API 명세서

ForwardMax B2B 중고차 수출 플랫폼

문서 버전: 2.0
작성일: 2025-12-29
프로젝트: ForwardMax (carivdealer)
작성자: 기획/개발팀 류지환 작성
대상 독자: 내부 이해관계자

목차

---

1. 개요

1.1 API 기본 정보

- **기본 URL**: `https://asia-northeast3-carivdealer.cloudfunctions.net`
- **프로토콜**: HTTPS
- **인코딩**: UTF-8

- **데이터 형식**: JSON

1.2 인증 방식

현재 상태: 인증 미구현 (프로토타입 단계)

계획: Firebase Auth 토큰 기반 인증

```
Authorization: Bearer {firebase_auth_token}
```

1.3 공통 규칙

요청 헤더

```
Content-Type: application/json
```

응답 형식

- **성공**: HTTP 200 + JSON 본문
- **에러**: HTTP 4xx/5xx + JSON 에러 메시지

CORS

- 모든 엔드포인트에서 CORS 허용 (Firebase Functions v2 설정)

리전

- 모든 Functions 는 `asia-northeast3` 리전에 배포됨

---

2. 엔드포인트 목록

2.1 구현된 엔드포인트

API 식별자	엔드포인트명	HTTP 메서드	경로	상태
API-0002	사업자 인증 API	POST	`/verifyBusinessAPI`	✅ 구현됨
API-0100	등록원부 OCR API	POST	`/ocrRegistrationAPI`	✅ 구현됨
API-0101	검차 신청 API	POST	`/inspectionRequestAPI`	✅ 구현됨

API 식별자	엔드포인트명	HTTP 메서드	경로	상태
API-0200	경매 입찰 API	POST	`/bidAPI`	✅ 구현됨
API-0201	즉시구매 API	POST	`/buyNowAPI`	✅ 구현됨
API-0300	판매 방식 변경 API	POST	`/changeSaleMethodAPI`	✅ 구현됨

2.2 Mock API (프로토타입)

API 식별자	엔드포인트명	HTTP 메서드	경로	상태
-	제안 수락/거절	POST	Mock (프론트엔드)	🔶 Mock
-	구매 의사 확인	POST	Mock (프론트엔드)	🔶 Mock
-	탁송 일정 조율	POST	Mock (프론트엔드)	🔶 Mock
-	배차 요청	POST	Mock (프론트엔드)	🔶 Mock
-	배차 확정	POST	Mock (프론트엔드)	🔶 Mock
-	인계 승인	POST	Mock (프론트엔드)	🔶 Mock
-	정산 완료 알림	POST	Mock (프론트엔드)	🔶 Mock

---

3. 엔드포인트 상세 명세

3.1 API-0002: 사업자 인증

엔드포인트: `POST /verifyBusinessAPI`

기능 설명: 사업자등록증 이미지를 업로드하여 OCR 처리 및 진위 확인을 수행합니다.

요청 헤더:

```
Content-Type: multipart/form-data
```

요청 본문 (FormData): - `business_registration_image`: 이미지 파일

응답 본문 (200 OK): - `success`: 성공 여부 (불린) - `verified`: 인증 완료 여부 (불린) - `business_info`: 사업자 정보 객체 - `companyName`: 회사명 (문자열) - `businessRegNo`: 사업자등록번호 (문자열) - `representativeName`: 대표자명 (문자열) - `message`: 메시지 (문자열)

에러 응답: - `error`: 에러 메시지 (문자열)

에러 코드: - `400`: 파일이 제공되지 않음 - `405`: POST 메서드가 아님 - `500`: 서버 오류

구현 상태: Mock 응답 반환 (프로토타입 단계)

상세 요청/응답 스키마는 부록 6.2 를 참조하시기 바랍니다 [1].

3.2 API-0100: 등록원부 OCR

엔드포인트: `POST /ocrRegistrationAPI`

기능 설명: 차량번호를 입력받아 등록원부에서 차량 기본정보를 OCR 로 추출합니다.

요청 헤더:

```
Content-Type: application/json
```

요청 본문: - `car_no`: 차량번호 (문자열)

응답 본문 (200 OK): - `vin`: 차대번호(VIN) (문자열) - `manufacturer`: 제조사 (문자열) - `model`: 모델명 (문자열) - `year`: 연식 (문자열) - `mileage`: 주행거리 (문자열)

에러 응답: - `error`: 에러 메시지 (문자열)

에러 코드: - `400`: 차량번호가 제공되지 않음 - `405`: POST 메서드가 아님 - `500`: 서버 오류

구현 상태: Mock 응답 반환 (프로토타입 단계)

상세 요청/응답 스키마는 부록 6.2 를 참조하시기 바랍니다 [2].

---

[1] 부록 6.2: 요청/응답 스키마 상세 - 사업자 인증

3.3 API-0101: 검차 신청

엔드포인트: `POST /inspectionRequestAPI`

기능 설명: 차량에 대한 검차 신청을 처리하고 Firestore 에 검차 데이터를 저장합니다.

요청 헤더:

```
Content-Type: application/json
```

요청 본문: - `vehicle_id`: 차량 식별자 (문자열) - `preferred_date`: 희망 날짜 (문자열, YYYY-MM-DD 형식) - `preferred_time`: 희망 시간 (문자열, HH:mm 형식)

응답 본문 (200 OK): - `success`: 성공 여부 (불린) - `inspection_id`: 검차 식별자 (문자열) - `message`: 메시지 (문자열)

에러 응답: - `error`: 에러 메시지 (문자열)

에러 코드: - `400`: 필수 파라미터 누락 - `405`: POST 메서드가 아님 - `500`: 서버 오류

구현 상태: ✅ Firebase Functions 구현됨

데이터 저장: Firestore 의 검차 컬렉션에 저장 - 컬렉션: `inspections` - 필드: 차량 식별자, 희망 날짜, 희망 시간, 상태(대기 중), 생성 일시

상세 요청/응답 스키마는 부록 6.2 를 참조하시기 바랍니다 [3].

3.4 API-0200: 경매 입찰

엔드포인트: `POST /bidAPI`

---

[2] 부록 6.2: 요청/응답 스키마 상세 - 등록원부 OCR

[3] 부록 6.2: 요청/응답 스키마 상세 - 검차 신청

기능 설명: 경매에 입찰을 처리합니다. 동시성 제어를 위해 Firestore 트랜잭션을 사용합니다.

요청 헤더:

`Content-Type: application/json`

요청 본문: - `auction_id`: 경매 식별자 (문자열) - `bid_amount`: 입찰 금액 (숫자)

응답 본문 (200 OK): - `success`: 성공 여부 (불린) - `message`: 메시지 (문자열)

에러 응답: - `error`: 에러 메시지 (문자열)

에러 코드: - `400`: 경매 식별자 또는 입찰 금액 누락 - `400`: 입찰 금액이 현재 최고가보다 낮음 - `404`: 경매를 찾을 수 없음 - `400`: 경매가 활성 상태가 아님 - `405`: POST 메서드가 아님 - `500`: 서버 오류

구현 상태: ✅ Firebase Functions 구현됨

동시성 제어: Firestore 트랜잭션 사용

데이터 업데이트: Firestore 의 경매 컬렉션 업데이트 - 컬렉션: `auctions/{auctionId}` - 필드: 현재 최고 입찰가, 업데이트 일시 - 참고: 최고가 업데이트는 화면에 비노출 (Blind Auction)

상세 요청/응답 스키마는 부록 6.2 를 참조하시기 바랍니다 [4].

3.5 API-0201: 즉시구매

엔드포인트: `POST /buyNowAPI`

기능 설명: 경매에서 즉시구매를 처리합니다. 원자성을 보장하기 위해 Firestore 트랜잭션을 사용합니다.

---

[4] 부록 6.2: 요청/응답 스키마 상세 - 경매 입찰

요청 헤더:

`Content-Type: application/json`

요청 본문: - `auction_id`: 경매 식별자 (문자열)

응답 본문 (200 OK): - `success`: 성공 여부 (불린) - `contract_id`: 계약 식별자 (문자열) - `message`: 메시지 (문자열)

에러 응답: - `error`: 에러 메시지 (문자열)

에러 코드:

- `400`: 경매 식별자 누락

- `404`: 경매를 찾을 수 없음

- `400`: 경매가 활성 상태가 아님

- `400`: 즉시구매가 설정되지 않음

- `405`: POST 메서드가 아님

- `500`: 서버 오류

구현 상태: Firebase Functions 구현됨

보장: Firestore 트랜잭션 사용

데이터 업데이트: - Firestore 의 경매 컬렉션 업데이트 - 컬렉션: `auctions/{auctionId}` - 필드: 상태(판매 완료), 현재 최고 입찰가, 종료 일시 - Firestore 의 차량 컬렉션 업데이트 - 컬렉션: `vehicles/{vehicleId}` - 필드: 상태(잠금), 업데이트 일시

상세 요청/응답 스키마는 부록 6.2 를 참조하시기 바랍니다 [5].

---

[5] 부록 6.2: 요청/응답 스키마 상세 - 즉시구매

3.6 API-0300: 판매 방식 변경

엔드포인트: `POST /changeSaleMethodAPI`

기능 설명: 일반 판매에서 경매로 판매 방식을 변경하고 경매를 생성합니다.

요청 헤더:

```
Content-Type: application/json
```

요청 본문: - `vehicle_id`: 차량 식별자 (문자열) - `auction_settings`: 경매 설정 객체 - `start_price`: 시작가 (숫자, 필수) - `buy_now_price`: 즉시구매가 (숫자, 선택)

응답 본문 (200 OK): - `success`: 성공 여부 (불린) - `auction_id`: 경매 식별자 (문자열)

에러 응답: - `error`: 에러 메시지 (문자열)

에러 코드: - `400`: 차량 식별자 또는 경매 설정 누락 - `400`: 시작가 누락 - `404`: 차량을 찾을 수 없음 - `405`: POST 메서드가 아님 - `500`: 서버 오류

구현 상태: Firebase Functions 구현됨

데이터 저장/업데이트:

- Firestore 의 경매 컬렉션에 새 문서 생성

- 컬렉션: `auctions`

- 필드: 차량 식별자, 시작가, 즉시구매가, 현재 최고 입찰가(null), 상태(활성), 생성 일시

- Firestore 의 차량 컬렉션 업데이트

- 컬렉션: `vehicles/{vehicleId}` - 필드: 상태(입찰 중), 경매 식별자, 업데이트 일시

상세 요청/응답 스키마는 부록 6.2 를 참조하시기 바랍니다 [6].

---

[6] 부록 6.2: 요청/응답 스키마 상세 - 판매 방식 변경

4. Mock API

프로토타입 단계에서 프론트엔드에서 Mock 응답을 반환하는 API 들입니다. 향후 Firebase Functions 로 구현 예정입니다.

4.1 일반 판매 제안 수락/거절

기능 설명: 일반 판매 제안을 수락하거나 거절합니다.

요청 데이터: - 제안 식별자 (문자열) - 동작 (수락/거절) (문자열)

응답 데이터: - 성공 여부 (불린) - 메시지 (문자열)

향후 엔드포인트: `POST /acceptProposalAPI`

구현 코드는 부록 6.3 을 참조하시기 바랍니다 [7].

4.2 바이어 최종 구매 의사 재확인

기능 설명: 바이어의 최종 구매 의사를 재확인합니다.

요청 데이터: - 제안 식별자 (문자열) - 확인 여부 (불린)

응답 데이터: - 성공 여부 (불린) - 메시지 (문자열)

향후 엔드포인트: `POST /confirmProposalAPI`

구현 코드는 부록 6.3 을 참조하시기 바랍니다 [8].

---

[7] 부록 6.3: 구현 코드 참조 - 제안 수락/거절

[8] 부록 6.3: 구현 코드 참조 - 구매 의사 확인

4.3 탁송 일정 조율

기능 설명: 탁송 일정을 조율합니다.

요청 데이터: - 일정 날짜 (문자열) - 일정 시간 (문자열) - 주소 (문자열)

응답 데이터: - 성공 여부 (불린) - 일정 식별자 (문자열)

향후 엔드포인트: `POST /scheduleLogisticsAPI`

구현 코드는 부록 6.3 을 참조하시기 바랍니다 [9].

4.4 배차 요청

기능 설명: 배차를 요청합니다.

요청 데이터: - 일정 식별자 (문자열)

응답 데이터: - 성공 여부 (불린) - 배차 식별자 (문자열)

향후 엔드포인트: `POST /dispatchLogisticsAPI`

구현 코드는 부록 6.3 을 참조하시기 바랍니다 [10].

4.5 배차 확정

기능 설명: 배차를 확정합니다.

요청 데이터: - 배차 식별자 (문자열)

---

[9] 부록 6.3: 구현 코드 참조 - 탁송 일정 조율

[10] 부록 6.3: 구현 코드 참조 - 배차 요청

응답 데이터: - 성공 여부 (불린) - 기사 정보 객체 - name: 기사명 (문자열) - phone: 전화번호 (문자열)

향후 엔드포인트: `POST /confirmDispatchAPI`

구현 코드는 부록 6.3 을 참조하시기 바랍니다 [11].

4.6 인계 승인

기능 설명: 탁송 기사로부터 차량 인계를 승인합니다.

요청 데이터: - 탁송 식별자 (문자열) - PIN (6 자리) (문자열)

응답 데이터: - 성공 여부 (불린) - 인계 시각 (ISO 8601 형식 문자열)

향후 엔드포인트: `POST /approveHandoverAPI`

보안 참고: PIN 번호는 로그에서 마스킹 처리됨

구현 코드는 부록 6.3 을 참조하시기 바랍니다 [12].

4.7 정산 완료 알림

기능 설명: 정산 완료 알림을 전송합니다.

요청 데이터: - 정산 식별자 (문자열)

응답 데이터: - 성공 여부 (불린) - 알림 식별자 (문자열)

향후 엔드포인트: `POST /notifySettlementAPI`

구현 코드는 부록 6.3 을 참조하시기 바랍니다 [13].

---

[11] 부록 6.3: 구현 코드 참조 - 배차 확정

[12] 부록 6.3: 구현 코드 참조 - 인계 승인

5. 에러 처리

5.1 공통 에러 코드

HTTP 상태 코드	의미	설명
200	OK	요청 성공
400	Bad Request	잘못된 요청 (필수 파라미터 누락, 형식 오류 등)
404	Not Found	리소스를 찾을 수 없음
405	Method Not Allowed	허용되지 않은 HTTP 메서드
500	Internal Server Error	서버 내부 오류

5.2 에러 응답 형식

표준 에러 응답:

```
{
  "error": "에러 메시지"
}
```

예제:

```
{
  "error": "vehicle_id, preferred_date, and preferred_time are required"
}
```

5.3 에러 처리 가이드

프론트엔드 에러 처리: - HTTP 응답 상태 코드 확인 - JSON 에러 메시지 파싱 - 사용자에게 적절한 에러 메시지 표시

백엔드 에러 처리: - try-catch 블록을 통한 예외 처리 - 적절한 HTTP 상태 코드 반환 - 에러 로깅

---

[13] 부록 6.3: 구현 코드 참조 - 정산 완료 알림

상세 에러 처리 코드는 부록 6.3 을 참조하시기 바랍니다 [14].

---

6. 부록

6.1 용어집

용어	설명
엔드포인트	API 서비스의 특정 기능에 접근하기 위한 URL 경로
요청 본문	API 호출 시 전송하는 데이터
응답 본문	API 호출 결과로 받는 데이터
Mock API	실제 구현 전 테스트를 위한 가짜 API
트랜잭션	데이터베이스 작업의 원자성을 보장하는 메커니즘
CORS	Cross-Origin Resource Sharing, 다른 도메인 간 리소스 공유를 허용하는 메커니즘
PIN	Personal Identification Number, 개인 식별 번호

6.2 요청/응답 스키마 상세

사업자 인증 (API-0002)

요청 스키마: - FormData 형식 - `business_registration_image`: File 객체

응답 스키마:

```
{
  "success": true,
  "verified": true,
  "business_info": {
```

---

[14] 부록 6.3: 구현 코드 참조 - 에러 처리

```
    "companyName": "Global Motors",
    "businessRegNo": "123-45-67890",
    "representativeName": "홍길동"
  },
  "message": "인증이 완료되었습니다."
}
```

등록원부 OCR (API-0100)

요청 스키마:

```
{
  "car_no": "82 가 1923"
}
```

응답 스키마:

```
{
  "vin": "KMHXX00XXXX000000",
  "manufacturer": "Hyundai",
  "model": "Porter II",
  "year": "2018",
  "mileage": "136000"
}
```

검차 신청 (API-0101)

요청 스키마:

```
{
  "vehicle_id": "v-101",
  "preferred_date": "2025-01-25",
  "preferred_time": "14:00"
}
```

응답 스키마:

```
{
  "success": true,
  "inspection_id": "insp-1234567890",
  "message": "검차 신청이 완료되었습니다."
}
```

경매 입찰 (API-0200)

요청 스키마:

```
{
  "auction_id": "auc-1234567890",
  "bid_amount": 650000
}
```

응답 스키마:

```
{
  "success": true,
  "message": "입찰이 완료되었습니다."
}
```

즉시구매 (API-0201)

요청 스키마:

```
{
  "auction_id": "auc-1234567890"
}
```

응답 스키마:

```
{
  "success": true,
  "contract_id": "contract-1234567890",
  "message": "즉시구매가 완료되었습니다."
}
```

판매 방식 변경 (API-0300)

요청 스키마:

```
{
  "vehicle_id": "v-101",
  "auction_settings": {
    "start_price": 500000,
    "buy_now_price": 700000
  }
}
```

응답 스키마:

```
{
  "success": true,
  "auction_id": "auc-1234567890"
}
```

6.3 구현 코드 참조

프론트엔드 API 클라이언트: `src/services/api.ts`

백엔드 Functions: - `functions/src/index.ts` (엔드포인트 등록) - `functions/src/member/verifyBusiness.ts` (사업자 인증) - `functions/src/vehicle/ocrRegistration.ts` (등록원부 OCR) - `functions/src/vehicle/inspection.ts` (검차 신청) - `functions/src/auction/bid.ts` (경매 입찰) - `functions/src/auction/buyNow.ts` (즉시구매) - `functions/src/trade/changeSaleMethod.ts` (판매 방식 변경)

설정 파일: - `firebase.json` (Firebase 설정)

2025 12 29 류지환 작성

문서 끝.