Conversion of Specific Technical Deliverable Template into Quick Grid

Please verify the Quick Grids in

Process-->Templates-->Quick Grids-->**Specific Technical Deliverable NAD EP.**

Specific Technical Deliverable NAD EP is one of the Quick Grids developed form Template.

The objects are created based on the models and the number of phases.

For example the EP NAD model contains 3 phases.

1. Prepare Kickoff
2. Technical Development
3. Production & Commercialization

As based on the phases designed in the template, the dynamic objects (**Drop down boxes/Multi line text boxes**) are created in the Quick grid.

The Drop down boxes/Multi line text boxes on each phase based on the **List Script/String Script**.

For example, the metric **CTQ** is the object (List script) which is created through JavaScript code based on the template design and the phase on that model.

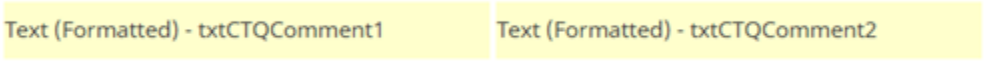
Each object should be named (any name) in the sequence as 1, 2, 3..

Example we here named it as txtCTQ1, txtCTQ2. .

As same as CTQ, the metric **CTQComments** is the object (String script) which is created through JavaScript code based on the template design and the phase on that model.

Each object should be named (any name) in the sequence as 1, 2..

Example we here named it as, txtCTQcomment1, txtCTQcomment2.



Below are steps and the description of the code in the Quick Grid

1. Initialize the Objects, metric types, phase names etc.
2. Create the object based on metrics.
3. Bind the values in the created objects.
4. Insert/Update the Metric values
5. Change the color based on the conditions

Need to initialize the matrix name, Phase name, Current Phase name, Matrix types (Drop down, Text box) as an array in the JavaScript code.

```
var qgSystemName = 'qgSpecificTechnicalDeliverableNADEP';
var mxMatrics = ["TechnicalDeliverableName","CTQ","CTQComment","ProductDesignFMEA","ProductDesignFMEAComment","ProcessFMEA","ProcessFMEAComment","ProductDesign","ProductDesignComment","ProductControlPlan","ProductControlPlanComment","ProductSpecifications","Pro
var mxMatricTypes = ["String","List","Long String","List","Long String","List","Long String","List","Long String","List","Long String","List","Long String","List","Long String","List","Long String","List","Long String","List","Long String","List","List","List",
var modelPhases = ["Prepare Kickoff","Technical Development","Production & Commercialization"];
var phaseLabel = ["PrepareKickoff","TechnicalDevelopment","ProductionCommercialization"];
var phaseName = getProjectMetadata('DeliverableStageName');
var currentPhase = getProjectMetadata('ProjectCurrentStageName');
```

Need to create the Object based on the metrics.

```
function createObjects() {
   var strScript='';
   var ListScript = '<select id="ObjectID"> <option value=""></option> <option value="Yes">Yes</option> <option value="No">No</option> <option value
   var StringScript = '<input type="text" name="ObjectID" id="ObjectID"></input>';
   var LongStringScript ='<textarea name="ObjectID" id="ObjectID" rows="3" style="resize:none;"></textarea>';
   var gridObj,gridCellName, newObject, newObjectID, metricName, matricType;
   var workingPhase = modelPhases.indexOf(phaseName);
   workingPhase = workingPhase+1;// to use phaseid as part of objectname on eachphase
   for(var c = 0; c< mxMatrics.length; c++){ // each metrics of matrix
      for (var p = 0; p < modelPhases.length; p++){ // each phase of model
         metricName = mxMatrics[c];
         matricType = mxMatricTypes[c];
         gridCellName = 'txt'.concat(metricName,p+1);
         gridObj = getElementByControlSystemName(gridCellName);
         newObjectID = 'obj'.concat(metricName,p+1);
         if (gridObj !==null) {

            if (matricType ==="String"){
               strScript = StringScript;
            }else if (matricType ==="List"){
               strScript = ListScript;
            }else if (matricType ==="Long String"){
               strScript = LongStringScript;
            }
            strScript = strScript.replace("ObjectID", newObjectID);
            strScript = strScript.replace("ObjectID", newObjectID);
            gridObj.append(strScript);
```

Need to bind the matrix values to the created objects.

```
// To bind the values to the objects created

function bindMXtoObjects(){
   var myMatrix = readMatrixGridToArray(qgSystemName,'mxTechnical');
   var myValue, metricName, matricType, ObjectID, Obj;

   if ( myMatrix.length < modelPhases.length){
      for (var i=myMatrix.length ; i<modelPhases.length; i++ ){
         addNewRowToMatrixGrid(qgSystemName, 'mxTechnical');
      }
   }
   for (var irow=1; irow<=modelPhases.length;irow++){
      setMatrixCellValue(qgSystemName, 'mxTechnical', 1, irow, modelPhases[irow-1]);
   }

   for(var r = 0; r< myMatrix.length; r++){ //metrics rows
      for(var c = 0; c< myMatrix[r].length; c++){
         myValue = myMatrix[r][c];
         metricName = mxMatrics[c];
         matricType = mxMatricTypes[c];
         ObjectID = 'obj'.concat(metricName,r+1);
         Obj = document.getElementById(ObjectID);
         if ( Obj !== null){

            if (matricType ==="String" || matricType ==="Long String"){
               Obj.value = myValue;
            }else if (matricType ==="List"){
               Obj.value = myValue;
               colorChange(Obj, myValue);
```

Insert/Update the values of each metrics.

```
function changeValues(){
   var ObjectID = event.target.id;
   var ObjType = event.target.type;
   var mxCol = getMatrixGridColumnIndex(qgSystemName, 'mxTechnical', ObjectID.substring(3, ObjectID.length-1), 'metricSystemName');
   var EventPhaseNumber = ObjectID.substring(ObjectID.length-1,ObjectID.length);
   var workingPhase = modelPhases.indexOf(phaseName);
   workingPhase = workingPhase+1;
   var newValue = "";

   if ( parseInt(EventPhaseNumber) === workingPhase ){ //only allow change of deliverable phase
      if (ObjType === "textarea" || ObjType === "input"){
         newValue = document.getElementById(ObjectID).value;
      }else{
         var ddlObj = document.getElementById(ObjectID);
         newValue = ddlObj.options[ddlObj.selectedIndex].value;
         colorChange(ddlObj, newValue);
      }
      setMatrixCellValue(qgSystemName, 'mxTechnical', mxCol, workingPhase, newValue);
   }
}
```

We need to change the color, based on the conditions given in the template.

Below coding explains the back ground color changes in the drop down box (List Script) based on the conditions.

Conditions:

- Not Applicable - Yellow
- Yes - Green
- No - Red.

```
//To change the backgroundColor based on the object value

function colorChange(ddlObj, ddlValue){

if(ddlObj !== null){
        if(ddlValue === 'Yes'){
       ddlObj.parentNode.parentNode.style.backgroundColor = '#26b73c';
          }else if(ddlValue === 'No'){
             ddlObj.parentNode.parentNode.style.backgroundColor = '#ff4545';
          }else {ddlObj.parentNode.parentNode.style.backgroundColor = '#ffffcc';}

}else {
    var ObjectID = event.target.parentNode.parentNode;
    if(ddlValue === 'Yes'){
     ObjectID.style.backgroundColor = '#26b73c';
    }else if(ddlValue === 'No'){
       ObjectID.style.backgroundColor = '#ff4545';
    }else {ObjectID.style.backgroundColor = '#ffffcc';}
}
```

Accolade