Finesse를 위한 CORS(Cross-Origin Resource Sharing) 이해

목차

소개

이 문서에서는 문제 해결 과정에서 기본 프로세스를 철저히 파악할 수 있도록 교차 출처 리소스 공유에 대해 자세히 설명합니다.

사전 요구 사항

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- Cisco UCCE(Unified Contact Center Enterprise) 릴리스 12.6.X
- Cisco PCCE(Packaged Contact Center Enterprise) 릴리스 12.6.X
- Cisco Finesse 릴리스 12.6.X
- Cisco CUIC(Unified Intelligence Center) 릴리스 12.6.X

사용되는 구성 요소

이 문서의 정보는 다음 소프트웨어 및 하드웨어 버전을 기반으로 합니다.

- UCCE 릴리스 12.6.2
- Finesse 릴리스 12.6.2
- CUIC 릴리스 1.6.2

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

배경 정보

CORS란?

CORS(Cross-Origin Resource Sharing)는 서버가 리소스에 액세스할 수 있는 웹 사이트(도메인, 프로토콜, 포트)를 제어하는 방법입니다. 일반적으로 브라우저는 서로 다른 원본(동일 원본 정책)의 요청을 차단하지만, CORS는 서버에 이 제한을 선택적으로 완화할 수 있는 권한을 부여합니다. 기본적으로 서버는 브라우저에 어떤 출처가 허용되는지, 어떤 유형의 요청이 허용되는지(예: GET, POST 등), 어떤 사용자 지정 헤더가 포함될 수 있는지 알려주기 위해 특수 HTTP 헤더를 사용합니다. 이를 통해 서버는 API에 액세스할 수 있는 사람과 방법을 결정할 수 있으며, 이는 완전 개방부터 엄격하게 제한된 액세스까지 다양합니다. CORS는 브라우저와 서버가 이러한 HTTP 헤더를 통해 통신하여 교차 원본 요청을 관리하도록 합니다.

CORS는 HTTP 헤더를 사용하여 제어 교차 원본 요청을 활성화합니다. 브라우저와 서버는 이러한 헤더를 통해 통신하며, 서버는 허용되는 원본, 메서드 및 헤더를 지정합니다. 서버의 응답 헤더가 없거나 잘못된 경우 브라우저는 응답을 차단하여 동일한 원본 정책을 적용합니다. 특정 요청의 경우 브라우저가 먼저 서버에 프리플라이트 요청을 보내 실제 교차 원본 요청을 수락하는지 확인합니다.

브라우저에서는 프리플라이트 요청을 사용하여 실제 요청을 보내기 전에 서버에서 교차 원본 요청을 허용하는지 확인합니다. 이러한 프리플라이트 요청에는 HTTP 메서드 및 사용자 지정 헤더와 같은 세부사항이 포함됩니다. 그런 다음 CORS 지원 서버는 실제 요청을 허용하거나 거부하여 응답할 수 있습니다. 서버가 CORS에 대해 구성되지 않은 경우 프리플라이트(preflight)에 올바르게 응답하지 않으며 브라우저가 실제 요청을 차단하므로 원치 않는 교차 원본 액세스로부터 서버를 보호합니다.

CORS(Cross-Origin Resource Sharing)는 웹 보안 및 기능에 매우 중요합니다. 다른 원본(도메인, 프로토콜, 포트)에서 리소스에 대한 제어된 액세스를 허용합니다. 이는 브라우저가 일반적으로 이러한 액세스를 차단하는 동일 원본 정책을 적용하기 때문에 필요합니다.

CORS의 라이프사이클

CORS 요청은 다음 두 가지 측면으로 구성됩니다. 요청을 생성하는 클라이언트와 요청을 수신하는 서버입니다. 클라이언트 측에서 개발자는 JavaScript 코드를 작성하여 서버에 요청을 보냅니다. 서버는 교차 원본 요청이 허용됨을 나타내기 위해 특수 CORS 관련 헤더를 설정하여 요청에 응답합니다. 클라이언트와 서버가 모두 참여하지 않으면 CORS 요청이 실패합니다.

CORS 요청의 주요 플레이어는 클라이언트, 브라우저 및 서버입니다. 클라이언트는 JSON API 응답 또는 웹 페이지의 콘텐츠와 같은 서버의 일부 데이터를 원합니다. 브라우저는 클라이언트가 서버에서 데이터에 액세스할 수 있는지 확인하기 위해 신뢰할 수 있는 중재자 역할을 합니다.

클라이언트:

클라이언트는 웹 사이트에서 실행되는 JavaScript 코드 조각이며 CORS 요청을 시작하는 역할을 합니다

참고: Finesse는 웹 애플리케이션입니다. 서버에 설치되며 에이전트는 웹 브라우저를 사용하

여 간편하게 액세스할 수 있으므로 클라이언트측 설치 또는 플러그인 또는 기타 소프트웨어 유지 보수가 필요하지 않습니다. CORS in Action with Cisco Finesse 예제에서 보여주는 것처럼, 이 아키텍처는 라이브 데이터 보고서와 같은 기능을 지원합니다. 이러한 맥락에서 Cisco Finesse Live Data 가젯의 JavaScript 코드는 클라이언트 역할을 하고, Cisco CUIC는 CORS 라이프사이클 내에서 서버 역할을 합니다. 기본적으로 브라우저 기반 Finesse 클라이언트는 CUIC 서버와 상호 작용하여 라이브 데이터를 검색합니다.

클라이언트 대 사용자:

간혹 client와 user라는 단어가 혼용되어 사용되기도 하지만 CORS의 맥락에서는 다르다. 사용자는 웹 사이트를 방문하는 사람이거나 Finesse 사용자(상담원 또는 수퍼바이저)가 이 컨텍스트에서 Finesse에 액세스하는 반면, 클라이언트는 해당 웹 사이트에서 제공하는 실제 코드입니다. 여러 사용자가 동일한 웹 사이트를 방문하여 동일한 JavaScript 클라이언트 코드를 제공할 수 있습니다.

브라우저:

사용자 에이전트라고도 하는 브라우저에서는 클라이언트측 코드를 호스팅합니다. 또한 발신 요청에 추가 정보를 추가하여 서버가 클라이언트를 식별할 수 있도록 함으로써 CORS에서 중요한 역할을 합니다. 또한 브라우저는 서버의 응답을 해석하여 데이터를 클라이언트에 전달할지 아니면 오류를 반환할지 결정합니다. 이러한 브라우저 측 작업은 동일한 출처 정책에서 제공하는 보안을 유지하는 데 필수적입니다. 브라우저의 CORS 규칙 시행 없이, 클라이언트는 이 중요한 보안 메커니즘을 손상시키는 무단 요청을 할 수 있습니다.

서버:

서버는 CORS 요청의 대상이며 Cisco Finesse의 Live Data 가젯 예제를 위한 CUIC입니다. 서버는 클라이언트가 원하는 데이터를 저장하고, CORS 요청의 허용 여부에 대한 최종 결정권을 가진다.

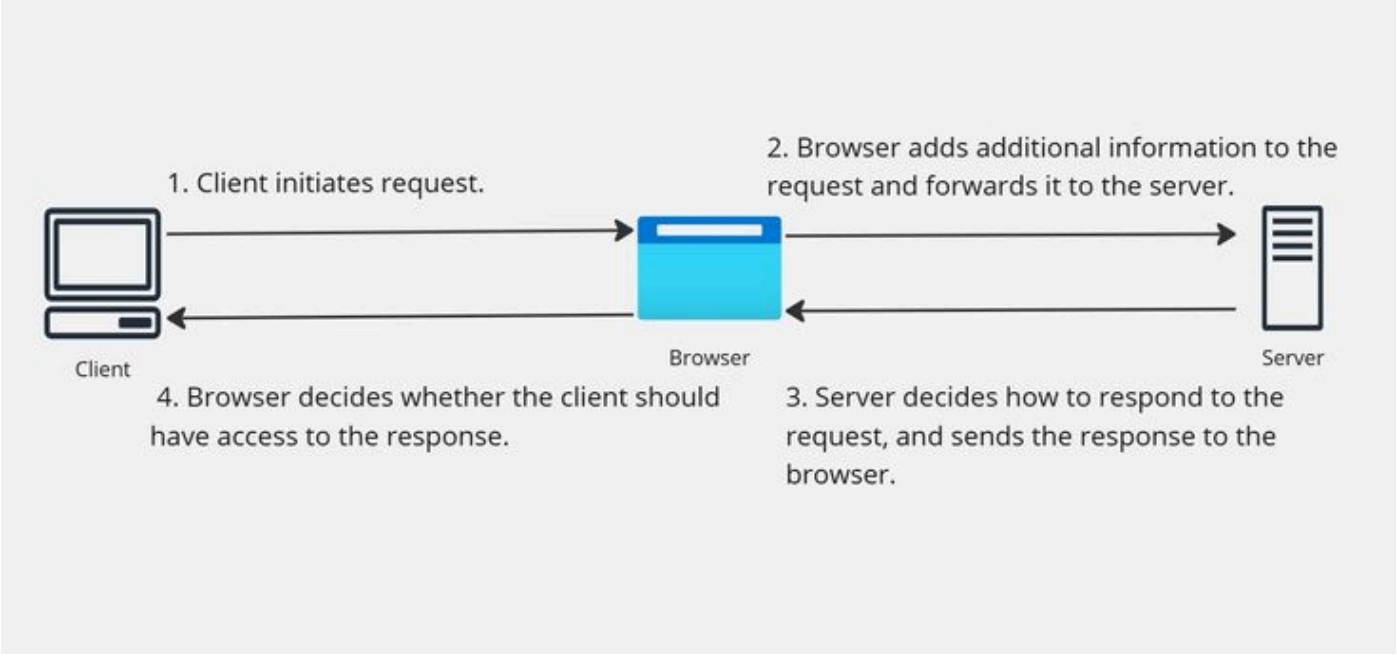
이제 CORS 요청에 누가 관련되어 있는지 알게 되었으니, 이들이 어떻게 협력하는지 살펴보도록 하겠습니다. 다음 그림에서는 상위 레벨 CORS 라이프사이클을 보여줍니다.

1. 클라이언트가 요청을 시작합니다.

2. 브라우저에서 요청에 추가 정보를 추가하고 서버에 전달합니다.

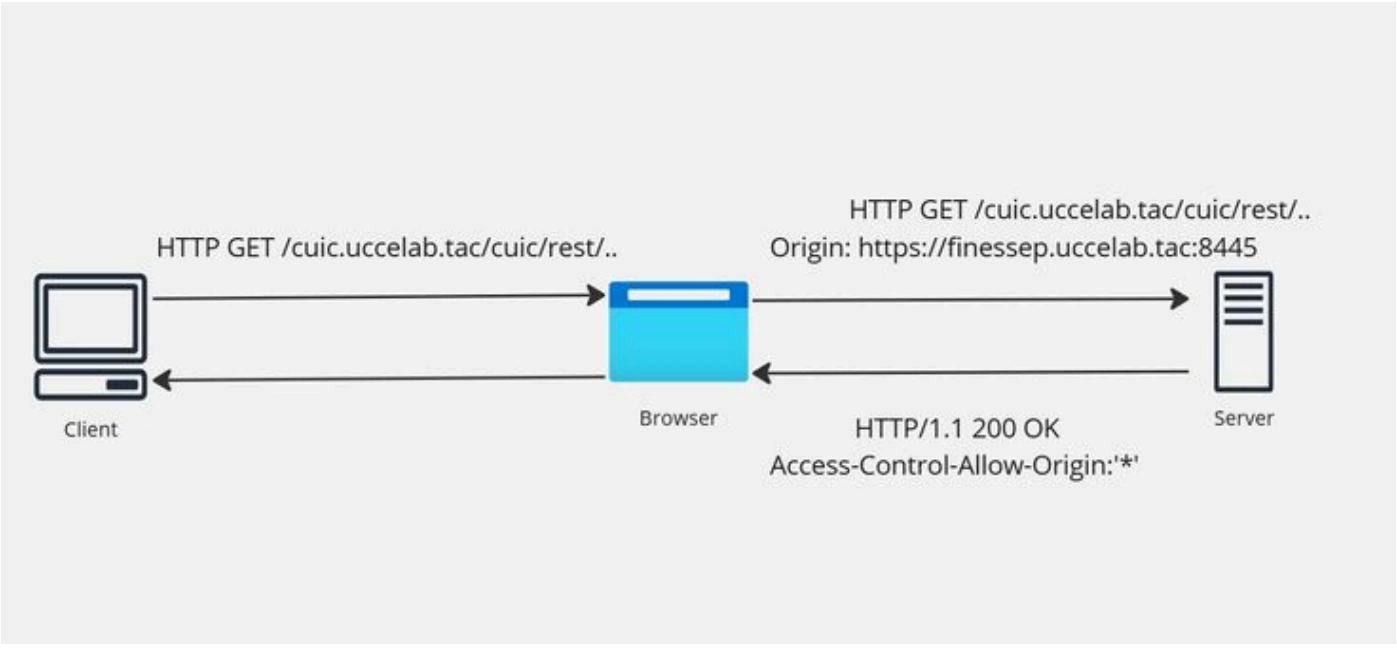
3. 서버가 요청에 응답하는 방법을 결정하고, 응답을 브라우저로 전송합니다.

4. 브라우저는 클라이언트가 응답에 액세스할 수 있어야 하는지 여부를 결정하고, 응답을 클라이언트로 전달하거나 오류를 반환합니다.

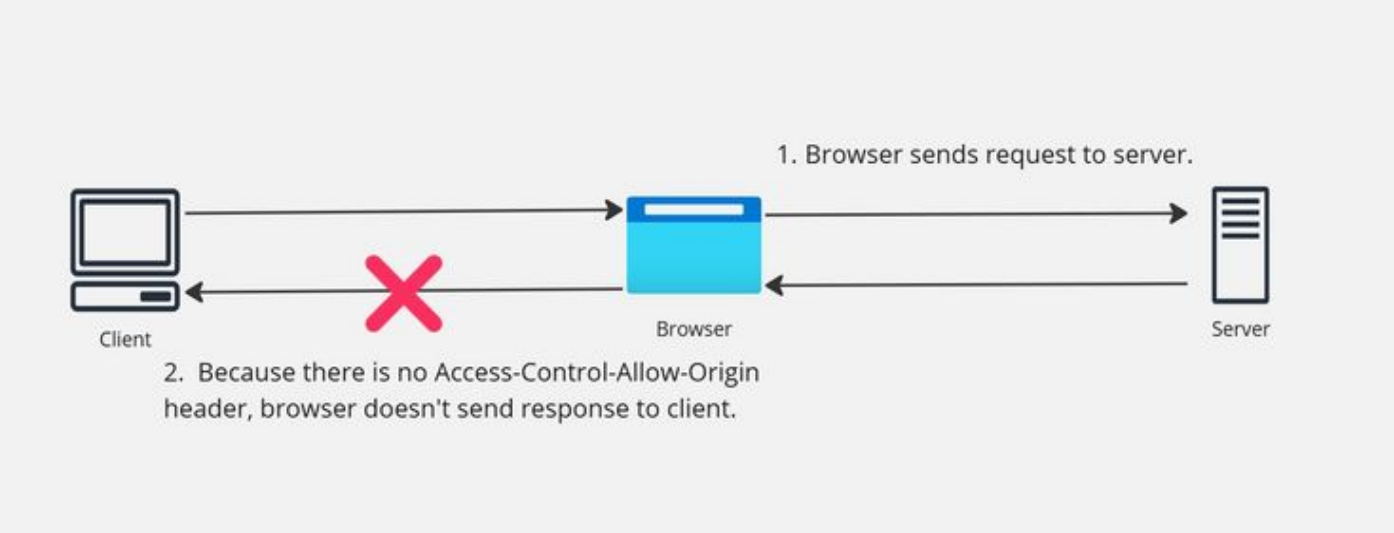


교차 원본 요청을 보내기 전에 브라우저는 HTTP 요청에 원본 헤더를 자동으로 추가합니다. 클라이언트가 수정할 수 없는 이 헤더는 CORS의 중요한 부분이며 클라이언트의 출처(클라이언트 리소스가 로드된 도메인, 프로토콜 및 포트)를 식별하는 역할을 합니다. 이 보안 측정은 클라이언트가 다른 원본을 가장하는 것을 방지합니다. 클라이언트가 서버의 출처를 알려주는 방식이므로 오리진 헤더는 CORS의 기본이 됩니다.

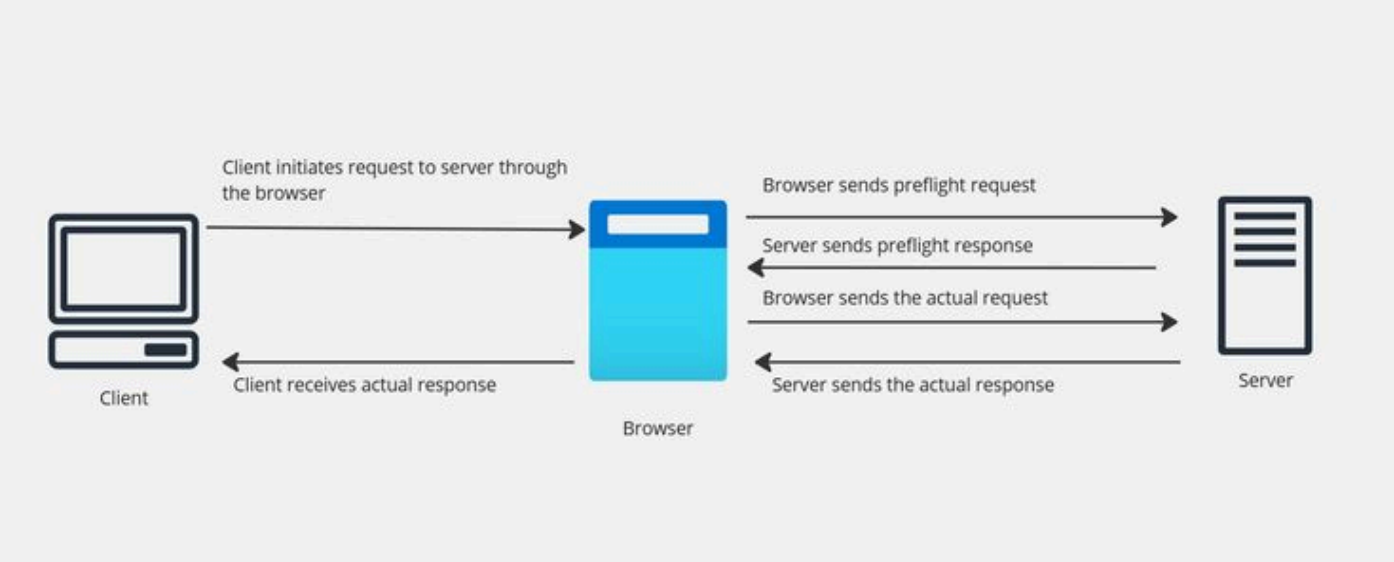
CORS(Cross-Origin Resource Sharing) 상호 작용에서 초기 요청의 Origin 헤더로 클라이언트의 원본을 식별합니다. 그런 다음 서버는 응답에서 Access-Control-Allow-Origin 헤더를 사용하여 클라이언트가 요청된 리소스에 액세스할 수 있는지 여부를 나타냅니다. 이 응답 헤더는 매우 중요합니다. 없으면 CORS 요청이 실패합니다. Access-Control-Allow-Origin 헤더에는 와일드카드(*)를 포함하여 모든 오리진에서 액세스를 허용하거나 특정 오리진에서 해당 특정 클라이언트에 대한 액세스만 허용할 수 있습니다. 이 그림에서는 Access-Control-Allow-Origin을 보여 줍니다. *, CUIC가 모든 원본을 허용한다는 것을 암시하는 경우, CUIC는 일반적으로 실제 시나리오에서 특정 원본과 함께 이 헤더를 전송합니다.



브라우저가 CORS 요청을 거부하면 클라이언트가 서버의 응답에 대한 정보를 받지 못함을 의미합니다. 고객은 오류가 발생했다는 사실만 알고 있을 뿐 구체적인 문제에 대한 세부 사항은 알지 못합니다. 따라서 CORS 오류를 다른 유형의 오류와 구분하기 어렵기 때문에 디버깅 CORS 오류가 어려울 수 있습니다. 초기 요청이 서버로 전송되더라도 서버의 응답에 유효한 Access-Control-Allow-Origin 헤더가 없으면 브라우저가 응답을 차단하고 클라이언트 측에서 오류를 트리거하므로 클라이언트가 서버의 자세한 응답을 볼 수 없습니다.



이 이미지는 특정 유형의 교차 원본 요청을 처리하는 데 필수적인 프리플라이트 단계에 중점을 두고 전체 CORS 프로세스를 설명합니다.



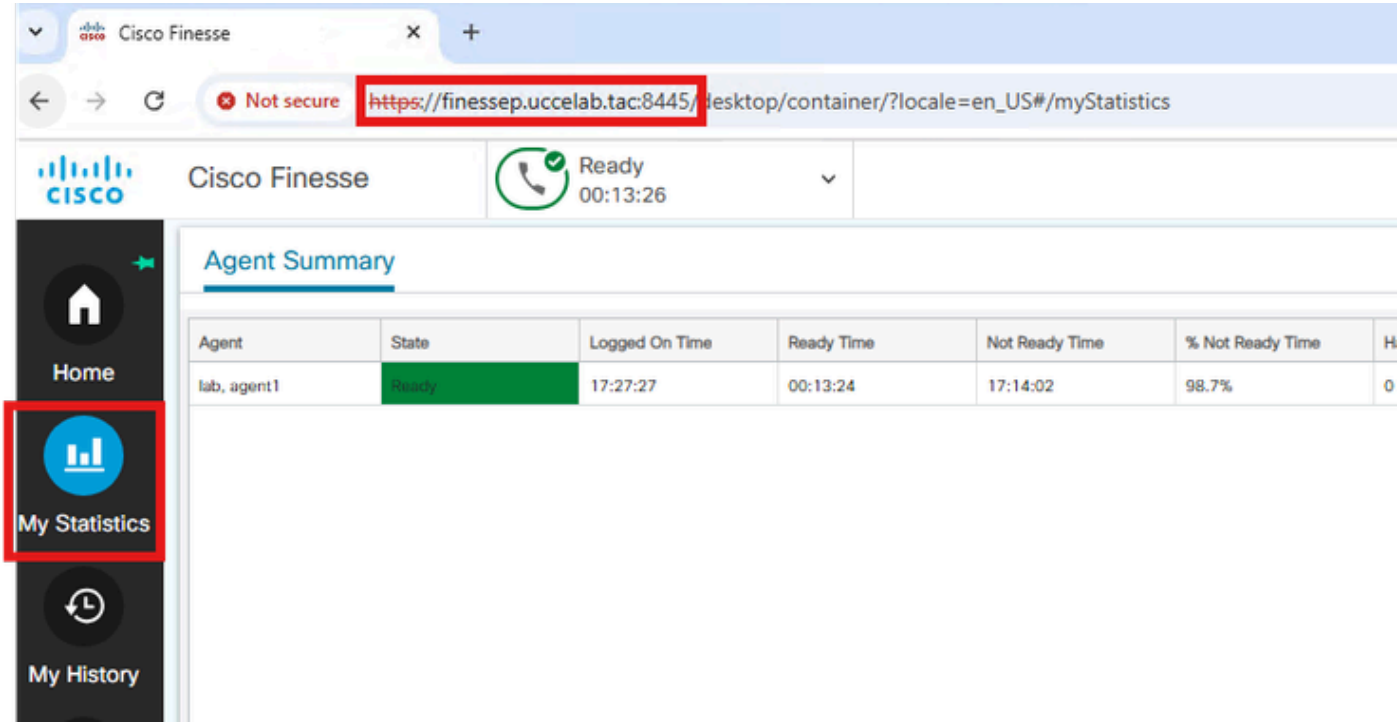
Cisco Finesse를 통한 CORS 활용 사례

예시: Live Data 가젯으로 CORS 동작 분석

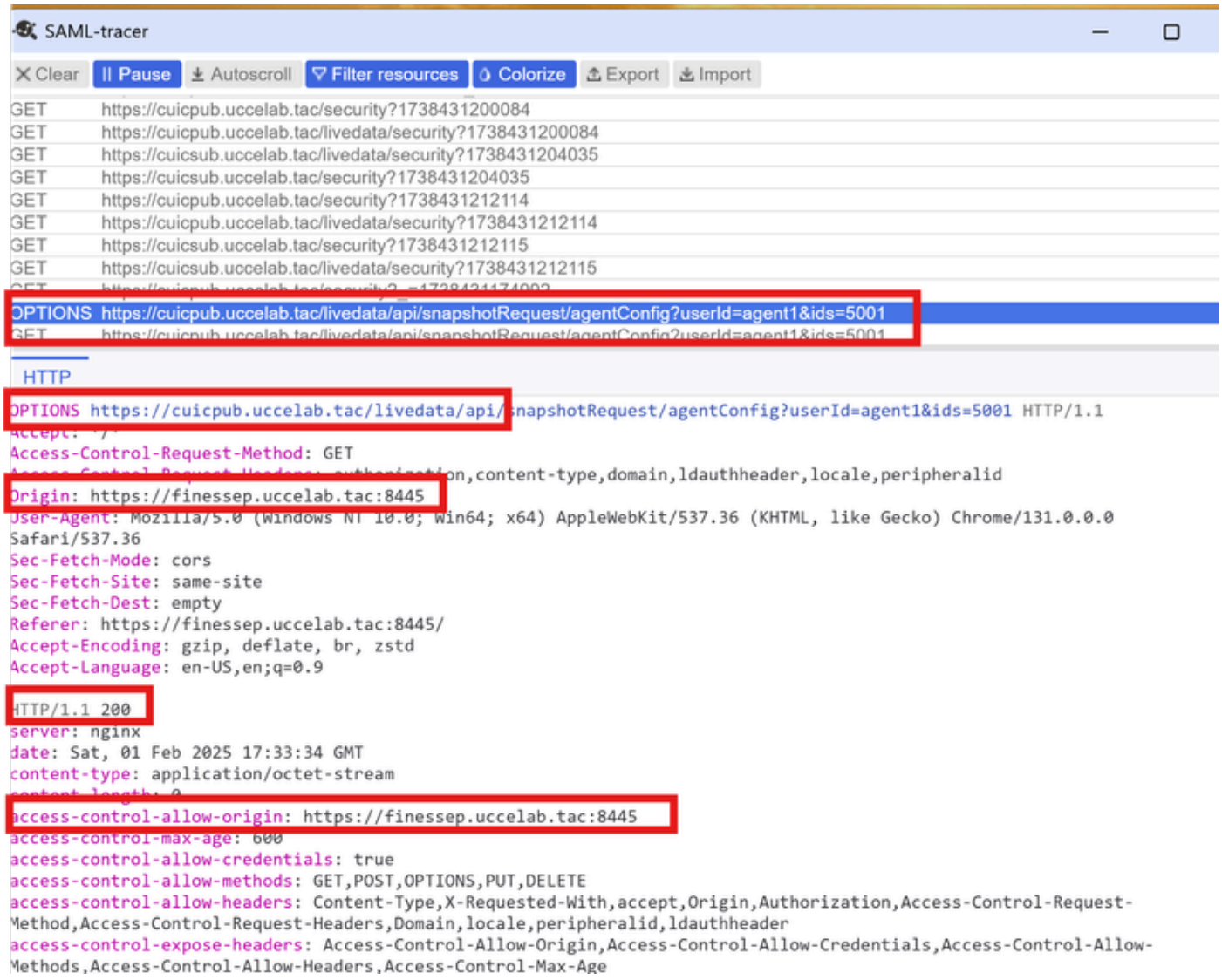
이 섹션에서는 컨택 센터에서 Cisco Finesse와 CORS(Cross-Origin Resource Sharing)를 사용하는 일반적인 방법을 설명합니다. 상담원과 수퍼바이저는 일반적으로 Cisco Finesse를 사용하여 실시간 데이터 보고서에 액세스합니다(이미지 예시에 나와 있음).

상담원이나 수퍼바이저가 보고서 가젯을 클릭하면 해당 작업에서 데이터 검색 요청을 시작합니다.

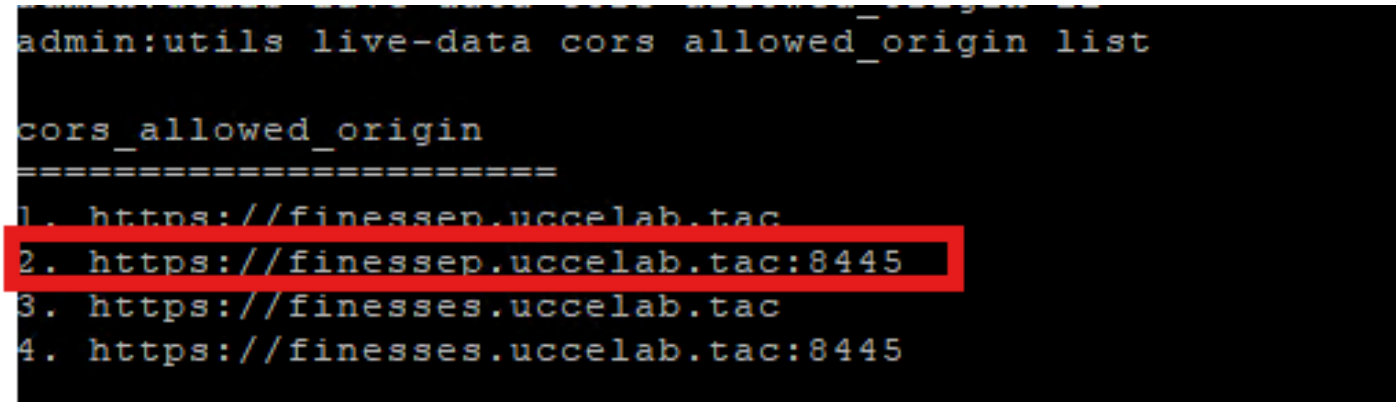
이 요청은 Finesse 애플리케이션의 JavaScript 코드(클라이언트 역할)에서 GET 메서드를 사용하여 CUIC/Live Data 서버로 전송됩니다. SAML 추적기 이미지에 나타난 것처럼, 브라우저는 먼저 서버에 프리플라이트 요청을 보냅니다(앞서 설명한 CORS 라이프사이클).

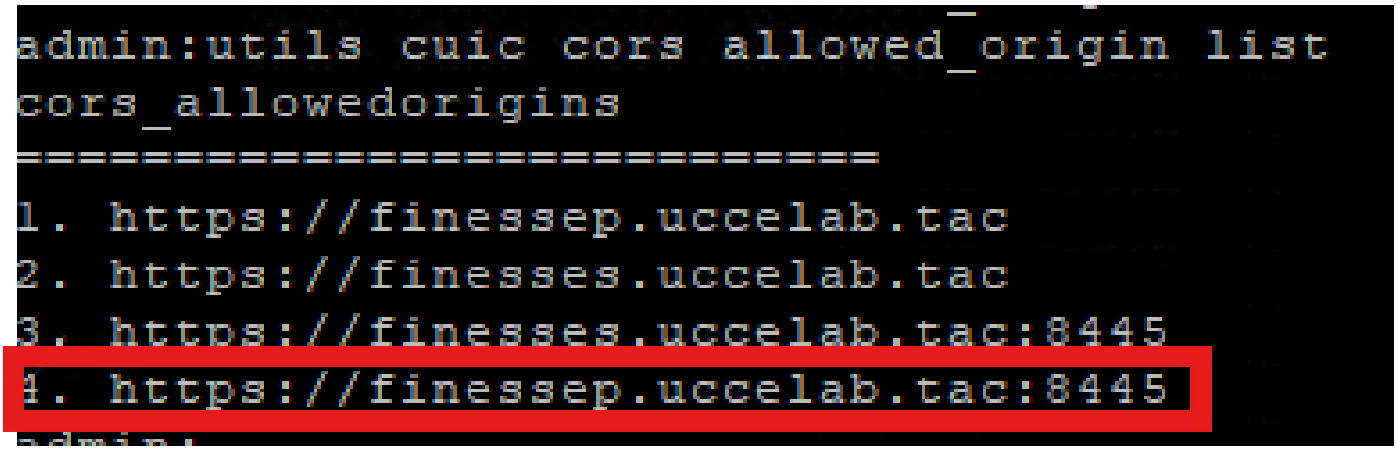


HTTP OPTIONS 요청(프리플라이트 요청)이 CUIC/Live Data 서버로 전송됩니다. 이 요청은 포트 8445를 포함하여 Finesse 서버의 FQDN(정규화된 도메인 이름)으로 원본을 지정합니다. 상담원이 Cisco Finesse 애플리케이션에 액세스하는 데 사용하는 주소와 포트가 동일합니다.



CUIC/Live Data 서버의 CLI(command-line interface) 명령은 라이브 데이터 리소스에 액세스할 수 있는 원본을 제어합니다. Finesse 서버의 원본(FQDN 및 포트)이 이러한 설정에 구성되어 있으면 에이전트는 Finesse 내에서 라이브 데이터 가젯 세부 정보를 볼 수 있습니다.





CORS 연결 테스트를 위한 TAC 툴

서버 측의 CORS 컨피그레이션 오류로 인해 Cisco Finesse의 서드파티 또는 라이브 데이터 가젯에 문제가 발생할 수 있습니다. 이 문서에서는 CORS Quick Check 가젯에 대한 링크를 제공합니다. 이 가젯의 문제 해결 툴은 라이브 데이터 표시 및 기타 서드파티 통합을 비롯하여 Finesse 가젯에 영향을 미치는 Cross-Origin 리소스 공유 문제를 진단할 수 있도록 설계되었습니다.

기술적으로 이 가젯은 Cisco Finesse 클라이언트에서 프리플라이트 요청을 지정된 대상 리소스로 전송하는 방식으로 작동합니다. 이 빠른 확인 기능은 CORS 관련 문제를 신속하게 파악하고 해결하여 문제 해결 프로세스를 가속화합니다.

Finesse 데스크톱에서 Contact Center CORS Quick Check 12.6-v1.0 가젯을 배포하려면

1. Contact Center CORS Quick Check 12.6-v1.0 folder.2에서 가젯 파일을 다운로드합니다.

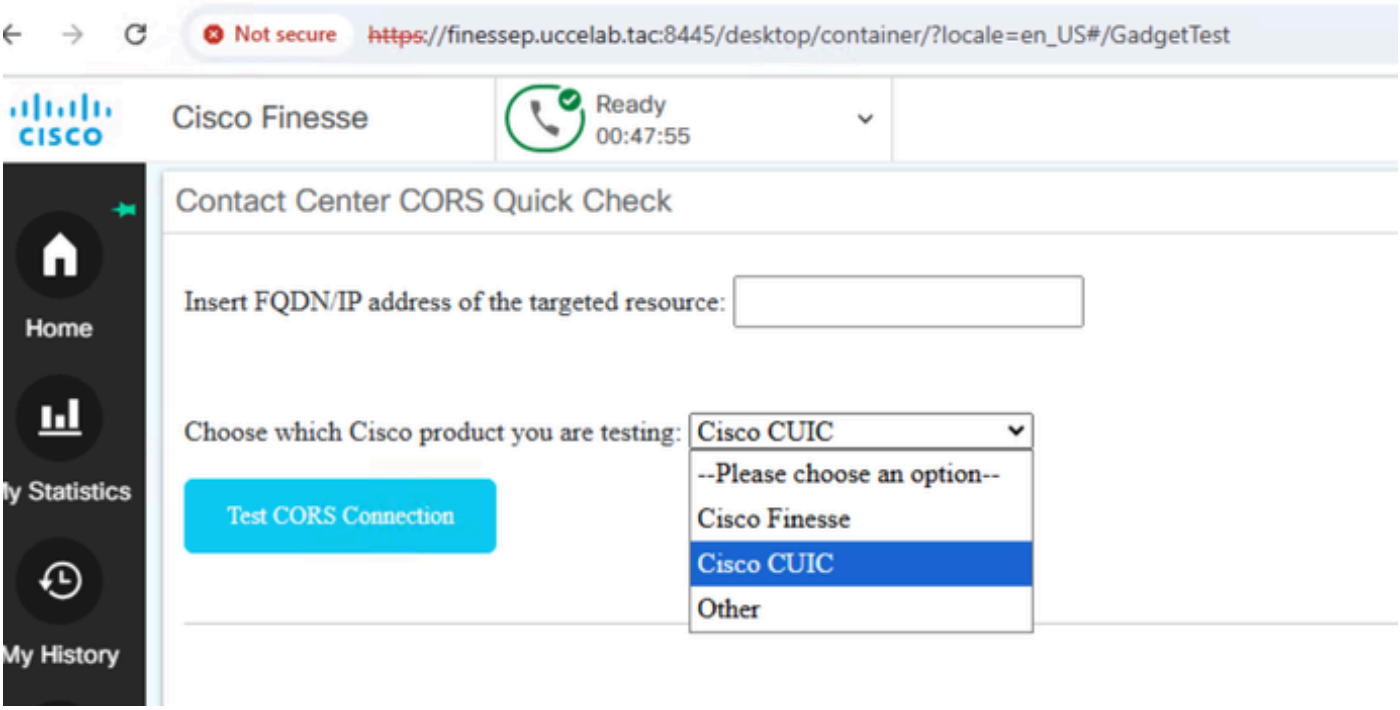
2. Contact Center CORS Quick Check 12.6-v1.0 폴더의 내용을 Finesse 설치 내의 3rdpartygadget 디렉토리에 복사합니다.

3. 가젯을 Finesse 데스크톱 레이아웃의 원하는 사용자 역할(상담원, 수퍼바이저 등)에 추가합니다. 제공된 예제 XML은 이 가젯을 추가하기 위한 올바른 구성을 보여줍니다.

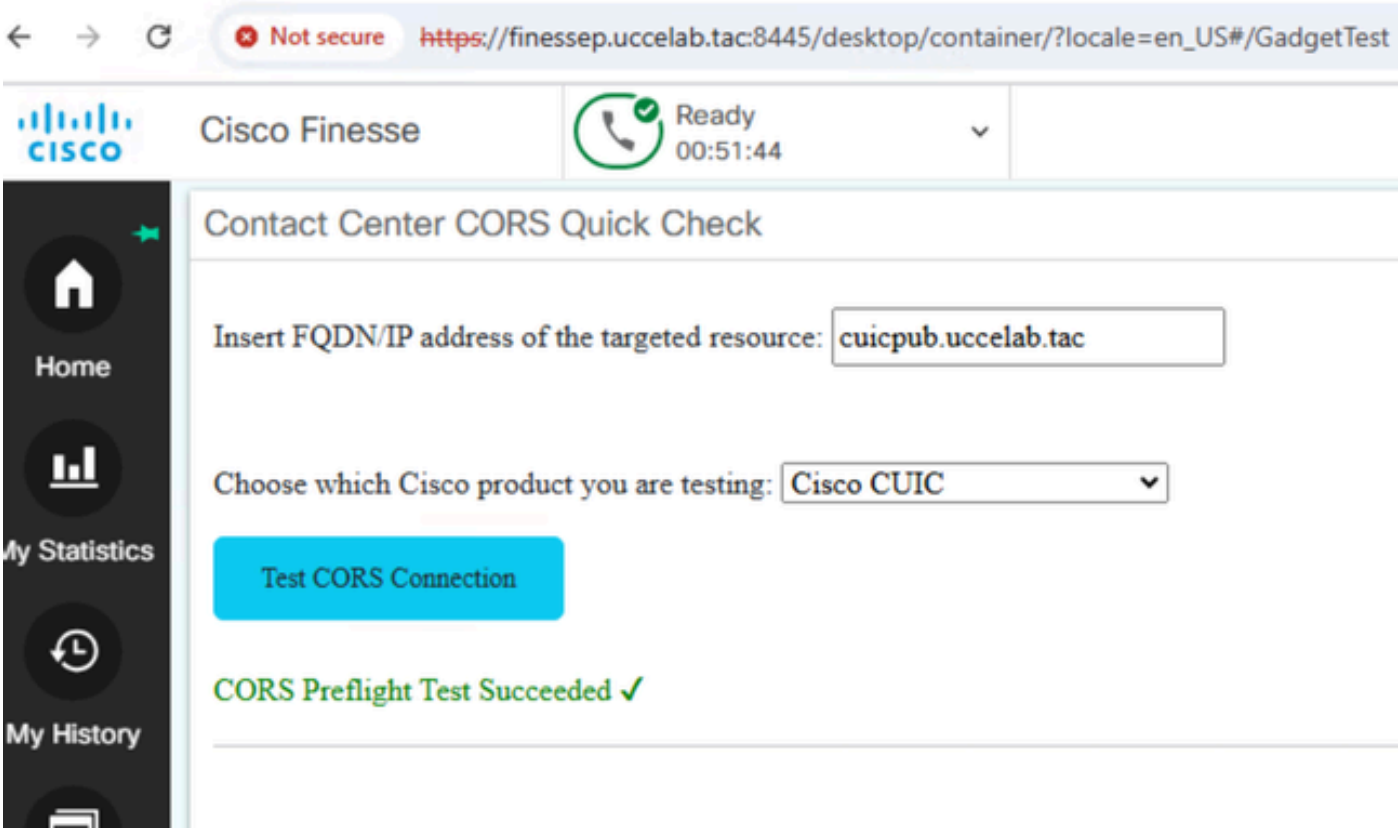
```
<gadget>/3rdpartygadget/files/TestCORSgadget.xml</gadget>
```

타사 가젯을 업로드하고 바탕 화면에 추가하는 방법에 대한 자세한 내용은 Finesse Developer Guide의 Third Party Gadgets 장을 참조하고, Finesse Administration Guide의 Manage Third-Party Gadgets 장을 참조하십시오.

가젯 파일이 업로드되고 Cisco Finesse Tomcat 서비스가 다시 시작되면 가젯을 사용할 수 있으며 GUI(그래픽 사용자 인터페이스)가 표시됩니다.



맨 위의 드롭다운 목록에서 CUIC를 선택할 수 있습니다. 제공된 필드에 CUIC 서버의 FQDN(정규화된 도메인 이름)을 입력합니다. 성공적인 테스트는 여기에 표시된 대로 진행될 것입니다.



테스트가 성공하면 CUIC 서버가 Finesse 서버와의 CORS(Cross-Origin Resource Sharing)에 맞게 구성되었음을 의미합니다. 브라우저의 SAML 추적기 로그에는 HTTP OPTIONS 요청(CORS 프리플라이트)이 CUIC 서버로 전송되었음을 보여줍니다. 이 요청에는 Finesse 서버의 주소가 Origin 헤더에 포함되어 있습니다. CUIC 서버가 200 OK HTTP 메시지로 응답했으며, 응답의 Access-Control-Allow-Origin 헤더에도 Finesse 서버의 주소가 포함되어 있었습니다. 이렇게 하면 CUIC 서

버가 Finesse 서버 출처의 요청을 허용하도록 구성되어 CORS가 올바르게 설정되었는지 확인합니다.

<#root>

OPTIONS https://cuicpub.uccelab.tac/cuic/ HTTP/1.1

sec-ch-ua-platform: "Windows"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome..
sec-ch-ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A Brand";v="24"
sec-ch-ua-mobile: ?0
Accept: */*

Origin: https://finessep.uccelab.tac:8445

Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://finessep.uccelab.tac:8445/
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9

<#root>

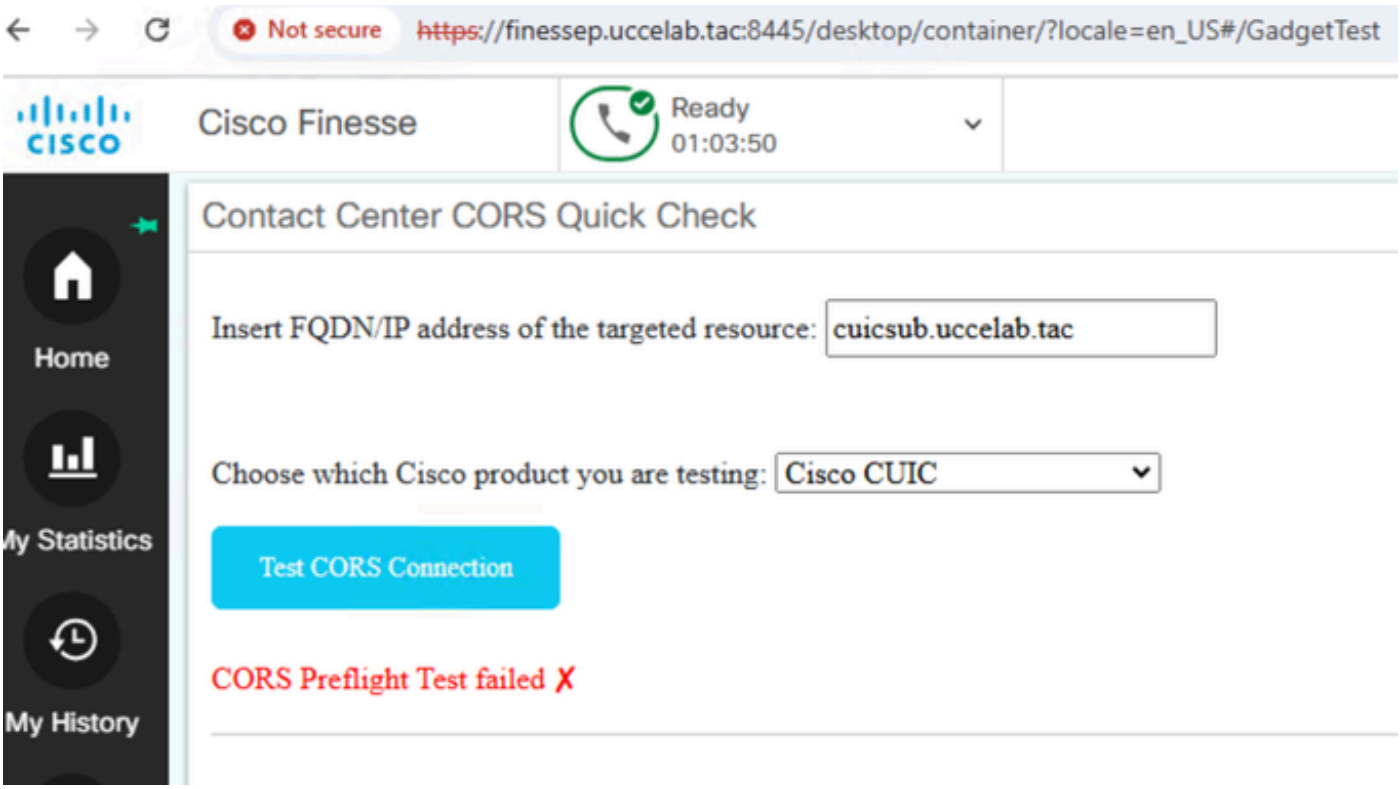
HTTP/1.1 200

server: nginx
date: Sat, 08 Feb 2025 01:27:47 GMT
content-length: 0
strict-transport-security: max-age=31536000; includeSubDomains
set-cookie: JSESSIONID=bE73993C4A7C1Fc1b33A7AaF897B8428; Path=/cuic; Secure; HttpOnly; SameSite=Strict
pragma: No-cache
cache-control: no-cache
expires: Thu, 01 Jan 1970 00:00:00 GMT
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
x-content-type-options: nosniff
content-security-policy: default-src 'self' ; script-src 'self' data: 'unsafe-inline' 'unsafe-eval' ; s
vary: origin,access-control-request-method,Access-Control-Request-Headers

access-control-allow-origin: https://finessep.uccelab.tac:8445

access-control-allow-credentials: true
access-control-expose-headers: access-control-allow-origin,access-control-allow-credentials,access-cont
access-control-max-age: 600
access-control-allow-methods: DELETE,POST,GET,OPTIONS,PUT
access-control-allow-headers: referer,peripheralid,origin,access-control-request-method,locale,accept,a
allow: GET,POST,OPTIONS,PUT,DELETE

이 시나리오에서 툴은 작동하지 않는 컨피그레이션을 보여줍니다. 앞의 예와 달리 Finesse 서버는 CUIC 서버에서 가입자로 구성되지 않습니다. 대신 CUIC 게시자에서만 구성됩니다. 그 결과 CORS 프리플라이트 요청이 실패하고 CUIC 서버가 HTTP 403(Forbidden) 오류로 응답합니다.



<#root>

OPTIONS https://cuicsub.uccelab.tac/cuic/ HTTP/1.1

Accept: */*
Access-Control-Request-Method: OPTIONS

Origin: https://finessep.uccelab.tac:8445

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome..
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Sec-Fetch-Dest: empty
Referer: https://finessep.uccelab.tac:8445/
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9

<#root>

HTTP/1.1 403

server: nginx
date: Sat, 08 Feb 2025 01:54:52 GMT
content-type: text/html;charset=utf-8
content-length: 2143
strict-transport-security: max-age=31536000; includeSubDomains
set-cookie: JSESSIONID=1C7606841B83d7847486c3d18D31cEfD; Path=/cuic; Secure; HttpOnly; SameSite=Strict
pragma: No-cache
cache-control: no-cache
expires: Thu, 01 Jan 1970 00:00:00 GMT
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
x-content-type-options: nosniff

CUIC 가입자 CLI(Command Line Interface)의 출력에서 볼 수 있듯이 Cisco Finesse는 나열되지 않습니다. 이는 Finesse가 현재 이 CUIC 서버의 가입자로 구성되어 있지 않음을 나타냅니다.

<#root>

```
admin:utils cuic cors allowed_origin list

cors_allowedorigins
===========================
1. https://finessep.uccelab.tac
2. https://finesses.uccelab.tac
3. https://finesses.uccelab.tac:8445
```

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.