SNMP Dynamic Application Development

SL1 version 12.2.0

Table of Contents

Chapter 1

Introduction to SNMP Dynamic Applications

Overview

This chapter describes Simple Network Management Protocol (SNMP), the Internet protocol for managing devices across a network. In SL1, an SNMP Dynamic Application is a Dynamic Application that uses SNMP to retrieve data from devices.

Use the following menu options to navigate the SL1 user interface:

- To view a pop-out list of menu options, click the menu icon (☰).
- To view a page containing all of the menu options, click the Advanced menu icon (···).

This chapter covers the following topics:

What is SNMP?

SNMP is a protocol that is supported by most enterprise-level network equipment and most server equipment as well as by many software applications.

SNMP allows SL1 to query and collect information about a device or application. SNMP ensures that SL1 can use the same parameters to query any type of device or application. SNMP also ensures that the data sent to SL1 will be of the same format, regardless of the device or application that returns the data.

Many third-party hardware agents are SNMP-compliant. These agents complement standard SNMP agents by providing additional data about the device.

If SNMP is enabled on a device and a third-party agent is also running on a device, SL1 can gather very detailed data about the device or application.

Prerequisites

This manual does not describe how to plan, design, use, and troubleshoot Dynamic Applications for your network. This manual assumes that you are already familiar with the common elements and concepts of Dynamic Applications. For general information on planning, designing, using, and troubleshooting Dynamic Applications, see the manual ***Dynamic Application Development***.

SNMP Dynamic Applications use the SNMP protocol. This manual assumes that you are familiar with the SNMP protocol.

What is an SNMP Dynamic Application?

Simple Network Management Protocol, or SNMP, is the Internet protocol for managing devices across a network. In SL1, an SNMP Dynamic Application is a Dynamic Application that uses SNMP to retrieve data from devices. During collection, SL1 performs an SNMP walk command for each OID specified in the Dynamic Application. The Dynamic Application developer defines the SNMP OIDs in collection objects.

SNMP Dynamic Applications have the following elements in common with other Dynamic Application types:

- ***Archetypes***. Defines what type of data is being collected and how it will be displayed in SL1. SNMP Dynamic Applications can be either the Performance or Configuration archetypes.
- ***Properties***. Allows for version control, release notes, collection, and retention settings.
- ***Collection Objects***. Defines the individual data-points that will be retrieved by the Dynamic Application. These data points are called collection objects. Collection objects define what type of data is being collected (gauge, counter, etc.) and how it is grouped. Collection objects have settings that are different from collection objects in other types of Dynamic Applications. For more information, see ***SNMP Collection Objects***.
- ***Presentations***. For Performance Dynamic Applications, defines how collected values will be displayed by SL1.

- ***Thresholds***. Can be used to define a default threshold value that can be included in alerts. The threshold appears in the **Device Thresholds** page for each device the Dynamic Application is aligned with.
- ***Alerts***. Triggers events based on the values retrieved by the Dynamic Application. If the collected data meets the conditions defined in the alert, the alert can insert a message into device logs and trigger events.
- ***Credentials***. Define how authentication should occur for each Dynamic Application on each device. SNMP Dynamic Applications use SNMP credentials. When an SNMP Dynamic Application is aligned with a device, SL1 will, by default, use the SNMP Read credential defined in the **Device Properties** page for the device to perform collection for the Dynamic Application. You can select a different SNMP credential to use with the Dynamic Application in the **[Collections]** tab for a device.
- ***Relationships***. Dynamic Applications can be configured to automatically create relationships between devices. For example, the Dynamic Applications in the VMware vSphere and NetApp PowerPacks are configured to create relationships between VMware Datastore component devices and their associated NetApp Volume component devices. Relationships created by Dynamic Applications are used and visualized by the platform in the same manner as relationships created by topology collection, Dynamic Component Mapping, and manually in the user interface. The settings for configuring the creation of relationships in a configuration SNMP Dynamic Application are the same as the relationship settings for other Dynamic Application protocols.

What is Concurrent SNMP Collection?

To increase the scale for SNMP collection, you can enable ***Concurrent SNMP Collection***.
Concurrent SNMP Collection uses the standalone container called the SL1 SNMP Collector.

The SNMP Collector is an independent service that runs as a container on a Data Collector. When you enable Concurrent SNMP Collection, each Data Collector will contain four (4) SNMP Collector containers.

NOTE: On each Data Collector, SL1 will restart each of the SNMP Collector containers periodically to ensure that each container remains healthy. When one SNMP Collector container is restarted, the other three SNMP Collector containers continue to handle the workload.

With Concurrent SNMP Collection, SNMP collection tasks can run in parallel. A single failed task will not prevent other tasks from completing.

Concurrent SNMP Collection provides:

- Improved throughput for SNMP Dynamic Applications
- Reduced use of resources on each Data Collector
- More dependable collection from high-latency Devices

For more information, see ***Concurrent SNMP Collection***.

Chapter 2

Concurrent SNMP Collection

Overview

This chapter describes how to configure and use Concurrent SNMP Collection to run SNMP collection jobs.

Use the following menu options to navigate the SL1 user interface:

- To view a pop-out list of menu options, click the menu icon (☰).
- To view a page containing all of the menu options, click the Advanced menu icon (•••).

This chapter covers the following topics:

Using Concurrent SNMP Collection

To increase the scale for SNMP collection, you can enable ***Concurrent SNMP Collection***. Concurrent SNMP Collection uses the standalone container called the SL1 SNMP Collector.

The SNMP Collector is an independent service that runs as a container on a Data Collector. When you enable Concurrent SNMP Collection, each Data Collector will contain four (4) SNMP Collector containers.

NOTE: On each Data Collector, SL1 will restart each of the SNMP Collector containers periodically to ensure that each container remains healthy. When one SNMP Collector container is restarted, the other three SNMP Collector containers continue to handle the workload.

With Concurrent SNMP Collection, SNMP collection tasks can run in parallel. A single failed task will not prevent other tasks from completing.

Concurrent SNMP Collection provides:

- Improved throughput for SNMP Dynamic Applications
- Reduced use of resources on each Data Collector
- More dependable collection from high-latency Devices

You can enable all, one, or multiple Collector Groups to use concurrent SNMP collection.

Enabling Concurrent SNMP Collection

NOTE: This feature is disabled by default.

To enable Concurrent SNMP Collection in SL1:

1. Go to the **Behavior Settings** page (System > Settings > Behavior).
2. Check the ***Enable Concurrent SNMP Collection*** field.
3. Click **[Save]**.

TIP: If you do not want all of your SL1 Collectors to use Concurrent SNMP Collection, you can specify which Collector Units should use it in *Enabling a Collector Group to Use Concurrent SNMP Collection.*

Enabling a Collector Group to Use Concurrent SNMP Collection

Depending on the needs of your SL1 environment, you can enable or prevent a Collector Group from using concurrent SNMP collection.

To enable Concurrent SNMP Collection with a SL1 Collector Group:

1. Go to the **Collector Group Management** Page (System > Settings > Collector Groups).
2. Click the wrench icon () for the Collector Group you want to edit. The fields at the top of the page are updated with the data for that Collector Group.
3. Select an option in the ***Enable Concurrent SNMP Collection*** drop-down field:
   - ***Use system-wide default***. Select this option if you want this Collector Group to use or not use Concurrent SNMP Collection based on the ***Enable Concurrent SNMP Collection*** field on the **Behavior Settings** page. This is the default.
   - ***Yes***. Select this option to enable Concurrent SNMP Collection for this Collector Group, even if you did not enable it on the **Behavior Settings** page.
   - ***No***. Select this option to prevent this Collector Group from using Concurrent SNMP Collection, even if you did enable it on the **Behavior Settings** page.
4. Update the remaining fields as needed, and then click **[Save]**.

Chapter 3

SNMP Collection Objects

Overview

This chapter describes how to define collection objects for SNMP Dynamic Applications.

All other elements of SNMP Dynamic Applications, such as presentation objects and alerts, behave in the same manner as other Dynamic Application types. For details on other parts of SNMP Dynamic Applications, see the manual ***Dynamic Application Development***.

Use the following menu options to navigate the SL1 user interface:

- To view a pop-out list of menu options, click the menu icon (☰).
- To view a page containing all of the menu options, click the Advanced menu icon (⋯).

This chapter covers the following topics:

Collection Objects

Like all Dynamic Application types, SNMP Dynamic Applications contain collection objects. Collection objects for SNMP Dynamic Applications share common characteristics with collection objects for other Dynamic Application types, such as naming, data typing, and grouping. For details on these fields, see the manual ***Dynamic Application Development***.

Unlike collection objects for other Dynamic Application types, collection objects for SNMP Dynamic Applications are based on SNMP OIDs. Because of this, Collection Objects for SNMP Dynamic Applications have the following unique field:

- ***SNMP OID***. The Object Identifier associated with the object. SL1 will use the OID to perform an SNMP walk on the device aligned with the Dynamic Application.

You can find this field and other information about Collection Objects for an SNMP Dynamic Application by clicking the wrench icon (🔧) for a Dynamic Application, and then selecting the **[Collections]** tab. A list of Collection Objects aligned with that Dynamic Application appears at the bottom of the window.

NOTE: You can also add collection objects to an SNMP Dynamic Application through the **OID Browser** page (System > Tools > OID Browser) or through the **SNMP Walker** tool (Device Administration panel > Toolbox > SNMP Walker). For details, see the section on ***Viewing MIBs and Using the SNMP Walker Tool.***

TIP: ***Scalar OIDs*** represent a single object; ***tabular OIDs*** represent groups of objects in a MIB table. For improved efficiency, you can add a ".0" to the end of all scalar OIDs in an SNMP Dynamic Application. For example, for the singular object "sysDescr", you could enter ".1.3.6.1.2.1.1.1.0" in the ***SNMP OID*** field.

Extending the SNMP OID Field

SL1 allows you to use the ***SNMP OID*** field to perform more sophisticated OID evaluations and operations when defining the value of a collection object.

TIP: You can find the ***SNMP OID*** field by clicking the wrench icon (🔧) for a Dynamic Application, and then selecting the **[Collections]** tab.

Using Multiple OIDs and/or Existing Collection Objects to Define a Single Collection Object

In the ***SNMP OID*** field, you can concatenate the values from one or more OIDs or one or more collections objects.

To concatenate the values from one or more OIDs in a single SNMP collection object, use the following syntax in the ***SNMP OID*** field:

OID number or partial OID number.collection object.integer

where:

- *OID number or partial OID number*. Enter the OID number or partial OID number. SL1 will retrieve the value of this OID.
- *collection_object*. You can specify one or more collection object names (in SL1, these name usually begin with "o_"). SL1 will retrieve the value of the collection object and include it in the collection object.
- *integer*. You can specify an integer to append to include in the collection object.

For example, you could enter the following in the ***SNMP OID*** field:

- *<partial oid>.<collection object>*

  ```
  .1.3.6.1.2.1.1.9.1.o_1
  ```

- *<partial oid>.<collection object>.<integer>*

  ```
  .1.3.6.1.2.1.1.9.1.o_1.0
  ```

- *<partial oid>.<collection object 1>.<collection object 2>.<integer>*

  ```
  .1.3.6.1.2.1.1.9.o_1.o_2.0
  ```

NOTE: If a collection object is dependent upon the value of another collection object, during polling SL1 polls the values in order of dependency. For example, if the definition of o_123 includes the value of o_456, SL1 polls o_456 before polling o_123.

Using Python Operators to Define a Collection Object

The ***SNMP OID*** field can include:

Arithmetic operators (+, -, *, /), comparison operators (>, <, !=, ==), and/or boolean operators ("and", "or", "not").

NOTE: The "==" operator is used to test equality. The single "=" assignment operator is not supported, except when used inside some ScienceLogic functions.

- For example, you could enter the following in the ***SNMP OID*** field:

  ```
  o_16752 + o_16753
  ```

  and the new collection object will be populated with the sum of the two collection objects.

- If you want to perform a calculation with **multiple OIDs** (*not*multiple collection objects) in the ***SNMP OID*** field, use the **em7snmp.collect** function. For example, you could enter the following in the ***SNMP OID*** field:

  ```
  em7snmp.collect (".1.3.6.1.2.1.1.1") + em7snmp.collect
  (".1.3.6.1.2.1.1.2")
  ```

  and the new collection object will be populated with the sum of the values from the two OIDs.

Using Variables in the SNMP Field to Specify Component Devices

For details on configuring Dynamic Applications to discover component devices, see the manual ***Dynamic Application Development***.

In the Dynamic Application that discovers component devices, SL1 will store the value of the collection object designated as the Component Identifier into variables as follows:

Component Identifier	Variable Name
Device Name	comp_name
Distinguished Name	comp_dn
Unique Identifier	comp_unique_id
GUID	comp_guid

Suppose you have created a discovery Dynamic Application and discovered three component devices. Suppose you want to create a Dynamic Application that will collect data about these new component devices. In the ***SNMP OID*** field for each collection object, you can append ***comp_name***, ***.comp_dn*** , ***comp_unique_id***, or ***.comp_guid*** to the OID. This ensures that the collected data for each component device is easy to identify in SL1 and can be aligned with the appropriate component device.

To specify that a value varies for each component device, use the following syntax:

OID number or partial OID number.component_identifier_variable

where:

- *OID number or partial OID number*. Enter the OID number or partial OID number. SL1 will retrieve the value of this OID.

- *component_identifier_variable*. You can specify a variable for a component identifier. SL1 will substitute the value for each component device into the variable.

For example:

Suppose we have will use a single SNMP Dynamic Application to collect data about two component devices:

Component Device Name	Component DN Value
Device 1	1
Device 2	2

Suppose our SNMP data looks like this:

OID	OID Value
.1.2.3.1	number1_3
.1.2.3.2	number2_3
.1.2.4.1	number1_4
.1.2.4.2	number2_4

Suppose we entered following in the ***SNMP OID*** field:

```
.1.2.3.comp_dn
```

- For device 1, SL1 will collect the value of "1.2.3.1". That value is "number1_3".
- For device 2, SL1 will collect the value of "1.2.3.2". That value is "number2_3".

Suppose we enter following in the ***SNMP OID*** field:

```
.1.2.4.comp_dn
```

- For device 1, SL1 will collect the value of "1.2.4.1". That value is "number1_4".
- For device 2, SL1 will collect the value of "1.2.4.2". That value is "number2_4".

Using Caching to Define a Collection Object

You can use SNMP objects to define a cache for use by one or more Dynamic Applications. For SNMP Dynamic Applications, caching is defined in the ***SNMP OID*** field in the **Collection Objects** page instead of for the entire Dynamic Application.

NOTE: The ***Caching*** field in the **Dynamic Applications Properties Editor** page does not affect caching for SNMP Dynamic Applications.

To define a cache for an SNMP collection object, you ***define the cache in the SNMP OID field in all Dynamic Applications that will use the cache***, both those that will create the cache and those that will consume the cache. If the cache has timed out, the next request will re-populate the cache, making it available to all other Dynamic Applications. All Dynamic Applications can consume the cached value and refresh the cache.

To define a cache for an SNMP collection object, use the following syntax in the ***SNMP OID*** field:

em7snmp.collect ('*collection_object_definition*', enable_cache=*true_or_false*, time_to_expire=*time_in_minutes*, time_to_recollect=*time_in_minutes*)

where:

- *collection_object_definition*. The OID or extended OID for which you want SL1 to cache a value or retrieve a cached value.
- enable_cache=*true_or_false*. To either create a cache or use a cached value, specify *true*.
- time_to_expire=*time_in_minutes*. Number of minutes that the collected value will remain in the cache. This value cannot be less than the value in the time_to_recollect argument..
- time_to_recollect=*time_in_minutes*. Frequency, in minutes, at which SL1 will try to collect a value for the collection object. If collection succeeds, SL1 will populate the collection object with the new value and store the new value in the cache. If collection fails, SL1 will populate the collection object with the previously cached value.

For example:

```
em7snmp.collect('.1.3.6.1.2.1.25.2.3.1.3', enable_cache=True,time_to_
expire=15, time_to_recollect=15)
```

- In this example, the collection object will be populated with the value of the OID '.1.3.6.1.2.1.25.2.3.1.3'.
- The collected value will be stored in the cache.
- The value will be re-collected every 15 minutes.
- If we use this collection object in another Dynamic Application, that collection object can both populate the cache and retrieve values from the cache.

Filtering Results to Define a Collection Object

In the ***SNMP OID*** field , you can use the function **em7utils.filter** to filter the results.

The **em7utils.filter** function also supports the following operators:

- eq (equals)
- gt (greater than)
- ge (greater than or equal to)
- lt (less than)
- le (less than or equal to)

Suppose the OID .1.3.6.1.2.1.1.9.1.4 is an SNMP table, and for device 1, the data looks like this:

OID	OID Value
.1.3.6.1.2.1.1.9.1.4.1	1
.1.3.6.1.2.1.1.9.1.4.2	3
.1.3.6.1.2.1.1.9.1.4.3	9
.1.3.6.1.2.1.1.9.1.4.4	2
.1.3.6.1.2.1.1.9.1.4.5	4

OID	OID Value
.1.3.6.1.2.1.1.9.1.4.6	8
.1.3.6.1.2.1.1.9.1.4.7	6
.1.3.6.1.2.1.1.9.1.4.8	10
.1.3.6.1.2.1.1.9.1.4.9	100
.1.3.6.1.2.1.1.9.1.4.10	1000

We could enter the following in the ***SNMP OID*** field :

```
em7utils.filter(em7snmp.collect(".1.3.6.1.2.1.1.9.1.4"), gt (10))
```

For device 1, the collection object would contain the array:

.9, 100

.10, 1000

For another example, suppose you want to collect information about interfaces, but you don't want any information about loopback interfaces. Loopback interfaces use the interface type "24". You could enter the following in the ***SNMP OID*** field:

```
em7utils.filter(em7snmp.collect(".1.3.6.1.2.1.2.2.1.3", lambda x:x!="24"))
```

The OID .1.3.6.1.2.1.2.2.1.3 maps to the interfacetype object in the interface table in the IF-MIB. The "lambda x " syntax creates a sub-function called "x". The logic then says "collect values where 'x' is not equal to '24'". Those values will be stored in the collection object. SL1 retrieves all the interface types that do not have a value of "24".

For details on the lambda syntax, consult the Python documentation https://docs.python.org/2/tutorial/controlflow.html?highlight=lambda#lambda-expressions

Defining "Fallback" Values for a Collection Object

In the ***SNMP OID*** field, you can use syntax that tells SL1:

- If the first OID is available, use the first OID
- If the first OID is not available, use the second OID
- If the second OID is not available, use the third OID
- and so on

To include multiple OIDs in the ***SNMP OID*** field, use this syntax:

```
em7snmp.collect (".1.3.6.1.2.1.1.1") or em7snmp.collect
(".1.3.6.1.2.1.1.2") or em7snmp.collect (".1.3.6.1.2.1.1.3")
```

To include multiple collection objects in the ***SNMP OID*** field:

```
o_1234 or o_1234 or o_1236
```

To include a mixture of OIDs and collection objects in the ***SNMP OID*** field:

```
o_1234 or em7snmp.collect ('.1.3.6.1.2.1.1.2')
```

Using Index Values and Parsed Index Values to Define a Collection Object

If you are creating a collection object of type "Index", in the ***SMNP OID*** field, you can collect an entire index or parse an index.

- To collect an entire index, use the syntax:

  em7snmp.full_index (*object_name*)

  For example:

  ```
  em7snmp.full_index (o_1234)
  ```

  This example returns the full index for the collection object o_1234 and stores the index in a new collection object.

- To parse a multi-part index, you can enter the following in the ***SNMP OID*** field:

  ```
  em7snmp.parse_index (collection_object,[index_format], index_element_
  number)
  ```

  where:

  - *collection_object* is the collection object for which you want to retrieve an index
  - *index_format* describes the data type and order of the elements in the index. The index_format is surrounded by square brackets and each element is separated by a comma. The basic index types used in SNMP are described in detail in RFC 2578 Section 7.7, (https://tools.ietf.org/html/rfc2578#section-7). You can include the following elements:

    INDEX_TYPE_INTEGER

    INDEX_TYPE_IP

    INDEX_TYPE_STRING

    INDEX_TYPE_IMPLIEDSTRING

    INDEX_TYPE_OID

    INDEX_TYPE_IMPLIEDOID

- *index_element_number* is the position of the index_element that you want to collect. The first element in a multi-part index has the index_element_number of "0" (zero).

For example, suppose we have a collection object "o_224". Suppose the index for the OID in o_224 includes the following elements:

ipCidrRouteDest, ipCidrRouteMask, ipCidrRouteTos, ipCidrRouteNextHop

Here is the information about the example index:

OID Name	Data Type	Element Position in the Index
ipCidrRouteDest	INDEX_TYPE_IP	0
ipCidrRouteMask	INDEX_TYPE_IP	1
ipCidrRouteTos	INDEX_TYPE_INTEGER	2
ipCidrRouteNextHop	INDEX_TYPE_IP	3

To retrieve this index and assign only the value of ***ipCidrRouteNextHop*** to the new index collection object , we would enter the following in the ***SNMP OID*** field :

```
em7snmp.parse_index (o_224, [INDEX_TYPE_IP, INDEX_TYPE_IP, INDEX_TYPE_
INTEGER, INDEX_TYPE_IP], 3)
```

Chapter 4

Viewing MIBs and Using the SNMP Walker Tool

Overview

This chapter describes a management information base (MIB) and how to view, import, and compile MIBs. This chapter also describes the OID Browser and the SNMP Walker Tool.

Use the following menu options to navigate the SL1 user interface:

- To view a pop-out list of menu options, click the menu icon (☰).
- To view a page containing all of the menu options, click the Advanced menu icon (···).

This chapter covers the following topics:

The Management Information Base

A management information base (MIB) is a collection of definitions associated with a device or application. A MIB is stored in a file and uses a standardized format defined by the SNMP protocol.

A MIB file is usually associated with a hardware manufacturer or software manufacturer. Some manufacturers use a single MIB that contains information about all their products. Other manufacturers create a separate MIB for each product. A MIB file is generic—it is not associated with a specific installed instance of a device, but rather applies to all devices or applications of that make and model.

Each aspect of a device or application that can be managed or monitored is represented within the MIB as an object ID (OID). In SL1, we use these OIDs to define Collection Objects. For example, suppose we want to monitor the temperatures of a Force 10 switch/router. The MIB for the Force 10 switch/router includes an OID that contains the current temperature of the entire unit. To monitor the temperature of a Force 10 switch/router, we could use this OID in a Dynamic Application.

Before creating a Dynamic Application, you must ensure that SL1 contains the appropriate MIB file. This section will describe how to view and examine MIBs and how to import a MIB.

Viewing MIBs

By default, SL1 includes an extensive set of MIBs. Before creating a new Dynamic Application, you must make sure that SL1 already includes the MIB file and that the MIB file includes the data points that you want to monitor with SL1. If SL1 does not include the MIB file, you must ***import it***.

To view the list of MIBs included with SL1:

1. Go to the **MIB Compiler** page (System > Tools > MIB Compiler).
2. For each MIB file, the **MIB Compiler** page displays the following:

- ***MIB Name***. Name of the MIB file.
- ***Vendor***. For MIBs that use SMIv2, displays the vendor associated with the MIB.
- ***MIB Revision Date***. Date the original author last revised and released the MIB.
- ***Language***. Either SMIv1 or SMIv2.
- ***Compiled***. Specifies whether or not the MIB has been compiled in SL1.
- ***Type Defs***. Number of data types within the MIB.
- ***Nodes***. Number of objects that can be monitored with this MIB
- ***Traps***. Number of traps that can be monitored with this MIB.
- ***Logs***. Number of error messages in the log file after compiling the MIB.
- ***Last Edit***. Date and time the MIB was created or last edited.
- ***User Edit***. User who imported or last edited the MIB.

TIP: To sort the list of MIB files, click on a column heading. The list will be sorted by the column value, in ascending order. To sort by descending order, click the column heading again. The ***Last Edit*** column sorts by descending order on the first click; to sort by ascending order, click the column heading again.

Viewing the Contents of a MIB file

From the **MIB Compiler** page, you can view the contents of a MIB file. To do this:

1. Go to the **MIB Compiler** page (System > Tools > MIB Compiler).
2. In the **MIB Compiler** page, find the MIB you want to view. Select its information icon (ℹ).
3. The **MIB Viewer** modal page appears and displays the contents of the MIB file. You can view but cannot edit the MIB file from this modal page.

Importing a MIB

By default, SL1 includes a large number of commonly-used MIB files. However, you can import additional MIB files into SL1. If you need to monitor a device or application for which SL1 does not include a MIB file, you can import and compile the appropriate MIB file. You might also want to update an existing MIB file with a later version.

To import a MIB file into SL1:

1. Go to the **MIB Compiler** page (System > Tools > MIB Compiler).
2. In the **MIB Compiler** page, select the **[Import]** button.
3. In the **MIB Import** modal page, navigate to the location of the MIB file on your local computer. Select the **[Import]** button.
4. The new MIB file appears in the list of MIB files in the **MIB Compiler** page.
5. You must compile the MIB file before SL1 can use it. To compile a MIB, select its lightning bolt icon (⚡).

Compiling a MIB

Before SL1 can use a MIB file to monitor data points on a device or an application, you must ensure that the MIB file has been compiled. When you compile the MIB file, you are telling SL1 to read and parse the object definitions in the MIB file. SL1 can then use that data when monitoring devices or applications.

To compile a MIB:

1. Go to System > Tools > MIB Compiler.
2. In the **MIB Compiler** page, find the MIB you want to compile. Select its lightning bolt icon (⚡).
3. To determine if a MIB compiled successfully, you can view the ***Compile*** column for that MIB.
4. If a MIB does not compile successfully, you can view information about the compile operation. To do this, find the MIB file in the list of MIBs and select its page icon (📄).

5. If a MIB compiles successfully, but shows a value other than "0" (zero) in the ***Messages*** column, you can view information about the compile operation. To do this, find the MIB file in the list of MIBs and select its page icon (📄). In some cases, a MIB might compile successfully but might also include incorrectly defined data-points (called "objects" or "OIDs"). These data-points might not be included in the compiled MIB.

IMPORTANT: If a MIB does not compile, please view the compile-log for the MIB. MIBs that use incorrect format or syntax may fail to compile. Contact the MIB's vendor for corrections or updates. ScienceLogic is not responsible for other vendor's MIB files.

Viewing Error Logs for a MIB

After compiling a MIB file, you can view the error log for the MIB to check on the status of the compile operation. To do this:

1. Go to System > Tools > MIB Compiler.
2. In the **MIB Compiler** page, find the MIB you want to view information about. Select its page icon (📄).
3. The **MIB Compiler Log Messages** modal page displays error messages generated when a MIB is compiled and the effect of that error.
4. The format of each message is:

   MIB file name:line number:error message

   where:

   - ***MIB file name***. Is the name of the MIB file.
   - ***line number***. Is the line on which the error occurred.
   - ***error message***. Is a description of the error.

5. Some common error messages are:

   - ***access 'write-only' is no longer allowed in SMIv2***. SMIv2 changed the MAX-ACCESS write-only to read-write.
   - ***ACCESS is SMIv1 style, use MAX-ACCESS in SMIv2 MIBs instead***. A MIB that is considered SMIv2 compliant has the label MAX-ACCESS instead of ACCESS. Either remove the SMIv2 specification from the import or rename all of the ACCESS identifiers to MAX-ACCESS.
   - ***default value does not match underlying enumeration type***. The default value for an enumeration does not match the values declared in the enumeration. The values must match.
   - ***description missing in object definition***. All objects in an SMIv2 MIB have descriptions associated with them. This can be a blank string ("") or have text describing the object.
   - ***identifier 'identifier ' cannot be imported from module 'module name'***. This means either the type definition was not found in module name. The type definition probably exists in another module.
   - ***'identifier' should start with a lower case letter***. The identifier in a MIB element should start with a lower case alpha character. A type definition starts with an upper case alpha character.

- ***index element 'index identifier' of row 'identifier' must have a size restriction***. All tables declared in a MIB have an index element. An index element should have a size restriction on the index element to make it a compliant MIB. To correct an integer index, add (mix.max) after the type definition to correct this issue.
- ***invalid status 'mandatory' in SMIv2 MIB***. A MIB that is considered SMIv2 compliant has a STATUS of current, depreciated, or obsolete. Mandatory is an older SMIv1 standard that was changed with SMIv2.
- ***last subidentifier assigned to 'identifier' may not be zero***. All SMIv2 MIBs must not have an identifier of zero in the MIB. The values have to be between 1 and 65536.
- ***length of hexadecimal string 'string' is not a multiple of 2***. Hexadecimal strings (also called octet strings) need to have 2 elements in the default value. For example, a declaration of '0f' (zero - f) is correct as it is 2 bytes. A declaration of 'f' is not correct as it is only 1 byte.
- ***lexically unexpected character, skipping to end of line***. There is an unknown character in the MIB file at the given line. Removing the character should allow the parser to continue parsing the file. Check for a back tick (`) and other strange characters and remove them from that line.
- ***macro 'macro type' has not been imported from module 'module name'***. A macro was called in the MIB however the macro type was not imported at the top of the MIB. To correct this, add the import statement at the top of the MIB.
- ***parse error, unexpected '{', expecting UPPERCASE_IDENTIFIER or LOWERCASE_IDENTIFIER or NUMBER***. The object did not parse correctly. There could be a spelling error for a MIB type or an extra comma or a single dash in the declaration. Check the MIBs syntax for any issues.
- ***parse error, unexpected COLON_COLON_EQUAL***. There is typically a parser error above this error. If you correct the parser error, the parser will find the ::= in the MIB and correctly build this identifier.
- ***parse error, unexpected DESCRIPTION***. There was a parser error above this line. Correct the parse error and this error will be corrected.
- ***parse error, unexpected MODULE_COMPLIANCE, expecting OBJECT***. The MODULE_ COMPLIANCE macro was not imported from SNMPv2-CONF. All macros need to be imported into the MIB for the parser to correctly build the MIB tree.
- ***parse error, unexpected MODULE_IDENTITY, expecting OBJECT***. The MODULE-IDENTITY macro was not imported from SNMPv2-SMI. All macros need to be imported into the MIB for the parser to correctly build the MIB tree.
- ***parse error, unexpected VARIABLES***. There was a parser error above this line. Correct the parser error and this error will be corrected.
- ***redefinition of identifier 'identifier'***. An identifier was declared again in the MIB. Check the first declaration is correct or if the re-declaration is correct. Remove the one that is not correct.
- ***row's parent node must be a table node***. A row in a table must have a table identifier to link together. These typically show up as a parse error above this error.
- ***scalar's parent node must be simple node***. A scalar must have a container (OBJECT IDENTIFIER) as a parent node. A scalar cannot be part of a table, row, or another scalar.
- ***SMIv2 base type 'type definition' must be imported from SNMPv2-SMI***. The MIB is missing an import statement for the type definition.

- ***subtyping not allowed***. A table or scalar cannot have rows or objects typed in the scalar or table definition. They should be typed as the row declaration or as the object definition.
- ***table's SEQUENCE OF type does not match row type***. The table was either declared out of order from the row declaration or the rows have a mistyped sequence.
- ***TRAP-TYPE macro is not allowed in SMIv2***. Traps are to be declared in separate SMIv1 MIBs. You cannot add traps to an SMIv2 MIB.
- ***type of 'identifier ' in sequence and object type definition do not match***. This means the sequence of an object and the object definition are not the same type definition. This is more than likely a syntax error.
- u***nknown object identifier label 'identifier'***. The object identifier was not declared in the MIB file or it was not imported into the MIB. These typically show up as a parser error because that identifier was not parsed correctly.
- ***unknown object identifier label 'label name'***. This means an OID was cast as a new identifier and the MIB compiler was unable to find that identifier definition. These are called type definitions in the SMIv2 specification.
- ***unknown type 'type definition'***. The type definition was not declared in the MIB or it was not imported from another MIB definition. To correct this, add an import statement at the top of the MIB.

Rebuilding the SNMP Tree

If you have added new MIB files to SL1, you should rebuild the SNMP tree. When you rebuild, SL1 searches for any newly added MIB files, compiles all the new MIB files, and rebuilds the SNMP tree to include the new files.

To rebuild the SNMP tree:

1. Go to System > Tools > MIB Compiler.
2. In the **MIB Compiler** page, select the **[Rebuild]** button.

Exporting a MIB

If you want to examine or edit a MIB file or if you want to install a MIB file on another computer, you can export the MIB file from SL1. To do this:

1. Go to System > Tools > MIB Compiler.
2. In the **MIB Compiler** page, find the MIB you want to export. Select its diskette icon (💾).
3. You will be prompted to save the MIB file to a location on your local computer.

Compiling Multiple MIB Files or Deleting Multiple MIB Files

The **MIB Compiler** page contains a drop-down field in the lower right called ***Select Action***. This field allows you to apply an action to multiple MIB files at once.

To apply an action to multiple MIB files:

1. In the **MIB Compiler** page, select the checkbox for each MIB file you want to apply the action to. To select all checkboxes for all MIB files, select the select the red checkbox (☑) at the top of the page.
2. In the ***Select Action*** drop-down list, select one of the following actions:

   - *Delete MIB and Objects*. MIB and OIDs are removed from the ScienceLogic MIB library. Existing Dynamic Applications that use this MIB and OIDs are unaffected. However, users cannot create new Dynamic Applications based on this MIB.
   - *Compile MIB*. Compiles the selected MIBs

3. Select the **[Go]** button to apply the selected action to the selected MIB files.

The OID Browser

The **OID Browser** page (System > Tools > OID Browser) allows you to view a list of all objects in all MIBs that have been compiled in SL1. The objects are arranged in the standard SNMP tree structure. The **OID Browser** page allows you to search a list of OIDs, "drill down" and view information under an OID, and add an OID to a Dynamic Application.

Drilling Down in OIDs

You can drill down the tree and view the information under an OID in the **OID Browser** page (System > Tools > OID Browser). To do so:

1. Go to the **OID Browser** page (System > Tools > OID Browser).
2. In the **OID Browser** page, find the OID that you want to drill down. Click on its value in the ***Object Name*** column.
3. The **OID Browser** page is refreshed and displays only the selected OID and all the OIDs in the next level of the OID.
4. You can continue clicking on values in the ***Object Name*** column to continue drilling down. When a checkbox appears for an OID, you cannot drill down any further.

NOTE: You can also find the OID that you want to drill down and select its Show OID Tree icon (). The **OID Browser** page is refreshed and displays only the selected OID and the OIDs in the next three levels of the OID's tree.

Searching the List of OIDs

You can search the list of OIDs in all MIBs that have been compiled in SL1 in the **OID Browser** page (System > Tools > OID Browser). To search the list of OIDs in SL1:

1. Go to the **OID Browser** page (System > Tools > OID Browser).
2. The search fields at the top of the **OID Browser** page allow you to search for OIDs by one of the following parameters:

- ***Search where.*** Specifies the parameter you want to search by. You can select from the following:
    - *Where OID is like*. Searches all OID definitions for those that have the same object identifier as that entered in the regular expression field.
    - *Where Symbolic is like*. Searches all OID definitions for those that have the same symbolic name as that entered in the regular expression field.
    - *Where Name is like*. Searches all OID definitions for those that have the same object name as that entered in the regular expression field.
    - ***Where MIB Name is like***. Searches all OID definitions for those that have the same MIB name as that entered in the regular expression field.
- ***regular expression***. In this field you manually enter the text to search for. You can use the following special characters in this field:
    - * Match zero or more characters preceding the asterisk. For example:

      "dell*" would match "dell", "dell2650", "dell7250" and "dell1700N".

      "*dell*" would match "mydell", "dell", "dell2650", "dell7250" and "dell1700N".
    - % Match zero or more characters preceding the asterisk. This special character behaves in the same way as the asterisk.

3. When you select the **[Search]** button, the **OID Browser** page will be refreshed and will display only the OIDs that match the search parameters.

Adding an OID to a Dynamic Application from the OID Browser

When you add an object to a Dynamic Application, SL1 collects values for the object from each device and application that is monitored with the Dynamic Application.

You can view OIDs in the **OID Browser** page and then add them to a Dynamic Application.

To do this:

1. Go to the**OID Browser** page (System > Tools > OID Browser).
2. In the **OID Browser** page, ***drill down*** to an object you want to add to a Dynamic Application.
3. Select the checkbox for the object you want to include in a Dynamic Application.
4. Repeat steps 2 and 3 for each OID you want to include in a Dynamic Application.
5. From the ***Select Action*** drop-down, choose the Dynamic Application to which you want to add the selected OIDs. You can also choose to add the selected OIDs to a new SNMP Configuration or SNMP Performance Dynamic Application.
6. Select the **[Go]** button to add the OID to the Dynamic Application.

The SNMP Walker Tool

The **SNMP Walker** page allows you to "walk" one or more SNMP OIDs on a single device, so you can see a "real life" example of the type of information stored in one or more OIDs. From the **SNMP Walker** page, SL1 will poll the device and retrieve and display a value for each selected OID. The**SNMP Walker** page allows you to examine OIDs and also to add OIDs to new or existing Dynamic Applications.

To perform an SNMP walk:

1. Go to the **Device Manager** page (Registry > Devices > Device Manager).
2. In the **Device Manager** page, select the wrench icon (🔧) for a device on which you want to perform an SNMP walk.
3. In the **Device Properties** page, select the **[Toolbox]** tab. To perform an SNMP walk, select the SNMP Walker icon in the **Device Toolbox** pane.
4. To execute an SNMP walk provide values in the following fields:

    - ***Select OID***. Users can select from the drop-down list or manually enter an OID. The SNMP values for all sub-OIDs under the selected OID will be displayed in the results pane.

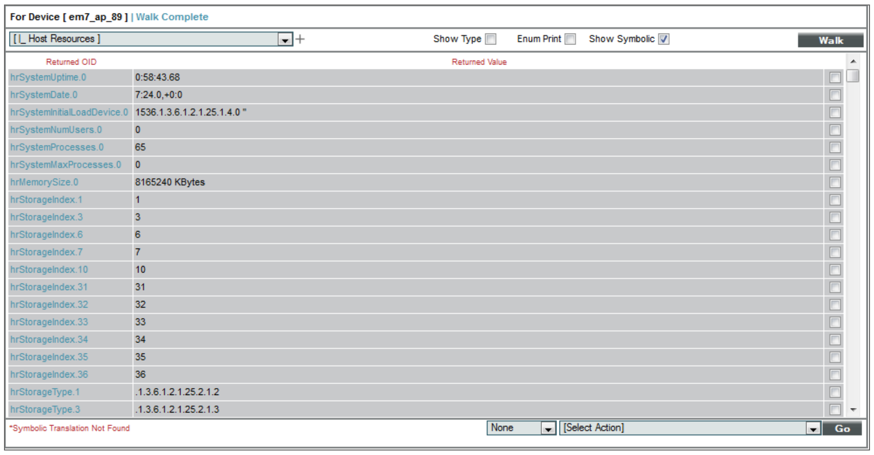
        - To enter an OID manually, select the plus sign icon. Then enter the OID in the SNMP OID field.
        - To select from the drop-down list of OIDs, highlight a MIB entry or OID entry. Choices are grouped by Common OIDs, Standard RFCs, and existing Dynamic Applications.

    - ***Show Type***. Displays the data-type (Integer, Counter, IP Address, etc.) at the start of the returned value field.
    - ***ENum Print***. Displays the list of possible values for each OID of type ENum.
    - ***Show Symbolic***. If selected, for OIDs for which SL1 contains the MIB file, displays the text-based, descriptive name, rather than the numeric ID.

5. The results pane displays the following:

    - ***Returned OID***. Either numeric ID or symbolic (text-based) ID for each retrieved OID.
    - ***Returned Value***. Value for each OID, retrieved from the device.
    - ***Checkbox***. If the MIB is already included with SL1, a checkbox appears in the rows for one or more OIDs. Selecting the checkbox allows you to apply the actions from the Select Actions field to the OID

If you walked the Host Resource MIB on a device with the ***Show Symbolic*** checkbox selected, the SNMP Walker tool might return the following information:

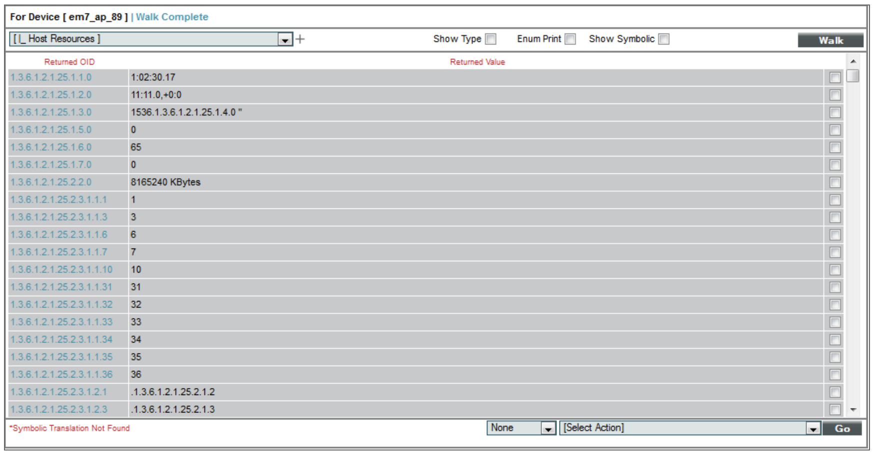

For Device [em7_ap_89] | Walk Complete

[|_ Host Resources] — Show Type ☐ Enum Print ☐ Show Symbolic ☑

Returned OID	Returned Value
hrSystemUptime.0	0:58:43.68
hrSystemDate.0	7:24.0,+0:0
hrSystemInitialLoadDevice.0	1536.1.3.6.1.2.1.25.1.4.0 "
hrSystemNumUsers.0	0
hrSystemProcesses.0	65
hrSystemMaxProcesses.0	0
hrMemorySize.0	8165240 KBytes
hrStorageIndex.1	1
hrStorageIndex.3	3
hrStorageIndex.6	6
hrStorageIndex.7	7
hrStorageIndex.10	10
hrStorageIndex.31	31
hrStorageIndex.32	32
hrStorageIndex.33	33
hrStorageIndex.34	34
hrStorageIndex.35	35
hrStorageIndex.36	36
hrStorageType.1	.1.3.6.1.2.1.25.2.1.2
hrStorageType.3	.1.3.6.1.2.1.25.2.1.3

*Symbolic Translation Not Found

If you walked the Host Resource MIB on a device with the ***Show Symbolic*** checkbox unselected, the SNMP Walker tool might return the following information:


For Device [em7_ap_89] | Walk Complete

[|_ Host Resources] — Show Type ☐ Enum Print ☐ Show Symbolic ☐

Returned OID	Returned Value
1.3.6.1.2.1.25.1.1.0	1:02:30.17
1.3.6.1.2.1.25.1.2.0	11:11.0,+0:0
1.3.6.1.2.1.25.1.3.0	1536.1.3.6.1.2.1.25.1.4.0 "
1.3.6.1.2.1.25.1.5.0	0
1.3.6.1.2.1.25.1.6.0	65
1.3.6.1.2.1.25.1.7.0	0
1.3.6.1.2.1.25.2.2.0	8165240 KBytes
1.3.6.1.2.1.25.2.3.1.1.1	1
1.3.6.1.2.1.25.2.3.1.1.3	3
1.3.6.1.2.1.25.2.3.1.1.6	6
1.3.6.1.2.1.25.2.3.1.1.7	7
1.3.6.1.2.1.25.2.3.1.1.10	10
1.3.6.1.2.1.25.2.3.1.1.31	31
1.3.6.1.2.1.25.2.3.1.1.32	32
1.3.6.1.2.1.25.2.3.1.1.33	33
1.3.6.1.2.1.25.2.3.1.1.34	34
1.3.6.1.2.1.25.2.3.1.1.35	35
1.3.6.1.2.1.25.2.3.1.1.36	36
1.3.6.1.2.1.25.2.3.1.2.1	.1.3.6.1.2.1.25.2.1.2
1.3.6.1.2.1.25.2.3.1.2.3	.1.3.6.1.2.1.25.2.1.3

*Symbolic Translation Not Found

Adding an OID to a Dynamic Application

From the **SNMP Walker** page, you can view data from an OID and determine if you want to add the OID to a Dynamic Application. To add an OID to a Dynamic Application (either new or existing):

1. Execute an SNMP Walk (described in the steps above).
2. In the **Results** pane, select the checkbox for each OID you want to add to a Dynamic Application.
3. In the ***Select Actions*** field, select the Dynamic Application to which you want to add the selected OID(s).
4. Select the **[Go]** button.
5. The **Collection Objects** page for the new or existing Dynamic Application appears, where you can use the OID to define a collection object, edit the parameters for the collection object, and save the collection object in the Dynamic Application.

You can also add an OID to a Dynamic Application from the **OID Browser** page. To learn more, see the section *Adding an OID to a Dynamic Application from the OID Browser*.

Example

1

Creating an SNMP Performance Dynamic Application

Overview

In this chapter we will create an SNMP Dynamic Application. Our Dynamic Application will:

- Collect information about file systems from the Host Resources MIB.
- Include collection objects that get size and usage values.
- Include a presentation object that displays percentage used.
- Include a collection object to of type "label", to label the graph.
- Include a discovery object.

Use the following menu options to navigate the SL1 user interface:

- To view a pop-out list of menu options, click the menu icon (☰).
- To view a page containing all of the menu options, click the Advanced menu icon (···).

This chapter covers the following topics:

Defining the Dynamic Application Properties

To create the Dynamic Application and define the general properties for this Dynamic Application, perform the following steps:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the **[Actions]** button, and then select *Create New Dynamic Application*. The **Dynamic Applications Create New Application** page appears.
3. Supply values in the following fields:

   - ***Application Name***. Enter "SNMP Performance Dynamic Application" in this field.
   - ***Application Type***. Select *SNMP Performance*.
   - ***Poll Frequency***. Select *Every 1 Minute* to see data as quickly as possible.

4. For this example, you can leave the remaining fields set to their default value. Select the **[Save]** button to save the Dynamic Application.

Adding the Discovery Object

This Dynamic Application includes one discovery object. A discovery object helps SL1 to align the Dynamic Application to devices that use the Host Resources MIB.

A discovery object is a variable that is unique to a specific hardware component or software application. Ideally, a discovery object is a variable that is always available if SNMP is running on a device, regardless of the processes running on the device or the state of the device. For example, an OID that contains a serial number of a hardware version number would make a good discovery object.

During initial discovery and nightly auto-discovery, SL1 checks each discovered device against the list of already-defined Dynamic Applications. SL1 searches each discovered device to find "discovery objects" and aligns devices with the appropriate Dynamic Application(s).

For example:

- Suppose your network includes servers from Dell.
- Suppose your SL1 system includes a Dynamic Application for Dell servers.
- Suppose the Dynamic Application for Dell servers uses the SNMP variable for chassis ID to create a discovery object. This chassis-ID variable is defined by Dell and published in their MIB files. Following SNMP standards, the variable will have a unique, numeric name, used only for Dell servers.
- Because a chassis is included with each Dell server and because the chassis is not affected by software processes or the operating system, the chassis-ID variable will always be available if SNMP is running on a Dell server.
- During initial discovery and nightly auto-discovery, if SL1 can retrieve a value for the unique chassis-ID variable, SL1 concludes that the device is a Dell server. SL1 will then assign the Dynamic Application for Dell

servers to the device. SL1 will later use the Dynamic Application for Dell servers to retrieve information from the device.

- During initial discovery and nightly auto-discovery, if SL1 cannot retrieve a value for the chassis-ID variable, SL1 concludes that the device is not a Dell server. SL1 will not assign the Dynamic Application for Dell servers to the device.

For more details on discovery, see the manual on ***Discovery and Credentials***.

To create the discovery object to the Dynamic Application:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the wrench icon (🔧) for the *SNMP Performance Dynamic Application*. The **Dynamic Applications Properties Editor** page appears.
3. Select the **[Collections]** tab. The **Dynamic Applications | Collections Objects** page appears.
4. Supply values in the following fields:
   - ***Object Name***. Enter "Discovery Object" in this field.
   - ***SNMP OID***. Enter ".1.3.6.1.2.1.25.2.3.1.3" in this field. This is the OID for Storage Description, as specified in the Host Resources MIB. We chose this OID to use for the discovery object because a device that uses the Host Resources MIB to report file system information will always include this OID
   - ***Class Type***. Select *100.Discovery*. SL1 will gather values from all objects in the Dynamic Application.
5. Select the **[Save]** button, and the new object will be added and new fields will appear. Supply values in the following fields:
   - ***Alignment Condition***. Select *Align if OID is present*. SL1 will align the Dynamic Application with a device only if the discovery object returns a value.
   - ***Validity Check***. We did not specify a value in this field.
6. Select the **[Save]** button to save the discovery object.

Adding the Collection Objects

Our example Dynamic Application contains three collection objects:

- Storage size
- Storage used
- Storage description

To create these collection objects, perform the following steps:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the wrench icon (🔧) for the *SNMP Performance Dynamic Application*. The **Dynamic Applications Properties Editor** page appears.
3. Select the **[Collections]** tab. The **Dynamic Applications | Collections Objects** page appears.
4. To create the collection object for ***storage size***, supply values in the following fields:

- ***Object Name***. Enter "Storage Size" in this field.
- ***SNMP OID***. Enter ".1.3.6.1.2.1.25.2.3.1.5". This is the OID for storage size, as specified in the Host Resources MIB.
- ***Class Type***. Storage size is a number that can go up or down between polls. Select *4 Performance Gauge* in this field.
- ***Group Number***. Select *Group 1*. For performance Dynamic Applications, SL1 uses the ***Group Number*** setting to associate performance values with the appropriate labels. For the performance graph for this example to display labels correctly, all the collection objects must be in the same group.

5. For this example, you can leave the remaining fields set to their default values.
6. Select the **[Save]** button.
7. Select the **[Reset]** button to clear the form fields.
8. To create the collection object for storage used, supply values in the following fields:
    - ***Object Name***. Enter "Storage Used" in this field.
    - ***SNMP OID***. Enter ".1.3.6.1.2.1.25.2.3.1.6". This is the OID for storage description , as specified in the Host Resources MIB.
    - ***Class Type***. Storage size is a number that can go up or down between polls. Select *4 Performance Gauge* in this field.
    - ***Group Number***. Select *Group 1*. For performance Dynamic Applications, SL1 uses the ***Group Number*** setting to associate performance values with the appropriate labels. For the performance graph for this example to display labels correctly, all the collection objects must be in the same group.
9. For this example, you can leave the remaining fields set to their default values.
10. Select the **[Save]** button.
11. Select the **[Reset]** button to clear the form fields.
12. To create the collection object for the ***storage description label***, supply values in the following fields:
    - ***Object Name***. Enter "Storage Description" in this field.
    - ***SNMP OID***. Enter ".1.3.6.1.2.1.25.2.3.1.3". This is the storage description OID specified in the Host Resources MIB.
    - ***Class Type***. Select *Label (Always Polled)* in this field. In performance Dynamic Applications, collection objects that use this class type are string values that SL1 uses to label the lines on a performance graph.
    - ***Group Number***. Select *Group 1*. For performance Dynamic Applications, SL1 uses the Group Number setting to associate performance values with the appropriate labels. For the performance graph for this example to display labels correctly, all the collection objects must be in the same group.
13. For this example, you can leave the remaining fields set to their default values.
14. Select the **[Save]** button.

Creating the Presentation Object

When you create a collection object in a Dynamic Application of type Performance, SL1 automatically creates a presentation object that corresponds to that collection object. In this example, we will remove these presentation objects and create a new presentation object that displays file system usage in percent.

To create the presentation object:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the wrench icon (🔧) for the *SNMP Performance Dynamic Application*. The **Dynamic Applications Properties Editor** page appears.
3. Select the **[Presentations]** tab. The **Dynamic Applications Presentation Objects** page appears.
4. In the **Dynamic Applications Presentation Objects** page, the *Storage Size* and *Storage Used* collection objects have been created by default. Select each presentation object's bomb icon (💣) to delete them.
5. Select the **[Reset]** button. To create a new presentation object that displays storage used, in percent, enter values in the following fields:

    - **Report Name**. Enter "Percentage Used" in this field.
    - **Active State**. Select *Enabled*. SL1 will generate a report of the presentation object.
    - **Data Unit**. Enter "Percent" into this field.
    - **Abbreviation/Suffix**. Enter "%" into this field.
    - **Show as Percent**. Select *Yes*. The graph will display percent values.
    - **Formula Editor**. We want our graph to display Percentage Used, so we will use the following formula:

        *Storage Used/Storage Size * 100*

        We can enter this formula manually, or by selecting the Object IDs in the ***Formula Editor*** and selecting the **[Add]** button. Using the Object IDs provided by SL1, enter the following into the ***Formula Editor***:

        *(o_9223)/(o_9222)*100*

    NOTE: The object IDs in your Dynamic Application will be unique to your system. In the provided formula, you must replace "o_5511" and "o_5513" with the object IDs for the ***Storage Used*** and ***Storage Size*** collection objects in your system.

6. In this example, you can leave the remaining fields at their default value.
7. Select the **[Save As]** button to save the presentation object.

Manually Aligning the Dynamic Application to a Device

In this example we will align the Dynamic Application to a device that supports the Host Resource MIB. By manually aligning the Dynamic Application to a device, we can immediately view the Percentage Used data in the presentation object we defined.

To manually align the Dynamic Application to a device:

1. Go to the **Device Manager** page (Registry > Devices > Device Manager).
2. In the **Device Manager** page, find the device you want to align the Dynamic Application to. In this example, we are aligning the Dynamic Application to a server. Select the device's wrench icon ().
3. The **Device Properties** page appears. Select the **[Collections]** tab.
4. In the **Dynamic Application Collections** page, select the **[Action]** button and select *Add Dynamic Application*.
5. The **Dynamic Application Alignment** page appears. Select *SNMP Performance Dynamic Application* to align this Dynamic Application with the server in the left pane. In the right pane, select *Default SNMP Credential*.
6. Select the **[Save]** button to add the Dynamic Application to the device.

Viewing the Report

After the Dynamic Application has collected data from the aligned server, you can view the performance report for that device. To view the performance report for a device with the *SNMP Performance Dynamic Application* aligned to it:

1. Go to the **Device Manager** page (Registry > Devices > Device Manager).
2. In the **Device Manager** page, find the device you want to align the Dynamic Application to. In this example, we are aligning the Dynamic Application to a server. Select the device's wrench icon ().
3. The **Device Properties** page appears. Select the **[Collections]** tab.
4. From the **Dynamic Application Collections** page, select the **[Reset]** button to update the page with the latest information.
5. Locate the *SNMP Performance Dynamic Application*. If the graph icon () is colored, the performance report is available. Select the graph icon for the ***Percentage Used*** presentation object.

Or:

1. Go to the **Device Manager** page (Registry > Devices > Device Manager).
2. In the **Device Manager** page, find the device you aligned the *SNMP Performance Dynamic Application* to. Select the device's graph icon ().
3. The **Device Summary** page appears. Select the **[Performance]** tab.

4. In the left NavBar, select *SNMP Performance Dynamic Application*, then select ***Percentage Used***.
5. The SNMP Performance Dynamic Application | Percentage Used report is displayed.
    - The report returns collection objects as a percentage value. You can mouseover different data point on the report, and the report will display the percentage values for the given time
    - The percentage is displayed to the left of the report. The values for each label object are displayed in the graph key at the bottom of the report.
6. To learn more above device performance reports, see the manual ***Device Management***.