Euphoria Documentation for Salesforce

What is the Purpose of this document?

This document explains what is needed when creating Webhooks for the Euphoria TMS to interact with a Salesforce account.

For the webhooks to work in the Euphoria Agent Workspace, the user will need to go through 4 set-up phases.

- Salesforce Account Configuration and Workspace Setup
- Ensuring Internet Settings Allow for Redirects
- Creation of the Webhooks in TMS
- Adding the created Webhook to a queue or Campaign.

Salesforce Account Configuration and Workspace Setup

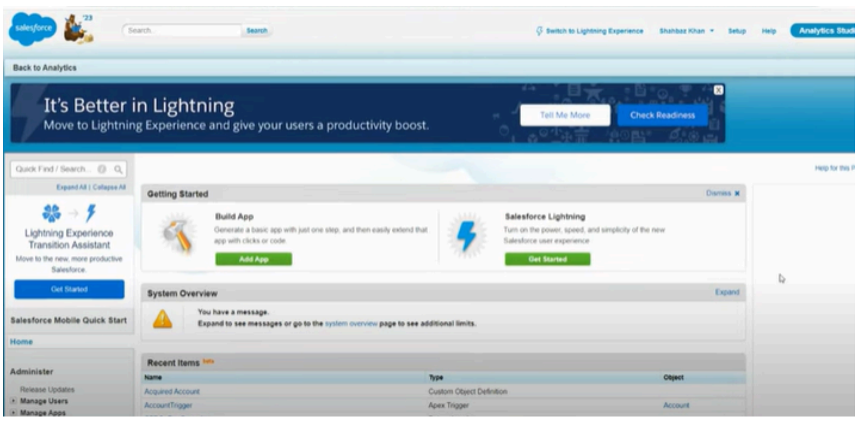
A user can only link the Webhooks from the Euphoria TMS to Salesforce if they have access to Salesforce. Thus, the first step is to create a Salesforce account. If the user already has a Salesforce account, follow the *Create the Organisations Details* steps to ensure that the Euphoria platform is able to access the user's Salesforce account.

Register Salesforce Accounts.

For any assistance with the steps for Salesforce, contact an agent to assist on the [Salesforce Contact Page]

A trial Salesforce account will not allow the TMS webhooks to activate. A paid business account will be needed. Alternatively, a Developer account can be activated. Go to developer.salesforce.com/signup. (When creating a password, do not use any special characters.)

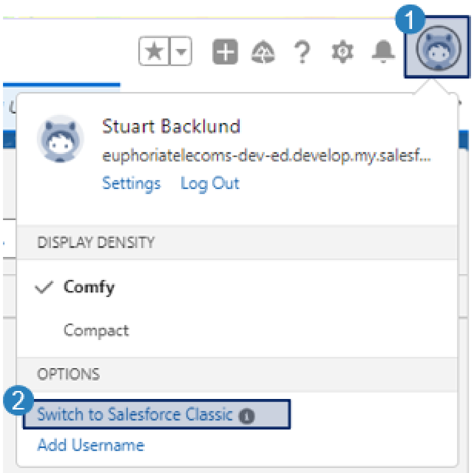
Create a business account using the Salesforce steps. (A Salesforce consultant can assist with this when setting up the account). Below is an example of how the Business presence will look on Salesforce.



Create the Organisations Details

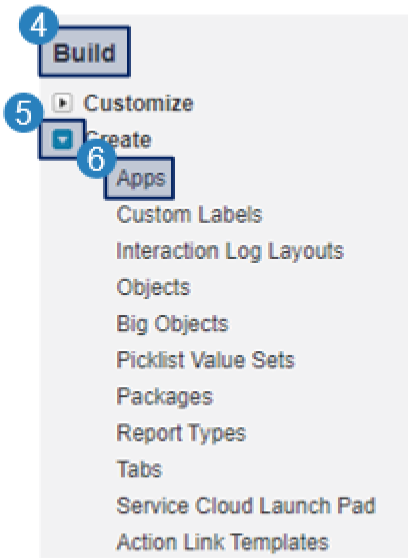
Note: The domain URL “euphoriatelecoms-dev-ed.develop.my.salesforce.com” used in this documentation and pictured below is an example of a customer domain seen in the URL section of a browser. Each user will have their own unique domain, which is used in creating API calls, and will be referred to as the Salesforce Classic Customer URL. (This is the Salesforce Classic domain name, not the Lightening Domain Name)

- If not already on the Classic Salesforce page, select the 1 profile icon on the right, and then 2 *Switch to Salesforce Classic.*

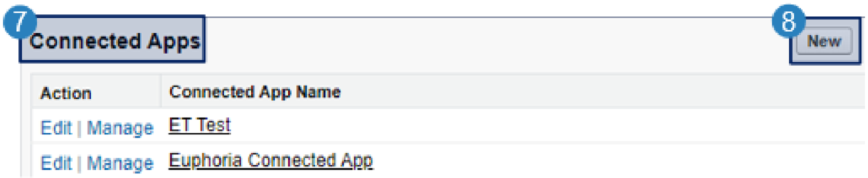


- Select 3 Setup in the top right corner of the screen.

- Scroll down to 4 *Build* in the left menu item pane. Click on the 5 Play button next to create. This will open the create options. Select 6 *Apps* from the side menu. This will open the Apps page.



- From the Apps page, scroll down to 7 Connected Apps. Connected Apps act as the API endpoint for external calls to Salesforce. A user can have multiple connected apps within their Salesforce Organisation. Click 8 *New* to create a new connected application. This will open the *New Connected App* page.

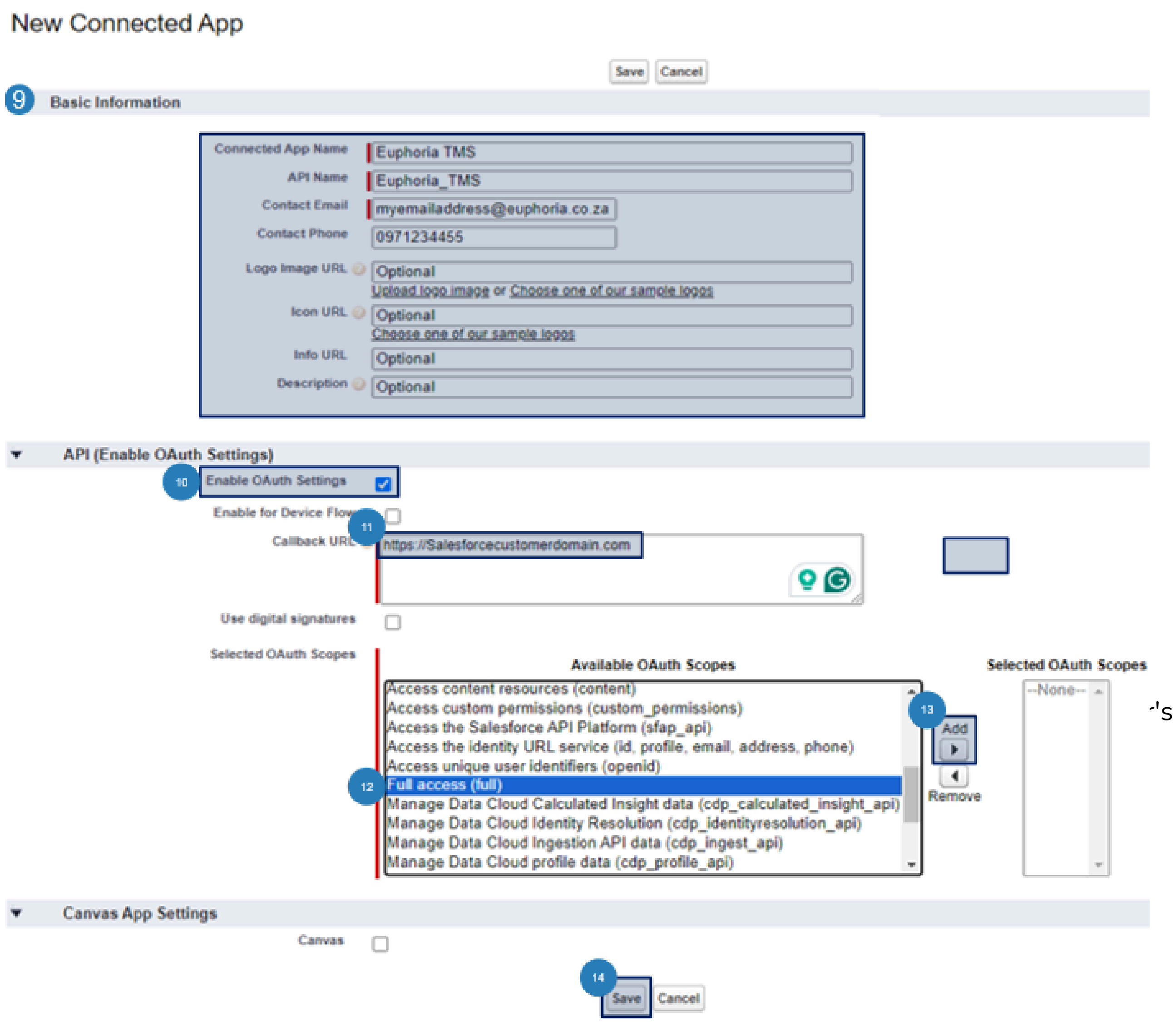


- Complete the 9 form, providing the information as required under the Basic Information header:
    - Connected App Name: The system being integrated with: TMS
    - API Name: Name to display on the Connected Apps section.
    - Contact Email: User's work email address.
    - Contact Phone: Company phone number.
    - Logo, Icon and Info URL: Optional information about the TMS is not required.

- Under the *API (Enable OAuthSetting)* heading, enable the 10 *Enable OAuthorisation Settings* tick box. This will open the section with the available OAuth Scopes.
    - In the *Callback URL* text box, 11 type the Salesforce Classic Customer domain URL. This is where the user will be redirected after authentication from the third party. [(See explanation above)](#)

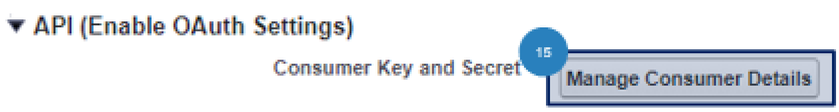
      Example URL: https://Salesforce_customer_domain.com

    - In the *Available OAuth Scope* select box, select 12 *Full Access* (for the time being). Select 13 Add. Scroll to the bottom of the page and click 14 *Save.*

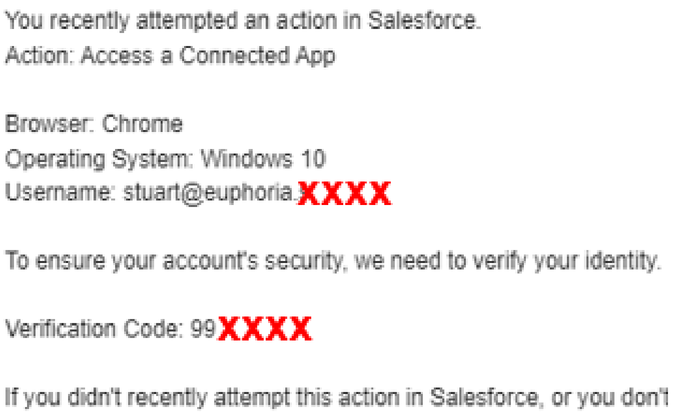


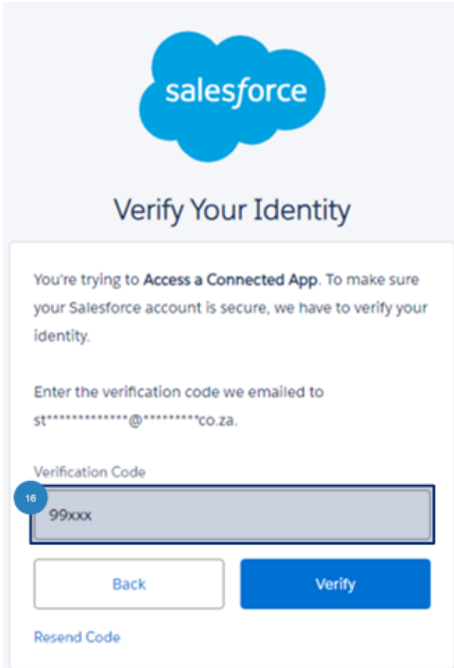
Note: Creating a connected application takes approximately 10 minutes to initialise after it has been saved.

- Press continue, and a new screen will open on the application details. Select 15 *Manage Consumer Details* to view the Authorization token information. This information will ONLY show once; therefore, it is essential to copy and save to a safe space as it will be used during the creation of Webhooks in the third Party.



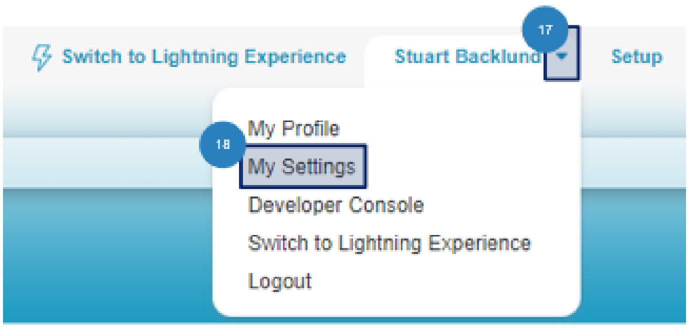
- When Manage Consumer Details is selected, a verification window will appear. Enter the 16 Verification code that has been sent to your email address and press *Verify.*



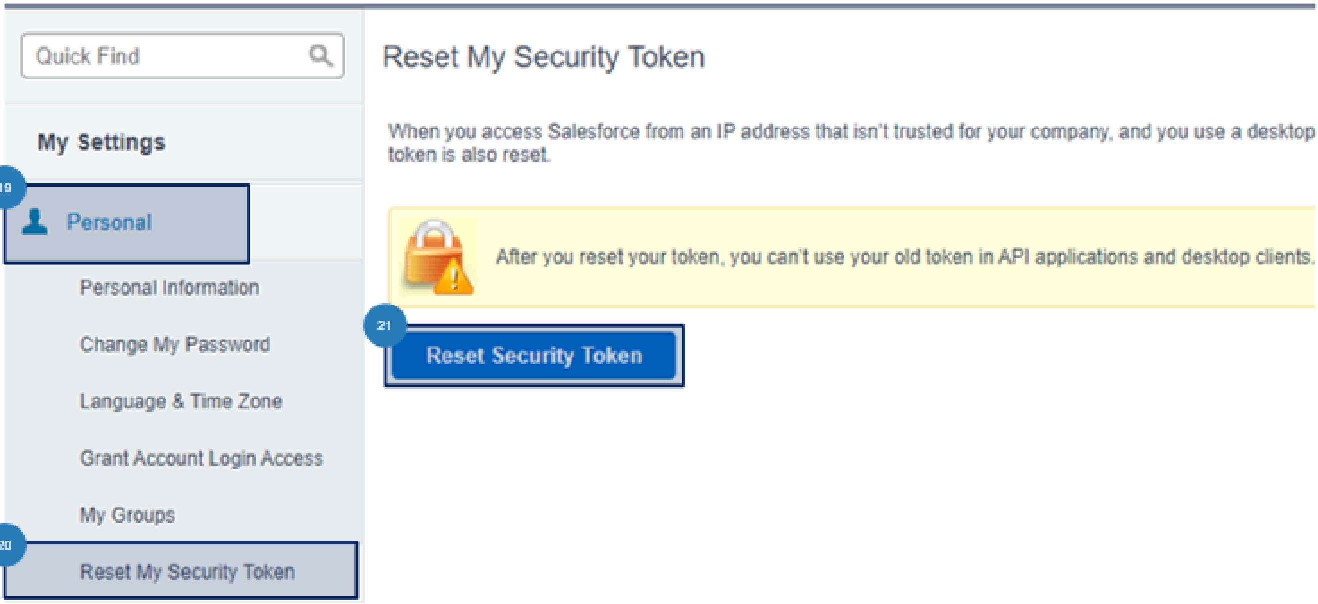


- The page will open and display two values that are needed when creating webhooks in the TMS. Save these in a safe place. These two values are:
  - Consumer Key
  - Consumer Secret

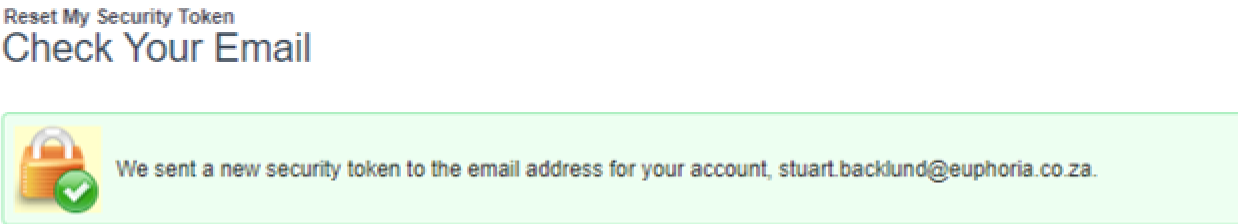
- Once the Key and Secret are saved, a reset of the Authorisation token needs to be done in order to ensure security. To do this, select the 17 user name in the top right corner and then 18 *My Settings*.

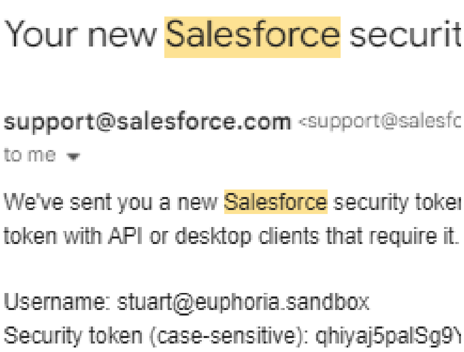


- On the *My Settings* page, Select 19 *Personal*, and then 20 *Reset My Security Token*. The Reset My Security Token window will open. Select the 21 *Reset Security Token* button.



- This will send an email with the new token. Save this email as this token will be needed when creating the Integration Group.



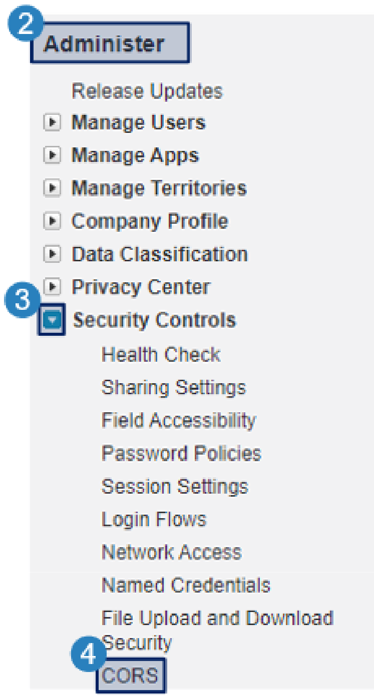


Whitelist Authorisation

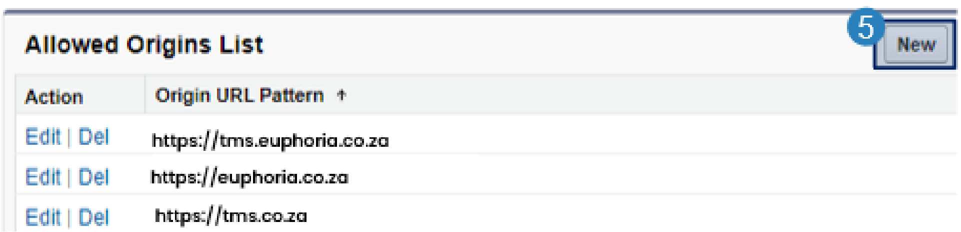
This step is necessary for the secondary level of security.

- Select 1 Setup in the top right corner of the screen.

- Scroll down in 2 *Administer* in the left menu item pane. Click on the 3 *Play* button next to *Security Controls*. This will open the options. Select 4 *CORS* from the side menu. This will open the *CORS* page. This page lists origins that are allowed for cross-origin resource sharing.

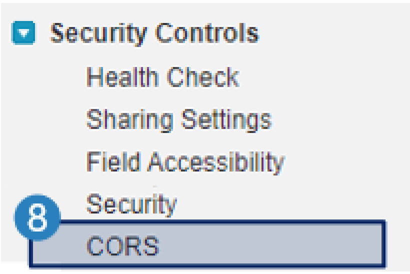


- Click 5 *New* to add to the list of allowed Third Parties. This will open the page to add the Third Party URL.

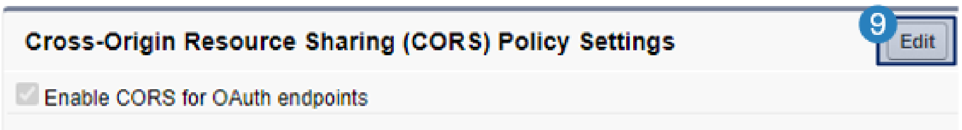


- Add the 6 TMS URL https://tms.euphoria.co.za and press 7 *Save*.

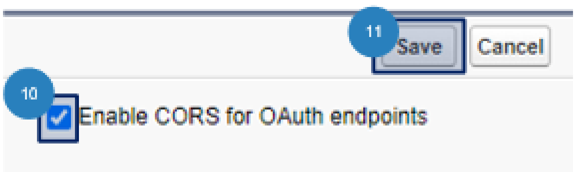
- Select 8 *CORS* from the left side menu again to refresh the page.



- Select 9 *Edit* on the Cross-Origin Resource Sharing (CORS) Policy Settings

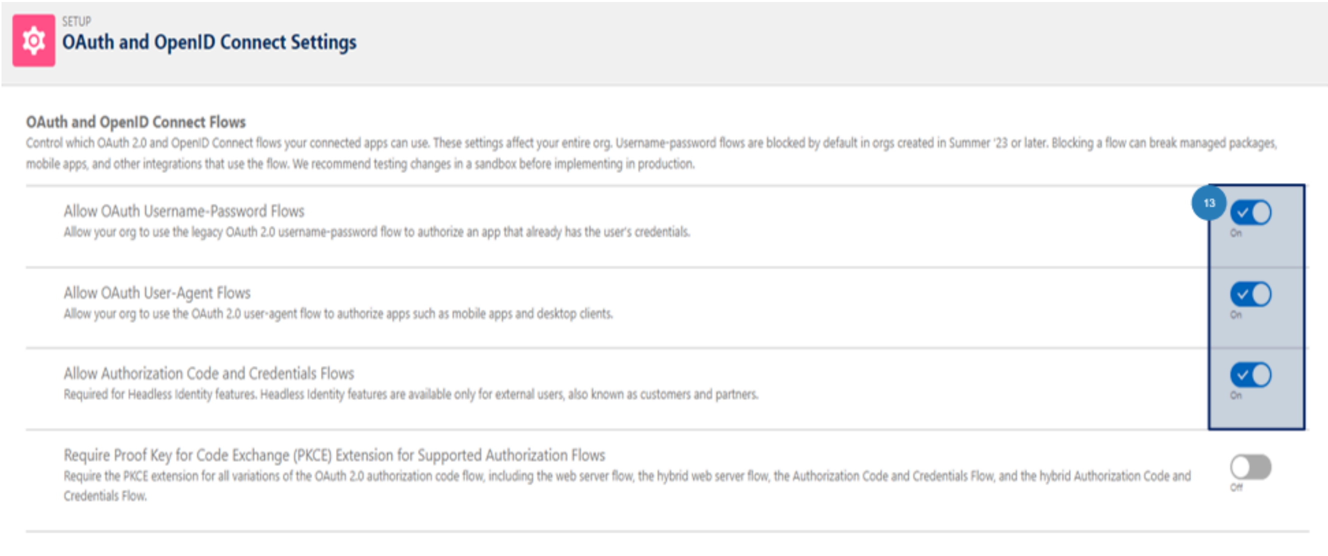


- Enable the CORS for OAuth endpoints 10 checkbox. Select 11 Save to apply the change.

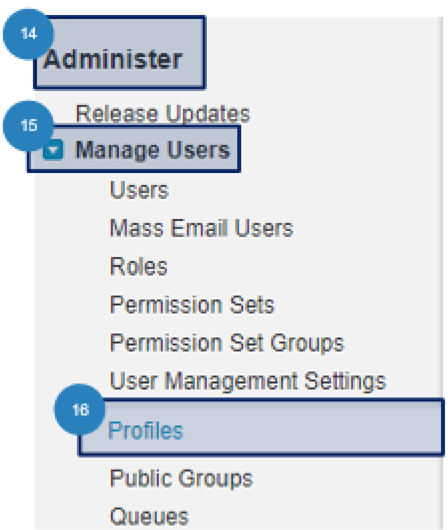


- Select 12 *OAuth and OpenID Connect Settings* from the left side menu.

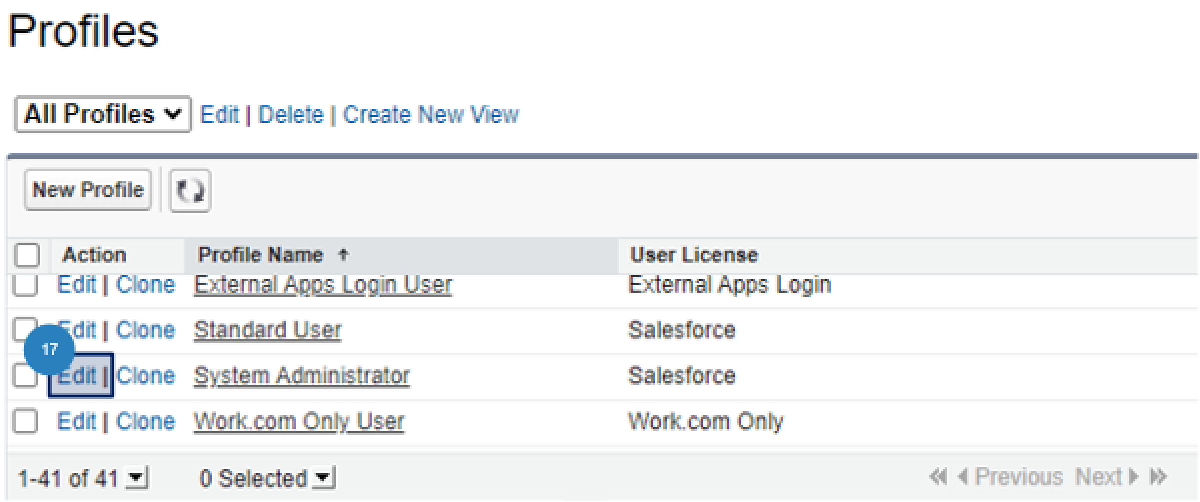
- Ensure Allow *OAuth Username-Password Flows, Allow OAuth User-Agent Flows* and *Allow Authorization Code and Credentials Flows* are all switched on. 13 Refresh the Salesforce page.



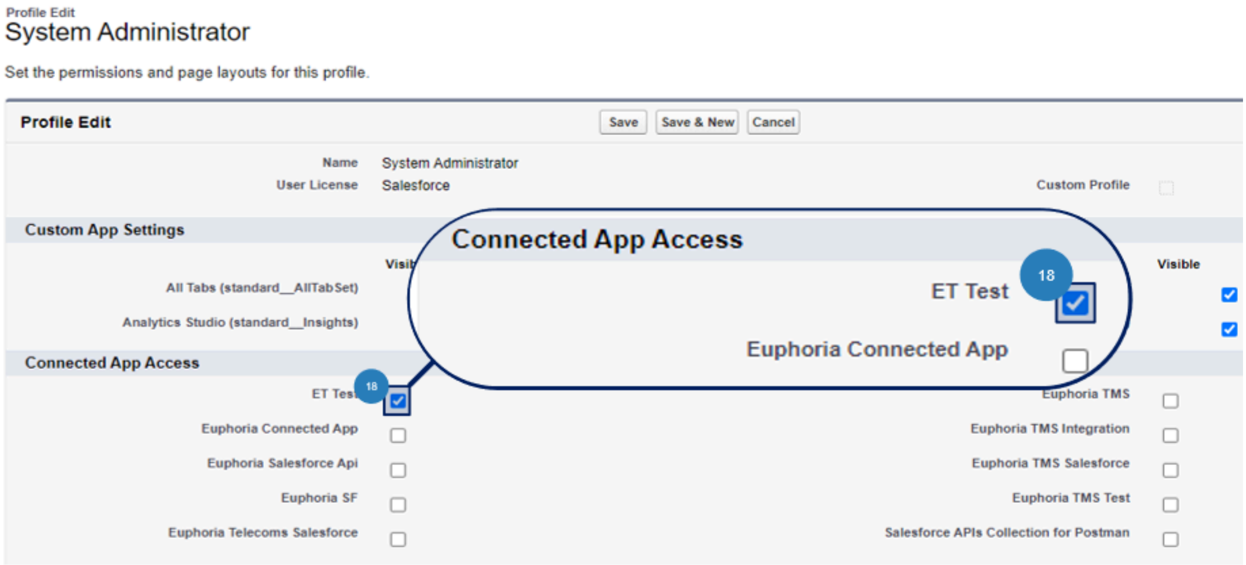
- The last part of the autorisation is to ensure the connected app is allowed access. Scroll down in 14 *Administer* in the left menu item pane. Click on the 15 *Play* button next to *Manage Users*. This will open the options. Select 16 *Profiles* from the side menu. This will open the *Profiles* page. This page lists the different user types in Salesforce.

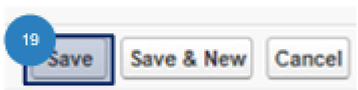


- Scroll down to the System Administrator line, and press 17 Edit.



- On the profiles page enable 18 YOUR connected app to ensure that the webhook has access into Salesforce. Scroll down to the bottom of the page and press 19 Save.





TMS Integration Centre - Webhook Creation

The Euphoria API (Application Programming Interface) is a web service that can interact with other systems to achieve several tasks. Some of these tasks may be actions the system can perform, such as dialling a number or retrieving certain information for use in another system like a CRM application (hereafter referred to as the “target system”).

To achieve any level of integration between systems, both may need the ability to interact via API, so find out what APIs the other system can offer before embarking on an integration project.

Note: System webhooks usage is restricted to extensions with agent functionality enabled.

Integration Centre

The Integration Centre has two primary uses:

1. To send data to another system, this may be information such as numbers, call duration, attending agent or even call outcomes.
2. To query and retrieve information from the other system, often based on information available in the call, such as the number calling in or supplied by an agent, such as a reference/ticket number.

To converse with the other system, the Integration Centre allows the creation of “webhooks”. A webhook is essentially a way to talk to an API on the other system for processing (whatever that may involve), meaning it needs to understand (and be configured for) how the API on the other system expects to interact.

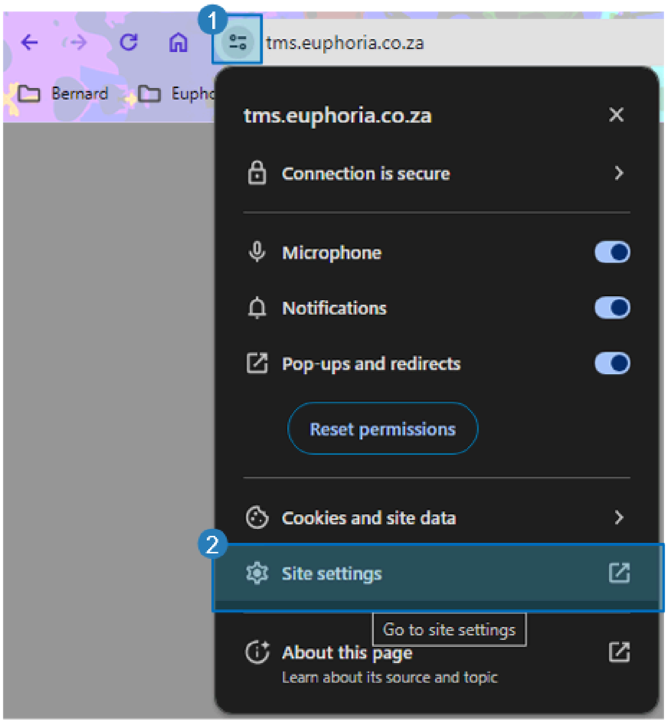
To create webhooks, creating the ground rules for communication in the Integration Group is necessary. Most often, these include parameters like authentication, and the particulars should be available in the other system's documentation.

For any of the below webhooks to work, the agent must be logged in to their Salesforce Account, and the TMS browser settings need to allow Salesforce redirects to open. The next section will indicate how to check the browser settings.

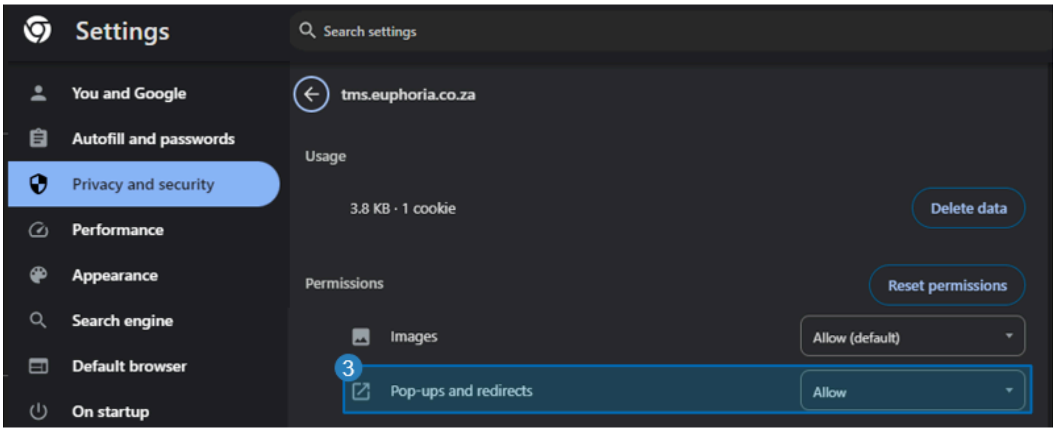
Internet Settings Allow for Redirects.

In order for some of the Webhooks to trigger and open the Salesforce pages, browser settings need to be checked for Google Chrome. This is the recommended Browser.

- On a browser window that has the TMS open, Select the 1 Settings icon or lock icon. Select 2 *Site Settings* to go to the permissions page.



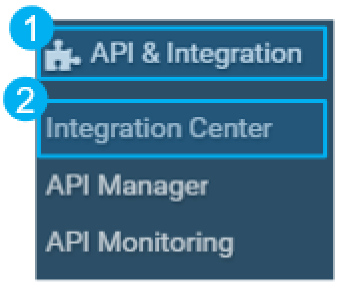
- Ensure that *Pop-Ups and redirects* is switched to 3 *Allow.*



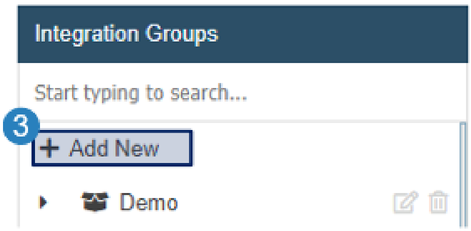
Below are the steps to learn how to create a group. Once a group has been created, webhooks can be created. This document will cover the creation of 8 Webhooks. ***Add This Contact, Add This Contact with Account (Account Lookup), Contact Lookup, Alternative Contact Lookup by Name, Alternative Contact Lookup by Number, Add Activity*** and ***Get Contact Webhooks.***

Integration Center: How to Create an Integration Group.

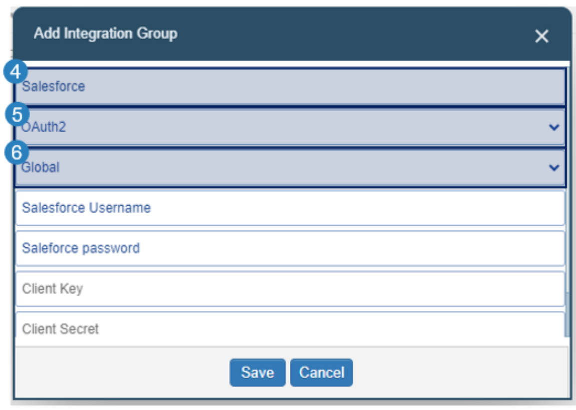
- Go to the TMS URL: https://tms.euphoria.co.za/
- Choose the desired account. A user needs access to the menu Item in order to complete the below steps. This menu item access is either in a Super User account or a manager account.
- Click on the 1 *API & Integration Center* menu item. Click on the 2 *Integration Center* sub-menu item.



- Click on 3 *+Add* New to add an integrations Group.



- Give the group the name 4 *Salesforce* (The name will represent the group under which all webhooks fall). Select the 5 *OAuth2* authentication type. OAuth stands for Open Authorization. It is an open standard for access delegation, commonly used to allow third-party applications to access resources on a user's behalf without exposing their credentials. Select the 6 *Global* authentication level. *Global* is a single set of credentials used for all interactions with the target system.



.

When a Global authentication level is selected, add the Client Salesforce username and Password. This is the same username and Password that is mentioned in the *Create the Organisation Details* section.

a. Username: Salesforce username.

b. Password: A combination of a Salesforce password and New Security Token value which is emailed to you from Salesforce i.e. (Password and New Token value together).

	Password	**Token Value**
Example	: Apple S23xxxx9	ma y twt C2 xxxxxxx REg T7xxxxxn

- Add the 7 Consumer Key And Consumer Secret: As explained in *Create the Organisation Details* section. Scroll down and add YOUR 8 Salesforce Classic Domain URL. Click 9 *Save* to create the group.

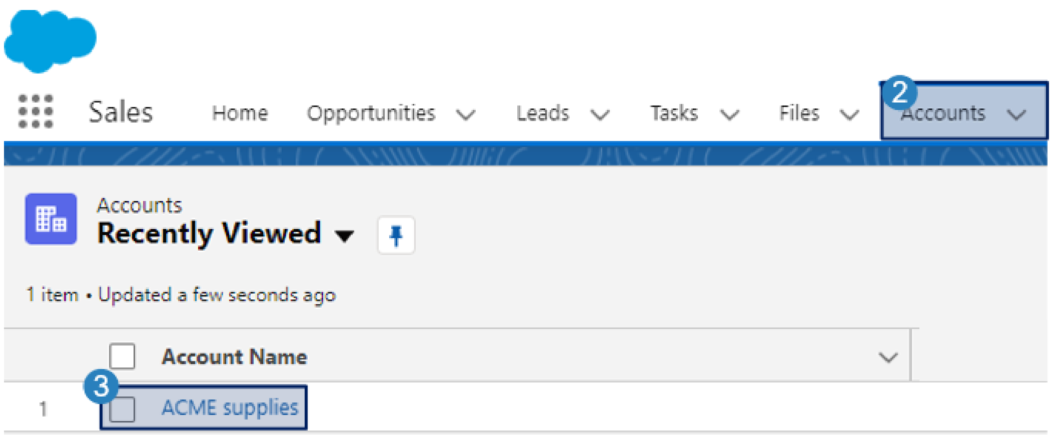
How to Create the Add This Contact Webhook

The Webhook will allow agents to add the current caller to their Salesforce contact list. This webhook will be seen as a button in the Agent workspace and will be selectable during a call. This function is dependent on a company account lookup webhook, as a contact will need to be linked to a company in Salesforce. In order to do this, the company ID is required as one of the webhook parameters. For purposes of illustration, an example company ID will be taken from the Salesforce demo instance to aid in the testing of this webhook's initial setup; this will not be required when this is in production, as further steps are taken below in this document to assist with this. This ID needs to be saved as it will be used in the testing and configuration step. If this is the first time on Salesforce, create a new account.

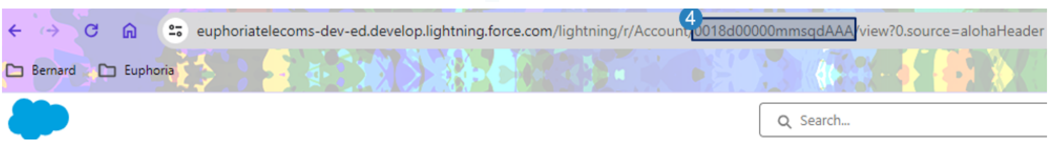
To locate the Company Account ID:

- In Salesforce, select 1 *Switch to Lightning Experience* to access the company and contacts pages. If sent to a page other than the *Home* page, speak to a Salesforce consultant.

- Select the 2 *Accounts* tab and click on an account to open it. 3 (If no account is created, add an account first)

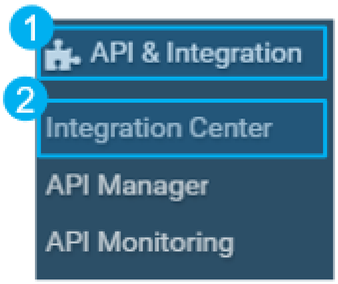


- The Account ID will be in the URL. 4 Copy and save it for when it's needed in the test values.

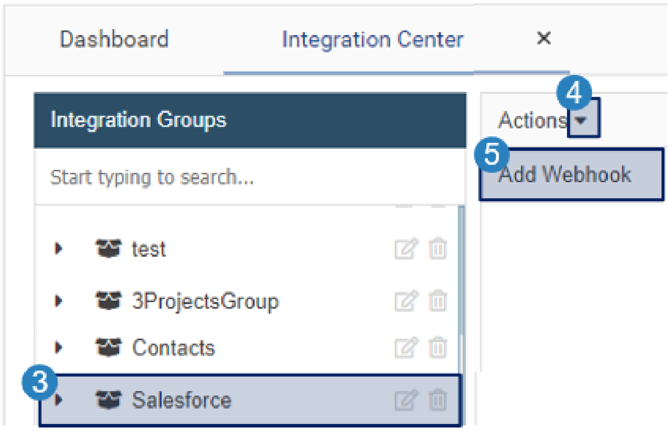


The webhook can be created now that the Account ID is copied and saved.

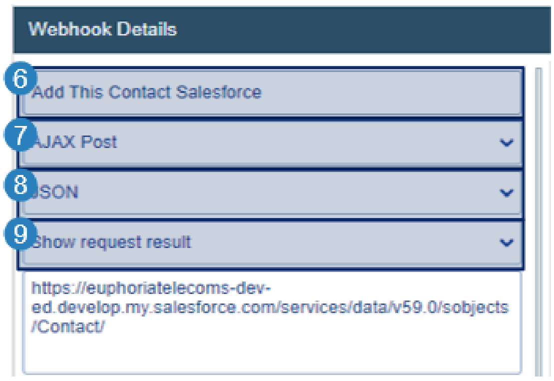
- Click on the 1 *API & Integration Center* menu item. Click on the 2 *Integration Center* sub-menu item.



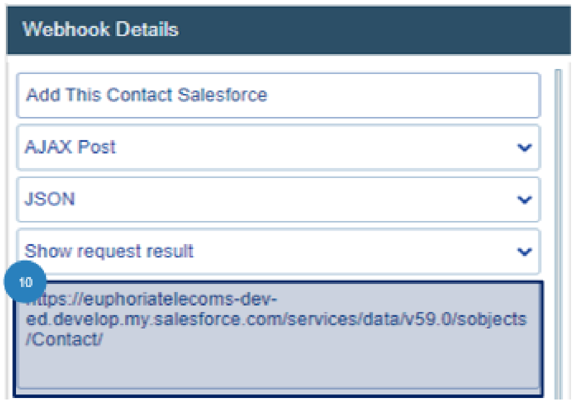
- Select the 3 *Salesforce* Group. Select the drop-down arrow next to 4 *Actions* and then 5 *Add Webhook*.



- Give the webhook a 6 name: Used to identify the webhook, it is helpful to make this descriptive. Select the 7 *AJAX Post* Webhook Type. Select the 8 JSON *Webhook Data Type*. Select the 9 *Show Request Result* Response Type, which indicates how the result should be handled.



- Add the Salesforce Classic Customer Domain URL that the webhook needs to access. The first part of the URL is the company URL; the second part is the page that needs to be accessed.

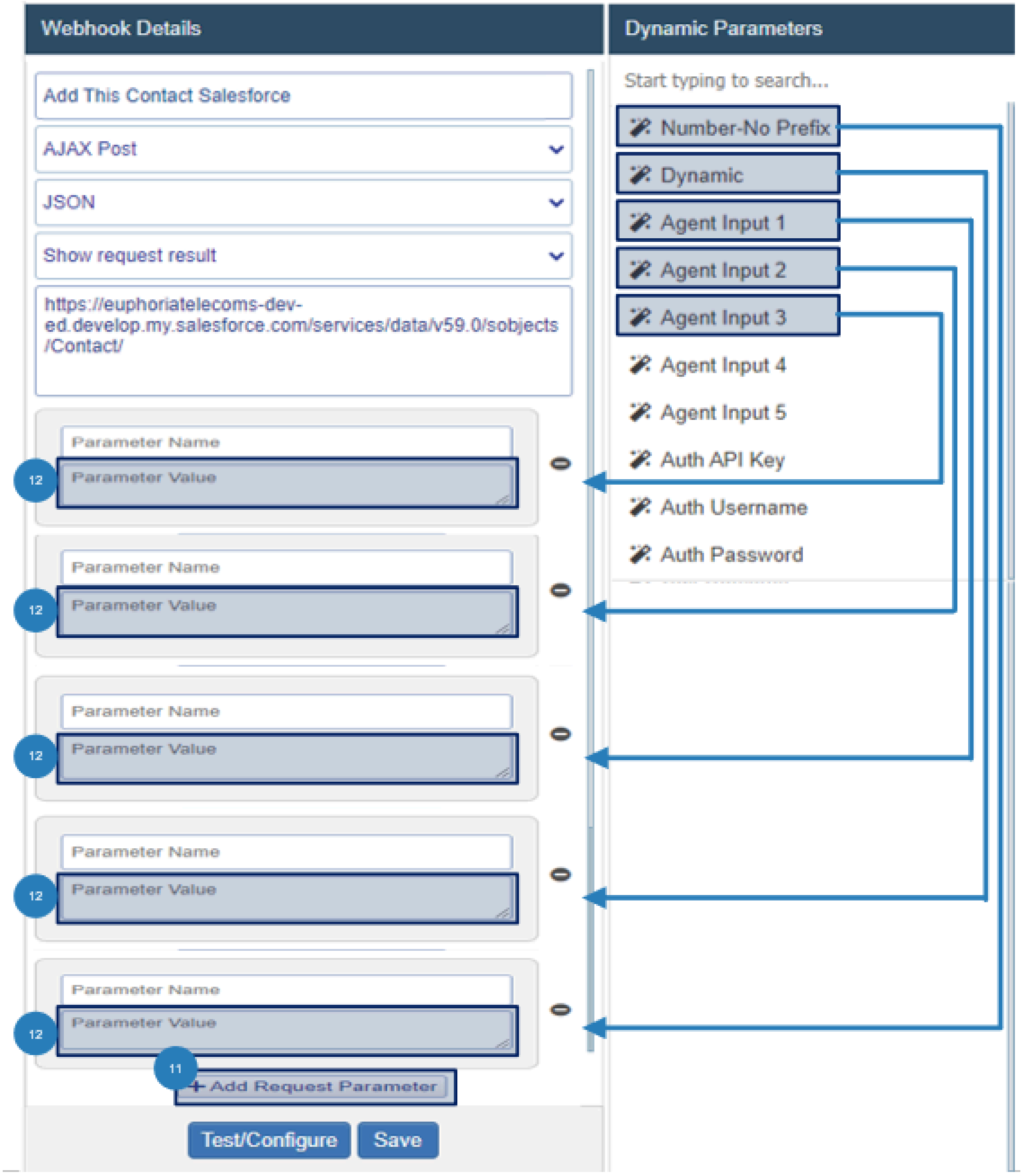


Add parameters:

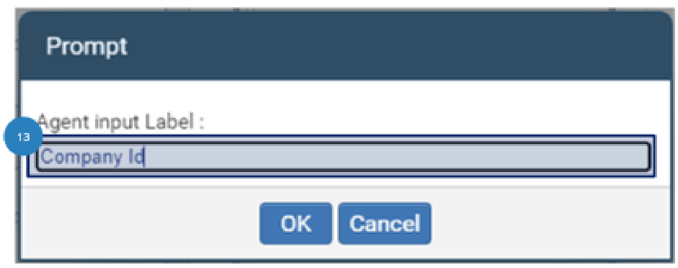
Dynamic Parameters are added when the request parameter button is selected. These variables are found in the target system's API documentation, and the capitalisation as per the document is important.

https://developer.salesforce.com/docs/atlas.en-us.object_reference.meta/object_reference/sforce_api_objects_contact.htm

- Select the 11 *+ Add Request Parameter* button. Then drag and drop a 12 *Dynamic Parameter* into a value box. This will need to be done five separate times to have all the required *Dynamic Parameters* for the webhook. After each drag and drop of an *Agent Input* parameter, a Prompt pop-up will open. These labels are what the agent will see when the value is requested.



- An *Agent Input* type allows for an agent to type in a value when this webhook runs, like in this case to specify the contact details. This will open the Agent Input label prompt. These labels are what the agent will see when the value is requested. The labels must say Name, Surname, Phone Number, Email, and Company Account ID to find the correct Company in Salesforce. 13



- Add the parameter name as explained in the Salesforce API document as seen below.

  For a new contact, the user needs to add parameters for the First Name, Last Name, Phone Number, Email and Company ID. The company ID is classified as an Account Id in Salesforce and was saved in the first steps.

14

- Name: FirstName
- Parameter Value: {AGENT_INPUT_1}

15

- Name: LastName
- Parameter Value: {AGENT_INPUT_2}

16

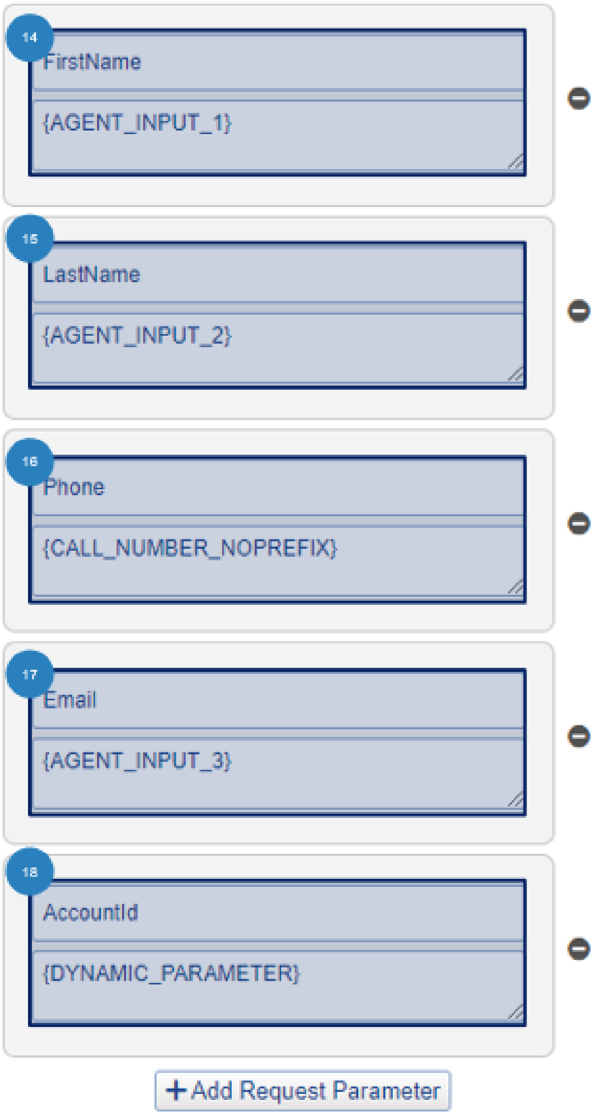
- Name: Phone
- Parameter Value: {CALL_NUMBER_NOPREFIX}

17

- Name: Email
- Parameter Value: {AGENT_INPUT_3}

18

- Name: AccountId
- Parameter Value:{DYNAMIC_PARAMETER}

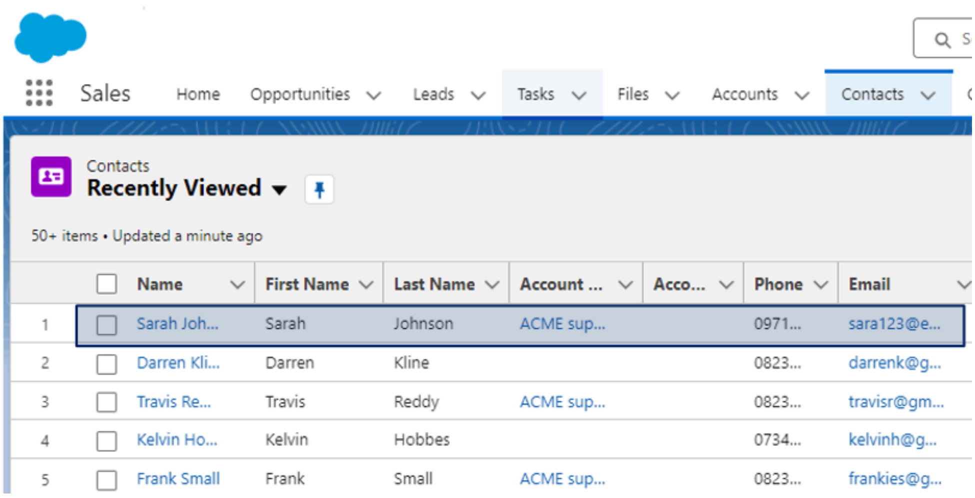


- Click 19 *Save* to create the webhook. The webhook will close. Open the webhook again and select the 20 *Test and Configure* button.

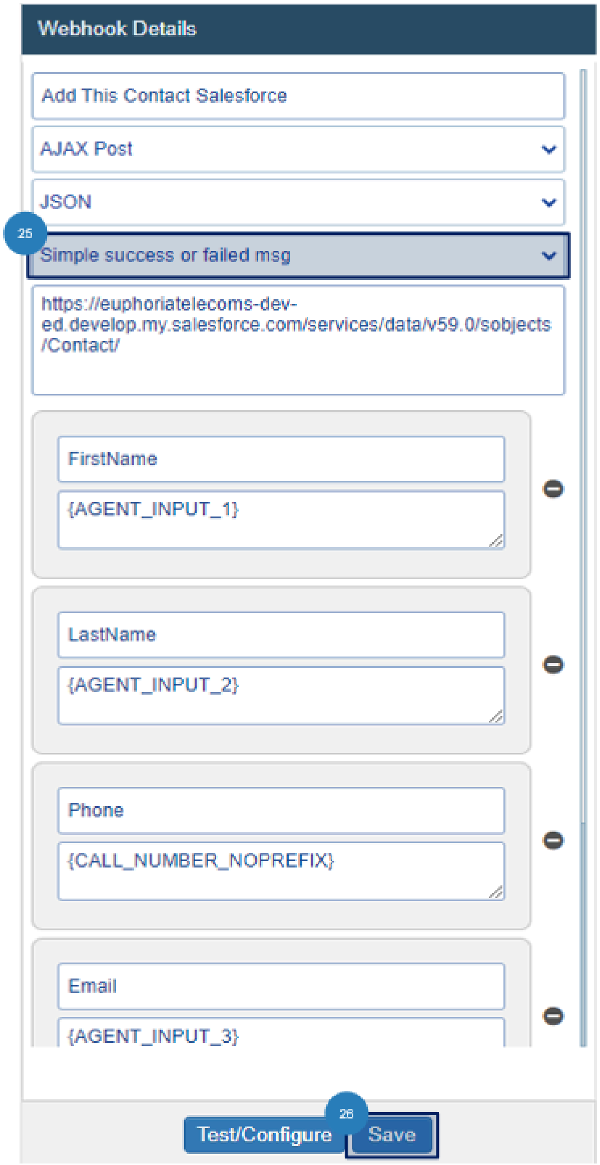
- A pop-up window opens requesting a test value. Add the 21 *Test Values* requested and Press 22 *Apply*. For Account ID, use the saved details from Salesforce as explained above. These test values are only requested in this step to ensure that a result is given. The Agent will not need to add test values in the normal usage of the webhook.

- A *Result* page will open with the information requested by the webhook. Select the result options the agent should see. Select 23 *Agent View* and press 24 *Apply* to see the results the same way the agent would.

The result of adding a contact can be viewed in the contacts list view on the Salesforce contacts page, as well as on a results page in the TMS.



- Once the result shows correctly, change the response type in the webhook to 25 *Simple success or failed message. This will allow a pop-up message to show when the contact has been added successfully or a failed message when it has not.* Click 26 *Save* again to apply the change.

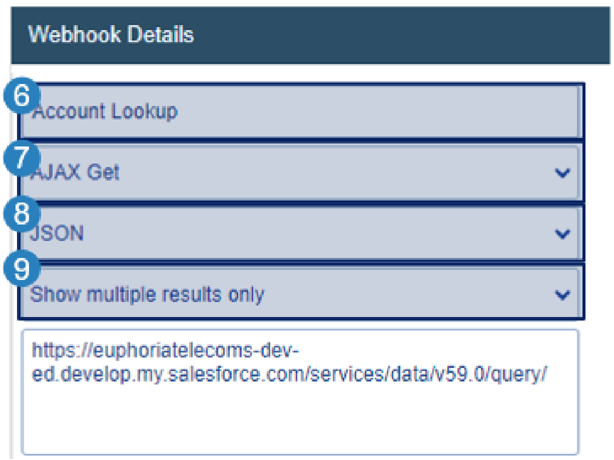


How to Create an Account Lookup Webhook

The Account Lookup Webhook will be used to access Company information from Salesforce. This is necessary when adding a contact to Salesforce. The webhook will be linked to the *Add this Contact* webhooks. Once the Company account has been selected, the *Add This Contact* webhook will activate.

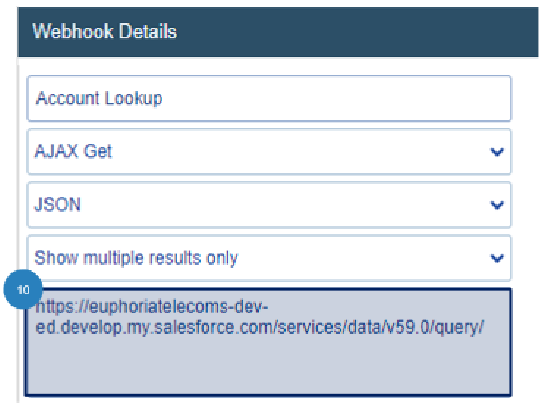
To create the webhook, follow steps 1 - 5 as per the How to Create the Add This Contact Webhook.

- Give the webhook a 6 name: Used to identify the webhook, it is helpful to make this descriptive. Select the 7 *AJAX Get Webhook* Type. Select the 8 *JSON Webhook Data Type*. Select the 9 **Show Multiple Requests only* Type, which indicates how the result should be handled.



****Note: When testing the webhook, select Show Request Result. This will open a result page, even when there is only one result, allowing the selection of viewable fields which will be covered in later steps.***

- Add the Salesforce Classic Customer Domain URL that the webhook needs to access. The first part of the URL is the company URL, and the second part is the page that needs to be accessed.

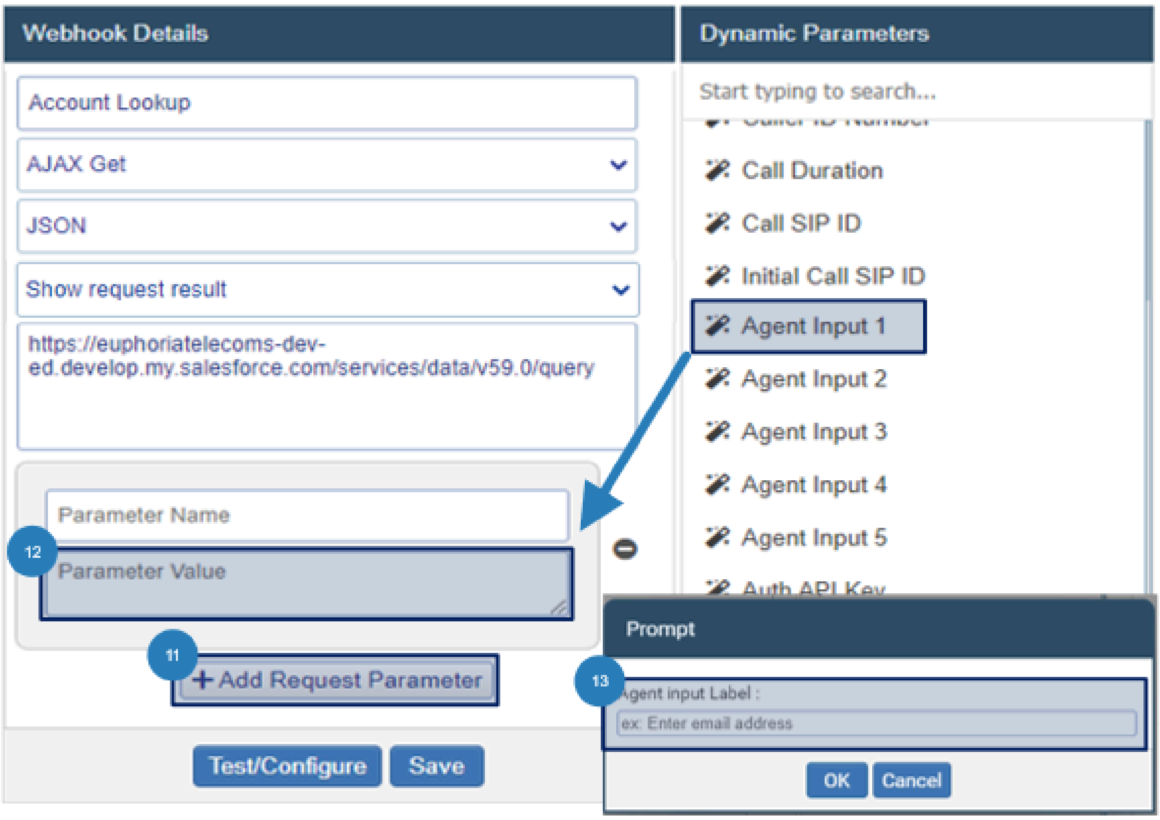


Add parameters:

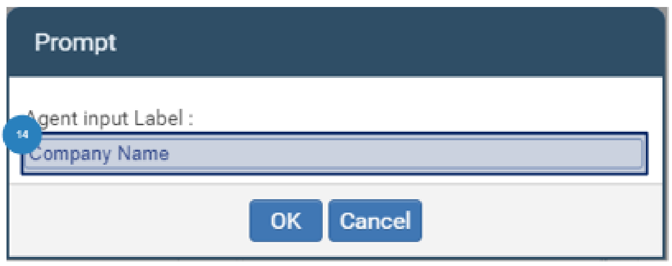
Dynamic Parameters are added when the request parameter button is selected. These variables are found in the target system's API documentation, and the capitalisation as per the document is important.

https://developer.salesforce.com/docs/atlas.en-us.object_reference.meta/object_reference/sforce_api_objects_accountshare.htm

- Select the 11 *+ Add Request Parameter* button. Drag and drop the correct parameter into the 12 parameter value box. This will open the 13 Agent Input label prompt.



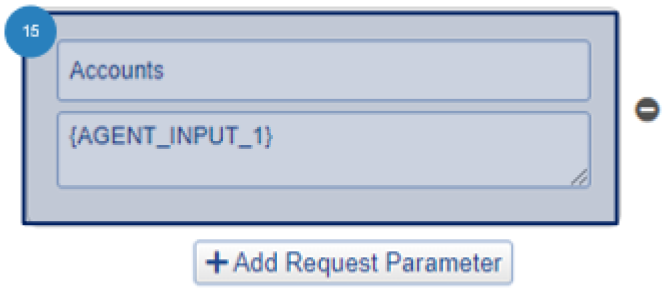
- This label is what the agent will see when the value is requested. The label must say 14 *Company Name* to find the correct Company in Salesforce. (Account is Salesforce are considered companies, therefore it will be one of the account names)



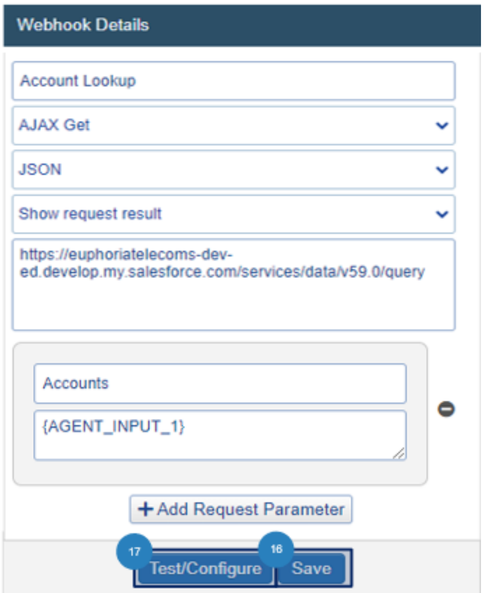
- Add the parameter name as explained in the Salesforce API document as seen below.

  15

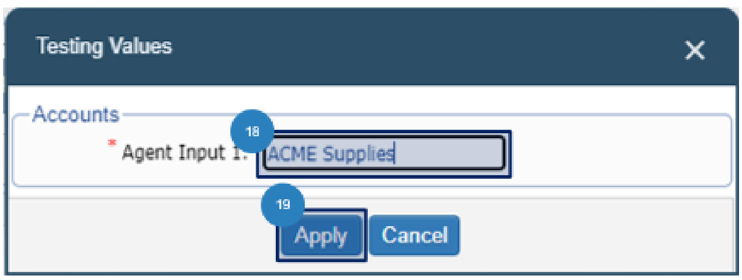
  - Name: Accounts
  - Parameter Value: {AGENT_INPUT_1}



- Click 16 *Save* to create the webhook. The webhook will close. Open the webhook again and select the 17 *Test and Configure* button.

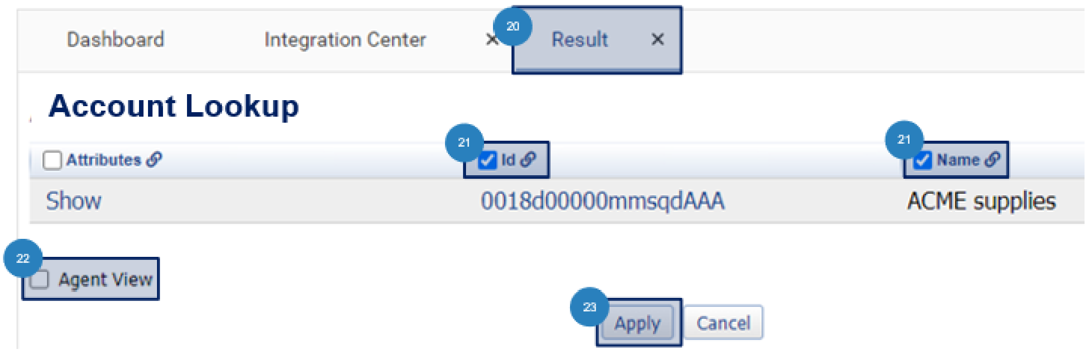


- A pop-up window opens requesting a test value. Add an 18 Account Name of a company that can be found in Salesforce and press 19 *Apply*. This test value is only requested in this step to ensure that a result is given. The Agent will not need to add test values during regular webhook usage.

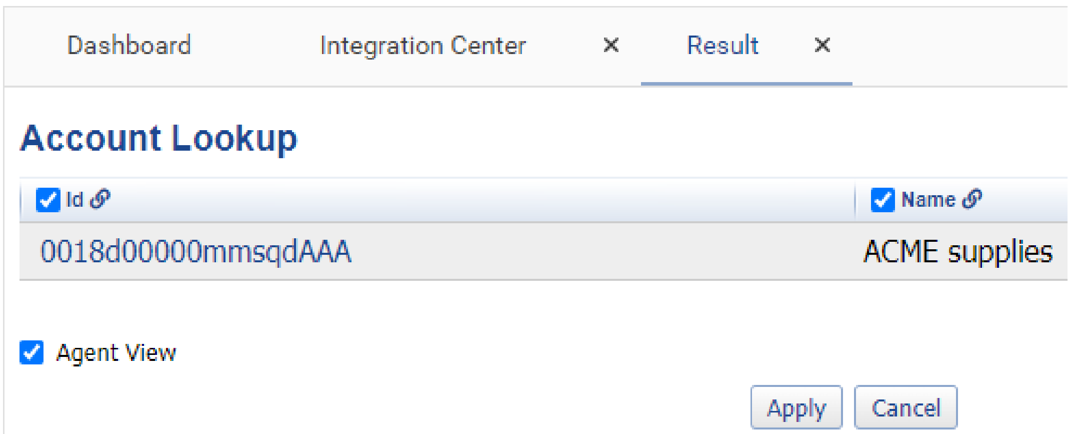


- A 20 *Result* page will open with the information requested by the webhook. For some webhooks, the result page will have many columns of information, but not all of the information will be needed by an agent. Therefore it is important to limit what the agent sees. To do this select the 21 tick box next to the column of information the agent should see. Click on 22 *Agent View* and press 23 Apply to see the results the same way the agent would.

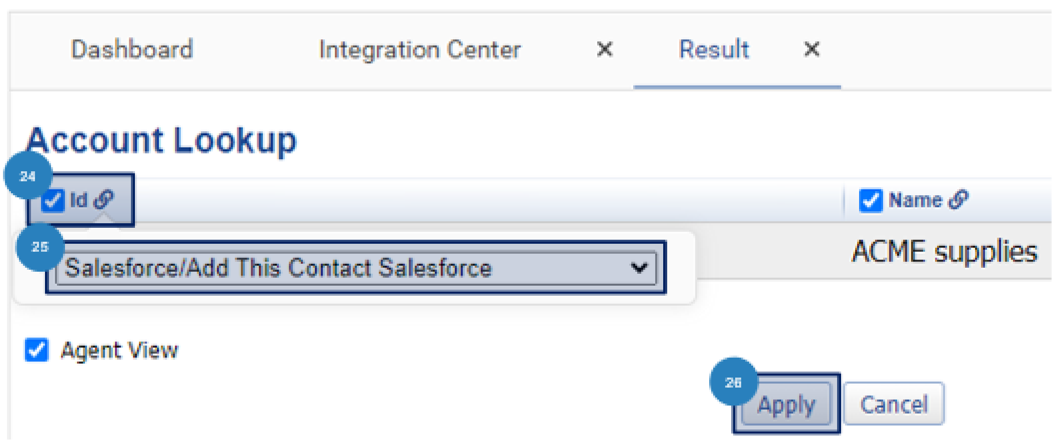
Unfiltered



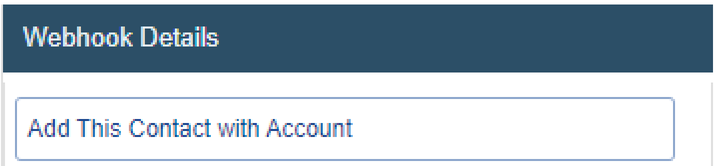
Filtered



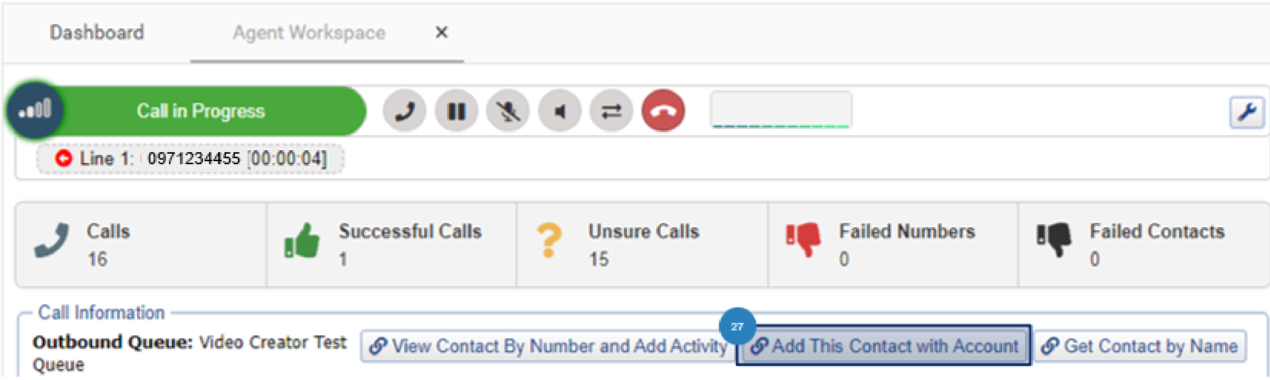
- Select the 24 *Id* link chain icon to link another webhook to the *Account Lookup* Webhook. Select the 25 *Add This Contact* Webhook. Press 26 *Apply*. This will allow the agent to add the person on the call as a contact in Salesforce under a selected company. (Go back to the integration Centre page to the created webhook)



- Change the name of the webhook from Account Lookup to *Add This Contact with Account*, as that is what the agent will see in the agent workspace. Select Save.



Sample of agent workspace.



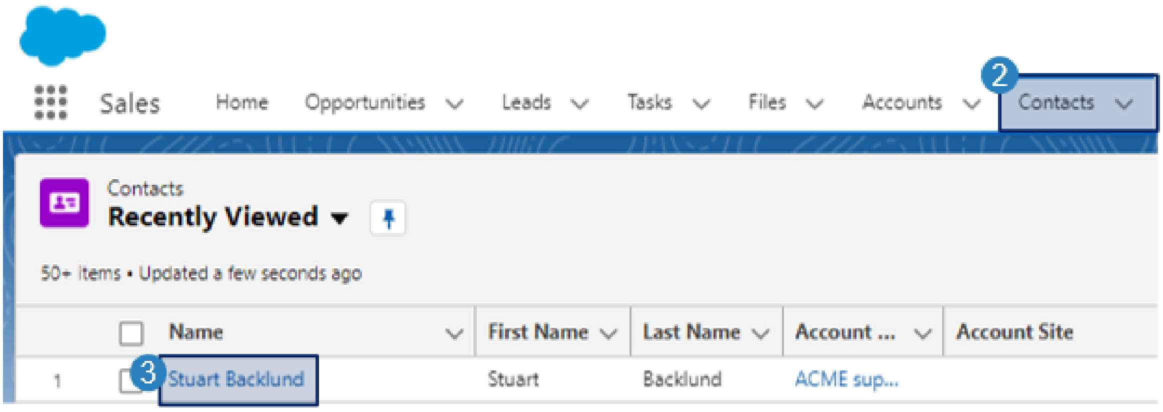
How to Create the *Open Salesforce* Webhook

The Open Salesforce Webhook will be used to open a contact's profile in Salesforce with either their number or name. It is created to link with the *Contact Lookup*, *Alternative Contact Lookup by Name* and *Alternative Contact Lookup by Number* webhooks to ensure that the correct Salesforce page opens. This webhook will not be added to a queue or seen in the Agent Workspace.

In order to create and test this webhook, the user will need the Contact ID (not to be confused with the Company Account ID). This ID will be used during the Test and Configure stage of the webhook creation but will not be needed during the normal use of the webhooks in the Agent Workspace. To locate the Contact ID, follow the below steps:

- In Salesforce select 1 *Switch to Lightning Experience* if not already in this space.

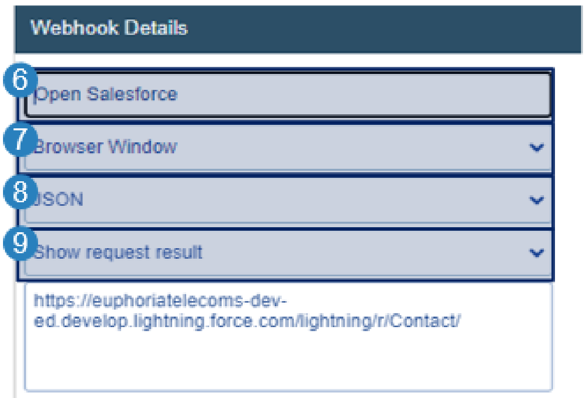
- Select the 2 *Contacts* tab, and select any 3 contact in the list to open it.



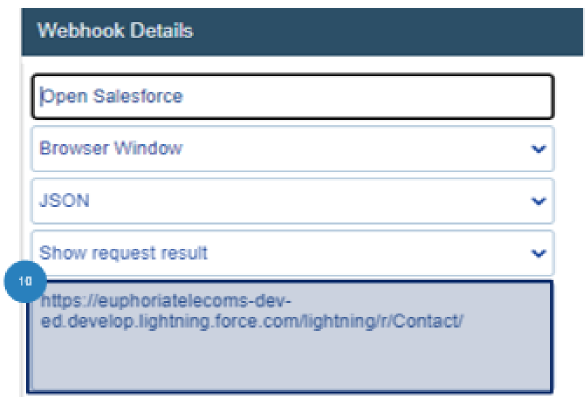
- The Contact ID will be in the URL. 4 Copy and save it for when it's needed in the test values.

To create the webhook, follow steps 1 - 5 as per the How to Create the Add This Contact Webhook.

- Give the webhook a 6 name: Used to identify the webhook, it is helpful to make this descriptive. Select the 7 *Browser Window Webhook* Type. Select the 8 JSON *Webhook Data Type.* Select the 9 *Show Request Result* Response Type, which indicates how the result should be handled.



- Add the Salesforce Lightening Customer Domain URL that the webhook needs to access. The first part of the URL is the company URL, the second part is the page that needs to be accessed in lightening.

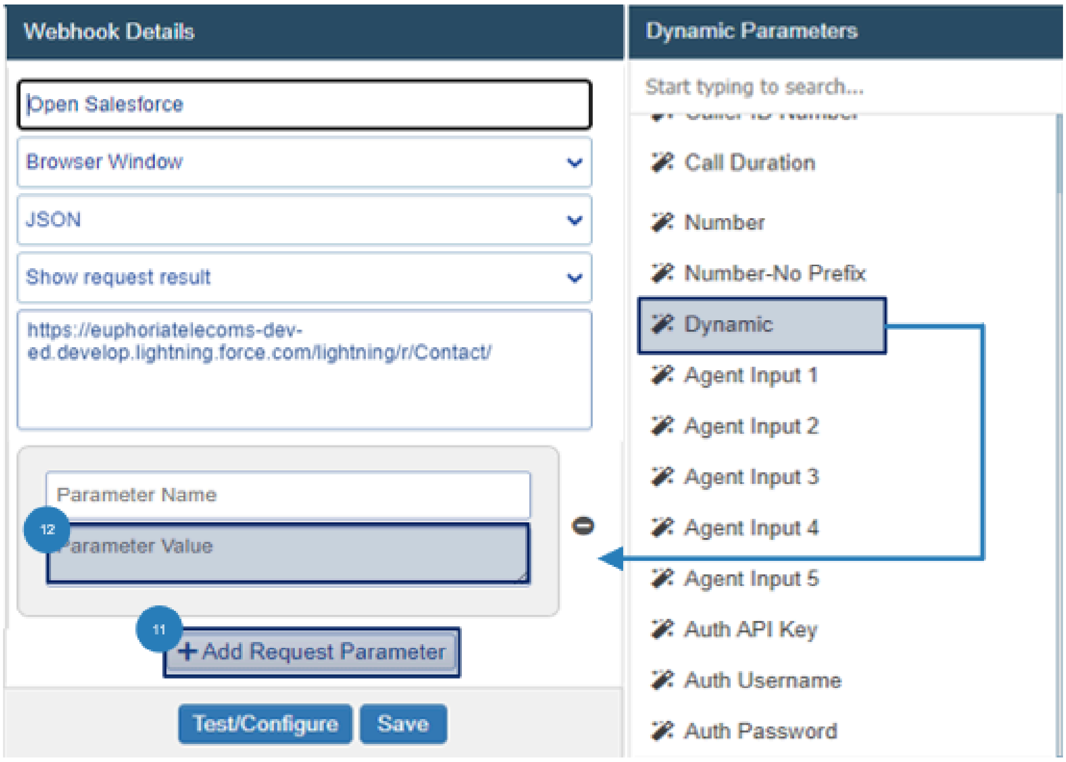


Add parameters:

Dynamic Parameters are added when the request parameter button is selected. These variables are found in the target system's API documentation, and the capitalisation as per the document is important.

https://developer.salesforce.com/docs/atlas.en-us.object_reference.meta/object_reference/sforce_api_objects_contact.htm

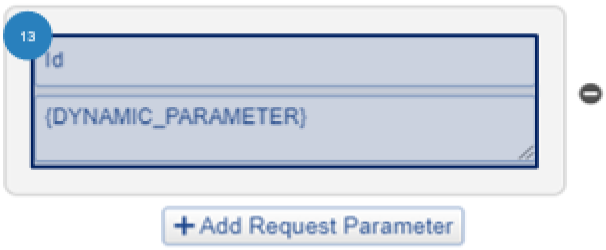
- Select the 11 *+ Add Request Parameter* button. Drag and drop the correct parameter into the parameter value box. 12



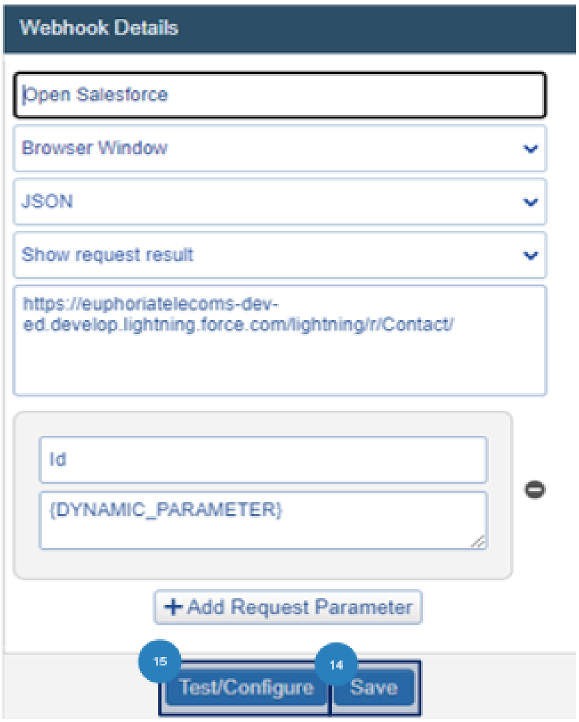
- Add the parameter Name as explained in the Salesforce API document as seen below.

  13

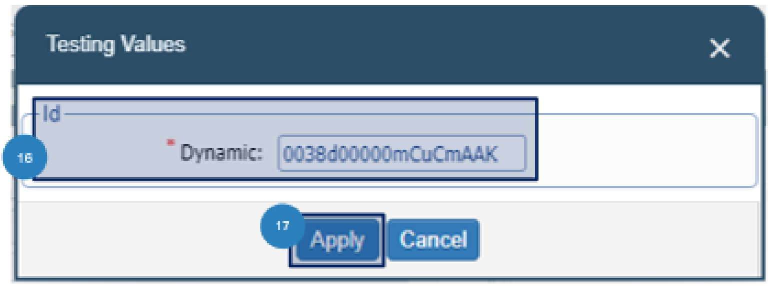
  - Name: Id
  - Parameter Value: {DYNAMIC_PARAMETER}



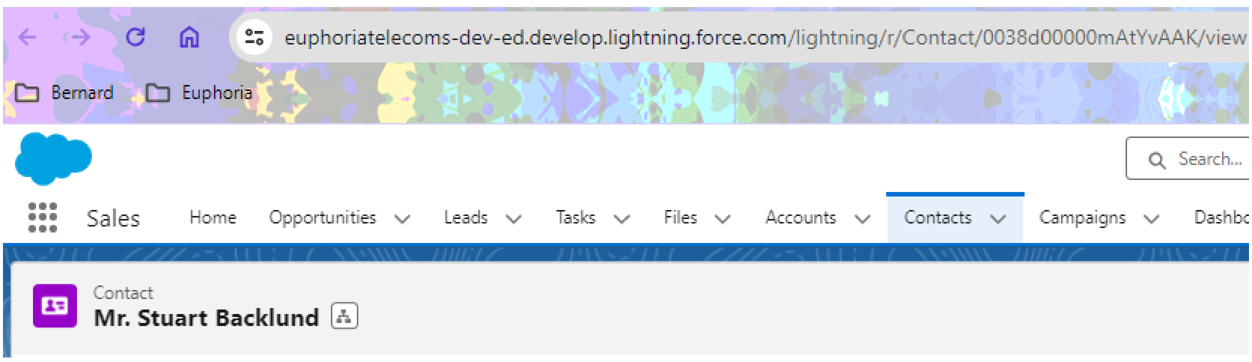
- Click 14 *Save* to create the webhook. The webhook will close. Open the webhook again and select the 15 *Test and Configure* button,



- A pop-up window opens requesting a test value, Add a 16 Contact Id from Salesforce and press 17 *Apply*. This test value is only requested in this step to ensure that a result is given. The Agent will not need to add test values in the normal usage of the webhook.



- The Salesforce Contact page will open for the requested test value.



How to Create the *Contact Lookup by Phone Number* Webhook (Linked to Open Salesforce)

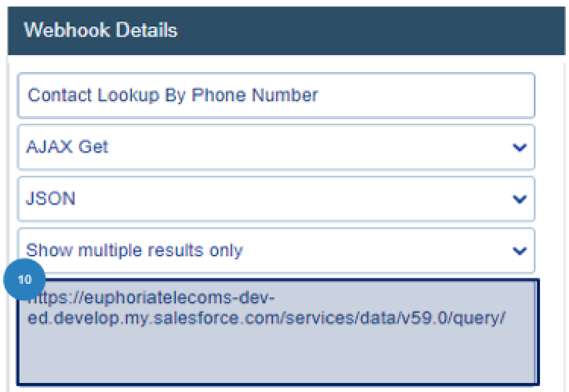
The Webhook is one of 3 webhooks that allows an agent to have a look if the caller has a profile in Salesforce and to view their information. This Webhook will automatically trigger at the beginning of a call. If the caller is a Salesforce contact, or the agent calls a Salesforce contact, their profile information will show. The second and third Webhooks will be used to search for a caller during the call.

To create the webhook, follow steps 1 - 5 as per the How to Create the Add This Contact Webhook.

- Give the webhook a 6 name: Used to identify the webhook, it is helpful to make this descriptive. Select the 7 *AJAX Get Webhook* Type. Select the 8 JSON *Webhook Data Type*. Select the 9 **Show Multiple Result* Response Type, which indicates how the result should be handled.

***Note: When testing the webhook, *select Show Request Result. This will open a result page, even when there is only one result, allowing the selection of viewable fields which is covered in later steps*.**

- Add the Salesforce Classic Customer Domain URL that the webhook needs to access. The first part of the URL is the company URL, and the second part is the page that needs to be accessed.

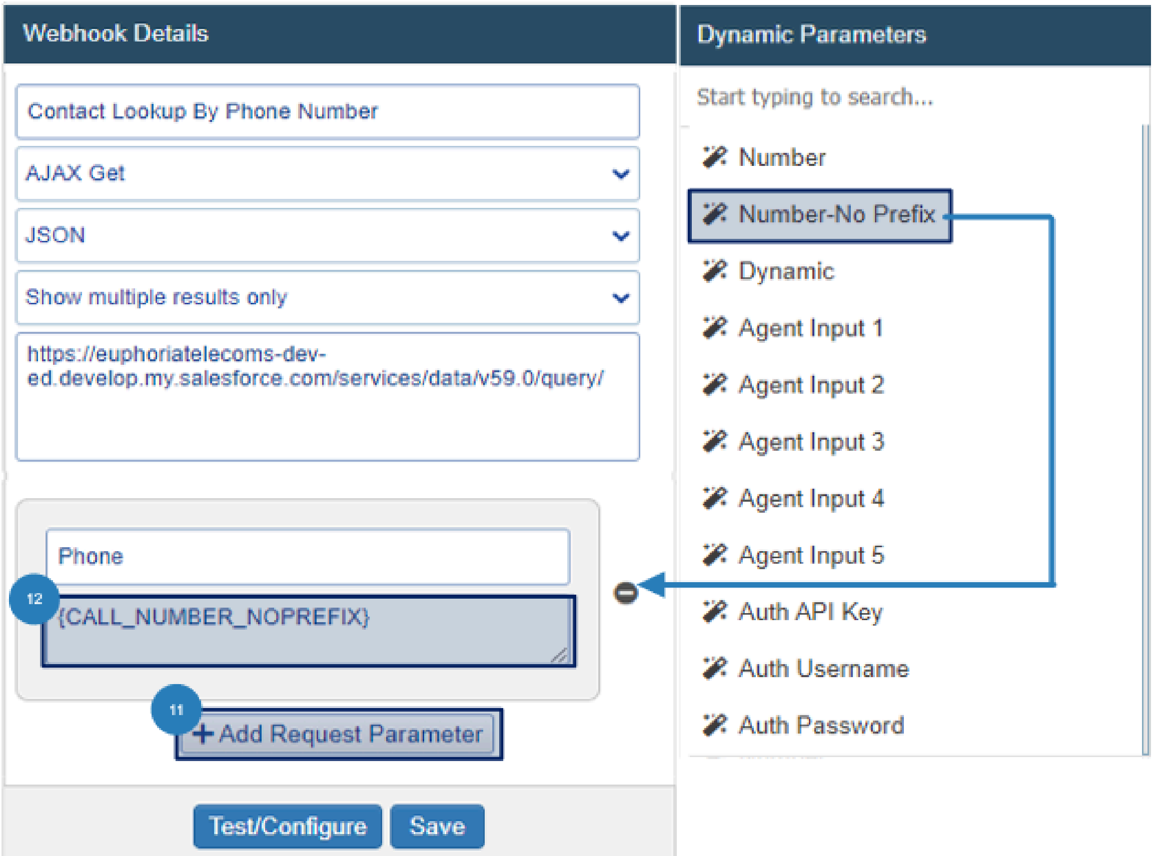


Add parameters:

Dynamic Parameters are added when the request parameter button is selected. These variables are found in the target system's API documentation, and the capitalisation as per the document is important.

https://developer.salesforce.com/docs/atlas.en-us.object_reference.meta/object_reference/sforce_api_objects_contact.htm

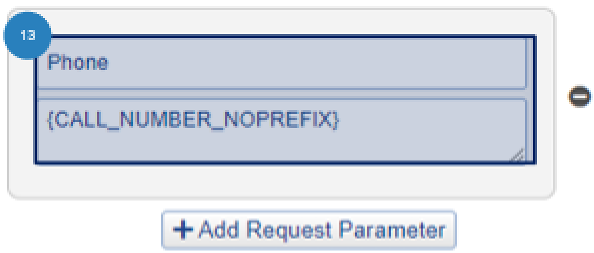
- Select the 11 *+ Add Request Parameter* button. Drag and drop the correct parameter into the parameter value box. 12



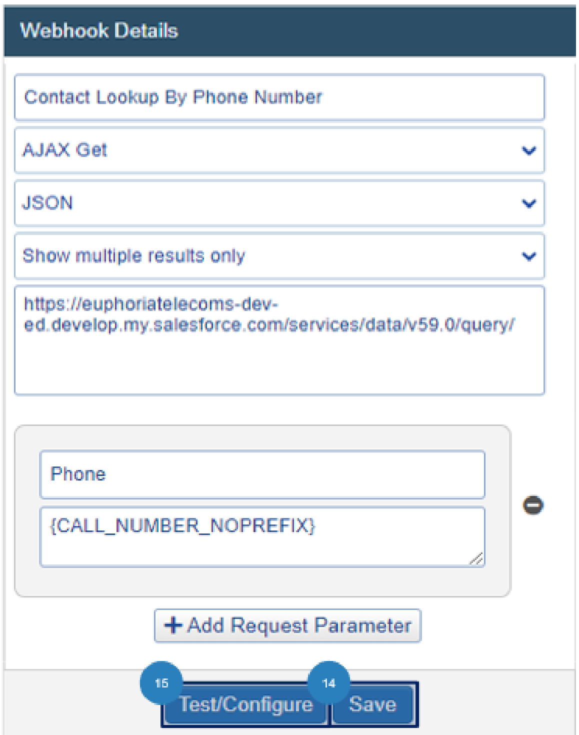
- Add the parameter name as explained in the Salesforce API document as seen below.

  13

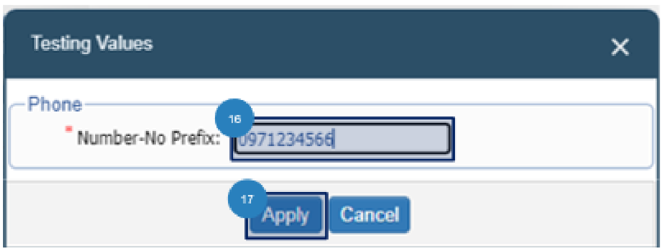
  - Name: Phone
  - Parameter Value: {CALL_NUMBER_NOPREFIX}



- Click 14 *Save* to create the webhook. The webhook will close. Open the webhook again and select the 15 *Test and Configure* button.

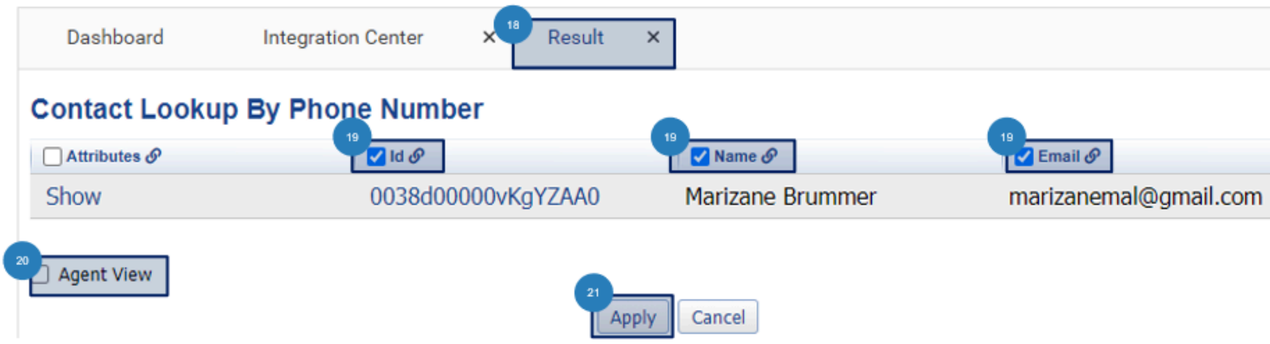


- A pop-up window opens requesting a test value, Add a 16 Phone number of a contact in Salesforce and press 17 *Apply*. This test value is only requested in this step to ensure that a result is given. The Agent will not need to add test values during regular webhook usage.

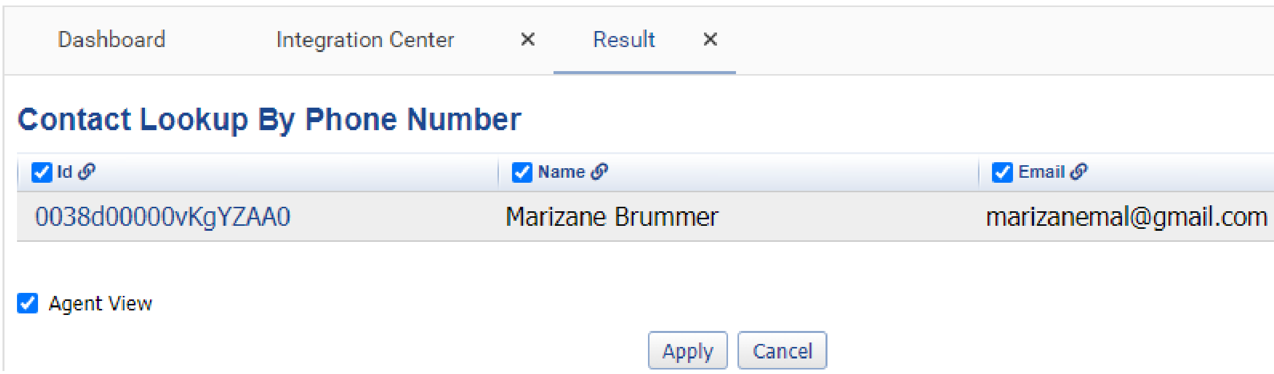


- A 18 *Result* page will open with the information requested by the webhook. For some webhooks, the result page will have many columns of information, but not all of the information will be needed by an agent. Therefore it is important to limit what the agent sees. To do this select the 19 tick box next to the column of information the agent should see. Click on 20 *Agent View* and press 21 Apply to see the results the same way the agent would.

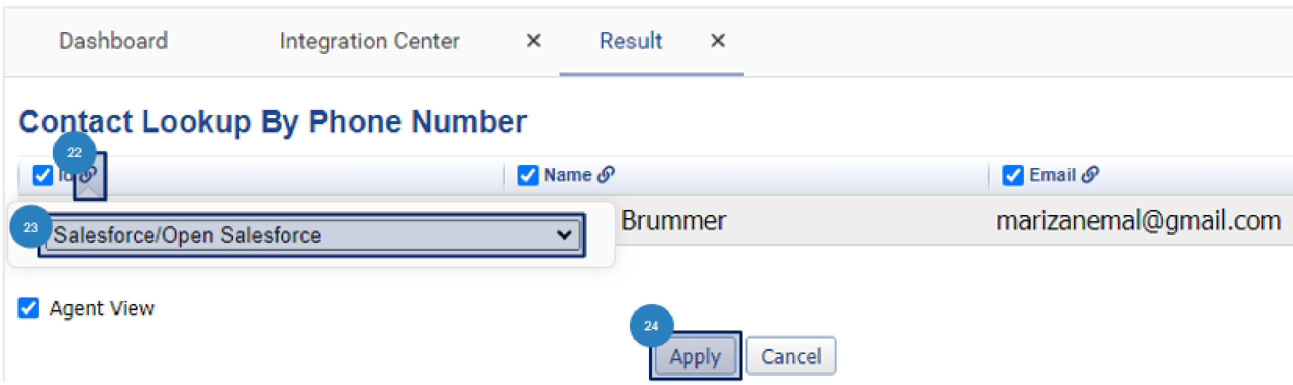
<u>Unfiltered</u>



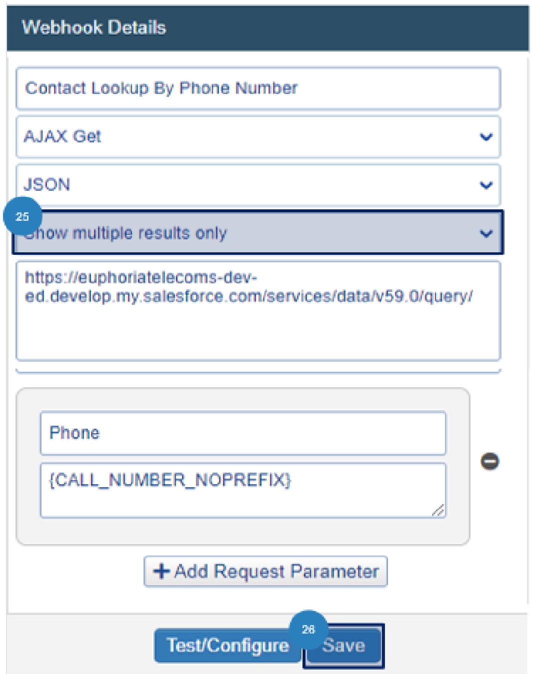
<u>Filtered</u>



- Select the 22 *Id* link chain icon to link another webhook to the *Contact Lookup* Webhook. Select the 23 *Open Salesforce* Webhook. Press 24 *Apply*. This will allow the Contact Lookup webhook to open a Salesforce result when one is available.

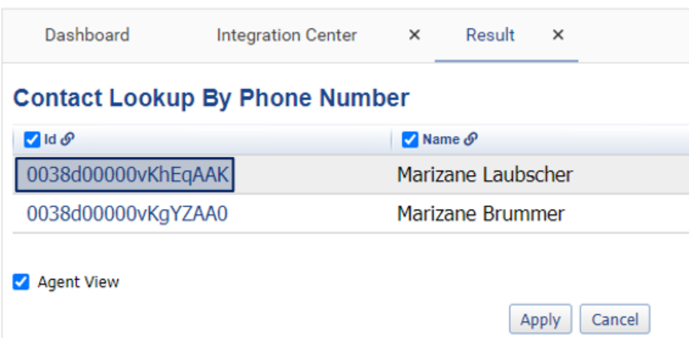


- Once the result shows correctly, change the response type in the webhook to 25 *Show Multiple Results Only. This will allow a single result to open directly on Salesforce*, and a multiple result *to* open on a result page where one of the options can be selected. Click 26 *Save* again to apply the change.



- For singular results, a Salesforce page will open. (Web Browsers might be slow to respond, refresh the TMS page if need be.)

- If multiple contact profiles are available, a result page will open where the Id field is clickable. This will then open the contact in Salesforce.



How to Create the *Alternative Contact Lookup by Phone Number* Webhook (Linked to Open Salesforce)

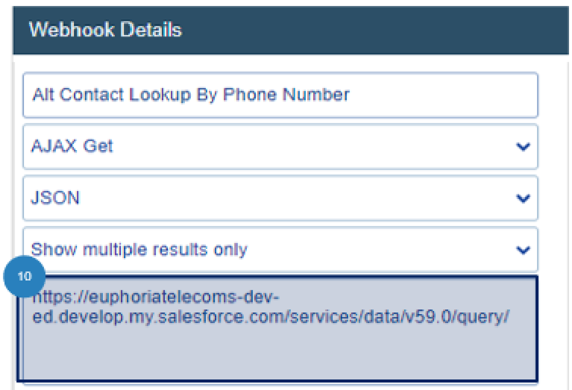
The Alt Contact Lookup by Phone Number Webhook allows agents to look for a caller's contact profile and opens the page in Salesforce to view their information. This will be seen as a button in the Agent workspace.

To create the webhook, follow steps 1 - 5 as per the How to Create the Add This Contact Webhook.

- Give the webhook a 6 name: Used to identify the webhook, it is helpful to make this descriptive. Select the 7 *AJAX Get Webhook* Type. Select the 8 JSON *Webhook Data Type*. Select the 9 **Show Multiple Result* Response Type, which indicates how the result should be handled.

***Note: When testing the webhook, *select Show Request Result. This will open a result page, even when there is only one result, allowing the selection of viewable fields which will be covered in later steps*.**

- Add the Salesforce Classic Customer Domain URL that the webhook needs to access. The first part of the URL is the company URL, and the second part is the page that needs to be accessed.

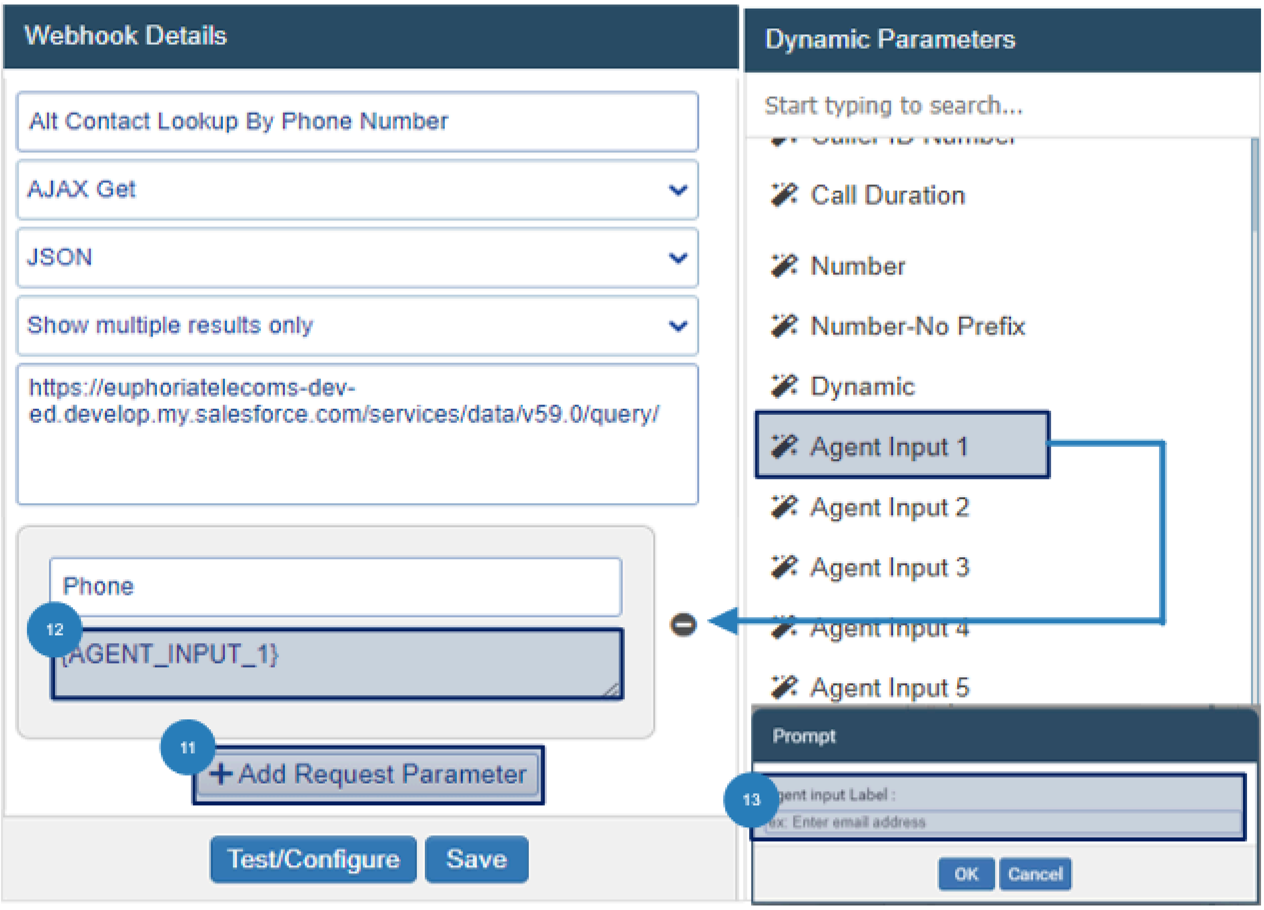


Add a parameter:

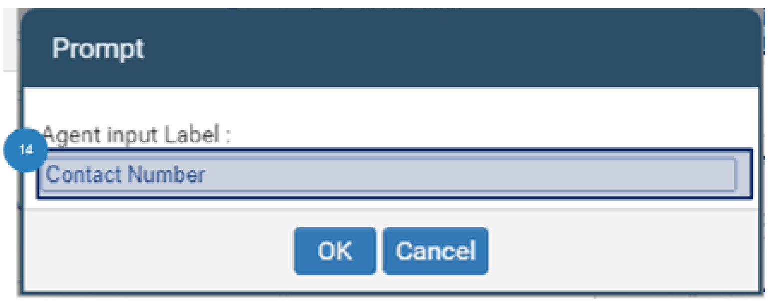
Dynamic Parameters are added when the request parameter button is selected. These variables are found in the target system's API documentation, and the capitalisation as per the document is important.

https://developer.salesforce.com/docs/atlas.en-us.object_reference.meta/object_reference/sforce_api_objects_contact.htm

- Select the 11 + *Add Request Parameter* button. Drag and drop the desired parameter into the 12 parameter value box. This will open the 13 Agent Input label prompt.



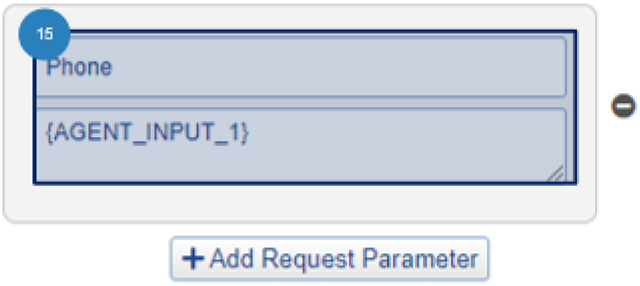
- This label is what the agent will see when the value is requested. The label must say 14 *Contact Number* to find the contact in Salesforce.



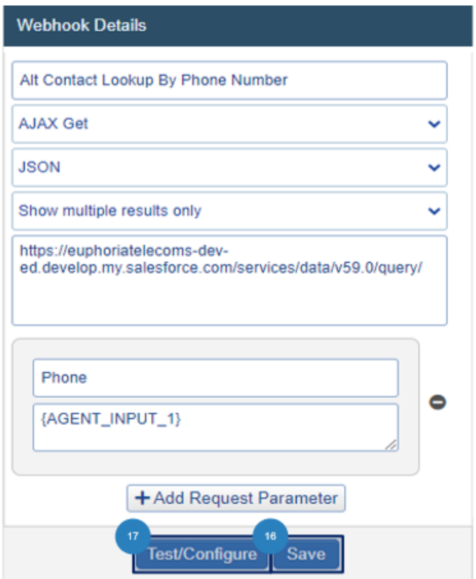
- Add the parameter name as explained in the Salesforce API document, as seen below.

  15

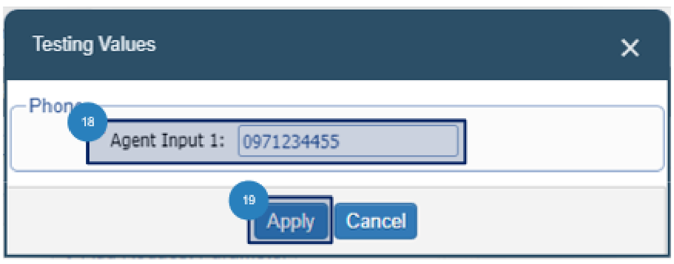
  - Name: Phone
  - Parameter Value: {AGENT_INPUT_1}



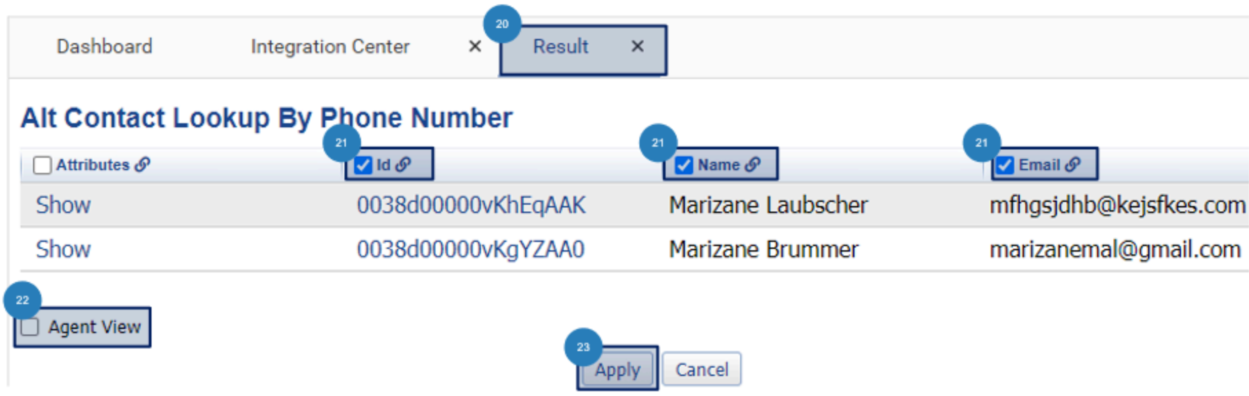
- Click 16 *Save* to create the webhook. The webhook will close. Open the webhook again and select the 17 *Test and Configure* button.



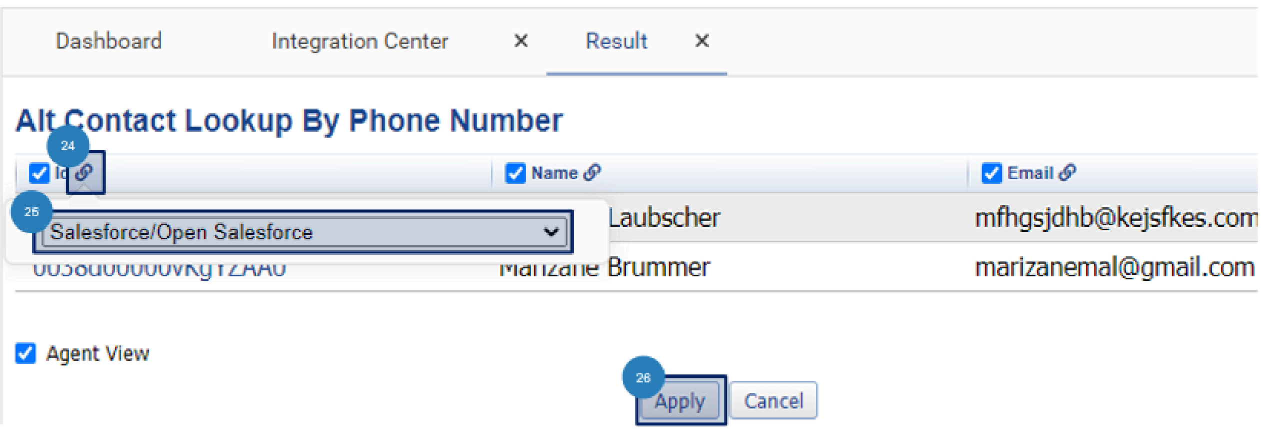
- A pop-up window opens requesting a test value, Add a 18 Phone number of a contact in Salesforce and press 19 *Apply*. This test value is only requested in this step to ensure that a result is given. The Agent will not need to add test values during regular webhook usage.



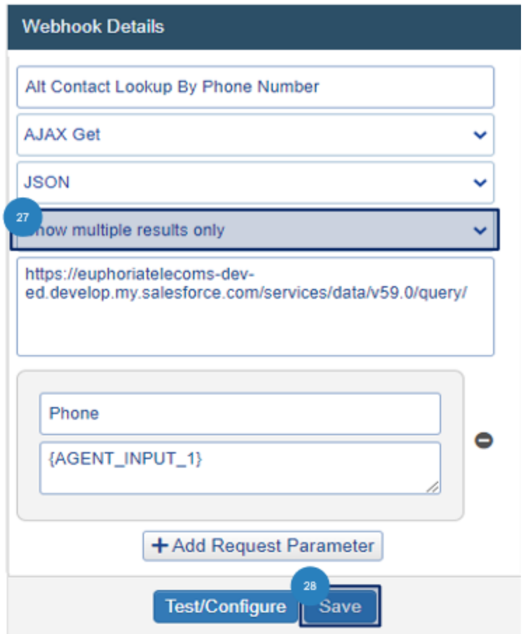
- A 20 *Result* page will open with the information requested by the webhook. For some webhooks, the result page will have many columns of information, but not all of the information will be needed by an agent. Therefore it is important to limit what the agent sees. To do this select the 21 tick box next to the column of information the agent should see. Click on 22 *Agent View* and press 23 Apply to see the results the same way the agent would.



- Select the 24 *Id* link chain icon to link another webhook to the Alt *Contact Lookup By Phone Number* webhook. Select the 25 *Open Salesforce* Webhook. Press 26 *Apply*. This will allow the Alt *Contact Lookup By Phone Number* webhook to open a Salesforce result when one is available.

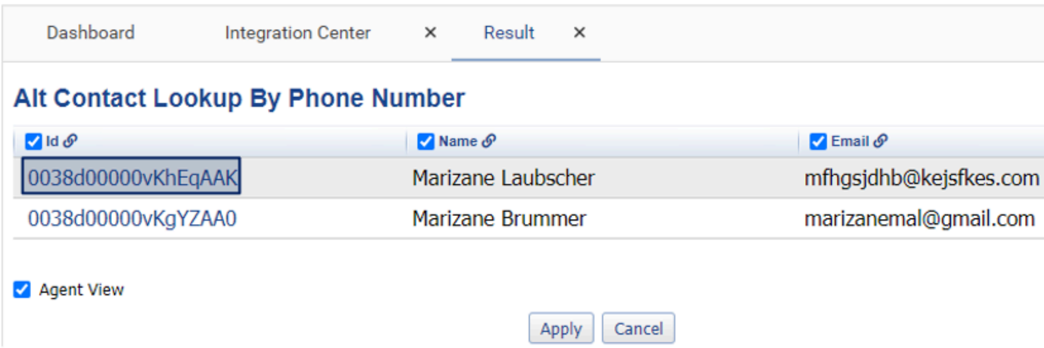


- Once the result shows correctly, change the response type in the webhook to 27 *Show Multiple Results Only. This will allow a single result to open directly on Salesforce*, and a multiple result *to* open on a result page where one of the options can be selected. Click 28 *Save* again to apply the change.



- For singular results, a Salesforce page will open. (Web Browsers might be slow to respond, refresh the TMS page if need be.)

- If multiple contact profiles are available, a result page will open, where the Id field is clickable. Clicking this will open the contact in Salesforce.



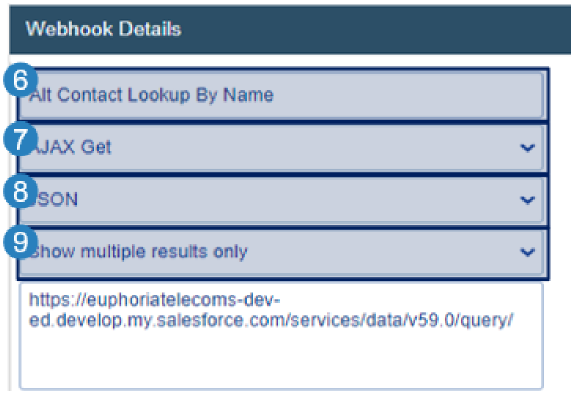
How to Create the *Alternative Contact Lookup by Name* Webhook (Linked to Open Salesforce)

The Alt Contact Lookup by Name Webhook will allow agents to search for a contact that exists in Salesforce with a full or partial name. The results will show any and all names related to the letters that have been requested on a results page. If multiple contacts are shown in the results sections, the agent will be able to select a contact and delve into their details further on Salesforce.

This webhook will be seen as a button in the Agent workspace and will be selectable. This is useful as not all calls that the agent makes or receives will be related to contacts in Salesforce.

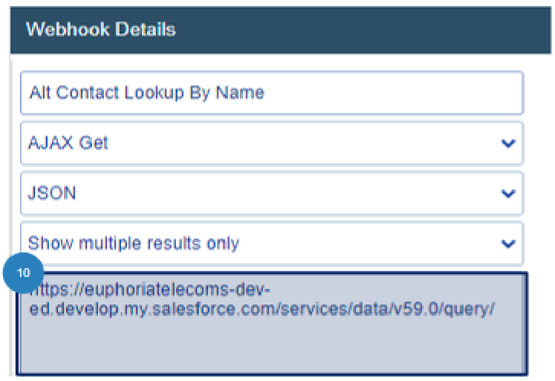
To create the webhook, follow steps 1 - 5 as per the How to Create the Add This Contact Webhook.

- Give the webhook a 6 name: Used to identify the webhook, it is helpful to make this descriptive. Select the 7 *AJAX Get Webhook* Type. Select the 8 JSON *Webhook Data Type*. Select the 9 **Show Multiple Result* Response Type, which indicates how the result should be handled.



***Note: When testing the webhook, *select Show Request Result. This will open a result page, even when there is only one result, allowing the selection of viewable fields which will be covered in later steps*.**

- Add the Salesforce Classic Customer Domain URL that the webhook needs to access. The first part of the URL is the company URL, and the second part is the page that needs to be accessed.

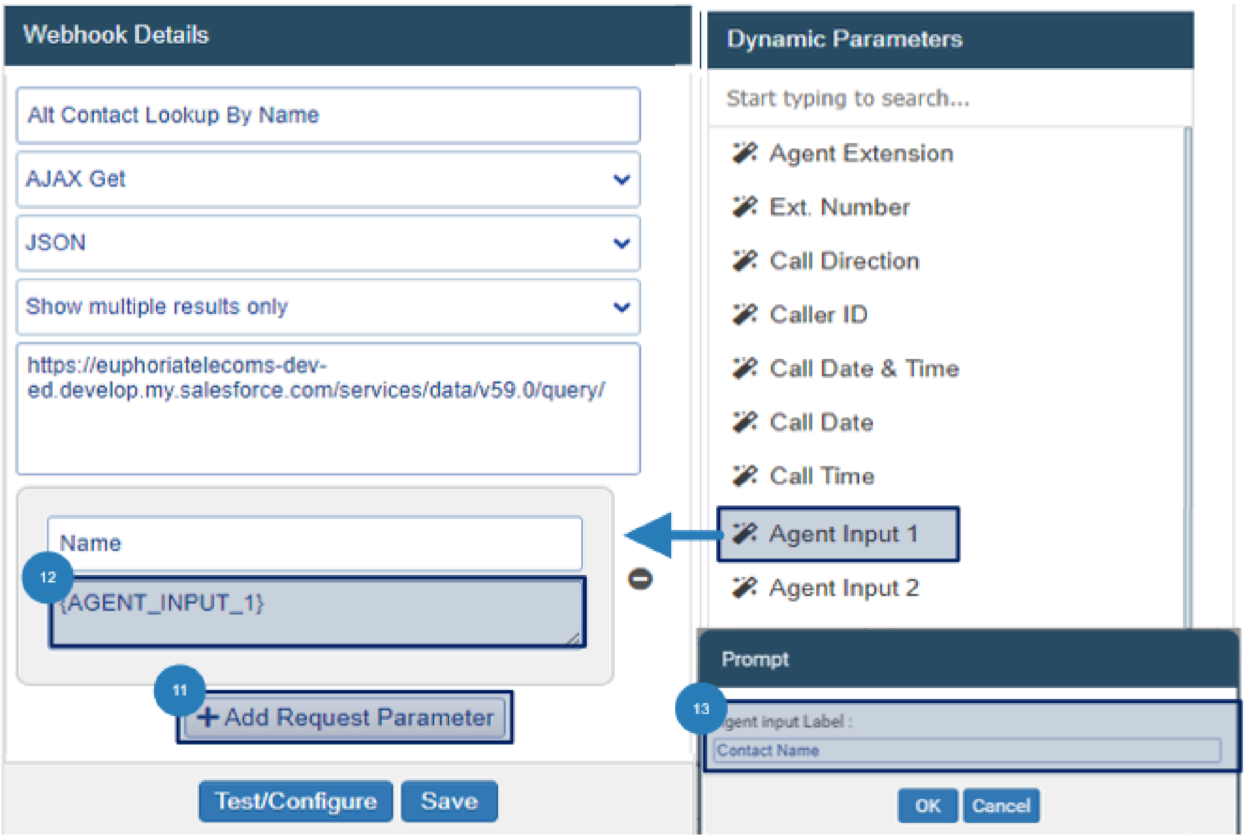


Add parameters:

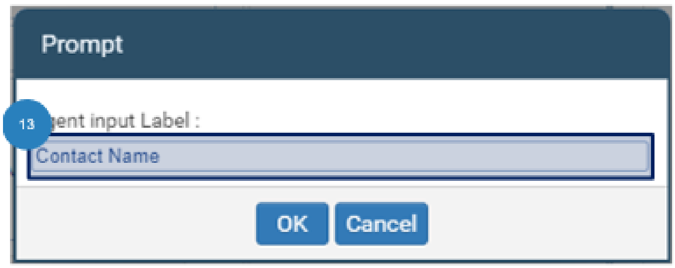
Dynamic Parameters are added when the request parameter button is selected. These variables are found in the target system's API documentation, and the capitalisation as per the document is important.

https://developer.salesforce.com/docs/atlas.en-us.object_reference.meta/object_reference/sforce_api_objects_contact.htm

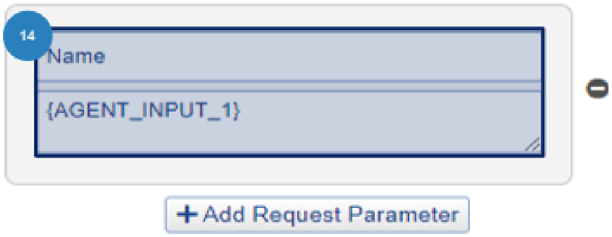
- Select the 11 + *Add Request Parameter* button. Drag and drop the correct parameter into the parameter value box. 12 This will open the 13 Agent Input label prompt.



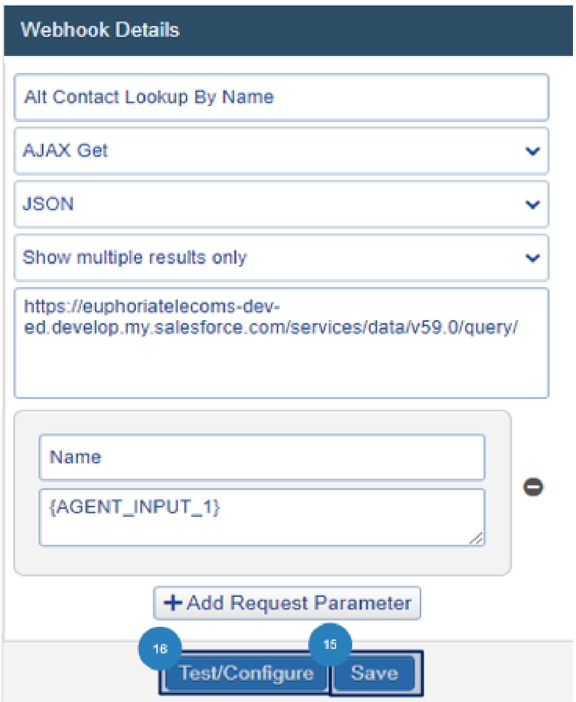
- This label is what the agent will see when the value is requested.



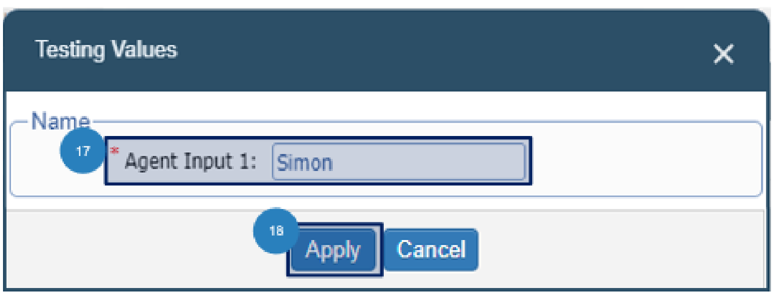
- Add the parameter name as explained in the Salesforce API document as seen below. 14
    - Name: Name
    - Parameter Value: {AGENT_INPUT_1}



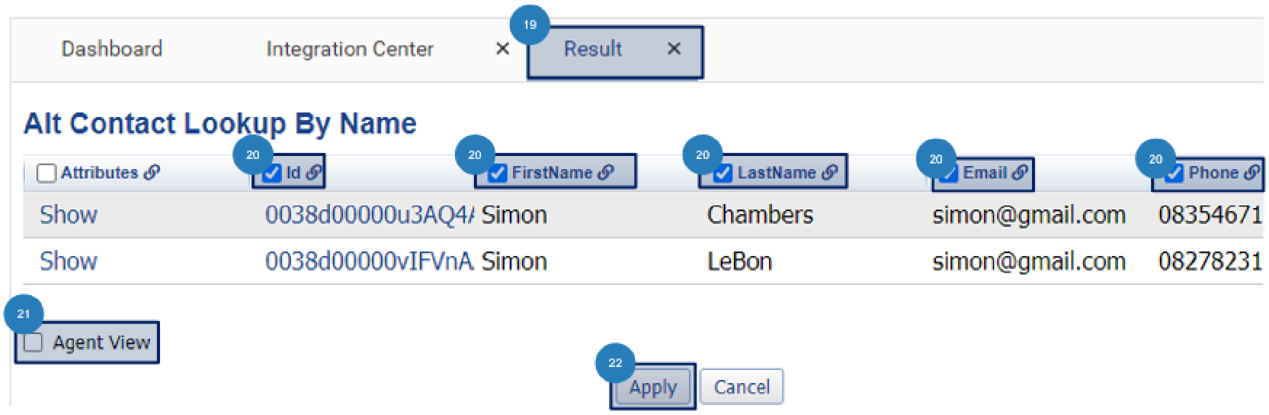
- Click 15 *Save* to create the webhook. The webhook will close. Open the webhook again and select the 16 *Test and Configure* button,



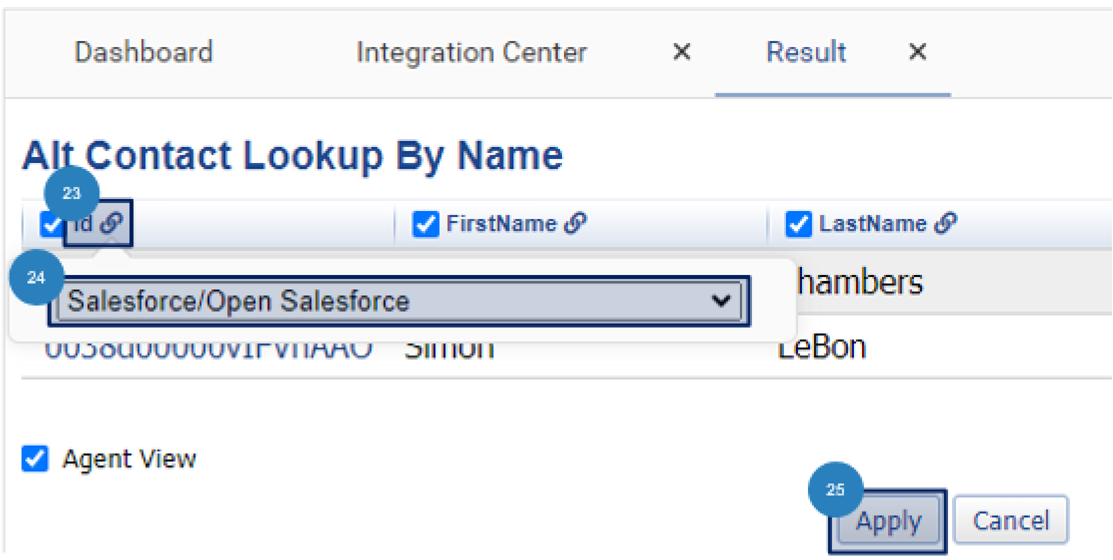
- A pop-up window opens requesting a test value, Add a 17 Name, or the start of a name of a contact in Salesforce and press 18 *Apply*. This test value is only requested in this step to ensure that a result is given. The Agent will not need to add test values during regular webhook usage.



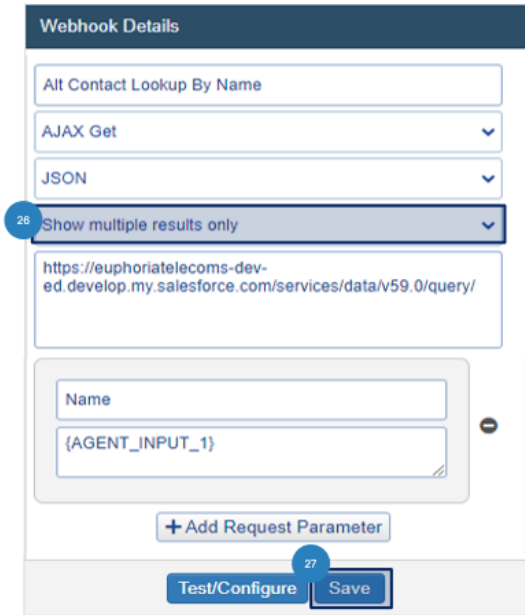
- A 19 *Result* page will open with the information requested by the webhook. For some webhooks, the result page will have many columns of information, but not all of the information will be needed by an agent. Therefore it is important to limit what the agent sees. To do this select the 20 tick box next to the column of information the agent should see. Click on 21 *Agent View* and press 22 Apply to see the results the same way the agent would.



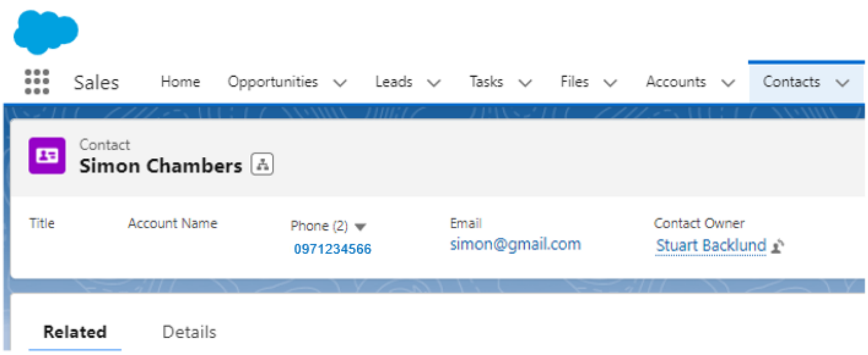
- Select the 23 *Id* link chain icon to link another webhook to the Alt *Contact Lookup By Name* webhook. Select the 24 *Open Salesforce* Webhook. Press 25 *Apply*. This will allow the Alt *Contact Lookup By Name* webhook to open a Salesforce result when one is available.



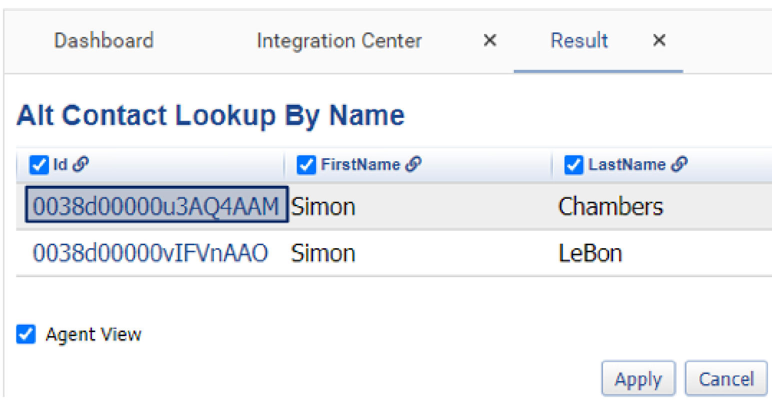
- Once the result shows correctly, change the response type in the webhook to 26 *Show Multiple Results Only. This will allow a single result to open directly on Salesforce*, and a multiple result *to* open on a result page where one of the options can be selected. Click 27 *Save* again to apply the change.



- For singular results, a Salesforce page will open. (Web Browsers might be slow to respond, refresh the TMS page if need be.)



- If multiple contact profiles are available, a result page will open, where the Id field is clickable. Clicking this will open the contact in Salesforce.



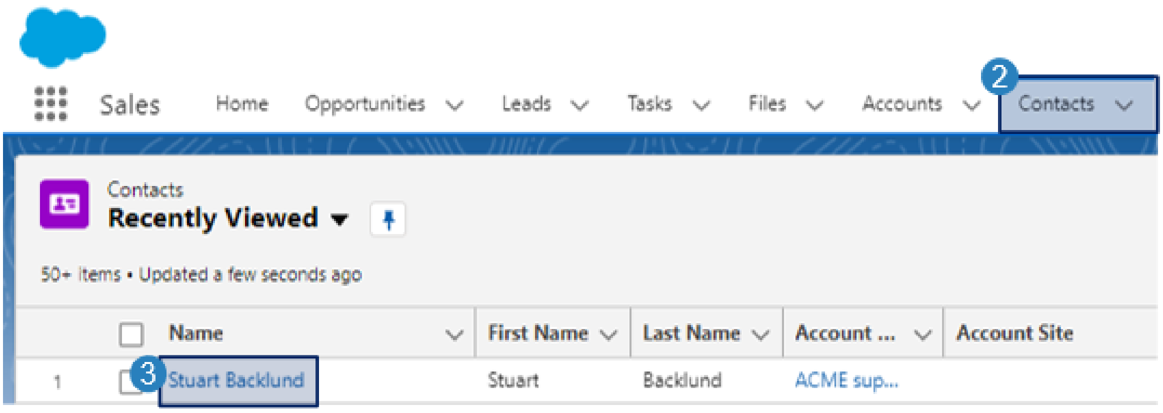
How to Create the *Add Activity to Contact* Webhook

The Add Activity Webhook will be used to add the call activity to a contact's history in Salesforce. The webhook will be linked to the *View Contact By Number and Add Activity webhook* and will not be seen in the Agent Workspace.

In order to create and test this webhook, the user will need the Contact Account ID (not to be confused with the Owner Account ID), which is a contact's ID in Salesforce. This ID will be used during the Test and Configure stage of the webhook creation(This is the WhoID parameter) but will not be needed during the normal use of the webhooks in the Agent Workspace. To locate the Owner ID, follow the below steps:

- In Salesforce select 1 *Switch to Lightning Experience* if not already in this space.

- Select the 2 *Contacts* tab, and select any 3 contact in the list to open it.

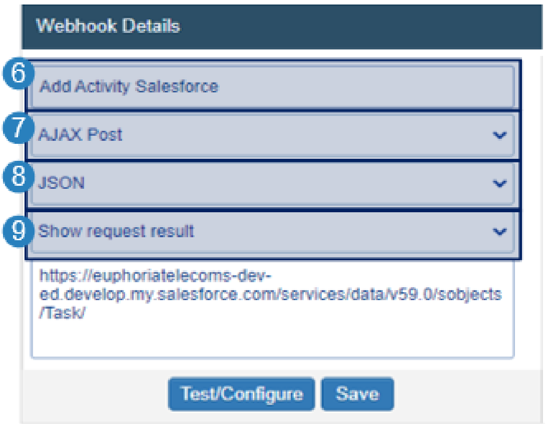


- The Contact ID will be in the URL. 4 Copy and save it for when it's needed in the test values.

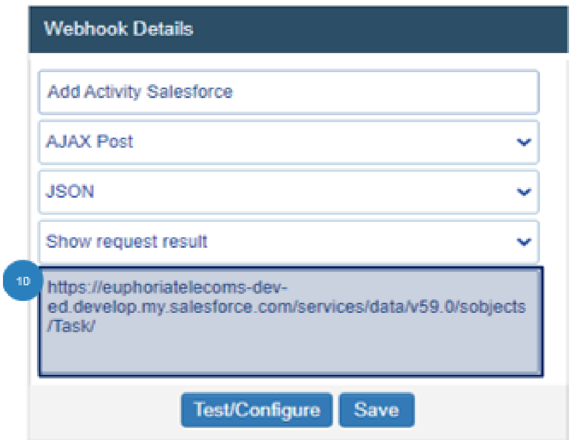
Now that the Contact Account ID is copied and saved, the webhook can be created.

To create the webhook, follow steps 1 - 5 as per the How to Create the Add This Contact Webhook.

- Give the webhook a 6 name: Used to identify the webhook, it is helpful to make this descriptive. Select the 7 *AJAX Post* Webhook Type. Select the 8 JSON *Webhook Data Type*. Select the 9 *Show Request Result* Response Type, which indicates how the result should be handled.



- Add the Salesforce Classic Customer Domain URL that the webhook needs to access. The first part of the URL is the company URL, the second part is the page that needs to be accessed.

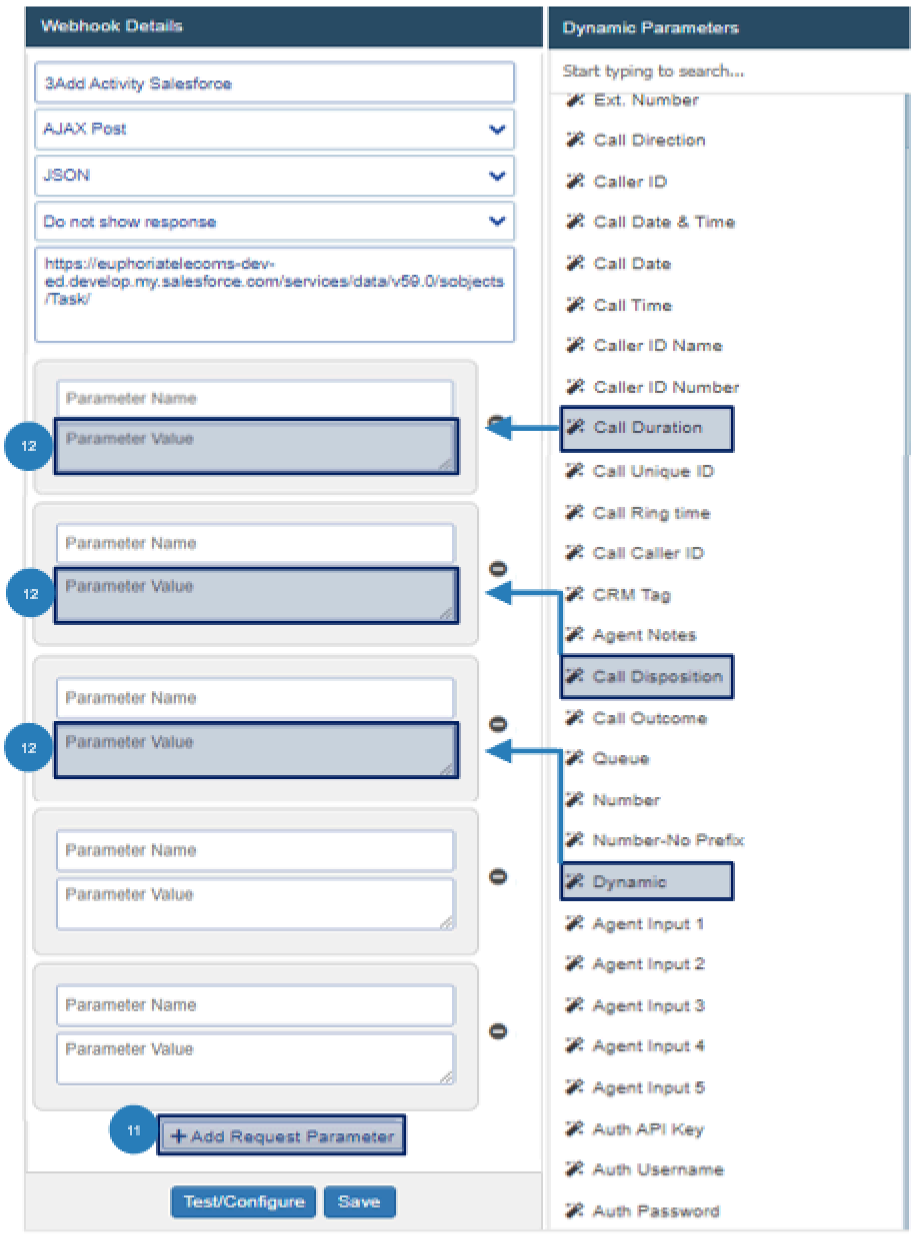


Add parameters

Dynamic Parameters are added when the request parameter button is selected. These variables are found in the target system's API documentation, and the capitalisation as per the document is important.

https://developer.salesforce.com/docs/atlas.en-us.object_reference.meta/object_reference/sforce_api_objects_activityhistory.htm

- Select the 11 *+ Add Request Parameter* button. Then drag and drop a 12 Dynamic Parameter into a value box. You will need to do this three separate times to have all the required Dynamic Parameters for the webhook. Two of the webhooks are static and do not need Dynamic Parameters.



- Add the parameter name as explained in the Salesforce API document as seen below.

13

- Name: Subject*(Static Value)
- Parameter Value: Call

*Contains the subject of the task or event.

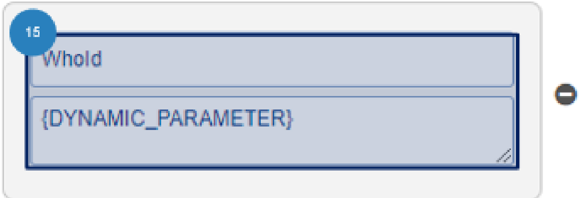
14

- Name: Status (Static Value)
- Parameter Value: Completed

15

- Name: WhoId (Read as “Who ID”)
- Parameter Value: {DYNAMIC_PARAMETER}

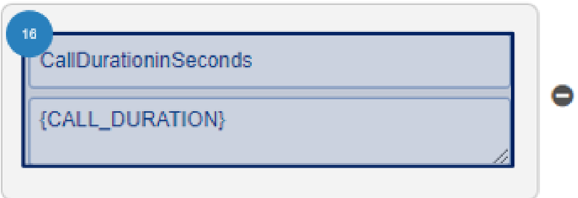
*The WhoId represents a human, such as a lead or a contact. WhoIds are polymorphic. Polymorphic means a WhoId is equivalent to a contact’s ID or a lead’s ID



16

- Name: CallDurationinSeconds*
- Parameter Value: {CALL_DURATION}

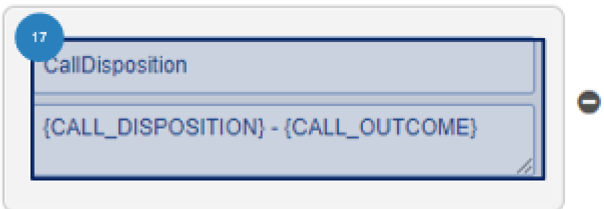
*Call duration measured in seconds and will never contain a decimal example: 12 and not 0:12



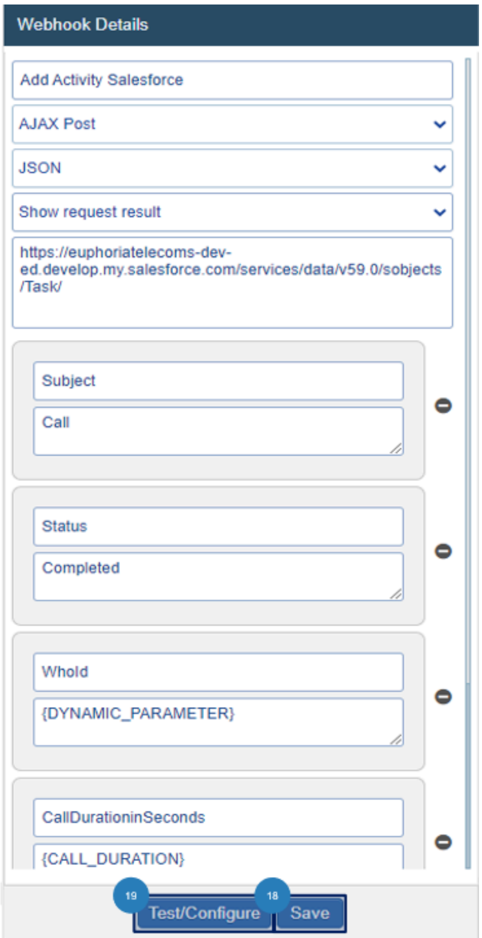
17

- Name: CallDisposition*
- Parameter Value: {CALL_DISPOSITION} - {CALL_OUTCOME}

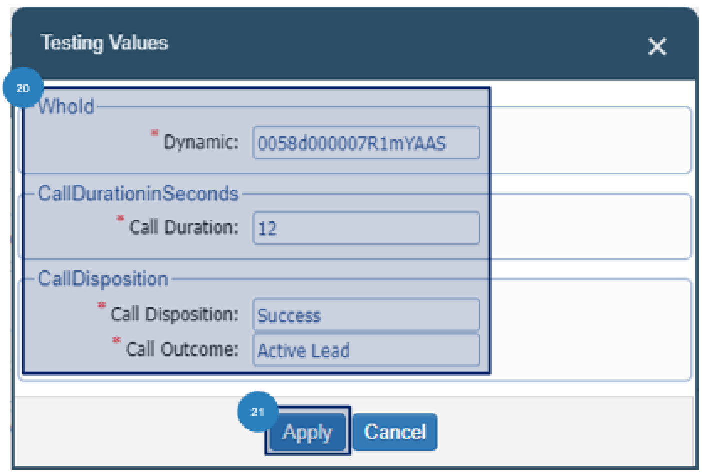
*Call Disposition can be a combination of the Disposition and the Outcome or just the disposition. This depends on the client's preference.



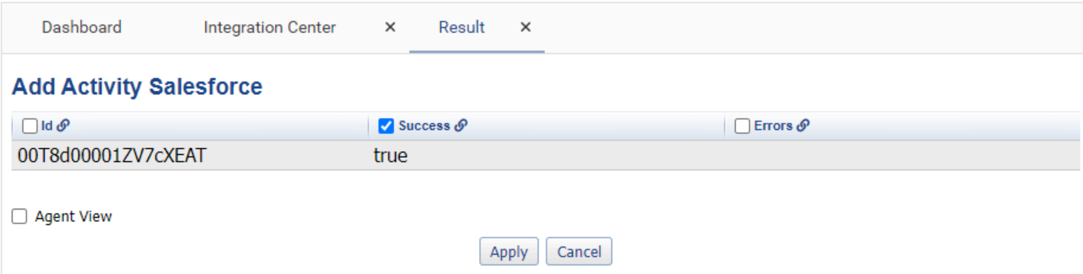
- Click 18 *Save* to create the webhook. The webhook will close. Open the webhook again and select the 19 *Test and Configure* button.



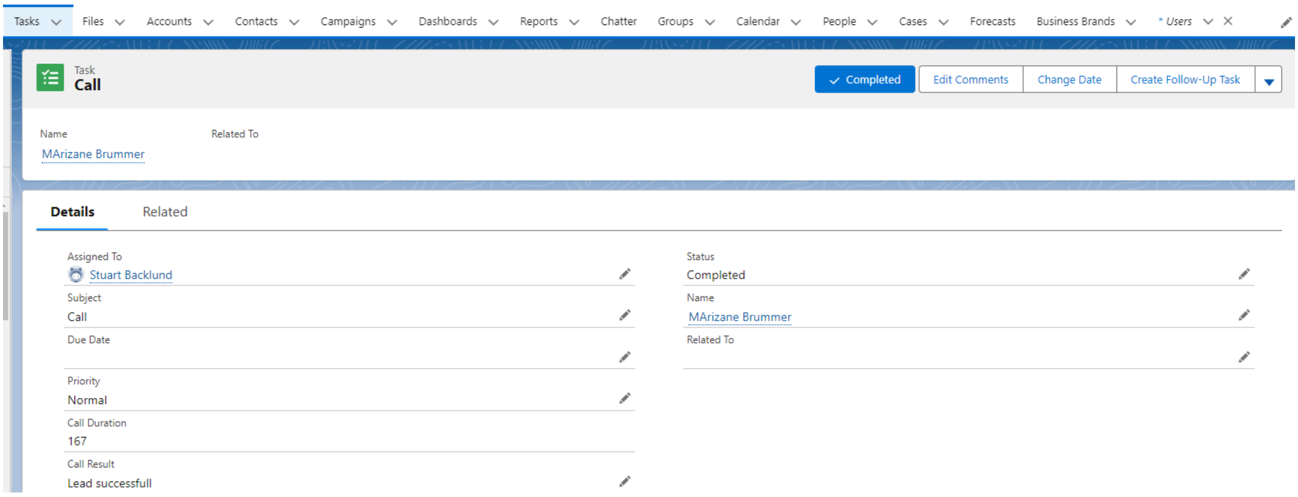
- A pop-up window opens requesting test values, Add the 20 Account Owner ID as saved in the beginning of this webhook creation, add the call duration in Seconds and a Disposition. Press 21 *Apply*. This test value is only requested in this step to ensure that a result is given. The Agent will not need to add test values during regular webhook usage.



- A *Result* page will open with the information requested by the webhook. (The Disposition and Call Duration might not Automatically display; thus, follow the additional information steps below)



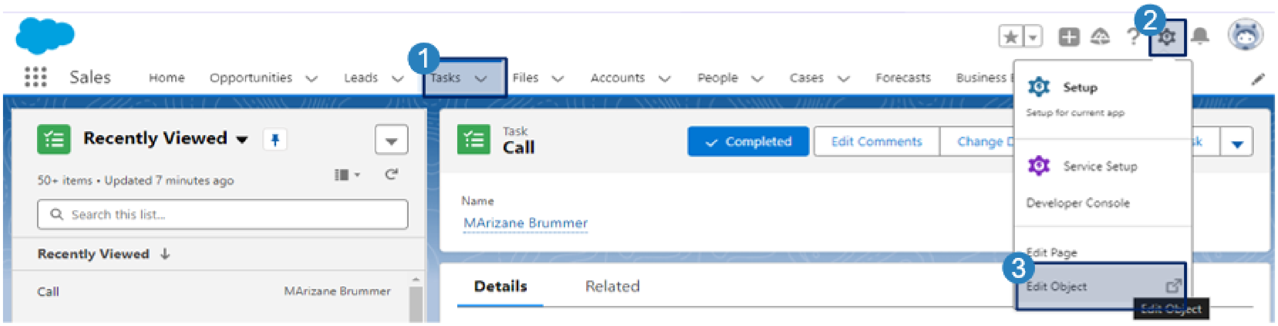
The results of adding a phone call can be viewed on the Salesforce Tasks page.



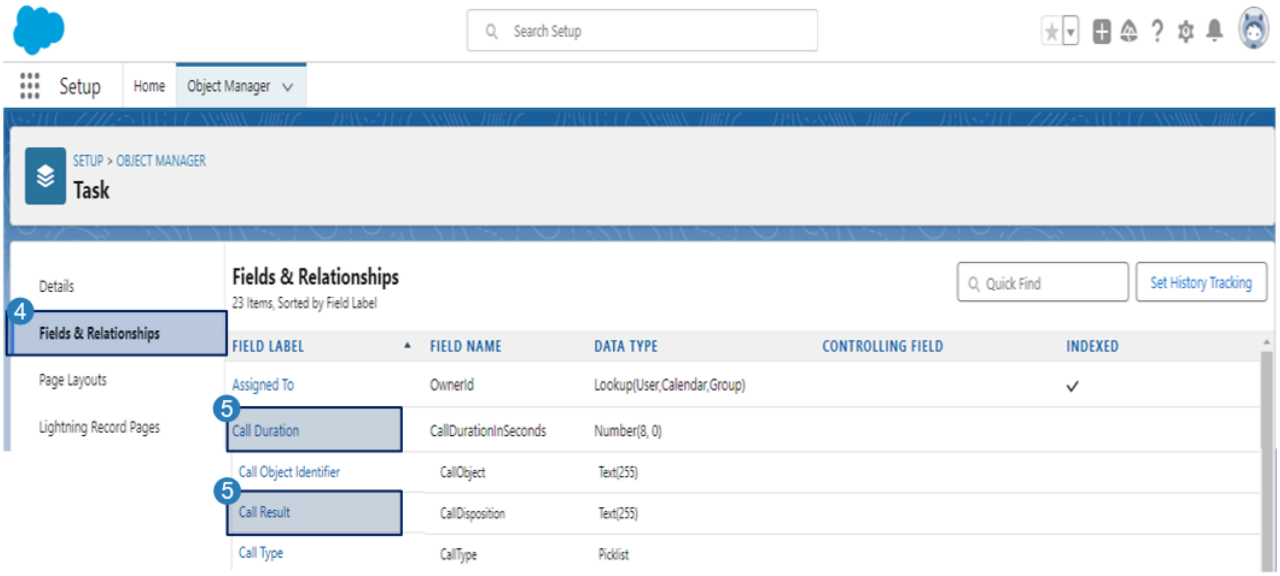
Additional Information

The Disposition and Call Duration might not Automatically display; thus, these two options need to be enabled in Salesforce.

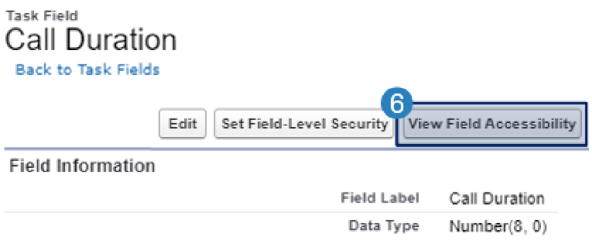
- On the 1 *Task* page, select the 2 settings icon and then 3 *Edit Object*. This will open a new page.



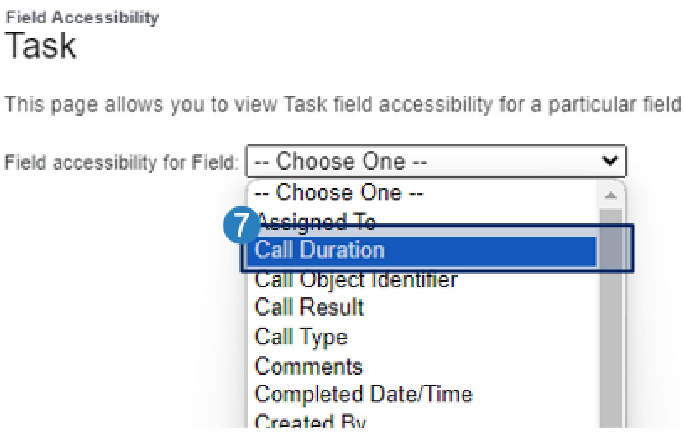
- Select the 4 *Fields & Relationships* option on the left menu bar. Click on the 5 *Call Duration* Field Label to enable the Call Duration to be seen. The next few steps will follow the Call Duration, but are exactly the same for Call Result.



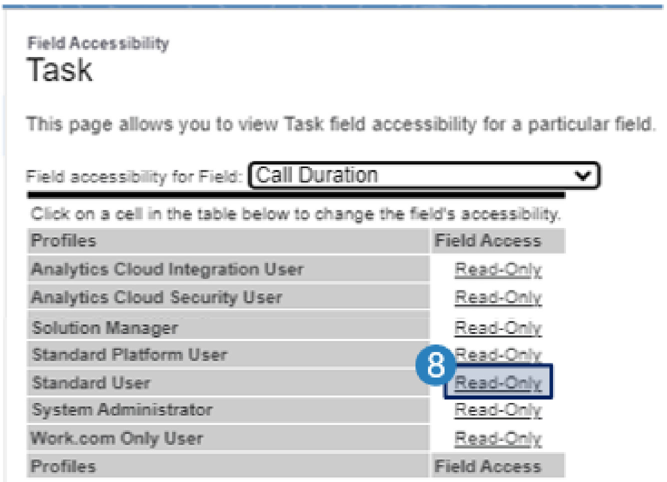
- On the Call duration page, select the 6 *View Field Accessibility* button.



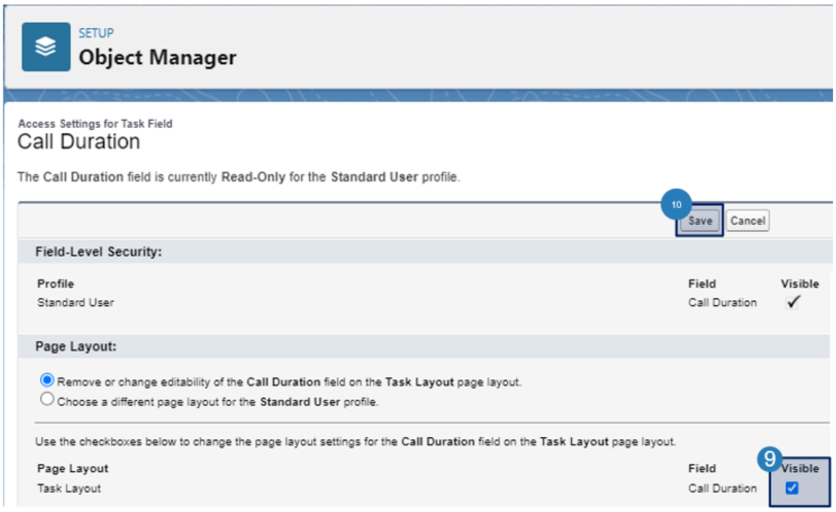
- The field accessibility page will open, Select 7 *Call duration* in the drop down menu. This page allows the user to view Task field accessibility for a particular field and change it.



- Select the 8 Read Only cell in the text table next to the Standard User. This will open the Access Settings for the Task Field Call Duration page.



- Check the 9 *Visible* Check box and click 10 *Save.*

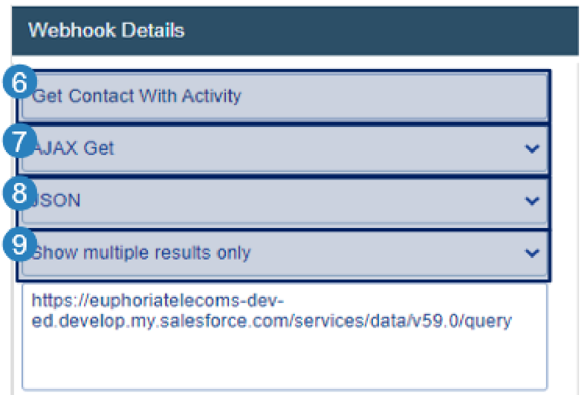


How to Create the Webhook (Linked to Add Activity)

The Get Contact Webhook will allow agents to add the call activity to a client in Salesforce with a full or partial phone Number. If a singular client profile is located the activity will automatically add to their profile when the call is dispositioned. If multiple contacts are found, the results page will open and a client profile can be selected. The call activity will then be added to that client's profile and the results page will close. The Add activity Webhook will be linked to ensure that the call activity is added to the correct contact profile.

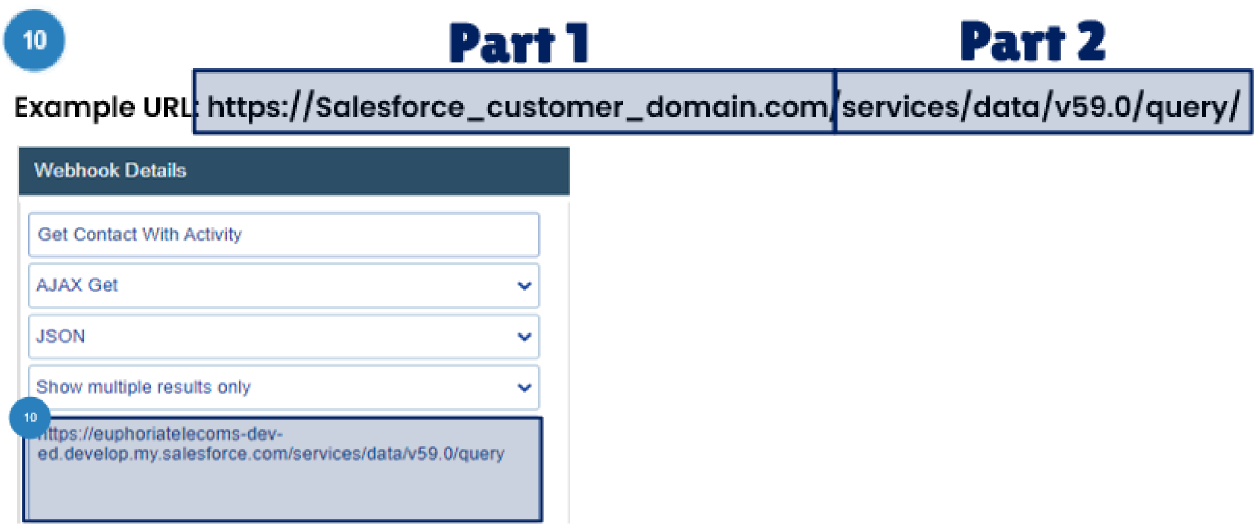
To create the webhook, follow steps 1 - 5 as per the How to Create the Add This Contact Webhook.

- Give the webhook a 6 name: Used to identify the webhook, it is helpful to make this descriptive. Select the 7 *AJAX Get Webhook* Type. Select the 8 JSON *Webhook Data Type*. Select the 9 **Show Multiple Result* Response Type, which indicates how the result should be handled.



***Note: When testing the webhook, *select Show Request Result. This will open a result page, even when there is only one result, allowing the selection of viewable fields which will be covered in later steps*.**

- Add the Salesforce Classic Customer Domain URL that the webhook needs to access. The first part of the URL is the company URL, the second part is the page that needs to be accessed.

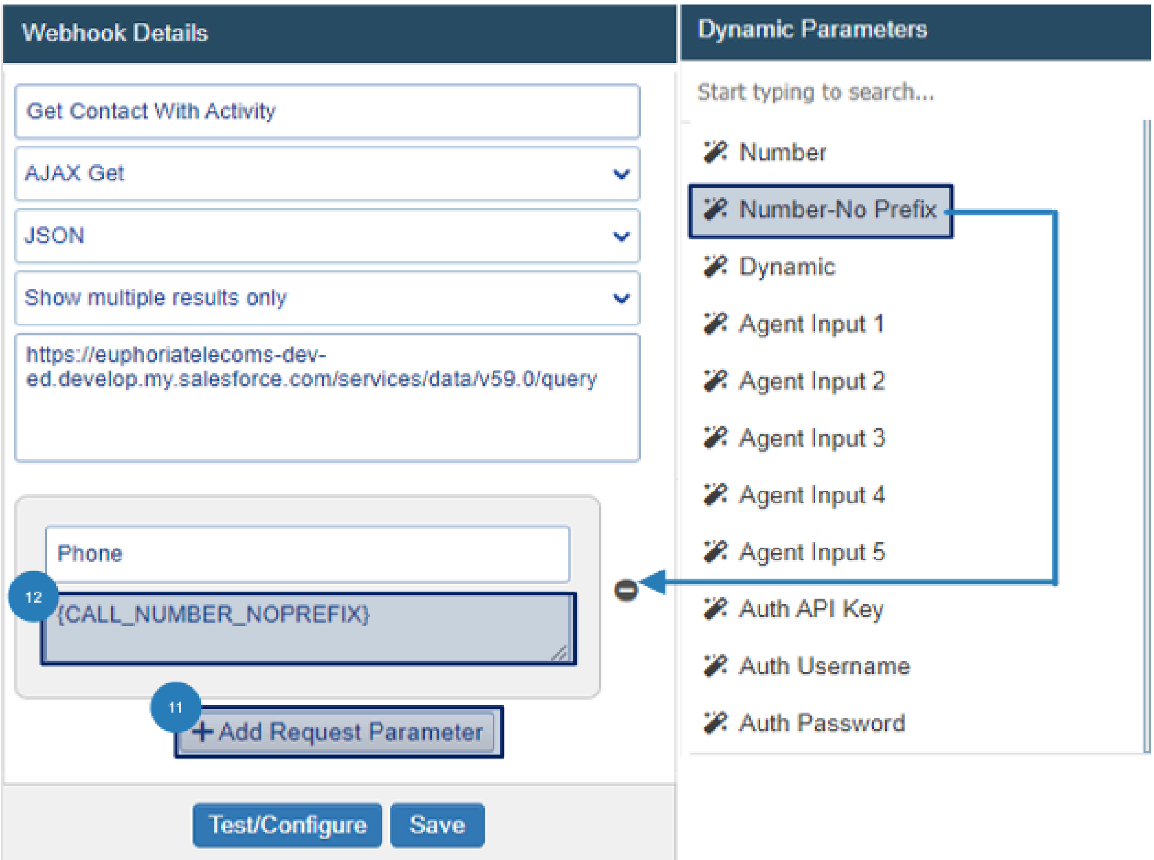


Add parameters:

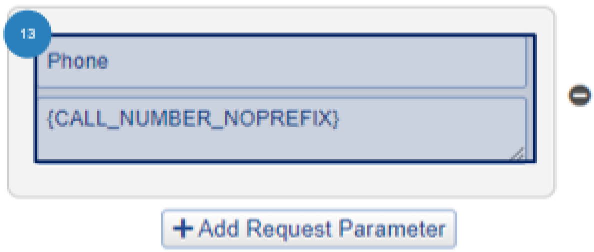
Dynamic Parameters are added when the request parameter button is selected. These variables are found in the target system's API documentation, and the capitalisation as per the document is important.

https://developer.salesforce.com/docs/atlas.en-us.object_reference.meta/object_reference/sforce_api_objects_contact.htm

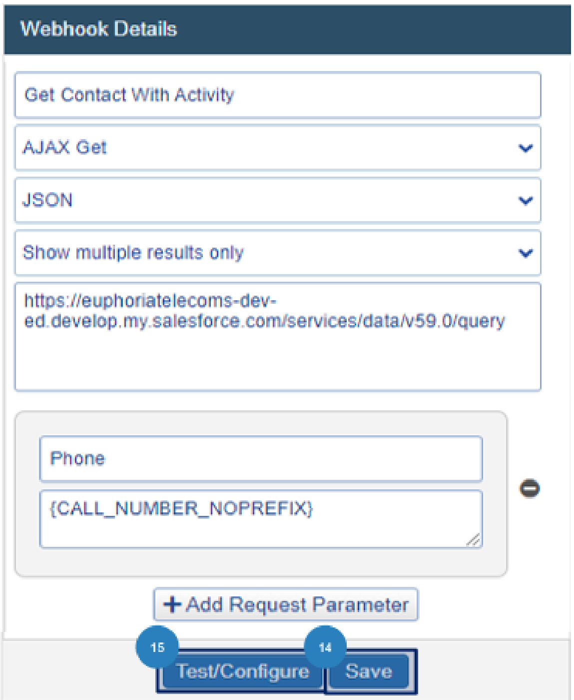
- Select the 11 *+ Add Request Parameter* button. Drag and drop the correct parameter into the parameter value box. 12



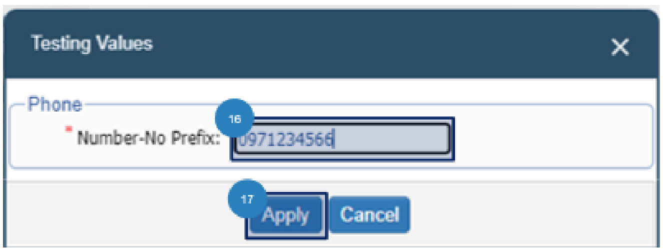
- Add the parameter Name as explained in the Salesforce API document as seen below. 13
  - Name: Phone
  - Parameter Value: {CALL_NUMBER_NOPREFIX}



- Click 14 *Save* to create the webhook. The webhook will close. Open the webhook again and select the 15 *Test and Configure* button,

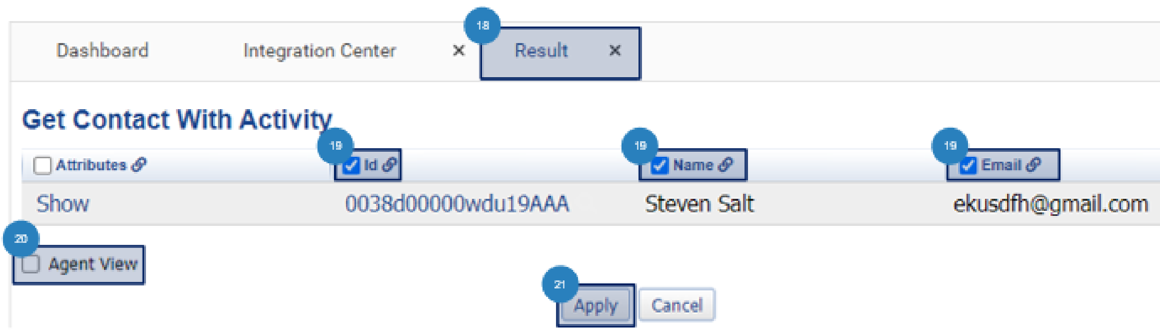


- A pop-up window opens requesting a test value, Add a 16 Phone number of a contact in Salesforce and press 17 *Apply*. This test value is only requested in this step to ensure that a result is given. The Agent will not need to add test values in the normal usage of the webhook.

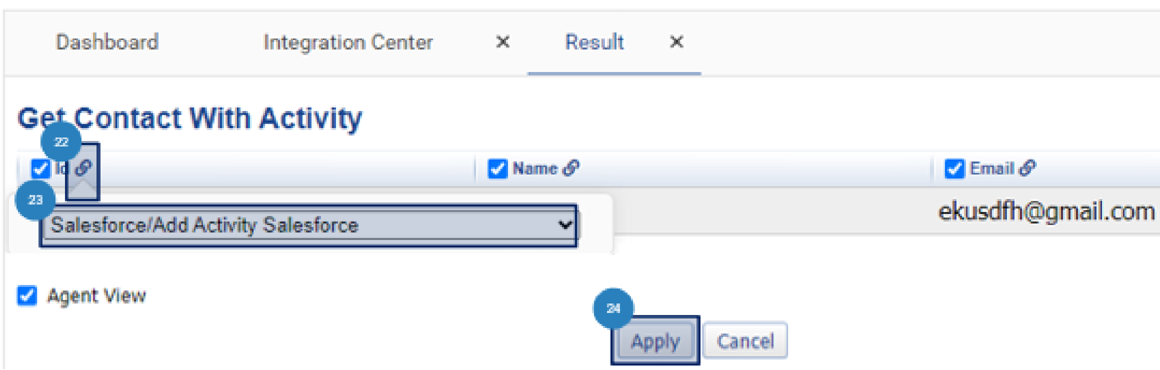


- A 18 *Result* page will open with the information requested by the webhook. For some webhooks, the result page will have many columns of information, but not all of the information will be needed by an agent. Therefore it is important to limit what the agent sees. To do this select the 19 tick box next to the column of information the agent should see. Click on 20 *Agent View* and press 21 Apply to see the results the same way the agent would.

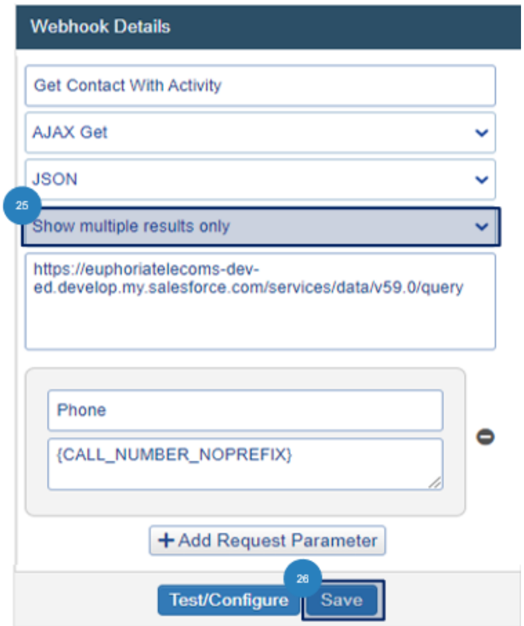
Unfiltered



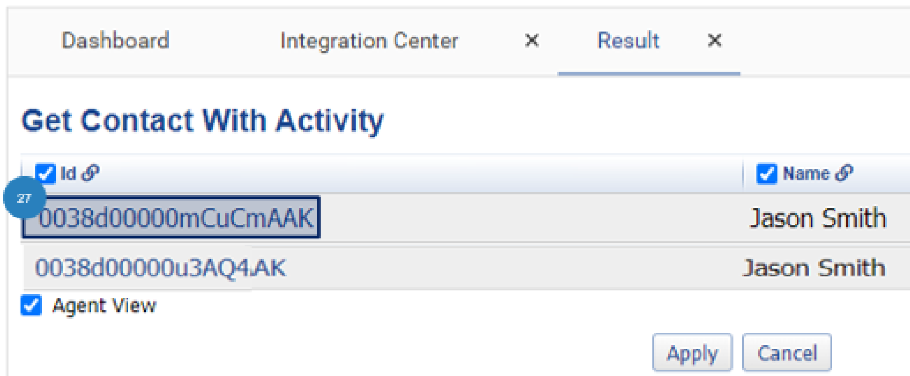
- Select the 22 *Id* link chain icon to link another webhook to the *Get Contact with Activity*. Select the 23 *Add Activity* Webhook. Press 24 *Apply*. This will allow the *Get Contact with Activity* webhook to add a call task to a contact when the caller is a Salesforce client.



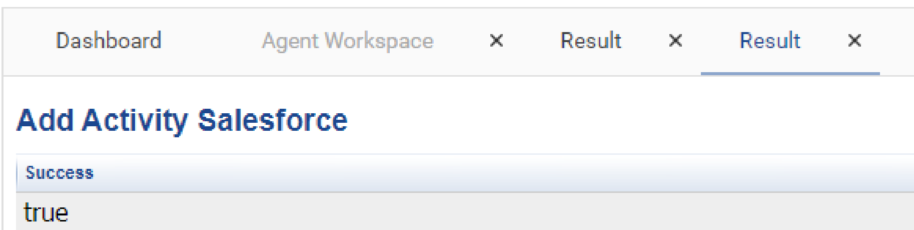
- Once the result shows correctly, change the response type in the webhook to 25 *Show Multiple Results Only. This will allow a single result to send a task directly to the correct contact in Salesforce*, and a multiple result *to* open on a result page where one of the contacts can be selected. Click 26 *Save* again to apply the change.



- For singular results, the activity will add to the contact's task list. If multiple contact profiles are available, a result page will open, where the Id field is clickable. Select the 27 Contact Id to add the disposition to their task list.



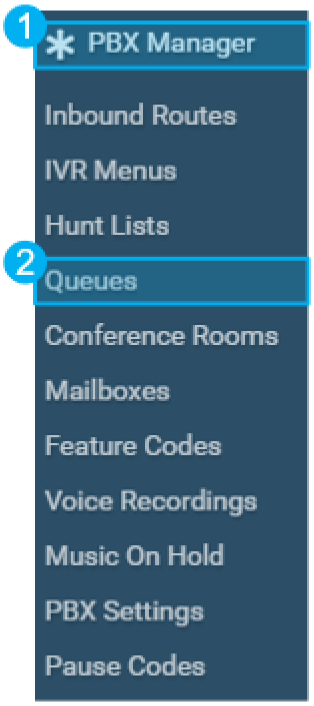
- A second result page will open, showing that the activity has been added successfully.



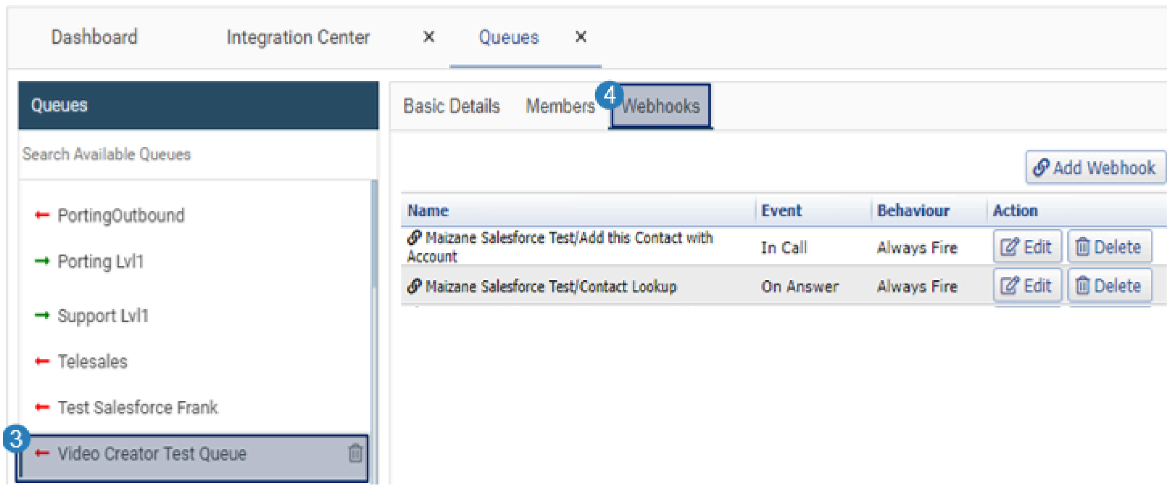
Configuration of Webhooks in Queues

Once a webhook has been created, it can be added to a queue or campaign.

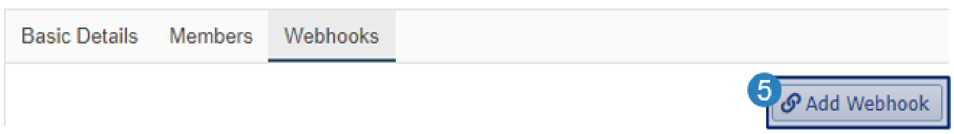
- Click on the 1 *PBX Manager* menu item. Click on the 2 *Queues* sub-menu item.



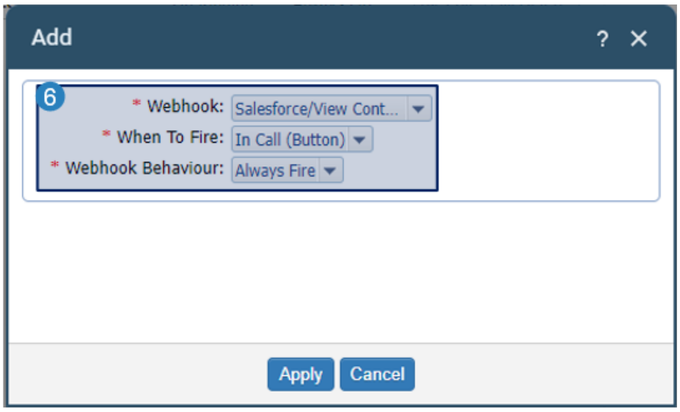
- Choose the desired queue 3. When testing an outbound queue, add the webhook to THAT queue. Select the 4 *Webhooks* tab. Ensure the Agent is part of the selected queue.



- Click the 5 *Add Webhook* button.



- Choose the desired Webhook and its 6 *behaviour.*

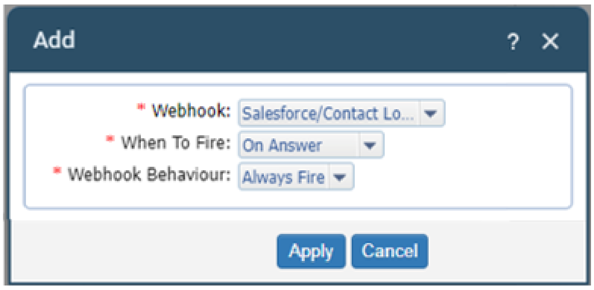


- A form will show requesting three values from the dropdown lists provided
    - Webhook: the webhook name
    - When To Fire: what event should trigger the webhook to fire
    - Webhook Behaviour: Under what condition/behaviour the webhook should fire

Note: There is no need to configure the *Add Activity*, Account Lookup or Open Salesforce webhooks as they are linked in the integration centre.

Contact Lookup By Phone Number

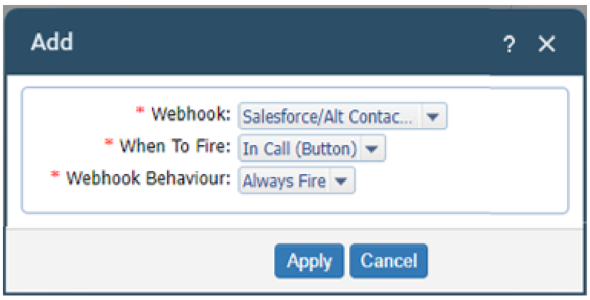
- Select *Contact Lookup By Phone Number*
- Select *On Answer Button*
- Select *Always fire*



Alt Contact Lookup By Phone Number

- Select Alt Contact *View Contact*
- Select *In Call Button*
- Select *Always fire*

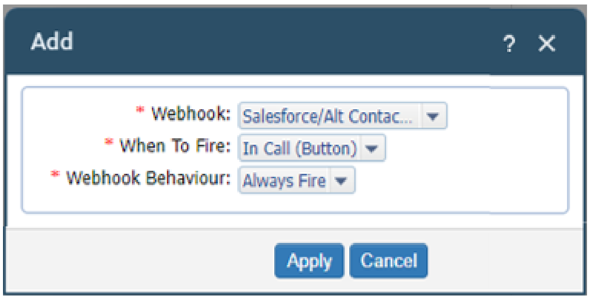
**It* allows the agent to search for a different contact by number whilst on a call if needed, for example, if the caller asks the agent to see if their spouse has a profile.



Alt Contact Lookup By Name

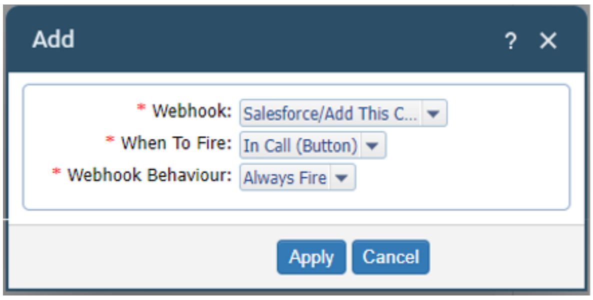
- Select *Get Contact By Name*
- Select *In Call/button*
- Select *Always fire*

**It* allows the agent to search for a different contact by name whilst on a call if needed.



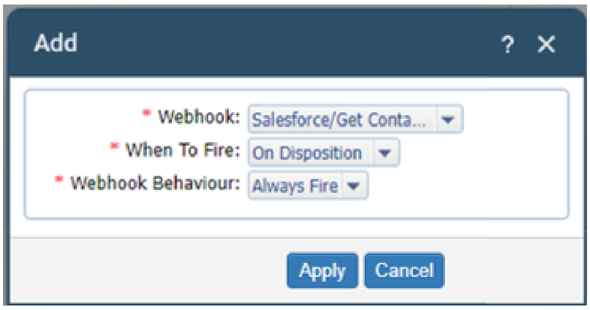
Add This Contact with Account to Salesforce

- Select *Add This Contact with Account*
- Select *In Call Button*
- Select *Always fire*

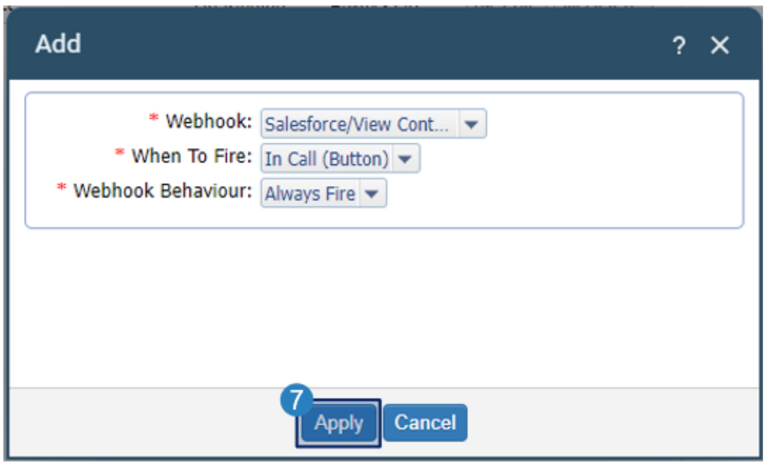


Get Contact with Activity

- Select *Get Contact with Activity*
- Select *On Disposition*
- Select *Always fire*



- Click 7 *Apply* to save the changes.

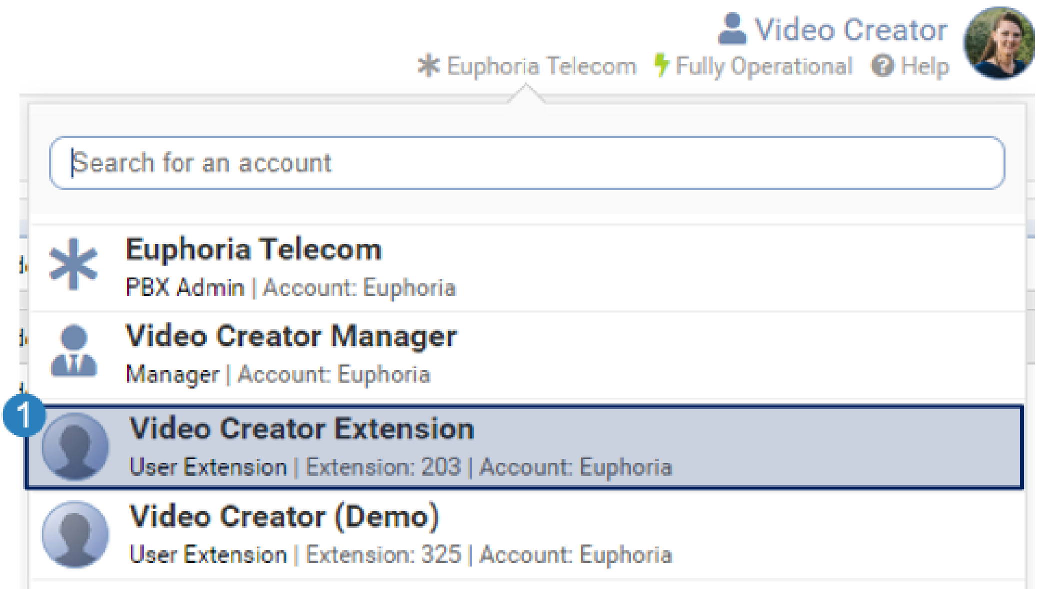


These same steps will be applied when adding webhooks to a campaign.

Webhooks In Use in Agent Workspace

From an agent's point of view, to use webhooks, the user must be logged into the TMS system as an **agent,** and be logged into their Salesforce account.

- Log into the TMS system as an **Agent**. This can be done either at the login screen or by changing the 1 *Account* type.



- Select the 2 *Agent* Menu Item, and then the 3 *Agent Workspace* Sub-menu item.

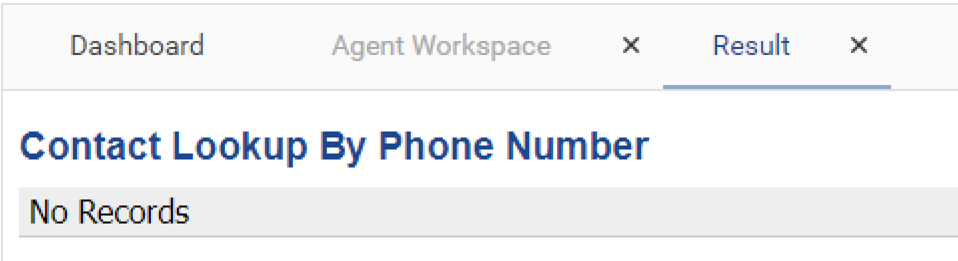
- Once logged into the workspace, with the phone idle, select the 4 dial icon to make a call.

During a call, the Webhooks will show as buttons as set up. If all 3 Webhooks have been added to the queue, three buttons should appear on the page.

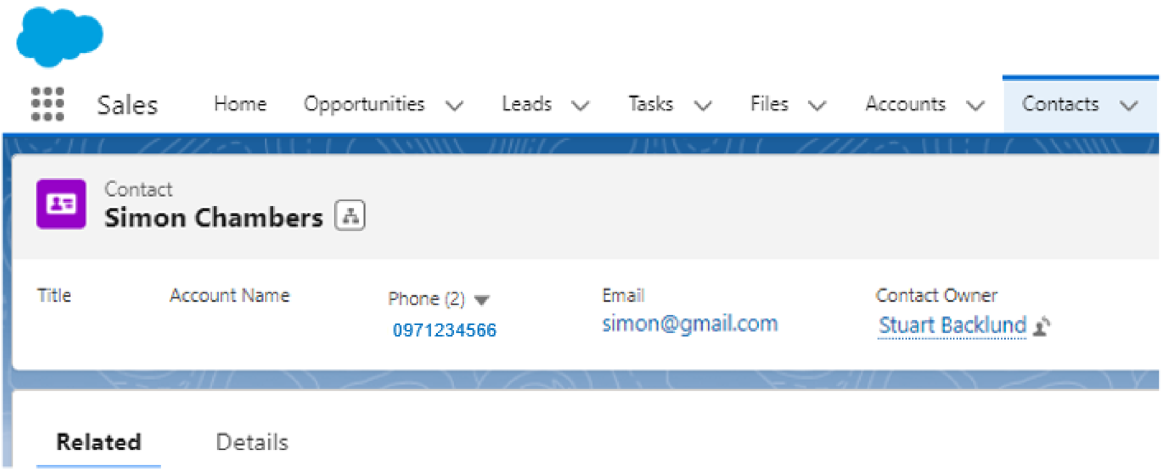
Contact Lookup *by Phone Number* Webhook (On Answer)

The purpose of this webhook is to know if the caller is a Salesforce client and has a profile.

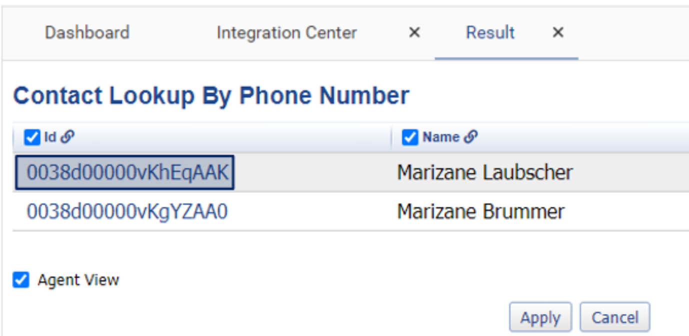
- The webhook will fire and automatically use the dialled number as a parameter in calling the webhook.. This is not a button that can be seen in the Agent Workspace. If a caller does not have a Salesforce account, a *No Records* message will show on a results page.



- If one contact is linked to the number the Salesforce Contact Page will open on a separate browser tab.



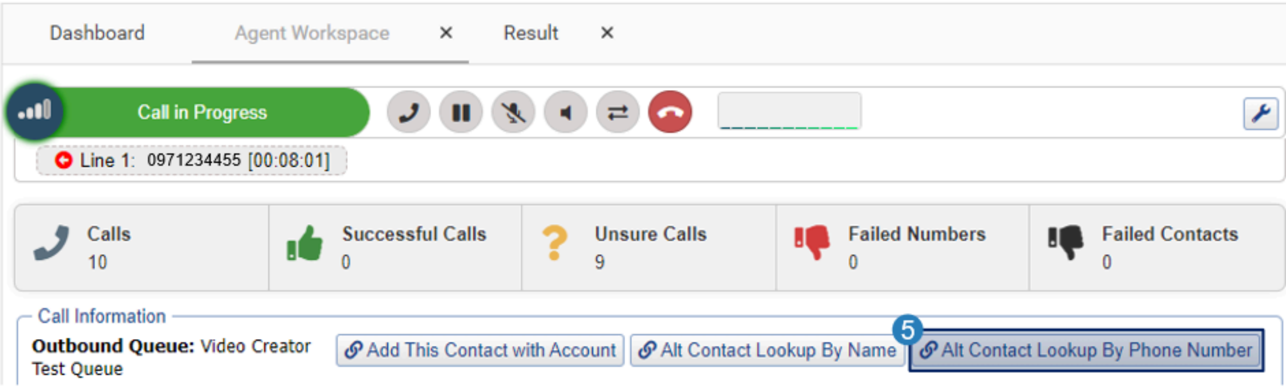
- If multiple contacts are linked to the number, the results page will open. Select a contact Id to open the contact in Salesforce.



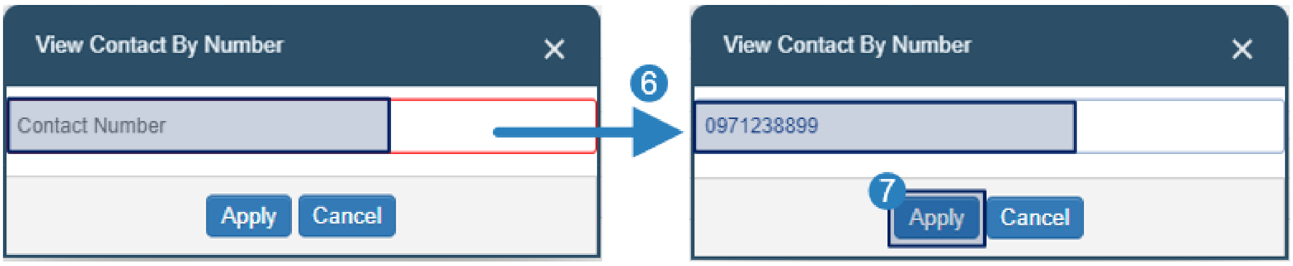
Alternative Contact Lookup by Phone Number Webhook

The purpose of this webhook is to view a contact's information who is not the caller, and see if they have a profile in Salesforce. An example of this would be when the caller asks the agent to check if their spouse if a client and if they have a Salesforce account.

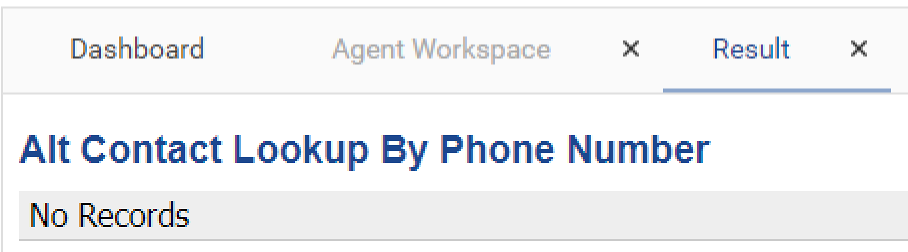
- Select the 5 *Alt Contact Lookup by Phone Number* button.



- A pop-up will request a number. Enter 6 a number and press 7 Apply.

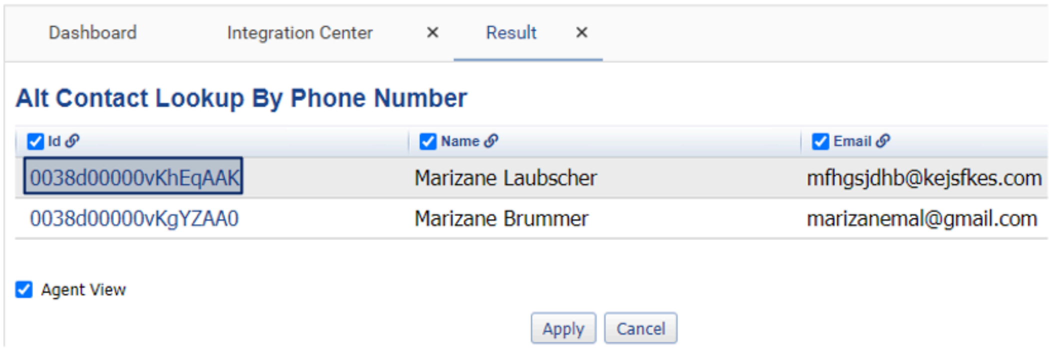


- If a caller does not have a Salesforce account, a *No Records* message will show on a results page.



- If one contact is linked to the number the Salesforce Contact Page will open on a separate browser tab.

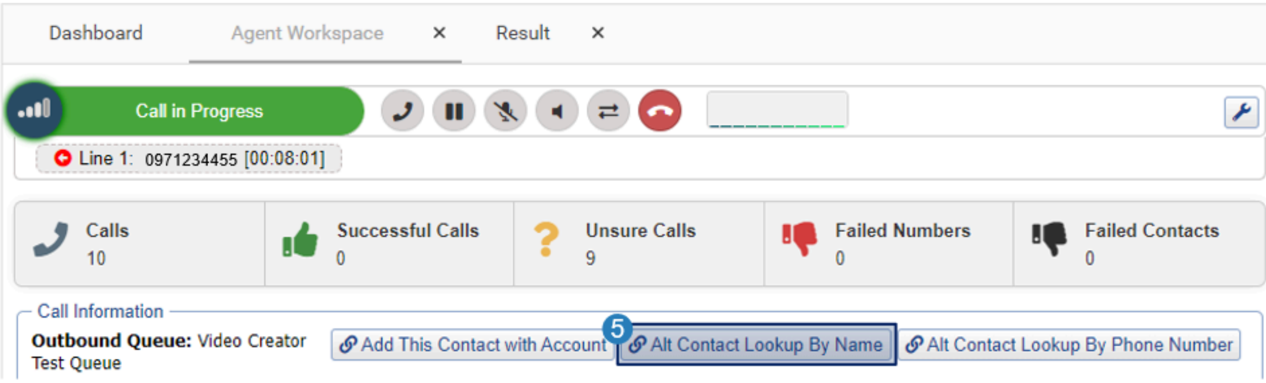
- If multiple contacts are linked to the number, the results page will open. Select a contact Id to open the contact in Salesforce



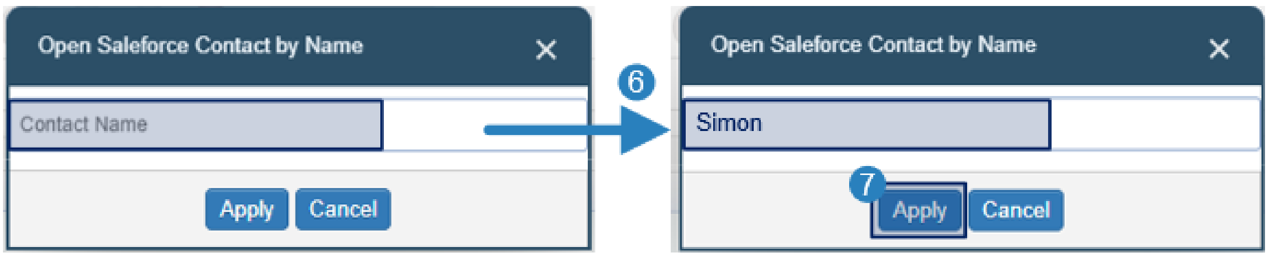
Alternative Contact Lookup by Name Webhook

The purpose of this webhook is to search for a Contact in Salesforce by their name and show them in a pop-up Salesforce page.

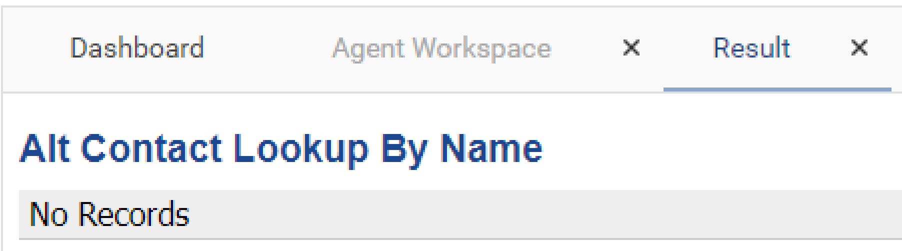
- Select the 5 *Alt Contact By Name* button.



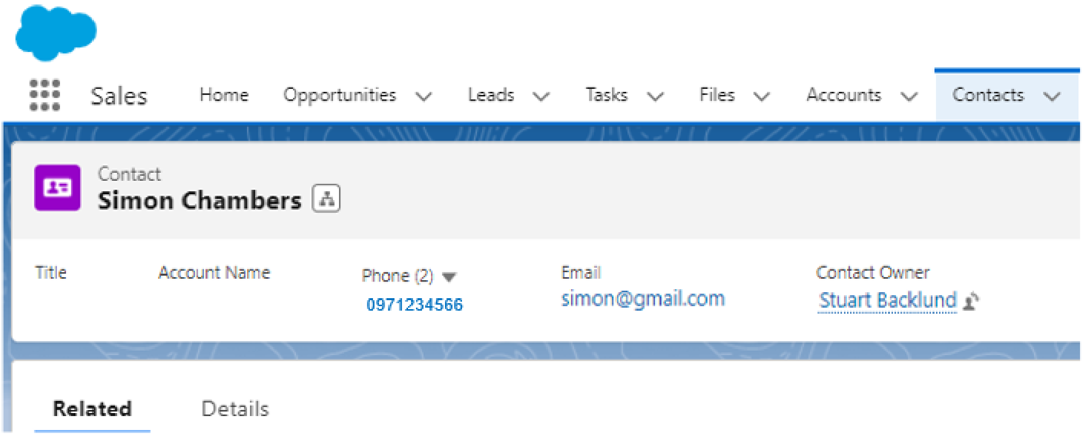
- A pop-up will request a name. Enter 6 a name and press 7 Apply.



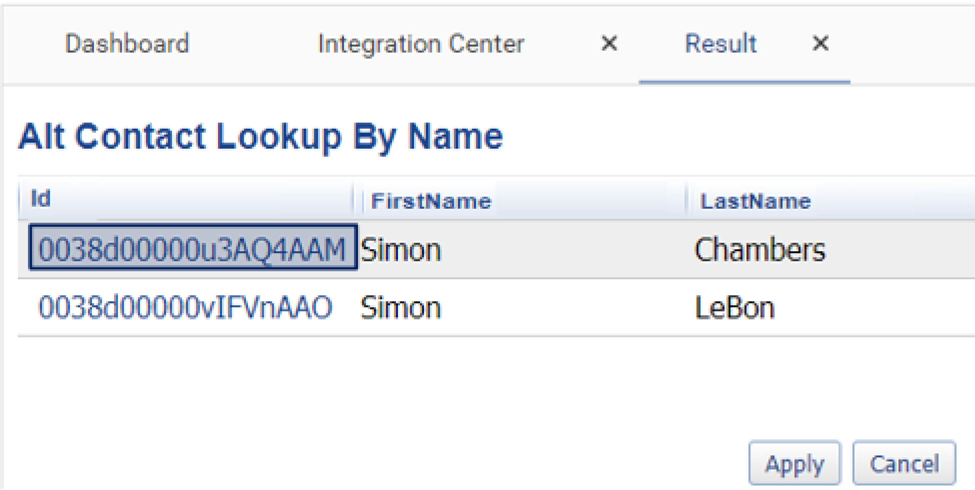
- If a caller does not have a Salesforce account, a *No Records* message will show on a results page.



- If one contact is linked to the number the Salesforce Contact Page will open on a separate browser tab.



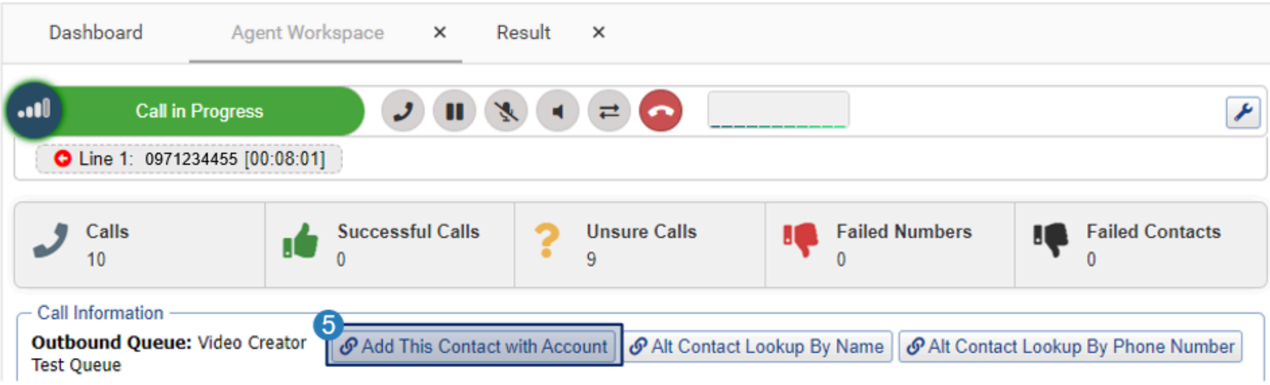
- If multiple contacts are linked to the number, the results page will open. Select a contact Id to open the contact in Salesforce



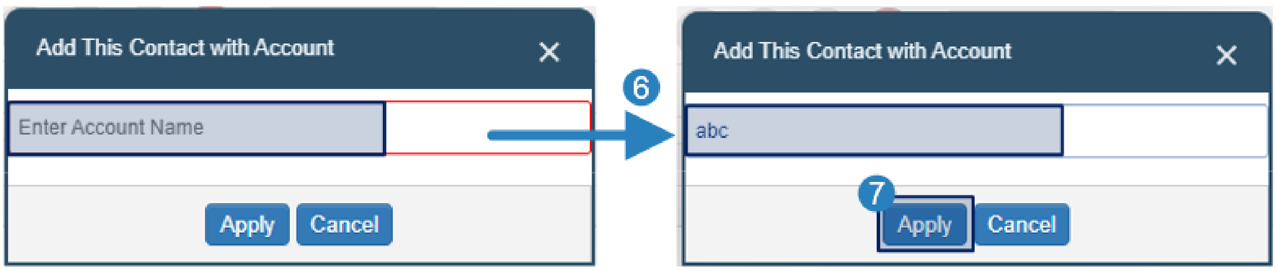
Add This Contact Webhook

These webhooks aim to add a new contact from the Agent Workspace to Salesforce.

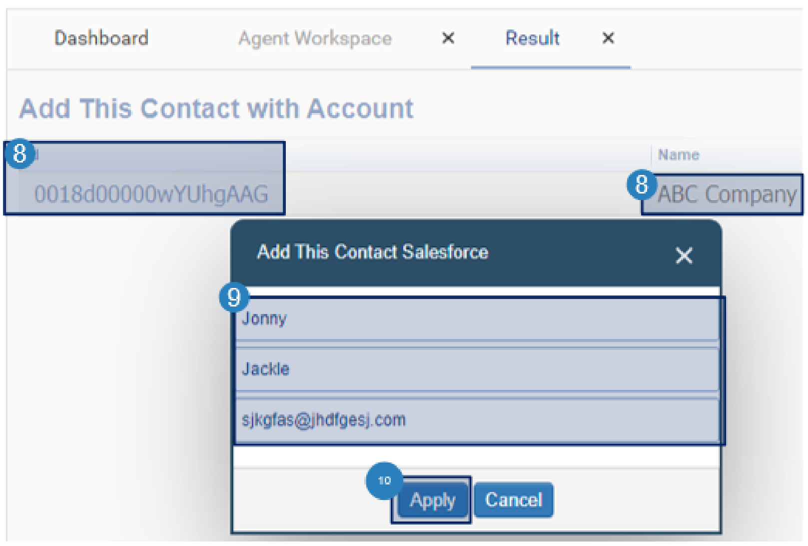
- Select the 5 *Add This Contact with Account* button.



- A pop-up will request an Account Name. Enter the 6 Company name that the contact should be associated with and press 7 Apply.

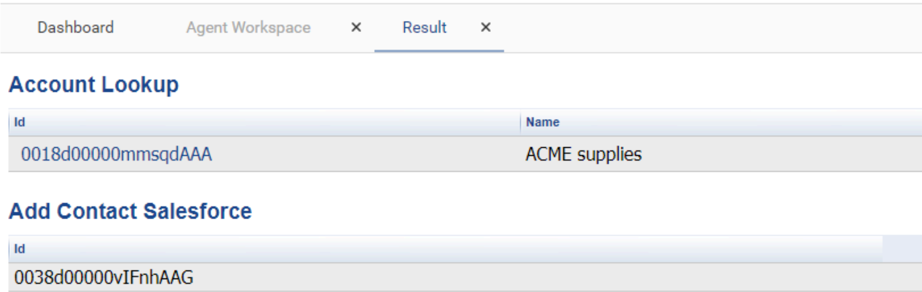


- The webhook will fire and return account information for the name entered from Salesforce. The value returned pre-populates the 8 *Account Id* field in the Add Contact webhook. The linked webhook will fire automatically and open a 9 form for the agent to complete the required fields, which are Name, Last Name and Email address. Select 10 Apply to save the information.



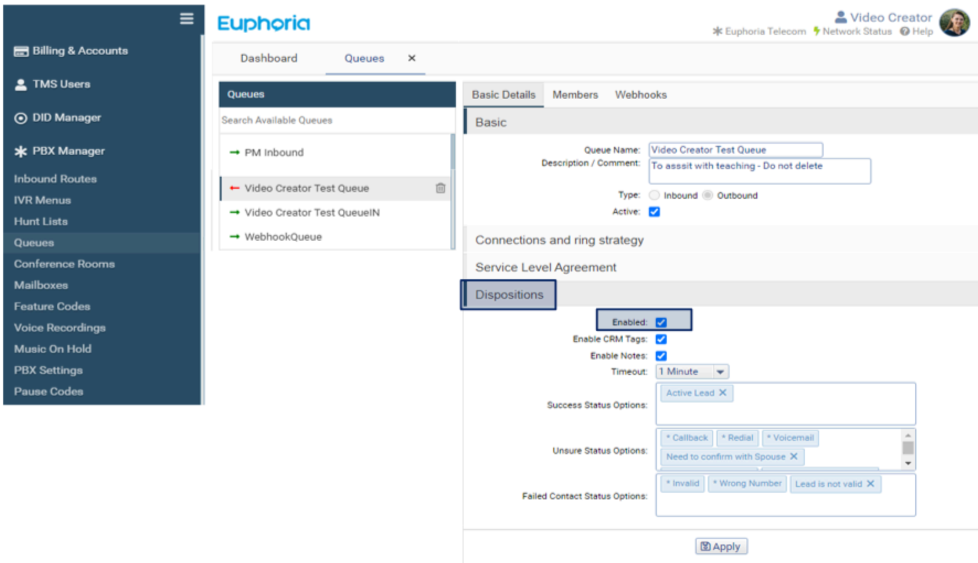
Note: The phone number and account values are pre-populated from the dialled number and account information already retrieved.

- A results tab will open And show the new Contact Id.

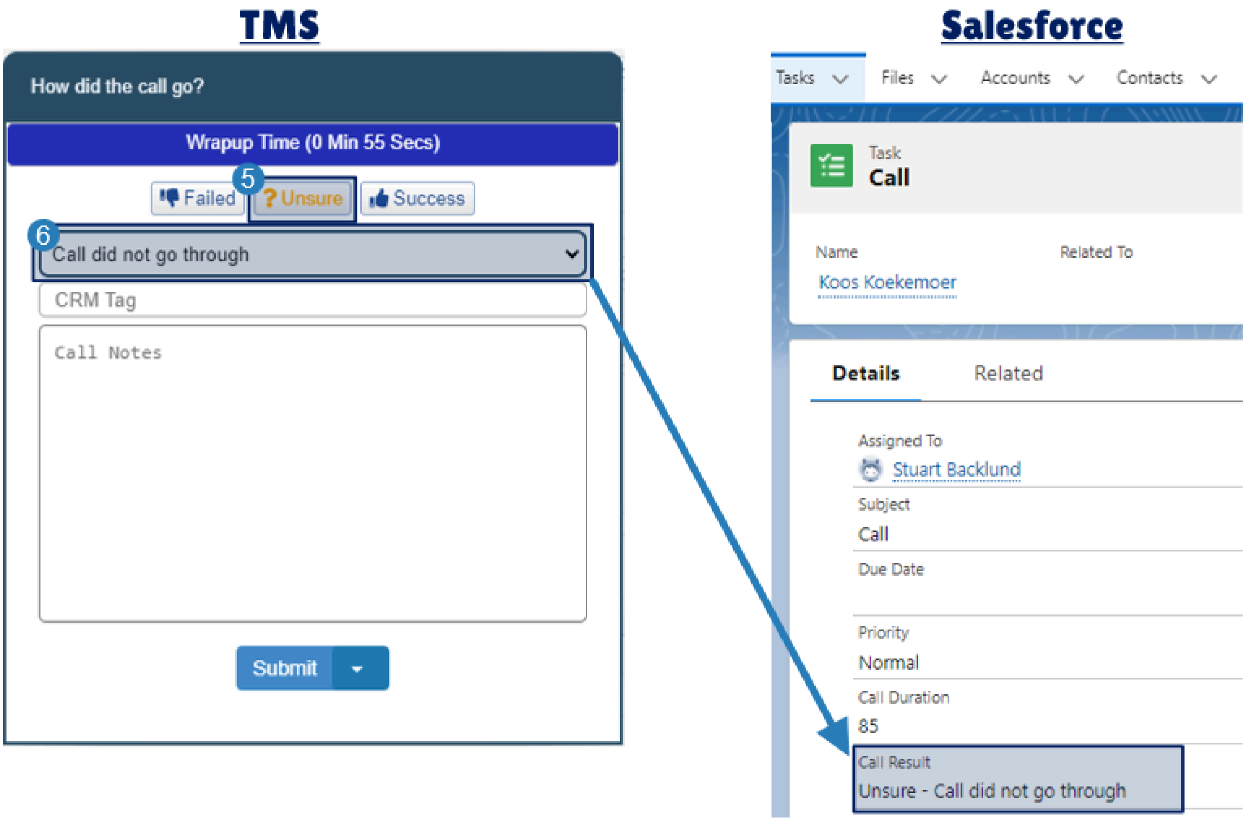


Get Contact Webhook with Activity) (On Disposition)

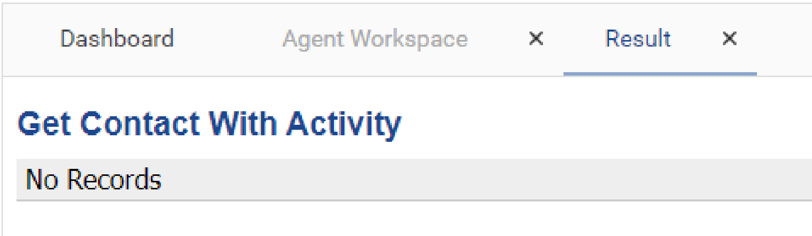
The purpose of this webhook is to add a call activity to the caller account if they have one in Salesforce at the end of the call when it is dispositioned. As this webhook is activated on disposition, it will not open if the Disposition section is not enabled on a queue. See [Feature Article on Queues] for more information.



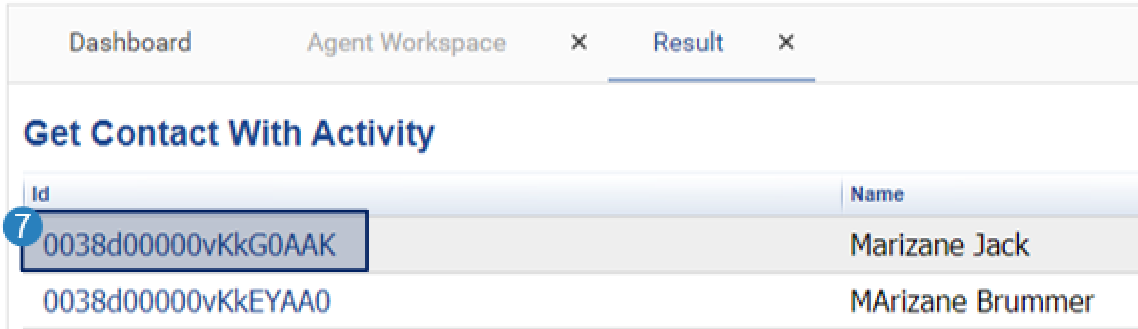
- When the call ends, select the 5 disposition and its 6 outcome. If one contact is linked to the number, the disposition will trigger the webhook to add the outcome as a task to the caller's contact account. If a disposition is not added, or timed out, the webhook will not trigger,



- If the caller was not a client in Salesforce, or added as a client to Salesforce, a results page will open with a No Records message



- If multiple contacts are linked to the number, the results page will open. Select the 7 *Contact Id* to add the disposition to their task list.



- A second result page will open, showing that the activity has been added successfully.