SET-CLIPBOARD-DATA Action

This chapter covers the following topics:

- Description
- Parameters

Description

Places data on the local clipboard in the specified Natural data format. If the local clipboard already contains some data in this format, the data is replaced by this call.

Natural supports both pre-defined and private data formats (see below). The advantage of using predefined formats is that they are recognized and understood by many other Windows applications. Pre-defined formats are represented by numeric strings, for which symbolic constants have been defined in the local data area NGULKEY1. For example, the format CF-TEXT indicates standard text format, where the data consists of lines of text separated by carriage return/line feed character pairs. Likewise, the CF-FILELIST format indicates that the data consists of file and/or directory path names, separated by null-terminators (characters with value 0), with an extra trailing null-terminator used to designate the end of the list. Such a file list corresponds to the standard clipboard format used by the Windows Explorer, and many other applications supporting file drag and drop, such as Natural itself. Note that Natural automatically inserts the required delimiters (carriage return/line feed or null-terminator) between the data items between each field (scalar or array element) it writes, as well as the trailer (if any) after the last data item, depending on the format. However, if you use a single large field (e.g. a dynamic alpha variable) to write multiple items (lines of text or file names), you must specify the delimiters between the data items explicitly. Note that if array ranges are used, the array range is implicitly expanded internally and handled as if each array element within the range had been explicitly specified in turn (see example below). Note that, for CF-TEXT and CF-FILELIST formats, the incoming data is automatically converted to alpha format in a manner compatible with the MOVE statement if the data is not already in alpha format.

Private formats are more flexible, but are only understood by Natural. In this case, a copy of the data is stored, without any data conversion, except that any trailing blanks belonging to any alpha operands are removed to save space. Private formats are represented by any string that does not begin with a digit. Both the source and target simply need to know the name of the format to use it in the SET-CLIPBOARD-DATA and GET-CLIPBOARD-DATA actions, respectively. Note that, although the same internal data structure is used for all private formats, the ability to use multiple private formats prevents the data from being interpreted incorrectly. For example, you may wish to transfer both ActiveX tree view node data and list box item data in private format. By specifying distinct formats in each case (e.g. ’TVNODEDATA’ and ’LBITEMDATA’), you can conveniently prevent the list box from attempting to interpret the tree view data, and vice versa. To decide whether or not a paste or drop operation may occur, each control needs to simply pass its respective format string to the INQ-FORMAT-AVAILABLE action, and does not need to inspect the data itsef.

Note that although the local clipboard can contain information in multiple formats at any one time, the data for each format must be written via separate calls to SET-CLIPBOARD-DATA. Furthermore, the data for each format must be specified in a single call to SET-CLIPBOARD-DATA. It is not possible to build up the data for each format bit-by-bit by using separate calls to this action.

The clipboard must be open in order to use this action. If you are using this action to prepare drag-drop data in the BEGIN-DRAG event, you do not need to explicitly open the clipboard, as this has already been done implicitly by Natural. Otherwise (e.g., if preparing data for the Windows clipboard) you need to precede this action with an explicit call to OPEN-CLIPBOARD.

Note that this action does not write data to the Windows clipboard. The data will only be available on the Windows clipboard (and hence visible to other applications) after you issue a subsequent call to the CLOSE-CLIPBOARD action. This is, however, not necessary for drag-drop operations, because the GET-CLIPBOARD-DATA action used by the drop target works preferentially directly on the drag-drop clipboard (the local clipboard in the source process) in this case.

Parameters

Name/Data Type	Explanation
Format (A253)	Input Clipboard Natural data format. Can be standard numerc string for predefined formats (e.g., CF-TEXT) or user-defined string for private formats.
Data (List of any type except handles)	Input Data to be written. Can consist of any number of non-handle scalar and/or array operands, including array index ranges and multidimensional arrays. Dynamic alpha variables are also supported.
Response (I4)	Output Natural error (if applicable).

Example:

```
DEFINE DATA LOCAL
1 #DYN (A) DYNAMIC
1 #ARR (A20/3) INIT<'Line 1', 'Line 2', 'Line 3'>
1 #RESPONSE (I4)
END-DEFINE
*
* NOTE: All three calls to SET-CLIPBOARD-DATA below have the same effect!
*
/* Example 1 - Items specified as individual fields
PROCESS GUI ACTION SET-CLIPBOARD-DATA WITH CF-TEXT #ARR(1) #ARR(2) #ARR(3)
GIVING #RESPONSE
*
/* Example 2 - Items specified as array index range
PROCESS GUI ACTION SET-CLIPBOARD-DATA WITH CF-TEXT #ARR(*)
  GIVING #RESPONSE
*
/* Example 3 - Items specified using dynamic variable/* (note the use of the explicit delimiter)
COMPRESS #ARR(1) END-OF-LINE #ARR(2) END-OF-LINE #ARR(3) INTO #DYN
  LEAVING NO SPACE
PROCESS GUI ACTION SET-CLIPBOARD-DATA WITH CF-TEXT #DYN #ARR(3)
  GIVING #RESPONSE
```