Program Logic Support

Persist Functionality

When android runs low on resources the project activity can be destroyed if its in a suspended state and then restored when its resumed. This will result in a loss of variable state in the logic script. To restore variable state in the logic script use the following method to define the name of a single object that will saved and restored when the activity is destroyed and restored. For more information on how to use this function please look at the cookbook.

- **name** the name of the object to restore

persistObject(String name) ;

Tab / Tab Group Functionality

Show the tab group with the following reference and clear all values in the tab group.

- **ref** the reference to the tab group

newTabGroup(String ref) ;

Show the tab with the following reference and clear all values in the tab.

- **ref** the reference to the tab

newTab(String ref) ;

Show the tab group with the following reference. This will retain all the current values in the tab group.

- **ref** the reference to the tab group

showTabGroup(String ref);

Show the tab group with the following reference and load the values of the supplied entity or relationship id into the tab group.

- **ref** the reference of the tab group
- **id** the id of the entity or relationship

showTabGroup(String ref, String id);

Show the tab with the following reference. This will retain all the current values in the tab.

- **ref** the reference to the tab

showTab(String ref) ;

Show the tab with the following reference and load the values of the supplied entity or relationship id into the tab.

- **ref** the reference to the tab
- **id** the id of the entity or relationship

showTab(String ref, String id) ;

Close the tab group with the following reference with an option to show a warning dialog if there are changes that haven't been saved.

- **ref** the reference to the tab group
- **warn** set to true to show a warning dialog if there are changes that haven't been saved

cancelTabGroup(String ref, boolean warn);

Close the tab with the following reference with an option to show a warning dialog if there are changes that haven't been saved.

- **ref** the reference to the tab
- **warn** set to true to show a warning dialog if there are changes that haven't been saved

cancelTab(String ref, boolean warn);

Dialog Functionality

Show 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) ;

Show an alert dialog to the user with the given message.

- **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);

Show 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) ;

Show 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 Functionality

Set the field with the following reference to the given value. If the field reference is not found a logic error dialog will appear. For more information on how to use this function please look at the cookbook.

- **ref** the reference to the field
- **value** the value to set the field to

setFieldValue(String ref, Object value);

Set the certainty of the field with the following reference to the given value If the field reference is not found a logic error dialog will appear.

- **ref** the reference to the field
- **value** the value to set the certainty for the field to

setFieldCertainty(String ref, Object value);

Set the annotation of the field with the following reference to the given value If the field reference is not found a logic error dialog will appear.

- **ref** the reference to the field
- **value** the value to set the annotation for the field to

setFieldAnnotation(String ref, Object value);

Get value of the field with the following reference. If the field reference is not found a logic error dialog will appear.

- **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 with the following reference. If the field reference is not found a logic error dialog will appear.

- **ref** the reference to the field

return the certainty of the field

Object getFieldCertainty(String ref);

Get annotation of the field with the following reference. If the field reference is not found a logic error dialog will appear.

- **ref** the reference to the field

return the annotation of the field

Object getFieldAnnotation(String ref);

Get the current time of the application

return the current time e.g. 2013-01-20 13:20:01

String getCurrentTime();

Clear a dirty field with the following reference

- **ref** the reference to the field

clearFieldDirty(String ref) ;

Event Callback Functionality

Binding an event to the field with the given reference. If the field reference is not found a logic error dialog will appear.

- **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 event is triggered

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

Binding a focus/blur event to the field with the given reference. If the field reference is not found a logic error dialog will appear.

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

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

User Functionality

Set the current user of the application. Is a requirement for insert records into the database.

- **user** the user of the application

setUser(User user);

Archaeological Entity / Relationship Functionality

Insert a new or update an existing archaeological entity record and return the id of the saved record.

- **entity_id** the id of the entity to be saved, set to null to save a new record
- **entity_type** the type of the entity to be saved, must be one specified in the data schema
- **geo_data** the list of geometries to be associated with the entity
- **attributes** the list of attributes to be associated with the entity

return the id of the saved entity

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

Insert a new or update an existing relationship record and return the id of the saved record.

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

return the id of the saved relationship

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

Deletes the specified archaeological entity.

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

return boolean value true if deleted, false if not

deleteArchEnt(String entity_id);

Deletes the specified relationship.

- **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 existing archaeological entity to an existing relationship and the verb of the relation.

- **entity_id** the id of the entity
- **rel_id** the id of the relationship
- **verb** the relation verb that is defined in the data schema for the specified relationship

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

Create an attribute list.

return new attribute list

createAttributeList() ;

Create an entity attribute to be added to the attribute list.

- **name** the name of the attribute
- **text** the text of the entity attribute, could be null
- **vocab** the vocab id 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, could be null

return an entity attribute

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

Create an entity attribute to be added to the attribute list.

- **name** the name of the attribute
- **text** the text of the entity attribute, could be null
- **vocab** the vocab id 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, could be null
- **isDeleted** set to true to delete the attribute

return an entity attribute

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

Create a relationship attribute to be added to the attribute list.

- **name** the name of the attribute
- **text** the text of the relationship attribute, could be null
- **vocab** the vocab id of the relationship attribute (obtained from the database), could be null
- **certainty** the certainty of the relationship attribute, could be null

return a relationship attribute

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

Create a relationship attribute to be added to the attribute list.

- **name** the name of the attribute
- **text** the text of the relationship attribute, could be null
- **vocab** the vocab id of the relationship attribute (obtained from the database), could be null
- **certainty** the certainty of the relationship attribute, could be null

- **isDeleted** set to true to delete the attribute

return a relationship attribute

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

Populate dropdown field with the given collection of values. If the field reference is not found a logic error dialog will appear.

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

populateDropDown(String ref, Collection values);

Populate radio group field with the given collection of values. If the field reference is not found a logic error dialog will appear.

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

populateRadioGroup(String ref, Collection values) ;

Populate checkbox group field with the given collection of values. If the field reference is not found a logic error dialog will appear.

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

populateCheckBoxGroup(String ref, Collection values) ;

Populate list with the given collection of values. If the field reference is not found a logic error dialog will appear.

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

populateList(String ref, Collection values);

Used in conjunction with the click event on a list. This returns last selected value in the list.

return the last selected value in the list

String getListItemValue() ;

Populate picture gallery field with the given collection of values. If the field reference is not found a logic error dialog will appear.

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

populatePictureGallery(String ref, Collection values);

Populate camera picture gallery field with the given collection of values. If the field reference is not found a logic error dialog will appear.

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

populateCameraPictureGallery(String ref, Collection values);

Populate video gallery field with the given collection of values. If the field reference is not found a logic error dialog will appear.

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

populateVideoGallery(String ref, Collection values);

Populate audio list field with the given collection of values. If the field reference is not found a logic error dialog will appear.

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

populateAudioList(String ref, Collection values);

Fetch the archaeological entity record with the specified id.

- **id** the id of the entity

return an entity or null

Object fetchArchEnt(String id);

Fetch the relationship record with the specified id.

- **id** the id of the relationship

return a relationship or null

Object fetchRel(String id);

Fetch the result of a query. This will return only a single row.

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

return Collection of String

Object fetchOne(String query);

Fetch the results of a query. This will return a collection of rows.

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

return Collection of Collection of String

Collection fetchAll(String query);

Fetch a list of entities. Each row in the collection is a list with the first item being the id of the entity and the second the identifier of the entity.

- **type** the type of arch entity to filter

return Collection of Collection of String

Collection fetchEntityList(String type);

Fetch a list of relationships. Each row in the collection is a list with the first item being the id of the relationship and the second the identifier of the relationship.

- **type** the type of relationship to filter

return Collection of Collection of String

Collection fetchRelationshipList(String type);

Navigation Functionality

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

goBack();

GPS Functionality

Set the GPS update interval to determine how often the GPS should update. Default value is 10 seconds.

- **seconds** the length of the interval in seconds

setGPSUpdateInterval(int seconds);

Start using the internal GPS to update the location.

startInternalGPS();

Start using the external GPS to update the location. A dialog will be presented to the user to choose which bluetooth device to be used as external GPS.

startExternalGPS();

Get the GPS position as 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 as longitude and latitude in the project projection 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 as 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 GPS track log with as either time based or distance based.

- **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
- **callback** code to execute when interval limit is reached

startTrackingGPS(String type, int value, String callback);

Stop the GPS track log.

stopTrackingGPS();

Map Functionality

Bind map click and select events to the map with the following reference.

- **ref** the reference to the map view
- **clickCallback** the callback that is executed when the map view is clicked
- **selectCallback** the callback that is executed when a element is clicked

onMapEvent(String ref, String clickCallback, selectCallback) ;

Used in conjunction with the map click event. This returns the last point clicked on the map.

return the clicked map point

getMapPointClicked() ;

Used in conjunction with the map select event. This returns the last element selected on the map.

return the selected geometry

getMapGeometrySelected() ;

Add a raster map layer to the map view with the following reference and make it the base layer. You can only have a single base layer.

- **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) ;

Add a raster map layer to the map view with the following reference. You can have multiple raster layers.

- **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) ;

Add a shape map layer to the map view with the following reference.

- **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) ;

Add a vector map layer to the map view with the following reference.

- **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) ;

Add a database map layer to the map view with the following reference.

- **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) ;

Add a canvas layer to the map view with the given reference.

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

createCanvasLayer(String ref, String layerName) ;

Set the focus point of the map view using longitude and latitude specified in the projects projection.

- **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 using longitude and latitude specified in the projects projection.

- **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 with the given rotation.

- **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 with the given level.

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

setMapZoom(String ref, float zoom) ;

Set the tilt of the map with the given tilt.

- **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) ;

Remove a layer from the map view with the given reference.

- **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) ;

Center the map based on the current GPS position if there is GPS position.

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

centerOnCurrentPosition(String ref);

Change the visibility of the specified layer in the map view with the given reference.

- **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) ;

Draw a point on the map view with the given reference by specifying the point and style.

- **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 with the given reference by specifying the points and style.

- **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 with the given reference by specifying the points and style.

- **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 with the given reference by specifying the geometry id.

- **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 with the given reference by specifying the list of geometry.

- **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 longitude and 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 longitude and 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 longitude and 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 with the given reference.

- **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 with the given reference by specifying the geometry id.

- **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 with the given reference by specifying the geometry and style.

- **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 a point style with the given 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 a line style with the given 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 a polygon style with the given 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 a text style with the given 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) ;

This sets the tilt of the map to be locked at 90 degrees to give a 2D view of the map.

- **ref** the reference to the map view
- **lock** set to true to lock the map view

lockMapView(String ref, boolean lock) ;

Add the specified geometry to the list of highlighted geometry on the map with the given reference.

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

addGeometryHighlight(String ref, int geomId) ;

Remove the specified geometry from the list of highlighted geometry on the map with the given reference.

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

removeGeometryHighlight(String ref, int geomId) ;

Call this method to prepare the geometry in the highlisted list to be transformed to their new position.

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

prepareHighlightTransform(String ref) ;

Call this method after calling prepareHighlightTransform once your ready to transform the geometry to their new position.

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

doHighlightTransform(String ref) ;

Clear the highlighted geometry list for the map view with the given reference.

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

clearGeometryHighlights(String ref) ;

Get the highlighted geometry list for the map view with the given reference.

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

getGeometryHighlights(String ref) ;

Add a database layer query to the map with the given reference which will then be used when loading database layers via the layer manager.

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

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

Add a track log layer query to the map with the given reference which will then be used when loading track log layers via the layer manager.

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

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

Add a query builder to the map view which will then be used by the database selection tool via the tools bar.

- **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 selection tool by providing the sql query and adding parameters.

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

createQueryBuilder(String sql) ;

Add a legacy query builder to the map view which will then be used by the legacy selection tool via the tools bar.

- **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 legay query builder for legacy selection tool by providing the sql query and adding parameters.

- **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 is used in conjunction with the tool create event. This returns the last created geometry id.

getMapGeometryCreated() ;

This is used in conjunction with the tool load event. Thi will return the last selected geometry id.

getMapGeometryLoaded() ;

This is used in conjunction with the tool load event. This will return the lsat selected geometry type (either entity or relationship).

getMapGeometryLoadedType() ;

Refresh all the layers of the map.

- **reference** to the map view

refreshMap(String ref) ;

Sync Functionality

Push the full database from the app to the server and execute the callback when finished.

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

pushDatabaseToServer(String callback) ;

Pull the full database from the server to the app and execute the callback when finished.

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

pullDatabaseFromServer(String callback) ;

Enable or disable syncing the database from the app to the server.

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

setSyncEnabled(boolean value) ;

Bind to the sync start, success and failure events.

- **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 in seconds

setSyncMinInterval(float value) ;

Set the maximum interval for the sync to happen.

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

setSyncMaxInterval(float value) ;

Set the delay interval to add to the current sync interval when a sync failure happens.

- **value** the delay interval for each sync in seconds.

setSyncDelay(float value) ;

Set whether files should also be synced with the database.

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

setFileSyncEnabled(boolean enabled) ;

Static Data Functionality

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();

Get the static data for permit issued by.

return permit issued by

public String getPermitIssuedBy() ;

Get the static data for permit type.

return permit type

public String getPermitType() ;

Get the static data for copyright holder.

return copyright holder

public String getCopyrightHolder() ;

Get the static data for client/sponsor.

return client/sponsor

public String getClientSponsor() ;

Get the static data for land owner.

return land owner

public String getLandOwner() ;

Get the static data for whether containing sensitive data or not.

return true or false

public String hasSensitiveData() ;

File attachment Functionality

Show the file browser and execute the callback once a file is selected.

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

showFileBrowser(String callback) ;

This is used in conjuction with showFileBrowser. This returns the filename of the last selected file.

return name of last selected file

getLastSelectedFilename() ;

This is used in conjuction with showFileBrowser. This returns the filepath of the last selected file.

return path of last selected file

getLastSelectedFilepath() ;

Copy a file into the projects server or app folders. If the sync is set to true then the file is copied to the app folder. If the sync is set to false then the file is copied to the server folder. Files in the app folder are sync to the server and other apps. Files in the server folder are only synced to 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) ;

Copy a file into the projects server or app folders. If the sync is set to true then the file is copied to the app folder. If the sync is set to false then the file is copied to the server folder. Files in the app folder are sync to the server and other apps. Files in the server folder are only synced to 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
- **dir** the directory to be added to the app/server folfer, can be null

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

Copy a file into the projects server or app folders. If the sync is set to true then the file is copied to the app folder. If the sync is set to false then the file is copied to the server folder. Files in the app folder are sync to the server and other apps. Files in the server folder are only synced to 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
- **dir** the directory to be added to the app/server folfer, can be null
- **callback** code to execute when file is finished copying into the directory

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

List all attached files of an archaeological entity by providing the id of the entity.

- **id** the id of the entity

viewArchEntAttachedFiles(String id);

List all attached files of a relationship by providing the id of the relationship.

- **id** the id of the relationship

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);

This is used in conjunction with openCamera. This returns the file path of the last picture taken.

return the path of the last taken picture

String getLastPictureFilePath();

This is used in conjunction with openVideo. This returns the file path of the last video taken.

return the path of the last recorded video

String getLastVideoFilePath();

This is used in conjunction with recordVideo. This returns the file path of the last audio taken.

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 file path relative to the projects folder of file

stripAttachedFilePath(String file) ;

MISC

Executes a given string of code.

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

execute(String code);