Table of Contents

Programmers Reference, Client Side Plugins

Writing client-side plugins

In **Aware IM** you can not only add plugins for the server (such as custom processes, channels or functions), but you can also add plugins that execute on the client within a web browser. Most of the time you would write these plugins in order to add your custom user interface functionality, or modify the default **Aware IM** user interface behaviour.

All client-side plugins must be written in Javascript and in most cases you need the knowledge of the Kendo UI Javascript library from Telerik and a popular open source Javascript library called jQuery . The description that follows assumes that the reader is familiar with Javascript, Kendo UI library and jQuery.

There are several types of the client-side plugins you can add in **Aware IM**:

1. Modify the default behaviour and presentation of forms
2. Modify the default behaviour and presentation of form sections in forms
3. Modify the default presentation of individual fields within forms
4. Modify the default presentation and behaviour of queries
5. Modify the default presentation and behaviour of content panels inside visual perspectives

We will look at each of these client-side plugins separately

Send email to the selected contact

We need to use the pickContact method of the navigator.contacts object to display a list of contacts, let the user pick one and then we need to start a process in the application to send an email to the email address of the contact picked by the user.

The email address returned by the plugin needs to be saved in some temporary object and then this object can be used in the process. So we will create a temporary business object (persistence type – memory) called ContactParam with the single EmailAddress attribute. We will then create a process called SendEmailToContact with the ContactParam object as its input. The process will then use the SEND action to send any email to this email address (the email can use tag expressions to retrieve some information from the system – for example, from SystemSettings or from the logged in user).

To start a process we will use the AwareApp.startProcessWithInit function. It has the following signature:

startProcessWithInit: function (procName, renderOption, objName, initValues, context)

Here procName is the name of the process to start, renderOption is where to display the results of the process (we can use null), objName is the name of the parameter object, initValues is the object storing values of the parameter object and context contains additional parameter objects (null in our case)

So our Javascript can look like this:

```
navigator.contacts.pickContact (function (contact)

{

var email = contact.emals[0].value;

AwareApp.startProcessWithInit (

"SendEmailToContact",

null,

"ContactParam",

{ "EmailAddress" : email }

},

function (error { console.log (error); }

);
```

Then we just need to add a command of the "Execute Script" type to the mobile menu of the application.

- [Architecture of the client-side code](#)
- [Modifying default behavior and presentation of queries](#)
- [Modifying default behavior and presentation of forms](#)
- [Modifying default behavior and presentation of form sections](#)
- [Modifying default presentation of individual fields on forms](#)
- [Modifying default behavior of menu in visual perspectives](#)
- [Modifying default behavior and presentation of content panels in visual perspectives](#)
- [AwareApp object](#)
- [Using Javascript to integrate custom Cordova plugins for native mobile applications](#)