Professional Validation And More
What's New In Version 3.0

Introduction

Professional Validation And More v3.0 is an extensive upgrade. This document describes the changes in detail. It is also a change in the licensing model. If you are upgrading, you should know the following:

- If you had Level 1, you have licenses to a subset of the complete product. Your licenses are for the modules Essential Validators and Client-Side Toolkit. You will see the rest of the Professional Validation And More feature set but you will not be able to use them at runtime.
- If you have Level 2, you have the complete feature set to Professional Validation And More (call the "Suite"). It includes the Visual Input Security™ product.

The documentation has been overhauled. There are separate User's Guides for Validation, Data Entry Controls, and Client-Side Toolkit features. Notable enhancements include the descriptions of validators, how to use the CustomValidator, and step-by-step usage for each type of control whether using design mode, ASP.NET text, or program code.

Features that enhance existing sites are marked ▲. This means that after installing you either immediately benefit from them or will benefit after a quick property change. Features without this marker may require adding controls or changing the design of your page.

Quick List: New Controls

These controls are documented in detail later: DuplicateEntryValidator, SelectedIndexRangesValidator, UnwantedWordsValidator, TextBox, MultiSegmentDataEntry, MultiSegmentDataEntryValidator, CalculationController, FSCOnCommand, and MultiFSCOnCommand.

Validation Enhancements

These features are documented in the **Validation User's Guide**. Page references are given here.

New Validators

Note: All of these Validators require a license for the "Suite" or "Special Validators" module.

- The **DuplicateEntryValidator** evaluates three or more data entry controls to determine if there are any duplicate entries. You can compare textual values in any of the controls supported: TextBox, ListBox, and DropDownList. For ListBox and DropDownList, you can compare either the textual value or the **SelectedIndex**. Page 63.
- The **UnwantedWordsValidator** compares the contents of a textbox to a list of strings. If it matches one of the strings, the condition fails. It is often used for blocking curse words. It has rules to match "whole words", case insensitive, and "broken words". A broken word is a word that is separated by non-letter characters in an attempt to avoid detection. For example, if the word "flower" is unwanted, an error will be reported on "f*l*o*w*e*r". Page 67.
- The **SelectedIndexRangesValidator** evaluates the **SelectedIndex** property on ListBoxes, DropDownLists, and RadioButtonLists. It is an extension of the idea behind the SelectedIndexValidator where you indicate a **SelectedIndex** that's valid. In this Validator, you provide one or more ranges that the **SelectedIndex** can match to. For example, if you have a dropdownlist of 20 items and you want items 1, 10-12, and 15-20 to be valid, this is the right Validator. Page 71.

Enhancements to existing Validators and Conditions

- Validators and Condition objects that handle textual data now can evaluate hidden input fields. The hidden input field must be an HtmlInput Hidden class or be defined as `<input type='hidden' runat=server id='id' >`.
- ▲Validators and Conditions that use regular expressions can now cache them and call an event handler to set them up only once. This provides speed optimization the Regex object takes CPU time when its created.

  The **GlobalStorageName** property lets you define a name to cache the Regex object in the Application collection. If you use the same expression in several Validators, assigning the same name to **GlobalStorageName** will allow them to share the Regex object, further optimizing speed.

  The **PrepareProperties** delegate should be used when you do extensive work to develop the information associated with the regular expression, such as querying a database to get a list of words used by the CompareToStringsValidator (a regex user). If that query never changes, **PrepareProperties** assures you that it only runs once.

These features are found on RegexValidator (page 42), CharacterValidator (page 53), CompareToStringsValidator (page 55), UnwantedWordsValidator (page 67), and EmailAddressValidator. (Plus the Condition objects for those Validators.) EmailAddressValidator automatically starts using it because it has a fixed internal expression.

- The new **IgnoreBlankText** property on the TextLengthValidator (page 47) and WordCountValidator (page 59) lets you ignore blank fields even when you establish a Minimum value. The Minimum will only be used when at least one character is entered. When the textbox does not require text but needs a minimum, use it.

Error Formatting

- The new **NoErrorFormatter** property on a Validator lets you show text and/or an image when there is no error. Page 126. It has many modes to define when it appears. The modes include:
  - Always show when there is no error. It will appear as the page is first displayed. It will be removed as an error is revealed. It will return after the error is fixed.
  - Show after the user fixes the error. It appears after the Validator displays the error and the user corrects it.
  - Show after the field has been validated and has no errors. The user does not have to edit to cause this. For example, if a field requires text and opens with text in the textbox, when the user clicks Submit, the RequiredTextValidator will validate and the NoErrorFormatter appears.
  - Show only on a new page, prior to any edits. It is hidden after validation occurs. Never show on a post back page because validation has already occurred. It lets you attract users to fields that have not been edited yet.
- ▲ The **ImageURL** property on AlertImageErrorFormatter and TooltipImageErrorFormatter now defaults to a global setting that you find in the Global Settings Editor in the **DefaultImageErrorFormatterImageURL** property. This allows you to change the image across the site quickly. Page 124.
- ▲ The **ErrorMessage** and **SummaryErrorMessage** properties support a new token, {NEWLINE}. It is replaced by the appropriate character(s) for creating a linebreak, whether the error message appears on the web form, in an alert, or in a tooltip. Page 110.

ValidationSummary Control

- ▲ There is a new **DisplayMode** called `Table`. It lets you define alternating style sheets for each error message by putting error messages into individual table rows. You define the alternative style sheet in the new **TableAltRowCssClass** property. Page 206.
- ▲ The new **ListRowSeparator** property is used when **DisplayMode** is `List` to add a separator between each row. You can use HTML to define the appearance of that separator, such as an image. Page 207.
- ▲ ValidationSummary control that is already shown will hide if the user submits with a different validation group. This lets the user see only the error messages to the controls they are attempting to save.
- ▲ Bug fix: ValidationSummary with a specific validation group would show error messages from other groups at certain times. It will now show only the error messages for its own group.

Other Enhancements

- Validation groups has been expanded in two ways:
  - The Validator's **Group** property now permits a pipe delimited list of group names. This way, a Validator can be included in several groups. Page 137.
  - VAM controls containing within a DataGrid or Repeater can automatically create unique group names for each row. Add the plus (+) character before the group name in each **Group** property. Page 137.
- ▲ The VAM Submit controls and `PeterBlum.VAM.Globals.Page.RegisterSubmitControl()` method now support a confirm message string that overrides the confirm message on **PeterBlum.VAM.Globals.Page.ConfirmMessage**. This allows more specialized messages. The VAM Submit controls have the new **ConfirmMessage** property. Page 152.

- ▲ There is more client-side support for Validators that only evaluate on the server side. It especially helps CustomValidators where you did not write a client-side evaluation function and IgnoreConditionValidators. But also when a Validator has **EnableClientScript**=`false`.

The features are:

  - Server-side only Validators show their error message in the client-side generated ValidationSummary after a post back.
  - **PeterBlum.VAM.Globals.Page.ShowAlertOnSubmit** has a "Delayed" alert mode. After client-side validation approves everything, the page submits. If there are errors found by server-side only Validators, when the page is redrawn, the Alert is immediately show with the additional errors. Later, if there are any client-side errors shown in the alert, it will continue to reflect these server-side only errors until all client-side errors are fixed.
  - Server-side only Validators provide hyperlinks on error messages in the ValidationSummary when its **HyperLinkToFields** property is used.
  - When using **PeterBlum.VAM.Globals.FocusOnSubmit**, VAM sets focus after post back to the data entry control associated with the first server-side only Validator with an error.
  - IgnoreConditionValidator is a server-side only control that now supports these features when you use the new **RelatedControlID** property to identify the control that your validator evaluates. Page 94.
    - **PeterBlum.VAM.Globals.FocusOnSubmit**, after post back
    - **PeterBlum.VAM.Globals.FocusAfterAlert**
    - **PeterBlum.VAM.Globals.ControlErrorCssClass**
    - **ValidationSummary.HyperLinkToFields**

- The **PeterBlum.VAM.Globals.Page.ControlErrorCssClass** has three new companion properties. **ListErrorCssClass** and **CheckBoxErrorCssClass** let you have different style sheet class names for ListBoxes, DropDownLists, CheckBoxes and RadioButtons. This allows different appearances for control types. For example, you might like the background of a textbox to be changed while the font is changed on a listbox.

CheckBoxes and radiobuttons have two parts, the input field and the label. Use the **CheckBoxECCMode** property to determine which of them, or both, get the style sheet class.

Page 131.

▲ You can quickly update your site by creating your new style sheets and editing these properties in the **Global Settings Editor**.

- Third party controls that do not map the **ClientID** of the control to the data entry HTML tag can now support client-side validation. The custom.vam.config file's `<ThirdPartyControls>` section supports new properties to define the correct ID to the data entry HTML tag and to allow custom JavaScript to retrieve data. See "Supporting Third Party Data Entry Controls" in the **Installation Guide**.
- The **PeterBlum.VAM.Globals.Page.Validators** property provides a list of Validator controls defined on the page. Use it if you need to interate through these controls. Page 227.
- The `PeterBlum.VAM.Globals.Page.EnableValidators()` method lets you change the **Enabled** property for a list of validators based on a validation group name. Page 227.
- The `PeterBlum.VAM.Globals.Page.EnableClientScriptValidators()` method lets you change the **EnableClientScript** property for a list of validators based on a validation group name. Page 227.
- Many features of validators depend on the **Page.IsPostBack** property. For example, ValidationSummary doesn't appear until IsPostBack is true. Now you can use **PeterBlum.VAM.Globals.Page.IsPostBack** to coerce VAM to thinking the **IsPostBack** state is different. Perhaps you want to redirect to a page and immediately show validator errors along with the ValidationSummary. Page 226.

Data Entry Controls

These features are documented in the **Data Entry Controls User's Guide**. Page references are given here.

Note: All of these features require a license for the "Suite" or "Data Entry Controls" module.

New Data Entry Controls

- TextBox – This direct subclass of System.Web.UI.WebControls.TextBox introduces a number of common client-side enhancements. Consider using it everywhere you already use the TextBox control. See page 15. Its enhancements include:
  - Smarter detection of "onchange" events. The client-side onchange event is a signal that the textbox has changed after focus leaves the field. Validator controls and the TextBoxes that autoreformat rely on the onchange event. However, browsers have two limitations that prevent them from firing the onchange event, even after a change is made:
    - The Autocomplete feature on Internet Explorer does not cause it to fire. So users who pick from the Autocomplete list do not get the immediate feedback of validation.
    - When JavaScript assigns a new value to the textbox, it does not cause it to fire.

    `PeterBlum.VAM.TextBox` installs code that always fires the onchange event when focus is lost and a change occurred.
  - The **ValueWhenBlank** property lets you show text when the TextBox is blank. For example, "Fill this in". The **ValueWhenBlankCssClass** property lets you change the style sheet on a blank TextBox. These two properties have a further rule, on **PeterBlum.VAM.Globals.Page.ValueWhenBlankMode**, that determines what happens when focus enters the textbox: does the text clear out and or the style sheet switch to its normal value?
  - The **DisableAutoComplete** property lets you omit the AutoComplete list from the textbox. There are cases where Autocomplete does not belong. Supported by several browsers.
  - The **DisablePaste** property which turns off the ability to paste on Internet Explorer and any other browser that supports the client-side onpaste event.
  - The AutoPostBack feature now supports client-side validation of the textbox before submitting the page when using the **AutoPostBackValidates** property. This way, it only submits if the changed textbox is also valid.
  - Auto tabbing to another field in three ways:
    - Use the **TabAtMaxLength** property to tab when the text fills to the **MaxLength**.
    - Use the **TabOnEnterKey** property to tab when the user types ENTER.
    - Use the **TabOnTheseChars** to define a list of characters that tab when any are typed.
  - Use the **EnterSubmitsControlID** property to specify a button that will be clicked when ENTER is typed. *This feature requires a license for the Client Side Toolkit.*
  - Use the **Hint** property to display a hint elsewhere on the page as the user tabs into the field. The hint is removed as tab departs the field. *This feature requires a license for the Client Side Toolkit.*
  - The **TextAlign** property lets you choose between left, center, and right alignment.
  - The **ToolTip** supports the String Lookup System through its **ToolTipLookupID** property. This provides localization on the ToolTip.
  - The Properties Editor provides a window where you can enter lengthy text more easily in the **Text** and **ToolTip** properties.
- MultiSegmentDataEntry is a substitute for a TextBox when you have a strongly patterned data type. It is a similar idea to a masked textbox, where each character position requires a specific character. For example, this control and masked textboxes are used to enter phone numbers, IP addresses, and dates (although PeterBlum.com's **Peter's Date Package** provides much better date entry.) Its user interface provides a more interactive and understandable interface than masked textboxes. It also works with more browsers than masked textboxes due to limitations of DHTML. See page 55.

- MultiSegmentDataEntryValidator is a validator for the MultiSegmentDataEntry control to confirm the text matches the rules you defined. Page 87.

Numeric TextBox Enhancements

Numeric TextBoxes are IntegerTextBox, DecimalTextBox, and CurrencyTextBox.

- The Numeric TextBoxes all subclass from VAM's new TextBox, giving them all of the features described above.
- ▲ DecimalTextBox has the new **MaxDecimalPlaces** property. Use it to report an error when the user exceeds the number of digits defined with this property. The DataTypeCheckValidator will report the error. Page 40.
- ▲ The Spinner feature now has minimum and maximum values. When you set **SpinnerMinValue** and **SpinnerMaxValue**, the spinners are constrained to that range. (This does not affect validation. You still need to setup a RangeValidator to report range errors.) Page 33.
- ▲ IntegerTextBox has the new **FillLeadZeros** property, used to draw extra lead zeros to fill up the textbox. They only affect appearance, not the numeric value. Page 32.
- The **IntegerTextBox.IntegerValue** property can be assigned the value Int32.MinValue to clear the field. It is an alternative to setting **Text**="". Page 32.
- The **DecimalTextBox.DoubleValue** and **CurrencyTextBox.DoubleValue** properties can be assigned the value Double.MinValue to clear the field. It is an alternative to setting **Text**="". Pages 39 and 46.

Client-Side Toolkit

These features are documented in the **Client-Side Toolkit User's Guide**. Page references are given here.

Note: All of these features require a license for the "Suite" or "Client-Side Toolkit" module.

New Controls

- Use the CalculationController to create mathematical expressions involving textboxes on the page. It can show the result of the calculation in a label or textbox, which is updated as the user edits the textboxes. Its result can be used by Validators and Conditions. Page 62.
- FSCOnCommand is a variation of the FieldStateController that is designed to be fired by a button. While the FieldStateController selects from two fields states in ConditionTrue and ConditionFalse, FSCOnCommand only applies one field state. It supports the same state settings as the FieldStateController. A good example is to add a Select All button to mark all checkboxes in a CheckBoxList. Page 43.
- MultiFSCOnCommand is like the FSCOnCommand except it updates more than one control's state settings. Page 43.

Enhancements to FieldStateControllers

- The state settings contain the new **Checked** property. Use it to change the state of checkboxes and radiobuttons. When the **ControlIDToChange** is a CheckBoxList, it can mark or unmark all of them. Page 35.
- You can use the state settings **InnerHTML**, **CssClass**, and **FieldValue** to set their associated values to an empty string. Previously, the empty string indicated to use the original value of the field. For example, clear the className of a tag. Page 35.
- When you set the new **RunOnPageLoad** property to `false`, it no longer initializes the field state as the page loads. It only sets it as you make a change to the control being monitored. Page 32.
- ▲ When you set the new **ValidateChangedControls** property to true, it fires the Validators associated with the control identified by the **ControlIDToChange** or **ControlConnections** properties. Use it instead of the **UseValidateGroup** property to validate fewer controls. Page 37.

Other Enhancements

- Interactive Hints lets you show a description of the data entry control currently with focus. It's a label and optionally containing panel that is shown and hidden and focus is moved. This allows all of the controls to share a common location for their descriptions, usually giving room for much more details than when each control always shows its own hint. The

feature extends the all of VAM's TextBoxes and MultiSegmentDataEntry control. Plus you can call a method to attach hints to any other control that supports focus. Page 93.

- Direct keystrokes to click buttons. You can easily setup a control to click another control (like a button) when the user types a specific keystroke. For example, ENTER clicks a particular submit button on a page with several submit buttons. The feature extends VAM's TextBoxes and MultiSegmentDataEntry control with the **EnterSubmitsControlID** property. Page 103.

Other Features

- VAM's controls can work inside AJAX frameworks, so that they can be added, updated, or removed by a callback. Page 233 of the **Validation User's Guide**.
- ▲ VAM can now detect that JavaScript is not available (either disabled or non-existant). It initially does setup for JavaScript support but after the first post back, it knows the browser's support of JavaScript is off and sets up further pages wthout JavaScript support. You can see its state and even override it with the **PeterBlum.VAM.Globals.Page.JavaScriptEnabled** property. You can disable this feature with the **PeterBlum.VAM.Globals.Page.DetectJavaScript** property. Page 226 of the **Validation User's Guide**.
- It's easier to setup the Disable Button on Submit feature. You can now set the **DisableOnSubmit** property directly on any of VAM's submit controls. Page 145 of the **Validation User's Guide**.
- Use the new `PeterBlum.VAM.Globals.Page.AttachCodeToEvent()` method to connect your own JavaScript to a control that is already using JavaScript on an event that you want. For example, the VAM TextBox control uses the onfocus event. If you want to use that same event, use this method to safely join the code together. Page 95 of the **Data Entry Controls User's Guide**.