Kenntnis der Cross-Origin Resource Sharing (CORS) für Finesse

Inhalt

Einleitung

In diesem Dokument wird die gemeinsame Nutzung ursprungsübergreifender Ressourcen umfassend beschrieben, sodass die zugrunde liegenden Prozesse bei der Fehlerbehebung gründlich verstanden werden.

Voraussetzungen

Anforderungen

Cisco empfiehlt, dass Sie über Kenntnisse in folgenden Bereichen verfügen:

- 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

Verwendete Komponenten

Die Informationen in diesem Dokument basierend auf folgenden Software- und Hardware-Versionen:

- UCCE-Version 12.6.2
- Finesse Version 12.6.2
- CUIC Version 1.6.2

Die Informationen in diesem Dokument beziehen sich auf Geräte in einer speziell eingerichteten

Testumgebung. Alle Geräte, die in diesem Dokument benutzt wurden, begannen mit einer gelöschten (Nichterfüllungs) Konfiguration. Wenn Ihr Netzwerk in Betrieb ist, stellen Sie sicher, dass Sie die möglichen Auswirkungen aller Befehle kennen.

Hintergrundinformationen

Was ist CORS?

Mit Cross-Origin Resource Sharing (CORS) können Server steuern, welche Websites (Domänen, Protokolle und Ports) auf ihre Ressourcen zugreifen dürfen. Während Browser normalerweise Anfragen unterschiedlicher Herkunft blockieren (die gleiche Ursprungsrichtlinie), gibt CORS Servern die Möglichkeit, diese Einschränkung selektiv zu lockern. Im Wesentlichen verwenden Server spezielle HTTP-Header, um dem Browser mitzuteilen, welche Ursprungs- und Anforderungstypen zulässig sind (z. B. GET, POST usw.) und welche benutzerdefinierten Header eingeschlossen werden können. So können die Server selbst entscheiden, wer wie auf ihre APIs zugreifen darf - von vollständig offenem Zugriff bis hin zu strikt eingeschränktem Zugriff. CORS sorgt dafür, dass der Browser und der Server über diese HTTP-Header kommunizieren, um Anfragen unterschiedlicher Herkunft zu verwalten.

CORS verwendet HTTP-Header, um kontrollierte Ursprungübergreifende Anfragen zu aktivieren. Der Browser und der Server kommunizieren über diese Header, wobei der Server zulässige Ursprungs-, Methoden- und Header-Typen angibt. Wenn die Antwort-Header des Servers fehlen oder ungültig sind, blockiert der Browser die Antwort und erzwingt die gleiche Ursprungsrichtlinie. Bei bestimmten Anfragen sendet der Browser zunächst eine Preflight-Anfrage an den Server, um sicherzustellen, dass er die eigentliche Cross-Origin-Anfrage annimmt.

Browser verwenden Preflight-Anfragen, um zu überprüfen, ob ein Server eine Cross-Origin-Anfrage zulässt, bevor die eigentliche Anfrage gesendet wird. Diese Anfragen vor dem Flug beinhalten Details wie die HTTP-Methode und benutzerdefinierte Header. CORS-fähige Server können dann reagieren, indem sie die eigentliche Anforderung entweder zulassen oder ablehnen. Wenn ein Server nicht für CORS konfiguriert ist, antwortet er nicht korrekt auf den Preflight, und der Browser blockiert die eigentliche Anfrage, sodass der Server vor unerwünschtem, ursprungsübergreifendem Zugriff geschützt ist.

Cross-Origin Resource Sharing (CORS) ist für die Web-Sicherheit und -Funktionalität von entscheidender Bedeutung. Sie ermöglicht einen kontrollierten Zugriff auf Ressourcen unterschiedlicher Herkunft (Domänen, Protokolle, Ports), was notwendig ist, da Browser eine Richtlinie gleicher Herkunft durchsetzen, die diesen Zugriff normalerweise blockiert.

Lebenszyklus eines CORS

Eine CORS-Anfrage besteht aus zwei Seiten: der Client, der die Anforderung stellt, und der Server, der die Anforderung empfängt. Auf der Clientseite schreibt der Entwickler JavaScript-Code, um die Anforderung an den Server zu senden. Der Server reagiert auf die Anforderung, indem er spezielle CORS-spezifische Header einstellt, um anzuzeigen, dass die Cross-Origin-Anforderung zulässig ist. Ohne die Beteiligung des Clients und des Servers schlägt die CORS-

Anforderung fehl.

Die wichtigsten Player einer CORS-Anforderung sind der Client, der Browser und der Server. Der Client benötigt einige Daten vom Server, z. B. eine JSON-API-Antwort oder den Inhalt einer Webseite. Der Browser fungiert als vertrauenswürdiger Vermittler, um sicherzustellen, dass der Client auf die Daten vom Server zugreifen kann.

Kunde:

Der Client ist ein Ausschnitt aus JavaScript-Code, der auf einer Website ausgeführt wird, und ist für die Initiierung der CORS-Anforderung verantwortlich.

Anmerkung: Finesse ist eine Web-Anwendung. Es wird auf einem Server installiert, und die Agenten greifen einfach über ihren Webbrowser darauf zu, sodass keine Client-seitige Installation oder Wartung von Plugins oder anderer Software mehr erforderlich ist. Wie im Beispiel CORS in Aktion mit Cisco Finesse gezeigt, unterstützt diese Architektur Funktionen wie Live-Datenberichte. In diesem Zusammenhang fungiert der JavaScript-Code des Cisco Finesse Live Data Gadgets als Client, während Cisco CUIC als Server innerhalb des CORS-Lebenszyklus dient. Im Wesentlichen interagiert der browserbasierte Finesse-Client mit dem CUIC-Server, um Live-Daten abzurufen.

Client/Benutzer:

Manchmal werden die Wörter "client" und "user" synonym verwendet, im Kontext von CORS unterscheiden sie sich jedoch. Ein Benutzer ist eine Person, die eine Website besucht, oder ein Finesse-Benutzer (Agent oder Supervisor), der in diesem Zusammenhang auf Finesse zugreift, während ein Kunde der eigentliche Code ist, der von dieser Website bereitgestellt wird. Mehrere Benutzer können dieselbe Website besuchen und denselben JavaScript-Client-Code erhalten.

Browser:

Der Browser, auch Benutzer-Agent genannt, hostet den clientseitigen Code. Es spielt eine entscheidende Rolle bei der CORS, da es ausgehenden Anfragen zusätzliche Informationen hinzufügt, sodass der Server den Client identifizieren kann. Darüber hinaus interpretiert der Browser die Antwort des Servers und bestimmt, ob die Daten an den Client gesendet oder ein Fehler zurückgegeben werden soll. Diese browserseitigen Aktionen sind für die Gewährleistung der Sicherheit, die von derselben Richtlinie gewährleistet wird, von entscheidender Bedeutung. Ohne die Durchsetzung der CORS-Regeln durch den Browser können Clients unbefugte Anfragen stellen, was diesen wichtigen Sicherheitsmechanismus kompromittiert.

Server:

Der Server ist das Ziel der CORS-Anforderung, und er ist CUIC für das Live-Daten-Gadget am Beispiel von Cisco Finesse. Der Server speichert die vom Client gewünschten Daten und hat das letzte Wort, ob die CORS-Anforderung zulässig ist oder nicht.

Nachdem Sie nun wissen, wer an einer CORS-Anfrage beteiligt ist, wollen wir uns ansehen, wie sie alle zusammenarbeiten. Die folgenden Abbildungen zeigen den CORS-Lebenszyklus auf

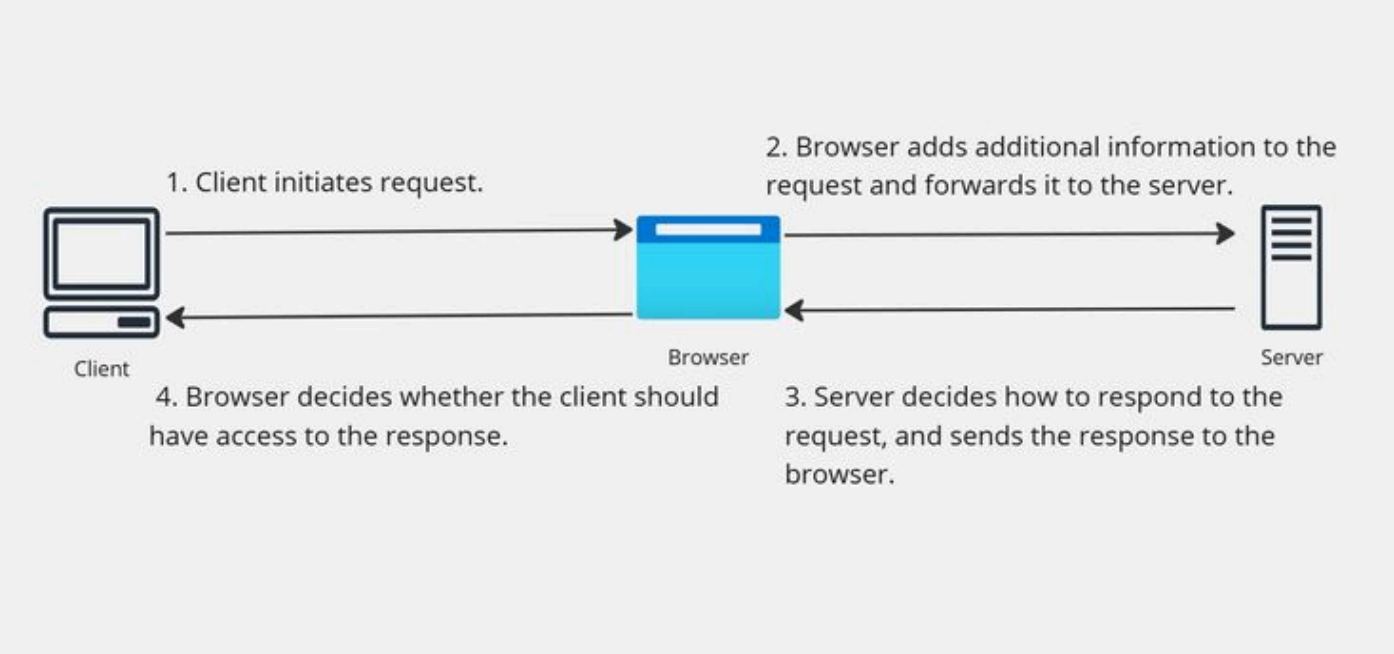
höchster Ebene:

1. Der Client initiiert die Anforderung.

2. Der Browser fügt der Anfrage zusätzliche Informationen hinzu und leitet diese an den Server weiter.

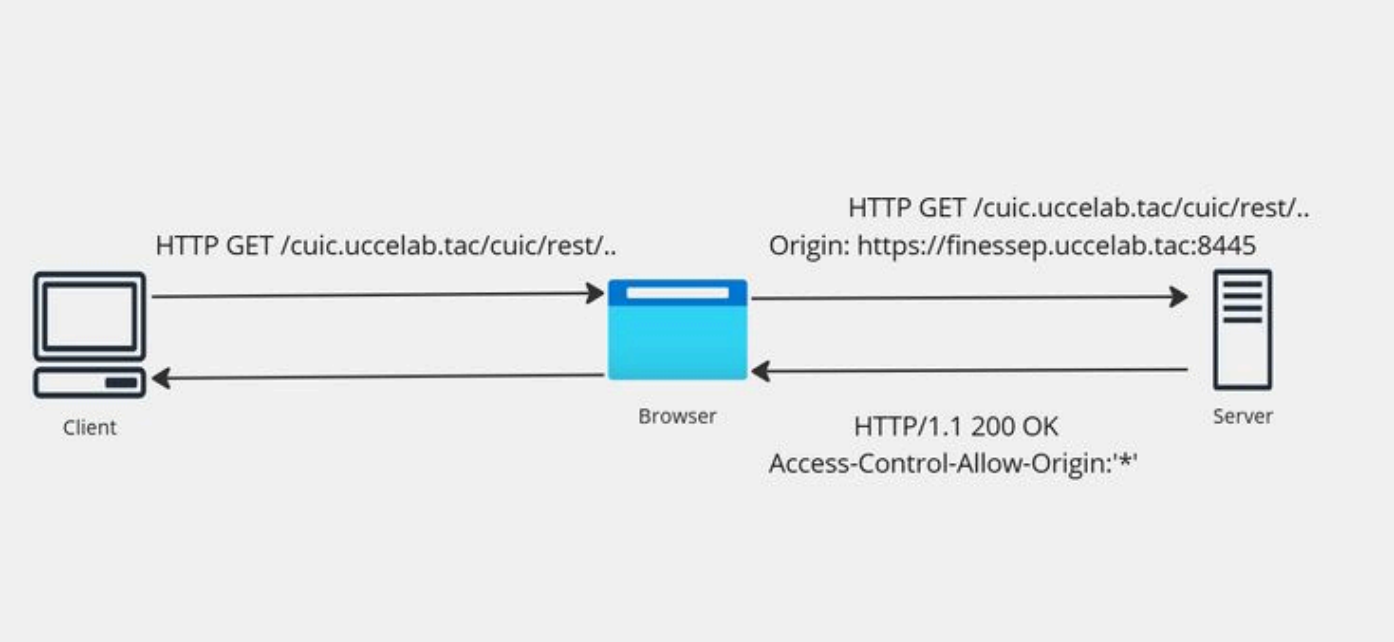
3. Der Server entscheidet, wie er auf die Anfrage antwortet, und sendet die Antwort an den Browser.

4. Der Browser entscheidet, ob der Client Zugriff auf die Antwort haben muss, und gibt die Antwort entweder an den Client weiter oder gibt einen Fehler zurück.

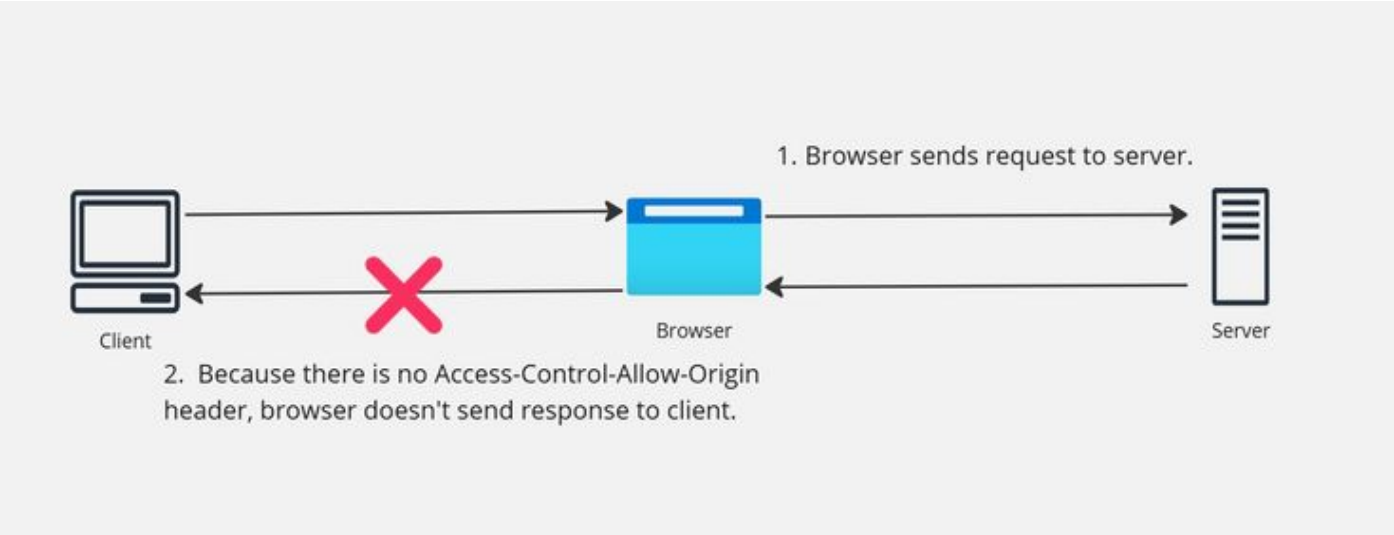


Vor dem Senden einer Cross-Origin-Anfrage fügt der Browser der HTTP-Anfrage automatisch einen Origin-Header hinzu. Dieser Header, den der Client nicht ändern kann, ist ein wichtiger Teil von CORS und dient dazu, den Ursprung des Clients (d. h. die Domäne, das Protokoll und den Port, von dem die Client-Ressource geladen wurde) zu identifizieren. Diese Sicherheitsmaßnahme verhindert, dass Kunden andere Ursprünge imitieren. Der Ursprungs-Header ist für CORS von grundlegender Bedeutung, da der Client dem Server mitteilt, woher er stammt.

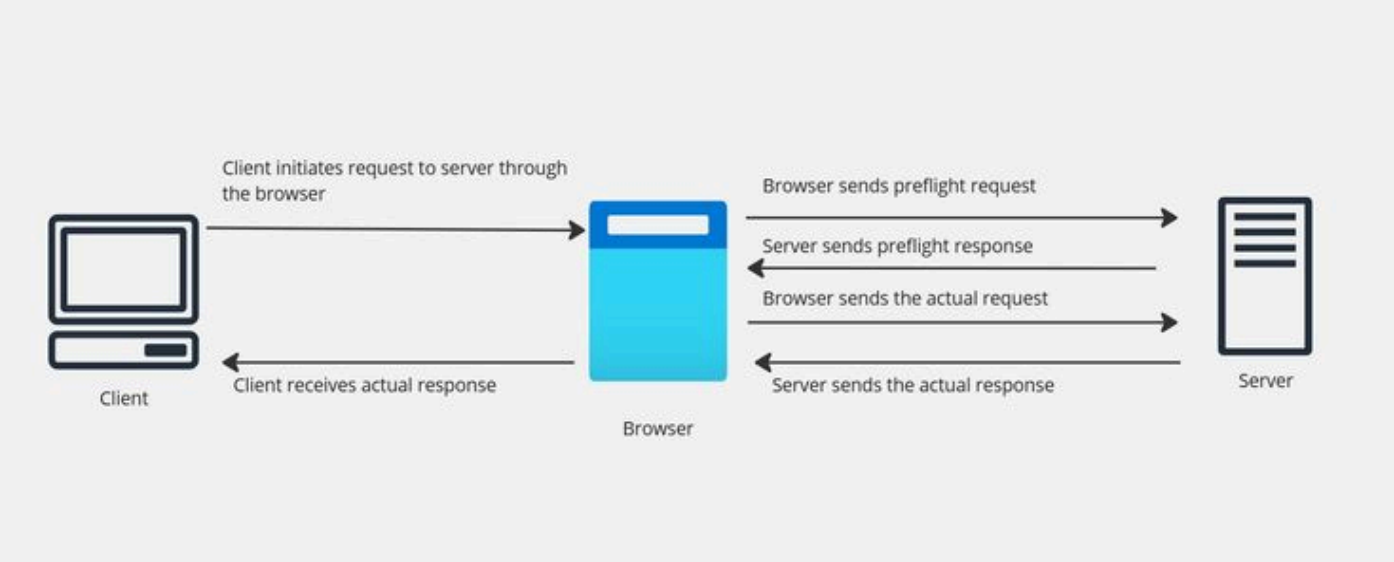
Bei einer CORS-Interaktion (Cross-Origin Resource Sharing) wird der Ursprung des Clients durch den Origin-Header in der ursprünglichen Anforderung identifiziert. Der Server verwendet dann den Access-Control-Allow-Origin-Header in seiner Antwort, um anzugeben, ob der Client auf die angeforderte Ressource zugreifen darf. Dieser Response-Header ist von entscheidender Bedeutung. Wenn sie nicht vorhanden ist, schlägt die CORS-Anfrage fehl. Der Header "Access-Control-Allow-Origin" kann einen Platzhalter (*) enthalten, der den Zugriff von einem beliebigen Ursprung aus erlaubt, oder einen bestimmten Ursprung, der nur diesem bestimmten Client Zugriff gewährt. Während im Bild "Access-Control-Allow-Origin" angezeigt wird: * Das bedeutet, dass CUIC alle Anfänge zulässt, und dass CUIC diesen Header in der Regel in realen Szenarien mit einem bestimmten Ursprung sendet.



Wenn ein Browser eine CORS-Anfrage ablehnt, bedeutet dies, dass der Client keine Informationen über die Antwort des Servers erhält. Der Client weiß nur, dass ein Fehler aufgetreten ist, aber es fehlen Details zum jeweiligen Problem. Dies kann das Debuggen von CORS-Fehlern schwierig gestalten, da es schwierig ist, einen CORS-Fehler von anderen Fehlertypen zu unterscheiden. Auch wenn die erste Anfrage an den Server gesendet wird, blockiert der Browser, wenn die Antwort des Servers keinen gültigen Access-Control-Allow-Origin-Header hat, die Antwort und löst einen Fehler auf der Client-Seite aus, sodass der Client die detaillierte Antwort des Servers nicht sehen kann.



Dieses Bild erklärt den gesamten CORS-Prozess, mit besonderem Fokus auf der Preflight-Phase, die für die Bearbeitung bestimmter Arten von Cross-Origin-Anfragen wesentlich ist.

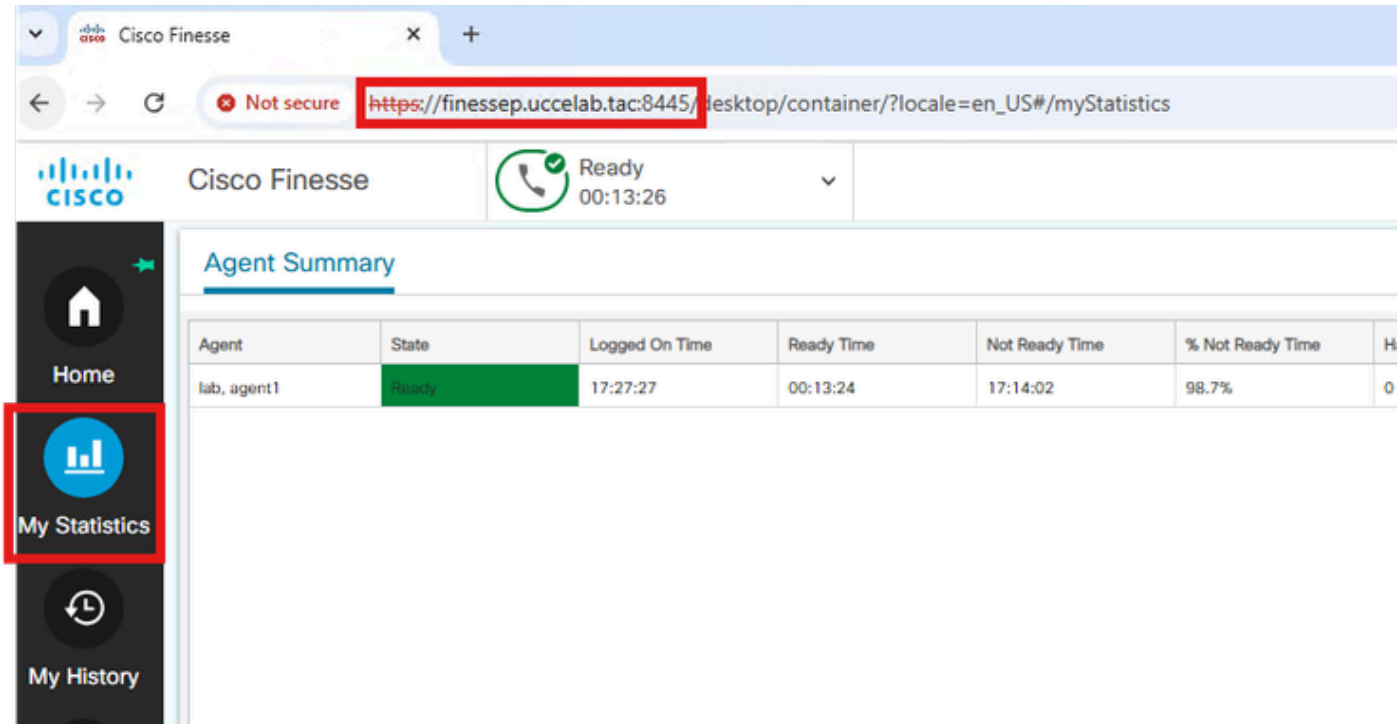


CORS in Aktion mit Cisco Finesse

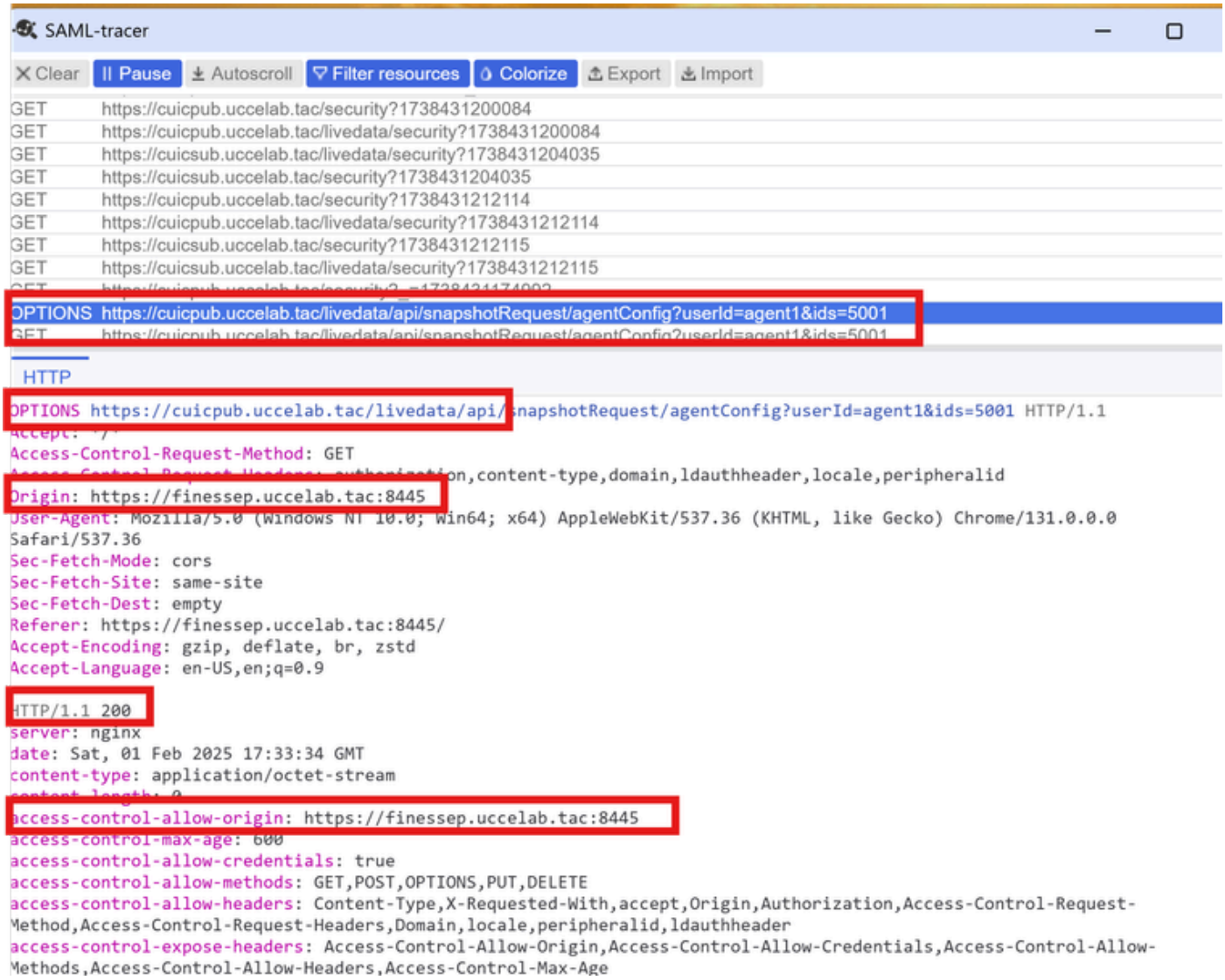
Beispieltext: Analyse des CORS-Verhaltens mit dem Live Data Gadget

In diesem Abschnitt wird die typische Verwendung von Cross-Origin Resource Sharing (CORS) mit Cisco Finesse in Contact Centern beschrieben. Agenten und Supervisoren greifen in der Regel mit Cisco Finesse auf Echtzeitdatenberichte zu (wie im Beispielbild dargestellt).

Wenn ein Agent oder Supervisor auf ein Berichts-Gadget klickt, wird durch seine Aktion eine Datenabfrageanforderung ausgelöst. Diese Anforderung wird vom JavaScript-Code der Finesse-Anwendung (der als Client fungiert) mithilfe einer GET-Methode an den CUIC/Live Data-Server gesendet. Wie im SAML-Tracer-Image gezeigt, sendet der Browser zunächst eine Preflight-Anfrage an den Server, den zuvor beschriebenen CORS-Lebenszyklus.



Eine HTTP OPTIONS-Anfrage (die Preflight-Anfrage) wird an den CUIC/Live Data-Server gesendet. Diese Anforderung gibt den Ursprung als vollqualifizierten Domänennamen (Fully Qualified Domain Name, FQDN) des Finesse-Servers an, einschließlich Port 8445. Dies sind die Adressen und Ports, die die Agenten für den Zugriff auf die Cisco Finesse-Anwendung verwenden.



Die Befehle der Befehlszeilenschnittstelle (CLI) auf dem CUIC/Live Data-Server steuern, welche Quellen auf die Live-Datenressourcen zugreifen dürfen. Wenn der Ursprung des Finesse-Servers (FQDN und Port) in diesen Einstellungen konfiguriert ist, können die Agenten die Details des Live-Daten-Gadgets in Finesse anzeigen.

```
admin:utils live-data cors allowed_origin list

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

```
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
4. https://finessep.uccelab.tac:8445
```

TAC Tool für CORS Verbindungstests

Serverseitige CORS-Fehlkonfigurationen können in Cisco Finesse manchmal zu Problemen mit Drittanbieter- oder Live-Daten-Gadgets führen. Dieser Artikel enthält einen Link zu einem CORS Quick Check-Gadget. Ein Tool zur Fehlerbehebung unterstützt die Diagnose von Problemen bei der gemeinsamen Ressourcennutzung über mehrere Herkunftsländer hinweg, die sich auf Finesse-Gadgets auswirken, einschließlich Live-Datenanzeigen und anderer Integrationen von Drittanbietern.

Technisch gesehen funktioniert dieses Gadget, indem Präflight-Anfragen vom Cisco Finesse-Client an eine bestimmte Zielressource gesendet werden. Diese Schnellprüfungsfunktion hilft dabei, CORS-bezogene Probleme schnell zu erkennen und zu beheben und beschleunigt so die Fehlerbehebung.

So stellen Sie das Contact Center CORS Quick Check 12.6-v1.0-Gadget auf dem Finesse-Desktop bereit:

1. Laden Sie Gadget-Dateien von Contact Center CORS Quick Check 12.6-v1.0 Ordner.2.

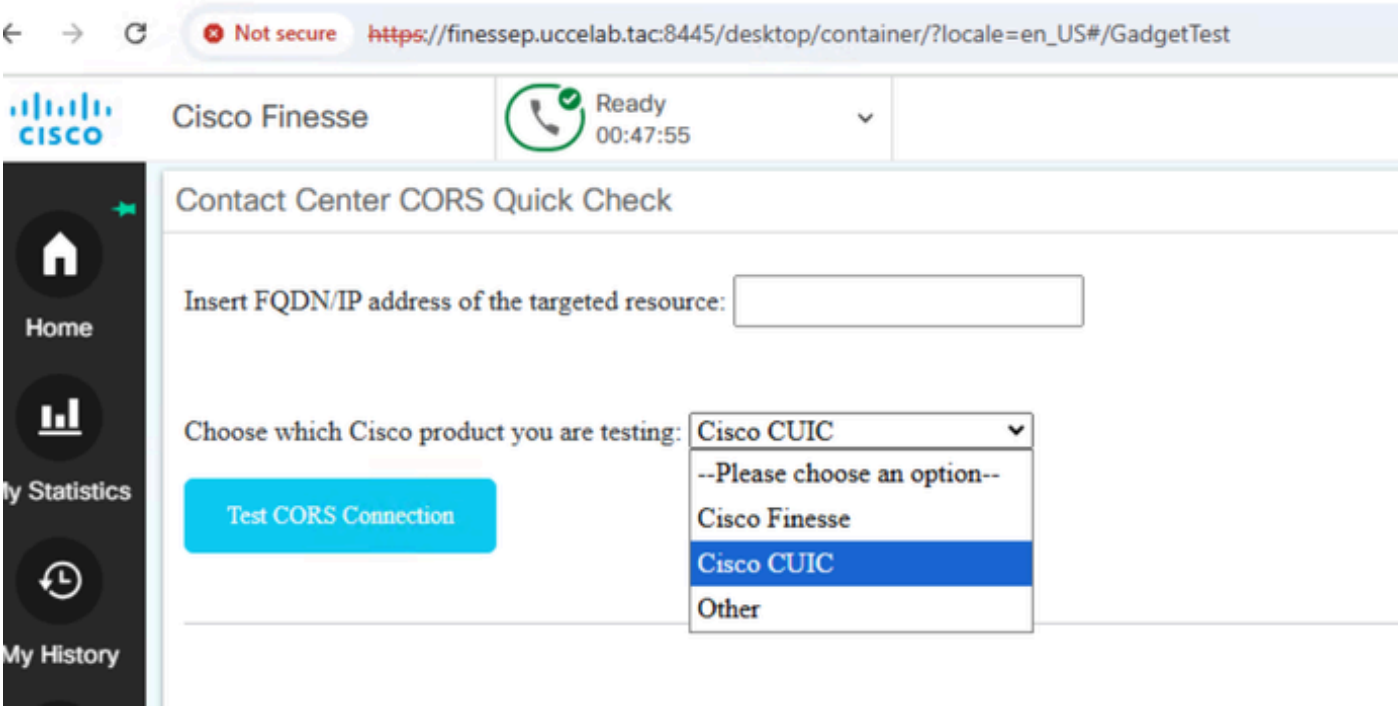
2. Kopieren Sie den Inhalt des Ordners Contact Center CORS Quick Check 12.6-v1.0 in das Verzeichnis 3rdpartygadget Ihrer Finesse-Installation.

3. Fügen Sie das Gadget der gewünschten Benutzerrolle (Agent, Supervisor usw.) im Finesse-Desktop-Layout hinzu. Im angegebenen Beispiel-XML wird die richtige Konfiguration zum Hinzufügen dieses Gadgets veranschaulicht.

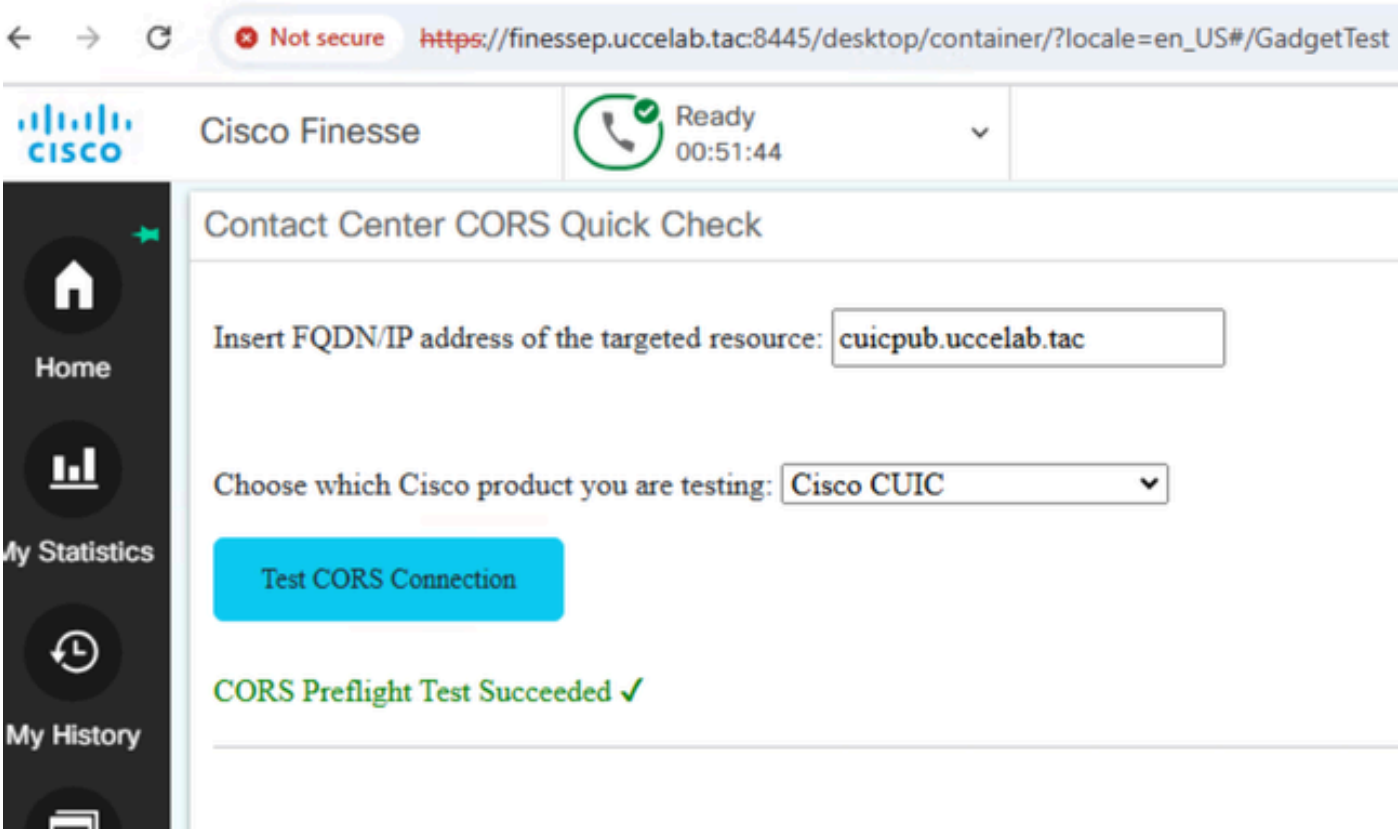
```
<gadget>/3rdpartygadget/files/TestCORSgadget.xml</gadget>
```

Weitere Informationen zum Hochladen von Drittanbieter-Gadgets und zum Hinzufügen von Drittanbieter-Gadgets finden Sie im Kapitel "Third Party Gadgets" in der Finesse-Administrationsanleitung im Finesse-Entwicklerhandbuch und im Kapitel "Manage Third-Party Gadgets".

Sobald die Gadget-Dateien hochgeladen und der Cisco Finesse Tomcat-Dienst neu gestartet wurden, ist das Gadget verfügbar und zeigt die grafische Benutzeroberfläche (GUI) an.



Sie können CUIC in der oberen Dropdown-Liste auswählen. Geben Sie den vollqualifizierten Domänennamen (Fully Qualified Domain Name, FQDN) des CUIC-Servers in das bereitgestellte Feld ein. Ein erfolgreicher Test wird wie hier gezeigt durchgeführt.



Ein erfolgreicher Test bedeutet, dass der CUIC-Server korrekt für Cross-Origin Resource Sharing (CORS) mit dem Finesse-Server konfiguriert ist. Die SAML-Tracer-Protokolle des Browsers zeigen an, dass eine HTTP OPTIONS-Anfrage (der CORS-Preflight) an den CUIC-Server gesendet wurde. Diese Anforderung enthielt die Adresse des Finesse-Servers im Origin-Header. Der CUIC-Server antwortete mit einer HTTP-Nachricht von 200 OK. Der Header "Access-Control-Allow-Origin" in der Antwort enthielt auch die Adresse des Finesse-Servers. Dadurch wird bestätigt, dass der CUIC-Server so konfiguriert ist, dass Anfragen vom Ursprung des Finesse-Servers zugelassen werden, und es wird überprüft, ob CORS korrekt eingerichtet ist.

<#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>

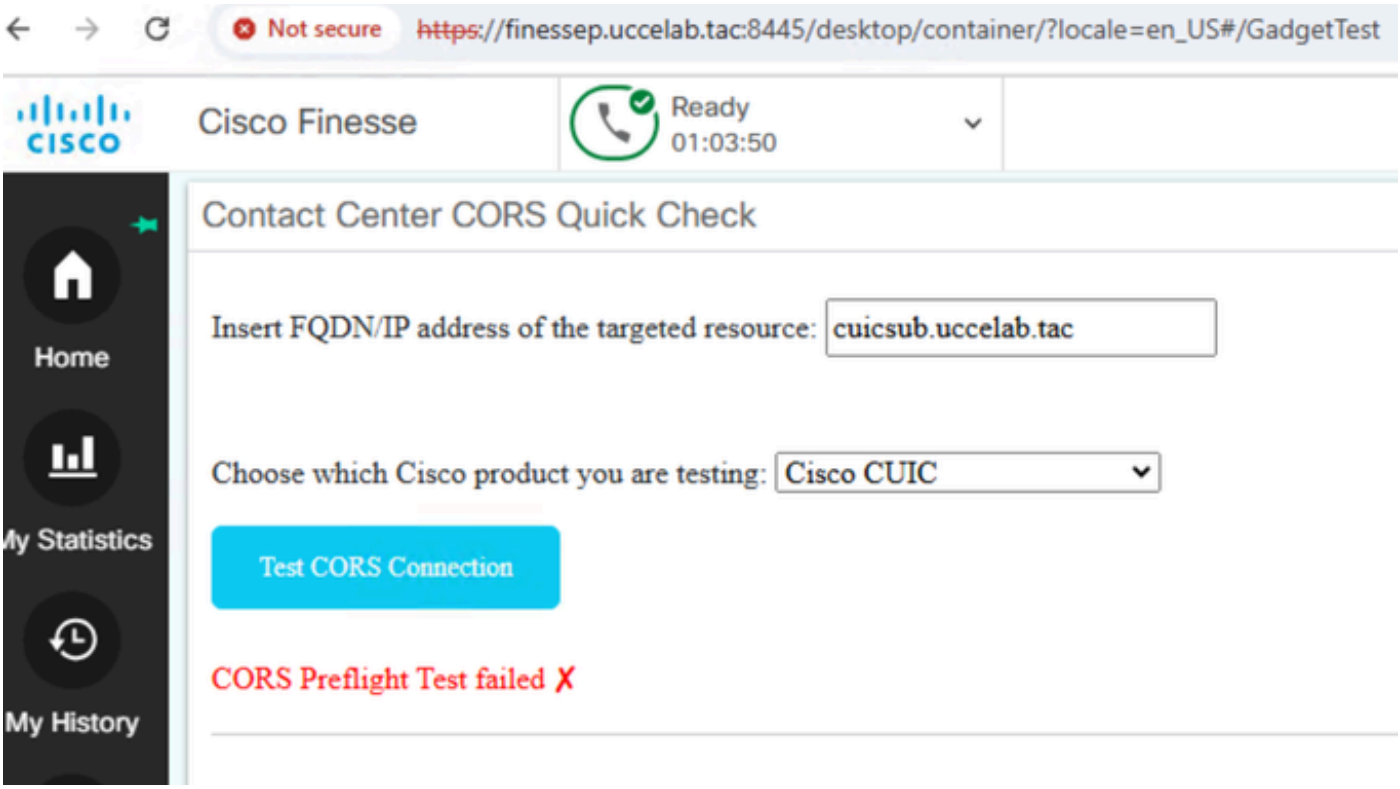
```
HTTP/1.1 200

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
```

In diesem Szenario zeigt das Tool eine nicht funktionierende Konfiguration. Im Gegensatz zum vorherigen Beispiel ist der Finesse-Server nicht als Subscriber auf dem CUIC-Server konfiguriert. Stattdessen wird sie nur auf dem CUIC-Publisher konfiguriert. Daher schlägt die CORS-Preflight-Anforderung fehl, und der CUIC-Server antwortet mit einem HTTP-403-Fehler (Verboten).



<#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
```

Wie Sie der Ausgabe der Befehlszeilenschnittstelle (CLI) des CUIC-Teilnehmers entnehmen können, ist Cisco Finesse nicht aufgeführt. Dies weist darauf hin, dass Finesse derzeit nicht als Abonnent auf diesem CUIC-Server konfiguriert ist.

<#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
```

Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.