Enhanced Process Monitor - Usage Guide

🔍 Overview

The enhanced Process Monitor script provides comprehensive monitoring of critical system processes with AppDynamics integration, improved error handling, and flexible output formats.

🚀 Quick Start

Basic Usage (AppDynamics Format)

```powershell
# Run with default settings
.\ProcessMonitor.ps1

# Output example:
# name=Custom Metrics|ProcessMon|splunkd,value=1
# name=Custom Metrics|ProcessMon|httpd,value=1
```

Using Configuration File

```powershell
# Use custom configuration
.\ProcessMonitor.ps1 -ConfigFile "processes.json"
```

Different Output Formats

```powershell
# JSON output for integration
.\ProcessMonitor.ps1 -OutputFormat JSON

# CSV output for reporting
.\ProcessMonitor.ps1 -OutputFormat CSV

# Console output for interactive use
.\ProcessMonitor.ps1 -OutputFormat Console
```

📋 Parameters

Parameter	Type	Default	Description
`ConfigFile`	String	None	Path to JSON configuration file
`OutputFormat`	String	AppDynamics	Output format: AppDynamics, JSON, CSV, Console
`LogFile`	String	None	Path to log file for debugging
`IncludeDetails`	Switch	False	Include CPU and memory metrics
`Quiet`	Switch	False	Suppress console output except metrics

📂 Configuration File

Create a `processes.json` file to customize monitoring:

```json
{
    "ProcessNames": [
        "splunkd",
        "httpd",
        "java",
        "CSFalconService"
    ],
    "MetricPrefix": "Custom Metrics|ProcessMon",
    "TimeoutSeconds": 30
}
```

💡 Usage Examples

1. AppDynamics Integration (Default)

```powershell
# Standard AppDynamics metrics output
.\ProcessMonitor.ps1

# With detailed metrics (CPU, Memory)
.\ProcessMonitor.ps1 -IncludeDetails

# With custom configuration
.\ProcessMonitor.ps1 -ConfigFile "production-processes.json"
```

2. Logging and Debugging

```powershell
```

```powershell
# Enable detailed logging
.\ProcessMonitor.ps1 -LogFile "C:\Logs\process-monitor.log"

# Quiet mode (only output metrics)
.\ProcessMonitor.ps1 -Quiet -LogFile "C:\Logs\process-monitor.log"
```

3. Reporting and Analysis

```powershell
# Generate JSON report
.\ProcessMonitor.ps1 -OutputFormat JSON -IncludeDetails > process-report.json

# Generate CSV for Excel
.\ProcessMonitor.ps1 -OutputFormat CSV -IncludeDetails > process-report.csv

# Interactive console view
.\ProcessMonitor.ps1 -OutputFormat Console -IncludeDetails
```

4. Scheduled Monitoring

```powershell
# Create scheduled task for continuous monitoring
.\ProcessMonitor.ps1 -ConfigFile "processes.json" -Quiet -LogFile "C:\Logs\process-$(Get-Date -Format 'yyyyMMdd').lo
```

🔧 Advanced Configuration

Environment-Specific Configurations

Development Environment (`dev-processes.json`):

```json
{
  "ProcessNames": [
    "java",
    "httpd",
    "splunkd"
  ],
  "MetricPrefix": "Custom Metrics|ProcessMon|Dev"
}
```

Production Environment (`prod-processes.json`):

```json
{
  "ProcessNames": [
    "CSFalconService",
    "BESClient",
    "QualysAgent",
    "splunkd",
    "httpd",
    "java",
    "ora_pmon_cdb19300",
    "tnslsnr"
  ],
  "MetricPrefix": "Custom Metrics|ProcessMon|Prod"
}
```

Integration with AppDynamics Machine Agent

1. **Copy script to Machine Agent directory:**

```powershell
Copy-Item "ProcessMonitor.ps1" "C:\AppDynamics\machine-agent\monitors\ProcessMonitor\"
```

2. **Create monitor.xml:**

```xml
<monitor>
  <name>ProcessMonitor</name>
  <type>managed</type>
  <description>Custom Process Monitor</description>
  <monitor-configuration>
    <execution-style>periodic</execution-style>
    <execution-frequency-in-seconds>60</execution-frequency-in-seconds>
    <properties>
      <property name="command" value="powershell.exe"/>
      <property name="command-arguments" value="-File ProcessMonitor.ps1 -ConfigFile processes.json -Quie
    </properties>
  </monitor-configuration>
</monitor>
```

📊 Output Format Examples

AppDynamics Format

```
name=Custom Metrics|ProcessMon|splunkd,value=1
name=Custom Metrics|ProcessMon|httpd,value=1
name=Custom Metrics|ProcessMon|java,value=1
```

JSON Format

```
json

{
  "Timestamp": "2025-01-15 10:30:00",
  "Processes": [
    {
      "Name": "splunkd",
      "Id": 1234,
      "CPU": 15.25,
      "WorkingSet": 512.75,
      "Status": "Running"
    }
  ],
  "Summary": {
    "Total": 3,
    "MonitoredProcesses": 25
  }
}
```

Console Format

```
=== Process Monitor Results ===
Timestamp: 2025-01-15 10:30:00
Processes Found: 3 / 25 monitored

Name      Id   CPU   WorkingSet Status
----      --   ---   ---------- ------
splunkd   1234 15.25 512.75     Running
httpd     5678 8.50  256.30     Running
java      9012 25.75 1024.50    Running
```

🔍 Troubleshooting

Common Issues

1. **"Access Denied" errors:**

```powershell
# Run PowerShell as Administrator
Start-Process PowerShell -Verb RunAs
```

2. **Configuration file not found:**

```powershell
# Verify file path
Test-Path "processes.json"
```

3. **No processes found:**

```powershell
# Enable detailed logging
.\ProcessMonitor.ps1 -LogFile "debug.log" -OutputFormat Console
```

Performance Optimization

For systems with many processes:

```powershell
# Use smaller process lists in configuration
# Enable quiet mode to reduce output overhead
.\ProcessMonitor.ps1 -ConfigFile "critical-only.json" -Quiet
```

📝 Best Practices

1. **Use configuration files** for different environments
2. **Enable logging** for troubleshooting
3. **Start with critical processes** and expand gradually
4. **Test in quiet mode** before production deployment
5. **Monitor script performance** in high-frequency scenarios
6. **Use meaningful metric prefixes** for different environments

🔄 Integration with CI/CD

```yaml
# Azure DevOps Pipeline example
- task: PowerShell@2
  displayName: 'Process Health Check'
  inputs:
    filePath: 'ProcessMonitor.ps1'
    arguments: '-ConfigFile "$(environment)-processes.json" -OutputFormat JSON'
    workingDirectory: '$(Build.SourcesDirectory)/monitoring'
```

📋 Maintenance

Regular Tasks

- Review and update process lists quarterly
- Rotate log files to prevent disk space issues
- Test configuration changes in non-production first
- Monitor script execution time and resource usage