How to work with repeating groups in FixEdge

Preface

FIXEdge offers solution for Business Rules purposed to provide routing, transferring, and data manipulations for FIX messages that go through the Business Layer of FIXEdge. See BL Scripting with JavaScript for more information.

Changing field values in repeating groups using JavaScript

This example describes how to change the Party Role (452) 's value from '1' (Executing Firm) to '17' (Contra Firm) in Execution Report (35=8).

Rule

The rule on the call of the JavaScript. Condition section sets up the condition for filtering messages by field MsgType (35) = 8.

BL_Config.xml

```
<Rule Description="Save Parties for every incoming Execution Report (35=8)">
        <Source Name=".*" />
                <Condition>
                            <MsgType>
                                        <Val>8</Val>
                            </MsgType>
                </Condition>
                <Action>
                        <Script Language="JavaScript" FileName ="updatePartyRole.js"/>
                        <!-- Do other necessary transformations and actions-->
                        <!-- ... -->
                </Action>
</Rule>
```

See Rule element for more information

JavaScript

Further, use JavaScript "updatePartyRole.js" on repeating group transformation:

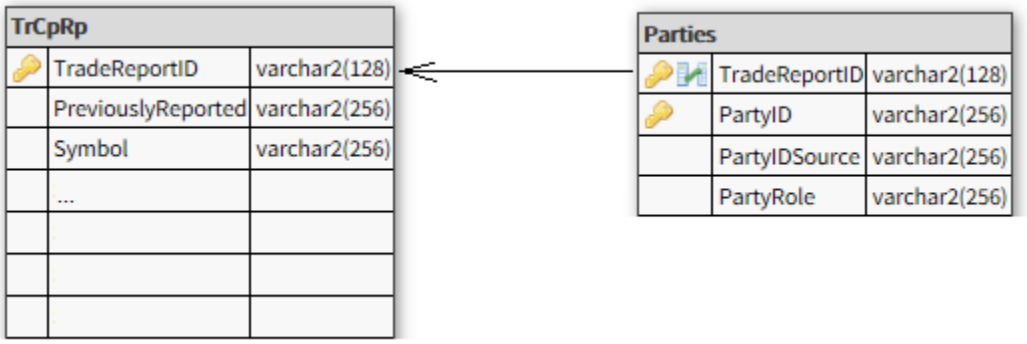
updatePartyRole.js

```
parties = getGroup(453);
groupSize = getNumField(453);

if ( (groupSize > 0) && isGroupValid(parties) )  // check if message is valid
{
    for (i=0; i < groupSize; ++i)                   // Repeat for all members in the repeating group
    {
            if (getNumField(parties, i, 452) == 17)               // If PartyRole in the first group equals
'17' change it to '1'
            {
                    setNumField(parties, i, 452, 1)                // Notice, that numbering starts from zero.
            }
    }
}
```

Storing repeating groups in Database

This example describes how to write the values from the Parties repeating group in Trade Capture Report (35=AE) to specific DB columns.



History Configuration

It is necessary to create the History definition in the Business Layer Configuration file and to register all the fields that are in the database table and in the order in which they are registered in the database.

BL_Config.xml

```
<History Name="Parties" StorageType="ODBC" MaxNumberOfRecords="15000" TableName="Parties" ColumnSize="256"
ConnectionString="DSN=TradeCap;UID= admin;Pwd=temp_pass;">
        <KeyField ColumnName="TradeReportID" ColumnSize="128">571</KeyField>
        <KeyField ColumnName="PartyID" ColumnSize="256">448</KeyField>
        <Field ColumnName="PartyIDSource" ColumnSize="256">447</Field>
        <Field ColumnName="PartyRole" ColumnSize="256">452</Field>
</History>
```

See Histories for more information

Rule

The rule on the call of the JavaScript

BL_Config.xml

```
<Rule Description="Save Parties for every incoming AE message">
        <Source Name=".*" />
        <Condition>
                <EqualField Field="35" Value="AE" />
        </Condition>
        <Action>
                <Script Language="JavaScript" FileName="TrdCapt.js" />
        </Action>
</Rule>
```

See Rule element for more information

JavaScript

JavaScript "TrdCapt.js" for storing in a DB

TrdCapt.js

```
tags = new Array(447, 452);                //In this array, we assign non-key tag numbers, below which the
loop will go through this array with getStringField.
partyKey = new Array();                    //An array in which the key fields 571 and 448 will be stored
partyKey.push(getStringField(571)); //We write down the first part of the key. In this example, this is
TradeReportID

hndl = getGroup(453);
groupSize = getNumField(453);
if (null != groupSize  and  isGroupValid(hndl))
{
        for (i = 0; i < groupSize; ++i)
        {
                partyData = new Array();                // An array for data to be stored in db. the values of
the 447 and 452 tags
                entry = i;
                partyKey.push(getStringField(hndl, i, 448));                        //Record the second part
of the key, PartyID
                if (null == getRecordFromHistory("Parties", partyKey))  //Check if there is already such an
entry in the database

{
                        for (j = 0; j < tags.length; ++j)                  // go through all fields to be
stored.

{                                                                                                   //
                                val = getStringField(hndl, i, tags[j]);
                                if (undefined == val) val = "";                                 // missing tag
= "", In the database for non-key fields it is desirable to enable write NULL
                                partyData.push(val);
                        }
                        saveToHistory("Parties", partyKey, partyData, "");                // Save records to
the database
                }
                else

{
                        //If such a record already exists in the database, then we write a notification and do
nothing.
                        print("Do nothing. Party duplicate has found in DB for " + partyKey);
                }
        }
}
```

Messages with the same TradeReportID (571) and PartyID (448) tags are considered as duplicates and are filtered.

For changing this behavior, KeyFields in History configuration should be modified/extended

Removing block from a repeating groups

For removing repeating group block we recommend the following steps:

1. Clear all fields from a group entry.
2. Copy all fields from the last group entry.
3. Resize group, i.e. set a new group size value to a specific tag.

Please note that this approach has certain limitations:

- Validation for repeating groups checks should be enabled. Messages with undefined tags and unexpected structure should be rejected or skipped.
- All possible fields should be handled in JavaScript. Unexpected tags should be restricted in fix dictionaries. Otherwise, JavaScript would make a mess with tags.

BL_Config.xml

The rule on the call of the JavaScript for removing a block from a repeating group.

BL_Config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<FIXEdge>
 <BusinessLayer>
  <Rule>
   <Source>
    <FixSession SenderCompID=".*" TargetCompID="FIXEDGE"/>
   </Source>
   <Action>
    <Script Language="JavaScript" FileName="FIXEdge1/conf/removePartyRole17.js"/>
    <Send Name="RECIPIENT" />
   </Action>
  </Rule>

  <DefaultRule>
   <Action>
    <DoNothing/>
   </Action>
  </DefaultRule>
 </BusinessLayer>
</FIXEdge>
```

JavaScript

Further, use JavaScript "removePartyRole17.js" for removing a block from a repeating group:

removePartyRole17.js

```
partiesGroupTags = new Array( 448, 447, 452, 2376 );

partiesNoCount = getNumField(453);
partiesGrp = getGroup(453);
partiesNewSize = partiesNoCount;
print("[DEBUG] Initial message has " + partiesNoCount.toString() + " parties"); // for debugging
if (0 < partiesNoCount && isGroupValid(partiesGrp))
{
 var i = 0;
 while (i < partiesNoCount && partiesNewSize > 0)
 {
  var partyRole = getNumField(partiesGrp, i, 452);
  if (partyRole == 17)
  {
   // Copy the last block to this position field by field if it exist
   for (var n=0; n < partiesGroupTags.length; ++n)
   {
    if( i == partiesNewSize-1) // is this is the last block
    {
     removeField(partiesGrp, i, partiesGroupTags[n]);
    }
    else
    {
     // partiesNewSize-1 - reference to the last block.
     lastBlockField = getStringField(partiesGrp, partiesNewSize-1, partiesGroupTags[n])
     currentBlockField = getStringField(partiesGrp, i, partiesGroupTags[n])
     print("[DEBUG] Current value of Field " + partiesGroupTags[n].toString() + " :" + currentBlockField + "
Last Field: " + lastBlockField ); // for debugging
     if ( lastBlockField != null && currentBlockField != null) // swap values with last group and clean it.
     {
      swapFields(partiesGrp, partiesNewSize-1, partiesGroupTags[n], partiesGrp, i, partiesGroupTags[n]);
      removeField(partiesGrp, partiesNewSize-1, partiesGroupTags[n]);
     }
     else if ( lastBlockField != null)
     {
      setStringField(partiesGrp, i, partiesGroupTags[n], lastBlockField ); // assumed only field that can be
converted to string can be absent
     }
     else //  currentBlockField != null and should be removed.
     {
      removeField(partiesGrp, i, partiesGroupTags[n]);
     }
    }
   }
   --partiesNewSize;
   print("[DEBUG] Removed party block #" + i.toString() + " New parties size = " + partiesNewSize.toString() );
// for debugging
   // Do not increment i because copied block could have partyRole = 17 and should be rechecked.
  }
  else
  {
   ++i;
  }
 }
 if (partiesNewSize > 0)
 {
  setNumField(453, partiesNewSize);
 }
 else // remove parties if there are no entries;
 {
  removeField(453);
 }
 print("[DEBUG] New parties size = " + partiesNewSize.toString()); // for debugging
}
```

Example of transformation

```
Sent message:
8=FIX.4.4|9=230|35=D|49=FIXEdge|56=FIXCLIENT|34=2|52=20180409-09:43:06.927
|11=E20170000000000000000001|453=3|448=TestValue1|447=D|452=11|448=TestValue2|447=D|452=122|448=XYZT|447=D|452=17
|1=EPAM-TEST|55=USD/CNH|54=1|15=USD|58=Remove block #3 of 3|10=045|

Processed message:
8=FIX.4.4|9=208|35=D|49=FIXCLIENT|56=FIXEdge|34=2|52=20180409-09:43:06.930
|11=E20170000000000000000001|453=2|448=TestValue1|447=D|452=11|448=TestValue2|447=D|452=122|1=EPAM-TEST|55=USD
/CNH|54=1|15=USD|58=Remove block #3 of 3|10=140|
```