INRIX Client Library Quick Start Guide

Requirements

The following items are required to incorporation the INRIX Client Library into an application. The Sample Application provided with each release of the library is meant to illustrate these requirements.

Register at a Vendor

Obtain a valid vendorID and token from INRIX

Add Dependencies

1. Create a dependency on the INRIX Client Library jar file (com.inrix.sdk.jar)
2. Create a dependency on the following jar files
    a. Gson Library (gson-2.2.4.jar)
    b. Simple XML (simple-xml-2.7.1.jar)
    c. Volley (vollylib.jar)
    d. Android Support Lib (android-support-v4.jar)
3. Add a dependency on the included Google Play Services project (each release of the library will contain the required version of the Google Play Services project)

Add Minimum Android SDK Version

The INRIX Client Library supports Android SDK Version 9 and up. Your AndroidManifest.xml will need to have the following information.

```
<uses-sdk
      android:minSdkVersion="9"
      android:targetSdkVersion="<Your-Target-SdkVersion>"
 />
```

Add Permissions

Add the following lines to your AndroidManifest.xml file:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />
```

Add Service

Declare service somewhere inside <application> tag.
PLEASE NOTE, without this service declared SDK might use more battery power!

```
<application>
  ……
      <service
```

```
        android:name="com.inrix.sdk.activityrecognition.ActivityRecognitionIntentService"
            android:exported="false"
            android:label="@string/app_name" >
        </service>
</application>
```

Initialization

Initialize method

All that is required for initialization of the INRIX Client Library is to call the following method.

Inrix.*initialize*(**final Context context**);

This method will initialize the INRIX Client Library. This will involve establishing a protected channel of communication with the INRIX servers. Please note that internally, application context will be used, so in order to clear all the application context references, you must call Inrix.shutdown() when application is about to close.

Authentication

The INRIX Client library utilizes a protected channel of communication. The library manages all the details and data internally. The application does not need to manage this.

Managers

Access to data is provided through managers. The Client Library provides the following managers:

- AlertManager – Provides incident alerting capabilities
- GasStationManager – Provides access to GasStation
- IncidentManager – Provides access to Incidents
- LocationsManager – Provides access to saved locations. Locations are saved in association with a User.
- ParkingManager – Provides access to Parking Lots
- RouteManager – Provides access to Routes
- TileManager – Provides access to Traffic Tiles
- UserManger – Provides User login/logout and User account management

Current Location Parameter

Each of our API calls takes one or more parameters consisting of latitude, longitude, heading, speed and an anonymous identifier that allows us to ensure that we are giving the best traffic data in the API responses. That data is fully anonymous and is used to calculate which road the user is on, in which direction they are heading and determining the traffic conditions around that user.

IDataResponseListener

Each request to provide data from a manager requires a listener. Each request has a specific listener that implements the IDataResponseListener.

```
interface IDataResponseListener<T> {
	void onResult(T data);

	void onError(Error error);
}
```

onResult

Method that will be called when the request was successful and the response is good. The specific listener defines the type of data passed to this method.

onError

Method that will be called when the request was not successful. The data passsed to this method is an object containing information about the error such as the server error id and error message.

AlertsManager

The AlertsManager provides the mechanism to support alerting on Incidents that occur around the devices current position.

createIncidentAlert

Call createIncidentAlert to create an alert of Incidents that meet specific criteria. The listener will be call with the set on Incidents that meet the criteria.

```
public final IncidentAlert createIncidentAlert(
      final IIncidentsAlertListener listener,
      IncidentAlertOptions alertParams) throws InrixException
```

IncidentAlertOptions

Specify the desired alert interval (in milliseconds) and the filter to the constructor of the IncidentAlertOptions.

```
public IncidentAlertOptions(
      int alertInterval,
      IFilter<Incident> filter)
```

The radius in which to evaluate Incidents will vary based on the speed reported by Location Services. The radius (in miles) to use is calculated by taking the speed (in MPH) and dividing that by the speed factor (which is 10 by default). Calling the following method on the IncidentAlert Options before setting the alert can set the speed factor:

```
public void setSpeedFactor(float speedFactor)
```

IIncidentsAlertListener

On success, the IIncidentAlertListener will return a list of Incidents.

```
public interface IIncidentsAlertListener extends
            IDataResponseListener<List<Incident>> {
}
```

Code Snippet (from IncidentAlertsActivty)

```
public class IncidentAlertsActivity extends FragmentActivity
      implements IIncidentsAlertListener {

   . . .

   @Override
   protected void onStart() {
      super.onStart();
      AlertsManager alertManager = new AlertsManager();
      progressBar.setVisibility(View.VISIBLE);
      timestamp.setText("Loading...");
      alert = alertManager.createIncidentAlert(this,
         new IncidentAlertOptions(
            ALERT_INTERVAL,
            new IFilter<Incident>() {
               @Override
               public boolean isItemAllowed(Incident item) {
                  return (true);
               }
            }));
   }

   @Override
   protected void onStop() {
      super.onStop();
      if (this.alert != null) {
         this.alert.cancel();
         this.alert = null;
      }
   }
```

GasStationManager

The GasStationManager provides access to gas stations, either by specifying a center point and radius or a box to contain the incidents.

```
public final ICancellable getGasStationsInRadius(
      GasStationsRadiusOptions requestParameters,
            final IGasStationResponseListener listener)
            throws  InrixException

public final ICancellable getGasStationsInBox(
      GasStationsBoxOptions requestParameters,
            final IGasStationResponseListener listener)
            throws  InrixException
```

In either case, the IGasStationsResponseListener will receive a list of Gas Stations that match the requestParam.

GasStationRadiusOptions Constructor

```
public GasStationsRadiusOptions(
      GeoPoint center,
      double radius,
      boolean metric)
```

GasStationBoxOptions Constructor

```
public GasStationsBoxOptions(
      GeoPoint boxStart,
      GeoPoint boxEnd)
```

Both GasStationRadiusOptions and GasStationBoxOptions are inherited from GasStationOption, which provides the following methods to further control the Gas Station request:

Set the type of Gas Station request :

```
public void setProductTypes(int productTypes)
```

All types of fuels	PRODUCT_TYPE_ALL (default)
A fuel used in diesel engines, and made of vegetable oil or animal fat	PRODUCT_TYPE_BIODIESEL
Diesel fuel	PRODUCT_TYPE_DIESEL
A diesel fuel with special additives to improve performance	PRODUCT_TYPE_DIESEL_PLUS
A diesel fuel used by trucks	PRODUCT_TYPE_DIESEL_TRUCK
LPG (liquid petroleum gas, usually propane, also called Autogas)	PRODUCT_TYPE_LPG

Natural gas (also called Compressed Natural Gas, or CNG	PRODUCT_TYPE_METHANE
Regular grade gasoline	PRODUCT_TYPE_GASOLINE_REGULAR
Middle grade gasoline	PRODUCT_TYPE_GASOLINE_MIDGRADE
Premium grade gasoline	PRODUCT_TYPE_GASOLINE_PREMIUM
E85 Ethanol/gasoline mixture	PRODUCT_TYPE_GASOLINE_E85
Leaded gasoline	PRODUCT_TYPE_GASOLINE_NORMAL
Unleaded gasoline, with a 92 octane rating	PRODUCT_TYPE_GASOLINE_SP92
Unleaded gasoline, with a 95 octane rating	PRODUCT_TYPE_GASOLINE_SP95
Unleaded gasoline, with a 95 octane rating and 10% ethanol	PRODUCT_TYPE_GASOLINE_SP95_E10
Unleaded gasoline, with a 98 octane rating	PRODUCT_TYPE_GASOLINE_SP98

Set the fields that are return by the request:

```
public void setOutputFields( int outputFields )
```

All the fields	OUTPUT_FIELDS_ALL (default)
gas station name or brand	OUTPUT_FIELDS_BRAND
the latitude and longitude of the gas station	OUTPUT_FIELDS_LOCATION
the address of the gas station	OUTPUT_FIELDS_ADDRESS
Products sold in the gas station	OUTPUT_FIELDS_PRODUCTS
billing currency code	OUTPUT_FIELDS_CURRENCY_CODE

Code Snippet (from GasStationListActivty)

```
public class GasStationsListActivity extends FragmentActivity {
      . . .

      @Override
      protected void onCreate(Bundle savedInstanceState) {

      . . .

            // Get the gas stations for the selected city and radius
            int outputOptions =
                  GasStationsOptions.OUTPUT_FIELDS_BRAND |
                  GasStationsOptions.OUTPUT_FIELDS_ADDRESS |
                  GasStationsOptions.OUTPUT_FIELDS_LOCATION |
                  GasStationsOptions.OUTPUT_FIELDS_CURRENCY_CODE |
                  GasStationsOptions.OUTPUT_FIELDS_PRODUCTS;
            int productTypes = GasStationsOptions.PRODUCT_TYPE_ALL;
            GasStationsRadiusOptions params = new
                  GasStationsRadiusOptions(SEATTLE_POSITION,
                        REQUEST_RADIUS,
```

```
                false,
                outputOptions,
                productTypes);

        this.client.getGasStationManager().getGasStationsInRadius(
              params,
              new IGasStationResponseListener() {

              @Override
              public void onResult(GasStationCollection data) {
                    if( null != data
                          && null != data.getGasStations() ){
                          setGasStationList(data.getGasStations());
                    }
              }

              @Override
              public void onError(Error error) {
                    setGasStationList(null);
              }

        });
```

IncidentManager

The IncidentManager provides access to incidents, either by specifying a center point and radius or a box to contain the incidents.

```
public final ICancellable getIncidentsInRadius(
      final IIncidentsResponseListener listener,
      IncidentRadiusOptions requestParams)

public final ICancellable getIncidentsInBox(
      final IIncidentsResponseListener listener,
      IncidentBoxOptions requestParams)
```

In either case, the IIncidentsResponseListener will receive a list of Incidents that match the requestParam.

IncidentRadiusOptions Constructor

```
public IncidentRadiusOptions(GeoPoint center)
```

IncidentBoxOptions Constructor

```
public IncidentBoxOptions(GeoPoint corner1,
      GeoPoint corner2)
```

IncidentOptions

Both IncidentRadiusOptions and IncidentBoxOptions are inherited from IncidentOptions, which provides the following methods to further control the Incidents request:

setIncidentType

Set the type of incidents to request:

```
public IncidentOptions setIncidentType(int incidentType)
```

This option returns all unusual incidents that may slow down traffic such as a car accident.	INCIDENT_RESULT_TYPE_INCIDENTS
This option returns only construction incidents.	INCIDENT_RESULT_TYPE_CONSTRUCTION
This option returns unusual events slated for the area such as a major sporting event.	INCIDENT_RESULT_TYPE_EVENTS
This option returns reports about the slowing down of traffic on your route.	INCIDENT_RESULT_TYPE_FLOW
This option returns reports about the police presence.	INCIDENT_RESULT_TYPE_POLICE

This option returns unusual weather incidents that could alter traffic speed.	INCIDENT_RESULT_TYPE_WEATHER
Selecting this option returns all incidents.	INCIDENT_RESULT_TYPE_ALL (default)

setIncidentSource

Set the source of the incident request:

```
public IncidentOptions setIncidentSource(
      int incidentSource)
```

Return incidents from non-commercial sources.	INCIDENT_SOURCE_INRIXONLY
Return incidents from community sources.	INCIDENT_SOURCE_COMMUNITY
Return incidents from all sources.	INCIDENT_SOURCE_ALL (default)

setIncidentOutputFields

Set the fields that are return by the request:

```
public IncidentOptions setIncidentOutputFields(
      int incidentOutputFields)
```

The unique identifier of an incident.	INCIDENT_OUTPUT_FIELDS_ID
The version number of the incident report, incremented each time an incident report is updated.	INCIDENT_OUTPUT_FIELDS_VERSION
The type of the incident (Incidents, Construction, Events, Flow, Area, or Weather). Incidents can be determined from the Alert-C event code, Construction indicates the presence of road construction, Events can be weather-related or a scheduled sporting/public event, and Flow indicates a blocking incident.	INCIDENT_OUTPUT_FIELDS_TYPE
The severity of the incident. This value can be in the range of 0-4, with 4 indicating the highest severity.	INCIDENT_OUTPUT_FIELDS_SEVERITY
The event code of the incident. These are standard Alert-C event codes.	INCIDENT_OUTPUT_FIELDS_EVENT_CODE
The latitude and longitude of the incident.	INCIDENT_OUTPUT_FIELDS_LATLONG
Whether the incident impacts traffic flow. This field is set if the appearance of the incident changes the traffic flow below a certain percentage from that which is	INCIDENT_OUTPUT_FIELDS_IMPACTING

normally expected for the given segment of road at that time, given the current conditions.	
The starting time of the incident.	INCIDENT_OUTPUT_FIELDS_STARTTIME
The ending time of the incident.	INCIDENT_OUTPUT_FIELDS_ENDTIME
Provide the delay in minutes versus typical conditions and versus free flow conditions.	INCIDENT_OUTPUT_FIELDS_DELAY_IMPACT
The points in a polygon that describes a weather incident that is returned, in GML format. For more information about GML format, see http://www.opengeospatial.org/standards/gml.	INCIDENT_OUTPUT_FIELDS_AREA
The Radio Data System data.	INCIDENT_OUTPUT_FIELDS_RDS
This option returns all of the options available.	INCIDENT_OUTPUT_FIELDS_ALL (default)

Code Snippet (from IncidentListActivty)

```
public class IncidentListActivity extends FragmentActivity {

	. . .

	@Override
	protected void onCreate(Bundle savedInstanceState) {

	. . .

		// Get the Incidents for the selected city and radius
		IncidentRadiusOptions params = new IncidentRadiusOptions(
				SEATTLE_POSITION, INCIDENT_RADIUS);
		this.client.getIncidentManager().getIncidentsInRadius(
				new IIncidentsResponseListener() {

					@Override
					public void onResult(List<Incident> data)
					{
						setIncidentList(data);
					}

					@Override
					public void onError(Error error) {
						setIncidentList(null);
					}

				}, params);

	}
```

LocationManager

<under construction>

ParkingManager

The ParkingManager provides access to list of parking lots, either by specifying a center point and radius or a box to contain the incidents.

```
public final ICancellable getParkingLotsInRadius(
      final GeoPoint center,
      final int radius,
      final ParkingOptions options,
      final IParkingResponseListener listener)
            throws InvalidParameterException

public final ICancellable getParkingLotsInBox(
      final GeoPoint corner1,
      final GeoPoint corner2,
      final ParkingOptions options,
      final IParkingResponseListener listener)
            throws InvalidParameterException
```

In either case, the IParkingResponseListener will receive a list of Parking Lots that match the options.

ParkingRadiusOptions Constructor

```
public ParkingRadiusOptions(
      GeoPoint center,
      double radius,
      boolean metric)
```

ParkingBoxOptions Constructor

```
publicParkingBoxOptions(
      GeoPoint boxStart,
      GeoPoint boxEnd)
```

Both ParkingRadiusOptions and ParkingBoxOptions are inherited from ParkingOption, which provides the following methods to further control the Gas Station request:

setOutputFields

Set the fields that are return by the request:

```
public void setOutputFields( int outputFields )
```

Basic information about the parking lot: name, address, location.	PARKING_OUTPUT_FIELD_BASIC
Pricing information for the parking lots.	PARKING_OUTPUT_FIELD_PRICING

Geometry of the parking lot.	PARKING_OUTPUT_FIELD_GEOMETRY
Dynamic fill rate.	PARKING_OUTPUT_FIELD_DYNAMIC
Same as basic, with additional information: photo link, gate information, pricing, etc.	PARKING_OUTPUT_FIELD_STATIC
All available information about the parking lot.	PARKING_OUTPUT_FIELD_ALL

setSortBy

Set the sort order of the results

```
public final ParkingOptions setSortBy(final SORT_BY sortBy)
```

setUnits

Set the units used by the request

```
public final ParkingOptions setUnits(final UNIT units)
```

Code Snippet (from ParkingListActivty)

```
protected void onCreate(Bundle savedInstanceState) {
	// Initialize INRIX
	initializeINRIX();

	// Get the parking lots for the selected city and radius
	ParkingInRadiusOptions options =
		new ParkingInRadiusOptions(SEATTLE_POSITION,
			REQUEST_RADIUS);

		this.client.getParkingManager()
			.getParkingLotsInRadius(
				new IParkingResponseListener() {

				@Override
				public void onResult(List<ParkingLot> data) {
					setParkingLotList(data);
				}

				@Override
				public void onError(Error error) {
					setParkingLotList(null);
				}

			}, options);

}
```

RouteManager

The RouteManager provides access to routes and routing data.

requestRoutes

Use the requestRoutes method to request routes.

```
public ICancellable requestRoutes(
      final RouteOptions params,
      final IRouteResponseListener listener)
            throws InvalidParameterException
```

The routes returned will be calculated using the criteria specified in the RouteOptions.

RouteOptions

```
public RouteOptions(GeoPoint start, GeoPoint end)
```

In either case, the IParkingResponseListener will receive a list of Parking Lots that match the options.

RouteOptions also provides the following methods to further control Routes request:

setWaypoints

The routes requested will go from the start point, through the waypoints in the order they are specified, and to the end point.

```
public RouteOptions setWaypoints(List<GeoPoint> waypoints)
      throws InvalidParameterException
```

setTolerance

Tolerance reduces the number of latitude/longitude points returned. A higher tolerance will result in few points returned.

```
public RouteOptions setTolerance(int tolerance)
```

setNumAlternates

Determines the number of alternate routes calculated. By default, only one route is returned but you can request up to 2 alternates.

```
public RouteOptions setNumAlternates(int numAlternates)
```

Code Snippet (from BusyCommuterActivty)

```
RouteOptions routeParams = new
      RouteOptions(selectedCity.getPoint(),
                  location.getPoint());
      routeParams.setTolerance(ROUTES_TOLERANCE);
      routeParams.setNumAlternates(ROUTES_NUM_ALTERNATES);


routeManager().requestRoutes(routeParams,
      new IRouteResponseListener() {

            @Override
            public void onResult(RoutesCollection data) {
                  incidentList.addRoutesList(location,
                                          data.getRoutes());
            }

            @Override
            public void onError(Error error) {
                  showError(error.toString());
            }
});
```

requestTravelTimes

Use the requestTravelTimes method to request the travel times over a period of time for a given route.

```
public final ICancellable requestTravelTimes(
      TravelTimeOptions requestParameters,
      final ITravelTimeResponseListener listener )
            throws  InvalidParameterException
```

The travel times returned will be calculated using the criteria specified in the TravelTimeOptions.

TravelTimeOptions

```
public TravelTimeOptions( Route route,
      int travelTimeCount,
      int travelTimeInterval)
```

route - The route object
travelTimeCount – Number of travel times requested. (min = 1 max = 96)
travelTimeInterval - interval between travel times in minutes (min = 1 max = 1440)

The travel time returned will be calculated using the criteria specified in the TravelTimeOptions.

setDepartureTime

Set the time of departure. Defaults to "now" if not specified.

```
public void setDepartureTime(Date departureTime)
```

Code Snippet (from TravelTimesActivty)

```
// Request the route
GeoPoint START_LOCATION = new GeoPoint(47.602633, -122.336243);
GeoPoint END_LOCATION = new GeoPoint(47.616853, -122.193044);

RouteOptions routeParams = new RouteOptions(START_LOCATION,
                  END_LOCATION);
routeParams.setTolerance(ROUTES_TOLERANCE);
routeParams.setNumAlternates(ROUTES_NUM_ALTERNATES);

routeManager.requestRoutes(routeParams,
      new IRouteResponseListener() {

      @Override
      public void onResult(RoutesCollection data) {
            route = data.getRoutes().get(0);
      }

      @Override
      public void onError(Error error) {
      }
});
```

```
// Get the Travel Times for this route
TravelTimeOptions travelTimeOptions = new
      TravelTimeOptions(route,
            COLOR_CONTROL_IDS.length, 60);

routeManager.requestTravelTimes(travelTimeOptions,
      new ITravelTimeResponseListener() {

            @Override
            public void onResult(
                  TravelTimeResponse travelTimeResult) {
                        travelTimeResponse = travelTimeResult;
                        updateColors();
            }

            @Override
            public void onError(Error error) {
            }
      });
```