Informazioni approfondite sull'architettura UCCX Finesse

Sommario

Introduzione

Questo documento descrive l'architettura Finesse in modo approfondito in modo che i processi sottostanti abbiano senso durante la risoluzione dei problemi di finesse.

Prerequisiti

Requisiti

Cisco raccomanda la conoscenza dei seguenti strumenti e funzionalità:

JTAPI - API di telefonia Java

API - Interfaccia di programmazione delle applicazioni

UCCX - Unified Contact Center Express

CUCM - Cisco Unified Communications Manager

CTI - Integrazione con la telefonia informatica

Componenti usati

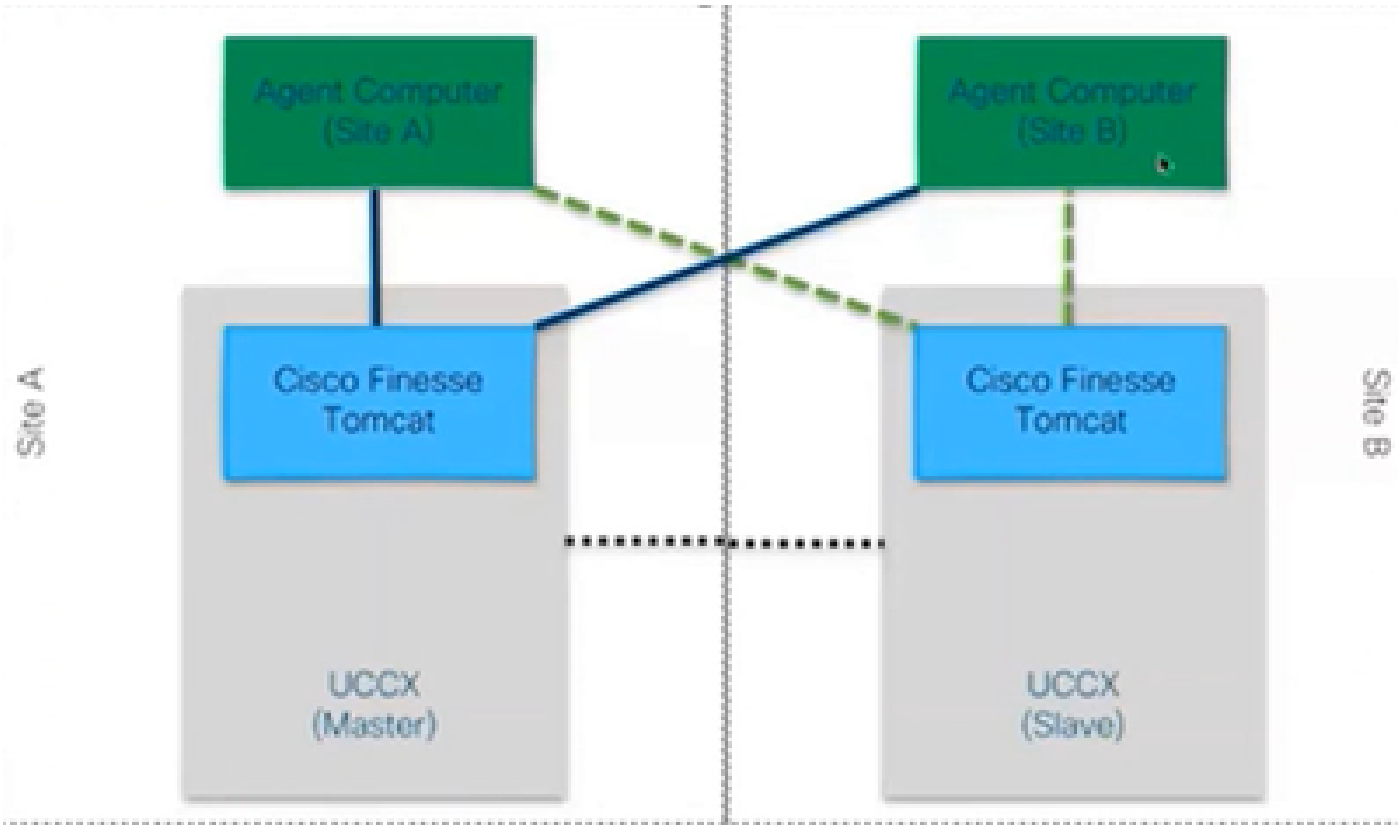
- Cisco Unified Contact Center Express (UCCX)

Le informazioni discusse in questo documento fanno riferimento a dispositivi usati in uno specifico ambiente di emulazione. Su tutti i dispositivi menzionati nel documento la configurazione è stata ripristinata ai valori predefiniti. Se la rete è operativa, valutare attentamente eventuali conseguenze derivanti dall'uso dei comandi.

Premesse

Questo documento descrive l'architettura Finesse a partire da una panoramica di alto livello e quindi il flusso del segnale in profondità con esempi e diagrammi.

50.000 piedi di vista



visualizzazione 50000

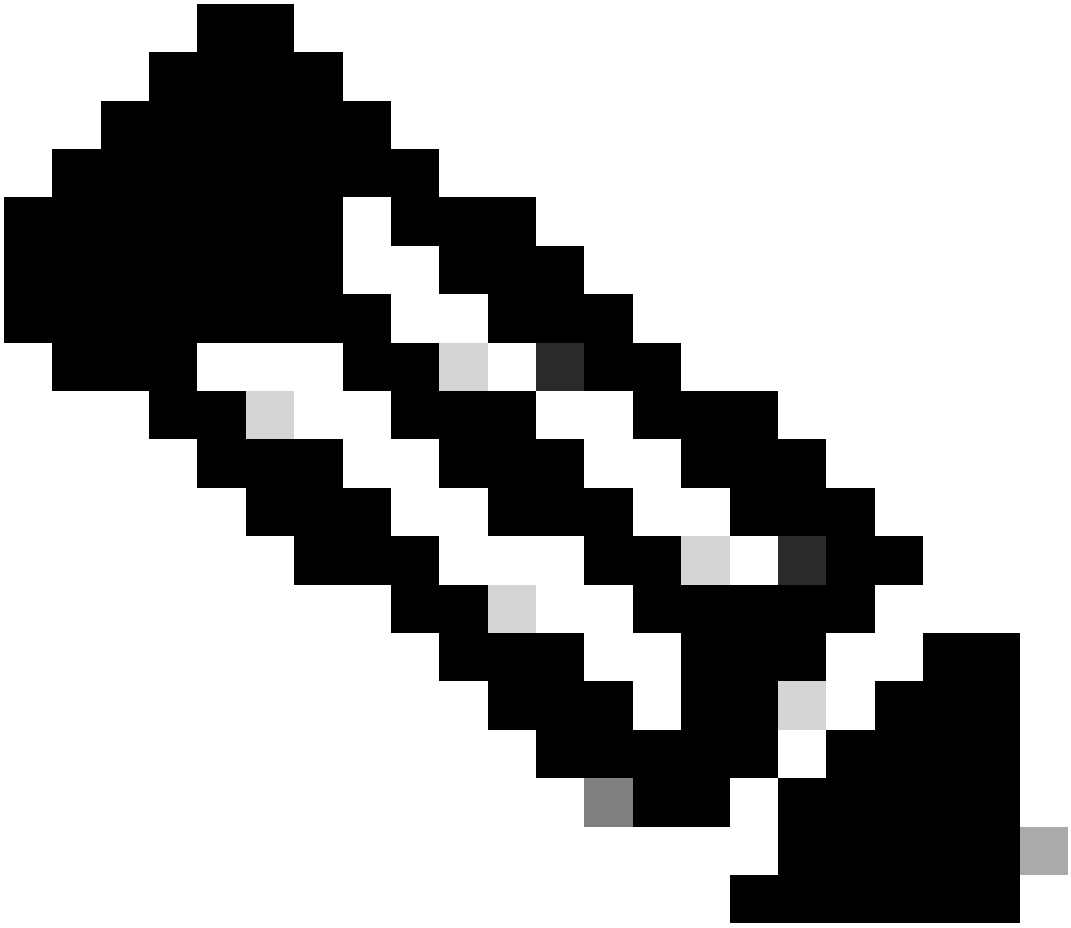
Finesse Tomcat

Finesse Tomcat è simile a Cisco Tomcat in CUCM in quanto la funzionalità è la stessa per caricare le pagine Web ma per un servizio diverso chiamato Finesse. Tomcat è stato progettato per Finesse perché è un'applicazione Web separata. È possibile accedere a finesse solo utilizzando il nodo master CCX come nelle versioni precedenti alla 11.5. Dalla versione 11.6 in poi, è possibile accedere a qualsiasi nodo (ma non consigliato) solo durante il failover. Pertanto, se si considera un cluster WAN, in cui entrambi gli agenti (sul sito A e sul sito B) sono connessi a Finesse sul nodo master, esiste sempre una connessione inattiva all'altro nodo da Cisco Finesse al motore, ovvero una richiesta CTI openconf necessaria per il failover.

La richiesta Openconf viene inviata regolarmente per verificare se il nodo è attivo o in standby. In caso di failover, il servizio di notifica utilizza un'API denominata systeminfo API che indica al client che il nodo è inattivo e che è necessario un failover. Viene quindi eseguito un file che reindirizza il browser all'altro nodo e quindi viene inviato openconf rejavascriptsponse. Questo failover funziona solo se i certificati vengono accettati. (per stabilire una connessione protetta al server).

Sono presentati 3 certificati:

- Certificato del servizio di notifica per il nodo.
- Certificato del servizio di notifica per il nodo remoto.
- Certificato del servizio Finesse per il nodo remoto.



Nota: affinché il failover funzioni, è necessario accettare i certificati del nodo remoto.

HTTP/I

Si tratta di un protocollo a livello di applicazione per l'invio di documenti HyperMedia, ad esempio HTML. È progettato per la comunicazione tra browser Web e server Web.È un protocollo senza stato che significa che il server non mantiene alcun dato tra le due richieste. Si tratta di un protocollo server client. Per UCCX, il client Finesse viene eseguito sul browser agente (PC). Effettua una richiesta al server utilizzando HTTP. Di seguito sono riportati alcuni metodi HTTP comuni:

GET - per ottenere informazioni da un server.
POST - per inviare informazioni a un server.

PUT - per sostituire qualsiasi elemento su un server.
DELETE - per rimuovere le informazioni da un server.

Finesse utilizza le richieste api systeminfo all'interno della richiesta http. Ad esempio, se si desidera modificare lo stato di un agente, il browser invia un PUT anziché un POST perché, se il POST viene inviato, il server si confonde in quanto ha due stati in mano, quale selezionare? Quindi utilizzando PUT, sostituisce lo stato corrente.

XMPP

Protocollo Extensible Messaging and Presence

Si tratta di un insieme di protocolli utilizzati per la messaggistica istantanea e la presenza. Tutte le entità XMPP vengono identificate utilizzando i relativi ID Jabber o JID. Una delle estensioni del protocollo XMPP utilizzato per i gadget è denominata PUBSUB.

PUBSUB

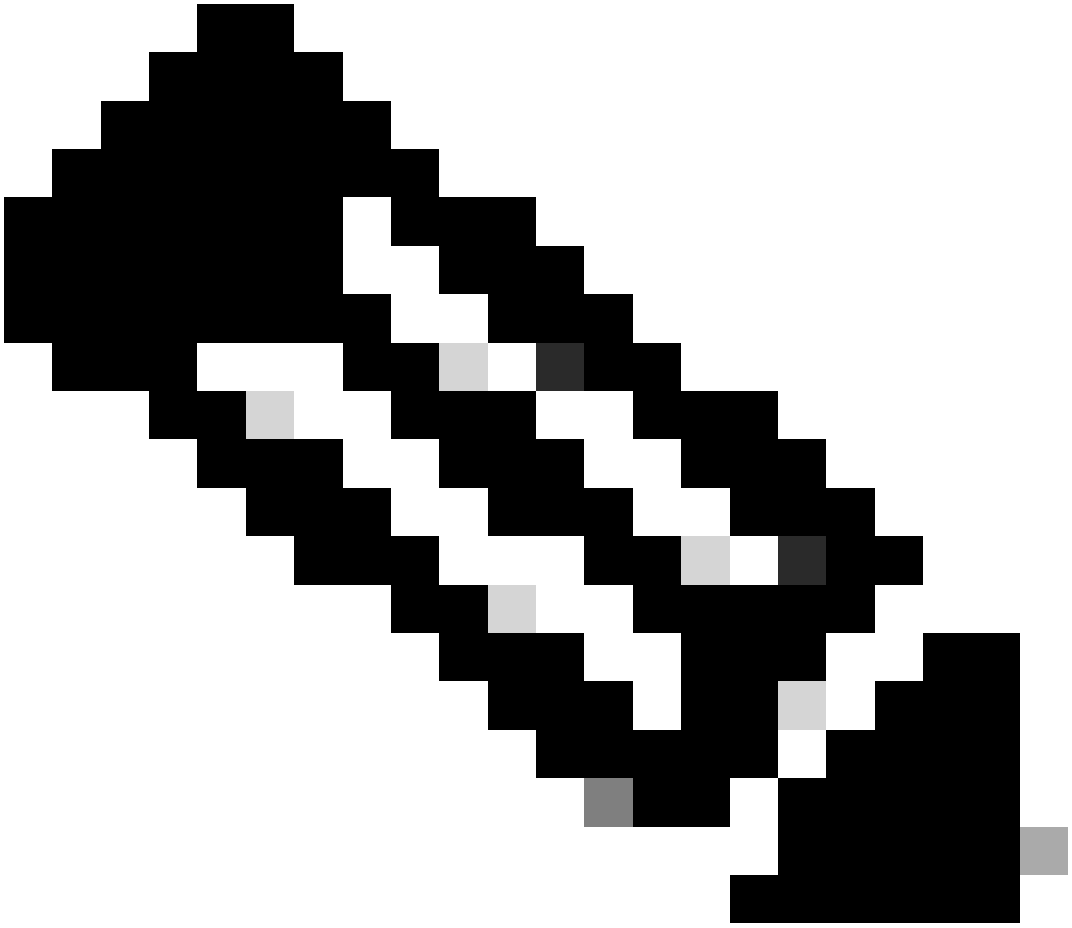
Non è il sottoscrittore del publisher a cui si può pensare. Continua a pubblicare e sottoscrivere, ma non ha nulla a che fare con i database. XMPP utilizza un meccanismo denominato nodi. Il nodo sta fondamentalmente monitorando l'entità a cui tieni. Qualsiasi cosa sia importante per voi e desideri monitorarlo, aggiungete un nodo a esso. PUBSUB esegue quindi la sottoscrizione per gli aggiornamenti o le notifiche su tale nodo. Potete avere nodi per ogni tipo di entità, ad esempio agente, finestra di dialogo e così via. Se si crea un nodo per un agente, si viene sottoscritti al nodo su quell'agente e quindi qualsiasi operazione eseguita dall'agente viene notificata.

Lo scopo di questa specifica è consentire al server XMPP (Servizio di notifica) di ottenere informazioni pubblicate sui nodi XMPP (argomenti) e quindi di inviare eventi XMPP alle entità sottoscritte a tale nodo.

Nel caso di Finesse, il server CTI (Computer Telephony Integration) invia messaggi CTI al servizio Web Finesse per comunicare a Finesse gli aggiornamenti della configurazione quali, ma non solo, la creazione di un agente o la creazione di una coda CSQ (Contact Service Queue) o le informazioni su una chiamata. Queste informazioni vengono quindi convertite in un messaggio XMPP che il servizio Web Finesse pubblica al servizio Finesse Notification. Il servizio Finesse Notification invia quindi messaggi XMPP over BOSH agli agenti che hanno sottoscritto determinati nodi XMPP.

BOSH - Flussi bidirezionali su HTTP sincrono

BOSH è una connessione HTTP di lunga durata in cui il server conserva la richiesta per un periodo di tempo maggiore fino a quando non ha una risposta, altrimenti invia una risposta vuota. Questa opzione funziona per i client XMPP e i server XMPP, ma può essere utilizzata anche per le applicazioni non XMPP.



Nota: il protocollo XMPP è con conservazione dello stato, mentre il protocollo HTTP è senza conservazione dello stato (non memorizza le informazioni sull'ultima richiesta)

Se un'applicazione Web deve funzionare con XMPP, possono verificarsi numerosi problemi.

Problema 1: i browser non supportano XMPP su TCP (Transmission Control Protocol) in modo nativo.

Soluzione 1: i server Web e i browser comunicano tramite messaggi HTTP (HyperText Transfer Protocol), pertanto Finesse e altre applicazioni Web inseriscono i messaggi XMPP all'interno dei messaggi HTTP.

Problema 2: HTTP è un protocollo senza stato.

Soluzione 2: per questa operazione è possibile utilizzare cookie/inserire dati.

Problema 3: il terzo problema è il comportamento unidirezionale di HTTP, che significa che solo il client invia le richieste e il server può solo rispondere. L'impossibilità del server di eseguire il push dei dati rende innaturale l'implementazione di XMPP su HTTP.

Soluzione 3: per risolvere questo problema, è necessario disporre di un collegamento tra HTTP e XMPP.

Le soluzioni proposte sono:

- Polling (protocollo legacy): richieste HTTP ripetute che richiedono nuovi dati definiti: polling HTTP

- Il polling lungo è anche noto come BOSH: protocollo di trasporto che emula la semantica di una connessione TCP bidirezionale di lunga durata tra due entità utilizzando in modo efficiente più coppie di richiesta/risposta HTTP sincrone senza richiedere l'uso di polling frequenti. Il motivo per utilizzare BOSH è quello di coprire il fatto che il server non deve rispondere non appena c'è una richiesta. La risposta viene posticipata fino a un tempo specificato finché il server non dispone dei dati per il client, quindi viene inviata come risposta. Non appena il client riceve la risposta, effettua una nuova richiesta e così via.

Il client desktop Finesse (applicazione Web) stabilisce una connessione BOSH non aggiornata sulla porta TCP 7443 ogni 30 secondi. Dopo 30 secondi, se non sono disponibili aggiornamenti dal servizio di notifica Finesse, il servizio di notifica invia una risposta HTTP con 200 OK e un corpo di risposta (quasi) vuoto. Se il Servizio di notifica dispone di un aggiornamento sulla presenza di un agente o di un evento di conversazione (chiamata), ad esempio, i dati vengono inviati immediatamente al client Web Finesse.

Per riepilogare:

Il client Web Finesse dispone di una connessione HTTP non aggiornata (http-bind) impostata sul server Finesse tramite la porta TCP 7443. Questo è noto come un sondaggio di BOSH. Finesse Notification Service è un servizio di presenza che pubblica aggiornamenti relativi allo stato di un agente, di una chiamata e così via. Se il servizio di notifica dispone di un aggiornamento, risponde alla richiesta http-bind con l'aggiornamento dello stato come messaggio XMPP nel corpo della risposta HTTP. Se non sono disponibili aggiornamenti dello stato 30 secondi dopo la ricezione della richiesta http-bind, il servizio di notifica risponde senza aggiornamenti dello stato per consentire al client Web Finesse di inviare un'altra richiesta http-bind. In questo modo il servizio di notifica può sapere che il client Web Finesse è ancora in grado di connettersi al servizio di notifica e che l'agente non ha chiuso il browser o messo il computer in sospensione e così via.

CTI

È possibile utilizzare l'integrazione Telefonia computer (CTI, Computer Telephony Integration) per sfruttare le funzioni di elaborazione del computer durante le chiamate telefoniche, la ricezione e la gestione. Le applicazioni CTI consentono di eseguire attività quali il recupero di informazioni utente da un database utilizzando un ID chiamante o di utilizzare le informazioni raccolte da un sistema IVR (Interactive Voice Response) per indirizzare una chiamata proveniente dall'utente insieme alle relative informazioni al rappresentante del servizio appropriato. CTI Manager su CUCM risponde alle richieste JTAPI di UCCX. La porta TCP del server CTI è 12018. In questo modo Finesse Server e Engine (CTI Server) comunicano tra loro.

Di seguito alcune delle INFORMAZIONI SCAMBIATE TRAMITE CTI:

- Configurazione corrente del sistema e aggiornamenti futuri.

- Agenti e loro stati.

- Le chiamate e i loro stati.

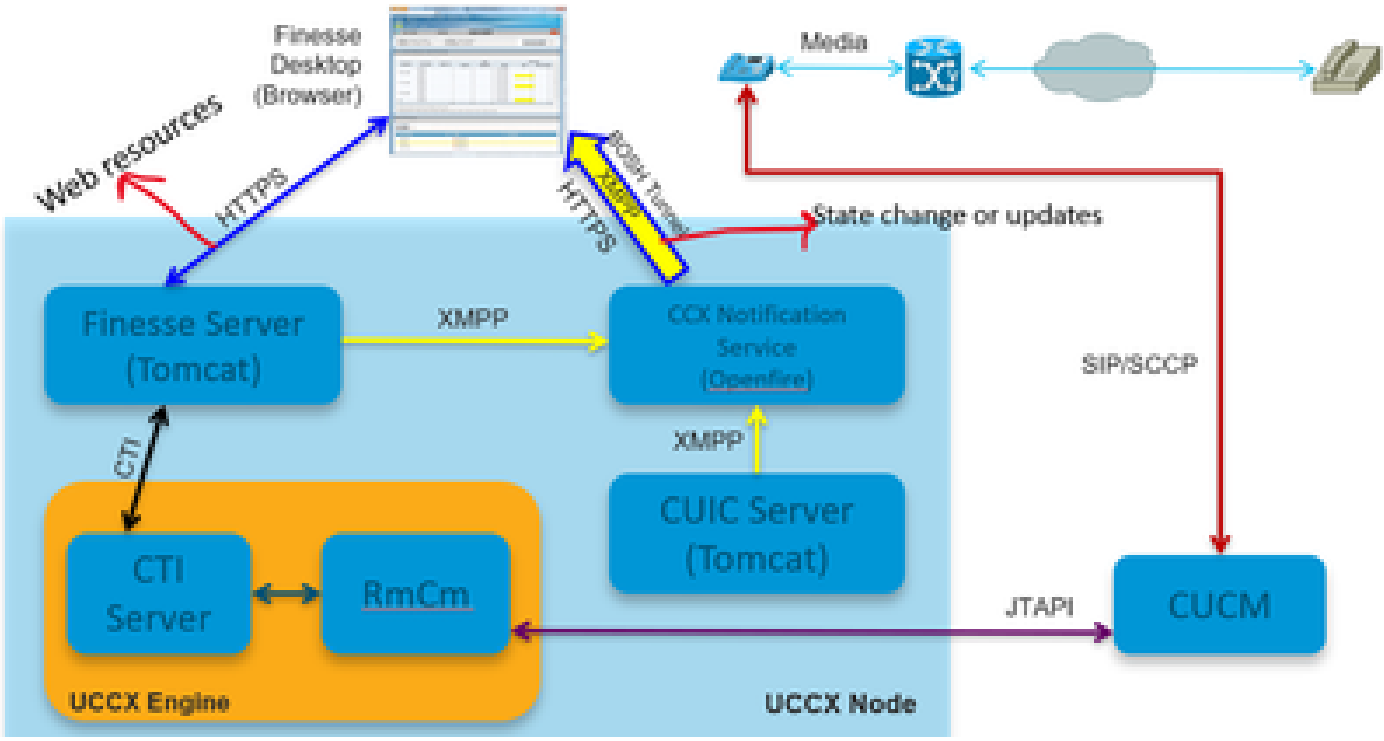
- Statistiche per agenti, chiamate e code in tempo reale.

JTAPI

Cisco Unified JTAPI è uno standard di interfaccia di programmazione sviluppato da Sun Microsystems per l'utilizzo con applicazioni di telefonia informatica basate su Java. Cisco JTAPI implementa la specifica Sun JTAPI 1.2 con estensioni Cisco aggiuntive. Qualsiasi comunicazione tra UCCX e CUCM risiede su JTAPI. Questo è il modo in cui CUCM e Engine (sottosistema Telefonia) comunicano tra loro. JTAPI viene utilizzato per controllare e monitorare i telefoni CUCM, instradare le chiamate tramite le porte CTI e i punti di instradamento, avviare e interrompere le registrazioni su CUCM e per qualsiasi funzionalità di instradamento delle chiamate

30.000 piedi di vista

Il diagramma seguente descrive come UCCX Engine, Finesse, CUCM e Browser comunicano tra loro.

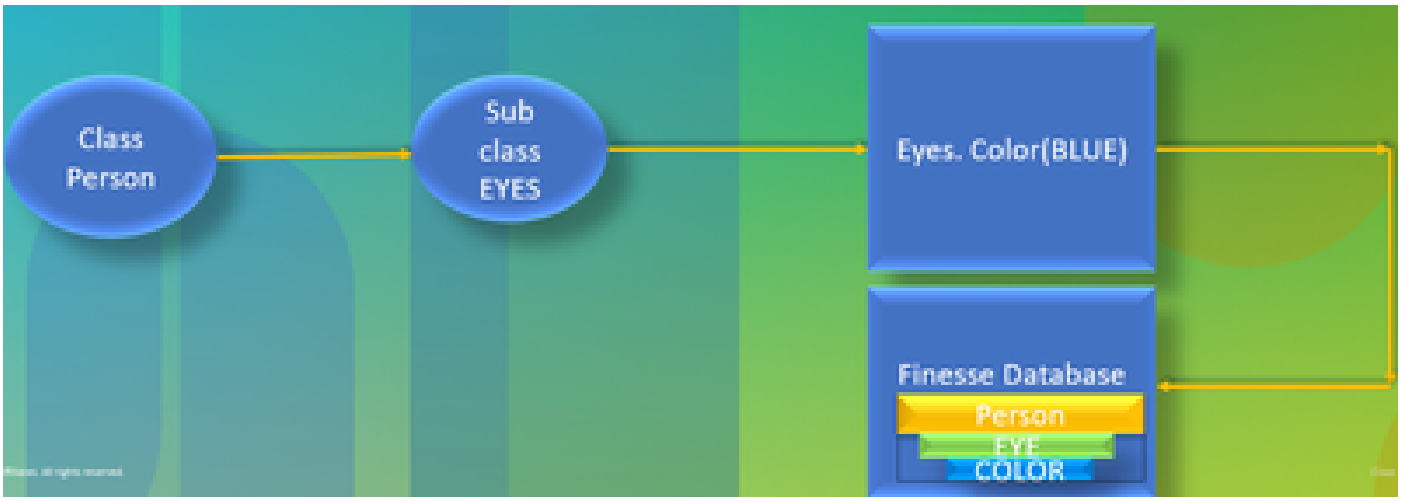


visualizzazione 30000

Consideriamo che la chiamata è stabilita con l'agente. Ora, RmCm che sta monitorando l'estensione dell'agente tramite JTAPI, comunica al server CTI la modifica dello stato comunicata dall'agente. Queste informazioni vengono inviate dal server CTI (all'interno del motore CCX) a Finesse Server (Tomcat) utilizzando CTI. Il server Finesse invia queste informazioni al servizio di notifica CCX utilizzando XMPP per la modifica dello stato. Il servizio di notifica (Openfire) apre un tunnel BOSH al browser dell'agente e aggiorna le informazioni sulla modifica dello stato. In questo modo l'agente passa allo stato RISERVATO. Qualsiasi tipo di risorse Web viene richiesto al server finesse utilizzando HTTPS come file WAR, gadget e così via (se non già nella cache).

IBERNAZIONE

Nel diagramma seguente viene illustrato il servizio di sospensione.



ibernare

HIBERNATE è noto come High-Performance Object/Relational Persistence and Query Service. In altre parole, mappa le classi JAVA alle tabelle di database. Si supponga, ad esempio, di disporre di un oggetto JAVA denominato Team e di una tabella di database nel database finesse denominata Team. La classe JAVA controlla quali informazioni sono contenute nella tabella e HIBERNATE è la causa di questo comportamento. Anziché utilizzare query SQL, utilizza classi Java per aggiornare le informazioni.

AXL

XML amministrativo.

XML è l'acronimo di eXtensible Markup Language ed è un linguaggio di markup che definisce alcune regole relativamente semplici per la codifica dei dati. È stato progettato principalmente per trasmettere e ricevere dati strutturati in un formato ben definito che entrambi i sistemi sono in grado di comprendere. Nel formato più semplice, XML definisce i tag racchiusi tra parentesi angolari (<>) che racchiudono i dati descritti dal tag. I tag possono formare una gerarchia con tag all'interno di altri tag. Ad esempio, per definire un dispositivo telefonico di base, è possibile dire che un dispositivo telefonico richiede tre parametri, un nome, una descrizione e un numero di telefono.

Si tratta di un'API basata su SOAP che consente il provisioning remoto su CUCM. Viene utilizzato per aggiungere, aggiornare, rimuovere o recuperare informazioni dal database CUCM. Le funzionalità di recupero includono il controllo dell'autenticazione utente e l'esecuzione di query SQL. L'API AXL consente di accedere all'intero database CUCM. L'API AXL viene utilizzata esclusivamente per il provisioning e non consente l'accesso ai dati di runtime o delle prestazioni.

L'API AXL utilizza l'autenticazione di base HTTPS. Qualsiasi utente CUCM (Application o EndUser) dispone di accesso in lettura/scrittura tramite AXL se è membro del gruppo di controllo di accesso **Utenti privilegiati CCM standard** o di qualsiasi gruppo a cui è assegnato il ruolo di **accesso API AXL standard**. Ciò significa che tutti gli account utente con privilegi avanzati hanno implicitamente già accesso all'API AXL. Per creare un account dedicato all'uso dell'API AXL, è innanzitutto necessario creare un gruppo di controllo dell'accesso e assegnarvi il ruolo di **accesso** all'**API AXL standard**, quindi associare l'utente dell'applicazione al gruppo appena creato. Per consentire l'accesso all'API AXL di sola lettura, è possibile creare un gruppo di controllo di accesso separato e assegnarvi solo il ruolo di **accesso all'API di sola lettura AXL standard**.

SAPONE

Il protocollo SOAP (Simple Object Access Protocol) consente di passare informazioni tra applicazioni in formato XML. I messaggi SOAP vengono trasmessi dall'applicazione di invio all'applicazione ricevente, in genere su una sessione HTTP. Il messaggio SOAP effettivo è costituito dall'elemento Envelope, che contiene un elemento Body e un elemento Header facoltativo.

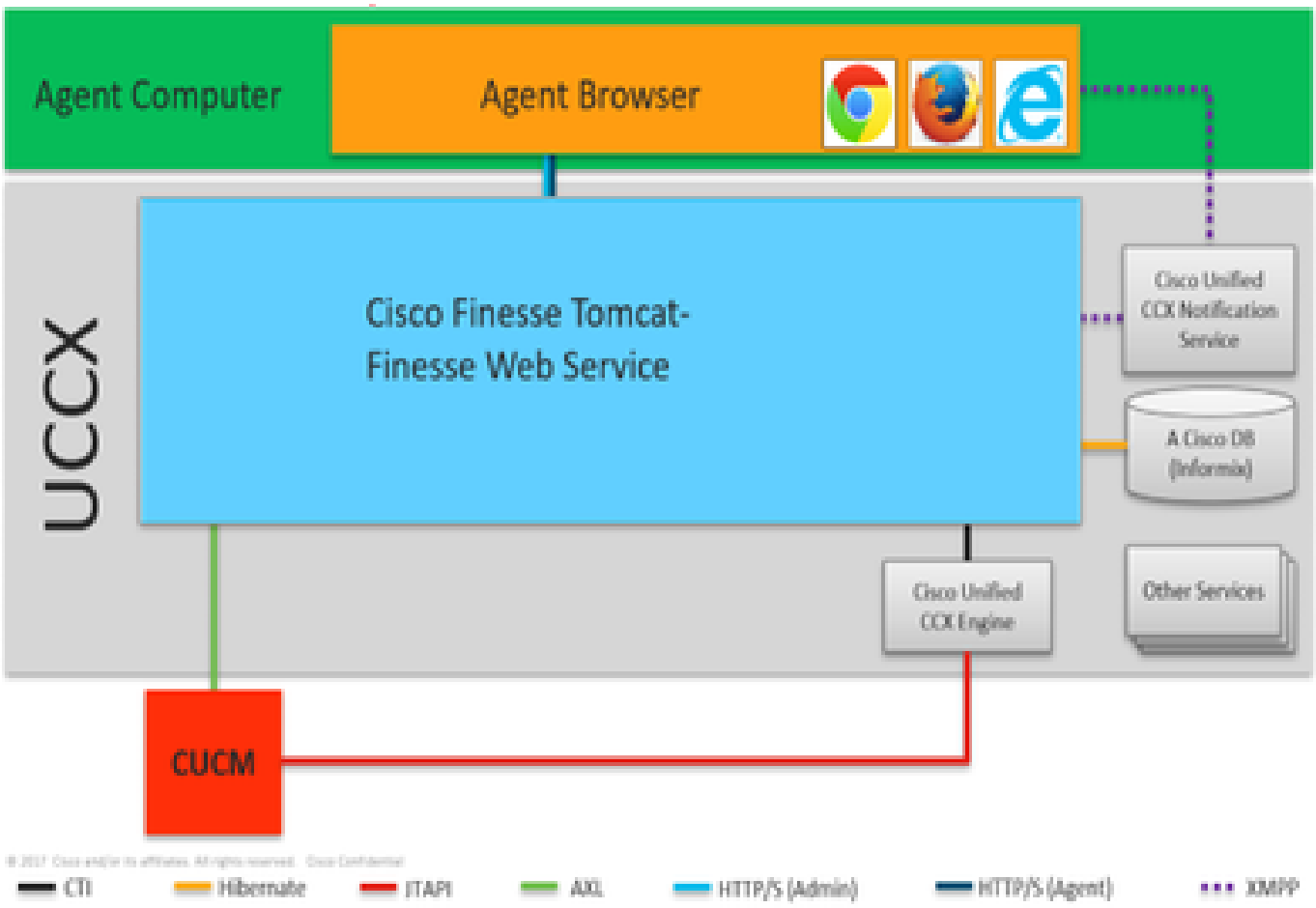
- Busta**:** questo elemento obbligatorio è la radice del messaggio SOAP, che identifica il file XML trasmesso come pacchetto SOAP. Un involucro contiene una sezione corpo e una sezione intestazione facoltativa.

- Intestazione - Questo elemento facoltativo fornisce un meccanismo di estensione che indica le informazioni di elaborazione per il messaggio. Se ad esempio l'operazione che utilizza il messaggio richiede credenziali di protezione, tali credenziali devono far parte dell'intestazione della busta.

- Body - Questo elemento contiene il payload del messaggio, i dati non elaborati trasmessi tra le applicazioni di invio e di ricezione. Il corpo stesso può essere costituito da più elementi figlio, con uno schema XML che in genere definisce la struttura di questi dati.

20.000 piedi di vista

Il diagramma seguente illustra in modo più dettagliato i protocolli coinvolti nell'architettura Finesse.



visualizzazione 20000

Questi sono i protocolli responsabili della comunicazione tra i diversi componenti UCCX.

- UCCX Engine e Finesse Server parlano sul protocollo CTI.

- UCCX Engine e CUCM parlano sul protocollo JTAPI.

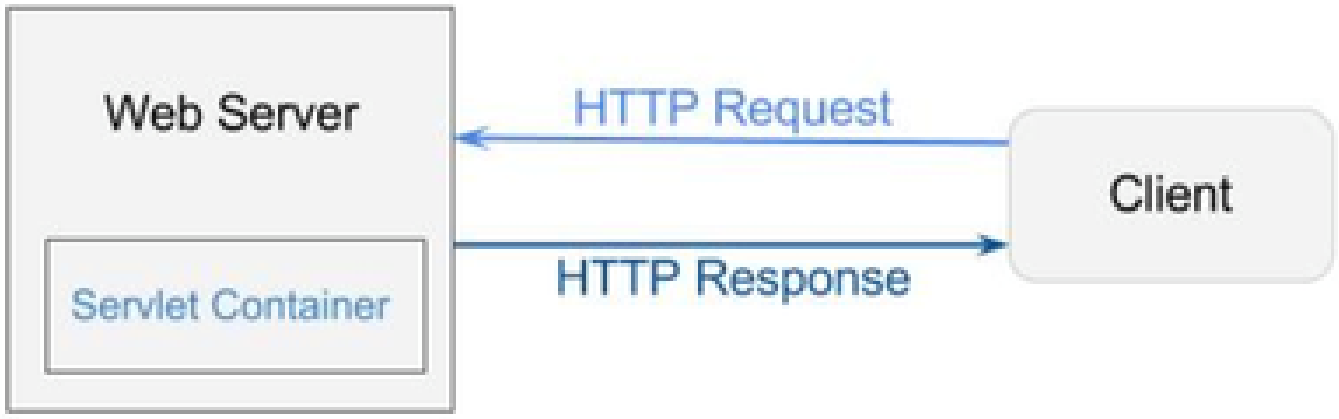
- Finesse Tomcat e CUCM parlano su AXL.

- Presentazione del servizio di notifica Finesse e del browser dell'agente su XMPP/BOSH.

- Finesse Tomcat e il database parlano dell'ibernazione.

- Finesse Tomcat e Finesse Notification parlano su XMPP.

APACHE SHINDIG



Shindig

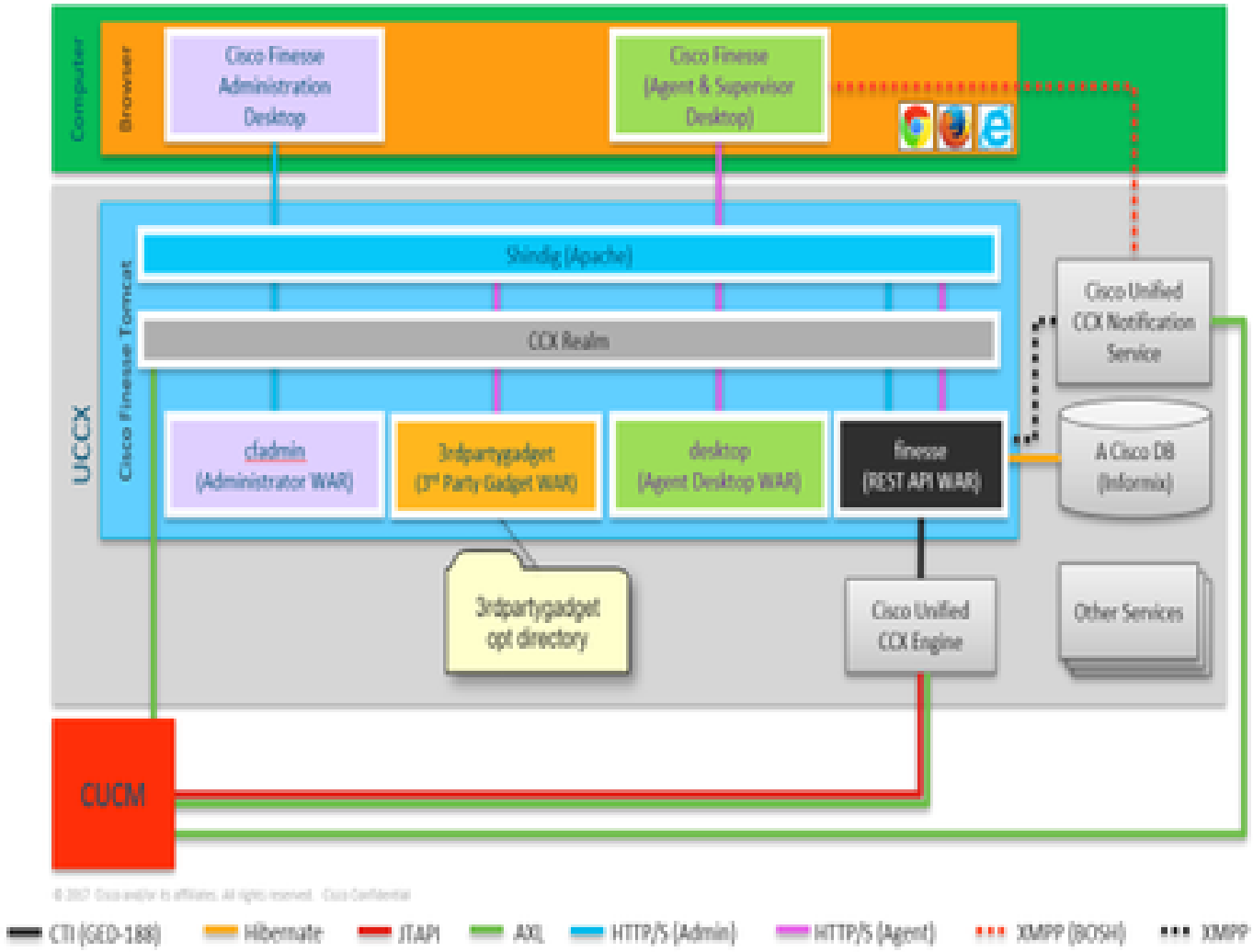
Apache Shindig è un contenitore OpenSocial e consente di avviare rapidamente l'hosting delle app OpenSocial fornendo il codice per il rendering di gadget, richieste proxy e la gestione delle richieste REST e RPC. OpenSocial è un set di API per la creazione di applicazioni di social networking che vengono eseguite sul Web. (Web/Servlet) Il contenitore viene utilizzato da un server Web per generare dinamicamente le pagine Web.

FILE WAR

WAR è l'acronimo di Web Archive. Contiene i file di un progetto Web. Può includere servlet, XML, JSP, image, HTML, CSS, JS e così via. I log Catalina contengono le informazioni relative ai file WAR che vengono distribuiti.

10.000 piedi di vista

Nel diagramma successivo viene illustrato in dettaglio il funzionamento del flusso di autenticazione all'interno dei componenti di UCCX e Finesse.



visualizzazione 10000

I file WAR sono necessari per visualizzare e creare la pagina a seconda della modalità di accesso. Browser chiede a Shindig che deve eseguire il rendering di un gadget, shindig quindi parla con CUIC per eseguire il rendering del gadget. CCX Realm viene utilizzato per l'autenticazione con CUCM tramite AXL. Il servizio di notifica esegue inoltre l'autenticazione con CUCM utilizzando AXL.

Finesse Rest API WAR è il repository principale che comunica effettivamente con il servizio di notifica, CCX Engine e DB. Shindig parla solo con l'API Finesse Rest (WebServices) perché cfadmin e desktop WAR sono solo per visualizzare la pagina. Tutto ciò che arriva a Finesse Rest API WAR, si può vedere che nei log di Finesse WebServices che sono i log più importanti per finesse. Si parla di HTTP tra Shindig e il servizio Web Finesse (Rest API WAR). Il servizio Web Finesse (Rest API WAR) e il motore parlano tra loro tramite CTI.

AJAX - La bellezza di Finesse

AJAX è l'acronimo di Asynchronous Javascript and XML. Non è un linguaggio di programmazione, ma un metodo per accedere ai server Web da una pagina Web. AJAX è un meccanismo che consente di eseguire aggiornamenti parziali delle pagine. Consente di aggiornare sezioni di una pagina con dati provenienti da un server senza dover aggiornare l'intera pagina.

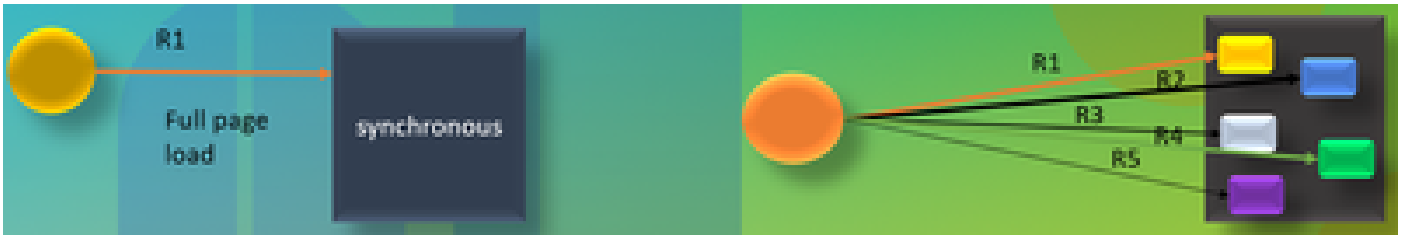
Ad esempio, se si parla di Facebook Messenger, quando arriva un nuovo messaggio non è necessario aggiornare l'intera pagina per ricevere il messaggio, ma la sezione dei messaggi della pagina stessa viene aggiornata e i nuovi messaggi vengono ricevuti in tempo reale senza dover aggiornare l'intera pagina.

Ogni browser dispone di un oggetto incorporato denominato XMLHTTPREQUEST (denominato anche **XHR**). Ogni richiesta ad AJAX nel

server passa attraverso questa richiesta XML. Contiene le specifiche degli elementi da aggiornare.

Vantaggi dell'utilizzo di AJAX

Nella figura seguente viene illustrata la differenza tra richieste asincrone e sincrone.



AJAX

Nel caso di una richiesta sincrona, è necessario attendere che la prima richiesta venga elaborata e quindi inviare la seconda richiesta. Ad esempio, è necessario aggiornare la pagina e non è possibile eseguire alcuna operazione finché la pagina non viene aggiornata. D'altra parte, in caso di richiesta asincrona, non è necessario attendere il completamento della prima richiesta per inviare la seconda. È possibile inviare più richieste contemporaneamente. Ad esempio, i gadget delle app meteo sui siti Web. È possibile aggiornare la sezione meteo della pagina e nel frattempo lavorare anche sulle altre sezioni del sito web simultaneamente senza dover aggiornare l'intera pagina. Questo è il vantaggio principale della richiesta asincrona.

FUNZIONAMENTO DI AJAX

AJAX è una combinazione di XMLHTTPREQUEST (**XHR**) utilizzata per inviare e ricevere aggiornamenti dal server Web insieme a Javascript e HTML utilizzati per visualizzare o utilizzare i dati.

INVIO RICHIESTA CON AJAX AL SERVER

Si tratta di un processo in tre fasi che viene descritto di seguito:

1. Creazione di una variabile e memorizzazione dell'oggetto **XHR** in tale variabile.

Richiesta Var = new XMLHttpRequest();

2. Accesso alla variabile di richiesta con payload all'interno dell'oggetto XHR.

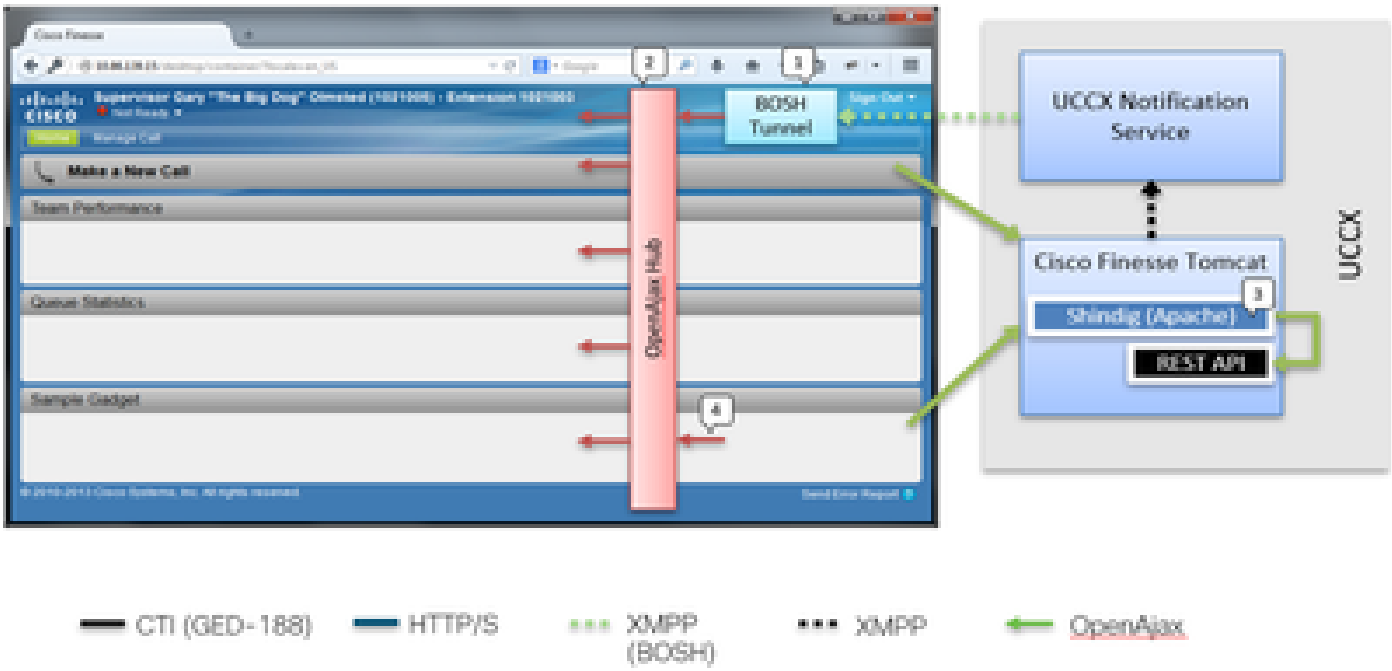
richiesta.**open(GET, URL)** ;

3. Invio della richiesta

Request.send() ;

Architettura desktop

Nel diagramma seguente viene illustrato il flusso dei segnali AJAX durante il rendering dei gadget nella pagina Web.

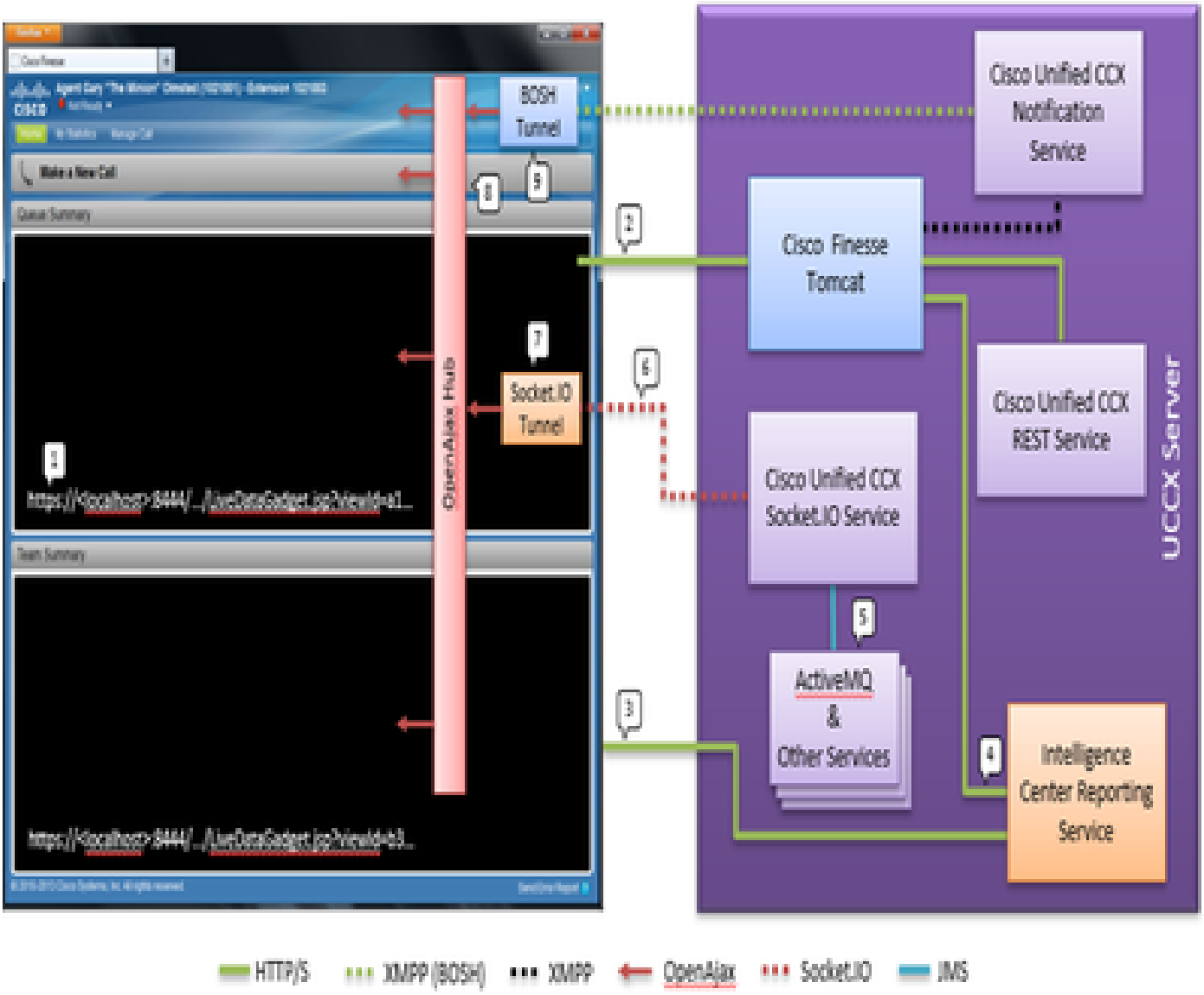


architettura desktop

IFrame risiede nel contenitore per ospitare il tunnel BOSH. L'hub OPENAJAX viene fornito per pubblicare messaggi tra i gadget (utilizzando il metodo pubsub). Le richieste REST vengono inoltrate tramite Shinding anche ad altri server. I gadget possono pubblicare i propri messaggi sull'hub AJAX.

Architettura gadget

Nel diagramma seguente viene illustrata in dettaglio l'architettura dei gadget di Finesse.



architettura gadget

A differenza dei gadget tipici, i gadget CUIC ricevono anche un feed XMPP in tempo reale da OpenFire. Nel caso di UCCX, in cui CUIC e Finesse risiedono insieme a UCCX, è presente un'istanza condivisa di OpenFire. La maggior parte del contenuto dei gadget e tutte le API REST vengono inoltrate tramite Shinding in Finesse Server. Questo vale per Gadget Finesse e API REST, nonché per istanze di Gadget CUIC e API REST. I gadget CUIC utilizzano una griglia D per il rendering dei report. È necessario eseguire un processo di bootstrap, che viene eseguito direttamente in combinazione con CUIC. Per questo motivo, i gadget CUIC inizialmente parlano direttamente con CUIC Server durante il processo di caricamento. Per questo motivo, il certificato CUIC deve essere accettato nel browser utente (oltre al tunnel Socket.IO). I contenuti dei gadget e le API REST vengono inoltrati al client tra Finesse e CUIC. Le chiamate API REST vengono effettuate sia al Servizio di reporting del Centro intelligence che al Servizio Web CCX. Il servizio CCX Live Data Socket.IO riceve i messaggi da Live Data tramite JMS da ActiveMQ. Il servizio CCX Live Data Socket.IO pubblica il file JSON di report in tempo reale sulla connessione Socket.IO dal client. Analogamente al modo in cui Finesse Desktop dispone di un iFrame del tunnel BOSH che mantiene la connessione BOSH con il servizio di notifica Cisco Finesse, il gadget master Live Data dispone di un iFrame del tunnel Socket.IO che mantiene la connessione Socket.IO (websocket) con il servizio CCX Live Data Socket.IO.

L'hub OpenAjax distribuisce tutti gli eventi ai listener sottoscritti. Si tratterebbe sia di Gadget che di parti dello stesso Container Finesse. Finesse Desktop dispone di un iFrame del tunnel BOSH che mantiene la connessione BOSH con il servizio di notifica CCX unificato di Cisco. In questo modo gli eventi vengono pubblicati sull'hub OpenAjax.

Link di riferimento

- Guida per gli sviluppatori di Finesse Web Services

Informazioni su questa traduzione

Cisco ha tradotto questo documento utilizzando una combinazione di tecnologie automatiche e umane per offrire ai nostri utenti in tutto il mondo contenuti di supporto nella propria lingua. Si noti che anche la migliore traduzione automatica non sarà mai accurata come quella fornita da un traduttore professionista. Cisco Systems, Inc. non si assume alcuna responsabilità per l’accuratezza di queste traduzioni e consiglia di consultare sempre il documento originale in inglese (disponibile al link fornito).