

AppWorks Platform Example Project

Send Email To Many

Home Page ▾

Home > 00008

Print

Delete

Send email

Description

Concat related emails into To address

Interested Party Emails

ggreen@innovate.com;bbrown@innovate.com;bbblue@innovate.com

Interested Parties

☒ Gary Green

☒ Ben Brown

☐ AppWorks Developer

☒ Barbara Blue

New email

To


ggreen@innovate.com;bbrown@innovate.com;bbblue@innovate.com

Cc

Subject

Sending email to multiple recipients

Attachments




↩ ↪

B *I* U ~~S~~ *I*_x

Font ▾

Size ▾



This email is being sent to multiple people

Introduction	3
Business Use Case	3
The Design Challenge	3
The Solution	3
Implementation	4
1. Create Primary Entity	4
2. Add Property	4
3. Add List	5
4. Add Relationship	5
5. Add Web Services	6
6. Add Forms	6
Create	6
Default	7
7. Add Business Process	8
Process Messages	8
Read Related Entities	9
Join Email Addresses	10
Update Entity	12
8. Business Rules	14
On Interested Party Change	14
On Removing Last Relationship	14
9. Testing	15
About OpenText	16

Introduction

This guide outlines the configuration of the Example AppWorks Platform Project. Please note that the Examples Workspace has been built using version 16.4.2 (16.4 EP2) of AppWorks Platform.

If you have an older version of the product installed you will not be able to import the Workspace. In that case, you can use the detailed tutorial in the next section of this guide to re-create the project in your environment,

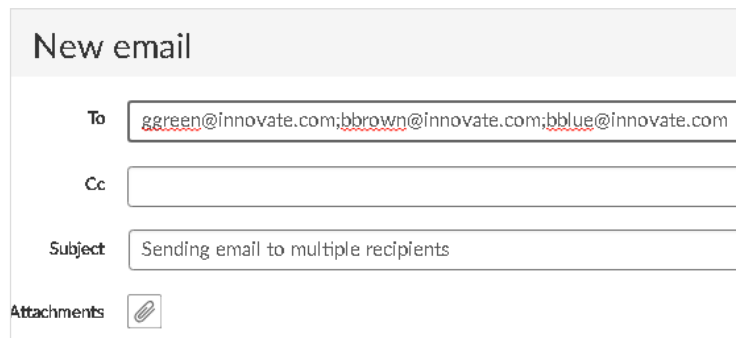
Business Use Case

The use case in this example project is the sending of an email to multiple 'interested parties' of a case, where a one-to-many relationship to an entity is used to maintain the list of 'interested parties'. It also applies to any situation where you want to use repeating entity values but use a concatenated list of those values in the primary entity user interface, such as in a list or on a form.

The Design Challenge

When using a one-to-many relationship between entities, the expression language does not allow you to access the values of the multiple related entities, apart from some arithmetic calculations such as totalling numeric values.

In this use case there is a need to build the list of addresses from the related entities so that it can be used to pre-populate the To field of an Email Template as shown here.



The screenshot shows a 'New email' form with the following fields:

- To:** ggreen@innovate.com;bbrown@innovate.com;bblue@innovate.com
- Cc:** (empty)
- Subject:** Sending email to multiple recipients
- Attachments:** (empty)

The Solution

The solution is to use an additional property on the primary entity that holds the concatenated list of values, and a business process that concatenates the appropriate value(s) from the related entities and updates the new property of the primary entity each time a relationship changes. The process is triggered through Rules when a related entity is added or removed. An additional Rule is used for a specific situation where the last relationship is removed.

Implementation

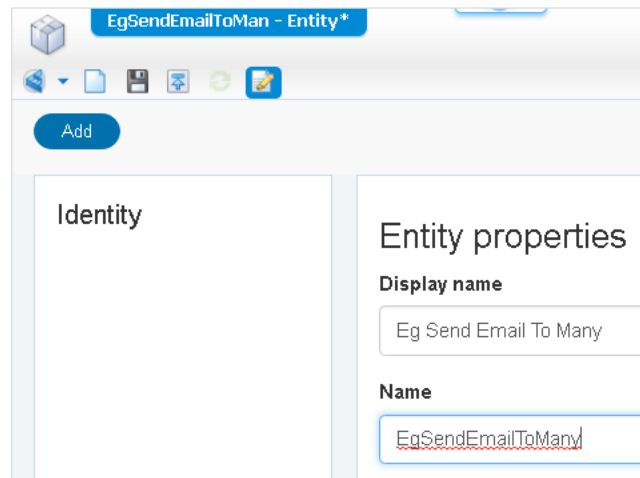
The following tutorial takes you through the steps needed to re-create the project in your own environment. If you are going to add this functionality to your own application directly, then there may be steps that can be skipped, such as creating the primary entity used for the example, and the forms and layouts. Where possible these steps are highlighted.

This tutorial assumes you are familiar with AppWorks Low Code Platform and know how to configure such things as entities and business processes. Deployment and testing is left until the end, but you can always publish and test along the way if you wish to. If you do this, don't forget to enable use of the application in AppWorks Administrator the first time you publish.

1. Create Primary Entity

This step can be skipped if you are using an existing application and already have an entity that you want to use.

Creating the primary Entity and give it a name such as *EgSendEmailToMany*.

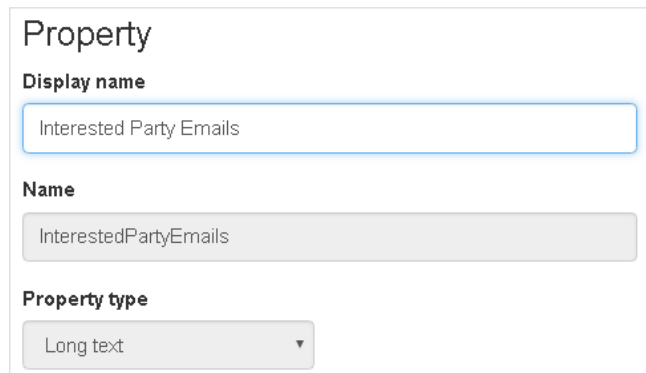


2. Add Property

Add a property to the Entity to hold the concatenated list of values.

Give it a name such as *Interested Party Emails* and make it of type **Long Text** so that there will be no issues with the number of values.

Save the changes but keep the entity open.



3. Add List

This step can be skipped if you are using an existing application, although you may want to add the new property to one or more existing lists at this step.

Add a List called *Eg Email Send To Many* and add **ID** and **Interested Party Emails** to it.

Save the list and close it, and then save the changes to the entity but keep it open.

4. Add Relationship

This step can be skipped if you are using an existing application and the list of repeating values is in an existing one-to-many related Entity.

Add a Relationship called *Interested Parties* and make it of **Type Peer** and **Multiplicity To many**.

Click **Browse** and select the *Person* Entity which comes from the Identity Package, as it should already contain your systems users with their email addresses if you are using OTDS. If it does not you may need to create a dedicated entity for this purpose.

Save the changes but keep the entity open.

5. Add Web Services

Add the Web service Building Block for your primary entity.

Enable the **Update** services from the Basic section.

Expand the Relationships section and enable the **Get** service for your related entity.

Save the changes but keep the entity open.

Web service

Basic

Namespace
http://schemas/ExamplesSendEmailToMany/EgSendEmailToMany/operations

Operations
☐ Create ☐ Read ☒ Update ☐ Delete

Relationships

InterestedParties
☐ AddTo ☒ Get ☐ RemoveFrom

6. Add Forms

This step can be skipped if you using an existing application, although you may want to add the property to one or more existing forms at this step.

Create

Start by adding a form called *Create* and configure it by adding the **Interested Parties** relationship to it. Make the control **Type** *Multiselect*, and choose *Select user* for the **Browse list**, and *Display name* for the **Property**. You can also change the label to *Interested Parties*.

Create - Form*

Components

- Identity
 - 10 ID
 - 1 Item ID
 - Entity type
 - 1 Item status
- Action Buttons
 - Delete
 - Print
- Relationships
 - [0..*] Interested Parties
- Properties
 - Interested Party Emails

Presentation

Form

Interested Parties

- ☒ Item One
- ☒ Item Two

Interested Parties
OneToOne

Presentation

Type
Multiselect

Browse list
Select user

Property
Display name

☒ Show label
Interested Parties

In this example we have added a filter to the **Browse list** (see the dark funnel icon in the previous screenshot). This is to remove any users that do not have a *Display name* value in OTDS, such as some internal system users. You can click on the funnel icon to set this if you want but it is not essential.

The expression `Properties.DisplayName=ne(Null)` removes values where the *DisplayName* is 'not equal' to Null.

Save the form and close it, and then save the changes to the entity but keep it open.

Default

Add a second form called *Default*, and this time add the same **Interested Parties** relationship (configured as above), but also add the **Interested Party Emails** property, making it of **Type Multi-line** and Display type of *Read-only*.

Filter

Enter an expression to filter the browse list
Ex: `Properties.Type={item.Properties.Type}`

`Properties.DisplayName=ne(Null)`

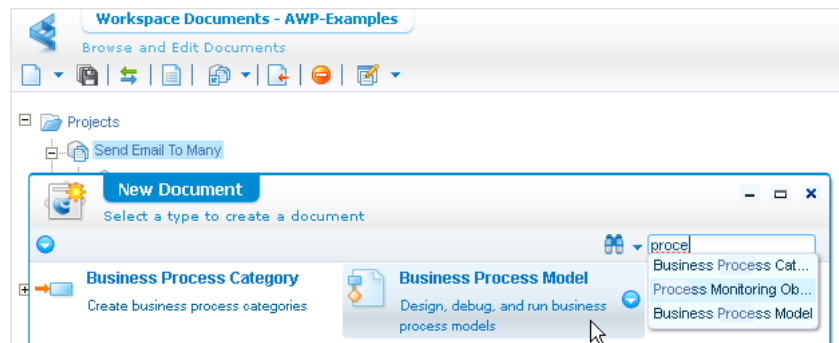
This form will allow you to see the property being updated dynamically as you add/remove relationships.

Save the form and close it, and then save the changes to the entity and close it for now.

7. Add Business Process

Start by adding a new Business Process Model to your application. You can use a filter to make it faster to find it as shown here.

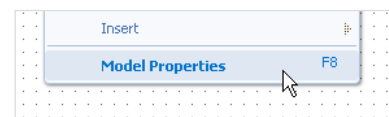
It is a good idea to save it immediately and give it a name such as *Util Update Interested Party Emails*.



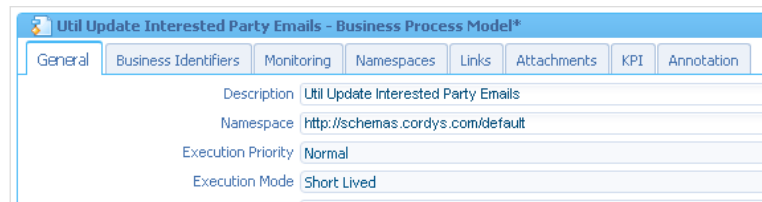
On the **Model** tab draw the basic flow of the process, with three connected activities.



Double-click the canvas (or right-click and select Model Properties from the context menu) to bring up the properties panel.



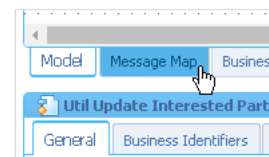
Under **General**, select an **Execution Mode** of *Short Lived*.



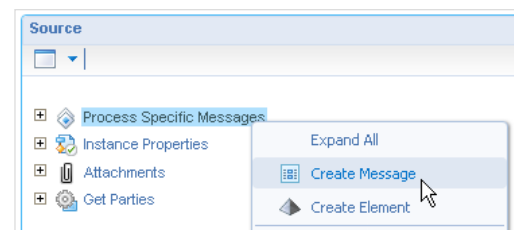
Process Messages

Before configuring the activities further it is necessary to add a Process Specific Message that will hold the concatenated list of values.

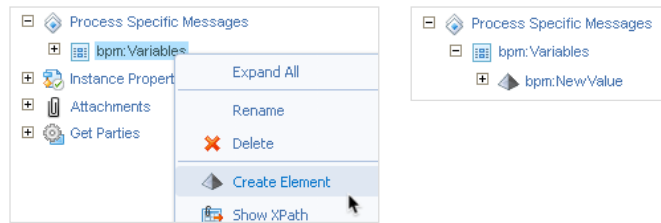
On the main canvas, click the **Message Map** tab.



Under the **Source** panel right-click and choose **Create Message** from the context menu. Give it a name of Variables.



Right-click on the new message and choose **Create Element**. Give it a name of `NewValue`.



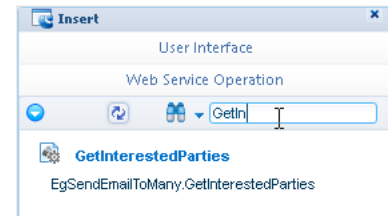
Read Related Entities

The first activity in the process will call the *Get* Web Service of the relationship to read the related parties to the primary entity.

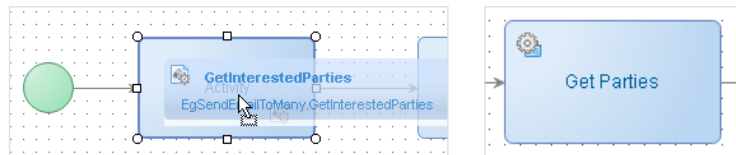
Go back to the **Model** view of the process. In the left panel of the designer select the **Insert** tab at the bottom, and click the **Web Service Operation** header to access all the web services that are available to you.



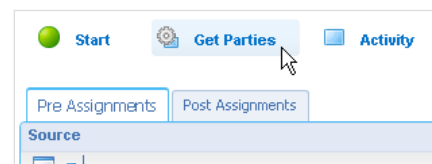
Use the filter to locate the *Get* web service from your entity (*GetInterestedParties* in this case).



Drag the web service operation from the left onto the first activity. The name and icon of the activity should change. You can then double-click the name and change it to *Get Parties*.

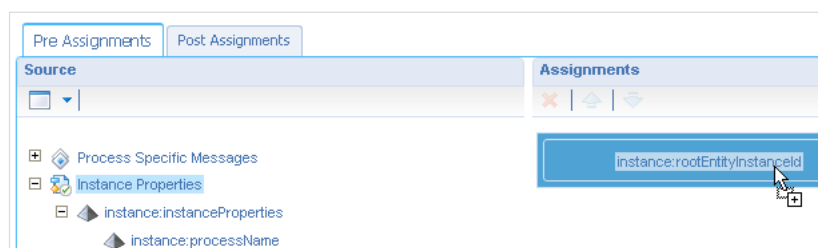


Go back to the **Message Map** view and select the *Get Parties* activity.



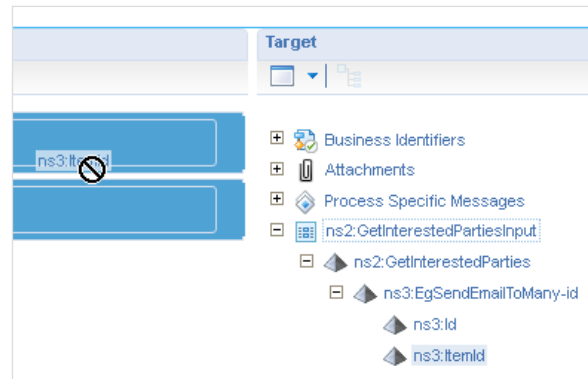
On the **Source** side of the **Pre Assignments** tab expand the **Instance Properties** > **instance:instanceProperties** node and drag the **instance:rootEntityInstancelId** value to the left hand side of the **Assignments** panel.

This value is automatically populated with the Entity ID when the process is started from a Rule, which is configured later in this tutorial.



On the **Target** side of the screen expand the `ns2:GetInterestedPartiesInput` node out fully and drag the value `ns3:ItemId` across to the right hand side of the **Assignments** panel.

This is the input value to the *Get* Web Service, and this configuration will pass the **ItemId** value of the entity into the web service when the process runs, and get the related party results, which are used in subsequent activities.



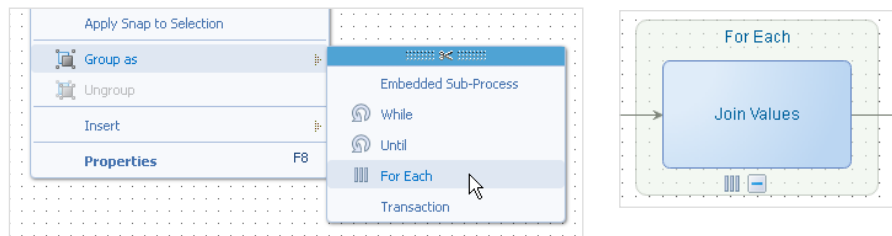
Join Email Addresses

The second activity will loop through each of the related parties retrieved in the previous activity, and add the email address to the *NewValue* process variable. We will also add a filter to this so that only those with an email address are included, in case some do not.

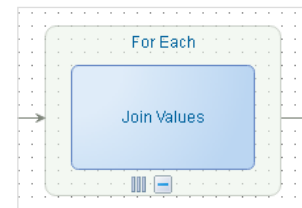
This is probably the most complicated part of the whole tutorial!

Go back to the **Model** view and right-click on the second activity and choose **Group as > For Each** from the context menu. This will add a repeating container around it.

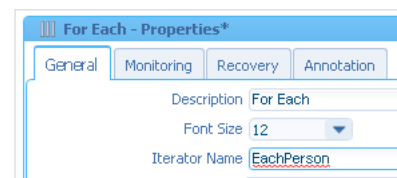
Change the name of the activity to *Join Values*.



Although not essential, it is good practice to delete the connector into and out of the Join Values activity, and to re-draw them to the For Each group instead like this.

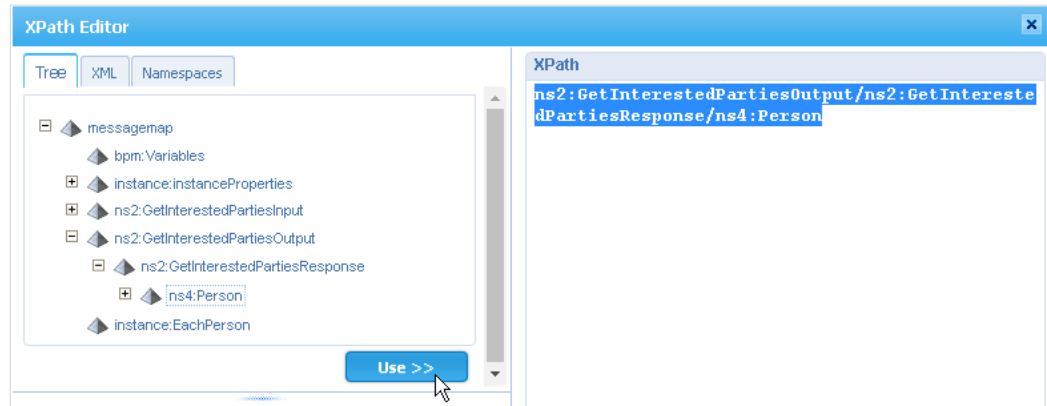


Double-click the **For Each** container (or right-click and choose **Properties**) to open its properties. Change the **Iterator Name** to *EachPerson*. You will use this name later when looping through each Person entity.



Click the Lookup icon next to the Select Condition value to configure the XPath expression for the repeating element of the data from the web service.

In the XPath Editor dialog, expand the `ns2:GetInterestedPartiesOutput` node (this is the output of the Get web service) and locate the `ns4:person` element, which is the repeating element. Note that the namespace prefixes may be different in your environment. Double click or click the **Use >>** button to add it to the expression.



To filter out the people without an Email address value add `[ns4:Email!='']` to the end of the XPath expression, without any spaces. Again, make sure the namespace matches that of the Person in your environment (`ns4` in this case).

XPath
`ns2:GetInterestedPartiesOutput/ns2:GetInterestedPartiesResponse/ns4:Person[ns4:Email!='']`

Click the **Validate** button to check that there are no errors, and then click **OK** to save the expression.

NOTE be careful if copying and pasting apostrophe characters into the expressions - you may need to delete and re-type them as they sometimes get replaced with invalid characters.

Go back to the **Message Map** view and select the **Join Values** activity. The mapping you create here will append each email address to the **NewValue** variable.

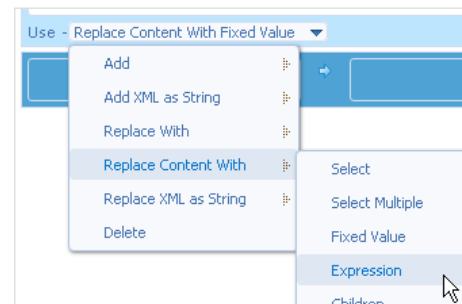
Under **Target** expand the **Process Specific Messages** node fully and drag `bpm:NewValue` to the right-hand side of the assignment.



Select **Replace Content with Expression** from the dropdown menu underneath the assignment.

The expression which you configure next will concatenate the existing value of the `NewValue` variable with a semicolon (which is what the To address in the Email Building Block requires), and then append the current persons email address from within the loop.

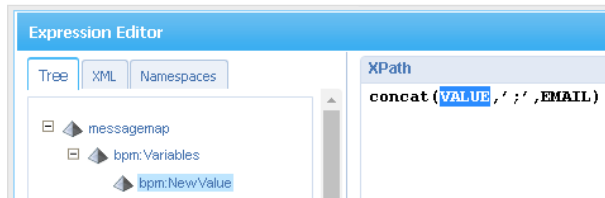
Click the **Expression Editor** link to open the XPath editor.



Start by typing the expression `concat(VALUE, ';' ,EMAIL)` into the XPath value.

XPath
`concat(VALUE, ';' ,EMAIL)`

Highlight the **VALUE** part. Expand the **messagemap** on the left and double-click the `bpm:NewValue` node to insert it into the expression.



XPath
`concat(bpm:Variables/bpm:NewValue/text(), ';' ,EMAIL)`

For the **EMAIL** part, select it as above and double-click the `instance:EachPerson` node. This is the iterator name defined earlier. It signifies that the root of this expression is the next in the loop.

XPath
`concat(bpm:Variables/bpm:NewValue/text(), ';' ,instance:EachPerson/text())`

This XPath expression needs a little more editing to make sure it is valid and that it selects the Email address of the Person and not the whole Person (which is what it is doing now). Edit the Expression by adding `/ns4:Email` after `EachPerson`. Again, your namespace may be different. You can check the value to add here by expanding the **messagemap** on the left.

XPath
`concat(bpm:Variables/bpm:NewValue/text(), ';' ,instance:EachPerson/ns4:Email/text())`

Update Entity

The final activity in the process will call the *Update* Web Service of the entity to set the property with the concatenated string.

Go back to the **Model** view, and as you did with the first activity, locate the Web Service Operation for the Update web service of the entity (in this case you can use a filter of *UpdateEg*).

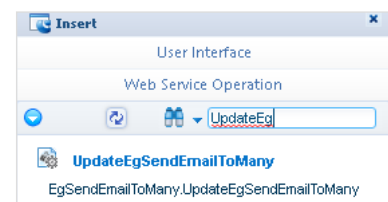
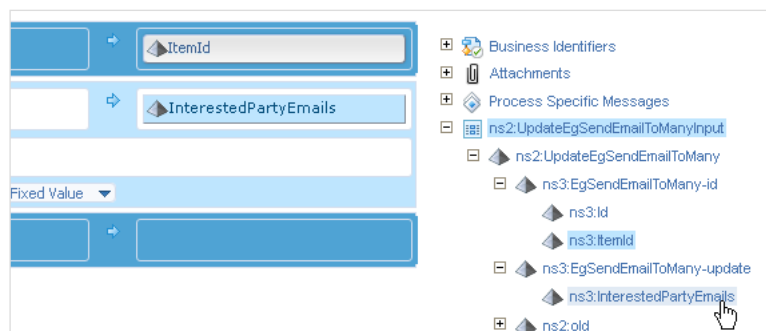
Drag the web service onto the final activity in the process and rename it to *Update*.

In the **Message Map** view select the *Update* activity, and prepare two assignments by dragging these nodes from under the **Target** section:

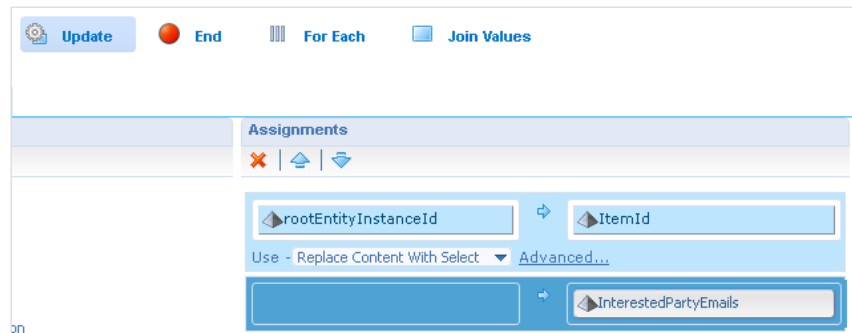
`ns3:ItemId`

`ns3InterestedPartyEmails`

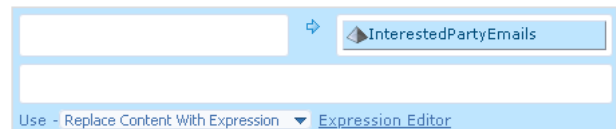
These are the inputs to the *Update* web service on the primary entity.



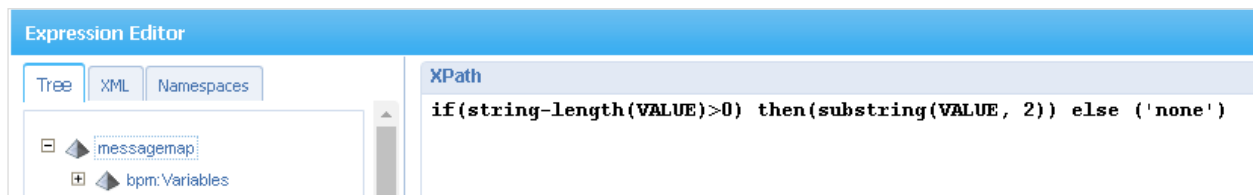
From the **Source** section, again drag the **Instance Properties** value `instance:rootEntityInstanceId` into the **ItemId** assignment.



Select the second assignment, choose **Replace Content with Expression** from the dropdown, and click **Expression Editor** to edit the expression.



Enter the string `if(string-length(VALUE)>0) then(substring(VALUE, 2)) else ('none')` into the XPath expression. You will replace the **VALUE** placeholders in the next step.

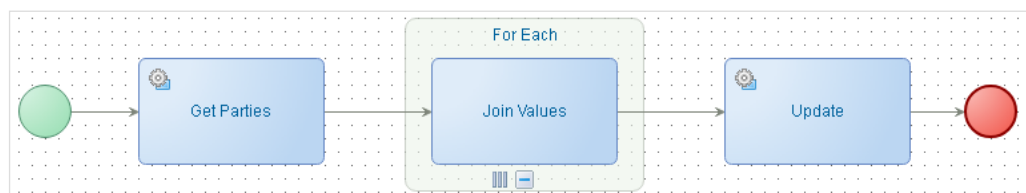


This expression says that if the concatenated string is longer than 0 (i.e. there was at least one email address) then update the entity with the string value without the first semicolon (which is always added hence the substring), otherwise return a string of 'none'. This is required because the *Update* web service cannot easily set a null string.

Select the first **VALUE** placeholder, and from the **messagemap** on the right, expand the node `bpm:Variables` and double-click `bpm:NewValue`. Repeat this for the second **VALUE** placeholder. The expression should look like this. Click OK to close the editor.



Your final process should now look like this. Save and close the model.



8. Business Rules

The final thing to configure before testing things is to add the business rules that will trigger the process.

On Interested Party Change

Add another Rule called *OnChange Interested Party* and open the Rule editor.

For the **Event** section choose *A relationship changes* for the **When** value, and *InterestedParties* for the **Relationship** value.

For the **Condition** section select the property *Item ID* and a condition of *not empty*. This ensures that the process will not run while the user is creating the entity for the first time with the Create form, when the ItemId is not set (and so the entity cannot be updated).

For the **Action** section select *Start process* and then click the find icon to choose the business process you just configured.

Save and close the Rule editor.

The screenshot shows the Rule editor interface. The 'When' section is set to 'A relationship changes' and the 'Relationship' is set to 'InterestedParties'. The 'If' section is set to 'all of the following are true' with a condition for 'Item ID' being 'not empty'. The 'Then' section is set to 'Start process' and a search for 'Util Update Interested Party Emails' is shown.

On Removing Last Relationship

This last Rule is a special case that is required because when the final relationship is removed, although the process is executed, the concatenated list of values is empty, and the Update web service of the entity does not clear the value of the property - instead it leaves the last value there. This is why the process was configured to return a value of 'none' when there are no more relationships. This rule then clears the property when that occurs.

Add a Rule called *OnChange Interested Party Clear* and open the Rule editor.

For the **Event** section choose *A property changes* for the **When** value.

In the **Condition** section add a condition for when *Interested Party Emails equal to none*.

The screenshot shows the Rule editor interface. The 'When' section is set to 'A property changes'. The 'If' section is set to 'all of the following are true' with a condition for 'Interested Party Emails' being 'equal to' 'none'.

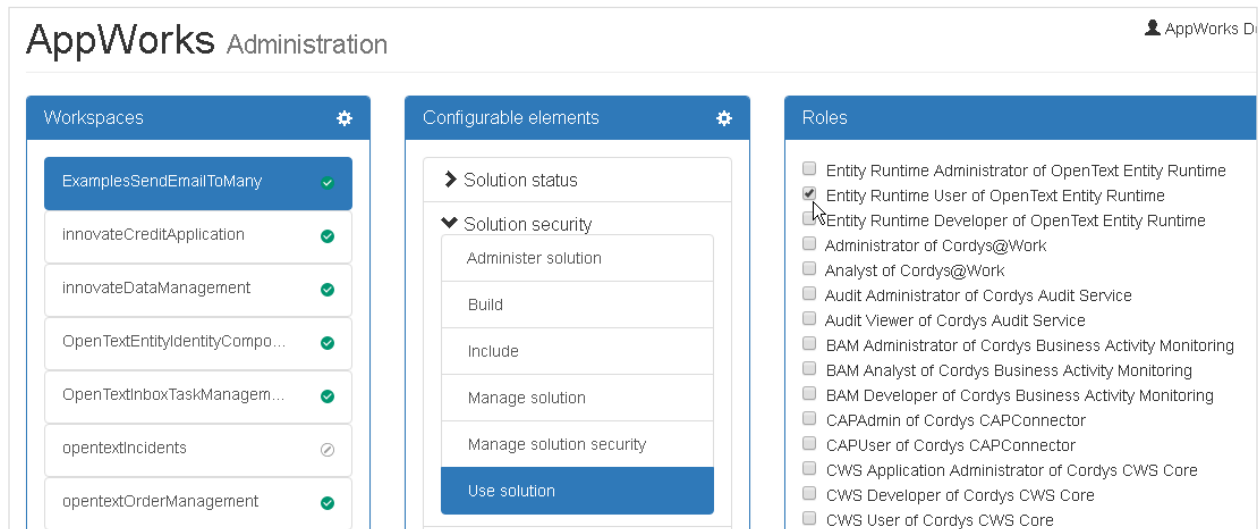
In the **Action** section choose Set property with a **Target** of `item.Properties.InterestedPartyEmails`, and a **Source** of null string (single quotes)

Save and close the Rule editor.

The screenshot shows the Rule editor interface. The 'Then' section is set to 'Set property'. The 'Target' is set to 'item.Properties.InterestedPartyEmails' and the 'Source' is set to an empty string ' ' '.

9. Testing

When done, save all changes and publish the application. Don't forget to grant access to users in AppWorks Administration.



When you log in now and create an instance of your entity, and then open it from the list, you can see the process dynamically updating the list of values in the *Default* form.

Interested Party Emails

ggreen@innovate.com;awpdev@innovate.com;bblue@innovate.com

Interested Parties

- ☒ Gary Green
- ☐ Ben Brown
- ☒ AppWorks Developer
- ☒ Barbara Blue

You can go further and add an Email Building Block, Email template and Layout to achieve the UI shown on the front page of this guide. In the Email Template the To value is taken from this concatenated list.

To {item.Properties.InterestedPartyEmails}

About OpenText

OpenText enables the digital world, creating a better way for organizations to work with information, on premises or in the cloud. For more information about OpenText (NASDAQ: OTEX, TSX: OTC) visit opentext.com.

Connect with us:

[OpenText CEO Mark Barrenechea's blog](#)

[Twitter](#) | [LinkedIn](#) | [Facebook](#)