Wordify Numbers

User Documentation

Wordify Numbers is a user-friendly app designed to convert numerical values into their corresponding written words. Users can input any numerical value, and the app will instantly generate the word representation of the number.

How Does It Work?

The app utilizes an Apex Class that can be used in a Salesforce Flow or an Apex trigger. For a documentation on flows see the Salesforce Help on Flow Builder. If you want to use triggers, please refer to the official Apex Developer Guide on Triggers. After installing the app through AppExchange you can use the class, either as an Apex Action in a flow or in a trigger that can be created in the Object Manager.

Flows can be created on your production org but triggers cannot. So, you will need to add it on a sandbox or developer org that is connected to your production org. If you don't have one, yet, please refer to this help article.

Currently Supported Parameters

Number: Decimal or currency numbers with an absolute value up to 999,999,999 are supported. Up to two decimal places are supported.

Language:

- German
- English

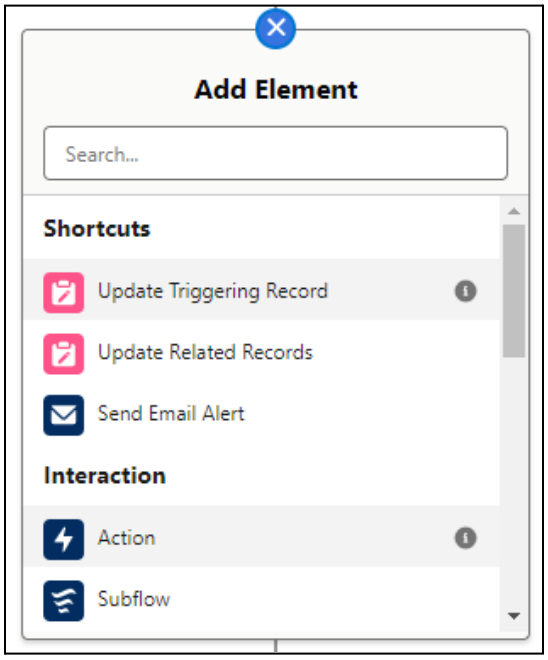
Currency:
- EUR
- USD
- CHF
- Decimal

Note: If no Currency parameter is given in the method call when using the Apex class, the number will be interpreted as decimal.

1. Converting Numbers via a Flow

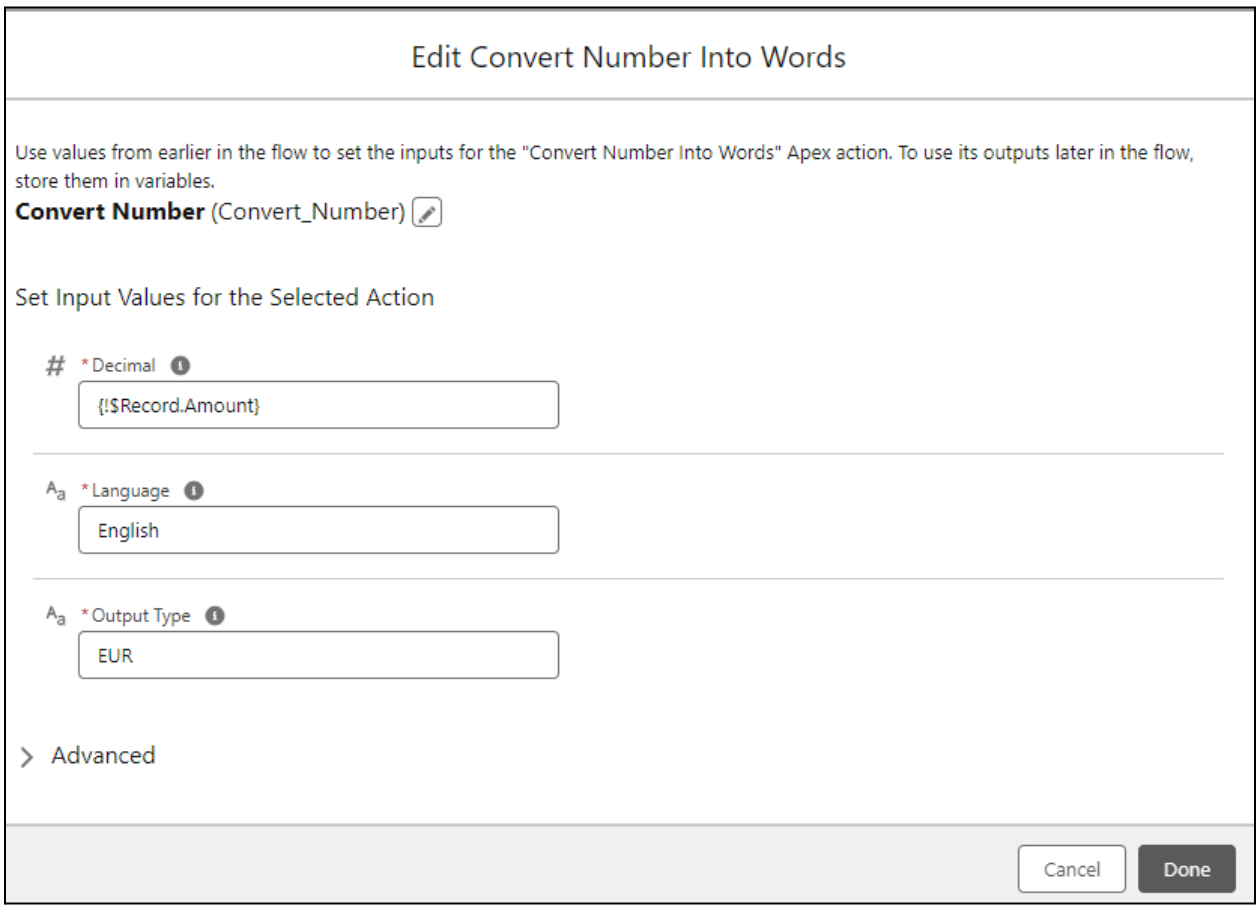
1. Log into your org.
   Note: Before using the app on your production org, we recommend installing it on your sandbox or developer org first and test it for your use-case.

2. Go to Setup.
3. Use the search to find ‘Flows’.
4. Create a new flow or edit an existing one.
   Note: The flow needs to be of type ‘Actions and Related Actions’. Otherwise, you cannot use Apex actions.
5. Add a new element of type ‘Action’.



6. Type ‘Convert Number’ and you find the right class (‘Convert Number Into Words - apex-wordifyNumbers_WordifyNumbers’).
7. Edit the element and set the Input Values:
    a. **Decimal**: The number or currency value to convert.
    b. **Language**: The language to be used (currently supported are ‘English’ and ‘German’).
    c. **Output Type**: The currency code (‘EUR’, ‘USD’, or ‘CHF’) or ‘Decimal’.

    Note: All values can be field references, too.



8. Click ‘Done’.
9. After the action element, you can use an update element to store the value in a field of the record.

2. Converting Numbers via a Trigger

Using a trigger might have several advantages depending on the use-case. You can test it automatically via Test Classes. You already might be using triggers in your org so it could make sense to extend them. To use the app in a trigger you need to add a trigger first.

Adding the Trigger

10. Log into your sandbox or developer org that is connected to your production org.
11. Go to Setup.
12. Go to the Object Manager.
13. Find the object you want to use Wordify Number on.
14. Go to Triggers.
15. Click New.
16. Go to the tab Version Settings and reduce the Salesforce.com API by one unti (e.g. instead of 59.0 select 58.0).
    Note: Since your sandbox or developer org usually runs on the next release for testing purposes, we need to adjust the version to the one of the production org.
17. Go back to the tab Apex Trigger, mark all code and replace it by:

```
rigger TriggerName on ObjectName (before insert, before update) {
   ObjectName[] records = Trigger.new;

    for (ObjectName r: records){
        r.FieldName =
wordifyNumbers.WordifyNumbers.convert(r.FieldName, 'Language',
'Currency');
    }
}
```

Replace the `ObjectName` by the object name and the `FieldName` by the field name you want to use to display the amount as a word.
Valid inputs for `Language` at this point are `English` and `German`.
For `Currency` you can enter `EUR`, `USD`, or `CHF`. If you prefer only the decimal number you can omit the currency parameter.

In this example below, the ObjectName is a custom object named “Payment”. We want to see the amount in words on three fields: Once in German (“Amount in Words (German) EUR”) in Euros and once in English (“Amount in Words (English)”) as decimal.

```
trigger WordifyNumbersTrigger on Payment__c (before insert, before
update){
```

```
    Payment__c[] payments = Trigger.new;

    for (Payment__c p: payments){
        p.Amount_in_Words_German_EUR__c =
wordifyNumbers.WordifyNumbers.convert(p.Amount__c, 'German', 'EUR');
        p.Amount_in_Words_English__c =
wordifyNumbers.WordifyNumbers.convert(p.Amount__c, 'English');
    }
}
```

Note: In the trigger, we need to use the *technical* names (API names). Therefore, you see underscores instead of spaces and the "__c" at the end for custom objects and fields. If you are not sure what the API object name is you can look it up in Details or in the default code that will show when you create the trigger. For the field names, please go to Fields & Relationships and use the Field Name. You can only save the trigger if all object and field references are correct.

For more information on triggers go to Defining Triggers in the Apex Developer Guide.

Test the Trigger

Before deploying a trigger, we need to add a unit test to perform the actions that fire the trigger and verify expected results.

1. On your sandbox, go to Setup and find `Apex Classes` via the search.
2. Create a new Apex class by clicking on `New`.
3. Insert the code below and replace the placeholders with the respective object and field names.

```
@isTest
private class TestTriggerName {
    @isTest static void testInsertObjectName() {
        // Test data setup

        // TODO: Potentially create required records to insert object

        // Create a record with amount
        ObjectName r = new ObjectName(Amount__c=123);
        insert r;

        // Select inserted record
        r = [SELECT Amount_in_Words_German_EUR__c,Amount_in_Words_English__c FROM
ObjectName where Id=:r.Id];

        // Verify
```

```
        System.assertEquals('einhundertdreiundzwanzig
Euro',r.Amount_in_Words_German_EUR__c);
        System.assertEquals('one hundred and twenty
three',r.Amount_in_Words_English__c);
    }
}
```

Note: The test class for the trigger needs to include all the fields that are used on the trigger.

In this example below, the object where the trigger is implemented is a custom object named “npe01__OppPayment__c”. It is part of the Non-Profit Package and therefore has the prefix “npe01__”. To be able to create a record of this payment object, we need to have an Account and an Opportunity. In the end, we need to check whether the amount in words is equal to the expected result.

```
@isTest
private class TestWordifyNumbersTrigger {
    @isTest static void testInsertPayment() {
        // Test data setup
        // Create an Account
        Account a = new Account(Name='Wordify Test');
        insert a;

        // Create a new Opportunity
        Opportunity o = new Opportunity(Name='Wordify Test Opportunity',
AccountId=a.Id, CloseDate=System.today().addMonths(1), StageName='Qualification');
        insert o;

        // Create a Payment
        npe01__OppPayment__c p = new
npe01__OppPayment__c(npe01__Opportunity__c=o.Id,npe01__Payment_Amount__c=123);
        insert p;

        // Select inserted Payment record
        p = [SELECT Amount_in_Words_German__c,Amount_in_Words_English__c FROM
npe01__OppPayment__c where Id=:p.Id];

        // Verify
        System.assertEquals('einhundertdreiundzwanzig',p.Amount_in_Words_German__c);
        System.assertEquals('one hundred and twenty
three',p.Amount_in_Words_English__c);
    }
}
```

Run the Test Class

If you want to make sure that the test class has been implemented correctly you can click ‘Run Test’ on the test class.

In the window that opens the test class will run. Once it has finished you can see either a green tick or a red x depending on whether it has completed successfully or not.

Create the Outbound Change Set

To transfer the trigger to your production org, you need to create an outbound change set.

1. To view outbound change sets, from Setup, enter `Outbound Change Sets` in the `Quick Find` box, then select Outbound Change Sets.
2. To create a new change set, click `New`.
3. Click `Add` next to `Change Set Components`.
4. From the choosebox, select the component `Apex Trigger`.
5. Select the trigger you have created earlier and click `Add to Change Set`.
6. Click `Add` next to `Change Set Components`.
7. From the choosebox, select the component `Apex Class`.
8. Select the test class you have created earlier and click `Add to Change Set`.
9. Click `Upload`.
10. Select your production org and click `Upload`.
11. You will be notified via email when the change set was uploaded.

For more information on outbound change sets see Upload Outbound Change Sets.

Deploy the Inbound Change Set

To deploy the trigger on your **production** org, you need to use an inbound change set.

1. From Setup, enter `Inbound Change Sets` in the `Quick Find` box, then select Inbound Change Sets.
2. Click Deploy next to the change set you had uploaded earlier.
3. If you choose `Default` all tests except the ones that originate from managed packages are executed. You can optionally choose to run all or specified tests, for example, only the trigger test class that is part of the inbound change set.

4. Wait for the deployment to be finished. You can track the deployment status on the page that is linked under `Deployment Started`.
   Note: Make sure that all objects and related field fields exist on your production org and that the Wordify App is installed.
5. When all components are deployed successfully, the status will show a green complete circle. If any errors occurred, please view them and get in touch with our support in case of doubts.

For more information on outbound change sets see Deploy Inbound Change Sets.

Who Can Help Me?

If you have doubts or run into problems, please contact us on info@synterface.io.