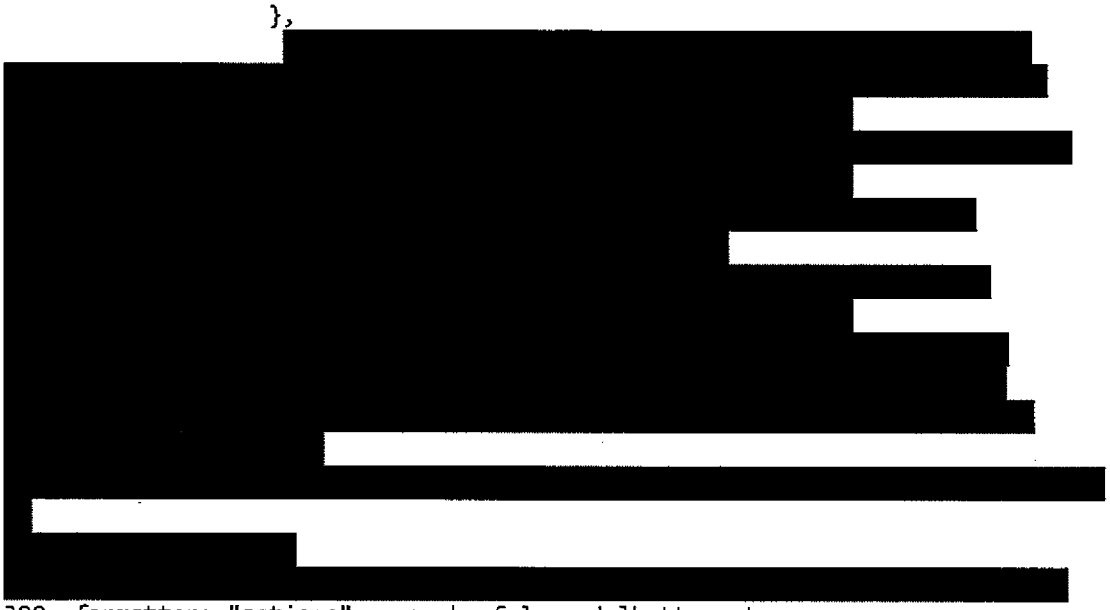
```
                },
```



```
300, formatter: "actions", search: false, delbutton: true,
                        formatoptions: {
                            afterSave: function (rowid, par2) {
                                DelRec(rowid, null);
                            },
                            keys: true,
                            editOptions: {

                            },
                            addOptions: {

                            },
                        }
                    },

                ],
                loadonce: true,
                viewrecords: true,
                width: 580,
                height: 850,
                rowNum: 20,
                direction: "rtl",
                pager: "#jqGridPager",
                grouping: true,
                multiselect: false,
                groupingView: {
                    //groupField: ["homL_AccountId"],
                    groupColumnShow: [true],
                    groupText: ["<b>{0}</b>"],
                    groupOrder: ["asc"],
                    groupSummary: [true],
                    groupCollapse: false
                }


            });

            //F_G_d();

            function F_G_d(rules) {
                var gridArrayData = [];
                for (i = 0; i < rules.length; i++) {
```

```
                    rule = rules[i];
                    var fieldVarN = rule.field;
                    var operator = rule.op;
                    var data = rule.data;
                    if (fieldVarN == "Var_I") {
                        if (!IsValidGovermentId(data)) {
                            alert("שגויה זהות תעודת");
                            isValid = false;
                        }
                        else {
                            isValid = true;
                        }
                    }
                    if (isValid || fieldVarN != "Var_I") {
                        // show loading message
                        $("#jqGrid")[0].grid.beginReq();
                        isAsync = typeof isAsync !== 'undefined' ? isAsync :
false;
                        jQuery.support.cors = true;
                        var GRID_DATA = odataPath11 +
"ContactSet?$select=FullVarN,GenderCode,el_id_city_main,MobilePhone,FirstVarN,A
ddress1_Line1,LastVarN,EMailAddress1,BirthDate,Var_I,DoNotBulkEMail,EMailAddres
s1,CreditLimit,ContactId";
                        if (operator == "cn") {
                            GRID_DATA += "&$filter=substringof('" + data + "',"
+ fieldVarN + ")";
                        } else {
                            GRID_DATA += "&$filter=" + fieldVarN + " " +
operator + " '" + data + "'";
                        }

                        var oDataUri = encodeURI(GRID_DATA);
                        $.ajax({
                            type: "GET",
                            contentType: "application/json; charset=utf-8",
                            datatype: "json",
                            url: oDataUri,
                            beforeSend: function (XMLHttpRequest) {
                                XMLHttpRequest.setRequestHeader("OData-
MaxVersion", "4.0");
                                XMLHttpRequest.setRequestHeader("OData-
Version", "4.0");
                                XMLHttpRequest.setRequestHeader("Accept",
"application/json");
                                XMLHttpRequest.setRequestHeader("Prefer",
"odata.include-annotations=\"*\"");
                            },
                            async: true,
                            success: function (data, textStatus,
XmlHttpRequest) {
                                for (var i = 0; i < data.d.results.length; i++)
{

                                    var item = data.d.results[i];
                                    pickListValue = item.GenderCode.Value;
                                    gridArrayData.push({
                                        FullVarN: item.FullVarN,
                                        FirstVarN: item.FirstVarN,
                                        GenderCode: getPicklistLabel("contact",
"gendercode", item.GenderCode.Value),
                                        City:
setNullValue(item.el_id_city_main.VarN + ";" + item.el_id_city_main.Id),
                                        Address: item.Address1_Line1,
```

```
                                    LastVarN: item.LastVarN,
                                    MobilePhone: item.MobilePhone,
                                    Var_I: item.Var_I,
                                    BirthDate: person_VarN(item.BirthDate),
                                    DoNotBulkEMail: new
Boolean(item.DoNotBulkEMail),
[redacted]
                                });
                            }

                            for (var i = 0; i <= gridArrayData.length; i++)
{
                                try {
                                    $("#jqGrid").jqGrid('addRowData', i +
1, gridArrayData[i]);
                                } catch (error) {
                                }
                            }

                            //// hide the show message
                            $("#jqGrid")[0].grid.endReq();
                            //// refresh the grid
                            $("#jqGrid").trigger('reloadGrid');
                        },
                        error: function (XmlHttpRequest, textStatus,
errorThrown) {
                            res = null
                        }

                    });
                }
                //return res;
            }
        }
$('#jqGrid').navGrid('#jqGridPager',
            // the buttons to appear on the toolbar of the grid
            { del: true, edit: true, add: true, search: true, refresh:
false, cancel: true, view: false, position: "left", cloneToTop: false },//del:
false,
            // options for the Edit Dialog
            {
                editCaption: "רשומה ערוך",
                bSubmit: "שלח",
                bCancel: "בטל",
                bClose: "סגור",
                saveData: "שינויים לשמור?",
                bYes: "כן",
                bNo: "לא",
                recreateForm: true,
                checkOnUpdate: true,
                checkOnSubmit: false,
                closeAfterEdit: true,
                errorTextFormat: function (data) {
                    return 'שגיאה: ' + data.responseText
                },
                onclickSubmit: function (params, posdata) {
                    DelRec(null, posdata);
                    return true;
                },
            },
```

```
            // options for the Add Dialog
            {
                addCaption: "רשומה הוסף",
                bSubmit: "שלח",
                bCancel: "בטל",
                bClose: "סגור",
                saveData: "שינויים לשמור?",
                bYes: "כן",
                bNo: "לא",
                closeAfterAdd: true,
                recreateForm: true,
                checkOnUpdate: true,
                checkOnSubmit: true,
                errorTextFormat: function (data) {
                    return 'שגיאה: ' + data.responseText
                },
                onclickSubmit: function (params, posdata) {
                    MakeGR(null, posdata);
                    return true;
                }

            },
            // options for the Delete Dailog
            {
                caption: "רשומה הסר",
                bSubmit: "מחק",
                bCancel: "בטל",
                bClose: "סגור",
                msg: "מהתצוגה הרשומה את להסיר ברצונך האם?",

                errorTextFormat: function (data) {
                    return 'שגיאה: ' + data.responseText
                },
                onclickSubmit: function (params, posdata) {
                    AddRec(null, posdata);
                    return true;
                }
            },
        // options for the Search Dailog
        {

            closeAfterSearch: isValid,
            searchOperators: true,
            stringResult: true,
            searchOnEnter: true,
            search: true,
            sopt: ['eq', 'ne', 'cn'],
            onSearch: function () {
                $("#jqGrid").jqGrid("clearGridData");
                var i, l, rules, rule, $grid = $('#jqGrid'),
                    postData = $grid.jqGrid('getGridParam', 'postData'),
                    filters = $.parseJSON(postData.filters);

                if (filters && typeof filters.rules !== 'undefined' &&
filters.rules.length > 0) {
                    rules = filters.rules;
                    F_G_d(rules);
                    postData.filters = JSON.stringify(filters);
                }
            },
        });
```

```
            $('#jqGrid').hideCol('ContactId');
            $('#jqGrid').inlineNav('#jqGridPager',
              // the buttons to appear on the toolbar of the grid
              {
                  edit: true,
                  add: true,
                  del: true,
                  cancel: true,
                  editParams: {
                      keys: true,
                      aftersavefunc: function (rowid) {
                          DelRec(rowid, null)
                      }

                  },
                  addParams: {
                      keys: true,
                      position: "last",
                      addRowParams: {
                          aftersavefunc: function (rowid) {
                              MakeGR(rowid)
                          }
                      }
                  }

              });

            function getServerUrl() {
                var GRID_DATA = odataPath16 +=
"homL_system_parameterses?$select=homL_value,homL_VarN&$filter=homL_VarN eq
'Var_N_a'";
                var oDataUri = encodeURI(GRID_DATA);
                var res;
                $.ajax({
                    type: "GET",
                    contentType: "application/json; charset=utf-8",
                    datatype: "json",
                    url: oDataUri,
                    beforeSend: function (XMLHttpRequest) {
                        XMLHttpRequest.setRequestHeader("OData-MaxVersion",
"4.0");
                        XMLHttpRequest.setRequestHeader("OData-Version",
"4.0");
                        XMLHttpRequest.setRequestHeader("Accept",
"application/json");
                        XMLHttpRequest.setRequestHeader("Prefer",
"odata.include-annotations=\"*\"");
                    },
                    async: false,
                    success: function (data, textStatus, XmlHttpRequest) {
                        if (data.value.length > 0) {
                            res = data.value[0].homL_value;
                        }

                    },
                    error: function (XmlHttpRequest, textStatus, errorThrown) {
                        var message =
xmlHttpRequest.responseJSON.error.message.value;
                        alert(message);
                    }

                });
```

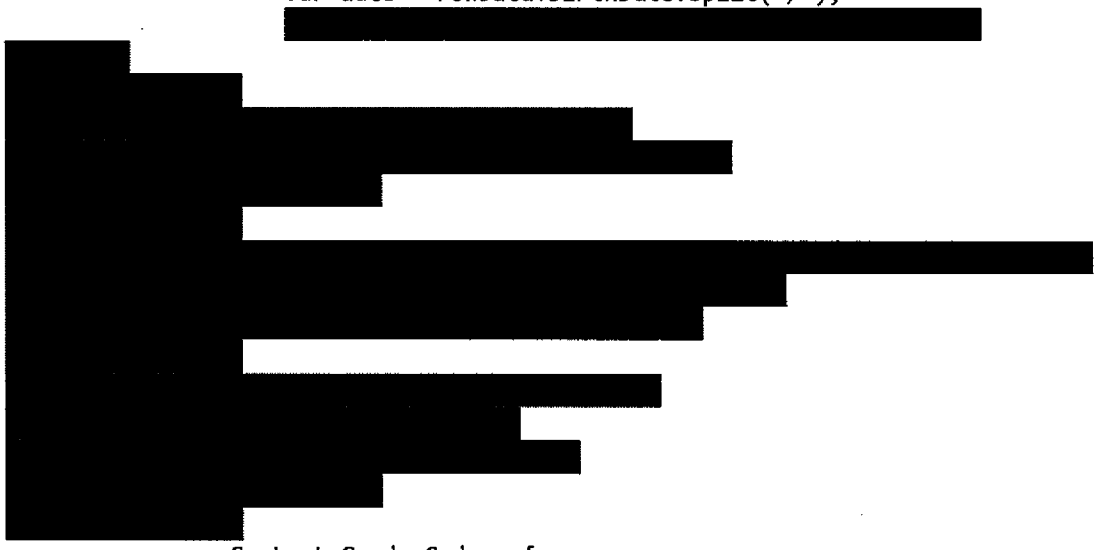
```
            return res;
        }

        //function to lookup fieled

        function Get_data_f(cellValue, options, rowObject) {
            var str = "";
            var Id;
            var searchEntity = "";
            var entityVarNOpenDialog = options.colModel.lookupVarN;
            var style = "display:none;border:none;width:15 px;"
            style = "border:none;width:15 px;"
            var iconsearchurl = serverUrl +
"/WebResources/homL_searchicon";
            if (cellValue != undefined) {
                if (cellValue.indexOf("href") == -1) {
                    var indx = cellValue.lastIndexOf(";");
                    Id = cellValue.substring(indx + 1, cellValue.length);

                    searchEntity = entityVarNOpenDialog.toLowerCase();

                    cellValue = cellValue.substring(0, indx);
                    cellValue = cellValue.length > 15 ?
cellValue.substring(0, 15) + "..." : cellValue;
                    str = '<div style="direction:rtl;text-align:right;" >'
                        + '<a href="javascript:openNewJQgrid(\'' + Id +
'\');">' + cellValue + '<img id="imgsearch' + options.rowId + options.pos + '"
style=' + style + '" src="' + iconsearchurl + '"></a></div>';
                    return str;
                }
            }
            if (cellValue == undefined) {
                str = '<div style="direction:rtl;text-align:right;" >'
                    + '<a href="javascript:openNewJQgrid(\'' +
entityVarNOpenDialog + '\');"><img id="imgsearch' + options.rowId + options.pos
+ '"style=' + style + '" src="' + iconsearchurl + '"></a></div>';
                return str;
            }
            return cellValue;
        };

        function OpenEntity(guid, entityVarN) {
            return serverUrl + "/main.aspx?etn=" + entityVarN +
"&newWindow=true&pagetype=entityrecord&id=" + guid;
        }

        function setNullValue(item) {
            if (item == "null;null")
            { return ""; }
            return item;
        }

        function formatLink(cellValue, options, rowObject) {

            if (rowObject.ContactId) {
                return "<a target='_blank' href=" +
OpenEntity(rowObject.ContactId) + ">" + cellValue + "</a>";
            } else {
                return cellValue;
            }
        };
function getData(etityVarN, fieldVarN) {
```

```
                var vals;
                var acvals = '', html = '';
                DeleAtr(etityVarN, fieldVarN, null, true, function (data) {
                    var entityVarN, PrimaryIdAttribute, PrimaryVarNAttribute;
                    if (data._type == "LookupAttributeMetadata") {
                        for (var i = 0; i < data.Targets.length; i++) {
                            RetrieveEntity(1, data.Targets[i], null, true,
                                function (res) {

                                    entityVarN = res.SchemaVarN;
                                    DeleAtr(res.SchemaVarN,
res.PrimaryIdAttribute, null, true, function (attributeData) {
                                        PrimaryIdAttribute =
attributeData.SchemaVarN;

                                    }, function () {
                                        alert("error")
                                    }, null);
                                    DeleAtr(res.SchemaVarN,
res.PrimaryVarNAttribute, null, true, function (attributeData) {
                                        PrimaryVarNAttribute =
attributeData.SchemaVarN;

                                    $.getJSON(var_N_a + entityVarN +
'Set?$select=' + PrimaryIdAttribute + ',' + PrimaryVarNAttribute, function
(rows) {

                                        var len = rows.d.results.length, j,
item;
                                        //For default value empty
                                        if (len > 0 && html == '') {
                                            html += "" + ':' + ';';
                                        }

                                        for (j = 0; j < len; j++) {
                                            item = rows.d.results[j];
                                            //   html += item.AccountId +
':' + item.VarN + ';';
                                            html +=
item[PrimaryIdAttribute] + ':' + item[PrimaryVarNAttribute] + ';';
                                        }

                                        //Sub last ';'
                                        if (html.length > 0 && i ==
data.Targets.length - 1) {
                                            html = html.substr(0,
html.length - 1);
                                        }
                                        acvals = html;
                                        //if (i == data.Targets.length) {
                                        //    return acvals;
                                        //}
                                        $("#jqGrid").jqGrid('setColProp',
fieldVarN, {
                                            lookupVarN: entityVarN
                                        });

                                    });
                                }, function () {
                                    alert("error")
                                }, null);
                            },
```

```
                        function (response) {
                            alert(response);
                        }, null);
                    }
                }
                else if (data._type == "PicklistAttributeMetadata" ||
data._type == "StatusAttributeMetadata") {
                    if (data.OptionSet.Options.length > 0) {
                        html += "" + ':' + ';';
                    }
                    for (var i = 0; i < data.OptionSet.Options.length; i++)
{
                        html += data.OptionSet.Options[i].Value + ':' +
data.OptionSet.Options[i].Label.LocalizedLabels[0].Label + ';';
                    }
                    //Sub last ';'
                    if (html.length > 0) {
                        html = html.substr(0, html.length - 1);
                    }
                    acvals = html;
                }
                else if (data._type == "BooleanAttributeMetadata") {
                    acvals = "true:false";
                }
            },
            errorCallBack, null);
            return acvals;
        }

        function look_id(fieldVal) {
            var entRef = {};
            if (fieldVal != null && fieldVal != "" && fieldVal !=
undefined) {
                var id = fieldVal.split("(")[1];
                if (id != null && id != undefined) {
                    id = id.substring(1, 37);
                    var regex = /[a-f0-9]{8}(?:-[a-f0-9]{4}){3}-[a-f0-
9]{12}/i;
                    var match = regex.exec(id);
                    if (match != null) {
                        entRef.Id = id;
                        entRef.LogicalVarN = "el_city";
                        return entRef;
                    }
                }
            }
            return null;
        }


        //returns true if id number is valid according to Israeli id format
        function IsValidGovermentId(id) {
            var result = false;

            //test for valid input
            if ((id != null) && (id != "")) {
                //add preceding zeros if id does not contain 9 digits
                while (id.length < 9) {
                    id = '0' + id;
                }

                var pattern = /^\d{9}$/;
```

```
                    //validate 9 digits
                    if (pattern.test(id)) {
                        //verify validation digit
                        var totalSum = 0,
                            tempSum;

                        for (var i = 0; i < 9; i++) {
                            tempSum = Number(id[i]);
                            tempSum *= (i % 2) + 1;

                            if (tempSum > 9) {
                                tempSum -= 9;
                            }

                            totalSum += tempSum;
                        }

                        result = (totalSum % 10 == 0)
                    }
                }

                return result;
            };

            function person_VarN(jsonDate) {
                if (jsonDate != null) {
                    var offset = new Date().getTimezoneOffset();
                    var parts = /\/Date\((-?\d+)([+-
]\d{2})?(\d{2})?.*/.exec(jsonDate);

                    if (parts[2] == undefined)
                        parts[2] = 0;

                    if (parts[3] == undefined)
                        parts[3] = 0;
                    return new Date(+parts[1] + parts[2] * 3600000 + parts[3] *
60000);//.toLocaleDateString();
                }
            };

            function updateUrl() {
                var rowid = getSelectedRow();
                if (rowid) {
                    var rowData = $("#jqGrid").getRowData(rowid);
                    return "http://xrm13-
demo1/jqgridDev/XRMServices/2011/Organizationdata.svc/ContactSet(guid'" +
rowData.ContactId + "')";
                }
                return "";
            }

            function editRow(id) {
                var grid = $("#jqGrid");
                if (id && id !== lastSelection) {
                    grid.jqGrid('restoreRow', lastSelection, { keys: true });
                    lastSelection = id;
                }
                grid.jqGrid('editRow', id);
            }

            function editSuccessful() {
```

```
            alert("success");
        }  function editFailed() {
            alert("fail");
        }

        function MakeGR(rowid, posdata) {
            var rowData;
            if (posdata) {
                rowData = posdata;
            } else {
                rowData = $("#jqGrid").getRowData(rowid);
            }
            if (rowData.GenderCode == "זכר") {
                GenderCode = 1;
            }
            else if (rowData.GenderCode == "נקבה") {
                GenderCode = 2;
            }
            else if (rowData.GenderCode == "") {
                GenderCode = null;
            }

            var Contact = {};
            if (rowData.BirthDate && rowData.BirthDate.length > 1) {
                //Convert date to format mm/dd/yyy
                var date = rowData.BirthDate.split("/");
```



```
                Contact.GenderCode = {
                    "__metadata": { "type":
"Microsoft.Crm.Sdk.Data.Services.OptionSetValue" },
                    "Value": GenderCode,
                }
                Contact.FirstVarN = rowData.FirstVarN;

                Contact.LastVarN = rowData.LastVarN;
                //Contact.el_id_city_main = row.City;
                Contact.Address1_Line1 = rowData.Address;
                var city = look_id(rowData.City);
                if (city != null) {
                    Contact.el_id_city_main = {
                        Id: city.Id,
                        LogicalVarN: city.LogicalVarN
                    }
                    Contact.MobilePhone = rowData.MobilePhone;
                }
```

```
                Contact.MobilePhone = rowData.MobilePhone;
                var jsonEntity = JSON.stringify(Contact);
                var oDataURI = odataPath11 + "ContactSet"

                $.ajax({
                    type: "POST",
                    contentType: "application/json; charset=utf-8",
                    datatype: "json",
                    url: oDataURI,
                    data: jsonEntity,
                    beforeSend: function (XMLHttpRequest) {
                        XMLHttpRequest.setRequestHeader("Accept",
"application/json");
                    },
                    success: function (response) {
                        if (response != null && response.d != null) {
                            rowData.ContactId = response.d.ContactId;
                            if (rowid) {
                                $("#jqGrid").jqGrid('setRowData', rowid,
rowData);
                            }
                            alert("בהצלחה נוספה רשומה");
                        }
                    },
                    error: function (xmlHttpRequest, textStatus, errorThrown) {
                        alert("Status: " + textStatus + "; ErrorThrown: " +
errorThrown);
                    }
                });

            }

            function AddRec(rowid, posdata) {
                if (posdata) {
                    var rowData = $("#jqGrid").getRowData(posdata);
                    $("#jqGrid").jqGrid('delRowData', posdata);
                }
            }

            function DelRec(rowid, posdata) {
                isUpdate = true;
                var ContactId;
                var GenderCode;
                var rowData;
                if (posdata) {
                    rowData = posdata;
                    if (posdata.jqGrid_id) {
                        rowid = posdata.jqGrid_id;
                        ContactId =
$("#jqGrid").getRowData(posdata.jqGrid_id).ContactId;
                    }
                } else {
                    rowData = $("#jqGrid").getRowData(rowid);
                    ContactId = rowData.ContactId;
                }
                var GenderCode;
                if (rowData.GenderCode == "זכר") {
                    GenderCode = 1;
                }
                else if (rowData.GenderCode == "נקבה") {
                    GenderCode = 2;
                }
```

```
                else if (rowData.GenderCode == "") {
                    GenderCode = null;
                }
                var Contact = {};
                if (!ContactId) {
                    if (rowid)
                        MakeGR(rowid, null);
                    return;
                }
                if (rowData.BirthDate && rowData.BirthDate.length > 1) {
                    //Convert date to format mm/dd/yyy
                    var date = rowData.BirthDate.split("/");
                    Contact.BirthDate = new Date(date[2], date[1] - 1,
date[0]);
                }
```

```
                Contact.FirstVarN = rowData.FirstVarN;
                if (rowData.LastVarN == null) {
                    alert("משפחה שם עבור ערך להזין עליך")
                    return;
                }
                Contact.GenderCode = {
                    "__metadata": { "type":
"Microsoft.Crm.Sdk.Data.Services.OptionSetValue" },
                    "Value": GenderCode,
                }
                Contact.LastVarN = rowData.LastVarN;
                Contact.MobilePhone = rowData.MobilePhone;
                Contact.EMailAddress1 = rowData.EMailAddress1;
                Contact.Address1_Line1 = rowData.Address;
                var city = look_id(rowData.City);
                if (city != null) {
                    Contact.el_id_city_main = {
                        Id: city.Id,
                        LogicalVarN: city.LogicalVarN
                    }
                }

                var dealSON = JSON.stringify(Contact);
                var oDataURI = odataPath11 + "ContactSet"
                if (ContactId) {
                    oDataURI += "(guid'" + ContactId + "')";
                }
                //Synchronous post
                var req = new XMLHttpRequest();
                req.open("POST", encodeURI(oDataURI), !ContactId);
                req.setRequestHeader("Accept", "application/json");
                req.setRequestHeader("Content-Type", "application/json;
charset=utf-8");
                req.setRequestHeader("X-HTTP-Method", "MERGE");
                req.onreadystatechange = function () {
                    if (this.readyState == 4 /* complete */) {
```

```
                        req.onreadystatechange = null;
                        if (this.status == 204 || this.status == 1223) {
                            if (rowid)
                                $("#jqGrid").jqGrid('setRowData', rowid,
rowData);
                            alert("בהצלחה בוצע עדכון");
                        }
                        else {
                            alert("שגיאה " + this.responseText);
                        }
                    }
                };
                req.send(dealSON);
                //// refresh the grid
                $("#jqGrid").trigger('reloadGrid');
            }
//fonctoin to get data from metadata
            function getPicklistLabel(etityVarN, fieldVarN, value) {

                var label;
                DeleAtr(etityVarN, fieldVarN, null, true, function (data) {
                    var entityVarN, PrimaryIdAttribute, PrimaryVarNAttribute;

                    if (data._type == "PicklistAttributeMetadata" || data._type
== "StatusAttributeMetadata") {
                        var options = data.OptionSet.Options;
                        for (var i = 0; i < options.length; i++) {
                            if (options[i].Value == value) {
                                label =
options[i].Label.LocalizedLabels[0].Label;
                            }

                        }

                    }
                },
                    errorCallBack, null);
                return label;
            }

            function errorCallBack(response) {

            }

            function DeleAtr(EntityLogicalVarN, LogicalVarN, MetadataId,
RetrieveAsIfPublished, successCallBack, errorCallBack, passThroughObject) {
                ///<summary>
                /// Sends an asynchronous DeleAtr Request to retrieve a
specific entity
                ///</summary>
                ///<returns>AttributeMetadata</returns>
                ///<param VarN="EntityLogicalVarN"  optional="true"
type="String">
                /// The logical VarN of the entity for the attribute requested.
A null value may be used if a MetadataId is provided.
                ///</param>
                ///<param VarN="LogicalVarN" optional="true" type="String">
                /// The logical VarN of the attribute requested.
                ///</param>
                ///<param VarN="MetadataId" optional="true" type="String">
                /// A null value may be passed if an EntityLogicalVarN and
LogicalVarN is provided.
```

```
                ///</param>
                ///<param VarN="RetrieveAsIfPublished" type="Boolean">
                /// Sets whether to retrieve the metadata that has not been
published.
                ///</param>
                ///<param VarN="successCallBack" type="Function">
                /// The function that will be passed through and be called by a
successful response.
                /// This function must accept the entityMetadata as a
parameter.
                ///</param>
                ///<param VarN="errorCallBack" type="Function">
                /// The function that will be passed through and be called by a
failed response.
                /// This function must accept an Error object as a parameter.
                ///</param>
                ///<param VarN="passThroughObject" optional="true"
type="Object">
                /// An Object that will be passed through to as the second
parameter to the successCallBack.
                ///</param>
                if (EntityLogicalVarN == null && LogicalVarN == null &&
MetadataId == null) {
                    throw new Error("SDK.Metadata.DeleAtr requires either the
EntityLogicalVarN and LogicalVarN  parameters or the MetadataId parameter not
be null.");
                }
                if (MetadataId != null && EntityLogicalVarN == null &&
LogicalVarN == null) {
                    if (typeof MetadataId != "string")
                    { throw new Error("SDK.Metadata.RetrieveEntity MetadataId
must be a string value."); }
                }
                else { MetadataId = "00000000-0000-0000-0000-000000000000"; }
                if (EntityLogicalVarN != null) {
                    if (typeof EntityLogicalVarN != "string") {
                        { throw new Error("SDK.Metadata.DeleAtr
EntityLogicalVarN must be a string value."); }
                    }
                }
                if (LogicalVarN != null) {
                    if (typeof LogicalVarN != "string") {
                        { throw new Error("SDK.Metadata.DeleAtr LogicalVarN
must be a string value."); }
                    }

                }
                if (typeof RetrieveAsIfPublished != "boolean")
                { throw new Error("SDK.Metadata.DeleAtr RetrieveAsIfPublished
must be a boolean value."); }
                if (typeof successCallBack != "function")
                { throw new Error("SDK.Metadata.DeleAtr successCallBack must be
a function."); }
                if (typeof errorCallBack != "function")
                { throw new Error("SDK.Metadata.DeleAtr errorCallBack must be a
function."); }

                var entityLogicalVarNValueNode;
                if (EntityLogicalVarN == null) {
                    entityLogicalVarNValueNode = "<b:value i:nil=\"true\" />";
                }
                else {
```

```
                    entityLogicalVarNValueNode = "<b:value i:type=\"c:string\"
xmlns:c=\"http://www.w3.org/2001/XMLSchema\">" +
_xmlEncode(EntityLogicalVarN.toLowerCase()) + "</b:value>";
                }
                var logicalVarNValueNode;
                if (LogicalVarN == null) {
                    logicalVarNValueNode = "<b:value i:nil=\"true\" />";
                }
                else {
                    logicalVarNValueNode = "<b:value i:type=\"c:string\"
xmlns:c=\"http://www.w3.org/2001/XMLSchema\">" +
_xmlEncode(LogicalVarN.toLowerCase()) + "</b:value>";
                }
                var request = [
                "<soapenv:Envelope
xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\">",
                //Allows retrieval if ImageAttributeMetadata objects
                "<soapenv:Header><a:SdkClientVersion
xmlns:a=\"http://schemas.microsoft.com/xrm/2011/Contracts\">6.0</a:SdkClientVer
sion></soapenv:Header>",
                 "<soapenv:Body>",
                  "<Execute
xmlns=\"http://schemas.microsoft.com/xrm/2011/Contracts/Services\"
xmlns:i=\"http://www.w3.org/2001/XMLSchema-instance\">",
                   "<request i:type=\"a:DeleAtrRequest\"
xmlns:a=\"http://schemas.microsoft.com/xrm/2011/Contracts\">",
                    "<a:Parameters
xmlns:b=\"http://schemas.datacontract.org/2004/07/System.Collections.Generic\">
",
                     "<a:KeyValuePairOfstringanyType>",
                      "<b:key>EntityLogicalVarN</b:key>",
                       entityLogicalVarNValueNode,
                     "</a:KeyValuePairOfstringanyType>",
                     "<a:KeyValuePairOfstringanyType>",
                      "<b:key>MetadataId</b:key>",
                      "<b:value i:type=\"ser:guid\"
xmlns:ser=\"http://schemas.microsoft.com/2003/10/Serialization/\">" +
_xmlEncode(MetadataId) + "</b:value>",
                     "</a:KeyValuePairOfstringanyType>",
                      "<a:KeyValuePairOfstringanyType>",
                      "<b:key>RetrieveAsIfPublished</b:key>",
                    "<b:value i:type=\"c:boolean\"
xmlns:c=\"http://www.w3.org/2001/XMLSchema\">" +
_xmlEncode(RetrieveAsIfPublished.toString()) + "</b:value>",
                     "</a:KeyValuePairOfstringanyType>",
                     "<a:KeyValuePairOfstringanyType>",
                      "<b:key>LogicalVarN</b:key>",
                       logicalVarNValueNode,
                     "</a:KeyValuePairOfstringanyType>",
                    "</a:Parameters>",
                    "<a:RequestId i:nil=\"true\" />",
                    "<a:RequestVarN>DeleAtr</a:RequestVarN>",
                   "</request>",
                  "</Execute>",
                 "</soapenv:Body>",
                "</soapenv:Envelope>"].join("");
                var req = new XMLHttpRequest();
                req.open("POST", _getUrl() +
"/XRMServices/2011/Organization.svc/web", false);
                try { req.responseType = 'msxml-document' } catch (e) { }
                req.setRequestHeader("Accept", "application/xml, text/xml,
*/*");
```

```
            req.setRequestHeader("Content-Type", "text/xml; charset=utf-
8");
            req.setRequestHeader("SOAPAction",
"http://schemas.microsoft.com/xrm/2011/Contracts/Services/IOrganizationService/
Execute");
            req.onreadystatechange = function () {
                if (req.readyState == 4 /* complete */) {
                    req.onreadystatechange = null; //Addresses potential
memory leak issue with IE
                    if (req.status == 200) {
                        //Success
                        var doc = req.responseXML;
                        try { _setSelectionVarNspaces(doc); } catch (e) { }
                        var a = _objectifyNode(_selectSingleNode(doc,
"//b:value"));

                        successCallBack(a, passThroughObject);
                    }
                    else {
                        //Failure
                        errorCallBack(_getError(req));
                    }
                }
            };
            req.send(request);


        };
        function RetrieveEntity(EntityFilters, LogicalVarN, MetadataId,
RetrieveAsIfPublished, successCallBack, errorCallBack, passThroughObject) {
            ///<summary>
            /// Sends an asynchronous RetrieveEntity Request to retrieve a
specific entity
            ///</summary>
            ///<returns>entityMetadata</returns>
            ///<param VarN="EntityFilters" type="Number">
            /// SDK.Metadata.EntityFilters provides an enumeration for the
filters available to filter which data is retrieved.
            /// Include only those elements of the entity you want to
retrieve. Retrieving all parts of all entitities may take significant time.
            ///</param>
            ///<param VarN="LogicalVarN" optional="true" type="String">
            /// The logical VarN of the entity requested. A null value may
be used if a MetadataId is provided.
            ///</param>
            ///<param VarN="MetadataId" optional="true" type="String">
            /// A null value or an empty guid may be passed if a
LogicalVarN is provided.
            ///</param>
            ///<param VarN="RetrieveAsIfPublished" type="Boolean">
            /// Sets whether to retrieve the metadata that has not been
published.
            ///</param>
            ///<param VarN="successCallBack" type="Function">
            /// The function that will be passed through and be called by a
successful response.
            /// This function must accept the entityMetadata as a
parameter.
            ///</param>
            ///<param VarN="errorCallBack" type="Function">
            /// The function that will be passed through and be called by a
failed response.
```

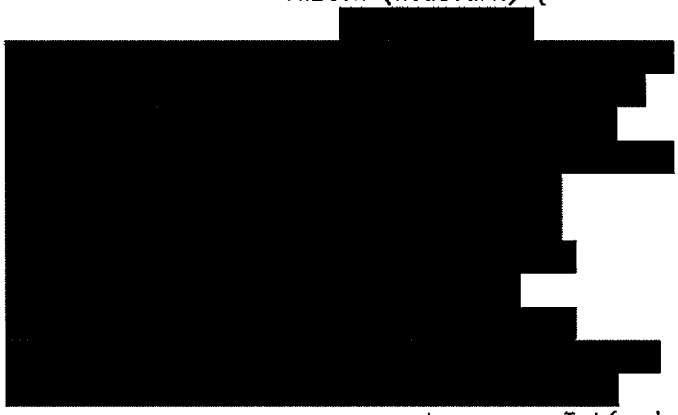
```
                /// This function must accept an Error object as a parameter.
                ///</param>
                ///<param VarN="passThroughObject" optional="true"
type="Object">
                /// An Object that will be passed through to as the second
parameter to the successCallBack.
                ///</param>
                if ((typeof EntityFilters != "number") || (EntityFilters < 1 ||
EntityFilters > 15))
                { throw new Error("SDK.Metadata.RetrieveEntity EntityFilters
must be a SDK.Metadata.EntityFilters value."); }
                if (LogicalVarN == null && MetadataId == null) {
                    throw new Error("SDK.Metadata.RetrieveEntity requires
either the LogicalVarN or MetadataId parameter not be null.");
                }
                if (LogicalVarN != null) {
                    if (typeof LogicalVarN != "string")
                    { throw new Error("SDK.Metadata.RetrieveEntity LogicalVarN
must be a string value."); }
                    MetadataId = "00000000-0000-0000-0000-000000000000";
                }
                if (MetadataId != null && LogicalVarN == null) {
                    if (typeof MetadataId != "string")
                    { throw new Error("SDK.Metadata.RetrieveEntity MetadataId
must be a string value."); }
                }
                if (typeof RetrieveAsIfPublished != "boolean")
                { throw new Error("SDK.Metadata.RetrieveEntity
RetrieveAsIfPublished must be a boolean value."); }
                if (typeof successCallBack != "function")
                { throw new Error("SDK.Metadata.RetrieveEntity successCallBack
must be a function."); }
                if (typeof errorCallBack != "function")
                { throw new Error("SDK.Metadata.RetrieveEntity errorCallBack
must be a function."); }
                var entityFiltersValue = _evaluateEntityFilters(EntityFilters);

                var entityLogicalVarNValueNode = "";
                if (LogicalVarN == null)
                { entityLogicalVarNValueNode = "<b:value i:nil=\"true\" />"; }
                else
                { entityLogicalVarNValueNode = "<b:value i:type=\"c:string\"
xmlns:c=\"http://www.w3.org/2001/XMLSchema\">" +
_xmlEncode(LogicalVarN.toLowerCase()) + "</b:value>"; }
                var request = [
                <soapenv:Envelope
xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\">",
                  //Allows retrieval if ImageAttributeMetadata objects
                <soapenv:Header><a:SdkClientVersion
xmlns:a=\"http://schemas.microsoft.com/xrm/2011/Contracts\">6.0</a:SdkClientVer
sion></soapenv:Header>",
                 <soapenv:Body>",
                  <Execute
xmlns=\"http://schemas.microsoft.com/xrm/2011/Contracts/Services\"
xmlns:i=\"http://www.w3.org/2001/XMLSchema-instance\">",
                   "<request i:type=\"a:RetrieveEntityRequest\"
xmlns:a=\"http://schemas.microsoft.com/xrm/2011/Contracts\">",
                    <a:Parameters
xmlns:b=\"http://schemas.datacontract.org/2004/07/System.Collections.Generic\">
,
                     <a:KeyValuePairOfstringanyType>",
                      <b:key>EntityFilters</b:key>",
```

```
                    <b:value i:type=\"c:EntityFilters\
xmlns:c=\"http://schemas.microsoft.com/xrm/2011/Metadata\"> +
_xmlEncode(entityFiltersValue) + "</b:value>,
                   </a:KeyValuePairOfstringanyType>,
                   <a:KeyValuePairOfstringanyType>,
                    <b:key>MetadataId</b:key>,
                    <b:value i:type=\"ser:guid\
xmlns:ser=\"http://schemas.microsoft.com/2003/10/Serialization/\">" +
_xmlEncode(MetadataId) + "</b:value>,
                   </a:KeyValuePairOfstringanyType>,
                   <a:KeyValuePairOfstringanyType>,
                    <b:key>RetrieveAsIfPublished</b:key>,
                    <b:value i:type=\"c:boolean\
xmlns:c=\"http://www.w3.org/2001/XMLSchema\"> +
_xmlEncode(RetrieveAsIfPublished.toString()) + "</b:value>,
                   </a:KeyValuePairOfstringanyType>,
                   <a:KeyValuePairOfstringanyType>,
                    <b:key>LogicalVarN</b:key>,
                     entityLogicalVarNValueNode,
                   </a:KeyValuePairOfstringanyType>,
                  </a:Parameters>",
                  <a:RequestId i:nil=\"true\" />,
                  <a:RequestVarN>RetrieveEntity</a:RequestVarN>,
                 </request>,
                </Execute>,
               </soapenv:Body>",
              </soapenv:Envelope>].join("");
              var req = new XMLHttpRequest();
              /// Changed by Neomy
              //req.open("POST", _getUrl() +
"/XRMServices/2011/Organization.svc/web", true);
              req.open("POST", _getUrl() +
"/XRMServices/2011/Organization.svc/web", false);
              try { req.responseType = 'msxml-document' } catch (e) { }
              req.setRequestHeader("Accept", "application/xml, text/xml,
*/*");
              req.setRequestHeader("Content-Type", "text/xml; charset=utf-
8");
              req.setRequestHeader("SOAPAction",
"http://schemas.microsoft.com/xrm/2011/Contracts/Services/IOrganizationService/
Execute");
              req.onreadystatechange = function () {
                  if (req.readyState == 4 /* complete */) {
                      req.onreadystatechange = null; //Addresses potential
memory leak issue with IE
                      if (req.status == 200) {
                          var doc = req.responseXML;
                          try { _setSelectionVarNspaces(doc); } catch (e) { }
                          var a = _objectifyNode(_selectSingleNode(doc,
"//b:value"));
                          a._type = "EntityMetadata";
                          successCallBack(a, passThroughObject);
                      }
                      else {
                          //Failure
                          errorCallBack(_getError(req));
                      }
                  }

              };
              req.send(request);
```

```
        };
  function _getError(resp) {
                ///<summary>
                /// Private function that attempts to parse errors related to
connectivity or WCF faults.
                ///</summary>
                ///<param VarN="resp" type="XMLHttpRequest">
                /// The XMLHttpRequest representing failed response.
                ///</param>

                //Error descriptions come from
http://support.microsoft.com/kb/193625
                if (resp.status == 12029)
                { return new Error("The attempt to connect to the server
failed."); }
                if (resp.status == 12007)
                { return new Error("The server VarN could not be resolved."); }
                var faultXml = resp.responseXML;
                var errorMessage = "Unknown (unable to parse the fault)";
                if (typeof faultXml == "object") {

                    var faultstring = null;
                    var ErrorCode = null;

                    var bodyNode = faultXml.firstChild.firstChild;

                    //Retrieve the fault node
                    for (var i = 0; i < bodyNode.childNodes.length; i++) {
                        var node = bodyNode.childNodes[i];

                        //NOTE: This comparison does not handle the case where
the XML VarNspace changes
                        if ("s:Fault" == node.nodeVarN) {
                            for (var j = 0; j < node.childNodes.length; j++) {
                                var testNode = node.childNodes[j];
                                if ("faultstring" == testNode.nodeVarN) {
                                    faultstring = _getNodeText(testNode);
                                }
                                if ("detail" == testNode.nodeVarN) {
                                    for (var k = 0; k <
testNode.childNodes.length; k++) {
                                        var orgServiceFault =
testNode.childNodes[k];
                                        if ("OrganizationServiceFault" ==
orgServiceFault.nodeVarN) {
                                            for (var l = 0; l <
orgServiceFault.childNodes.length; l++) {
                                                var ErrorCodeNode =
orgServiceFault.childNodes[l];
                                                if ("ErrorCode" ==
ErrorCodeNode.nodeVarN) {
                                                    ErrorCode =
_getNodeText(ErrorCodeNode);
                                                    break;
                                                }
                                            }
                                        }
                                    }

                                }
                            }
```

```
                            break;
                        }

                    }
                }
                if (ErrorCode != null && faultstring != null) {
                    errorMessage = "Error Code:" + ErrorCode + " Message: " +
faultstring;
                }
                else {
                    if (faultstring != null) {
                        errorMessage = faultstring;
                    }
                }
                return new Error(errorMessage);
            };

            function _Context() {
                var errorMessage = "Context is not available.";
                if (typeof GetGlobalContext != "undefined")
                { return GetGlobalContext(); }
                else {
                    if (typeof Xrm != "undefined") {
                        return Xrm.Page.context;
                    }
                    else { return new Error(errorMessage); }
                }
            };

            function _getUrl() {
                var url = var_N_a;
                return url;
            };

            function _evaluateEntityFilters(EntityFilters) {
                var entityFilterArray = [];
                if ((1 & EntityFilters) == 1) {
                    entityFilterArray.push("Entity");
                }
                if ((2 & EntityFilters) == 2) {
                    entityFilterArray.push("Attributes");
                }
                if ((4 & EntityFilters) == 4) {
                    entityFilterArray.push("Privileges");
                }
                if ((8 & EntityFilters) == 8) {
                    entityFilterArray.push("Relationships");
                }
                return entityFilterArray.join(" ");
            };
    function _is_M_D(elementVarN) {
                var _arrayElements = ["Attributes",
                                      "ManyToManyRelationships",
                                      "ManyToOneRelationships",
                                      "OneToManyRelationships",
                                      "Privileges",
                                      "LocalizedLabels",
                                      "Options",
                                      "Targets"];
                for (var i = 0; i < _arrayElements.length; i++) {
                    if (elementVarN == _arrayElements[i]) {
                        return true;
```

```
                }
            }
            return false;
        };

        function _objectifyNode(node) {
            Check for null
            if (node.attributes != null && node.attributes.length == 1) {
                if (node.attributes.getVarNdItem("i:nil") != null &&
node.attributes.getVarNdItem("i:nil").nodeValue == "true") {
                    return null;
                }
            }

            Check if it is a value
            if ((node.firstChild != null) && (node.firstChild.nodeType ==
3)) {
                var nodeVarN = _getNodeVarN(node);

                switch (nodeVarN) {
```



```
                    return parseInt(node.firstChild.nodeValue, 10);
                    // Boolean values
                case "AutoRouteToOwnerQueue":
                case "CanBeChanged":
                case "CanTriggerWorkflow":
                case "IsActivity":
                case "IsAIRUpdated":
                case "IsActivityParty":
                case "IsAvailableOffline":
                case "IsChildEntity":
                case "IsCustomEntity":
                case "IsCustomOptionSet":
                case "IsDocumentManagementEnabled":
                case "IsEnabledForCharts":
                case "IsGlobal":
                case "IsImportable":
                case "IsIntersect":
                case "IsManaged":
                case "IsReadingPaneEnabled":
                case "IsValidForAdvancedFind":
                case "CanBeSecuredForCreate":
                case "CanBeSecuredForRead":
                case "CanBeSecuredForUpdate":
                case "IsCustomAttribute":
                case "IsManaged":
                case "IsPrimaryId":
                case "IsPrimaryVarN":
                case "IsSecured":
                case "IsValidForCreate":
```

```
                            return (node.firstChild.nodeValue == "true") ? true
: false;
                            //OptionMetadata.Value and
BooleanManagedProperty.Value and AttributeRequiredLevelManagedProperty.Value
                        case "Value":
                            //BooleanManagedProperty.Value
                            if ((node.firstChild.nodeValue == "true") ||
(node.firstChild.nodeValue == "false")) {
                                return (node.firstChild.nodeValue == "true") ?
true : false;
                            }
                            //AttributeRequiredLevelManagedProperty.Value
                            if (
                             (node.firstChild.nodeValue == "ApplicationRequired")
||
                             (node.firstChild.nodeValue == "None") ||
                             (node.firstChild.nodeValue == "Recommended") ||
                             (node.firstChild.nodeValue == "SystemRequired")
                             ) {
                                return node.firstChild.nodeValue;
                            }
                            var numberValue =
parseInt(node.firstChild.nodeValue, 10);
                            if (isNaN(numberValue)) {
                                //FormatVarN.Value
                                return node.firstChild.nodeValue;
                            }
                            else {
                                //OptionMetadata.Value
                                return numberValue;
                            }
                            break;
                            //String values
                        default:
                            return node.firstChild.nodeValue;
                    }

                }

                //Check if it is a known array
                if (_is_M_D(_getNodeVarN(node))) {
                    var arrayValue = [];

                    for (var i = 0; i < node.childNodes.length; i++) {
                        var objectTypeVarN;
                        if ((node.childNodes[i].attributes != null) &&
(node.childNodes[i].attributes.getVarNdItem("i:type") != null)) {
                            objectTypeVarN =
node.childNodes[i].attributes.getVarNdItem("i:type").nodeValue.split(":")[1];
                        }
                        else {

                            objectTypeVarN = _getNodeVarN(node.childNodes[i]);
                        }
```

```
                    var b = _objectifyNode(node.childNodes[i]);
                    b._type = objectTypeVarN;
                    arrayValue.push(b);

                }

                return arrayValue;
            }

            //Null entity description labels are returned as <label/> - not
using i:nil = true;
            if (node.childNodes.length == 0) {
                return null;
            }

            //Otherwise return an object
            var c = {};
            if (node.attributes.getVarNdItem("i:type") != null) {
                c._type =
node.attributes.getVarNdItem("i:type").nodeValue.split(":")[1];
            }
            for (var i = 0; i < node.childNodes.length; i++) {
                if (node.childNodes[i].nodeType == 3) {
                    c[_getNodeVarN(node.childNodes[i])] =
node.childNodes[i].nodeValue;
                }
                else {
                    c[_getNodeVarN(node.childNodes[i])] =
_objectifyNode(node.childNodes[i]);
                }

            }
            return c;
        };

        function _selectNodes(node, XPathExpression) {
            if (typeof (node.selectNodes) != "undefined") {
                return node.selectNodes(XPathExpression);
            }
            else {
                var out_a = [];
                var XPathResults = node.evaluate(XPathExpression, node,
_NSResolver, XPathResult.ANY_TYPE, null);
                var result = XPathResults.iterateNext();
                while (result) {
                    out_a.push(result);
                    result = XPathResults.iterateNext();
                }
                return out_a;
            }
        };

        function _selectSingleNode(node, xpathExpr) {
            if (typeof (node.selectSingleNode) != "undefined") {
                return node.selectSingleNode(xpathExpr);
            }
            else {
                var xpe = new XPathEvaluator();
                var m_p_t_N = xpe.evaluate(xpathExpr, node, _NSResolver,
XPathResult.FIRST_ORDERED_NODE_TYPE, null);
                return (m_p_t_N != null) ? m_p_t_N.singleNodeValue : null;
```

```
            }
        };

        function _selectSingleNodeText(node, xpathExpr) {
            var x = _selectSingleNode(node, xpathExpr);
            if (_isNodeNull(x))
            { return null; }
            if (typeof (x.text) != "undefined") {
                return x.text;
            }
            else {
                return x.textContent;
            }
        };
function _getNodeText(node) {
            if (typeof (node.text) != "undefined") {
                return node.text;
            }
            else {
                return node.textContent;
            }
        };

        function _isNodeNull(node) {
            if (node == null)
            { return true; }
            if ((node.attributes.getVarNdItem("i:nil") != null) &&
(node.attributes.getVarNdItem("i:nil").value == "true"))
            { return true; }
            return false;
        };

        function _getNodeVarN(node) {
            if (typeof (node.baseVarN) != "undefined") {
                return node.baseVarN;
            }
            else {
                return node.localVarN;
            }
        };

        function _setSelectionVarNspaces(doc) {
            var VarNspaces = [
            "xmlns:s='http://schemas.xmlsoap.org/soap/envelope/'",
[redacted]
            ];
            doc.setProperty("SelectionVarNspaces", VarNspaces.join(" "));

        }

        function _NSResolver(prefix) {
            var ns = {
                "s": "http://schemas.xmlsoap.org/soap/envelope/",
                "a": "http://schemas.microsoft.com/xrm/2011/Contracts",
                "i": "http://www.w3.org/2001/XMLSchema-instance",
                "b":
"http://schemas.datacontract.org/2004/07/System.Collections.Generic",
                "c": "http://schemas.microsoft.com/xrm/2011/Metadata"
```

```
            };
            return ns[prefix] || null;
        };

        function _xmlEncode(strInput) {
            var c;
            var XmlEncode = '';
            if (strInput == null) {
                return null;
            }
            if (strInput == '') {
                return '';
            }
            for (var cnt = 0; cnt < strInput.length; cnt++) {
                c = strInput.charCodeAt(cnt);
                if (((c > 96) && (c < 123)) ||
                        ((c > 64) && (c < 91)) ||
                        (c == 32) ||
                        ((c > 47) && (c < 58)) ||
                        (c == 46) ||
                        (c == 44) ||
                        (c == 45) ||
                        (c == 95)) {
                    XmlEncode = XmlEncode + String.fromCharCode(c);
                }
                else {
                    XmlEncode = XmlEncode + '&#' + c + ';';
                }
            }
            return XmlEncode;
        };


$("#search_jqGrid").click();
            $("#searchmodfbox_jqGrid").width(500);
            $("#searchmodfbox_jqGrid").height(115);
            $("#fbox_jqGrid_reset").hide();
            $("#searchmodfbox_jqGrid").css('top', 5 + 'px');
            $("#searchmodfbox_jqGrid").css('left', 5 + 'px');
            $("#jqg1").width(100);
            $(".ui-jqgrid-bdiv").height(100);
            //  $("#del_jqGrid").hide();
            $("#add_jqGrid").hide();
            $("#edit_jqGrid").hide();
            //alerthd_jqGrid
            //del_jqGrid

        });

        //on click city colum open new window with cities view
        function openNewJQgrid() {
            var subStringToCheck = "input";
            var stringToCheck =
$("#jqGrid").getRowData($("#jqGrid").jqGrid('getGridParam',
'selrow')).FirstVarN;
            if (stringToCheck.indexOf(subStringToCheck) != -1) {
                var url =
serverUrl+"/WebResources/be_jqGrid_view_cities?preview=1";
                var myObject = new Object();
                myObject.Org = org;
```

```
                var result = showModalDialog(url, myObject,
"dialogHeight:360px;dialogWidth:500px;center:yes;
resizable:1;status:no;scroll:no");
                if (result != undefined) {
                    $("#jqGrid").jqGrid("setCell",
$("#jqGrid").jqGrid('getGridParam', 'selrow'), "City", result.retVal);
                }
            }
        }

        function getAccountId() {
            //Get the any query string parameters and load them
            //into the vals array
            var vals = new Array();
            if (location.search != "") {
                vals = location.search.substr(1).split("&");
                for (var i in vals) {
                    vals[i] = vals[i].replace(/\+/g, " ").split("=");
                }
                //look for the parameter VarNd 'data'
                var found = false;
                for (var i in vals) {
                    if (vals[i][0].toLowerCase() == "data") {
                        return vals[i][1];
                    }
                }
            }
        }

    </script>



</body>
                                        </html>
```

be_showPreviousIncident

```
<html>
<head>
    <meta http-equiv="x-ua-compatible" content="IE=9 ie=edge">
    <meta charset="utf-8">
    <script src="homL_jquery" type="text/javascript"></script>
    <script src="homL_json" type="text/javascript"></script>
    <script src="homL_jquery.jqGridTest" type="text/javascript"></script>
    <script src="homL_grid.localeenTest" type="text/javascript"></script>
    <script src="homL_jquery.ui" type="text/javascript"></script>
    <script src="homL_jquery.ui" type="text/javascript"></script>
    <link href="homL_jquery_ui" rel="stylesheet" type="text/css"
media="screen">
    <!-- The link to the CSS that the grid needs -->
    <link href="homL_ui_jqgrid" rel="stylesheet" type="text/css"
media="screen">

</head>
<body style="direction: rtl;">
    <div align="right">
        <table id="jqGrid"></table>
    </div>
    <div align="center" id="jqGridPager"></div>
    <script type="text/javascript">

        var serverUrl;
        var odataPath;
        var org;
        var resultObj;

        $(document).ready(function () {
          // debugger;
            var lastSelection;
            var PR = window.location.PR;
            var hostVarN = window.location.hostVarN;
            var path = window.location.pathVarN.split("/");
            org = path[1];
            serverUrl = PR + "//" + hostVarN + "/" + org;
            odataPath = serverUrl + "/api/data/v8.0/";

            $.ajaxSetup({
                async: false
            });

            $("#jqGrid").jqGrid({
                datatype: "local",
                editurl: "clientArray",
[redacted]
                colModel: [
                { label: 'פנימי מספר', VarN: 'be_autonumber', sorttype: 'string',
width: 100 },
                    {
                        label: 'ת. הפגישה', VarN: 'be_meetingdate', width:
150, editable: true, formatter: 'SortableDateTime',
                        formatoptions: { srcformat: 'datetime', newformat:
'd/m/Y' },
```

```
                        editoptions: {
                            dataInit: function (element) {
                                $(element).datepicker({
                                    formatter: 'date',
                                    formatoptions: {
                                        srcformat: 'm/d/Y',
                                        newformat: 'd/m/Y'
                                    },
                                    id: 'receptionDate_datePicker',
                                    dateFormat: 'dd/mm/yy',
                                    showOn: 'focus'
                                });
                            }
                        }
                    },

                    { label: 'פנית סוג', VarN: 'be_incidenttype', sorttype:
'number', width: 150 },
                    { label: 'מטפל מרכז', VarN: 'be_businessunitid', sorttype:
'string', width: 115 },
                    { label: 'פנית סטאטוס', VarN: 'statuscode', sorttype:
[redacted]
                    { label: 'ישות סוג', VarN: 'type', sorttype: 'string', width:
1, hidden: true },
                ],
                loadonce: true,
                viewrecords: true,
                width: 750,
                height: 90,
                rowNum: 4,
                rownumbers: false,
                direction: "rtl",
                pager: "#jqGridPager",
                ondblClickRow: openDB,
                multiselect: false,

            });

            function openDB() {
                $("#jqGrid").jqGrid('saveRow', lastSelection);
                var grid = $("#jqGrid");
                var rowKey = grid.getGridParam("selrow");
                // var selectedIDs = grid.getGridParam("selarrrow");
                rowData = $("#jqGrid").getRowData(rowKey);
                var recordId = rowData.recordId;
                var entityVarN = rowData.type;
                OpenRecord(serverUrl, entityVarN, recordId);
            }

            function OpenRecord(serverUrl, entityVarN, recordId) {
                window.open(serverUrl + "/main.aspx?etn=" + entityVarN +
"&pagetype=entityrecord&id=" + recordId);
            }


            F_G_d();

            function F_G_d() {
                var gridArrayData = [];
                $("#jqGrid")[0].grid.beginReq();
```

```
            isAsync = typeof isAsync !== 'undefined' ? isAsync : false;
            jQuery.support.cors = true;
            setPreviousPatientInciden();
        };

    });

    //set all the inciden of patient
    function setPreviousPatientInciden() {
        var gridArrayData = [];
        var index = 0;
        $("#jqGrid").jqGrid("clearGridData");
        var GRID_DATA = odataPath +=
"be_benefits_exhaustion_incidents?$select=be_autonumber,be_benefits_exhaustion_
incidentid,_be_businessunitid_value,be_incidenttype,be_meetingdate,statuscode&$
filter=be_idnumber ne null and be_idnumber eq " +
window.parent.Xrm.Page.getAttribute("be_idnumber").getValue() + " and
be_autonumber  ne " +
window.parent.Xrm.Page.getAttribute("be_autonumber").getValue() + "";
        //Get all inciden of patient with the same Var_I
        $.ajax({
            type: "GET",
            contentType: "application/json; charset=utf-8",
            datatype: "json",
            url: GRID_DATA,
            beforeSend: function (XMLHttpRequest) {
                XMLHttpRequest.setRequestHeader("OData-MaxVersion", "4.0");
                XMLHttpRequest.setRequestHeader("OData-Version", "4.0");
                XMLHttpRequest.setRequestHeader("Accept",
"application/json");
                XMLHttpRequest.setRequestHeader("Prefer", "odata.include-
annotations=\"*\"");
            },
            async: true,

            success: function (data, textStatus, XmlHttpRequest) {
                for (var i = 0; i < data.value.length; i++) {
                    var item = data.value[i];
                    gridArrayData.push({
                        be_autonumber: item.be_autonumber,
                        be_meetingdate: (item.be_meetingdate == null ? "" :
item["be_meetingdate@OData.Community.Display.V1.FormattedValue"]),
[redacted]
item["_be_businessunitid_value@OData.Community.Display.V1.FormattedValue"],
                        statuscode:
item["statuscode@OData.Community.Display.V1.FormattedValue"],
                        recordId: item.be_benefits_exhaustion_incidentid,
                        type: "be_benefits_exhaustion_incident"
                    });
                }


                for (var i = 0; i <= gridArrayData.length; i++) {
                    try {
                        $("#jqGrid").jqGrid('addRowData', i + 1,
gridArrayData[i]);
                    } catch (error) {
                    }
```

```
                    }


                    //// hide the show message
                    $("#jqGrid")[0].grid.endReq();
                    //// refresh the grid
                    $("#jqGrid").trigger('reloadGrid');
                },
                error: function (XmlHttpRequest, textStatus, errorThrown) {
                    res = null
                }
            });

        }



        //get value from url
        function GetValueFormUrl(paramter) {
            var vars = {};
            if (window.location.search.length !== 0)
                window.location.search.replace(/[?&]+([^=&]+)=([^&]*)/gi,
function (m, key, value) {
                    key = decodeURIComponent(key);
                    if (typeof vars[key] === "undefined") { vars[key] =
decodeURIComponent(value); }
                    else { vars[key] = [].concat(vars[key],
decodeURIComponent(value)); }
                });

            return vars;
        }
    </script>


</body>
                                    </html>
```

Javascript

be_benefits_exhaustion_incident

```
(function (benefits_exhaustion_incident) {
    //constants
    this.Constants = (function () {
        this.CALLING_MODULE_VARN = "be_benefits_exhaustion_incident.js";
        this.GENERAL_ERROR_MESSAGE = "שגיאה הלה, המערכת למנהל פנה אנא .שגיאה מזהה: ";
        this.INDIVIDUAL_RIGHTS_CHECK = 134310002;
        this.GET_GENERAL_INFORMATION = 134310000;
        this.MORE_CHILDREN_BELOW18 = "הכולל הילדים ממספר גדול 18 לגיל מתחת הילדים מספר";
        this.ERROR_MESSAGE_IMMIGRATION_YEAR = "נוכחית משנה גדולה עליה שנת";
        this.RETIREMENT_AGE_MALE = "RetirementAgeMale";
        this.RETIREMENT_AGE_FEMALE = "RetirementAgeFemale";
        this.PDF_CREATION_SERVICE_URL = "PdfCreationServiceURL";
        this.FORM_TYPE_CREATE = 1;
        this.FORM_TYPE_UPDATE = 2;
        this.MALE_VALUE = 1;
        this.FEMALE_VALUE = 2;
        this.FAMILY_RELATIONSHIP_IS_OTHER = 98;
        this.KNOW_LEDGE_BASE_LINK = "KnowledgeBaseLink";
        this.NOTIFICATION_LEVEL = "WARNING";
        this.MISSING_BUSINESSUNIT_ID = "WARNING";
        this.MISSING_BUSINESSUNIT = "מטפל במרכז ערך להזין יש";
        this.DEACTIVATE = 5;
        this.NOTIFICATION_NOTACCEPTED_DOCUMENT = "לבך לתשומת, טרם התקבל מסמך וס''ר
כה ויפר!";
        return this;
    })();
    benefits_exhaustion_incident.HandleFormLoadEvent = function () {
        try {
            onChangeRepresentHimself();    //show the section contact
            onChangeIncidentType();        //open more details tab
            onChangeRepresentAtiveType();  //enable be_representativeothertype
if be_representativetype=other
            lockForm();                    //Lock all fields if
request_submitted = 'true'
            showPreviousIncident();        //Show previous incident
            disableEnablePatientFields();  //Disable patient fields if exists,
enable if doesnot
            setRetirementKnowledgeBaseLink();


Xrm.Page.getAttribute("be_transplantdate").addOnChange(CheckWhetherItHasBeenSix
Months);

            var formType = Xrm.Page.ui.getFormType();

            switch (formType) {
                case FORM_TYPE_CREATE:
                    setCurrentUserBusinessunit();
                    setMeetingDate();
                    //save for search contact
                    Xrm.Page.data.entity.save();
                    break;
                case FORM_TYPE_UPDATE:
```

```
                notificationTypeIncident();

                break;
            default:
        }
    }
    catch (error) {
        Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
    }
};
benefits_exhaustion_incident.HandleFormSaveEvent = function (context) {
    try {
        checkNumberOfChildrenBelow18(context);
        checkImmigrationYear(context);

        if (context.getEventArgs().getSaveMode() == Constants.DEACTIVATE) {
            setCloseIncidentDate();
        }
    }
    catch (error) {
        Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
    }
};
benefits_exhaustion_incident.HandleControlChangeEvent = function (context)
{
    try {
        var sourceAttributeVarN = context.getEventSource().getVarN();

        switch (sourceAttributeVarN) {
            case "be_representhimself":
                emptyingRepresentativeFields();
                onChangeRepresentHimself();
                break;
            case "be_incidenttype":
                onChangeIncidentType();
                break;
            case "be_date_of_birth":
                calculateAge();
                calculateAgeGroup();
                break;
            case "be_gender":
                calculateAgeGroup();
                break;
            case "be_representativetype":
                onChangeRepresentAtiveType();
                break;
            case "be_patientfullVarN":
                showPreviousIncident();
                disableEnablePatientFields();
                break;
            case "be_childrenbelow18":
                checkNumberOfChildrenBelow18();
                break;
            case "be_immigrationyear":
                checkImmigrationYear();
                break;
            case "be_immigrationyear":
                setValueInstitute();
                break;
            case "be_businessunitid":
```

```
                    setValueInstitute();
                    break;

            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //if be_representhimself = no show the section
    onChangeRepresentHimself = function () {
        try {
            var representHimself =
Xrm.Page.getAttribute("be_representhimself").getValue();
            if (representHimself == 0)

Xrm.Page.ui.tabs.get("summaryTab").sections.get("summaryTab_section_4").setVisi
ble(true);  //Shows a section

            else

Xrm.Page.ui.tabs.get("summaryTab").sections.get("summaryTab_section_4").setVisi
ble(false);
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //if be_incidenttype = INDIVIDUAL_RIGHTS_CHECK open tab
    onChangeIncidentType = function () {
        try {
            var incidentType =
Xrm.Page.getAttribute("be_incidenttype").getValue();

            if (incidentType == INDIVIDUAL_RIGHTS_CHECK) {

Xrm.Page.ui.tabs.get("MoreDetailsTab").setDisplayState("expanded");

Xrm.Page.ui.tabs.get("incidentReason").setDisplayState("expanded");

Xrm.Page.ui.tabs.get("medicalConditionTab").setDisplayState("expanded");
            }
            else {

Xrm.Page.ui.tabs.get("MoreDetailsTab").setDisplayState("collapsed");

Xrm.Page.ui.tabs.get("incidentReason").setDisplayState("collapsed");

Xrm.Page.ui.tabs.get("medicalConditionTab").setDisplayState("collapsed");
            }

            notificationTypeIncident()
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
```

```
    //Trigger: OnLoad
    //Lock all fields if request_submitted = 'true'
    lockForm = function () {
        try {
            var request =
Xrm.Page.getAttribute("be_request_submitted").getValue();
            var enableFields = new Array("summaryTab", "previousIncident_tab",
"tab_7", "tab_8");
            if (request != null && request) {
                Common.DisableFieldsOnFormByTabs(enableFields, true);
            }
            else if (request == false) {
                Common.DisableFieldsOnFormByTabs(enableFields, false);
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
                                        };

//if be_representativetype=other enable be_representativeothertype to update
    onChangeRepresentAtiveType = function () {
        try {
            var representActiveType =
Xrm.Page.getAttribute("be_representativetype").getValue();
            if (representActiveType != null) {
                var result = Common.Retrieve("edm_family_relationship",
representActiveType[0].id, new Array("edm_code"));
                if (result != null && result != undefined && result.edm_code !=
null) {
                    if (result.edm_code == FAMILY_RELATIONSHIP_IS_OTHER) {

Xrm.Page.ui.controls.get("be_representativeothertype").setVisible(true);
                    }
                    else {

Xrm.Page.ui.controls.get("be_representativeothertype").setVisible(false);
                    }
                }
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };

    ////////onClickButtonCompleteQuestionnaire = function () {
    ////////    try {
    ////////        var Iscompletequestionnaire =
Xrm.Page.getAttribute("be_completequestionnaire").getValue();
    ////////        if (Iscompletequestionnaire == true)
    ////////
Xrm.Page.getAttribute("be_completequestionnaire").setValue(false);
    ////////        else
    ////////
Xrm.Page.getAttribute("be_completequestionnaire").setValue(true);
    ////////
Xrm.Page.getAttribute("be_completequestionnaire").setSubmitMode("always");
    ////////        Xrm.Page.data.save();
    ////////    }
```

```
    ////////    catch (error) {
    ////////        Common.HandleExecutionError(error,
Constants.CALLING_MODULE_VARN, "ERROR",
Common.Constants.GENERAL_ERROR_MESSAGE_ID);
    ////////    }
    ////////};

    //Calculate age group by age
    calculateAgeGroup = function () {
        try {
            var dateOfBirth =
Xrm.Page.getAttribute("be_date_of_birth").getValue();
            var gender = Xrm.Page.getAttribute("be_gender").getValue();

            if (dateOfBirth != null && gender != null) {
                var currentDate = Date.now();
                var diffDays = getDifferenceDays(dateOfBirth, currentDate);

                //Check that the birth date is correct (If diffDays <= 0 => the
birth date is future)
                if (diffDays > 0) {
                    age = getAge(dateOfBirth);
                    var retirementAge = getRetirementAge(gender);

                    if (Common.IsNotNullOrUndfined(retirementAge)) {
                        //תינוק - 1-90 יום
                        if (diffDays >= 1 && diffDays <= 90)
                            Xrm.Page.getAttribute("be_agegroup").setValue(1);
                        else
                            //עד קטין 3
                            if (age.years < 3 || (age.years == 3 && age.months
< 1 && age.days < 1))

Xrm.Page.getAttribute("be_agegroup").setValue(2);
                                else
                                    if (age.years < 18 || ((age.years == 18 &&
age.months < 3) || (age.years == 18 && age.months == 3 && age.days < 1)))

Xrm.Page.getAttribute("be_agegroup").setValue(3);
                                    else
                                        if (age.years < retirementAge || (age.years
== retirementAge && age.months < 1 && age.days < 1))

Xrm.Page.getAttribute("be_agegroup").setValue(4);
                                        else
Xrm.Page.getAttribute("be_agegroup").setValue(5);
                    }
                }
                                        else {
                    Xrm.Page.getAttribute("be_agegroup").setValue(null);
                }
            }
            else
                Xrm.Page.getAttribute("be_agegroup").setValue(null);
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //calculateAge = function (birthday) {
    //    try {
```

```
            var ageDifMs = Date.now() - birthday.getTime();
            var ageDate = new Date(ageDifMs); // miliseconds from epoch
            return Math.abs(ageDate.getUTCFullYear() - 1970);
        }
    //    catch (error) {
    //        Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
    //    }
    //};
    Get retirement age by gender
    getRetirementAge = function (gender) {
        try {
            var retirementAgeGender;
            var retirementAge;

            switch (gender) {
                case MALE_VALUE:
                    retirementAgeGender = RETIREMENT_AGE_MALE;
                    break;
                case FEMALE_VALUE:
                    retirementAgeGender = RETIREMENT_AGE_FEMALE;
                    break;
                default:
            }

            var queryUrl =
[redacted]
retirementAgeGender + "'";

            Common.WebAPI.RetrieveMultiple(queryUrl, false,
                //In success: Return the appropriate retirement age for gender
                function (result) {
                    if (result.value.length > 0) {
                        if
(Common.IsNotNullOrUndfined(result.value[0].homL_value)) {
                            retirementAge = result.value[0].homL_value;
                        }
                    }
                },
                Common.WebAPI.WebApiQueryError);

            return retirementAge;
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };

    setRetirementKnowledgeBaseLink = function () {
        try {
            var KnowledgeBaseLink = KNOW_LEDGE_BASE_LINK;
            var queryUrl =
[redacted]
KnowledgeBaseLink + "'";

            Common.WebAPI.RetrieveMultiple(queryUrl, false,
                //In success: Return the appropriate retirement
                function (result) {
                    if (result.value.length > 0) {
                        if
(Common.IsNotNullOrUndfined(result.value[0].homL_value)) {
```

```
                        KnowledgeBaseLink = result.value[0].homL_value;

Xrm.Page.getAttribute("be_knowledgebaselink").setValue(KnowledgeBaseLink);
                    }
                }
            },
            Common.WebAPI.WebApiQueryError);
        //return KnowledgeBaseLink;
    }
    catch (error) {
        Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
    }
};

CheckWhetherItHasBeenSixMonths = function () {
    try {
        var isHalfYearSinceTransplant = false;
        var transplantdate =
Xrm.Page.getAttribute("be_transplantdate").getValue();
        if (!Common.IsNotNullOrUndfined(transplantdate)) {

Xrm.Page.getAttribute("be_halfyearsincetransplant").setValue(false);
        }
        else {
            var yReceptionDate = transplantdate.getFullYear();
            var mReceptionDate = (transplantdate.getMonth() + 1);
            var dReceptionDate = transplantdate.getDate();

            var sixMonthAgo = new Date();
            sixMonthAgo = new
Date(sixMonthAgo.setMonth(sixMonthAgo.getMonth() - 6));
            var ySixMonthAgo = sixMonthAgo.getFullYear();
            var mSixMonthAgo = (sixMonthAgo.getMonth() + 1);
            var dSixMonthAgo = sixMonthAgo.getDate();

            if (ySixMonthAgo > yReceptionDate)
                isHalfYearSinceTransplant = true;
            else if (mSixMonthAgo > mReceptionDate && (ySixMonthAgo >
yReceptionDate || ySixMonthAgo == yReceptionDate))
                isHalfYearSinceTransplant = true;
            else if (dSixMonthAgo > dReceptionDate && (mSixMonthAgo >
mReceptionDate || mSixMonthAgo == mReceptionDate) && (ySixMonthAgo >
yReceptionDate || ySixMonthAgo == yReceptionDate))
                isHalfYearSinceTransplant = true;
            if (isHalfYearSinceTransplant == true) {

Xrm.Page.getAttribute("be_halfyearsincetransplant").setValue(true);
            }

            else {

Xrm.Page.getAttribute("be_halfyearsincetransplant").setValue(false);
            }

        }
    }
    catch (error) {
        Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
    }
};
```

```
//Get difference of 2 dates
    getDifferenceDays = function (date1, date2) {
        try {
            var timeDiff = Math.abs(date2 - date1.getTime());
            timeDiff = date2 - date1.getTime();

            var diffDays = Math.ceil(timeDiff / (1000 * 3600 * 24));
            return diffDays;
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Get age (incloude of years, months and days) by birthdate
    getAge = function (birthDate) {
        try {
            var now = new Date();
            var today = new Date(now.getYear(), now.getMonth() + 1,
now.getDate());
            var dateString = birthDate.toString();
            var yearNow = now.getYear();
            var monthNow = now.getMonth() + 1;
            var dateNow = now.getDate();
            var yearDob = birthDate.getYear();
            var monthDob = birthDate.getMonth() + 1;
            var dateDob = birthDate.getDate();
            var age = {};
            var ageString = "";
            var yearString = "";
            var monthString = "";
            var dayString = "";

            yearAge = yearNow - yearDob;

            if (monthNow >= monthDob)
                var monthAge = monthNow - monthDob;
            else {
                yearAge--;
                var monthAge = 12 + monthNow - monthDob;
            }

            if (dateNow >= dateDob)
                var dateAge = dateNow - dateDob;
            else {
                monthAge--;
                var dateAge = 31 + dateNow - dateDob;

                if (monthAge < 0) {
                    monthAge = 11;
                    yearAge--;
                }
            }

            age = {
                years: yearAge,
                months: monthAge,
                days: dateAge
            };

            return age;
        }
```

```
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };

    "completedQuestionnaire" button click
    benefits_exhaustion_incident.completedQuestionnaireClick = function () {

        Calculate benefits for current incident
        benefitsCalculation.calculateBenefits();

    }
    //"generatePDF" button click
    benefits_exhaustion_incident.generatePDFClick = function () {
        try {
            var recordId = Common.GetCurrentRecordId();
            var fetchXml = "<fetch version='1.0' out_a-format='xml-platform'
mapping='logical' distinct='false'>"
                                + "<entity VarN='be_incident_benefit'>"
                                + "<attribute VarN='be_remarks' />"
                                + "<attribute VarN='be_benefitid' />"
                                  + "<order attribute='be_benefitid'
descending='false' />"
                                  + "<filter type='and'>"
                                    + "<condition
attribute='be_benefitsexhaustionincidentid' operator='eq' value='" + recordId +
"' />"
                                      + "<condition attribute='statecode'
operator='eq' value='0' />"
                                  + "</filter>"
                                  + "<link-entity VarN='be_benefit'
[redacted]
                                    + "<attribute VarN='be_code' />"
                                  + "</link-entity>"
                                + "</entity>"
                              + "</fetch>";
            Common.WebAPI.PredefinedQuery("be_incident_benefits", fetchXml,
false, successRetIncidentBenefits, Common.WebAPI.WebApiQueryError);
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }
    //Success function of retrieve incident benefits
    function successRetIncidentBenefits(results) {
        try {
            debugger;
            var center = Common.ExtractAttribute("be_businessunitid");
            if (Common.IsNotNullOrUndfined(center) &&
Common.IsNotNullOrUndfined(center) &&
Common.IsNotNullOrUndfined(center.getValue())
                && Common.IsNotNullOrUndfined(center.getValue()[0]) &&
Common.IsNotNullOrUndfined(center.getValue()[0].VarN)) {
                Common.IsNotNullOrUndfined(caseNumber)
                var caseNumber = Common.ExtractAttribute("be_autonumber");
                if (Common.IsNotNullOrUndfined(caseNumber))
                    caseNumber = caseNumber.getValue();
                if (Common.IsNotNullOrUndfined(caseNumber)) {
```

```
                    var generalComments =
Common.ExtractAttribute("be_generalcomments");
                    if (Common.IsNotNullOrUndfined(generalComments))
                        generalComments =
jsonSpecialChars(generalComments.getValue());
                    var benefitCode;
                    var jsonObj = '{"benefitsList": {'
                                + '"CaseNumber": "' + caseNumber + '",'
                                + '"Center": "' + center.getValue()[0].VarN +
'",'
                                + '"Benefits": ['
                    for (var i = 0; i < results.value.length; i++) {
                        if
(Common.IsNotNullOrUndfined(results.value[i]["_be_benefitid_value"]))
                            benefitCode =
getBrainiBenefitCode(results.value[i]["_be_benefitid_value"]);

                        jsonObj += '{'
                         + '"BenefitID": "'
                        if (Common.IsNotNullOrUndfined(benefitCode)) {
                            jsonObj += benefitCode;

                        }
                        jsonObj += '"'
                        + ',"Comments": "'
                        if
(Common.IsNotNullOrUndfined(results.value[i]["be_remarks"])) {
                            jsonObj +=
jsonSpecialChars(results.value[i]["be_remarks"]);
                        }
                        jsonObj += '"'
                        + "},"
                    }

                    if (results.value.length > 0)
                        jsonObj = jsonObj.slice(0, -1);

                    jsonObj += "  ],"
                    + '"GeneralComment": "'
                    if (Common.IsNotNullOrUndfined(generalComments))
                        jsonObj += generalComments
                    jsonObj += '"'
                    + "}"
                    + "}";

                    var serviceUri = getPdfCreationServiceURL();

                    var pdfData = escape(jsonObj);
                    getPdfData(pdfData, serviceUri);

                }
            }
            else {
                Common.SetFormNotification(MISSING_BUSINESSUNIT,
NOTIFICATION_LEVEL, MISSING_BUSINESSUNIT_ID);
            }

        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
```

```
    }

    function escape(val) {
        if (typeof (val) != "string") return val;
        else {
            return val
                  .replace(/[\/]/g, '\\/')
                  .replace(/[\b]/g, '\\b')
                  .replace(/[\f]/g, '\\f')
                  .replace(/[\n]/g, '\\n')
                  .replace(/[\r]/g, '\\r')
                  .replace(/[\t]/g, '\\t');
        }
    }

    function jsonSpecialChars(val) {
        if (typeof (val) != "string") return val;
        else {
            return val.replace(/\"/g, '\\\"')
                      .replace(/\\'/g, "\\'");

        }
    }
//Get pdf data
    getPdfData = function (pdfData, serviceUri) {
        try {
            $.ajax({
                type: "POST",
                contentType: "application/json; charset=utf-8",
                url: serviceUri,
                data: pdfData,
                xhrFields: {
                    withCredentials: true
                },
                success:
                    function (response) {

                        var res;
                        if (typeof (response) == "string") {
                            debugger;
                            res = JSON.parse(response);

                        } else {
                            res = response;
                        }
                        if (res.dataFromInput.PDFFile != null) {
                            var base64 =
arrayBufferToBase64(res.dataFromInput.PDFFile);
                            createDocument(base64);
                        }
                        else {
                            alert(res.dataFromInput.ResponseMessage);
                        }
                    },
                error:
                    function (err) {
                        try {
                            var parser, xmlDoc;
                            parser = new DOMParser();
                            xmlDoc = parser.parseFromString(err.responseText,
"text/xml");
                            var responseText = xmlDoc.body.innerText;
```

```
                            throw {
                                message: responseText,
                                description: "Status: " + err.status + ",
Status Text: " + err.statusText + ", Message: " + responseText
                            }
                        }
                        catch (error) {
                            Common.HandleExecutionError(error,
Constants.CALLING_MODULE_VARN, "ERROR",
Common.Constants.GENERAL_ERROR_MESSAGE_ID);
                        }
                    }
            });
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }

    //Get benefit code
    getBrainiBenefitCode = function (benefitId) {
        try {
            var benefitCode = null;
            var queryUrl =
"be_benefits?$select=be_brainicode&$filter=be_benefitid eq " +
Common.RemoveGuidBraces(benefitId);

            Common.WebAPI.RetrieveMultiple(queryUrl, false,
                function (result) {
                    if (result.value.length > 0) {
                        if
(Common.IsNotNullOrUndfined(result.value[0].be_brainicode)) {
                            benefitCode = result.value[0].be_brainicode;
                        }
                    }
                },
                Common.WebAPI.WebApiQueryError);

            return benefitCode;
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }
    //Convert array buffer to base 64
    arrayBufferToBase64 = function (buffer) {
        try {
            var binary = '';
            var bytes = new Uint8Array(buffer);
            var len = bytes.byteLength;
            for (var i = 0; i < len; i++) {
                binary += String.fromCharCode(bytes[i]);
            }
            return window.btoa(binary);
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }
```

```
    //Convert base 64 to array buffer
    base64ToArrayBuffer = function (base64) {
        try {
            var binaryString = window.atob(base64);
            var binaryLen = binaryString.length;
            var bytes = new Uint8Array(binaryLen);
            for (var i = 0; i < binaryLen; i++) {
                var ascii = binaryString.charCodeAt(i);
                bytes[i] = ascii;
            }
            return bytes;
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }
    //create pdf document
    createDocument = function (base64Doc) {
        try {
            var benefitsExhaustionIncidentId = Xrm.Page.data.entity.getId();
            pdfDocumentCreation.createPdfDocument(base64Doc,
benefitsExhaustionIncidentId);
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }

    //Onchange show all his Previous Incident
    showPreviousIncident = function () {
        // if (setVisiblePreviousIncidentSection()) {
        var serverUrl = Xrm.Page.context.getClientUrl();
        var iFrame = "IFRAME_Previous_Incident";
        var url = serverUrl + "/WebResources/be_showPreviousIncident";
        SetUrlIFrame(iFrame, null);
        SetUrlIFrame(iFrame, url);

        //}
    }
    //Set url IFRAME
    SetUrlIFrame = function (iFrame, url) {

        var iframe = Xrm.Page.ui.controls.get(iFrame).setSrc(url);
    };
  Show/Hide previousIncident_section
    setVisiblePreviousIncidentSection = function () {
        var isVisible = false;
        var patientfullVarN =
Xrm.Page.getAttribute("be_patientfullVarN").getValue();
        var patientIdNumber = Xrm.Page.getAttribute("be_idnumber").getValue();
    //    if (patientfullVarN != null && patientIdNumber != null) {
    //        Xrm.Page.ui.tabs.get("previousIncident_tab").setVisible(true);
    //        isVisible = true
    //    }
    //    else {
    //        Xrm.Page.ui.tabs.get("previousIncident_tab").setVisible(false);
    //        isVisible = false;
    //    }
    //    return isVisible;
    //}
```

```
    //Trigger: DuplicationIncident button
    //Open new incident form and copy feilds from current incident
    benefits_exhaustion_incident.openDuplicateIncident = function () {
        try {
            // debugger;
            var currentIncident = Common.GetCurrentRecordId();
            var newIncidentId;
            var paramsArray = new Array();
            var entityVarN = "be_benefits_exhaustion_incident";

paramsArray.push([Common.PARAMETER_TYPE.EntityReference.value.toString(),
"Target", currentIncident, entityVarN]);
            Common.ExeAction("be_copy_incident", paramsArray, function (param)
{
                if (Common.IsNotNullOrUndfined(param) != null &&
Common.IsNotNullOrUndfined(param.NewIncident)) {
                    newIncidentId = param.NewIncident.slice(0, 36);
                }
            }, false);
            if (Common.IsNotNullOrUndfined(newIncidentId)) {
                Common.OpenEntityForm(Xrm.Page.data.entity.getEntityVarN(),
newIncidentId, null);

Xrm.Page.getAttribute("be_patientfullVarN").addOnChange(showPreviousIncident);
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };


    //Trigger:be_childrenbelow18.onChange, onSave
    //Show message if number of childrens below 18 bigger than number of whole
childrens
    checkNumberOfChildrenBelow18 = function (context) {
        try {
            var numberOfChildren =
Xrm.Page.getAttribute("be_numberofchildren").getValue();
            var childrenBelow18 =
Xrm.Page.getAttribute("be_childrenbelow18").getValue();
            if (childrenBelow18 != null) {
                if (numberOfChildren == null || (numberOfChildren != null &&
numberOfChildren < childrenBelow18)) {
                    alert(Constants.MORE_CHILDREN_BELOW18);
                    if (context != null && context != undefined) {
                        context.getEventArgs().preventDefault();
                    }
                }
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Trigger:be_immigrationyear.onChange, onSave
    //Show error message if immigration year bigger than current year
    checkImmigrationYear = function (context) {
        try {
```

```
            var immigrationYear =
Xrm.Page.getAttribute("be_immigrationyear").getValue();
            var currentYear = new Date().getFullYear();
            if (Common.IsNotNullOrUndfined(immigrationYear)) {
                if (immigrationYear > currentYear) {
                    alert(Constants.ERROR_MESSAGE_IMMIGRATION_YEAR);
                    if (Common.IsNotNullOrUndfined(context)) {
                        context.getEventArgs().preventDefault();
                    }
                }
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Trigger: onLoad in status create
    //Set businessunit feild to businessunit of user create the record
    setCurrentUserBusinessunit = function () {
        try {
            var userId = Common.XRMContext.getUserId();
            var queryUrl = "select=_businessunitid_value";
            Common.WebAPI.Retrieve("systemusers",
Common.RemoveGuidBraces(userId), queryUrl, false,
                                                    function (result) {
                                                        if
(Common.IsNotNullOrUndfined(result._businessunitid_value)) {

Xrm.Page.getAttribute("be_businessunitid").setValue([{ id:
result["_businessunitid_value"], VarN:
result["_businessunitid_value@OData.Community.Display.V1.FormattedValue"],
entityType:
result["_businessunitid_value@Microsoft.Dynamics.CRM.lookuplogicalVarN"] }]);

Xrm.Page.getAttribute("be_businessunitid").setSubmitMode("always");
                                                        }
                                                    },

Common.WebAPI.WebApiQueryError);
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Disable patient fields if exists, enable if doesnot
    disableEnablePatientFields = function () {
        try {
            var mokedPatientId = Xrm.Page.getAttribute("be_mokedpatientid");

            if (Common.IsNotNullOrUndfined(mokedPatientId) &&
Common.IsNotNullOrUndfined(mokedPatientId.getValue())) {
                Xrm.Page.getControl("be_agegroup").setDisabled(true);
                Xrm.Page.getControl("be_gender").setDisabled(true);
                Xrm.Page.getControl("be_moked_city_VarN").setDisabled(true);

                // Xrm.Page.getControl("be_city_VarN").setDisabled(true);
            }
            else {
                Xrm.Page.getControl("be_agegroup").setDisabled(false);
                Xrm.Page.getControl("be_gender").setDisabled(false);
```

```
                Xrm.Page.getControl("be_moked_city_VarN").setDisabled(false);

                //Xrm.Page.getControl("be_city_VarN").setDisabled(false);
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };

    //Emptying representative contact fields
    emptyingRepresentativeFields = function () {
        try {
            Xrm.Page.getAttribute("be_representativeVarN").setValue(null);
            Xrm.Page.getAttribute("be_representativephone").setValue(null);
            Xrm.Page.getAttribute("be_representativeemail").setValue(null);
            Xrm.Page.getAttribute("be_representativeidnumber").setValue(null);
            Xrm.Page.getAttribute("be_representativetype").setValue(null);
            Xrm.Page.getAttribute("be_representativeothertype").setValue(null);
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };

    //get pdf creation service url
    getPdfCreationServiceURL = function () {
        try {
            var pdfCreationServiceURL = null;

            var queryUrl =
"homL_system_parameterses?$select=homL_value&$filter=homL_VarN eq '" +
PDF_CREATION_SERVICE_URL + "'";

            Common.WebAPI.RetrieveMultiple(queryUrl, false,
                function (result) {
                    if (result.value.length > 0) {
                        if
(Common.IsNotNullOrUndfined(result.value[0].homL_value)) {
                            pdfCreationServiceURL = result.value[0].homL_value;
                        }
                    }
                },
                Common.WebAPI.WebApiQueryError);

            return pdfCreationServiceURL;
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }

    //set current date
    setMeetingDate = function () {
        Common.setCurrentDate("be_meetingdate");
    }

  //set current date
    setCloseIncidentDate = function () {
```

```
        Common.setCurrentDate("be_date_of_close_incident");
        Xrm.Page.data.entity.save();
    }

    //set value  insitution by institute businness unit
    setValueInstitute = function () {

        var businessunit =
Xrm.Page.getAttribute("be_businessunitid").getValue();
        if (businessunit != null) {
            var queryUrl = "select=_be_institute_value"
            Common.WebAPI.Retrieve("businessunits",
Common.RemoveGuidBraces(businessunit[0].id), queryUrl, false, function (result)
{

                if (Common.IsNotNullOrUndfined(result._be_institute_value)) {
[redacted]
                }

            }, Common.WebAPI.WebApiQueryError);
        }

    }

    //show and hide notification accept  Disclaimer of Confidentiality and
power of attorney
    notificationTypeIncident = function () {
        var incidentID = Common.GetCurrentRecordId();

        var queryUrl =
"be_incident_benefits?$select=be_incident_benefitid&$filter=_be_benefitsexhaust
ionincidentid_value eq " + Common.RemoveGuidBraces(incidentID) + " and (
be_detailedbenefitcheckup eq true or  be_helpcontactoffices eq true or
be_helptofillforms eq true ) and  statecode eq 0";

        Common.WebAPI.RetrieveMultiple(queryUrl, false,
            function (result) {
                if (result.value.length > 0) {
                    if
(!Xrm.Page.getAttribute("be_accept_doc_and_poa").getValue()) {
[redacted]
                    }
                    else
                        Xrm.Page.ui.clearFormNotification("1");
                }
            },
            Common.WebAPI.WebApiQueryError);

    }

    //calculate decimal date
    calculateAge = function () {
```

```
        if (Xrm.Page.getAttribute("be_date_of_birth") != null &&
Xrm.Page.getAttribute("be_date_of_birth").getValue() != null) {
            var birthDate =
Xrm.Page.getAttribute("be_date_of_birth").getValue()
            var age = Common.CalculateAge(birthDate);
            Xrm.Page.getAttribute("be_decimal_age").setValue(age);
        }
        else
            Xrm.Page.getAttribute("be_decimal_age").setValue(null);

    }


})(window.benefits_exhaustion_incident = window.benefits_exhaustion_incident ||
                                        {});
```

be_email

```
(function (ns) {

    //constants
    this.Constants = (function () {
        this.ENTITY_LOGICAL_VARN = "be_email";
        this.CALLING_MODULE_VARN = "be_email.js";
        this.SERVER_URL = Xrm.Page.context.getClientUrl();
        this.GLOBAL_ODATA_PATH = SERVER_URL +
"/XRMServices/2011/OrganizationData.svc";
        this.HANDLING_STEP_SEND_EMAIL_TO_PATIENT =
"HandlingStepSendEmailToPatient";
        return this;
    })();

    /* public methods */

    //handle form load event
    ns.HandleFormLoadEvent = function (context) {
        try {
            debugger;
            var formType = Xrm.Page.ui.getFormType();
            switch (formType) {
                case FORM_TYPE_CREATE:
                    setFieldsAccordingToIncident();
                    sendEmailFromBusinessUnitQueue();
                    setHandlingStep();
                    break;
                case FORM_TYPE_UPDATE:
                    break;
                default:
            }

            checkStatusCode();

        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }

    //handle form save event
    ns.HandleFormSaveEvent = function (context) {
        try {
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }

    //handle control change event
    ns.HandleControlChangeEvent = function (context) {
        try {
            //extract source attribute VarN
            var sourceAttributeVarN = context.getEventSource().getVarN();

            switch (sourceAttributeVarN) {
                case "":
                    break;
```

```
                default:
                    break;
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }


    checkStatusCode = function () {
        var statusCode = Xrm.Page.getAttribute("statuscode").getValue();
        if(statusCode==4)
        {
            Xrm.Page.getAttribute("be_handlingstep").setRequiredLevel("none");
        }
    }
    /* */
    setFieldsAccordingToIncident = function () {
        if (Xrm.Page.getAttribute("be_email_address").getValue()==null &&
Xrm.Page.getAttribute("be_patientfullVarN").getValue()==null)
            {
           var incident =
Xrm.Page.getAttribute("regardingobjectid").getValue();
            if (Common.IsNotNullOrUndfined(incident) &&
Common.IsNotNullOrUndfined(incident[0]) &&
                incident[0].entityType == "be_benefits_exhaustion_incident") {
                var result = Common.Retrieve("be_benefits_exhaustion_incident",
incident[0].id, new Array("be_emailaddress", "be_patientfullVarN"));
                if (result != null && result != undefined) {
                    var result =
Common.Retrieve("be_benefits_exhaustion_incident", incident[0].id, new
Array("be_emailaddress", "be_patientfullVarN"));
                    if (Common.IsNotNullOrUndfined(result.be_emailaddress &&
result.be_emailaddress != " "
                        && result.be_emailaddress != "")) {

Xrm.Page.getAttribute("be_email_address").setValue(result.be_emailaddress);
                    }
                    if (Common.IsNotNullOrUndfined(result.be_patientfullVarN))
{

Xrm.Page.getAttribute("be_patientfullVarN").setValue(result.be_patientfullVarN)
;
                    }

                }
            }

        }

    }

    //set from filed with BusinessUnit queue
    function sendEmailFromBusinessUnitQueue() {
        var queueRecord;
```

```
                [redacted]
                if (queueRecord != null) {
                    if (queueRecord._queueid_value != null) {
                        Common.SetLookUpValue("from",
queueRecord._queueid_value,
queueRecord['_queueid_value@OData.Community.Display.V1.FormattedValue'],
"queue");
                    }
                }
            }
        }
    }

    //get businessunit
    getBusinessUnit = function (systemUserId) {
        try {
            var businessUnit = null;
            var queryUrl =
"systemusers?$select=_businessunitid_value&$filter=systemuserid eq " +
Common.RemoveGuidBraces(systemUserId) + "";
            Common.WebAPI.RetrieveMultiple(queryUrl, false,
                function (result) {
                    if (result.value.length > 0) {
                        if (Common.IsNotNullOrUndfined(result.value[0]) &&
Common.IsNotNullOrUndfined(result.value[0]._businessunitid_value)) {
                            businessUnit =
result.value[0]._businessunitid_value;
                        }
                    }
                },
                Common.WebAPI.WebApiQueryError);
            return businessUnit;
        } catch (e) {
            throw e;
        }
    }


    //get queue
    getQueue = function (businessUnitId) {
        try {
            var queueRecord = null;
            var queryUrl =
"teams?$select=_queueid_value&$filter=_businessunitid_value eq " +
Common.RemoveGuidBraces(businessUnitId) + "";
            Common.WebAPI.RetrieveMultiple(queryUrl, false,
                function (result) {
                    if (result.value.length > 0) {
                        if (Common.IsNotNullOrUndfined(result.value[0])) {
                            queueRecord = result.value[0];
                        }
                    }
                },
                Common.WebAPI.WebApiQueryError);
            return queueRecord;
        } catch (e) {
            throw e;
        }
    }

    getSystemParameters=function(key)
    {
```

```
        var value;

        var queryUrl =
"homL_system_parameterses?$select=homL_value&$filter=homL_VarN eq '" + key +
"'";

        Common.WebAPI.RetrieveMultiple(queryUrl, false,
            function (result) {
                if (result.value.length > 0) {
                    if (Common.IsNotNullOrUndfined(result.value[0].homL_value))
{
                        value = result.value[0].homL_value;
                    }
                }
            },
            Common.WebAPI.WebApiQueryError);
        return value;
    }

    setHandlingStep = function () {
        if (Xrm.Page.getAttribute("be_handlingstep").getValue() == null) {
            var HandlingStepId =
getSystemParameters(Constants.HANDLING_STEP_SEND_EMAIL_TO_PATIENT);
            if (Common.IsNotNullOrUndfined(HandlingStepId)) {

                var queryUrl = "select=be_handling_step_typeid,be_VarN";

                Common.WebAPI.Retrieve("be_handling_step_types",
Common.RemoveGuidBraces(HandlingStepId), queryUrl, false, function (result) {

                    if
(Common.IsNotNullOrUndfined(result.be_handling_step_typeid)) {
                        var VarN = result["be_VarN"];
                        var logicalVarN = "be_handling_step_type"
                        Common.SetLookUpValue("be_handlingstep",
result["be_handling_step_typeid"], VarN, logicalVarN);
                    }

                }, Common.WebAPI.WebApiQueryError);
            }

        }
    }

})(window.be_email = window.be_email || {});
```

benefitsCalculation

```
(function (benefitsCalculation) {
    //constants
    this.Constants = (function () {
        this.CALLING_MODULE_VARN = "benefitsCalculation.js";
        [redacted]
        this.PARENT_BENEFIT_STATUS = null;
        this.BENEFIT_STATUS = null;
        this.IS_CREATED_BENEFIT = false;
        this.OPERATOR_VALUES = {
            BIGGER: 1,
            SMALLER: 2,
            EEQUAL: 3,
            CHANGED: 4,
            IN_RANGE: 5,
        };
        this.AUTO_INSERT_SHAPE = 134310000;
        return this;
    })();
    //Calculate benefits for current incident
    benefitsCalculation.calculateBenefits = function () {
        try {

            var PR = window.location.PR;
            var hostVarN = window.location.hostVarN;
            var path = window.location.pathVarN.split("/");
            org = path[1];
            serverUrl = PR + "//" + hostVarN + "/" + org;

            var tempfunc = Alert.hide;
            //show loading
            Alert.show("...המתן אנא, זכויות מחשב", null, [], "LOADING", 420, 115,
serverUrl, true);

           setTimeout(function () {
                //Change be_completequestionnaire field - in order to
disassociate "incidentBenefitGrid" grid records (by activating "disassociate"
custom workflow)
                changeCompleteQuestionnaireBit();
                //Calculate conditions for current incident
                calculateConditions();
                // Retrieve benefits records, sorted by parent benefits and
then by childrenben
                var queryUrl =
"be_benefits?$select=be_code&$orderby=be_hasparentbenefit
desc,be_haschildrenbenefits";
                Common.WebAPI.RetrieveMultiple(queryUrl, false,
retrieveBenefitSuccess, Common.WebAPI.WebApiQueryError);

               //hide loading
                tempfunc();
                window.Alert.hide = tempfunc;

            },800);

        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
```

```
    //Calculate conditions for current incident
    calculateConditions = function () {
        try {
            // Retrieve conditions records
            var queryUrl =
"be_conditions?$select=be_defaultvalue,be_fieldVarN,be_VarN,be_operator,be_sche
maVarN,be_value";
            Common.WebAPI.RetrieveMultiple(queryUrl, false,
retrieveConditionsSuccess, Common.WebAPI.WebApiQueryError);
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Success function of retrieve conditions query
    retrieveConditionsSuccess = function (result) {
        try {
            if (result.value.length > 0) {
                for (var i = 0, len = result.value.length; i < len; i++) {
                    if
(Common.IsNotNullOrUndfined(Common.IsNotNullOrUndfined(result.value[i].be_condi
tionid) && result.value[i].be_schemaVarN) &&
Common.IsNotNullOrUndfined(result.value[i].be_operator) &&
Common.IsNotNullOrUndfined(result.value[i].be_value) &&
Common.IsNotNullOrUndfined(result.value[i].be_defaultvalue)) {

                        var existsCondition =
checkExistenceOfCondition(result.value[i].be_schemaVarN, result.value[i])

                        if (Common.IsNotNullOrUndfined(existsCondition) &&
Common.IsNotNullOrUndfined(existsCondition.value) &&
Common.IsNotNullOrUndfined(existsCondition.conditionId)) {
                            var incidentConditionId =
createIncidentCondition(existsCondition.conditionId, existsCondition.value);
                        }
                    }
                }
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
                                        };

//Check the existence of conditions in the incident
    checkExistenceOfCondition = function (fieldVarN, condition) {
        try {
            var conditionValue, intakeFieldValue, conditionId;
            var existsCondition = {}; // Properties: value, conditionId

            if (Common.IsNotNullOrUndfined(Xrm.Page.getAttribute(fieldVarN))) {
                var fieldType =
Xrm.Page.getAttribute(fieldVarN).getAttributeType();
                var intakeFieldValue = getIntakeFieldValue(fieldVarN,
fieldType); //get intake field value by field type

                existsCondition.conditionId = condition.be_conditionid;

                if (Common.IsNotNullOrUndfined(intakeFieldValue)) {
```

```
                    //Check whether the condition satisfies the values of the
intake field, the condition value, and the condition operator
                    switch (condition.be_operator) {
                        //Bigger than:
                        case OPERATOR_VALUES.BIGGER:
                            existsCondition.value = intakeFieldValue >
condition.be_value ? true : false;
                            break;
                            //Smaller than:
                        case OPERATOR_VALUES.SMALLER:
                            existsCondition.value = intakeFieldValue <
condition.be_value ? true : false;
                            break;
                            //Eequal to
                        case OPERATOR_VALUES.EEQUAL:
                            existsCondition.value = intakeFieldValue ==
condition.be_value ? true : false;
                            break;
                            //Changed from:
                        case OPERATOR_VALUES.CHANGED:
                            existsCondition.value = condition.be_value !=
intakeFieldValue ? true : false;
                            break;
                            //In range:
                        case OPERATOR_VALUES.IN_RANGE:
                            var rangeValues = condition.be_value.split("-");
                            var isNumLowValue = /^\d+$/.test(rangeValues[0]);
                            var isNumHighValue = /^\d+$/.test(rangeValues[1]);
                            if (rangeValues.length == 2 && isNumLowValue &&
isNumHighValue) {
                                existsCondition.value = intakeFieldValue >=
rangeValues[0] && intakeFieldValue <= rangeValues[1] ? true : false;
                            }
                            break;
                        default:
                    }
                }
                else
                    existsCondition.value = condition.be_defaultvalue;

                return existsCondition;
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    // Get condition properties: be_conditionid, be_value & be_operator
    getConditionProperties = function (condition, intakeFieldValue) {
        try {
            var conditionProperties = {};

            //If there is a single qualifying condition record (depending on
the changed field value);
            if (condition.value.length == 1 &&
Common.IsNotNullOrUndfined(condition.be_conditionid)
                                &&
Common.IsNotNullOrUndfined(condition.be_value)
                                &&
Common.IsNotNullOrUndfined(condition.be_operator)) {
                conditionProperties.operator = condition.be_operator;
```

```
                conditionProperties.value = condition.be_value;
                conditionProperties.id = condition.be_conditionid;
            }
            else
                //Else - If there are several appropriate condition records
(depending on the changed field value);
                if (condition.value.length > 1) {
                    var suitableCondition = condition.value.filter(function
(obj) {
                        return obj.be_value == intakeFieldValue;
                    });

                    if (Common.IsNotNullOrUndfined(suitableCondition) &&
suitableCondition.length > 0 &&
Common.IsNotNullOrUndfined(suitableCondition[0].be_conditionid)

&& Common.IsNotNullOrUndfined(suitableCondition[0].be_value)

&& Common.IsNotNullOrUndfined(suitableCondition[0].be_operator)) {
                        conditionProperties.operator =
suitableCondition[0].be_operator;
                        conditionProperties.value =
suitableCondition[0].be_value;
                        conditionProperties.id =
suitableCondition[0].be_conditionid;
                    }
                }

            return conditionProperties;
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //get intake field value by field type
    getIntakeFieldValue = function (fieldVarN, fieldType) {
        try {
            var intakeFieldValue = null;

            if
(Common.IsNotNullOrUndfined(Xrm.Page.getAttribute(fieldVarN).getValue())) {
                switch (fieldType) {
                    case "lookup":
                        intakeFieldValue =
Xrm.Page.getAttribute(fieldVarN).getValue()[0].VarN;
                        break;
                    case "optionset":
                        intakeFieldValue =
Xrm.Page.getAttribute(fieldVarN).getText();
                        break;
                    case "boolean":
                        intakeFieldValue =
Xrm.Page.getAttribute(fieldVarN).getText();
                        break;
                    default:
                        intakeFieldValue =
Xrm.Page.getAttribute(fieldVarN).getValue();
                }
            }

            return intakeFieldValue;
```

```
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Create incident condition
    createIncidentCondition = function (conditionId, incidentConditionIValue) {
        try {
            var incidentConditionId;
            var entity = {};
            var incidentConditionRecord = new
Microsoft.Xrm.Sdk.Entity("be_incident_condition");

            var benefitsExhaustionIncidentId = Xrm.Page.data.entity.getId();
            entity["be_conditionid@odata.bind"] = "/be_conditions(" +
conditionId + ")";
            entity["be_benefitsExhaustionIncidentid@odata.bind"] =
"/be_benefits_exhaustion_incidents(" +
benefitsExhaustionIncidentId.replace("{", "").replace("}", "") + ")";
            entity.be_value = incidentConditionIValue;

            Common.WebAPI.Create("be_incident_conditions", entity, false,
                function (result) {
                    if (Common.IsNotNullOrUndfined(result)) {
                        incidentConditionId = result;
                    }
                },
                Common.WebAPI.WebApiQueryError);

            return incidentConditionId;
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Success function of retrieve benefit query
    retrieveBenefitSuccess = function (result) {
        try {
            if (result.value.length > 0) {
                for (var i = 0, len = result.value.length; i < len; i++) {
                    if
(Common.IsNotNullOrUndfined(result.value[i].be_benefitid)) {
                        var benefitId =
Common.RemoveGuidBraces(result.value[i].be_benefitid);
                        IS_CREATED_BENEFIT = false
                        // Retrieve benefits check tests, according to the
current id
                        var queryUrl =
"be_benefit_tests?$select=be_BENcODE&$filter=_be_benefitid_value eq " +
benefitId;
                        Common.WebAPI.RetrieveMultiple(queryUrl, false,
function (result) { retrieveBenefitTestSuccess(result, benefitId) },
Common.WebAPI.WebApiQueryError);
                    }
                }
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
```

```
        }
    };//Success function of retrieve benefit test query
    retrieveBenefitTestSuccess = function (result, benefitId) {
        try {
            if (result.value.length > 0) {
                for (var i = 0, len = result.value.length; i < len &&
!IS_CREATED_BENEFIT; i++) {
                    if (Common.IsNotNullOrUndfined(result.value[i].be_BENcODE))
{
                        PARENT_BENEFIT_STATUS = null;
                        var testCode = result.value[i].be_BENcODE;
                        // Retrieve records from a benefit for tests table, by
test code
                        var queryUrl =
"be_benefit_for_tests?$select=be_testtype,_be_benefitid_value&$filter=be_BENcOD
E eq " + result.value[i].be_BENcODE;
                        Common.WebAPI.RetrieveMultiple(queryUrl, false,
function (result) { retrieveBenefitForTestsSuccess(result, testCode, benefitId)
}, Common.WebAPI.WebApiQueryError);
                    }
                }
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Success function of retrieve benefit for test query
    retrieveBenefitForTestsSuccess = function (result, testCode, benefitId) {
        try {
            //Tenefit for test found by test code?
            if (result.value.length > 0) {
                // Yes - reference to parent benefits

                //Have benefits for test not reviewed?
                for (var i = 0, len = result.value.length; i < len; i++) {
                    //כ:
                    if
(Common.IsNotNullOrUndfined(result.value[i]._be_benefitid_value) &&
Common.IsNotNullOrUndfined(result.value[i].be_testtype)) {
                        var benefitsExhaustionIncidentId =
Common.RemoveGuidBraces(Xrm.Page.data.entity.getId());
                        var testTypeBenefitForTest =
result.value[i].be_testtype; // Test type in benefit for test record

                        // Check whether the benefit is set to the patient
(there is a record for the same benefit in the table "incident benefits" )
                        //Is there a incident benefit according to a benefit id
and an incident id ?
                        var queryUrl =
"be_incident_benefits?$filter=_be_benefitid_value eq " +
Common.RemoveGuidBraces(result.value[i]._be_benefitid_value) + " and
_be_benefitsexhaustionincidentid_value eq " + benefitsExhaustionIncidentId;
                        Common.WebAPI.RetrieveMultiple(queryUrl, false,
function (result) { retrieveIncidentBenefitsSuccess(result,
testTypeBenefitForTest) }, Common.WebAPI.WebApiQueryError);
                    }
                }
                //Is the PARENT_BENEFIT_STATUS empty?
                if (!Common.IsNotNullOrUndfined(PARENT_BENEFIT_STATUS)) {
```

```
                    // All parent benefits exist in the same test =>
PARENT_BENEFIT_STATUS = true
                    PARENT_BENEFIT_STATUS = true;
                }
            }
            //Is PARENT_BENEFIT_STATUS different from "No"?
            if (PARENT_BENEFIT_STATUS != false) {
                //yes => BENEFIT_STATUS = null
                BENEFIT_STATUS = null;
                checkConditionsForCurrentBenifit(testCode, benefitId); // Check
conditions for current benefit
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Success function of retrieve incident benefits query
    retrieveIncidentBenefitsSuccess = function (result, testTypeBenefitForTest)
{
        try {
            // Is there a incident benefit according to a benefit id code and a
incident id?
            if (result.value.length > 0 &&
Common.IsNotNullOrUndfined(result.value[0].be_incident_benefitid)) {
                // Is test type in benefit for test record positive?
                if (testTypeBenefitForTest == false) {
                    //No - there is a benefit that negates the current benefit
that is being examined => PARENT_BENEFIT_STATUS = false;
                    PARENT_BENEFIT_STATUS = false;
                }
            }
            else {
                // Is test type in benefit for test record positive?
                if (testTypeBenefitForTest == true) {
                    //Yes - The required right is lacking =>
PARENT_BENEFIT_STATUS = false
                    PARENT_BENEFIT_STATUS = false;
                }
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Check conditions for current benefit
    checkConditionsForCurrentBenifit = function (testCode, benefitId) {
        try {
            // Retrieve records from a conditions for tests table by test code
            var queryUrl =
"be_condition_for_tests?$select=_be_conditionid_value,be_testtype,_be_benefitid
_value&$filter=be_BENcODE eq " + testCode;
            Common.WebAPI.RetrieveMultiple(queryUrl, false, function (result) {
retrieveConditionForTestSuccess(result, benefitId) },
Common.WebAPI.WebApiQueryError);

        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
```

```
        }
    }; //Success function of retrieve condition for test query
    retrieveConditionForTestSuccess = function (result, benefitId) {
        try {
            // Have conditions for test found by test code?
            if (result.value.length > 0) {
                //Yes:
                //Are there any condition for test that have not been tested?
                for (var i = 0, len = result.value.length; i < len; i++) {
                    //Yes:
                    if
(Common.IsNotNullOrUndfined(result.value[i]._be_conditionid_value) &&
Common.IsNotNullOrUndfined(result.value[i].be_testtype)) {
                        var testTypeConditionForTest =
result.value[i].be_testtype;
                        // Check if the condition is met for the patient (there
is a record for the same condition in incident conditions table)
                        var benefitsExhaustionIncidentId =
Common.RemoveGuidBraces(Xrm.Page.data.entity.getId());
                        var queryUrl = "be_incident_conditions?$filter=be_value
eq true and _be_conditionid_value eq " +
Common.RemoveGuidBraces(result.value[i]._be_conditionid_value) + " and
_be_benefitsexhaustionincidentid_value eq " + benefitsExhaustionIncidentId;
                        Common.WebAPI.RetrieveMultiple(queryUrl, false,
function (result) { retrieveIncidentConditionsSuccess(result,
testTypeConditionForTest) }, Common.WebAPI.WebApiQueryError);
                    }
                }
                // No, all conditions for the test have been treated
                //is BENEFIT_STATUS empty?
                if (!Common.IsNotNullOrUndfined(BENEFIT_STATUS)) {
                    //Yes - Checking conditions is correct =>  Add a new
incident benefit record with benefit id and benefits exhaustion incident id
                    if (Common.IsNotNullOrUndfined(benefitId)) {
                        createIncidentBenefitRecord(benefitId);
                    }
                }
            }
            else {
                //No - the benefit is granted unconditionally =>  Add a new
incident benefit record with benefit id and benefits exhaustion incident id
                if (Common.IsNotNullOrUndfined(benefitId)) {
                    createIncidentBenefitRecord(benefitId);
                }
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Success function of retrieve incident condition query
    retrieveIncidentConditionsSuccess = function (result,
testTypeConditionForTest) {
        try {
            //Is there incident condition by condition id and benefits
exhaustion incident id?
            if (result.value.length > 0) {
                if
(Common.IsNotNullOrUndfined(result.value[0].be_incident_conditionid)) {
                    // Is test type in Conditions for test record positive?
                    if (testTypeConditionForTest == false) {
```

```
                        //No - There is a condition that negates the current
benefit that is being examined => BENEFIT_STATUS = false
                        BENEFIT_STATUS = false;
                    }
                }
            }
                //No - There is no incident condition by condition id and
benefits exhaustion incident id:
            else {
                // Is test type in Conditions for test record positive?
                if (testTypeConditionForTest == true) {
                    //Yes -  required condition is missing => BENEFIT_STATUS =
false
                    BENEFIT_STATUS = false;
                }
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Add a new incident benefit record with benefit id and benefits exhaustion
incident id
    createIncidentBenefitRecord = function (benefitId) {
        try {
            var entity = {};
            var benefitsExhaustionIncidentId =
Common.RemoveGuidBraces(Xrm.Page.data.entity.getId());
            benefitId = Common.RemoveGuidBraces(benefitId);
            entity["be_benefitid@odata.bind"] = "/be_benefits(" + benefitId +
")";
            entity["be_benefitsExhaustionIncidentid@odata.bind"] =
"/be_benefits_exhaustion_incidents(" + benefitsExhaustionIncidentId + ")";
            entity["be_generationway"] = Constants.AUTO_INSERT_SHAPE;
            Common.WebAPI.Create("be_incident_benefits", entity, false,
function (result) {
                IS_CREATED_BENEFIT = true;
            }, Common.WebAPI.WebApiQueryError);
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Change be_completequestionnaire field - in order to disassociate
"incidentBenefitGrid" grid records (by activating "disassociate" custom
workflow)
    changeCompleteQuestionnaireBit = function () {
        try {
            Xrm.Page.getAttribute("be_completequestionnaire").setValue(true);

Xrm.Page.getAttribute("be_completequestionnaire").setSubmitMode("always");
            Xrm.Page.data.save();
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };

    })(window.benefitsCalculation = window.benefitsCalculation || {});
```

pdfDocumentCreation

```
(function (pdfDocumentCreation) {
    //constants
    this.Constants = (function () {
[redacted]
        this.Document_Type_Create_PDF = "DocumentTypeCreatePDF"
        return this;
    })();

    //Test document creation
    pdfDocumentCreation.createPdfDocument = function (body, regardingObjectId)
{
        try {

            var fileVarN = "pdf.למטופל זכויות סיכום";
            var entityType = "be_benefits_exhaustion_incident";
            var documentTypeId = getDocumentTypeId();
            if (Common.IsNotNullOrUndfined(documentTypeId)) {
                var documentId = CreatePdfDocumentRecord(fileVarN,
regardingObjectId, documentTypeId, entityType);
                if (Common.IsNotNullOrUndfined(documentId)) {
                    CreateAnnotation(documentId, body, fileVarN);
                    RelateDocument(entityType, regardingObjectId, documentId);
                    UpdateUploadReady(documentId);
                    activeBookableresourcecharacteristicCreatingWF();
                }
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }

    //get document type id
    getDocumentTypeId = function () {
        try {

            var documentTypeId = "";
            var queryUrl =
[redacted]
Constants.Document_Type_Create_PDF + "'";

            Common.WebAPI.RetrieveMultiple(queryUrl, false,
                function (result) {
                    if (result.value.length > 0) {
                        if
(Common.IsNotNullOrUndfined(result.value[0].homL_value)) {
                            documentTypeId = result.value[0].homL_value;
                        }
                    }
                },
                Common.WebAPI.WebApiQueryError);
```

```
            return documentTypeId;
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }

    //update upload ready
    UpdateUploadReady = function (documentId) {
        try {
            Document = {};
            Document.homL_uploadready = true;
            UpdateRecord(Document, documentId, "homL_dm_documents");
            //Xrm.Page.data.refresh();

            var url = Xrm.Page.context.getClientUrl() +
"/main.aspx?etn=homL_dm_document&id=" + documentId + "&pagetype=entityrecord";
            window.showModalDialog(url, null,
[redacted]
resizable:1;status:no;scroll:yes");

        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }
    //relate doucment record to business record
    RelateDocument = function (businessEntityVarN, businessEntityId,
documentId) {
        try {
            var relationConfigRecords = {};

            //retrieve defined relation config records
            relationConfigRecords =
RetrieveRelationConfigRecords(businessEntityVarN);
            relationConfigRecords = relationConfigRecords.d.results;

            //test for valid results
            if (relationConfigRecords.length > 0) {

                //iterate through relation config records
                for (i = 0; i < relationConfigRecords.length; i++) {
                    {
                        RelateDocumentManyToOne(relationConfigRecords[i],
businessEntityVarN, businessEntityId, documentId);
                    }
                }
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };

    //retrieve Relation Config Records
    RetrieveRelationConfigRecords = function (businessEntityVarN) {
        try {
            var res = false;
```

```
            var GRID_DATA = odataPath +
"/homL_document_relation_configSet?$select=homL_businessEntityVarN,homL_lookupV
arN&$filter=homL_businessEntityVarN eq '" + businessEntityVarN + "'"

            $.ajax({
                type: "GET",
                contentType: "application/json; charset=utf-8",
                datatype: "json",
                async: isAsync,
                url: GRID_DATA,

                beforeSend: function (XMLHttpRequest) {
                    XMLHttpRequest.setRequestHeader("Accept",
"application/json");
                },
                success: function (data, textStatus, XmlHttpRequest) {
                    res = data;

                },
                error: function (XmlHttpRequest, textStatus, errorThrown) {
                    res = null
                }
            });

            return res;
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };

    //relate document to business entity in many:1 relation (lookup)
    RelateDocumentManyToOne = function (relationConfigRecord,
businessEntityVarN, businessEntityId, documentId) {
        try {
            var relationAttributeVarN = relationConfigRecord.homL_lookupVarN;
            var documentRecord = {};
            var logicalVarN = "homL_dm_documents";

            businessEntityId = businessEntityId.replace("{", "").replace("}",
"");
            documentRecord[relationAttributeVarN] = { LogicalVarN:
businessEntityVarN, Id: businessEntityId };
            UpdateRecord(documentRecord, documentId, logicalVarN);
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
//get values from url
    GetValueFormUrl = function (paramter) {
        try {
            var vars = {};
            var result;
            if (window.location.search.length !== 0)
                window.location.search.replace(/[?&]+([^=&]+)=([^&]*)/gi,
function (m, key, value) {
                    key = decodeURIComponent(key);
                    if (typeof vars[key] === "undefined") { vars[key] =
decodeURIComponent(value); }
```

```
                    else { vars[key] = [].concat(vars[key],
decodeURIComponent(value)); }
                });

            result = vars[paramter];
            return result;
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }

    //create document record
    CreatePdfDocumentRecord = function (fileVarN, regardingObjectId,
documentType, entityType) {
        try {
            var homL_DM_DocumentId
            var values = fileVarN.split(".");
            var extention = values[values.length - 2];
            var VarN = values[values.length - 1];
            var entity = {};

            entity.homL_VarN = VarN;
            entity.homL_extention = extention;
            entity["homL_documenttype@odata.bind"] = "/homL_dm_documenttypes("
+ Common.RemoveGuidBraces(documentType) + ")";
            entity.homL_parententityguid = regardingObjectId;
            entity.homL_parententityVarN = entityType;
            entity.homL_productiondate = new Date();
            entity.homL_creationmethod = DOCUMENT_CREATION_METHOD;
            Common.WebAPI.Create("homL_dm_documents", entity, false,
                function (result) { Common.WebAPI.Update
                    if (Common.IsNotNullOrUndfined(result)) {
                        homL_DM_DocumentId = result;
                    }
                },
                Common.WebAPI.WebApiQueryError);

            return homL_DM_DocumentId;
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    }

    //create crm record
    CreateRecord = function (entityRecord, oDataURI) {
        try {
            var jsonEntity = JSON.stringify(entityRecord);
            var result;

            $.ajax({
                type: "POST",
                contentType: "application/json; charset=utf-8",
                datatype: "json",
                url: oDataURI,
                data: jsonEntity,
                beforeSend: function (XMLHttpRequest) {
                    XMLHttpRequest.setRequestHeader("Accept",
"application/json");
```

```
            },
            success: function (response) {
                if (response != null && response.d != null) {
                    result = response.d;
                }
            },
            error: function (xmlHttpRequest, textStatus, errorThrown) {
                alert("Status: " + textStatus + "; ErrorThrown: " +
errorThrown);
            }
        });

        return result;
    }
    catch (error) {
        Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
    }
}

//create annotation
CreateAnnotation = function (documentId, body, fileVarN) {
    try {
        var entity = {};
        [redacted]
        entity.documentbody = body;
        entity.fileVarN = fileVarN;

        Common.WebAPI.Create("annotations", entity, false,
            function (result) {
            },
            Common.WebAPI.WebApiQueryError);
    }
    catch (error) {
        Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
    }
};

//update crm record
UpdateRecord = function (entity, entityId, logicalVarN) {
    try {
        Common.WebAPI.Update(logicalVarN, entity, entityId, false,
            function (result) {
            },
            Common.WebAPI.WebApiQueryError);
    }
    catch (error) {
        Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
    }
};

//Set be_isCreatedPdf field to true in order to create
Bookableresourcecharacteristic record (by activating appropriate WF)
activeBookableresourcecharacteristicCreatingWF = function () {
    try {
        Xrm.Page.getAttribute("be_iscreatedpdf").setValue(1);
        Xrm.Page.getAttribute("be_iscreatedpdf").setSubmitMode("always");
        Xrm.Page.data.save();
```

```
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    })(window.pdfDocumentCreation = window.pdfDocumentCreation || {});
```

be_incident_benefit

```
(function (incident_benefit) {
    //constants
    this.Constants = (function () {
        this.CALLING_MODULE_VARN = "be_incident_benefit.js";
        this.GENERAL_ERROR_MESSAGE = "שגיאה זהלה, המערכת למנהל פנה אנא .שגיאה מזהה: "
        this.STATUS_REALIZED=134310000;
        return this;
    })();

    incident_benefit.HandleFormLoadEvent = function () {
        try {
            debugger;
            showIframeUrl();
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }

    };
    incident_benefit.HandleFormSaveEvent = function (context) {
        try {
        }

        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };

    incident_benefit.HandleControlChangeEvent = function (context) {
        try {

            var sourceAttributeVarN = context.getEventSource().getVarN();
            switch (sourceAttributeVarN) {

                case "statuscode":
                    onChangeStatusCode();
                   break;
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    onChangeStatusCode = function () {
        var statusCode = Xrm.Page.getAttribute("statuscode").getValue();
        if (statusCode == STATUS_REALIZED)
        {
            Xrm.Page.getAttribute("be_benefitdate").setValue(Date.now());
            Xrm.Page.getAttribute("be_benefitdate").setSubmitMode("always");

        }
    }
    showIframeUrl = function () {
        try {
            if (Xrm.Page.getAttribute("be_benefitid").getValue() != null) {
                var detailsBenefit =
Xrm.Page.getAttribute("be_benefitid").getValue()[0].id;
```

```
                if (detailsBenefit != null) {
                    var iframe =
Xrm.Page.ui.controls.get("IFRAME_benefitInfo");
                    var benefitUrl = getBenefitUrl(detailsBenefit);
                    if (Common.IsNotNullOrUndfined(benefitUrl)) {
                        iframe.setSrc(benefitUrl);
                    }
                }
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    getBenefitUrl = function (detailsBenefit) {
        var fetchXml = "<?xml version='1.0'?>"
                        + "<fetch distinct='false' mapping='logical' out_a-
format='xml-platform' version='1.0'>"
                        + "<entity VarN='be_benefit'>"
                        + "<attribute VarN='be_benefitid'/>"
                        + "<attribute VarN='be_benefiturl'/>"
                        + "<order descending='false'
attribute='be_benefiturl'/>"
                        + "<filter type='and'>"
                        + "<condition attribute='be_benefitid' value='" +
detailsBenefit + "' uitype='be_benefit' operator='eq'/>"
                        + "</filter>"
                        + "</entity>"
                        + "</fetch>";
        var result = Common.RetrieveMultiple(fetchXml);
        if (Common.IsNotNullOrUndfined(result[0]) &&
Common.IsNotNullOrUndfined(result[0].be_benefiturl)) {
            return result[0].be_benefiturl;
        }
    };
          })(window.incident_benefit = window.incident_benefit || {});
```

be_condition_for_test

```
// JavaScript source code
(function (condition_for_test) {
    //constants
    this.Constants = (function () {
        this.CALLING_MODULE_VARN = "be_condition_for_test.js";
        return this;
    })();

    condition_for_test.HandleFormLoadEvent = function () {
        try {

        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    condition_for_test.HandleFormSaveEvent = function (context) {
        try {
            setBenefitFields();   //Set benefitTestId & benefitId fields by
benefit test code
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    condition_for_test.HandleControlChangeEvent = function (context) {
        try {
            var sourceAttributeVarN = context.getEventSource().getVarN();
            switch (sourceAttributeVarN) {

                case "be_BENcODE":
                    break;
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Trriger:form save
    //Set benefitTestId and benefitId feilds according to be_BENcODE value
    setBenefitFields = function () {
        try {
            var BENcODE = Xrm.Page.getAttribute("be_BENcODE").getValue();
            var queryUrl =
"be_benefit_tests?$select=be_benefit_testid,_be_benefitid_value,be_VarN&$filter
=be_BENcODE eq " + BENcODE;
            Common.WebAPI.RetrieveMultiple(queryUrl, false,
                //In success: Set benefitTestId and benefitId feilds
                function (result) {
                    if (result.value.length > 0) {
                        if
(Common.IsNotNullOrUndfined(result.value[0].be_benefit_testid))

Xrm.Page.getAttribute("be_benefittestid").setValue([{ id:
result.value[0].be_benefit_testid, VarN: result.value[0].be_VarN, entityType:
"be_benefit_test" }]);
                        else
```

```
Xrm.Page.getAttribute("be_benefittestid").setValue(null);

                        if
(Common.IsNotNullOrUndfined(result.value[0]._be_benefitid_value))
                            Xrm.Page.getAttribute("be_benefitid").setValue([{
id: result.value[0]._be_benefitid_value, VarN:
[redacted]
result.value[0]["_be_benefitid_value@Microsoft.Dynamics.CRM.lookuplogicalVarN"]
}]);
                        else

Xrm.Page.getAttribute("be_benefitid").setValue(null);
                    }
                    else
                    {

Xrm.Page.getAttribute("be_benefittestid").setValue(null);
                        Xrm.Page.getAttribute("be_benefitid").setValue(null);
                    }

                },
                Common.WebAPI.WebApiQueryError);
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
})(window.condition_for_test = window.condition_for_test || {});
```

be_benefit_for_test

```
// JavaScript source code
(function (benefit_for_test) {
    //constants
    this.Constants = (function () {
        this.CALLING_MODULE_VARN = "be_benefit_for_test.js";
        return this;
    })();

    benefit_for_test.HandleFormLoadEvent = function () {
        try {

        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    benefit_for_test.HandleFormSaveEvent = function (context) {
        try {
            setBenefitFields();   //Set benefitTestId & benefitId fields by
benefit test code
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    benefit_for_test.HandleControlChangeEvent = function (context) {
        try {
            var sourceAttributeVarN = context.getEventSource().getVarN();

            switch (sourceAttributeVarN) {
                default:
                    break;
            }
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
    //Trriger:form save
    //Set benefitTestId and benefitId feilds according to be_BENcODE value
    setBenefitFields = function () {
        try {
            var BENcODE = Xrm.Page.getAttribute("be_BENcODE").getValue();
            var queryUrl =
"be_benefit_tests?$select=be_benefit_testid,_be_benefitid_value,be_VarN&$filter
=be_BENcODE eq " + BENcODE;
            Common.WebAPI.RetrieveMultiple(queryUrl, false,
                //In success: Set benefitTestId and benefitId feilds
                function (result) {
                    if (result.value.length > 0) {
                        if
(Common.IsNotNullOrUndfined(result.value[0].be_benefit_testid))

Xrm.Page.getAttribute("be_benefittestid").setValue([{ id:
result.value[0].be_benefit_testid, VarN: result.value[0].be_VarN, entityType:
"be_benefit_test" }]);
                        else
```

```
Xrm.Page.getAttribute("be_benefittestid").setValue(null);

                        if
(Common.IsNotNullOrUndfined(result.value[0]._be_benefitid_value))
                            Xrm.Page.getAttribute("be_benefitid").setValue([{
id: result.value[0]._be_benefitid_value, VarN:
result.value[0]["_be_benefitid_value@OData.Community.Display.V1.FormattedValue"
], entityType:
result.value[0]["_be_benefitid_value@Microsoft.Dynamics.CRM.lookuplogicalVarN"]
}]);
                        else

Xrm.Page.getAttribute("be_benefitid").setValue(null);
                    }
                    else {

Xrm.Page.getAttribute("be_benefittestid").setValue(null);
                        Xrm.Page.getAttribute("be_benefitid").setValue(null);
                    }

                },
                Common.WebAPI.WebApiQueryError);
        }
        catch (error) {
            Common.HandleExecutionError(error, Constants.CALLING_MODULE_VARN,
"ERROR", Common.Constants.GENERAL_ERROR_MESSAGE_ID);
        }
    };
})(window.benefit_for_test = window.benefit_for_test || {});
```

PLUGIN

BenefitsExhaustionIncidentOnAssign

```
using Microsoft.Xrm.Sdk;
using Microsoft.Xrm.Sdk.Query;
using System;
using System.Linq;

VarNspace BenefitsExhaustionIncidentOnAssign
{
    //Assignment to a user who has the role of an information person or content
specialist or a department administrator for the extraction of rights
    public class assign : IPlugin
    {
        #region registration_Step
        //Message:Assign
        //Primary Entity:be_benefits_exhaustion_incident
        //Filtering Attributes:
        //Enevt:post
        //Execution:Synchronous
        #endregion

        #region const
        const string ASSIGN_MESSAGE = "Assign";
[redacted]
        const string TRANSFER_TO_SECOND_LINE_COUNSELING_GUID =
"TransferToSecondlineCounselingGuid";
        const string ASSIGN_INCIDENT = "AssignIncident";
        const string SYSTEM_CONTENT_MANAGER =
"SecurityRoleManagerSystemicContent";
[redacted]

        #endregion

        #region members
        IOrganizationService service;
        IPluginExecutionContext context;
        ITracingService tracingService = null;
        be_benefits_exhaustion_incident benefitsExhaustionIncident = null;
                                        #endregion

  public void Execute(IServiceProvider serviceProvider)
        {
            try
            {
                context =
(IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionCont
ext));
```

```
                IOrganizationServiceFactory serviceFactory =
(IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationSer
viceFactory));
                service =
serviceFactory.CreateOrganizationService(context.UserId);
                tracingService =
(ITracingService)serviceProvider.GetService(typeof(ITracingService));
                be_benefits_exhaustion_incident PreImage = null;
                XrmServiceContext xrmServiceContext = new
XrmServiceContext(service);
                EntityReference benefitsExhaustionIncidentRef =
(EntityReference)context.InputParameters["Target"];
                if (context.MessageVarN == ASSIGN_MESSAGE)
                {
                    if ((context.InputParameters != null) &&
(context.InputParameters.Count > 0) &&
(context.InputParameters.Contains("Target")) &&
(context.InputParameters["Target"] is EntityReference))
                    {
                        benefitsExhaustionIncidentRef =
(EntityReference)context.InputParameters["Target"];
                        benefitsExhaustionIncident =
service.Retrieve(benefitsExhaustionIncidentRef.LogicalVarN,
benefitsExhaustionIncidentRef.Id, new ColumnSet(new string[] { "ownerid"
})).ToEntity<be_benefits_exhaustion_incident>();
                        if (benefitsExhaustionIncident != null &&
benefitsExhaustionIncident.OwnerId != null)
                        {
                            if (context.PreEntityImages.Contains("PreImage") &&
context.PreEntityImages["PreImage"] is Entity)
                            {
                                PreImage =
((Entity)context.PreEntityImages["PreImage"]).ToEntity<be_benefits_exhaustion_i
ncident>();
                                if (PreImage.OwnerId != null)
                                {
                                    //trace main exit
                                    tracingService.Trace(string.Format("{0}
Start function CreateGenericHandlingStep: {1}", MODULE_VARN,
DateTime.Now.ToLongTimeString()));

RetriveRoleOfTheCurrentOwner(xrmServiceContext, context,
benefitsExhaustionIncident, PreImage, benefitsExhaustionIncident);
                                }
                            }
                        }
                    }
                }
            }
            catch (ArgumentNullException ex)
            {
                Manager mgr = new Manager(service);
                //log exception
                string errorToken = mgr.LogException(ex,
Manager.ErrorSourceType.Plugin, MODULE_VARN, FRIENDLY_ERROR_MSG, ex.Message,
context.InitiatingUserId);
            }
            catch (Exception ex)
            {
                Manager mgr = new Manager(service);
                //log exception
```

```
                string errorToken = mgr.LogException(ex,
Manager.ErrorSourceType.Plugin, MODULE_VARN, FRIENDLY_ERROR_MSG, ex.Message,
context.InitiatingUserId);
            }
            //trace main exit
            tracingService.Trace(string.Format("{0} Exit function Execute:
{1}", MODULE_VARN, DateTime.Now.ToLongTimeString()));
        }
        /// <summary>
        /// Create Generic Handling Step
        /// </summary>
        /// <param VarN="context"></param>
        /// <param VarN="CurrentIncident"></param>
        /// <param VarN="preImage"></param> public void
CreateGenericHandlingStep(XrmServiceContext xrmServiceContext,
be_benefits_exhaustion_incident benefitsExhaustionIncident,
be_benefits_exhaustion_incident CurrentIncident,
be_benefits_exhaustion_incident PreImage, bool hasSecurityRole)
        {
            if (CurrentIncident != null)
            {
                be_Generic_Handling_Step genericHandlingStep = new
be_Generic_Handling_Step();
                tracingService.Trace(string.Format("{0} Start function
RetriveSystemParameter: {1}", MODULE_VARN, DateTime.Now.ToLongTimeString()));
                homL_system_parameters systemParameter = new
homL_system_parameters();
                if (hasSecurityRole)
                    systemParameter = RetriveSystemParameter(xrmServiceContext,
TRANSFER_TO_SECOND_LINE_COUNSELING_GUID);
                else
                    systemParameter = RetriveSystemParameter(xrmServiceContext,
ASSIGN_INCIDENT);
                ActivityParty toParty = null;
                if (systemParameter != null && systemParameter.homL_value !=
null)
                {
                    Guid guid = new Guid(systemParameter.homL_value);

                    genericHandlingStep.be_HandlingStep = new
EntityReference(be_Handling_step_type.EntityLogicalVarN, guid);//Treatment
course = Transfer to second line counseling

                    if (PreImage.OwnerId != null &&
PreImage.OwnerId.LogicalVarN == SystemUser.EntityLogicalVarN)
                    {
                        toParty = new ActivityParty
                        {
                            PartyId = new
EntityReference(SystemUser.EntityLogicalVarN,
benefitsExhaustionIncident.OwnerId.Id)
                        };
                    }
                    ActivityParty fromParty = new ActivityParty
                    {
                        PartyId = new
EntityReference(SystemUser.EntityLogicalVarN, context.UserId)
                    };
                    genericHandlingStep.From = new ActivityParty[] { fromParty
};// from = owner
                    genericHandlingStep.To = new ActivityParty[] { toParty
};//To = the user to whom a referral is assigned
```

```
                    genericHandlingStep.ScheduledEnd =
DateTime.Now.ToLocalTime().AddDays(2);//Due date = Current date plus two days
                    genericHandlingStep.RegardingObjectId = new
EntityReference(be_benefits_exhaustion_incident.EntityLogicalVarN,
(Guid)CurrentIncident.be_benefits_exhaustion_incidentId);// About = current
referral
                }
                try
                {
                    service.Create(genericHandlingStep);
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }
        }
        /// <summary>
        /// Retrive System Parameters
        /// </summary>
                        /// <returns>systemParamete</returns>

  public homL_system_parameters RetriveSystemParameter(XrmServiceContext
xrmServiceContext, string parameterVarN)
        {
            try
            {
                homL_system_parameters systemParamete = ((from sp in
xrmServiceContext.homL_system_parametersSet
                                                          where sp.homL_VarN ==
parameterVarN
                                                          select new
homL_system_parameters
                                                          {
                                                              homL_value =
sp.homL_value
                                                          }).FirstOrDefault());

                return systemParamete;
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// Retrive Role Of The Current Owner
        /// </summary>
        /// <param VarN="xrmServiceContext"></param>
        /// <param VarN="context"></param>
        /// <param VarN="CurrentIncident"></param>
        /// <param VarN="PostImage"></param>
        /// <returns></returns>
        public SystemUserRoles RetriveRoleOfTheCurrentOwner(XrmServiceContext
xrmServiceContext, IPluginExecutionContext context,
be_benefits_exhaustion_incident CurrentIncident,
be_benefits_exhaustion_incident PreImage, be_benefits_exhaustion_incident
benefitsExhaustionIncident)
        {
            //SystemUserRoles SystemUserRoles = new SystemUserRoles();
```

```
            string fetchXml = "<fetch version='1.0' out_a-format='xml-platform'
mapping='logical' distinct='true'> "
[redacted]
/> "
                              + "    <link-entity VarN='systemuserroles'
from='roleid' to='roleid' visible='false' intersect='true'> "
                              + "      <link-entity VarN='systemuser'
from='systemuserid' to='systemuserid' alias='ac'> "
                              + "        <filter type='and'> "
                              + "          <condition attribute='systemuserid'
operator='eq' value='" + benefitsExhaustionIncident.OwnerId.Id + "' /> "
                              + "        </filter> "
                              + "      </link-entity> "
                              + "    </link-entity> "
                              + "  </entity> "
                              + "</fetch>";
            EntityCollection collection = service.RetrieveMultiple(new
FetchExpression(fetchXml));
            if (collection.Entities.Count > 0 && collection.Entities != null)
            {
[redacted]
RetriveSystemParameter(xrmServiceContext,
DIRECTOR_OF_RIGHTS_EXTRACTION_DEPARTMENT_GUID);
                if (KnowledgeManager != null || SystemContentManager != null ||
DirectorOfRightsExtractionDepartment != null)
                {
                    // user has security role
                    if
(collection.Entities[0].GetAttributeValue<string>("VarN") ==
KnowledgeManager.homL_value.ToString() ||
collection.Entities[0].GetAttributeValue<string>("VarN") ==
SystemContentManager.homL_value.ToString() ||
collection.Entities[0].GetAttributeValue<string>("VarN") ==
DirectorOfRightsExtractionDepartment.homL_value.ToString())
                    {

                        tracingService.Trace(string.Format("{0} Start function
CreateGenericHandlingStep: {1}", MODULE_VARN,
DateTime.Now.ToLongTimeString()));
                        CreateGenericHandlingStep(xrmServiceContext,
benefitsExhaustionIncident, CurrentIncident, PreImage, true);

                    }
                    // user hasn"t security role
                    else
                    {
                        CreateGenericHandlingStep(xrmServiceContext,
benefitsExhaustionIncident, CurrentIncident, PreImage, false);
                    }
                }
                else
                {
                    throw new ArgumentNullException(MISSING_SYSTEM_PARAMETER);
```

```
                }
            }
            else
            {
                throw new ArgumentNullException(MISSING_SYSTEM_PARAMETER);
            }

            return null;

        }
    }

}
```