The Motif Clipboard

Motif provides a set of routines for dealing with the CLIPBOARD selection. Although these routines are not obsolete, applications should use UTM routines, instead of these clipboard routines, whenever possible.

The Motif clipboard interface allows an application to assert ownership of the selection and request conversion of the selection. The interface stores the data in the selection and other information about the selection on the server. The owner can place the selection value in these server data structures either at the time it asserts ownership or at the time a client requests conversion.

By copying the selection value at the time it asserts ownership, an application can simplify conversion and make the data available for retrieval even if the owner is killed. By copying the selection value when a client requests it, an application can avoid converting data that no client may request. However, in this case, the application may need to make a copy of the data to be transferred. With either copying mechanism, the data is stored in the Motif clipboard's server data structures the first time a client requests the data.

Copying Data to the Clipboard

To assert ownership and copy data to the clipboard, an application takes these steps:
1.It calls **XmClipboardStartCopy** to begin the interaction.
2.It makes one or more calls to **XmClipboardCopy** to place data on the clipboard.
3.It terminates the interaction by calling **XmClipboardEndCopy** or **XmClipboardCancelCopy**.

An application begins an interaction to copy data to the clipboard by calling **XmClipboardStartCopy**. The application passes the following: a display pointer and timestamp; the ID of a window in the application; a compound string that could be used to label the data; and, if the application intends to delay copying the data until it is requested, a widget ID and a function to be called to convert the data. **XmClipboardStartCopy** returns in one of the arguments a data ID that the application must later pass to **XmClipboardEndCopy** or **XmClipboardCancelCopy**. The application must also pass the same window ID to subsequent clipboard calls in this sequence that it uses in the call to **XmClipboardStartCopy**.

After calling **XmClipboardStartCopy**, the application makes one or more calls to **XmClipboardCopy** to place data on the clipboard. Each call associates the data with a single target (called a format in the clipboard interface). The application can associate the same data or different data with more than one target, but it must do so by making separate calls to **XmClipboardCopy**.

If the application passes a NULL data buffer to **XmClipboardCopy**, it asserts that it intends to transfer the actual data for that target when a client requests it. Otherwise, **XmClipboardCopy** transfers data to be stored on the clipboard by **XmClipboardEndCopy**. If the application makes more than one call to **XmClipboardCopy** for the same target, the data is appended to the previously transferred data for that target.

XmClipboardCopy returns in one of its arguments a data ID that identifies the data and target specified in this call. An application that provides actual data at the time a client requests it uses this ID in its conversion routine to identify the data and target to be converted. Such an application must store a mapping of the data ID to the data and target after **XmClipboardCopy** returns.

The application terminates the interaction by calling either **XmClipboardEndCopy** or **XmClipboardCancelCopy**. **XmClipboardEndCopy** stores in the server data structures the data transferred by the calls to **XmClipboardCopy** during this interaction sequence. It also asserts ownership of the CLIPBOARD selection. If the application calls **XmClipboardCancelCopy** instead of **XmClipboardEndCopy**, the interaction is terminated without storing any of the transferred data or asserting ownership of the selection.

If a client later requests data that the owner has declared it would provide at the time of the request, the clipboard interface invokes the conversion routine that the owner registered in the call to **XmClipboardStartCopy**. This routine receives the following as arguments: the widget ID passed to **XmClipboardStartCopy**; the data ID for this data and target returned by **XmClipboardCopy**; a private ID the application may have supplied in the call to **XmClipboardCopy**; and a reason for invoking the routine.

The conversion routine is responsible for converting the data to the requested target. In order to do this it must consult the mapping it established between the data ID or the private ID and the data and target when it called **XmClipboardCopy**. Once the conversion routine has determined the proper target, it copies the data to the clipboard. To do this it calls **XmClipboardCopyByName**, using the data ID passed to the conversion routine. The application can call **XmClipboardCopyByName** more than once, if necessary, to convert all the data for this target.

Once an application has copied data to the clipboard in this way, it no longer asserts that it will convert the same data to the same target in the future. It can remove the data ID from its mapping of data IDs to data and targets, and it can free any data it has associated with this ID if it is not needed for any other purpose.

The clipboard interface calls the conversion routine when a data item intended for later conversion has been removed from the clipboard and is no longer needed. For example, another application may have copied new data to the clipboard. In this case, the conversion routine can remove the data ID from its mapping of data IDs to data and targets, and it can free any data it has associated with this ID if it is not needed for any other purpose. If the conversion routine is being called because an item has been removed from the clipboard, the *reason* argument to the conversion routine is **XmCR_CLIPBOARD_DATA_DELETE**. If the conversion routine is being called because a client has requested data conversion, the *reason* argument is **XmCR_CLIPBOARD_DATA_REQUEST**.

An application can use **XmClipboardWithdrawFormat** to rescind its assertion that it will convert data to a particular target on request.

XmClipboardUndoCopy removes the last item placed on the clipboard by an application using the same *display* and *window* arguments. This function also restores to the clipboard the item that was on the clipboard before the cancelled copy was done. If the application calls **XmClipboardUndoCopy** a second time, the function restores to the clipboard the item that was removed by the first call to **XmClipboardUndoCopy**.

Retrieving Data from the Clipboard

To retrieve data from the clipboard, an application takes these steps:
1.It calls **XmClipboardStartRetrieve** to begin the interaction.
2.It makes one or more calls to **XmClipboardRetrieve** to retrieve data from the clipboard.
3.It terminates the interaction by calling **XmClipboardEndRetrieve**.

An application begins an interaction to retrieve data from the clipboard by calling **XmClipboardStartRetrieve**. The application passes a display pointer, a timestamp, and the ID of a window in the application. The application must pass the same window ID to subsequent clipboard calls in this sequence that it uses in the call to **XmClipboardStartRetrieve**. **XmClipboardStartRetrieve** locks the clipboard.

After calling **XmClipboardStartRetrieve**, the application makes one or more calls to **XmClipboardRetrieve** to retrieve data from the clipboard, converted to a given target. The application passes **XmClipboardRetrieve** a buffer to receive the data. If this buffer is not large enough to contain all the data for the given target, **XmClipboardRetrieve** returns **XmClipboardTruncate**. The application can make repeated calls to **XmClipboardRetrieve** to retrieve the remainder of the data. The function **XmClipboardInquireLength** returns the length of the data on the clipboard for the given target. This allows the application to allocate a buffer of the correct size.

XmClipboardEndRetrieve unlocks the clipboard and ends the interaction.

Utility Routines

The Motif clipboard interface has routines to lock and unlock the clipboard, to make inquiries about its contents, and to register new targets.

XmClipboardLock prevents another application from gaining access to the Motif clipboard. **XmClipboardUnlock** allows other applications to gain access. The clipboard interface automatically locks the clipboard during calls to **XmClipboardStartRetrieve** and **XmClipboardEndRetrieve**. At other times, an application can use **XmClipboardLock** and **XmClipboardUnlock** to lock the clipboard explicitly.

The clipboard interface includes four routines for making inquiries about the clipboard contents:

- **XmClipboardInquireCount** returns the number of targets for which data exists on the clipboard.
- **XmClipboardInquireFormat** returns the name of the target for a given index of targets on the clipboard. An application could retrieve the names of all the targets associated with data on the clipboard by first calling **XmClipboardInquireCount** to find out how many such targets exist and then calling **XmClipboardInquireFormat** with indices from 1 to the number of targets, inclusive. Note that the first index for **XmClipboardInquireFormat** is 1, not 0.
- **XmClipboardInquireLength** returns the number of bytes of data associated with a given target on the clipboard.
- **XmClipboardInquirePendingItems** returns a list of pairs of data ID and private ID for a given target if that target exists on the clipboard and if the owner has asserted that it will supply the actual data on request (but has not yet done so).

An application that makes more than one call to an inquiry function at a time should use **XmClipboardLock** and **XmClipboardUnlock** to lock the clipboard for the duration of the interaction.

XmClipboardRegisterFormat registers a new target with the clipboard interface. The application supplies the length of the data in bits along with the name of the target so that the correct byte order will be maintained when transferring data across platforms. All targets defined in ICCCM are preregistered; the application does not have to call **XmClipboardRegisterFormat** for these.