Comprender en profundidad la arquitectura UCCX Finesse

Contenido

Introducción

Este documento describe la arquitectura de Finesse de una manera exhaustiva para que los procesos subyacentes tengan sentido mientras se solucionan problemas de finesse.

Prerequisites

Requirements

Cisco recomienda conocer estas herramientas y funciones:

JTAPI: API de telefonía Java

API: interfaz de programación de aplicaciones

UCCX: Unified Contact Center Express

CUCM: Cisco Unified Communications Manager

CTI - Integración de telefonía y ordenador

Componentes Utilizados

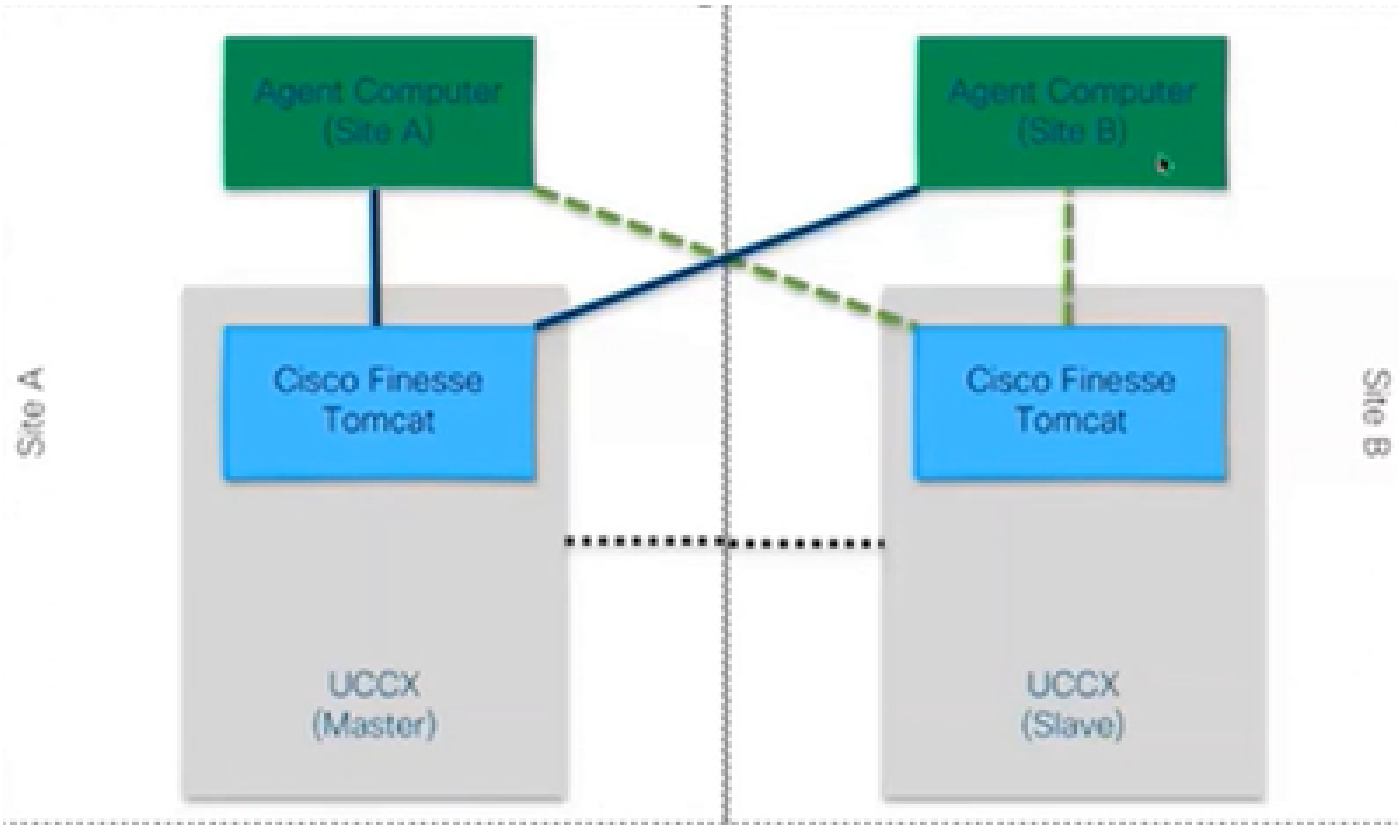
- Cisco Unified Contact Center Express (UCCX)

La información que contiene este documento se creó a partir de los dispositivos en un ambiente de laboratorio específico. Todos los dispositivos que se utilizan en este documento se pusieron en funcionamiento con una configuración verificada (predeterminada). Si tiene una red en vivo, asegúrese de entender el posible impacto de cualquier comando.

Antecedentes

Este documento describe la arquitectura Finesse a partir de una descripción general de alto nivel y luego el flujo de señal en profundidad junto con ejemplos y diagramas.

Vista de 50 000 pies



vista 50000

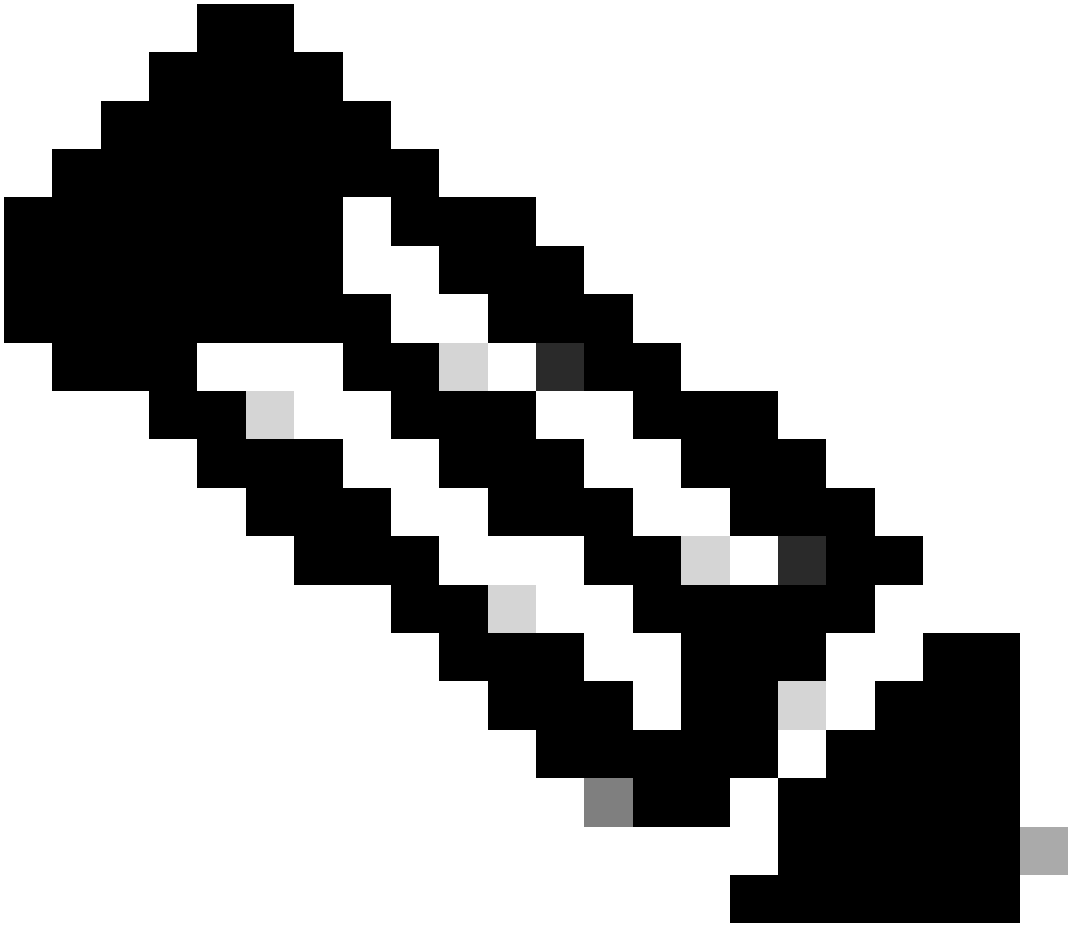
Finesse Tomcat

Finesse Tomcat es similar a Cisco Tomcat en CUCM, ya que la funcionalidad es la misma para cargar las páginas web, pero para un servicio diferente llamado Finesse. Tomcat se ha diseñado para Finesse porque es una aplicación web independiente. Solo puede iniciar sesión en finesse utilizando el nodo maestro CCX según las versiones anteriores a la 11.5. A partir de la versión 11.6, puede iniciar sesión en cualquier nodo (pero no se recomienda), solo durante la conmutación por error. Por lo tanto, si considera un clúster WAN, donde ambos agentes (en el sitio A y el sitio B) están conectados a Finesse en el nodo maestro, entonces en todo momento, hay una conexión inactiva con el otro nodo desde Cisco Finesse al motor , que es una solicitud CTI openconf que se requiere para la conmutación por fallas.

La solicitud Openconf se envía regularmente para verificar si el nodo está activo o en espera. En caso de que haya una conmutación por error, el servicio de notificación utiliza una API denominada systeminfo API que indica al cliente que este nodo está inactivo y que se necesita una conmutación por error. Luego se ejecuta un archivo que redirige el navegador al otro nodo y luego se envía openconf rejavascriptesponse. Esta conmutación por error sólo funciona si se aceptan los certificados. (para establecer una conexión segura con el servidor).

Se presentan tres certificados, a saber:

- Certificado de servicio de notificación para ese nodo.
- Certificado de servicio de notificación para el nodo remoto.
- Certificado de servicio Finesse para nodo remoto.



Nota: Los certificados de nodo remoto deben aceptarse para que funcione la conmutación por error.

HTTP(S)

Es un protocolo de capa de aplicación para enviar documentos hipermedia como HTML. Está diseñado para la comunicación entre los navegadores web y los servidores web.Es un protocolo sin estado que significa que el servidor no mantiene ningún dato entre las dos solicitudes. Este es un protocolo de servidor cliente. Para UCCX, el cliente Finesse se ejecuta en el explorador del agente (PC). Realiza una solicitud al servidor mediante HTTP. Estos son algunos métodos HTTP comunes:

GET: para obtener información de un servidor.
POST: para enviar información a un servidor.

PUT: para sustituir cualquier elemento de un servidor.
ELIMINAR: para quitar información de un servidor.

Finesse utiliza solicitudes api de systeminfo dentro de la solicitud http. Por ejemplo, si desea cambiar el estado de un agente, el navegador envía un PUT en lugar de POST porque si se envía POST, el servidor se confunde ya que tiene 2 estados en la mano, ¿cuál seleccionar? Así que usando PUT, reemplaza el estado actual.

XMPP

Protocolo extensible de mensajería y presencia

Se trata de un conjunto de protocolos que se utilizan para la mensajería instantánea y la presencia. Todas las entidades XMPP se identifican mediante sus JID o IDS de Jabber. Una de las extensiones de este protocolo XMPP que se utiliza para gadgets se conoce como PUBSUB.

PUBSUB

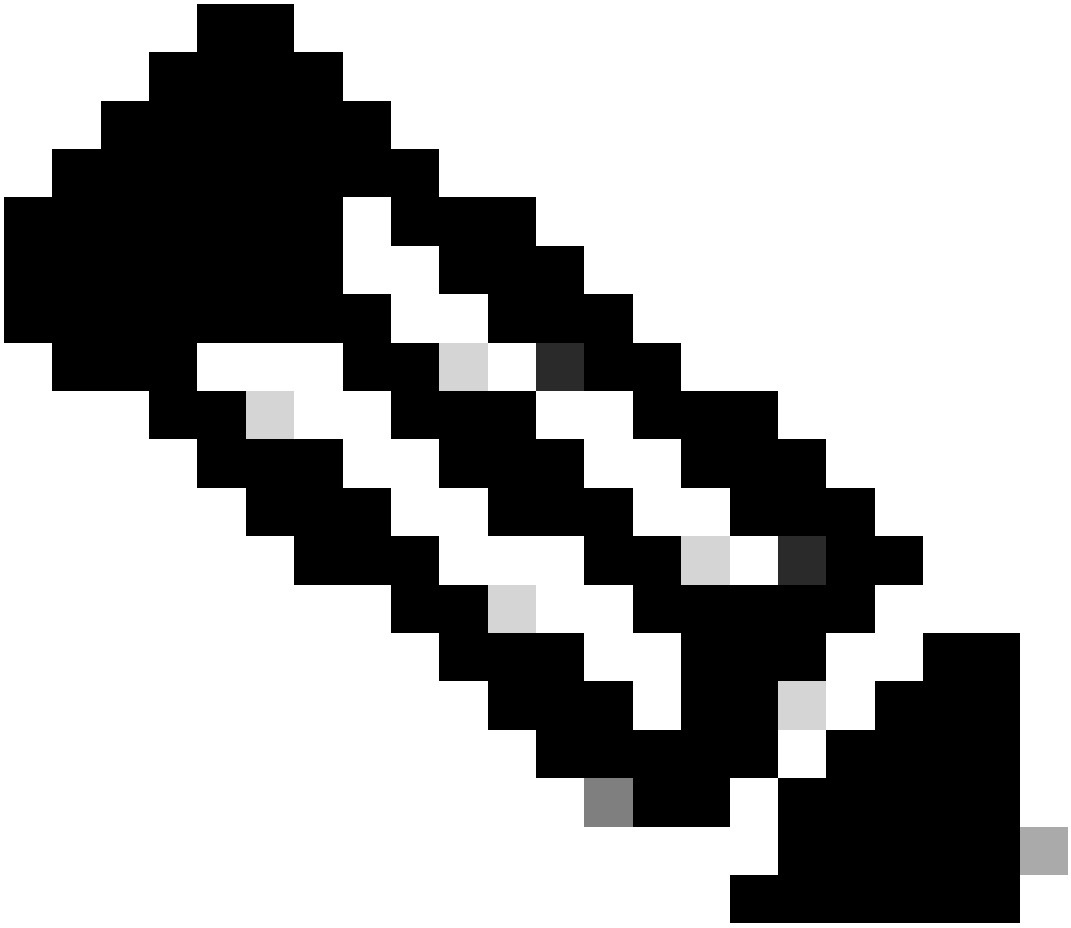
No es el editor suscriptor en el que se puede pensar. Todavía publica y suscribe, pero no tiene nada que ver con las bases de datos. XMPP utiliza un mecanismo denominado nodos. El nodo es básicamente la supervisión de la entidad que le importa. Cualquier cosa que sea importante para usted y desee monitorearlo, usted le agrega un nodo. Entonces, lo que PUBSUB hace es suscribirse a las actualizaciones o notificaciones en ese nodo. Puede tener nodos para cada tipo de entidad, como agentes, diálogos, etc. Si crea un nodo para un agente, se le suscribe al nodo de ese agente y, a continuación, se le notifica lo que haga el agente.

El propósito de esta especificación es permitir que el servidor XMPP (servicio de notificación) obtenga información publicada en nodos XMPP (temas) y, a continuación, envíe eventos XMPP a entidades suscritas a ese nodo.

En el caso de Finesse, el servidor de Integración de telefonía y ordenador (CTI) envía mensajes CTI al servicio web de Finesse para indicarle a Finesse las actualizaciones de configuración, como, entre otras, la creación de agentes o colas de servicio de contacto (CSQ) o información sobre una llamada. Esta información se convierte en un mensaje XMPP que el servicio web Finesse publica en el servicio de notificación Finesse.El servicio de notificación Finesse envía a continuación mensajes XMPP sobre BOSH a los agentes que están suscritos a determinados nodos XMPP.

BOSH: Flujos bidireccionales sobre HTTP síncrono

BOSH es una conexión HTTP de larga duración en la que el servidor mantiene la solicitud durante más tiempo hasta que tiene una respuesta; de lo contrario, envía una respuesta vacía. Esto funciona para clientes XMPP y servidores XMPP, pero también se puede utilizar para aplicaciones que no sean XMPP.



Nota: XMPP es stateful mientras que HTTP es stateless (no almacena la información sobre la última solicitud)

Si una aplicación web necesita funcionar con XMPP, surgen varios problemas.

Problema 1: Los exploradores no admiten XMPP a través del protocolo de control de transmisión (TCP) de forma nativa.

Solución 1: los servidores y exploradores Web se comunican a través de mensajes HTTP (Protocolo de transferencia de hipertexto), por lo que Finesse y otras aplicaciones Web incluyen los mensajes XMPP dentro de los mensajes HTTP.

Problema 2: HTTP es un protocolo sin estado.

Solución 2: Puede utilizar cookies/publicar datos para esto.

Problema 3: El tercer problema es el comportamiento unidireccional de HTTP, lo que significa que sólo el cliente envía solicitudes y el servidor sólo puede responder. La incapacidad del servidor para insertar datos hace que no sea natural implementar XMPP sobre HTTP.

Solución 3: para solucionar este problema, debe disponer de un puente entre HTTP y XMPP.

Las soluciones propuestas son:

- Sondeo (protocolo heredado): solicitudes HTTP repetidas que solicitan nuevos datos definidos: sondeo HTTP

- El sondeo largo también se conoce como BOSH: protocolo de transporte que emula la semántica de una conexión TCP bidireccional de larga duración entre dos entidades mediante el uso eficiente de múltiples pares de solicitud/respuesta HTTP sincrónicos sin requerir el uso de sondeos frecuentes. La razón para utilizar BOSH es encubrir el hecho de que el servidor no tiene que responder tan pronto como hay una solicitud. La respuesta se retrasa hasta un tiempo especificado hasta que el servidor tiene datos para el cliente y, a continuación, se envía como respuesta. Tan pronto como el cliente obtiene la respuesta, el cliente hace una nueva solicitud y así sucesivamente.

El cliente de escritorio Finesse (aplicación web) establece una conexión BOSH obsoleta a través del puerto TCP 7443 cada 30 segundos. Después de 30 segundos, si no hay actualizaciones del servicio de notificación de Finesse, el servicio de notificación envía una respuesta HTTP con 200 OK y un cuerpo de respuesta (casi) vacío. Si el servicio de notificación tiene una actualización de la presencia de un agente o un evento de diálogo (llamada), por ejemplo, los datos se envían inmediatamente al cliente web de Finesse.

Para resumir:

El cliente web Finesse tiene una conexión HTTP obsoleta (http-bind) configurada en el servidor Finesse a través del puerto TCP 7443. Esto se conoce como una encuesta larga de BOSH. El servicio de notificación de Finesse es un servicio de presencia que publica actualizaciones relacionadas con el estado de un agente, una llamada, etc. Si el servicio de notificación tiene una actualización, responde a la solicitud http-bind con la actualización de estado como un mensaje XMPP en el cuerpo de la respuesta HTTP. Si no hay actualizaciones de estado 30 segundos después de recibir la solicitud http-bind, el servicio de notificación responde sin ninguna actualización de estado para permitir que el cliente web Finesse envíe otra solicitud http-bind. Esto sirve como una manera para que el servicio de notificación sepa que el cliente web Finesse todavía puede conectarse al servicio de notificación y que el agente no cerró su navegador o puso su computadora en suspensión, y así sucesivamente.

CTI

Puede utilizar la Integración de telefonía y ordenador (CTI) para aprovechar las funciones de procesamiento informático mientras realiza, recibe y gestiona llamadas telefónicas. Las aplicaciones CTI le permiten realizar tareas como recuperar información de usuario de una base de datos mediante un ID de la persona que llama o trabajar con la información recopilada por un sistema de respuesta de voz interactiva (IVR) para enrutar una llamada procedente del usuario junto con su información al representante de servicio adecuado. CTI Manager en CUCM responde a las solicitudes JTAPI de UCCX. El puerto TCP del servidor CTI es 12018. Así es como Finesse Server y Engine (CTI Server) se comunican entre sí.

He aquí parte de la información intercambiada a través de CTI:

- Configuración actual del sistema y actualizaciones futuras.

- Agentes y sus estados.

- Llamadas y sus estados.

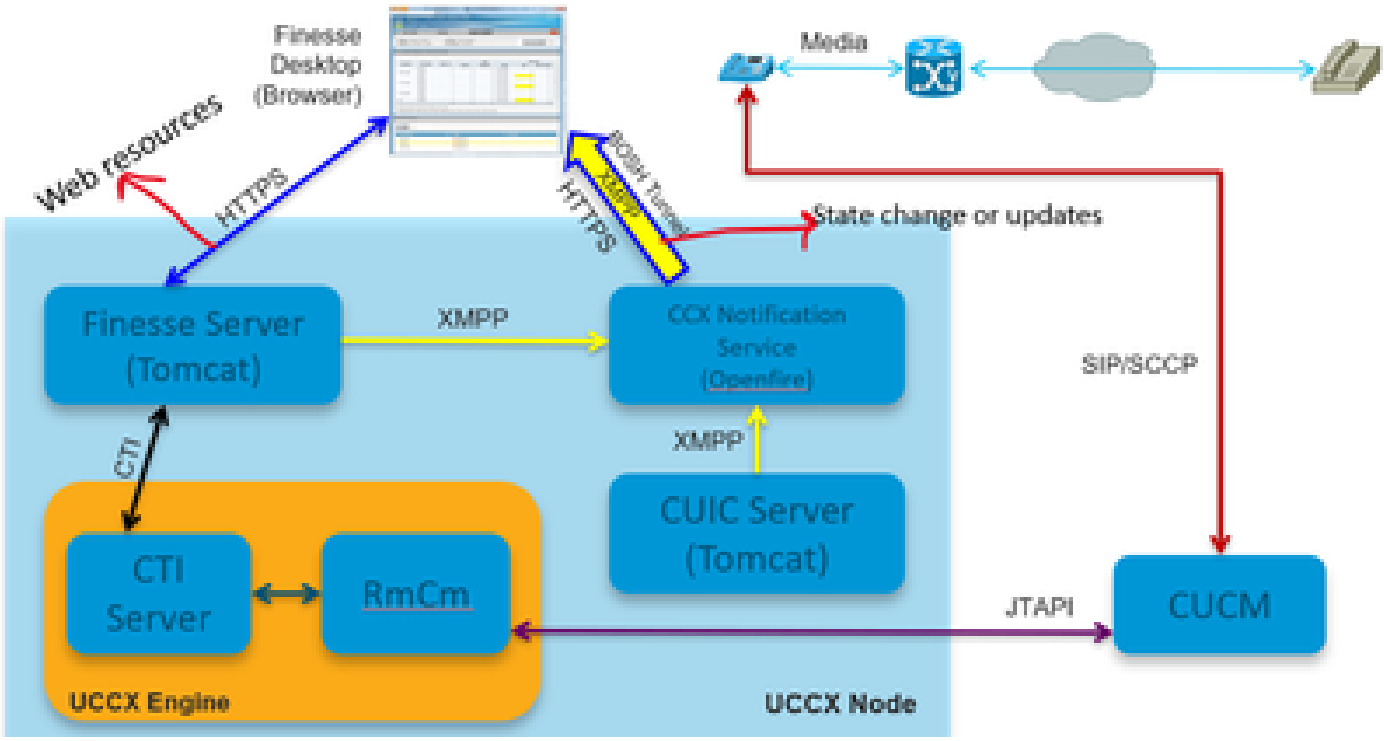
- Estadísticas de agentes, llamadas y colas en tiempo real.

JTAPI

Cisco Unified JTAPI actúa como un estándar de interfaz de programación desarrollado por Sun Microsystems para su uso con aplicaciones de telefonía y ordenador basadas en Java. Cisco JTAPI implementa la especificación Sun JTAPI 1.2 con extensiones adicionales de Cisco. Cualquier comunicación entre UCCX y CUCM reside en JTAPI. Así es como CUCM y Engine (subsistema de telefonía) se comunican entre sí. JTAPI se utiliza para controlar y supervisar teléfonos CUCM, enrutar llamadas mediante puertos CTI y puntos de ruta, iniciar y detener grabaciones en CUCM y para cualquier funcionalidad de enrutamiento de llamadas

vista de 30000 pies

El siguiente diagrama describe cómo el motor UCCX, Finesse, CUCM y Browser se comunican entre sí.

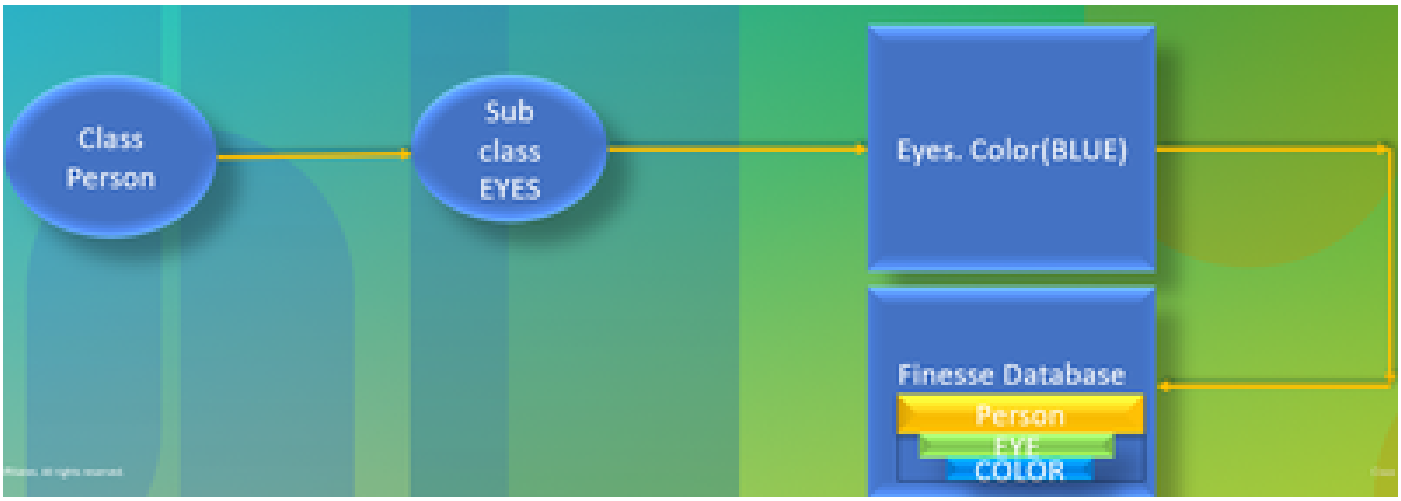


vista 30000

Consideremos que la llamada se establece con el agente. Ahora, RmCm que monitorea la extensión del agente a través de JTAPI, le informa al servidor CTI sobre el cambio de estado que el agente está hablando. Esta información se envía desde el servidor CTI (dentro del motor CCX) al servidor Finesse (Tomcat) mediante CTI. El servidor Finesse envía esta información al servicio de notificación de CCX mediante XMPP sobre el cambio de estado. El servicio de notificación (Openfire) abre un túnel BOSH al navegador del agente y actualiza la información sobre el cambio de estado y es así como ve que el agente pasa al estado RESERVED. Se solicita cualquier tipo de recursos web al servidor Finesse mediante HTTPS, como archivos WAR, gadgets, etc. (si no se encuentra ya en la caché).

HIBERNAR

El siguiente diagrama explica el servicio de hibernación.



hibernar

HIBERNATE se denomina Servicio de consultas y persistencia relacional/objeto de alto rendimiento. En pocas palabras, asigna clases JAVA a tablas de bases de datos. Por ejemplo, tiene un objeto JAVA denominado Team y una tabla de base de datos en la base de datos finesse denominada Team. La clase JAVA controla qué información se encuentra dentro de la tabla e HIBERNATE es lo que hace que esto suceda. En lugar de utilizar consultas SQL, utiliza clases Java para actualizar la información.

AXL

XML administrativo.

XML significa lenguaje de marcado extensible y es un lenguaje de marcado que define algunas reglas relativamente simples para codificar datos. Se diseñó principalmente para transmitir y recibir datos estructurados en un formato bien definido que ambos sistemas puedan comprender. En el formato más básico, XML define etiquetas que se incluyen entre corchetes angulares (<>) y estas etiquetas rodean los datos descritos por la etiqueta. Las etiquetas pueden formar una jerarquía con etiquetas dentro de otras etiquetas. Por ejemplo, para definir un dispositivo telefónico básico, puede decir que un dispositivo telefónico necesita tres parámetros, un nombre, una descripción y un número de teléfono.

Es una API basada en SOAP que permite el aprovisionamiento remoto en CUCM. Se utiliza para agregar, actualizar, quitar o recuperar información de la base de datos de CUCM. Las funciones de recuperación incluyen la comprobación de la autenticación de usuario y la ejecución de consultas SQL. La API AXL proporciona acceso a toda la base de datos de CUCM. La API de AXL es únicamente para el aprovisionamiento y no proporciona acceso a los datos de rendimiento o en tiempo de ejecución.

La API de AXL utiliza la autenticación básica de HTTPS. Cualquier usuario de CUCM (aplicación o usuario final) tiene acceso de lectura y escritura a través de AXL si es miembro del grupo de control de acceso **Superusuarios de CCM estándar** o de cualquier grupo con la función de **acceso API AXL estándar** asignada. Esto significa que todas las cuentas de superusuario implícitamente ya tienen acceso a la API AXL. Para crear una cuenta dedicada para el uso de la API AXL, primero debe crear un grupo de control de acceso y asignarle la función **Standard AXL API Access** y, a continuación, asociar el usuario de la aplicación con el grupo recién creado. Para proporcionar acceso API AXL de sólo lectura, puede crear un grupo de control de acceso independiente y asignarle únicamente la función **Standard AXL Read Only API Access**.

JABÓN

El Protocolo simple de acceso a objetos (SOAP) es una forma de pasar información entre aplicaciones en formato XML. Los mensajes SOAP se transmiten desde la aplicación de envío a la aplicación de recepción, normalmente a través de una sesión HTTP. El mensaje SOAP real se

compone del elemento Envelope, que contiene un elemento Body y un elemento Header opcional.

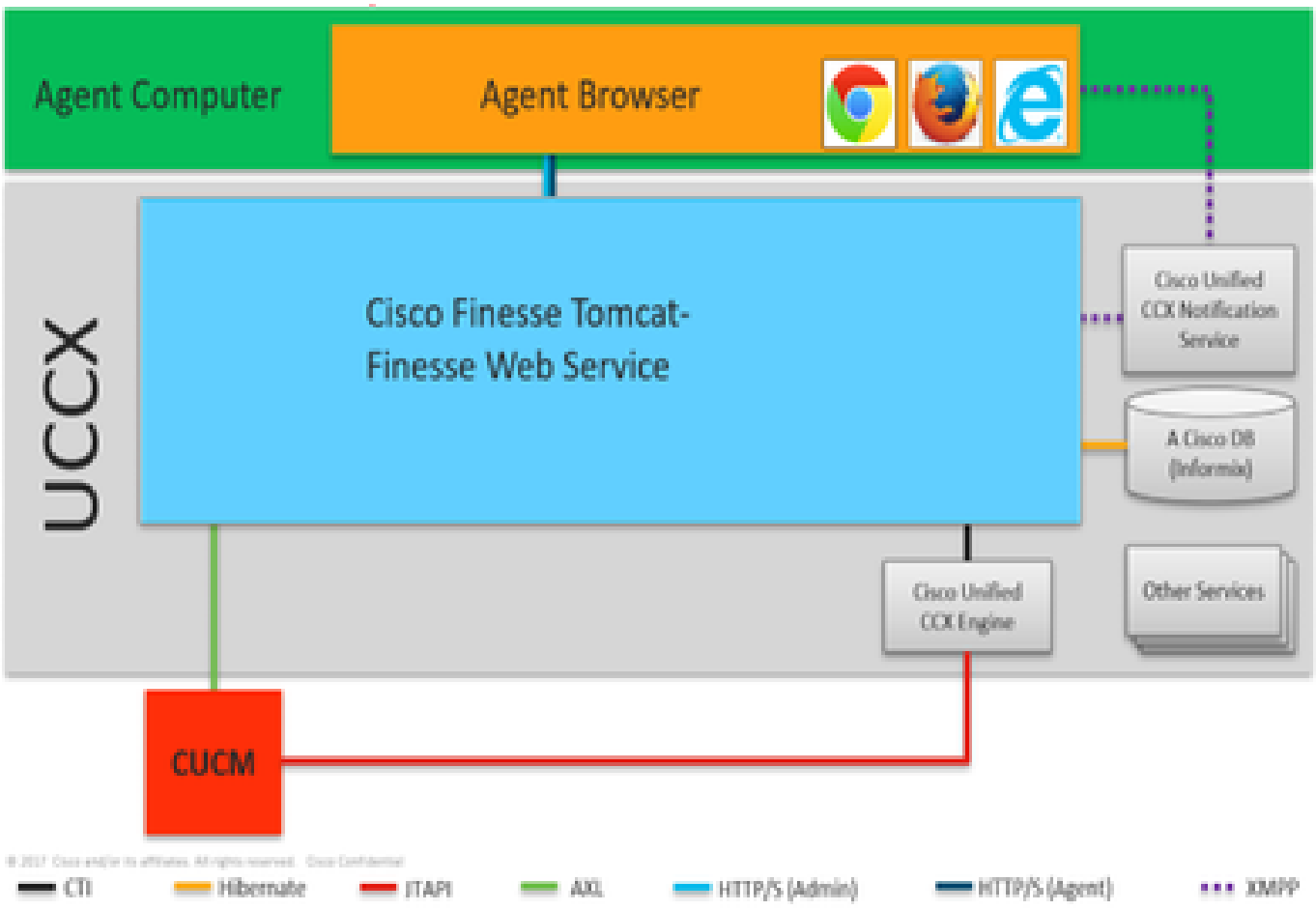
- Envelope - Este elemento obligatorio es la raíz del mensaje SOAP, identificando el XML transmitido como un paquete SOAP. Un sobre contiene una sección de cuerpo y una sección de encabezado opcional.

- Header - Este elemento opcional proporciona un mecanismo de extensión que indica la información de procesamiento del mensaje. Por ejemplo, si la operación que utiliza el mensaje requiere credenciales de seguridad, dichas credenciales deben formar parte del encabezado del sobre.

- Body - Este elemento contiene la carga útil del mensaje, los datos sin procesar que se transmiten entre las aplicaciones de envío y recepción. El propio cuerpo puede constar de varios elementos secundarios, con un esquema XML que suele definir la estructura de estos datos.

vista de 20000 pies

En el siguiente diagrama se explican de forma más detallada los protocolos implicados en la arquitectura Finesse.



vista 20000

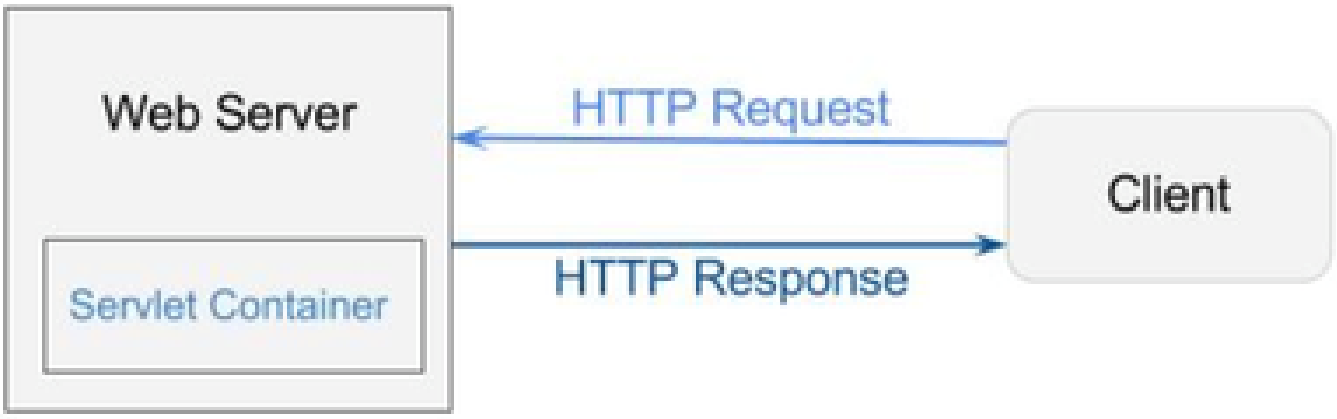
Estos son los protocolos responsables de la comunicación entre los diferentes componentes de UCCX.

- El motor UCCX y el servidor Finesse hablan sobre el protocolo CTI.

- El motor UCCX y CUCM hablan sobre el protocolo JTAPI.

- Finesse Tomcat y CUCM hablan sobre AXL.
- Servicio de notificación Finesse y conversación del navegador del agente en XMPP/BOSH.
- Finesse Tomcat y la base de datos hablan sobre Hibernate.
- Finesse Tomcat y Finesse Notification hablan sobre XMPP.

APACHE SHINDIG



Shindig

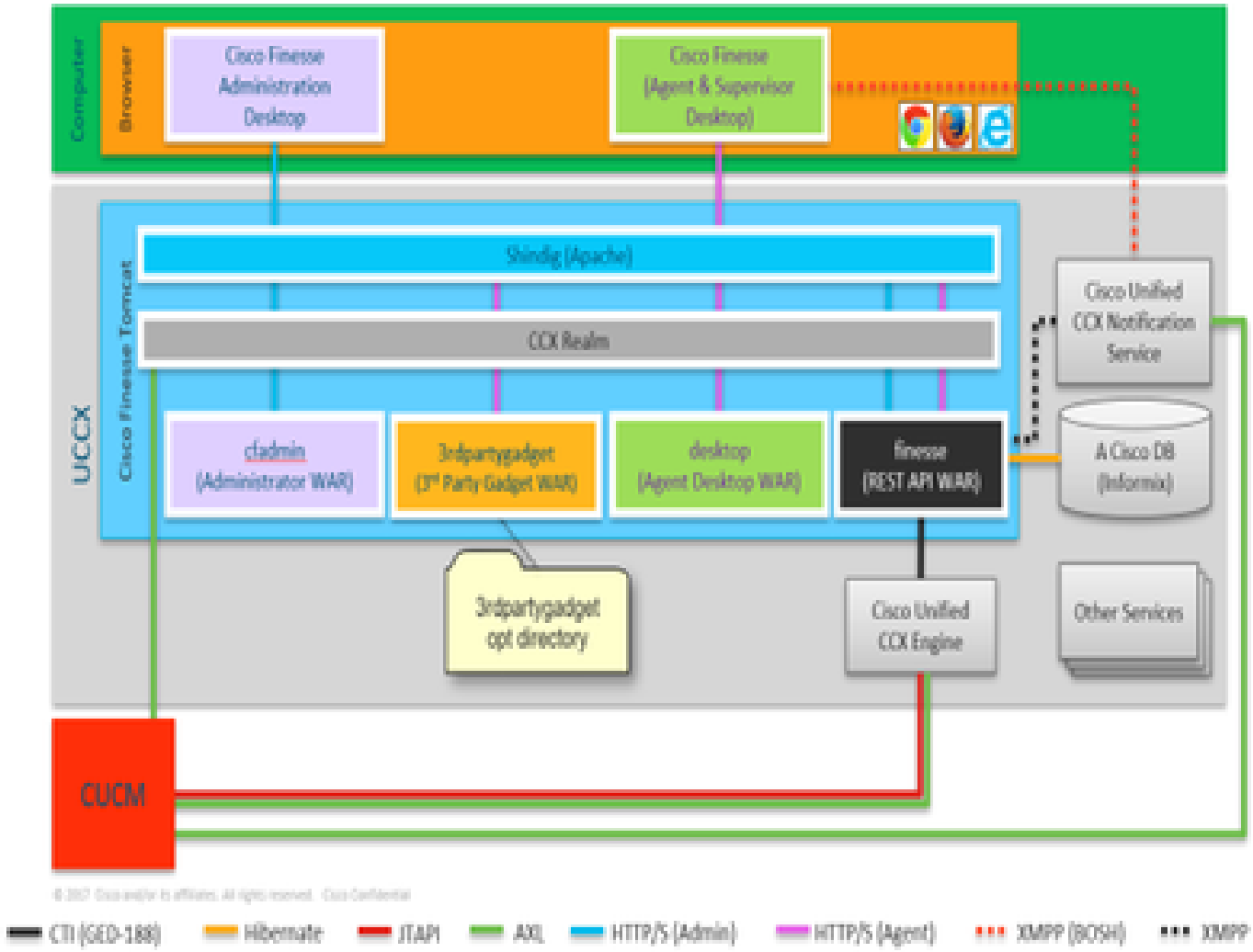
Apache Shinding es un contenedor de OpenSocial y le ayuda a empezar a alojar aplicaciones de OpenSocial rápidamente proporcionando el código para representar gadgets, solicitudes de proxy y gestionar solicitudes REST y RPC. OpenSocial es un conjunto de API para crear aplicaciones sociales que se ejecutan en la Web. (Web/Servlet) Un servidor Web utiliza el contenedor para generar páginas Web de forma dinámica.

ARCHIVOS DE GUERRA

WAR significa Archivo Web. Contiene archivos de un proyecto Web. Puede tener servlet, XML, JSP, imagen, HTML, CSS, JS, etc. Los registros de Catalina contienen la información acerca de los WARs que se implementan.

vista de 10000 pies

El siguiente diagrama explica en detalle cómo funciona el flujo de autenticación dentro de los componentes de UCCX y Finesse.



vista 10000

Los archivos WAR son necesarios para mostrar y crear la página en función de cómo inicie sesión. El explorador le pregunta a Shindig que debe representar un gadget. A continuación, Shindig se dirige a CUIC para representar el gadget. El rango CCX se utiliza para la autenticación con CUCM mediante AXL. El servicio de notificación también autentica con CUCM mediante AXL.

Finesse Rest API WAR es el repositorio principal que realmente se comunica con el servicio de notificación, el motor CCX y DB. Shinding solo habla con la API Finesse Rest (WebServices) porque cfadmin y los WAR de escritorio son solo para mostrar la página. Cualquier cosa que venga a la guerra de API de Finesse Rest, usted puede ver que en los registros de Finesse WebServices que son los registros más importantes para finesse. Se habla de HTTP entre el servicio web Shinding y Finesse (API de resto WAR). El servicio web Finesse (Rest API WAR) y Engine se comunican entre sí a través de CTI.

AJAX - La belleza de Finesse

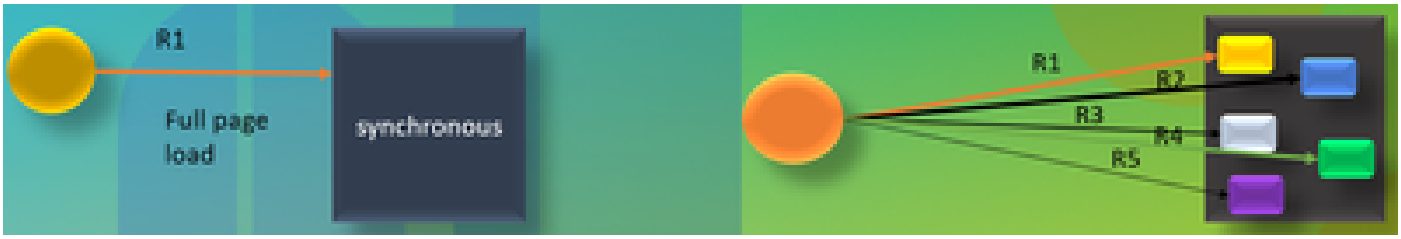
AJAX significa Asynchronous Javascript and XML. No es un lenguaje de programación, sino un método para tener acceso a los servidores Web desde una página Web. AJAX es un mecanismo para realizar actualizaciones parciales de la página. Permite actualizar secciones de una página con datos procedentes de un servidor sin necesidad de actualizar toda la página.

Por ejemplo, si habla de Facebook Messenger, cuando llega un nuevo mensaje, no necesita actualizar toda la página para obtener el mensaje, en lugar de eso, la sección de mensajes de la propia página se actualiza y obtiene los nuevos mensajes en tiempo real sin necesidad de actualizar toda la página.

Cada navegador tiene un objeto integrado llamado XMLHTTPREQUEST (también llamado **XHR**). Cada solicitud a AJAX en el servidor pasa a través de esta solicitud XML. Contiene los detalles específicos de lo que debe actualizar.

Ventajas del uso de AJAX

El siguiente diagrama explica la diferencia entre las solicitudes asincrónicas y sincrónicas.



AJAX

En caso de una solicitud síncrona, debe esperar a que se procese la primera solicitud y, a continuación, puede enviar la segunda solicitud. Por ejemplo, es necesario actualizar la página y no puede hacer nada hasta que se actualice la página. Por otro lado, en caso de una solicitud asincrónica, no tiene que esperar a que se complete la primera solicitud para enviar la segunda solicitud. Puede enviar varias solicitudes simultáneamente. Por ejemplo, gadgets de aplicaciones meteorológicas en sitios web. Puede actualizar la sección de tiempo de la página y, mientras tanto, también trabajar en las otras secciones del sitio web simultáneamente sin necesidad de actualizar toda la página. Esta es la principal ventaja de la solicitud asincrónica.

TRABAJO DE AJAX

AJAX es una combinación de XMLHTTPREQUEST (**XHR**) que se utiliza para enviar y recibir actualizaciones desde el servidor web junto con Javascript y HTML que se utilizan para mostrar o utilizar los datos.

ENVIANDO SOLICITUD CON AJAX AL SERVIDOR

Este es un proceso de 3 pasos que se menciona a continuación:

1. Crear una variable y almacenar el objeto **XHR** en ella.

Var request = new XMLHttpRequest();

2. Acceso a la variable de solicitud que tiene la carga útil dentro del objeto XHR.

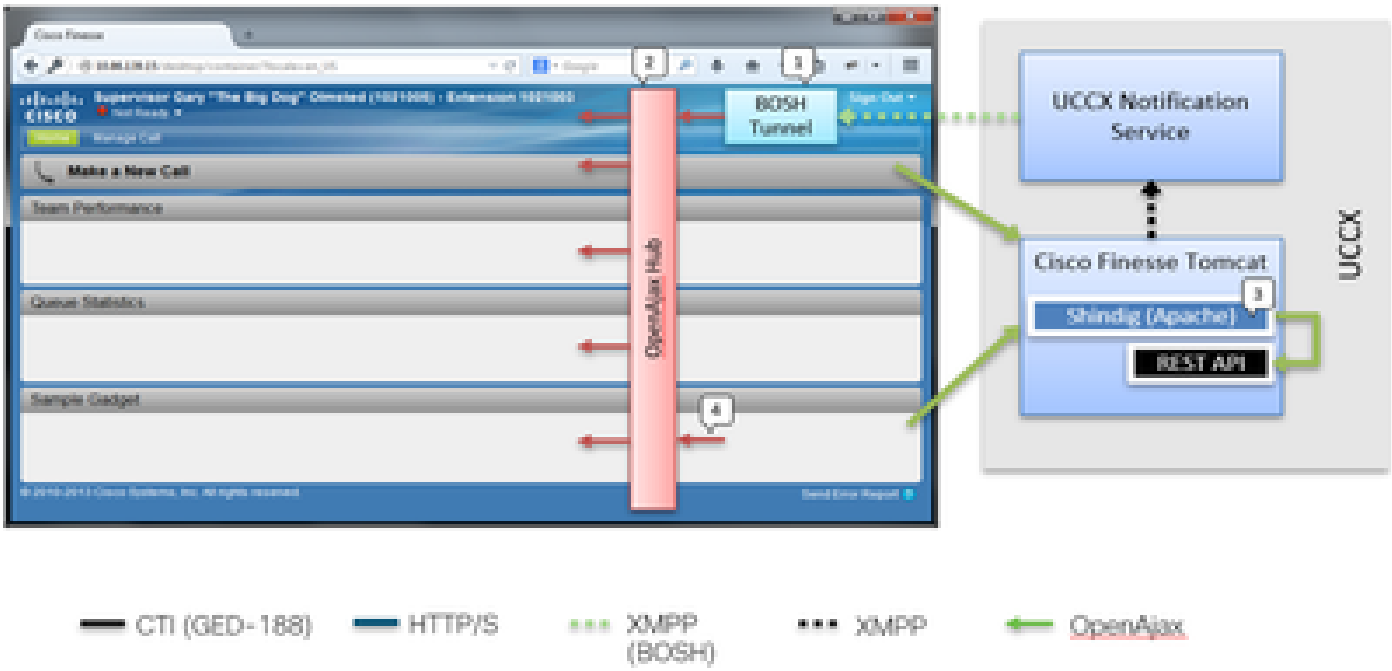
petición;.**open(GET, URL)**

3. Envío de la solicitud

Request.send() ;

Arquitectura de escritorio

En el siguiente diagrama se explica el flujo de señales AJAX cuando se procesan gadgets en la página web.

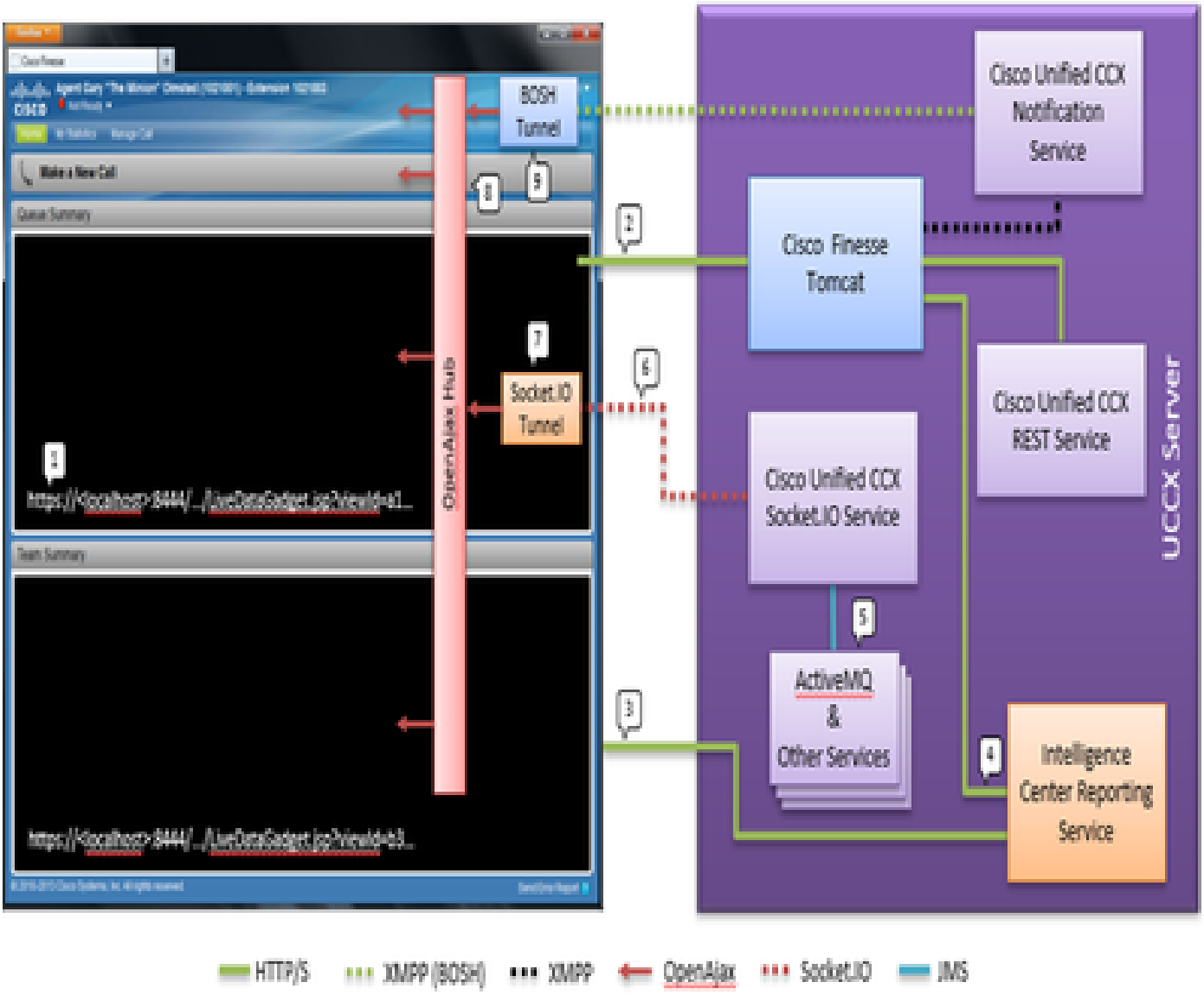


arquitectura de escritorio

IFrame reside en el contenedor para alojar el túnel BOSH. El concentrador OPENAJAX se proporciona para publicar mensajes a través de los gadgets (mediante el método pubsub). Las solicitudes REST también se procesan como proxy a través de Shinding a otros servidores. Los gadgets pueden publicar sus propios mensajes en el concentrador AJAX.

Arquitectura de gadgets

El siguiente diagrama explica la arquitectura del gadget de Finesse en detalle.



arquitectura de gadgets

A diferencia de los gadgets habituales, los gadgets de CUIC también reciben una fuente XMPP en tiempo real de OpenFire. En el caso de UCCX, donde CUIC y Finesse son co-residentes con UCCX, hay una instancia compartida de OpenFire. La mayor parte del contenido de los gadgets y todas las API REST se procesan como proxy a través de Shinding en el servidor Finesse. Esto va para Gadgets Finesse y API REST, así como instancias de Gadgets CUIC y API REST. Los gadgets de CUIC utilizan una cuadrícula D para representar sus informes. Se debe realizar un proceso de bootstrap junto con CUIC directamente. Por esta razón, los gadgets de CUIC se comunican inicialmente con el servidor de CUIC directamente durante el proceso de carga. Por este motivo, el certificado de CUIC debe aceptarse en el explorador del usuario (además del túnel Socket.IO). El contenido de los gadgets y las API REST se procesan como proxy para el cliente entre Finesse y CUIC. Las llamadas a la API de resto se realizan tanto al servicio de informes de Intelligence Center como al servicio web de CCX. El servicio CCX Live Data Socket.IO obtiene los mensajes de Live Data a través de JMS desde ActiveMQ. El servicio CCX Live Data Socket.IO publica JSON de informes en tiempo real a través de la conexión Socket.IO del cliente. De forma similar a como el escritorio Finesse tiene un iFrame de túnel BOSH que mantiene la conexión BOSH con el servicio de notificación Cisco Finesse, el Gadget de datos en directo principal tiene un iFrame de túnel Socket.IO que mantiene la conexión Socket.IO (websocket) con el servicio CCX Live Data Socket.IO.

El OpenAjax Hub distribuye todos los eventos a los oyentes suscritos. Esto sería tanto Gadgets como partes del propio contenedor Finesse. Finesse Desktop tiene un iFrame de túnel BOSH que mantiene la conexión BOSH con el servicio de notificación de Cisco Unified CCX. Esto publica eventos en el OpenAjax Hub.

Enlaces de referencia

- Guía del desarrollador de Finesse Web Services

Acerca de esta traducción
Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).