

DeskBridge Developers' Guide

---

TABLE OF CONTENTS

What is DeskBridge? .....2

How do I (a developer) work with DeskBridge?.....3

Drawing on the Screen .....4

Monitoring View mode .....6

Handling key input .....7

Displaying messages.....8

Widgets .....8

Document history .....10

# SE-lifestyle

# DeskBridge

# SDK

## WHAT IS DESKBRIDGE?

It's an elf development platform. Its main goals are:

- to create a new desktop surface
- to hide all unnecessary information from the screen
- to reduce CPU load
- to increase stability of elves that draw on standby
- to support landscape display on standby

## How is this done?

The elf has 2 different modes.

- Legacy: this mode does not create the new desktop, uses the old standby screen. It's important to have this, as many elves are not compatible with the new method.
- Own GUI (/Landscape): this supports many things; the most important one is certainly the landscape view. Additionally it is a totally empty surface, no clock, date, battery... nothing annoying, the internal captions system draws them, the way you like. It can be switched to landscape from standing (and back) with a press of a button (BCFG).

## How can it reduce CPU load and increase stability?

There are 2 problems with older elves:

- they apply **InvalidateRect(<standby>)**; when it's not even necessary (see the refresh time in elves' BCFG), this should be done centrally, only one elf should do it, at a certain frame rate.
- they modify the *"OnRedraw"* function for Standby like this:
  1. Get the old redraw function.
  2. Modify it to my own.
  3. In my own, apply the old redraw

And this goes on forever. When an elf is killed, it restores the old redraw. The redraw that was ok, when the elf itself was loaded. So then any other elf, loaded after the killed one won't be showing up on standby. This is fixed perfectly in Bridge.

## DeskBridge

## HOW DO I (A DEVELOPER) WORK WITH DESKBRIDGE?

The elf acts like a server. It has its own functions that can be run from any other elf.

```
typedef struct
{
    BOOK book; //to be able to find it

    int version; //to know if this bridge build is compatible

    int platform; //2010 or 2020 in integer

    int ScreenHeight; //these 2 change when you switch landscape on/off
    int ScreenWidth;

    int sleeping; //the phone is in sleep mode
    int walkman_on; //walkman or media player is running

    DATETIME datetime; //current date and time
    BATT battery; //battery info

    int (*Add_Draw)(void (*Draw)(DISP_OBJ * db,int r1, int r2,int r3),int
refresh_time);
    int (*Reset_Draw)(void (*Draw)(DISP_OBJ * db,int r1, int r2,int r3),int
refresh_time);
    int (*Remove_Draw)(void (*Draw)(DISP_OBJ * db,int r1, int r2,int r3));

    int (*Add_OnKey)(int (*OnKey)(void *p, int key, int i2, int i3, int mode));
    int (*Remove_OnKey)(int (*OnKey)(void *p, int key, int i2, int i3, int mode));

    int (*Modify_View_Hook)(int (*proc)(int mode),int mode);
    int (*Modify_Walkman_Hook)(int (*proc)(int running),int mode);

    void * (*Get_Widget)(wchar_t * name);
    void (*ShowMessage)(wchar_t * message);
}BridgeBook;
```

The thing you see up there is Bridge's book (you must have this in some main.h of your elf) that can be found this way:

## DeskBridge

```
BridgeBook * BB=NULL; //it's not a must but better to have it all the time

int isBridgeBook(BOOK * book)
{
    if(!strcmp(book->xbook->name,"DeskBridge")) return(1);
    return(0);
}

int main(wchar_t *elfname, wchar_t *path, wchar_t *fname)
{
    BB=(BridgeBook*)FindBook(isBridgeBook);
    if (!BB) //if deskbridge is not running, the book won't be found
    {
        #define ShowMessage(__STR__)
            MessageBox(LGP_NULL,Str2ID(__STR__,0,SID_ANY_LEN),0,1,0,0)
        ShowMessage(L"This cannot be run without DeskBridge!");
        SUBPROC(elf_exit); //then there's no reason/way to run this elf either
        return(0);
    }
    ...
}
```

Now you can use Bridge's functions and values from *BB*. This is also a place to check the version (*BB->version==1* in the first release).

## DRAWING ON THE SCREEN

The first thing you need to do is add your own **draw function** like this:

## DeskBridge

```
void Draw(DISP_OBJ * db,int r1, int r2,int r3)
{
    //here do everything as you used to with onredraw functions, except for one
    //thing, no need to execute oldredraw.
    /*you can access details of the phone like
    BB->platform: 2010 or 2020 in (int)
    BB->ScreenHeight and BB->ScreenWidth are for the screen you are drawing
    on, they change when you switch to/from landscape
    */
}

int main(wchar_t *elfname, wchar_t *path, wchar_t *fname)
{
    BB=(BridgeBook*)FindBook(isBridgeBook);
    if (!BB) //if deskbridge is not running, the book won't be found
    {
        ShowMessage(L"This cannot be run without DeskBridge!");
        SUBPROC(elf_exit); //then there's no reason/way to run this elf either
        return(0);
    }

    //here create your elf's book and do everything necessary....

    int refresh_time=1000; //in milli second
    BB->Add_Draw(Draw,refresh_time);
}

static void onMyBookClose(BOOK * book)
{
    //do what you have to.
    BB->Remove_Draw(Draw); //if you don't do this, the phone calls a non-existing
    //function and reboots (WSOD).
}
```

## MONITORING VIEW MODE

Not in all cases, but in many, you may want to know if the view changes (from landscape to normal or the opposite).

For this purpose, I made a hook system that works much like the keyhooks of the phone.

Let us continue the example from before, add and remove our **view hook**...

```
enum {LEGACY,OWN_GUI,LANDSCAPE};

int myhook(int mode)
{
    switch (mode)
    {
        case LEGACY:
            //...
            break;
        case OWN_GUI:
            //...
            break;
        case LANDSCAPE:
            //...
            break;
    }
}

int main(wchar_t *elfname, wchar_t *path, wchar_t *fname)
{
    //...
    BB->Modify_View_Hook(myhook,1);
    //...
}

static void onMyBookClose(BOOK * book)
{
    //...
    BB->Modify_View_hook(myhook,0);
    //...
```

The function **myhook** will be executed whenever the user changes the view mode.

## MONITORING WALKMAN

You can do the same thing with Walkman or Media Player on non-W series. A hook like this is installed the same way with **Modify\_Walkman\_Hook**. int hook(int running)'s running==1 if Walkman is started, 0 if closed.

## DeskBridge

## HANDLING KEY INPUT

The last important thing is using **OnKey functions**.

Now the original (should I say conventional?) OnKey functions are like this:

```
void OnKey(void *p, int key, int i2, int i3, int mode)
{
    void (*OldOnKey)(void *, int, int, int, int);
    OldOnKey=(void (*)(void *,int,int,int,int))oldOnKey;

    if (key==KEY_ENTER)
    {
        //do what you wish
        return;
    }

    //if nothing works ...
    OldOnKey(p,key,i2,i3,mode);
}
```

Where you had to save the old onkey function of the standby to a pointer, and execute it if your own elf doesn't use the actual key. This was a nice thing, working all right but all elves on exit restored the onkey that was actual when they were launched, just like draw functions did. I wanted to make them like KeyHooks though, to have a return value (which the original onkey does not have). See the example (still continuing that example):

```
int OnKey(void *p, int key, int i2, int i3, int mode)
{
    if (key==KEY_ENTER && mode==KBD_SHORT_PRESS)
    {
        //do whatever
        return(-1); //returning -1 or any non-zero value will keep Bridge from
executing all the other onkey functions
    }
    //check all the keys you want to
    return(0); //if no keys were ok, the function will return 0 (zero) and so
Bridge will be told to move on to the next onkey
}

int main(wchar_t *elfname, wchar_t *path, wchar_t *fname)
{
    //...
    BB->Add_OnKey(OnKey);
    //...
}

static void onMyBookClose(BOOK * book)
{
    //...
    BB->Remove_OnKey(OnKey);
    //...
}
```

# DeskBridge

## DISPLAYING MESSAGES

You can use DeskBridge's built-in messagebox gui to show messages to the user like this:

```
BB->ShowMessage(L"my message");

//or you can use a more complex way to print any variable...
wchar_t * buf[100];
snwprintf(buf,99,L"%s : %d","some_str",92);
BB->ShowMessage(buf);
```

## WIDGETS

A very important new feature in this elf is the widget support. Widgets (or gadgets in Windows) here are small elf applications, written in c++. The difference between normal elves and them is mainly that they don't create a book, DeskBridge loads and unloads them. And their name is \*.widget.

You can start learning how to make them by reading my sources. If you do so, you'll see, that: (in widget.cpp and widget.h) the widget first looks for the DeskBridge book, if it's not found, it unloads. Then asks Bridge for the WIDGET in the memory. It stands for the current widget, like stores its settings (position, size). If it's not found, again it exits.

If everything's fine, you can initialize, like load pictures. Then it assigns its Draw and kill\_widget fuctions, so that Bridge will know what to call. If you load a picture, you must also unload it in kill\_widget() function.

You can also use ActiveStrings, it's like captions in Bridge, you enter \$d1.\$d2.\$d3 and it will make them 2009.12.07 or so. For more info on usage, see Calendar.wiget.

If you do so, you'll also see:

```
int scale=0;

void Draw(DISP_OBJ * db,int r1, int r2,int r3)
{
    if (scale!=widget->scale)
    {
        scale=widget->scale;
        font_big=0;
        FONT_DESC *fonts=GetFontDesc();
        for (int i=0, max=*GetFontCount(); i<max; i++)
        {
            if (fonts[i].name[wstrlen(fonts[i].name)-1]==L'R')
            {
                SetFont(fonts[i].id);
                if (GetImageHeight(' ')<=WidgetHeight*0.5)
                    font_big=fonts[i].id;
                else
                    break;
            }
        }
    }
}
```

This code can set up fonts (font\_big) to go with the widget size, according to scale. For it to work, you also need:



## DeskBridge

```
#define WidgetWidth GetImageWidth(bg->ImageID) *scale/100
#define WidgetHeight GetImageHeight(bg->ImageID) *scale/100
```

It's quite obvious what they do.

```
IMG bg[1];
putchar(get_DisplayGC(), widget->x, widget->y, WidgetWidth,
GetImageHeight(bg[0].ImageID) *scale/100, bg[0].ImageID);
```

You should always draw images like this, so that the scaling will work.

To set up the about box, simply:

```
widget->about_widget=L"my about info";
```

To have BCFG support, do what you always do when you add bcfg to an elf... except: it must be in /usb/other/ZBin/BridgeWidgets directory, named as the widget itself but with bcfg extension (this is set up in conf\_loader.cpp).

```
void reConfig()
{
    Destroy(); //calendar has to unload the old background
    InitConfig();
    Init(path); //and kiad the new one
}

int main(wchar_t *elfname)
{
    ...
    InitConfig();
    ...
    widget->reinit_cfg=reConfig; //here you could put InitConfig
                                //if you don't have to reload images and stuff
    ...
    return 0;
}
```

See Calendar or Clock widget for more info.

In IAR, right click on project's name on the left, hit *Options*. Go to *Linker* on the left, select *Override default* and name your elf like *MyWgt.widget* (use **.widget** extension). Move this widget file to /usb/other/ZBin/BridgeWidgets. You can use directories but the folder's name must be the widget's name, so for *MyWgt.widget*, the full path will be /usb/other/ZBin/BridgeWidgets/**MyWgt/MyWgt.widget**.

## DOCUMENT HISTORY

- 2009.07.09. – Creation
- 2009.12.07. – Added info on ShowMessage and Widgets.
- 2009.12.12. – Added info on new widget features (about box and bcfg), removed no longer actual data
- 2009.12.18. – Actualized BridgeBook info
- 2009.12.23. – Implemented Walkman hooks and also added them