Entenda o detalhamento da arquitetura do UCCX Finesse

Contents

Introdução

Este documento descreve a arquitetura Finesse de uma maneira completa para que os processos subjacentes façam sentido durante a solução de problemas finesse.

Pré-requisitos

Requisitos

A Cisco recomenda o conhecimento destas ferramentas e recursos:

JTAPI - API de telefonia Java

API - Interface de programação de aplicativos

UCCX - Unified Contact Center Express

CUCM - Cisco Unified Communications Manager

CTI - Integração entre telefonia e computador

Componentes Utilizados

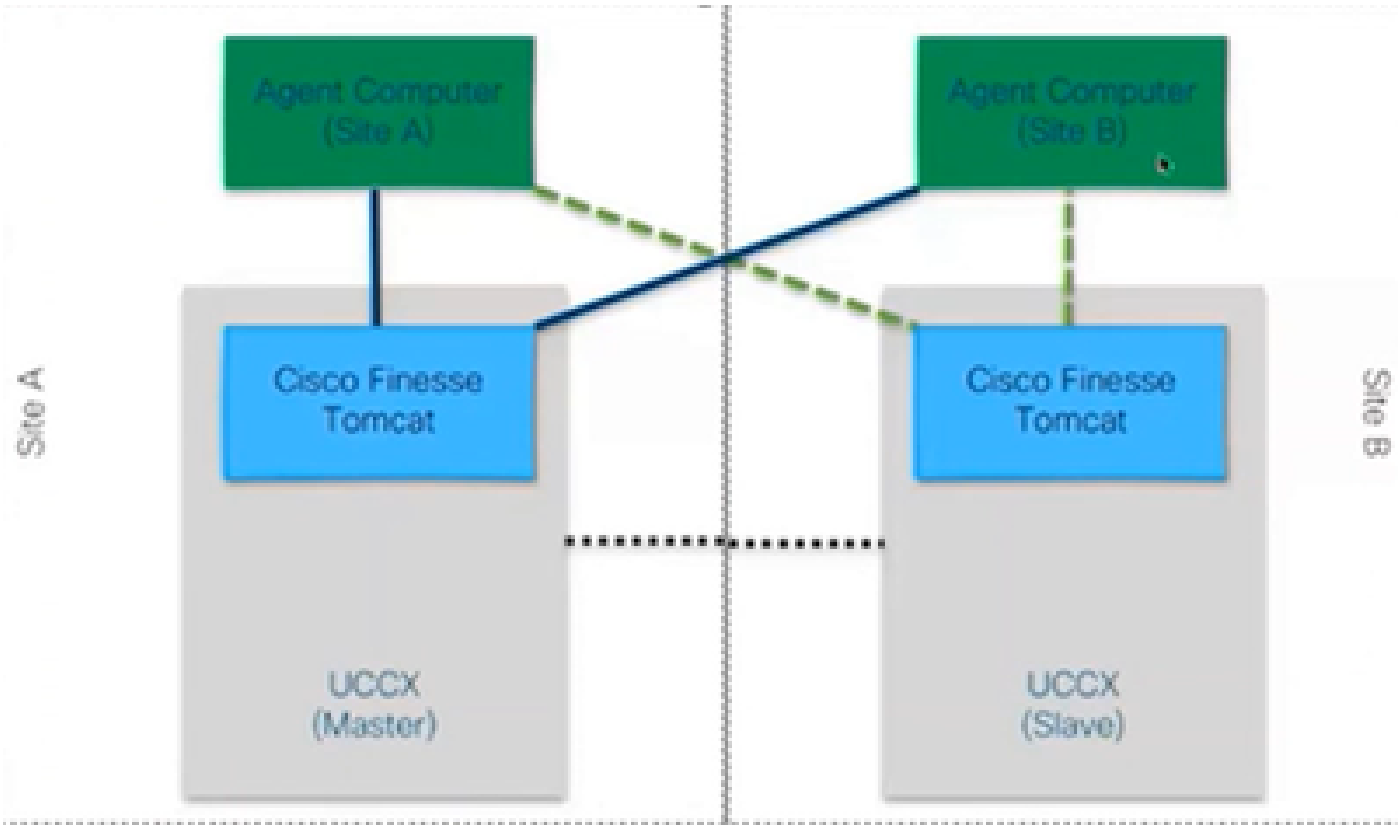
- Cisco Unified Contact Center Express (UCCX)

As informações neste documento foram criadas a partir de dispositivos em um ambiente de laboratório específico. Todos os dispositivos utilizados neste documento foram iniciados com uma configuração (padrão) inicial. Se a rede estiver ativa, certifique-se de que você entenda o impacto potencial de qualquer comando.

Informações de Apoio

Este documento descreve a arquitetura Finesse a partir de uma visão geral de alto nível e depois o fluxo de sinal em profundidade, juntamente com exemplos e diagramas.

50.000 pés de vista



exibição 50000

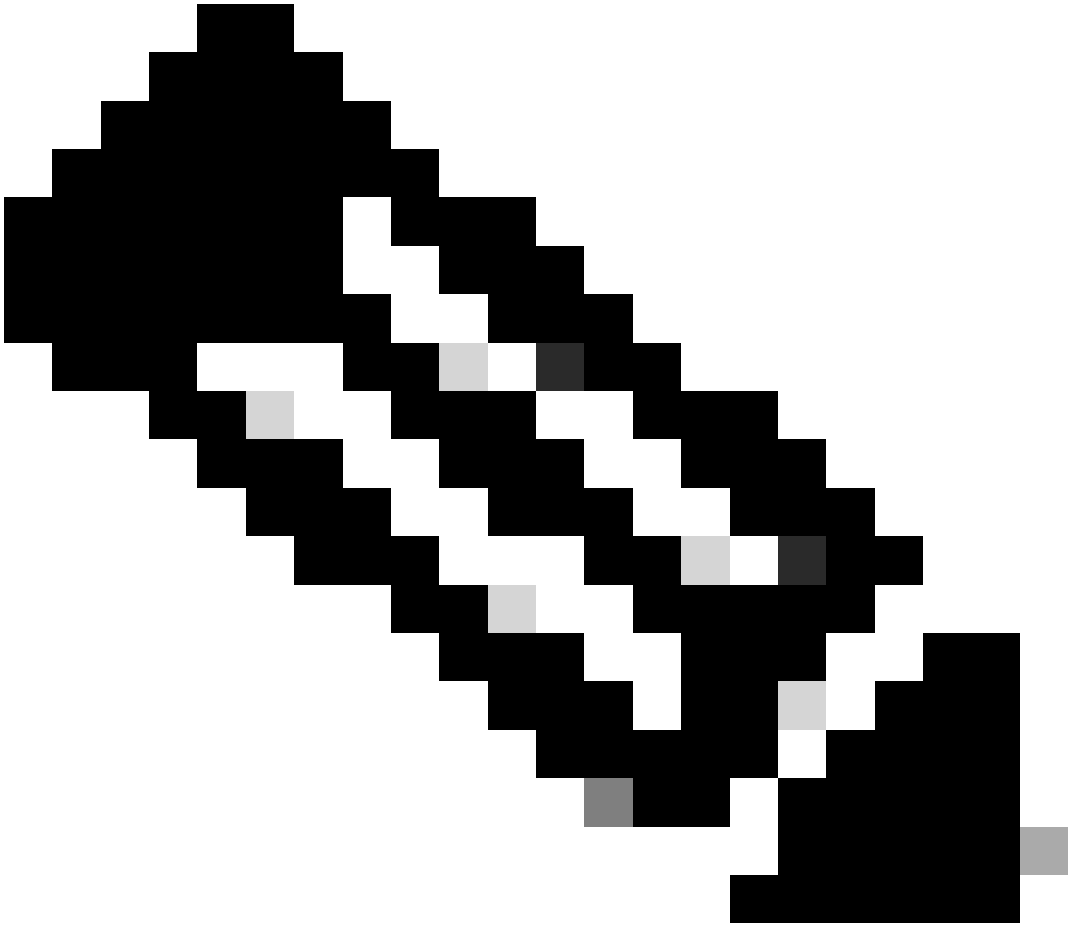
Finesse Tomcat

O Finesse Tomcat é semelhante ao Cisco Tomcat no CUCM, pois a funcionalidade é a mesma para carregar as páginas da Web, mas para um serviço diferente chamado Finesse. O Tomcat foi projetado para Finesse porque é uma aplicação Web separada. Você só pode fazer login no finesse usando o nó mestre do CCX de acordo com as versões anteriores à 11.5. A partir da versão 11.6, você pode efetuar login em qualquer nó (mas não recomendado), somente durante o failover. Assim, se você considerar um cluster de WAN, onde ambos os agentes (no site A e no site B) estão conectados ao Finesse no nó principal, então, em todos os momentos, há uma conexão inativa com o outro nó do Cisco Finesse para o mecanismo , que é uma solicitação CTI openconf que é necessária para failover.

A solicitação de Openconf é enviada regularmente para verificar se o nó está ativo ou em espera. Caso haja um failover, o serviço de notificação usa uma API chamada systeminfo API que informa ao cliente que esse nó está inativo e um failover é necessário. Em seguida, é executado um arquivo que redireciona o navegador para o outro nó e, em seguida, o comando openconf rejavascriptsponse é enviado. Este failover funciona somente se os certificados forem aceitos. (para estabelecer uma conexão segura com o servidor).

São apresentados três certificados:

- Certificado do serviço de notificação para esse nó.
- Certificado do serviço de notificação para o nó remoto.
- Certificado de serviço Finesse para nó remoto.



Observação: os certificados do nó remoto precisam ser aceitos para que o failover funcione.

HTTP(S)

É um protocolo da camada de aplicação para o envio de documentos de hipermídia, como HTML. Ele é projetado para comunicação entre navegadores e servidores Web.É um protocolo stateless que significa que o servidor não mantém nenhum dado entre as duas solicitações. Este é um protocolo cliente-servidor. Para o UCCX, o cliente Finesse é executado no navegador do agente (PC). Ele faz uma solicitação ao servidor usando HTTP. Aqui estão alguns métodos HTTP comuns:

GET - para obter informações de um servidor.
POST - para enviar informações a um servidor.

PUT - para substituir qualquer coisa em um servidor.
DELETE - para remover informações de um servidor.

O Finesse usa solicitações de API systeminfo dentro da solicitação http. Por exemplo, se você desejar alterar o estado de um agente, o navegador enviará um PUT em vez de POST porque, se o POST for enviado, o servidor ficará confuso porque tem 2 estados à mão, qual selecionar? Então, usando a PUT, ela substitui o estado atual.

XMP

eXtensible Messaging and Presence Protocol (Protocolo de Presença e Mensagens Extensíveis)

É um conjunto de protocolos usados para mensagens instantâneas e presença. Todas as entidades XMPP são identificadas usando seus Jabber IDS ou JIDs. Uma das extensões desse protocolo XMPP usada para gadgets é conhecida como PUBSUB.

PUBSUB

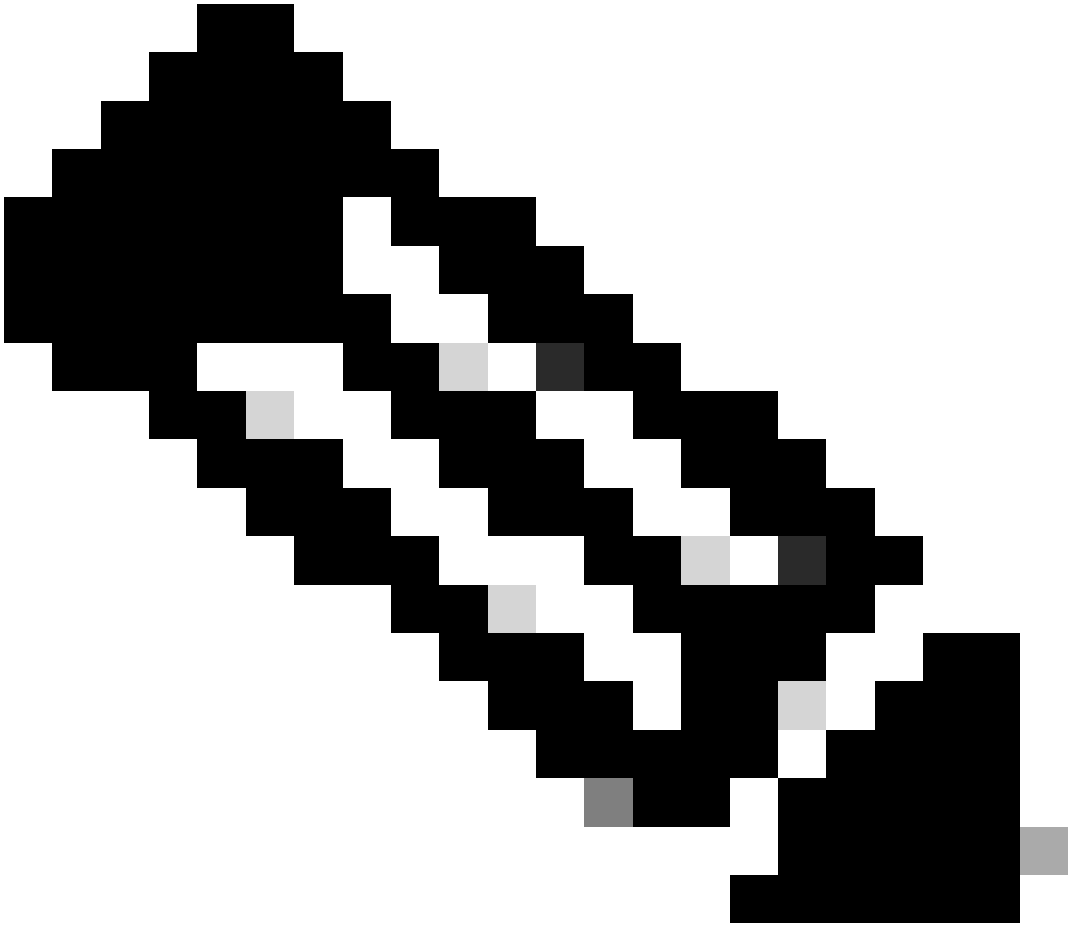
Não é o assinante do editor em que você pode pensar. Ele ainda publica e assina, mas não tem nada a ver com as bases de dados. O XMPP usa um mecanismo chamado nós. O nó está basicamente monitorando aquela entidade com a qual você se importa. Qualquer coisa que seja importante para você e deseje monitorá-la, você adiciona um nó a ela. Em seguida, o que o PUBSUB faz é assinar as atualizações ou notificações nesse nó. Você pode ter nós para cada tipo de entidade, como agente, caixa de diálogo e assim por diante. Se você criar um nó para um agente, será inscrito no nó desse agente e, em seguida, o que o agente fizer será notificado sobre isso.

A finalidade desta especificação é permitir que o servidor XMPP (serviço de Notificação) obtenha informações publicadas para nós XMPP (tópicos) e envie eventos XMPP para entidades inscritas nesse nó.

No caso do Finesse, o servidor de Integração entre Telefonia e Computador (CTI - Computer Telephony Integration) envia mensagens de CTI ao serviço da Web Finesse para informar ao Finesse sobre atualizações de configuração, como, mas não limitado a, criação de agentes ou de filas de serviço de contato (CSQ - Contact Service Queue) ou informações sobre uma chamada. Essas informações são então convertidas em uma mensagem XMPP que o serviço Web Finesse publica para o serviço de Notificação Finesse.O serviço de Notificação Finesse envia mensagens XMPP sobre BOSH para agentes que são assinados para determinados nós XMPP.

BOSH - Fluxos bidirecionais sobre HTTP síncrono

BOSH é uma conexão HTTP de longa duração em que o servidor mantém a solicitação por mais tempo até que tenha uma resposta a ela, caso contrário, envia uma resposta vazia. Isso funciona para clientes XMPP e servidores XMPP, mas também pode ser usado para aplicativos não XMPP.



Observação: o XMPP é stateful, enquanto o HTTP é stateless (ele não armazena as informações sobre a última solicitação)

Se uma aplicação Web precisar trabalhar com XMPP, vários problemas surgirão.

Problema 1: Os navegadores não suportam XMPP sobre TCP (Transmission Control Protocol) nativamente.

Solução 1: Os servidores Web e os navegadores se comunicam através de mensagens do protocolo de transferência de hipertexto (HTTP), de modo que o Finesse e outros aplicativos Web empacotam mensagens XMPP dentro de mensagens HTTP.

Problema 2: O HTTP é um protocolo stateless.

Solução 2: Você pode usar cookies/postar dados para isso.

Problema 3: O terceiro problema é o comportamento unidirecional do HTTP, o que significa que somente o cliente envia solicitações e o servidor só pode responder. A incapacidade do servidor de enviar dados torna não natural a implementação de XMPP sobre HTTP.

Solução 3: para superar esse problema, você precisa ter uma ponte entre HTTP e XMPP.

As soluções propostas são:

- Polling (protocolo herdado): solicitações HTTP repetidas solicitando novos dados definidos: Polling HTTP

- O polling longo também é conhecido como BOSH: protocolo de transporte que emula a semântica de uma conexão TCP bidirecional de longa duração entre duas entidades usando eficientemente vários pares de solicitação/resposta HTTP síncronos sem exigir o uso de polling frequente. A razão para usar BOSH é para encobrir o fato de que o servidor não precisa responder assim que há uma solicitação. A resposta é atrasada até um tempo especificado até que o servidor tenha dados para o cliente e, em seguida, é enviada como uma resposta. Assim que o cliente obtém a resposta, ele faz uma nova solicitação e assim por diante.

O cliente desktop Finesse (aplicação Web) estabelece uma conexão BOSH obsoleta sobre a porta TCP 7443 a cada 30 segundos. Após 30 segundos, se não houver atualizações do Finesse Notification Service, o serviço de Notificação enviará uma resposta HTTP com um corpo de resposta 200 OK e um corpo de resposta quase vazio. Se o serviço de notificação tiver uma atualização sobre a presença de um agente ou um evento de diálogo (chamada), por exemplo, os dados serão enviados imediatamente ao cliente Web Finesse.

Para resumir:

O cliente Web Finesse tem uma conexão HTTP obsoleta (http-bind) configurada para o servidor Finesse através da porta TCP 7443. Isso é conhecido como uma votação longa BOSH. O Finesse Notification Service é um serviço de presença que publica atualizações sobre o estado de um agente, chamada, etc. Se o serviço de Notificação tiver uma atualização, ele responderá à solicitação http-bind com a atualização de estado como uma mensagem XMPP no corpo da resposta HTTP. Se não houver atualizações de estado 30 segundos após o recebimento da solicitação http-bind, o Serviço de Notificação responderá sem nenhuma atualização de estado para permitir que o cliente Web Finesse envie outra solicitação http-bind. Isso serve como uma forma do serviço de Notificação saber que o cliente Web Finesse ainda pode se conectar ao serviço de Notificação e que o agente não fechou o navegador ou colocou o computador em suspensão, e assim por diante.

CTI

Você pode usar a Integração entre Telefonia e Computador (CTI - Computer Telephony Integration) para aproveitar as funções de processamento de computador enquanto faz, recebe e gerencia chamadas telefônicas. Os aplicativos CTI permitem executar tarefas como recuperar informações do usuário de um banco de dados usando um ID de chamador ou trabalhar com as informações reunidas por um sistema de Resposta de Voz Interativa (IVR) para rotear uma chamada proveniente do usuário, juntamente com suas informações, para o representante de serviço apropriado. O Gerenciador CTI no CUCM responde às solicitações JTAPI do UCCX. A porta TCP do servidor CTI está 12018. É assim que o Finesse Server e o Engine (Servidor CTI) se comunicam.

Aqui estão algumas das informações trocadas via CTI :

- Configuração atual do sistema e atualizações futuras.

- Agentes e seus estados.

- Chamadas e seus estados.

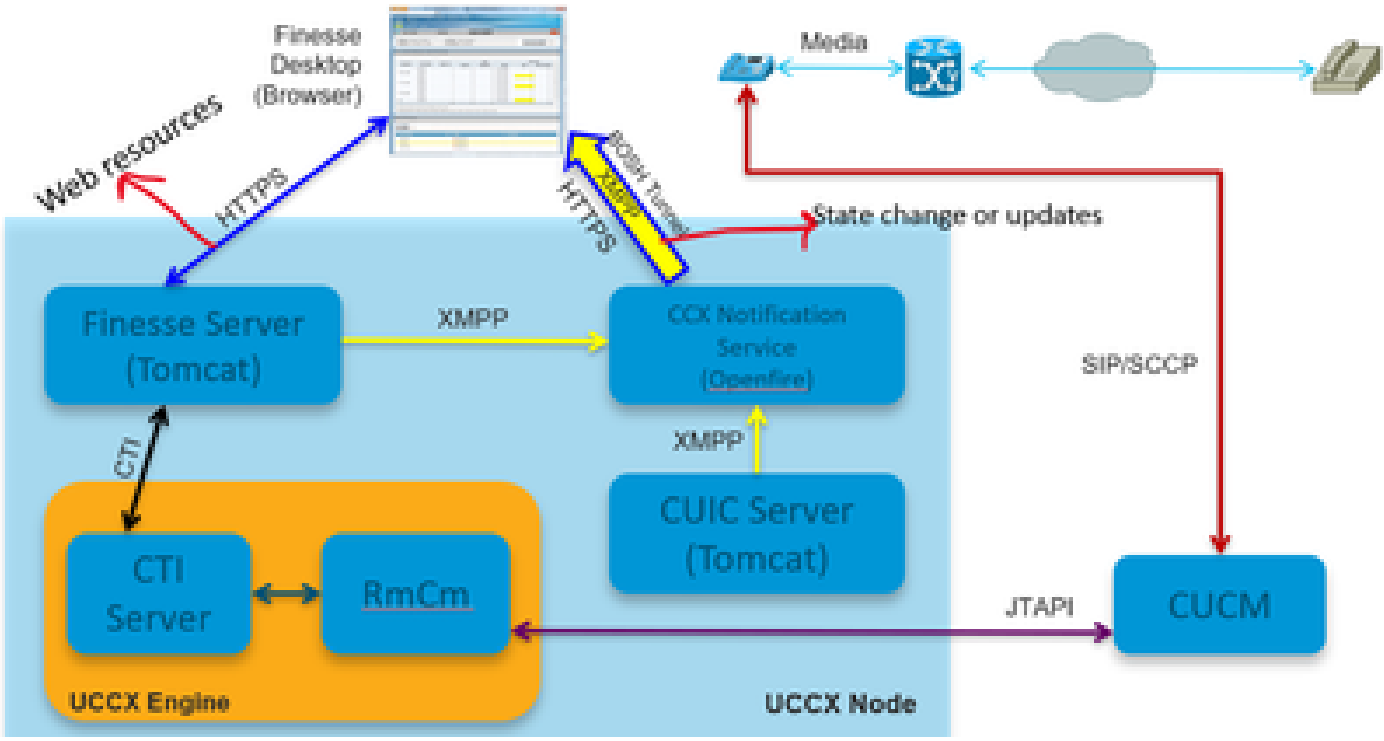
- Estatísticas de agentes, chamadas e filas em tempo real.

JTAPI

O Cisco Unified JTAPI serve como um padrão de interface de programação desenvolvido pela Sun Microsystems para uso com aplicativos de telefonia por computador baseados em Java. O Cisco JTAPI implementa a especificação Sun JTAPI 1.2 com extensões adicionais da Cisco. Qualquer comunicação entre UCCX e CUCM reside em JTAPI. É assim que o CUCM e o Engine (subsistema de telefonia) se comunicam. JTAPI é usado para controlar e monitorar telefones CUCM, rotear chamadas usando portas CTI e pontos de rota, iniciar e parar gravações no CUCM e para qualquer funcionalidade de roteamento de chamada

30000 pés de exibição

O próximo diagrama descreve como o Mecanismo UCCX, o Finesse, o CUCM e o Navegador se comunicam entre si.

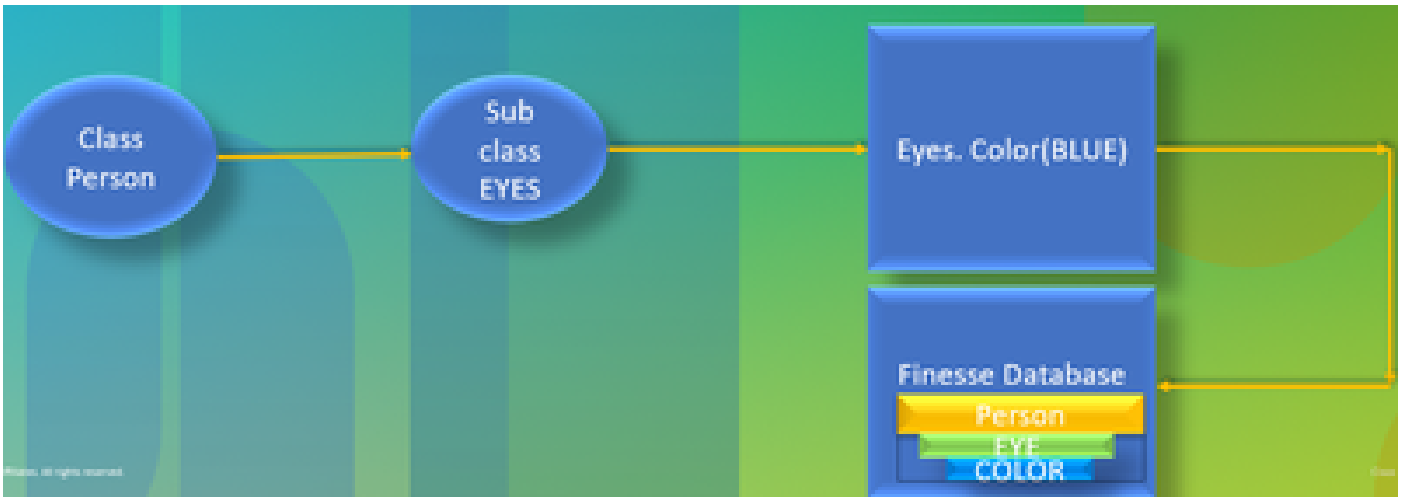


exibição 30000

Vamos considerar que a chamada seja estabelecida com o agente. Agora, o RmCm que está monitorando o ramal do agente através de JTAPI informa ao servidor CTI sobre a alteração de estado que o agente está falando. Essas informações são enviadas do servidor CTI (dentro do mecanismo CCX) para o Finesse Server (Tomcat) usando o CTI. O Finesse Server envia essas informações para o serviço de notificação do CCX usando XMPP sobre a alteração de estado. O serviço de notificação (Openfire) abre um túnel BOSH para o navegador do agente e atualiza as informações sobre a alteração de estado, e é assim que você vê o agente indo para o estado RESERVADO. Qualquer tipo de recurso da Web é solicitado para o servidor finesse usando HTTPS como arquivos WAR, gadgets e assim por diante (se já não estiver no cache).

HIBERNAR

O próximo diagrama explica sobre o serviço Hibernate.



hibernar

O HIBERNATE é conhecido como High-Performance Object/Relational Persistence and Query Service. Simplificando, ele mapeia classes JAVA para tabelas de banco de dados. Por exemplo, você tem um objeto JAVA chamado Equipe e uma tabela de banco de dados no banco de dados finesse chamado Equipe. A classe JAVA controla quais informações estão dentro da tabela e o HIBERNATE é o que faz isso acontecer. Em vez de usar consultas SQL, ele usa classes java para atualizar as informações.

AXL

XML Administrativo.

XML significa eXtensible Markup Language e é uma linguagem de marcação que define algumas regras relativamente simples para codificação de dados. Ele foi projetado principalmente para transmitir e receber dados estruturados em um formato bem definido que ambos os sistemas possam entender. Na forma mais básica, o XML define marcas que estão entre colchetes angulares (<>) e essas marcas envolvem os dados descritos pela marca. As marcas podem formar uma hierarquia com marcas dentro de outras marcas. Por exemplo, para definir um dispositivo telefônico básico, você pode dizer que um dispositivo telefônico precisa de três parâmetros, um nome, uma descrição e um número de telefone.

É uma API baseada em SOAP que permite o provisionamento remoto no CUCM. É usado para adicionar, atualizar, remover ou recuperar informações do banco de dados CUCM. Os recursos de recuperação incluem a verificação da autenticação do usuário e a execução de consultas SQL. A API AXL fornece acesso a todo o banco de dados CUCM. A API AXL é apenas para provisionamento e não fornece acesso a dados de tempo de execução ou de desempenho.

A API do AXL utiliza a Autenticação Básica HTTPS. Qualquer usuário do CUCM (aplicativo ou usuário final) terá acesso de leitura/gravação via AXL se for membro do grupo de controle de acesso **Superusuários CCM padrão** ou de qualquer grupo com a função **Acesso à API AXL padrão** atribuída a ele. Isso significa que todas as contas de superusuário implicitamente já têm acesso à API AXL. Para criar uma conta dedicada ao uso da API AXL, você deve primeiro criar um grupo de controle de acesso e atribuir a função **Acesso à API AXL Padrão** a ele, depois associar o usuário do aplicativo ao grupo recém-criado. Para fornecer acesso à API AXL somente leitura, você pode criar um Grupo de controle de acesso separado e atribuir somente a função **Acesso à API AXL somente leitura padrão** a ele.

SABÃO

O Simple Object Access Protocol (SOAP) é uma maneira de passar informações entre aplicativos em um formato XML. As mensagens SOAP são transmitidas do aplicativo emissor para o aplicativo receptor, geralmente por uma sessão HTTP. A mensagem SOAP real é composta do elemento Envelope, que contém um elemento Body e um elemento Header opcional.

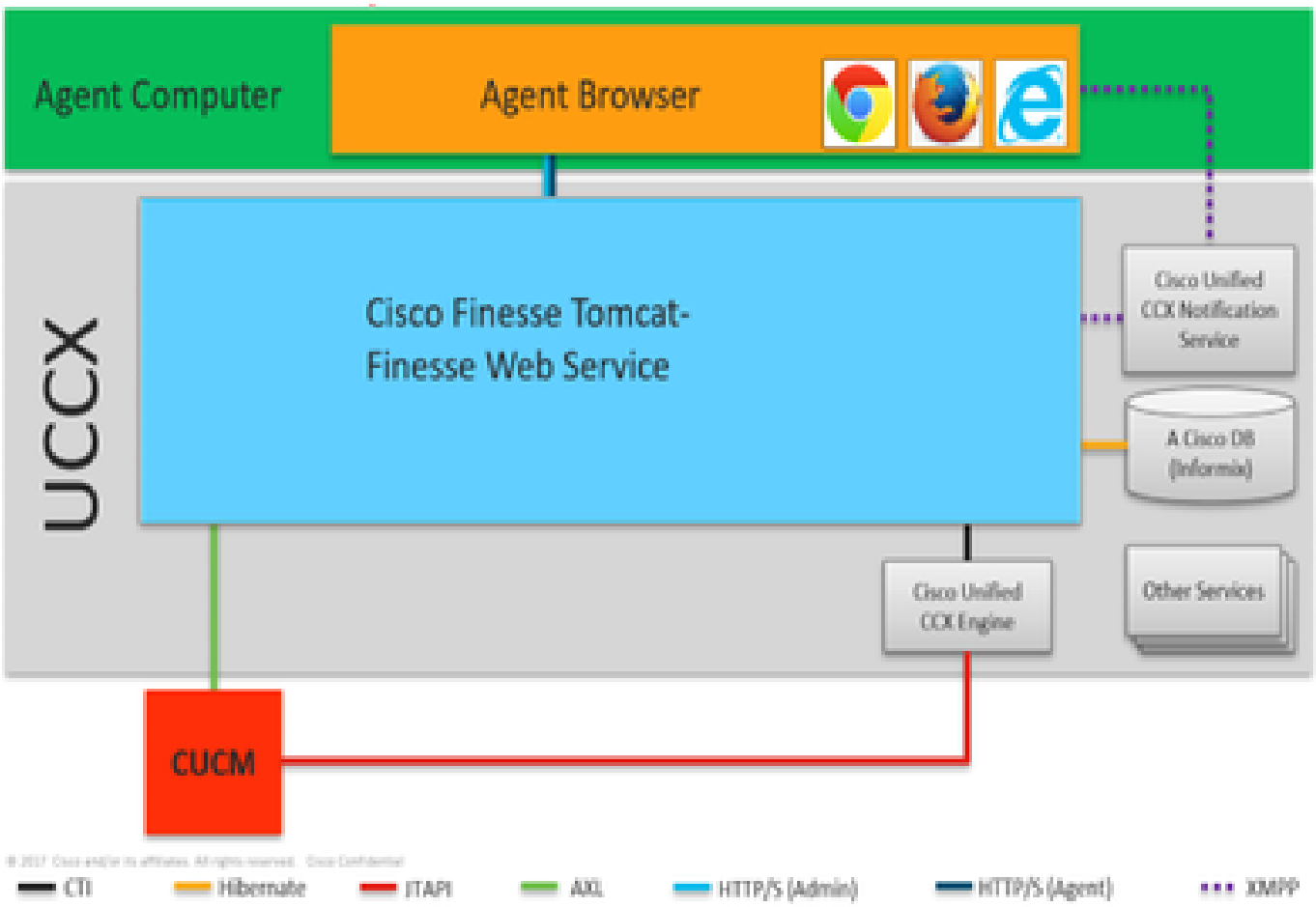
- Envelope - Este elemento obrigatório é a raiz da mensagem SOAP, identificando o XML transmitido como sendo um pacote SOAP. Um envelope contém uma seção de corpo e uma seção de cabeçalho opcional.

- Cabeçalho - Este elemento opcional fornece um mecanismo de extensão indicando informações de processamento para a mensagem. Por exemplo, se a operação que usa a mensagem exigir credenciais de segurança, essas credenciais deverão fazer parte do cabeçalho do envelope.

- Corpo - Este elemento contém o payload da mensagem, os dados brutos sendo transmitidos entre as aplicações de envio e recebimento. O próprio corpo pode consistir em vários elementos filho, com um esquema XML que normalmente define a estrutura desses dados.

20000 pés de exibição

O próximo diagrama explica de forma mais detalhada sobre os protocolos envolvidos na arquitetura Finesse.



exibição 20000

Esses são os protocolos responsáveis pela comunicação entre diferentes componentes do UCCX.

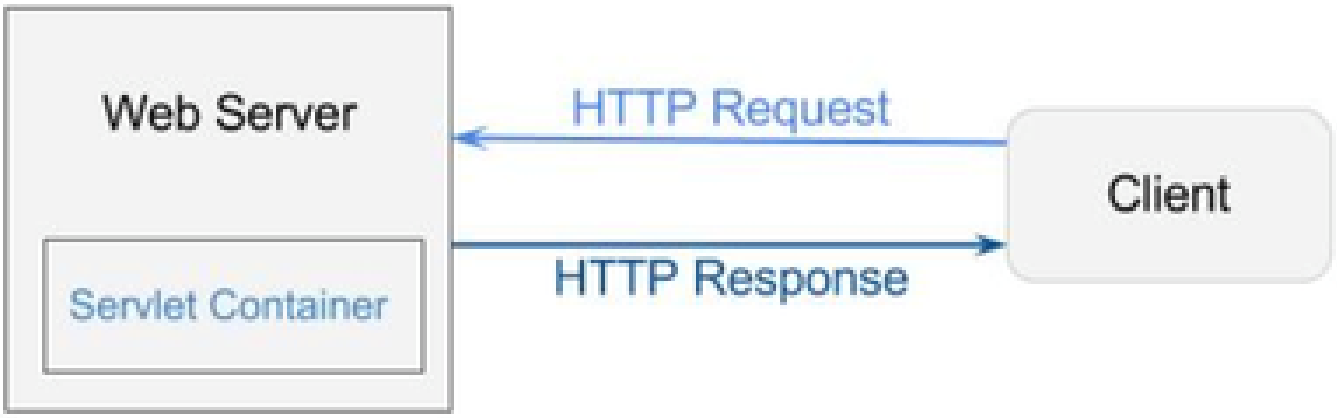
- O mecanismo UCCX e o Finesse Server falam sobre o protocolo CTI.

- Mecanismo UCCX e CUCM conversam sobre o protocolo JTAPI.

- Finesse Tomcat e CUCM conversam sobre AXL.

- O serviço Finesse Notification e o navegador do agente conversam em XMPP/BOSH.
- Finesse Tomcat e o banco de dados conversam sobre o Hibernate.
- Finesse Tomcat e Finesse Notification falam sobre XMPP.

APACHE SHINDIG



Desaparecimento

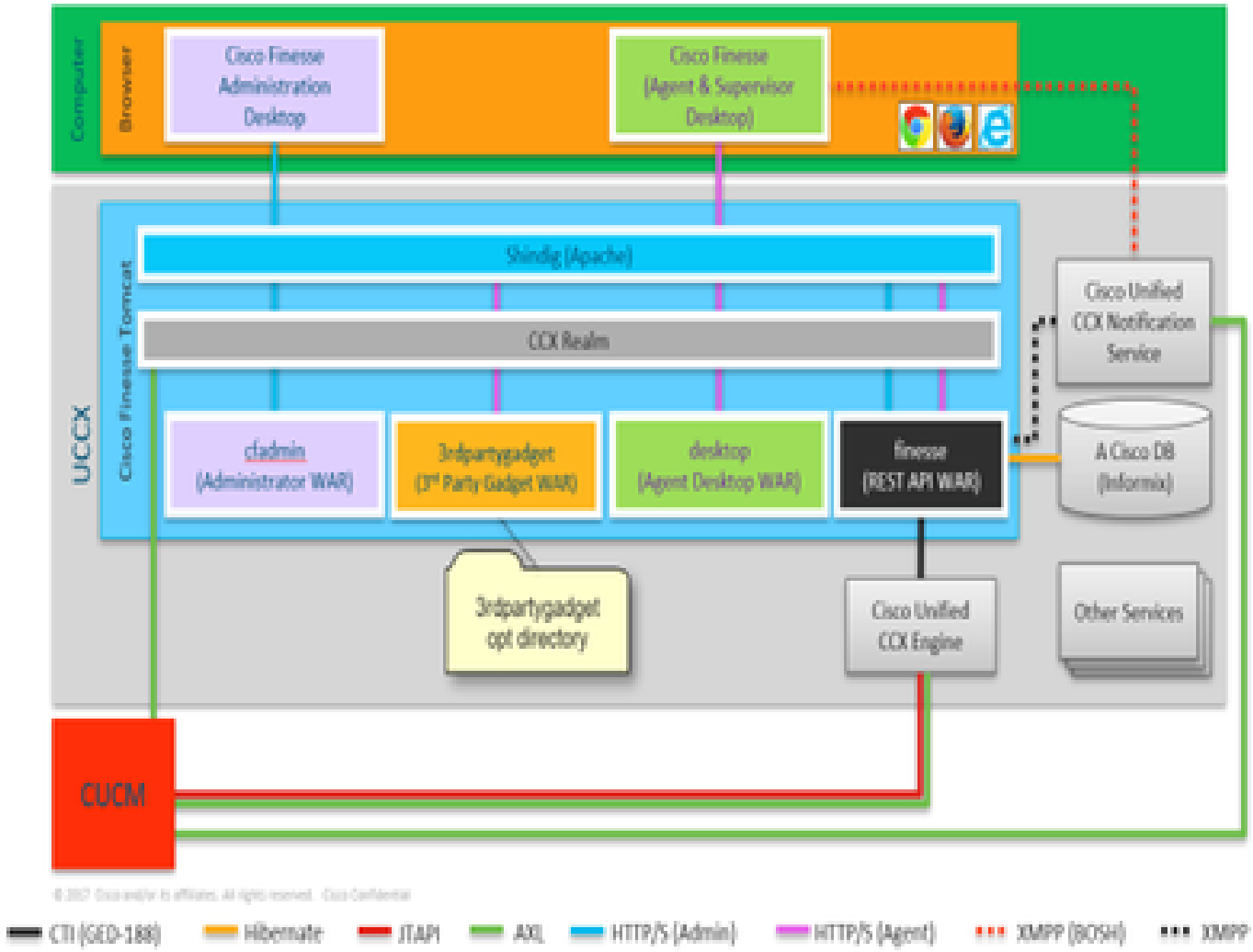
Apache Shindig é um contêiner do OpenSocial e ajuda você a começar a hospedar aplicativos OpenSocial rapidamente, fornecendo o código para renderizar gadgets, solicitações de proxy e lidar com solicitações REST e RPC. O OpenSocial é um conjunto de APIs para criar aplicativos sociais que são executados na Web. (Web/Servlet) O contêiner é usado por um servidor Web para gerar dinamicamente páginas da Web.

ARQUIVOS WAR

WAR significa Web Archive (Arquivo Web). Contém arquivos de um projeto da Web. Ele pode ter servlet, XML, JSP, imagem, HTML, CSS, JS e assim por diante. Os logs do Catalina contêm as informações sobre a implantação de WARs.

10000 pés de exibição

O próximo diagrama explica em detalhes como o fluxo de autenticação funciona dentro dos componentes do UCCX e do Finesse.



exibição 10000

Os arquivos WAR são necessários para exibir e criar a página, dependendo de como você se conecta. O navegador pergunta ao Shindig que ele precisa renderizar um gadget, o Shindig conversa com o CUIC para renderizar o gadget. O território CCX é usado para autenticação com CUCM usando AXL. O serviço de notificação também se autentica com o CUCM usando o AXL.

Finesse Rest API WAR é o repositório principal que realmente se comunica com o serviço de notificação, CCX Engine e DB. Shindig fala apenas com Finesse Rest API (WebServices), porque cfadmin e desktop WARs são apenas para exibir a página. Qualquer coisa que venha para o Finesse Rest API WAR, você pode ver isso nos registros do Finesse WebServices que são os registros mais importantes para o finesse. Você fala HTTP entre o serviço Web Shindig e Finesse (Rest API WAR). O serviço da Web Finesse (Rest API WAR) e o mecanismo conversam entre si via CTI.

AJAX - A beleza de Finesse

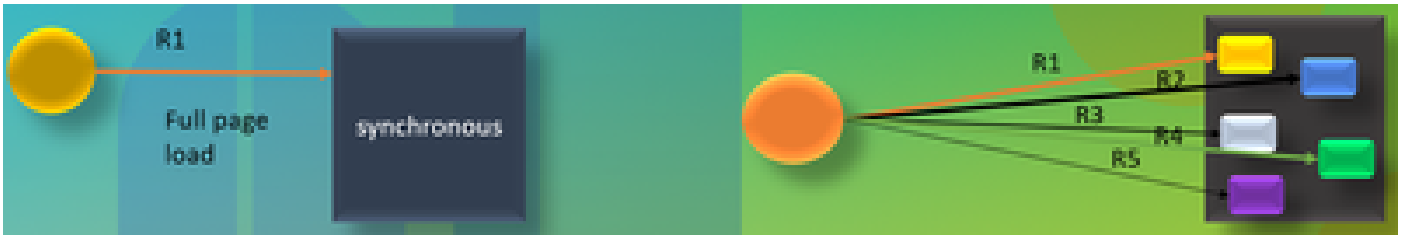
AJAX significa Asynchronous Javascript and XML. Não é uma linguagem de programação, mas um método para acessar servidores Web a partir de uma página da Web. O AJAX é um mecanismo para fazer atualizações parciais de página. Ele permite atualizar seções de uma página com dados que vêm de um servidor sem a necessidade de atualizar a página inteira.

Por exemplo, se você falar sobre o Facebook Messenger, quando uma nova mensagem chega, você não precisa atualizar a página inteira para obter a mensagem, em vez disso, a seção de mensagem da própria página é atualizada e recebe as novas mensagens em tempo real, sem a necessidade de atualizar a página inteira.

Cada navegador tem um objeto interno chamado XMLHTTPREQUEST (também chamado de **XHR**). Cada solicitação ao AJAX no servidor passa por essa solicitação XML. Ele contém as especificações do que você precisa atualizar.

Vantagens do uso do AJAX

O próximo diagrama explica a diferença entre as solicitações assíncronas e síncronas.



AJAX

No caso de uma solicitação síncrona, você tem que esperar que a primeira solicitação seja processada e então pode enviar a segunda solicitação. Por exemplo, a atualização de página é necessária e você não pode fazer nada até que a página seja atualizada. Por outro lado, no caso de uma solicitação assíncrona, você não precisa esperar que a primeira solicitação seja concluída para enviar a segunda solicitação. Você pode enviar várias solicitações simultaneamente. Por exemplo, gadgets de aplicativos meteorológicos em sites. Você pode atualizar a seção de clima da página e, enquanto isso, também trabalhar nas outras seções do site simultaneamente, sem a necessidade de atualizar a página inteira. Essa é a principal vantagem da solicitação assíncrona.

TRABALHO DO AJAX

O AJAX é uma combinação de um XMLHTTPREQUEST (**XHR**) que é usado para enviar e receber atualizações do servidor Web, juntamente com Javascript e HTML, que são usados para exibir ou usar os dados.

ENVIANDO SOLICITAÇÃO COM AJAX PARA O SERVIDOR

Este é um processo de 3 etapas que é mencionado a seguir:

1. Criando uma variável e armazenando o objeto **XHR** nela.

Solicitação Var = new XMLHttpRequest();

2. Acessando a variável request que tem o payload dentro do objeto XHR.

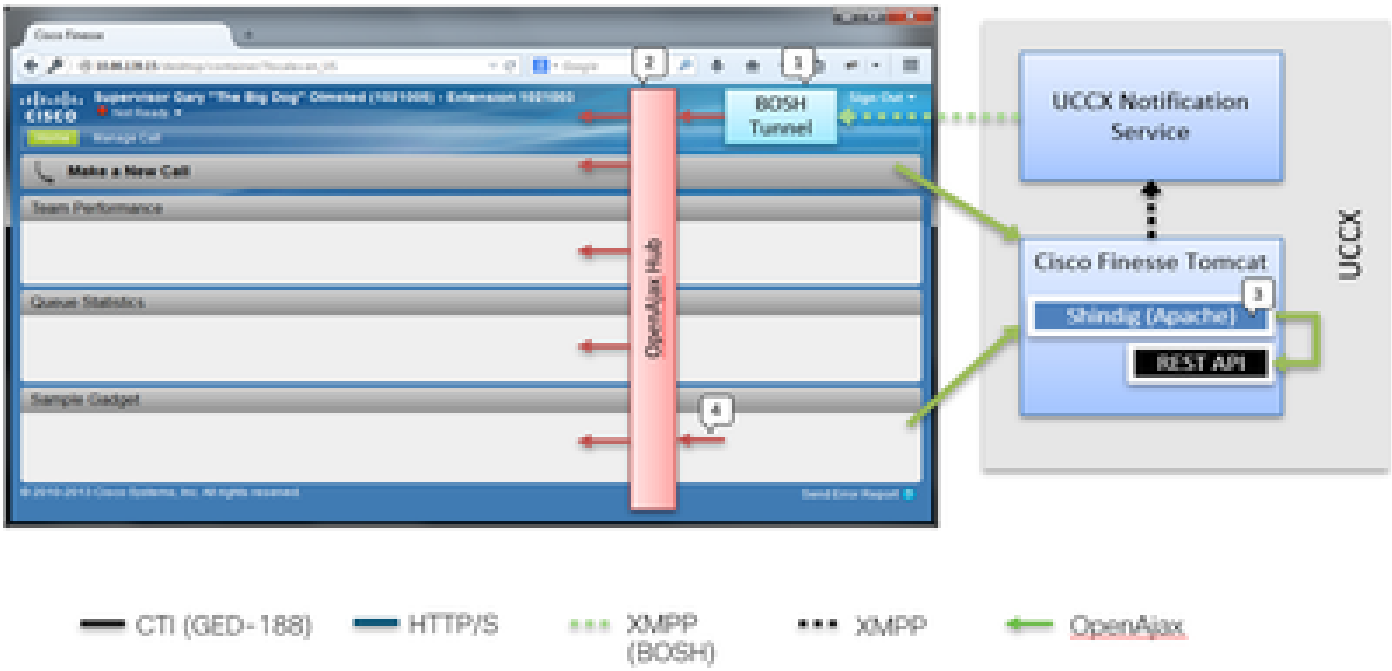
requisição;.**open(GET, URL)**

3. Enviando a solicitação

Request.send() ;

Arquitetura de desktop

O próximo diagrama explica o fluxo de sinais AJAX quando o gadget é renderizado na página da Web.

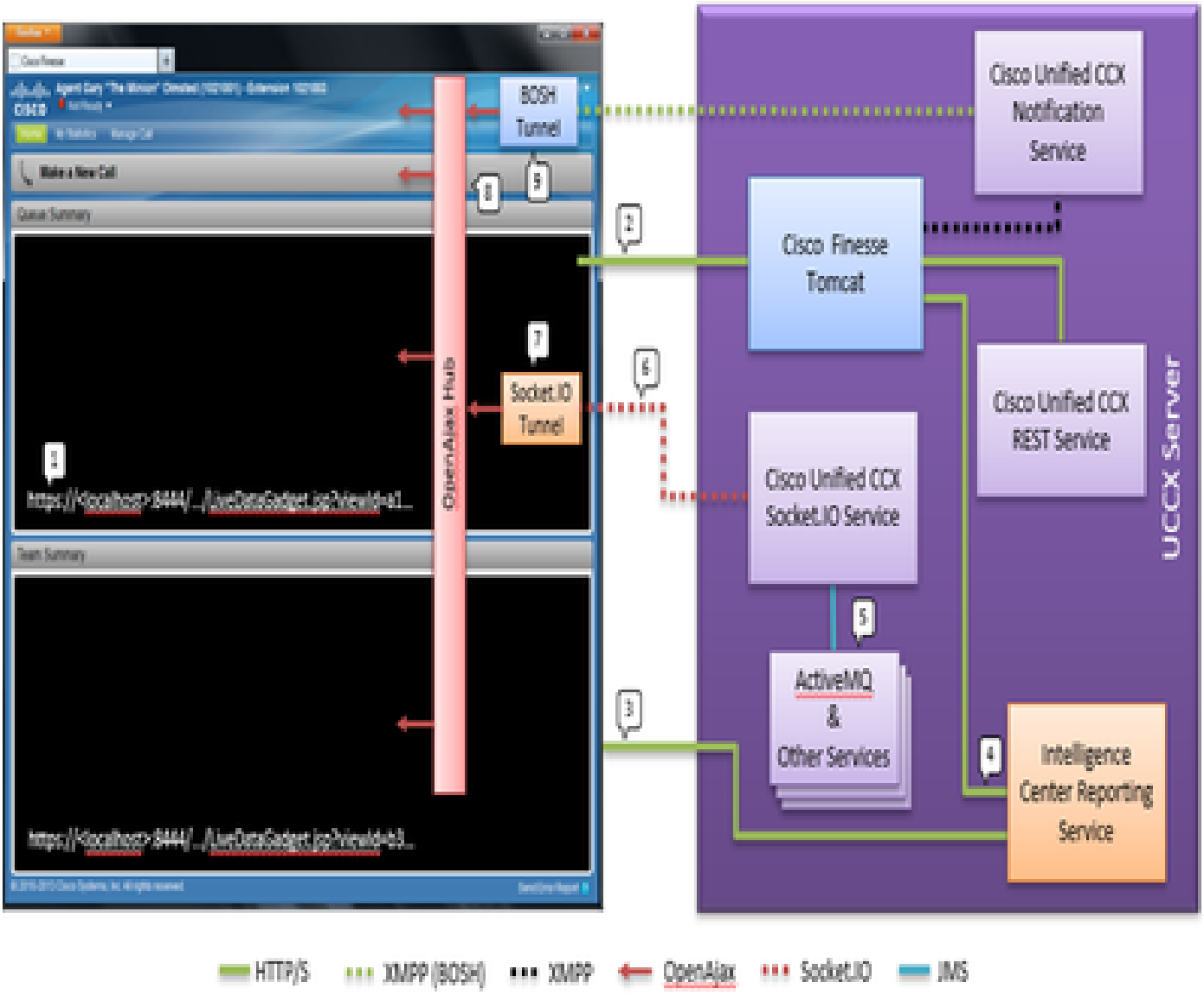


arquitetura de desktop

O IFrame reside no contêiner para hospedar o Túnel BOSH. O hub OPENAJAX é fornecido para publicar mensagens nos gadgets (usando o método pubsub). As solicitações REST são encaminhadas por proxy por meio do Shindig para outros servidores também. Os gadgets podem publicar suas próprias mensagens no hub AJAX.

Arquitetura de gadget

O próximo diagrama explica a arquitetura do gadget Finesse em detalhes.



arquitetura de gadget

Ao contrário dos gadgets típicos, os gadgets CUIC também recebem um feed XMPP em tempo real do OpenFire. No caso do UCCX, onde o CUIC e o Finesse são co-residentes com o UCCX, há uma instância compartilhada do OpenFire. A maior parte do conteúdo do gadget e todas as APIs REST são enviadas por proxy através do Shindig no Finesse Server. Isso vale para gadgets Finesse e API REST, bem como instâncias de gadget CUIC e API REST. Os gadgets CUIC usam uma D-Grid para renderizar seus relatórios. Há um processo de bootstrapping que deve ocorrer e isso é feito em conjunto com o CUIC diretamente. Por esse motivo, os gadgets CUIC inicialmente se comunicam com o servidor CUIC diretamente durante o processo de carregamento. Por esse motivo, o certificado CUIC deve ser aceito no navegador do usuário (além do túnel Socket.IO). O conteúdo do gadget e as APIs REST são enviadas por proxy para o cliente entre o Finesse e o CUIC. As chamadas à API Rest são feitas para o Intelligence Center Reporting Service e para o CCX Web Service. O Serviço CCX Live Data Socket.IO obtém as mensagens do Live Data via JMS do AtiveMQ. O Serviço Socket.IO do CCX Live Data publica o JSON de Relatórios em Tempo Real na conexão Socket.IO do cliente. Semelhante à maneira como o Finesse Desktop tem um iFrame de túnel BOSH que mantém a conexão BOSH com o Serviço de Notificação Cisco Finesse, o gadget de dados ao vivo mestre tem um iFrame de túnel Socket.IO que mantém a conexão Socket.IO (websocket) com o serviço CCX Live Data Socket.IO.

O Hub OpenAjax distribui todos os eventos para os ouvintes inscritos. Isso seria tanto Gadgets quanto partes do próprio Contêiner Finesse. O Finesse Desktop possui um iFrame de túnel BOSH que mantém a conexão BOSH com o serviço de notificação do Cisco Unified CCX. Isso publica eventos no Hub OpenAjax.

Links de Referência

- Guia do Desenvolvedor de Serviços Web Finesse

Sobre esta tradução
A Cisco traduziu este documento com a ajuda de tecnologias de tradução automática e humana para oferecer conteúdo de suporte aos seus usuários no seu próprio idioma, independentemente da localização.

Observe que mesmo a melhor tradução automática não será tão precisa quanto as realizadas por um tradutor profissional.

A Cisco Systems, Inc. não se responsabiliza pela precisão destas traduções e recomenda que o documento original em inglês (link fornecido) seja sempre consultado.