Populating a Hierarchical List Box Dynamically

In the examples above, the contents are entered at design time. They are static. Chances are, you need to populate the list box dynamically at run-time, by reading an OpenInsight table.

For this example, we use a simple database called **KNOWLEDGE_BASE** with three columns: **RECORD_NUM**, **PRODUCT, TYPE**, and **TITLE**. Do the following

1. Create this table, with **RECORD_NUM** as the key, and **PRODUCT, TYPE**, and **TITLE** as non-key fields.
2. Create a form to add the following 5 records:

RECORD_NUM	PRODUCT	TYPE	TITLE
1	OpenInsight	Tech Note	Hierarchical List Boxes Part 1
2	OpenInsight	Tech Note	Hierarchical List Boxes Part 2
3	OpenInsight	White Paper	How to Get to the Web with OpenInsight
4	Advanced Revelation	Tech Note	Using the NLM With Advanced Revelation
5	Advanced Revelation	White Paper	Converting Your Advanced Revelation Application to OpenInsight

3. Put a **BTREE** index on **RECORD_NUM**, **PRODUCT**, and **TYPE**.
4. To populate the list box, create a function called **TEST_LBD()**. Copy the following code:

```
subroutine test_lbd (VOID)
declare subroutine btree.extract, set_property, send_message, Msg
equ PRODUCT$ to 1
equ TYPE$ to 2
equ TITLE$ to 3
**Setup a Btree.Extract routine to get all the data in the table**
**This search_string passed to Btree.Extract will get all the records**
search_string = "RECORD_NUM":@VM:">0":@FM
table = "KNOWLEDGE_BASE"
**Open the associated dictionary file for KNOWLEDGE_BASE**
open "DICT.":table TO @DICT else
  Msg(@WINDOW , "Could not open the Dictionary for KNOWLEDGE_BASE")
return
end
keys = '' ;*Keys is what the list of keys is returned to
option = '' ;*No options are specified to send to Btree.Extract
flag = '' ;*No flags are set to send to Btree.Extract
Btree.Extract (search_string, table, @DICT, keys, option, flag)
**Get the number of keys returned by Btree.Extract
iKeyCount = count(keys,@VM) + (len(keys)>0)
convert @VM to @FM in keys
**Open the table so we can go through it to get the information for the list box**
open 'KNOWLEDGE_BASE' to hKNOWLEDGE_BASE else
  Msg(@WINDOW , "Could not open the KNOWLEDGE_BASE table in Populate click event")
  return
end
content = '' ;*This will hold the information from the table to put in the list box.
list = '' ;*This will hold a list of Products to ensure they do not repeat in the list view.
iContPos = 0 ;*Increments the position of the information in the CONTENT variable.
for i = 1 to iKeyCount
read RECORD from hKNOWLEDGE_BASE, keys<i> else
  call msg(@WINDOW , "Couldn’t read from KNOWLEDGE_BASE table in the loop")
end
** This will be true only the first time in the loop **
if i = 1 then
/*List is going to be an @FM delimited list of all the products
that were found. It will be used later to locate products in to
ensure that they only show a specific product once. iContPos is the
variable that holds which position the information will be in, in
the list box */
iContPos = iContPos + 1
content<iContPos> = RECORD<PRODUCT$>
list = RECORD<PRODUCT$>
** getAllTypes will get all the info according to RECORD<DATE$> **
gosub getAllTypes
end else
** Checking to see if the product is in the list of products **
```

```
locate RECORD<PRODUCT$> In list Using @FM Setting POS then
/* This Last Name is already there so don't add it to the
content again */
end else
iContPos = iContPos + 1
content<iContPos> = RECORD<PRODUCT$>
list = list:@FM:RECORD<PRODUCT$>
/* getAllTypes will get all the types according to
  RECORD<PRODUCT$>, which is Last Name in this case */
  gosub getAllTypes
 end
end
next I
/* All of the info has been retreived in the variable 'CONTENT', so now display it */
Set_Property(@WINDOW : '.LB_H', 'LIST', content)
** This code colapses the Hierarchical List Box **
Send_Message(@WINDOW : '.LB_H', 'EXPAND', 0, 0)
******************
getAllTypes:
/* These parameters will be passed to Btree.Extract to get all the TYPES according to the PRODUCT
of the current record
Only get the records that are associated with the product that the record is on */
search_string = "PRODUCT":@VM:"=":Record<PRODUCT$>:@FM
table = "KNOWLEDGE_BASE"
Open "DICT.":table To @DICT Else
Msg(@WINDOW , "Could not open KNOWLEDGE_BASE in the subroutine in the populate click")
return
End
keysSub = ''
optionSub = ''
flagSub = ''
list2 = '' ;*Going to hold the list of TYPES that will be searched to make
;*sure a TYPE doesn't get repeated in the list
/* IF Btree.Extract is successful, the variable keysSub holds the list of keys, @VM delimited */
Btree.Extract(search_string, table, @DICT, keysSub, optionSub, flagSub)
**Get the number of keys returned by Btree.Extract
iKeyCountSub = count(keysSub,@VM) + (len(keysSub)>0)
convert @VM to @FM in keysSub
open 'KNOWLEDGE_BASE' to hKNOWLEDGE_BASE else
Msg(@WINDOW , "Could not open the table 'KNOWLEDGE_BASE'")
return
end
/* If iKeyCountSub is empty then there are no matching criteria for the search so the loop will
not be executed */
for n = 1 to iKeyCountSub
read RECORD from hKNOWLEDGE_BASE, keysSub<n> else
Msg(@WINDOW , "Could not read from the Knowledge_Base table in the loop")
return
end
if n = 1 then
iContPos = iContPos + 1
content<iContPos> = '1-2:':RECORD<TYPE$>
list2 = RECORD<TYPE$>
/*We can now get all the TITLES in the table and add them to the CONTENT list of
information*/
gosub getAllTitles
end else
locate RECORD<TYPE$> in list2 using @FM setting pos then
**This TYPE is already there so don't add it again**
end else
iContPos = iContPos + 1
content<iContPos> = '1-2:':RECORD<TYPE$>
list2 = list2:@FM:RECORD<TYPE$>
/*We can now get all the TITLES in the table and add them to the CONTENT
list of information*/
gosub getAllTitles
end
end
next n
return
*************************************************************
```

```
getAllTitles:
/* These parameters will be passed to Btree.Extract to get all the TITLES according to the TYPE
of the current record.
This search string has to be sure to only get the TITLES that are associated with it's
corresponding TYPE and PRODUCT */
search_string = "TYPE":@VM:"=":Record<TYPE$>:@FM:"PRODUCT":@VM:"=":RECORD<PRODUCT$>:@FM
table = "KNOWLEDGE_BASE"
open "DICT.":table To @DICT else
Msg(@WINDOW , "Could not open the DICTIONARY in the getAllTitles subroutine in the POPULATE click")
return
end
keysSub2 = ''
optionSub2 = ''
flagSub2 = ''
list3 = ''
/* IF Btree.Extract is successful, the variable "KEYS" holds the list of keys, @VM delimited, for
responses for the main topic */
Btree.Extract(search_string, table, @DICT, keysSub2, optionSub2, flagSub2)
**Get the number of keys returned by Btree.Extract
iKeyCountSub2 = count(keysSub2,@VM) + (len(keysSub2)>0)
convert @VM to @FM in keysSub2
open 'KNOWLEDGE_BASE' to hKNOWLEDGE_BASE else
Msg(@WINDOW , "Could not open the table KNOWLEDGE_BASE in the subroutine getAllTitles in the POPULATE
click")
return
end
/* If iKeyCountSub2 is empty then there are no matching criteria for the search so the loop will
not be executed */
for t = 1 to iKeyCountSub2
read RECORD from hKNOWLEDGE_BASE, keysSub2<t> else
call Msg(@WINDOW , 'Could not read from the Knowledge_Base table')
return
end
iContPos = iContPos + 1
content<iContPos> = '3-3:':RECORD<TITLE$>
next t
return
```

5. Call this function from the CLICK event of the B_POPULATE button, as shown below. Substitute your account for **EXAMPLES**.