

---

## Vlocity

---

**Vlocity:** Vlocity is a leading industry-specific cloud and mobile software provider  
It is a managed package on top of the salesforce cloud.

Application / industry specific  
use cases

Loosely couple data model  
Cloud (Sales/Service) / Flow

Omni studio  
Flex card (More user centric  
flow) / Omni script  
Data Raptor/Integration

Salesforce Core platform  
Lightning component  
Apex/ Trigger / Bulk  
Integration

---

**Application:**  
**Media and entertainment sector :**

**Vlocity Clickstream Analytics:**

Vlocity Clickstream Analytics provides actionable insight into customer-facing business processes, enabling executives and analysts with metrics on the timing, integration, and outcomes of sales and services interactions).

**Communication sector :**

Vlocity Communications includes Sales, Marketing, Service, Retail, EPC, CPQ, Contract and Order Management applications.

**Energy sector :**

**Health sector**

Vlocity Health Insurance builds on the flexibility of the Salesforce Platform to drive the entire health insurance customer journey. Cloud apps that are enterprise-scale solutions designed to run and replace the Front and Mid Office for health plans

**Insurance sector :**

Velocity's omni channel solutions for the entire customer journey, from quote to service and claims).

**Government sector:**

(Vlocity Public Sector delivers modern applications for government contact centers, case management, and the administration of housing and human services programs that are 100% additive to Salesforce Service Cloud and Community Cloud

**Utilities :**

**Component:** (this component bundle call as omni studio(declarative approaches))

===== > Home/record/ lightning app builder / community <=====

Vlocity card / flex card / vlocity flex card (**front-end**)(only to show some data)

OmniScript (back-end) (**Business flow**)

Vlocity integration procedures (**data-load**) (**Apex class**) (**Complex logic**)

DataRaptors (**data-load**) (**simple logic**) (**lightning data services**)

---

Vlocity calculation matrix

Vlocity calculation procedure

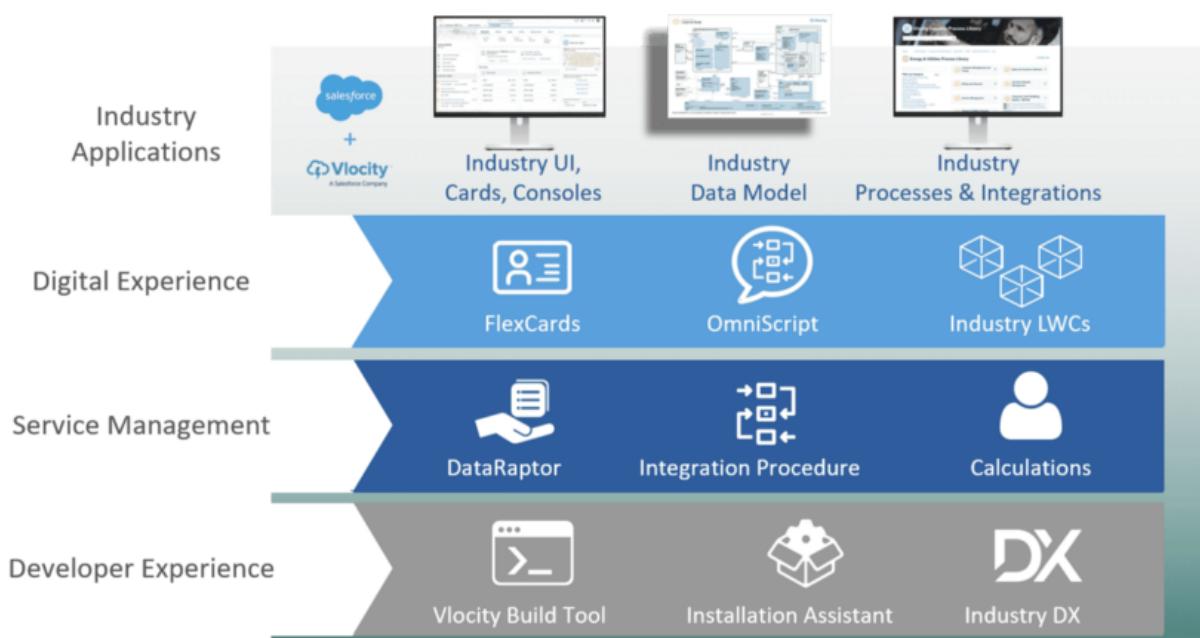
Industry console

Process engine

ClickStream Analytics

IDX Build tool / IDX Workbench

---



## **Detail Description:**

**Vlocity template: How data will be displayed {i mean one common format to show result}**

**Vlocity layout::: hold multiple cards in one layout.**

**Velocity card :: mobile/Desktop/ IPAD page look, preview (FLEX CARD) (Output)**

**Omnistudio Flex card : more interactive ui**

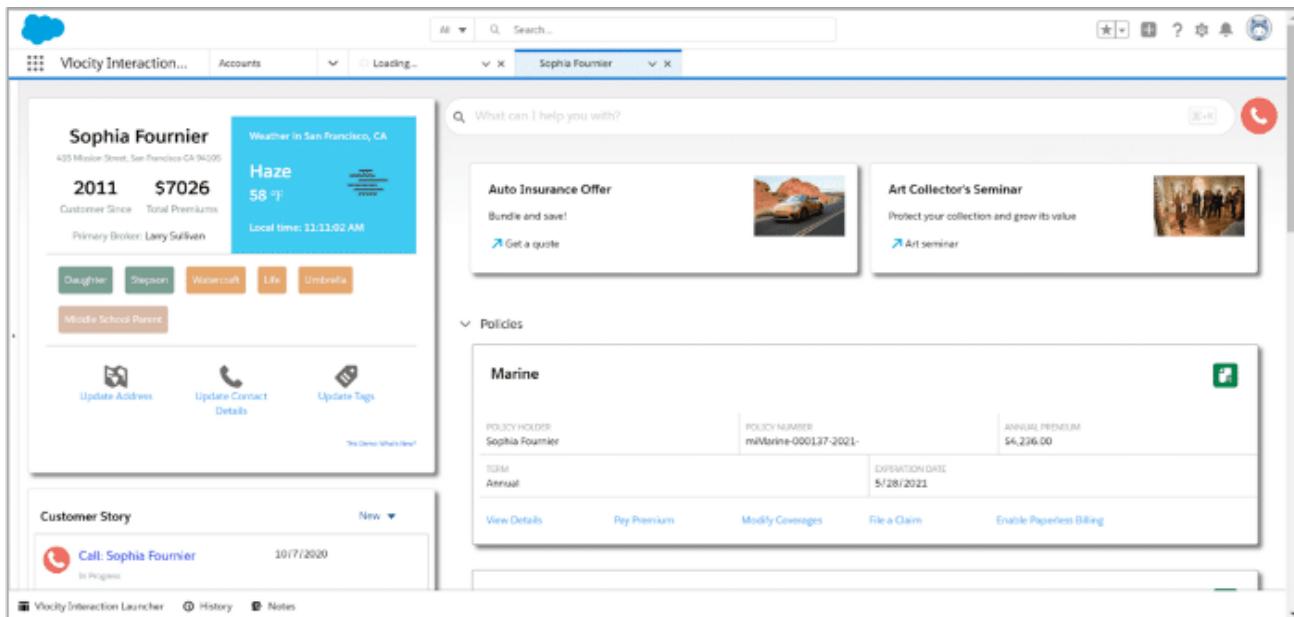
**FLYOUT : kind of pop up which will show more information when we click action item**

**Pass data from parent card to child card {records} {Account:Name}**

**Flex card : view the data in card format (u can filter incoming data here also state wise so what action need to be taken for particular data information can be done )**

State : what are the information u wanna see in the flex card

Action : what are the action u wanna see with the data state information



**If we want to pass any input from flex card**

**Then input Map**

**Id {{param.Id}}**

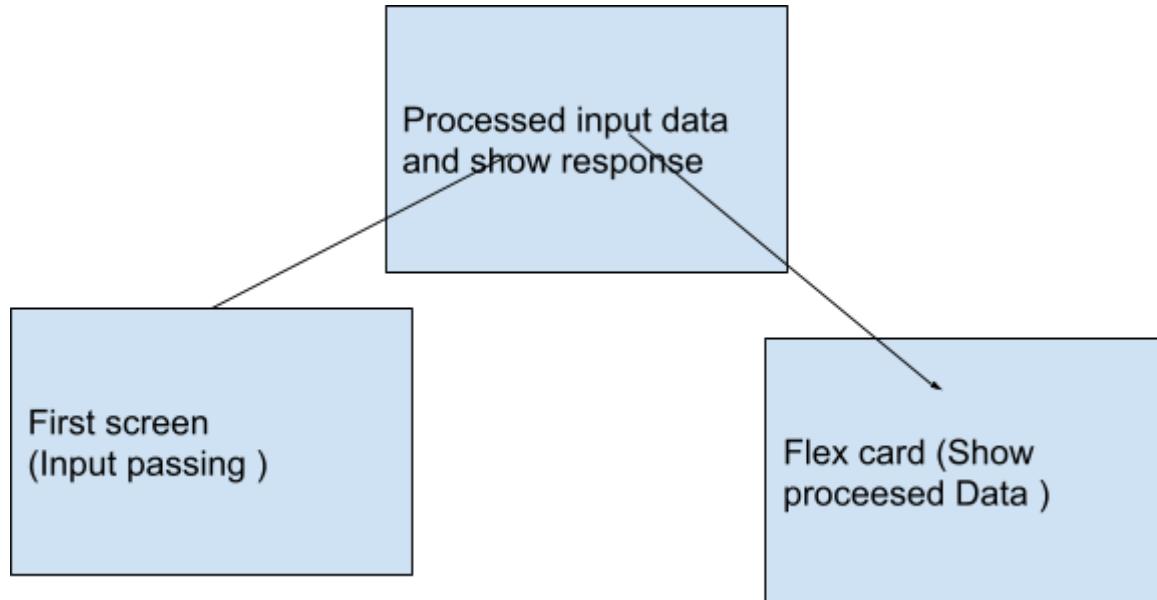
**TestVariable**

## Param.id

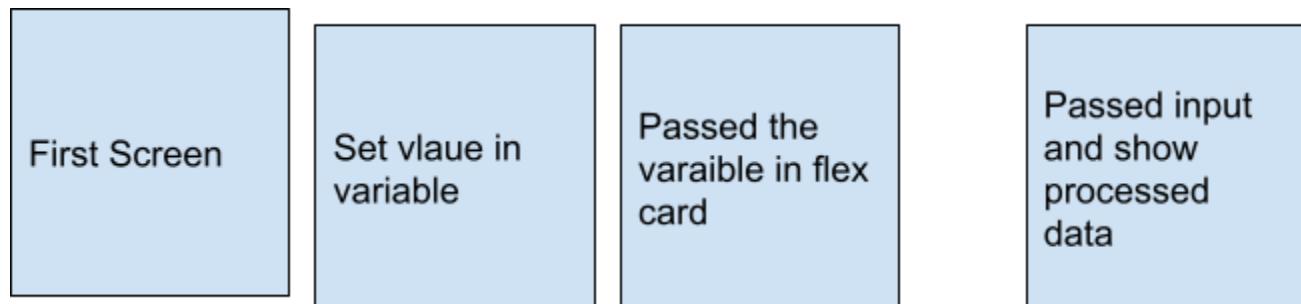
Vlocity flex card is a kind of separate output sourcing so you need to define source by passing the input parameter.

It doesn't work in the process where we get response from one screen and show the output in flex card

E.g:



No it doesn't work in this way as i said while loading flex card it has separated outsourcing  
So



## **OmniScript :: similar to flow :: more enhanced customer context centric experience**

Support condition business logic, exception flow, document integration and customer branding etc

Series of action users need to perform to complete a business logic.

### **Example:**

PRE POPULATED FIELD(Extract data raptors from salesforce))

Load data while searching (Integration procedure) (GOOGLE SEARCH) (**Input**)

---

### **Mode**

Design : Build / Property / setup

Preview : Debug

Width Adjustment : Automatically

Preview device mode : mobile / laptop/ tablet

Version / Activation

---

### **Build Element :**

- 1) Action :: (Email, PDF, Remote, HTTP)
  - 2) Display :: output (image, Screen )
  - 3) Function :: Aggregate, formula, conditional message, geolocation
  - 4) Group :: Grouping items (Step, radio group, block, action block)
  - 5) Input :: All input elements
  - 6) Omniscript :: reusable omniscrypt
- 

**Name :** API

**Label:** Field Label

### **Field Property :**

Field dependency : condition : if: true

Hide of field in JSON format

Required / help text / Validation

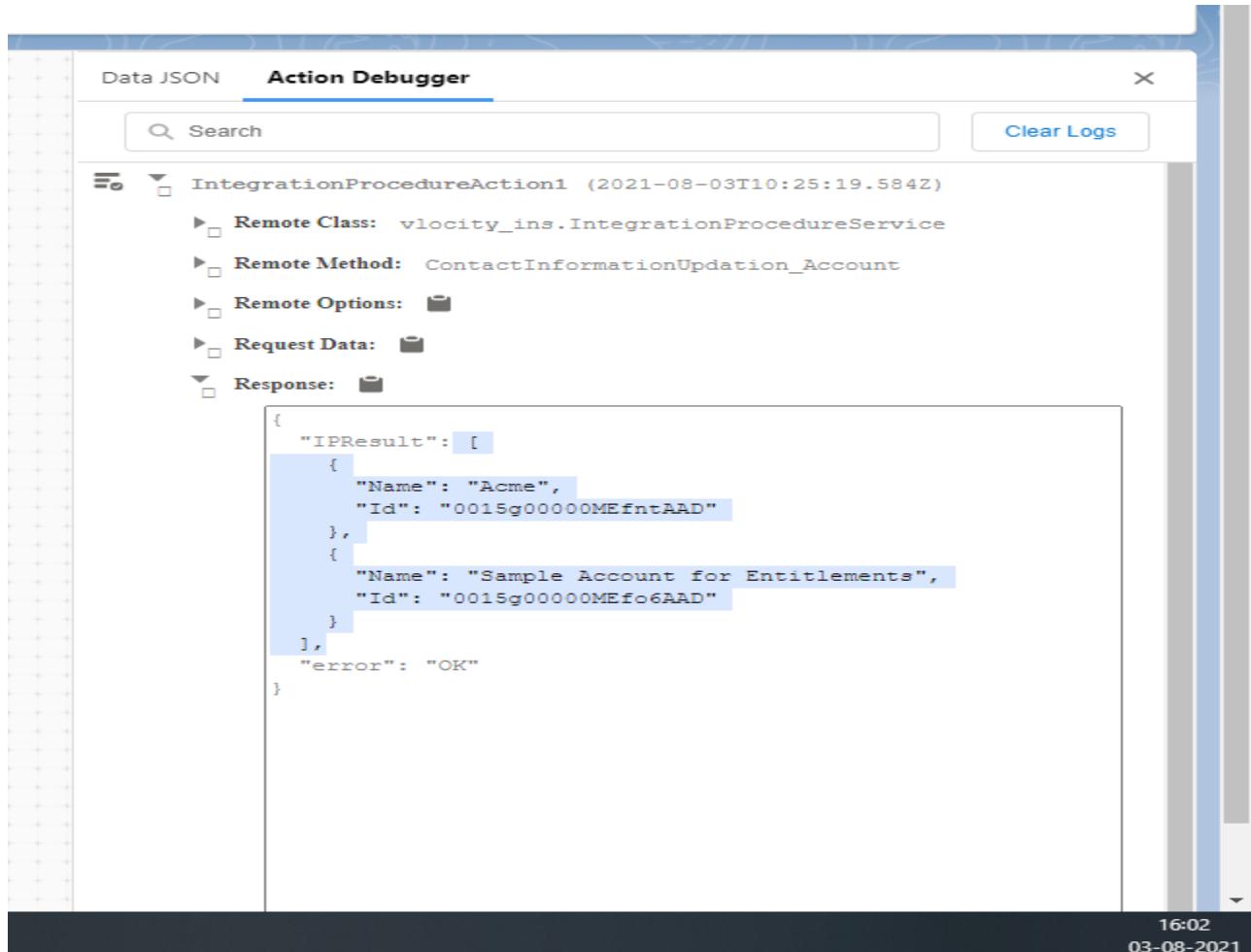
Restriction

### **Grouping:**

Steps : Screen : step chart : in setup

**TypeHead:** Searching : (Type and searching )

In response node data should be list<object> format



The screenshot shows the "Action Debugger" tab of a tool interface. At the top, there is a "Search" input field and a "Clear Logs" button. Below the header, a tree view displays the structure of the response. The "Response" section is expanded, showing the following JSON data:

```
{
  "IPResult": [
    {
      "Name": "Acme",
      "Id": "0015g00000MEfntAAD"
    },
    {
      "Name": "Sample Account for Entitlements",
      "Id": "0015g00000MEfo6AAD"
    }
  ],
  "error": "OK"
}
```

The JSON structure indicates a list of accounts, each with a name and ID. The "error" field is set to "OK". The timestamp at the bottom right of the window is 16:02 on 03-08-2021.

Key that u gonna use to : in this case Just **Name**

In IP input: just type TypeHead1 as input

**TYPE AHEAD BLOCK PROPERTIES**

Name: InsurancePolicy  
Field Label: Insurance Policy  
Required:  Read only:   
Minimum Length: 0  
Maximum Length: 255  
Typeahead Key: insuranceName

**DATARAPTOR EXTRACT ACTION PROPERTIES**

Name: DR\_Insurance\_ExtractAction1  
Field Label: DR\_Insurance\_Extract  
DataRaptor Interface: BE\_Phi\_DRT\_Extract\_Insurance  
Create New DataRaptor:  Ignore Cache:   
Input Parameters: Data Source: InsurancePolicy  
Filter Value: paramInsuranceId

**OMNISTUDIO DATARAPTOR BE\_PHI\_DRT\_Extract\_Insurance**

Interface Type	Input Type	Output Type	Required Permission (Optional)	Description
Extract	JSON	JSON		Extract Insurance policies from salesforce

**EXTRACT FORMULAS OUTPUT OPTIONS PREVIEW**

**1 - InsurancePolicy**

\* Extract Output Path: InsurancePolicy  
Filter: Name:  LIKE:  Filter Value: paramInsuranceId

**OMNISTUDIO DATARAPTOR BE\_PHI\_DRT\_Extract\_Insurance**

Interface Type	Input Type	Output Type	Required Permission (Optional)	Description
Extract	JSON	JSON		Extract Insurance policies from salesforce

**Input:****Field:**

Picklist value : objectname.fieldName

Lookup field: for lookup field user can not type anything they can only select available value for that u need to pass the value

So 2 option :-

- 1) **Provide default value or**
- 2) **Provide lookup filter value that value can come from dataraptor and assigned through type head or normal text earlier (set value and pass in filter )**

**Action:**

Set value

Set Error

HTTP action: call HTTP API

Remote action : execution of Apex class

PDF Action

Email Action

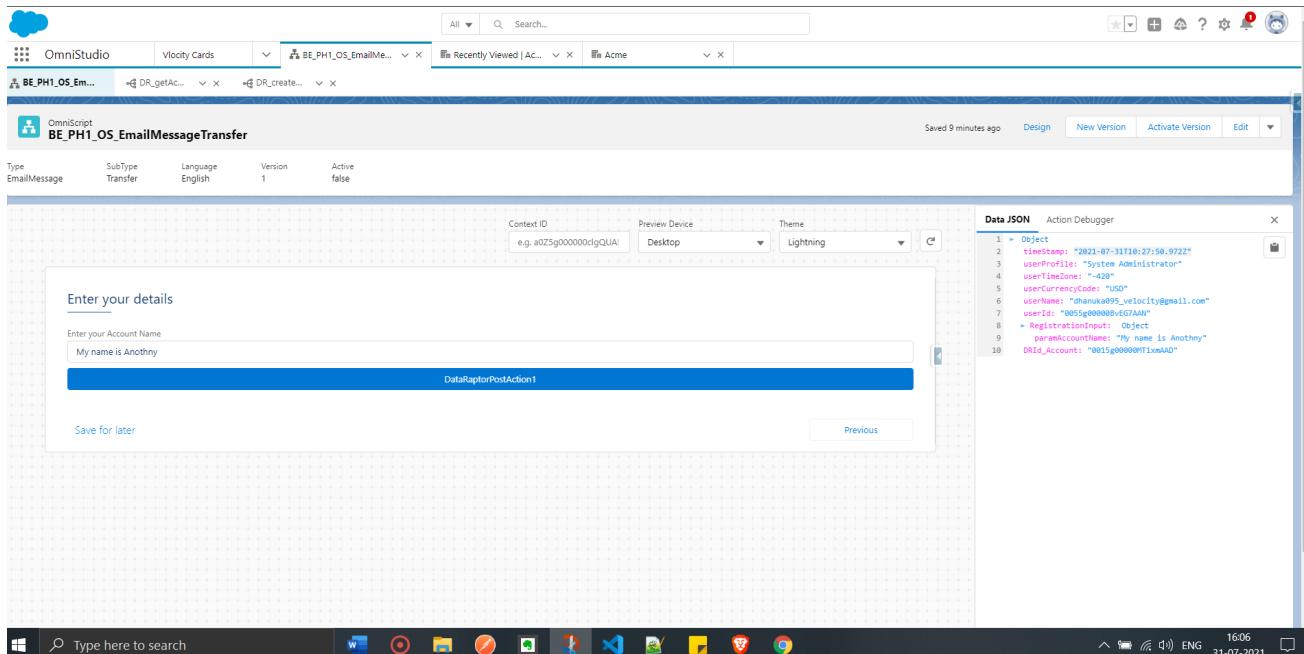
Navigate Action: where to direct/Redirect(dataraptor name (i guess it contains id) or ip response id)

Refresh/ Reload

---

**Example:**

**User will give Account number if account number present then fetch account and contact details else show error**



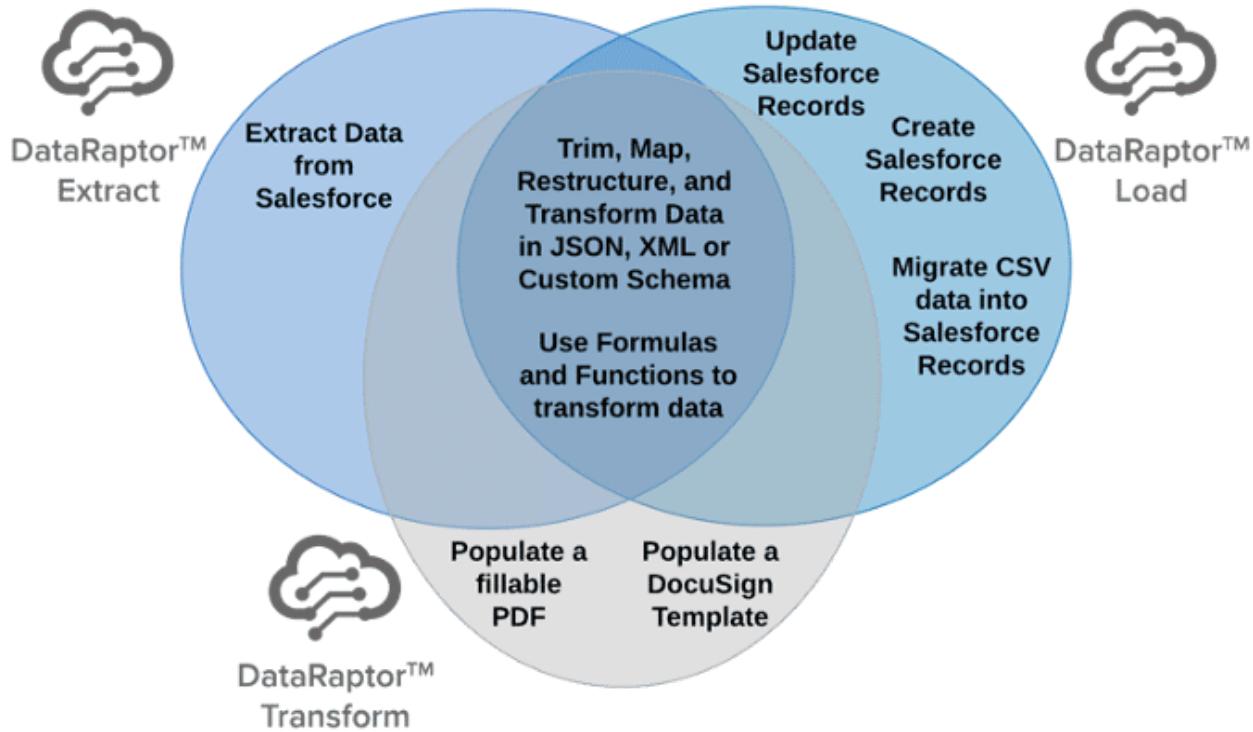
**DataRaptors** :: is one of declarative tool :: that is used to supply data to omniscrypts and omni flex card and write data from omniscrypt and flexcard back to salesforce  
**(IMPORT-EXPORT Functionality available)**

Note: it Run synchronously

#### Type of DataRaptors:

- 1) Data Raptor Turbo Extract : get data from single object (**single**)
- 2) Data Raptor Extract : get data from one or more salesforce object (**parent, child**)
- 3) Data Raptor Load : save the data in one or more salesforce object  
**(Updating/creating/Migrating)**
- 4) Data Raptor Transform : manipulate data (Transform xml to JSON)(JSON-JSON)(JSON-PDF)

Ex: pdf generation and data show



### Extract: for multiple related query use : feature to map in right way

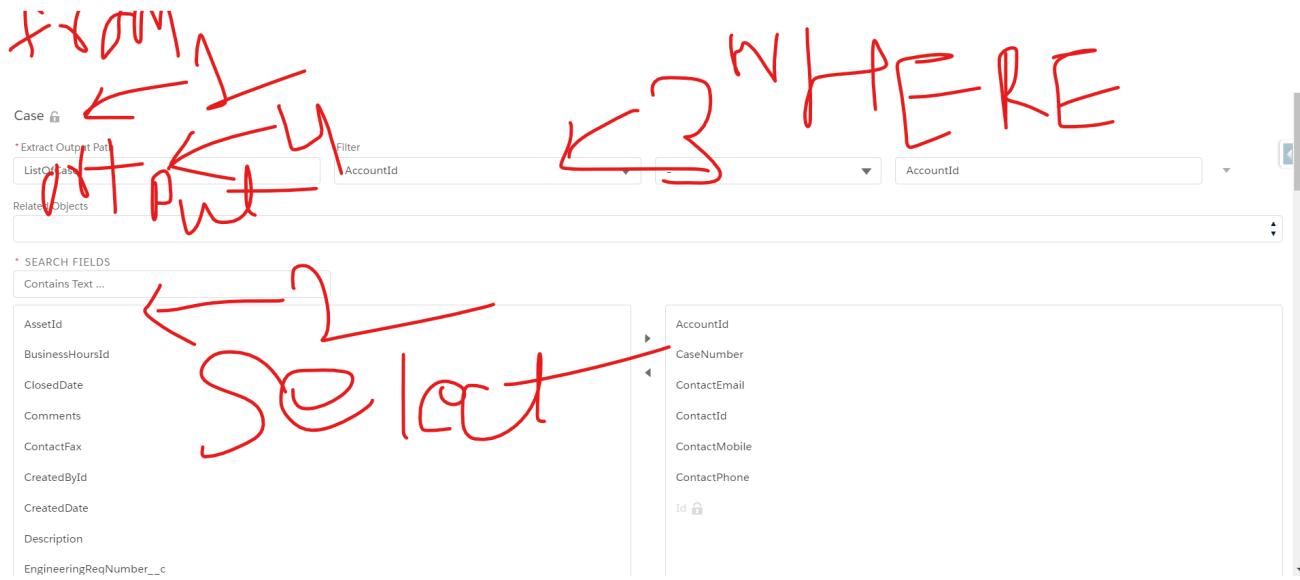
And keep **extract json path** and **output json path** same (OR u can use **quick match** option)  
 And to transform the data either use **formula field** or **transform key value pair** option

### Example: (Turbo Extract)

```
public class GetCaseInformation {
```

```
    public static List<Case> getCaseInformation(String AccountId){
        //return [select id, CaseNumber from Case where Account.Id =
        '0015g00000MEfntAAD']
        List<Case> li = [select id, CaseNumber from Case where Account.Id =: AccountId];
        System.debug('List of case' + li);
        return li;
    }
```

}



5:) you can also perform **limit**, **order by**, **offset** function and **AND**, **OR** query

**Load Data:** (u can utilize valid json for quick match )

Single Object single record

Single object multi record

Multiple object single record

Multiple Object multi record

The screenshot shows the OmniStudio LoadData interface. In the 'LoadData' tab, under the 'FIELDS' section, a new Case object is being created. It has two fields: 'AccountId' (linked to '1 - Account') and 'Status' (linked to '1 - Account'). The 'Active\_\_c' field is also present. Below the fields, there are buttons for '+ Add Link' and '+ Add Object'. The 'PREVIEW' tab is selected, showing the input JSON and the objects created:

```

Input:
{
  "paramName": "Shubham",
  "paramNumber": "123"
}

Objects Created:
• 1- Case: 5005g00000AJDBAA5
• 2- Account: 0015g00000MT1JUAT

```

The 'Errors/Debug Output' panel shows a 'Debug Log' entry for the Case creation.

**is lookup for finding relevant data and then updating the data**

**Transform data:** to get index value use **formula** and **pipe** option and index **start from 1**

Remember formula store value use as input in input json extract

1. Object to List Format: <https://github.com/TheVishnuKumar/Omn...>

2. Object List to List of Items:

<https://github.com/TheVishnuKumar/Omn...>

3. Object List to Multi-Select Value: <https://github.com/TheVishnuKumar/Omn...>

4. Data Value Conversion: <https://github.com/TheVishnuKumar/Omn...>

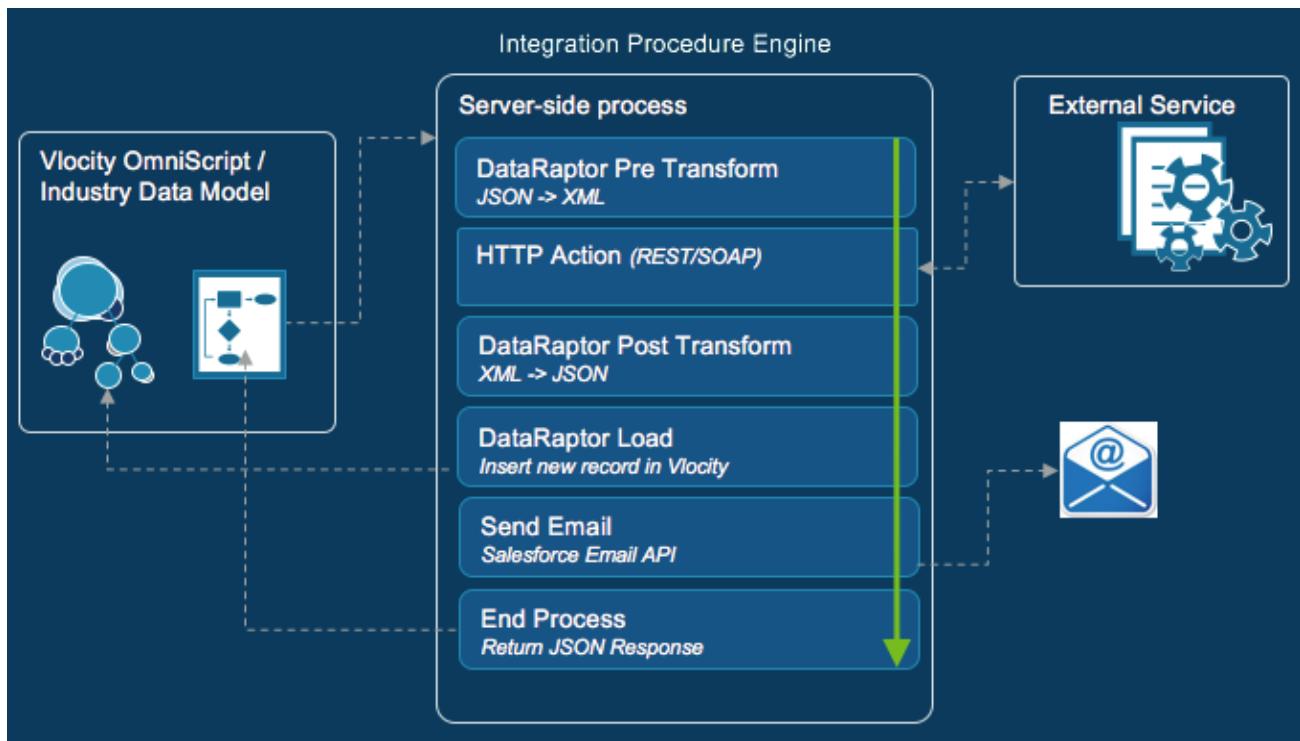
5. Object to Different Object Formation: <https://github.com/TheVishnuKumar/Omn...>

6. List of Item to Object: <https://github.com/TheVishnuKumar/Omn...>

7. Object to List of Item Using Index: <https://github.com/TheVishnuKumar/Omn...>

8. JSON to XML: <https://github.com/TheVishnuKumar/Omn...>

## Vlocity integration procedures:



integration (club data raptors) (Not database.insert, just insert)

Platform event

Named Credential :: end url, username, password

Batch processing

Error Handling

Call Any API

### Response Action:

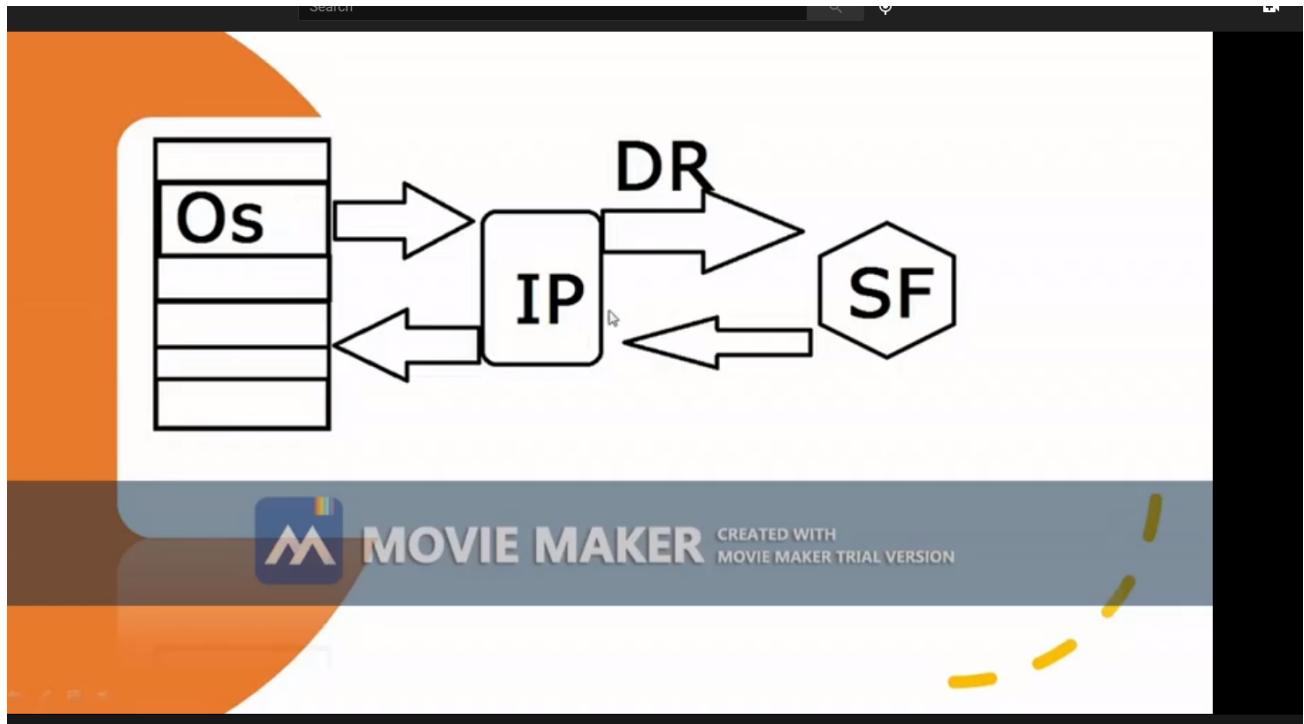
Send data to Omniscript

### Example:

Setvar

LoadDataraptor

responseAction



**URL parameter use directly : e.g: contextId**

**ScreenParameter : Step1:Text1**

-----> NO %

<----- Yes %

**Remote action through integration procedure : calling apex class through IP**

**Remote action : execution of Apex class :: Apex class data fetching**

We need to implement velocity interface first

Then we need to check method whether right method execution is happening or not

Then we need to allow them to execute method

Then we need to return the data in JSON format

**Velocity calculation matrix :: (permutation and computation )**

**Coverage ⇒ premium ⇒ calculation procedure → calculation matrix**

Complex Business logic

**Console: All in one**

**Project explore:** club all component together

**Deploying Omnistudio items :**

Salesforce provides two tools specifically for **Vlocity deployments.**

**IDX BUILD : CLI Based tools**

**IDX Workbench** : Manage and deploy (CLI and GUI)

---

Yml.xml : package.json

Source

Repos

Target

Project : all component (what component u need )

---

**IDX Workbench =====> retrieve files**

Add them into local repo

Push the code

---

---

## The Basic Structure of the

Vlocity Insurance Customer Model,  
the Vlocity Insurance Billing and Payments Model,  
and the Vlocity Insurance Policy Administration model.

---



## **Record Data Activity: General Question:**

**URI** : No need just use variable name

**Screen** : ScreenName:ComponentApiName

% :: value :: Payload/ remote action:- :: %Node:Key%

: :: Grouping key accessing with name

. :: Object Name.fieldName :: Source

### **Note: (PASSING VALUE)**

**API Name of input field in OMNIScript** : holds data/ reference of that input value

**Extract Output Path** : is how output of data stored (variable basically) (list of)

**INPUT JSON PATH** : is how we will pass the data (parameter basically)

**OUTPUT JSON Path** : where variable data will be stored in JSON format

**For formula also** : defining variable in formula used as param variable and will pass the data from json  
And outcome of formula will be stored in a variable and will be used as inputJSONpath

### **How to pass an input to a dataraptor through an omniscrypt?**

Ans: Define input parameter

In extract dataraptor u can use this:

**DataSource**: from where we are getting the value (API NAME OF that) (value could be hardcoded also)

**FilterValue**: paramValue (that will use in data raptor) (**assignment dont use %**)

**Value coming than use % (--) NO (-- yes)**

Dataloader load value without %% from os (mean os to dataraptor just use path)

But for **load raptor** there is no datasource and filter value so use InputJSONPATH

That u use while loading data raptor

Same will be defined in input API Name

Same will be defined in loadDataRaptor JSON Node

And pass additional parameter

### **How to use the output of a dataRaptor in an omniscrypt?**

Ans: to use output, create a text input value and use %Account:Name% in value

### **How to show a record id in an omniscrypt that was created through data raptors?**

Use that instance with %%

---

## How to use omni script in a custom button / vlocity action / omniaction ?

### Embed the code after /lightning/

```
/lightning/cmp/vlocity_ins__vlocityLWCOmniWrapper?c__target=c:emailMessageTransferEnglish&c__layout=lightning&c__tableId=custom:custom18&c__ContextId={!EmailMessage.Id}
```

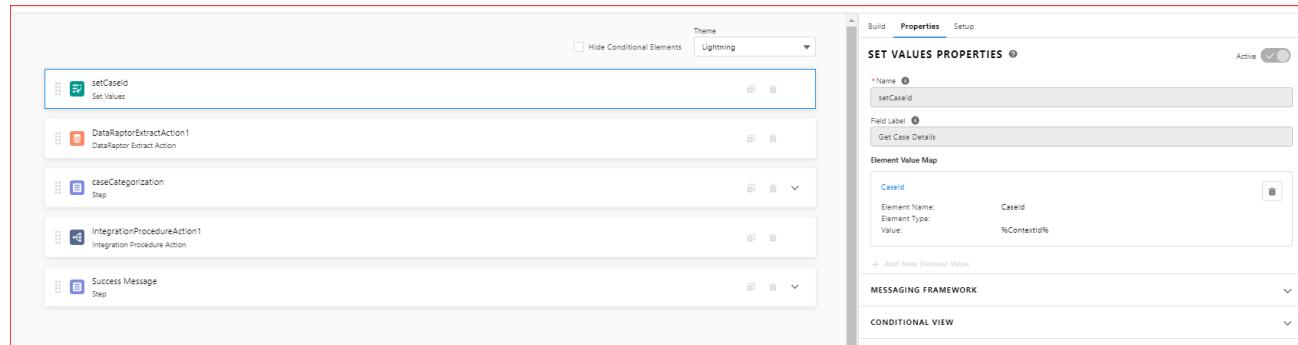
## How to pass record id/ context id into a button ? &c\_\_variable ={!}

### Through URL:

c\_\_ContextId = and usage %ContextId% and in datasource ContextId

## How to set context id variable value into an omniscrypt?

(loaded while component loaded and can pass the value from URI also)



## How to pass context id into integration procedure?

DataRaptorTurboAction2 DataRaptor Turbo Action Active Edit as JSON

Element Name ⓘ

DataRaptor Interface ⓘ

Ignore Cache

**INPUT PARAMETERS** ⓘ

Data Source ⓘ <input type="text" value="ContextId"/>	Filter Value ⓘ <input type="text" value="paramOldEmailRecordId"/>
--	---

**SEND/RESPONSE TRANSFORMATIONS**

Send JSON Path ⓘ <input type="text"/>	Send JSON Node ⓘ <input type="text"/>
Response JSON Path ⓘ <input type="text"/>	Response JSON Node ⓘ <input type="text"/>

**ADDITIONAL INPUT/OUTPUT/FAILURE RESPONSE**

Send Only Additional Input

Additional Input

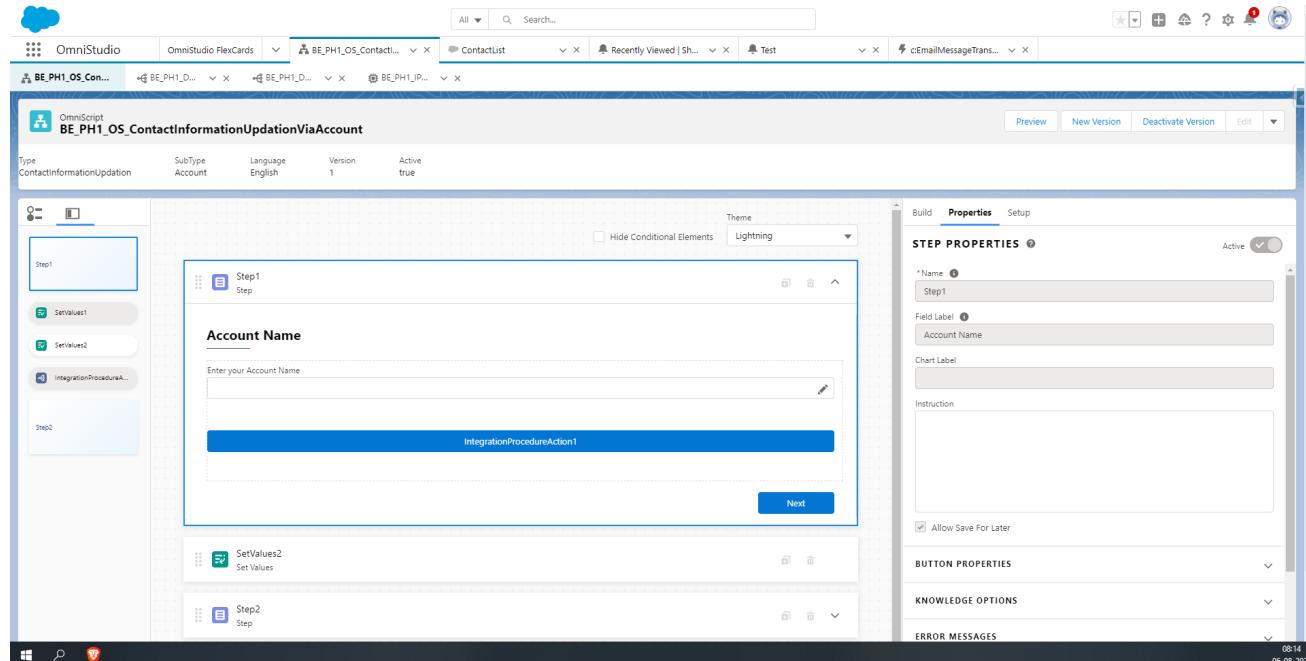
Return Only Additional Output

## How to set the predefined static value and dynamic value?

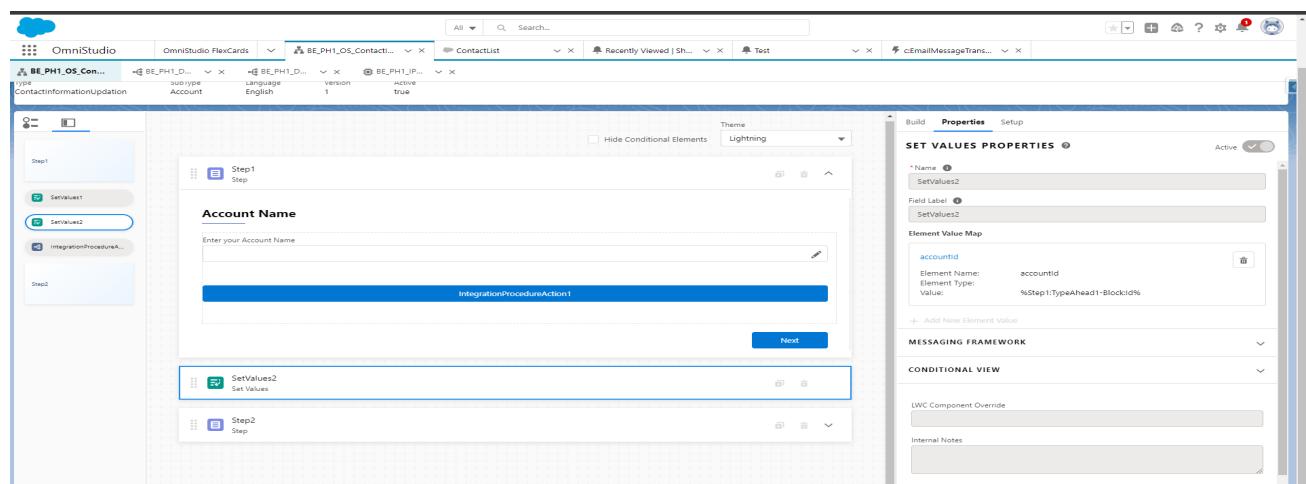
Static value: you can use set variable

For Omniscript:

First Screen:



## Set value in variable



## How to use set variables in an omniscrypt/ dataraptor/ integration procedure ?

Note: when u set static variable in set node, in further using  
you don't need to define setValue:node in omniscrypt

In os: setvalue doesn't work so pass name of variable (accountId)

Whereas in Integration procedure, yes you need to use set:node

And in IP with node name (setvalue1:accountId)

For omniscrypt:

Condition Type

Show Element if True

View Condition ⓘ

Show Element if True

(paramRole = Lawyer – Outsourcing OR paramNetworkLawyer <> NO)

LWC Component Override

For data Raptor: For Integration Procedure:-

for set variable :-

this is the way for set variable and for direct screen input chck(screen input in ip section)

VLOCITY DATARAPTOR BUNDLE  
BE\_PH1\_CaseRecordFetcher

Interface Type	Input Type	Output Type	Required Permission (Optional)	Description
Extract	JSON	JSON		

**EXTRACT FORMULAS OUTPUT OPTIONS PREVIEW**

1 - Case

\*Extract Output Path: `getCaseList` Filter: `Id` = `paramCaseId`

[+ Add Extract Step](#)

**Dynamic value from user database: for that u need to use data raptor and in extract data raptor define output json path as screen element value.**

VLOCITY DATARAPTOR BUNDLE  
BE\_PH1\_CaseRecordFetcher

Interface Type	Input Type	Output Type	Required Permission (Optional)	Description
Extract	JSON	JSON		

**EXTRACT FORMULAS OUTPUT OPTIONS PREVIEW**

1 - Case

\*Extract Output Path: `getCaseList` Interface Field API Name: `Id` = `paramCaseId`

[+ Add Extract Step](#)

VLOCITY DATARAPTOR BUNDLE  
BE\_PH1\_CaseRecordFetcher

Interface Type	Input Type	Output Type	Required Permission (Optional)	Description
Extract	JSON	JSON		

**EXTRACT FORMULAS OUTPUT OPTIONS PREVIEW**

**EXTRACT JSON PATH**  **OUTPUT JSON PATH**

Extract JSON Path	Output JSON Path
<code>getCaseList:Case-Language__c</code>	<code>caseCategorization:Language</code>
<code>getCaseList:Department__c</code>	<code>caseCategorization:Department</code>
<code>getCaseList:Priority</code>	<code>caseCategorization:Priority</code>
<code>getCaseList:Type</code>	<code>caseCategorization&gt;Type</code>

The screenshot shows the Oracle Forms interface for a step named "caseCategorization Step". The main area contains several input fields: "Insurance Policy", "Case Language", "Type", "Department", and "Origin". A chart titled "DR\_Insurance\_Extract" is also present. To the right, there is a sidebar with sections for "Field Label", "Chart Label", "Instruction", and various configuration options like "Allow Save For Later", "BUTTON PROPERTIES", "KNOWLEDGE OPTIONS", "ERROR MESSAGES", and "MESSAGING FRAMEWORK".

## How to use conditions in an omniscrypt?

The screenshot shows the Oracle Forms Conditional View dialog box. The "Condition Type" is set to "Show Element if True". The "View Condition" section contains a condition with the label "Show Element if True" and the expression "(Step2:Radio1 = userHasSelectedCase)". A delete icon is located to the right of the condition entry.

## How to set Error in omniscrypt?

Build Properties Setup

### SET ERRORS PROPERTIES

Active 

\*Name  SetErrors1

Field Label  SetErrors1

Element Error Map

Step1	
Element Name:	Step1
Element Type:	Step
Value:	%Step1:CustomLWC1:getClaimParticipantId%

+ Add New Element Value

MESSAGING FRAMEWORK 

CONDITIONAL VIEW 

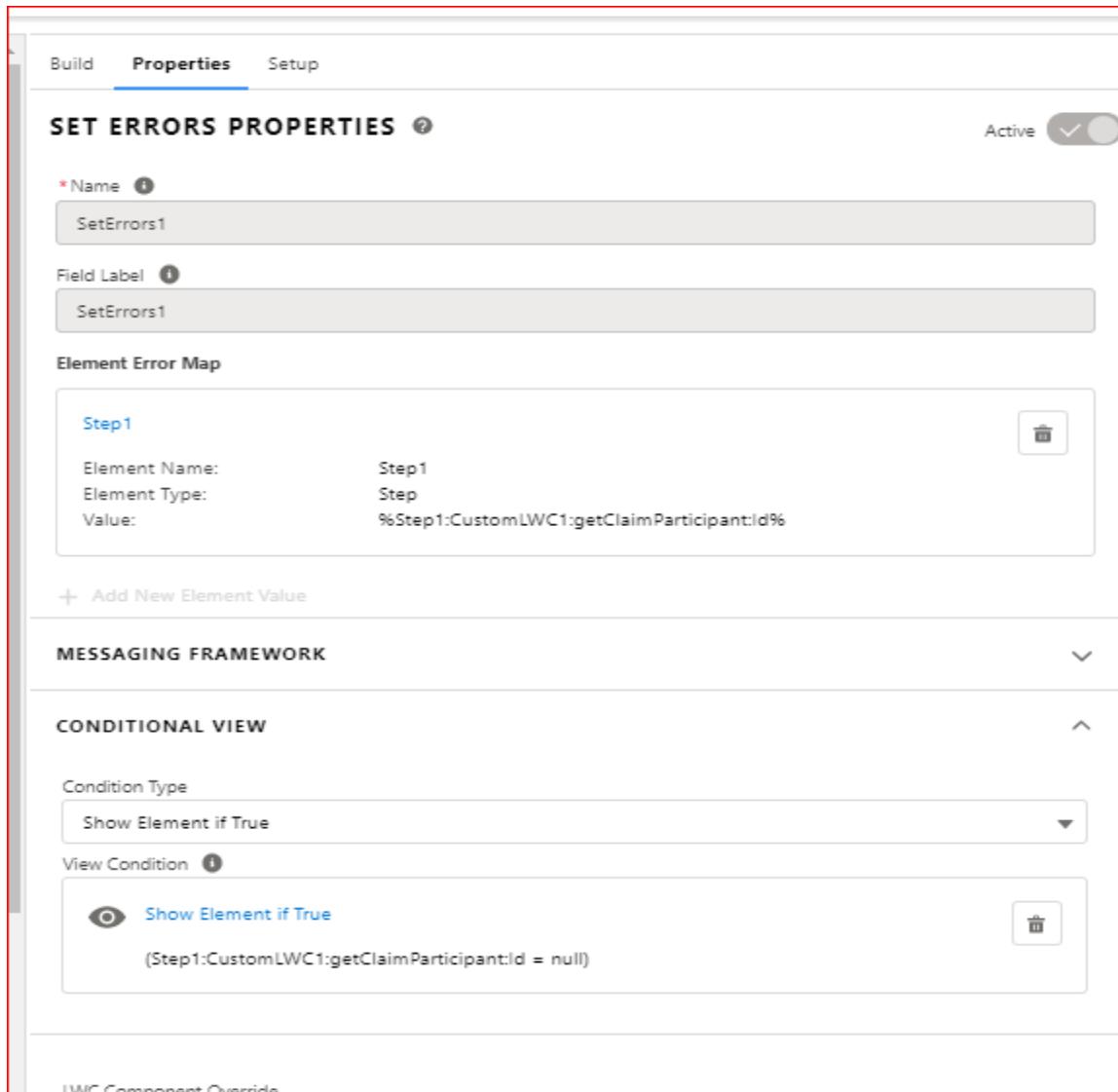
Condition Type

Show Element if True

View Condition 

 Show Element if True	
(Step1:CustomLWC1:getClaimParticipantId = null)	

LWC Component Override

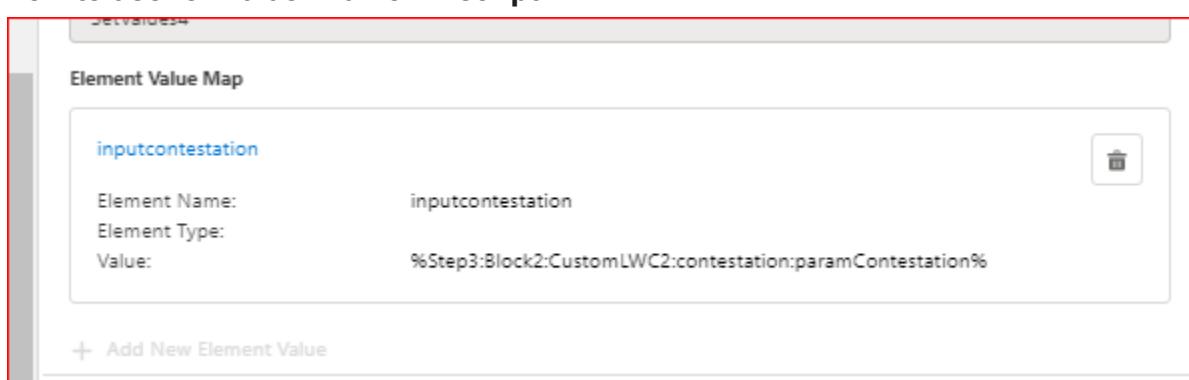


## How to use formulas in an omniscrypt ?

Element Value Map

inputcontestation	
Element Name:	inputcontestation
Element Type:	
Value:	%Step3:Block2:CustomLWC2:contestation:paramContestation%

+ Add New Element Value



\* Name ⓘ  
Formula1

Field Label ⓘ

Data Type  
Boolean

Expression ⓘ  
IF(%inputcontestation% == "true", true, false)

Hide

**CONDITIONAL VIEW**

LWC Component Override

Internal Notes

[Edit Properties As JSON](#)

Condition Type  
Show Element if True

View Condition ⓘ  
 Show Element if True  
(Formula1 <> false)

**How to navigate in omniscrypt?**

## NAVIGATE ACTION PROPERTIES ⓘ

Active

\* Name ⓘ  
NavigateAction3

Field Label ⓘ  
FINISH

Button Variant ⓘ  
brand

Icon Name ⓘ

Page Reference Type ⓘ  
Record

Replace ⓘ

Object API Name ⓘ  
Financial\_Account\_Transaction\_\_c

Target Parameters ⓘ

Record Action ⓘ  
View

Record ID  
%financialAccountTransactionOutput%

Validation Required  
None

**MESSAGING FRAMEWORK**

**CONDITIONAL VIEW**

## Data Raptor:

How to set value directly from Omniscript to data raptor?  
Just Use JSON input without using %% e.g: selected:node

## Multiple query inside an Extract Data Raptor:

The screenshot shows the Vlocity DataRaptor interface for the bundle BE\_PH1\_DRE\_ClaimParticipant. It displays three extract steps:

- 1 - ClaimParticipant**: Extract Output Path: getClaimParticipant, Filter: ClaimId = paramClaimId
- 2 - Claim**: Extract Output Path: getClaim, Filter: Id = paramClaimId
- 3 - Contact**: Extract Output Path: getContact, Filter: Id = getClaimParticipant.ParticipantContactId

At the bottom right of the interface, there is a button labeled "+ Add Extract Step".

The screenshot shows the Vlocity DataRaptor interface for the bundle BE\_PH1\_DRE\_OLDCASEEMAILMESSAGE. It displays two extract steps:

- 1 - EmailMessage**: Extract Output Path: oldCaseEmailMessageRecord, Filter: Id = paramOldEmailRecordId
- 2 - ContentDocumentLink**: Extract Output Path: getAttachmentRecord, Filter: LinkedEntityId = oldCaseEmailMessageRecord.Id

At the bottom right of the interface, there is a button labeled "+ Add Extract Step".

## Loading Data in Multiple Object Together :

The screenshot shows the Vlocity DataRaptor interface for the bundle BE\_PH1\_DR\_NEWCASEEMAILMESSAGE. It displays a load step for the EmailMessage object and a domain object step for the ContentDocumentLink object:

- 1 - EmailMessage**: Load
- 2 - ContentDocumentLink**: \*Domain Object: ContentDocumentLink, \*Domain Object Field: LinkedEntityId, United Object: 1 - EmailMessage, Id

At the bottom right of the interface, there are buttons for "Edit", "Quick Match", "Clone", and "Export".

## IN Operator in DataRaptor:

It is only available in turbo data raptor.

And the way to use it in:-

### Screenshot1:

The screenshot shows the Vlocity DataRaptor interface for a bundle named BE\_PH1\_ContactListFetcher. The 'EXTRACT' tab is selected. In the 'Contact' section, there is a 'Filter' field set to 'IN' with a parameter named 'paramName'. Below this, under 'SEARCH FIELDS', there is a dropdown menu containing 'AccountName', 'AssistantName', and 'AssistantPhone'. On the left, there is a list of fields: 'CleanStatus', 'CreatedById', and 'CreatedDate'. The 'paramName' field is highlighted with a red border.

### Screenshot2:-

The screenshot shows the Vlocity DataRaptor interface for the same bundle. The 'PREVIEW' tab is selected. The 'Input Parameters' section shows a JSON object with 'paramName' set to an array containing 'hey 786' and 'FromEmail21'. The 'Response' section displays the query results as a JSON array:

```
[{"getContact": [{"RecordTypeId": "0125f000001C5ZdAAK", "Email": "ilovenagpur@gmail.com", "Name": "FromEmail21", "LastName": "FromEmail21", "Id": "0035f000006cluYAAQ"}, {"RecordTypeId": "0125f000001C5ZdAAK", "Email": "dhanuka995@gmail.com", "Name": "hey 786", "LastName": "hey 786", "Id": "0035f000006cludAAA"}]}
```

The 'Errors/Debug Output' section shows the SQL query and its execution details:

```
2021-10-20T22:0 SELECT Id,Assis,AssistantName,A LastName,Name,P WHERE (Name IN ('FromEmail21')) 2021-10-20T22:0 Found: 2
```

## Null value Check In Dataraptor:-

**OMNISTUDIO DATARAPTOR BE\_PH1\_DRE\_BrokerContactListFetcher**

Interface Type	Input Type	Output Type	Required Permission (Optional)	Description
Extract	JSON	JSON		
EXTRACT	FORMULAS	<b>OUTPUT</b>	OPTIONS	PREVIEW
EXTRACT JSON PATH <input type="text"/>		OUTPUT JSON PATH <input type="text"/>		
getContactList:AccountId		AccountId		
getContactList:Contact BE_PH1_Language __c		ContactLanguage		
getContactList:Contact.Email		ContactEmail		
getContactList:Contact.Name		ContactName		
getContactList:Contact.Phone		ContactPhone		
getContactList:ContactId		ContactId		

DREExtract\_FetchRecordTypeIds  
DataRaptor Extract Action

DREToBrokerContact  
DataRaptor Extract Action

SetValue1  
Set Values

SetValue2  
Set Values

Response JSON Node: contactList

ERROR MESSAGES

MESSAGING FRAMEWORK

CONDITIONAL VIEW

BE\_PH2\_GetBrokerAccount  
DataRaptor Extract Action

ErrorMessage  
Step

DREExtract\_FetchRecordTypeIds  
DataRaptor Extract Action

DREToBrokerContact  
DataRaptor Extract Action

SetValue1  
Set Values

SetValue2  
Set Values

SelectContact

**SET VALUES PROPERTIES**

- Name: SetValues1
- Field Label: SetValues1
- Element Value Map:
  - NullCheck
    - Element Name: NullCheck
    - Element Type: =COUNT(%contactList:ContactName%)

+ Add New Element Value

MESSAGING FRAMEWORK

CONDITIONAL VIEW

SelectContact  
Step

Select Contact

CustomLWC3  
<<:BE\_PH2\_BrokerContactList />

Previous Next

MESSAGING FRAMEWORK

CONDITIONAL VIEW

Condition Type: Show Element if True

View Condition: Show Element if True  
(NullCheck > 0)

LWC Component Override

## Dataraptor JSON output converting Into ARRAY:-

The screenshot displays three distinct configurations within the Dataraptor interface:

- BE\_PH1\_DRE\_BrokerContactListFetcher:** This configuration is set up to extract contact data from a Broker Contact List. It uses an Extract interface type, Input Type JSON, and Output Type JSON. The output is mapped to various fields: AccountId, ContactLanguage, ContactEmail, ContactName, ContactPhone, and ContactId.
- DRExtract\_FetchRecordTypeIds:** A DataRaptor Extract Action step, which is part of a larger integration flow.
- DRToBrokerContact:** Another DataRaptor Extract Action step, also part of the integration flow.
- SetValues1:** A Set Values step, part of the integration flow.
- SetValues2:** A second Set Values step, part of the integration flow.
- SelectContact:** A Step component, likely used for further processing or mapping.
- CustomLWC3:** A Lightning Web Component (LWC) configuration, specifically for the bE\_PH2\_BrokerContactList component. It includes properties like records and a property source of %listView%.

### Note:-

**flexcard automatically do the conversion**  
**And in integration procedure we need to use set variable**

## Integration Procedure:

How to take screen input and pass into dataraptor via IP? **Screen:component**

The screenshot shows the configuration for a DataRaptor Turbo Action named "DataRaptorTurboAction1". It includes fields for Element Name (DataRaptorTurboAction1), DataRaptor Interface (BE\_PH1\_GetNewCaseID), and an "Ignore Cache" checkbox. Under "INPUT PARAMETERS", there is a "Data Source" field set to "Step1:Text1" and a "Filter Value" field set to "userInputCaseNumber". In the "SEND/RESPONSE TRANSFORMATIONS" section, there are fields for "Send JSON Path" and "Send JSON Node". Below this, under "ADDITIONAL INPUT/OUTPUT/FAILURE RESPONSE", there is a checkbox for "Send Only Additional Input" and an "Additional Input" field.

How to show output into screen via IP? Need to use response Action

Two point Needs to be noted down:

- 1) Use data transform before sending data back to omniscrypt
- 2) Or only send node that is required to send

Using data transform:

The screenshot shows the Integration Procedure Designer with a procedure named "BE\_PH1\_IP\_EmailProcessData". The "STRUCTURE" tab displays various actions: SetValues2, DataRaptorTurboAction1, ResponseAction2, ResponseAction1, DataRaptorTurboAction2, ResponseAction4, SetNewCaseId, ResponseAction5, DataRaptorPostAction1, DeleteAction1, DataRaptorTransformAction1, and ResponseAction3. The "PROPERTIES" tab is focused on "ResponseAction3", which is a "Response Action". It has fields for Element Name (ResponseAction3), Response Format (JSON), and Response Headers (Return Full Data JSON). Under "SEND/RESPONSE TRANSFORMATIONS", there are fields for "Send JSON Path" (DataRaptorTransformAction1) and "Send JSON Node". Below this, under "ADDITIONAL OUTPUT RESPONSE", there is a "Execution Conditional Formula" field and an "Internal Notes" field.

Will be done like what output we are getting in DataRaptor if no additional node %Email\_Message:Id% for record redirecting

## Using node only:

▼ SEND/RESPONSE TRANSFORMATIONS

Send JSON Path <input type="text"/>	Send JSON Node <input type="text"/>
Response JSON Path <input type="text"/>	Response JSON Node <input type="text"/>

▼ ADDITIONAL OUTPUT RESPONSE

Return Only Additional Output.

Additional Output

getCaseId	<input type="text"/> fx %DataRaptorPostAction1.Case_1.id%
-----------	---

Execution Conditional Formula

## How to set value in IP to pass into load raptor?

SetNewCaseId Set Values

Active [Edit as JSON](#)

Element Name ⓘ  
SetNewCaseId

Element Value Map

Element Name	Type	Value
newCaseId	JSON Node	<code>fx %DataRaptorTurboAction1:newCaseId.Id%</code>
newRelatedCaseId	JSON Node	<code>fx %DataRaptorTurboAction1:newCaseId.Id%</code>
oldEmailData	JSON Node	<code>fx %DataRaptorTurboAction2%</code>

[+ Add New Value](#)

▼ RESPONSE TRANSFORMATIONS

Response JSON Path ⓘ

Response JSON Node ⓘ

SetValues1 Set values

Active [Edit as JSON](#)

Element Name ⓘ  
SetValues1

Element Value Map

Element Name	Type	Value
paramLanguage	JSON Node	<code>fx %caseCategorization:Language%</code>
paramDepartment	JSON Node	<code>fx %caseCategorization:Department%</code>
paramType	JSON Node	<code>fx %caseCategorization:Type%</code>
paramPriority	JSON Node	<code>fx %caseCategorization:Priority%</code>
paramInternalComments	JSON Node	<code>fx %caseCategorization:InternalComments%</code>
paramCaseId	JSON Node	<code>fx %CaseId%</code>
paramPolicyName	JSON Node	<code>fx %caseCategorization:InsurancePolicy-Block-InsuranceId%</code>

[+ Add New Value](#)

▼ RESPONSE TRANSFORMATIONS

Response JSON Path ⓘ

Response JSON Node ⓘ

## How to pass JSON input into load data raptor through ip with additional input?

DataRaptorPostAction1 DataRaptor Post Action

Active [Edit as JSON](#)

Element Name ⓘ  
DataRaptorPostAction1

DataRaptor Interface ⓘ  
BE\_PH1\_DRT\_Load\_Case\_Details

▼ SEND/RESPONSE TRANSFORMATIONS

Send JSON Path ⓘ  
SetValues1

Response JSON Path ⓘ

Send JSON Node ⓘ

Response JSON Node ⓘ

▼ ADDITIONAL INPUT/OUTPUT/FAILURE RESPONSE

Send Only Additional Input

Additional Input  
[Add Key/Value Pair](#)

Return Only Additional Output

Additional Output  
[Add Key/Value Pair](#)

Return Only Failure Response

DataRaptorPostAction1 DataRaptor Post Action

Active [Edit as JSON](#)

Element Name ⓘ  
DataRaptorPostAction1

DataRaptor Interface ⓘ  
BE\_PH1\_DR\_NEWCASEEMAILMESSAGE

▼ SEND/RESPONSE TRANSFORMATIONS

Send JSON Path ⓘ  
%SetNewCaseId:oldEmailData%

Response JSON Path ⓘ

Send JSON Node ⓘ

Response JSON Node ⓘ

▼ ADDITIONAL INPUT/OUTPUT/FAILURE RESPONSE

Send Only Additional Input

Additional Input

newCaseId  %SetNewCaseId:newCaseId%

newRelatedCaseId  %SetNewCaseId:newRelatedCaseId%

---

## Condition in IP:

Conditional Block 1

Element Name ⓘ  
ConditionalBlock1

Execution Conditional Formula

```
%DataRaptorExtractAction1:getAttachmentRecord:Id% != Null
```

## Delete Action :

DeleteAction1 Delete Action

Active [Edit as JSON](#)

Element Name ⓘ  
DeleteAction1

Delete SObject

All Or None ⓘ  EmailMessage Type ⓘ

Response JSON Path ⓘ

Response JSON Node ⓘ

▼ RESPONSE TRANSFORMATIONS

Response JSON Path ⓘ

Response JSON Node ⓘ

▼ ADDITIONAL OUTPUT/FAILURE RESPONSE

Return Only Additional Output

Additional Output

## LOOP BLOCK:

```
for(Integer i: getContactList){  
    if(i==2){  
        List<Contact> Li = [[select id from Contact where Name =: i ]];  
        return li;  
    }  
}
```

```
for(Integer i: getContactList){ =====> Loop Block  
    if(i==2){ =====> Conditional Block  
        List<Contact> Li = [[select id from Contact where Name =: i ]]; => dataraptor  
        return li; =====> Loop Block  
    }  
}
```

Example:

The screenshot shows the DataRaptor interface with a 'LoopBlock1' configuration. The 'STRUCTURE' tab on the left lists 'Procedure Configuration', 'LoopBlock1' (selected), and 'ResponseAction1'. The 'PROPERTIES' tab on the right shows the following fields:

- Element Name: LoopBlock1
- Loop List: IterationList
- Additional Loop Output: DataRaptorTurboAction1 (%DataRaptorTurboAction1%)
- Execution Conditional Formula: (empty)
- Internal Notes: (empty)

Handwritten yellow annotations are present on the screen:

- The word "ITERAT" is written vertically along the right edge of the properties panel.
- The word "OUT PUT" is written vertically below the "Execution Conditional Formula" field.

**PROPERTIES** PREVIEW

DataRaptorTurboAction1 DataRaptor Turbo Action

Active [Edit as JSON](#)

Element Name

DataRaptor Interface

Ignore Cache

**INPUT PARAMETERS** [Add Input Parameter](#)

Data Source <input type="text" value="IterationList:LastName"/>	Filter Value <input type="text" value="paramLastName"/>
---	---

**SEND/RESPONSE TRANSFORMATIONS**

Send JSON Path

Response JSON Path

Send JSON Node

Response JSON Node

**ADDITIONAL INPUT/OUTPUT/FAILURE RESPONSE**

Send Only Additional Input

Additional Input [Add Key/Value Pair](#)

Return Only Additional Output

Additional Output

12:22  
24-08-2021

**PROPERTIES** PREVIEW

Input Parameters [Edit as Params](#)

```
{
  "iterationList": [
    {
      "LastName": "d"
    },
    {
      "LastName": "L"
    }
  ]
}
```

Response Browser: 894ms - Server: 206ms - Apex CPU: 240ms

Errors/Debug Output [Debug Log](#)

```

{
  "response": {},
  "ResponseAction1Status": true,
  "LoopBlock1": [
    {
      "DataRaptorTurboAction1": {
        "getContactList": [
          {
            "RecordType": "0125g000001Jnd8AAE",
            "LastName": "Dhanuka is great learning boy",
            "Id": "0035g000008cJ9mAAA"
          },
          {
            "RecordType": "0125g000001Jnd8AAE",
            "LastName": "Dhanuka is great learning boy",
            "Id": "0035g000008cJ9mAAA"
          },
          {
            "LastName": "Bond",
            "FirstName": "John",
            "Id": "0035g000008TYCIAAS"
          },
          {
            "LastName": "Davis",
            "FirstName": "Josh",
            "Id": "0035g000008TYCEAAS"
          },
          {
            "LastName": "O'Cruz",
            "FirstName": "Liz",
            "Id": "0035g000008TYCJAAS"
          },
          {
            "LastName": "Nedarker",
            "FirstName": "Siddartha",
            "Id": "0035g000008TYCJAAS"
          }
        ]
      }
    }
  ]
}
  
```

Errors/Debug Output [Debug Log](#)

```

{
  "response": {},
  "ResponseAction1Status": true,
  "LoopBlock1": [
    {
      "DataRaptorTurboAction1": {
        "getContactList": [
          {
            "RecordType": "0125g000001Jnd8AAE",
            "LastName": "Ripley",
            "FirstName": "Tom",
            "Id": "0035g000008TYCIAAS"
          },
          {
            "RecordType": "0125g000001Jnd8AAE",
            "LastName": "Llorrac",
            "FirstName": "Jake",
            "Id": "0035g000008TYCNAAS"
          },
          {
            "RecordType": "0125g000001Jnd8AAE",
            "LastName": "Dhanuka is great learning boy",
            "Id": "0035g000008cJ9mAAA"
          },
          {
            "RecordType": "0125g000001Jnd8AAE",
            "LastName": "Dhanuka is great learning boy",
            "Id": "0035g000008cJ9mAAA"
          }
        ]
      }
    }
  ]
}
  
```

12:22  
24-08-2021

## Try catch Block:

```
try{  if(x!= Null) {  System.debug(""); }}
```

```
catch(Exception e ){      System.debug(Exception e); }
```

```
Try{                                     =====> on data raptor
```

```
    if(x!= Null){  System.debug(""); }}
```

```
catch(Exception e ){                     =====> on try block
```

```
    System.debug(Exception e); }
```

STRUCTURE      PROPERTIES      PREVIEW

Procedure Configuration

TryCatchBlock3 Try Catch Block

Element Name: TryCatchBlock3

Failure Response

    FailedMessage: fuckoff

Add Key/Value Pair

CUSTOM FAILURE RESPONSE

    Remote Class: Remote Method:

Fall On Block Error

Execution Conditional Formula

Internal Notes

STRUCTURE      PROPERTIES      PREVIEW

Procedure Configuration

TryCatchBlock3

DataRaptorTurboAction2

ResponseAction1

Additional Output

Add Key/Value Pair

Return Only Failure Response

Failure Response

Add Key/Value Pair

Execution Conditional Formula

Fall On Step Error

Failure Conditional Formula: ISBLANK(%DataRaptorTurboAction2%)

Chain On Step

Action Message

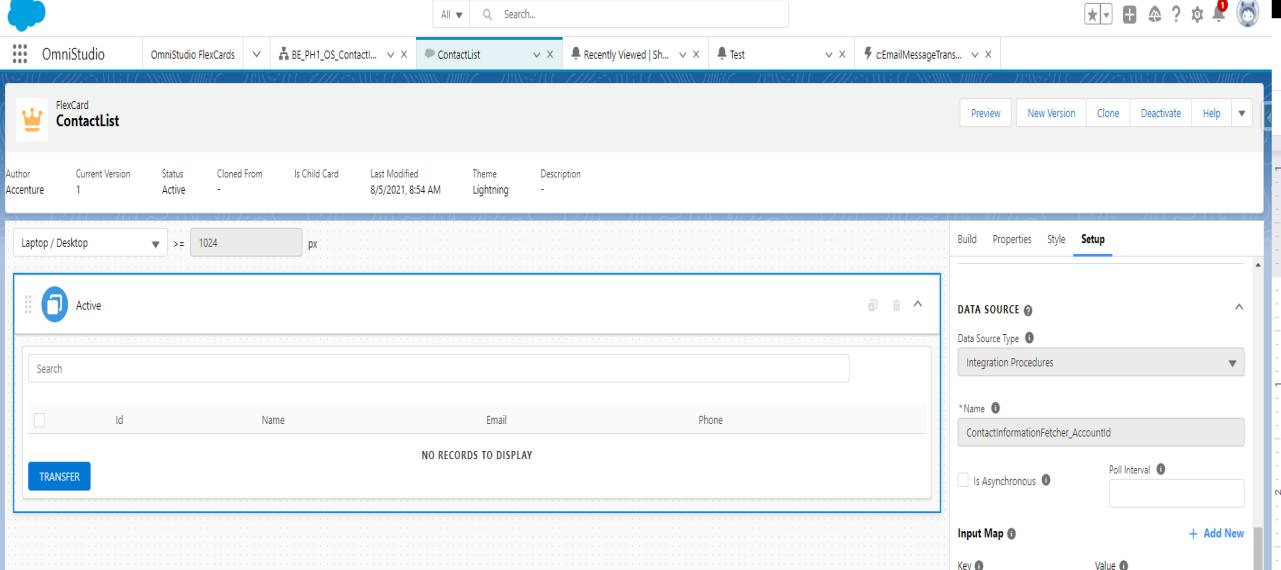
Internal Notes

## Passed variable in flex card

The screenshot shows the OmniStudio interface with the following details:

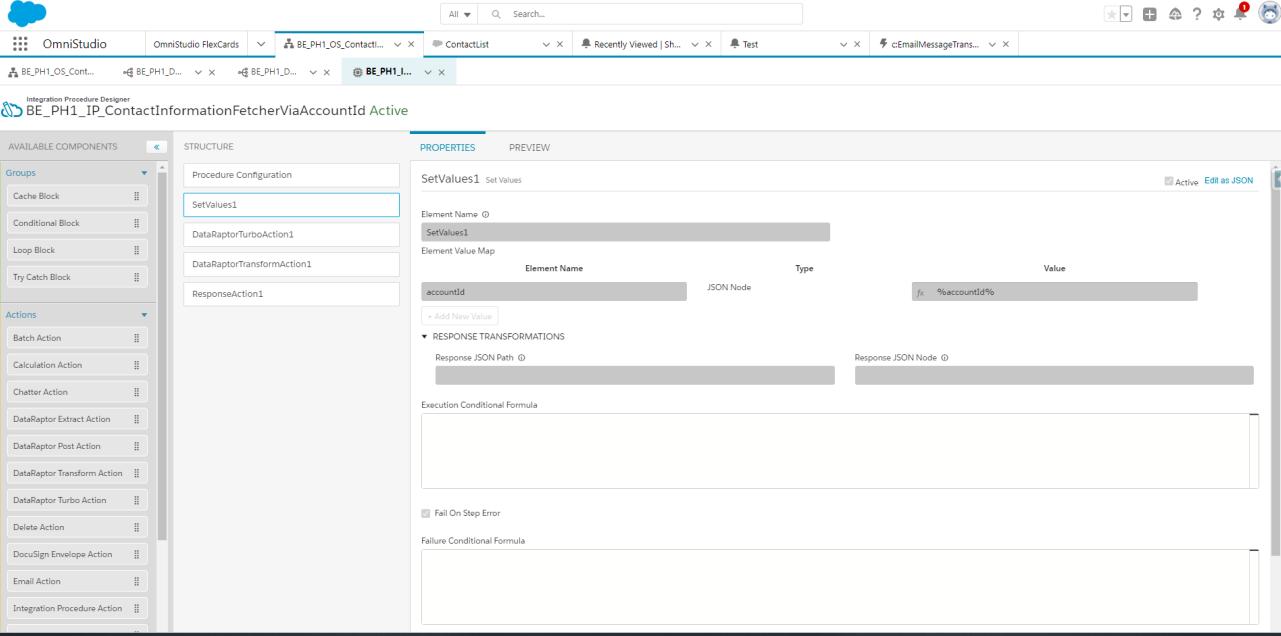
- OmniStudio FlexCards** tab is selected.
- BE\_PH1\_OS\_Con...** card is open, showing a ContactInformationUpdate record with type: Account, subtype: Account, language: English, version: 1, active: true.
- IntegrationProcedureAction1** step is active, containing an **Account Name** field where "Enter your Account Name" is displayed.
- SetValues1** and **SetValues2** steps are shown below.
- IntegrationProcedureAction2** step is partially visible.
- Step2** is the next step in the flow.
- Contact List** step displays a component with the code: `CustomLWC1  
kc:cContactList_1_Accenture />`.
- Properties** tab is selected for the **CUSTOM LIGHTNING WEB COMPONENT PROPERTIES** section.
- Name**: CustomLWC1
- Field Label**: CustomLWC1
- Lightning Web Component Name**: cContactList\_1\_Accenture
- Standalone LWC**: checked
- Property Name**: record-id
- Property Source**: %SetValues2%
- Conditional View** section is collapsed.
- Internal Notes** section is collapsed.
- Edit Properties As JSON** link is present.

## Passed input in integration procedure via flex card



The screenshot shows the OmniStudio FlexCard ContactList interface. At the top, there's a header with tabs like 'All', 'Search...', and various status indicators. Below the header is a table with columns: Author, Current Version, Status, Cloned From, Is Child Card, Last Modified, Theme, and Description. The theme is set to 'Lightning'. The main area displays a table with columns 'Id', 'Name', 'Email', and 'Phone'. A search bar is at the top of the table. A large blue button labeled 'TRANSFER' is located at the bottom left of the table area.

## Processed input variable in integration procedure



The screenshot shows the Integration Procedure Designer for the procedure 'BE\_PH1\_IP\_ContactInformationFetcherViaAccountId'. The 'STRUCTURE' tab is active, showing a list of actions: 'SetValues1', 'DataRaptorTurboAction1', 'DataRaptorTransformAction1', and 'ResponseAction1'. The 'SetValues1' action is currently selected. In the 'PROPERTIES' tab, the 'Element Name' is set to 'SetValues1'. Under 'Element Value Map', there is a single entry for 'accountId' with a value of '%\$accountid%'. Below this, under 'RESPONSE TRANSFORMATIONS', there are fields for 'Response JSON Path' and 'Response JSON Node'. There are also sections for 'Execution Conditional Formula' and 'Failure Conditional Formula'.

## Proceed output data through integration procedure (transform, [0,0,0])

The screenshot shows the OmniStudio Integration Procedure Designer interface. A 'ResponseAction1' component is selected in the Properties tab. The 'Response Format' is set to 'JSON'. The 'Send JSON Path' is set to 'DataRaptorTransformAction1'. The 'Response JSON Path' is set to 'Response JSON Node'. The 'Structure' tab shows other components like 'Procedure Configuration', 'SetValue1', 'DataRaptorTurboAction1', and 'DataRaptorTransformAction1'.

The screenshot shows the OmniStudio Integration Procedure Designer interface. A 'BE\_PH1\_DRT\_ContactListTransformation' component is selected. The 'Input' section shows a JSON array of records. The 'Response' section shows the transformed data. The 'Errors/Debug Output' section shows a debug log with the message 'Debug Log'.

Or u can use Result JSON PATH Option in flex card.

Set value passing in flex card : **record-id %%**

#### CUSTOM LIGHTNING WEB COMPONENT PROPERTIES

Property Name **i**

record-id

Property Source **i**

%ContextId%



+ Add New Property

Pass value in IP :

**accountId {recordId}**

#### Input Map **i**

+ Add New

Key **i**

paramCaseId

Value **i**

{recordId}



#### Options >

#### TEST PARAMETERS **i**

+ Add New

Key **i**

recordId

Value **i**

5005g00000AJDBAAA5



Result JSON Path **i**

In dataraptor:

1 - EmailMessage



\* Extract Output Path

Interface Field API Name

getEmailList

ParentId

=

paramCaseId

+ Add Extract Step

## How to Pass attributes with multiple values from OS to flexcard?

First set the dynamic value or static value in set node: through JSON Editor

**SET VALUES PROPERTIES** 

Active 

\* Name  SetValues1

Field Label  SetValues1

**Element Value Map**

SetValues1	
Element Name:	SetValues1
Element Type:	Set Values
Value:	Object (click 'Edit as JSON' to modify)

+ Add New Element Value

Build **Properties** Setup

**SET VALUES PROPERTIES** 

Active 

```
1  {
2    "controlWidth": 12,
3    "label": "SetValues1",
4    "elementValueMap": {
5      "SetValues1": {
6        "coverage": "%SelectCoverageType:Cov
7        "case": "%ContextId%",
8        "policy": "%CaseDetails:InsurancePolicy%"
9      }
10     },
11     "showPersistentComponent": [
12       true,
13       false
14     ],
15     "show": null,
16     "HTMLTemplateId": "",
17     "wpm": false,
18     "ssm": false,
19     "message": {},
20     "pubsub": false
21   }
```

Build **Properties** Setup

## SET VALUES PROPERTIES

Active 

```
1  {
2      "controlWidth": 12,
3      "label": "SetValues1",
4      "elementValueMap": {
5          "SetValues1": {
6              "coverage": "0YG5e0000004C93GAE",
7              "case": "5005e000006SUNsAAO",
8              "policy": "0YT5e000000NEXQGA4"
9          }
10     },
11     "showPersistentComponent": [
12         true,
13         false
14     ],
15     "show": null,
16     "HTMLTemplateId": "",
17     "wpm": false,
18     "ssm": false,
```

## Second map with record-id variables:-

The screenshot shows the 'Properties' tab for a custom Lightning Web Component (LWC) named 'CustomLWC1'. The component is active. It has a field label 'CustomLWC1' and a lightning web component name 'cfBE\_PH1\_ShowExistingClaim'. The 'Standalone LWC' checkbox is checked. Below this, there is a section for 'CUSTOM LIGHTNING WEB COMPONENT PROPERTIES' with a property named 'record-id' and a source of '%SetValues1%'. A button to 'Add New Property' is also present. At the bottom, there is a 'CONDITIONAL VIEW' section and an 'Internal Notes' area.

Build    **Properties**    Setup

**CUSTOM LIGHTNING WEB COMPONENT PROPERTIES**

\* Name ⓘ  
CustomLWC1

Field Label ⓘ  
CustomLWC1

\* Lightning Web Component Name ⓘ  
cfBE\_PH1\_ShowExistingClaim

Standalone LWC ⓘ

**CUSTOM LIGHTNING WEB COMPONENT PROPERTIES**

Property Name ⓘ  
record-id

Property Source ⓘ  
%SetValues1%

[+ Add New Property](#)

**CONDITIONAL VIEW**

Internal Notes

Third in flex card use as parent child relationship:

Build Properties Style **Setup**

**Input Map** i + Add New

Key <small>i</small>	Value <small>i</small>
paramPolicyId	{recordId.policy}
paramCaseId	{recordId.case}
paramCoverageId	{recordId.coverage}

**Options** >

**TEST PARAMETERS** i + Add New

Key <small>i</small>	Value <small>i</small>
recordId.policy	0YT5e000000NEXQGA4
recordId.case	5005e000006SUNsAAO
recordId.coverage	0YG5e0000004C93GAE

## How to pass data from flex card to child flex card:-

### Parent card:

The screenshot shows the 'Setup' tab of a Parent card configuration. It includes sections for 'CUSTOM FIELDS', 'DATA SOURCE', and 'TEST PARAMETERS'. The 'DATA SOURCE' section contains a 'Data Source Type' dropdown set to 'SOQL Query' and a query editor with the SOQL query: 'select id,Name from account'. The 'TEST PARAMETERS' section has a 'Result JSON Path' field.

The screenshot shows the 'Element Name' field set to 'FlexCard-3'. The 'FlexCard Name' field is set to 'Learning\_Parent\_Child'. The 'Data Node' field is set to 'Data Node'. The 'Select State' dropdown is set to 'Active'. The 'Enable Tracking' checkbox is checked. Under 'Attributes', there is an attribute 'accountId' with a value '{Id}'. The 'Conditions' section is present at the bottom.

## Child card:-

Build    Properties    Style    **Setup**

**DATA SOURCE**

Data Source Type SOQL Query

Query

```
Select id , Name from contact where Account.Id= '{Parent.accountId}'
```

**Options**

**TEST PARAMETERS** [+ Add New](#)

Key	Value
Parent.accountId	0015e00000EqR9hAAF

Result JSON Path

```
[{"id": "0015e00000EqR9hAAF", "Name": "Test Contact"}]
```

## Data transfer in between component:

Action element :

Update in omniscrypt

Update in card and then fire the event

launch new omniscrypt in flyout or in new tab

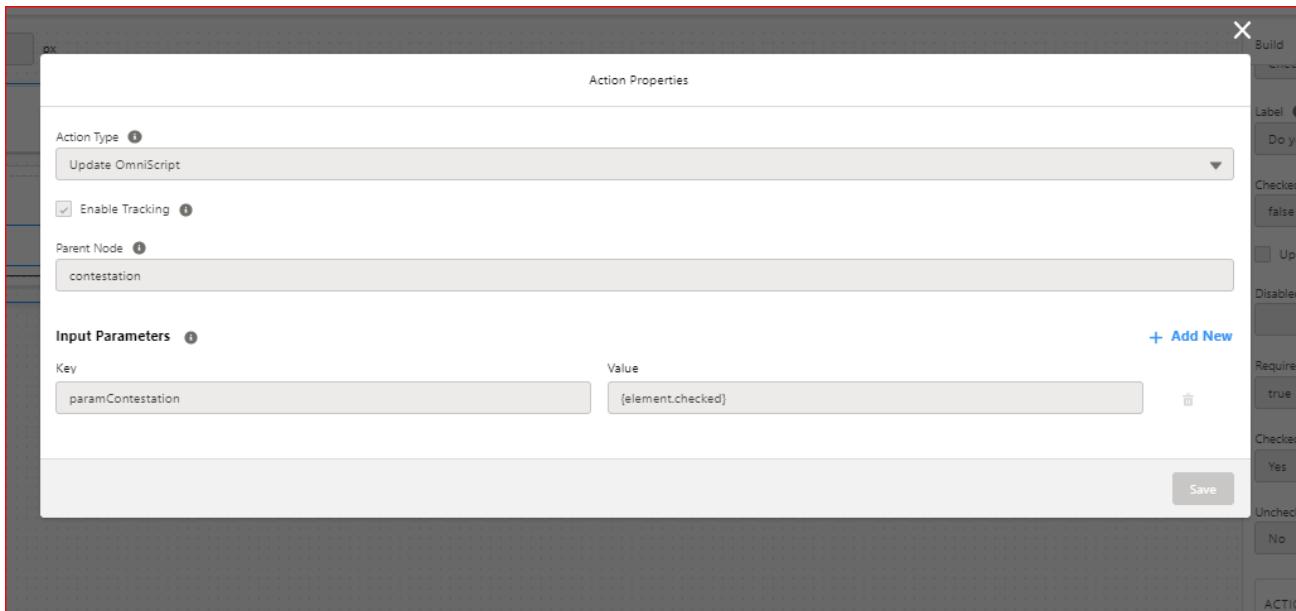
## update omniscrypt :

define parent node and path (here parent node is step2 and path is Text1)

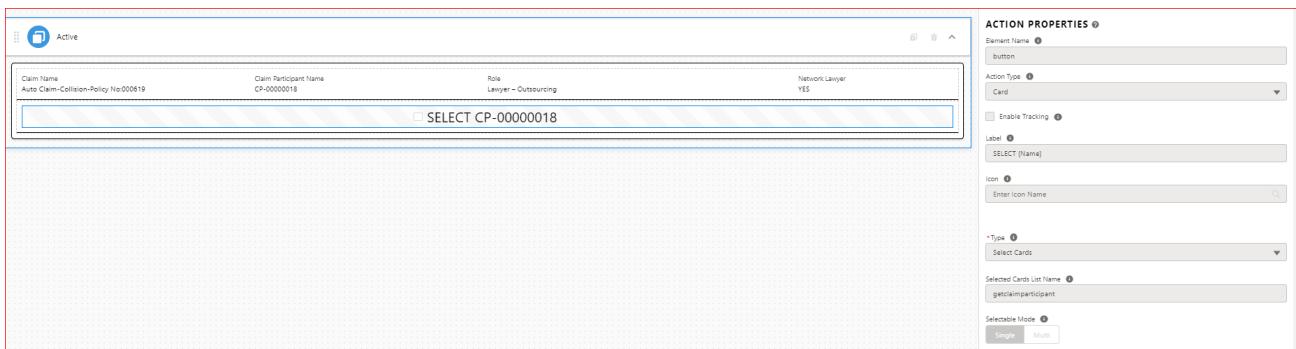
it will come in step1:custom list value

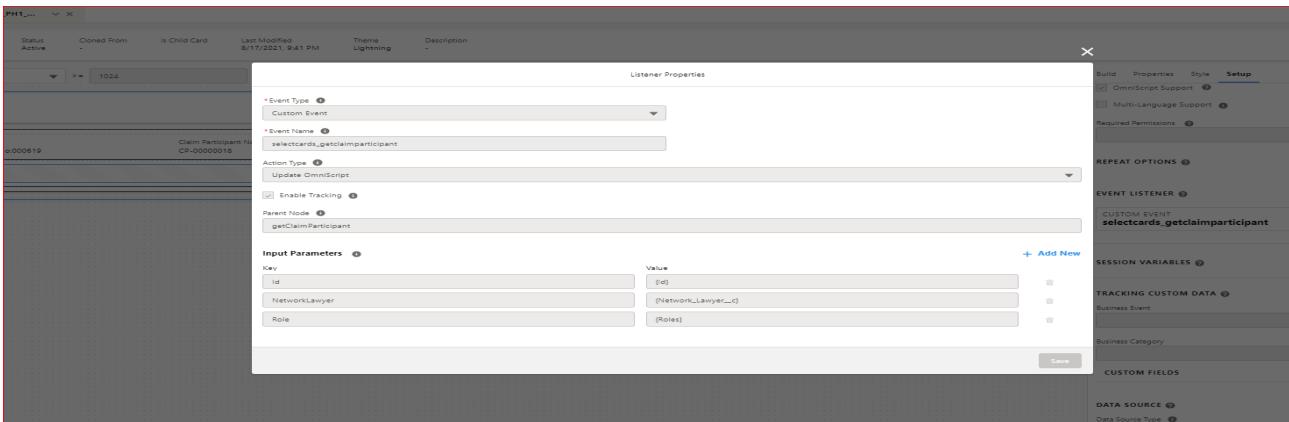
"Step1":{"CustomLWC1":{"Step2":{"Text1":

## Checkbox example:



## Update in card and then fire the event to update omniscrypt :

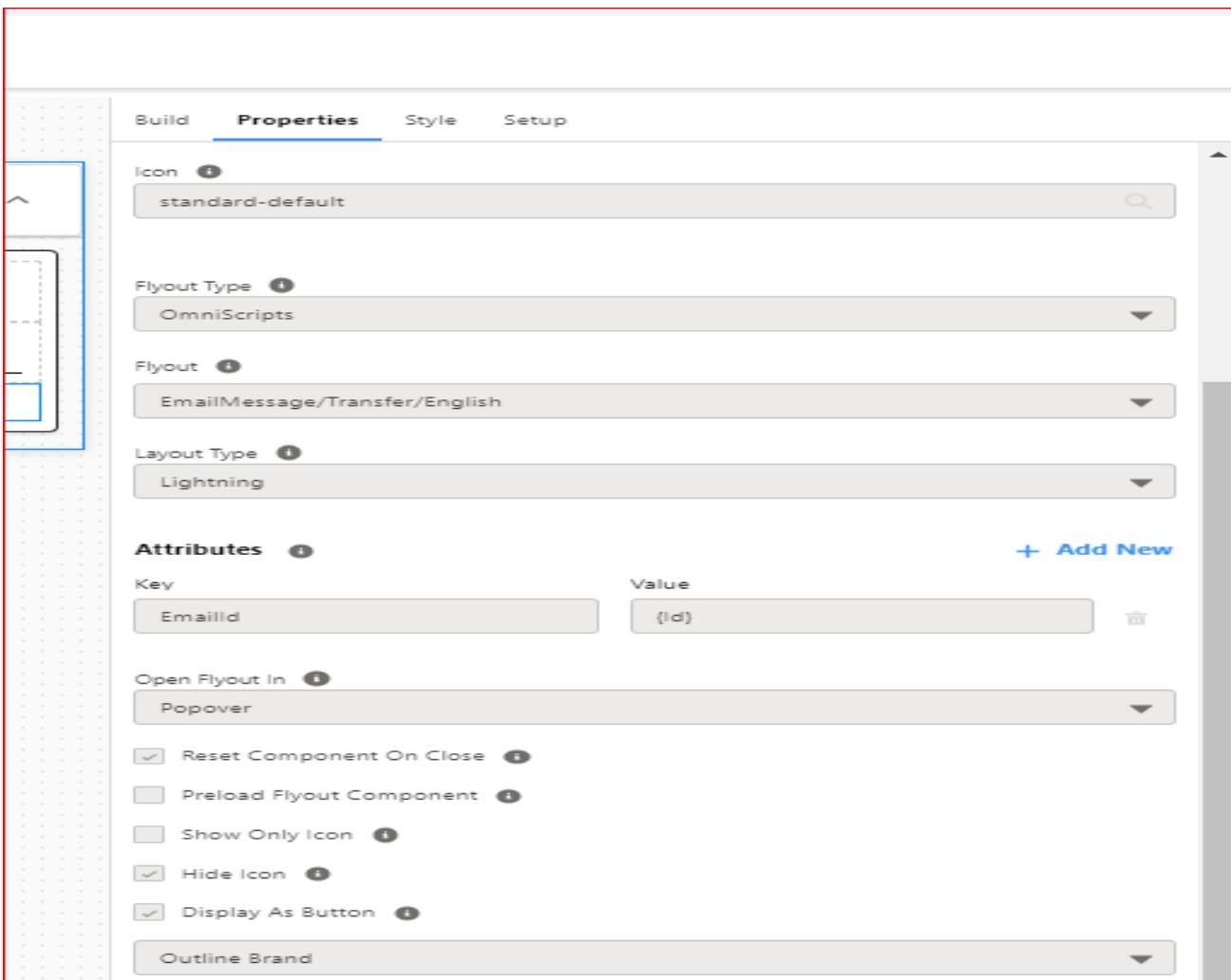




**launch new omniscrypt in flyout or in new tab**

define contextId {Id}

and use that %contextId%



**And use EmailId in wherever u want (Omniscrypt / IP)**

---

---

**Normally : (reverse way)**

FlexCard to OmniScript: **record-id**              {}  
And in omniscrypt:        %record-id%

---

**input in flex card:** for data table use **records**

**In Datatable:** records, {records} {records[0]}

for output field use **key name** it will automatically iterate [{"key":value}{}]

define action item on that and use **{id}** to pass that particular select **record value**

---

**Data Passing :**

---

flex card works in data list format. It means [{} ,{} ,{} ]

custom lwc works in object list format [**getList:[{} ,{} ]**]

---

### **SOME KEY POINTS**

---

First disable your cache logging

To preview flex card in omniscrypt you need to activate it

For any changes you need to recompile it (**deactivate->recompile, recompile in omniscrypt and refresh in button**)

When u define custom lightning web component, it doesn't work in version

Use Only the name

If same result you are getting again and again after disabling and enabling it again

Then turn on your debug mode and clear your storage

---

**Standalone LWC (Check box importance)**

**Else it will be treated as parent child component**

---

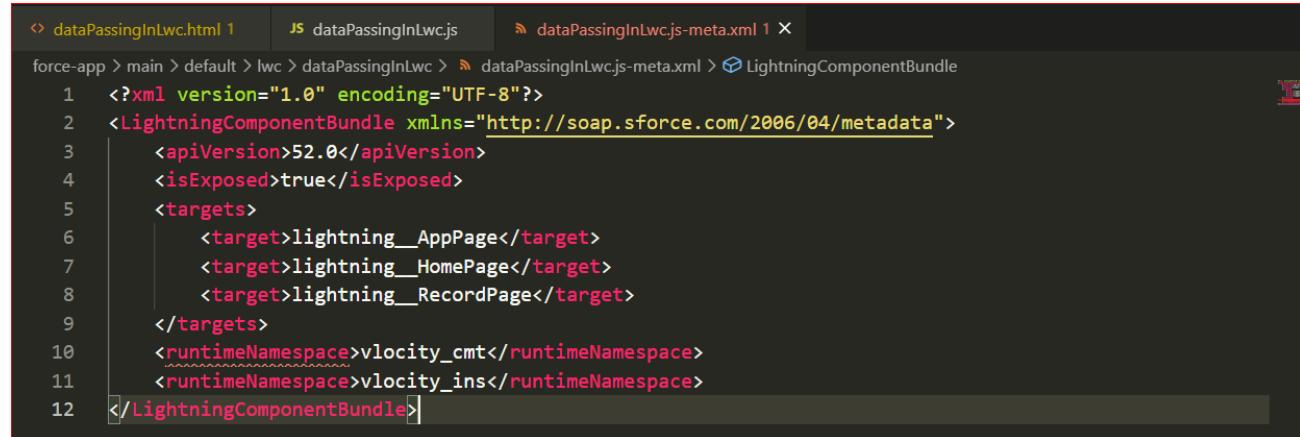
## CUSTOM LWC INSIDE AN OMNISCIPT

Prerequisites:

In metadata:

```
<runtimeNamespace>vlocity_cmt</runtimeNamespace>
<runtimeNamespace>vlocity_ins</runtimeNamespace>
```

Metafile configuration: (run time name space )



```
dataPassingInLwc.html 1 | JS dataPassingInLwc.js | dataPassingInLwc.js-meta.xml 1 X
force-app > main > default > lwc > dataPassingInLwc > dataPassingInLwc.js-meta.xml > LightningComponentBundle
1  <?xml version="1.0" encoding="UTF-8"?>
2  <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3      <apiVersion>52.0</apiVersion>
4      <isExposed>true</isExposed>
5      <targets>
6          <target>lightning__AppPage</target>
7          <target>lightning__HomePage</target>
8          <target>lightning__RecordPage</target>
9      </targets>
10     <runtimeNamespace>vlocity_cmt</runtimeNamespace>
11     <runtimeNamespace>vlocity_ins</runtimeNamespace>
12 </LightningComponentBundle>
```

inJS:

```
import { OmniscriptBaseMixin } from
"vlocity_ins/omniscriptBaseMixin";
```

```
import { getNamespaceDotNotation } from "vlocity_cmt/omniscriptInternalUtils";
import { OmniscriptActionCommonUtil } from "vlocity_cmt/omniscriptActionUtils";
```

Wrapper

```
export default class TestingOffFlexCard extends
OmniscriptBaseMixin(LightningElement) {}
```

## Javascript model: (import, wrapper, api decorator, connected call back method)

```
dataPassingInLwc.js - CustomLwcDataPassing - visual Studio Code
dataPassingInLwc.html 1 JS dataPassingInLwc.js 1
force-app > main > default > lwc > dataPassingInLwc > JS dataPassingInLwc.js > DataPassingInLwc
1 import { LightningElement, track, api } from 'lwc';
2 import { OmniscriptBaseMixin } from "vlocity_ins/omniscriptBaseMixin";
3 //import { getNamespaceDotNotation } from "vlocity_cmt/omniscriptInternalUtils";
4 //import { OmniscriptActionCommonUtil } from "vlocity_cmt/omniscriptActionUtils";
5
6 const columns = [
7     { label: 'Contact Name', fieldName: 'Name' },
8     { label: 'Contact Email', fieldName: 'Email' },
9     { label: 'Contact Phone', fieldName: 'Phone', type: 'phone' },
10];
11
12 export default class DataPassingInLwc extends OmniscriptBaseMixin(LightningElement) {
13     @api records;
14     columns = columns;
15     data = [];
16     connectedCallback() {
17         this.data = this.records;
18         console.log("All contact Record" + this.records);
19     }
20
21     handleClick() {
22         alert("This is great!!!");
23     }
24 }
25
```

## HTML Model:

```
dataPassingInLwc.html - CustomLwcDataPassing - visual Studio Code
dataPassingInLwc.html 1 JS dataPassingInLwc.js 1
force-app > main > default > lwc > dataPassingInLwc > dataPassingInLwc.html > template
1 <template>
2     <h1>Hello this is testing of custom LWC</h1>
3     <div style="height: 300px;">
4         <lightning-datatable key-field="id" data={data} columns={columns}>
5             </lightning-datatable>
6         </div>
7         <lightning-button variant="brand-outline" label="Show Alert" title="Primary action with lighter lo
8             onclick={handleClick} class="slds-m-left_x-small"></lightning-button>
9
10    </template>
```

## How to pass data from OS screen to Custom LWC?

As i said, it works in object list format so

The screenshot shows the 'Properties' tab of the 'Set Values Properties' page. At the top, there is a 'Name' field containing 'SetValues1' and a 'Field Label' field also containing 'SetValues1'. Below these, the 'Element Value Map' section contains one entry: 'SetValues1' with 'Element Name' 'SetValues1', 'Element Type' 'Set Values', and 'Value' '%GetContactList%'. There is a 'Delete' button next to this entry. Below the map, there is a '+ Add New Element Value' button. The page also includes sections for 'MESSAGING FRAMEWORK' and 'CONDITIONAL VIEW'. At the bottom right, there is a timestamp '16:17 09-08-2021'.

The screenshot shows the 'Properties' tab of the 'Custom Lightning Web Component Properties' page. At the top, there is a 'Name' field containing 'CustomLWC2' and a 'Field Label' field also containing 'CustomLWC2'. Below these, there is a 'Lightning Web Component Name' field containing 'testingOffFlexCard'. There is a checkbox for 'Standalone LWC' which is unchecked. The page includes a 'Property Name' field with 'records' and a 'Property Source' field with '96SetValues1%'. Below these, there is a '+ Add New Property' button. The page also includes sections for 'CUSTOM LIGHTNING WEB COMPONENT PROPERTIES' and 'CONDITIONAL VIEW'. At the bottom right, there is a timestamp '16:17 09-08-2021'.

**Please Note that it does not work When we are having one record because it comes in object format and data in datatable performed in ArrayList format.  
So entire structure would be like:-**

You can pass entire JSON data as well and processing of that.

```
@api _omniJsonData
@api _customers;
connectedCallback() {
    var arr = [];
    this._omniJsonData=this.omniJsonData;
    this._customers = this._omniJsonData.result;
}
```

Processing of data with a particular node:-

```
ce-app > main > default > lwc > testingOffFlexCard > 55 testingOffFlexCard.js > ↗ testingOffFlexCard > ⌂
1 import { LightningElement, track, api } from 'lwc';
2 import { OmniscriptBaseMixin } from "vlocity_ins/omniscriptBaseMix
3 const columns = [
4     { label: 'Contact Name', fieldName: 'Name' },
5     { label: 'Contact Email', fieldName: 'Email' },
6     { label: 'Contact Phone', fieldName: 'Phone', type: 'phone' },
7 ];
8 export default class TestingOffFlexCard extends OmniscriptBaseMixin
9     columns = columns;
10    data = [];
11    @track areDetailsVisible = true;
12    @api records
13    connectedCallback() {
14        console.log("this.cryptoName", this.records);
15    }
16 }
```

```

}];*/
export default class BE_PH2_BrokerContactList extends OmniscriptBaseMixin(LightningElement) {
    columns = columns;
    //data = data;
    @api records;
    data = [];

    connectedCallback() {
        this.data = this.records;
        console.log("All contact Record" + this.records);
    }
    handleRowSelection = event => {
        var el = this.template.querySelector('lightning-datatable');
        var selected = el.getSelectedRows()[0];
        console.log(JSON.parse(JSON.stringify(selected)));
        this.omniApplyCallResp({
            "selectedRow": selected
        })
    }
}

```

### **Remember to assign records in connected callback method**

#### **Processing of data with a custom attribute:-**

Custom attribute passed from os to custom lwc

Property name : Name

Property Source : Source

```

let parms = this.omniJsonDef.propSetMap.customAttributes;
// Find the values in the list (don't want to be fussy about the order)
parms.forEach((val) => {
    if (val.name === "headers") {
        this.parms_headers = val.source;
        console.log('Headers = ' + this.parms_headers);
    }
})

```

**@api directly assign**

**Special case:** How to pass **record** and **column** of data table from os to custom lwc?

**Data :** u pass in normal array format {record}

**CouumnName:-**

**Column name passing in array format from set value**

```
let columns = [
    { label: 'Contact Name', fieldName: 'ContactName' },
    { label: 'Contact Phone', fieldName: 'ContactPhone', type: 'phone' },
    { label: 'Contact Email', fieldName: 'ContactEmail' },
    { label: 'Contact Language', fieldName: 'ContactLanguage' },
];
console.log(JSON.stringify(columns));
column = JSON.stringify(columns);
console.log(JSON.parse(column));
console.log('<----->');
```

**Column name passing in string format**

```
headers=[];
parms_headers = 'Name, Email';
parms_headers.split(',').forEach((item,i)=>{
    var x = {};
    x.label = i;
    x.fieldName = item.trim();
    headers.push(x);
});
console.log('Headers:' + JSON.stringify(headers));
console.log('<----->');

headers=[];
parms_label = 'Contact Name, Contact Phone';
parms_fieldName = 'ContactName, ContactPhone';
array_label = parms_label.split(',');
array_fieldName = parms_fieldName.split(',');
console.log(array_label);
console.log(array_fieldName);

parms_label.split(',').forEach((item,i)=>{
    var x = {};
    x.label =array_label [i].trim();
    x.fieldName = array_fieldName[i].trim();
    headers.push(x);
});
console.log('Headers:' + JSON.stringify(headers));
head= JSON.stringify(headers);
console.log(JSON.parse(head));
```

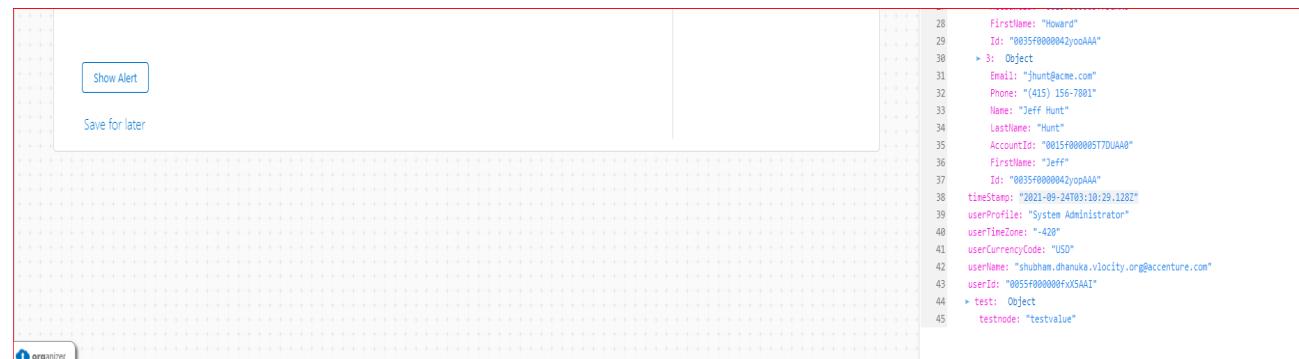
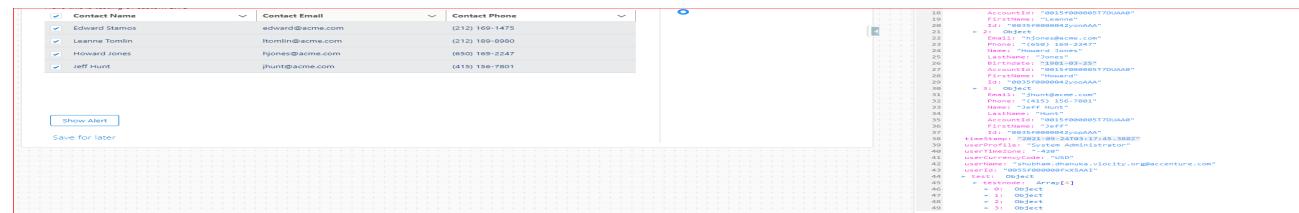
## How to get back data from Custom LWC to OS screen Back?

You need to use this function and in this function you need to pass nodes and data values.

```
this.omniApplyCallResp()
```

```
    handleClick() {
        alert("This is great!!!");
        this.omniApplyCallResp({
            "test": { "testnode": "testvalue" }
        });
    }
}
```

```
19
20
21     handleClick() {
22         alert("This is great!!!");
23
24         var el = this.template.querySelector('lightning-datatable');
25         console.log(el);
26         var selected = el.getSelectedRows();
27         console.log(selected);
28         this.omniApplyCallResp({
29             "test": { "testnode": selected }
30         });
31     }
32 }
33
```



### Note:

For multiple parameter passing or different attribute passing code is in code learning note

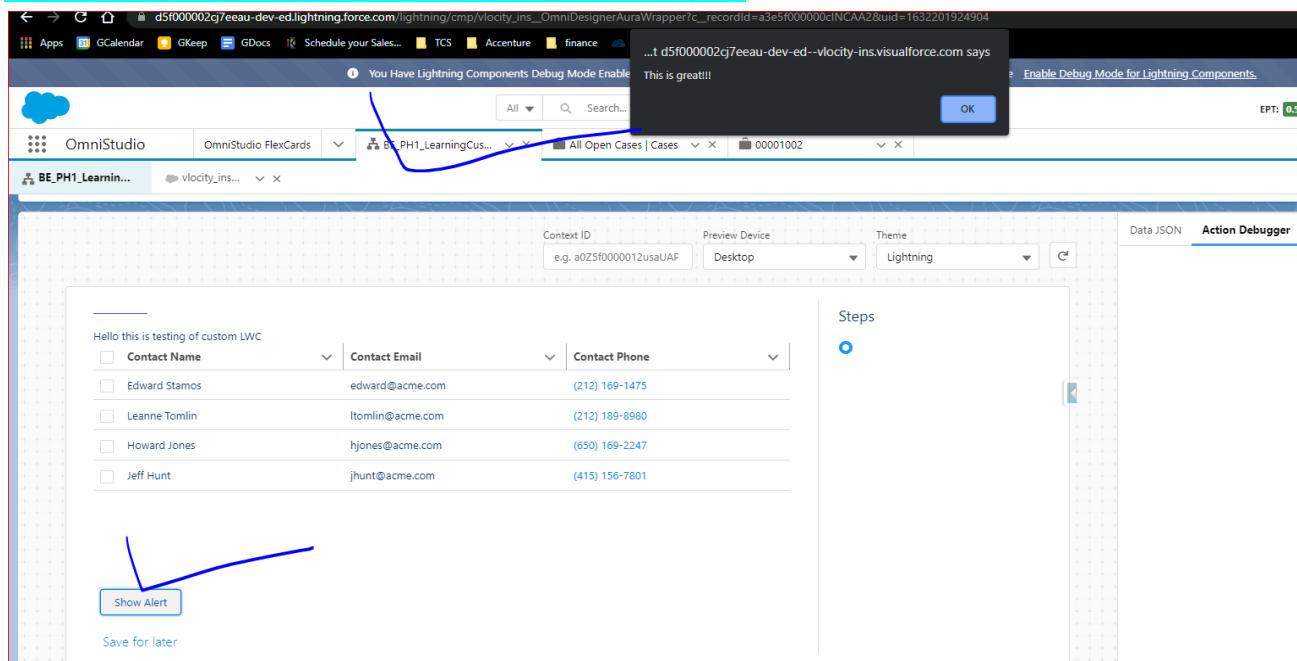
## How to processed data further in OS?

```
export default class BE_PH2_BrokerContactList extends OmniscriptBaseMixin(LightningElement) {
    columns = columns;
    //data = data;
    @api records;
    data = [];

    connectedCallback() {
        this.data = this.records;
        console.log("All contact Record" + this.records);
    }
    handleRowSelection = event => {
        var el = this.template.querySelector('lightning-datatable');
        var selected = el.getSelectedRows()[0];
        console.log(JSON.parse(JSON.stringify(selected)));
        this.omniApplyCallResp({
            "selectedRow": selected
        })
    }
}
```

BE_PH2_CreateQuote	
ClaimHistoryDetails: IsInsuranceCancelled	BE_PH2_Was_Insurance_Cancelled__c
ClaimHistoryDetails: LawyerInterventions	BE_PH2_LawyerIntervention__c
ClaimHistoryDetails: LegalAssistances	BE_PH1_MoreLegalAssistances__c
ClaimHistoryDetails: NumberOfLegalAssistances	BE_PH2_NoOfLegalAssistances__c
ClaimHistoryDetails: PolicyNumber	BE_PH2_Old_Policy_Number__c
ClaimHistoryDetails: ReasonOfDispute	BE_PH1_Nature_of_the_Disputes__c
ClaimHistoryDetails: WasInsuredForLegalAssistance	BE_PH2_IsInsuredForLegalAssistanceInPast__c
ClaimHistoryDetails: whichCompany	BE_PH2_Past_Legal_Assistance_Company__c
ClaimHistoryDetails: WhichCompany	BE_PH2_CompRefused_For_LegalAidCover__c
DRId_Opportunity	OpportunityId
Id	vlocity_ins__AgencyBrokerageId__c
SearchCustomer:TA_SearchCustomer-Block:AccountId	AccountId
selectedRow: ContactId	vlocity_ins__ProducerId__c
	Name

## Button in omniscrypt / toast msg in omniscrypt: Use custom LWC



**More button like navigation work and setting up the value other than any custom.**

**Cancel Button in omniscrypt:-**

In setup of omniscrypt cancel button there to deactivate and navigate

**Update Omniscrypt is not getting refreshed if we deselect it again:**

refresh the event or omniscrypt

**General Note:**

**target config in metadata**

**record design like aura**

**This are reusable property can be used in lightning app builder page layout for reference to pass or value to pass**

## Calling Apex Class in Integration procedure:

### Passing Parameter:

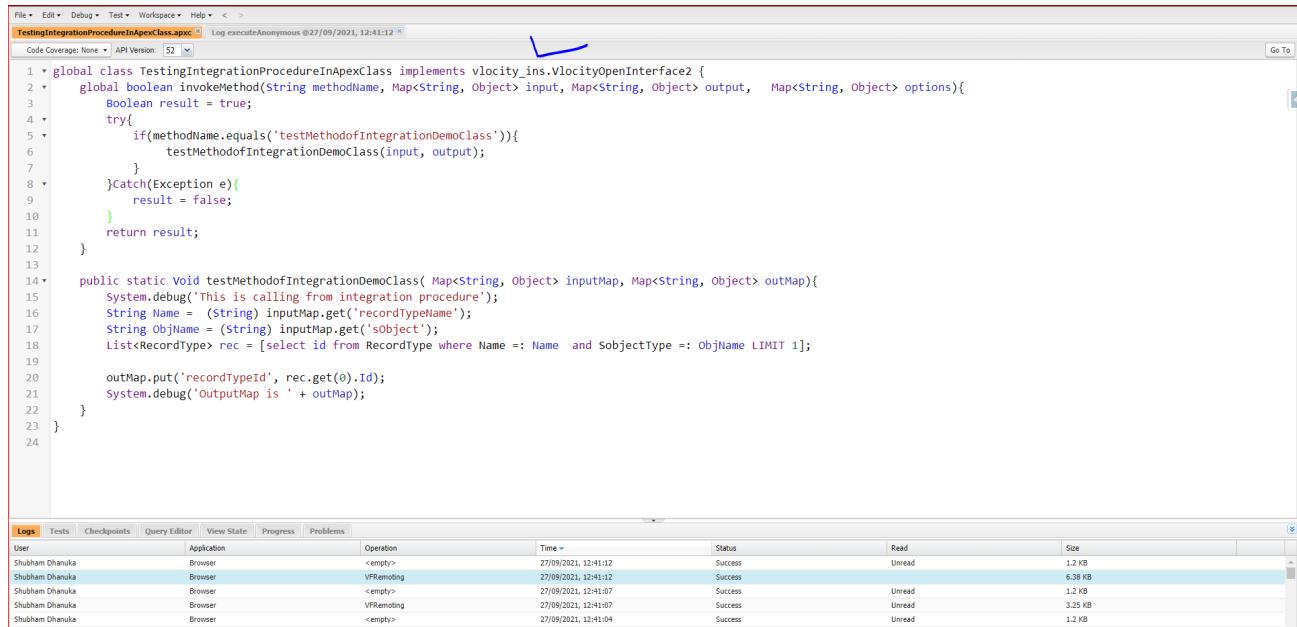
The screenshot shows the 'STRUCTURE' tab of the 'Procedure Configuration' for a 'RemoteAction1' Remote Action. The 'Element Name' is set to 'RemoteAction1'. The 'Remote Class' is 'TestingIntegrationProcedureInApexClass' and the 'Remote Method' is 'testMethodofIntegrationDemoClass'. Under 'Remote Options', there is a section for 'SEND / RESPONSE TRANSFORMATIONS' with fields for 'Send JSON Path' and 'Response JSON Path'. Below this is a section for 'ADDITIONAL INPUT/OUTPUT/FAILURE RESPONSE' where 'Send Only Additional Input' is checked. It includes fields for 'Additional Input' with dropdowns for 'recordTypeName' (set to 'Pharmacy') and 'sObject' (set to 'Account'). There are also sections for 'Additional Output' and 'Failure Response'.

### Receiving output:

The screenshot shows the 'Errors/Debug Output' panel with the 'Debug Log' section expanded. A dropdown menu shows 'RemoteAction1'. The log output displays the following JSON object:

```
{  
    "recordTypeId": "0125f0000001C5anAAC",  
    "errorCode": "INVOKE-200",  
    "error": "OK"  
}
```

## ApexClassHandling:



The screenshot shows the Salesforce IDE interface. The top bar includes File, Edit, Debug, Test, Workspace, Help, and a dropdown for Code Coverage (None) and API Version (52). Below the bar is the code editor for 'TestingIntegrationProcedureInApexClass.apc'. The code implements the vvelocity\_ins.VlocityOpenInterface2 interface. It contains a try-catch block for a specific method name and a debug statement. The logs panel at the bottom shows five entries from the user 'Shubham Dhanuka' in a browser, all marked as unread.

```
1 * global class TestingIntegrationProcedureInApexClass implements vvelocity_ins.VlocityOpenInterface2 {
2     global boolean invokeMethod(String methodName, Map<String, Object> input, Map<String, Object> output, Map<String, Object> options){
3         Boolean result = true;
4         try{
5             if(methodName.equals('testMethodofIntegrationDemoClass')){
6                 testMethodofIntegrationDemoClass(input, output);
7             }
8         }catch(Exception e){
9             result = false;
10        }
11    return result;
12 }
13
14 public static void testMethodofIntegrationDemoClass( Map<String, Object> inputMap, Map<String, Object> outMap){
15     System.debug('This is calling from integration procedure');
16     String Name = (String) inputMap.get('recordTypeName');
17     String ObjName = (String) inputMap.get('sObject');
18     List<RecordType> rec = [select id from RecordType where Name =: Name and SobjectType =: ObjName LIMIT 1];
19
20     outMap.put('recordTypeId', rec.get(0).Id);
21     System.debug('OutputMap is ' + outMap);
22 }
23 }
```

User	Application	Operation	Time	Status	Read	Size
Shubham Dhanuka	Browser	<empty>	27/09/2021, 12:41:12	Success	Unread	1.2 KB
Shubham Dhanuka	Browser	VRremote	27/09/2021, 12:41:12	Success	Unread	6.38 KB
Shubham Dhanuka	Browser	<empty>	27/09/2021, 12:41:07	Success	Unread	1.2 KB
Shubham Dhanuka	Browser	VRremote	27/09/2021, 12:41:07	Success	Unread	3.25 KB
Shubham Dhanuka	Browser	<empty>	27/09/2021, 12:41:04	Success	Unread	1.2 KB

### Code:

```
global class TestingIntegrationProcedureInApexClass implements vvelocity_ins.VlocityOpenInterface2 {
    global boolean invokeMethod(String methodName, Map<String, Object> input, Map<String, Object> output,
    Map<String, Object> options){
        Boolean result = true;
        try{
            if(methodName.equals('testMethodofIntegrationDemoClass')){
                testMethodofIntegrationDemoClass(input, output);
            }
        }catch(Exception e){
            result = false;
        }
        return result;
    }
    public static void testMethodofIntegrationDemoClass( Map<String, Object> inputMap, Map<String, Object>
    outMap){
        System.debug('This is calling from integration procedure');

        String Name = (String) inputMap.get('recordTypeName');
        String ObjName = (String) inputMap.get('sObject');
        List<RecordType> rec = [select id from RecordType where Name =: Name and SobjectType =: ObjName
        LIMIT 1];
        outMap.put('recordTypeId', rec.get(0).Id);
        System.debug('OutputMap is ' + outMap);
    }
}
```

**Note: Sample code has also downloaded**

**To use it further output of apex class:**

**Use element name and item**

**Example:**

RemoteAction1:recordTypeld

## General Question resolve: General Hack :: Element Learning ::

### FORMULA in OS:-

It is the same as in the validation rule, U need to convert your value into string format and then compare.

This screenshot shows the configuration of a text element. The fields are as follows:

- Name: Text4
- Field Label: Text4
- Placeholder: (empty)
- Required:
- Default Value: test
- Read-only:
- Mask: (empty)

This screenshot shows the configuration of a conditional view. The expression is:

```
IF(%Step1:Text4% == "test", "YES", "NO")
```

There is also a 'Hide' checkbox.

This screenshot shows the configuration of a view condition for 'Show Element if True' with the formula (Formula1 = NO).

This screenshot shows the configuration of a second view condition for 'Show Element if True' with the formula (Formula1 = YES).

## Formula in DataRaptor:

EXTRACT FORMULAS OUTPUT OPTIONS PREVIEW

Input Parameters [Edit as JSON](#)

Key	Value
test	Yes

[+ Add New Key/Value Pair](#)

Response Performance Metrics - Browser

```
{ "result": true }
```

EXTRACT FORMULAS OUTPUT OPTIONS PREVIEW

1 Formula

```
IF(%test% == "Yes", true, false)
```

⋮

Formula Result Path  
result  Is Disabled

[+ Add Formula](#)

Interface Type	Input Type	Output Type	Required Permission (Optional)	Description
Extract	JSON	JSON		

EXTRACT FORMULAS OUTPUT OPTIONS PREVIEW

EXTRACT JSON PATH  OUTPUT JSON PATH

result	result

## **DataRaptor Awdm checkpoint Learning:**

**Note: it is as easy to manipulate the list like as easy to manipulate with data List. Query**

### **Parent/Child Query:**

Note: u are not able to see in extractor still u can use it.

By using standard way r.field\_Name

### **Related object fetch by passing id:**

name:Id in where clause

### **Doable things:**

Control on Output of data type

Transformation of data

Validation of data

## **Integration Procedure Awdm Checkpoint learning:**

Send response path, Send response node

Receive response path, receive response node

Condition formula

Setback error

### **Array Element:**

How to access particular index element from a list

Here indexing starts from 1

And to access it, nodename |1 (**node\_name| (pipe symbol and index )**)

### **In Flexcard:-**

records (entire database records),

record (current record),

record[0] (iteration record)

### **Card State:**

- 1) Active State :: Can we used when we have data (Active State)
- 2) InActive State :: Can we used when we don't have data (Make sure to check the checkbox (grey out indicator))

**Action. pop over (vlocity action) (button vlocity) => flex card (launch omniscrypt)**

### **Vlocity action**

### **Standard Class:**

- 1) Duplicate Check: standard class ([Find duplicate](#))
- 2) Approval Process: standard class ([Approval](#))
- 3) Notification : standard class ([Messaging.customNotification](#))
- 4) Email: standard class ([Messaging.sendEmail](#))
- 5) Email inbound : ([Messaging.InboundEmailHandler](#))
- 6) standard class ([Process Instance](#))
- 7) Pattern Matching: ([Matcher](#))

### **Object:**

- 1) Task: creating task (field will be created on activity object)
- 2) Note: creating notes
- 3) EmailMessage: creating email  
*(attachment linking will be on content document link)*
- 4) Action Plan : creating action plan template  
*(id will be taken from Action Plan Template Version)*

### **Select Element:**

Suppose before loading anything if we set the value contextId and context id is not there or it wasn't loaded then it will throw an error so how to overcome?

Ans: Use Condition block to check the value first, same we do in template:if block

### **Typehead new :**

So it is kind of label that is used to pass null value if user are not able to find what they are looking into the list

Based on this both the way at frontend and backend we can configure based on condition block

## To find where dataraptor is used ??

```
String dataRaportname = 'TEST_Dataraptor';

for(vlocity_ps_Element_c omniElement : [SELECT
Id,vlocity_ps_OmniScriptId_r.Name,vlocity_ps_OmniScriptId_r.vlocity_ps_IsProcedure_c,vlocity_ps_PropertySet_c FROM vlocity_ps_Element_c WHERE vlocity_ps_Type_c LIKE '%Dataraptor%'])
{
if(omniElement.vlocity_ps_PropertySet_c.contains(dataRaportname))
{
System.debug('-->' + omniElement.vlocity_ps_OmniScriptId_r.Name);
}
}

Form : <vlocity namespace>_Element_C e.g:- vlocity_ins_element_c
```

**And by using idx workbench also we can see where these component are used.**

### Update record iteration without loop getting all the id :

By using the update element in flow, it will do the SOSL Query

Find all Contact records where AccountId Equals {!\$Record.Id}

### Task :

u cant create any custom field on task object, you have to create all those filed on activity object.

**Activity date: is the due date field on task object coming from activity object.**

### Record triggering flow:

Record triggering flow automatically fires when we do updation in record

### AutoLaunched flow:

Autolaunched flow does not fire itself so u need to explicitly call the flow

It can either be a process builder or trigger itself or class.

Since process builder now gonna deprecated so better to use trigger or class and code would be something like this:-

```
Flow.Interview flowInterval =
Flow.Interview.createInterview('BE_PH1_InternalRequestAfterAppProcess',
new Map<String,Internal_request_c>{ 'NewInternalRequestRecord'=>irRecord}) ;
flowInterval.start();
```

## **Security wise model:-**

Field creation and security setting.

Profile

Permission set

Permission set group

Sharing rule

Custom permission :: role identify :: Validation rule

Validation rule

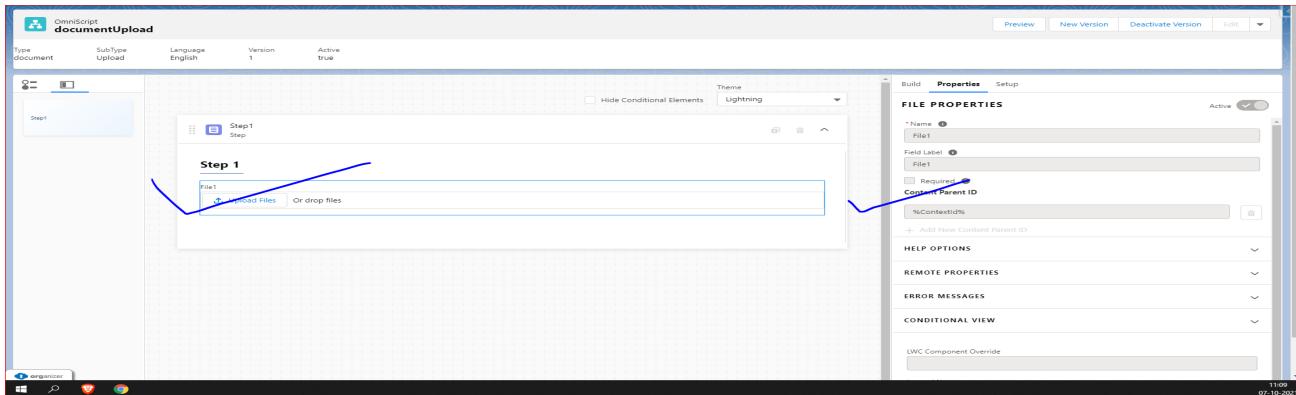
Suspending of sharing rule calculation

**Defer Sharing Calculations: depends org wise**

---

## **Document Attached through OS:-**

U just need to define the input element as a document element and parent id where it will link. It will automatically store a content document link and will be linked to the parent id which is defined.



Case  
Seeking guidance on electrical wiring installation for GC5060

Priority: Low Status: New Case Number: 00001002

Feed Related

- Open Activities (0)
- Activity History (0)
- Case Comments (0)
- Attachments (1)
- Case History (0)

New Task New Event View All New Upload Files

Step 1

File1  
Upload Files Or drop files

Save for later

Hello this is testing of custom LWC

Contact Name Contact Email Contact Phone

## Messaging element in vlocity:-

Messaging element in vlocity is used to show the message on when the condition is met, it is kind of the same to use indicator.

OmniScript  
OmniScriptMessagingElement

Saved an hour ago Preview New Version Activate Version Edit

Type: OmniScriptMessaging SubType: Element Language: English Version: 1 Active: false

Theme: Lightning Hide Conditional Elements

Build Properties Setup

**MESSAGING PROPERTIES**

Name: Messaging1 Field Label: Messaging1 Hide Label:  Messages: true Success: Here u go Active: true false Requirements: Not ready to go Active: true

**VALIDATE EXPRESSION**

Validate Expression:  Edit Validate Expression (Formula1 = Yes)

**CONDITIONAL VIEW**

LWC Component Override

OmniScript  
OmniScriptMessagingElement

Saved an hour ago Design New Version Activate Version Edit

Type: OmniScriptMessaging SubType: Element Language: English Version: 1 Active: false

Context ID: e.g. a0Zf00000012usaJAF Preview Device: Desktop Theme: Lightning

Step 1

Formula1: Yes

Here u go

Save for later

Steps Step 1

**Data JSON**

```

1 -> Object
2   timestamp: "2021-10-11T11:28:43.041Z"
3   userProfile: "System Administrator"
4   userTimeZone: "-420"
5   userCurrencyCode: "USD"
6   userId: "shubham.manuka.vlocity.org@accenture.com"
7   test: "Yes"
8   > Step1: Object
9     Formula1: "Yes"
10    Messaging1: true
11

```

### General Note:-

Queue has to have access of object and person should be there in queue to test

**Check out how many flows are getting fired on that object.**

Any record/field updation:

check it from where it is used?

if package and object content does not have permission then also it will give deployment **insufficient access object** error.

### Database Management:-

Data load activity:-

through **workbench**, through **salesforce inspector** as well (in excel, maps it correctly and then deploy)

The screenshot shows the Bulk API Job Status page in the Salesforce Workbench. The job ID is 7505f000001hWLXAA2. The page displays various metrics such as Status (Closed), Object (Account), Operation (Upsert), External Id (Id), API Processing time (506 ms), and Creation time (07:11:06 AM). It also shows records processed (24), records failed (0), concurrency mode (Parallel), API version (52.0), apex processing time (318 ms), last modified time (07:11:08 AM), and total processing time (933 ms). Below this, a table titled 'Batches' lists one batch with ID 7515f000000r7AAAAY, status Completed, 24 records processed, 0 records failed, created at 07:11:07 AM, and last modified at 07:11:09 AM. A note at the bottom indicates the request was completed in 0.751 sec using Workbench 53.0.0.

### Flow Formula fields:-

The screenshot shows the 'Edit Formula' dialog in the Salesforce Flow builder. The formula is named 'checkFormulaValue'. The data type is set to Boolean. The formula itself is: IF( TEXT({!\$Record.Type})= 'Technology Partner', true, false). There are 'Cancel' and 'Done' buttons at the bottom right.

**decisionMaking** (decisionMaking)

**Outcomes** For each path the flow can take, create an outcome. For each outcome, specify the conditions that must be met for the flow to take that path.

**OUTCOME ORDER** **OUTCOME DETAILS**

**Default Outcome**

\*Label: true \*Outcome API Name: true

Condition Requirements to Execute Outcome: All Conditions Are Met (AND)

Resource: checkFormulaValue Operator: Equals Value: True

+ Add Condition

When to Execute Outcome:

- If the condition requirements are met
- Only if the record that triggered the flow to run is updated to meet the condition requirements

**Done**

## How to pass Ip inside another IP?

Use remote Action and select pass in UserQueueable option as true.

## DataFilling in OS:-

Prepopulating the data on Screen:-

Setting up the context id/ record id:-

**Omniscript** NewFunctionalityLearning

Type: NewFunctionality SubType: Learning Language: English Version: 1 Active: false Description: Saved 3 minutes ago

**caseCategorization**

**Step 1**

PrePopulating the Data

FillTheDataThroughButton

AutoFill

**Properties**

**SET VALUES PROPERTIES**

Name: SetValues1 Field Label: Get Case Details Element Value Map: CaseId Element Name: CaseId Element Type: Value: %ContextId%

**MESSAGING FRAMEWORK**

**CONDITIONAL VIEW**

Fetching up the record:-

**OMNISTUDIO DATARAPTOR BE\_PH1\_CaseRecordFetcher**

Interface Type: Extract Input Type: JSON Output Type: JSON Required Permission (Optional): Description:

**EXTRACT** FORMULAS OUTPUT OPTIONS PREVIEW

1 - Case

\*Extract Output Path: getCaseList Filter: Id paramCaseId

+ Add Extract Step

**Input/Output**

> Input JSON

Extraction Step JSON

```
Show all sObject Fields
{
  "getCaseList": [
    {}
  ]
}
```

## Setting up the output json value with screenmapping:-

**BE\_PH1\_CaseRecordFetcher**

Interface Type: Extract, Input Type: JSON, Output Type: JSON, Required Permission (Optional):

**EXTRACT FORMULAS OUTPUT OPTIONS PREVIEW**

**EXTRACT JSON PATH**  **OUTPUT JSON PATH**

getCaseList:BE_PH1_Broker__r.Name	caseCategorization:Broker
getCaseList:Case__Language__c	caseCategorization:Language
getCaseList:Department__c	caseCategorization:Department
getCaseList:Insurance_Policy__r.Name	caseCategorization:InsurancePolicy
getCaseList:Priority	caseCategorization:Priority
getCaseList:Type	caseCategorization:type

**Input/Output**

- > Extraction Step JSON
- > Expected JSON Output
- < Current JSON Output

```
{
  "caseCategorization": {
    "Broker": "Text",
    "Type": "Text",
    "Priority": "Text",
    "Department": "Text",
    "InsurancePolicy": "Text",
    "Language": "Text"
  }
}
```

**OmniScript NewFunctionalityLearning**

Type: NewFunctionality, SubType: Learning, Language: English, Version: 1, Active: false, Description:

**Build Properties Setup**

**STEP PROPERTIES**

- Name: caseCategorization
- Field Label: Step 1
- Chart Label:
- Instruction:
- Allow Save For Later

**BUTTON PROPERTIES**

**OmniScript NewFunctionalityLearning**

Type: NewFunctionality, SubType: Learning, Language: English, Version: 1, Active: false, Description:

**Build Properties Setup**

**TEXT PROPERTIES**

- Name: Language
- Field Label: Prepopulating the Data
- Placeholder:
- Required
- Read-only
- Default Value:
- Mask:
- Minimum Length: 0
- Maximum Length: 1000

The screenshot shows the 'New Functionality Learning' page. At the top, there's a navigation bar with 'OmniScript' and 'NewFunctionalityLearning'. Below it is a table with columns: Type (Type>NewFunctionality), SubType (Learning), Language (English), Version (1), Active (false), and Description. To the right of the table are buttons for 'Saved a few seconds ago', 'Design', 'New Version', 'Activate Version', and 'Edit'. The main area has a 'Context ID' field containing '5001X0000010dxMQAR' and dropdowns for 'Preview Device' (Desktop) and 'Theme' (Lightning). A yellow box highlights the Context ID field.

**Step 1**

PrePopulating the Data  
French

FillTheDataThroughButton

AutoFill

Save for later

**Steps**

**Step 1**

**Data JSON**

```
1  * Object
2    * ContextId: "5001X0000010dxMQAR"
3    * timestamp: "2022-04-11T02:05:10.207Z"
4    * userProfile: "System Administrator"
5    * userTimeZoneName: "Europe/Paris"
6    * userCurrencyCode: "EUR"
7    * username: "shubham.chanukya@arag.com.dey"
8    * userId: "0051X0000000w5SLQAR"
9    * caseCategory: "Test"
10   * casePriority: "Low"
11   * caseCategorization: Object
12   Priority: "Not Urgent"
13   Importance: "Low"
14   Department: "Op"
15   Type: "Claim"
16   Language: "French"
```

A yellow box highlights the 'Data JSON' section.

## How to fill data in the fields through a button?

Same process like prepopulating just invoke it or set the value when click of a button

The screenshot shows the 'Set Values Properties' page for an element named 'SetValues1'. The 'Element Value Map' section is highlighted with yellow annotations. It contains a table with one row:

Text1	Element Name: Text	Text1
	Element Type: Value:	%caseCategorization.Department%

A blue button labeled 'AutoFill' is located at the bottom of the 'Element Value Map' section. The entire 'Element Value Map' section is outlined with a thick yellow border.

The screenshot shows a Microsoft Power Automate flow titled "caseCategorization". The flow consists of three main steps: "SetValues1" (Set Values), "DataRaptorExtractAction1" (DataRaptor Extract Action), and "caseCategorization" (Step). Below the steps, there is a "PrePopulating the Data" section containing an "AutoFill" trigger for the "FillTheDataThroughButton" action. On the right side, the "TEXT PROPERTIES" pane is open for the "Text1" field, displaying various configuration options like Name, Field Label, Placeholder, Required, Default Value, Mask, Minimum Length, Maximum Length, and Autocomplete. The "REPEAT SETTINGS" section is currently collapsed.

OMNISTUDIO DATARAPTOR  
BE\_PH1\_CaseRecordFetcher

Interface Type	Input Type	Output Type	Required Permission (Optional)	Description
Extract	JSON	JSON		
<b>EXTRACT</b>	<b>FORMULAS</b>	<b>OUTPUT</b>	<b>OPTIONS</b>	<b>PREVIEW</b>
EXTRACT JSON PATH <input type="text" value="getCaseList.BE_PH1_Broker__r.Name"/>		OUTPUT JSON PATH <input type="text" value="caseCategorization:Broker"/>		
getCaseList.Case_Language__c		caseCategorization:Language		
getCaseList.Department__c		caseCategorization:Department		
getCaseList:Insurance_Policy__r.Name		caseCategorization:InsurancePolicy		
getCaseList:Priority		caseCategorization:Priority		
getCaseList>Type		caseCategorization>Type		

**Input/Output**

- > Extraction Step JS<sup>Code</sup>
- > Expected JSON Out<sup>Code</sup>
- ▼ Current JSON Out<sup>Code</sup>

```
{
  "caseCategoriza": {
    "Broker": "Te",
    "Type": "Text",
    "Priority": "High",
    "Department": "InsurancePol",
    "Language": "English"
  }
}
```

The beauty of this, it will work in screen loading as well as setting up the value through button