**PHP NETFLOW MANGEMENT SCRIPT**

***Behailu B. Bekera***

***Department of Computer Science and Mathematics,***

***Westminster College***

***Westminster Ave., Fulton, MO, 65251***

***Summer Internships in Science and Technology (SIST) Program***

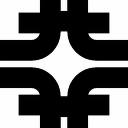


***Supervisor: Joe Klemencic***

***Computing Division-Network Security***

***Fermi National Accelerator Laboratory,***

***Batavia, IL 60510***



Contents

[ABSTRACT 3](#_Toc237099559)

[INTRODUCTION 4](#_Toc237099560)

[MATERIALS AND METHODS 5](#_Toc237099561)

[RESULTS 6](#_Toc237099562)

[CONCLUSION 8](#_Toc237099563)

[ACKNOWLEDGEMENTS 9](#_Toc237099564)

[REFERENCES 10](#_Toc237099565)

[APPENDIX-1 11](#_Toc237099566)

[APPENDIX-2 15](#_Toc237099567)

## ABSTRACT

The Fermilab Computer Security Team administers the laboratory’s computer security program and provides the Fermilab community with technical expertise and up-to-date information and resources for improving computer security. Devices that are active in the Fermilab network, but are not sending log files to the central logging server, could pose a huge threat to the ongoing research. It is therefore profoundly necessary to have all devices that offer a network service send log files to the central logging server and identify those that are not. To do this, the Netflow Management Project was designed and implemented. After identifying the problem and what the solution must do, I did some work to learn about Linux, PHP and the Git Version Control System which dominated the early stage of this project. The next task was to develop a logical sequence of steps used to solve the problem. In the implementation phase, the algorithm was translated into a PHP code which was then tested by executing it several times and debugging the errors. When executed, the product opens a file containing information about devices offering network services, examines the status of all files included under IP address and hostname of each device and uses this information to identify the ones that are not sending log files to the central logging server. If directories of IP address and hostname are not found the device is not forwarding its log files at all. If the same file is checked to contain zero bytes in both of its locations, it means that the device has stopped sending log files to the central logging system at some point. If files are not in the given time variant then the device has not been forwarding its logs to the central logging server during that period of time. The flow engine will notify the network security team as any of the above problems is incurred and necessary measures will be taken to avoid any damage to the network system.

## INTRODUCTION

As a security precaution, any device that offers a network service at Fermilab must forward its logs to the central logging server. If it is not, the situation may be intriguing to a network security management team because that will leave ongoing research activities at high risk. The device may have been hacked in which case immediate measures need to be taken to prevent any potentially dangerous unauthorized activities and any unanticipated delay in the course of research. The Netflow Management Project is intended to identify systems that are not logging to the central logging server, and alerting the computer security team of such occurrences. To do this, a flow engine is designed and is made available for execution. By executing the engine we intend to discover devices that offer services to the outside world and ensure that they are forwarding their system logs to us. Some of the tasks performed by the engine are: reading an IP address of an on-site device from a file, resolving the corresponding hostname, checking if the directories associated with the IP address and hostname exist, getting the status of all files under each directory, and eventually checking to see if each of these files is modified within a given time variant. It was necessary that I learn PHP to write the code, VI to edit the code and the Git Version Control system to work with my supervisors on the same code base, without having to constantly send files back and forth. There was no background work done regarding the project and hence it was picked up from scratch. This project focuses on design, coding and testing a PHP script. The script will enormously be helpful in improving computer security in order to safeguard research activities at Fermilab.

## MATERIALS AND METHODS

PHP, Linux and the Git Version Control System are the primary tools that were used throughout the project. Books and electronic resources such as online tutorials were used to learn how to use these tools. Once I felt confident in these areas, I began developing some code for the preliminary stages of the project. After spending much time using these tools, I was eventually able to write code for more complicated sections of the project. At first, the code receives inputs like *php engine/flow-engine.php -f netflow-output.txt -v 5h* and the required actions are initiated from the menu array based on the information provided in the input command.

*$log = App\_Log::getInstance(IDENT);*

*$config = Ini\_Config::getInstance();*

*$cg = new Zend\_Console\_Getopt(*

*Array(*

*'help|h'=> 'Display this help and exit',*

*'flow-file|f=s' => 'Run against the provided flow file',*

*'output|o=s' => 'Output format to write results to',*

*'time|t=s' => "Alert if the machine hasn't reported in this much time",*

*'timeVar|v=s' => "time variant to check the files for"*

*)*

The class controller is then instantiated to take over the entire process. It invokes the setFilename, setTimeVariant and Dispatch member sequentially (see Appendix-1 for details),

*$controller = Engine\_Front::getInstance();*

*$controller->setFilename($flowfile);*

*$controller->setTimeVariant($variant);*

*$controller->dispatch();*

In the Dispatch function (see Appendix-2 for details), a file is opened and each line in the file is consumed in a loop. The trim function is used to eliminate empty spaces from the beginning and end of each line. Trimmed lines are split on boundaries formed by spaces, returning an array of strings. One of the array components is the destination IP address for which the program resolves the corresponding hostname. There are a number of preconditions and post conditions included in the code that allow future developers to hook into different portions of the engine to change its behavior at runtime. Eventually, the filecheck function (see Appendix-1 for details) is called twice to check the IP address and hostname directories separately.

Sample File from Which IP Addresses are Consumed:

Start End Sif SrcIPaddress SrcP DIf DstIPaddress DstP P Fl Pkts Octets

0617.14:24:08.248 0617.14:25:08.280 58 18.7.24.66 50228 94 131.225.208.243 16490 6 0 2 110

0617.14:24:08.376 0617.14:24:58.360 58 18.7.24.66 50930 94 131.225.208.243 52433 6 0 2 110

0617.14:24:08.376 0617.14:25:08.408 58 76.13.15.30 5050 94 131.225.169.173 49545 6 0 2 104

0617.14:24:08.312 0617.14:25:08.344 58 18.7.24.66 37287 94 131.225.208.243 55432 6 0 2

The engine checks for the existence of directories associated with the IP address and hostname, prior to dealing with the included files. The engine throws an alert if neither of the directories exists. For each existing directory, the engine provides a list of included files with their status (file name, file size, last time it was modified and last time it was accessed). The code also throws an exception if the same file, in both locations, contains zero bytes. All the times, including the time variant, are converted to seconds before any comparison. For non-zero byte files, if the time variant is less than the difference between now and last time it was modified, then we know that file is updated within the given time variant. Otherwise, it is not updated within the given time variant.

## RESULTS

**Sample Results Written to a File as the Engine is Executed:**

**Example 1: Both Directories exist**

2009-07-10T15:45:12-05:00 Engine\_Front [debug] 131.225.189.82 is the source

2009-07-10T15:45:12-05:00 Engine\_Front [debug] 131.225.189.82 is an IP address

2009-07-10T15:45:12-05:00 Engine\_Front [debug] with a hostname cmsstor82

2009-07-10T15:45:12-05:00 Engine\_Plugin\_Test [debug] Notified of preDispatch

2009-07-10T15:45:12-05:00 Engine\_Front [debug] /logging/syslog-ng/131.225.189.82 is a directory

2009-07-10T15:45:12-05:00 Engine\_Front [debug] /logging/syslog-ng/cmsstor82 is a directory

2009-07-10T15:45:12-05:00 Engine\_Front [debug] Files in /logging/syslog-ng/131.225.189.82

2009-07-10T15:45:13-05:00 Engine\_Front [debug]

2009-07-10T15:45:13-05:00 Engine\_Front [debug] messages

2009-07-10T15:45:13-05:00 Engine\_Front [debug] 0 bytes

2009-07-10T15:45:13-05:00 Engine\_Front [debug]

2009-07-10T15:45:13-05:00 Engine\_Front [debug] secure

2009-07-10T15:45:13-05:00 Engine\_Front [debug] 0 bytes

2009-07-10T15:45:13-05:00 Engine\_Front [debug]

2009-07-10T15:45:13-05:00 Engine\_Front [debug] Files in /logging/syslog-ng/cmsstor82

2009-07-10T15:45:13-05:00 Engine\_Front [debug]

2009-07-10T15:45:13-05:00 Engine\_Front [debug] messages

2009-07-10T15:45:13-05:00 Engine\_Front [debug] 4742729 bytes

2009-07-10T15:45:13-05:00 Engine\_Front [debug] Last access July 05 2009 04:03:34.

2009-07-10T15:45:13-05:00 Engine\_Front [debug] was last modified July 10 2009 15:45:00.

2009-07-10T15:45:13-05:00 Engine\_Front [debug] is in variant

2009-07-10T15:45:13-05:00 Engine\_Front [debug]

2009-07-10T15:45:13-05:00 Engine\_Front [debug] secure

2009-07-10T15:45:13-05:00 Engine\_Front [debug] 2837807 bytes

2009-07-10T15:45:13-05:00 Engine\_Front [debug] Last access July 05 2009 04:03:59.

2009-07-10T15:45:13-05:00 Engine\_Front [debug] was last modified July 10 2009 15:45:00.

2009-07-10T15:45:13-05:00 Engine\_Front [debug] is in variant

**Example 2: One of the directories exist**

2009-07-10T15:45:14-05:00 Engine\_Front [debug] /logging/syslog-ng/131.225.219.144 is not a directory

2009-07-10T15:45:14-05:00 Engine\_Front [debug] /logging/syslog-ng/d0srv097 is a directory

2009-07-10T15:45:14-05:00 Engine\_Front [debug] Files in /logging/syslog-ng/d0srv097

2009-07-10T15:45:14-05:00 Engine\_Front [debug]

2009-07-10T15:45:14-05:00 Engine\_Front [debug] messages

2009-07-10T15:45:14-05:00 Engine\_Front [debug] 375015 bytes

2009-07-10T15:45:14-05:00 Engine\_Front [debug] Last access July 06 2009 15:03:54.

2009-07-10T15:45:14-05:00 Engine\_Front [debug] was last modified July 07 2009 14:47:09.

2009-07-10T15:45:14-05:00 Engine\_Front [debug] Not in the given time variant

2009-07-10T15:45:14-05:00 Engine\_Front [debug]

2009-07-10T15:45:14-05:00 Engine\_Front [debug] secure

2009-07-10T15:45:14-05:00 Engine\_Front [debug] 19707 bytes

2009-07-10T15:45:14-05:00 Engine\_Front [debug] Last access July 06 2009 15:03:54.

2009-07-10T15:45:14-05:00 Engine\_Front [debug] was last modified July 07 2009 10:33:42.

2009-07-10T15:45:14-05:00 Engine\_Front [debug] Not in the given time variant

**Example 3: Both Directories do not exist**

2009-07-10T15:45:11-05:00 Engine\_Front [debug] Notifying plugins of engine startup

2009-07-10T15:45:12-05:00 Engine\_Plugin\_Test [debug] Notified of dispatchLoopStartup

2009-07-10T15:45:12-05:00 Engine\_Front [debug] 131.225.208.243 is the source

2009-07-10T15:45:12-05:00 Engine\_Front [debug] 131.225.208.243 is an IP address

2009-07-10T15:45:12-05:00 Engine\_Front [debug] with a hostname fcdfgcb2

2009-07-10T15:45:12-05:00 Engine\_Plugin\_Test [debug] Notified of preDispatch

2009-07-10T15:45:12-05:00 Engine\_Front [debug] /logging/syslog-ng/131.225.208.243 is not a directory

2009-07-10T15:45:12-05:00 Engine\_Front [debug] Both Dirs don’t exist

2009-07-10T15:45:12-05:00 Engine\_Front [debug]

**Example 4: 0 bytes file under both directories**

2009-07-10T15:45:14-05:00 Engine\_Front [debug] 131.225.207.12 is the source

2009-07-10T15:45:14-05:00 Engine\_Front [debug] 131.225.207.12 is an IP address

2009-07-10T15:45:14-05:00 Engine\_Front [debug] with a hostname cmssrm

2009-07-10T15:45:14-05:00 Engine\_Plugin\_Test [debug] Notified of preDispatch

2009-07-10T15:45:14-05:00 Engine\_Front [debug] /logging/syslog-ng/131.225.207.12 is a directory

2009-07-10T15:45:14-05:00 Engine\_Front [debug] /logging/syslog-ng/cmssrm is a directory

2009-07-10T15:45:14-05:00 Engine\_Front [debug] Files in /logging/syslog-ng/131.225.207.12

2009-07-10T15:45:14-05:00 Engine\_Front [debug]

2009-07-10T15:45:14-05:00 Engine\_Front [debug] messages

2009-07-10T15:45:14-05:00 Engine\_Front [debug] 0 bytes

2009-07-10T15:45:14-05:00 Engine\_Front [debug] /logging/syslog-ng/cmssrm/messages : 0 bytes

2009-07-10T15:45:14-05:00 FlowEngine [err] Files contain zero bytes

## CONCLUSION

The first example above shows that the directories associated with both the IP address and hostname do exist. Both message and secure files are found at least in one of the directories and contain more than 0 bytes. The files have also been updated within the time given by the time variant variable and therefore, we can conclude that the destination with an IP 131.225.189.82 or host name of cmsstor82 was working just fine and is forwarding its log files to the central logging server. The other three examples demonstrate the opposite. In the second case, one of the directories exists and does contain non-zero bytes files. However, these files are not in the given time variant which means that the device has not been forwarding its log files to the central logging server during that period of time. In the third one, both directories are not found and an exception is thrown to notify the computer security team that this device was never sending its logs. The final example demonstrates a situation when the same file is checked to have 0 bytes in both of its locations. This depicts that the device in question was logging to the central server at one point in time, but has since stopped. If the device is no longer offering any services, this is not a problem. If it is still offering services though, the system must be reconfigured to forward its logs. In the second, third and fourth cases, the network team must be notified so that they will find out the problem with the device and avoid unexpected interruption of the regular activities at Fermilab.

In conclusion, I would like to mention that this project can be further enhanced. It could be linked to the e-mail server so that each alert is directed to a particular technician or team of workers. The precondition and post condition checks enhance reliability and efficiency of the engine in terms of fully offering the intended service. The codes for these checks are expected to be included in the future.

## ACKNOWLEDGEMENTS

This projected was implemented at Feynman Computing Center. I thank the Fermilab SIST committee for giving me the opportunity to participate in this program, chance to meet incredible people and have a great learning experience. I would like to extend my heartfelt gratitude to my supervisors, Joe Klemencic and Tim Rupp without whom I could not have learned so much about PHP, Linux and the Git Version Control System. I would also like to thank my mentor Elmie A Peoples-Evans, and David Peterson for their relentless effort to make my stay at the lab as comfortable as possible. Further thanks go to all those who made my experience at Fermilab a memorable one.

## REFERENCES

[1] “A Full Web Building Tutorials,” Available: <http://www.w3schools.com/>. [Accessed: Jun. 18, 2009]

[2] “git the fast version control system,” Available: <http://git-scm.com/> . [Accessed: July 10, 2009]

[3] Luke Welling and Laura Thomson, PHP and MYSQL Web Development, 3rd ed. Indianapolis: Sams Publishing, 2005.

[4] “PHP Function List,” Available: <http://www.php.net/quickref.php/> . [Accessed: Jun 25, 2009]

**[**5**] “Zend Framework,” Available:** [**http://framework.zend.com/**](http://framework.zend.com/)**. [**Accessed: **Jun. 18, 2009]**

## APPENDIX-1

This appendix provides detailed information about the engine, where all major tasks are initiated.

<?php

set\_time\_limit(0);

if (!defined('\_ABSPATH')) {

define('\_ABSPATH', dirname(\_\_FILE\_\_));

}

if (!defined('IDENT')) {

define('IDENT', 'FlowEngine');

}

require \_ABSPATH.'/lib/Autoload.php';

$log = App\_Log::getInstance(IDENT);

$config = Ini\_Config::getInstance();

$cg = new Zend\_Console\_Getopt(

array(

'help|h'=> 'Display this help and exit',

'flow-file|f=s' => 'Run against the provided flow file',

'output|o=s' => 'Output format to write results to',

'time|t=s' => "Alert if the machine hasn't reported in this much time",

'timeVar|v=s' => "time variant to chek the files for"

)

);

$variant=null;

$flowfile=null;

$includeTask = null;

$excludeTask = null;

$forbidden = array('Abstract', 'Broker', 'Exception');

try {

$opts = $cg->parse();

} catch (Zend\_Console\_Getopt\_Exception $e) {

usage($e);

exit;

}

if (isset($opts->h)) {

usage($opts);

exit;

}

if (isset($opts->f)) {

$flowfile =$opts->f;

}

if (isset($opts->o)) {

$output = $opts->o;

}

if (isset($opts->t)) {

$time = $opts->t;

}

if(isset($opts->v)) {

$variant = $opts->v;

}

if (is\_null($flowfile)) {

usage($opts);

exit;

}

$controller = Engine\_Front::getInstance();

**$controller->setFilename($flowfile);**

**$controller->setTimeVariant($variant);**

// Register maintenance plugins provided in the standard distribution

if (is\_dir($config->plugins->directory->system)) {

$dir = new DirectoryIterator($config->plugins->directory->system);

foreach($dir as $file ) {

if(!$file->isDot() && !$file->isDir()) {

$filename = $file->getPathname();

if (in\_array(basename($filename,'.php'), $forbidden)) {

continue;

}

registerClass($controller, $filename);

}

}

}

// Register the include path for site specific plugins

if (is\_dir($config->plugins->single->path)) {

appendToPath($config->plugins->single->path);

}

// Register site specific directory of plugins

if (is\_dir($config->plugins->directory->user)) {

$dir = new DirectoryIterator($config->plugins->directory->user);

foreach($dir as $file ) {

if(!$file->isDot() && !$file->isDir()) {

$filename = $file->getPathname();

if (in\_array(basename($filename,'.php'), $forbidden)) {

continue;

}

registerClass($controller, $filename);

}

}

}

try {

$single = $config->plugins->single->toArray();

// Register individual site specific plugins

if (!empty($single['register'])) {

if (is\_array($single['register'])) {

$tasks = $single['register'];

} else {

$tasks = array($single['register']);

}

foreach($tasks as $task) {

if ($controller->hasPlugin($task)) {

continue;

} else {

$controller->registerPlugin(new $task);

}

}

}

// Un-register any plugins that the site owner doesnt want to run

if (!empty($single['unregister'])) {

if (is\_array($single['unregister'])) {

$tasks = $single['unregister'];

} else {

$tasks = array($single['unregister']);

}

foreach($tasks as $task) {

if (!$controller->hasPlugin($task)) {

continue;

} else {

$controller->unregisterPlugin($task);

}

}

}

/\*\*

// Include tasks specified on the command line

if (!is\_null($includeTask)) {

$tasks = explode(',', $includeTask);

unset($includeTask);

foreach ($tasks as $task) {

if ($controller->hasPlugin($task)) {

continue;

} else {

$controller->registerPlugin(new $task);

}

}

}

// Exclude tasks specified on the command line

if (!is\_null($excludeTask)) {

$tasks = explode(',', $excludeTask);

unset($excludeTask);

foreach ($tasks as $task) {

if (!$controller->hasPlugin($task)) {

continue;

} else {

$controller->unregisterPlugin(new $task);

}

}

}

\*\*/

// Dispatch execution of all registered plugins

**$controller->dispatch();**

} catch (Exception $error) {

$log->log($error->getMessage(), App\_Log::ERR);

}

function usage($error) {

echo sprintf("\n%s\n", $error->getUsageMessage());

}

/\*\*

\* Code borrowed from

\*

\* http://stackoverflow.com/questions/928928/determining-what-classes-are-defined-in-a-php-class-file

\*/

function registerClass($controller, $filename) {

$php = file\_get\_contents($filename);

@$tokens = token\_get\_all($php);

$class\_token = false;

foreach ($tokens as $token) {

if (is\_array($token)) {

if ($token[0] == T\_CLASS) {

$class\_token = true;

} else if ($class\_token && $token[0] == T\_STRING) {

if (!$controller->hasPlugin($token[1])) {

$controller->registerPlugin(new $token[1]);

}

$class\_token = false;

}

}

}

}

?>

## APPENDIX-2

This appendix includes further details for function calls in the engine. It is illustrates how data abstraction was implemented in this project.

<?php

/\*\*

\*/

class Engine\_Front

{ const IDENT = \_\_CLASS\_\_;

/\*\*

\* Directory|ies where controllers are stored

\*

\* @var string|array

\*/

protected $\_controllerDir = null;

/\*\*

\*\*/

protected $\_flowfile =null;

protected $\_timev= null;

/\*\*

\* Singleton instance

\* Marked only as protected to allow extension of the class. To extend,

\* simply override {@link getInstance()}.

\* @var Engine\_Front

\*/

protected static $\_instance = null;

/\*\*

\* Instance of Engine\_Plugin\_Broker

\* @var Engine\_Plugin\_Broker

\*/

protected $\_plugins = null;

/\*\*

\* Constructor

\* Instantiate using {@link getInstance()}; front controller is a singleton

\* object.

\* Instantiates the plugin broker.

\* @return void

\*/

protected function \_\_construct() {

$this->\_plugins = new Engine\_Plugin\_Broker();

}

/\*\*

\* Enforce singleton; disallow cloning

\*

\* @return void

\*/

private function \_\_clone() {

}

/\*\*

\* Singleton instance

\* @return Engine\_Front

\*/

public static function getInstance() {

if (null === self::$\_instance) {

self::$\_instance = new self();

}

return self::$\_instance;

}

public function setFilename($file){

if (file\_exists($file)) {

$this->\_flowfile=$file;

}

}

public function filecheck($fileDir, $timeV,$otherDir) {

$log = App\_Log::getInstance(self::IDENT);

if (is\_dir($fileDir)) {

if ($handle2 = opendir($fileDir)) {

$log->log(sprintf("Files in %s\n", $fileDir),App\_Log::DEBUG);

while (false !== ($File = readdir($handle2))) {

if ($File == '.' || $File == '..') {

continue;

}

$log->log(sprintf("\n"),App\_Log::DEBUG);

$log->log(sprintf("%s ", $File),App\_Log::DEBUG);

$fileat=sprintf('%s/%s',$fileDir, $File);

$fSize=filesize($fileat);

$log->log(sprintf("%s bytes",$fSize),App\_Log::DEBUG);

if(filesize($fileat) == 0) {

if( is\_dir($otherDir)){

$fileat2=sprintf('%s/%s', $otherDir,$File);

$fSize2=filesize($fileat2);

if(filesize($fileat2)==0){

$log->log(sprintf("%s : 0 bytes", $fileat2 ),App\_Log::DEBUG);

throw new Exception("Files containt zero bytes");

}

}

else {

continue;

}

}

else {

$dateA= date("F d Y H:i:s.",fileatime($fileat));

$dateB=date ("F d Y H:i:s.", filemtime($fileat));

$log->log(sprintf("Last access %s",$dateA),App\_Log::DEBUG);

$DateA= date("F d Y H:i:s.",fileatime($fileat));

$log->log(sprintf("was last modified %s",$dateB),App\_Log::DEBUG);

$modtime = date(filemtime($fileat));

$timenow = time();

$unit = substr($timeV, -1);

$unit= strtolower($unit);

$inputvariant = substr($timeV, 0, -1);

#precondition: unit is a character

switch ($unit) {

case "h":

default:

$v = "-$inputvariant hours";

break;

case "d":

$v = "-$inputvariant days";

break;

case "m":

$v = "-$inputvariant months";

break;

case "y":

$v = "-$inputvariant years";

break;

}

$variant = strtotime($v,time());

if ($modtime >= $variant) {

$log->log(sprintf("is in variant\n"),App\_Log::DEBUG);

}

else {

$log->log(sprintf("Not in the given time variant\n"),App\_Log::DEBUG);

if(is\_dir($otherDir)){

$fileat3=sprintf('%s/%s', $otherDir,$File);

If (file\_exists($fileat3)){

$modtime = date(filemtime($fileat3));

if (!($modtime >= $variant)){

throw new Exception("files not in variant");

}

}

}

}

}

}

$log->log(sprintf("\n"),App\_Log::DEBUG);

}

else {

$log->log(sprintf("no files were found\n"),App\_Log::DEBUG);

}

}

else{

}

}

public function setTimeVariant($Variant) {

$this->\_timev=$Variant;

}

/\*\*

\* Resets all object properties of the singleton instance

\* Primarily used for testing; could be used to chain front controllers.

\* Also resets action helper broker, clearing all registered helpers.

\* @return void

\*/

public function resetInstance()

{

$reflection = new ReflectionObject($this);

foreach ($reflection->getProperties() as $property) {

$name = $property->getName();

switch ($name) {

case '\_instance':

break;

case '\_controllerDir':

case '\_invokeParams':

$this->{$name} = array();

break;

case '\_plugins':

$this->{$name} = new Engine\_Plugin\_Broker();

break;

case '\_throwExceptions':

case '\_returnResponse':

$this->{$name} = false;

break;

case '\_moduleControllerDirectoryName':

$this->{$name} = 'controllers';

break;

default:

$this->{$name} = null;

break;

}

}

Engine\_Action\_HelperBroker::resetHelpers();

}

/\*\*

\* Register a plugin.

\* @param Engine\_Plugin\_Abstract $plugin

\* @param int $stackIndex Optional; stack index for plugin

\* @return Engine\_Front

\*/

public function registerPlugin(Engine\_Plugin\_Abstract $plugin, $stackIndex = null)

{

$this->\_plugins->registerPlugin($plugin, $stackIndex);

return $this;

}

/\*\*

\* Unregister a plugin.

\* @param string|Engine\_Plugin\_Abstract $plugin Plugin class or object to unregister

\* @return Engine\_Front

\*/

public function unregisterPlugin($plugin)

{

$this->\_plugins->unregisterPlugin($plugin);

return $this;

}

/\*\*

\* Is a particular plugin registered?

\* @param string $class

\* @return bool

\*/

public function hasPlugin($class)

{

return $this->\_plugins->hasPlugin($class);

}

/\*\*

\* Retrieve a plugin or plugins by class

\* @param string $class

\* @return false|Engine\_Plugin\_Abstract|array

\*/

public function getPlugin($class)

{

return $this->\_plugins->getPlugin($class);

}

/\*\*

\* Retrieve all plugins

\*

\* @return array

\*/

public function getPlugins()

{

return $this->\_plugins->getPlugins();

}

/\*\*

\* Dispatch an HTTP request to a controller/action.

\* @param Engine\_Request\_Abstract|null $request

\* @param Engine\_Response\_Abstract|null $response

\* @return void|Engine\_Response\_Abstract Returns response object if returnResponse() is true

\*/

**public function dispatch(Engine\_Request\_Abstract $request = null, Engine\_Response\_Abstract $response = null)**

{

$log = App\_Log::getInstance(self::IDENT);

$this->\_request=new Engine\_Request\_Simple;

$handle = fopen( $this->\_flowfile, "r");

if (!is\_resource($handle)) {

$log->log('Flow file was not found', App\_Log::ERR);

exit;

}

$timeVariant= $this->\_timev;

// Begin dispatch

try {

$log->log('Notifying plugins of engine startup',App\_Log::DEBUG);

/\*\*

\* Notify plugins of dispatch loop startup

\*/

$this->\_plugins->dispatchLoopStartup($this->\_request);

/\*\*

\* Attempt to dispatch the controller/action. If the $this->\_request

\* indicates that it needs to be dispatched, move to the next

\* action in the request.

\*/

$this->\_request->setDispatched(true);

while(!feof($handle)){

$line = fgets($handle,4096);

$trimmed = trim($line);

$pos = strpos($trimmed, 'Start');

if((strpos($trimmed, 'Start') !== false) || empty($trimmed)){

continue;

}

$splits = explode(' ',$trimmed);

$counter=0;

foreach ($splits as $value) {

if ($value == '') {

} else {

$tempArr[$counter] = $value;

$counter++;

}

}

$source=$tempArr[6];

$log->log(sprintf("%s is the source",$source),App\_Log::DEBUG);

$srcsplt=explode(".",$source);

$count = count($srcsplt);

if (Ip:: isIpAddress($source)){

$IsIP = TRUE;

}

else {

$IsIP = FALSE;

}

if ($IsIP == TRUE) {

$ip = $source;

$log->log(sprintf("%s is an Ip address", $source),App\_Log::DEBUG);

$hostname\_1 = gethostbyaddr($source); // resolves the hostname

$splthst = explode(".",$hostname\_1); // cuts off the fnal.gov part of hostname

$hostname = $splthst[0];

$log->log(sprintf("with a hostname %s",$hostname),App\_Log::DEBUG);

}

else {

$hostname = $source;

$log->log(sprintf("%s is a hostnames", $source),App\_Log::DEBUG);

$ip = gethostbyname($source);

$log->log(sprintf("with an Ip %s",$ip),App\_Log::DEBUG);

}

$this->\_plugins->postNameResolution($this->\_request);

# postcondition: IP and Hostname are resolved

#precondition:IP and Hostname are resolved

$loggingDir = '/logging/syslog-ng';

$filenameIp = sprintf('%s/%s', $loggingDir, $ip);

$filenameHost = sprintf('%s/%s', $loggingDir, $hostname);

/\*\*

\* Notify plugins of dispatch startup

\*\*/

$this->\_plugins->preDispatch($this->\_request);

$this->\_plugins->preDirectoryCheck($this-> \_request);

if ( is\_dir($filenameIp)) {

$log->log(sprintf("%s is a directory\n",$filenameIp),App\_Log::DEBUG);

if (is\_dir($filenameHost)) {

$log->log(sprintf("%s is a directory\n",$filenameHost),App\_Log::DEBUG);

$this->\_plugins->preIpFileCheck($this-> \_request);

$this->filecheck($filenameIp,$timeVariant,$filenameHost );

$this->\_plugins->postIpFileCheck($this-> \_request);

$this->\_plugins->preHostFileCheck($this-> \_request);

$this->filecheck($filenameHost,$timeVariant,$filenameIp);

$this->\_plugins->postHostFileCheck($this-> \_request);

}

else {

$log->log(sprintf("%s is a directory\n",$filenameHost),App\_Log::DEBUG);

$this->\_plugins->preIpFileCheck($this-> \_request);

$this->filecheck($filenameIp,$timeVariant,$filenameHost );

$this->\_plugins->postIpFileCheck($this-> \_request);

}

}

else {

$log->log(sprintf("%s is not a directory\n", $filenameIp),App\_Log::DEBUG);

if (is\_dir($filenameHost)) {

$log->log(sprintf("%s is a directory\n",$filenameHost),App\_Log::DEBUG);

$this->\_plugins->preHostFileCheck($this-> \_request);

$this->filecheck($filenameHost,$timeVariant,$filenameIp);

$this->\_plugins->postHostFileCheck($this-> \_request);

}

else {

$log->log(sprintf("Both Dirs dont exist\n",$filenameHost),App\_Log::DEBUG);

$log->log(sprintf("\n"),App\_Log::DEBUG);

continue;

}

}

$this->\_plugins->postDirectoryCheck($this-> \_request);

try {

$this->\_plugins->dispatch($this->\_request);

} catch (Exception $e) {

throw $e;

}

/\*\*

\* Notify plugins of dispatch completion

\*/

$this->\_plugins->postDispatch($this->\_request);

}

fclose($\_flowfile);

} catch (Exception $e) {

if (is\_resource($handle)) {

closedir($handle);

}

throw $e;

}

/\*\*

\* Notify plugins of dispatch loop completion

\*/

try {

$this->\_plugins->dispatchLoopShutdown();

}catch (Exception $e) {

throw $e;

}

shutdown',App\_Log::DEBUG);

if (is\_resource($handle)) {

closedir($handle);

}

}

}

?>