

Windows Event Forwarding to Linux server using Powershell script

Overview

This PowerShell script forwards Windows event logs to a Linux server using the syslog protocol. It captures specific event logs, sends them to the specified syslog server, and ensures that duplicate events are not sent.

Prerequisites

- PowerShell on Windows
- Syslog server running on Linux (e.g., Rocky Linux) with an accessible IP
- UDP port 514 open for communication

Powershell Script

Save the file as **.ps1** (e.g sendlogs.ps1).

Script:

```
# Define the syslog server IP address and port
$syslogServerIP = "192.168.20.24" # Replace with your Rocky server's IP
$syslogPort = 514

# File to store last sent event info.
# Change this directory if necessary
$logFilePath = "C:\Users\Administrator\Desktop\lastEventInfo.txt"

# Initialize last sent events
$lastSentEvents = @{ }
```

```

# Check if the file exists and read the last sent events
if (Test-Path $logFilePath) {
    Write-Host "Loading last sent events from file."
    $lastSentEvents = Get-Content $logFilePath | ConvertFrom-Json
}

# Loop for sending logs
while ($true) {
    Write-Host "Checking for new logs..."

    # Define the logs you want to forward
    $logNames = @("Application", "Security", "Setup", "System")

    # Loop through each log
    foreach ($logName in $logNames) {
        Write-Host "Processing log: $logName"

        # Get new logs
        $logs = Get-WinEvent -LogName $logName | Sort-Object TimeCreated

        foreach ($log in $logs) {
            # Create a unique key based on Event ID and TimeCreated
            $eventKey = "$($log.Id)-$($log.TimeCreated.Ticks)"

            # Check if the event has already been sent
            if (-not $lastSentEvents.ContainsKey($eventKey)) {
                # Create syslog message format
                $message = "<134>" + $log.TimeCreated.ToString("yyyy-MM-dd HH:mm:ss") + " " +
                $log.ProviderName + ": " + $log.Message

                # Send the message to the syslog server
                $client = New-Object System.Net.Sockets.UdpClient
                $client.Connect($syslogServerIP, $syslogPort)
                $bytes = [System.Text.Encoding]::ASCII.GetBytes($message)
                $client.Send($bytes, $bytes.Length)
                $client.Close()

                # Mark the event as sent
                $lastSentEvents[$eventKey] = $true
                Write-Host "Sent log: $message"
            }
        }
    }
}

```

```

    } else {
        Write-Host "Log already sent: $eventKey"
    }
}
}

# Save the last sent events to the file
$lastSentEvents | ConvertTo-Json | Set-Content -Path $logFilePath
Write-Host "Last sent events updated."

# Wait for 1 second before running again
Start-Sleep -Seconds 1
}

```

Script Components

1. Define Variables:

- `$syslogServerIP`: IP address of the Linux syslog server.
- `$syslogPort`: Port number for syslog (default is 514).
- `$logFilePath`: Path to store the last sent event information.

2. Initialize Last Sent Events:

- Loads previously sent events from a file, if it exists.

3. Main Loop:

- Continuously checks for new logs from specified log categories: `Application`, `Security`, `Setup`, and `System`.

4. Processing Logs:

- Retrieves new logs, sorts them, and creates a unique key based on the event ID and timestamp.
- Sends new logs to the syslog server in a specified message format.
- Updates the log file with the newly sent events.

5. Error Handling:

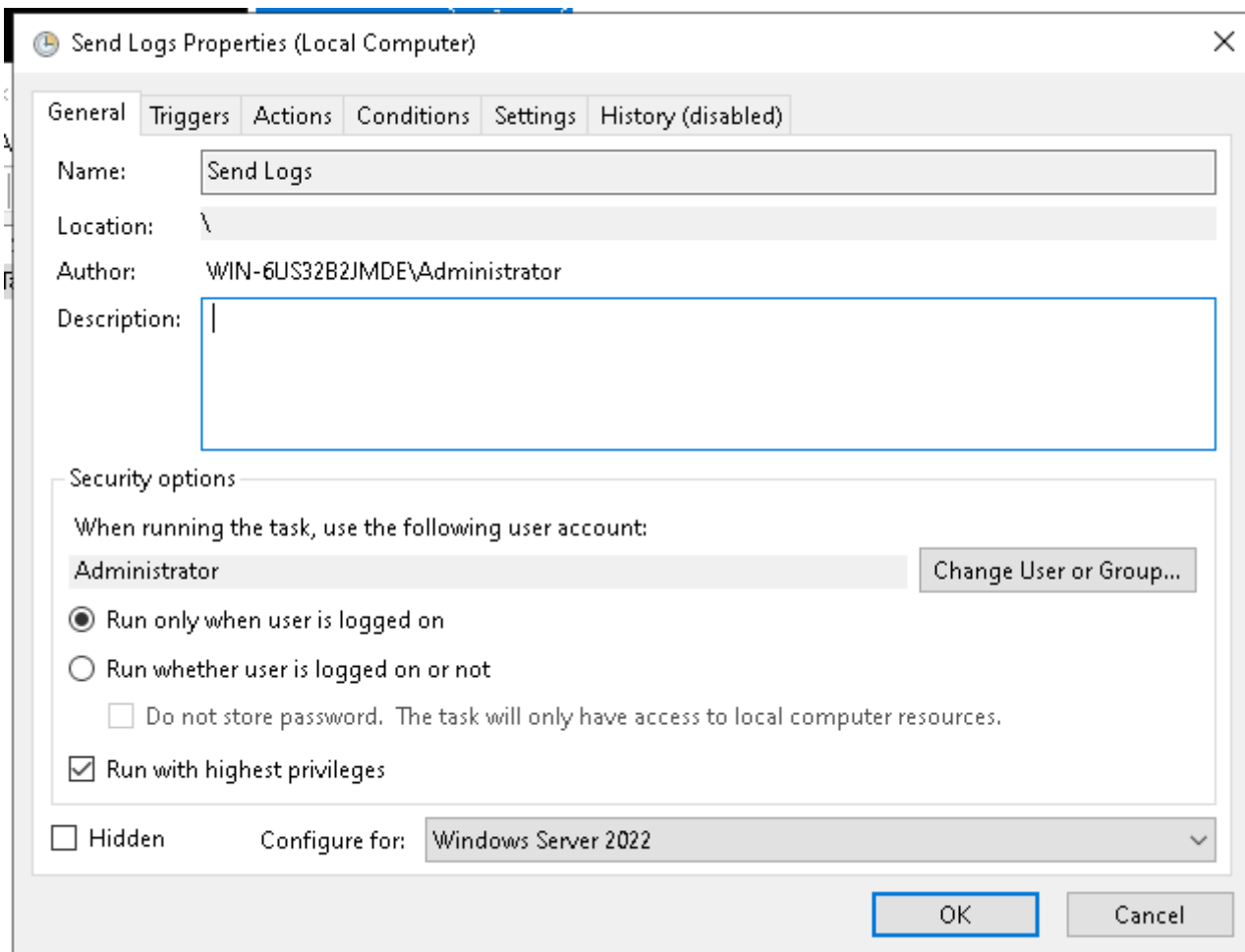
- Logs messages indicating whether an event has been sent or is a duplicate.

Usage

1. Update the `$syslogServerIP` and `$logFilePath` variables.
2. Run the script in PowerShell. It will run indefinitely, checking for new logs every second.

Task Scheduler

Make sure to enable the "Run with highest privileges"



Add a new action

1. Fill in the Program/Script Text Box with: **powershell.exe**
2. Fill in the Add arguments Text Box with: **-ExecutionPolicy Bypass -File "C:\Users\Administrator\Desktop\sendlogs.ps1"**

tion

sp

l.ex

De

pti

...

Edit Action

×

You must specify what action this task will perform.

Action: Start a program

▼

Settings

Program/script:

powershell.exe

Browse...

Add arguments (optional):

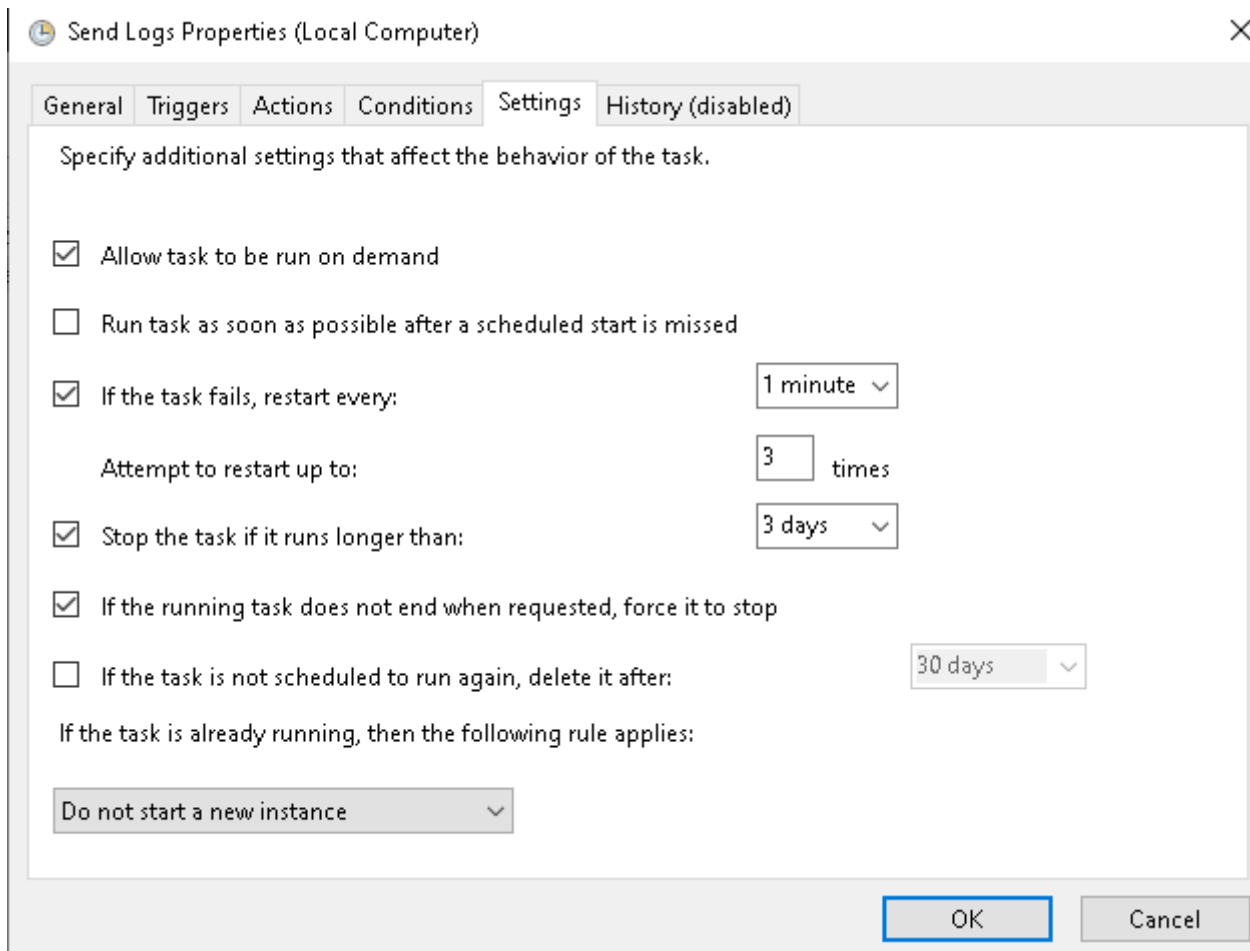
-ExecutionPolicy Bypass

Start in (optional):

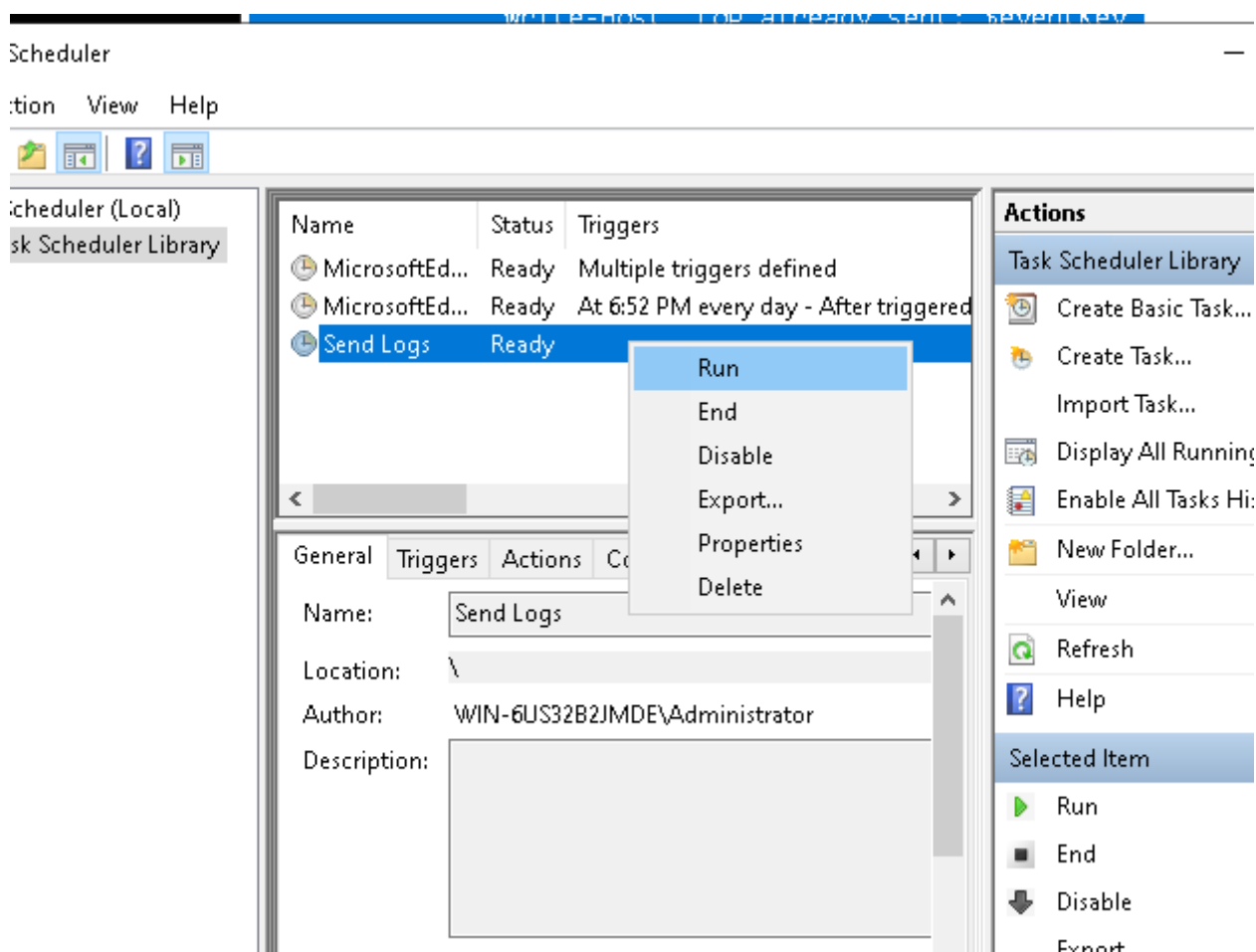
OK

Cancel

Settings Configuration



After the creating the task, you can enable the script by activating the task.



Revision #2

Created 22 October 2024 07:44:41 by Eduardo Dominico Llosa

Updated 17 December 2024 10:32:49 by Aldion Pueblos