MCC_NListtree

COLLABORATORS

	TITLE : MCC_NListtree	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 23, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

MCC_NListtree

1.1 MCC_NListtree.doc

```
                                NListtree.mcc

background

MUIA_NListtree_Active         MUIA_NListtree_ActiveList
MUIA_NListtree_AutoVisible    MUIA_NListtree_CloseHook
MUIA_NListtree_CompareHook    MUIA_NListtree_ConstructHook
MUIA_NListtree_DestructHook   MUIA_NListtree_DisplayHook
MUIA_NListtree_DoubleClick    MUIA_NListtree_DragDropSort
MUIA_NListtree_DupNodeName    MUIA_NListtree_EmptyNodes
MUIA_NListtree_FindNameHook   MUIA_NListtree_Format
MUIA_NListtree_MultiSelect    MUIA_NListtree_MultiTestHook
MUIA_NListtree_OpenHook       MUIA_NListtree_Quiet
MUIA_NListtree_Title          MUIA_NListtree_TreeColumn
MUIM_NListtree_Close          MUIM_NListtree_Copy
MUIM_NListtree_Exchange       MUIM_NListtree_FindName
MUIM_NListtree_GetEntry       MUIM_NListtree_GetNr
MUIM_NListtree_Insert         MUIM_NListtree_Move
MUIM_NListtree_MultiTest      MUIM_NListtree_NextSelected
MUIM_NListtree_Open           MUIM_NListtree_Redraw
MUIM_NListtree_Remove         MUIM_NListtree_Rename
MUIM_NListtree_Select         MUIM_NListtree_Sort
MUIM_NListtree_TestPos
```

1.2 NListtree.mcc/background (information)

```
    There are two possible entry-types in a NListtree class list:
    Leaves and nodes. Leaves are simple entries which have no special
    features except they are holding some data. Nodes are almost
    the same type, holding data too, but having a list attached
    where you can simply add other entries which can be again leaves
    or nodes.

    Every node is structured as follows:
```

```
    struct MUI_NListtree_TreeNode {

        struct    MinNode    tn_Node;
        STRPTR    tn_Name;
        UWORD    tn_Flags;
        APTR    tn_User;
    };
```

It contains a name field tn_Name, flags tn_Flags and a pointer to user data tn_User.

The tn_Flags field can hold the following flags:

TNF_LIST	The node contains a list where other nodes can be inserted.
TNF_OPEN	The list node is open, sub nodes are displayed.
TNF_FROZEN	The node doesn't react on doubleclick or open/close by the user.
TNF_NOSIGN	The indicator of list nodes isn't shown.
TNF_SELECTED	The entry is currently selected.

These flags, except TNF_SELECTED, can be used in MUIM_NListtree_Insert at creation time. They will be passed to the newly created entry. Also you can do a quick check about the state and kind of each entry. But - NEVER EVER - modify any flag yourself or NListtree will crash. Be warned!

****
THE ABOVE STRUCT IS READ-ONLY!! NEVER CHANGE ANY ENTRY OF THIS STRUCTURE DIRECTLY NOR THINK ABOUT THE CONTENTS OF ANY PRIVATE FIELD OR YOU WILL DIE IN HELL!
****

You can create very complex tree structures. NListtree only uses one list which holds all information needed and has no extra display list like other list tree classes ;-)

The tree nodes can be inserted and removed, sorted, moved, exchanged, renamed or multi selected. To sort you can also drag&drop them. Modifications can be made in relation to the whole tree, to only one level, to a sub-tree or to only one tree node.

The user can control the listtree by the MUI keys, this means a node is opened with "Right" and closed with "Left". Check your MUI prefs for the specified keys.

You can define which of the columns will react on double-clicking. The node toggles its status from open or closed and vice versa.

Drag&Drop capabilities:

```
    If you set MUIA_NList_DragSortable to TRUE, the list tree will
    become active for Drag&Drop. This means you can drag and drop
    entries on the same list tree again. While dragging an indicator
    shows where to drop.

        Drag a    Drop on        Result

        leaf    leaf        Exchange leaves.
        node    leaf        Nothing happens.
        entry    closed node    Move entry, MUIA_NListtree_CompareHook is use
d.
        entry    open node    Move entry to defined position.

    You can not drop an entry on itself, nor can you drop an opened node on
    any of its members.

    To exchange data with other objects, you have to create your own
    subclass of NListtree class and react on the drag methods.


    Author: Carsten Scholling    (c)1999-2000    email: cs@@aphaso.de
```

1.3 NListtree.mcc/MUIA_NListtree_Active

```
   NAME

        MUIA_NListtree_Active -- [.SG], struct MUI_NListtree_TreeNode *


   SPECIAL VALUES

        MUIV_NListtree_Active_Off
        MUIV_NListtree_Active_Parent
        MUIV_NListtree_Active_First
        MUIV_NListtree_Active_FirstVisible
        MUIV_NListtree_Active_LastVisible


   FUNCTION

        Setting this attribute will move the cursor to the defined tree node
        if it is visible. If the node is in an opened tree the listview is
        scrolling into the visible area. Setting MUIV_NListtree_Active_Off will
        vanish the cursor.

        MUIV_NListtree_Active_First/FirstVisible/LastVisible are special values
        for activating the lists first or the top/bottom visible entry.

        See MUIA_NListtree_AutoVisible for special activation features.

        If this attribute is read it returns the active tree node. The result
        is MUIV_NListtree_Active_Off if there is no active entry.
```

NOTIFICATIONS

You can create a notification on MUIA_NListtree_Active. The TriggerValue is the active tree node.

SEE ALSO

MUIA_NListtree_AutoVisible, MUIA_NList_First, MUIA_NList_Visible, MUIA_NListtree_ActiveList

1.4 NListtree.mcc/MUIA_NListtree_ActiveList

NAME

MUIA_NListtree_ActiveList -- [..G], struct MUI_NListtree_TreeNode *

SPECIAL VALUES

MUIV_NListtree_ActiveList_Off

FUNCTION

If this attribute is read it returns the active list node. The active list node is always the parent of the active entry. The result is MUIV_NListtree_ActiveList_Off if there is no active list (when there is no active entry).

NOTIFICATIONS

You can create notifications on MUIA_NListtree_ActiveList. The TriggerValue is the active list node.

SEE ALSO

MUIA_NListtree_Active

1.5 NListtree.mcc/MUIA_NListtree_AutoVisible

NAME

MUIA_NListtree_AutoVisible -- [ISG], struct MUI_NListtree_TreeNode *

SPECIAL VALUES

```
      MUIV_NListtree_AutoVisible_Off
      MUIV_NListtree_AutoVisible_Normal
      MUIV_NListtree_AutoVisible_FirstOpen
      MUIV_NListtree_AutoVisible_Expand


  FUNCTION

      Set this to make your list automatically jump to the active
      entry.

          MUIV_NListtree_AutoVisible_Off:
              The display does NOT scroll the active entry into the
              visible area.

          MUIV_NListtree_AutoVisible_Normal:
              This will scroll the active entry into the visible area
              if it is visible (entry is a member of an open node).

          MUIV_NListtree_AutoVisible_FirstOpen:
              Nodes are not opened, but the first open parent node of
              the active entry is scrolled into the visible area if the
              active entry is not visible.

          MUIV_NListtree_AutoVisible_Expand:
              All parent nodes are opened until the first open node is
              reached and the active entry will be scrolled into the
              visible area.


  NOTIFICATIONS


  SEE ALSO

      MUIA_NListtree_Active, MUIA_NList_AutoVisible
```

1.6 NListtree.mcc/MUIA_NListtree_CloseHook

```
  NAME

      MUIA_NListtree_CloseHook -- [IS.], struct Hook *


  SPECIAL VALUES


  FUNCTION

      The close hook is called after a list node is closed, then you can
      change the list.

      The close hook will be called with the hook in A0, the object in A2
      and a MUIP_NListtree_CloseMessage struct in A1 (see nlisttree_mcc.h).
```

```
     To remove the hook set this to NULL.

NOTIFICATION

SEE ALSO

     MUIA_NListtree_Open, MUIA_NListtree_CloseHook
```

1.7 NListtree.mcc/MUIA_NListtree_CompareHook

```
NAME

     MUIA_NListtree_CompareHook -- [IS.], struct Hook *

SPECIAL VALUES

     MUIV_NListtree_CompareHook_Head
     MUIV_NListtree_CompareHook_Tail
     MUIV_NListtree_CompareHook_LeavesTop
     MUIV_NListtree_CompareHook_LeavesMixed
     MUIV_NListtree_CompareHook_LeavesBottom

FUNCTION

     Set this attribute to your own hook if you want to sort the entries in
     the list tree by your own way.

     When you sort your list or parts of your list via MUIM_NListtree_Sort,
     using the insert method with MUIV_NListtree_Insert_Sort or dropping an
     entry on a closed node, this compare hook is called.

     There are some builtin compare hooks available, called:

         MUIV_NListtree_CompareHook_Head
             Any entry is inserted at head of the list.

         MUIV_NListtree_CompareHook_Tail
             Any entry is inserted at tail of the list.

         MUIV_NListtree_CompareHook_LeavesTop
             Leaves are inserted at top of the list, nodes at bottom. They are
             alphabetically sorted.

         MUIV_NListtree_CompareHook_LeavesMixed
             The entries are only alphabetically sorted.

         MUIV_NListtree_CompareHook_LeavesBottom
             Leaves are inserted at bottom of the list, nodes at top. They are
             alphabetically sorted. This is default.
```

```
     The hook will be called with the hook in A0, the object in A2 and
     a MUIP_NListtree_CompareMessage struct in A1 (see nlisttree_mcc.h). You
     should return something like:

         <0    (TreeNode1 <  TreeNode2)
          0    (TreeNode1 == TreeNode2)
         >0    (TreeNode1 >  TreeNode2)


NOTIFICATION


SEE ALSO

     MUIA_NListtree_Insert, MUIM_DragDrop,
     MUIA_NList_CompareHook
```

1.8 NListtree.mcc/MUIA_NListtree_ConstructHook

```
NAME

     MUIA_NListtree_ConstructHook -- [IS.], struct Hook *


SPECIAL VALUES

     MUIV_NListtree_ConstructHook_String

     MUIV_NListtree_ConstructHook_Flag_AutoCreate
         If using the KeepStructure feature in MUIM_NListtree_Move or
         MUIM_NListtree_Copy, this flag will be set when calling your
         construct hook. Then you can react if your hook is not simply
         allocating memory.


FUNCTION

         The construct hook is called whenever you add an entry to your
         listtree. The pointer isn't inserted directly, the construct hook is
         called and its result code is added.

         When an entry shall be removed the corresponding destruct hook is
         called.

         The construct hook will be called with the hook in A0, the object in
         A2 and a MUIP_NListtree_ConstructMessage struct in A1 (see ↩
            nlisttree_mcc.
     h).
         The message holds a standard kick 3.x memory pool pointer. If you want ↩
            ,
         you can use the exec or amiga.lib functions for allocating memory
         within this pool, but this is only an option.
```

```
          If the construct hook returns NULL, nothing will be added to the list.

          There is a builtin construct hook available called
          MUIV_NListtree_ConstructHook_String. This expects that the 'User' data
          in the treenode is a string, which is copied. Of course you have to
          use MUIV_NListtree_DestructHook_String in this case!

          To remove the hook set this to NULL.


NOTIFICATION


SEE ALSO

     MUIA_NList_ConstructHook, MUIA_NListtree_DestructHook,
     MUIA_NListtree_DisplayHook
```

1.9 NListtree.mcc/MUIA_NListtree_DestructHook

```
NAME

     MUIA_NListtree_DestructHook -- [IS.], struct Hook *


SPECIAL VALUES

     MUIV_NListtree_DestructHook_String


FUNCTION

          Set up a destruct hook for your listtree. The destruct hook is called
          whenevere you remove an entry from the listtree. Here you can free  ↩
             memory
          which was allocated by the construct hook before.

          The destruct hook will be called with the hook in A0, the object in
          A2 and a MUIP_NListtree_DestructMessage struct in A1 (see  ↩
             nlisttree_mcc.h
     ).
          The message holds a standard kick 3.x memory pool pointer. You must
          use this pool when you have used it inside the construct hook to
          allocate pooled memory.

          There is a builtin destruct hook available called
          MUIV_NListtree_DestructHook_String. This expects that the 'User' data
          in the treenode is a string and you have used
          MUIV_NListtree_ConstructHook_String in the construct hook!

          To remove the hook set this to NULL.


NOTIFICATION
```

SEE ALSO

MUIA_NList_ConstructHook, MUIA_NListtree_ConstructHook,
MUIA_NListtree_DisplayHook

1.10 NListtree.mcc/MUIA_NListtree_DisplayHook

NAME

MUIA_NListtree_DisplayHook -- [IS.],

SPECIAL VALUES

FUNCTION

You have to supply a display hook to specify what should be shown in the listview, otherwise only the name of the nodes is displayed.

The display hook will be called with the hook in A0, the object in A2 and a MUIP_NListtree_DisplayMessage struct in A1 (see nlisttree_mcc ↩ .h)

.

The structure holds a pointer to a string array containing as many entries as your listtree may have columns. You have to fill this array with the strings you want to display. Check out that the array pointer of the tree column isn't used or set to NULL, if the normal name of the node should appear.

You can set the array pointer of the tree column to a string, which is diplayed instead of the node name. You can use this to mark nodes.

See MUIA_NList_Format for details about column handling.

To remove the hook and use the internal default display hook set this to NULL.

NOTIFICATION

SEE ALSO

MUIA_NList_Format, MUIA_Text_Contents

1.11 NListtree.mcc/MUIA_NListtree_DoubleClick

NAME

MUIA_NListtree_DoubleClick -- [ISG], ULONG

SPECIAL VALUES

MUIV_NListtree_DoubleClick_Off
MUIV_NListtree_DoubleClick_All
MUIV_NListtree_DoubleClick_Tree

FUNCTION

A doubleclick opens a node if it was closed, it is closed if the node was open. You have to set the column which should do this.

Normally only the column number is set here, but there are special values:

MUIV_NListtree_DoubleClick_Off:
A doubleclick is not handled.

MUIV_NListtree_DoubleClick_All:
All columns reacts on a doubleclick.

MUIV_NListtree_DoubleClick_Tree
Only a doubleclick on the defined tree column is recognized.

NOTIFICATION

The TriggerValue of the notification is the tree node you have double-clicked, you can GetAttr() MUIA_NListtree_DoubleClick for the column number. The struct 'MUI_NListtree_TreeNode *'~is used for trigger.

The notification is done on leaves and on node columns, which are not set in MUIA_NListtree_DoubleClick.

SEE ALSO

1.12 NListtree.mcc/MUIA_NListtree_DragDropSort

NAME

MUIA_NListtree_DragDropSort -- [IS.], BOOL

SPECIAL VALUES

FUNCTION

Setting this attribute to FALSE will disable the ability to sort the list tree by drag & drop. Defaults to TRUE.

NOTIFICATION

SEE ALSO

1.13 NListtree.mcc/MUIA_NListtree_DupNodeName

NAME

MUIA_NListtree_DupNodeName -- [IS.], BOOL

SPECIAL VALUES

FUNCTION

If this attribute is set to FALSE the names of the node will not be duplicated, only the string pointers are used. Be careful the strings have to be valid everytime.

NOTIFICATION

SEE ALSO

1.14 NListtree.mcc/MUIA_NListtree_EmptyNodes

NAME

MUIA_NListtree_EmptyNodes -- [IS.], BOOL

SPECIAL VALUES

FUNCTION

Setting this attribute to TRUE will display all empty nodes as leaves, this means no list indicator is shown. Nevertheless the entry is handled like a node.

NOTIFICATION

SEE ALSO

1.15 NListtree.mcc/MUIA_NListtree_FindNameHook

NAME

MUIA_NListtree_FindNameHook -- [IS.],

SPECIAL VALUES

MUIV_NListtree_FindNameHook_CaseSensitive
MUIV_NListtree_FindNameHook_CaseInsensitive
MUIV_NListtree_FindNameHook_Part
MUIV_NListtree_FindNameHook_PartCaseInsensitive

FUNCTION

You can install a FindName hook to specify your own search criteria.

The find name hook will be called with the hook in A0, the object in A2 and a MUIP_NListtree_FindNameMessage struct in A1 (see nlisttree_mcc.h). It should return ~ 0 for entries which are not matching the pattern and a value of 0 if a match.

The find name message structure holds a pointer to a string containing the name to search for and pointers to the name- and user-field of the node which is currently processed.

The MUIV_NListtree_FindNameHook_CaseSensitive will be used as default.

NOTIFICATION

SEE ALSO

1.16 NListtree.mcc/MUIA_NListtree_Format

NAME

MUIA_NListtree_Format -- [IS.], STRPTR

SPECIAL VALUES

```
FUNCTION

     Same as MUIA_NList_Format, but one column is reserved for the tree
     indicators and the names of the nodes.

     For further detailed information see MUIA_NList_Format!


NOTIFICATION


SEE ALSO

     MUIA_NList_Format, MUIA_NListtree_DisplayHook,
     MUIA_Text_Contents
```

1.17 NListtree.mcc/MUIA_NListtree_MultiSelect

```
NAME

     MUIA_NListtree_MultiSelect -- [I..],


SPECIAL VALUES

     MUIV_NListtree_MultiSelect_None
     MUIV_NListtree_MultiSelect_Default
     MUIV_NListtree_MultiSelect_Shifted
     MUIV_NListtree_MultiSelect_Always

     MUIV_NListtree_MultiSelect_Flag_AutoSelectChilds


FUNCTION

     Four possibilities exist for a listviews multi select
     capabilities:

         MUIV_NListtree_MultiSelect_None:
             The list tree cannot multiselect at all.

         MUIV_NListtree_MultiSelect_Default:
             The multi select type (with or without shift)
             depends on the users preferences setting.

         MUIV_NListtree_MultiSelect_Shifted:
             Overrides the users prefs, multi selecting only
             together with shift key.

         MUIV_NListtree_MultiSelect_Always:
             Overrides the users prefs, multi selecting
             without shift key.
```

NOTIFICATION

NOTES

Multi selection is highly experimental for now. Please report
bugs or suggestions whenever possible.
Also ideas about other selection methods are very welcome.

SEE ALSO

MUIA_NListtree_MultiTestHook, MUIM_NListtree_MultiSelect

1.18 NListtree.mcc/MUIA_NListtree_MultiTestHook

NAME

MUIA_NListtree_MultiTestHook -- [IS.], struct Hook *

SPECIAL VALUES

FUNCTION

If you plan to have a multi selecting list tree but not
all of your entries are actually multi selectable, you
can supply a MUIA_NListtree_MultiTestHook.

The multi test hook will be called with the hook in A0, the object
in A2 and a MUIP_NListtree_MultiTestMessage struct in A1 (see
nlisttree_mcc.h) and should return TRUE if the entry is multi
selectable, FALSE otherwise.

To remove the hook set this to NULL.

NOTIFICATION

SEE ALSO

MUIA_NListtree_ConstructHook, MUIA_NListtree_DestructHook

1.19 NListtree.mcc/MUIA_NListtree_OpenHook

NAME

MUIA_NListtree_OpenHook -- [IS.], struct Hook *

SPECIAL VALUES

FUNCTION

The open hook is called whenever a list node will be opened, so you can change the list before the node is open.

The open hook will be called with the hook in A0, the object in A2 and a MUIP_NListtree_OpenMessage struct in A1 (see nlisttree_mcc.h).

To remove the hook set this to NULL.

NOTIFICATION

SEE ALSO

MUIA_NListtree_Open, MUIA_NListtree_CloseHook

1.20 NListtree.mcc/MUIA_NListtree_Quiet

NAME

MUIA_NListtree_Quiet -- [.S.], QUIET

SPECIAL VALUES

FUNCTION

If you add/remove lots of entries to/from a listtree, this will cause lots of screen action and slow down the operation. Setting MUIA_NListtree_Quiet to TRUE will temporarily prevent the listtree from being refreshed, this refresh will take place only once when you set it back to FALSE again.

Do not use MUIA_NList_Quiet here!

NOTIFICATION

SEE ALSO

MUIM_NListtree_Insert, MUIM_NListtree_Remove

1.21 NListtree.mcc/MUIA_NListtree_Title

NAME

MUIA_NListtree_Title -- [IS.], BOOL

SPECIAL VALUES

FUNCTION

Specify a title for the current listtree.

For detailed information see MUIA_NList_Title!

NOTIFICATION

BUGS

The title should not be a string as for single column listviews. This attribute can only be set to TRUE or FALSE.

SEE ALSO

1.22 NListtree.mcc/MUIA_NListtree_TreeColumn

NAME

MUIA_NListtree_TreeColumn -- [ISG], ULONG

SPECIAL VALUES

FUNCTION

Specify the column of the list tree, the node indicator and the name of the node are displayed in.

NOTIFICATION

SEE ALSO

MUIA_NListtree_DisplayHook, MUIA_NListtree_Format

1.23 NListtree.mcc/MUIM_NListtree_Close

```
NAME

     MUIM_NListtree_Close -- Close the specified list node. (V1)


SYNOPSIS

     DoMethod(obj, MUIM_NListtree_Close,
         struct MUI_NListtree_TreeNode *listnode,
         struct MUI_NListtree_TreeNode *treenode,
         ULONG flags);


FUNCTION

     Close a node or nodes of a listtree. It is checked if the tree node
     is a node, not a leaf!

     When the active entry was a child of the closed node, the closed node
     will become active.


INPUTS

     listnode -    Specify the node which list is used to find the entry. The
                 search is started at the head of this list.

         MUIV_NListtree_Close_ListNode_Root
             The root list is used.

         MUIV_NListtree_Close_ListNode_Parent
             The list which is the parent of the active list is used.

         MUIV_NListtree_Close_ListNode_Active
             The list of the active node is used.


     treenode -    The node which should be closed. If there are children
                 of the node, they are also closed.

         MUIV_NListtree_Close_TreeNode_Head
             The head of the list defined in 'listnode' is closed.

         MUIV_NListtree_Close_TreeNode_Tail:
             Closes the tail of the list defined in 'listnode'.

         MUIV_NListtree_Close_TreeNode_Active:
             Closes the active node.

         MUIV_NListtree_Close_TreeNode_All:
             Closes all nodes of the list which is specified in
             'listnode'.
```

```
RESULT

EXAMPLE

     /* Close the active list. */
     DoMethod(obj, MUIM_NListtree_Close,
         MUIV_NListtree_Close_ListNode_Active,
         MUIV_NListtree_Close_TreeNode_Active, 0);


NOTES

BUGS

SEE ALSO
     MUIM_NListtree_Open
```

1.24 NListtree.mcc/MUIM_NListtree_Copy

```
NAME

     MUIM_NListtree_Copy -- Copy an entry (create it) to the spec. pos. (V1)


SYNOPSIS

     DoMethod(obj, MUIM_NListtree_Copy,
         struct MUI_NListtree_TreeNode *srclistnode,
         struct MUI_NListtree_TreeNode *srctreenode,
         struct MUI_NListtree_TreeNode *destlistnode,
         struct MUI_NListtree_TreeNode *desttreenode,
         ULONG flags);


FUNCTION

     Copy an entry to the position after a defined node. The complete
     child structure will be copied.


INPUTS

     srclistnode -    Specify the node which list is used to find the
                     entry. The search is started at the head of this
                     list.

         MUIV_NListtree_Copy_SourceListNode_Root
             The root list is used as the starting point.

         MUIV_NListtree_Copy_SourceListNode_Active
             The active list (the list of the active node) is used as
             the starting point.
```

srctreenode - Specifies the node which should be copied.

MUIV_NListtree_Copy_SourceTreeNode_Head
The head of the list defined in 'srclistnode' is copied.

MUIV_NListtree_Copy_SourceTreeNode_Tail
The tail of the list defined in 'srclistnode' is copied.

MUIV_NListtree_Copy_SourceTreeNode_Active
The active node is copied.

destlistnode - Specify the node which list is used to find the entry. The search is started at the head of this list.

MUIV_NListtree_Copy_DestListNode_Root
The root list.

MUIV_NListtree_Copy_DestListNode_Active
The list of the active node.

desttreenode - This node is the predecessor of the entry which is inserted.

MUIV_NListtree_Copy_DestTreeNode_Head
The node is copied to the head of the list defined in 'destlistnode'.

MUIV_NListtree_Copy_DestTreeNode_Tail
The node is copied to the tail of the list defined in 'destlistnode'.

MUIV_NListtree_Copy_DestTreeNode_Active:
The node is copied to one entry after the active node.

MUIV_NListtree_Copy_DestTreeNode_Sorted:
The node is copied to the list using the sort hook.

flags - Some flags to adjust moving.

MUIV_NListtree_Copy_Flag_KeepStructure
The full tree structure from the selected entry to the root list is copied (created) at destination.

RESULT

EXAMPLE

```
/* Copy the active entry to the head of
** another list node.
*/
DoMethod(obj,
    MUIV_NListtree_Copy_SourceListNode_Active,
    MUIV_NListtree_Copy_SourceTreeNode_Active,
    anylistnode,
    MUIV_NListtree_Copy_DestTreeNode_Head,
```

```
          0);

NOTES

BUGS

SEE ALSO

     MUIM_NListtree_Insert, MUIM_NListtree_Remove,
     MUIM_NListtree_Exchange, MUIA_NListtree_CompareHook,
     MUIM_NListtree_Move
```

1.25 NListtree.mcc/MUIM_NListtree_Exchange

```
NAME

     MUIM_NListtree_Exchange -- Exchanges two tree nodes. (V1)


SYNOPSIS

     DoMethod(obj, MUIM_NListtree_Exchange,
         struct MUI_NListtree_TreeNode *listnode1,
         struct MUI_NListtree_TreeNode *treenode1,
         struct MUI_NListtree_TreeNode *listnode2,
         struct MUI_NListtree_TreeNode *treenode2,
         ULONG flags);


FUNCTION

     Exchange two tree nodes.


INPUTS

          listnode1 -    Specify the list node of the entry which should be  ↩
             exchang
     ed.

               MUIV_NListtree_Exchange_ListNode1_Root
                   The root list is used.

               MUIV_NListtree_Exchange_ListNode1_Active
                   The active list (the list of the active node) is used.

          treenode1 -    Specify the node which should be exchanged.

               MUIV_NListtree_Exchange_TreeNode1_Head
                   The head of the list defined in 'listnode1' is
                   exchanged.

               MUIV_NListtree_Exchange_TreeNode1_Tail
```

```
                The tail of the list defined in 'listnode1' is
                exchanged.

            MUIV_NListtree_Exchange_TreeNode1_Active
                The active node is exchanged.

        listnode2 -    Specify the second list node which is used for exchange ↩
            .

            MUIV_NListtree_Exchange_ListNode2_Root
                The root list.

            MUIV_NListtree_Exchange_ListNode2_Active
                The list of the active node.

        treenode2 -    This node is the second entry which is exchanged.

            MUIV_NListtree_Exchange_TreeNode2_Head
                The node 'treenode1' is exchanged with the head of the
                list defined in 'listnode2'.

            MUIV_NListtree_Exchange_TreeNode2_Tail
                The node 'treenode1' is exchanged with the tail of the
                list defined in 'ln2'.

            MUIV_NListtree_Exchange_TreeNode2_Active:
                The node 'treenode1' is exchanged with the active node.

            MUIV_NListtree_Exchange_TreeNode2_Up:
                The node 'treenode1' is exchanged with the entry
                previous to the one specified in 'treenode1'.

            MUIV_NListtree_Exchange_TreeNode2_Down:
                The node 'treenode1' is exchanged with the entry next
                (the successor) to the one specified in 'treenode1'.


RESULT

EXAMPLE

     /* Exchange the active entry with the successor. */
     DoMethod(obj,
         MUIV_NListtree_Exchange_ListNode1_Active,
         MUIV_NListtree_Exchange_TreeNode1_Active,
         MUIV_NListtree_Exchange_ListNode2_Active,
         MUIV_NListtree_Exchange_TreeNode2_Down,
         0);


NOTES

BUGS

SEE ALSO

     MUIM_NListtree_Move, MUIM_NListtree_Insert,
```

```
     MUIM_NListtree_Remove
```

1.26 NListtree.mcc/MUIM_NListtree_FindName

```
NAME

     MUIM_NListtree_FindName -- Find node using name match. (V1)


SYNOPSIS

     struct MUI_NListtree_TreeNode *treenode =
         DoMethod(obj, MUIM_NListtree_FindName,
             struct MUI_NListtree_TreeNode *listnode,
             STRPTR name, ULONG flags);


FUNCTION

     Find a node which name matches the specified one using the list node as
     start point..


INPUTS
     listnode -    Specify the node which list is used to find the name.

         MUIV_NListtree_FindName_ListNode_Root
             Use the root list as the base list.

         MUIV_NListtree_FindName_ListNode_Active
             Use the list of the active node as the base.

     name -        Specify the name of the node to find.

     flags:

         MUIV_NListtree_FindName_Flag_SameLevel
             Only nodes on the same level are affected.

         MUIV_NListtree_FindName_Flag_Visible
             The node is only returned if it is visible (only visible
             entries are checked).

         MUIV_NListtree_FindName_Flag_Activate
             If found, the entry will be activated.


RESULT

     Returns the found node if available, NULL otherwise.


EXAMPLE
```

```
    /* Find 2nd node by name. */
    struct MUI_NListtree_TreeNode *treenode =
        DoMethod(obj, MUIM_NListtree_FindName,
            listnode, "2nd node",
            MUIV_NListtree_FindName_SameLevel|
            MUIV_NListtree_FindName_Visible);

    if ( treenode == NULL )
    {
        PrintToUser( "No matching entry found." );
    }
```

NOTES

BUGS

SEE ALSO

```
    MUIM_NListtree_GetEntry
```

1.27 NListtree.mcc/MUIM_NListtree_GetEntry

NAME

```
    MUIM_NListtree_GetEntry -- Get another node in relation to this. (V1)
```

SYNOPSIS

```
    struct MUI_NListtree_TreeNode *rettreenode =
        DoMethod(obj, MUIM_NListtree_GetEntry,
            struct MUI_NListtree_TreeNode *treenode,
            LONG pos, ULONG flags);
```

FUNCTION

```
    Get another node in relation to the specified list or node.
```

INPUTS

```
    treenode -    Define the node which is used to find another one.
                  This can also be a list node, if the position is
                  related to a list.

        MUIV_NListtree_GetEntry_ListNode_Root
            The root list is used.

        MUIV_NListtree_GetEntry_ListNode_Active:
            The list with the active entry is used.

    pos -    The relative position of the node 'treenode'.
```

```
            MUIV_NListtree_GetEntry_Position_Head
                The head of the list is returned.

            MUIV_NListtree_GetEntry_Position_Tail
                The tail of the list is returned.

            MUIV_NListtree_GetEntry_Position_Active
                The active node is returned. If there is no active entry,
                NULL is returned.

            MUIV_NListtree_GetEntry_Position_Next
                The node next to the specified node is returned. Returns NULL
                if there is no next entry.

            MUIV_NListtree_GetEntry_Position_Previous
                The node right before the specified node is returned.
                Returns NULL if there is no previous entry (if 'treenode'
                is the head of the list.

            MUIV_NListtree_GetEntry_Position_Parent
                The list node of the specified 'treenode' is returned.

        flags:

            MUIV_NListtree_GetEntry_SameLevel:
                Only nodes on the same level are affected.

            MUIV_NListtree_GetEntry_Flag_Visible:
                The position is counted on visible entries only.


RESULT

        Returns the requested node if available, NULL otherwise.


EXAMPLE

        /* Get the next entry. */
        struct MUI_NListtree_TreeNode *treenode =
            DoMethod(obj, MUIM_NListtree_GetEntry, treenode,
            MUIV_NListtree_GetEntry_Position_Next, 0);

        if ( treenode != NULL )
        {
            PrintToUser( "Next entry found!" );
        }


NOTES

BUGS

SEE ALSO
```

```
     MUIM_NList_GetEntry
```

1.28 NListtree.mcc/MUIM_NListtree_GetNr

```
NAME

     MUIM_NListtree_GetNr -- Get the position number of a tree node. (V1)


SYNOPSIS

     ULONG number = DoMethod(obj, MUIM_NListtree_GetNr,
         struct MUI_NListtree_TreeNode *treenode, ULONG flags);


FUNCTION

     Get the position number of the specified tree node.


INPUTS

     treenode -    Specify the node to count the position of.

         MUIV_NListtree_GetNr_TreeNode_Active:
             The position is counted related to the active node.

     flags:

         MUIV_NListtree_GetNr_Flag_CountAll
             Returns the number of all entries.

         MUIV_NListtree_GetNr_Flag_CountLevel
             Returns the number of entries of the list the
             specified node is in.

         MUIV_NListtree_GetNr_Flag_CountList
             Returns the number of the entries of the active list node
             (the specified node is in).

         MUIV_NListtree_GetNr_Flag_ListEmpty
             Returns TRUE if the specified list node is empty.


RESULT

EXAMPLE

     /* Check if the active (list) node is empty. */
     ULONG empty = DoMethod(obj, MUIM_NListtree_GetNr,
         MUIV_NListtree_GetNr_TreeNode_Active,
         MUIV_NListtree_GetNr_Flag_ListEmpty);

     if ( empty == TRUE )
```

```
    {
        AddThousandEntries();
    }

NOTES

BUGS

SEE ALSO

    MUIM_NListtree_GetEntry
```

1.29 NListtree.mcc/MUIM_NListtree_Insert

```
NAME

    MUIM_NListtree_Insert -- Insert an entry at the specified position. (V1)


SYNOPSIS

    struct MUI_NListtree_TreeNode *treenode =
        DoMethod(obj, MUIM_NListtree_Insert,
            STRPTR name, APTR userdata,
            struct MUI_NListtree_TreeNode *listnode,
            struct MUI_NListtree_TreeNode *prevtreenode,
            ULONG flags);


FUNCTION

    Insert an entry at the position, which is defined in 'listnode'
    and 'prevtreenode'. Name contains the name of the entry as string
    which is buffered. The user entry can be used as you like.


INPUTS

    listnode -        Specify the node which list is used to insert
                    the entry.

        MUIV_NListtree_Insert_ListNode_Root
            Use the root list.

        MUIV_NListtree_Insert_ListNode_Active
            Use the list of the active node.

        MUIV_NListtree_Insert_ListNode_LastInserted
            Insert entry in the list the last entry was inserted.


    prevtreenode -    The node which is the predecessor of the node
                    to insert.
```

```
            MUIV_NListtree_Insert_PrevNode_Head
                The entry will be inserted at the head of the list.

            MUIV_NListtree_Insert_PrevNode_Tail
                The entry will be inserted at the tail of the list.

            MUIV_NListtree_Insert_PrevNode_Active
                The entry will be inserted after the active node of
                the list.

            MUIV_NListtree_Insert_PrevNode_Sorted:
                The entry will be inserted using the defined sort hook.

        flags:

            MUIV_NListtree_Insert_Flag_Active
                The inserted entry will be set to active. This means the
                cursor is moved to the newly inserted entry. If the entry
                was inserted into a closed node, it will be opened.

            MUIV_NListtree_Insert_Flag_NextNode
                'prevtreenode' is the successor, not the predecessor.


    RESULT

        A pointer to the newly inserted entry.


    EXAMPLE
        /* Insert an entry after the active one and make it active. */
        DoMethod(obj, MUIM_NListtree_Insert,
            MUIV_NListtree_Insert_ListNode_Active,
            MUIV_NListtree_Insert_TreeNode_Active,
            MUIV_NListtree_Insert_Flag_Active);


    NOTES

    BUGS

        Not implemented yet:
            MUIV_NListtree_Insert_Flag_NextNode


    SEE ALSO

        MUIA_NListtree_ConstructHook, MUIA_NListtree_CompareHook
```

1.30 NListtree.mcc/MUIM_NListtree_Move

```
    NAME

        MUIM_NListtree_Move -- Move an entry to the specified position. (V1)
```

SYNOPSIS

```
DoMethod(obj, MUIM_NListtree_Move,
    struct MUI_NListtree_TreeNode *oldlistnode,
    struct MUI_NListtree_TreeNode *oldtreenode,
    struct MUI_NListtree_TreeNode *newlistnode,
    struct MUI_NListtree_TreeNode *newtreenode,
    ULONG flags);
```

FUNCTION

Move an entry to the position after a defined node.

INPUTS

oldlistnode - Specify the node which list is used to find the entry. The search is started at the head of this list.

MUIV_NListtree_Move_OldListNode_Root
The root list is used as the starting point.

MUIV_NListtree_Move_OldListNode_Active
The active list (the list of the active node) is used as the starting point.

oldtreenode - Specify the node which should be moved.

MUIV_NListtree_Move_OldTreeNode_Head
The head of the list defined in 'oldlistnode' is moved.

MUIV_NListtree_Move_OldTreeNode_Tail
The tail of the list defined in 'oldlistnode' is moved.

MUIV_NListtree_Move_OldTreeNode_Active
The active node is moved.

newlistnode - Specify the node which list is used to find the entry. The search is started at the head of this list.

MUIV_NListtree_Move_NewListNode_Root
The root list.

MUIV_NListtree_Move_NewListNode_Active
The list of the active node.

newtreenode - This node is the predecessor of the entry which is inserted.

MUIV_NListtree_Move_NewTreeNode_Head
The node is moved to the head of the list defined in 'newlistnode'.

```
        MUIV_NListtree_Move_NewTreeNode_Tail
            The node is moved to the tail of the list defined in
            'newlistnode'.

        MUIV_NListtree_Move_NewTreeNode_Active:
            The node is moved to one entry after the active node.

        MUIV_NListtree_Move_NewTreeNode_Sorted:
            The node is moved to the list using the sort hook.

    flags -        Some flags to adjust moving.

        MUIV_NListtree_Move_Flag_KeepStructure
            The full tree structure from the selected entry to
            the root list is moved (created at destination).


RESULT

EXAMPLE

    /* Move an entry to the head of another list-node. */
    DoMethod(obj,
        MUIV_NListtree_Move_OldListNode_Active,
        MUIV_NListtree_Move_OldTreeNode_Active,
        somelistmode,
        MUIV_NListtree_Move_NewTreeNode_Head,
        0);


NOTES

BUGS

SEE ALSO

    MUIM_NListtree_Insert, MUIM_NListtree_Remove,
    MUIM_NListtree_Exchange, MUIA_NListtree_CompareHook,
    MUIM_NListtree_Copy
```

1.31 NListtree.mcc/MUIM_NListtree_MultiTest

```
NAME

    MUIM_NListtree_MultiTest -- Called for every selection. (V1)


SYNOPSIS

    DoMethod(obj, MUIM_NListtree_MultiTest,
        struct MUIP_NListtree_MultiTest *multimessage);
```

```
FUNCTION

     This method must not be called directly. It will be called by
     NListtree just before multiselection. You can redefine it and
     return TRUE or FALSE whether you want the entry to be multi-
     selectable or not.


INPUTS

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO

     MUIM_NListtree_Select, MUIA_NListtree_MultiTest
```

1.32 NListtree.mcc/MUIM_NListtree_NextSelected

```
NAME

     MUIM_NListtree_NextSelected -- Select the specified tree node. (V1)


SYNOPSIS

     DoMethod(obj, MUIM_NListtree_NextSelected,
         struct MUI_NListtree_TreeNode **treenode);


FUNCTION

     Iterate through the selected entries of a tree. This method steps
     through the contents of a (multi select) list tree and returns
     every entry that is currently selected. When no entry is selected
     but an entry is active, only the active entry will be returned.

     This behaviour will result in not returning the active entry when
     you have some other selected entries somewhere in your list. Since
     the active entry just acts as some kind of cursor mark, this seems
     to be the only sensible possibility to handle multi selection
     together with keyboard control.


INPUTS

     treenode -  A pointer to a pointer of struct MUI_NListtree_TreeNode
                 that will hold the returned entry. Must be set to
                 MUIV_NListtree_NextSelected_Start at start of iteration
```

```
                and is set to MUIV_NListtree_NextSelected_End when
                iteration is finished.

        MUIV_NListtree_NextSelected_Start    Set this to start iteration.
        MUIV_NListtree_NextSelected_End        Will be set to this, if
                                             last selected entry reached.


RESULT

EXAMPLE

     /* Iterate through a list */
     struct MUI_NListtree_TreeNode *treenode;

     treenode = MUIV_NListtree_NextSelected_Start;

     for (;;)
     {
         DoMethod(listtree, MUIM_NListtree_NextSelected, &treenode);

         if (treenode==MUIV_NListtree_NextSelected_End)
             break;

         printf("selected: %s\n", treenode->tn_Name);
     }


NOTES

BUGS

SEE ALSO

     MUIM_NListtree_Select
```

1.33 NListtree.mcc/MUIM_NListtree_Open

```
NAME

     MUIM_NListtree_Open -- Open the specified tree node. (V1)


SYNOPSIS

     DoMethod(obj, MUIM_NListtree_Open,
         struct MUI_NListtree_TreeNode *listnode,
         struct MUI_NListtree_TreeNode *treenode,
         ULONG flags);


FUNCTION

     Opens a node in the listtree. To open a child, which isn't displayed,
```

```
     use 'MUIV_NListtree_Open_ListNode_Parent' to open all its parents, too.

     Only nodes can be opened.


INPUTS

     listnode -    Specify the node which list is used to open the node.

         MUIV_NListtree_Open_ListNode_Root
             The root list is used.

         MUIV_NListtree_Open_ListNode_Parent
             Indicates, that all the parents of the node specified in
             'treenode' should be opened too.

         MUIV_NListtree_Open_ListNode_Active
             The list of the active node is used.

     treenode -    The node to open.

         MUIV_NListtree_Open_TreeNode_Head
             Opens the head node of the list.

         MUIV_NListtree_Open_TreeNode_Tail
             Opens the tail node of the list.

         MUIV_NListtree_Open_TreeNode_Active
             The active node will be opened.

         MUIV_NListtree_Open_TreeNode_All:
             All the nodes of the list are opened.


RESULT

EXAMPLE
     /* Open the active list. */
     DoMethod(obj, MUIM_NListtree_Open,
         MUIV_NListtree_Open_ListNode_Active,
         MUIV_NListtree_Open_TreeNode_Active, 0);


NOTES

BUGS

SEE ALSO
     MUIM_NListtree_Close
```

1.34 NListtree.mcc/MUIM_NListtree_Redraw

```
NAME
```

```
     MUIM_NListtree_Redraw -- Redraw the specified tree node. (V1)


SYNOPSIS

     DoMethod(obj, MUIM_NListtree_Redraw,
         struct MUI_NListtree_TreeNode *treenode, ULONG flags);


FUNCTION

     Redraw the specified entry. See special values for completeness.


INPUTS

     treenode -    The tree node to be redrawn.

         MUIV_NListtree_Redraw_Active
             Redraw the active entry.

         MUIV_NListtree_Redraw_All
             Redraw the complete visible tree.


     flags:

         MUIV_NListtree_Redraw_Flag_Nr
             The data specified in 'treenode' is the entry number,
             not the tree node itself.


RESULT

EXAMPLE

     /* Redraw the active entry. */
     DoMethod(obj, MUIM_NListtree_Redraw,
         MUIV_NListtree_Redraw_Active, 0);


NOTES

BUGS

SEE ALSO

     MUIM_NList_TestPos
```

1.35 NListtree.mcc/MUIM_NListtree_Remove

```
NAME

     MUIM_NListtree_Remove -- Remove the specified entry(ies). (V1)
```

SYNOPSIS

```
DoMethod(obj, MUIM_NListtree_Remove,
    struct MUI_NListtree_TreeNode *listnode,
    struct MUI_NListtree_TreeNode *treenode,
    ULONG flags);
```

FUNCTION

Removes a node or nodes from the listtree. When the active entry is removed, the successor will become active.

INPUTS

listnode - Specify the node which list is used to find the entry which should be removed. The search is started at the begin of this list.

MUIV_NListtree_Remove_ListNode_Root
The root list is used.

MUIV_NListtree_Remove_ListNode_Active
The list of the active node is used.

treenode - The node which should be removed. If there are children of this node, they are also removed.

MUIV_NListtree_Remove_TreeNode_Head
The head of the list defined in 'listnode' is removed.

MUIV_NListtree_Remove_TreeNode_Tail
The tail of the list defined in 'listnode' is removed.

MUIV_NListtree_Remove_TreeNode_Active
Removes the active node.

MUIV_NListtree_Remove_TreeNode_All
All nodes of the list which is specified in 'listnode', are removed. Other nodes of parent lists are not affected.

RESULT

EXAMPLE

```
/* Remove the active entry if delete is pressed! */
DoMethod(bt_delete, MUIM_Notify, MUIA_Pressed, FALSE,
    lt_list, 4, MUIM_NListtree_Remove,
    MUIV_NListtree_Remove_ListNode_Active,
    MUIV_NListtree_Remove_TreeNode_Active, 0);
```

```
NOTES

BUGS

SEE ALSO

     MUIM_NListtree_Insert, MUIA_NListtree_DestructHook,
     MUIM_NList_Active
```

1.36 NListtree.mcc/MUIM_NListtree_Rename

```
NAME

     MUIM_NListtree_Rename -- Rename the specified node. (V1)


SYNOPSIS

     struct MUI_NListtree_TreeNode *treenode =
         DoMethod(obj, MUIM_NListtree_Rename,
             struct MUI_NListtree_TreeNode *treenode,
             STRPTR newname, ULONG flags);


FUNCTION

     Rename the specified node.

     If you want to rename the tn_User field (see flags below), the construct
     and destruct hooks are used!
     If you have not specified these hooks, only the pointers will be copied.


INPUTS

     treenode -    Specifies the node which should be renamed.

         MUIV_NListtree_Rename_TreeNode_Active:
             Rename the active tree node.

     newname -    The new name or pointer.

     flags:

         MUIV_NListtree_Rename_Flag_User
             The tn_User field is renamed.

         MUIV_NListtree_Rename_Flag_NoRefresh
             The list entry will not be refreshed.


RESULT

     Returns the pointer of the renamed tree node.
```

```
EXAMPLE

     /* Rename the active tree node. */
     DoMethod(obj, MUIM_NListtree_Rename,
         MUIV_NListtree_Rename_TreeNode_Active,
         "Very new name", 0);


NOTES

BUGS

SEE ALSO

     MUIA_NListtree_ConstructHook, MUIA_NListtree_DestructHook
```

1.37 NListtree.mcc/MUIM_NListtree_Select

```
NAME

     MUIM_NListtree_Select -- Select the specified tree node. (V1)


SYNOPSIS

     DoMethod(obj, MUIM_NListtree_Select,
         struct MUI_NListtree_TreeNode *treenode, LONG seltype,
         LONG selflags, LONG *state);


FUNCTION

     Select or unselect a tree entry or ask an entry about its state.
     See special values for completeness.


INPUTS

     treenode -    The tree node to be selected/unselected/asked.

         MUIV_NListtree_Select_Active    For the active entry.
         MUIV_NListtree_Select_All        For all entries.
         MUIV_NListtree_Select_Visible    For all visible entries.

     seltype -    Type of selection/unselection/ask

         MUIV_NListtree_Select_Off        Unselect entry.
         MUIV_NListtree_Select_On        Select entry.
         MUIV_NListtree_Select_Toggle    Toggle entries state.
         MUIV_NListtree_Select_Ask        Just ask about the state.

     selflags -    Some kind of specials.
```

```
          MUIV_NListtree_Select_Flag_Force
              Adding this flag to seltype forces the selection by
              bypassing the multi test hook.

     state -        Pointer to a longword. If not NULL, it will be filled
                 with the current selection state of the entry.


RESULT

EXAMPLE

     /* Select the active entry. */
     LONG retstate;

     DoMethod(obj, MUIM_NListtree_Select,
         MUIV_NListtree_Select_Active, MUIV_NListtree_Select_On,
         0, &retstate);

     /*    We must check this, because the multi test hook may
         cancel our selection. */
     if (retstate == MUIV_NListtree_Select_On) {
         ...
     }


NOTES

     If treenode==MUIV_NListtree_Select_All and
     seltype==MUIV_NListtree_Select_Ask, state will be filled with
     the total number of selected entries.
     If only the active entry is selected, has a cursor mark (see
     MUIM_NListtree_NextSelected for that), you will receive 0 as
     the number of selected entries.


BUGS

SEE ALSO

     MUIA_NListtree_MultiTestHook
```

1.38 NListtree.mcc/MUIM_NListtree_Sort

```
NAME

     MUIM_NListtree_Sort -- Sort the specified list node. (V1)


SYNOPSIS

     DoMethod(obj, MUIM_NListtree_Sort,
         struct MUI_NListtree_TreeNode *listnode,
```

```
        ULONG flags);

FUNCTION

     Sort the specified list node using the sort hook.

INPUTS

     listnode -    List node to sort.

         MUIV_NListtree_Sort_ListNode_Root
             Sort the root list.

         MUIV_NListtree_Sort_ListNode_Active
             Sort the list node of the active entry.

         MUIV_NListtree_Sort_TreeNode_Active
             Sorts the childs of the active entry if a list.

     flags -        Control the part where sorting is done.

         MUIV_NListtree_Sort_Flag_RecursiveOpen
             Sort the list recursive. All open child nodes of the
             node specified in 'listnode' will be sorted too.

         MUIV_NListtree_Sort_Flag_RecursiveAll
             Sort the list recursive with ALL child nodes of the
             node specified in 'listnode'.

RESULT

EXAMPLE

     /* Sort the list of the active node. */
     DoMethod(obj, MUIM_NListtree_Sort,
         MUIV_NListtree_Sort_ListNode_Active, 0);

NOTES

BUGS

SEE ALSO

     MUIA_NListtree_SortHook
```

1.39 NListtree.mcc/MUIM_NListtree_TestPos

```
NAME

     MUIM_NListtree_TestPos -- Get information about entry at x/y pos. (V1)
```

SYNOPSIS

```
DoMethod(obj, MUIM_NListtree_TestPos, LONG xpos, LONG ypos,
    struct MUI_NListtree_TestPos_Result *testposresult);
```

FUNCTION

Find out some information about the currently displayed entry at a certain position (x/y-pos).

This is very useful for Drag&Drop operations.

INPUTS

```
xpos -            X-position.
ypos -            Y-position.
testposresult -    Pointer to a valid MUI_NListtree_TestPos_Result
                  structure.
```

RESULT

```
tpr_TreeNode -    The tree node under the requested position or NULL
                  if there is no entry displayed.
```

The tpr_Type field contains detailed information about the relative position:

```
    MUIV_NListtree_TestPos_Result_Above
    MUIV_NListtree_TestPos_Result_Below
    MUIV_NListtree_TestPos_Result_Onto
    MUIV_NListtree_TestPos_Result_Sorted
```

```
tpr_Column -    The column unter the specified position or -1 if
                no valid column.
```

EXAMPLE

```
/* Entry under the cursor? */
struct MUI_NListtree_TestPos_Result tpres;

DoMethod(obj, MUIM_NListtree_TestPos, msg->imsg->MouseX,
    msg->imsg->MouseY, &tpres);

if ( tpres.tpr_Entry != NULL )
{
    /* Do something very special here... */
}
```

NOTES

BUGS

```
SEE ALSO

     MUIM_NList_TestPos
```