UCCX Finesse 아키텍처 심층 분석

목차

소개

이 문서에서는 Finesse 문제를 트러블슈팅하는 동안 기본 프로세스가 적절하도록 Finesse 아키텍처에 대해 자세히 설명합니다.

사전 요구 사항

요구 사항

Cisco에서는 다음 툴 및 기능에 대한 지식을 권장합니다.

JTAPI - Java 텔레포니 API

API - 애플리케이션 프로그래밍 인터페이스

UCCX - Unified Contact Center Express

CUCM - Cisco Unified Communications Manager

CTI - 컴퓨터 텔레포니 통합

사용되는 구성 요소

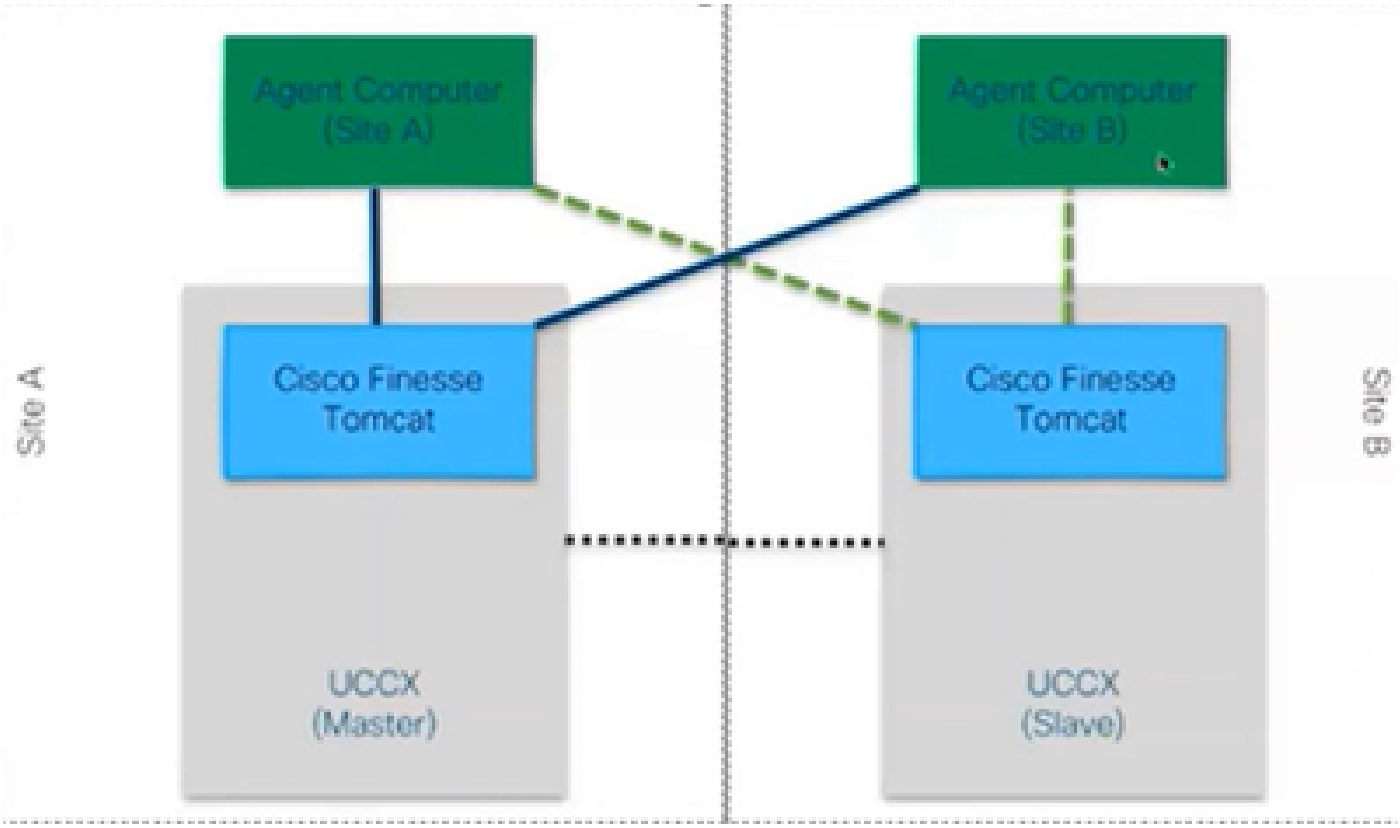
- Cisco UCCX(Unified Contact Center Express)

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

배경 정보

이 문서에서는 Finesse 아키텍처를 개괄적으로 설명한 다음 예시 및 다이어그램과 함께 심층적인 신호 흐름을 설명합니다.

50,000피트 뷰



50000 보기

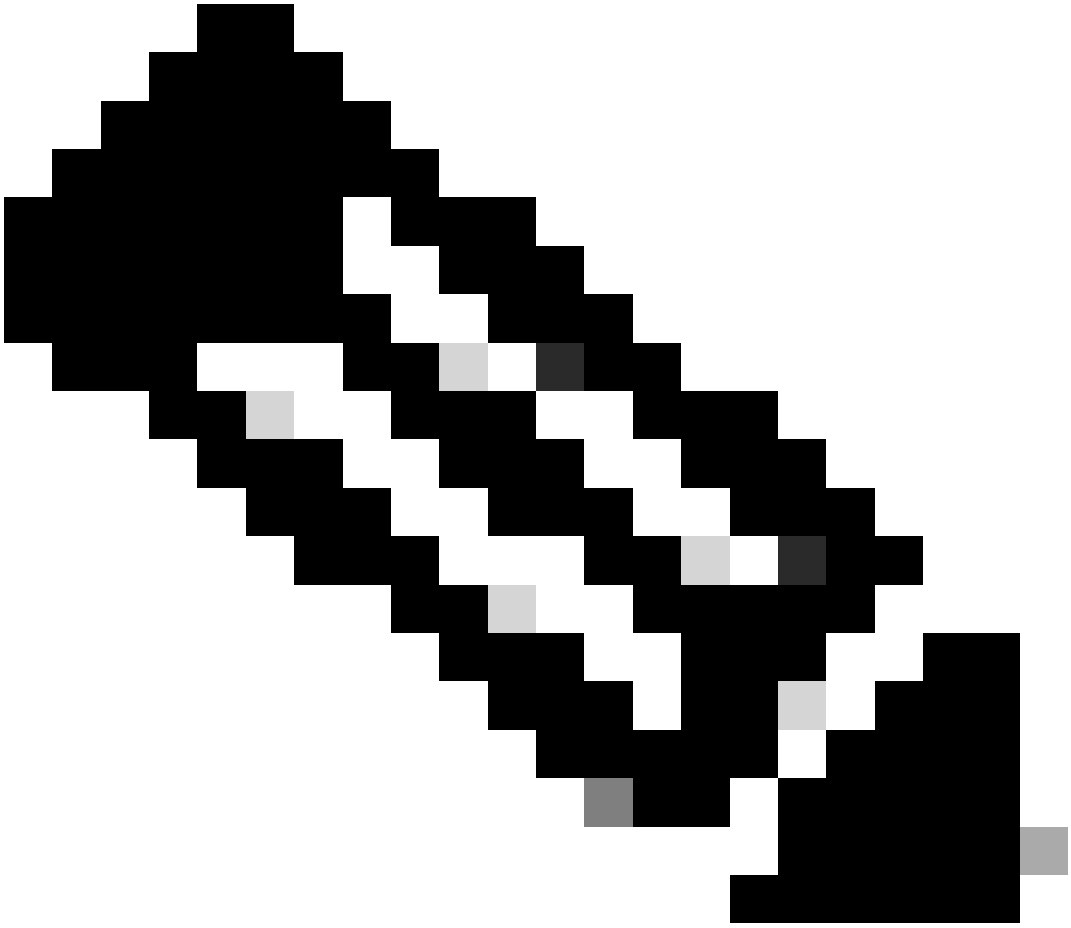
파인세 톰캣

Finesse Tomcat은 CUCM의 Cisco Tomcat과 유사합니다. 웹 페이지를 로드하는 기능은 동일하지만 Finesse라는 다른 서비스에 사용됩니다. Tomcat은 별도의 웹 애플리케이션이므로 Finesse용으로 설계되었습니다. 11.5 이전 버전에 따라 CCX 마스터 노드를 통해서만 finesse에 로그인할 수 있습니다. 11.6 이상에서는 장애 조치 중에만 모든 노드에 로그인할 수 있습니다(권장되지는 않음). 따라서 두 에이전트(사이트 A 및 사이트 B)가 모두 마스터 노드의 Finesse에 연결되는 WAN 클러스터를 고려할 경우, 항상 Cisco Finesse에서 엔진으로의 다른 노드로의 비활성 연결이 존재하는데, 이는 장애 조치에 필요한 CTI openconf 요청입니다.

Openconf 요청은 노드가 활성 또는 대기 상태인지 확인하기 위해 정기적으로 전송됩니다. 장애 조치가 있을 경우 알림 서비스는 이 노드가 다운되어 장애 조치가 필요함을 클라이언트에 알리는 systeminfo API라는 API를 사용합니다. 그런 다음 파일을 실행하여 브라우저를 다른 노드로 리디렉션한 다음 openconf rejavascriptsponse가 전송됩니다. 이 장애 조치는 인증서가 수락된 경우에만 작동합니다. (서버에 대한 보안 연결을 설정합니다.)

다음과 같은 3개의 인증서가 표시됩니다.

- 해당 노드에 대한 알림 서비스 인증서
- 원격 노드에 대한 알림 서비스 인증서
- 원격 노드에 대한 Finesse 서비스 인증서



참고: 장애 조치가 작동하려면 원격 노드 인증서를 수락해야 합니다.

HTTP(s)

HTML과 같은 하이퍼미디어 문서를 전송하기 위한 애플리케이션 레이어 프로토콜입니다. 웹 브라우저와 웹 서버 간의 통신을 위해 설계되었습니다.이 프로토콜은 스테이트리스 프로토콜로서 서버가 두 요청 간에 데이터를 유지하지 않음을 의미합니다. 클라이언트 서버 프로토콜입니다. UCCX의 경우 Finesse 클라이언트는 에이전트 브라우저(PC)에서 실행됩니다. HTTP를 사용하여 서버에 요청합니다. 다음은 몇 가지 일반적인 HTTP 방법입니다.

GET - 서버에서 정보를 가져옵니다.
POST -

서버에 정보를 전송합니다.
PUT - 서버의 모든 항목을 교체합니다.
DELETE - 서버에서 정보를 제거합니다.

Finesse는 http 요청 내에서 systeminfo api 요청을 사용합니다. 예를 들어 상담원의 상태를 변경하려면 브라우저에서 POST 대신 PUT를 보냅니다. POST를 보내면 서버에 두 가지 상태가 있으므로 혼동되기 때문입니다. 어떤 것을 선택해야 합니까? 그래서 PUT를 사용하면 현재의 상태를 대체합니다.

XMPP

확장 가능한 메시징 및 프레즌스 프로토콜

인스턴트 메시징 및 프레즌스에 사용되는 프로토콜 집합입니다. 모든 XMPP 엔터티는 해당 Jabber IDS 또는 JID를 사용하여 식별됩니다. 가젯에 사용되는 이 XMPP 프로토콜의 확장 중 하나를 PUBSUB라고 합니다.

펍서브

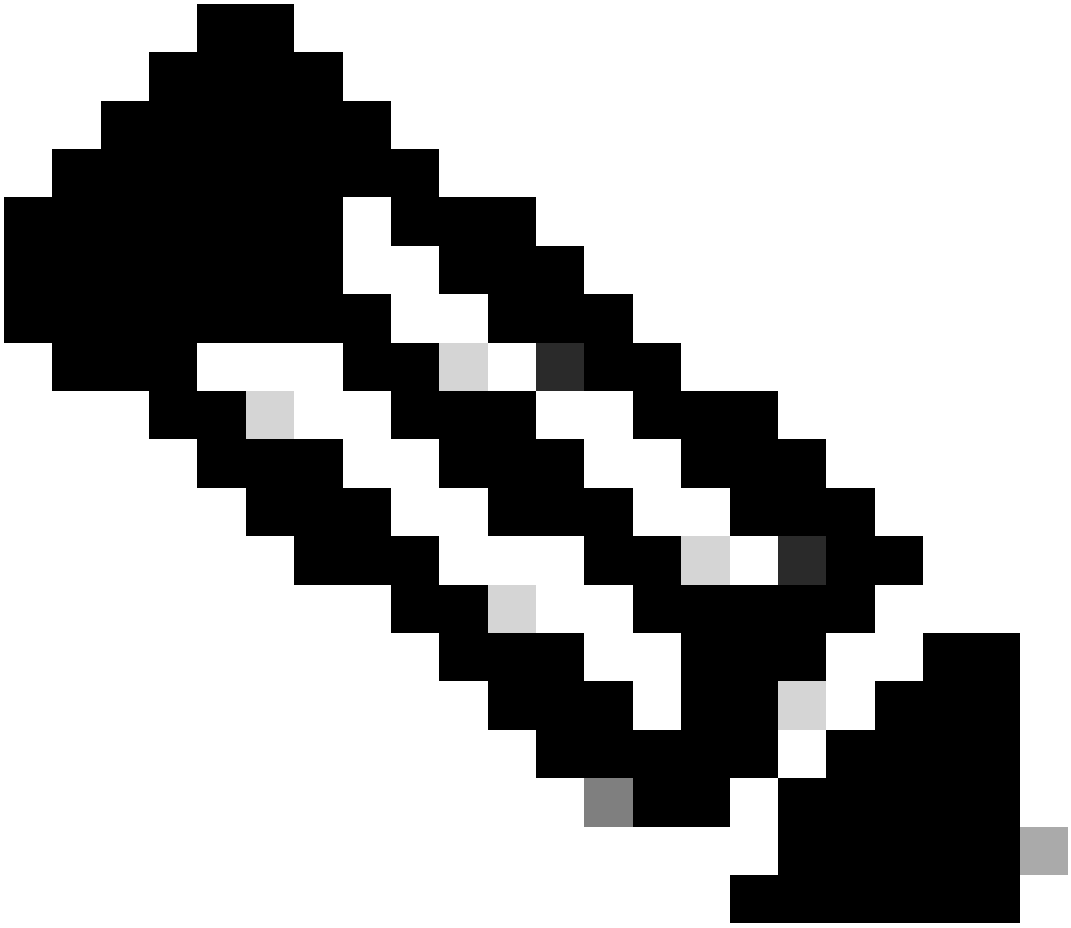
당신이 생각할 수 있는 게시자 구독자가 아닙니다. 여전히 게시하고 구독하고 있지만 데이터베이스와는 관련이 없습니다. XMPP는 노드라는 메커니즘을 사용합니다. 노드는 기본적으로 사용자가 관심을 갖는 엔티티를 모니터링합니다. 당신에게 중요하고 그것을 모니터링 하고 싶은 모든, 당신은 그것에 노드를 추가합니다. 그런 다음 PUBSUB는 해당 노드에서 업데이트 또는 알림을 구독합니다. 에이전트, 대화 상자 등 각 엔터티 유형에 대한 노드를 가질 수 있습니다. 에이전트의 노드를 생성하면 해당 에이전트의 노드에 가입된 다음 어떤 작업을 수행하든 그에 대한 알림을 받게 됩니다.

이 사양의 목적은 XMPP 서버(알림 서비스)가 XMPP 노드(항목)에 게시된 정보를 가져온 다음 해당 노드에 가입된 엔터티에 XMPP 이벤트를 보낼 수 있도록 하는 것입니다.

Finesse의 경우 CTI(Computer Telephony Integration) 서버는 CTI 메시지를 Finesse 웹 서비스에 전송하여 에이전트 또는 CSQ(Contact Service Queue) 생성 또는 통화 정보와 같은 컨피그레이션 업데이트에 대해 Finesse에게 알립니다. 그런 다음 이 정보는 Finesse 웹 서비스가 Finesse Notification 서비스에 게시하는 XMPP 메시지로 변환됩니다. 그러면 Finesse Notification 서비스는 특정 XMPP 노드에 가입된 에이전트에게 BOSH 메시지를 통해 XMPP를 전송합니다.

BOSH - 동기식 HTTP를 통한 양방향 스트림

BOSH는 서버에서 요청에 대한 응답이 있을 때까지 더 오랜 시간 동안 요청을 보관하고 그렇지 않으면 빈 응답을 보내는 HTTP 연결입니다. 이는 XMPP 클라이언트 및 XMPP 서버에서 작동하지만 비 XMPP 애플리케이션에도 사용할 수 있습니다.



참고: XMPP는 상태 저장 형식인 반면 HTTP는 상태 비저장 형식입니다(마지막 요청에 대한 정보는 저장하지 않음).

웹 애플리케이션이 XMPP와 연동해야 하는 경우 여러 문제가 발생합니다.

문제 1: 브라우저에서 기본적으로 XMPP over TCP(Transmission Control Protocol)를 지원하지 않습니다.

해결 방법 1: 웹 서버와 브라우저는 HTTP(HyperText Transfer Protocol) 메시지를 통해 통신하므로 Finesse 및 기타 웹 애플리케이션은 XMPP 메시지를 HTTP 메시지 내에 래핑합니다.

문제 2: HTTP는 스테이트리스 프로토콜입니다.

해결 방법 2: 쿠키/게시 데이터를 사용할 수 있습니다.

문제 3: 세 번째 문제는 클라이언트만 요청을 전송하고 서버만 응답할 수 있는 HTTP의 단방향 동작입니다. 서버에서 데이터를 푸시할 수 없으므로 HTTP를 통해 XMPP를 구현하는 것이 부자연스럽습니다.

해결 방법 3: 이 문제를 해결하려면 HTTP와 XMPP 간에 브리지가 필요합니다.

제안 솔루션은 다음과 같습니다.

- 폴링(레거시 프로토콜): 정의된 새 데이터를 요청하는 반복된 HTTP 요청: HTTP 폴링

- 긴 폴링은 BOSH: 전송 프로토콜이라고도 합니다. 이 전송 프로토콜은 자주 폴링을 사용하지 않고도 여러 동기 HTTP 요청/응답 쌍을 효율적으로 사용하여 두 엔터티 간의 긴 수명의 양방향 TCP 연결을 에뮬레이션합니다. BOSH를 사용하는 이유는 서버가 요청이 있는 즉시 대응할 필요가 없다는 사실을 은폐하기 위해서다. 서버에 클라이언트에 대한 데이터가 있을 때까지 응답이 지정된 시간까지 지연된 다음 응답으로서 전송됩니다. 고객이 응답을 받는 즉시 고객은 새로운 요청 등을 합니다.

Finesse 데스크톱 클라이언트(웹 애플리케이션)는 30초마다 TCP 포트 7443을 통해 오래된 BOSH 연결을 설정합니다. 30초 후에 Finesse Notification Service에서 업데이트가 없는 경우 알림 서비스는 200 OK 및 (거의) 빈 응답 본문이 포함된 HTTP 응답을 보냅니다. 알림 서비스에 에이전트 또는 대화(통화) 이벤트 존재 여부에 대한 업데이트가 있는 경우 데이터는 Finesse 웹 클라이언트로 즉시 전송됩니다.

요약하자면,

Finesse 웹 클라이언트에는 TCP 포트 7443을 통해 Finesse 서버에 대한 부실 HTTP 연결(http-bind)이 설정되어 있습니다. 이 설문조사는 BOSH의 긴 설문조사라고 합니다. Finesse 알림 서비스는 상담원 상태, 통화 등에 대한 업데이트를 게시하는 프레즌스 서비스입니다. 알림 서비스에 업데이트가 있는 경우 HTTP 응답 본문에 XMPP 메시지로 상태 업데이트를 사용하여 http-bind 요청에 응답합니다. http-bind 요청을 받은 후 30초 후에 상태 업데이트가 없는 경우 알림 서비스는 상태 업데이트 없이 회신하여 Finesse 웹 클라이언트가 다른 http-bind 요청을 보낼 수 있도록 합니다. 이는 알림 서비스가 Finesse 웹 클라이언트가 여전히 알림 서비스에 연결할 수 있으며 에이전트가 브라우저를 닫거나 컴퓨터를 절전 모드로 전환하지 않았음을 알리는 등의 역할을 합니다.

CTI

CTI(Computer Telephony Integration)를 사용하여 전화 통화를 걸고 받고 관리하는 동안 컴퓨터 처리 기능을 활용할 수 있습니다. CTI 애플리케이션을 사용하면 발신자 ID를 사용하여 데이터베이스에서 사용자 정보를 검색하는 등의 작업을 수행하거나 IVR(Interactive Voice Response) 시스템에서 수집한 정보를 사용하여 사용자가 발신한 통화와 해당 정보를 적절한 서비스 담당자에게 라우팅할 수 있습니다. CUCM의 CTI Manager가 UCCX의 JTAPI 요청에 응답합니다. CTI 서버 TCP 포트가 12018. 이는 Finesse 서버와 엔진(CTI 서버)이 서로 통신하는 방법입니다.

다음은 CTI를 통해 교환된 정보입니다.

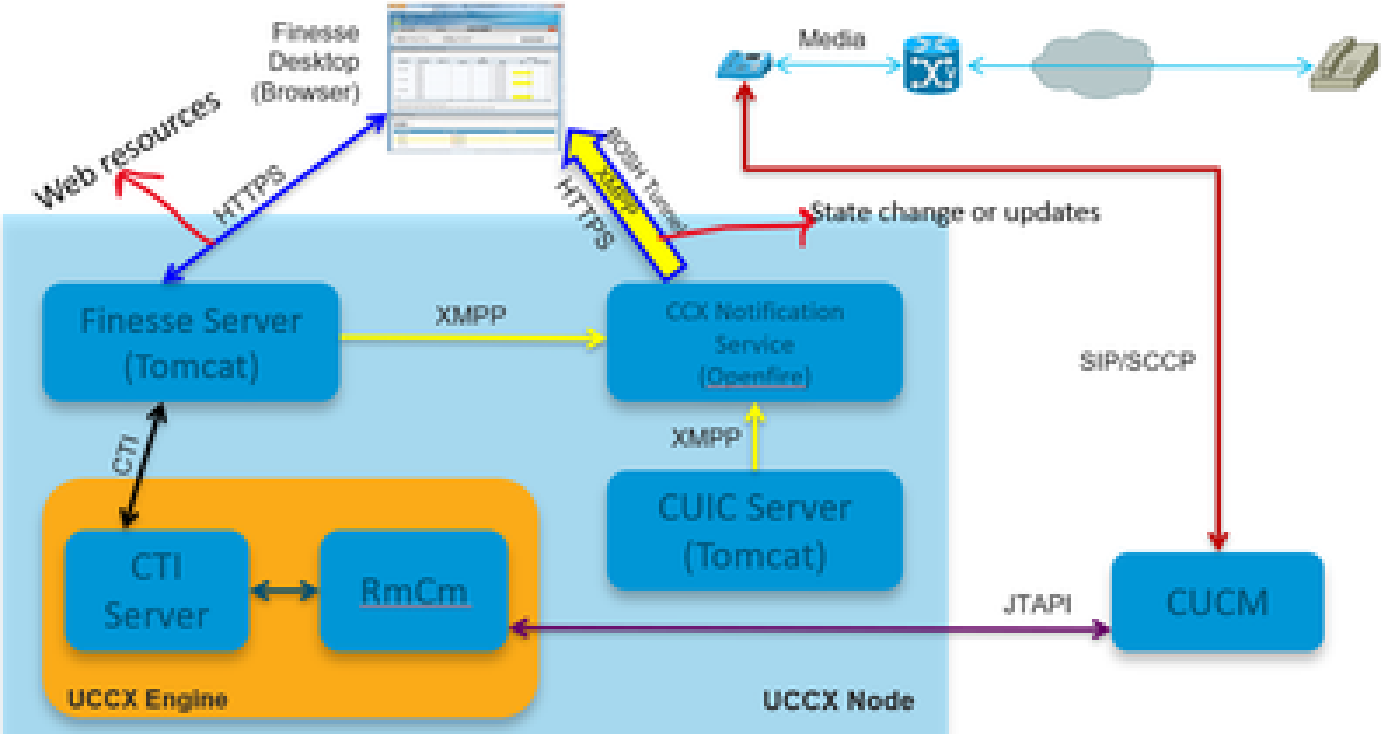
- 현재 시스템 구성 및 향후 업데이트
- 상담원 및 해당 상태
- 통화 및 해당 상태.
- 실시간으로 상담원, 통화 및 대기열에 대한 통계

JTAPI

Cisco Unified JTAPI는 Sun Microsystems에서 Java 기반 컴퓨터 전화 응용 프로그램과 함께 사용하기 위해 개발한 프로그래밍 인터페이스 표준으로 사용됩니다. Cisco JTAPI는 추가 Cisco 확장과 함께 Sun JTAPI 1.2 사양을 구현합니다. UCCX와 CUCM 간의 모든 통신은 JTAPI에 있습니다. 이는 CUCM과 엔진(텔레포니 하위 시스템)이 서로 통신하는 방법입니다. JTAPI는 CUCM 전화기를 제어 및 모니터링하고, CTI 포트 및 경로 포인트를 사용하여 통화를 라우팅하고, CUCM에서 녹음을 시작 및 중지하고, 모든 통화 라우팅 기능을 지원하는 데 사용됩니다

30000피트 뷰

다음 다이어그램에서는 UCCX 엔진, Finesse, CUCM 및 브라우저가 서로 통신하는 방법을 설명합니다.

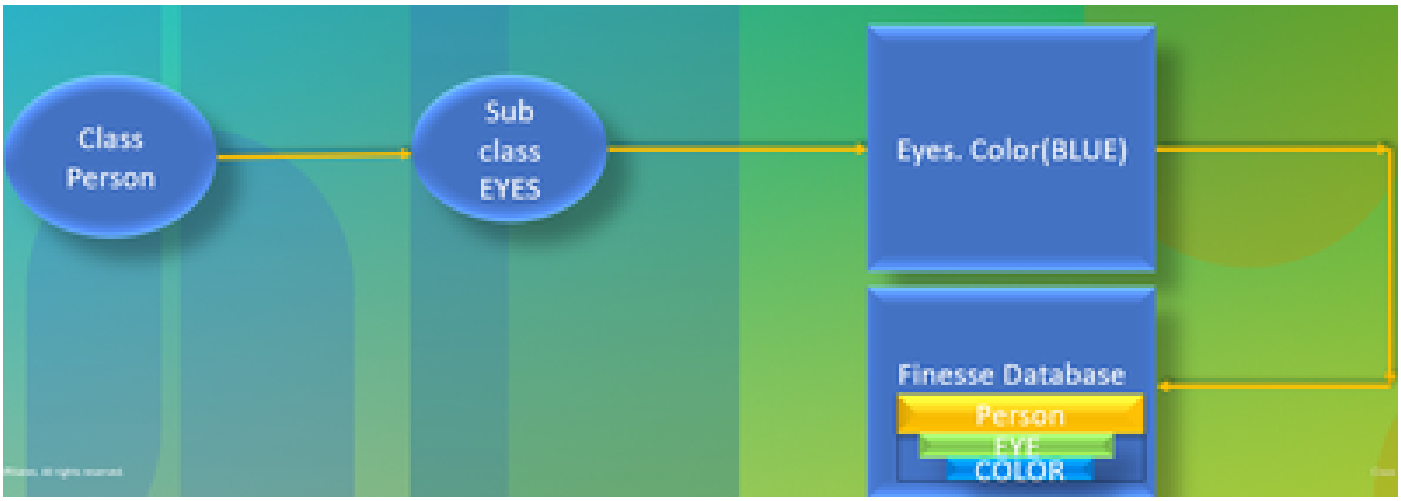


30000 보기

상담원과 통화가 설정된 것으로 가정합니다. 이제 JTAPI를 통해 에이전트 확장을 모니터링하는 RmCm이 CTI 서버에 에이전트가 통화 중인 상태 변경에 대해 알려줍니다. 이 정보는 CTI를 사용하여 CTI 서버(CCX 엔진 내부)에서 Finesse 서버(Tomcat)로 전송됩니다. Finesse 서버는 상태 변경에 대해 XMPP를 사용하여 이 정보를 CCX 알림 서비스에 보냅니다. 알림 서비스(Openfire)는 상담원 브라우저에 BOSH 터널을 열고 상태 변경 및 상담원이 RESERVED(예약됨) 상태로 전환되는 것에 대한 정보를 업데이트합니다. 모든 유형의 웹 리소스는 WAR 파일, 가젯 등과 같은 HTTPS를 사용하여 Finesse 서버에 요청됩니다(캐시에 없는 경우).

최대 절전 모드

다음 다이어그램에서는 최대 절전 모드 서비스에 대해 설명합니다.



최대 절전 모드

HIBERNATE를 고성능 개체/관계형 지속성 및 쿼리 서비스라고 합니다. 간단히 말해, JAVA 클래스를 데이터베이스 테이블에 매핑합니다. 예를 들어 Team이라는 JAVA 객체가 있고 Team이라는 finesse 데이터베이스에 데이터베이스 테이블이 있습니다. JAVA 클래스는 테이블 내의 정보를 제어하며 HIBERNATE는 이를 가능하게 하는 요소입니다. SQL 쿼리를 사용하는 대신 java 클래스를 사용하여 정보를 업데이트합니다.

축

관리 XML입니다.

XML은 eXtensible Markup Language를 의미하며 데이터 인코딩에 대한 비교적 간단한 몇 가지 규칙을 정의하는 마크업 언어입니다. 이는 주로 양 시스템이 이해할 수 있는 잘 정의된 형식으로 구조화된 데이터를 전송하고 수신하도록 설계되었다. 가장 기본적인 형식에서 XML은 꺾쇠 괄호(<>)로 묶인 태그를 정의하며 이러한 태그는 태그에 의해 설명된 데이터를 둘러쌉니다. 태그는 다른 태그 내부에 있는 태그로 계층을 형성할 수 있습니다. 예를 들어, 기본 전화기 디바이스를 정의하려면 전화기 디바이스에 이름, 설명 및 전화 번호의 세 가지 매개변수가 필요하다고 할 수 있습니다.

CUCM에서 원격 프로비저닝을 활성화하는 SOAP 기반 API입니다. CUCM 데이터베이스에서 정보를 추가, 업데이트, 제거 또는 검색하는 데 사용됩니다. 검색 기능에는 사용자 인증 확인 및 SQL 쿼리 실행이 포함됩니다. AXL API는 전체 CUCM 데이터베이스에 대한 액세스를 제공합니다. AXL API는 순수하게 프로비저닝을 위한 것이며 런타임 또는 성능 데이터에 대한 액세스를 제공하지 않습니다.

AXL API는 HTTPS 기본 인증을 사용합니다. 모든 CUCM 사용자(애플리케이션 또는 엔드 유저)는 **표준 CCM 수퍼 유저** 액세스 제어 그룹 또는 표준 AXL **API 액세스 역할이** 할당된 임의의 그룹의 멤버인 경우 AXL을 통해 읽기/쓰기 액세스를 갖습니다. 즉, 모든 슈퍼 사용자 어카운트가 이미 AXL API에 대한 액세스 권한을 갖고 있습니다. AXL API 사용 전용 계정을 생성하려면 먼저 액세스 제어 그룹을 생성하고 여기에 **표준 AXL API 액세스 역할**을 할당한 다음 애플리케이션 사용자를 새로 생성된 그룹과 연결해야 합니다. 읽기 전용 AXL API 액세스를 제공하려면 별도의 액세스 제어 그룹을 생성하고 여기에 **표준 AXL 읽기 전용 API 액세스 역할만**을 할당할 수 있습니다.

비누

SOAP(Simple Object Access Protocol)는 XML 형식으로 응용 프로그램 간에 정보를 전달하는 방법입니다. SOAP 메시지는 일반적으로 HTTP 세션을 통해 전송 애플리케이션에서 수신 애플리케이션으로 전송됩니다. 실제 SOAP 메시지는 Body 요소 및 선택적 Header 요소가 포함된 Envelope 요소로 구성됩니다.

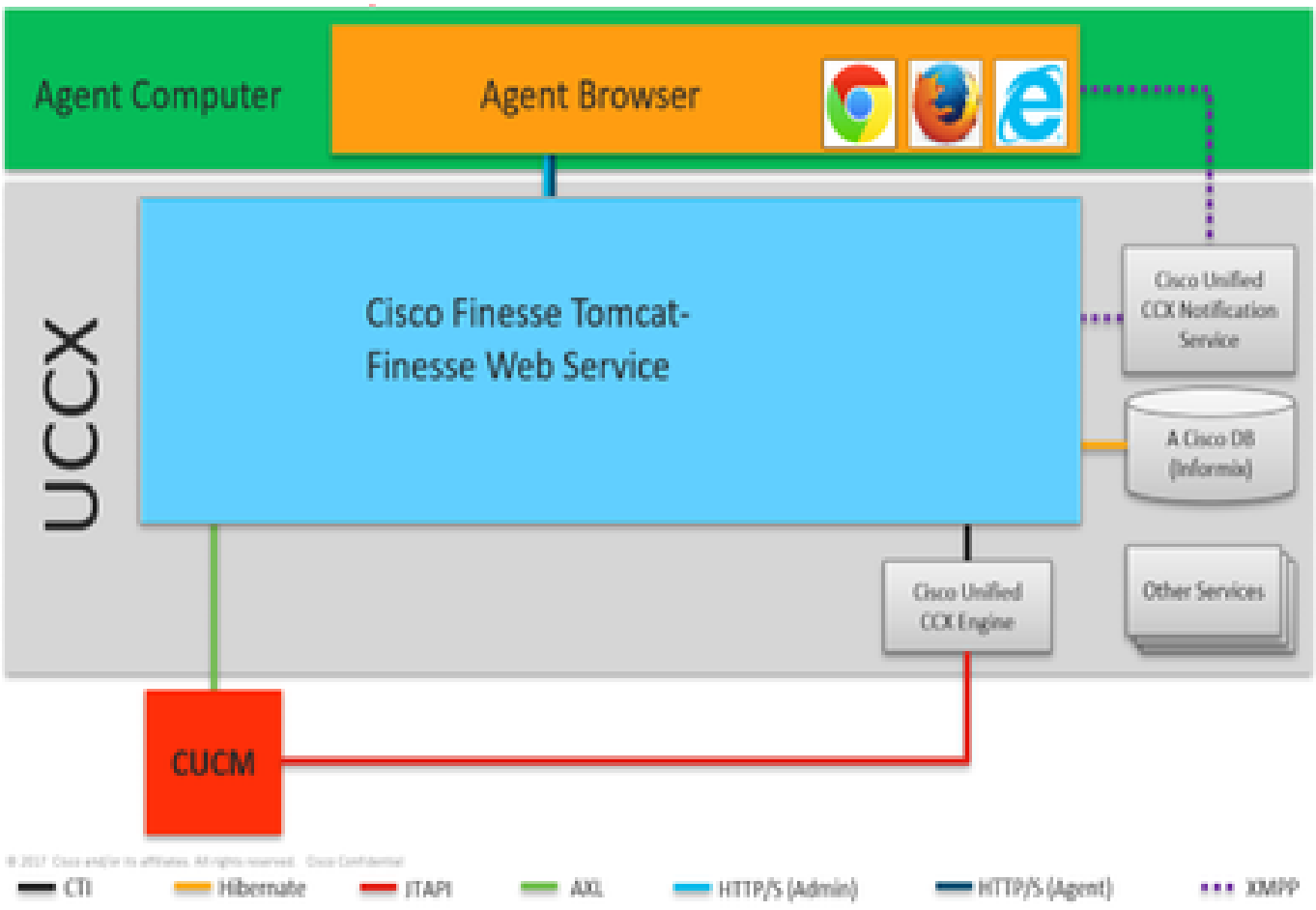
- Envelope - 이 필수 요소는 SOAP 메시지의 루트로, 전송된 XML을 SOAP 패킷으로 식별합니다. 봉투에는 본문 섹션과 선택적 헤더 섹션이 포함됩니다.

- 헤더 - 이 선택적 요소는 메시지에 대한 처리 정보를 나타내는 확장 메커니즘을 제공합니다. 예를 들어, 메시지를 사용하는 작업에 보안 자격 증명이 필요한 경우 해당 자격 증명은 봉투 헤더의 일부여야 합니다.

- Body - 이 요소는 메시지 페이로드를 포함하며, 원시 데이터는 전송 및 수신 애플리케이션 간에 전송됩니다. 본문 자체는 여러 자식 요소로 구성될 수 있으며 일반적으로 이 데이터의 구조를 정의하는 XML 스키마가 있습니다.

20000피트 뷰

다음 다이어그램에서는 Finesse 아키텍처와 관련된 프로토콜에 대해 좀 더 자세히 설명합니다.



20000 보기

서로 다른 UCCX 구성 요소 간의 통신을 담당하는 프로토콜입니다.

- UCCX 엔진 및 Finesse 서버는 CTI 프로토콜을 통해 통신합니다.

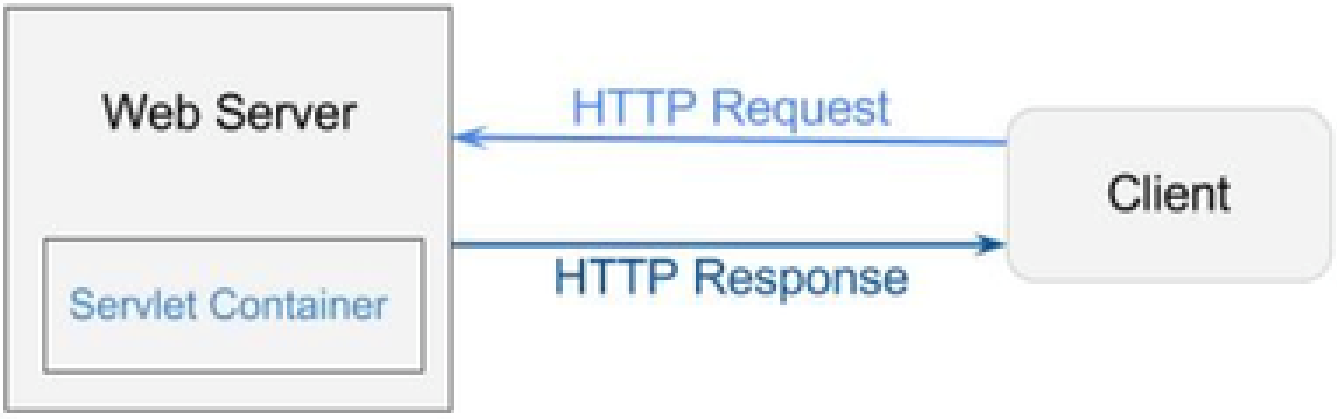
- UCCX 엔진 및 CUCM은 JTAPI 프로토콜을 통해 통신합니다.

- Finesse Tomcat과 CUCM은 AXL을 통해 대화합니다.

- Finesse 알림 서비스 및 에이전트 브라우저는 XMPP/BOSH에서 통신합니다.

- Finesse Tomcat과 데이터베이스는 최대 절전 모드에서 통신합니다.
- Finesse Tomcat 및 Finesse 알림은 XMPP를 통해 통신합니다.

아파치 신디그



신동

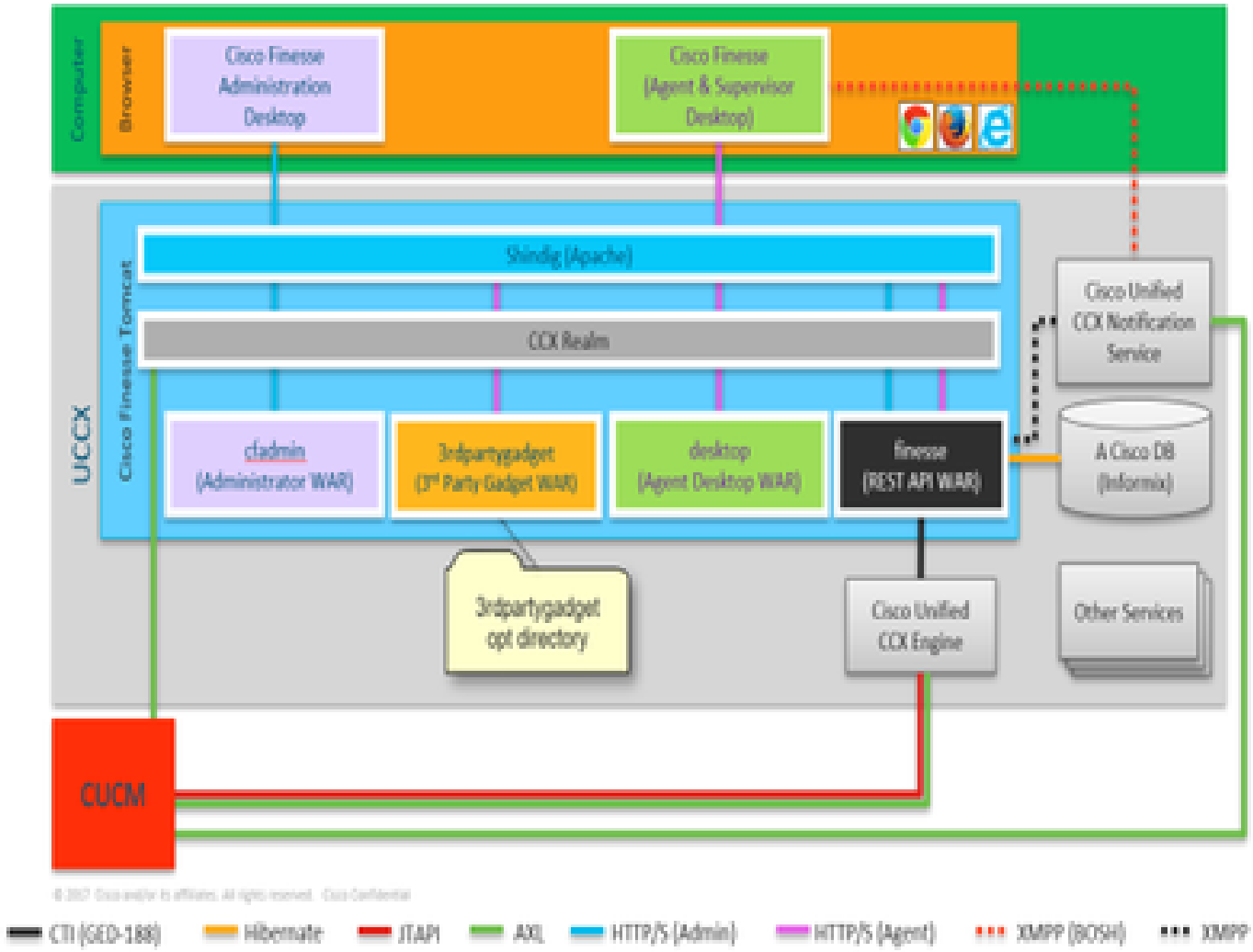
Apache Shindigg는 OpenSocial 컨테이너이며 가젯 렌더링, 프록시 요청, REST 및 RPC 요청 처리를 위한 코드를 제공하여 OpenSocial 앱 호스팅을 신속하게 시작할 수 있도록 지원합니다. OpenSocial은 웹에서 실행되는 소셜 애플리케이션을 구축하기 위한 API 집합입니다. (웹/서블릿) 컨테이너는 웹 서버에서 웹 페이지를 동적으로 생성하는 데 사용됩니다.

WAR 파일

WAR은 Web Archive를 의미합니다. 웹 프로젝트의 파일이 들어 있습니다. 서블릿, XML, JSP, 이미지, HTML, CSS, JS 등을 포함할 수 있습니다. 카탈리나 로그에는 구축되는 WAR에 대한 정보가 포함되어 있습니다.

10000피트 뷰

다음 다이어그램에서는 UCCX 및 Finesse 구성 요소 내에서 인증 흐름이 작동하는 방식에 대해 자세히 설명합니다.



10000 보기

WAR 파일은 로그인 방법에 따라 페이지를 표시하고 작성해야 합니다. 브라우저는 Shindigg에게 가젯을 렌더링해야 한다고 요청하고, Shindigg은 CUIC에 가젯을 렌더링하기 위해 이야기합니다. CCX 영역은 AXL을 사용하여 CUCM과의 인증에 사용됩니다. 알림 서비스는 또한 AXL을 사용하여 CUCM으로 인증합니다.

Finesse Rest API WAR은 알림 서비스, CCX 엔진 및 DB와 실제로 통신하는 주요 저장소입니다. Shindigg는 Finesse Rest API(WebServices)와만 대화합니다. cfadmin 및 desktop WARs는 페이지를 표시하는 데 불과하기 때문입니다. Finesse Rest API WAR에 대한 모든 내용은 Finesse WebServices 로그에서 확인할 수 있습니다. 이는 Finesse에 가장 중요한 로그입니다. Shindigg와 Finesse 웹 서비스(Rest API WAR) 간에 HTTP를 대화합니다. Finesse 웹 서비스(Rest API WAR)와 Engine은 CTI를 통해 서로 통신합니다.

아약스 - 피네세의 아름다움

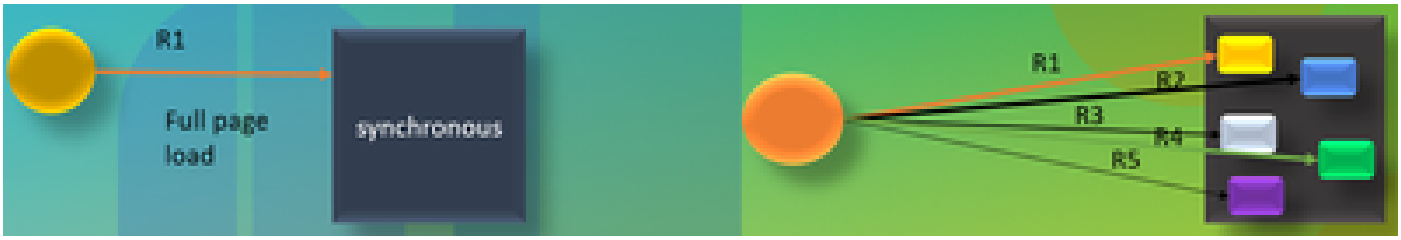
AJAX는 비동기 Javascript 및 XML을 나타냅니다. 프로그래밍 언어가 아니라 웹 페이지에서 웹 서버에 액세스하는 방법입니다. AJAX는 부분 페이지 업데이트를 위한 메커니즘입니다. 전체 페이지를 새로 고칠 필요 없이 서버에서 가져온 데이터로 페이지의 섹션을 업데이트할 수 있습니다.

예를 들어 Facebook 메신저에 대해 이야기할 때 새 메시지가 들어오면 전체 페이지를 새로 고쳐 메시지를 받을 필요가 없으며, 대신 페이지 자체의 메시지 섹션이 새로 고쳐지고 전체 페이지를 새로 고칠 필요 없이 실시간으로 새 메시지를 받을 수 있습니다.

모든 브라우저에는 XMLHTTPREQUEST(XHR이라고도 함)라는 기본 **개체가** 있습니다. 서버의 AJAX에 대한 모든 요청은 이 XML 요청을 거칩니다. 여기에는 업데이트해야 할 사항에 대한 세부 정보가 포함되어 있습니다.

AJAX 사용의 장점

다음 다이어그램에서는 비동기 요청과 동기 요청의 차이점에 대해 설명합니다.



아약스

동기식 요청의 경우 첫 번째 요청이 처리될 때까지 기다렸다가 두 번째 요청을 보낼 수 있습니다. 예를 들어, 페이지 새로 고침이 필요하며 페이지를 새로 고칠 때까지 아무 작업도 수행할 수 없습니다. 반면 비동기 요청의 경우 첫 번째 요청이 완료되어 두 번째 요청을 보낼 때까지 기다리지 않아도 됩니다. 여러 요청을 동시에 보낼 수 있습니다. 예를 들어, 웹 사이트에 날씨 앱 가젯. 페이지의 날씨 섹션을 새로 고칠 수 있으며, 동시에 전체 페이지를 새로 고칠 필요 없이 웹 사이트의 다른 섹션에서 작업할 수도 있습니다. 이는 비동기 요청의 주요 장점입니다.

AJAX 작업

AJAX는 데이터를 표시하거나 사용하는 Javascript 및 HTML과 함께 웹 서버에서 업데이트를 보내고 받는 데 사용되는 XHR(XMLHTTPREQUEST)의 조합입니다.

AJAX가 포함된 요청을 서버로 보내는 중

다음은 3단계 프로세스로, 다음에 설명되어 있습니다.

1. 변수를 생성하고 그 **변수**에 XHR 객체를 저장합니다.

Var 요청 = 새 XMLHttpRequest();

2. XHR 객체 내에 페이로드가 있는 요청 변수에 액세스합니다.

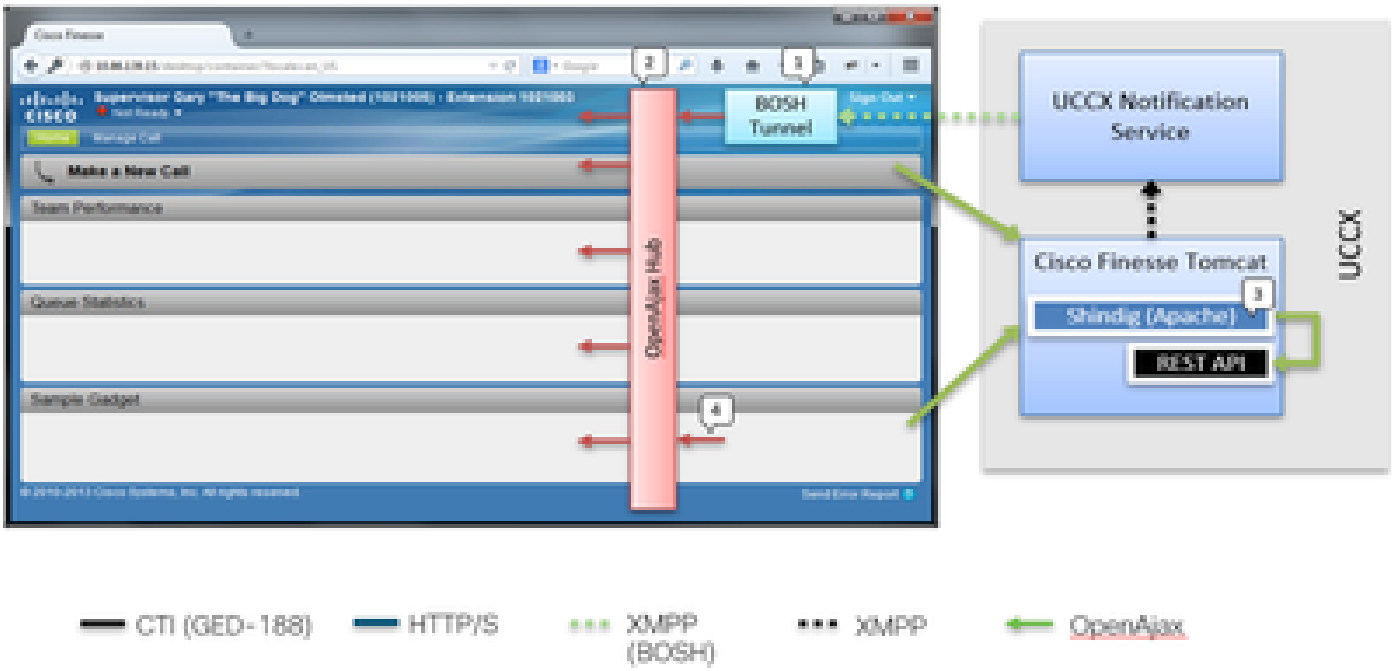
요청.**open(GET, URL)** ;

3. 요청 전송

Request.send() ;

데스크톱 아키텍처

다음 다이어그램에서는 가젯이 웹 페이지에서 렌더링될 때 AJAX 신호의 흐름을 설명합니다.

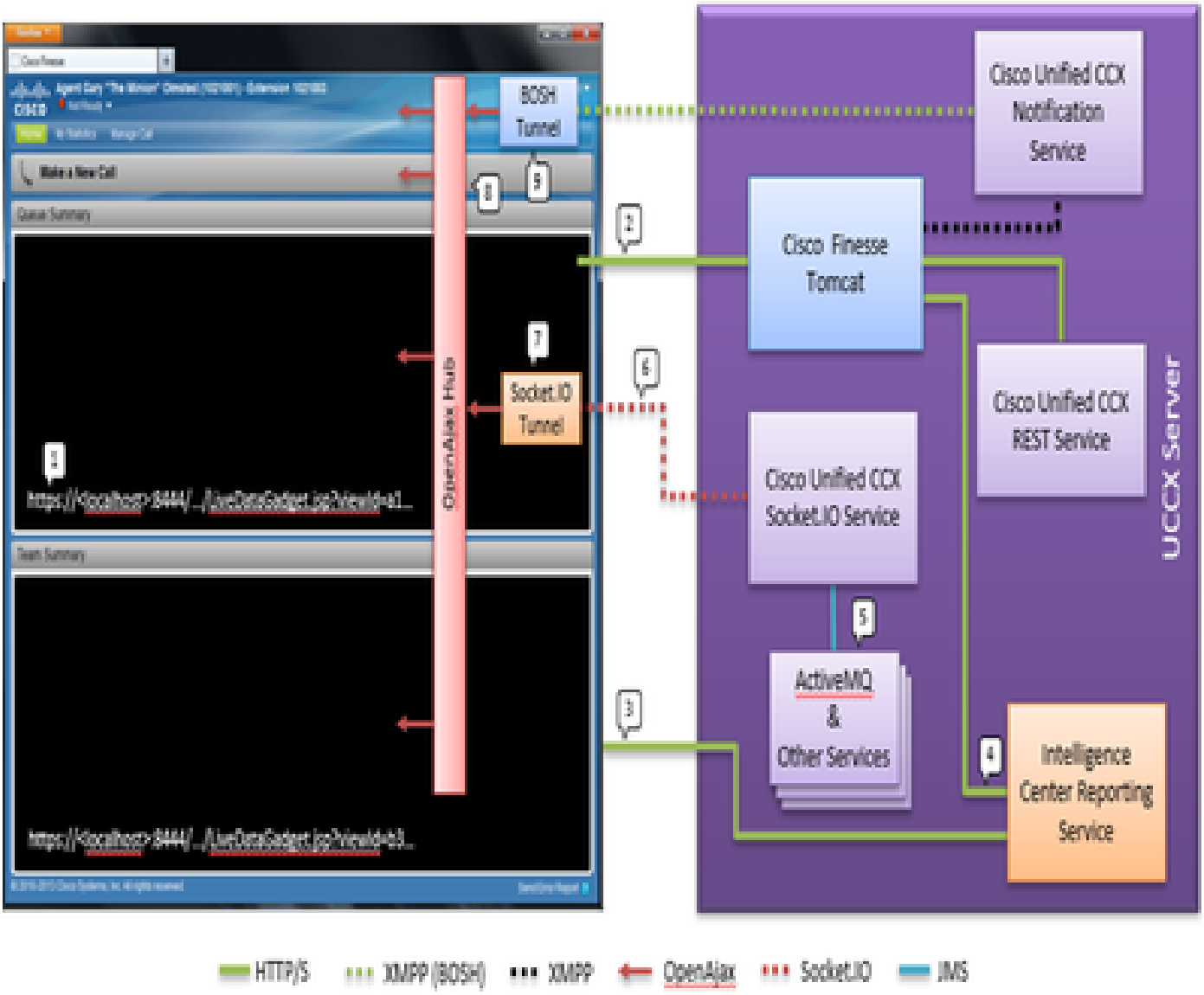


데스크톱 아키텍처

IFrame은 BOSH 터널을 호스트하기 위해 컨테이너에 상주합니다. OPENAJAX 허브는 가젯(pubsub 메서드 사용)을 통해 메시지를 게시하도록 제공되며 REST 요청은 Shindigg를 통해 다른 서버로도 프록시됩니다. 가젯은 AJAX 허브에 자신의 메시지를 게시할 수 있습니다.

가젯 아키텍처

다음 다이어그램에서는 Finesse 가젯 아키텍처에 대해 자세히 설명합니다.



가젯 아키텍처

일반적인 가젯과 달리 CUIC 가젯은 OpenFire에서도 실시간 XMPP 피드를 수신합니다. CUIC와 Finesse가 UCCX와 함께 상주하는 UCCX의 경우 공유 OpenFire 인스턴스가 있습니다. 대부분의 가젯 콘텐츠와 모든 REST API는 Finesse Server의 Shindigg를 통해 프록시됩니다. 이는 Finesse 가젯 및 REST API는 물론 CUIC 가젯 인스턴스 및 REST API에도 적용됩니다. CUIC 가젯은 D-Grid를 사용하여 보고서를 렌더링합니다. 부트스트래핑 프로세스가 있어야 하며, 이는 CUIC와 직접 연계하여 이루어집니다. 이러한 이유로 CUIC 가젯은 처음에 로드 프로세스 중에 CUIC 서버와 직접 통신합니다. 따라서 CUIC 인증서는 Socket.IO 터널과 함께 사용자 브라우저에서 승인되어야 합니다. 가젯 컨텐츠와 REST API는 Finesse와 CUIC 사이에서 클라이언트에 프록시됩니다. Rest API 호출은 Intelligence Center Reporting Service 및 CCX Web Service 모두에 대해 수행됩니다. CCX Live Data Socket.IO 서비스는 ActiveMQ의 JMS를 통해 Live Data에서 메시지를 가져옵니다. CCX Live Data Socket.IO 서비스는 클라이언트에서 Socket.IO 연결을 통해 실시간 보고 JSON을 게시합니다. Finesse Desktop에 Cisco Finesse Notification Service와의 BOSH 연결을 유지 관리하는 BOSH 터널 iFrame이 있는 것과 마찬가지로, 마스터 Live Data 가젯에는 CCX Live Data Socket.IO 서비스와의 Socket.IO(웹소켓) 연결을 유지 관리하는 Socket.IO 터널 iFrame이 있습니다.

OpenAjax 허브는 모든 이벤트를 구독된 리스너에 배포합니다. 이는 가젯과 Finesse 컨테이너 자체의 일부가 됩니다. Finesse 데스크톱에는 Cisco Unified CCX 알림 서비스와의 BOSH 연결을 유지 관리하는 BOSH 터널 iFrame이 있습니다. 이렇게 하면 OpenAjax 허브에 이벤트가 게시됩니다.

참조 링크

- Finesse 웹 서비스 개발자 가이드

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.