Compreendendo e pesquisar defeitos dados vivos relata em UCCX

Índice

Introdução

Este documento descreve o relatório vivo dos dados na solução expressa unificada do centro de contato (UCCX) que parte da versão 11.x e os esboços que pesquisam defeitos pontos de verificação junto com a análise de apoio do log. O artigo igualmente esboça várias advertências conhecidas com suas ações alternativas.

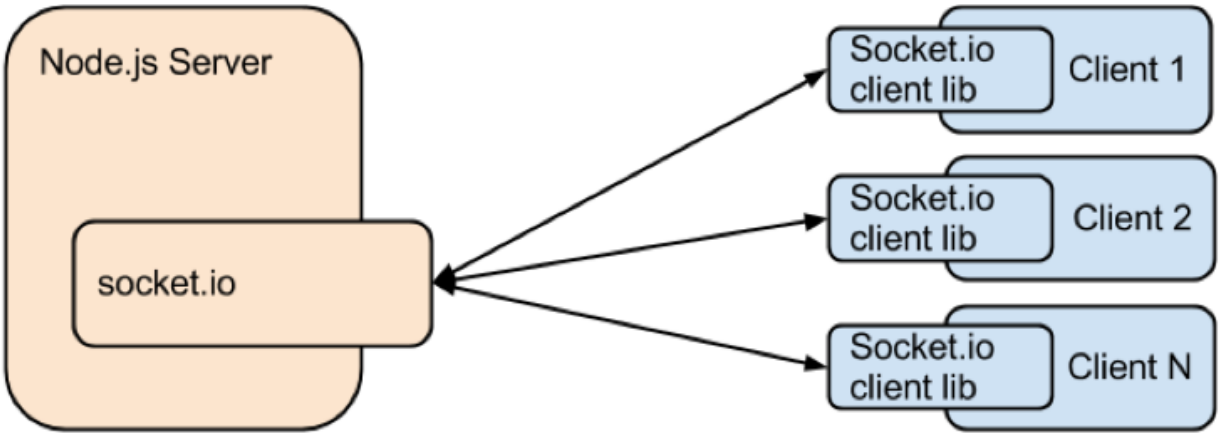
Mudanças em dados vivos com SocketIO

SocketIO baseou o projeto de dados vivo substitui Openfire server XMPP como a infraestrutura da mensagem do relatório do padrão de **UCCX 11.x** avante
Éuma solução escalável e eficiente evento-conduzida que seja baseada no impementation Node.js, com umas despesas gerais reduzidas de uma comunicação usando o Request For Comments 6455 do protocolo de WebSocket (RFC 6455) para enviar mensagens do tempo real ao navegador (clientes).

Mais cedo em UCCX 10.x, havia um único **serviço de notificação de Cisco CCX, com base em** OpenFire a aplicação XMPP) que foi usada para ambos fineza, bate-papo, notificações de Email assim como relatórios vivos dos dados em dispositivos do agente da fineza e em CUIC.

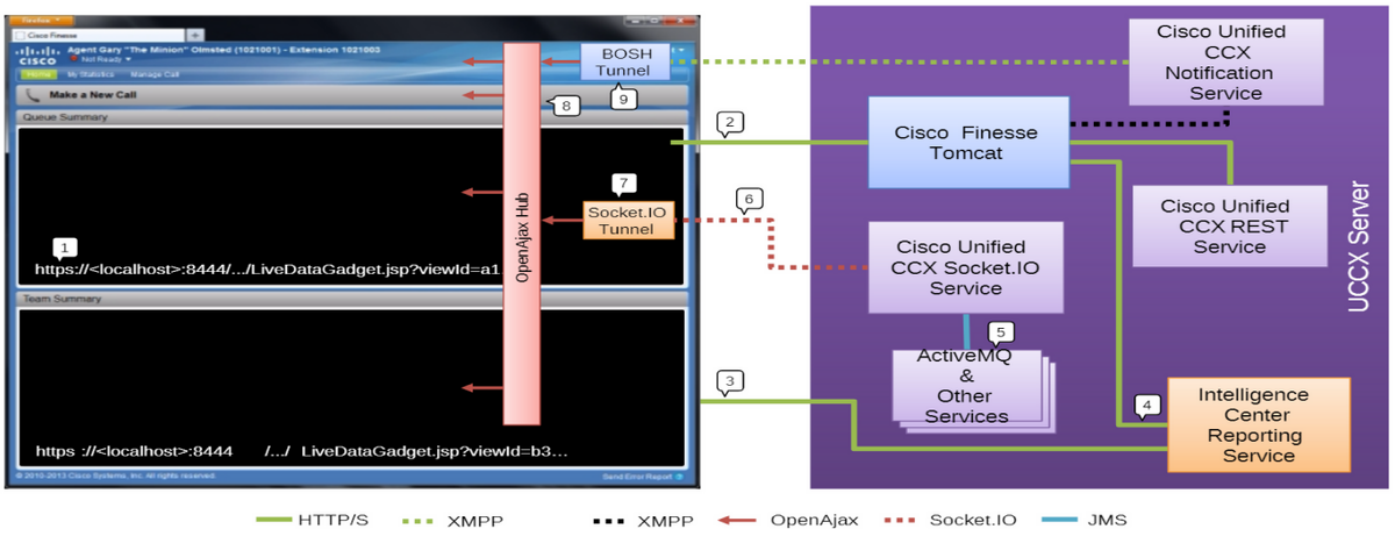
De UCCX 11.x, com a introdução de SocketIO para relatórios Live - o bate-papo e o email do cliente da fineza continuam a usar o **serviço de notificação de Cisco CCX**, isto é Openfire como o server XMPP, mas o módulo vivo do relatório dos dados usa o **serviço** novo do **serviço de** CiscoUnified **CCX Socket.IO**.

Isto reduz o relatório vivo dos dados em cima, e faz SocketIO unicamente responsável para esboçar e despachar as atualizações vivas dos dados ao navegador cliente, segundo as indicações da imagem do server de SocketIO.



Visão geral técnica

Esta imagem é de SocketIO baseou a visão geral arquitetural viva dos dados.



De uma perspectiva de nível elevado, os dados para relatórios dos dados Live são enviados na notação do objeto do Javascript (JSON) do módulo do gerente dos dados de tempo real do motor UCCX (RTDM), através do barramento do serviço de mensagem das Javas (JM) (uso Apache ActiveMQ executar isto, isto é fila ativa de Messsage), ao server Socket.IO hospedado em UCCX (exposto através do serviço unificado CCX Socket.IO) que recupera então esta informação, processam-no (consome) em salas e distribuem-nos então (expedições) este aos clientes que usam conexões HTTPS WebSocket sobre o TCP.

ActiveMQ é um middleware que execute a programação de aplicativo Interface(API) JM e seja usado enviando mensagens entre dois componentes, UCCX vive server dos dados (motor UCCX) e do SocketIO.

WebSocket contra o mecanismo de polling longo

Há duas maneiras que os navegadores podem subscrever para receber atualizações vivas dos dados, usando a tecnologia WebSocket(WS)/WebSocket Secure(WSS), ou usando o mecanismo de polling longo.

WebSocket (RFC 6455)

- O protocolo de WebSocket (RFC 6455) é um protocolo com base em TCP independente. Seu somente relacionamento ao HTTP é que seu aperto de mão está interpretado por Server do HTTP porque um pedido da elevação, e usado para estabelecer uma conexão persistente com o server para atualizações contínuas.

- Após um aperto de mão bem sucedido de WebSocket, os clientes e servidor transferem dados para a frente e para trás nas unidades conceptuais menores (objetos JSON) referidas nesta especificação como **mensagens**. A mensagem de WebSocket não corresponde necessariamente a uma camada de rede particular que molda, porque uma mensagem fragmentada pode ser coalescida ou rachado por um intermediário.

- Este processo começa com o cliente que envia um pedido do HTTP regular ao server.

- Um encabeçamento da elevação é incluído neste pedido que informa o server que o cliente deseja estabelecer uma conexão de WebSocket. isto é **elevação dos protocolos do interruptor**

HTTP/1.1101: **conexão do websocket: Elevação.**

- Após o esse, uma sala é aberta e as mensagens são despachadas continuamente ao cliente/navegador.
- WebSocket é conhecido para reduzir-se processar despesas gerais no servidor de publicação, e fornece atualizações vivas seguras e dinâmicas.

Votação longa

- Éuma variação da técnica tradicional da votação e permite a emulation de um impulso da informação de um server a um cliente.
- Com votação longa, a informação dos pedidos do cliente do server em uma maneira similar a uma votação normal. Uma conexão é guardada aberta entre o cliente web e o servidor de Web de modo que quando o server tem a informação nova possa a empurrar para o cliente e a conexão seja fechada então.
- Se o server não tem nenhuma informações disponíveis para o cliente, em vez da emissão uma resposta vazia, o server guarda o pedido e espera alguma informação para estar disponível.
- Uma vez que a informação se torna disponível (ou após um intervalo apropriado), uma resposta completa está enviada ao cliente.
- Os re-pedidos do cliente normalmente então imediatamente a informação do server, de modo que o server tenha sempre um pedido de espera disponível que possa usar para entregar dados em resposta a um evento.
- A votação longa causa despesas gerais adicionais no server que é pedido para esta informação.

Número de verificação de clientes que executam a votação longa

- Para UCCX vivem os dados, **WebSocket é favorável e recomendado** sobre o mecanismo tradicional da longo-votação.
- Para verificar se os clientes estejam usando WebSocket OU o mecanismo de polling longo para dados vivos, você pode analisar o SocketIO debuga e vê as mensagens.

Etapas:

- A volta acima do soquete IO debuga para nivelar DEBUGA (do padrão ADVIRTA) - navega à **utilidade > ao traço UCCX**.
- Ate os log de servidor os mais atrasados de SocketIO:

  Isto será exibido:

E.g: IE compatibility mode -
TimeStamp <LOG message> (POLLING): …

0000000135: 10.78.91.238: Nov 28 2016 20:19:44.297 +0530: %CCBU_nioEventLoopGroup-5-1-6-MessageDispatcher: %[address=/10.107.11.107:51809(WEBSOCKET)][message=Subscribe-{AgentCSQStats=[Email_CSQ]}][socket_io_server_type=WSS]: Subscription Request

E.g: Firefox/Chrome/IE Native with HTML5/websocket enabled:
TimeStamp <LOG message> (WEBSOCKET): …

Fluxo de dados vivo

O fluxo de dados vivo pode ser resumido como: **Dispositivos dos clientes do → do server de SocketIO do → do → JM ActiveMQ do motor CCX (ICD_RTDM, SS_RMCM) (navegador da Web)**

Estes são logging facilities que precisam de ser permitidas no motor UCCX:

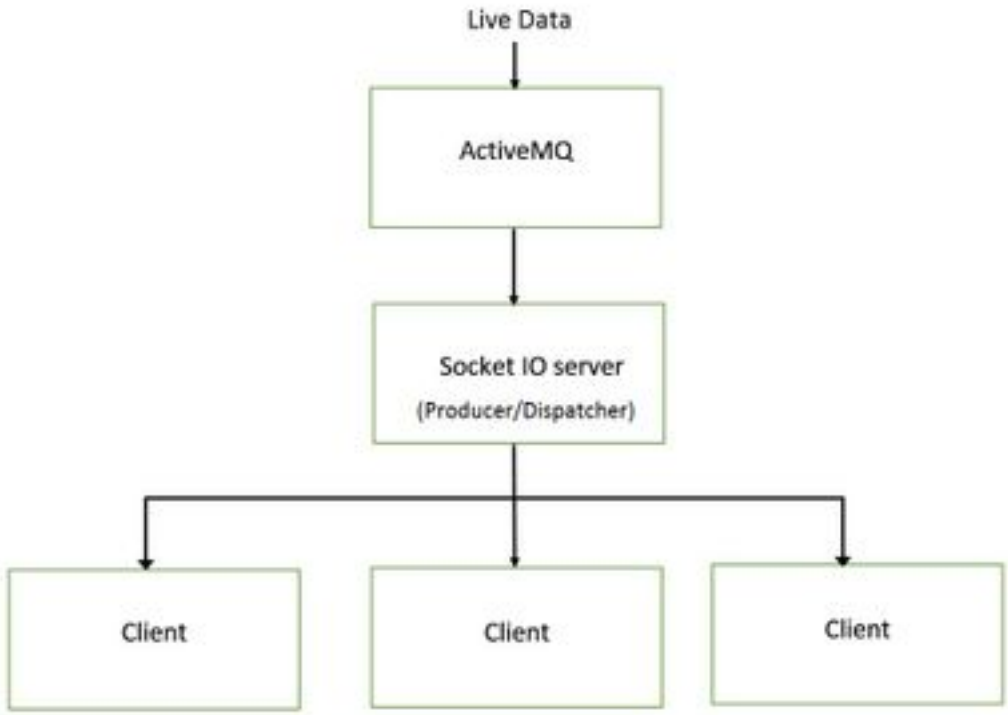
ICD_RTDM = gerente dos dados de tempo real de distribuição de chamada inteligente

SS_RMCM = contact manager do gerenciador de recurso do subsistema

JM ActiveMQ = aplicação ativa da fila da Mensagem do serviço de transferência de mensagem das Javas

O server vivo de SocketIO dos dados é projetado primeiramente receber a informação da origem de dados viva (tal como JM) e publicá-la, através de WebSocket, para viver clientes dos dados (como um dispositivo do cliente na fineza ou Cisco unificou o aplicativo de web do centro da inteligência (CUIC).

Este o diagrama esquemático lógico descreve o fluxo de dados do motor UCCX (dados vivos) em ActiveMQ, onde o server Socket.IO a seguir recupera a informação e a distribui aos clientes, segundo as indicações da imagem de SocketIO baseou assinaturas do cliente:



Os dados vivos publicam eventos nos grupos sobre assuntos através de ActiveMQ.

O server de SocketIO subscreve a estes assuntos e recebe os grupos do evento.

Os espaços de servidor & assuntos de SocketIO

O server de SocketIO é configurado com várias salas ou os assuntos definidos em **/opt/cisco/uccx/socketioserver/conf/socketioservice.properties** arquivam.

Todos os relatórios vivos dos dados são traçados a um assunto JM, que seja útil ao analisar gramaticalmente os logs.

A atualização do assunto JM acontece em cada 3 segundos, relata com certeza que estes são evento baseado (uma atualização é afixada somente quando um evento é provocado, como uma alteração de estado de agente).

São alistados na tabela do relatório ao mapeamento do assunto JM:

Relate o nome	Nome do assunto JM
Relatórico de estatística do agente CSQ	AgentCSQStats ---> o evento baseou o ass
Relatório do log do estado de agente	AgentStateDetailStats ---> o evento baseou o as
Relatório das estatísticas de agente	ResourceIAQStats
Relatórios sumário do equipe de agente	ResourceIAQStats
Relatório do estado da equipe	ResourceIAQStats
Relatórios sumário da equipe	ResourceIAQStats

Relatórios de detalhes da Voz CSQ	VoiceCSQDetailsStats
Sumário da Voz CSQ	VoiceIAQStats
Relatórios sumário do bate-papo CSQ	ChatQueueStatistics ---> o evento baseou o ass
Relatório das estatísticas de agente do bate-papo	ChatAgentStats ---> o evento baseou o assunt
Relatório das estatísticas de agente do email	EmailAgentStats ---> o evento baseou o assunt
Relatórios sumário do email CSQ	EmailQueueStatistics ---> o evento baseou o ass

Os clientes como o server CUIC conectam ao server Socket.IO para subscrever às salas alistadas acima a fim receber os eventos vivos dos dados do server de SocketIO.

O pedido da assinatura está enviado do CUIC enquanto um relatório é lançado logo do CUIC. Por exemplo os **relatórios sumário de** lançamento da **Voz CSQ no** CUIC provocam um pedido da assinatura à sala/assunto correspondentes (VoiceIAQStats) ao server SIO.

Produtor

1. Recebe grupos do evento de ActiveMQ.
2. Quebra-os para baixo em seus eventos discretos.
3. Auges em cada evento para obter a identificação do evento.
4. Distribui o evento à sala baseada no assunto combinado com a identificação obtida.

Expedidor

O expedidor recebe a sala discreta nomeada eventos do produtor e envia estes eventos a todos os clientes que subscrevem a essa sala.

Clientes

1. Os clientes com base na Web (CUIC OU de agentes/supervisor da fineza dispositivos) conectam ao server de SocketIO sobre o soquete da Web.
2. Uma vez que conectados, estes clientes subscrevem às salas no server do soquete IO.
3. Diversos clientes podem subscrever à mesma sala.
4. Os clientes recebem um evento de cada vez.

Exemplo:

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
```

```
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

O server de SocketIO começa então publicar eventos ao cliente com esta assinatura.

Autenticação

O server Socket.IO precisa de autenticar os clientes de conexão (como o dispositivo dos dados da fineza OU o CUIC vivo) antes de enviar os dados.

1. O server Socket.IO propõe usar a autenticação baseada token.

2. Os dispositivos vivos dos dados executam um pedido de autenticação inicial que lhes entregue um token do AUTH.

3. Isto é passado mais sobre ao server Socket.IO para autenticar a criação de uma conexão WSS (server de WebSocket). Para isto, Socket.IO chama o API fornecido por UCCX com o token que obteve dos dispositivos.

4. Toda a autenticação é feita usando uma corda simbólica. A authenticação do cliente é permitida no serviço Socket.IO através da propriedade Client.Authentication.Rest.URL.

5. Todos conectar clientes permitidos devem fornecer esta corda como um parâmetro na conexão URL.

6. Os clientes ao server Socket.IO conectam, subscrevem, cancelar assinatura, e disconexão. A conexão inicial pode ser apenas uma conexão ou igualmente incluir a assinatura.

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

WS = websocket

wss = websocket seguro

O keystore que contém os Certificados associados para WSS (UCCX usa o padrão: Certificados do aplicativo de Cisco Tomcat)

O lugar da frase de passagem para WSS (padrão: Tomcat do sistema)

Traçado vivo dos dados

Para registrar, há 3 pontos chaves do fluxo em dados vivos UCCX usando a arquitetura de

SocketIO em 11.x **(motor MIVR_RTDM, motor SS_RMCM > JM ActiveMQ > o server de SocketIO**)

- O motor UCCX registra (MIVR com grupo RTDM e RMCM a Xdebugging5) a emissão da mensagem aos JM (cada 3 segundos como visto no subsistema RM debugam no motor (MIVR) o SS_RMCM)

- Logs de SocketIO (o registro do serviço de SocketIO deve ser ajustado à eliminação de erros completa onde os JM enviam mensagens ao consumidor de SocketIO, que recebe todas as mensagens que estão sendo publicadas pelo motor).
- Logs do console do navegador (F12): SocketIO que processa as atualizações recebidas, criando salas apropriadas para cada assunto, e enviando então estes aos navegadores da Web dos clientes através do expedidor da mensagem. (Estes são vistos nos logs do navegador da Web Console/F12).

Estes traços têm que ser permitidos a fim considerar o detalhado messaged nos logs.

1. Motor UCCX: **ICD_RTDM, SS_RMCM** com nível 5 de Xdebugging
2. Registro do serviço de SocketIO: Nível de debug.
3. Logs do console do navegador (F12) para verificar atualizações vivas entrantes dos dados.

SocketIO debuga logs diz somente o estado dos assuntos como conectado/desligou-o e subscreveu-o/unsubscribed.

O traçado detalhado tem que ser permitido a fim considerar eventos específicos em um assunto. Os logs do produtor e do expedidor de SocketIO estão disponíveis somente depois a possibilidade do traçado detalhado (tac Cisco do contato para permitir traçado detalhado).

Pesquisando defeitos dados vivos

Healthcheck básico

A primeira etapa para edições vivas dos dados do Troubleshooting é estabelecer uma linha de base da configuração suportada e executar um exame médico completo básico do sistema.

As edições encontradas em pontos de verificação desta configuração podem causar edições na rendição viva dos dados no dispositivo da fineza OU no página da web CUIC.

Você deve ordenar para fora todos estes pontos antes de continuar com o Troubleshooting baseado log.

Resumo

- A versão de navegador deve ser compatível.

- O CPU e a utilização de memória no UCCX devem estar dentro dos limites.

- A origem de dados viva dos dados deve ser em linha na página das origens de dados CUIC.

- O NTP e o DNS no UCCX devem ser em serviço, com o NTP na sincronização, e o DNS para a frente/consultas reversas na ordem correta.

- Os ÓVULOS VM devem ser corretamente fornecida com exigências do vRAM e do vCPU de acordo com diretrizes dos ÓVULOS de Cisco.

- O NIC no uso para o VM deve ser VMXNET3.

Verifique a versão de navegador

Verifique o comportamento de dados Live através dos navegadores e das versões de navegador diferentes.

UCCX apoia atualmente as seguintes versões de navegador que começam 11.5: http://docwiki.cisco.com/wiki/Unified_CCX_Software_Compatibility_Matrix_for_11.5(1)#Supported_Browsers

Recorde DESABILITAR o modo de compatibilidade no IE desde que isso não é apoiado para dados Live. O IE cai de volta ao mecanismo de polling longo quando em longo votar assim isto é uma consideração importante.

Verifique o CPU e a utilização de memória

Verifique o USO de CPU e a utilização de memória no server usando RTMT ou comandos CLI.

Por exemplo: RTMT >

CLI: mostre o processador central da carga de processo

CLI: mostre a memória da carga de processo

Verifique para ver se há estes processos se dados vivos de vista em CUIC: UCCX_Engine, socketioservice, cuicreporting.

Verifique que a fonte viva do fluxo de dados é em linha

A fim verificar que a fonte viva do fluxo de dados é em linha, navegue ao **Web page CUIC > ao sistema > às origens de dados**

Verifique o NTP.

Verifique que o NTP é acessível e UCCX é sincronizado ao servidor de NTP no estrato <=5

CLI: estado NTP dos utils

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
```

0000133771FC3C33F57F543","isKeyField":true}] report-layer.js:1414:11

sincronizado ao servidor de NTP (xx.xx.xx.xx) no estrato 2

Verifique o DNS

Verifique que o DNS está resolvendo.

CLI: <hostname> do host de rede dos utils

Resolução local:

resoluções hostname.domainname localmente a xx.xx.xx.xx

Definição externo:

hostname.domainname tem o endereço xx.xx.xx.xx

Execute um teste diagnóstico

Você pode executar um teste de diagnósticos do CLI para identificar todo o outro problema potencial com o server (por exemplo latência de Tomcat, questões de rede, etc.)

CLI: os utils diagnosticam o teste

Verifique o hostname

O hostname UCCX deve mim no lowercase.

Consulte por favor instalam/guias da elevação e este defeito.

http://cdets.cisco.com/apps/dumpcr?identifier=CSCva75058&content=summary&format=html

Verifique a configuração VM

Verifique que isso o molde aberto correto do dispositivo da virtualização (ÓVULOS) está usado conforme estes links:

https://www.cisco.com/c/dam/en/us/td/docs/voice_ip_comm/uc_system/virtualization/virtualization-cisco-unified-contact-center-express.html#11.5

https://software.cisco.com/download/type.html?mdfid=286287033&flowid=76362

O ajuste do adaptador de rede UCCX VM (NIC) deve ser **VMXNET3**.

Se o adaptador correto não é selecionado então UCCX não publica os dados de tempo real em constante avalia.

Se o ajuste do adaptador de rede VM não é VMXNET3 a seguir seguem este link para corrigir os

ajustes do adaptador:
http://docwiki.cisco.com/wiki/Virtualization_for_Cisco_Unified_Contact_Center_Express

Verifique Certificados UCCX

Certificate a má combinação ou quando os Certificados não são aceitados pode fazer com que a fonte viva do fluxo de dados vá desligado/off line.

- Aceite o certificado quando alertado.
- Assegure-se de que os Certificados da terceira parte não estejam expirados.
- Verifique que o hostname no certificado está correto. Se não, siga o procedimento abaixo.

No lado CCX:

1. Edite este arquivo **/usr/local/platform/conf/platformConfig.xml** e corrija o hostname e certifique-se de que as entradas de DNS são conforme o nome neste arquivo.
2. Suprima e regenere dos Certificados após ter verificado os detalhes acima no conjunto UCCX usando os comandos CLI:
   CLI: ajuste CERT REGEN TomCat

3. Reinicie o serviço Cisco Tomcat em UCCX.
   CLI: utils service restart Cisco Tomcat

Mais em Certificados:

http://www.cisco.com/c/en/us/support/docs/customer-collaboration/unified-contact-center-express/118855-configure-uccx-00.html

Verifique a configuração de execução sob medida apoiada

Assegure-se de que os limites da configuração máxima estejam aderidos na instalação. Para isto, consulte o Guia de Design do nível da solução UCCX para a configuração do servidor e limites máximos:

UCCX
11.5: http://www.cisco.com/c/en/us/td/docs/voice_ip_comm/cust_contact/contact_center/crs/express_11_5/desi..

UCCX 11.0: http://www.cisco.com/c/en/us/td/docs/voice_ip_comm/cust_contact/contact_center/crs/express_11_0/desi..

Para confirmar isto você pode executar estes comandos CLI, verificando a diretriz da cola, na instalação para obter o número exato.

INFORMAÇÃO REQUERIDA	CONSULTA
Total não dos supervisores atribuídos às equipes	**execute a contagem seleta da db_cra sql do uccx (*) de (resourceloginI distinto seleto do supervisor onde active = "t ")**
Totalize não de supervisores configurados	**execute a contagem SELETA da db_cra sql do uccx (*) do recurso onde ative='t e resourcetype=2**
Totalize não de agentes configurados	**execute a contagem SELETA da db_cra sql do uccx (*) do recurso onde ative='t e resourcetype=1**
Totalize não de equipes configuradas	**execute a contagem SELETA da db_cra sql do uccx (*) da equipe onde ative='t**

Não dos agentes pela equipe	execute a db_cra team.teamname SELETO sql do uccx, count(resource.resourceid) COMO NumOfAgents do recurso da JUNTA INTERNA da equipe em team.teamid=resource.assignedTeamID onde GRUPO resource.active='t e team.active='t pelo teamname
Não dos supervisores pela equipe	execute a db_cra team.teamName SELETO sql do uccx, count(Supervisor.recordID) COMO NumOfSupervisors da equipe da JU INTERNA do supervisor em Supervisor.managedTeamID=team.teamid onde GRUPO Supervisor.active='t e team.active='t pelo teamName
Número de usuários pelo papel (CUIC)	execute sql g.name seleto como o nome de grupo, conte-o (*) como groupsize do cuic_data: cuicgroup g, cuic_data: gm do cuicgroupmemb onde grupo g.id = gm.groupid por g.name
Número de CSQ configurados	execute a contagem seleta da db_cra sql do uccx (*) do contactservice onde ative='t

Caveats conhecidos

CSCva13838 **:** Atraso vivo dos dados em CUIC

CSCus17605 **:** Atraso devido às permissões excessivas do relatório aos supervisores (carga elevada)

CSCux33949 **:** Edições vivas dos dados devido às licenças aumentadas.

CSCvb67761 **:** Relatório do log do estado de agente que causa a alta utilização da CPU que impacta dados vivos.

CSCva95411 **:** Alta utilização da CPU do serviço de SocketIO devido aos objetos namespace que tomam a memória excessiva do montão.

CSCvb75279 **:** Fonte viva do fluxo de dados que vai off line durante fora da senha da sincronização MIVR Keystore

CSCvc45189: Fonte viva do fluxo de dados off line devido à operação alternativa e da restauração

CSCut04158: Edições vivas dos dados devido ao adaptador NIC incorreto

> Nota: Para confirmar ou a ação alternativa as advertências acima, envolve por favor o tac Cisco abrindo um pedido do serviço.

Os dados vivos registram o seguimento

Logs do motor

- Os dados vivos publicam eventos nos grupos sobre assuntos através de MQ ativo.
- O server de SocketIO subscreve a estes assuntos e recebe os grupos do evento.

Logs MIVR: Os dados vivos são enviados do motor através de uma mensagem JM sobre os

assuntos (por exemplo: VoiceCSQDetailsStats, VoiceIAQStats, ResourceIAQStats etc…)
Para o motor baseado evento dos assuntos (ChatAgentStats, ChatQueueStatistics AgentCSQStats, AgentStateDetailStats) envia mensagens vazias JM após cada 3 segundos para manter a sala de SocketIO viva.

Estes traços areseen somente depois a possibilidade destes traços: SocketIO detalhou o seguimento

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

Logs de SocketIO (1)

SocketIO recebe grupos do evento de MQ ativo

Enquanto os grupos de eventos chegam, o server de SocketIO faz este:

- Quebra-os para baixo em seus eventos discretos.
- Auges em cada evento para obter a identificação do evento.
- Distribui o evento à sala baseada no assunto combinado com a identificação obtida

Os eventos discretos com nomes da sala são enviados à fila no expedidor

> Nota: O produtor de SocketIO que tem o ouvinte/consumidor JM obtém os dados do barramento JM nos assuntos através de ActiveMQ e envia-os às salas de SocketIO que são enviadas à fila ao expedidor.

Estes traços são considerados somente depois a possibilidade dos traços detalhados em SocketIO (tac Cisco do contato para permitir o mesmos).

Exemplo 1: Assunto de VoiceIAQStats.

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

Exemplo 2: Assunto de VoiceCSQDetailsStats.

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11
```

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

Exemplo 3: Assunto de ResourceIAQStats

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

Logs de SocketIO (2)

Expedidor do soquete IO: Obtém eventos do produtor e entrega-os aos clientes

O expedidor recebe a sala discreta nomeada eventos do produtor e envia estes eventos a todos os clientes que subscrevem a essa sala.

Estes traços são considerados somente depois a possibilidade dos traços detalhados em SocketIO (tac Cisco do contato para permitir o mesmos).

Exemplo 1: Assunto de VoiceIAQStats.

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

Exemplo 2: Assunto de VoiceCSQDetailsStats.

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

Exemplo 3: Assunto de ResourceIAQStats

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

Cliente Logs(1)

Mostras de registro do cliente que conectam, subscrevendo e recebendo eventos.

1. Os clientes com base na Web conectam ao server como WebSocket ou clientes longos da votação.
2. Uma vez que conectado, ou como parte de conectar estes clientes subscreva às salas no server do soquete IO.
   - Diversos clientes podem subscrever à mesma sala.
   - Cada sala somente obterá/recebe eventos desse tipo
3. Os clientes receberão 1 evento de cada vez.

Ser executado relatórios significa a subscrição aos eventos do server SIO.

O relatório de lançamento CUIC provoca um pedido da assinatura ao assunto que corresponde ao relatório ao server de SocketIO.

Example1: CUIC que envia o pedido da assinatura após ter executado **relatórios sumário da Voz CSQ** (assunto VoiceIAQStats)

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

Example2: CUIC que envia o pedido da assinatura após ter executado **relatórios de detalhes do agente da Voz CSQ** (assunto VoiceCSQDetailsStats)

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

Example3: CUIC que envia o pedido da assinatura após **relatório** running das **estatísticas de agente** (assunto: ResourceIAQStats)

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

O server de SocketIO recebeu o pedido da assinatura e JUNTA-SE à sala.

Logs de SocketIO

Example1: O server SIO JUNTA-SE após ter recebido o pedido da assinatura (assunto

VoiceIAQStats)

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

Example2: O server SIO JUNTA-SE após ter recebido o pedido da assinatura (assunto VoiceCSQDetailsStats)

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

Example3: O server SIO JUNTA-SE após ter recebido o pedido da assinatura (assunto ResourceIAQStats)

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

Logs do cliente (2)

Server de SocketIO após ter recebido o pedido da assinatura, começos que enviam os eventos através do expedidor da mensagem. Estão abaixo as atualizações do evento que são recebidas no navegador.

Log do navegador: A atualização é recebida no navegador (imprensa F12 no navegador para ver a atualização)

Example1: O navegador CUIC recebe as mensagens no navegador (assunto VoiceIAQStats)

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
<VoiceIAQStats=cssCsq>  report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}]  report-layer.js:1414:11
```

Example2: O navegador CUIC recebe as mensagens no navegador (assunto VoiceCSQDetailsStats)

```
1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= Subscribing for rooms:
```

<VoiceIAQStats=cssCsq> report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}] report-layer.js:1414:11

Example2: O navegador CUIC recebe as mensagens no navegador (assunto ResourceIAQStats)

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= **Subscribing for rooms:**
<VoiceIAQStats=cssCsq> report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}] report-layer.js:1414:11

A fineza recebe a mesma atualização no agente/supervisor desktop. (Imprensa F12 no navegador para ver as atualizações)

Example1:

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= **Subscribing for rooms:**
<VoiceIAQStats=cssCsq> report-layer.js:1414:11

1/18/2017, 10:14:07 PM=INFOC8E2DB0C10000140000000A40A4E5E6B= ReportManager : filterString
[{"fieldId":"720298FB10000140000000A10A4E5E6F","fieldType":"VALUELIST","name":"VoiceIAQStats.esd
Name","operator":"SetValues","value":[{"key":"cssCsq","desc":"cssCsq"}],"valuelistId":"01FB2C011
0000133771FC3C33F57F543","isKeyField":true}] report-layer.js:1414:11