HBW widget integration manual

Config parameters

These params are used for connection third-party application to HydraOMS:

- `hbw_url` – HydraOMS usl (e.g., **http://localhost:3000**);
- `hbw_public_path` – url of HydraOMS which is assessible using browser (e.g. **https://some.domain:8100** or **https://homs.some.domain**, used for loading static files and opening web-sockets. You can use the same value for both `hbw_url` and `hbw_public_path`);
- `hbw_login` – login (e.g., **user@example.com**);
- `hbw_token` – token (e.g., **renewmeplease**).

Fixed parameters

These params correspond to the certain third-party application and entity type will be used by running business process:

- `entity_class` – integration type (e.g., **billing_customer (Hydra Service Provider Console application), self_care_customer (Hydra Self Care Portal application), customer_care_portal (Hydra Customer Care Portal application)** or other application name);
- `entity_type` – entity type (e.g., **customer, account, operator** or other string value).

Dynamic parameters

These params correspond to running business process instanse:

- `entity_code` – unique entity identifier or code;
- `user_identifier` - email of user starts business process instance (usually the same as `hbw_login`);
- `initial_variables` – additinal variables will be set while starting a business process instance.

Helpers

The following methods need to be implemented for sending GET, POST, PUT and DELETE requests to the HydraOMS backend:

```
def request_widget_backend(path, method = :get, parameters = {})
  request_params = {
    method:   method,
    url:      build_bpm_widget_path(path),
    user:     bpm_config.hbw_login,
    password: bpm_config.hbw_token
  }

  if method == :get
    request_params.merge!({headers: {params: parameters.merge(bpm_parameters)}})
  else
    request_params.merge!({payload: parameters.merge(bpm_parameters)})
  end

  RestClient::Request.execute(request_params)
end

private

def bpm_parameters
  {
    user_identifier: bpm_config[:hbw_login],
    entity_type:     params[:entity_type],
    entity_code:     params[:entity_code],
    entity_class:    params[:entity_class]
  }
end

def build_bpm_widget_path(path = '')
  URI.join(bpm_config.url, '/widget/', path).to_s
end

def bpm_config
  YourApplication::Config.widgets.bpm
end
```

GET and DELETE parameters should be passed as query part of url, e.g.:

```
GET /widget/tasks?user_identifier=user@example.com&entity_type=&entity_code=ORD-
6&entity_class=billing_customer;
```

POST and PUT parameters should be passed as body part of request. **Request body's `content-type` is `application/json`.**

HydraOMS answer `content-type` is `application/json`.

Proxy controllers

Proxy controllers are used as an middleware for requests from third-party application to HydraOMS. They check data send by widget and pass it to HydraOMS backend with adding auth headers and other necessary information.

```
# ANY match 'widget/*path'
def proxy
  method = request.method.downcase.to_sym
  result = request_bpm_backend(params[:path], method, permitted_params)

  if method == :put
    if result
      head :no_content
    else
      head :bad_request
    end
  else
    render json: result
  end
end

private

def permitted_params
  params.symbolize_keys.except(*service_params)
end

def service_params
  %i[controller action path format]
end
```

Embedding JS

Here you can find an example of embedding HydraOMS widget into HTML page of third-party application. After external JS and CSS are loaded, HydraOMS widget will be initialized with application and entity data and then `render()` function is being called.

HTML page example

```
<html>
<head>
  <title>HydraOMS Widget</title>
  <script type="text/javascript" src="${hbw_public_path}/assets/hbw.js"></script>
  <link rel="stylesheet" type="text/css" href="${hbw_public_path}/assets/hbw.css">
</head>
<body>
  <div class="hbw-styles">
   <div id='hbw-container'>
   </div>
  </div>
  <script type="text/javascript">
    var config = {
      widgetURL: 'https://homs.some.domain', // hbw_public_path, important for WebSocket connection
      entity_class: 'crm_account',
      entity_type: 'customer',
      container_id: 'hbw-container', // Same as <div> id
      userIdentifier: 'user@example.com', // user identifier for Web-Socket connection
      locale: {
        code: 'en', // locale code
        dateTimeFormat: 'MM/DD/YYYY HH:mm aaa' // date-fns format, see https://date-fns.org/v1.30.1/docs/format
for more details
      }
    };
    var entityId = ...; // Set here id or other uniq value of entity, like customerId

    window.hbw_widget = new (modulejs.require('HBW'))({
      userIdentifier: config.userIdentifier,
      widgetContainer: `#${config.container_id}`,
      widgetURL: config.widgetURL,
      widgetPath: '/widget',
      entity_class: config.entity_class,
      entity_type: config.entity_type,
      entity_code: `${entityId}`,
      locale: config.locale,
      payload: {
        variables: {
          someInitialVariable: { // You can pass other useful information to process initial_variables
            value: 'initialValue',
            type: 'string'
          }
        }
      }
    });

    window.hbw_widget.render();

    // If you use some kind ot SPA (Single Page Application), call this before exiting current page:
    // window.hbw_widget.unmountWidget();
  </script>
</body>
</html>
```