

2012

SJJ Shell Product Manual

Detailed Description & User Guide



Copyright © 2010, 2011, 2012 SJJ Embedded Micro Solutions, LLC, All Rights Reserved

No part of this guide may be copied, duplicated, reprinted, and stored in a retrieval system by any means, mechanical or electronic, without the written permission of the copyright owner.

First Release: January 2012

Published in the United States by

SJJ Embedded Micro Solutions, LLC.

6432 Glendale Dr.

Yorba Linda, CA 92886 USA

www.sjjmicro.com

Attempts have been made to properly reference all copyrighted, registered, and trademarked material. All copyrighted, registered, and trademarked material remains the property of the respective owners.

The publisher, author, and reviewers make no warranty for the correctness or for the use of this information, and assume no liability for direct or indirect damages of any kind arising from the information contained herewith, technical interpretation or technical explanations, for typographical or printing errors, or for any subsequent changes in this article.

The publisher and author reserve the right to make changes in this publication without notice and without incurring any liability.

Windows, .Net, Windows XP Embedded, Windows Embedded Standard 7, and Visual Studio are registered trade mark of Microsoft Corporation.

All trademarks and copyrights are property of their respective owners.

Table of Contents

1	WELCOME	4
2	SJJ SHELL PRODUCT DESCRIPTION	5
2.1	SJJ SHELL GUI.....	5
2.2	SJJ SHELL XML CONFIGURATION FILE: SJJShellConfig.xml	8
2.2.1	<i>Basic XML Configuration File Architecture</i>	10
2.2.2	<i>“Config” Element Details</i>	11
2.2.3	<i>“Button” Element Details</i>	14
3	UNPACKING THE SJJ_SHELL	17
4	USING THE SJJ SHELL WITH XPE / WES 2009	18
4.1	ADDING THE SJJ_SHELL.SLD FILE TO THE COMPONENT DATABASE	18
4.2	ADDING SJJ SHELL TO YOUR BUILD	19
5	USING THE SJJ SHELL WITH WINDOWS EMBEDDED STANDARD 7	21
5.1	ADDING THE SJJ SHELL FILES TO THE WES 7 DISTRIBUTION SHARE	21
5.2	ADDING SJJ SHELL TO YOUR BUILD	22
6	FEEDBACK AND TECHNICAL SUPPORT	28
A	BIBLIOGRAPHY	29
A.1	BOOKS:.....	29
A.2	WEBSITES.....	29

1 Welcome

Welcome to the SJJ Embedded Micro Solutions customizable Windows Embedded shell component, the SJJ Shell. Branding is an important part of embedded system products, i.e. giving the product a unique look and feel that helps identify the product and removes the desktop Windows look and feel. One of the ways to custom brand your embedded system product is to replace the standard Windows Explorer shell with a custom shell. We have been asked over the years to develop custom shells for different XP Embedded and Windows Embedded Standard 7 products. Most of the shells we have developed have some common items, but there are differences which require modification going from device to device. To make this process simpler, we wanted to create a shell that was easily configurable without code modification.

The SJJ Shell gives a contemporary look and feel. The active controls in the SJJ Shell are graphic buttons that can be touched, if a touchscreen used, or clicked with a mouse. These graphics are like shortcut icons on the desktop which launch other applications. Each graphic button has its own label. The screen is viewed as a grid with a configurable number of rows and columns. An arbitrary number of graphic buttons with their associated labels can be positioned on this grid.

Configurability without code modification was the driving design goal for the SJJ Shell. Therefore, customization is managed with a single configuration file. Within this configuration file the number of rows and columns on the screen grid are configurable. The graphic files that are used to paint the graphic buttons (.jpg files, .bmp files, etc.) are up to the user and specified in the configuration file. The location of each graphic button on the screen grid and the button's label text is customizable in the configuration file. Finally, the action that each button will take is customizable as well.

Having a base shell launch the main application to the embedded system has also been a popular customer request. SJJ_Shell has a configuration option to launch another application after startup. SJJ_Shell acts as a behind the scenes shell with the main application taking of the role of the primary shell. This allows the main application to be closed, upgraded or modified without losing user interface control capability. With the ability to configure what is in the SJJ_Shell GUI, you could put in shortcuts to administration functions like control panel, touch screen calibration tools, or other utilities. Administrators can then simply close the main application and access these administration functions through SJJ Shell.

The SJJ Shell provides a custom GUI look and feel to your embedded product. Once built into your Windows Embedded system, it can be customized and modified at will simply by modifying the configuration file and adding button graphic files. You can tune the look and feel for your specific product esthetics or you can even provide a customized experience for individual customers if you desire.

2 SJJ Shell Product Description

The SJJ Shell is a configurable shell that can be customized without having to do any code modifications, rebuilding of the shell, or rebuilding of your Windows Embedded OS build. The layout, look, and operations of the shell are customized through the use of a single XML configuration file. The following will describe each of the configuration parameters of the XML configuration file in detail.

2.1 SJJ Shell GUI

To best understand the configuration parameters, let's look at 3 examples of the shell screen layouts. All three configurations use the same 12 sample graphic files that are included as part of the SJJ Shell release to paint buttons, but each figure shows a configuration with a different number of rows and columns, different button positions for each of the graphic buttons and their associated labels, and a different background color.



Figure 2-1 - 4 Rows X 7 Columns, Blue BackColor

Figure 2-1 was configured for a maximum number of 4 rows and 7 columns and Blue background color. The positions of the graphic buttons were chosen to show how buttons can be distributed or grouped on the screen. Note the size of the buttons, the label text, and the separation between buttons.

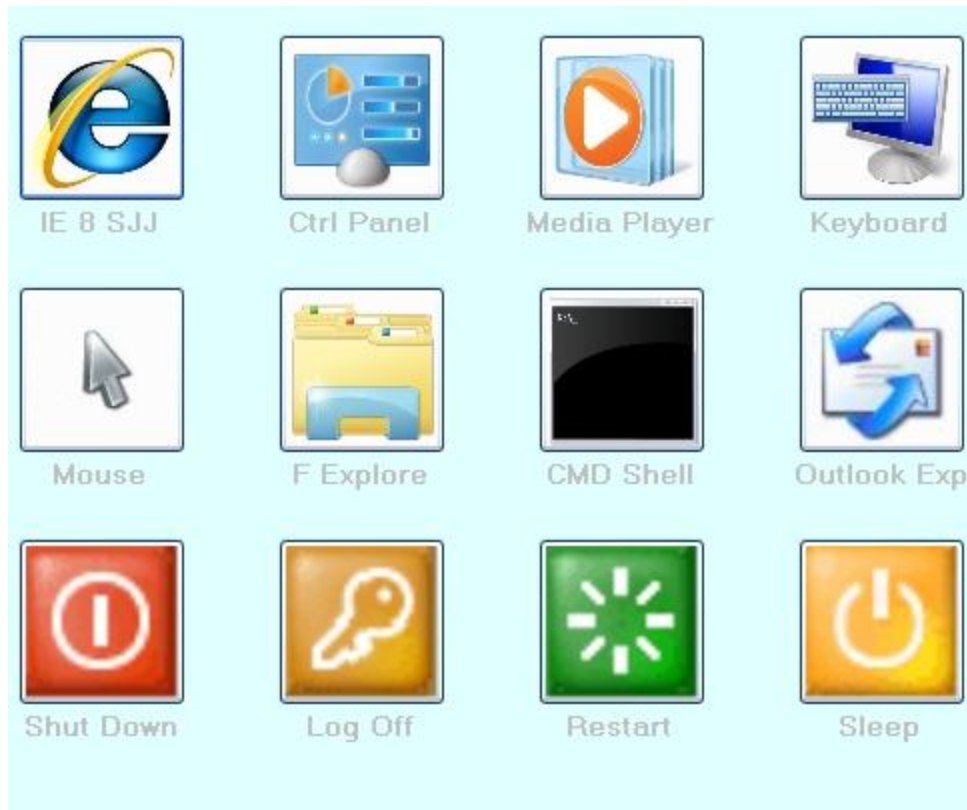


Figure 2-2 - 3 Rows X 4 Columns, LightCyan BackColor

Figure 2-2 was configured for a maximum number of 3 rows and 4 columns and LightCyan background color. The button positions were changed from the configuration in Figure 2-1 so that all button positions would place the buttons on the active display grid. Again, note the size of the buttons, the label text, and the separation between buttons as compared to the layout in Figure 2-1.



Figure 2-3 - 4 Rows X 3 Columns, OrangeRed BackColor

Figure 2-3 was configured for a maximum number of 4 rows and 3 columns and OrangeRed background color. Again, the individual button positions were modified to fit within the new screen grid. Now, compare the button size, the size of the label text, and the separation between buttons for this configuration with the previous two.

Button sizing, label text sizing, and button grid spacing are not configuration parameters. The SJJ Shell uses a best-fit algorithm to determine how to size and position the button objects and their labels. Once the grid size is determined by specifying the number of rows and columns, the shell calculates the maximum size for a square button and determines the vertical and horizontal spacing of the buttons. The font for the button labels is also scaled to match the button size. The columns and rows are then distributed evenly. If you compare Figure 2-1 and Figure 2-2, you can see that the buttons and their labels are scaled smaller for Figure 2-1 to accommodate the larger number of rows and columns. Both geometries allow for a uniform distribution of rows and columns. Now, compare Figure 2-2 and Figure 2-3. You can see that the grid dimensions of Figure 2-3 put some restrictions on the sizing. The larger number of rows puts a restriction on the vertical size of each button, which translates to a restriction on the horizontal size of the button, as well, because we are restricting the buttons to be square. With the small number of columns and the restriction on the horizontal size of the buttons, the column separation becomes relative large compared to the row spacing.

It is important to understand the nuances of this best-fit algorithm when you are choosing the grid size for your particular configuration. Now that we see how the button layout algorithm works and what the shell actually looks like when deployed, let's look at the details of the configuration file and see how we specify these and other parameters.

2.2 SJJ Shell XML Configuration File: SJJShellConfig.xml

The SJJ Shell configuration file must be named "SJJShellConfig.xml" and must be in the same folder as the SJJ Shell executable, SJJ_Shell.exe. The specifics of the file location will be discussed in the Windows Embedded build chapters that follow.

The following is a listing of the sample XML configuration file that is included with the SJJ Shell release. This is, in fact, the configuration file that was used for Figure 2-1.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This is a sample XML custom shell config file -->
<Items>
  <Config>
    <NumRows>4</NumRows>
    <NumCols>7</NumCols>
    <BackColor>Blue</BackColor>
    <ShellLauncher></ShellLauncher>
    <ShellLauncherArgs>-arg1 -arg2 -arg3</ShellLauncherArgs>
    <ShellStartIn></ShellStartIn>
    <License></License>
    <ClickMax>True</ClickMax>
    <RtClickExit>True</RtClickExit>
  </Config>
  <Button>
    <Image>C:\Shell\cmd.JPG</Image>
    <Name>CMD Shell</Name>
    <Row>2</Row>
    <Col>3</Col>
    <App>CMD.EXE</App>
    <AppArgs></AppArgs>
    <StartIn></StartIn>
    <Confirm>Yes</Confirm>
  </Button>
  <Button>
    <Image>C:\Shell\IE8.JPG</Image>
    <Name>IE 8 SJJ</Name>
    <Row>1</Row>
    <Col>1</Col>
    <App>iexplore.exe</App>
    <AppArgs>www.sjjmicro.com</AppArgs>
  </Button>
  <Button>
    <Image>C:\Shell\Explorer.JPG</Image>
    <Name>F Explore</Name>
    <Row>2</Row>
    <Col>2</Col>
    <App>explorer.exe</App>
  </Button>
  <Button>
    <Image>C:\Shell\Control.JPG</Image>
    <Name>Ctrl Panel</Name>
    <Row>1</Row>
    <Col>2</Col>
    <App>control.exe</App>
    <StartIn></StartIn>
  </Button>
  <Button>
    <Image>C:\Shell\WinMediaPlayer.JPG</Image>
```



```
<Name>Media Player</Name>
<Row>1</Row>
<Col>3</Col>
<App>wmplayer.exe</App>
</Button>
<Button>
  <Image>C:\Shell\mouse.jpg</Image>
  <Name>Mouse</Name>
  <Row>1</Row>
  <Col>5</Col>
  <App>control.exe</App>
  <AppArgs>Mouse</AppArgs>
</Button>
<Button>
  <Image>C:\Shell\Restart.bmp</Image>
  <Name>Restart</Name>
  <Row>4</Row>
  <Col>4</Col>
  <App>ShutDown.exe</App>
  <AppArgs>-r -t 5</AppArgs>
  <Confirm>Yes</Confirm>
</Button>
<Button>
  <Image>C:\Shell\LogOff.bmp</Image>
  <Name>Log Off</Name>
  <Row>4</Row>
  <Col>3</Col>
  <App>ShutDown.exe</App>
  <AppArgs>-l -t 5</AppArgs>
  <Confirm>Yes</Confirm>
</Button>
<Button>
  <Image>C:\Shell\ShutDown.bmp</Image>
  <Name>Shut Down</Name>
  <Row>4</Row>
  <Col>2</Col>
  <App>ShutDown.exe</App>
  <AppArgs>-s -t 5</AppArgs>
  <Confirm>Yes</Confirm>
</Button>
<Button>
  <Image>C:\Shell\onscreen_keyboard.JPG</Image>
  <Name>Keyboard</Name>
  <Row>1</Row>
  <Col>4</Col>
  <App>osk.exe</App>
</Button>
<Button>
  <Image>C:\Shell\OutlookExpress.JPG</Image>
  <Name>Outlook Exp</Name>
  <Row>1</Row>
  <Col>7</Col>
  <App>NullApp.exe</App>
  <AppArgs>Outlook Express</AppArgs>
  <StartIn></StartIn>
</Button>
<Button>
```

```
<Image>C:\Shell\Sleep.bmp</Image>
<Name>Sleep</Name>
<Row>4</Row>
<Col>5</Col>
<App>C:\Windows\System32\rundll32.exe</App>
<AppArgs>powrprof.dll,SetSuspendState</AppArgs>
<Confirm>Yes</Confirm>
</Button>
</Items>
```

2.2.1 Basic XML Configuration File Architecture

XML files basically consist of declarations, tags, elements, content, and comments. The first line of the sample XML configuration file is a declaration, and it specifies which XML version is being used in the file and what character encoding is being used. When you create your own configuration files, always start with the `<?xml version="1.0" encoding="utf-8"?>` declaration.

The next line in the sample XML configuration file is a comment. Comments are not processed. They are used for notation and documentation purposes. Comments can be placed anywhere in the configuration file except the first line. A comment may span one or more lines. A comment line starts with “`<!--`” and ends with “`-->`” with all the characters in between comprising the comment. Comments can also be used to temporarily exclude elements from being processed in the file, but leave the context behind as a comment field to indicate what was eliminated and allow it to be reactivated, later, without having to retype the entire element. For example, if I wanted to comment out the element:

```
<ClickMax>True</ClickMax>
```

I could edit it as follows:

```
<!-- ClickMax>True</ClickMax -->
```

And it would be treated as a comment. I could then edit it back to the original to reactivate the element.

The meat of the configuration information is organized and identified using XML tags. Tags start with the ‘`<`’ character and end with the ‘`>`’ character. There are start tags and end tags. An end tag is differentiated from a start tag by the inclusion of the ‘`/`’ character. So a start-tag/end-tag pair would look like the following:

```
<TagName></TagName>
```

The name of the tag appears in both the start tag and the end tag. The start-tag/end-tag pair defines an element. The element may be presented on a single line or it can extend over multiple lines. The content of an element is whatever is placed between the start tag and the end tag. Our `<TagName>` example above is an empty element because it has no content.

The first tag in the configuration file is the `<Items>` start tag. It is used to specify the area of the configuration file that contains all the configuration data. At this time only configuration data is contained in the file so the `</Items>` end tag is placed on the last line of the configuration. When the SJJ Shell reads the configuration file, it will look between the “Items” start tag and end tag for configuration data and ignore anything that would be placed after the “Items” end tag.

The next tag is the `<Config>` start tag. The contents between the `<Config>` start tag and the `</Config>` end tag are the basic shell configuration parameters. It does not contain any of the button configuration data.

Following the “Config” element are a series of “Button” elements. Between each <Button> start tag and </Button> end tag is the configuration data for a single button. There should be a “Button” element for every button that is to be placed and rendered on the SJJ Shell screen.

2.2.2 “Config” Element Details

The “Config” element contains sub-elements that specify all the general shell configuration parameters that are not related to a specific button. The following elements configure the shell and must be placed within the <Config> start tag and the </Config> end tag.

2.2.2.1 The Button Grid: Number of Rows and Columns

The button grid defines the total number of rows and columns of buttons to be displayed. There is a separate element for the number of rows and the number of columns. The desired decimal number is placed between the start and end tags. For example:

```
<NumRows>4</NumRows>
```

Specifies the total number of rows to be 4, and:

```
<NumCols>7</NumCols>
```

Specifies the total number of columns to be 7. Based on detail of the button art that you chose to paint your buttons with, the number of characters in the button labels, and the screen size and resolution of your target hardware’s display screen, there will be a practical limit to the button grid. Fortunately, because the SJJ Shell is so easy to configure, it will be easy to test various button grid layouts until you find the one that best suits your design.

2.2.2.2 Background Color

The background color of the shell can be set as one of the parameters in the <Config> element. It is entered as: <BackColor>*Color*</BackColor> where *Color* can be any of the following .NET Framework 3.5 Color Structure property names entered as a string (*Note that the colors are case dependent*):

AliceBlue	DarkGoldenrod	Gainsboro
AntiqueWhite	DarkGray	GhostWhite
Aqua	DarkGreen	Gold
Aquamarine	DarkKhaki	Goldenrod
Azure	DarkMagenta	Gray
Beige	DarkOliveGreen	Green
Bisque	DarkOrange	GreenYellow
Black	DarkOrchid	Honeydew
BlanchedAlmond	DarkRed	HotPink
Blue	DarkSalmon	IndianRed
BlueViolet	DarkSeaGreen	Indigo
Brown	DarkSlateBlue	Ivory
BurlyWood	DarkSlateGray	Khaki
CadetBlue	DarkTurquoise	Lavender
Chartreuse	DarkViolet	LavenderBlush
Chocolate	DeepPink	LawnGreen
Coral	DeepSkyBlue	LemonChiffon
CornflowerBlue	DimGray	LightBlue
Cornsilk	DodgerBlue	LightCoral
Crimson	Firebrick	LightCyan
Cyan	FloralWhite	LightGoldenrodYellow
DarkBlue	ForestGreen	LightGray
DarkCyan	Fuchsia	LightGreen

LightPink	Moccasin	Salmon
LightSalmon	NavajoWhite	SandyBrown
LightSeaGreen	Navy	SeaGreen
LightSkyBlue	OldLace	SeaShell
LightSlateGray	Olive	Sienna
LightSteelBlue	OliveDrab	Silver
LightYellow	Orange	SkyBlue
Lime	OrangeRed	SlateBlue
LimeGreen	Orchid	SlateGray
Linen	PaleGoldenrod	Snow
Magenta	PaleGreen	SpringGreen
Maroon	PaleTurquoise	SteelBlue
MediumAquamarine	PaleVioletRed	Tan
MediumBlue	PapayaWhip	Teal
MediumOrchid	PeachPuff	Thistle
MediumPurple	Peru	Tomato
MediumSeaGreen	Pink	Turquoise
MediumSlateBlue	Plum	Violet
MediumSpringGreen	PowderBlue	Wheat
MediumTurquoise	Purple	White
MediumVioletRed	Red	WhiteSmoke
MidnightBlue	RosyBrown	Yellow
MintCream	RoyalBlue	YellowGreen
MistyRose	SaddleBrown	

2.2.2.3 Shell Launcher

The SJJ Shell has the capability to launch a secondary shell. Thus the SJJ Shell would be like a boot loader for another shell application that was added after the Windows Embedded image was built with the SJJ Shell as the active shell. Therefore, if you want to add another shell to your configuration, SJJ Shell gives you that capability without having to rebuild the Windows Embedded image.

The secondary shell application is specified by a string between the <ShellLauncher> start tag and the </ShellLauncher> end tag. The string to specify the secondary shell application should be the full drive, path, and full application name with extension. For example:

```
<ShellLauncher>C:\NewShell\NewShell.exe</ShellLauncher>
```

Would launch the NewShell.exe application located at "C:\NewShell" as the active shell. SJJ Shell remains active in the background so you can exit the secondary shell and have SJJ Shell resume. SJJ Shell could be configured with shutdown, restart, logoff, and sleep buttons for example.

2.2.2.4 Shell Launcher Arguments

The secondary shell may require command line arguments to be passed to it when it is launched, and SJJ Shell has provisions to accommodate that. The element between the <ShellLauncherArgs> start tag and the </ShellLauncherArgs> end tag provides a string with the command line arguments that are to be passed to the secondary shell when it is launched. For example:

```
<ShellLauncherArgs>-arg1 -arg2 -arg3</ShellLauncherArgs>
```

Would pass the arguments: -arg1, -arg2, and -arg3 to the secondary shell application when it is launched.

2.2.2.5 Shell Launcher Starting Folder

Often when an application is launched, the default run-time working directory that is desired for the application is different from the folder where the application executable actually resides. SJJ Shell provides the capability for specifying a different start-in folder for the application. The element between the <ShellStartIn> start tag and the </ShellStartIn> end tag provides a string to specify the directory. Make the start-in directory string a full drive and path string. For example:

```
<ShellStartIn>C:\ShellStartDir</ShellStartIn>
```

Will launch the specified secondary shell from the directory specified for the executable file, but it will change the working directory for the secondary shell application to the "C:\ShellStartDir" directory.

2.2.2.6 License

The SJJ Shell is provided as a fully function shell for evaluation. In the evaluation mode, the SJJ Embedded Micro Solutions copyright notice will appear at the bottom of the screen. Once you decide to license the shell, SJJ will provide you with a license code. The element between the <License> start tag and the </License> end tag is the location to put your license code. For example, if the license code were "12345", then:

```
<License>12345</License>
```

Would provide the license code to the shell. Once your valid license code is entered, the SJJ copyright notice will be suppressed.

2.2.2.7 Full Screen/Windowed Switch

The SJJ Shell defaults to running full screen. Sometimes there is the need to run the system on monitors larger than the target system video monitor for testing purposes. There is a diagnostic mode that allows you to activate switching from full screen to a reduced video area that is about 25% of the full screen area. If you activate this mode, clicking or touching anywhere that is not a button will toggle the screen area from full screen to reduced screen area and back again. The element between the <ClickMax> start tag and the </ClickMax> end tag takes a string of "True" to activate this feature, or "False" to deactivate it. For example:

```
<ClickMax>True</ClickMax>
```

Will activate the feature.

If you leave the element empty:

```
<ClickMax></ClickMax>
```

Or remove this element from the configuration file, the default will be to deactivate this feature.

2.2.2.8 Shell Exit on Right Click

The final feature is to allow exiting of the shell if you right-click or use whatever right-click touch alternative is employed if you have a touch screen. Typically, you do not exit the shell, but shut down, log off, restart, or sleep the system from the shell. This usually is used as a diagnostic test feature when developing your system configuration. The element between the <RtClickExit> start tag and the </RtClickExit> end tag takes a string of "True" to activate this feature, or "False" to deactivate it. For example:

```
<RtClickExit>True</RtClickExit>
```

Will activate the feature.

If you leave the element empty:

```
<RtClickExit></RtClickExit>
```

Or remove this element from the configuration file, the default will be to deactivate this feature.

2.2.3 “Button” Element Details

For each button that is displayed on the shell grid, a button element must be included in the configuration file. The format and sub-elements are the same for all buttons. The contents of each sub-element is unique from button to button which customizes each button’s look and action. The button sub-elements are as follows:

2.2.3.1 *Button Image*

The button image specifies the graphic file that will be used to paint the button. It is specified by a string between the <Image> start tag and the </Image> end tag. The string to specify the graphic image file should be the full drive, path, and full graphic file name with extension. For example:

```
<Image>C:\Shell\Explorer.JPG</Image>
```

Will load the Explorer.JPG JPEG file located a “C:\Shell” and paint the button image area with it.

2.2.3.2 *Button Name*

The button name specifies the string that will be placed in the button label. It is specified by a string between the <Name> start tag and the </Name> end tag. For example:

```
<Name>F Explore</Name>
```

Will place the “F Explore” string into the button label below the button.

2.2.3.3 *Button Position*

The button position is set by two sub-elements, one that specifies the row for the button placement and one for the column for the button placement. Row and column are specified from the upper-left corner of the screen starting with row 1, column 1. The row specified must not exceed the maximum rows, and the column specified must not exceed the maximum columns set in the config section. If either the row or column exceed the maximum row and column, the button will not be displayed and a warning dialog will appear when the configuration file is being loaded so you know there has been an error.

The row is specified by an integer between 1 and the maximum rows. The row integer is placed between the <Row> start tag and the </Row> end tag. For example:

```
<Row>2</Row>
```

Sets the button row to 2.

The column is specified by an integer between 1 and the maximum columns. The column integer is placed between the <Col> start tag and the </Col> end tag. For example:

```
<Row>2</Row>
```

Sets the button row to 2.

2.2.3.4 *Button Application*

The button application specifies the application executable file that will be run when the button is clicked. The button application is specified by a string between the <App> start tag and the

</App> end tag. The string to specify the button application should be the full drive, path, and full application name with extension. If the path to the executable is in the system path, then the path does not need to be specified. For example:

```
<App> C:\Windows\System32\rundll32.exe</App>
```

Would run the rundll32.exe application located at “C:\Windows\System32” when the button is clicked.

2.2.3.5 *Button Application Arguments*

The button application arguments are command line arguments that are passed to the application when it is run. The application arguments are specified as a string between the <AppArgs> start tag and the </AppArgs> end tag. For example:

```
<AppArgs>powrprof.dll,SetSuspendState</AppArgs>
```

Specifies that the command line arguments “powrprof.dll,SetSuspendState” are passed to the application when it is run.

2.2.3.6 *Button Application Starting Folder*

The button application starting folder specifies a current working directory for the application when it is run. If this element is not provided for the button, then the current working directory will be the folder of the executable file for the application. The starting folder is specified as a string between the <StartIn> start tag and the </StartIn> end tag and should specify a full drive and path. For example:

```
<StartIn>C:\Shell</StartIn>
```

Will set the current working directory for application to “C:\Shell” when it is run.

2.2.3.7 *Button Activation Confirmation*

The button activation confirmation is an option that can be activated that will cause a confirmation dialog to be displayed when the button is clicked. This will give the chance to either confirm that the application should be run, or you can cancel running the application in case you decide that the button was clicked by mistake. This is usually activated with buttons that run critical applications, like shutting down the system, logging off, putting the system to sleep, etc. The button activation confirmation is enabled by place the string “Yes” between the <Confirm> start tag and the </Confirm> end tag. For example:

```
<Confirm>Yes</Confirm>
```

Will activate the confirmation dialog when the button is clicked. If the “CMD Shell” button was configured for confirmation, on clicking of the button, the confirmation dialog would be launched with the button name in the title bar, see Figure 2-4. If you click “OK” the button action will be carried out, and the application associated with the button will be run. If you click “Cancel” the button operation will be cancelled, no application will be run, and the confirmation dialog will close.

If the button activation confirmation element is missing, empty, or set to “No”, then clicking the button will immediately run the application.



Figure 2-4 - Button Confirmation

3 Unpacking the SJJ_SHELL

The SJJ Shell files are supplied in the “SJJ_SHELL Release Vx.x.x.x.zip” file, where “x.x.x.x” is the version number of the release. After you have unzipped the files to a folder in your system, you will see a folder and a SLD file as shown in Figure 3-1.



Name	Date modified	Type	Size
 SJJ_SHELL	12/29/2011 8:50 PM	File folder	
 SJJ_Shell.sld	12/29/2011 8:53 PM	Microsoft Compo...	104 KB

Figure 3-1 - SJJ_Shell Contents

The SJJ_SHELL is packaged for easy integration into either Windows XP Embedded / WES 2009 or Windows Embedded Standard 7 projects. The following two sections cover the details of working with each Windows Embedded product.

The SLD file is for Windows XP Embedded / WES 2009 and is linked to the files under the SJJ_SHELL folder. The folder and the files must be kept together for importing into the database.

The SJJ_SHELL folder is structured for a quick drop into a Windows Embedded Standard 7 distribution share. Here you will find the files for the SJJ_SHELL.EXE, NullApp.exe, the SJJShellConfig XML file, and some sample picture files.

- SJJ_Shell.exe is the main shell application.
- NullApp.exe is dummy application for testing buttons.
- SJJShellConfig.xml is the configuration file that SJJ_Shell.exe reads upon boot. The previous section discusses the structure and the changes you can make to the XML configuration file to customize your embedded system.

4 Using the SJJ Shell with XPE / WES 2009

For XPE / WES 2009, the SJJ_Shell.sld component is the shell component that includes all the resources to support running the SJJ_Shell applications. The following will describe how to add the SJJ_Shell.sld file to the component database and how to include the shell in your custom build.

4.1 Adding the SJJ_Shell.sld File to the Component Database

Once you have extracted the files from the “SJJ_SHELL Release Vx.x.x.x.zip” file, you can import the SLD file into the database.

1. Open Component Database Manager.
2. Click on the Import button.
3. Click on the button with the 3 dots ...
4. Locate the SJJ_Shell.sld file, Figure 4-1.

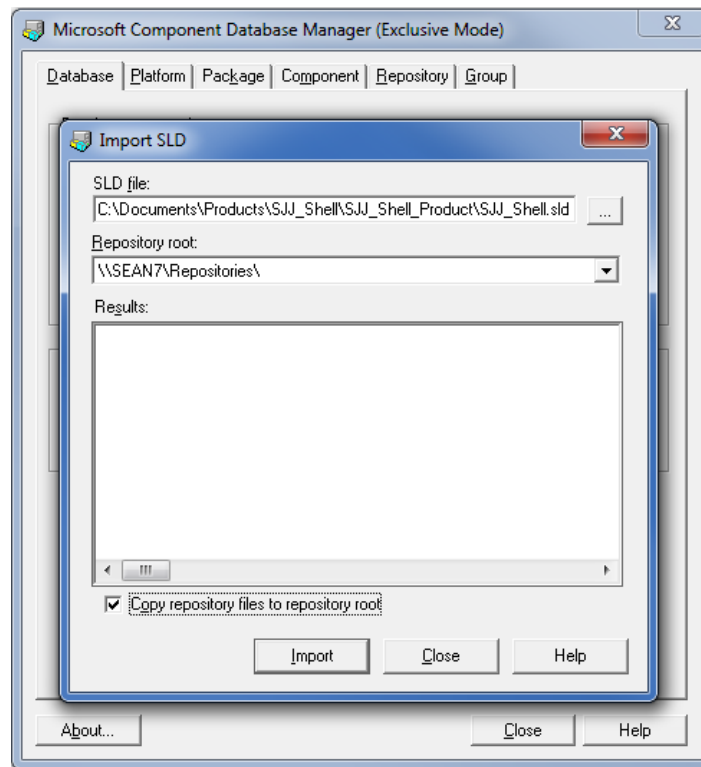


Figure 4-1 SJJ_Shell located

5. Make sure the “Copy repository files to repository root” is checked, and click the Import button.

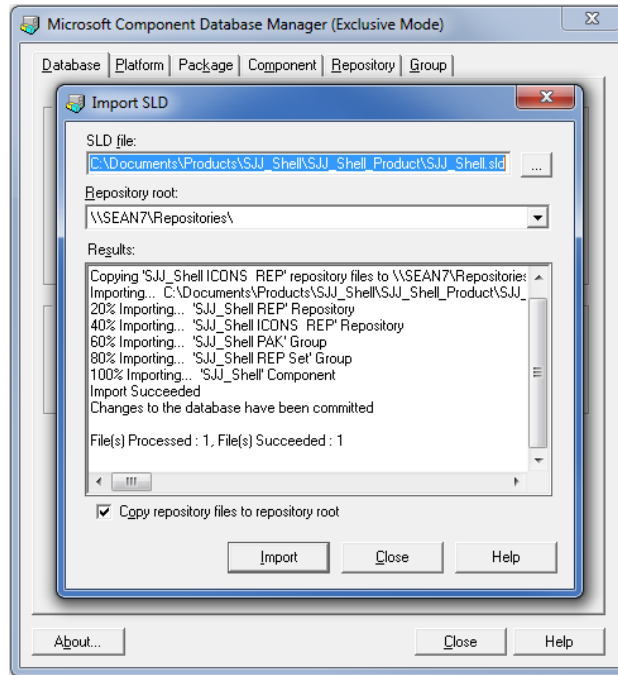


Figure 4-2 Successful Import

6. After the import completes, Figure 4-2, click the Close button.
7. Click the Close button to close Component Database Manager.

4.2 Adding SJJ Shell to Your Build

With the component and the files in the database and repository, you can create a configuration in Target Designer and add the SJJ_Shell component to the configuration.

1. Open Target Designer.
2. Create a New configuration and give it a name of your choosing.
3. Once the configuration has been created, navigate the component browser to the location of SJJ_Shell component, which can be found under Software->System->User Interface->Shells, Figure 4-3. You can alternatively search for the component.

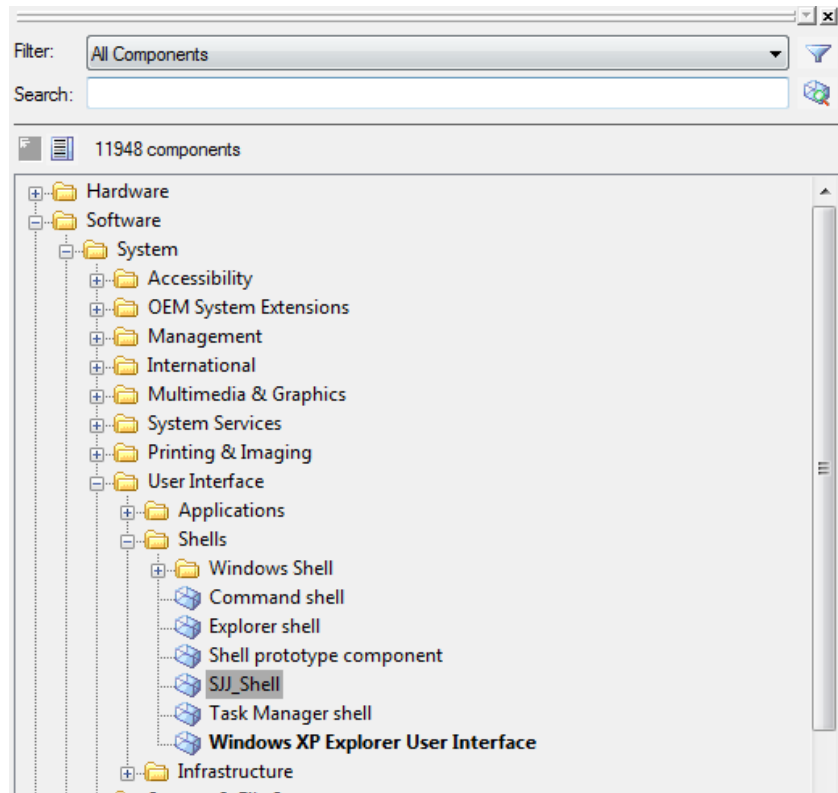


Figure 4-3 - SJJ_Shell in the Component Browser

4. Double click on the SJJ_Shell component to add it to the configuration

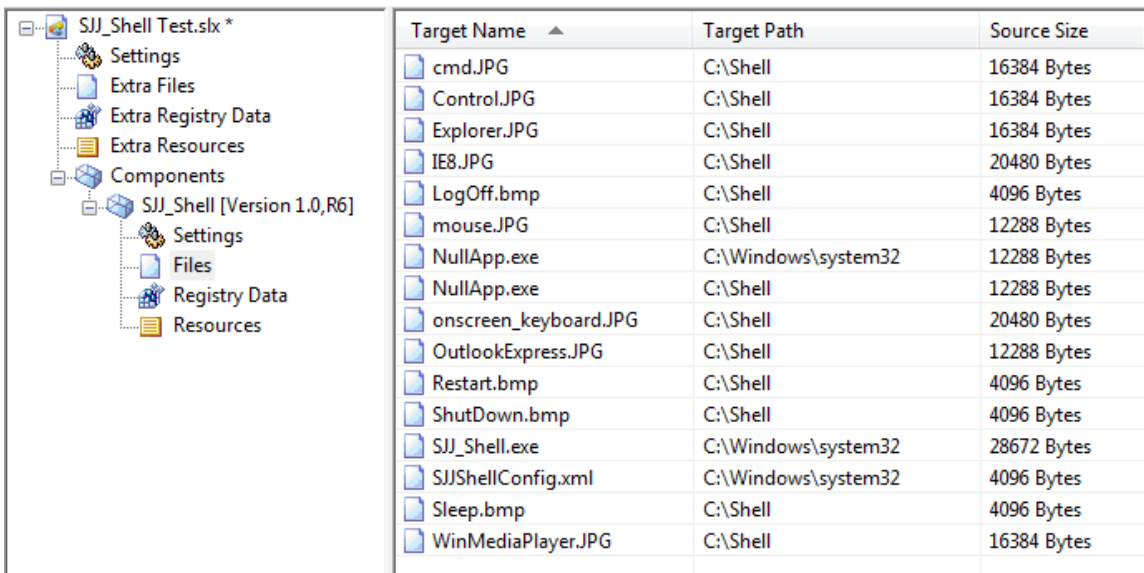


Figure 4-4 - SJJ_Shell is in the Configuration

You now have the SJJ Shell added to your configured to boot to the SJJ Shell, Figure 4-4. You can add the rest of the components, run dependency check with auto resolve enabled, save the configuration file and build the image.

5 Using the SJJ Shell with Windows Embedded Standard 7

For Windows Embedded Standard 7 (WES 7), there is a complete directory tree of all the files in the proper directories for the WES 7 Distribution Share. The following will describe how to place the SJJ Shell files into the WES 7 Distribution Share and how to include the shell in your custom build.

5.1 Adding the SJJ Shell Files to the WES 7 Distribution Share

Since WES 7 SP1 has been released, you may have as many as 4 distribution shares of the directory format:

C:\Program Files\Windows Embedded Standard 7\DSxxx

Where “DSxxx” can be:

DS	32-bit distribution share
DSSP1	32-bit Service Pack 1 distribution share
DS64	64-bit distribution share
DS64SP1	64-bit Service Pack 1 distribution share

The following example will be given for the DS64SP1 distribution share, but should be repeated for each distribution share that you will want to use. That is to say, if you want to be able to build from all 4 distribution shares and include the SJJ Shell, you will have 4 copies of the SJJ Shell file tree, one in each of the 4 distribution shares.

Each distribution share has an OEM folders tree named “\$OEM\$ Folders”. For example, the DS64SP1 distribution share has its OEM folders root at:

C:\Program Files\Windows Embedded Standard 7\DS64SP1\OEM\$ Folders

Each distribution share has a similar “\$OEM\$ Folders” directory entry point for OEM build files. It is under this folder that the SJJ Shell file directory tree needs to be. To copy the SJJ Shell directory tree into the WES 7 distribution share:

1. Find the directory where you copied the “SJJ_SHELL Release Vx.x.x.x.zip” file in File Explorer and select it in the Folders window.
2. In the files window you will see the root folder of the SJJ Shell distribution share directory tree, “SJJ_SHELL”.
3. Click on the “SJJ_SHELL” folder and drag it to the “\$OEM\$ Folders” directory of the distribution share in which you want to have the SJJ Shell available for inclusion in your builds.
4. Expand the folder in the distribution share and you will see the directory tree for SJJ Shell that is now part of the OEM folders, Figure 5-1.

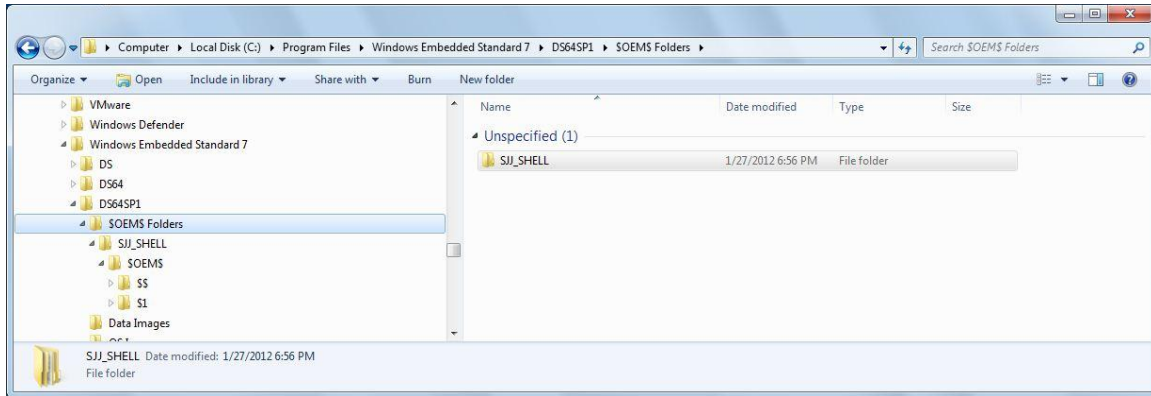


Figure 5-1 - SJJ Shell OEM Folders Directory Tree

5. Open the Image Configuration Editor (ICE).
6. Select the distribution share that you just copied the SJJ Shell files into (in this case DS64SP1).
7. Expand the distribution share and the “\$OEM\$ Folders”
8. You should see the “SJJ_SHELL” folder under the “\$OEM\$ Folders”, Figure 5-2.

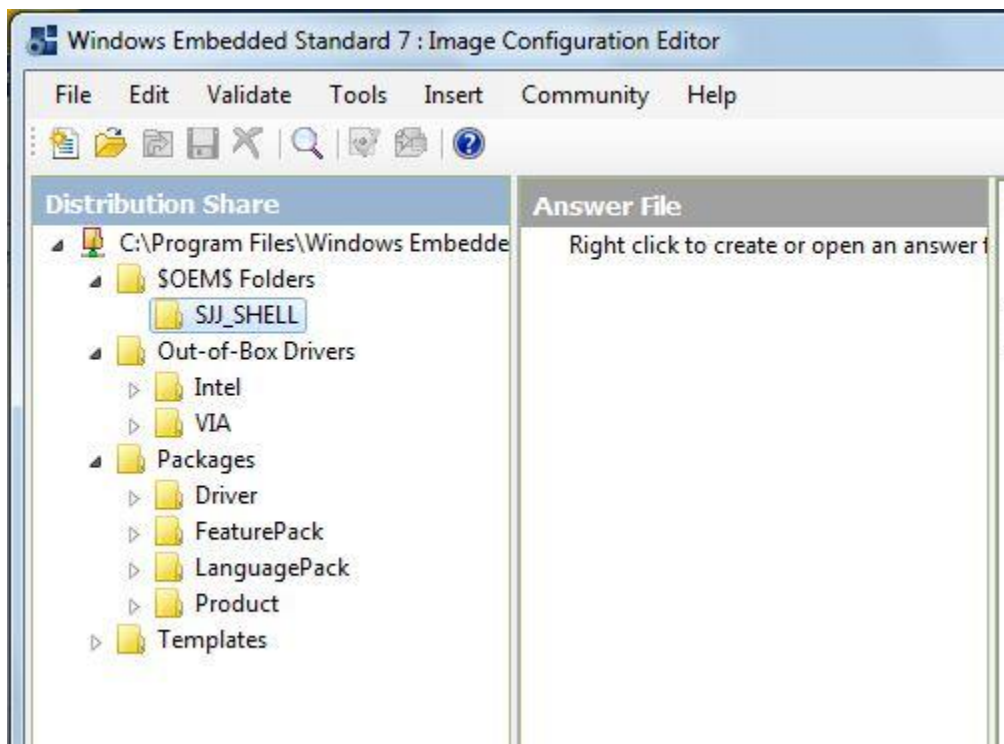


Figure 5-2 - SJJ Shell in Distribution Share

The SJJ Shell is now available to be included in your builds.

5.2 Adding SJJ Shell to Your Build

To add the SJJ Shell to your build, add the necessary support packages for the SJJ Shell, and to configure the build to boot to the SJJ Shell, do the following:

1. Open ICE and choose a distribution share that contains SJJ Shell.
2. In the Answer File pane, right click and create a new answer file, Figure 5-3.

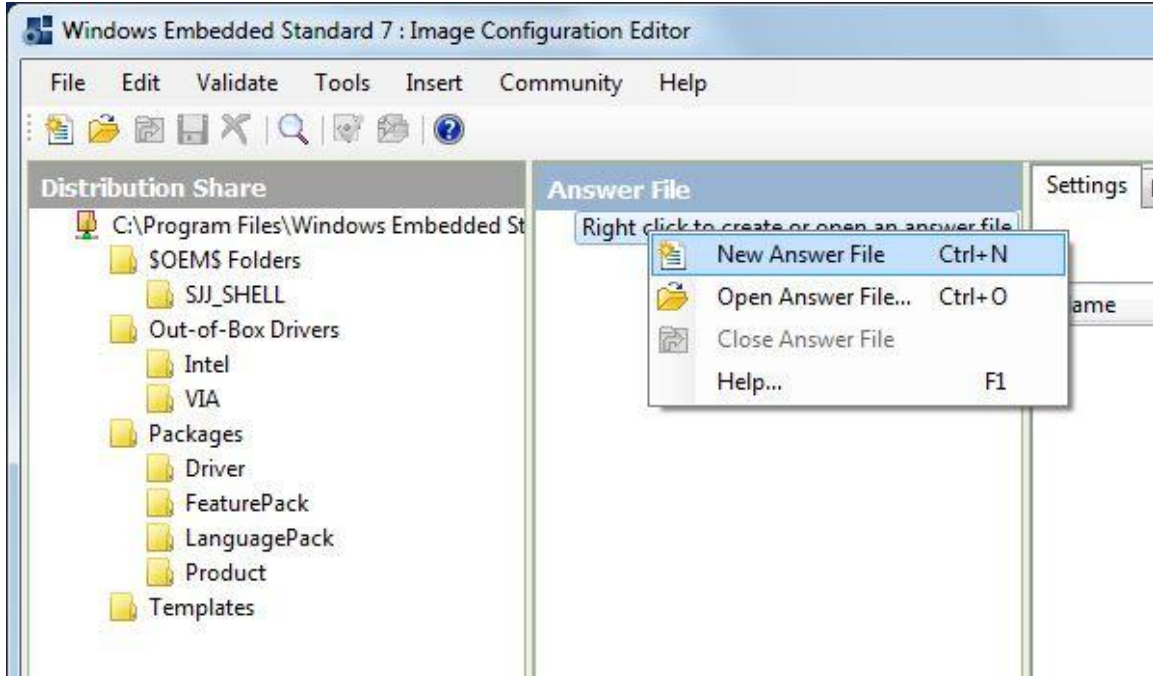


Figure 5-3 - Create New Answer File

3. In the Distribution Share, expand the “\$OEM\$ Folders” so that the SJJ_SHELL folder is visible, Figure 5-4

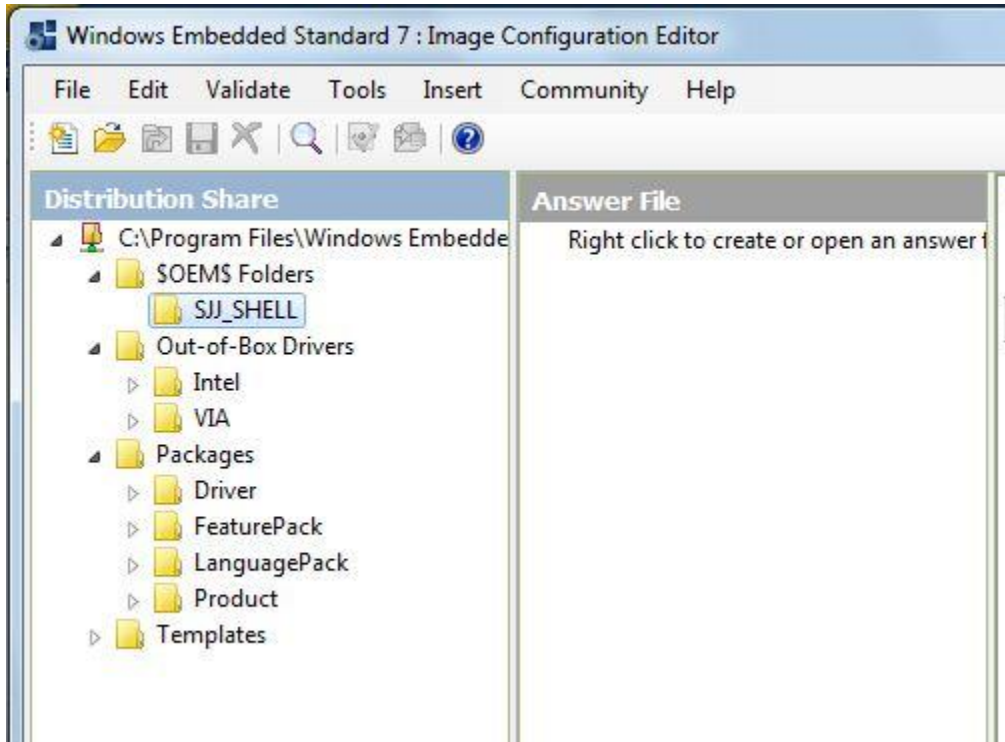


Figure 5-4 - Find SJJ_SHELL in \$OEM\$ Folders

- Right-click on the SJJ_SHELL folder and choose “Insert Oem Folders Path”, Figure 5-5. This adds the SJJ Shell to the new build image.

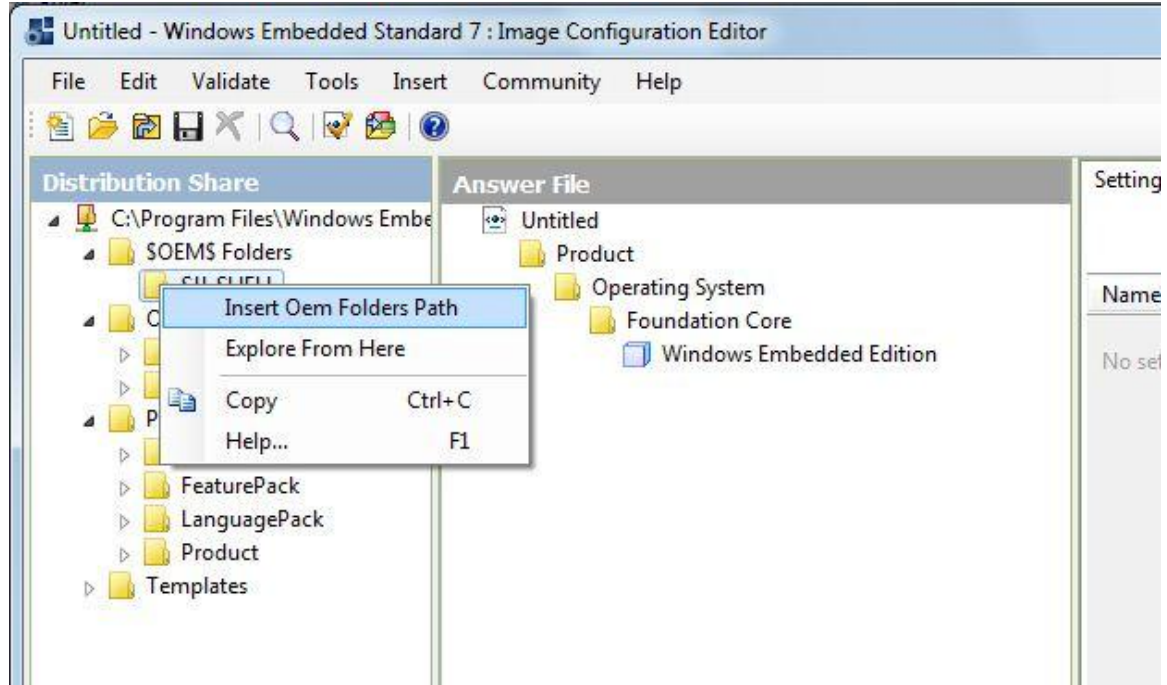


Figure 5-5 - Insert SJJ_SHELL \$OEM\$ Folders into Answer File

- The SJJ Shell requires .NET Framework 3.5 to run; so in the Distribution Share, expand Packages. Then under Packages expand FeaturePack. Under FeaturePack expand .NET Framework. Finally, under .NET Framework, right-click on .NET Framework 3.5 (**not** .NET Framework 3.5 Client Profile) and select “Add to Answer File” Figure 5-6.

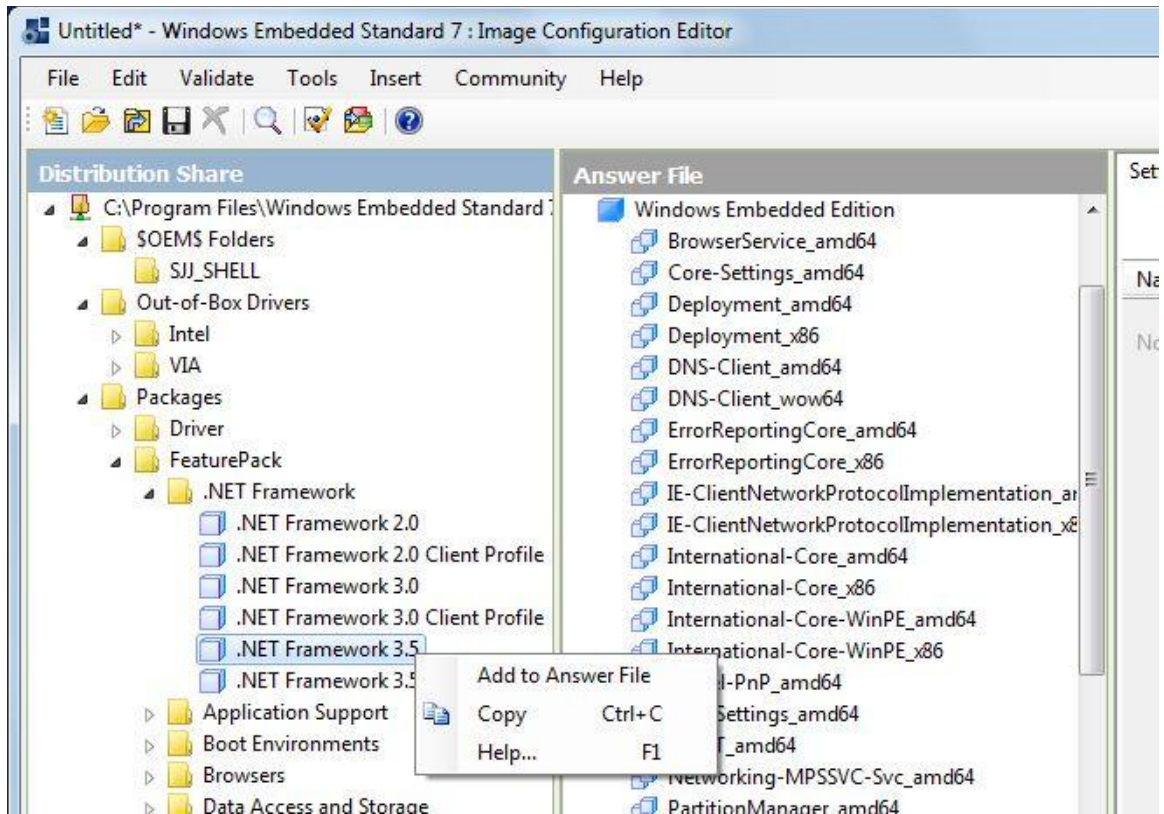


Figure 5-6 - Add .NET Framework 3.5 to Answer File

6. To use a custom shell, the “Command Prompt Shell with Custom Shell Support” package must be added. In the Distribution Share under Packages->FeaturePack. Expand “User Interface. Under “User Interface” right-click on “Command Prompt Shell with Custom Shell Support” and select “Add to Answer File”, Figure 5-7. This will add the Command Prompt Shell with Custom Shell Support to the answer file

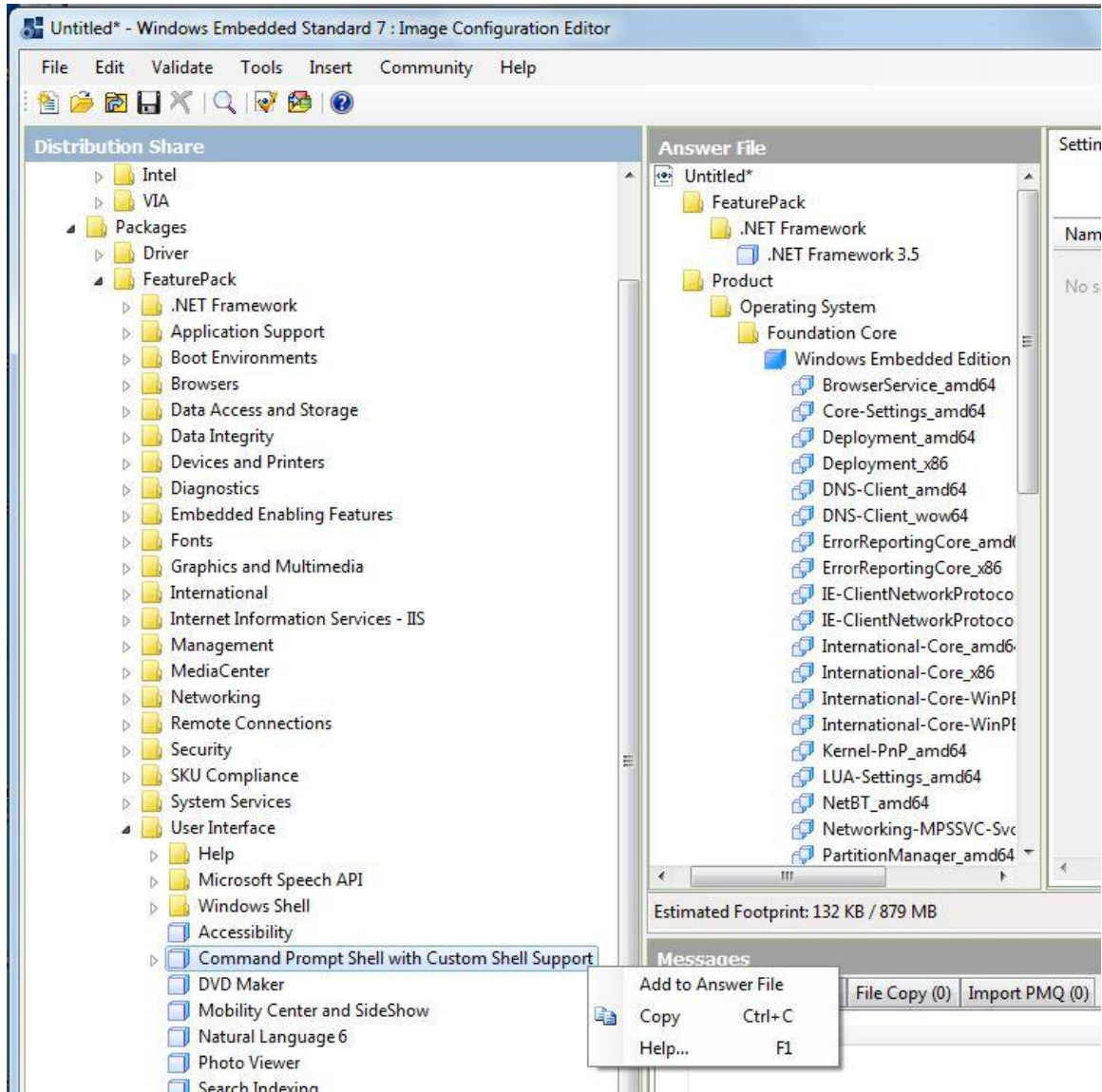


Figure 5-7 - Add Command Prompt Shell to Answer File

7. To configure the “Command Prompt Shell” to launch the SJJ Shell, in the Answer File pane under FeaturePack->User Interface, click on “Command Prompt Shell with Custom Shell Support”, Figure 5-8.

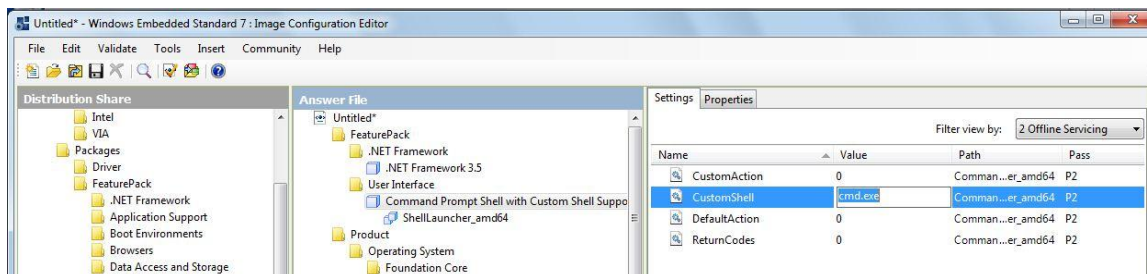


Figure 5-8 - Find Command Prompt Shell in Answer File

- In the properties tab to the right, you will see that the Command Prompt Shell properties are listed for Pass P2. In the “Filter view by” drop-down to the upper right, click the down arrow and select “2 Offline Servicing”, Figure 5-9.

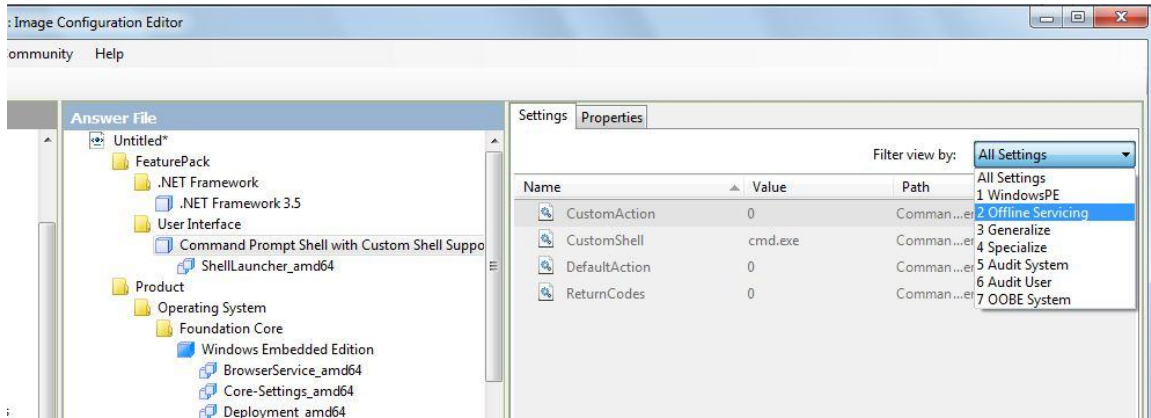


Figure 5-9 - Filter Command Prompt Shell Settings

- Click on the Value of the CustomShell setting, and change “cmd.exe” to “SJJ_SHELL.exe”, Figure 5-10.

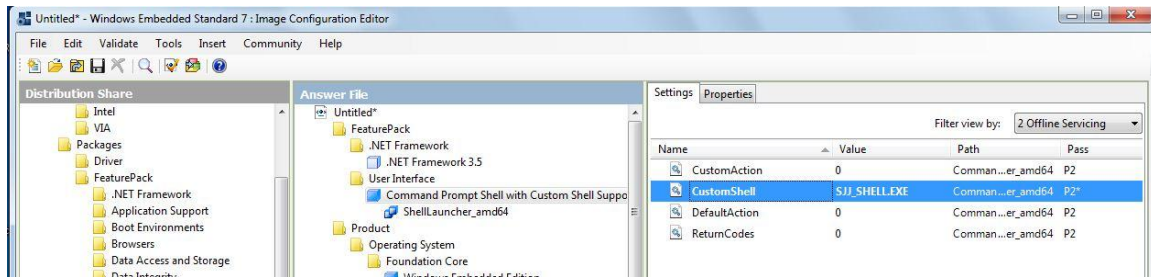


Figure 5-10 - Set CustomShell Setting to SHH_SHELL.EXE

You now have the SJJ Shell added to your build and configured to boot to the SJJ Shell. You can complete the build configuration, validate the configuration and add any required package dependencies, save the answer file and build your final configuration.

6 Feedback and Technical Support

We are always looking to add new features. If you have suggestions or need technical support, please use our contact page:

<http://www.sjjmicro.com/Contact.html>

A Bibliography

For additional information check out the following books and websites:

A.1 Books:

Professional's Guide to Windows Embedded Standard 7, Sean D. Liming, Annabooks, 2010, ISBN-10: 0-9842801-1-1; ISBN-13:978-0-9842801-1-7.

http://www.annabooks.com/Book_PGWES7.html

Windows XP Embedded Supplemental Toolkit, Sean D. Liming, Cedar Hill Publishing, 2005, ISBN 1-932373-96-9

http://www.annabooks.com/Book_XPES.html

Available on Kindle: <http://www.amazon.com/dp/B005IZDAD6>

Windows XP Embedded Advanced, Sean D. Liming, RTC Books, 2003, ISBN: 0-929392-77-9

http://www.annabooks.com/Book_XPEA.html

Available on Kindle: <http://www.amazon.com/dp/B005LKMRTQ>

Real-Time Development from Theory to Practice, John R. Malin and Sean D. Liming, Annabooks, 2009, ISBN-13: 978-0-9842801-0-0

http://www.annabooks.com/Book_RTD.html

A.2 Websites

Annabooks: <http://www.annabooks.com/>

SJJ Embedded Micro Solutions: <http://www.sjjmicro.com/>