```
var lineBreaker = "\n";
var rules = [validateTable, validateField];

var beforeSendValidate = function(CURRENT_STATE, NEXT_STATE)
{
    var errorMsg = "";
    if (CURRENT_STATE == NEXT_STATE) {
        return;
    }

    var fields = requiredFields.getFields();
    for (var i = 0; i < fields.length; i++) {
        if (fields[i].activities.indexOf(CURRENT_STATE) >= 0)
{
            var selector = (fields[i].name.indexOf("___") >
0) ?
                '[name^="' + fields[i].name + '"]' :
                '[name="' + fields[i].name + '"]';
            $(selector).each(function() {
                for (rule in rules) {
                    var validation = rules[rule](this,
selector);
                    if (validation.status != "success") {
                        if (validation.status === "error") {
                            errorMsg += validation.message;
                        }
                        break;
                    }
                }
            });
        }
    }

    // VALIDAÇÃO CUSTOMIZADA DOS CAMPOS
    errorMsg += validateCustomField(CURRENT_STATE);

    if (errorMsg != "") {
        throw errorMsg;
    }
}

function validateCustomField(CURRENT_STATE) {
    var errorMsg = "";
    var customFields = new Fields();

    // Definição manual de estados caso não estejam globais
    var INICIO_0 = 0;
```

```
    var INICIO = 4;

    if (CURRENT_STATE == INICIO_0 || CURRENT_STATE == INICIO)
{

        if ($("input[name='rdTipoInclusao']:checked").val()
== undefined) {
            errorMsg += "Campo Tipo de inclusão é
obrigatório!" + lineBreaker;
        }

        // 1. Validar Data de Emissão Manual
        var dataEmissaoStr = $("#txt_dataEmissao").val();
        // Correção: Verifica se é undefined ou vazio
        if (!dataEmissaoStr || dataEmissaoStr == "") {
            errorMsg += "A Data de Emissão Manual é
obrigatória!" + lineBreaker;
        }

        if ($("input[name='rdTipoInclusao']:checked").val()
== "proposta") {

            if ($("input[name='rdMenorIdade']:checked").val()
== undefined) {
                errorMsg += "Campo Menor de Idade é
obrigatório! " + lineBreaker;
            }

            var indices =
retornaIndices("dadositensNewProposta");

            if (indices.length == 0) {
                errorMsg += "Informe ao menos uma proposta
para a solicitação! " + lineBreaker;
            } else {
                for (var i in indices) {
                    if ($("input[name='txt_proReg___" +
indices[i] + "']:checked").val() == undefined) {
                        errorMsg += "Campo Regime de
Tributação é obrigatório! " + lineBreaker;
                    }

                    if ($("input[name='statusProposta___" +
indices[i] + "']:checked").val() == undefined) {
                        errorMsg += "Campo Status da Proposta
é obrigatório! " + lineBreaker;
                    }
```

```
                    if ($("input[name='txt_mensAportNew___" +
indices[i] + "']:checked").val() == undefined) {
                        errorMsg += "Campo Mensalidade ou
Aporte é obrigatório! " + lineBreaker;
                    }

                    if (!validacaoJson($("#jsonFundos___" +
indices[i]).val())) {
                        errorMsg += "Informe ao menos um
fundo para a proposta na linha " + (parseInt(i) + 1) + "! " +
lineBreaker;
                    }

                    if
(!validacaoJson($("#beneficiarioJson___" +
indices[i]).val())) {
                        errorMsg += "Informe ao menos um
beneficiário para a proposta na linha " + (parseInt(i) + 1) +
"! " + lineBreaker;
                    }

                    var indicesComissao =
retornaIndices("dadositenssseguradora");
                    if (indicesComissao.length > 0) {
                        errorMsg += "Para inclusões do tipo
proposta, não é permitido incluir dados de comissão! " +
lineBreaker;
                    }
                }
            }
        }

        if ($("input[name='rdTipoInclusao']:checked").val()
== "saldo" ||
            $("input[name='rdTipoInclusao']:checked").val()
== "comissao" ||
            $("input[name='rdTipoInclusao']:checked").val()
== "resgate") {

            var indices =
retornaIndices("dadositenssseguradora");

            if (indices.length == 0) {
                errorMsg += "Informe ao menos uma linha de
saldo/comissão/resgate! " + lineBreaker;
            } else {
```

```
                // --- INÍCIO DA VALIDAÇÃO (CORRIGIDO:
REMOVIDA LÓGICA DE SPLIT QUE CAUSAVA ERRO) ---
                if (dataEmissaoStr && dataEmissaoStr != "") {

                    for (var i in indices) {
                        var indice = indices[i];
                        var dataVencimentoStr =
$("#txt_seguradoraVencimento___" + indice).val();

                        // Correção: Verifica se é undefined
ou vazio para não quebrar
                        if (!dataVencimentoStr ||
dataVencimentoStr == "") {
                            errorMsg += "O campo Vencimento
da linha " + (parseInt(i) + 1) + " é obrigatório!" +
lineBreaker;
                        }

                        // REMOVIDO: Blocos de split e
comparação de data.
                        // Se for apenas para validar
obrigatoriedade, o código acima basta.
                    }
                }
                // --- FIM DA VALIDAÇÃO DE DATAS ---

                var seguradoraMesComp = [];

                if
($("input[name='rdTipoInclusao']:checked").val() == "saldo"
||
$("input[name='rdTipoInclusao']:checked").val() == "resgate")
{

                    for (var i in indices) {
                        seguradoraMesComp.push(
                            $("#txt_cdSeguradora___" +
indices[i]).val() +
                            "-" +
                            $("#txt_seguradoraMesComp___" +
indices[i]).val() +
                            "-" +
                            // Correção para evitar undefined
na chave única
```

```
                        ($("#txt_propostaSeguradora___" +
indices[i]).val() || "")
                    )
                }

                if
(linhasDuplicadasSaldoComissao(seguradoraMesComp)) {
                    errorMsg += "Não é possível incluir
mais de uma linha com a mesma seguradora e mês de
competência" + lineBreaker;
                }
            }

            if
($("input[name='rdTipoInclusao']:checked").val() ==
"comissao") {
                for (var i in indices) {
                    seguradoraMesComp.push(
                        $("#txt_cdSeguradora___" +
indices[i]).val() +
                        "-" +
                        $("#txt_seguradoraMesComp___" +
indices[i]).val() +
                        "-" +
                        $("#txt_cdNatureza___" +
indices[i]).val())
                }

                if
(linhasDuplicadasSaldoComissao(seguradoraMesComp)) {
                    errorMsg += "Não é possível incluir
mais de uma linha com a mesma seguradora, mês de competência
e Tipo de Comissão" + lineBreaker;
                }
            }
        }
    }

    var fields = customFields.getFields();
    for (var i = 0; i < fields.length; i++) {
        if (fields[i].activities.indexOf(CURRENT_STATE) >= 0)
{
            var selector = (fields[i].name.indexOf("___") >
0) ? '[name^="' +
                fields[i].name + '"]' : '[name="' +
fields[i].name + '"]';
```

```
            $(selector).each(function() {
                for (rule in rules) {
                    var validation = rules[rule](this,
selector);
                    if (validation.status != "success") {
                        if (validation.status === "error") {
                            errorMsg += validation.message;
                        }
                        break;
                    }
                }
            });
        }
    }
    // console.log("return error message custom: " +
errorMsg);
    return errorMsg;
}

function validateTable(el) {
    var validation = new Validation();
    if ($(el).prop("tagName") === "TABLE") {
        validation.status = "ignore";
        if ($(el).find("tr").length <= 2) {
            validation.message = "Insira ao menos um registro
no pai x filho: " +
                getLabel($(el).attr("name")) + lineBreaker;
            validation.status = "error";
        }
    }
    return validation;
}

function validateField(el, selector) {
    var validation = new Validation();
    if ([ "radio", "checkbox" ].indexOf($(el).attr("type"))
>= 0 && $(selector +
            ':checked').length <= 0) ||
        (el.value == "" && el.tagName != "SPAN")) {
        validation.message = "Campo " +
getLabel($(el).attr("name")) +
            " é obrigatório!" + lineBreaker;
        validation.status = "error";
    }
    return validation;
}
```

```
function Validation() {
    this.status = "success";
    this.message = "";
}

function Fields() {
    this.fields = [];
    this.addField = function(name, arrayActivities) {
        this.fields.push({
            "name": name,
            "activities": arrayActivities
        });
    }
    this.getFields = function() {
        return this.fields;
    }
}

var requiredFields = new Fields();

function setRequired(name, isRequired) {
    name = name.split("___")[0];
    (isRequired) ? $("[for='" + name +
"']").addClass('required') : $("[for='" + name +
        "']").removeClass('required');
}

function getLabel(name) {
    name = name.split("___")[0];
    return $("[for='" + name + "']").html() || name;
}

function getClosestByElement(element, selector) {
    var max = 10;
    var returnElement = undefined;
    while (returnElement == null && max >= 0) {
        if ($(element).is(selector)) {
            returnElement = element;
        } else {
            element = element.parent();
        }
        max--;
    }
    return returnElement;
}
```