NerveCenter 6.1 MIB Perl

White Paper

February 2014

NCRN61-MP-01

Copyright

Disclaimers

LogMatrix, Inc. ("LogMatrix") makes no representations or warranties, either expressed or implied, by or with respect to anything in this manual, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose or for any indirect, special or consequential damages.

These applications are available through separate, individual licenses. Not every feature or application described herein is licensed to every customer. Please contact LogMatrix if you have licensing questions.

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, photocopying, recording or otherwise, without prior written consent of LogMatrix. While every precaution has been taken in the preparation of this book, LogMatrix assumes no responsibility for errors or omissions. This publication and the features described herein are subject to change without notice.

The program and information contained herein are licensed only pursuant to a license agreement that contains use, reverse engineering, disclosure and other restrictions.

Trademarks

LogMatrix is registered in the U.S. Patent and Trademark Office. NerveCenter and the LogMatrix Logo are trademarks of LogMatrix, Inc.

All other products or services mentioned in this manual may be covered by the trademarks, service marks, or product names as designated by the companies who market those products.

Contacting LogMatrix

LogMatrix, Inc.
2 Mount Royal Ave, Suite 250
Marlborough, MA 01752

Phone 508-597-5300
Fax 774-348-4953

contact email: info@logmatrix.com

Website: www.logmatrix.com
Forum: http://community.logmatrix.com/LogMatrix
Blog: www.logmatrix.com/blog

NerveCenter 6.1 MIB Perl

NC6.1 adds a Perl module NC::MIB that allows user access to NerveCenter Server's MIB. The prepared MIB, file /opt/OSInc/mibs/nervectr.mib, contains a large set of static information. Customers that wish to access this data, now can. For example, when looking at a retrieved value for an enumerated type (such as ifTable's ifType, ifAdminStatus or ifOperStatus), they'd like to be able to retrieve the associate label.

This document goes over each of the functions provided by the NC:MIB perl module.

Contents

NOTE: For all cases where your Perl code contains a named Base Object plus an Attribute, as for example in "my $oid = NC::MIB::get_oid("ifEntry\.ifOperStatus");", be sure to follow the syntax just shown. What you mean to do is pass ifEntry.ifOperStatus as a literal string value into the function. However to do so you must satisfy two hurdles. First, you must avoid NerveCenter's run-time replacement logic. Normally when specifying something like ifEntry.ifOperStatus , you want your logic to see the *returned value* for this SNMP entity at this point in your code. To avoid NerveCenter from doing this, precede the dot with a backslash. Thus ifEntry\.ifOperStatus . If you fail to do this, you'll be passing the returned value into the function - which is not your intention. Second, you need to state this in manner acceptable to Perl. String literals need to be surrounded by quotation marks - either single or double. Since you are augmenting the dot with a backslash, use the double quote mark. Thus **"**ifEntry**\.**ifOperStatus**"** is what you need to state.

Getting the syntax right
my $oid = NC::MIB::get_oid("ifEntry\.ifOperStatus"); This works. You will get back the OID for this MIB entity.
my $oid = NC::MIB::get_oid("ifEntry.ifOperStatus"); This does not work because a '\' is not used before the '.' NerveCenter will replace your the named MIB entity with the value from the Agent. Your function would end up being called with the *value of* this object, which in this case would be an integer.
my $oid = NC::MIB::get_oid(ifEntry\.ifOperStatus); This does not work because the surrounding double quotes are missing. The backslash before the dot avoids having NerveCenter spot and replace the name. However Perl will not agree with ifEntry.ifOperStatus and throw a syntax error.

1. NC::MIB::get_name($oid)

The function get_name() separates a given OID into its Base Object, Attribute and Instance components. There are many ways to express an OID. However, given NerveCenter's view of MIB Entities, which operates on the viewpoint of a Base Object with a set of Attributes hanging off of it, this function steers the user in this direction. The call can be used as such:

```
my $name = NC::MIB::get_name( $oid );
my ( $baseobject, $attribute, $instance ) = split( '\.', $name, 3 );
```

The function takes a single argument, a Perl scalar value, which needs to be an OID. (ex: 1.3.6.1.2.1.1) The OID may have a leading dot (ex: .1.3.6) or a trailing dot (ex: 1.3.6.) but must otherwise contain only numeric digits and the dot separators.

Sample values for $oid	
Valid OIDs	**Invalid OIDs**
1	.
.1	..1
1.	1..
1.3	system
1.3.6	1.3.6.1.2.1.1.sysDescr
1.3.6.1.4.1.1043	1.3.dod.1
1.3.6.111.333.223333.4444444	dod.1.4.1.cisco

The function returns a dot-separated evaluation of the input $oid. The first value will be the Base Object, if one can be determined, followed by the Attribute name and then by any trailing Instance values.

If the function is given an invalid OID, then it returns an empty value.

This table shows returns for a set of valid OIDs. This is a logical representation of the results when using the above code sample - including the use of split() to separate the return into its components.

Input	Output	split(\. , $name, 3)			Description
$oid	$name	$baseobject	$attribute	$instance	
1.3.6	dod	dod			Three examples showing the resolution of an OID that maps directly to either a Base Object or else an interior node within the overall MIB structure.
1.3.6.1.2.1.1	system	system			
1.3.6.1.2.1.31.1.1.1	ifXEntry	ifXEntry			
1.3.6.1.2.1.1.6	system.sysLocation	system	location		Examples showing the resolution of OIDs that map to a Base Object plus an Attribute.
1.3.6.1.2.1.31.1.1.1.18	ifXEntry.ifAlias	ifXEntry	ifAlias		
1.3.6.1.2.1.1.6.0	system.sysLocation.0	system	location	0	Examples showing the resolution of OIDs, as found in responses from an SNMP Agent when a scalar has been polled. The Agent uses a '.0' at the end of the scalar's OID.
1.3.6.1.2.1.2.1.0	interfaces.ifNumber.0	interfaces	ifNumber	0	
1.3.6.1.2.1.7.5.1.127.0.0.1.8080	udpEntry.udpLocalPort.127.0.0.1.8080	udpEntry	udpLocalPort	127.0.0.1.8080	Examples showing the resolution of OIDs where a cell within a table is identified.
1.3.6.1.2.1.31.1.1.1.18.33	ifXEntry.ifAlias.33	ifXEntry	ifAlias	33	

2. NC::MIB::get_oid($name)

The function get_oid() is the compliment of get_name() . Given a *baseobject.attribute* name, the function returns the assigned OID.

The input to get_oid() needs to be a Perl scalar that identifies a base object and an attribute. Thus "system.sysName" is valid, but neither "system" or "sysName" is valid.

If the value provided in $name does not match anything in the MIB, then an empty value is returned.

Sample usage for
$oid = NC::MIB::get_oid($name)

Input	Output
$name	$oid
ifXEntry.ifHCInOctets	1.3.6.1.2.1.31.1.1.1.6
system.sysDescr	1.3.6.1.2.1.1.1

3. NC::MIB::get_type($name)

The function get_type() returns the SNMP Data Type for a known $name. In the MIB each attribute entity is given a type. The range of possible types is

- Counter
- Counter32
- Counter64
- Gauge
- INTEGER
- IpAddress
- ObjectID
- OctetString
- TimeTicks

The input to get_type() is the same as with get_oid() .

As with get_oid(), if the value provided in $name does not match anything in the MIB, then an empty value is returned.

Sample usage for
$type = NC::MIB::get_oid($name)

Input	Output
$name	$type
hostEntry.hostInPkts	Counter
snmp.snmpInPkts	Counter32
ifXEntry.ifHCInOctets	Counter64
nlHostEntry.nlHostInPkts	Gauge
system.sysServices	INTEGER
tcpConnEntry.tcpConnLocalAddress	IpAddress
system.sysObjectID	ObjectID
system.sysDescr	OctetString
system.sysUpTime	TimeTicks

4. NC::MIB::get_enum_value($name, $label)

The function get_enum_value() returns the numeric value assigned to a label. This applies to MIB entries which define an enumerated range, such as ifType and ifAdminStatus.

The input to get_enum_value() is both a $name and a $label. The $name is the same as from get_oid() . The $label is a word named as part of the enumerated range.

As with get_oid(), if the value provided in $name does not match anything in the MIB, then a zero is returned.

If the provided $name does not name an entity that specifies an enumeration, then a zero is returned.

If the provided $label is not known to the enumerated range, then a zero is returned.

Sample usage for
$value = NC::MIB::get_enum_value($name, $label)

Input		Output	*from* IF-MIB
$name	$label	$name	

ifEntry.ifOperStatus	up	1	ifOperStatus
			OBJECT-TYPE SYNTAX INTEGER {
			up(1), -- ready to pass packets
			down(2),
			testing(3), -- in some test mode
			unknown(4), -- status can not be determined
			-- for some reason.
			dormant(5),
			notPresent(6), -- some component is missing
			lowerLayerDown(7) -- down due to state of
			-- lower-layer interface(s)
			}

ifEntry.ifOperStatus	down	2	
ifEntry.ifOperStatus	testing	3	
ifEntry.ifOperStatus	monkey	0	

5. NC::MIB::get_enum_label($name, $value)

The function get_enum_label() is the compliment of get_enum_value() . This function returns the label assigned to the numeric value for an enumerated set.

The input to get_enum_value() is both a $name and a $value. The $name is the same as from get_oid() . The $value is an integer.

As with get_oid(), if the value provided in $name does not match anything in the MIB, then an empty value is returned.

If the provided $name does not name an entity that specifies an enumeration, then an empty value is returned.

If the provided $value is not known to the enumerated range, then an empty value is returned.

Sample usage for
$label = NC::MIB::get_enum_value($name, $value)

Input		Output	*from* IF-MIB
$name	$value	$label	

ifEntry.ifOperStatus	1	up	ifOperStatus
			OBJECT-TYPE SYNTAX INTEGER {
			up(1), -- ready to pass packets
			down(2),
			testing(3), -- in some test mode
			unknown(4), -- status can not be determined
			-- for some reason.
			dormant(5),
			notPresent(6), -- some component is missing
			lowerLayerDown(7) -- down due to state of
			-- lower-layer interface(s)
			}

ifEntry.ifOperStatus	2	down	
ifEntry.ifOperStatus	3	testing	
ifEntry.ifOperStatus	10		

LogMatrix Technical Support

LogMatrix is committed to offering the industry's best technical support to our customers and partners. You can quickly and easily obtain support for our NerveCenter proactive IT management software.

Professional Services

LogMatrix offers professional services, when customization of our software is the best solution for a customer. These services enable us, in collaboration with our partners, to focus on technology, staffing, and business processes as we address a specific need.

Educational Services

LogMatrix is committed to providing ongoing education and training in the use of our products. Through a combined set of resources, we can offer quality classroom style or tailored on-site training.

Contacting the Customer Support Center

Telephone Support

Phone: 1-800-892-3646 or 1-508-597-5300

E-Email support

E-mail: techsupport@logmatrix.com.

Online Access

For additional NerveCenter support information, please go the LogMatrix website www.logmatrix.com for access to the following sections of information.

Software Alerts – latest software alerts relative to NerveCenter.

User Community Access

You can seek as well as share advice and tips with other NerveCenter users at

http://community.logmatrix.com/LogMatrix/ .

Contact Support@logmatrix.com for:

- Patches and Updates – latest installation files, patches and updates including documentation for NerveCenter.