Nagios XI – How To Write Custom Components

Nagios®

The Industry Standard in IT Infrastructure Monitoring

Purpose

This document describes how to use the Nagios XI Framework to access data and build a successful component. Components are developed to extend the functionality of your Nagios XI installation.

Target Audience

This document is intended for use by Nagios XI Administrators and Developers looking to customize their installation by creating their own components.

Overview

This document will cover the following topics:

Example Component Code

Each of the code examples from this document will be used from the following **Nagios XI Mass Acknowledgment Component**, which can be obtained from the link below. Download this component and review the functions listed throughout this document as each are explained.
http://exchange.nagios.org/directory/Addons/Components/Mass-Acknowledgement-Component/details

General Developer Guidelines

The development guidelines for Nagios XI Components are still somewhat loosely defined, but the following conditions will maximize compatibility, security, and reliability of the component. 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.

- Components must be free from all **fatal**, **syntax**, and **notice** error messages. This includes accounting for undefined variables and array indices.
- Components should not 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. If the component requires that this be done, the component code should be reviewed by the Nagios XI Development Team if it's going to be published for public use.
- To maintain security within Nagios XI, avoid interacting directly with the $_POST, $_GET, or $_GLOBALS arrays. To access variables submitted in forms, use the “grab_request_var()”. For example:

  ```
  $form_variable = grab_request_var('indexName', 'default_value');
  ```

  Would replace the following call:

  ```
  $form_variable = $_POST['indexName'] or $_GET['indexName'] or 'default_value';
  ```

  This will use some of the security feature built into Nagios XI to clean any input variables and prevent XSS vulnerabilities.

Updated – February, 2017

Setting Up Nagios XI For A Development Environment

Different developers have their own preferences as to how to set up their development environment in PHP. The following setup will be the simplest way to work with and debug a component while it's in development.

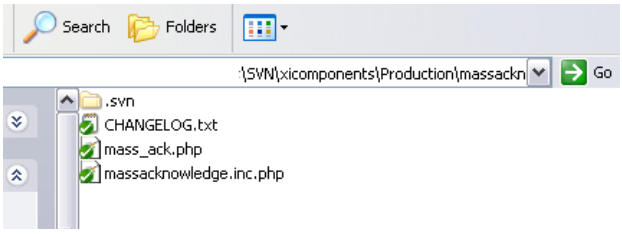
Open the /etc/php.ini file in your preferred text editor, and change the existing settings to match the following:

```
error_reporting  =  E_ALL & ~E_WARNING
display_errors = On
ignore_repeated_errors = On
```

Run `service httpd restart` from the command-line to restart apache and load the new PHP settings.
This will generate error output directly to the browser window, but also create enough filtering so that the error output is easier to decipher.

Component Registration and Initialization

In order to register a component with Nagios XI, a special include script is required. For this and further examples, we will use code examples from the **Mass Acknowledgement** component. The ID we will give this component is: **massacknowledge**, therefore we will use that same name in several places throughout this component. Start by creating a directory called massacknowledge. All of the files will be stored inside of this directory. In the top level of the directory there needs to be a file called **massacknowledge.inc.php**. This is a required include script that the Nagios XI framework will look for upon component installation, and it will contain all of the component registration information.



Component Initialization Code

The best way to handle component initialization for a new development, is to start with an existing component's registration script and modify it to fit the new component. Make sure all of the functions in the registration script (yourcomponent.inc.php) are prepended with the component's ID. Examples for the massacknowledge component are list:

```
function massacknowledge_component_checkversion(),
function massacknowledge_component_addmenu($arg=null)
function massacknowledge_component_init()
```

Examples for your component (yourcomponent) would be as follows:

```
function <yourcomponent>_component_checkversion(),
function <yourcomponent>_component_addmenu($arg=null)
function <yourcomponent>_component_init()
```

The `<componentname>_component_init()` function submits a list of constants to Nagios XI to register the component. Here is the complete list of possible options to initialize a component.

- `COMPONENT_TITLE`
- `COMPONENT_VERSION`
- `COMPONENT_AUTHOR`
- `COMPONENT_DESCRIPTION`
- `COMPONENT_DATE`
- `COMPONENT_COPYRIGHT`
- `COMPONENT_LICENSE`
- `COMPONENT_HOMEPAGE`
- `COMPONENT_CONFIGFUNCTION`

Adding A Component To the Menu

Review the function "massacknowledge_component_addmenu()" for the example code. The menu constants in Nagios XI can be used to identify the top menu items on the blue navigation bar, such as MENU_HOME, MENU_CONFIGURE, and MENU_ADMIN. The following example finds a location within the main home menu, and then adds a link into Nagios XI.

```
$mi=find_menu_item(MENU_HOME,"menu-home-acknowledgements","id");
$order=grab_array_var($mi,"order","");  //extract this variable from the $mi array
$neworder=$order+0.1; //determine my menu order

//add this to the main home menu
add_menu_item(MENU_HOME,array(
       "type" => "link",
       "title" => "Mass Acknowledge",
       "id" => "menu-home-massacknowledge",
       "order" => $neworder,
       "opts" => array(
              //this is the page the menu will actually point to.
              //all of my actual component workings will happen on this script
              "href" => $urlbase."/mass_ack.php",
              )
       ));
```

A list containing most of the default menu items is located in the **/usr/local/nagiosxi/html/includes/utils-menu.inc.php** script. This script can be viewed to determine menu ID's for placement of your new menu item.

Using The Backend API To Get XML Data

The Nagios XI framework has a large collection of functions to retrieve XML data from the backend. Although this document won't be able to cover all methods and options for the data retrieval, it will cover some of the primary functions and options needed for most components.

Necessary Functions for All Nagios XI Data

The following functions should be at the top of all web-accessible scripts for any Nagios XI Component. These will verify the session authentication and connect the script to all databases, and give full access to any of the necessary includes for the Nagios XI framework.

```
require_once(dirname(__FILE__).'/../../common.inc.php');

// initialization stuff
pre_init();
// start session
init_session();
// grab GET or POST variables
grab_request_vars();
// check prereqs
check_prereqs();
// check authentication
check_authentication(false);
```

Previewing XML structure in Nagios XI:

By accessing the following URL in Nagios XI, commands can be submitted to retrieve XML for use in components and external applications.
http://<yourserver>/nagiosxi/backend/?cmd=gethoststatus

Below is a list of common backend commands. The complete list of commands available from this page are located in the file **/usr/local/nagiosxi/html/backend/index.php:**

- gethoststatus
- getservicestatus
- gethosts
- getservices
- getcomments
- getprogramstatus
- getusers
- getparenthosts
- getcontacts
- gethostgroups
- gethostgroupmembers
- getservicegrouphostmembers
- getcustomhostvariablestatus
- getstatehistory
- getnotifications

Internal PHP Functions for Backend Data

Example for retrieving a brief list of host status data:

```
$backendargs["cmd"]="gethoststatus";
$backendargs["brevity"]=1;
$xml=get_xml_host_status($backendargs);
```

Common Backend Functions:

```
get_xml_program_status($args)
get_xml_service_status($args)
get_xml_custom_service_variable_status($args)
get_xml_host_status($args)
get_xml_custom_host_variable_status($args)
get_xml_comments($args)
```

Common Backend $arg options:

```
$backendargs["cmd"]="getservicestatus";
$backendargs["combinedhost"]=1;
$backendargs["current_state"]="in:1,2,3";
$backendargs["has_been_checked"]=1;
$backendargs["problem_acknowledged"]=0;
$backendargs["scheduled_downtime_depth"]=0;
$backendargs['is_active']=1;
//sort list by last state change
$backendargs["orderby"]='last_state_change:d';
//get the XML data
$xml=get_xml_service_status($backendargs);
```

Adding Nagios XI's Javascript and CSS

If your component doesn't require any special Javascript or CSS specifications, the simple way to start a web-accessible PHP script for your component is with:

```
do_page_start(array("page_title"=>"MyComponent"),true);
```

This will take care of adding all necessary HTML head information for a standard Nagios XI web page. If you just wish to access Nagios XI's stylesheets and JS includes, but you want to be able to specify your own HTML head information (such as additional CSS styles or Javascript functions), then you can just use:

```
do_page_head_links();
```

And this will include only the head tags for Nagios XI's CSS and Javascript includes.

Finishing Up

If you have additional questions about developing a component using the Nagios XI framework, or for any other support related questions, please visit the Nagios Support Forums at:

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