```
        var p_uid;
        var userCacInfo
        var user_last_name;
        var user_first_name;
        var aoiDS;
        var urlRoot;
        var tbfiltbut;
        var toolbar2;
        var gridlayer;
        var grid_polyCtrl, grid_modCtrl, grid_sav, grid_clr;
        var uniqueID;
        var loadO3D;
        var savePrompt;
        var mquerySelect;


        Array.prototype.getUnique = function(){
        var u = {}, a = [];
         for(var i = 0, l = this.length; i < l; ++i){
          if(u.hasOwnProperty(this[i])) {
          continue;
         }
         a.push(this[i]);
         u[this[i]] = 1;
        }
       return a;
       }

       function getUserInfo(){

          Ext.Ajax.request({ url: gSchema+".grid_functions.get_uid",
                                 success: procUid,
                                 method: "POST",
                                 disableCaching: false});

        };

        function procUid(resp){
         var udata = Ext.decode(resp.responseText,true);

         p_uid = udata.user_id;
         session_id = udata.sessionid;
         userCacInfo = udata.userCacInfo;
         if (aoiDS) {
                 aoiDS.proxy.setApi({read:
gSchema+".grid_functions.aoiSearch?puid="+p_uid});
                 aoiDS.load();
         }

         if (cmOver) {
                 cmOver.params.puid = p_uid;
                 cmOver.params.sessionid = session_id;
         }
        }


        function initbaseToolbar() {



         Ext.override(Ext.form.TextField, {
             setEmpty: function(v){
```

```
            this.emptyText = v;
            this.clearValue();
            this.setValue("");
            this.applyEmptyText();
            this.autoSize();
            return this;
            }
        });


            tbfiltbut = new Ext.Button({ text: "Filter PointClouds", handler:
function() {
                  filterLayerName = "Lidar_Boundaries";
                  filter_display();
            }});
          // toolbar2.add(tbfiltbut);


            Ext.override(Ext.PagingToolbar, {
            paramNames: {
                start:'p_start',
                limit:'p_limit'
            }
        });

        Ext.form.ComboBox.override({
            getParams : function(q){
                var p = {};
                if(this.pageSize){
                    p['p_start'] = 0;
                    p['p_limit'] = this.pageSize;
                }
                return p;
            }
        });


       var resultTpl = new Ext.XTemplate(
                '<tpl for="."><div class="search-item">',
                    '{name} ',
                    '</div></tpl>' );

      var ss_resultTpl = new Ext.XTemplate(
                '<tpl for="."><div class="search-item">',
                    '{name}',
                    '<br>{complete_name}',
                    '</div></tpl>' );


       aoiDS = new Ext.data.Store({
                proxy: new Ext.data.ScriptTagProxy({
                        nocache: false,
                    callbackParam: 'p_callback'
                }),
                reader: new Ext.data.JsonReader({
                    root: 'rowset',
                    totalProperty: 'totalrows',
                    id: 'id'
                }, [
                    {name: 'name',        mapping: 'name'},
                    {name: 'extent',      mapping: 'extent'}
                ])
            });
```

```
     aoiSearch =  new Ext.form.ComboBox({
                store: aoiDS,
                displayField:'name',
                valueField:'id',
                typeAhead: false,
                itemSelector: 'div.search-item',
                emptyText:'Select an AOI',
                width: 200,
                loadingText: 'Searching...',
                pageSize:50,
                tpl: resultTpl,
                triggerClass:'x-form-search-trigger',
                triggerAction:'all',
                minChars:1,
                queryParam:'p_query',
                onSelect: function(record){ // override default onSelect to do
redirect
                   //console.log(record.data.description);
                    this.value = record.data.name;
                    zoomToProject(record.data.extent);

                    tree.getRootNode().cascade(function(node) {
                    if (node.attributes.layerName == "user_extracts" ) {
                      node.getUI().toggleCheck(true); }
                    } );
                    this.collapse();
                    this.clearValue();
             }

            });


      locationDS = new Ext.data.Store({
                proxy: new Ext.data.ScriptTagProxy({
                        url: "/"+urlRoot+"/corpsmap/smartsearch?",
                        nocache: false,
                    callbackParam: 'p_callback'
                }),
                reader: new Ext.data.JsonReader({
                    root: 'rowset',
                    totalProperty: 'totalrows',
                    id: 'id'
                }, [
                    {name: 'name',        mapping: 'name'},
                            {name: 'complete_name',  mapping: 'complete_name'},

                            {name: 'lat',         mapping: 'lat'},
                    {name: 'lon',         mapping: 'lon'}
                ])
            });



     locationSearch =  new Ext.form.ComboBox({
                store: locationDS,
                displayField:'name',
                valueField:'id',
                typeAhead: false,
                itemSelector: 'div.search-item',
                emptyText:'Enter an Lon/Lat, MGRS or Location Name',
                width: 300,
```

```
                loadingText: 'Searching...',
                pageSize:50,
                allowBlank: false,
                lastQuery: '',
                tpl: ss_resultTpl,
                triggerClass:'x-form-search-trigger',
                triggerAction:'query',
                listeners: {
                    beforequery: function(qe){
                        delete qe.combo.lastQuery;
                    }
                 },
                minChars:3,
                queryParam:'p_query',
                onSelect: function(record){ // override default onSelect to do
redirect
                   //console.log(record.data.description);
                    this.value = record.data.name;

zoomToSmartLL(record.data.lat,record.data.lon,record.data.name+"<br>"+record.data.co
mplete_name);
                    this.collapse();
                    this.clearValue();
             }

            });

                                                mquerySelect = new
OpenLayers.Control.SelectFeature(mqueryLayer);
            mquerySelectAction = new GeoExt.Action({control: mquerySelect,

text: 'SS?',

tooltip: 'Select Smart Search Points',

toggleGroup: 'aoi',
                                                                                id:
'mqueryBtn',
                                                                                map:
map});


                                                mquerySelectClear = new
Ext.Toolbar.Button({iconCls: 'mfclear', tooltip: 'Clear SmartSearch Features',
handler: function(){ destroyPopups(); }});


           //  toolbar2.add(new Ext.Toolbar.Separator());
           //  toolbar2.add(new Ext.Toolbar.Spacer());
             toolbar2.add(new Ext.Toolbar.TextItem({text: 'AOI: '}));
             toolbar2.add(aoiSearch);
             toolbar2.add(new Ext.Toolbar.Separator());
             toolbar2.add(new Ext.Toolbar.Spacer());
             toolbar2.add(new Ext.Toolbar.TextItem({text: 'SmartSearch: '}));
             toolbar2.add(locationSearch);
             toolbar2.add(mquerySelectAction);
                                                toolbar2.add(mquerySelectClear);
             toolbar2.add(new Ext.Toolbar.Separator());
             toolbar2.add(new Ext.Toolbar.Spacer());



     };
```

```
            function zoomToSmartLL(lat,lon,description){
            var feat;

               ll = new OpenLayers.LonLat(lon,lat);
                                ll.transform(new
OpenLayers.Projection("EPSG:4326"), new OpenLayers.Projection("EPSG:900913"));
                                map.setCenter(ll,12);
                                feat =  new OpenLayers.Feature.Vector( new
OpenLayers.Geometry.Point(ll.lon, ll.lat));
                                feat.attributes.description=description;
                                mqueryLayer.addFeatures([feat]);
                                addSmartPopup(feat);

                               // drawLayer.addFeatures(ll);
             // drawLayer.redraw();

                 }


         function addSmartPopup(feature){
                        popup = new GeoExt.Popup({
                    title: "Search Result",
                    location: feature,
                    html: unescape(feature.attributes.description),
                    panIn: true,
                    layout: 'fit',
                    width: 200,
                    height: 80,
                    map: map
                    /*
                    ,

                    listeners: {
                     "close": function() {
                     mquerySelect.unselect(this);
                  }
                            }
                            */
                    });
                 popup.show();

                 popup.on({
            close: function(){
                if (OpenLayers.Util.indexOf(mqueryLayer.selectedFeatures,
this.location) > -1) {
                    mquerySelect.unselect(this.location);
                }
            }
        });

        }



          function filtBtnStatusSet(){

          if (layerFilterArray.containsKey("Lidar_Boundaries")){
                 tbfiltbut.setText("Filter PointClouds: On");
            }else{
                 tbfiltbut.setText("Filter PointClouds: Off");
```

```
            }
        }


        toolbar2 = new mapfish.widgets.toolbar.Toolbar({
                    map: map,
                    enableOverflow: true,
                    id: "toolbar2",
                    configurable: false
          });

     function addLOSTool(){



          tbLOSctl = new OpenLayers.Control.DrawFeature(drawLayer,
OpenLayers.Handler.Point, {featureAdded: sendLOS,
            handlerOptions: {layerOptions: {  styleMap: new OpenLayers.StyleMap({
            'default': new OpenLayers.Style({
                 fillColor: "#FFEC8B",
                 fillOpacity: 0.3,
                 stroke: true,
                 cursor: "crosshair",
                 strokeColor: "#000000",
                 strokeWidth: 2,
                 pointRadius: 15
              })
           })}}});

        //  toolbar.addControl(tbLOSctl);
        //  toolbar2.addControl(tbLOSctl, {id: 'tbLOSctl', text: 'LOS', tooltip:
'Get Line of Sight Info for clicked location',  toggleGroup: 'los'});
        //  toolbar.removeControl(tbLOSctl);
     }

     function sendLOS(){
        var losurl = "";
        var elev;
        feat = drawLayer.features[0];
        var geom = feat.geometry;
        geom.transform( map.getProjectionObject(), new
OpenLayers.Projection("EPSG:4326"));
        pt = new google.maps.LatLng(geom.y, geom.x);
        var locations = [];
        locations.push(pt);
        //getElevation(locations, elev);
        alert(losurl+"?x="+geom.x+"&y="+geom.y);
    }


     function addGRiDAOI() {

        gridLayer = new OpenLayers.Layer.Vector("grid", {
        styleMap: new OpenLayers.StyleMap({
              'default': new OpenLayers.Style({
                 fillColor: "#23778E",
                 fillOpacity: 0.5,
                 stroke: true,
                 strokeColor: "#000000",
                 strokeWidth: 2,
                 pointRadius: 15
              })
           }),
```

```
        displayInLayerSwitcher: false, minZoomLevel: 0, numZoomLevels: 21 });
        map.addLayer(gridLayer);

        var ctrl, toolbarItems = [], action, actions = {};
        toolbar2.add(new Ext.Toolbar.TextItem({text: 'AOI Tools: '}));
        grid_polyCtrl = new GeoExt.Action({
        iconCls: 'drawpolygon',
        id: "grid_polyCtrl",
        control: new OpenLayers.Control.DrawFeature(
            gridLayer, OpenLayers.Handler.Polygon, {callbacks: {"point":
updateArea},
             handlerOptions: {layerOptions: {  styleMap: new OpenLayers.StyleMap({
            'default': new OpenLayers.Style({
                 fillColor: "#FFEC8B",
                 fillOpacity: 0.3,
                 stroke: true,
                 strokeColor: "#000000",
                 strokeWidth: 2,
                 pointRadius: 15
              })
           })}}}
        ),
        map: map,
        // button options
        toggleGroup: "aoi",
        allowDepress: true,
        tooltip: "Add a AOI Extract Area (shift-Click or dbl-click to end)"
    });
    toolbar2.add(grid_polyCtrl);
    grid_polyCtrl.control.events.register("activate",map,function(e) {
           if (dragPan.active==false){
              dragPan.activate()
              grid_polyCtrl.control.deactivate();
              grid_polyCtrl.control.activate();
           }
           if (gridLayer.features.length>0) { alert("Maximum Features. Please Clear
or Save");  }
          });


    grid_modCtrl = new GeoExt.Action({
          iconCls: 'modifyfeature',
          toggleGroup: 'aoi',
          id: "grid_modCtrl",
          tooltip: 'Modify an AOI area. Drag light colored circles to add new
vertex. Pres "del" key while hovered over vertex to remove',
          allowDepress: true,
          disabled: true,
          control: new OpenLayers.Control.ModifyFeature(gridLayer),
          map:map
    });
    //toolbar.addControl(grid_modCtrl);
    toolbar2.add(grid_modCtrl);

    /* grid_modCtrl = new OpenLayers.Control.ModifyFeature(gridLayer);
     toolbar2.addControl(grid_modCtrl,  {id: 'grid_modCtrlBtn',
                                                                iconCls:
'modifyfeature',
                                                                toggleGroup: 'map',
                                                                tooltip: 'Modify a
geometry. Drag light colored circles to add new vertex. Pres "del" key while hovered
over vertex to remove'
     }); */
```

```
    //grid_modCtrl = new OpenLayers.Control.ModifyFeature(gridLayer);
    //toolbar.addControl(grid_modCtrl);
    //toolbar2.addControl(grid_modCtrl, { iconCls: 'modifyfeature', toggleGroup:
"aoi", allowDepress: true, map: map});
    //toolbar.removeControl(grid_modCtrl);


    grid_sav = new Ext.Toolbar.Button({id: "grid_sav", iconCls: 'mfsave', tooltip:
'Save an AOI and Close', handler: function(){ doGridGeomStore(); }});
    toolbar2.add(grid_sav);
    //grid_modCtrl.disable();
    grid_sav.disable();
    grid_clr = new Ext.Toolbar.Button({id: "grid_clr", iconCls: 'mfclear', tooltip:
'Clear AOI', handler: function(){ doGridGeomClear(); }});
    toolbar2.add(grid_clr);
    grid_clr.disable();
    gridLayer.events.register("featureadded",map,function(e) {
               grid_polyCtrl.control.deactivate();
               grid_polyCtrl.disable();
               grid_modCtrl.enable();
               grid_sav.enable();
               grid_clr.enable();
     });
  }


      function doGridGeomStore(){

        feat = gridLayer.features[0];
        var geom = feat.geometry;
        var featid = feat.id;
        var statusBar = Ext.getCmp("my-status");
        statusBar.showBusy({text: "Looking up Name...."} );
        var apiCall = gSchema+".grid_Functions.getAOIDefaultName";
        var llgeom =  geom.clone();
        llgeom.transform( new OpenLayers.Projection("EPSG:900913"), new
OpenLayers.Projection("EPSG:4326"));
        var geomcenter = llgeom.getCentroid();

        Ext.Ajax.request({ url: apiCall, //+"?centroid="+geomcenter+"&puid="+p_uid,
                                success: doGridGeomStoreName,
                                failure: doGridGeomStoreName,
                                timeout: 5000,
                                method: "POST",
                                params: {"centroid": geomcenter, "puid": p_uid },
                                disableCaching: false});
     }

      function doGridGeomStoreName(response){

        var statusBar = Ext.getCmp("my-status");
        statusBar.clearStatus({useDefaults:true});
        //kbdControl.deactivate();
        var repData = Ext.decode(response.responseText,true);
        var defName = repData.defName;

        if ((p_uid)&&(!uniqueID)) {
           savePrompt = Ext.Msg.prompt("Save AOI", "Please enter a name for this
AOI",
            function(btn, text){
```

```
            if (btn == "ok"){
                post_doGridGeomStore(text);
             }
          },this,false,defName);
        } else {
          post_doGridGeomStore();
        }
      }


      function post_doGridGeomStore(textval){

           feat = gridLayer.features[0];

            var geom = feat.geometry;
            var featid = feat.id;
            var statusBar = Ext.getCmp("my-status");
             statusBar.showBusy({text: "Saving AOI and Calculating Statistics...."}
);


            var apiCall = gSchema+".cm2_geom_util.saveShape";
            var llgeom =  geom.clone();
            llgeom.transform( new OpenLayers.Projection("EPSG:900913"), new
OpenLayers.Projection("EPSG:4326"));

             Ext.Ajax.request({ url: apiCall,
//+"?p_geom="+llgeom+"&p_uid="+p_uid+"&projname="+textval+"&p_geom_id="+uniqueID,
                              success: grid_geoStoreSuccess,
                               failure: grid_geoStoreFailure,
                               method: "POST",
                               params: {"p_geom": llgeom, "p_uid": p_uid,
"projname": textval, "p_geom_id":uniqueID },
                               disableCaching: false});
          }


       function grid_geoStoreSuccess(response){

           resArray = response.responseText.split(";");
           var completeUrl = "/"+urlRoot+"/export/"+parseFloat(resArray[1])+"/";

           var statusBar = Ext.getCmp("my-status");
           statusBar.clearStatus({useDefaults:true});

           if (resArray[0]=="OK") {

            Ext.Msg.show({title: "Saved",
                           msg: resArray[2],
                       buttons: Ext.Msg.OK,
                            fn: function(btn, text){
                                  if (btn=="ok")  grid_redirUrl(completeUrl);
                             }
                           });

              } else {
                     ///kbdControl.activate();
             Ext.Msg.show({title: "Not Saved",
                          msg:  resArray[1],
                          buttons: Ext.Msg.OK,
                          icon: Ext.MessageBox.Error
                         });
            }
```

```
         }

            function grid_geoStoreFailure(response){

                        //kbdControl.activate();
            Ext.Msg.show({title: "Status",
                          msg:  response.responseText,
                          buttons: Ext.Msg.OK,
                          icon: Ext.MessageBox.Error
                          });

           }
       function grid_redirUrl(url){
                 parent.window.location=url;

           }

        function doGridGeomClear(){

            gridLayer.destroyFeatures();
                              grid_polyCtrl.enable();
                              grid_modCtrl.disable();
            grid_sav.disable();
            grid_clr.disable();
            setTip(null);
           }

     function doGridGeomGet(){


            uniqueID =
window.location.href.substr(window.location.href.indexOf("p_unique_id")+12).split("&
")[0];
                        var apiCall = gSchema+".cm2_geom_util.getShape";

            Ext.Ajax.request({ url: apiCall,
//+"?p_uid="+uniqueID+"&p_layer=null&p_role_id=null",
                                success: geoGridGetSuccess,
                                failure: geoGridGetFailure,
                                method: "POST",
                                params: {"p_uid": uniqueID, "p_layer": "",
"p_role_id":""},
                                disableCaching: false});
          }


          function geoGridGetSuccess(response) {

             var in_options = {
                "internalProjection": new OpenLayers.Projection("EPSG:900913"),
                "externalProjection": new OpenLayers.Projection("EPSG:4326")
            }


            var wkt =  new OpenLayers.Format.WKT(in_options);
            var element = response.responseText;
            var features = wkt.read(element);

            var bounds;
            if(features) {
                if(features.constructor != Array) {
                    features = [features];
                }
```

```
            for(var i=0; i<features.length; ++i) {
                if (!bounds) {
                    bounds = features[i].geometry.getBounds();
                } else {
                    bounds.extend(features[i].geometry.getBounds());
                }

            }

           gridLayer.addFeatures(features);
           gridLayer.redraw();
          }
        }

         function geoGridGetFailure(response){

          Ext.Msg.show({title: "Status",
                        msg:  "Error retrieving geometry",
                        buttons: Ext.Msg.OK,
                        icon: Ext.MessageBox.Error
                        });

         }


   function addLidarListView(){

         westPanel = Ext.getCmp("west-panel");

     lidarListStore = new Ext.data.JsonStore({
     url: gSchema+".grid_Functions.getCollectionList",
     root: "lidar",
     nocache: false,
     fields: [
         "collection",
         {name: "coll_date", type: "date" , dateFormat: "m/d/y"},
         "layer",
         "maplayer"]
      });

   lidarListView = new Ext.list.ListView({
           store: lidarListStore,
       emptyText: 'No Lidar Data in Current View',
       loadingText: "Updating List in View",
       multiSelect: true,
       reserveScrollOffset: true,
       columns: [
         {header: 'Collection' , dataIndex: 'collection', width: 0.6   }
        ,{header: 'Date', dataIndex: 'coll_date', tpl: '{coll_date:date("m/d/y")}',
width: 0.2}
        ,{header: 'Layer',dataIndex: 'layer'}
       ]
   });


    listPanel = new Ext.Panel({
       id:'listLidar',
       title:'Collection Info',
       border:false,
       map: map,
       autoScroll: true,
       listeners: {'expand': function(e) {
               lidarListStore.load({params: {"mapext":
```

```
map.getExtent().toGeometry().transform(new OpenLayers.Projection("EPSG:900913"), new
OpenLayers.Projection("EPSG:4326")),
                                        "layers":
"'"+tree.getChecked("layerName").join("','")+"'"}});
        }},
        layout: 'fit',
        iconCls:'nav',
        tbar: { layout:'toolbar',
                layoutConfig:{align:'left'},
                items: [{xtype:"buttongroup", items: [
                         { text: "<span style='font-size:x-small;'>Filter
Layer</span>",
                          handler: function() {
                               listFilterPC();
                           }
                }]},{xtype: 'tbseparator'},
                  {xtype:"buttongroup", items: [
                   { text: "<span style='font-size:x-small;'>Clear Filters</span>",
                    handler: function() {
                             layerFilterArray.eachKey(function(k) {
                                mapClearFilters(k);
                              });
                           }
                        }]}]
                }
     });



    lidarListStore.on('load', function(){
        //console.log("selecting data");
        //if (layerFilterArray.get("Lidar_Boundaries")!==undefined)
        layerFilterArray.eachKey(function(k) {

Ext.each(Ext.pluck(layerFilterArray.get(k),"category||collect_name")[0].split(","),
                        function(item, index, fullarray){
                                console.log(item, index, fullarray);
                                lidarListSelect("collection",item)});
        //console.log("data selected");
      });
    });
    cmOver.events.register("loadend",map, function (e){
        if (!listPanel.collapsed) {
          lidarListStore.load({params: {"mapext":
map.getExtent().toGeometry().transform(new OpenLayers.Projection("EPSG:900913"), new
OpenLayers.Projection("EPSG:4326")),
                                        "layers":
"'"+tree.getChecked("layerName").join("','")+"'"}});
        }
    });

    westPanel.items.insert(2,listPanel);
    //listPanel.items.add(lidarListView);
    westPanel.doLayout();
    listPanel.items.add(lidarListView);

   }

   function listFilterPC(){
          //pclist = "";
          layer_dict = []
          layers = lidarListView.store.collect('maplayer');
          Ext.each(layers,function(item,index,fullarray){
```

```
                layer_dict[item]="";
          });

          Ext.each(lidarListView.getSelectedIndexes(),
            function(item, index, fullarray){
                  layer_dict[lidarListView.store.getAt(item).data.maplayer] +=
lidarListView.store.getAt(item).data.collection+",";
                }
          );

      for (index = 0; index < layers.length; ++index) {

mapSetFilters(layers[index],{"category||collect_name":layer_dict[layers[index]]});

      }
   }

   function lidarListSelect(field,item){
          if (field=="flightmonth") {

lidarListView.select([lidarListStore.find(field,Date.parseDate(item,"M-Y"))],true);
          }else{
                lidarListView.select([lidarListStore.find(field,item)],true);
          }
   }

   function addPPTXButton(){
    locpdfPanel = Ext.getCmp("pdfPanel");
    locpdfPanel.addButton({

            text: "Create PowerPoint",
            id: "createPPPTXButton",
            hidden: true,
            handler: function() { doPPTX(); }
          });

   }


   function doPPTX() {
           pdfbut = Ext.getCmp("createPDFButton");

           pdfbut.handler(pdfbut);

   }

   function adjustPrintButtons(box,rec,id){
   layoutName = rec.json.name;
   pdfbut = Ext.getCmp("createPDFButton");
   pptbut = Ext.getCmp("createPPPTXButton");

   if (layoutName=="Powerpoint_Slide"){
                pdfbut.hide();
                pptbut.show();
                printProvider.addListener("beforedownload", function(e,u) {
generatePPTX(u); return false},  {single: true});
                printProvider.events["beforeprint"].clearListeners();
                printProvider.addListener("beforeprint", function() {
                               statusBar = Ext.getCmp('my-status');
                               statusBar.showBusy({text: "Generating Powerpoint
Output...."} );});
    }else{
      pdfbut.show();
```

```
                pptbut.hide();
                printProvider.events["beforedownload"].clearListeners();
                printProvider.events["beforeprint"].clearListeners();
                printProvider.addListener("beforeprint", function() {
                              statusBar = Ext.getCmp('my-status');
                              statusBar.showBusy({text: "Generating PDF
Output...."} );});
    }
   }

   function generatePPTX(url) {


          apiCall = window.location.origin + "/" +
window.location.pathname.split('/')[1] + "/corpsmap/ppt";
          pdf_name = url.split('/')[url.split('/').length-1];
          /*Ext.Ajax.request({ url: apiCall,
                              success: generatePPTXSuccess(pdf_name),
                              failure: generatePPTXFailure,
                              method: "GET",
                              params: {"pdf_name": pdf_name},
                              disableCaching: false});
      */
      document.getElementById("excelFrame").src = apiCall+"?pdf_name="+pdf_name;

   }



    function cm2PostFunctions(){

          mqueryLayer.styleMap = new OpenLayers.StyleMap({
              'default': new OpenLayers.Style({
                fillColor: "#55238E",
                fillOpacity: 0.3,
                stroke: true,
                strokeColor: "#000000",
                strokeWidth: 2,
                pointRadius: 5
              })
           });
        mqueryLayer.events.on({
                        featureselected: function(e) {
        addSmartPopup(e.feature);
        }
      });
        //console.log("grid functions loaded");
        urlRoot = window.location.pathname.split( '/' )[1];
        //getUserInfo();
        initbaseToolbar();
        toolbar2.doLayout();
        addGRiDAOI();
        if (Ext.getCmp("west-panel").collapsed) Ext.getCmp("west-panel").expand();
      //  addLidarListView();
 //       addPPTXButton()
        toolbar2.doLayout();
      //  layoutBox=Ext.getCmp("pdfPanel").items.get(2);
      //  layoutBox.on("select", function(layoutBox,theRecord,theId) {
adjustPrintButtons(layoutBox,theRecord,theId);});
      //  addPPTXButton();
      //  if (typeof(loadO3D())!="undefined") { addLidarListView(); };
        if ( typeof(loadO3D) == "function" ) loadO3D();
      //  if (window.location.href.indexOf("p_editlayer")>0) doGridGeomGet();
```

```
}
```