

Building an Integrated Solution

Guideline to Integrate the Orbit Web Client

Here is the guideline to build and layout your host-integrated solution. This will help you to understand the subsequent steps to be taken to engage the SDK, and the user interface options that you can provide. Develop your solution according to the standards for user interaction that are used in your host software.

The Orbit Web Client gives access to 3 components, which may or may not be available according to server settings and license of the plugin at hand :

- Map component
- Mobile Mapping component
- Obliques component

Next to that, there are common services available.

1. Getting Access

First make the Orbit Web Client available in your host. If your host provides a Map component, you most probably wish to hide that part from the Orbit Web Client. There is an API method to do so. These chapters assume that you need to integrate your host's map component with the Orbit Web Client Mobile Mapping or Obliques viewing capabilities, instead of using the Orbit Web Client map component.

The Orbit Web Client needs to connect to an Orbit Publisher, either Mobile Mapping or Obliques. The Mobile Mapping and Obliques Publisher server extensions provide the Orbit Server administrator the necessary tools to configure data setups, access permissions, startup parameters and more. Such a setup is called a "**Publication**". A user may have access to more than 1 Publication.

So the first thing to do is to get the client connected. You will need the following parameters:

- Server URL
- User Name
- Password
- Publication

The login procedure should follow these steps, for which several API methods are available :

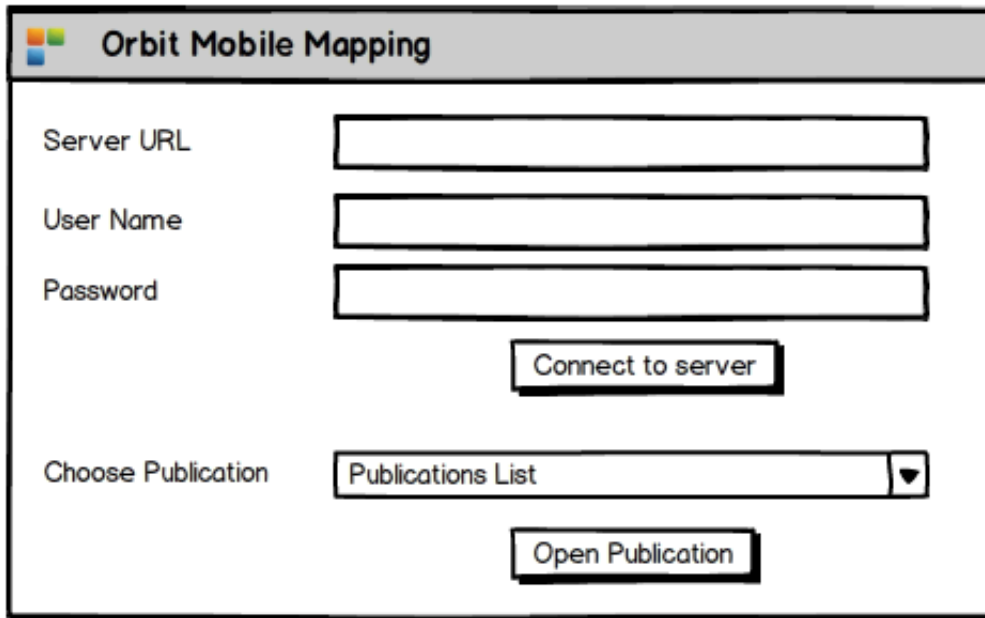
1. Request user input for server URL, and the login credentials : user name and password (see GUI example below)
2. Login to the server using URL, user name and password. A list of accessible Publications is returned.
3. Request the Publications and present these to the user.
4. The user chooses a Publication, then opens it.

You may choose to keep the panorama viewer blank and await further user action. This is typically done for host-extensions where a map viewing is already present.

You may also choose to open a Publication with a default first panoramic image. To do so, request the startup parameters for this publication, and use these to load a first panorama. The startup parameters are set by the administrator on the server. This is typically done for web-based implementations where no pre-defined map location is available.

The API allows to log in with or without the choice of a Publication. If without, you then need to request the list of Publications and present it to the user for choosing. To continue, a Publication must be chosen to view data. If there's only one Publication available to the user, you may choose to open that publication automatically without presenting a choice to the user. If the users wishes to switch between Publications, he can e.g. return to a settings window as shown below, make a different choice, and click "Open Publication".

GUI : provide a tool, configuration or window to setup and store these parameters for reuse. The user will most probably reuse the login credentials several times, if not permanent. Here's an example for Mobile Mapping, equally valid for Obliques :



The screenshot shows a window titled "Orbit Mobile Mapping". It contains the following elements:

- Server URL: A text input field.
- User Name: A text input field.
- Password: A text input field.
- Connect to server: A button.
- Choose Publication: A dropdown menu with "Publications List" selected.
- Open Publication: A button.

Make sure you can save these credentials or keep a history for easy retrieval. Every user most probably wants to continue the next day with the same server, login and publication settings.

Note that you can only show a list of Publications after a successful 'Connect To Server'. Only then a Publication choice and Login can be executed. Keep these both items dimmed until a successful connection is established.

2. Getting started with Mobile Mapping

A panorama needs to be *selected* to allow visualisation. In order to do so, the GUI needs to provide a way to allow the user to select a panorama.

2.1. Presenting Photo Recording Positions on the Map

First thing is to visualise the panorama photo positions on the map. Use a method to request these positions : this request is limited by a bounding box as it is not feasible to request all. Most commonly, request the photo positions with a bounding box covering the current map display, but only from a *reasonable* scale on (e.g. use 1/5 000 as a threshold scale). When zooming out to a smaller scale than the threshold scale, do not request or display these positions anymore as it most probably will cause sluggish performance and cluttered viewing. Note : When zooming out, you may want to display the *coverage* instead of the individual photo positions (available from 11.0). This represents a simplified representation of the locations where pano's are available. Here too, limit the viewing to a scale of e.g. 1/50 000)

Make the visualization of the panorama photo positions as automated as possible. There is no need for the user to select an area : automatically refresh the photo positions after every pan or zoom action so that all relevant information is always present.

Finally, be smart in your requests : cache the photo positions if you can to avoid unnecessary overhead and lower performance.

Display the photo positions on your map view as a layer. Use a nice symbol or icon : a slightly transparent blue circular dot is common use. Here are some examples to use for free.

standard icons : 

icons with shadow : 

You may provide a tool to switch the viewing of these photo positions on and off.

Display a Coverage as a wide, transparent line.

These new photo positions objects in your map view do not need to be selectable by the standard tools from your host system. They only need to be selectable by a specific 'select panorama' method as described below. Depending on the standards for user interaction in your host system, set this as pleased.

Finally, show these photo positions in the correct color associated with the pano's.

2.2. Selecting a panorama

Provide a function that targets panorama selection. The user clicks on a panorama photo position representation on the map view. Select that object. You now know the co-ordinates (X, Y) and an ID (attribute). Using this, use the appropriate method to load and view a panorama. the Orbit Web Client will open the image and you can now pan, zoom and navigate.

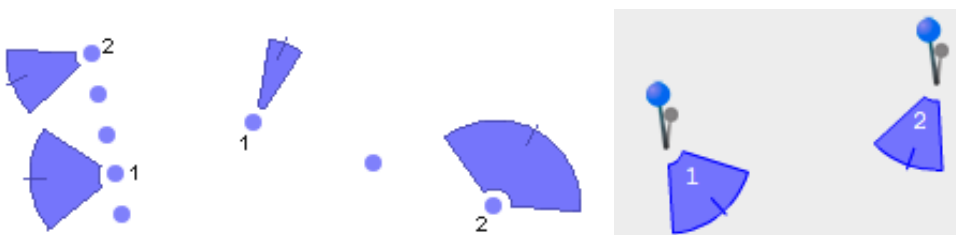
Smart selection methods

Users may click just next to a position : you can ask the Orbit Web Client to load the nearest panorama by providing the clicked co-ordinate. You may also ask the Orbit Web Client to load the 2 or 3 nearest panorama's. Alternatively, you may integrate an address search tool which at its turn results in a co-ordinate, replacing the manual point-and-click.

Giving proper user feedback

You can help the user a lot when you represent the current panorama viewing direction and viewing angle on the 2D map. You can do so by listening to the State events that the Orbit Web Client sends, and draw a pie chart slice on the map, representing the direction (center direction of the slice) and field of view angle (width of the slice).

If multiple panoramas are opened, do this for every panorama. Remember to also display the viewer number in the according slice. Here are some examples :



You can download [here](#) some example code to render this type of pie slices. Do render these pie slices in the according color.

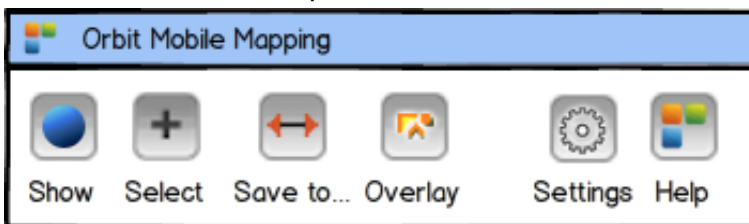
Managing multiple panoramas

The Orbit Web Client can open up to 4 panoramas simultaneously. In the Orbit Web Client, the user can close a viewer (hover the panorama number to see the “x”-button in the top left corner of each panorama) or select an extra one (click the “+”-button next to the number). In your code, you may thus need to manage an array of panoramaviewers and keep track of each of them. Note that, when selecting a second panorama, the first panorama remains unchanged.

GUI : You may wish to provide a toolbar offering several dedicated functions for the panorama viewer. A first one will most probably be the Settings for login. A second tool can be this selection functionality. An old-fashioned toolbar example :



A Ribbon toolbar example :



These are the used icons :



Show Photo Positions



Select a Panorama



Save a Measurement as new Feature



Manage Overlays or Show a Layer as Overlay on the Panorama



Login and other Settings



Knowledge Base Help (URL link)

You can download these icons here : 16x16 px or 32x32 px

2.3. Overlay Vector Data

There are 2 ways to overlay vector data in panoramas :

- include vector layers in the Publication (requires declaration of the data on server level, by the server administrator)
- send data over from your host application to the panorama viewer to display on the fly (using the API).

Layers that are declared in the Publication can be controlled by the tools within the panorama viewer. You can also use the API to switch visibility on/off for each of them. Note that in such case, you'll need to have a GUI for that in your host application. (That would probably be overhead as this on/off switching is available in the Orbit Web Client.)

Overlaying host vector data

You can create a vector layer in the Orbit Web Client, propagate it with objects and set appearance parameters. To do so, you'll need to cover the following steps :

1. Determine the data from your host that you wish to overlay
2. push that data over to the Orbit web Client using the API
3. Set the graphic appearance such as color of symbol
4. Control its visibility in the Orbit Web Client, either via the built in HUD or using the API (see above).

Depending on the standards of user interaction in your host system, allow the user to select a portion of the vector data that he would wish to see overlayed in the panorama, or simply select a layer and do the rest automatically (preferred). There are different ways to do that :

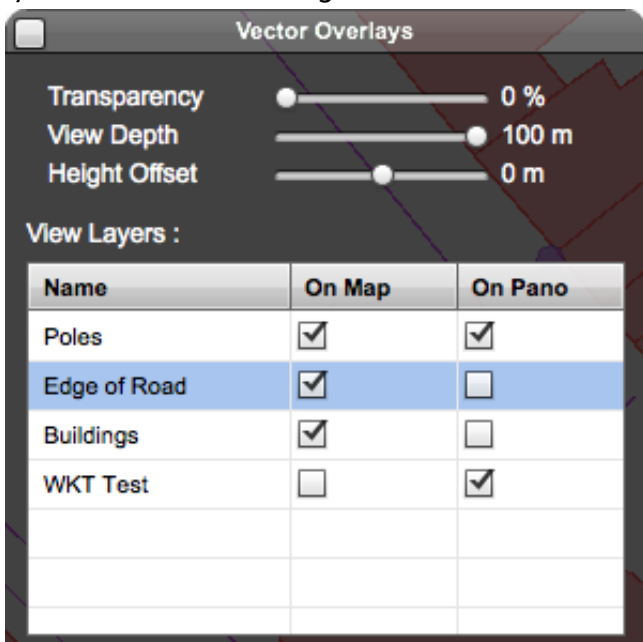
- select (a range of) specific objects
- select a query result
- automate the selection by using a distance-from-viewpoint within a selected layer or all visible layers. (Preferred option)

The latter option skips the physical selection of objects, so the user only needs to choose a layer. The objects themselves are determined by their distance from the photo position. Posting more objects to the Orbit Web Client will not result in improved visibility, but in reduced performance and clutter along the horizon in the panorama.

In a panorama, it is irrelevant to display a complete layer. Only the objects nearest to the photo position are useful to the user. In the Orbit Web Client, this is called the "View Depth" presented in the Vector Overlay HUD as a slider setting that can be controlled by the user.

GUI : provide a panel or other UI widget with a list of layers that can be overlaid in your host. Provide a simple toggle to switch its visibility in the Orbit Web Client on and off.

For Example, this is the HUD standard available within the Orbit web client to switch layers on/off; It only lists the layers known by the Orbit web client, which are not all the layers known by your host system that the user might like to switch on for viewing in the panorama :



Keeping overlays up to date

Preferable, implement a class that keeps track of each panorama and the layers that are overlaid. Post

just the necessary amount of data (as explained above – a radius from the photo position) on the Orbit Web Client. When the choice of panorama image changes, you will need to refresh the overlay data as well. You can do so by replacing the objects that were posted to the Orbit Web Client to a new set of objects that are visible around the new panorama photo position. Remember to clear or remove the layer before re-posting data.

About 2D data

Many vector data is 2D. A panorama view is in 3D. The absence of a Z co-ordinate forces an as-good-as-possible assumption. The Orbit Web Client panorama viewer will calculate a Z-value equal to the approximate ground level. That ground level is calculated down from the distance of the camera above ground. The overlaid data is thus projected on a fixed height, which may not always be effective. To adjust the height of a 2D layer, use the panorama Vector Overlay HUD.

If you wish the Orbit Web Client to accept the data as 2D, do not include a Z co-ordinate when you post the data. As soon as a Z coordinate is detected, Orbit will treat the data as correct 3D data and disable the height adjustment in the HUD for that layer.

2.4. Measurements

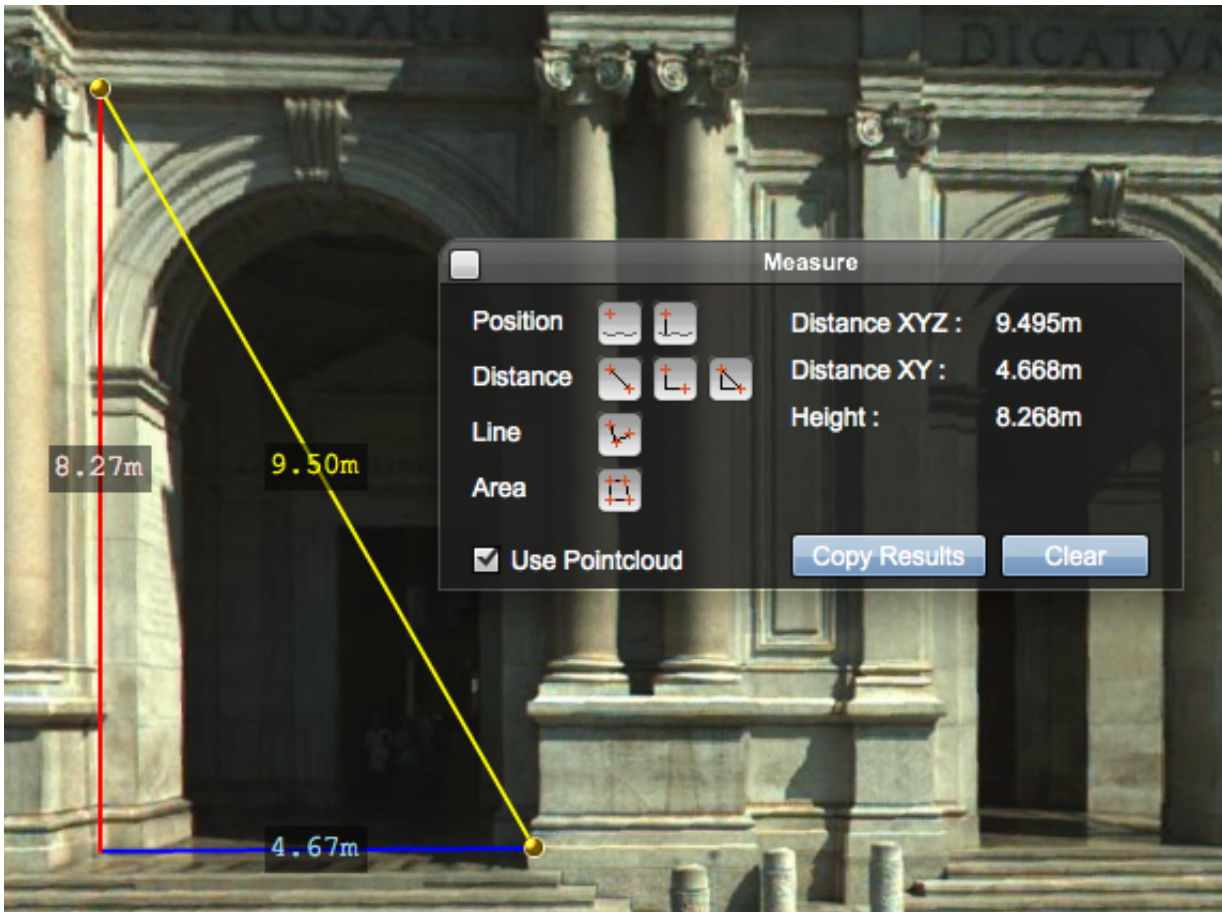
The Orbit Web Client provides a HUD panel with all measurement functions. The user can simply open this panel and execute any of the measurements without any development from your side. The measurement results are displayed in the same HUD and can be copied to the clipboard.

Integrating Measurements

When a measurement function is started, the Orbit Web Client launches an event that can be monitored. Equally, when a measurement function is stopped, an event is launched. You can use these events to follow the progress of the measurement, and when stopped request all the results. Depending on the type of measurements, the list of result values differs (see API documentation).

Alternatively, you can launch any of the measurement functions from within your host software. In combination with the events, you can skip the HUD completely and display the results in your host.

Measurement results can be used to create new objects : a point, line or area measurement includes the list of co-ordinates as part of the result set. You can implement a function in your host system that allows the user to add a measurement as a new object in a chosen layer. The API provides the geometry type (point, line area) so you can present only those layers that qualify for accepting the measurement as new object. You may also implement to add measurement results data as attribute values.



2.5. Snapshots

To create a snapshot of the current panorama, the user can click the “snapshot” button.

If the host component wishes to do something with these snapshots, the host component can listen to snapshot update events, retrieve the snapshot image and do something with it. A snapshot update event is triggered every time the user creates a snapshot.

If the host component does nothing snapshot-related, the snapshot buttons should be removed from the panoviewer toolbar using an API method.

3. Getting Started with Obliques

When visualizing oblique imagery, a focus position is used to focus all viewers on a single image detail. When the focus coordinate changes, the best overlapping obliques are loaded automatically by the client.

3.1. Tracking the Focus Coordinate

Make sure that every time the map is recentered, the focus coordinate is also updated to the new map center, so the oblique imagery is updated.

You may choose to display the focus co-ordinate using a small cross icon in the center of the map :



3.2. Viewer Layout

Based on the integration context, the best viewer layout should be selected : horizontal, vertical or tiled.

3.3. Scale

Only update the focus coordinate when it makes sense to do so. When the user is zoomed too far out, it makes no sense to keep the oblique viewers centered on a detail that is not discernible anymore. By default the web client use 1/5.000 as a threshold scale.

3.4. Following Focus

To analyze an oblique photo more closely, the user can enable the follow-focus mode. When follow-focus is enabled, the user changes the focus coordinate by tracking mouse clicks & drags. Whenever the user clicks or drags, the focus coordinate is updated to that coordinate. As a result, all other oblique viewers are recentered.

The host component should behave in a similar fashion and update the focus coordinate when the user clicks or drags. When updating the focus coordinate multiple times in quick succession (like during a mouse drag) the host component should use the “approximate” flag when updating the focus coordinate. This flag tells the component that a full coordinate transformation (including a round-trip to the server) is not wanted. Instead an approximate conversion is attempted. After the drag don't forget to update the focus coordinate without approximation to get an accurate, final result.

The web client can make approximate transformations between the Publication CRS and the Oblique Project CRS because it can make an approximation of the scale factor between both systems. Therefore approximation can only work when you specify a coordinate in one of these systems.

3.5. Following Zoom

To apply zoom changes in all viewers at once, the user can enable the follow-zoom mode. Whenever the zoom factor changes, all other oblique viewers zoom with an equal measure.

The host component should update the zoom factors for all oblique viewers, when updating scale.

3.6. Snapshots

To create a snapshot of the current oblique view, the user can click the “snapshot” button.

If the host component wishes to do something with these snapshots, the host component can listen to snapshot update events, retrieve the snapshot image and do something with it. A snapshot update event is triggered every time the user creates a snapshot.

If the host component does nothing snapshot-related, the snapshot buttons should be removed from the oblique toolbars using an API method.

Logged in as: Bonne Peter (access,orbit,administrator,orbitAdministrator)