Oracle SaaS Redwood SCM-Procurement pages.

Export 10000+ records to csv from any Redwood page.

PROBLEM NOT FULLY SOLVED YET.

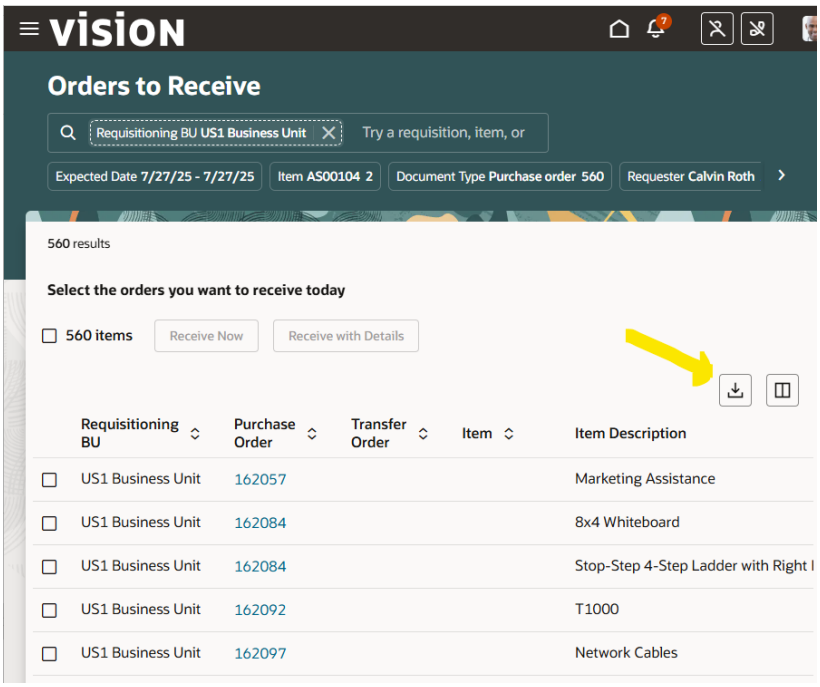
Work in progress…keep reading.

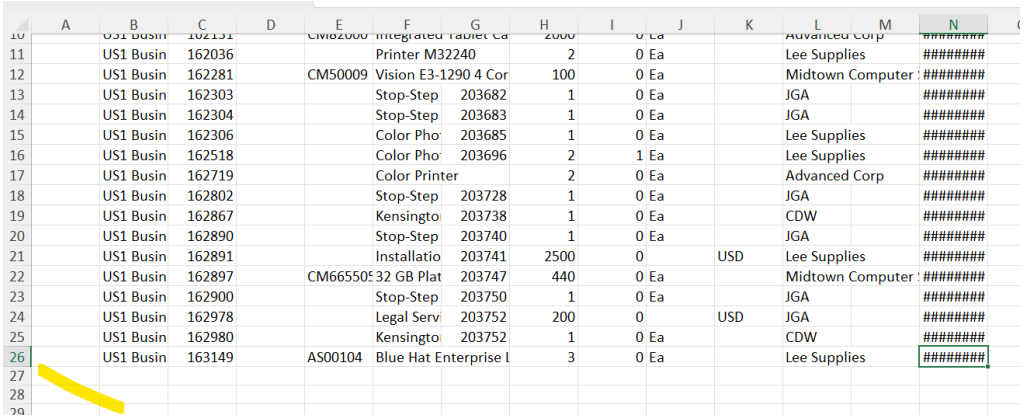
Note: to Visual Builders experts in JavaScript, you will be welcome to take the code and improve performance. Please, post here!

Introduction/Goal.

In Oracle Customer connect we have seen quite a few requirements in several modules (Redwood approach), to export 10000+ records to csv from the following type of pages.

Receiving.

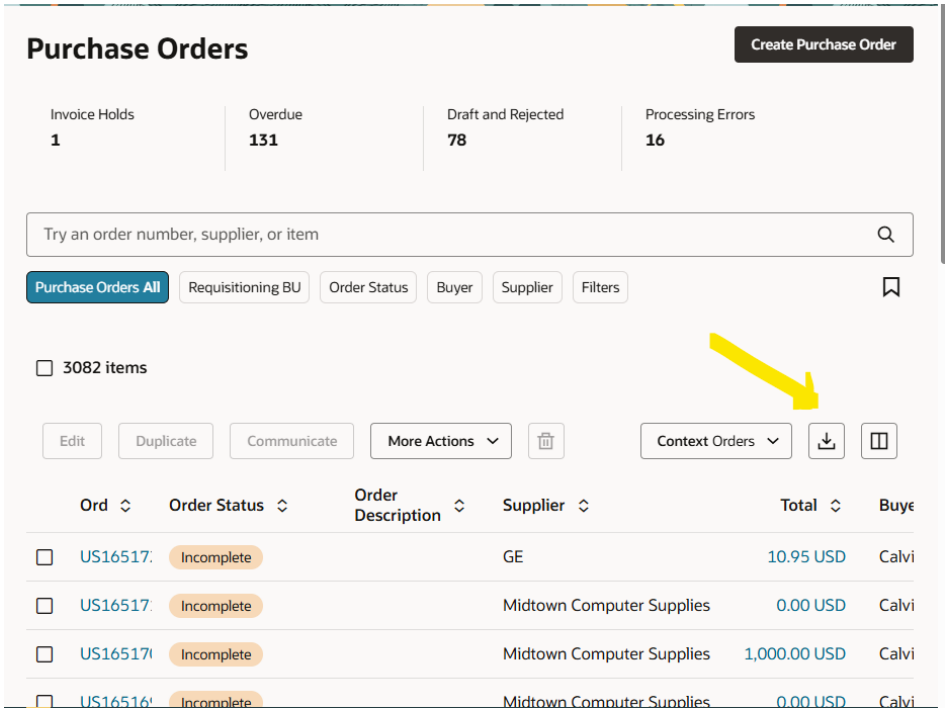




	A	B	C	D	E	F	G	H	I	J	K	L	M	N
11		US1 Busin	162036			Printer M32240		2	0	Ea		Lee Supplies		########
12		US1 Busin	162281		CM50009	Vision E3-1290 4 Cor		100	0	Ea		Midtown Computer		########
13		US1 Busin	162303			Stop-Step	203682	1	0	Ea		JGA		########
14		US1 Busin	162304			Stop-Step	203683	1	0	Ea		JGA		########
15		US1 Busin	162306			Color Pho	203685	1	0	Ea		Lee Supplies		########
16		US1 Busin	162518			Color Pho	203696	2	1	Ea		Lee Supplies		########
17		US1 Busin	162719			Color Printer		2	0	Ea		Advanced Corp		########
18		US1 Busin	162802			Stop-Step	203728	1	0	Ea		JGA		########
19		US1 Busin	162867			Kensingto	203738	1	0	Ea		CDW		########
20		US1 Busin	162890			Stop-Step	203740	1	0	Ea		JGA		########
21		US1 Busin	162891			Installatio	203741	2500	0		USD	Lee Supplies		########
22		US1 Busin	162897		CM665505	32 GB Plat	203747	440	0	Ea		Midtown Computer		########
23		US1 Busin	162900			Stop-Step	203750	1	0	Ea		JGA		########
24		US1 Busin	162978			Legal Servi	203752	200	0		USD	JGA		########
25		US1 Busin	162980			Kensingto	203752	1	0	Ea		CDW		########
26		US1 Busin	163149		AS00104	Blue Hat Enterprise L		3	0	Ea		Lee Supplies		########
27														
28														

Maximum number of records=10.000

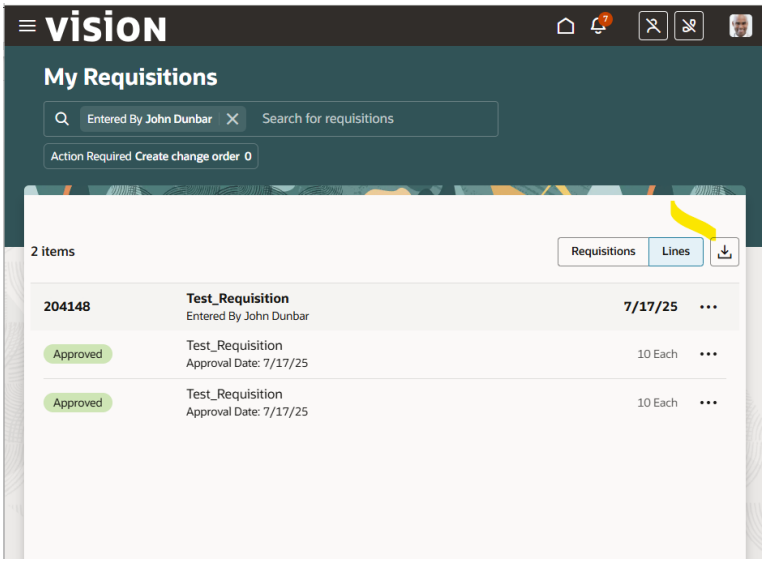
Purchase Orders.



This one takes a while to complete-download.

3075	162034	Closed		Lee Supplies	830.01 USD	Calvin Roth		US1 Business Unit	Calvin Roth
3076	162033	Canceled		Lee Supplies	0.00 USD	Calvin Roth		US1 Business Unit	Calvin Roth
3077	162032	Closed for Receivir		Midtown Compute	85,136.25 USD	Calvin Roth		US1 Business Unit	Calvin Roth
3078	162031	Closed for Receivir		Lee Supplies	349.26 USD	Calvin Roth	Multiple	US1 Business Unit	Calvin Roth
3079	162030	Closed		JGA	219.00 USD	Calvin Roth	203642	US1 Business Unit	Calvin Roth
3080	162029	Closed for Receivir		Dell Inc.	1,335.11 USD	Calvin Roth	203638	US1 Business Unit	Calvin Roth
3081	162028	Closed for Receivir		Lee Supplies	2,897.37 USD	Calvin Roth		US1 Business Unit	Calvin Roth
3082	162027	Closed		Lee Supplies	163.16 USD	Calvin Roth	203637	US1 Business Unit	Calvin Roth
3083	162026	Closed for Receivir		JGA	305.51 USD	Calvin Roth	203636	US1 Business Unit	Calvin Roth
3084									

Purchase requisitions.



We have analyzed the challenge to cover this gap, and we have “failed”, well, **we have succeeded** but the resulting performance is not good.

Sure, there will be other approaches to accomplish the goal, but not this one for now.

Anyway, we have learned so much in the process that we consider it is worth sharing the code and ideas.

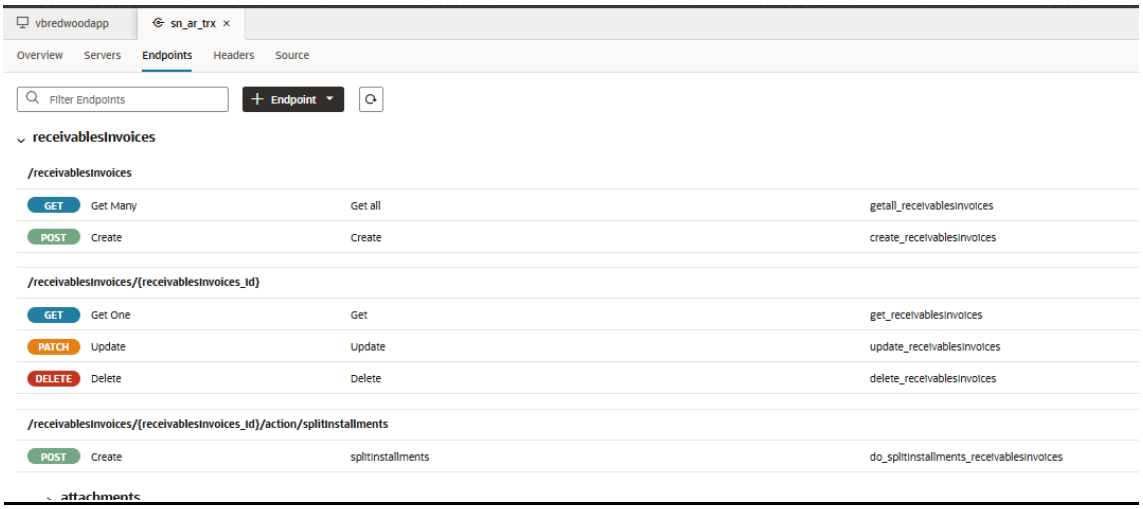
The most important features shown would be:

a) Differences between ADP and SDP in Visual Builder.
b) Some JavaScript code to manage ADP variables.
c) Export component provided by Visual Builder.
d) REST API offset feature.
e) JavaScript function internal to Action Chain.

Preparatory steps

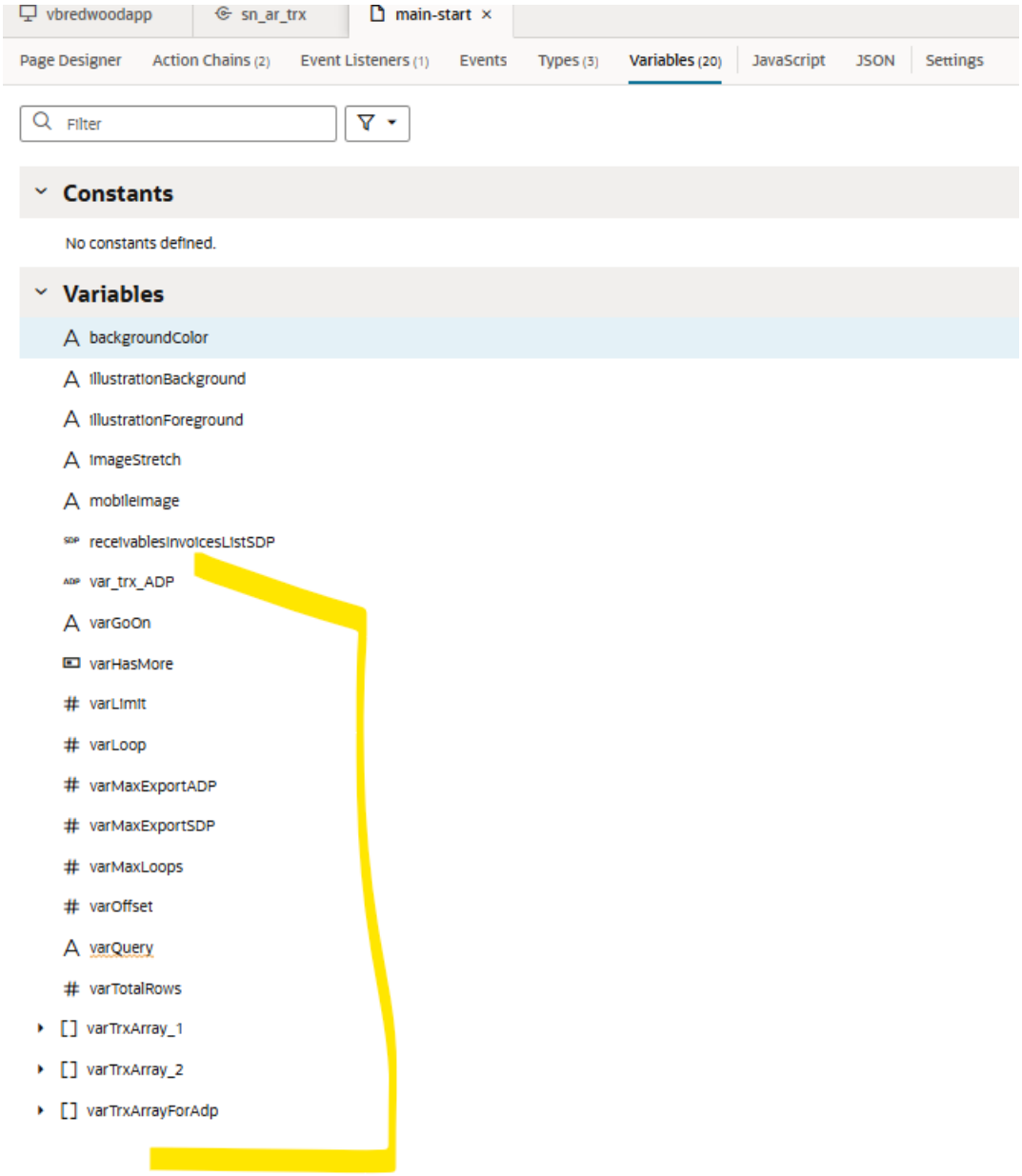
Connection.

We will use the Receivables transactions REST API for our testing.

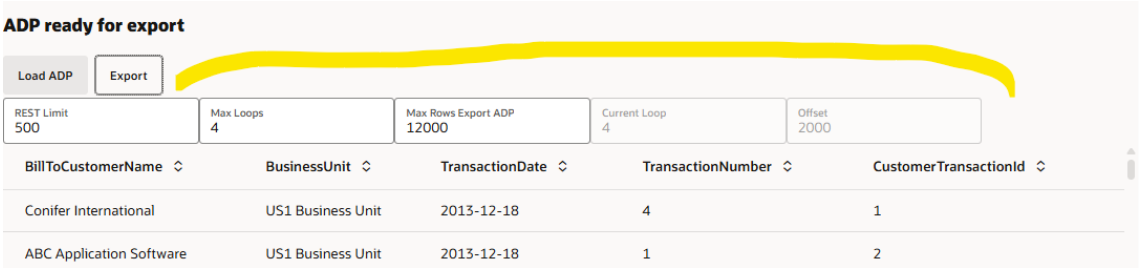


Steps

- Create app. using “Welcome page template”.
- Create these variables.

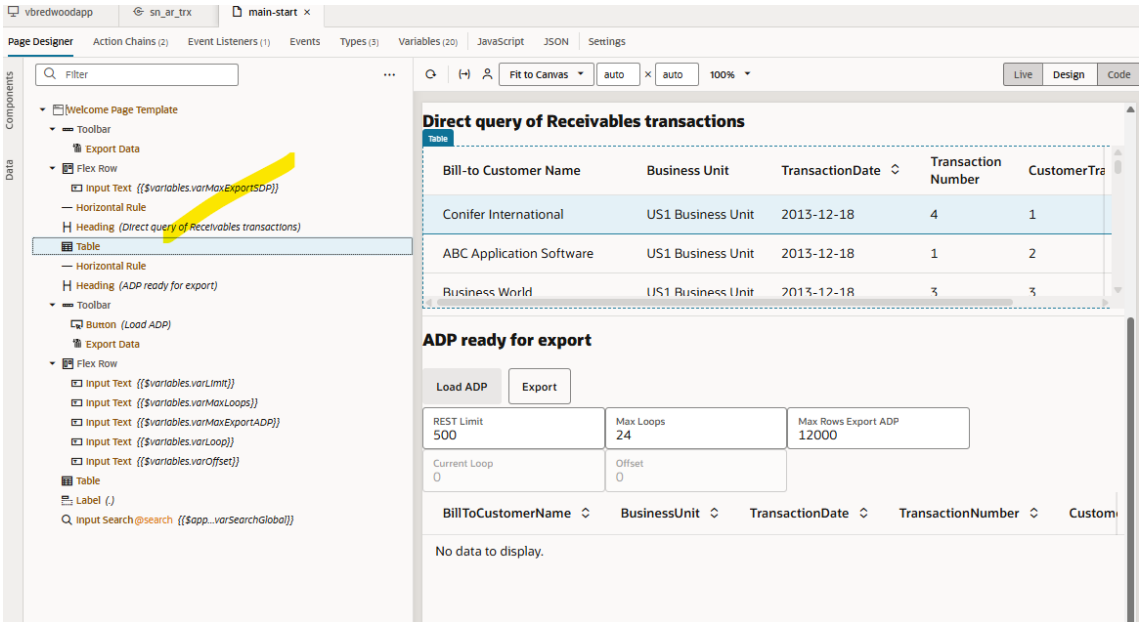


- Create these input components.

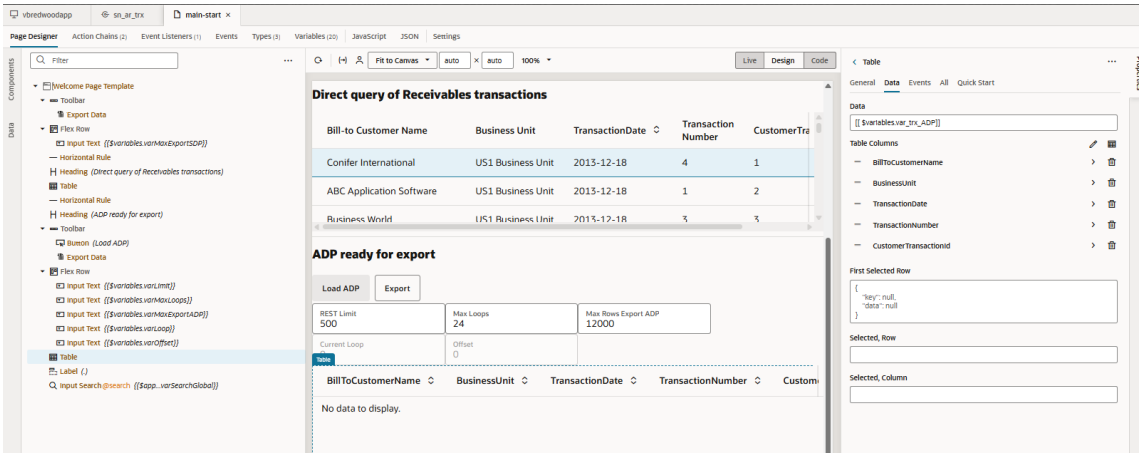


- Create these table components.

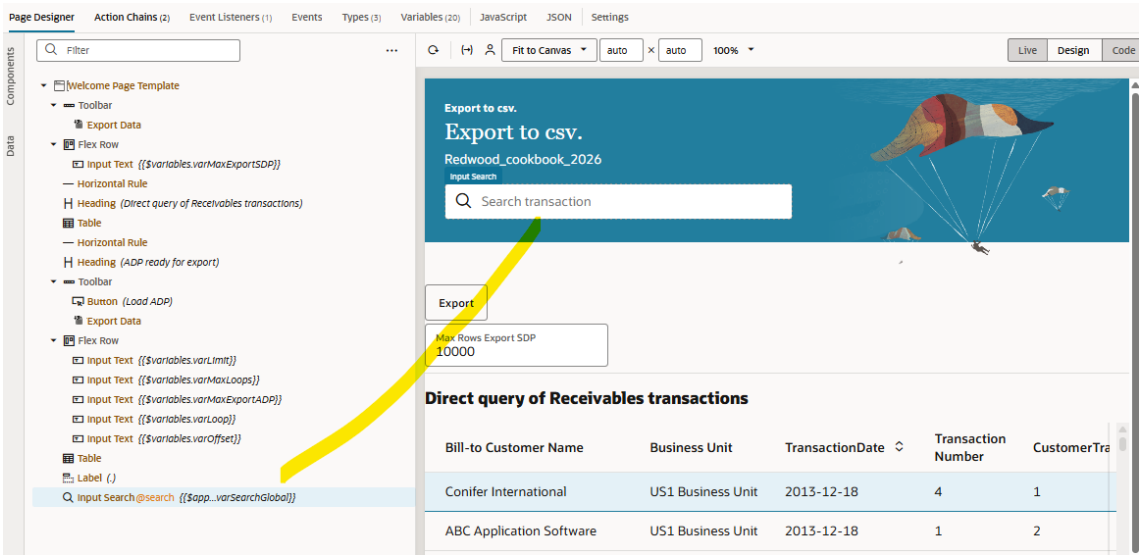
a) For SDP.



b) For ADP.

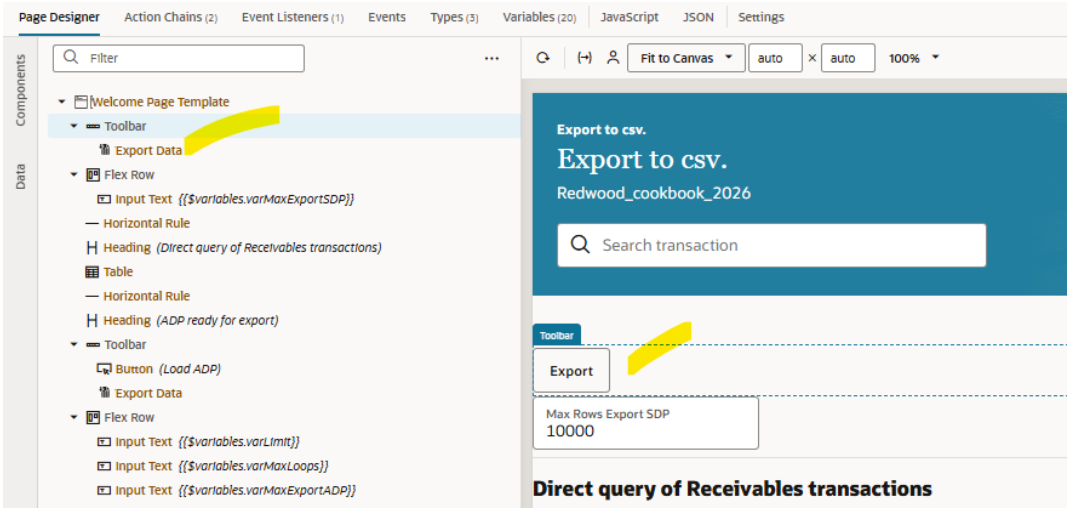


- Create this basic filter.



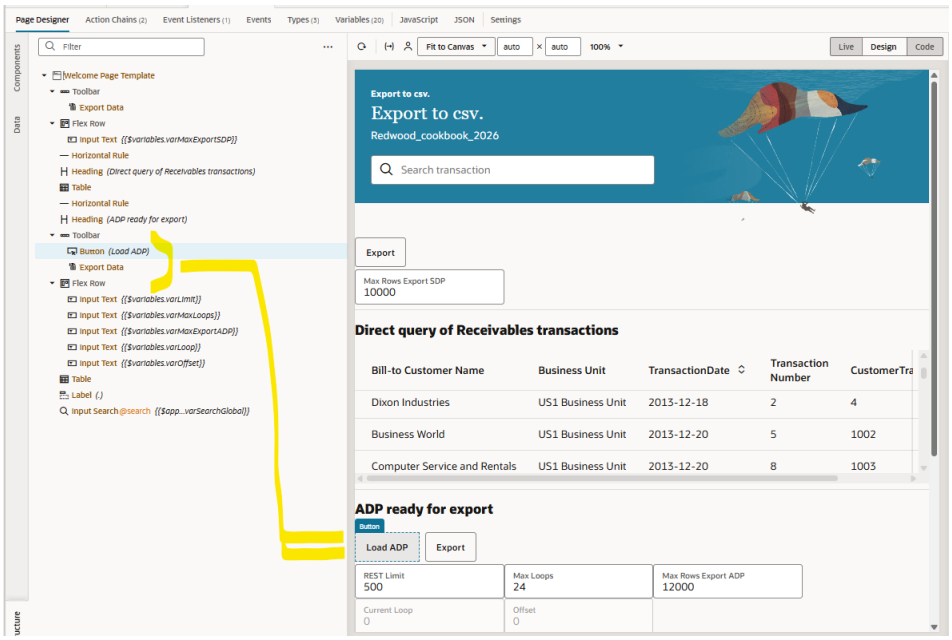
- Create this toolbar 1.

For exporting SDP variable.



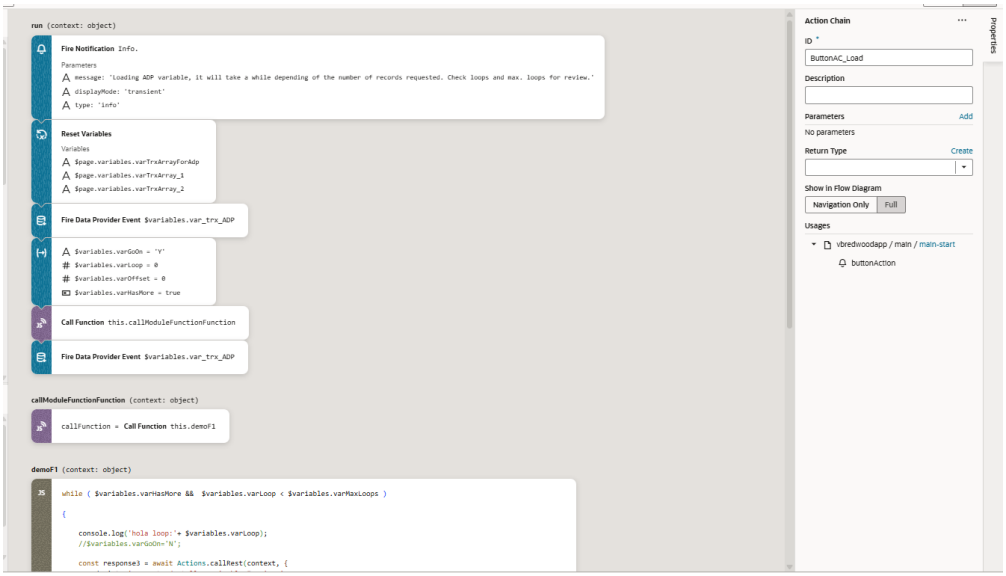
- Create this toolbar 2.

For creating ADP and exporting ADP with JavaScript.



Button Load ADP.

a) Action chain Overview



b) Action chain code.

```
define([
 'vb/action/actionChain',
 'vb/action/actions',
 'vb/action/actionUtils',
], (
 ActionChain,
 Actions,
 ActionUtils
) => {
 'use strict';

 class ButtonAC_Load extends ActionChain {

  /**
   * @param {Object} context
   */
```

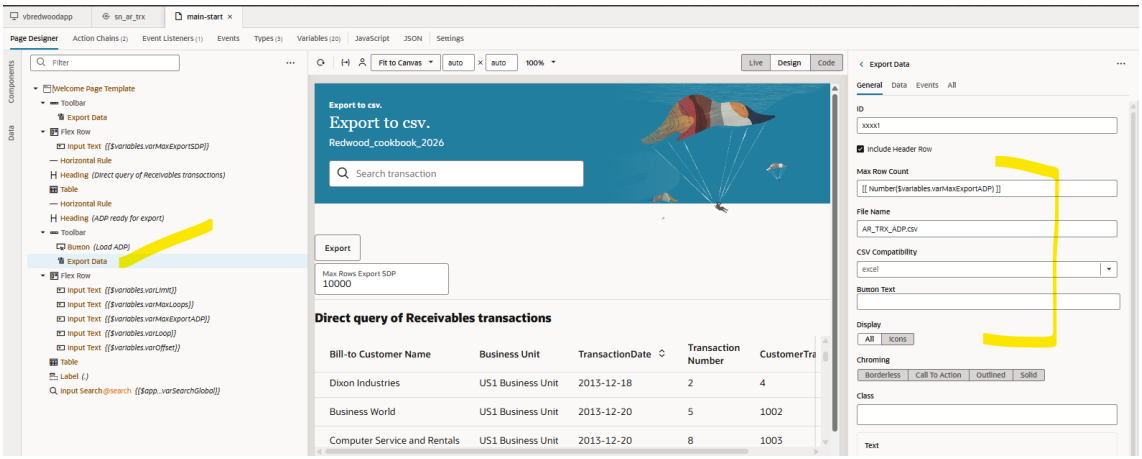
```
  async run(context) {
    const { $page, $flow, $application, $constants, $variables, $functions } = context;

    await Actions.fireNotificationEvent(context, {
      summary: 'Info.',
      message: 'Loading ADP variable, it will take a while depending of the number of records requested. Check loops and max.
loops for review.',
      displayMode: 'transient',
      type: 'info',
    });

    await Actions.resetVariables(context, {
      variables: [
  '$page.variables.varTrxArrayForAdp',
  '$page.variables.varTrxArray_1',
  '$page.variables.varTrxArray_2',
 ],
    });

    await Actions.fireDataProviderEvent(context, {
      target: $variables.var_trx_ADP,
      refresh: null,
    });

    $variables.varGoOn = 'Y';
    $variables.varLoop = 0 ;
    $variables.varOffset = 0 ;
    $variables.varHasMore = true ;

    await this.callModuleFunctionFunction(context);

    await Actions.fireDataProviderEvent(context, {
      target: $variables.var_trx_ADP,
      refresh: null,
    });

  }

  /**
   * @param {Object} context
   */
  async callModuleFunctionFunction(context) {
    const { $page, $flow, $application, $constants, $variables } = context;

    const callFunction = await this.demoF1(context);
  }

  /**
   * @param {Object} context
   */
  async demoF1(context) {
    const { $page, $flow, $application, $constants, $variables } = context;

    while ( $variables.varHasMore &&  $variables.varLoop < $variables.varMaxLoops )

    {

      console.log('demo loop:'+ $variables.varLoop);
      //$variables.varGoOn='N';

      const response3 = await Actions.callRest(context, {
        endpoint: 'sn_ar_trx/getall_receivablesInvoices',
        responseBodyFormat: 'json',
        responseType: 'getallReceivablesInvoicesResponse',
        uriParams: {
          limit: $variables.varLimit,

          orderBy: 'CustomerTransactionId',
          onlyData: true,
          offset: $variables.varOffset,
        },
```

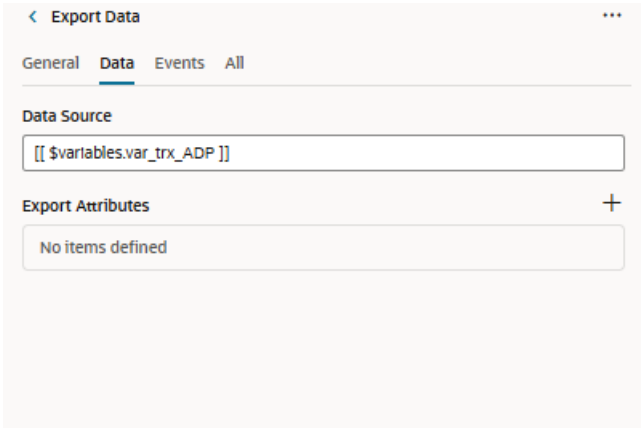
```
        }, { id: 'fn00003' });

        $variables.varTrxArray_2 = response3.body.items;

        const addAdp = await $page.functions.add_adp($variables.varTrxArray_1, $variables.varTrxArray_2);
        $variables.varTrxArray_1 = addAdp;

        console.log('demo zz0:' );

        //$variables.varTotalRows = response3.body.totalResults;

        $variables.varHasMore = response3.body.hasMore;
        console.log('demo 0:'+' hasmore:'+ $variables.varHasMore );
        console.log('demo 2:'+' xx:'+ $variables.varTrxArray_1.length );
        console.log('demo 2:'+' xx:'+ $variables.varTrxArray_2.length );
        console.log('demo 2:'+' xx:'+ $variables.varTrxArrayForAdp.length );

        $variables.varOffset=Number($variables.varOffset)+Number($variables.varLimit);

        $variables.varLoop = $variables.varLoop + 1 ;

      }

      $variables.varTrxArrayForAdp = $variables.varTrxArray_1;

    }

  }

  return ButtonAC_Load;

});
```

c) JavaScript code.

```
define([], () => {
  'use strict';

  class PageModule {
  }

  PageModule.prototype.add_adp = function (totalAdp, newAdp) {

    let varTotalAdp = totalAdp;

    for (let i = 0; i < newAdp.length; i++) {

      let ChargeComponentObj = {};
      ChargeComponentObj.BillToCustomerName = newAdp[i].BillToCustomerName;
      ChargeComponentObj.BusinessUnit = newAdp[i].BusinessUnit;
      ChargeComponentObj.CustomerTransactionId = newAdp[i].CustomerTransactionId;
      ChargeComponentObj.TransactionDate = newAdp[i].TransactionDate;
      ChargeComponentObj.TransactionNumber = newAdp[i].TransactionNumber;

      varTotalAdp.push(ChargeComponentObj);

    }

    return varTotalAdp;

  };

  return PageModule;
});
```

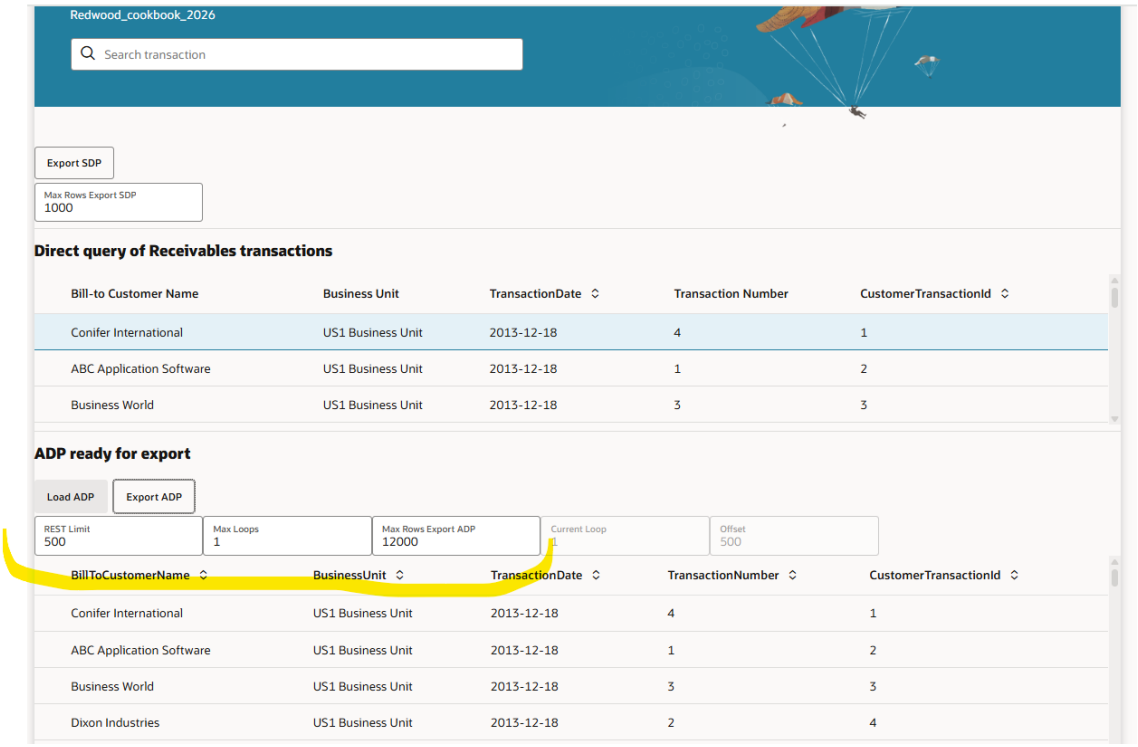
Button Export.





Testing.

Play with these 3 fields for your testing.



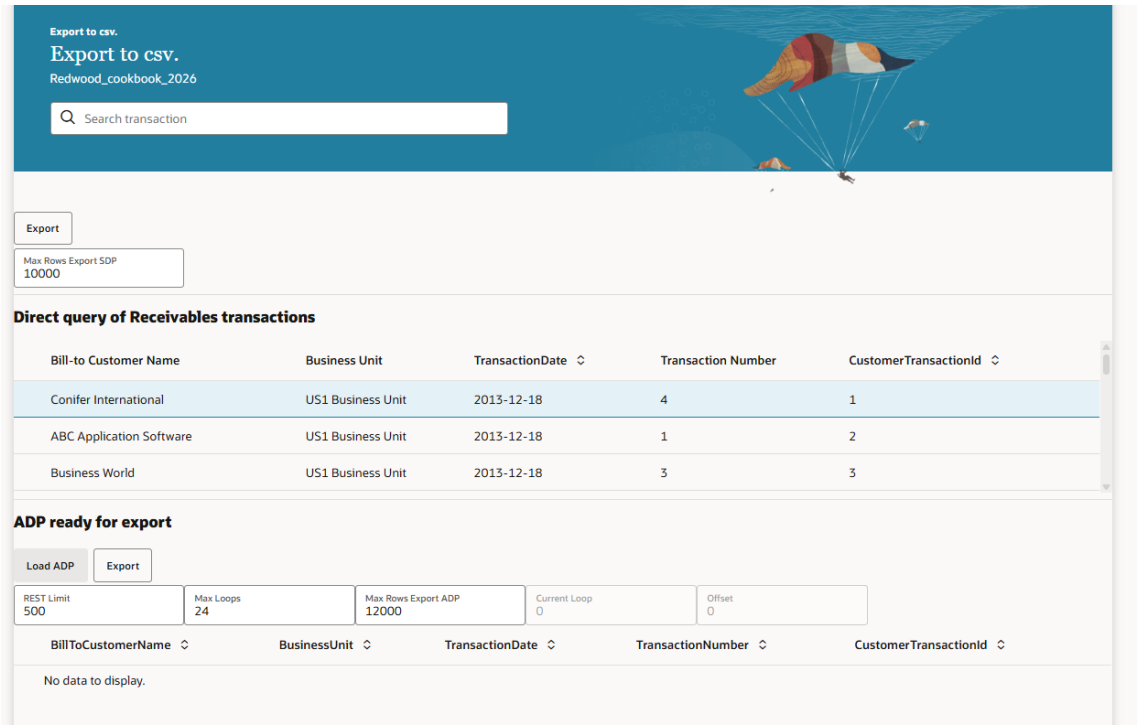
REST Limit means the number of records retrieved in each REST API call.

Max loops means the number of loops you want.

Max Rows Export ADP is the number of records you want really exported. It could be a number smaller than the 2 previous multiplied. Example: 500x3=1500 retrieved, but you could export only 600.

Conclusion

We have achieved the goal of exporting 10.000+ records from Redwood page (BUT…)



But the performance is poor when 10.000 records are requested, so perhaps we should not use this approach.

We have learned a lot in the process.

Technical

Code example	Comments
XX_AR_TRX_20K_EXPORT-1.0.3.zip	Demo code in exercise.