iConnect 186362 Design Approach – Address Validation

Project Overview

This project is to integrate with the Lob.com address validation API in order to accomplish address validation on a set of fields without the user needing to trigger the validation (through a custom action button, for example).

Validation will be set through the admin tool via a rule node that will return various string values, allowing the rule to further determine logic based upon the success or failure of the validation check (such as error messages or warnings). Any messaging from Lob.com or from the FireLight server will additionally be shown as a warning message on an indicated data item.

The API to be used is described at https://lob.com/docs/java#us_verifications.

Features/Requirements

- A rule will be created to capture the various necessary inputs to accomplish the address validation. The rule will fully execute once all specified fields have input – any optional inputs (as defined for the rule) will be used if they have a value, otherwise will be ignored (note that an optional input can still trigger the full execution of the rule, so long as all other required inputs have value).
    - Rule will return “success” string if address is validated against the Lob.com API.
    - Rule will return “fail” string if address is not validated against the Lob.com API, or if Lob.com returned a validated address that includes a corrected address.
    - Rule will return “technicalFail” string if the call did not work due to technical issues.
    - Rule will return “didNotRun” string if the call was not accomplished due to not enough input (like a missing Line 1 for the address data).
    - Rule will have a “dataitemid” attribute, which will have any message details posted to it as a warning. For example, if the API call returned any messaging details on why the validation failed, this data item would receive that message as a warning.
- If Lob.com indicates that the given address has corrections (Lob returned a success message, but also a corrected address), a pop-up message will be shown to the user asking whether they wish to accept the corrected address, or if they wish to keep their inputted address.
    - If the corrected address is accepted, the various corrected address items (line 1, line 2, city, state, zip) will be inserted into the relevant data items as indicated with the rule.
    - If the corrected address is not accepted, all address items indicated as needing corrections will have warning messages posted on them.

App Changes

A new rule node (and sub-nodes) will be created to handle the desired rule. The rule will have logic to execute fully when all required fields are filled in by the user – this execution will include the action of

calling out to the Lob.com address validation API, and returning a string value based on the response received from the API call.

All required fields follow the following criteria:

- Street Line 1 is required.
- City & State fields are required if zip is not present.
- Zip is required if City & State fields are not present.

As long as the above required fields have input in the necessary manner, the Lob.com API call will occur.

The rule format will be as follows:

```
<AddressValidation dataitemid="dataItemToPostDetailsAsWarningsTo">
  <!--Line 1 street-->
  <AddressInput type="Address1" dataitemid="someDataItem line 1" />
  <!--Line 2 street - optional-->
  <AddressInput type="Address2" dataitemid="someDataItem line 2" />
  <!--City-->
  <AddressInput type="City" dataitemid="someDataItem_City" />
  <!--State-->
  <AddressInput type="State" dataitemid="someDataItem_State" />
  <!--zip-->
  <AddressInput type="Zip5" dataitemid="someDataItem_Zip" />
</AddressValidation>
```

The <AddressValidation> node has a 'dataitemid' attribute – this data item will receive any details from the API call, and FireLight will post those details as a warning message. Any further rules are free to suppress these warning messages if desired.

The <AddressValidation> node will return a string value as defined in the Requirements section.

Admin Changes

For the new rule, all admin functionality that normally accompanies rules will be implemented – auto-complete, errors & warnings, node help, etc.

Use Cases / Workflow Changes

In order to use this rule, various address data items will need to be set up to capture the relevant information needed for the rule. The top node in the rule (AddressValidation) will have a 'dataitemid' attribute – it uses this data item to post a warning message into based on the result of the API call.

The rule will return a string value, so that any resulting behavior can be set up in response. This resulting behavior could be something like an error/validation message, forcing the user to correct the issue before being able to submit the activity. However, it may be more useful to use a warning message instead, depending on the desired business logic.

The below example shows a rule utilizing the string return logic of the AddressValidation rule:

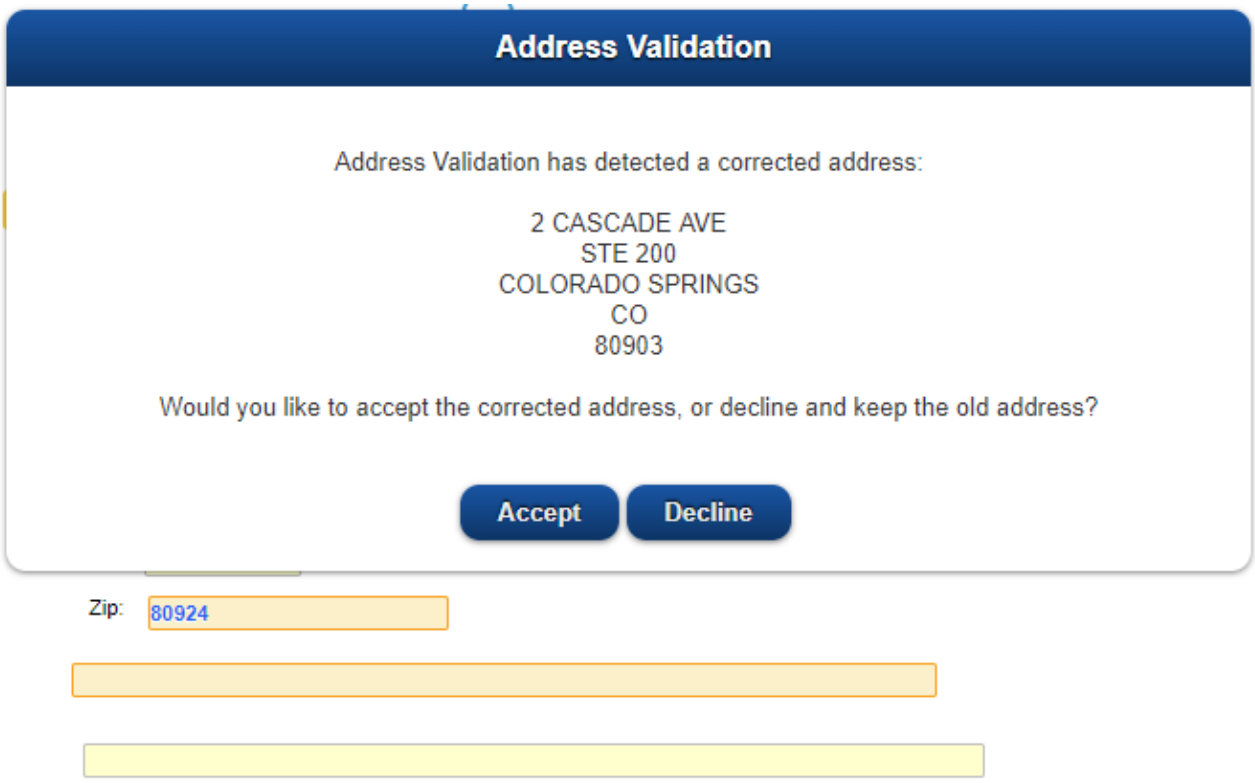
```
<if>
       <set name="variableName">
      <AddressValidation dataitemid="lob_Error_Text">
        <!--Line 1 street-->
        <AddressInput type="Address1" dataitemid="lob_address1" />
        <!--Line 2 street - optional-->
        <AddressInput type="Address2" dataitemid="lob_address2" />
        <!--City-->
        <AddressInput type="City" dataitemid="lob_city" />
        <!--State-->
        <AddressInput type="State" dataitemid="lob_state" />
        <!--zip-->
        <AddressInput type="Zip5" dataitemid="lob_zip" />
      </AddressValidation>
       </set>
       <condition>
              <compare op="==">
                     <const value="fail" type="String" />
                     <get name="variableName" />
              </compare>
       </condition>
       <postmessage dataitemid="lob_address1">
              <const value="Address did not validate. Please correct it." />
       </postmessage>
       <elseif>
              <compare op="==">
                     <const value="technicalFail" type="String" />
                     <get name="variableName" />
              </compare>
       </elseif>
       <postmessage dataitemid="lob_address1" option="Warning">
              <const value="Address Validation service not working." />
       </postmessage>
       <elseif>
              <isnullorwhitespace>
        <diget dataitemid="lob_address1" />
      </isnullorwhitespace>
       </elseif>
       <postmessage dataitemid="lob_address1">
              <const value="Please enter a full address." />
       </postmessage>
       <else />
       <removemessage dataitemid="lob_address1" />
</if>
```

In the above rule, any description received back from the Lob.com service will be shown as a warning message on the “lob_Warning_Text” data item. If the validation does not pass, an error message will be shown on the “lob_address1” data item. If there was a technical problem, a warning message will be shown on the “lob_address1” data item. If the rule did not fully execute for any reason (such as a missing Line1), an error message will be shown on the “lob_address1” data item, indicating to the user to enter a value into that field.

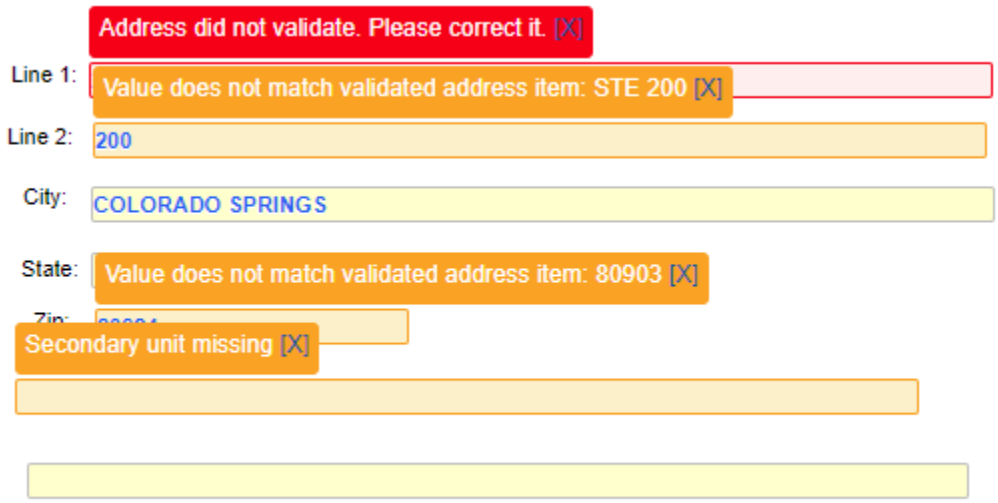
UI Mock Ups

Non-Valid Address:

Pop-Up Message indicating a corrected address:



Warning Messages after declining a corrected address (rule is also set up to post an error in the event of a failed validation, which may not be your desired set-up):



Areas Impacted

System Area	Yes	Comment
Admin Tool		
- Form Library		
- Design Forms	X	The rule editing window should contain the basic/default information relating to the new rule.
- Profile Administration	X	Rulesets should contain the basic/default information relating to the new rule.
- Reports		
- Deployment		
FireLight App		
- New Application		
- Edit Application	X	The rule will execute an API call to Lob.com while the user is in the flow of filling out the application, as defined.

- Signature Process		
- Review Queue		
- Manual Review		
- User Preferences		
- Inbound Integration		
- Outbound Integration		
- PDF Generation		
- Email System		
FireLight Console		
- Windows		
- iOS		
Other Systems		
- DTCC Integration		
- Commission Netting		
- Activity Reporting		