Program Logic Support

Persist Functionalities

Persisting a beanshell object to store the value of the object name when the application is destroyed and restore the value when the application is restored

- **name** the name of the object

persistObject(String name) ;

Tab / Tab Group Functionalities

Showing the tab group with the label and clear all values if there was any value before in the tab group

- **label** the label of the tab group

newTabGroup(String label) ;

Showing the tab with the label and clear all values if there was any value before in the tab

- **label** the label of the tab

newTab(String label) ;

Showing the tab group with the label and keep all values if there was any value before in the tab group

- **label** the label of the tab group

showTabGroup(String label);

Showing the tab group with the label and load values for a given uuid to the tab group

- **id** the reference of the tab group
- **uuid** the id of the record

showTabGroup(String id, String uuid);

Showing the tab with the label and keep all values if there was any value before in the tab

- **label** the label of the tab

showTab(String label) ;

Showing the tab with the label and load values for a given uuid to the tab

- **id** the reference of the tab
- **uuid** the id of the record

showTab(String id, String uuid) ;

Cancelling tab group will close the tab group. It will only give warning if the warning is set to be true and if there is any changes in any field in the tab group

- **id** the reference of the tab group
- **warn** a boolean to set whether a warning dialog should appear or not

cancelTabGroup(String id, boolean warn);

Cancelling tab will close the tab. It will only give warning if the warning is set to be true and if there is any changes in any field in the tab

- **id** the reference of the tab
- **warn** a boolean to set whether a warning dialog should appear or not

cancelTab(String id, boolean warn);

Dialog Functionalities

Showing a toast to the user with the given message, the toast will last for about 1 second

- **message** the message to be shown to the user

showToast(String message) ;

Showing an alert dialog to the user with the given message and a specific callback for ok and cancel buttons

- **title** the title of the dialog
- **message** the message to be shown to the user
- **okCallback** the callback that is executed when OK button is pressed
- **cancelCallback** the callback that is executed when cancel button is pressed

showAlert(String title, String message, String okCallback, String cancelCallback);

Showing a warning dialog to the user with the given message

- **title** the title of the dialog
- **message** the message to be shown to the user

showWarning(String title, String message) ;

Showing a busy dialog to the user with the given message

- **title** the title of the dialog
- **message** the message to be shown to the user

showBusy(String title, String message) ;

Setter / Getter Functionalities

Set value to the field that has the reference with given value. If the field reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the field
- **value** the value to be set to the field, could be a collection, number, or String

setFieldValue(String ref, Object value);

Set certainty to the field that has the reference with given value. If the field reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the field

- **value** the value to be set to the field, could be a number or String

setFieldCertainty(String ref, Object value);

Set annotation to the field that has the reference with given value. If the field reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the field
- **value** the value to be set to the field, only accept String

setFieldAnnotation(String ref, Object value);

Get value of the field that has the reference. If the field reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the field

return the value of the field, could be collection or String

Object getFieldValue(String ref);

Get certainty of the field that has the reference. If the field reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the field

return the certainty of the field in String representation

Object getFieldCertainty(String ref);

Get annotation of the field that has the reference. If the field reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the field

return the annotation of the field in String representation

Object getFieldAnnotation(String ref);

Get the current time of the application

return the current time in String representation

String getCurrentTime();

Clear the dirty button from the field that has the reference. If the field reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the field

clearFieldDirty(String ref) ;

Event Callback Functionalities

Binding an event to the field that has the reference with a callback. If the field reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the field
- **type** the type of event, one of click, show, or load
- **callback** the callback that is executed when the binded field is on the set event

onEvent(String ref, String type, String callback) ;

Binding a focus/blur event to the field that has the reference with a callback. If the field reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the field
- **focusCallback** the callback that is executed when the field is on focus event
- **blurCallback** the callback that is executed when the field is on blur event

onFocus(String ref, String focusCallback, String blurCallback) ;

User Functionalities

Set the current user of the application, used to determine who creates the record.

- **user** the user of the application

setUser(User user);

Arch Ent / Relationship Functionalities

Saving arch entity into the database and return the entity_id of the saved entity

- **entity_id** the id of the arch entity to be saved, might be null to save new entity
- **entity_type** the type of the entity to be saved, must be coherent with the type specified in data schema
- **geo_data** the list of geometries to be associated with the saved entity
- **attributes** the list of attributes to be associated with the saved entity

return the entity_id of the saved entity

saveArchEnt(String entity_id, String entity_type, List geo_data, List attributes) ;

Saving relationship into the database and return the rel_id of the saved relationship

- **rel_id** the id of the relationship to be saved, might be null to save new relationship
- **rel_type** the type of the relationship to be saved, must be coherent with the type specified in data schema
- **geo_data** the list of geometries to be associated with the saved relationship
- **attributes** the list of attributes to be associated with the saved relationship

return the rel_id of the saved relationship

saveRel(String rel_id, String rel_type, List geo_data, List attributes) ;

Delete an arch entity from the database

- **entity_id** the id of the arch entity to be deleted

return boolean value true if deleted, false if not

deleteArchEnt(String entity_id);

Delete a relationship from the database

- **rel_id** the id of the relationship to be deleted

return boolean value true if deleted, false if not

deleteRel(String rel_id);

Add an arch entity to the relationship by specifying the verb as the relation.

- **entity_id** the id of the arch entity to be associated with relationship
- **rel_id** the id of the relationship to be associated to
- **verb** the relation that is defined in the dataschema

addReln(String entity_id, String rel_id, String verb) ;

Create attribute list to be saved to the entity/relationship

return new attribute list

createAttributeList() ;

Create entity attribute to be added to the attribute list

- **name** the name of the attribute, should be coherent to the attribute name specified in the schema to be saved
- **text** the text of the entity attribute, could be null
- **vocab** the vocab of the entity attribute (obtained from the database), could be null
- **measure** the measure of the entity attribute, could be null
- **certainty** the certainty of the entity attribute, defaulted to 100% certainty

return EntityAttribute

createEntityAttribute(String name, String text, String vocab, String measure, String certainty);

Create entity attribute to be added to the attribute list

- **name** the name of the attribute, should be coherent to the attribute name specified in the schema to be saved
- **text** the text of the entity attribute, could be null
- **vocab** the vocab of the entity attribute (obtained from the database), could be null
- **measure** the measure of the entity attribute, could be null
- **certainty** the certainty of the entity attribute, defaulted to 100% certainty
- **isDeleted** set whether the entity attribute is deleted or not

return EntityAttribute

createEntityAttribute(String name, String text, String vocab, String measure, String certainty, boolean isDeleted) ;

Create relationship attribute to be added to the attribute list

- **name** the name of the attribute, should be coherent to the attribute name specified in the schema to be saved
- **text** the text of the relationship attribute, could be null
- **vocab** the vocab of the relationship attribute (obtained from the database), could be null
- **certainty** the certainty of the relationship attribute, defaulted to 100% certainty

return RelationshipAttribute

createRelationshipAttribute(String name, String text, String vocab, String certainty);

Create relationship attribute to be added to the attribute list

- **name** the name of the attribute, should be coherent to the attribute name specified in the schema to be saved
- **text** the text of the relationship attribute, could be null
- **vocab** the vocab of the relationship attribute (obtained from the database), could be null
- **certainty** the certainty of the relationship attribute, defaulted to 100% certainty
- **isDeleted** set whether the relationship attribute is deleted or not

return RelationshipAttribute

createRelationshipAttribute(String name, String text, String vocab, String certainty, boolean isDeleted) ;

Populate dropdown from values to the field that has the reference. If the field reference is not found or wrong type of field, there will be a logic error dialog appearing.

- **ref** the reference to the field
- **values** the collection of values to populate the dropdown

populateDropDown(String ref, Collection values);

Populate radio group from values to the field that has the reference. If the field reference is not found or wrong type of field, there will be a logic error dialog appearing.

- **ref** the reference to the field
- **values** the collection of values to populate the radio group

populateRadioGroup(String ref, Collection values) ;

Populate checkbox from values to the field that has the reference. If the field reference is not found or wrong type of field, there will be a logic error dialog appearing.

- **ref** the reference to the field
- **values** the collection of values to populate the checkbox

populateCheckBoxGroup(String ref, Collection values) ;

Populate list from values to the field that has the reference. If the field reference is not found or wrong type of field, there will be a logic error dialog appearing.

- **ref** the reference to the field
- **values** the collection of values to populate the list

populateList(String ref, Collection values);

Populate picture gallery from values to the field that has the reference. If the field reference is not found or wrong type of field, there will be a logic error dialog appearing.

- **ref** the reference to the field
- **values** the collection of values to populate the picture gallery

populatePictureGallery(String ref, Collection values);

Populate camera picture gallery from values to the field that has the reference. If the field reference is not found or wrong type of field, there will be a logic error dialog appearing. This is usually used to show picture after taking picture from camera.

- **ref** the reference to the field
- **values** the collection of values to populate the camera picture gallery

populateCameraPictureGallery(String ref, Collection values);

Populate video gallery from values to the field that has the reference. If the field reference is not found or wrong type of field, there will be a logic error dialog appearing. This is usually used to show video after recording video.

- **ref** the reference to the field

- **values** the collection of values to populate the camera picture gallery

populateVideoGallery(String ref, Collection values);

Populate audio list from values to the field that has the reference. If the field reference is not found or wrong type of field, there will be a logic error dialog appearing. This is usually used to show audio after recording audio.

- **ref** the reference to the field
- **values** the collection of values to populate the camera picture gallery

populateAudioList(String ref, Collection values);

Fetching the data of arch entity with the given id, if the id is not found, a logic error dialog will appear

- **id** the entity id of the arch entity

return ArchEntity object or null

Object fetchArchEnt(String id);

Fetching the data of relationship with the given id, if the id is not found, a logic error dialog will appear

- **id** the entity id of the relationship

return Relationship object or null

Object fetchRel(String id);

Fetching the data from the database by running query specified by user. It will only return one result.

- **query** the query to be run against the database

return Collection of String, might be empty

Object fetchOne(String query);

Fetching the data from the database by running query specified by user. It will return all result.

- **query** the query to be run against the database

return Collection of Collection of String, might be empty

Collection fetchAll(String query);

Fetching the list of arch entity to show to the user so the user can see what entities have been saved.

- **type** the type of arch entity to be shown

return Collection of Collection of String, might be empty

Collection fetchEntityList(String type);

Fetching the list of relationship to show to the user so the user can see what entities have been saved.

- **type** the type of relationship to be shown

return Collection of Collection of String, might be empty

Collection fetchRelationshipList(String type);

Get the selected value from the list to be used in the logic

return _list_item_value of the selected value in the list

String getListItemValue() ;

Navigation Functionalities

Provide functionality to go back as if the user press the hardware back button

goBack();

GPS Functionalities

Set the GPS update interval to determine how often the GPS should update, defaulted to 10 seconds

- **seconds** the interval to be set

setGPSUpdateInterval(int seconds);

Start using the internal GPS to update the location

startInternalGPS();

Start using the external GPS to update the location, it will bring up dialog to the user to choose which bluetooth device to be used as external GPS

startExternalGPS();

Get the GPS position containing longitude and latitude from external GPS or internal GPS

return GPSPosition if starting GPS or null if no GPS started or position found

Object getGPSPosition();

Get the GPS position containing longitude and latitude with the projection selected from external GPS or internal GPS

return projected GPSPosition if starting GPS or null if no GPS started or position found

Object getGPSPositionProjected() ;

Get the GPS accuracy from external GPS or internal GPS

return accuracy of the gps or null

Object getGPSEstimatedAccuracy();

Get the GPS heading from external GPS or internal GPS

return heading of the gps or null

Object getGPSHeading();

Get the GPS position containing longitude and latitude from selected GPS

- **gps** the type of gps used, one of "internal" or "external"

return GPSPosition if starting GPS or null if no GPS started or position found

Object getGPSPosition(String gps);

Get the GPS accuracy from selected GPS

- **gps** the type of gps used, one of "internal" or "external"

return accuracy of the selected gps or null

Object getGPSEstimatedAccuracy(String gps);

Get the GPS heading from selected GPS

- **gps** the type of gps used, one of "internal" or "external"

return heading of the selected gps or null

Object getGPSHeading(String gps);

Start track log of the GPS and save it to the database so the user can look at the track log in the map view

- **type** either "distance" or "time"
- **value** the value of the tracking, if type is "distance", the value will be in meter, if type is "time", the value will be in seconds

startTrackingGPS(String type, int value);

Stopping the track log of the GPS

stopTrackingGPS();

Map Functionalities

Binding map click event and vector click event to the map view that has the reference. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **clickCallback** the call back that is executed when the map view is clicked
- **selectCallback** the call back that is executed when the vector element is clicked

onMapEvent(String ref, String clickCallback, selectCallback) ;

Show base map to the map view that has the reference. If the map view reference is not found, there will be a logic error dialog appearing. the map view will be the base map.

- **ref** the reference to the map view
- **layerName** the layer name for the raster map
- **filename** the filename of the map view, will show error if the file does not exist

showBaseMap(String ref, String layerName, String filename) ;

Show raster map to the map view that has the reference. If the map view reference is not found, there will be a logic error dialog appearing. You have include multiple raster maps.

- **ref** the reference to the map view
- **layerName** the layer name for the raster map
- **filename** the filename of the map view, will show error if the file does not exist

showRasterMap(String ref, String layerName, String filename) ;

Set the focus point of the map view that has the reference by specifying the float value of longitude and latitude. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **longitude** the longitude of the focus point
- **latitude** the latitude of the focus point

setMapFocusPoint(String ref, float longitude, float latitude) ;

Set the focus point of the map view that has the reference by specifying the double value of longitude and latitude. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **longitude** the longitude of the focus point
- **latitude** the latitude of the focus point

setMapFocusPoint(String ref, double longitude, double latitude) ;

Set the rotation of the map view that has the reference by specifying the rotation value. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **rotation** the rotation of the map view in degrees

setMapRotation(String ref, float rotation) ;

Set the zoom level of the map view that has the reference by specifying the zoom value. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **zoom** the zoom value of the map view

setMapZoom(String ref, float zoom) ;

Set the perspective view of the map view that has the reference. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **tilt** the tilt value default is 90 degrees for 2d view, minimum is 30 degrees

setMapTilt(String ref, float tilt) ;

Show a shape layer to the map view that has the reference. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **layerName** the layer name for the shape layer

- **filename** the file path of the shape map, will show error if the file does not exist
- **pointStyle** the styling to points appearing in the layer
- **lineStyle** the styling to lines appearing in the layer
- **polygonStyle** the styling to polygons appearing in the layer

showShapeLayer(String ref, String layerName, String filename, GeometryStyle pointStyle, GeometryStyle lineStyle, GeometryStyle polygonStyle) ;

Show a spatial layer to the map view that has the reference. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **layerName** the layer name for the shape layer
- **filename** the file path of the shape map, will show error if the file does not exist
- **tableName** the table name to be loaded from the database
- **idColumn** the id column from the table to be loaded from the database
- **labelColumn** the label column from the table to be loaded from the database to be shown as the label
- **pointStyle** the styling to points appearing in the layer
- **lineStyle** the styling to lines appearing in the layer
- **polygonStyle** the styling to polygons appearing in the layer
- **textStyle** the styling to all text labels appearing in the layer

showSpatialLayer(String ref, String layerName, String filename, String tableName, String idColumn, String labelColumn, GeometryStyle pointStyle, GeometryStyle lineStyle, GeometryStyle polygonStyle, GeometryTextStyle textStyle) ;

Show a database layer to the map view that has the reference. If the map view reference is not found, there will be a logic error dialog appearing. The layer shows all saved geometries in the database.

- **ref** the reference to the map view
- **layerName** the layer name for the shape layer
- **isEntity** a boolean value to determine whether it is entity or relationship
- **queryName** the query name for the executed query
- **querySql** the sql to be executed for the layer
- **pointStyle** the styling to points appearing in the layer
- **lineStyle** the styling to lines appearing in the layer
- **polygonStyle** the styling to polygons appearing in the layer
- **textStyle** the styling to all text labels appearing in the layer

showDatabaseLayer(String ref, String layerName, boolean isEntity, String queryName, String querySql,GeometryStyle pointStyle, GeometryStyle lineStyle, GeometryStyle polygonStyle, GeometryTextStyle textStyle) ;

Remove a layer from the map view that has the reference. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **layerId** the layerid to be removed, if not found, a logic error will appear.

removeLayer(String ref, int layerId) ;

Centering the map view that has the reference based on the current GPS position if there is GPS position. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view

centerOnCurrentPosition(String ref);

Create canvas layer and add it to the map view that has the reference. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **layerName** the name of the canvas layer

createCanvasLayer(String ref, String layerName) ;

Set the visibility of a layer in the map view that has the reference. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **layerId** the layerId to set visibility, will show logic error dialog if not found
- **visible** the boolean to set whether it is visible or not

setLayerVisible(String ref, int layerId, boolean visible) ;

Get the clicked position in the map view

return the clicked map point

getMapPointClicked() ;

Draw a point on the map view that has the reference by specifying the point and style. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **layerId** the layerId to draw the point to, will show logic error dialog if not found
- **point** the map view point to be drawn
- **style** the style that will be applied to the point

drawPoint(String ref, int layerId, MapPos point, GeometryStyle style) ;

Draw a line on the map view that has the reference by specifying the points and style. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **layerId** the layerId to draw the point to, will show logic error dialog if not found
- **points** the map view points to be drawn as a line
- **style** the style that will be applied to the line

drawLine(String ref, int layerId, List points, GeometryStyle style) ;

Draw a polygon on the map view that has the reference by specifying the points and style. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **layerId** the layerId to draw the point to, will show logic error dialog if not found
- **points** the map view points to be drawn as polygon
- **style** the style that will be applied to the polygon

drawPolygon(String ref, int layerId, List points, GeometryStyle style) ;

Clear a geometry from the map view that has the reference by specifying the geometry id. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **geomId** the id of the geometry to be cleared, will show logic error dialog if not found

clearGeometry(String ref, int geomId) ;

Clear a list of geometries from the map view that has the reference by specifying the list of geometry id. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **geomList** the list of id of the geometries to be cleared, will show logic error dialog if not found

clearGeometryList(String ref, List geomList) ;

Create a point from string longitude and string latitude

- **lon** the longitude of the point in String format
- **lat** the latitude of the point in String format

return MapPos the new point object for the map view

createPoint(String lon, String lat) ;

Create a point from float longitude and float latitude

- **lon** the longitude of the point in float format
- **lat** the latitude of the point in float format

return MapPos the new point object for the map view

createPoint(float lon, float lat) ;

Create a point from double longitude and double latitude

- **lon** the longitude of the point in double format
- **lat** the latitude of the point in double format

return MapPos the new point object for the map view

createPoint(double lon, double lat) ;

Get all geometries in a specified layer from the map view that has the reference. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **layerId** the layerId to get the geometries from, will show logic error dialog if not found

return list of the geometries or null

getGeometryList(String ref, int layerId) ;

Get a geometry from the map view that has the reference by specifying the geometry id. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **geomId** the id of the geometry, will show logic error dialog if not found

return list of the geometries or null

getGeometry(String ref, int geomId) ;

Get the layer name of the associated geometry on a canvas layer @geomId the id of the geometry

- **ref** the reference to the map view

return the layer name of the canvas layer the geometry is on

getGeometryLayerName(String ref, int geomId) ;

Draw geometry on the map view that has the reference by specifying the geometry and style. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **layerId** the layerId to draw the geometry to, will show logic error dialog if not found
- **geom** the geometry to be drawn
- **style** the style that will be applied to the geometry

return the id of the created geometry

drawGeometry(String ref, int layerId, Geometry geom, GeometryStyle style) ;

Create styling for point by specifying the minZoom, color, size, and pickSize

- **minZoom** the minimum level of zoom for the point to show up
- **color** the color of the point
- **size** the size of the point, range from 0.0 - 1.0
- **pickSize** the picking size of the point, used for selecting the point, range from 0.0 - 1.0

return the new point style

createPointStyle(int minZoom, int color, float size, float pickSize) ;

Create styling for line by specifying the minZoom, color, width, pickWidth, and pointStyle

- **minZoom** the minimum level of zoom for the line to show up
- **color** the color of the line
- **width** the width of the line, range from 0.0 - 1.0
- **pickWidth** the picking width of the line, used for selecting the line, range from 0.0 - 1.0
- **pointStyle** the styling of the point for the line

return the new line style

createLineStyle(int minZoom, int color, float width, float pickWidth, GeometryStyle pointStyle) ;

Create styling for polygon by specifying the minZoom, color, and lineStyle

- **minZoom** the minimum level of zoom for the polygon to show up
- **color** the color of the polygon
- **lineStyle** the styling of the line for the polygon

return the new polygon style

createPolygonStyle(int minZoom, int color, GeometryStyle lineStyle) ;

Create styling for text label by specifying the minZoom, color, size, and font

- **minZoom** the minimum level of zoom for the text label to show up
- **color** the color of the text label
- **size** the size of the text label
- **font** the font of the text label

return the new text style

createTextStyle(int minZoom, int color, int size, android.graphics.Typeface font) ;

Unlock or locking the map view has the reference from moving, useful when editing geometries. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **lock** boolean value to set locking or unlocking

lockMapView(String ref, boolean lock) ;

Adding a geometry to the higlighted geometries list to the map view has the reference. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **geomId** the id of the geometry to be highlighted

addGeometryHighlight(String ref, int geomId) ;

Removing a geometry from the higlighted geometries list to the map view has the reference. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **geomId** the id of the geometry to be removed from highlighted list

removeGeometryHighlight(String ref, int geomId) ;

This locks highlights in place so that when doHighlightTransform is called the highlighted geometry will be transformed to their new location

- **ref** the reference to the map view

prepareHighlightTransform(String ref) ;

This unlocks highlights so that geometries that were locked using prepareHighlightTransform get transformed to their new position, rotation and scaling

- **ref** the reference to the map view

doHighlightTransform(String ref) ;

Clearing all higlighted geometries list from the map view has the reference. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view

clearGeometryHighlights(String ref) ;

Get all higlighted geometries list from the map view has the reference. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view

getGeometryHighlights(String ref) ;

Get the selected geometry on the map view

return the selected geometry

getMapGeometrySelected() ;

Adding a database layer query to the map view has the reference to show the saved geometries in the database. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **name** the name of the query added
- **sql** the query to be executed when selected

addDatabaseLayerQuery(String ref, String name, String sql) ;

Adding a tracklog layer query to the map view has the reference to show the GPS tracking for all users. If the map view reference is not found, there will be a logic error dialog appearing.

- **ref** the reference to the map view
- **name** the name of the query added
- **sql** the query to be executed when selected

addTrackLogLayerQuery(String ref, String name, String sql) ;

Adding a query builder to the map view has the reference to select geometries based on the query. If the map view reference is not found, there will be a logic error dialog appearing. The selection will only select the geometries in the database layer.

- **ref** the reference to the map view
- **name** the name of the query added
- **builder** the query builder to be executed when selected

addSelectQueryBuilder(String ref, String name, QueryBuilder builder) ;

Create a query builder for database select by providing the sql query

- **sql** the query to be executed when selected

createQueryBuilder(String sql) ;

Adding a query builder to the map view has the reference to select geometries based on the query. If the map view reference is not found, there will be a logic error dialog appearing. The selection will only select the geometries in the legacy layer.

- **ref** the reference to the map view
- **name** the name of the query added
- **dbPath** the path of the database file
- **tableName** the name of the table for the database executed against
- **builder** the query builder to be executed when selected

addLegacySelectQueryBuilder(String ref, String name, String dbPath, String tableName, QueryBuilder builder) ;

Create a legacy query builder for legacy select by providing the sql query

- **sql** the query to be executed when selected

createLegacyQueryBuilder(String sql) ;

Convert a map position from one projection to another projection

- **fromSrid** the projection to convert
- **toSrid** the projection to be converted to
- **p** the map position to be converted

convertFromProjToProj(String fromSrid, String toSrid, MapPos p) ;

Enable or disable map tools for the map view

- **ref** the reference to the map view
- **enabled** true or false to enable or disable map tools

setToolsEnabled(String ref, boolean enabled) ;

Add tool specific events (create or load). The create event is called when the create point, line or polygon tools generate their geometry The load event is called when the load data tool is used to select geometry @callback the callback code to execute

- **ref** the reference to the map view
- **type** the event type (create or load)

onToolEvent(String ref, String type, String callback) ;

This used with the tool creat event callback to get the created geometry id

getMapGeometryCreated() ;

This used with the tool load event callback to get the uuid of the selected geometry

getMapGeometryLoaded() ;

This used with the tool load event callback to get the type (entity or relationship) of the selected geometry

getMapGeometryLoadedType() ;

This refreshes the map layers. Useful if you have saved entities or relationships into the database and want to update the map layers.

- **reference** to the map view

refreshMap(String ref) ;

Sync Functionalities

Pushing a database to the server which is used for syncing, after finished, it will execute the callback

- **callback** the callback that will be executed when the operation finished

pushDatabaseToServer(String callback) ;

Pulling a database from the server which is used for syncing, after finished, it will execute the callback

- **callback** the callback that will be executed when the operation finished

pullDatabaseFromServer(String callback) ;

Set whether the sync should be enabled or not

- **value** boolean value to set the sync enabled or not

setSyncEnabled(boolean value) ;

An event listener for sync that will execute call back for start, success, and failure of the sync

- **startCallback** the callback that will be executed when sync is started
- **successCallback** the callback that will be executed when sync is finish
- **failureCallback** the callback that will be executed when sync is failing

onSyncEvent(String startCallback, String successCallback, String failureCallback) ;

Set the minimum interval for the sync to happen

- **value** the minimum interval for the sync to happen

setSyncMinInterval(float value) ;

Set the maximum interval for the sync to happen

- **value** the maximum interval for the sync to happen

setSyncMaxInterval(float value) ;

Set the delay interval for each sync

- **value** the delay interval for each sync

setSyncDelay(float value) ;

Set whether the file sync should be enabled or not

- **value** boolean value to set the file sync enabled or not

setFileSyncEnabled(boolean enabled) ;

Static Data Functionalities

Get the static data for current project name

return project name

String getProjectName();

Get the static data for current project projection

return project projection

String getProjectSrid();

Get the static data for current project id

return project id

String getProjectId();

Get the static data for current project season

return project season

String getProjectSeason();

Get the static data for current project description

return project description

String getProjectDescription();

Get the static data for current permit number

return permit number

String getPermitNo();

Get the static data for current permit holder

return permit holder

String getPermitHolder();

Get the static data for current contact and address

return contact and address

String getContactAndAddress();

Get the static data for current participants

return participants

String getParticipants();

File attachment Functionalities

Get the name of last selected file from the file browser

return name of last selected file

getLastSelectedFilename() ;

Get the path of last selected file from the file browser

return path of last selected file

getLastSelectedFilepath() ;

Show the file browser and execute the callback after finish

- **callback** the callback to be executed once the operation finished

showFileBrowser(String callback) ;

Attach a file to the project folder by specifying the path of the file.If sync is true, the file will be available on the other devices syncing to the server. If not, the file will only available on the server.

- **filepath** the path of the file to be synced
- **sync** the boolean value to tell whether the file should be copied to the app/server folder

attachFile(String filePath, boolean sync) ;

Attach a file to the project folder by specifying the path of the file.If sync is true, the file will be available on the other devices syncing to the server. If not, the file will only available on the server. If the dir is defined, it will create the dir in the app/server folder.

- **filepath** the path of the file to be synced
- **sync** the boolean value to tell whether the file should be copied to the app/server folder
- **dir** the directory to be added to the app/server folfer, might be null

attachFile(String filePath, boolean sync, String dir) ;

Attach a file to the project folder by specifying the path of the file.If sync is true, the file will be available on the other devices syncing to the server. If not, the file will only available on the server. If the dir is defined, it will create the dir in the app/server folder.

- **filepath** the path of the file to be synced
- **sync** the boolean value to tell whether the file should be copied to the app/server folder
- **dir** the directory to be added to the app/server folfer, might be null
- **callback** code to execute when file is finish copying into files directory

attachFile(String filePath, boolean sync, String dir, String callback) ;

List all attached file to an arch entity by providing the id of the arch entity.

- **id** the id of the arch entity, it will get the logic error dialog if null

viewArchEntAttachedFiles(String id);

List all attached file to an arch entity by providing the id of the relationship.

- **id** the id of the relationship, it will get the logic error dialog if null

viewRelAttachedFiles(String id);

Open the camera and then execute the callback after finish

- **callback** the callback executed after operation is finish

openCamera(String callback);

Open the video recorder and then execute the callback after finish

- **callback** the callback executed after operation is finish

openVideo(String callback);

Open the audio recorder and then execute the callback after finish

- **callback** the callback executed after operation is finish

recordAudio(String callback);

Get the path of last taken picture

return the path of the last taken picture

String getLastPictureFilePath();

Get the path of last recorded video

return the path of the last recorded video

String getLastVideoFilePath();

Get the path of last recorded audio

return the path of the last recorded audio

String getLastAudioFilePath();

return returns true if files are currently being attached

isAttachingFiles();

return returns the full path to attached file

getAttachedFilePath(String file) ;

return returns the full path to attached file

stripAttachedFilePath(String file) ;

MISC

Executes a given string of code

- **code** the string of code to execute

execute(String code);