AppPerfect

Salesforce Connector Bot - AppPerfect Corporation

Bot Overview

- Performs **Create, Get, Update & Delete** operations (both **single & bulk** operations are supported) for Custom as well as following Standard Objects:
    - Account
    - Asset
    - Case & CaseFeed
    - Contact
    - Contract
    - Lead
    - Opportunity
    - Order
    - Pricebook
    - Product
    - Quote
    - Task

- Read & Delete the **History** of Salesforce objects.

- Run Salesforce queries to search your organization's Salesforce data for specific information.

Pre-Requisites

- Automation Anywhere Enterprise v11.3.3 or above.
- Connected App for API access to Salesforce CRM. (More details on this documented at end of the document)

Installation

- Download the Salesforce Connector Bot provided by AppPerfect Corporation from Automation Anywhere Bot Store. (https://botstore.automationanywhere.com)
- Double click on <Bot Name>.msi and follow the installation instructions below.

For first time users, the "Bot Store" folder is created under <AA Directory>/My Tasks (on your local disk).

- ***Installer creates the following folder structure with content under the <AA Directory>***

<AA Directory>
- My Tasks
    - Bot Store
        - SalesforceConnectorBot-AppPerfectCorporation
            - My tasks
                - Salesforce Connector Bot.atmx
            - Error Folder

- Log (Folder)
  - Input Error Logs Month-Day-Year.txt
- Snapshots (Folder)
  - Error Snap Month-Day-Year HourMinuteSecond.png

- Input Folder
  - Input.csv

- Output Folder
  - Output.csv

- My Metabots
  - Salesforce Connector Bot.mbot

How to Configure the Bot:

Use the following information to configure your bot parameters:

Parameter Name	**Type**	**Direction**	**Additional Info**
SalesforceCredentials(UserName)	String	Input	Provide the username for salesforce login. Create Credential Vault variable "SalesforceCredentials" which has an attribute called "UserName". https://docs.automationanywhere.com/bundle/enterprise-v11.3/page/topics/aae-developer/aae-use-crendential-valult-to-store-sensitive-data.html#Zj0vY2F0ZWdvcnkvYnVpbGQ/cD1CdWlsZA==
SalesforceCredentials(Password)	String	Input	Provide the password for salesforce login. Create Credential Vault variable "SalesforceCredentials" which has an attribute called "Password".
SalesforceCredentials(ClientId)	String	Input	Provide the client ID. Create Credential Vault variable "SalesforceCredentials" which has an attribute called "ClientId". More info: https://auth0.com/docs/connections/social/salesforce
SalesforceCredentials(ClientSecret)	String	Input	Provide the client secret. Create Credential Vault variable "SalesforceCredentials" which has an attribute called "ClientSecret". More Info: https://auth0.com/docs/connections/social/salesforce
vOutputFileName	String	Input	Provide output file path (in csv) to store obtained result set. For Eg. C:\Users\Administrator\Desktop\Result.csv
vLookupFieldName	String	Input	Provide lookup field name to perform Get / Update / Delete operations.

			For Eg. Lets say if you want to get Account object with name as AppPerfect then provide the vLookupFieldName as "Name".
vLookupFieldValue	String	Input	Provide lookup field value to perform Get / Update / Delete operations. For Eg. Lets say if you want to get Account object with account name as AppPerfect then provide the vLookupFieldValue as "AppPerfect".
vObjectJson	String	Input	Provide JSON object to perform single Create or update operation. Leave it blank in case you are doing a bulk operation. For Eg. JSON object to insert a single Account would be: {"Name":"AppPerfect", "Site" : "California"}
vInput	String	Input	Define this in case you are performing bulk operations. We support CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. Provide input file path Or you can directly provide JSON array as an input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Accounts.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Accounts.csv" In case you don't want to use input file but need to provide array of Objects to insert directly, then you can define the input as JSON array. For eg. JSON Array to insert Accounts would be : [{"Name":"AppPerfect", "Site" : "California"}, {"Name":"Automation Anywhere", "Site" : "California"}]
vOperation	String	Input	Provide operation type as described in subsequent tables.
vEnvironment	String	Input	[Optional] Provides the environment type for Salesforce. Values are added from the config file "ConfigurationFile.txt" in the Input Folder. Format for ConfigurationFile.txt: 1. For Sandbox environment: Environment = Sandbox

			2. For Production Environment: Environment = Production By default, Production environment will be used.
vResponse	String	Output	1. In case of successful Get operation. 1.a. Returns 'SUCCESS', if vOutputFileName is provided. 1.b. Returns, list of objects if vOutputFileName is empty. 2. In case of successful Insert / Update / Delete operations 2.a. Returns ID of the record which was inserted or updated or deleted in case of single operation. 2.b. Returns List of record IDs which were inserted / updated or deleted in case of bulk operation.
vErrorFolder	String	Input	This is error folder inside bot folder which contains Logs and Snapshots folder. By default Logs and Snaphosts folders will be created in this folder. If you need Logs and Snaphosts to be saved at different location then you can provide the folder location here.
vLogFolder	String	Input	This folder contains Log file in case of error. By default error logs will be created in this folder. If you need error logs to be saved at different location then you can provide the logs folder location here.
vSnapshotFolder	String	Input	This folder contains Screenshot in case of error. By default error screenshots will be saved in this folder. If you need screenshots to be saved at different location then you can provide the screenshots folder location here.
vInputFolder	String	Input	This is Input folder inside bot folder which contains Input files. By default input files are stored in this folder. If you need input files to be stored at different location then you can provide the input folder location here.
vOutputFolder	String	Input	This is Output folder inside bot folder which contains Output files. By default output files are stored in this folder. If you need output files to be stored at different location then you can provide the output folder location here.

For **Account** Operations configure following parameters:

Functions	Parameter Values
1. Insert an Account	**vOperation :** Insert Account **vObjectJson** : Provide JSON object to perform single insert operation. For Eg. JSON object to insert a single Account would be: {“Name”:"AppPerfect", “Site": “California”}
2. Update an Account	**vOperation :** Update Account **vObjectJson** : Provide JSON object to perform single update operation. For Eg. JSON object to update a single Account would be: {“Name”:"AppPerfectCorporation", “Site": “California”} **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update Account object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Update operation. For Eg. If you want to update Account object with account name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.
3. Delete an Account	**vOperation :** Delete Account **vLookupFieldName :** Provide lookup field name to perform Delete operation. For Eg. If you want to delete Account object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. If you want to delete Account object with account name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.
4. Get Accounts from Salesforce	**vOperation :** Get Account **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. If you want to get Account object with Account name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. If you want to get Account object with Account name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.

	vOutputFileName: Provide output file path (in CSV) to store obtained result set. For Eg. C:\Users\Administrator\Desktop\Result.csv
5. Insert Accounts in bulk	**vOperation :** Bulk Insert Account **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Accounts.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Accounts.csv" In case you don't want to use input file but need to provide array of Objects to insert directly, then you can define the input as JSON array. For eg. JSON Array to insert Accounts would be : [{"Name":"AppPerfect", "Site" : "California"}, {"Name":"Automation Anywhere", "Site" : "California"}]
6. Update Accounts in bulk	**vOperation :** Bulk Update Account **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update Account object with Account name as given in file or JSON array then provide the vLookupFieldName as "Name". **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Accounts.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Accounts.csv" In case you don't want to use input file but need to provide array of Objects to update directly, then you can define the input as JSON array. For Eg. JSON Array to update Accounts would be : [{"Name":"AppPerfect Corporation", "Site" : "California"}, {"Name":"Automation Anywhere", "Site" : "USA"}]

7. Delete Accounts in bulk	**vOperation :** Bulk Delete Account **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Accounts.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Accounts.csv" In case you don't want to use input file but need to provide array of Objects to delete directly, then you can define the input as JSON array. For eg. JSON Array to delete Accounts would be : [{"Name":"AppPerfect Corporation"}, {"Name":"Automation Anywhere"}]

For **Asset** Operations configure following parameters:

Functions	**Parameter Values**
1. Insert an Asset	**vOperation :** Insert Asset **vObjectJson** : Provide JSON object to perform single insert operation. For Eg. JSON object to insert a single Asset would be: {"Name":"AppPerfect", "AccountId":" 0012v00002LgZcZAAV"}
2. Update an Asset	**vOperation :** Update Asset **vObjectJson** : Provide JSON object to perform single update operation. For Eg. JSON object to update a single Asset would be: {"Name":"AppPerfect Corporation", "AccountId":" 0012v00002LgZcZAAV"} **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update Asset object with name as AppPerfect then provide the vLookupFieldName as "Name". **vLookupFieldValue** : Provide lookup field value to perform Update operation. For Eg. If you want to update Asset object with name as AppPerfect then provide the vLookupFieldValue as "AppPerfect".

3. Delete an Asset	**vOperation :** Delete Asset **vLookupFieldName :** Provide lookup field name to perform Delete operation. For Eg. If you want to delete Asset object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. If you want to delete Asset object with name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.
4. Get Assets from Salesforce	**vOperation :** Get Asset **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. If you want to get Asset objects with asset name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. If you want to get Asset object with asset name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”. **vOutputFileName:** Provide output file path (in CSV) to store obtained result set. For Eg. C:\Users\Administrator\Desktop\Result.csv
5. Insert Assets in bulk	**vOperation :** Bulk Insert Asset **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Assets.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Assets.csv” In case you don’t want to use input file but need to provide array of Objects to insert directly, then you can define the input as JSON array. For eg. JSON Array to insert Assets would be : [{“Name”:"AppPerfect", “AccountId”:" 0012v00002LgZcZAAV"},{“Name”:" Automation Anywhere", “AccountId”:" 0012v00002Liel7AAB "}]
6. Update Assets in bulk	**vOperation :** Bulk Update Assets **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update Asset object with asset

	name as given in file or JSON array then provide the vLookupFieldName as "Name". **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Assets.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Assets.csv" In case you don't want to use input file but need to provide array of Objects to update directly, then you can define the input as JSON array. For eg. JSON Array to update Assets would be : [{"Name":"AppPerfect Corporation", "AccountId":" 0012v00002LgZcZAAV"},{"Name":" Automation Anywhere", "AccountId":" 0012v00002Liel7AAB "}]
7. Delete Assets in bulk	**vOperation :** Bulk Delete Asset **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the single field name & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Assets.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Assets.csv" In case you don't want to use input file but need to provide array of Objects to delete directly, then you can define the input as JSON array. For eg. JSON Array to delete Assets would be : [{"Name":"AppPerfect Corporation"}, {"Name":"Automation Anywhere"}]

For **Case** Operations configure following parameters:

Functions	**Parameter Values**
1. Insert a Case	**vOperation :** Insert Case **vObjectJson** : Provide JSON object to perform single insert operation. For Eg. JSON object to insert a single Case would be: {"Type":"IT Profession"," Priority ":"Medium"," ContactId ":"0032v00002l0ajLAAQ"}

2. Update a Case	**vOperation :** Update Case **vObjectJson** : Provide JSON object to perform single update operation. For Eg. JSON object to update a single Case would be: {"Type":"IT Industry"," Priority ":"Medium"," ContactId ":"0032v00002l0ajLAAQ"} **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update Case object with type as IT Profession then provide the vLookupFieldName as "Type". **vLookupFieldValue** : Provide lookup field value to perform Update operation. For Eg. If you want to update Case object with type as IT Profession then provide the vLookupFieldValue as "IT Profession".
3. Delete a Case	**vOperation :** Delete Case **vLookupFieldName :** Provide lookup field name to perform Delete operation. For Eg. If you want to delete Case object with type as IT Profession then provide the vLookupFieldName as "Type". **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. If you want to delete Case with type as IT Profession then provide the vLookupFieldValue as "IT Profession".
4. Get Cases from Salesforce	**vOperation :** Get Case **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. If you want to get Case object with type as IT Profession then provide the vLookupFieldName as "Type". **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. If you want to get Case with type as IT Profession then provide the vLookupFieldValue as "IT Profession". **vOutputFileName:** Provide output file path (in CSV) to store obtained result set. For Eg. C:\Users\Administrator\Desktop\Result.csv
5. Insert Cases in bulk	**vOperation :** Bulk Insert Case

	vInput : Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Cases.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Cases.csv” In case you don’t want to use input file but need to provide array of Objects to insert directly, then you can define the input as JSON array. For eg. JSON Array to insert Cases would be : [{“Type”:”IT Profession”,” Priority “:”Medium”,” ContactId “:”0032v00002l0ajLAAQ”}, {“Type”:”Electrical”,” Priority “:”Medium”,” ContactId “:”0032v00002l0ajLAAQ”}]
6. Update Cases in bulk	**vOperation :** Bulk Update Case **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update Case object with type as IT Profession then provide the vLookupFieldName as “Type”. **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Cases.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Cases.csv” In case you don’t want to use input file but need to provide array of Objects to update directly, then you can define the input as JSON array. For eg. JSON Array to update Cases would be : [{“Type”:”IT Profession”,” Priority “:”Medium”,” ContactId “:”0032v00002l0ajLAAQ”}, {“Type”:”Electrical”,” Priority “:”Medium”,” ContactId “:”0032v00002l0ajLAAQ”}]
7. Delete Cases in bulk	**vOperation :** Bulk Delete Case **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the

	input file should be a header row which defines the single field name & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Cases.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Cases.csv” In case you don’t want to use input file but need to provide array of Objects to delete directly, then you can define the input as JSON array. For eg. JSON Array to delete Cases would be : [{“Type”:”IT Industry”}, {“Type”:”Electrical”}]

For **CaseFeed** Operations configure following parameters:

Functions	**Parameter Values**
1. Delete a CaseFeed	**vOperation :** Delete CaseFeed **vLookupFieldName :** Provide lookup field name to perform Delete operation. For Eg. If you want to delete CaseFeed object with type as CreateRecordEvent then provide the vLookupFieldName as “Type”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. If you want to delete CaseFeed object with type as CreateRecordEvent then provide the vLookupFieldValue as “CreateRecordEvent”.
2. Get CaseFeeds from Salesforce	**vOperation :** Get CaseFeed **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. If you want to get CaseFeed objects type as CreateRecordEvent then provide the vLookupFieldName as “Type”. **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. If you want to get CaseFeed objects with type as CreateRecordEvent then provide the vLookupFieldValue as “CreateRecordEvent”. **vOutputFileName:** Provide output file path (in CSV) to store obtained result set. For Eg. C:\Users\Administrator\Desktop\Result.csv
3. Delete CaseFeeds in bulk	**vOperation :** Bulk Delete CaseFeed **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the single field name & subsequent records should be the field values in CSV or Excel file.

	For Eg. If you have your input defined in CaseFeeds.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\ CaseFeeds.csv” In case you don’t want to use input file but need to provide array of Objects to delete directly, then you can define the input as JSON array. For eg. JSON Array to delete CaseFeeds would be : [{“Type”:" CreateRecordEvent"}, {“Type”:" DeleteRecordEvent"}]

For **Contact** Operations configure following parameters:

Functions	**Parameter Values**
1. Insert a Contact	**vOperation :** Insert Contact **vObjectJson** : Provide JSON object to perform single insert operation. For E.g. JSON object to insert a single Contact would be: {“LastName”:"AppPerfect",”Email”:”appperfect@gmail.com”}
2. Update a Contact	**vOperation :** Update Contact **vObjectJson** : Provide JSON object to perform single Update operation. **vLookupFieldName :** Provide lookup field name to perform Update operation. For E.g. if you want to Update Contact object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For E.g. if you want to Update Contact object with Contact name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”. For Eg. JSON object to Update a single Contact would be: {“LastName”:"AppPerfect",”Email”:”appperfect@gmail.com”, "Phone":"2211445566"}
3. Delete a Contact	**vOperation :** Delete Contact **vLookupFieldName :** Provide lookup field name to perform Delete operation. For Eg. Lets say if you want to delete Contact object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. if you want to delete Contact object with Contact name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.

	Eg: **vLookupFieldName :** Name & **vLookupFieldValue :** AppPerfect
4. Get Contacts From Salesforce	**vOperation :** Get Contact **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. if you want to Get Contact object with Contact name as AppPerfect then provide the vLookupFieldName as "Name". **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. if you want to Get Contact object with Contact name as AppPerfect then provide the vLookupFieldValue as "AppPerfect". **vOutputFileName:** Provide output file path (in csv) to store obtained result set. Ex. C:\Users\Administrator\Desktop\Result.csv
5. Insert Contacts in Bulk	**vOperation :** Bulk Insert Contact **vFilePath** : set FilePath as C:\Users\Administrator\Desktop\FieldName.xlsx Provide csv or excel file path to perform bulk operation **vInput** : Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Contacts.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Contacts.csv" In case you don't want to use input file but need to provide array of Objects to insert directly, then you can define the input as JSON array. For eg. JSON Array to insert Contacts would be : [{"LastName":"AppPerfect","Email":"appperfect@gmail.com"}]
6. Update Contacts in Bulk	**vOperation** : Bulk Update Contact **vLookupFieldName** : Provide lookup field name to perform Update operation. For Eg. If you want to update Contacts object with Contact name as given in file or JSON array then provide the vLookupFieldName as "Name". **vInput** : Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file

	formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Contacts.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Contacts.csv" In case you don't want to use input file but need to provide array of Objects to Update directly, then you can define the input as JSON array. For eg. JSON Array to update Contacts would be : [{"Name":"AppPerfect Corporation", "Site" : "California"}, {"Name":"Automation Anywhere", "Site" : "USA"}]
7. Delete Contacts in bulk	vOperation : Bulk Delete Contact vInput : Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the single field name & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Contacts.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Contacts.csv" In case you don't want to use input file but need to provide array of Objects to delete directly, then you can define the input as JSON array. For eg. JSON Array to delete Contacts would be : [{"Name":"AppPerfect Corporation"}, {"Name":"Automation Anywhere"}]

For **Opportunity** Operations configure following parameters:

Functions	**Parameter Values**
1. Insert an Opportunity	**vOperation :** Insert Opportunity **vObjectJson** : Provide JSON object to perform single insert operation. For E.g. JSON object to insert a single Opportunity would be: {"Name":"Appperfect","CloseDate":"1997-01-01","StageName":"Prospecting"}
2. Update an Opportunity	**vOperation :** Update Opportunity **vObjectJson** : Provide JSON object to perform single Update operation.

	vLookupFieldName : Provide lookup field name to perform Update operation. For E.g. if you want to Update Opportunity object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For E.g. if you want to Update Opportunity object with Opportunity name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”. For Eg. JSON object to Update a single Opportunity would be: {"Name":"Sales","CloseDate":"1997-01-01","StageName":"Prospecting" }
3. Delete an Opportunity	**vOperation :** Delete Opportunity **vLookupFieldName :** Provide lookup field name to perform Delete operation. For Eg. if you want to delete Opportunity object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. if you want to delete Opportunity object with Opportunity name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.
4. Get an Opportunity	**vOperation :** Get Opportunity **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. if you want to Get Opportunity object with Opportunity name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. if you want to Get Opportunity object with Opportunity name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”. **vOutputFileName:** Provide output file path (in csv) to store obtained result set. Ex. C:\Users\Administrator\Desktop\Result.csv
5. Insert Opportunity in Bulk	**vOperation :** Bulk Insert Opportunity **vFilePath** : set FilePath as C:\Users\Administrator\Desktop\FieldName.xlsx Provide csv or excel file path to perform bulk operation **vInput** : Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm”) file formats as input.

	In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Opportunity.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Opportunity.csv” In case you don’t want to use input file but need to provide array of Objects to insert directly, then you can define the input as JSON array. For eg. JSON Array to insert Opportunity would be : [{"Name":"Appperfect","CloseDate":"1997-01-1","StageName":"Prospecting" },{"Name":"Salesforce","CloseDate":"1997-01-01","StageName":"Prospecting" }]
6. Update Opportunity in Bulk	**vOperation** : Bulk Update Opportunity **vLookupFieldName** : Provide lookup field name to perform Update operation. For Eg. If you want to update Opportunity object with Opportunity name as given in file or JSON array then provide the vLookupFieldName as “Name”. **vInput** : Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Opportunity.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Opportunity.csv” In case you don’t want to use input file but need to provide array of Objects to Update directly, then you can define the input as JSON array. For eg. JSON Array to update Opportunity would be : [{"Name":"RPA","CloseDate":"1997-01-1","StageName":"Prospecting" },{"Name":"AutomationAnywhere","CloseDate":"1997-01-01","StageName":"Prospecting" }]
7. Bulk Delete Opportunity	vOperation : Bulk Delete Opportunity vInput : Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the single field name & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Opportunity.csv file then provide

	complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Opportunity.csv” In case you don’t want to use input file but need to provide array of Objects to delete directly, then you can define the input as JSON array. For eg. JSON Array to delete Opportunity would be : [{"Name":"RPA”},{"Name":"AutomationAnywhere"}]

For **Single Operations on Custom** Operations configure following parameters:

Functions	Parameter Values
1. Insert an Object	**vOperation :** Insert Object **vObjectJson** : Provide JSON object to perform single insert operation. **vObjectType :** Provide Salesforce object type to perform Insert operations. For E.g. vObjectType= Lead. For E.g. JSON object to insert a single Object would be: {“LastName”:"AppPerfect",”Email”:”appperfect@gmail.com”}
2. Update an Object	**vOperation :** Update Object **vObjectJson** : Provide JSON object to perform single Update operation. **vObjectType :** Provide Salesforce object type to perform Update operations. For E.g. vObjectType= Lead. **vLookupFieldName :** Provide lookup field name to perform Update operation. For E.g. if you want to Update Object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For E.g. if you want to Update Object with Object name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”. For Eg. JSON object to Update a single Object would be: {“LastName”:"AppPerfect",”Email”:”appperfect@gmail.com”, "Phone":"2211445566"}
3. Delete an Object	**vOperation :** Delete Object **vLookupFieldName :** Provide lookup field name to perform Delete operation. For Eg. Lets say if you want to delete Object with name as AppPerfect then provide the vLookupFieldName as “Name”.

	vObjectType : Provide Salesforce object type to perform Delete operations. For E.g. vObjectType= Lead. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. if you want to delete Object with Object name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.
4. Run Query in Salesforce	**vOperation :** Query Object **vQuery :** Provide any salesforce query. For Eg. select Id, Name from Account **vOutputFileName:** Provide output file path (in csv) to store obtained result set. For Eg. C:\Users\Administrator\Desktop\Result.csv

For **Bulk Operations on Custom Objects** configure following parameters:

Functions	Parameter Values
1. Insert Objects in bulk	**vOperation :** Insert Bulk Object **vObjectJson** : Provide JSON Bulk Object to perform single insert operation. **vObjectType :** Provide Salesforce Object type to perform Insert operations. For E.g. vObjectType= Lead. For E.g. JSON object to insert a single Object would be: {“LastName”:"AppPerfect",”Email”:”appperfect@gmail.com”}
2. Update Objects in bulk	**vOperation :** Update Bulk Object **vObjectJson** : Provide JSON object to perform single Update operation. **vObjectType :** Provide Salesforce object type to perform Update operations. For E.g. vObjectType= Lead. **vLookupFieldName :** Provide lookup field name to perform Update operation. For E.g. if you want to Update Object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For E.g. if you want to Update Object with Object name as AppPerfect then

	provide the vLookupFieldValue as “AppPerfect”. For Eg. JSON object to Update a single Object would be: {“LastName”:"AppPerfect",”Email”:”appperfect@gmail.com”, "Phone":"2211445566"}
3. Delete Objects in bulk	**vOperation :** Delete Bulk Object **vLookupFieldName :** Provide lookup field name to perform Delete operation. For Eg. Lets say if you want to delete Object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vObjectType :** Provide Salesforce object type to perform Delete operations. For E.g. vObjectType= Lead. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. if you want to delete Object with Object name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.
4. Get Objects from Salesforce	**vOperation :** Get Contract **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. if you want to Get Contract object with Contract name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. if you want to Get Contract object with Contract name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”. **vOutputFileName:** Provide output file path (in csv) to store obtained result set. Ex. C:\Users\Administrator\Desktop\Result.csv **vObjectType :** Provide Salesforce object type to perform Delete operations. For Eg. Lead.

For **Contract** Operations configure following parameters:

Functions	Parameter Values
1. Insert a Contract	**vOperation :** Insert Contract **vObjectJson** : Provide JSON object to perform single insert operation. For E.g. JSON object to insert a single Contract would be: {"AccountId":"0012v00002LiSIGAA3","Status":"Draft","ContractTerm":"6"}
2. Update a	**vOperation :** Update Contract

Contract	**vObjectJson** : Provide JSON object to perform single Update operation. **vLookupFieldName :** Provide lookup field name to perform Update operation. For E.g. if you want to Update Contract object with name as AppPerfect then provide the vLookupFieldName as "Name". **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For E.g. if you want to Update Contract object with Contract name as AppPerfect then provide the vLookupFieldValue as "AppPerfect". For Eg. JSON object to Update a single Contract would be: {"AccountId":"0012v00002LiSIGAA3","Status":"Activated","ContractTerm":"8"}
3. Delete a Contract	**vOperation :** Delete Contract **vLookupFieldName :** Provide lookup field name to perform Delete operation. For Eg. if you want to delete Contract object with name as AppPerfect then provide the vLookupFieldName as "Name". **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. if you want to delete Contract object with Contract name as AppPerfect then provide the vLookupFieldValue as "AppPerfect".
4. Get Contracts from Salesforce	**vOperation :** Get Contract **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. if you want to Get Contract object with Contract name as AppPerfect then provide the vLookupFieldName as "Name". **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. if you want to Get Contract object with Contract name as AppPerfect then provide the vLookupFieldValue as "AppPerfect". **vOutputFileName:** Provide output file path (in csv) to store obtained result set. Ex. C:\Users\Administrator\Desktop\Result.csv
5. Insert Contracts in Bulk	**vOperation :** Bulk Insert Contract **vFilePath** : set FilePath as C:\Users\Administrator\Desktop\FieldName.xlsx Provide csv or excel file path to perform bulk operation

	vInput : Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Contracts.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Contracts.csv" In case you don't want to use input file but need to provide array of Objects to insert directly, then you can define the input as JSON array. For eg. JSON Array to insert Contracts would be : [{"AccountId":"0012v00002LiSIGBB3","Status":"Draft","ContractTerm":"6" }, {"AccountId":"0012v00002LiSIHHA3","Status":"Draft","ContractTerm":"5" }]
6. Update Contracts in bulk	**vOperation** : Bulk Update Contract **vLookupFieldName** : Provide lookup field name to perform Update operation. For Eg. If you want to update Contracts object with Contract name as given in file or JSON array then provide the vLookupFieldName as "Name". **vInput** : Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Contracts.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Contracts.csv" In case you don't want to use input file but need to provide array of Objects to Update directly, then you can define the input as JSON array. For eg. JSON Array to update Contracts would be : [{"AccountId":"0012v00002LiSIGBB3","Status":"Activated","ContractTerm":"7" }, {"AccountId":"0012v00002LiSIHHA3","Status":"Activated","ContractTerm":"8"}]
7. Delete Contracts	vOperation : Bulk Delete Contract

in bulk	vInput : Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the single field name & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Contracts.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Contracts.csv” In case you don’t want to use input file but need to provide array of Objects to delete directly, then you can define the input as JSON array. For eg. JSON Array to delete Contracts would be : [{"AccountId":"0012v00002LiSIGBB3"}, {"AccountId":"0012v00002LiSIHHA3"}]

For **Lead** Operations configure following parameters:

Functions	Parameter Values
1. Insert a Lead	**vOperation :** Insert Lead **vObjectJson** : Provide JSON object to perform single insert operation. For E.g. JSON object to insert a single Lead would be: {"LastName":"Johny","Company":" AppPerfect " }
2. Update a Lead	**vOperation :** Update Lead **vObjectJson** : Provide JSON object to perform single Update operation. **vLookupFieldName :** Provide lookup field name to perform Update operation. For E.g. if you want to Update Lead object with name as AppPerfect then provide the vLookupFieldName as “LastName”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For E.g. if you want to Update Lead object with Leads name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”. For Eg. JSON object to Update a single Lead would be: {"LastName":"Harry","Company":"RedHat" }
3. Delete a Lead	**vOperation :** Delete Lead

	vLookupFieldName : Provide lookup field name to perform Delete operation. For Eg. if you want to delete Lead object with name as AppPerfect then provide the vLookupFieldName as “LastName”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. if you want to delete Lead object with Leads name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.
4. Get Leads from Salesforce	**vOperation :** Get Lead **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. if you want to Get Lead object with Lead name as AppPerfect then provide the vLookupFieldName as “LastName”. **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. if you want to Get Lead object with Lead name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”. **vOutputFileName:** Provide output file path (in csv) to store obtained result set. Ex. C:\Users\Administrator\Desktop\Result.csv
5. Insert Leads in Bulk	**vOperation :** Bulk Insert Lead **vFilePath** : set FilePath as C:\Users\Administrator\Desktop\FieldName.xlsx Provide csv or excel file path to perform bulk operation **vInput** : Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm”) file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Leads.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Leads.csv” In case you don’t want to use input file but need to provide array of Objects to insert directly, then you can define the input as JSON array. For eg. JSON Array to insert Leads would be : [{"LastName":"Johny","Company":" RedHat"

	},{"LastName":"keper","Company":" AppPerfectCorp " }]
6. Update Leads in Bulk	**vOperation** : Bulk Update Lead **vLookupFieldName** : Provide lookup field name to perform Update operation. For Eg. If you want to update Leads object with Lead name as given in file or JSON array then provide the vLookupFieldName as "LastName". **vInput** : Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Leads.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Leads.csv" In case you don't want to use input file but need to provide array of Objects to Update directly, then you can define the input as JSON array. For eg. JSON Array to update Leads would be : [{"LastName":"Johny","Company":" RedHat" },{"LastName":"keper","Company":" AppPerfectCorp " }]
7. Delete Leads in bulk	vOperation : Bulk Delete Lead vInput : Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the single field name & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Leads.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Leads.csv" In case you don't want to use input file but need to provide array of Objects to delete directly, then you can define the input as JSON array. For eg. JSON Array to delete Leads would be : [{"LastName":"Johny" },{"LastName":"keper" }]

For **Order** Operations configure following parameters:

Functions	Parameter Values
1. Insert an Order	**vOperation :** Insert Order **vObjectJson :** Provide JSON object to perform single insert operation. For Eg. JSON object to insert a single Order would be: {"AccountId":"AB12323132", "OwnerId" : "23AN9900","Pricebook2Id":"R23232323"}
2. Update an Order	**vOperation :** Update Order **vObjectJson :** Provide JSON object to perform single update operation. For Eg. JSON object to update a single Order would be: {"AccountId":"AB12323142", "OwnerId" : "23AN9900"," Pricebook2Id":"R23232323"} **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update Order object with OwnerId as R23232323 then provide the vLookupFieldName as "R23232323". **vLookupFieldValue** : Provide lookup field value to perform Update operation. For Eg. If you want to update Order object with Order OwnerId as R23232323 then provide the vLookupFieldValue as "R23232323".
3. Delete an Order	**vOperation :** Delete Order **vLookupFieldName :** Provide lookup field name to perform Delete operation. For Eg. If you want to delete Order object with name as AppPerfect then provide the vLookupFieldName as "Name". **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. If you want to delete Order object with Order OwnerId as R23232323 then provide the vLookupFieldValue as "R23232323".
4. Get Orders from Salesforce	**vOperation :** Get Order **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. If you want to get Order object with Order OwnerId as R23232323 then provide the vLookupFieldName as "OwnerId". **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. If you want to get Order object with Order OwnerId as R23232323 then provide the vLookupFieldValue as "R23232323".

	vOutputFileName: Provide output file path (in CSV) to store obtained result set. For Eg. C:\Users\Administrator\Desktop\Result.csv
5. Insert Orders in bulk	**vOperation :** Bulk Insert Order **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Orders.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Orders.csv" In case you don't want to use input file but need to provide array of Objects to insert directly, then you can define the input as JSON array. For eg. JSON Array to insert Orders would be : [{"AccountId":"AB12323132", "OwnerId" : "23AN9900"," Pricebook2Id":"R23232323"}, {"AccountId":"AB12323132", "OwnerId" : "23AN9900"," Pricebook2Id":"R23342323"}]
6. Update Orders in bulk	**vOperation :** Bulk Update Order **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update Order object with Order name as given in file or JSON array then provide the vLookupFieldName as "ownerId". **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Orders.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Orders.csv" In case you don't want to use input file but need to provide array of Objects to update directly, then you can define the input as JSON array. For eg. JSON Array to update Orders would be : [{"AccountId":"AB12323132", "OwnerId" : "23AN9900","

	Pricebook2Id":"R23232323"}, {"AccountId":"AB12323132", "OwnerId" : "23AN9900"," Pricebook2Id":"R23342323"}]
7. Delete Orders in bulk	**vOperation :** Bulk Delete Order **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the singlr field name & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Orders.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Orders.csv" In case you don't want to use input file but need to provide array of Objects to delete directly, then you can define the input as JSON array. For eg. JSON Array to delete Orders would be : [{"AccountId":"AB12323132"{"AccountId":"AB12323132"}]

For **PriceBook** Operations configure following parameters:

Functions	Parameter Values
1. Insert a PriceBook	**vOperation :** Insert PriceBook **vObjectJson :** Provide JSON object to perform single insert operation. For Eg. JSON object to insert a single PriceBook would be: {"Name":"AppPerfect", "Description" : "Price book for APS"}
2. Update a PriceBook	**vOperation :** Update PriceBook **vObjectJson :** Provide JSON object to perform single update operation. For Eg. JSON object to update a single PriceBook would be: {"Name":"AppPerfectCorporation", "Description" : "Price book for APS"}} **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update PriceBook object with name as AppPerfect then provide the vLookupFieldName as "Name". **vLookupFieldValue** : Provide lookup field value to perform Update operation. For Eg. If you want to update PriceBook object with PriceBook name as AppPerfect then provide the vLookupFieldValue as "AppPerfect".

3. Delete a PriceBook	**vOperation :** Delete PriceBook **vLookupFieldName :** Provide lookup field name to perform Delete operation. For Eg. If you want to delete PriceBook object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. If you want to delete PriceBook object with PriceBook name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.
4. Get PriceBooks from Salesforce	**vOperation :** Get PriceBook **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. If you want to get PriceBook object with PriceBook name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. If you want to get PriceBook object with PriceBook name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”. **vOutputFileName:** Provide output file path (in CSV) to store obtained result set. For Eg. C:\Users\Administrator\Desktop\Result.csv
5. Insert PriceBooks in bulk	**vOperation :** Bulk Insert PriceBook **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in PriceBooks.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\PriceBooks.csv” In case you don’t want to use input file but need to provide array of Objects to insert directly, then you can define the input as JSON array. For eg. JSON Array to insert PriceBooks would be : [{“Name”:"AppPerfect", “Description" : “Price book for APS”}}, {“Name”:"Automation Anywhere", “Description" : “Price book for APS”}}]
6. Update PriceBooks in bulk	**vOperation :** Bulk Update PriceBook

	vLookupFieldName : Provide lookup field name to perform Update operation. For Eg. If you want to update PriceBook object with PriceBook name as given in file or JSON array then provide the vLookupFieldName as “Name”. **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in PriceBooks.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\PriceBooks.csv” In case you don’t want to use input file but need to provide array of Objects to update directly, then you can define the input as JSON array. For eg. JSON Array to update PriceBooks would be : [{“Name”:"AppPerfect Corporation", “Description" : “Price book for APS”}}, {“Name”:"Automation Anywhere", “Description" : “Price book for APS”}}]
7. Delete PriceBooks in bulk	**vOperation :** Bulk Delete PriceBook **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the singlr field name & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in PriceBooks.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\PriceBooks.csv” In case you don’t want to use input file but need to provide array of Objects to delete directly, then you can define the input as JSON array. For eg. JSON Array to delete PriceBooks would be : [{“Name”:"AppPerfect Corporation"}, {“Name”:"Automation Anywhere"}]

For **Product** Operations configure following parameters:

Functions	Parameter Values
1. Insert a Product	**vOperation :** Insert Product **vObjectJson** : Provide JSON object to perform single insert operation. For Eg. JSON object to insert a single Product would be: {“Name”:"Digital bot -23", “defaultPrice" : “1000”}
2. Update a Product	**vOperation :** Update Product **vObjectJson** : Provide JSON object to perform single update operation. For Eg. JSON object to update a single Product would be: {“Name”:"Digital bot -23", “defaultPrice" : “1000”} **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update Product object with name as Digital bot -23 then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Update operation. For Eg. If you want to update Product object with Product name as Digital bot -23 then provide the vLookupFieldValue as “Digital bot -23”.
3. Delete a Product	**vOperation :** Delete Product **vLookupFieldName :** Provide lookup field name to perform Delete operation. For Eg. If you want to delete Product object with name as Digital bot -23 then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. If you want to delete Product object with Product name as Digital bot -23 then provide the vLookupFieldValue as “Digital bot -23”.
4. Get Products from Salesforce	**vOperation :** Get Product **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. If you want to get Product object with Product name as Digital bot -23then provide the vLookupFieldName as “Name”. **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. If you want to get Product object with Product name as Digital bot -23then provide the vLookupFieldValue as “Digital bot -23”. **vOutputFileName:** Provide output file path (in CSV) to store obtained result set.

	For Eg. C:\Users\Administrator\Desktop\Result.csv
5. Insert Products in bulk	**vOperation :** Bulk Insert Product **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Products.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Products.csv" In case you don't want to use input file but need to provide array of Objects to insert directly, then you can define the input as JSON array. For eg. JSON Array to insert Products would be : [{"Name":"Digital bot -23", "defaultPrice" : "1000"}]
6. Update Products in bulk	**vOperation :** Bulk Update Product **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update Product object with Product name as given in file or JSON array then provide the vLookupFieldName as "Name". **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Products.csv file then provide complete path of the CSV file here, like "C:\Users\Administrator\Desktop\Products.csv" In case you don't want to use input file but need to provide array of Objects to update directly, then you can define the input as JSON array. For eg. JSON Array to update Products would be : [{"Name":"Digital bot -23", "defaultPrice" : "1000"} , {"Name":"Digital bot -24", "defaultPrice" : "1000"}]
7. Delete Products in bulk	**vOperation :** Bulk Delete Product **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file

	formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the singlr field name & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Products.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Products.csv” In case you don’t want to use input file but need to provide array of Objects to delete directly, then you can define the input as JSON array. For eg. JSON Array to delete Products would be : [{“Name”:" Digital bot -23"}, {“Name”:" Digital bot -25"}]

For **Quote** Operations configure following parameters:

Functions	Parameter Values
1. Insert a Quote	**vOperation :** Insert Quote **vObjectJson** : Provide JSON object to perform single insert operation. For Eg. JSON object to insert a single Quote would be: {“Name”:"AppPerfect", “Site" : “California”}
2. Update a Quote	**vOperation :** Update Quote **vObjectJson** : Provide JSON object to perform single update operation. For Eg. JSON object to update a single Quote would be: {“Name”:"AppPerfectCorporation", “Site" : “California”} **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update Quote object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Update operation. For Eg. If you want to update Quote object with Quote name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.
3. Delete a Quote	**vOperation :** Delete Quote **vLookupFieldName :** Provide lookup field name to perform Delete operation.

	For Eg. If you want to delete Quote object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. If you want to delete Quote object with Quote name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.
4. Get Quotes from Salesforce	**vOperation :** Get Quote **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. If you want to get Quote object with Quote name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. If you want to get Quote object with Quote name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”. **vOutputFileName:** Provide output file path (in CSV) to store obtained result set. For Eg. C:\Users\Administrator\Desktop\Result.csv
5. Insert Quotes in bulk	**vOperation :** Bulk Insert Quote **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Quotes.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Quotes.csv” In case you don’t want to use input file but need to provide array of Objects to insert directly, then you can define the input as JSON array. For eg. JSON Array to insert Quotes would be : [{“Name”:"AppPerfect", “Site" : “California”}, {“Name”:"Automation Anywhere", “Site" : “California”}]
6. Update Quotes in bulk	**vOperation :** Bulk Update Quote **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update Quote object with Quote name as given in file or JSON array then provide the vLookupFieldName as “Name”. **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file

	formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Quotes.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Quotes.csv” In case you don’t want to use input file but need to provide array of Objects to update directly, then you can define the input as JSON array. For eg. JSON Array to update Quotes would be : [{“Name”:"AppPerfect Corporation", “Site" : “California”}, {“Name”:"Automation Anywhere", “Site" : “USA”}]
7. Delete Quotes in bulk	**vOperation :** Bulk Delete Quote **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the singlr field name & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Quotes.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Quotes.csv” In case you don’t want to use input file but need to provide array of Objects to delete directly, then you can define the input as JSON array. For eg. JSON Array to delete Quotes would be : [{“Name”:"AppPerfect Corporation"}, {“Name”:"Automation Anywhere"}]

For **Task** Operations configure following parameters:

Functions	**Parameter Values**
1. Insert a Task	**vOperation :** Insert Task **vObjectJson :** Provide JSON object to perform single insert operation. For Eg. JSON object to insert a single Task would be: {“Name”:"AppPerfect", “Site" : “California”}
2. Update a Task	**vOperation :** Update Task **vObjectJson :** Provide JSON object to perform single update operation. For Eg. JSON object to update a single Task would be: {“Name”:"AppPerfectCorporation", “Site" : “California”}

	vLookupFieldName : Provide lookup field name to perform Update operation. For Eg. If you want to update Task object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Update operation. For Eg. If you want to update Task object with Task name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.
3. Delete a Task	**vOperation :** Delete Task **vLookupFieldName :** Provide lookup field name to perform Delete operation. For Eg. If you want to delete Task object with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. If you want to delete Task object with Task name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”.
4. Get Tasks from Salesforce	**vOperation :** Get Task **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. If you want to get Task object with Task name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. If you want to get Task object with Task name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”. **vOutputFileName:** Provide output file path (in CSV) to store obtained result set. For Eg. C:\Users\Administrator\Desktop\Result.csv
5. Insert Tasks in bulk	**vOperation :** Bulk Insert Task **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Tasks.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Tasks.csv” In case you don’t want to use input file but need to provide array of Objects to

	insert directly, then you can define the input as JSON array. For eg. JSON Array to insert Tasks would be : [{“Name”:"AppPerfect", “Site" : “California”}, {“Name”:"Automation Anywhere", “Site" : “California”}]
6. Update Tasks in bulk	**vOperation :** Bulk Update Task **vLookupFieldName :** Provide lookup field name to perform Update operation. For Eg. If you want to update Task object with Task name as given in file or JSON array then provide the vLookupFieldName as “Name”. **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the field names & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Tasks.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Tasks.csv” In case you don’t want to use input file but need to provide array of Objects to update directly, then you can define the input as JSON array. For eg. JSON Array to update Tasks would be : [{“Name”:"AppPerfect Corporation", “Site" : “California”}, {“Name”:"Automation Anywhere", “Site" : “USA”}]
7. Delete Tasks in bulk	**vOperation :** Bulk Delete Task **vInput :** Provide input file path or you can directly provide JSON array as an input. It supports CSV, Excel ("xlsx", "xls", "xlt", "xlsm", "xltx", "xltm") file formats as input. In case of input file, provide the file path of the input file. The first row in the input file should be a header row which defines the singlr field name & subsequent records should be the field values in CSV or Excel file. For Eg. If you have your input defined in Tasks.csv file then provide complete path of the CSV file here, like “C:\Users\Administrator\Desktop\Tasks.csv” In case you don’t want to use input file but need to provide array of Objects to delete directly, then you can define the input as JSON array. For eg. JSON Array to delete Tasks would be : [{“Name”:"AppPerfect Corporation"}, {“Name”:"Automation Anywhere"}]

For **History** of any Object:

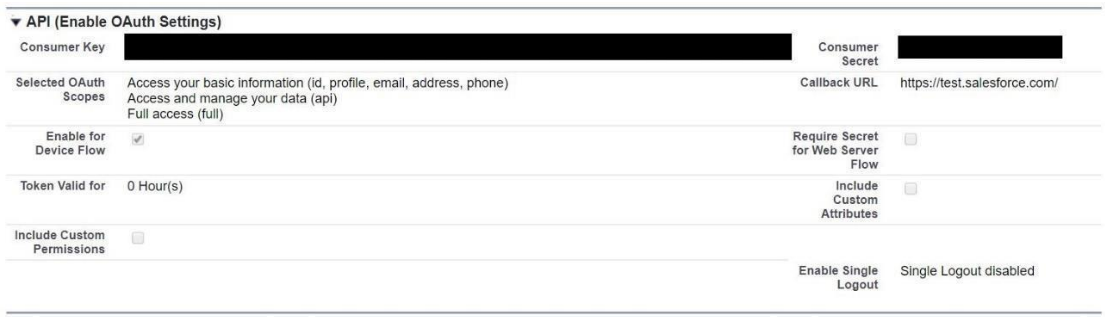
Functions	Parameter Values
1. Delete History of any object	**vOperation :** Delete History **vLookupFieldName :** Provide lookup field name to perform Delete operation on any sales force object. For Eg. If you want to delete History object for contact with name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue** : Provide lookup field value to perform Delete operation. For Eg. If you want to delete History object with History name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”. **vObjectType**: Type of the object for which history is required. For Eg. Contact
2. Get History of object from Salesforce	**vOperation :** Get History **vLookupFieldName :** Provide lookup field name to perform Get operation. For Eg. If you want to get History object with History name as AppPerfect then provide the vLookupFieldName as “Name”. **vLookupFieldValue :** Provide lookup field value to perform Get operation. For Eg. If you want to get History object with History name as AppPerfect then provide the vLookupFieldValue as “AppPerfect”. **vOutputFileName:** Provide output file path (in CSV) to store obtained result set. For Eg. C:\Users\Administrator\Desktop\Result.csv **vObjectType**: Type of the object for which history is required. For Eg. Contact

Error Handling

- Each Bot folder contains the below hierarchy.
    - Error Folder
        - Logs
            - Error Logs Month-Day-Year.txt: In case of any error, this file logs error message along with time stamp.
        - Snapshots:
            - Error Snap Month-Day-Year HourMinuteSecond.png: In case of any error, this file captures screenshot of error with time stamp.
- Task Status of bot is set to failed in case of error.

Steps to setup a Connected App on Salesforce CRM:

- Login to Salesforce CRM, navigate to Sales Admin (Username) > Setup.
- On the left pane under App Setup navigate to Create > Apps.
- Click on Connected App and you can view your Consumer Key and Consumer Secret on app information page. Consumer Key and Consumer Secret corresponds to ClientId and ClientSecret attributes of Credential Vault.



▼ API (Enable OAuth Settings)

Consumer Key	[redacted]	Consumer Secret	[redacted]
Selected OAuth Scopes	Access your basic information (id, profile, email, address, phone) Access and manage your data (api) Full access (full)	Callback URL	https://test.salesforce.com/
Enable for Device Flow	✓	Require Secret for Web Server Flow	☐
Token Valid for	0 Hour(s)	Include Custom Attributes	☐
Include Custom Permissions	☐		
		Enable Single Logout	Single Logout disabled

- For More details on Connected App please refer: https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/intro_defining_remote_access_applications.htm
- Add suitable IP Relaxation. For removing all restrictions navigate to manage -> Edit Policies. In IP Relaxation under OAuth policies select **Relax IP restrictions**.
- For details on type of IP Relaxation please refer: https://help.salesforce.com/articleView?id=connected_app_continuous_ip.htm&type=5

Important points to consider:

- It is possible for user having admin privileges to read and save the privileged files (open and write file functions), so user of the bot should not have admin access.

- Credential Vault uses multiple encryptions to store sensitive information (usernames/passwords /ClientID/ ...). These variables are used for various purposes in task bots. In response to a potential leak or compromise, Credential Vault credentials must be changed/rotated periodically

- https://docs.automationanywhere.com/bundle/enterprise-v11.3/page/topics/aae-developer/aae-use-crendential-valult-to-store-sensitive-data.html#Zj0vY2F0ZWdvcnkvYnVpbGQ/cD1CdWlsZA==

- User needs to enable History Tracking for the object to be able to track the history. User can enable history tracking at Object Manager > [Object Name] > Fields and Relationships > Set History Tracking > Enable [Object] History > [Select the fields user needs to track] > Save.

- To be able to delete the History of objects, follow these steps:

  - Click the Gear icon and select Setup.
  - Enter User Interface in the Quick Find box and select User Interface.
  - Under the Setup heading, select the "Delete from Field History" and "Delete from Field History Archive" checkbox.
  - Click Save.
  - After enabling the permissions above, grant your users the system permissions below via permission set or custom profile. Enable 'Delete From Field History'. Enable 'Delete From Field History Archive'.

- For information regarding how to input your Access Code, please refer the following link- https://botstore.automationanywhere.com/inputting-your-access-code/

Bot Insight Details
To know how to use Automation Anywhere Bot Insight to track bot process data from for analytic analysis, please refer: https://docs.automationanywhere.com/bundle/enterprise-v11.3/page/topics/bot-insight/user/bot-insight-introduction.html

Troubleshooting & Support

Please visit our Support Portal for any assistance on Bot functionality or Feature.

Automation Anywhere provides a Product Documentation portal that can be accessed for more information about AA's products and guidance on building bots and Digital Workers.

The "Build" section of the portal includes these sections:

- Getting Started - information on building bots recommended practices (including use of the Credential Vault)
- Build Advanced Bots - details on MetaBots and the approach to integrating code into them
- Build Digital Workers - high-level architecture