

Function Key Access to EnterpriseLink

Customers often wish to fully customize heavily used portions of their applications to assist novice users. However, the bulk of the application (for people familiar with it) is left to "default" with some help from EnterpriseLink.

One of the key issues that can arise with EnterpriseLink applications that are deployed to a browser is how to handle heads-down data-entry type operators who are used to the interactions of a terminal emulator.

In those environments, users hit the F1 key on the keyboard to send a PF1 to the host. PF1 through PF12 are often used, PF13 through 24 rarely used, and on occasion Program Attention keys PA1 through PA3 are employed by host applications.

The trouble has been how to define an EnterpriseLink project so that these keystrokes are trapped and sent to the host. This document describes how to set up a project. Accompanying this document is a [pfkey-example.zip](#), which includes the additional files needed for this example.

First, we create a project with a minimum of two pages and no screens. EnterpriseLink Builder can be used to construct such a project. The page name SSDISCONNECT contains the "thanks for using us" message and SSDEFAULT sets up a default translation template for all unmatched screens that EnterpriseLink Server encounters. If there are no screens captured, then every screen will display SSDEFAULT. Placing a 24 row, 80 column "Multi-Line Edit" object on the SSDEFAULT will be replaced with the IBM 3270 Model 2 24x80 image of text and field elements making up each screen. SSDEFAULT should have the "Preformatting" HTML export option on (General tab of the Page properties) as well as unchecked the "Display Button Bar" (View tab of the Page properties). The former will achieve the same formatting as the original host application through the use of spaces and a fixed-width font. The latter will cause the button bar to be eliminated from these pages as we are going to allow the end-user to use F1..F12.

The remainder of SSDEFAULT can be customised as necessary, for example adding the useful "Enter" and "Quit" (i.e., mapped to Disconnect or PF3) buttons offer the two most often used operations. Putting the company logo on the page will brighten it up a bit.

JavaScript "on key down" event can be used to trap keys in the browser. The following JavaScript (and EnterpriseLink Event processing) is placed into the "Processing" tab of the Page properties.

```
function doPFKey() {
    var x;
    if ((window.event.keyCode >= 112) && (window.event.keyCode <= 123)) {
        if (window.event.altKey && (window.event.keyCode <= 114)) {
            x = window.event.keyCode-111;
            keyname = "PA" + x.toString();
        }
        if (window.event.shiftKey) {
            x = window.event.keyCode-99;
            keyname = "PF" + x.toString();
        } else {
            x = window.event.keyCode-111;
            keyname = "PF" + x.toString();
        }
        document.form1.PFKEY.name = keyname;
        document.form1.PFKEY.value = keyname;
        window.event.returnValue = false;
        document.form1.submit();
        alert("key was " + keyname);
    }
}
BEGIN-EVENT-HANDLER[onKeyDown]
doPFKey()
END-EVENT-HANDLER[onKeyDown]
```

When you replace the Language field of this tab with "JAVASCRIPT EVENT", EnterpriseLink Server will look for the "BEGIN-EVENT-HANDLER" and "END-EVENT-HANDLER" tags and place the

Function Key Access to EnterpriseLink

JavaScript within them in the associated "on" event. In this case, the BODY tag of each EnterpriseLink generated HTML page will contain 'onKeyDown="doPFKey()". Because the definition of the function "doPFKey" is outside these tags, it will be placed in the HEAD section of each page.

The "doPFKey()" function above will be invoked whenever a user depresses any key. The function looks specifically for F1 (keyCode 112) through F12 (keyCode 123). It will also notice whether or not the shiftKey or altKey are depressed at the same time. Also defined on this page is an input field object named PFKEY. The object's display type (General tab of the Input field object property) is overwritten so as to be "Hidden". Dynamically changing this field's name and value to "PF5", for instance, prior to submitting the page, will cause Server to send a PF5 to the host. This set of JavaScript and objects will implement the key mappings:

Sent to Host	Keyboard Combination
PF1...PF12	F1...F12
PF13...PF24	Shift F1...F12
PA1...PA3	Alt F1...F3

Other keyboard mappings (CLEAR, EOF, Erase Input) could be added to this JavaScript. One can check for the Page Up, Page Down, Home and other infrequently used ASCII keyboard keys.

The JavaScript function "doPFKey()" performs a page submittal whenever it notices that keys F1...F12 are pressed. This will cause the Program Function key to take effect immediately. It also performs one more action, which is to set the "return value" of the keyboard event to "false". This causes the default browser action to be suspended. For example, F1 will normally cause Microsoft Internet Explorer to display the help, F3 the "Search Files" dialog box, and F11 will cause the screen to maximise.

Unfortunately, effectively turning off these default actions is prohibited from HTML pages downloaded from a web server. Otherwise, Microsoft believes a hacker-built web site could take control over the end-user's station. Therefore, there is one final important piece to the function key mapping problem. One must deploy an "HTML Application". HTML Applications are "trusted" in that they can override the default Browser operations. HTML Applications are those contained in a file with the ".hta" suffix. For example, the following contained in a file name "start-HollisPFKey.hta" will, upon double-clicking, invoke the EnterpriseLink project HollisPFKey in a trusted manner.

```
<HTML><HEAD>
<TITLE>Example Function Key Input to Hollis</TITLE>
<HTA:APPLICATION ID="oMyApp" APPLICATIONNAME="HollisPFKey"
    SINGLEINSTANCE="yes" SHOWTASKBAR="yes" HEIGHT="600">
</HEAD>
<IFRAME SRC="http://localhost/stscripts/run.stn/HollisPFKey"
    APPLICATION="yes" SCROLLING="no" FRAMEBORDER="no" WIDTH="100%"
    HEIGHT="100%" MARGINWIDTH=0 MARGINHEIGHT=0></IFRAME>
</HTML>
```

The example ZIP file contains an EnterpriseLink Repository with the HollisPFKey project (defined with SSDEFAULT and SSDISCONNECT), a start-up HTML Application file, and a graphic file (place in /stdocs) that's referenced from SSDEFAULT.

The application will connect to Harvard On-Line Library System (HOLLIS). You can select "hollis" on the first screen, "hu" on the second and then hit F3.

This action will send a PF key to the host, which will back out of the application.