Cisco live!

January 28 – February 1, 2019 · Barcelona

CISCO

IMAGINE

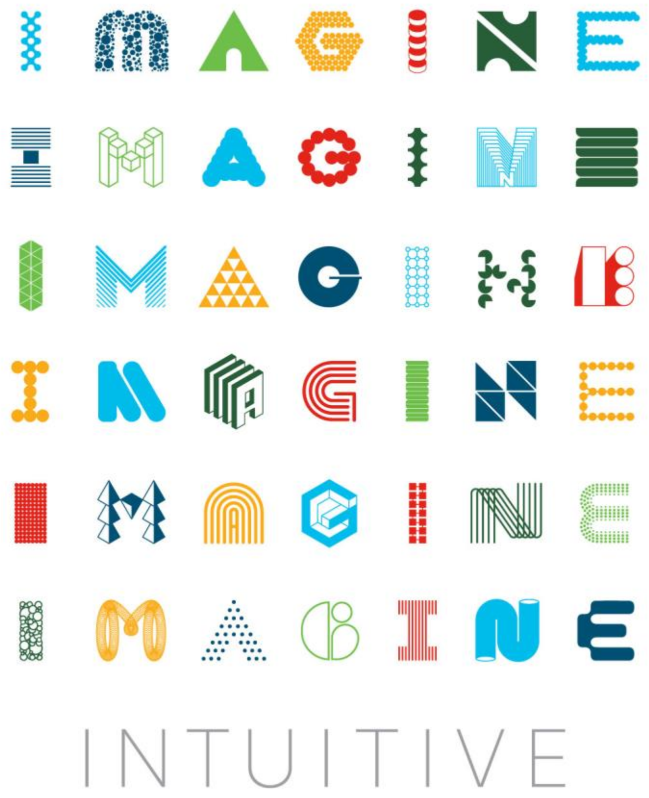
INTUITIVE

BRKCCT-2541

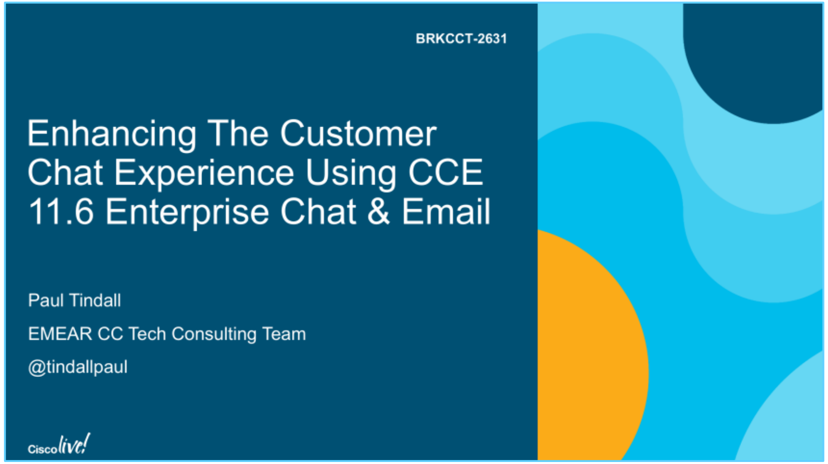
Implementing AI-Driven Conversational IVR on CVP and CCX

Paul Tindall

EMEAR CC Technical Consulting



Barcelona, One Year Ago



We explored ...

How to provide slick, pain-free chat experiences, supporting the chat client of choice as well as exploiting Artificial Intelligence for chat responses.

Barcelona, 12 Months On

The Natural Follow-On Question ...

Seeing what's possible with text chat and Artificial Intelligence,

how can I achieve exactly the same thing with voice calls?

Agenda

- Examine the use case and understand the challenge
- Transcription using Google Speech To Text
- CCX and CVP applications with IBM Watson Assistant
- See and hear it working
- Taking it further – other possible use cases

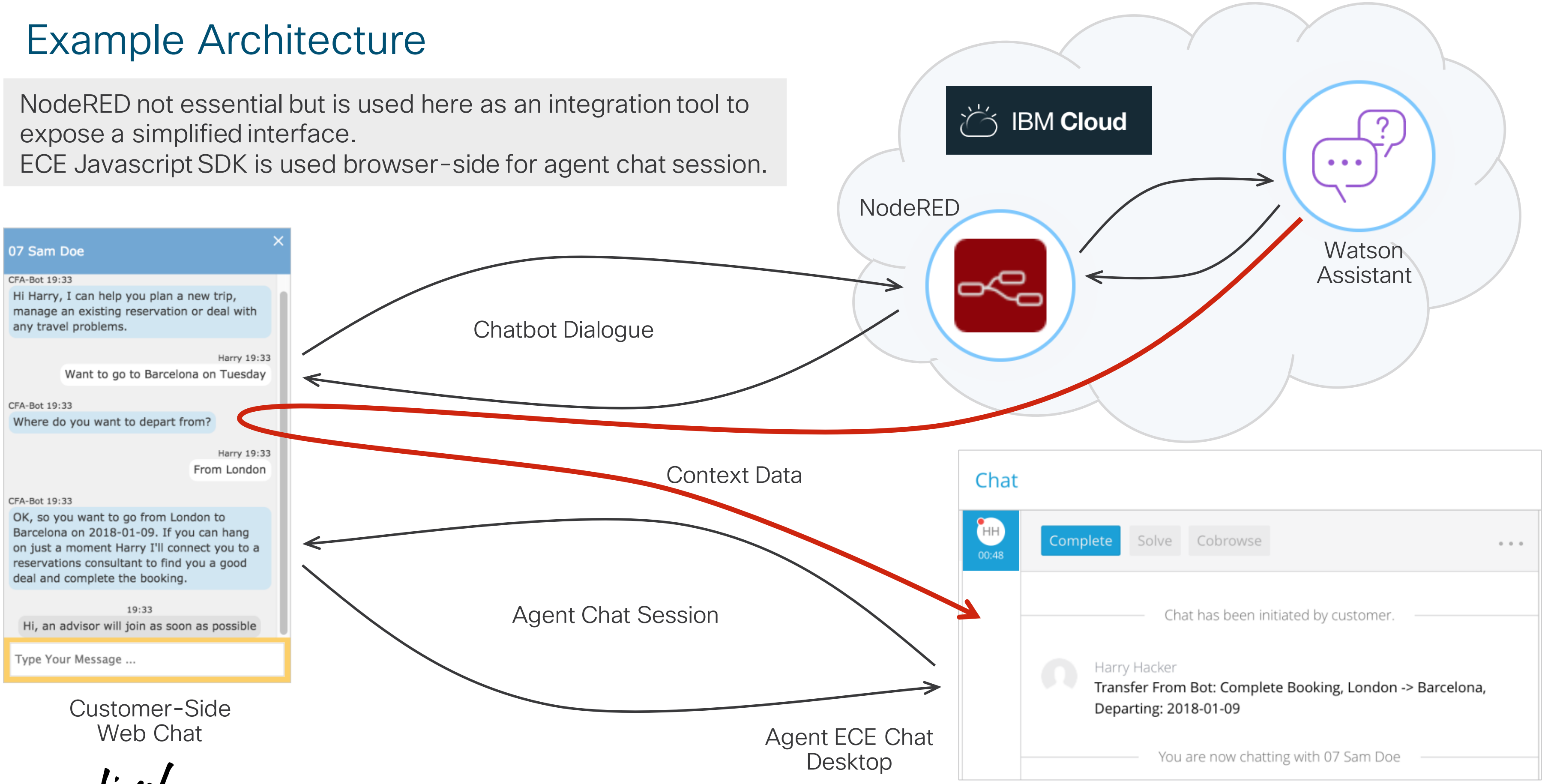
Brief Recap – Text Chat, AI, Seamless Transfer

- Customer engages initially with AI-driven chatbot in natural language dialogue
- For chats that remain unsatisfied by the chatbot conversation –
  - Seamless transition to agent chat using the same chat user interface
  - Pass useful context and caller intent derived from the chatbot session
- Customer chats using generic messaging client
  - Messenger applications such as Telegram, LINE, WhatsApp (API semi-available)
  - Social media messaging – Twitter, Facebook
  - SMS text messaging

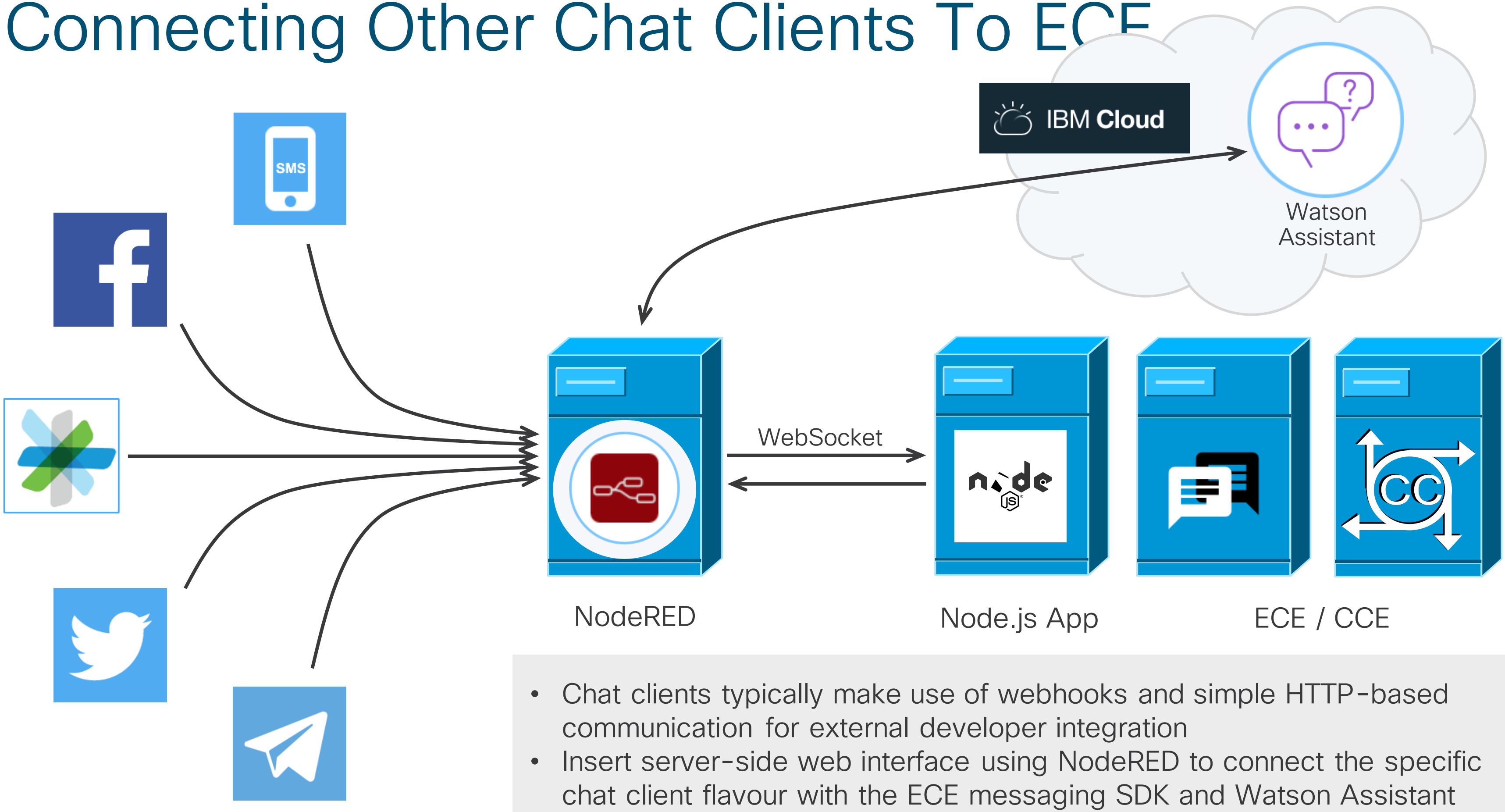
ECE Web Chat + Watson Assistant

Example Architecture

NodeRED not essential but is used here as an integration tool to expose a simplified interface.
ECE Javascript SDK is used browser-side for agent chat session.



Connecting Other Chat Clients To ECE



- Chat clients typically make use of webhooks and simple HTTP-based communication for external developer integration
- Insert server-side web interface using NodeRED to connect the specific chat client flavour with the ECE messaging SDK and Watson Assistant
- Remote server-side Node.js app that uses the ECE Javascript SDK

The Follow-on Question

How can we do exactly the same thing but with voice calls?

- Voice calls terminating on CVP or CCX
- AI-driven IVR dialogue, using the same AI conversation engine as for chat
- Subsequent call transfer to agent with context and intent

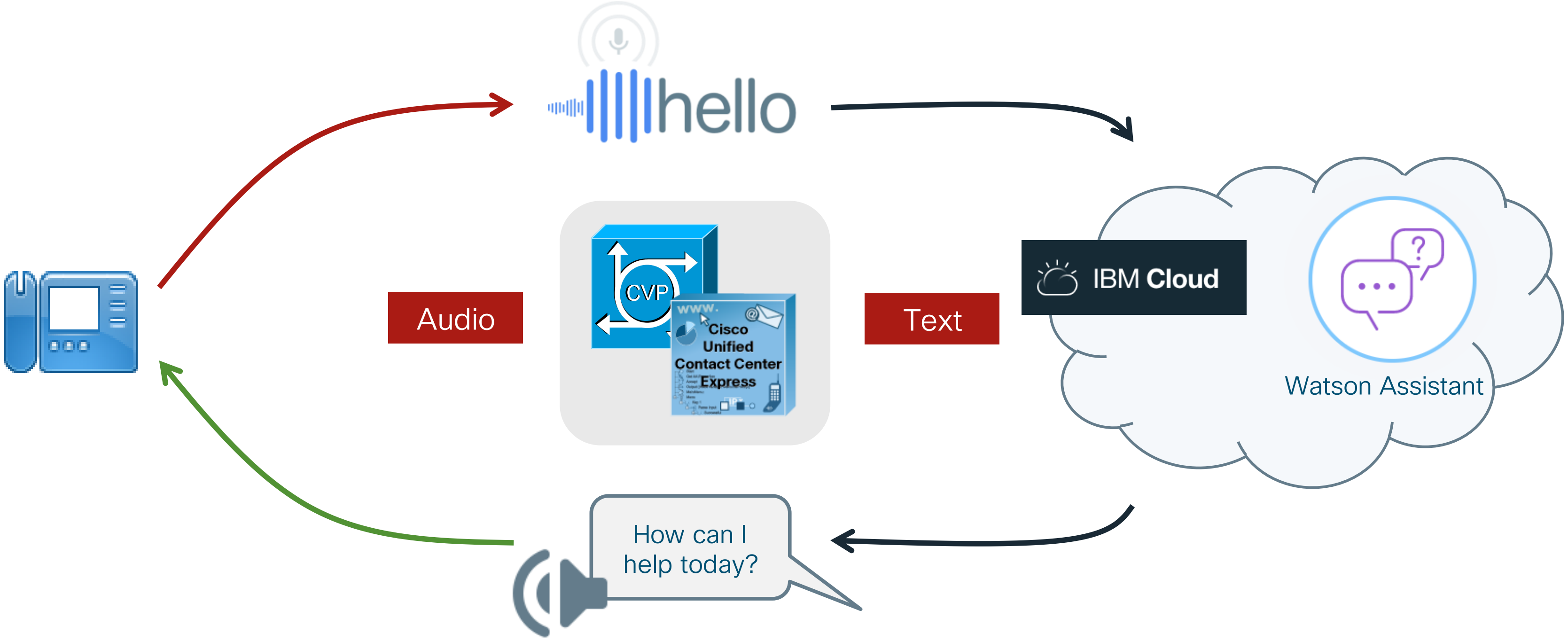
The Follow-on Question

How can we do exactly the same thing but with voice calls?

- Voice calls terminating on CVP or CCX
- AI-driven IVR dialogue, using the same AI conversation engine as for chat
- Subsequent call transfer to agent with context and intent

This one is the challenge
Everything else is just standard IVR

AI-Driven IVR Dialogue



Consider The Things We Need To Do

In decreasing order of difficulty

1. Access the caller’s media stream
2. Use a transcription service for speech to text
3. Use text as input to AI conversation
4. Play text output from AI conversation

   The simplest part, use existing TTS if you have it

Biggest Challenge – Accessing Caller's Media

The simple approach ...

- Record caller utterance until final-silence detected
- Upload to transcription service
- Useful as a quick way to show the overall concept

But ...

- Delays on silence detection, upload & transcription
- Need a real-time approach with transcript as soon as end of utterance detected

Use MRCP To Access Caller's Media

It's what ASR does ...

- Good architectural fit with IVR the way it works already
- Media streamed in real-time from voice browser

But ...

- Have to build custom MRCP server
- Controlled via VoiceXML documents
- Ties a useful capability just to IVR
- Would require core product development work for both CVP and CCX
- Can't use it alongside standard MRCP server deployment when using VVB
    - IOS VB did use com.cisco.asr-server property to dynamically override server

Alternative Approach – Gateway Media Forking

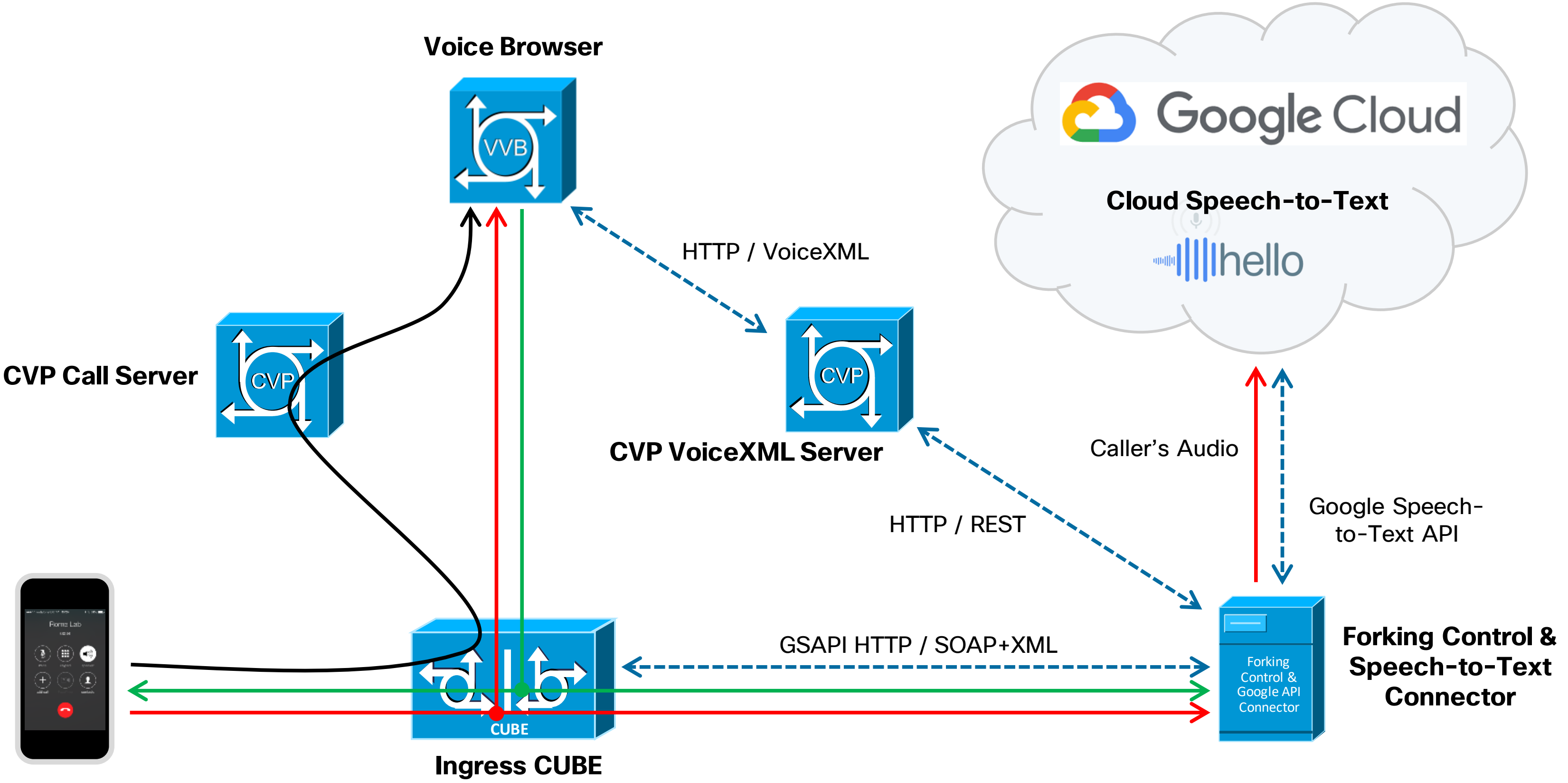
It’s what CUCM Network Based Recording (NBR) uses …

- Simple way to access caller, agent or both media streams
- Control from IVR application via back-end messaging
- Same solution for both CVP and CCX
- Use the same service for agent desktop-triggered media processing operations
- Makes possible a whole range of back-end services needing access to media or real-time call state
- More flexibility than MRCP on how it can be used
  - Could trigger totally asynchronous and continuous media processing

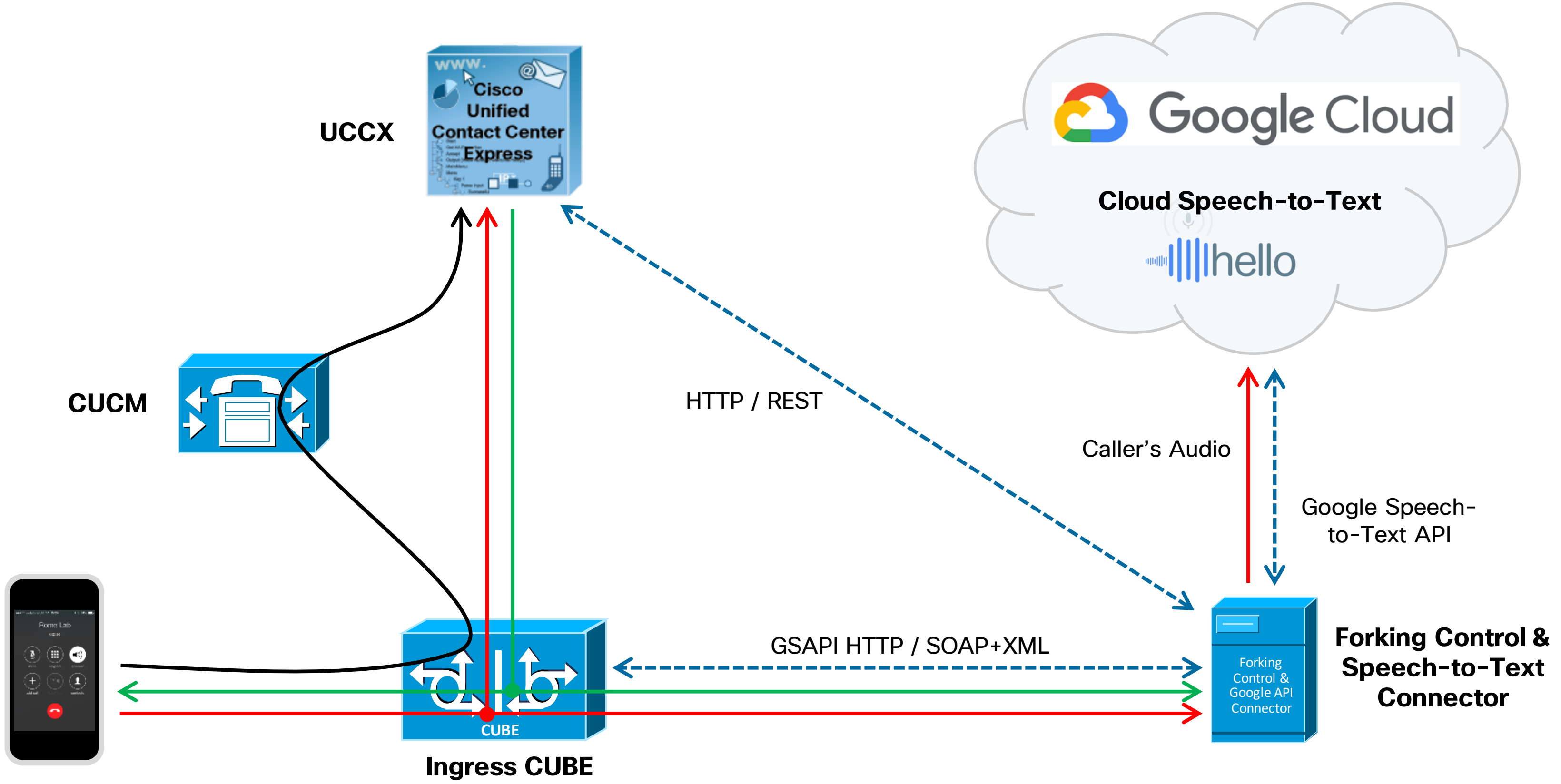
But …

- Gateway Services API (GSAPI) not used by many applications and few examples

CVP Real-Time Streaming Transcription



CCX Real-Time Streaming Transcription



Gateway Services API – How Do We Use It?

Documentation link: [Cisco UC Gateway Services API Guide]

- HTTP/SOAP+XML web services hosted on the voice gateway
  - Call Control
  - CDRs
  - Serviceability
  - Media Forking
- Application can start/stop media forking to specified IP address/port
- How do we use the Extended Media Forking (XMF) service?

Extended Media Forking (XMF) Service

Main messages we need to use:

- RequestXmfRegister (application → gateway)
  - Application initiates communication with the gateway
- RequestXmfCallMediaForking (application → gateway)
  - Start and Stop media forking
- NotifyXmfConnectionData (gateway → application)
  - Receive info on calls connected / disconnected

START Media Forking

RequestXmfCallMediaForking

```
<env:Body>
    <ns2:RequestXmfCallMediaForking xmlns:ns2="http://www.cisco.com/schema/cisco_xmf/v1_0">
        <ns2:action>
            <ns2:enableMediaForking>
                <ns2:farEndAddr>
                    <ns2:ipv4>10.58.16.187</ns2:ipv4>
                    <ns2:port>16435</ns2:port>
                </ns2:farEndAddr>
                <ns2:nearEndAddr>
                    <ns2:ipv4>10.58.16.187</ns2:ipv4>
                    <ns2:port>16434</ns2:port>
                </ns2:nearEndAddr>
            </ns2:enableMediaForking>
        </ns2:action>
        <ns2:callID>143</ns2:callID>
        <ns2:msgHeader>
            <ns2:transactionID>183</ns2:transactionID>
            <ns2:registrationID>C3A7E718:XMF:com.cisco.pt.cvp.forking:51</ns2:registrationID>
        </ns2:msgHeader>
    </ns2:RequestXmfCallMediaForking>
</env:Body>
```

START media forking for this call ID

Destination addr & port for called party media

Destination addr & port for calling party media

Gateway's INTERNAL call ID

Where Does The Call ID Come From?

```
<ns2:callID>143</ns2:callID>
```

- ID that's required in requests to Start / Stop forking
- But, not available to CVP or CCX scripts
- Call connection notifications from the gateway include the Call ID
- And, they also include other fields that we can use in CVP/CCX scripts
- XMF application must maintain a map of Call ID to other call data fields
  - CVP – use the GUID
  - CCX – use a unique destination DN with correlation ID appended

Connection Data Notification

NotifyXmfConnectionData

```
<SOAP:Body>
    <NotifyXmfConnectionData xmlns="http://www.cisco.com/schema/cisco_xmf/v1_0">
        <msgHeader> ... </msgHeader>
        <callData>
            <callID>143</callID>
            <state>ACTIVE</state>
        </callData>
        <connData>
            <connID>1263129</connID>
            <state>CONNECTED</state>
        </connData>
        <event>
            <connected>
                <connDetailData>
                    <guid>0xD3A4CCF9-0xF0B911E8-0x8095001B-0x0CFAA768</guid>
                    <guidAltFormat>3550792953-4038660584-2157248539-0217753448</guidAltFormat>
                    <callingAddrData> ... </callingAddrData>
                    <calledAddrData>
                        <type>E164</type>
                        <addr>651964190</addr>
                    </calledAddrData>
```

For CONNECTED calls add entry to Call ID map, joining to GUID for CVP and called number for CCX

GUID format edited to remove -0x will match CVP Data.Session.callid

Called number, used in CCX implementation (Revisit later how it's used)

Application Connecting To The Gateway

- Application sends registration request to –
  - http://gateway-host:8090/cisco_xmf

- Request includes –
  - Application identification
  - Servlet URL to receive unsolicited notifications
    - For example, call connected and disconnected events
  - List of call events to subscribe to
  - List of media events to subscribe to

Register With Gateway XMF Service

RequestXmfRegister

```
<env:Body>
    <ns2:RequestXmfRegister xmlns:ns2="http://www.cisco.com/schema/cisco_xmf/v1_0">

        <ns2:applicationData>
            <ns2:url>http://10.58.16.187:19090/forkctrl/forking</ns2:url>
            <ns2:name>com.cisco.pt.cvp.forking</ns2:name>
        </ns2:applicationData>

        <ns2:connectionEventsFilter>CONNECTED DISCONNECTED</ns2:connectionEventsFilter>
        <ns2:mediaEventsFilter>MEDIA_ACTIVITY</ns2:mediaEventsFilter>
        <ns2:msgHeader> ... </ns2:msgHeader>

        <ns2:providerData>
            <ns2:url>http://10.58.16.172:8090/cisco_xmf</ns2:url>
        </ns2:providerData>
    </ns2:RequestXmfRegister>
</env:Body>
```

Application servlet URL to receive gateway events must match IOS uc wsapi configuration

Gateway event types to be reported

Gateway XMF provider URL in request must match the register message target gateway

At The Gateway

Configuration

```
uc wsapi
 probing interval keepalive 30
 !
 provider xmf
  remote-url 2 http://10.58.16.187:19090/forkctrl/forking
  remote-url 3 http://10.58.16.175:8090/ucm_xmf
```

Pre-configure external application URLs that will register with gateway services

Status

```
rmlab-cube2#sho wsapi registration xmf
Provider XMF
=====================================================
registration index: 5
  id: 5B35FD7C:XMF:com.cisco.pt.cvp.forking:52
  appUrl:http://10.58.16.187:19090/forkctrl/forking
  appName: com.cisco.pt.cvp.forking
  provUrl: http://10.58.16.172:8090/cisco_xmf
  prober state: STEADY
  connEventsFilter: CONNECTED|DISCONNECTED
  mediaEventsFilter: MEDIA_ACTIVITY
```

Show XMF provider status and see registration information and state for currently connected applications

Working With XMF Service SOAP Interface

- Normally start with WSDL (Web Services Description Language)
  - But, not found one that builds successfully
  - Trying to fix non-working WSDLs is a guaranteed path to insanity

- Stay sane, use alternative approach, do this ...
  1. Take existing sample messages or generate some manually (Postman)
  2. Generate an XML schema definition (XSD) automatically from SOAP/XML
     - Tools such as https://www.freeformatter.com/xsd-generator.html (Salami Slice mode best)
  3. For Java, use the little-known XJC command line tool
     - Generate Java classes from the XSD we just created
  4. Use JAXB methods to read/write XML message content to/from Java classes

Other XMF Service Considerations

- Handle keepalives and recovery from lost connection
  - SolicitXmfProbing messages from gateway at configured probing interval
  - Application must send ResponseXmfProbing
  - Re-register if probe interval expires and no messages received from the gateway

- Clear zombies from the call ID map after reasonable max call duration
  - Zombied map entries if call disconnection while application not registered

- Gateway Call ID is not unique across multiple gateways
  - Call ID mapping must include gateway host/IP for uniqueness

Functions The Forking Connector Performs

1. Communicates with gateways
   - Receives call events and maintains Call ID table
   - Turns media forking on/off
2. Media destination for forked streams
   - Discards packets or extracts payload for processing
3. Performs back-end function such as interfacing with Google Cloud
   - Creates transcription session
   - Forwards audio payload
   - Waits for transcription results
4. Handles HTTP requests from applications – CVP, CCX, desktop

Handling The Media Streams

- Remember the RequestXmfCallMediaForking message?
- Needs destination IP address and port for the two voice streams
- Could direct the forked media directly to a separate service, but ...
  - Must be able to handle RTP packets
  - Convenient if transcription service provided addresses/ports, but no ...
- Google Speech To Text service requires raw audio payload only
- Have to receive RTP packets and forward using the Speech To Text API
  - https://cloud.google.com/speech-to-text/docs/streaming-recognize

Receiving The Media RTP Packets

- Create/open UDP channel, assign port, wait for packet, process contents

Create UDP datagram socket to receive RTP packets on dynamically allocated port

```
chn = DatagramChannel.open();
chn.socket().bind(new InetSocketAddress(addr, newport));
```

Running asynchronously, wait for packet to arrive

```
chn.receive(rxbuf);
```

Then, asynchronously process the packet, inspect/check the RTP header if required, extract the audio payload

```
int pktlen = rxbuf.position();

byte[] hdr = Arrays.copyOfRange(rxbuf.array(), 0, 12);
byte[] payload = Arrays.copyOfRange(rxbuf.array(), 12, pktlen);

pkthandler.accept(payload);
```

Pass the audio payload to handler for forwarding to cloud speech to text service

Invoking Transcription (Google Speech To Text)

- Create a speech client, first request sent is configuration

```
reccfg = RecognitionConfig.newBuilder()
            .setEncoding(RecognitionConfig.AudioEncoding.MULAW)
            .setSampleRateHertz(8000)
            .setLanguageCode(lang)
            .build();

strcfg = StreamingRecognitionConfig.newBuilder()
            .setConfig(reccfg)
            .setSingleUtterance(true)
            .build();

try (SpeechClient speech = SpeechClient.create()) {

    stream = speech.streamingRecognizeCallable().call();
    cfgreq = StreamingRecognizeRequest.newBuilder().setStreamingConfig(strcfg).build();

    stream.send(cfgreq);
```

Audio config: codec, sampling and langauge

Return results after single utterance detected

Create streaming speech session

Send configuration as the first request

Sending Audio (Google Speech To Text)

- Subsequent requests send the audio payload

```
rtp.start();

rtp.processMedia((raw) -> {
    StreamingRecognizeRequest.Builder strreq = StreamingRecognizeRequest.newBuilder();
    strreq.setAudioContent(ByteString.copyFrom(raw));

    stream.send(strreq.build());
});


for (StreamingRecognizeResponse rsp : stream) {

    // Handle transcription results

    if (rsp.getSpeechEventType().equals(END_OF_SINGLE_UTTERANCE)) {
        ...
}
```

Start receiving RTP packets and processing

Set packet handling function

RTP packet handler creates streaming request, adds audio payload and sends it

Wait for transcription events and results

Transcription Results (Google Speech To Text)

- Process events and build the transcription outcome as JSON object

```
if (rsp.getSpeechEventType().equals(END_OF_SINGLE_UTTERANCE)) {
    rtp.discardMedia();
    stream.closeSend();

} else if (rsp.getError().getCode() != 0) {
    // Handle error condition

} else {
    StreamingRecognitionResult result = rsp.getResultsList().get(0);

    if (result.getIsFinal()) {
        SpeechRecognitionAlternative alt = result.getAlternatives(0);
        outcome.put("transcript", alt.getTranscript())
               .put("confidence", (new DecimalFormat("0.00")).format(alt.getConfidence()));

        stream.cancel();
    }
}
```

Utterance detected so stop sending media

Format JSON object with transcript results

```
{
    "transcript": "To be or not to be that is the question.",
    "confidence": "0.97"
}
```

Transcription Results (Google Speech To Text)

- Process events and build the transcription outcome as JSON object

CCX Transcript Results

Name	Type	Value	Attributes
callid	String	"7705000019572"	
forkingUrl	String	"http://198.18.133.37:19090/forkctrl"	Parameter
jsonResp	Document	TEXT[{\"transcript\":\"To be or not to be that is the question.\",\"confidence\":\"0.97\"}]	
transcript	String	"To be or not to be that is the question."	
xbrResponse	String	U"{\"transcript\":\"To be or not to be that is the question.\",\"confidence\":\"0.97\"}"	
xbrStatusCause	String	"PUT http://198.18.133.37:19090/forkctrl/transcription/7705000019572 returned a response status of ...	
xbrStatusCode	String	"202"	

ding media

Format JSON object with transcript results

```
    if (result.getIsFinal()) {
        SpeechRecognitionAlternative alt = result.getAlternatives(0);
        outcome.put("transcript", alt.getTranscript())
               .put("confidence", (new DecimalFormat("0.00")).format(alt.getConfidence()));

        stream.cancel();
    }
}
```

```
{
    "transcript": "To be or not to be that is the question.",
    "confidence": "0.97"
}
```

Transcription And Forking Requests

Simple Web Application

- HTTP requests with JSON format body

Transcriptio
n

```
{
    "party": "calling",
    "language": "es-ES"
}
```

- URL path includes the unique call ID

```
http://<host:port/path>/transcription/<call_leg_ID>
```

Transcription And Forking Requests

Simple Web Application

- HTTP requests with JSON format body

Transcriptio
n

```
{
    "party": "calling",
    "language": "es-ES"
}
```

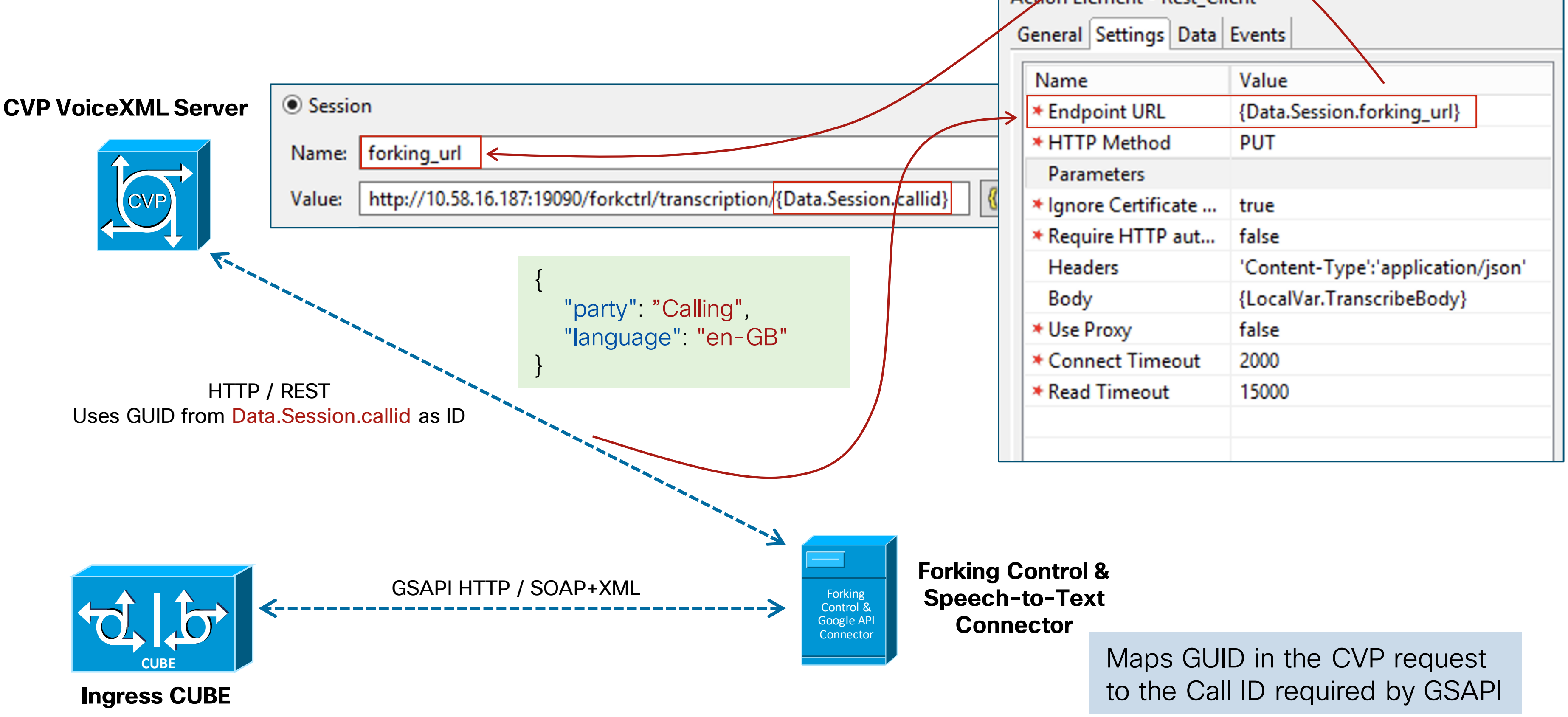
- URL path includes the unique call ID

```
http://<host:port/path>/transcription/<call_leg_ID>
http://<host:port/path>/forking/<call_leg_ID>
```

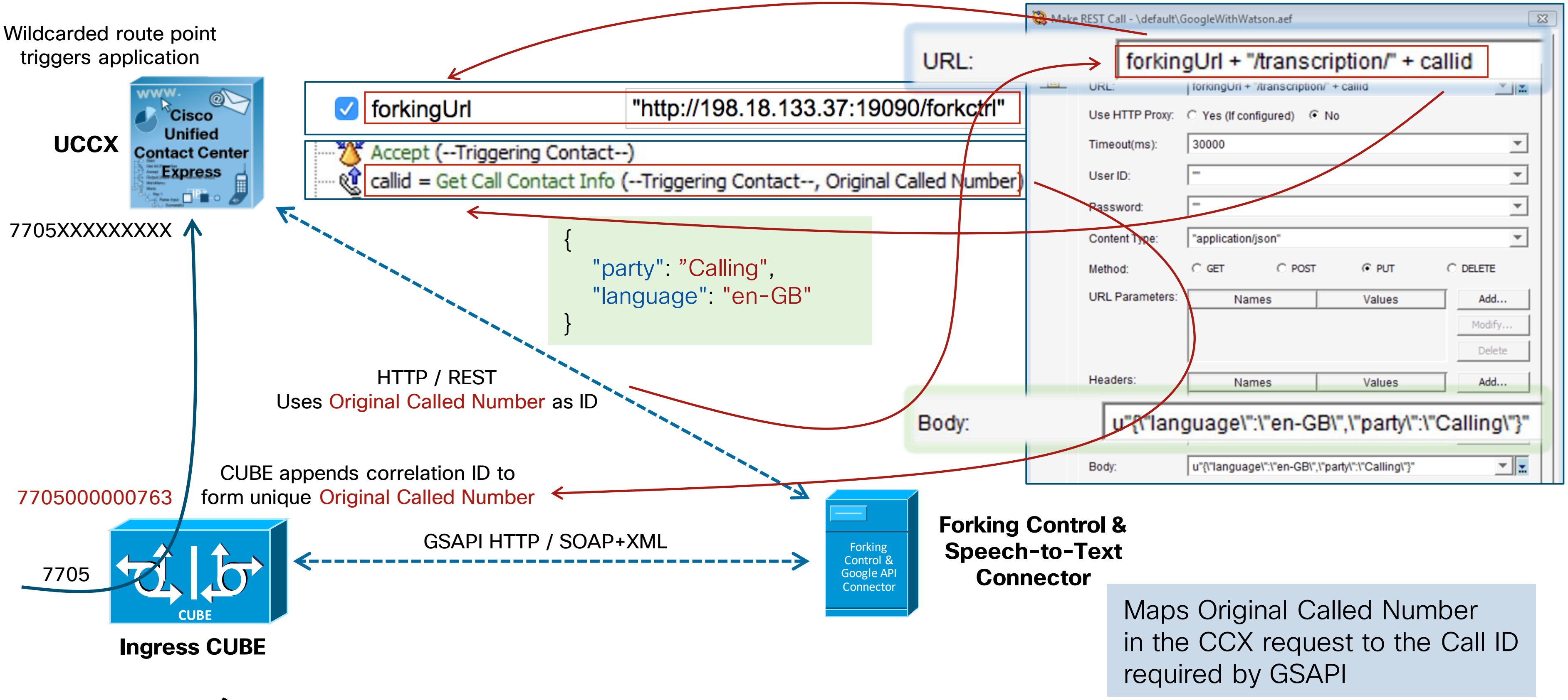
Media Forking

```
{
    "action": "START",
    "calling": {
        "address": "10.61.196.19",
        "port": "16400"
    },
    "called": {
        "address": "10.61.196.19",
        "port": "16401"
    }
}
```

CVP Using GUID To Identify Call



CCX Using Correlation ID Suffix To Identify Call



Adding Correlation ID To Dialed Number (CCX)

Why? ...

- To deliver the call with unique DNIS
- Used to lookup the gateway Call ID for use in media forking requests

How? ...

- Simple TCL application on the ingress gateway, uses call leg ID as unique ID

```
Nov 29 17:00:39.087: //763//TCL :/tcl_PutsObjCmd: APPENDLEGID, incoming call from
sip:7740449595@198.18.133.3 to 7705, forwarded to 7705000000763

Nov 29 17:00:39.227: //763//TCL :/tcl_PutsObjCmd: APPENDLEGID, event ev_connected on
call leg 764 received in state CALL_INIT

Nov 29 17:00:39.229: //763//TCL :/tcl_PutsObjCmd: APPENDLEGID, event ev_setup_done on
call leg 764 763 received in state CALL_INIT
```

Adding Correlation ID To Dialed Number (CCX)

Configuration

- Copy TCL application to gateway flash
- Add service and dial peer configuration

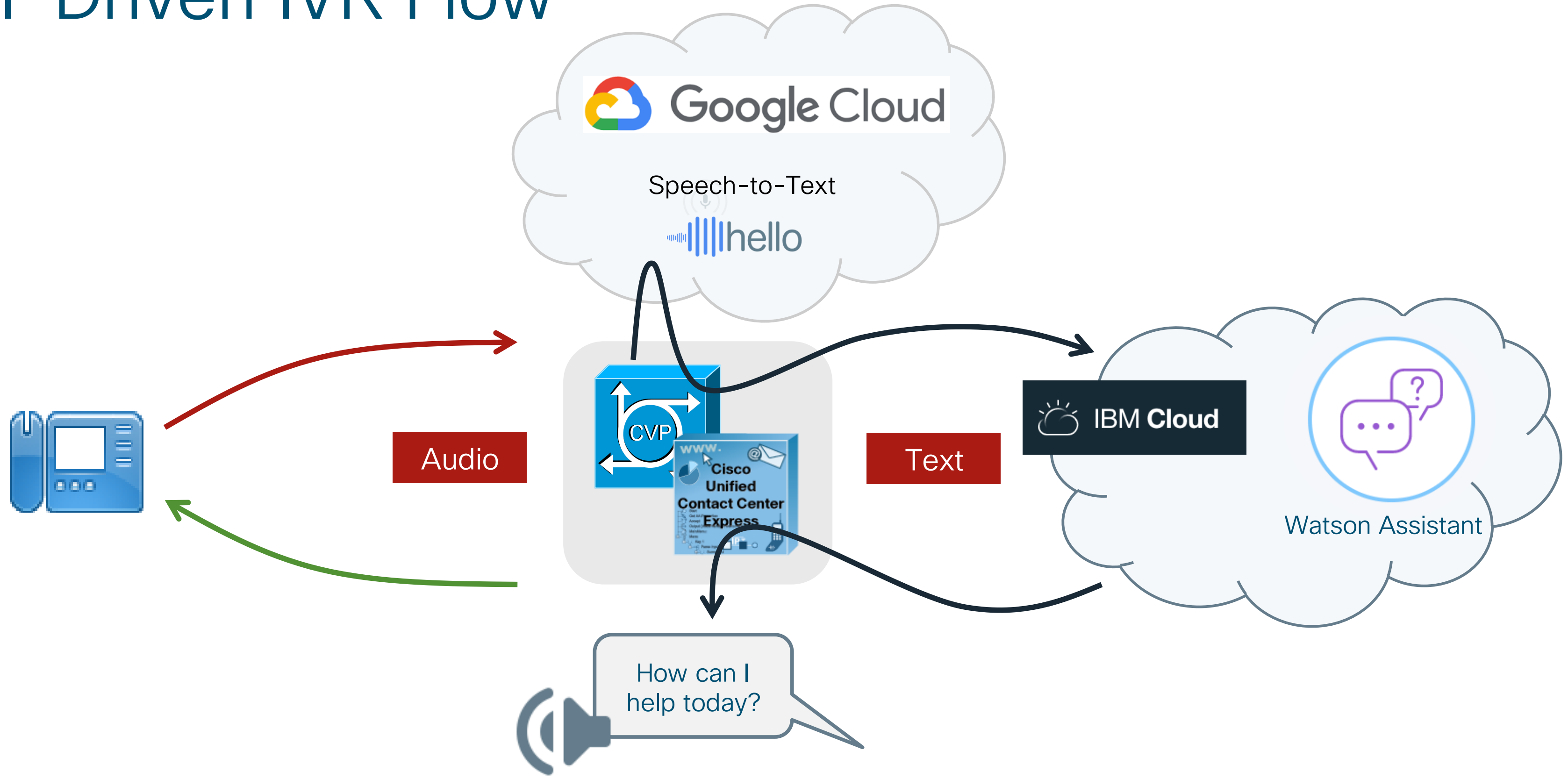
```
application
 service appendlegid flash:appendlegid.tcl
!
dial-peer voice 7700 voip
 description Inbound 770x service numbers to CCX
 service appendlegid
 destination-pattern 77T
 session protocol sipv2
 session target ipv4:198.18.133.3
 session transport tcp
 incoming called-number 770[0-6]
 dtmf-relay rtp-nte
 codec g711ulaw
 no vad
```

Define service pointing to TCL application file

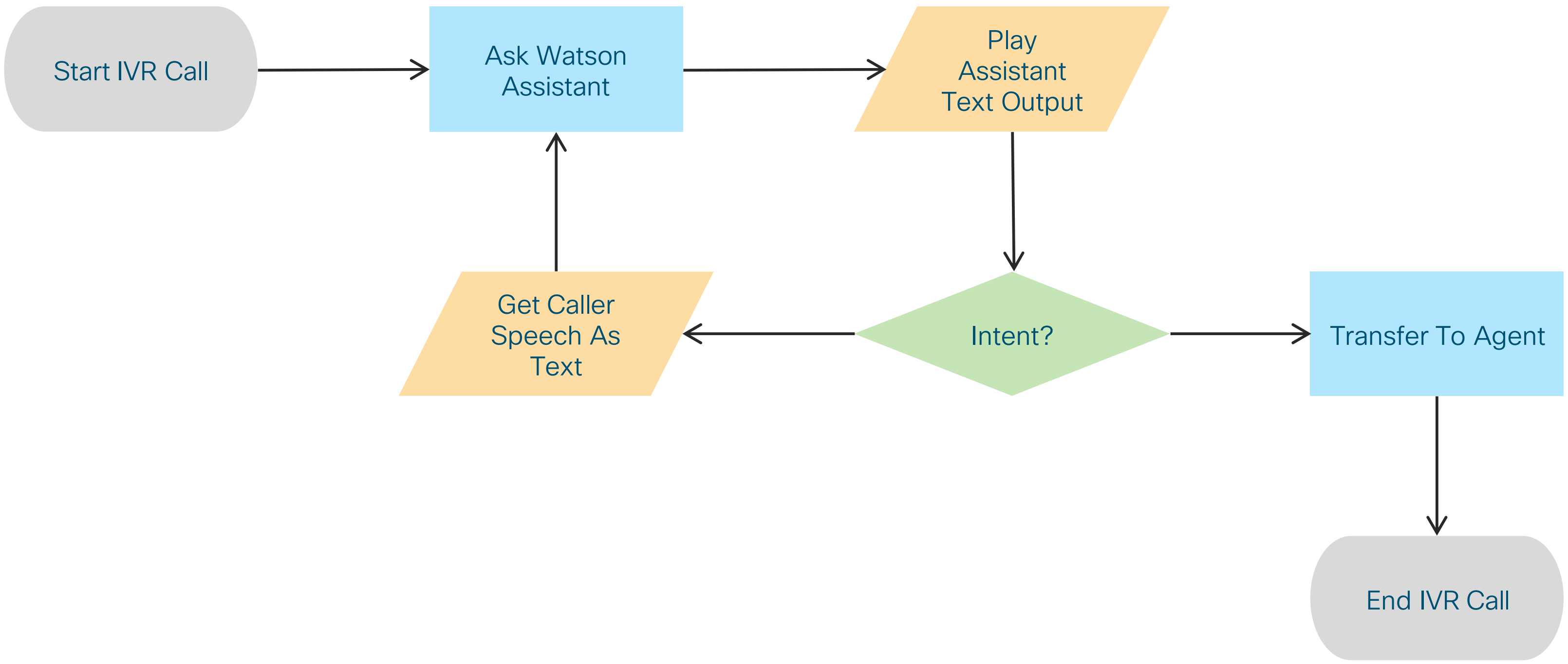
Reference the service so it's invoked on incoming calls

If the same number is delivered to multiple gateways, for uniqueness –
- Use translation rules to add extra digits per gateway
- Or, modify TCL to include gateway specific digit(s)

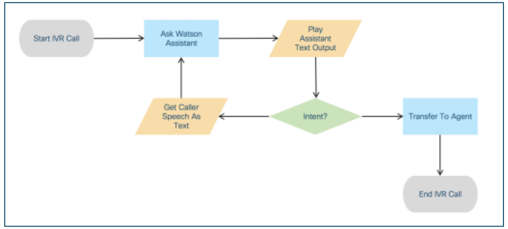
AI-Driven IVR Flow



Generic Application Flow



CCX Script




```
Start
Accept (--Triggering Contact--)
callid = Get Call Contact Info (--Triggering Contact--, Original Called Number)
Watson:
/* Send message to Watson Assistant */
Make REST Call
    Successful
        jsonResp = Create JSON Document
        ctxJSON = Get JSON Document Data (jsonResp, "$.context")
        Set convContext = (String) ctxJSON
        convOutput = Get JSON Document Data (jsonResp, "$.output.text")
        Set convMsgOutput = ("" + java.util.Arrays.asList( convOutput )).replaceAll("(^.|.$)", "").replace(", ", " ")
        convIntent = Get JSON Document Data (jsonResp, "$.intents[0].intent")
    Unsuccessful
Conversation Output:
Play Prompt (--Triggering Contact--, TTS[ convMsgOutput ] )
/* Check intent, need to exit assistant? */
If ("goodbyes".equals( convIntent ) ) Then
    True
        Goto All Done
    False
        /* Invoke transcription */
        Make REST Call
```

Send caller input (initially blank) to Watson Assistant

Extract items from the response – context data, output text and intents

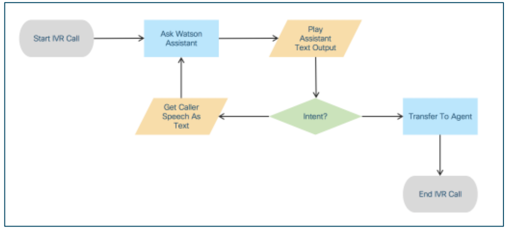
Play Assistant output using TTS

Check intents detected by Assistant

Exit Assistant phase

Otherwise get caller speech transcript

CCX Script



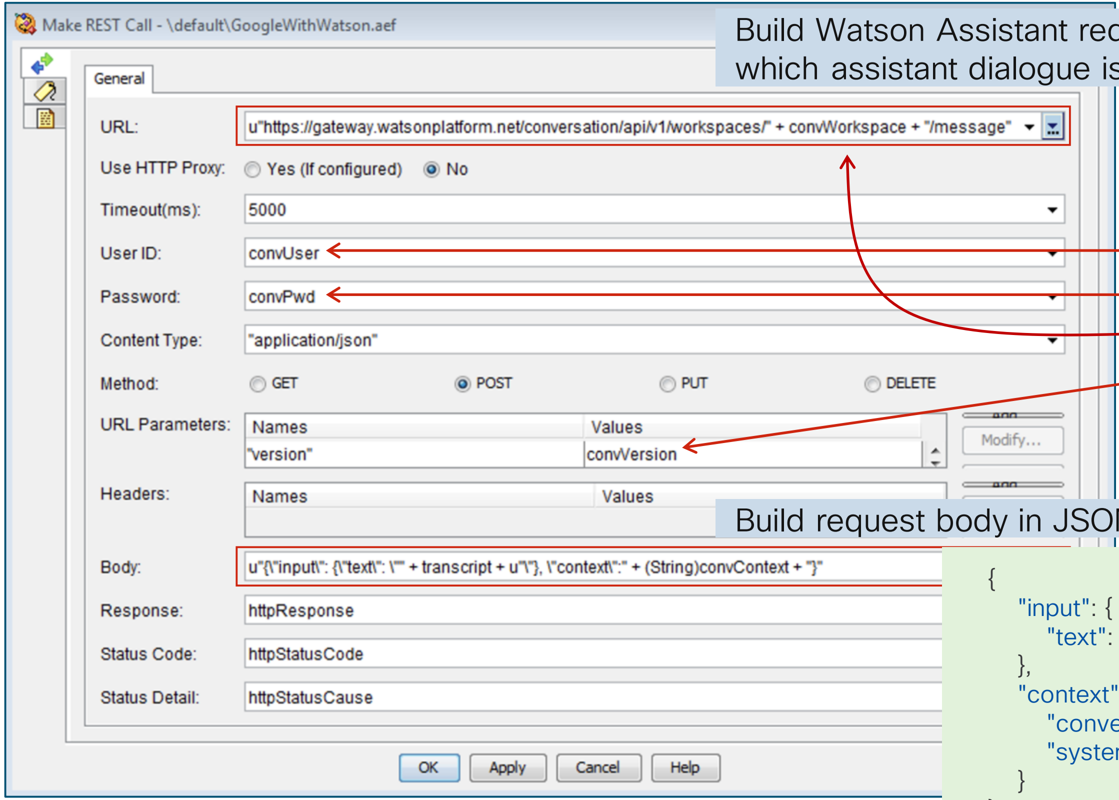

```
/* Invoke transcription */
Make REST Call
  Successful
    jsonResp = Create JSON Document
    transcript = Get JSON Document Data (jsonResp, "$.transcript")
    If (transcript == null || transcript.isEmpty()) Then
      True
      False
        Goto Watson
  Unsuccessful
All Done:
Terminate (--Triggering Contact--)
End
```

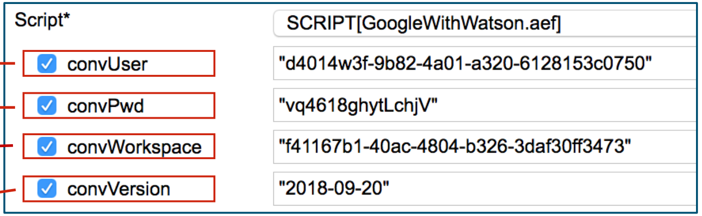
Start media forking and Google transcription

If speech to text successful, loop back to Watson Assistant input step

CCX Watson Assistant Integration

Build Watson Assistant request URL – workspace ID in path determines which assistant dialogue is invoked

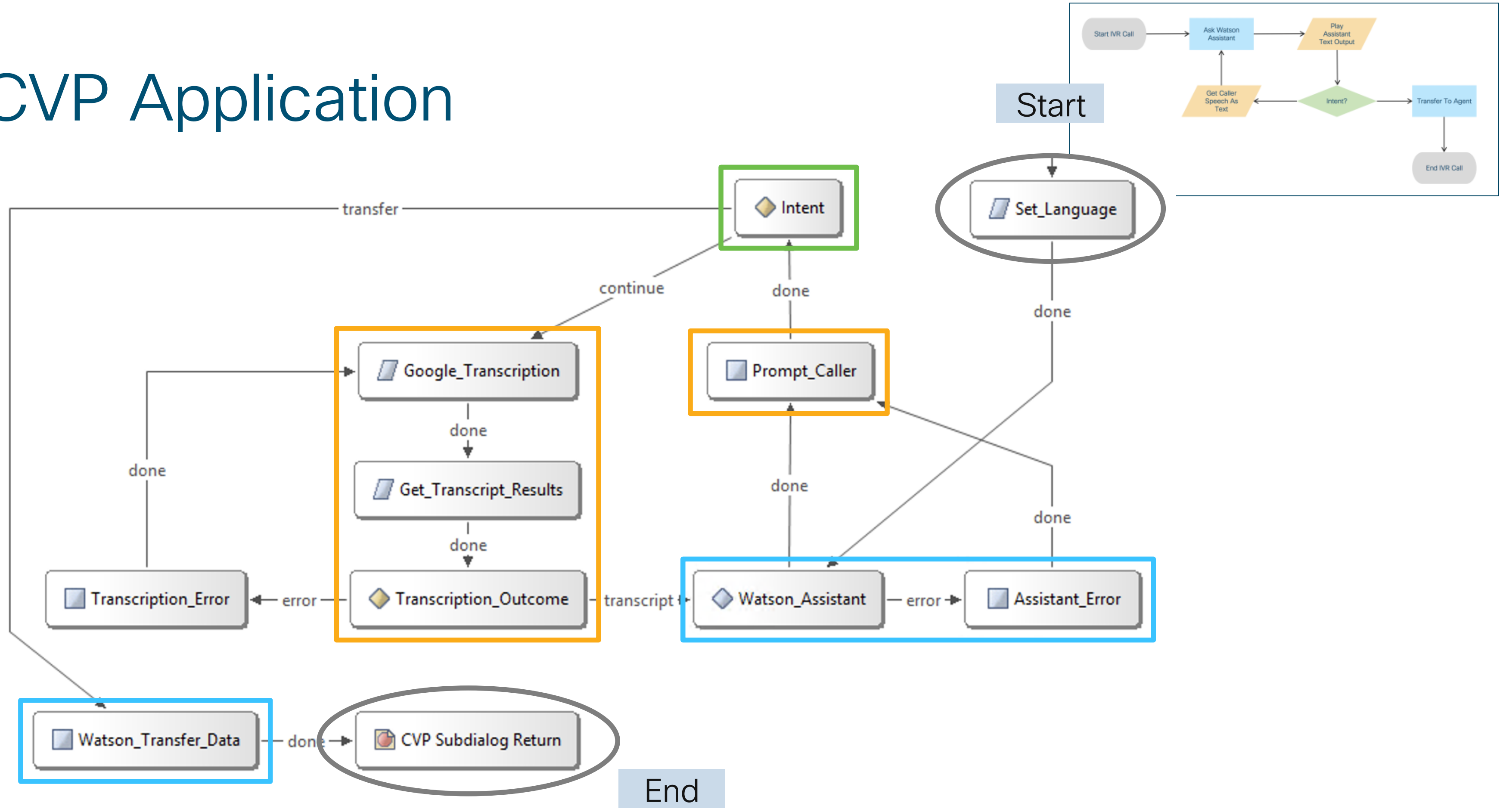




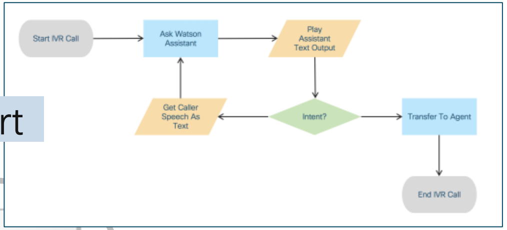
Build request body in JSON format – caller transcript passed as input.text

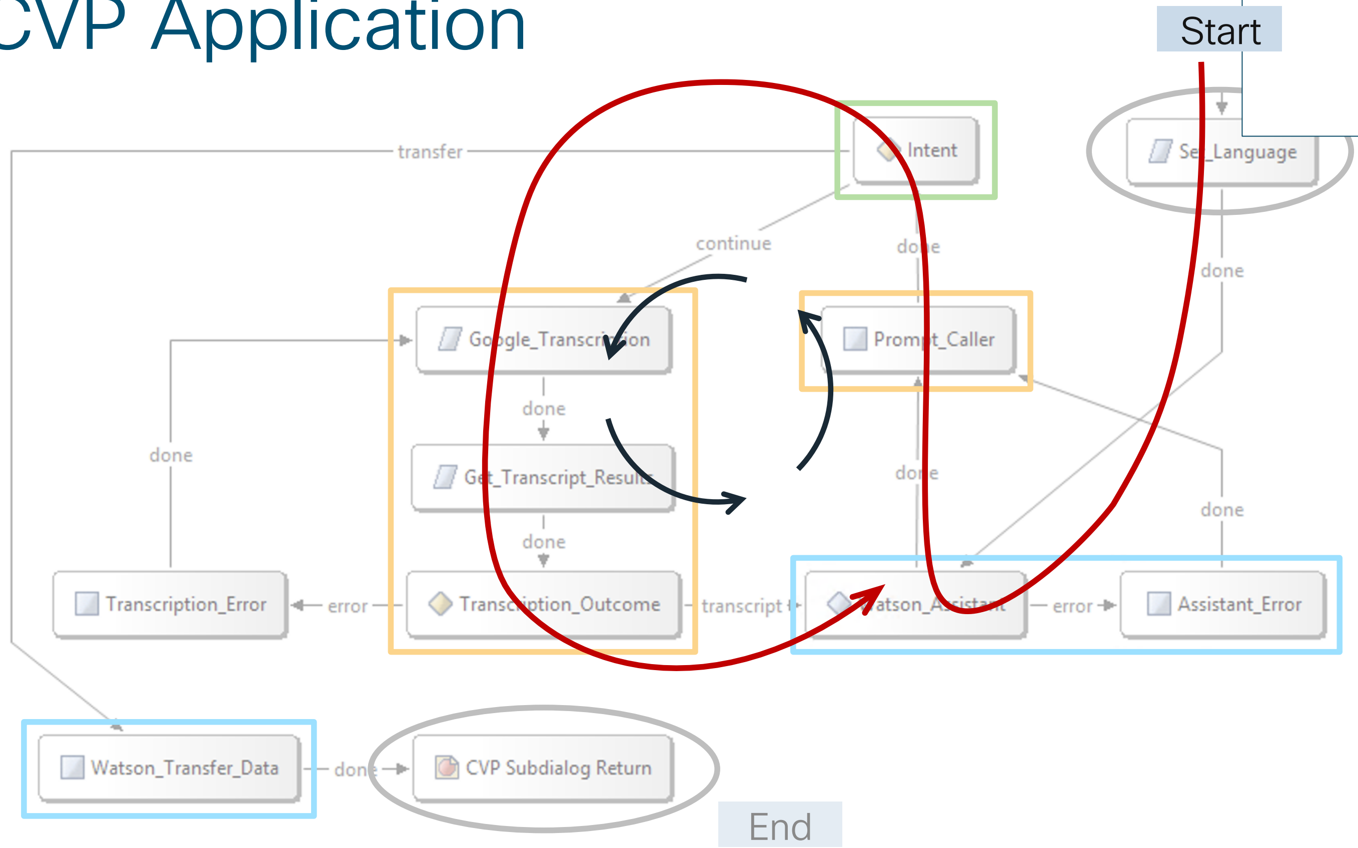
```
{
  "input": {
    "text": "Hi, I'd like to make a reservation"
  },
  "context": {
    "conversation_id": "79e7deaa-9204-471d-8cc9-c55835cb5700",
    "system": {...}
  }
}
```

CVP Application



CVP Application



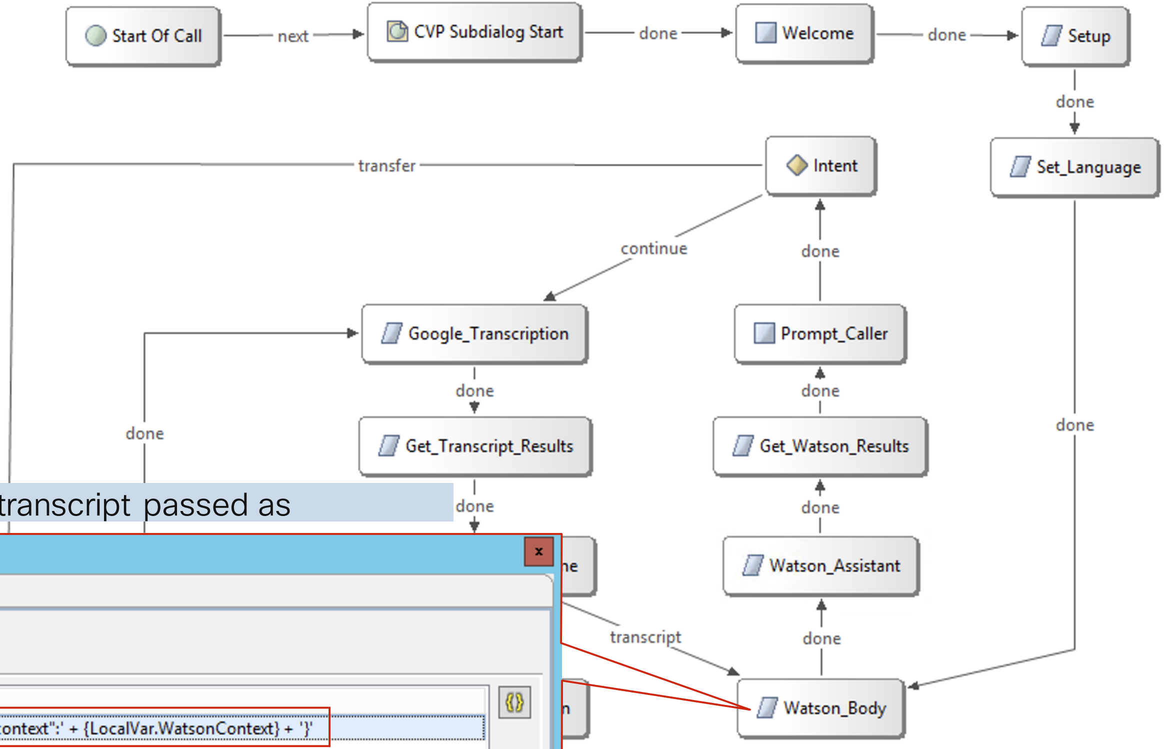


Cisco live!

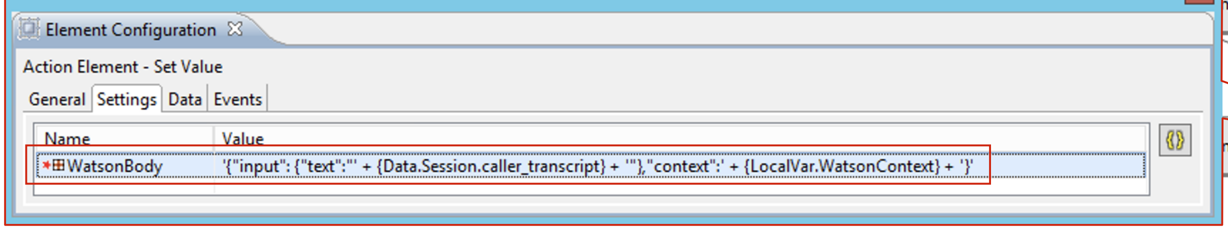
CVP Watson Assistant Integration

Use local variables that use Javascript syntax to build JSON format content.

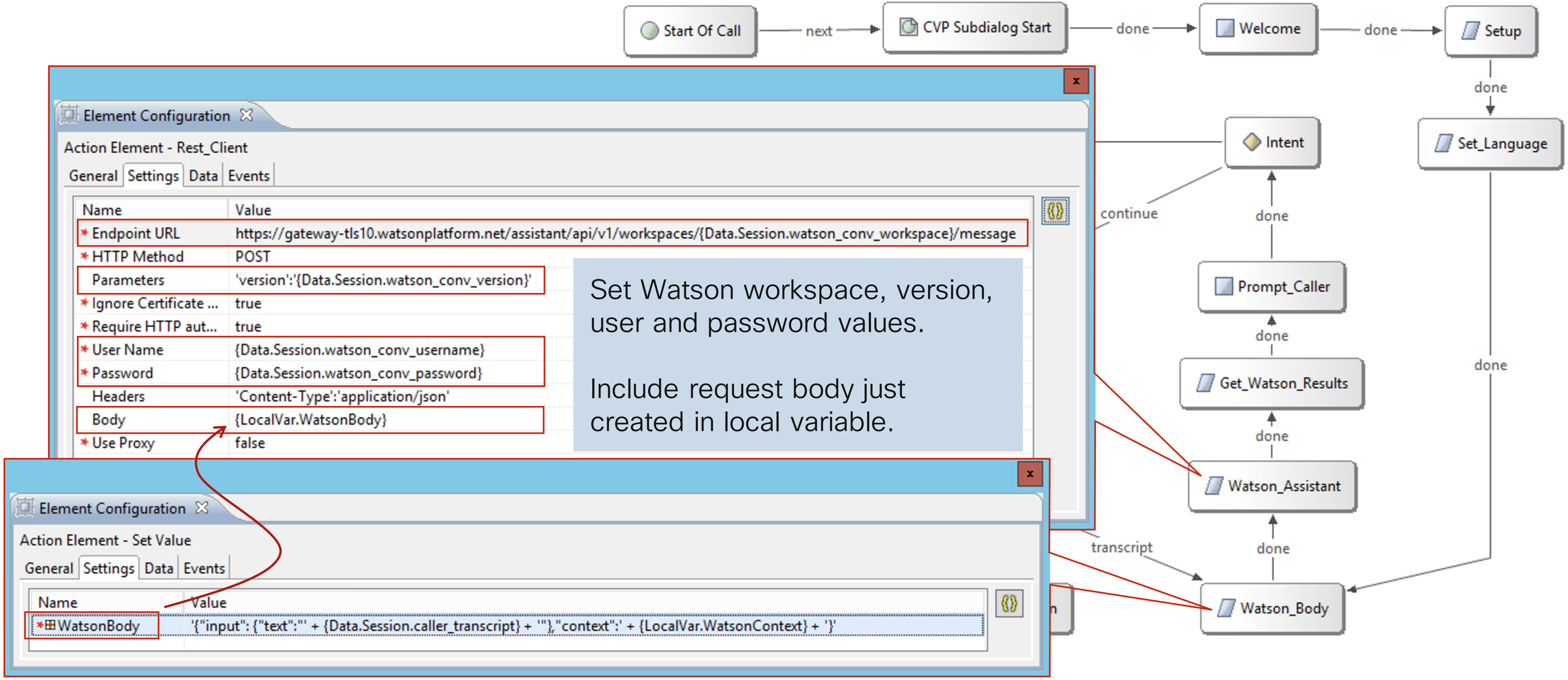
Avoids the conflict with { } you get when building element or session data using variable substitution.



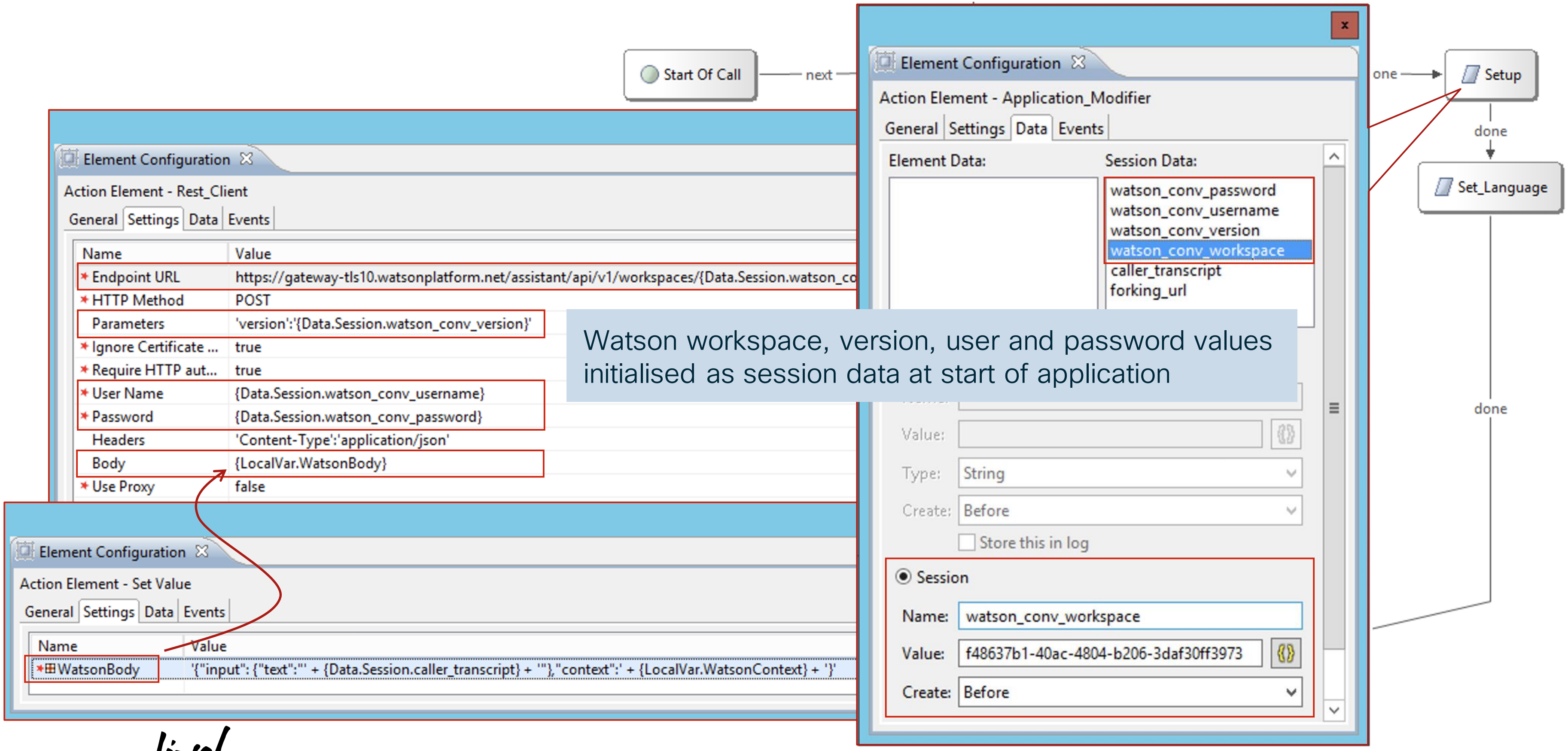
Build request body in JSON format – caller transcript passed as input.text



CVP Watson Assistant Integration

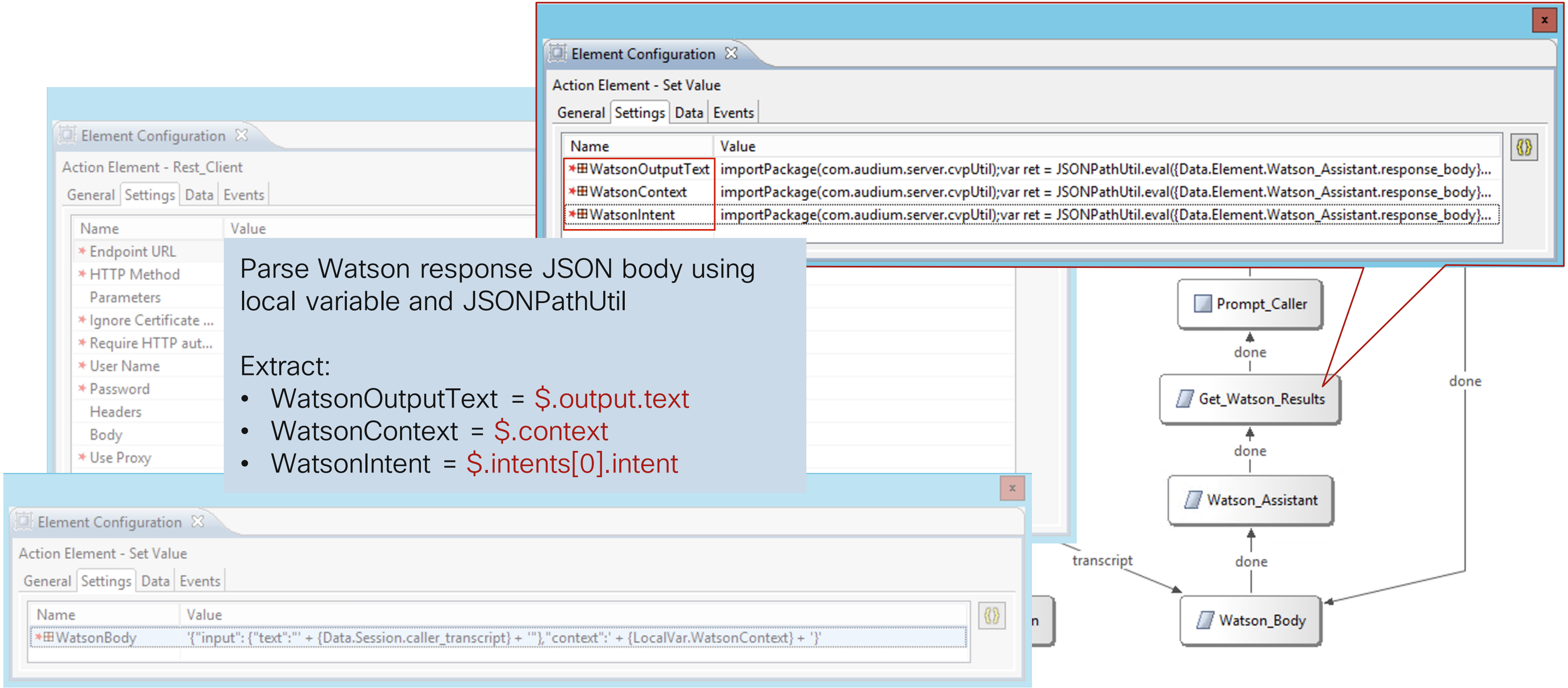


CVP Watson Assistant Integration



Watson workspace, version, user and password values initialised as session data at start of application

CVP Watson Assistant Integration



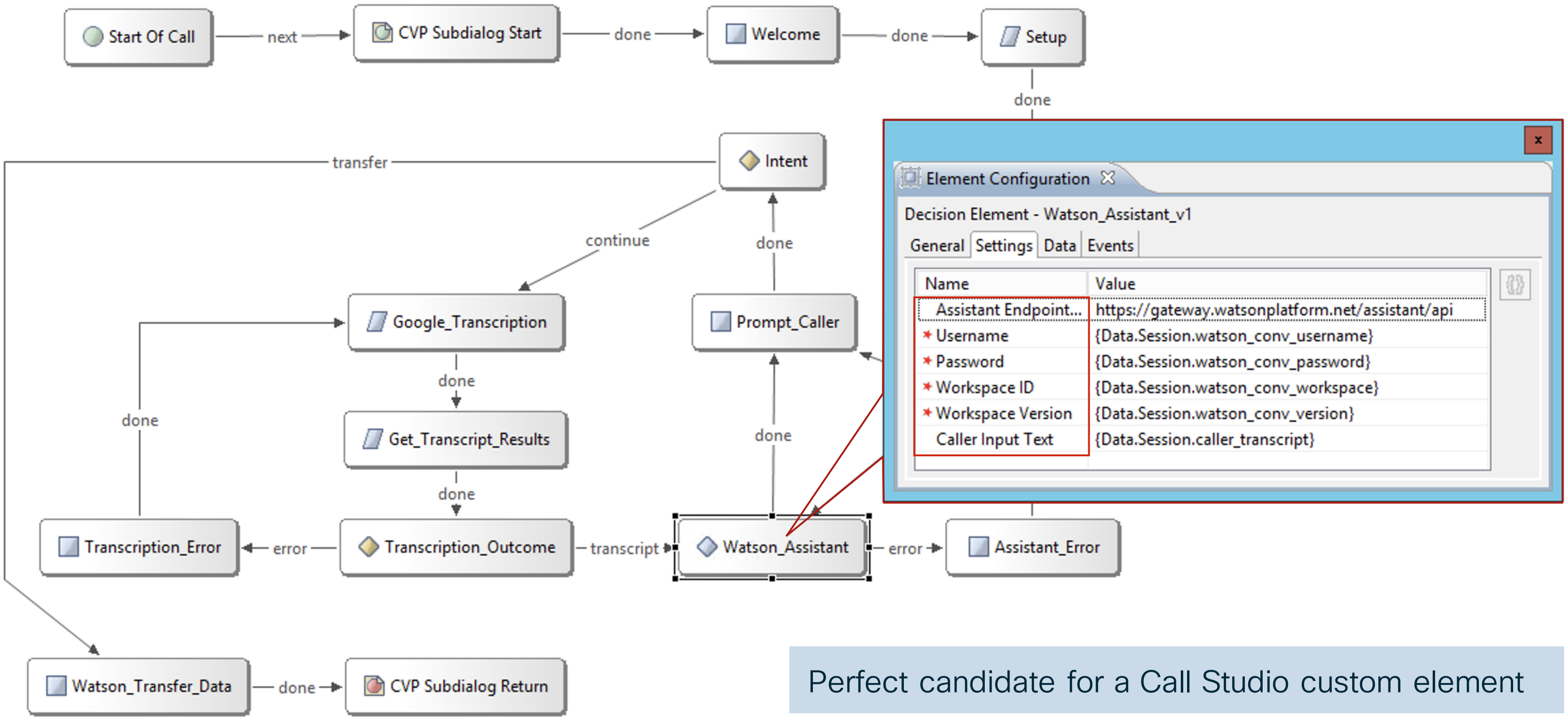
Parse Watson response JSON body using local variable and JSONPathUtil

Extract:

- WatsonOutputText = $.output.text
- WatsonContext = $.context
- WatsonIntent = $.intents[0].intent

Cisco live!

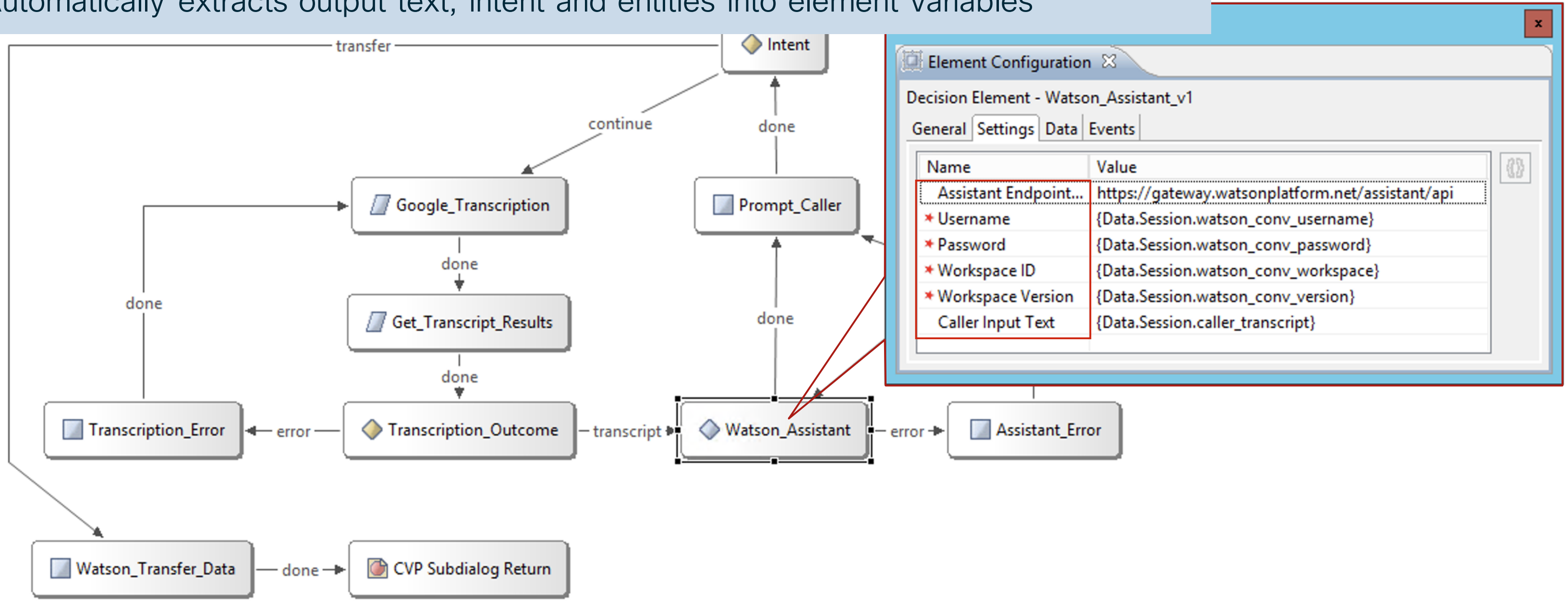
CVP Watson Assistant Integration



Perfect candidate for a Call Studio custom element

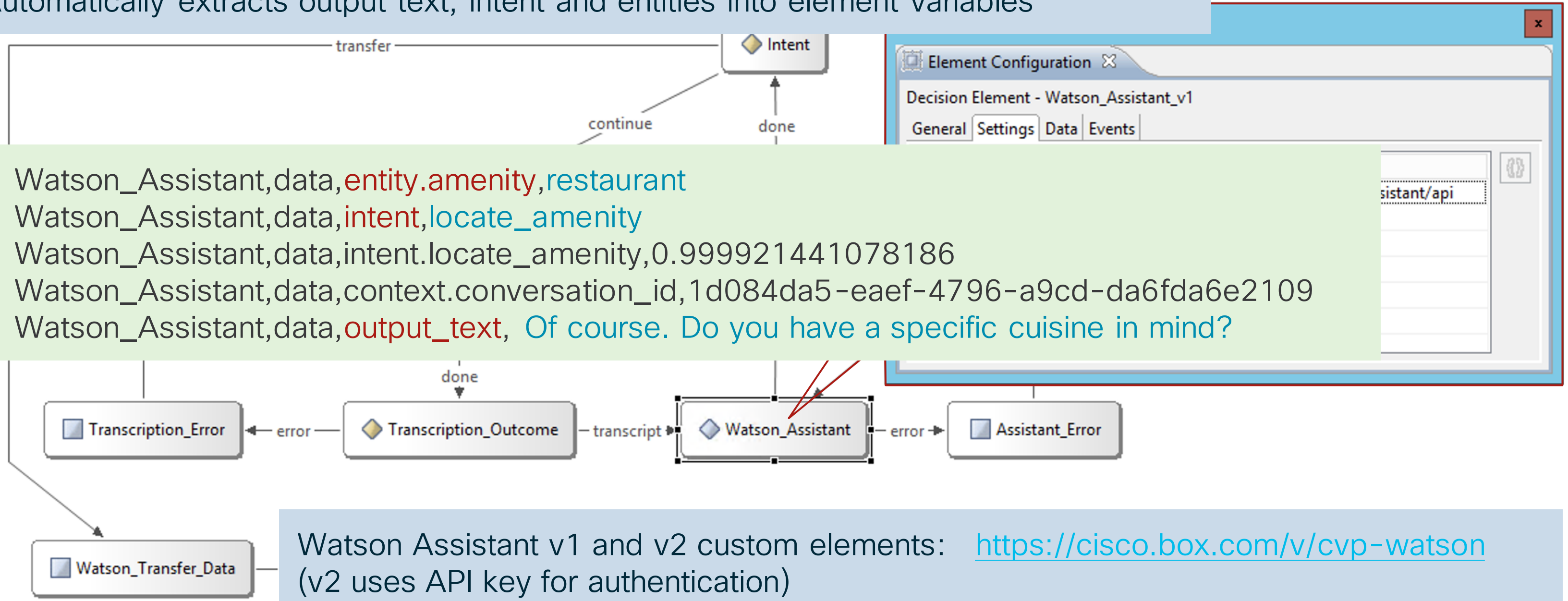
CVP Watson Assistant Integration

- Single element to send request and extract response data
- Not necessary to build JSON body or parse JSON results using Studio elements
- Maintains Watson context data invisibly in session data
- Automatically extracts output text, intent and entities into element variables



CVP Watson Assistant Integration

- Single element to send request and extract response data
- Not necessary to build JSON body or parse JSON results using Studio elements
- Maintains Watson context data invisibly in session data
- Automatically extracts output text, intent and entities into element variables



Watson_Assistant,data,entity.amenity,restaurant
Watson_Assistant,data,intent,locate_amenity
Watson_Assistant,data,intent.locate_amenity,0.999921441078186
Watson_Assistant,data,context.conversation_id,1d084da5-eaef-4796-a9cd-da6fda6e2109
Watson_Assistant,data,output_text, Of course. Do you have a specific cuisine in mind?

Watson Assistant v1 and v2 custom elements: https://cisco.box.com/v/cvp-watson
(v2 uses API key for authentication)

Demo

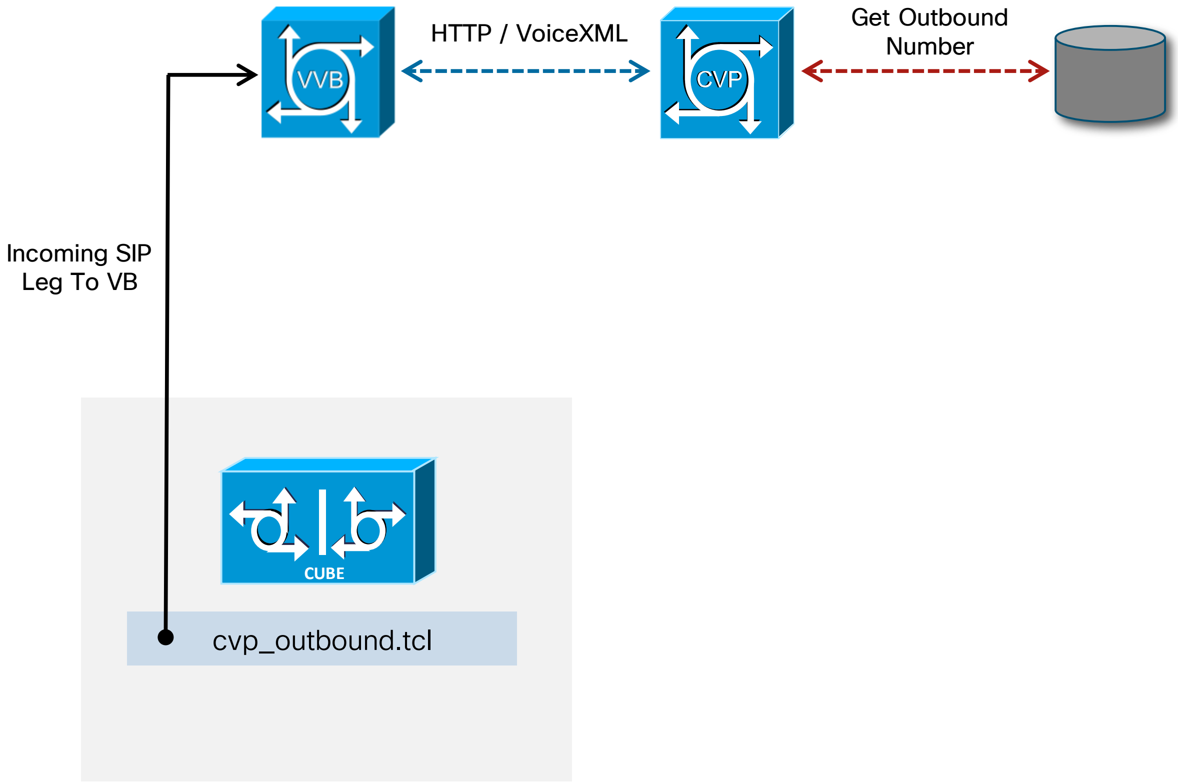
Taking It Further – Other Things Possible

Anything that can process the call participants' media stream
... or transcribed media stream

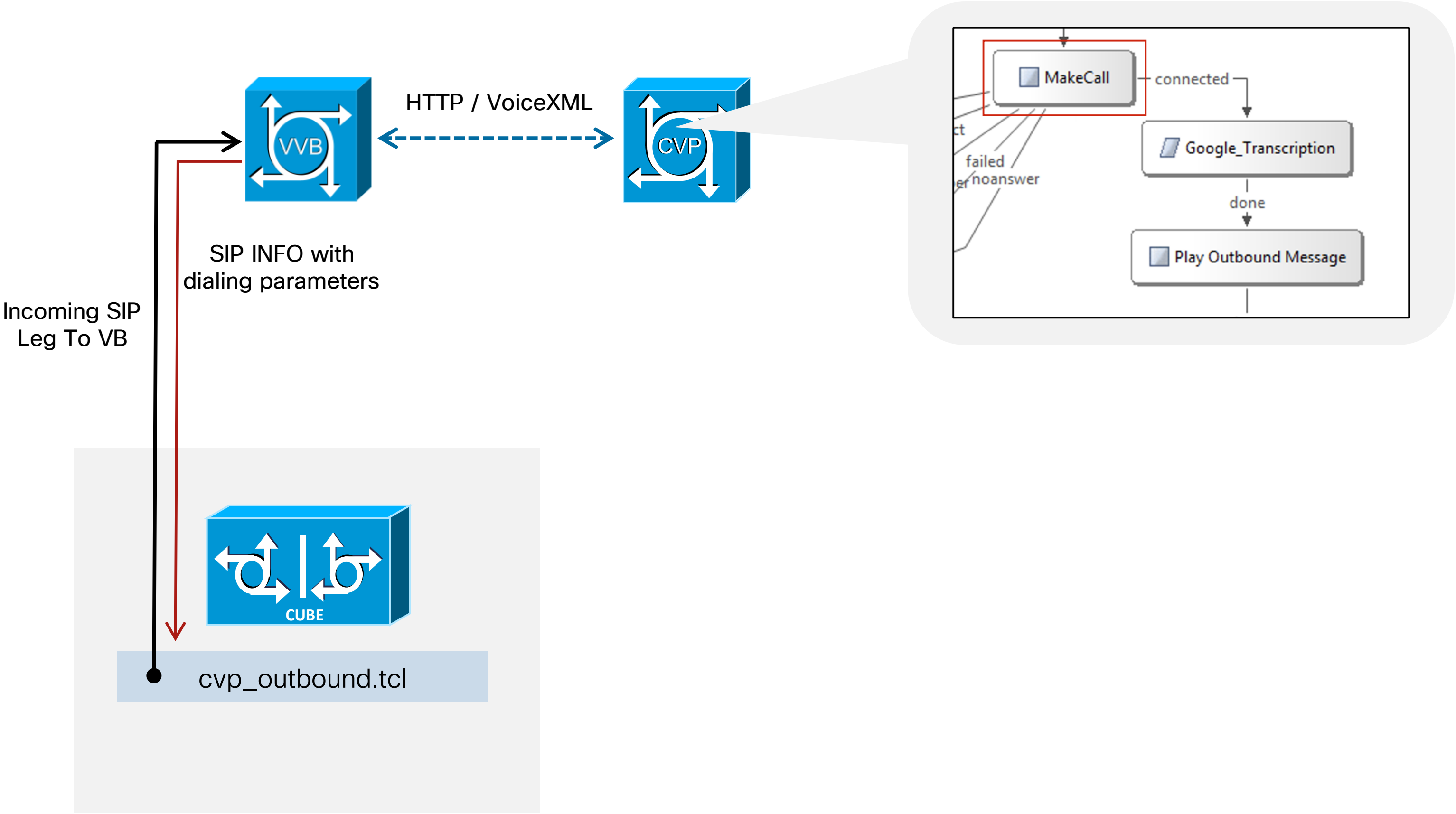
Could be IVR, agent desktop phase or both

- Sentiment analysis
- Passive voice authentication or fraud detection
- Assisted translation at the agent desktop
- Agent assist and auto suggestions
- CVP standalone outbound – silence detection and greeting analysis using AI
- Automated language detection during IVR
- Call recording snippets

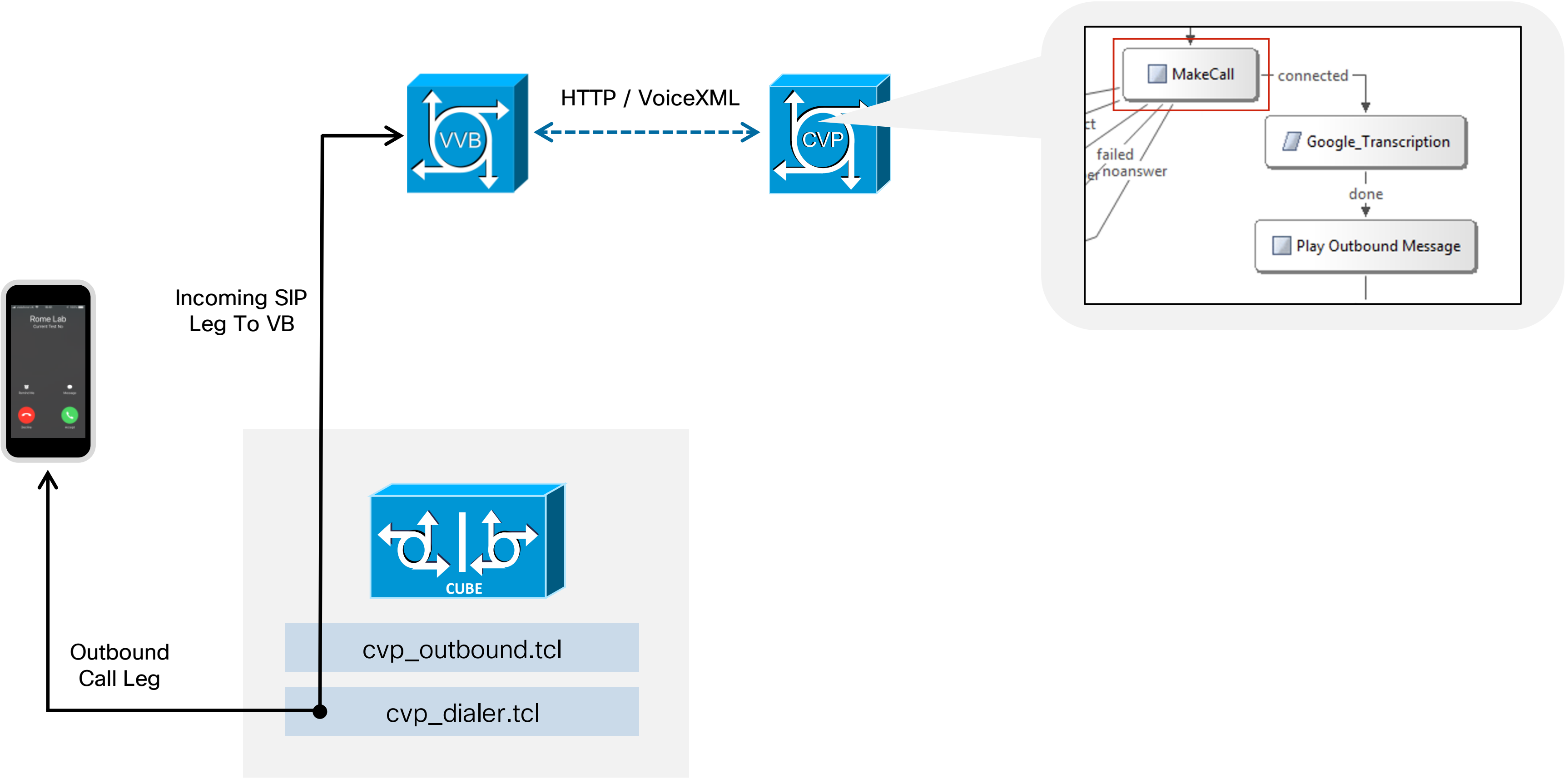
CVP Standalone Outbound Greeting Detection



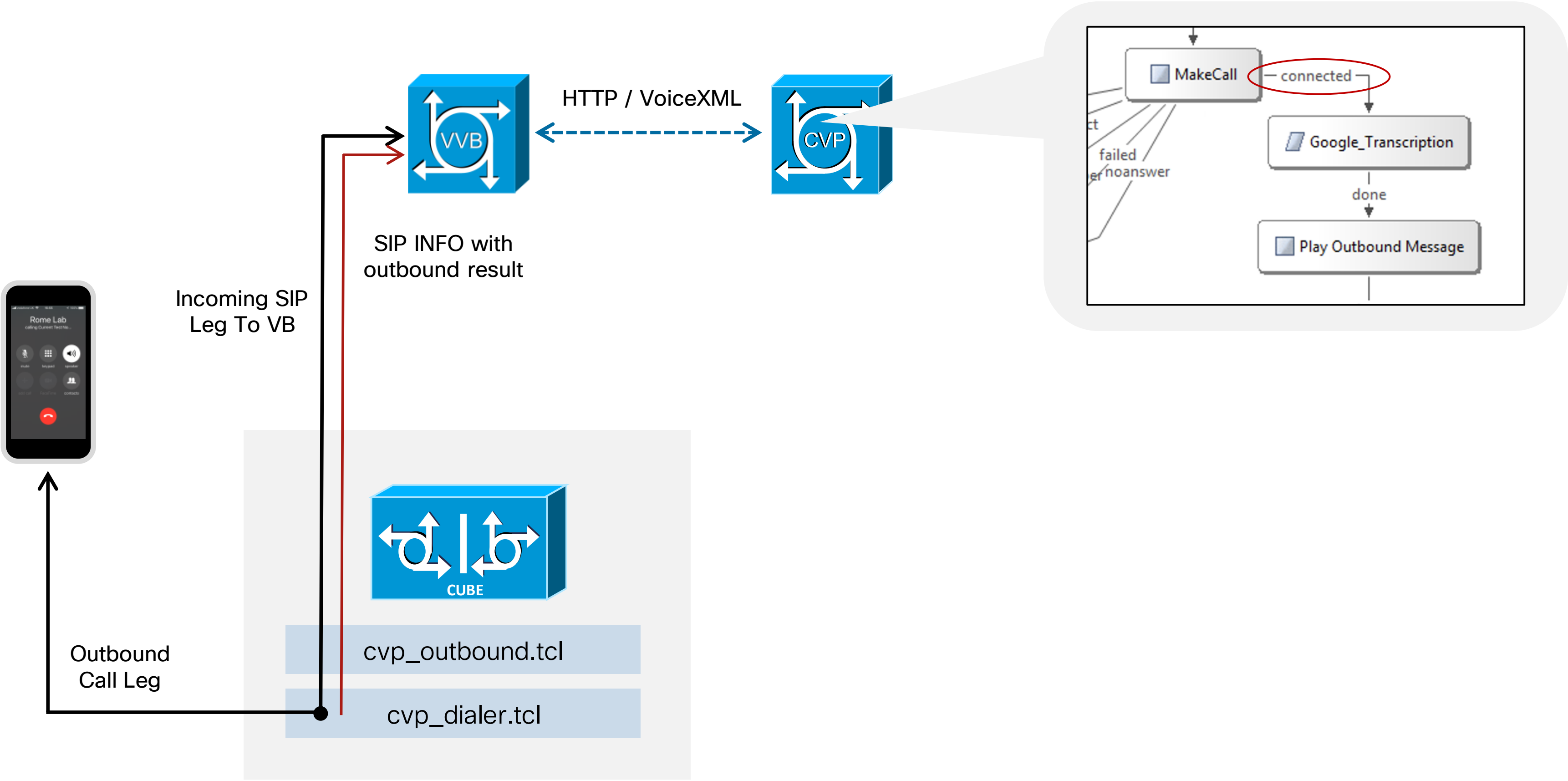
CVP Standalone Outbound Greeting Detection



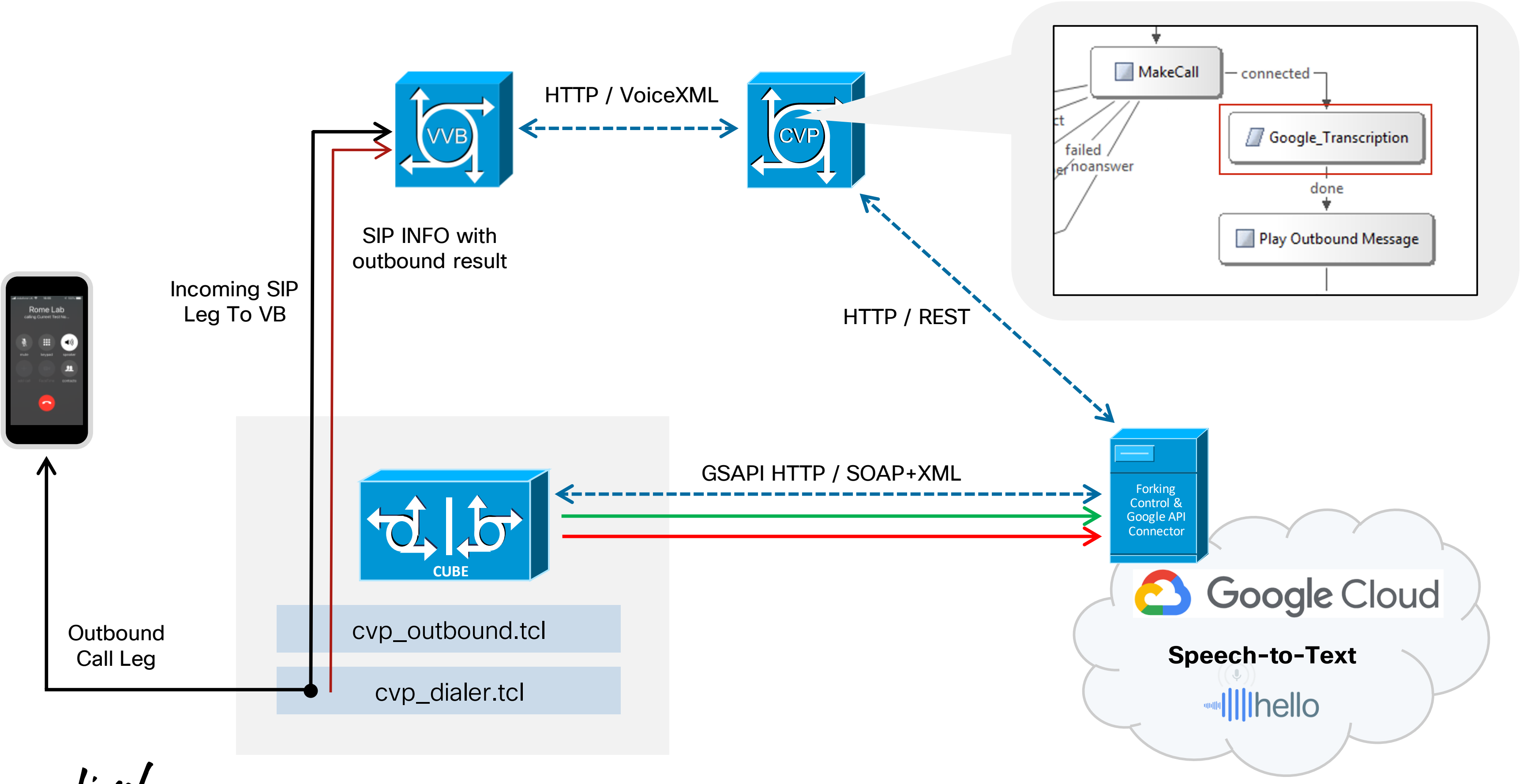
CVP Standalone Outbound Greeting Detection



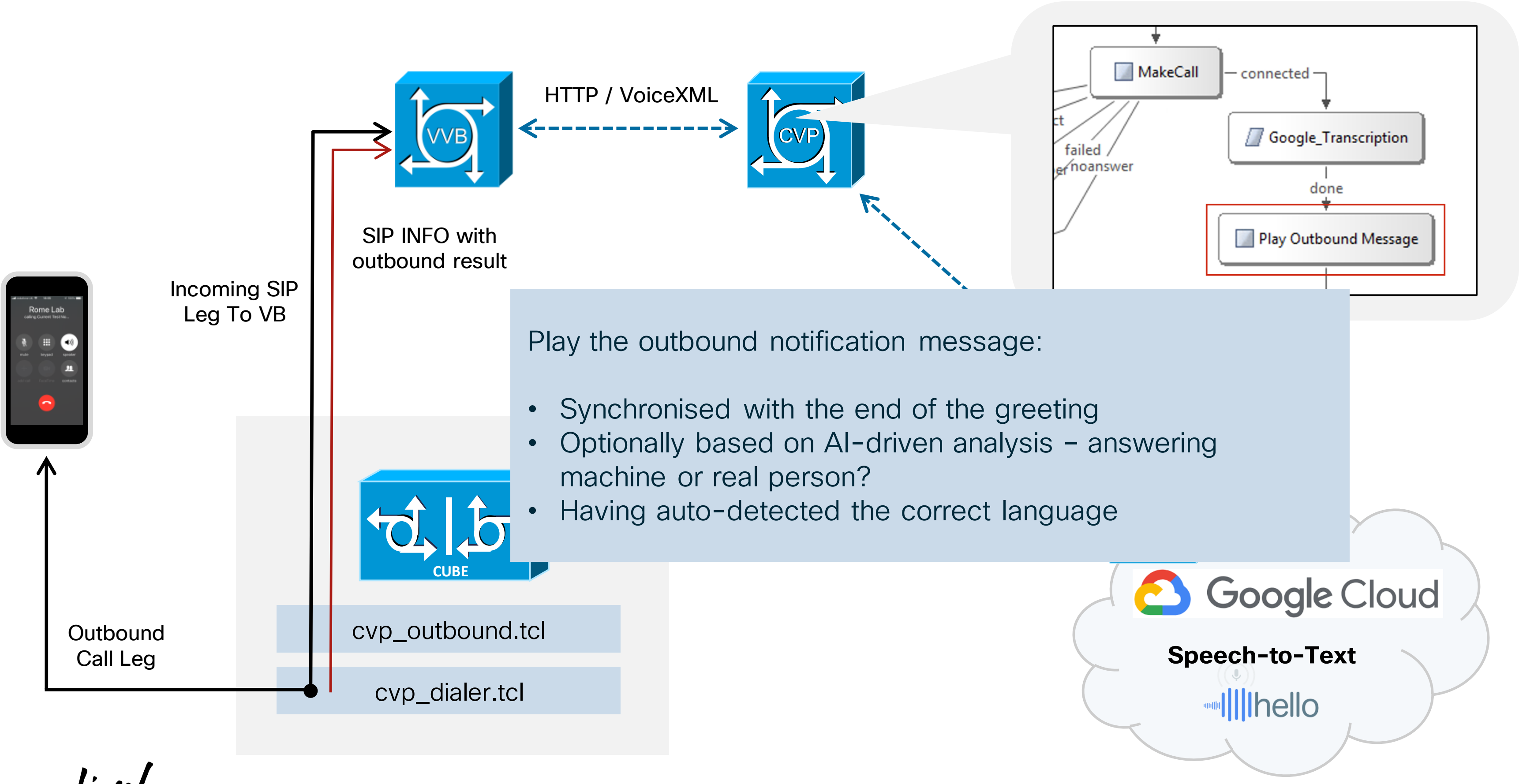
CVP Standalone Outbound Greeting Detection



CVP Standalone Outbound Greeting Detection



CVP Standalone Outbound Greeting Detection

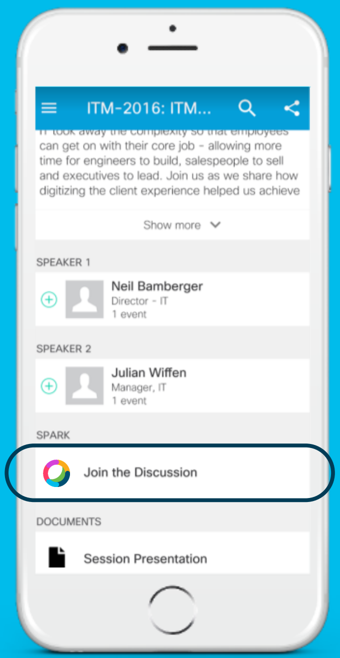


Play the outbound notification message:

- Synchronised with the end of the greeting
- Optionally based on AI-driven analysis – answering machine or real person?
- Having auto-detected the correct language

Getting Started Links

- Things from the Tindall workbench
  - Forking connector https://cisco.box.com/v/google-transcription
  - Watson CVP elements https://cisco.box.com/v/cvp-watson
  - Twitter @tindallpaul to catch anything that’s new / updated
- Gateway Services API
  - Documentation Cisco UC Gateway Services API Guide
- Google Speech To Text API
  - https://cloud.google.com/speech-to-text/docs/streaming-recognize



cs.co/ciscolivebot#BRKCCT-2541

Cisco Webex Teams

Questions?

Use Cisco Webex Teams (formerly Cisco Spark) to chat with the speaker after the session

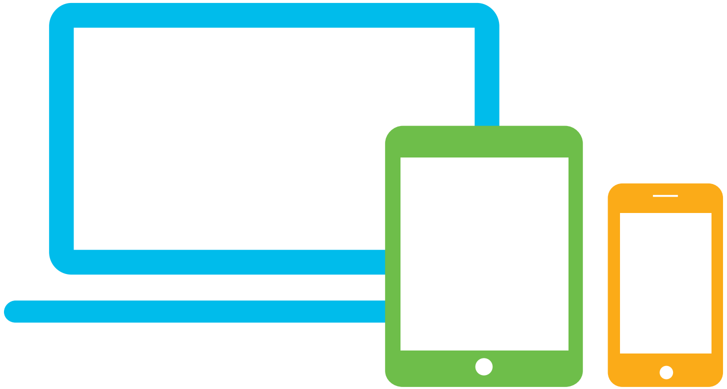
How

1. Find this session in the Cisco Events Mobile App
2. Click “Join the Discussion”
3. Install Webex Teams or go directly to the team space
4. Enter messages/questions in the team space

Complete your online session survey

- Please complete your Online Session Survey after each session
- Complete 4 Session Surveys & the Overall Conference Survey (available from Thursday) to receive your Cisco Live T-shirt
- All surveys can be completed via the Cisco Events Mobile App or the Communication Stations

Don’t forget: Cisco Live sessions will be available for viewing on demand after the event at ciscolive.cisco.com



Cisco live!

Continue Your Education

Demos in the Cisco Showcase

Walk-in self-paced labs

Meet the engineer 1:1 meetings

Related sessions

Thank you

INTUITIVE

IMAGINE

INTUITIVE