



XML, SOAP, and XSLT Dynamic Application Development

ScienceLogic Version 12.2.0

Table of Contents

Introduction to XML, SOAP, and XSLT Dynamic Application Development	4
XML, SOAP, and XSLT Protocols	5
What is XML?	5
Elements of XML, SOAP, and XSLT Dynamic Applications	5
SOAP and XSLT Requests	7
Viewing the Requests in a Dynamic Application	9
Creating a Request	9
Defining XSLT Request Code	10
The XSLT Request Code Field	10
The Cached XSLT Request Field	12
Defining XSLT Parser Code	12
Substitution Characters	14
Collection Object Substitution Characters	14
Substitution Characters from Credentials	14
Substitution Characters from Component Devices	14
Editing a Request	15
Deleting a Request	15
Collection Objects	16
Protocol-Specific Fields for Collection Objects	17
XML Dynamic Applications	17
SOAP Dynamic Applications	17
XSLT Dynamic Applications	17
Specifying XML Tags and SOAP Tags	18
Parsing an XML Element	18
Specifying XSLT Tags	19
Session ID Objects	20
Creating an XML Dynamic Application	22
Creating the Dynamic Application	23
Defining XML Data-Points to Monitor	23
Defining the Collection Objects	26
Creating the Presentation Object	27

Testing the Dynamic Application	27
Creating a Credential	28
Manually Aligning the Dynamic Application to the Test Device	28
Viewing the Reports	29
Dynamic Component Mapping & Caching with XSLT	30
Design	32
Creating the "Example Dynamic Component Mapping General" Dynamic Application	33
Defining the Dynamic Application Properties	33
Adding the XSLT Request	34
Adding the Discovery Object	36
Creating the "Example Dynamic Component Mapping Discovery" Dynamic Application	37
Defining the Dynamic Application Properties	37
Adding the XSLT Requests	37
Adding the Collection Objects	41
Creating a Device Class for the Component Devices	44
Creating the "Example Component Performance" Dynamic Application	45
Defining the Dynamic Application Properties	45
Adding the XSLT Request	45
Adding the Collection Objects	47
Editing the Presentation Objects	49
Automatically Aligning the Dynamic Application to Component Devices	49
Using the Dynamic Applications	50
Configuring a Test Device	50
Editing the Device Class for the Test Device	50
Creating a Credential	51
Discovering the Test Device	51
Verifying the Dynamic Application Alignments	52
Viewing the Device Components Registry	52
Viewing the Component Device Map	52
Viewing the Performance Graphs	52
Expanding this Example	53

Chapter

1

Introduction to XML, SOAP, and XSLT Dynamic Application Development



Overview

This manual describes how to create Dynamic Applications that collect data by parsing collection objects from XML documents.

This manual does not cover elements of Dynamic Application development that are common to all Dynamic Application types; you should be familiar with the common elements and concepts of Dynamic Applications. For details on the common elements of Dynamic Applications, see the manual *Dynamic Application Development*. XML, SOAP, and XSLT Dynamic Applications use XML markup language.

You should be familiar with the XML markup language before developing XML, SOAP, or XSLT Dynamic Applications. The requests in XSLT Dynamic Applications are written using XSL transformations. You must be familiar with the structure, syntax, and elements of XSL transformations before developing XSLT Dynamic Applications.

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:

XML, SOAP, and XSLT Protocols	5
What is XML?	5
Elements of XML, SOAP, and XSLT Dynamic Applications	5

XML, SOAP, and XSLT Protocols

This manual describes how to create Dynamic Applications that collect data by parsing collection objects from XML documents. There are three Dynamic Application protocols that parse collection objects from XML documents:

- **XML.** For this Dynamic Application protocol, SL1 performs an HTTP GET request for the XML document specified in the associated credential. The collection objects in an XML Dynamic Application specify how SL1 should parse values from the XML document.
- **SOAP.** For this Dynamic Application protocol, SL1 performs one or more HTTP POST requests on the URL specified in the associated credential. The collection objects in a SOAP Dynamic Application specify which response returns values for the collection object and how SL1 should parse values from the returned XML document. The requests in a SOAP Dynamic Application are performed in a specified order. You can substitute a collected value from a request in the POST content of a future request. The collection object that is substituted must collect a single value, not a list of values.
- **XSLT.** For this Dynamic Application protocol, SL1 performs one or more HTTP POST or GET requests on the URL specified in the associated credential. The collection objects in an XSLT Dynamic Application specify which response returns values for the collection object and how SL1 should parse values from the returned XML document. The requests in a XSLT Dynamic Application are performed in a specified order. If the credential specifies a POST method, the POST content is generated by performing an XSL transformation on an XML document that contains all the values collected using the preceding requests. Therefore, unlike a SOAP Dynamic Application, you can use the values collected for a collection object that returns a list of values to constrict a request. All requests must specify an XSL transformation that is applied to the response returned for that request. The result of the XSL transformation that is performed on the response must be in a specific format from which SL1 parses the collection objects for that request.

What is XML?

XML stands for "eXtensive Markup Language". XML was designed to transport and store data, with focus on what data is. By itself, XML does not actually do anything. XML was created to structure, store, and transport information. XML stores data in plain-text format. This allows XML data to be exchanged between disparate hardware and software.

XML data is frequently sent using HTTP requests. Users can send SL1 data by writing client code that makes HTTP calls and sends the XML data.

When XML data is received, the receiving computer must include programs to receive, parse, and respond to XML requests. Most browsers include built-in parsers for XML.

To learn more about XML, see <http://www.w3schools.com/xml/default.asp>

Elements of XML, SOAP, and XSLT Dynamic Applications

XML, SOAP, and XSLT 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. XML, SOAP, and XSLT Dynamic Applications can be either the Performance or Configuration archetypes.
- **Properties.** Allows for version control, release notes, collection, and retention settings.
- **Collection Objects.** Define the individual data-points that will be retrieved by the Dynamic Application. These data points are called collection objects. Defines what type of data is being collected (gauge, counter, etc) and how it is grouped. XML, SOAP, and XSLT Dynamic Applications have collection object settings that are unique to the respective protocol. These settings are described in the [Collection Objects](#) section.
- **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. The threshold value can be edited for each device without affecting the behavior of the Dynamic Application for other devices.
- **Alerts.** Evaluate collected data. 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. XML, SOAP, and XSLT Dynamic Applications use SOAP/XML credentials. When an XML, SOAP, or XSLT Dynamic Application is aligned with a device, you must align a SOAP/XML credential to that Dynamic Application for collection to occur. The value specified in the **Method** field for the aligned SOAP/XML credential must be:
 - *GET* for XML Dynamic Applications.
 - *POST* for SOAP Dynamic Applications.
 - *GET* or *POST* for XSLT Dynamic Applications.
- **Caching.** When SL1 requests information from a device during Dynamic Application collection, SL1 can optionally **cache** the response from the device. If a response is cached, other Dynamic Applications that use the same protocol can use the cached response to retrieve collection object values. Caching responses reduces the number of requests performed by SL1 and speeds up collection.
- **Dynamic Component Mapping.** Dynamic Component Mapping allows SL1 to use Dynamic Application data that has been collected from a single management system, such as a VMware ESX server, to create multiple device records for the entities managed by that single management system.
- **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 XML, SOAP, and XSLT Dynamic Application are the same as the relationship settings for other Dynamic Application protocols.

Chapter

2

SOAP and XSLT Requests

Overview

SOAP and XSLT Dynamic Applications must include one or more **requests** that define how SL1 should request data from a device. Each request specifies an operation that will gather a response from the device.



Each collection object in a SOAP or XSLT Dynamic Application is associated with a request. The collection object specifies how to parse a data point from the response. A single request can be used to populate multiple collection objects.

For SOAP Dynamic Applications, a request specifies the POST content that SL1 uses to perform the request. The associated collection objects are parsed from the response.

For XSLT Dynamic Applications, a request specifies:

- An XSL transformation that SL1 applies to an XML document that contains the collected values from the previous requests. The result of this transformation is used as the POST content that SL1 uses to perform the request.
- A second XSL transformation that SL1 applies to the response from the device. The result of this transformation must be in a specific format from which SL1 parses the associated collection objects.

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:

Viewing the Requests in a Dynamic Application	9
Creating a Request	9
Defining XSLT Request Code	10
Defining XSLT Parser Code	12

<i>Substitution Characters</i>	14
<i>Editing a Request</i>	15
<i>Deleting a Request</i>	15


Viewing the Requests in a Dynamic Application

To view the existing requests in a SOAP or XSLT Dynamic Application:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Find the Dynamic Application you want to view the requests for. Select its wrench icon (). The **Dynamic Applications Properties Editor** page is displayed.
3. Select the **[Requests]** tab. The **Dynamic Applications Request Editor and Registry** page is displayed. The **Request Registry** pane at the bottom of the page displays the following information about each request:
 - **Request Name**. The name of the request.
 - **State**. The state of the request. Possible values are:
 - *Enabled*. SL1 will perform this request during collection for this Dynamic Application.
 - *Disabled*. SL1 will not perform this request during collection for this Dynamic Application.
 - **Sequence**. The order in which the requests in this Dynamic Application will be performed.
 - **ID**. The unique ID assigned to the request by SL1. The unique ID will always start with "req_".
 - **Date Edit**. The last time a user edited this request.

Creating a Request

To define a request for a SOAP or XSLT Dynamic Application:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Locate the Dynamic Application for which you want to define a request. Select its wrench icon (). The **Dynamic Applications Properties Editor** page is displayed.
3. Select the **[Requests]** tab. The **Dynamic Applications Request Editor and Registry** page is displayed.
4. Supply values in the following fields:
 - **Request Name**. Enter a name for the request.
 - **Execution Sequence**. Specifies the order in which SL1 should perform the requests in this Dynamic Application. Select a numeric value in this field. The request with the lowest **Execution Sequence** value will be performed first during collection, then the request with the next lowest **Execution Sequence** value, etc.
 - **Active State**. Specifies whether SL1 should perform this request during collection for this Dynamic Application. Choices are:

- *Enabled*. SL1 will perform this request during collection for this Dynamic Application.
- *Disabled*. SL1 will not perform this request during collection for this Dynamic Application.

NOTE: If a collection object that has a **Class Type** of [109] SOAP/XSLT Session ID is associated with a request, the request is executed only under certain conditions. For more information about Session ID collection objects, see the [Collection Objects](#) section.

5. Depending on the type and configuration of the Dynamic Application, define the request code:
 - If you are creating a request for a SOAP Dynamic Application, supply code in the **SOAP Request Code**. This will be the POST content that SL1 will use for this request. You can use substitution characters in the request code.
 - If you are creating a request for an XSLT Dynamic Application that does not consume cached responses, supply code in the **XSLT Request Code** and **XSLT Parser Code** fields. For a full description of these fields, see the [Defining XSLT Request Code](#) and [Defining XSLT Parser Code](#) sections.
 - If you are creating a request for an XSLT Dynamic Application that consumes cached responses, supply a value in the **Cached XSLT Request** field and supply code in the **XSLT Parser Code** field. For a full description of these fields, see the [Defining XSLT Request Code](#) and [Defining XSLT Parser Code](#) sections.
6. Select the **[Save]** button.

Defining XSLT Request Code

If your Dynamic Application has *No Caching* or *Cache Results* selected in the **Caching** drop-down list in the **Dynamic Applications Properties Editor** page, you must supply an XSL transformation in the **XSLT Request Code** field.

If your Dynamic Application has *Consume Cached Results* selected in the **Caching** drop-down list in the **Dynamic Applications Properties Editor** page, you must select a cached XSLT request in the **Cached XSLT Request** field.

The XSLT Request Code Field

When SL1 performs collection for an XSLT Dynamic Application, SL1 generates the POST content for each request by applying the XSL transformation you supply in the **XSLT Request Code** field to an XML document. The XML document contains the values collected for collection objects that have already been collected by the Dynamic Application during this poll period.

NOTE: If your Dynamic Application will be used with a credential that specifies an HTTP GET request, the result of the transformation specified in this field will not be used. However, SL1 will still apply the supplied XSL transformation to the XML document that contains the collected values. You must always supply a valid transformation in the **XSLT Request Code** field.

You must supply a valid XSL transformation in the **XSLT Request Code** field. SL1 will apply the specified XSL transformation to an XML document that has the following structure:

```
<objects>
```

```
<o_XXXX>
```

```
<i_Y></i_Y>
```

```
<i_Z></i_Z>
```

```
.
```

```
.
```

```
</o_XXXX>
```

```
<o_AAAA>
```

```
<i_B></i_B>
```

```
<i_C></i_C>
```

```
.
```

```
.
```

```
</o_AAAA>
```

```
</objects>
```

Where:

- o_XXXX and o_AAAA are collection object ID's
- i_Y and i_Z are indexes associated with values in the list of values collected for collection object o_XXXX.
- i_B and i_C are indexes associated with values in the list of values collected for collection object o_AAAA.

The XML file of objects contains only objects collected from other XSLT requests in the same polling period on the same device. For example, the XML file provided to the XSLT request with an **Execution Sequence** of "2" would

contain all the objects collected in the same polling period from the XSLT request with an **Execution Sequence** of "0" and all the objects collected in the same polling period from the XSLT request with an **Execution Sequence** of "1". The XML file provided to the XSLT request with an **Execution Sequence** of "0" will contain no collected objects.

You can use the substitution characters described in the [Substitution Characters](#) section in your **XSLT Request Code**.

The Cached XSLT Request Field

The **Cached XSLT Request** displays all XSLT requests defined in Dynamic Applications that have *Cache Results* selected in the **Caching** drop-down list in the **Dynamic Applications Properties Editor** page.

To generate the XML document that contains the collection objects associated with this XSLT request, SL1 transforms the currently cached response for the XSLT request you select in this field using the **XSLT Parser Code** defined for this XSLT request.

Defining XSLT Parser Code

The **XSLT Parser Code** for an XSLT request is XSL transformation that will be used by SL1 to transform the response of the SOAP request. SL1 parses the collection objects associated with the request from the transformed response.

In the **XSLT Parser Code** field, you must supply an XSL transformation that transforms the response from the device in to an XML document with the following structure, where `column_1` and `column_2` are element names.:

```
<response>
```

```
<row>
```

```
<column_1></column_1>
```

```
<column_2></column_2>
```

```
•
```

```
•
```

```
</row>
```

```
<row>
```

```
<column_1></column_1>
```

```
<column_2></column_2>
```

```
•
```

```
    .  
  </row>  
</response>
```

The element names are used by SL1 to parse collection object values from the XML document.

Each "<row>" block specifies an entry in the list of collection objects. For example, if the transformed response looks like this:

```
<response>  
  <row>  
    <label>Label 1</label>  
    <value>10</label>  
  </row>  
  <row>  
    <label>Label 2</label>  
    <value>20</label>  
  </row>  
  <row>  
    <label>Label 3</label>  
    <value>30</label>  
  </row>  
</response>
```

A list of three values (Label 1, Label 2, and Label 3) will be collected for the collection object associated with the *label* element.

A list of three values (10, 20, and 30) will be collected for the collection object associated with the *value* element.

You can use the substitution characters described in the [Substitution Characters](#) section in your **XSLT Parser Code**.

Substitution Characters

You can include substitution characters in *SOAP Request Code*, *XSLT Request Code*, and *XSLT Parser Code*. Depending on the type of substitution character that you use, SL1 will replace the substitution character with:

- The value of a collected object.
- A value specified in the associated credential.
- A property of the component device for which collection is being performed.

NOTE: Substitution characters for collected objects can be used only in *SOAP Request Code*.

Collection Object Substitution Characters

In *SOAP Request Code*, you can include the ID of one or more collection objects in the same Dynamic Application. All collection object IDs begin with "o_" (lowercase "oh", underscore), e.g. "o_123". To use a collection object ID in *SOAP Request Code*, the collection object must:

- Be associated with a request that has a lower *Execution Sequence* setting than the request that uses the collection object in the *SOAP Request Code*.
- Collect a single value, not a list of values.

Substitution Characters from Credentials

SOAP/XML credentials can specify up to four substitution characters that can be used in *SOAP Request Code*, *XSLT Request Code*, or *XSLT Parser Code*.

When you use substitution characters from a credential in *SOAP Request Code*, *XSLT Request Code*, or *XSLT Parser Code*, you must ensure that the credential(s) that are used with the Dynamic Application define values for these substitution characters:

- **%1.** SL1 will supply the value specified in the *Embed Value [%1]* field for the credential associated with the Dynamic Application.
- **%2.** SL1 will supply the value specified in the *Embed Value [%2]* field for the credential associated with the Dynamic Application.
- **%3.** SL1 will supply the value specified in the *Embed Value [%3]* field for the credential associated with the Dynamic Application.
- **%4.** SL1 will supply the value specified in the *Embed Value [%4]* field for the credential associated with the Dynamic Application.

Substitution Characters from Component Devices



In SOAP and XSLT Dynamic Applications that collect data from component devices, you can use the following variables in the *SOAP Request Code*, *XSLT Request Code*, or *XSLT Parser Code*:

- **%C**. Distinguished name. SL1 will supply the distinguished name of the component device SL1 is currently collecting data from.
- **%N**. Device name. SL1 will supply the device name of the component device SL1 is currently collecting data from.
- **%U**. Unique identifier. SL1 will supply the unique identifier of the component device SL1 is currently collecting data from.

For more information about Dynamic Component Mapping, see the *Dynamic Application Development* manual.



Editing a Request

To edit a request in a SOAP or XSLT Dynamic Application, perform the following steps:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Find the Dynamic Application you want to edit. Select its wrench icon (). The **Dynamic Applications Properties Editor** page is displayed.
3. Select the **[Requests]** tab. The **Dynamic Applications Request Editor and Registry** page is displayed.
4. In the **Request Registry** pane at the bottom of the page, locate the request you want to edit. Select the wrench icon () for the request you want to edit.
5. The fields in the top pane will be populated with the saved values for the selected request. Edit the values in one or more fields. For a description of each field, see the [Creating a Request](#) section.
6. Select the **[Save]** button to save your changes to the request. Select the **[Save As]** button to save the changes as a new request.

Deleting a Request

To delete a request from a SOAP or XSLT Dynamic Application, perform the following steps:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Find the Dynamic Application you want to edit. Select its wrench icon (). The **Dynamic Applications Properties Editor** page is displayed.
3. Select the **[Requests]** tab. The **Dynamic Applications Request Editor and Registry** page is displayed.
4. In the **Request Registry** pane at the bottom of the page, locate the request you want to delete. Select its bomb icon (). The request is deleted from SL1. You must edit all collection objects that were associated with the request to associate those collection objects with other requests in the Dynamic Application.

Chapter

3

Collection Objects



Overview

This chapter describes how to define collection objects for XML, SOAP, and XSLT Dynamic Applications.

This chapter describes only the fields specific to Defining a Collection Object for a XML, SOAP, and XSLT Dynamic Application. All the remaining fields, for both performance and configuration archetypes, are described in detail in the manual *Dynamic Application Development*. All other elements of XML, SOAP, and XSLT Collection Objects, such as presentation objects and alerts, behave in the same manner as other Dynamic Application types.

For details on other parts of XML, SOAP, and XSLT 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:

Protocol-Specific Fields for Collection Objects	17
Specifying XML Tags and SOAP Tags	18
Specifying XSLT Tags	19
Session ID Objects	20

Protocol-Specific Fields for Collection Objects

Similar to other Dynamic Application types, XML, SOAP, and XSLT Dynamic Applications contain collection objects. Collection objects for XML, SOAP, and XSLT Dynamic Applications have characteristics common to collection objects for other Dynamic Application types, such as naming, data typing, and grouping. This section describes the collection object fields that are specific to XML, SOAP, and XSLT Dynamic Applications.

XML Dynamic Applications

Collection objects for XML Dynamic Applications have the following unique fields:

- **XML Tags.** Specifies how SL1 should parse the collection object from the XML document that is requested for this Dynamic Application. For a full description of this field, see the [Specifying XML Tags and SOAP Tags](#) section of this chapter.
- **XML Document.** Appears only for Dynamic Applications that have *Consume cached results* selected in the **Caching** drop-down list in the **Dynamic Applications Properties Editor** page. This field specifies the Dynamic Application that retrieves the XML document that this collection object will be collected from. The list of Dynamic Applications in this field is limited to XML Dynamic Applications that have *Cache results* selected in the **Caching** drop-down list in the **Dynamic Applications Properties Editor** page.

SOAP Dynamic Applications

Collection objects for SOAP Dynamic Applications have the following unique fields:

- **SOAP Request.** Select the SOAP request that returns a value for this collection object.
 - For Dynamic Applications that have *No Caching* or *Cache Results* selected in the **Caching** drop-down list in the **Dynamic Applications Properties Editor** page, this drop-down list contains all SOAP requests defined in the same Dynamic Application.
 - For Dynamic Applications that have *Consume cached results* selected in the **Caching** drop-down list in the **Dynamic Applications Properties Editor**, this drop-down list contains requests defined in SOAP Dynamic Applications that have *Cache results* selected in the **Caching** drop-down list in the **Dynamic Applications Properties Editor** page.
- **SOAP Tags.** Specifies how SL1 should parse the collection object from the XML document that is collected using the request specified in the **SOAP Request** field. For a full description of this field, see the [Specifying XML Tags and SOAP Tags](#) section.

XSLT Dynamic Applications

Collection objects for XSLT Dynamic Applications have the following unique elements:

- **XSLT Request.** Select the XSLT request that returns a value for this collection object.
 - For Dynamic Applications that have *No Caching* or *Cache Results* selected in the **Caching** drop-down list in the **Dynamic Applications Properties Editor** page, this drop-down list contains all XSLT requests defined in the same Dynamic Application.
 - For Dynamic Applications that have *Consume cached results* selected in the **Caching** drop-down list in the **Dynamic Applications Properties Editor**, this drop-down list contains requests defined in XSLT Dynamic Applications that have *Cache results* selected in the **Caching** drop-down list in the **Dynamic Applications Properties Editor** page.
- **XSLT Tags.** Specifies how SL1 should parse the collection object from the XML document that is returned by the request specified in the **XSLT Request** field. For a full description of this field, see the [Specifying XSLT Tags](#) section.

Specifying XML Tags and SOAP Tags

In the **XML Tags** field (for XML Dynamic Applications) or **SOAP Tags** field (for SOAP Dynamic Applications), you must specify how SL1 should parse a value (or values) from the returned XML document.

In this field, you must specify a series of XML tags that specify an element or attribute in the XML document.

Parsing an XML Element

To specify that SL1 should parse the value of an element (that is, a value that is inside an opening and closing XML tag), specify the series of opening XML tags that lead to that element. For example, suppose the XML document looks like this:

```
<LoginMonitor>
  <CustomerID>
    <AuthFailures>10</AuthFailures>
  </CustomerID>
</LoginMonitor>
```

If you wanted to parse the value of the `<AuthFailures>` element (in this example, the value "10"), you would enter the following in the **XML Tags** field or **SOAP Tags** field:

```
<LoginMonitor><CustomerID><AuthFailures>
```

To specify that SL1 should parse the value of an XML tag attribute, specify the series of opening XML tags that lead to the XML tag that contains the attribute. In the XML tag that contains the attribute, specify the attribute in the XML tag with the substitution character %V as the attribute value. For example, suppose the XML document looks like this:

```
<LoginMonitor>
  <CustomerID>
    <AuthFailures type="1">10</AuthFailures>
  </CustomerID>
</LoginMonitor>
```

If you wanted to parse the value of the "type" attribute from the <AuthFailures> tag (in this example, the value "1"), you would enter the following in the **XML Tags** field or **SOAP Tags** field:

```
<LoginMonitor><CustomerID><AuthFailures type="%V">
```

A collection object in an XML or SOAP Dynamic Application will return a list of values if the **XML Tags** or **SOAP Tags** specify an element or attribute that appears multiple times in the XML document. For example, if the XML document looks like this:

```
<LoginMonitor>
  <CustomerID>
    <AuthFailures type="1">10</AuthFailures>
  </CustomerID>
  <CustomerID>
    <AuthFailures type="2">20</AuthFailures>
  </CustomerID>
</LoginMonitor>
```

The example collection objects in this section will both collect a list of two values.

Specifying XSLT Tags

Unlike XML documents in XML and SOAP Dynamic Applications, the XML document returned by an XSLT request is in a specific format. To specify how SL1 should parse a value (or values) from the XML document returned by an XSLT request, enter the element tag name for the value you want to collect in the **XSLT Tags** field.

For example, suppose the result of the XSLT Parser transformation is:

```
<response>
```

```
<row>
  <cpupercent>90</cpupercent>
</row>
</response>
```

If you wanted to parse the value of the `<cpupercent>` element (in this example, the value "90"), you would enter "cpupercent" in the **XSLT Tags** field.

NOTE: Do not include the less-than (<) or greater-than (>) characters in the **XSLT Tags** field.

Session ID Objects

For SOAP and XSLT Dynamic Applications, an additional option is available in the **Class Type** field for collection objects: *[109] SOAP/XSLT Session ID*.

Some SOAP servers require that a management system like SL1 use a unique, persistent session ID every time the management system authenticates. This means that when SL1 monitors such a SOAP server, the same session ID value must be used for every poll period.

You can tell SL1 to use the same session ID for every poll period by using the *[109] SOAP/XSLT Session ID* class type. When a collection object uses this class type:

- The request that returns this collection object will be considered a "login" request to the SOAP server. Usually, this request will be performed before all other requests in the Dynamic Application. This SOAP or XSLT request must not return collection objects that do not use the *[109] SOAP/XSLT Session ID* class type.
- At the start of each poll period, SL1 checks to see if a value has previously been collected and stored for this collection object:
 - If a value has been previously collected and stored, SL1 **does not perform the SOAP or XSLT request the collection object is associated with**. The previously collected value is used when this collection object is substituted in to the other requests (for SOAP Dynamic Applications) or when this collection object appears in the XML document that contains all collected values (for XSLT Dynamic Applications).
 - If a value has not been previously collected and stored, SL1 performs the SOAP or XSLT request the collection object is associated with as normal. The collected value is stored for use in future polling periods.
- If a previously collected value is used to populate the collection object and one of the subsequent requests returns a SOAP fault, SL1 will:

- Perform the SOAP request that retrieves the SOAP/XSLT Session ID collection object, storing the new value for use in future polling periods.
- Resume collection by retrying the request that returned the SOAP fault.

NOTE: The request that returns a SOAP/XSLT Session ID collection object will be performed only once per poll period. After the request that returns a SOAP/XSLT Session ID collection object is performed, subsequent request that return SOAP faults will not be retried.

Example



A

Creating an XML Dynamic Application

Overview

This chapter describes how to create and test an XML Performance Dynamic Application.

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:

Creating the Dynamic Application	23
Testing the Dynamic Application	27

Creating the Dynamic Application

In this example, the Dynamic Application will collect data from a Dial Number Identification Service on a transaction switch. To define the properties for the Dynamic Application, perform the following steps:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the **[Actions]** button, then select *Create New Dynamic Application*. The **Dynamic Applications Create New Application** page appears.
3. Supply values in the following fields:
 - **Application Name**. Enter "Dial Number Identification Service" in this field.
 - **Application Type**. Select *XML 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.

Defining XML Data-Points to Monitor

After defining the properties for the XML Dynamic Application, we must determine what data-points are available for the Dynamic Application to collect. Each data point will be used to create a collection object.

This example Dynamic Application will collect data from an XML document that looks like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<TS-IENMonitor>
```

```
<DNIS>
```

```
<DNIS-ID>997</DNIS-ID>
```

```
<X25CallsSuccess>100</X25CallsSuccess>
```

```
<X25CallsFail>10</X25CallsFail>
```

```
<AuthsSuccess>100</AuthsSuccess>
```

```
<AuthsFail>10</AuthsFail>
```

```
<AuthsFailRate>10</AuthsFailRate>
```

```
<SettlementsSuccess>100</SettlementsSuccess>
```

```
<SettlementsFail>10</SettlementsFail>
```

```
<SettlementsFailRate>10</SettlementsFailRate>
</DNIS>
<DNIS>
  <DNIS-ID>998</DNIS-ID>
  <X25CallsSuccess>100</X25CallsSuccess>
  <X25CallsFail>10</X25CallsFail>
  <AuthsSuccess>100</AuthsSuccess>
  <AuthsFail>10</AuthsFail>
  <AuthsFailRate>10</AuthsFailRate>
  <SettlementsSuccess>100</SettlementsSuccess>
  <SettlementsFail>10</SettlementsFail>
  <SettlementsFailRate>10</SettlementsFailRate>
</DNIS>
<DNIS-Summary>
  <X25CallsTotal>220</X25CallsTotal>
  <AuthsTotal>110</AuthsTotal>
  <SettlementsTotal>110</SettlementsTotal>
</DNIS-Summary>
</TS-IENMonitor>
```


We will define the following collection objects in SL1:

Name	XML Tag	Description
DNIS ID	<DNIS -ID>	<p>DNIS stands for dialed number identification service. DNIS is a service sold by telecommunications companies to corporate clients that lets them determine which telephone number was dialed by a customer. For example, a company may have a different toll free number for each product line it sells. If a call center is handling calls for multiple product lines, the switch that receives the class can examine the DNIS, then play the appropriate recorded greeting.</p> <p>The collection object "DNIS ID" specifies the phone number called by customers</p>
Successful X25 Calls	<X25CallsSuccess>	<p>X.25 is a data communications protocol developed to describe how data passes into and out of public data communication networks. The protocol is used primarily by telephone companies.</p> <p>The collection object "Successful X25 Calls" specifies how many incoming and outgoing phone calls using X25 were successfully connected.</p>
Failed X25 Calls	<X25CallsFail>	<p>The collection object "Failed X25 Calls" specifies how many incoming and outgoing phone calls using X25 were not successfully connected.</p>
Successful Authentications	<AuthsSuccess>	<p>Authentication means the verification of the identity of a person or process placing the call.</p> <p>The collection object "Successful Authentication" specifies how many calls were successfully authenticated.</p>
Failed Authentications	<AuthsFail>	<p>The collection object "Failed Authentication" specifies how many calls were not successfully authenticated.</p>
Authentication Failure Rate	<AuthsFailRate>	<p>The collection object "Authentication Failure Rate" specifies the percentage of authentications that failed.</p>
Successful Settlements	<SettlementsSuccess>	<p>A Settlement is the process by which merchant banks and cardholder banks exchange financial data resulting from sales transactions, cash disbursements, and merchandise credits.</p> <p>The collection object "Successful Settlements" specifies how many settlement transactions were completed successfully.</p>

Name	XML Tag	Description
Failed Settlements	<SettlementsFail>	The collection object "Failed Settlements" specifies how many settlement transactions were not completed successfully.
Settlement Failure Rate	<SettlementsFailRate>	The collection object "Settlement Failure Rate" specifies the percentage of settlements that failed.
Total X25 Calls	<X25CallsTotal>	The collection object "Total X25 Calls" specifies the total number of successful and failed, incoming and outgoing phone calls using X25 on all monitored DNIS numbers.
Total Authentications	<AuthsTotal>	The collection object "Total Authentications" specifies the total number of successful and failed authentications on all monitored DNIS numbers.
Total Settlements	<SettlementsTotal>	The collection object Total Settlements specifies the total number of successful and failed settlements on all monitored DNIS numbers.

Defining the Collection Objects

After determining what data-points are available, you can define your collection objects. To define the collection objects:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the wrench icon () for the *Dial Number Identification Service* Application.
3. Select the **[Collections]** tab. The **Dynamic Applications | Collections Objects** page appears.
4. To create the collection object for DNIS ID, supply values in the following fields:
 - **Object Name.** Enter "DNIS ID" in this field.
 - **XML Tags.** In this field you specify the element or attribute that should be parsed from the XML document for this collection object. Enter "<TS-IENMonitor> <DNIS> <DNIS-ID>" in this field
 - **Class Type.** Select *Label (Always Polled)*. This value will be used to label the graph lines on the performance graph.
 - **Group Number.** We selected *Group 1*. All collection objects that appear inside the <DNIS> section of the XML document will be in this group. Including all these collection objects in the same group means that this collection object will be used to label the graph lines on every performance graph that uses these collection objects.
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.

- For the remaining nine collection objects, follow the example above. Once complete, the **Collection Object Registry** will look like this:

Collection Object Registry							
Object Name	Class Type	Class ID	XML Tags	Group	ID	Edit Date	<input type="checkbox"/>
1. AuthsSuccess	Performance Gauge	4	<TS-ENMonitor><DNIS><AuthsSuccess>	1	o_8255	2012-06-21 16:50:42	<input type="checkbox"/>
2. AuthsTotal	Performance Gauge	4	<TS-ENMonitor><DNIS-Summary><AuthsTotal>	2	o_8256	2012-06-21 16:51:32	<input type="checkbox"/>
3. Call Fails	Performance Gauge	4	<TS-ENMonitor><DNIS><X25CallsFail>	1	o_8254	2012-06-21 16:50:11	<input type="checkbox"/>
4. Call Success	Performance Gauge	4	<TS-ENMonitor><DNIS><X25CallsSuccess>	1	o_8257	2012-06-21 16:52:10	<input type="checkbox"/>
5. DNIS ID	Label (Always Polled)	104	<TS-ENMonitor><DNIS><DNIS-ID>	1	o_8253	2012-06-21 16:48:24	<input type="checkbox"/>
6. Settlement Success	Performance Gauge	4	<TS-ENMonitor><DNIS><SettlementsSuccess>	1	o_8258	2012-06-21 16:52:48	<input type="checkbox"/>
7. SettlementsFail	Performance Gauge	4	<TS-ENMonitor><DNIS><SettlementsFail>	1	o_8259	2012-06-21 16:53:19	<input type="checkbox"/>
8. SettlementsFailRate	Performance Gauge	4	<TS-ENMonitor><DNIS><SettlementsFailRate>	1	o_8260	2012-06-21 16:53:51	<input type="checkbox"/>
9. SettlementsTotal	Performance Gauge	4	<TS-ENMonitor><DNIS-Summary><SettlementsTotal>	2	o_8261	2012-06-21 16:54:37	<input type="checkbox"/>
10. X25CallsTotal	Performance Gauge	4	<TS-ENMonitor><DNIS-Summary><X25CallsTotal>	2	o_8262	2012-06-21 16:55:09	<input type="checkbox"/>

[Select Action]

NOTE: Select **Performance Gauge** as the **Class Type** for the remaining collection objects, because these collection objects contain performance values that can go up or down between poll periods. Include every collection object in **Group 1**, except for the **Total X25 Calls**, **Total Settlements**, and **Total Authentications**. Because these values must not be associated with the "DNS ID" labels, group these three collection objects in **Group 2**.

Creating the Presentation Object

When you create a collection object in a Performance Dynamic Application, SLI automatically creates a presentation object that corresponds to that collection object. You must enable these presentation objects for SLI to generate a graph using those presentation objects. To enable a presentation object:

- Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
- Select the wrench icon () for the *Dial Number Identification Service* Application.
- Select the **[Presentations]** tab. The **Dynamic Applications Presentation Objects** page appears.
- In the **Dynamic Applications Presentation Objects** page, select the wrench icon () for the presentation object you want to enable.
- In the **Active State** drop-down, select *Enabled*.
- Select the **[Save]** button.
- Repeat the steps above for the remaining nine presentation objects.

Testing the Dynamic Application

To test the *Dial Number Identification Service* Dynamic Application in this example, you must configure a web server to host the XML document. The web server must:

- Return the XML document listed in the [Defining XML Data-Points to Monitor](#) section in response to a GET request.
- Be discoverable by SL1 as an SNMP device or a Non-SNMP device.

This example uses an Administration Portal as the web server. The root directory for an Administration Portal is:

```
/usr/local/silo/gui/ap/www/
```

For this example:

- A directory called "xml" was created in the root directory of the Administration Portal.
- A file called "ienmonitor.xml", which contains the XML listed in the [Defining XML Data-Points to Monitor](#) section, was created in the "xml" directory.
- The permissions on the "xml" directory and the "ienmonitor.xml" file were changed to 755.
- The Administration Portal was discovered in SL1.

Creating a Credential


For SL1 to collect data for the *Dial Number Identification Service* Dynamic Application from a test device, you must create a SOAP/XML credential. To create the SOAP/XML credential:

1. Go to the **Credential Management** page (System > Manage > Credentials).
2. Select the **[Create]** button, then select *SOAP/XML Host Credential* from the drop-down list. The **Create New CURL/SOAP Credential** page appears.
3. Supply values in the following fields:
 - **Profile Name.** Enter "Dial Number XML".
 - **Method.** Select *GET* in this field.
 - **URL.** Enter the URL of the XML document. Use the "%D" substitution character to supply the IP address of the test device. This example uses the URL "http://%D/xml/ienmonitor.xml".
4. For this example, you can leave the remaining fields set to the default values.
5. Select the **[Save]** button.

Manually Aligning the Dynamic Application to the Test Device

After you have configured the test device, you can align the *Dial Number Identification Service* Dynamic Application to the device. After aligning the Dynamic Application to the device, SL1 will collect data and we can view the Presentation objects we defined in the [Creating a Presentation Object](#) section.


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 test device you configured for this example. Select its 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*. The **Dynamic Application Alignment** page appears.
5. In the Dynamic Applications pane, select the Dial Number Identification Service Dynamic Application. In the Credentials pane, select Dial Number XML.
6. Select the **[Save]** button to assign the credential.

Viewing the Reports

After the Dynamic Application has collected the data specified in the collection objects, you can view the performance report for the test device. To view the performance report for the test device with the *Dial Number Identification Service* Dynamic Application aligned to it:

1. From the **Dynamic Application Collections** page, select the **[Reset]** button to update the page with the latest information.
2. Locate the *Dial Number Identification Service*. If the graph icon () is colored, the performance graph is available. Select the graph icon for the presentation object you want to view.

Or:

1. Go to the **Device Manager** page (Registry > Devices > Device Manager).
2. In the **Device Manager** page, find the test device you aligned the *Dial Number Identification Service* application to. Select the device's graph icon () .
3. The **Device Summary** page appears. Select the **[Performance]** tab.
4. In the left NavBar, select *Dial Number Identification Service*, then select the presentation object you want to view. For example, select **AuthsSuccess**.
5. The AuthsSuccess report is displayed.
 - You can mouse over different data points on the report, and the report will display the collected value for the given time.
 - The values for each label object are displayed in the graph key at the bottom of the report.
6. To learn more about device performance reports, see the manual **Monitoring Device Infrastructure Health**.

Example

2

Dynamic Component Mapping & Caching with XSLT

Overview

The following sample describes the development of three XSLT Dynamic Applications. The Dynamic Applications in this example demonstrate:

- How to create Dynamic Applications to discover component devices using Dynamic Component Mapping.
- How to use caching to associate performance data with component devices.

The Dynamic Applications in this example collect component device information from a hypothetical management system. For simplicity, the management system used in this example does not require some features that would typically be used in XSLT Dynamic Applications:

- The management system reports all the necessary data in a single static XML document. This allows the Dynamic Applications to make only one request.
- The management system returns the XML document in response to a GET request. This means that the Dynamic Applications do not require XSLT Request code. This simplification also means you can try the Dynamic Applications described in this example by configuring a Web Server to act as the management system.

The XML document returned by the management system looks like this:

```
<xml>
```

```
<component>
```

```
<name>component-one</name>
```

```
<id>00001</id>
```

```
<cpu>80</cpu>
```

```
<memory>80</memory>
```

```
</component>
```

```
<component>
```

```
<name>component-two</name>
```

```
<id>00002</id>
```

```
<cpu>60</cpu>
```

```
<memory>60</memory>
```

```
</component>
```

```
<component>
```

```
<name>component-three</name>
```

```
<id>00003</id>
```

```
<cpu>70</cpu>
```

```
<memory>70</memory>
```

```
</component>
```

```
<component>
```

```
<name>component-four</name>
```

```
<id>00004</id>
```

```
<cpu>55</cpu>
```

```
<memory>55</memory>
```

```
</component>
```

```
<component>
```

```
<name>component-five</name>
```

```
<id>00005</id>
```

```
<cpu>20</cpu>  
  
<memory>20</memory>  
  
</component>  
  
</xml>
```

Using this XML document, the Dynamic Applications in this example:

- Cache the returned XML document, so that the Dynamic Applications must make only a single request to the management system during a polling period.
- Retrieves the *name* and *id* values for each component to model each component as a separate device in SL1.
- Use the *cpu* and *memory* values to create performance graphs for each component device. The Dynamic Application that collects the CPU and Memory data will be aligned with each component device separately; the performance graph for each component device will show the CPU and Memory statistics for that component device only.

NOTE: Because the XML document used in this example is static, the values on the CPU and Memory graphs for each component device will be constant.

Design

This example uses three Dynamic Applications:

- **Example Dynamic Component Mapping General.** This Dynamic Application:
 - Requests and caches the XML page from the management system.
 - Contains a discovery object so that SL1 can automatically align the Dynamic Application with the management system during discovery.
 - Contains no other collection objects. Remember that a Dynamic Application that caches responses cannot include collection objects that need to be collected at regular intervals.

- **Example Dynamic Component Mapping Discovery.** This Dynamic Application:
 - Contains a discovery object so that SL1 can automatically align the Dynamic Application with the management system during discovery.
 - Uses the cached response collected by the **Example Dynamic Component Mapping General** Dynamic Application to collect the Name and ID values from the XML document.
 - Uses the Name and ID values to model the component devices. The Name value populates the *Device Name* component identifier, and the ID value populates the *Unique Identifier* component identifier.
 - Is associated with a Device Class. SL1 assigns that device class to each component device that is created.
 - Automatically aligns the **Example Component Performance** Dynamic Application to each component device that is created.

NOTE: The **Example Dynamic Component Mapping Discovery** meets the minimum requirements for creating component devices by including a *Device Name* component identifier, a *Unique Identifier* component identifier, and an associated device class.

- **Example Component Performance.** This Dynamic Application:
 - Is automatically aligned to component devices by the **Example Dynamic Component Mapping Discovery** Dynamic Application.
 - Uses the cached response collected by the **Example Dynamic Component Mapping General** Dynamic Application to collect the CPU and Memory values from the XML document.
 - The XSLT Parser Code specified in this Dynamic Application uses the %U substitution (the *Unique Identifier* component identifier) to collect only the CPU and Memory values that are associated with the current component device.
 - Contains Presentation Objects to display the collected CPU and Memory values in a graph.

Creating the "Example Dynamic Component Mapping General" Dynamic Application

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, then select *Create New Dynamic Application™*. The **Dynamic Applications Create New Application** page appears.


3. Supply values in the following fields:
 - **Application Name.** Enter "Example Dynamic Component Mapping General" in this field.
 - **Application Type.** This example uses the XSLT protocol. Dynamic Applications of type *Performance* must include a presentation object to work correctly. Because this Dynamic Application does not include collection objects that can be used in a presentation object, this Dynamic Application is of type *Configuration*. Select *XSLT Config [17]* in this field.
 - **Poll Frequency.** To see data as quickly as possible, select *Every 1 Minute* in this field.
 - **Caching.** This Dynamic Application must cache the response from the management system. Select *Cache Results* in this field.
 - **Component Mapping.** This Dynamic Application does not include collection objects that are used to create component devices. Leave this checkbox unchecked.
4. For this example, you can leave the remaining fields set to the default values.
5. Select the **[Save]** button.

Adding the XSLT Request

The XSLT request in this Dynamic Application retrieves the XML document from the management system. The retrieved XML document is cached for use by the other Dynamic Applications in this example.

The XSLT request in this Dynamic Application contains XSLT Parser Code that transforms the XML document to collect the discovery object for this Dynamic Application. The XSLT Parser code applies only to this Dynamic Application. The other Dynamic Applications in this example contain different XSLT Parser code that will perform different transformations on the cached XML document.

To create the XSLT request for this Dynamic Application, perform the following steps:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the wrench icon () for the **Example Dynamic Component Mapping General** Dynamic Application. The **Dynamic Applications Properties Editor** page appears.
3. Select the **[Requests]** tab. The **Dynamic Applications Request Editor and Registry** page appears.
4. Supply values in the following fields:
 - **Request Name.** Enter "Get Full Document" in this field.
 - **Execution Sequence.** This Dynamic Application contains only one XSLT request. Select *0* in this field.
 - **Active State.** This XSLT request must be executed to collect data. Select *Enabled* in this field.
 - **XSLT Request Code.** Because SL1 performs an HTTP GET request to retrieve the XML document from the management system, we are not required to generate an XML document to POST to the management system. However, SL1 still performs a transformation using the **XSLT Request Code** before performing the request. Therefore, the XSLT Request Code used in this example performs a transformation that outputs a blank XML document. Enter the following code in this field:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
<xsl:output method="xml" version="1.0" encoding="iso-8859-1"
indent="yes"/>
```

```
</xsl:stylesheet>
```

- **XSLT Parser Code.** The discovery object in this Dynamic Application determines whether the returned XML document contains at least one "<component>" element that contains a value for the "<name>" element. Therefore, the **XSLT Parser Code** must transform the returned XML document into an XML document with the following structure:

```
<response>
```

```
<row>
```

```
<name>name value</name>
```

```
</row>
```

```
</response>
```

The **XSLT Parser Code** uses a single **xsl:value-of** element to select the first component name that appears in the returned XML document. The component name is inserted inside the required structure. Enter the following code in this field:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
<xsl:output method="xml" version="1.0" encoding="iso-8859-1"
indent="yes"/>
```

```
<xsl:template match="/">
```

```
<response>
```

```
<row>
```

```
<name>
```

```
<xsl:value-of select="xml/component/name"/>
```

```
</name>
```

```
</row>
</response>
</xsl:template>
</xsl:stylesheet>
```

5. Select the **[Save]** button.

Adding the Discovery Object

This Dynamic Application includes one discovery object that tells SL1 to automatically align the Dynamic Application to the management system. Because a Dynamic Application that caches responses cannot include collection objects that need to be collected at regular intervals, there are no other collection objects in this Dynamic Application. The discovery object in this Dynamic Application will determine whether the returned XML document contains at least one "<component>" element that contains a value for the "<name>" element.

To create the discovery object for this Dynamic Application, perform the following steps:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the wrench icon (🔧) for the **Example Dynamic Component Mapping Discovery** 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" in this field.
 - **XSLT Tags.** The value you enter in this field must correspond with the name of an XML tag in the transformed response. Enter "name" in this field. The XSLT Request that you defined in the previous section returns the following transformed response:

```
<response>
  <row>
    <name></name>
  </row>
</response>
```

This discovery object looks for the presence of a component name in the transformed response.

- **Class Type.** Select *100 Discovery* in this field. This specifies that the object is a discovery object.
 - **XSLT Request.** Select the *Get Full Document* request you created in the previous section.
5. For this example, you can leave the remaining fields set to the default values.

6. Select the **[Save]** button. After you save the discovery object, the form will change to show additional fields that apply only to discovery objects. These additional fields are not used by this example.

Creating the "Example Dynamic Component Mapping Discovery" Dynamic Application

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, then select *Create New Dynamic Application™*. The **Dynamic Applications Create New Application** page appears.
3. Supply values in the following fields:
 - **Application Name**. Enter "Example Dynamic Component Mapping Discovery" in this field.
 - **Application Type**. This example uses the XSLT protocol. The data collected by this Dynamic Application, the component name and ID, are best displayed in tabular format, so this example uses the Configuration type. Select *XSLT Config [17]* in this field.
 - **Poll Frequency**. To see data as quickly as possible, select *Every 1 Minute* in this field.
 - **Caching**. This Dynamic Application uses the XML document collected by the **Example Dynamic Component Mapping General** Dynamic Application. Select *Consume Cached Results* in this field.
 - **Component Mapping**. This Dynamic Application contains collection objects that SL1 will use to create component devices. Check this checkbox.
4. For this example, you can leave the remaining fields set to the default values.
5. Select the **[Save]** button.

Adding the XSLT Requests

This Dynamic Application includes two XSLT requests. Each XSLT request contains XSLT Parser Code that transforms the XML document cached by the **Example Dynamic Component Mapping General** Dynamic Application. The transformed XML document contains:

- For the first request, the discovery object for this Dynamic Application. The discovery object in this Dynamic Application is the same as the discovery object in the **Example Dynamic Component Mapping General** Dynamic Application.
- For the second request, the component name and ID values that SL1 uses to create the component devices.

To create the XSLT requests for this Dynamic Application, perform the following steps:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the wrench icon () for the **Example Dynamic Component Mapping Discovery** Dynamic Application. The **Dynamic Applications Properties Editor** page appears.

3. Select the **[Requests]** tab. The **Dynamic Applications Request Editor and Registry** page appears.
4. To create the first request, supply values in the following fields:
 - **Request Name.** Enter "Discovery" in this field. This request returns the XML document that contains the discovery object for this Dynamic Application.
 - **Execution Sequence.** This Dynamic Application does not require SL1 to execute the XSLT requests in a specific order; however, the two requests must have unique values for this field. Select *0* in this field.
 - **Active State.** This XSLT request must be executed to collect data. Select *Enabled* in this field.
 - **Cached XSLT Request.** This field specifies the cached XML document to transform. Locate the **Example Dynamic Component Mapping Discovery** Dynamic Application and select *Get Full Document* from this list.
 - **XSLT Parser Code.** This Dynamic Application uses the same discovery object as is used in the **Example Dynamic Component Mapping General** Dynamic Application. Therefore, this request uses the same **XSLT Parser Code** as the request in the **Example Dynamic Component Mapping General** Dynamic Application. Enter the following code in this field:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="xml" version="1.0" encoding="iso-8859-1"
indent="yes"/>

<xsl:template match="/">

  <response>

    <row>

      <name>

        <xsl:value-of select="xml/component/name"/>

      </name>

    </row>

  </response>

</xsl:template>

</xsl:stylesheet>
```

5. Select the **[Save]** button.
6. Select the **[Reset]** button to clear the form fields.

7. To create the second request, supply values in the following fields:

- **Request Name.** Enter "Get Components" in this field. This request will return the XML document that contains the name and ID values for the component devices.
- **Execution Sequence.** This Dynamic Application does not require SL1 to execute the XSLT requests in a specific order; however, the two requests must have unique values for this field. Select *1* in this field.
- **Active State.** This XSLT request must be executed to collect data. Select *Enabled* in this field.
- **Cached XSLT Request.** This field specifies the cached XML document to transform. Locate the **Example Dynamic Component Mapping Discovery** Dynamic Application and select *Get Full Document* from this list.
- **XSLT Parser Code.** Two collection objects will be parsed from the output of this XSLT request: the component name and the component ID. The XML document returned by the management system might contain information about multiple component devices, so both the name and ID collection objects might be a list of values. The association of component name to component ID must be maintained for each pair of values, that is, they must reside in the same row in the transformed XML document. Therefore, the **XSLT Parser Code** must transform the returned XML document into an XML document with the following structure:

```
<response>

  <row>

    <name>name value for the first component device in the
    response</name>

    <id>ID value for the first component device in the
    response</id>

  </row>

  <row>

    <name>name value for the second component device in the
    response</name>

    <id>ID value for the second component device in the
    response</id>

  </row>

  .

  .

  .
```

```

<row>

  <name>name value for the last component device in the
  response</name>

  <id>ID value for the first component device in the
  response</id>

</row>

</response>

```

The **XSLT Parser Code** uses an **xsl:for-each** element to iterate through the component devices in the returned XML document, creating a `<row>` block for each component device. The component name and ID values are inserted inside the required structure. Enter the following code in this field:

```

<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="xml" version="1.0" encoding="iso-8859-1"
indent="yes"/>

<xsl:template match="/">

  <response>

    <xsl:for-each select="xml/component">

      <row>

        <xsl:element name="name">

          <xsl:value-of select="name"/>

        </xsl:element>

        <xsl:element name="id">

          <xsl:value-of select="id"/>

        </xsl:element>

      </row>

```



```
</xsl:for-each>
</response>
</xsl:template>
</xsl:stylesheet>
```

8. Select the **[Save]** button.


Adding the Collection Objects

This Dynamic Application includes a discovery object that tells SL1 to automatically align the Dynamic Application to the management system. The discovery object in this Dynamic Application determines whether the returned XML document contains at least one "<component>" element that contains a value for the "<name>" element.

To create a component device, a Dynamic Component Mapping Dynamic Application must:

- Collect an object that maps to the *Unique Identifier* component identifier. This Dynamic Application includes a collection object that parses the component ID from the XML document. The component ID maps to the *Unique Identifier*.
- Collect an object that maps to the *Device Name* component identifier. This Dynamic Application includes a collection object that parses the component name from the XML document. The component name maps to the *Device Name*.
- Be associated with a component device class. The device class for this Dynamic Application is described in the [Creating a Device Class for the Component Devices](#) section.

To create the collection objects for this Dynamic Application, perform the following steps:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the wrench icon () for the **Example Dynamic Component Mapping Discovery** 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 discovery object, supply values in the following fields:
 - **Object Name.** Enter "Discovery" in this field.
 - **XSLT Tags.** The value you enter in this field must correspond with the name of an XML tag in the transformed response. Enter "name" in this field. The "Discovery" XSLT Request you defined in the previous section returns the following transformed response:

```
<response>
<row>
<name></name>
```

```
</row>
```

```
</response>
```

This discovery object should look for the presence of a component name in the transformed response.

- **Class Type.** Select *100 Discovery* in this field. This specifies that the object is a discovery object.
- **XSLT Request.** Select the *Discovery* request you created in the previous section.

5. For this example, you can leave the remaining fields set to the default values.

6. Select the **[Reset]** button to clear the form fields.

7. To create the collection object for the component ID, supply values in the following fields:

- **Object Name.** Enter "Component ID" in this field.
- **XSLT Tags.** Enter "id" in this field. The "Get Components" XSLT Request you defined in the previous section returns the following `<row>` block for each component device:

```
<row>
```

```
<name></name>
```

```
<id></id>
```

```
</row>
```

This collection object parses the component ID from each `<row>` block.

- **Class Type.** Select *10 Config Character* in this field. To maintain the leading zeroes in the collected component ID values, the ID is stored as a string.
- **Group Number.** To maintain the association between the collected component ID and component name values, you must ensure that SL1 uses the same internal indexes for each list. SL1 maintains internal indexes for each group of collection objects. Therefore, the component ID and component name collection objects must be in the same group for SL1 to use the same internal index for both collection objects. Select *Group 1* in this field.
- **Index.** By default, SL1 assigns internal indexes to the list of collected values based on the order in which they appear in the response. Suppose that "component-four" is disabled on the management system. Suppose that when "component-four" is disabled, it no longer appears in the XML response from the management system. When "component-four" appeared in the XML response, the following internal indexes would have been assigned:
 - Index 0 = component-one
 - Index 1 = component-two
 - Index 2 = component-three

- Index 3 = component-four
- Index 4 = component-five

When "component-four" is disabled, it no longer appears in the XML response from the management system. During the next poll period, the index for "component-five" will change:

- Index 0 = component-one
- Index 1 = component-two
- Index 2 = component-three
- Index 3 = component-five

In cases where the order or size of the list of values might change, you must designate a collection object as an index. This example designates the component ID collection object as the index. The collection object that is designated as an index must meet the following requirements:

- The list of values returned by the collection object must be unique. In this example, the component ID is unique.
- If a previously collected value in the list of values appears in a new list of collected values, that collected value is associated with the same set of values. In this example, a set of values is a component ID/component name pair. When a previously collected component ID is collected again, it is always associated with the same component name; therefore, the component ID can be used as the index.

Select the **Index** checkbox.

- **XSLT Request.** Select the *Get Components* request you created in the previous section.
- **Component Identifiers.** Each value collected for this collection object will be used as the unique identifier for a component device. Select *Unique Identifier (%U)* in this field.

8. For this example, you can leave the remaining fields set to the default values.
9. Select the **[Save]** button.
10. Select the **[Reset]** button to clear the form fields.
11. To create the collection object for the component name, supply values in the following fields:
 - **Object Name.** Enter "Component Name" in this field.
 - **XSLT Tags.** Enter "name" in this field. The "Get Components" XSLT Request you defined in the previous section returns the following <row> block for each component device:

```
<row>
```

```
<name></name>
```

```
<id></id>
```

```
</row>
```

This collection object parses the component name from each <row> block.

- **Class Type.** Select *10 Config Character* in this field. The component name is a string.
- **Group Number.** To maintain the association between the collected values for component ID and component name, you must ensure that SL1 uses the same internal indexes for each list. SL1 maintains internal indexes for each group of collection object. Therefore, the component ID and component name collection objects must be in the same group for SL1 to use the same internal index for both collection objects. Select *Group 1* in this field.
- **Index.** Leave this checkbox unchecked. The collection object for the component ID has already been designated as the index for this group of collection objects.
- **XSLT Request.** Select the *Get Components* request you created in the previous section.
- **Component Identifiers.** Each value collected for this collection object will be used as the device name of a component devices. Select *Device Name (%N)* in this field.

12. For this example, you can leave the remaining fields set to the default values.

13. Select the **[Save]** button.

Creating a Device Class for the Component Devices

For SL1 to create a component device, the Dynamic Component Mapping Dynamic Application must include collection objects that map to the *Device Name* and *Unique Identifier* component identifiers. In addition, the same Dynamic Application must be associated with a component Device Class. When you associated a component Device Class with a Dynamic Application, you are telling SL1 to assign that device class to each component device that is created by the Dynamic Application.

To create a device class that is associated with the **Example Dynamic Component Mapping Discovery** Dynamic Application, perform the following steps:

1. Go to the **Device Class Editor** page (System > Customize > Device Classes).
2. In the **Device Class Editor** pane at the top of the page, supply values in the following fields:
 - **Device Type.** This device class will be assigned to component devices. Select *Component* in this field.
 - **Root Device.** The component devices in this example will not act as root devices. Leave this checkbox unchecked.
 - **Device Class.** Enter "Example" in this field.
 - **Description.** Enter "Component Device" in this field.
 - **Dynamic App Alignment.** Select *Example Dynamic Component Mapping Discovery* in this field.
3. Select the **[Save]** button.

Creating the "Example Component Performance" Dynamic Application

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, then select *Create New Dynamic Application™*. The **Dynamic Applications Create New Application** page appears.
3. Supply values in the following fields:
 - **Application Name.** Enter "Example Component Performance" in this field.
 - **Application Type.** This example uses the XSLT protocol. The data collected by this Dynamic Application will be displayed in a graph, so this example uses the Performance archetype. Select *XSLT Performance* in this field.
 - **Poll Frequency.** To see data as quickly as possible, select *Every 1 Minute* in this field.
 - **Caching.** This Dynamic Application will use the XML document collected by the **Example Dynamic Component Mapping General** Dynamic Application. Select *Consume Cached Results* in this field.
 - **Component Mapping.** Although this Dynamic Application will be aligned with component devices, this Dynamic Application does not include collection objects that are used to create component devices. Leave this checkbox unchecked.
4. For this example, you can leave the remaining fields set to the default values.
5. Select the **[Save]** button.

Adding the XSLT Request

This Dynamic Application includes one XSLT request. The XSLT request will contain XSLT Parser Code that transforms the XML document cached by the **Example Dynamic Component Mapping General** Dynamic Application. The transformed response will return the CPU and memory utilization values for the component device.

The XSLT request uses the "%U" substitution character to filter the cached XML document. Before performing the transformation, SL1 will replace "%U" with the unique identifier of the component device for which data is currently being collected. By filtering the cached XML document using the unique identifier, the transformed response will contain the CPU and Memory utilization only for the component device for which data is currently being collected.

This Dynamic Application will be aligned with each of the five component devices in this example. Therefore, during each poll, the transformation will be performed five times, resulting in five different sets of CPU and memory utilization data (one set for each component device).

To create the XSLT request for this Dynamic Application, perform the following steps:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the wrench icon (🔧) for the **Example Component Performance** Dynamic Application. The **Dynamic Applications Properties Editor** page appears.
3. Select the **[Requests]** tab. The **Dynamic Applications Request Editor and Registry** page appears.
4. Supply values in the following fields:
 - **Request Name.** Enter "Get CPU+Memory" in this field. This request will return the XML document that contains the CPU and memory utilization values for the component devices.
 - **Execution Sequence.** This Dynamic Application will contain only one XSLT request. Select *0* in this field.
 - **Active State.** This XSLT request must be executed to collect data. Select *Enabled* in this field.
 - **Cached XSLT Request.** This field specifies the cached XML document to transform. Locate the **Example Dynamic Component Mapping Discovery** Dynamic Application and select *Get Full Document* from this list.
 - **XSLT Parser Code.** Two collection objects will be parsed from the output from this XSLT request: the CPU utilization and the memory utilization. The transformed response must include the set of CPU and memory utilization values only for the component device for which data is currently being collected. Therefore, the **XSLT Parser Code** must transform the returned XML document into an XML document with the following structure:

```
<response>

  <row>

    <cpu>CPU utilization value</cpu>

    <memory>Memory utilization value</memory>

  </row>

</response>
```

The **XSLT Parser Code** uses an **xsl:for-each** element to iterate through the component devices in the returned XML document. On each iteration, an **xsl:if** element is used to determine whether the ID value in the XML document matches the component ID value of the component device for which data is currently being collected (the "%U" substitution). If the ID values match, the **<row>** block is created using the CPU and memory values that appear in the XML document. Enter the following code in this field:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
<xsl:output method="xml" version="1.0" encoding="iso-8859-1"
indent="yes"/>

<xsl:template match="/">

  <response>

    <xsl:for-each select="xml/component">

      <xsl:if test="id = '%U'">

        <row>

          <xsl:element name="cpu">

            <xsl:value-of select="cpu"/>

          </xsl:element>

          <xsl:element name="memory">

            <xsl:value-of select="memory"/>

          </xsl:element>

        </row>

      </xsl:if>

    </xsl:for-each>

  </response>


</xsl:template>

</xsl:stylesheet>
```

5. Select the **[Save]** button.

Adding the Collection Objects

This Dynamic Application contains one collection object for the CPU utilization value and one collection object for the memory utilization value. To create the collection objects for this Dynamic Application, perform the following steps:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the wrench icon () for the **Example Component 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 the CPU utilization, supply values in the following fields:
 - **Object Name.** Enter "CPU Usage" in this field.
 - **XSLT Tags.** Enter "cpu" in this field. The "Get CPU+Memory" XSLT Request you defined in the previous section returns the following XML structure:

```
<row>
<cpu></cpu>
<memory></memory>
</row>
```

This collection object parses the CPU utilization from the transformed response.

- **Class Type.** The CPU utilization is a number that can go up or down between polls. Select *4 Performance Gauge* in this field.
 - **XSLT Request.** Select the *Get CPU+Memory* request you created in the previous section.
5. For this example, you can leave the remaining fields set to the default values.
 6. Select the **[Save]** button.
 7. Select the **[Reset]** button to clear the form fields.
 8. To create the collection object for the memory utilization, supply values in the following fields:

```
<row>
<cpu></cpu>
<memory></memory>
</row>
```

This collection object parses the memory utilization from the transformed response.

- **Class Type.** The memory utilization is a number that can go up or down between polls. Select *4 Performance Gauge* in this field.
 - **XSLT Request.** Select the *Get CPU+Memory* request you created in the previous section.
9. For this example, you can leave the remaining fields set to the default values.

10. Select the **[Save]** button.

Editing the Presentation Objects

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. This example makes some minor changes to the presentation objects that were automatically created. To edit the presentation objects for this Dynamic Application, perform the following steps:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the wrench icon (🔧) for the **Example Component Performance** Dynamic Application. The **Dynamic Applications Properties Editor** page appears.
3. Select the **[Presentations]** tab. The **Dynamic Applications Presentation Objects** page appears.
4. Select the wrench icon for the **CPU Usage** presentation object. Edit the following fields:
 - **Active State.** Select *Enabled* in this field.
 - **Show as Percent.** The CPU utilization values are reported as percentage used. Select Yes in this field.
5. For this example, you can leave the remaining fields set to the default values.
6. Select the wrench icon for the **Memory Usage** presentation object. Edit the following fields:
 - **Active State.** Select *Enabled* in this field.
 - **Show as Percent.** The memory utilization values are reported as percentage used. Select Yes in this field.
7. For this example, you can leave the remaining fields set to the default values.
8. Select the **[Save]** button.

Automatically Aligning the Dynamic Application to Component Devices

When SL1 creates component devices using data collected by a Dynamic Application, SL1 can automatically align other Dynamic Applications to those component devices as they are created. In this example, the **Example Component Performance** Dynamic Application should be aligned to the component devices created by the **Example Dynamic Component Mapping Discovery** Dynamic Application.

The **Example Component Performance** Dynamic Application does not include a discovery object. Instead of a discovery object, we explicitly tell SL1 to automatically align the **Example Component Performance** Dynamic Application with each component device that is created with the **Example Dynamic Component Mapping Discovery** Dynamic Application.

To do this, perform the following steps:

1. Go to the **Dynamic Applications Manager** page (System > Manage > Applications).
2. Select the wrench icon (🔧) for the **Example Dynamic Component Mapping Discovery** Dynamic Application. The **Dynamic Applications Properties Editor** page appears.
3. Select the **[Component]** tab. The **Dynamic App Component Alignment** modal page appears:

4. Select the **Example Component Performance** Dynamic Application from the **Unaligned Dynamic Apps** list.
5. Select the >> button. The **Example Component Performance** Dynamic Application moves to the **Aligned Dynamic Apps** list.
6. Select the **[Save]** button.
7. SL1 will automatically align the **Example Component Performance** Dynamic Application with each component device created by the **Example Dynamic Component Mapping Discovery** Dynamic Application.

Using the Dynamic Applications

Configuring a Test Device

To test the Dynamic Applications in this example, you must configure a web server to act as the management system. The web server must:

- Return the XML document listed in the [Overview](#) section in response to a GET request.
- Be discoverable by SL1 as an SNMP device or a Non-SNMP device.

This example uses an Administration Portal as the web server. The root directory for an Administration Portal is:


```
/usr/local/silo/gui/ap/www/
```

For this example:

1. A directory called "xml" was created in the root directory of the Administration Portal.
2. A file called "components.xml", which contains the XML listed in the [Overview](#) section, was created in the "xml" directory.
3. The permissions on the "xml" directory and the "components.xml" file were changed to 755.

Editing the Device Class for the Test Device

You must edit the device class that SL1 will assign to the device that hosts the web server. The device class for the test device must be enabled as a **root device**. A **root device** is a device for which SL1 can create children component devices. Perform the following steps to enable the device class as a root device:

1. Go to the **Device Class Editor** page (System > Customize > Device Classes).
2. In the **Device Class Register** at the bottom of the page, locate the device class that SL1 will assign to the device that hosts the web server. In our example, the device that hosts the web server will be assigned the device class with the description "EM7 G3 Admin Portal". Select the wrench icon () for the device class.
3. Check the **Root Device** checkbox.
4. Select the **[Save]** button.

Creating a Credential

For SL1 to align the Dynamic Applications in this example to the device that hosts the web server, we must include a SOAP/XML credential in the discovery session. SL1 will use the SOAP/XML credential to collect the XML document from the web server. To create the credential for this example:

1. Go to the **Credential Management** page (System > Manage > Credentials).
2. Select the **[Create]** button, and then select *SOAP/XML Host Credential* from the drop-down list. The **Create New CURL/SOAP Credential** page is displayed:
3. Supply values in the following fields:
 - **Profile Name**. Enter "Components XML".
 - **Method**. Select *GET* in this field.
 - **URL**. Enter the URL of the XML document on the web server. Use the "%D" substitution character to supply the IP address of the test device. This example uses the URL "http://%D/xml/components.xml".
4. For this example, you can leave the remaining fields set to the default values.
5. Select the **[Save]** button.

Discovering the Test Device

To discover the device that hosts the web server:

1. Go to the **Discovery Control Panel** page (System > Manage > Classic Discovery).
2. Select the **[Create]** button. The **Discovery Session Editor** page is displayed in a new window:
3. Supply values in the following fields:
 - **IP Address Discovery List**. Enter the IP address of your device.
 - **SNMP Credentials**. If your device responds to SNMP, select the appropriate SNMP credential in this field.
 - **Other Credentials**. Select the *Components XML* credential you created in the previous section.
 - **Initial Scan Level**. Select at least *1. Initial Population of Apps* in this field.
 - **Discover Non-SNMP**. If your device does not respond to SNMP, check this checkbox.
4. For this example, you can leave the remaining fields set to the default values.
5. Select the **[Save]** button.
6. In the **Discovery Control Panel** page, select the **[Reset]** button. The discovery session you created will appear in the list of discovery sessions.
7. Select the lightning bolt icon (⚡) for the discovery session you created to run the discovery session. The Discovery Session log will appear.

Verifying the Dynamic Application Alignments

To verify that the discovery process aligned the appropriate Dynamic Applications to the test device and that SL1 has created the component devices, perform the following steps:

1. Go to the **Device Manager** page (Registry > Devices > Device Manager).
2. Select the wrench icon (🔧) for the device that hosts the web server. The **Device Properties** page is displayed in a new window.
3. Select the **[Logs]** tab. The **Device Logs & Messages** page is displayed.
4. Enter "Component" in the search bar and select the **[Search]** button. The search results should show:
 - The **Example Dynamic Component Mapping General** and **Example Dynamic Component Mapping Discovery** Dynamic Applications were aligned to the device during discovery.
 - That SL1 created child component devices. The logs will contain one message for each component device that SL1 created.

Viewing the Device Components Registry

The **Device Components** page (Registry > Devices > Device Components) displays a list of all root devices and component devices discovered by SL1. The **Device Components** page is similar to the **Device Manager** page (Registry > Devices > Device Manager) page. The **Device Components** page displays all root devices and component devices in an indented view, so you can easily view the hierarchy and relationships between child devices, parent devices, and root devices. You can expand or hide the child devices for each root device and for each parent device.

To view the device that hosts the web server and its child component devices in the **Device Components** page:

1. Go to the **Device Components** page (Registry > Devices > Device Components).
2. Select the plus icon (+) for the device that hosts the web server. The list is expanded to show the component devices.


Viewing the Component Device Map

The **Component Map** page allows you to view root devices and their children component devices in a graphical map. To view the test device and the child component devices in the **Component Map** page:

1. Go to the **Component Map** page (Classic Maps > Device Maps > Components).
2. If there are multiple root devices discovered in your SL1 system, select the appropriate device from the drop-down list in the upper right of the page. A map of the device that hosts the web server and the child component devices is displayed.

Viewing the Performance Graphs

To view the performance graphs generated by the **Example Component Performance** Dynamic Application:

1. Go to the **Device Manager** page (Registry > Devices > Device Manager).
2. Select the graph icon () for one of the component devices. The **Device Summary** page is displayed in a new window.
3. Select the **[Performance]** tab. In the NavBar to the left of the page, select Example Component Performance > CPU Usage. The CPU Usage graph is displayed. Because the XSLT request in this Dynamic Application filtered the returned XML document by component ID, the graph displays only the CPU data for this component.

Expanding this Example

If you would like to create additional Dynamic Component Mapping Dynamic Applications, here are some suggestions for ways you could try modifying this example:

- Although the caching feature and the Dynamic Component Mapping feature are typically used together, you are not required to do so. Try combining the **Example Dynamic Component Mapping General** and **Example Dynamic Component Mapping Discovery** Dynamic Applications in to a single Dynamic Application that does not use the caching feature but still creates component devices. Use the **XSLT Request Code** from the **Example Dynamic Component Mapping General** Dynamic Application to create the XSLT requests for your combined Dynamic Application.
- Modify the **Example Component Performance** Dynamic Application so that it does not use caching. Again, use the **XSLT Request Code** from the **Example Dynamic Component Mapping General** Dynamic Application to create the XSLT requests for the new Dynamic Application. For your new Dynamic Application to collect data, you must manually align a credential that uses the "%D" substitution in the **URL** field. When a credential that uses "%D" in the **URL** field is used with a component device, SL1 will substitute the IP address of the root device in to the URL.
- Modify the XML document on the test device to include additional layers in the component tree, i.e. include additional components that are children of the existing components. Create an additional Dynamic Application that creates component devices as children of the existing component devices. You can then add your new Dynamic Application to the list of Dynamic Applications that SL1 should automatically align to each component device created by the **Example Dynamic Component Mapping Discovery** Dynamic Application. The modified XML document might look like this:

```
<xml>

  <component>

    <name>component-one</name>

    <id>00001</id>

    <cpu>80</cpu>

    <memory>80</memory>

    <sub_component>

      <name>sub-component-one</name>
```

```
<id>10001</id>
</sub_component>
<sub_component>
  <name>sub-component-two</name>
  <id>10002</id>
</sub_component>
</component>
<component>
  <name>component-two</name>
  <id>00002</id>
  <cpu>60</cpu>
  <memory>60</memory>
  <sub_component>
    <name>sub-component-three</name>
    <id>10003</id>
  </sub_component>
</component>
<component>
  <name>component-three</name>
  <id>00003</id>
  <cpu>70</cpu>
  <memory>70</memory>
  <sub_component>
    <name>sub-component-four</name>
```

```
<id>10004</id>
</sub_component>
<sub_component>
  <name>sub-component-five</name>
  <id>10005</id>
</sub_component>
<sub_component>
  <name>sub-component-six</name>
  <id>10006</id>
</sub_component>
</component>
<component>
  <name>component-four</name>
  <id>00004</id>
  <cpu>55</cpu>
  <memory>55</memory>
</component>
<component>
  <name>component-five</name>
  <id>00005</id>
  <cpu>20</cpu>
  <memory>20</memory>
  <sub_component>
    <name>sub-component-six</name>
```

```
<id>10006</id>
```

```
</sub_component>
```

```
<sub_component>
```

```
<name>sub-component-seven</name>
```

```
<id>10007</id>
```

```
</sub_component>
```

```
</component>
```

```
</xml>
```


© 2003 - 2024, ScienceLogic, Inc.

All rights reserved.

LIMITATION OF LIABILITY AND GENERAL DISCLAIMER

ALL INFORMATION AVAILABLE IN THIS GUIDE IS PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED. SCIENCELOGIC™ AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT.

Although ScienceLogic™ has attempted to provide accurate information on this Site, information on this Site may contain inadvertent technical inaccuracies or typographical errors, and ScienceLogic™ assumes no responsibility for the accuracy of the information. Information may be changed or updated without notice. ScienceLogic™ may also make improvements and / or changes in the products or services described in this Site at any time without notice.

Copyrights and Trademarks

ScienceLogic, the ScienceLogic logo, and EM7 are trademarks of ScienceLogic, Inc. in the United States, other countries, or both.

Below is a list of trademarks and service marks that should be credited to ScienceLogic, Inc. The ® and ™ symbols reflect the trademark registration status in the U.S. Patent and Trademark Office and may not be appropriate for materials to be distributed outside the United States.

- ScienceLogic™
- EM7™ and em7™
- Simplify IT™
- Dynamic Application™
- Relational Infrastructure Management™

The absence of a product or service name, slogan or logo from this list does not constitute a waiver of ScienceLogic's trademark or other intellectual property rights concerning that name, slogan, or logo.

Please note that laws concerning use of trademarks or product names vary by country. Always consult a local attorney for additional guidance.

Other

If any provision of this agreement shall be unlawful, void, or for any reason unenforceable, then that provision shall be deemed severable from this agreement and shall not affect the validity and enforceability of any remaining provisions. This is the entire agreement between the parties relating to the matters contained herein.

In the U.S. and other jurisdictions, trademark owners have a duty to police the use of their marks. Therefore, if you become aware of any improper use of ScienceLogic Trademarks, including infringement or counterfeiting by third parties, report them to Science Logic's legal department immediately. Report as much detail as possible about the misuse, including the name of the party, contact information, and copies or photographs of the potential misuse to: legal@sciencelogic.com. For more information, see <https://sciencelogic.com/company/legal>.

ScienceLogic

800-SCI-LOGIC (1-800-724-5644)

International: +1-703-354-1010