Automated XML/XSL Transformation

Creator: Richard Ngo

Overview:

As a start-up mortgage company specializing in CRM software, Insellerate works with multiple clients and lead providers that do not always comply with the basic lead-posting format that Insellerate offers. Our initial workaround was integrating through Zapier to help convert their posting format into ours, but setting up a Zapier can quickly become very tedious on both parties (especially as we continue to grow and sign off with larger corporate clients). Recently, me and the rest of my Data Integrations Team have been working with XML/XSL transformations so that lead providers could use our posting APIs and send in lead data without having to change their format. While this is incredibly user-friendly, the process of hard-coding the field mapping so that their format works within our application, is just as tedious as setting up a Zapier.

As a way to make the XSLT process much more efficient for me and the rest my team, I created this notebook. Utilizing the Python knowledge I obtained from my time at university, I was able to automate both the mapping process and the rough creation of the XSLT through the use of widgets, file manipulation, and if-then logic.

Much thanks to @Riley Cohen, one of my best colleagues who provided me with the necessary inspiration, tools, and background knowledge that went into creating this notebook.

---

1. Definitions

Each cell in this notebook should be ran individually from start to finish.

In this cell, I import the necessary regex and widget packages and store code into functions and long strings into variables that I use more than once in my notebook.

The current function I have below takes an XML/XSL formatted text file and extracts the field names enclosed in the '<>' brackets.

The variables store XSL code that has been converted into strings for the dropdown fields my company uses. Unlike most of our other fields, these fields require specific upload values in order for them to be mapped correctly in Insellerate. The XSL code stored contains logic that iterates through various combinations of key words lead providers typically use for those fields (e.g. values like "VacationHome", "VACA" and "2nd" will be applicable and mapped to the same correct dropdown value in our system for fieldname: Residence Type). Including this logic for our dropdown fields is another feature that goes into the user-friendliness of my XSLT builder.

In [121…

```
import re
import ipywidgets as widgets

def showfile(txt):
    with open(txt,'r') as file:
        for line in file.readlines():
            print(line)
    file.close()

def fieldnames(txt):
    array=[]
    with open(txt, 'r') as file:
        for line in file:
            if line.find('</') != -1:
                array.append(line.rstrip()[line.find('</')+2:-1].rstrip())
    file.close()
    return(array)

loanpurpose = """\n<xsl:choose>\n<xsl:when test="contains(translate($loanPurp, $upperca
loanratetype = """\n<xsl:choose>\n<xsl:when test="contains(translate($rateType, $upperc
loantype = """\n<xsl:choose>\n<xsl:when test="contains(translate($loanType,$uppercase,$
propertytype = """\n<xsl:choose>\n<xsl:when test="(contains(translate($propType,$upperc
propertyuse = """\n<xsl:choose>\n<xsl:when test="contains(translate($propUse,$uppercase
employmentstatus = """\n<xsl:choose>\n<xsl:when test="contains(translate($empStatus,$up
```

2. File Paths

This is a cell that needs to be updated by the user to pinpoint where the existing text files are located and where the new files should be saved on their computer.

For the XSLT, we need two existing text files:

- Sample lead post from the vendor containing all of the fieldnames that need to be mapped

> *This needs to be in XML format. If in JSON, CSV, or web-encoded format, you can use third party websites such as freeformatter.com to convert this into XML*

- Library of database names in which the lead provider's fieldnames would be mapped

> *Note: This is technically static and does not to be stored in a designated file path (code would need to be configured), but I use an existing XSL file to retrieve my company's database names. While Insellerate always carries an updated list of available database names in their Dropbox, I prefer to extract the list of updated database names from the main source that is used in production...* 😋

We also need to decide where the created text files will be located:

- Cleaned use version of the sample lead post (removes whitespaces, corrects syntax, etc.)
- XSLT to be made

In [123…
```
xmlpost = 'C:/Users/RichardNgo/Downloads/leadpops.txt' # original XML lead post
xmlpost2 = 'C:/Users/RichardNgo/Downloads/leadpops2.txt' # cleaned ver of XML lead post
insxslt = 'C:/Users/RichardNgo/Documents/XSLT/Insellerate.xslt' # Insellerate XSLT
newxslt = 'C:/Users/RichardNgo/Downloads/test.txt' # file for new XSLT
```

3. Clean and Store Data

The following code copies the contents of the lead post and cleans up the format to be used for the XSLT. The vendor's fieldnames and Insellerate database names are also stored in arrays to be used for the mapping that comes before.

In [128…

```
with open(xmlpost,'r') as fin, open(xmlpost2,'w') as fout:
    for line in fin:
        fout.write(re.sub('\s*([>])', r'\1', re.sub('([</])\s*', r'\1', line)))
fin.close()
fout.close()

fields = fieldnames(xmlpost2)
dbnames = fieldnames(insxslt)

showfile(xmlpost2) # display lead post
```

```
<MortgageLead>

    <LeadSourceId>98</LeadSourceId>

    <SourceLeadId>a2632734b2920bf39a2da66097152804162132 7686-16213276862013</SourceLeadI
d>

    <SourceCreated>2021-05-18 02:04:18</SourceCreated>

    <HasCoBorrower>No</HasCoBorrower>

    <IPAddress>124.29.215.37</IPAddress>

    <ContactInfo>

        <Email>test@test-leadpops.com</Email>

        <Phone1>(111) 111-1111</Phone1>

        <Phone1Type>Home</Phone1Type>

    </ContactInfo>

    <Borrower>

        <FirstName>test</FirstName>

        <LastName>lead</LastName>

        <MailCity>SCHENECTADY</MailCity>

        <MailState>NY</MailState>

        <MailZip>12345</MailZip>

        <CreditRating>Excellent</CreditRating>

    </Borrower>

    <Property>

        <PropertyValue>230001</PropertyValue>

        <PropertyType>SingleFamily</PropertyType>

        <PropertyUsage>firsthome</PropertyUsage>

    </Property>

    <Loan>
```

```
            <LoanType>Refinance</LoanType>

            <LoanAmount>190001</LoanAmount>

            <LoanToValue>17.39</LoanToValue>

            <FirstMortgageBalance>190001</FirstMortgageBalance>

            <FirstMortgageRate>5.00%</FirstMortgageRate>

            <DesiredRateType>Fixed</DesiredRateType>

        </Loan>

        <LatePayments>1 Late Payment</LatePayments>

        <FhaLoan>No</FhaLoan>

        <MonthlyIncome>3501</MonthlyIncome>

        <MonthlyExpenses>2501</MonthlyExpenses>

        <ProofOfIncome>No</ProofOfIncome>

        <RecentBankruptcy>No</RecentBankruptcy>

        <EmploymentStatus>Self Employed</EmploymentStatus>

        <AdditionalCash>$0 (No Cash)</AdditionalCash>

        <SecondMortgage>Yes</SecondMortgage>

        <ExistingRateType>Fixed</ExistingRateType>

        <FirstPurchase>1</FirstPurchase>

        <YearOfPurchase>2010-2021</YearOfPurchase>

        <DeviceType>Computer</DeviceType>

        <StickyBarLead>No</StickyBarLead>

        <StickyBarURL>N/A</StickyBarURL>

        <PubID>123456</PubID>

    </MortgageLead>
```

4. Mapping UI

The following code creates a selection UI that is used for the mapping. User must choose from the searchbars which database names maps to the corresponding vendor's fieldnames.

In [129…

```python
mappings = []
for i in fields:
    mappings.append(widgets.Combobox(
    options=dbnames,
    placeholder='Select database name',
    description=i,
    style={'description_width': '150px'},
    layout = {'width': '350px'},
    ensure_option=True,
    disabled=False,
))
widgets.VBox(mappings)
```

5. The XSLT

Once the mapping is complete, the last piece of code will read the mapping and write the expected XSL file. The location of this XSL file is determined by the file path of 'newxslt'.

In [130…

```
dbmapped = [] # stores the ordered database names that were selected for the mapping
for j in range(len(fields)):
    dbmapped.append(mappings[j].value)

with open(xmlpost2,'r+') as fin, open(newxslt,'w') as fout:
    fout.write("""<?xml version="1.0" encoding="UTF-8"?>\n<xsl:stylesheet version="1.0"
    for line in fin:
        for k in range(len(fields)): # searches for patterns in the lead post and repla
            if re.search('<%s>' % fields[k], line) and not re.search('</%s>' % fields[k
                fout.write(re.sub('<%s>' % fields[k], """<xsl:for-each select='%s'>"""
            if re.search('<%s>.*?</%s>' % (fields[k], fields[k]), line):
                # check for Insellerate dropdown fields and add separate code (stored i
                if(dbmapped[k]=='FirstLoanPurposeTypeProposed'):
                    fout.write(re.sub('<%s>.*?</%s>' % (fields[k], fields[k]), """<%s>%
                elif(dbmapped[k]=='FirstLoanRateTypeProposed'):
                    fout.write(re.sub('<%s>.*?</%s>' % (fields[k], fields[k]), """<%s>%
                elif(dbmapped[k]=='FirstLoanTypeProposed'):
                    fout.write(re.sub('<%s>.*?</%s>' % (fields[k], fields[k]), """<%s>%
                elif(dbmapped[k]=='PropertyTypeId'):
                    fout.write(re.sub('<%s>.*?</%s>' % (fields[k], fields[k]), """<%s>%
                elif(dbmapped[k]=='ResidenceTypeId'):
                    fout.write(re.sub('<%s>.*?</%s>' % (fields[k], fields[k]), """<%s>%
                elif(dbmapped[k]=='BorrowerEmploymentStatus'):
                    fout.write(re.sub('<%s>.*?</%s>' % (fields[k], fields[k]), """<%s>%
                else: # write normal select statement
                    fout.write(re.sub('<%s>.*?</%s>' % (fields[k], fields[k]), """<%s><
            if re.search('</%s>' % fields[k], line) and not re.search('<%s>' % fields[k
                fout.write(re.sub('</%s>' % fields[k], """</xsl:for-each>""", line))
fin.close()
fout.close()

showfile(newxslt) # display XSLT
```

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:variable name="smallcase" select="'abcdefghijklmnopqrstuvwxyz'" />

<xsl:variable name="uppercase" select="'ABCDEFGHIJKLMNOPQRSTUVWXYZ'" />

<xsl:template match="/insellerate">

<xsl:variable name="propUse" select="*[translate(local-name(), $uppercase, $smallcase)
='residencetypeid']"/>

<xsl:variable name="propType" select="*[translate(local-name(), $uppercase, $smallcase)
='propertytypeid']"/>

<xsl:variable name="loanPurp" select="*[translate(local-name(), $uppercase, $smallcase)
='loan1_purposetype_proposed']"/>
```

```
<xsl:variable name="loanType" select="*[translate(local-name(), $uppercase, $smallcase)
='loan1_type_proposed']"/>

<xsl:variable name="rateType" select="*[translate(local-name(), $uppercase, $smallcase)
='loan1_ratetype_proposed']"/>

<xsl:variable name="empStatus" select="*[translate(local-name(), $uppercase, $smallcase)
='employment_status']"/>

<xsl:for-each select='MortgageLead'>

    <RefId><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='lead
sourceid']"/></RefId>

    <><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='sourcelea
did']"/></>

    <><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='sourcecre
ated']"/></>

    <><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='hascoborr
ower']"/></>

    <><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='ipaddres
s']"/></>

    <xsl:for-each select='ContactInfo'>

        <BorrowerEmailAddress><xsl:value-of select="*[translate(local-name(), $uppercas
e, $smallcase)='email']"/></BorrowerEmailAddress>

        <BorrowerHomePhone><xsl:value-of select="*[translate(local-name(), $uppercase,
$smallcase)='phone1']"/></BorrowerHomePhone>

        <><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='phone
1type']"/></>

    </xsl:for-each>

    <xsl:for-each select='Borrower'>

        <BorrowerFirstName><xsl:value-of select="*[translate(local-name(), $uppercase,
$smallcase)='firstname']"/></BorrowerFirstName>

        <BorrowerLastName><xsl:value-of select="*[translate(local-name(), $uppercase, $s
mallcase)='lastname']"/></BorrowerLastName>

        <BorrowerMailingCity><xsl:value-of select="*[translate(local-name(), $uppercase,
$smallcase)='mailcity']"/></BorrowerMailingCity>

        <BorrowerMailingState><xsl:value-of select="*[translate(local-name(), $uppercas
e, $smallcase)='mailstate']"/></BorrowerMailingState>

        <BorrowerMailingPostalCode><xsl:value-of select="*[translate(local-name(), $uppe
rcase, $smallcase)='mailzip']"/></BorrowerMailingPostalCode>

        <BorrowerCreditRating><xsl:value-of select="*[translate(local-name(), $uppercas
e, $smallcase)='creditrating']"/></BorrowerCreditRating>

    </xsl:for-each>

    <xsl:for-each select='Property'>

        <HomeValueCurrent><xsl:value-of select="*[translate(local-name(), $uppercase, $s
mallcase)='propertyvalue']"/></HomeValueCurrent>
```

```
        <PropertyTypeId>

<xsl:choose>

<xsl:when test="(contains(translate($propType,$uppercase,$smallcase), 'sfr') or contains
(translate($propType,$uppercase,$smallcase), 'sing')) and contains(translate($propTyp
e,$uppercase,$smallcase), 'attach')">17</xsl:when>

<xsl:when test="(contains(translate($propType,$uppercase,$smallcase), 'sfr') or contains
(translate($propType,$uppercase,$smallcase), 'sing')) and contains(translate($propTyp
e,$uppercase,$smallcase), 'detach')">18</xsl:when>

<xsl:when test="(contains(translate($propType,$uppercase,$smallcase), 'pud') or (contain
s(translate($propType,$uppercase,$smallcase), 'plan') and contains(translate($propTyp
e,$uppercase,$smallcase), 'urban'))) and contains(translate($propType,$uppercase,$smallc
ase), 'attach')">19</xsl:when>

<xsl:when test="(contains(translate($propType,$uppercase,$smallcase), 'pud') or (contain
s(translate($propType,$uppercase,$smallcase), 'plan') and contains(translate($propTyp
e,$uppercase,$smallcase), 'urban'))) and contains(translate($propType,$uppercase,$smallc
ase), 'detach')">20</xsl:when>

<xsl:when test="(contains(translate($propType,$uppercase,$smallcase), 'manufactur') or c
ontains(translate($propType,$uppercase,$smallcase), 'mfd')) and contains(translate($prop
Type,$uppercase,$smallcase), 'condo')">10</xsl:when>

<xsl:when test="(contains(translate($propType,$uppercase,$smallcase), 'pud') or (contain
s(translate($propType,$uppercase,$smallcase), 'plan') and contains(translate($propTyp
e,$uppercase,$smallcase), 'urban'))) and contains(translate($propType,$uppercase,$smallc
ase), 'condo')">10</xsl:when>

<xsl:when test="(contains(translate($propType,$uppercase,$smallcase), 'coop') or contain
s(translate($propType,$uppercase,$smallcase), 'co-op')) and contains(translate($propTyp
e,$uppercase,$smallcase), 'condo')">10</xsl:when>

<xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'mix') and contains
(translate($propType,$uppercase,$smallcase), 'use') and contains(translate($propType,$up
percase,$smallcase), 'high') and contains(translate($propType,$uppercase,$smallcase), 'c
ondo')">14</xsl:when>

<xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'no') and contains
(translate($propType,$uppercase,$smallcase), 'warrant') and contains(translate($propTyp
e,$uppercase,$smallcase), 'condo')">13</xsl:when>

<xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'high') and contain
s(translate($propType,$uppercase,$smallcase), 'condo')">7</xsl:when>

<xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'mid') and contains
(translate($propType,$uppercase,$smallcase), 'condo')">11</xsl:when>

<xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'low') and contains
(translate($propType,$uppercase,$smallcase), 'condo')">16</xsl:when>

<xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'site') and contain
s(translate($propType,$uppercase,$smallcase), 'condo')">12</xsl:when>

<xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'detach') and conta
ins(translate($propType,$uppercase,$smallcase), 'condo')">9</xsl:when>

<xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'condotel')">15</xs
l:when>

<xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'condo') or contain
s(translate($propType,$uppercase,$smallcase), 'town')">3</xsl:when>
```

```
        <xsl:when test="(contains(translate($propType,$uppercase,$smallcase), 'coop') or contain
        s(translate($propType,$uppercase,$smallcase), 'co-op')) and (contains(translate($propTyp
        e,$uppercase,$smallcase), 'pud') or (contains(translate($propType,$uppercase,$smallcas
        e), 'plan') and contains(translate($propType,$uppercase,$smallcase), 'urban')))">10</xs
        l:when>

        <xsl:when test="(contains(translate($propType,$uppercase,$smallcase), 'manufactur') or c
        ontains(translate($propType,$uppercase,$smallcase), 'mfd')) and (contains(translate($pro
        pType,$uppercase,$smallcase), 'pud') or (contains(translate($propType,$uppercase,$smallc
        ase), 'plan') and contains(translate($propType,$uppercase,$smallcase), 'urban')))">10</x
        sl:when>

        <xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'pud') or (contains
        (translate($propType,$uppercase,$smallcase), 'plan') and contains(translate($propType,$u
        ppercase,$smallcase), 'urban'))">8</xsl:when>

        <xsl:when test="(contains(translate($propType,$uppercase,$smallcase), 'manufactur') or c
        ontains(translate($propType,$uppercase,$smallcase), 'mfd')) and (contains(translate($pro
        pType,$uppercase,$smallcase), 'coop') or contains(translate($propType,$uppercase,$smallc
        ase), 'co-op'))">10</xsl:when>

        <xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'coop') or contains
        (translate($propType,$uppercase,$smallcase), 'co-op')">4</xsl:when>

        <xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'manufactur') or co
        ntains(translate($propType,$uppercase,$smallcase), 'mfd')">5</xsl:when>

        <xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'mh') and contains
        (translate($propType,$uppercase,$smallcase), 'select')">6</xsl:when>

        <xsl:when test="contains(translate($propType,$uppercase,$smallcase), 'attach') or contai
        ns(translate($propType,$uppercase,$smallcase), 'multi') or (contains(translate($propTyp
        e,$uppercase,$smallcase), 'unit') and $propType != translate($propType,'23456789
        0',''))">1</xsl:when>

        </xsl:choose>

        </PropertyTypeId>

                <ResidenceTypeId>

        <xsl:choose>

        <xsl:when test="contains(translate($propUse,$uppercase,$smallcase), 'primary') or contai
        ns(translate($propUse,$uppercase,$smallcase), 'own') or contains(translate($propUse,$upp
        ercase,$smallcase), 'first') or contains(translate($propUse,$uppercase,$smallcase), '1s
        t')">1</xsl:when>

        <xsl:when test="contains(translate($propUse,$uppercase,$smallcase), 'vaca') or contains
        (translate($propUse,$uppercase,$smallcase), 'second') or contains(translate($propUse,$up
        percase,$smallcase), '2nd')">2</xsl:when>

        <xsl:when test="contains(translate($propUse,$uppercase,$smallcase), 'rent') or contains
        (translate($propUse,$uppercase,$smallcase), 'invest')">3</xsl:when>

        </xsl:choose>

        </ResidenceTypeId>

            </xsl:for-each>

            <xsl:for-each select='Loan'>

                <FirstLoanTypeProposed>
```

```
<xsl:choose>

<xsl:when test="contains(translate($loanType,$uppercase,$smallcase), 'jumbo')">12</xsl:w
hen>

<xsl:when test="contains(translate($loanType, $uppercase, $smallcase), 'equity') or cont
ains(translate($loanType, $uppercase, $smallcase), 'heloc')">8</xsl:when>

<xsl:when test="contains(translate($loanType,$uppercase,$smallcase), 'cal') and contains
(translate($loanType,$uppercase,$smallcase), 'hfa')">11</xsl:when>

<xsl:when test="(contains(translate($loanType,$uppercase,$smallcase), 'fha') or contains
(translate($loanType,$uppercase,$smallcase), 'fed'))">2</xsl:when>

<xsl:when test="contains(translate($loanType,$uppercase,$smallcase), 'usda') or contains
(translate($loanType,$uppercase,$smallcase), 'rural')">4</xsl:when>

<xsl:when test="contains(translate($loanType,$uppercase,$smallcase), 'construct')">6</xs
l:when>

<xsl:when test="contains(translate($loanType,$uppercase,$smallcase), 'fixed') and (conta
ins(translate($loanType,$uppercase,$smallcase), 'second') or contains(translate($loanTyp
e,$uppercase,$smallcase), '2nd'))">7</xsl:when>

<xsl:when test="contains(translate($loanType,$uppercase,$smallcase), 'reverse')">9</xsl:
when>

<xsl:when test="contains(translate($loanType,$uppercase,$smallcase), 'no') and (contains
(translate($loanType,$uppercase,$smallcase), 'qm') or contains(translate($loanType,$uppe
rcase,$smallcase), 'qualif'))">10</xsl:when>

<xsl:when test="contains(translate($loanType,$uppercase,$smallcase), 'conv')">3</xsl:whe
n>

<xsl:when test="contains(translate($loanType,$uppercase,$smallcase), 'vet') or contains
(translate($loanType,$uppercase,$smallcase), 'va')">1</xsl:when>

<xsl:when test="$loanType != ''">5</xsl:when>

</xsl:choose>

</FirstLoanTypeProposed>

        <FirstLoanInitialAmountProposed><xsl:value-of select="*[translate(local-name(),
$uppercase, $smallcase)='loanamount']"/></FirstLoanInitialAmountProposed>

        <LoanToValue><xsl:value-of select="*[translate(local-name(), $uppercase, $smallc
ase)='loantovalue']"/></LoanToValue>

        <FirstLoanBalanceCurrent><xsl:value-of select="*[translate(local-name(), $upperc
ase, $smallcase)='firstmortgagebalance']"/></FirstLoanBalanceCurrent>

        <FirstLoanRateCurrent><xsl:value-of select="*[translate(local-name(), $uppercas
e, $smallcase)='firstmortgagerate']"/></FirstLoanRateCurrent>

        <FirstLoanRateTypeProposed>

<xsl:choose>

<xsl:when test="contains(translate($rateType, $uppercase, $smallcase), 'equity') or cont
ains(translate($rateType, $uppercase, $smallcase), 'heloc')">3</xsl:when>

<xsl:when test="contains(translate($rateType, $uppercase, $smallcase), 'fix')">1</xsl:wh
en>
```

```
<xsl:when test="contains(translate($rateType, $uppercase, $smallcase), 'grad')">2</xsl:w
hen>

<xsl:when test="contains(translate($rateType, $uppercase, $smallcase), 'adj')">4</xsl:wh
en>

<xsl:when test="$rateType != ''">5</xsl:when>

</xsl:choose>

</FirstLoanRateTypeProposed>

    </xsl:for-each>

    <><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='latepayme
nts']"/></>

    <><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='fhaloa
n']"/></>

    <BorrowerBaseIncome><xsl:value-of select="*[translate(local-name(), $uppercase, $sma
llcase)='monthlyincome']"/></BorrowerBaseIncome>

    <BorrowerTotalLiabilities><xsl:value-of select="*[translate(local-name(), $uppercas
e, $smallcase)='monthlyexpenses']"/></BorrowerTotalLiabilities>

    <><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='proofofin
come']"/></>

    <Bankruptcy><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)
='recentbankruptcy']"/></Bankruptcy>

    <BorrowerEmploymentStatus>

<xsl:choose>

<xsl:when test="contains(translate($empStatus,$uppercase,$smallcase), 'retire') and cont
ains(translate($empStatus,$uppercase,$smallcase), 'work')">4</xsl:when>

<xsl:when test="contains(translate($empStatus,$uppercase,$smallcase), 'unemploy') or (co
ntains(translate($empStatus,$uppercase,$smallcase), 'not') and contains(translate($empSt
atus,$uppercase,$smallcase), 'employ'))">5</xsl:when>

<xsl:when test="contains(translate($empStatus,$uppercase,$smallcase), 'home') or contain
s(translate($empStatus,$uppercase,$smallcase), 'house')">7</xsl:when>

<xsl:when test="contains(translate($empStatus,$uppercase,$smallcase), 'student')">8</xs
l:when>

<xsl:when test="contains(translate($empStatus,$uppercase,$smallcase), 'part')">2</xsl:wh
en>

<xsl:when test="contains(translate($empStatus,$uppercase,$smallcase), 'self')">6</xsl:wh
en>

<xsl:when test="contains(translate($empStatus,$uppercase,$smallcase), 'full') or contain
s(translate($empStatus,$uppercase,$smallcase), 'employed')">1</xsl:when>

<xsl:when test="contains(translate($empStatus,$uppercase,$smallcase), 'retire')">3</xsl:
when>

<xsl:when test="$empStatus != ''">9</xsl:when>

</xsl:choose>
```

```
</BorrowerEmploymentStatus>

    <RequestedCashoutAmount><xsl:value-of select="*[translate(local-name(), $uppercase,
$smallcase)='additionalcash']"/></RequestedCashoutAmount>

    <Other102><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='s
econdmortgage']"/></Other102>

    <FirstLoanRateTypeCurrent><xsl:value-of select="*[translate(local-name(), $uppercas
e, $smallcase)='existingratetype']"/></FirstLoanRateTypeCurrent>

    <BorrowerIsFirstHomeBuyer><xsl:value-of select="*[translate(local-name(), $uppercas
e, $smallcase)='firstpurchase']"/></BorrowerIsFirstHomeBuyer>

    <Other47><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='ye
arofpurchase']"/></Other47>

    <><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='devicetyp
e']"/></>

    <><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='stickybar
lead']"/></>

    <><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='stickybar
url']"/></>

    <Other43><xsl:value-of select="*[translate(local-name(), $uppercase, $smallcase)='pu
bid']"/></Other43>

</xsl:for-each>
```

Closing Notes

As always, code is subject to change-- either as a reflection of other changes in the company or for broader optimization purposes. More optimized logic can be accounted for and new ideas will be discovered for a better use of this notebook.