

Partner: AWS

Version: 1.0

Reference: Amazon Connect configuration guide

**Sensitivity: *Public***

Amazon Connect PCI Pal Integration

11-February-2021

Table of Contents

[2 Purpose 2](#_Toc63931264)

[3 Architecture overview 2](#_Toc63931265)

[4 Integration Configuration 4](#_Toc63931266)

[4.1 Amazon Web Services 4](#_Toc63931267)

[i. AWS Lambda 5](#_Toc63931268)

[ii. DynamoDB 5](#_Toc63931269)

[4.2 AWS Amazon Connect 5](#_Toc63931270)

[iii. Generic configurations 5](#_Toc63931271)

[iv. Contact Flows 6](#_Toc63931272)

[v. Create the contact flows 7](#_Toc63931273)

[vi. DIDs 13](#_Toc63931274)

[vii. Quick Connects 13](#_Toc63931275)

[viii. Queues 14](#_Toc63931276)

[ix. Routing Profiles 14](#_Toc63931277)

[x. Agents (User Management) 15](#_Toc63931278)

[5 Amazon Connect Agent Operation during PCI Pal Payments 15](#_Toc63931279)

[5.1 How to take a secure payment 16](#_Toc63931280)

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Modified by | Modified On | Checked On |
| 1.0 | Cesar Branco | 11th February 2021 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Controlled Distribution**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Sent by/Date | Sent to | Method |
| 1.0 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Purpose

This document describes the architecture and necessary configurations on the Amazon Connect tenant (the common PCI Pal and Amazon Connect customer) to integrate with the PCI Pal service and allow the agents to take secure (PCI) payments over Amazon Connect calls, maintaining constant communication with the cardholder (end customer on the phone), without putting that end customer on hold during payment or transfer to IVR.

PCI Pal will intercept and remove the credit card digits from the audio stream, in real time and without delays, during the agent and cardholder conversation, execute the payment on your (the customer) behalf, while informing the agent of what is happening throughout so she/he can guide the cardholder during the payment.

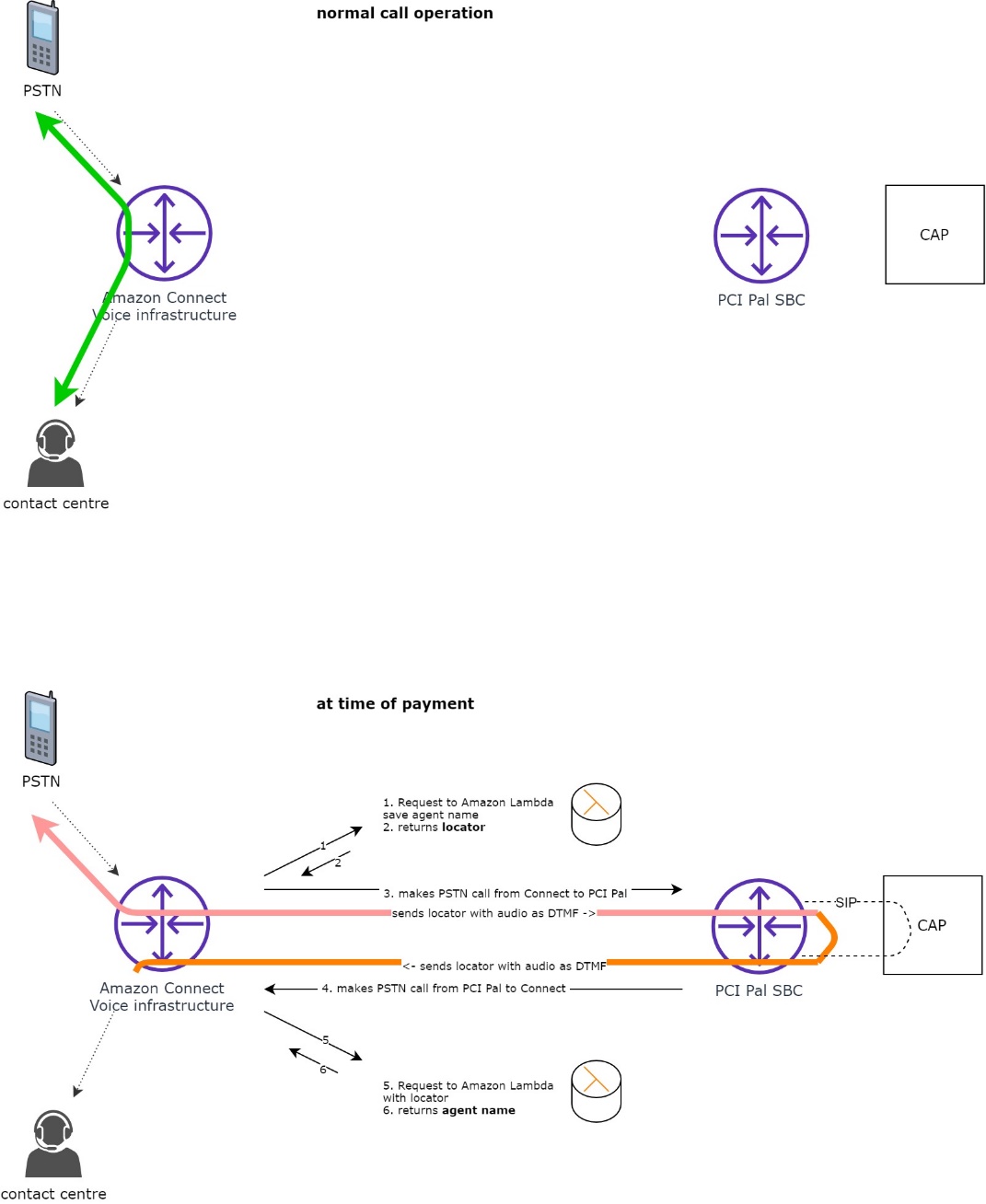
# Architecture overview

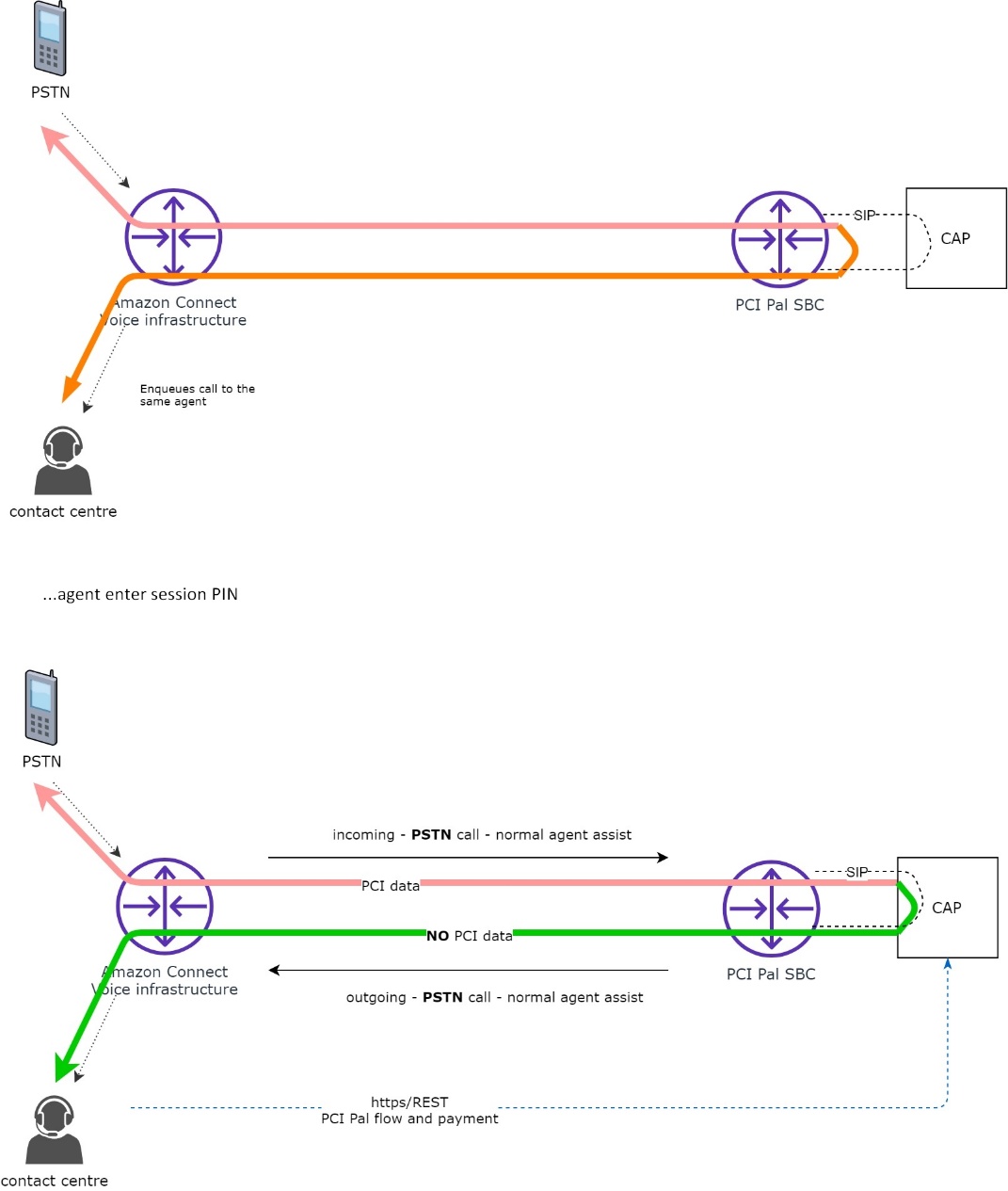
For this integration architecture it is relevant to be aware that during normal call operation PCI Pal does not have any visibility of the Amazon Connect calls. Only which agents are logged in to PCI Pal’s portal.

At time of payment the agent will transfer the call to PCI Pal the “PCI Pal secure payment” “quick connect” contact on the agent’s soft phone.

1. invoke an AWS Lambda function that will pass the agent name and receive a locator (4 digits).
2. make a PSTN call to a PCI Pal DID associated with the customer (your tenant). PCI Pal will communicate this DID.
3. PCI Pal will make a PSTN call to an Amazon Connect DID associated with the customer payment service and an Amazon Connect **merchant** “contact flow”, creating a 2-call loop through PCI Pal.
4. when the call is answered by the **merchant** flow, the **cardholder** flow will send the locater PIN as DTMF tones.
5. the merchant “contact flow” will read the locator PIN and use that PIN to invoke an AWS Lambda function that will return the agent’s username.
6. the merchant flow will use the agent name to enqueue the call to the same agent the cardholder was talking to.
7. the agent disconnects the original call on its side and is connected to the new, secure, call - where it can continue speaking with the cardholder who was never disconnected.
8. now the agent and the cardholder can continue as an “agent assist” call, where the credit card details are cleaned from the merchant leg using the PCI Pal payment flows and gateway integrations as usual.

See sequence below:





# Integration Configuration

The integration requires adding the Lambda functions and DynamoDB table to the merchant’s AWS account (using the Quick Start), making the functions available for Amazon Connect and some other simple configurations, and creating the route(s) on PCI Pal’s portal (done by PCI Pal and communicated to you the customer during the deployment project).

## Amazon Web Services

The required AWS services are deployed automatically by the AWS Amazon Connect Quick Start CloudFormation script provided by PCI Pal on the Quick Start page.

### AWS Lambda

There are 2 lambda functions to be used:

AddAgent: saves the agent name before the 2-step transfer and provides a locator.

input: the agent’s Amazon Connect username.

returns: a 4-digit locator (or error)

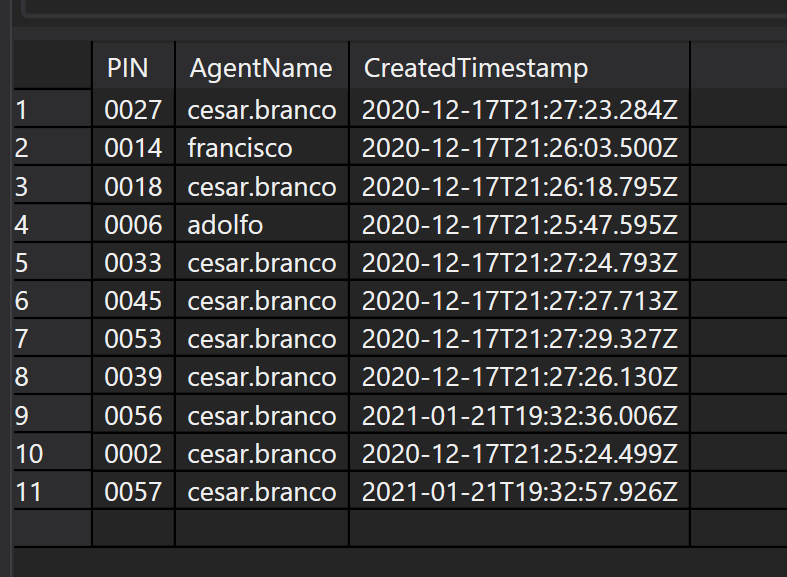
GetAgent: retrieves the agent name using the locator after the call loops through PCI Pal.

input: a 4-digit locator

returns: the agent’s Amazon Connect username (or error)

### DynamoDB

There is 1 DynamoDB table that stores agents' names, locators and timestamps for the secure call request (this is not visible to the agent or Amazon Connect – described here just for information purposes):

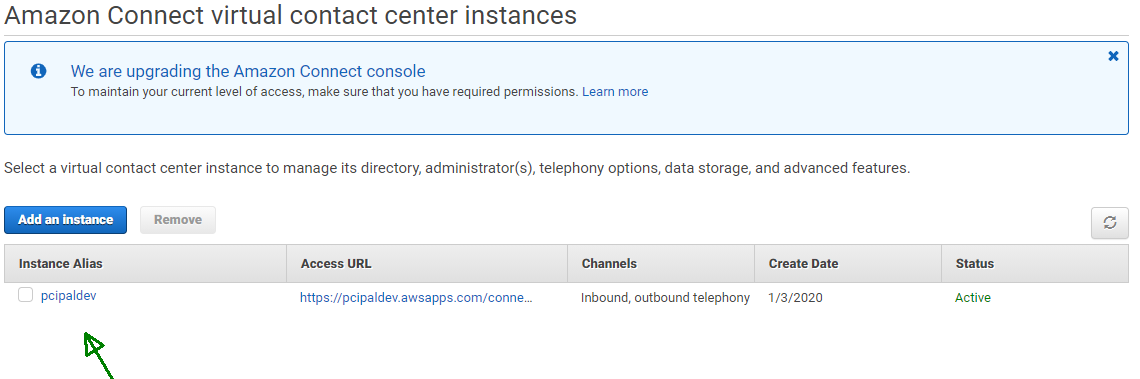


These rows only live for a few seconds, the time to transfer the call through PCI Pal for secure payment and that call being answered by Amazon Connect before enqueuing to the same agent.

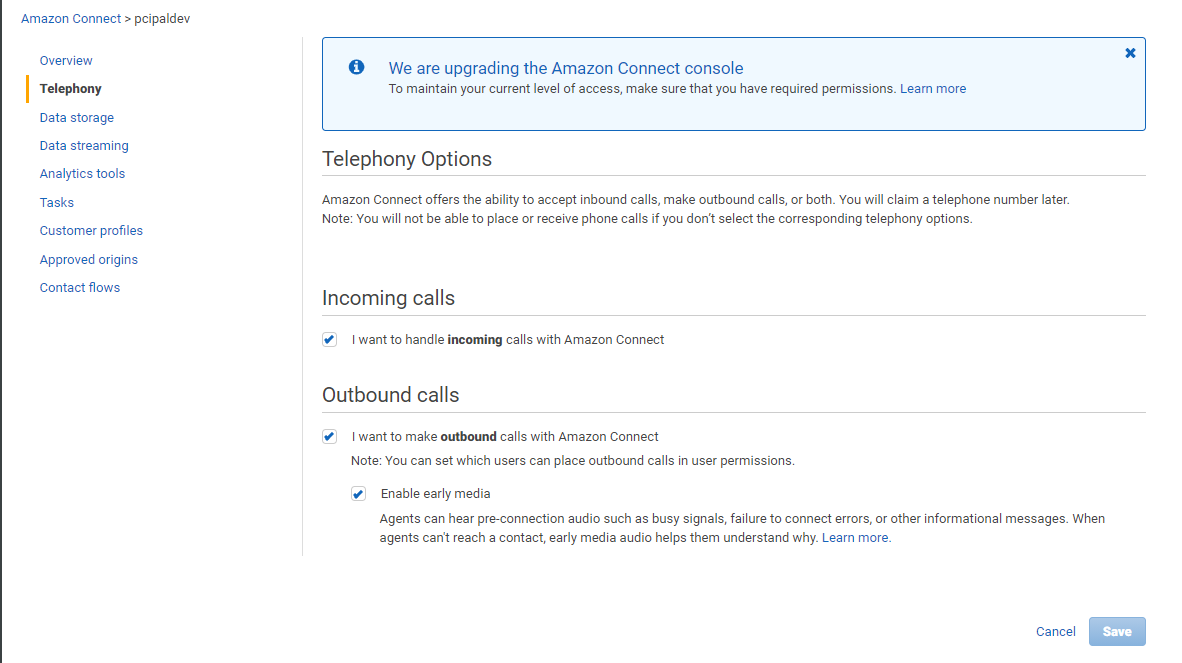
## AWS Amazon Connect

### Generic configurations

Open the Amazon Connect system configurations:



And click **telephony** to enable inbound and outbound calls:

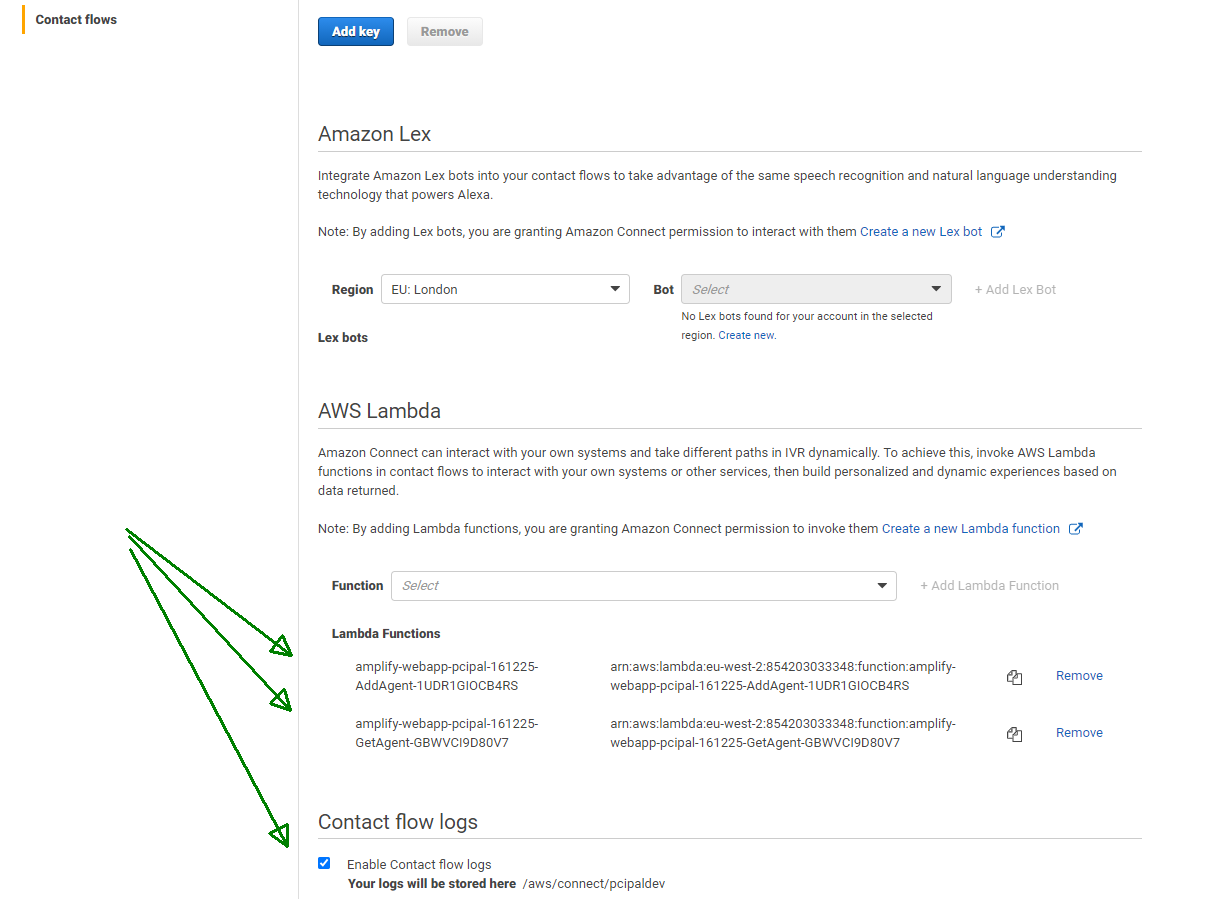


Then click contact flows and select two of the pre published Lambda functions, that will make them available while configuring the call centre logic:

<something>**AddAgent**<some GUUID>

<something>**GetAgent**<some GUUID>

and enable the logs (Contact Flow Logs) for debugging.



### Contact Flows

Contact flows define the contact centre logic and behaviour for ACD, queues, IVR, transfer and hold. The PCI Pal integration requires only three flows as defined in this section.

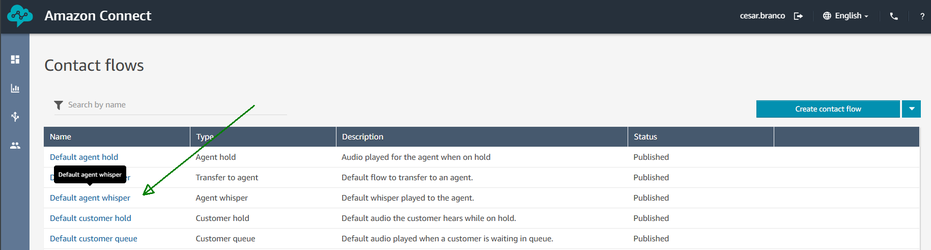
<https://docs.aws.amazon.com/connect/latest/adminguide/create-contact-flow.html>

|  |  |
| --- | --- |
| **Type** | **When to use** |
| Inbound contact flow | This is the generic contact flow type that is created when you choose the **Create contact flow** button, and don't select a type using the drop-down arrow. It creates an inbound contact flow.  This contact flow works with voice, chat, and tasks. |
| Agent whisper flow | Use to manage what the agent experiences as part of an inbound call immediately before being joined with a customer. The agent and customer whispers are played to completion, then the two are joined.  This contact flow works with voice, chat, and tasks. |
| Transfer to queue flow | Use to manage what the agent experiences when transferring to another queue. This type of flow is associated with transfer to queue quick connects, and often plays messaging, then completes the transfer using the **Transfer to queue** block.  This contact flow works with voice, chat, and tasks. |

### Create the contact flows

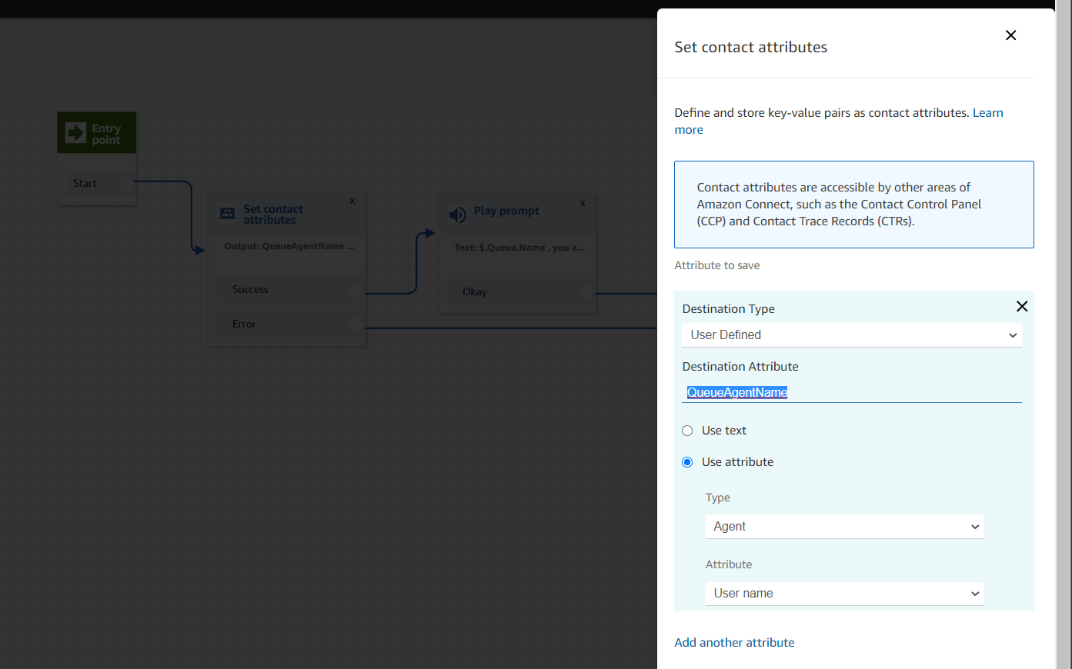
#### Save agent name - Agent Whisper Flow

This flow will store the username of the agent handling the call just before she/he answers it. We will use the default, however it can be changed from another contact flow, so in that case change the non-default flow.

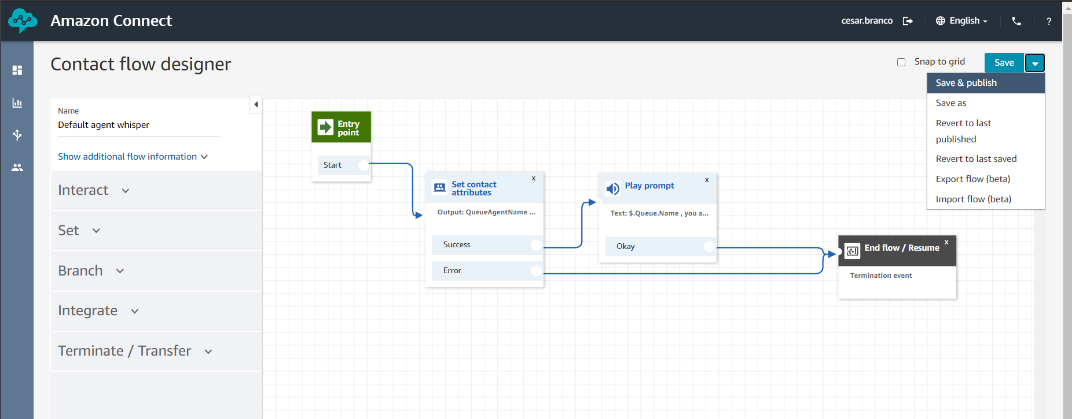




We’ll store the agent username as a user defined Attribute, named **QueueAgentName** that we will refer to when the call is 2 step transferred to PCI Pal, using the “Set contact attributes” block:



The play prompt contact block is not necessary, it is just for debugging purposes. Save and publish:



Exported flow (link to github):

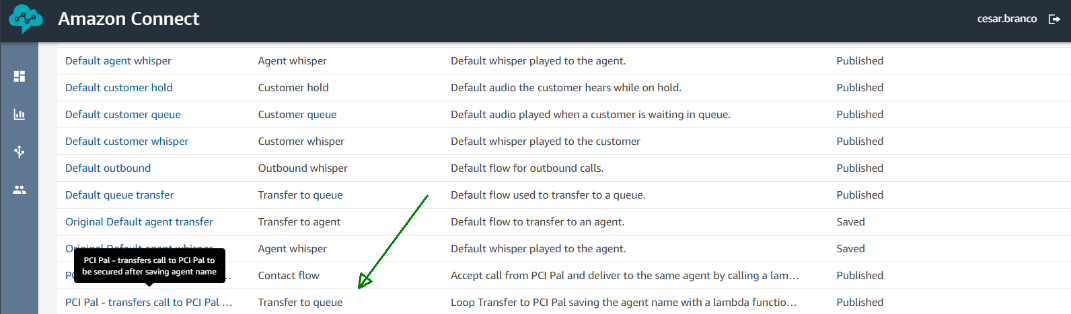


<https://docs.aws.amazon.com/connect/latest/adminguide/default-agent-whisper.html>

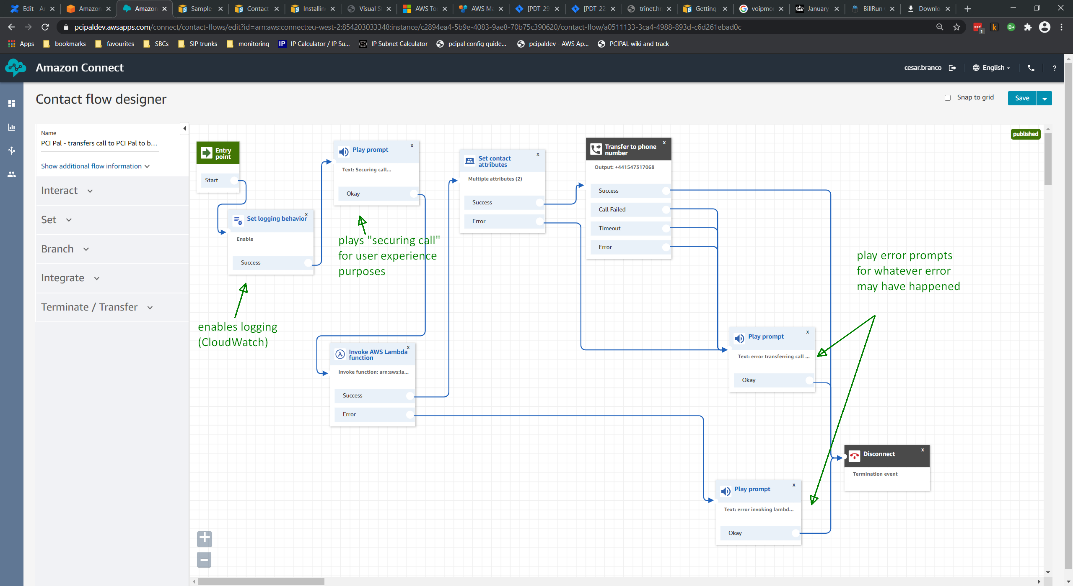
<https://docs.aws.amazon.com/connect/latest/adminguide/set-whisper-flow.html>

#### Transfer Call to PCI Pal - Transfer to Queue Flow

The Queue Transfer Flow will be associated with the “quick connect” the agent will use for the 2 step transfer that loops the call through PCI Pal, and will store the agent user name, retrieve a locator, **transfer** the call to a PSTN DDI (that resides at PCI Pal’s platform) **and** send the locator as a DTMF sequence to be retrieved by the Amazon Connect flow that answers the call as it loops.

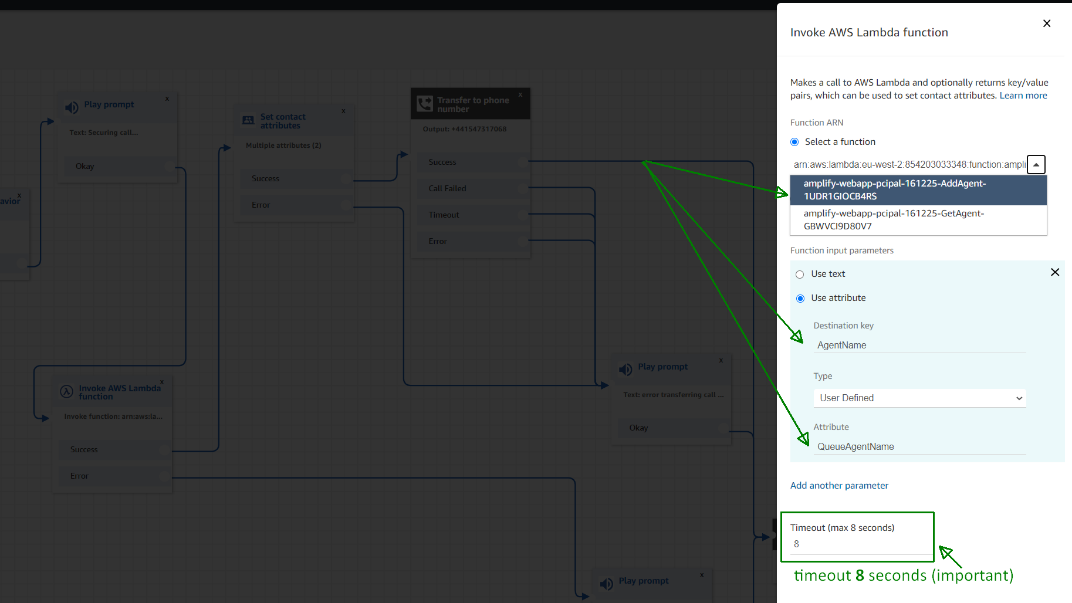


Full flow (name and description set on the upper left corner):



“Invoke Lambda Function”: select the **AddAgent** function, then pass the “**QueueAgentName**”attribute saved on the whisper flow as the Lambda function parameter “**AgentName**”.

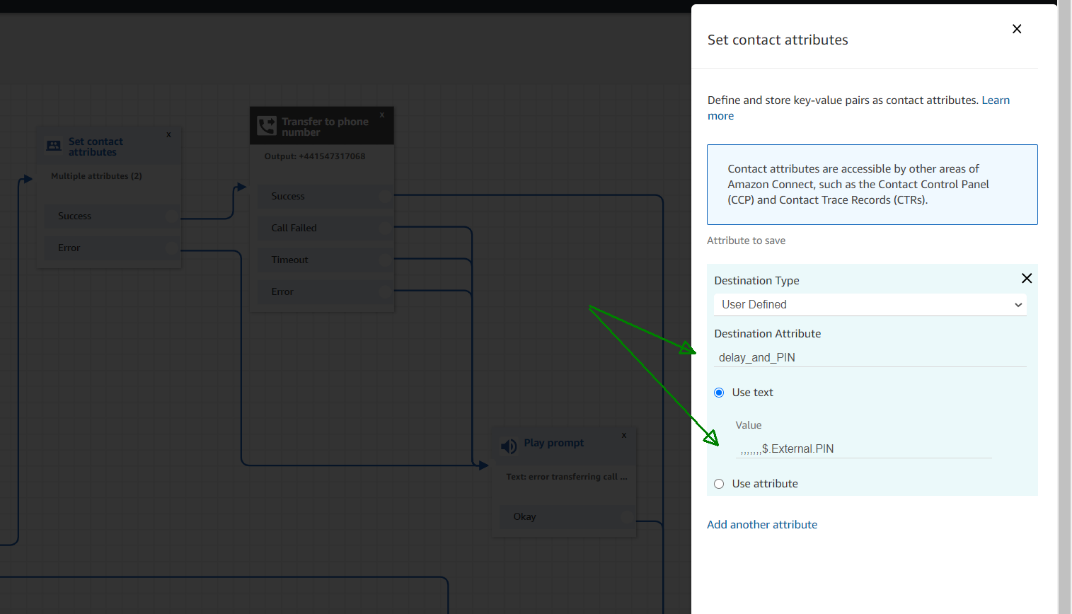
|  |
| --- |
| **NOTE:** |
| Set the timeout to **8** seconds, that is important because sometimes the function takes 3-4 seconds to “wake-up” if it has not been called in a while and so will timeout and raise an error, typically the response time is < 200 ms. |



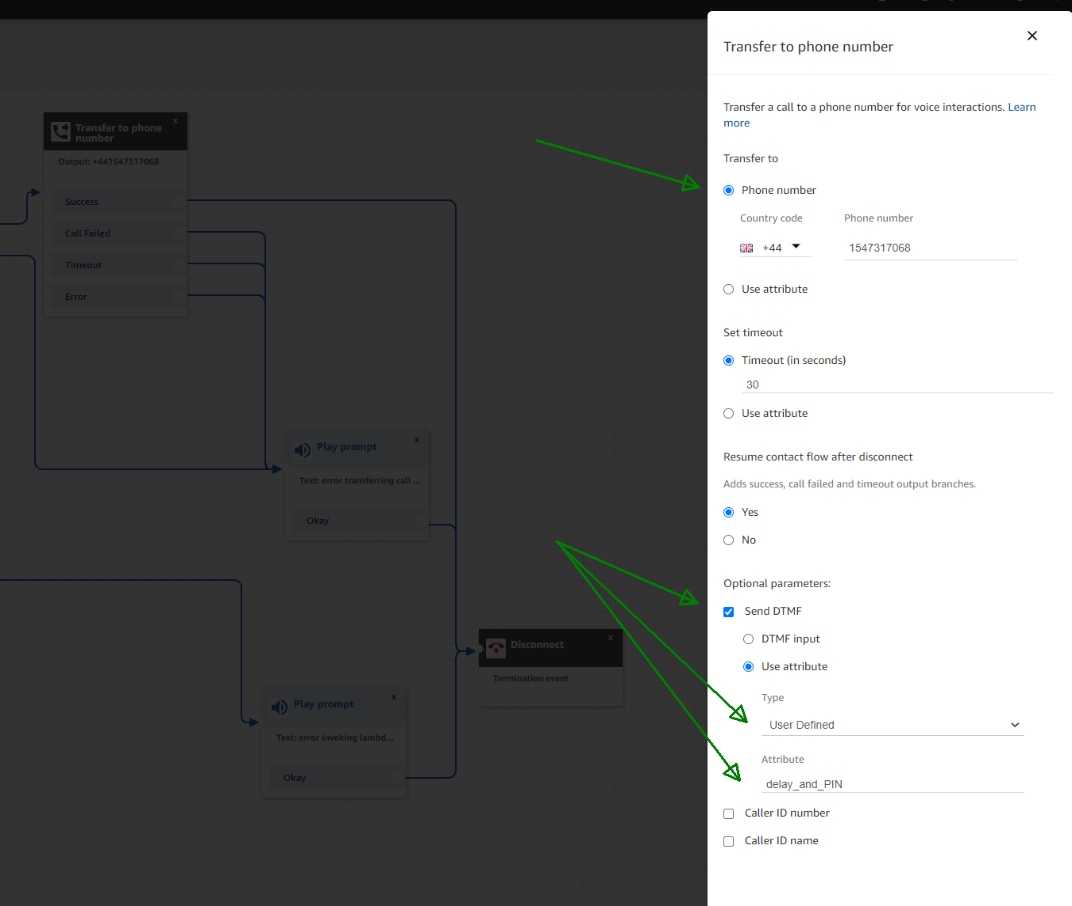
“Set contact attributes”: creates an **User Defined** attribute named “**delay\_and\_PIN**” to be used by the transfer node to send as DTFM to be read by the Contact Flow that answers the looped call. The value is:

**,,,,,,,$.External.PIN**

where the commas represent 1 second of delay each (,,,,,,,, it can/should be reduced) and **$.External**.**PIN** refers to the **PIN** attribute returned by the Lambda function invoked before.



“Transfer to phone number”: this block executes the transfer to PCI Pal’s platform and the route DDI created for the customer’s integration. The **Phone number** is the PCI Pal route DDI (configured ahead) that our team will supply the customer. For the “Optional parameters” enable “**Send DTMF**”, “**Use attribute**”, Type “**User Defined**“ and Attribute “**delay\_and\_PIN**” as created before:



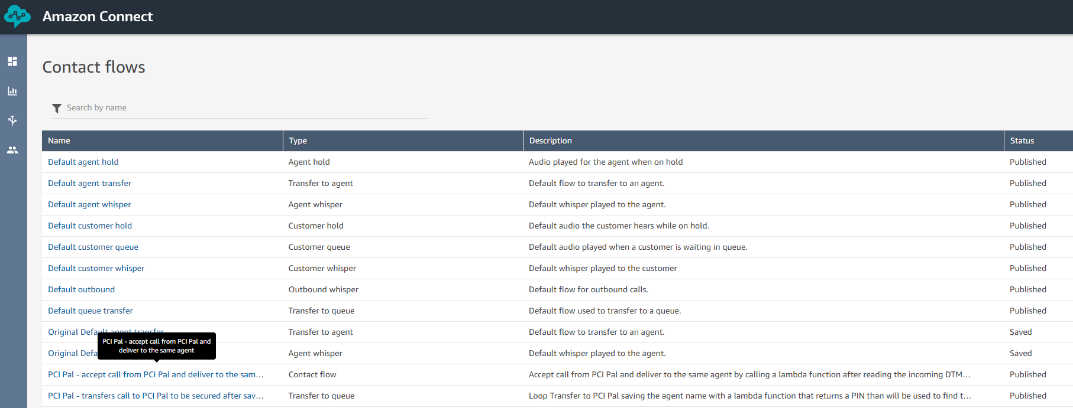
And this is the extent of the transfer to PCI Pal flow. Save and publish.

Exported here (link to github):

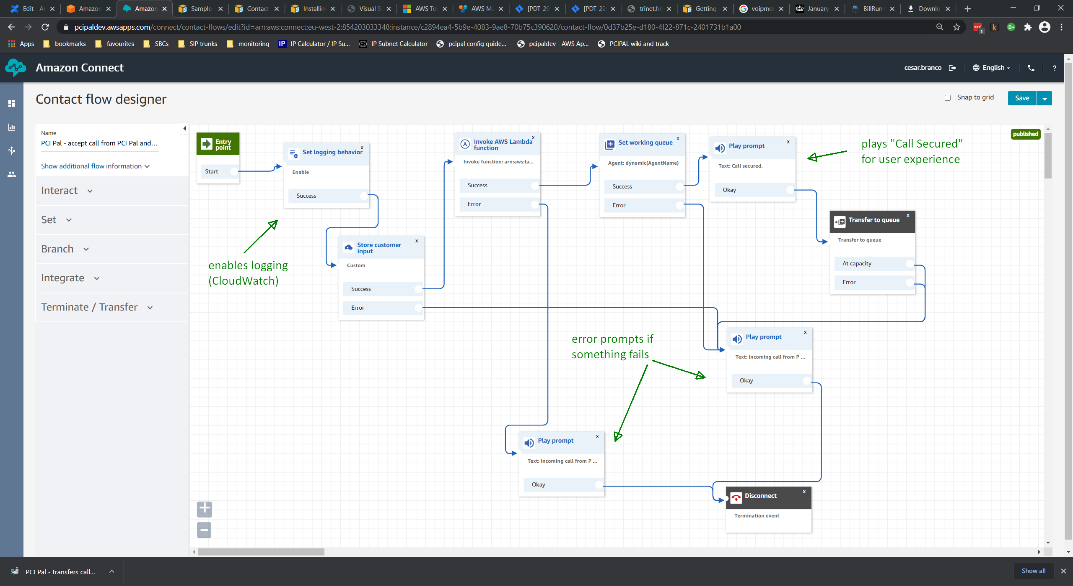


#### Answers Call from PCI Pal - Contact Flow

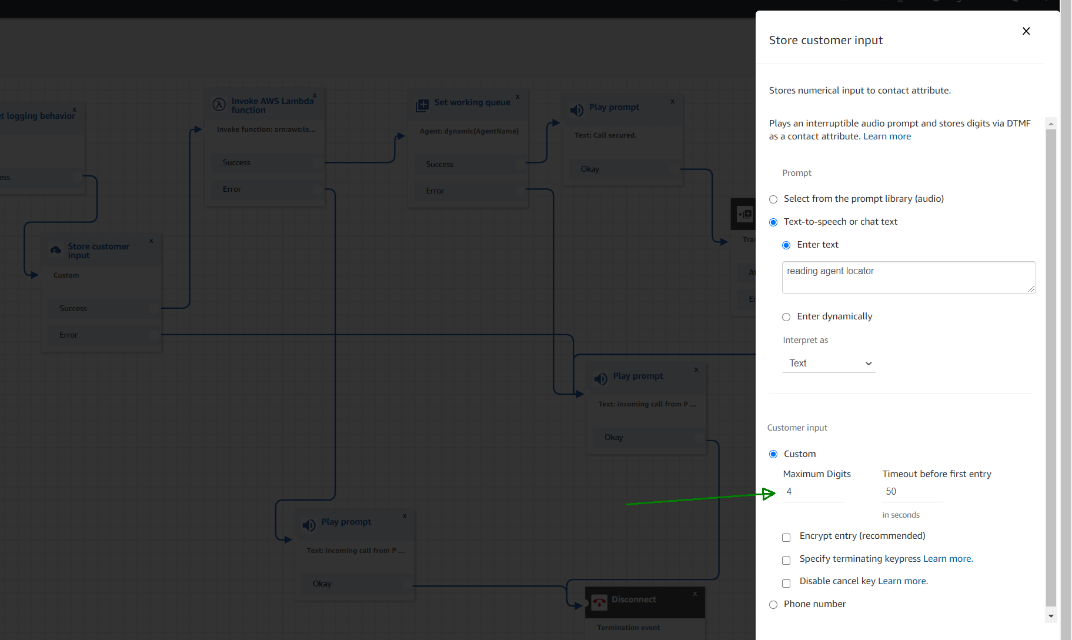
The “Answers Call from PCI Pal“ Contact Flow will be answering the returning call (outgoing) from PCI Pal, completing the loop and will be associated with an Amazon Connect DID (as explained in the **DIDs** section ahead). This flow listens for the DTMFs sent by the “Transfer Call to PCI Pal” Transfer to Queue Flow, uses another Lambda function to get the agent username and then enqueue the call **directly** to that agent - taking **precedence** over any other call that may be on queue.



Full flow:



“Store customer input”: listens and stores the DTMFs sent by the transfer flow. Set “**Maximum** **Digits**” to **4** and timeout to 30-50 seconds depending on the PSTN connection times. Add some announcement saying for user experience purposes if required like “…reading agent locator”. It helps debug and keeps the agent informed at the beginning. But likely will not be used in production.

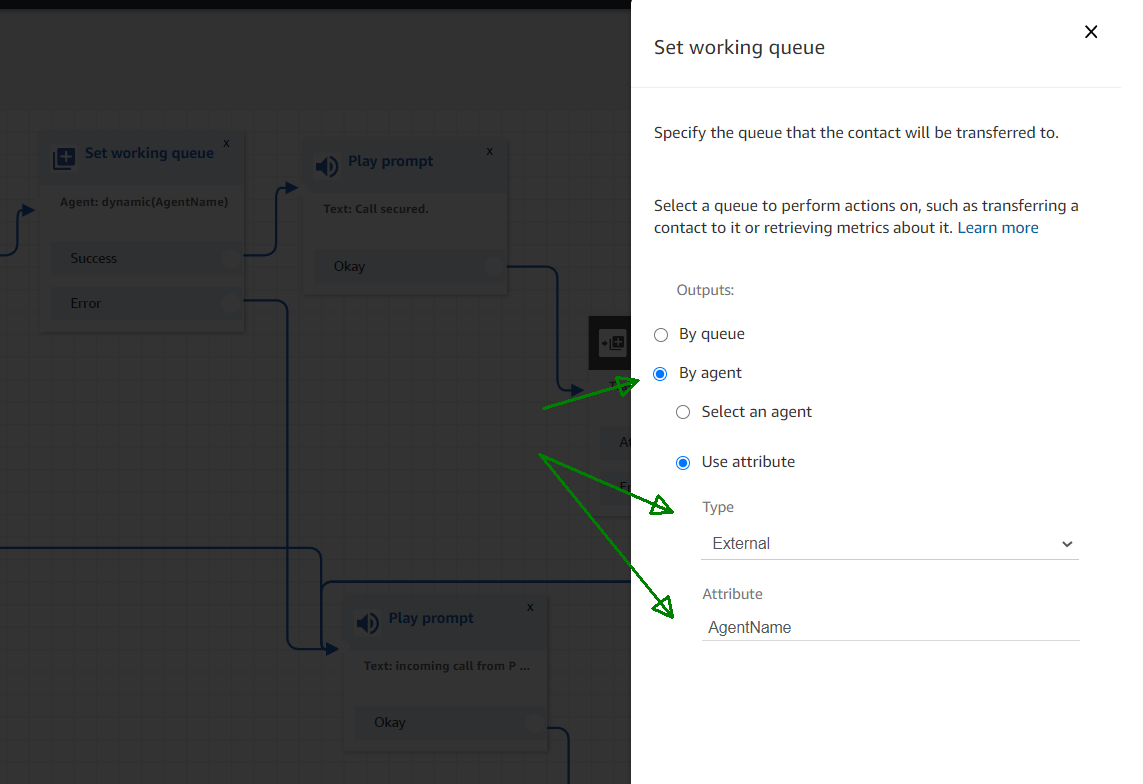


“Invoke Lambda function”: this block uses the locator retrieved on the previous block and sends it to the Lambda function to find the agent username to enqueue the transferred call to (i.e. the same).

For the function select **GetAgent**, for the function input parameters select “use attribute”, then “Destination key” “**PIN**”, which is the name of the input variable recognized by the function. Type “**System**” and “Attribute” is “**Stored customer input**” which is the DTMF sequence just collected. Timeout **8** seconds, same as on the transfer flow.



“Set working queue”: and finally set the working queue which will not be a regular queue, but the agent herself/himself, so the looped call has priority over every other call. Select “**By agent**” and then “Use attribute”, “Type” “**External**” and “Attribute” “**AgentName**” which is the name of the attribute returned by the Lambda function.



Save and publish.

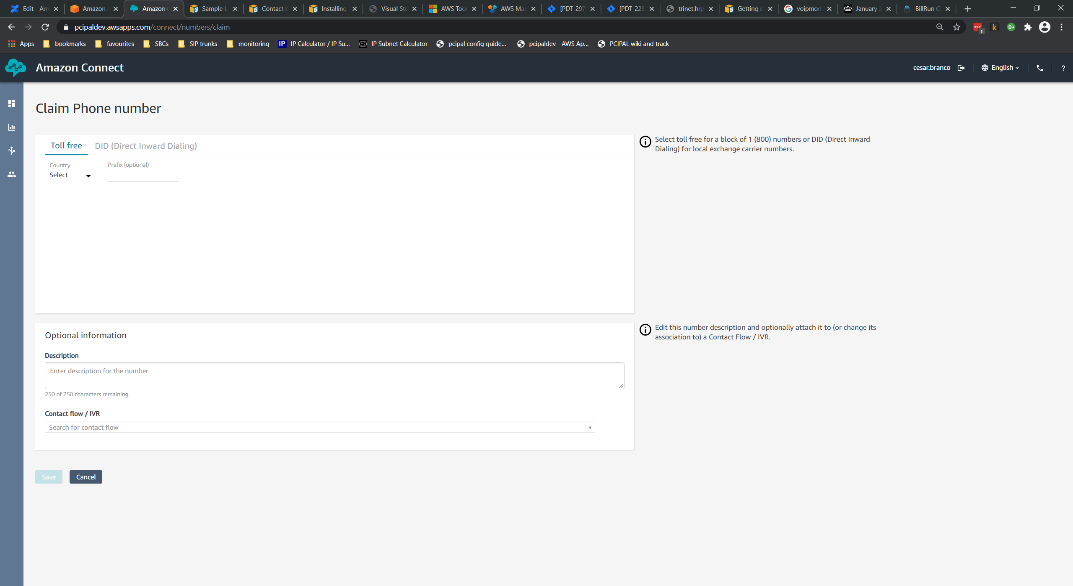
Exported here (link to github):



### DIDs

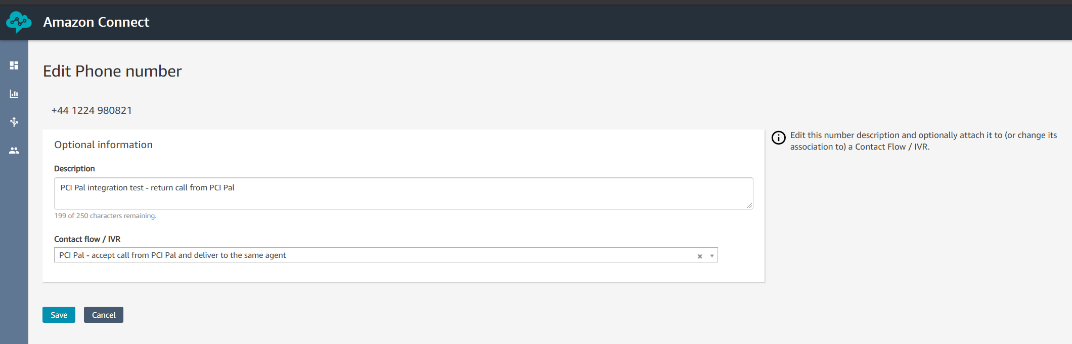
PCI Pal requires a DID to be used by the returning leg of the loop (i.e., the outgoing call to the agent) to answer the call using the “**Answers Call from PCI Pal**” Contact flow. This is not a public DID in the sense that it is only used by PCI Pal, not the cardholder.

First claim (buy) a DID or select one already created:



Note that AWS only allows customers to buy DIDs for the region(s) where the Amazon Connect instance is running.

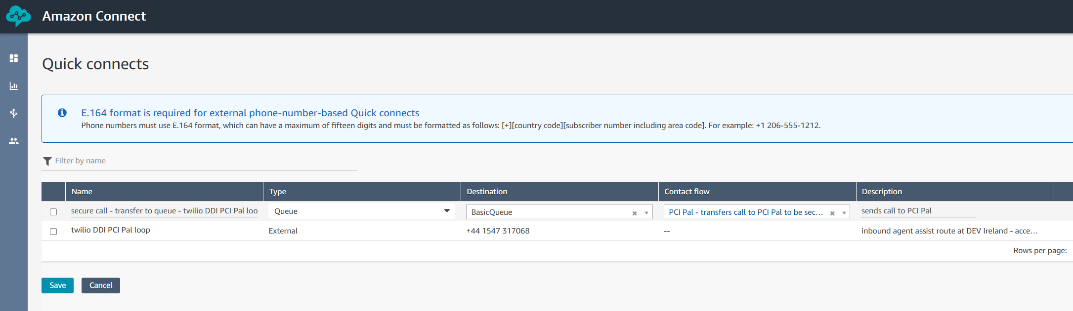
Then configure the DID by giving it a description and selecting the flow that answers the call:



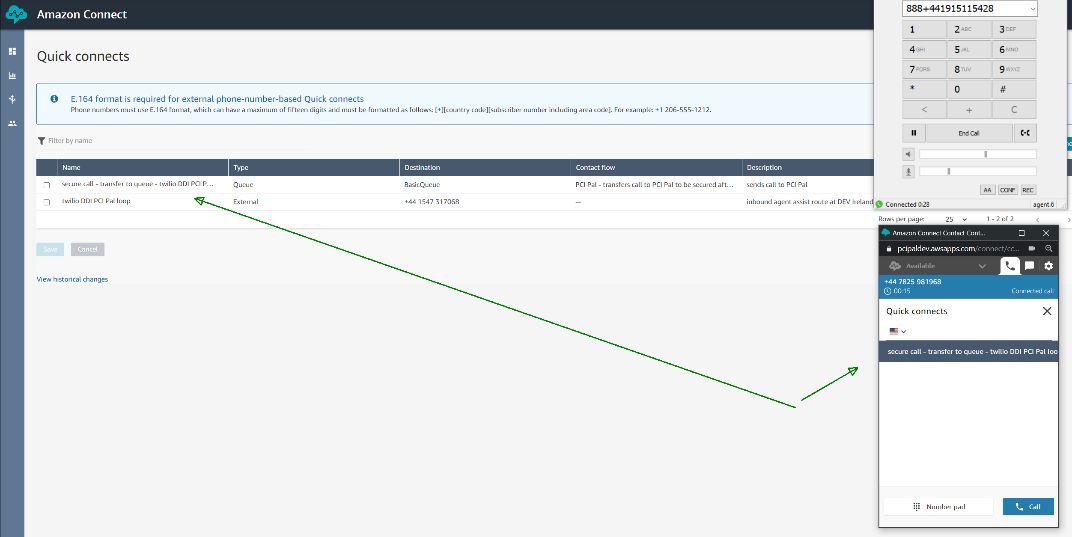
### Quick Connects

A Quick Connect is a contact that will be available to the agent to click and transfer the call to when she/he needs to secure a payment. This Quick Connect is associated with a flow (transfer to queue flow) and one or more queues (which determines which agents see which Quick Connects). The queue defined on the actual Quick Connect has no impact on the integration because the call is eventually transferred not to a queue, but to a phone number - so pick one.

Depending on the customer there may be multiple “Quick Connects” that secure calls for different queues for different regions for example.



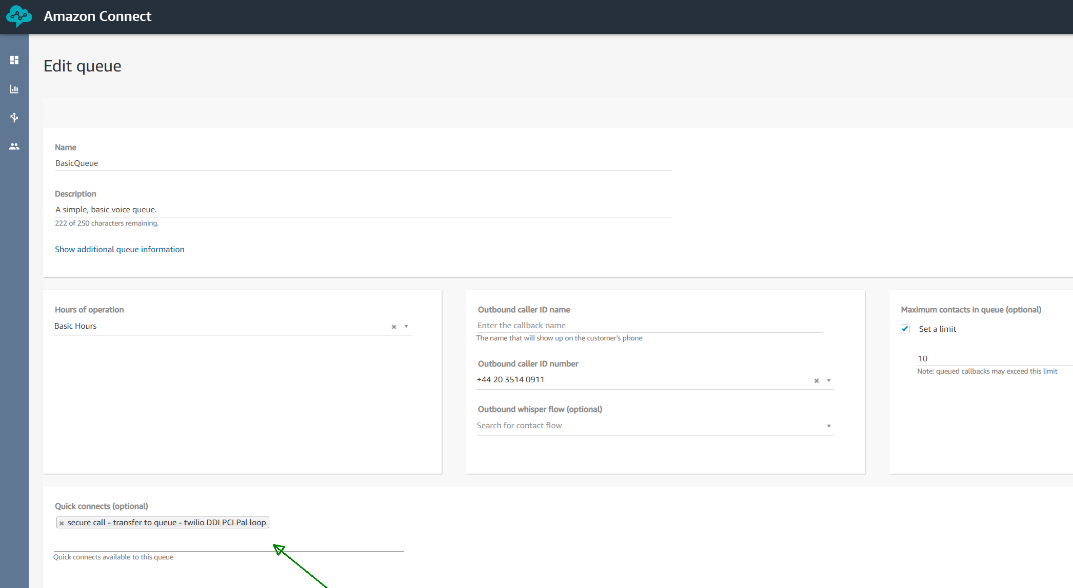
Create a “Quick Connect” suitably named, “Secure Call” or something similar, and associate with the “Transfer to PCI Pal” transfer to queue flow created before.



The “Quick connects” only show **during** call established.

### Queues

For all the queue(s) the agents may take payments on, associate the relevant secure call “Quick Connects”.



As noted before, depending on the customer, there may be multiple “Quick Connects” that secure calls for different queues for different regions for example.

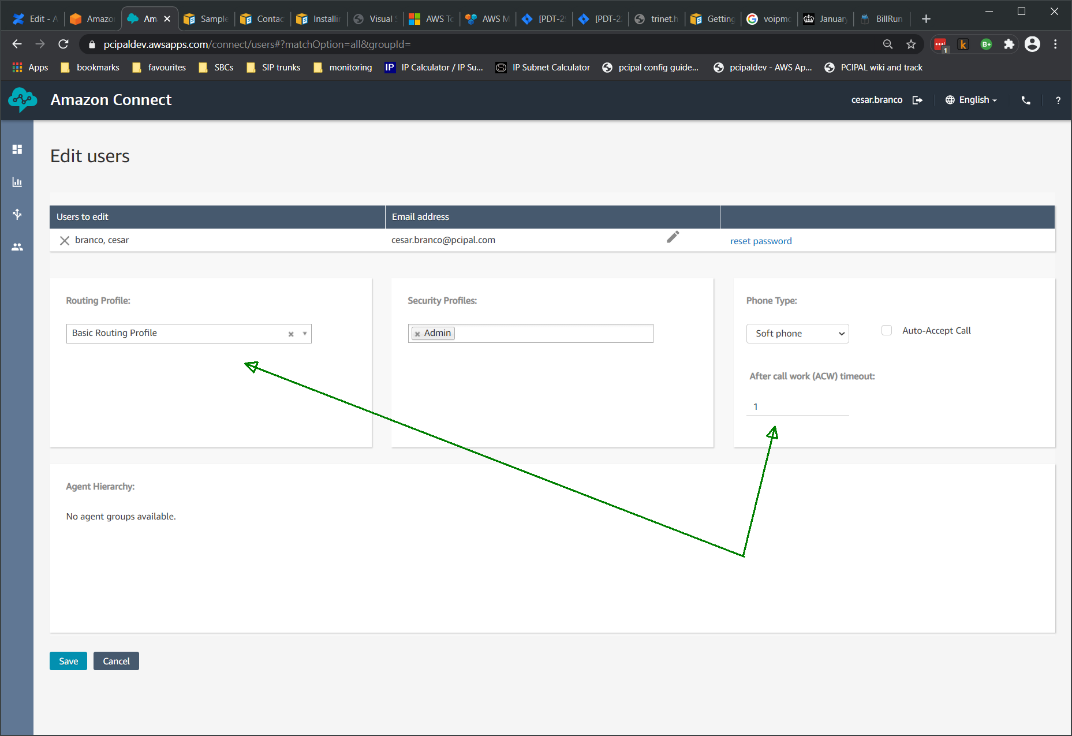
### Routing Profiles

Check the agent routing profile to make sure the relevant queues are there. In our case just “Basic Queue”:



### Agents (User Management)

Finally, for the agents confirm they are associated with the correct routing profile (the one that has the correct queues, which are the ones that have the correct secure call quick connects…).



|  |
| --- |
| **NOTE:** |
| The “After call work (ACW) timeout” value is the time the system gives the agent to finish her/his work after a customer disconnects before another call is delivered, for this example we configured **1 second**; if **0** is configured then the agent will not receive another call (aka contact) until she/he “Closes contact” on the soft phone. |

# Amazon Connect Agent Operation during PCI Pal Payments

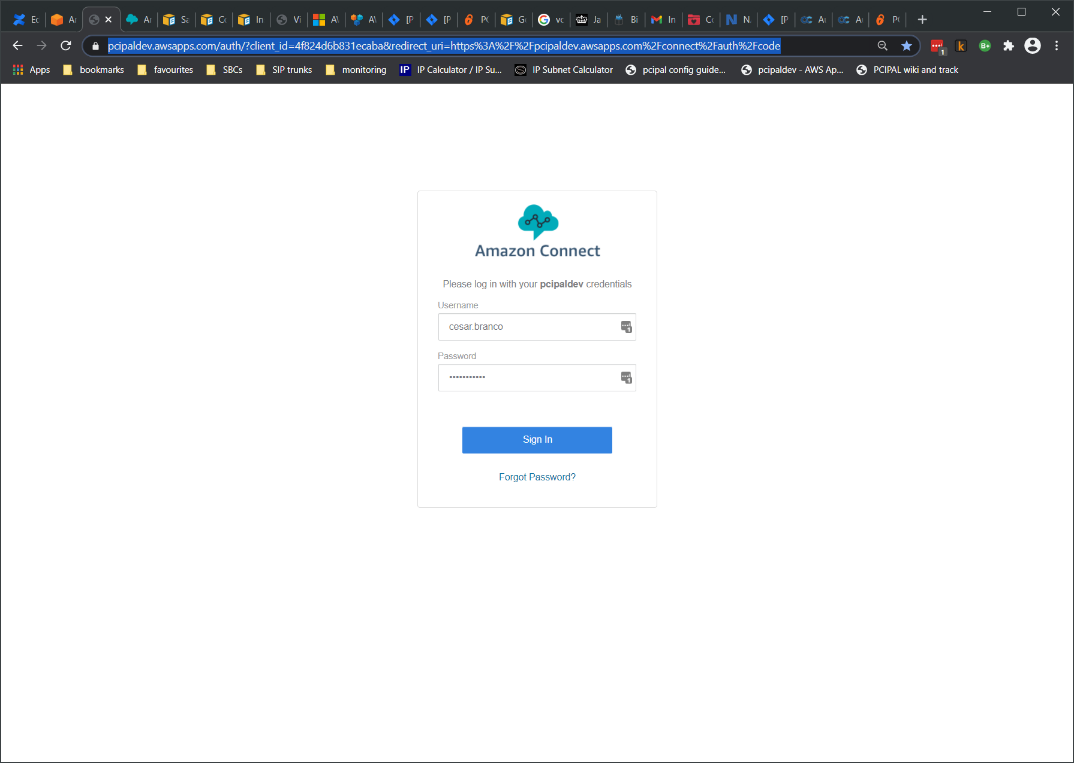
Read the [Agent training guide on AWS >>](https://docs.aws.amazon.com/connect/latest/adminguide/agent-user-guide.html)

for generic guidance on agent operation (it is not complex).

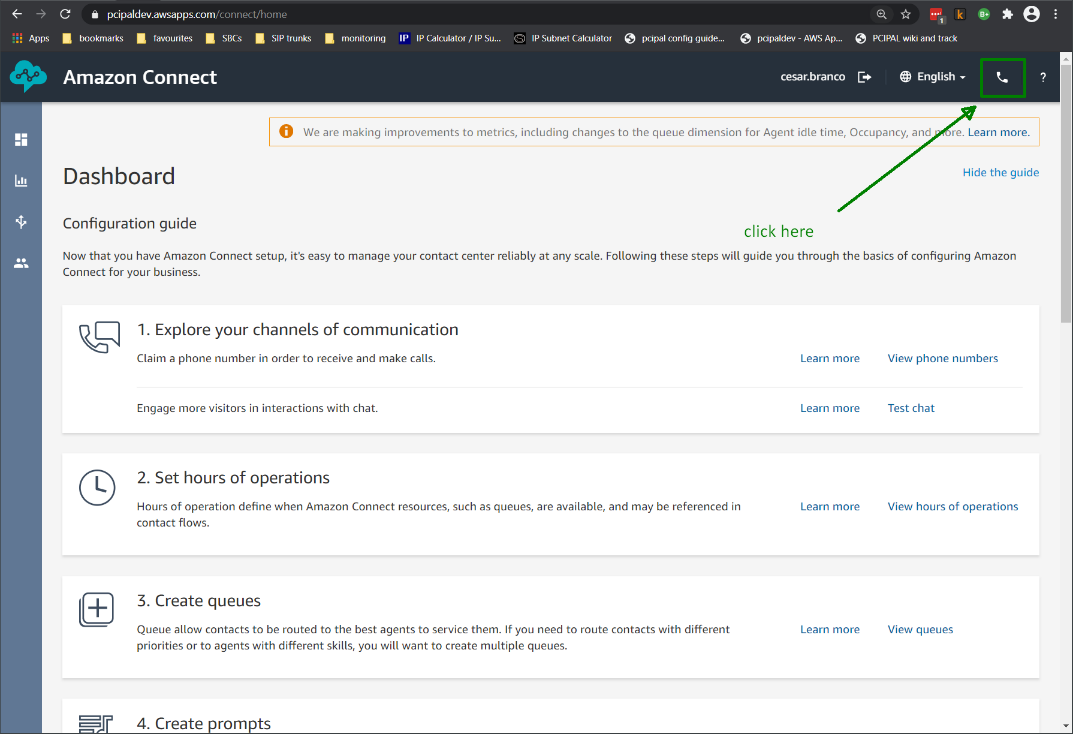
## How to take a secure payment

This section describes what the agent must do to secure a call and take a PCI compliant payment.

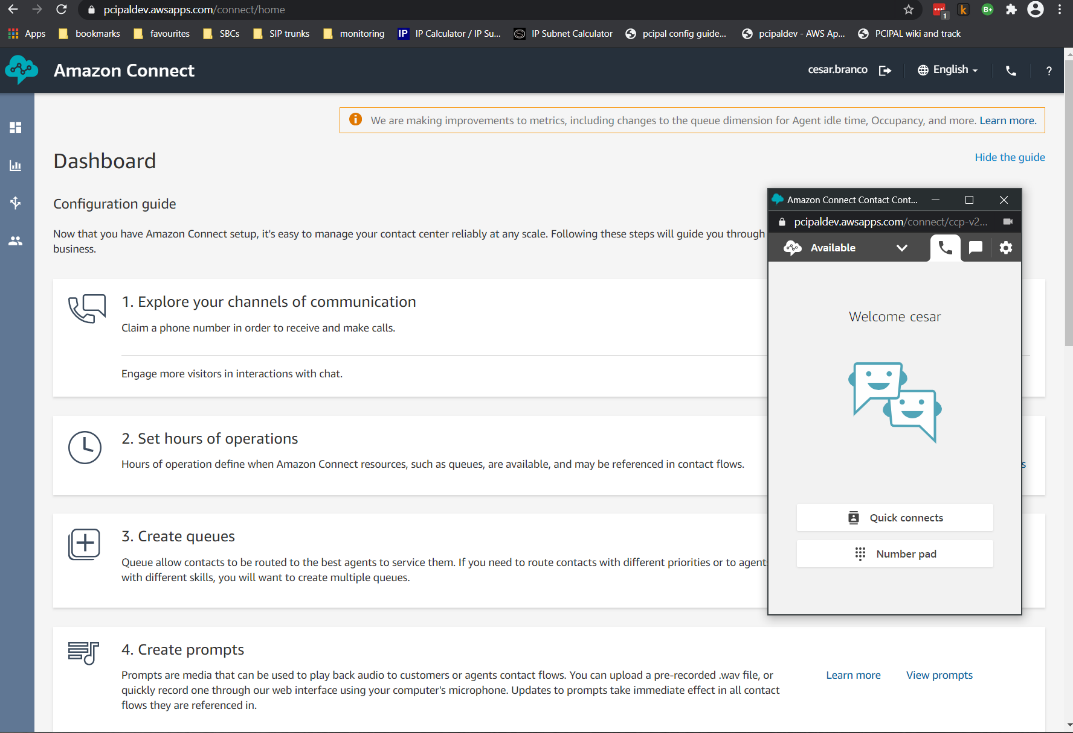
Open the Amazon Connect console: [https://<your connect tenant>.awsapps.com/connect/login](https://pcipaldev.awsapps.com/connect/login):



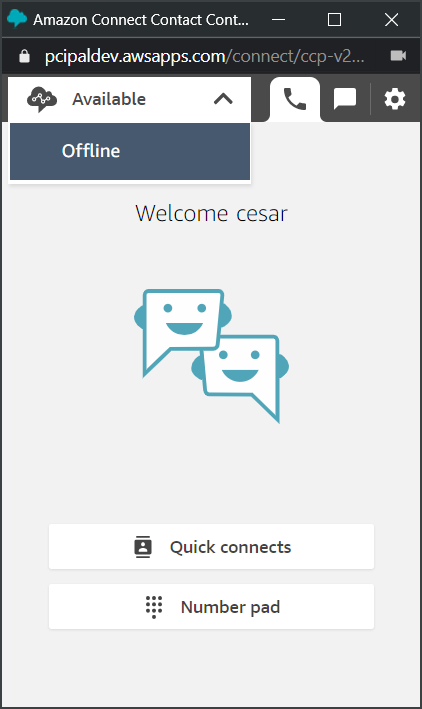
Open the agents CCP (contact control panel):



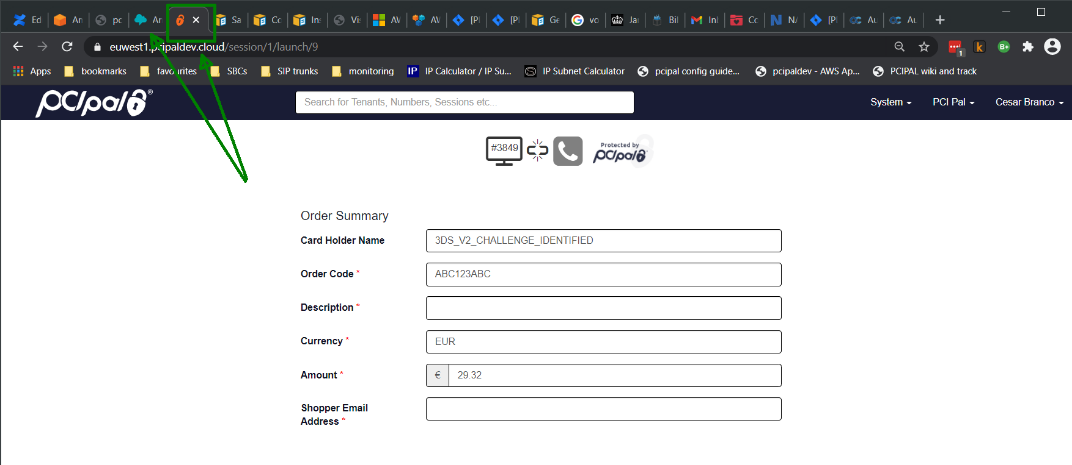
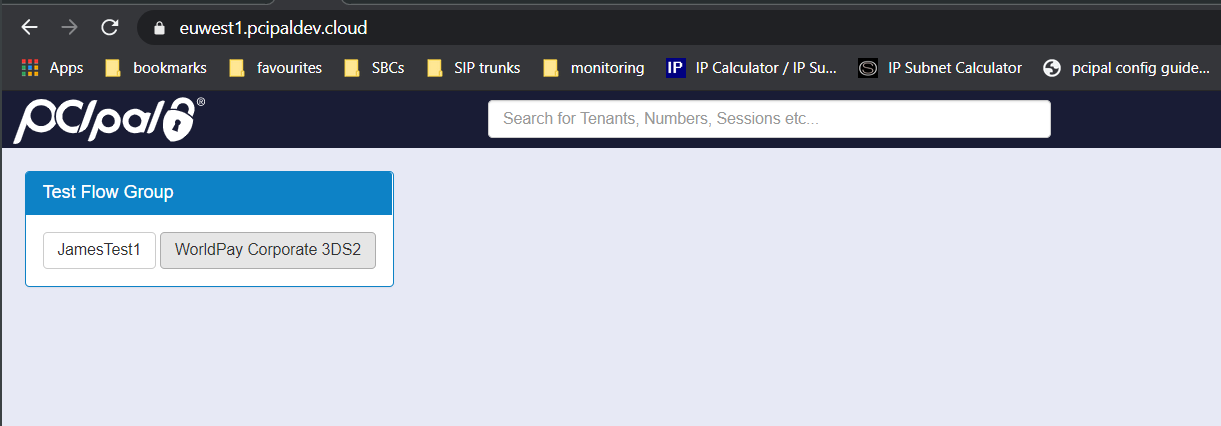
The CCP pops up (this is a WebRTC soft client whose URL is [https://<your connect tenant>.awsapps.com/connect/ccp-v2/softphone](https://pcipaldev.awsapps.com/connect/ccp-v2/softphone)):



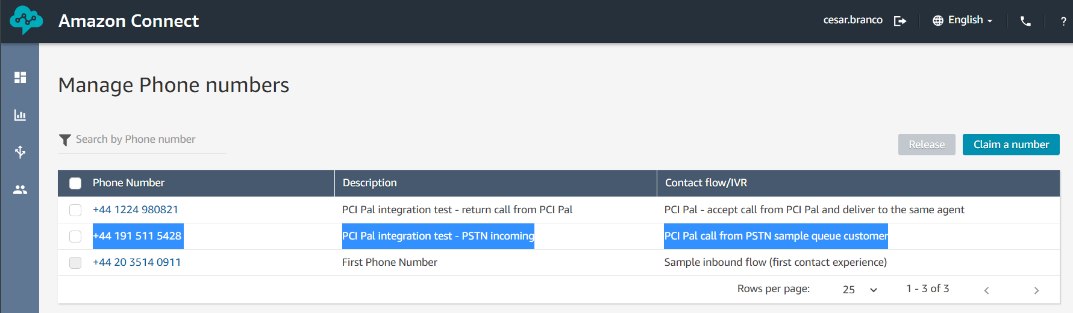
Set the agent available:



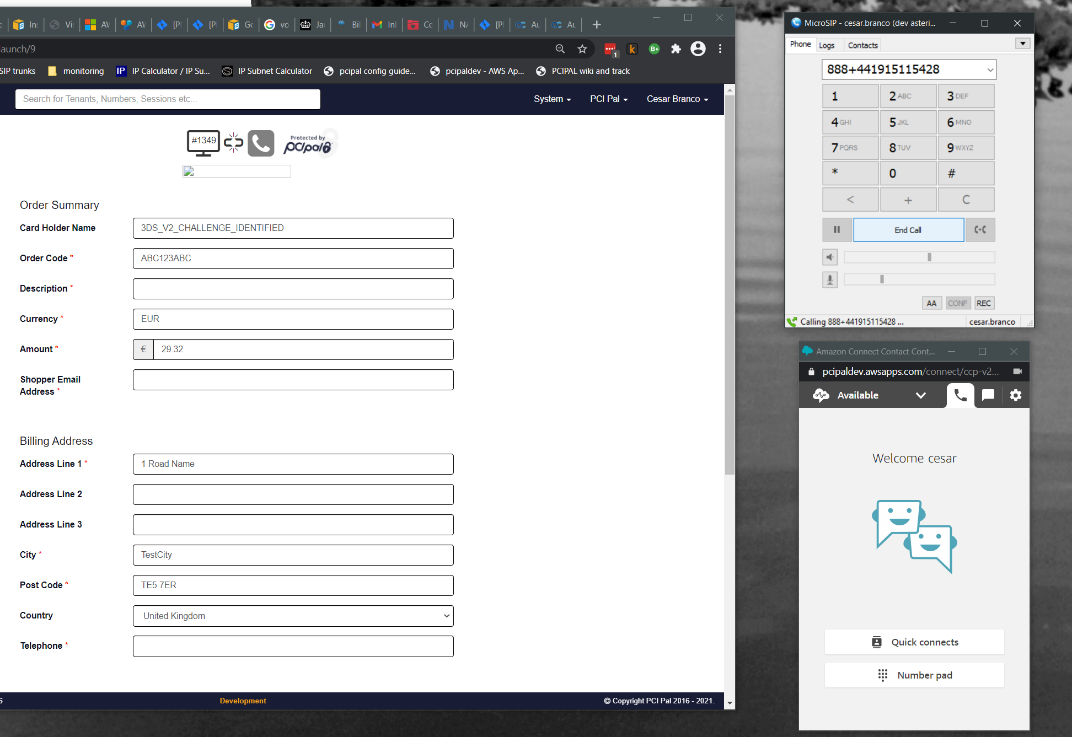
Open the flow the agent will use to take the payment (it will be on a separate tab as usual). Production deployments will have the CRM integrated with the flows and sometimes even the CCP together with the CRM:



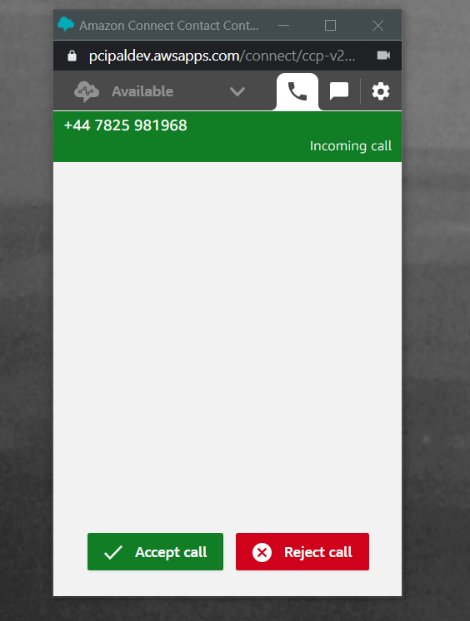
Then make a call to the PSTN number visible to the cardholder:



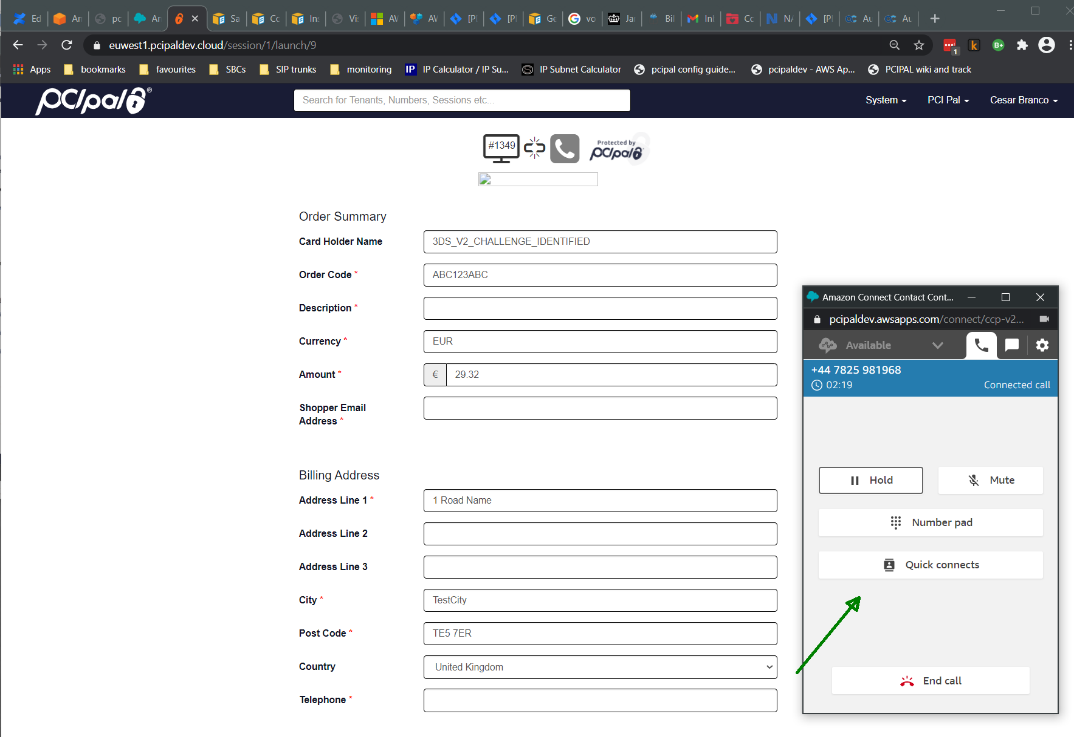
Note the PSTN call, the payment flow and the agent’s soft phone:



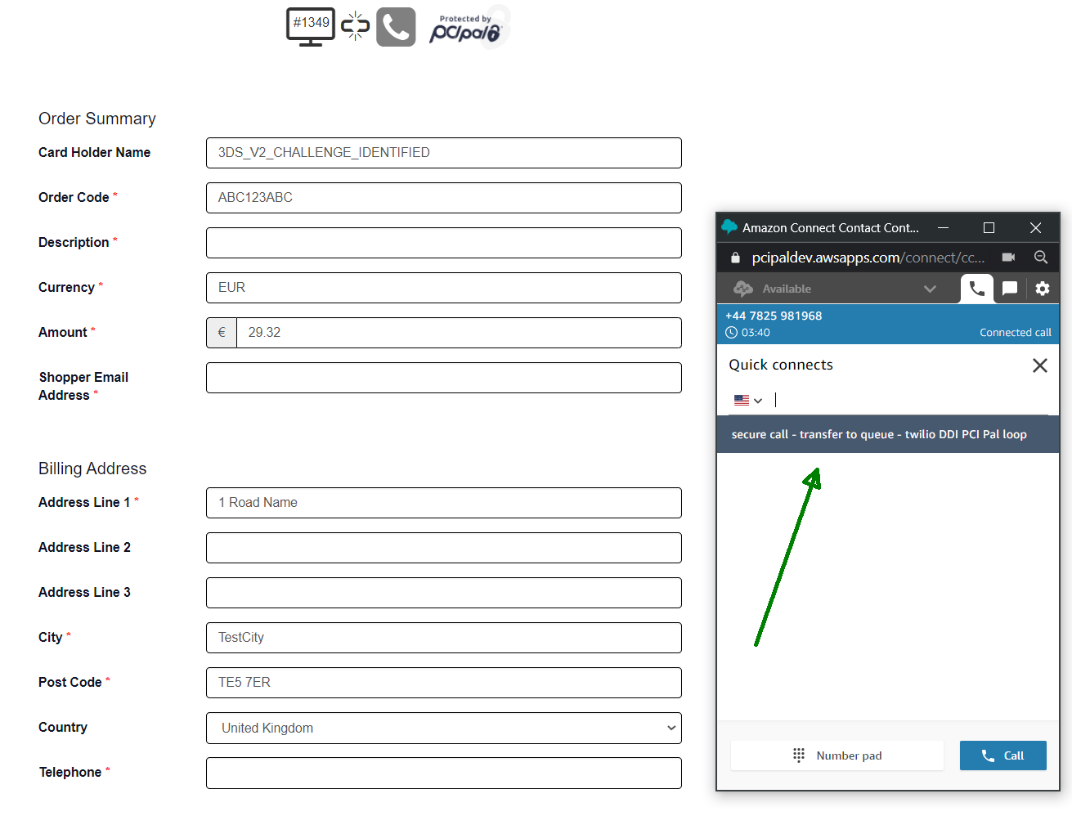
and accept the call:



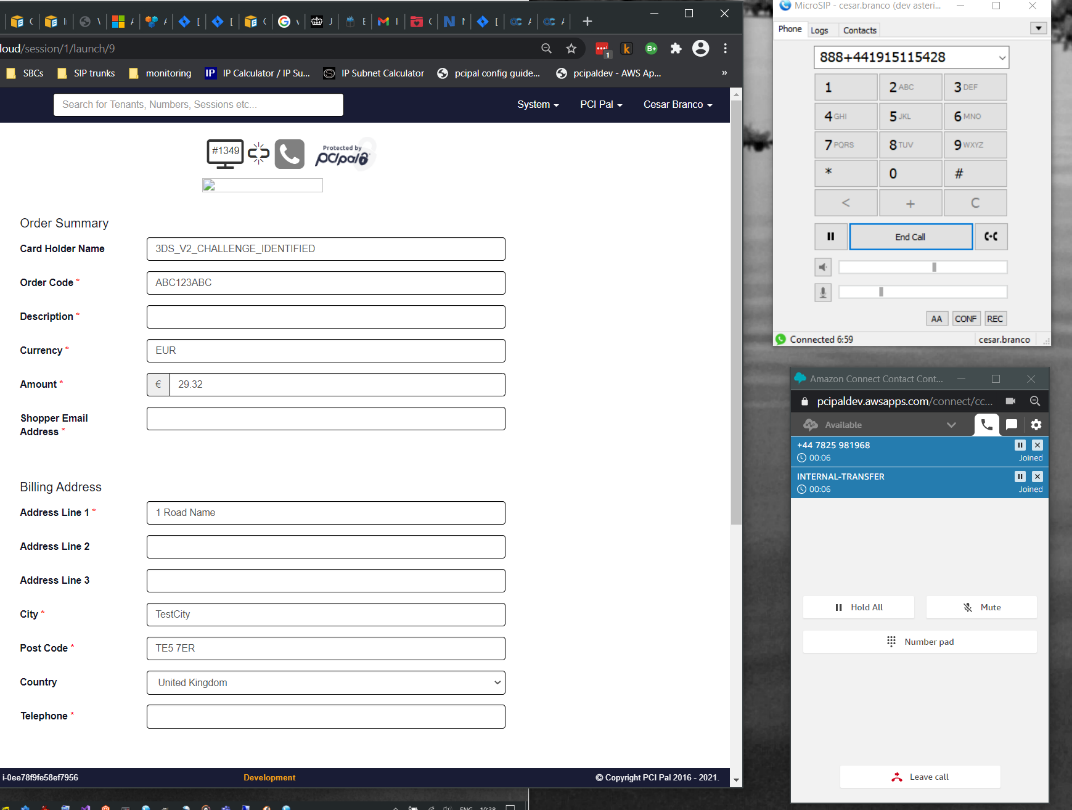
The call is answered. When the agent needs to take a payment click “**Quick connects**”:



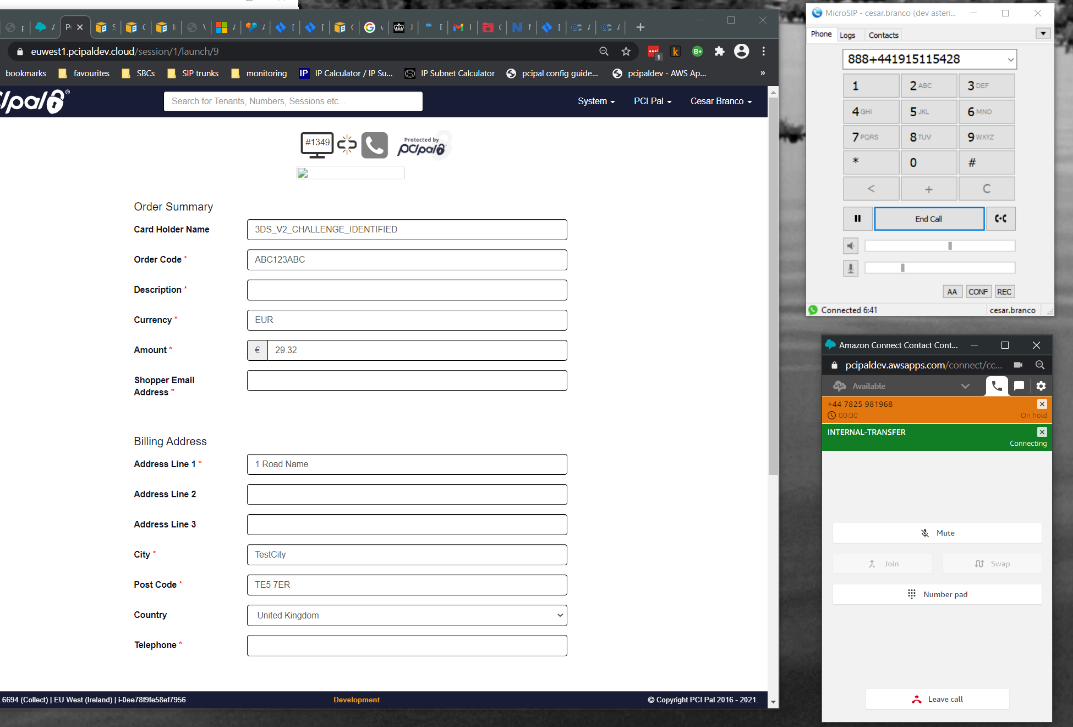
Then double click on the “**secure call - PCI Pal etc**” quick connect created before:



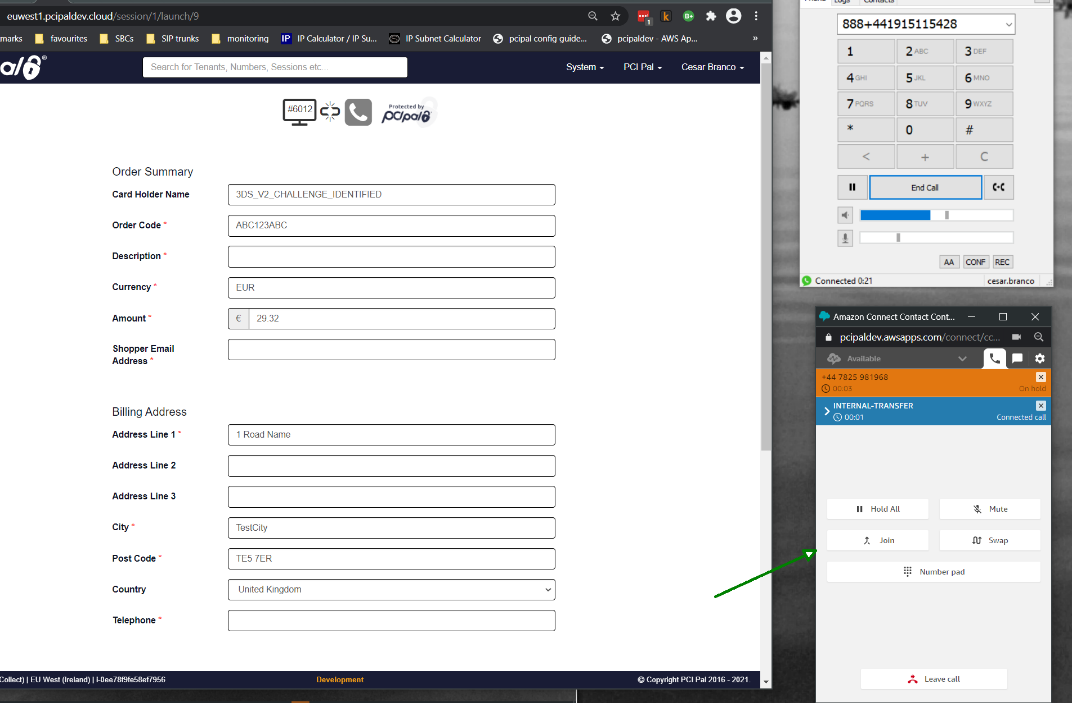
Amazon Connect initiates the 2-step transfer:



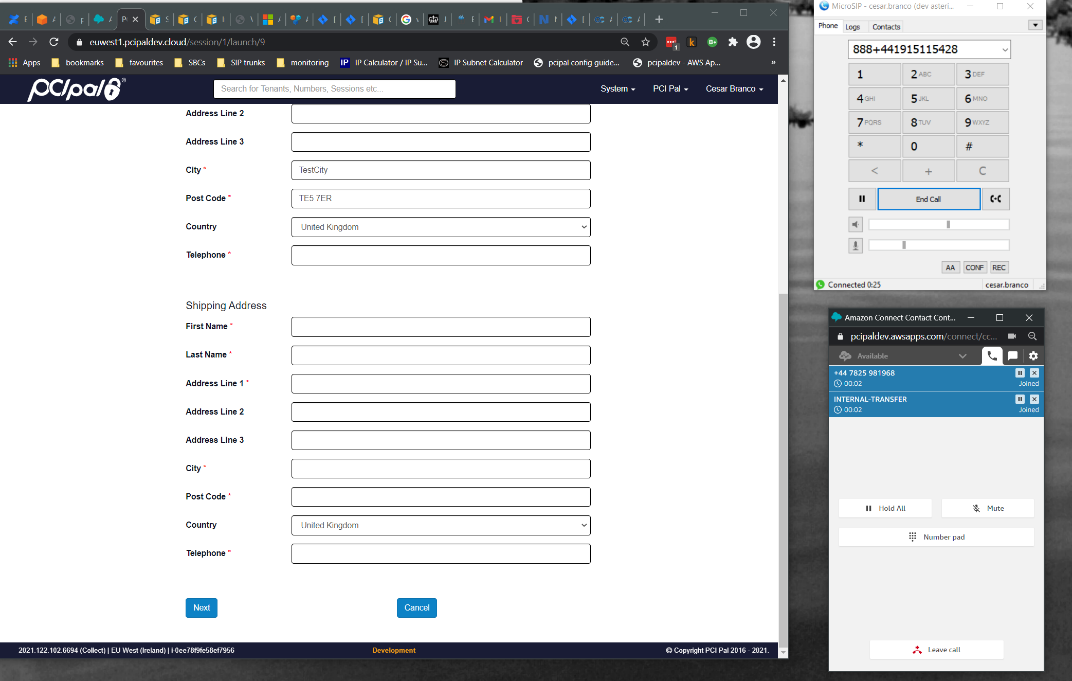
The call reaches PCI Pal and then loops back to Amazon Connect:



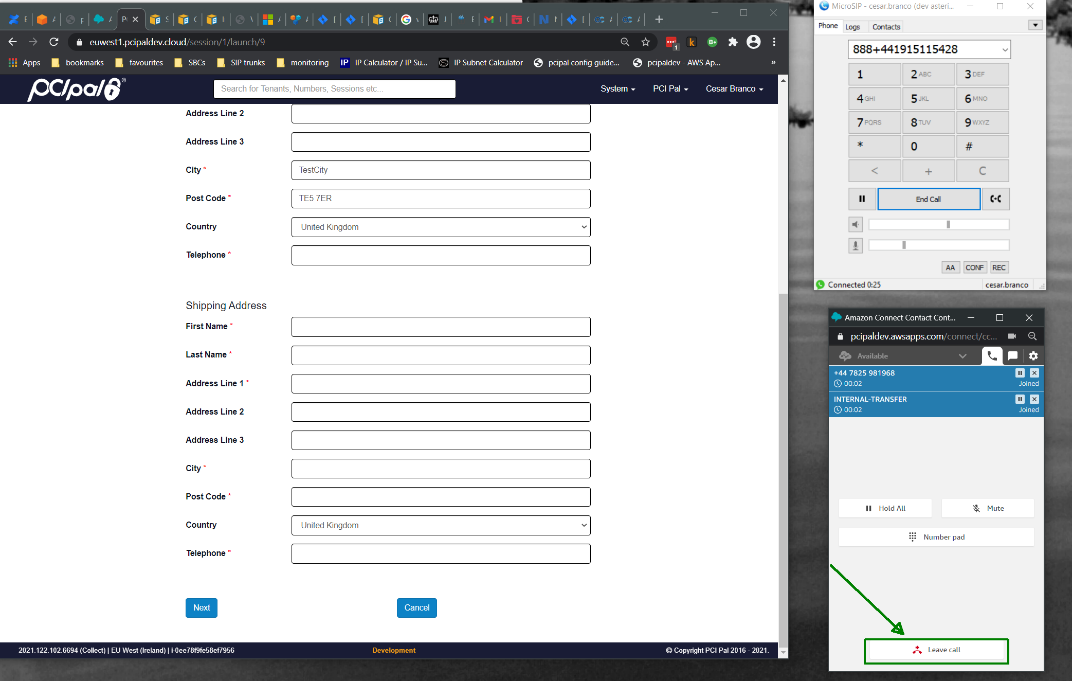
Click “**Join**” so the cardholder is no more than 1 or 2 seconds on music:



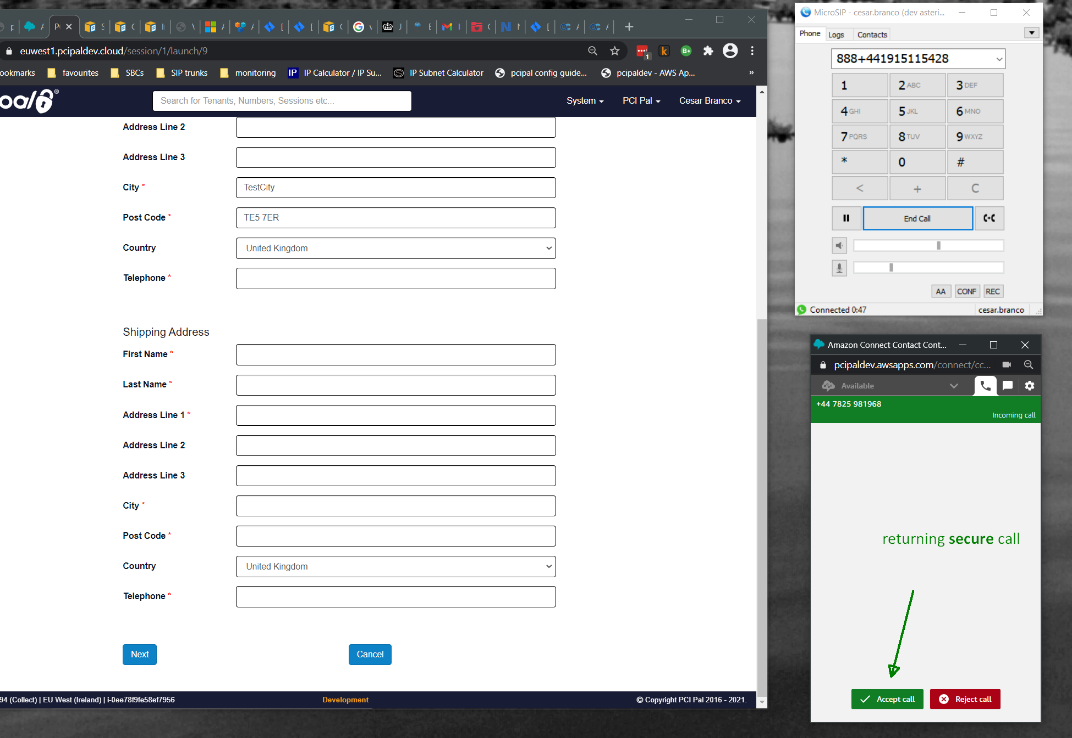
The agent and the cardholder are joined and talking again:



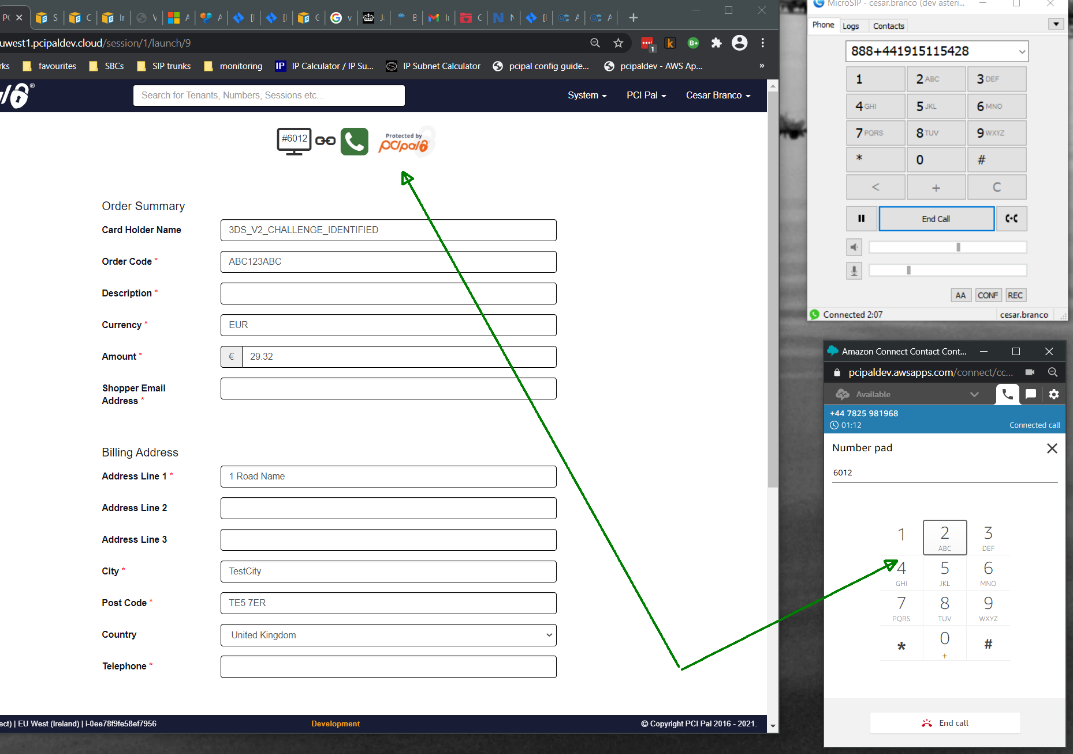
The “**accept call from PCI Pal**” flow will say “**call secure**” and the agent can “**leave call**”:



Which will immediately ring again on the agent personal queue, the agent clicks “**accept call**” and is reunited with the cardholder:



Click “Number pad”, and enter the PIN to link the session:



And take the payment:



The call will remain looped through PCI Pal until either the cardholder or the agent disconnects.

And this is the end of the functional integration.