Comprendre le partage de ressources entre origines (CORS) pour Finesse

Table des matières

Introduction

Ce document décrit complètement le partage de ressources entre origines afin que, pendant le dépannage, les processus sous-jacents soient parfaitement compris.

Conditions préalables

Exigences

Cisco vous recommande de prendre connaissance des rubriques suivantes :

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

Composants utilisés

Les informations contenues dans ce document sont basées sur les versions de matériel et de logiciel suivantes :

- UCCE version 12.6.2
- Finesse version 12.6.2
- CUIC version 12.6.2

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. Si votre réseau

est en ligne, assurez-vous de bien comprendre l’incidence possible des commandes.

Informations générales

Qu'est-ce que CORS

Le partage de ressources d'origine croisée (CORS) permet aux serveurs de contrôler les sites Web (domaines, protocoles et ports) autorisés à accéder à leurs ressources. Alors que les navigateurs bloquent normalement les requêtes provenant de différentes origines (la politique de même origine), CORS donne aux serveurs le pouvoir d'assouplir sélectivement cette restriction. Les serveurs utilisent essentiellement des en-têtes HTTP spéciaux pour indiquer au navigateur quelles origines sont autorisées, quels types de requêtes sont autorisés (comme GET, POST, etc.) et quels en-têtes personnalisés peuvent être inclus. Cela permet aux serveurs de décider qui peut accéder à leurs API et comment, allant d'un accès complètement ouvert à un accès strictement limité. CORS fonctionne en faisant communiquer le navigateur et le serveur via ces en-têtes HTTP pour gérer les requêtes inter-origines.

CORS utilise des en-têtes HTTP pour activer les requêtes d'origine croisée contrôlées. Le navigateur et le serveur communiquent via ces en-têtes, le serveur spécifiant les origines, méthodes et en-têtes autorisés. Si les en-têtes de réponse du serveur sont manquants ou non valides, le navigateur bloque la réponse, en appliquant la stratégie de même origine. Pour certaines requêtes, le navigateur envoie d'abord une requête de contrôle en amont au serveur pour s'assurer qu'il accepte la requête d'origine croisée réelle.

Les navigateurs utilisent les requêtes de contrôle en amont pour vérifier si un serveur autorise une requête entre origines avant d'envoyer la vraie requête. Ces demandes de contrôle en amont incluent des détails tels que la méthode HTTP et les en-têtes personnalisés. Les serveurs compatibles CORS peuvent alors répondre, soit en autorisant, soit en refusant la requête proprement dite. Si un serveur n'est pas configuré pour CORS, il ne répond pas correctement au contrôle en amont et le navigateur bloque la requête réelle, protégeant ainsi le serveur contre les accès non désirés entre origines.

Le partage de ressources inter-origines (CORS) est essentiel pour la sécurité et la fonctionnalité Web. Il permet un accès contrôlé aux ressources de différentes origines (domaines, protocoles, ports), ce qui est nécessaire car les navigateurs appliquent une politique de même origine qui bloque normalement cet accès.

Cycle de vie d'un CORS

Une requête CORS se compose de deux parties : le client qui fait la demande et le serveur qui la reçoit. Côté client, le développeur écrit du code JavaScript pour envoyer la requête au serveur. Le serveur répond à la demande en définissant des en-têtes spécifiques à CORS pour indiquer que la demande d'origine croisée est autorisée. Sans la participation du client et du serveur, la requête CORS échoue.

Les principaux acteurs d’une requête CORS sont le client, le navigateur et le serveur. Le client souhaite obtenir des données du serveur, telles qu'une réponse JSON API ou le contenu d'une

page Web. Le navigateur agit en tant qu'intermédiaire de confiance pour vérifier que le client peut accéder aux données à partir du serveur.

Client :

Le client est un extrait de code JavaScript exécuté sur un site Web et il est chargé d'initier la requête CORS

Remarque : Finesse est une application Web. Il est installé sur un serveur et les agents y accèdent simplement à l'aide de leurs navigateurs Web, ce qui élimine le besoin d'installations côté client ou de maintenance de plugins ou d'autres logiciels. Comme l'illustre l'exemple de CORS in Action avec Cisco Finesse, cette architecture prend en charge des fonctionnalités telles que les rapports de données en direct. Dans ce contexte, le code JavaScript du gadget de données en direct Cisco Finesse agit en tant que client, tandis que Cisco CUIC sert de serveur dans le cycle de vie CORS. Le client Finesse basé sur navigateur interagit essentiellement avec le serveur CUIC pour récupérer des données en direct.

Client et utilisateur :

Parfois, les mots client et utilisateur sont utilisés de façon interchangeable, mais ils sont différents dans le contexte du SCRO. Un utilisateur est une personne visitant un site Web ou un utilisateur Finesse (agent ou superviseur) accédant à Finesse dans ce contexte, alors qu'un client est le code réel desservi par ce site Web. Plusieurs utilisateurs peuvent visiter le même site Web et recevoir le même code client JavaScript.

Navigateur :

Le navigateur, également appelé agent utilisateur, héberge le code côté client. Il joue un rôle crucial dans CORS en ajoutant des informations supplémentaires aux requêtes sortantes, permettant au serveur d'identifier le client. En outre, le navigateur interprète la réponse du serveur, déterminant s'il faut remettre les données au client ou renvoyer une erreur. Ces actions côté navigateur sont essentielles pour maintenir la sécurité fournie par la stratégie de même origine. Sans l'application des règles CORS par le navigateur, les clients pourraient effectuer des requêtes non autorisées, ce qui compromettrait ce mécanisme de sécurité vital.

Serveur :

Le serveur est la destination de la requête CORS, et c'est CUIC pour l'exemple de gadget de données en direct avec Cisco Finesse. Le serveur stocke les données souhaitées par le client et il a le dernier mot quant à savoir si la requête CORS est autorisée ou non.

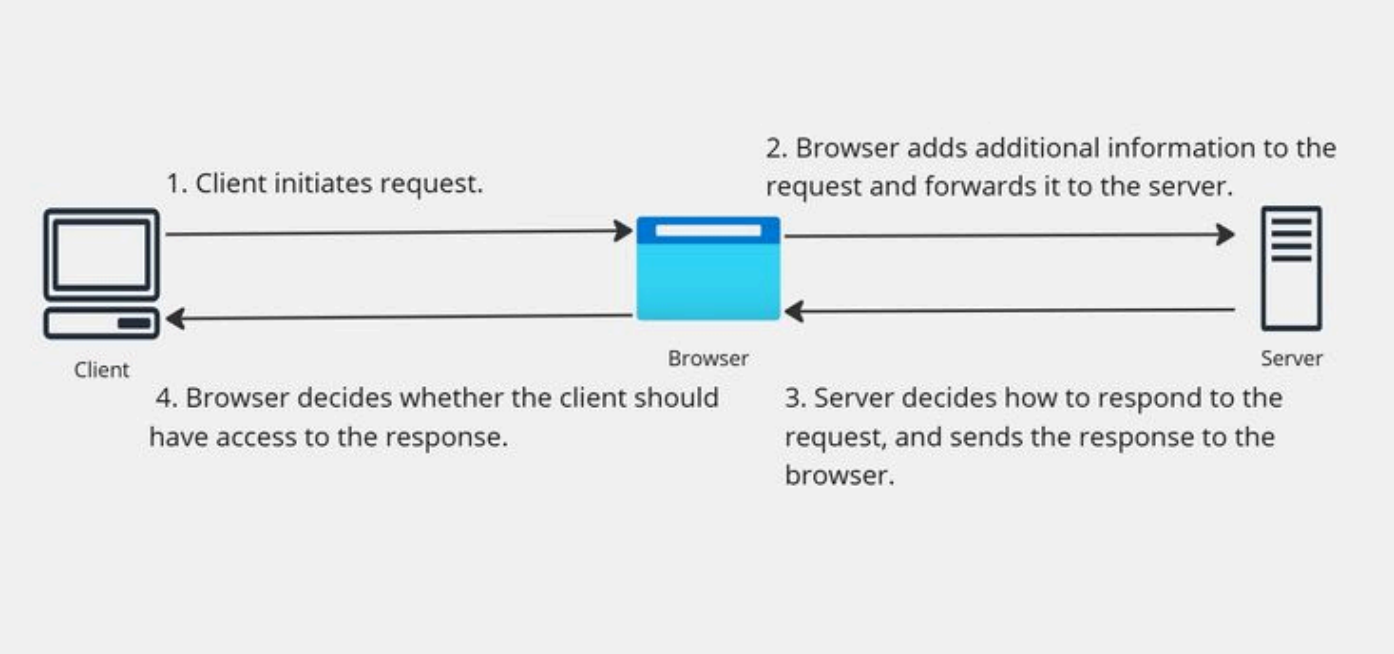
Maintenant que vous savez qui est impliqué dans une demande du SCRO, jetons un oeil à la façon dont ils fonctionnent tous ensemble. Les images suivantes illustrent le cycle de vie de haut niveau du système CORS :

1. Le client lance la requête.

2. Le navigateur ajoute des informations supplémentaires à la demande et la transmet au serveur.

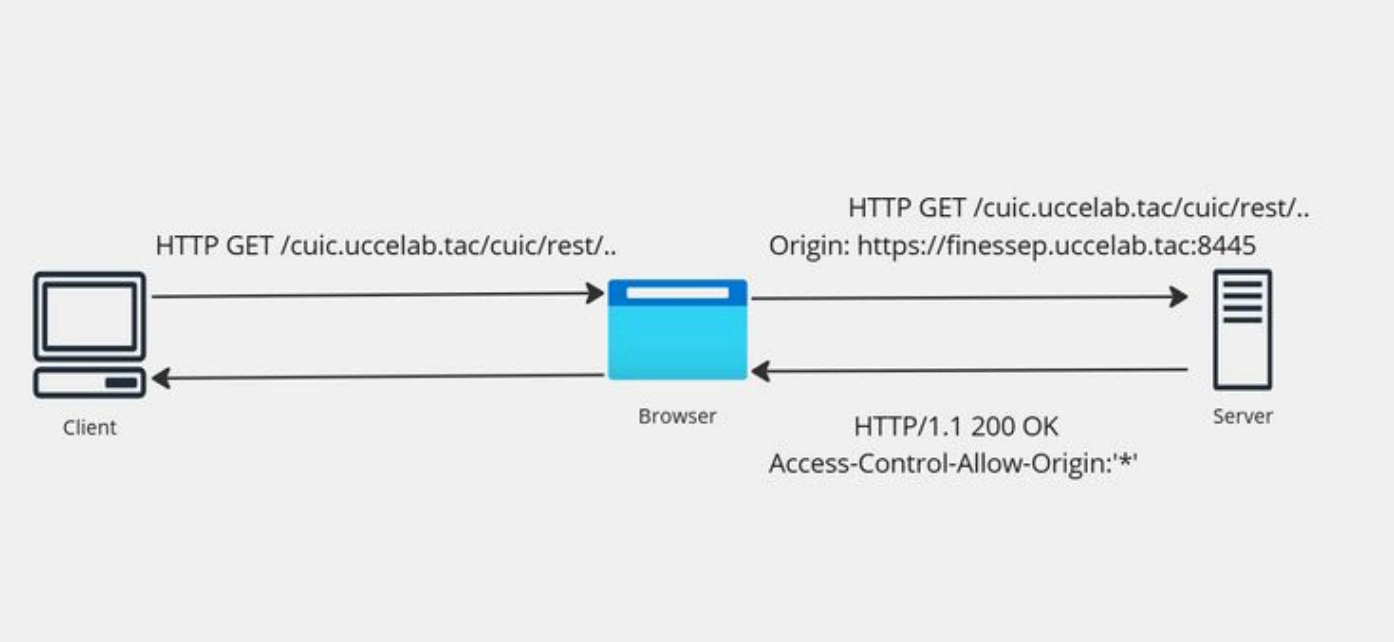
3. Le serveur décide de la manière de répondre à la demande et envoie la réponse au navigateur.

4. Le navigateur décide si le client doit avoir accès à la réponse et transmet la réponse au client ou renvoie une erreur.

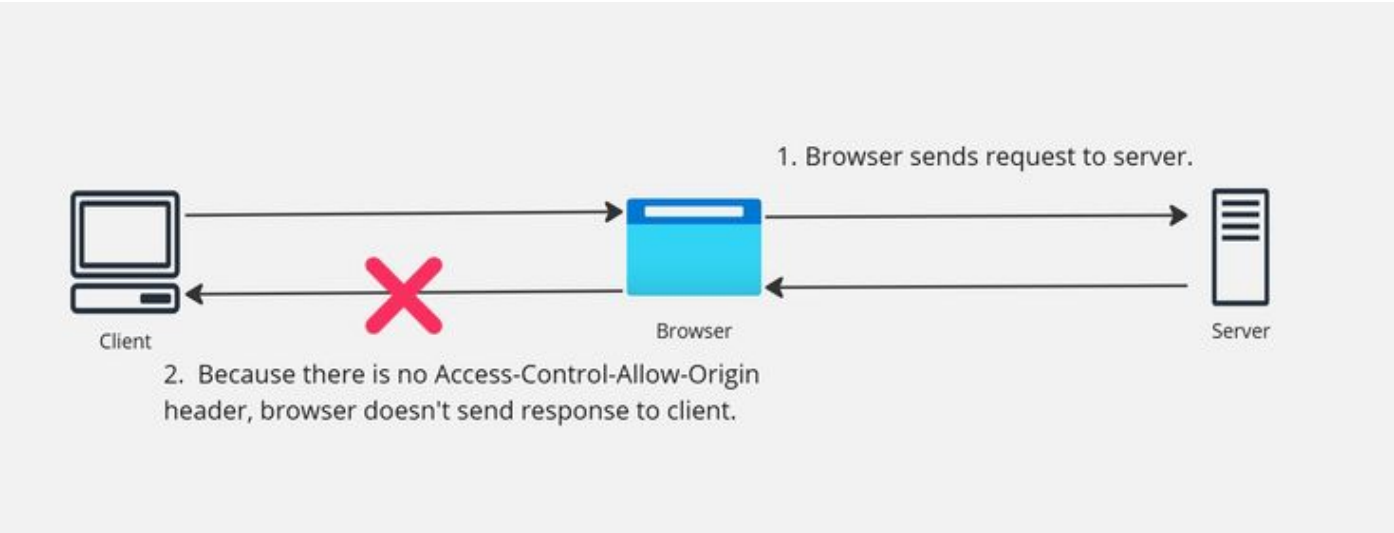


Avant d'envoyer une requête d'origine croisée, le navigateur ajoute automatiquement un en-tête d'origine à la requête HTTP. Cet en-tête, que le client ne peut pas modifier, est une partie essentielle de CORS et sert à identifier l'origine du client (c'est-à-dire le domaine, le protocole et le port à partir duquel la ressource client a été chargée). Cette mesure de sécurité empêche les clients de se faire passer pour d'autres origines. L'en-tête d'origine est fondamental pour CORS, car il indique au serveur d'où il vient.

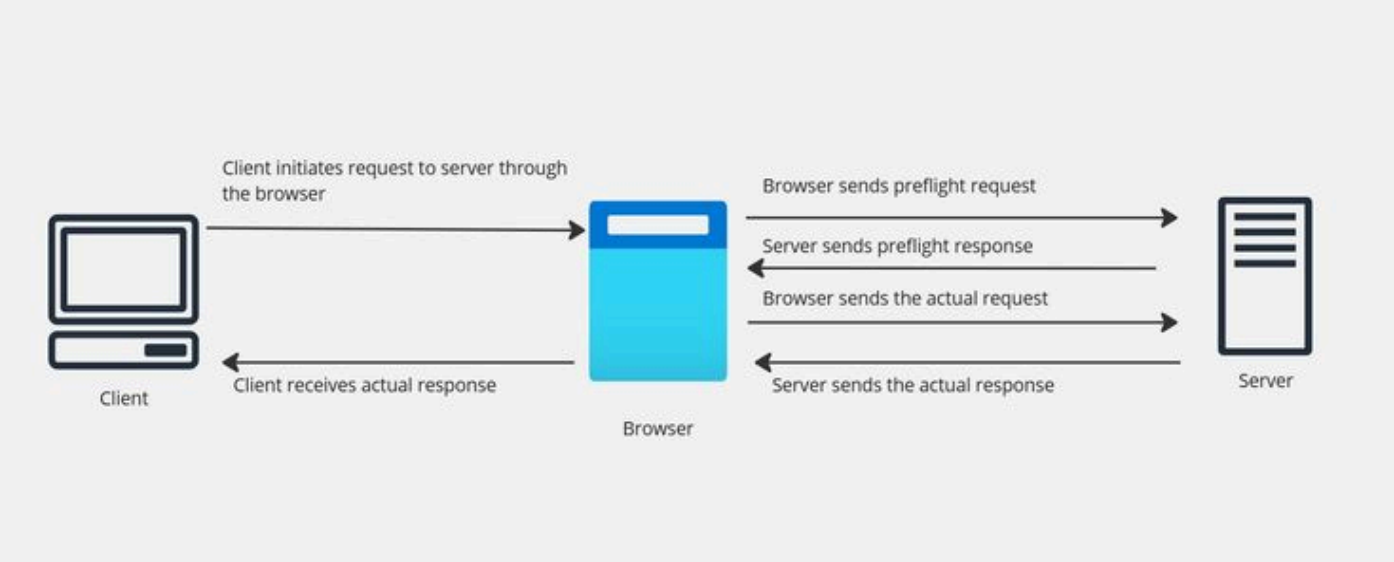
Dans une interaction CORS (Cross-Origin Resource Sharing), l'origine du client est identifiée par l'en-tête Origine dans la demande initiale. Le serveur utilise ensuite l'en-tête Access-Control-Allow-Origin dans sa réponse pour indiquer si le client est autorisé à accéder à la ressource demandée. Cet en-tête de réponse est crucial ; en cas d'absence, la requête CORS échoue. L'en-tête Access-Control-Allow-Origin peut contenir soit un caractère générique (*), autorisant l'accès à partir de n'importe quelle origine, soit une origine spécifique, accordant l'accès uniquement à ce client particulier. Alors que l'image montre Access-Control-Allow-Origin: *, ce qui implique que CUIC autorise toutes les origines, CUIC envoie généralement cet en-tête avec une origine spécifique dans des scénarios réels.



Lorsqu'un navigateur rejette une requête CORS, cela signifie que le client ne reçoit aucune information sur la réponse du serveur. Le client sait seulement qu'une erreur s'est produite, mais il manque des détails sur le problème spécifique. Cela peut rendre le débogage des erreurs CORS difficile, car il est difficile de distinguer une défaillance CORS des autres types d'erreurs. Même si la requête initiale est envoyée au serveur, si la réponse du serveur ne comporte pas d'en-tête Access-Control-Allow-Origin valide, le navigateur bloque la réponse et déclenche une erreur côté client, empêchant ainsi le client de voir la réponse détaillée du serveur.



Cette image explique l'ensemble du processus CORS, avec un accent particulier sur la phase de précontrôle, qui est essentielle pour traiter des types spécifiques de demandes d'origine croisée.

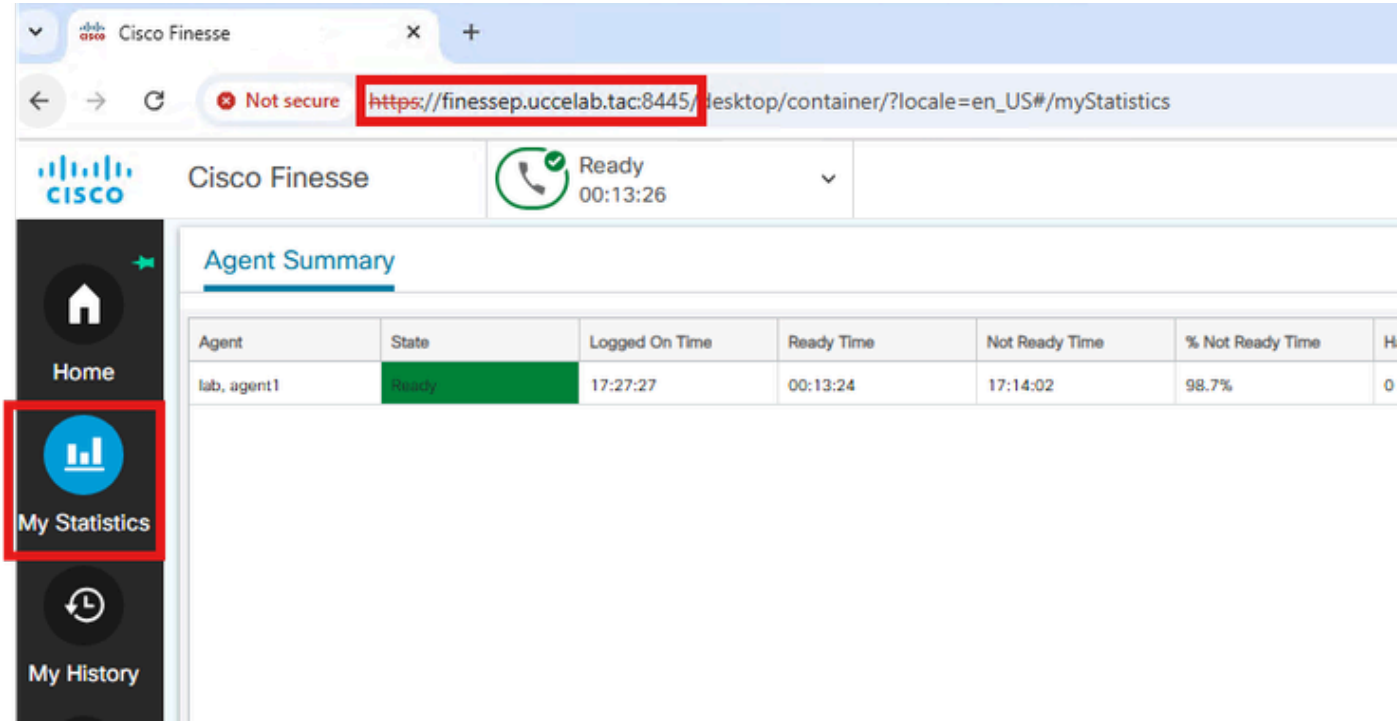


CORS en action avec Cisco Finesse

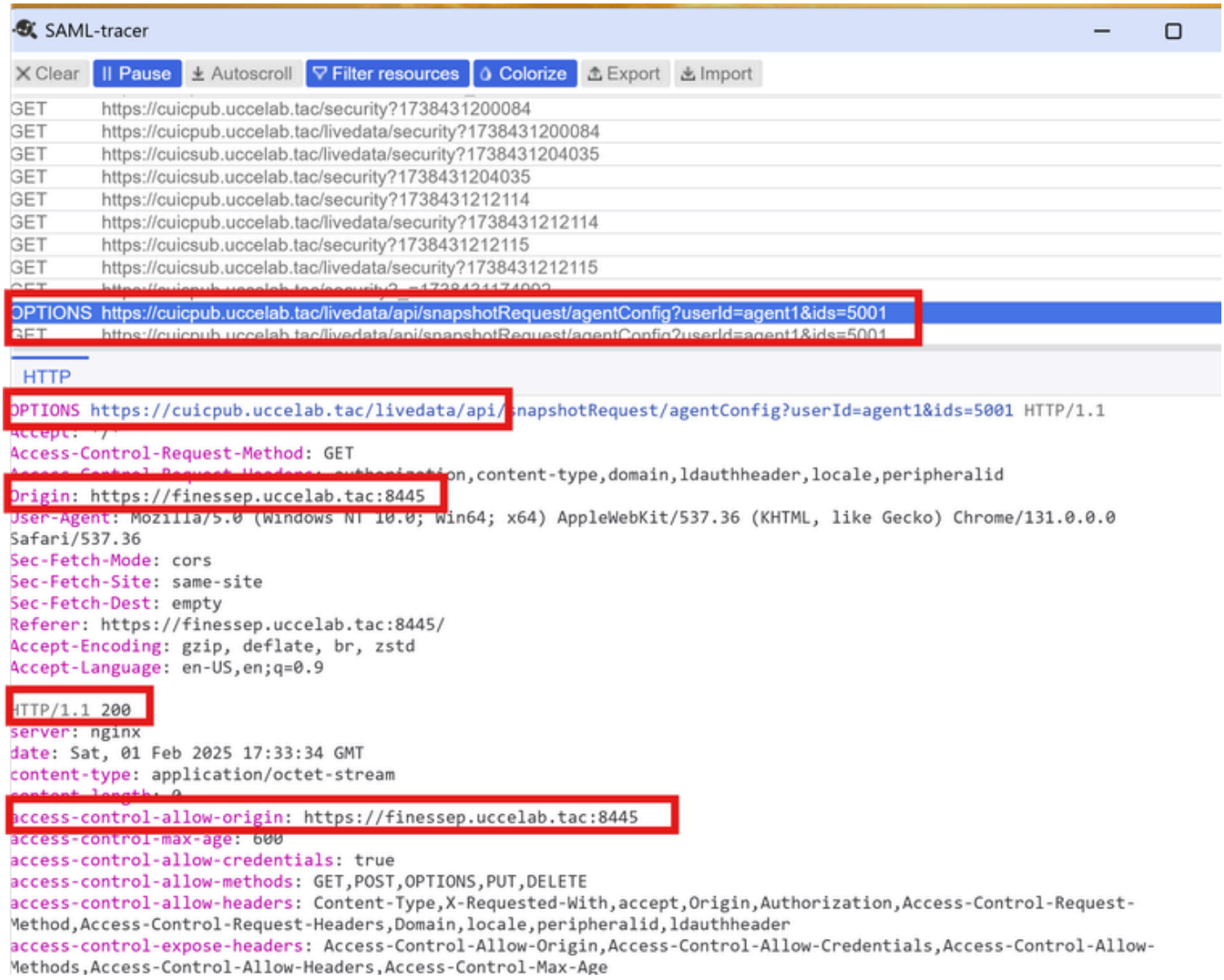
Exemple illustratif : Analyse du comportement de CORS avec le gadget Live Data

Cette section décrit l'utilisation typique du partage de ressources d'origine croisée (CORS) avec Cisco Finesse dans les centres de contact. Les agents et les superviseurs utilisent généralement Cisco Finesse pour accéder aux rapports de données en temps réel (comme illustré dans l'exemple d'image).

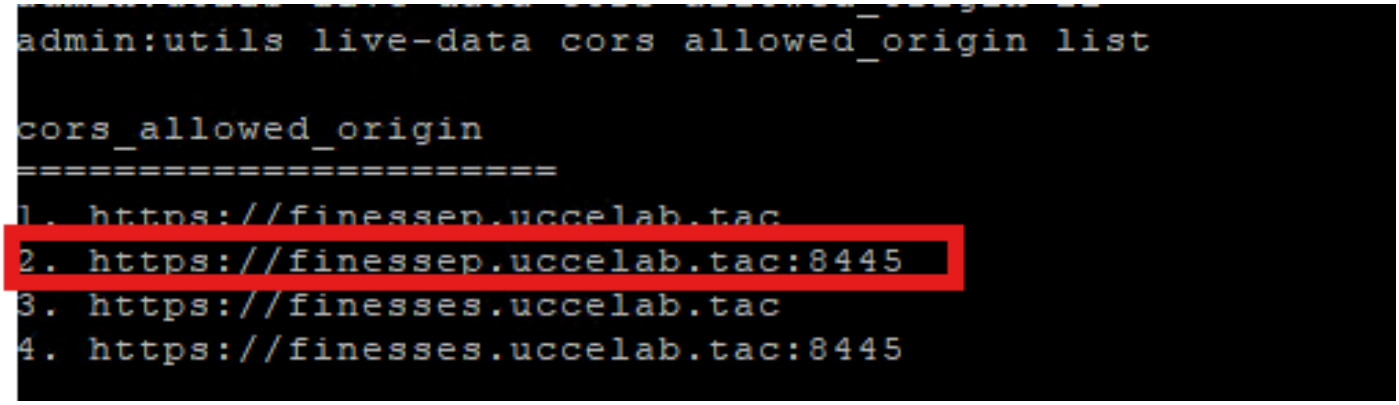
Lorsqu'un agent ou un superviseur clique sur un gadget de rapport, son action lance une demande de récupération de données. Cette requête est envoyée depuis le code JavaScript de l'application Finesse (agissant comme le client) vers le serveur CUIC/Live Data à l'aide d'une méthode GET. Comme l'illustre l'image du traceur SAML, le navigateur envoie d'abord une demande de contrôle en amont au serveur, le cycle de vie CORS décrit précédemment.

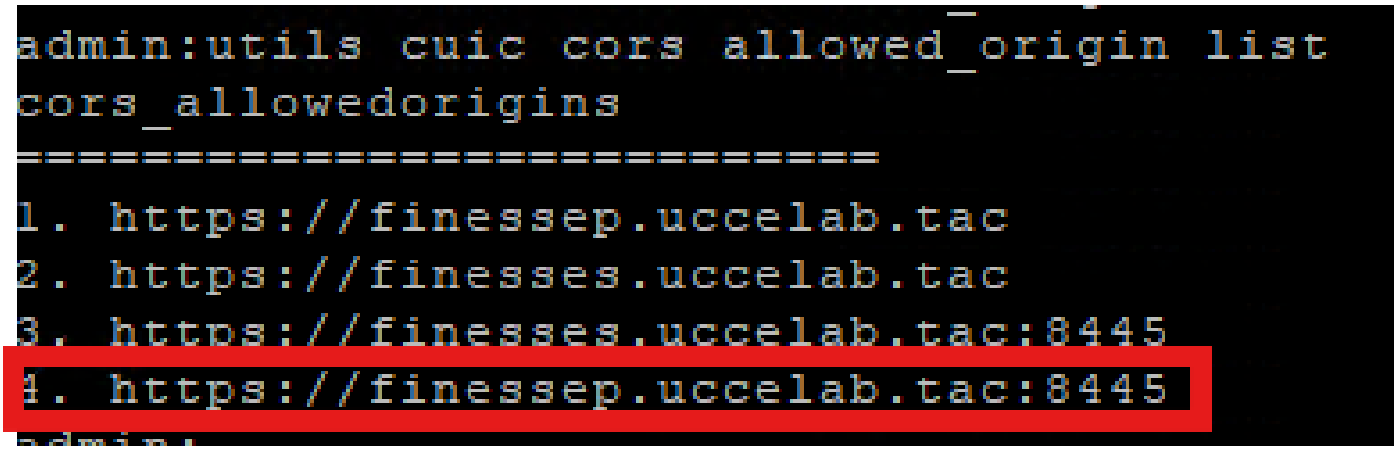


Une demande HTTP OPTIONS (la demande de contrôle en amont) est envoyée au serveur CUIC/Live Data. Cette demande spécifie l'origine comme nom de domaine complet (FQDN) du serveur Finesse, y compris le port 8445. Il s'agit de l'adresse et du port que les agents utilisent pour accéder à l'application Cisco Finesse.



Les commandes de l'interface de ligne de commande (CLI) sur le serveur CUIC/Live Data contrôlent les origines autorisées à accéder à ses ressources de données en direct. Si l'origine du serveur Finesse (son nom de domaine complet et son port) est configurée dans ces paramètres, les agents peuvent afficher les détails du gadget de données en direct dans Finesse.





Outil TAC pour les tests de connexion CORS

Les erreurs de configuration CORS côté serveur peuvent parfois entraîner des problèmes avec des gadgets de données tiers ou en direct dans Cisco Finesse. Cet article fournit un lien vers un gadget de vérification rapide de CORS, un outil de dépannage conçu pour aider à diagnostiquer les problèmes de partage de ressources entre origines affectant les gadgets Finesse, y compris les affichages de données en direct et d'autres intégrations tierces.

Techniquement, ce gadget fonctionne en envoyant des requêtes de contrôle en amont du client Cisco Finesse à une ressource cible spécifiée. Cette fonctionnalité de vérification rapide permet d'identifier et de résoudre rapidement les problèmes liés à CORS, accélérant ainsi le processus de dépannage.

Pour déployer le gadget Contact Center CORS Quick Check 12.6-v1.0 sur le bureau Finesse :

1. Téléchargez les fichiers du gadget à partir du dossier Contact Center CORS Quick Check 12.6-v1.0.2.

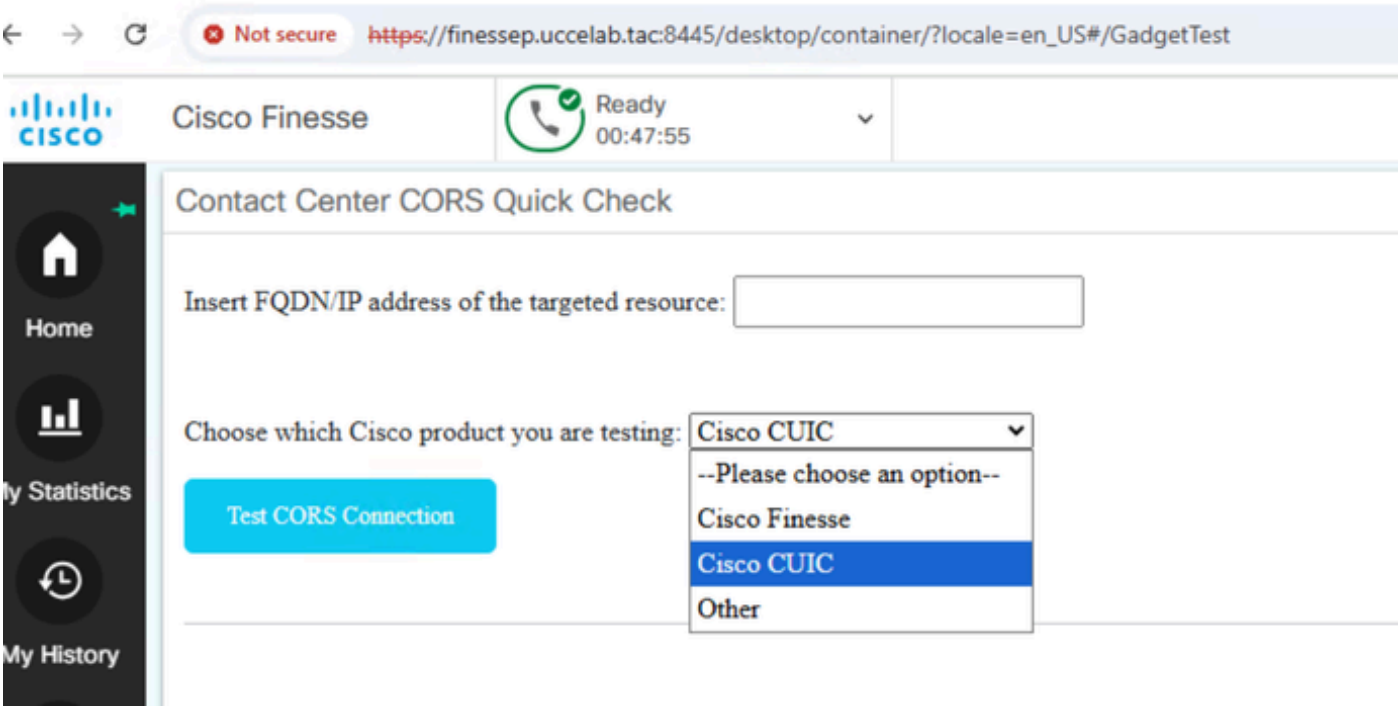
2. Copiez le contenu du dossier Contact Center CORS Quick Check 12.6-v1.0 dans le répertoire 3rdpartygadget de votre installation Finesse.

3. Ajoutez le gadget au rôle d'utilisateur souhaité (Agent, Superviseur, etc.) dans la présentation du bureau Finesse. L'exemple de code XML fourni illustre la configuration correcte pour l'ajout de ce gadget.

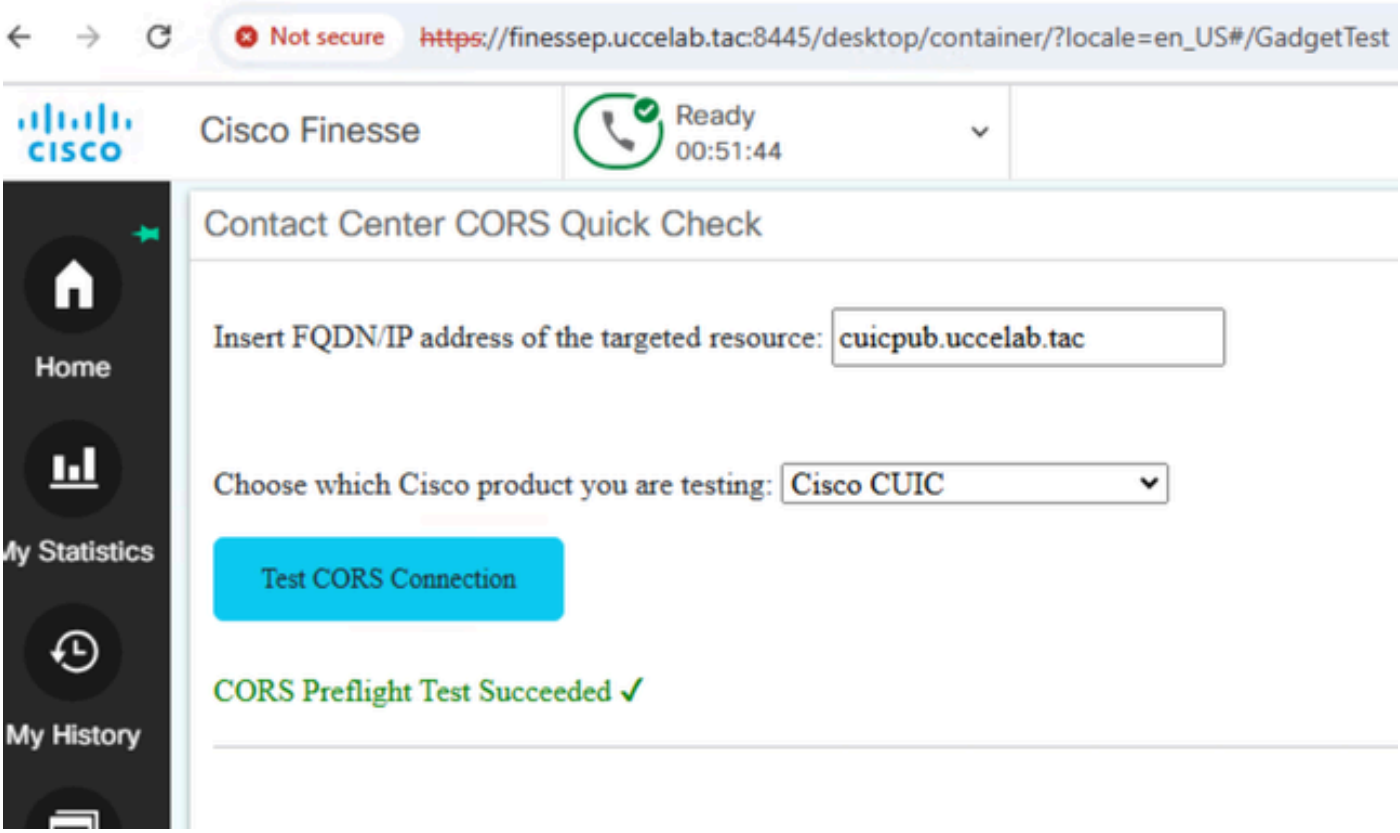
```
<gadget>/3rdpartygadget/files/TestCORSgadget.xml</gadget>
```

Consultez le chapitre Gadgets tiers dans leGuide du développeur Finesse et le chapitre Gérer les gadgets tiers dans leGuide d'administration Finesse pour plus d'informations sur le téléchargement de gadgets tiers et leur ajout au Bureau.

Une fois les fichiers de gadget téléchargés et le service Cisco Finesse Tomcat redémarré, le gadget est disponible et affiche l'interface utilisateur graphique (GUI).



Vous pouvez sélectionner CUIC dans la liste déroulante en haut de la page. Saisissez le nom de domaine complet (FQDN) du serveur CUIC dans le champ prévu à cet effet. Un test réussi sera effectué comme indiqué ici.



Un test réussi signifie que le serveur CUIC est correctement configuré pour le partage de ressources d'origine croisée (CORS) avec le serveur Finesse. Les journaux du traceur SAML du navigateur indiquent qu'une requête HTTP OPTIONS (le contrôle en amont CORS) a été envoyée au serveur CUIC. Cette requête incluait l'adresse du serveur Finesse dans l'en-tête Origin. Le

serveur CUIC a répondu avec un message HTTP 200 OK, et surtout, l'en-tête Access-Control-Allow-Origin dans la réponse contenait également l'adresse du serveur Finesse. Cela confirme que le serveur CUIC est configuré pour autoriser les requêtes provenant de l'origine du serveur Finesse, en vérifiant que CORS est correctement configuré.

<#root>

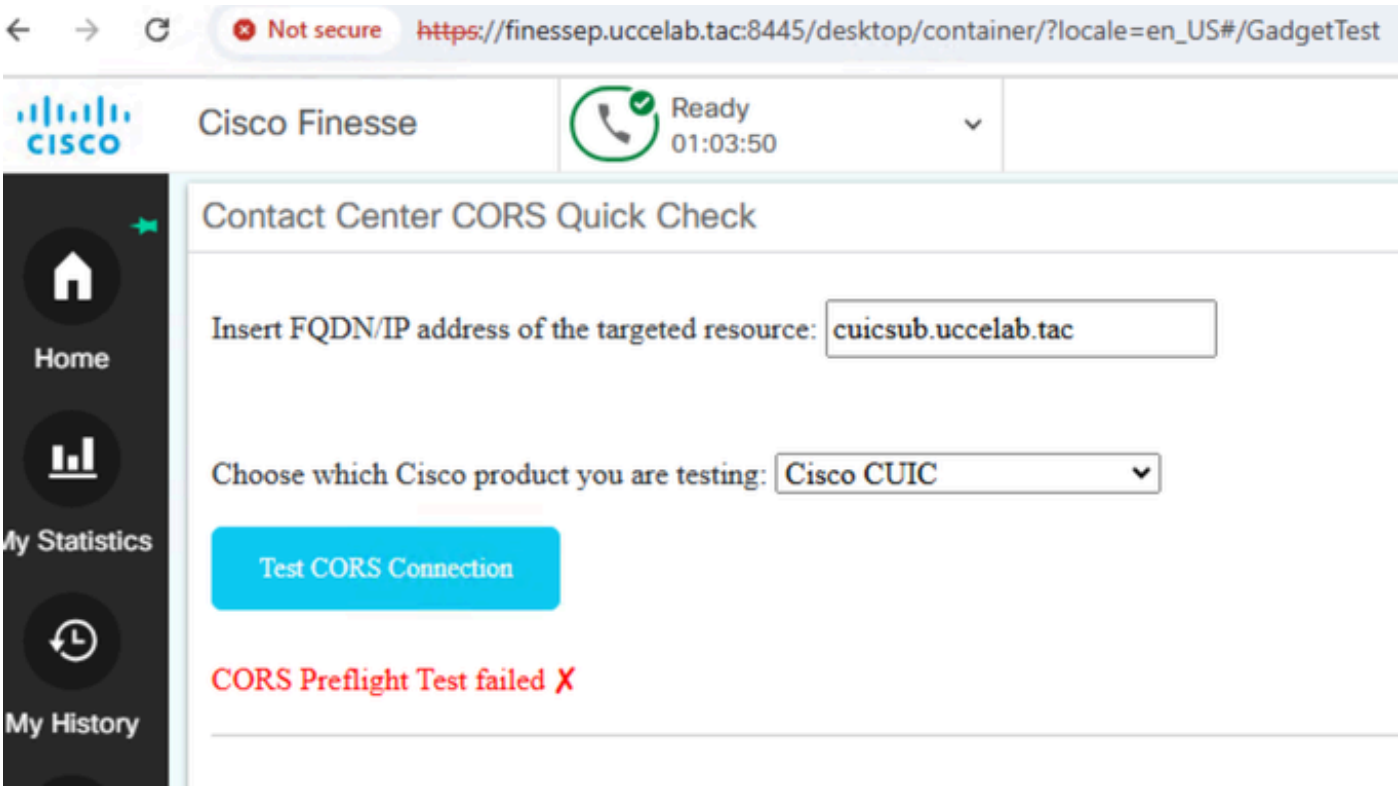
```
OPTIONS https://cuicpub.uccelab.tac/cuic/ HTTP/1.1

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
```

Dans ce scénario, l'outil présente une configuration qui ne fonctionne pas. Contrairement à l'exemple précédent, le serveur Finesse n'est pas configuré en tant qu'abonné sur le serveur CUIC. Au lieu de cela, il est configuré uniquement sur l'éditeur CUIC. Par conséquent, la demande

de contrôle en amont CORS échoue et le serveur CUIC répond avec une erreur HTTP 403 (Interdit).



<#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
```

Comme vous pouvez le voir dans le résultat de l'interface de ligne de commande (CLI) de l'abonné CUIC, Cisco Finesse n'est pas répertorié. Cela indique que Finesse n'est pas actuellement configuré en tant qu'abonné sur ce serveur CUIC.

<#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
```

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.