The bridge to possible

Intelligently Handling Call Traffic Between Premise & Cloud Contact Center

Paul Tindall
@tindallpaul

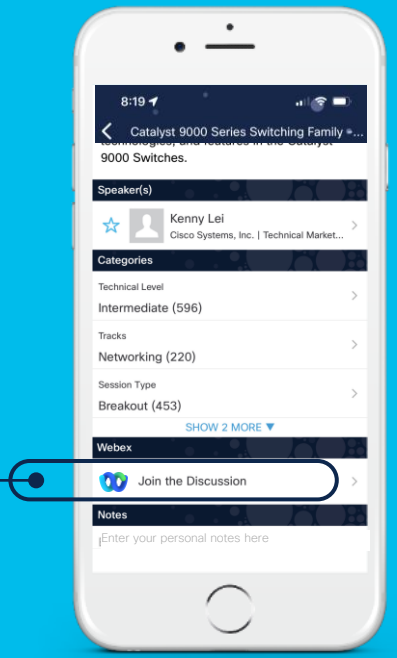
Cisco Webex App

Questions?

Use Cisco Webex App to chat
with the speaker after the session

How

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



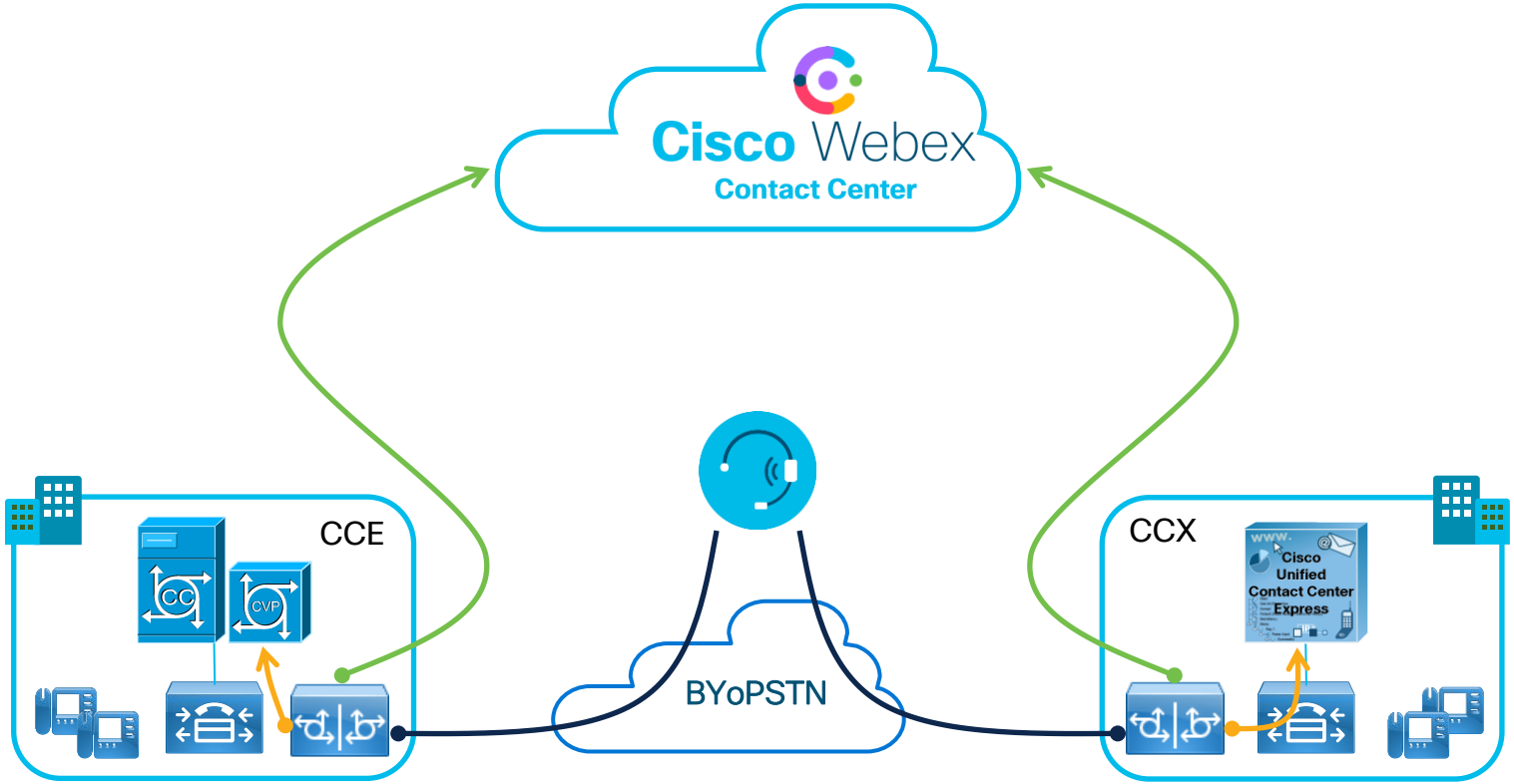
Webex spaces will be moderated until February 24, 2023.

Agenda

- Examine the use case and understand the challenge
- How to insert active routing decisions
- See it working
- Taking hybrid call flows further
- Try it for yourself

The Use Case

Premise and Cloud Contact Center in Parallel



When Is This Required?

- Integrating a new location
- Expanding existing operation
- Traffic overflow handling
- Starting migration from premise to cloud
- Bring Your Own PSTN to cloud CC, call rejection and fallback

Why Not Just Throw The Switch?

- Controlled at CUBE, local gateway dial-peer level
- Could distribute calls using dial-peer configuration
- Can use max-conn per dial-peer to apply limits
- Manage using static changes on local CUBE/gateway
- Gets ugly across multiple ingress devices
- Cannot intelligently route using current state of contact centres

Active Routing Decisions

The End Goals

- Route to premise or cloud based on current state
  - Could be queue size, agents signed-on/available, wait times, open hours
  - All the typical parameters that you use to make CC routing decisions
- Preferred routing for different services
- Not send to cloud at all, reject calls or play announcements locally
  - Avoid Webex CC surge protection rejecting calls
  - Apply your own traffic level controls

Deconstruct The Problem

Four main areas for consideration

1. The data that feeds decision making
2. Intercepting calls on premise and triggering the decision
3. Where to consume data and make routing/treatment decisions
4. Controlling the call and implementing the decision

Getting Real-Time Data From Webex Contact Center

cisco Live!

Data Input To Routing Decisions

- On-premise CC real-time state accessible to routing scripts
- But, what about cloud?

Webex Contact Center Developer Portal

webex Contact Center for Developers

OVERVIEW

Getting Started

Authentication

Common API Errors

Integrations

Introduction to APIs

Getting Started

The Webex Contact Center open platform and APIs enable you to access the Webex Contact Center platform programmatically to customize, fine tune and build complimenting integrations. Developers can build, enhance and customize their Customer Experience solution with the rich set of APIs, that includes Contact Center, AI, Journey, Orchestration and Experience Management. Getting started with the APIs is easy. We provide detailed API reference docs as well as the Try Out functionality which help to familiarize you with our APIs and get started quickly.

https://developer.webex-cx.com/

https://devportal.wxcc-eu1.cisco.com/

Routing Decisions Need Real-Time State Info

- API to retrieve EWT by Queue ID
- Sounds right for the job
- But is it?

webex Contact Center for Developers

Dial Number
Dial Plan
Entry Point
Estimated Wait Time
Multimedia Profile
Outdial ANI
Queues

Estimated Wait Time > Get Estimated Wait Time

Version 1

Get Estimated Wait Time

GET /v1/ewt

Retrieve Estimated Wait Time information for a specified look back interval for a specific orgId and queueId combination, with ability to customomize maxCV and minValidSamples (See description above).

Retrieve Estimated Wait Time information for a specified look back interval for a specific orgId and queueId combination, with ability to customomize maxCV and minValidSamples (See description above).

Use EWT API? Not For This Use Case

Why not?

- EWT value returned is statistical value derived from recent history
- No guarantee you will get a value returned
- Still have to make a decision, but on some other basis
- Sub-optimal approach for call-by-call requests due to throttling

Use EWT API? Not For This Use Case

Rate Limiting

Our APIs are protected with rate limiting. You are responsible for architecting and coding your application responsibly so as to not create too much traffic.

Response Format

Every rate limited request will have a `429` response with a `retry-after` header included. This means your request has not been processed. Do not try again until after the `retry-after` duration (in seconds) has passed, else you will receive another `429` response. For example, a 4 second wait time will look like so:

Architectural

If you are polling an API to find changes, consider subscribing to webhooks instead.

Let's Consider Webhooks

- An HTTP endpoint to listen for and receive asynchronous events
- Allows a custom application to track Webex CC state in real-time

webex Contact Center for Developers

Multimedia Profile
Outdial ANI
Queues
Search
Site
Skill
Skill Profile
Subscriptions

Using Webhooks

Webhooks enable you to receive notifications and react to events occurring within Webex Contact Center.

Use Cases

Webhooks form a great system for working with near-real-time data, and for reacting to Contact Center events using custom logic. If you want to stay on top of a busy contact center, or have an application which can't wait for slow batch reports or expensive API polling-and-diffing solutions, then you're in the right place.

Let’s Consider Webhooks

- An HTTP endpoint to listen for and receive asynchronous events
- Allows a custom application to track Webex CC state in real-time

webex Contact Center for Developers

Multimedia Profile
Outdial ANI
Queues
Search

Using Webhooks

Webhooks enable you to receive notifications and react to events occurring within Webex Contact Center.

Webhooks form a great system for working with near-real-time data, and for reacting to Contact Center events using custom logic. If you want to stay on top of a busy contact center, or have an application which can’t wait for slow batch reports or expensive API polling-and-diffing solutions, then you're in the right place.

Using Webhooks To Track State

1. Authenticate API user to get access token and refresh token
   - Follow instructions on Developer Portal
   - Save refresh token for use by application later
   - Need access token in Authorization header on every API request
2. Subscribe to agent and task notifications
   - Tell Webex CC what events we want
   - Can do subscription using the Developer Portal
   - Persistent, set and forget, until you need to update or delete later
   - Be aware, there is a limit on the number of concurrent subscriptions

Using Webhooks To Track State

1. Authenticate API us
   - Follow instructions
   - Save refresh toke
   - Need access toke
2. Subscribe to agent
   - Tell Webex CC wh
   - Can do subscripti

later

riptions

webex Contact Center for Developers

Multimedia Profile
Outdial ANI
Queues
Search
Site
Skill
Skill Profile
Subscriptions

Subscriptions > Register Subscription

Version 1

Register Subscription

POST /v1/subscriptions

Create a subscription which would allow consumers to listen to events. If creating a subscription causes the org-level limit to be exceeded, the subscription registration will be denied. Requires `cjp:config_write` scope.

Create a subscription which would allow consumers to listen to events. If creating a subscription causes the org-level limit to be exceeded, the subscription registration will be denied. Requires `cjp:config_write` scope.

Agent & Task Event Subscriptions

POST to /v1/subscriptions

JSON body specifying –
- which events of interest
- where to send them

```
{
  "name": "Task Events",
  "description": "Track task state changes",
  "eventTypes": [
    "task:new",
    "task:parked",
    "task:connect",
    "task:connected",
    "task:ended"
  ],
  "destinationUrl": "https://pcce.vpod1628.dc-01.com/wxcc/event/task",
}
```

Selected task events

Custom app webhook URL for task events

```
{
  "name": "Agent Events",
  "description": "Track agent state changes",
  "eventTypes": [
    "agent:login",
    "agent:logout",
    "agent:state_change"
  ],
  "destinationUrl": "https://pcce.vpod1628.dc-01.com/wxcc/event/agent",
}
```

Selected agent events

Custom app webhook URL for agent events

Consuming State Change Notifications

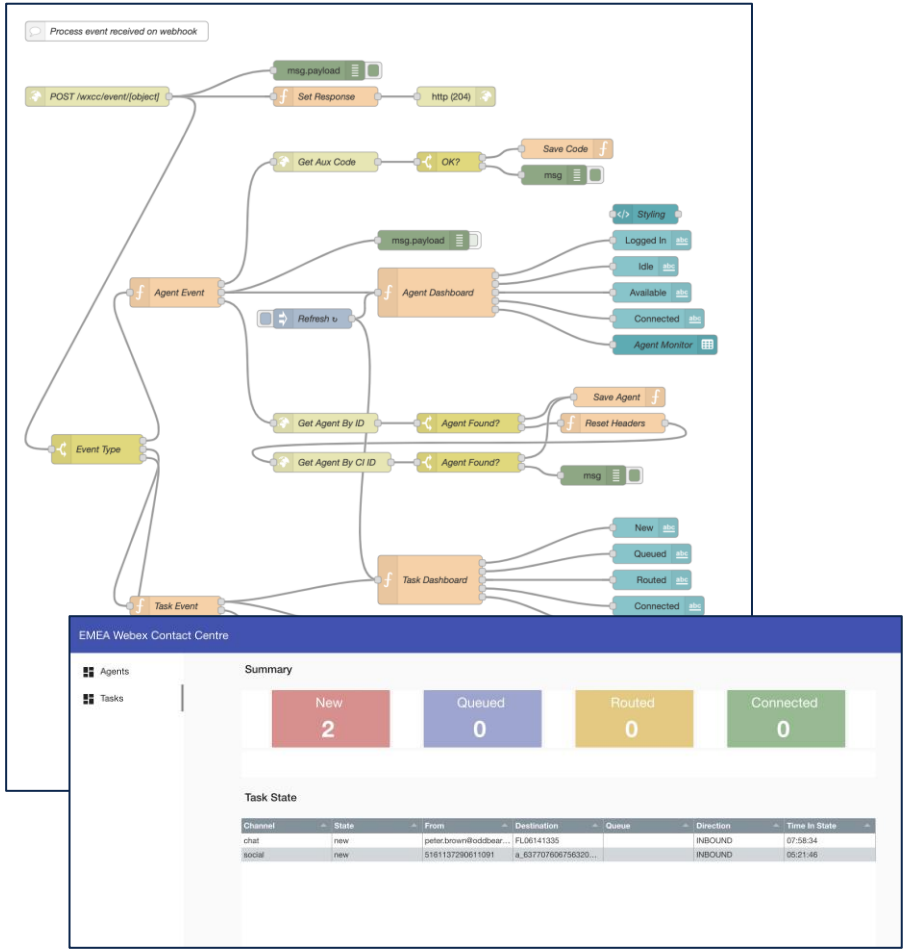
Three questions ...

1. How do we expose the receiving webhooks?
2. How do we process the notifications when they arrive?
3. Where does all this happen architecturally?

Answer: In a custom web application

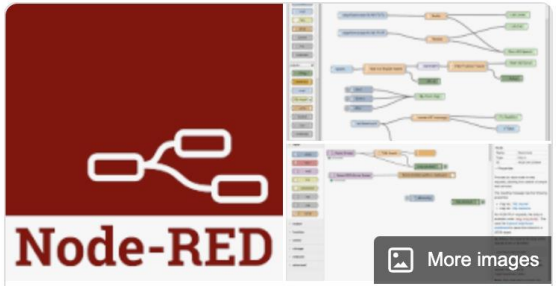
Don’t Panic!

- Sample for you to download
- It’s a Node-RED flow
- Deploy easily and quickly
- Exposes webhooks
- Processes real-time events
- Collates and maintains data you can query with web requests
- Even provides a dashboard to monitor activity



Node-RED, What Is It?

- One of the most useful tools you'll ever put in your toolbox
- Runs anywhere you can run Node.js
- It's free, it's open
- Developer community add-ons
- Graphical flow with embedded JS if needed
- Perfect for integrations and prototyping
- Import the sample, configure and go



Node-RED

Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions. Wikipedia

Developer(s): JS Foundation

License: Apache License 2.0

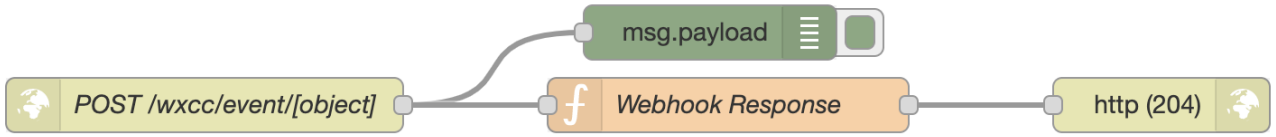
Platform: Node.js

Stable release: 2.2.0 / January 27, 2022; 3 months ago

Original author(s): IBM Emerging Technology; • Nick O'Leary; • Dave Conway-Jones

Written in: JavaScript

Webhooks Receiving Event Notifications



- Web application listening for incoming HTTP(S) requests
- Must be reachable from Webex Contact Center
- IANA-listed top-level domain name
- Notifications will fail if web app is using self-signed certificates

Example: Agent State Change Notification

```
{
  "data": {
    "agentId": "c248bed6-9fd0-4a89-b927-112ce8b9507e",
    "taskId": "2958a718-c8c8-4d08-8e74-c50b8eaa25bf",
    "queueId": "2a11e7dc-900c-4ce4-bf7b-da32b3fb6204",
    "teamId": "de223c79-30f9-4645-acd5-4deec8ba66cf",
    "origin": "+447740550595",
    "destination": "1004",
    "createdTime": 1653866206726,
    "currentState": "connected"
  },
  "type": "agent:state_change",
    ...
}
```

Related object references

Agent state

CALL ANSWERED BY AGENT

JSON payload containing –

- current agent state
- **timestamp**
- connected task details
- other related information

Handle event with 2 main actions:

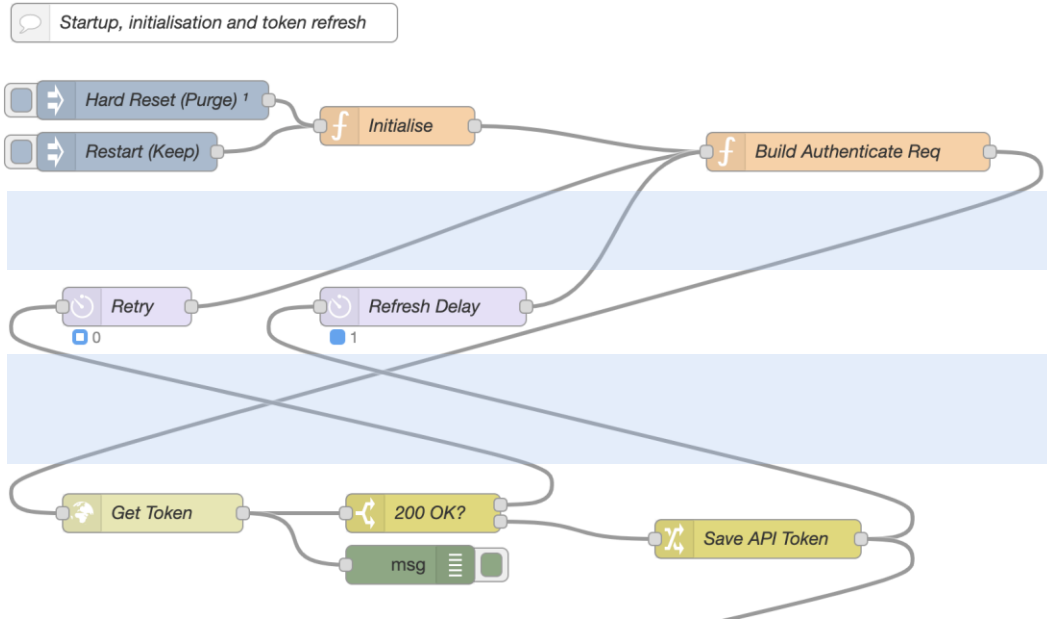
1. Update the agent object to reflect current state
2. Issue API calls to resolve other references of interest such as Queue and Team (if we don't already have them from previous events)
   Friendlier to expose names rather than internal hex IDs

We Get The Notifications, What Next?

Maintain data that can easily be consumed in routing decisions

- Build and update objects for agents, tasks, queues, teams
- Collect and cache real-time state for the cloud contact centre
- Expose web APIs for simple queries, for example –
  - Queue length
  - Longest waiting time in queue
  - Summary agent state by team
  - Calls in progress
- Could also overlay rate-limited requests for EWT in background

Essential Upfront Processing

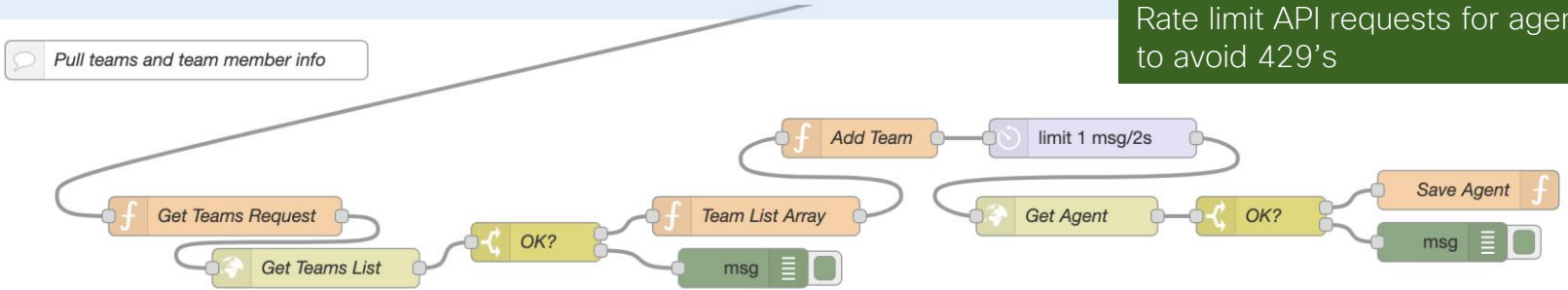


Initialisation of internal data and authentication request

Wait for token refresh / retry

API request for new access token
Save if OK and queue next refresh or retry if error

Prepopulate Teams & Agents



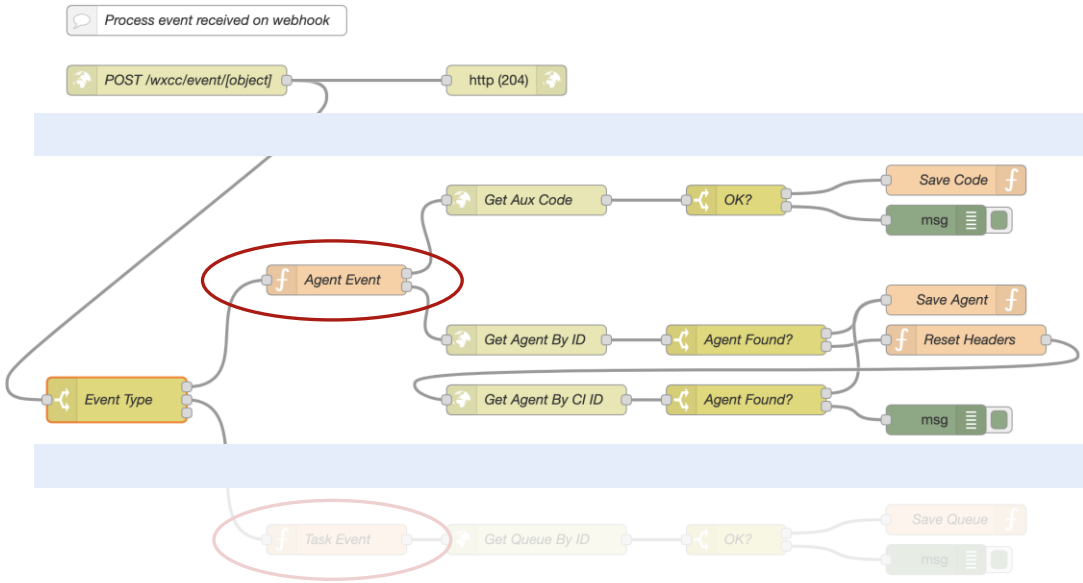
Rate limit API requests for agent info to avoid 429's

Get list of all teams as JSON array
Each team includes membership as array of agent IDs
Issue API request for each agent ID

Real-Time Event Handling

Incoming notification
Respond 204 No Content

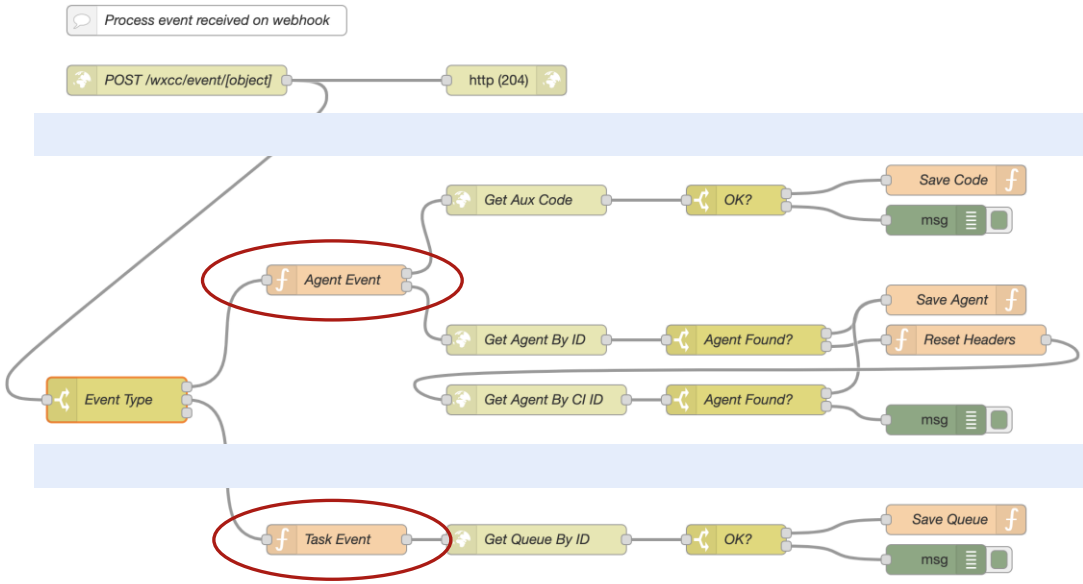
Real-Time Event Handling



Incoming notification
Respond 204 No Content

Update agent status
Get agent config if unknown
Get idle/wrap-up code

Real-Time Event Handling



Incoming notification
Respond 204 No Content

Update agent status
Get agent config if unknown
Get idle/wrap-up code

Update task status
Create if new
Delete if ended
Resolve queue if unknown

Using The Data

Before We Can Use The Data, One Last Thing … Expose Endpoints To Query Current State

GET https://pcce.vpod1628.dc-01.com/wxcc/queueinfo/Q_Voice_Paul Send

Status: 200 OK Size: 117 Bytes Time: 328 ms

Query Headers 2 Auth Body Tests

Response Headers 9 Cookies Results

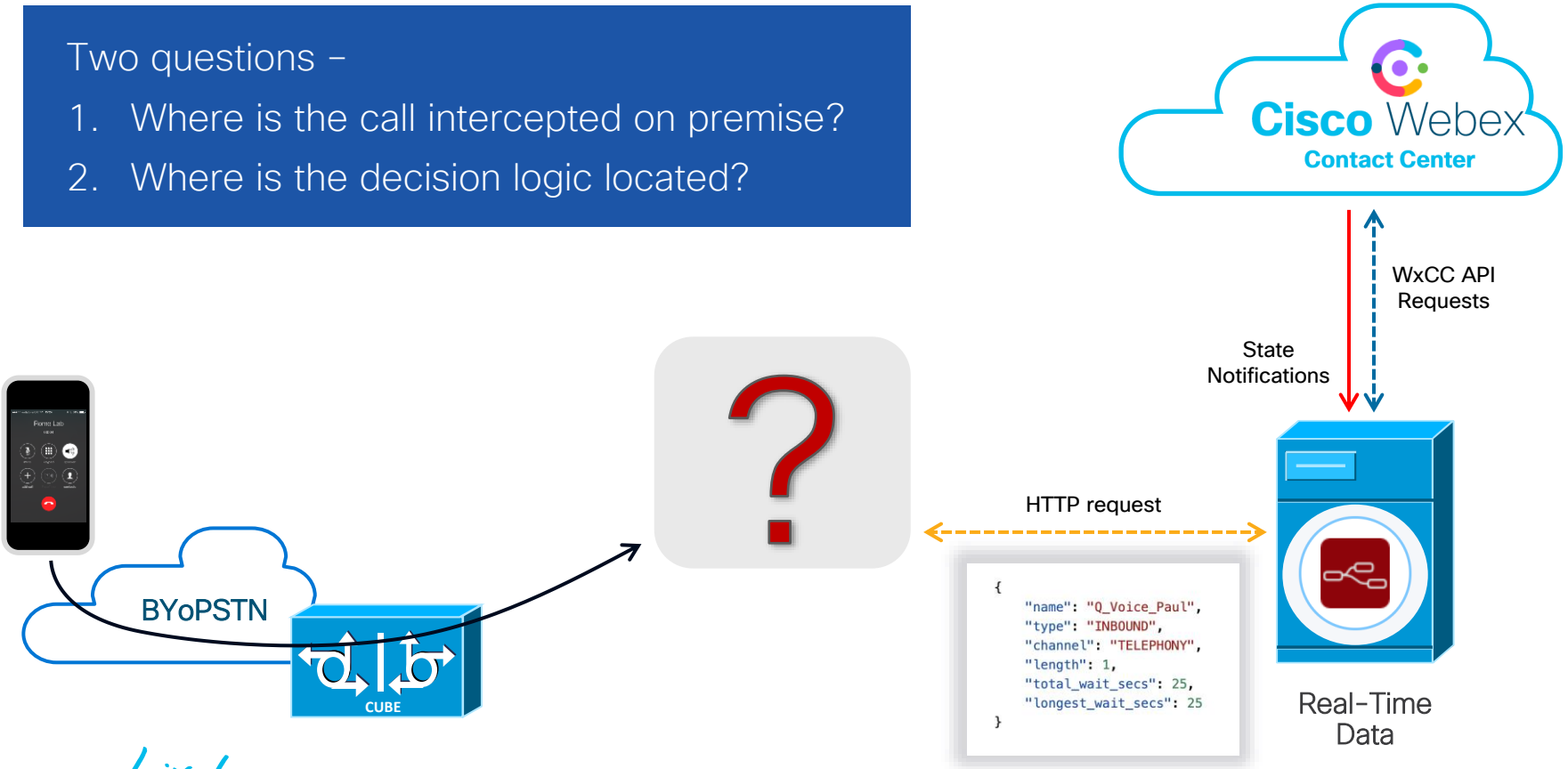

```
1 {
2   "name": "Q_Voice_Paul",
3   "type": "INBOUND",
4   "channel": "TELEPHONY",
5   "length": 1,
6   "total_wait_secs": 25,
7   "longest_wait_secs": 25
8 }
```


Now we have it …
… data we can use

Intercepting Calls & Making Routing Decisions

Two questions –

1. Where is the call intercepted on premise?
2. Where is the decision logic located?



Intercepting Calls & Making Routing Decisions

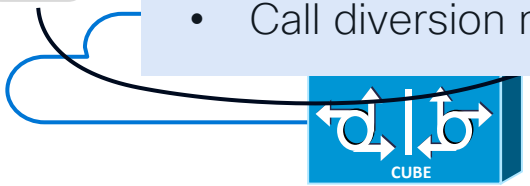
Two questions –

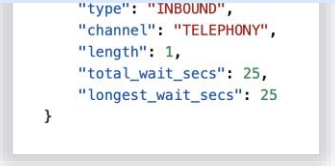
1. Where is the call intercepted on premise?
2. Where is the decision logic located?

WxCC API Requests

Depends on –

- The premise platform: CCX, CCE, CVP, CUCM, CUBE
- Integration preferences: out-the-box only or custom use of APIs
- Call diversion mechanism: transfer, refer, redirect pre-answer



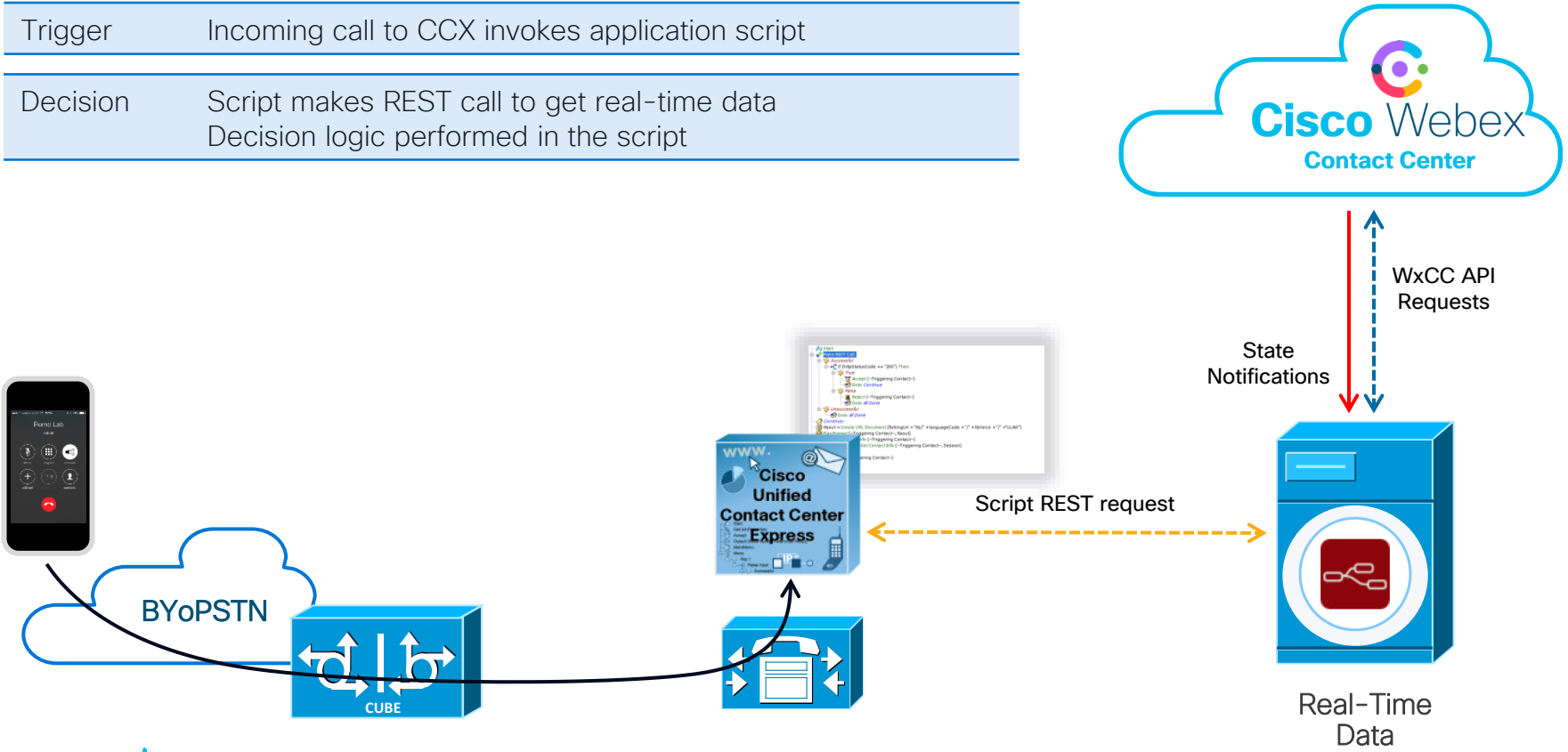


Real-Time Data

Contact Center Express

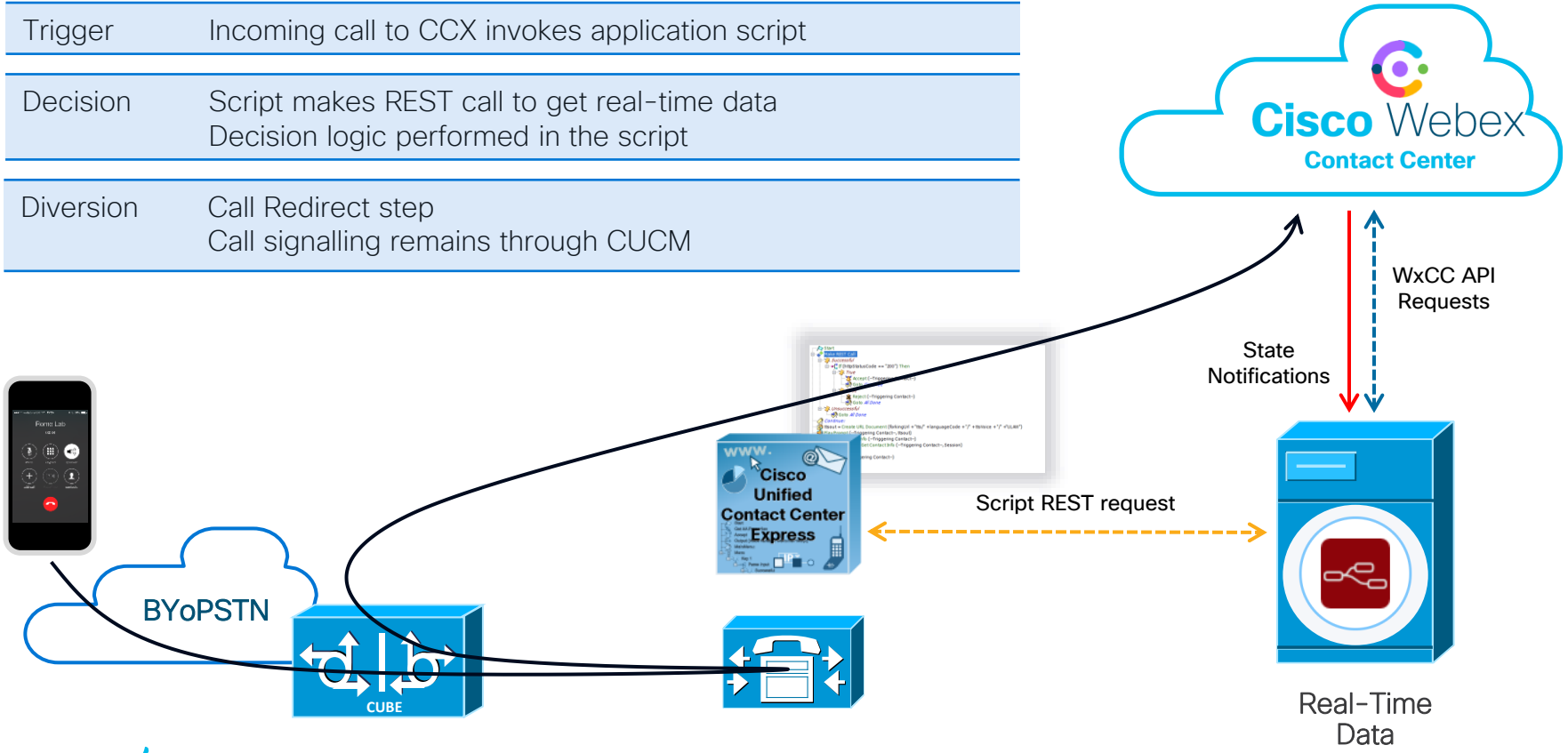
Contact Center Express (CCX)

Trigger	Incoming call to CCX invokes application script
Decision	Script makes REST call to get real-time data Decision logic performed in the script



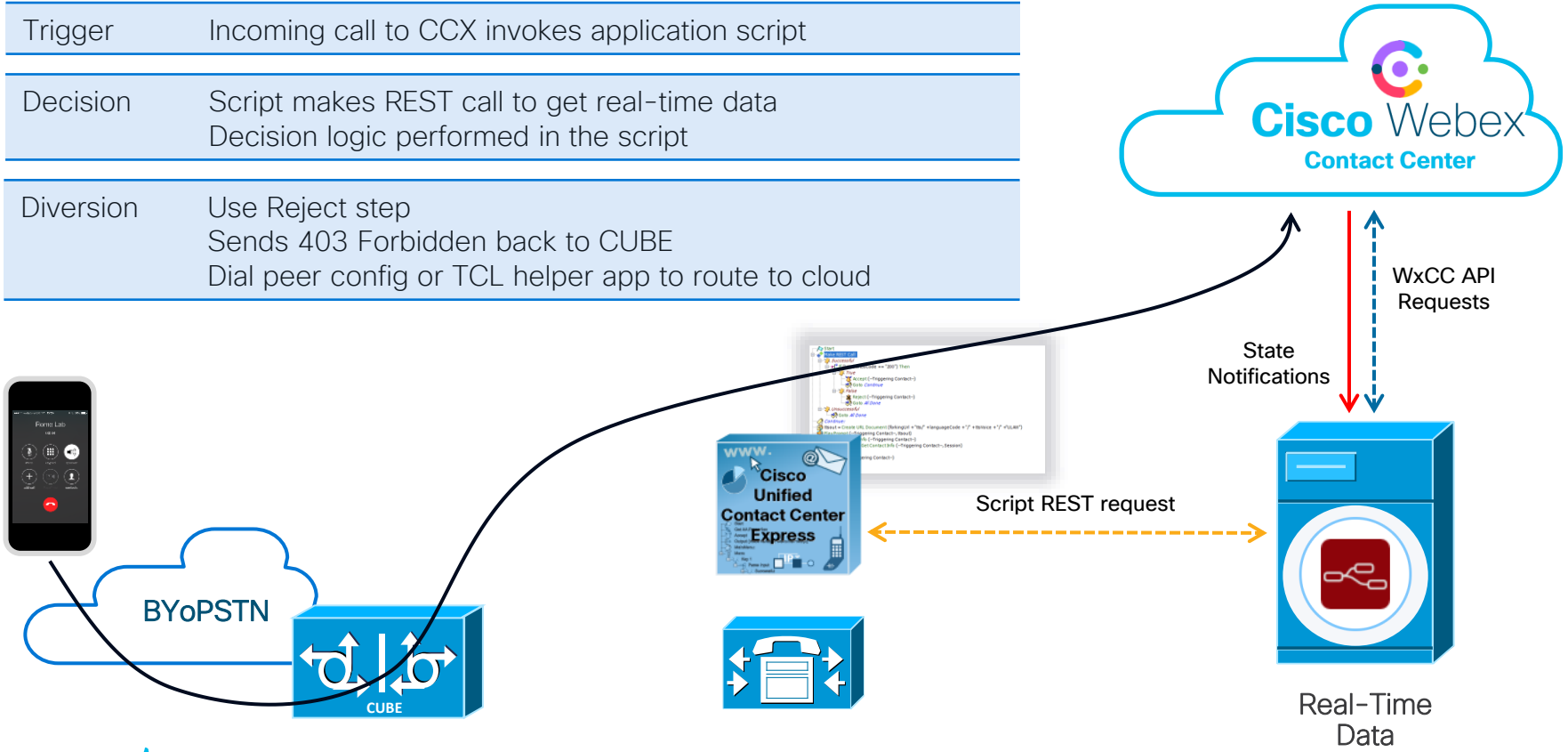
Contact Center Express (CCX)

Trigger	Incoming call to CCX invokes application script
Decision	Script makes REST call to get real-time data Decision logic performed in the script
Diversion	Call Redirect step Call signalling remains through CUCM



Contact Center Express (CCX)

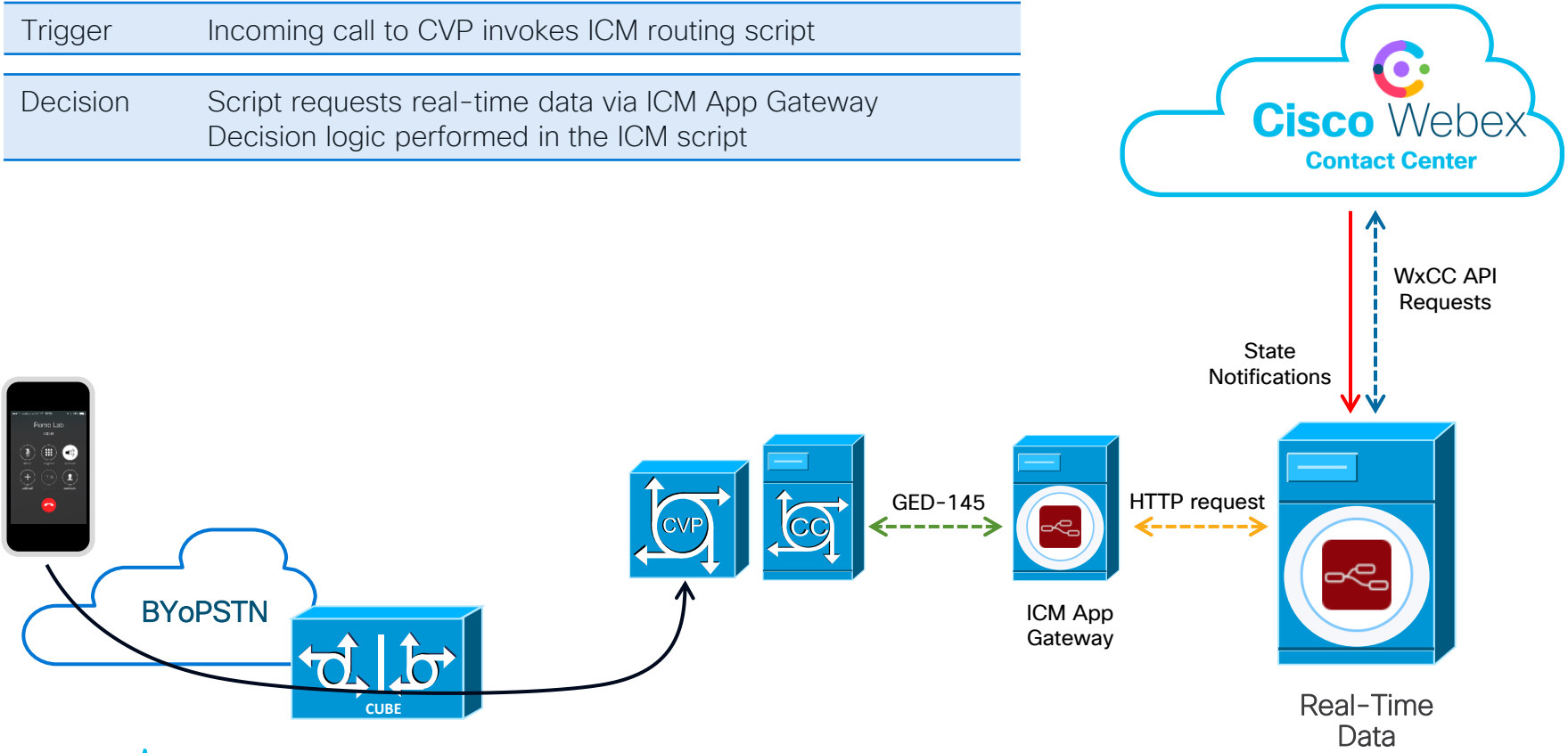
Trigger	Incoming call to CCX invokes application script
Decision	Script makes REST call to get real-time data Decision logic performed in the script
Diversion	Use Reject step Sends 403 Forbidden back to CUBE Dial peer config or TCL helper app to route to cloud



Contact Center Enterprise

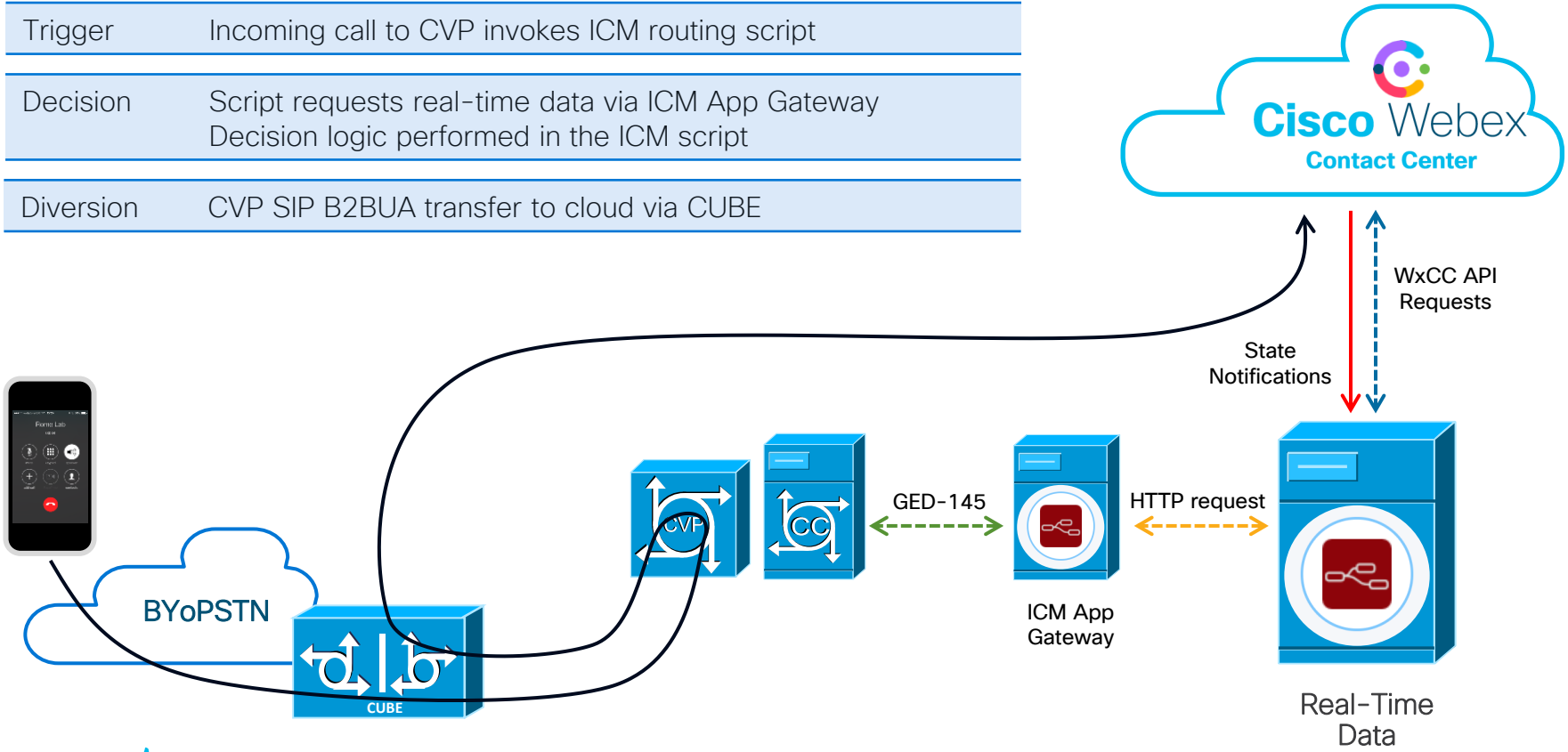
Contact Center Enterprise (CCE)

Trigger	Incoming call to CVP invokes ICM routing script
Decision	Script requests real-time data via ICM App Gateway Decision logic performed in the ICM script



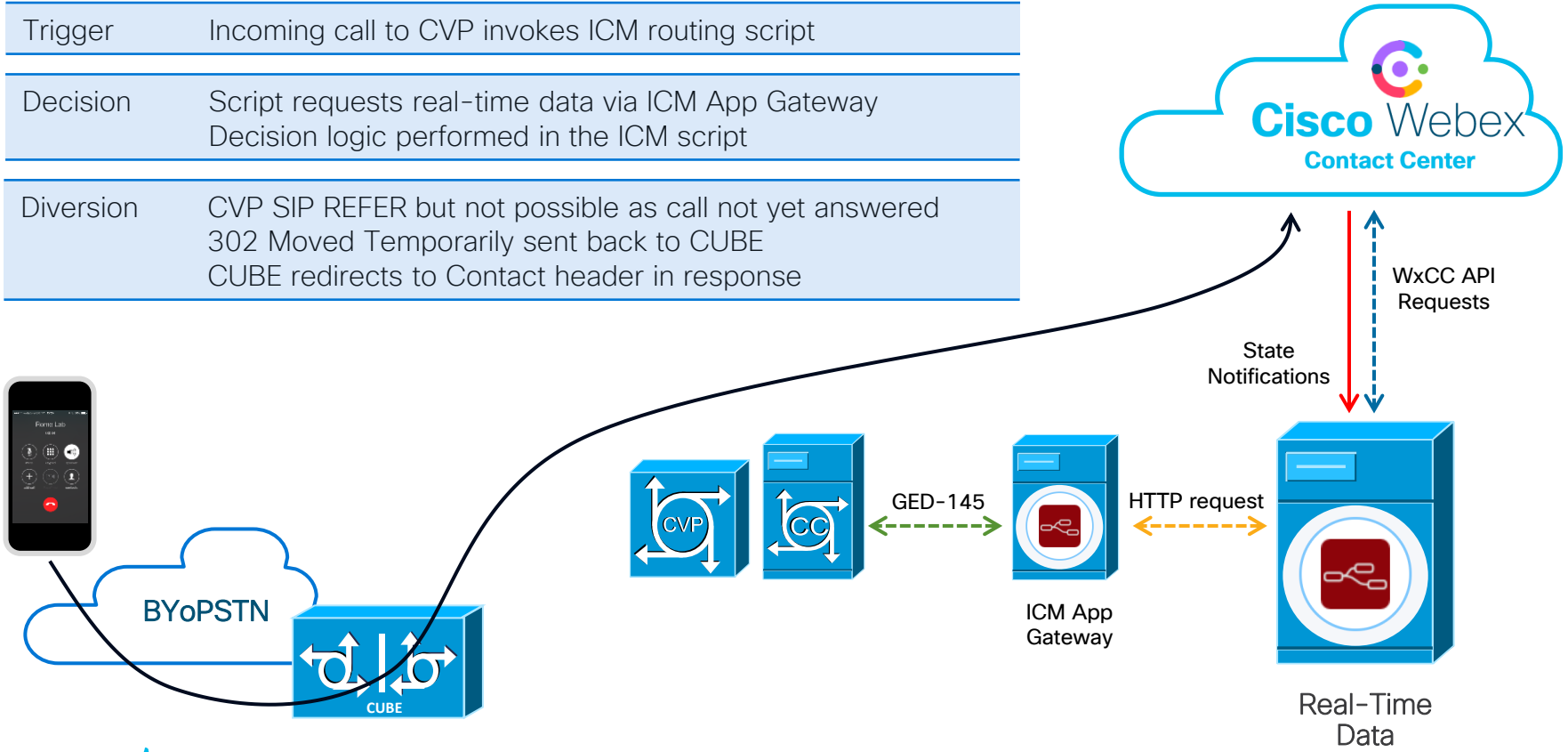
Contact Center Enterprise (CCE)

Trigger	Incoming call to CVP invokes ICM routing script
Decision	Script requests real-time data via ICM App Gateway Decision logic performed in the ICM script
Diversion	CVP SIP B2BUA transfer to cloud via CUBE



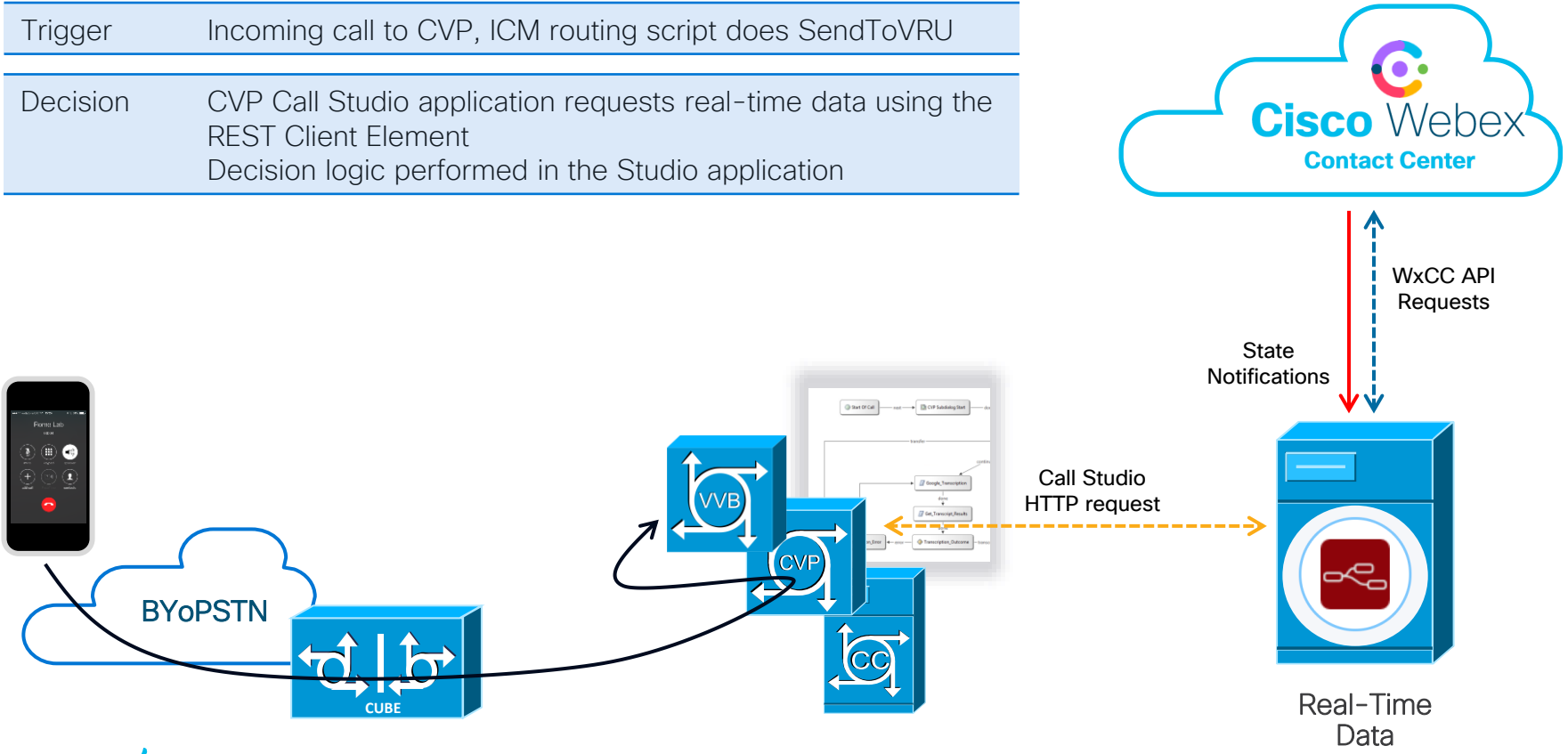
Contact Center Enterprise (CCE)

Trigger	Incoming call to CVP invokes ICM routing script
Decision	Script requests real-time data via ICM App Gateway Decision logic performed in the ICM script
Diversion	CVP SIP REFER but not possible as call not yet answered 302 Moved Temporarily sent back to CUBE CUBE redirects to Contact header in response



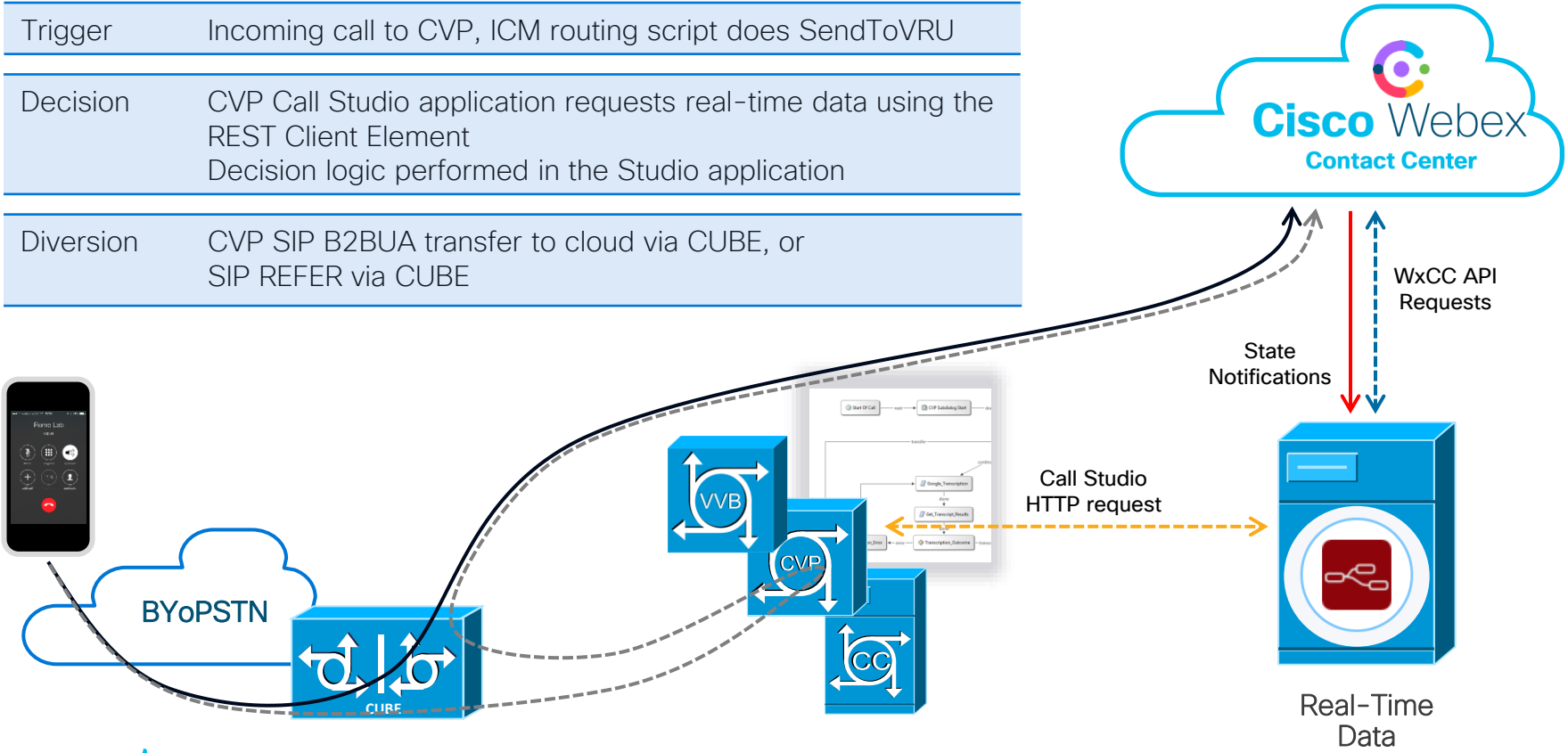
Contact Center Enterprise (CCE)

Trigger	Incoming call to CVP, ICM routing script does SendToVRU
Decision	CVP Call Studio application requests real-time data using the REST Client Element Decision logic performed in the Studio application



Contact Center Enterprise (CCE)

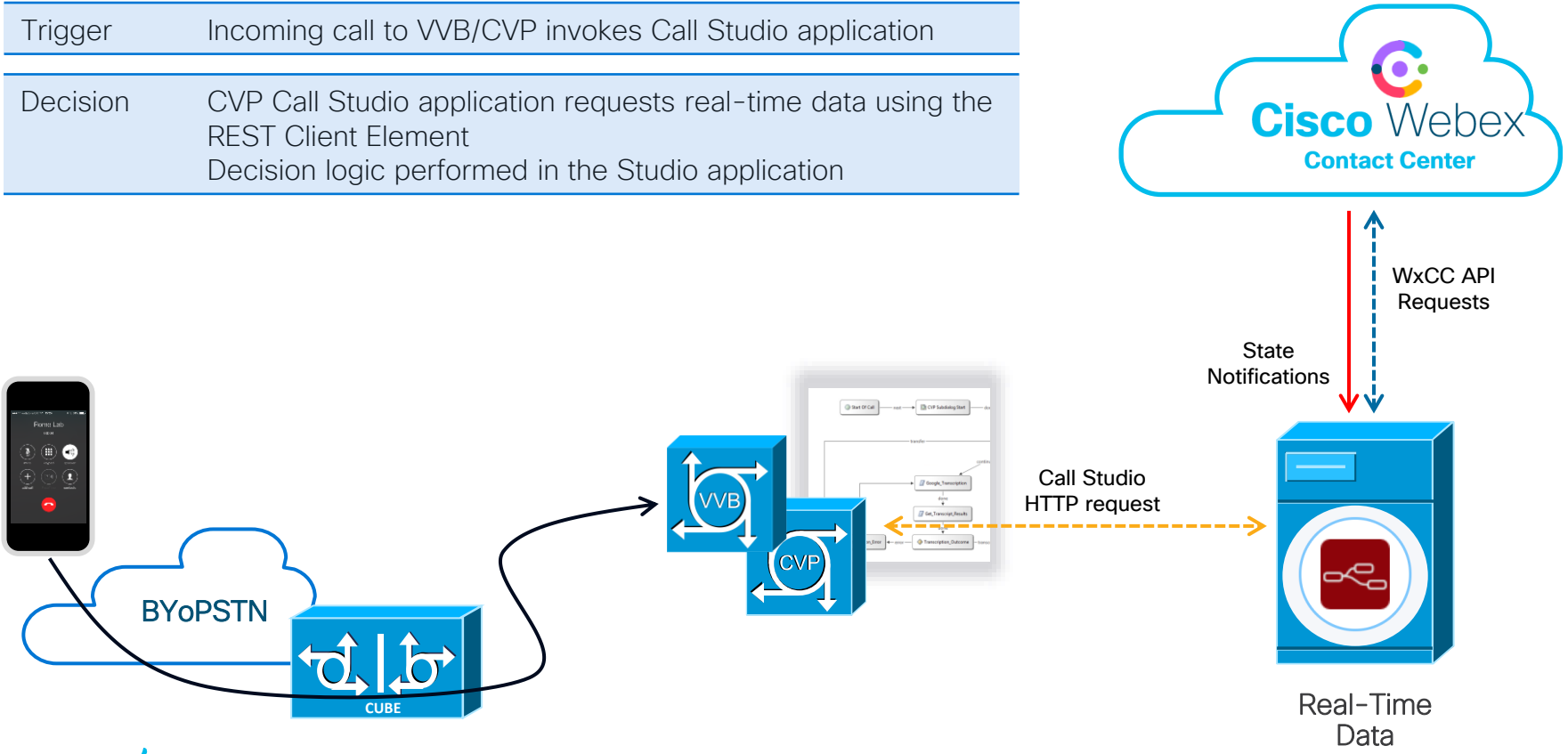
Trigger	Incoming call to CVP, ICM routing script does SendToVRU
Decision	CVP Call Studio application requests real-time data using the REST Client Element Decision logic performed in the Studio application
Diversion	CVP SIP B2BUA transfer to cloud via CUBE, or SIP REFER via CUBE



CVP Standalone

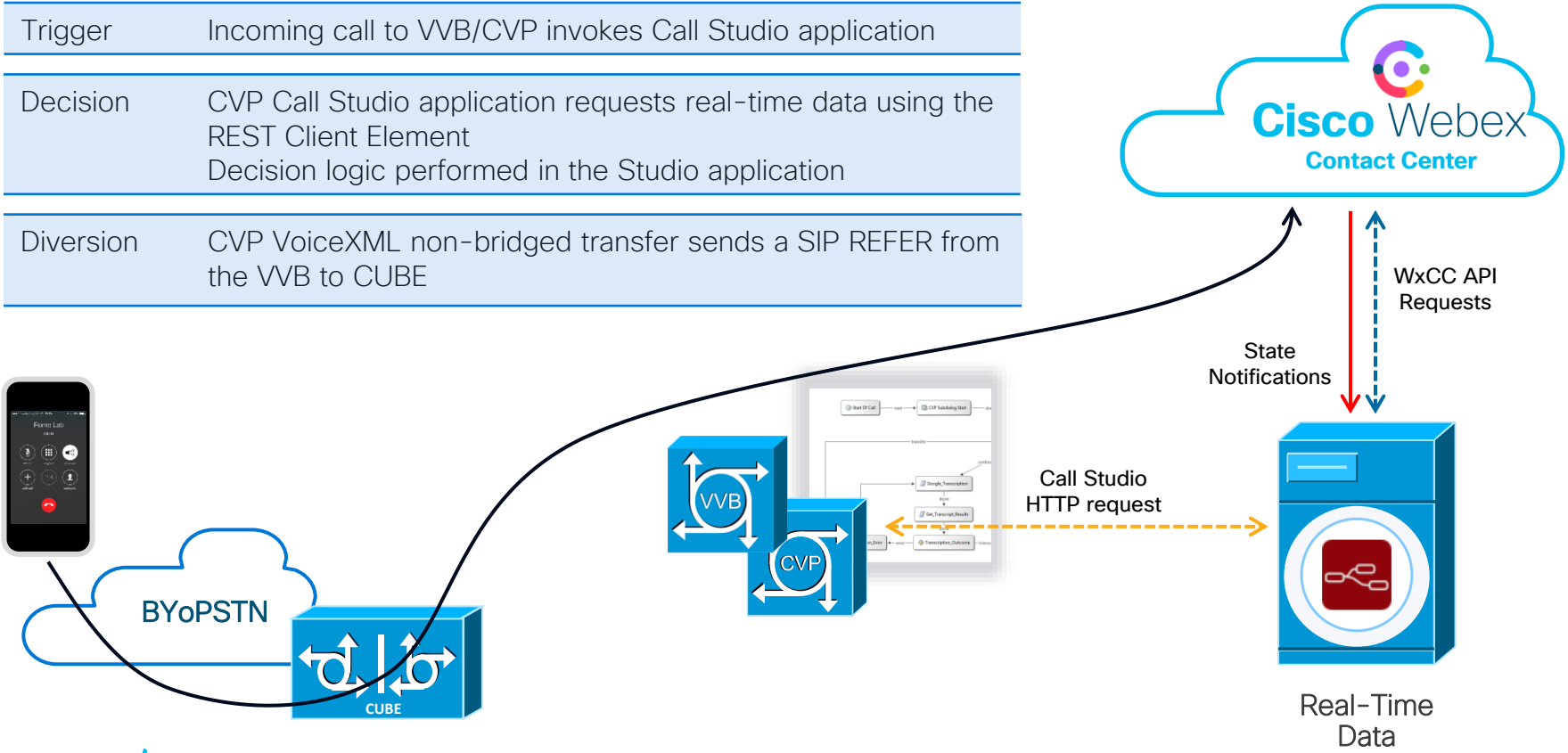
CVP Standalone

Trigger	Incoming call to VVB/CVP invokes Call Studio application
Decision	CVP Call Studio application requests real-time data using the REST Client Element Decision logic performed in the Studio application



CVP Standalone

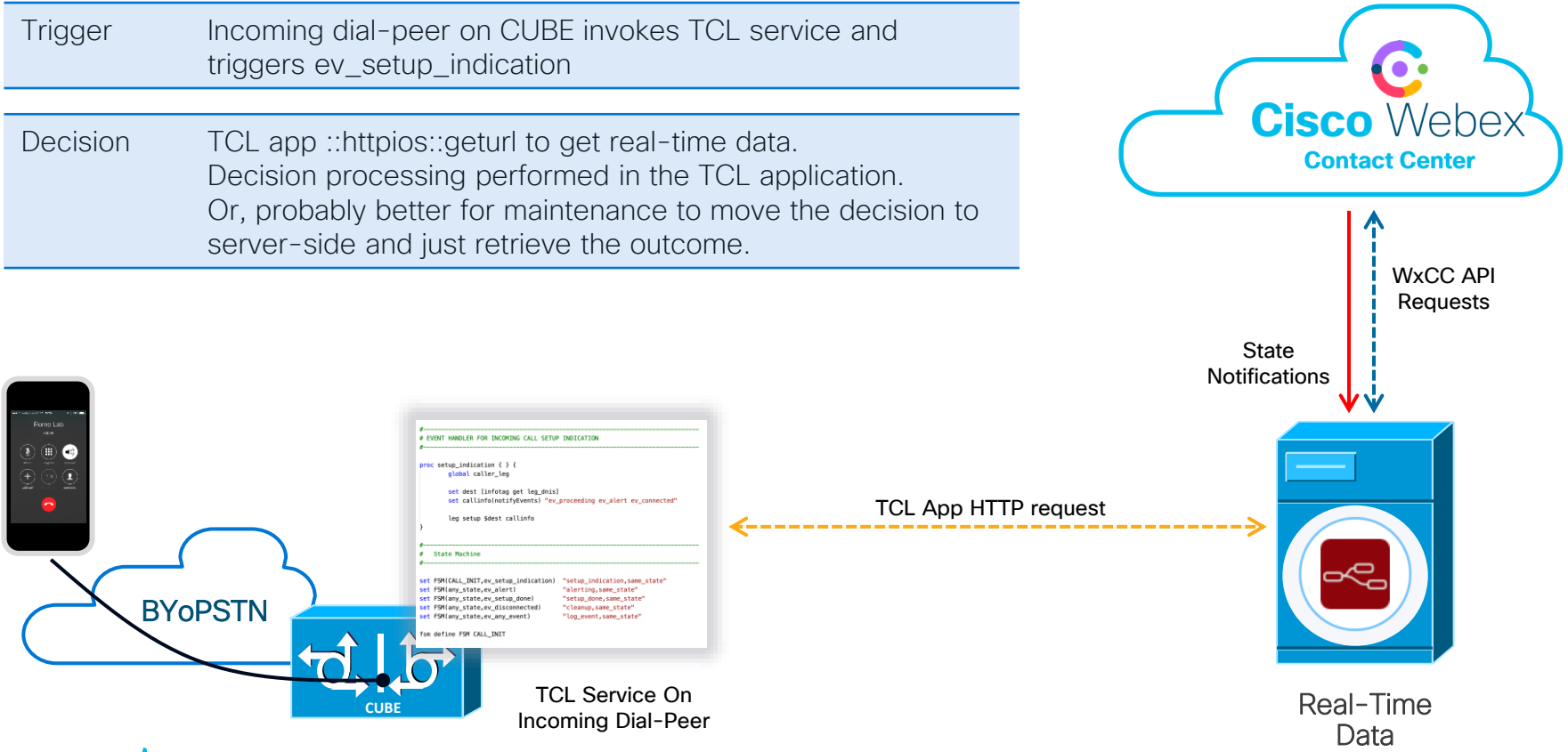
Trigger	Incoming call to VVB/CVP invokes Call Studio application
Decision	CVP Call Studio application requests real-time data using the REST Client Element Decision logic performed in the Studio application
Diversion	CVP VoiceXML non-bridged transfer sends a SIP REFER from the VVB to CUBE



CUBE

CUBE Method 1 – TCL Application

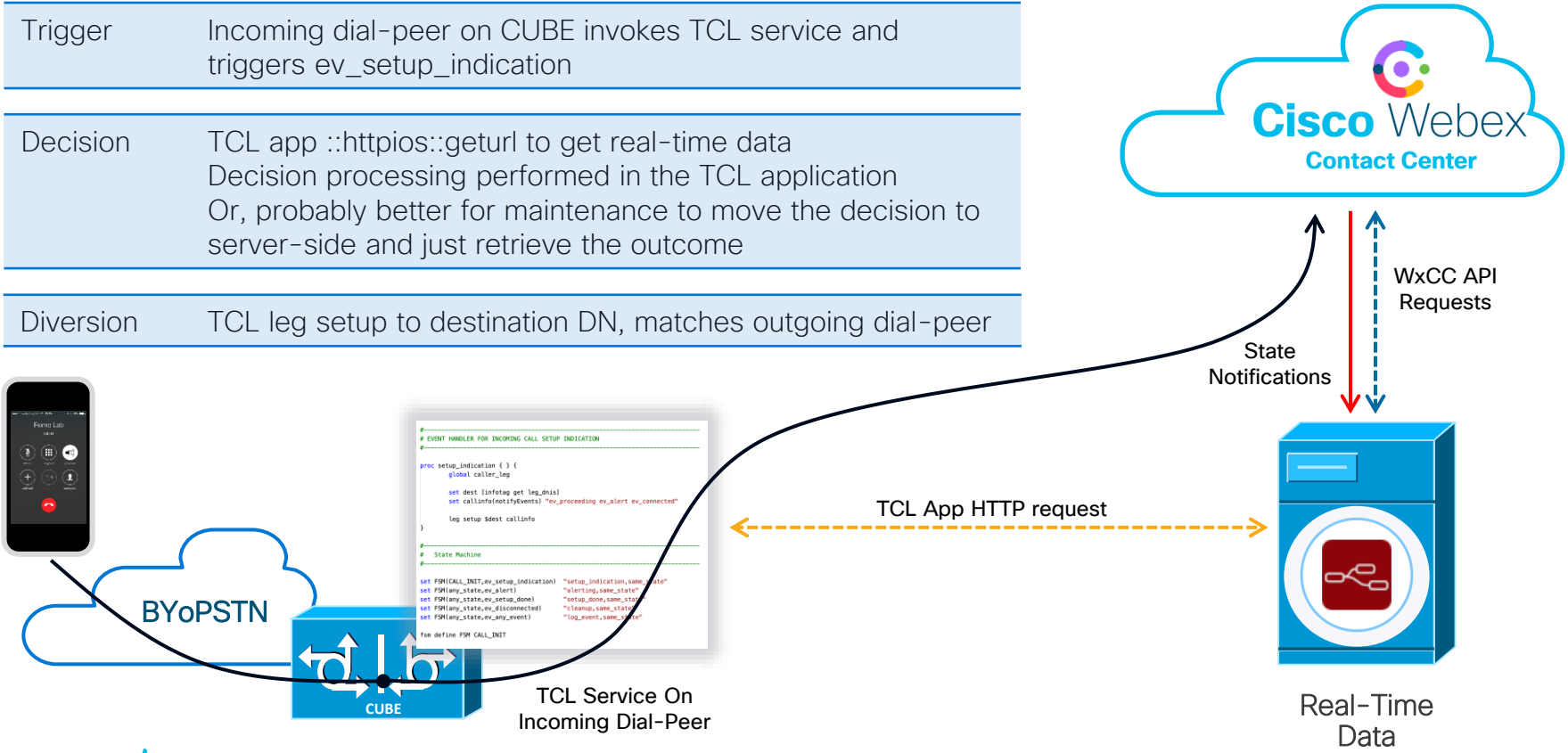
Trigger	Incoming dial-peer on CUBE invokes TCL service and triggers ev_setup_indication
Decision	TCL app ::httpios::geturl to get real-time data. Decision processing performed in the TCL application. Or, probably better for maintenance to move the decision to server-side and just retrieve the outcome.




```
#-----------------------------------------------------------------------
# EVENT HANDLER FOR INCOMING CALL SETUP INDICATION
#-----------------------------------------------------------------------

proc setup_indication { } {
        global caller_leg

        set dest [infotag get leg_dnis]
        set callinfo(notifyEvents) "ev_proceeding ev_alert ev_connected"

        leg setup $dest callinfo
}

#-----------------------------------------------------------------------
#   State Machine
#-----------------------------------------------------------------------

set FSM(CALL_INIT,ev_setup_indication)  "setup_indication,same_state"
set FSM(any_state,ev_alert)             "alerting,same_state"
set FSM(any_state,ev_setup_done)        "setup_done,same_state"
set FSM(any_state,ev_disconnected)      "cleanup,same_state"
set FSM(any_state,ev_any_event)         "log_event,same_state"

fsm define FSM CALL_INIT
```

CUBE Method 1 – TCL Application

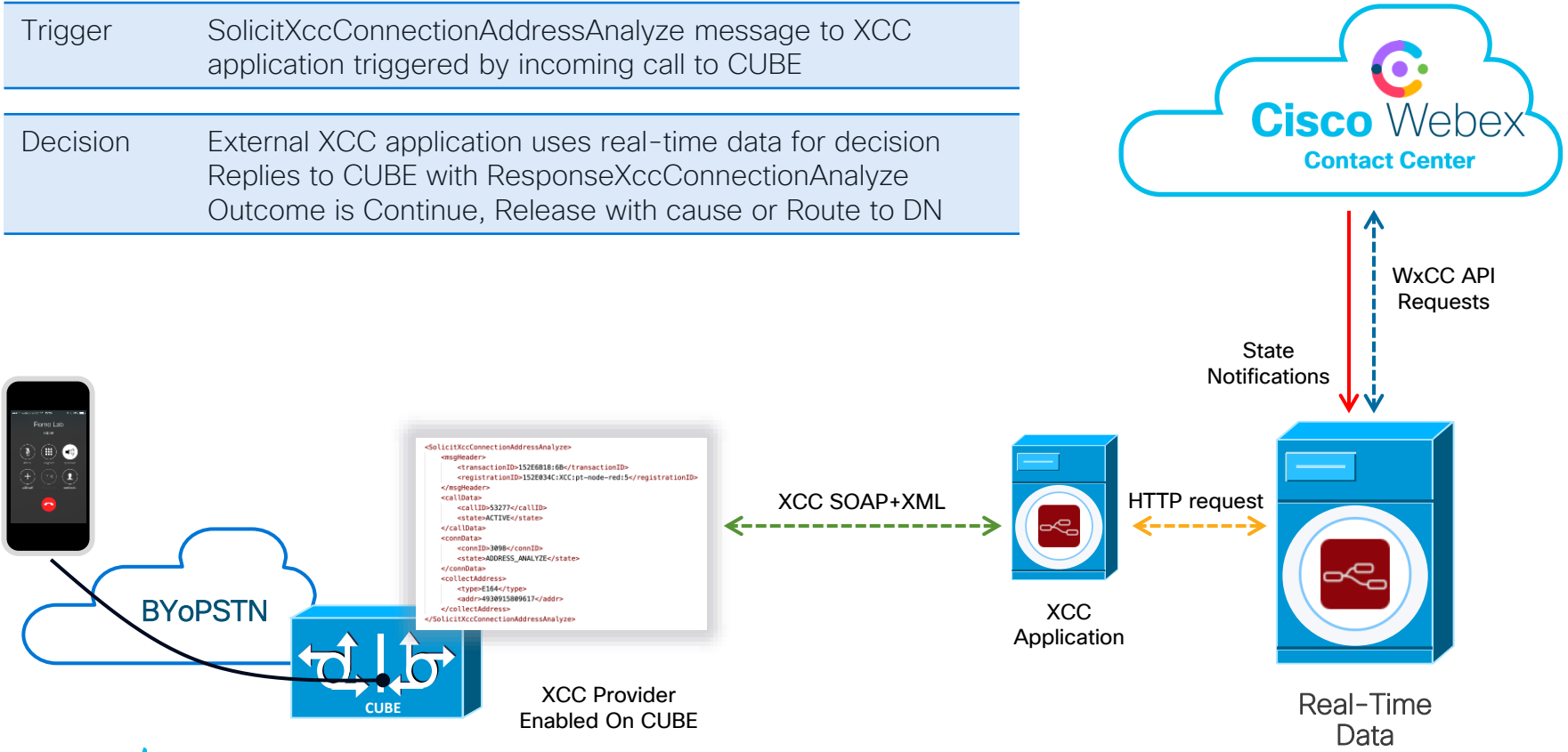
Trigger	Incoming dial-peer on CUBE invokes TCL service and triggers ev_setup_indication
Decision	TCL app ::httpios::geturl to get real-time data Decision processing performed in the TCL application Or, probably better for maintenance to move the decision to server-side and just retrieve the outcome
Diversion	TCL leg setup to destination DN, matches outgoing dial-peer




```
#-----------------------------------------------------------------------
# EVENT HANDLER FOR INCOMING CALL SETUP INDICATION
#-----------------------------------------------------------------------

proc setup_indication { } {
        global caller_leg

        set dest [infotag get leg_dnis]
        set callinfo(notifyEvents) "ev_proceeding ev_alert ev_connected"

        leg setup $dest callinfo
}

#-----------------------------------------------------------------------
#   State Machine
#-----------------------------------------------------------------------

set FSM(CALL_INIT,ev_setup_indication)  "setup_indication,same_state"
set FSM(any_state,ev_alert)             "alerting,same_state"
set FSM(any_state,ev_setup_done)        "setup_done,same_state"
set FSM(any_state,ev_disconnected)      "cleanup,same_state"
set FSM(any_state,ev_any_event)         "log_event,same_state"

fsm define FSM CALL_INIT
```

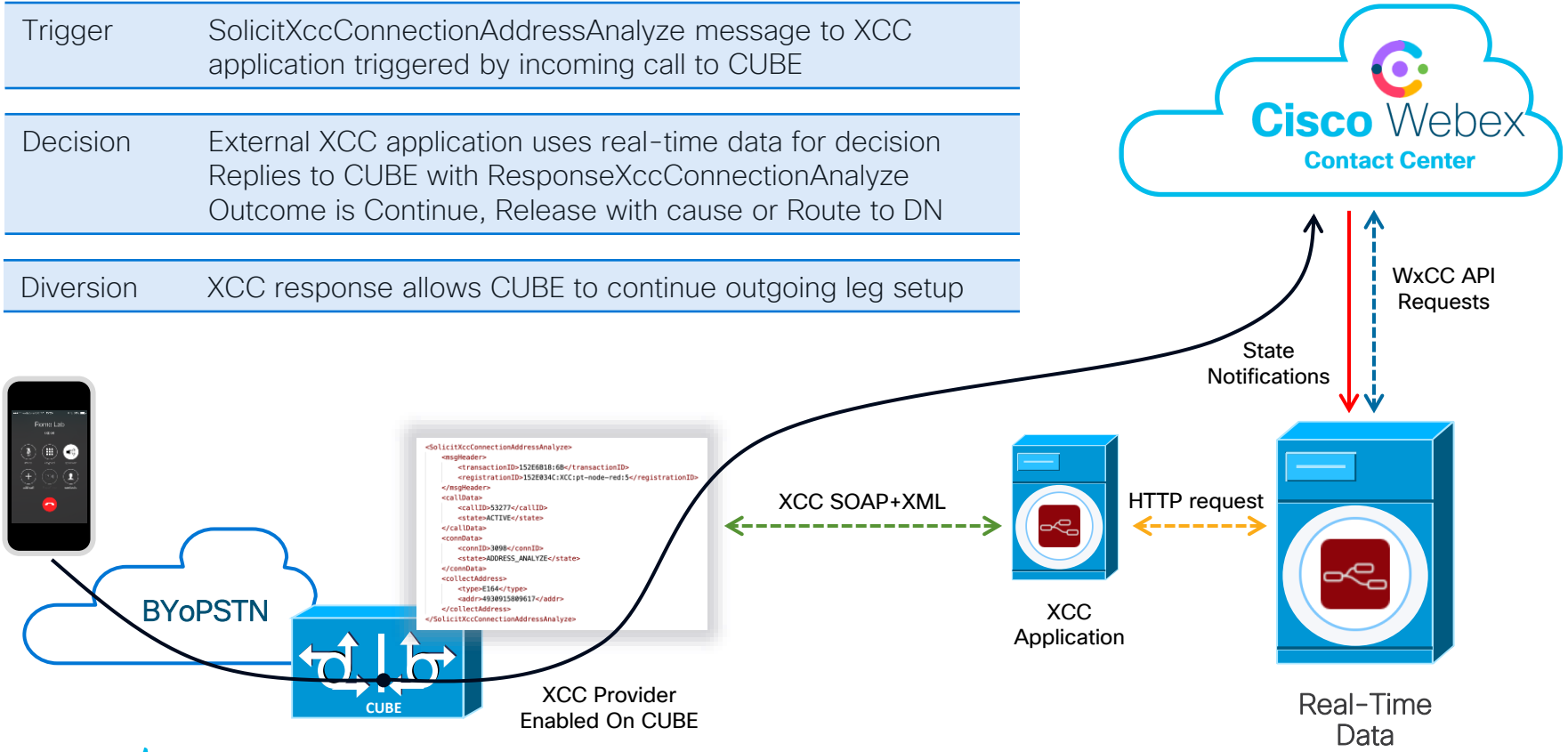
CUBE Method 2 – Gateway Services XCC API

Trigger	SolicitXccConnectionAddressAnalyze message to XCC application triggered by incoming call to CUBE
Decision	External XCC application uses real-time data for decision Replies to CUBE with ResponseXccConnectionAnalyze Outcome is Continue, Release with cause or Route to DN



CUBE Method 2 – Gateway Services XCC API

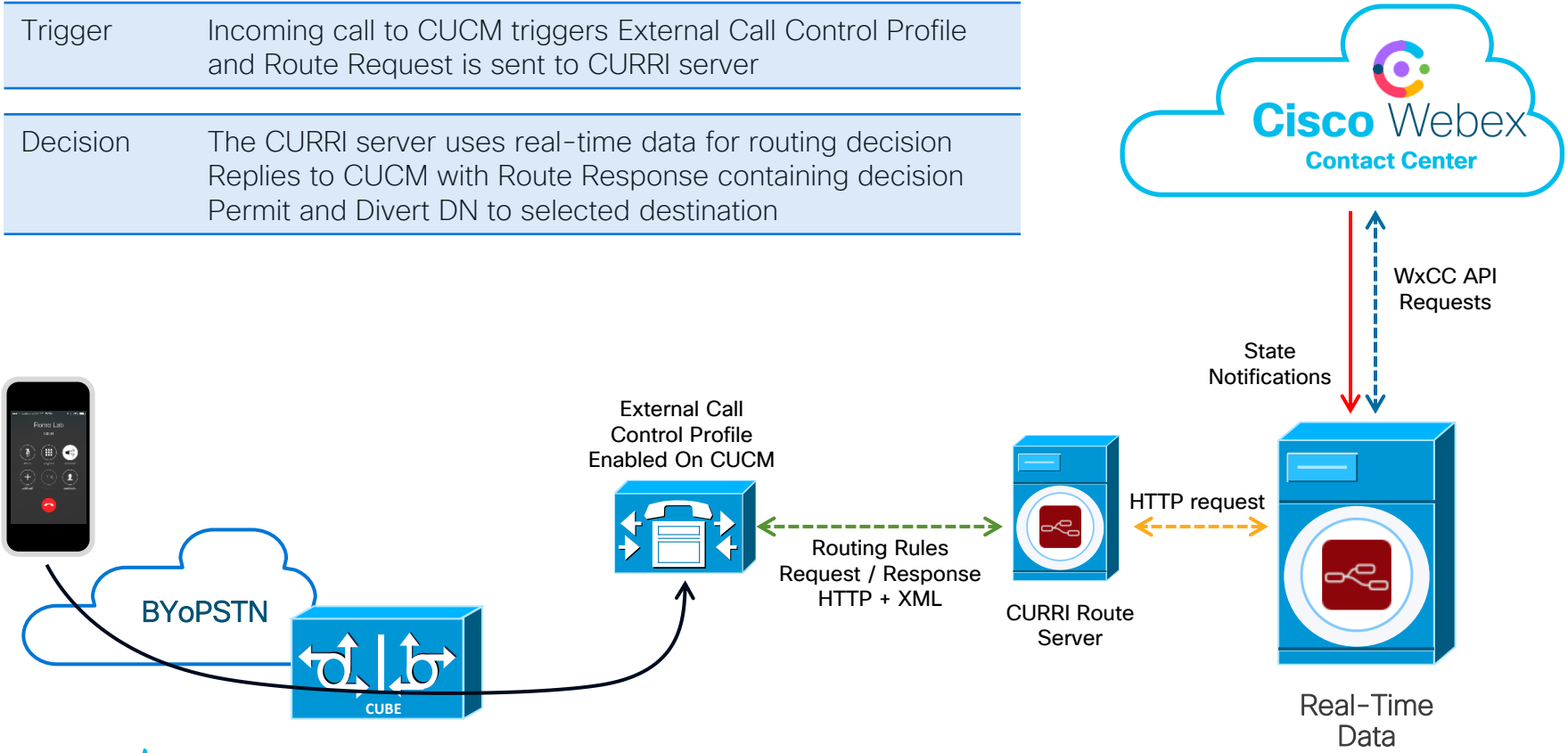
Trigger	SolicitXccConnectionAddressAnalyze message to XCC application triggered by incoming call to CUBE
Decision	External XCC application uses real-time data for decision Replies to CUBE with ResponseXccConnectionAnalyze Outcome is Continue, Release with cause or Route to DN
Diversion	XCC response allows CUBE to continue outgoing leg setup



Communications Manager

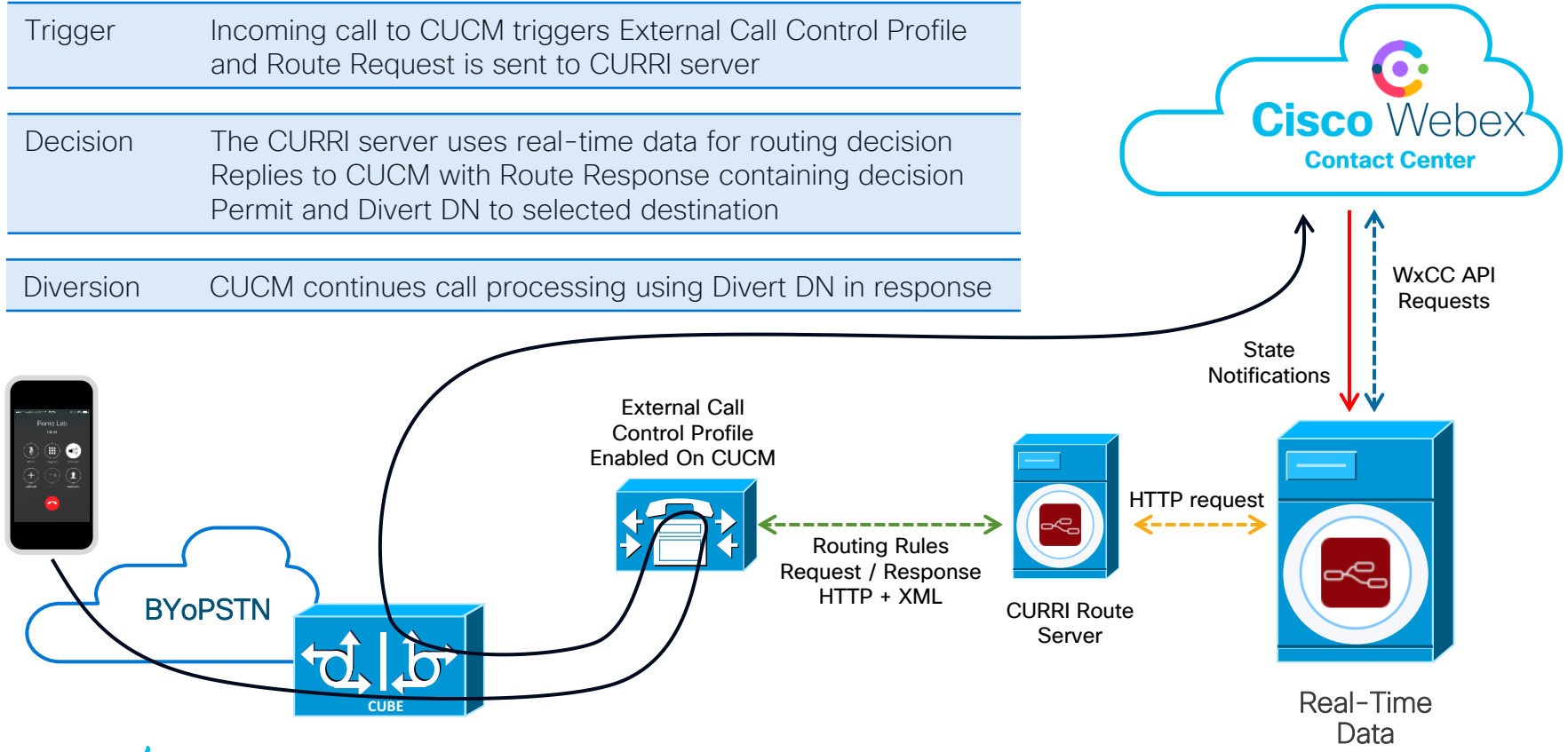
Communications Manager (CUCM)

Trigger	Incoming call to CUCM triggers External Call Control Profile and Route Request is sent to CURRI server
Decision	The CURRI server uses real-time data for routing decision Replies to CUCM with Route Response containing decision Permit and Divert DN to selected destination

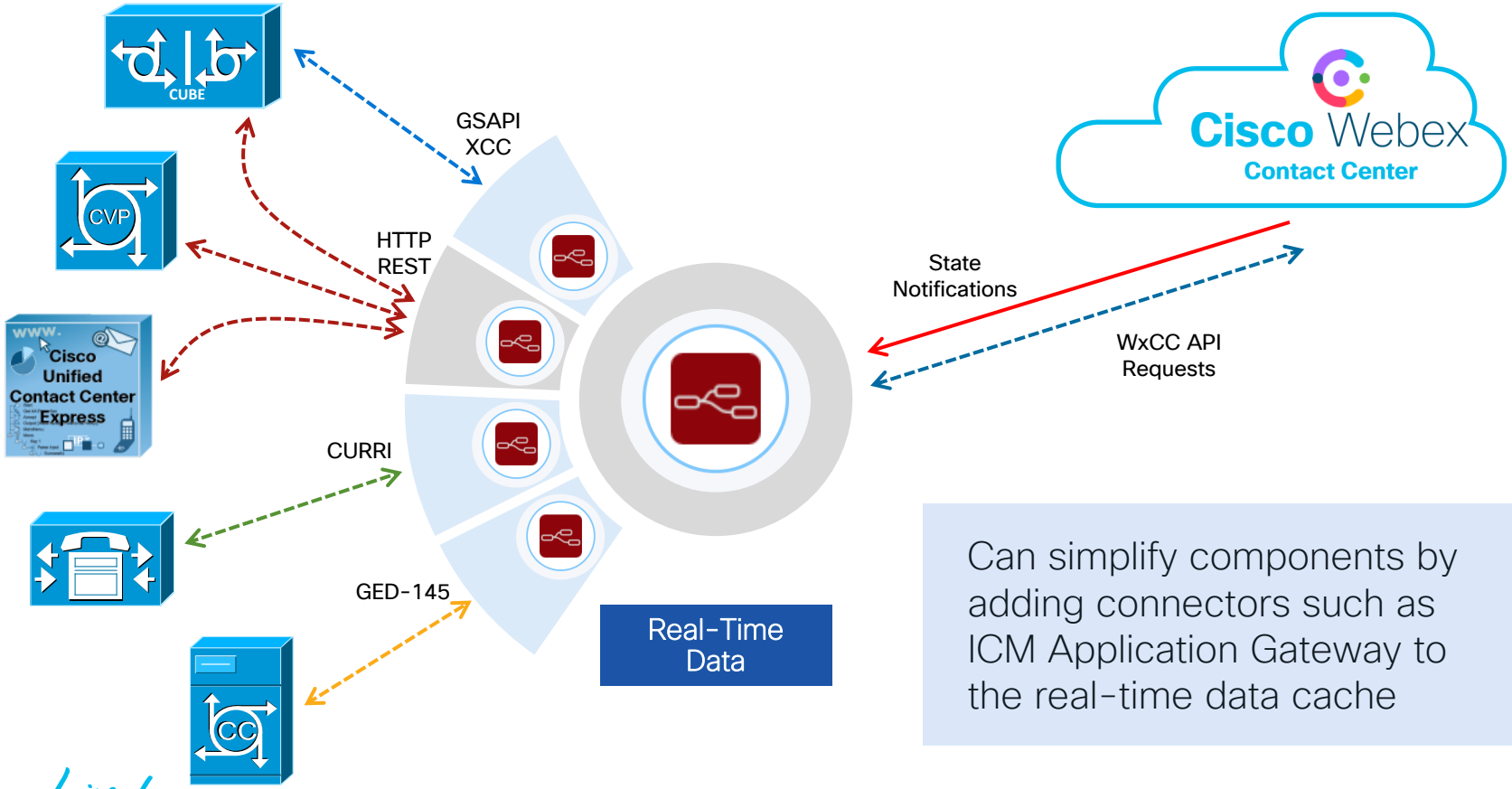


Communications Manager (CUCM)

Trigger	Incoming call to CUCM triggers External Call Control Profile and Route Request is sent to CURRI server
Decision	The CURRI server uses real-time data for routing decision Replies to CUCM with Route Response containing decision Permit and Divert DN to selected destination
Diversion	CUCM continues call processing using Divert DN in response



Optionally Merge Connector With Data Server



Can simplify components by adding connectors such as ICM Application Gateway to the real-time data cache

Time To See It In Action

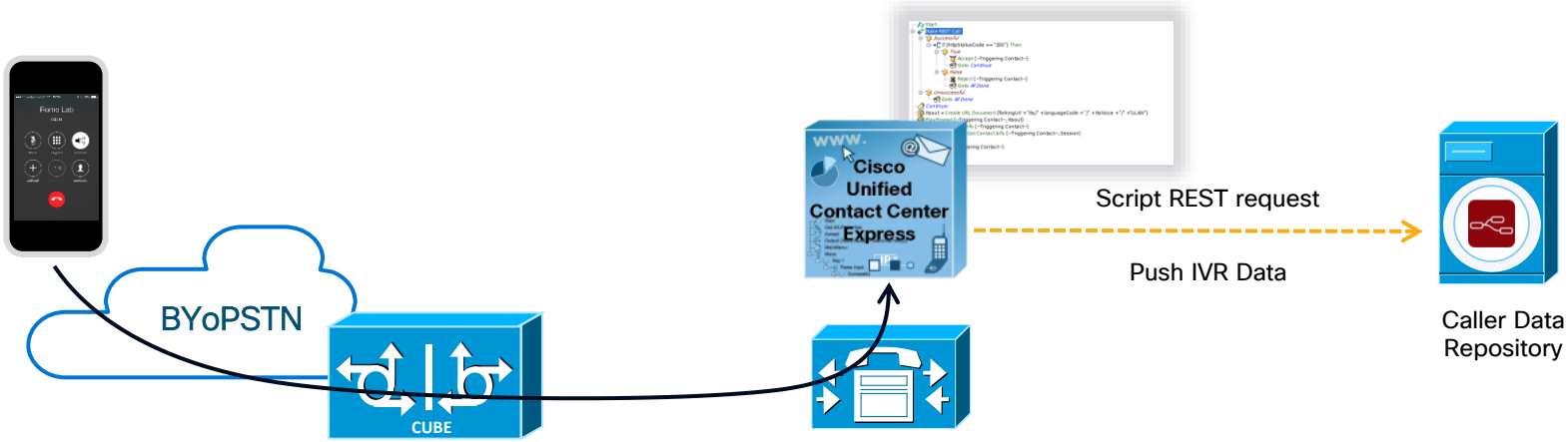
What We'll Do

- See real-time state changes generated by Webex CC
- Make calls to CCX from PSTN
- Call diverted to cloud if the cloud queue is EMPTY
- Else, call handled on CCX if cloud queue length > 0
- Hear IVR message to confirm where the call landed
- CCX IVR using female voice, Webex CC using male voice

Taking Hybrid Call Flows Further

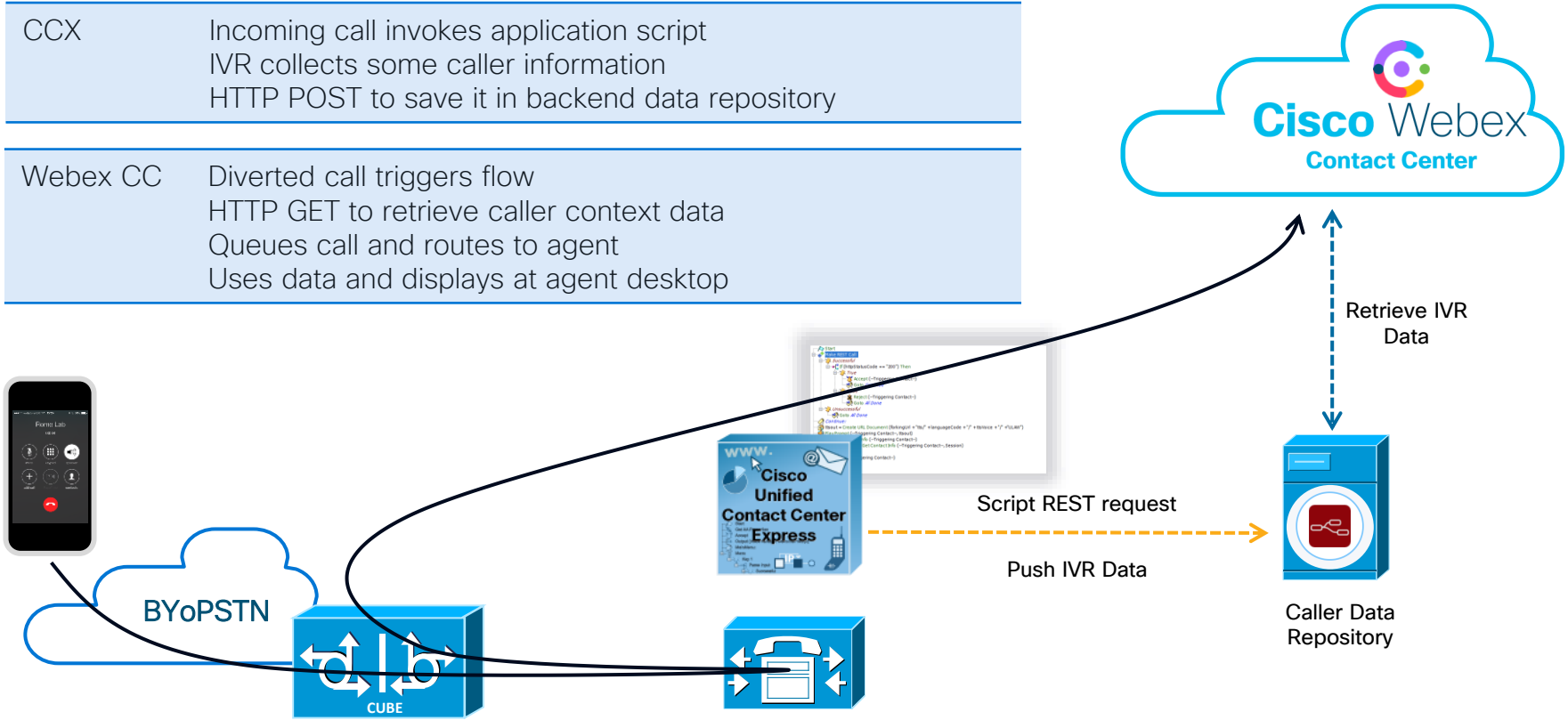
Premise IVR, Webex CC Agent, Pass Context

CCX	Incoming call invokes application script IVR collects some caller information Saves it in backend data repository



Premise IVR, Webex CC Agent, Pass Context

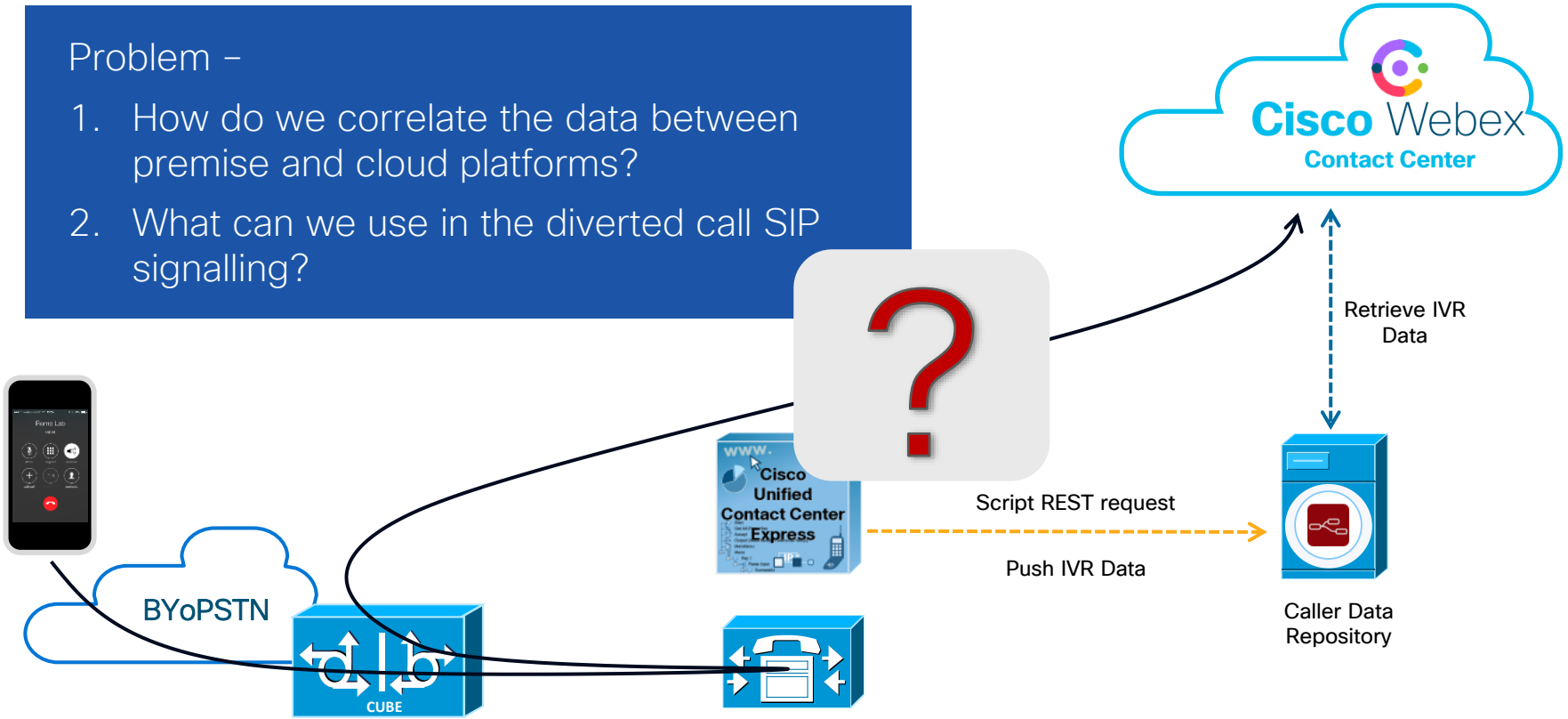
CCX	Incoming call invokes application script IVR collects some caller information HTTP POST to save it in backend data repository
Webex CC	Diverted call triggers flow HTTP GET to retrieve caller context data Queues call and routes to agent Uses data and displays at agent desktop



Premise IVR, Webex CC Agent, Pass Context

Problem –

1. How do we correlate the data between premise and cloud platforms?
2. What can we use in the diverted call SIP signalling?



Correlation: What Are The Options?

Calling Number

- Works much of the time but not 100% reliable
- Could be withheld or not personal
- Could spoof it to achieve uniqueness
- CLI override available on CCE/CVP, but ...
- ... Harder on CCX – CURRI or ingress CUBE TCL app needed
- Using fake calling numbers makes things harder for tracing and reporting
- And affects what the agent sees on the desktop

Correlation: What Are The Options?

Called Number – Include a correlation ID in the divert destination DN

- Add extra digits to achieve uniqueness
- Works if the destination supports wildcarded incoming numbers
- But unfortunately, …
- … Webex CC doesn't

Correlation: What Are The Options?

Called Number – Translation routing approach using DN number pool

- Allocate divert DN from pool of numbers
- Save context using DN as key
- Send the call to Webex CC
- Retrieve context using DN as lookup key
- Deallocate the temporary number
- But, …
- … Adds complexity with backend mechanism to manage the number pool
- It is a reliable mechanism though

Correlation: What Are The Options?

Send DTMF -- Outpulse correlation ID on transfer call leg

- DESPERATE ATTEMPT OF LAST RESORT
- IVR must be able to set up an outgoing consult call, send DTMF and then complete the transfer of the incoming call leg
- CCX has a call control step that does this
- CVP requires custom workarounds
- Can be error prone, synchronising the sending and receiving ends

Correlation: What Are The Options?

Custom SIP Header Passing

- Send unique call ID in a custom header on the transfer leg
- YES – THE OPTIMAL SOLUTION (BUT NOT QUITE YET)
- Webex CC feature in development to make SIP headers visible to flows
- Straightforward mechanism for CCE/CVP to make use of
- Possible but not out the box for CCX, ...
- ... Another case for TCL app on the ingress CUBE

Time To See It In Action

What We'll Do

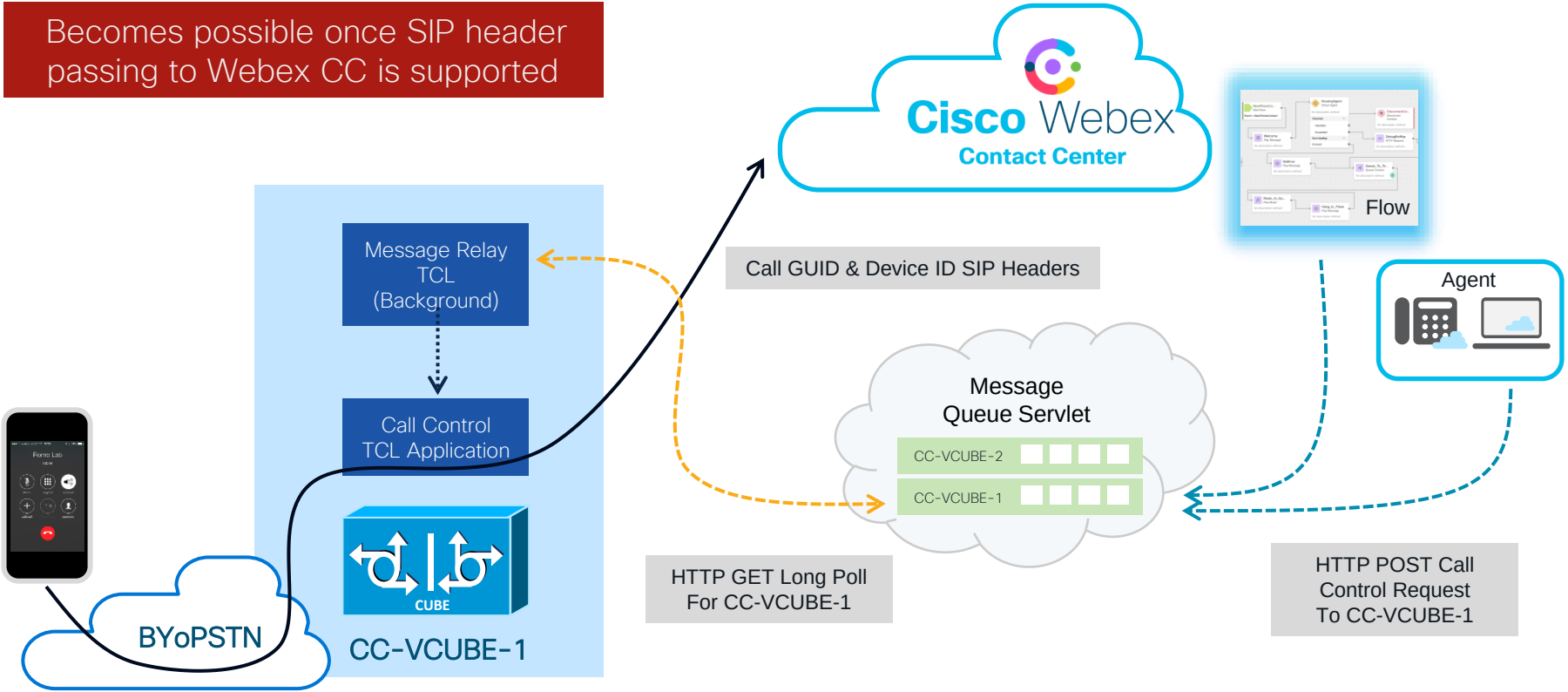
- Make a call to CCX from PSTN
- Collect some data from the caller using DTMF
- CCX script saves the context data
- Redirect the call to Webex CC through CUBE Local Gateway
- Correlate using calling number
- Webex CC flow retrieves the context data
- See the data passed from CCX on Webex CC agent desktop
- CCX IVR using female voice, Webex CC using male voice

Other Possible Hybrid Scenarios

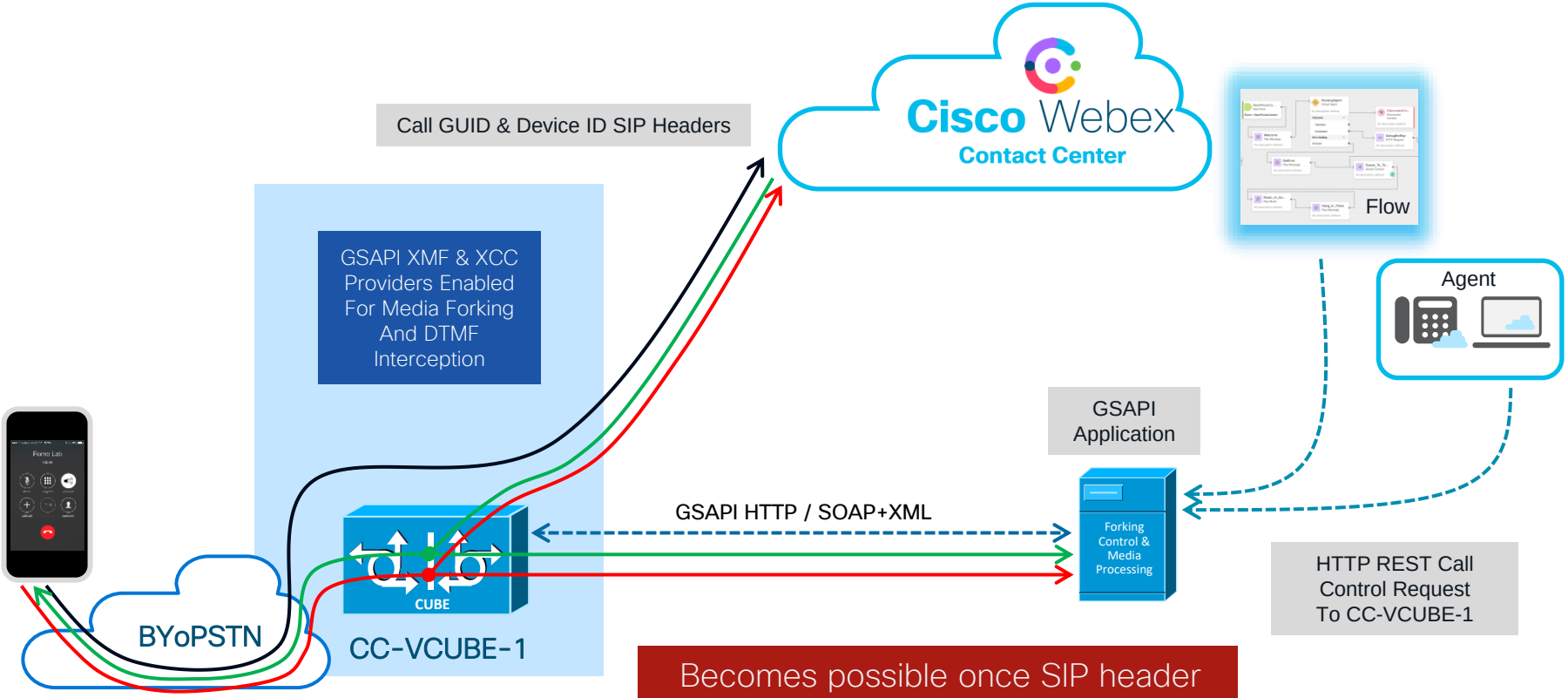
- Call control and media operations at the ingress gateway under Webex CC control – routing flow or desktop
- Things it makes possible:
  - Local media forking for recording or analysis
  - Queuing treatment at the edge
  - Call transfers at the ingress gateway
  - Temporary IVR handoff
  - DTMF interception
- Need a call ID such as Cisco-Guid SIP header and a gateway ID

Cloud Invoking Local Call Actions At Gateway

Becomes possible once SIP header passing to Webex CC is supported



Cloud Invoking GSAPI Actions At Gateway



Becomes possible once SIP header passing to Webex CC is supported

Try It Yourself

Getting Started Links

- Things from the Tindall workbench
  - Materials related to this session Samples Download
  - Twitter @tindallpaul to catch anything that's new / updated
- Node-RED https://nodered.org/
- Webex Contact Center Developer Portal
  - https://developer.webex-cx.com/
  - https://devportal.wxcc-eu1.cisco.com/

Complete your Session Survey

- Please complete your session survey after each session. Your feedback is very important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (open from Thursday) to receive your Cisco Live t-shirt.

- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Session Catalog and clicking the "Attendee Dashboard" at https://www.ciscolive.com/emea/learn/sessions/session-catalog.html

Continue Your Education

Visit the Cisco Showcase for related demos.

Book your one-on-one Meet the Engineer meeting.

Attend any of the related sessions at the DevNet, Capture the Flag, and Walk-in Labs zones.

Visit the On-Demand Library for more sessions at ciscolive.com/on-demand.

Collaboration

Cisco Contact Center

Learn about Webex Contact Center and transitioning from premise contact center to the cloud. Understand how digital channels and customer interaction automation can optimize the customer engagement experience for both cloud and premise solutions.

You Are Here

START

Feb 6 | 08:45
TECCCT-3001
Webex Contact Center Workshop: Differentiating your Customer Experience

Feb 7 | 08:30
BRKCCT-2460
Next Gen Contact Center using CCAI

Feb 7 | 14:00
LTRCCT-2011
Webex Contact Center Analyzer - Data and Analytics Lab

Feb 8 | 08:30
BRKCCT-2724
Exploring Webex Contact Center functionality and use cases

Feb 8 | 16:45
BRKCCT-3735
Intelligently Handling Call Traffic Between Premise & Cloud Contact Centre

Feb 9 | 10:45
BRKCCT-2722
Understanding Webex Connect as the platform for customer engagement using digital channels

Feb 9 | 14:00
LTRCCT-3001
Webex Contact Centre New Digital Channels Bot Capabilities

Feb 9 | 14:15
BRKCCT-2027
Contact Center Enterprise (CCE) digital channels integration powered by Webex Connect

Feb 10 | 11:00
FINISH
BRKCCT-2723
Demystifying voice connectivity and real-time media handling in Webex Contact Center

If you are unable to attend a live session, you can watch it On Demand after the event

The bridge to possible

Thank you

cisco Live!

CISCO Live!

ALL IN