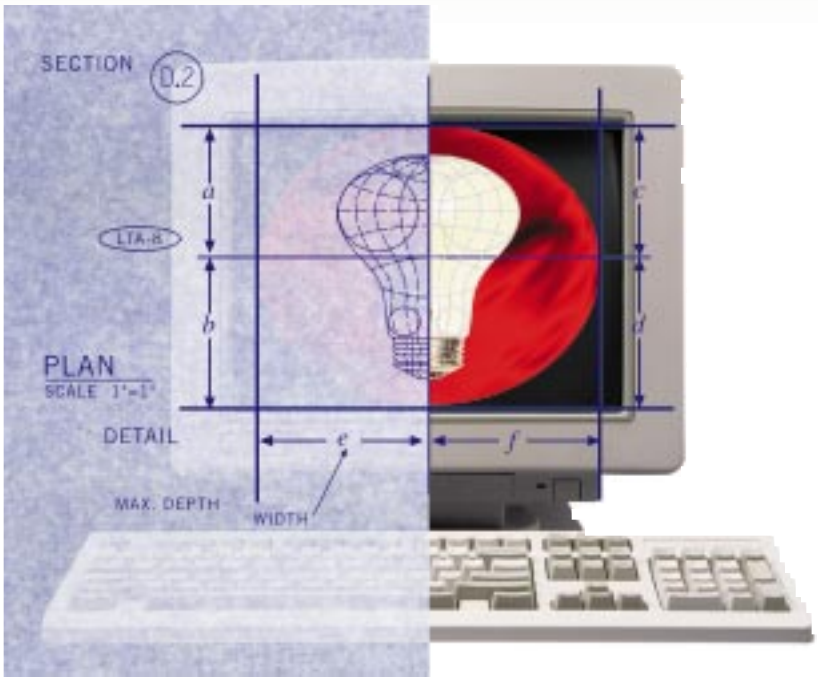
by Peter Collinson, Hillside Systems



MICHELLE FRIESENHAHN WILBY

Customizing CDE

A year ago, I wrote an article entitled "The Common Desktop Environment" (June 1996, Page 22) in which I discussed the basics of the Common Desktop Environment (CDE). I've been meaning to do a follow-up article on CDE actions ever since then. As a key part of CDE, the Action mechanism is used to translate mouse clicks on objects sitting on the screen into running applications doing the work that you need.

Actually, as a beginning user, you are not really aware of the Action system, even though it underpins much of what is happening in the CDE. If you click on the front-panel icon that is a file cabinet drawer with some tools peeking out, you are presented with a window titled *Application Manager*. The Application Manager seems somewhat unspectacular. It looks exactly the same as the regular file browser, so you may not think much of it. Well, it is the file browser, but it's opened at a "secret" directory that was created when your CDE session started. This directory contains a set of symbolic links that collect pointers to several directories, each holding a set of actions.

On the default system setup, the Application Manager window will contain five folders: `Desktop_Apps`, `Desktop_Tools`, `Information`, `OpenWindows` and `System_Admin`. If you look in the `Desktop_Apps` folder, you'll find some icons that you will have seen on the CDE front panel. There are also some icons that are not loaded into the default setup.

The `Desktop_Tools` folder contains many icons: Several map onto familiar UNIX commands; several are unfamiliar and new. Some of the new actions provide you with information about aspects of the CDE; others can be used to manage your CDE environment.

The `OpenWindows` folder contains actions that invoke OpenWindows commands, many of which have been replaced by CDE commands. However, this folder allows you to access the older code should you wish to use it. The `Information` folder should have an icon that starts the online Answerbook documentation and will contain some text files. The `Information` and `System_Admin` folders are intended to be used by local systems administrators to provide you with local information and commands.

If you want to use any of the applications that you will find under the Application Manager, such as the Calculator, then you can double-click on the appropriate icon to start the program running. Some of the applications take arguments and may give you a dialog box so you can supply appropriate text. You can also drag-and-drop a file onto an icon to start the program running with the dropped file as an argument.

If you are a frequent Calculator user, then you might want to place the icon in a more accessible place. One alternative is to drag the icon onto the desktop and park it somewhere convenient. On the desktop, the icon behaves as before, so you can double-click or drag-and-drop as appropriate.

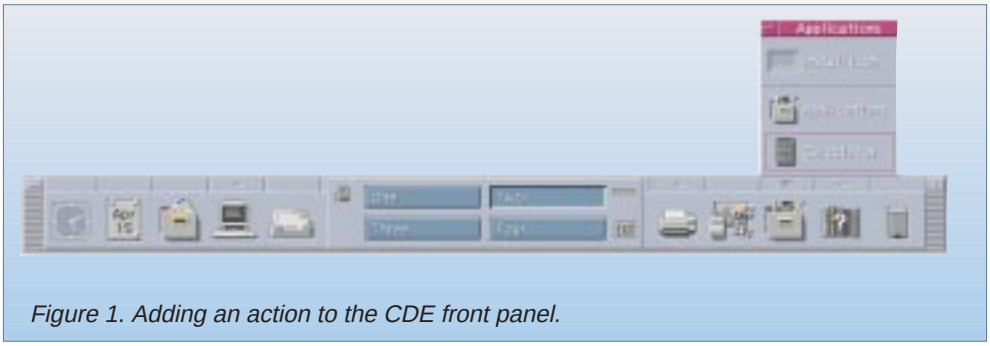


Figure 1. Adding an action to the CDE front panel.

Another alternative is to place the icon into the front panel. By now, you will have discovered that each section in the front panel can have a pop-up menu (see Figure 1). You can add a pop-up menu to an unused section of the front panel by clicking with the right mouse button on the icon sitting in that area of the front panel and then selecting `Add Subpanel`. Each pop-up menu has a drop-zone at the top, titled `Install Icon`, and dropping an application icon into the drop-zone will add the action to that menu. The icon is added to the bottom of the list of actions in the menu, so if you want to impose some ordering on the contents, you have to be circumspect when building the menu.

Once you have more than one icon installed on a menu, you can choose which one is displayed as the "top icon" on the front panel by clicking the right button on the item when the menu is popped up. So the applications that you use most can be just a click away. I've started this process in Figure 1. I've added a subpanel to the `Applications` section and loaded the Calculator into it.

Adding Actions

Actions consist of two files: an *Action Definition File* and an *Action File*. The Action Definition File contains a text specification giving the name and properties of the action. The file is always named *something*`.dt`. CDE recognizes the file by looking for the name. The Action Definition File can hold several actions and may also house specifications for any data files that are associated with the action. These specifications define a *datatype*. Double-clicking on a file that matches the criteria given in a datatype can then invoke the appropriate action.

The contents of the Action File are irrelevant. If you look, it's usually a shell script that echoes a helpful message when run in the shell. The Action File is there to provide a name in the file system that has the same name as an Action Definition. When the Action File is selected using a double-click in a CDE application, its file name is used to invoke the associated action.

You can invoke actions from the standard shell by using the `dtaction` command. It occurs to me that the Action File could use the command so that the file would become useful in the shell as well as CDE. It seems possible to add

```
dtaction `basename $0` $*
```

to the end of the Action File. The idea seems to work, and I don't know why it's not done by default.

The standard place to find the system Action Definition Files is `/usr/dt/appconfig/types/C`. The set of Action Files that map onto entries in the Application Manager is `/usr/dt/appconfig/appmanager/C`. The "C" here is the locale.

You may have some local systemwide Action Definitions stored in `/etc/dt/appconfig/types`, with associated Action Files stored in a subdirectory of `/etc/dt/appconfig/appmanager`. In general, local systemwide aspects of the CDE can be stored in appropriate locations in `/etc/dt`, so you can make local changes without compromising the ability to update the CDE main tree at a later date.

When you define your own personal actions, their definition files will be stored in `~/.dt/types` in your home directory (I am using the standard tilde notation to indicate your home directory). You can also create subdirectories in `~/.dt/appmanager` that will hold your private action files. You will recall that I said the Application Manager is an instance of the File Manager that is started looking at a directory of symlinks that point to application directories. Well, the CDE start-up code will look in the `/etc/dt/appmanager` and your private `~/.dt/appmanager` directory to find local applications and will provide a symbolic link to any directories that it may find there, so local and private applications will be available to you in a coherent folder structure in the Application Manager.

The easiest way to create your own actions is to use the `Create Action` tool that you will find in the Application Manager's `Desktop_Apps` folder. The command can only make simple Action Definition Files, but I've found that it's a good way to start. Figure 2 shows me creating an action to run the Netscape program. I've typed in the name of the action, the command that will be used to run the action, and some text that will be invoked when the `Help` menu is selected for the action. I've extended the window to show the `Advanced` settings. These are not used at the moment.

I've selected a set of icons that will be used. You don't have to create a new bitmap set from scratch for every application. The system comes equipped with many bitmap sets that can often give you something that is appropriate for the task in hand. However, for Netscape, using its own icon is the right thing to do, and I've made my own icon set.

CDE uses four sizes of icons in different circumstances, depending on context and also the resolution of your screen. Consult your documentation for details on sizes and the naming convention that you must follow. I created the Netscape icon set by taking the Netscape icon from a screenshot, editing the bitmap down to size with the Image Viewer and then creating icons with the Icon Editor.

The four bitmap files are stored in the `~/.dt/icon` using the standard icon naming convention. Actually, you'll find that the Icon Editor cannot see a directory called `~/.dt`. It ignores "hidden" directories, which is somewhat inconvenient. I have a

symbolic link called `icons` in my home directory to sidestep this problem.

As you can see, setting up a simple action is trivial. Selecting `Save` from the `File` menu will save the action. Two files are created: `Netscape.dt` is the Action Definition File and will be placed in your `~/.dt/types` directory. The Action File, `Netscape`, will be put in your home directory. Once you have finished creating and testing, you can move it to your `~/.dt/appmanager/personal` directory. You should do the move before you incorporate the action into the front panel, because the front panel definition file contains the pathname of the Action File.

To ensure that the action is registered, you need to find the `Reload Actions` icon in the Application Manager. When I'm developing actions, I drag the icon onto the desktop for convenience. I find that I use it several times, and having it in a handy place saves time. Also, I seem to get buggy side effects with the icons on the front panel when I use the `Reload Actions` tool. Using the `Reload Workspace` option on the desktop right-button menu usually fixes the problem. I tend to reload the workspace or log out when I have finished the development process. Also, I have a SPARCstation 2 (once considered a really fast machine), and it can take some time for a `Reload Actions` to percolate around the various open windows. So remember, on some machines, patience is a virtue.

Because the new Action File is stored in your home directory, you will now have to start the File Manager and, hopefully, you will see the file sporting the Netscape icon. Double-clicking on the icon will start the program.

Before moving on, let's look at the contents of `~/.dt/types/Netscape.dt`:

```
ACTION Netscape
{
   LABEL          Netscape
   TYPE           COMMAND
   EXEC_STRING    /usr/local/bin/netscape
   ICON           netscape
   WINDOW_TYPE    NO_STDIO
   DESCRIPTION    Run Netscape Navigator
}
```

There's actually a bunch of comments before the text, giving you dire warnings about editing this file by hand. The first line provides the action name, and the braces enclose various attributes of the action. The attributes are fairly self-explanatory and are derived directly from the `Create Action` GUI.

Arguments

Well, the action I've just made may seem good enough to you. However, it would be nice to be able to open Netscape with a URL that you type in. Also, more and more documentation is being distributed as HTML files, and you may want to run Netscape with an argument that is an HTML file. Before we start looking at this possibility, change into `~/.dt/types` and save the `Netscape.dt` file by copying it to some other name. We'll use its contents later.

Figure 2. To create an action to run the Netscape program, I've typed in the name of the action, the command that will run the action, and some Help text for that action.

All you need to do to give the action an argument is to change the `Command when Action is Opened` field to

```
/usr/local/bin/netscape $1
```

and save the action. Execute `Reload Actions` and wait for the activity light to stop flashing. Patience. Now you'll find that if you double-click on the Netscape icon, you should see the program run as before. Also, if you find an HTML file and drop it onto the icon, the page will be displayed.

Take a look at the Action Definition File. You'll notice that the command line has changed to

```
EXEC_STRING   /usr/local/bin/netscape %Arg_1%
```

I found the use of dollar variables very confusing at first. The `Create Action` GUI translates the `$1` into the internal form used by the CDE Action system. If you are creating actions by hand, then you must use the correct internal form. We'll see why there is a translation process when we look at the next step.

The action we have created doesn't do quite what we want. We'd like to be able to ask the user for a URL or a file. They will type the name into a dialog box. To provide a dialog box with a prompt, we fill in the `When Action Opens` box in the

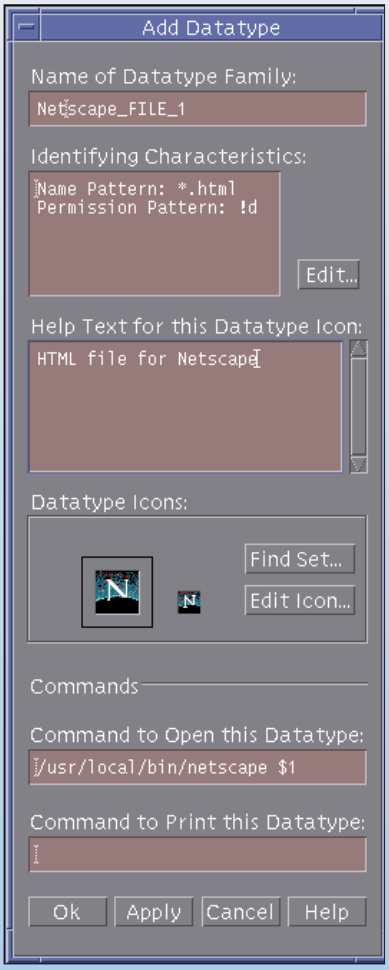


Figure 3. We can add datatypes to the current action by clicking the Add button in the Create Action GUI. The Add Datatype dialog box will pop up (shown here).

`Advanced` section of the GUI. Type `URL:` into the box. Again we will save the file, reload the actions and wait for all that system stuff. Now when we double-click on the icon, we'll get a dialog box that shows the prompt string. If you type the name of a file containing HTML into the box, you will get Netscape to start looking at that file. You can achieve the same result if you drag-and-drop a file onto the Netscape icon.

The command line in the Action Definition File now looks like

```
EXEC_STRING  /usr/local/bin/netscape %Arg_1"URL:"%
```

and you can see how the `Create Action` GUI has massaged the `$1` into the correct internal form. The string in the quotes is the prompt that we supplied.

What happens if we double-click on the icon and type a URL into the dialog box? Here, we hit a snag. With the argument specification shown above, the system assumes we are dealing with a file. It will attempt to expand the string to a full pathname and will object when it cannot find the path. CDE expands file arguments to full pathnames and so avoids having to specify files relative to the current working directory of the application. The application's working directory is usually very different from the working directory of the user. So it's hard to deal with an argument that may be either a file or a URL. There are solutions to the problem, but they have other side effects. As we shall see, perhaps we are prepared to live without the ability anyway.

Datatypes

We now have an action that is represented by an icon on the screen. We can start the application with an argument by dropping a suitable file onto the icon. We can double-click on the icon and start the program with a file, supplying the file name in a dialog box. We can run it with no arguments by selecting `OK` on the dialog box when its text field is empty. However, it is a nice touch to be able to wander around the file system, see an `.html` file and double-click on it to start Netscape looking at that file. Providing clickable files is done by creating a datatype that associates actions with a file type.

We can add datatypes to the current action by clicking the `Add` button in the `Create Action` GUI. The `Add Datatype` dialog box will pop up (shown in Figure 3). The top entry, the name, is filled in automatically, but we can change it if we like. I've filled in the `Identifying Characteristics` box by hitting the `Edit` button and filling in the dialog screen that is produced. The system provides several different methods for identifying a file type. We can match a name using shell file name expansion expressions, which is what I have done here. We can also select files or directories, look at the access permissions set on the file, or even examine the contents of the file.

Next, we select an icon to be used when a file of this type is identified by the File Manager. I've used the Netscape icon again. The `Command` field was filled in automatically by the program. The final line is left blank, because we use Netscape to print HTML files.

Again, we save the action and reload. An HTML file in the system that is named *something*`.html` will appear with the Netscape icon, and double-clicking on the icon should result in Netscape starting up and loading that page.

You can have several different file types mapping onto the same action, so a repeat of the process above using `*.htm` as the match string will make the system work for HTML files that are constrained by the DOS naming convention.

Pulling It Together

We've gone as far as we can with the `Create Action` GUI. To make things hang together in a slightly more natural way, we need to edit the Action Definition File, and the GUI will refuse to edit the file once we have done that.

You will have noticed that our first attempt with the `Create Action` GUI gave us an icon that we could double-click to start Netscape. Along the way, we've lost that simple double-click ability and now are always are presented with a dialog box. The dialog box is somewhat unfriendly because it doesn't include a file browser. If we want to start Netscape looking at a particular file, it's probably better to use the File Manager to find the file, which is now a datatype and can be double-clicked to start Netscape. Getting rid of the dialog box also neatly sidesteps the issue of starting Netscape via a URL. We can just start the program, and the user can type their URL into it.

So we want to take different actions depending on the number of arguments. There will be two cases: no arguments when the `Netscape` action is double-clicked; and an argument when a file is dropped on the icon or when a datatyped HTML file is double-clicked. You can also drag-and-drop several files.

We can make our Action Definition File contain more than one action called `Netscape` and choose which one to use depending on various criteria, one of which is the number of arguments. It's easy. Pick up the `ACTION` definition for Netscape that I suggested you save earlier and edit it into your current definition file before the current ACTION line. Inside the curly braces for the inserted `ACTION` add

```
ARG_COUNT   0
```

You now have two actions called `Netscape`, the first will be called when the Netscape icon is double-clicked and there are no arguments, the second when there are some associated data files.

We'd also like to deal with the problem of file context. CDE will call Netscape with the full pathname of a data file that is loaded, but ideally we would like Netscape to be running with its current directory in the same directory as the data file. Images and links from the HTML page can then be expressed relative to the page, and we will be presented with a page from a disk that should show the correct information. So we'd like to change directory before Netscape is called.

I tend to tackle this type of problem by making the `Action` call a shell script. The script looks at its arguments, and then calls the target program. We create a shell script called `call_netscape` like

```
#!/bin/sh

if [ $# -gt 0 ] then
    for name in $*
    do
         case $name in
         -*)     ;;
          *)
                 cd `dirname $name`
                 break
         esac
    done
fi
exec /usr/local/bin/netscape -install $*
```

The script examines its arguments. It seems good practice to ignore any argument that is a program option starting with a minus sign. When it finds an argument, it assumes that it's a file name and uses the `dirname` command to generate its path. It calls `cd` to change into the directory. Finally, the `Netscape` program is called.

Notice that I've called `Netscape` here using the `-install` flag to ensure that it allocates a private colormap. A private map is always good practice, because the program tends to be greedy with colormap resources. I've omitted the `-install` flag from the examples above to avoid having to explain it until now.

More Reading

I've derived most of the information herein from the Solaris CDE Answerbook 1.0. The Answerbook should be online on your system. If not, you should ask your systems administrator for it.

There are several printed books that describe CDE, but the ones that look reasonable are often printed versions of the information already available to you in the Answerbook. So, unless you want the information on paper, these books are not going to tell you anything new. I think that on the whole, the CDE documentation is somewhat impenetrable. There's nothing on how it works. You have to make best guesses from the available evidence. I suppose that what you need to know to use CDE is often there, but somehow it's presented in a way that doesn't make it too accessible.

The scripts, icons and actions described in this article are available on my Web site: `http:/www.hillside.co.uk/articles/sunexpert.html`. Look for the section on this article for further links.

***Peter Collinson** runs his own UNIX consultancy, dedicated to earning enough money to allow him to pursue his own interests: doing whatever, whenever, wherever... He writes, teaches, consults and programs using Solaris running on a SPARCstation 2. Email:* `pc@cpg.com`.