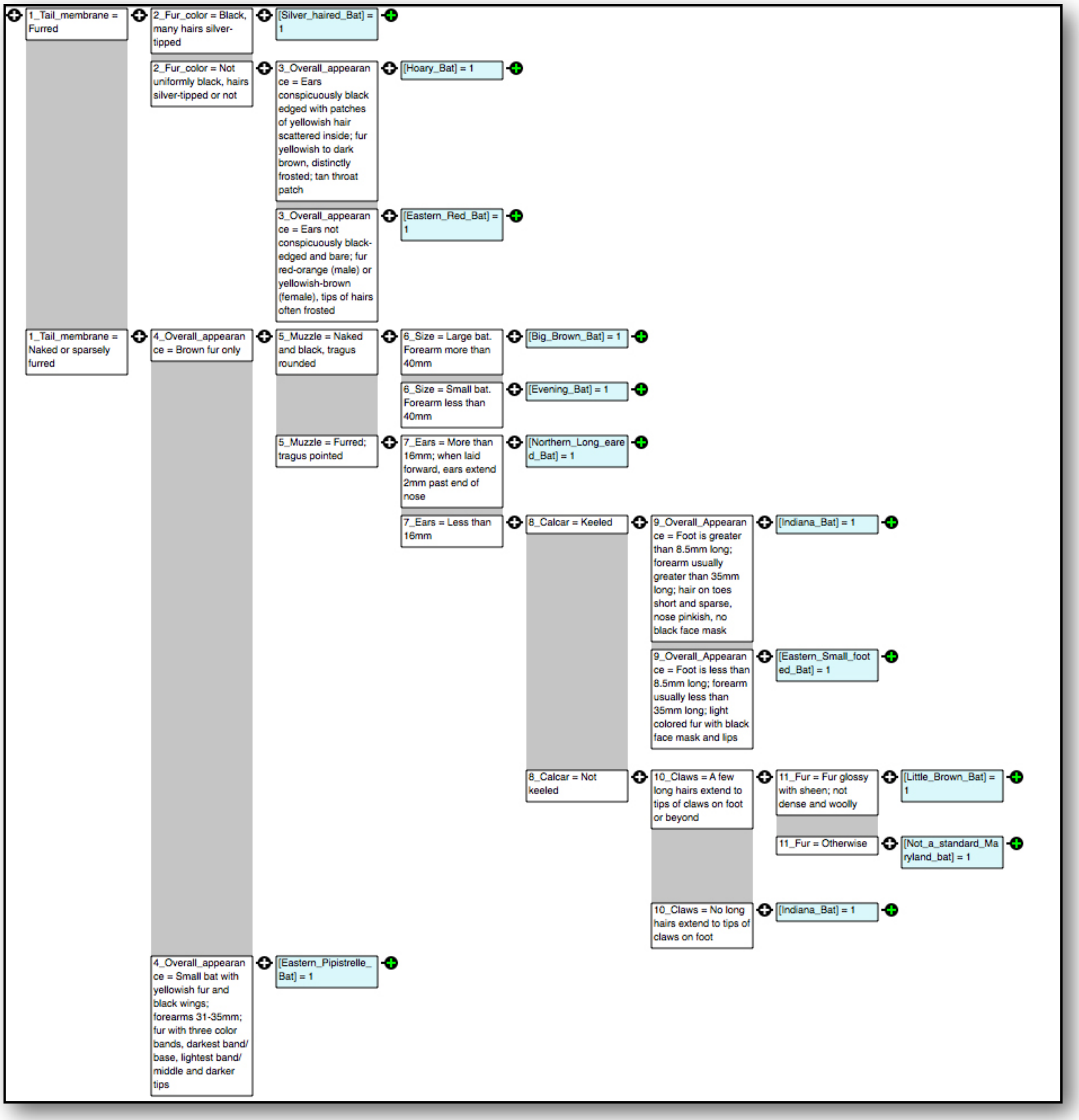
4. For each of the 2 values, follow the key down to find the question that leads to. Here the “furred” value leads to question 2 and “naked” leads to question 4. Add those questions to the associated insert points, each with its 2 values.


1_Tail_membrane = Furred
2_Fur_color = Black, many hairs silver-tipped
2_Fur_color = Not uniformly black, hairs silver-tipped or not
1_Tail_membrane = Naked or sparsely furred
4_Overall_appearance = Brown fur only
4_Overall_appearance = Small bat with yellowish fur and black wings; forearms 31-35mm; fur with three color bands, darkest band/base, lightest band/middle and darker tips

5. Repeat this for each of the new nodes. Using the key line number makes it much easier to replicate the key. When you come to one of the final identifications, add a THEN node assigning the associated confidence variable a value of 1.

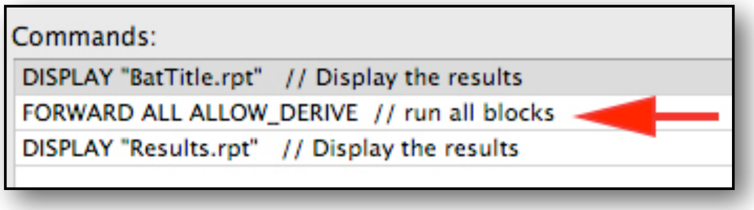

1_Tail_membrane = Furred
2_Fur_color = Black, many hairs silver-tipped
2_Fur_color = Not uniformly black, hairs silver-tipped or not
1_Tail_membrane = Naked or sparsely furred
4_Overall_appearance = Brown fur only
4_Overall_appearance = Small bat with yellowish fur and black wings; forearms 31-35mm; fur with three color bands, darkest band/base, lightest band/middle and darker tips
[Eastern_Pipistrelle_Bat] = 1

6. Continue doing this until the full tree is built out. Since Corvid expands the tree automatically as values are added and the variable numbers match the key, it is easy to quickly build out the tree.



```
1_Tail_membrane = Furred
    2_Fur_color = Black, many hairs silver-tipped
        [Silver_haired_Bat] = 1
    2_Fur_color = Not uniformly black, hairs silver-tipped or not
        3_Overall_appearance = Ears conspicuously black edged with patches of yellowish hair scattered inside; fur yellowish to dark brown, distinctly frosted; tan throat patch
            [Hoary_Bat] = 1
        3_Overall_appearance = Ears not conspicuously black-edged and bare; fur red-orange (male) or yellowish-brown (female), tips of hairs often frosted
            [Eastern_Red_Bat] = 1
1_Tail_membrane = Naked or sparsely furred
    4_Overall_appearance = Brown fur only
        5_Muzzle = Naked and black, tragus rounded
            6_Size = Large bat. Forearm more than 40mm
                [Big_Brown_Bat] = 1
            6_Size = Small bat. Forearm less than 40mm
                [Evening_Bat] = 1
        5_Muzzle = Furred; tragus pointed
            7_Ears = More than 16mm; when laid forward, ears extend 2mm past end of nose
                [Northern_Long_eared_Bat] = 1
            7_Ears = Less than 16mm
                8_Calcar = Keeled
                    9_Overall_Appearance = Foot is greater than 8.5mm long; forearm usually greater than 35mm long; hair on toes short and sparse, nose pinkish, no black face mask
                        [Indiana_Bat] = 1
                    9_Overall_Appearance = Foot is less than 8.5mm long; forearm usually less than 35mm long; light colored fur with black face mask and lips
                        [Eastern_Small_footed_Bat] = 1
                8_Calcar = Not keeled
                    10_Claws = A few long hairs extend to tips of claws on foot or beyond
                        11_Fur = Fur glossy with sheen; not dense and woolly
                            [Little_Brown_Bat] = 1
                        11_Fur = Otherwise
                            [Not_a_standard_Maryland_bat] = 1
                    10_Claws = No long hairs extend to tips of claws on foot
                        [Indiana_Bat] = 1
    4_Overall_appearance = Small bat with yellowish fur and black wings; forearms 31-35mm; fur with three color bands, darkest band/base, lightest band/middle and darker tips
        [Eastern_Pipistrelle_Bat] = 1
```

7. Since this system matches the procedural structure of the key, it can be run with the default forward chaining command FORWARD ALL ALLOW_DERIVE. (However, since confidence variables have been used for the goal, it could also be run with a backward chaining DERIVE CONF command.)



Commands:

```
DISPLAY "BatTitle.rpt"   // Display the results
FORWARD ALL ALLOW_DERIVE  // run all blocks
DISPLAY "Results.rpt"   // Display the results
```

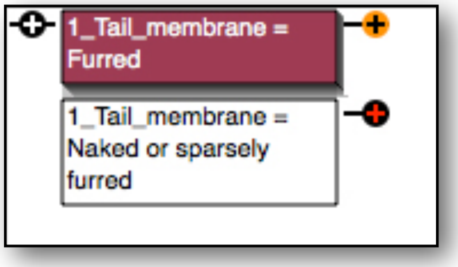
This approach has the advantage that it only requires one Logic Block and it is easy to see the full logic structure - actually much easier than looking at the key. The disadvantage is that the Corvid representation, while equivalent to the key, does not match the exact structure of the key. It requires careful checking that all branches correspond to the key. Changes to the key may require rebuilding most or all of the Logic Blocks.

Also, while there is no limit on the size of Logic Block trees, there are practical limits to navigating in very big trees. Here the key is small enough to convert into a manageable single tree. However, if it had twice as many key questions, the tree would be getting pretty large, and for large keys conversion to a singe tree, while possible, is not the best approach. For these, Corvid has an alternative approach that closely matches the key structure and works for keys of any size.

Conversion to a Multiple Logic Blocks

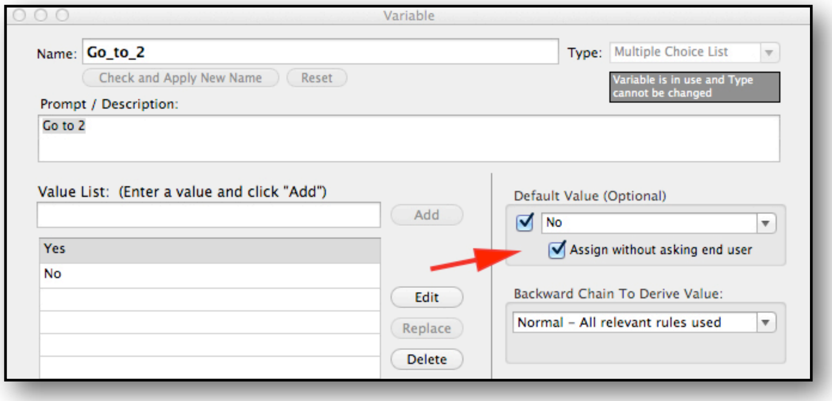
This approach uses the same List and Confidence variables as the single tree approach, but add additional List variables for each of the "Goto" lines in the key and builds a separate Logic Block for each question in the key. This may seem like more work, but it allow the Corvid system to exactly match the key and to handle keys of any size without large tree diagrams.

1. Create a new Logic Block. Since this will only have the logic for the first question in the key, it is good to give it a name related to this, such as "1 Tail membrane". (However since the Logic Blocks are automatically named "Logic Block X", that numbering systems can also be used.) Add the first question with its 2 possible values.



2. In the key the "furred" value leads to a "Go to 2" action. In the single tree we just put in question 2 at this point. Here we will add a THEN that matches the key.

   Create a List variable named "Go To 2" with values of "Yes" and "No". The important step is to set the default value to "No", with the "Do not ask the user" option selected. Repeat this for the second line in the key be creating a "Go to 4" list variable with the same values and options. (You will eventually need similar "Go to X" variables for each question except #1, so build these for each of the questions in the system, this will be values of 3 to 11.)



Variable

Name: Go_to_2

Type: Multiple Choice List

Variable is in use and Type cannot be changed

Prompt / Description: Go to 2

Value List: (Enter a value and click "Add")

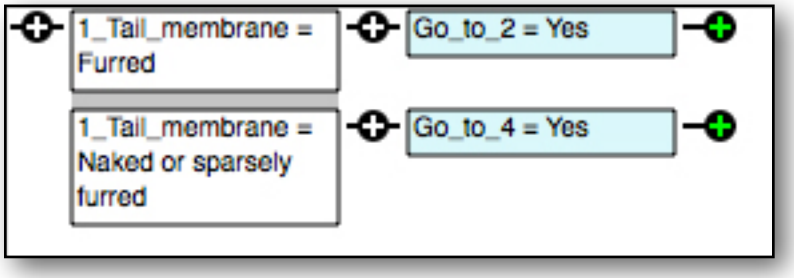
Yes
No

Default Value (Optional): ✓ No

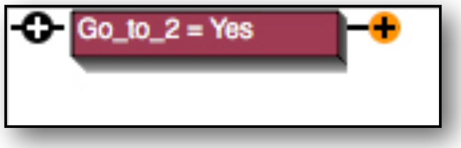
✓ Assign without asking end user

Backward Chain To Derive Value: Normal - All relevant rules used

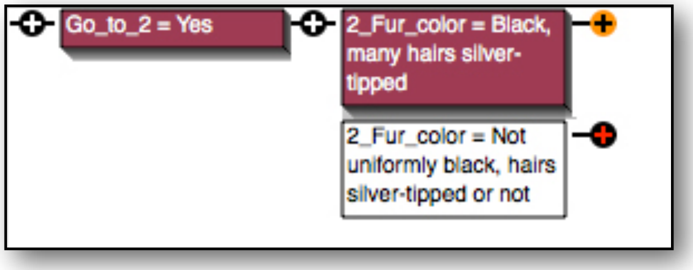
3. Now use these variables to finish the Logic Block so that it matches the lines in the key. Add a THEN assignment of "Go_to_2 = Yes" for the first line and "Go_to_4 = Yes" for the second value. That is all that is need for that Logic Block. It only holds the logic of the first question in the key.



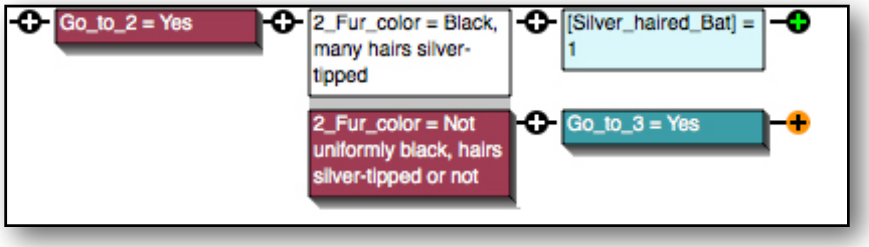
4. Add a new Logic Block for the 2nd question in the key. Start this block with a single IF node that test if the "Go_to_2" variable was set to "Yes". This will only happen if an earlier Logic Block set it to "Yes" since the default that will automatically be assigned is the "No" value. If a proceeding line in the key does not have a "Go to 2", the rules in this block will be skipped.



5. Now add the nodes for the 2nd question.

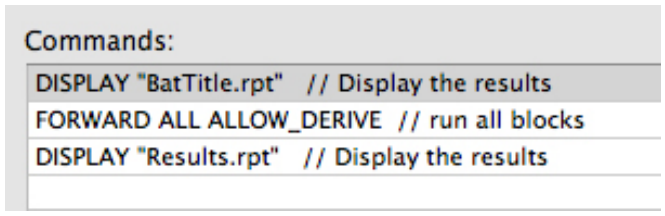

Go_to_2 = Yes
2_Fur_color = Black, many hairs silver-tipped
2_Fur_color = Not uniformly black, hairs silver-tipped or not

6. The first value goes to a specific identification, and the second ends in a "Go to 3". Add these to the Logic Block.


Go_to_2 = Yes
2_Fur_color = Black, many hairs silver-tipped
[Silver_haired_Bat] = 1
2_Fur_color = Not uniformly black, hairs silver-tipped or not
Go_to_3 = Yes

7. Add a new Logic Block to the system for each question in the key. Start each with a single node testing "Go_to_X = YES" and the X questions with its values. Add the corresponding actions from the key. This should be done for each question in the key.

8. Since we want the blocks run in order, the Command Block should use the FORWARD ALL ALLOW_DERIVE command. (Here too, it can be run in backward chaining, but the chaining is complicated by the "Go to" variables. It will work, but this was designed with more of the forward chaining methodology of the key and that is a better approach to use to match the key.



Commands:

```
DISPLAY "BatTitle.rpt"   // Display the results
FORWARD ALL ALLOW_DERIVE   // run all blocks
DISPLAY "Results.rpt"   // Display the results
```

This approach requires some extra variables, but they all have the same structure and are easy to create. It allows the Logic Blocks in the system to exactly match the key and can handle keys of any size and complexity. It is easy to check and validate the system by just comparing each Logic Block to the associated question in the key. The Corvid Inference Engine does all the rest.

Servlet Version

While the system created for the Corvid Applet Runtime will work when run with the Corvid Servlet Runtime, it contains advice to the user that they must make sure their browser has any “popup blocker” turned off before clicking on links to other pages.

Many browsers consider html pages opened from a java applet to be “popups”. However the servlet is using only normal html links with “target=_blank”. Browsers do not consider this to be a popup, so the message is not needed when running with the servlet runtime. The BatID_S system is identical to the BatID system except for deleting the popup warning messages.

EXSYS, Inc.
6565 Americas Parkway. NE
Suite 200
Albuquerque, NM 87110 U.S.A.
Tel: +1.505.563.5987
info@exsys.com
www.exsys.com

EXSYS®
Capture Knowledge
Deliver Answers