Details zur UCCX Finesse-Architektur

Inhalt

Einleitung

In diesem Dokument wird die Finesse-Architektur ausführlich beschrieben, sodass die zugrunde liegenden Prozesse bei der Behebung von Finesse-Problemen sinnvoll sind.

Voraussetzungen

Anforderungen

Cisco empfiehlt die Kenntnis dieser Tools und Funktionen:

JTAPI - Java Telefony API

API - Application Programming Interface

UCCX - Unified Contact Center Express

CUCM - Cisco Unified Communications Manager

CTI = Computer Telefony Integration

Verwendete Komponenten

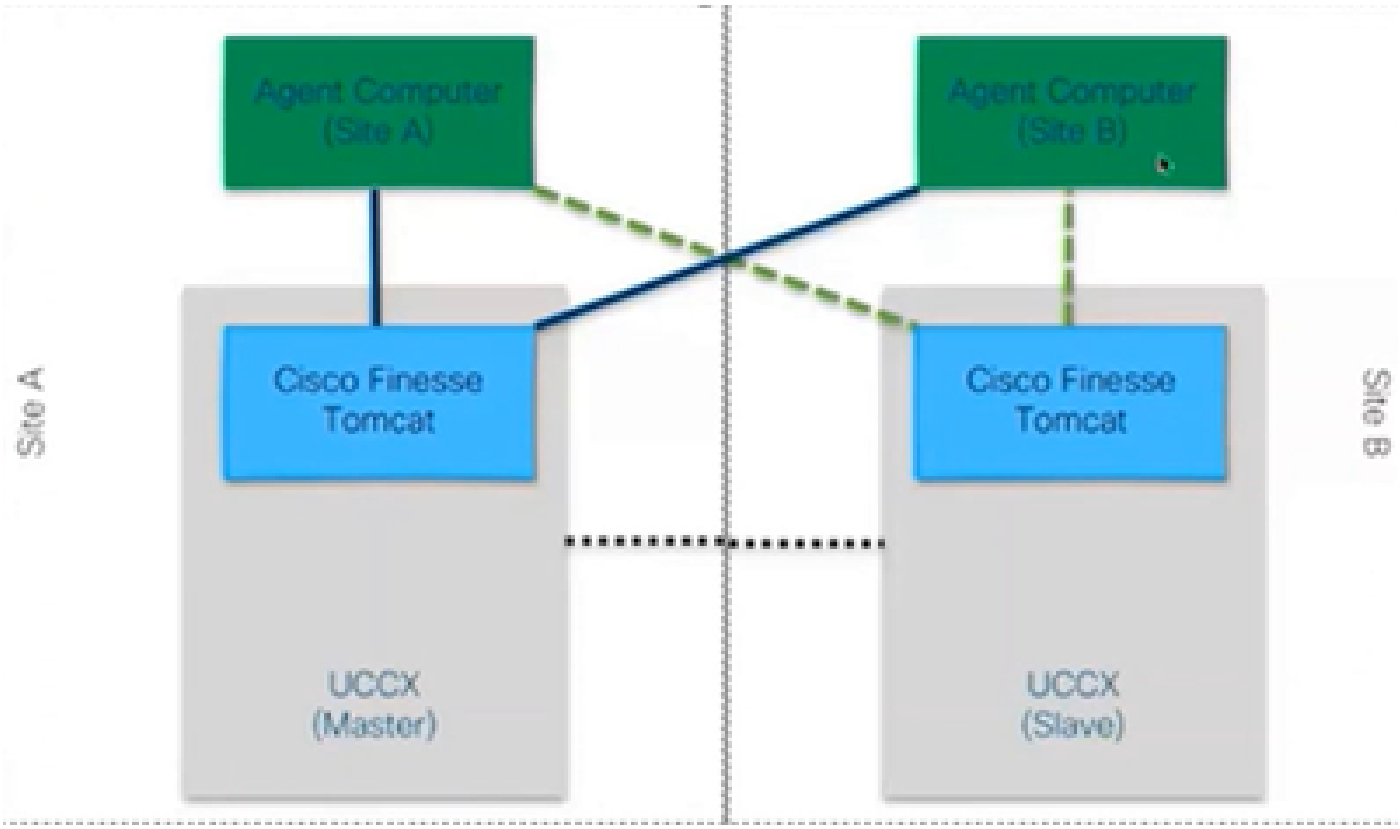
- Cisco Unified Contact Center Express (UCCX)

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

In diesem Dokument wird die Finesse-Architektur anhand eines allgemeinen Überblicks und des detaillierten Signalflusses sowie anhand von Beispielen und Diagrammen beschrieben.

15.000 m



50000 Ansicht

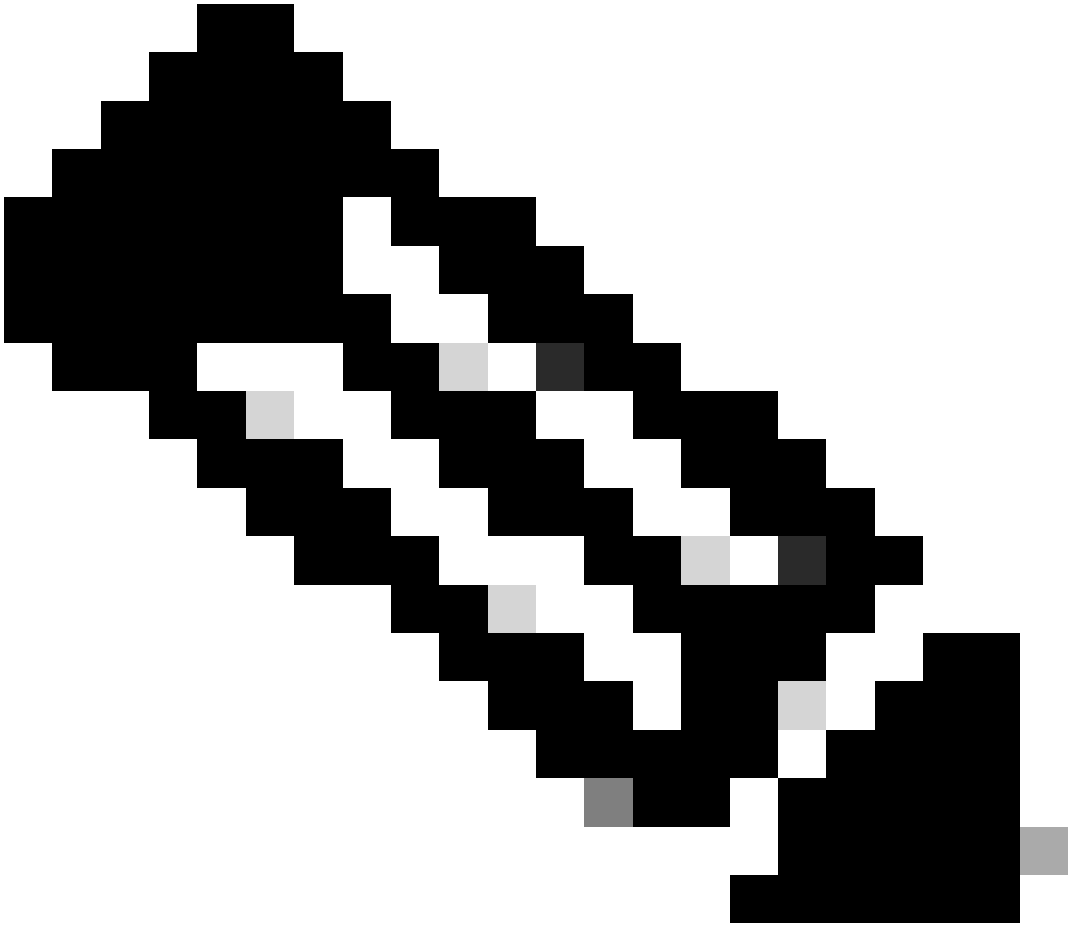
Finesse Tomcat

Finesse Tomcat ähnelt Cisco Tomcat in CUCM, da die Funktion zum Laden der Webseiten identisch ist, jedoch für einen anderen Service namens Finesse. Tomcat wurde für Finesse entwickelt, da es eine separate Webanwendung ist. Sie können sich nur mit dem CCX-Masterknoten gemäß den Versionen vor 11.5 bei finesse anmelden. Ab 11.6 können Sie sich nur noch während des Failovers bei jedem beliebigen Knoten anmelden (nicht empfohlen). Wenn Sie also einen WAN-Cluster in Betracht ziehen, bei dem beide Agenten (an Standort A und B) mit Finesse am Master-Knoten verbunden sind, dann besteht zu jeder Zeit eine inaktive Verbindung zum anderen Knoten von Cisco Finesse zur Engine. Dies ist eine CTI-OpenConf-Anforderung, die für Failover erforderlich ist.

Die Openconf-Anfrage wird regelmäßig gesendet, um zu überprüfen, ob der Knoten aktiv oder im Standby-Modus ist. Falls ein Failover vorliegt, verwendet der Benachrichtigungsdienst eine API, die als systeminfo API bezeichnet wird und den Client darüber informiert, dass dieser Knoten ausgefallen ist und ein Failover erforderlich ist. Dann läuft eine Datei, die den Browser zum anderen Knoten umleitet und dann wird die openconf rejavascriptsponse gesendet. Dieser Failover funktioniert nur, wenn die Zertifikate akzeptiert werden. (um eine sichere Verbindung zum Server herzustellen).

Es werden 3 Bescheinigungen vorgelegt, die

- Benachrichtigungsdienstzertifikat für diesen Knoten.
- Benachrichtigungsdienstzertifikat für Remote-Knoten.
- Finesse-Dienstzertifikat für Remote-Knoten.



Hinweis: Die Remote-Knoten-Zertifikate müssen akzeptiert werden, damit das Failover funktioniert.

HTTP(S)

Hierbei handelt es sich um ein Protokoll auf Anwendungsebene zum Senden von Hypermedia-Dokumenten wie HTML. Es ist für die Kommunikation zwischen Webbrowsern und Webservern konzipiert. Es handelt sich um ein Stateless-Protokoll, das bedeutet, dass der Server keine Daten zwischen den beiden Anforderungen speichert. Dies ist ein Client-Server-Protokoll. Für UCCX wird der Finesse-Client im Agent-Browser (PC) ausgeführt. Er sendet eine Anfrage über HTTP an den Server. Hier einige gebräuchliche HTTP-Methoden:

GET - zum Abrufen von Informationen von einem Server.
POST: Senden von Informationen an einen Server.

PUT - zum Ersetzen aller Komponenten auf einem Server.
LÖSCHEN - Zum Entfernen von Informationen von einem Server.

Finesse verwendet systeminfo api-Anfragen innerhalb der http-Anfrage. Wenn Sie zum Beispiel den Status eines Agenten ändern möchten, sendet der Browser eine PUT statt einen POST, denn wenn ein POST gesendet wird, wird der Server verwirrt, da er zwei Zustände hat, die ausgewählt werden sollen. Mit PUT wird der aktuelle Status ersetzt.

XMPP

Erweiterbares Messaging- und Presence-Protokoll

Dabei handelt es sich um eine Reihe von Protokollen, die für Instant Messaging und Presence verwendet werden. Alle XMPP-Einheiten werden mithilfe ihrer Jabber-IDS oder JIDs identifiziert. Eine der Erweiterungen dieses XMPP-Protokolls, das für Gadgets verwendet wird, ist als PUBSUB bekannt.

PUBSUB

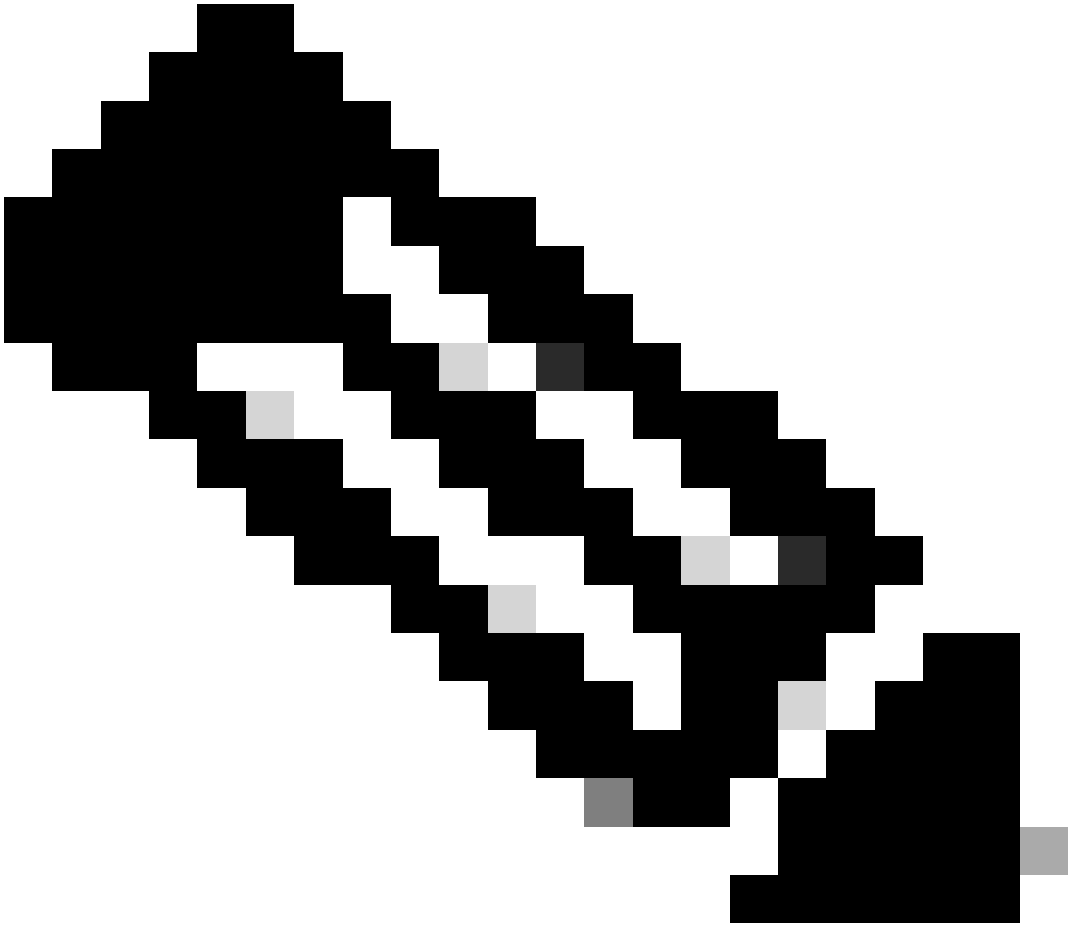
Es ist nicht der Publisher-Abonnent, den Sie sich vorstellen können. Es veröffentlicht und abonniert noch, aber es hat nichts mit den Datenbanken zu tun. XMPP verwendet einen Mechanismus, der als Knoten bezeichnet wird. Der Knoten überwacht im Grunde die Entität, die Ihnen wichtig ist. Alles, was für Sie wichtig ist und überwacht werden soll, fügen Sie einen Knoten hinzu. Dann, was PUBSUB tut, ist, abonnieren Sie für die Updates oder Benachrichtigungen auf diesem Knoten. Sie können Knoten für jeden Typ von Entität wie Agent, Dialog usw. haben. Wenn Sie einen Knoten für einen Agenten erstellen, werden Sie für den Knoten auf diesem Agenten abonniert, und was auch immer der Agent tut, Sie werden darüber benachrichtigt.

Der Zweck dieser Spezifikation ist es, dem XMPP-Server (Benachrichtigungsdienst) zu ermöglichen, Informationen abzurufen, die an XMPP-Knoten (Themen) veröffentlicht werden, und dann XMPP-Ereignisse an Entitäten zu senden, die für diesen Knoten abonniert sind.

Im Fall von Finesse sendet der CTI-Server (Computer Telefony Integration) CTI-Nachrichten an den Finesse-Webservice, um Finesse über Konfigurationsaktualisierungen zu informieren, wie z. B. die Erstellung eines Agenten oder einer CSQ (Contact Service Queue) oder Informationen zu einem Anruf. Diese Informationen werden dann in eine XMPP-Nachricht umgewandelt, die der Finesse-Webdienst an den Finesse Notification Service veröffentlicht.Der Finesse Notification Service sendet dann XMPP über BOSH-Nachrichten an Agenten, die an bestimmten XMPP-Knoten abonniert sind.

BOSH = Bi-Directional Streams over Synchronous HTTP

BOSH ist eine langlebige HTTP-Verbindung, bei der der Server die Anfrage für eine längere Zeit hält, bis er eine Antwort darauf hat, ansonsten eine leere Antwort sendet. Dies funktioniert für XMPP-Clients und XMPP-Server, kann aber auch für andere Anwendungen als XMPP verwendet werden.



Hinweis: XMPP ist zustandsbehaftet, während HTTP zustandslos ist (es speichert keine Informationen über die letzte Anforderung)

Wenn eine Webanwendung mit XMPP arbeiten muss, treten mehrere Probleme auf.

Problem 1: Browser unterstützen XMPP über Transmission Control Protocol (TCP) nicht.

Lösung 1: Webserver und Browser kommunizieren über HTTP-Nachrichten (HyperText Transfer Protocol), sodass Finesse und andere Webanwendungen XMPP-Nachrichten in HTTP-Nachrichten einbinden.

Problem 2: HTTP ist ein Stateless-Protokoll.

Lösung 2: Hierfür können Sie Cookies/Postdaten verwenden.

Problem 3: Das dritte Problem ist das unidirektionale Verhalten von HTTP, d. h. nur der Client sendet Anfragen, und der Server kann nur antworten. Da der Server keine Daten per Push bereitstellen kann, ist die Implementierung von XMPP über HTTP unnatürlich.

Lösung 3: Um dieses Problem zu beheben, benötigen Sie eine Brücke zwischen HTTP und XMPP.

Die vorgeschlagenen Lösungen sind:

- Polling (Legacy-Protokoll): wiederholte HTTP-Anforderungen mit der Aufforderung, neue Daten zu erhalten, definiert: HTTP-Polling

- Langes Polling wird auch als BOSH bezeichnet: Transportprotokoll, das die Semantik einer langlebigen, bidirektionalen TCP-Verbindung zwischen zwei Einheiten emuliert, indem mehrere synchrone HTTP-Anfrage/Antwort-Paare effizient verwendet werden, ohne dass ein häufiges Polling erforderlich ist. Der Grund für die Verwendung von BOSH besteht darin, die Tatsache zu vertuschen, dass der Server nicht sofort antworten muss, wenn eine Anfrage eingeht. Die Antwort wird bis zu einem bestimmten Zeitpunkt verzögert, bis der Server über Daten für den Client verfügt. Anschließend wird sie als Antwort gesendet. Sobald der Client die Antwort erhält, stellt er eine neue Anforderung usw.

Der Finesse Desktop-Client (Web-Anwendung) stellt alle 30 Sekunden eine veraltete BOSH-Verbindung über den TCP-Port 7443 her. Wenn nach 30 Sekunden keine Updates vom Finesse-Benachrichtigungsdienst vorliegen, sendet der Benachrichtigungsdienst eine HTTP-Antwort mit 200 OK und einem (fast) leeren Antworttext. Wenn der Notification Service z.B. über ein Update bei Vorliegen eines Agenten oder eines Dialogereignisses (Call) verfügt, werden die Daten sofort an den Finesse Web Client gesendet.

Zusammenfassung:

Der Finesse-Web-Client hat eine veraltete HTTP-Verbindung (http-bind) zum Finesse-Server über den TCP-Port 7443 eingerichtet. Dies wird als lange BOSH-Umfrage bezeichnet. Der Finesse Notification Service ist ein Presence-Service, der Updates zum Status eines Agenten, einen Anruf usw. bereitstellt. Wenn der Benachrichtigungsdienst über eine Aktualisierung verfügt, antwortet er auf die HTTP-Bind-Anforderung mit der Statusaktualisierung als XMPP-Nachricht im HTTP-Antworttext. Wenn 30 Sekunden nach Empfang der HTTP-Bind-Anforderung keine Statusaktualisierungen vorliegen, antwortet der Benachrichtigungsdienst ohne Statusaktualisierungen, damit der Finesse-Webclient eine weitere HTTP-Bind-Anforderung senden kann. Auf diese Weise kann der Benachrichtigungsdienst feststellen, dass der Finesse-Web-Client immer noch eine Verbindung zum Benachrichtigungsdienst herstellen kann und dass der Agent den Browser nicht geschlossen oder den Computer nicht in den Ruhezustand versetzt hat usw.

CTI

Sie können Computer Telefony Integration (CTI) verwenden, um während des Tuns, Empfangens und der Verwaltung von Telefonanrufen Computerverarbeitungsfunktionen zu nutzen. Mit CTI-Anwendungen können Sie Benutzerinformationen aus einer Datenbank mit einer Anrufer-ID abrufen oder mit den Informationen eines IVR-Systems (Interactive Voice Response) arbeiten, um einen Anruf vom Benutzer zusammen mit dessen Informationen an den zuständigen Service-Mitarbeiter weiterzuleiten. Der CTI-Manager auf dem CUCM antwortet auf die JTAPI-Anforderungen von UCCX. Der TCP-Port des CTI-Servers ist 12018. So kommunizieren Finesse Server und Engine (CTI-Server) miteinander.

Hier sind einige der über CTI ausgetauschten Informationen:

- Aktuelle Systemkonfiguration und zukünftige Updates.

- Agenten und ihre Zustände.

- Anrufe und deren Status.

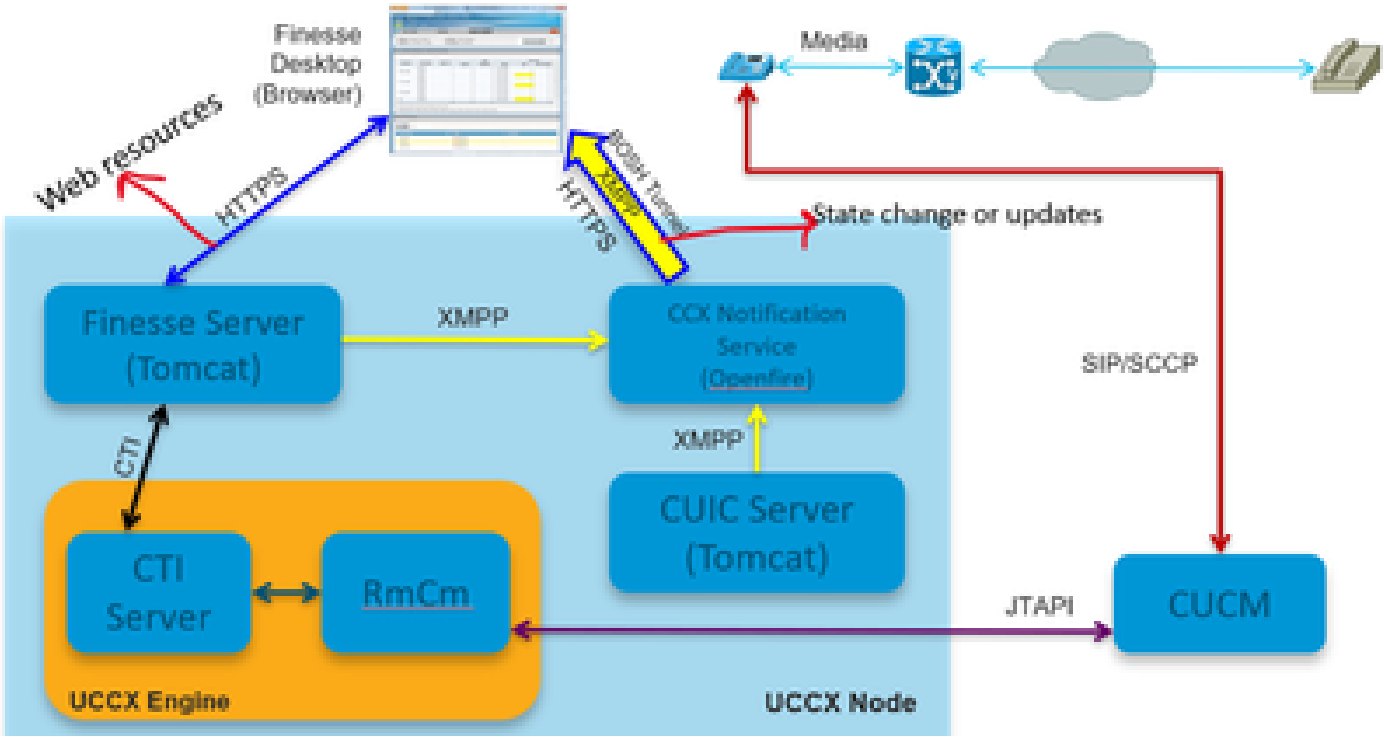
- Statistiken für Agenten, Anrufe und Warteschlangen in Echtzeit.

JTAPI

Cisco Unified JTAPI dient als Programmierschnittstellenstandard, der von Sun Microsystems für den Einsatz mit Java-basierten Computertelefonie-Anwendungen entwickelt wurde. Cisco JTAPI implementiert die Sun JTAPI 1.2-Spezifikation mit zusätzlichen Cisco-Erweiterungen. Die Kommunikation zwischen UCCX und CUCM erfolgt über JTAPI. So kommunizieren CUCM und Engine (Telefonie-Subsystem) miteinander. JTAPI wird zur Steuerung und Überwachung von CUCM-Telefonen, zur Weiterleitung von Anrufen über CTI-Ports und Weiterleitungspunkte, zum Starten und Stoppen von Aufzeichnungen auf dem CUCM sowie für alle Anrufweiterleitungsfunktionen verwendet.

30000 ft

Im folgenden Diagramm wird die Kommunikation zwischen UCCX Engine, Finesse, CUCM und Browser dargestellt.

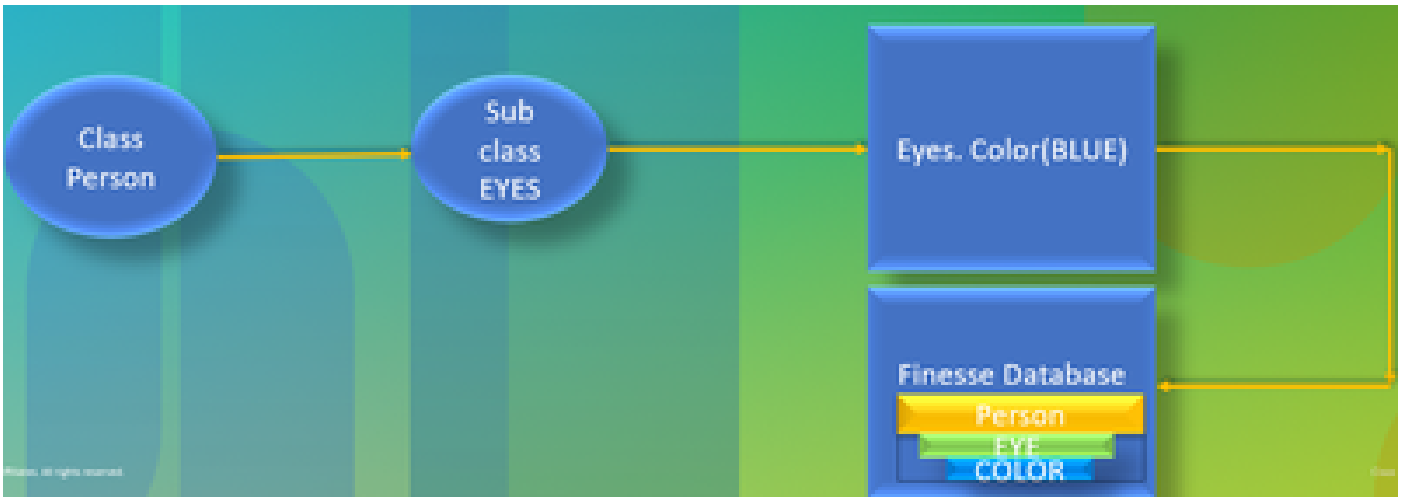


30000 Ansicht

Wir sollten bedenken, dass der Anruf mit dem Agenten durchgeführt wird. RmCm, der die Agentenerweiterung über JTAPI überwacht, teilt dem CTI-Server nun mit, über welche Statusänderung der Agent spricht. Diese Informationen werden vom CTI-Server (innerhalb der CCX-Engine) mithilfe von CTI an den Finesse-Server (Tomcat) gesendet. Finesse Server sendet diese Informationen über die Statusänderung mithilfe von XMPP an den CCX-Benachrichtigungsdienst. Der Benachrichtigungsdienst (Openfire) öffnet einen BOSH-Tunnel zum Agentenbrowser und aktualisiert die Informationen über die Statusänderung. So wird der Agent in den Status RESERVIERT versetzt. Alle Arten von Web-Ressourcen werden an den Finesse-Server mit HTTPS wie WAR-Dateien, Gadgets und so weiter (wenn nicht bereits im Cache) angefordert.

RUHESTAND

Das nächste Diagramm erläutert den Hibernate-Dienst.



Ruhezustand

HIBERNATE wird als Hochleistungsobjekt-/relationaler Persistenzdienst und Abfragedienst bezeichnet. Einfach ausgedrückt, es ordnet JAVA-Klassen Datenbanktabellen zu. Sie haben zum Beispiel ein JAVA-Objekt namens Team und eine Datenbanktabelle in der Finesse-Datenbank namens Team. Die JAVA-Klasse steuert, welche Informationen in der Tabelle enthalten sind, und HIBERNATE steuert, was dies bewirkt. Anstatt SQL-Abfragen zu verwenden, verwendet es Java-Klassen, um die Informationen zu aktualisieren.

ACHSE

Administrator-XML.

XML steht für eXtensible Markup Language und ist eine Markup-Sprache, die einige relativ einfache Regeln für das Codieren von Daten definiert. Es wurde hauptsächlich entwickelt, um strukturierte Daten in einem definierten Format zu übertragen und zu empfangen, das beide Systeme verstehen können. In der einfachsten Form definiert XML Tags, die in eckige Klammern (<>) eingeschlossen sind, und diese Tags umgeben die vom Tag beschriebenen Daten. Tags können eine Hierarchie mit Tags innerhalb anderer Tags bilden. Um beispielsweise ein einfaches Telefongerät zu definieren, können Sie sagen, dass ein Telefongerät drei Parameter benötigt: einen Namen, eine Beschreibung und eine Telefonnummer.

Es handelt sich um eine SOAP-basierte API, die die Remote-Bereitstellung auf dem CUCM ermöglicht. Diese dient zum Hinzufügen, Aktualisieren, Entfernen oder Abrufen von Informationen aus der CUCM-Datenbank. Zu den Abruffunktionen gehören die Überprüfung der Benutzerauthentifizierung und die Ausführung von SQL-Abfragen.Die AXL-API bietet Zugriff auf die gesamte CUCM-Datenbank. Die AXL-API dient lediglich zur Bereitstellung und bietet keinen Zugriff auf Laufzeit- oder Leistungsdaten.

AXL API verwendet HTTPS-Standardauthentifizierung. Jeder CUCM-Benutzer (Anwendung oder Endbenutzer) hat Lese-/Schreibzugriff über AXL, wenn er Mitglied der Zugriffssteuerungsgruppe **Standard CCM Super Users** oder einer Gruppe mit der ihm zugewiesenen Rolle **Standard AXL API Access** ist. Dies bedeutet, dass alle Superuser-Konten implizit bereits Zugriff auf die AXL-API haben. Um ein dediziertes Konto für die AXL API-Verwendung zu erstellen, müssen Sie zunächst eine Zugriffssteuerungsgruppe erstellen und ihr die **Standard-AXL API**-Zugriffsrolle zuweisen. Anschließend müssen Sie den Anwendungsbenutzer der neu erstellten Gruppe zuordnen. Für den schreibgeschützten AXL API-Zugriff können Sie eine separate Zugriffssteuerungsgruppe erstellen und dieser nur die **Standard AXL Read Only API Access**-Rolle zuweisen.

SOAP

Mit dem Simple Object Access Protocol (SOAP) können Informationen zwischen Anwendungen im XML-Format weitergegeben werden.

SOAP-Nachrichten werden von der sendenden Anwendung an die empfangende Anwendung übertragen, in der Regel über eine HTTP-Sitzung. Die eigentliche SOAP-Nachricht besteht aus dem Envelope-Element, das ein Body-Element und ein optionales Header-Element enthält.

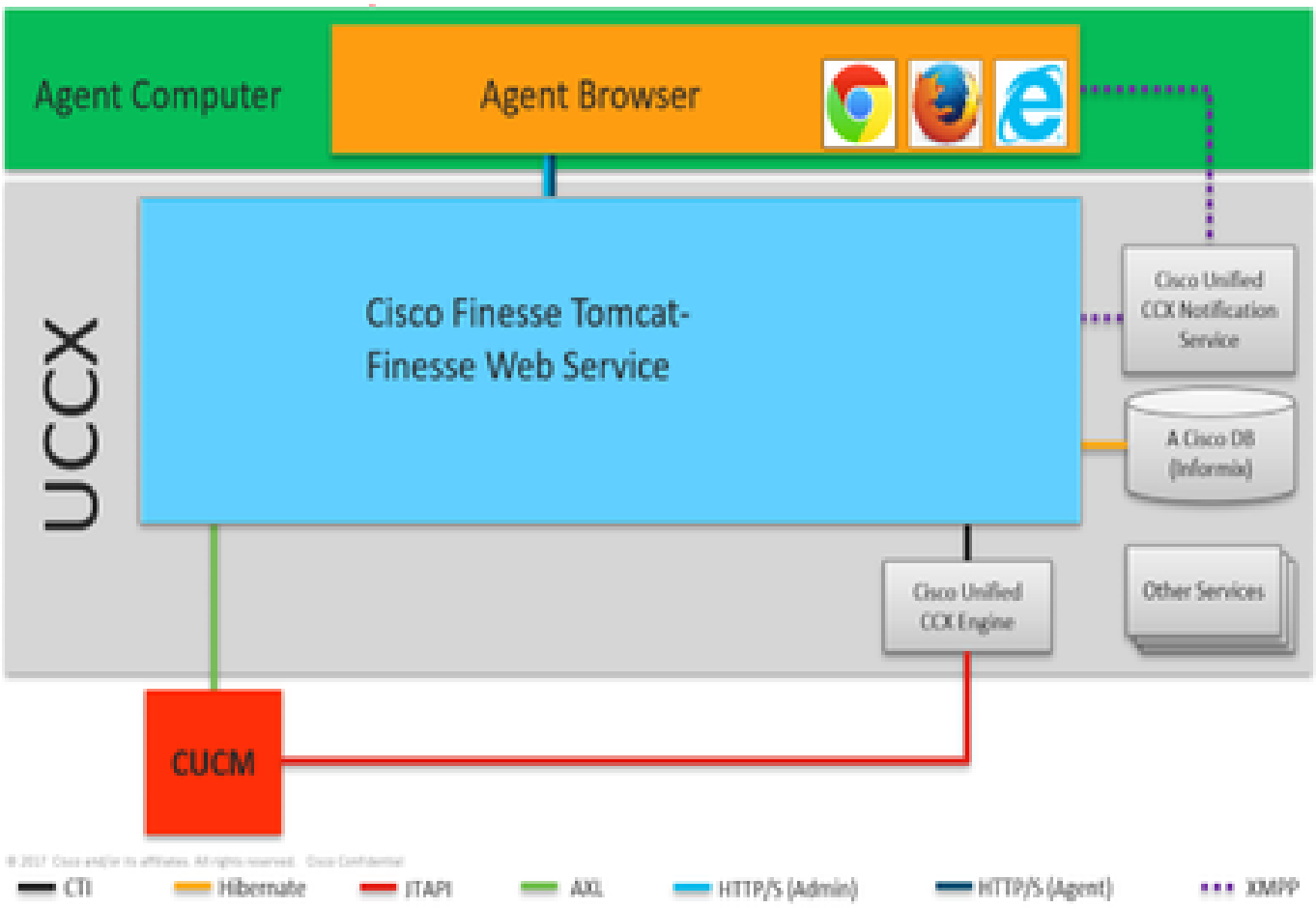
- Umschlag - Dieses obligatorische Element ist der Stamm der SOAP-Nachricht und identifiziert die übertragene XML als ein SOAP-Paket. Ein Umschlag enthält einen Hauptabschnitt und einen optionalen Kopfabschnitt.

- Header - Dieses optionale Element stellt einen Erweiterungsmechanismus bereit, der die Verarbeitungsinformationen für die Nachricht angibt. Wenn für den Vorgang, der die Nachricht verwendet, z. B. Sicherheitsanmeldeinformationen erforderlich sind, müssen diese Anmeldeinformationen Teil des Envelope Headers sein.

- Body - Dieses Element enthält die Nutzlast der Nachricht, wobei die Rohdaten zwischen der sendenden und der empfangenden Anwendung übertragen werden. Der Text selbst kann aus mehreren untergeordneten Elementen bestehen, wobei die Struktur dieser Daten in der Regel durch ein XML-Schema definiert wird.

20000 ft

Im folgenden Diagramm werden die Protokolle der Finesse-Architektur genauer erläutert.



20000 Ansicht

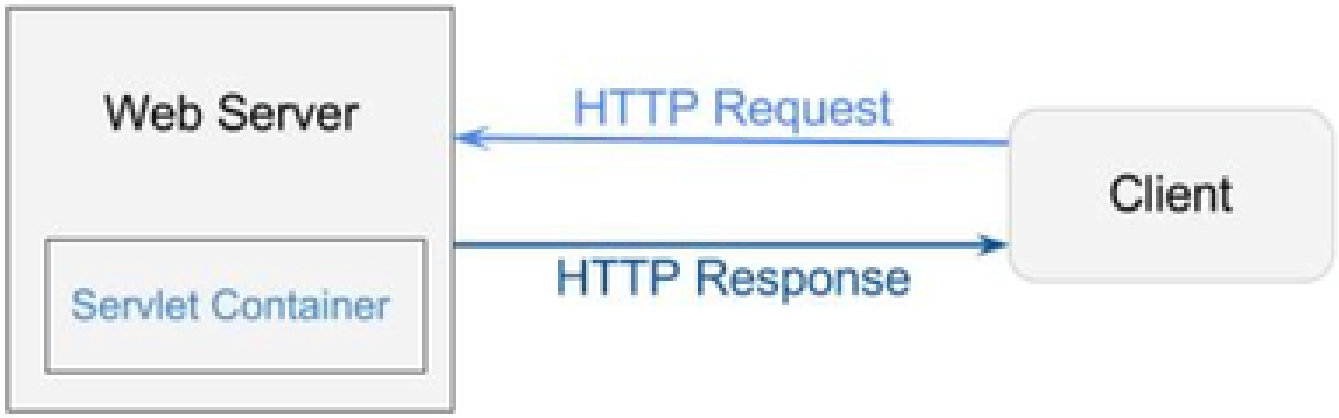
Diese Protokolle sind für die Kommunikation zwischen verschiedenen UCCX-Komponenten verantwortlich.

- UCCX Engine und Finesse Server kommunizieren über das CTI-Protokoll.

- UCCX-Engine und CUCM kommunizieren über das JTAPI-Protokoll.

- Finesse Tomcat und CUCM kommunizieren über AXL.
- Finesse Notification Service und Agent Browser kommunizieren auf XMPP/BOSH.
- Finesse Tomcat und die Datenbank unterhalten sich über Hibernate.
- Finesse Tomcat und Finesse Notification kommunizieren über XMPP.

APACHE SHINDIG



Shindig

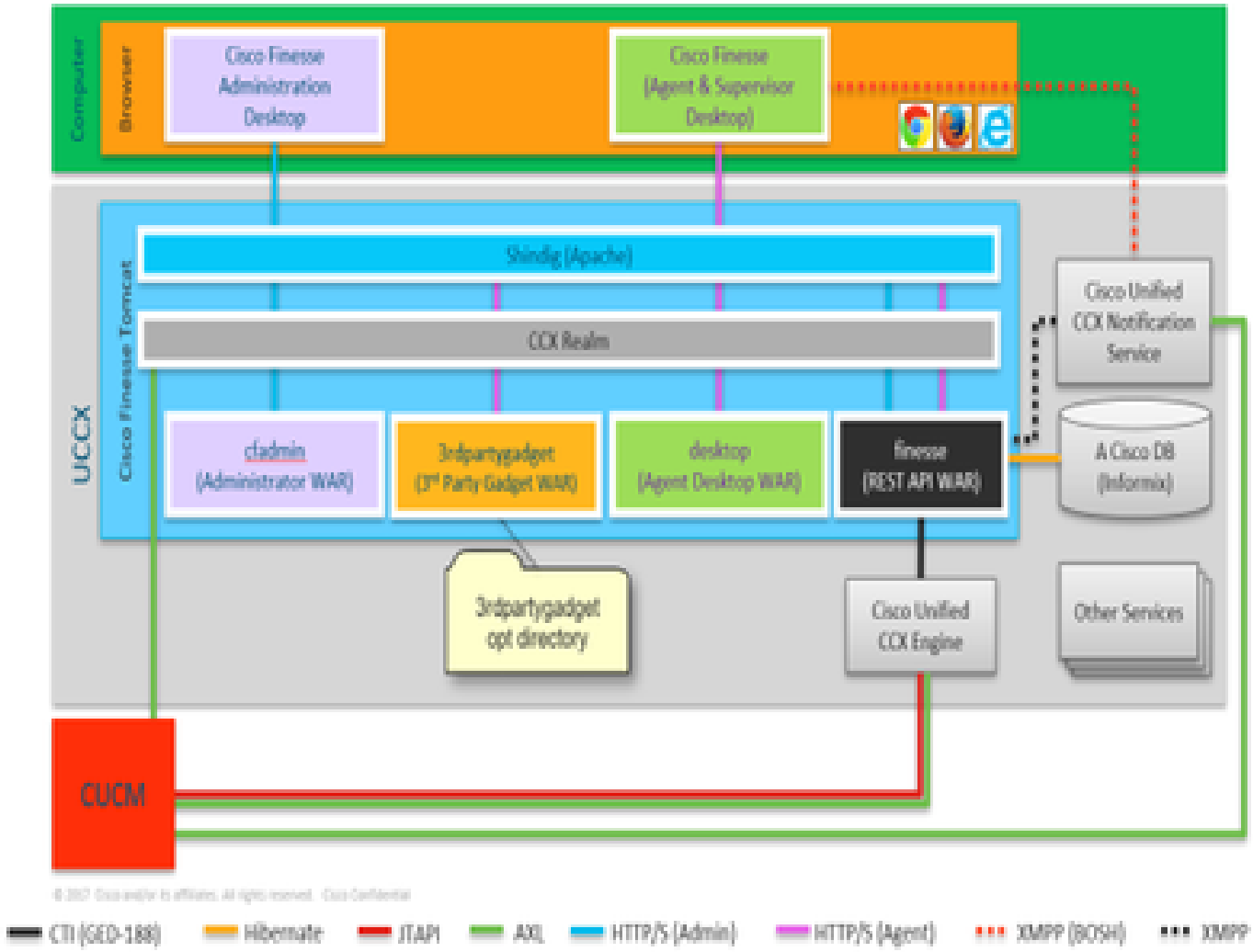
Apache Shindig ist ein OpenSocial-Container und hilft Ihnen, OpenSocial-Apps schnell zu hosten, indem der Code bereitgestellt wird, um Gadgets, Proxy-Anfragen und REST- und RPC-Anfragen zu bearbeiten. OpenSocial ist eine Reihe von APIs zum Erstellen von Social Media-Anwendungen, die im Internet ausgeführt werden. (Web/Servlet) Container wird von einem Webserver zum dynamischen Generieren von Webseiten verwendet.

KRIEGSDATEIEN

WAR steht für Web Archive. Es enthält Dateien eines Webprojekts. Es kann Servlet, XML, JSP, Bild, HTML, CSS, JS, und so weiter haben. Die Catalina-Protokolle enthalten Informationen über die Bereitstellung von WARs.

10000 ft

Im nächsten Diagramm wird detailliert erläutert, wie der Authentifizierungsfluss innerhalb der Komponenten von UCCX und Finesse funktioniert.



10000 Ansicht

WAR-Dateien sind erforderlich, um die Seite anzuzeigen und zu erstellen, je nachdem, wie Sie sich anmelden. Browser fragt Shindig, dass es ein Gadget rendern muss, shindig spricht dann mit CUIC, um das Gadget zu rendern. Der CCX-Bereich wird für die Authentifizierung mit CUCM über AXL verwendet. Der Benachrichtigungsdienst authentifiziert sich auch mit CUCM und verwendet AXL.

Finesse Rest API WAR ist das Haupt-Repository, das tatsächlich kommuniziert mit Benachrichtigungs-Service, CCX-Engine und DB. Shindig spricht nur mit Finesse Rest API (WebServices), da cfadmin und Desktop WARs sind nur auf die Anzeige der Seite. Alles, was in die Finesse Rest API WAR kommt, können Sie sehen, dass in den Finesse WebServices-Logs, die die wichtigsten Protokolle für Finesse sind. Sie sprechen HTTP zwischen Shindig und Finesse Web-Service (Rest API WAR). Finesse Web Service (Rest API WAR) und Engine kommunizieren über CTI miteinander.

AJAX - Die Schönheit von Finesse

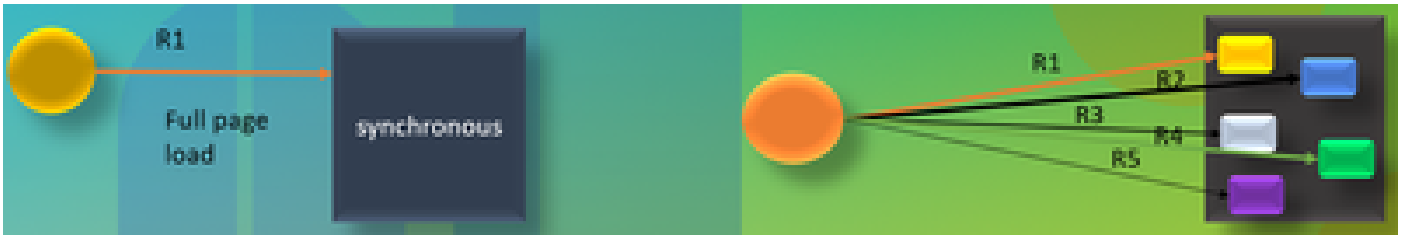
AJAX steht für Asynchronous Javascript und XML. Es ist keine Programmiersprache, sondern eine Methode, um von einer Webseite aus auf Webserver zuzugreifen. AJAX ist ein Mechanismus zum Durchführen von Teilaktualisierungen von Seiten. Es ermöglicht Ihnen, Abschnitte einer Seite mit Daten zu aktualisieren, die von einem Server stammen, ohne die gesamte Seite aktualisieren zu müssen.

Zum Beispiel, wenn Sie über Facebook Messenger sprechen, wenn eine neue Nachricht kommt, müssen Sie nicht die ganze Seite aktualisieren, um die Nachricht zu erhalten, sondern der Nachrichtenabschnitt selbst aktualisiert und bekommt die neuen Nachrichten in Echtzeit, ohne die ganze Seite aktualisieren zu müssen.

Jeder Browser hat ein eingebautes Objekt namens XMLHTTPREQUEST (auch **XHR** genannt). Jede Anforderung an AJAX im Server durchläuft diese XML-Anforderung. Hier finden Sie Einzelheiten zu den zu aktualisierenden Komponenten.

Vorteile der Verwendung von AJAX

Im nächsten Diagramm wird der Unterschied zwischen asynchronen und synchronen Anforderungen erläutert.



AJAX

Bei einer synchronen Anfrage müssen Sie warten, bis die erste Anfrage verarbeitet wird, und können dann eine zweite Anfrage senden. Beispielsweise ist eine Seitenaktualisierung erforderlich, und Sie können erst dann etwas unternehmen, wenn die Seite aktualisiert wurde. Andererseits müssen Sie im Fall einer asynchronen Anforderung nicht warten, bis die erste Anforderung abgeschlossen ist, um die zweite Anforderung zu senden. Sie können mehrere Anforderungen gleichzeitig senden. Zum Beispiel Wetter-App-Gadgets auf Websites. Sie können den Wetterbereich der Seite aktualisieren und gleichzeitig auch an den anderen Bereichen der Website arbeiten, ohne die gesamte Seite aktualisieren zu müssen. Dies ist der Hauptvorteil von Asynchronous Request.

ARBEITEN VON AJAX

AJAX ist eine Kombination aus einem XMLHTTPREQUEST (**XHR)**, der zum Senden und Empfangen von Updates vom Webserver zusammen mit Javascript und HTML verwendet wird, die zur Anzeige oder Verwendung der Daten verwendet werden.

ANFORDERUNG MIT AJAX AN SERVER SENDEN

Dies ist ein dreistufiger Prozess, der als Nächstes erwähnt wird:

1. Erstellen einer Variablen und Speichern des **XHR**-Objekts in ihr.

Var request = new XMLHttpRequest();

2. Zugriff auf die Anforderungsvariable, die die Nutzlast innerhalb des XHR-Objekts hat.

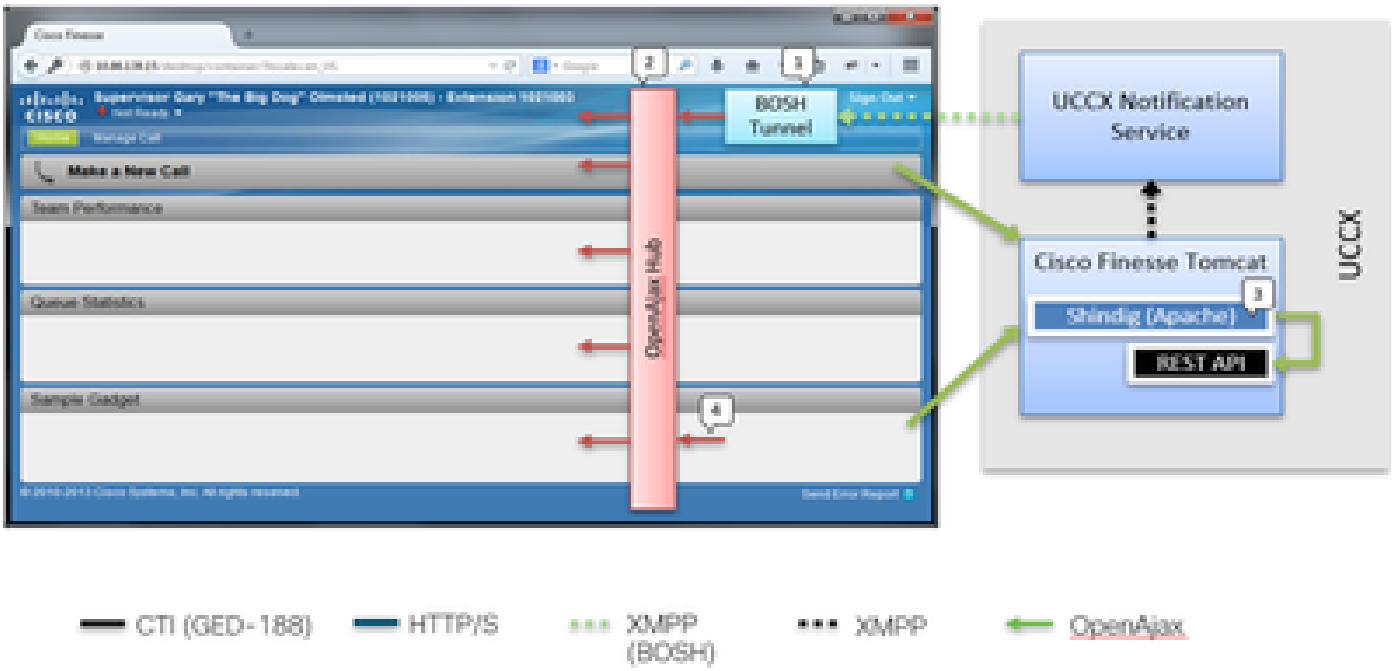
Anfrage.**open(GET, URL)**;

3. Versenden des Antrags

Request.send()

Desktop-Architektur

Im nächsten Diagramm wird der Fluss von AJAX-Signalen bei der Wiedergabe von Gadgets auf der Webseite erläutert.

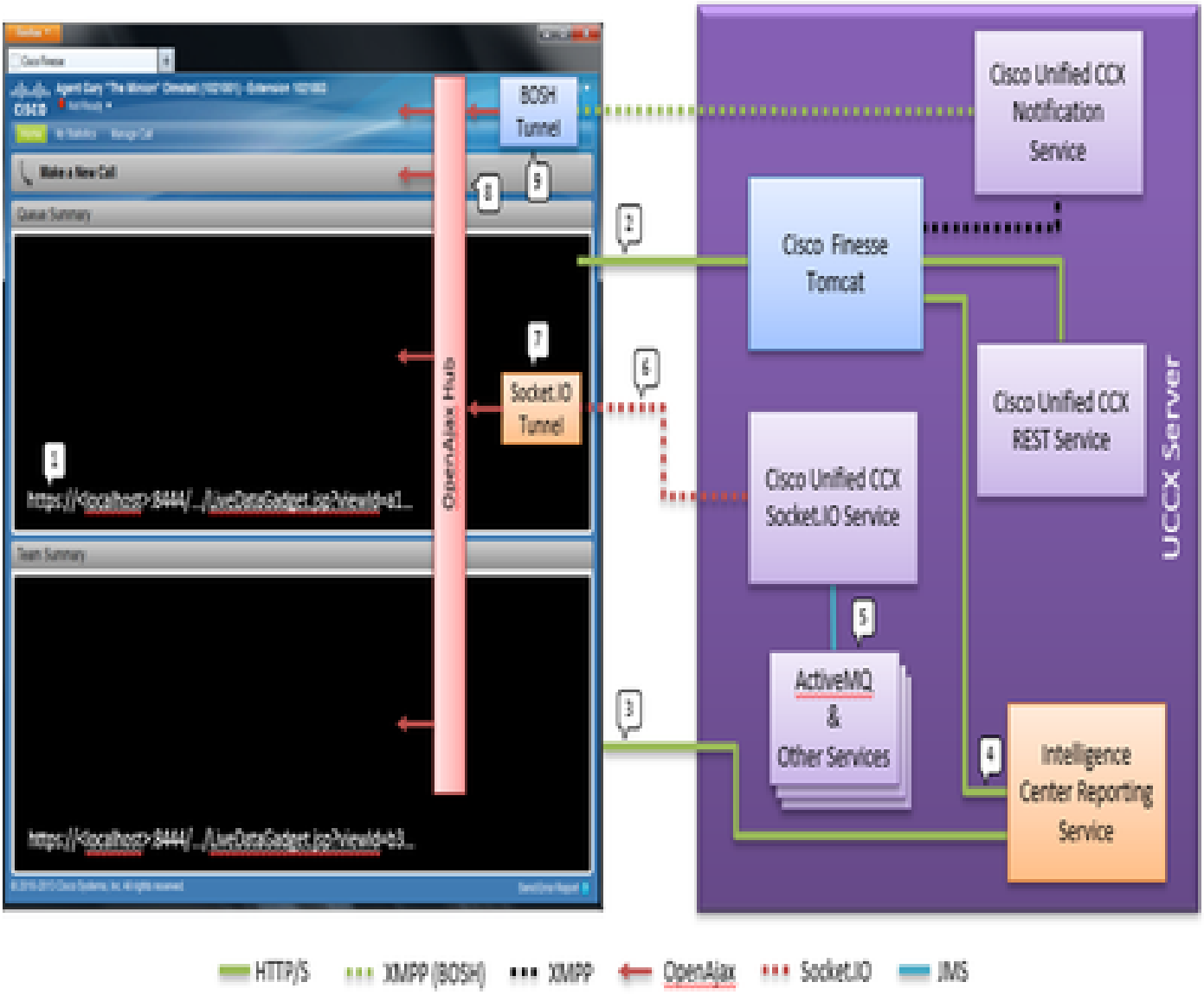


Desktop-Architektur

IFrame befindet sich im Container, in dem der BOSH-Tunnel gehostet wird. OPENAJAX-Hub wird bereitgestellt, um Nachrichten über die Gadgets (mit der pubsub-Methode) zu veröffentlichen. REST-Anfragen werden ebenfalls über Shindig an andere Server weitergeleitet. Gadgets können ihre eigenen Nachrichten auf dem AJAX-Hub veröffentlichen.

Gadget-Architektur

Im folgenden Diagramm wird die Architektur des Finesse Gadgets ausführlich erläutert.



Gadget-Architektur

Im Gegensatz zu typischen Gadgets erhalten CUIC Gadgets auch einen Echtzeit-XMPP-Feed von OpenFire. Im Fall von UCCX, bei dem CUIC und Finesse gemeinsam mit UCCX genutzt werden, gibt es eine gemeinsam genutzte OpenFire-Instanz. Der Großteil der Gadget-Inhalte und alle REST-APIs werden über Shindig im Finesse Server bereitgestellt. Dies gilt für Finesse-Gadgets und REST-API sowie für CUIC-Gadget-Instanzen und REST-API. CUIC-Gadgets verwenden ein D-Grid für die Wiedergabe ihrer Berichte. Es muss ein Bootstrapping-Prozess stattfinden, der direkt mit CUIC durchgeführt wird. Aus diesem Grund kommunizieren die CUIC-Gadgets während des Ladevorgangs zunächst direkt mit dem CUIC-Server. Aus diesem Grund muss das CUIC-Zertifikat (zusätzlich zum Socket.IO-Tunnel) im Benutzerbrowser akzeptiert werden. Der Gadget-Inhalt und die REST-APIs werden dem Client zwischen Finesse und CUIC als Proxys hinzugefügt. Rest-API-Aufrufe werden sowohl an den Intelligence Center Reporting Service als auch an den CCX-Webdienst getätigt. Der CCX Live Data Socket.IO Service ruft die Nachrichten von Live Data über JMS von ActiveMQ ab. Der CCX Live Data Socket.IO-Dienst veröffentlicht Real-Time Reporting JSON über die Socket.IO-Verbindung vom Client. Ähnlich wie der Finesse Desktop über einen BOSH Tunnel iFrame verfügt, der die BOSH-Verbindung mit dem Cisco Finesse Notification Service aufrechterhält, verfügt das Master-Live Data-Gadget über einen Socket.IO Tunnel iFrame, der die Socket.IO (WebSocket)-Verbindung mit dem CCX Live Data Socket.IO Service aufrechterhält.

Der OpenAjax Hub verteilt alle Ereignisse an die abonnierten Listener. Dies wären sowohl Gadgets als auch Teile des Finesse Containers selbst. Finesse Desktop verfügt über einen BOSH Tunnel iFrame, der die BOSH-Verbindung mit dem Cisco Unified CCX Notification Service aufrechterhält. Damit werden Veranstaltungen auf dem OpenAjax Hub veröffentlicht.

Referenzlinks

- Finesse Web Services - Entwicklerhandbuch

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.