



---

# Monitoring with the SL1 Agent

SL1 version 12.3.0

---

# Table of Contents

<b>Introduction to the SL1 Agent</b> .....	<b>6</b>
What is the SL1 Agent? .....	7
How the SL1 Agent Works in Different Environments .....	7
What System Vital Metrics Can the SL1 Agent Collect? .....	10
Metric Descriptions .....	10
Supported Data Collection Methods for Monitoring Windows .....	12
Supported Data Collection Methods for Monitoring Linux .....	12
Extensible Collection .....	13
Agent PowerPacks .....	14
Agent-Compatible PowerPacks .....	16
Windows Devices .....	16
Java Management Extensions (JMX) Resources .....	17
Agent Architecture .....	17
SL1 Distributed Architecture .....	17
SL1 Extended Architecture .....	18
<b>Installing the SL1 Agent</b> .....	<b>21</b>
Caveats .....	22
Installing an Agent from the Agents Page .....	22
Installing a Linux Agent .....	24
Using Agents with SELinux .....	26
Installing an AIX Agent .....	26
Installing a Solaris Agent .....	28
Installing a Windows Agent .....	29
Configuring the Windows Agent Using the Command Line .....	31
Configuring the Windows Agent to Run as Local System .....	32
Configuring the Windows Agent to Run as a Dedicated User .....	32
Example Install Commands .....	33
Example Upgrade Commands .....	34
Troubleshooting the Windows Installation Agent .....	34
Additional Considerations .....	35
Configuring an SL1 Agent to Use a Proxy Server Connection .....	36

Configuring a Linux Agent to Use a Proxy .....	36
Installing and Configuring with the Proxy Settings .....	37
Configuring a Windows Agent to Use a Proxy .....	37
Installing and Configuring an Agent with the Proxy Settings .....	39
Upgrading an Agent .....	39
Stopping an Agent .....	42
Uninstalling an Agent .....	42
Uninstalling a Linux Agent .....	43
Uninstalling an AIX Agent .....	43
Uninstalling a Solaris Agent .....	43
Uninstalling a Windows Agent .....	44
Installing an Agent from the Device Manager Page .....	44
Installing an Agent on a Linux System .....	45
Upgrading an Agent on a Linux System .....	46
Installing an Agent on a Windows System .....	46
Configuring an SL1 Agent to Work with Phone Home Collectors .....	46
Remotely Installing an SL1 Agent .....	49
Remotely Installing a Linux Agent .....	50
Remotely Installing a Windows Agent .....	54
<b>Configuring the SL1 Agent .....</b>	<b>61</b>
Configuring Agent Monitoring Based on SL1 Architecture .....	62
Configuring Agent Settings from the Device Investigator Page .....	63
Configuring an Agent to Export Data to Skylar .....	66
Enabling Skylar Export Using Deploy .....	66
Manually Enabling Skylar .....	67
Validating the Skylar Export .....	68
Configuring Extensible Collection .....	69
Configuring Run Book Automations for an SL1 Agent .....	71
Action Type .....	71
Configuring a Run Book Automation for an Agent Device .....	72
Configuring an SL1 Agent on the Device Manager Page .....	73
Adding the "SL Agent" Column to the Device Manager Page .....	74

Configuring Agent Settings on a Device .....	74
<b>Monitoring Logs Using the SL1 Agent .....</b>	<b>76</b>
What is a Log File Monitoring Policy? .....	77
Viewing the List of Log File Monitoring Policies .....	77
Creating a Log File Monitoring Policy .....	78
Aligning a Log File Monitoring Policy to Devices .....	79
Unaligning Log File Monitoring Policies from Devices .....	81
Editing a Log File Monitoring Policy .....	81
Deleting Log File Monitoring Policies .....	81
Viewing the List of Log File Monitoring Policies and Aligned Devices .....	82
Filtering the List of Log File Monitoring Policies and Aligned Devices .....	82
Creating an Event Policy for Agent Logs .....	83
Creating an Event Policy for Agent Logs in the Classic User Interface .....	87
<b>SL1 Agent Troubleshooting .....</b>	<b>90</b>
Was the Agent Download Successful? .....	91
Is the Windows installation or upgrade failing? .....	91
Is the Agent Process Running? .....	91
Is the Agent Configuration File Valid? .....	92
Has SL1 Discovery Completed? .....	93
Is the Agent Able to Upload Data? .....	93
Check the Agent Upload Directory .....	93
Run the Agent in Debug Mode (Linux) .....	94
Is SL1 Receiving Agent Data? .....	95
Is the Agent Not Reporting Vital Data and Metrics? .....	95
Can SL1 Process Agent Data? .....	95
Is the Number of Processes Inconsistent with Other Applications? .....	96
Does MySQL on the Data Collector Have a Connection Issue? .....	96
Is the Agent Unable to Connect to the Streamer Endpoint? .....	97
Troubleshooting Examples .....	99
Example /var/log/streamer_prime/streamer_prime.log for successful discovery .....	99
Example /var/log/uwsgi/streamer.log for successful discovery in the SL1 Distributed Architecture .....	101
Save incoming data for a specific device ID in the SL1 Distributed Architecture .....	101

Save incoming data for a specific device ID in the SL1 Extended Architecture .....	101
Additional Troubleshooting Situations and Best Practices .....	101
Agent Communication with SL1 .....	102
Return Codes .....	102
Sub-codes .....	102

---

# Chapter

# 1

## Introduction to the SL1 Agent



---

### Overview

This chapter describes the SL1 Agent and provides instructions for viewing device and interface data collected by the agent.

**NOTE:** If your current ScienceLogic SL1 solution subscription does not include the SL1 Agent, contact your ScienceLogic Customer Success Manager or Customer Support to learn more.

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:

<i>What is the SL1 Agent?</i> .....	7
<i>How the SL1 Agent Works in Different Environments</i> .....	7
<i>What System Vital Metrics Can the SL1 Agent Collect?</i> .....	10
<i>Agent PowerPacks</i> .....	14
<i>Agent Architecture</i> .....	17

---

## What is the SL1 Agent?

The **SL1 agent** is a program that you can install on a device monitored by SL1. There is a Windows agent, an AIX agent, a Solaris agent, and a Linux agent. The agent collects data from the device and pushes that data back to SL1.

Similar to a Data Collector or Message Collector, the agent collects data about infrastructure and applications.

You can configure an agent to communicate with either the Message Collector or the Compute Cluster.

**NOTE:** The following minimum agent versions are required for SL1 12.1.1 and later: **Windows** version 131; **Linux** version 174; **AIX** version 180; and **Solaris** version 180. Users who require agent-based log collection on a device with a Windows agent or a Linux agent must have the minimum Windows agent (131), or for a Linux agent (174). ScienceLogic recommends that users perform an upgrade, if they do not have the minimum required agent versions, via the Upgrade button on the Agent page in the current user interface, or by downloading and upgrading the agent manually.

---

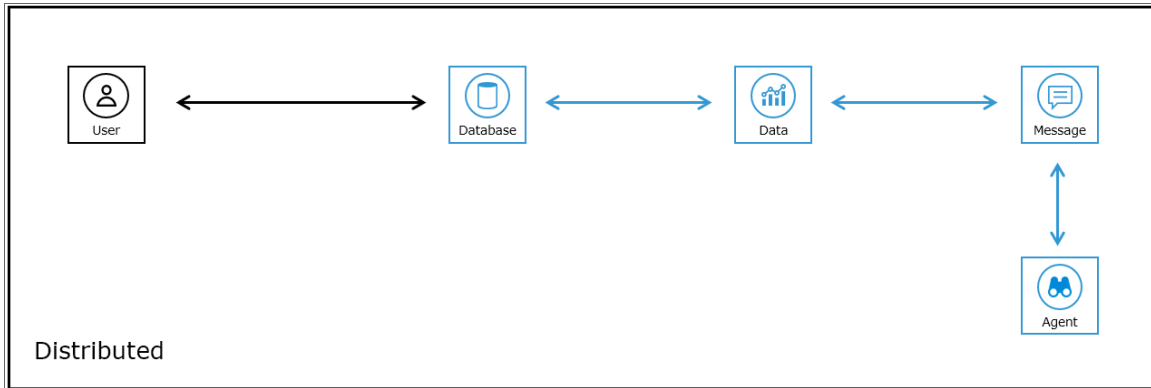
## How the SL1 Agent Works in Different Environments

There are two supported "generations" of the SL1 agent that you can use to gather data: Gen 1 and Gen 3. The agent itself is the same from one generation to the next; the only difference between the generations is the *environment* where the agent is used.

In brief, a **Gen 1 agent** uses the SL1 Distributed Environment to upload data directly to a Message Collector (MC), while a **Gen 3 agent** uses the SL1 Extended Architecture to upload data to the "Streamer" service running on the SL1 Compute Node cluster.

The following list provides more details about how SL1 uses the different generations of the SL1 agent:

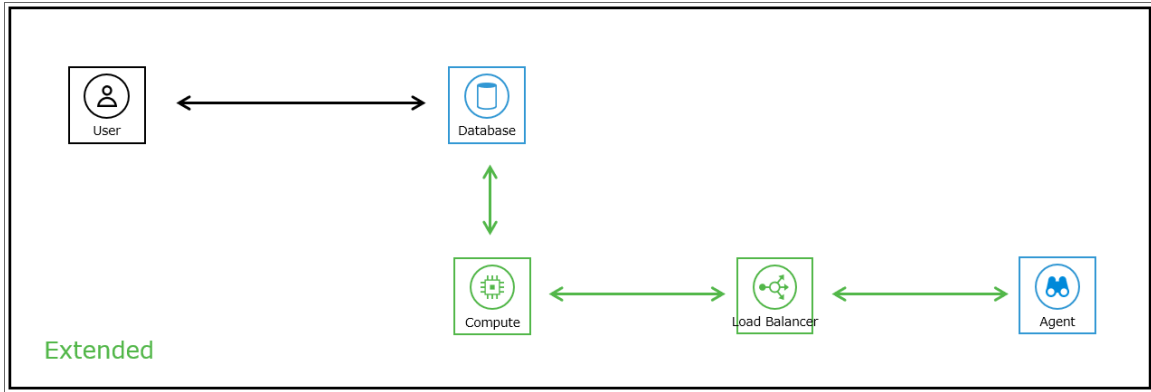
- **Gen 1 agent:** Introduced in SL1 version 8.2.0, the Gen 1 agent uses a Distributed Environment to upload data directly to a Message Collector (MC). Next, Dynamic Applications running on a Data Collector query data from the Message Collector with an API. The Data Collector (DC) then sends the data to the SL1 Database Server:



The Gen 1 agent provides limited infrastructure health reporting, including Log File Monitoring, Processes, and System Vitals like Availability, CPU Usage, Memory Usage, and Disk Usage. This agent is sometimes called the "PO" agent.



- **Gen 2 agent:** This version of the agent has been deprecated.
- **Gen 3 agent:** Introduced in SL1 version 8.12.0, the Gen 3 agent uses the SL1 Extended Architecture to upload data through a load balancer to the "Streamer" service running on the SL1 Compute Node cluster. The Streamer service then forks and forwards data to other services, and eventually some services will store parsed data in the SL1 Database Server:



The Gen 3 agent provides full infrastructure health reporting, including system vitals (file system, network interface, and Windows service data), log monitoring, and optionally allows you to align PowerShell Dynamic Applications to your device. Also, the Dynamic Applications in the *ScienceLogic: Agent* PowerPack are automatically aligned with the device with the Gen 3 agent installed on it.

If you are already running the Gen 1 agent in a distributed environment and you upgrade your SL1 system to use the SL1 Extended Architecture, any existing devices monitored by SL1 agents will work the same as before (streaming data to a Message Collector). However, if you upgrade to the SL1 Extended Architecture, ScienceLogic recommends that you start streaming your agents to the Compute Node cluster instead of to a Message Collector.

To identify the version of the agent installed on a device:

- **Gen 1.** On the **Devices > Device Investigator > [Settings]** tab for that device, the **Collection Poller** field displays the name of the collector group that includes the Message Collector used by the agent. On the **Device Manager** page (Devices > Device Manager), the **Collection Group** column displays the name of the collector group that includes the Message Collector used by the agent.
- **Gen 3.** On the **Devices > Device Investigator > [Settings]** tab for that device, the **Collection Poller** field displays the label *Agents*. On the **Device Manager** page (Devices > Device Manager), the **Collection Group** column displays the label *Agents*.

The following table provides an overview of the features available when using the Gen 1 or the Gen 3 agent:

Product Capability	Gen 1 Agent	Gen 3 Agent
<b>System Vitals</b>		
Availability and Uptime	Yes	Yes
CPU, Memory, File Systems, and Network Interface	CPU and Memory only	Yes
Processes and Windows Services	Processes only	Yes
Installed Software	No	Yes
<b>Log Monitoring</b>		
Event Logs	Yes	Yes
Syslog	Yes	Yes
Text Logs	Yes	Yes
<b>Extensible Collection</b>		
PowerShell	No	Yes
JMX	No	Yes

For a detailed list of the system vital metrics you can monitor with the SL1 agent, see [What System Vital Metrics Can the SL1 Agent Collect?](#)

For a current list of supported operating systems and host system requirements for the SL1 agent, see the [System Requirements for the Agent](#) page at the ScienceLogic Support Site.

**NOTE:** Users who are running version 102 or later of the *Microsoft: Windows Server PowerPack* can collect data via the SL1 agent. For more information, see the ***Monitoring Windows Systems with PowerShell*** manual.

## What System Vital Metrics Can the SL1 Agent Collect?

The following sections describe the system vitals that can be collected with SL1 and with the SL1 Agent, including definitions of each metric type and the collection methods that are and are not supported for each.

### Metric Descriptions

The following table describes the system vital metrics that can be collected with SL1 and the SL1 Agent:

Metric	Type	Description
Availability	Performance	The ability to communicate with the managed entity or device.
File Systems	Configuration	The configuration of the file systems found within a managed entity that can include attributes like name, size, and type.
File Systems	Performance	Time series data associated with file system utilization that can include metrics

Metric	Type	Description
		like free space, size, and usage percentage.
Installed Software	Configuration	The software found on a managed entity that can include attributes like name, version, and installation date.
Network Interfaces	Configuration	The configuration of the network interface found within a managed entity that includes attributes like MAC address, IP address, position, and speed.
Network Interfaces	Performance	Time series data associated with physical memory utilization that includes metrics like inbound and outbound utilization, number of errors, and discard and usage percentage.
Physical Memory	Configuration	The configuration of the physical memory found within a managed entity that can include attributes like memory size.
Physical Memory	Performance	Time series data associated with physical memory utilization that can include metrics like memory used.
Ports	Configuration	The ports discovered on a managed entity.
Ports	Performance	Time series data associated with port availability.
Ports (Illicit)	Performance	An analysis of ports. When a port from the illicit port list is found on a managed system, the system will trigger an event indicating an illicit port has been found.
Processes	Configuration	The processes found on a managed entity that can include attributes like name, process ID (PID), and state.
Processes	Performance	Time series data associated with process performance that can include metrics like availability percentage.
Processor	Configuration	The configuration of the processor found within a managed entity that can include attributes like number of cores, processor model, processor speed, cache size, and CPU ID.
Processor	Performance	Time series data associated with processor utilization that can include metrics like CPU idle time, CPU wait time, and overall CPU time.
Restarts	Performance	An analysis of uptime. When uptime is less than 15 minutes, the system triggers an event indicating the system was restarted.
SSL Certificates	Configuration	The certificates found on a managed system.
SSL Certificates	Performance	An analysis of certificate expiration date. The system will trigger an event when certificates are nearing expiration.
Uptime	Performance	The timespan since the managed entity was last initialized.
Virtual Memory (Swap)	Configuration	The configuration of the virtual memory found within a managed entity.
Virtual Memory (Swap)	Performance	Time series data associated with virtual memory utilization.
Windows Services	Configuration	The services found on a managed entity that can include attributes like name and state.
Windows Services	Performance	Time series data associated with service performance that can include metrics like availability percentage.

## Supported Data Collection Methods for Monitoring Windows

The following table describes which methods of data collection are supported when running SL1 and the SL1 Agent on monitored Windows systems:

Metric	Type	Agentless			Agent-Based	
		SNMP	WMI	PowerShell	Gen-01	Gen-03
Availability	Performance	Yes	Yes	Yes	Yes	Yes
File Systems	Configuration	Yes	Some	Yes	Some	Yes
File Systems	Performance	Yes	Some	Yes	Some	Yes
Installed Software	Configuration	Yes	No	Yes	No	Yes
Network Interfaces	Configuration	Yes	Some	Yes	Some	Yes
Network Interfaces	Performance	Yes	Some	Yes	Some	Yes
Physical Memory	Configuration	Yes	Yes	Yes	Yes	Yes
Physical Memory	Performance	Yes	Yes	Yes	Yes	Yes
Ports	Configuration	Yes	No	Yes	Yes	No
Ports	Performance	Yes	No	Yes	Yes	No
Ports (Illicit)	Performance	Yes	No	Yes	Yes	No
Processes	Configuration	Yes	Some	Yes	Yes	Yes
Processes	Performance	Yes	No	Yes	Yes	Yes
Processor	Configuration	Yes	Yes	Yes	Yes	Yes
Processor	Performance	Yes	Yes	Yes	Yes	Yes
Restarts	Performance	Yes	No	Yes	Yes	Yes
SSL Certificates	Configuration	Yes	No	No	No	No
SSL Certificates	Performance	Yes	No	No	No	No
Uptime	Performance	Yes	No	Yes	Yes	Yes
Virtual Memory (Swap)	Configuration	Yes	Yes	Yes	Yes	Yes
Virtual Memory (Swap)	Performance	Yes	Yes	Yes	Yes	Yes
Windows Services	Configuration	Yes	Some	Yes	No	Yes
Windows Services	Performance	Yes	Some	Yes	No	Yes

## Supported Data Collection Methods for Monitoring Linux

The following table describes which methods of data collection are supported when running SL1 and the SL1 Agent on monitored Linux systems:

Metric	Type	Agentless		Agent-Based	
		SNMP	SSH	Gen-01	Gen-03
Availability	Performance	Yes	Yes	Yes	Yes

Metric	Type	Agentless		Agent-Based	
		SNMP	SSH	Gen-01	Gen-03
File Systems	Configuration	Yes	Yes	Some	Yes
File Systems	Performance	Yes	Yes	Some	Yes
Installed Software	Configuration	Yes	No	No	Yes
Network Interfaces	Configuration	Yes	Yes	Some	Yes
Network Interfaces	Performance	Yes	Yes	Some	Yes
Physical Memory	Configuration	Yes	Yes	Yes	Yes
Physical Memory	Performance	Yes	Yes	Yes	Yes
Ports	Configuration	Yes	Yes	Yes	No
Ports	Performance	Yes	Yes	Yes	No
Ports (Illicit)	Performance	Yes	Yes	Yes	No
Processes	Configuration	Yes	Yes	Yes	Yes
Processes	Performance	Yes	Yes	Yes	Yes
Processor	Configuration	Yes	Yes	Yes	Yes
Processor	Performance	Yes	Yes	Yes	Yes
Restarts	Performance	Yes	Yes	Yes	Yes
SSL Certificates	Configuration	Yes	No	No	No
SSL Certificates	Performance	Yes	No	No	No
Uptime	Performance	Yes	Yes	Yes	Yes
Virtual Memory (Swap)	Configuration	Yes	Yes	Yes	Yes
Virtual Memory (Swap)	Performance	Yes	Yes	Yes	Yes
Windows Services	Configuration	N/A	N/A	N/A	N/A
Windows Services	Performance	N/A	N/A	N/A	N/A

## Extensible Collection

In addition to the capabilities listed above, you can use the SL1 agent for "extensible collection", where you align the agent with Dynamic Applications to gather metrics and attributes from other infrastructures and applications.

The SL1 Extended Architecture supports aligning PowerShell Dynamic Applications to devices monitored by the SL1 Windows agent. The SL1 Extended Architecture supports aligning JMX Dynamic Applications to devices monitored by the SL1 Linux agent.

In addition, Dynamic Applications that leverage the Low Code No Code CLI/SSH framework can execute using the agent.

For more information, see [Configuring Extensible Collection](#).

---

## Agent PowerPacks

SL1 includes two PowerPacks that can be used to collect agent-based system configuration and performance data: the **ScienceLogic: Agent PowerPack** and the **Host Agent PowerPack**.

Both PowerPacks are installed by default on your SL1 system, and they include the following features:

- Dynamic Applications that collect configuration data and performance metrics from devices that are using agent-based collection
- Event Policies and alerts that are triggered when devices that are using agent-based collection meet certain status criteria

The **ScienceLogic: Agent PowerPack** collects agent-based data for devices on SL1 systems running on the SL1 Extended Architecture (Gen 3 agents). This PowerPack contains two Dynamic Applications:

- The "ScienceLogic Agent: System Configuration" Dynamic Applications collects the following data:
  - CPU
  - CPU Information
  - CPUs
  - Hardware Totals
  - Memory
  - Speed (MHz)
  - Swap Capacity
- The "ScienceLogic Agent: System Performance" Dynamic Applications collects the following data:
  - CPU Name
  - CPU Utilization
  - CPU Utilization Breakdown
  - Disk Average Queue Length
  - Disk IO Utilization
  - Disk Name
  - Memory Utilization
  - Network Read
  - Network Write

- Sample Time
- Swap Utilization

The **Host Agent PowerPack**, which collects agent-based data for devices on SL1 systems running on a Distributed Architecture (Gen 1 agents). This PowerPack contains two Dynamic Applications:

- The "Host Agent: System Config" Dynamic Applications collects the following data:
  - CPU
  - CPU Information
  - CPUs
  - Disk
  - Disk Information
  - Disk Space
  - Disks
  - Hardware Totals
  - Memory
  - Size
  - Speed (MHz)
- The "Host Agent: System Perf" Dynamic Applications collects the following data:
  - CPU Name
  - CPU Utilization
  - CPU Utilization Breakdown
  - Disk Average Queue Length
  - Disk Name
  - Disk Utilization
  - Memory Utilization
  - Network Read
  - Network Write
  - Sample Time
  - Storage Available
  - Storage Name

- Storage Total
- Storage Utilization

**NOTE:** Because the *ScienceLogic: Agent* PowerPack is required to collect data from devices that are using agent-based collection, SL1 does not enable you to delete or modify this PowerPack.

## Agent-Compatible PowerPacks

In addition to the *ScienceLogic: Agent* PowerPack and the *Host Agent* PowerPack, there are several other Agent-compatible PowerPacks that you can use to collect data from specific device types.

### Windows Devices

The following PowerPacks include the SL1 Agent PowerShell Default credential and SL1 Agent device template, which you can use to execute the SL1 Agent on Windows devices with PowerShell:

- Microsoft: Windows Server
- SL1 Agent Templates for Microsoft PowerPacks, which includes templates for the following:
  - Microsoft: DHCP Server
  - Microsoft: DNS Server
  - Microsoft: Exchange Server

**NOTE:** The *Microsoft: Exchange Server* PowerPack has two device templates. If the Exchange server monitored contains all Exchange roles, use the "SL1 Agent for Microsoft: Exchange Server Template." If your Exchange server has an Exchange Transport role, use the "SL1 Agent for Microsoft: Exchange Transport Server Template."

- Microsoft: IIS Server
- Microsoft: Lync Server
- Microsoft: SharePoint Server
- Microsoft: SQL Server
- Microsoft: Windows Server

**NOTE:** For more information, see the *Monitoring Windows with PowerShell* manual.



## Java Management Extensions (JMX) Resources

You can also use the *JMX Base Pack* PowerPack to monitor JMX resources with the SL1 agent.

For more information, see the *Monitoring Java Management Extensions (JMX)* manual.

---

## Agent Architecture

The following sections describe how the SL1 agent works in the SL1 Distributed Architecture and in the SL1 Extended Architecture.

### SL1 Distributed Architecture

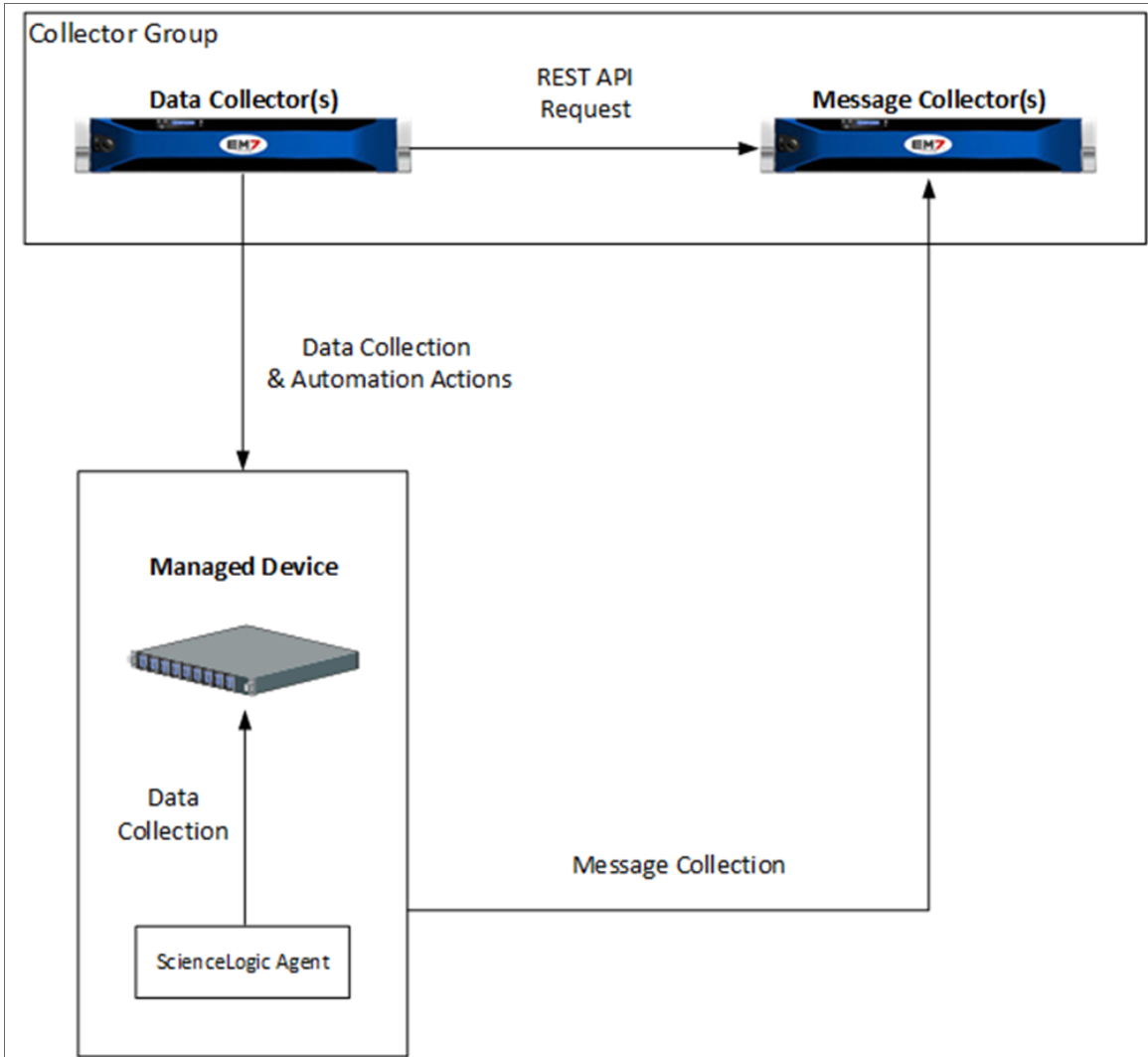
In an SL1 Distributed Architecture, the SL1 Agent collects data from the device on which it is installed and transfers that data to a Message Collector in an SL1 system using the HTTPS protocol. The Data Collector on which the Dynamic Applications and collection processes run then poll the Message Collector using the HTTPS protocol to transfer data to SL1.

TCP port 443 must be open between the Message Collector and the device on which an agent is installed.

In a Distributed Architecture, the SL1 agent requires a standalone, dedicated Message Collector. The Message Collector does not need to be dedicated to agent usage, but the Message Collector cannot be a Data Collector that also performs message collection.

**NOTE:** Message Collectors that process data from the agent have different system requirements than Message Collectors that do not process data from the agent. For more information about the system requirements when running agents in a Distributed Architecture, see the [System Requirements](#) page at the ScienceLogic Support Site.

The diagram below shows the collection layer of a Distributed System containing both Data Collectors and Message Collectors in which the SL1 Agent is installed on a managed device.



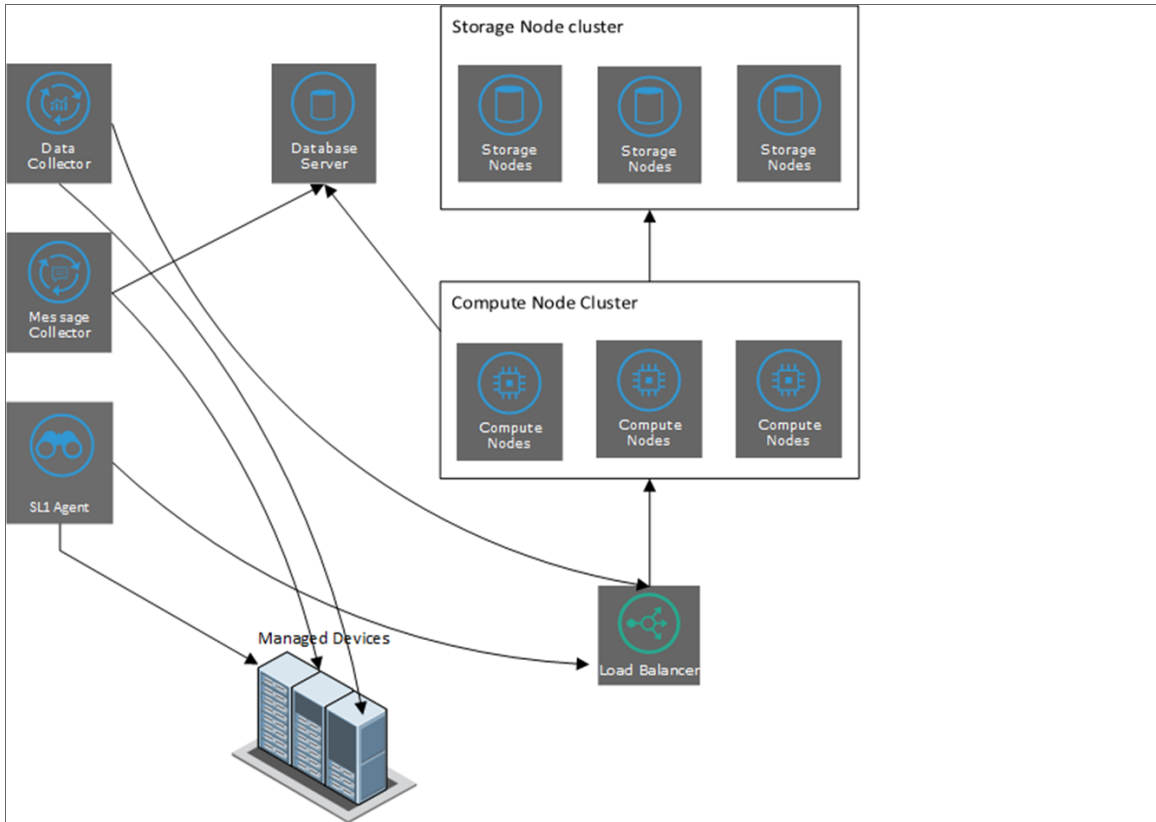
## SL1 Extended Architecture

In the SL1 Extended Architecture, an **SL1 agent** collects data from the device on which it is installed and sends that data to a Load Balancer in front of a Compute Cluster. The Compute Cluster transforms the data and stores high-volume performance data in the Storage Cluster and other performance and configuration data in the Database Server.

If required, agents can use an HTTP proxy server as an intermediate step in sending data to SL1.

In the diagram below:

- The **SL1 agent** collects data from managed devices and sends the data to the Load Balancer and Compute Node cluster for processing.
- The optional **Message Collector** collects asynchronous traps and syslog messages and sends them to the Database Server.
- The **Data Collector** collects data from managed devices and sends the data to the Load Balancer and Compute Node cluster for processing and then storage.



Using an agent in the SL1 Extended Architecture provides more configuration and performance data than using an agent in a Distributed Architecture. This additional data includes system vitals, log data, and extensible collection.

**NOTE:** Uploads that occur in 20-second intervals, sometimes called "snapshot uploads", are no longer supported for users using the non-Scylla pipeline. These 20-second uploads were replaced with the 1-minute upload default in SL1 version 11.2. ScienceLogic highly recommends that you ensure your agents are uploading in one-minute summarized uploads prior to upgrading. You can verify your uploads from the **[Settings]** tab in the current SL1 user interface. The agent pipeline is able to consume and summarize 1-minute and 5-minute payloads without the need for Scylla.

**NOTE:** For more information about the system requirements when running agents in an Extended Architecture, see the [System Requirements](#) page at the ScienceLogic Support Site.

---

# Chapter

# 2



## Installing the SL1 Agent

---

### Overview

This chapter describes how to install, upgrade, and uninstall SL1 Agents for Windows, Linux, AIX, and Solaris operating systems. All agent installations, upgrades, and uninstallations require command-line interface access to the targeted device.

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:

<i>Caveats</i> .....	22
<i>Installing an Agent from the Agents Page</i> .....	22
<i>Configuring an SL1 Agent to Use a Proxy Server Connection</i> .....	36
<i>Upgrading an Agent</i> .....	39
<i>Stopping an Agent</i> .....	42
<i>Uninstalling an Agent</i> .....	42
<i>Installing an Agent from the Device Manager Page</i> .....	44
<i>Remotely Installing an SL1 Agent</i> .....	49

---

## Caveats

- If you use the **SL1 Extended Architecture** (which includes Compute Nodes, Storage Nodes, and a Management node), install the agent from the **Agents** page (Devices > Agent). For details, see [Installing an Agent from the Agents Page](#).

**NOTE:** With on-premises SL1 Extended Architecture systems, the TLS handshake can fail between the Windows Agent on a monitored device and the SL1 streamer service. For details and the workaround, see the following Knowledge Base article: [Extended Architecture](#).

- If you use the **SL1 Distributed Architecture** (where the SL1 Agent collects and transfers data to a Message Collector), install the agent from the **Device Manager** page (Devices > Device Manager). For details, see [Installing an Agent from the Device Manager Page](#).

**NOTE:** If you are using PhoneHome Message Collectors in a Distributed Architecture, see [Configuring an SL1 Agent to Work with PhoneHome Collectors](#) before you install the agent.

- RPM installer packages must be signed. Therefore, when installing an RPM package, you might receive a warning message if the RPM store does not contain ScienceLogic's public GPG key.

To address or prevent this warning, you can obtain the ScienceLogic key and then add it to the RPM store:

1. Go to <http://keys.openpgp.org/search?q=devops%40sciencelogic.com>.
2. Download the key.
3. Import the key into the RPM store using the following command:

```
rpm --import <file name>
```

---

## Installing an Agent from the Agents Page

When running SL1 on the SL1 Extended Architecture, you can use the **Agents** page (Devices > Agents) to help you install the SL1 agent on a Linux, Windows, AIX, or Solaris device.

Device	Hostname	OS	Version	Newest Version	Date Added	Last Uploaded
<input type="checkbox"/> dclaire-win-133	dclaire-win-133	Windows	140	✓	Mar 18, 2022, 3:54 PM	Mar 18, 2022, 3:55 PM
<input checked="" type="checkbox"/> auto-cdb-10-2-17-148	auto-cdb-10-2-17-148	Linux	180	✓	Mar 18, 2022, 3:55 PM	Mar 18, 2022, 3:55 PM
<input type="checkbox"/> auto-sn3-10-2-17-143	auto-sn3-10-2-17-143	Linux	180	✓	Mar 22, 2022, 11:28 AM	Mar 22, 2022, 11:29 AM
<input type="checkbox"/> auto-cn1-10-2-17-144	auto-cn1-10-2-17-144	Linux	180	✓	Mar 22, 2022, 11:31 AM	Mar 22, 2022, 11:31 AM
<input type="checkbox"/> auto-cn2-10-2-17-145	auto-cn2-10-2-17-145	Linux	180	✓	Mar 22, 2022, 11:35 AM	Mar 22, 2022, 11:35 AM
<input type="checkbox"/> auto-cn3-10-2-17-146	auto-cn3-10-2-17-146	Linux	180	✓	Mar 22, 2022, 11:37 AM	Mar 22, 2022, 11:37 AM
<input type="checkbox"/> auto-lb-10-2-17-147	auto-lb-10-2-17-147	Linux	180	✓	Mar 22, 2022, 11:38 AM	Mar 22, 2022, 11:38 AM

You can also upgrade and delete agents from SL1 on the **Agents** page.

**TIP:** You can filter the items on this inventory page by typing filter text or selecting filter options in one or more of the filters found above the columns on the page. For more information, see "Filtering Inventory Pages" in the *Introduction to SL1* manual.

**TIP:** The **Agents** page appears as an option on the **Devices** menu *only* when you are using the SL1 Extended Architecture. If your version of SL1 does not have an **Agents** page, see [Installing an Agent from the Device Manager Page](#).

**NOTE:** If you are using an SL1 system with the SL1 Extended Architecture, you *must* use the **Agents** page (Devices > Agents) to install agents, including AIX and Solaris agents. AIX and Solaris agents are not available for non-SL1 Extended systems.

**NOTE:** If you want to remotely install one or more agents, see [Remotely Installing an SL1 Agent](#).

You do not install the agent from the **Agents** page itself. Instead, the **Agents** page enables you to gather the information or files you need to then install the agent on a particular device.

The instructions on the relevant tab of the **Agents** page also include command line parameters such as **URLFRONT**, which is used to define the Streamer service you are using as part of the SL1 Extended Architecture.

If the agent installation is successful, one of the following automatically happens:

- If the primary IP address of the device is not currently monitored by SL1, then SL1 creates a device record for the device and populates the device record with data provided by the agent. The device record is assigned a device class based on data reported by the agent.
- If the primary IP address of the device is currently monitored by SL1, the device record for the existing device is updated with data provided by the agent.

During initial discovery, the agent returns operating system type and version information to SL1. Based on this information, SL1 assigns the corresponding device classes to a device monitored only by an agent. If a device is monitored by an agent and via SNMP, the device class assigned by SNMP discovery will take precedence.

## Installing a Linux Agent

For a Linux system, the **Agents** page provides version-specific commands that you will need to run on the Linux system you want to monitor. This page includes commands for the following operating systems:

- Ubuntu and Debian (64-bit)
- Red Hat and CentOS (64-bit)
- Red Hat and CentOS - OS Libs (64-bit)

**NOTE:** Linux version 174 is the minimum version required for SL1 12.1.1.

The Linux agent installation program checks for the relevant version of libcurl before installing the agent. If libcurl is not installed, the agent installation program installs the most recent version of libcurl.

SL1 supports the use of proxy server connections when using the agent on Linux systems. For more information about installing and configuring a Linux agent with a proxy server, see [Configuring the Linux Agent to Use a Proxy](#).

Beginning with Linux Agent version 184, by default the agent will install and run as a dedicated non-root user with the user name "scilog". To run the Linux agent as root, use the installation command which includes `RUN_AS_ROOT=1`. This configuration has the following limitations:

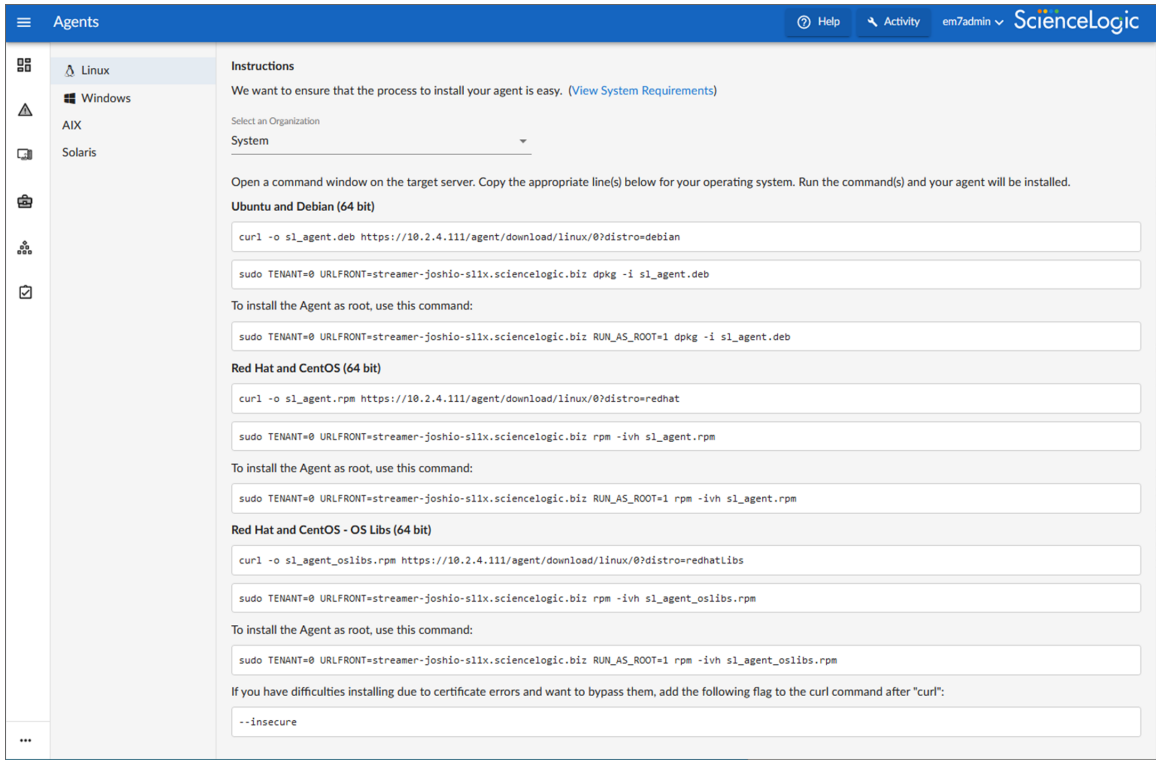
- Process data is only Non-Intercepted Process Data (NIPD). There are no intercepts or library data.
- Polled Data only works for the current user.
- Log files can only be monitored if they are readable by the `scilog user` or `scilog group`.
- You cannot upgrade the agent from the SL1 user interface because the installation and uninstallation processes require root privileges.

If you are installing on SELinux, see [Using Agents with SELinux](#).



To install a Linux agent:

1. On the **Agents** page (Devices > Agents), click **[New Agent]**. The **Agent Installation** page appears:



2. Click the **[Linux]** tab at top left.
3. From the **Select an Organization** drop-down list, select an organization for the device on which you are installing the agent. The **TENANT** value in the commands on this page is updated with the SL1 Organization ID for the organization you selected.

**TIP:** The **URLFRONT** value is the URL for the Streamer service you are using as part of the SL1 Extended Architecture.

4. Based on the version of Linux running on the device you want to monitor, run the relevant commands from the **[Linux]** tab on the Linux device (the "target server"). After the commands complete, the Linux agent starts running in the background. The device on which you installed the agent appears on the **Agents** page (Devices > Agents), and the device is also added to the list of devices on the **Devices** page.

You can also install the Linux agent via the command line.

To install the Linux agent (Debian or Ubuntu) using the command line, execute the following command:

```
sudo TENANT=<organization id> URLFRONT=<streamer_URL> dpkg -i sl_agent.rpm
```

- `TENANT` and `URLFRONT` are required.
- The `RUN_AS` state can be set via command line with `RUN_AS_ROOT=<0 | 1>`.
- The default `RUN_AS` state is "scilog" when there is no command line `RUN_AS_ROOT=X` parameter given.

To install the Linux agent (RHEL or CentOS) using the command line, execute the following command:

```
sudo TENANT=<organization_id> URLFRONT=<streamer_URL> rpm -ivh sl_
agent.rpm
```

- `TENANT` and `URLFRONT` are required.
- The `RUN_AS` state can be set via command line with `RUN_AS_ROOT=<0 | 1>`.
- The default `RUN_AS` state is "scilog" when there is no command line `RUN_AS_ROOT=X` parameter given.

## Using Agents with SELinux

The Linux agent supports running in SELinux for CentOS or Red Hat in both permissive and enforcing modes. The RPM installer for Linux will detect if SELinux is active (permissive/enforcing) upon installation. The agent automatically creates and installs an appropriate SELinux policy for use with the agent daemon.

With the Linux Agent, a shared library gets injected into other processes. For non-root processes, this shared library should not generate any errors or audit logs. However, for root processes, this shared library might generate some additional audit logs where some of the shared library functionality is restricted by SELinux. If this is the case, you can update the agent configuration to exclude pre-loading into these root processes.

## Installing an AIX Agent

Beginning with AIX agent version 184, by default the agent will install and run as a dedicated non-root system user named "scilog". To run the AIX agent as root, use the installation command which includes `RUN_AS_ROOT=1`.

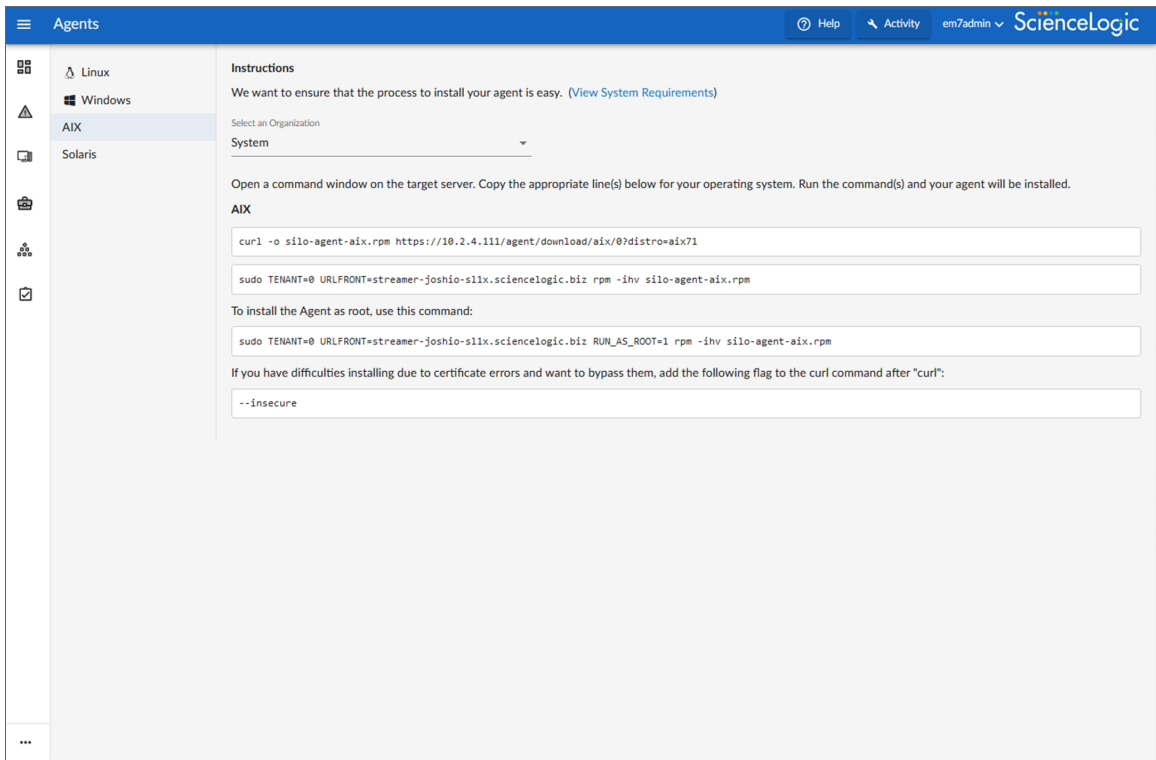
For an AIX system, the **Agents** page provides a set of commands that you will need to run on the AIX system you want to monitor.

**NOTE:** The AIX agent is a Beta feature.

**NOTE:** AIX agent version 180 is the minimum version required for SL1 12.1.1.

To download and install an AIX agent:

1. On the **Agents** page (Devices > Agents), click **[New Agent]**. The **Agent Installation** page appears.
2. Click the **[AIX]** tab:



3. From the **Select an Organization** drop-down list, select an organization for the device on which you are installing the agent. The **Tenant** value in the commands on this page is updated with the SL1 Organization ID for the organization you selected.

**TIP:** The **URLFront** value is the URL for the Streamer service you are using as part of the SL1 Extended Architecture.

4. Run the relevant commands from the **[AIX]** tab on the AIX device (the "target server"). After the commands complete, the AIX agent starts running in the background. The device on which you installed the agent appears on the **Agents** page (Devices > Agents), and the device is also added to the list of devices on the **Devices** page.

You can also install the AIX agent via the command line by executing the following command:

```
sudo TENANT=<organization_id> URLFRONT=<streamer_URL> rpm -ihv sl_agent_aix.rpm
```

- `TENANT` and `URLFRONT` are required.
- The `RUN_AS` state can be set via command line with `RUN_AS_ROOT=<0|1>`.
- The default `RUN_AS` state is "scilog" when there is no command line `RUN_AS_ROOT=X` parameter given.

# Installing a Solaris Agent

Beginning with Solaris agent version 184, by default the agent will install and run as a dedicated non-root system user named "scilog". To run the Solaris agent as root, add the file /etc/scilog/run\_state.txt containing `RUN_AS_ROOT=1`.

For a Solaris system, the **Agents** page provides version-specific commands that will need to run on the Solaris system you want to monitor. This page includes commands for the following versions:

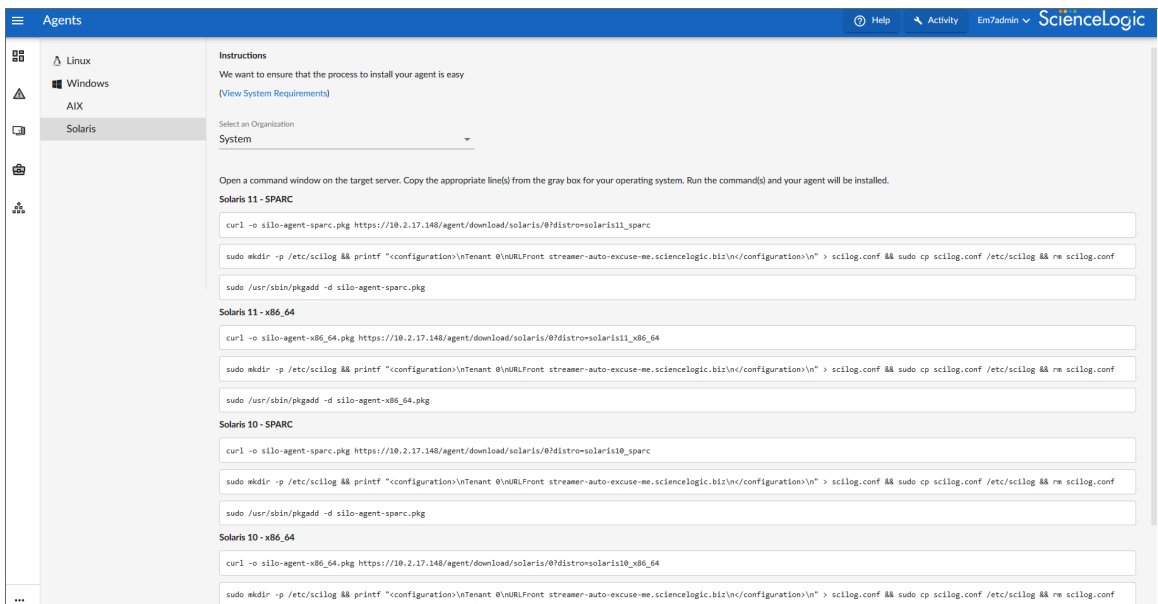
- Solaris 11 - SPARC
- Solaris 11 - x86\_64
- Solaris 10 - SPARC
- Solaris 10 - x86\_64

**NOTE:** The Solaris agent is a Beta feature.

**NOTE:** Solaris agent version 180 is the minimum version required for SL1 12.1.1.

To download and install a Solaris agent:

1. On the **Agents** page (Devices > Agents), click the **[New Agent]** button. The **Agent Installation** page appears.
2. Click the **[Solaris]** tab:



3. From the **Select an Organization** drop-down list, select an organization for the device on which you are installing the agent. The **Tenant** value in the commands on this page is updated with the SL1 Organization ID for the organization you selected.

**TIP:** The **URLFront** value is the URL for the Streamer service you are using as part of the SL1 Extended Architecture.

4. Based on the version of Solaris running on the device you want to monitor, run the relevant commands from the **[Solaris ]** tab on the Solaris device (the "target server"). After the commands complete, the Solaris agent starts running in the background. The device on which you installed the agent appears on the **Agents** page (Devices > Agents), and the device is also added to the list of devices on the **Devices** page.

You can also install the Solaris agent via the command line by executing the following commands:

```
mkdir -p /etc/scilog && printf "<configuration>\nTenant <organization_id>\n\nURLFront <streamer_URL>\n\n<configuration>\n" > scilog.conf && sudo cp scilog.conf /etc/scilog && rm scilog.conf
```

```
sudo pkgadd -d silo-agent.pkg
```

- The file /etc/scilog/scilog.conf with fields `TENANT` and `URLFRONT` are required. Note that "<configuration>" is a valid component of the file content and not meant to be substituted with any company or organizational value.
- The `RUN_AS` state can be set by adding file /etc/scilog/run\_state.txt with `RUN_AS_ROOT=<0|1>`.
- The default `RUN_AS` state is "scilog" when a /etc/scilog/run\_state.txt file doesn't exist.

## Installing a Windows Agent

For a Windows system, the **Agents** page provides an executable file (.exe) that you will need to run on the Windows system you want to monitor.

ScienceLogic provides an executable (.exe) installer called SiloAgent-install.exe. You will need to install and run the SiloAgent-install.exe file on the Windows system you want to monitor.

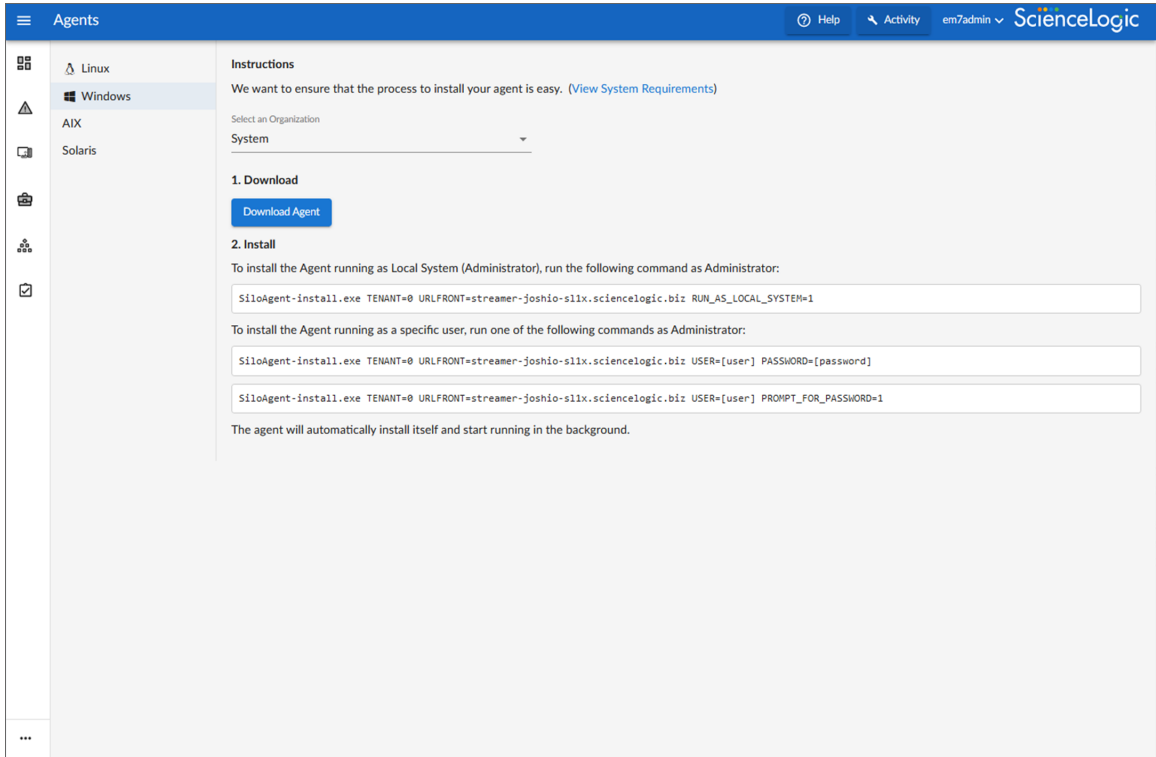
ScienceLogic also provides an MSI (.msi) installer upon request. The .exe installer is essentially a wrapper around the .msi file that allows for additional checks before installation, unwraps and runs the .msi file, prints information to the terminal, and provides information about failed installations. ScienceLogic recommends you use the .exe installer except for special cases. You can ask your ScienceLogic contact for an .msi file, if needed.

**NOTE:** SL1 supports the use of proxy server connections when using the SL1 agent on Windows systems. For more information about installing and configuring a Windows agent with a proxy server, see [Configuring the Windows Agent to Use a Proxy](#).

**NOTE:** Windows version 131 is the minimum required version for SL1 12.1.1.

To download and install a Windows agent in SL1:

1. On the **Agents** page (Devices > Agents), click **[New Agent]**. The **Agent Installation** page appears.
2. Click the **Windows** tab at top left:



3. From the **Select an Organization** drop-down list, select an organization for the new agent. The **Tenant** value in the command on this page is updated with the SL1 Organization ID for the organization you selected.
4. Click **[Download Agent]**. An **Opening SiloAgent-install.exe** dialog appears.
5. Save the **SiloAgent-install** .exe file for installing the agent.
6. Copy the **SiloAgent-install** file you downloaded to the Windows system on which you want to run the agent. To do so, you can either go to the console of the Windows system or use a utility like WinSCP.

7. To install the agent, use Windows PowerShell as an Administrator to run the command on the Windows system (the command is listed in step 2 on the **Windows** tab on the **Agents** page):

```
.\SiloAgent-install.exe tenant=<organization_id> urlfront=<streamer_
URL> RUN_AS_LOCAL_SYSTEM=1
```

or

```
.\SiloAgent-install.exe tenant=<organization_id> urlfront=<streamer_
URL> USER=<user> PASSWORD=<password>
```

where `<organization_id>` is Organization ID for the organization you selected in step 3, and `<streamer_URL>` is the URL for the Streamer service you are using as part of the SL1 Extended Architecture.

For example:

```
.\SiloAgent-install.exe tenant=1
urlfront=streamer2020.int.sciencelogic.com RUN_AS_LOCAL_SYSTEM=1
```

**NOTE:** `RUN_AS_LOCAL_SYSTEM=1` indicates the agent service will run with administrator privileges. `USER` and `PASSWORD` designate a dedicated user to run the service. See [Configuring the Windows Agent](#) for more information.

8. When the agent installation completes, the PowerShell window displays "info: install complete", and the Windows agent starts running in the background. The device on which you installed the agent appears on the **Agents** page (Devices > Agents), and the device is also added to the list of devices on the **Devices** page.
9. To verify that the installation was successful, open the Windows Task Manager or enter the `TASKLIST` command to view running processes. The `SiloAgent` process will be running on the Windows machine.

**NOTE:** When you upgrade to a newer version of the Windows agent, the installer upgrades the agent without generating a new CollectorID. The previous version's configuration file is passed to the upgraded version of the agent. The upgraded Windows agent continues to upload data to the same device record in SL1.

## Configuring the Windows Agent Using the Command Line

The Windows agent can be configured with two command-line modes, `RUN_AS_LOCAL_SYSTEM` and `RUN_AS_USER`. You should choose which command line configuration your device needs. Prior to the Windows agent version 145, the agent was configured with user permissions equal to an administrator user when the agent ran under the Local System user account. For version 145 and later, you can specify any user account as the Windows agent runner. For security purposes, when a non-administrator user account is designated, the agent will have the same permissions and rights as that user account.

## Configuring the Windows Agent to Run as Local System

Configuring the Windows agent service to run as a Local System with administrator privileges requires that you include `RUN_AS_LOCAL_SYSTEM=1` in your installation command. This configuration is simple, allows the agent to collect many types of data from your device, and guarantees that Dynamic Applications (e.g. PowerShell collections through the agent) execute without issue.

## Configuring the Windows Agent to Run as a Dedicated User

Configuring Windows agent service to run as a dedicated user account requires that you specify `RUN_AS_USER=1` on the install or upgrade command to indicate that the agent should run as a specific user account. Using this mode allows you to have precise control over the agent permissions and an improved ability to audit agent actions.

Before installing the Windows agent:

1. Create a Windows user account (local user on the device or domain user) that will be used to run the SiloAgent Service.
2. Grant the user the exact permissions required to run the agent. Some permissions may be granted as part of the installation command but you may also configure the user before installation.

Using `RUN_AS_USER=1` requires additional command-line options that must be configured properly for the agent to collect data and execute actions.

1. Include `USER=<user>` in your installation command (required):
  - `USER`. Defines the user that will run the agent service when using `RUN_AS_USER=1`. `USER` may be a username, which indicates a local user account, or it may be in the `domain\username` form. `USER` is required for the agent to successfully install with `RUN_AS_USER=1`. It is not necessary to specify `USER` during an upgrade except to modify the property.
2. Include `PASSWORD=<password>` or `PROMPT_FOR_PASSWORD=1`. One or the other is required:
  - `PASSWORD`. Defines the password associated with `USER`. The password is required to set the `USER` as the agent's service runner and to add domain accounts to groups. `PASSWORD`, or `PROMPT_FOR_PASSWORD` is required if `RUN_AS_USER=1` is designated. It is not necessary to specify `PASSWORD` during an upgrade except to modify the property.
  - `PROMPT_FOR_PASSWORD`. This property is only available if using the .exe installer. Use `PROMPT_FOR_PASSWORD=1` on the command line instead of `PASSWORD` so that the .exe installer will prompt for a password before starting the installation. The value entered takes the place of `PASSWORD`. This option can be used to avoid entering plaintext passwords into the command line.

There are additional command line options in your installation command that grant permissions to the account specified by `USER`. You may want to grant permissions to the `USER` before the installation for the most control. There are a few permissions that are essential to the agent's central functions and you have the option to grant these during installation.

- `LOG_ON_AS_A_SERVICE`. User accounts require the "Log on as a service" user right to run services. `LOG_ON_AS_A_SERVICE=1` grants this permission to the user. The agent installation may succeed, but the agent service will fail to start if the `USER` does not have this right.



**IMPORTANT:** The "Log on as a service" user right is **required** to run the the SiloAgent service as a dedicated user. You must either use the LOG\_ON\_AS\_A\_SERVICE=1 option, or manually grant the user this right.

- **DEBUG\_PROGRAMS.** To view the process details of processes owned by other users, a user account must have the required "Debug programs" user right. `DEBUG_PROGRAMS=1` grants this right to `USER`. Without this permission, only processes owned by `USER` can be monitored.
- **PERFORMANCE\_MONITOR\_USERS.** `PERFORMANCE_MONITOR_USERS=1` adds the `USER` to the Windows group "Performance Monitor Users" which enables the agent to read Windows Performance Counters such as CPU utilization, memory utilization, and disk utilization. This permission is required for the Microsoft Windows PowerShell dynamic applications to work correctly and allows the Windows Management Instrumentation (WMI) queries to execute properly.
- **PERFORMANCE\_LOG\_USERS.** `PERFORMANCE_LOG_USERS=1` adds the `USER` to the Windows group "Performance Log Users" which enables the agent to monitor application events through log monitoring and allows local WMI queries to execute properly.
- **EVENT\_LOG\_READERS.** `EVENT_LOG_READERS=1` adds the `USER` to the Windows group "Event Log Readers" and enables the agent to monitor application events through Log Monitoring.

**NOTE:** If neither mode is specified in a new install, the agent will default to `RUN_AS_USER=1`. If neither mode is specified during an upgrade, the agent will default to the mode already designated. Therefore, any agent running on version 144 and earlier will upgrade to `RUN_AS_LOCAL_SYSTEM` if not specified in the command line.

## Example Install Commands

When installing the agent for the first time on a device running as Local System:

```
SiloAgent-install.exe TENANT=0 URLFRONT=10.1.1.2 RUN_AS_LOCAL_SYSTEM=1
```

1. **Tenant** and **URLFront** values must be specified.
  - **TENANT.** Corresponds to Organization ID in SL1. `TENANT` is required for agent installation. The Windows agent will continue to use the same **Tenant** value during an upgrade.
  - **URLFRONT.** The URL or IP for the streamer service. `URLFRONT` is required for Agent Installation. The Windows agent will continue to use the same **URLFront** value during an upgrade.
2. `RUN_AS_LOCAL_SYSTEM=1` must be specified.

When installing the agent for the first time on a device running as a dedicated user named "scilogrunner" (local account on the device) with a password "qwerty".

```
SiloAgent-install.exe TENANT=0 URLFRONT=10.1.1.2 RUN_AS_USER=1  
USER=scilogrunner PASSWORD=qwerty
```

When installing the agent for the first time on a device running as a dedicated user named "MYDOMAIN\scilogrunner" (domain account) on "MYDOMAIN" with password "qwerty".

```
SiloAgent-install.exe TENANT=0 URLFRONT=10.1.1.2 RUN_AS_USER=1  
USER=MYDOMAIN\scilogrunner PASSWORD=qwerty
```

When installing the agent for the first time on a device running as a dedicated local user named "scilogrunner" with the password "qwerty", and granting the user the "Log on as a service" user right, the "Debug programs" user right and adding the user to the "Performance Monitor Users" group.

```
SiloAgent-install.exe TENANT=0 URLFRONT=10.1.1.2 RUN_AS_USER=1  
USER=scilogrunner PASSWORD=qwerty LOG_ON_AS_A_SERVICE=1 DEBUG_PROGRAMS=1  
PERFORMANCE_MONITOR_USERS=1
```

1. Grant the user the "Log on as a service" and "Debug programs" user rights.
2. Add to the Performance Monitor Users group.

## Example Upgrade Commands

When you upgrade the agent installation:

To upgrade an existing agent installation currently running as Local System, you are not required to specify any properties. The properties configured in the previous version are stored and the new version of the agent will continue to run as Local System, using the same **Tenant** and **URLFront**.

```
SiloAgent-install.exe
```

To upgrade an existing agent installation currently running as a dedicated user, you are not required to specify any properties. The upgraded agent stores the preferences from the previous version. The new version of the agent will continue to run as **USER** with the same **Tenant** and **URLFront**.

```
SiloAgent-install.exe
```

To change from a current agent installation running as Local System to a dedicated user during an upgrade, you are not required to provide **Tenant** and **URLFront**, but the **RUN\_AS\_USER=1** and **USER** and **PASSWORD** must be specified. Once specified, the new version of agent will run as **USER**.

```
SiloAgent-install.exe RUN_AS_USER=1 USER=scilogrunner PASSWORD=qwerty
```

To change from a current agent installation running dedicated user to Local System during an upgrade, you are not required to provide **Tenant** and **URLFront**, but **RUN\_AS\_LOCAL\_SYSTEM=1** must be specified.

```
SiloAgent-install.exe RUN_AS_LOCAL_SYSTEM=1
```

To change the **USER** and **PASSWORD** of an existing agent installation, use the same installer and specify the new **USER** and **PASSWORD**.

```
SiloAgent-install.exe USER=scilogrunner_new PASSWORD=different_password
```

## Troubleshooting the Windows Installation Agent

If you experience issues during the installation process, review the example scenarios below.

**TIP:** When an installation fails, a `SiloAgent_install_log.txt` file is generated in the current directory. If you need to contact ScienceLogic Support, please have that file when you contact them.

The agent installation fails when attempting to install with `RUN_AS_USER=1`

- Specify a user account that already exists on the computer or domain.
- Ensure that either `PASSWORD=[password]` or `PROMPT_FOR_PASSWORD=1` is specified.
- Ensure that the `USER` and `PASSWORD` provided are valid.
- Ensure that all property names are spelled correctly.

The agent installation succeeds, but the SiloAgent service does not start. If `USER` and `PASSWORD` are valid, but `USER` does not have the "Log on as a service" user right, the installation will succeed but will not start. In this situation, you have two choices:

- manually grant `USER` the "Log on as a service" user right and then start the service
- re-run the installation or upgrade command with `LOG_ON_AS_A_SERVICE=1`

**CAUTION:** Please ensure that your `USER` account or containing groups do not have the "Deny log on as a service" user right. This right supersedes the "Log on as a service" right and will prevent the agent service from starting.

## Additional Considerations

Beyond installation errors, there may be other service-related concerns you could experience.

- The **[Upgrade]** button present on the SL1 agent page sends a simple upgrade command to the Windows agent. The upgraded agent will have the same run-as mode as the previous version. There is currently no way to change `USER` which runs the agent service from the current SL1 user interface. The upgrade must be done from the command line.
- For upgrades from the SL1 user interface to be successful, the `USER` must be in the administrator group because administrator access is required for the agent to upgrade itself (i.e. initiating installation of new software). Otherwise, the upgrade must be conducted by an administrator via the command line.
- Other `USER` permissions should be limited (e.g. "Deny log on locally" or "Deny Access to this computer from the network").
- Exercise caution when adding the `USER` to the administrator group as you are passing administrator credentials during the installation.
- The `USER` must have read access to any directories where log file monitoring will occur or the log file monitoring will fail.
- The `USER` must have any permissions required to execute PowerShell commands or commands passed by Dynamic Applications or those commands will fail.

- Additional permissions or configuration may be required for other Dynamic Applications which execute scripts through the agent, such as PowerShell Performance and PowerShell Configuration Dynamic Applications. For example, the "Microsoft: SQL Server" PowerPack applications will not operate correctly unless the `USER` is granted permissions to the SQL server instance and databases.

---

## Configuring an SL1 Agent to Use a Proxy Server Connection

The following section covers how you can configure a proxy server connection when using an agent on Linux or Windows systems.

### Configuring a Linux Agent to Use a Proxy

To configure the Linux agent to use a proxy server connection:

1. Create a configuration file named `scilog_proxy.conf` and add the following proxy information to the new file:

```
Proxy-Server
```

```
Proxy-Port
```

```
Proxy-Username
```

```
Proxy-Password
```

For example:

```
Proxy-Server 10.1.1.1
```

```
Proxy-Port 8080
```

```
Proxy-Username admin1
```

```
Proxy-Password passw0rd
```

**NOTE:** The agent requires at least the proxy server and proxy port information. If no authentication is required, leave the username and password commented out with a "#". If values are present for the username and password, the agent will try to use them. The agent currently supports Basic Authentication.

2. Add the new file to the `/etc/scilog` directory on the SL1 server. This directory is also where the `scilog.conf` file resides.

**NOTE:** If the `scilog_proxy.conf` file exists, the agent will try to connect to a proxy. If you do not need a proxy, remove or rename this file to prevent the agent from accessing it. Alternatively, you can leave the parameters blank in the `scilog_proxy.conf` file.

## Installing and Configuring with the Proxy Settings

You can set the proxy parameters at install time by including them as parameters for the install command, just as you do with the **Tenant** and **URLFront** values. Use the package name for the package being installed.

For example:

### deb

```
sudo TENANT=0 URLFRONT=10.1.1.2 proxyserver=10.2.1.1 proxyport=3128  
proxyusername=1 proxypassword=1 dpkg -i sl_agent_v173.15.deb
```

### rpm

```
sudo TENANT=0 URLFRONT=10.1.1.2 proxyserver=10.2.1.1 proxyport=3128  
proxyusername=1 proxypassword=1 rpm -ivh sl_agent_v173.16.rpm
```

### aix

```
sudo TENANT=0 URLFRONT=10.1.1.2 proxyserver=10.2.1.1 proxyport=3128  
proxyusername=1 proxypassword=1 rpm -Uvh scilogd-0.173.17-  
0.aix7.2.noarch.rpm
```

**NOTE:** If no proxy is used, you can remove the proxy parameters in the install command. Similarly, if a proxy is used without credentials, remove the `proxyusername` and `proxypassword` parameters in the install command, and only include the `proxyserver` and `proxyport` parameters .

**NOTE:** Also note that the **Tenant** and **URLFront** values can now be passed on the install command for AIX.

## Configuring a Windows Agent to Use a Proxy

To configure the Windows agent to use a proxy server connection:

1. Create a configuration file named **scilog\_proxy.conf** and add the following proxy information to the new file:

```
Proxy-Server
```

```
Proxy-Port
```

```
Proxy-Username
```

```
Proxy-Password
```

For example:

```
Proxy-Server 10.1.1.1
```

```
Proxy-Port 8080
```

```
Proxy-Username admin1
```

```
Proxy-Password passw0rd
```

**NOTE:** The agent requires at least the proxy server and proxy port information. If no authentication is required, leave the username and password values blank. If values are present for the username and password, the agent will try to use them. The agent currently supports Basic Authentication. The agent will discover the type of supported authentication, so there is no need to specify it in the configuration file.

2. Add the new file to the **C:\Program Files\ScienceLogic\SiloAgent\conf** directory on the SL1 server. This directory is also where the **scilog.conf** file resides.

**NOTE:** If the **scilog\_proxy.conf** file exists, the agent will try to connect to a proxy. If you do not need a proxy, remove or rename this file to prevent the agent from accessing it. Alternatively, you can leave the parameters blank in the **scilog\_proxy.conf** file. If you use the .msi installer, you can input the parameters in the dialog box for the installer.

When configuring a proxy password with the above procedure, SL1 encodes the proxy password in the **scilog\_proxy.conf** file. The password displays as a seemingly random string of characters.

**NOTE:** You can manually edit the **scilog\_proxy.conf** file, you can manually enter the proxy password. The agent will still work with a proxy password in plain text. The agent will detect if the setup script encoded the password; otherwise, it assumes the password is in plain text.

## Installing and Configuring an Agent with the Proxy Settings

You can set the proxy parameters at install time by including them as parameters for the install command, just as you do with the **Tenant** and **URLFront** values.

For example:

```
.\SiloAgent-install.exe Tenant=0 URLFront=10.1.1.2 proxyserver=10.2.1.1  
proxyport=3128 proxyusername=1 proxypassword=1
```

**NOTE:** If no proxy is used, you can remove the proxy parameters in the install command. Similarly, if a proxy is used without credentials, remove the `proxyusername` and `proxypassword` parameters in the install command, and only include the `proxyserver` and `proxyport` parameters.

**NOTE:** If no parameters are passed to **SiloAgent-install.exe**, including **Tenant** and **URLFront**, a pop-up dialog will appear where you can enter these values. If you do not require proxy credentials, you can leave those values blank.

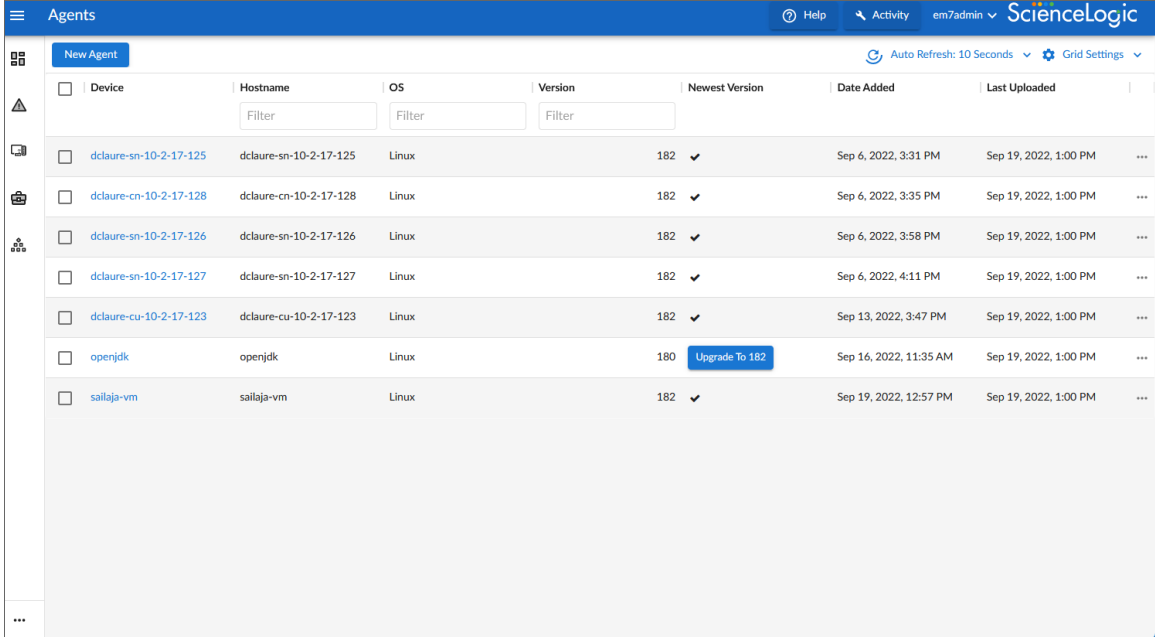
---

## Upgrading an Agent

When you have the latest version of the SL1 agent, a check mark icon ( ✓ ) appears in the **Newest Version** column for that agent. If you do not have the latest version of the agent, an **[Upgrade]** button appears in the **Newest Version** column on the **Agents** page.

To upgrade to the latest version of an agent:

1. On the **Agents** page (Devices > Agents), locate the agent you want to upgrade:



Device	Hostname	OS	Version	Newest Version	Date Added	Last Uploaded
<input type="checkbox"/>	dclaure-sn-10-2-17-125	dclaure-sn-10-2-17-125	Linux	182 ✓	Sep 6, 2022, 3:31 PM	Sep 19, 2022, 1:00 PM
<input type="checkbox"/>	dclaure-cn-10-2-17-128	dclaure-cn-10-2-17-128	Linux	182 ✓	Sep 6, 2022, 3:35 PM	Sep 19, 2022, 1:00 PM
<input type="checkbox"/>	dclaure-sn-10-2-17-126	dclaure-sn-10-2-17-126	Linux	182 ✓	Sep 6, 2022, 3:58 PM	Sep 19, 2022, 1:00 PM
<input type="checkbox"/>	dclaure-sn-10-2-17-127	dclaure-sn-10-2-17-127	Linux	182 ✓	Sep 6, 2022, 4:11 PM	Sep 19, 2022, 1:00 PM
<input type="checkbox"/>	dclaure-cu-10-2-17-123	dclaure-cu-10-2-17-123	Linux	182 ✓	Sep 13, 2022, 3:47 PM	Sep 19, 2022, 1:00 PM
<input type="checkbox"/>	openjdk	openjdk	Linux	180 <b>Upgrade To 182</b>	Sep 16, 2022, 11:35 AM	Sep 19, 2022, 1:00 PM
<input type="checkbox"/>	sailaja-vm	sailaja-vm	Linux	182 ✓	Sep 19, 2022, 12:57 PM	Sep 19, 2022, 1:00 PM

2. Click the **[Upgrade]** button for that agent. The agent starts the upgrade process, which might take up to a minute to complete.
3. Click the **[Refresh]** button in your browser to ensure that a check mark icon ( ✓ ) appears in the **Newest Version** column for that agent.

**NOTE:** It is not possible to configure the service `USER` when upgrading through the SL1 user interface as the agent will continue using the service that `USER` already uses.

It is also possible to upgrade the agent from device command line.

To upgrade from the Windows command line:

1. Open the command line as an administrator and run the new .exe installer. It is not necessary to specify **Tenant** or **URLFront** values during an upgrade as those parameters carry over.
2. Change or configure any properties described in [Installing a Windows Agent](#) directly in the upgrade command line.

To upgrade with Linux :

#### Debian or Ubuntu

```
sudo dpkg -i sl_agent.deb
```



- The `RUN_AS` state can be changed via the command line with `RUN_AS_ROOT=1` or by updating the file `/etc/scilog/run_state.txt` with the same information.
  - The command line option will always take precedence.
  - The default `RUN_AS` state is "root" when no command line `RUN_AS_ROOT=X` parameter is given and there is no `/etc/scilog/run_state.txt` file.

### RHEL or CentOS

```
sudo rpm -U sl_agent.rpm
```

- The `RUN_AS` state can be changed via the command line with `RUN_AS_ROOT=1` or by updating the file `/etc/scilog/run_state.txt` with the same information.
  - The command line option will always take precedence.
  - The default `RUN_AS` state is "root" when no command line `RUN_AS_ROOT=X` parameter is given and there is no `/etc/scilog/run_state.txt` file.

To upgrade with AIX:

```
sudo rpm -U sl_agent_aix.rpm
```

- The `RUN_AS` state can be changed via the command line with `RUN_AS_ROOT=1` or by updating the file `/etc/scilog/run_state.txt` with the same information.
  - The command line option will always take precedence.
  - The default `RUN_AS` state is "root" when no command line `RUN_AS_ROOT=X` parameter is given and there is no `/etc/scilog/run_state.txt` file.

To upgrade with Solaris:

```
sudo pkgadd -a /usr/share/overwrite.conf -d sl_agent.pkg scilogd
```

- The `RUN_AS` state can be set by adding the file `/etc/scilog/run_state.txt` with `RUN_AS_ROOT=<0|1>`.
- The default `RUN_AS` state is "root" when no `/etc/scilog/run_state.txt` file exists.

**NOTE:** Upgrades of the same version can be used to change the `RUN_AS` state.

---

## Stopping an Agent

You can use the "delete" option on the **Agents** page to stop an agent from running. When you delete an agent, SL1 stops the agent from running on the device, deletes the data gathered by that agent, and removes that agent from the **Agents** page.

**NOTE:** Using the "delete" option for an agent **does not** actually remove the agent from the device. As a best practice, use the delete process to delete the data gathered by the agent (the uninstallation process does not delete this data), and then uninstall that agent, if needed. For uninstallation details, see [Uninstalling an Agent](#). Also, ScienceLogic recommends that you delete the *device* that had the agent installed on it, or at least remove the agent-related Dynamic Applications, to prevent SL1 from attempting to use the data the agent gathered from that device.

To delete an agent and its data:

1. On the **Agents** page (Devices > Agents), locate the agent you want to delete.
2. Click the **[Actions]** button (\*\*\*) for that agent and select *Delete*. SL1 stops the agent from collecting data.

**NOTE:** You can also delete an agent from SL1 using the **Device Manager** page in the classic user interface. Click the checkbox for the device with the agent and select Delete Selected Devices from the Select Action menu at the bottom of the page.

You can also stop an agent through the command line.

To stop an agent on a Linux system (Debian, Ubuntu, RHEL, or CentOS), execute the following command:

```
sudo service scilogd stop
```

To stop an agent on an AIX system, execute the following command:

```
sudo /etc/rd.d/init.d/scilogd stop
```

To stop an agent on a Solaris system, execute the following command:

```
sudo svcadm disable scilogd
```

---

## Uninstalling an Agent

When you uninstall an agent, you remove that agent completely from SL1, but you do not lose the data collected by that agent.

**IMPORTANT:** When uninstalling the agent, be sure that the `/var/log/jmx-collector` and the `/temp/scilog` directories are removed along with the `/var/log/scilogd.log` file.

## Uninstalling a Linux Agent

To uninstall an agent on a Linux system:

1. Log in to the Linux system via the console or SSH as a user that has sudo administrator permissions.
2. For Red Hat, CentOS, and Oracle, execute the following command:

```
sudo rpm -e scilogd
```

3. For Debian and Ubuntu, execute the following command:

```
sudo dpkg -r scilogd
```

4. Optionally, you can remove the agent configuration directory from the Linux system. The configuration directory can be found at:

```
/etc/scilog (rm -rf /etc/scilog)
```

## Uninstalling an AIX Agent

To uninstall an agent on an AIX system:

1. Log in to the AIX system via the console or SSH as a user that has sudo administrator permissions.
2. To determine which package version is currently running, execute the following command:

```
sudo rpm -ga|grep -i scilogd
```

To determine which package version is running from a previous command, execute the following command:

```
sudo rpm -e
```

3. Optionally, you can remove the agent configuration directory from the Linux system. The configuration directory can be found at:

```
/etc/scilog (rm -rf /etc/scilog)
```

## Uninstalling a Solaris Agent

To uninstall an agent on a Solaris system:

1. Log in to the Solaris system via the console or SSH as a user that has sudo administrator permissions.
2. To uninstall the agent, execute the following command:

```
sudo pkgrm scilogd
```

3. Optionally, you can remove the agent configuration directory from the Linux system. The configuration directory can be found at:

```
/etc/scilog (rm -rf /etc/scilog)
```

## Uninstalling a Windows Agent

To uninstall an agent on a Windows system:

1. On the Windows system, open the **Control Panel**.
2. Go to the **Programs and Features** page (Control Panel > Programs > Uninstall a program).
3. Select the SiloAgent program from the list, and then click [**Uninstall**].

---

## Installing an Agent from the Device Manager Page

If you are using a distributed SL1 system *without* the SL1 Extended Architecture (which includes Compute Nodes, Storage Nodes, and a Management Node), you **must** use the **Agent Installation** page to install the agent. You can access the **Agent Installation** page from the **Device Manager** page.

TCP port 443 must be open between the Message Collector and the device on which an agent is installed.

**NOTE:** If you are using an SL1 system with the SL1 Extended Architecture, you **must** use the **Agents** page (Devices > Agents) to install agents, including AIX and Solaris agents. AIX and Solaris agents are not available for non-SL1 Extended systems.

**NOTE:** If you are *not* using the SL1 Extended Architecture, the **Agents** page (Devices > Agents) will not be available in the SL1 user interface.

To install an agent from the **Device Manager** page, you first need to gather installation information from the **Agent Installation** page:

- For a Linux system, the **Agent Installation** page provides commands that must be executed on that system.
- For a Windows system, the **Agent Installation** page provides an executable file to run on the Windows system.

To gather the necessary commands and executable files to install an agent on a device:

1. Go to the **Device Manager** page (Device > Device Manager in the SL1 user interface, or Registry > Devices > Device Manager in the classic user interface).
2. Click [**Actions**] and select *Download/Install Agent*. The **Agent Installation** page appears.

3. Complete the following fields:

- **Select an OS.** Select the operating system running on the device on which you want to install the agent.

**NOTE:** If you require a FIPS-compliant version of the SL1 agent, select *RedHat/CentOS 64-bit (OS Libs)*.

- **Select an Organization.** Select an organization from the list of possible organizations. The list of organizations is dependent on your user account. If the agent discovers a new device, that device will be associated with the organization you select here.

**NOTE:** If you are installing an agent on a device that has already been discovered, you must select the organization that is already aligned with the existing device.

- **Select a Message Collector.** Select the Message Collector to which the agent will send its collected data.
4. If you selected a Linux operating system in the **Select an OS** field, the **Agent Installation** page displays a list of commands to execute on the Linux system. Copy the commands for use during the [installation on the Linux device](#).
  5. If you selected a Windows operating system in the **Select an OS** field, the **Agent Installation** page displays a **Download Windows Agent** link. Click the link and save the executable file for use during the [installation on the Windows device](#).

**TIP:** If you are installing an agent on multiple devices that run the same operating system, are part of the same organization, and connect to the same Message Collector, you can re-use the same commands or executable file on each of those devices.

## Installing an Agent on a Linux System

To install an agent on a Linux system:

Copy and execute the appropriate commands from the Agent Installation page to download and install the agent. For more information, see [Installing a Linux agent](#) for details about install commands.

**NOTE:** SL1 supports the use of proxy server connections when using the SL1 agent on Linux systems. To do so, open a command window on the target server and first configure curl to use a proxy server using the `CURLOPT_PROXY` option, and then to use a username and password combination for that proxy server using the `CURLOPT_PROXYUSERPWD` option.

## Upgrading an Agent on a Linux System

To upgrade the agent on a Linux system:

When using the classic SL1 user interface, agents must be upgraded from the device command line. For more information, see [Upgrading an Agent](#).

## Installing an Agent on a Windows System

To install an agent on a Windows system:

1. Copy the SiloAgent-install.exe file you downloaded from the **Agent Installation** page to the Windows system.
2. Execute the appropriate command copied from the **Agent Installation** page to install the agent. For more details about install commands see [Installing a Windows Agent](#).

**NOTE:** SL1 supports the use of proxy server connections when using the SL1 agent on Windows systems. In Windows 10, go to Settings > Network & Internet > Proxy to set up a proxy; in older versions of Windows, you can do this by clicking on the Control Panel and going to Internet Options > Connections > LAN Settings. If a proxy server requires authentication, you will need to provide a username and password before traffic is allowed through the proxy server and to the Internet.

## Configuring an SL1 Agent to Work with Phone Home Collectors

If you are using PhoneHome Message Collectors in a Distributed Architecture, you will need to configure the settings for the Message Collector to prevent SL1 from incorrectly configuring the SL1 agent to use the loopback IP address when it should be using the local IP address for the Message Collector.

If you do not perform the configuration steps below, when you try to install the agent from a Message Collector when it is configured as a PhoneHome Collector, the URLs for both the download and the internal configuration of the agent reference the loopback address instead of the accessible public/private IP address of the Message Collector.

**NOTE:** For more information about PhoneHome Collectors, see the "Configuring SL1 for PhoneHome Communication" chapter in the *Installation* manual.

To configure the SL1 agent to work with a PhoneHome Collector:

1. Go to the **Appliance Manager** page (System > Settings > Appliances):

The screenshot shows the 'Appliance Manager' interface. At the top right, there are links for 'System Status', 'Output', 'Reset', and 'Guide'. Below these is a configuration form with the following fields:

- Host Name:
- IP Address:
- Model type:
- Description:

Buttons for 'Save' and 'Save As' are located to the right of the form. Below the form is a table listing various appliances:

Name	IP Address	Module Type	Collector Group	Description	Build	MetaDB	Capacity	Allocation	ID	Validated	Endpoint	Edit Date	Edit User	Create Date	
1. st206-cdb	10.64.164.241	Database	n/a	Database: 10.64.164.241	10.1.0.BETA3 r2097	10.4.12	7,000	n/a	1	Yes	--	2020-05-19 18:50:56	em7admin	2020-05-19	
2. st206-ap	10.64.164.242	Administration Portal	n/a	Application Server: 10.64.164.242	10.1.0.BETA3 r2097	10.4.12	n/a	n/a	2	Yes	--	2020-05-19 18:49:02	em7admin	2020-05-19	
3. st206-cu-benedict	10.64.164.253	Data Collection Unit	CUG-Benedict	collector unit: 10.64.164.253	10.1.0.BETA3 r2097	10.4.12	n/a	n/a	9	19	Yes	--	2020-05-19 18:52:07	em7admin	2020-05-19
4. st206-cu-moss1	10.64.164.251	Data Collection Unit	CUG_MOSS	collector unit: 10.64.164.251	10.1.0.BETA3 r2097	10.4.12	n/a	0	14	Yes	--	2020-05-19 18:51:58	em7admin	2020-05-19	
5. st206-cu-moss2	10.64.164.252	Data Collection Unit	CUG_MOSS	collector unit: 10.64.164.252	10.1.0.BETA3 r2097	10.4.12	n/a	1	16	Yes	--	2020-05-19 18:52:01	em7admin	2020-05-19	
6. st206-cu-racecondition1	10.64.164.246	Data Collection Unit	CUG-RaceCondition	collector unit: 10.64.164.246	10.1.0.BETA3 r2097	10.4.12	n/a	2	8	Yes	--	2020-05-19 18:51:36	em7admin	2020-05-19	
7. st206-cu-racecondition2	10.64.164.247	Data Collection Unit	CUG-RaceCondition	collector unit: 10.64.164.247	10.1.0.BETA3 r2097	10.4.12	n/a	3	9	Yes	--	2020-05-19 18:51:37	em7admin	2020-05-19	
8. st206-cu-racecondition3	10.64.164.248	Data Collection Unit	CUG-RaceCondition	collector unit: 10.64.164.248	10.1.0.BETA3 r2097	10.4.12	n/a	1	10	Yes	--	2020-05-19 18:51:35	em7admin	2020-05-19	
9. st206-cu-mgs	10.64.164.245	Data Collection Unit	CUG-Solutions	collector unit: 10.64.164.245	10.1.0.BETA3 r2097	10.4.12	n/a	2	7	Yes	--	2020-05-19 18:51:39	em7admin	2020-05-19	
10. st206-cu-scrumandcoke	10.64.164.254	Data Collection Unit	CUG-SAC	collector unit: 10.64.164.254	10.1.0.BETA3 r2097	10.4.12	n/a	0	21	Yes	--	2020-05-19 18:52:04	em7admin	2020-05-19	
11. st206-cu-shared	10.64.164.244	Data Collection Unit	CUG-Solutions	collector unit: 10.64.164.244	10.1.0.BETA3 r2097	10.4.12	n/a	7	6	Yes	--	2020-05-19 18:51:40	em7admin	2020-05-19	
12. st206-cu-usualsuspects1	10.64.164.249	Data Collection Unit	CUG-UsualSuspects	collector unit: 10.64.164.249	10.1.0.BETA3 r2097	10.4.12	n/a	2	11	Yes	--	2020-05-19 18:51:39	em7admin	2020-05-19	
13. st206-cu-usualsuspects2	10.64.164.250	Data Collection Unit	CUG-UsualSuspects	collector unit: 10.64.164.250	10.1.0.BETA3 r2097	10.4.12	n/a	2	12	Yes	--	2020-05-19 18:51:51	em7admin	2020-05-19	
14. st206-mc	10.64.164.243	Message Collection Unit	--	collector unit: 10.64.164.243	10.1.0.BETA3 r2097	10.4.12	n/a	n/a	5	Yes	10.64.164.243	2020-05-19 18:52:06	em7admin	2020-05-19	
15. st206-cn01	10.64.164.233	Compute Node	n/a	Compute Node: 10.64.164.233	Unknown	10.4.12	n/a	n/a	54	Yes	--		em7admin		
16. st206-cn02	10.64.164.234	Compute Node	n/a	Compute Node: 10.64.164.234	Unknown	10.4.12	n/a	n/a	55	Yes	--		em7admin		
17. st206-cn03	10.64.164.235	Compute Node	n/a	Compute Node: 10.64.164.235	Unknown	10.4.12	n/a	n/a	56	Yes	--		em7admin		

2. In the **Message Collection Unit** row, click the Agent Endpoint Configuration icon (🔗) for the agent. The **Agent Endpoint Configuration** modal appears:

Agent Endpoint Configuration

Endpoint [13] Reset

**Appliance Details**  
collector unit: 10.64.171.180: 10.64.171.180

**Download Link Visibility**  
[ Enabled ]

**Endpoint Name for Download Link**  
10-1-22-180-MC

**Hostname/IP for Agent Download**  
10.1.22.180

**Hostname/IP for Data Collection**  
(Leave blank to use value specified in streamer settings on appliance)  
10.1.22.180

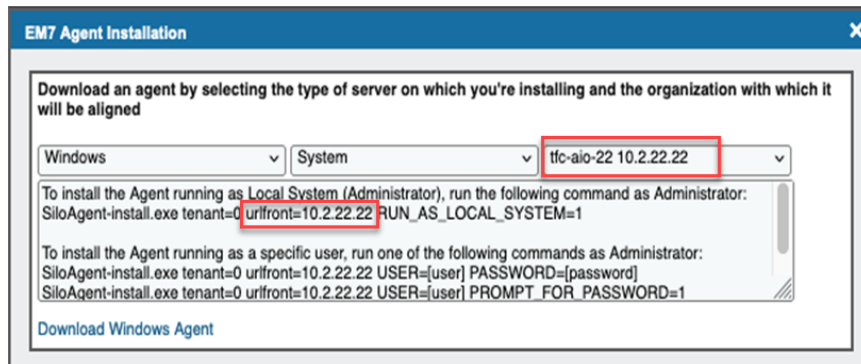
**Default CUG Alignment**  
CUG-Solutions

Save

3. Make sure that the configuration information under the **Appliance Details** field is accurate.
4. Complete the following fields:



- **Download Link Visibility.** Select whether you want to display the download link in the **Agent Installation** modal page.
  - If you select *Enabled*, the link to the Message Collector appears on the **Agent Installation** modal, which is available on the **Device Manager** page (Devices > Device Manager or Registry > Device > Device Manager in the classic user interface) when you click the **[Actions]** button and select *Download/Install Agent*:



- If you select *Disabled*, the link to the Message Collector does not appear in the **Agent Installation** modal page.
- **Endpoint Name for Download Link.** This field contains the name of the Message Collector. If the name is not populated by default, you can enter the name.
  - **Hostname/IP for Agent Download.** Make sure that this field contains the IP address for the Message Collector.
  - **Hostname/IP for Data Collection.** Enter the IP address for the Message Collector.
  - **Default CUG Alignment.** Select the Collector Group aligned with the agent.
5. Click **Save**. The agent is now configured to work with a Message Collector when it is configured as a PhoneHome Collector.

## Remotely Installing an SL1 Agent

This section covers how to configure remote installations of the Linux agent and the Windows agent.

**NOTE:** The following procedures are examples of how you could set up these installations, using Ansible as the tool for the installation. You might use Ansible in a different way, or you might use another tool altogether. These examples represent just one way to configure the remote installation of agents.

## Remotely Installing a Linux Agent

This topic provides an example of how you can use Ansible to remotely install the SL1 Linux agent. Please refer to the [Ansible product documentation](#) as needed.

To remotely install the Linux agent using Ansible:

1. Set up your Ansible control node, using the documentation at [Installing Ansible](#).
2. Download the SL1 Linux agent installers from the **Agents** page (Devices > Agents > New Agent button) in SL1. For example:
  - **silos-agent-x86\_64.rpm**
  - **silos-agent-x86\_64.deb**
3. Create the **hosts.yml** file, using the following code as an example:

```
servers:
  hosts:
    centos_host:
      ansible_host: 172.16.54.195
      ansible_user: root
    debian_host:
      ansible_host: 10.2.16.125
      ansible_user: root
```

4. Create the **playbook.yml** file, using the following code as an example:

```
---
- hosts: servers
  vars:
    tenant: 0
    url_front: 10.2.16.49
    package_rpm: silo-agent-x86_64.rpm
    package_deb: silo-agent-x86_64.deb
  tasks:

    - name: check if agent exists
      stat:
        path: /usr/bin/scilogd
      register: agent_binary

    - name: copy rpm installer
      copy:
        src: "{{ package_rpm }}"
        dest: ~/{{ package_rpm }}
        when: (agent_binary.stat.exists == False and ansible_facts['os_
family'] == 'RedHat')

    - name: copy deb installer
      copy:
        src: "{{ package_deb }}"
        dest: ~/{{ package_deb }}
        when: (agent_binary.stat.exists == False and ansible_facts['os_
family'] == 'Debian')

    - name: install rpm agent
      shell: TENANT={{ tenant }} URLFRONT={{ url_front }} rpm -ihv ~/{{
package_rpm }}
      args:
        creates: /usr/bin/scilogd
        when: (agent_binary.stat.exists == False and ansible_facts['os_
family'] == 'RedHat')

    - name: install deb agent
      shell: TENANT={{ tenant }} URLFRONT={{ url_front }} dpkg -i ~/{{
package_deb }}
```

```
    args:
      creates: /usr/bin/scilogd
      when: (agent_binary.stat.exists == False and ansible_facts['os_
family'] == 'Debian')

- name: remove rpm installer
  file:
    path: ~/{{ package_rpm }}
    state: absent
    when: (agent_binary.stat.exists == False and ansible_facts['os_
family'] == 'RedHat')

- name: remove deb installer
  file:
    path: ~/{{ package_deb }}
    state: absent
    when: (agent_binary.stat.exists == False and ansible_facts['os_
family'] == 'Debian')
```

5. Run the playbook by executing the following command:

```
ansible-playbook playbook.yml -i hosts.yml --ask-pass
```

**NOTE:** This example uses the `--ask-pass` parameter so that you are prompted for a single password to use with all remote computers.

The following code is an example of the output:

```
$ ansible-playbook playbook.yml -i hosts.yml --ask-pass
SSH password:

PLAY [servers]
*****
*****

TASK [Gathering Facts]
*****
*****

ok: [centos_host]
ok: [debian_host]

TASK [check if agent exists]
*****
*****

ok: [centos_host]
ok: [debian_host]

TASK [copy rpm installer]
*****
*****

skipping: [debian_host]
changed: [centos_host]

TASK [copy deb installer]
*****
*****

skipping: [centos_host]
changed: [debian_host]
```

```

TASK [install rpm agent]
*****
*****
skipping: [debian_host]
changed: [centos_host]

TASK [install deb agent]
*****
*****
skipping: [centos_host]
changed: [debian_host]

TASK [remove rpm installer]
*****
*****
skipping: [debian_host]
changed: [centos_host]

TASK [remove deb installer]
*****
*****
skipping: [centos_host]
changed: [debian_host]

PLAY RECAP
*****
*****
centos_host           : ok=5    changed=3    unreachable=0
failed=0    skipped=3    rescued=0    ignored=0
debian_host          : ok=5    changed=3    unreachable=0
failed=0    skipped=3    rescued=0    ignored=0

```

## Remotely Installing a Windows Agent

This topic provides examples of how you can use the PowerShell console or Ansible to remotely install the SL1 Windows agent. Please refer to the PowerShell or Ansible product documentation as needed.

To remotely install the Windows agent using `InstallSoftwareRemotely.ps1`:

1. Download the Windows Agent installer from the **Agents** page (Devices > Agents > New Agent button) in SL1 and add the installer to your domain-controller in its own directory. For example, **C:\files\SiloAgent-install.exe**.

2. Download the third-party PowerShell script to your domain-controller.
3. Create a comma-separated value (CSV) file with the names of the computers where you want to install the agent. For example: **C:\computers.csv**, such as:

```
rstlsw12t16ls01
```

```
rstlsw12t16ex01
```

```
computer3
```

```
computer99
```

4. Run the following command from a PowerShell console:

```
.\InstallSoftwareRemotely.ps1 -CSV C:\computers.csv -AppPath 'C:\foo\SiloAgent-install.exe' -AppArgs "Tenant=<organization_id> urlFront=<streamer_URL> RUN_AS_LOCAL_SYSTEM=1"
```

where:

- `-CSV` uses the full path to the `.csv` file you created in step 3, above.
- `-AppPath` uses the full path to the installer in step 1, above. Make sure the file is in its own directory.
- `RUN_AS_LOCAL_SYSTEM=1` is an example configuration. See [Configuring the Windows Agent](#) for more information.
- You can optionally use the `-Credential` option to be prompted for a different username and password to access all remote machines. Otherwise the script will use your current login account to access all remote machines.

The following code is an example of the output:

```
PS C:\> .\InstallSoftwareRemotely.ps1 -CSV C:\computers.csv -AppPath 'C:\-foo\SiloAgent-install.exe' -AppArgs "Tenant=0 URLFront=streamer-shared-e2e-testing.int.sciencelogic.com"
Start remote installation on 2020-56-20 09:56:17
No credential specified. Using logon account
-----
Attempt 1 of 5
-----
COMPUTER RSTLSW12T16LS01 (1 of 2)
Coping C:\foo to \\rstlsw12t16ls01\C$\temp
Installing SiloAgent-install.exe on RSTLSW12T16LS01
Executing C:\temp\foo\SiloAgent-install.exe Tenant=0 URLFront=streamer-shared-e2e-testing.int.sciencelogic.com
```

```
Deleting C:\temp\foo
SiloAgent-install.exe installed successfully.
COMPUTER RSTLSW12T16EX01 (2 of 2)
Coping C:\foo to \\rstlsw12t16ex01\C$\temp
Installing SiloAgent-install.exe on RSTLSW12T16EX01
Executing C:\temp\foo\SiloAgent-install.exe Tenant=0 URLFront=streamer-
shared-e2e-testing.int.sciencelogic.com
Deleting C:\temp\foo
SiloAgent-install.exe installed successfully.
-----
100% Success installing SiloAgent-install.exe on 2 of 2 computers:
rstlsw12t16ls01 rstlsw12t16ex01
```

To remotely install the Windows agent using Ansible:

1. Set up your Ansible control node. Refer to Ansible documentation for installing on other operating systems: [Installing Ansible with pip](#).
2. Update your Ansible control node to manage Windows servers by running the following command:

```
sudo pip3 install pywinrm
```

3. Ensure that remote machines are set up with PowerShell 3 or later:

```
https://github.com/jborean93/ansible-
windows/blob/master/scripts/Upgrade-PowerShell.ps1
```

4. Ensure that remote machines are setup for Windows Remote Management:

```
https://github.com/ansible/ansible/blob/devel/examples/scripts/Config
ureRemotingForAnsible.ps1
```

5. Download the SL1 Windows agent installer, **SiloAgent-install.exe**, from the **Agents** page (Devices > Agents > New Agent button) in SL1.



6. Create the **hosts.yml** file, using the following code as an example:

```
group1:
  hosts:
    win2008r2:
      ansible_host: 10.2.16.41
      ansible_user: Administrator
    win2012:
      ansible_host: 10.2.16.42
      ansible_user: Administrator
    win2012r2:
      ansible_host: 10.2.16.40
      ansible_user: Administrator
group2:
  hosts:
    dc01:
      ansible_host: 10.2.16.60
      ansible_user: administrator@T16TESTDOMAIN
    ls01:
      ansible_host: 10.2.16.62
      ansible_user: administrator@T16TESTDOMAIN
    es01:
      ansible_user: administrator@T16TESTDOMAIN
      ansible_host: 10.2.16.64
```

6. Create the **my\_playbook.yml** file, using the following code as an example:

```
---
- hosts: group1
  gather_facts: no
  tasks:
    - name: Ping Windows Hosts
      win_ping:
    - name: Copy agent installer
      win_copy:
        src: ./SiloAgent-install.exe
        dest: C:\Temp\SiloAgent-install.exe
    - name: Exec Installer
      win_command: C:\Temp\SiloAgent-install.exe Tenant=yyy
      URLFront=zzz
    - name: Remove agent installer
      win_file:
        path: C:\Temp\SiloAgent-install.exe
        state: absent
```

7. Run the playbook by executing the following command:

```
ansible-playbook playbook.yml -i hosts.yml --ask-pass
```

**NOTE:** This example uses the `--ask-pass` parameter so that you are prompted for a single password to use with all remote computers. Alternately, you could include the password in `hosts.yml` or `./group_vars/all.yml`.

The following code is an example of the output:

```
[em7admin@jkay88 demo_ansible]$ ansible-playbook my_playbook.yml -i hosts.yml --ask-pass
SSH password:

PLAY [group1]
*****
*****

TASK [Ping Windows Hosts]
*****
*****

ok: [win2012]
```

```

ok: [win2008r2]

TASK [Copy agent installer]
*****
*****
changed: [win2012]
changed: [win2012r2]
changed: [win2008r2]

TASK [Exec Installer]
*****
*****
changed: [win2012]
changed: [win2012r2]
changed: [win2008r2]

PLAY RECAP
*****
*****
win2008r2          : ok=3    changed=2    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
win2012          : ok=3    changed=2    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
win2012r2       : ok=3    changed=2    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0

```



---

# Chapter

# 3



## Configuring the SL1 Agent

---

### Overview

This chapter covers how to configure the agent based on the SL1 architecture you are using. This chapter also describes additional agent configurations, such as enabling extensible collection with an SL1 agent.

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:

<i>Configuring Agent Monitoring Based on SL1Architecture</i> .....	62
<i>Configuring Agent Settings from the Device Investigator Page</i> .....	63
<i>Configuring an Agent to Export Data to Skylar</i> .....	66
<i>Configuring Extensible Collection</i> .....	69
<i>Configuring Run Book Automations for an SL1 Agent</i> .....	71
<i>Configuring an SL1 Agent on the Device Manager Page</i> .....	73

---

## Configuring Agent Monitoring Based on SL1 Architecture

Based on your SL1 architecture, you will have access to different workflows for configuring the monitoring settings on an SL1 agent:

- If the SL1 agent is used in an **SL1 Extended Architecture** (which includes Compute Nodes, Storage Nodes, and a Management node), use the following configuration workflows:
  - To configure basic settings related to how the agent collects data, see [Configuring Agent Settings from the Device Investigator Page](#).
  - To configure log file monitoring and Windows event log monitoring with the agent, see [Monitoring Device Logs Using an Agent](#).
  - To configure and view data collected by the agent, see the relevant tabs on the **Device Investigator** page, including the following tabs:
    - Settings
    - Collections
    - Interfaces
    - Logs
    - Processes
    - Services

For more information about these tabs, see the "Using the Device Investigator" chapter in the **Device Management** manual.

- If the SL1 agent is used in an **SL1 Distributed Architecture** (where the SL1 Agent collects and transfers data to a Message Collector), use the following configuration workflows:
  - To configure basic settings related to how the agent collects data, see [Configuring an SL1 Agent on the Device Manager Page](#).
  - To configure log file monitoring and Windows event log monitoring with the agent, see [Monitoring Device Logs Using an Agent](#).
  - To configure and view data collected by the agent, see the relevant tabs on the **Device Investigator** page, including the following tabs:
    - Settings
    - Collections

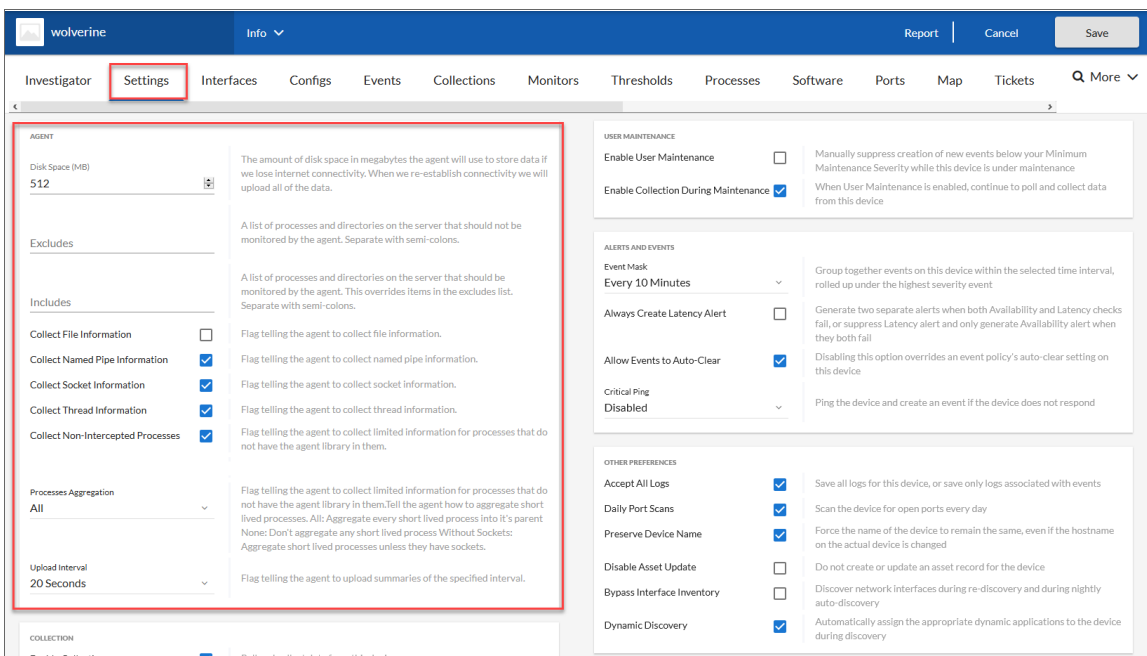
- Interfaces
- Logs
- Processes
- Services

For more information about these tabs, see the "Using the Device Investigator" chapter in the *Device Management* manual.

- To configure system vitals monitoring with the agent, see the "Monitoring Vitals Using an Agent" chapter of the *Device Infrastructure Health* manual.
- To configure port monitoring with the agent, see the "Monitoring Ports" chapter of the *Device Infrastructure Health* manual.

## Configuring Agent Settings from the Device Investigator Page

If a device does *not* have an agent running on it, the **[Settings]** tab of the **Device Investigator** page for the device will display the **Collection** section at top left. However, if the device has an agent running on it, the **Agents** section appears above the **Collection** section:



In the **Agents** section, you can configure the disk space, excludes, includes, and other settings related to how you want the agent to collect data.

**NOTE:** The **Agents** section appears on the **[Settings]** tab *only* when you are using the SL1 Extended Architecture. If your version of SL1 does *not* have an **Agents** section, see [Configuring an SL1 Agent on the Device Manager Page](#).

To configure agent settings:

1. From the **Devices** page, select the device with the agent that you want to configure. The **Device Investigator** page appears.

**TIP:** To quickly find all devices on the **Devices** page that have agents installed on them, type "agent" in the **Search** field and select the "ANY: agent" search.

2. Click the **[Settings]** tab and click **[Edit.]**
3. In the **Agent** section, complete the following fields, as needed:



- **Disk Space.** Specify the amount of disk space in MB that the agent can use to store data. If the agent loses connectivity to SL1, this disk space will be used to store collected data until the connection to SL1 is restored. When connectivity is re-established, the agent uploads all of its stored data.
- **Excludes.** Type a list of processes and directories, separated by semi-colons, that you do not want the agent to monitor.
- **Includes.** Type a list of processes and directories, separated by semi-colons, that you want the agent to monitor. This field ensures that specific processes are monitored.

**NOTE:** If a process or directory is included in both the **Excludes** field and the **Includes** field, the item in the **Includes** field will override the item in the **Excludes** field.

- **Collect File Information.** Select this option if you want the agent to report the names of files accessed by each monitored process.
- **Collect Named Pipe Information.** Select this option if you want the agent to collect named pipe information.
- **Collect Socket Information.** Select this option if you want the agent to collect socket information.
- **Collect Thread Information.** Select this option if you want the agent to collect thread information.
- **Collect Non-Intercepted Processes.** Select this option if you want the agent to collect limited information for processes that do not contain the agent library.
- **Processes Aggregation.** Specify how you want the agent to collect limited information for processes that do not have the agent library in them, and how to aggregate short-lived processes. Your options include the following:
  - *All:* Aggregate every short-lived process into its parent.
  - *None:* Do not aggregate any short-lived process.
  - *Without Sockets:* Aggregate short-lived processes unless those processes have sockets.
- **Upload Interval.** Specify how often the agent should upload data. Your options include the following:

- **20 Seconds.** Upload a data snapshot every 20 seconds.
- **60 Seconds.** Upload a data summary every 60 seconds. This is the default setting starting with SL1 version 11.1.0, and version 174 of the Linux agent and version 133 for the Windows agent. This option uses an improved data format that requires fewer SL1 resources. The SL1 agent continues to internally collect and poll data every 20 seconds, but the agent summarizes and uploads that data every 60 seconds. There is no data loss even though the data is uploaded less frequently.

**NOTE:** Starting with SL1 version 11.3.0, if you specify 60 seconds for the upload interval, the summary upload now will include "watched" or "monitored" files, just like the snapshot upload does.

4. Click the **[Save]** button to save your configuration settings for the agent.

**TIP:** You can view and configure additional agent settings from the **Device Investigator** page for the device on which the agent is installed, including the following tabs: **[Interfaces]**, **[Processes]**, **[Services]**, and **[Collections]**. For more information about these tabs, see the *Using the Device Investigator* chapter in the *Device Management* manual.

---

## Configuring an Agent to Export Data to Skylar

You can send agent data to Skylar AI so it can analyze the data for anomaly detection, predictive alerting, and new data visualizations.

For Gen1 agents, which use the SL1 Distributed Environment, you only need to configure the in the "Getting Started with Skylar Analytics" topic in the *Skylar Analytics* manual.

For Gen3 agents, which use the SL1 Extended Environment, you will need to complete the steps in the "Getting Started with Skylar Analytics" topic, and then complete the following additional procedures.

### Enabling Skylar Export Using Deploy

After you complete the steps in the "Getting Started with Skylar Analytics" topic:

1. Retrieve the following values from the `/etc/sl-otelcol/sl-otelcol.conf` and `/etc/sl-otelcol/sl-otelcol-system-id.conf` files:
  - `OTELCOL_SYSTEM_ID`
  - `OTELCOL_SKYLAR_ENDPOINT`
  - `OTELCOL_SKYLAR_KEY`
  - `OTELCOL_SKYLAR_KEY_HEADER`
2. On the Management Node, use the prior values from the Database Server and add them to `sl1x.inv.yml`:

- `enable_skylar_integration: true`
- `skylar_otel_endpoint: "<OTELCOL_SKYLAR_ENDPOINT>"`
- `skylar_otel_key: "<OTELCOL_SKYLAR_KEY>"`
- `skylar_otel_key_header: "<OTELCOL_SKYLAR_KEY_HEADER>"`
- `skylar_system_id: "<OTELCOL_SYSTEM_ID>"`

3. Run the following command:

```
run docker-compose -f sl1x-deploy/docker-compose.external.yml run --rm deploy app
```

As long as `deploy` is set to run for SL1 12.3.0 and later, the deploy will pull down the necessary services and configure them to enable the export of data to Skylar.

## Manually Enabling Skylar

This procedure enables the export of data to the Skylar platform from SL1 Extended by installing or upgrading helm charts from the command line.

On the Database Server, complete the following steps if you have not done them already:

1. Generate the Skylar config files:

```
sudo sl-otelcol-mgmt.py --skylar-api-key <API KEY --skylar-endpoint <SKYLAR_ENDPOINT> --skylar-all --simple-logging skylar
```

2. Ensure that the Skylar service connection is enabled on the Database Server from the SL1 user interface by going to the **Service Connections** page (Manage > Service Connections) and clicking **[Add Service Connection]**.
3. Get the following values from `/etc/sl-otelcol/sl-otelcol.conf` file:
  - `OTELCOL_SYSTEM_ID`
  - `OTELCOL_SKYLAR_ENDPOINT`
  - `OTELCOL_SKYLAR_KEY`

4. On the Extended Management Node, use the values from **sl-otelcol.conf** when installing the helm charts that enable Skylar:

```
helm repo update s11

helm upgrade --version 1.0.42 agent-vitals-service s11/agent-vitals-
service
--set env.SKYLAR_EXPORT_ENABLED=true
helm upgrade --version 1.3.8 da-postprocessing-service s11/da-
postprocessing-service
--set skylarExportEnabled=True -f output-files/da-postprocessing-
service-values.yml
helm upgrade --version 2.1.26 avail-store s11/avail-store
--set skylarExportEnabled=true
-f output-files/avail-store-values.yml
helm upgrade --version 2.1.32 interface-store s11/interface-store
--set skylarExportEnabled=true -f output-files/interface-store-
values.yml

# Might need to run with --install if not in place already.
helm upgrade --version 0.0.2 apl-optel-publisher s11/apl-optel-
publisher
--set optelSettings.skylar.enabled=true
--set optelSettings.skylar.endpoint=<OTELCOL_SKYLAR_ENDPOINT>
--set optelSettings.skylar.key=<OTELCOL_SKYLAR_KEY>
"--set optelSettings.skylar.systemId=<OTELCOL_SYSTEM_ID>
```

**NOTE:** Line breaks were added to the lines of code, above, to allow the code sample to display properly.

After the helm charts are installed, data will begin exporting to the Skylar platform.

## Validating the Skylar Export

Check the logs for any of the services:

- agent-vitals-service
- da-postprocessing-service
- avail-store
- interface-store

You should see logging similar to the following:

```
interface-store-8548989dcc-hjkzz 2024-09-17
15:30:00,861::INFO::1::interface_store.handlers.223:::Sending skylar
message for did: 5, sink: skylar.data
```

- apl-optel-publisher

```
apl-optel-publisher-67b7c6558b-shb2z 2024-09-17
15:31:21,643::INFO::1::apl_optel_publisher.handlers.109::: ** Published 1
payloads apl-optel-publisher-67b7c6558b-shb2z {'__OTLP_Resource__':
{'attributes': {'sl.service.name': 'Extended', 'sl.datatype.name':
'dynamic_app'}, 'schema_url': ''}}
```

If the helm charts were installed prior to 12.3.0, you might need to manually create the Kafka topic:

```
JMX_PORT=5557 /opt/bitnami/kafka/bin/kafka-topics.sh --bootstrap-server
kafka-service-headless:9092 --create --topic skylar.data --partitions 5 --
replication-factor 3
```

---

## Configuring Extensible Collection

You can use the SL1 agent for "extensible collection", where you align an SL1 agent with Dynamic Applications to gather metrics and attributes from other infrastructures and applications.

**Dynamic Applications** are customizable policies, created for a specific vendor and a specific type of device or system, that tell SL1:

- What data to collect from devices
- How to present the data that has been collected
- When to generate alerts and events based on the data that has been collected

SL1 includes Dynamic Applications for the most common hardware and software. You can customize these default Dynamic Applications to best work in your environment. You can also create custom Dynamic Applications.

You can align the agent with PowerShell Dynamic Applications (for Windows agents) and JMX Dynamic Applications (for Linux agents).

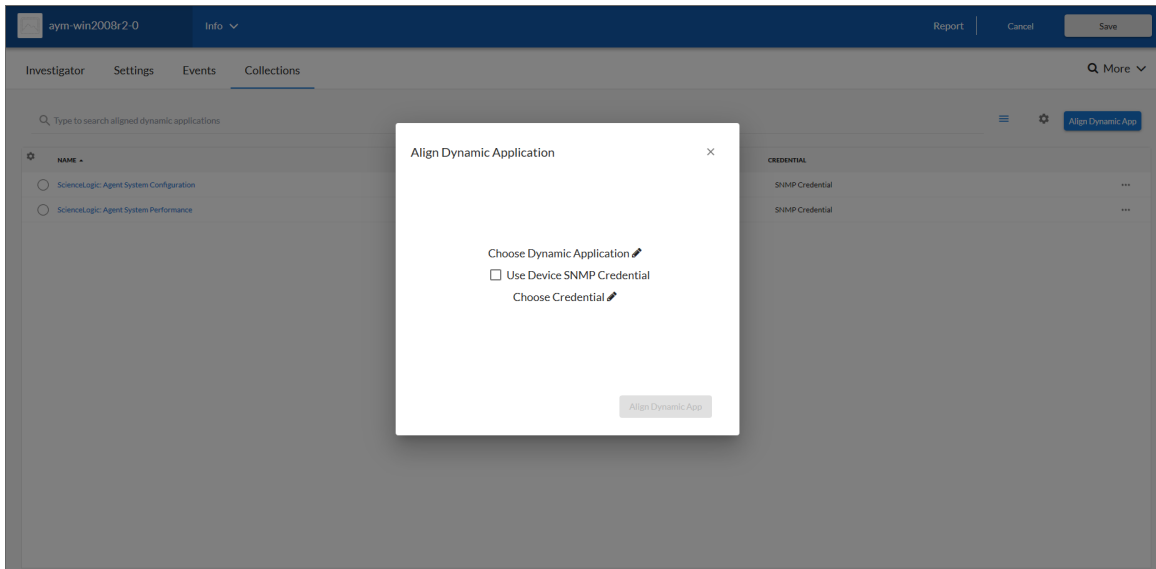
In addition, Dynamic Applications that leverage the Low Code No Code CLI/SSH framework can execute using the agent.

To enable extensible collection:

1. On the **Devices** page, search for the device with the agent installed on it.

**TIP:** To quickly find all devices that have agents installed on them, type "agent" in the **Search** field on the **Devices** page, and select the "ANY: agent" search.

2. Select the device. The **Device Investigator** page appears.
3. Click the **[Collections]** tab, click **[Edit]**, and click **[Align Dynamic App]**. The **Align Dynamic Application** window appears:

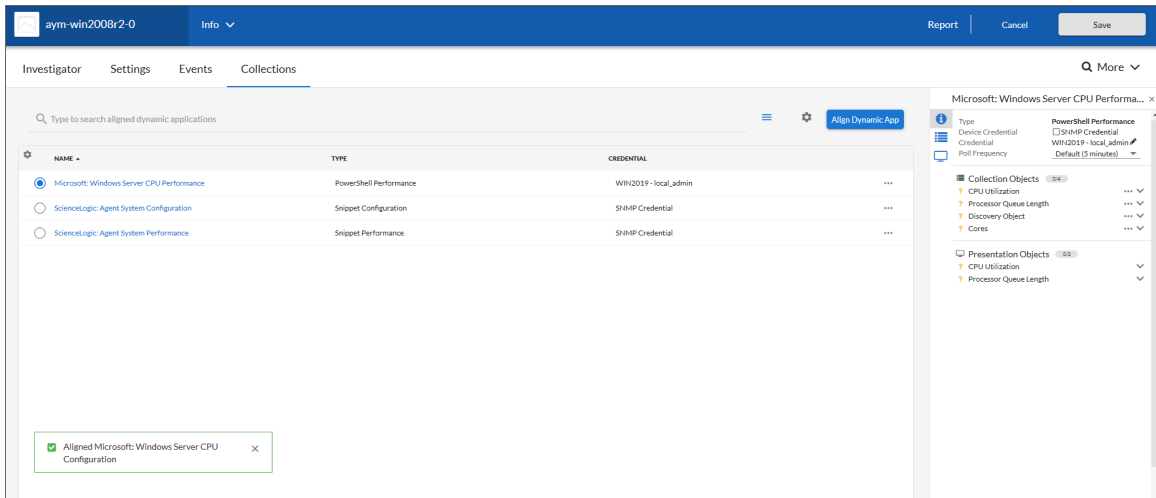


4. Click **Choose Dynamic Application** and search for the Dynamic Application you want to align.
5. Select the Dynamic Application and click **[Select]**. The **Align Dynamic Application** window appears again.
6. Click **Choose Credential** to search for a credential for the agent. The **Choose Credential** page appears.

**NOTE:** The **Account Type** for the credential must be *Local* for the agent. Do not use the SNMP credential.

7. Select the credential and click **[Select]**. The **Align Dynamic Application** window appears again.

- Click **[Align Dynamic App]**. The **[Collections]** tab appears, and a pop-up message displays when the Dynamic Application is aligned.



- Click **[Save]**.
- To view the metrics from that newly aligned Dynamic Application, click the **[Investigator]** tab.
- Click the **[EDIT]** button in the right-hand pane of the **[Investigator]** tab.
- In the **Add a metric** field, select a metric from the new Dynamic Application to view its data in the right-hand pane. The widget is added to the tab.

**NOTE:** The metrics from the new Dynamic Application might take a minute or two to display in the list of metrics. Click **[Refresh]** in your browser if needed.

## Configuring Run Book Automations for an SL1 Agent

You can enable an SL1 agent to execute run book automation by customizing the automation and action policies in the following PowerPacks:

- "Linux SSH Automations" PowerPack version 103 or later
- "Windows PowerShell Automations" PowerPack version 103 or later

### Action Type

For your agent-based run book action, you will need to use one of the following action types:

- "Execute PowerShell Request " from the "Windows PowerShell Automations PowerPack" version 103 or later
- "Execute Shell Commands" from the "Linux SSH Automations" PowerPack version 103 or later

These action types (Registry > Run Book > Action Types) contain the code in the **Input Parameters Definition** pane that lets you need to run the actions on the agent:

Action Type Editor | Editing Action Type [111] Reset

Name:  Version:  State:  Organization:

Description:  Execution Environment:

Input Parameters Definition

```
{
  "name": "commands",
  "type": "string"
},
{
  "name": "request_key",
  "type": "string"
},
{
  "name": "credential_id",
  "type": "number"
}
}
```

Output Parameters Definition

```
{
  "name": "command_list_out",
  "type": "string"
}
}
```

Snippet

```
from silo_network_tools.client.powershell_client import PowerShellClient
from silo_network_tools.client.client_strategy import ClientStrategy
from silo_network_tools.event_logger import EventLogger
from silo_network_tools.tools.credential.rba_credential_manager import RBACredentialManager, CredentialType
from silo_network_tools.tools.credential.exception import NotFoundRBACredentialError
from silo_network_tools.client.exception import ExecuteCommandError

from silo_text_formatter.tools.enrichment_decorator import enrichment_snippet
from silo_text_formatter.enrichment_decorators.memory_calculator import calculate_memory
from silo_text_formatter.commands_formatter import CommandsFormatter
import json
import sys

POSITION_TO_TRUNCATE = 1
# Dynamic Application:
# Microsoft: Windows Server OS Configuration
# Microsoft: Windows Server Configuration Cache
# Microsoft: Windows Server Performance Cache
DA_GUIDS = ["4431471AC568815F17A55B5BAA5D6F80",
            "4CFAE849555607EFB79056784BAD9F13",
            "D69A8FDDDCB39CABCF8A5F92073E8DA7"]
```

## Configuring a Run Book Automation for an Agent Device

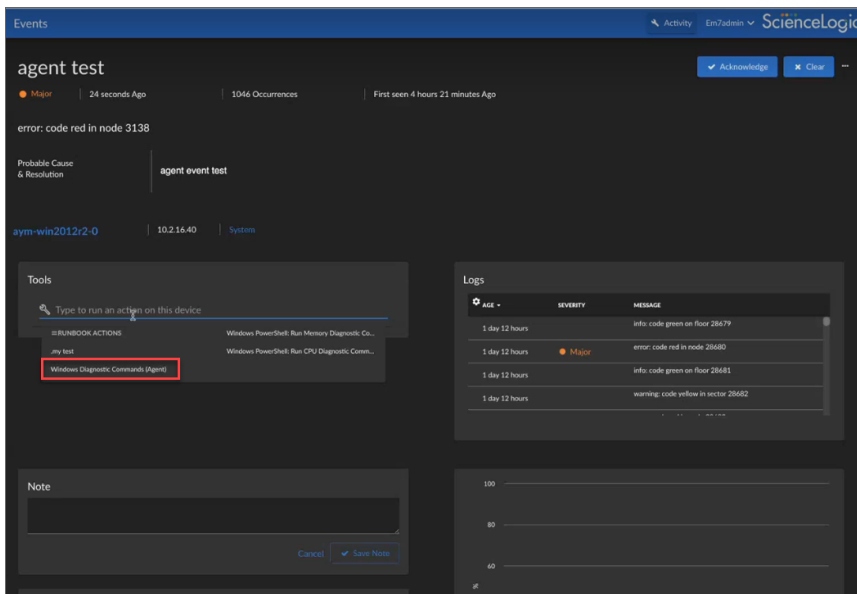
You configure a run book automation with an SL1 agent in the same way you configure an automation on a non-agent device. For more information about creating run book action and automation policies, see the **Run Book Automation** manual.

The following steps provide an example of configuring a User-initiated Automation with an SL1 agent:

1. Make sure you have the latest version of the "Linux SSH Automations" or "Windows PowerShell Automations" PowerPacks installed on your SL1 system.
2. Go to the **Automation Policy Manager** page (Registry > Run Book > Automation).
3. Click the **[Create]** button to create a new run book automation policy or click the edit icon (🔧) to edit an existing PowerShell or Linux SSH automation policy.
4. In the **Aligned Devices** field, make sure that the SL1 agent device is selected.
5. In the **Available Events** pane, select the SL1 event that will trigger this automation and click the right-arrow button.
6. In the **Available Actions** pane, select an action policy by highlighting it and clicking the right arrow-button.
7. Complete the other fields as needed, and then click **[Save]**.



8. Go to the **Actions** page (Registry > Run Book > Actions).
9. Click the **[Create]** button to create a new run book action or click the edit icon (🔧) to edit an existing PowerShell or Linux SSH action.
10. For the **Action Type** field, select *Execute PowerShell Request* or *Execute Shell Commands*.
11. Update the commands in the **Input Parameters** section as needed.
12. Click **[Save]** and close the **Policy Editor** window.
13. When an event occurs on the agent device, you can open the **Event Investigator** page for that event and run this user-initiated run book action by selecting the action from the **[Tools]** pane:



## Configuring an SL1 Agent on the Device Manager Page

**NOTE:** To configure agent settings for an agent, you must first add the **SL Agent** column to the **Device Manager** page. For more information about adding the **SL Agent** column, see [Adding the SL Agent Column to the Device Manager Page](#).

**NOTE:** If you are using the SL1 Extended Architecture, you can access these settings from the **[Settings]** tab of the **Device Investigator** page for the device on which the agent is installed. For more information, see [Configuring Agent Settings from the Device Investigator Page](#).

You can control how an agent runs on a device by configuring the following agent settings on the **Device Manager** page (Devices > Device Manager or Registry > Devices > Device Manager):

- **Disk Space.** Controls the amount of disk space that the agent can use to store data. If an agent loses connectivity to SL1, this disk space will be used to store collected data until the connection to SL1 is restored.
- **Data Directory.** Defines the directory in which the agent will store temporary data.
- **Excludes.** Defines the list of processes and directories to explicitly exclude from monitoring by the agent.
- **Includes.** Defines the list of processes and directories that must be explicitly monitored by the agent. Use the **Includes** field to ensure that specific processes are monitored.

**NOTE:** If a process or directory is included in both the **Excludes** field and the **Includes** field, that process or directory will be monitored by the agent.

## Adding the "SL Agent" Column to the Device Manager Page

The **SL Agent** column allows you to access the configuration settings for the agent on a device. For more information about agent configuration settings, see [Configuring Agent Settings on a Device](#). By default, the **SL Agent** column is not displayed in the **Device Manager** page.

To add the **SL Agent** column to the **Device Manager** page:

1. Go to the **Device Manager** page (Devices > Device Manager or Devices > Classic Devices, or Registry > Devices > Device Manager in the classic SL1 user interface in the classic user interface).
2. Click **[Actions]**, and then select *Device Manager Preferences*. The **Edit Device Manager Preferences** modal page appears.
3. In the **Device Manager Columns** field, control-click *Agent*.
4. Click **[Save]**.

## Configuring Agent Settings on a Device

To configure agent settings, you must first add the **SL Agent** column to the **Device Manager** page. For more information about adding the **SL Agent** column, see [Adding the SL Agent Column to the Device Manager Page](#).

To configure agent settings on a device:

1. Go to the **Device Manager** page (Devices > Device Manager or Devices > Classic Devices, or Registry > Devices > Device Manager in the classic SL1 user interface).
2. Find the device for which you want to edit agent settings. In the **SL Agent** column, click the gear icon (⚙️) for the device. The **Agent Configuration** page appears.
3. Supply values in the following fields:
  - **Disk Space.** Enter the amount of disk space that the agent can use to store data. If the agent loses connectivity to SL1, this disk space will be used to store collected data until the connection to SL1 is restored.
  - **Data Directory.** Enter the directory in which the agent will store temporary data.

- **Excludes.** Enter a semi-colon delimited list of processes and directories to explicitly exclude from monitoring by the agent.
- **Includes.** Enter a semi-colon delimited list of processes and directories that must be monitored by the agent. Use the **Includes** field to ensure that specific processes are monitored.

**NOTE:** If a process or directory is included in both the **Excludes** field and the **Includes** field, that process or directory will be monitored by the agent.

4. Click **[Save]**.



## Monitoring Logs Using the SL1 Agent

---

### Overview

This chapter describes how to use the SL1 Agent to monitor logs with Log File Monitoring policies. SL1 supports multiple methods for ingesting log data, which you can use to generate events.

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:

<i>What is a Log File Monitoring Policy?</i> .....	77
<i>Viewing the List of Log File Monitoring Policies</i> .....	77
<i>Creating a Log File Monitoring Policy</i> .....	78
<i>Aligning a Log File Monitoring Policy to Devices</i> .....	79
<i>Editing a Log File Monitoring Policy</i> .....	81
<i>Deleting Log File Monitoring Policies</i> .....	81
<i>Viewing the List of Log File Monitoring Policies and Aligned Devices</i> .....	82
<i>Creating an Event Policy for Agent Logs</i> .....	83
<i>Creating an Event Policy for Agent Logs in the Classic User Interface</i> .....	87

---

## What is a Log File Monitoring Policy?

A Log File Monitoring policy specifies:

- A file or Windows log on the host device that an agent will monitor
- The logs from the file or Windows log that an agent will send to SL1

You can create, edit, and delete Log File Monitoring policies from the **Log File Monitoring Policies** page (System > Manage > Log File Monitoring Policies). After creating a Log File Monitoring policy, you must align the policy to one or more devices either from the **Log File Monitoring** page or by using a Device Template.

You can view logs collected by the SL1 agent on the **Logs** panel on the **Device Investigator** page for the device on which the agent is installed. The same logs also appear on the **[Logs]** tab in the **Device Properties** and **Device Summary** pages for that device. You can define event policies that specify how logs collected by an agent will trigger events.

Log File Monitoring policies can be included in a PowerPack. For information about including a Log File Monitoring Policy in a PowerPack, see the **PowerPacks** manual.

**NOTE:** When running the agent as a dedicated user on Windows or Linux systems, the user requires Read access to any directories being monitored. To monitor system events on Windows, the user must be in the Event Log Readers group. See [Installing a Windows Agent](#) for more information.

---

## Viewing the List of Log File Monitoring Policies

The **Log File Monitoring Policies** page displays a list of all Log File Monitoring policies. From this page, you can also create, edit, and delete Log File Monitoring policies.

**TIP:** To sort the list of Log File Monitoring policies, click on a column heading. The list will be sorted by the column value, in ascending order. To sort by descending order, click the column heading again. The **Last Edited** column sorts by descending order on the first click; to sort by ascending order, click the column heading again.

For each Log File Monitoring Policy, the page displays:

- **Name.** Name of the Log File Monitoring policy.
- **Policy ID.** Unique numeric ID, automatically assigned by SL1 to each Log File Monitoring policy.
- **Source Type.** The source of the logs on the monitored device. Possible values are:
  - *File.* The agent will monitor a file on the file system of the device(s).
  - *Event Log.* The agent will monitor the Windows log on the device(s).

- **Source.** The full path of the log file or the name of the log that the agent will monitor.

**TIP:** For Linux or Unix operating systems, use "/" in the file paths. For Windows you can use "\" in the file paths, but a double slash "\\" will escape the "\" to ensure the file path ends up with a legitimate slash in it.

- **Filter.** The regular expression that the agent uses to determine whether a log message is sent to SL1.
- **Subscribers.** The number of devices with which the policy is aligned.
- **Edited By.** SL1 user who created or last edited the Log File Monitoring policy.
- **Last Edited.** Date and time the Log File Monitoring policy was created or last edited.

---

## Creating a Log File Monitoring Policy

To create a Log File Monitoring policy:

1. Go to the **Log File Monitoring Policies** page (System > Manage > Log File Monitoring Policies).
2. Click **[Create]**. The **Log Monitoring Policy** modal appears.
3. Supply values in the following fields:
  - **Name.** Enter a name for the policy.
  - **Type.** Select the source of the logs on the monitored device. Choices are:
    - *File.* The agent will monitor a file on the file system of the device(s).
    - *Event Log.* The agent will monitor the Windows log on the device(s).
  - **File Path.** If you selected *File* in the **Type** field, this field is displayed. Enter the full path of the file to monitor. You can use a \* to match multiple files, such as `/var/log/em7/*.log`.
  - **Source.** If you selected *Event Log* in the **Type** field, this field is displayed. Select the Windows log to monitor. Choices are:
    - *application*
    - *system*
    - *security*
    - *other*
  - **Description.** If you selected *other* in the **Source** field, this field is displayed. Type the name of the event log source type.
  - **Limit.** The maximum log messages the agent sends to SL1 per minute. If the number of matching logs exceeds this value, the agent will stop sending logs to the platform for the remainder of the minute. The limit resets at the beginning of the next minute.

For example, suppose you set this field to 10,000. Suppose the agent monitors a device that has 30,000 log messages. The agent will retrieve 10,000 logs and then wait until the beginning of the next minute. The agent will then retrieve the next 10,000 logs and then wait until the beginning of the next minute. The agent will continue to retrieve 10,000 logs per minute until it has retrieved all the logs from the device.

- **Filter.** Specify a regular expression that will be used to evaluate the log messages in the specified file or Windows log. If a log message matches this regular expression, the agent will send that log message to SL1. If a log message does not match this regular expression, the agent will not send that log message to SL1.

**NOTE:** For Windows event logs, the SL1 Agent adds the **Event ID** to the value in the *Message* portion of the Windows log before applying the value in the **Filter** field. The agent does not apply the value in the **Filter** field to the *Instance ID* or any other property of a Windows event log entry.

**TIP:** Avoid adding a leading "." in a filter, such as ".ERROR", as that character might increase the time it takes the agent to execute the filter, and on busy SL1 systems, that character in the filter can negatively impact the CPU.

4. Click **[Save]**.
5. Before you can use this Log File Monitoring policy, you will need to align the policy with one or more devices. For more information, see [Aligning a Log File Monitoring Policy to Devices](#).

---

## Aligning a Log File Monitoring Policy to Devices

You can align Log File Monitoring policies to devices either from the **Log File Monitoring** page or by using a Device Template.

This section describes how to align a Log File Monitoring policy from the **Log File Monitoring** page. It also describes how to use a one-off Device Template to align a Log File Monitoring policy. For more information on Device Templates, including the other methods you can use to create, save, and apply Device Templates, see the **Device Groups and Device Templates** manual.

To align Log File Monitoring policies to one or more devices *from the **Log File Monitoring** page*:

1. Go to the **Log File Monitoring** page (Registry > Monitors > Logs).
2. Click **[Create]**. The Log File Monitor modal appears.
3. In the Log File Monitor modal, supply values in the following fields:
  - **Device.** Select a device to align with the Log File Monitoring policy.
  - **Log Policy.** Select the Log File Monitoring policy to align with the selected device. Only policies that are appropriate for the selected device will appear. For example, if you chose a Linux device in the **Device** field, the **Log Policy** field will not show policies of the *Event Log* type.

4. If desired, click the names of the following fields to enable and edit them. These fields allow you to override settings of the policy you selected in the **Log Policy** field for the device selected in the **Device** field:
  - **File Path**. Enter the full file path or the file name to monitor. This field appears only if the type of the policy is *File*.
  - **Limit**. The maximum log messages the agent sends to SL1 per minute. If the number of matching logs exceeds this value, the agent will stop sending logs to the platform for the remainder of the minute. The limit resets at the beginning of the next minute. For example, suppose you set this field to 10,000. Suppose the agent monitors a device that has 30,000 log messages. The agent will retrieve 10,000 logs and then wait until the beginning of the next minute. The agent will then retrieve the next 10,000 logs and then wait until the beginning of the next minute. The agent will continue to retrieve 10,000 logs per minute until it has retrieved all the logs from the device.
  - **File**. Specify a regular expression that will be used to evaluate the log messages in the specified file or Windows log. If and only if a log message matches this regular expression, the agent will send the log message to SL1.
5. Click **[Save]**.

To align Log File Monitoring policies to one or more devices using a *Device Template*:

1. Go to the **Device Manager** page (Devices > Device Manager).
2. Select the checkboxes for the devices with which you want to align Log File Monitoring policies.
3. In the **Select Action** drop-down list, select *MODIFY by Template*.
4. Click **[Go]**. The **Device Template Editor** modal appears.
5. Click the **[Logs]** tab.
6. Click the Add New Log Policy Sub-Template icon (+).
7. Supply values in the following fields:
  - **Align Log Monitoring Policy With**. Select the devices to which the Log File Monitoring policy will be applied.
  - **Log Monitoring Policy**. Select the Log File Monitoring policy you want to align with the selected devices.
8. Optionally, you can override one or more settings from the Log File Monitoring policy specifically for the selected devices. To do this, click the field label for each setting you want to override to enable the fields and supply a value in those fields. For a description of each field, see the [Creating a Log File Monitoring Policy](#) section.
9. Repeat steps 6 and 7 for each Log File Monitoring policy you want to align with the devices you selected in step 2.
10. If you want to save this Device Template for future use, select the **Save When Applied & Confirmed** checkbox and enter a name for the Device Template in the **Template Name** field.
11. Click **[Apply]**. The **Setting Confirmation** page is displayed.
12. Click **[Confirm]**. The aligned Log File Monitoring policy will appear on the **Log File Monitoring** page (Registry > Monitors > Logs).



## Unaligning Log File Monitoring Policies from Devices

To delete Log File Monitoring Policies, you must first unalign the policy from any devices. You can unalign a Log File Monitoring policy by from the **Log File Monitoring** page.

To unalign devices from a Log File Monitoring policy:

1. Go to the **Log File Monitoring** page (Registry > Monitors > Logs)
2. Select the checkboxes for the devices from which the policy must be unaligned.
3. In the **Select Action** drop-down menu, choose *Delete Log File Monitors*.


**NOTE:** This action does not delete the Log File Monitoring *policy*.

4. Click **[Go]** to unalign the Log File Monitoring policy from the devices.

---

## Editing a Log File Monitoring Policy

To edit a Log File Monitoring policy:

1. Go to the Log File Monitoring Policies page (System > Manage > Log File Monitoring Policies).
2. Click the wrench icon () for the Log File Monitoring Policy you want to edit. The **Log Monitoring Policy** modal appears.
3. Edit the value in one or more fields. For a description of each field, see the [Creating a Log File Monitoring Policy](#) section.
4. Click **[Save]**.

---

## Deleting Log File Monitoring Policies

**NOTE:** Before you delete a Log File Monitoring Policy, you must unalign that policy from all devices. [See Unaligning Log File Monitoring Policies](#) for more information.

To delete one or more Log File Monitoring policies:

1. Go to the **Log File Monitoring Policies** page (System > Manage > Log File Monitoring Policies).
2. Select the checkboxes for the **Log File Monitoring Policies** you want to delete.
3. In the **Select Action** drop-down list, select *DELETE Log File Monitoring Policies*.
4. Click **[Go]**.

---

## Viewing the List of Log File Monitoring Policies and Aligned Devices

The **Log File Monitoring** page (Registry > Monitors > Logs) displays a list of existing relationships between devices and Log File Monitoring policies. From the **Log File Monitoring** page, you can also align and unalign devices and Log File Monitoring policies.

For each aligned Log File Monitoring policy and device, the page displays:

- **Name**. The name of the Log File Monitoring policy.
- **Device Name**. The name of the device aligned to the Log File Monitoring policy.
- **ID**. The unique numeric ID of the Log File Monitoring policy. The ID is automatically assigned by SL1.
- **Source Type**. The source of the logs in the monitored device. The possible values are:
  - *File*. The agent monitors a file on the file system of the device. Usually, this is used to monitor Linux log files.
  - *Event Log*. The agent monitors to Windows log on the device.
- **Source**. The full path of the log file or the name of the Windows log that the agent monitors.
- **Filter**. The regular expression the agent uses to determine if a log should be sent to SL1.
- **Limit**. The maximum log messages the agent sends to SL1 per minute. If the number of matching logs exceeds this value, the agent will stop sending logs to the platform for the remainder of the minute. The limit resets at the beginning of the next minute. For example, suppose you set this field to *10,000*. Suppose the agent monitors a device that has 30,000 log messages. The agent will retrieve 10,000 logs and then wait until the beginning of the next minute. The agent will then retrieve the next 10,000 logs and then wait until the beginning of the next minute. The agent will continue to retrieve 10,000 logs per minute until it has retrieved all the logs from the device.
- **Edited By**. The user who created or last edited the alignment between the device and Log File Monitoring policy.
- **Last Edited**. The date and time the alignment between the device and Log File Monitoring policy was created or last edited.

## Filtering the List of Log File Monitoring Policies and Aligned Devices

You can filter the list of Log File Monitoring policies and aligned devices on the **Log File Monitoring** page using the search fields at the top of each column. When you type in each search field, the list of results on the page is automatically updated to match the text, including partial matches.

You can use special characters in each search field to filter. For more information about filtering using special characters, see the [Filtering the List of Log File Monitoring Policies](#) section.

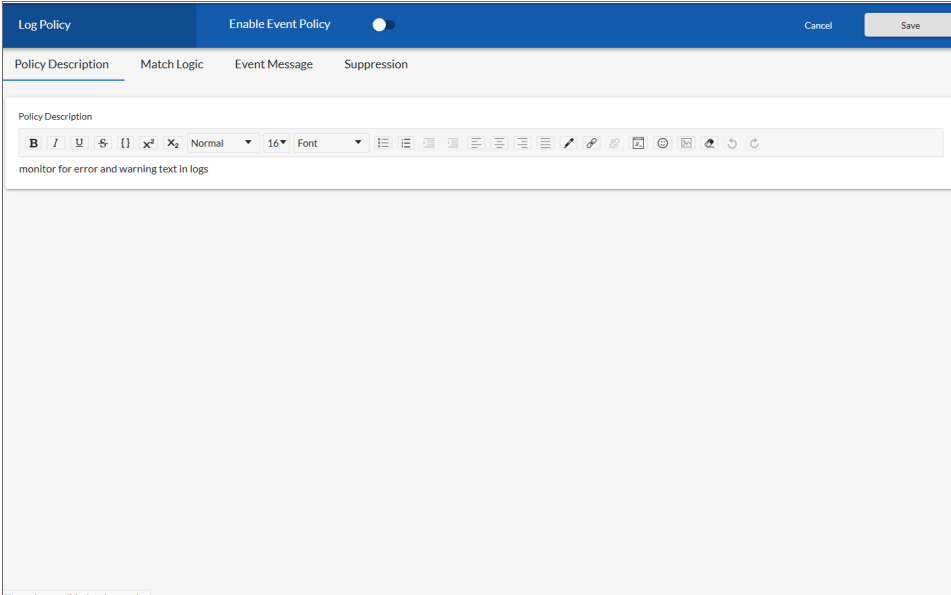
## Creating an Event Policy for Agent Logs

To trigger events based on log messages collected by the agent, you must create an event policy on the **Event Policies** page, and then associate that event policy with a Log File Monitoring policy.

**NOTE:** If you are using the classic user interface for SL1, see [Creating an Event Policy for Agent Logs in the Classic User Interface](#).

To create an event policy that triggers based on log data collected by the agent:

1. Go to **Event Policies** page (Events > Event Policies).
2. On the **Event Policies** page, click the **[Create Event Policy]** button. The **Event Policy Editor** page appears:



3. On the **Event Policy Editor** page and set of tabs, you can define a new event. The **Event Policy Editor** page contains the following fields and tabs:
  - **Policy Name**. At the top left of the **[Policy Description]** tab, type a name for the event policy in this field.
  - **Enable Event Policy**. This toggle allows you to enable and disable the event policy.
  - **Policy Description**. In this field, type a description of the event policy.

4. Click the **[Match Logic]** tab to define pattern-matching for the event:

The screenshot shows the 'Log Policy' configuration window with the 'Match Logic' tab selected. The 'Event Source' is set to 'ScienceLogic Agent' and the 'Log Policy' is 'event log errors'. The 'Match Logic' section includes a 'String' dropdown, 'Match String' and 'Second Match String (Optional)' text boxes, and two toggle switches for expiration and multiple triggers. The 'MATCHING' section has a 'Detection Weight' of 0, a checked 'Multi Match' checkbox, and an unchecked 'Message Match' checkbox.

5. Complete the following fields:

- **Event Source.** Select *ScienceLogic Agent* as the source for the event. The fields below this field will change based on your selection.
- **Log Policy.** Select the Log File Monitoring Policy the agent will use to collect the log message.

6. After selecting and defining your **Event Source**, enter values in the fields on the right side of the **Match Logic** tab specify the text that must appear in the log message for the event policy to trigger:

- **String/Regular Expression.** Use this dropdown to select *String* or *Regular Expression*.
- **Match String.** A string used to correlate the event with a log message. Can be up to 512 characters in length. To match this event policy, the text of a log message or alert must match the value you enter in this field. Can be any combination of alpha-numeric and multi-byte characters. SL1's expression matching is case-sensitive. This field is required for events generated with a source of Syslog, API, and Email.
- **Second Match String (Optional).** A secondary string used to match against the originating log message. Can be up to 512 characters in length. Can be any combination of alpha-numeric and multi-byte characters. To match this event policy, the text of a log message or alert must match the value you enter in this field and the value you entered in the **Match String** field. This field is optional.
- **Allow event to expire if it doesn't reoccur within a time frame.** If toggled on, enter the time in which an active event will be cleared automatically if there is no reoccurrence of the event in the fields that appear. You can enter time in minutes or hours.
- **Require multiple triggers within a time frame.** If toggled on, enter the number of events and the time in which an event requires multiple triggers to occur in the fields that appear. You can enter time in minutes or hours.

- **Detection Weight.** If two event definitions are very similar, the weight field specifies the order in which SL1 should match messages against the similar event definitions. The event definition with the lowest weight will be matched first. This field is most useful for events that use expression matching. Options range from 0 (first) - 20 (last).
- **Multi Match.** By default, SL1 will match a log message or alert to only one event policy. If a log message or alert matches multiple event policies, SL1 will use the **Detection Weight** setting to determine which event policy the log message or alert will match. If you select the **Multi Match** checkbox in all event policies that can match the same log message or alert, SL1 will generate an event for every event policy that matches that single log message or alert.
- **Message Match.** If SL1 has generated an event and then a second log message or alert matches the same event policy for the same entity, SL1 will not generate a second event, but will increase the **count** value for the original event on the **Events** page and in the **Viewing Events** page. By default, this behavior occurs regardless of whether the two log messages or alerts contain the same message. If you select the **Message Match** checkbox, this behavior will occur only if the log messages or alerts contain the same message.

7. Click the **[Event Message]** tab to configure the event:

The screenshot shows the 'Enable Event Policy' configuration window. The 'Event Message' tab is active. The 'Event Message' field contains '%M'. The 'Event Severity' is set to 'Major'. There are four toggle options: 'Use Modifier' (unchecked), 'Correlate this event with external system' (unchecked), 'Categorize event for external system' (unchecked), and 'Extract sub-entity using a regular expression' (unchecked). On the right, the 'Autoclear' toggle is checked. Below it, there is a section for 'TOPOLOGY MASKING' with two unchecked options: 'Mask events on child devices' and 'Maskable under a parent device's event'. There are buttons for 'Add Event Policy' and 'Add Category'.

8. Complete the following fields:

- **Event Message.** The message that appears in the **Event Console** page or the **Viewing Events** page when this event occurs. This field defaults to "%M" for new event policies upon creation. The message can be any combination of alphanumeric and multi-byte characters. Variables include the characters "%" (percent) and "|" (bar). You can also use regular expressions and variables that represent text from the original log message to create the **Event Message**.

To include regular expressions in the Event Message, surround the regular expression with **%R** and **%/R**. For example, **%RFilename: .\*? %/R** would search for the first instance of the string "Filename: " (Filename-colon-space) followed by any number of any characters up to the line break. The **%R** indicates the beginning of a regular expression. The **%/R** indicates the end of a regular expression.

SL1 uses the regular expression to search the log message and use the matching text in the event message. For details on the regular expression syntax allowed by SL1, see <http://www.python.org/doc/howto/>.

You can also use the following variables in the **Event Message** field:

- **%I** ("eye"). This variable contains the value that matches the **Identifier Pattern** field in the **[Advanced]** tab.
  - **%M**. The full text of the log message that triggered the event will be displayed in **Event Message** field.
  - **%T**. Threshold value from the log file will be displayed in **Event Message** field.
- **Event Severity.** Defines the severity of the event. Choices are:
    - **Healthy.** Healthy events indicate that a device or condition has returned to a healthy state. Frequently, a healthy event is generated after a problem has been fixed.
    - **Notice.** Notice events indicate a condition that does not affect service but about which users should be aware.
    - **Minor.** Minor events indicate a condition that does not currently impair service, but the condition needs to be corrected before it becomes more severe.
    - **Major.** Major events indicate a condition that impacts service and requires immediate investigation.
    - **Critical.** Critical events indicate a condition that can seriously impair or curtail service and requires immediate attention (i.e., service or system outages).
  - **Use Modifier.** If selected, when the event is triggered, SL1 will check to see if the interface associated with this event has a custom severity modifier. If so, the event will appear in the **Event Console** with that custom severity modifier applied to the severity in the **Event Severity** field. For example, if an interface with an **Event Severity Adjust** setting of *Sev -1* triggers an event with an **Event Severity** of *Major* and that event has the **Use Modifier** checkbox selected, the event will appear in the **Event Console** with a severity of *Minor*.

9. Optionally, supply values in the other fields on this page. For more information on the remaining fields, as well as the **[Suppressions]** tab, see the *Defining and Editing Event Policies* chapter in the **Events** manual.
10. Click **[Save]**.

---

## Creating an Event Policy for Agent Logs in the Classic User Interface

To trigger events based on log messages collected by the agent in the classic user interface for SL1, you must create an event policy that is associated with a Log File Monitoring policy.

To create an event policy in the classic user interface based on log data collected by the agent:

1. Go to **Event Policy Manager** page (Registry > Events > Event Manager in the classic user interface).
2. In the **Event Policy Manager** page, click **[Create]**. The **Event Policy Editor** page appears.
3. In the **Event Policy Editor** page and set of tabs, you can define a new event. The **Event Policy Editor** page contains three tabs:
  - **Policy**. Define basic parameters for the event.
  - **Advanced**. Define pattern-matching for the event and also define event roll-ups and suppressions.
  - **Suppressions**. Suppress the event on selected devices. When you suppress an event, you are specifying that, in the future, if this event occurs again on a specific device, the event will not appear in the **Event Console** page or the **Viewing Events** page for the device.
4. Supply values in the following fields:
  - **Event Source**. Select *ScienceLogic Agent*.
  - **Policy Name**. The name of the event. Can be any combination of alphanumeric characters, up to 48 characters in length.
  - **Operational State**. Specifies whether event is to be operational or not. Choices are *Enabled* or *Disabled*.
  - **Event Message**. The message that appears in the **Event Console** page or the **Viewing Events** page when this event occurs. Can be any combination of alphanumeric characters.
    - You can use regular expressions that represent text from the original log message to create the **Event Message**:
      - **%R**. Indicates a regular expression. Surround the regular expression with **%R** and **%/R**. For example, **%Rfilename: .\*? %/R** would search for the first instance of the string "filename: " followed by any number of any characters up to the line break. For details on the regular expression syntax allowed by SL1, see <http://www.python.org/doc/howto/>.
      - **%l** ("eye"). This variable contains the value that matches the **Identifier Pattern** field in the **[Advanced]** tab.
      - **%M**. The full text of the log message that triggered the event will be displayed in **Event**

**Message** field.

- **%T**. Threshold value from the log file will be displayed in **Event Message** field.
  - **Event Severity**. Defines the severity of the event. Choices are:
    - **Healthy**. Healthy Events indicate that a device or condition has returned to a healthy state. Frequently, a healthy event is generated after a problem has been fixed.
    - **Notice**. Notice Events indicate a condition that does not affect service but about which users should be aware.
    - **Minor**. Minor Events indicate a condition that does not currently impair service, but the condition needs to be corrected before it becomes more severe.
    - **Major**. Major Events indicate a condition that is service impacting and requires immediate investigation.
    - **Critical**. Critical Events indicate a condition that can seriously impair or curtail service and require immediate attention (i.e. service or system outages).
  - **Use Modifier**. If selected, when the event is triggered, SL1 will check to see if the interface associated with this event has a custom severity modifier. If so, the event will appear in the **Event Console** with that custom severity modifier applied to the severity in the **Event Severity** field. For example, if an interface with an **Event Severity Adjust** setting of *Sev -1* triggers an event with an **Event Severity** of *Major* and that event has the **Use Modifier** checkbox selected, the event will appear in the **Event Console** with a severity of *Minor*.
  - **Policy Description**. Text that explains what the event means and what possible causes are.
5. Select the **[Advanced]** tab.
  6. In the **Log Policy** field, select the Log File Monitoring policy that the agent will use to collect the log message.
  7. Enter values in the following fields to specify specific text that must appear in the log message for the event policy to trigger:
    - **First Match String**. A string used to match against the originating log message. To match this event policy, the text of a log message must match the value you enter in this field. Can be any combination of alphanumeric characters. Expression matching in SL1 is case-sensitive.
    - **Second Match String**. A secondary string used to match against the originating log message. To match this event policy, the text of a log message must match the value you enter in this field and the value you entered in the **First Match String** field. This field is optional.

**NOTE:** The **Match Logic** field specifies whether SL1 should process **First Match String** and **Second Match String** as simple text matches or as regular expressions.



8. Optionally, supply values in the other fields on this page. For more information on the remaining fields, as well as the **[Suppressions]** tab, see the **Events** manual.
9. Click **[Save]**.

---

# Chapter

# 5

## SL1 Agent Troubleshooting

---



### Overview

This chapter contains troubleshooting processes that you can use to address issues with an SL1 agent.

**TIP:** On a Windows agent, you can run the following diagnostic command to generate a \*.tar.gz file that contains useful information for troubleshooting issues: `...\SiloAgent.exe --diag`

**TIP:** On a Linux agent, use `--diag` for the diagnostic option.

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 (.

To troubleshoot potential issues with SL1 agents, perform the following procedures, in the following order:

<i>Was the Agent Download Successful?</i> .....	91
<i>Is the Windows installation or upgrade failing?</i> .....	91
<i>Is the Agent Process Running?</i> .....	91
<i>Is the Agent Configuration File Valid?</i> .....	92
<i>Has SL1 Discovery Completed?</i> .....	93
<i>Is the Agent Able to Upload Data?</i> .....	93
<i>Is SL1 Receiving Agent Data?</i> .....	95

<i>Is the Agent Not Reporting Vital Data and Metrics?</i> .....	95
<i>Can SL1 Process Agent Data?</i> .....	95
<i>Is the Number of Processes Inconsistent with Other Applications?</i> .....	96
<i>Does MySQL on the Data Collector Have a Connection Issue?</i> .....	96
<i>Is the Agent Unable to Connect to the Streamer Endpoint?</i> .....	97
<i>Troubleshooting Examples</i> .....	99
<i>Additional Troubleshooting Situations and Best Practices</i> .....	101
<i>Agent Communication with SL1</i> .....	102

---

## Was the Agent Download Successful?

If the Windows agent download failed with a "500 Internal Server Error", restart the **uwsgi** service on the SL1 system from which you are downloading the agent.

From an administrator command prompt in Windows, run the following command on the SL1 system:

```
systemctl restart uwsgi
```

---

## Is the Windows installation or upgrade failing?

See [Installing a Windows Agent](#) for more information.

---

## Is the Agent Process Running?

**NOTE:** When running the agent as a dedicated user on Windows, the "Log on as a service" user right is required for the service to start.

As a first step, always locate the following logs from the Message Collectors when troubleshooting:

- `/var/log/streamer_prime/streamer_prime.log`
- `/var/log/uwsgi/streamer_prime.log`

To determine if the agent process is running:

1. Check the Windows Task Manager or run the "tasklist" or "top" command.
2. Look for **SiloAgent.exe** (Windows) or **scilogd** (Linux):
  - Windows: If **SiloAgent.exe** is not running, check the "Application" event log for events with `source=SiloAgent`.
  - Linux: If **scilogd** is not running, check `/var/log/messages` or `/var/log/syslog` for relevant log messages.

If you are using the SL1 Extended Architecture, determine if the agent was deleted from the **[Agents]** tab instead of uninstalling the agent.

**If the agent was deleted in the SL1 user interface**, SL1 shuts down the agent instead of uninstalling the agent. You should re-install the agent that you deleted in SL1.

To re-install the agent that was shut down:

1. Uninstall the agent that you shut down.
2. Delete that agent's configuration from one of the following locations:
  - Windows: **C:\Program Files\ScienceLogic\SiloAgent\conf\scilog.conf**
  - Linux: **/etc/scilog/scilog.conf**
3. Install a new agent.

**If the agent was not deleted**, then the issue could be with the agent. You should generate diagnostics information to share with your ScienceLogic contact.

To generate diagnostics information for an agent:

1. From an administrator command prompt, run one of the following commands:
  - Windows: `C:\Program Files\ScienceLogic\SiloAgent\bin\SiloAgent.exe -diag`
  - Linux: `/usr/bin/scilogd --diag`
2. Share the contents of the newly created diagnostic file in the current directory with your ScienceLogic contact. Depending on your operating system, the file name is:
  - Windows: **scilog-*<current date>*.diag.tgz**
  - Linux: **sl-diag.tar.gz**

---

## Is the Agent Configuration File Valid?

1. Check the agent configuration file in one of the following locations:
  - Windows: **C:\Program Files\ScienceLogic\SiloAgent\conf\scilog.conf**
  - Linux: **/etc/scilog/scilog.conf**
2. Check the configuration item **CollectorID**:
  - If there is *no* **CollectorID** tag, then the agent has not been able to reach the stream or message collector.
  - The **CollectorID** should be a GUID similar to "4179b06ef502129c3023a0f8d58f3c37".
3. Check the configuration item **URLfront**, which is where the agent attempts to get the configuration file:

- Determine if you can ping the **URLfront**.
- If you are using the SL1 Distributed Architecture, **URLfront** should be the URL of the Message Collector.
- If you are using the SL1 Extended Architecture, **URLfront** should be the URL of the Streamer container, such as `pod9-streamer0`.
- If the URL for **URLfront** is not correct, then re-install the agent.
- If the URL for **URLfront** is correct, then determine if you can ping the host portion of **URLfront**. If you cannot ping, then there are customer firewall or NAT issues.

---

## Has SL1 Discovery Completed?

To check if SL1 discovery finished:

1. Check the agent record in the SL1 Distributed Architecture by using SSH to communicate with the Message Collector and running the following commands:

```
redis-cli -p 6380
```

```
keys *
```

```
hgetall agent_<GUID>
```

where `<GUID>` is the specific value from **scilog.conf**.

2. Look for a field "did". The following line is the EM7-device-id.
3. If the EM7-device-id is not present, then discovery has not completed.
4. Alternately, you can look on the **Devices** page for the device to compare the shown device-id to the value in the agent record.

---

## Is the Agent Able to Upload Data?

### Check the Agent Upload Directory

Check the upload directory for the agent for directories and files in one of the following locations:

- Windows: **C:\Program Files\ScienceLogic\SiloAgent\data**
- Linux: **opt/scilog/data**

If there are many items, then the agent is unable to upload.

If the number of items is *decreasing*, the agent might have an issue. The agent is slowly catching up, but this situation indicates that a previous issue existed.

If the number of items continues to *increase* overall, check the configuration item URL:

- The URL is the location where the agent attempts to upload files.
- Determine if the host portion of the URL is reachable. If the host portion is reachable, the name of the oldest item indicates the approximate time of the issue.

**NOTE:** To prevent consuming the disk with backed-up data, the agent limits the size and count of items in the upload directory.

A procedural note regarding backed-up data:

For a new installation, the agent reaches out to the streamer for a configuration file. If the configuration file can't reach the streamer, the streamer goes into a slow poll mode, waiting for a good configuration file. In the meantime, the streamer does nothing else (it does not generate data or log files). As a result, even though it looks like there is no backup of data files, in reality, there are no data files.

After the Streamer container receives a valid configuration file:

- After a restart, the agent reaches out to the Streamer container for a new configuration file.
- If the agent can't reach the Streamer, the agent will still generate data files, because it has a valid configuration file from a previous run. In this situation, you will see data files backing up if the Streamer is unreachable.

**NOTE:** You have the ability to set the `RequireWebCert` to true in the configuration file the streamer sends to the agent so the validation process is successful.

In summary, if you have a valid configuration, you will get data files. If you do not have a backup, Streamer can be reached.

## Run the Agent in Debug Mode (Linux)

**NOTE:** You might need to preface the following commands with `sudo` if you are in root-privileged mode.

1. Stop the agent daemon by running the following command:

```
service scilogd stop
```

2. Start the agent from the command line:

```
scilogd -d 2>&1 | tee /tmp/scilogd.log
```

3. Let the agent run for about five minutes.
4. Press **Ctrl+C** and examine the output file.
5. Restart the agent by running the following command:

```
service scilogd start
```

---

## Is SL1 Receiving Agent Data?

If you are using the SL1 Distributed Architecture:

1. SSH into the Message Collector and run the following command:

```
sudo tail -n 100 /var/log/uwsgi/streamer_prime_uwsgi.log
```

2. Look for lines containing the hostname of the monitored device, such as the following:

```
10.2.16.40 - - [19/Apr/2018:17:04:55 +0000] "POST /SaveData.py/save_data HTTP/1.1" 200 59 "-" "Windows SiloAgent : aym-win2012r2-0"
```

3. If there are no matching lines, then the Streamer container is not getting data from that agent.

If you are using the SL1 Extended Architecture:

1. SSH into the Management Node and view the logs.
2. Look for lines containing the HOSTNAME of the monitored device.
3. If there are no matching lines, then the Streamer container is not getting data from that agent.

---

## Is the Agent Not Reporting Vital Data and Metrics?

If the agent is not reporting Windows vitals or PowerShell Dynamic Applications are not being collected:

1. Ensure the agent is running as dedicated user.
2. Ensure the user is enrolled in the "Performance Monitor Users" group to be able to collect the desired metrics

---

## Can SL1 Process Agent Data?

Check the Message Collector log files or SL1 Streamer log files:

1. If you are using the SL1 Distributed Architecture, locate the following files from the SL1 Message Collector and provide the files to your ScienceLogic contact:
  - **/var/log/uwsgi/streamer\_prime\_uwsgi.log**
  - **/var/log/streamer\_prime/streamer\_prime.log**
2. If you are using the SL1 Extended Architecture, check all logs for *ERR* or *Except*:

```
for p in $(kubectl get pods | cut -f 1 -d ' '); do echo $p; kubectl logs $p | grep -E "(ERR|Except)"; done
```

3. Search for "ERROR", "Exception", and "HARAKIRI".
4. Contact your ScienceLogic contact with any error messages you find in the log files.

**NOTE:** If you do not find any error messages, then the issue is most likely with the Dynamic Application that runs on the Data Collector.

---

## Is the Number of Processes Inconsistent with Other Applications?

- On Linux, many outputs from the `ps` command list the kernel threads (the processes listed in square brackets). Because the agent is not in the kernel, it will not list kernel threads.
- Be aware that the agent reports processes that are running as well as processes that started and may have stopped, while `top` or `ps` commands show processes that exist when they are executed.
- Check the agent configuration. Due to back-end space limitations, many configuration combinations can limit what data the agent sends. A combination of parameters to get all processes include the following:
  - **NIPD True.** The agent library can not get into all processes at times, often on install. Non-intercepted process discovery reports processes that are not intercepted via the library.
  - **SLPAggregation.** This parameter takes short-lived processes that exist for less than 80 seconds and rolls information about the processes into the information for their parents. As a result, the short-lived processes will not be seen.

---

## Does MySQL on the Data Collector Have a Connection Issue?

If MySQL on the Data Collector has more connections than the configured limit, the Data Collector will raise a connection issue and cause the agent connection failure. The following error message is an example of this situation:

```
Unhandled exception: Traceback (most recent call last):
```

```
File "/opt/em7/backend/process/proc_pda.py", line 223, in __init__
```

```
File "/opt/em7/backend/silo_collect/streamer/main.py", line 353, in main
```

```
File "/opt/em7/backend/silo_collect/shared.py", line 108, in run_collection
```

```
File "/opt/em7/backend/silo_collect/shared.py", line 156, in _run_collect_async
```



```
File "/usr/lib64/python2.7/multiprocessing/pool.py", line 554, in get
raise self._value ChildProcessException: Traceback (most recent call
last):
```

```
File "/opt/em7/backend/silo_common/misc.py", line 520, in new_f File
"/opt/em7/backend/silo_collect/streamer/main.py", line 225, in streamer_
collect
```

```
File "/opt/em7/backend/silo_common/database.py", line 701, in local_db
```

```
File "/opt/em7/backend/silo_common/database.py", line 92, in callFunc
```

```
File "/opt/em7/backend/silo_common/database.py", line 693, in get_dbc
MySQLError: Error attempting to connect to database with SSL enabled
False: (2013, 'Lost connection to MySQL server at \'reading initial
communication packet\', system error: 0 "Internal error/check (Not system
error)''')
```

To address this connection issue:

1. To determine the current connection limit, run the following command on the Data Collector:

```
silosql -e 'show variables like "max_connections";'
```

2. To view the high-water mark for concurrently active connections, run the following command on the Data Collector:

```
show status where 'Variable_name' = 'max_used_connections';
```

3. If the value is more than the limit value, try to dynamically but temporarily increase the limit. For example:

```
silosql -e 'set global max_connections=350'
```

---

## Is the Agent Unable to Connect to the Streamer Endpoint?

Some SL1 agents on Windows 2012 R2 might have issues connecting with the streamer endpoint if there is not a match with the default TLS ciphers.

To address this issue, add the ssl-ciphers configuration to the existing Kubernetes ConfigMap **nginx-configuration**.

To add ssl-ciphers in on-premises environments:

1. SSH to the Manager Node and enter to the Ansible shell.
2. Run the following commands to install vim:

```
sudo apt update
```

```
sudo apt install vim -y
```

3. Run the following command:

```
kubectl edit configmaps -n ingress-nginx nginx-nginx-ingress-  
controller
```

4. In the **nginx-configuration** file, add the missing ciphers after `apiVersion`:

**NOTE:** Line breaks were added to the following list of ciphers to make the code more readable.

```
data:  
  ssl-ciphers: ECDHE-ECDSA-AES128-GCM-SHA256;ECDHE-RSA-AES128-GCM-  
SHA256:  
ECDHE-ECDSA-AES256-GCM-SHA384;ECDHE-RSA-AES256-GCM-SHA384:  
ECDHE-ECDSA-CHACHA20-POLY1305;ECDHE-RSA-CHACHA20-POLY1305:  
DHE-RSA-AES128-GCM-SHA256;DHE-RSA-AES256-GCM-SHA384:  
DHE-RSA-CHACHA20-POLY1305;ECDHE-ECDSA-AES128-SHA256;ECDHE-RSA-AES128-  
SHA256:  
ECDHE-ECDSA-AES128-SHA;ECDHE-RSA-AES128-SHA;ECDHE-ECDSA-AES256-  
SHA384:  
ECDHE-RSA-AES256-SHA384;ECDHE-ECDSA-AES256-SHA;ECDHE-RSA-AES256-SHA:  
DHE-RSA-AES128-SHA256;DHE-RSA-AES256-SHA256;AES128-GCM-SHA256:  
AES256-GCM-SHA384;AES128-SHA256;AES256-SHA256;AES128-SHA;AES256-SHA:  
DES-CBC3-SHA  
  ssl-protocols: TLSv1 TLSv1.1 TLSv1.2 TLSv1.3  
  use-proxy-protocol: "false"
```

5. Exit with `:wq`. In a few minutes, the agent will be automatically recognized. If the agent is not recognized, try restarting the agent.

To add `ssl-ciphers` in AWS environments:

1. SSH Bastion/JH and enter to the Ansible shell.
2. Run the following command:

```
kubectl edit configmaps -n ingress-nginx nginx-nginx-ingress-controller
```

3. In the **nginx-configuration** file, add the missing ciphers after apiVersion:

```
data:
  ssl-ciphers: ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:
ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:
ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:
DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:
DHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:
ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-SHA384:
ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:
DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:AES128-GCM-SHA256:
AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:
DES-CBC3-SHA
  ssl-protocols: TLSv1 TLSv1.1 TLSv1.2 TLSv1.3
  use-proxy-protocol: "true"
```

4. Exit with `:wq`. In a few minutes, the agent will be automatically recognized. If the agent is not recognized, try restarting the agent.

---

## Troubleshooting Examples

### Example `/var/log/streamer_prime/streamer_prime.log` for successful discovery

```
2019-01-04T17:07:42.355291+00:00 amateen-em7 journal: SCILO_
SP:6954|logger:log_info:132|INFO|Agent config request received with init
flag set to True. Generated Temp AID: 2ae22a6b4489457abb14373cd3816076.
Request: <WSGIRequest: GET '/api/collector/config/?collector_
key=aEf34$aq3TGSDdf&tenant_id=0&host_name=aym-win2012r2-
1&init=&os=windows&collector_id=0'>
```

```
2019-01-04T17:07:42.619082+00:00 amateen-em7 journal: SCILO_
SP:6954|logger:log_info:132|INFO|Calling Agent version with: <QueryDict:
{'collector_id': ['2ae22a6b4489457abb14373cd3816076'], 'type':
['windows_64'], 'tenant_id': ['0'], 'host_name': ['aym-win2012r2-1'],
'collector_key': ['aEf34$aq3TGSDdf'], 'version': ['115']}>
```

```
2019-01-04T17:07:43.028457+00:00 amateen-em7 journal: SCILO_
SP:16717|logger:log_warning:127|WARNING|System file received from aym-
win2012r2-1
```

```
2019-01-04T17:07:43.032897+00:00 amateen-em7 journal: SCILO_
SP:16717|logger:log_info:132|INFO|Making discovery call for agent
2ae22a6b4489457abb14373cd3816076
```

```
2019-01-04T17:07:43.746284+00:00 amateen-em7 journal: SCILO_
SP:30843|logger:log_warning:127|WARNING|System file received from aym-
win2012r2-1
```

```
2019-01-04T17:07:43.750553+00:00 amateen-em7 journal: SCILO_
SP:30843|logger:log_warning:127|WARNING|Discovery call within time
threshold, sleeping.
```

```
2019-01-04T17:07:46.676827+00:00 amateen-em7 journal: SCILO_
SP:16717|logger:log_info:132|INFO|Update agent request did: 4, oid: 0,
ip: 10.7.6.119, agent id: 2ae22a6b4489457abb14373cd3816076
```

```
2019-01-04T17:07:46.677114+00:00 amateen-em7 journal: SCILO_
SP:16717|logger:log_info:132|INFO|Discovery complete, getting new agent
device id. Downloading new config for device:
2ae22a6b4489457abb14373cd3816076.
```

```
2019-01-04T17:07:47.420509+00:00 amateen-em7 journal: SCILO_
SP:6954|logger:log_warning:127|WARNING|Agent id:
2ae22a6b4489457abb14373cd3816076 being given a return code: 2
```

## Example /var/log/uwsgi/streamer.log for successful discovery in the SL1 Distributed Architecture

```
10.234.196.19 - - [29/Sep/2017:14:04:52 +0000] "POST /api/update_
agent/agent/ HTTP/1.1" 200 59 "-" "python-requests/2.7.0 CPython/2.7.5
Linux/3.10.0-514.10.2.el7.x86_64"
```

## Save incoming data for a specific device ID in the SL1 Distributed Architecture

```
PYTHONPATH=/opt/em7/lib/python3:/opt/streamer_prime python3
/opt/streamer_prime/streamer_prime/manage.py agent_save_xml -d <agent
guid> -e true
```

## Save incoming data for a specific device ID in the SL1 Extended Architecture

```
kubectl exec -it $(kubectl get pods -l app=streamer -o jsonpath="{.items
[0].metadata.name}") -- python -m streamer agent_save_data --host_id
<host id> --enable true
```

You can find the host id from the ADS url, such as [https://<sl1\\_address>/ads/servers/13/system](https://<sl1_address>/ads/servers/13/system)). You can locate the files in the `/tmp/save_agent_data` directory.

---

## Additional Troubleshooting Situations and Best Practices

The following situations might occur while configuring or working with agents:

Situation	Cause / Resolution
Two device records exist for the same device on the <b>Devices</b> page in SL1.	<p>This situation occurs when SL1 first discovered this device with SNMP, and then the agent was installed and started polling that device. This duplication of records also occurs if the agent was installed first, and then you ran an SNMP discovery.</p> <p>To address this issue, you can <b>merge</b> the device records using the classic user interface. For more information, see the "Managing a Single Device with the Device Administration Panel" chapter in the <b>Device Management</b> manual.</p>
The SNMP device record has IPv4, but the agent device record has IPv6.	The agent reports all network interfaces to the message collector. The Message Collector uses the first "bound" IP address reported by the agent.

Situation	Cause / Resolution
	To address this issue, you can manually edit the agent device record in the "classic" user interface and update the IP address.
If you uninstall an agent and then run a different installation executable file, you still see the same organization ID for the agent record.	<p>After you uninstall the agent, the <b>scilog.conf</b> file is left on the server in case the agent is reinstalled. SL1 can reuse the same device record and maintain historical performance data for that agent.</p> <p>To address this issue, delete the <b>scilog.conf</b> file after you run the uninstallation. If you install this agent again, SL1 assigns a new organization ID to the agent and creates a new device record.</p>

## Agent Communication with SL1

This section covers the various codes that are sent to the SL1 agent when additional actions are needed.

### Return Codes

The core method of communication between the SL1 agent and SL1 involves the data files the agent uploads to the Streamer. When the agent uploads a file to the Streamer, a number of conditions are checked and the return code given to the agent determines what action should be taken, if any.

Return Code	Meaning
200	No action needed.
409	The data file uploaded by the agent has a problem. The Streamer did not accept it and it should be deleted from the host.
503	Data uploads are occurring more quickly than normal. The agent should refrain from sending its next file until a specified amount of time has passed.
505	The agent should stop streaming.

### Sub-codes

The Streamer communicates another set of actions in a JSON response included with the return code to the SL1 agent.

JSON Response Field	Type	Meaning
send_system_file	boolean	Send a system config file as the next upload from this agent.
get_agent_config	boolean	Retrieve an updated agent config from the Streamer.
get_agent_update	boolean	Download a new version of the agent to update.
get_pd_config	boolean	Retrieve an updated polled data config.
get_log_config	boolean	Retrieve an updated log config.
purge_uploads	boolean	Delete any backed-up data and only upload new files.
set_data_backoff	integer	Do not upload a data file until the set number of seconds have passed.

JSON Response Field	Type	Meaning
set_log_backoff	integer	Do not upload a log file until the set number of seconds have passed.

© 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](mailto: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