|  |
| --- |
| **FatosMap Software Development Kit**  **V2.0.4** |

표준 제/개정이력

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **버전** | **표준 제/개정일자** | **표준 제/개정내용** | **작성자** | **승인자** |
| 1.0 | 2017. 6. 18 | V1.0 초안 배포 | 우경일 | 박현재 |
| 1.01 | 2017. 6. 20. | V1.01 기능 수정 사항 반영 | 우경일 | 박현재 |
| 1.02 | 2017. 10. 31. | V1.02 기능 수정 사항 반영 | 우경일 | 박현재 |
| 1.03 | 2017. 11. 1. | V1.03 Android Studio 3.0 사항 반영 | 우경일 | 박현재 |
| 1.04 | 2017. 11. 1. | V1.04 프로젝트 설정 관련 추가 반영 | 우경일 | 박현재 |
| 1.05 | 2017. 11. 8. | V1.05 hybrid mod 및 gps 속도 interface 추가 | 우경일 | 박현재 |
| 1.06 | 2017. 11. 8. | 샘플 프로젝트 보안 | 우경일 | 박현재 |
| 1.07 | 2017. 11. 18. | Drive info 함수 추가 및 샘플 프로젝트 업데이트 | 우경일 | 박현재 |
| 1.08 | 2017. 12. 2. | 버그 수정 | 우경일 | 박현재 |
| 1.09 | 2017. 12. 11. | Api key관련 업데이트 | 우경일 | 박현재 |
| 1.10 | 2017. 12. 21. | 주행 화면 및 추가 정보 제공 | 우경일 | 박현재 |
| 1.11 | 2017. 12. 26. | 진입 불가 도로 코드 추가 | 우경일 | 박현재 |
| 1.12 | 2017. 12. 26. | Beacon 정보 입력 인터페이스 추가 | 우경일 | 박현재 |
| 1.13 | 2017. 12. 27. | FM\_CancelRoute 추가, 샘플 프로젝트에 모의주행 버튼 추가, beacon 입력 정보에 distance 추가,  층 정보 입력 변수명 수정 | 우경일 | 박현재 |
| 1.14 | 2017. 12. 27. | 내부 함수 업데이트 | 우경일 | 박현재 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **버전** | **표준 제/개정일자** | | **표준 제/개정내용** | **작성자** | **승인자** |
| 1.26 | 2018. 01. 12 | | 진입 불가,CarType 추가 및 버그 수정. | 우경일 | 박현재 |
| 1.27 | 2018. 01. 16 | | FM\_UpdateLGDLayer() 추가 . | 우경일 | 박현재 |
| 1.28 | 2018. 01. 20 | | FM\_Search\_Hybrid, RGCommonFragment 추가 . | 우경일 | 박현재 |
| 1.29 | 2018. 01. 20 | | 검색 sort 기능 추가, 추천어 검색 추가, Map Status callback 추가 . 검색 결과 얻는 함수/추천어 결과 얻는 함수 추가 | 우경일 | 박현재 |
| 1.30 | 2018. 01. 23 | | FM\_GetMapCenterPos추가 , 재탐색 안 걸리는 현상 수정 | 우경일 | 박현재 |
| 1.31 | 2018. 01. 25 | | FM\_StartRGService, FM\_RouteSummary, FM\_SelectRoute 추가, | 우경일 | 박현재 |
| 1.32 | 2018. 01. 26 | | FM\_RouteSummaryDetail 및 기능 개선 | 우경일 | 박현재 |
| 1.33 | 2018. 01. 29 | | FM\_FireWaterPoint추가 및 버그 수정 | 우경일 | 박현재 |
| 1.34 | 2018. 02. 1 | | - 경로탐색 상황에 대한 callback 추가  - FM\_AutoSetMapCenter - 지도 중심 설정 api 추가  - 좌표 기준 지도 이동 함수 추가  - 경로요약 중심 설정 수정. | 우경일 | 박현재 |
| 1.35 | 2018. 02. 2 | | - POI 중복으로 전달 되는 현상 수정.  - 경로안내 종료 후 Trip 창 뜨는 현상 수정.  - 지도 떨리는 현상 수정. | 우경일 | 박현재 |
| 1.36 | 2018. 02. 6 | | - FM\_RouteSummaryDetail(int nType) 추가. | 우경일 | 박현재 |
| 1.37 | 2018. 02. 8 | | -  FM\_CancelRoute(.) 호출 시 TBT/차기TBT layout 안 보이게 수정.  -  사용자 경로 안내 종료/목적지 경로안내 종료 시 start/goal flag 안 그리게 수정.  -  목적지 경로안내 종료 시 onRouteFinish() callback 이 호출 안 되던 현상 수정.  -  FM\_GetAddress(…) 추가 -> 좌표 기준 행정동명 얻는 함수  hybrid로 동작하므로 통신 상태가 아닐 경우 map 복사 필요  - SubMap에서도 onMove Callback과 화면 중심 좌표를 얻어 오는 FM\_GetSubMapCenterPos 추가 | 우경일 | 박현재 |
| 1.38 | 2018. 02. 9 | | - 자동 길안내 종료 때도 출/목 Flag 사라지게 수정되었습니다 | 우경일 | 박현재 |
| 1.39 | 2018. 03. 8 | | - FM\_MoveCurPosition,  getM\_nTotalRemainderDist(),  getM\_nServiceLinkRemainderTime() 추가 | 우경일 | 박현재 |
| 1.40 | 2018. 05. 25 | | - 경유지 기능 추가  FM\_RouteVol2\_Via, FM\_GetViaPOIList,  FM\_SetSimulationSpeed, 추가 | 우경일 | 박현재 |
| 1.41 | 2018. 05. 29 | | - 각 경유지의 남은 거리/시간 제공  - 항공모드 설정 관련 api 제공  - 기타 버그 수정. | 우경일 | 박현재 |
| 1.42 | 2018.06.06 | | - 가로 주행 layout 수정  - 경로 잡는 경유지 속성 추가 | 우경일 | 박현재 |
| 1.43 | 2018.06.21 | | - 경유지 통과 예외 처리 추가 | 우경일 | 박현재 |
| 1.44 | 2018.06.25 | | - 가로 모드 layout 수정  - 기타 내부 모듈 수정. | 우경일 | 박현재 |
| 1.45 | 2018.06.30 | | - 주행 UI 변경 및 내부 모듈 수정 | 우경일 | 박현재 |
| 1.46 | | 2018.07.04 | - 항공/위성 안 나오는 현상 수정.  - 일반 주행 SDI 나오게 수정. | 우경일 | 박현재 |
| 1.47 | | 2018.07.14 | - 주행 중 자동 현위치 시나리오 적용 - FM\_GoNextGoal 예제 추가  - 용량 다이어트 . | 우경일 | 박현재 |
| 2.0.0 | | 2018.07.23 | - 버전 리네임  - SDK인증 관련 수정 | 우경일 | 박현재 |
| 2.0.1 | | 2018.07.27 | - 경유지 명칭 길 때 약자 처리 | 우경일 | 박현재 |
| 2.0.2 | | 2018.08.13 | - 내부 음성 볼륨 조절 api 추가 | 우경일 | 박현재 |
| 2.0.3 | | 2018.08.20 | - DriveInfo에 Trip 정보 추가(총주행거리/시간) | 우경일 | 박현재 |
| 2.0.4 | | 2018.08.21 | 특정 사이트 Voice DB asset 관리 | 우경일 | 박현재 |

# 문서 개요

## 목적

본 문서는 FatosMap SDK의 API 및 기능을 정의함으로써 FatosMap SDK를 사용한 S/W 설계 및 개발 대한 요건을 명확히 정의함을 그 목적으로 한다.

## 문서 범위

본 문서는 FatosMap SDK의 구조설계나 알고리즘은 다루지 않고, 인터페이스 사양, UI 사양 등 주요 기능에 대해 정의 한다.

## S/W 사양 ID 규칙

본 프로젝트의 S/W 사양 ID 구성체계는 다음과 같이 관리 된다.

### S/W 사양 ID 구성 체계

|  |  |  |
| --- | --- | --- |
| **S/W 사양 ID 구성** | | [프로젝트 코드]\_SDK\_[사양 구분 ID]\_[항목 구분 ID]\_[일련번호] |
| **구성 항목 상세** | **프로젝트 코드** | 프로젝트에 부여된 코드 |
| **사양 구분 ID** | 기능 / 성능 / 기능안전 등 사양의 성격을 구분하는 구분자 ID |
| **항목 구분 ID** | 사양 구분 내 유사 항목의 묶음을 구분하는 구분자 ID |
| **일련번호** | 항목 내 사양에 부여하는 4자리 일련번호 (항상 0001 부터 시작함) |
| **S/W 사양 ID 예시** | | FM\_SDK\_FS\_SAM\_0001 |

### S/W 사양 ID 관리번호

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 사양 구분 | 사양 구분 ID | 항목 구분 | 항목 구분 ID | 비고 |
| Fatosmap SDK | FS | FatosMap SDK | SDK |  |

## 용어 및 약어

### 주요 용어

|  |  |
| --- | --- |
| 용어 | 정의 |
| FM Interface | Fatos SDK 개발자가 사용 또는 참조해야 할 Interface package Name |
| Fatos Map Fragment | Fatos SDK를 사용해 Map을 표출하고자 할 때 사용해야 하는 Fragment |
| Fatos RG Fragment | Fatos SDK를 사용해 경로 안내 정보를 표출하고자 할 때 사용해야 하는 Base Fragment |
|  |  |

### 약어

|  |  |  |
| --- | --- | --- |
| 약어 | 풀이 | 정의 |
| FM | Fatos Map 의 약자 | Interface Class, Base Activity, Basa Application Class 등 Fatos Map의 Interface 관련 Class 앞에 구분자로 정의 한다. |
| FMS | Fatos Map SDK의 약자 (버전 생략) |  |
| AAR | Android Archive Package | Android studio library |
| FME | Fatos Map Error 의 약자 | Interface Class의 성공/실패 여부에 대한 정의가 되어 있는 Class |

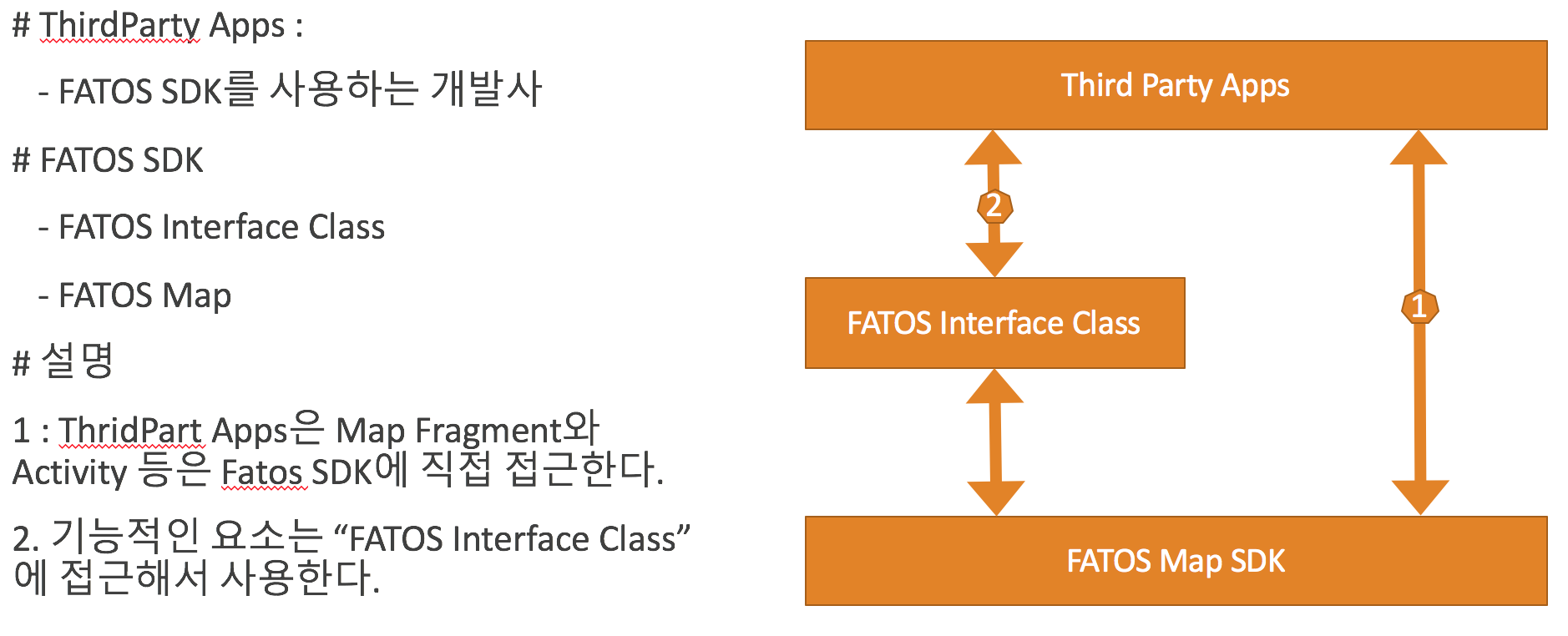
## 참조 문서

|  |  |
| --- | --- |
| 참조 문서 | 정의 |
|  |  |
|  |  |

# FMS 구조 및 기능 소개

본 항목에서는 FMS의 대략적인 구조 및 기능을 소개한다.

## FMS 구조



## FMS 기능

* Map
* FatosMap Fragment 제공
* Map Event, 상태 정보 등에 대한 Callback 제공
* 유동적으로 FatosMap을 사용 할 수 있게 함.
* Map Option 기능 제공.
* 지도 모드 변경, 좌표 이동, Zoom In/Out, Map Picker 등등에 대한 Map Contro API 제공.
* Search
* FMS v1.01은 T map 검색 결과 제공
* Route
* FMS v1.01은 T map 경로 탐색 결과 제공
* 경로 탐색 데이터 관리 기능 제공
* Route Guidance
* 길안내 가로/세로 Fragment 제공 및 Custom Fragment 연동 기능 제공.
* Interface & Base Class제공.
* Map , Route Guidance 관련 기능 및 제어를 Base Class를 통해 기능 제공해 중복 작업을 줄이고, Navigation 기능을 손 쉽게 추가 하도록 한다.

# FMS 초기 Initialize 및 Destroy관련

본 항목에서는 FMS의 초기 실행 시 설정해야 하는 값을 설명한다.

## FMS 제공 배포 파일 설명

FMS가 제공 배포하는 파일을 설명한다.

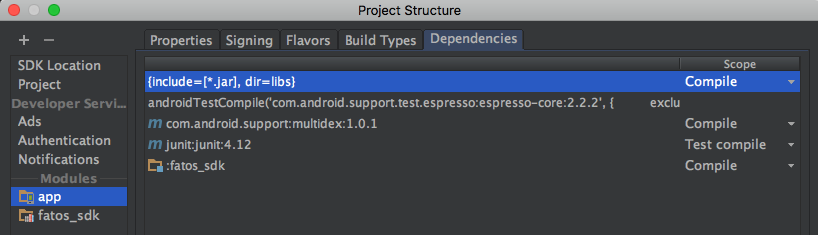
### [FM\_SDK\_FS\_SAM\_0001] FMS 파일 설명

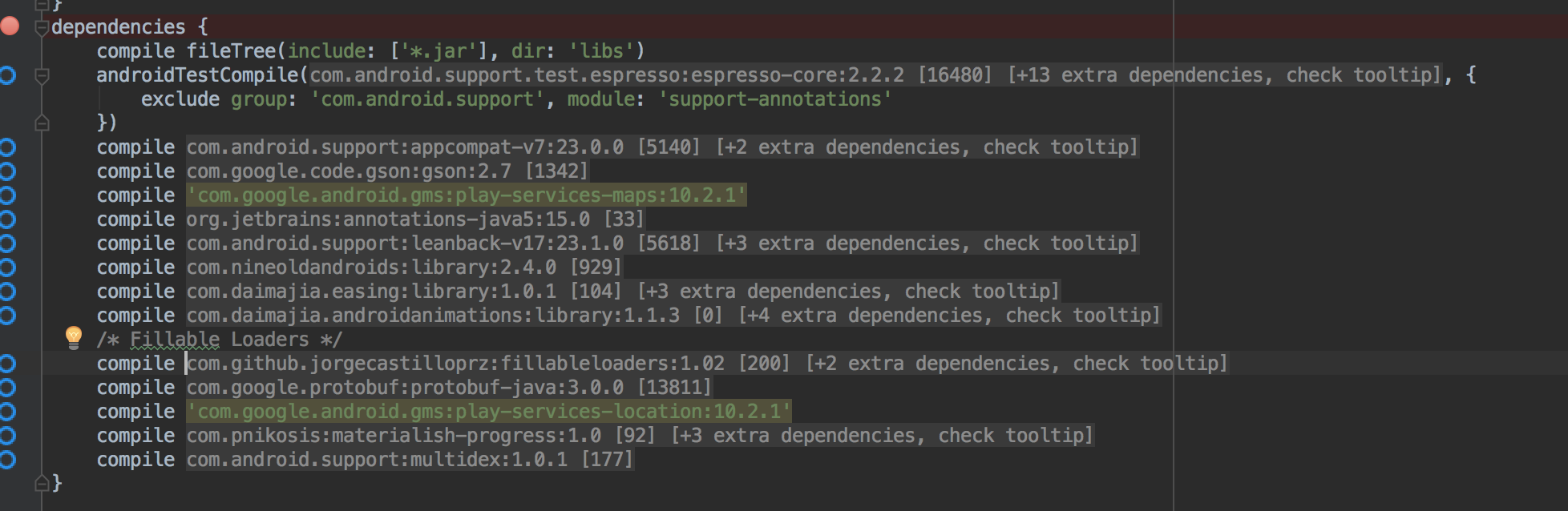
* **“**fatosmap\_sdk\_v1.03.aar” : FMS Android Studio Library File
* “FatosMapBaseSDK” : FMS Sample Project
* “(20170618)\_FatosMap\_V1.03\_SDK” : SDK 설명 파일

## FMS aar Project Link

FMS는 aar file 형태로 제공 된다. 이 aar file을 해당 프로젝트 연결해 사용한다.

### [FM\_SDK\_FS\_SAM\_0002] 프로젝트 설정

* Android Stuido -> “Project Structure” -> ‘+’ 버튼을 눌러 “fatosmap\_sdk\_vx.xx.aar” 추가한다.
* 위치측위 관련, 파일 read/write permission은 할당 되야 한다.
* Module의 App을 선택 -> Dependencies탭을 선택 -> Library dependency -> app에 연결한다.
* 
* Sample Project에서와 같이 compile option을 맞춰 준다.



* FMS는 Project의 Application Class가 ANaviApplication을 상속 받아야 한다.
* App build.gradle에   
  defaultConfig {

..

versionCode FATOSMAP\_VERSION\_CODE

versionName FATOSMAP\_VERSION\_NAME

을 추가해야 하고,

Project build.gradle에   
ext {

FATOSMAP\_VERSION\_CODE = 177

FATOSMAP\_VERSION\_NAME = "v1.0.8"

}

을 추가 해야 한다.

### [FM\_SDK\_FS\_SAM\_0003] Android Studio 3.0 관련

- 17년 11월 01일 기준 Android Studio 3.0에서 aar/jar를 import하지 못하는 문제가 있어   
 수동 import를 권장한다.

* 프로젝트에 libs 폴더가 없을 경우 생성하고 fatos\_sdk…aar 파일을 복사한다. .
* App의 build.gradle에   
  repositories {

flatDir {

dirs 'libs'

}

}

을 추가하고 dependencies에   
compile(name:'fatos\_sdk-release\_1031', ext:'aar')

* 프로젝트의 <manifest 에 xmlns:tools=<http://schemas.android.com/tools> 을 추가한다.
* <application에 tools:replace="android:icon" 을 추가한다.

## FMS App Initialize

연결된 FMS aar을 이용해 App initialize & Destory 과정을 설명한다.

### [FM\_SDK\_FS\_SAM\_0003] App Initialize & Destroy

* FMS Interface에 “FMInterface” Class가 제공되는데 App Init과 Destory등 기능 요소를 담당한다.

( Interface package에 관해서는 3번 항목에서 설명)

App 실행 시 : FMInterface “void FM\_Init(Context context, String strApiKey) ” 호출을 통해 FMS을 Initialize 해야 함.

App 종료 시 : FMInterface “void FM\_Destory() ” 호출을 통해 FMS을 Destory 해야 함.

# FMS Interface & Base Class

본 항목에서는 FMS의 Interface Class와 Base Class를 설명한다.

### [FM\_SDK\_FS\_SAM\_0004] Interface Class와 Base Class

* Interface Class, Base Class

초기화, 검색, 경로탐색, 지도 기능 등등의 FMS의 기능 요소를 정의하고 있음.

|  |  |
| --- | --- |
|  | **public int initFatosNaviEngine()** |
| Parameter & Return | 1 : 성공  INIT\_ERROR\_JNI = -1; // -1: JNI 초기화 실패  INIT\_ERROR\_SDL = -2; // -2 : SDL 초기화 실패  INIT\_ERROR\_RP = -3; // -3 : 네트워크 초기화 실패  INIT\_ERROR\_FORCEUPDATE = -4; // -4 : 강제 업데이트  INIT\_ERROR\_DAL = -5; // -5 : DAL 기타 에러  INIT\_ERROR\_CONFIG = -6; // -6 :컨피그 로딩 실패  INIT\_ERROR\_LOCAL\_SEARCH\_DB = -7;//-7 : 로컬(검색) DB 없음  INIT\_ERROR\_LOCAL\_RP\_DB = -8; //-8 : 도로(검색) DB 없음 |
| 설명 | FMS를 사용하는 App에서 처음 실행 시 FMBaseActivity의 initFatosNaviEngine을 호출해야 한다. |
| 예재 | private int m\_iEngineInit = 0;  @Override  protected void onCreate(Bundle savedInstanceState) {  FatosBuildConfig.buildFATOSauto = false;  super.onCreate(savedInstanceState);  **m\_iEngineInit = initFatosNaviEngine();**  if(m\_iEngineInit != 1)  return;  ..  …  } |

|  |  |
| --- | --- |
|  | **public int FM\_Init(Context context, String strApiKey)** |
| Parameter & Return | Context : Application Context를 전달.  Return : FMError.Success -> 성공  FMError.Error -> 실패 |
| 설명 | FMS를 사용하는 App에서 초기화 시에 반드시 호출 해야 한다. |
| 예재 | private String API\_KEY = "9cc1d283310ac1a5d8c9ee7aaa459a6a";  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_fatos\_map\_base\_sdk);  m\_MainActivity = this;  m\_Context = this;  m\_gApp = (ANaviApplication) m\_Context.getApplicationContext();  **m\_FMInterface.FM\_Init(m\_Context, API\_KEY, new ANaviApplication.MapStatusListener() {**  **@Override**  **public void onRouteFinish() {**  **}**  **@Override**  **public void onMapMove(boolean bMapMove) {**  **// Log.e(TAG,"## onMapMove : " + bMapMove);**  **}**  **});**  ...  …  } |

|  |  |
| --- | --- |
|  | **public int FM\_Destory()** |
| Parameter & Return | Return : FMError.Success -> 성공  FMError.Error -> 실패 |
| 설명 | App 종료 시에 호출 되야 하는 함수. |

|  |  |
| --- | --- |
|  | **public void FM\_SetRouteDriveListener(final ANaviApplication.MapStatusListener mapStatusListener)** |
| Parameter & Return | MapStatusListener |
| 설명 | 경로 안내 종료 / Map 상태 정보에 대한 callback 함수이다.  FM\_Init(..)에서도 설정하는데 필요에 따라 해당 함수로 재설정 가능하다. |

|  |  |
| --- | --- |
|  | **MapStatusListener** |
| Parameter & Return | public void onRouteFinish()  public void onMapMove(boolean bMapMove) |
| 설명 | 경로 안내 종료 / Map 상태 정보에 interface |

|  |  |
| --- | --- |
|  | **public void setMainMapOption(boolean bDrawStartImg,boolean bDrawGoalImg,boolean bDrawPinImg,boolean bDrawEVWare)** |
| Parameter & Return | NativeNavi.MAP\_OBJ\_FONTTBT,   * Font TBT Draw 여부 설정   NativeNavi.MAP\_OBJ\_FLAG\_START,   * 출발지 깃발 Draw 여부 설정   NativeNavi.MAP\_OBJ\_FLAG\_GOAL,   * 목적지 깃발 Draw 여부 설정   NativeNavi.MAP\_OBJ\_FLAG\_PIN   * poi Pin image draw 여부 설정   NativeNavi.MAP\_OBJ\_EVWHERE,   * Map에 전기차 정보 object draw 여부 설정   NativeNavi.MAP\_OBJ\_ADVENCED\_TBT,   * Advance tbt draw 여부 설정   NativeNavi.MAP\_OBJ\_SCALEBAR   * Scale bar draw 여부 설정 |
| 설명 | Map Option 상태를 설정하는 함수 |

* Base Class 연동

FMS를 연동하는 Activity는 FMBaseActivity를 상속 받아야 한다.

FMBaseActivity는 Map Touch 처리/ 중심 처리 등등.. Map Control에 대한 사항을 담당한다.

# FMS Map Drawing

본 항목에서는 FMS으로 Map Drawing하는 과정을 설명한다.

### [FM\_SDK\_FS\_SAM\_0005] Map Fragment

* FMS Init 후에 App단에서는 AMapBaseFragment로 부터 상속 받은 Fragment를 생성해야 한다.

Ex)

public class FatosMapFragment extends AMapBaseFragment

implements FatosMapBaseSDKActivity.OnFatosMapListener,

FatosMainMapView.OnFatosMapListener {

…

…

}

그 후 Map을 그리고자 하는 Activity에서 다음과 같이

<biz.fatos.exlib.fatoslayout.AMapSCLinearLayout

android:layout\_width="300dp"

android:layout\_height="300dp"

android:layout\_alignParentBottom="true"

android:layout\_centerHorizontal="true"

android:layout\_marginBottom="55dp"

android:background="@color/titlebar\_color"

android:gravity="center"

android:orientation="horizontal">

<fragment

android:name="kr.fatos.fatosmapbasesdk.FatosMapFragment"

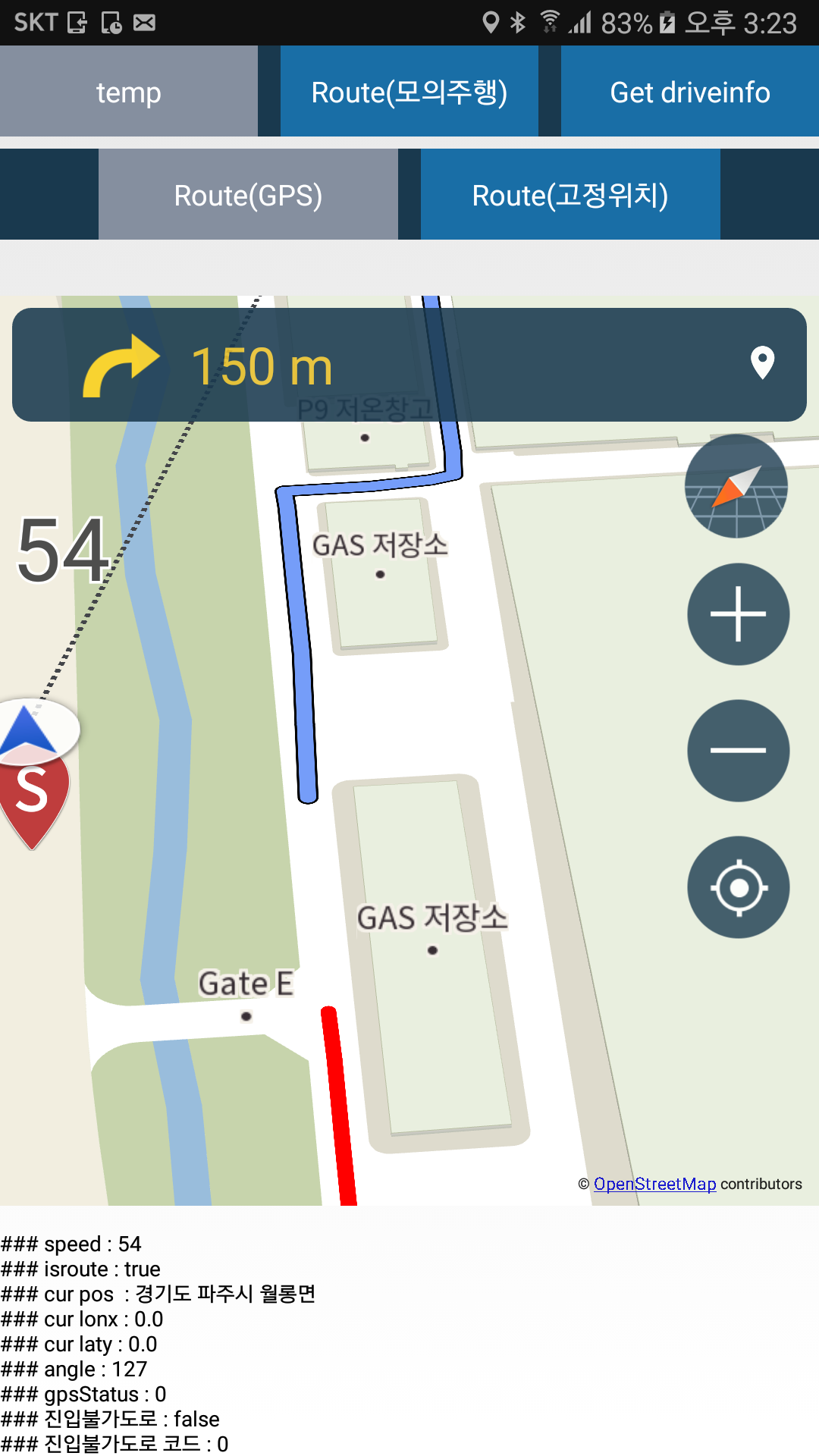
android:id="@+id/map\_fragment"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

</biz.fatos.exlib.fatoslayout.AMapSCLinearLayout>

설정 할 경우 기본 Map이 그려지게 된다.



* 하지만 Map Flicking, Two Touch Zoom In/out 등 지도 기능이 동작하기 위해서는 callback 함수에 대한 기능 정의가 되어야 한다.

|  |  |
| --- | --- |
|  | **public void onMapReady()** |
| 설명 | FMS를 통해 Map 생성 후 Map Drawing할 준비가 되어 있을 때 오는 Callback 함수로 해당 Callback을 받은 수 Map Control Api를 호출해야 한다. |
| 예재 | @Override  public void onMapReady() {  ((FMBaseActivity)getActivity()).onMapReady();  } |

|  |  |
| --- | --- |
|  | **public void onMapAnimation(MapAnimation aniInfo)** |
| Parameter & Return | aniInfo : Map animate Option 정보 |
| 설명 | Map Animate 효과를 주기 위한 API |
| 예재 | @Override  public void onMapAnimation(MapAnimation aniInfo) {  if(mView != null)  mView.setAniInfo(aniInfo);  } |

|  |  |
| --- | --- |
|  | **public class MapAnimation** |
| 설명 | Map Animate Option  public static final int MAP\_ANI\_NOT\_USE = -1;   * 사용하지 않음.   public static final int MAP\_ANI\_TYPE\_EVEN = 0;   * 이동 속도가 동일함.   public static final int MAP\_ANI\_TYPE\_ACCEL = 1;   * 지원예정 (첨엔 느리다가 점점 빠르게)   public static final int MAP\_ANI\_TYPE\_DCCEL = 2;   * (첨엔 빠르다가 점덤 느리게)   public static final int MAP\_ANI\_TYPE\_DIRECT= 3;   * 바로이동.   public static final int MAP\_ANI\_TYPE\_CURPOS= 4;   * 현위치 이동 .   public static final int MAP\_ANI\_TYPE\_CUSTOM\_ZOOMINOUT = 5;   * ani zoom in/out .   public float m\_fMap\_yCenter;  public float m\_fMap\_xCenter;  public int m\_nAniType\_Center;   * 지도 중심 관련 설정 값   public float m\_fTargetTilt;  public int m\_nAniType\_Tilt;   * 지도 틸트. (현재 지도가 진북모드일경우 무시됨)   public float m\_fTargetAngle;  public int m\_nAniType\_Angle;   * 지도 각도. (맵매칭 사용중에는 무시됨)   public float m\_fVIALevel;  public float m\_fGoalLevel;  public int m\_nAniType\_Level;   * 레벨의 경우 2단계 Animation이 가능하도록 한다. * m\_fVIALevel 0일 경우 사용하지 않음. |

|  |  |
| --- | --- |
|  | **public void FM\_MoveCurPosition(FMBaseActivity.OnFatosMapListener onFatosMapListener)** |
| 설명 | 현 위치로 이동 |
| 예재 | m\_FMInterface.FM\_MoveCurPosition(FMBaseActivity.OnFatosMapListener) |

|  |  |
| --- | --- |
|  | **public void FM\_SetMapAirMode(int nMapType)** |
| Parameter & Return | nMapType :  FMInterface.MAPMODE\_AIR\_ON\_BUILDING\_ON; // 항공모드 on 건물 on  FMInterface.MAPMODE\_AIR\_ON\_BUILDING\_OFF; // 항공모드 on 건물 off  FMInterface.MAPMODE\_AIR\_OFF\_BUILDING\_ON; // 항공모드 off 건물 on |
| 설명 | 항공모드 설정 |
| 예재 | m\_FMInterface.FM\_SetMapAirMode(FMInterface.MAPMODE\_AIR\_ON\_BUILDING\_ON) |

|  |  |
| --- | --- |
|  | **public void FM\_EnableSmartZoomInOut(Boolean bEnable)** |
| Parameter & Return | bEnable  true : 자동 주행 모드 on  false : 자동 주행 모드 off |
| 설명 | 주행 중 자동 축척 모드 사용 할지에 대한 여부 설정 |
| 예재 | m\_FMInterface.FM\_EnableSmartZoomInOut (true); |

|  |  |
| --- | --- |
|  | **public void FM\_EnableWalkingMode(Boolean bEnable)** |
| Parameter & Return | bEnable  true : 도보 모드 on  false : 도보 모드 off |
| 설명 | 도보 모드 사용 할지에 대한 여부 설정 |
| 예재 | m\_FMInterface.FM\_EnableWalkingMode (true); |

|  |  |
| --- | --- |
|  | **public void FM\_ViewFirePOI(int[] arType, Boolean[] arMapShow, Boolean[] arTextShow)** |
| Parameter & Return | arType : 주제도 Type  public final static int BASEMAP\_LAYER\_FIRE\_NONE = 0; // 없는 레이어  public final static int BASEMAP\_LAYER\_FIRE\_WATER = 1; // 용수  public final static int BASEMAP\_LAYER\_FIRE\_OBJECT = 2; // 대상물  public final static int BASEMAP\_LAYER\_FIRE\_DANGEROUS = 3; // 위험물  …  …  arMapShow : 각 주제도 지도 표출 여부 설정  arTextShow : 각 주제도 텍스트 표출 여부 설정 |
| 설명 | 용수, 대상물,위험물 등의 지도 표출 여부를 설정한다. |
| 예재 | private int[] m\_arBaseLayerOption= new int[4];  private boolean[] m\_arBaseLayerVisibleOption = new boolean[4];  ..  m\_FMInterface.FM\_ViewCustomPOI (arBaseLayerOption, arBaseLayerVisibleOption); |

|  |  |
| --- | --- |
|  | **public ArrayList<POI2\_INFO>** **FM\_SearchFireObj(int nType, int nRadius,** **double xlon, double ylat)** |
| Parameter & Return | # Param  nType : 주제도 Type  public final static int BASEMAP\_LAYER\_FIRE\_NONE = 0; // 없는 레이어  public final static int BASEMAP\_LAYER\_FIRE\_WATER = 1; // 용수  public final static int BASEMAP\_LAYER\_FIRE\_OBJECT = 2; // 대상물  public final static int BASEMAP\_LAYER\_FIRE\_DANGEROUS = 3; // 위험물  ….  …  nRadius : 검색 반영 (m 단위)  xlon, ylat : 반경 검색 기준 좌표  # Return  ArrayList<POI2\_INFO> : 주제도 Type에 대한 검색 데이터 리스트 |
| 설명 | 용수, 대상물,위험물 등의 주제도를 기준 좌표로 반경 검색한다. . |
| 예재 |  |

|  |  |
| --- | --- |
|  | **public POI2\_INFO** **FM\_SearchTelephonePoleObj(int nID)** |
| Parameter & Return | # Param  nType : 전신주 ID  # Return  전신주 POI 정보 |
| 설명 | 해당 ID 기준 전신주 데이터를 검색한다. |
| 예재 |  |

|  |  |
| --- | --- |
|  | **public int**  **FM\_FireGetAddress(Handler \_resultHandler, double xlon, double ylat)** |
| Parameter & Return | #Param \_resultHandler : 검색 결과 또는 상태를 받을 handler  xlon, ylat : 주소를 얻으려는 좌표 |
| 설명 | 기준 좌표 기준 행정동명을 얻을 때. |
| 예재 |  |

|  |  |
| --- | --- |
|  | **public int**  **FM\_GetPosFromScreen(int x, int y, double[] xlonlat)** |
| Parameter & Return | #Param x,y : 지도 터치한 화면 position  xlonlat : 터치 위치에 대한 wgs84 좌표 |
| 설명 | 화면 터치 point의 wgs84 좌표를 반환 한다. |
| 예재 |  |

|  |  |
| --- | --- |
|  | **public int FM\_SearchPOI\_HybridVol2(Handler \_resultHandler,String strSearchName, boolean bSaveRecent)** |
| Parameter & Return | \_resultHandler : 검색 결과 또는 상태를 받을 handler  strSearchName : POI 검색 명칭  bSaveRecent : 최근 검색 결과 목록에 저장 할지 여부 |
| 설명 | 특정 명칭으로 SK Poi와 국가 주요 POI의 검색 결과를 얻고자 할 때의 API  통신가능상태가 아닐 때는 local 탐색 한다. |
| 예재 | m\_FMInterface.FM\_SearchPOI\_HybridVol2(new HttpResultHandler(FatosMapBaseSDKActivity.this),"서울",false);  static class HttpResultHandler extends Handler {  private final WeakReference<FatosMapBaseSDKActivity> mActivity;  HttpResultHandler(FatosMapBaseSDKActivity activity) {  mActivity = new WeakReference<FatosMapBaseSDKActivity>(activity);  }  @Override  public void handleMessage(Message msg) {  FatosMapBaseSDKActivity activity = mActivity.get();  if (activity != null) {  activity.handleMessage(msg);  }  }  }  private void handleMessage(Message msg) {  String result = msg.getData().getString(AMapGoogleSearchUtil.RESULT);  ArrayList<String> searchList = new ArrayList<String>();  if (result.equals(ErrorMessage.SUCCESS\_SEARCH\_SUCCESS)) {  Toast.makeText(m\_Context,"SUCCESS\_SEARCH\_SUCCESS",Toast.LENGTH\_SHORT).show();  }  else{  Toast.makeText(m\_Context,result,Toast.LENGTH\_SHORT).show();  }  } |

|  |  |
| --- | --- |
|  | **public void FM\_SetMapFireObjPosition(FMBaseActivity.OnFatosMapListener onFatosMapListener,int nType, double xlon, double ylat)** |
| Parameter & Return | #Param  onFatosMapListener : Mapview와 연결된 listener  nType : 주제도 Type  public final static int POI\_LAYER\_FIRE\_NONE = 0; // 없는 레이어  public final static int POI \_LAYER\_FIRE\_WATER = 1; // 용수  public final static int POI \_LAYER\_FIRE\_OBJECT = 2; // 대상물  public final static int POI\_LAYER\_FIRE\_DANGEROUS = 3; // 위험물  public final static int POI\_LAYER\_FIRE\_SK = 4; // SK POI  public final static int POI\_LAYER\_FIRE\_KOREA = 5; // 국가 POI  public final static int POI\_LAYER\_FIRE\_ACCIDENT = 6; // 재난 위치  public final static int POI\_LAYER\_FIRE\_APPLICANT = 7; // 재난 위치  xlon, ylat : 이동 하려는 위치의 WGS84 좌표 |
| 설명 | 해당 좌표로 해당 Type의 핀 이미지 기반으로 지도 위치를 옮긴다. |
| 예재 | double dLonX = 126.767723;  double dLatY = 37.809287; m\_FMInterface.FM\_SetMapFireObjPosition(FMBaseActivity.onFatosMapListener, POI\_LAYER\_FIRE\_SK ,dLonX,dLatY); |

|  |  |
| --- | --- |
|  | **public int**  **FM\_ChangeAccidentPos(int nID, int x, int y)** |
| Parameter & Return | #Param  nID : 재난 위치 ID x,y : 변경 할 위치 좌표 |
| 설명 | 해당 재난 위치를 변경한다. . |
| 예재 |  |

|  |  |
| --- | --- |
|  | **public ArrayList<GPSPOS>**  **FM\_GetLastGPSTrace()** |
| Parameter & Return | #Return  ArrayList<GPSPOS> : GPS 궤적 정보 List |
| 설명 | 경로안내 시작 부터의 GPS 궤적 정보를 가져온다. . |
| 예재 |  |

|  |  |
| --- | --- |
|  | **public void**  **FM\_DrawGrid(double[] x, double[] y)** |
| Parameter & Return | #Param  x,y : 그리드 좌표 리스트 |
| 설명 | 지도에 그리드를 그린다. |
| 예재 |  |

# FMS Route

본 항목에서는 FMS의 이용해 경로탐색(Route) 관련해서 설명한다.

## FMS POI Route

FMS Route는 T map 경로 기준이며, 출발지 설정을 할 수 있다.

### [FM\_SDK\_FS\_SAM\_0006] FMS POI Search Interface와 예제

|  |  |
| --- | --- |
|  | **public void FM\_RouteVol2\_Hybrid(Handler \_resultHandler ,FMRoutePos startPos, FMRoutePos endPos,FMRouteOption routeOpt)** |
| Parameter & Return | \_resultHandler : 검색 결과 또는 상태를 받을 handler  startPos : 출발지 정보 ( null 일 경우 GPS 기준 현 위치 )  endPos : 목적지 정보  routeOpt : 추후 기능 제공 예정. |
| 설명 | 통신 가능 상태 : 서버 경로 탐색 (Center Mode)  통신 불가능 상태 : 로컬 경로 탐색 (Local Mode)  경로 요청한 출발지 / 목적지로 탐색 후 자동 경로 주행 모드로 전환된다. |
| 예재 | FMRoutePos startPos = new FMRoutePos();  FMRoutePos endPos = new FMRoutePos();  FMRouteOption routeOpt = new FMRouteOption();  startPos.dLonX = 127.3221;  startPos.dLatY = 36.1235;  startPos.strPosName = "LG";  startPos.strAddr = "1F";  endPos.dLonX = 127.5221;  endPos.dLatY = 36.5235;  endPos.strPosName = "LG";  endPos.strAddr = "2F";  endPos.carType = 1;  m\_FMInterface.FM\_RouteVol2\_LG(startPos,endPos,routeOpt);  m\_FMInterface.FM\_RouteVol2\_Hybrid(new HttpResultHandler(FatosMapBaseSDKActivity.this),startPos,endPos,routeOpt); |

|  |  |
| --- | --- |
|  | **public void FM\_RouteVol2\_Local(FMRoutePos startPos, FMRoutePos endPos,FMRouteOption routeOpt)** |
| Parameter & Return | startPos : 출발지 정보 ( null 일 경우 GPS 기준 현 위치 )  endPos : 목적지 정보  routeOpt : 추후 기능 제공 예정. |
| 설명 | 로컬 모드로 경로요청한 출발지 / 목적지로 탐색 후 자동 경로 주행 모드로 전환된다. |
| 예재 | FMRoutePos startPos = new FMRoutePos();  FMRoutePos endPos = new FMRoutePos();  FMRouteOption routeOpt = new FMRouteOption();  startPos.dLonX = 127.3221;  startPos.dLatY = 36.1235;  startPos.strPosName = "LG";  startPos.strAddr = "1F";  endPos.dLonX = 127.5221;  endPos.dLatY = 36.5235;  endPos.strPosName = "LG";  endPos.strAddr = "2F";  endPos.carType = 1;  m\_FMInterface.FM\_RouteVol2\_LG(startPos,endPos,routeOpt); |

|  |  |
| --- | --- |
|  | **public void FM\_RouteVol2\_Center(FMRoutePos startPos, FMRoutePos endPos,FMRouteOption routeOpt)** |
| Parameter & Return | startPos : 출발지 정보 ( null 일 경우 GPS 기준 현 위치 )  endPos : 목적지 정보  routeOpt : 추후 기능 제공 예정. |
| 설명 | 서버 모드 기준으로 경로요청한 출발지 / 목적지로 탐색 후 자동 경로 주행 모드로 전환된다. |
| 예재 | FMRoutePos startPos = new FMRoutePos();  FMRoutePos endPos = new FMRoutePos();  FMRouteOption routeOpt = new FMRouteOption();  startPos.dLonX = 127.3221;  startPos.dLatY = 36.1235;  startPos.strPosName = "LG";  startPos.strAddr = "1F";  endPos.dLonX = 127.5221;  endPos.dLatY = 36.5235;  endPos.strPosName = "LG";  endPos.strAddr = "2F";  endPos.carType = 1;  m\_FMInterface.FM\_RouteVol2\_LG(startPos,endPos,routeOpt); |

|  |
| --- |
| **public void FM\_RouteVol2\_LG(FMRoutePos startPos, FMRoutePos endPos,FMRouteOption routeOpt)** |
| startPos : 출발지 정보 ( null 일 경우 GPS 기준 현 위치 )  endPos : 목적지 정보  routeOpt : cartype과 같은 경로 옵션 정보 . |
| 서버 모드 기준으로 경로요청한 출발지 / 목적지로 탐색 후 자동 경로 주행 모드로 전환된다. |
| FMRoutePos startPos = new FMRoutePos();  FMRoutePos endPos = new FMRoutePos();  FMRouteOption routeOpt = new FMRouteOption();  startPos.dLonX = 127.3221;  startPos.dLatY = 36.1235;  startPos.strPosName = "LG";  startPos.strAddr = "1F";  endPos.dLonX = 127.5221;  endPos.dLatY = 36.5235;  endPos.strPosName = "LG";  endPos.strAddr = "2F";  endPos.carType = 1;  m\_FMInterface.FM\_RouteVol2\_LG(startPos,endPos,routeOpt); |

|  |  |
| --- | --- |
|  | **public class FMRoutePos** |
| Parameter & Return | public double dLon = 0;  public double dLat = 0;   * 해당 위치 경/위도 좌표   public String strPosName = "";   * 명칭 정보   public String strAddr = "";   * 주소 정보 |
| 설명 | 경로 요청할 해당 위치에 대한 정보를 설정한다. . |

|  |  |
| --- | --- |
|  | **public void FM\_StartRGService(FMBaseActivity.OnFatosMapListener onFatosMapListener)** |
| Parameter & Return | onFatosMapListener : Mapview와 연결된 listener |
| 설명 | 경로요약에서 경로 선택 후 주행 화면으로 갈 때 호출해야 한다. |

|  |  |
| --- | --- |
|  | **public ArrayList<RouteSummaryData> FM\_RouteSummary(FMBaseActivity.OnFatosMapListener onFatosMapListener)** |
| Parameter & Return | Parm: onFatosMapListener : Mapview와 연결된 listener  Return : 각 경로 옵션에 대한 정보 list |
| 설명 | 경로 탐색 성공 후에 각 경로 옵션에 대한 정보를 얻기 위해 호출한다. |

|  |  |
| --- | --- |
|  | **public void FM\_SelectRoute(int nType)** |
| Parameter & Return | # Parm  nType : RouteSummaryData의 type 정보 |
| 설명 | 경로요약 화면에서 경로 옵션에 대해 선택 시 호출해 준다. |

|  |  |
| --- | --- |
|  | **public class RouteSummaryData** |
| Parameter & Return | 경로 요약 정보 |
| 설명 | POI 검색 결과 구조체  public String strTypeName;   * 경로탐색 옵션 타이틀   public int nType;   * 경로탐색 옵션 (최소, 최단, 추천 등)   public int nLength;   * 경로 총 길이 (단위 미터)   public int nTime;   * 경로 총 소요시간 (단위 초)   public int nFee;   * 요금 정보 (원)   public int nAvgSpeed;   * 구간 평균 속도   public int nTurnCongestion;   * 구간 혼잡도   public int nOptionColor;   * 경로 옵션 Color |

|  |  |
| --- | --- |
|  | **public ArrayList<RouteSummaryDataDetail> FM\_RouteSummaryDetail(int nType, int nFromIdx)** |
| Parameter & Return | # Parm  nType : RouteSummaryData의 type 정보  nFromIdx : 현재 RG Index  ( nFromIdx는 샘플과 같이 public void updateServiceLinkIdx(int nIdx, boolean bReset)에서 얻어야 한다. ) |
| 설명 | 각 Type별 세부 경로 정보를 전달한다.. |

|  |  |
| --- | --- |
|  | **public class RouteSummaryDataDetail** |
| Parameter & Return | 경로 요약 상세 정보 |
| 설명 | public int nTbtCode;   * tbt code (1~53) * SDK에 포함 된. R.drawable.list\_01\_straight ~ R.drawable.list\_53\_via3 사용   public int nLength;   * 해당 Turn 길이 (단위 미터)   public int nTime;   * 해당 Turn 통과시간 (단위 초)   public double dLonX;  public double dLatY;   * Turn 지점 좌표 정보   public String strDongName;   * Turn 지점 행정동명   public int nTurnCongestion;   * 구간 혼잡도   public int nAvgSpeed;   * 구간 평균 속도   public String strTurnInfoTextName;   * Turn Text Info |

|  |  |
| --- | --- |
|  | **public ArrayList<RouteSummaryDataDetail> FM\_RouteSummaryDetail(int nType)** |
| Parameter & Return | # Parm  nType : RouteSummaryData의 type 정보 |
| 설명 | 각 Type별 세부 경로 전체 정보를 전달한다.. |

|  |  |
| --- | --- |
|  | **public void FM\_CancelRoute()** |
| Parameter & Return |  |
| 설명 | 경로가 있을 경우 해당 경로를 취소한다.  (경로가 취소 되면서 경로선/출,목 깃발 등을 그리지 않는다)  취소 후에는 onRouteFinish Callback 호출 된다. |

|  |  |
| --- | --- |
|  | **public void FM\_SetSimulationSpeed(int nSpeed)** |
| Parameter & Return | nSpeed : 모의주행 속도 |
| 설명 | 모의 주행 중 속도를 설정한다. |

|  |  |
| --- | --- |
|  | **public List<RoutePosition>FM\_GetViaPOIList()** |
| Parameter & Return | Return : 좌표 정보를 담을 list return |
| 설명 | 경유지 탐색을 위한 Poi 정보 list로 경로탐색 요청시 최소 2개 이상 설정 되어 있어야 한다. |

|  |  |
| --- | --- |
|  | **public void FM\_RouteVol2\_Via(Handler \_resultHandler, List<RoutePosition> m\_PoiList)** |
| Parameter & Return | \_resultHandler : 검색 결과 또는 상태를 받을 handler  m\_PoiList : 출,경,목 poi 정보 List |
| 설명 | M\_PoiList에 설정된 좌표 기준으로 경유지 탐색 한다.  ( 0 idx : 출발지 , 마지막 idx : 목적지, 그 외 경유지)  \_resultHandler의 에러 코드 중 FMError.FME\_ERROR\_VIACOUNT\_ROUTE는 좌표 리스트가 2개 미만 일 때 발생함.  Ex)  RoutePosition positionList0 = new RoutePosition();  positionList0.x = 0;  positionList0.y = 0;  positionList0.name = "출발지";  positionList.add(positionList0);  RoutePosition positionList1 = new RoutePosition();  positionList1.x = 127.46534606;  positionList1.y = 37.28005176;  positionList1.name = "이천시법원";  positionList1.addr = "경기 이천시 안흥동";  positionList.add(positionList1);  RoutePosition positionList2 = new RoutePosition();  positionList2.x = 127.43412687;  positionList2.y = 37.27288535;  positionList2.name = "이천시청";  positionList2.addr = "경기 이천시 중리동";  positionList.add(positionList2);  RoutePosition positionList3 = new RoutePosition();  positionList3.x = 127.46534606;  positionList3.y = 37.28005176;  positionList3.name = "이천종합버스터미널";  positionList3.addr = "경기 이천시 안흥동";  m\_FMInterface.FM\_RouteVol2\_Via(new  HttpResultHandler(FatosMapSDKSampleLandFragment.this),positionList); |
|  |  |

|  |  |
| --- | --- |
|  | **public class RoutePosition** |
| Parameter & Return |  |
| 설명 | 경유지 poi의 좌표를 담고 있는 구조체  아래의 정보 이외의 정보는 SDK 상에서 쓰지 않는다.  public int nPosKind; // 0 출발지 , 1목적지, 2 입력 대기  public double x = 0;  public double y = 0;   * Poi wgs84 좌표   public String name; // 명칭 ex) “강남역(2호선)”  public int nPoiID; // POI ID (없을 경우 0)  public String poiFlag; // POI RPFlag ( 없을 경우 0)  public int nFloor = 1; // 층수(1=1층, -2=지하2층, ...)  public boolean bDir = true; // 방향성체크  public String addr; // 주소 정보(없을 경우 null)  public Boolean bPassingPoint = false; // 경로만 잡는 경유지 인지 설정 |

|  |  |
| --- | --- |
|  | **public Boolean FM\_GoNextGoal()** |
| Parameter & Return | # Parm  return : 정상적으로 다음경유지로 설정된 경우 : true  다음경유지로 설정하지 못하는 경우 : false  (목적지 밖에 안 남았을 경우) |
| 설명 | 경유지 주행에서 바로 앞 경유지를 패스하고 다음 경유지 부터 경로 잡고 주행 할 때 사용한다. |

# FMS POI Search

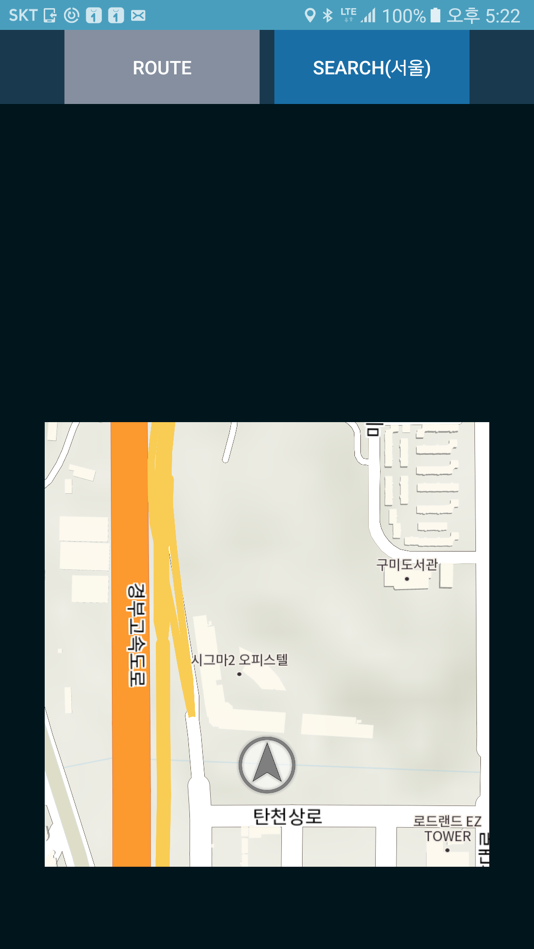
본 항목에서는 FMS의 이용해 POI Search 관련해서 설명한다.

## FMS POI Search

FMS Search는 하나의 POI 명칭에 “T map”의 검색 결과를 제공한다.

### [FM\_SDK\_FS\_SAM\_0007] FMS POI Search Interface와 예제

* Sample Project의 “서울” 검색 기준으로 설명



|  |  |
| --- | --- |
|  | **public int FM\_SearchPOI\_Hybrid(Handler \_resultHandler,String strSearchName, boolean bSaveRecent)** |
| Parameter & Return | \_resultHandler : 검색 결과 또는 상태를 받을 handler  strSearchName : POI 검색 명칭  bSaveRecent : 최근 검색 결과 목록에 저장 할지 여부 |
| 설명 | 특정 명칭으로 어디야,google,naver, tmap등의 검색 결과를 얻고자 할 때의 API  통신가능상태가 아닐 때는 local 탐색 한다. |
| 예재 | m\_FMInterface.FM\_SearchPOI(new HttpResultHandler(FatosMapBaseSDKActivity.this),"서울",false);  static class HttpResultHandler extends Handler {  private final WeakReference<FatosMapBaseSDKActivity> mActivity;  HttpResultHandler(FatosMapBaseSDKActivity activity) {  mActivity = new WeakReference<FatosMapBaseSDKActivity>(activity);  }  @Override  public void handleMessage(Message msg) {  FatosMapBaseSDKActivity activity = mActivity.get();  if (activity != null) {  activity.handleMessage(msg);  }  }  }  private void handleMessage(Message msg) {  String result = msg.getData().getString(AMapGoogleSearchUtil.RESULT);  ArrayList<String> searchList = new ArrayList<String>();  if (result.equals(ErrorMessage.SUCCESS\_SEARCH\_SUCCESS)) {  Toast.makeText(m\_Context,"SUCCESS\_SEARCH\_SUCCESS",Toast.LENGTH\_SHORT).show();  }  else{  Toast.makeText(m\_Context,result,Toast.LENGTH\_SHORT).show();  }  } |

|  |  |
| --- | --- |
|  | **public int FM\_SearchPOI(Handler \_resultHandler,String strSearchName, boolean bSaveRecent)** |
| Parameter & Return | \_resultHandler : 검색 결과 또는 상태를 받을 handler  strSearchName : POI 검색 명칭  bSaveRecent : 최근 검색 결과 목록에 저장 할지 여부 |
| 설명 | 특정 명칭으로 어디야,google,naver, tmap등의 검색 결과를 얻고자 할 때의 API |
| 예재 | m\_FMInterface.FM\_SearchPOI(new HttpResultHandler(FatosMapBaseSDKActivity.this),"서울",false);  static class HttpResultHandler extends Handler {  private final WeakReference<FatosMapBaseSDKActivity> mActivity;  HttpResultHandler(FatosMapBaseSDKActivity activity) {  mActivity = new WeakReference<FatosMapBaseSDKActivity>(activity);  }  @Override  public void handleMessage(Message msg) {  FatosMapBaseSDKActivity activity = mActivity.get();  if (activity != null) {  activity.handleMessage(msg);  }  }  }  private void handleMessage(Message msg) {  String result = msg.getData().getString(AMapGoogleSearchUtil.RESULT);  ArrayList<String> searchList = new ArrayList<String>();  if (result.equals(ErrorMessage.SUCCESS\_SEARCH\_SUCCESS)) {  Toast.makeText(m\_Context,"SUCCESS\_SEARCH\_SUCCESS",Toast.LENGTH\_SHORT).show();  }  else{  Toast.makeText(m\_Context,result,Toast.LENGTH\_SHORT).show();  }  } |

|  |  |
| --- | --- |
|  | **public int FM\_SearchPOI\_Local(String strSearchName)** |
| Parameter & Return | strSearchName : POI 검색 명칭 |
| 설명 | 특정 명칭으로 Local 기준 검색 결과를 얻고자 할 때의 API |
| 예재 | ArrayList<POI2\_INFO> poiSearchResultList;  poiSearchResultList = m\_FMInterface.FM\_SearchPOI\_Local("서울");  Log.d("FATOS", "search size : " + poiSearchResultList.size()); |

|  |  |
| --- | --- |
|  | **public int FM\_RecommendWord(Handler \_resultHandler,String strSearchName)** |
| Parameter & Return | strSearchName : 추천어 검색 문자(초성포함) |
| 설명 | 초성 포함 검색 문자에 따른 추천어 리스트를 준다. |
| 예재 | m\_FMInterface.FM\_RecommendWord(new HttpResultHandler(FatosMapSDKSampleFragment.this),"ㄱ"); |

|  |  |
| --- | --- |
|  | **public ArrayList<POIItem> FM\_GetSearchResult(int nSortOption)** |
| Parameter & Return | nSortOption  0 : 정확도 검색  1 : 거리순 검색 |
| 설명 | 통합 검색 결과를 가져 올 때 쓴다. nSortOption 값에 따라 정렬한다. |
| 예재 | if (result.equals(FMError.FME\_SUCCESS\_SEARCH\_SUCCESS)) {  ArrayList<POIItem> searchResultPOI = m\_FMInterface.FM\_GetSearchResult(FMSortOption.FM\_SORT\_BY\_ACCURACY);  if(searchResultPOI != null)  Toast.makeText(m\_Context,"FME\_SUCCESS\_SEARCH\_SUCCESS Size : " + searchResultPOI.size(),Toast.LENGTH\_SHORT).show();  else  Toast.makeText(m\_Context,"searchResultPOI is null",Toast.LENGTH\_SHORT).show();  } |

|  |  |
| --- | --- |
|  | **public ArrayList<String> FM\_GetRecommentWordResult()** |
| Parameter & Return |  |
| 설명 | 추천어 검색 결과를 가져 온다. |
| 예재 | if(result.equals(FMError.FME\_MESSAGE\_STATE\_POI\_AUTO))  {  ArrayList<String> arDessert = m\_FMInterface.FM\_GetRecommentWordResult();  if(arDessert != null)  Toast.makeText(m\_Context,"FME\_MESSAGE\_STATE\_POI\_AUTO Size : " + arDessert.size(),Toast.LENGTH\_SHORT).show();  else  Toast.makeText(m\_Context,"arDessert is null",Toast.LENGTH\_SHORT).show();  } |

|  |  |
| --- | --- |
|  | **public class POI2\_INFO** |
| Parameter & Return | Local 검색 결과 구조체 |
| 설명 | POI 검색 결과 구조체  // poi data items  public String strPoiID;   * POI ID   public String name;   * 검색 결과 명칭   public String telNo;   * 전화 번호   public VERTEX32 wpCenter;   * poi 중심 좌표   public VERTEX32 wpGuide;   * poi 입구점 좌표     public String upperBizName;  public String lowerBizName;   * 업종 정보   public String rpFlag;   * rpFlag |

|  |  |
| --- | --- |
|  | **public class POIItem** |
| Parameter & Return | 검색 방법에 따라 POIItem으로 전달 될 수 있다. |
| 설명 | POI 검색 결과 구조체  // poi data items  public String id = ""; // poi id  public String name = ""; // poi name  public String telNo; // poi telephone  public String frontLat = null; // 입구점 좌표  public String frontLon = null; // 입구점 좌표  public String noorLat; // 중심 좌표  public String noorLon; // 중심 좌표  …  …  public int nFromCurPosDist; // 현위치에서 해당 위치까지의 거리 (meter 단위) |

|  |  |
| --- | --- |
|  | **public int FM\_GetAddress(Handler \_resultHandler, double xlon, double ylat)** |
| Parameter & Return | #Param \_resultHandler : 검색 결과 또는 상태를 받을 handler  xlon, ylat : 주소를 얻으려는 좌표 |
| 설명 | 특정 좌표 Local 기준 검색 결과를 얻고자 할 때의 API |
| 예재 | # 샘플 앱 참고 |

# FMS ETC

본 항목에서는 FMS의 이용해 부가적인 기능에 대해 설명한다.

## Get GPS Speed

GPS 속도를 얻어온다. .

|  |  |
| --- | --- |
|  | **public int FM\_GetGpsSpeed()** |
| Parameter & Return | Return : 현재 수신 된 GPS 속도(km/h) |
| 설명 | 특정 명칭으로 어디야,google,naver, tmap등의 검색 결과를 얻고자 할 때의 API |
| 예재 | Int nGpsSpeed = m\_FMInterface.FM\_GetGpsSpeed(); |

## Get Drive Info

주행 중 행정동명, 좌표 정보, gps 속도 등등의 정보를 얻는다. . .

|  |  |
| --- | --- |
|  | **public FMDriveInfo FM\_GetDriveInfo()** |
| Parameter & Return | Return : FMDriveInfo 구조체 리턴 |
| 설명 | 주행 중 또는 일반 맵 상태에서 현재 위치 좌표/행정동명 등의 정보를 얻는다. |
| 예재 | FMDriveInfo driveInfo = m\_FMInterface.FM\_GetDriveInfo(); |

|  |
| --- |
| **public class FMDriveInfo** |
| 주행 정보를 담고 있는 구조체 |
| private int m\_iGpsSpeed;   * gps speed   private boolean m\_bIsRoute;   * true: 경로 주행 중 false : 일반 주행 중   private String m\_strCurPosName;   * 주행 중인 행정동명   private double m\_fCurLonX = -180;   * 현위치 좌표(경도, x)   private double m\_fCurLatY = -180;   * 현위치 좌표(위도, y)   private int m\_nCurAngle = 0;   * 현위치 좌표(각도)   private int m\_nGpsStatus = 0;   * gps 상태 값   0 : bad  1 : simulation  2 : good  3: test  good 일 때의 좌표만 유효한 값임.  private boolean m\_bCannotEnterRoad = false;   * 진입 불가 도로   private int m\_nCannotEnterRoadCode = 0;   * 진입 불가 도로   private int m\_nGateNearbyDist = 0;   * 게이트 주변 거리   private int m\_nTotalRemainderDist= 0;   * 목적지까지 남은 거리   private int m\_nServiceLinkRemainderTime= 0;   * 목적지까지 남은 시간   public int nViaTotalCount = 0;   * 경유 목적지 전체 수   public int[] listViaRemainderDist;   * 경유 목적지까지 남은 거리(미터 단위)   public int[] listViaRemainderTime;   * 경유 목적지까지 남은 시간(초 단위)   private int m\_nTripTime = 0;   * 총 주행 시간   private int m\_nTripDist= 0;   * 총 주행 거리 |

|  |  |
| --- | --- |
|  | **public void FM\_SetBeaconInfo(String strUUID, String strMajor, String strMinor, String strRssi, String strDistance)** |
| Parameter & Return | strUUID : IBeacon UUID  strMajor : IBeacon Major Value  strMinor : IBeacon Minor Value  strRssi : IBeacon Rssi Value  strDistance : IBeacon Distance Value |
| 설명 | 1초 간격으로 IBeacon 값을 설정한다. |
| 예재 | m\_FMInterface.FM\_SetBeaconInfo(…..) |

|  |  |
| --- | --- |
|  | **public void FM\_UpdateLGDLayer ()** |
| Parameter & Return |  |
| 설명 | LGD 일 경우 map fragment에 포함 된 activity에서mapReady()에서 호출해준다. |
| 예재 | @Override  public void mapReady() {  m\_FMInterface.FM\_UpdateLGDLayer();  } |

|  |  |
| --- | --- |
|  | **public void FM\_StartSimulation ()** |
| Parameter & Return |  |
| 설명 | 경로 탐색 후 호출 시 모의 주행 한다. |
| 예재 |  |

|  |  |
| --- | --- |
|  | **public void FM\_StopSimulation ()** |
| Parameter & Return |  |
| 설명 | 모의 주행을 종료하고 주행 모드로 돌아 갈 때 호출 한다 |
| 예재 |  |

|  |  |
| --- | --- |
|  | **public void FM\_GetMapCenterPos(double[] lonlat)** |
| Parameter & Return | 지도 중심 좌표를 전달 받을 double array |
| 설명 | 지도 중심의 WGS84 좌표를 얻는다. |
| 예재 | double[] latlon = new double[2];  m\_FMInterface.FM\_GetMapCenterPos(latlon); |

|  |  |
| --- | --- |
|  | **public void FM\_GetSubMapCenterPos(double[] lonlat)** |
| Parameter & Return | 지도 중심 좌표를 전달 받을 double array |
| 설명 | 서브 맵 WGS84 좌표를 얻는다. |
| 예재 | double[] latlon = new double[2];  m\_FMInterface.FM\_GetSubMapCenterPos(latlon); |

|  |  |
| --- | --- |
|  | **public void FM\_FireWaterPoint(Boolean bDisplayFireWater)** |
| Parameter & Return | #Param  bDisplayFireWater : 소방용수 layer를 그릴지에 대한 여부 |
| 설명 | MainMapView에 소방용수 Layer 표출 여부를 설정한다. |
| 예재 | m\_FMInterface.FM\_FireWaterPoint(true); |

|  |  |
| --- | --- |
|  | **public void FM\_SetMapCenter(float hCenter, float vCenter)** |
| Parameter & Return | #Param  hCenter : 지도 영역의 가로 위치  vCenter : 지도 영역의 세로 위치 |
| 설명 | 왼쪽 하단 0.f, 0.f 오른쪽 상단 : 1.f, 1.f 기준 중심을 설정한다.  ex) 0.5f, 0.5f 는 지도 화면 중심 위치 |
| 예재 | Float x = 0.5f;  Float y = 0.5f  m\_FMInterface.FM\_SetMapCenter(x,y); |

|  |  |
| --- | --- |
|  | **public void FM\_AutoSetMapCenter(boolean bMapMove)** |
| Parameter & Return | #Param  bMapMove : 현재 맵 이동 상태인지에 대한 여부 |
| 설명 | bMapMove 값에 따라 지도 중심을 주행모드와 지도 이동 모드로 구분해 이동 시킨다. |
| 예재 | m\_FMInterface.FM\_SetRouteDriveListener(new ANaviApplication.MapStatusListener() {  @Override  public void onRouteFinish() {  }  @Override  public void onMapMove(boolean bMapMove) {  m\_FMInterface.FM\_AutoSetMapCenter(bMapMove);  }  }); |

|  |  |
| --- | --- |
|  | **public void FM\_SetMapPosition(FMBaseActivity.OnFatosMapListener onFatosMapListener, double xlon, double ylat)** |
| Parameter & Return | #Param  onFatosMapListener : Mapview와 연결된 listener  xlon, ylat : 이동 하려는 위치의 WGS84 좌표 |
| 설명 | bMapMove 값에 따라 지도 중심을 주행모드와 지도 이동 모드로 구분해 이동 시킨다. |
| 예재 | double dLonX = 126.767723;  double dLatY = 37.809287; m\_FMInterface.FM\_SetMapPosition(FMBaseActivity.onFatosMapListener,dLonX,dLatY); |

|  |  |
| --- | --- |
|  | **public void FM\_GetTbtImageRes(int nTbtCode)** |
| Parameter & Return | #Param  nTbtCode : summary 정보에 있는 Tbtcode |
| 설명 | TBTCode를 받아서 해당 이미지 리소스 ID를 return |
| 예재 | /\*\*  m\_FristTbtImg : ImageView  nTbtCode : summary 정보에 있는 Tbtcode  \*/  m\_FirstTbtImg.setImageResource(m\_FMInterface. FM\_GetTbtImageRes(nTbtCode)); |

|  |  |
| --- | --- |
|  | **public void FM\_SetNaviVolume(int volume)** |
| Parameter & Return | #Param  volume : 볼륨 크기(0~100) – 0은 mute로 본다. |
| 설명 | 기본 음성의 볼륨을 조절 한다. |
| 예재 | m\_FMInterface.FM\_SetNaviVolume(100) |

|  |  |
| --- | --- |
|  | **public void FM\_IsPlaying()** |
| Parameter & Return | #return  true : 기본 음성이 발성 중 일 때  false : 기본 음성이 발성 중이 아닌 상태 |
| 설명 | 현재 기본 음성이 발생 중인지 체크 한다. |
| 예재 | If(m\_FMInterface.FM\_IsPlaying())  {  // 현재 음성 발성 중  } |

|  |  |
| --- | --- |
|  | **public void FM\_GetMapVersion(NaviVersion versionInfo)** |
| Parameter & Return | #Param  NaviVersion 구조체 정보 참고 |
| 설명 | 현재 배경, 검색, 경로탐색, 기타(SDI,EV..)의 Map 이름/Version, 업데이트 가능 여부, 업데이트 상태 등의 정보를 얻어 온다. |
| 예재 | NaviVersion versionInfo = new NaviVersion();  m\_FSInterface.FMGetMapVersion(versionInfo); |

|  |
| --- |
| **public class NaviVersion** |
| Map Data의 정보를 담고 있는 구조체 |
| int nMapDate;   * 배경 DB 빌드날짜   DBVersion[] listMap;   * 배경 DB 정보 리스트   int nSearchDate;   * 검색 DB 빌드날짜   DBVersion[] listSearch;   * 검색 DB 정보 리스트   int nNetworkDate;   * 경로탐색 DB 빌드날짜   DBVersion[] listNetwork;   * 경로탐색 DB 정보 리스트   DBVersion[] listEtc;   * 기타 DB 정보 리스트 (SDI, EV...) |

|  |
| --- |
| **public class DBVersion** |
| 배경, 검색, 경로탐색, 기타(SDI,EV..) 각각의 DB 정보를 담고 있는 구조체 |
| String strName;   * DB 이름   int nCurDate;   * 현재 DB 빌드날짜   int nUpdateDate;   * 업데이트 가능한 DB 빌드날짜   double fVersion;   * 버전   boolean bUpdate;   * 업데이트 가능 여부   int nUpdateStatus;   * 0:다운로드안됨, 1:다운로드중, 2:다운로드완료 |

|  |  |
| --- | --- |
|  | **public void FM\_RequestUpdate(int nDownloadGubun, long freeSpace, final MapUpdateListenerCallback mapUpdateListener)** |
| Parameter & Return | #Param  nDownloadGubun : 배경, 검색, 경로탐색, 기타(SDI,EV..)의 구분  # Gubun값 정의  public static final int MAP\_DOWNLOAD\_BASEMAP = 1;  public static final int MAP\_DOWNLOAD\_SEARCH = 2;  public static final int MAP\_DOWNLOAD\_RP = 3;  public static final int MAP\_DOWNLOAD\_ANDO = 4;  freeSpace : 현재 Map DB가 있는 storage의 남은 용량  mapUpdateListener : update 상태 정보를 받기 위한 Listener |
| 설명 | **DBVersion** 기준 정보의 업데이트 가능 여부 (bUpdate) 값 기준으로 각 구분 별 Map Download를 요청한다. |
| 예재 | m\_FSInterface.FS\_RequestUpdate(nDownloadGubun, (AMapUtil.getAvailableInternalMemorySize()/(1024\*1204)), new UpdateManager.MapUpdateListenerCallback(){  @Override  public void onData(UpdateManager.mapDownloadResult resultMapDownload) {  }  @Override  public void onError(int nErrorCode) {  }  } |

|  |
| --- |
| **public interface MapUpdateListenerCallback** |
| 배경, 검색, 경로탐색, 기타(SDI,EV..) 각각의 DB 정보를 담고 있는 구조체 |
| void onData(mapDownloadResult resultMapDownload);  void onError(int nErrorCode); |

|  |  |
| --- | --- |
|  | **public void FM\_MapUpdateCancel()** |
| Parameter & Return | #Param |
| 설명 | 맵 업데이트 중 취소 할 때 호출한다. |
| 예재 | m\_FSInterface.MapUpdateCancel() |