Nagios XI - Creating Nagios XI Wizards

Article Number: 53 | Rating: Unrated | Last Updated: Wed, Nov 28, 2018 at 9:46 PM

Purpose

This document describes how to write custom configuration wizards in Nagios® XI™. This document will cover how to create a new wizard using a custom plugin and also cover how to utilize some of the programming tools within the Nagios XI framework.

Target Audience

This document is intended for use by Nagios XI Administrators and Developers wanting to create custom configuration wizards, and is intended for an audience that has some familiarity with programming and HTML.

Sample Wizard Code

This document does not go into great detail on how to structure or write PHP code for a wizard. A sample wizard has been created that can be used in conjunction with this document and used as the basis of a new wizard.

You can download the example wizard code and structure using the following link:

http://assets.nagios.com/downloads/nagiosxi/wizards/wizarddemo.zip

Setting Up The Development Environment

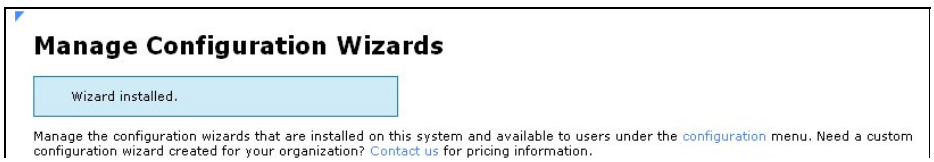
Developers have unique preferences as to how to set up their development environment in PHP, however, the following setup will be the simplest way to work with and debug a wizard while it's in development.

Please refer to the Configure PHP Development Environment KB article for these steps.

This will generate error output directly to the browser window but will also create enough filtering so that the error output is easier to decipher. Before proceeding to the actual wizard code structure, see the section below on **Debugging Tips** to save time in the development process.

Debugging Tips

Login to the Nagios XI web interface and navigate to the **Admin** menu. Right click on the **Manage Config Wizards** under **System Extensions** and select



"Open In New Tab." Leave that page there for a moment, and access the **Configure** menu, and right click the **Run the Monitoring Wizard** link and select "Open In New Tab" as well. Both of these tabs will be beneficial when testing and developing the wizard.

While developing the wizard, we recommend developing the code on a local workstation and then uploading the zip file periodically to ensure the full functionality and compatibility of the wizard. If you were to access the **Manage Configuration Wizards** page directly and you uploaded a wizard that had a php syntax error on it, you should clearly see the error output on screen. This allows you to clearly identify any fatal errors in the code before proceeding to the stages of the wizard.

To fix the syntax error, modify your local copy, rezip the directory, and then upload the wizard again. The old wizard will simply be overwritten. Once the wizard loads with no syntax errors, leave this page open, but select the tab opened to the **Monitoring Wizard – Step 1** page. You will likely have to upload the wizard several more times until debugging is complete.

For debugging the wizard stages (**monitoringwizard.php**), error outputs will appear at the top of the page. Be sure to account for any undefined variables to prevent the Apache logs from being cluttered with error messages, and also to prevent bugs in the wizard itself.

Every time you upload a new wizard into Nagios XI, select the URL for the **monitoringwizard.php** page, and hit "Enter" to start the wizard over from stage one, and to clear any POST variables that will affect wizard navigation. You will likely need to repeat the process of rezipping the files, uploading to Nagios XI, and restarting the wizard several times, so leave both tabs open while you develop.

XI Wizard Development Guidelines

The development guidelines for Nagios XI Wizards are still somewhat loosely defined, but the following conditions will maximize compatibility, security, and reliability of the Configuration Wizard. Contact the Nagios XI Support Team if you have questions about your code, and see php.net for the best reference on PHP syntax and built-in functions.

- Wizards must be free from all **fatal**, **syntax**, and **notice** error messages. This includes accounting for undefined variables and array indices.
- Wizards should never run UPDATE or INSERT SQL queries directly into the nagios or nagiosql databases. This will have unpredictable results and will most likely break a monitoring configuration.
- To maintain security within Nagios XI, avoid interacting directly with the $_POST, $_GET, or $_GLOBALS arrays. To access variables submitted in wizard forms, use the "grab_array_var()" function as documented below. The $inargsarray contains all of the POST data from each stage of the form.

```
$form_variable = grab_array_var($inargs, $variable_name, $default_value)
```

This will use some of the security features built into Nagios XI to clean any input variables and prevent XSS vulnerabilities. There is an exception to this in the wizard example, but the input variable is processed and cleaned of vulnerabilities.

- Wizard data can be passed forward with either a $_SESSION array, which is demonstrated in the wizarddemo.zip, or by serializing the data and passing it along through hidden form inputs, which will be seen in

wizarddemo.zip, or by serializing the data and passing it along through hidden form inputs, which will be seen in most wizards prior to Nagios XI 2011R1.3. The new $_SESSION method is simpler for repopulating the form if a user selects the "Back" button.

Overview of Configuration Wizard Stages

The Nagios XI wizard framework is designed to handle user-defined wizard stages in the following progression. Each of these stages are prepended with CONFIGWIZARD_MODE_

- GETSTAGE1HTML: HTML header reads this as **Step 2**
- VALIDATESTAGE1DATA: Form validation for stage 1 HTML
- GETSTAGE2HTML: HTML header reads this as **Step 3**
- VALIDATESTAGE2DATA: Form validation for stage 2 HTML
- GETSTAGE3HTML: HTML header reads this as **Step 4**. This stage can be empty . Used mostly to process and save input variables from previous step.
- VALIDATESTAGE3DATA: Form validation for stage 3 HTML
- GETSTAGE3OPTS: [optional] Allows "check settings" to be hidden and/or overridden
- GETSTAGE4OPTS: [optional] Allows "alert settings" to be hidden and/or overridden
- GETFINALSTAGEHTML: A final stage to confirm the Apply Configuration for the new settings

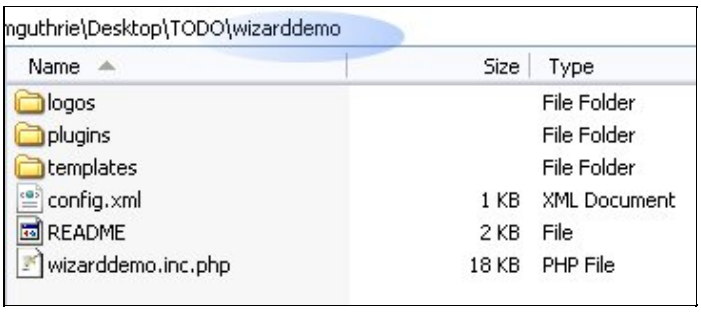
Overriding Stages

Currently, the wizard framework allows for a limited amount of customization to the later stages of the Configuration Wizard, primarily the check settings and alert settings. These override options have been documented internally in the wizarddemo example code for the available options and how to use them. Future versions of Nagios XI will include options to skip entire stages and allow for manual definition of hostgroup, servicegroup, and parent relationships.

Wizard File Structure Overview

The files in a configuration wizard must be structured and named according to the following conventions in order for the wizard to function correctly. The following example will demonstrate naming conventions for a wizard with the name "wizarddemo." See the zip file from the **Sample Wizard Code** for more details.

All files must be placed inside of a directory called "wizarddemo" and once completed the entire directory should be zipped into a file called "wizarddemo.zip"



PHP SCRIPTS:

- The main include file must be named wizarddemo.inc.php
- Any additional includes referenced by the wizarddemo.inc.php file don't require a naming convention

LOGOS:

- Must be placed in the "logos" directory and named wizarddemo.png or wizarddemo.jpg, etc.
- The image size for the wizard logo should be 40x40px.

PLUGINS:

- Check plugins can be placed in a directory called "plugins" and must match the name specified in the config.xml file

CONFIG TEMPLATES:

- Config templates and definitions must be placed in a "templates" directory and defined in a file named wizarddemo.cfg.
- New host or service template definitions must be named with the following convention:
    - `xiwizard_wizarddemo_service`
    - `xiwizard_wizarddemo_host`

ADDITIONAL FILES:

Any additional files that are not PHP scripts must be specified in a file called config.xml. Example of the wizarddemo's config.xml below:

```
<configwizard>
<templates>
<template filename="wizarddemo.cfg" />
</templates>
<plugins>
<plugin filename="check_cap" />
</plugins>
<logos>

<logo filename="wizarddemo.png" />
</logos>
</configwizard>
```

Using A Session Array For Wizard Data

The wizarddemo code demonstrates in detail how to use a $_SESSION array to work with wizard stages and data. The following code is an example of how to establish a session array for a wizard in the stage: CONFIGWIZARD_MODE_GETSTAGE1HTML.

```
//check to see if this is a fresh wizard run, or if we're coming back from a later stage
$back = htmlentities(grab_array_var($_POST,'backButton',false),ENT_QUOTES);
```

```
//clear any previous session data for this wizard, start a new session array
if(!$back)
{
unset($_SESSION['wizarddemo']);

//create a new session array to hold data from different stages
$_SESSION['wizarddemo'] = array();
}
```

Final Thoughts

For any support related questions please visit the Nagios Support Forums at:

http://support.nagios.com/forum/

Posted by: **swilkerson** - Tue, Feb 3, 2015 at 3:06 PM. This article has been viewed 11365 times.

Online URL: https://support.nagios.com/kb/article/nagios-xi-creating-nagios-xi-wizards-53.html