# Ciansoft ClipManagerX User Manual

## Introduction

Ciansoft ClipManagerX is an ActiveX control that collects data from the Windows Clipboard and makes the data available to your application. Using ClipManagerX you can:

- Retain all items that are copied to the clipboard (with the Windows Clipboard only the last item to be copied is available).
- Collect data from any application running on the user's computer.
- Collect data in any clipboard format.
- Configure ClipManagerX to collect only the specific data types that you require, e.g., collect text that is copied, but not images.

For further information, visit our website: www.ciansoft.com.

Alternatively, contact us by email, we will be pleased to answer your questions: info@ciansoft.com.

## How to use these Instructions

The use of ClipManagerX is described under the following headings:

- General (Placing the control on a form, activating and deactivating the control, etc.).
- Configuring the control to collect the required data.
- Retrieving data from the control.

At the end of these instructions is an alphabetical listing of all the methods and properties available. We recommend that you first read the sections of the instructions describing the use of the control, then look in the listing to find the specific syntax for the methods and properties to be used. Links are provided in the main part of the instructions to take you directly to the required method or property.

If you are new to programming with the clipboard, read the section "About Clipboard Formats" which contains useful background information.

## Installation

The OCX file, ClipManagerX.ocx, must be registered on the computer running the application. This should be done during the installation, but if it is not, the command line utility Regsvr32.exe can be used. This is usually found in the Windows system folder and runs using the syntax:

regsvr32 *ocxname*

where *ocxname* is the path and name of the ocx file to register. When deploying an application that uses ClipManagerX, the ocx file will also need to be registered on the target machine. Note that the licence file, ClipManagerX.lic, is needed to use this control in a design environment and this file must not be distributed with an application, as is confirmed in your licence agreement.

To use this control in a Visual Basic project, select Project and Components from the pull down menu. This gives a list of available controls. Select ClipManagerX Library and click Apply. This adds the control to the component palette. The class name is "ClipMan" and this will appear in the Object Browser where the properties and methods are listed. For other design environments, consult the documentation for importing ActiveX controls.

As an ActiveX control, ClipManagerX can be used in a range of Windows based environments and languages. There will be slight differences in syntax especially with object creation and the use of parentheses/brackets surrounding method parameters.

# The Trial Version

The trial version of ClipManagerX is supplied as a different OCX file, called ClipManagerXTrial.ocx, and has a separate installation programme. The trial version has all the functionality of the full version of the control. The only limitation is that it has a trial period of 30 days, starting from the first use of the control. After 30 days of use, the control will be unable to be activated and a message will be displayed explaining that the trial period has ended. Visit www.ciansoft.com to purchase the full version.

# General

The ClipManagerX control is used in an application simply by placing or dragging a copy of the control onto a form. At design time, the control is visible as an icon on the form, but at run time, it will be invisible.

To start collecting data, the control must be activated by calling the Activate method. Once activated, ClipManagerX will continuously watch the Windows Clipboard and will collect data that is copied to the clipboard from any application. This will continue until the Deactivate method is called.

Once it is activated and collecting data, ClipManagerX can collect up to 65535 separate data items, with each of these possibly being stored in more than one format. In practice, this could result in the control using too much of the system resources (memory), so the maximum number of data items to be held at any one time should be specified using the MaxItems property.

When the control is full of data (as defined by MaxItems), new data copied to the clipboard will still be collected, but the oldest data item will be deleted to make space available. The number of data items at any time can be found using the Count property.

All the data held by the control can be deleted by using the Clear method.

Multiple instances of the control can be used simultaneously if required.

# Configuration

The Windows Clipboard can hold data in many different formats. These include standard formats which are defined in the Windows operating system, plus custom formats which are defined by installed applications and which may be able to be read by other applications. A single data item copied to the clipboard can be stored simultaneously in several different formats, with an application that receives the data choosing the most appropriate format to use for reading.

ClipManagerX supports all Windows clipboard formats, both standard and custom.

When ClipManagerX has been activated to receive data, it can be configured to receive and store only those formats required by the user. First, the AllFormats property defines whether all possible formats will be collected. If this property is set to False, then a list of formats stored by the control will determine the behaviour. This list is managed by using the AddFormat , AddFormatByName or AddFormatByNum methods (for adding a new format to the list), the RemoveFormat method (for removing a format from the list) and the ClearFormats method (for emptying the list).

# Data Retrieval

There are four alternative ways to retrieve items of data from ClipManagerX. These are:

- By copying back to the Windows clipboard using the CopyToClipboard method, then pasting from the clipboard.

  In many cases this is the simplest way to retrieve data as it may not require any knowledge of how

the receiving application interprets the data. There is an example of this on the Ciansoft website at http://www.ciansoft.com/samples/cmxword1.htm in which data in Rich Text Format (RTF) is pasted into a MS Word document. It is not necessary for the programmer to know anything about the structure of RTF or know anything about how Word stores its data.

This is the slowest method as it involves the data being copied in memory twice (once from ClipManagerX to the clipboard, once from the clipboard to the application). However, this loss of performance is unimportant in most applications.

- By accessing a read-only property specific to the required data format. These properties are available only for a limited range of possible formats. The available properties are:

| **Format:** | **Property:** |
|---|---|
| Plain Text | Text |
| Unicode Text | UnicodeText |
| Bitmap | BMPPicture |
| Enhanced Metafile | EMFPicture |
| Rich Text | RichText |
| HTML | HTML |

- By using the DataHandle property to obtain a Handle to the data object in memory.

  If ClipManagerX is used in a programming environment which supports pointers, e.g., MS Visual C++ or Borland Delphi, the DataHandle property enables the user to obtain a pointer to an object in memory. In programming environments which do not support pointers, e.g., Visual Basic, the DataHandle property is difficult to use and the WriteBinary method should be used instead.

- By writing binary data to an OLE variant array using the WriteBinary method. The OLE variant created by this method is an array of bytes containing the data.

In all cases, an Index value is referenced to determine which data item is retrieved. The Index value of the most recent item copied is 1, the next most recent is 2, etc.

# Complete Listing of Properties and Methods

**Activate** ( ) - Starts watching the Windows Clipboard and collecting all data copied to the clipboard, subject to the configuration of chosen formats.

**AddFormat** (*Format* As TxClipboardFormat) As Long - Adds a clipboard format, indicated by the value *Format*, to the list of formats that will be collected by the control. The possible values for *Format* are given in the Format Summary Table at the end of these instructions. The list of formats will be updated regardless of the value of *AllFormats*, but if *AllFormats* is True, the list is not used.

This method is a function which returns the integer value used by Windows to reference this clipboard format. This return value should always be assigned to a variable, as it will be needed later when retrieving data from the control.

**AddFormatByName** (*Format* As String) As Long - Adds a clipboard format with the name given by *Format*, to the list of formats that will be collected by the control. The name is not case sensitive, but must be spelled exactly right, including any spaces. The list of formats will be updated regardless of the value of *AllFormats*, but if *AllFormats* is True, the list is not used.

This method is a function which returns the integer value used by Windows to reference this clipboard format. This return value should always be assigned to a variable, as it will be needed later when retrieving data from the control.

**AddFormatByNum** (*Format* As Long) - Adds a clipboard format, indicated by the integer *Format*, to the list of formats that will be collected by the control. *Format* is the clipboard format number used by Windows. For standard clipboard formats this is a fixed number, but for custom clipboard formats it will have been assigned at the time the format was registered on the system, and will vary. The list of formats will be updated regardless of the value of *AllFormats*, but if *AllFormats* is True, the list is not used.

**AllFormats** As Boolean - If set to True, the control will collect data in any of the supported formats. Otherwise, the formats to be used are determined by the format list using the *AddFormat*, *AddFormatByName*, *AddFormatByNum*, *RemoveFormat* and *ClearFormats* methods. (Default = True).

**BMPPicture** (*Index* As Long) As IPictureDisp - Read-only. Returns a Picture of the data held in Bitmap format (Format = cfBitmap) and referenced by the value *Index* (*Index* = 1 corresponds to the item most recently copied to the clipboard, etc.). An empty picture is returned if there is no data held in this format for the given *Index* value.

**Clear** ( ) - Deletes all data from the control.

**ClearFormats** ( ) - Removes all formats from the list of formats that will be collected by the control. The list of formats will be cleared regardless of the value of *AllFormats*, but if *AllFormats* is True, the list is not used.

**CollectOnActivate** As Boolean - If set to True, the control will collect the contents of the clipboard immediately when the *Activate* method is called. Otherwise, nothing is collected until the first copy action that occurs after activation. (Default = False).

**CopyToClipboard** (*Index* As Long, *Format* As Long) - Copies the data item referenced by *Index* and in the specified *Format* back to the Windows clipboard. ClipManagerX is temporarily deactivated during this process, so that a duplicate copy of the data is not collected.

**Count** As Long - Read-only. The number of data items currently stored in the control.

**DataHandle** (*Index* As Long, *Format* As Long) As Long - Read-only. Returns a Handle to the data item referenced by *Index* and in the specified *Format*. *Format* is the Windows clipboard format number as used by *AddFormatByNum* or as returned by *AddFormat* or *AddFormatByName*.

For most formats, the return value is a Handle to a Global memory object (HGLOBAL). This can be accessed by using the Windows API function GlobalLock and related functions. In a few cases, the return value will be a Handle to the specific object type. These cases are:

| Format: | Handle Type: |
|---|---|
| cfBitmap: | HBITMAP |
| cfEnhMetafile: | HMETAFILE |

The return value is zero if the requested data is unavailable, e.g., a format is specified for which no data was collected.

**Deactivate** ( ) - Stops the control from collecting data. Note that calling the *Deactivate* method does not delete any data currently stored in the control. All data will remain available for retrieval, unless the *Clear* method is called.

**DelayMode** As Long - Delay in milliseconds between data being copied to the clipboard and being retrieved by ClipManagerX. Under normal circumstances this should always be set to zero. In this mode, ClipManagerX responds immediately to retrieve data whenever a new item of data is copied to the clipboard.

When using ClipManagerX with Microsoft Word 2000 (and later versions), a crash can occur when deleting text from Word that has previously been copied to the clipboard. This problem can be prevented by setting *DelayMode* to a positive value, for example, 10. This ensures that Word is not

attempting to delete data at the same time that ClipManagerX is reading the same data. This is the only situation in which a non-zero value should be used. (Default = 0).

**EMFPicture** (*Index* As Long) As IPictureDisp - Read-only. Returns a Picture of the data held in Enhanced Metafile format (Format = cfEnhMetafile) and referenced by the value *Index* (*Index* = 1 corresponds to the item most recently copied to the clipboard, etc.). An empty picture is returned if there is no data held in this format for the given *Index* value.

**Enabled** As Boolean - Indicates whether or not the control is currently watching the clipboard and collecting data. Setting the value of *Enabled* is directly equivalent to calling the *Activate* or *Deactivate* methods. (Default = False).

**HasFormat** (*Index* As Long, *Format* As Long) As Boolean - Read-only. Indicates whether or not the data item referenced by *Index* is available in the specified *Format*.

**HTML** (*Index* As Long) As String - Read-only. Returns a text string of the data held in HTML Format (Format = cfHTML) and referenced by the value *Index* (*Index* = 1 corresponds to the item most recently copied to the clipboard, etc.). An empty string is returned if there is no data held in this format for the given *Index* value.

**ImportBinary** (*Data* As Variant, *Format* As Long) - Copies the contents of the variant array *Data* as a new data item to the control as if it had been collected from the clipboard. The data format is given by *Format*.

**ImportHTML** (*HTMLData* As String) - Copies the character string *HTMLData* as a new data item to the control in HTML Format, as if it had been collected from the clipboard. By using this method in conjunction with *CopyToClipboard*, ClipManagerX can be used to transfer the contents of a string variable to the clipboard for pasting into an application.

**ImportRichText** (*RTFData* As String) - Copies the character string *RTFData* as a new data item to the control in Rich Text Format, as if it had been collected from the clipboard. By using this method in conjunction with *CopyToClipboard*, ClipManagerX can be used to transfer the contents of a string variable to the clipboard for pasting into an application.

**ImportText** (*TextData* As String) - Copies the character string *TextData* as a new data item to the control in plain Text Format, as if it had been collected from the clipboard. By using this method in conjunction with *CopyToClipboard*, ClipManagerX can be used to transfer the contents of a string variable to the clipboard for pasting into an application.

**MaxItems** As Long - Sets the maximum number of data items that can be stored in the control. This must be set to a value between 1 and 65535. If a new data item is collected which would cause *Count* to exceed *MaxItems*, the oldest data item is deleted to make space. (Default = 32).

**RemoveFormat** (*Format* As Long) - Removes a clipboard format, indicated by the integer *Format*, from the list of formats that will be collected by the control. The list of formats will be updated regardless of the value of *AllFormats*, but if *AllFormats* is True, the list is not used.

**RichText** (*Index* As Long) As String - Read-only. Returns a text string of the data held in Rich Text Format (Format = cfRichText) and referenced by the value *Index* (*Index* = 1 corresponds to the item most recently copied to the clipboard, etc.). An empty string is returned if there is no data held in this format for the given *Index* value.

**Text** (*Index* As Long) As String - Read-only. Returns a text string of the data held in Plain text format (Format = cfText) and referenced by the value *Index* (*Index* = 1 corresponds to the item most recently copied to the clipboard, etc.). An empty string is returned if there is no data held in this format for the given *Index* value.

**UnicodeText** (*Index* As Long) As String - Read-only. Returns a Unicode text string of the data held in this format (Format = cfUnicodeText) and referenced by the value *Index* (*Index* = 1 corresponds to the item most recently copied to the clipboard, etc.). An empty string is returned if there is no data

held in this format for the given *Index* value. The String returned is actually in the OLE data type BSTR, which is a string of double-byte characters. Note that some programming environments are not Unicode compliant and will interpret the string incorrectly as single-byte ANSI characters, thereby losing information.

**WriteBinary** (*Index* As Long, *Format* As Long) As Variant - Writes the data held in the specified *Format* and referenced by the value *Index* as binary data to an OLE Variant array. This array contains a sequence of bytes corresponding to the actual memory contents. If the requested format or index is not available, the return value will be 0. This method is a function and must be called by assigning to a variable. For example, in VB, the syntax would be of the form:

```
VariantName = WriteBinary (4, 2)
```

This would write the 4th oldest data item to a variant array using the Bitmap (2) format.

# Events

The following events are raised by the control:

**OnNewData** ( ) - This event occurs when the control reads a new item of data from the clipboard.

**OnClipboard** ( ) - This event occurs when any data is copied to the clipboard from any application, regardless of whether or not ClipManagerX collects that data.

# About Clipboard Formats

Most Windows applications provide some kind of cut/copy/paste functionality that makes use of the clipboard. Whenever a piece of data is copied, it is stored on the clipboard and can then be pasted into a new location in either the same or a different application. Copying data to the clipboard erases any existing data, so the clipboard can only ever hold the last item copied.

The copied data can be anything – text, a picture, a sound clip, etc. Depending on the type of data, one of many different "clipboard formats" will be used to store the data. An application placing data on the clipboard can use many different formats so that the application receiving the data has many choices for how the data can be read.

Windows operating systems support several standard clipboard formats. These mainly cover common data structures such as plain text, bitmap images, etc. In addition, custom formats can be defined by an application. Some of these formats will be very specialised, but many are common and are used by many different applications, e.g., rich text format used by word processors. Despite being common, this is still a custom format, meaning that its definition is not part of the core Windows operating system.

When programming with the clipboard, each format is referred to by an integer value which identifies it. For the standard formats, these values are fixed. For the custom formats, the value is assigned by the operating system when an application first registers the format, so the value can be different every time the format is used on a different computer, or even on the same computer at a different time.

# Format Summary Table

This table summarises the standard Windows clipboard formats and also some of the commonly encountered custom formats. It is not a complete list of formats supported by this control, as ClipManagerX will support any clipboard format.

The following information is included in the table:

- Description:
- Windows Name (Standard formats only): The name used by the Windows API for this format. This name may be recognised in some programming environments, allowing it to be used in place of the integer value.
- Integer Value (Standard formats only): The value corresponding to the Windows API name. This is the value that must be passed as an argument in any properties or methods dealing with formats, e.g., the AddFormatByNum method. Note that custom formats also have an integer value, but this is not fixed and must be obtained by calling either the AddFormat or AddFormatByName method.
- TxClipboardFormat: The enumerated constant used by ClipManagerX to identify this format.
- The integer value corresponding to the TxClipboardFormat enumeration. Note that this must not be confused with the integer value used by Windows even though for standard formats these are the same.
- Property: The name of the ClipManagerX property that provides direct access for retrieval of data in this format.
- Format Name (Custom formats only): The string used to identify this format. This is needed if the AddFormatByName method is to be used.

**Standard Formats:**

| Description: | Windows Name: | Integer Value: | TxClipboardFormat: | | Property: |
|---|---|---|---|---|---|
| | | | Constant: | Value: | |
| Plain text | CF_TEXT | 1 | cfText | 1 | Text |
| Bitmap | CF_BITMAP | 2 | cfBitmap | 2 | BMPPicture |
| Windows Metafile | CF_METAFILEPICT | 3 | cfMetafile | 3 | |
| Symbolic Link Format (SYLK) | CF_SYLK | 4 | cfSYLK | 4 | |
| Data Interchange Format (DIF) | CF_DIF | 5 | cfDIF | 5 | |
| Tagged-Image File Format (TIFF) | CF_TIFF | 6 | cfTIFF | 6 | |
| OEM Text | CF_OEMTEXT | 7 | cfOEMText | 7 | |
| Device-Independent Bitmap | CF_DIB | 8 | cfDIB | 8 | |
| Colour Palette | CF_PALETTE | 9 | cfPalette | 9 | |
| Pen extensions | CF_PENDATA | 10 | cfPenData | 10 | |
| Audio Data (RIFF) | CF_RIFF | 11 | cfRIFF | 11 | |
| Audio Data (WAVE) | CF_WAVE | 12 | cfWAVE | 12 | |
| Unicode Text | CF_UNICODETEXT | 13 | cfUnicodeText | 13 | UnicodeText |
| Enhanced Metafile | CF_ENHMETAFILE | 14 | cfEnhMetafile | 14 | EMFPicture |
| File List | CF_HDROP | 15 | cfHDROP | 15 | |
| Locale Identifier | CF_LOCALE | 16 | cfLocale | 16 | |

| Custom Formats: | | | | |
|---|---|---|---|---|
| **Description:** | **Format Name:** | **TxClipboardFormat:** | | **Property:** |
| | | **Constant:** | **Value:** | |
| Rich Text Format (RTF) | “Rich Text Format” | cfRichText | 18 | RichText |
| HTML Format | “HTML Format” | cfHTML | 19 | HTML |

# Deploying an Application

In order to deploy an application that uses ClipManagerX you will need to distribute the OCX file, ClipManagerX.ocx, together with the files that make up your application. This file will need to be registered on the machine running your application and you may wish to use a proprietary installer to do this. When ClipManagerX was installed on your system our installer will have copied the OCX file to the directory "Program Files\Ciansoft\ClipManagerX\", assuming you used the installer and accepted the defaults.

The number of copies of the OCX file that may be distributed is not limited by the licence. In order to use the control in a design environment the licence file, ClipManagerX.lic, is also required. A separate licensed copy of ClipManagerX is required for each computer it is used on in the design environment.

# Revision History

The current version of ClipManagerX is 2.1.

New in Version 2.0

Support for all standard and custom clipboard formats.
CopyToClipboard method.
DataHandle property.
RichText property.
TxClipboardFormat enumeration for use in AddFormat method.
Methods AddFormatByNum and AddFormatByName.

New in Version 2.1

HTML property.
CollectOnActivate property.
ImportRichText, ImportHTML, ImportText and ImportBinary methods.