SideClick API Specification
Version 0.6
October 24, 2004
by Eric Shepherd

Using SideClick's API

SideClick provides a simple interface for adding items to its contextual menu. This is accomplished by querying clients when the user control-clicks. Items are only added to the contextual menu at that time.

When the user control-clicks, SideClick sends out a `contextMenuSaysClicked` message to all interested parties. If an application (or desk accessory, or whatever) decides based on the context of the click that it wishes to add one or more items to the contextual menu, it responds by sending one `tellContextMenuAddItem` request for each item it wishes to add.

The contextual menu only exists for the duration of the user's interaction with it—it's created when the user control-clicks, and is deleted when the user releases the mouse button. This allows the menu to be different with every click, depending on what software is running, or on the context in which the click occurs.

For example, a spelling checker might receive `contextMenuSaysClicked`, and look at the `CMClickInfoRec.control` field to see if the click occurred within a TextEdit control. If so, the spell checker could highlight the word at the cursor position as indicated by `CMClickInfoRec.event.where`, then respond with `tellContextMenuAddItem` to add a "Check Spelling..." option to the contextual menu.

If the spelling checker then receives a `contextMenuSaysItemSelected` message indicating that the "Check Spelling..." option was chosen, it should perform the spell check. If it receives a `contextMenuSaysNeverMind` message, the spell checker knows that it shouldn't check the spelling, and might, for example, restore the previous selection.

Constants

`cmRequestString`
The string used to send requests to SideClick.

```
#define cmRequestString "\pSyndicomm~SideClick~"
```

`CM_CURRENT_VERSION`

Identifies the current version of the SideClick Contextual Menu Manager at compile time.

```
#define CM_CURRENT_VERSION 0x0000
```

IPC Calls Sent to SideClick

`askContextMenuAreYouThere`

Asks if SideClick is available. Reports an error if it's not. Typically you won't need to call this, since the only messages you send to SideClick are sent in response to messages it sends to you first. However, it's available if you need it for some reason.

```
#define askContextMenuAreYouThere 0x8100
```

There are no inputs or outputs.

`tellContextMenuAddItem`

Adds an item to the SideClick menu. Clients should call this in response to the `contextMenuSaysClicked` call if they wish to add items to the contextual menu.

A client may add multiple items if it wishes to do so, by simply sending this request repeatedly. The client may use any flags or other item options it wishes, but may **not** use keyboard equivalents.

```
#define tellContextMenuAddItem 0x8101
```

The `dataIn` parameter should be a pointer to a structure like this:

```
typedef struct CMMenuItemRec {
      struct CMMenuItemRec *previous;
      struct CMMenuItemRec *next;
      unsigned int version;
      unsigned int userID;
      unsigned int itemID;
      MenuItemTemplate *itemTemplate;
} CMMenuItemRec;
```

`previous`
: Reserved for use by SideClick.

`next`
: Reserved for use by SideClick.

`version`

The value of `CM_CURRENT_VERSION` corresponding to the version of the record being passed.

`userID`
The Memory Manager user ID of the program to receive notification when the item is selected by the user.

`itemID`
An ID for the menu item, which must be unique for the specified `userID`; because SideClick has to assign new menu item IDs to the items as they're added to the contextual menu, it's this ID that will be used to identify the item when it's selected.

A client can add as many items as it wishes to the menu by specifying a unique ID for each item. The combination of user ID and item ID will uniquely identify a given menu item.

`itemTemplate`
A pointer to a Menu Manager menu item template for the item to add to the menu.

IPC Calls Sent By SideClick

`contextMenuSaysClicked`

Sent by SideClick to every request procedure when the user control-clicks. Clients can respond by adding an item to the contextual menu (using `tellContextMenuAddItem`) if they wish to do so.

```
#define contextMenuSaysClicked 0x3681
```

The `dataIn` parameter when this is received will be a pointer to a structure like this:

```
typedef struct CMClickInfoRec {
     unsigned int version;
     EventRecordPtr event;
     WindowPtr window;
     CtlRecHndl control;
} CMClickInfoRec;
```

`version`
The value of `CM_CURRENT_VERSION` corresponding to the version of the record being passed.

`event`
An event that provides information about where (in global coordinates) the

mouse was clicked, what keys are held down at the time of the click, and so forth.

`window`
: A pointer to the window record of the window the mouse was clicked in. `NULL` if the click isn't in a window.

`control`
: Handle of the control the click occurred in, or `NULL` if it wasn't in a control.

The recipient can use the passed event, window, and control information to determine whether or not to add items to the menu, and to make decisions on exactly what items to add.

contextMenuSaysItemSelected

Sent by SideClick when a menu item is selected from the contextual menu. The recipient should act upon the selection as appropriate.

```
#define contextMenuSaysItemSelected 0x3682
```

`dataIn` contains the item number of the selected menu item (specified by the `itemID` field of the `CMMenuItemRec` when the item was added to the menu).

contextMenuSaysNeverMind

Sent by SideClick when the contextual menu is dismissed without selecting an option, or if an option is selected corresponding to another client.

For example, if your application adds an option to the contextual menu when `contextMenuSaysClicked` is received, but another application's item is selected by the user, your application will receive this message. This allows your application to cancel any special actions you may have taken in preparation for handling the a possible command.

```
#define contextMenuSaysNeverMind 0x3683
```

`dataIn` and `dataOut` are not used and should be `NULL`.

Note: Handling `contextMenuSaysNeverMind` is optional. You only need to support it if you need to cancel a pending operation of some kind of the menu is dismissed without selecting one of your menu items.