Comprendre l'architecture UCCX Finesse

Table des matières

Introduction

Ce document décrit l'architecture Finesse de manière approfondie afin que les processus sous-jacents aient un sens lors du dépannage des problèmes de finesse.

Conditions préalables

Exigences

Cisco recommande de connaître ces outils et fonctionnalités :

JTAPI - API de téléphonie Java

API - Interface de programmation d'applications

UCCX - Unified Contact Center Express

CUCM - Cisco Unified Communications Manager

CTI - Intégration de la téléphonie informatique

Composants utilisés

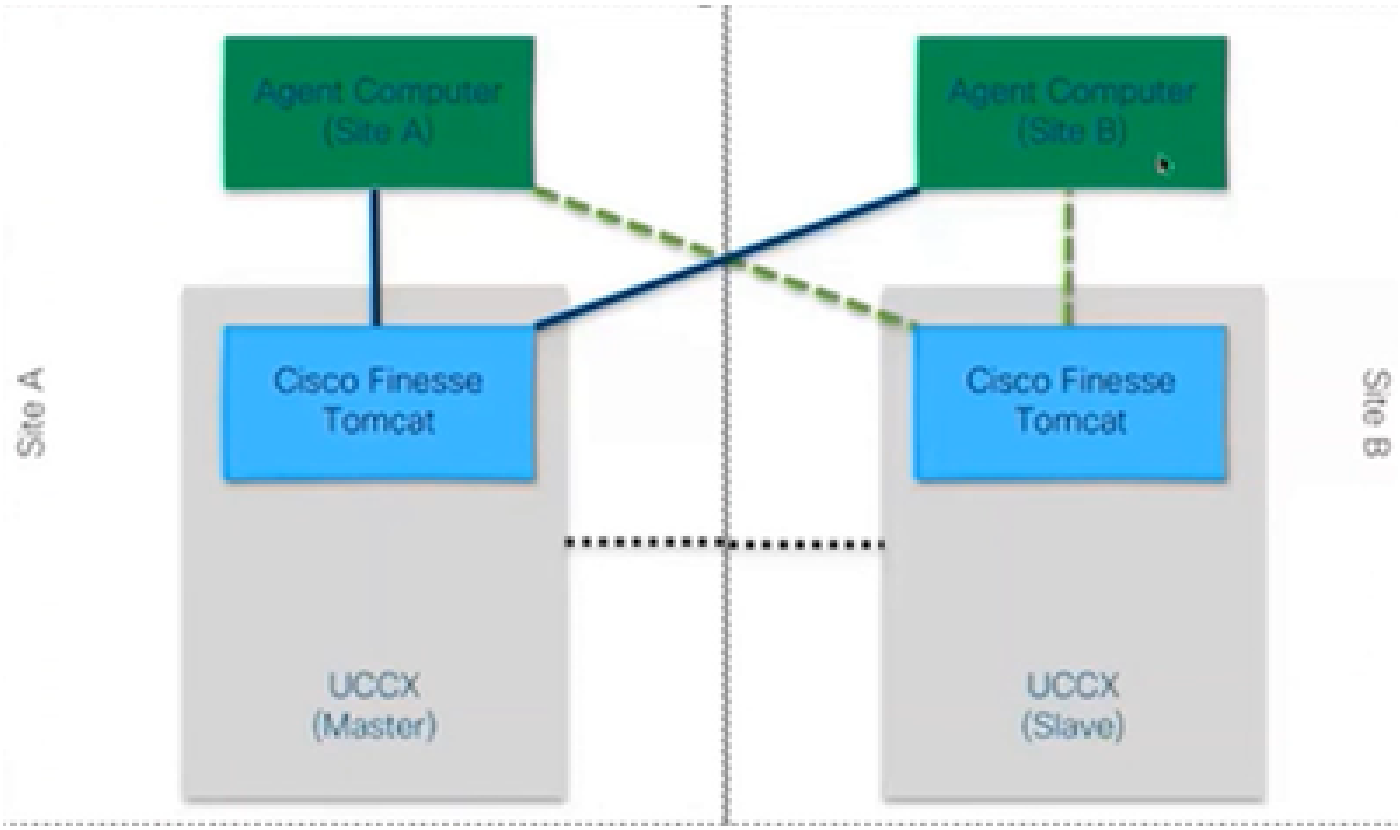
- Cisco Unified Contact Center Express (UCCX)

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

Ce document décrit l'architecture Finesse en partant d'une vue d'ensemble de haut niveau, puis du flux de signal en profondeur, avec des exemples et des diagrammes.

Vue à 15 000 m



50000 vue

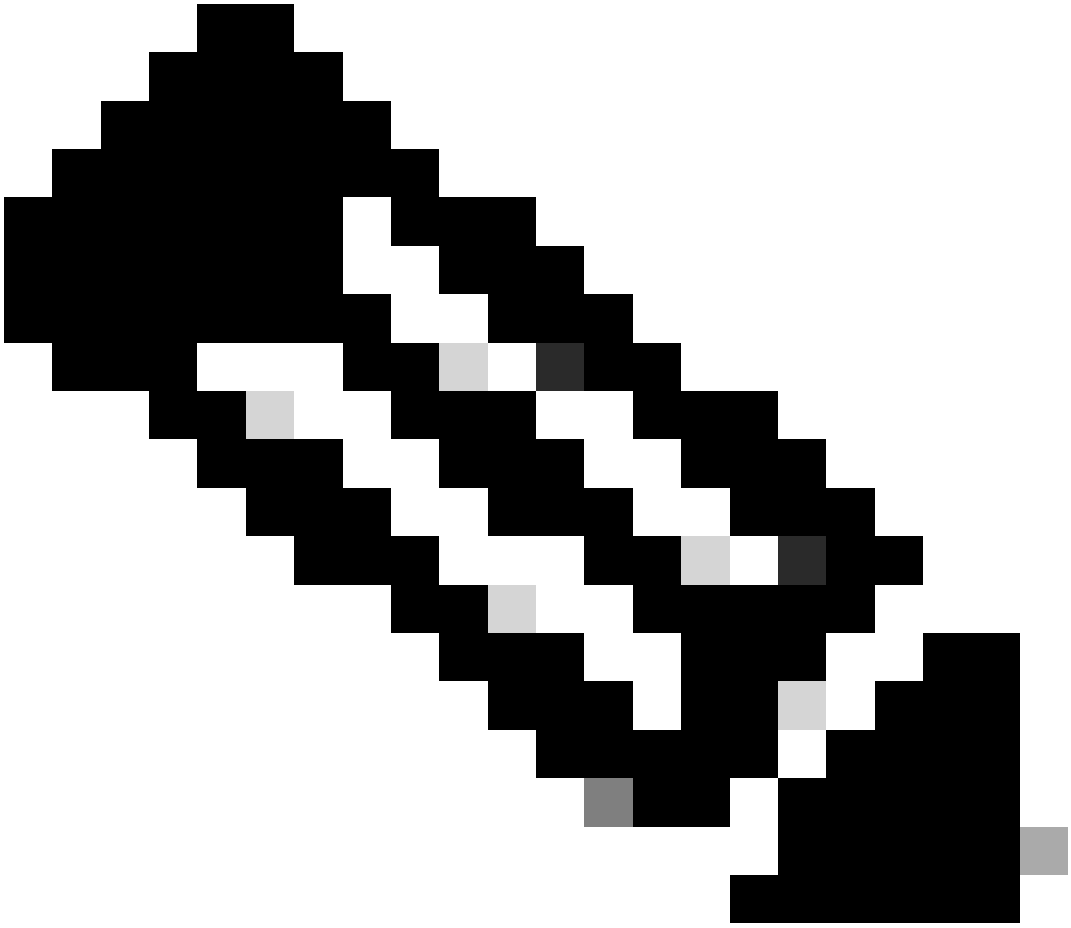
Finesse Tomcat

Finesse Tomcat est similaire à Cisco Tomcat dans CUCM, car la fonctionnalité est identique pour charger les pages Web, mais pour un service différent appelé Finesse. Tomcat a été conçu pour Finesse car il s'agit d'une application Web distincte. Vous ne pouvez vous connecter à finesse qu'en utilisant le noeud maître CCX selon les versions antérieures à 11.5. À partir de la version 11.6, vous pouvez vous connecter à n'importe quel noeud (mais ce n'est pas recommandé), uniquement pendant le basculement. Ainsi, si vous considérez un cluster WAN, où les deux agents (sur le site A et le site B) sont connectés à Finesse sur le noeud maître, alors à tout moment, il y a une connexion inactive à l'autre noeud depuis Cisco Finesse vers le moteur , qui est une demande CTI openconf qui est requise pour le basculement.

La requête Openconf est envoyée régulièrement pour vérifier si le noeud est actif ou en veille. En cas de basculement, le service de notification utilise une API appelée l'API systeminfo qui indique au client que ce noeud est arrêté et qu'un basculement est nécessaire. Ensuite, un fichier s'exécute qui redirige le navigateur vers l'autre noeud, puis la réponse Javascript openconf est envoyée. Ce basculement ne fonctionne que si les certificats sont acceptés. (pour établir une connexion sécurisée au serveur).

3 certificats sont présentés qui sont :

- Certificat de service de notification pour ce noeud.
- Certificat de service de notification pour le noeud distant.
- Certificat de service Finesse pour le noeud distant.



Remarque : les certificats de noeud distant doivent être acceptés pour que le basculement fonctionne.

HTTP(S)

Il s'agit d'un protocole de couche application pour l'envoi de documents hypermédia tels que HTML. Il est conçu pour la communication entre les navigateurs Web et les serveurs Web. Il s'agit d'un protocole sans état qui signifie que le serveur ne conserve aucune donnée entre les deux requêtes. Il s'agit d'un protocole client-serveur. Pour UCCX, le client Finesse s'exécute sur le navigateur de l'agent (PC). Il envoie une requête au serveur via HTTP. Voici quelques méthodes HTTP courantes :

GET : pour obtenir des informations à partir d'un serveur.
POST : pour envoyer des informations à un serveur.

PUT : pour remplacer tout élément sur un serveur.
DELETE : pour supprimer des informations d'un serveur.

Finesse utilise les requêtes api systeminfo dans la requête http. Par exemple, si vous souhaitez modifier l'état d'un agent, le navigateur envoie un PUT au lieu d'un POST, car si ce dernier est envoyé, le serveur devient confus car il a 2 états en main, lequel sélectionner ? Donc, en utilisant PUT, il remplace l'état actuel.

XMPP

Protocole eXtensible Messaging and Presence

Il s'agit d'un ensemble de protocoles utilisés pour la messagerie instantanée et la présence. Toutes les entités XMPP sont identifiées à l'aide de leurs ID ou JID Jabber. L'une des extensions de ce protocole XMPP utilisé pour les gadgets est appelée PUBSUB.

PUBSUB

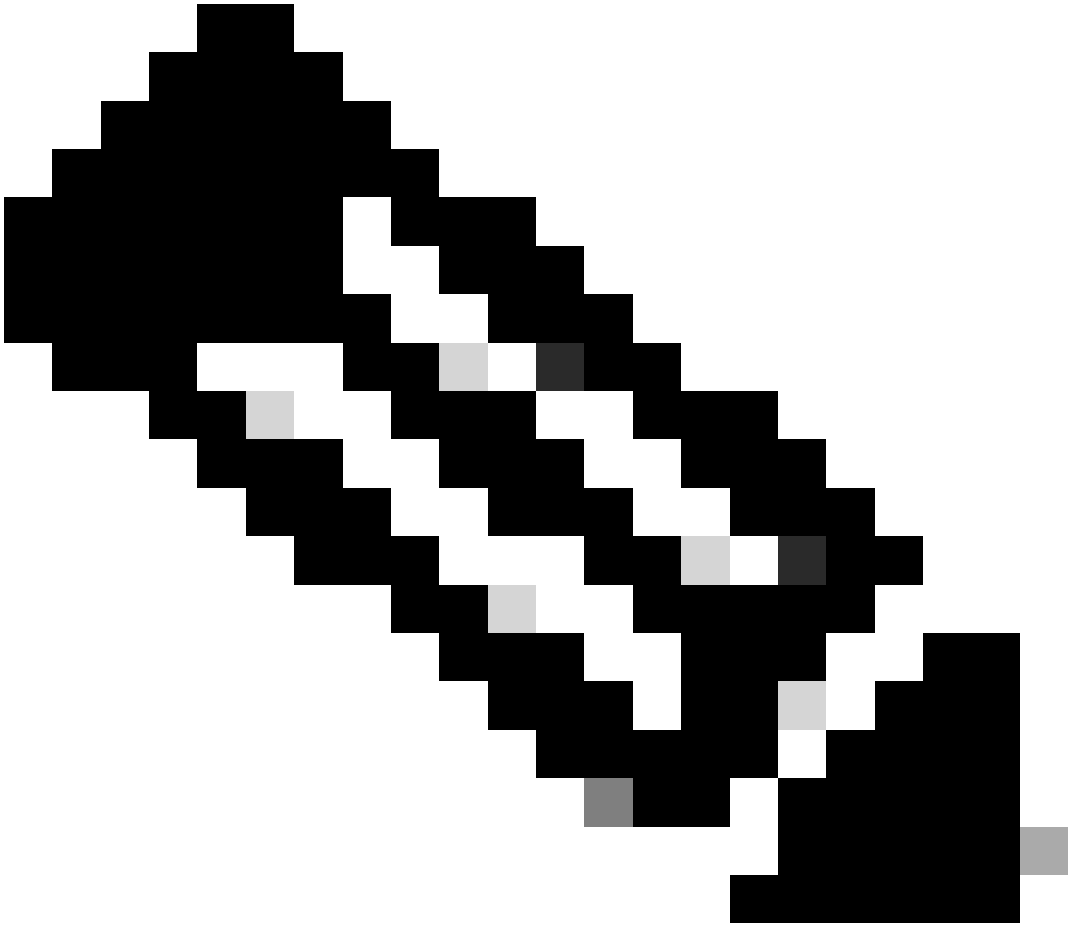
Ce n'est pas l'abonné éditeur auquel vous pouvez penser. Il publie et s'abonne toujours, mais il n'a rien à voir avec les bases de données. XMPP utilise un mécanisme appelé noeuds. Noeud surveille essentiellement l'entité qui vous tient à coeur. Tout ce qui est important pour vous et que vous souhaitez surveiller, vous y ajoutez un noeud. Ensuite, ce que fait PUBSUB, c'est que vous vous abonnez aux mises à jour ou notifications sur ce noeud. Vous pouvez avoir des noeuds pour chaque type d'entité comme agent, dialogue, etc. Si vous créez un noeud pour un agent, vous êtes abonné au noeud sur cet agent, puis quoi qu'il fasse, vous en êtes averti.

L'objectif de cette spécification est de permettre au serveur XMPP (service de notification) d'obtenir des informations publiées sur les noeuds XMPP (rubriques), puis d'envoyer des événements XMPP aux entités abonnées à ce noeud.

Dans le cas de Finesse, le serveur CTI (Computer Telephony Integration) envoie des messages CTI au service Web Finesse pour informer Finesse des mises à jour de configuration telles que, mais sans s'y limiter, la création d'agent ou de file d'attente de service de contact (CSQ) ou des informations sur un appel. Ces informations sont ensuite converties en un message XMPP que le service Web Finesse publie auprès du service de notification Finesse.Le service de notification Finesse envoie ensuite des messages XMPP sur BOSH aux agents qui sont abonnés à certains noeuds XMPP.

BOSH - Flux bidirectionnels sur HTTP synchrone

BOSH est une connexion HTTP de longue durée dans laquelle le serveur conserve la requête plus longtemps jusqu'à ce qu'il y réponde, sinon il envoie une réponse vide. Cela fonctionne pour les clients XMPP et les serveurs XMPP, mais peut également être utilisé pour des applications non XMPP.



Remarque : XMPP est avec état alors que HTTP est sans état (il ne stocke pas les informations relatives à la dernière requête)

Si une application Web doit fonctionner avec XMPP, plusieurs problèmes surviennent.

Problème 1 : les navigateurs ne prennent pas en charge XMPP sur TCP (Transmission Control Protocol) en mode natif.

Solution 1 : les serveurs et les navigateurs Web communiquent via des messages HTTP (HyperText Transfer Protocol), de sorte que Finesse et d'autres applications Web encapsulent les messages XMPP à l'intérieur des messages HTTP.

Problème 2 : HTTP est un protocole sans état.

Solution 2 : Vous pouvez utiliser des cookies/données de publication pour cela.

Problème 3 : Le troisième problème est le comportement unidirectionnel de HTTP, ce qui signifie que seul le client envoie des requêtes et que le serveur ne peut que répondre. L'incapacité du serveur à transmettre des données rend anormale la mise en oeuvre de XMPP sur HTTP.

Solution 3 : pour résoudre ce problème, vous devez disposer d'un pont entre HTTP et XMPP.

Les solutions proposées sont les suivantes :

- Interrogation (protocole hérité) : requêtes HTTP répétées demandant de nouvelles données définies : Interrogation HTTP

- L'interrogation longue est également appelée BOSH : protocole de transport qui émule la sémantique d'une connexion TCP bidirectionnelle à longue durée de vie entre deux entités en utilisant efficacement plusieurs paires requête/réponse HTTP synchrones sans nécessiter l'utilisation d'interrogations fréquentes. La raison d'utiliser BOSH est de dissimuler le fait que le serveur n'a pas à répondre dès qu'il y a une demande. La réponse est retardée jusqu'à une durée spécifiée jusqu'à ce que le serveur dispose de données pour le client, puis elle est envoyée en tant que réponse. Dès que le client reçoit la réponse, il fait une nouvelle demande, etc.

Le client de bureau Finesse (application Web) établit une connexion BOSH périmée sur le port TCP 7443 toutes les 30 secondes. Au bout de 30 secondes, en l'absence de mises à jour du service de notification Finesse, le service de notification envoie une réponse HTTP avec un 200 OK et un corps de réponse (presque) vide. Si le service de notification dispose d'une mise à jour sur la présence d'un agent ou d'un événement de dialogue (appel), par exemple, les données sont envoyées immédiatement au client Web Finesse.

Pour récapituler :

Le client Web Finesse dispose d'une connexion HTTP périmée (http-bind) configurée sur le serveur Finesse via le port TCP 7443. C'est ce qu'on appelle un sondage long BOSH. Le service de notification Finesse est un service de présence qui publie des mises à jour concernant l'état d'un agent, d'un appel, etc. Si le service de notification dispose d'une mise à jour, il répond à la requête http-bind avec la mise à jour d'état sous la forme d'un message XMPP dans le corps de la réponse HTTP. S'il n'y a aucune mise à jour d'état 30 secondes après la réception de la demande http-bind, le service de notification répond sans aucune mise à jour d'état pour permettre au client Web Finesse d'envoyer une autre demande http-bind. Cela permet au service de notification de savoir que le client Web Finesse est toujours en mesure de se connecter au service de notification et que l'agent n'a pas fermé son navigateur ou mis son ordinateur en veille, etc.

CTI

Vous pouvez utiliser CTI (Computer Telephony Integration) pour tirer parti des fonctions de traitement informatique lors de l'établissement, de la réception et de la gestion des appels téléphoniques. Les applications CTI vous permettent d'effectuer des tâches telles que la récupération d'informations utilisateur à partir d'une base de données à l'aide d'un ID d'appelant, ou de travailler avec les informations collectées par un système de réponse vocale interactif (IVR) pour acheminer un appel provenant d'un utilisateur avec leurs informations, vers le représentant du service approprié. CTI Manager sur CUCM répond aux requêtes JTAPI d'UCCX. Le port TCP du serveur CTI est 12018. C'est ainsi que Finesse Server et le moteur (serveur CTI) communiquent entre eux.

Voici quelques-unes des informations échangées via CTI :

- Configuration actuelle du système et mises à jour futures.

- Agents et leurs états.

- Appels et leurs états.

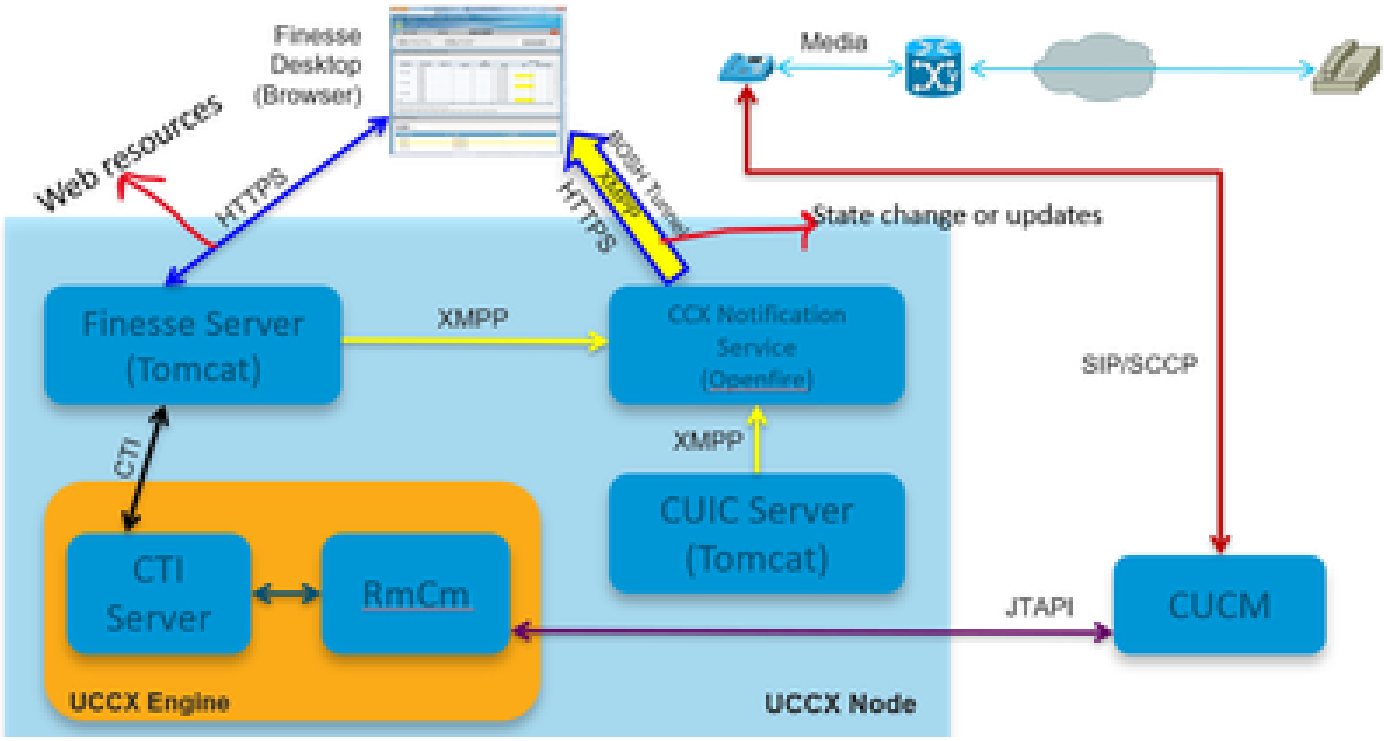
- Statistiques des agents, appels et files d'attente en temps réel.

JTAPI

Cisco Unified JTAPI est une norme d'interface de programmation développée par Sun Microsystems pour une utilisation avec des applications de téléphonie et d'informatique Java. Cisco JTAPI met en oeuvre la spécification Sun JTAPI 1.2 avec des extensions Cisco supplémentaires. Toute communication entre UCCX et CUCM réside sur JTAPI. C'est ainsi que CUCM et le moteur (sous-système de téléphonie) communiquent entre eux. JTAPI est utilisé pour contrôler et surveiller les téléphones CUCM, acheminer les appels à l'aide des ports CTI et des points de routage, démarrer et arrêter les enregistrements sur CUCM et pour toutes les fonctionnalités de routage d'appels

Vue de 30000 pieds

Le schéma suivant décrit comment le moteur UCCX, Finesse, CUCM et le navigateur communiquent entre eux.

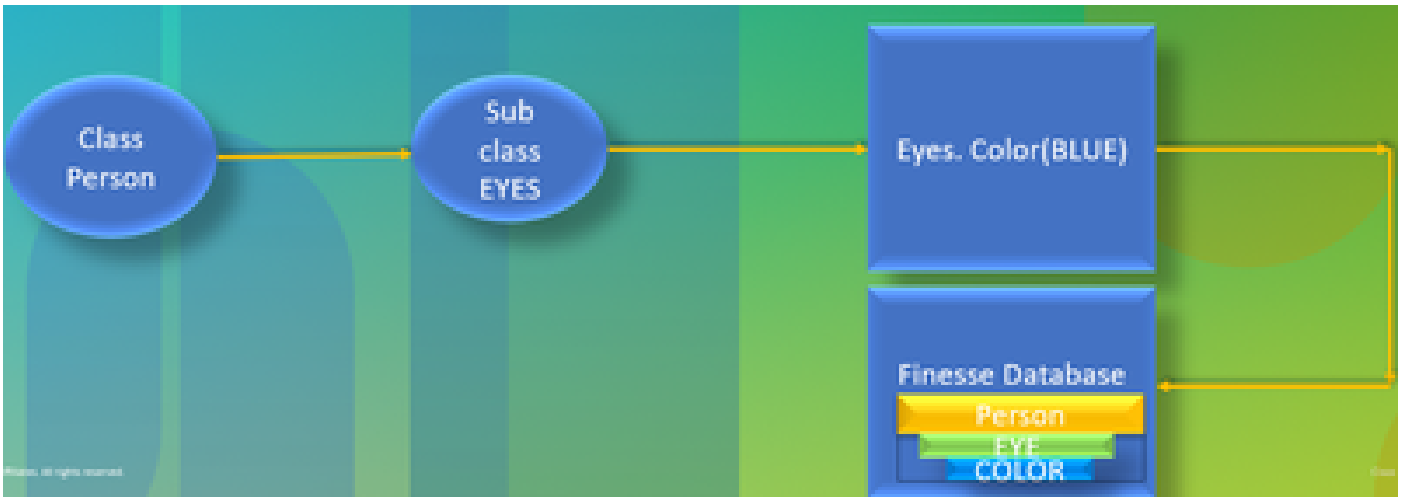


30000 vue

Considérons que l'appel est établi avec l'agent. Maintenant, RmCm, qui surveille l'extension de l'agent via JTAPI, informe le serveur CTI du changement d'état en cours de conversation de l'agent. Ces informations sont envoyées du serveur CTI (à l'intérieur du moteur CCX) au serveur Finesse (Tomcat) à l'aide de CTI. Le serveur Finesse envoie ces informations au service de notification CCX à l'aide de XMPP concernant le changement d'état. Le service de notification (Openfire) ouvre un tunnel BOSH vers le navigateur de l'agent et met à jour les informations sur le changement d'état et c'est ainsi que vous voyez l'agent passer à l'état RESERVED. Tout type de ressources web est demandé au serveur finesse en utilisant HTTPS comme les fichiers WAR, les gadgets et ainsi de suite (si pas dans le cache déjà).

HIBERNER

Le schéma suivant explique le service de veille prolongée.



hiberner

HIBERNATE est appelé Service de requête et de persistance relationnelle/objet hautes performances. En bref, il mappe les classes JAVA aux tables de base de données. Par exemple, vous avez un objet JAVA appelé Équipe et vous avez une table de base de données dans la base de données Finesse appelée Équipe. La classe JAVA contrôle les informations contenues dans la table et HIBERNATE est ce qui permet d'y parvenir. Au lieu d'utiliser des requêtes SQL, il utilise des classes Java pour mettre à jour les informations.

AXL

XML administratif.

XML est l'acronyme de eXtensible Markup Language et est un langage de balisage qui définit des règles relativement simples pour le codage des données. Il a été principalement conçu pour transmettre et recevoir des données structurées dans un format bien défini que les deux systèmes peuvent comprendre. Dans la forme la plus simple, XML définit des balises qui sont placées entre crochets (<>) et ces balises entourent les données décrites par la balise. Les balises peuvent former une hiérarchie avec des balises à l'intérieur d'autres balises. Par exemple, pour définir un périphérique téléphonique de base, vous pouvez dire qu'un périphérique téléphonique a besoin de trois paramètres : un nom, une description et un numéro de téléphone.

Il s'agit d'une API SOAP qui active le provisionnement à distance sur CUCM. Il permet d'ajouter, de mettre à jour, de supprimer ou d'extraire des informations de la base de données CUCM. Les fonctionnalités de récupération incluent la vérification de l'authentification des utilisateurs et l'exécution de requêtes SQL.L'API AXL vous permet d'accéder à l'ensemble de la base de données CUCM. L'API AXL est uniquement destinée au provisionnement et ne permet pas d'accéder aux données d'exécution ou de performances.

L'API AXL utilise l'authentification HTTPS de base. Tout utilisateur CUCM (Application ou Utilisateur final) dispose d'un accès en lecture/écriture via AXL s'il est membre du groupe de contrôle d'accès **Supérieurs utilisateurs CCM standard** ou de tout groupe auquel est affecté le rôle **Accès API AXL standard**. Cela signifie que tous les comptes de super-utilisateur ont implicitement déjà accès à l'API AXL. Pour créer un compte dédié à l'utilisation de l'API AXL, vous devez d'abord créer un groupe de contrôle d'accès et lui attribuer le rôle **Standard AXL API Access**, puis associer l'utilisateur de l'application au groupe nouvellement créé. Pour fournir un accès en lecture seule à l'API AXL, vous pouvez créer un groupe de contrôle d'accès distinct et lui attribuer uniquement le rôle **Standard AXL Read Only API Access**.

SAVON

Le protocole SOAP (Simple Object Access Protocol) est un moyen de transmettre des informations entre les applications dans un format XML. Les messages SOAP sont transmis de l'application émettrice à l'application réceptrice, généralement via une session HTTP. Le message SOAP

réel se compose de l'élément Envelope, qui contient un élément Body et un élément Header facultatif.

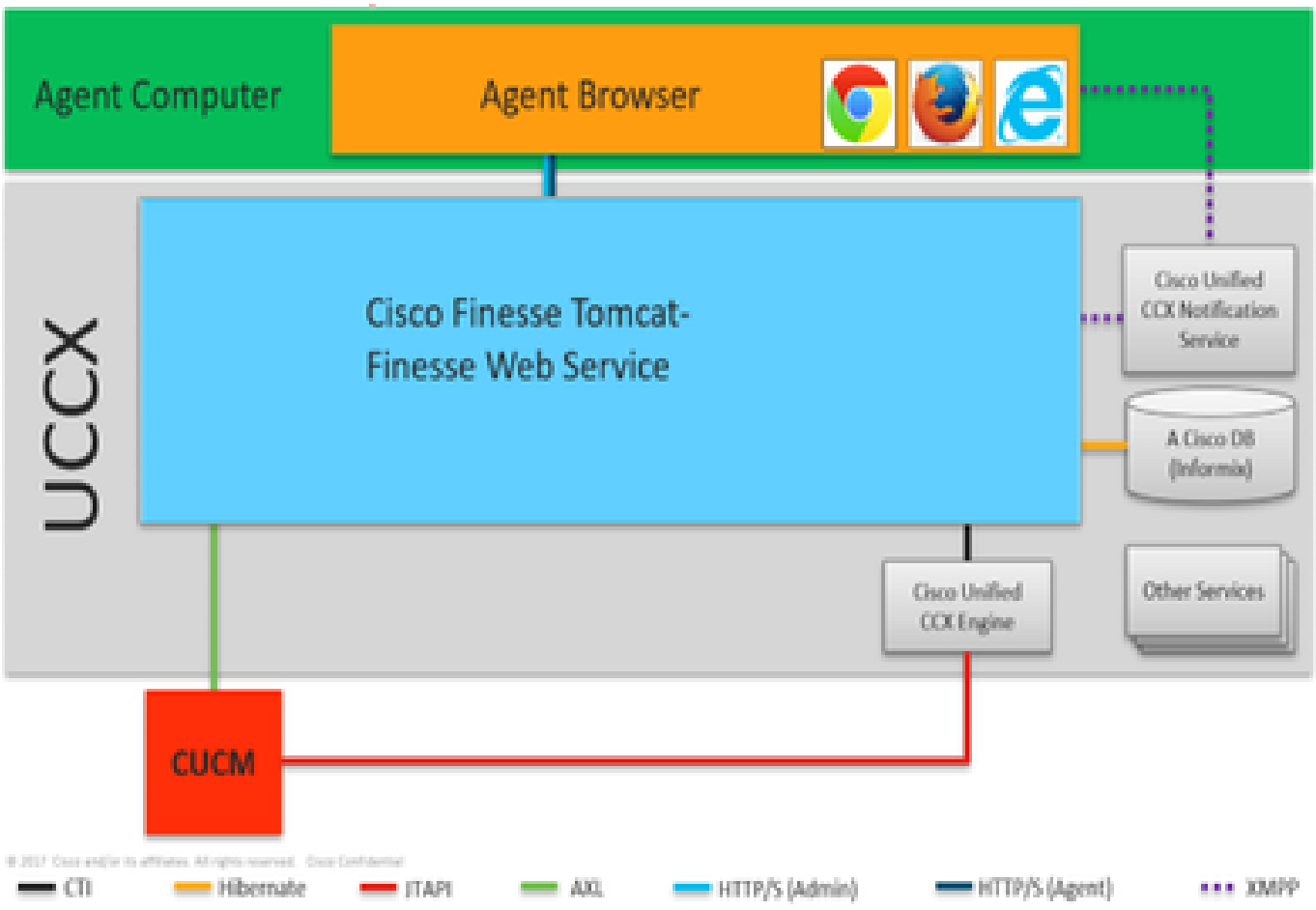
- Enveloppe - Cet élément obligatoire est la racine du message SOAP, identifiant le XML transmis comme étant un paquet SOAP. Une enveloppe contient une section de corps et une section d'en-tête facultative.

- En-tête - Cet élément facultatif fournit un mécanisme d'extension indiquant les informations de traitement du message. Par exemple, si l'opération utilisant le message nécessite des informations d'identification de sécurité, ces informations d'identification doivent faire partie de l'en-tête de l'enveloppe.

- Corps - Cet élément contient la charge utile du message, les données brutes étant transmises entre les applications émettrice et réceptrice. Le corps lui-même peut être constitué de plusieurs éléments enfants, un schéma XML définissant généralement la structure de ces données.

Vue de 20000 pieds

Le schéma suivant explique de manière plus détaillée les protocoles impliqués dans l'architecture Finesse.



20000 vue

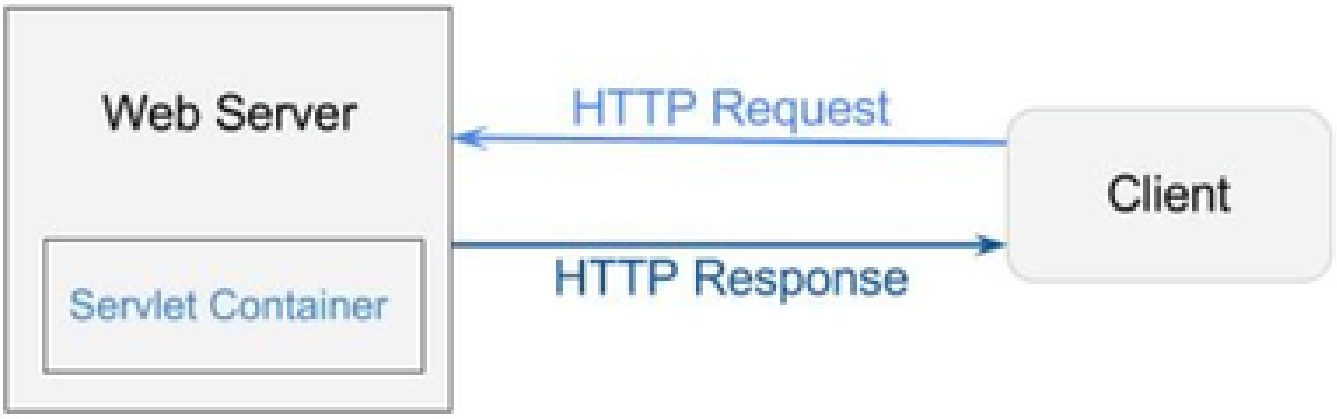
Il s'agit des protocoles responsables de la communication entre les différents composants UCCX.

- Le moteur UCCX et le serveur Finesse communiquent via le protocole CTI.

- Le moteur UCCX et CUCM communiquent via le protocole JTAPI.

- Finesse Tomcat et CUCM parlent sur AXL.
- Le service de notification Finesse et le navigateur Agent parlent sur XMPP/BOSH.
- Finesse Tomcat et la base de données parlent de Hibernate.
- Finesse Tomcat et Finesse Notification parlent de XMPP.

APACHE SHINDIG



Beuverie

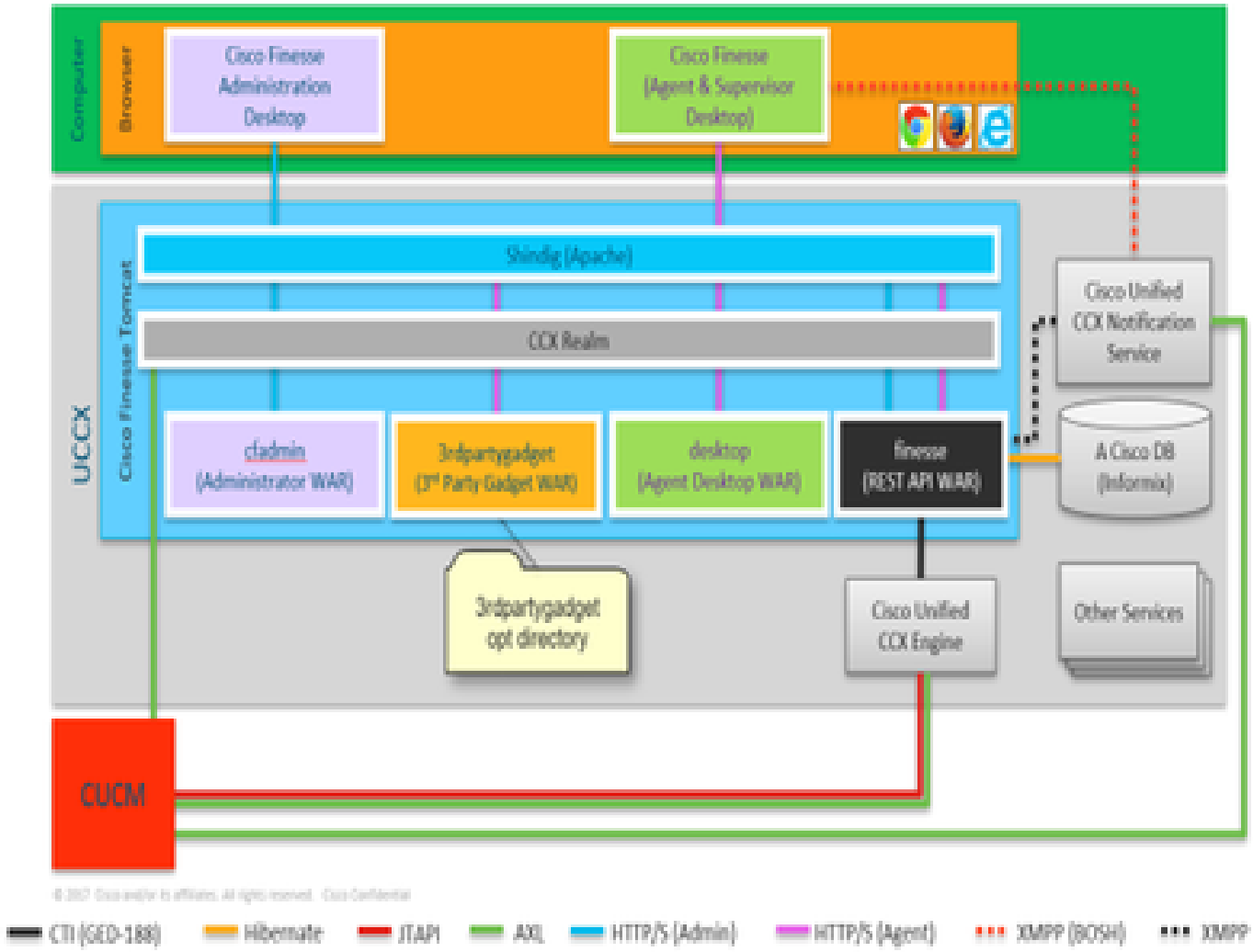
Apache Shindig est un conteneur OpenSocial et vous aide à commencer à héberger rapidement des applications OpenSocial en fournissant le code pour rendre des gadgets, des requêtes proxy et gérer les requêtes REST et RPC. OpenSocial est un ensemble d'API permettant de créer des applications sociales qui s'exécutent sur le Web. (Web/Servlet) Le conteneur est utilisé par un serveur Web pour générer dynamiquement des pages Web.

FICHIERS DE GUERRE

WAR est l'acronyme de Web Archive. Il contient les fichiers d'un projet Web. Il peut avoir servlet, XML, JSP, image, HTML, CSS, JS, etc. Les journaux Catalina contiennent des informations sur les WAR déployés.

Vue de 10000 pieds

Le schéma suivant explique en détail le fonctionnement du flux d'authentification au sein des composants d'UCCX et de Finesse.



10000 vue

Les fichiers WAR sont nécessaires pour afficher et créer la page, selon la façon dont vous vous connectez. Le navigateur demande à Shindig qu'il doit rendre un gadget, shindig parle ensuite à CUIC pour rendre le gadget. Le domaine CCX est utilisé pour l'authentification avec CUCM à l'aide d'AXL. Le service de notification s'authentifie également auprès de CUCM via AXL.

Finesse Rest API WAR est le référentiel principal qui communique réellement avec le service de notification, CCX Engine et DB. Shindig ne parle qu'avec Finesse Rest API (WebServices) parce que cfadmin et les WAR de bureau sont juste pour afficher la page. Tout ce qui vient à la Finesse Rest API WAR, vous pouvez voir que dans les journaux de Finesse WebServices qui sont les journaux les plus importants pour la finesse. Vous parlez HTTP entre Shindig et le service Web Finesse (Rest API WAR). Le service Web Finesse (Rest API WAR) et le moteur communiquent entre eux via CTI.

AJAX - La beauté de la Finesse

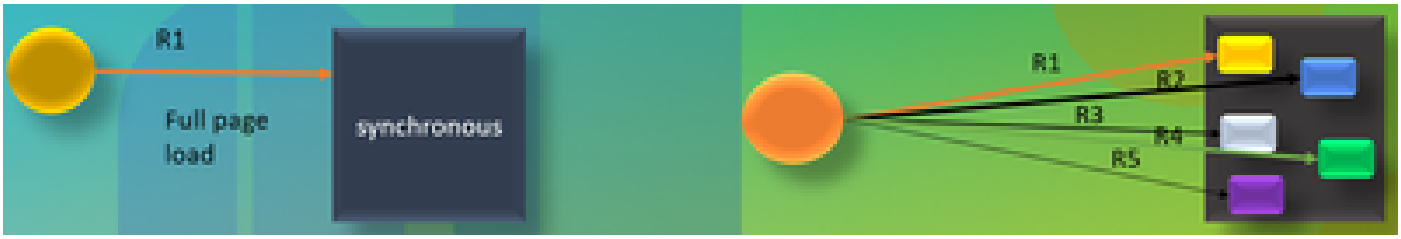
AJAX est l'acronyme de Asynchronous Javascript and XML. Il ne s'agit pas d'un langage de programmation, mais d'une méthode permettant d'accéder aux serveurs Web à partir d'une page Web. AJAX est un mécanisme de mise à jour partielle des pages. Il vous permet de mettre à jour des sections d'une page avec des données provenant d'un serveur sans avoir à actualiser la page entière.

Par exemple, si vous parlez de Facebook Messenger, quand un nouveau message arrive, vous n'avez pas besoin d'actualiser la page entière pour obtenir le message, à la place la section de message de la page elle-même actualise et obtient les nouveaux messages en temps réel sans avoir besoin d'actualiser la page entière.

Chaque navigateur comporte un objet intégré appelé XMLHTTPREQUEST (également appelé **XHR**). Chaque requête à AJAX dans le serveur passe par cette requête XML. Il contient les détails de ce que vous devez mettre à jour.

Avantages de l'utilisation d'AJAX

Le schéma suivant explique la différence entre les requêtes asynchrones et synchrones.



AJAX

Dans le cas d'une requête synchrone, vous devez attendre que la première requête soit traitée, puis vous pouvez envoyer une seconde requête. Par exemple, l'actualisation de la page est requise et vous ne pouvez rien faire tant que la page n'est pas actualisée. En revanche, dans le cas d'une requête asynchrone, vous n'avez pas besoin d'attendre que la première requête soit terminée pour envoyer la seconde requête. Vous pouvez envoyer plusieurs demandes simultanément. Par exemple, les gadgets d'applications météo sur les sites Web. Vous pouvez actualiser la section météo de la page et entre-temps également travailler sur les autres sections du site Web simultanément sans avoir besoin d'actualiser la page entière. C'est le principal avantage de la requête asynchrone.

FONCTIONNEMENT D'AJAX

AJAX est une combinaison d'un XMLHTTPREQUEST (**XHR**) qui est utilisé pour envoyer et recevoir des mises à jour du serveur Web avec Javascript et HTML qui sont utilisés pour afficher ou utiliser les données.

ENVOI DE LA REQUÊTE AVEC AJAX AU SERVEUR

Il s'agit d'un processus en 3 étapes qui est mentionné ci-dessous :

1. Création d'une variable et stockage de l'objet **XHR** dans.

Requête Var = new XMLHttpRequest();

2. Accès à la variable de demande qui contient la charge utile dans l'objet XHR.

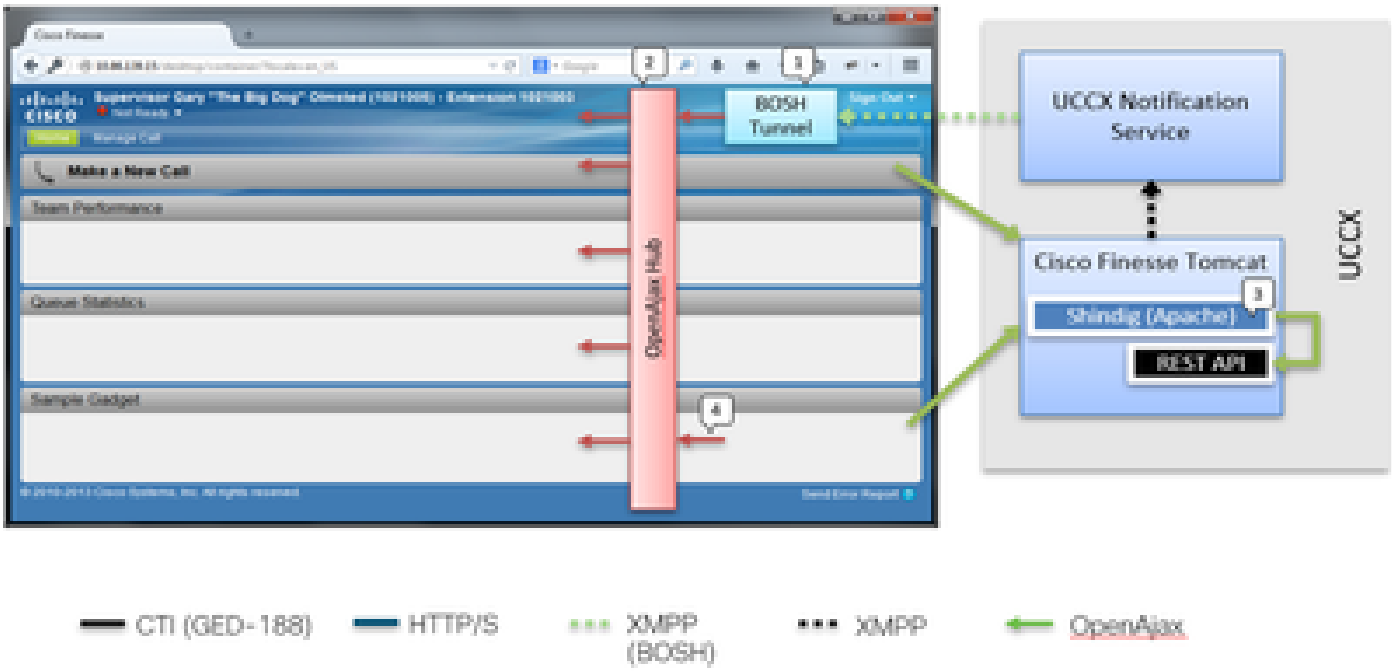
requête.**open(GET, URL)** ;

3. Envoi de la demande

Request.send() ;

Architecture de bureau

Le schéma suivant explique le flux des signaux AJAX lors du rendu du gadget sur la page Web.

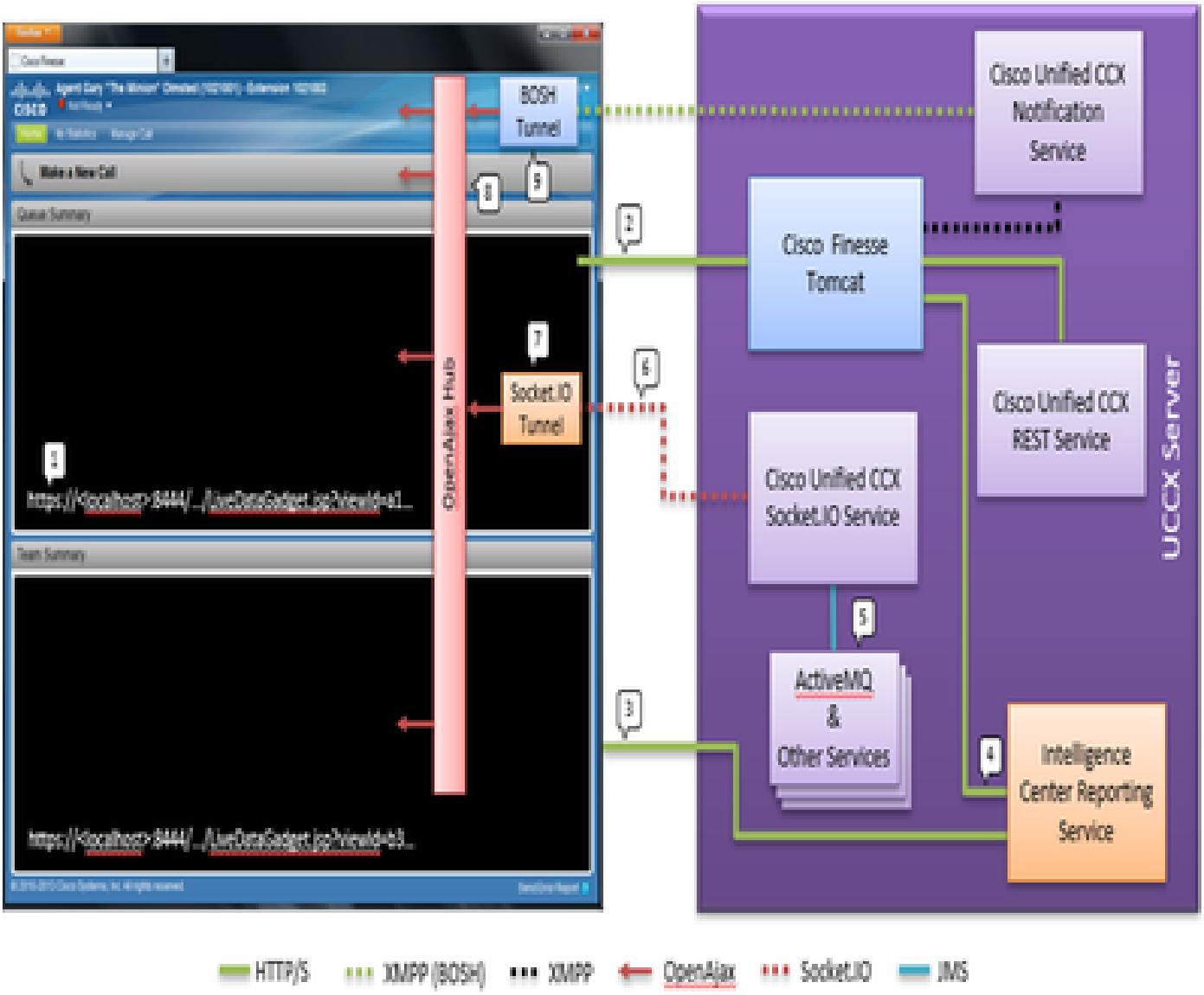


architecture de bureau

IFrame réside dans le conteneur pour héberger le tunnel BOSH. Le concentrateur OPENAJAX est fourni pour publier des messages à travers les gadgets (en utilisant la méthode pubsub). Les requêtes REST sont également transmises par proxy via Shindig à d'autres serveurs. Les gadgets peuvent publier leurs propres messages sur le concentrateur AJAX.

Architecture de gadget

Le schéma suivant explique en détail l'architecture du gadget Finesse.



architecture de gadget

Contrairement aux gadgets classiques, les gadgets CUIC reçoivent également un flux XMPP en temps réel d'OpenFire. Dans le cas d'UCCX, où CUIC et Finesse sont co-résidents avec UCCX, il existe une instance OpenFire partagée. La plupart du contenu du gadget et toutes les API REST sont mises en proxy via Shindig dans le serveur Finesse. Cela s'applique aux gadgets Finesse et à l'API REST, ainsi qu'aux instances de gadget CUIC et à l'API REST. Les gadgets CUIC utilisent un D-Grid pour le rendu de leurs rapports. Il y a un processus d'amorçage qui doit se produire et cela est fait en conjonction avec CUIC directement. Pour cette raison, les gadgets CUIC parlent d'abord directement au serveur CUIC pendant le processus de chargement. Pour cette raison, le certificat CUIC doit être accepté dans le navigateur de l'utilisateur (en plus du tunnel Socket.IO). Le contenu des gadgets et les API REST sont transmis par proxy au client entre Finesse et CUIC. Les appels de l'API restante sont passés au service Intelligence Center Reporting et au service Web CCX. Le service CCX Live Data Socket.IO reçoit les messages de Live Data via JMS depuis ActiveMQ. Le service CCX Live Data Socket.IO publie des rapports JSON en temps réel sur la connexion Socket.IO du client. De la même manière que Finesse Desktop dispose d'une iFrame de tunnel BOSH qui maintient la connexion BOSH avec le service de notification Cisco Finesse, le gadget de données en direct maître dispose d'une iFrame de tunnel Socket.IO qui maintient la connexion Socket.IO (websocket) avec le service CCX Live Data Socket.IO.

Le concentrateur OpenAjax distribue tous les événements aux auditeurs abonnés. Il s'agirait à la fois de gadgets et de parties du conteneur Finesse lui-même. Finesse Desktop dispose d'une iFrame de tunnel BOSH qui maintient la connexion BOSH avec le service de notification Cisco Unified CCX. Les événements sont alors publiés sur le Hub OpenAjax.

Liens de référence

- Guide du développeur de services Web Finesse

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.