# 測試帖子（無許可）

## 目錄



## 簡介

本文檔介紹對APIC GUI體驗緩慢進行故障排除的一般方法。

## 快速入門

人們經常發現，APIC GUI問題緩慢是由於來自指令碼、整合或應用程式的API請求速率過高造成的。APIC的access.log會記錄每個已處理的API請求。可以使用Github資料中心組aci-tac-scripts專案中的訪問日誌分析器指令碼快速分析APIC的access.log。

# 背景資訊

## APIC作為Web伺服器 — NGINX

NGINX是負責每個APIC上可用的API終端的DME。如果NGINX關閉，則無法處理API請求。如果NGINX擁塞，則API擁塞。每個APIC都運行自己的NGINX進程，因此如果任何主動式查詢器僅針對APIC，則可能只有一個APIC會出現NGINX問題。

APIC UI執行多個API請求以填充每個頁面。類似地，所有APIC show命令（NXOS樣式CLI）都是執行多個API請求、處理響應然後提供給使用者的python指令碼的包裝。

## 相關日誌

| 日誌檔名 | 位置 | 它位於哪一技術支援中 | 備註 |
|---|---|---|---|
| access.log | /var/log/dme/log | APIC 3of3 | 與ACI無關，每個API請求提供1行 |
| error.log | /var/log/dme/log | APIC 3of3 | ACI不可知，顯示nginx錯誤（包括限制） |
| nginx.bin.log | /var/log/dme/log | APIC 3of3 | 特定於ACI，記錄DME事務 |
| nginx.bin.warnplus.log | /var/log/dme/log | APIC 3of3 | ACI特定包含警告+嚴重性的日誌 |

# 方法

### 隔離初始觸發器

什麼受到影響？

- 哪些APIC受到影響；一個、多個或所有APIC?
- 哪裡可以看到遲緩；通過UI和/或CLI命令？
- 哪些特定UI頁面或命令速度較慢？

這種緩慢的感受如何？

- 對於單個使用者而言，是否會在多個瀏覽器中看到這種情況？
- 多個使用者是否報告速度緩慢？還是僅報告單個/子集使用者？
- 受影響的使用者是否共用從瀏覽器到APIC的相似地理位置或網路路徑？

遲緩是何時被首次注意到的？

- 最近是否新增了ACI整合或指令碼？
- 最近是否啟用了瀏覽器擴展？
- ACI配置最近有變化嗎？

## 檢查NGINX使用情況和運行狀況

Access.log條目格式

access.log是NGINX的一項功能，因此與APIC無關。每行代表APIC收到的1個HTTP請求。參考此日誌，瞭解APIC的NGINX使用情況。

ACI 5.2+版上的預設access.log格式：

```
log_format proxy_ip '$remote_addr ($http_x_real_ip) - $remote_user [$time_local]'
                    '"$request" $status $body_bytes_sent '
                    '"$http_referer" "$http_user_agent"';
```

此行表示執行moquery -c fvTenant時的access.log條目：

```
127.0.0.1 (-) - - [07/Apr/2022:20:10:59 +0000]"GET /api/class/fvTenant.xml HTTP/1.1" 200 15863 "-" "Pyt
```

示例access.log條目對映到log_format:

| log_format欄位 | 示例內容 | 備註 |
|---|---|---|
| $remote_addr | 127.0.0.1 | 傳送此請求的主機的IP |
| $http_x_real_ip | - | 使用代理時最後一個請求者的IP |
| $remote_user | - | 一般不使用。選中nginx.bin.log以跟蹤登入執行請求的使用者 |
| $time_local | 2022年4月07日：20:10:59 +0000 | 處理請求的時間 |
| $request | 獲取/api/class/fvTenant.xml HTTP/1.1 | Http方法(GET、POST、DELETE)和URI |

| | | |
|---|---|---|
| $status | 200 | HTTP響應狀態代碼 |
| $body_bytes_sent | 1586 | 響應負載大小 |
| $http_referer | - | - |
| $http_user_agent | Python-urllib | 傳送請求的客戶端型別 |

Access.log行為

在一段較長的時間內發生高速請求突發：

- 每秒40多個請求的連續突發會導致使用者介面速度變慢
- 確定負責查詢的主機
- 減少或禁用查詢源，檢視這是否提高了APIC響應時間。

一致的4xx或5xx響應：

- 如果找到，請識別來自nginx.bin.log的錯誤消息

可以使用Github資料中心組aci-tac-scripts專案中的訪問日誌分析器指令碼快速分析APIC的access.log。

檢查NGINX資源使用情況

可以使用APIC中的top命令檢查NGINX CPU和記憶體使用情況：

<#root>

```
top - 13:19:47 up 29 days, 2:08, 11 users, load average: 12.24, 11.79, 12.72
Tasks: 785 total, 1 running, 383 sleeping, 0 stopped, 0 zombie
%Cpu(s): 3.5 us, 2.0 sy, 0.0 ni, 94.2 id, 0.1 wa, 0.0 hi, 0.1 si, 0.0 st
KiB Mem : 13141363+total, 50360320 free, 31109680 used, 49943636 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 98279904 avail Mem

PID   USER PR NI VIRT    RES   SHR   S %CPU %MEM TIME+     COMMAND
21495 root 20 0  4393916 3.5g 217624 S
```

**2.6**

2.8 759:05.78

**nginx.bin**

高NGINX資源使用率直接與高處理請求率相關。

## 檢查核心

NGINX崩潰不常用於慢速APIC GUI問題。但是，如果找到NGINX核心，請將其連線到TAC SR進行分析。有關檢查核心的步驟，請參閱ACI技術支援指南。

## 檢查客戶端到伺服器的延遲

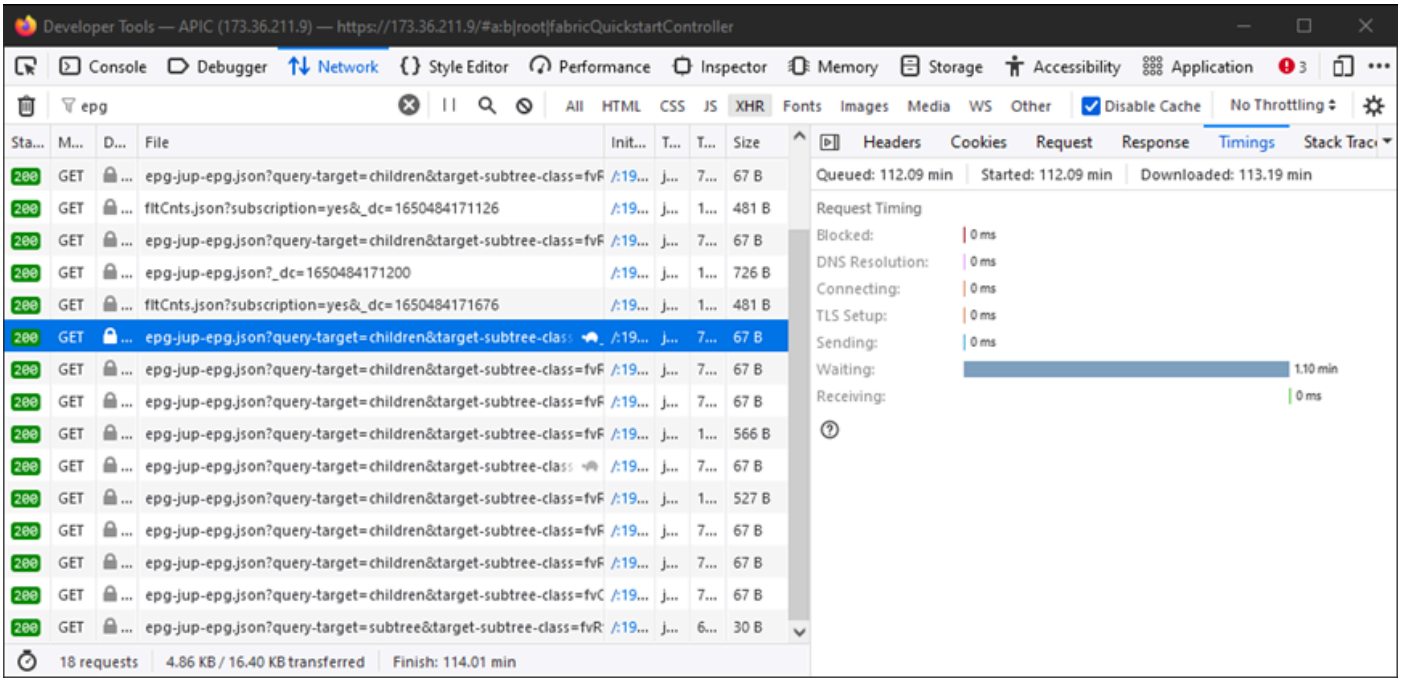
如果找不到快速請求，但使用者繼續表現出UI緩慢，則問題可能是客戶端（瀏覽器）到伺服器(APIC)延遲。

在這些情況下，驗證從瀏覽器到APIC的資料路徑（地理距離、VPN等）。 如果可能，請部署並測試從與APIC位於同一地理區域或資料中心內的跳轉伺服器進行的訪問。驗證其他使用者是否顯示類似的延遲量。

### 瀏覽器開發工具網路頁籤

所有瀏覽器都能夠通過其Browser Development Toolkit(通常在Network（網路）頁籤中)驗證HTTP請求和響應。

此工具可用於驗證來自瀏覽器的請求的每個階段所需的時間，如下圖所示。




等待1.1分鐘以供APIC響應的瀏覽器示例

### 特定UI頁面的增強功能

「策略組」頁：

思科漏洞ID CSCvx14621 - APIC GUI在交換矩陣頁籤中的IPG策略上緩慢載入。

Inventory頁面下的介面：

思科錯誤ID CSCvx90048 — 「Layer 1 Physical Interface Configuration」（第1層物理介面配置）Operational（操作）頁籤的初始載入為長/導致「凍結」。

客戶端>伺服器延遲的一般建議

預設情況下，某些瀏覽器（如Firefox）允許每個主機擁有更多的Web連線。

- 檢查此設定是否可以在使用的瀏覽器版本上配置
- 對於多查詢頁（如「策略組」頁），這一點更為重要

VPN和與APIC的距離會根據客戶端瀏覽器請求和APIC響應旅行時間增加整體UI延遲。在APIC上本地的跳轉框可顯著減少瀏覽器到APIC的旅行時間。

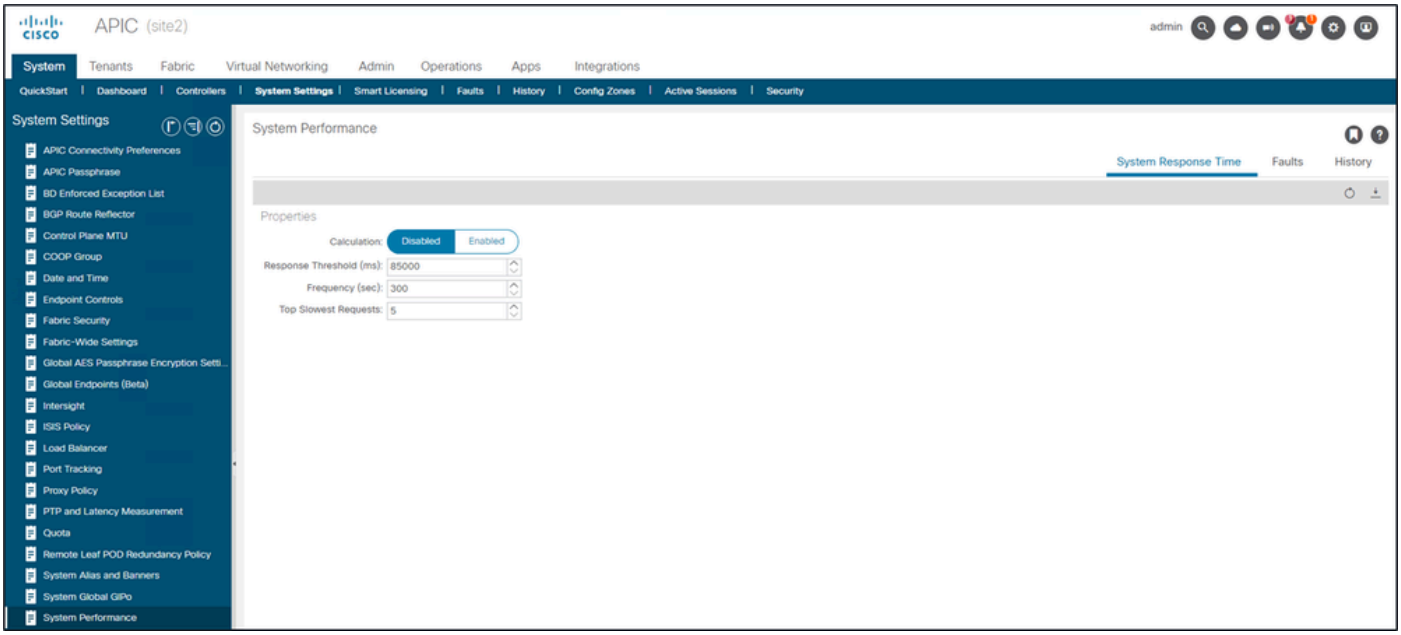
## 檢查Long-Web請求

如果Web伺服器（APIC上的NGINX）處理大量的長Web請求，這可能會影響並行接收的其他請求的效能。

對於具有分散式資料庫的系統（如APIC）尤其如此。單個API請求可能需要向結構中的其他節點傳送額外的請求和查詢，這可能會導致預期的更長的響應時間。這些長Web請求在短時間幀內突發，會增加所需的資源量，導致意外的更長的響應時間。此外，接收的請求隨後會超時（90秒），導致從使用者角度而言出現意外的系統行為。

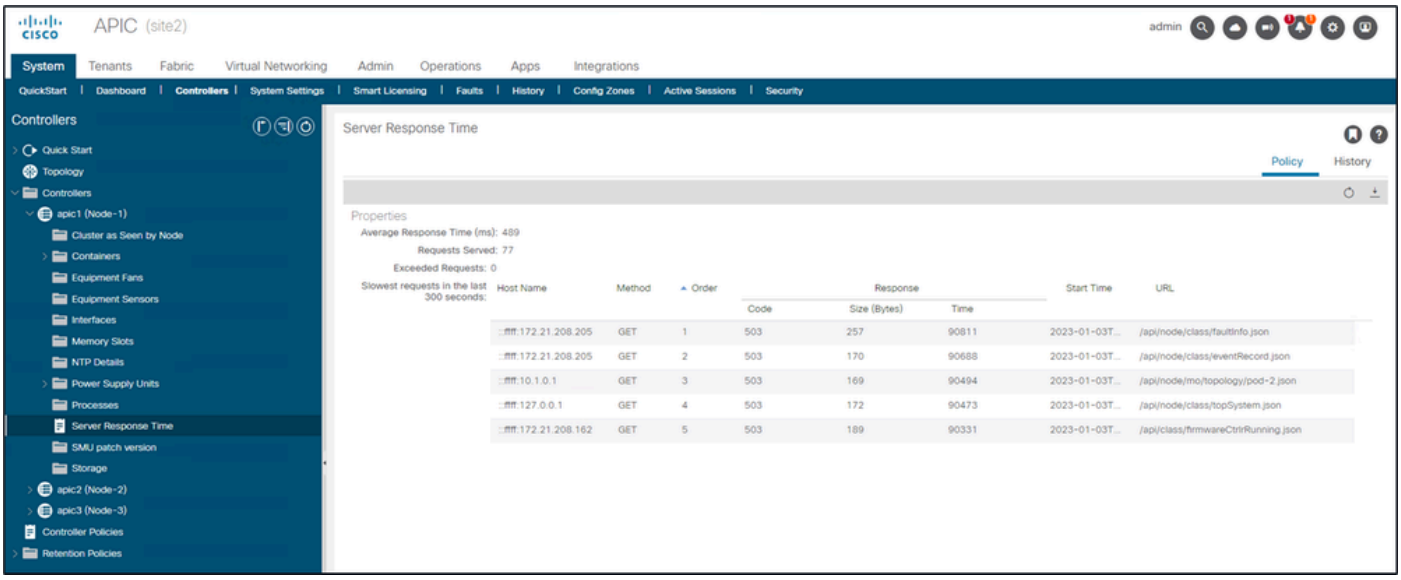
系統響應時間 — 啟用伺服器響應時間的計算

在4.2(1)+中，使用者可以啟用「系統效能計算」，跟蹤並突出顯示花費時間處理的API請求。




可以從系統 — 系統設定 — 系統效能啟用計算

啟用「計算」後，使用者可以導航到控制器下的特定APIC，檢視最近300秒內最慢的API請求。




System - Controllers - Controllers資料夾 — APIC x — 伺服器響應時間

# APIC API使用注意事項

## 確保指令碼不會損害Nginx的一般指標

- 每個APIC運行自己的NGINX DME。
  - 只有APIC 1的NGINX處理對APIC 1的請求。APIC 2和3的NGINX不處理這些請求。
- 一般來說，在很長一段時間內每秒40個以上的API請求會使NGINX衰弱。
  - 如果找到，則減少請求的攻擊性。
  - 如果無法修改請求主機，請考慮對APIC執行NGINX速率限制。

## 解決指令碼效率低下問題

- 在每個API請求之前不要登入/註銷。
  - 一個登入會話的預設超時為10分鐘。此同一會話可用於多個請求，並可以刷新以延長有效時間。
  - 請參閱思科APIC REST API配置指南 — 訪問REST API — 身份驗證和維護API會話。
- 如果您的指令碼查詢共用父級的多個DN，而不是使用查詢過濾器將查詢摺疊為單個邏輯父級查詢。
  - 請參閱Cisco APIC REST API配置指南 — 合成REST API查詢 — 應用查詢範圍篩選器。
- 如果您需要更新對象或對象類，請考慮websocket預訂而不是快速API請求。

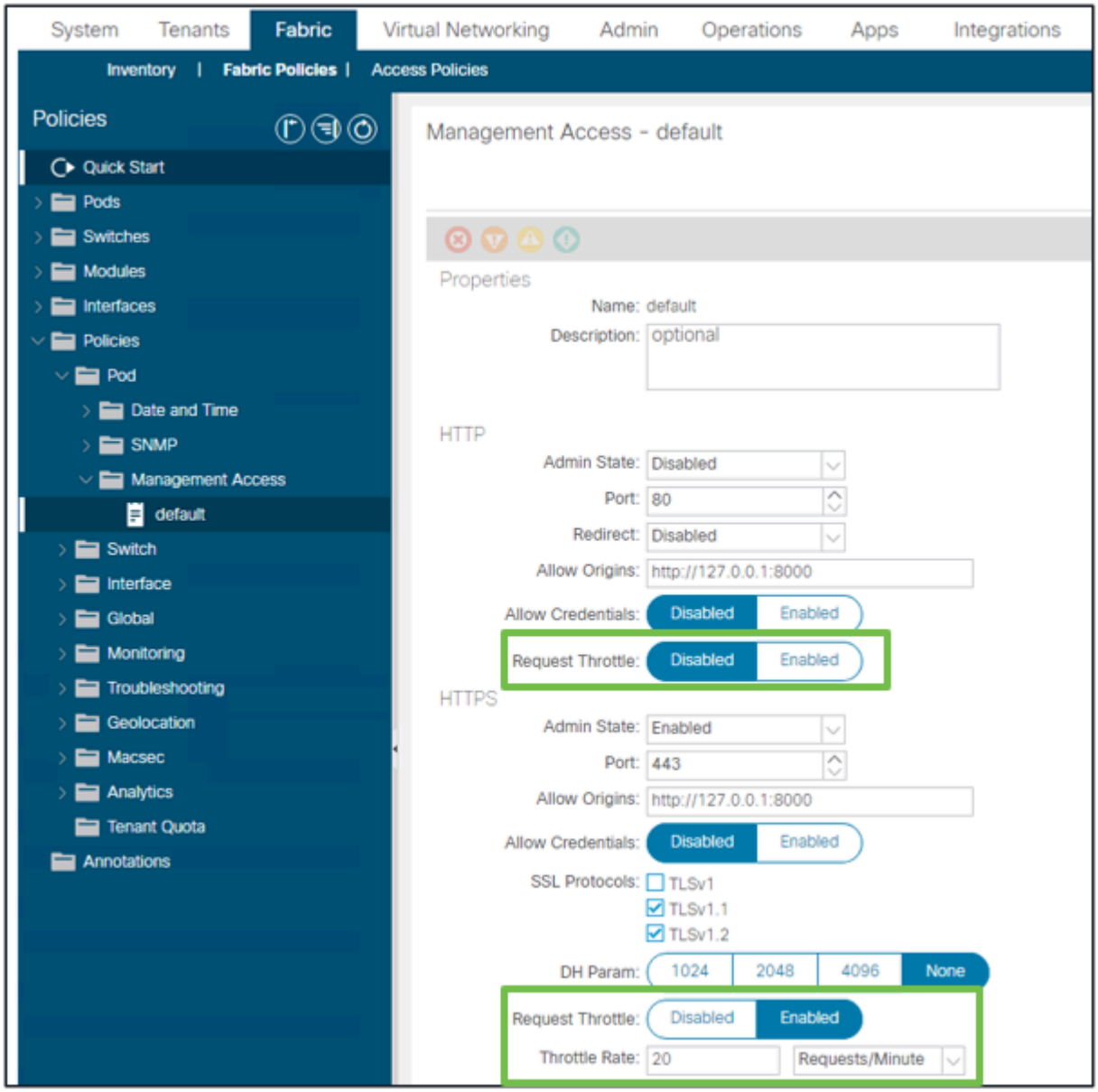
## NGINX請求限制

4.2(1)+版本中提供，使用者可以獨立啟用針對HTTP和HTTPS的請求限制。

---

註：從ACI 6.1(2)版本開始，此功能支援的最大速率從10,000 r/m降低到每秒40個請求(r/s)或每分鐘2400個請求(r/m)。

---




交換矩陣 — 交換矩陣策略 — 策略資料夾 — 管理訪問資料夾 — 預設

啟用時：

- 重新啟動NGINX以應用配置檔案更改
  - 新區域httpsClientTagZone將寫入nginx config
- 限制速率可以設定為每分鐘請求數(r/m)或每秒請求數(r/s)。
- 請求限制依賴於NGINX中包括的速率限制實施
  - 針對/api/URI的API請求使用使用者定義的限制速率+突發=（限制速率x 2）+無延遲
    - /api/aaaLogin和/api/aaaRefresh有一個不可配置的限制(zone aaaApiHttps)，速率限製為2r/s + burst=4 + nodelay
  - 基於每個客戶端IP地址跟蹤請求限制
  - 源自APIC自身IP(UI + CLI)的API請求繞過限制
  - 任何超過使用者定義的限制速率+突發閾值的客戶端IP地址都會收到來自APIC的503響應
  - 這些503可在訪問日誌中關聯
  - error.log具有指示何時啟用限制(區域httpsClientTagZone)以及針對哪些客戶端主機的條目

<#root>

apic#

**less /var/log/dme/log/error.log**

...
2023/04/17 20:19:14 [error] ...

**limiting requests**

, excess: 40.292 by zone "

**httpsClientTagZone**

", client: h.o.s.t, ... request: "GET /api/class/...", host: "a.p.i.c"
2023/04/17 20:19:14 [error] ...

**limiting requests**

, excess: 40.292 by zone "

**httpsClientTagZone**

", client: h.o.s.t, ... request: "GET /api/node/...", host: "a.p.i.c"

通常，Request Throttle僅用於保護伺服器(APIC)免受查詢激進型客戶端引起的類似DDOS的症狀。在應用/指令碼邏輯中瞭解並隔離請求攻擊型客戶端，以獲得最終解決方案。

# 建議

這些建議旨在幫助降低APIC的負載和運營壓力，尤其是在沒有單一來源負責大量API呼叫的情況下。通過實施這些最佳實踐，您可以最大限度地減少交換矩陣中不必要的處理、日誌記錄和事件生成，從而提高系統穩定性和效能。這些建議在聚合行為而不是孤立事件導致APIC緊張的環境中尤其重要。

## 禁用ACL日誌記錄

確保在正常操作期間關閉ACL日誌記錄。僅在計畫維護時段啟用該功能，以便進行故障排除或調試。連續記錄可能會生成過多資訊性消息，尤其是當多台交換機上的大量流量丟棄時，會增加APIC工作負載。

有關詳細資訊，請參閱《思科APIC安全配置指南》（5.2.x指南的連結）：
https://www.cisco.com/c/en/us/td/docs/dcn/aci/apic/5x/security-configuration/cisco-apic-security-configuration-guide-release-52x/security-policies-52x.html

## 將系統日誌轉換為嚴重事件

配置系統，以便僅將嚴重性為ALERT的系統日誌消息轉換為eventRecords。避免轉換INFORMATION級別（包括ACL.logging），以防止噪音事件淹沒APIC:

1.導航到Fabric → Fabric Policies → Policies → Monitoring → Common Policy → Syslog Message Policies → Default。
2.調整設施過濾器，將系統日誌嚴重性設定為警報。

## 靜音非必需事件代碼

抑制（抑制）與監控無關的事件代碼需要降低噪音。
要壓制事件代碼E4204939，請在任何APIC CLI上使用以下命令：

```
bash
icurl -k -sX POST -d '<fabricInst><monCommonPol><eventSevAsnP code="E4204939" sev="squelched"/></monCom
```

要驗證：

```
bash
icurl -k -sX GET 'https://localhost/api/node/class/eventSevAsnP.xml' | xmllint --format -
```

或者，通過UI檢查：
Fabric > Fabric Policies > Policies > Monitoring > Common Policy > Event嚴重性分配策略

## 最佳化ND訂用刷新

對於由3.2.2m或4.1.1g之前的ND版本管理的交換矩陣，請升級到其中一個版本或更高版本以最佳化訂閱刷新間隔。早期版本每45秒刷新一次MO，這樣在規模上每天可以產生300,000多個APIC請求。更新的版本將訂閱超時增加至3600秒（1小時），將刷新減少到每天約5,000次。

## 監視Intersight相關查詢

支援Intersight的交換矩陣從DC聯結器生成定期的topsystem查詢（每15秒），從而增加APIC負載。在6.1.2及更高版本中，此查詢已針對減少的開銷進行了最佳化。

## 調整記錄的保留策略

將eventRecord 、 faultRecord和healthRecord的保留策略設定為1,000以防止記錄過度累積。當您針對任何特定操作活動定期提取這些記錄時，此功能尤其有用。請始終根據您的操作和故障排除要求評估降低監控粒度帶來的影響。

關於此翻譯

思科已使用電腦和人工技術翻譯本文件，讓全世界的使用者能夠以自己的語言理解支援內容。請注意，即使是最佳機器翻譯，也不如專業譯者翻譯的內容準確。Cisco Systems, Inc. 對這些翻譯的準確度概不負責，並建議一律查看原始英文文件（提供連結）。