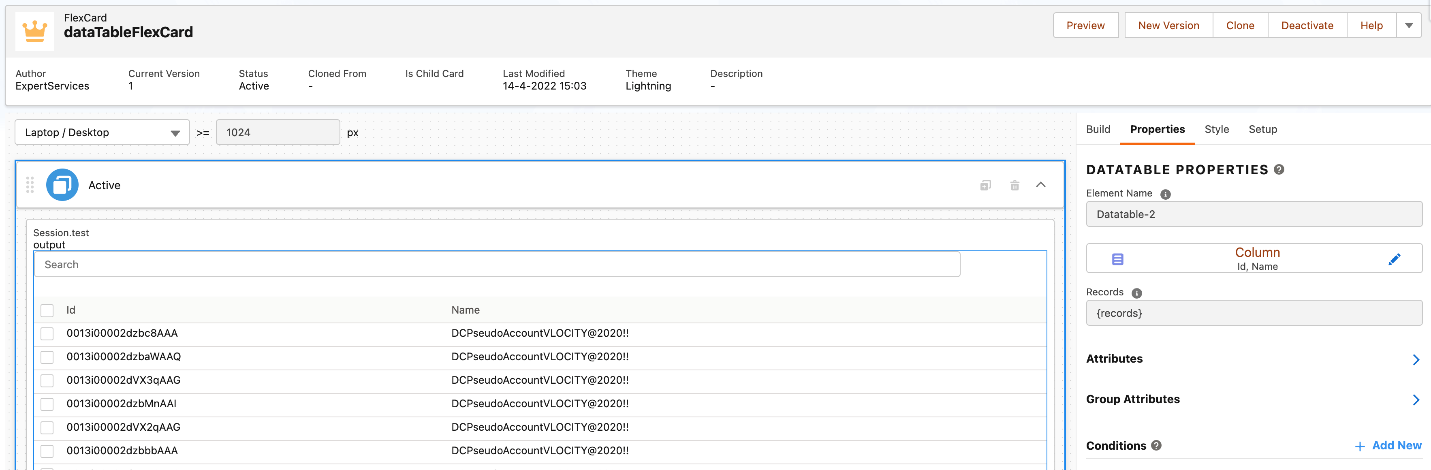
**Embedding a FlexCard Inside OmniScript**

In this example, the FlexCard, which holds the data table element, is embedded into an OmniScript, captures the row selection, and persists within the OmniScript data JSON. Restore the selection when navigating between OmniScript Steps.  
Let’s examine this scenario.

**Step One**

Create a FlexCard that uses the data table element.



**Step Two**

Create a new Lightning Web Component (LWC) to embed the FlexCard and listen to the ***rowselection***event. Here’s an HTML template example.

<template>

<c-cf-data-table-flex-card

parent-data="true"

records={\_records}

onselectrow={getSelectHandler}

></c-cf-data-table-flex-card>

</template>

JS Class receives the initial payload for the FlexCard using a public property and handles the data restoration and selection of rows.

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

import { OmniscriptBaseMixin } from "omniscript/omniscriptBaseMixin";

import { cloneDeep } from "omniscript/lodash";

export default class TmnlWrapperFlexCard extends OmniscriptBaseMixin(

LightningElement

) {

@track \_records;

@track \_omniCustomState;

@track \_omniJsonDef;

@track selectedRecords = [];

UNIQUE\_RECORD\_KEY = "Id";

SORT\_KEY = "Id";

SAVED\_STATE\_KEY = "savedList";

@api

set records(data) {

if (data) {

this.restoreData(data);

}

}

get records() {

return this.\_records;

}

@api

set omniCustomState(data) {

if (data) {

this.\_omniCustomState = data;

}

}

get omniCustomState() {

return this.\_omniCustomState;

}

/\*\*

\* Restore the selection. We need to merge the lists as there is seperate attribute for selected items

\* @param {\*} data

\*/

restoreData(data) {

if (

this.selectedRecords.length === 0 &&

this.\_omniCustomState !== undefined

) {

this.\_records = cloneDeep(data);

this.selectedRecords = cloneDeep(

this.\_omniCustomState[this.SAVED\_STATE\_KEY]

);

let restoredRecordsList = cloneDeep(this.\_omniCustomState.savedList);

this.\_records = this.merge(

this.\_records,

restoredRecordsList,

this.UNIQUE\_RECORD\_KEY

);

this.\_records.sort((a, b) => {

return a[this.SORT\_KEY] - b[this.SORT\_KEY];

});

} else {

this.\_records = cloneDeep(data);

this.\_records.sort((a, b) => {

return a[this.SORT\_KEY] - b[this.SORT\_KEY];

});

}

}

/\*\*

\* Merge two arrays based on unique key

\* @param {\*} a

\* @param {\*} b

\* @param {\*} prop

\* @returns

\*/

merge(a, b, prop) {

let reduced = a.filter(

(aitem) =>

!b.find((bitem) => {

delete bitem.originalIndex;

return aitem[prop] === bitem[prop];

})

);

return reduced.concat(b);

}

/\*\*

\* Handle row selection

\* @param {\*} event

\*/

getSelectHandler(event) {

if (event.detail.result === "all") {

this.selectedRecords = [];

this.selectedRecords = cloneDeep(this.\_assetInfo);

this.selectedRecords.forEach((item) => {

item.selectrow = true;

});

} else if (event.detail.result === "none") {

this.selectedRecords = [];

} else {

let selectedItem = event.detail.result;

delete selectedItem.originalIndex;

if (selectedItem.selectrow === false) {

this.selectedRecords = this.selectedRecords.filter(

(item) =>

item[this.UNIQUE\_RECORD\_KEY] !==

selectedItem[this.UNIQUE\_RECORD\_KEY]

);

} else {

this.selectedRecords.push(selectedItem);

}

}

this.selectedRecords = this.selectedRecords.map((row) => ({

...row,

vlcSelected: true

}));

this.omniApplyCallResp({

selectedRecords: this.selectedRecords

});

this.omniSaveState(this.selectedRecords, this.SAVED\_STATE\_KEY);

}

}

Add the newly created LWC into the OmniScript. ***records***is the public property and passes sample data sets within SetValues before Step 1 of the OmniScript.

