# Автозаповнення атрибутів «Форма власності» та «Валюта»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Опис задачі (Що необхідно зробити, Як буде тестуватись, Очікуваний результат)** | | | | |
| Необхідно реалізувати автозаповнення атрибутів «Форма власності» та «Валюта».  Заповнення атрибутів має відбуватись на зміну атрибуту «Створити контрагента». Вносити зміни в ці атрибути має бути заборонено для користувачів.  «Форма власності» має бути заповнено:  {  "code": "counterpartyOwnershipType",  "value": "Юридична особа",  "text": null  }  «Валюта» має бути заповнено:  {  "code": "Currency",  "value": "1",  "text": "UAH",  "itemCode": "980",  "itemDictionary": "Currency"  }  Врахувати, що «Валюта» це атрибут в таблиці «Розрахункові рахунки». | | | | |
| **Опис сценарію (що потрібно реалізувати за допомогою JS. Обов’язково використовуючи «ЯКЩО» -> «ТО» -> «ІНАКШЕ» і тригером (умови(a), або дія при яких має виконатись сценарій)** | | | | |
| 1. ЯКЩО Створити контрагента=true, ТО Форма власності= Юридична особа ТА Валюта= UAH, редагування заборонено, ІНАКШЕ атрибути незаповнені. | | | | |
| 1. При очищенні атрибуту Створити контрагента очищується значення в атрибутах Форма власності ТА Валюта. | | | | |
|  | | | | |
| **Атрибути картки, які задіяні в JS** | | | | |
| **Назва** | **Код** | **Тип в SP** | **Сист.атрибут** | **Код довідника** |
| Створити контрагента | CreateCounterparty | Так\Ні | - | - |
| Форма власності | counterpartyOwnershipType | Вибір | - | - |
| Валюта | Currency | Багаторядковий текст | Довідник | Currency |
| Розрахункові рахунки | TableOrganizationAccountNumber | Багаторядковий текст | Таблиця | - |

# Автозаповнення контрагента із зовнішнього довідника

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Опис задачі (Що необхідно зробити, Як буде тестуватись, Очікуваний результат)** | | | | |
| Необхідно реалізувати автозаповнення даних контрагента із зовнішнього довідника контрагентів в відповідні атрибути картки після успішного створення нового контрагента.  Автозаповнення має здійснюватись автоматично після отримання відповіді від зовнішньої системи з ідентифікатором нового контрагента.  Автозаповнення здійснюється із зовнішнього довідника контрагентів по ідентифікатору.  Приклад скрипта на отримання даних по контрагенту нижче.  Параметри:  counterpartyId – docId з відповіді від зовнішньої системи  counterpartyCode – ContractorData.data.Code  counterpartyName – ContractorData.data.ShortName | | | | |
| **Опис сценарію (що потрібно реалізувати за допомогою JS. Обов’язково використовуючи «ЯКЩО» -> «ТО» -> «ІНАКШЕ» і тригером (умови(a), або дія при яких має виконатись сценарій)** | | | | |
| 1. ЯКЩО нового контрагента створено, ТО counterpartyId має заповнитись значенням ідентифікатора у відповіді ТА атрибути organizationCode, counterpartyName мають автоматично заповнитись із довідника контрагентів відповідними значеннями. | | | | |
| 1. ЯКЩО контрагента не створено через помилку при створенні, ТО відобразити помилку. | | | | |
|  | | | | |
| **Атрибути картки, які задіяні в JS** | | | | |
| **Назва** | **Код** | **Тип в SP** | **Сист.атрибут** | **Код довідника** |
| Ідентифікатор | counterpartyId | Однорядковий текст | Контрагент.Ідентифікатор | - |
| Код ЄДРПОУ/ІПН для ФОП | counterpartyCode | Однорядковий текст | Контрагент.Код ЄДРПОУ | - |
| Назва | counterpartyName | Однорядковий текст | Контрагент.Назва | - |
| **Приклад** | | | | |
| function onChangeContractorID() {      debugger;      var ContractorID = EdocsApi.getAttributeValue("ContractorID");        var type;        if (ContractorID.value) {        var counterpartyType = EdocsApi.getAttributeValue("counterpartyType");        if (counterpartyType.value) {          if (counterpartyType.value == "Покупець") {            type = "2";          } else {            type = "1";          }        } else {          throw "Не зазначено тип контрагента";        }          var ContractorData = EdocsApi.runExternalFunction("Contractors", "edocsGETCONTRACTOR?format=json&CONTRACTORID=" + ContractorID.value + "&CONTRACTORTYPE=" + type + "&MAXRESULTCOUNT=200", null, "get");          EdocsApi.setAttributeValue({ code: "counterpartyCode", value: ContractorData.data.Code });        EdocsApi.setAttributeValue({ code: "counterpartyName", value: ContractorData.data.ShortName });        EdocsApi.setAttributeValue({ code: "foreignShortName", value: ContractorData.data.ForeignShortName });        EdocsApi.setAttributeValue({ code: "counterpartyFullName", value: ContractorData.data.FullName });        EdocsApi.setAttributeValue({ code: "foreignFullName", value: ContractorData.data.ForeignFullName });        EdocsApi.setAttributeValue({ code: "counterpartyTaxStatus", value: ContractorData.data.TaxPayerStatus });        EdocsApi.setAttributeValue({ code: "counterpartyVatNumber", value: ContractorData.data.TaxId });        EdocsApi.setAttributeValue({ code: "ContractorIsResident", value: (ContractorData.data.NonResident && ContractorData.data.NonResident == true) || ContractorData.data.NonResident == "true" ? true : false });        //EdocsApi.setAttributeValue({code: 'counterpartyTaxStatusAddInfo', value: ContractorData.data.counterpartyTaxStatusAddInfo});        EdocsApi.setAttributeValue({ code: "counterpartyOwnershipType", value: setTypeOfOwnership(ContractorData.data.TypeOfOwnership) });        EdocsApi.setAttributeValue({ code: "CertificateNumber", value: ContractorData.data.VatNumber });        EdocsApi.setAttributeValue({ code: "counterpartyIdDocuments", value: ContractorData.data.IdDocument });        EdocsApi.setAttributeValue({ code: "counterpartyLegalAddress", value: ContractorData.data.LegalAddress });        EdocsApi.setAttributeValue({ code: "guid", value: ContractorData.data.ContractorId });        EdocsApi.setAttributeValue({ code: "additionalInfo", value: ContractorData.data.Information });        /\*if (EdocsApi.findElementByProperty('type', 'Phone', ContractorData.data.contractorContacts).contact) {                EdocsApi.setAttributeValue({code: 'counterpartyPhone', value: EdocsApi.findElementByProperty('type', 'Phone', ContractorData.data.contacts).contact});            }            if (EdocsApi.findElementByProperty('type', 'Email', ContractorData.data.contractorContacts).contact) {                EdocsApi.setAttributeValue({code: 'counterpartyEmail', value: EdocsApi.findElementByProperty('type', 'Email', ContractorData.data.contacts).contact});            }\*/        // if (ContractorData.data.PayerVAT) {        if (ContractorData.data.PayerVAT == true) {          EdocsApi.setAttributeValue({ code: "PayerVAT", value: "Так" });        } else {          EdocsApi.setAttributeValue({ code: "PayerVAT", value: "Ні" });        }        /\*} else {                clearAttribute ('PayerVAT');            }\*/        if (ContractorData.data.contacts && ContractorData.data.contacts.length > 0) {          EdocsApi.setAttributeValue({ code: "counterpartyPhone", value: ContractorData.data.contacts.find(x => x.Type == "Phone")?.Contact });          EdocsApi.setAttributeValue({ code: "counterpartyEmail", value: ContractorData.data.contacts.find(x => x.Type == "Email")?.Contact });            EdocsApi.setAttributeValue({ code: "PhoneID", value: ContractorData.data.contacts.find(x => x.Type == "Phone")?.Id });          EdocsApi.setAttributeValue({ code: "EmailID", value: ContractorData.data.contacts.find(x => x.Type == "Email")?.Id });        }          if (ContractorData.data.postalAddresses && ContractorData.data.postalAddresses.length > 0) {          var TablePostalAddress = EdocsApi.getAttributeValue("TableСounterpartyPostalAddress");          TablePostalAddress.value = [];            for (var i = 0; i < ContractorData.data.postalAddresses.length; i++) {            var row = ContractorData.data.postalAddresses[i];            var dataRow = [];            dataRow.push({ code: "counterpartyPostalAddress", value: row.PostalAddress });            dataRow.push({ code: "counterpartyPostalAddressCity", value: row.PostalAddressCity });              dataRow.push({ code: "counterpartyPostalAddressID", value: row.Id });              TablePostalAddress.value.push(dataRow);          }          EdocsApi.setAttributeValue(TablePostalAddress);        }        if (ContractorData.data.authorisedPersons && ContractorData.data.authorisedPersons.length > 0) {          var signersTable = EdocsApi.getAttributeValue("counterpartyContactsTable");          signersTable.value = [];            for (var i = 0; i < ContractorData.data.authorisedPersons.length; i++) {            var row = ContractorData.data.authorisedPersons[i];            var dataRow = [];            dataRow.push({ code: "counterpartyContactsName", value: row.FullName });            /\*dataRow.push({code:'counterpartyContactsNameGenitive', value:row.NameGenitive }); \*/            dataRow.push({ code: "counterpartyContactsPosition", value: row.Position });            /\*dataRow.push({code:'counterpartyContactsPositionGenitive', value:row.PositionGenitive });\*/            dataRow.push({ code: "ContractorBasisAct", value: row.ActingUnderThe });            dataRow.push({ code: "counterpartyContactsID", value: row.Id });              signersTable.value.push(dataRow);          }          EdocsApi.setAttributeValue(signersTable);        }        if (ContractorData.data.accounts && ContractorData.data.accounts.length > 0) {          var accountsTable = EdocsApi.getAttributeValue("TableOrganizationAccountNumber");          accountsTable.value = [];            for (var i = 0; i < ContractorData.data.accounts.length; i++) {            var row = ContractorData.data.accounts[i];            var curr = EdocsApi.getDictionaryData('Currency', row.Currency)[0];            var dataRow = [];            dataRow.push({ code: "OrganizationAccountNumber", value: row.Number });            dataRow.push({ code: "OrganizationBankName", value: row.Bank });            dataRow.push({ code: "MFO", value: row.Mfo });            dataRow.push({ code: "Currency", value: curr.id, text: curr.value, itemCode: curr.code });            dataRow.push({ code: "OrganizationAccountNumberID", value: row.Id });              accountsTable.value.push(dataRow);          }          EdocsApi.setAttributeValue(accountsTable);        }        if (ContractorData.data.managers && ContractorData.data.managers.length > 0) {          var managersTable = EdocsApi.getAttributeValue("counterpartyManagerTable");          managersTable.value = [];            for (var i = 0; i < ContractorData.data.managers.length; i++) {            var row = ContractorData.data.managers[i];            var dataRow = [];            dataRow.push({ code: "counterpartyManagerName", value: row.Name });            dataRow.push({ code: "counterpartyManagerPosition", value: row.Position });            dataRow.push({ code: "counterpartyManagerEmail", value: row.Email });            dataRow.push({ code: "counterpartyManagerPhone", value: row.Phone });              dataRow.push({ code: "counterpartyManagerID", value: row.Id });              managersTable.value.push(dataRow);          }          EdocsApi.setAttributeValue(managersTable);        }        onChangecounterpartyOwnershipType();        showChangeIdDocuments(ContractorData.data.TypeOfOwnership);          //onChangecounterpartyTaxStatus();        //onChangeContractorIsResident();      } else {        clearContractorData();      }    } | | | | |