NEO PLUGIN
USER MANUAL
VERSION 1.0.3.2 – APRIL 7, 2019

Contents

I. What is Neo Plugin? 2

II. Advantages of NeoPlugin 3

III. How to Obtain the CRM Solution 3

IV. How to Install the CRM Solution 3

V. How to Create a New Plugin 4

VI. Execution Script Syntax 13

VII. List of Build-In Functions 17

- Math Functions 17
- String Functions 18
- XML Functions 19
- Date/Time Functions 21
- Flow Functions 21
- CRM Data Retrieval & Manipulation Functions 22
- CRM Plugin Functions 22
- Other Functions 22

VIII. Example Use Cases 23

IX. Export and Import Neo Plugins 35

X. How to Get Support for Neo Plugin 41

I. What is Neo Plugin?

Neo Plugin is an add-on for Dynamics 365, which enables to create Plugin in Dynamics 365 without going through code development and plugin registration process.

Neo Plugin allows users to create a Dynamics 365 plugin easily; run in server-side either synchronously, or asynchronously, being triggered with methods: Create, Update, Delete, SetState, etc, and with steps: Pre-Validation, Pre-Operation and Post-Operation.

Neo Plugin provides a middle layer, using a simple scripting language to define what the plugin will do. The execution script is entered into a text area in the Dynamics 365 UI, and when the record is saved, the execution script is verified syntactically and semantically. If there is any error in the script, for example an input or output attribute does not actually exist in the entity, an error message pops up, and user should fix the script to be able to proceed.

Besides regular mathematical operations, Neo Plugin provides a set of functions that allow users to achieve more complex tasks. The list of built-in functions can be found in Section VI of this document. Please keep in mind that this list is not final, and we are open to any suggestion that you think it will improve the tool.

Of course, there are some limitations to what you can do by using Neo Plugin. Section VII of this document gives some set of example use cases, which will give you some idea about what you can do with Neo Plugin and what you cannot. If you are not sure whether Neo Plugin can achieve a specific task or not, you can always contact us and we will provide you some information about how to do it, if it is possible.

Triggers of the plugin are setup in the CRM form. Neo Plugin UI provides all features of the regular CRM plugin triggers.

Executions of Neo Plugins are logged within CRM for debugging and error reporting purposes. It is also possible to avoid logging successful executions of the plugin to reduce space usage by setting the “Log Policy” field to “Log Errors Only”, in the plugin setup.

This tool is very useful, especially when you need to do some mathematical calculations based on some input CRM fields and write results back to some CRM fields. By using this tool, you can easily convert your legacy excel worksheet pricing formulas into CRM plugins just by using the CRM UI.

II. Advantages of NeoPlugin

Neo Plugin has several advantages over other alternatives:

- All setups of the plugins are done via CRM UI.
- Complex operations can be achieved with NeoPlugin that cannot be done with CRM Workflows or Business Rules, like:
  - Retrieve multiple entity records by constructing fetchXml on the fly.
  - Looping with **foreach** keyword over multiple entity records to create, update, delete, etc.
  - Looping with **while** keyword
  - Conditional value assignment using **if-then-else** keywords.
  - Use pre-image values.
  - Call web services.
  - Use local variables to achieve complex calculations.
  - Use mathematical functions like log, sqrt, round, etc.
  - Use date/time functions like min, max, dateDiff, etc.
  - Use string functions like length, substring, trim, padLeft, etc.
  - Use conditional actions.
  - Check user roles.
  - Check team membership
- Very simple scripting language with rich built-in functions.
- Execution logs are stored with details.
- Easily export and import NeoPlugins between CRM organizations.

III. How to Obtain the CRM Solution

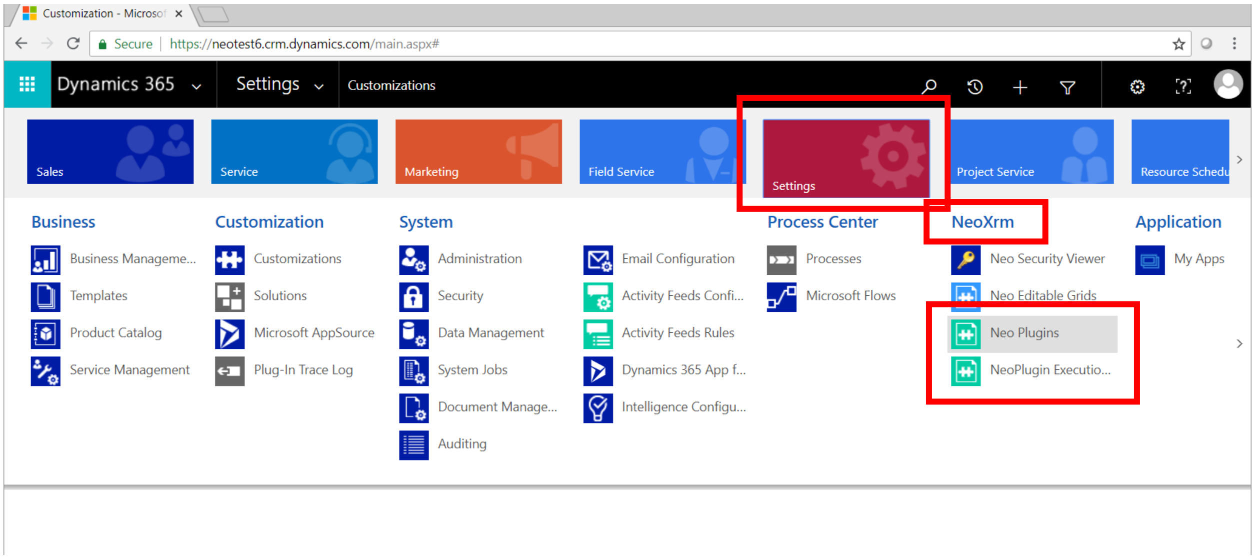
Neo Plugin is delivered as a managed CRM solution file when you register for FREE product from our website, https://www.neoxrm.com. At the time of registration, you need to provide the Dynamics 365 organization unique name that you want to install the solution. The delivered solution file will only work for this specific CRM instance. If you have provided a wrong Dynamics 365 organization name, please contact to NeoXrm support team via email support@neoxrm.com or phone 1-844-NEOXRM1.

IV. How to Install the CRM Solution

The next step is to import the downloaded Dynamics 365 solution into your environment. You need to have system admin or system customizer role to be able to import the Dynamics 365 solution in general. Once the solution is imported successfully, your Neo Plugin is ready to use.

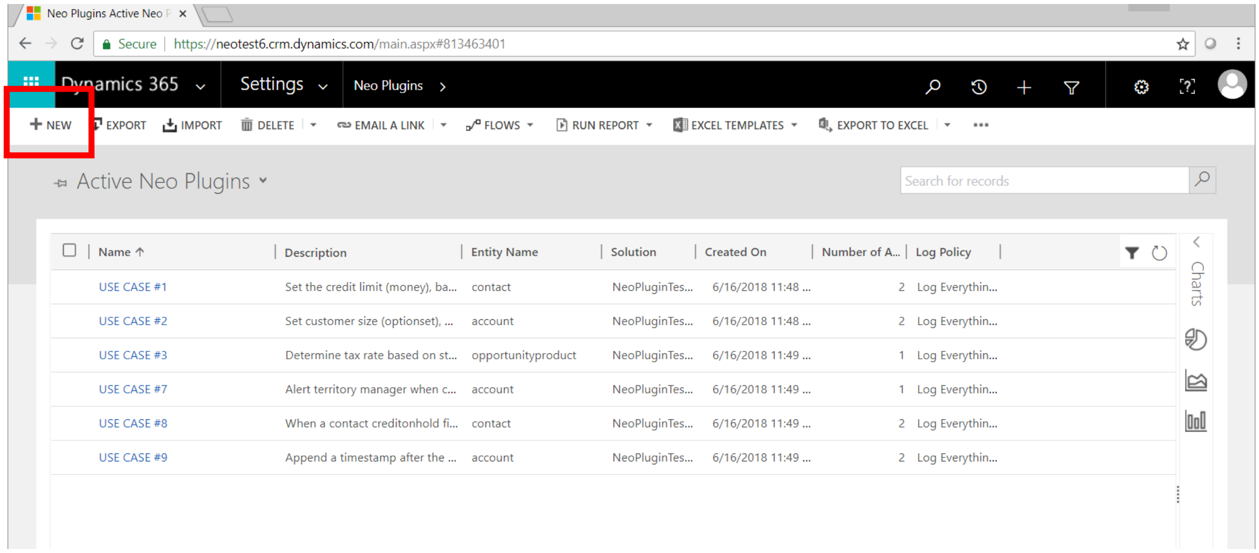
V. How to Create a New Plugin

Once the Neo Plugin solution is imported into Dynamics 365, Neo Plugins menu item is listed under the Settings -> NeoXrm section.



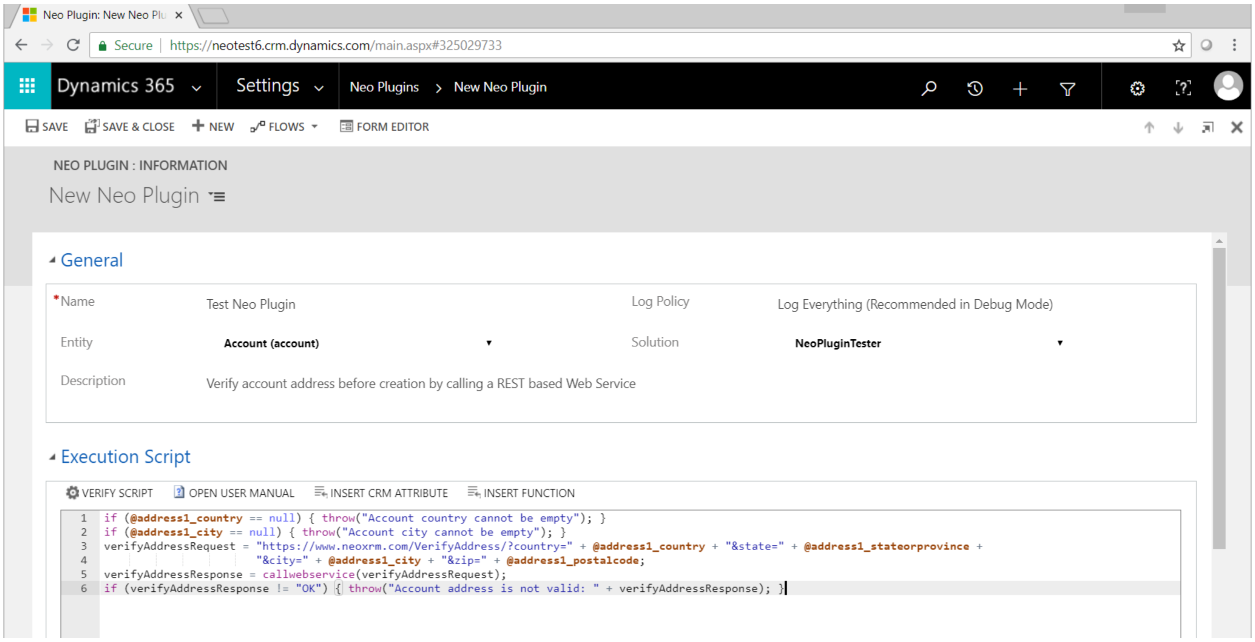
Click on the **Neo Plugins** button, the opened view will list all plugins.

Click on the **New** button to create a new plugin. It will open a new Neo Plugin form.



Name ↑	Description	Entity Name	Solution	Created On	Number of A...	Log Policy
USE CASE #1	Set the credit limit (money), ba...	contact	NeoPluginTes...	6/16/2018 11:48 ...	2	Log Everythin...
USE CASE #2	Set customer size (optionset), ...	account	NeoPluginTes...	6/16/2018 11:48 ...	2	Log Everythin...
USE CASE #3	Determine tax rate based on st...	opportunityproduct	NeoPluginTes...	6/16/2018 11:49 ...	1	Log Everythin...
USE CASE #7	Alert territory manager when c...	account	NeoPluginTes...	6/16/2018 11:49 ...	1	Log Everythin...
USE CASE #8	When a contact creditonhold fi...	contact	NeoPluginTes...	6/16/2018 11:49 ...	2	Log Everythin...
USE CASE #9	Append a timestamp after the ...	account	NeoPluginTes...	6/16/2018 11:49 ...	2	Log Everythin...

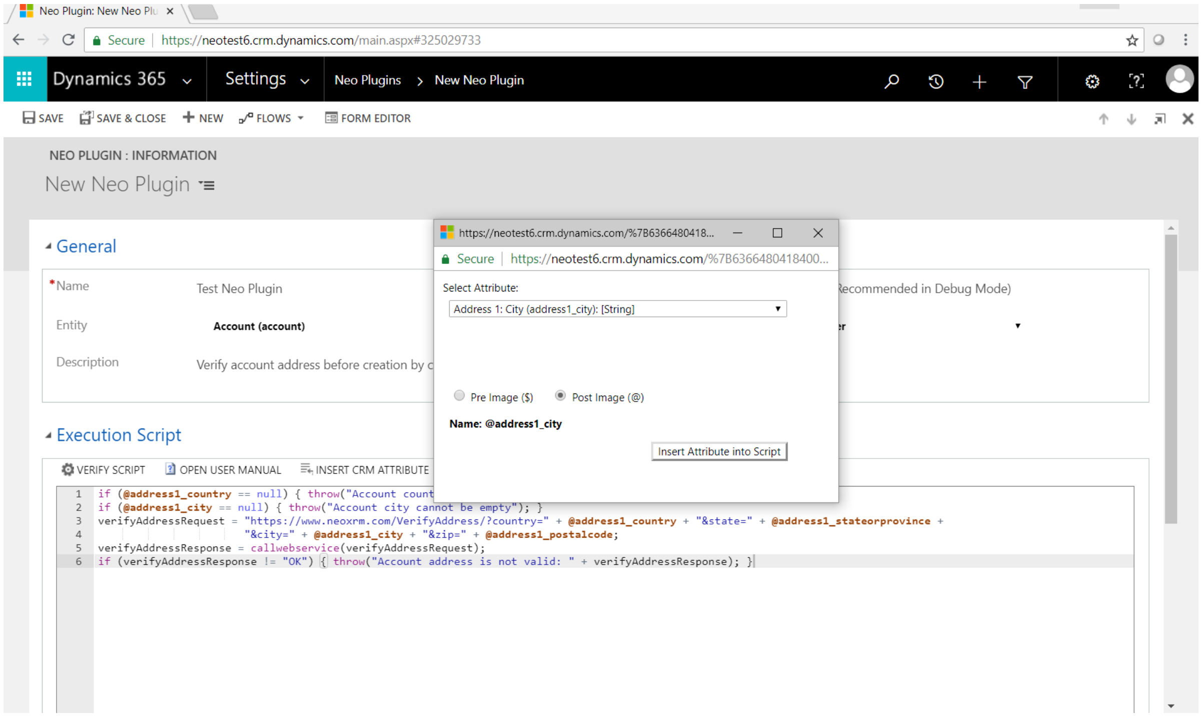
Enter a name, select the entity from the optionset, select a solution from the optionset, provide a description and provide an execution script. Click on the SAVE button to create the plugin. Remember that the **Entity** and **Solution** fields cannot be changed after creation.



*Name	Test Neo Plugin	Log Policy	Log Everything (Recommended in Debug Mode)
Entity	Account (account)	Solution	NeoPluginTester
Description	Verify account address before creation by calling a REST based Web Service		

```
if (@address1_country == null) { throw("Account country cannot be empty"); }
if (@address1_city == null) { throw("Account city cannot be empty"); }
verifyAddressRequest = "https://www.neoxrm.com/VerifyAddress/?country=" + @address1_country + "&state=" + @address1_stateorprovince +
                       "&city=" + @address1_city + "&zip=" + @address1_postalcode;
verifyAddressResponse = callwebservice(verifyAddressRequest);
if (verifyAddressResponse != "OK") { throw("Account address is not valid: " + verifyAddressResponse); }
```

Click on the INSERT ENTITY ATTRIBUTE button to add a entity attribute into execution script as input or output variable.



Neo Plugin: New Neo Plu
Secure | https://neotest6.crm.dynamics.com/main.aspx#325029733
Dynamics 365
Settings
Neo Plugins > New Neo Plugin
SAVE
SAVE & CLOSE
NEW
FLOWS
FORM EDITOR
NEO PLUGIN : INFORMATION
New Neo Plugin
General
Execution Script
VERIFY SCRIPT
OPEN USER MANUAL
INSERT CRM ATTRIBUTE
https://neotest6.crm.dynamics.com/%7B6366480418...
Secure | https://neotest6.crm.dynamics.com/%7B6366480418400...
Insert Attribute into Script

*Name: Test Neo Plugin

Entity: Account (account)

Description: Verify account address before creation by c

ecommended in Debug Mode)

Select Attribute: Address 1: City (address1_city): [String]

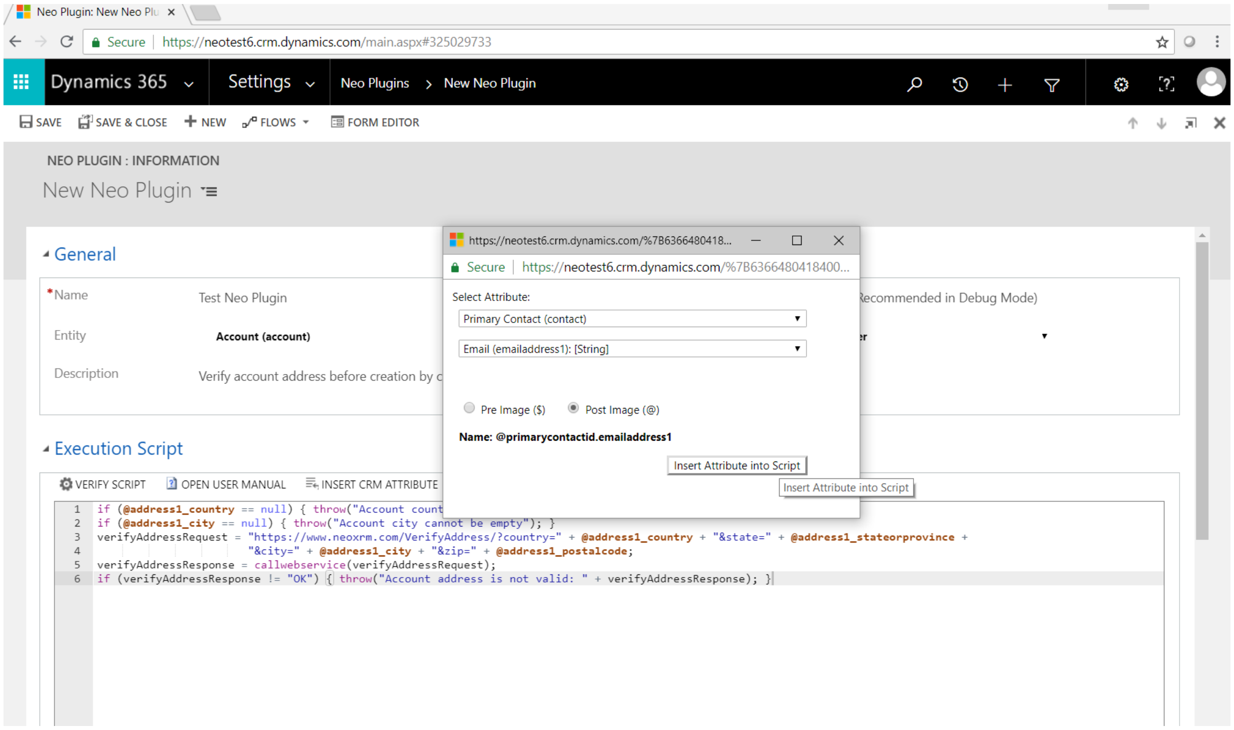
○ Pre Image ($) ● Post Image (@)

Name: @address1_city

```
if (@address1_country == null) { throw("Account count
if (@address1_city == null) { throw("Account city cannot be empty"); }
verifyAddressRequest = "https://www.neoxrm.com/VerifyAddress/?country=" + @address1_country + "&state=" + @address1_stateorprovince +
                       "&city=" + @address1_city + "&zip=" + @address1_postalcode;
verifyAddressResponse = callwebservice(verifyAddressRequest);
if (verifyAddressResponse != "OK") { throw("Account address is not valid: " + verifyAddressResponse); }
```

For some plugin messages (e.g. Update), pre-image value is also available. Attribute names starting with $ sign means reading the previous value of that attribute before the update.

For Reference type (Lookup) attributes, attributes of the referenced record can also be used in the execution.



Neo Plugin: New Neo Plu
Secure | https://neotest6.crm.dynamics.com/main.aspx#325029733
Dynamics 365
Settings
Neo Plugins > New Neo Plugin
SAVE
SAVE & CLOSE
NEW
FLOWS
FORM EDITOR

NEO PLUGIN : INFORMATION

New Neo Plugin

General

*Name: Test Neo Plugin

Entity: **Account (account)**

Description: Verify account address before creation by c

ecommended in Debug Mode)

https://neotest6.crm.dynamics.com/%7B6366480418...
Secure | https://neotest6.crm.dynamics.com/%7B6366480418400...

Select Attribute:

Primary Contact (contact)

Email (emailaddress1): [String]

○ Pre Image ($) ● Post Image (@)

Name: @primarycontactid.emailaddress1

Insert Attribute into Script
Insert Attribute into Script

Execution Script

VERIFY SCRIPT
OPEN USER MANUAL
INSERT CRM ATTRIBUTE

```
if (@address1_country == null) { throw("Account count
if (@address1_city == null) { throw("Account city cannot be empty"); }
verifyAddressRequest = "https://www.neoxrm.com/VerifyAddress/?country=" + @address1_country + "&state=" + @address1_stateorprovince +
                       "&city=" + @address1_city + "&zip=" + @address1_postalcode;
verifyAddressResponse = callwebservice(verifyAddressRequest);
if (verifyAddressResponse != "OK") { throw("Account address is not valid: " + verifyAddressResponse); }
```

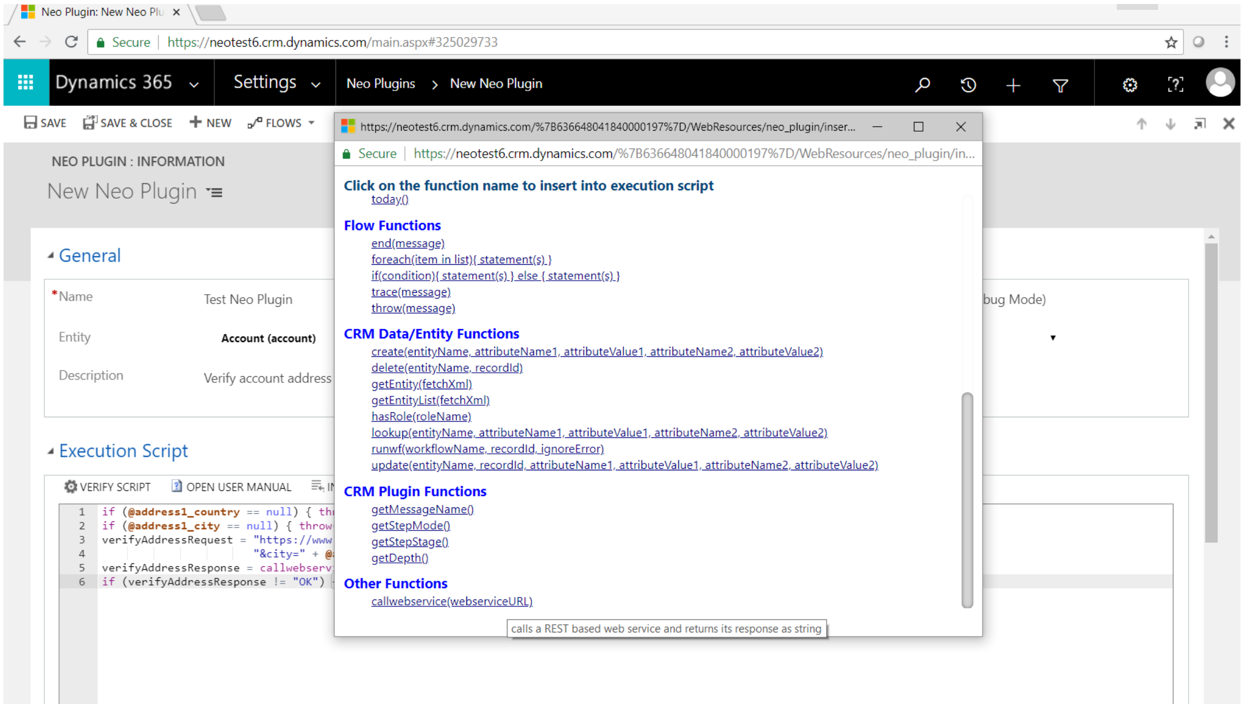
In the above example where the Entity is account, **@primarycontactid.emailaddress1** brings the email address value of the primary contact of the account.

Reading the referenced entity values is possible up-to three levels.

It is possible to use referenced attributes for pre-event values as well.

For example, **$primarycontactid.emailaddress1** is the email address value of the previous primary contact, when plugin is triggered with update of primary contact of the account.

Using the INSERT FUNCTION button, one of the functions can be inserted into execution script, by clicking on the function name.


Neo Plugin: New Neo Plu
Secure | https://neotest6.crm.dynamics.com/main.aspx#325029733
Dynamics 365
Settings
Neo Plugins > New Neo Plugin
SAVE
SAVE & CLOSE
NEW
FLOWS
NEO PLUGIN : INFORMATION
New Neo Plugin
VERIFY SCRIPT
OPEN USER MANUAL
https://neotest6.crm.dynamics.com/%7B636648041840000197%7D/WebResources/neo_plugin/inser...
Secure | https://neotest6.crm.dynamics.com/%7B636648041840000197%7D/WebResources/neo_plugin/in...

General

*Name	Test Neo Plugin
Entity	Account (account)
Description	Verify account address

Execution Script

```
if (@address1_country == null) { th
if (@address1_city == null) { throw
verifyAddressRequest = "https://www
                       "&city=" + @
verifyAddressResponse = callwebserv
if (verifyAddressResponse != "OK")
```

Click on the function name to insert into execution script

today()

Flow Functions

- end(message)
- foreach(item in list){ statement(s) }
- if(condition){ statement(s) } else { statement(s) }
- trace(message)
- throw(message)

CRM Data/Entity Functions

- create(entityName, attributeName1, attributeValue1, attributeName2, attributeValue2)
- delete(entityName, recordId)
- getEntity(fetchXml)
- getEntityList(fetchXml)
- hasRole(roleName)
- lookup(entityName, attributeName1, attributeValue1, attributeName2, attributeValue2)
- runwf(workflowName, recordId, ignoreError)
- update(entityName, recordId, attributeName1, attributeValue1, attributeName2, attributeValue2)

CRM Plugin Functions

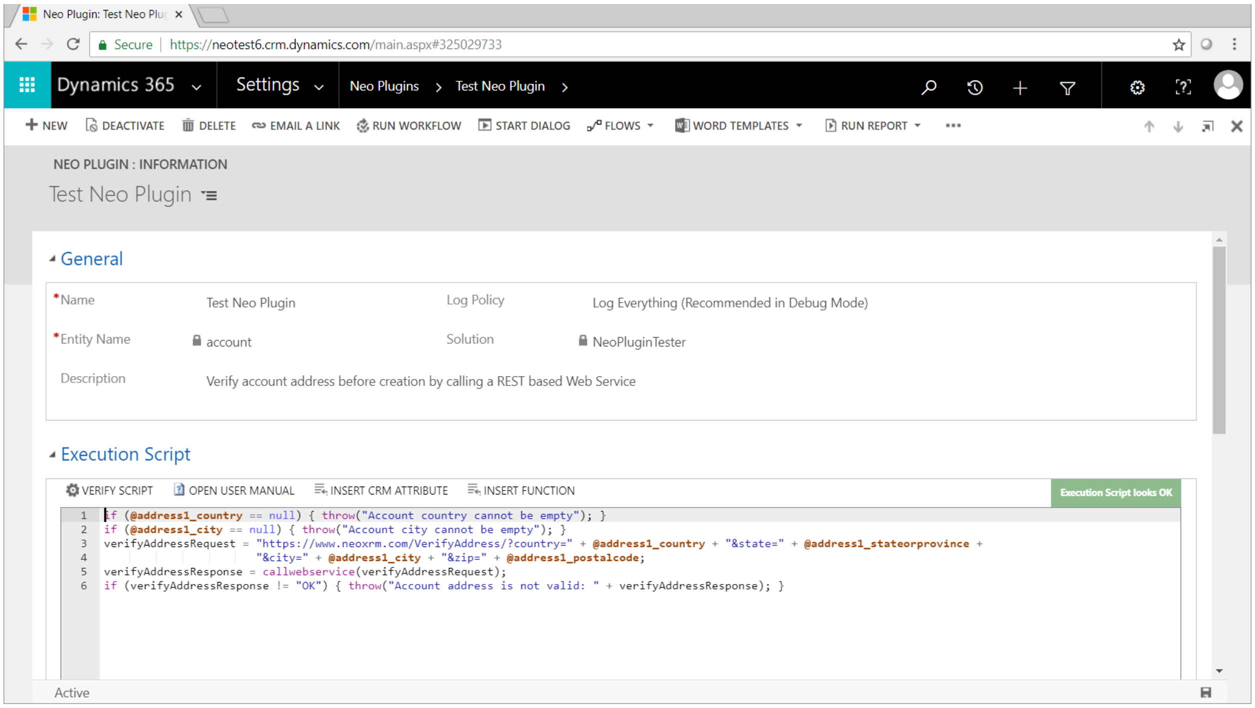
- getMessageName()
- getStepMode()
- getStepStage()
- getDepth()

Other Functions

- callwebservice(webserviceURL)

calls a REST based web service and returns its response as string

Clicking on the VERIFY SCRIPT button verifies the script for syntax and entity attribute names and shows the error.



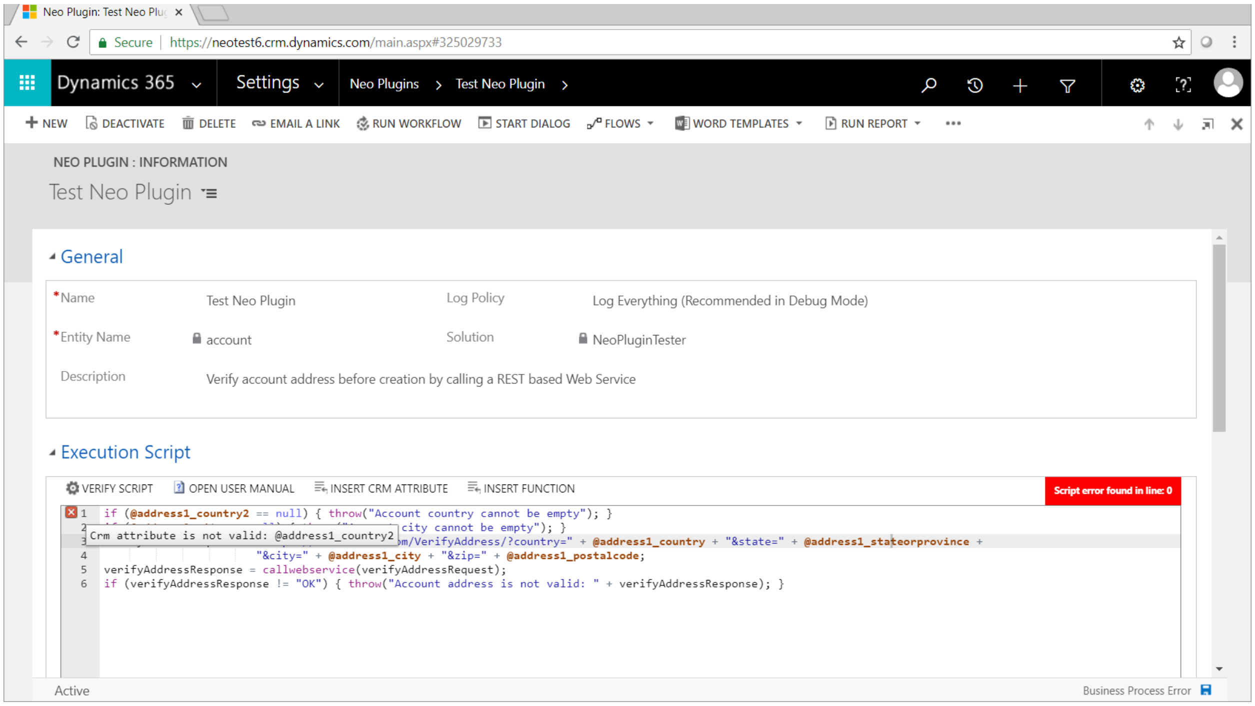
General

*Name	Test Neo Plugin	Log Policy	Log Everything (Recommended in Debug Mode)
*Entity Name	account	Solution	NeoPluginTester
Description	Verify account address before creation by calling a REST based Web Service		

Execution Script

Execution Script looks OK

```
if (@address1_country == null) { throw("Account country cannot be empty"); }
if (@address1_city == null) { throw("Account city cannot be empty"); }
verifyAddressRequest = "https://www.neoxrm.com/VerifyAddress/?country=" + @address1_country + "&state=" + @address1_stateorprovince +
                       "&city=" + @address1_city + "&zip=" + @address1_postalcode;
verifyAddressResponse = callwebservice(verifyAddressRequest);
if (verifyAddressResponse != "OK") { throw("Account address is not valid: " + verifyAddressResponse); }
```



NEO PLUGIN : INFORMATION

Test Neo Plugin

General

*Name	Test Neo Plugin	Log Policy	Log Everything (Recommended in Debug Mode)
*Entity Name	account	Solution	NeoPluginTester
Description	Verify account address before creation by calling a REST based Web Service		

Execution Script

Script error found in line: 0

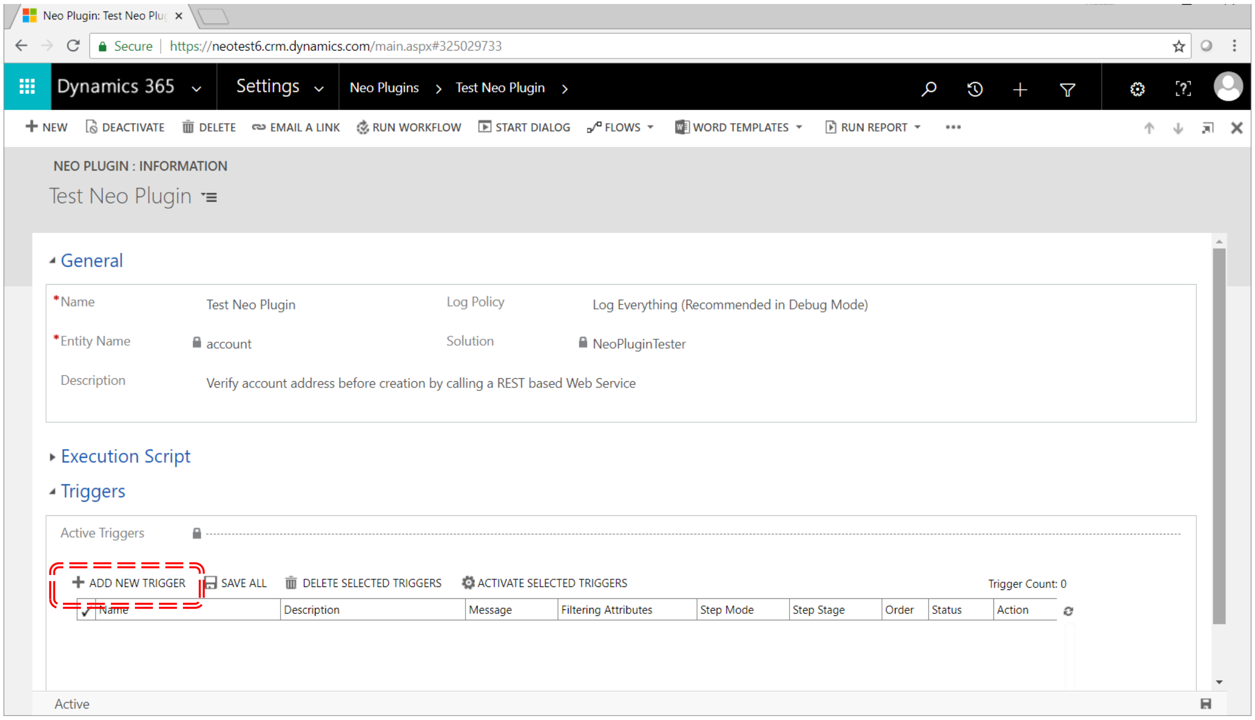
```
if (@address1_country2 == null) { throw("Account country cannot be empty"); }
                                          city cannot be empty"); }
                                          m/VerifyAddress/?country=" + @address1_country + "&state=" + @address1_stateorprovince +
                      "&city=" + @address1_city + "&zip=" + @address1_postalcode;
verifyAddressResponse = callwebservice(verifyAddressRequest);
if (verifyAddressResponse != "OK") { throw("Account address is not valid: " + verifyAddressResponse); }
```

Crm attribute is not valid: @address1_country2

Active

Business Process Error

Once the plugin record is created, Triggers section becomes visible to allow adding triggers to the plugin.



NEO PLUGIN : INFORMATION

Test Neo Plugin

General

*Name	Test Neo Plugin	Log Policy	Log Everything (Recommended in Debug Mode)
*Entity Name	account	Solution	NeoPluginTester
Description	Verify account address before creation by calling a REST based Web Service		

Execution Script

Triggers

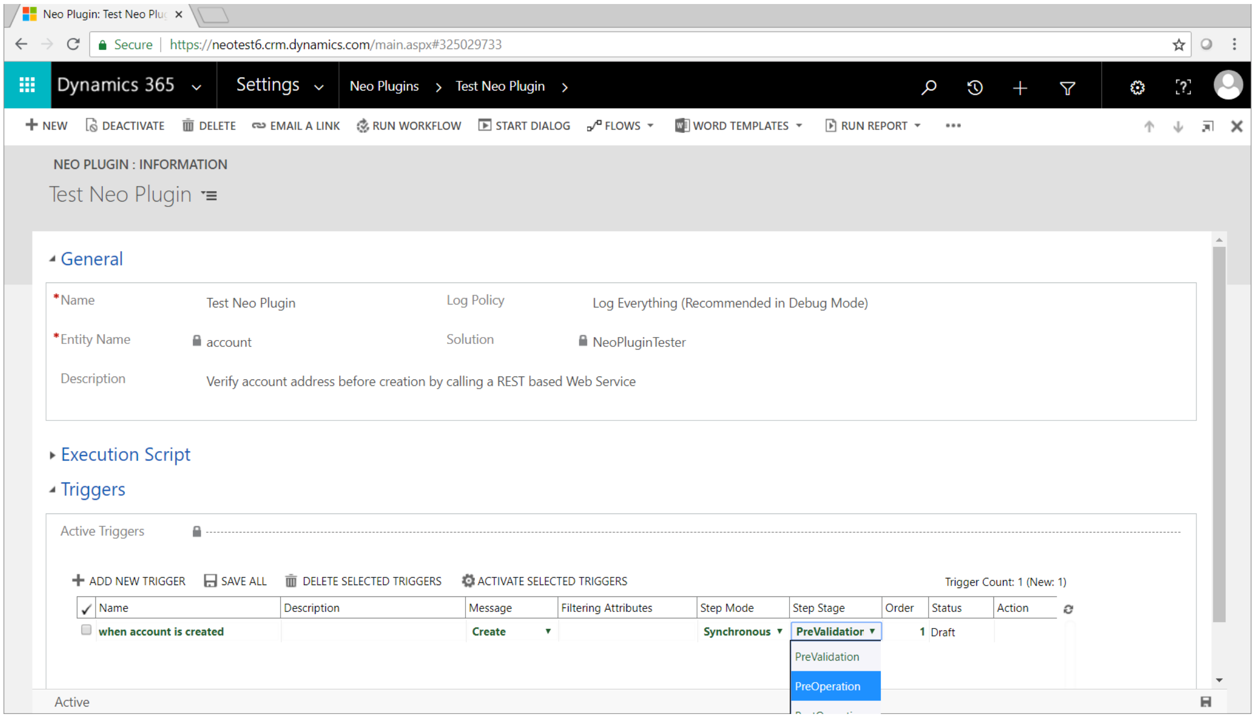
Active Triggers

Trigger Count: 0

Name	Description	Message	Filtering Attributes	Step Mode	Step Stage	Order	Status	Action

Active

Click on the “Add New Trigger” button. A new row will be added to grid. Provide information in the grid. Multiple triggers can be added for the plugin.



NEO PLUGIN : INFORMATION

Test Neo Plugin

General

*Name	Test Neo Plugin	Log Policy	Log Everything (Recommended in Debug Mode)
*Entity Name	account	Solution	NeoPluginTester
Description	Verify account address before creation by calling a REST based Web Service		

Execution Script

Triggers

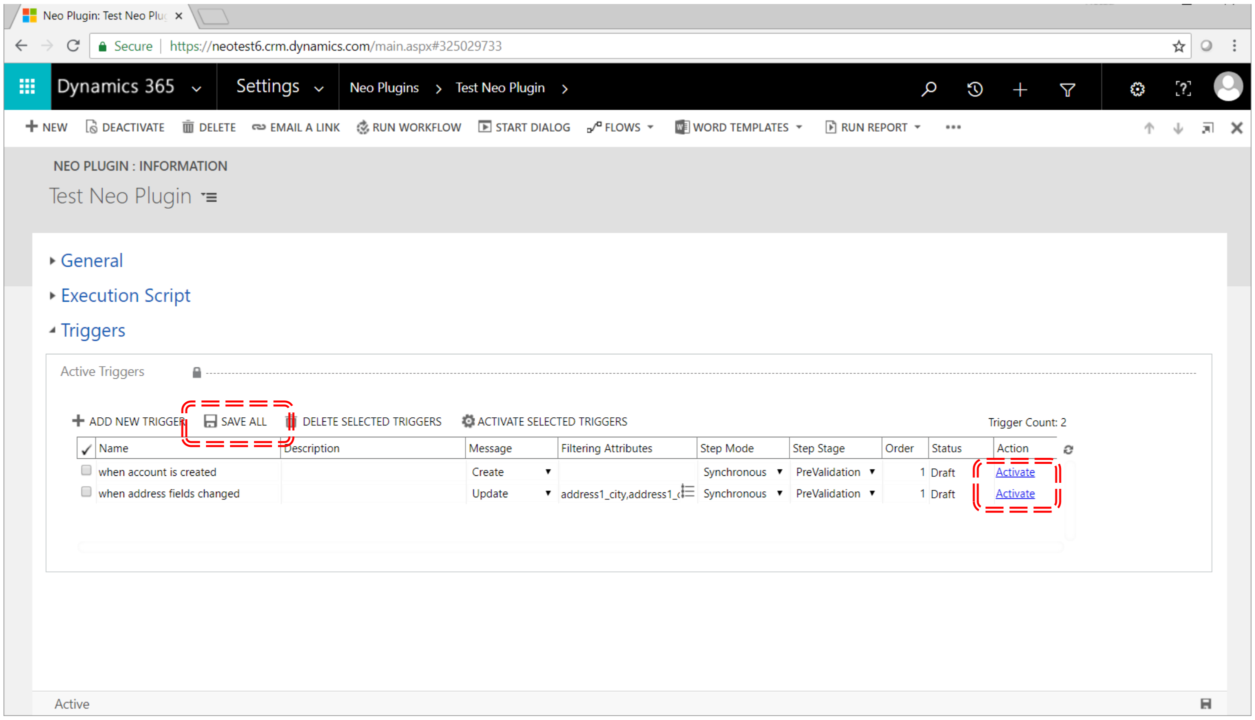
Active Triggers

Trigger Count: 1 (New: 1)

Name	Description	Message	Filtering Attributes	Step Mode	Step Stage	Order	Status	Action
□ when account is created		Create		Synchronous	PreValidation	1	Draft	

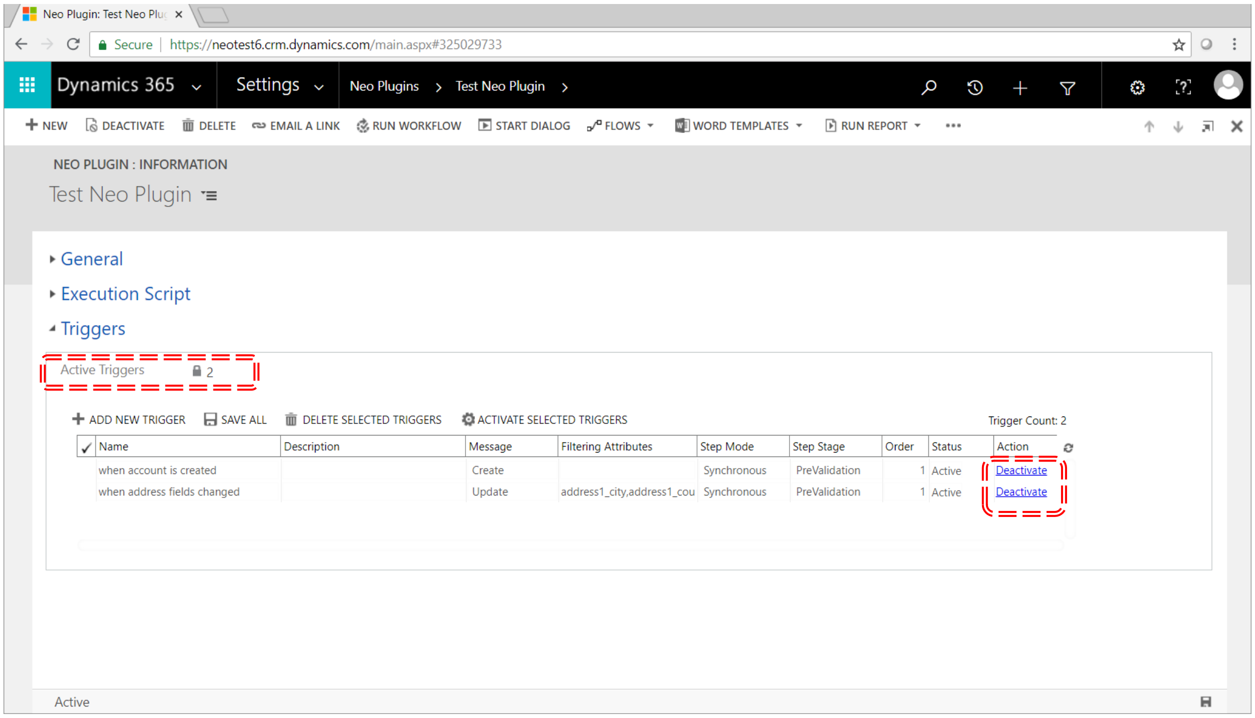
Active

Click on the Save All button.



✓	Name	Description	Message	Filtering Attributes	Step Mode	Step Stage	Order	Status	Action
□	when account is created		Create		Synchronous	PreValidation	1	Draft	Activate
□	when address fields changed		Update	address1_city,address1_c	Synchronous	PreValidation	1	Draft	Activate

Trigger will be created but not activated yet. Click on the Activate button to activate it. Once the trigger is activated, the execution script field becomes read only, meaning that to be able to edit execution scripts all active triggers should be deactivated.



✓	Name	Description	Message	Filtering Attributes	Step Mode	Step Stage	Order	Status	Action
	when account is created		Create		Synchronous	PreValidation	1	Active	Deactivate
	when address fields changed		Update	address1_city,address1_cou	Synchronous	PreValidation	1	Active	Deactivate

Once activated, a trigger cannot be edited. It needs to be deactivated to be able to make changes.

Now, you are done with your plugin creation and you can start testing it.

VI. Execution Script Syntax

Execution script is a list of statements separated with semi-column. Statements are executed in order of appearance. There are several types of statements.

1- CRM Input Reading: Reads the CRM attribute value and assign to a variable.

```
income = @annualincome;
```

This line reads the CRM attribute (`annualincome`) value and assigns to the variable `income`. Please note that **variable declaration is implicit**, meaning that first usage of a variable name on the left side of an equation declares a new variable and its type is determined automatically based on assigned values.

2- Variable manipulation:

```
credit=income*3.85;
```

This line declares a new variable with name `credit` and assign a value to it by multiplying another variable `income` with 3.85.
Besides four mathematical operations, built-in functions can also be used to manipulate variable values. See section VI for list of available functions.

```
lastUpdateDate = min(@modifiedon, @neo_activiydate);
```

It is also possible to use **conditional assignment** (if-then-else) to assign variable value based on the first satisfying condition.

```
customerSize =
if(@revenue>=1000000000) then "Large Enterprise"
if(@revenue>=100000000) then "Medium Enterprise"
if(@revenue>=10000000) then "Small Enterprise"
if(@revenue>=1000000) then "Medium-Sized Business"
else "Small Business";
```

3- Function execution: This type of statements do not change any variable value but executes some actions, for example running a CRM workflow. They can also be called from conditional expressions.

```
if (@revenue < 100000) {
    throw("Revenue cannot be less than $100,000");
}
```

This statement stops the plugin execution in some condition and shows CRM user an error message if registered with a synchronous trigger.

```
if (@revenue > 1000000000) {
    runwf("Email Large Enterprise Territory Exec WF", account);
}
```

This statement runs a CRM workflow for the given account record, if the condition satisfies.

4- CRM outputs: Updates CRM attribute values.

```
@creditlimit = credit;
```

The contact entity *creditlimit* attribute is updated with the value of *credit* variable. Please note that actual update of the attribute *does* happen at the very latest phase of the plugin execution. For this reason, even if you assign a value to the CRM attribute output, reading the same attribute value at later statement as CRM input will still return the initial value of the attribute.
Consider the below simple example.

```
@revenue = 100000;
trace("Revenue is " + @revenue);
```

Assume that the current value of the annual revenue of an account is $3,000,000. That is what we see in the execution log trace:

```
Input count: 1
CrmInput[Post]: revenue(Decimal)
Variable count: 0
CrmOutput count: 1
CrmOutput: revenue(Money)
Input [post] value: revenue = 3000000.0000
Executing statement: @revenue = 100000
Output value: revenue = 100000
Executing statement: trace("Revenue is " + @revenue)
Revenue is 3000000.0000
Executed method: neo.xrm.neoplugin.logic.function.TraceMethod
```

Notice that it prints the revenue as 3000000, the current value in the account record, not 10000, the next value to be assigned.

If you want to use the updated value in later stages of the execution, you need to assign it to a *variable*, instead of CRM attribute. CRM attribute assignments are used only for setting current entity attribute values which happens at the very latest stage of the execution, after all statements are executed.

5- Branching with **if-else** keywords: By using if conditions, multiple statements can be executed conditionally.

```
fetchXml = "…";

contactList = getEntityList(fetchXml);

if(contactList.Count > 0 && @primarycontactid == null)
{
   firstContact = contactList.First;
   @primarycontactid = firstContact.contactid;
}
else
{
    trace("no contact found");
}
```

6- Looping with **while** keyword

```
index = 0;
while(index < 10)
{
     trace("index = " + index) ;
     index = index + 1;
}
```

7- Looping with **foreach** keyword: Entity lists or string arrays can be looped over for each element in the list.

```
fetchXml =
  "<fetch version='1.0' output-format='xml-platform' mapping='logical'
distinct='false'>
    <entity name='contact'>
      <attribute name='contactid' />
      <filter type='and'>
         <condition attribute='parentcustomerid' operator='eq' value='" +
@parentcustomerid + "' />
      </filter>
      <order attribute='createdon' descending='true' />
    </entity>
  </fetch>";

contactList = getEntityList(fetchXml);

foreach(contact in contactList)
{
    update("contact", contact.contactid,
           "paymenttermscode", paymenttermscode);
}
```

```
nameArray = ["John", "Smith", "Jane", "Jennifer"];


foreach(name in nameArray)
{
    trace("Name of the person is " + name);
}
```

Looping over string array can also be achieved by using the while loop.

```
trace("The length of the name array is " + nameArray.Count);
trace("The first name in the array is " + nameArray.First);
trace("The last name in the array is " + nameArray.Last);


index = 0;
while(index < nameArray.Count)
{
   trace("A name from the array is " + nameArray.Item(index));
   index = index + 1;
}
```

VII. List of Build-In Functions

Math Functions

```
abs(number)
  returns absolute value of a given number

ave(number1, number2, …)
  returns average of given numbers

ceiling(number)
  returns the least integer greater than or equal to a given number

floor(number)
  returns the greatest integer less than or equal to a given number

log(number, base)
  returns mathematical logarithmic value of given number in given base.

max(number1, number2, …)
  returns the maximum of given numbers

min(number1, number2, …)
  returns the minimum of given numbers

parseDouble(string)
   returns double number value of a given string

parseInt(string)
   returns integer number value of a given string

pow(number, power)
  returns mathematical power value of a given number in given power.

round(number, decimals)
 rounds and returns the given number to a specified number of digits

sqrt(number)
  returns mathematical square-root value of a given number.
```

String Functions

endsWith(string, searchString)
returns true if a given string ends with a given search string.

htmlDecode(text)
returns html decoded version a given string

htmlEncode(text)
returns html encoded version a given string

newLine()
returns a platform specific new-line character

indexOf(string, searchString, [index])
returns the position of the first occurrence of a specified value in a string.

length(string)
returns the length of a given string

padLeft(string, padchar, length)
returns a new string that is padded with leading padChar characters so that its total length is *length* characters

padRight(string, padchar, length)
returns a new string that is padded with following padChar characters so that its total length is *length* characters

replace(string, oldText, newText)
returns a new string after replacing the old text with new one.

startsWith(text, search)
returns true if a given string starts with a given search string.

split(string, separator)
splits a string into an array of strings using the specified separator and returns the string array.

substring(string, start, length)
returns the substring of a given string starting at the *start* position with given length.

toLower(string)
returns all lower-case version of a given string

toUpper(string)
returns all upper-case version of a given string

trim(text)
returns a string with the leading and trailing white-spaces are removed from a given string.

XML Functions

getXmlAttribute(xml, xpath, attributeName, [index])
returns an attribute value in a given xml string, with given xpath.

```
responseXml = "<response resultCount='2'>
                  <company score='0.99'>
                      <id>80627426</id>
                      <name>Ozgencil Software LLC</name>
                      <phone>1-844-636-9761</phone>
                      <website>www.neoxrm.com</website>
                      <address>
                          <country>USA</country>
                          <state>NJ</state>
                          <city>Mahwah</city>
                          <postalcode>07430</postalcode>
                      </address>
                  </company>
                  <company score='0.12'>
                      <id>12345678</id>
                      <name>Ozgencil Carpentry LLC</name>
                      <phone>1-555-555-5555</phone>
                      <address>
                          <country>USA</country>
                          <state>NY</state>
                          <city>Syracuse</city>
                          <postalcode>13205</postalcode>
                      </address>
                  </company>
              </response>";
```

E.g.: getXmlAttribute(responseXml, "response/company", "score") return 0.99 and getXmlAttribute(responseXml, "response/company", "score", 1) returns 0.12

getXmlAttributeWithNS(xml, namespaceAlias, namespaceValue, xpath, attributeName, [index])
returns an attribute value in a given xml string, with given namespace and xpath.

```
responseXml = "<response resultCount='2'
xmlns='http://schemas.datacontract.org/2004/07/PropertyWS'>
                  <company score='0.99'>
                      <id>80627426</id>
                      <name>Ozgencil Software LLC</name>
                      <phone>1-844-636-9761</phone>
                      <website>www.neoxrm.com</website>
                      <address>
                          <country>USA</country>
                          <state>NJ</state>
                          <city>Mahwah</city>
                          <postalcode>07430</postalcode>
                      </address>
                  </company>
                  <company score='0.12'>
                      <id>12345678</id>
                      <name>Ozgencil Carpentry LLC</name>
                      <phone>1-555-555-5555</phone>
                      <address>
```

```
                                <country>USA</country>
                                <state>NY</state>
                                <city>Syracuse</city>
                                <postalcode>13205</postalcode>
                            </address>
                        </company>
                    </response>";
```

E.g.: getXmlAttributeWithNS(responseXml, "i", "http://schemas.datacontract.org/2004/07/PropertyWS", "/i:response//i:company", "score") return 0.99

getXmlElementCount(xml, xpath)

returns number of occurrences of a given xpath in a given xml string.

E.g.: getXmlElementCount(responseXml, "response/company") returns 2.

getXmlElementCountWithNS(xml, namespaceAlias, namespaceValue, xpath)

returns number of occurrences of an element searched with a given namespace and xpath.

E.g.: getXmlElementCountWithNS(responseXml, "i", "http://schemas.datacontract.org/2004/07/PropertyWS", "/i:response//i:company") returns 2.

getXmlElementInnerText(xml, xpath, [index])

returns inner text of an xml node in a given xml string searched with a given xpath. This function is generally used to get text of a node.

E.g: getXmlElementInnerText(responseXml,"response/company/name") returns "Ozgencil Software LLC" while getXmlElementInnerText(responseXml,"response/company/name", 1) returns "Ozgencil Carpentry LLC"

getXmlElementInnerTextWithNS(xml, namespaceAlias, namespaceValue, xpath, [index])

returns inner text of an xml node in a given xml string searched with a given xpath. This function is generally used to get text of a node.

E.g: getXmlElementInnerTextWithNS(responseXml, "i", "http://schemas.datacontract.org/2004/07/PropertyWS", "/i:response//i:company//i:name") returns "Ozgencil Software LLC" while getXmlElementInnerTextWithNS(responseXml, "i", "http://schemas.datacontract.org/2004/07/PropertyWS", "/i:response//i:company//i:name", 1) returns "Ozgencil Carpentry LLC"

getXmlElementOuterText(xml, xpath, [index])
getXmlElementOuterTextWithNS(xml, namespaceAlias, namespaceValue, xpath, [index])

returns outer text of an xml node in a given xml string searched with a given xpath. This function is generally used to get the whole node for further processing. E.g: getXmlElementOuterText(responseXml,"response/company/address") returns the whole address node:

```
<address>
    <country>USA</country>
    <state>NJ</state>
    <city>Mahwah</city>
    <postalcode>07430</postalcode>
</address>
```

Date/Time Functions

dateDiff(datePart, startDate, endDate)
returns the difference between two dates in terms of given date part ("year", "month", "week", "day", "hour", "minute", "second", "millisecond").
E.g.: dateDiff("hour", @createdon, utcNow()) returns total number of hours since the record was created.

getDatePart(datePart, date)
returns the date part ("year", "month", "week", "day", "hour", "minute", "second") of a given date in numbers. E.g.: getDatePart("year", utcToday()) returns 2019.

max(date1, date2, …)
returns maximum of given dates.

min(date1, date2, …)
returns minimum of given dates.

parseDate(string)
parses a given string and returns its date/time value.

utcNow()
returns the current time in Coordinated Universal Time (UTC) time zone.

utcToday()
returns the current date in Coordinated Universal Time (UTC) time zone.

Flow Functions

end(message)
ends the plugin processing and logs a given message to trace log.

foreach(item in list) { statement(s) }
loops over the given list (must be either entity list or string array) and executes given statements for each item in the list

if(condition) { statement(s) } else { statement(s) }
executes statements based on condition

throw(message)
ends the plugin processing by throwing a InvalidPluginExecution exception. If plugin trigger is synchronous, all database changes are reverted back. Notice that plugin execution record is not created in that case.

trace(message)
adds a message to trace log.

while(condition) { statement(s) }
executes given statements as long as the condition is true. Conditions is checked at the beginning and after each iteration.

CRM Data Retrieval & Manipulation Functions

```
addBCCRecipient(emailId, recipientRef)
addCCRecipient(emailId, recipientRef)
addTORecipient(emailId, recipientRef)
addToTeam(teamName, userRef)
associate(relationshipName, record1Ref, record2Ref)
attachTextFile(emailId, filename, textFileContentString)
attachBinaryFile(emailId, filename, binaryFileContent)
create(entityName, attributeName1, attributeValue1, …)
delete(entityName, recordId)
disassociate(relationshipName, record1Ref, record2Ref)
entityRef(entityName, recordId)
getEntity(fetchXml)
getEntityList(fetchXml)
hasRole(roleName, userId)
isMemberOfTeam(teamName, userId)
lookup(entityName, attributeName1, attributeValue1, …)
removeFromTeam(teamName, userId)
retrieve(entityName, recordId, attributeName1, attributeName2, …)
runwf(workflowName, recordId, ignoreError)
sendEmail(emailId, ignoreError)
setOwner(recordRef, ownerId)
setStateStatus(entityRef, statecode, statuscode)
update(entityName, recordId, attributeName1, attributeValue1, …)
```

CRM Plugin Functions

```
getCurrentRecord()
getCurrentUser()
getDepth()
getLatestError()
getMessageName()
getStepMode()
getStepStage()
```

Other Functions

`callWebService(webserviceURL)`

makes a GET based web call to a given URL and returns the response string. E.g.: `callWebService("https://www.neoxrm.com/account/time.aspx")` returns a string like "2019-01-01 18:11:30 UTC"

`callWebServiceForBinary(webserviceURL)`

makes a GET based web call to a given URL and returns the response as binary (byte array). E.g.: `pdfContent = callWebService("https://www.neoxrm.com/pdf/NeoPluginUserManual.pdf")` returns a binary of PDF file. This binary can be attached to an email, with a command like: `attachBinaryFile(emailId, "usermanual.pdf", pdfContent)`

VIII. Example Use Cases

Below use-case examples will help you better understand the capabilities of Neo Plugin, what you can do with it and what you cannot do with it. We will start with easy examples and then continue to demonstrate examples that are more complex.

Use Case #1

Entity	Contact
Task	Set the credit limit (money), based on annual income (money) with factor 3.85.
Trigger(s)	1) Synchronous, Post-Operation, On Create 2) Synchronous, Post-Operation, On Update (of annual income)

Execution Script	`income=@annualincome;`
	`credit=income*3.85;`
	`@creditlimit=credit;`

Remarks	

Use Case #2

Entity	Account
Task	Set customer size (optionset), based on revenue (money)
Trigger(s)	1) Synchronous, Post-Operation, On Create 2) Synchronous, Pre-Validation, On Update (of revenue)

Execution Script	`revenue=@revenue;`
	`customerSize=`
	`if(revenue>=1000000000) then "Large Enterprise"`
	`if(revenue>=100000000) then "Medium Enterprise"`
	`if(revenue>=10000000) then "Small Enterprise"`
	`if(revenue>=1000000) then "Medium-Sized Business"`
	`else "Small Business";`
	`@customersizecode=customerSize;`

Remarks	In this example, optionset type attribute is set based on optionset **label**. This is especially useful when editor does not know actual optionset value for each label. It is also possible to use **integer value** of option instead of its label. Neo Plugin is smart enough to determine if assigned value is integer or text and handles both input cases accordingly.

Use Case #3

Entity	Opportunity Product
Task	Determine tax rate based on state/province code (text) of account of opportunity, and set the new total amount (money) by adding the tax to the extended amount (money). Exclude some products from the tax. (PS has no tax)
Trigger(s)	1) Synchronous, Post-Operation, On Update (baseamount,manualdiscountamount)

Execution Script	`stateCode=@opportunityid.parentaccountid.address1_stateorprovince;`
	`productCode=@productid.productnumber;`
	`base=@baseamount;`
	`discount=@manualdiscountamount;`
	`amount=round(base-discount);`
	`taxRate=`
	`if(stateCode="NJ") then 0.07`
	`if(stateCode="NY") then 0.0825`
	`if(stateCode="PA") then 0.05`
	`if(stateCode="CT") then 0.095`
	`if(stateCode="VR") then 0.065`
	`else 0;`
	`taxRate = if(productCode="PS") then 0 else taxRate;`
	`taxAmount=amount*taxRate;`
	`@tax=taxAmount;`

Remarks	This example demonstrates reading referenced entity values from CRM. We have followed the chain from opportunity product to opportunity, and then account, and read the state or province text field value of account, with a single line: `stateCode=@opportunityid.parentaccountid.address1_stateorprovince`

Use Case #4

Entity	Account
Task	Assume you have a custom entity called Country. In account entity, you use a lookup Country field (neo_countryid), instead of OOB address1_country text field, to ensure data accuracy. You also have a third party solution, let's say D&B integrator, which imports the accounts from D&B, but it sets the OOB address1_country field, instead of your custom country lookup field. The task is to set the country lookup field (neo_countryid), based on OOB country field (address1_country), when account is created or country field updated.
Trigger(s)	1) Synchronous, Post-Operation, On Create 2) Synchronous, Post-Operation, On Update (address1_country)

Execution Script	`if(@address1_country == $address1_country) {`
	`   end("country not really changed");`
	`}`
	`if (@address1_country == @neo_countryid.neo_name) {`
	`   end("countries are same already");`
	`}`
	
	`// search for an active country by name`
	`country = lookup("neo_country","neo_name", @address1_country,`
	`                               "statecode", 0);`
	
	`// update country lookup field, if country record was found`
	`if (country != null)`
	`    @neo_countryid = country;`

Remarks	This example demonstrates the usage of the `lookup` function, which returns an entity reference for the provided entity and criteria. It is possible to pass multiple criteria connected with AND operation. The usage of the `end` operation is also demonstrated here: the plugin ends quietly, but leaves a debug message to Plugin Execution Log record, when “Do Not Log (if successful)” field is assigned No.

Use Case #5

Entity	Account
Task	This is opposite of task in use case #4. Assume that you need to synchronize the neo_countryid with address1_country. Update the address1_country, whenever account is created or neo_countryid is changed. Consider the use case #4 and prevent circular triggering of plugins #4 and #5 infinitely (would cause a plugin error).
Trigger(s)	1) Synchronous, Post-Operation, On Create 2) Synchronous, Post-Operation, On Update (address1_country)

Execution Script	`countryLookupOld = $neo_countryid.neo_name;`
	`countryLookupNew = @neo_countryid.neo_name;`
	
	`// check if lookup country has value`
	`if(countryLookupNew == null) { end(); }`
	
	`// check if lookup country value really changed`
	`if(countryLookupOld == countryLookupNew) { end(); }`
	`countryText = @address1_country;`
	
	`// check if lookup country is different than country text`
	`if(countryLookupNew == countryText) { end(); }`
	`@address1_country = countryLookupNew;`

Execution Script (shorter version)	`// check if lookup country has value`
	`// check if lookup country has really changed`
	`// check if lookup country is different than the country text`
	`if(@neo_countryid != null &&`
	`   @neo_countryid.neo_name != null &&`
	`   @neo_countryid != $neo_countryid &&`
	`   @neo_countryid.neo_name != @address1_country)`
	`   @address1_country = @neo_countryid.neo_name;`

Remarks	

Use Case #6

Entity	Case (incident)
Task	Set the number of hours passed since case is created, when the case is closed.
Trigger(s)	1) Synchronous, Pre-Operation, On SetStateDynamicEntity

Execution Script	`caseState=@statecode;`
	`if (caseState!=0) { end(); }`
	`createdOn=@createdon;`
	`numberOfHours=round(dateDiff("hour", createdOn, now()), 1);`
	`@neo_numberofhourspassed=numberOfHours;`

Remarks	datediff function returns difference between two given dates in given terms.

Use Case #7

Entity	Account
Task	Alert territory manager when credit limit is too high (based to revenue)
Trigger(s)	1) Synchronous, Post-Operation, On Update (creditlimit,revenue)

Execution Script	`accountId = @accountid;`
	`revenue = @revenue;`
	`creditLimit = @creditlimit;`
	`if (creditLimit > revenue * 2) {`
	`    runwf("Alert Territory Manager for credit", accountId, false);`
	`}`

Remarks	The `runwf` function runs a CRM workflow in the background for the given record. An active WF with given name should exist in CRM.

Use Case #8

Entity	Contact
Task	When a contact creditonhold field is set to Yes, we want to set its account creditonhold field to Yes, as well.
Trigger(s)	1) Asynchronous, Post-Operation, On Create 2) Asynchronous, Post-Operation, On Update (creditonhold)

Execution Script	`if (@parentcustomerid != null && @creditonhold == true) {`
	`    update("account", @parentcustomerid, "creditonhold", true);`
	`}`

Remarks	The `update` function updates one or more attributes of an entity record in CRM.

Use Case #9

<table>
<tr><td>Entity</td><td>Account</td></tr>
<tr><td>Task</td><td>Call a web service to lookup account information in D&B. The returned text is an XML. Parse the XML and use the D&B corrected account name, address, website and phone, if the service returns a company record with high score</td></tr>
<tr><td>Trigger(s)</td><td>1) Synchronous, Pre-Validation, On Create
2) Synchronous, Pre- Validation, On Update (name)</td></tr>
<tr><td>Execution Script</td><td><pre>
request = "https://www.neoxrm.com/account/DNBLookup.aspx?"+
                        "name=" + htmlEncode(@name) +
                        "&country=" + htmlEncode(@address1_country) +
                        "&state=" + htmlEncode(@address1_stateorprovince) +
                        "&city=" + htmlEncode(@address1_city);
responseXml = callwebservice(request);
resultCount = GetXmlElementCount(responseXml, "response/company");
if (resultCount == 1) {
  score = parseDouble(GetXmlAttribute(responseXml,
                               "response/company", "score"));
  if (score > 0.90) {
    @name = GetXmlElementInnerText(responseXml,
                               "response/company/name");
    @telephone1 = GetXmlElementInnerText(responseXml,
                               "response/company/phone");
    @websiteurl = GetXmlElementInnerText(responseXml,
                               "response/company/website");
    @address1_city = GetXmlElementInnerText(responseXml,
                               "response/company/address/city");
    @address1_country = GetXmlElementInnerText(responseXml,
                               "response/company/address/country");
    @address1_postalcode = GetXmlElementInnerText(responseXml,
                               "response/company/address/postalcode");
    @address1_stateorprovince = GetXmlElementInnerText(responseXml,
                               "response/company/address/state");
  }
}
</pre></td></tr>
<tr><td>Remarks</td><td>The callwebservice function calls the web service in GET mode and returns the string returned from web service as response. Build-in XML functions are used to extract data from XML very easily. This is an example response XML used in this use case:<pre>
    <response resultCount='1'>
       <company score='0.99'>
           <id>80627426</id>
           <name>Ozgencil Software LLC</name>
           <phone>1-844-6369761</phone>
           <website>www.neoxrm.com</website>
           <address>
              <country>USA</country>
              <state>NJ</state>
              <city>Mahwah</city>
              <postalcode>07430</postalcode>
           </address>
       </company>
    </response>
</pre></td></tr>
</table>

Use Case #10

Entity	Opportunity
Task	When opportunity won, create a SalesOrder record out of it. Also create salesorderdetail records from opportunityproduct list.
Trigger(s)	1) Synchronous, Post- Operation, SetStateDynamicEntity

Execution Script

```
if (@statecode != 1) end(); // 1: won.
//if not won, just end the plugin. If won, continue to create sales order

orderId = create("salesorder",
                 "name", @name,
                 "description", @description,
                 "customerid", @customerid,
                 "pricelevelid", @pricelevelid,
                 "transactioncurrencyid", @transactioncurrencyid,
                 "opportunityid", entityRef("opportunity", @opportunityid)
                 // add more attribute mappings here in format:
                 //"salesorder_attribute", @opportunity_attribute
          );

productLineList = getEntityList("<fetch distinct='false' mapping='logical'
output-format='xmlplatform' version='1.0'>
  <entity name='opportunityproduct'>
    <attribute name='extendedamount'/>
    <attribute name='productid'/>
    <attribute name='quantity'/>
    <attribute name='lineitemnumber'/>
    <attribute name='priceperunit'/>
    <attribute name='tax'/>
    <attribute name='transactioncurrencyid'/>
    <attribute name='uomid'/>
    <filter type='and'>
      <condition attribute='opportunityid' operator='eq' value='" +
@opportunityid + "'/>
    </filter>
  </entity>
</fetch>");

foreach(productLine in productLineList) {
    create("salesorderdetail",
           "salesorderid", orderId,
           "quantity", productLine.quantity,
           "productid", productLine.productid,
           "lineitemnumber", productLine.lineitemnumber,
           "priceperunit", productLine.priceperunit,
           "tax", productLine.tax,
           "transactioncurrencyid", productLine.transactioncurrencyid,
           "uomid", productLine.uomid
           // add more mappings here in format:
           //"salesorderdetail_attribute",
productLine.opportunityproduct_attribute
           // be sure to add opportunity product attribute to above fetchXml
    );
}
```

Use Case #11

<table>
<tr><td>Entity</td><td>Opportunity Line</td></tr>
<tr><td>Task</td><td>Set the total of parts and labor in the opportunity, whenever new line is added or an existing line is updated</td></tr>
<tr><td>Trigger(s)</td><td>1) Synchronous, Post-Operation, On Create
2) Synchronous, Post- Operation, On Update (productid, extendedamount)</td></tr>
<tr><td>Execution Script</td><td>

```
fetchXml = "<fetch distinct='false' mapping='logical' output-format='xml-platform' version='1.0'>
<entity name='opportunityproduct'>
    <attribute name='extendedamount'/>
    <filter type='and'>
        <condition attribute='opportunityid' operator='eq' value='" + @opportunityid + "'/>
    </filter>
    <link-entity name='product' alias='pr' link-type='outer' to='productid' from='productid'>
        <attribute name='producttypecode'/>
    </link-entity>
</entity></fetch>";
productLineList = getEntityList(fetchXml);
laborTotal = 0;
partsTotal = 0;
foreach(productLine in productLineList) {
    if (productLine.pr.producttypecode == 1) {    // parts
        partsTotal = partsTotal + productLine.extendedamount;
    }
    if (productLine.pr.producttypecode == 3) {    // labor
        laborTotal = laborTotal + productLine.extendedamount;
    }
}
if (@opportunityid.neo_labortotal != laborTotal) {
    update("opportunity", @opportunityid, "neo_labortotal", laborTotal);
}
if (@opportunityid.neo_partstotal != partsTotal) {
    update("opportunity", @opportunityid, "neo_partstotal", partsTotal);
}
```

</td></tr>
<tr><td>Remarks</td><td>Construct the fetchXml and call the `getEntityList` function. Loop over the returned entity list using the `foreach` command. Notice how attributes of the entity record and linked-entity record are accessed.</td></tr>
</table>

Use Case #12

Entity	Lead
Task	Promote a Lead to an Opportunity: Search for contact and account in CRM. If they do not exist, create new account and contact. Create a new opportunity from the lead. Add an opportunity product. Close the lead as Qualified.
Trigger(s)	1) Synchronous, Post- Operation, On Update (neo_promote)
Execution Script	

```
if (@neo_promote != true) end();

contactId = null;
accountId = null;
newContact = false;
if (@contactid != null)
{
    contactId = @contactid;
    if (@contactid.accountid != null)
    {
        accountId = @contactid.accountid;
    }
}
else
{
    contactId = lookup("contact", "emailaddress1", @emailaddress1);
    if (contactId != null)
    {
        trace("contact found with email: " + @emailaddress1);
        contact = retrieve("contact", contactId, "accountid");
        if (contact != null && contact.accountid != null)
        {
            accountId = contact.accountid;
        }
    }
    else
    {
        contactId = create("contact",
               "firstname", @firstname,
               "lastname", @lastname,
               "emailaddress1", @emailaddress1,
               "address1_city", @address1_city,
               "address1_stateorprovince", @address1_stateorprovince,
               "address1_postalcode", @address1_postalcode,
               "address1_country", @address1_country);
        if (contactId != null)
        {
            trace("new contact was created");
            newContact = true;
        }
        else
        {
            trace("new contact cannot be created");
        }
    }
}

if (accountId == null)
{
    if (@parentaccountid != null)
    {
        accountId = @parentaccountid;
    }
    else
    {
        accountId = lookup("account",
                  "name", @companyname,
                  "address1_city", @address1_city);
        if (accountId != null)
        {
            trace("account found with name and city");
        }
        else
        {
            accountId = create("account",
               "name", @companyname,
               "address1_city", @address1_city,
               "address1_stateorprovince", @address1_stateorprovince,
               "address1_postalcode", @address1_postalcode,
               "address1_country", @address1_country);
            if (accountId != null)
            {
                trace("new account was created");
            }
            else
            {
                trace("new account cannot be created");
            }
        }
    }
    if (accountId != null && newContact == true)
    {
        trace("set account of newly created contact1");
        update("contact", contactId, "parentcustomerid", accountId);
        trace("set account of newly created contact2");
    }
}

productId = lookup("product", "name", "Audi A4 Quattro");
if (productId == null)
{
    throw("product not found");
}
product = retrieve("product", productId, "defaultuomid", "pricelevelid");
if (product == null)
{
    throw("product not found2");
}

opportunityId = create("opportunity",
               "customerid", accountId,
               "parentaccountid", accountId,
               "parentcontactid", contactId,
               "pricelevelid", product.pricelevelid,
               "originatingleadid", getCurrentRecord(),
               "name", @subject);
if (opportunityId == null)
    throw("opportunity cannot be created");

setOwner(opportunityId, @ownerid); // assign opportunity owner from lead


opportunityProductId = create("opportunityproduct",
      "opportunityid", opportunityId,
      "productid", productId,
      "uomid", product.defaultuomid,
      "quantity", 1.0);

update("lead", getCurrentRecord(),
       "qualifyingopportunityid", opportunityId);

setStateStatus(getCurrentRecord(), 1, 3); // close lead as qualified
```

Use Case #13

Entity	Lead
Task	Send an email to lead when annual revenue is changed
Trigger(s)	1) Synchronous, Post- Operation, On Update (revenue)

Execution Script	`to = null;`
	`if (@parentcontactid != null)`
	`{`
	`    to = @parentcontactid;`
	`}`
	`else`
	`{`
	`    if (@emailaddress1 != null)`
	`    {`
	`        to = @emailaddress1;`
	`    }`
	`}`
	`emailId = create("email",`
	`       "to", to,`
	`       "from", getCurrentUser(),`
	`       "subject", "This is a test email",`
	`       "regardingobjectid", getCurrentRecord(),`
	`       "description", "<img src='https://www.neoxrm.com/img/logo.png'`
	`/><br/><div style='color: #001276; font-size:16px;font-weight:bold;'>`
	` New Annual Revenue is "+ @transactioncurrencyid.currencysymbol +`
	` @revenue + "</div>");`
	
	`addCCRecipient(emailId, "support@neoxrm.com");`
	`sendEmail(emailId);`

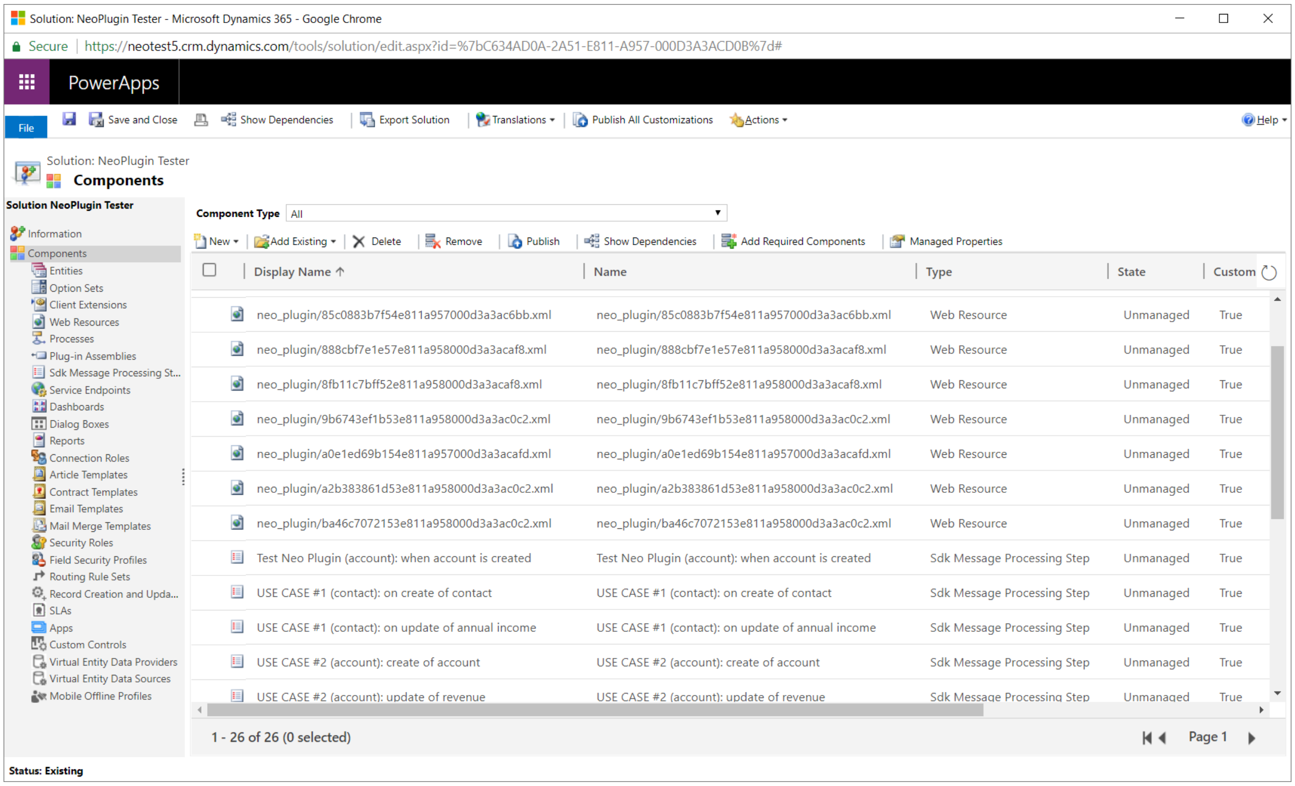
Remarks	Use the `create` function to create an email in Draft mode. Notice that email body can be Html content constructed dynamically. Add extra TO, CC or BCC recipients. The `sendEmail` function actually sends the email created in Draft mode.

IX. Export and Import Neo Plugins

It is always a good practice to have separate development and staging environments, and to make customizations in development environment, rather than directly making in production environment. NeoPlugin provides you two different approaches to support this practice.

When you create a new Plugin, you need to select a solution name. After you create the plugin and publish its triggers, the solution you have selected will be a container for required components to be able to move your Neo Plugins from one CRM environment to another as a ***runnable only plugin***. When you export this solution from your development environment, and then import into your production environment, all Neo Plugins created under this solution will automatically work in the production environment.

The example NeoPluginTester solution in below contains all required components to be able to **run** plugins.



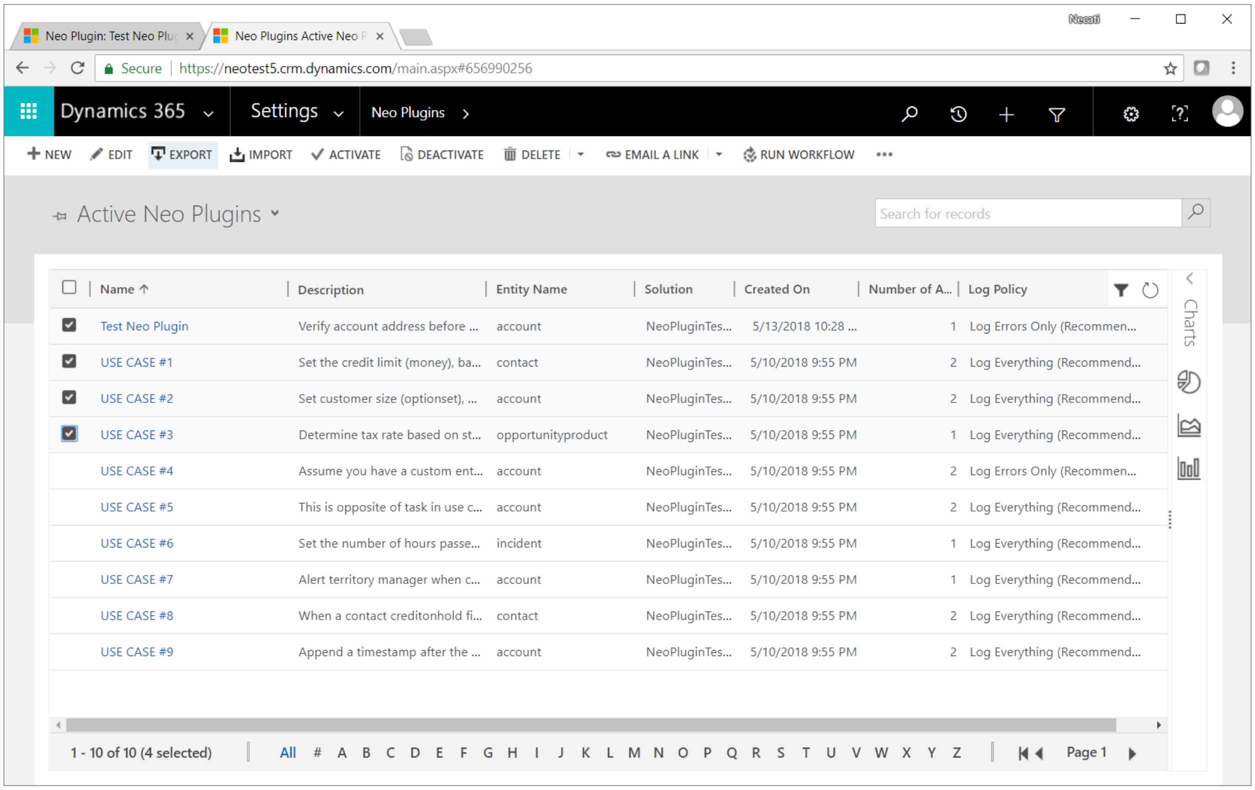
Display Name	Name	Type	State	Custom
neo_plugin/85c0883b7f54e811a957000d3a3ac6bb.xml	neo_plugin/85c0883b7f54e811a957000d3a3ac6bb.xml	Web Resource	Unmanaged	True
neo_plugin/888cbf7e1e57e811a958000d3a3acaf8.xml	neo_plugin/888cbf7e1e57e811a958000d3a3acaf8.xml	Web Resource	Unmanaged	True
neo_plugin/8fb11c7bff52e811a958000d3a3acaf8.xml	neo_plugin/8fb11c7bff52e811a958000d3a3acaf8.xml	Web Resource	Unmanaged	True
neo_plugin/9b6743ef1b53e811a958000d3a3ac0c2.xml	neo_plugin/9b6743ef1b53e811a958000d3a3ac0c2.xml	Web Resource	Unmanaged	True
neo_plugin/a0e1ed69b154e811a957000d3a3acafd.xml	neo_plugin/a0e1ed69b154e811a957000d3a3acafd.xml	Web Resource	Unmanaged	True
neo_plugin/a2b383861d53e811a958000d3a3ac0c2.xml	neo_plugin/a2b383861d53e811a958000d3a3ac0c2.xml	Web Resource	Unmanaged	True
neo_plugin/ba46c7072153e811a958000d3a3ac0c2.xml	neo_plugin/ba46c7072153e811a958000d3a3ac0c2.xml	Web Resource	Unmanaged	True
Test Neo Plugin (account): when account is created	Test Neo Plugin (account): when account is created	Sdk Message Processing Step	Unmanaged	True
USE CASE #1 (contact): on create of contact	USE CASE #1 (contact): on create of contact	Sdk Message Processing Step	Unmanaged	True
USE CASE #1 (contact): on update of annual income	USE CASE #1 (contact): on update of annual income	Sdk Message Processing Step	Unmanaged	True
USE CASE #2 (account): create of account	USE CASE #2 (account): create of account	Sdk Message Processing Step	Unmanaged	True
USE CASE #2 (account): update of revenue	USE CASE #2 (account): update of revenue	Sdk Message Processing Step	Unmanaged	True

1 - 26 of 26 (0 selected)

Of course, you need to have a Neo Plugin tool solution installed with a separate license in any environment you want to work on.

Please note that you cannot directly modify Neo Plugins in production environment, as you do not have corresponding Neo Plugin records in the production. They exist and run as regular CRM plugins, but they do not exist as “Neo Plugin”. If you go to Neo Plugins page, you will see no record.

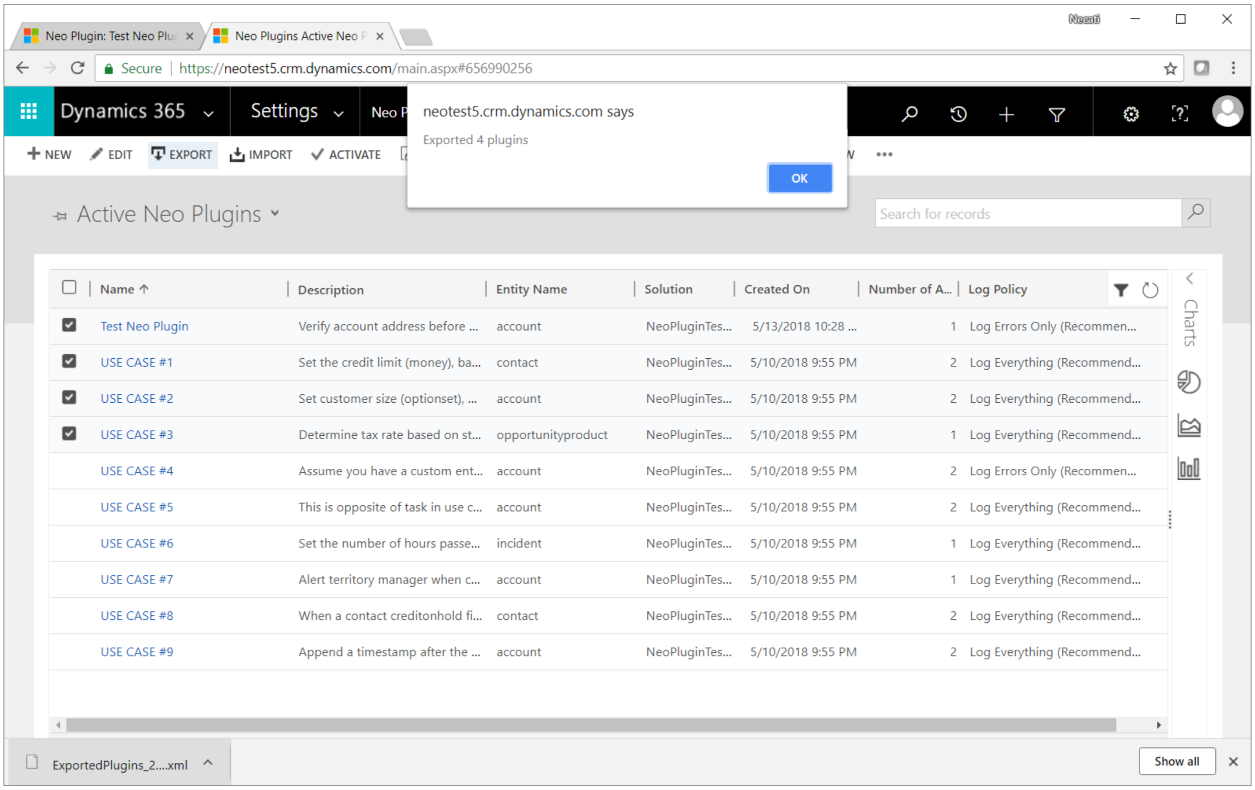
If you want to modify plugins directly in your production environment, you need to import them by using the export/import feature provided in the Neo Plugins page as seen below. Assume that this is our development environment that we want to export from.



	Name ↑	Description	Entity Name	Solution	Created On	Number of A...	Log Policy
☑	Test Neo Plugin	Verify account address before ...	account	NeoPluginTes...	5/13/2018 10:28 ...	1	Log Errors Only (Recommen...
☑	USE CASE #1	Set the credit limit (money), ba...	contact	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...
☑	USE CASE #2	Set customer size (optionset), ...	account	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...
☑	USE CASE #3	Determine tax rate based on st...	opportunityproduct	NeoPluginTes...	5/10/2018 9:55 PM	1	Log Everything (Recommend...
	USE CASE #4	Assume you have a custom ent...	account	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Errors Only (Recommen...
	USE CASE #5	This is opposite of task in use c...	account	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...
	USE CASE #6	Set the number of hours passe...	incident	NeoPluginTes...	5/10/2018 9:55 PM	1	Log Everything (Recommend...
	USE CASE #7	Alert territory manager when c...	account	NeoPluginTes...	5/10/2018 9:55 PM	1	Log Everything (Recommend...
	USE CASE #8	When a contact creditonhold fi...	contact	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...
	USE CASE #9	Append a timestamp after the ...	account	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...

1 - 10 of 10 (4 selected) All # A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Page 1

Select the Neo Plugins you want to export, and click on the Export button on the menu. An xml file will be downloaded into your machine.

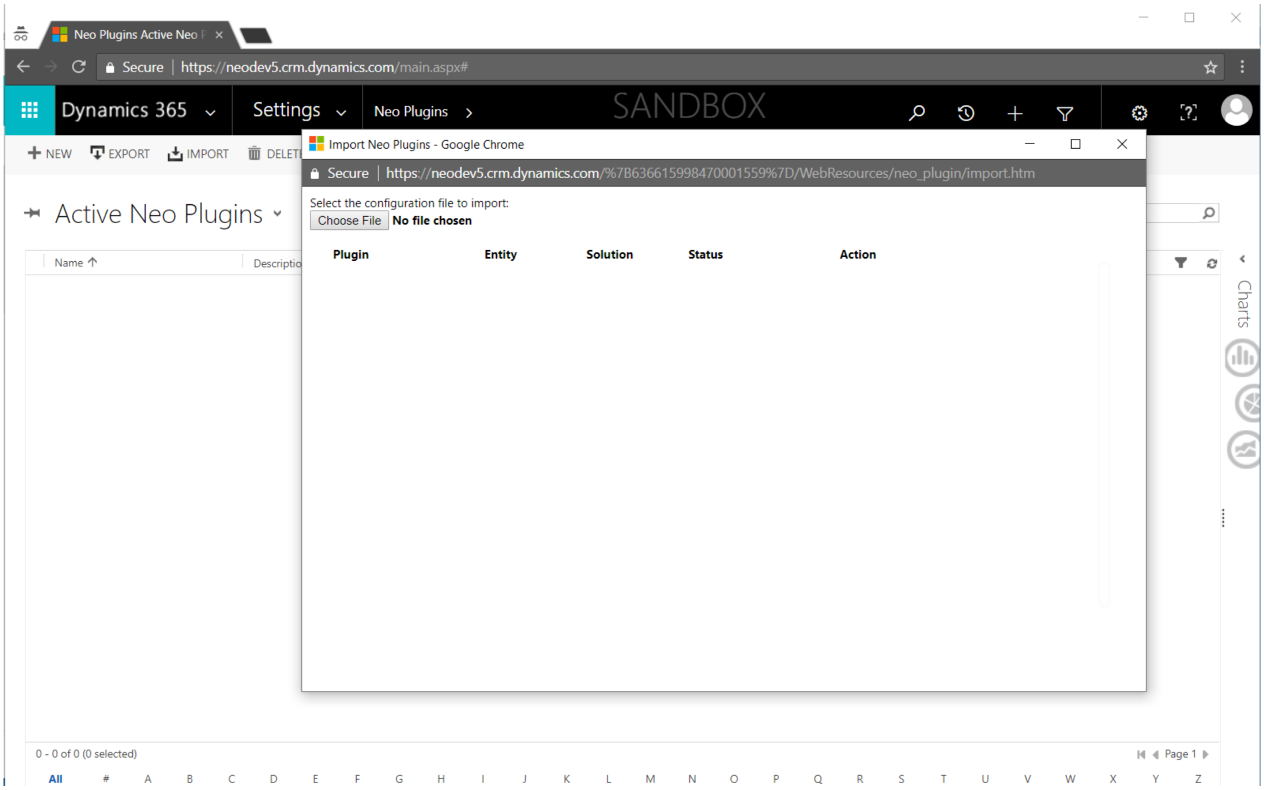


neotest5.crm.dynamics.com says

Exported 4 plugins

☐	Name	Description	Entity Name	Solution	Created On	Number of A...	Log Policy
☑	Test Neo Plugin	Verify account address before ...	account	NeoPluginTes...	5/13/2018 10:28 ...	1	Log Errors Only (Recommen...
☑	USE CASE #1	Set the credit limit (money), ba...	contact	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...
☑	USE CASE #2	Set customer size (optionset), ...	account	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...
☑	USE CASE #3	Determine tax rate based on st...	opportunityproduct	NeoPluginTes...	5/10/2018 9:55 PM	1	Log Everything (Recommend...
	USE CASE #4	Assume you have a custom ent...	account	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Errors Only (Recommen...
	USE CASE #5	This is opposite of task in use c...	account	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...
	USE CASE #6	Set the number of hours passe...	incident	NeoPluginTes...	5/10/2018 9:55 PM	1	Log Everything (Recommend...
	USE CASE #7	Alert territory manager when c...	account	NeoPluginTes...	5/10/2018 9:55 PM	1	Log Everything (Recommend...
	USE CASE #8	When a contact creditonhold fi...	contact	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...
	USE CASE #9	Append a timestamp after the ...	account	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...

Go to another Dynamics 365 environment and click on the Import button on the Neo Plugins page. A small window will open.



Import Neo Plugins - Google Chrome

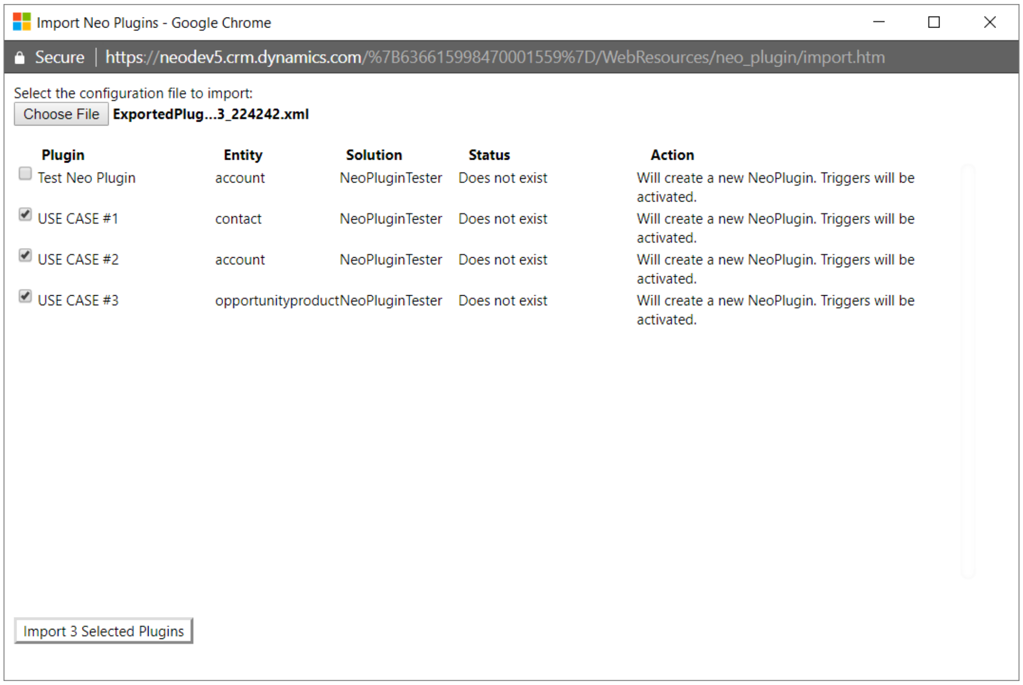
https://neodev5.crm.dynamics.com/%7B636615998470001559%7D/WebResources/neo_plugin/import.htm

Select the configuration file to import:

Choose File No file chosen

Plugin	Entity	Solution	Status	Action

Click on the Choose File button and select the file you have just exported. Status for each row will say “Exists As NeoPlugin”, “Exists As Plugin” or “Does not exist”. By importing them, corresponding Neo Plugin records of the existing plugins will be created.



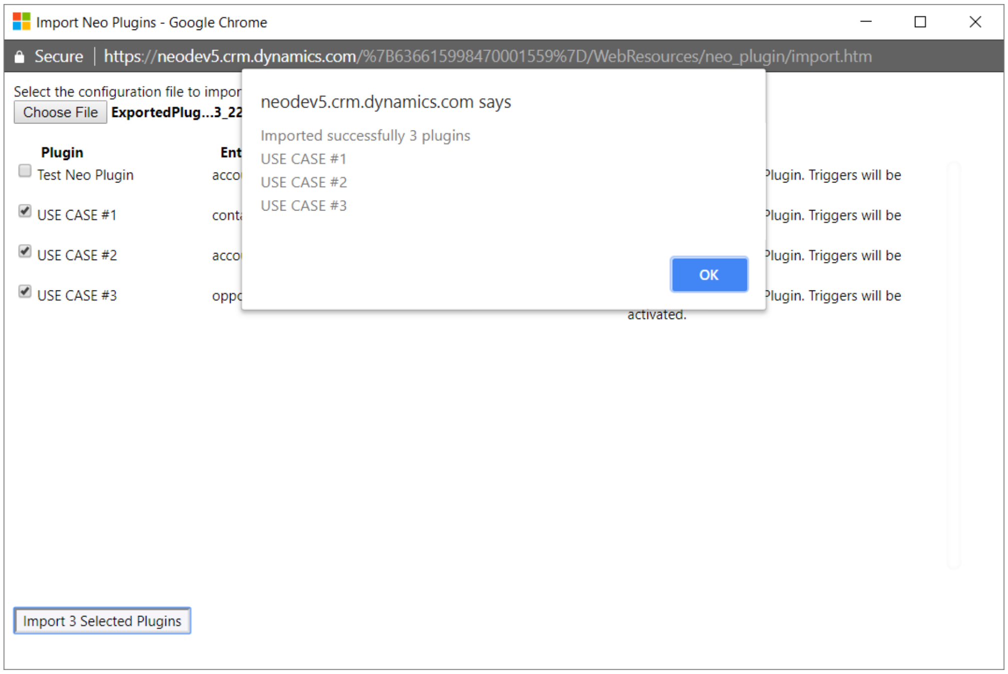
Import Neo Plugins - Google Chrome
Secure | https://neodev5.crm.dynamics.com/%7B636615998470001559%7D/WebResources/neo_plugin/import.htm
Choose File
Import 3 Selected Plugins

Select the configuration file to import:

ExportedPlug...3_224242.xml

	Plugin	Entity	Solution	Status	Action
□	Test Neo Plugin	account	NeoPluginTester	Does not exist	Will create a new NeoPlugin. Triggers will be activated.
☑	USE CASE #1	contact	NeoPluginTester	Does not exist	Will create a new NeoPlugin. Triggers will be activated.
☑	USE CASE #2	account	NeoPluginTester	Does not exist	Will create a new NeoPlugin. Triggers will be activated.
☑	USE CASE #3	opportunityproduct	NeoPluginTester	Does not exist	Will create a new NeoPlugin. Triggers will be activated.

Check the box for Neo Plugins you want to import and click on the Import button. Neo Plugin records will be created and then can be modified directly now.



Import Neo Plugins - Google Chrome
Secure | https://neodev5.crm.dynamics.com/%7B636615998470001559%7D/WebResources/neo_plugin/import.htm

Select the configuration file to impor

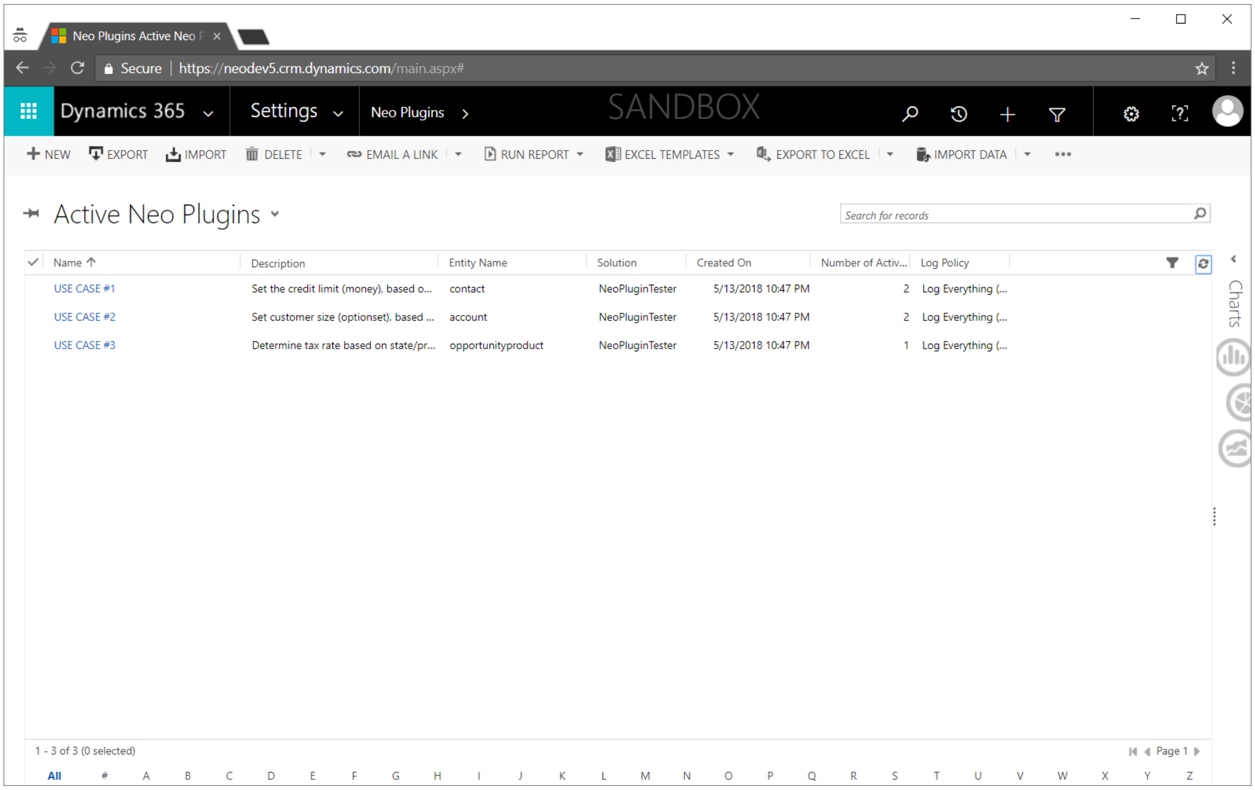
ExportedPlug...3_22

Plugin	Ent	
□ Test Neo Plugin	acco	Plugin. Triggers will be
☑ USE CASE #1	cont	Plugin. Triggers will be
☑ USE CASE #2	acco	Plugin. Triggers will be
☑ USE CASE #3	oppc	Plugin. Triggers will be activated.

neodev5.crm.dynamics.com says

Imported successfully 3 plugins
USE CASE #1
USE CASE #2
USE CASE #3

OK
Import 3 Selected Plugins

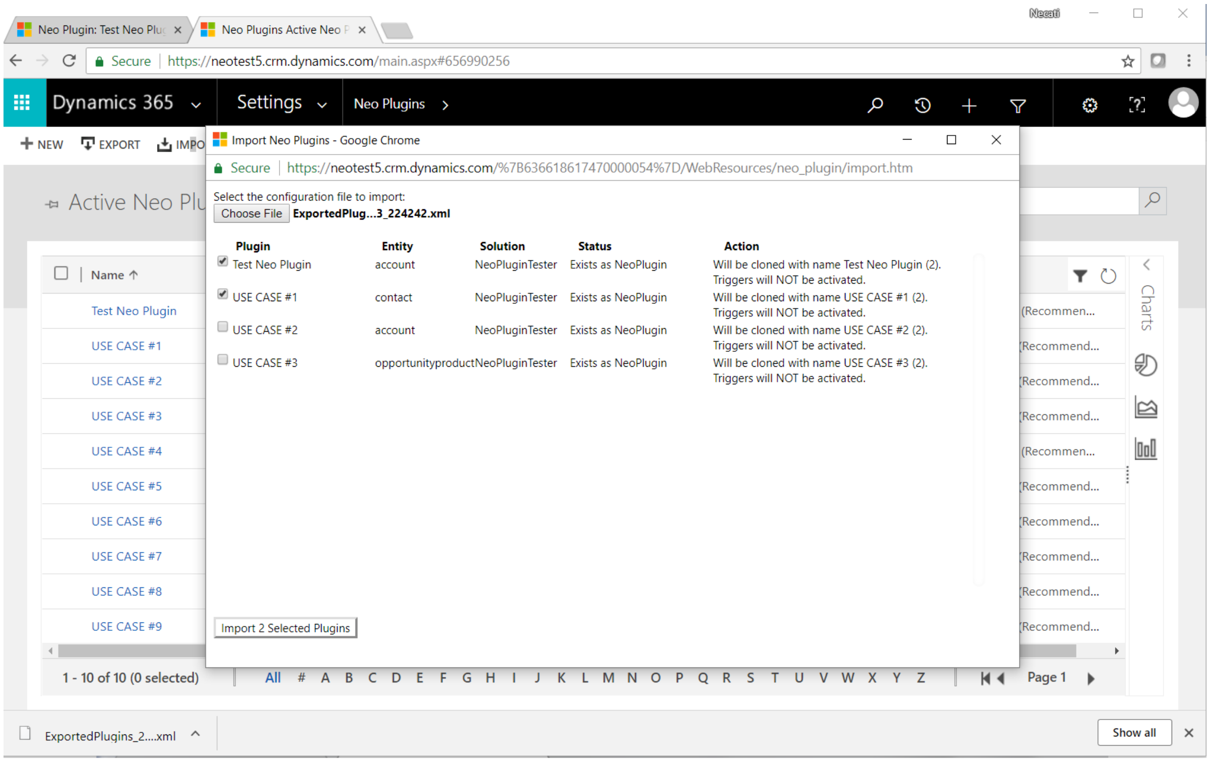


Active Neo Plugins

Name ↑	Description	Entity Name	Solution	Created On	Number of Activ...	Log Policy
USE CASE #1	Set the credit limit (money), based o...	contact	NeoPluginTester	5/13/2018 10:47 PM	2	Log Everything (...
USE CASE #2	Set customer size (optionset), based ...	account	NeoPluginTester	5/13/2018 10:47 PM	2	Log Everything (...
USE CASE #3	Determine tax rate based on state/pr...	opportunityproduct	NeoPluginTester	5/13/2018 10:47 PM	1	Log Everything (...

1 - 3 of 3 (0 selected)

You can use export/import feature to clone Neo Plugins in the same environment, as well. If a Neo Plugin already exists in the CRM, and you want to import it again, it will be cloned with a different name. Its triggers will be left in draft status and you need to publish them, if you want to activate.



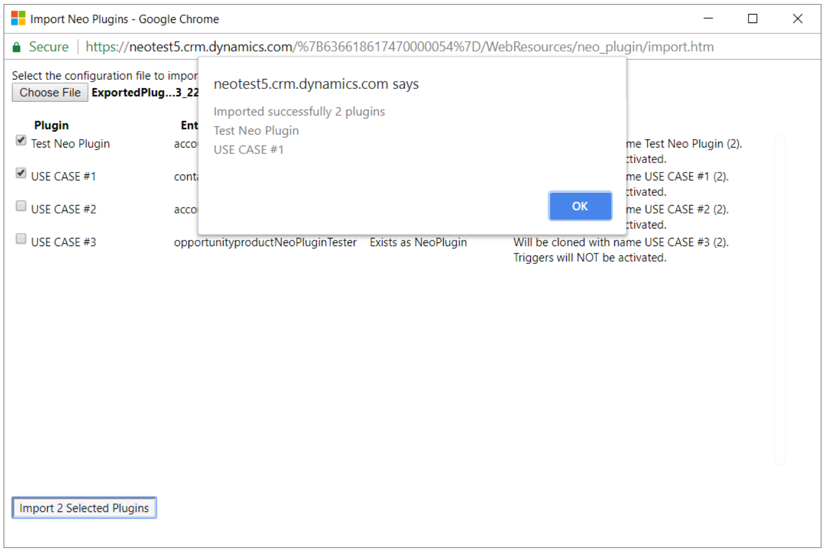
Import Neo Plugins - Google Chrome

Secure | https://neotest5.crm.dynamics.com/%7B636618617470000054%7D/WebResources/neo_plugin/import.htm

Select the configuration file to import:
Choose File ExportedPlug...3_224242.xml

Plugin	Entity	Solution	Status	Action
☑ Test Neo Plugin	account	NeoPluginTester	Exists as NeoPlugin	Will be cloned with name Test Neo Plugin (2). Triggers will NOT be activated.
☑ USE CASE #1	contact	NeoPluginTester	Exists as NeoPlugin	Will be cloned with name USE CASE #1 (2). Triggers will NOT be activated.
□ USE CASE #2	account	NeoPluginTester	Exists as NeoPlugin	Will be cloned with name USE CASE #2 (2). Triggers will NOT be activated.
□ USE CASE #3	opportunityproduct	NeoPluginTester	Exists as NeoPlugin	Will be cloned with name USE CASE #3 (2). Triggers will NOT be activated.

Import 2 Selected Plugins



Import Neo Plugins - Google Chrome

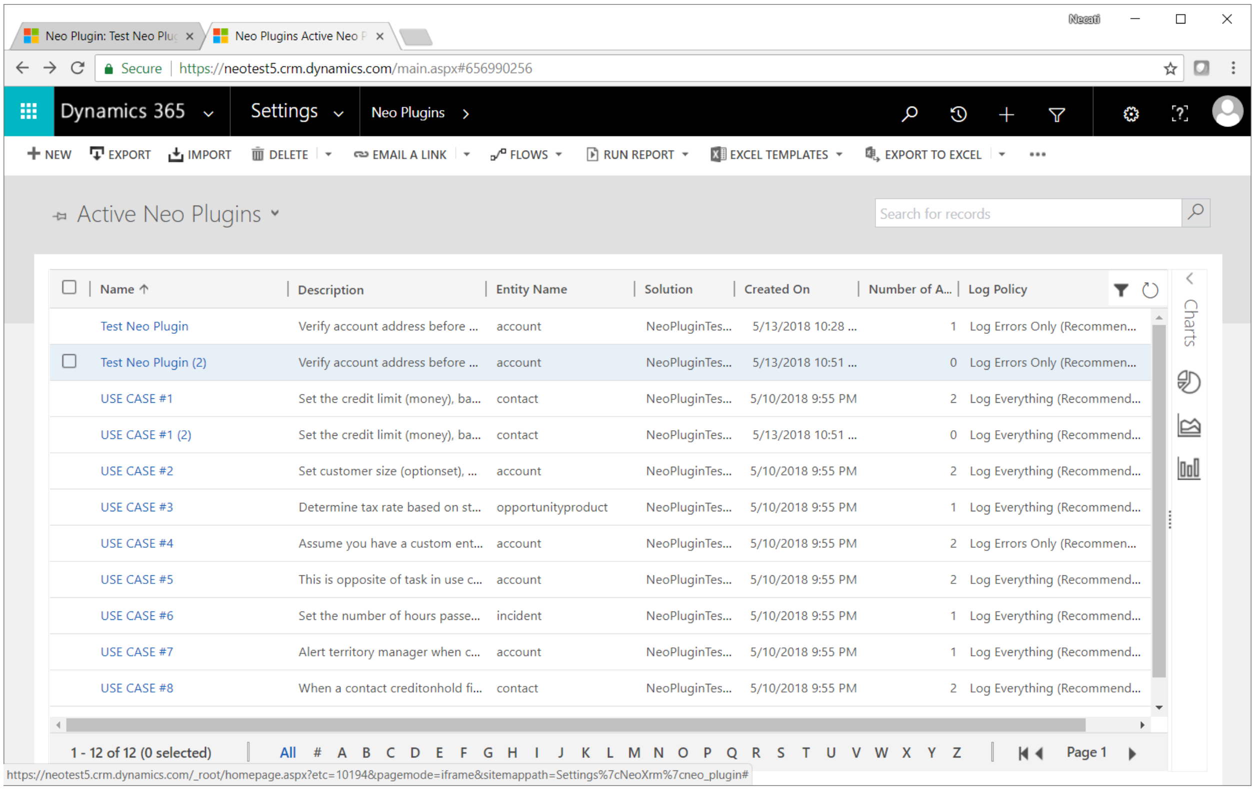
Secure | https://neotest5.crm.dynamics.com/%7B636618617470000054%7D/WebResources/neo_plugin/import.htm

neotest5.crm.dynamics.com says

Imported successfully 2 plugins
Test Neo Plugin
USE CASE #1

OK

Select the configuration file to impor
Choose File ExportedPlug...3_22
Plugin
Ent
Test Neo Plugin
USE CASE #1
USE CASE #2
USE CASE #3
opportunityproductNeoPluginTester Exists as NeoPlugin
Will be cloned with name USE CASE #3 (2).
Triggers will NOT be activated.
Import 2 Selected Plugins



Name ↑	Description	Entity Name	Solution	Created On	Number of A...	Log Policy
Test Neo Plugin	Verify account address before ...	account	NeoPluginTes...	5/13/2018 10:28 ...	1	Log Errors Only (Recommen...
Test Neo Plugin (2)	Verify account address before ...	account	NeoPluginTes...	5/13/2018 10:51 ...	0	Log Errors Only (Recommen...
USE CASE #1	Set the credit limit (money), ba...	contact	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...
USE CASE #1 (2)	Set the credit limit (money), ba...	contact	NeoPluginTes...	5/13/2018 10:51 ...	0	Log Everything (Recommend...
USE CASE #2	Set customer size (optionset), ...	account	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...
USE CASE #3	Determine tax rate based on st...	opportunityproduct	NeoPluginTes...	5/10/2018 9:55 PM	1	Log Everything (Recommend...
USE CASE #4	Assume you have a custom ent...	account	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Errors Only (Recommen...
USE CASE #5	This is opposite of task in use c...	account	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...
USE CASE #6	Set the number of hours passe...	incident	NeoPluginTes...	5/10/2018 9:55 PM	1	Log Everything (Recommend...
USE CASE #7	Alert territory manager when c...	account	NeoPluginTes...	5/10/2018 9:55 PM	1	Log Everything (Recommend...
USE CASE #8	When a contact creditonhold fi...	contact	NeoPluginTes...	5/10/2018 9:55 PM	2	Log Everything (Recommend...

1 - 12 of 12 (0 selected) All # A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Page 1

https://neotest5.crm.dynamics.com/_root/homepage.aspx?etc=10194&pagemode=iframe&sitemappath=Settings%7cNeoXrm%7cneo_plugin#

X. How to Get Support for Neo Plugin

You can always get help for Neo Plugin by contacting to our support team via email at support@neoxrm.com or via phone 1-844-NEOXRM1.