FinesseのCross-Origin Resource Sharing(CORS)について

内容

はじめに

このドキュメントでは、トラブルシューティング時に基盤となるプロセスを完全に理解できるように、クロスオリジンリソース共有について詳しく説明します。

前提条件

要件

次の項目に関する知識があることが推奨されます。

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

使用するコンポーネント

このドキュメントの情報は、次のソフトウェアとハードウェアのバージョンに基づいています。

- UCCEリリース12.6.2
- Finesseリリース12.6.2
- CUICリリース12.6.2

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認して

ください。

背景説明

CORSとは

Cross-Origin Resource Sharing(CORS)は、サーバが、リソースへのアクセスを許可するWebサイト（ドメイン、プロトコル、ポート）を制御する方法です。ブラウザは通常、異なるオリジン（同じオリジンのポリシー）からの要求をブロックしますが、CORSはサーバにこの制限を選択的に緩和する機能を提供します。基本的に、サーバーは特別なHTTPヘッダーを使用して、許可されているオリジン、許可されている要求の種類（GET、POSTなど）、および含めることができるカスタムヘッダーをブラウザに伝えます。これにより、サーバはAPIにアクセスできるユーザとその方法（完全にオープンなアクセスから厳密に制限されたアクセスまで）を決定できます。CORSは、ブラウザとサーバがこれらのHTTPヘッダーを介して通信し、オリジン間の要求を管理することによって機能します。

CORSはHTTPヘッダーを使用して、制御されたクロスオリジン要求を有効にします。ブラウザとサーバはこれらのヘッダーを介して通信し、サーバは許可されたオリジン、メソッド、およびヘッダーを指定します。サーバの応答ヘッダーが欠落しているか無効な場合、ブラウザは応答をブロックし、同じオリジンポリシーを適用します。特定の要求では、ブラウザは最初にプリフライト要求をサーバに送信して、実際のクロスオリジン要求を受け入れるようにします。

ブラウザはプリフライト要求を使用して、実際の要求を送信する前に、サーバがクロスオリジン要求を許可しているかどうかを確認します。これらのプリフライト要求には、HTTPメソッドやカスタムヘッダーなどの詳細が含まれます。その後、CORS対応サーバは、実際の要求を許可または拒否して応答できます。サーバがCORS用に設定されていない場合は、プリフライトに正しく応答せず、ブラウザが実際の要求をブロックするため、不要なクロスオリジンアクセスからサーバが保護されます。

Webのセキュリティと機能には、Cross-Origin Resource Sharing(CORS)が重要です。異なるオリジン（ドメイン、プロトコル、ポート）からのリソースへの制御されたアクセスが可能になります。これは、通常はこのようなアクセスをブロックするSame-Origin Policyがブラウザによって適用されるために必要です。

CORSのライフサイクル

CORS要求は、要求を行うクライアントと要求を受信するサーバの2つの側で構成されます。クライアント側では、開発者がJavaScriptコードを書き込み、要求をサーバに送信します。サーバは、クロスオリジン要求が許可されていることを示す特別なCORS固有のヘッダーを設定して、要求に応答します。クライアントとサーバの両方が参加しないと、CORS要求は失敗します。

CORS要求のキープレーヤーは、クライアント、ブラウザ、およびサーバです。クライアントは、JSON API応答やWebページのコンテンツなど、サーバからのデータの一部を必要としています。ブラウザは、クライアントがサーバからデータにアクセスできることを確認する信頼できる仲介者として機能します。

クライアント:

クライアントはWebサイトで実行されるJavaScriptコードのスニペットで、CORS要求を開始する責任があります

注:FinesseはWebアプリケーションです。サーバにインストールされ、エージェントはWebブラウザを使用して簡単にアクセスできるため、クライアント側のインストールやプラグインやその他のソフトウェアのメンテナンスは不要です。「CORS in Action with Cisco Finesse」の例で示すように、このアーキテクチャはライブデータレポートなどの機能をサポートします。このコンテキストでは、Cisco Finesseライブデータガジェットの JavaScriptコードがクライアントとして機能し、Cisco CUICがCORSライフサイクル内のサーバとして機能します。基本的に、ブラウザベースのFinesseクライアントはCUICサーバと通信してライブデータを取得します。

クライアントとユーザ：

クライアントとユーザという用語は同じ意味で使用されることがありますが、CORSのコンテキストでは異なります。ユーザとは、このコンテキストでFinesseにアクセスするWebサイトまたはFinesseユーザ（エージェントまたはスーパーバイザ）を訪問するユーザです。クライアントとは、そのWebサイトで提供される実際のコードです。複数のユーザが同じWebサイトにアクセスし、同じJavaScriptクライアントコードを使用できます。

ブラウザ:

ブラウザはユーザエージェントとも呼ばれ、クライアント側のコードをホストします。発信リクエストに追加情報を追加し、サーバがクライアントを識別できるようにすることで、CORSで重要な役割を果たします。さらに、ブラウザはサーバの応答を解釈して、データをクライアントに配信するか、エラーを返すかを決定します。これらのブラウザ側のアクションは、同じオリジンポリシーによって提供されるセキュリティを維持するために不可欠です。ブラウザでCORSルールを適用しないと、クライアントは不正な要求を行い、この重要なセキュリティメカニズムを危険にさらす可能性があります。

[Server]：

サーバはCORS要求の宛先であり、Cisco FinesseのライブデータガジェットのCUICの例です。サーバはクライアントが必要とするデータを保存し、CORS要求が許可されるかどうかを判断するための最終的な判断を行います。

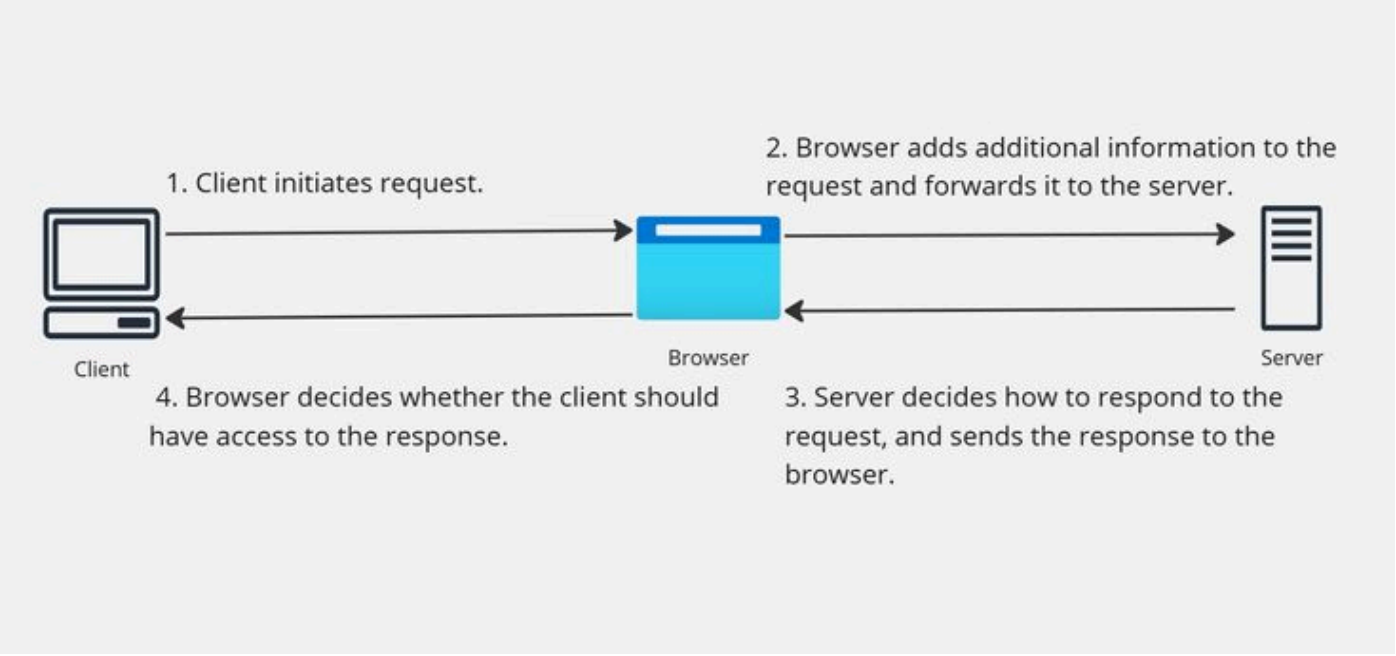
CORSリクエストに関係するユーザを確認できたので、それらがどのように連携して動作するかを見てみましょう。以降の図は、CORSのライフサイクルの概要を示しています。

1. クライアントが要求を開始します。

2. ブラウザが要求に追加情報を追加し、サーバーに転送します。

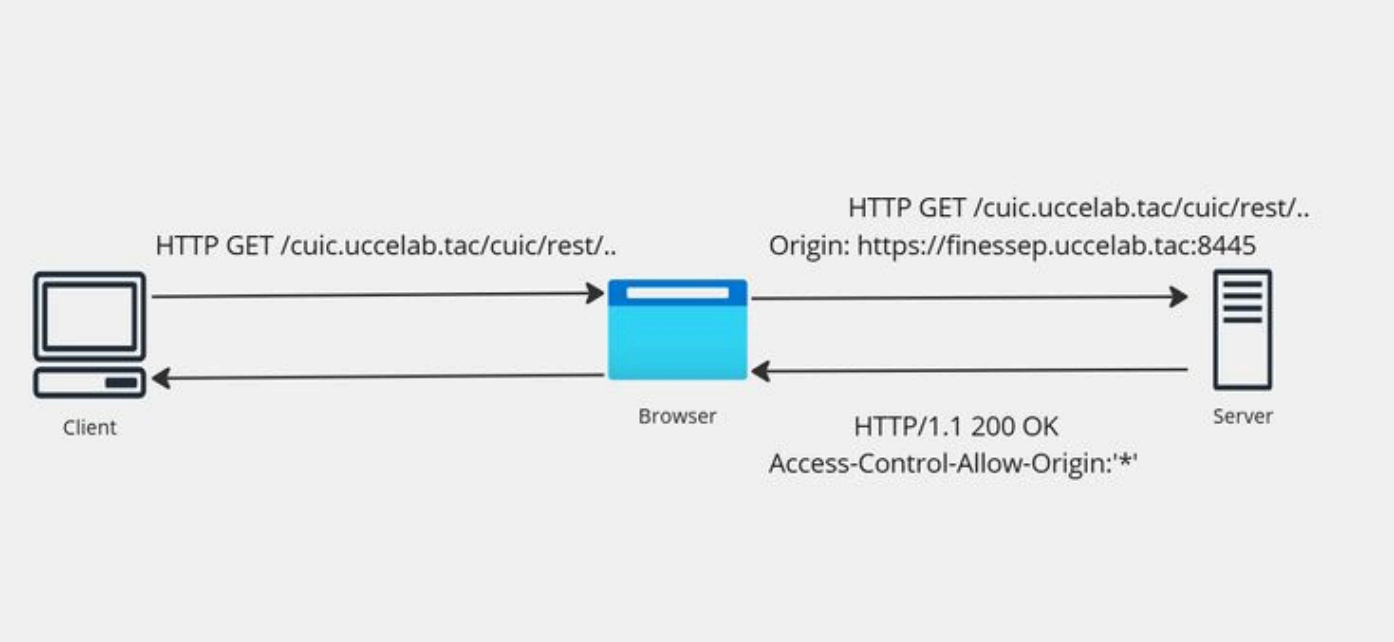
3. サーバーは要求への応答方法を決定し、ブラウザに応答を送信します。

4. ブラウザは、クライアントが応答にアクセスする必要があるかどうかを判断し、応答をクライアントに渡すか、エラーを返します。

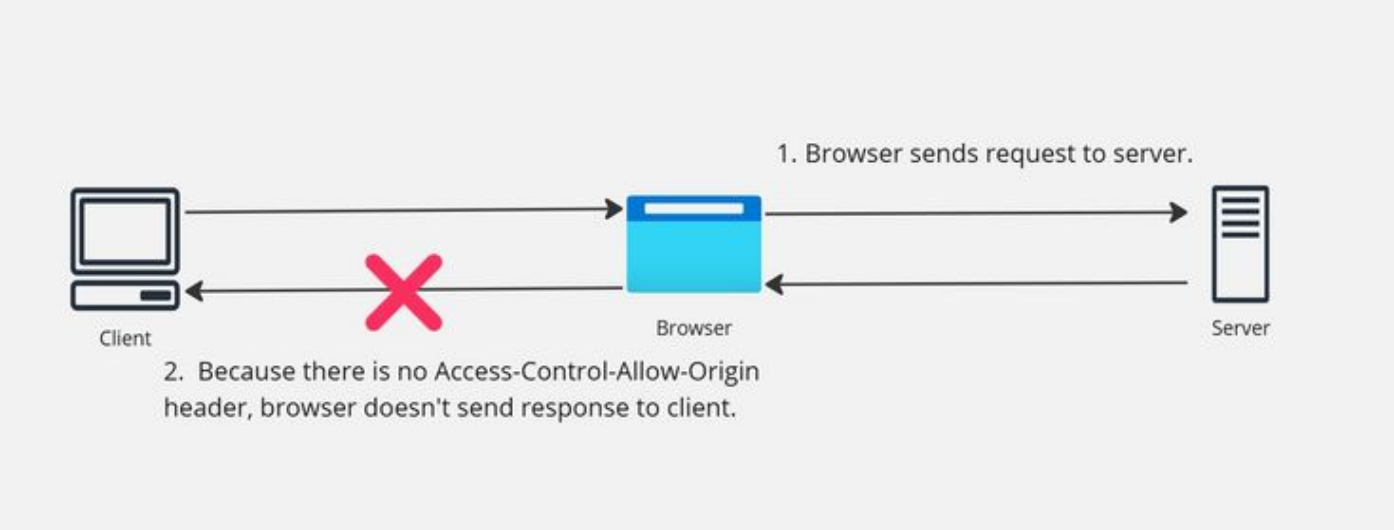


オリジン間要求を送信する前に、ブラウザは自動的にオリジンヘッダーをHTTP要求に追加します。このヘッダーは、クライアントでは変更できません。CORSの重要な部分であり、クライアントの発信元（つまり、クライアントリソースのロード元のドメイン、プロトコル、ポート）を識別するために使用されます。 このセキュリティ対策は、クライアントが他のオリジンを偽装することを防ぎます。オリジンヘッダーはCORSの基本であり、クライアントがサーバにオリジンヘッダーの送信元を通知する方法です。

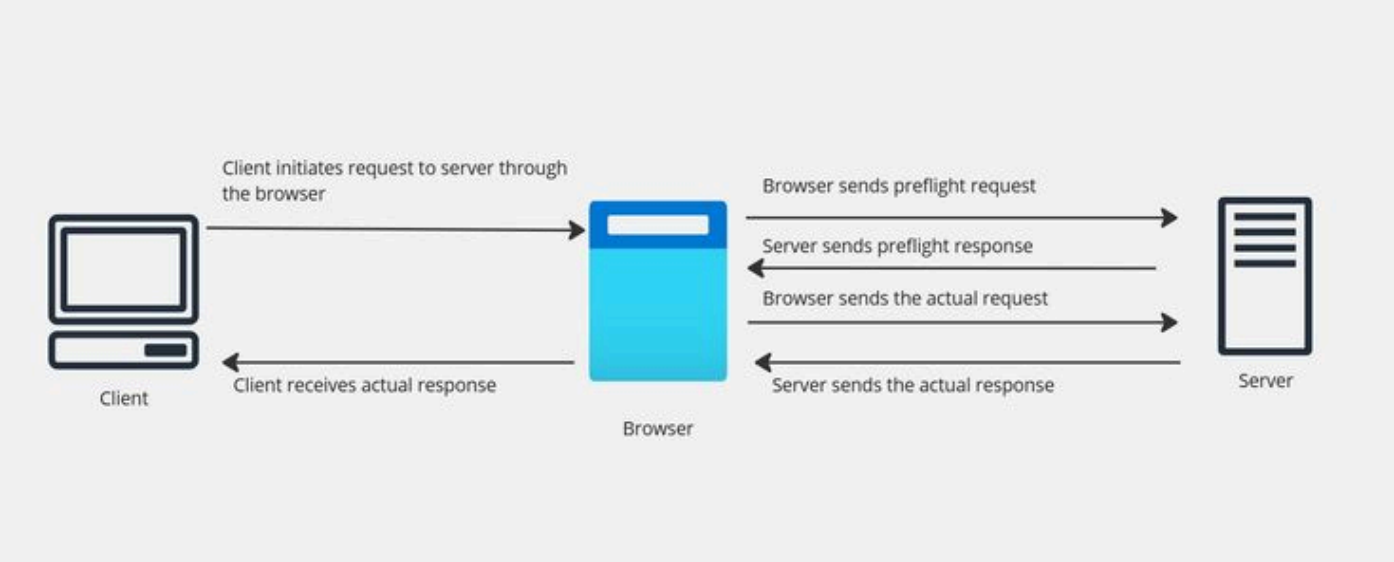
Cross-Origin Resource Sharing(CORS)インタラクションでは、クライアントのオリジンは、最初の要求のオリジンヘッダーによって識別されます。次に、サーバはその応答でAccess-Control-Allow-Originヘッダーを使用して、クライアントが要求されたリソースへのアクセスを許可されているかどうかを示します。この応答ヘッダーは非常に重要です。存在しない場合、CORS要求は失敗します。Access-Control-Allow-Originヘッダーには、任意の送信元からのアクセスを許可するワイルドカード(*)、または特定の送信元からのアクセスを許可してその特定のクライアントにのみアクセスを許可するワイルドカードを含めることができます。図にはAccess-Control-Allow-Origin: *が示されていますが、これはCUICがすべての起点(origin)を許可することを意味し、通常はCUICが実際のシナリオでこのヘッダーを特定の起点(origin)で送信します。



ブラウザがCORS要求を拒否した場合、クライアントはサーバの応答に関する情報を受信していないことを意味します。クライアントは、エラーが発生したことだけを認識しますが、特定の問題に関する詳細情報はありません。これにより、CORSエラーを他のタイプのエラーと区別することが困難になるため、CORSエラーのデバッグが困難になる可能性があります。最初の要求がサーバに送信されても、サーバの応答に有効なAccess-Control-Allow-Originヘッダーがない場合、ブラウザは応答をブロックし、クライアント側でエラーをトリガーします。これにより、クライアントはサーバの詳細な応答を見ることができなくなります。



この図は、特定のタイプのクロスオリジン要求を処理するために不可欠なプリフライトステージに特に焦点を当てて、CORSプロセス全体を説明しています。

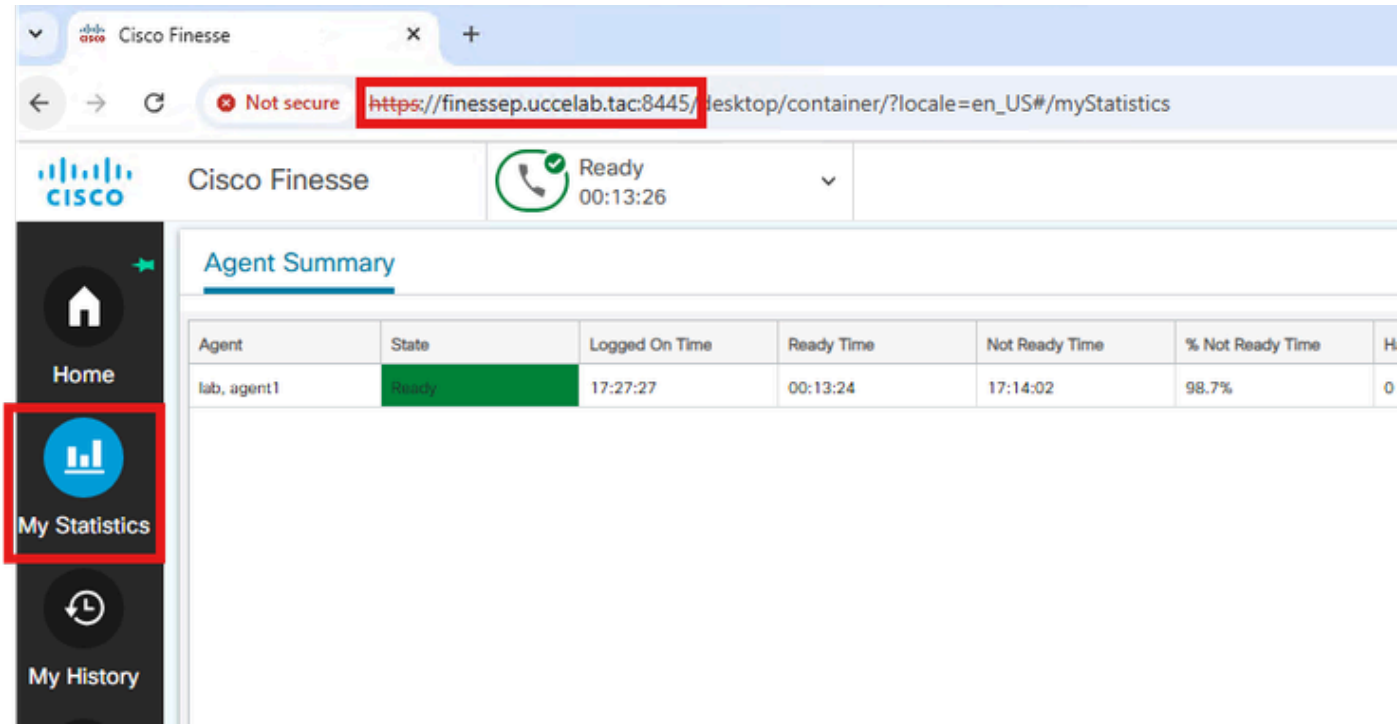


Cisco Finesseと連携したCORS

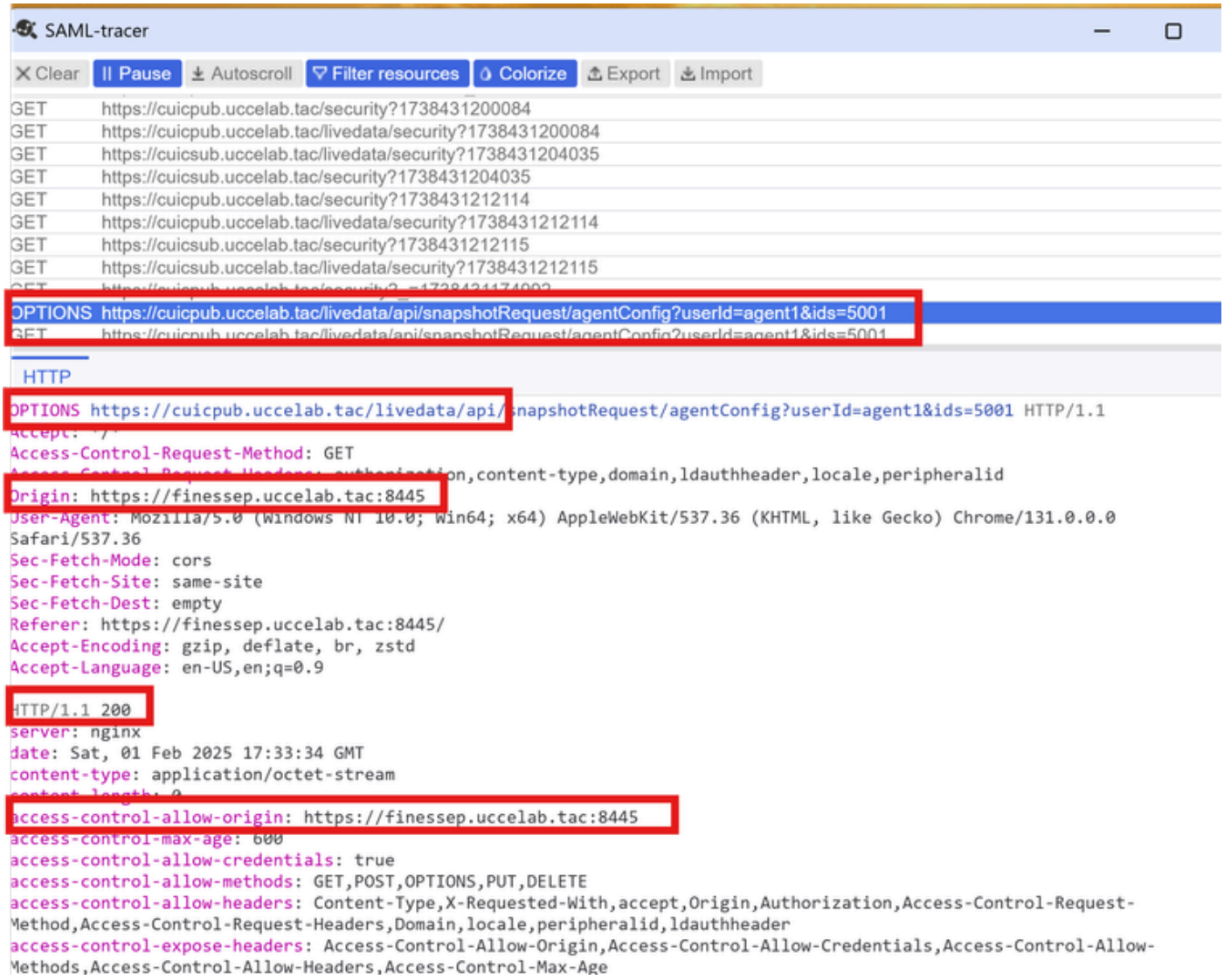
例：ライブデータガジェットによるCORS動作の分析

このセクションでは、コンタクトセンターでCisco Finesseを使用したCross-Origin Resource Sharing(CORS)の一般的な使用方法について説明します。エージェントとスーパーバイザは、一般にCisco Finesseを使用してリアルタイムデータレポートにアクセスします（図の例を参照）。

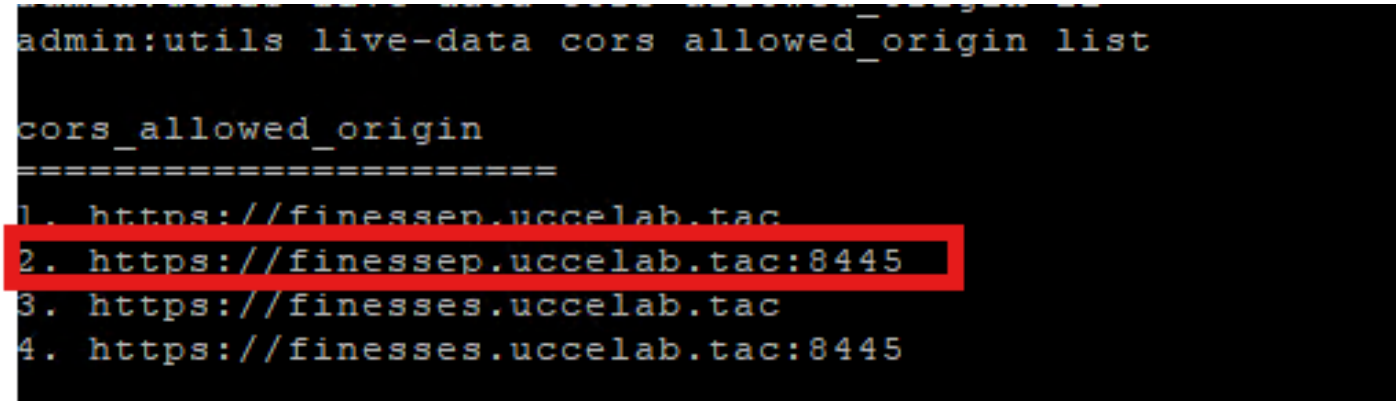
エージェントまたはスーパーバイザがレポートガジェットをクリックすると、アクションによってデータ取得要求が開始されます。この要求は、GETメソッドを使用して、Finesseアプリケーションの（クライアントとして動作する）JavaScriptコードからCUIC/Live Dataサーバに送信されます。SAMLトレーサの画像に示されているように、ブラウザは最初にプリフライト要求をサーバに送信します（前述のCORSライフサイクル）。

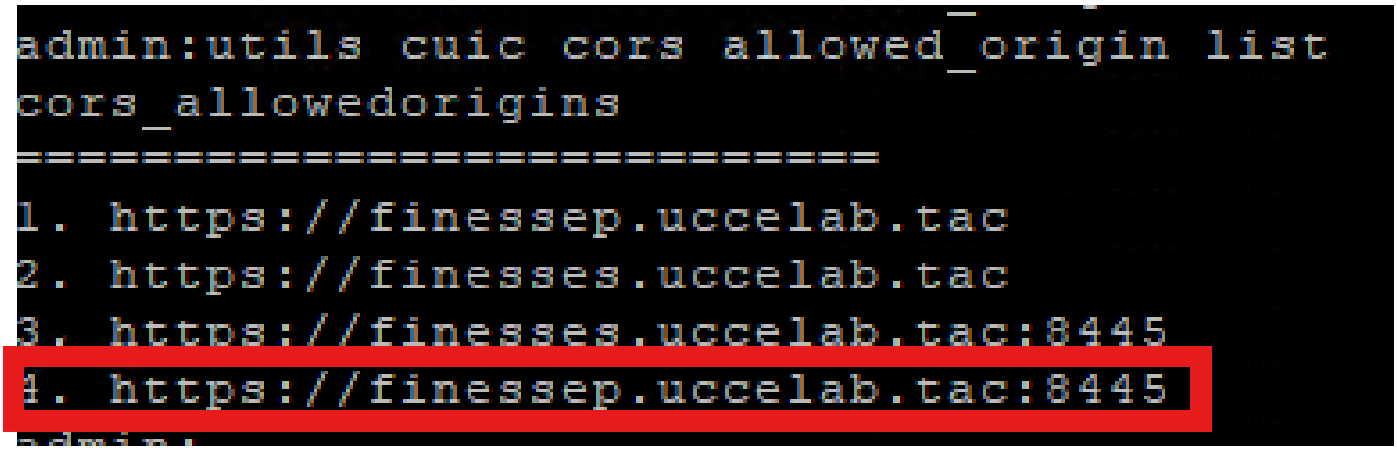


HTTP OPTIONS要求（プリフライト要求）がCUIC/ライブデータサーバに送信されます。この要求は、送信元をポート8445を含むFinesseサーバの完全修飾ドメイン名(FQDN)として指定します。これは、エージェントがCisco Finesseアプリケーションにアクセスするために使用するのと同じアドレスおよびポートです。



CUIC/ライブデータサーバのコマンドラインインターフェイス(CLI)コマンドは、ライブデータリソースへのアクセスを許可するオリジンを制御します。Finesseサーバの送信元（FQDNとポート）がこれらの設定で設定されている場合、エージェントはFinesse内でライブデータガジェットの詳細を表示できます。





CORS接続テスト用のTACツール

サーバ側のCORSの設定ミスは、Cisco Finesseのサードパーティ製ガジェットやライブデータガジェットで問題を引き起こす可能性があります。この記事では、CORSクイックチェックガジェットへのリンクを提供します。トラブルシューティングツールは、ライブデータ表示やその他のサードパーティ統合など、Finesseガジェットに影響を与えるオリジン間リソース共有の問題を診断するために設計されています。

技術的には、このガジェットは、Cisco Finesseクライアントから指定されたターゲットリソースにプリフライト要求を送信することによって機能します。このクイックチェック機能は、CORS関連の問題を迅速に特定して解決するのに役立ち、トラブルシューティングプロセスを高速化します。

Finesseデスクトップ内でContact Center CORS Quick Check 12.6-v1.0ガジェットを展開するには、次の手順を実行します。

1. Contact Center CORS Quick Check 12.6-v1.0 folder.2からガジェットのファイルをダウンロードします。

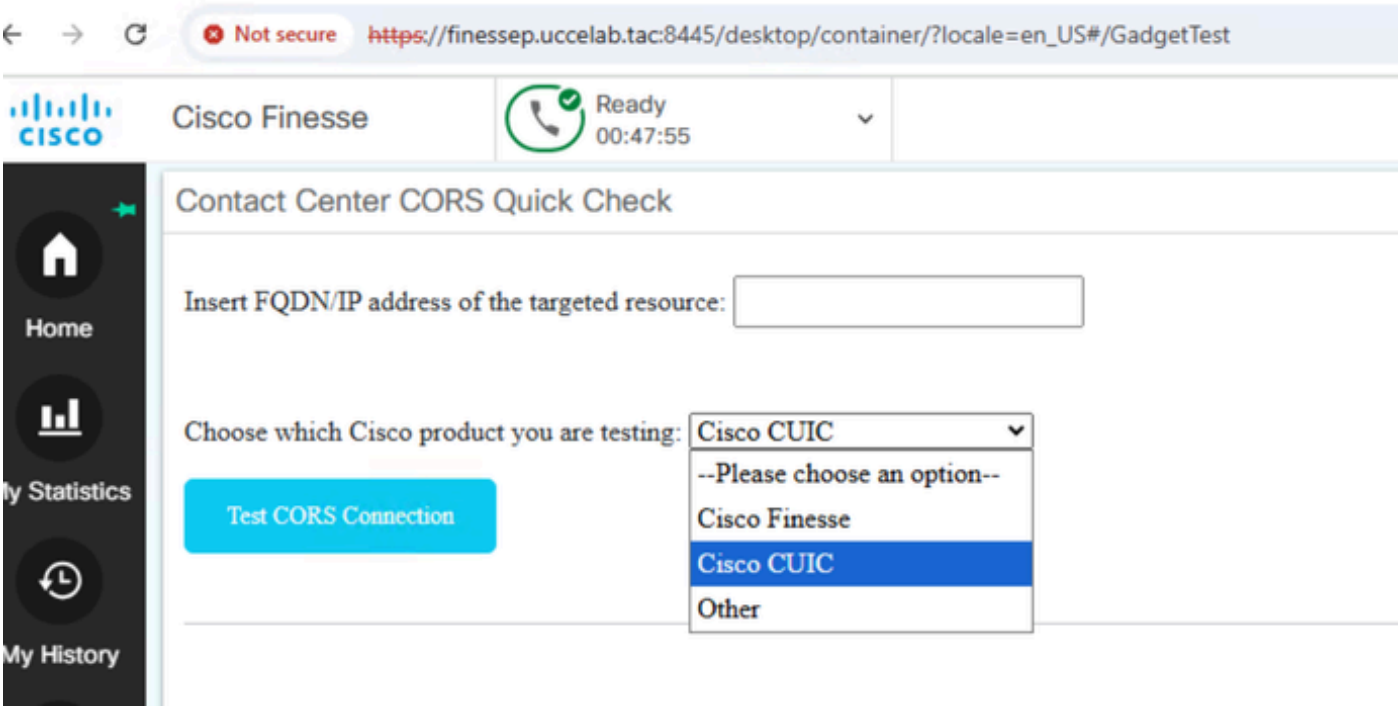
2. コンタクトセンターのCORS Quick Check 12.6-v1.0フォルダの内容を、Finesseインストール内の3rdpartygadgetディレクトリにコピーします。

3. Finesseデスクトップレイアウトで、ガジェットを目的のユーザロール（エージェント、スーパーバイザなど）に追加します。次のXMLの例は、このガジェットを追加するための正しい設定を示しています。

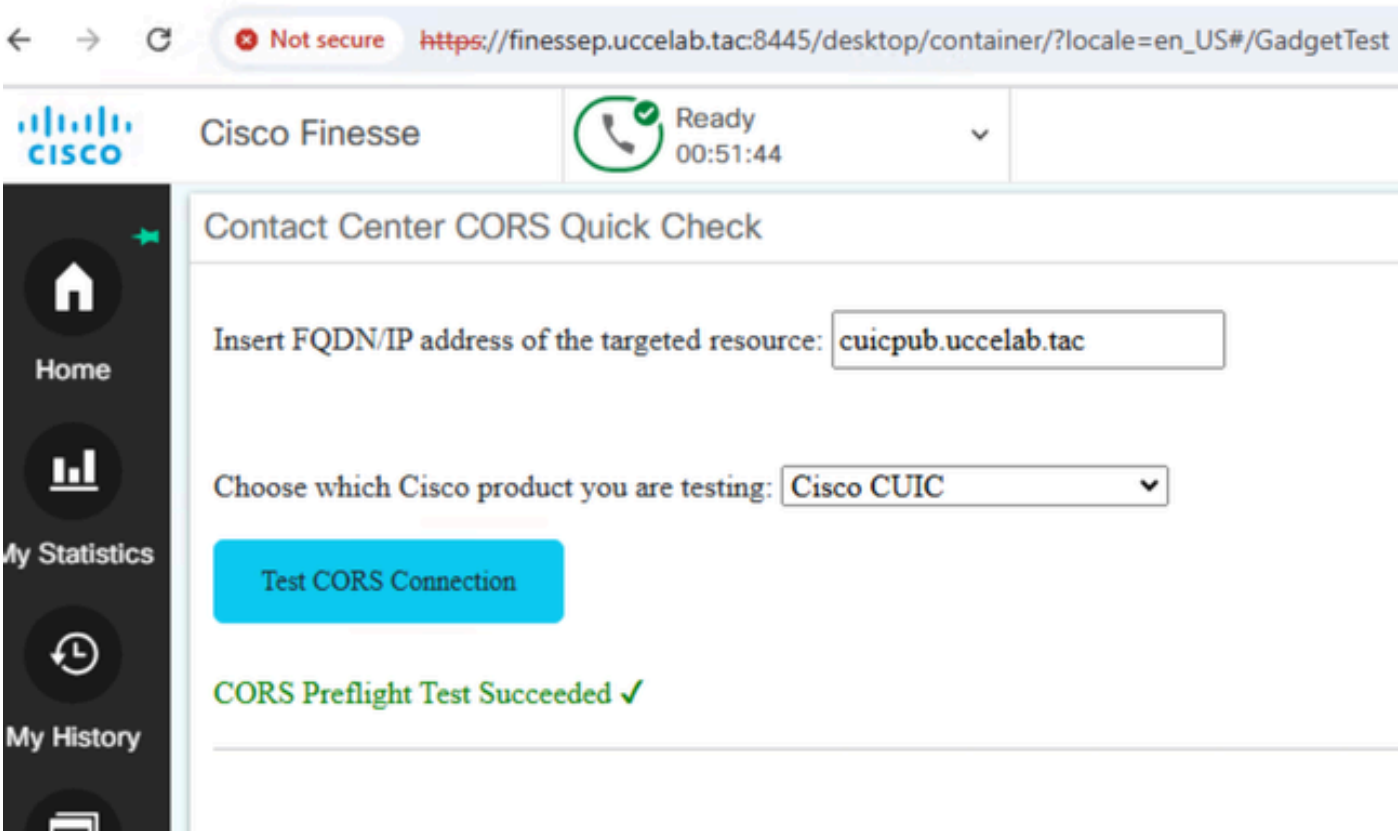
```
<gadget>/3rdpartygadget/files/TestCORSgadget.xml</gadget>
```

サードパーティ製ガジェットのアップロードとデスクトップへの追加の詳細については、『Finesse開発者ガイド』の「サードパーティ製ガジェット」の章、および『Finesse管理ガイド』の「サードパーティ製ガジェットの管理」の章を参照してください。

ガジェットファイルがアップロードされ、Cisco Finesse Tomcatサービスが再起動されると、ガジェットが使用可能になり、グラフィカルユーザインターフェイス(GUI)が表示されます。



上部のドロップダウンリストからCUICを選択できます。表示されたフィールドに、CUICサーバの完全修飾ドメイン名(FQDN)を入力します。テストが成功すると、次のようになります。



テストに成功すると、CUICサーバでFinesseサーバとのCross-Origin Resource Sharing(CORS)が正しく設定されます。ブラウザのSAMLトレーサログに、HTTP OPTIONS要求（CORSプリフライト）がCUICサーバに送信されたことが示されます。この要求には、オリジンヘッダーにFinesseサーバのアドレスが含まれています。CUICサーバは200 OK HTTPメッセージで応答しました。重要なのは、応答内のAccess-Control-Allow-Originヘッダーにもfinesseサーバのアドレス

が含まれていることです。これにより、CUICサーバがFinesseサーバの送信元からの要求を許可するように設定されていることを確認し、CORSが正しく設定されていることを確認できます。

<#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>

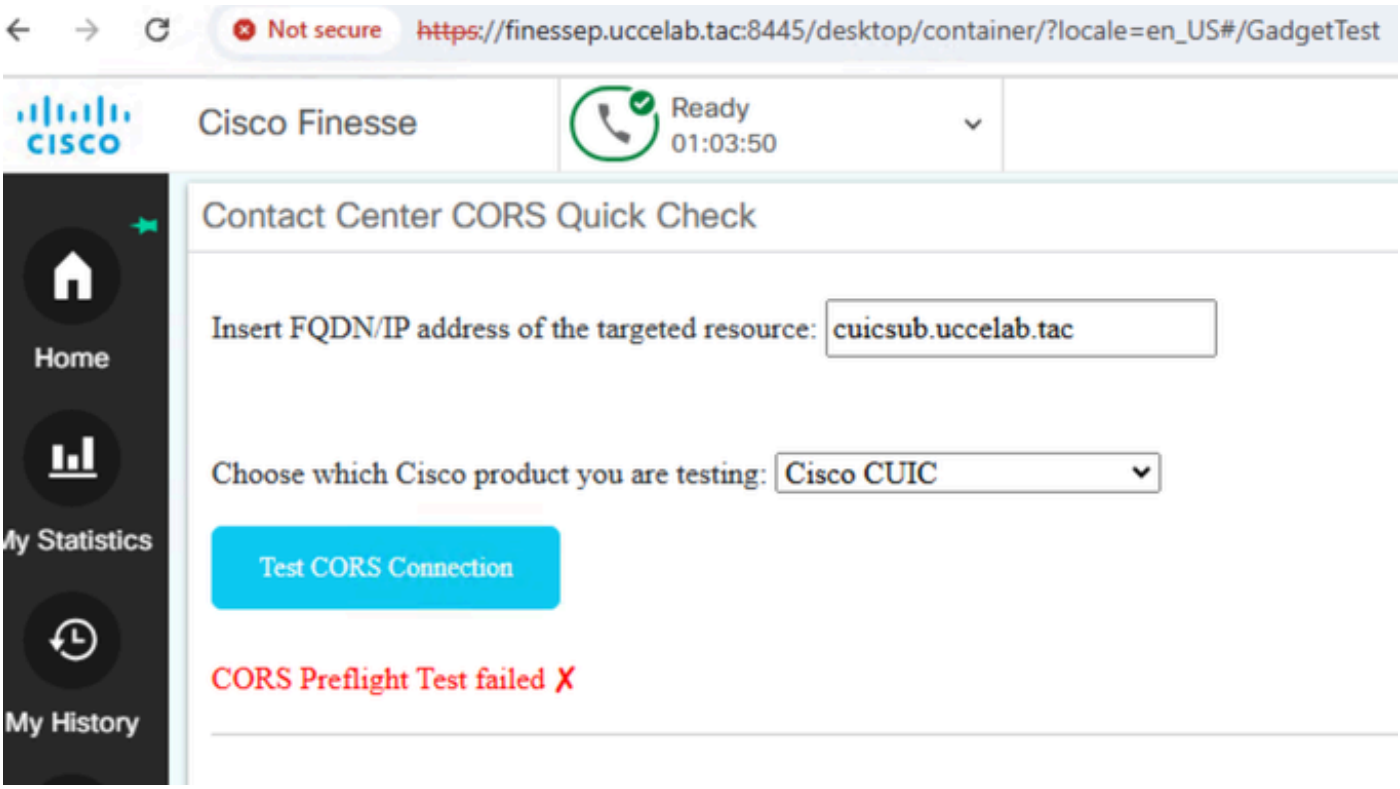
```
HTTP/1.1 200

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
```

このシナリオでは、ツールは動作しない設定を示します。前の例とは異なり、Finesseサーバは CUICサーバのサブスクライバとして設定されていません。代わりに、CUICパブリッシャでのみ設定されます。その結果、CORSプリフライト要求が失敗し、CUICサーバがHTTP 403(Forbidden)エラーで応答します。



<#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サブスクライバのコマンドラインインターフェイス(CLI)の出力からわかるように、Cisco Finesseはリストされていません。これは、FinesseがこのCUICサーバのサブスクライバとして現在設定されていないことを示します。

<#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
```

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。