

Trinity Wizard Programmers Guide

Abstract

The WebWizard is a standard functionality within the Trinity Operating system. Regardless of which Trinity-Device is being used, the WebWizard is a helpful tool to reduce time for installs and it helps to simplify the configuration efforts for deployments.

On the following Link, more information about Patton devices running Trinity can be found.

- [Trinity Patton Products](#)

For WebWizards to be added to Trinity devices, the WebWizard Community Platform offers several Wizards to be downloaded. Also, this Platform offers the WebWizards to be executed online on:

- WebWizard Community Platform
<https://www.patton.com/wizard>

Contents

1	Introduction	3
2	Wizard handling.....	4
2.1	Accessing the WEB Wizard	4
2.2	Executing a WEB Wizard.....	4
2.3	Adding new WEB Wizard files	5
2.4	Structure of Wizard files.....	5
2.5	Examples	5
2.6	WebWizard Community Platform.....	6
2.7	Wizard Programming Services	6
3	Trinity WEB Wizard Creation.....	7
3.1	Sections overview	7
3.2	XML Schema Reference	10
3.2.1	Config-Generator	11
3.2.2	Overview of available Elements	12
3.2.3	Gui	12
3.2.4	Common Page Element Attributes	13
3.2.5	Config-Snippet	20
3.2.6	Config.....	21

1 Introduction

The Patton Trinity WEB Wizard is the perfect tool for integrators, technicians and network engineers who would like to create their own WEB interface for Trinity products.

Wherever an application specific setup is required, where only a few parameters differ from one customer installation to another, the WEB wizard will help to significantly reduce the installation time.



Trinity WEB Wizards are simple text files that contain a template configuration and an XML description of the Graphical User interface that is presented to the installer.

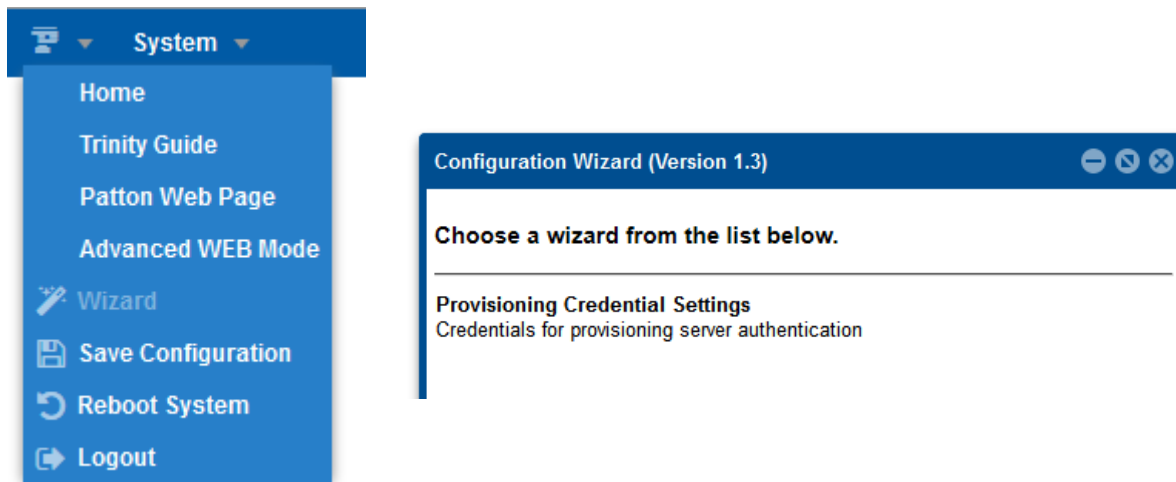
The Template configuration contains all the application specific settings which are static while the WEB GUI gives simple access to settings which are customer or installation specific.

This document describes how you can create, install and use WEB Wizards on Patton Trinity products.

2 Wizard handling

2.1 Accessing the WEB Wizard

There are one or more WEB Wizards on every unit running Trinity. These WEB wizards can be found under the  Menu or by clicking on the magic wand  in the top right corner of the Trinity GUI.



Clicking on the Wizard Menu item will open a window containing all the pre-loaded wizards. In the example above, just the Digest Authentication Wizard for HTTPs provisioning is visible “Provisioning Credential Settings”.

2.2 Executing a WEB Wizard

Execute a Wizard simply by clicking on an entry in the list. The Wizard will open in a new window. There the user will have to enter the configuration parameters based on the available fields. A wizard may have multiple pages which are accessible by clicking on the “next” button at the bottom of the Wizard window. Once all parameters are set, the user can

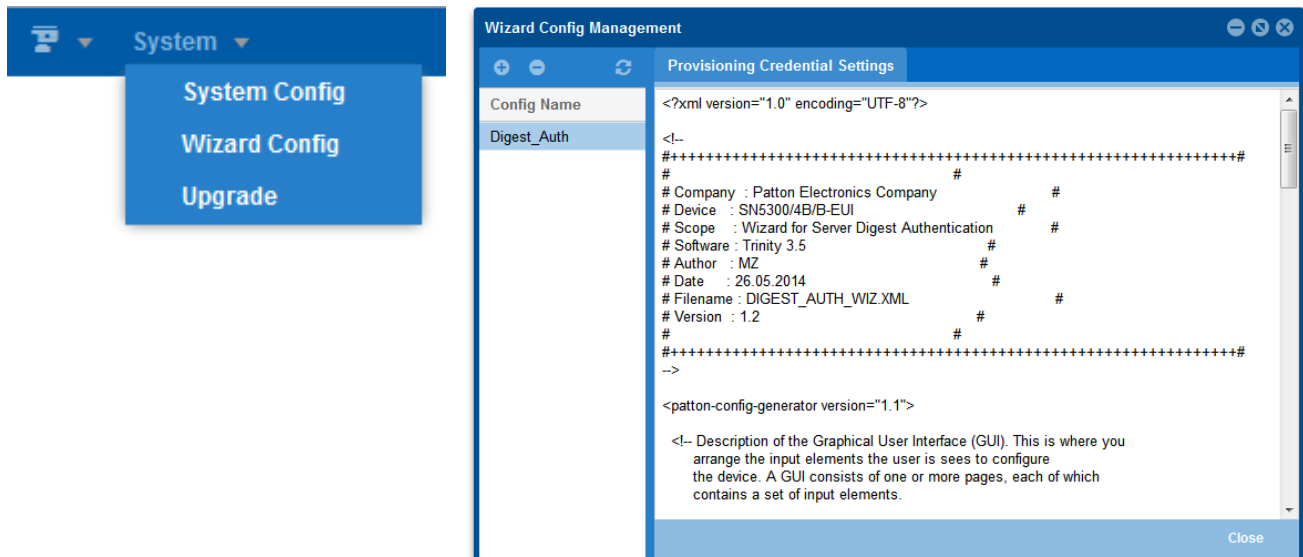
- 1) Verify the generated CLI config, by hitting the “Preview” button
- 2) Apply the generated config to the unit by hitting the “Save & Reboot” button.

Note: a reboot of the device will be executed in order for the config to become active.

2.3 Adding new WEB Wizard files

Under the System Menu, the Wizard config can be accessed. Here new Wizards can be imported or existing ones can be deleted.

All the Wizard files are XML files which must have a filename ending with .xml



In order to add a new Wizard file, just click on the “+” button, select your .xml wizard file and hit OK.

A preview of the uploaded Wizard file can be seen on the right section of the Wizard Config Management window.

2.4 Structure of Wizard files

A Wizard XML file consists of 3 sections.

- 1) WEB Wizard GUI elements description using XML
- 2) The config-snippet element defines conditional variables that can be included inside the final system configuration file. controlled by item 1)
- 3) The config element defines the actual system configuration file that will be uploaded to the device

In the following Chapters, a detailed overview is given on how a WEB Wizard can be created. It includes a description of the GUI fields available and how they turn in to CLI config values, by generating a config file.

2.5 Examples

For example files of WEB Wizards refer to the following sources:

- Any Patton Trinity device contains at least one Wizard files you may use as an example
- The WebWizard platform, contains Wizards from the [Patton Wizard Community](https://www.patton.com/wizard).
<https://www.patton.com/wizard>
- Contact us: support@patton.com

2.6 WebWizard Community Platform

The [WebWizard](#) Platform is for Carriers, System integrators and End-Users, to speed up Patton device installations.

Join the [WebWizard](#) Platform today, and benefit discover the two community memberships

- **WebWizard Users**

Benefits

- Save time when configuring your Patton device
- Speed up your installations
- Profit from know-how of more than 1`000 certified specialists from all over the world

Permissions

- Users can download, execute and comment on Wizards that are online

- **WebWizard Creators**

Benefits

- Support your customers with ready-to-user customized configurations
- Speed up your installations
- Raise community awareness of your company
- Direct interactions with other market players

Permissions

- Creators can create, upload and download Wizards
- They are notified by email when users add comments
- Creators are expected to reply to all user comments about their Wizards

The usage of the Platform is free of charge!

2.7 Wizard Programming Services

If you would like Patton to program a Wizard for your application contact our sales team we will give you a professional services offer based on your requirement specification.

- [Contact us for a Quote](#)

3 Trinity WEB Wizard Creation

This chapter describes the format of the Wizard XML Configuration file. Each wizard configuration consists of three main sections. The series of images below progress through each section of the config file. The first provides an overview of the whole config file.

3.1 Sections overview

gui

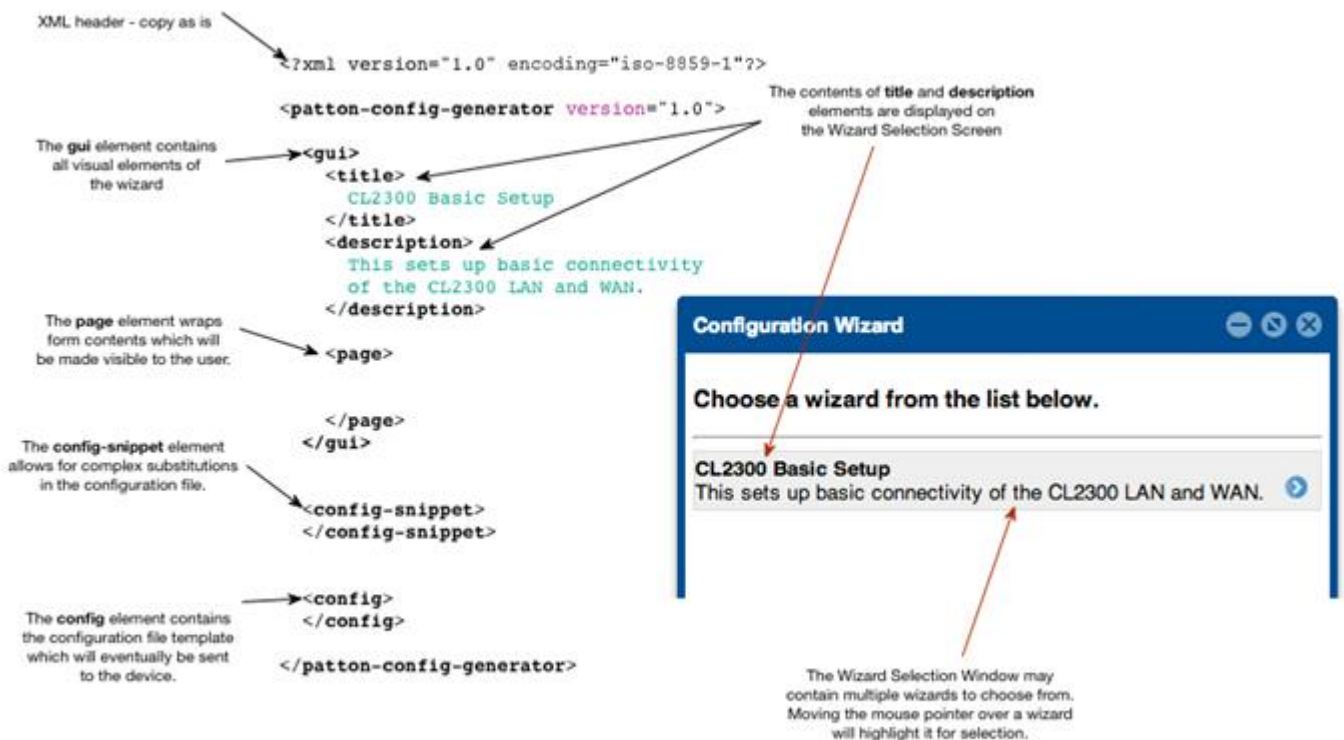
The `gui` element wraps the form definition for the user.

config-snippet

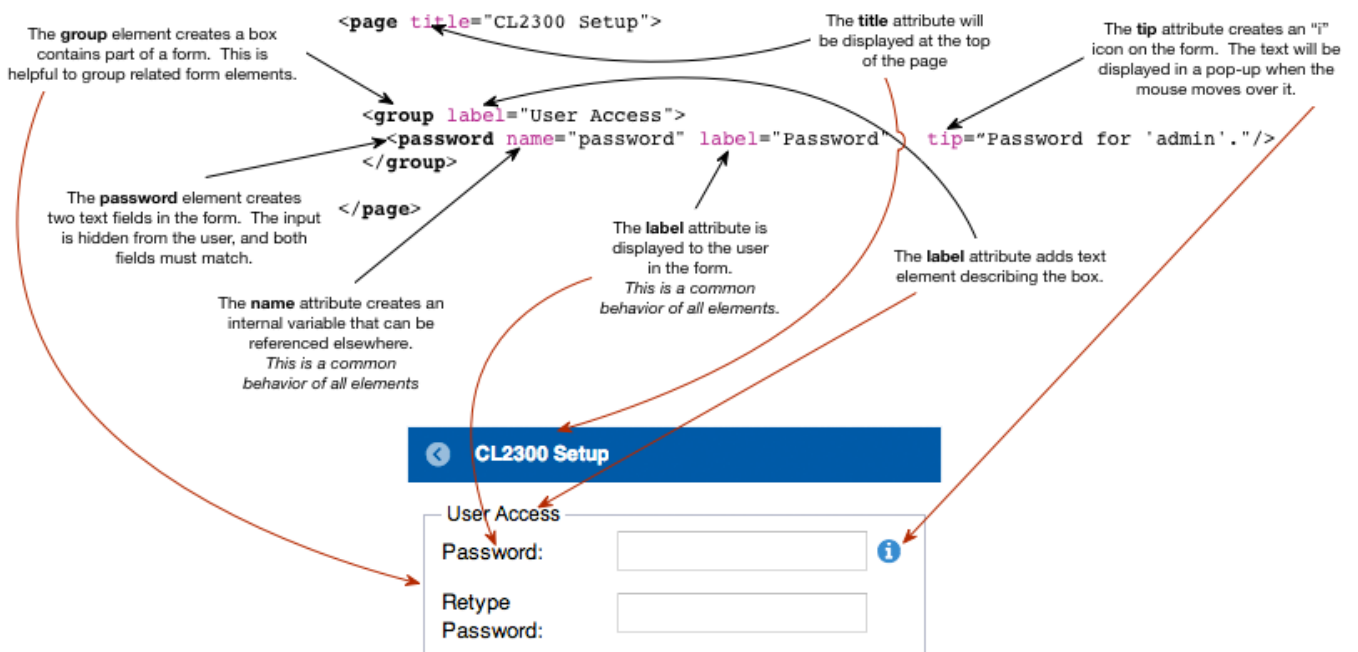
The `config-snippet` element defines conditional variables that can be included inside the final system configuration file.

config

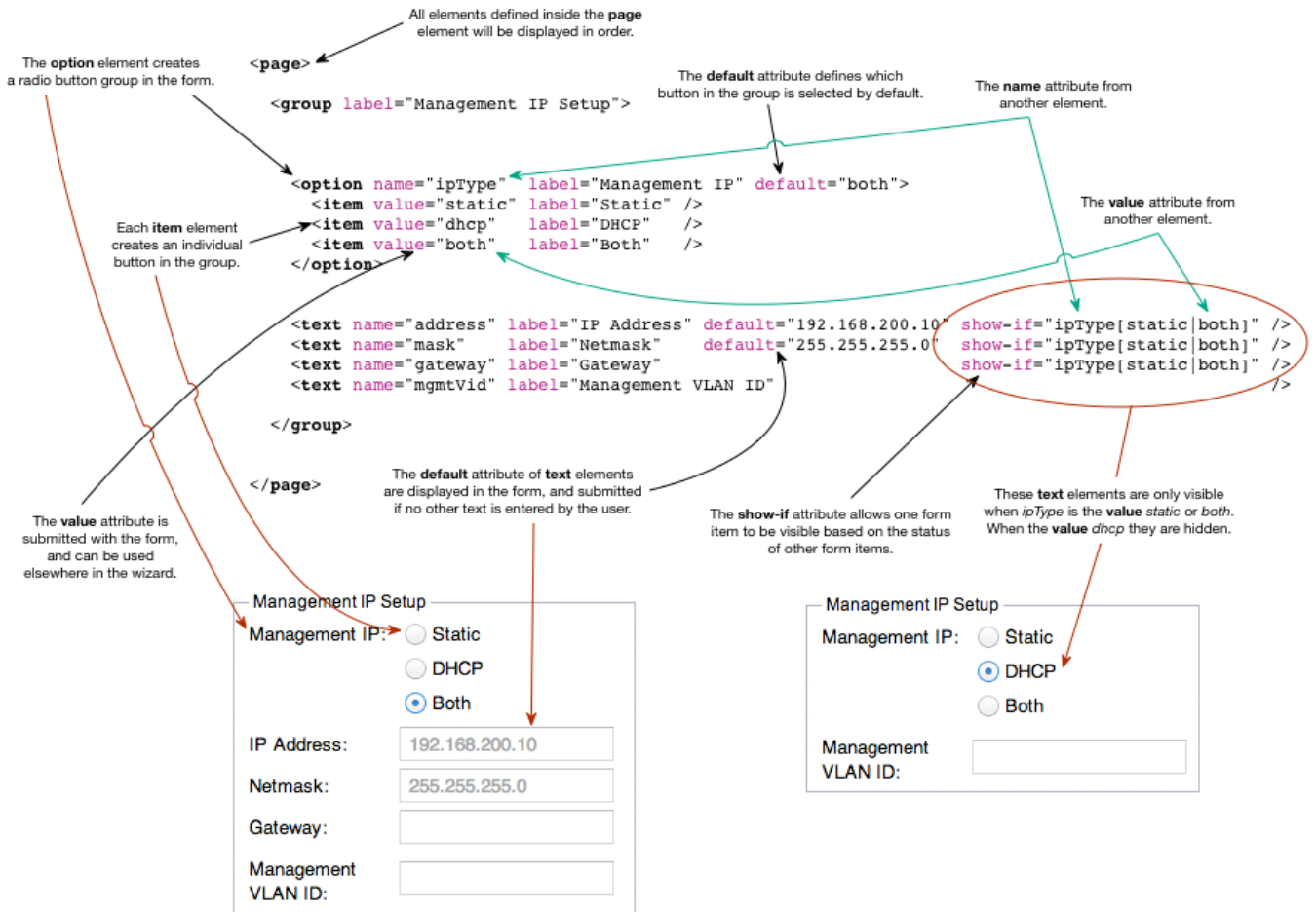
The `config` element defines the actual system configuration file that will be uploaded to the device.



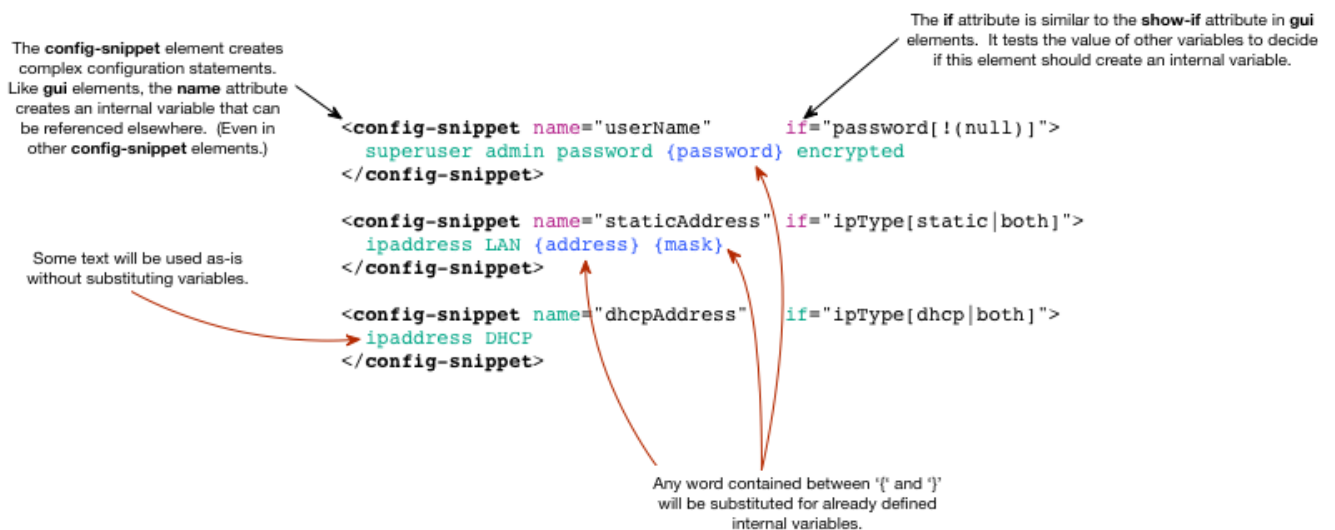
This image describes the **page**, **group**, and **password** elements.



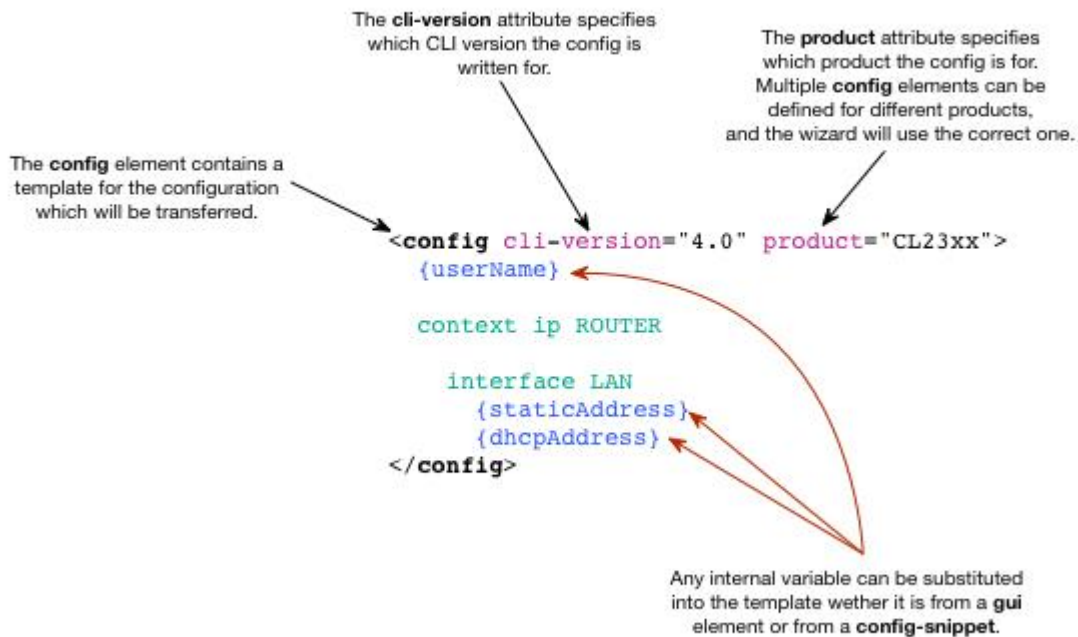
This image describes how elements inside the `page` can be hidden based on form input.



This image describes how complex System Config Snippets can be created for substitution in the main configuration.



This final image shows the main configuration file.



3.2 XML Schema Reference

The wizard XML can be validated using the wizard-schema.xsd

(available on the Patton website) <http://patton.com/wizard/schema/1.6/wizard-schema.xsd>

and a text editor with XML tools. This can be a very useful setup, as common errors and mistakes will be discovered immediately once the schema file is able to be discovered by the XML-enabled text editor. You can then edit your wizard and validate it on the fly. Note that the wizard still needs to be tested by using it to generate a config, especially with 'if' and 'show-if' attributes.

The wizard XML just needs to be edited slightly to point to the schema file. The schema file can be referenced directly from the Patton website, or downloaded first then placed in the same directory as the wizard being edited. To get started with an xml-enabled text editor like Notepad++:

1. In Notepad++, go to the plugins menu and install "XML tools".
2. In Notepad++, open the wizard XML file you want to validate, then change the start tag of the config-generator element from this:

```
<config-generator
  version="1.6">
```

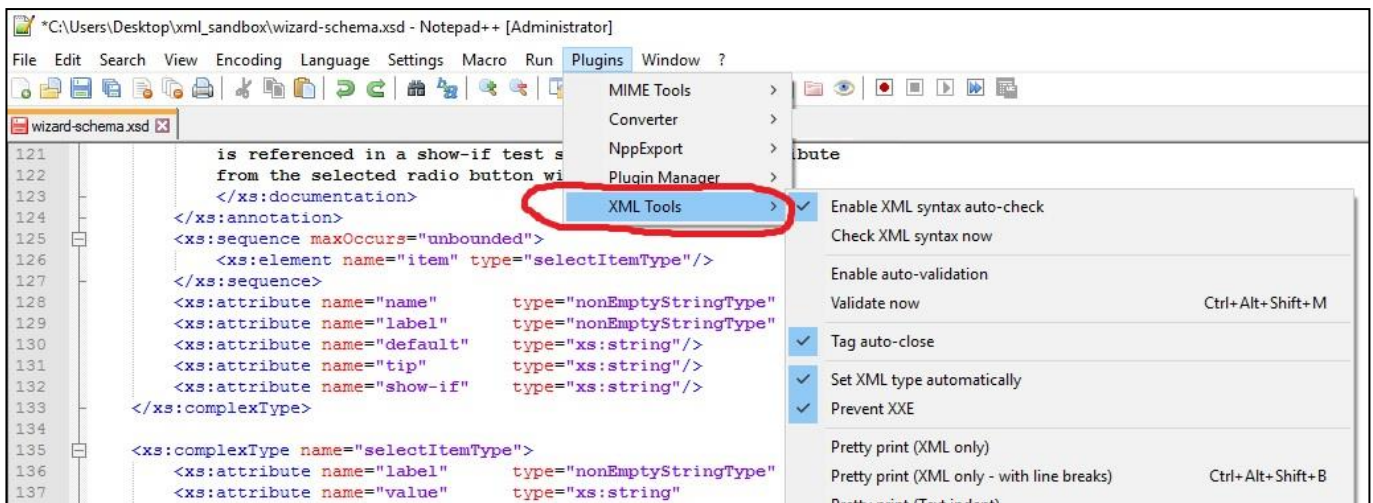
to this:

```
<config-generator
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://patton.com/wizard/schema/1.6/wizard-schema.xsd"
version="1.6">
```

- Note that the specific Patton website link above includes the version number (1.6 in this case).
- Validate through Notepad++'s Plugin menu, or with the hotkey: Ctrl + Alt + Shift + m. Common errors and missing attributes should be discovered immediately. However, the validation will fail instantly if the version number is not present, or if the root element <config-generator> is absent. This will be the case for wizards that still call that element "patton-config-generator", or are on a different version. They can quickly be converted by fixing those two issues.

The same validation procedure can be done offline by downloading the schema first, placing it in the same directory as your wizard XML file, then setting the xsi:noNamespaceSchemaLocation attribute accordingly:

```
xsi:noNamespaceSchemaLocation="wizard-schema.xsd"
```



3.2.1 Config-Generator

Each Wizard Config starts and ends with a config-generator element. This identifies a wizard configuration to the system, and what version of the interface is supported.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<config-generator
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://patton.com/wizard/schema/1.6/wizard-schema.xsd"
version="1.6">
  <!-- gui, config-snippet, and config elements go here -->
</config-generator>
```

Attribute	Status	Default	Version	Description
version	<i>required</i>		1.0	Specifies the version of Wizard engine that will support this. The current supported versions are 1.0, 1.1, 1.3, 1.4 and 1.5 (latest version 1.5)

3.2.2 Overview of available Elements

- [Title: Is being displayed on the Wizard Selection Screen](#)
- [Option: Creates a radio button group in the form](#)
- [Select: Creates drop-down list on the form](#)
- [Text: Creates a normal text input on the form](#)
- [Password: Creates two password input boxes on the form](#)
- [Number: Creates a text input on the form which will be evaluated as a number](#)
- [Check: Creates a check-box on the form](#)
- [Display: Creates a static display element on the form](#)
- [List: Creates grid inside the form that will allow a group of variables to be repeated](#)
- [Value: This defines a row that will be added to the list when the wizard is loaded](#)

3.2.3 Gui

A single gui element MUST exist inside the Wizard Config. This has the sole purpose of defining the visible information that the user will see. The sub-elements title and description are important because this information is displayed on the "Wizard Selection Screen".

Attribute	Status	Default	Version	Description
web-if-type	optional	""	1.1	This allows access to the wizard to be restricted to some user type.
reboot	optional	"true"	1.2	This allows a wizard to not automatically reboot the device.
action-label	optional	"Save & Reboot / Apply"	1.3	This allows a wizard to customize the text of the apply button. Its default value is dependent on reboot tag
action-tip	optional	"Save configuration and reboot device / / Save configuration without rebooting"	1.3	This allows a wizard to customize the help text which is shown when hovering over the apply button. Its default value is dependent on reboot tag.
page-type	optional	"sequential"	1.3	This allows the wizard to be displayed in a sequential or tab layout.

Notes for the **web-if-type** attribute:

When configured, this may cause a wizard to not appear in the "Wizard Selection Window" for some users. The only value currently supported is `basic-only`. When user is defined with the `terminal-type http web-basic-only` restriction logs in, they can only access wizards with this option set.

Notes for the **page-type** attribute:

Valid values are `"sequential"` and `"tab"`.

3.2.3.1 Title

The **title** element has no attributes. The text between the start and end tags is displayed on the "Wizard Selection Screen".

3.2.3.2 Description

The **description** element has no attributes. The text between the start and end tags is displayed on the "Wizard Selection Screen".


```
<gui>
  <title>
    CL2300 Basic Setup
  </title>
  <description>
    This sets up basic connectivity of the CL2300 LAN and WAN.
  </description>
  <!-- Content Omitted -->
</gui>
```

3.2.3.3 Page

At least one **page** element is required, however multiple elements are allowed. Each **page** element contains form definition elements between the start and end tags. When multiple **page** elements are defined, the wizard interface will cycle through the visual representation with next/previous buttons.

Attribute	Status	Default	Version	Description
title	<i>required</i>		1.0	This text value will be displayed at the top of the window, and is intended as a description of what the current page represents.

3.2.4 Common Page Element Attributes

Attribute	Description
name	This attribute creates an internal variable that will be evaluated in the wizard.
label	The text from this attribute will be displayed at the left of a form input element.
tip	The text from this attribute will be displayed to the user when hovering over the  icon to the right of a form input element.
show-if	This attribute allows a form input element to visible based on the current value of option , select , or check elements.
product	This attribute allows any element to be present only on certain products.

Notes for the **name** attribute:

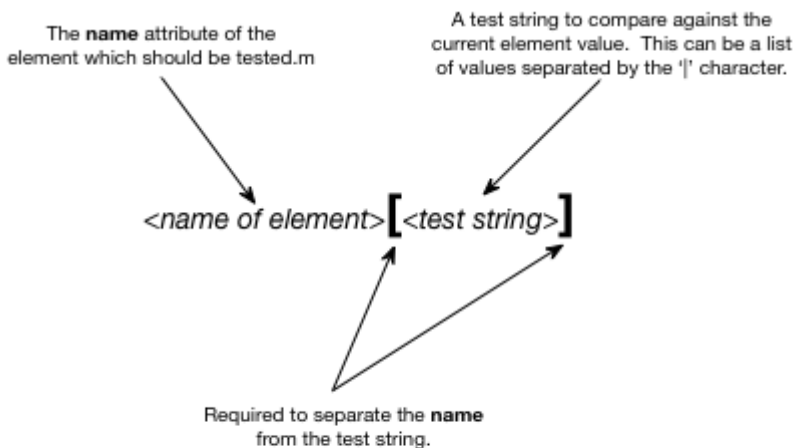
The value for this attribute **MUST** be unique for each element defined in the Wizard Configuration. This attribute is used three ways by the Wizard. It first may be referenced by the **show-if** attribute to control what form elements are currently visible to the user. In this case form elements will appear and disappear immediately as the user makes changes on the form. When the form is submitted, an internal variable will be created with the current value of input field. Any occurrences of the variable name between '{' and '}' will be substituted for the value inside the text content of **config-snippet** or **config** elements. (See example below.)

```
<text name='test'>
<config>
  {test}
</config>
```

Notes for the **show-if** attribute:

This attribute is used to control which form elements are visible based on the current value of one other form element. The image below shows a breakdown of the attribute contents. You must supply the name of the element being tested, and a test string inside square brackets. The test string provides one or more desired values of the parent element. If there are multiple values, they must be separated by the '|' (pipe) character.

- No spaces are allowed in the attribute contents.
- The value of only one element can be tested.
- The check element has the value "true" when checked and "false" when unchecked.
- An element will always be made not visible if the element name listed in show-if not visible.
- If an element is NOT visible, it will not be submitted with the form. (Even if it has a default value.)



Notes for the **product** attribute:

This attribute value will be compared against the **Model** string viewable with the CLI command "show system info". A list of possible values can be present separated by the '|' (pipe) character. The **Model** string only needs to START with the product string. I.e.. If **product** is "OS330"; it will match "OS3301", "OS3302", and "OS3304". While if **product** is "OS3302|OS3304"; it will only match "OS3302" and "OS3304".

3.2.4.1 Group

The **group** element creates visible box around multiple form elements, which provides a way to organize sections of the form.

NOTE: It is not required to use a **group** element. Any form element can be created directly inside a **page**.

```
<group label="Management IP Setup">
  <!-- Content Omitted -->
</group>
```

3.2.4.2 Option

The **option** element creates a radio button group in the form. At least one **item** element is required between the *start* and *stop* tags. When evaluating the form submission, the **value** attribute from the selected radio button will be used. When the **name** attribute is referenced in a **show-if** test string, the **value** attribute from the selected radio button will be used.

Attribute	Status	Default	Version	Description
name	<i>required</i>		1.0	See Common Attributes .
label	<i>required</i>		1.0	See Common Attributes .
default	<i>optional</i>	""	1.0	The value attribute of item which will be selected by default. If omitted (or invalid), the first item will be selected.
tip	<i>optional</i>	""	1.0	See Common Attributes .
show-if	<i>optional</i>	""	1.0	See Common Attributes .

```
<option name="ipType" label="Management IP" default="both" tip="Use a
Static IP, DHCP, or Both for management">
  <item label="Static" value="static" />
  <item label="DHCP" value="dhcp" />
  <item label="Both" value="both" />
</option>
```

3.2.4.3 Item

Each **item** element defines one radio button in the radio group.

Attribute	Status	Default	Version	Description
label	<i>required</i>		1.0	See Common Attributes .
value	<i>required</i>		1.0	This can be any text value, but should be unique among the other items in this group. The value of the selected radio button will be submitted with the form and used for show-if tests.

3.2.4.4 Select

The **select** element creates drop-down list on the form. At least one **item** element is required between the *start* and *stop* tags. When evaluating the form submission, the value attribute from the selected radio button will be used. When the **name** attribute is referenced in a **show-if** test string, the **value** attribute from the selected radio button will be used.

Attribute	Status	Default	Version	Description
name	<i>required</i>		1.0	See Common Attributes .
label	<i>required</i>		1.0	See Common Attributes .

default	optional	""	1.0	The value attribute of item which will be selected by default. If omitted (or invalid), the first item will be selected.
tip	optional	""	1.0	See Common Attributes .
show-if	optional	""	1.0	See Common Attributes .

3.2.4.5 Item

Each **item** element defines one option in the list.

Attribute	Status	Default	Version	Description
label	<i>required</i>		1.0	See Common Attributes .
value	<i>required</i>		1.0	This can be any text value, but should be unique among the other items in this list. The value of the selected item will be submitted with the form and used for show-if tests.
product	optional		1.2	See Common Attributes .

3.2.4.6 Text

The **text** element creates a normal text input on the form. Any text entered will be submitted with the form.

Attribute	Status	Default	Version	Description
name	<i>required</i>		1.0	See Common Attributes .
label	<i>required</i>		1.0	See Common Attributes .
allow-blank	optional	"true"	1.2	This attribute identifies if an empty value is permitted.
default	optional	""	1.0	This attribute allows a default value to be specified. It will be displayed in the text input as slightly grayed out. When the text area is selected for input, the default value will disappear. If no text is entered into the input box, this value will be submitted with the form.
tip	optional	""	1.0	See Common Attributes .

3.2.4.7 Password

The **password** element creates two password input boxes on the form. Text enter into either box will be masked out. The first box will use the **label** field as usual, however the second box will have "Retype" prepended to the **label**.

As text is entered into the box it will be verified against the other box, and if any password validation fields are included (*required-types*, *excluded-types*, *excluded-specific*, *max-repetitions*, *min-length*, *max-length*), the password will be checked for those requirements as well. When the text boxes do not match or fail any given requirements, they will be highlighted red, a pop-up error message will appear when moving the mouse over, and the submit buttons will be disabled.

If the password is required (by setting *min-length* to a value greater than "0"), the validation attributes will be enforced: *required-types*, *excluded-types*, *excluded-specific*, *max-repetitions*, *min-length*, *max-length*. Be sure to set the *min-length* attribute, otherwise blank/empty passwords will be permitted. To require or exclude character types, use any combination of 'upper', 'lower', 'number', and 'symbol' in the *required-types* and *excluded-types* attributes, separated by |, i.e. *required-types*="number|upper" would require that at least one number and uppercase letter are present in the password. To prohibit specific characters, they can be listed directly in the *excluded-specific* attribute. Some special characters need to be escaped however (for example & as & and " as "). When the text boxes do not match or fail validations, they will be highlighted red, a pop-up error message will appear when moving the mouse over, and the submit buttons will be disabled.

When the form is submitted, the password will be encrypted by the device. The encrypted password will be used when "{name}" is used in `config-snippet` or `config` elements. A '_' character can be appended to `name` if the plain text password is required for substitution.

NOTE: The password will be sent to the device as clear readable text during the encryption process. For added security, use HTTPS to access the WEB interface.

NOTE: This provides only the encrypted password. Some CLI commands may require an extra keyword to identify it as encrypted. The following user name example.

```
# Encrypted version
superuser admin password {password} encrypted
# Unencrypted version
superuser admin password {password_}
```

Attribute	Status	Default	Version	Description
name	<i>required</i>		1.0	See Common Attributes .
label	<i>required</i>		1.0	See Common Attributes .
label-prefix	optional	"Retype"	1.2	This attribute allows the prefix applied label for the second text input to be set.
tip	optional	""	1.0	See Common Attributes .
required-types	optional	""	1.5	Required character types. Valid values are upper, lower, number, symbol, or any combination separated by i.e. upper lower number symbol
excluded-types	optional	""	1.5	Excluded character types. Valid values are upper, lower, number, symbol, or any combination separated by i.e. upper lower number symbol
excluded-specific	optional	""	1.5	Any specific characters that are not allowed in the password, listed with no separator, i.e. ~!@-#
max-repetitions	optional	""	1.5	The maximum times any character can appear in the password, to prohibit passwords like 111111, or aaaaaa
min-length	optional	"0"	1.5	The minimum length of the password. Note: min-length="0" allows to set an empty password.
max-length	optional	""	1.5	The maximum length of the password.

3.2.4.8 Number

The `number` element creates a text input on the form which will be evaluated as a number. Any text entered will be submitted with the form.

Attribute	Status	Default	Version	Description
name	<i>required</i>		1.0	See Common Attributes .
label	<i>required</i>		1.0	See Common Attributes .
allow-blank	optional	"true"	1.2	This attribute identifies if an empty value is permitted.
default	optional	""	1.0	This attribute allows a default value to be specified. It will be displayed in the text input as slightly grayed out. When the text area is selected for input, the default value will disappear. If no text is entered into the input box, this value will be submitted with the form.
tip	optional	""	1.0	See Common Attributes .

3.2.4.9 Check

The **check** element creates a check-box on the form. The element has a value of "true" when checked and "false" when unchecked, which will be submitted with the form and used for **show-if** tests.

Attribute	Status	Default	Version	Description
name	required		1.0	See Common Attributes .
label	required		1.0	See Common Attributes .
default	optional	"false"	1.0	The valid values are "true" or "false". This will make the box start "checked" or "unchecked" in the form.
tip	optional	""	1.0	See Common Attributes .
show-if	optional	""	1.0	See Common Attributes .

3.2.4.10 Display

The **display** element was redefined in **version 1.6**. Previous versions, until **version 1.5**, are **incompatible** with the new definition.

The **display** element creates a static display element on the form. (This is really only useful inside the **list** element.)

An example of a display text placed on top of a wizard page:

```

...
<page>
  <display>
    This page sets up user credentials and access methods.
  </display>
  ...
</page>
...

```

```

...
<group>
  ...
  <display>
    The next field requires some more explanation. And this
    can even happen on multiple lines in the wizard code and
    on the wizard page.
  </display>
  ...
</group>
...

```

3.2.4.11 List

The **list** element creates grid inside the form that will allow a group of variables to be repeated. At least one GUI element is required between the *start* and *stop* tags. The GUI elements will be displayed in the grid and inside a popup window. The popup window will be shown when pressing the '+' button or double-clicking on a grid row.

Attribute	Status	Default	Version	Description
-----------	--------	---------	---------	-------------

name	<i>required</i>		1.3	See Common Attributes .
title	<i>required</i>		1.3	A title displayed at the top of the grid.
sort-field	optional	""	1.3	The name of a gui element used to sort the grid. As of version 1.4 this will also enforce a unique row.
allow-new	optional	"true"	1.4	Show the '+' button, and allow new entries.
allow-del	optional	"true"	1.4	Show the '-' button, and allow deleting entries.
height	optional	200	1.4	Set the grid height.

Notes for the **name** attribute:

Unlike other GUI elements, this cannot be used directly in **config** or **config-snippet** contents. It can only be used in the **for** attribute of a **config-snippet**.

3.2.4.12 Value

Each **value** element defines a row that will be added to the list when the wizard is loaded. The data for the row MUST be defined between the *start* and *stop* tags of the **value** element. The data for each field MUST be included in a tag which uses the field **name**. See the example below.

NOTE: A row added with the **value** element can be deleted if the **allow-del** attribute is not set.

Attribute	Status	Default	Version	Description
if	optional		1.4	Add or remove row based on the status of another row.

Notes for the **if** attribute:

This attribute is used to control if the **value** is added to the list based on the current status of other list items. This attribute requires a test string with a similar format as **show-if**. The **if** attribute allows multiple variables to be tested by adding a '+' character between each test. There is an added reference to the specific row to be tested (indexed by the **sort-field**). I.e.. Here is a sample test string: "0/0{service[2-wire|4-wire]}+0/2{service[2-wire]}".

- No spaces are allowed in the attribute contents.
- The value of multiple variables can be tested.
- If a variable does not exist, it will be evaluated as "(null)"

```

<list name="lines" title="Lines" allow-new="false" allow-del="false"
sort-field="line" height="1200">
  <display name="line" label="Line"/>
  <text name="useProfile" label="Profile Name"/>
  <check name="enabled" label="Enable" default="true"/>
  <select name="service" label="Service Mode">
    <item value="2-wire" label="2-wire" />
    <item value="4-wire" label="4-wire" />
    <item value="8-wire" label="8-wire" />
  </select>

  <!-- This is always present -->
  <value>
    <line >0/0< /line>
    <useProfile>DEFAULT</useProfile>
    <enabled >true< /enabled>
    <service >2-wire< /service>
  </value>
  <!-- This is only present when 0/0 is 2-wire -->
  <value if="0/0{service[2-wire]}">
    <line >0/1< /line>
    <useProfile>DEFAULT</useProfile>
    <enabled >true< /enabled>
    <service >2-wire< /service>
  </value>
  <!-- This is only present when 0/0 is 2-wire or 4-wire -->
  <value if="0/0{service[2-wire|4-wire]}">
    <line >0/2< /line>
    <useProfile>DEFAULT</useProfile>
    <enabled >true< /enabled>
    <service >2-wire< /service>
  </value>
  <!-- This is only present when 0/0 is 2-wire or 4-wire and 0/2 is 2-
wire -->
  <value if="0/0{service[2-wire|4-wire]}+0/2{service[2-wire]}">
    <line >0/3< /line>
    <useProfile>DEFAULT</useProfile>
    <enabled >true< /enabled>
    <service >2-wire< /service>
  </value>
</list>

```

3.2.5 Config-Snippet

The **config-snippet** element creates an internal variable AFTER form submission that will be evaluated by the wizard. This allows complex configuration snippets to be created using data entered on the form. The form data can be used two ways:

1. For any occurrences of the **name** attribute of form elements found between '{' and '}' inside the text content; the submitted data will be substituted into the text. In the [GUI Elements example](#) above, a **text** element was created named *address* and *mask*. Later in the [Config Snippet example](#), the **config-snippet** *staticAddress* is created with the following text: "ipaddress LAN {address} {mask}". The values for *address* and *mask* from the form will be used here.
2. The value of form elements can be tested to decide if the **config-snippet** should be evaluated. This behavior is very similar to how the **show-if** attribute works for **gui** elements.

NOTE: The wizard will evaluate `config-snippet` elements in the order they exist in the document. Once evaluated, they can be used by other `config-snippet` elements just like form data.

Attribute	Status	Version	Description
<code>name</code>	<i>required</i>	1.0	This attribute is used as the internal variable name.
<code>if</code>	optional	1.0	This allows the element to be evaluated based on the current value of any item submitted in the form or other <code>config-snippet</code> elements.
<code>product</code>	optional	1.2	See Common Attributes .
<code>for</code>	optional	1.3	Evaluate the contents for each item in a <code>list</code> .

Notes for the `name` attribute:

The value for this attribute **SHOULD** be unique, but it is not required. If the name is not unique, the old value will be replaced by the newly evaluated `config-snippet`. There are two cases where it may be desirable to use a non-unique `name`.

1. Based on the value of one form element, you want to replace the value of another element.
2. Multiple config snippets are created with the same `name`, and based on the `if` and/or `product` tests only one will be evaluated.

Notes for the `if` attribute:

This attribute is used to control if the element is evaluated based on the current value of other variables. This behaves similar to the `show-if` attribute used on `gui` elements, however there is no limitation for which elements are evaluated. This attribute requires a test string with the same format as `show-if`. The `if` attribute allows multiple variables to be tested by adding a '+' character between each test. (This is an AND operation, so the test **MUST** pass for each variable. I.e.. Here is a sample test string: `"ipType[static|both]+mgmtVid[(null)]"`.

- No spaces are allowed in the attribute contents.
- The value of multiple variables can be tested.
- The `check` element has the value `"true"` when checked and `"false"` when unchecked.
- If a variable does not exist, it will be evaluated as `"(null)"`

Notes for the `for` attribute:

The contents will be evaluated for each row in the names `list`. The the `name` attribute is referenced the complete expanded text will be used.

3.2.6 Config

The `config` element contains a CLI configuration file template which will be uploaded to the device. Like the `config-snippet` element, the text content will be evaluated for all the internal variables. The value of internal variables will be substituted into the text for each occurrence of the name found between '{' and '}' characters. At least one `config` element **MUST** exist, however it is possible to define multiple elements.

The `config` elements are evaluated in the order they exist in the document. If the wizard decides multiple configuration files will be sent to the device, it is possible that they are attempting to save as the same file name. In this case, the last one submitted will be used by the device.

Attribute	Status	Default	Version	Description
cli-version	<i>required</i>		1.0	The target CLI Version. This adds "cli version XXX" to the generated configuration file.
product	optional		1.2	See Common Attributes .
file	optional	"startup-config"	1.1	The attribute defines the configuration file name on the target.
persist	optional	"true"	1.1	This attribute tells the system to apply the text content to the <code>running-config</code> instead of saving to a file.
backup	optional	"true"	1.4	This attribute tells the system to backup the configuration specified in the file attribute.

Notes for the **file** attribute:

Any file name can be used here. If the file already exists on the device, the system will automatically back it up with the current date and time in the file name. **NOTE:** The `minimal-config` cannot be replaced.

Notes for the **persist** attribute:

This attribute allows any CLI command which the user has access to be applied to the running device. Unless a command to save the configuration is present, the changes may not be persistent over reboot. **NOTE:** The wizard will still reboot the device unless the **reboot** attribute of the **gui** element is configured as *false*.