# <Document Title>

# NOTICE

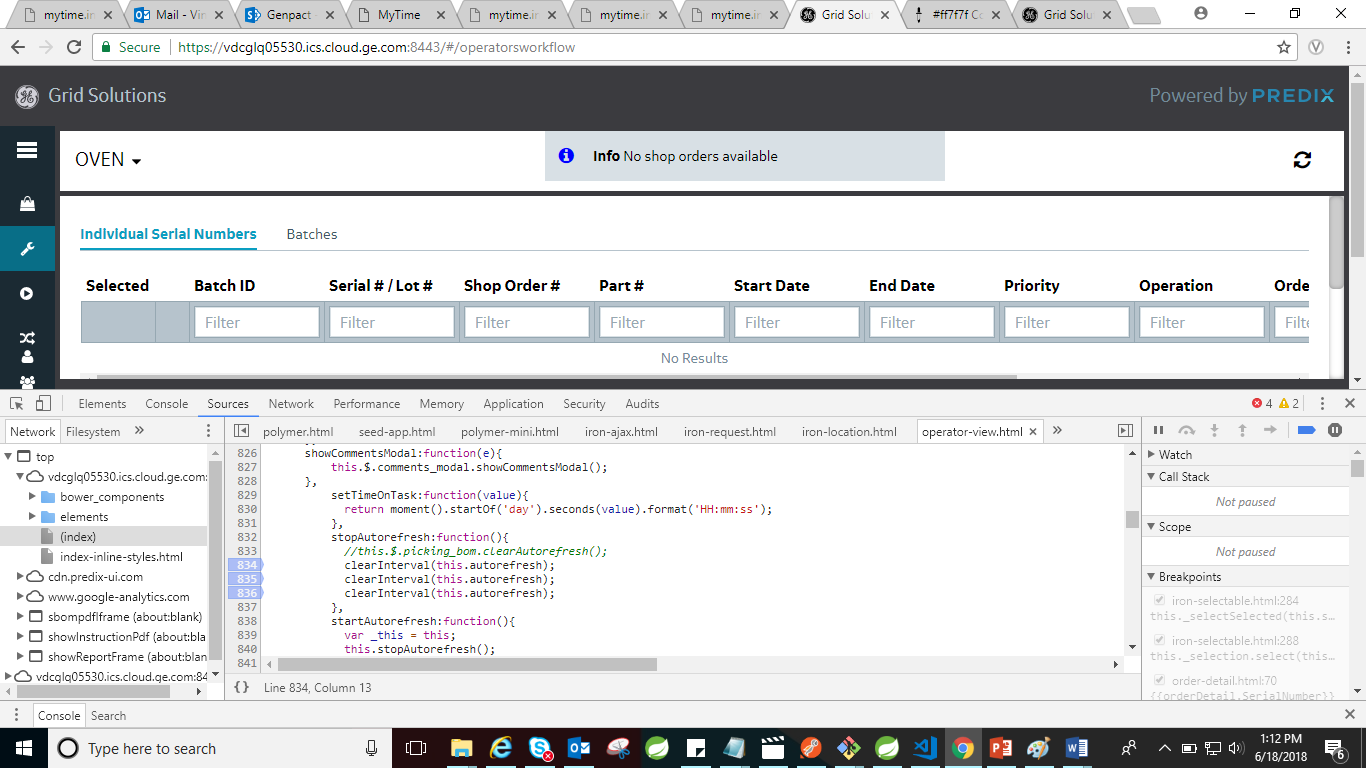
The information contained in this document is not to be used for any purpose other than the purposes for which this document is furnished by GENPACT, nor is this document (in whole or in part) to be reproduced or furnished to third parties or made public without the prior express written permission of GENPACT.

**Version Control**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version No.** | **Version Date** | **Changes** | **Owner/ Author** | **Date of Review/Expiry** |
|  |  |  |  |  |
|  |  |  |  |  |

**Issue:**

On selecting any Plant from OperatorWorkflow tab some times users get data populated and some time its shows modal with information “No Shop orders available”.



On again selecting the same plant the data is loaded over the page.

**Analysis:**

**Flow of application:**

On selecting any plant an event handler (Event: Click) is called

(Line 513 operatorsworkflow view.html)

itemOpenHandler: function(item) {

\_this.goBack = false;

localStorage.removeItem("isNonBatched");

\_this.displayOperatorListData(item);

\_this.$.context\_workcenters.querySelector("px-context-browser-typeahead").clearSearchTerm();

}

This future calls the displayOperatorListData function which internally call’s loadOperatorList function which takes item as parameter which is the plant which we selected object.

Item: {identifier: "650033A7-9E88-440A-B9A5-A484F583030E", name: "OVEN", isOpenable: true, hasChildren: false, selectedAsset: true, …

}

loadOperatorList: function (item) {

console.log(item);

this.isKpiOperatorPresent = false;

document.querySelector("#shopOrderCard").style = "display:block"

this.$.loading.classList.remove("hide");

this.$.loading2.classList.remove("hide");

this.clockActionEnabled = true

this.ItemSelected = item;

this.\_shopOrderData = [];

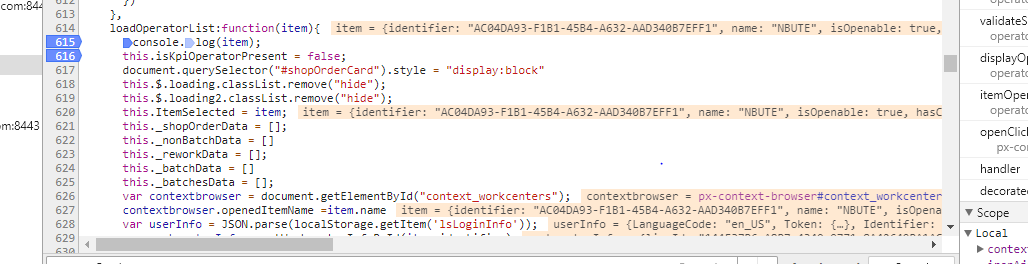
this.\_nonBatchData = []

this.\_reworkData = [];

this.\_batchData = []

this.\_batchesData = [];

this.opListResponseComplete = false



This function is responsible for API calls that are triggered on selecting the plant.

else {

var ironAjax = document.getElementById("requestShopOders");

var params = {

"name": "/Workcenter/Operator",

"namespace": "/ShopOrders",

"paramnames": "workcenter,operator,language",

"paramvalues": item.identifier + "," + userInfo.Identifier + ",en\_US"

}

ironAjax.params = params;

ironAjax.headers = this.getHeaders();

ironAjax.generateRequest();

localStorage.setItem("workcenterParamValues", JSON.stringify(params));

localStorage.removeItem("BatchParamValues");

if (item.details.isBatchingEnabled) {

var ironAjaxBatches = document.getElementById("retrive\_BatchesTab");

var paramsBatches = {

"name": "/Operations/Batch/All",

"namespace": "/ShopOrders",

"paramnames": "workcenter,user",

"paramvalues": item.identifier + "," + userInfo.Identifier

}

ironAjaxBatches.params = paramsBatches;

ironAjaxBatches.headers = this.getHeaders();

ironAjaxBatches.generateRequest();

localStorage.setItem("BatchParamValues", JSON.stringify(paramsBatches));

}

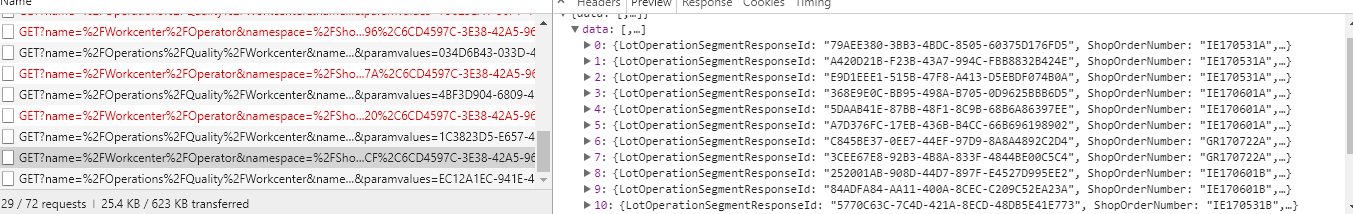
}

There are two API calls that are happening

1. <https://vdcglq05530.ics.cloud.ge.com:8443/ProxyService/GET?name=/Workcenter/Operator&namespace=/ShopOrders&paramnames=workcenter,operator,language&paramvalues=AC04DA93-F1B1-45B4-A632-AAD340B7EFF1,41A7A10E-1201-40B1-93CA-4525E7BE58E9,en_US>

Here the first param value is item.identifier and other is userinfo.identifier

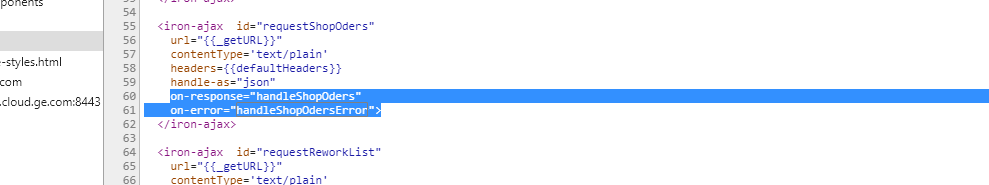
this return the list of lotoperation data



while the second API calls the workcenter details.

<https://vdcglq05530.ics.cloud.ge.com:8443/ProxyService/GET?name=/Operations/Quality/Workcenter&namespace=/ShopOrders&paramnames=workcenter&paramvalues=AC04DA93-F1B1-45B4-A632-AAD340B7EFF1>

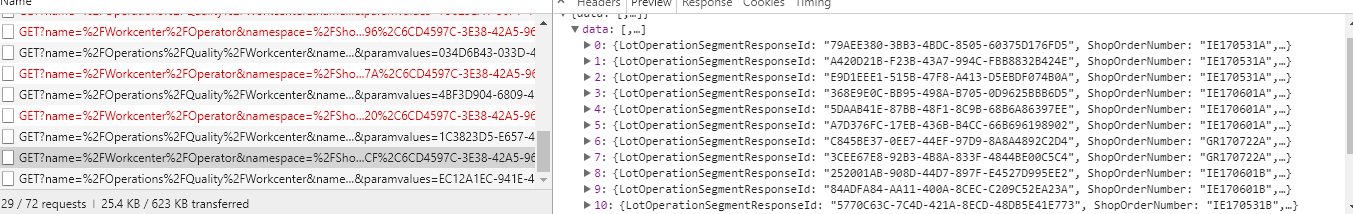
We found that for these Ajax call the Polymer component used (Iron-Ajax) has a success and failure function which depending on the response of the API are triggered.



On Success we call handleShopOders function while on error we call handleShopOdersError function.

HandleShopOrders function iterates the response and create the table

For example the response is



This is iterated here to create the response table with detail and icons.

handleShopOders: function (e) {

// var usersResponse = data.detail.\_\_data\_\_.response;

var data = e.detail.response.data;

this.\_shopOrderData = [];

this.\_nonBatchData = []

this.\_batchData = []

for (var i = 0; i < data.length; i++) {

var node = data[i];

node.id = i;

node.isSelected = false

node.iconElement = this.getIconTextItem(false, this.\_getIcon(node.Icon));

node.selectSerialElement = this.getSerialInputCheckboxItem(false, node.id, "serialElement")

node.BatchIdElement = this.getIconSerialTextItem(false, ((node.BatchId == null) ? this.\_labels.notbatchedLabel : node.BatchId), node.BatchId);

node.SerialLotNumberElement = this.\_getLink(false, i, node.LotOperationSegmentResponseId, node.SerialLotNumber);

node.SerialLotNumberElementNonBatch = node.SerialLotNumber

node.ShopOrderNumberElement = node.ShopOrderNumber

node.PartNumberElement = node.PartNumber

node.startDateElement = node.StartDate

node.endDateElement = node.EndDate

node.CompletedQuantity = new Number(node.CompletedQuantity);

node.Priority = new Number(node.Priority);

node.LotQuantity = new Number(node.LotQuantity);

node.OrderQty = new Number(node.OrderQty);

**Where as on error condition we show a generic error message irrespective of error type.**

handleShopOdersError: function(e){

var \_this = this;

// var usersResponse = data.detail.\_\_data\_\_.response;

this.\_shopOrderData = [];

this.\_nonBatchData = []

this.\_batchData = []

this.$.loading.classList.add("hide");

this.message = this.\_labels.handleShopOdersError;

this.$.alert\_message.classList.remove("hide");

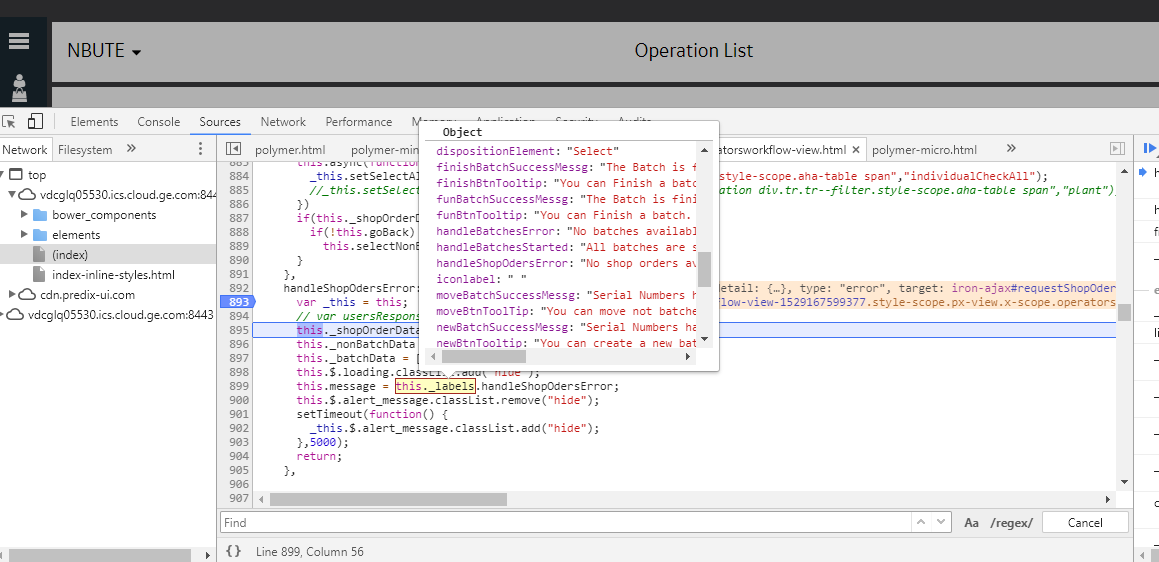
setTimeout(function() {

\_this.$.alert\_message.classList.add("hide");

},5000);

return;

},



As we can see there is a json of object created which have hardcoded error list if some how the response is a failure it will always show the generic message

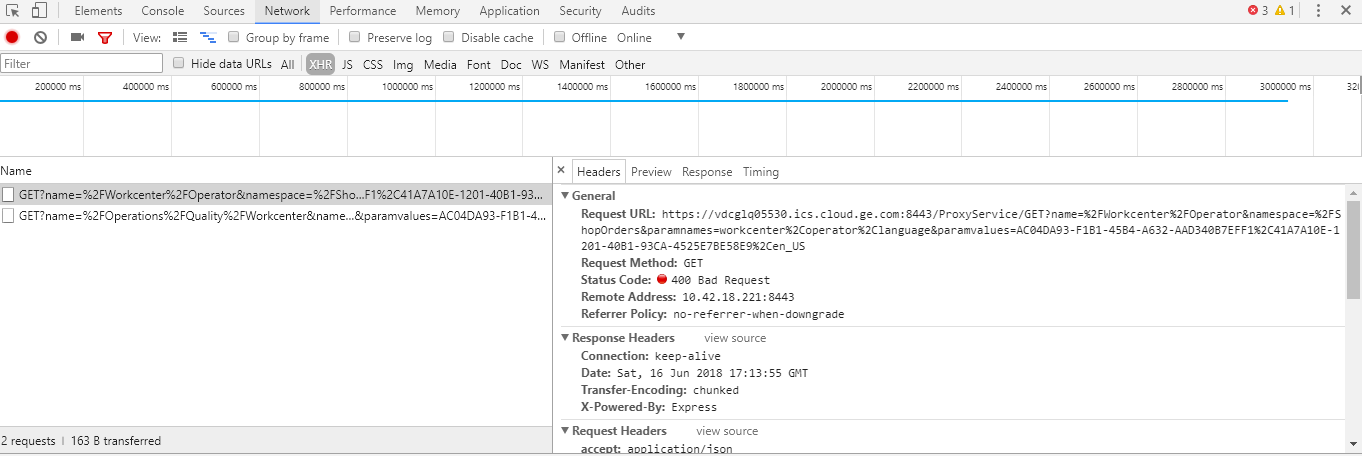
Here the list of generic messages are:

1. BatchId:"Batch ID"
2. BatchIdElement:"Batch ID"
3. BatchesTab:"Batches"
4. CompletedQuantity:"Completed Qty"
5. EndDate:"End Date"
6. FinishButton:"Finish"
7. FinishandUnbatch:"Finish and Unbatch"
8. LotQuantity:"Lot Qty"
9. MoveButton:"Move"
10. MoveModalNegative:"Close"
11. MoveModalPositive:"Apply"
12. MoveModalSubtittle:"Select Batch to add selected"
13. MoveModalTittle:"Move to Selected Branch"
14. NewButton:"New"
15. Operation:"Operation"
16. OrderQtyTitle:"Order Qty"
17. PartNumber:"Part Number"
18. Priority:"Priority"
19. Quantity:"# of Serial Number/Lots"
20. RemoveButton:"Remove From Batch"
21. Selected:"Selected"
22. SerialLotNumber:"Serial # / Lot #"
23. SerialNumbersTab:"Individual Serial Numbers"
24. SerialOrderElement:"# of Serial Numbers/Lots"
25. ShopOrderNumberElement:"Shop Order #"
26. StartButton:"Start"
27. StartDate:"Start Date"
28. Status:"Status"
29. StopButton:"Stop"
30. affectedQty:"Affected Qty"
31. clockModalTitle:"Users Access"
32. clockOffTime:"Clock ON Date & Time"
33. contextBrowserDetaulfSelection:"Select Work Station"
34. createdBy:"Created By"
35. createdOn:"Created On"
36. dispositionElement:"Select"
37. finishBatchSuccessMessg:"The Batch is finished, The Clock is OFF"
38. finishBtnTooltip:"You can Finish a batch. Operation will be complete. The clock will be OFF"
39. funBatchSuccessMessg:"The Batch is finished, The Clock is OFF and was removed"
40. funBtnTooltip:"You can Finish a batch. Operation will be complete. The clock will be OFF. The batches are removed"
41. handleBatchesError:"No batches available"
42. handleBatchesStarted:"All batches are started"
43. handleShopOdersError:"No shop orders available"
44. iconlabel:" "
45. moveBatchSuccessMessg:"Serial Numbers have been moved to a Batch"
46. moveBtnToolTip:"You can move not batched Serial #'s into an existing batch"
47. newBatchSuccessMessg:"Serial Numbers have been added to a Batch"
48. newBtnTooltip:"You can create a new batch from serial #'s that are not part of a batch"
49. nonConformanceID:"NCR ID"
50. notbatchedLabel:"Not Batched"
51. operation:"Operation"
52. removeBatchSuccessMessg:"Serial Numbers have been removed from the Batch"
53. removeBtnToolTip:"You can remove Serial Numbers from a batch, they will become not batched"
54. reworkStatus:"Rework Status"
55. serialNumber:"Serial #"
56. shopOrder:"Shop Order"
57. startBatchSuccessMessg:"The Batch has started, The Clock is ON"
58. startBtnTooltip:"You can start operation on a batch. The clock will be ON"
59. stopBatchSuccessMessg:"The Batch has stop, The Clock is OFF"
60. stopBtnTooltip:"You can stop operation on a batch. The clock will be OFF"
61. titlePage:"Operation List"
62. user:"User"
63. users:"User(s)"

And we called this.\_labels.handleShopOdersError;

As the error message is “No shop orders available” this is generic message irrespective of whether the API was failed due to empty data or due to invalid token.

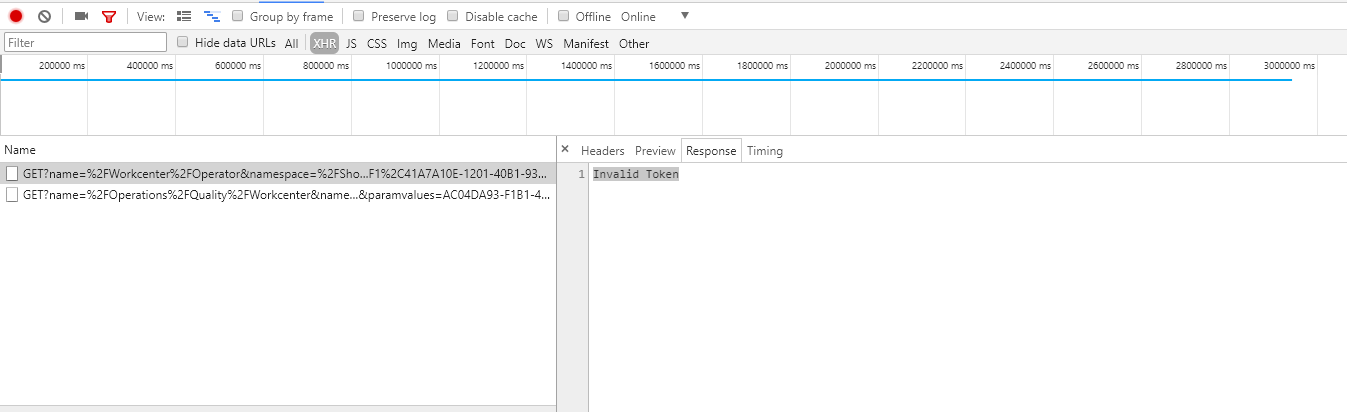
I found that the API failed due to token expire but although I got the same error message as “No shop orders available”



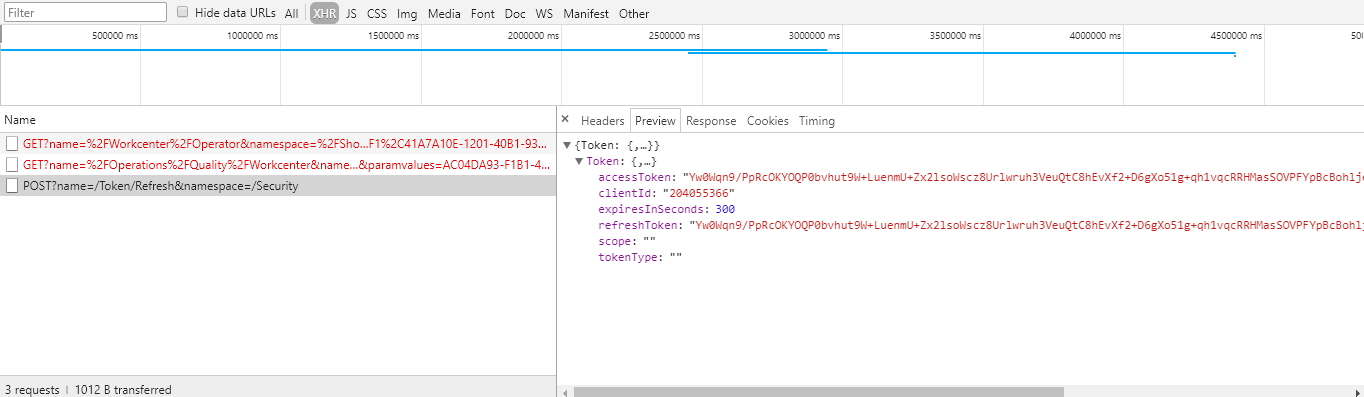
API : <https://vdcglq05530.ics.cloud.ge.com:8443/ProxyService/GET?name=/Workcenter/Operator&namespace=/ShopOrders&paramnames=workcenter,operator,language&paramvalues=AC04DA93-F1B1-45B4-A632-AAD340B7EFF1,41A7A10E-1201-40B1-93CA-4525E7BE58E9,en_US>

Response:

Invalid Token



Once the error message alert box hide again a post API to refresh the token is fired which set the new session token <https://vdcglq05530.ics.cloud.ge.com:8443/ProxyService/POST?name=/Token/Refresh&namespace=/Security>



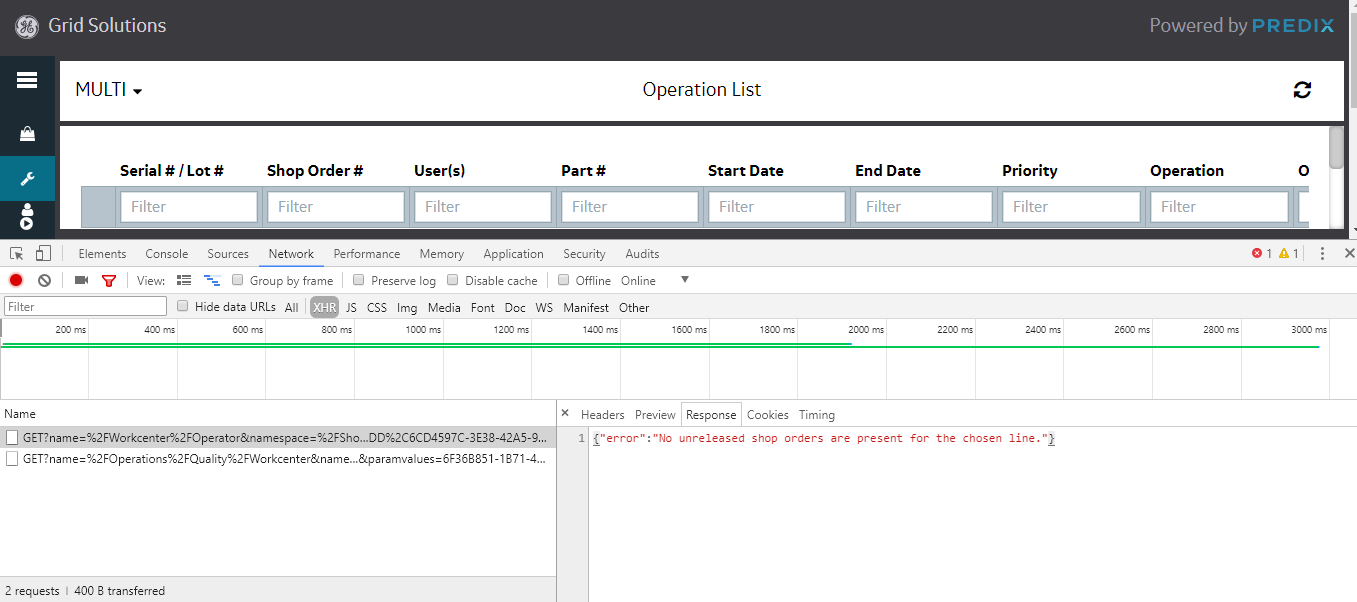
**Recommendation/Solutions**

1. We can show the actual error message that the API return irrespective of showing a generic error message.

As shown below when there is no data available and we have a 404 error we get a error message as

{"error":"No unreleased shop orders are present for the chosen line."}

So we can show the specific error



1. We can show two error message one when actually there is no response from the API as no shop order present while the second one is when our session gets expire then other error message specifying that the session is expire please wait.

**Implementation:**

On local we made the following changes :

handleShopOdersError: function(e){

var \_this = this;

this.opListResponseComplete = true;

// var usersResponse = data.detail.\_\_data\_\_.response;

this.\_shopOrderData = [];

this.\_nonBatchData = []

this.\_batchData = []

this.$.loading.classList.add("hide");

/\*check if error message available in response\*/

if(e.detail.request.xhr.response){

this.message = e.detail.request.xhr.response.error;

this.$.alert\_message.classList.remove('sessionExpire');

}else{

this.$.alert\_message.classList.add('sessionExpire');

this.message = 'Session is expired please wait..';

}

// this.message = this.\_labels.handleShopOdersError;

this.$.alert\_message.classList.remove("hide");

setTimeout(function() {

\_this.$.alert\_message.classList.add("hide");

\_this.$.alert\_message.classList.remove('sessionExpire');

},5000);

return;

},

<style>

.sessionExpire ::content>#alert {

background-color: #ff7f7f;

color: white;

}

</style>

The green highlighted and the style added in bottom are the new code changes. Here we are checking if there is any response from the API on failure and if yes then will show the same error on the modal else will show the session expire error message.

Here I am adding a class “sessionExpire” which will apply different style on session expire.

this.$.alert\_message.classList.add('sessionExpire');

this.message = 'Session is expired please wait..';

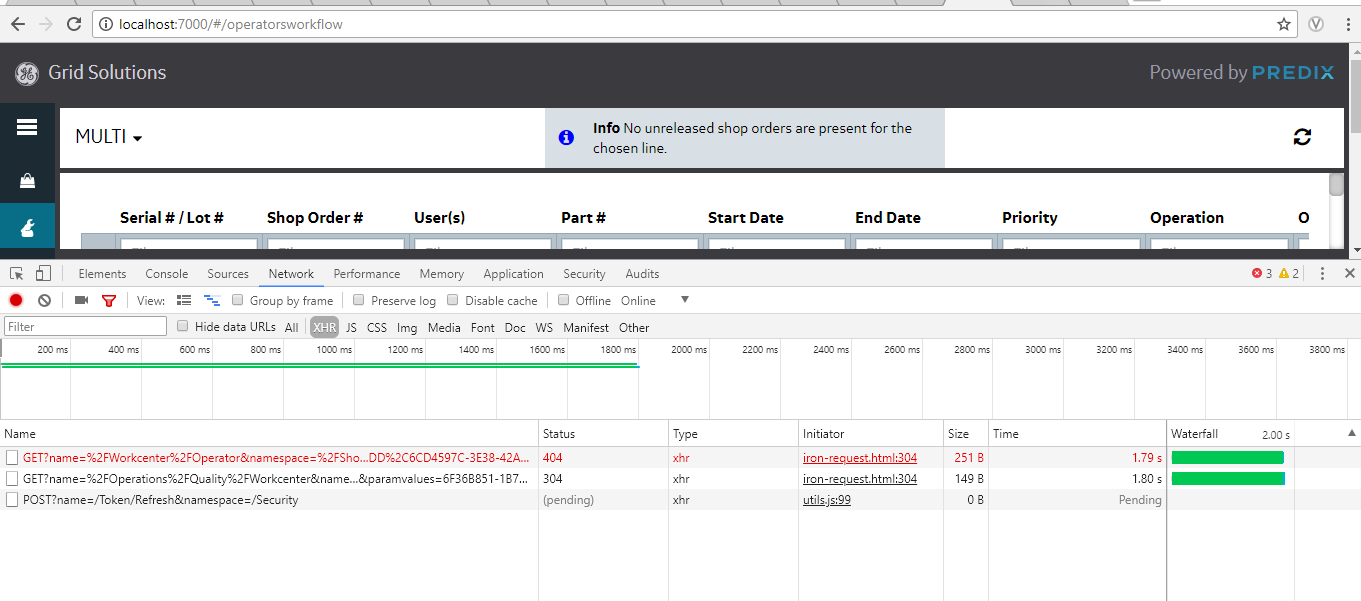
ToDo:

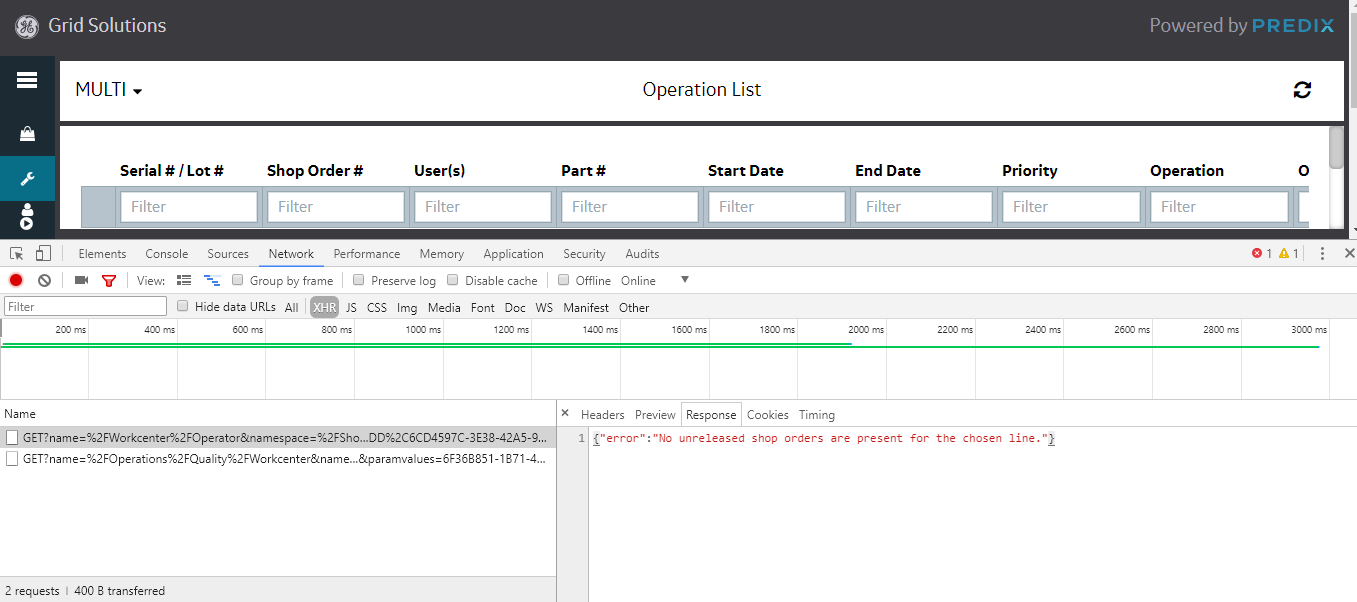
Instead of hardcoding the error message it can be placed in the same file where all the messages are added.

**this.\_labels.sessionExpireError;**

**Output:**

1. When blank data returned





1. When session gets expire

