Mesa Platform Development Kit

Software Customization Guide

Version	Date	Description	Author(s)
0.1	9-Feb-10	Initial Version: Still missing Home Screen App customization and Splash Screen customization.	J. McKell
0.2	12-Feb-10	Added Splash & Progress screen customization	J. McKell, Z. Johnson
0.3	26-May-10	Added Home Screen customization information	J. McKell
0.4	02-Jun-10	Reconfigured information for PDK	J. McKell
1.0	05-Aug-10	Removed information exclusive to Branding.	J. McKell
1.1	05-Aug-10	Updated review issues	J. McKell
1. 2	30-Aug-10	Minor changes	J. McKell
1. 3	1-Oct-10	Added more registry settings	J. McKell
1. 4	26-Oct-10	Added information about updating JSHome gadgets & apps “on-the-fly”.	J. McKell
1. 5	20-Dec-10	Added more information about how to update JSHome on the fly	J. McKell
1.6	6-Jan-12	Added description and example code to create a gadget for JSHome	J. McKell

Table of Contents

1 Keypad Customization 3
1.1 System Level Keypad Customization 3
1.1.1 Press/Release Configuration 4
1.1.2 Press/Hold Configuration 4
1.1.3 Press/Hold Timeout 4
1.2 User Level Customization 5
2 Record Button Customizations 5
3 Audio Driver Customizations 5
4 Subdued Lighting Mode (boot backlight brightness) 5
5 USB ActiveSync 6
5.1 Clock Resume Behavior 6
5.2 Connection Behavior 6
6 USB Host 6
7 SD Card 6
8 Serial Port Settings 7
8.1 Power to COM1 pins 7
8.2 Power to DTR 7
8.3 Power to RI 7
8.4 Port Enable/Disable 7
8.5 Custom/Non-Standard Baud Rates 8
9 Displayed Battery Average 8
10 Getting Started 9
11 Home Screen 9
11.1 User Customizations 9
11.2 System Customizations 9
11.2.1 “Favorite” Applications 9
11.2.2 Lock Application Choices 10
11.2.3 Gadget Configuration 10
11.2.4 Update JSHome 11
12 Appendix A: Scan Codes for Keypad Customization 12
13 Appendix B: Microsoft Documentation of Application Button Configuration 13
14 Appendix C: Example Getting Started Content File 14
15 Appendix D: CAB File To Update JSHome 15
15.1 Problem Statement 15
15.2 Background 15
15.3 Application Installation 16
15.4 Step-By-Step 16
15.5 Sample code 17

Introduction

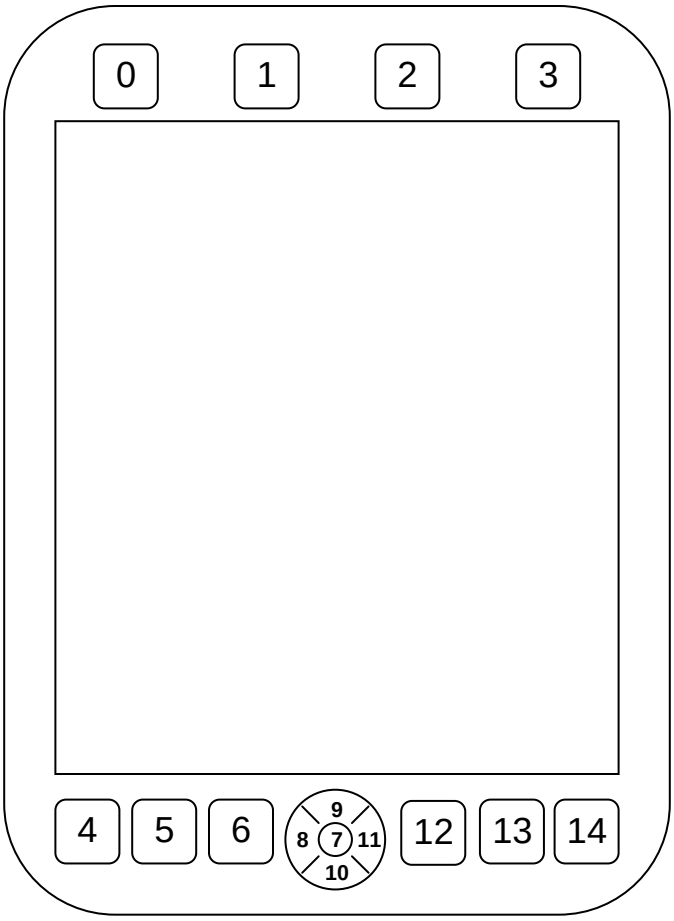
This document is intended for Resellers and OEMs that want to customize the Mesa device for their application or domain. These instructions describe the changes that can be made in the behavior and appearance of the Mesa device. Customizers can change the definition/layout of the buttons, the contents of the Getting Started application, the Home Screen application and Splash Screens.

1 Keypad Customization

There are two levels of customization of the functions that keys can perform, system level and user level. System level customization allows customizers to change the behavior of every single button on the device except for the power button. User level customization is buttons that can be customized by users in the Button Control panel.

1.1 System Level Keypad Customization

At the system level, there are two actions that each key can perform. One when a key is pressed and released, the other when the key is pressed and held. There is a configurable timeout for the hold time. The keys are numbered according to this diagram:



1.1.1 Press/Release Configuration

Each key has its own registry entry that defines either a scan code (see [insert link to Appendix A]) or an application name. Note that if both are provided, the scan code setting will take precedence. If an application name is provided, an optional command line parameter can also be provided.

Press and Release key customizations are stored in **HKEY_CURRENT_USER\ControlPanel\Keybd\PressRelease\<key number>.**

Name	Type	Description
ScanCode	REG_DWORD	This is the scan code that should be produced when this key is pressed
AppName	REG_SZ	This is the name of the application that is to be launched when this key is pressed. Note that if **ScanCode** is defined, this key is ignored.
CmdLine	REG_SZ	(Optional) This is the command line for the application to be launched. Note that if **ScanCode** is defined, this key is ignored.

1.1.2 Press/Hold Configuration

The action that is to occur when the user presses & holds a key up to the time out time is defined in a very similar manner.

Press and Hold Key customizations are stored in **HKEY_CURRENT_USER\ControlPanel\Keybd\PressHold\<key number>.**

Name	Type	Description
ScanCode	REG_DWORD	This is the scan code that should be produced when this key is pressed
AppName	REG_SZ	This is the name of the application that is to be launched when this key is pressed. Note that if **ScanCode** is defined, this key is ignored.
CmdLine	REG_SZ	(Optional) This is the command line for the application to be launched. Note that if **ScanCode** is defined, this key is ignored.

1.1.3 Press/Hold Timeout

The length of time to choose between a Press and Hold versus a Press and Release of a key is configurable.

The timeout is stored in **HKEY_CURRENT_USER\ControlPanel\Keybd\PressHold**.

Name	Type	Description
Delay	REG_DWORD	This value configures the amount of time in Milliseconds

		that should pass before a held key results in a Press and Hold action. The default value is 1500.

1.2 User Level Customization

The customizer can decide whether keys can be configured by the user or not. In order to make a key configurable by the user, the customizer must perform a System Level customization for that key: Scan codes A1 through A6 are used for this purpose. See Appendix [TWO] for Microsoft OEM documentation.

2 Record Button Customizations

The behavior of the record button can be customized as well. The characteristics of the recording can be changed if desired. These settings are found in **HKEY_LOCAL_MACHINE\Software\JuniperSys\RecordButton.**

Name	Type	Description
MaxDuration	REG_DWORD	Maximum length of the recording, default is 2BF20 or 3 minutes.
NumChannels	REG_DWORD	Number of recording channels to use. The default is 1.
BitsPerSample	REG_DWORD	Number of bits per sample, possibilities are 8 and 16. The default is 8.
SampleRate	REG_DWORD	Sampling rate. The default is 5DC0 or 24000

Note that these values determine the size of the buffer that will be used, which is set aside at the beginning of recording. If these numbers are set too high, the buffer will fail to be created and no recording will be made.

3 Audio Driver Customizations

For some situations, it may be desirable to change the gain on the speaker as well as the quality. These registry keys can be found in **HKEY_LOCAL_MACHINE\Drivers\BuiltIn\WaveDev.**

Name	Type	Description
OverGain	REG_DWORD	Change the volume. Can be used to overdrive the audio output. The default is 0x10 or 16, which is unity. Increase to overdrive the audio output.
OutSamplingRate	REG_DWORD	Audio output sampling rate. Can be used to change the quality of the audio output. The default is 0x7D00 or 32000. Other rates are untested.

4 Subdued Lighting Mode (boot backlight brightness)

Both the screen and keypad backlight brightness can be set to either a default value or to the value set in the backlight control panel. This can be toggled at boot time by pressing and holding (until the progress screen is displayed) <Home> + <Up> + <Brightness Up>.

5 USB ActiveSync

5.1 Clock Resume Behavior

Mesa has different behavior from other Juniper Systems products. If a unit is left in suspend and connected to a PC through USB ActiveSync for a fairly long period of time (over Midnight, for example), the unit will periodically resume to perform bookkeeping operations for appointments, alarms, and such. Other Juniper Systems devices will connect with ActiveSync and will potentially remain on and drain the battery. Mesa will not connect through ActiveSync and will shut back down when the **AlwaysResume** value is set to 0. It will behave like other Juniper Systems products when it is 1.

5.2 Connection Behavior

Mesa will delay a connection to ActiveSync with the **ResumeDelay** value. This will avoid issues where a unit is connected when booting or taking a long time to resume. A partial connection can put the ActiveSync connection into a bad state. This prevents a partial connection until the device is ready. These settings are located in **HKEY_LOCAL_MACHINE\Drivers\BuiltIn\UsbFn**

Name	Type	Description
AlwaysResume	REG_DWORD	0=disable USB Client if resuming from RTC, 1=always run activesync, Default is 0
ResumeDelay	REG_DWORD	Number of milliseconds to wait after a resume before making an ActiveSync connection. Default value is 15000 milliseconds or 0x3A98

6 USB Host

The USB Host port can be configured to provide USB power even when the host is in suspend. This setting can be found in **HKEY_LOCAL_MACHINE\Drivers\BuiltIn\OHCI**

Name	Type	Description
SuspendPower	REG_DWORD	0=IO Port and dock power down during suspend, 1=keep it powered always. The default is 0, or off in suspend

7 SD Card

The SD Card can be configured to provide power even when the host is suspended. For some applications that maintain open files to SD storage cards, this will maintain open file handles and allow applications to continue writing (or reading) immediately. The setting can be found in **HKEY_LOCAL_MACHINE\Drivers\BuiltIn\SDHC1**

Name	Type	Description
SuspendPower	REG_DWORD	0=IO Port and dock power down during suspend, 1=keep it powered always. The default is 0 or off in suspend.

8 Serial Port Settings

There are several settings that can be customized depending upon the scenario desired by the user.

8.1 Power to COM1 pins

The power to the pins on COM1 can be configured to be powered off when the COM port is closed. This can help in environments where corrosion is high. This setting can be found in

HKEY_LOCAL_MACHINE\Drivers\BuiltIn\Serial1

Name	Type	Description
TransAlwaysOn	REG_DWORD	0= Transceiver is enabled/disabled when COM port opened or closed, 1= Transceiver is always enabled. The default is 1 (always enabled).

8.2 Power to DTR

The DTR pin can be configured such that it can either be powered from the serial port protocol or when the COM port is opened or closed. This setting can be found in

HKEY_LOCAL_MACHINE\Drivers\BuiltIn\Serial1

Name	Type	Description
DtrAuto	REG_DWORD	0= Only apps set/clear DTR, 1= COM driver clears DTR during COM_Close, if value doesn't exist then default is 1 (Apps set and clear)

8.3 Power to RI

The power to Ring In or RI can be configured in one of three ways. The Ring In pin can be controlled independently by JSAPI (default), can be tied to DTR behavior (set when DTR is set and vice versa) or can be enabled/powered when the COM port is open. This setting can be found in

HKEY_LOCAL_MACHINE\Drivers\BuiltIn\Serial1

Name	Type	Description
LinkRI	REG_DWORD	0= RI power is only controlled by JSAPI, 1= RI power is set whenever DTR is set, 2= RI power is set whenever the COM port is open, if value doesn't exist then default is 0. The default is 0.

8.4 Port Enable/Disable

Sometimes it is useful to disable a physical COM port in order to free up the "COM" name-space for other services like Bluetooth. Any physical port (COM1, COM3 and COM7) can be disabled if desired. COM3 is disabled by default. This setting can be found in

HKEY_LOCAL_MACHINE\Drivers\BuiltIn\Serialx where x is 1, 2 or 3

Name	Type	Description
Enable	REG_DWORD	0= Do not load this COM port, 1= Load this COM port.

		The default is 1 for COM1, 0 for COM3 (Expansion) and 1 for COM7 (GPS).

8.5 Custom/Non-Standard Baud Rates

It is possible to set a custom or non-standard baud rate. The following table lists standard baud rate keys. One can change a standard baud rate to a non-standard baud rate by changing the divisor associated with that baud rate. Only change these values if you are willing to accept any consequences. There is no warrantee express or implied in sharing this information. These settings can be found in **HKEY_LOCAL_MACHINE\Drivers\BuiltIn\Serial1**

Name	Default Value
50	0x4800
75	0x3000
150	0x1800
300	0xC00
600	0x600
1200	0x300
1800	0x200
2400	0x180
3600	0x100
4800	0xC0
7200	0x80
9600	0x60
12800	0x48
14400	0x40
19200	0x30
23040	0x28
28800	0x20
38400	0x18
57600	0x10
115200	0x8
230400	0x4
921600	0x1

9 Displayed Battery Average

The displayed battery average is calculated based on a 5 second sample. Currently, the battery average is calculated over a two minute period or 24 samples (24 samples * 5 samples/second = 120 seconds = 2 minutes). To change the period of the displayed battery average, change the number of samples in the key found here **HKEY_LOCAL_MACHINE\Drivers\BuiltIn\Battery**

Name	Type	Description
AveTTEsamples	REG_DWORD	Default is 0x18 (=24dec - 24*5 = 120 --> Averages over a 2 minute time period based on 5 second polling)

10 Getting Started

The Getting Started application is the main place for new device users to learn about and possibly configure their new device. This is a good place to provide new users with documentation about how to set up their device – to configure applications or sensors, to make network connections. Getting Started content is written in HTML. See Appendix [THREE] for an example file. In addition to the content, changes must be made to the registry to organize the content in the Getting Started application.

Registry keys are located in
HKEY_LOCAL_MACHINE\System\WelcomeCenter\<language ID>\<topic name>.

Name	Type	Description
image	REG_SZ	Path to the image to be displayed for this entry
url	REG_SZ	Path to the HTML file that contains the information for this topic
name	REG_SZ	Text to be displayed in the Getting Started application
order	REG_DWORD	List position

11 Home Screen

The custom home screen for Mesa has two levels of customization: User customizations and system customizations.

11.1 User Customizations

Users can change several aspects of the Home Screen:

- User can enable or disable it (Start->Settings->Home->Items)
- User can select the gadgets that are visible (Menu->Configure)
- User can select the application short cuts that are visible (Press and Hold on App icon)
- User can select either a dark theme or a light theme (Menu->Switch Color Theme)

11.2 System Customizations

In an effort to "focus" users on tasks/information/applications that the Mesa is incorporated into, there are several customizations that can be done to focus the user to the gadgets and applications that are key to a given solution.

11.2.1 "Favorite" Applications

The Home Screen has four "slots" for favorite applications. The default presentation for the user is (left to right, top to bottom): "Getting Started", "File Explorer", "Task Manager", and the last slot is unused. These four default "slots" can be customized to other applications. JSHome can launch applications from anywhere, but will list for the user all application shortcuts found in \Windows\Start Menu\Programs\. To be most robust, create a shortcut for your application and place it there. The for slots for applications are controlled by a registry key:

HKEY_LOCAL_MACHINE\Software\JuniperSys\JSHome\Apps

Name	Type	Description
App1	REG_SZ	Full path & name of link file to launch application for slot 1
App2	REG_SZ	Full path & name of link file to launch application for slot 2
App3	REG_SZ	Full path & name of link file to launch application for slot 3
App4	REG_SZ	Full path & name of link file to launch application for slot 4

11.2.2 Lock Application Choices

The application choices can also be "locked" – that is, the user cannot change them. This is controlled by a registry key as well:

HKEY_LOCAL_MACHINE\Software\JuniperSys\JSHome

Name	Type	Description
AppLock	REG_DWORD	Locks application choices. Press & Hold on Application icon has no effect. 1 = locked, 0 or missing means not locked.

11.2.3 Gadget Configuration

As indicated above, Gadgets can be configured by users. The Home Screen implements gadgets as separate Dynamic Link Libraries. There are eleven gadgets provided:

Gadget	DLL
Battery	JSHome_Battery.dll
Bluetooth	JSHome_Bluetooth.dll
Calendar	JSHome_Calendar.dll
Clock	JSHome_Clock.dll
Email	JSHome_Email.dll
3G Modem	JSHome_GPRS.dll
GPS Status	JSHome_GPSStatus.dll
GPS Compass	JSHome_GPSCompass.dll
Texting	JSHome_SMS.dll
Tasks	JSHome_Tasks.dll
Wi-Fi	JSHome_Wifi.dll

The user manual has more information about each gadget. The Home Screen has "slots" for six gadgets. The top row is always Wifi, Clock and Bluetooth. The Bottom row varies depending upon the model. The first slot on the bottom row is Battery or else is 3G Modem if it is present. The second and third slots are Calendar and Email, but if the unit is a "Geo" unit, they are GPS Status and GPS Compass respectively.

The defaults can be changed to other choices via registry key.

HKEY_LOCAL_MACHINE\Software\JuniperSys\JSHome\Gadgets

Name	Type	Description
Gadget1	REG_SZ	Name of the DLL to be used in Gadget Slot 1, default is JSHome_Wifi.dll
Gadget2	REG_SZ	Name of the DLL to be used in Gadget Slot 2, default is JSHome_Clock.dll
Gadget3	REG_SZ	Name of the DLL to be used in Gadget Slot 3, default is JSHome_Bluetooth.dll

Gadget4	REG_SZ	Name of the DLL to be used in Gadget Slot 4, default is JSHome_GPRS.dll if a Cell Modem is present, it is JSHome_Battery.dll otherwise.
Gadget5	REG_SZ	Name of the DLL to be used in Gadget Slot 5, default is JSHome_GPSStatus.dll if the built-in GPS is present, it is JSHome_Calendar.dll otherwise.
Gadget6	REG_SZ	Name of the DLL to be used in Gadget Slot 6, default is JSHome_GPSCompass.dll if the built-in GPS is present, it is JSHome_Email.dll otherwise.

11.2.4 Update JSHome

The changes to the favorite applications and gadgets will take effect after: 1) a power cycle, 2) a reset, 3) disabling and re-enabling JSHome (called "Mesa Dashboard" in Start->Settings->Home->Items) or 4) by sending an event programmatically to JSHome. The following code snippet provides the core of what needs to be done:

```
Handle hReloadEvent;


// Open the event named, "JSHOME_RELOAD"
hReloadEvent = OpenEvent(EVENT_ALL_ACCESS, FALSE, L"JSHOME_RELOAD");

// Set the event (causes JSHome to update configured faves AND gadgets.
SetEvent(hReloadEvent);

// Close the event
CloseHandle(hReloadEvent);
```

For a more detailed sample, please refer to Appendix D: CAB File to Update JSHome below.

11.2.5 Create Custom Gadget

A custom gadget can be created that can provide interactive capabilities. A home screen gadget can:

- Show the state of a connected device or sensor such as a custom pod or a configured Bluetooth device.
- Show the state of a program
- Launch a program
- Turn power on or off to a connected device or sensor
- Trigger an event, possibly for data collection

The existing gadgets do things like:

- Show the current battery level and provide access to the battery control panel
- Show or toggle the current state of a radio like GPS, Bluetooth, Wi-Fi and Cellular
- Interact with a GPS "Compass"

An example gadget is provided below to show how to toggle the green LED.

12 Appendix A: Scan Codes for Keypad Customization

Customizing keys involves knowing the correct scan code that must be used.

Character/Key	Scan Code
A	1C
B	32
C	21
D	23
E	24
F	2B
G	34
H	33
I	43
J	3B
K	42
L	4B
M	3A
N	31
O	44
P	4D
Q	15
R	2D
S	1B
T	2C
U	3C
V	2A
W	1D
X	22
Y	35
Z	1A
0	45
1	16
2	1E
3	26
4	25
5	2E
6	36
7	3D
8	3E
9	46
F1	05
F2	06
F3	04
F4	0C
F5	03
F6	0B
F7	83
F8	0A
F9	01
F10	09
F11	78
F12	07
F13	08
F14	10

Character/Key	Scan Code
F15	18
F16	20
F17	28
F18	30
F19	38
F20	40
F21	48
F22	50
F23	57
F24	5F
UP	5C
DOWN	60
LEFT	62
RIGHT	63
TAB	0D
BACK QUOTE	0E
CLEAR	0F
LEFT ALT	11
LEFT SHIFT	12
LEFT WINDOWS	13
LEFT CONTROL	14
SPACE	29
COMMA	41
PERIOD	49
SLASH	4A
SEMICOLON	4C
HYPHEN	4E
DELETE	4F
APOSTROPHE	52
INSERT	53
LEFT BRACKET	54
RIGHT BRACKET	5B
EQUAL	55
ZOOM	56
CAPITAL	58
RIGHT SHIFT	59
RETURN	5A
BACKSLASH	5D
HELP	5E
CONVERT	64
BACK	66
NOCONVERT	67
TAB	68
NUMPAD0	70
NUMPAD1	69
NUMPAD2	72
NUMPAD3	7A
NUMPAD4	6B
NUMPAD5	73
NUMPAD6	74

Character/Key	Scan Code
NUMPAD7	6C
NUMPAD8	75
NUMPAD9	7D
DECIMAL	71
ESCAPE	76
NUMLOCK	77
ADD	79
SUBTRACT	7B

Character/Key	Scan Code
MULTIPLY	7C
SCROLL	7E
HOME	7F
END	80
PRIOR	81
NEXT	82
SNAPSHOT	84

13 Appendix B: Microsoft Documentation of Application Button Configuration

Microsoft has provided information about how to initialize User Customizable Keys. The proper way of modifying the registry is by using the following registry keys.
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Shell\Keys\<*app button key code*>
For app button 1: <*app button key code*> = 40c1, scan code = A1
For app button 2: <*app button key code*> = 40c2, scan code = A2
For app button 3: <*app button key code*> = 40c3, scan code = A3
For app button 4: <*app button key code*> = 40c4, scan code = A4
For app button 5: <*app button key code*> = 40c5, scan code = A5
For app button 6: <*app button key code*> = 40c6, scan code = A6

Name	Type	Description
@	REG_SZ	REQUIRED. Location/Name of app to launch
Name	REG_SZ	REQUIRED. Name that will appear on button settings
Icon	REG_SZ	RECOMMENDED. Location/Name of icon to be used (may be in a DLL with a reference to that resource)
ResetCmd	REG_SZ	REQUIRED if Flags specifies an app, should be same as the default value for the key, above
Flags	REG_DWORD	REQUIRED. Specifies if button executes a special command instead of an application
ResetFlags	REG_DWORD	REQUIRED if Flags specifies a special command.

App flags used:
0 – Launch specified app (no special command)
1 – Start menu
2 – Toggle SIP
4 – Show Today screen
5 – Scroll up
6 – Scroll down
6 – Scroll left
8 – Scroll right
9 – Do nothing

10 – Done button (OK/X)
11 – Context menu
12 – Rotate Screen
13 – Left Soft Key
14 – Right Soft Key

Each User Customizable key that has been defined in the System Level customization should be configured here.

```
; Task Manager
[HKEY_LOCAL_MACHINE\Software\Microsoft\Shell\Keys\40c1]
    @="\"\\Windows\\Start Menu\\Programs\\Task Manager.lnk\""
    "ResetCmd"="\"\\Windows\\Start Menu\\Programs\\Task Manager.lnk\""
    "Flags"=dword:0
    "Name"="App 1"

; Toggle Soft Input Panel
[HKEY_LOCAL_MACHINE\Software\Microsoft\Shell\Keys\40c2]
    "Flags"=dword:2         ; Toggle Sip
    "Name"="App 2"

; Brightness Down
[HKEY_LOCAL_MACHINE\Software\Microsoft\Shell\Keys\40c3]
    @="\"\\Windows\\BrightnessDown.lnk\""
    "ResetCmd"="\"\\Windows\\BrightnessDown.lnk\""
    "Flags"=dword:0
    "Name"="App 3"

; Brightness Up
[HKEY_LOCAL_MACHINE\Software\Microsoft\Shell\Keys\40c4]
    @="\"\\Windows\\BrightnessUp.lnk\""
    "ResetCmd"="\"\\Windows\\BrightnessUp.lnk\""
    "Flags"=dword:0
    "Name"="App 4"

; Left Soft Key
[HKEY_LOCAL_MACHINE\Software\Microsoft\Shell\Keys\40c5]
    "Flags"=dword:D         ; Left Soft
    "Name"="Left Soft"

; Right Soft Key
[HKEY_LOCAL_MACHINE\Software\Microsoft\Shell\Keys\40c6]
    "Flags"=dword:E         ; Right Soft
    "Name"="Right Soft"
```

14 Appendix C: Example Getting Started Content File

Getting Started content is stored in HTML files. The file below is the content for the Bluetooth topic.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="en">
<head>
<title>Getting Started</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css"
href="file://\windows\wc_style.css" />
```

```
</head>

<body>
    <p><img src="\Windows\bth.gif" alt="" /> <span> Set up a
Bluetooth device</span></p>
    <table>
        <tr>
            <td>You can set up a Bluetooth device in <b>Settings</b> >
<b>Bluetooth</b>.</td>
            <td><a href="\Windows\ctlpnl.exe::cplmain 23"><img
src="shortcut.gif" alt="" /></a></td>
        </tr>
    </table>
    <table>
        <tr>
            <td>The following steps explain how to prepare your unit
and Bluetooth device for a connection:</td>
        </tr>
        <tr>
            <td>1.  Turn on your Bluetooth device and place it within a
few feet of your unit.</td>
        </tr>
        <tr>
            <td>2.  Set your Bluetooth device to visible. This allows
your unit to detect it and establish a connection.</td>
        </tr>
        <tr>
            <td>3.  On the <b>Bluetooth</b> page in <b>Settings</b>,
click <b>Add a new device</b>.</td>
        </tr>
        <tr>
            <td>4.  Select your Bluetooth device from the list, and
then follow the instructions on the screen.</td>
        </tr>
    </table>
</body>
</html>
```

15 Appendix D: CAB File To Update JSHome

15.1 Problem Statement

Resellers and application developers can take advantage of the ability to install an application and have that application immediately available in the “Applications” bar of the Juniper Systems Home Screen. This paper will provide a description of the steps that must be taken to install an application that will be available to the Juniper Systems Home Screen and to make it visible on the “Applications” bar.

15.2 Background

The JS Home Screen uses the registry to define which gadgets and applications show up where. The end user can change which gadgets and applications are shown. When this change is made, JS Home Screen first updates registry settings that indicate which gadget and application go where, and then updates the appearance and linkages during that operation.

Note 1: The end user can be locked out of making any changes to the gadget and application configuration of JS Home Screen. This is documented in the

customization guide.

Note 2: when the user presses and holds on an application icon in the "Applications" bar, it will display a list of applications. This list is generated dynamically from the applications found in the Start Menu. The Start Menu and JS Home Screen use the shortcuts found in "\Windows\Start Menu\Programs". Newly installed applications should add a shortcut file there directing them to the actual executable.

This installation method assumes that a CAB file is being used to install the application. The CAB file can be augmented to execute your code during the installation / un-installation process, before and/or after performing the actual installation operations. This is done through what is known as a setup dynamic link library or setup.dll. The Windows CE/Windows Mobile installer, **wceload.exe**, looks at the configuration data in the CAB file for a configured setup.dll that contains functions that are called as part of the process.

Note 3: More information about Setup DLLs in CAB files can be found at this Microsoft Developer Network web page. Also search for an article titled, "Optional Setup.dll Files for Installation".

15.3 Application Installation

When an application is installed, no further action is taken by JS Home Screen, except that when the user chooses to change an application showing on the home screen, the list of available applications will include the new application (if the install process for the application included a shortcut in the "\Windows\Start Menu\Programs" directory). The installer can change the registry so that the application is configured as part of the "Applications" bar on the JS Home Screen, but until further action is taken (a reboot, enabling and disabling JS Home Screen, etc), no change is made to the functionality of JS Home Screen.

In order to trigger the change, JS Home Screen must receive some sort of notification. This takes the form of a named system event. Events are used to synchronize and communicate between processes on the system. The event that is used is named, "JSHOME_RELOAD". It will cause JS Home to update the gadgets and applications that are shown. This event can be set or triggered as part of the installation process.

15.4 Step-By-Step

Follow these steps to have your application displayed as a choice on the "Applications" bar of JS Home Screen. This assumes that you can change the installer for the application and that the installer is a CAB file. You can use Visual Studio to create the CAB file for you or you can use the command line tool, **cabwiz.exe**. This step-by-step should be able to work with both methods.

Step 1: Add the registry change to make your application show up in the "Applications" bar:

```
[HKEY_LOCAL_MACHINE\Software\JuniperSys\JSHome\Apps]
    "App4"="\Windows\Start Menu\Programs\<app name>.lnk"
```

Note that it calls out the shortcut file discussed above.

Step 2: Build a DLL as described in this MSDN page: http://msdn.microsoft.com/en-us/library/aa924308.aspx, include this code in the routine Install_Exit:

```
HANDLE hReloadEvent;


// Open the named event
hReloadEvent = OpenEvent(EVENT_ALL_ACCESS, FALSE, L"JSHOME_RELOAD");

// Set the event (forces JSHome to update configured faves AND gadgets.
SetEvent(hReloadEvent);

// Close the event
CloseHandle(hReloadEvent);
```

Add this to your CAB file configuration, either in Visual Studio or in the .ini file for your CAB file if building with **cabwiz**.

Step 3: Configure the setup DLL to be the one used by **wceload**:
In Visual Studio, select your "Smart Device" CAB project, click on the "Properties" tab (NOTE: NOT the "Properties" item in the right-click context menu!), make sure that "CE Setup DLL" is set to the name of your DLL which you have added to your Visual Studio solution.
In the .ini file, set:

```
CESetupDLL=<nameofsetup.dll>
```

Where <nameofsetup.dll> is the name of your setup dll file, which should already be a part of your .ini file.

Step 4: Build your cab file!

15.4.1 Sample code

JSHome registry settings for the configured gadgets and applications does not take effect immediately when the registry settings are changed. As mentioned above, there are other methods that can be used to cause these settings to take effect, this is a description of how this can be done through a CAB file. This sample was done using Visual Studio 2008 and the Windows Mobile 6.0 (or 6.5) SDK, so there may be some variation in steps. The basic concept is this. CAB files allow the addition of a DLL, usually called "setup.dll" that contains functions that are executed at the beginning of installation, after installation is complete, before un-installation, and after un-installation is complete. See this MSDN article for more information. This walks through the process of creating a setup DLL and how to include it in a CAB file.

I have created a Visual Studio project that consists of:

- A sample application (CustomCABApp)
- The setup DLL (CustomCABSetup)
- The Smart Device CAB file (CustomCABInstaller)
    - Sets the registry for JSHome
    - Creates a shortcut for the application so it will show up in the Start Menu (and the JSHome application list)
    - Installs the application
    - Installs and runs the Setup.dll

The project is available upon request, but the following shows the most significant parts:

Here is the code from my sample DLL (I called this file CustomCABSetup.cpp):

```
#include <windows.h>
#include <ce_setup.h>

codeINSTALL_INIT
Install_Init(
    HWND        hwndParent,
    BOOL        fFirstCall,     // is this the first time this function
is being called?
    BOOL        fPreviouslyInstalled,
    LPCTSTR     pszInstallDir
)
{
    // For our purposes, there is nothing left to do here, just
continue.
    return codeINSTALL_INIT_CONTINUE;
}


codeINSTALL_EXIT
Install_Exit(
    HWND    hwndParent,
    LPCTSTR pszInstallDir,
    WORD    cFailedDirs,
    WORD    cFailedFiles,
    WORD    cFailedRegKeys,
    WORD    cFailedRegVals,
    WORD    cFailedShortcuts
)
{
    HANDLE hReloadEvent;

    // Were there any failures to install items?
    if ((cFailedDirs == 0) &&
        (cFailedFiles == 0) &&
        (cFailedRegKeys == 0) &&
        (cFailedRegVals == 0) &&
        (cFailedShortcuts == 0))
    {   // NO.  Get a handle to the JSHome Reload event
        hReloadEvent = OpenEvent(EVENT_ALL_ACCESS, FALSE,
L"JSHOME_RELOAD");

        // Set the event (will force JSHome to update the configured
favorites AND gadgets.
        SetEvent(hReloadEvent);

        // Close the event
        CloseHandle(hReloadEvent);
```

```
    }
    return codeINSTALL_EXIT_DONE;
}

codeUNINSTALL_INIT
Uninstall_Init(
    HWND        hwndParent,
    LPCTSTR     pszInstallDir
)
{
    // TODO: Add custom uninstallation code here
    return codeUNINSTALL_INIT_CONTINUE;
}

codeUNINSTALL_EXIT
Uninstall_Exit(
    HWND    hwndParent
)
{
    // TODO: Add custom uninstallation code here
    return codeUNINSTALL_EXIT_DONE;
}
```

Visual Studio generates an INF file for CABWIZ.exe to use to create the CAB file:

```
[Version]
Signature="$Windows NT$"
Provider="Custom Company"
CESignature="$Windows CE$"

[CEStrings]
AppName="CustomCABInstaller"
InstallDir=%CE1%\%AppName%

[Strings]
Manufacturer="Custom Company"

[CEDevice]
VersionMin=4.0
VersionMax=6.99
BuildMax=0xE0000000

[DefaultInstall]
CEShortcuts=Shortcuts
AddReg=RegKeys
CopyFiles=Files.Common1,Files.Common2
CESetupDLL="CustomCABSetup.dll"

[SourceDisksNames]
1=,"Common1",,"C:\Documents and Settings\JohnM\My Documents\Visual
Studio 2008\Projects\Samples\CustomCABInstaller\CustomCABSetup\Windows
Mobile 6.5.3 Professional DTK (ARMV4I)\Release\"
2=,"Common2",,"c:\Documents and Settings\JohnM\My Documents\Visual
Studio 2008\Projects\Samples\CustomCABInstaller\CustomCABApp\Windows
Mobile 6.5.3 Professional DTK (ARMV4I)\Release\"

[SourceDisksFiles]
"CustomCABSetup.dll"=1
"CustomCABApp.exe"=2
```

```
[DestinationDirs]
Shortcuts=0,%CE2%\Start Menu
Files.Common1=0,"%CE2%"
Files.Common2=0,"%CE1%"

[Files.Common1]
"CustomCABSetup.dll","CustomCABSetup.dll",,0

[Files.Common2]
"CustomCABApp.exe","CustomCABApp.exe",,0


[Shortcuts]
"CustomCABApp",0,"CustomCABApp.exe","%CE11%"

[RegKeys]
"HKLM","Software\JuniperSys\JSHome\Apps","App4","0x00000000","\Windows\
Start Menu\Programs\CustomCABApp.lnk"
```

16 Appendix E: Sample Gadget to Toggle Green LED

The following code implements a gadget for JSHome. This code is intended to be used in a Visual Studio 2008 project, with the Windows Mobile 6.5.3 DTK installed. It is also intended for use with JSAPI or the Juniper Systems API set. The project is available as a separate ZIP archive.

Gadget.cpp

```
// Copyright 2011 Juniper Systems, Inc.
// This example source code is only to be used with permission from
Juniper
// Systems under NDA.


#include <windows.h>
#include "JSHomeGadget.h"
#include "JSAPI.h"


// This is a very simple sample of how to create a Gadget for JSHome.
This
// sample simply toggles the green LED when the gadget is pressed.  The
notes
// above each function and in JSHomeGadget.h explain the interface
between
// JSHome, and the gadgets.  If there is something that doesn't make
sense, or
// if you have any questions, please contact us.
//
// This sample links to JSAPI only to toggle the green LED.  Linking to
JSAPI
// or including JSAPI.h is not needed for it to be a gadget in JSHome.


// The project settings deploy this and the two graphics to the
\Windows folder
// on the device.
//
// Manually add the following registry keys to have JSHome see this
gadget
```

```
//
[HKEY_LOCAL_MACHINE\Software\JuniperSys\JSHome\Gadgets\Example_Gadget]
// This subkey name doesn't matter
//      "DLL"="ExampleGadget.dll"       // If we were going to put our
gadget DLL somewhere besides the windows directory, we would probably
need the path included here are well.
//      "Name"="My Button Gadget"       // Name of the gadget that
shows up in the Gadget select menu.
//
// After changing the above registry key, you can either reboot, or
unload and
// reload JSHome (Start->Settings->Home->Items tab->Date->OK; Home-
>Items->Dashboard)
//
// Then to load this gadget, go to the JSHome menu and select
"Configure",
// click on the gadget location you want, and then select "My Button
Gadget".
//
// During development, after making code changes, deselect the gadget
from
// JSHome (so that the DLL unloads), deploy with the new changes, and
re-select
// this gadget back into JSHome.
//
// An installer may want to set this gadget to be one of the six
gadgets
// selected into the JSHome screen.  The following registry key
specifies the
// six gadgets that are selected.
//  [HKEY_LOCAL_MACHINE\Software\JuniperSys\JSHome\Gadgets]
//      "Gadget0"="ExampleGadget.dll"
// There are also values for "Gadget1" through "Gadget5".
//
// There is a JSHome reload event that can be triggered.  Setting this
event
// will caause JSHome to check the registry and reload the gadgets and
// application shortcuts.
// #define EVENT_JSHOME_RELOAD     TEXT("JSHOME_RELOAD")


// We actually recommend a size of 90x90 with the bottom part being a
// semi-transparent reflection of the icon, but as I was just after
some quick
// graphics I found online, I have 96x96 here.
// For a good example, look in the Windows directory on the device at
any of
// the JSHome_*.png files.
#define GRAPHIC_WIDTH       96      // Do not exceed GADGET_PIXEL_WIDTH
#define GRAPHIC_HEIGHT      96      // Do not exceed
GADGET_PIXEL_HEIGHT


BOOL g_bLEDon = FALSE;


// Nothing to do in here
BOOL APIENTRY DllMain(HANDLE hModule, DWORD  ul_reason_for_call, LPVOID
lpReserved)
{
    return TRUE;
}
```

```
// This is called once before JSHome loads this Gadget.
// This is our change to tell JSHome what we want, and how this gadget
is
// configgured.  We fill out the JS_HOME_GADGET_INFO structure here for
the
// first time.
// The only parts of the JS_HOME_GADGET_INFO structure that are
initialized
// before this function is called are the textColor, and the
hJSHomeWnd.
//
// Return
//      TRUE  - if we want to be loaded
//      FALSE - if there were problems.  JSHome will not load this.  It
will
//              also NOT call JSHomeGadgetDestroy(), so we would need
to do any
//              cleanup before returning.
extern "C" __declspec(dllexport) BOOL
JSHomeGadgetInit(JS_HOME_GADGET_INFO * pGadgetInfo)
{
    g_bLEDon = FALSE;

    pGadgetInfo->bPollingNOTevent = TRUE;   // Won't matter as we set
INFINITE and NULL below
    pGadgetInfo->dwPollInterval = INFINITE;
    pGadgetInfo->hEvent = NULL;

    pGadgetInfo->bPngFileNOTicon = TRUE;
    pGadgetInfo->dwResourceID = 0;
    memset(pGadgetInfo->wcName, 0, MAX_PATH*sizeof(WCHAR));
    swprintf(pGadgetInfo->wcName, L"\\Windows\\Gray-Button.png");

    pGadgetInfo->rcGraphic.left = (GADGET_PIXEL_WIDTH - GRAPHIC_WIDTH)
/ 2;
    pGadgetInfo->rcGraphic.right = pGadgetInfo->rcGraphic.left +
GRAPHIC_WIDTH;
    pGadgetInfo->rcGraphic.top = (GADGET_PIXEL_HEIGHT - GRAPHIC_HEIGHT)
/ 2;
    pGadgetInfo->rcGraphic.bottom = pGadgetInfo->rcGraphic.top +
GRAPHIC_HEIGHT;

    return TRUE;
}


// If JSHome exits or it loads a different gadget in place of this one,
this
// will get called to unload this gadget.
// This gives us a chance to shut down threads and cleanup.
extern "C" __declspec(dllexport) void JSHomeGadgetDestroy()
{
}


// This function is called after JSHome recieves our event
// (pGadgetInfo->hEvent), or the timeout occurs (pGadgetInfo-
>dwPollInterval).
// It we return TRUE from JSHomeGadgetColorChange, then this is called
// immediately after.
extern "C" __declspec(dllexport) BOOL
JSHomeGadgetGetImageInfo(JS_HOME_GADGET_INFO * pGadgetInfo)
{
    return FALSE;
```

```
}


// This function is called when the user presses (and releases
relatively
// quickly) on our gadget.  This gives us a chance to perform any
actions we
// want, and then to change the text or graphics being displayed.
// Typically the action to perform here is change the state of
something.
//
// Return TRUE to have JSHome to re-paint the graphics or text.  FALSE
// indicates that no graphics have changed.
extern "C" __declspec(dllexport) BOOL
JSHomeGadgetPressed(JS_HOME_GADGET_INFO * pGadgetInfo)
{
    g_bLEDon = !g_bLEDon;

    if(g_bLEDon)
    {
        swprintf(pGadgetInfo->wcName, L"\\Windows\\Green-Button.png");
        JSSetLEDStatusEx(LED_GREEN, LED_GREEN);
    }
    else
    {
        swprintf(pGadgetInfo->wcName, L"\\Windows\\Gray-Button.png");
        JSSetLEDStatusEx(0, LED_GREEN);
    }

    // Image changed, so return TRUE
    return TRUE;
}


// This function is called when the user presses and holds on our
gadget.  This
// gives us a chance to perform any actions we want, and then to change
the
// text or graphics being displayed.
// Typically the action here is to go to a control panel associated
with this
// gadget.
//
// Return TRUE to have JSHome to re-paint the graphics or text.  FALSE
// indicates that no graphics have changed.
extern "C" __declspec(dllexport) BOOL
JSHomeGadgetPressedAndHeld(JS_HOME_GADGET_INFO * pGadgetInfo)
{
    // No image change, so return FALSE
    return FALSE;
}


// When JSHome loses or gains visibility (not just focus), then it
calls this
// function.  This gives us a chance to pause worker threads so we
don't
// consume system resources for when we are in the background.
extern "C" __declspec(dllexport) void JSHomeGadgetVisible(BOOL
bVisible)
{
}
```

```
// If the user changes the color theme of JSHome, then the text Color
boxes
// get switched to the "preferred color".  This is called just after
that to
// give us a chance to overwrite that.  This would be useful if one of
our text
// boxes is on top of our graphic (such as a clock), and we always
wanted it to
// stay the same.
//
// If we do nothing here, then we will be drawing white text on a dark
// background or vica-versa depending on the color scheme of JSHome.
//
// Return TRUE to have JSHome call JSHomeGadgetGetImageInfo after this
returns.
// (Only needed if we changed the text color).
extern "C" __declspec(dllexport) BOOL
JSHomeGadgetColorChange(JS_HOME_GADGET_INFO * pGadgetInfo)
{
    return FALSE;
}
```

JSHomeGadget.h

```
// Copyright 2011 Juniper Systems, Inc.
// This example source code is only to be used with permission from
Juniper
// Systems under NDA.


#pragma once
#include <windows.h>


#define DBG_THREADS 0


#define GADGET_PIXEL_WIDTH          160
#define GADGET_PIXEL_HEIGHT         128

#define NUM_GADGET_TEXT_BOXES       4
#define GADGET_TEXT_LENGTH          100

typedef struct
{
    RECT        rcText;
    WCHAR       wcText[GADGET_TEXT_LENGTH];
    UINT        uFormat;
    COLORREF    textColor;      // JSHome will pass down the preferred
color to use here, the gadget has the option to change it
    LOGFONT     lf;
} JS_HOME_GADGET_TEXT_BOX;


// Used to pass info back and forth from gadget DLL and JSHome
typedef struct
{
    // bPollingNOTevent determines when JSHomeGadgetGetImageInfo gets
called.
    //      bPollingNOTevent    dwPollInterval      hEvent      *
Result *
    //      ------------------------------------------------------------
--
```

```
    //      FALSE               INFINITE            ourEvent    When we
set "ourEvent", then JSHome will call JSHomeGadgetGetImageInfo
    //      FALSE               5000                ourEvent    When we
set "ourEvent", or after 5 seconds, then JSHome will call
JSHomeGadgetGetImageInfo
    //      TRUE                5000                  n/a       Every 5
seconds, JSHome will call JSHomeGadgetGetImageInfo
    BOOL        bPollingNOTevent;
    DWORD       dwPollInterval;     // in ms
    HANDLE      hEvent;             // event that gets set if not
polling

    // JSHome can paint either a PNG from a file or an icon(.ico) from
a
    // resource DLL.  If bPngFileNOTicon is TRUE, then it looks for a
PNG file
    // located at wcName[].  If bPngFileNOTicon is FALSE, then it loads
an icon
    // from resource ID dwResourceID from the dll located at wcName[].
If
    // wcName is NULL, it will load the icon from this Gadget.
    // For a "Text Only" gadget, set wcName to NULL, bPngFileNOTicon to
FALSE and dwResourceID to 0.
    BOOL        bPngFileNOTicon;
    DWORD       dwResourceID;
    RECT        rcGraphic;          // Where on the gadget to display
the icon or png - should match the size of the PNG or icon as well.
    WCHAR       wcName[MAX_PATH];

    // text boxes (and colors)
    JS_HOME_GADGET_TEXT_BOX textBox[NUM_GADGET_TEXT_BOXES];
    HWND        hJSHomeWnd;
} JS_HOME_GADGET_INFO;
```