



# Web Browser Programming Guide

for Version 5.1

Naurtech Terminal Emulation and Web  
Browser Smart Clients

for Windows CE Devices

CETerm | CE3270 | CE5250 | CEVT220

## Copyright Notice

This document may not be reproduced in full, in part or in any form, without prior written permission of Naurtech Corporation.

Naurtech Corporation makes no warranties with respect to the contents of this document and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, Naurtech Corporation, reserves the right to revise this publication and to make changes to it from time to time without any obligation to notify any person or organization of such revision or changes.

## Trademarks

CETerm<sup>®</sup>, CE3270<sup>™</sup>, CE5250<sup>™</sup>, CEVT220<sup>™</sup> are trademarks of Naurtech Corporation.

Other product names mentioned in this manual may be trademarks or registered trademarks of their respective companies and are hereby acknowledged.

## Software Version

**This document is for version 5.1 of Naurtech Web Browser smart clients.**

## Table of Contents

Copyright Notice .....	2
Trademarks.....	2
Software Version .....	2
Table of Contents .....	3
Preface .....	5
Assumptions .....	5
Conventions used in this Manual.....	5
Additional Documentation.....	6
Online Knowledgebase.....	6
1.0 Introduction.....	7
1.1 Feature Highlights.....	7
2.0 Common Tasks .....	10
2.1 Scanner Input.....	10
2.2 Key Actions .....	11
2.3 Text Input Elements .....	12
2.4 IDA Action Codes.....	12
2.5 Device Control From JavaScript .....	13
2.6 Device Properties and CETerm Configuration.....	15
3.0 Special HTML META Tags .....	17
3.1 Application.....	18
3.2 Battery.....	19
3.3 BatteryNavigate .....	20
3.4 Command.....	21
3.5 CursorPos .....	22
3.6 ErrorNavigate.....	22
3.7 GetUnitInformation.....	23
3.8 HomeKey .....	24
3.9 IDA .....	24
3.10 MoveSIP.....	25
3.11 OnAllKeys .....	26
3.12 OnKey .....	27
3.13 PowerOn .....	29
3.14 Reboot.....	30
3.15 Scanner.....	30
3.16 ScannerNavigate .....	31
3.17 SetDate .....	33
3.18 SetTime.....	33
3.19 Signal .....	34
3.20 SignalNavigate.....	35
3.21 SIP .....	36
3.22 SIPUp.....	37
3.23 TextSize .....	37
3.24 TimerInterval .....	38
3.25 TimerNavigate.....	38
3.26 ZebraLabel_Complete or PLSeriesLabel_Complete .....	39
3.27 ZebraLabel_Print or PLSeriesLabel_Print.....	40
4.0 Advanced Topics .....	42

4.1 Navigating to Pre-configured URLs .....	42
4.2 Controlling the Scanner .....	42
4.3 Input Focus and the Tab Key.....	44
4.4 Session Launcher .....	48
4.5 How to Identify the Current Browser.....	49
4.6 Device Information .....	50
4.7 Symbol Web Client .....	50
5.0 Printing from HTML .....	53
5.1 Printing with a META Tag .....	53
5.2 PrintString and Print Methods.....	53
5.3 NAURTECH:PRINT Tag.....	53
5.4 ActiveX Printing Controls .....	54
6.0 CEBrowseX Control.....	54
Syntax .....	54
ClassID.....	55
Methods .....	55
PostIDA( IDACode, session ).....	55
SendIDA( IDACode, session ).....	55
SendText( text, session ).....	56
Value = GetProperty( propertyName ) .....	56
Status = SetProperty( propertyName, propertyValue ) .....	56
PlaySound( sound ).....	56
PlayTone( volume, frequency, duration ) .....	57
Status = PrintString( printData ) or Status = Print( printData) .....	57
Properties.....	57
EventHandlers.....	57
7.0 TextX Control.....	58
Syntax .....	58
ClassID.....	59
Methods .....	59
objectname.SetFocus( select ).....	60
objectname.ShowSIP( visible ).....	60
Properties.....	60
EventHandlers.....	61
void OnChange() .....	61
void OnClick( int x, int y ).....	61
void OnFocus().....	61
void OnLostFocus().....	61
void OnKeyDown( int vkey ).....	61
int OnKeyPress( int akey ).....	62
void OnKeyUp( int vkey ).....	62
Appendix 1 - Properties .....	63
Appendix 2 - IDA Action Codes .....	65
Appendix 3 - Virtual Key Codes.....	74
Glossary.....	78
Index .....	79

## Preface

All of us at Naurtech Corporation constantly strive to deliver the highest quality products and services to our customers. We are always looking for ways to improve our solutions. If you have comments or suggestions, please direct these to:

### **Naurtech Corporation**

e-mail: [contact@naurtech.com](mailto:contact@naurtech.com)

Phone: +1 (425) 837.0800

## Assumptions

This manual assumes you have working knowledge of:


- Microsoft Windows user interface metaphor and terminology.
- Stylus based touch screen navigation terminology.
- Dynamic HTML, the browser DOM, and JavaScript.
- Basic operations and requirements of the host applications you want to access with the Naurtech smart client.

## Conventions used in this Manual

This manual uses the following typographical conventions:

- All user actions and interactions with the application are in bold, as in **[Session] [Configure]**
- Any precautionary notes or tips are presented as follows

**Tip:** Text associated with a specific tip

-  represents new version specific information
- All text associated with samples is presented as follows. We use lower case in most samples for readability.

```
<html>
<head>
<title>Naurtech Web Browser</title>
<meta http-equiv="Scanner" content="Enabled">
```

```
<meta http-equiv="ScannerNavigate"
      Content="Javascript:onscan('%s','%s','%s','%s','%s');">
</head>
...
</html>
```

## Additional Documentation

The Naurtech Web Browser is an integral feature of Naurtech terminal emulation Smart Clients. Please refer to the User's Manual for these Smart Clients for detailed installation and configuration information. The User's Manual may be downloaded from the "Support" section of our website. You may also want to refer to the CETerm Scripting Guide for additional features to enhance the web browser.

## Online Knowledgebase

Although we continually strive to keep this manual up to date, you may find our online support knowledgebase useful for the latest issues, troubleshooting tips and updates. You can access the support knowledgebase from our website at:

[www.naurtech.com](http://www.naurtech.com) → Support → Knowledgebase

## 1.0 Introduction

The Naurtech Web Browser provides a robust and flexible environment for Web based applications which are accessed from a mobile device. This browser is available for Windows CE .NET, Windows CE 5.0, Windows Mobile 2003 and Windows Mobile 5.0 based devices.

Device tailored versions of the browser are available for most industrial terminals. These versions integrate with the peripherals on each device, such as barcode scanner, magnetic stripe reader, RFID reader and Bluetooth printer. The Naurtech Web Browser provides control of the peripherals and simplifies actions such as data collection, validation, and printing.

All Naurtech Web Browsers are integrated with one or more Terminal Emulations (TE) which allows a natural migration path from legacy text based TE applications to newer Web based applications. The Web based applications can be presented in a familiar single-purpose (locked down) configuration which uses keys, the touch screen, or both for user interactions.

The Naurtech Web Browser offers control of the device peripherals and settings via integrated JavaScript extensions, ActiveX controls, and special HTML META tags. This guide is written primarily to describe these extensions and custom features. Please consult the standard references for details on JavaScript, HTML syntax, the browser Document Object Model (DOM), and other aspects of Dynamic HTML. Please refer to the User's Manual for details on basic usage and configuration of the Naurtech Web Browser clients.

There is no current standard for the browser extensions and META tags that have been added to Web browsers for industrial handhelds. Within the Naurtech Browser, we strive to support all the ad-hoc and de-facto extensions available in other products. In many cases, the Naurtech Web Browser is a "drop-in" replacement for these other products. In addition, we support nearly uniform behavior across a wide spectrum of devices from every major hardware device manufacturers.

Beyond these basic extensions, we have added many unique features to enable you to build more powerful business applications.

### 1.1 FEATURE HIGHLIGHTS

Following are some of the special features and extensions in the Naurtech Web Browser.

- **Access Control / Device Lockdown.** Access controls allow administrators to hide the start bar and to prevent users from exiting the Naurtech client. You

can hide the Windows CE “Start” button, the whole Start bar and/or the application menu bars and toolbars. Users are prevented from navigating to un-authorized Web sites as is possible with Pocket IE.

- **Multiple Browser Sessions.** All Naurtech clients allow up to 4 simultaneous sessions. Each session can be connected to a different Web application. This allows quick access to separate applications. Each session supports a unique scanner configuration.
- **Enhanced native HTML text INPUT.** Under Pocket PC 2002 and Windows Mobile 2003, the standard HTML text INPUT element will popup the SIP when it receives focus, and will not respond to the Tab key to advance the focus. The Naurtech Web Browser can prevent the SIP popup and enables Tab based navigation, without resorting to ActiveX input objects. Native HTML text INPUT elements are easier to use and help maintain consistency across different platforms.
- **Scanner Control via JavaScript.** Scanner input can be intercepted by JavaScript methods for data validation and editing. The scanner can also be enabled or disabled with META tags or from JavaScript.
- **Integration with CTerm Scripting and Automation Objects.** The web browser has access to the independent JavaScript engine running within CTerm and the associated Automation Objects. This independent engine can enhance legacy web pages and add functionality. The Automation Objects provide access to more features in CTerm and the underlying Windows OS. Please refer to the CTerm Scripting Guide for full details.
- **Key Driven Interaction.** Hardware keys can be re-assigned with HTML META tags to activate any desired action. Keys can be used to navigate to specific pages, to clear fields, submit forms, and even switch to different sessions.
- **On Screen Indicators.** Battery and WLAN (RF) strength can be displayed with on-screen indicators. The meters may also be displayed within KeyBar buttons. On some devices, keypad state indicators can also be displayed on-screen.
- **Printing from HTML.** Several techniques are available to send print content from an HTML page to a printer. The printer may be accessed via a serial port, IrDA, Bluetooth or the WLAN.



- **Notification Features.** Sound (.wav) files, tone generators (beepers), and vibrators can all be activated on devices which have these capabilities
- **Context Menus.** Custom context menus can be defined to provide access to special actions without tying up valuable screen real-estate.

## 2.0 Common Tasks

This section describes some common ways that enhanced features can be used within a Web based application. Here we show how to manage scanner input, invoke actions via keys, and interact with the handheld device. Only small code “snippets” are shown. For complete details see the reference sections of this manual. These tasks help to illustrate the power of the Naurtech Web Browser for building Web based applications.

### **2.1 SCANNER INPUT**

The barcode scanner, and other readers such as magnetic card readers and RFID readers are typically integrated with the Naurtech client. The configuration of the reader is maintained by the Naurtech client and is managed to allow an independent configuration for each host session. When data is available from a reader, it is directed to the current host session. The data is typically inserted at the current cursor (or focus) location. For a Web page with multiple text input elements, this can be problematic. The focus may be in the wrong text element, or if the focus is not in a text element, the input may be lost.

A better way to receive scanner input is to use an extension, which directs the data to a JavaScript method or submits it to a URL. To enable this action, you must define a special HTML META tag and a JavaScript method to process the data. Here is a typical META tag:

```
<meta http-equiv="ScannerNavigate"
content="Javascript:onscan('%s','%s','%s','%s','%s');">
```

When the scanner reads a barcode, each of the ‘%s’ items will be filled with information from the scan and the onscan method will be invoked. Here is a typical method which shows what each argument contains

```
<script language=javascript>
function onscan(data, source, type, time, length)
{
    alert("The barcode scanned was " + data +
        "\nThe symbology was " + type +
        "\nScanned at " + time +
        "\nWith a length of " + length);
}
</script>
```

This method simply presents a popup message with the scanner data. More typically, the data would be validated and inserted into a text element. The HTML form containing that element may also be submitted:

```
<script language=javascript>
```

```
function onscan(data, source, type, time, length)
{
  if (length > 5)
  {
    document.form[0].barcode.value = data;
    document.form[0].submit();
  }
}
<\script>
```

## **2.2 KEY ACTIONS**

In many situations, a Web application will be written to make special use of the keys on a handheld. The device may be used without a stylus or there may be function keys (Fx) which the application uses to perform special actions. The Naurtech Web Browser has several special features to make use of the hardware keys. These features are especially important on Windows Mobile based systems. The Windows Mobile browser does not have native support for special key actions, but the Naurtech Web Browser overcomes this limitation.

The simplest way to assign a key to a special action is to use a special HTML META tag. This will instruct the browser to execute a JavaScript method or navigate to a pre-specified URL when the key is pressed:

```
<meta http-equiv="OnKey0x70" content="Javascript:onF1key();" ><! F1>
...
<script language=javascript>
function onF1key()
{
  // Clear entry
  document.form[0].barcode.value = "";
}
<\script>
```

or

```
<meta http-equiv="OnKey0x70" content="home.htm" ><! F1>
```

It is important to know that the OnKey META tag acts as a “hotkey” and it will activate the action even if the focus is in a text input element. If you assign an action to a normal key such as ‘1’ then you will be unable to enter the ‘1’ as a character anywhere on the page. Techniques are available to ignore the hotkey action within text input elements.

## **2.3 TEXT INPUT ELEMENTS**

All Web applications will use text input elements at some point to collect information such as a barcode or count. The standard text input element is the HTML INPUT such as

```
<input type="text" name="count" maxlength="10" size="10">
```

The capabilities of the text input element are different for different Windows CE platforms. The Windows Mobile platform has the most limited text input. For example, this input element does not support special event handlers such as “OnKeyPress”, does not respond to the Tab key to advance the focus, and will popup the Soft Input Panel (SIP) whenever focus is received by the element.

The Naurtech Web Browser corrects these deficiencies by adding Tab key support, and by locking down the SIP when desired. We can also provide an ActiveX control (TextX) which can be used in place of the native input element if it is essential to trap all keys with an “OnKeyPress” handler. However, with the unique behavior of the OnKey META tag within native text elements, along with the Tab key and SIP control, the TextX input control is not usually needed. Avoiding the TextX input control simplifies the HTML and accelerates application development. We describe the use of the TextX control in this manual for those situations where it is required.

Under the Windows CE .NET and CE 5.0 platform, the native text INPUT element fully supports the OnKeyPress handler as well as other events, and the TextX control is never needed.

## **2.4 IDA ACTION CODES**

An IDA Action Code is a special value that is used to invoke a device action, program action, or emulator action within the Naurtech Smart Client. IDA Action Codes can invoke special keys under terminal emulation, sound a tone, connect a session, or show the SIP. There are many IDA codes and these are documented Appendix 2 in this manual. Almost any action which can be invoked by a KeyBar or assigned to a hardware key, can be invoked by an IDA code. Under the Web Browser, IDA codes can be sent to the program in several different ways. They can be in a special META tag, in an HTML link, or sent via JavaScript.

Here is a sample which pops up the SIP when a page loads:

```
<meta http-equiv="IDA" content="IDA_SIP_SHOW">
```

Or, you can toggle the SIP visibility from a link:

```
<a href="ida:IDA_SIP_TOGGLEHIDE">Toggle Soft Input Panel (SIP)</a>
```

Or, you can perform the action from JavaScript by setting the document location:

```
<script language=javascript>
function togglehide()
{
  // Toggle the SIP visibility
  // This format may not work for Pocket PC
  location.href = "ida:IDA_SIP_TOGGLEHIDE";

  // Or (remove the comment characters)
  // document.location = "ida:IDA_SIP_TOGGLEHIDE";

  // Or (remove the comment characters)
  // window.navigate("ida:IDA_SIP_TOGGLEHIDE" );
}
</script>
```

The next section describes additional methods for invoking IDA Action Codes from within a JavaScript method.

## **2.5 DEVICE CONTROL FROM JAVASCRIPT**

The Naurtech Web Browser is tailored to the features of most handheld devices. If the handheld has a vibrator or tone generator, we provide access to those features through JavaScript. We also provide access to most operations of the Browser client, such as switching to other sessions, or retrieving device or configuration information.

To access these features you can use an ActiveX control or for the CE .NET or Windows CE 5.0 platforms, you can use the Document Object Model (DOM) “external” object. There are slight differences between these two approaches, however they basically work the same.

**ALERT:** The CEBrowseX control has been updated in CETerm V5.1 to provide access to the new “CETerm” and “OS” automation objects used in the CETerm script engine. The new “CETerm” object provides all the features that were previously available from CEBrowseX. Please note the revised syntax. Existing web pages will continue to work but should be converted to the new syntax as they are updated.

The following example uses the ActiveX CEBrowseX control to sound a tone.

```
<html>
<head>
<meta http-equiv="OnKey0x31" content="javascript:mybeep();"><! 1>
</head>
<body>
<object id="CEBrowseX" classid="clsid:D14943BD-4900-453E-8582-
725F21A57E0C" height=0, width=0></object>
<a href="Javascript:mybeep();">Tap me or press 1 for beep</a><br>
...
<script language=javascript>
function mybeep()
{
    document.CEBrowseX.CETerm.PostIDA( "IDA_BEEP_LOUD", 0 );
}
</script>
</body>
</html>
```

Within this manual, we often assign the name “CEBrowseX” to the CEBrowseX control when it is created. Usually this is followed by reference to the “CETerm” object which has the methods we wish to use.

The PostIDA method sends an IDA Action Code as described in the previous section. For example, action codes can activate the vibrator (for 500 millisecond) (IDA\_VIBRATE\_500), switch to a different CETerm session (IDA\_SESSION\_S1) and many more actions.

The sample shown above can be used on both the Windows Mobile and Windows CE .NET and 5.0 platforms. Under Windows CE, the example could also be written as

```
<html>
<head>
<meta http-equiv="OnKey0x31" content="javascript:mybeep();"><! 1>
</head>
<body>
<a href="Javascript:mybeep();">Tap me or press 1 for beep</a><br>
...
<script language=javascript>
function mybeep()
{
    external.CEBrowseX.CETerm.PostIDA( "IDA_BEEP_LOUD", 0 );
}
</script>
</body>
</html>
```

Note that we do not require the CEBrowseX object under Windows CE because we have direct access via the builtin "external" object.

## **2.6 DEVICE PROPERTIES AND CETERM CONFIGURATION**

The CEBrowseX object or the "external" object can be used to access device properties and to read or set portions of the CETerm configuration. The following sample shows how this can be used

```
...
<body onload="javascript:fetchvalues();">
<object id="CEBrowseX" classid="clsid:D14943BD-4900-453E-8582-
725F21A57E0C" height=0, width=0></object>
<form id="form1" name="form1">
Property Features<br>
<input type="text" id="serialnumber" name="serialnumber"
size="20"><br>
<input type="text" id="ipaddress" name="ipaddress" size="20"><br>
</form>
...
<script language=javascript>
function fetchvalues()
{
    document.form1.serialnumber.value =
        document.CEBrowseX.CETerm.GetProperty( "device.serialnumber" );
    document.form1.ipaddress.value =
        document.CEBrowseX.CETerm.GetProperty( "device.ipaddress" );
}
</script>
...
```

There are several different properties currently accessible and more will be added in future versions. These are documented in Appendix 1. One other useful property gives access to the "User Text" area of CETerm. User Text are strings of characters which may contain IDA codes and which can be "sent" to an emulator. User Text strings are often tied to hardware keys to simplify text entry or to create "mini-macros" of IDA actions. From the browser, the User Text can be used as a general device-local persistent storage. They can also be used to send the user to special pre-configured URL's.

Here is an example of using a User Text area for persistent storage:

```
...
<body onload="javascript:loadfields();">
<object id="CEBrowseX" classid="clsid:D14943BD-4900-453E-8582-
725F21A57E0C" height=0, width=0></object>
<form id="form1" name="form1">
Login Page<br>
<input type="text" id="user" name="user" size="20"><br>
<input type="password" id="password" name="password" size="20"><br>
```

```
</form>
...
<script language=javascript>
function loadfields()
{
    var username = document.CEBrowseX.CETerm.GetProperty(
                                                "app.usertext.1" );

    if (username)
    {
        // Load with cached user name for this device
        document.form1.user.value = username;
    }
}
...
</script>
```

When the login is successful the application would store the current user name for the next login attempt

```
...
function onlogin( username )
{
    // Login was successful
    document.CEBrowseX.CETerm.SetProperty( "app.usertext.1",
                                          username );
}
...
```

There are some limitations with User Text. The values are shared among all emulator sessions and currently there are only 64 slots available.



### 3.0 Special HTML META Tags

This section describes the special META tags (or elements) that are recognized by the Naurtech Web Browser and are used to convey special instructions to the browser. The META tags use the standard HTML format but are not recognized by standard browsers. These special META tags assign hotkeys, control the scanner, configure the device, and other tasks.

A META tag has the form

```
<meta http-equiv="identifier" content="valueString">
```

The Naurtech Browser supports a unique set of “identifier” values. However, because there is no current standard for these META tags, we also recognize most of the ad-hoc and de-facto extensions available in other products. In particular, we support most of the Symbol Pocket Browser and Intermec iBrowse META tag identifiers. For the iBrowse META tags, we recognize the identifiers with or without the leading “iBrowse\_” text.

The following table summarizes the supported identifier values and their functions. Following the table are individual sections with detailed descriptions of the identifiers. All identifiers are effective only on the page that contains them unless otherwise specified in the description.

Identifier	Description
Application	Exit the program
Battery	Display on-screen battery information
BatteryNavigate	JavaScript or URL invoked with battery information
Command	Exit the program
CursorPos	Set location of wait cursor
ErrorNavigate	JavaScript or URL invoked on error
GetUnitInformation	JavaScript or URL invoked with device information
HomeKey	Enable/disable home key (F5)
IDA	Invoke IDA Action Code
MoveSIP	Set location of SIP input panel
OnAllKeys	Bind all keys to JavaScript or URL action
OnKey	Bind key to JavaScript or URL action
PowerOn	JavaScript or URL invoked on resume from suspend
Reboot	Invoke a device reset
Scanner	Enable/disable the scanner
ScannerNavigate	JavaScript or URL invoked on successful scan
SetDate	Set the system date

Identifier	Description
SetTime	Set the system time
Signal	Display on-screen WLAN (RF) signal information
SignalNavigate	JavaScript or URL invoked with WLAN information
SIP	Control the SIP
SIPUp	Show the SIP
TextSize	Set text size factor (zoom)
TimerInterval	Set interval for TimerNavigate
TimerNavigate	JavaScript or URL invoked on TimerInterval
ZebraLabel_Print	Contents of data for printing
ZebraLabel_Complete	JavaScript or URL invoked to report print status

Several special tags contain a JavaScript statement or URL in the content. Depending on the tag, these may contain the text “%s” which is replaced by data which is unique to the tag action. You must have the correct number of replacement placeholders depending on the tag. If the number is incorrect, the tag will not be recognized.

When specifying a URL, it may be any standard form. The JavaScript: form is simply a type of URL. You may also use files local on the device:

```
"file:///Application Data/myapp/errorpage.htm?errno=%s&msg=%s"
```

You may also use the proprietary “ida:” type to invoke various actions.

### **3.1 APPLICATION**

The Application tag performs actions which affect the client application. Currently this only exits the web browser.

#### **Syntax**

```
http-equiv="Application"  
content="Quit"
```

#### **Comments**

This tag is supported for compatibility with other browsers. It is preferable to use the IDA Action Code “IDA\_PROGRAM\_EXIT” via the IDA META tag, an “ida:” URL or a PostIDA call.

#### **Example**

```
<html>
```

```
<head>
<meta http-equiv="Application" content="Quit">
</head>
<body>
This message should not be visible.
</body>
</html>
```

## **3.2 BATTERY**

The Battery tag is used to configure the on-screen battery strength meter. This meter overlays the HTML content and is updated at a specified interval. You can change the location and style of the meter. Available styles include a horizontal or vertical single bar meter or filled battery icon. The meter may be repositioned by a “touch and drag” stylus action if dragging is not disabled.

### **Syntax**

```
http-equiv="Battery"
content="Show"
      "Hide"
      "Right"
      "Left"
      "Top"
      "Bottom"
      "x=n"
      "y=m"
      "HBattery"
      "VBattery"
      "AllowDrag"
      "NoDrag"
```

Where x and y are the screen coordinates of the upper left corner of the meter icon. The screen coordinate (0, 0) is in the upper left corner of the screen with x increasing to the right and y increasing downward. Right, Left, Top, and Bottom change the orientation of the meter and the placement of the icon. HBattery enables a horizontal battery icon filled according to strength. VBattery enables a vertical battery icon filled according to strength. AllowDrag will allow the user to drag the meter, whereas NoDrag will prevent dragging.

### **Comments**

This tag is used to control the battery meter display. The meter may also be controlled within the CETerm configuration, independent of the Battery tag. The Battery tag will always override the internal configuration and will persist until changed by another Battery tag. Within the CETerm configuration you can

specify the update interval and a notification when the strength falls below a designated level.

### Example

```
<html>
<head>
<meta http-equiv="Battery" content="Show">
<meta http-equiv="Battery" content="HBattery">
<meta http-equiv="Battery" content="x=200">
<meta http-equiv="Battery" content="y=20">
</head>
<body>
The horizontal battery icon should be visible.
...
</body>
</html>
```

## **3.3 BATTERYNAVIGATE**

The BatteryNavigate tag causes the specified JavaScript or URL to be invoked with battery information on a regular interval. The interval can be specified within a CETerm configuration dialog.

### Syntax

```
http-equiv="BatteryNavigate"
content="javascript:OnBattery(%s, %s, %s, %s);"
<!-- or -->
content="/battery.htm?AC=%s&strength=%s&backup=%s&chemistry=%s"
```

Where the "%s" are replaced with (1) AC line status, (2) main battery strength as a percentage, (3) backup battery strength as a percentage, and (3) the chemistry of the battery.

### Comments

This tag will work with or without a visible battery meter. The strength normally ranges from 0 to 100. The special strength value of -1 indicates that the strength cannot be determined. If you use a URL for the action, in most cases, the URL will navigate away from the current page rather than repeatedly calling a JavaScript method.

### Example

```
<html>
```

```
<head>
<meta http-equiv="BatteryNavigate"
  content="javascript:OnBattery(%s, %s, %s, %s);">
</head>
<body>
<script language=javascript>
function OnBattery(ACstate, strength, backup, chemistry)
{
  if (strength == -1)
  {
    alert("Unable to determine battery strength.");
  }
  else
  {
    alert("Battery strength = " + strength);
  }
}
</script>
...
</body>
</html>
```

### **3.4 COMMAND**

The Command tag performs actions which affect the client application. Currently this only exits the web browser.

#### **Syntax**

```
http-equiv="Command"
content="exit"
```

#### **Comments**

This tag is supported for compatibility with other browsers. It is preferable to use the IDA Action Code "IDA\_PROGRAM\_EXIT" via the IDA META tag, an "ida:" URL or a PostIDA call.

#### **Example**

```
<html>
<head>
<meta http-equiv="Command" content="exit">
</head>
<body>
This message should not be visible.
</body>
</html>
```

### **3.5 CURSORPOS**

The CursorPos tag is used to reposition the “busy-cursor” when it is visible. The busy cursor may appear while waiting for pages to load. Use CursorPos to move the busy-cursor from the default location at the center of the screen

#### **Syntax**

```
http-equiv="CursorPos"
content="x=n"
      "y=m"
```

Where x and y are the screen coordinates with (0, 0) in the upper left corner of the screen and x increasing to the right and y increasing downward.

#### **Comments**

The new position only applies to the page loading busy-cursor; other busy-cursors will be unaffected. The new position will apply until changed with another CursorPos tag.

#### **Example**

```
<html>
<head>
<meta http-equiv="CursorPos" content="x=0">
<meta http-equiv=" CursorPos" content="y=0">
</head>
<body>
Busy cursor in upper left corner.
...
</body>
</html>
```

### **3.6 ERRORNAVIGATE**

The ErrorNavigate tag directs error messages to a JavaScript method or to a URL.

#### **Syntax**

```
http-equiv="ErrorNavigate"
content="javascript:MyErrorHandler(%s, %s);"
<!-- or ->
content="http://10.1.1.8/errorpage.htm?errno=%s&msg=%s"
<!-- or to hide errors ->
content="javascript:var HideErrors=' %s%s' ;"
```

The first “%s” is replaced by an error number and the second “%s” by an error message.

## Comments

This tag should be the first special META tag defined on the page, but not before any JavaScript methods that may be invoked by the content. If this tag is not specified, errors are reported via popup messages.

## Example

```
<html>
<head>
<script language=javascript>
function MyErrorHandler(errno, msg)
{
    alert( "Error Number=" + errno + "\nMessage=" + msg );
}
</script>
<meta http-equiv="ErrorNavigate"
    content="javascript:MyErrorHandler(%s, %s);">
</head>
<body>
...
</body>
</html>
```

## 3.7 GETUNITINFORMATION

The GetUnitInformation tag reports device and client information to the host or user.

## Syntax

```
http-equiv="GetUnitInformation"
content="javascript:ReportInfo(%s, %s, %s);"
<!-- or -->
content="http://10.1.1.8/uinfo.htm?serial=%s&uuid=%s&version=%s"
```

The first “%s” is replaced by the device serial number, the second %s by the Windows CE device UUID, and third “%s” by the Web Browser version.

## Comments

This tag is supported for compatibility with other browsers. It is preferable to use the “CETerm.GetProperty” method of the CEBrowseX control or the “external” object (under CE .NET or 5.0 only).

## Example

```
<html>
<head>
<meta http-equiv="GetUnitInformation"
    content="javascript:ReportInfo(%s, %s, %s);">
```

```
</head>
<body>
This page reports unit information.
...
<script language=javascript>
function ReportInfo(serial, uuid, version)
{
    alert( "Serial Number=" + serial +
          "\nUUID=" + uuid +
          "\nVersion=" + version );
}
</script>
</body>
</html>
```

### **3.8 HOMEKEY**

The HomeKey tag enables the “home key” (F5) to navigate to the current home URL.

#### **Syntax**

```
http-equiv="HomeKey"
content="Enabled"
       "Disabled"
```

#### **Comments**

This tag is supported for compatibility with other browsers. It is preferable to use the IDA Action Code “IDA\_URL\_HOME” via the IDA META tag, an “ida:” URL or a PostIDA call.

This action can also be achieved with the “OnKey” tag. This tag remains in effect until explicitly changed or the session ends.

#### **Example**

```
<html>
<head>
<meta http-equiv="HomeKey" content="Enabled">
</head>
<body>
Press F5 to navigate to the home URL.
</body>
</html>
```

### **3.9 IDA**

The IDA tag performs a wide range of actions to control the device and the client.



**Syntax**

```
http-equiv="IDA"  
content="IDA_symbolicname"
```

**Comments**

This tag offers rich functionality. It is used primarily when an action is needed upon loading a page. Alternatively, an “onload” handler can be used in the BODY element to perform actions via JavaScript.

A list of IDA Action Codes and their description can be found in the Appendix. All IDA symbolic names must be in upper case.

Any number of IDA meta tags may be specified on a single page. They are acted upon sequentially when parsed.

**Example**

```
<html>  
<head>  
<meta http-equiv="IDA" content="IDA_SIP_SHOW">  
</head>  
<body>  
The Soft Input Panel (SIP) should be visible.  
</body>  
</html>
```

**3.10 MOVE\_SIP**

The MoveSIP tag is used to reposition the Soft Input Panel (SIP). MoveSIP is not recommended for hiding the SIP off-screen. Use the SIP lockdown features within CETerm to prevent the SIP from popping up when not wanted.

**Syntax**

```
http-equiv="MoveSIP"  
content="x=n"  
"y=m"
```

Where x and y are the screen coordinates with (0,0) in the upper left corner of the screen and x increasing to the right and y increasing downward.

**Comments**

Use MoveSIP to move the default SIP location. **WARNING:** MoveSIP may move the SIP to a non-visible location. Usually, entering the CETerm configuration dialogs will temporarily restore the SIP to the default location. Also, window and scroll behavior may be erratic on Windows Mobile devices with the SIP in a non-standard location.

### Example

```
<html>
<head>
<meta http-equiv="MoveSIP" content="x=0">
<meta http-equiv="MoveSIP" content="y=240">
</head>
<body>
SIP is shifted down a bit.
...
</body>
</html>
```

## **3.11 ONALLKEYS**

The OnAllKeys tag assigns a single JavaScript action or URL to all hardware keys on the handheld device. The action will take place regardless of the focus location on the page.

### Syntax

```
http-equiv="OnAllKeys"

content="javascript:myKeyAction(%s);"
<!-- or -->
content="http://10.1.1.8/inventory.htm?key=%s"
```

The “%s” is replaced by the Windows CE “Virtual Key Code” (VK) value of the key pressed.

### Comments

See the Appendix 3 for a list of Virtual Key Codes, their symbolic names, hexadecimal representation, and the typical keyboard name. If an OnKey tag has been specified for an individual key, that tag’s action will be invoked in place of the OnAllKeys action. See the OnKey tag for additional information.

### Example

```
<html>
```

```
<head>
<meta http-equiv="OnAllKeys"
  content="javascript:myKeyAction(%s);"><! All keys >
</head>
<body>
Main Menu<br>
1. Cycle Count<br>
2. Inventory<br>
3. Receiving<br>
Select an action:<br>
Press 'A' to check version.<br>
...
<script language=javascript>
function myKeyAction(vkcode)
{
  alert( "Key pressed =" + vkcode);
}
</script>
</body>
</html>
```

### **3.12 ONKEY**

The OnKey tag assigns a JavaScript action or URL to hardware keys on the handheld device. The action will take place regardless of the focus location on the page.

#### **Syntax**

```
http-equiv="OnKey0xZZ"
           "OnKeyDDD"
           "OnKeyIgnoreInText"
           "OnKeyVK_name" (version 4.5.3 or later)
           "OnKey_name"   (version 4.5.3 or later)

content="javascript:myKeyAction();"
<!-- or -->
content="http://10.1.1.8/inventory.htm"
```

Where ZZ is a two digit hexadecimal number that represents the Windows CE “Virtual Key Code” (VK) for a physical key, or “DDD” is up to 3 decimal digits representing the VK code, or “name” is the portion of the Virtual Key symbolic name after the underscore.

#### **Comments**

See the Appendix for a list of Virtual Key Codes, their symbolic names, hexadecimal representation, and the typical keyboard name. This tag allows several different formats to specify the VK value. Using the symbolic name

format yields HTML that is easier to read and maintain. See `OnAllKeys` to direct all key input to a single action.

Although most VK codes are uniform across devices, some devices can remap the keyboard at a driver level to change the VK codes. Consult your hardware documentation to understand what VK codes are generated by the keys on your device.

Please note that some keys may be tied to operating system actions and they may not be sent to the running applications, thus they cannot be used. Other times, they are tied to an action and will still be sent to the application, so you may see side-effects of their use.

All key shift states such as CTRL, ALT, and Shift are ignored by the `OnKey` action. However, it is possible to use shift states via key remapping as described below. We recommended that you do **not** assign `OnKey` actions to the CTRL, ALT, or SHIFT keys themselves.

The Naurtech Web Browser has very flexible key remapping features. In most cases, any user action which simulates a key, such as a `KeyBar` button or scanner post-amble will invoke the `OnKey` action. For example, you may remap Shift+"1" within the Web Browser key remapping to perform the F1 key action. When Shift+"1" is pressed in a browser session, the `OnKey` action for F1 will be invoked. Tapping on an "F1" `KeyBar` button will also invoke the `OnKey` action.

Unlike the Naurtech Web Browser, some other browsers do not act on `OnKey` assignments when the focus is in a native text INPUT element or in an ActiveX text component. Under Pocket PC and Windows Mobile 2003, the "`OnKeyIgnoreInText`" tag can be used to ignore the `OnKey` assignments when the focus is in a native text input element or in a TextX input object. The content is ignored for "`OnKeyIgnoreInText`"

There is no limit to the number of `OnKey` assignments within a page.

## Example

```
<html>
<head>
<meta http-equiv="OnKey0x1B"
  content="javascript:document.form[0].clearbutton.click();"><! ESC >
<meta http-equiv="OnKey0x31" content="/cyclecount.htm"><! 1>
<meta http-equiv="OnKey0x32" content="/inventory.htm"><! 2 >
<meta http-equiv="OnKey0x33" content="/receiving.htm"><! 3 >
<meta http-equiv="OnKey0x41" content="ida:IDA_PROGRAM_ABOUT"><! A >
</head>
```

```
<body>
Main Menu<br>
4. Cycle Count<br>
5. Inventory<br>
6. Receiving<br>
Select an action:<br>
Press 'A' to check version.<br>
...
<form name="form1">
<input type="text" name="scan" value="" size=30><br>
<input type="button" name="clearbutton" value="Clear Scanned Data"
  onclick="javascript:document.form1.scan.value='';" >
</form>
</body>
</html>
```

### **3.13 POWERON**

The PowerOn tag specifies an action that will occur when the handheld device resumes operation after a power suspend.

#### **Syntax**

```
http-equiv="PowerOn"
content="javascript:PowerOnAction();"
<!-- or -->
content="http://10.1.1.8/login.htm?mode=resume"
```

#### **Comments**

This tag is useful to set the browser to a consistent URL or state after a suspend/resume cycle. For example, a user authentication can be required to maintain security.

#### **Example**

```
<html>
<head>
<meta http-equiv="PowerOn"
content="http://10.1.1.8/login.htm?mode=resume">
</head>
<body>
...
</body>
</html>
```

### **3.14 REBOOT**

The Reboot tag will invoke a warm (soft) or cold (hard) reset of the device.

**WARNING: Do not perform the cold reset unless you are prepared to lose all current settings, data, and add-on programs on the device.**

#### **Syntax**

```
http-equiv="Reboot"  
content="Warm"  
        "Cold"
```

#### **Comments**

This tag is supported for compatibility with other browsers. It is preferable to use the IDA Action Code "IDA\_WARMBOOT" via the IDA META tag, an "ida:" URL or a PostIDA call.

A warm reset will cause all un-saved work-in-progress to be lost. A cold boot will reset all RAM to factory original configurations. A cold boot will typically clear all network and device settings, user data files, and add-on programs that have not been placed in non-volatile memory.

The reset will occur as soon as the tag is parsed.

#### **Example**

```
<html>  
<head>  
<meta http-equiv="Reboot" content="Warm">  
</head>  
<body>  
This message should not be visible.  
</body>  
</html>
```

### **3.15 SCANNER**

The Scanner tag is used to enable or disable the barcode scanner.

#### **Syntax**

```
http-equiv="Scanner"  
content="Enabled"  
        "Disabled"
```

#### **Comments**

This tag is used to enable or disable the scanner when a page is first loaded. You can also use the IDA codes IDA\_SCAN\_RESUME and IDA\_SCAN\_SUSPEND to change the state from an IDA META tag, an "ida:" URL or dynamically via a PostIDA call.

We do not support the "AutoEnter" or "AutoTab" content values. These values are un-needed because these and more complex scanner post-ambles can be configured within the Naurtech Web Browser. See the User's Manual for more information about the scanner configuration.

Under Windows Mobile 2003, the Naurtech Web Browser will advance focus within native HTML text input objects when the Tab entered via a key or post-amble, without any extra handlers.

All other scanner configurations are maintained within the Web Browser configuration dialogs. These settings are session dependent and may be different for different Web browser or TE sessions.

### Example

```
<html>
<head>
<meta http-equiv="Scanner" content="Enabled">
</head>
<body>
The scanner will work on this page.
<form name="form1">
<input type="text" name="scan" value="" size=30><br>
...
</form>
</body>
</html>
```

## **3.16 SCANNERNAVIGATE**

The ScannerNavigate tag directs scanner input to the specified JavaScript method or URL. This tag should be used whenever possible to provide the most robust control of scanner input.

### Syntax

```
http-equiv="ScannerNavigate"
content="javascript:OnScan('%s', '%s', '%s', '%s', '%s');"
content="javascript:OnScan('%s', '%s', '%s');"
<!-- or -->
content="/scan.htm?data=%s&src=%s&type=%s&time=%s&length=%s"
content="/scan.htm?data=%s&type=%s&time=%s"
```

There are two variants of the ScannerNavigate command for compatibility with other browsers. The first uses 5 parameters and the second uses 3 parameters. For the 5 parameter version, the “%s” are replaced with (1) barcode data, (2) source scanner name, (3) symbology type, (4) timestamp, and (5) barcode length. For the 3 parameter version, the “%s” are replaced with (1) barcode data, (2) symbology type, and (3) timestamp.

The variant type is determined automatically. If there are other that 3 or 5 substitution parameters, the ScannerNavigate tag is ignored.

### Comments

Only one ScannerNavigate tag is permitted on a page. The ScannerNavigate tag should be used to ensure that the scanner data is inserted into the correct input element, or is submitted directly via the URL. The barcode data can be examined, validated and/or edited prior to use.

### Example

```
<html>
<head>
<meta http-equiv="ScannerNavigate"
  content="javascript:OnScan('%s', '%s', '%s', '%s', '%s');">
</head>
<body>
<object id="CEBrowseX"
  classid="clsid:D14943BD-4900-453E-8582-725F21A57E0C"
  height=0, width=0></object>
<form name=form1>
Fill With First Scan<br>
<input type=text name="scan1" value="" size=30><br>
Fill With Second Scan<br>
<input type=text name="scan2" value="" size=30><br>

<script language=javascript>
function onscan(data, source, type, time, length)
{
  if (document.form1.scan1.value == "")
  {
    document.form1.scan1.value = data;
  }
  else if (document.form1.scan2.value == "")
  {
    document.form1.scan2.value = data;
  }
  else
  {
    document.CEBrowseX.CETerm.PostIda("IDA_VIBRATE_500", 0);
  }
}
```



```
        alert("Form full, scan discarded.");
    }
}
</script>
</form>
</body>
</html>
```

### **3.17 SETDATE**

The SetDate tag is used to set the current system date on the handheld.

#### **Syntax**

```
http-equiv="SetDate"
content="mm/dd/yyyy"
        "mm, dd, yyyy"
        "mm-dd-yyyy"
```

Where mm is the month, dd is the day, and yyyy is the year. The date must be expressed in Coordinated Universal Time (UTC). You must use leading zeros for the day and month if less than 10.

#### **Comments**

If the date format is invalid, this tag is ignored.

#### **Example**

```
<html>
<head>
<meta http-equiv="SetDate" content="04/15/2004">
</head>
<body>
...
</body>
</html>
```

### **3.18 SETTIME**

The SetTime tag is used to set the current system time on the handheld.

#### **Syntax**

```
http-equiv="SetTime"
content="hh:mm"
        "hh.mm"
```

Where hh is the hour, and mm is the minute. The time must be expressed in Coordinated Universal Time (UTC). You must use leading zeros if less than 10.

## Comments

If the time format is invalid, this tag is ignored.

## Example

```
<html>
<head>
<meta http-equiv="SetTime" content="12:01">
</head>
<body>
...
</body>
</html>
```

## **3.19 SIGNAL**

The Signal tag is used to configure the on-screen WLAN (RF) signal strength meter. This meter overlays the HTML content and is updated at a specified interval. You can change the location and style of the meter. Available styles include a horizontal or vertical single bar meter and a stepped bar meter. The meter may be repositioned by a “touch and drag” stylus action if dragging is not disabled.

## Syntax

```
http-equiv="Signal"
content="Show"
      "Hide"
      "Right"
      "Left"
      "Top"
      "Bottom"
      "x=n"
      "y=m"
      "Steps"
      "AllowDrag"
      "NoDrag"
```

Where x and y are the screen coordinates of the upper left corner of the meter icon. The screen coordinate (0,0) is in the upper left corner of the screen with x increasing to the right and y increasing downward. Right, Left, Top, and Bottom change the orientation of the meter and the placement of the WLAN icon. Steps enables the step style meter. AllowDrag will allow the user to drag the meter, whereas NoDrag will prevent dragging.

## Comments

This tag is used to control the WLAN (RF) meter display. The meter may also be controlled within the CETerm configuration, independent of the Signal tag. The Signal tag will always override the internal configuration and will persist until changed by another Signal tag. Within the CETerm configuration you can specify the update interval and a notification when the strength falls below a designated level.

### Example

```
<html>
<head>
<meta http-equiv="Signal" content="Show">
<meta http-equiv="Signal" content="Steps">
</head>
<body>
The RF meter should be visible.
...
</body>
</html>
```

## **3.20 SIGNALNAVIGATE**

The SignalNavigate tag causes the specified JavaScript or URL to be invoked with WLAN signal information on a regular interval. The interval can be specified within a CETerm configuration dialog.

### Syntax

```
http-equiv="SignalNavigate"
content="javascript:OnSignal('%s', '%s', '%s');"
<!-- or -->
content="/signal.htm?strength=%s&ESSID=%s&MAC=%s"
```

Where the “%s” are replaced with (1) signal strength, (2) ESSID, and (3) the MAC address of the device.

### Comments

This tag will work with or without a visible WLAN signal meter. The strength normally ranges from 0 to 100. The special strength value of -2 indicates that the device is not associated with any access point. The special strength value of -1 indicates that the strength cannot be determined. If you use a URL for the action, in most cases, the URL will navigate away from the current page rather than repeatedly calling a JavaScript method.

## Example

```
<html>
<head>
<meta http-equiv="SignalNavigate"
  content="javascript:OnSignal('%s', '%s', '%s');">
</head>
<body>
<script language=javascript>
function OnSignal(strength, ESSID, MACAddress)
{
  if (strength == -2)
  {
    alert("RF radio is not associated.");
  }
  else if (strength == -1)
  {
    alert("Unable to determine RF signal strength.");
  }
  else
  {
    alert("RF Signal strength = " + strength);
  }
}
</script>
...
</body>
</html>
```

## **3.21 SIP**

The SIP tag controls the visibility of the Soft Input Panel (SIP).

### **Syntax**

```
http-equiv="SIP"
content="Show"
       "Hide"
       "Locked"
```

### **Comments**

This tag is used primarily when the SIP should be displayed upon loading a page. Alternatively, you can use an IDA Action Code via the IDA META tag, an “ida:” URL or a PostIDA call. These latter techniques can be used to change the SIP visibility during the user interaction with the page. The “Locked” value will prevent the SIP from popping up when focus is set to a text input element under Pocket PC 2002 or Windows Mobile 2003. The locked state can also be set in the CETerm configuration dialogs.

**Example**

```
<html>
<head>
<meta http-equiv="SIP" content="Show">
</head>
<body>
The Soft Input Panel (SIP) should be visible.
</body>
</html>
```

**3.22 SIPUp**

The SIPUp tag makes the Soft Input Panel (SIP) visible.

**Syntax**

```
http-equiv="SIPUp"
content=""
```

**Comments**

This tag is supported for compatibility with other browsers. It is preferable to use the SIP META tag or the IDA Action Code "IDA\_SIP\_SHOW" via the IDA META tag, an "ida:" URL or a PostIDA call. The content is ignored, but it must be present.

**Example**

```
<html>
<head>
<meta http-equiv="SIPUp" content="">
</head>
<body>
The Soft Input Panel (SIP) should be visible.
</body>
</html>
```

**3.23 TEXTSIZE**

The TextSize tag is used to set an overall text zoom level. The zoom level can also be changed manually with font sizing buttons on the Toolbar or KeyBar.

**Syntax**

```
http-equiv="TextSize"
content="Smallest"
       "Smaller"
       "Medium"
```

```
"Larger"  
"Largest"
```

### Comments

The TextSize tag is somewhat misnamed. Relative text sizes are determined by the HTML content. TextSize applies an overall zoom factor to the sizes set in the HTML. The initial zoom value will be the last value set for the session whether by a TextSize tag or by manual actions.

### Example

```
<html>  
<head>  
<meta http-equiv="TextSize" content="Largest">  
</head>  
<body>  
This is the Largest TextSize zoom.  
...  
</body>  
</html>
```

## **3.24 TIMERINTERVAL**

The TimerInterval tag is used to specify the interval between activations of the TimerNavigate action.

### Syntax

```
http-equiv="TimerInterval"  
content="milliseconds"
```

Where milliseconds is the number of milliseconds between activations.

### Comments

The effect of TimerInterval and TimerNavigate tags can be accomplished using the Javascript `setInterval()` and `setTimeout()` methods on the window DOM object. Using the Javascript methods is recommended.

### Example

See example under TimerNavigate.

## **3.25 TIMERNAVIGATE**

The TimerNavigate tag causes the specified JavaScript or URL to be invoked on a regular interval. The interval is specified with the TimerInterval tag.

## Syntax

```
http-equiv="TimerNavigate"  
content="javascript:OnTimer('%s');"  
  
content="/timer.htm?time=%s"
```

Where the “%s” is replaced with the current time in the form hh:mm:ss.

## Comments

The effect of TimerInterval and TimerNavigate tags can be accomplished using the Javascript `setInterval()` and `setTimeout()` methods on the window DOM object. Using the Javascript methods is recommended.

## Example

```
<html>  
<head>  
<meta http-equiv="TimerInterval" content="1000">  
<meta http-equiv="TimerNavigate"  
  content="javascript:OnTimer('%s');">  
</head>  
<body>  
Current time: <div id="timerDiv">junk</div>  
<script language=javascript>  
function onTimer(time)  
{  
  timerDiv.innerHTML = time;  
}  
</script>  
...  
</body>  
</html>
```

### **3.26 ZEBRALABEL COMPLETE OR PLSERIESLABEL COMPLETE**

The ZebraLabel\_Complete tag is used to report the status of a print from the ZebraLabel\_Print tag. The alternative identifier PLSeriesLabel\_Complete will also be recognized.

## Syntax

```
http-equiv="ZebraLabel_Complete"  
content="javascript:PrintStatus('%ld');"  
  
content="http://10.1.1.8/print.htm?status=%ld"
```

## Comments

This tag is supported for compatibility with other browsers. It is preferable to use other techniques for sending print content to a printer. See the advanced topic “Printing from HTML”.

The status code of the print action will be substituted into the “%ld” location. A status of 0 indicates success, 1 indicates failure. See ZebraLabel\_Print for more details.

## Example

```
<html>
<head>
<meta http-equiv="ZebraLabel_Complete"
      content= "http://10.1.1.8/print.htm?status=%ld"
<meta http-equiv="ZebraLabel_Print"
      content="! 0 200 200 581 1\r\nLABEL ... PRINT\r\n">
</head>
<body>
The print should be produced.
This message should not be visible.
</body>
</html>
```

## **3.27 ZEBRALABEL PRINT OR PLSERIESLABEL PRINT**

The ZebraLabel\_Print tag contains data that is sent to a Zebra printer. The ZebraLabel\_Complete tag must also be present to report the status of the print. The alternative identifier PLSeriesLabel\_Print will also be recognized.

## Syntax

```
http-equiv="ZebraLabel_Print"
content="! 0 200 200 581 1\r\nLABEL ... PRINT\r\n">
```

## Comments

This tag is supported for compatibility with other browsers. It is preferable to use other techniques for sending print content to a printer. See the advanced topic “Printing from HTML”.

The contents must be less than 1024 characters. Any “\r” sequences are replaced by the CR (0x0d) character. Any “\n” sequences are replaced by the NL



(0x0a) character. The sequence “\xXX” is replaced by a byte with the hexadecimal value XX. Use “\\” for a literal backslash.

The content is sent to the printer that is currently configured under the CETerm Printer tab. This may be a serial attached printer, Bluetooth, or network accessible printer.

### Example

```
<html>
<head>
<!-- NOTE - The content must be on one line -->
<!-- NOTE - It is split here for readability -->
<meta http-equiv=" ZebraLabel_Print "
  content="! 100 200 200 1225 1\r\n
        TEXT 0 2 1 0 Vendor\r\nTEXT 5 2 1 20 Adidas®\r\n
        TEXT 0 2 1 80 Model\r\nTEXT 5 2 1 100 Adidas®\r\n
        TEXT 5 2 1 135 Storm\r\nTEXT 5 0 1 200 Features\r\n
        TEXT 5 0 1 225 Cross-Training\r\n
        TEXT 5 0 1 250 Arch Support\r\n
        TEXT 5 0 1 275 Leather Upper\r\n
        TEXT 5 0 1 300 Cushioned Insole\r\n
        TEXT 4 1 100 435 $61.99\r\n
        TEXT 0 2 1 550 _____\r\n
        TEXT180 4 1 250 700 $61.99\r\n
        TEXT180 5 0 270 740 000000292818\r\n
        BARCODE UPCA 1 1 50 100 750 000000292818\r\n
        TEXT180 5 0 320 910 WHITE/RED\r\n
        TEXT180 5 0 320 935 Available in:\r\n
        TEXT180 0 2 320 1025 _____\r\n
        JOURNAL\r\nPRINT\r\n">
<meta http-equiv="ZebraLabel_Complete"
  content= "http://10.1.1.8/print.htm?status=%ld"
</head>
<body>
The print should be produced.
This message should not be visible.
</body>
</html>
```

## 4.0 Advanced Topics

In this chapter, we discuss several common, but advanced, topics for creating robust Web based applications. We discuss how to overcome some limitations of the Pocket IE browser behavior.

### **4.1 NAVIGATING TO PRE-CONFIGURED URLS**

The Naurtech Web Browser allows you to configure pre-defined URL's which can be associated with a hardware key, a KeyBar button, or activated from the within the browser via a link or via JavaScript.

The URL is defined within a User Text string and this text can be submitted for navigation by any of the ways listed above. For example, with the following content in User Text 1:

```
\IDA_URL\file:///Program Files/myhelppage.htm\r
```

You can navigate to this local help page by activating the IDA code IDA\_USTRING\_1 which has the friendly name "Text 1". You can associate a hardware key or a KeyBar button to this IDA code. Alternatively, you could use any of the ways listed in the IDA Action Codes section to trigger the navigation. For example the following link will send you to the help page:

```
<a href="ida:IDA_USTRING_1">Show Help Page</a>
```

Be sure to provide a way to navigate back to the application from your help page. Note the required "IDA\_URL" at the beginning of the URL and the "\r" at the end. If your URL contains a literal backslash, it must be escaped with a second backslash "\\".

It is possible to change the contents of User Text from within JavaScript with the "SetProperty" method of the CEBrowseX or "external" objects. Thus, this storage can be used to maintain some persistent, device-local, information.

### **4.2 CONTROLLING THE SCANNER**

We have shown how the Scanner and ScannerNavigate META tag identifiers can be used to control the scanner. You can also use IDA codes to provide additional control.

The scanner can be enabled or disabled via an IDA code. You can also activate a "soft trigger" on most scanner equipped handhelds. This will start the scanner without the user needing to hold the trigger. This mode is useful if the application is driving the scanner in a "read loop" until a desired number of scans are completed.

```
<meta http-equiv="Scanner" content="Enabled">
<meta http-equiv="ScannerNavigate"
  content="Javascript:onscan('%s','%s','%s','%s','%s');">
...
<object id="CEBrowseX"
  classid="clsid:D14943BD-4900-453E-8582-725F21A57E0C"
  height=0, width=0>
</object>
...
<input type="button" name="autoscan" value="Auto Scan"
onclick="autoscan();">
...
<script language="javascript">
function autoscan()
{
  if (autoscantrigger)
  {
    autoscantrigger = 0;
  }
  else
  {
    autoscantrigger = 1;
    document.CEBrowseX.CETerm.PostIda( "IDA_SCAN_TRIGGER", 0 );
  }
}

function onscan(data, source, type, time, length)
{
  if (document.form1.scan1.value == "")
  {
    document.form1.scan1.value = data;
  }
  else if (document.form1.scan2.value == "")
  {
    document.form1.scan2.value = data;
  }
  else if (document.form1.scan3.value == "")
  {
    document.form1.scan3.value = data;
    autoscantrigger = 0;
  }
  else
  {
    autoscantrigger = 0;
    document.CEBrowseX.CETerm.PostIda( "IDA_VIBRATE_500", 0 );
    alert("Form full, scan discarded.");
  }

  if (autoscantrigger)
  {
    document.CEBrowseX.CETerm.PostIda( "IDA_SCAN_TRIGGER", 0 );
  }
}
```

```
}  
</script>
```

### **4.3 INPUT FOCUS AND THE TAB KEY**

One of the first limitations you are likely to notice about Pocket Internet Explorer is that the Tab key does not always move the input focus as you expect from using the desktop Internet Explorer. Pocket IE on the Pocket PC 2002 and Windows Mobile platforms has the most limited behavior.

The Naurtech Web Browser provides enhanced focus control on the Pocket PC platforms. If you only need to move between native HTML text input elements, then the Tab key should perform as you expect from the desktop. The Tab key should also work as expected on Windows CE .NET or 5.0 platforms. However, currently a Tab post-amble will not advance the focus on all platforms.

Some other vendors provide a text input control like Naurtech's TextX to manage the focus under Pocket PC 2002 or Windows Mobile. TextX will perform similarly, but we recommend you avoid using ActiveX controls for input to simplify and accelerate development. If you must use the ActiveX text input, then you can provide additional focus management.

The following example shows how to track the focus and advance the focus when the Tab key is pressed. These techniques will work for the native HTML text input elements on Windows CE platforms or with the ActiveX text input on Pocket PC 2002 and Windows Mobile. Both support the OnKeyPress event handlers.

This first example uses native HTML text input on Windows CE .NET. Note the use of "document.activeElement" in this example:

```
<html>  
<head>  
<title>Naurtech Tab Demo Page</title>  
<meta http-equiv="Scanner" content="Enabled">  
<meta http-equiv="ScannerNavigate"  
content="Javascript:onscan('%s','%s','%s','%s','%s');">  
<meta http-equiv="OnKey0x0D"  
content="Javascript:document.form1.clear.click();">  
  
</head>  
<body scroll=no onload="Javascript:document.form1.scan1.focus();">  
<form name=form1>  
<center>  
<font size=+2>  
Naurtech Tab Demo<br>  
</font>
```

```
<br>
Focus starts in first input<br>
<input type=text name="scan1" value="" size=30
onkeypress="myonkey();"><br>
<input type=text name="scan2" value="" size=30
onkeypress="myonkey();"><br>
<input type=text name="scan3" value="" size=30
onkeypress="myonkey();"><br>
<input type=button name="clear" value="Clear Data"
onclick="myclear();" onkeypress="myonkey();">
</center>
</form>

<script for="document" event=onkeypress>
// IMPORTANT: This handler is used when focus is not already
in an input object
myonkey();
</script>

<script language=javascript>

// Handle the key event
function myonkey()
{
    if (window.event.keyCode == 9) // look for tab key
    {
        nextfield( document.activeElement );

        window.event.cancelBubble = true;
    }
}

// Move from the current field to the next field
function nextfield( current )
{
    if (current == document.form1.scan1)
    {
        document.form1.scan2.focus();
    }
    else if (current == document.form1.scan2)
    {
        document.form1.scan3.focus();
    }
    else
    {
        document.form1.scan1.focus();
    }
}

// Clear the fields
function myclear()
{
    document.form1.scan1.value = "";
```

```
        document.form1.scan2.value = "";
        document.form1.scan3.value = "";
        ring();
        document.form1.scan1.focus();
    }

    // Play a sound
    function ring()
    {
        external.OS.PlayTone(10, 1000, 200);
    }

    // Handle the scan data
    function onscan(data, source, type, time, length)
    {
        var current = document.activeElement;
        if (current == document.form1.scan1 ||
            current == document.form1.scan2 ||
            current == document.form1.scan3)
        {
            current.value = data;

            // NOTE: A postamble of "\t" will not be seen when using
            // NOTE: the ScannerNavigate feature.
            // NOTE: The following method advances the focus.
            nextfield( current );
        }
        else
        {
            alert("Focus is not in an input field.");
        }
    }
}

</script>
</body>
</html>
```

Here is a similar Tab navigation example using the TextX input object. This example does not use the ScannerNavigate tag. A Tab in the scanner postamble "\t" will advance the focus. However, a scan when focus is not in an input field will be lost. Note the use of hidden fields to pass the TextX contents back to the host.

```
<html>
<head>
<title>Naurtech TextX</title>
<meta http-equiv="Scanner" content="Enabled">
</head>
<body onload="javascript:document.myform1.textx1.SetFocus(1);">
<form name=myform1
```

```
onload="javascript:document.myform1.textx1.SetFocus();">
Naurtech TextX Tab Test
<br>
(Windows Mobile 2003)
<br><br>
<center>
TextX1 <object id="textx1"
  classid="clsid:6402E27B-CD4F-448C-BAEA-F3558242459D"
  height=22, width=150>
<span style="color:red">TextX1 failed to load!
-- Please check browser configuration --</span>
</object>
<br>
TextX2 <object id="textx2"
  classid="clsid:6402E27B-CD4F-448C-BAEA-F3558242459D"
  height=22, width=150>
</object>
<br>
<br>
<input type=button name="submitbutton" value="Submit"
OnClick="mysubmit()" >
<br>
<input type=hidden name="hiddentextx1">
<input type=hidden name="hiddentextx2">
</center>
</form>

<script language=javascript for="textx1" event="OnKeyPress(key)">
  return MyKeyPress(key, document.myform1.textx2);
</script>

<script language=javascript for="textx2" event="OnKeyPress(key)">
  return MyKeyPress(key, document.myform1.textx1);
</script>

<script language=javascript>
function mysubmit()
{
  // Put values of TextX into hidden input fields for host
  // Otherwise, host won't see control contents
  document.myform1.hiddentextx1.value = document.myform1.textx1.value;
  document.myform1.hiddentextx2.value = document.myform1.textx2.value;

  // Submit the form
  document.myform1.submit();
}

function MyKeyPress(key, control)
{
  if (key == 9)
  {
    // Advance the focus
    control.SetFocus(1);
  }
}
```

```
        // Focus has moved, cancel the event
        return -1;
    }
    // Just return the same key
    return 0;
}
</script>
</body>
</html>
```

## **4.4 SESSION LAUNCHER**

One interesting use of the Web Browser is as a launcher for other browser or terminal emulation sessions. The following example shows how a static web page on the device could be used to activate other sessions. This example assumes that this web page is displayed under Session 4. Each of the other sessions must be configured for the desired activity:

```
<html>
<head>
<title>Naurtech Launch Page</title>
<meta http-equiv="PowerOn" content="Javascript:poweron();">
<meta http-equiv="Scanner" content="Disabled">
<meta http-equiv="OnKey0x70"
    content="Javascript:startsession(1);"><! F1>
<meta http-equiv="OnKey0x71"
    content="Javascript:startsession(2);"><! F2>
<meta http-equiv="OnKey0x72"
    content="Javascript:startsession(3);"><! F3>
</head>
<body>
<object id="CEBrowseX"
    classid="clsid:D14943BD-4900-453E-8582-725F21A57E0C"
    height=0, width=0>
</object>
<form name=form1>
<font size=9>
<center>
Main Menu<br>
</center>
<a href="javascript:startsession(1);">F1. Pick</a><br>
<a href="javascript:startsession(2);">F2. Cycle Count</a><br>
<a href="javascript:startsession(3);">F3. Receive</a><br>
<br>
</font>
</form>

<script language=javascript>
function poweron()
{
    // Navigate to Main Menu on resume
```



```
    sendida( "IDA_SESSION_S4", 0 );
}

function startsession(id)
{
    if (id >= 1 && id <=4)
    {
        // Switch to session
        sendida("IDA_SESSION_S" + id, 0);
        // Connect if not connected
        sendida("IDA_SESSION_CONNECT", id);
        return;
    }
    else
    {
        alert("Unsupported session.");
    }
}

function sendida(ida, s)
{
    // PocketPC 2002
    //CEBrowseX.CETerm.SendIDA(ida, s);

    // Windows Mobile
    document.CEBrowseX.CETerm.SendIDA(ida, s);
}
</script>
</body>
</html>
```

## **4.5 HOW TO IDENTIFY THE CURRENT BROWSER**

The Naurtech Web Browser returns the same User-Agent HTTP value as does the standard Windows Pocket IE browser. This is necessary to indicate the fundamental capabilities of the browser to the Web server. There are other ways for the Web application to determine if the Naurtech Web Browser is the client.

You can use the GetUnitInformation META tag identifier to retrieve the version number and send this to the host. As of this writing, the current version number is "5.1.0". Most other browsers will not support the GetUnitInformation feature and your HTML must be prepared to return an empty value.

```
<html>
<head>
<meta http-equiv="GetUnitInformation"
      content="Javascript:saveunitinfo('%s','%s','%s');">
<script language=javascript>
```

```
var serial;
var ceuuid;
var version;

function saveunitinfo(serialarg, ceuuidarg, versionarg)
{
    serial = serialarg;
    ceuuid = ceuuidarg;
    version = versionarg;
}
</script>
</head>
<body onload="javascript:checkversion();" >
...check version variable and return info to host....
</body>
```

Alternatively, you can use a CEBrowseX or “external” object to retrieve the version in a CETerm.GetProperty call. Other browsers will be able to create a CEBrowseX object if it is installed, but CETerm.GetProperty will not return a valid result.

```
<object id="CEBrowseX"
    classid="clsid:D14943BD-4900-453E-8582-725F21A57E0C"
    height=0, width=0>
</object>
...
<script language=javascript>
function checkversion()
{
    version = document.CEBrowseX.CETerm.GetProperty( "app.version" );
    // check the version value, return info to host
}
```

## **4.6 DEVICE INFORMATION**

You can retrieve information about the device configuration using the CETerm.GetProperty method. See the CEBrowseX section for details

## **4.7 SYMBOL WEB CLIENT**

An early browser developed for Symbol devices running the Palm OS had some extensions for controlling the scanner and printing. These extensions were implemented through custom attributes on standard HTML tags and custom HTML tags. The Naurtech Web Browser running on Windows CE .NET or CE 5.0 platforms supports these extensions. These extensions are not supported on Pocket PC 2002 or Windows Mobile based devices. These extensions are supported for compatibility with existing Web based applications. New Web applications should use the META tags.

On the text INPUT element we support the “stiscan” and “stisubmit” attributes. If present, the stiscan attribute controls which symbologies may be inserted into the input element. If no value is specified, all symbologies are allowed. If an empty string is specified, no symbologies are allowed. When the “stisubmit” attribute is present, the enclosing form will be submitted after scanned data is inserted in the element.

```
<input type=text name=upc size=12 maxlength=16
stiscan="ABCDEFGHijklmn" stisubmit>
```

The custom element BEEP is supported to sound a tone:

```
<NAURTECH:beep frequency="f" repeat="n" duration="d">
```

where f is in Hz, n is a count, and d is in milliseconds. The handheld device must support a tone generator. The “NAURTECH:” namespace label is required.

The custom element PRINT is supported to send print content to the printer.

```
<NAURTECH:print name="printer" type="N" announce="yes" retries=3>
Print content here\n\r
</print>
```

where “printer” is the destination name of the printer, and type is the connection type. The “NAURTECH:” namespace label is required. The following connection types are supported:

Connection Type	Name Values	Description
Q	Name is the Windows print queue name	Print to Windows print queue
N	Printer IP address or hostname and port. For example, “192.168.1.101:6101”	Print directly to port
dddd	Same as ‘N’ but port specified as dddd.	Print directly to port
I, B, S, (other)	(ignored)	Print to configured printer for current session.

Standard escape characters such as “\r”, “\n”, and “\xXX” will be substituted in the print contents. Literal line breaks in the content are ignored, you must use “\r\n” if the printer language requires a line break.

You can set the values of the currently configured network printer using the `CETerm.SetProperty` method on the `CEBrowseX` or “external” objects.

## 5.0 Printing from HTML

There are numerous ways to print from the Naurtech Web Browser. Once a printer is configured, print content may be specified via a special META tag, the CEBrowseX or “external” objects, the custom PRINT tag, or with any ActiveX control designed to print from a browser.

The Naurtech Web Browser maintains the printer configuration within the Session configuration dialogs. Here you may specify a serial attached, Bluetooth, IrDA, or network attached printer. We support both Windows Print Queues and direct-to-port printing for network printers. See the CETerm User’s Manual for more details on configuring a printer.

All of the techniques for specifying print content allow common escape sequences to be embedded which will be converted to non-printable characters. These include the carriage return (CR – “\r”), linefeed (LF – “\n”), and general hexadecimal bytes (0xXX).

### **5.1 PRINTING WITH A META TAG**

See the “ZebraLabel\_Print” and “ZebraLabel\_Complete” identifiers in the META tag section for details on initiating a print from a META tag.

### **5.2 PRINTSTRING AND PRINT METHODS**

The PrintString method on the CEBrowseX control and the Print method on the “external” object behave similarly. In both cases, a string is constructed and sent to the current printer.

```
<a href="Javascript:myprint();">Test External Print</a>
<script language=javascript>
function myprint()
{
external.Print(
"! 100 200 200 1225 1\r\n" +
"TEXT 0 2 1 0 Vendor\r\n" +
...
"JOURNAL\r\n" +
"PRINT\r\n"
);
}
</script>
```

### **5.3 NAURTECH:PRINT TAG**

The custom PRINT tag is supported for compatibility with other browsers. See the Symbol Web Client section for a discussion of this tag.

## **5.4 ACTIVE X PRINTING CONTROLS**

There are numerous ActiveX controls available to send print content to a printer. These are available from handheld manufacturers, printer manufacturers, and third-party sources. All controls which follow accepted standards will work within the Naurtech Browser. Currently we do not recommend any specific control.

## **6.0 CEBrowseX Control**

The Naurtech CEBrowseX control is used to send commands to the Naurtech Web Browser and to access device and application information. This control is installed when you install the Naurtech Web Browser.

On Windows CE .NET or CE 5.0 based devices, most of the CEBrowseX methods are available from the Document Object Model (DOM) "external" object. In this case you do not need to instantiate a CEBrowseX object.

**ALERT:** The CEBrowseX control has been updated in CETerm V5.1 to provide access to the new "CETerm" and "OS" automation objects used in the CETerm script engine. The new "CETerm" object provides all the features that were previously available from CEBrowseX. The CEBrowseX object should now only be used to access the CETerm or OS objects. Existing web pages will continue to work but should be converted to the new syntax as they are updated.

**ALERT:** See the CETerm Scripting Guide for full details on the CETerm and OS objects and their methods and properties. Scripting in CETerm offers additional capabilities to enhance your web based applications.

## **SYNTAX**

```
<object id="CEBrowseX"  
  classid="clsid:D14943BD-4900-453E-8582-725F21A57E0C"  
  height=0, width=0>
```

The CEBrowseX control has no visible display, but you must specify height and width of zero to prevent the object from consuming space on the page.

## **CLASSID**

The CEBrowseX CLASSID is:

```
CLASSID="{clsid:D14943BD-4900-453E-8582-725F21A57E0C}"
```

## **METHODS**

The following methods are available. Note that most are marked as deprecated (**DEP**) in favor of using the CETerm or OS object and their associated methods. The CETerm and OS objects are obtained from the corresponding CEBrowseX properties.

<b>Method</b>	<b>Action</b>
PostIDA	Send a command to a session (asynchronous) ( <b>DEP</b> )
SendIDA	Send a command to a session (synchronous) ( <b>DEP</b> )
SendText	Send text content to a session ( <b>DEP</b> )
GetProperty	Get a property value from the Web Browser ( <b>DEP</b> )
SetProperty	Set a property value in the Web Browser ( <b>DEP</b> )
PlaySound	Play a tone or wave file on the device (CEBrowseX only) ( <b>DEP</b> )
PlayTone	Play a tone on the device (external only) ( <b>DEP</b> )
PrintString	Send content to a printer (CEBrowseX only)
Print	Send content to a printer (external only)

*PostIDA( IDACode, session )*

PostIDA sends an IDA action command to the Web Browser and directs it to the specified session. Valid session values are 1-4. The special session value of 0 will send the command to the current session. See the Appendix for IDA values.

The PostIDA command will return before the Web Browser acts on the command. We recommend using PostIDA rather than SendIDA. There are only rare situations when SendIDA must be used.

*SendIDA( IDACode, session )*

SendIDA sends an IDA action command to the Web Browser and directs it to the specified session. Valid session values are 1-4. The special session value of 0 will send the command to the current session. See the Appendix for IDA values.

The SendIDA method will attempt to complete the action before returning. We recommend using PostIDA rather than SendIDA. There are only rare situations when SendIDA must be used.

*SendText( text, session )*

SendText sends a text string to the Web Browser and directs it to the specified session. Valid session values are 1-4. The special session value of 0 will send the command to the current session.

The text string may include IDA symbolic names between backslash characters '\'. The IDA codes will be interpolated as the text is sent.

*Value = GetProperty( propertyName )*

GetProperty will return the named property value. This may be a device property, application property, or session property. See the Appendix for a list of available properties.

*Status = SetProperty( propertyName, propertyValue )*

SetProperty will assign the given value to the named property. See the Appendix for a list of available properties.

*PlaySound( sound )*

PlaySound will play a tone or wave file depending on the specified sound. Any wave file on the device may be specified. Use the complete file path if the file is not in the \Windows directory. PlaySound is not available on the "external" object (see PlayTone).

If the handheld device contains a programmable tone generator, the sound parameter may also be a string which defines a sequence of tones to play. The syntax is given below:

"vfffddd" – where

vv – is the volume 01-10

fff – is the frequency in 10's of MHz, 000-999

ddd – is the duration in 10's of millisec, 000-999



*PlayTone( volume, frequency, duration )*

PlayTone will play a tone if supported by the handheld hardware. PlayTone is not available on the CEBrowseX object (see PlaySound).

volume – is the volume 00 -10 (0 is off, 10 is loudest)

frequency – is the frequency in Hz.

duration – is the duration in millisec.

*Status = PrintString( printData ) or Status = Print( printData)*

PrintString and Print will send the printData to the currently configured printer.

PrintString is only available on the CEBrowseX object. Print is only available on the “external” object.

The printData string may contain escape characters for CR (\r), Newline (\n), and hexadecimal bytes (\xXX).

## **PROPERTIES**

The CEBrowseX properties are used to return the CETerm or OS objects. Both are read-only.

<b>Property</b>	<b>Description</b>
CETerm	CETerm automation object
OS	OS automation object

## **EVENTHANDLERS**

The CEBrowseX control has no event handlers.

## 7.0 TextX Control

The Naurtech TextX control is a replacement for the HTML text input element on Pocket PC 2002 and Windows Mobile 2003 devices. The native HTML text input element does not support most events and many properties. Using the TextX object gives much greater control over user input. However, there are drawbacks to using the control, such as increased HTML complexity and no desktop equivalent to simplify development. This control is not installed when you install the Naurtech Web Browser. Please contact Naurtech Support if you need this control.

You may create as many instances of this control, each as a separate input element, as you need on a page. Each instance must have a unique ID.

Some other browsers must use an ActiveX control for text input because they are not able to prevent the auto-popup of the SIP when focus is placed in a native HTML text input element. The Naurtech Web Browser controls the SIP behavior independently and thus does not require the TextX control for this purpose. Also, the Naurtech Web Browser implements Tab key behavior between native HTML text input elements, eliminating another primary reason to use TextX. The third most common reason to use an ActiveX text object is to submit a form when the Enter key is pressed or received as a scanner post-amble. We recommend using the ScannerNavigate META tag for robust scanner input but we also support the OnKey META tag within native HTML text input elements. This eliminates the third common reason for the TextX control.

There is no need to use the TextX control on Windows CE .NET platforms because the native HTML text input element has equivalent features.

In summary, we recommend against using the TextX control, but we provide it for those situations where the features are required.

### **SYNTAX**

```
<OBJECT ID="textx"  
  CLASSID="clsid:6402E27B-CD4F-448C-BAEA-F3558242459D"  
  HEIGHT=20,  
  WIDTH=200>  
<PARAM NAME=ALIGNMENT VALUE="left">  
<PARAM NAME=AUTOHSCROLL VALUE="true">  
<PARAM NAME=AUTOVSCROLL VALUE="false">  
<PARAM NAME=BORDER VALUE="true">  
<PARAM NAME=FONTBOLD VALUE="false">  
<PARAM NAME=FONTFIXEDPITCH VALUE="false">  
<PARAM NAME=FONTITALIC VALUE="false">  
<PARAM NAME=FONTNAME VALUE="Tahoma">  
<PARAM NAME=FONTSIZE VALUE=10 >
```

```
<PARAM NAME=FONTUNDERLINE VALUE="false">
<PARAM NAME=FONTWEIGHT VALUE="400">
<PARAM NAME=MAXLENGTH VALUE=20 >
<PARAM NAME=NUMBERONLY VALUE="false">
<PARAM NAME=PASSWORD VALUE="false">
<PARAM NAME=VALUE VALUE="-Empty-">
<PARAM NAME=WANTRETURN VALUE="false">
</OBJECT>

<SCRIPT LANGUAGE=JAVASCRIPT FOR="textx" EVENT="OnChange()">
MyOnChange();
</SCRIPT>
<
<SCRIPT LANGUAGE=JAVASCRIPT FOR="textx" EVENT="OnClick(x,y)">
MyOnClick(x, y);
</SCRIPT>

<SCRIPT LANGUAGE=JAVASCRIPT FOR="textx" EVENT="OnFocus()">
// Show the SIP
textx.ShowSIP( true );
</SCRIPT>

<SCRIPT LANGUAGE=JAVASCRIPT FOR="textx" EVENT="OnLostFocus()">
// Hide the SIP
textx.ShowSIP( false );
</SCRIPT>

<SCRIPT LANGUAGE=JAVASCRIPT FOR="textx" EVENT="OnKeyDown(key)">
MyOnKeyDown( key );
</SCRIPT>

<SCRIPT LANGUAGE=JAVASCRIPT FOR="textx" EVENT="OnKeyPress(key)">
// Check key value
return MyCheckKey( key );
</SCRIPT>

<SCRIPT LANGUAGE=JAVASCRIPT FOR="textx" EVENT="OnKeyUp(key)">
MyOnKeyUp( key );
</SCRIPT>
```

## **CLASSID**

The TextX CLASSID is:

```
CLASSID="clsid:6402E27B-CD4F-448C-BAEA-F3558242459D"
```

## **METHODS**

Two methods are available on the TextX object.

***objectname.SetFocus( select )***

SetFocus will set the current focus to this TextX object. If select is true, then all of the text in the control will be selected.

***objectname.ShowSIP( visible )***

ShowSIP will cause the SIP to be visible if the argument is true. It will hide the SIP if the argument is false. This may also be accomplished via an IDA code (see IDA Action Codes).

**PROPERTIES**

The TextX control has several properties to control the appearance and behavior of the object. All properties may be set in an initial "PARAM" statement or set dynamically via JavaScript, unless described otherwise.

<b>Property</b>	<b>Description</b>	<b>Values</b>	<b>Default Value</b>
Alignment	Text justification	LEFT, RIGHT, CENTER	LEFT
AutoHScroll	Text will autoscroll horizontally	TRUE, FALSE	TRUE
AutoVScroll	Text will autoscroll vertically on Enter	TRUE, FALSE	FALSE
Border	Show border lines	TRUE, FALSE	TRUE
FontBold	Use bold font	TRUE, FALSE	FALSE
FontFixedPitch	Use fixed pitch font	TRUE, FALSE	FALSE
FontItalic	Use italic font	TRUE, FALSE	FALSE
FontName	Font for object	(any on device)	System
FontSize	Size of font	6-28	10
FontUnderline	Use underlined font	TRUE, FALSE	FALSE
FontWeight	Weight of font	300=light 400=normal 700=bold 900=heavy	400
MaxLength	Maximum text length	0-8192 (0 – no limit)	0
NumberOnly	Only accept digits	TRUE, FALSE	FALSE
Password	Display text as '*'s	TRUE, FALSE	FALSE
Value	Text content	(any text)	(empty)
WantReturn	Enter gives a newline in Multiline control	TRUE, FALSE	FALSE

Some font properties may conflict or may not yield visible differences on all devices. `FontBold` is equivalent to `FontWeight=700`. Specifying a font name will override some other attributes.

To create a multi-line control, simply use a large `HEIGHT`. Multiline content with embedded “`\r\n`” cannot be set via a `PARAM` element. You can set such content dynamically in JavaScript via the `Value` property.

More than `MaxLength` characters can be inserted in the text box via the `Value` property.

## **EVENTHANDLERS**

The `TextX` control supports event handlers which are invoked when the user interacts with the control. Only `OnKeyPress` returns a value. This value can be set to cancel (or ignore) the event. The return values of all other handlers are ignored and the associated events cannot be canceled.

*void OnChange()*

The `OnChange` handler is called when the contents of the control have been changed and focus is lost from the control.

*void OnClick( int x, int y )*

The `OnClick` handler is called when the user taps the stylus on the control. The `x` and `y` coordinates of the tap are provided in the arguments.

*void OnFocus()*

The `OnFocus` handler is called when the control receives input focus.

*void OnLostFocus()*

The `OnLostFocus` handler is called when the control loses input focus.

*void OnKeyDown( int vkey )*

The `OnKeyDown` handler is called when the user presses a key with focus in the control. The `vkey` argument is the Windows `VK` Code for the key. This handler

may also be called when keys are simulated within the Web Browser via KeyBar buttons or remapped physical keys.

*int OnKeyPress( int akey )*

The OnKeyPress handler is called when a key is pressed which generates an ASCII character. The akey argument contains the ASCII character which is generated. The return value from this handler controls the event propagation. A return value of 0 will continue the event, a value of -1 will cancel the event. Any positive value will replace the akey in this event. This handler is called before the character is placed in the input object, so changing the return value will change the character placed in the object.

*void OnKeyUp( int vkey )*

The OnKeyUp handler is called when the user releases a key with focus in the control. The vkey argument is the Windows VK Code for the key. This handler is not called when keys are simulated from within the Web Browser.

## Appendix 1 - Properties

The properties listed in this appendix may be accessed via the GetProperty and SetProperty methods on the CEBrowseX ActiveX control or via the DOM “external” object under Windows CE .NET or 5.0. Properties marked (RO) are read-only and may not be set with SetProperty.

**ALERT:** Use the CETerm object methods GetProperty and SetProperty to access properties. There are many additional properties described in the CETerm Scripting Guide. The following list is included for older applications that used the deprecated GetProperty and SetProperty methods of the CEBrowseX control.

Property Name	Description
<b>Application Properties</b>	
app.buildid (RO)	Program build identifier
app.name (RO)	Program name
app.session.active (RO)	Currently active session
app.usertext.NN	User text # NN contents, NN is 1-64
app.version (RO)	Program version
<b>Device Properties</b>	
device.batterystatus (RO) device.battery.statustext (RO)	Current battery status string
device.battery.status (RO)	Current battery status -1 – unknown, 0 – critical, 1 – warning, 2 – low, 3 – medium, 4 – high, 5 - charging
device.battery.level (RO)	Current battery strength - 0 – 100 -1 – unknown
device.deviceid (RO)	Device ID string
device.ipaddress (RO)	IP Address of handheld
device.macaddress (RO)	MAC Address of handheld
device.platformid (RO)	Windows CE Platform ID
device.presetid (RO)	Windows CE Preset ID
device.rf.strength (RO)	RF signal strength 0-100, -2 – not associated with AP, -1 – unknown
device.rf.status (RO)	RF status -1 – unknown, 0 – unassociated, 1 – poor, 2 – fair, 3 – good, 4 – very good, 5 – excellent
device.serialnumber (RO)	Device serialnumber

<b>Property Name</b>	<b>Description</b>
<b>Session Properties</b>	
session.connection.host	Session host (or home URL)
session.connection.port	TE session port
session.connection.type	Session type 3270, 5250, VT220, HTML
session.printer.network.queue	



## Appendix 2 - IDA Action Codes

Many IDA codes apply only to a Terminal Emulation session but are listed here for completeness. Some IDA codes can only be used in restricted circumstances, such as IDA\_URL.

Symbolic Name	Friendly Name	Description
IDA_BEL	Bell	
IDA_BS	Backspace	
IDA_HT	Horizontal Tab	
IDA_TAB	Tab	
IDA_LF	Linefeed	
IDA_VT	Vertical Tab	
IDA_FF	Form Feed	
IDA_CR	Carriage Return	
<b>Printable ASCII</b>		
IDA_SPACE	<Space>	
IDA_EXCLAMATION_MARK	!	
IDA_DOUBLE_QUOTE	"	
IDA_NUMBER_SIGN	#	
IDA_DOLLAR_SIGN	\$	
IDA_PERCENT	%	
IDA_AMPERSAND	&	
IDA_SINGLE_QUOTE	'	
IDA_LEFT_PAREN	(	
IDA_RIGHT_PAREN	)	
IDA_ASTERISK	*	
IDA_PLUS	+	
IDA_COMMA	,	
IDA_HYPHEN	-	
IDA_PERIOD	.	
IDA_SLASH	/	
IDA_0	0	
IDA_1	1	
...	...	
IDA_9	9	
IDA_COLON	:	
IDA_SEMICOLON	;	
IDA_LESS_THAN	<	
IDA_EQUAL	=	
IDA_GREATER_THAN	>	

<b>Symbolic Name</b>	<b>Friendly Name</b>	<b>Description</b>
IDA_QUESTION_MARK	?	
IDA_AT	@	
IDA_A	A	
IDA_B	B	
...	...	
IDA_Z	Z	
IDA_LEFT_BRACKET	[	
IDA_BACKSLASH	\	
IDA_RIGHT_BRACKET	]	
IDA_CARET	^	
IDA_UNDERSCORE	_	
IDA_BACKTICK	`	
IDA_a	a	
IDA_b	b	
...	...	
IDA_z	z	
IDA_LEFT_BRACE	{	
IDA_PIPE		
IDA_RIGHT_BRACE	}	
IDA_TILDE	~	
IDA_DEL	DEL	
C1 ASCII Controls		
IDA_IND	Index	
IDA_NEL	Next Line	
IDA_HTS	Horiz Tab Set	
IDA_RI	Reverse Index	
IDA_SS2	Single Shift 2	
IDA_SS3	Single Shift 3	
IDA_DCS	Device Ctrl Str	
IDA_PU1	Private Use One	
IDA_PU2	Private Use Two	
IDA_CSI	Ctrl Seq Intro	
IDA_ST	String Term	
IDA_OSC	OS Command	
IDA_PM	Private Msg	
IDA_APC	App Prog Cmd	
<b>Internal Actions (TE only)</b>		

<b>Symbolic Name</b>	<b>Friendly Name</b>	<b>Description</b>
IDA_UPDATE_CURSOR	Update Cursor	
IDA_INHIBIT_UPDATE	Inhibit Update	Don't update display
IDA_UNINHIBIT_UPDATE	Uninhibit Update	Allow display update
IDA_UPDATE	Update	Force display update
IDA_INHIBIT_SEND	Inhibit Send	VT buffer characters
IDA_UNINHIBIT_SEND	Uninhibit Send	VT stop buffering
IDA_SEND_PENDING	Send Pending Chars	VT send buffered chars
<b>Program Actions</b>		
IDA_PROGRAM_ABOUT	Program About	Display About dialog
IDA_PROGRAM_EXIT	Program Exit	Exit program
IDA_PROGRAM_EXITSILENT	Program Exit Silent	Exit program silently
IDA_PROGRAM_HELP	Program Help	Display Help
IDA_SUSPEND_DEVICE	Suspend Device	Enter suspend state
IDA_BLUETOOTH_DISCOVERY	Bluetooth Discovery	Start discovery
IDA_WARMBOOT	Warm Boot	Warm boot device
IDA_COLDBOOT	Cold Boot	Cold boot device
IDA_MENU_TOPBOTTOM	Menu Top/Bot	Toggle menu location
IDA_MENU_TOGGLEHIDE	Menu Toggle	Toggle menu visibility
IDA_TOOLBAR_TOGGLE	ToolBar Toggle	Toggle toolbar visibility
IDA_START_TOGGLEHIDE	Start Menu Toggle	Toggle Start visibility
IDA_MENUBAR_TOGGLEHIDE	MenuBar Toggle	Toggle menubar visibility
IDA_SESSION_TOGGLECON	Connect/Discon	Toggle session connection
IDA_SESSION_CONFIGURE	Configure	Configure session
IDA_SESSION_CONNECT	Connect	Connect session
IDA_SESSION_DISCONNECT	Disconnect	Disconnect session
IDA_SESSION_NEXT_LIVE	Next Live Session	Switch to next live session
IDA_SESSION_PASSWORD	Password	Session password dialog
IDA_SESSION_PREV	Prev Session	Switch to previous session
IDA_SESSION_NEXT	Next Session	Switch to next session
IDA_SESSION_DISCON_ALL	Disconnect All	Disonnect all sessions
IDA_SESSION_S1	Session 1	Switch to session 1

<b>Symbolic Name</b>	<b>Friendly Name</b>	<b>Description</b>
IDA_SESSION_S2	Session 2	Switch to session 2
IDA_SESSION_S3	Session 3	Switch to session 3
IDA_SESSION_S4	Session 4	Switch to session 4
IDA_TOOLBAND_HIDE	Hide ToolBar	Hide full Toolbar
IDA_TOOLBAND_TOGGLEHIDE	Toggle ToolBar	Toggle Toolbar visibility
IDA_KEYBAR_HIDE	Hide KeyBar	Hide KeyBar
IDA_KEYBAR_TOGGLEHIDE	KeyBar Toggle	Toggle KeyBar visibility
IDA_KEYBAR_LEFT	Prev KeyBar	Switch to previous KeyBar
IDA_KEYBAR_RIGHT	Next KeyBar	Switch to next KeyBar
IDA_KEYBAR_SEPARATOR	--Separator--	Separator for KeyBar
IDA_KEYBAR_NONE	(Empty)	No action placeholder
IDA_HSCROLL_HIDE	HScroll Hide	
IDA_HSCROLL_VISIBLE	HScroll Show	
IDA_HSCROLL_TOGGLEHIDE	HScroll Toggle	
IDA_HSCROLL_PLUSONE	HScroll Right One	
IDA_HSCROLL_MINUSONE	HScroll Left One	
IDA_HSCROLL_PLUSHALF	HScroll Right Page	
IDA_HSCROLL_MINUSHALF	HScroll Left Page	
IDA_HSCROLL_PLUSEND	HScroll Right End	
IDA_HSCROLL_MINUSEND	HScroll Left End	
IDA_VSCROLL_HIDE	VScroll Hide	
IDA_VSCROLL_VISIBLE	VScroll Show	
IDA_VSCROLL_TOGGLEHIDE	VScroll Toggle	
IDA_VSCROLL_PLUSONE	VScroll Up One	
IDA_VSCROLL_MINUSONE	VScroll Down One	
IDA_VSCROLL_PLUSHALF	VScroll Up Page	
IDA_VSCROLL_MINUSHALF	VScroll Down Page	
IDA_VSCROLL_PLUSEND	VScroll Up End	
IDA_VSCROLL_MINUSEND	VScroll Down End	
IDA_FONT_PLUS	Font Inc	Increase font size
IDA_FONT_MINUS	Font Dec	Decrease font size
IDA_TOGGLE_FONT_BOLD	Font Bold	
IDA_SMARTPAD_OPEN	SmartPad Show	
IDA_SMARTPAD_CLOSE	SmartPad Hide	
IDA_SMARTPAD_TOGGLEHIDE	SmartPad Toggle	

<b>Symbolic Name</b>	<b>Friendly Name</b>	<b>Description</b>
IDA_SLEEP_10	Sleep 10msec	
IDA_SLEEP_50	Sleep 50msec	
IDA_SLEEP_200	Sleep 200msec	
IDA_SLEEP_1000	Sleep 1sec	
IDA_SLEEP_5000	Sleep 5sec	
IDA_SLEEP_20000	Sleep 20sec	
IDA_SLEEP_100000	Sleep 100sec	
IDA_SCAN_TRIGGER	Scan Trigger	Soft trigger scanner
IDA_MACRO_OPEN	Macro Show	Show Macro Tool
IDA_MACRO_CLOSE	Macro Hide	Hide Macro Tool
IDA_MACRO_TOGGLEHIDE	Macro Toggle	Toggle Macro Tool hiding
IDA_MACRO_RECORD	Macro Record	Start Macro record
IDA_MACRO_STOP	Macro Stop	Stop Macro record
IDA_MACRO_PLAY	Macro Play	Replay Macro
IDA_PRINT_SCREEN	Print Screen	Print current screen
IDA_OIA_HIDE	OIA Hide	Hide IBM OIA bar
IDA_OIA_VISIBLE	OIA Show	Show IBM OIA bar
IDA_OIA_TOGGLEHIDE	OIA Toggle	Toggle OIA bar visibility
<b>General IBM and VT Actions</b>		
IDA_PF1	F1	(Not VT PF1)
IDA_PF2	F2	(Not VT PF2)
IDA_PF3	F3	(Not VT PF3)
IDA_PF4	F4	(Not VT PF4)
...	...	
IDA_PF24	F24	
IDA_HOME	Home	
IDA_DOWN	Down	
IDA_UP	Up	
IDA_LEFT	Left	
IDA_RIGHT	Right	
IDA_ENTER	Enter	
IBM Actions		

<b>Symbolic Name</b>	<b>Friendly Name</b>	<b>Description</b>
IDA_IBM_HOME	IBM Home	
IDA_DELETE	Delete	
IDA_INSERT_ON	Insert On	
IDA_INSERT_OFF	Insert Off	
IDA_INSERT_TOGGLE	Insert Toggle	
IDA_ATTN	Attn	
IDA_CLEAR	Clear	
IDA_CURSOR_SELECT	Cursor Select	
IDA_DUP	DUP	
IDA_ERASE_EOF	Erase EOF	
IDA_ERASE_INPUT	Erase Input	
IDA_FIELD_MARK	Field Mark	
IDA_NEWLINE	Newline	
IDA_PA1	PA1	
IDA_PA2	PA2	
IDA_PA3	PA3	
IDA_RESET	Reset	
IDA_SYSREQ	Sys Request	
<b>5250 Specific Actions</b>		
IDA_FIELD_EXIT	Field Exit	
IDA_FIELD_PLUS	Field +	
IDA_FIELD_MINUS	Field -	
IDA_FIELD_ADVANCE	Field Advance	
IDA_FIELD_BACKSPACE	Field Backspace	
IDA_FIELD_SUB	Field SUB	
IDA_HELP	IBM Help	
IDA_ROLL_DOWN	Roll Down	
IDA_ROLL_UP	Roll Up	
IDA_ROLL_LEFT	Roll Left	
IDA_ROLL_RIGHT	Roll Right	
IDA_BACKSPACE	Backspace	
IDA_PRINT	IBM Print	
<b>VT Actions</b>		
IDA_ANSWERBACK	Answerback	
IDA_FIND	Find	
IDA_INSERT_HERE	Insert Here	
IDA_NEXT	Next	
IDA_PREVIOUS	Previous	

<b>Symbolic Name</b>	<b>Friendly Name</b>	<b>Description</b>
IDA_REMOVE	Remove	
IDA_SELECT	Select	
IDA_VT_PF1	VT PF1	Numpad PF1 key
IDA_VT_PF2	VT PF2	Numpad PF2 key
IDA_VT_PF3	VT PF3	Numpad PF3 key
IDA_VT_PF4	VT PF4	Numpad PF4 key
IDA_VT_COMMA	Numpad Comma	
IDA_NUMPAD_0	Numpad 0	
IDA_NUMPAD_1	Numpad 1	
IDA_NUMPAD_2	Numpad 2	
IDA_NUMPAD_3	Numpad 3	
IDA_NUMPAD_4	Numpad 4	
IDA_NUMPAD_5	Numpad 5	
IDA_NUMPAD_6	Numpad 6	
IDA_NUMPAD_7	Numpad 7	
IDA_NUMPAD_8	Numpad 8	
IDA_NUMPAD_9	Numpad 9	
IDA_VT_ENTER	Numpad Enter	
IDA_VT_MINUS	Numpad Minus	
IDA_VT_PERIOD	Numpad Period	
IDA_UDK_F6	UDK F6	VT User Defined Key F6
IDA_UDK_F7	UDK F7	VT User Defined Key F7
...	...	
IDA_UDK_F20	UDK F20	VT User Defined Key F20
IDA_VT_HELP	VT Help	
IDA_VT_DO	VT Do	
IDA_ADD	Add	
IDA_MULTIPLY	Multiply	
IDA_DIVIDE	Divide	
<b>Custom VT Sequences</b>		
IDA_VT_SAP0135	VT SAP0135	0x00 0x35
IDA_VT_CSI_M	VT CSI M	ESC [ M
IDA_VT_CSI_N	VT CSI N	ESC [ N
IDA_VT_CSI_O	VT CSI O	
IDA_VT_CSI_P	VT CSI P	

<b>Symbolic Name</b>	<b>Friendly Name</b>	<b>Description</b>
IDA_VT_CSI_Q	VT CSI Q	
IDA_VT_CSI_R	VT CSI R	
IDA_VT_CSI_S	VT CSI S	
IDA_VT_CSI_T	VT CSI T	
<b>Windows App Keys</b>		
IDA_APPKEY_K1	App Key 1	
IDA_APPKEY_K2	App Key 2	
...	...	
IDA_APPKEY_K16	App Key 16	
IDA_SCROLL_UPPERLEFT	Scroll Upper Left	
IDA_SCROLL_UPPERRGHT	Scroll Upper Right	
IDA_SCROLL_LOWERLEFT	Scroll Lower Left	
IDA_SCROLL_LOWERRGHT	Scroll Lower Right	
IDA_SCROLL_CENTER	Scroll Center	
IDA_SCROLL_CURSOR_CENTER	Scroll Cursor Center	
IDA_SCROLL_CURSOR_VISIBLE	Scroll Cursor Visible	
IDA_COPYALL	Copy All	Copy screen to clipboard
IDA_PASTE	Paste	Past clipboard
IDA_USTRING_0	Text 1	Send user text 1
IDA_USTRING_1	Text 2	Send user text 2
...	...	
IDA_USTRING_63	Text 64	Send user text 64
IDA_SCRIPT_1	Script 1	Run Script 1
IDA_SCRIPT_2	Script 2	Run Script 2
...	...	...
IDA_SCRIPT_64	Script 64	Run Script 64
IDA_SIP_HIDE	SIP Hide	
IDA_SIP_SHOW	SIP Show	
IDA_SIP_TOGGLEHIDE	SIP Toggle	
IDA_SIP_LOCKDOWN	SIP Lockdown	
IDA_SIP_UNLOCK	SIP Unlock	
IDA_SIP_UP	SIP Up	
IDA_SIP_DOWN	SIP Down	
IDA_SIP_FORCEDOWN	SIP Forcdown	



<b>Symbolic Name</b>	<b>Friendly Name</b>	<b>Description</b>
IDA_IM_KEYBOARD	IM Keyboard	
IDA_IM_LOCKED	IM Locked	
<b>HTML Actions</b>		
IDA_URL_HOME	URL Home	
IDA_URL_BACK	URL Back	
IDA_URL	URL	Defines start of URL
Special Actions		
IDA_VIBRATE_100	Vibrate 100ms	
IDA_VIBRATE_200	Vibrate 200ms	
IDA_VIBRATE_500	Vibrate 500ms	
IDA_VIBRATE_1000	Vibrate 1sec	
IDA_VIBRATE_2000	Vibrate 2sec	
IDA_VIBRATE_5000	Vibrate 5sec	
IDA_BEEP_OK	Beep	
IDA_BEEP_WARN	Beep Warn	
IDA_BEEP_LOUD	Beep Loud	
IDA_KBD_ALPHA	KeyMode Alpha	
IDA_KBD_NUMERIC	KeyMode Numeric	
IDA_KBD_ALPHANUM	KeyMode AlphaNum	
IDA_KBD_UPPERALPHA	KeyMode Upper Alpha	
IDA_KBD_LOWERALPHA	Keymode Lower Alpha	
IDA_KBD_FUNCMODE	KeyMode Func	
IDA_KBD_CYCLEMODE	KeyMode Cycle	Cycle to next mode
IDA_POPUP_IPADDRESS	Show IP Address	
IDA_POPUP_MACADDRESS	Show MAC Address	
IDA_POPUP_BATTERY	Show Battery	
IDA_POPUP_TIME	Show Time	
IDA_POPUP_SERIALNUMBER	Show Serial #	
IDA_POPUP_DEVICEID	Show Device ID	
IDA_POPUP_RFINFO	Show RF info	

## Appendix 3 - Virtual Key Codes

This appendix contains a list of Windows CE Virtual Key Codes (VK) which are used with the OnKey META tag.

Notice that there is no case distinction of the alphabetic keys. Also, note that the symbols on the tops of the digit keys are not listed because they are a shift state of the digit keys.

<b>Symbolic Name</b>	<b>Hexadecimal Value</b>	<b>Keyboard Equivalent</b>
VK_BACK	08	BACKSPACE key
VK_TAB	09	TAB key
VK_CLEAR	0C	CLEAR key
VK_RETURN	0D	ENTER key
VK_SHIFT	10	SHIFT key
VK_CONTROL	11	CTRL key
VK_MENU	12	ALT key
VK_PAUSE	13	PAUSE key
VK_CAPITAL	14	CAPS LOCK key
VK_ESCAPE	1B	ESC key
VK_SPACE	20	SPACEBAR
VK_PRIOR	21	PAGE UP key
VK_NEXT	22	PAGE DOWN key
VK_END	23	END key
VK_HOME	24	HOME key
VK_LEFT	25	LEFT ARROW key
VK_UP	26	UP ARROW key
VK_RIGHT	27	RIGHT ARROW key
VK_DOWN	28	DOWN ARROW key
VK_SELECT	29	SELECT key
VK_EXECUTE	2B	EXECUTE key
VK_SNAPSHOT	2C	PRINT SCREEN key
VK_INSERT	2D	INS key
VK_DELETE	2E	DEL key
VK_HELP	2F	HELP key
VK_0	30	0 key
VK_1	31	1 key

<b>Symbolic Name</b>	<b>Hexadecimal Value</b>	<b>Keyboard Equivalent</b>
VK_2	32	2 key
VK_3	33	3 key
VK_4	34	4 key
VK_5	35	5 key
VK_6	36	6 key
VK_7	37	7 key
VK_8	38	8 key
VK_9	39	9 key
VK_A	41	A key
VK_B	42	B key
VK_C	43	C key
VK_D	44	D key
VK_E	45	E key
VK_F	46	F key
VK_G	47	G key
VK_H	48	H key
VK_I	49	I key
VK_J	4A	J key
VK_K	4B	K key
VK_L	4C	L key
VK_M	4D	M key
VK_N	4E	N key
VK_O	4F	O key
VK_P	50	P key
VK_Q	51	Q key
VK_R	52	R key
VK_S	53	S key
VK_T	54	T key
VK_U	55	U key
VK_V	56	V key
VK_W	57	W key
VK_X	58	X key
VK_Y	59	Y key

<b>Symbolic Name</b>	<b>Hexadecimal Value</b>	<b>Keyboard Equivalent</b>
VK_Z	5A	Z key
VK_NUMPAD0	60	Numeric keypad 0 key
VK_NUMPAD1	61	Numeric keypad 1 key
VK_NUMPAD2	62	Numeric keypad 2 key
VK_NUMPAD3	63	Numeric keypad 3 key
VK_NUMPAD4	64	Numeric keypad 4 key
VK_NUMPAD5	65	Numeric keypad 5 key
VK_NUMPAD6	66	Numeric keypad 6 key
VK_NUMPAD7	67	Numeric keypad 7 key
VK_NUMPAD8	68	Numeric keypad 8 key
VK_NUMPAD9	69	Numeric keypad 9 key
VK_MULTIPLY	6A	Multiply key
VK_ADD	6B	Add key
VK_SEPARATOR	6C	Separator key
VK_SUBTRACT	6D	Subtract key
VK_DECIMAL	6E	Decimal key
VK_DIVIDE	6F	Divide key
VK_F1	70	F1 key
VK_F2	71	F2 key
VK_F3	72	F3 key
VK_F4	73	F4 key
VK_F5	74	F5 key
VK_F6	75	F6 key
VK_F7	76	F7 key
VK_F8	77	F8 key
VK_F9	78	F9 key
VK_F10	79	F10 key
VK_F11	7A	F11 key
VK_F12	7B	F12 key
VK_F13	7C	F13 key
VK_F14	7D	F14 key
VK_F15	7E	F15 key

<b>Symbolic Name</b>	<b>Hexadecimal Value</b>	<b>Keyboard Equivalent</b>
VK_F16	7F	F16 key
VK_F17	80	F17 key
VK_F18	81	F18 key
VK_F19	82	F19 key
VK_F20	83	F20 key
VK_F21	84	F21 key
VK_F22	85	F22 key
VK_F23	86	F23 key
VK_F24	87	F24 key

## Glossary

### **CEBrowseX**

A Naurtech ActiveX control which provides access to the Naurtech Web Browser configuration and actions.

### **external**

This is the name of an implied object in the DOM of the Windows CE .NET and CE 5.0 browser that gives access to special non-standard features. See the CEBrowseX documentation for details.

### **GetProperty**

A method available via the CEBrowseX ActiveX object or the CE .NET “external” object to get a configuration value from the Web Browser.

### **IDA Action Code**

An IDA Action Code defines a special device, application, or emulation action within the Naurtech Smart Clients. IDA codes can be tied to keys, or KeyBars, and invoked via META tags or JavaScript. See the Appendix for a list of values.

### **PostIDA**

A method available via the CEBrowseX ActiveX object or the CE .NET “external” object to post an IDA Action Code to the Web Browser.

### **SendIDA**

A method available via the CEBrowseX ActiveX object or the CE .NET “external” object to send an IDA Action Code to the Web Browser. The SendIDA is a synchronous activation of the action.

### **SetProperty**

A method available via the CEBrowseX ActiveX object or the CE .NET “external” object to set a configuration value in the Web Browser.

### **TextX**

A Naurtech ActiveX control that is used with Pocket PC or Windows Mobile 2003 platforms to provide a fully featured text input element which supports event handlers.

## Index

---

### *C*

CEBrowseX CLASSID · 53  
CEBrowseX Control · 53

---

### *E*

external object · 53

---

### *G*

GetProperty · 54, 62

---

### *H*

hidden fields · 45

---

### *I*

iBrowse META tags · 16  
IDA Action Codes · 11, 63

---

### *M*

#### **META Tag Identifiers** · 16

Application · 17  
Battery · 18  
BatteryNavigate · 19  
Command · 20  
CursorPos · 21  
ErrorNavigate · 21  
GetUnitInformation · 22  
HomeKey · 23  
IDA · 23  
MoveSIP · 24  
OnAllKeys · 25  
OnKey · 26  
PLSeriesLabel\_Complete · 38  
PLSeriesLabel\_Print · 39  
PowerOn · 28  
Reboot · 29

Scanner · 29  
ScannerNavigate · 30  
SetDate · 32  
SetTime · 32  
Signal · 33  
SignalNavigate · 34  
SIP · 35  
SIPUp · 36  
TextSize · 36  
TimerInterval · 37  
TimerNavigate · 37  
ZebraLabel\_Complete · 38  
ZebraLabel\_Print · 39

---

### *O*

OnChange · 60  
OnClick · 60  
OnFocus · 60  
OnKeyPress · 61  
OnKeyUp · 61  
OnLostFocus · 60

---

### *P*

PlaySound · 55  
PlayTone · 55  
PostIDA · 54  
Print · 55  
PrintString · 55

---

### *S*

SendIDA · 54  
SetFocus · 59  
SetProperty · 55, 62  
ShowSIP · 59  
stiscan · 50  
stisubmit · 50

---

### *T*

TextX CLASSID · 58  
TextX Control · 57