Other API Documentation

Overview

This section contains information common to all GroundWork RAPID-based Feeders.

CONTENTS

RELATED RESOURCES

- Developer Reference

WAS THIS PAGE HELPFUL?

- Leave Feedback

1.0 About Perl API Monarch

dassmonarch is a Perl API to GroundWork Monitor. It provides an API to program configuration changes of the GroundWork Monitor system. The Perl API is published under GPL and is free for download. Almost the same functionality is available as an additional SOAP interface, which lets you access and modify the GroundWork configuration by a remote SOAP client. The SOAP interface is available from dass IT under a commercial license. Thanks to dass-IT - Maik Aussendorf for contributing the Perl configuration API to Monarch.

1.1 Classes and Methods

Following is a complete reference of the *GWHelper Class* and the *dassmonarch Class*. Additionally you can view a simple example application for dassmonarch and extended information for externals routine calling sequences.

GWHelper Class Reference

This class provides methods related to the installed GroundWork system like determining GroundWork's release version, etc. This version is for use with GW Monitor 6.2 and later releases ONLY.

- GWHelper Class Reference - Select this link for a list of members and links to function documentation.

dassmonarch Class Reference

This class provides relevant methods to access the monarch DB in order to automatically import configuration data.

- dassmonarch Class Reference - This reference provides a list of members and links to function documentation which describes in detail every method including parameters and return values.
- dassmonarch Externals Routine Calling Sequences - Additionally, here you can access a more detailed look at externals routine information.

1.2 dassmonarch Perl API Example

This link displays a simple example application for dassmonarch Perl API to GroundWork Monarch. This is not for the SOAP interface, only for the local class.

- example.dassmonarch.pl

1.3 Standalone Monarch configuration scripts

Along with the extensions to dassmonarch for the 7.2.1 release, there are standalone scripts that exercise certain parts of dassmonarch functionality. This means that for certain common simple tasks, an administrator need not dive down to the dassmonarch level to get these tasks done from the command line. The names of the scripts are somewhat descriptive however further documentation can be obtained by listing out the help for each as shown below.

List of scripts

The scripts all live in the `/usr/local/groundwork/core/monarch/bin` directory.

```
monarch_assign_contact_to_contactgroup
monarch_assign_contactgroup_to_hostgroup
monarch_assign_contactgroup_to_service
monarch_assign_external_to_hostprofile
monarch_assign_service_to_host
monarch_assign_service_to_serviceprofile
monarch_assign_serviceprofile_to_host
monarch_assign_servicetemplate_to_service
monarch_clone_hostprofile
monarch_clone_servicetemplate
monarch_create_contact
monarch_create_contactgroup
monarch_create_servicegroup
monarch_create_serviceprofile
monarch_delete_all_hosts_in_hostgroup
monarch_delete_contact
monarch_delete_contactgroup
monarch_delete_host
monarch_delete_hostprofile
monarch_delete_service
monarch_disable_host_service_freshness_checks
monarch_enable_host_service_freshness_checks
monarch_get_hostlist
monarch_remove_contactgroup_from_service
monarch_remove_external_from_hostprofile
monarch_remove_service_from_host
monarch_set_host_service_freshness_threshold
monarch_set_host_service_notification_interval
monarch_set_service_command
monarch_set_service_commandline
```

Obtaining simple documentation

There are two means to obtain simple documentation on each of these scripts. The first is to run `perldoc` on the script; this will produce documentation in the style of a man page. For example:

```
cd /usr/local/groundwork/core/monarch/bin
/usr/local/groundwork/perl/bin/perldoc monarch_get_hostlist
```

Very terse documentation is also available by running the script with the `-h` option. This mostly just shows the available command-line options, with little explanation. For example:

```
cd /usr/local/groundwork/core/monarch/bin
monarch_clone_hostprofile -h
```

Additional scripts

In addition to those scripts, which are directly associated with the dassmonarch package, we have also added two other scripts:

`monarch_assign_contactgroups_to_hostgroup_hosts_and_host_services`
This can be used to manage contact groups at the hostgroup level and can save you from having to crawl through a lot of screens to manually do a lot of configuration work. Currently, there is no perldoc documentation for this script, but the `-h` option will spill out a long usage message, which currently reads like this:

```
# cd /usr/local/groundwork/core/monarch/bin
# monarch_assign_contactgroups_to_hostgroup_hosts_and_host_services -h

This tool is used to apply a set of contactgroups to all hosts associated
with a specified hostgroup.  All the services on those hosts will have
the exact same contactgroups also applied.  Existing contact groups on
the hosts and host services will be left intact unless an option is
specified to request that specified or all existing contactgroups on
those hosts and host services be cleared.

usage:  monarch_assign_contactgroups_to_hostgroup_hosts_and_host_services -l hostgroup
or:     monarch_assign_contactgroups_to_hostgroup_hosts_and_host_services {-a|-r} hostgroup
contactgroup ...
or:     monarch_assign_contactgroups_to_hostgroup_hosts_and_host_services -c hostgroup
where:  -l means list out all the hosts and host services associated with the hostgroup
        -a means add the named contactgroups to all the hosts associated
           with the hostgroup and the host services attached to those hosts
        -r means remove the named contactgroups from those hosts and host services
        -c means clear all contactgroups from those hosts and host services
```

`monarch_restore_from_backup`
This is referenced by the new Backup code within the Monarch UI. This script is not run automatically from the UI (Configuration > Control > Backup and restore). Instead, when you request the UI to restore a particular backup, it instead composes the full invocation of this script that you will need to run from the command line, and displays that command in the browser. Documentation is available in both `perldoc` and `-h` forms.

2.0 About Perl API

2.1 Classes and Methods

The Perl API is a module called CollageQuery which allows a Perl program to retrieve data from Collage. Also see the Perl API example using CollageQuery module. The following classes and methods are available for the Perl API.

CollageQuery

- `new` - Create the CollageQuery object. Required to use any of the following methods.
- `destroy` - Destroys the CollageQuery object. Should be called when the CollageQuery object is no longer needed.

CollageHostGroupQuery

- `getServicesForHostGroup(String hostGroup)` - Returns a reference to a hash host-service-attributes for a designated Host Group.
- `getHostsForHostGroup(String hostGroup)` - Returns a reference to a hash of all Host names, device names for a designated Host Group.
- `getHostGroups()` - Returns a reference to all Host Group names, descriptions.
- `getHostGroup(String hgName)` - Returns a hash containing the attributes for a Host Group.

CollageHostQuery

- `getServicesForHost(String host)` - Returns a reference to a hash of all services-attributes for a Host.
- `getHosts()` - Returns a reference to a hash of all Host-attributes.
- `getHostStatusForHost(String host)` - Returns a hash of all the status attributes for a Host.
- `getDeviceForHost(String host)` - Returns a hash of the device attributes for a Host.

CollageServiceQuery

- `getService(String serviceName, String hostName)` - Returns a hash of service attributes.
- `getServices()` - Returns a reference to a hash of host-service-attributes.

CollageMonitorServerQuery

- `getMonitorServers` - Returns a reference to a hash of monitorserver-attributes.
- `getHostsForMonitorServer(String MonitorServer)` - Returns a reference to a hash of Hosts for a designated monitorserver.
- `getHostGroupsForMonitorServer(String MonitorServer)` - Returns a reference to a hash of host groups-attributes.

CollageEventQuery

These methods return a reference to a hash of events with the event ID as primary key and attributes as secondary key.

⚠ timeField (String) can be "FirstInsertDate" or "LastInsertDate;" if it's NULL, no range will be applied.

- `getEventsForDevice(String identification, String timeField, Date fromDate, Date toDate)`
- `getEventsForService(String serviceDescription, String HostName, String timeField, Date fromDate, Date toDate)`
- `getEventsForHost(String HostName, String timeField, Date fromDate, Date toDate)`

2.2 Perl API Examples

The following code is a Perl API example using CollageQuery module.

```
# Use the CollageQuery module use CollageQuery ;
# Declare the CollageQuery object my $t;
# Create an object and make sure we can connect to Collage
$t=CollageQuery->new() or die "Error: connect to CollageQueryfailed\n";
# Print the attributes for a host group "demo-systems"
$getparam = "demo-systems";
# getHostGroup returns a hash of attributes and values
my %hash = $t->getHostGroup($getparam);
foreach my $key (sort keys %hash) {
print "\t$key=$hash{$key}\n";
}
# Now print the services and attributes for a host
$getparam = "demo-host";
# getServicesForHost returns a reference to a hash of service hashes
my $ref = $t->getServicesForHost($getparam);
# Iterate for each service
foreach my $service (sort keys %{$ref}) {
print "\tService=$service\n";
# Now iterate for each attribute.
foreach my $attribute (sort keys %{$ref->{$service}}) {
# print the attribute and attribute value
print "\t\t$attribute=".$ref->{$service}->{$attribute}."\n";
}
# Finished so destroy the object
$t->destroy();
```

The examples that use the Perl CollageQuery API are as follows. These files can be found at this link Perl Examples.

- `test1.pl` - This script exercises each method in the API. It is invoked via command line and returns the result to STDOUT.
- `api_sample2.pl` - This is a CGI program that allows you to select the Collage class and method to test. The drop down selection lists use the API to give you options from the Foundation data store. The form will show the results of the query.
- `api_sample3.pl` - This is a modified version of api_sample2.pl that outputs data in a more user-friendly form.

3.0 About PHP Foundation API

GroundWork Monitor includes an API called PHP Foundation API for PHP applications that internally uses the Foundation Web Services to query for monitoring data. In previous versions of GroundWork Monitor the API called, CollageDB API, is marked as deprecated and any new development should use the new PHP Foundation API.

The PHP Foundation API is a wrapper around a soap client that communicates to the Web Service Interface of GroundWork Foundation. With this architecture change applications written in PHP using the PHP Foundation API will be more scalable and perform better than with the previous API. Better performance was achieved by using the Web Service Interface that uses optimized queries and a query cache.

The PHP Foundation API requires GroundWork Foundation to be installed and accessible. The PHP wrapper classes (DAL) reside in the directory:

```
/usr/local/groundwork/core/foundation/api/php
```

3.1 PHP Foundation API Example

The /usr/local/groundwork/nagios/libexec/check_host_foundation.php script provides a simple example of the use of this API.

The shebang line (#!) in the example below mentions php/bin/php.bin instead of php/bin/php as you would normally expect. This is because of a known problem. Contact GroundWork Support for additional information.

```
#!/usr/local/groundwork/php/bin/php.bin
<?php
# This is a Nagios plugin that checks the status of all services on a host.
# It is intended to be used to provide an erzatz host check in the case where
# passive service checks are all that is available.
# Return values are UP if at least one service is in OK state,
# DOWN if all services are not OK.
set_include_path('/usr/local/groundwork/core/foundation/api/php/');
require_once('DAL/ServiceDAL.inc.php');
# Would be nice to use this, but it's broken at the moment in PHP
#require_once('DAL/StatisticsDAL.inc.php');

# Handle Arguments
$args = "";
$args .= "h:";
$args .= "d::";

$options = getopt($args);

if (!$options['h']) {
 print "Host name required! Use -h hostname\n";
 exit (1);
} else {
 $host2query = $options['h'];
}
if ($options['d']) {
 print "Debug set with -d. We will output any errors and all data. Do not use with Nagios in this
mode.\n";
 $debug_exceptions = 1;
 $debug_service_results = 1;
} else {
 $debug_exceptions = 0;
        $debug_service_results = 0;
}

$webservicesURL = 'http://localhost:8080/foundation-webapp/services';

try {
# $serviceDAL  = new StatisticsDAL($webservicesURL);
  $serviceDAL  = new ServiceDAL($webservicesURL);
 $results = $serviceDAL->getServicesByHostName($host2query);
#$results = $serviceDAL->getServiceStatisticsByHostName($host2query);
} catch (DALException $dalEx) {
    if ($debug_exceptions) {
        print "Caught exception:  ".$dalEx->getMessage()."\n";
    }
    print "ERROR:  Could not contact Foundation!\n";
    exit (1);
}
# Print debug data
if ($debug_service_results) {
    $returnArray = $results['Services'];
    if(count($returnArray)) {
        foreach ($returnArray as $service) {
     $returnDesc = $service['Description'];
            $returnState = $service['MonitorStatus'];
            print "$returnDesc = " . $returnState->Name . "\n";
        }
    }
}
```

```
# Set up output
$returnArray = $results['Services'];
if(count($returnArray)) {
    foreach ($returnArray as $service) {
        $returnState = $service['MonitorStatus']->Name;
 $result = preg_match('/OK/', $returnState);
 if ($result == 1) { # Nothing more to do - host is UP
     print "At least one service is OK. Host considered UP\n";
            exit (0);
        }
    }
 print "All services in non-OK state! Host considered DOWN\n";
 exit (2);
} else { # No services to check
 print "This host has no services defined. Setting status to UP.\n";
 exit (0);
}
?>
```

To use this API, you must provide the URL to the Foundation Web Service. This URL is passed to the constructor of each DAL object when it is created. It will look something like:

```
http://"serverIP":8080/foundation-webapp/services
```

If you are writing an application that is part of Groundwork Monitor, you can use the global variable `$foundationModule` and call the `getWebServiceURL()` method on it.

3.2 PHP Foundation API Documentation

Select the link PHP Foundation API to view the documentation.

4.0 About Web Services API

Adding a Web Service layer enables more applications to use and integrate with the existing Foundation Framework. The Web Service API uses SOAP (Simple Object Access Protocol) as the communication protocol. Since Web Services are a well defined and a widely used standard other technologies such as .NET or popular development tools such as Visual Studio or Java Studio Creator can be used to create UI or business components on the top of the Foundation platform.

The Web Services framework is able to scale and distribute the API more easily, than the previous API, which results in higher throughput and therefore better overall performance of large installations. The BIRT reporting feature uses the Foundation Web Service API to access data stored in the Foundation persistent store.

4.1 GroundWork Web Services

The Web Service API uses SOAP (Simple Object Access Protocol) as the communication protocol. Applications consume Web Service Description Language (WSDL) files in order to view and retrieve data from the endpoint (Foundation). SOAP requests are based on the WSDL file. In order to consume the Web Service, the SOAP library needs to know the following URLs:

- WSDL file location
- Schema file location
- SOAP end point

WSDL File Location (http://)

- `localhost:8080/foundation-webapp/services/wscommon?wsdl`
- `localhost:8080/foundation-webapp/services/wscategory?wsdl`
- `localhost:8080/foundation-webapp/services/wsevent?wsdl`
- `localhost:8080/foundation-webapp/services/wsdevice?wsdl`
- `localhost:8080/foundation-webapp/services/wshost?wsdl`
- `localhost:8080/foundation-webapp/services/wshostgroup?wsdl`
- `localhost:8080/foundation-webapp/services/wsservice?wsdl`
- `localhost:8080/foundation-webapp/services/wsstatistics?wsdl`
- `localhost:8080/foundation-webapp/services/wsrrd?wsdl`

Schema File Location (http://)

- `www.w3.org/2001/XMLSchema`

SOAP End Point(http://)

- `localhost:8080/foundation-webapp/services/wscommon`
- `localhost:8080/foundation-webapp/services/wscategory`
- `localhost:8080/foundation-webapp/services/wsevent`
- `localhost:8080/foundation-webapp/services/wsdevice`
- `localhost:8080/foundation-webapp/services/wshost`
- `localhost:8080/foundation-webapp/services/wshostgroup`
- `localhost:8080/foundation-webapp/services/wsservice`
- `localhost:8080/foundation-webapp/services/wsstatistics`
- `localhost:8080/foundation-webapp/services/wsrrd`

The Foundation Web Service API provides the web services listed in the table below. Web Service Description Language (WSDL) for GroundWork can be consumed by applications to view and retrieve data from the endpoint (Foundation). Select the .wsdl link in the last column of the table below to display its contents in a .pdf file.

Table: GroundWork Web Services

Web Service	Description	WSDL
Category	Categories are used to group any entities (Service, Host, Log Message, devices) and build a hierarchy. The web service allows creating, maintaining, and deleting Categories (grouping) but not the entities. The current version of GroundWork uses Categories to manage Service Groups.	fwscategory.wsdl
Common	Web Service API to retrieve meta data for Application Types, Statuses, etc. and to access the Action API used for custom actions from the GroundWork Console.	fwscommon.wsdl
Device	Web Service to maintain device entries in the foundation database. Devices are the addressable base entities for monitoring.	fwsdevice.wsdl
Event	Event Web Service is an API that manages LogMessage stored in Foundation.	fwsevent.wsdl
Host	Host Web Service is an API that manages Host entities in Foundation.	fwshost.wsdl
HostGroup	Host Web Service is an API that manages HostGroup entries in Foundation. HostGroup is a collection of Host entities.	fwshostgroup.wsdl
Model	Defines the basic types and objects used by the Foundation Web Service API.	fwsmodel.wsdl
RRD	Web service that retrieves images (PNG) for a set of RRD files. Calling the web service with just the Host argument will return an array of graphs that represent all services for that host. By providing the Host name/Service description, only the specific graph will be returned.	fwsrrd.wsdl
Service	Service web service is an API that manages all ServiceStatus objects in Foundation.	fwsservice.wsdl
Statistics	At runtime, Foundation generates statistics about the current state of Host/HostGroup and Service/ServiceGroup data. The web service has several methods to access this runtime data.	fwsstatistics.wsdl

Web Service - Methods

- `getServiceAvailabilityForHostgroup ( hostGroupName )`
  Input: hostGroupName = String that defines the HostGroup name for which the Services availability percentage should be retrieved.
  Return: Service availability as a double which represents the percentage of Services in the specified Hostgroup that are in Status OK

- `getHostAvailabilityForHostgroup ( hostGroupName )`
  Input: hostGroupName = String that defines the HostGroup name for which the Host availability percentage should be retrived.
  Return: Host availability as a double which represents the percentage of Hosts in the specified Hostgroup that are in Status UP.