

OSD++

version 1.0.0.1b

document version 1.0.0.1

Sep 8, 2009

Jason Sandys

WHAT IS IT

OSD++ is a better way to get input from the user and populate task sequences variables. It retrieves values from the registry, queries WMI, and/or prompts the user for input via an input box or a drop-down box. All values retrieved are stored in internal OSD++ variables that can be used as replacement variables in further data retrieval. They can also be written out to the registry or used to set task sequence variables.

OSD++ variables can also be persisted in a file that can be loaded by OSD++. This is useful during refresh scenarios to display the interface to the user before the task sequence begins, save their responses to the variable file, and then using another run of OSD++, output the saved responses to task sequence variables.

WHERE CAN I GET IT

OSD++ is available via my blog at <http://myitforum.com/cs2/blogs/jsandys/osdplusplus.aspx>.

Please send all bugs, features requests, comments, etc. to me using the contact form on the blog.

CHANGE LOG

1.0.0.1b Release

- Initial beta release

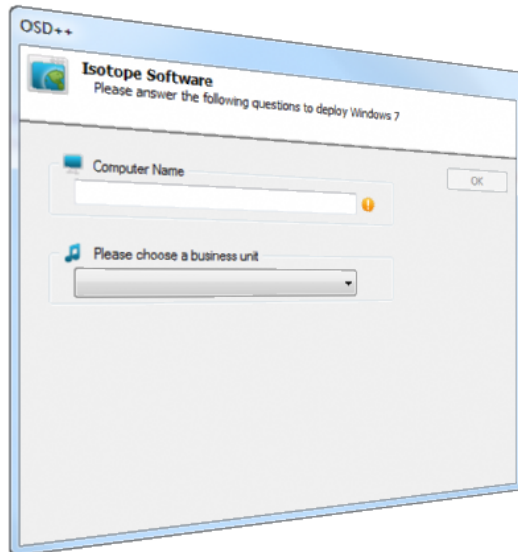
USAGE

Inside an OSD Task Sequence, OSD++ must be run from a command-line task to show any UI. Because of "by design" limitations, a Software Install task cannot show any UI or be interactive in any way and command-line tasks can only be interactive during the Windows PE portion of the task sequence.

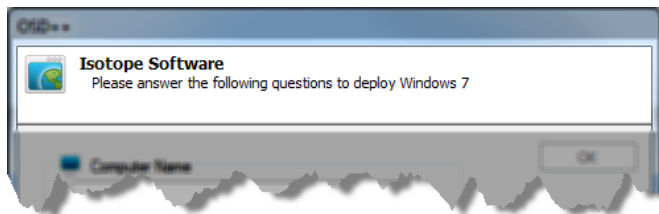
One problem with the above limitations is in refresh scenarios where you would like to present the UI to the user before the task sequence is initiated or wipes the hard drive. To get around this, OSD++ can be run separately before the task sequence using a chained program that the task sequence depends on. OSD++ will save the values entered by the user in a file. A second run of OSD++ at the beginning of the task sequence will load the values from the file and save them as task sequence variables.

DIALOGS

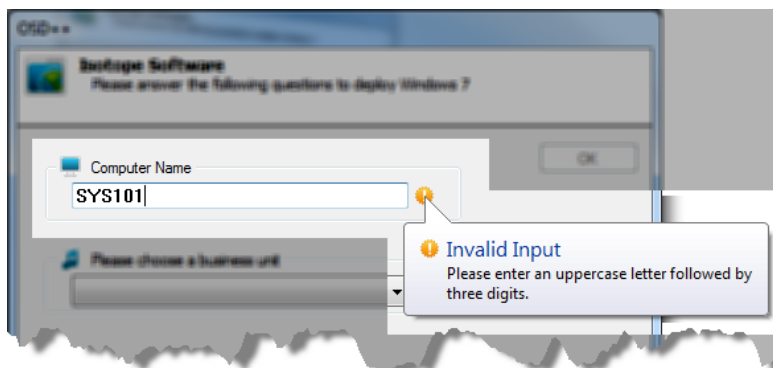
User input can take the form of one or more dialog boxes each using one or more input boxes or drop-down lists.



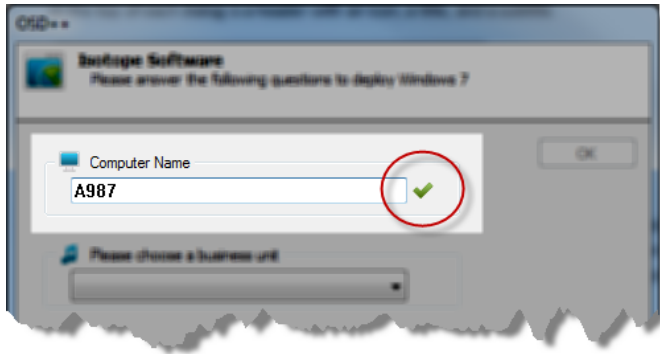
At the top of each dialog is a header with an icon, a title, and a subtitle.



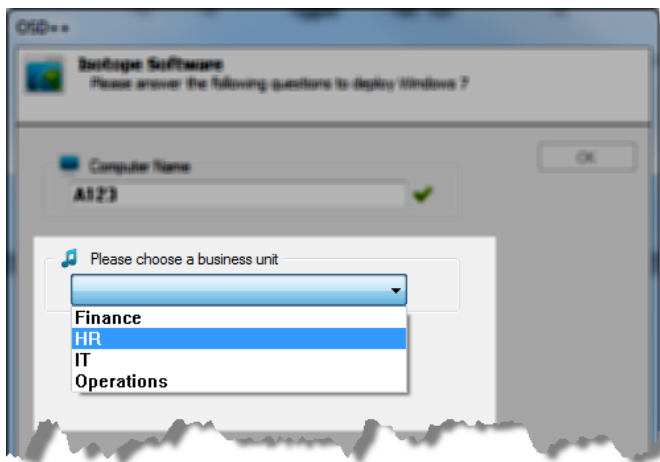
Each input box has its own icon and prompt. Text in input boxes can also be validated using a regular expression. If input does not conform to the regular expression, an exclamation mark will be shown after the input box. Hovering over this exclamation icon will reveal a configurable message indicating why the input is not acceptable.



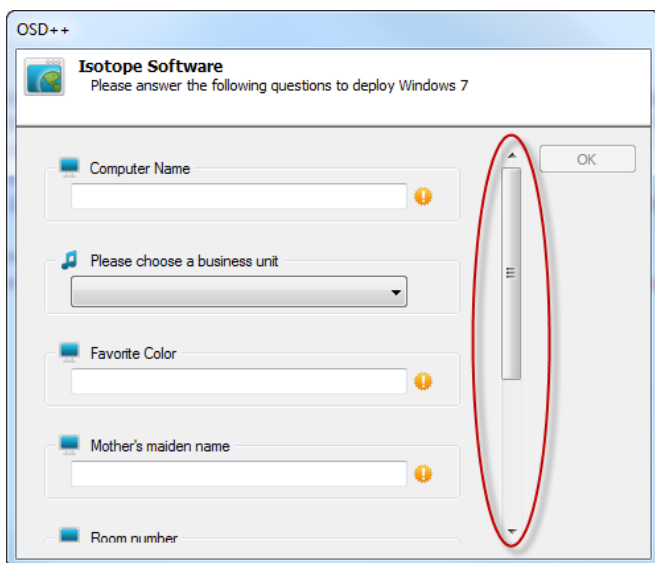
Once the input passes the regular expression check, a green check mark will replace the exclamation mark.



Drop down lists also have their own icon and prompt and aren't considered valid until a value is chosen. There is no visual indicator for a valid drop down box.



If you put in more input and dialog boxes than will fit on the dialog, a scroll bar will appear allowing the user to scroll the dialog.



The OK button will remain disabled until all input boxes have matched their respective regular expressions and all drop-down lists have a valid selection.

OSD++ is not limited to a single dialog box. Multiple dialog boxes can be presented. This may be useful to group specific types of requests together or to prompt for a piece of information, retrieve a value from WMI or the registry using the user's input, and then further prompt the user based on the information from WMI or the registry.

REGISTRY

Values can be retrieved and put back into the registry. There is no UI for this task and there is no limit to the number of values that you can work with. Values retrieved are stored as strings.

Values written to the registry can be written as either string (REG_SZ) or number (REG_DWORD) values. If a key does not exist, it can be automatically created.

WMI

Values can only be retrieved from WMI. There is no UI for this task and there is no limit to the number of values that you can work with. Values retrieved are stored as strings.

OSD++ VARIABLES

All values retrieved are stored as strings in OSD++ variables. These are similar to task sequence variables but are internal to OSD++. They are stored as named value pairs and indexed using a key just like task sequence variables. Thus, every value that is retrieved is stored using a defined name.

OSD++ variables can also be used as replacement variables for any value specified in the configuration file. OSD++ will replace any occurrence of an OSD++ variable name surrounded by percent signs in the configuration file with the actual value of that variable. Thus if you have an OSD++ variable named **CurrentVersion** and you define the following in the title attribute for a dialog, "Upgrade from Windows %CurrentVersion%", OSD++ will replace %CurrentVersion% with whatever value **CurrentVersion** represents.

As stated above, OSD++ variables can be also saved to a file and then reloaded during a future run of OSD++.

The final use of OSD++ variables is to directly write them out to task sequence variables for use elsewhere within the task sequence as conditionals or input to tasks.

CONFIGURATION

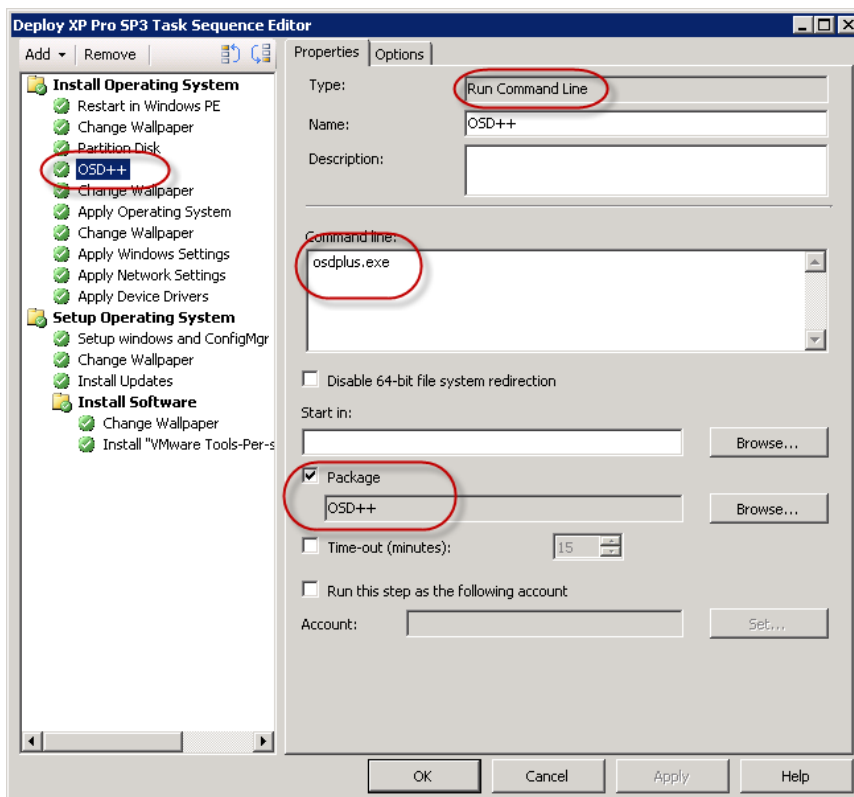
Create a configuration file named *OSDPlusConfig.xml*. To load an alternately named file, use the command-line parameter discussed below. See the "Configuration File" section for details.

1. Testing
 - a) Simply run OSD++ by double-clicking on the icon or from the command-line. The command-line parameters specified below are valid.
 - b) No visible output will be generated, check the log file for results. See the "Log File" section for details.
 - c) Task sequence variables cannot be created when running outside of a task sequence.

2. Production

- a) Make the OSDPlus.exe files available to the task sequence. This includes *OSDAppChooser.exe* and your configuration file -- the default name for the configuration file is *OSDPlusConfig.xml* but this can be changed using a command-line parameter (see Optional command-line parameters below).

There are multiple ways to do this but I prefer to put them into a software distribution package -- no program is necessary. This has the advantage in larger distributed environments of ensuring that the client accesses the files from the closest DP. The disadvantage is that you have to remember to update the DP anytime you update the configuration file. An alternative is to place the files in a shared folder and use a map folder task to make them available. This has the advantage of not having to update the DP every time you make a change. It also has an advantage when your TS is set to download and execute because the command-line task to run OSD++ (see the next step) can be run before the partition and format task. A hybrid approach is also possible where you put OSDPlus.exe in a package and make the configuration file available using a shared folder.



- b) Create a command-line task to run *OSDPlus.exe*. If you used a package for step 1, reference the package you placed the files in this task. If you used a shared folder, make sure you used a map folder task before this one and prefix *OSDPlus.exe* with the drive letter you mapped in that task.
- ## 3. Optional command-line parameters:
- **/vsave:<filename>** The filename to save OSD++ variables to. %Temp%\vars.osd is the default.
 - **/vload:<filename>** The filename to load OSD++ variables from. %Temp%\vars.osd is the default.
 - **/conf:<filename>** The filename of the configuration file to use. This can include a path if necessary. If not specified, *OSDPlusConfig.xml* will be used.

An advantage of using command-line arguments is that you can use other TS variables to set their values. For example, if you have multiple configuration files, you can populate a TS variable named MyConfig with the filename and then use the following command-line: OSDPlus /conf:% MyConfig%

LOG FILE

OSDPlus will generate or append to a log file name *OSDPlus.log* every time it runs. This log records major events including error events. If you launch OSDPlus and nothing happens, check this log file for details. To prevent any ugliness from being presented to an end user, everything is sent to this file without notifying the interactive user.

The log file is a standard ConfigMgr log file and is best viewed using Trace32 from the toolkit. If run inside a TS, the log file is located with the standard TS log file SMSTS.log: if you place the task at or near the beginning of the TS as described in the usage section, this will be *X:\Windows\Temp\SMSTS*. If you run OSD++ outside of a TS, the log file will be located in the current user's temp directory which can easily be located using the environment variable *%TEMP%*.

CONFIGURATION FILE

The configuration file is XML based. The following table describes the valid elements. Note that OSDPlus does not check the configuration file against an XSD schema so will most likely ignore additional or out of place elements. This is not guaranteed though, so definitely test your configuration file outside of OSD before using it. Also keep in mind that XML is case-sensitive.

Element Name	Valid Attributes	Valid Parents	Valid Children	Comments
OSDPlus	N/A	N/A	GetReg GetUser GetWmi PutReg PutTSVar	This is the root element of the file.
GetReg	hive – The registry hive to retrieve a value from, valid values include <i>HKLM, HKCU, HKU, and HKCR</i> . key – The registry key to retrieve a value from value – The value to retrieve from the registry varname – The OSD++ variable name to store the value in	OSDPlus	N/A	Retrieves the value from a registry key defined by the attributes. Note that 64-bit redirection is disabled in OSD++.
GetUser	title – The title to display in the header of the dialog box subtitle – The subtitle to display in the header of the dialog box icon – The icon to display in the header of the dialog box	OSDPlus	Input	Retrieves values from the interactive user using a dialog box.
GetWmi	namespace – The namespace to retrieve a value from class – The class to query for a value	OSDPlus	N/A	Retrieves values using a WMI query. If the property specified has multiple values, only the first is

	<p>property – the property to retrieve a value from</p> <p>varname – The OSD++ variable name to store the value in</p>			returned.
PutReg	<p>hive – The registry hive to write a value to, valid values include <i>HKLM</i>, <i>HKCU</i>, <i>HKU</i>, and <i>HKCR</i>.</p> <p>key – The registry key to write a value to</p> <p>value – The value name to write in the registry</p> <p>valtype – The type of registry value to write, valid values include <i>string</i> and <i>number</i></p> <p>createkey – Specifies whether to create the key if it does not already exist, valid values include <i>yes</i> and <i>no</i></p> <p>varname – The OSD++ variable name to write to the registry</p>	OSDPlus	N/A	Puts the value of an OSD++ variable into the registry.
PutTSVar	<p>varname – The OSD++ variable name to write to the registry</p> <p>tsvar – The name of task sequence variable to populate</p>	OSDPlus	N/A	Puts the value of an OSD++ variable into a task sequence variable. This is only valid if OSD++ is run from within a task sequence.
Textbox	<p>prompt – The prompt to display above the textbox</p> <p>icon – The icon to display with the prompt</p> <p>helptxt – The text to display when hovering over the exclamation icon when the input does not match the regular expressions</p> <p>regex – The regular expression to validate the textbox input against</p> <p>varname – The OSD++ variable name to store the input in</p>	GetUser	N/A	
Listbox	<p>prompt – The prompt to display above the listbox</p> <p>icon – The icon to display with the prompt</p> <p>varname – The OSD++ variable name to store the value in</p>	GetUser	Option	
Option	<p>default – Makes this option selected by default</p> <p>alt1 – The first alternate value when this option is selected</p> <p>alt2 – The second alternate value when this option is selected</p>	Listbox	N/A	

Note that all element and attribute names are case sensitive.

FEATURE CONFIGURATION & EXAMPLES

REGISTRY

The following snippet reads the `CurrentVersion` value from the key `SOFTWARE\Microsoft\Windows NT\CurrentVersion` in the HKLM hive:

```
<GetReg hive="HKLM" key="SOFTWARE\Microsoft\Windows NT\CurrentVersion"
value="CurrentVersion" varname="CurrentVersion" />
```

The following snippet stores the value of the OSD++ variable named `BusinessUnit` into the registry in the HKLM hive, in the key `SYSTEM\Isotope`, in the string value named `CorporateBusinessUnit`. It will create the key if it does not already exist. Note that values names and keys in the registry can contain spaces.

```
<PutReg hive="HKLM" key="SYSTEM\Isotope" value="CorporateBusinessUnit"
valtype="string" createkey="yes" varname="BusinessUnit" />
```

WMI

The following snippet queries WMI for the model number of the system.

```
<GetWmi namespace="root\cimv2" class="Win32_ComputerSystem"
property="model" varname="csmodel" />
```

TASK SEQUENCE VARIABLE

The following snippet puts the value of the `BusinessUnit` OSD++ variable into the task sequence variable names `tsbusunit`.

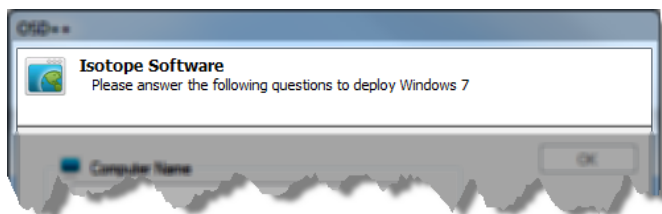
```
<PutTSVar varname=" BusinessUnit" tsvar=" tsbusunit" />
```

USER INPUT

The following snippet will create a dialog box with the specified information in the header:

```
<GetUser title="Isotope Software" subtitle="Please answer the following
questions to deploy Windows 7" icon="1">

</GetUser>
```



Dialog boxes can contain as many input and list boxes as you need. A scrollbar will automatically appear if the number of input boxes fills up the dialog box.

TEXTBOX

Text boxes are for free form user input in response to a question or prompt. Text in an input box can be validated against a regular expression. If the text does not match the regular expression, a warning exclamation mark will appear after the text box. Hovering over this exclamation mark will provide a popup with further information.

The following snippet creates a textbox prompting the user for the name to assign to the computer, validates it against a regular expression, and assigns the value to the *CompName* OSD++ variable.

```
<Textbox prompt="Computer Name" regex="[A-Z]\d{3}.*" varname="CompName"
icon="45" helptxt="Please enter an uppercase letter followed by three
digits."/>
```

Regular expressions are beyond the scope of this documentation but many excellent articles and tutorials are available on the web including <http://www.regular-expressions.info/reference.html>.

LISTBOX

List boxes provide a drop down for the user to choose from based on a question or prompt.

The following snippet creates a listbox prompting the user to choose a business unit and putting the value in an OSD++ variable named *BusinessUnit*:

```
<Listbox prompt="Are you in the finance department?" icon="46"
varname="finance">

</Listbox>
```

The choices presented in the listbox are defined by Option sub-elements. Based on the selection made by the user, the values of three OSD++ variables are populated: the main value that is the same as the selection made and two alternates. The value of the selection is placed in an OSD++ variable named using the varname attribute while the alternate values are placed in OSD++ variables named by concatenating *alt1* and *alt2* on the end of the varname attribute. The following snippet shows two options:

```
<Listbox prompt="Are you in the finance department?" icon="46"
varname="BusinessUnit">

    <Option alt1="1" alt2="True">Yes</Option>

    <Option alt1="0" alt2="False">No</Option>

</Listbox>
```

If the first option is chosen, the following OSD++ variables are set:

- Finance = Yes
- Financealt1 = 1
- Financealt2 = True

ICONS

The following table shows all the icons available and their indexes. These icons can be used for dialog box headers, list boxes, or text boxes.

0		1		2		3	
4		5		6		7	
8		9		10		11	
12		13		14		15	
16		17		18		19	
20		21		22		23	
24		25		26		27	
28		29		30		31	
32		33		34		35	
36		37		38		39	
40		41		42		43	
44		45		46		47	
48		49		50		51	
52		53		54		55	

56		57		58		59	
60		61		62		63	
64		65		66		67	
68		69		70		71	
72		73		74		75	
76		77		78			

VARIABLE REPLACEMENT

For any attribute of any element in the configuration file, you can insert the value of a previously captured OSD++ variable by inserting the name of the OSD++ variable surrounded by percent signs (%). The following snippet inserts the value of the OSD++ variable named *CompName* (previously retrieved) into the title of the dialog box.

```
<GetUser title="Isotope Software" subtitle="The current name of this
system is %CompName%. Please change it below if desired." icon="1">
```

VARIABLE PERSISTANCE

OSD++ has the ability to save OSD++ variables to a file for a future run of OSD++. This is perfect for two scenarios that I can think of. The first is pre-populating variables. In this scenario, you would create an administrative run of OSD++ to populate any desired OSD++ variables and have OSD++ save these to a file. Then, you can use this file for all future runs of OSD++ in a task sequence. The second scenario allows you to gather information, particularly from the user, before the start of the task sequence, and then create task sequence variables based on the information saved during the task sequence. As mentioned in the introduction, task sequences cannot display any interactive content while in Windows. Thus, to display a dialog box to the user, you must wait until the WinPE portion of the TS. This may not be desirable however. Creating two runs of OSD++ solves this issue: the first displays the dialog box(es) prompting the user for information and saving the information to a file. The second loads the file and creates applicable task sequence variables.

There are many other possibilities for this feature also.

COMPLETE EXAMPLE AND EXPLANATION

This example gets the current Windows version from the registry, gets the model name from WMI, and presents three dialog boxes to the user. It then creates two task sequence variables and adds a value to the registry.

```

<?xml version="1.0" encoding="utf-8"?>
<OSDPlus>
  1 <GetReg hive="HKLM" key="SOFTWARE\Microsoft\Windows
NT\CurrentVersion" value="CurrentVersion" varname="CurrentVersion" />
  2 <GetWmi namespace="root\cimv2" class="Win32_ComputerSystem"
property="model" varname="csmodel" />
  3 <GetUser title="Isotope Software" subtitle="Please answer the
following questions to deploy Windows 7 on your system" icon="64">
  <Textbox prompt="Primary User Name of this system" regex="."+
varname="UserName" icon="30" helptxt="Please enter the user name of the
primary user of this system."/>
  <Textbox prompt="Room Number" regex="\d{3}" varname="RoomNumber"
icon="33" helptxt="Please enter your three digit room number."/>
</GetUser>
  <GetUser title="Isotope Software" subtitle="Please answer the
following questions to deploy Windows 7 on your system" icon="64">
  <Listbox prompt="Please choose a business unit" icon="47"
varname="BusinessUnit">
  <Option alt1="HR%RoomNumber%\w\d{3}.*" alt2="Please enter HR
followed by the room number (%RoomNumber%), a letter, and then three
digits.">HR</Option>
  <Option alt1="IT%RoomNumber%\w\d{3}.*" alt2="Please enter IT
followed by the room number (%RoomNumber%), a letter, and then three
digits.">IT</Option>
  <Option alt1="FIN%RoomNumber%\w\d{3}.*" alt2="Please enter FIN
followed by the room number (%RoomNumber%), a letter, and then three
digits.">Finance</Option>
  <Option alt1="OPS%RoomNumber%\w\d{3}.*" alt2="Please enter OPS
followed by the room number (%RoomNumber%), a letter, and then three
digits.">Operations</Option>
  </Listbox>
</GetUser>
  5 <GetUser title="Isotope Software" subtitle="Please answer the
following questions to deploy Windows 7" icon="64">
  <Textbox prompt="Computer Name" regex="%BusinessUnitalt1%"
varname="CompName" icon="45" helptxt="%BusinessUnitalt2%"/>
</GetUser>
  6 <PutTSVar varname="CompName" tsvar="OSDComputerName"/>
  <PutTSVar varname="CurrentVersion" tsvar="WinVer"/>
  7 <PutReg hive="HKLM" key="SOFTWARE\Isotope" value="BusinessUnit"
valtype="number" createkey="yes" varname="BusinessUnit" />
</OSDPlus>

```

1. Retrieves the current Windows version from the registry and puts the value into the OSD++ variable named *CurrentVersion*.

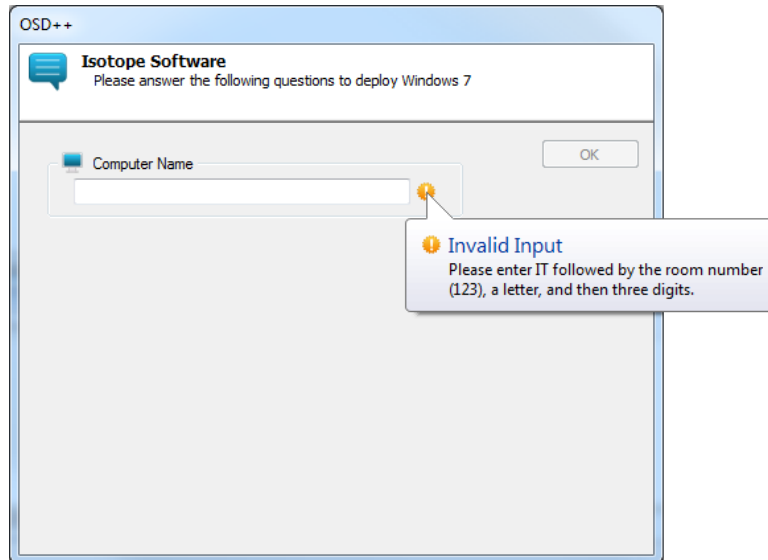
- Retrieves the model name of the system from WMI and puts it into the OSD++ variable named *csmodel*.
- Creates a dialog box with two prompts. The first is a textbox that requires at least one character and populates the OSD++ variable *UserName*. The second is a textbox that requires three digits and populates the OSD++ variable named *RoomNumber*.

The screenshot shows a dialog box titled "OSD++" with the "Isotope Software" logo and the instruction "Please answer the following questions to deploy Windows 7 on your system". There is an "OK" button in the top right. The dialog contains two input fields: "Primary User Name of this system" with a user icon and a warning icon, and "Room Number" with a house icon and a warning icon.

- Creates a dialog box with a single listbox. This listbox has four options. Three OSD++ variables are populated based upon the choice made: *BusinessUnit*, *BusinessUnitalt1*, and *BusinessUnitalt2*. All of the alternate variables are populated using a combination of static text and a replacement variable whose value is pulled from the room number value entered in the first dialog box and stored in the OSD++ variable *RoomNumber*.

The screenshot shows a dialog box titled "OSD++" with the "Isotope Software" logo and the instruction "Please answer the following questions to deploy Windows 7 on your system". There is an "OK" button in the top right. The dialog contains a dropdown listbox with the prompt "Please choose a business unit". The listbox is open, showing the following options: Finance, HR, IT, and Operations.

- Creates the third dialog box with a single textbox. The regular expression used is determined by the choice made in the previous dialog box and set using a substitution variable.



6. Creates two task sequence variables from two previously populated OSD++ variables.
7. Puts a value in the registry based on the previously populated OSD++ variable named *BusinessUnit*.

KNOWN ISSUES

1. The tab key does not properly navigate between items in the dialog box.

Please submit other issues encountered via the contact form on my blog at <http://myitforum.com/cs2/blogs/jsandys>.

CREDIT AND ACKNOWLEDGEMENTS:

David Sackstein for his great set of articles which I used to read the configuration file: <http://blogs.microsoft.co.il/blogs/davids/archive/2008/12/20/msxml-in-c-but-as-elegant-as-c.aspx>

C++/MFC helper classes:

- Pavel Antonov: <http://www.codeproject.com/KB/cpp/cmdlineparser.aspx>
- Paul Roberts: <http://www.codeproject.com/KB/miscctrl/CBloonMsg.aspx?display=Print>
- Jmgurgel: <http://www.codeproject.com/KB/dialog/dialogheader.aspx?display=Print>
- Keith A. Lewis: <http://www.codeproject.com/KB/system/CPPRegistryWrapper.aspx?display=PrintAll>
- Nschan: http://www.codeproject.com/KB/dialog/scrolling_support.aspx
- Hans Dietrich: <http://www.codeproject.com/KB/miscctrl/XGroupBox.aspx?display=Print>
- Boost: <http://www.boost.org>

Dialog and Application Icons: Based on Onebit icons (<http://icojoy.com/articles/44/>)

LICENSE

This application is released subject to the following license:

Microsoft Public License (Ms-PL)

This license governs use of the accompanying software. If you use the software, you accept this license. If you do not accept the license, do not use the software.

1. Definitions

The terms "reproduce," "reproduction," "derivative works," and "distribution" have the same meaning here as under U.S. copyright law.

A "contribution" is the original software, or any additions or changes to the software.

A "contributor" is any person that distributes its contribution under this license.

"Licensed patents" are a contributor's patent claims that read directly on its contribution.

2. Grant of Rights

(A) Copyright Grant- Subject to the terms of this license, including the license conditions and limitations in section 3, each contributor grants you a non-exclusive, worldwide, royalty-free copyright license to reproduce its contribution, prepare derivative works of its contribution, and distribute its contribution or any derivative works that you create.

(B) Patent Grant- Subject to the terms of this license, including the license conditions and limitations in section 3, each contributor grants you a non-exclusive, worldwide, royalty-free license under its licensed patents to make, have made, use, sell, offer for sale, import, and/or otherwise dispose of its contribution in the software or derivative works of the contribution in the software.

3. Conditions and Limitations

(A) No Trademark License- This license does not grant you rights to use any contributors' name, logo, or trademarks.

(B) If you bring a patent claim against any contributor over patents that you claim are infringed by the software, your patent license from such contributor to the software ends automatically.

(C) If you distribute any portion of the software, you must retain all copyright, patent, trademark, and attribution notices that are present in the software.

(D) If you distribute any portion of the software in source code form, you may do so only under this license by including a complete copy of this license with your distribution. If you distribute any portion of the software in compiled or object code form, you may only do so under a license that complies with this license.

(E) The software is licensed "as-is." You bear the risk of using it. The contributors give no express warranties, guarantees or conditions. You may have additional consumer rights under your local laws which this license cannot change. To the extent permitted under your local laws, the contributors exclude the implied warranties of merchantability, fitness for a particular purpose and non-infringement.