# Label Editing in Tree View and List View Controls

This document covers the following topics:

- Introduction
- Label Editing
- Changing an Item’s Label Programmatically

---

## Introduction

This section describes the process of editing item labels for both tree view and list view controls. The word "item" is therefore used throughout, in place of "tree view item" and "list view item", respectively.

## Label Editing

The editing of an item’s label, unless prohibited (see below), may be initiated in one of three ways:

1. By the user, by clicking on the label of a selected item.
2. By the user, by pressing the F2 key (whereupon the item with the focus rectangle, if any, is edited).
3. By the program, by calling the `EDIT-LABEL` action.

Regardless of the means of initiation, the sequence of actions taken by Natural in response is identical:

1. The control’s `MODIFIABLE` attribute is examined. If this is `FALSE` (e.g., the **Modifiable** option was not checked in the control’s attributes window), no further action occurs and label editing mode is not entered.
2. The control’s `ITEM` attribute is set to the handle of the item for which label editing was requested (the "target" item).
3. Unless suppressed, a `BEFORE-EDIT` event is raised for the control.
4. The target item’s `MODIFIABLE` attribute is examined. If this is `FALSE`, no further action occurs and label editing mode is not entered.
5. Label editing mode is entered. The user may cancel any changes he has made via the ESC key. In this case, the original label is restored, edit mode exited, and no further action taken. Alternatively, the user can commit the changes (e.g. by pressing the ENTER key or setting the focus to another window or control).
6. The target item’s `STRING` attribute is updated with the new label text.

7. Unless suppressed, an AFTER-EDIT event is raised for the control.

8. If the item's label is no longer identical to the item's STRING attribute (i.e., the application modified the attribute during the AFTER-EDIT event), the item's label is updated accordingly.

The purpose of the BEFORE-EDIT event is twofold. Firstly, it allows the application to dynamically set the item's MODIFIABLE attribute (thus allowing or preventing label editing from taking place) according to the particular context. Secondly, it gives the application a chance to save the original label in case it wishes to restore it later in the AFTER-EDIT event.

The AFTER-EDIT event has four options:

1. Do nothing, which case the new item label will be accepted.

2. Reject the new label, by restoring the previous value for the item's STRING attribute (as saved in the BEFORE-EDIT event).

3. Reject the new label, by setting the the item's STRING attribute to some other value that neither matches the new nor old label (e.g. silently "correcting" the label entered by the user).

4. Re-enter edit mode for the item, forcing the user to modify the label again (possibly after displaying a message box to inform the user that the newly entered label is invalid).

As an example demonstrating some of the above topics, consider the following example. Firstly, we define some local data variables which we will need later:

```
01 #CONTROL HANDLE OF GUI 01 #ITEM HANDLE OF GUI
01 #LABEL (A) DYNAMIC 01 #POS (I4)
```

Having done this, we can can write a trivial BEFORE-EDIT event, where we simply save the existing label of the item about to be edited in the dynamic variable #LABEL:

```
#CONTROL := *CONTROL #ITEM := #CONTROL.ITEM #LABEL
:= #ITEM.STRING
```

To illustrate a few of the above techniques, we use the following AFTER-EDIT handler:

```
#CONTROL := *CONTROL #ITEM := #CONTROL.ITEM IF
#ITEM.STRING = ' ' #ITEM.STRING := #LABEL ELSE EXAMINE #ITEM.STRING TRANSLATE
INTO LOWER EXAMINE #ITEM.STRING FOR ' ' GIVING POSITION #POS IF #POS > 0 PROCESS
GUI ACTION CALL-DIALOG WITH #DLG$WINDOW #ITEM 'EDIT-LABEL' FALSE END-IF END-IF
```

The above code performs the following actions:

1. If the new item label consists only of blank, the old item label, as saved in the BEFORE-EDIT event is restored.

2. Otherwise, the new item label is converted into lower case (EXAMINE TRANSLATE).

3. Then, if the new item label contains a blank, we treat the data as invalid and raise an asynchronous user-defined event for the item in order to request corrected data from the user (more later).

The asynchronous event allows an invalid item label to be provisionally accepted. However, on receipt of the user-defined EDIT-LABEL event, we display a message box to inform the user that the data is invalid and in need of correction, then re-enter label editing mode. This is done via the following code in the

DEFAULT event handler for the dialog:

```
IF *EVENT = ’EDIT-LABEL’ #ITEM := *CONTROL OPEN
DIALOG NGU-MESSAGEBOX USING #ITEM.PARENT WITH #BUTTON ’Invalid data - please re-enter’
’Label Edit’ ’!O’ PROCESS GUI ACTION EDIT-LABEL WITH #ITEM GIVING *ERROR END-IF
```

If it is sufficient to set an edit mask and/or maximum label length in characters, and/or specify that only upper case characters should be allowed, this can be achieved without any coding by setting the EDIT-MASK attribute, LENGTH attribute or "Upper case (U)" STYLE flag (respectively) for the appropriate item(s). If an edit mask is specified, Natural automatically restores the old label, issuing a beep (if enabled), if the entered label does not match the mask.

# Changing an Item’s Label Programmatically

An item’s label may be set directly via its STRING attribute. For example:

```
#ITEM.STRING := New label’
```

where #ITEM is the handle of the corresponding tree view or list view item.

In this case, only the item’s edit mask (if any) is used. All other aspects of label editing described above do not apply here. In particular:

1. The label is changed regardless of the value of the MODIFIABLE attribute for the control and the item.
2. No BEFORE-EDIT or AFTER-EDIT events are raised.
3. The control’s ITEM attribute is not set.
4. The text is not automatically translated to upper case if the item’s "Upper case (U)" STYLE flag is set.
5. The supplied label can exceed the limit (if any) imposed by the item’s LENGTH attribute.