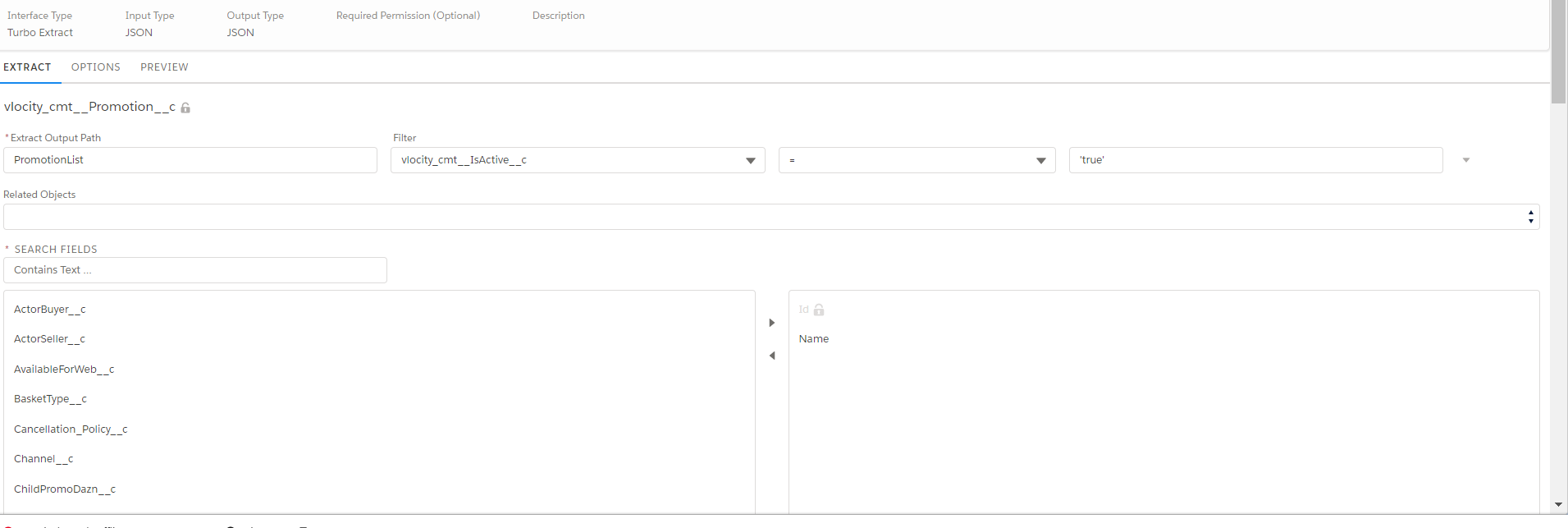
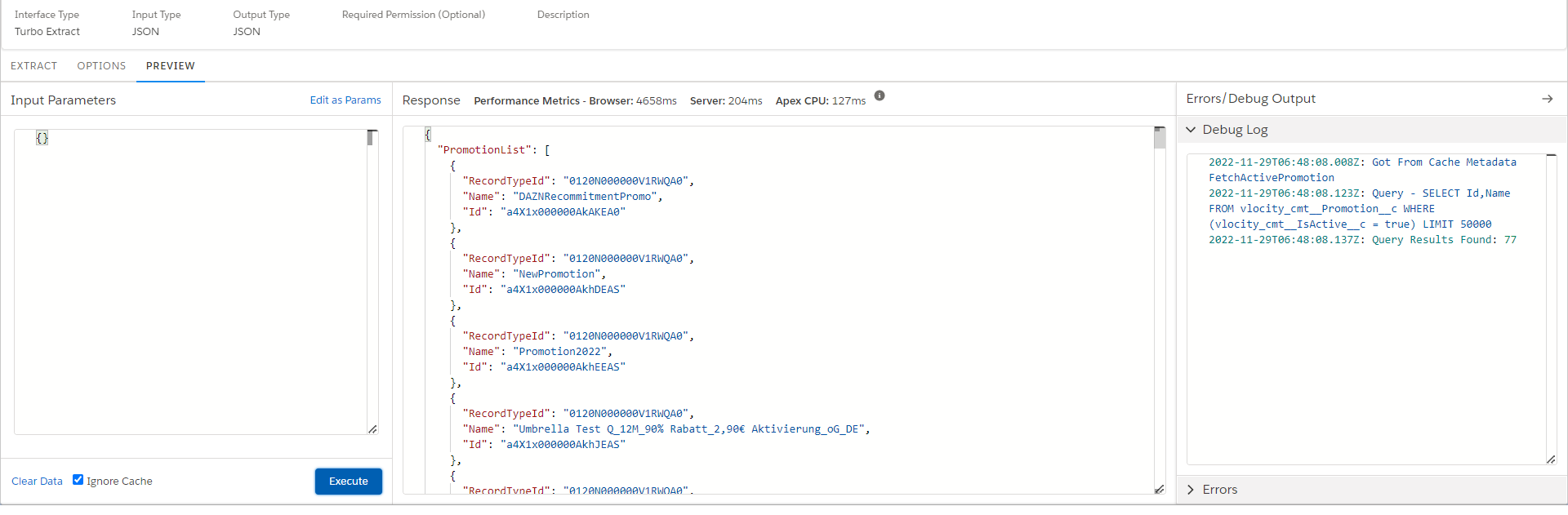
**Fetching the promotion list for LWC Combo Box by utilizing DataRaptor and Integration Procedure**

**Step 1:**

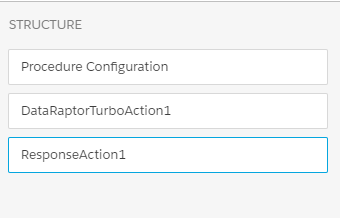
We need to create a DataRaptor Extract/Turbo-Extract which will fetch the promotion list from the org





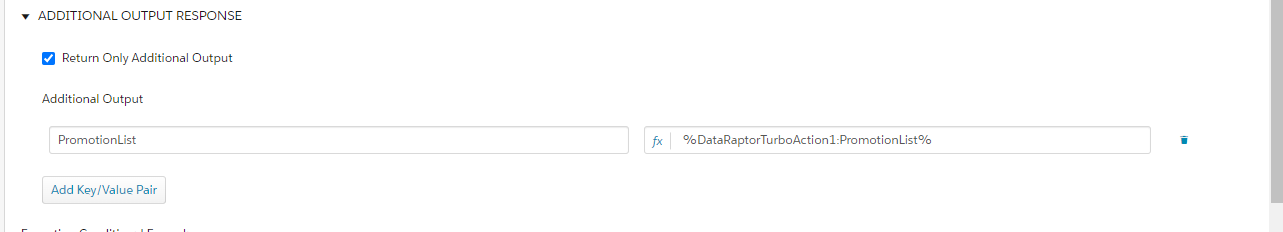
**Step 2:**

We need to create an integration procedure from where we will call the above DataRaptor



The Structurce of VIP will contain the three elements in the respective sequence

1. Procedure Configuration
2. DataRaptor Extract Action / DataRaptor TurboAction
   1. Here we will specify the DataRaptor Name in “DataRaptor Interface”
3. ResponseAction
   1. In Response Action we will specify the path which we want to return



**Step 3:**

We need to create two LWC Elements, one of which will be the parent component (prCombocox) in which we will pass the fetch promotion value and the second will be the child component (prChildComboBox) which will contain the combobox logic

Below is the code snippet for both the LWC

prComboCox.html

<template>

    <div class="slds-grid slds-wrap slds-m-left\_medium slds-m-bottom\_small promoSelect">

        <div

            class="slds-col slds-m-around\_x-small slds-size\_1-of-1 ">

            <c-pr-child-combo-box options={options} selected-value={selectedPromotion} minchar={minchar} isinitialised={isinitialised}

                place-holder="Angebot Auswählen" onselect={handleSelectPromotion}>

            </c-pr-child-combo-box>

        </div>

    </div>

</template>

prComboCox.js-meta.xml

<?xml version="1.0" encoding="UTF-8"?>

<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">

    <apiVersion>52.0</apiVersion>

    <isExposed>true</isExposed>

    <targets>

        <target>lightning\_\_AppPage</target>

        <target>lightning\_\_HomePage</target>

        <target>lightning\_\_RecordPage</target>

    </targets>

    <runtimeNamespace>omnistudio</runtimeNamespace>

</LightningComponentBundle>

prComboCox.js

import { LightningElement, track, api} from 'lwc';

import { OmniscriptBaseMixin } from 'omnistudio/omniscriptBaseMixin';

export default class PrCombocox extends OmniscriptBaseMixin(LightningElement) {

    @api promotionList;

    @api selectedPromotion;

    options;

    minchar = 1;

    isinitialised = true;

    connectedCallback(){

        let tmpStringList;

        console.log('promotionList-->',JSON.stringify(this.promotionList));

        this.options=this.promotionList;

        if (Array.isArray(this.options)) {

            tmpStringList = this.options.map(option => JSON.stringify(option));

        } else {

            tmpStringList = Array.of(this.options).map(option => JSON.stringify(option));

        }

        const tmpSet = new Set(tmpStringList);

        const tmpArray = [...tmpSet];

        this.options = tmpArray.map(option => JSON.parse(option));

        console.log('options-->',this.options);

    }

    handleSelectPromotion(event) {

         this.selectedPromotion = event.detail.value;

    }

}

prChildComboBox.html

<template>

    <div class="slds-combobox\_container">

        <div class="slds-combobox slds-dropdown-trigger slds-dropdown-trigger\_click slds-is-open" aria-expanded="true"

            aria-haspopup="listbox" role="combobox">

            <!-- SEARCH INPUT - START -->

            <div class="slds-combobox\_\_form-element slds-input-has-icon slds-input-has-icon\_right" role="none">

                <lightning-input disabled={disabled} class="inputBox orange-input" placeholder={placeHolder}

                    onblur={blurEvent} onclick={showOptions} onkeyup={filterOptions} value={searchString}

                    autocomplete="off" variant="label-hidden"

                    id="combobox-id-1" data-id="combobox-id-1"

                    onchange={handleChange}>

                </lightning-input>

                <lightning-icon class="slds-input\_\_icon orange-icon" icon-name="utility:down" size="x-small"

                    alternative-text="search"></lightning-icon>

            </div>

            <!-- SEARCH INPUT - END -->

            <!-- DROPDOWN LIST - START -->

            <template if:true={showDropdown}>

                <div id="listbox-id-1" class="slds-dropdown slds-dropdown\_length-5 slds-dropdown\_fluid test"

                    onmousedown={scrolling}>

                    <ul class="slds-listbox slds-listbox\_vertical recordListBox" role="presentation">

                        <template if:false={message}>

                            <template for:each={optionData} for:item="option">

                                <template if:true={option.isVisible}>

                                    <li key={option.value} data-id={option.value} onmousedown={selectItem}

                                        class="slds-listbox\_\_item eachItem">

                                        <span

                                            class="slds-media slds-listbox\_\_option\_entity verticalAlign slds-truncate">

                                            {option.label}

                                        </span>

                                    </li>

                                </template>

                            </template>

                        </template>

                        <template if:true={message}>

                            <li class="slds-listbox\_\_item">

                                <span class="slds-media slds-listbox\_\_option\_entity verticalAlign slds-truncate">

                                    {message}

                                </span>

                            </li>

                        </template>

                    </ul>

                </div>

            </template>

            <!-- DROPDOWN LIST - END -->

        </div>

    </div>

</template>

aaaa

PrChildComboBox.js

import { LightningElement,api,track } from 'lwc';

import { OmniscriptBaseMixin } from 'omnistudio/omniscriptBaseMixin';

export default class PrChildComboBox extends OmniscriptBaseMixin(LightningElement) {

    @api options;

    @api minchar;

    @api selectedValue;

    @api placeHolder;

    @api disabled = false;

    @api isinitialised;

    @track optionData;

    value;

    searchString;

    message;

    showDropdown;

    scrollClick = false;

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    \* LWC LIFECYCLE

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

    connectedCallback() {

        this.showDropdown = false;

        let optionData = this.options ? (JSON.parse(JSON.stringify(this.options))) : null;

        const value = this.selectedValue ? (JSON.parse(JSON.stringify(this.selectedValue))) : null;

        if (value) {

            optionData.forEach(item => {

                if (item.value === value) {

                    this.searchString = item.label;

                }

            });

        }

        this.value = value;

        this.optionData = optionData;

    }

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    \* EVENT HANDLERS

    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

    showOptions() {

        if (this.disabled == false && this.options)  {

            this.message = '';

            //this.searchString = '';

            //this.value = '';

            //this.template.querySelector('.inputBox').value = '';

            let options = JSON.parse(JSON.stringify(this.optionData));

            if (options) {

                options.forEach(item => item.isVisible = true);

                if (options?.length > 0) {

                    this.showDropdown = true;

                }

            }

            this.optionData = options;

        }

        if(!this.isinitialised){

            this.showDropdown = false;

        }

    }

    filterOptions(event) {

        this.searchString = event.target.value;

        console.log(this.minchar+'======searchString==='+this.searchString);

        console.log('======optiondata==='+this.optionData);

        if (this.searchString?.length > 0) {

            this.message = '';

            if (this.searchString.length >= this.minchar) {

                let flag = true;

                if (this.optionData?.length > 0) {

                    this.optionData.forEach(item => {

                        if (item.label.toLowerCase().trim().includes(this.searchString.toLowerCase().trim())) {

                            item.isVisible = true;

                            flag = false;

                        } else {

                            item.isVisible = false;

                        }

                    });

                }

                if (flag) {

                    this.message = `No results found for '${this.searchString}'`;

                }

            }

            if(this.isinitialised || (!this.isinitialised && this.searchString.length >= this.minchar)){

                this.showDropdown = true;

            }

        } else {

            this.showDropdown = false;

            this.dispatchEvent(new CustomEvent('select', {

                detail: {

                    value: this.value,

                }

            }));

        }

    }

    selectItem(event) {

        const selectedVal = event.currentTarget.dataset.id;

        if (selectedVal) {

            const options = JSON.parse(JSON.stringify(this.optionData));

            options.forEach(item => {

                if (item.value === selectedVal) {

                    this.value = item.value;

                    this.searchString = item.label;

                }

            });

            this.optionData = options;

            this.showDropdown = false;

        }

        event.stopPropagation();

    }

    blurEvent() {

        if (this.scrollClick) {

            const inputElement = this.template.querySelector('.inputBox');

            inputElement.focus();

            this.scrollClick = false;

        } else {

            if (this.optionData?.length > 0) {

                this.optionData.forEach(item => {

                    if (item.value === this.value) {

                        this.searchString = item.label;

                    }

                });

            }

            this.showDropdown = false;

            this.dispatchEvent(new CustomEvent('select', {

                detail: {

                    value: this.value,

                }

            }));

        }

    }

    scrolling() {

        this.scrollClick = true;

    }

    handleChange(){

        this.value = '';

    }

}

**Step 4:**

Now we need to create an OmniScript which will contain the above defined lwc and will show the promotion output in the combobox

Below are some snapshots of the configuration

