Using InduSoft Web Studio with Touch Screen PCs

Implementation Specifications or Requirements

Category	Item	
Software	IWS Version:	6.0 and later
	Service Pack:	N/A
	Windows Version:	WinXP/2000/NT and Windows CE
	Web Thin Client:	Yes
Equipment	Panel Manufacturer:	N/A
	Panel Model:	N/A
	Other Hardware:	N/A
	Comm. Driver:	All
	Controller (e.g.: PLC)	All
	Application Language:	N/A
Software Demo Application	N/A	

Summary

When using InduSoft Web Studio (IWS) with a touch screen PC, minor changes to the IWS application may be needed to make it "touch friendly", especially when there is no mouse or stylus that will be used. These changes include properly sizing objects, setting an object's command properties and changing communication settings (Driver and OPC) to give a touch screen PC the expected operational behavior.

These changes can be categorized as follows:

- Working with Simple Objects

- Working with Complex Objects

- Virtual Keyboards

- User Logon

- Ensuring Web Compatibility

- Command Properties for Screen objects

- Communication between the IWS application and devices (using Drivers and/or OPC Servers)
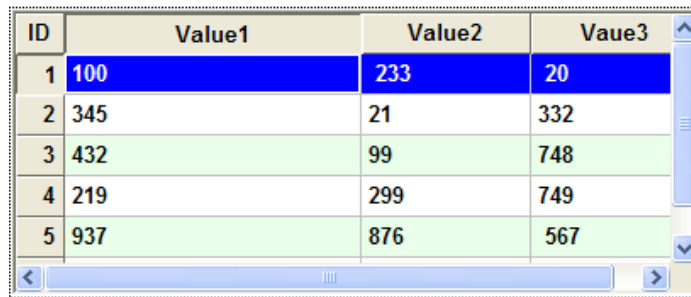
Working with Simple Objects

Simple objects include rectangles, pushbuttons, text I/O, bitmapped graphics objects, message boxes, etc. Human factors should be carefully considered when working with these simple objects to design a touch screen interface, especially for industrial applications. These considerations include:

- **Response time**
  A user typically expects to see a relatively instantaneous response to a selection (pressing) of an object on the display screen. For example, a button used to stop, start or jog a machine needs to be perceived as having a fast response, else the user will likely press the screen object again, perhaps producing undesired consequences. Response times of 0.25 seconds are usually acceptable, with response times up to 0.5 seconds being marginal. Response times longer that this are generally unacceptable from a user perspective. The object selected should change color, shape, or some other effect (e.g. 3-D) to indicate to the user that the object was selected. Device Drivers and OPC Servers need to be properly configured to insure timely communications with field devices.

- **Is the user wearing gloves?**
  Fingers come in many sizes. But designing a screen for a user that is wearing gloves can be more problematic, depending on the type of glove worn. The developer needs to understand the operational use environment to properly size of the object on the screen as well as set the spacing between objects so that an unintended object is not selected.

- **Does the user (machine operator) have impaired motor skills or sight?**
  Even if the touch screen PC is to be used in an environment where gloves are not used, it is important to remember that not all users have the same motor skills or eyesight. Objects also appear to shift when a user is looking at the screen from an angle instead of looking directly at the screen. This effect is often called parallax.

- **Impact of accidentally selecting an adjacent object instead of the intended one**
  Objects that indicate a machine or process status can typically be placed close together. However, objects (e.g. buttons) used to start or stop a machine or process, or to make critical adjustments, should be sufficiently spaced from other action-oriented objects such that there is not an accidental selection. But if there is an accidental selection, the developer should design the object such that an indication of the unintended selection is provided, and an object to deselect the unintended selection (by the same or a different button) should be immediately accessible. Objects that would have serious consequences if accidentally selected should be sufficiently isolated from other objects so that an accidental selection does not occur. In no case, should the touch screen PC be used for E-Stop purposes. This must be done with a physical pushbutton or switch. Machine or Process safety must be strongly considered when developing operator screens.

- **Choice of object color(s) to reduce effect of fingerprints and glare**
  Depending on the operating environment, the touch screen can be subjected to glare from sunlight or other lights, oil, grease, food products, etc. Suitable colors for the object should be used so that high contrast is always maintained.

Working with Complex Objects

It is important to make sure that objects placed on a display screen are sized appropriately to be pressed by a finger or stylus. Certain simple objects such as buttons, rectangles, circles, and symbols can be easily sized appropriately when placed on the display screen. However, more complex objects such as the Grid Object or Trend Control Object have scrollbars or combo boxes that do not scale with the size of the object.

Grid Object – with horizontal & vertical scrollbars
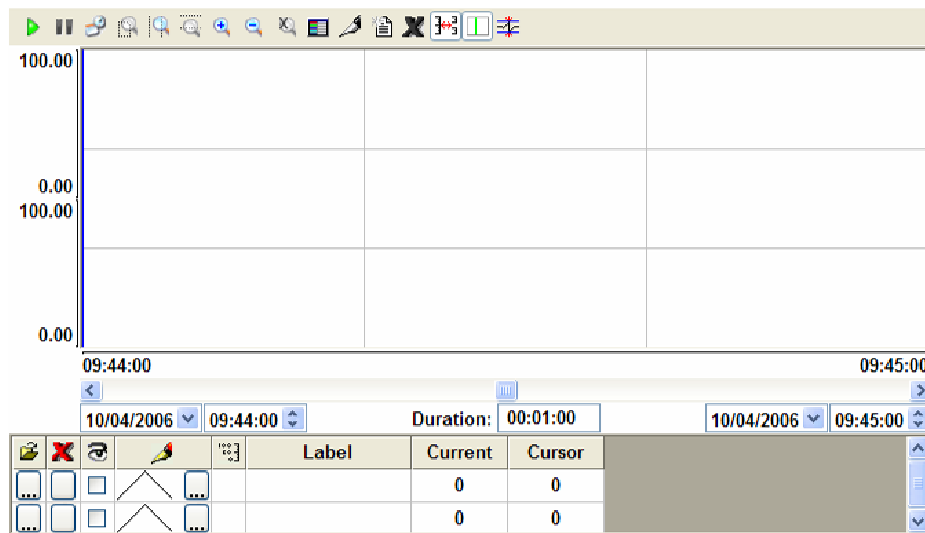
Trend Control Object – with horizontal & vertical scrollbars plus combo boxes
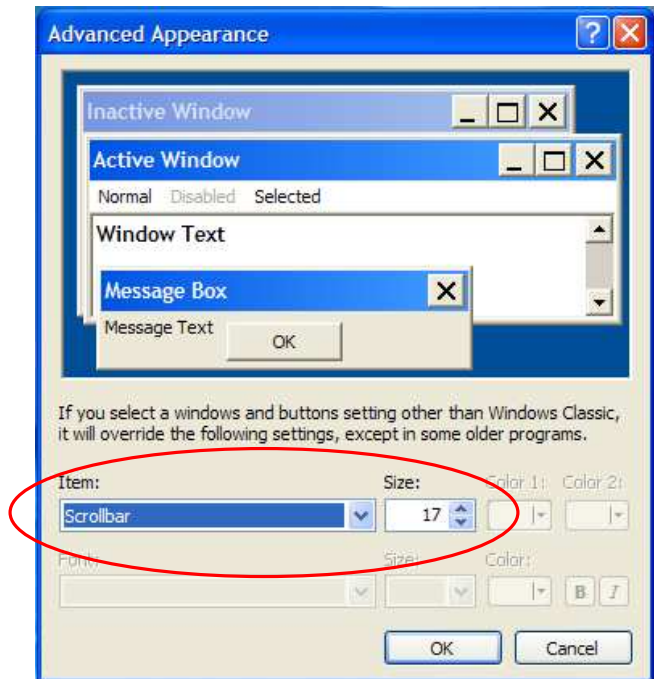
The size of the vertical and horizontal scrollbars is determined by settings specified in the Windows Operating System. These settings are located in the Display properties in the Windows System Control Panel. In a Windows XP operating system, for example, these settings are accessed by the following steps:

Start → Control Panel → Display → Appearance (tab) **→ Advanced** (button) **→ Item** (select scrollbar)

After changing the scrollbar size setting, click OK. This will close the Advanced Appearance dialog box. The Apply button in the Display Properties dialog box will no longer be shaded out. Click the Apply button. This will update all open applications to the new scrollbar size, and when a new application opens, it will use the new scrollbar size.
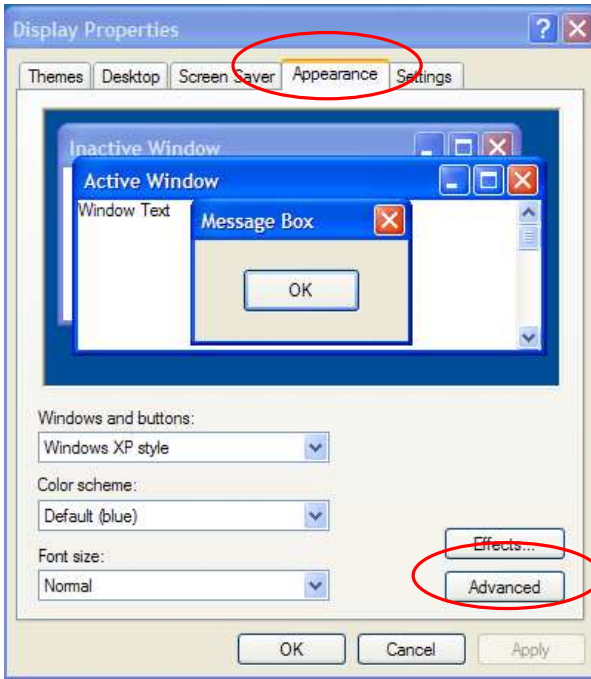
Display Properties – Appearance Tab   Advanced Appearance – Item: Scrollbar

As an example, a normal Windows XP setting for the scrollbar size might be approximately 17. Changing this scrollbar setting to 25-32 will make it larger, thus more "finger-friendly". The resultant size will be based on your screen resolution.
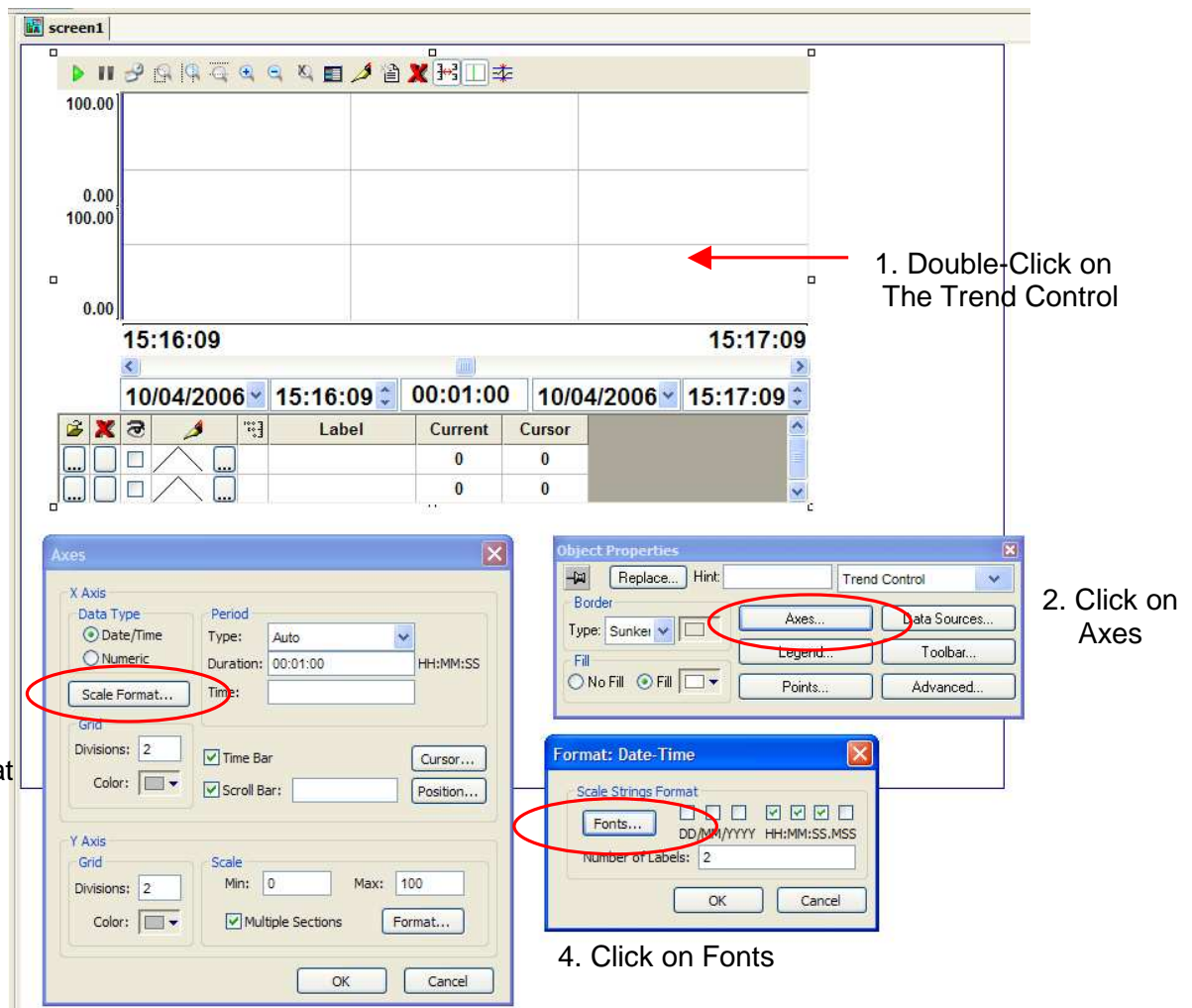

Grid Object with scrollbar setting set to size = 32

The above picture shows the Grid Object with the scrollbar setting set to a size of 32. Note however that the Grid object does not grow in size in the IWS application. Instead, the larger scrollbar will take away from the lower row(s) and right column(s) of the data fields.

Other objects that use scrollbars include the Alarm/Event Control and List Box Control. Note however, that Library symbols that include scrollbars do not necessarily scale their scrollbar size based on the Windows setting.

In addition to the scrollbar, the down arrow in a Combo Box used in the stand-alone Combo Box or with the date/time selection in the Trend Control object can be enlarged by changing the font size of the associated characters.

To change the size of the date/time characters in a Trend Control Object (thus enlarging the combo box down arrow), first double click on the Trend Control Object to access its Properties dialog box. Then click on **Axes**, then click on **Scale Format**, then click on **Fonts**. Set the size of the font to something larger (perhaps to 18 point font or larger). There is a practical limit to the font size due to the horizontal requirements for the date/time characters to be displayed vs. the available horizontal resolution for the Trend Control Object.


The vertical size of the Combo Box is determined by the font size of the characters, which in turn controls the size of the down arrow in the Combo Box. As with the Trend Control Object, if you increase the size of the font used in the Combo Box, be sure that you have sufficient horizontal width for the object on the

display screen to properly display the static labels or array tags used in the list contained in the Combo Box. Keep in mind that additional width may be required if runtime translation is used.

The steps to adjusting the font size of the list in the Combo Box are straight forward. First, double click on the object to access the Object Properties dialog box. Then click on **Fonts**. A change in the font size will increase or decrease the height of the Combo Box, and thus change the height of the down arrow. However, regardless of the width of the Combo Box, the width of the down arrow remains constant.

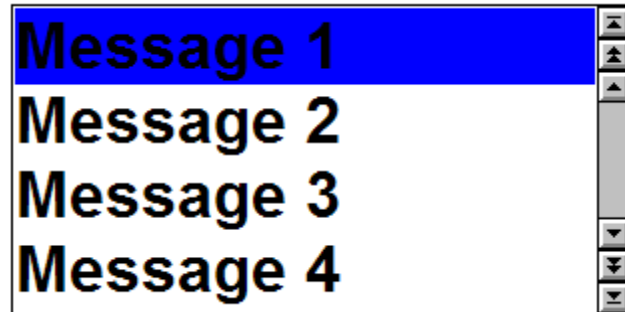Click on **Fonts**

Combo Box with smaller Font size
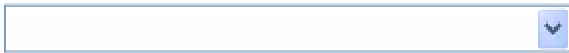
Combo Box with larger Font size
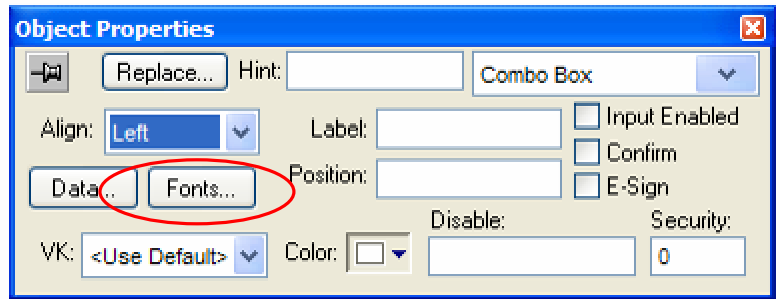
The font size using in a List Box object can be adjusted in a similar manner to the Combo Box. Double click on the List Box to access the Object Properties dialog box. Then click on **Fonts**. Note the example below with the identical List Box shown with font size = 12 and font size = 24. Note that the scroll bar width has not changed. As previously discussed, the scroll bar width can be adjusted in the Display Properties – Advanced Appearance dialog box, access from the Windows Control Panel.
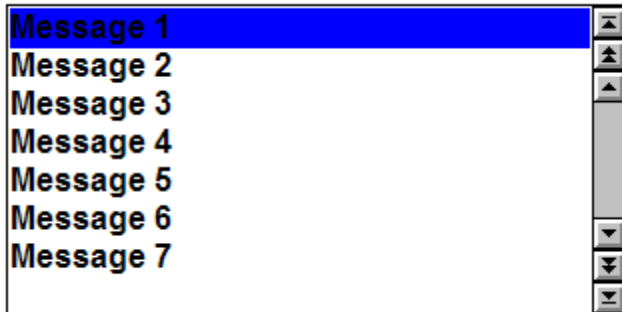
List Box Font Size = 12   List Box Font Size = 24

The IWS Check Box and Radio Button are two examples of objects that can utilize larger fonts but portions of the object (other than scrollbars) do not scale. Depressing the touch screen in the area where the word Check or Radio is located will select the Checkbox or Radio Button object, respectively.

□ **Check**   ← Font Size = 24 →   ○ **Radio**

□ Check   ← Font Size = 12 →   ○ Radio
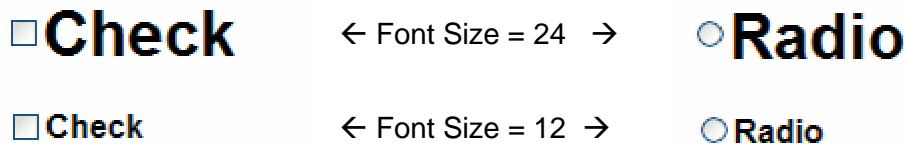
In addition to the above objects, the Alarm/Event Control and the Grid Object also allow the developer to adjust the font size used.

Touch Screen PCs
©Copyright InduSoft Systems LLC 2006
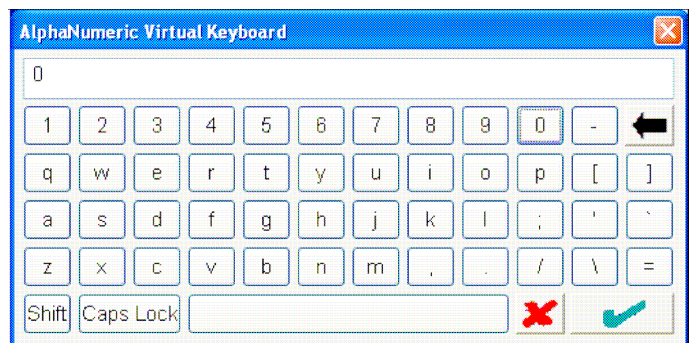
Virtual Keyboards

Since most touch screen computers do not have a built-in keyboard (although a keyboard can usually be added via the KBD connector or USB), InduSoft provides three different pop-up virtual keyboards. The Numeric Keypad is used for entry of numeric data. The Enhanced Keypad is used to enter alphanumeric data. The small size of the Enhanced Keypad makes it suitable from small displays such as with PDAs and HMIs with display sizes of 8" or less. The small arrow ($\rightarrow$) in the lower right corner of the Enhanced Keyboard is used to scroll through different panels of numeric and alphanumeric characters. The Alphanumeric Keypad is a QWERTY style alphanumeric keypad for entry of numbers and characters.
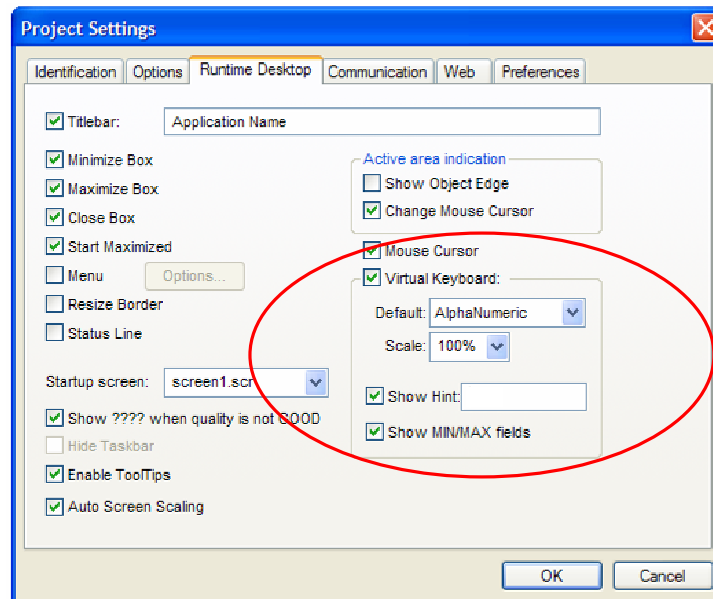

Numeric
Keypad
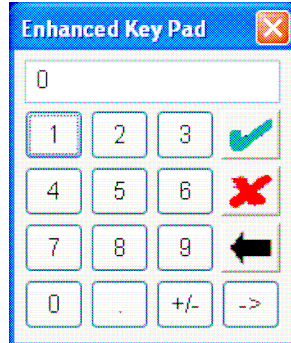
Enhanced Keypad

Alphanumeric
Keypad

Virtual keyboards are enabled for an IWS application by enabling the Virtual Keyboard option in the Runtime Desktop dialog box (from the menu toolbar choose Project $\rightarrow$ Settings $\rightarrow$ Runtime Desktop (tab))

In the Runtime Desktop dialog box, a default Virtual Keyboard style can be selected. However, the type of Virtual Keyboard used at runtime for a specific object can be specified by the object during development. Customization of the Virtual Keyboards can be accomplished by changing the key settings in the VK<Language>.INI file in the \BIN subfolder of the folder where the IWS program is located (usually C:\Program Files\InduSoft Web Studio v6.1). The language setting <Language> is a suffix to the "VK" file prefix. VKEN.INI (English) and VKGE.INI (German) files are included by default. These files can be copied, renamed and modified to support other languages. The keyboard language can be specified at runtime by calling the IWS built-in function **SetKeyboardLanguage**. For example, using a IWS Math Worksheet

Tag Name	Expression
TagRetVal	SetKeyboardLanguage("EN")
TagRetVal	SetKeyboardLanguage(TagLanguage)

In the first expression **SetKeyboardLanguage("EN")** refers to the file C:\Program Files\InduSoft Web Studio v6.1\VKEN.INI, which is the English language Virtual Keyboard file. The second expression, the tag **TagLanguage** is a string tag which contains the language suffix (e.g. "EN") of the VK<Language>.INI file. The IWS **TagRetVa**l is an optional tag used to test the returned value, which is 0 for a successful operation and 1 for an error.

The Virtual Keyboard is launched by either clicking on the object that requires a new value to be entered (e.g. Text I/O with the Input Enabled checkbox checked), or by invoking the IWS built-in function **Keypad().** The **Keypad** function can include the name of a tag to which the Keypad will write to. The full syntax of this IWS built-in function is:

> **Keypad(**"TagName", strOptKeyboardType, boolOptIsPassword**)**

>> **"TagName"** is the name of the tag which will be written (must be enclosed in double quote marks)

>> **strOptKeyboardType** is an optional parameter used to specify the name of the Virtual Keyboard to be launched. If omitted, the default Virtual Keyboard will be used. Valid types are:
>>> KeyPad
>>> EnhKeyPad
>>> Alphanumeric

>> **boolOptIsPassword** is an optional parameter used to specify whether the typed characters will be displayed as "*", useful when

The position of the Virtual Keyboard, when launched, is placed close to the object on the screen to which it is associated. However, the location of the Virtual Keyboard can be fixed by changing settings in the <app name>.APP file. The <app name> is the name of your application, and the application can be opened with Windows NotePad or another text editor. Add the following section and statements to this file:

> [Keypad]
> PosX = 0
> PosY = 0

In this case, the settings in the [Keypad] section apply to all Virtual Keypad types (Numeric Keypad, Enhanced Keypad, and Alphanumeric Keypad).

In addition to the type of Virtual Keypad, the Runtime Desktop dialog box provides some additional parameters that are useful when used with a touch screen PC. First is the scale. By default, the value is 100%, but this can be increased to make it more "touch friendly".

Starting with IWS 6.1 Service Pack 2, there are two additional settings available for Virtual Keypads in the Runtime Desktop dialog box. These new settings are "Show Hint" and "Show Min/Max Fields". The **Show Hint** setting, if checked, displays the text in the attached box when a Virtual Keyboard of the same type (Numeric, Enhanced or Alphanumeric) is opened. The Hint box in the Runtime Desktop is the default hint. However, IWS objects that display on a display screen and have input capabilities can have their own hint box (e.g. see the hint box in the Text I/O object below). If the object specifies a hint, this will override the default hint. The other new setting with IWS 6.1 Service Pack 2 is the **Show Min/Max Fields** setting. If this setting is checked, it will display the min and max value fields when the Numeric Keypad is launched. This feature only works for the Numeric Keypad, and is not available with the Enhanced Keypad or Alphanumeric Keypad. The values for the min and max fields are specified in the object for the tag being used.
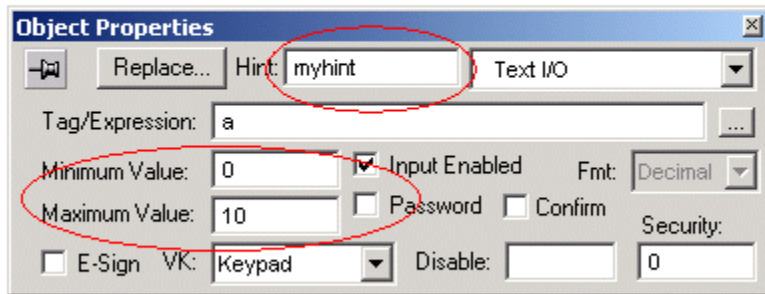
Text I/O Object Properties   Runtime Numeric Keyboard

In the above example above example, the Virtual Keyboard settings in the Runtime Desktop dialog box were set to:

Virtual Keyboard	Checked
Show Hint	Checked
Hint	Left Blank
Show MIN/MAX Fields	Checked

In the Object above (a Text I/O object), the Hint was set as follows:

Hint	myhint

Touch Screen PCs
©Copyright InduSoft Systems LLC 2006
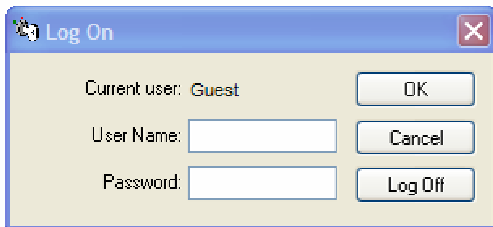
User Logoff and Login

User Logoff and Login can be accomplished at runtime by placing an object (e.g. a shape such as a rectangle, circle, etc.) on the display screen that has a command property associated with it. The command property would execute the function **LogOff()** or **LogOn()**. The function **LogOff()** has no arguments, but the **LogOn()** function can be used with or without arguments. If you know the username and password (or prompt the user for the username and password), they can be specified in the **LogOn()** command by using the following format:

> **LogOn(**strUserName, strPassword**)**
> > where strUserName is a string expression or string tag containing a valid IWS User Name,
> > > strPassword is a string expression or string tag containing a valid password

The **LogOff()** and **LogOn()** functions can be called from either an IWS Math Sheet or from VBScript. VBScript variables or expressions can be used as parameters for the **LogOn** function. Remember to specify the **LogOn** function as **$LogOn** if using it from VBScript.

If no arguments are provided to the **Logon()** function, the Log On dialog box will appear as shown as below. Although the LogOn dialog box does not have a setting from which to specify larger fonts, it will call a popup Virtual Keyboard whenever the User Name box or Password Box of the Logon Dialog Box is pressed. The style of the Virtual Keyboard is based on the default style as defined in the Runtime Desktop settings dialog (access via Project → Settings → Runtime Desktop (tab)). The size of the Virtual Keyboard can be set in the Runtime Desktop dialog.

LogOn Dialog Box

Virtual Keyboard for entering User Name & Password

Another approach to user logon and logoff is by putting dedicated buttons on a screen as shown below. One button with a caption of "Logoff" simply runs a VBScript code segment (Command Dynamic) calling the built-in IWS function **$Logoff()**. The "$" operator is put in front of the **Logoff()** function since the function is an IWS built-in function called from VBScript. The second button has a caption of "Logon" and has a VBScript Command Dynamic that opens the screen "Screen2.scr". A simple rectangle with the caption "UserName <cr> {username}" is added to the screen to display the current User Name at runtime. This portion of the caption {username} retrieves the internal IWS tag **username** which is the current name of the user that is logged on to the application. Since **username** is an internal IWS tag, it does not need to be declared.
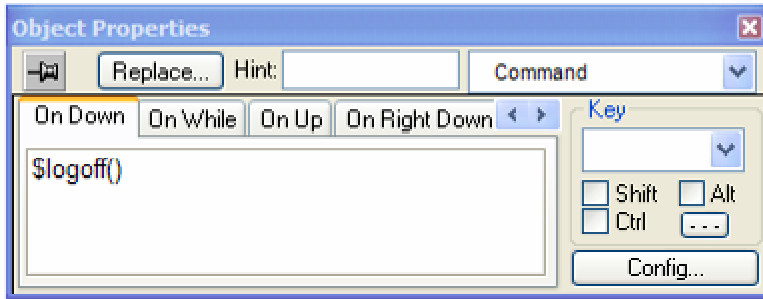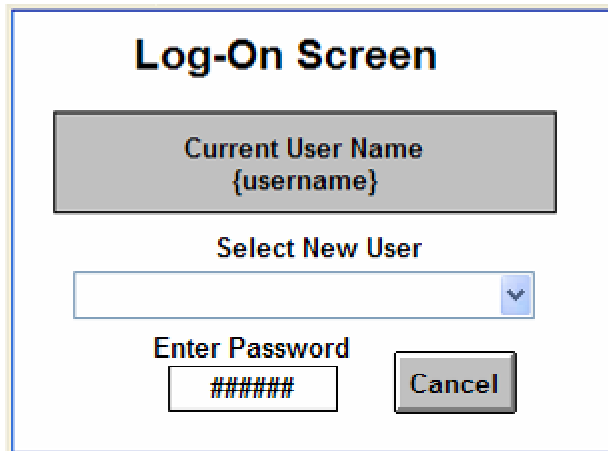
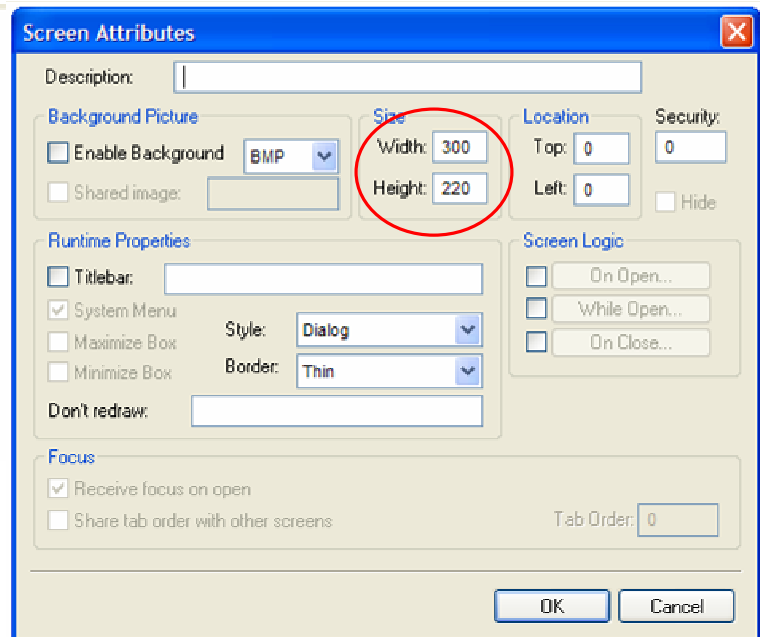Logoff Button – Command Dynamic (VBScript)
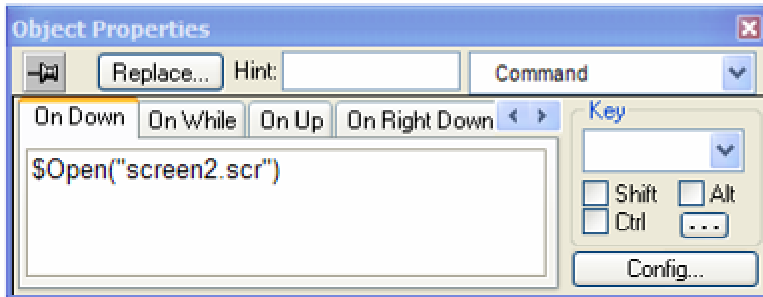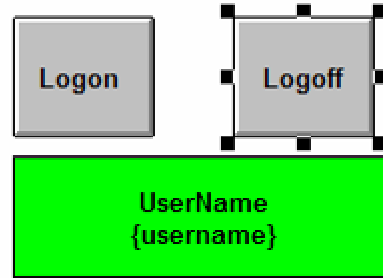
Logon Button – Command Dynamic (VBScript)
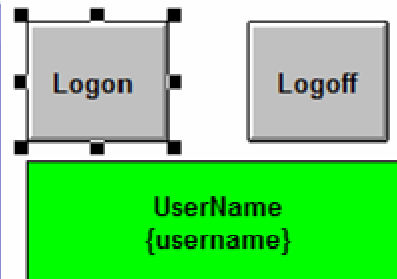
Screen2 (a file named "screen2.scr") is a small, simple popup screen used to interact with an operator for logging on to the application.


Screen2

Screen2 Attributes

Screen 2 has several objects. These objects are:
- Rectangle with Caption that displays current User Name (similar to rectangle on main screen)
- Combo Box that will be populated with existing User Names
- A Text I/O box for the Password. This Text I/O Box and the Text string "Enter User Name" become visible once a valid User Name is selected from the Combo Box.
- A button with the caption "Cancel" used to exit the logon process

As shown in the Screen2 Attributes settings, Screen2 is set to be a dialog window (will sit on top of other screens and remove itself when the screen is closed). It is located in the upper left corner, with a window width of 300 pixels and a height of 220 pixels.

The purpose of the Combo Box in Screen2 is to display a list of valid users at runtime. This way, the person logging on does not need to type in their user name. Set up of the Combo Box is straight forward.
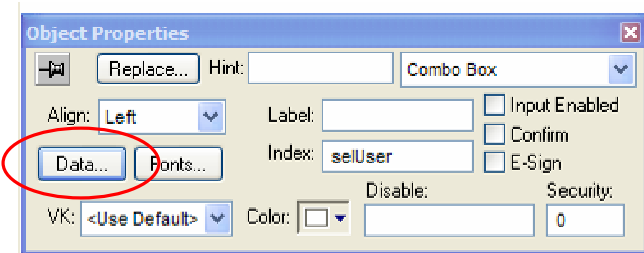
Combo Box Object Properties
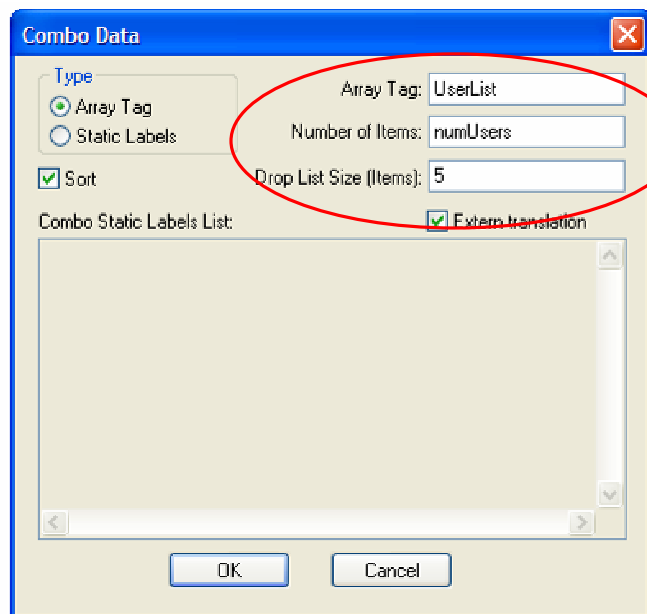
Combo Box Data Settings

As shown in the above Combo Box configuration settings, the index in the Combo Box (the item # that the user selects) is an IWS tag named selUser. The string tag array UserList contains the user names (the string array is populated from a VBScript routine (below), and the number of users is determined from the IWS built-in function **GetUserNames** (the number of users is the returned value).

The Text I/O object is used to enter the Password. The Object Property Dialog Box enables the object for input after the User Name is selected. An Alphanumeric Virtual Keyboard is used to input the Password.
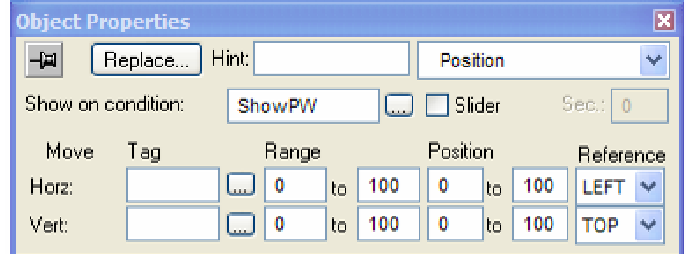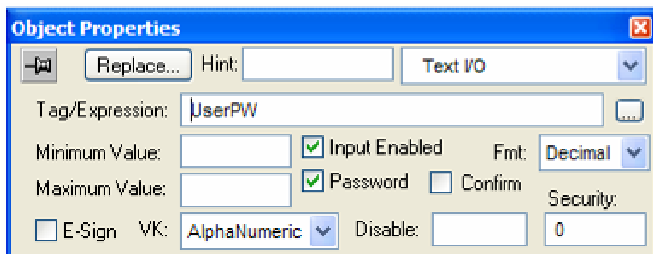
The Position property of the Text I/O object is used to enable visibility (show or hide) after a User Name is selected from the Combo Box.

The Cancel Button simply sets the tag **logOnCancel,** which is used in the VBScript routine below to sense that the user wants to cancel the logon operation and to take no further action. The Command Property for the Cancel Button is shown below.
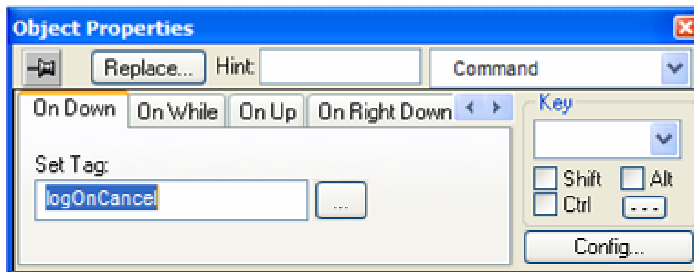
The VBScript code segment that is configured as a Screen Script for Screen2.scr is shown below:

```
'Variables available on this screen can be declared and initialized here.

'Procedures available on this screen can be implemented here

'This procedure is executed just once when this screen is open.
Sub Screen_OnOpen()
$showPW =0                              'Turn off visibility for the Password until ready
$selUser=-1                            'Set the index pointer to a null value (i.e not selected yet)
$numUsers = $GetUserNames("UserList",0) 'Set number of users for combo box, populate array of Users
$userPW = ""                           'Set password to null
$logOnCancel = 0                       'Turn off Cancel Button flag
End Sub

'This procedure is executed continuously while this screen is open.
Sub Screen_WhileOpen()
If $logOnCancel <> 1 Then                        'If Cancel not pressed, then execute logic
  If $selUser <> -1 Then                         'If user has selected a User from the Combo Box
    $showPW =1                                    'Show the Text I/O object to enter the Password
    If $userPW <> "" Or $userlist[$selUser] = "Guest" Then 'A password is entered or the username is Guest
      $Logon($userlist[$selUser], $userPW)       'Logon as new user (automatically logs off existing user)
      $Close("screen2.scr")                      'Close the logon screen
    End If
  End If
Else
  $Close("screen2.scr")                          'Close the logon screen since Cancel was pressed
End If
End Sub

'This procedure is executed just once when this screen is closed.
Sub Screen_OnClose()

End Sub
```

During runtime, when the LogOn button in the main screen is pressed, and a User Name is selected from the Combo Box, the following is a representation of how the application should appear:
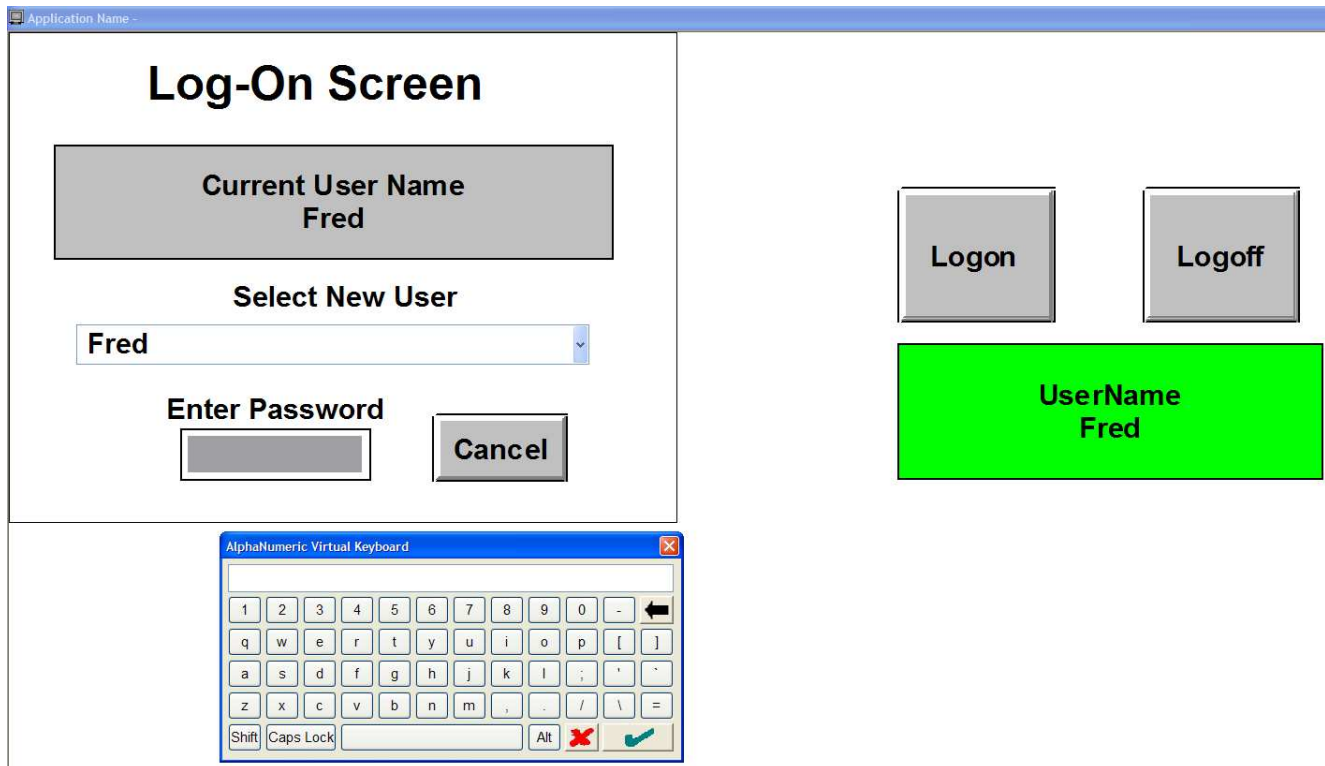
If the Logoff button in the main screen is pressed, then the User reverts to "Guest". If there is an error during the Logon, an error message will popup indicating there is an error in the password. This existing user will not be logged off.

Ensuring Web Compatibility

If an IWS application is to be used in a Web Thin Client environment, you may want to ensure that the IWS application does not call IWS built-in functions that are not supported in a Web Thin Client environment. To enforce compatibility, check the appropriate box in the Preferences tab of the Project Settings dialog box as shown below.
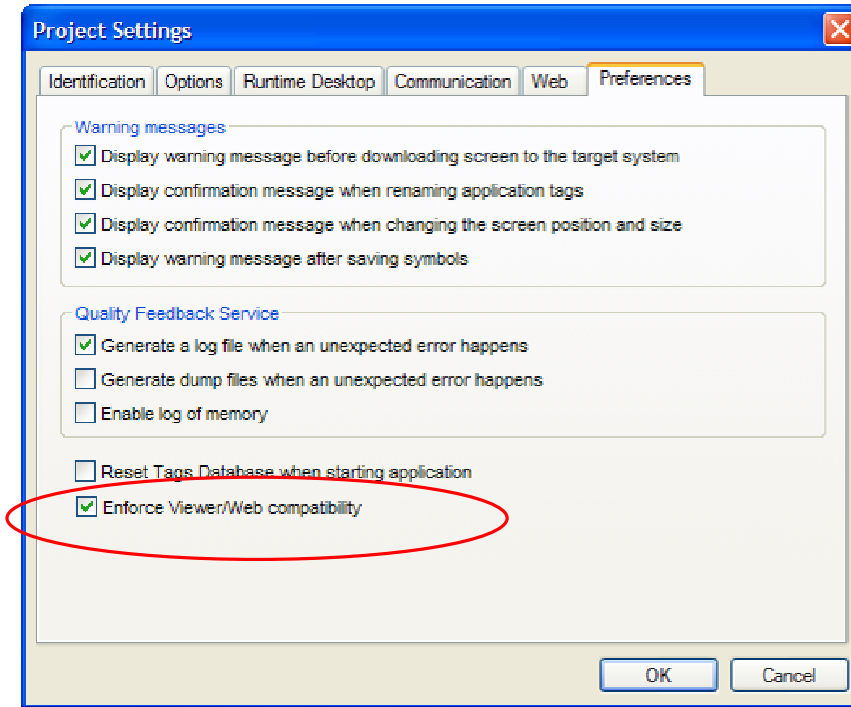
Project Settings - Runtime Desktop option to Enforce Viewer/Web compatibility

If the Enforce Viewer/Web Compatibility box is checked, then any built-in functions that are not supported for Web Thin Clients will not show up in the IWS object finder and will be removed from the list of available functions in VBScript's Intellisense.

The following is a list of IWS built-in functions that are not supported in a Web Thin Client environment. These functions are typically Server-only functions. Please check the IWS Users Manual for a complete listing and any changes.

IWS Built-in Function by Grouping	IWS Built-in Functions Alphanumerically Listed
Log Message Functions	AppSendKeys()
Trace()	CleanReadQueue()
Opening & Closing Windows Functions	Close()
Close()	CloseSplashWindow()
GetUserNames	DialError()
Module Activity Functions	DialGetClientIP()
AppSendKeys()	DialGetServerIP()
CleanReadQueue()	DialStatus()
CloseSplashWindow()	DialUp()

IWS Built-in Function by Grouping	IWS Built-in Functions Alphanumerically Listed
EndTask()	DialUpToCE()
ExitWindows()	DirCreate()
IsScreenOpen()	DirDelete()
IsTaskRunning()	DirLength()
IsViewerInFocus()	DirRename()
Math()	EndTask()
Recipe()	ExitWindows()
SendKeyObject()	FindAllDevices()
SetAppPath()	FindModem()
SetViewerInFocus()	For() & Next
ShutDown()	GetAppHorizontalResolution()
StartTask()	GetAppPath()
File Functions	GetAppVerticalResolution()
DirCreate()	GetFreeMemoryCE()
DirDelete()	GetHardKeyModel()
DirLength()	GetHardkeySN()
DirRename()	GetMemoryCE()
Hst2Text()	GetNetMACID()
Hst2TxtIsRunning()	GetUserNames
Print()	HangUp()
Graphic Function	Hst2Text()
PrintWindow()	Hst2TxtIsRunning()
System Information Functions	IsActiveXReg()
GetAppHorizontalResolution()	IsScreenOpen()
GetAppPath()	IsTaskRunning()
GetAppVerticalResolution()	IsViewerInFocus()
GetFreeMemoryCE()	Math()
GetHardKeyModel()	NoInputTime()
GetHardkeySN()	ODBCBeginTrans()
GetMemoryCE()	ODBCBindCol()
GetNetMACID()	ODBCCanAppend()
IsActiveXReg()	ODBCCanTransact()
NoInputTime()	ODBCCanUpdate()
SaveAlarmFile()	ODBCClose()
SaveAppAlarmPath()	ODBCCommitTrans()
SetWebConfig()	ODBCDelete()
Loop Functions	ODBCExecuteSQL()
For() & Next	ODBCInsert()
ODBC Functions	ODBCIsBOF()
ODBCBeginTrans()	ODBCIsDeleted()
ODBCBindCol()	ODBCIsEOF()
ODBCCanAppend()	ODBCIsFieldNull()
ODBCCanTransact()	ODBCIsFieldNullable()
ODBCCanUpdate()	ODBCMove()
ODBCClose()	ODBCMoveFirst()
ODBCCommitTrans()	ODBCMoveLast()

| IWS Built-in Function by Grouping	IWS Built-in Functions Alphanumerically Listed
ODBCDelete()	ODBCMoveNext()
ODBCExecuteSQL()	ODBCMovePrev()
ODBCInsert()	ODBCOpen()
ODBCIsBOF()	ODBCQuery()
ODBCIsDeleted()	ODBCRollback()
ODBCIsEOF()	ODBCSetFieldNull()
ODBCIsFieldNull()	ODBCSetFilter()
ODBCIsFieldNullable()	ODBCSetSort()
ODBCMove()	ODBCUnbindCol()
ODBCMoveFirst()	ODBCUpdate()
ODBCMoveLast()	PhoneDialUp()
ODBCMoveNext()	PhoneDisableListen()
ODBCMovePrev()	PhoneEnableListen()
ODBCOpen()	PhoneHangUp()
ODBCQuery()	PhoneStatus()
ODBCRollback()	Print()
ODBCSetFieldNull()	PrintWindow()
ODBCSetFilter()	Recipe()
ODBCSetSort()	SaveAlarmFile()
ODBCUnbindCol()	SaveAppAlarmPath()
ODBCUpdate()	SendKeyObject()
Dial-Up Functions	SetAppPath()
DialError()	SetViewerInFocus()
DialGetClientIP()	SetWebConfig()
DialGetServerIP()	ShutDown()
DialStatus()	StartTask()
DialUp()	SyncAlarm()
DialUpToCE()	SyncAlarmStatus()
FindAllDevices()	SyncEvent()
FindModem()	SyncEventStatus()
HangUp()	SyncTrend()
PhoneDialUp()	SyncTrendStatus()
PhoneDisableListen()	Trace()
PhoneEnableListen()	
PhoneHangUp()	
PhoneStatus()	
Database Functions	
SyncAlarm()	
SyncEvent()	
SyncTrend()	
SyncAlarmStatus()	
SyncEventStatus()	
SyncTrendStatus()	

Command Properties

Many IWS objects (e.g. button, rectangle, circle, symbols) can have command dynamics added to the object. Command dynamics are added by selecting the object and clicking on the **Command Property** button in the toolbox.
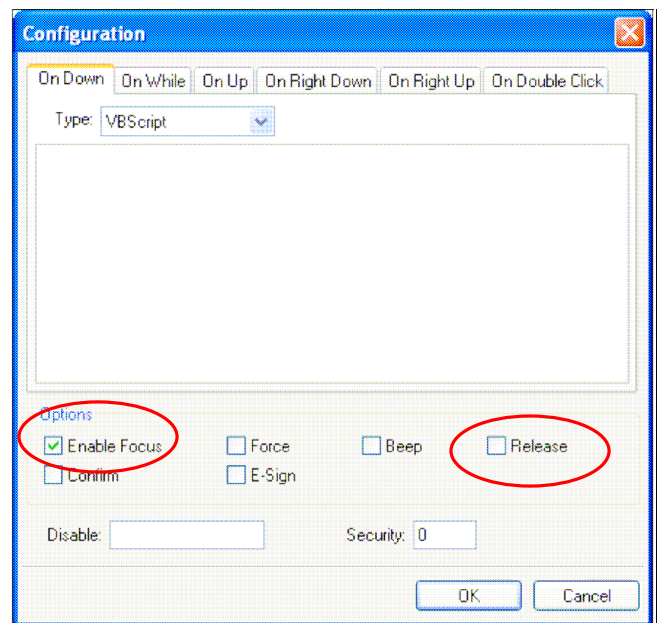

Command Property

The command property dialog box for an object is shown below. This dialog box allows developers to define various command dynamics for an object. Clicking on the **Config…** button launches the configuration dialog box, where the various command dynamics can be configured.
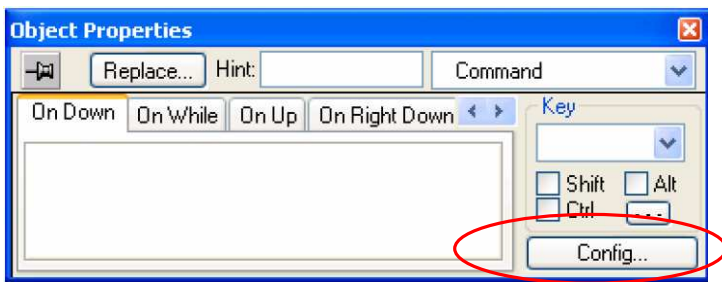
Command Property Dialog Box
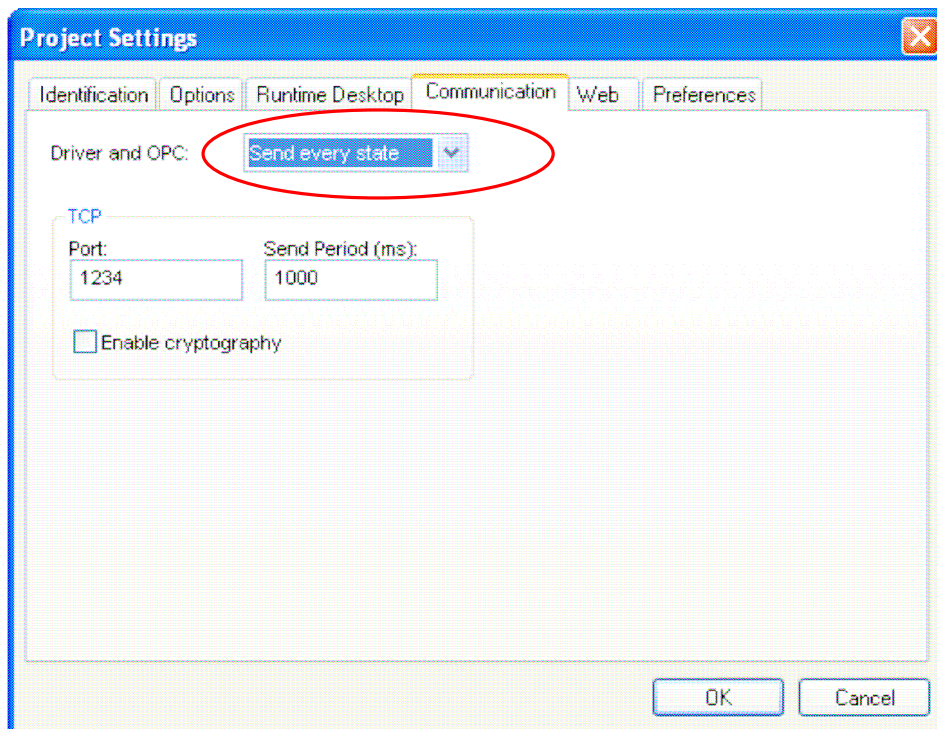

Configuration Dialog Box

For IWS applications with touch screen PCs, the configuration option **Enable Focus** should be unchecked and the configuration option **Release** should be checked. By having **Enable Focus** option unchecked prevents any navigation keys from accidentally selecting the object which has the command properties, and the **Release** option being checked allows the On Up event to be executed when your finger moves out of the object area, whether the button was released or not. This insures that the On Up event will always be executed after an On Down event, whether the developer uses the On Up event or nor.

Communications with Drivers and OPC Servers

When an IWS object such as a button is placed on a screen and the object is used to change a tag value associated with a PLC register (i.e. the tag value is to be written to the PLC register), it is important to set the Driver and OPC setting in the Project Settings → Communication tab to **Send every state**. This setting has two valid values: **Send every state** and **Send last state**. The **Send every state** option will insure that when the communication task (i.e. the Driver or the OPC Client) is configured to write tag values to the PLC (device) or OPC Server upon a change of tag value, the communication task will queue the tag values to be written to the PLC (device) or OPC Server. When the communication task is next executed, tag values from the queue will be retrieved in the order they were written and sent to the PLC (device) or OPC Server.

By comparison, when the **Send last state** option is selected and the Communication Task is configured to write tag values to the PLC (driver) or OPC Server upon a change of tag value, only the last tag value will be sent to the PLC (device) or OPC Server. While this setting is commonly used, it is advisable to use the **Send every state** option to allow queuing the communications so that if the Communications Task is not operating when the IWS object is selected (e.g. communications is busy with another device), the proper sequence of communications can occur once the Communications Task is available for execution.

By selecting **Send every state**, when an object (such as a button) is used for a machine jog or adjustment function, it will properly based on the number of times the object is pressed.

Project Settings Dialog Box