오너클랜 API

Version

`v0.11.0`

Document version

`v0.11.0.0`

Changes

`v0.10.17` -> `v0.11.0`

- 반품/교환 신청 기능 추가.
  - 자세한 사항은 *Methods* 섹션의 `requestRefundOrExchange` 쿼리 설명 참고.

`v0.10.13` -> `v0.10.17`

- 주문 생성시 여러 상품을 주문하는 경우, 다음 기준에 의해 필요한 경우, 자동으로 여러 주문으로 나뉘어 생성되도록 변경.
  - 상품 공급사가 다른 경우(`Item.metadata.vendorKey`)
  - 상품 배송비 부과 타입이 다른 경우(`Item.shippingType`)
- JWT 인증 토큰 발급 서버 URL 변경
  - 기존:
    - sandbox: API Authentication endpoint(Sandbox)(Deprecated)
    - production: API Authentication endpoint(Production)(Deprecated)
  - 변경:
    - sandbox: API Authentication endpoint(Sandbox)
    - production: API Authentication endpoint(Production)
  - 기존에 사용하던 서버는 일정 버전 이후 접근 불가능할 예정이므로 가급적 신규 엔드포인트 사용.

`v0.10.11` -> `v0.10.13`

- 해외배송 상품 주문 시 개인통관 고유번호/생년월일 입력 추가 및 형식 변경
  - 기존에 `string` 타입이었으나 `CustomsClearanceCodeInput` 타입으로 변경되었으며 필요없는 경우 아예 입력하지 않아야 함.
  - `createOrder` 항목 참조.
- 송장번호 입력일 기준으로 주문 검색하는 기능 추가.
  - `allOrders` 파라미터에 `shippedAfter`, `shippedBefore` 추가.
  - `shippedAfter`: 해당 시점 이후에 송장번호가 입력된 주문건만 검색.
  - `shippedBefore`: 해당 시점 이전에 송장번호가 입력된 주문건만 검색.

`v0.10.10` -> `v0.10.11`

- `key`의 배열로 상품 리스트를 조회하는 기능 추가.
  - `itemsByKeys` 항목 참조.

- 상품 등록(`createItem`) 및 수정(`updateItem`)시에 옵션명 및 옵션값 글자 수 제한 적용.
  - 옵션명(`Item.options.optionAttributes.name`): `euc-kr` 인코딩 기준 40bytes 이하.
  - 옵션값(`Item.options.optionAttributes.value`): `euc-kr` 인코딩 기준 20bytes 이하.

`v0.9.1` -> `v0.10.10`

- C/S WRITE API 추가.
  - 추가 내역:
    - `createSellerQnaArticle` - 1:1 문의 작성.
  - 자세한 사용 방법은 API method 항목 참조.

`v0.9.0` -> `v0.9.1`

- 다중 목록이미지 지원.
  - `Item.images`에 제공되는 링크 중 첫 번째 링크가 대표 목록이미지, 나머지가 추가 목록이미지.

`v0.8.0` -> `v0.9.0`

- C/S READ API 추가.
  - 추가 내역:
    - `sellerQnaArticle` / `allSellerQnaArticles` - 1:1 문의 게시판 글 조회.(단일 / 복수)
    - `emergencyMessage` / `allEmergencyMessages` - 긴급메시지 조회.(단일 / 복수)
    - `notice` / `allNotices` - 알림메모 조회.(단일 / 복수)
  - 자세한 사용 방법은 각 API method 항목 참조.

`v0.7.70` -> `v0.8.0`

- Order Update API 추가.
  - 추가 내역:
    - `cancelOrder` - 주문 취소.
    - `requestOrderCancellation` - 주문 취소 요청.
  - 자세한 사용 방법은 각 API method 항목 참조.
- GUI endpoint(GraphiQL -> GraphQL Playground) 변경
  - 기존에는 별도의 URL이 있었으나, v0.8부터는 GraphQL API Endpoint와 동일한 URL을 사용하고 브라우저로 접속시 GUI 사용 가능.
- `allItems` 쿼리의 `sortBy` 기준 추가.
  - 기존의 `keyAsc`, `keyDesc` 동작 버그 개선 및 상품코드 기준 정렬에서 내부 자체 기준 정렬로 변경.
  - 변경된 `keyAsc`, `keyDesc`는 반환하는 Item의 순서를 항상 보장함.
  - 상품코드(Item.key)로 정렬하는 `itemKeyAsc`, `itemKeyDesc` 추가. 변경 이전의 `keyAsc`, `keyDesc`와 동일함.
- 기타 버그 개선.

`v0.7.69` -> `v0.7.70`

- Order UPDATE API 추가.
  - 주문관리메모, 원청주문코드 수정 가능.(`updateOrderNotes`)
  - 사용법은 매뉴얼 참조.

`v0.7.65` -> `v0.7.69`

- Order CREATE API(`createOrder`) 추가.
  - 사용법은 매뉴얼 참조.
  - 예상 주문 금액 확인을 위한 `simulateCreateOrder` API 제공.

`v0.7.56` -> `v0.7.65`

- 외부 마켓 카테고리 매칭 정보 관련 버그 수정.
- Order READ API(order, allOrders) 추가.
  - `order`로 단일 주문내역, `allOrders`로 복수 주문내역 조회 가능.
  - 사용법은 매뉴얼 참조.

`v0.7.52` -> `v0.7.56`

- 판매자 한정 설명(`Item.sellerOnlyContent`) 조회 불가로 변경
  - 오너클랜 웹사이트 변경에 맞추어서 삭제.
  - 앞으로 사용 예정 없음.
- `allItems` 쿼리 정렬 조건(`sortBy`) 관련 변경
  - `default`는 기존에 오너클랜 웹사이트 내 검색 결과와 동일한 순서를 보장하기 위한 정렬 기준이었으나, 해당 정렬이 DB 부하를 많이 발생시켜서 일반적인 조건에서는 `registerDateDesc`와 동일하게 변경. 다만 `search` 파라미터를 같이 사용하는 경우에는 기존과 동일하게 동작.
- `allItems`의 일부 쿼리 형태에 대해 Quota 제약 강화
  - `sortBy`에서 `priceAsc`, `priceDesc`를 사용하는 경우
  - `search` 파라미터를 사용하는 경우
- 기타 버그 수정.

`v0.7.45` -> `v0.7.52`

- 카테고리에 전체 이름 추가.
  - 기존에는 카테고리의 현재 레벨에서의 이름(ex: "청소기")만을 `Category.name` 필드로 반환했으나, `Category.fullName` 필드에서 전체 이름(ex: "디지털/가전>생활가전>청소기")를 반환하도록 추가.
- 오픈마켓 연동 카테고리 정보에 각 카테고리에 대한 전체 이름 추가. 정보가 없는 경우 `null` 또는 빈 문자열 반환.
  - `Item.metadata.coupangCategoryFullName`
  - `Item.metadata.auctionCategoryFullName`
  - `Item.metadata.gmarketCategoryFullName`
  - `Item.metadata.st11CategoryFullName`
  - `Item.metadata.interparkCategoryFullName`
  - `Item.metadata.smartstoreCategoryFullName`
  - `Item.metadata.playautoCategoryFullName`
  - `Item.metadata.esellersCategoryFullName`

`v0.7.42` -> `v0.7.45`

- 외부 카테고리 정보 제공시 플레이오토, 이셀러스 카테고리 코드 추가 제공.
- `Item.options.status`에서 `null`로 표시되는 경우 개선.

`v0.7.40` -> `v0.7.42`

- 오너클랜 카테고리 체계 개편에 따른 카테고리 코드 변경
  - 카테고리 코드가 쓰이는 모든 필드 및 검색 조건은 신규 카테고리 정보로 변경.

`v0.7.39` -> `v0.7.40`

- `allItems` 쿼리에 `openmarketSellable` 필터 조건 추가 및 오픈마켓에서 판매 가능한 상품의 정보만 불러오도록 변경.
  - `openmarketSellable` 필터는 boolean 필터로, `true`, `false`의 두 가지 값만을 가짐. `true`이거나 생략하면 오픈마켓에서 판매 가능한 상품만 불러오며(Item.openmarketSellable의 값이 true), `false`이면 오픈마켓에서 판매 불가능한 상품만 불러옴.(Item.openmarketSellable의 값이 false)
  - `openmarketSellable`의 값이 `false`인 상품을 오픈마켓 등에 등록하는 경우 오픈마켓에서 제재를 받는 등의 불이익이 발생할 수 있음.
  - 단, `item` 쿼리를 이용한 단일 상품 조회 쿼리에는 위 필터가 없으므로 **반드시 openmarketSellable 필드를 참고하여 상품 정보를 사용할 것**을 권장.

`v0.7.35` -> `v0.7.39`

- `Item`의 `Category` 정보를 불러올 때 `id`, `name`, `registrable`, `attributes` 필드를 제대로 불러오지 못하는 버그 수정.
- `allItems` 쿼리의 검색어 검색(`search`) 버그 수정 및 오너클랜 검색 로직과 동기화.
- 상품 조회시 배송비 부과 타입의 선착불(`all`) 삭제.

`v0.7.28` -> `v0.7.35`

- 반품 접수 타입에서 1:1 문의 타입(`board`) 삭제.

`v0.7.27` -> `v0.7.28`

- 상품 인증정보(`Item.metadata.certificateInformation`) 타입 추가.
  - 방송통신기자재 적합인증(certificateCode: 19), 적합등록(certificateCode: 20), 잠정인증(certificateCode: 21) 타입 추가
  - 자세한 내용은 *인증정보(certificateInformation)* 섹션 참고.

`v0.7.22` -> `v0.7.27`

- 상품 판매상태 업데이트 가이드라인 추가.
  - API를 통해 조회한 상품 정보를 사용하는 경우에 필요한 상품 판매상태(`available`, `soldout`, `discontinued`, `unavailable`) 업데이트 가이드라인.
  - 위 가이드라인을 통해 적절히 업데이트 하지 않을 경우 오너클랜에서 판매하지 않는 상태의 상품 정보를 사용하게 될 수 있음.
  - 자세한 사항은 *상품 운영시 참조사항 - 상품 판매상태 업데이트 가이드라인* 참고.

`v0.7.18` -> `v0.7.22`

- `itemHistories` 쿼리 추가.
  - 품절/단종/재입고 리스트 조회 기능.
  - 1개 상품 조회 기능과 리스트 전체 조회 기능으로 나뉨.
  - 자세한 사항은 *Methods* 섹션의 `itemHistories` 쿼리 설명 참고.

`v0.7.12` -> `v0.7.18`

- 반품접수 유형 조회 항목 추가
  - `Item.returnCriteria` 필드 추가. 자세한 사항은 *데이터 타입 - ReturnCriteria* 참고.

`v0.7.5` -> `v0.7.12`

- 유통금지 상품 항목 추가
  - `Item.attributes` 필드에 `"PROHIBIT"`이 포함된 경우.

`v0.7.3` -> `v0.7.5`

- `allItems` 필드의 `category` 필드 검색 방식 변경
  - 일치 검색에서 `LIKE` 검색으로 변경. 즉, 해당 카테고리 및 하위 카테고리(모든 레벨)에 있는 모든 상품을 검색.

`v0.7.0` -> `v0.7.3`

- 필드 삭제
  - `Item.brand` -> 삭제
    - 현재 오너클랜 시스템에서 사용하지 않는 필드로, 삭제.

`v0.6.85` -> `v0.7.0`

- 필드명 및 위치 변경
  - `Item.isTaxFree` -> `Item.taxFree`
  - `Item.isMinorSellable` -> `Item.adultOnly`
    - `adultOnly = not isMinorSellable`의 관계에 있습니다.
  - (New) `Item.price`
    - 상품의 기준 가격입니다. 가격이 서로 다른 옵션이 있는 경우 이 가격을 기준으로 추가 혹은 할인 가격을 계산할 수 있습니다.
  - `Item.metadata.madein` -> `Item.origin`
  - `Item.metadata.production` -> `Item.production`
  - `Item.metadata.isStrictSellprice` -> `Item.pricePolicy`
    - Type: `Boolean` -> `Enum { free, fixed }`
    - `isStrictSellprice`가 `true`인 경우는 `fixed`와 같고, `false`인 경우는 `free`와 같습니다.
  - `Item.metadata.prContent` -> `Item.sellerOnlyContent`
  - `Item.metadata.deliveryFee` -> `Item.shippingFee`
  - `Item.metadata.deliveryType` -> `Item.shippingType`
  - `Item.metadata.consumerPrice` -> `Item.fixedPrice`
    - Type: `Int` -> `Price`
  - `Item.metadata.isRefundable` -> `Item.returnable`
  - `Item.metadata.rtRefuseMsg` -> `Item.noReturnReason`
  - `Item.metadata.safeDeliveryDay` -> `Item.guaranteedShippingPeriod`
  - `Item.metadata.isOpenmarketSellable` -> `Item.openmarketSellable`
  - `Item.metadata.minq` -> 삭제
  - `Item.metadata.maxq` -> `Item.boxQuantity`
  - `Item.metadata.vendorAlertMessage` -> `Item.metadata.vendorNotice`
  - `Item.metadata.closetime` -> `Item.closingTime`

    - `Item.metadata.nfCategoryInformation` -> `Item.metadata.productNotificationInformation`
- 필드 액세스 방식 변경
    - `Price`
        - 기존: 화폐 단위 이름의 하위 필드를 명시함으로써 액세스

```
# 입력
{
  priceField {
    KRW
  }
}
```

```
{
  "priceField": {
    "KRW": 12345
  }
}
```

        - 변경: 필드의 인자로 `currency`를 지정함. `currency`는 기존 방식에서 사용하던 화폐 단위 이름과 같은 enum.

```
# 입력
{
  priceField (currency: KRW)
}
```

```
{
  "priceField": 12345
}
```

        - 참고사항: 여러 화폐 단위의 값이 필요한 경우나 여러 번 요청해야 하는 경우 GraphQL Alias를 다음과 같이 사용.

```
# 입력
{
  price_KRW: priceField (currency: KRW)
  price_USD: priceField (currency: USD)
}
```

```
{
  "price_KRW": 12345,
  "price_USD": 12.345
}
```

- `images`
  - 기존: 사이즈 이름의 하위 필드를 명시함으로써 액세스

```
# 입력
{
  images {
    large
    medium
    small
  }
}
```

```
{
  "images": {
    "large": [
      "url1",
      "url2"
    ],
    "medium": [
      "url3",
      "url4"
    ],
    "small": [
      "url5",
      "url6"
    ]
  }
}
```

  - 변경: 필드의 인자로 `size`를 지정함. `size`는 기존 방식에서 사용하던 사이즈 이름과 같은 enum.

```
# 입력
{
  imagesField (size: large)
}
```

```
{
  "imagesField": [
```

```
    "url1",
    "url2"
  ]
}
```

- 참고사항: 여러 사이즈의 이미지가 필요한 경우나 여러 번 요청해야 하는 경우 GraphQL Alias를 다음과 같이 사용.

```
# 입력
{
  images_large: imagesField (size: large)
  images_medium: imagesField (size: medium)
  images_small: imagesField (size: small)
}
```

```
{
  "images_large": [
    "url1",
    "url2"
  ],
  "images_medium": [
    "url3",
    "url4"
  ],
  "images_small": [
    "url5",
    "url6"
  ]
}
```

Endpoints

API Endpoint

- URL:
  - Production: https://api.ownerclan.com/v1/graphql
  - Sandbox: https://api-sandbox.ownerclan.com/v1/graphql
- 사용법: **GraphQL 쿼리 보내기** 섹션 참고.

GUI Test Endpoint (GraphQL Playground)

- URL:
  - Production: https://api.ownerclan.com/v1/graphql
  - Sandbox: https://api-sandbox.ownerclan.com/v1/graphql
- 사용법: **GraphQL Playground** 섹션 참고.

Authentication Endpoint

- URL:
  - Production: https://auth.ownerclan.com/auth
  - Sandbox: https://auth-sandbox.ownerclan.com/auth
- 인증 방식: JWT
- 사용법: **JWT 인증** 섹션 참고.

사용법

GraphQL

오너클랜 API는 GraphQL로 구현되어 있습니다. GraphQL은 URL과 Method에 따라 query가 구분되었던 REST API와 다르게 GraphQL 언어를 통해 쿼리를 작성하고 단 하나의 엔드포인트로 모든 요청을 보내도록 되어있습니다. 쿼리의 내용에 어떤 데이터를 받을 지, 어떤 데이터를 수정할 지 명시하면 됩니다. 예를 들면 다음과 같은 방식으로 상품 1개에 대한 정보를 얻어올 수 있습니다.

```
# 예시 쿼리.
# 그대로 붙여넣기 해도 돌아가지는 않습니다.
query {
    item(key: "W000000") {
        name
        model
        options {
            price
            quantity
            optionAttributes {
                name
                value
            }
        }
    }
}
```

```
# 예시 응답.
{
    "data": {
        "name": "예시 상품",
        "model": "예시 모델",
        "options": [
            {
                "price": 35000,
                "quantity": 23,
                "optionAttributes": [
                    {
                        "name": "색상",
                        "value": "RED"
                    },
                    {
                        "name": "사이즈",
                        "value": "95"
```

```
                    }
                ]
            },
            {
                "price": 35000,
                "quantity": 23,
                "optionAttributes": [
                    {
                        "name": "색상",
                        "value": "RED"
                    },
                    {
                        "name": "사이즈",
                        "value": "100"
                    }
                ]
            }
        ]
    }
}
```

예시 쿼리와 응답을 보면 알 수 있듯이, GraphQL 쿼리는 쿼리에 적힌 그대로 응답을 주게 됩니다.

GraphQL 쿼리 보내기

GraphQL 쿼리는 REST API에서 요청의 종류에 따라(READ, CREATE, ...) HTTP method가 달라졌던 것에 비해 READ 요청은 모두 GET method를, CREATE, UPDATE, DELETE 요청은 모두 POST method를 사용합니다. 아래 **JWT 인증** 섹션에서 설명하는 JWT 토큰을 받아 Header에 `Authorization` header를 추가하면 되며, Bearer token 타입이므로 `Bearer YOUR_TOKEN`과 같은 형식으로 넣어주시면 됩니다.

- READ (query): GET method를 사용합니다. 쿼리 내용은 URL의 query parameter로 넘겨주시면 됩니다.
  - 예시: `https://api-sandbox.ownerclan.com/v1/graphql?query=EXAMPLE_QUERY`
  - 코드:

```
// 아래 코드는 작동하는 코드가 아닙니다.
// 작동하는 코드는 각 메소드 하단에 있는 예제를 참고해주세요.

// ajax를 위한 XMLHttpRequest 객체를 초기화합니다.
var client = new XMLHttpRequest();

// GraphQL Playground에서 사용한 쿼리를 그대로 사용할 수 있습니다.
var readQuery = "test query";

// ajax 요청을 셋팅합니다.
client.open("GET", "https://api-sandbox.ownerclan.com/v1/graphql?query=" +
encodeURIComponent(readQuery), true);
client.setRequestHeader("Authorization", "Bearer YOUR_TOKEN");
client.onreadystatechange = function (aEvt) {
    if (client.readyState === 4) {
        if (client.status === 200) {
            var response = JSON.parse(client.responseText);
```

```
            if (response.errors) {
                // API 서버 응답이 정상이지만 API 에러가 있다면 에러를 콘솔에 씁
니다.
                console.error(JSON.stringify(response.errors));
            } else {
                // API 서버 응답도 정상이고, API 에러도 없다면 반환된 데이터를 콘
솔에 씁니다.
                console.log(JSON.stringify(response.data));
            }
        } else {
            // API 서버 응답이 정상이 아닌 경우 에러와 HTTP status code를 콘솔에
씁니다.
            console.error(client.status, client.responseText);
        }
    }
}

// ajax 요청을 보냅니다.
client.send(null);
```

GraphQL Playground (Formerly GraphiQL)

오너클랜 API에서는 GraphQL Playground에서 UI를 갖추고 쿼리를 테스트할 수 있는 환경을 제공합니다. 어떤 쿼리가 있는지, 쿼리는 어떤 인자를 받고 어떤 데이터를 반환하는지까지 모두 볼 수 있습니다. GraphQL Playground는 Production 환경과 Sandbox 환경에서 모두 제공되며, API Endpoint와 동일한 Endpoint를 브라우저로 접속하면 됩니다. 특히 아래 **Method** 섹션에서 각 method별로 주어진 예시 쿼리를 시도해 볼 수 있습니다. 인증 token은 아래 **JWT 인증** 섹션에서 나온 방법대로 받아서 GraphQL Playground의 *Http headers* 항목에 JSON object 형식으로 지정하면 됩니다. (아래 예시 참조)

```
{
  "Authorization": "Bearer YOUR_TOKEN"
}
```

JWT 인증

오너클랜 API는 JWT를 기반으로 한 토큰 인증 방식을 사용합니다. 토큰은 API Authentication endpoint(Production) 혹은 API Authentication endpoint(Sandbox)로 오너클랜 아이디와 비밀번호를 POST 요청으로 보내서 받을 수 있습니다. 자바스크립트 예시는 다음과 같습니다.

```
var authData = {
    service: "ownerclan",
    userType: "seller",
    username: "판매사ID", // 오너클랜 판매사 ID.
    password: "판매사PW" // 판매사 비밀번호.
};

$.ajax({
    url: "https://auth-sandbox.ownerclan.com/auth", // Production 환경에서는
```

```
"https://auth.ownerclan.com/auth"를 사용합니다.
    type: "POST",
    contentType: "application/json",
    processData: false,
    data: JSON.stringify(authData),
    success: function(data) {
        // 성공한 경우, 토큰을 콘솔에 출력합니다.
        console.log(data);
    },
    error: function(data) {
        // 실패한 경우 에러와 HTTP status code를 기록합니다.
        console.error(data.responseText, data.status);
    }
});
```

Method

item

단일 상품 정보를 조회합니다.

- 파라미터:
  - **key**: String - 정보를 조회할 상품의 key입니다. 오너클랜 상품 코드와 같습니다.
  - lang: Language - Text 타입의 필드들의 lang 파라미터 값을 한 번에 설정합니다. 하위 필드에서 별도로 설정하는 경우 이 설정은 덮어씌워집니다. 사용 가능한 값은 *데이터 타입 - Language* 파트를 참고해주시면 됩니다. 기본값은 ko_KR입니다.
  - currency: Currency - 가격 정보 필드들의 currency 파라미터 값을 한 번에 설정합니다. 하위 필드에서 별도로 설정하는 경우 이 설정은 덮어씌워집니다. 사용 가능한 값은 *데이터 타입 - Currency* 항목을 참고해주시면 됩니다. 기본값은 KRW입니다.
- 반환 데이터: 상품 정보를 반환하며, 다음 내용을 액세스 할 수 있습니다.
  - createdAt: Int - 상품이 DB에 등록된 시각입니다. Unix timestamp로 주어집니다.
  - updatedAt: Int - 상품이 최종적으로 업데이트 된 시각입니다. Unix timestamp로 주어집니다.
  - key: String - 상품의 오너클랜 코드입니다.
  - name: Text - 상품의 이름입니다. lang 파라미터를 가질 수 있으며, 언어를 지정할 수 있습니다. 현재 지원되는 언어는 ko_KR입니다.
  - model: String - 상품의 모델명입니다.
  - production: String - 제조사입니다.
  - origin: String - 제조 국가 (국내 제조의 경우 지역 포함)입니다.
  - id: String - 상품의 API ID입니다.
  - price: Float - 상품의 가격입니다. 옵션마다 가격이 다른 경우 기준 가격이 됩니다. currency 파라미터를 가질 수 있으며, 현재 지원되는 Currency는 KRW입니다.
  - pricePolicy: PricePolicy - 상품 가격 정책입니다. 가능한 값들은 *데이터 타입 - PricePolicy* 항목을 참고해주시면 됩니다.
  - fixedPrice: Float - pricePolicy가 fixed인 경우 소비자 준수가격입니다. currency 파라미터를 가질 수 있으며, 현재 지원되는 Currency는 KRW입니다.
  - searchKeywords: [String] - 상품에 대한 검색 키워드 목록입니다.
  - category: Category - 상품에 대한 카테고리 정보입니다.

- `content`: `String` - 상품의 상세정보입니다.
- `shippingFee`: `Int` - 상품 배송비입니다.
- `shippingType`: `ShippingType` - 상품 배송비 부과 타입입니다. 가능한 값들은 *데이터 타입 - ShippingType* 항목을 참고해주시면 됩니다.
- `images`: `[URL]` - 상품의 이미지를 제공합니다. `size` 파라미터를 가지며, 현재 지원되는 `ImageSize`는 `auto`, `xlarge`, `large`, `medium`, `small` 입니다. 제공되는 URL 중 첫 번째 URL이 대표 목록이미지, 2개 이상의 URL이 제공된 경우 나머지 URL들이 추가 목록이미지입니다.
- `status`: `String` - 상품의 현재 상태입니다. `"soldout"`은 품절, `"available"`은 판매중, `"unavailable"`은 수량이 남아있으나 판매하지 않는(진열 X) 상태, `"discontinued"`은 단종을 의미합니다.
- `options`: `[ItemOption]` - 상품의 옵션 정보입니다. 옵션별 옵션 속성, 가격을 조회할 수 있습니다. `ItemOption`에 대한 내용은 *데이터 타입 - ItemOption* 항목을 참고해주시면 됩니다.
- `taxFree`: `Boolean` - 면세 상품인지의 여부입니다.
- `adultOnly`: `Boolean` - 미성년자 판매 가능 상품인지의 여부입니다. `false`면 가능, `true`면 불가능입니다.
- `returnable`: `Boolean` - 반품 가능 여부입니다.
- `noReturnReason`: `String` - 반품 불가 상품의 경우, 반품 불가 사유입니다.
- `guaranteedShippingPeriod`: `Int` - 안전배송일입니다.
- `openmarketSellable`: `Boolean` - 오픈마켓 판매가 가능한 상품인지의 여부입니다.
- `boxQuantity`: `Int` - 1박스 당 최대 묶음배송 가능 수량입니다.
- `attributes`: `[String]` - 상품 속성의 리스트입니다.
- `closingTime`: `String` - 배송 마감 시각입니다. `null`이나 `undefined`인 경우 따로 지정되지 않은 것이며, 지정된 경우 양식은 `HH:MM:SS`입니다.
- `returnCriteria`: 반품접수 유형입니다. 자세한 내용은 *데이터 타입 - ReturnCriteria* 항목을 참고해주시면 됩니다.
- `metadata`: `JSON` - 상품에 대한 추가 정보입니다.
  - `vendorKey`: `String` - 해당 상품을 공급하는 공급사의 고유 key입니다.
  - `grade`: `String` - 상품 등급입니다.
  - `productNotificationInformation`: `Object` - 상품 정보고시 정보입니다. 하단의 **정보고시** 섹션을 참고해주세요.
  - `certificateInformation`: `[Object]` - 상품 인증정보 내역입니다. 인증정보는 하단의 **인증정보** 섹션을 참고해주세요.
  - `vendorNotice`: `String` - 공급사 알림 메시지입니다. 오너클랜 사이트의 상품 상세페이지에서 상품 상세설명 바로 위에 노출되는, 상단문구와는 별개의 문구입니다.
  - `auctionCategoryCode`: `String` - 이 item의 옥션 카테고리 코드입니다.
  - `auctionCategoryFullName`: `String` - 이 item의 옥션 카테고리 이름입니다.
  - `gmarketCategoryCode`: `String` - 이 item의 지마켓 카테고리 코드입니다.
  - `gmarketCategoryFullName`: `String` - 이 item의 지마켓 카테고리 이름입니다.
  - `st11CategoryCode`: `String` - 이 item의 11번가 카테고리 코드입니다.
  - `st11CategoryFullName`: `String` - 이 item의 11번가 카테고리 이름입니다.
  - `interparkCategoryCode`: `String` - 이 item의 인터파크 카테고리 코드입니다.
  - `interparkCategoryFullName`: `String` - 이 item의 인터파크 카테고리 이름입니다.
  - `smartstoreCategoryCode`: `String` - 이 item의 스마트스토어(스토어팜) 카테고리 코드입니다.
  - `smartstoreCategoryFullName`: `String` - 이 item의 스마트스토어(스토어팜) 카테고리 이름입니다.

        - coupangCategoryCode: String - 이 item의 쿠팡 카테고리 코드입니다.
        - coupangCategoryFullName: String - 이 item의 쿠팡 카테고리 이름입니다.
        - playautoCategoryCode: String - 이 item의 플레이오토 카테고리 코드입니다.
        - playautoCategoryFullName: String - 이 item의 플레이오토 카테고리 이름입니다.
        - esellersCategoryCode: String - 이 item의 이셀러스 카테고리 코드입니다.
        - esellersCategoryFullName: String - 이 item의 이셀러스 카테고리 이름입니다.
- 예시 쿼리:

```
query testQuery {
  item(key: "W000000") {
    createdAt
    updatedAt
    key
    name
    model
    production
    origin
    id
    price
    pricePolicy
    fixedPrice
    searchKeywords
    category {
      key
      name
      fullName
    }
    content
    shippingFee
    shippingType
    images(size: large) # size는 필수 파라미터입니다.
    status
    options {
      optionAttributes {
        name
        value
      }
      price
      quantity
    }
    taxFree
    adultOnly
    returnable
    noReturnReason
    guaranteedShippingPeriod
    openmarketSellable
    boxQuantity
    attributes
    closingTime
    returnCriteria
    metadata
```

```
  }
}
```

- 예시 코드:
  - example1 - 단일 상품 정보 조회하기
  - example2 - 이미지 여러 사이즈 조회하기

`allItems`

복수의 상품을 조회합니다. 한 번에 최대 1000개의 상품 정보를 조회할 수 있으며, pagination을 위한 cursor를 제공합니다. 자세한 사용법은 하단 예시를 참고하면 됩니다.

- 파라미터:
  - Pagination 관련 파라미터:
    - `after`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이후의 상품만을 불러옵니다.
    - `before`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이전의 상품만을 불러옵니다.
    - `first`: `Int` - 조건을 만족하는 모든 상품들 중 처음 몇 개의 상품을 불러올 지를 나타냅니다.
    - `last`: `Int` - 조건을 만족하는 모든 상품들 중 마지막 몇 개의 상품을 불러올 지를 나타냅니다.
    - `count`: `Int` - 조회된 상품의 수입니다.
    - 현재 `before`, `last`는 정상적으로 동작하지 않을 수 있으니 가급적 `after`, `first`를 위주로 사용하는 것을 권장합니다.
  - Search 관련 파라미터:
    - `dateFrom`: `Timestamp` - 상품 수정 시각이 이 값 이후인 것들만을 불러옵니다. Unix timestamp이며, 생략하면 불러올 수 있는 가장 오래된 시각으로 설정됩니다.
    - `dateTo`: `Timestamp` - 상품 수정 시각이 이 값 이전인 것들만을 불러옵니다. Unix timestamp이며, 생략하면 현재로 설정됩니다.
    - `minPrice`: `Int` - 가격 검색시 최저가입니다. 이 필드가 값을 가지면, 명시된 가격 이상의 가격을 가지는 상품만을 불러옵니다.
    - `maxPrice`: `Int` - 가격 검색시 최고가입니다. 이 필드가 값을 가지면, 명시된 가격 이하의 가격을 가지는 상품만을 불러옵니다.
    - `search`: `String` - 검색어입니다. 오너클랜에서 볼 수 있는 검색 기능과 동일한 기능을 제공합니다. 상품코드를 입력하면 해당 상품코드를 검색하고, 그렇지 않으면 키워드 검색을 진행합니다. 카테고리 검색이 필요한 경우 `category` 필드를 이용하면 됩니다.
    - `hasOptions`: `Boolean` - [DEPRECATED]
    - `vendor`: `ID` - 주어진 공급사 코드를 가진 공급사의 상품만을 불러옵니다.
    - `status`: `ItemStatus` - 주어진 상태에 해당하는 상품만을 불러옵니다. `ItemStatus`의 가능한 값들은 아래를 참고해주세요.
    - `category`: `ID` - 주어진 카테고리 코드와 해당 카테고리의 하위 카테고리에 해당하는 상품들만을 불러옵니다.
    - `attributes`: `[String]` - 주어진 속성을 모두 갖는 상품만을 불러옵니다. 가능한 속성 값들은 아래를 참고해주세요.

        - `sortBy`: `ItemSortCriteria` - 정렬 조건을 설정합니다. 설정하지 않으면 key로 정렬된 결과를 반환합니다. 가능한 정렬 기준은 아래를 참고해주세요.
        - `shippingLocationType`: `ShippingLocationType` - 국내 배송 상품 혹은 해외 배송 상품을 구분하여 요청할 수 있습니다. 설정하지 않으면 국내/해외 배송 상품 모두 반환합니다. 가능한 기준은 *ShippingLocationType* 항목을 참고해주세요.
        - `openmarketSellable`: `Boolean` - 오픈마켓 판매 가능 여부에 대한 조건입니다. 명시하지 않거나 `true`를 사용하면 오픈마켓 판매가 가능한 상품만을 불러오며, `false`를 사용하면 오픈마켓 판매가 불가능한 상품만을 불러옵니다. 자세한 내용은 *상품 운영 시 참조사항 - 오픈마켓 판매 가능 여부* 항목을 참고해주세요.
    - 데이터 관련 파라미터
        - `lang`: `Language` - 현재 쿼리의 `Text` 타입의 필드들에 대한 기본 언어를 선택합니다. 별도로 설정하지 않으면 기본값은 `ko_KR`입니다. 만약 `Text` 타입의 하위 필드에 `lang` 파라미터 값이 설정되면 해당 필드는 설정된 언어로 값을 반환합니다.
        - `currency`: `Currency` - 현재 쿼리의 `Price` 타입의 필드들에 대한 기본 화폐단위를 선택합니다. 별도로 설정하지 않으면 기본값은 `KRW`입니다. 만약 `Price` 타입의 하위 필드에 `currency` 파라미터 값이 설정되면 해당 필드는 설정된 화폐단위로 값을 반환합니다.

- 반환 값: `allItems`는 pagination을 위한 정보와 상품 정보를 동시에 담고 있습니다.
    - `pageInfo`: pagination을 위한 정보입니다.
        - `hasNextPage`: 데이터가 뒤에 더 있는지를 나타냅니다.
        - `hasPreviousPage`: 데이터가 앞에 더 있는지를 나타냅니다.
        - `startCursor`: 현재 페이지의 첫 데이터에 대한 cursor 값입니다. 이 값을 `before` 파라미터에 넘겨주면 정확하게 이전 페이지의 데이터를 가져올 수 있습니다.
        - `endCursor`: 현재 페이지의 마지막 데이터에 대한 cursor 값입니다. 이 값을 `after` 파라미터에 넘겨주면 정확하게 다음 페이지의 데이터를 가져올 수 있습니다.
    - `edges`: 상품 데이터입니다.
        - `cursor`: 해당 상품의 pagination cursor입니다.
        - `node`: 상품에 대한 정보입니다. `item` 쿼리에서 노출하는 정보와 동일합니다.

- 예시 쿼리:

```
# 검색 없이 기본 조건만으로 가져오는 쿼리입니다.
query {
  allItems {
    pageInfo {
      hasNextPage
      hasPreviousPage
      startCursor
      endCursor
    }
    edges {
      cursor
      node {
        createdAt
        updatedAt
        key
        name
        model
```

```
        production
        origin
        id
        price
        pricePolicy
        fixedPrice
        searchKeywords
        category {
          key
          name
          fullName
        }
        content
        shippingFee
        shippingType
        images(size: large)
        status
        options {
          optionAttributes {
            name
            value
          }
          price
          quantity
        }
        taxFree
        adultOnly
        returnable
        noReturnReason
        guaranteedShippingPeriod
        openmarketSellable
        boxQuantity
        attributes
        closingTime
        returnCriteria
        metadata
      }
    }
  }
}
```

```
# 카테고리 key가 "50000108" 혹은 그 하위 카테고리의 key이고 가격이 10000원 이상, 50000
원 이하인 상품 중 첫 50개만 가져옵니다.
query {
  allItems(minPrice: 10000, maxPrice: 50000, category: "50000108", first: 50) {
    pageInfo {
      hasNextPage
      hasPreviousPage
      startCursor
      endCursor
    }
```

```
    edges {
      cursor
      node {
        createdAt
        updatedAt
        key
        name
        model
        production
        origin
        id
        price
        pricePolicy
        fixedPrice
        searchKeywords
        category {
          key
          name
          fullName
        }
        content
        shippingFee
        shippingType
        images(size: large)
        status
        options {
          optionAttributes {
            name
            value
          }
          price
          quantity
        }
        taxFree
        adultOnly
        returnable
        noReturnReason
        guaranteedShippingPeriod
        openmarketSellable
        boxQuantity
        attributes
        closingTime
        returnCriteria
        metadata
      }
    }
  }
}
```

```
# 카테고리 key가 "50000108" 혹은 그 하위 카테고리의 key인 상품을 수정일 기준 내림차순
정렬 후 첫 30개를 가져옵니다.
```

```
query {
  allItems(category: "50000108", sortBy: dateDesc, first: 30) {
    pageInfo {
      hasNextPage
      hasPreviousPage
      startCursor
      endCursor
    }
    edges {
      cursor
      node {
        createdAt
        updatedAt
        key
        name
        model
        production
        origin
        id
        price
        pricePolicy
        fixedPrice
        searchKeywords
        category {
          key
          name
          fullName
        }
        content
        shippingFee
        shippingType
        images(size: large)
        status
        options {
          optionAttributes {
            name
            value
          }
          price
          quantity
        }
        taxFree
        adultOnly
        returnable
        noReturnReason
        guaranteedShippingPeriod
        openmarketSellable
        boxQuantity
        attributes
        closingTime
        returnCriteria
        metadata
      }
    }
```

```
    }
}
```

- 예시 코드:
  - example1 - 상품 수정일 기준 검색
  - example2 - 가격 검색
  - example3 - 공급사 코드 검색
  - example4 - 카테고리 코드 검색
  - example5 - 복합 검색(가격 & 수정일 & 카테고리 코드)
  - example6 - 페이지네이션1(특정 상품 이후의 상품 중 첫 500개)
  - example7 - 페이지네이션2(특성 상품 이전의 상품 중 마지막 100개)
  - example8 - 페이지네이션 & 복합검색

`itemHistories`

오너클랜의 상품 품절/단종/재입고(변경 이력) 리스트를 조회합니다.

- 참고사항:
  - 오너클랜 웹사이트의 상품 품절/단종/재입고 리스트 조회와 거의 동일한 기능을 제공합니다.
  - 단, 오너클랜 웹사이트에서는 21가지 유형에 대해서 별도 조회가 가능하고, 나머지 유형은 *기/E/*로 처리하는 것에 비해, API에서는 41가지 유형에 대해 모두 별도 조회가 가능합니다.
  - 한편, 오너클랜 웹사이트에서 제공하는 검색어, 복수 개 상품코드, 공급사 코드 검색 기능은 제공하지 않습니다.
  - `itemKey` 파라미터를 명시하여 단일 상품에 대해 조회하는 경우에는 날짜 범위 제한이 없으나, `itemKey` 파라미터를 명시하지 않은 경우에는 최대 7일 간격으로만 조회가 가능합니다.
- 파라미터:
  - Pagination 관련 파라미터:
    - `after`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이후의 내역만을 불러옵니다.
    - `before`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이전의 내역만을 불러옵니다.
    - `first`: `Int` - 조건을 만족하는 모든 내역들 중 처음 몇 개의 내역을 불러올 지를 나타냅니다.
    - `last`: `Int` - 조건을 만족하는 모든 내역들 중 마지막 몇 개의 내역을 불러올 지를 나타냅니다.
    - 현재 `before`, `last`는 정상적으로 동작하지 않을 수 있으니 가급적 `after`, `first`를 위주로 사용하는 것을 권장합니다.
  - Search 관련 파라미터:
    - `dateFrom`: `Timestamp` - 내역이 등록된 일자가 이 값 이후인 것들만을 불러옵니다. Unix timestamp이며, 생략하면 24시간 전으로 설정됩니다.
    - `dateTo`: `Timestamp` - 내역이 등록된 일자가 이 값 이전인 것들만을 불러옵니다. Unix timestamp이며, 생략하면 현재로 설정됩니다.
    - `kind`: `ItemHistoryKind` - 기록 종류에 따른 검색입니다. 가능한 기록의 종류는 *ItemHistoryKind* 항목을 참고해주세요.
    - `itemKey`: `ID` - 주어진 상품 Key에 해당하는 내역만 조회합니다.

- 반환 값: `itemHistories`는 pagination을 위한 정보와 내역에 대한 정보를 동시에 담고 있습니다.
    - `pageInfo`: pagination을 위한 정보입니다.
        - `hasNextPage`: 데이터가 뒤에 더 있는지를 나타냅니다.
        - `hasPreviousPage`: 데이터가 앞에 더 있는지를 나타냅니다.
        - `startCursor`: 현재 페이지의 첫 데이터에 대한 cursor 값입니다. 이 값을 `before` 파라미터에 넘겨주면 정확하게 이전 페이지의 데이터를 가져올 수 있습니다.
        - `endCursor`: 현재 페이지의 마지막 데이터에 대한 cursor 값입니다. 이 값을 `after` 파라미터에 넘겨주면 정확하게 다음 페이지의 데이터를 가져올 수 있습니다.
    - `edges`: 상품 품절/단종/재입고(변경 이력) 데이터입니다.
        - `cursor`: 해당 내역의 pagination cursor입니다.
        - `node`: 해당 내역의 내용입니다.
            - `itemKey`: 내역이 등록된 상품의 key입니다.
            - `kind`: 내역의 종류입니다. 가능한 종류는 *ItemHistoryKind* 항목을 참고해주세요.
            - `title`: 내역의 이름입니다.
            - `valueBefore`: 변경 내역이 이전, 이후 값을 갖는 경우 변경 이전의 값입니다.
            - `valueAfter`: 변경 내역이 이전, 이후 값을 갖는 경우 변경 이후의 값입니다.
            - `createdAt`: 변경 내역이 기록된 시각입니다. Unix timestamp로 주어집니다.
- 예시 쿼리:

```
# 특정 상품에 대한 변경 이력을 최신 순으로 100개 가져오는 쿼리.
query {
  itemHistories(first: 100, itemKey: "W000001") {
    pageInfo {
      hasNextPage
      hasPreviousPage
      startCursor
      endCursor
    }
    edges {
      node {
        itemKey
        kind
        title
        valueBefore
        valueAfter
        createdAt
      }
    }
  }
}
```

```
# 모든 상품에 대해 품절 이력을 최신 순으로 1000개 가져오는 쿼리.
query {
  itemHistories(first: 1000, kind: soldout) {
    pageInfo {
      hasNextPage
```

```
      hasPreviousPage
      startCursor
      endCursor
    }
    edges {
      node {
        itemKey
        kind
        title
        valueBefore
        valueAfter
        createdAt
      }
    }
  }
}
```

```
# 특정 날짜(이 예시에서는 2019년 2월 1일) 단종 이력을 최신 순으로 1000개 가져오는 쿼리.
query {
  itemHistories(first: 1000, dateFrom: 1548946800000, dateTo: 1549033200000, kind:
discontinued) {
    pageInfo {
      hasNextPage
      hasPreviousPage
      startCursor
      endCursor
    }
    edges {
      node {
        itemKey
        kind
        title
        valueBefore
        valueAfter
        createdAt
      }
    }
  }
}
```

`itemsByKeys`

여러 상품을 상품의 `key`(오너클랜 상품코드) 리스트를 기준으로 조회합니다.

- 참고사항:
  - 한 번의 요청에 최대 5000개의 `key`를 요청할 수 있습니다.
  - 존재하지 않는 `Item`의 `key`인 경우에는 해당 `key`에 대한 상품 정보를 제외하고 반환합니다.
    - 예를 들어 100개의 상품에 대한 정보를 요청했고, 그 중 5개의 상품이 존재하지 않는다면 모두 반환값은 길이가 95인 배열입니다.

- 파라미터:
  - `keys`: `[String]` - 상품 데이터를 조회할 상품들의 `key`(오너클랜 상품코드)의 배열입니다.
- 반환 값: `item` 쿼리의 반환 형태와 동일한 형태의 데이터들의 배열입니다.
- 예시 쿼리:

```
# 3개의 상품(실제로 존재하는 상품이 아닌 예시 코드)에 대한 상품 정보를 불러오는 쿼리.
# 아래 상품코드는 실제 존재하는 상품이 아니므로 사용할 때는 변경해서 사용.
query ItemsByKeys {
  itemsByKeys(keys: ["W999999", "W999998", "W999997"]) {
    createdAt
    updatedAt
    key
    name
    model
    production
    origin
    id
    price
    pricePolicy
    fixedPrice
    searchKeywords
    category {
      key
      name
    }
    content
    shippingFee
    shippingType
    images(size: large) # size는 필수 파라미터입니다.
    status
    options {
      optionAttributes {
        name
        value
      }
      price
      quantity
      key
    }
    taxFree
    adultOnly
    returnable
    noReturnReason
    guaranteedShippingPeriod
    openmarketSellable
    boxQuantity
    attributes
    closingTime
    metadata
```

```
  }
}
```

- 예시 코드:
  - example1 - 상품 key값의 리스트로 검색

category

오너클랜의 단일 카테고리 정보를 조회합니다.

- 참고사항:
  - 모든 최상위 카테고리의 상위 카테고리는 `"ROOT"`라는 이름을 가지며, 코드는 `"00000000"`입니다. 이를 이용해 최상위 카테고리만 전부 가져오거나, 모든 카테고리를 가져올 수 있습니다. 자세한 내용은 아래 예시 코드를 참고해주세요.
- 파라미터:
  - `key`: `ID` - 정보를 조회할 카테고리의 key입니다. 실제 타입은 `String`이며, 오너클랜 카테고리 코드와 같습니다.
- 반환 데이터: 카테고리 정보를 반환하며, 다음 데이터에 액세스 할 수 있습니다.
  - `key`: `String` - 카테고리의 오너클랜 코드입니다.
  - `id`: `String` - 카테고리의 API ID입니다.
  - `name`: `String` - 카테고리의 이름입니다. `lang` 파라미터를 가질 수 있으며, 언어를 지정할 수 있습니다. 현재 지원되는 언어는 `"ko_KR"`입니다. `lang` 파라미터는 생략하면 `"ko_KR"`로 적용됩니다.
  - `fullName`: `String` - 현재 카테고리의 전체 이름입니다. 최상단 카테고리의 이름부터 현재 카테고리까지의 이름을 모두 포함합니다.
  - `attributes`: `[String]` - 카테고리의 속성 정보입니다. 자세한 항목은 아래를 참고해주세요.
  - `parent`: `Category` - 상위 카테고리(1단계 위)입니다. 상위 카테고리에 대해서는 `key`, `id`, `name`만 조회할 수 있습니다.
  - `children`: `[Category]` - 하위 카테고리(1단계 아래)들입니다. 하위 카테고리 각각에 대해서는 `key`, `id`, `name`만 조회할 수 있습니다.
  - `ancestors`: `[Category]` - 상위 카테고리 전체입니다. 상위 카테고리 각각에 대해서는 `key`, `id`, `name`만 조회할 수 있습니다.
  - `descendants`: `CategoriesConnection` - 하위 카테고리 전체이고, pagination을 위한 정보와 카테고리 정보를 동시에 담고 있습니다.
    - 파라미터:
      - Pagination 관련 파라미터:
        - `after`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이후의 카테고리만을 불러옵니다.
        - `before`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이전의 카테고리만을 불러옵니다.
        - `first`: `Int` - 조건을 만족하는 모든 카테고리들 중 처음 몇 개의 카테고리를 불러올 지를 나타냅니다.
        - `last`: `Int` - 조건을 만족하는 모든 카테고리들 중 마지막 몇 개의 카테고리를 불러올 지를 나타냅니다.

        - 현재 `before`, `last`는 정상적으로 동작하지 않을 수 있으니 가급적 `after`, `first`를 위주로 사용하는 것을 권장합니다.
    - `pageInfo`: pagination을 위한 정보입니다.
        - `hasNextPage`: 데이터가 뒤에 더 있는지를 나타냅니다.
        - `hasPreviousPage`: 데이터가 앞에 더 있는지를 나타냅니다.
        - `startCursor`: 현재 페이지의 첫 데이터에 대한 cursor 값입니다. 이 값을 `before` 파라미터에 넘겨주면 정확하게 이전 페이지의 데이터를 가져올 수 있습니다.
        - `endCursor`: 현재 페이지의 마지막 데이터에 대한 cursor 값입니다. 이 값을 `after` 파라미터에 넘겨주면 정확하게 다음 페이지의 데이터를 가져올 수 있습니다.
    - `edges`: 카테고리 데이터입니다.
        - `cursor`: 해당 카테고리의 pagination cursor입니다.
        - `node`: 해당 카테고리의 데이터입니다. `Category`에서 접근할 수 있는 모든 필드를 동일하게 접근할 수 있습니다.

- 예시 코드:

```
# 단일 카테고리 조회
query {
  category(key: "50000108") {
    key
    name
    parent {
      key
      name
      parent {
        key
        name
      }
    }
    children {
      key
      name
    }
    ancestors {
      key
      name
    }
    descendants {
      edges {
        node {
          key
          name
        }
      }
    }
  }
}
```

```
# 가상의 최상위 카테고리 정보 가져오기.
query {
  category(key: "00000000") {
    key
    name
  }
}
```

```
# 실제로 쓰이는 최상위 카테고리의 리스트 가져오기. (가상의 카테고리 이용)
# 반환된 데이터의 children 필드의 값들이 실제로 쓰이는 최상위 카테고리들입니다.
query {
  category(key: "00000000") {
    key
    name
    children {
      key
      name
    }
  }
}
```

```
# 전체 카테고리 정보 가져오기(1~100)
query {
  category(key: "00000000") {
    descendants(first: 100) {
      pageInfo {
        hasNextPage
        hasPreviousPage
        startCursor
        endCursor
      }
      edges {
        cursor
        node {
          key
          name
        }
      }
    }
  }
}
```

```
# 전체 카테고리 정보 가져오기(101~200)
query {
  category(key: "00000000") {
    descendants(after: "100", first: 100) {
```

```
        pageInfo {
          hasNextPage
          hasPreviousPage
          startCursor
          endCursor
        }
        edges {
          cursor
          node {
            key
            name
            fullName
          }
        }
      }
    }
  }
```

- 예시 코드:
  - example1 - 단일 카테고리 정보 가져오기
  - example2 - 전체 카테고리 가져오기
  - example3 - 전체 카테고리 가져오기 & 페이지네이션

order

단일 주문 정보를 조회합니다.

- 파라미터:
  - **key**: String - 정보를 조회할 주문 내역의 key입니다. 오너클랜 주문 코드와 같습니다.
- 반환 데이터: 주문 정보를 반환하며, 다음 내용을 액세스 할 수 있습니다.
  - key: String - 주문 내역의 key입니다. 오너클랜 주문 코드와 같습니다.
  - id: String - 주문 내역의 API ID입니다.
  - products: [OrderProduct] - 주문 제품 정보의 배열입니다. 하위 필드로 조회할 수 있는 내역은 아래와 같습니다.
    - quantity: Int - 주문 제품의 수량입니다.
    - price: Float - 수량이 반영되지 않은, 주문 제품의 가격입니다. currency 파라미터를 가질 수 있으며, 현재 지원되는 Currency는 KRW입니다.
    - shippingType: ShippingType - 배송비 결제 방식입니다. 가능한 값들은 *데이터 타입* - *ShippingType* 항목을 참고해주시면 됩니다.
    - itemKey: String - 주문 제품의 상품 key입니다.
    - itemOptionInfo: Object - 주문 제품의 옵션 정보입니다. 하위 필드로 조회할 수 있는 내역은 아래와 같습니다.
      - optionAttributes: [Object] - 상품의 옵션 정보입니다.
        - name: String - 옵션명입니다.
        - value: String - 옵션값입니다.

            - price: Float - 주문 제품의 옵션 가격입니다. 옵션 추가금액이 아닌, 추가금액이 반영된 전체 금액이므로 참고해주시면 됩니다.
        - trackingNumber: String - 택배 운송장 번호입니다.
        - shippingCompanyName: String - 택배사 이름입니다.
        - shippedDate: Int - 택배 운송장 입력 일시입니다. Unix timestamp로 주어집니다.
        - additionalAttributes: [Object] - 입력형 옵션을 포함한, 추가 속성 정보입니다.
            - name: String - 속성 이름입니다.
            - value: String - 속성 값입니다.
        - taxFree: Boolean - 면세상품인지의 여부입니다.
    - status: OrderStatus - 주문 상태입니다. 가능한 값들은 *데이터 타입 - OrderStatus* 항목을 참고해주시면 됩니다.
    - shippingInfo: ShippingInfo - 배송 정보입니다. 하위 필드로 조회할 수 있는 내역은 아래와 같습니다.
        - sender: Sender - 보내는 사람의 정보입니다. 판매사 계정의 정보와 동일합니다.
            - name: String - 보내는 사람의 이름입니다.
            - phoneNumber: String - 보내는 사람의 연락처입니다.
            - email: String - 보내는 사람의 이메일 주소입니다.
        - recipient: Recipient - 받는 사람의 정보입니다.
            - name: String - 받는 사람의 이름입니다.
            - phoneNumber: String - 받는 사람의 연락처입니다.
            - destinationAddress: Address - 받는 주소입니다. Address에 대한 내용은 *데이터 타입 - Address* 항목을 참고해주시면 됩니다.
        - shippingFee: Float - 배송비입니다.
    - createdAt: Int - 주문이 DB에 등록된 시각입니다. Unix timestamp로 주어집니다.
    - updatedAt: Int - 주문이 최종적으로 업데이트 된 시각입니다. Unix timestamp로 주어집니다.
    - note: String - 기타 메모 사항입니다. 오너클랜 웹사이트 주문내역 조회 페이지의 *원장주문코드* 항목에 해당합니다.
    - ordererNote: String - 최종 소비자가 요청한 주문 메모입니다. 오너클랜 웹사이트 주문내역 조회 페이지의 *고객메모* 항목에 해당합니다.
    - sellerNote: String - 판매사 메모 사항입니다. 오너클랜 웹사이트 주문내역 조회 페이지의 *주문관리 메모* 항목에 해당합니다.
    - isBeingMediated: Boolean - 중재중인 주문인지의 여부입니다.
    - adjustments: [Adjustment] - 조정 내역이 있는 경우 조정 내역입니다. Adjustment에 대한 내용은 *데이터 타입 - Adjustment* 항목을 참고해주시면 됩니다.
    - transactions: [Transaction] - 적립금 사용 내역입니다. Transaction에 대한 내용은 *데이터 타입 - Transaction* 항목을 참고해주시면 됩니다.
    - refundDetails: [RefundedOrder]- 반품 요청 내역들입니다. RefundedOrder에 대한 내용은 *데이터 타입 - RefundedOrder* 항목을 참고해주시면 됩니다.

- 예시 쿼리:

```
query {
  order(key: "20190000000000000000A") {
    key
    id
    products {
      quantity
```

```
      price
      shippingType
      itemKey
      itemOptionInfo {
        optionAttributes {
          name
          value
        }
        price
      }
      trackingNumber
      shippingCompanyName
      shippedDate
      additionalAttributes {
        key
        value
      }
      taxFree
    }
    status
    shippingInfo {
      sender {
        name
        phoneNumber
        email
      }
      recipient {
        name
        phoneNumber
        destinationAddress {
          addr1
          addr2
          postalCode
        }
      }
      shippingFee
    }
    createdAt
    updatedAt
    note
    ordererNote
    sellerNote
    isBeingMediated
    adjustments {
      reason
      price
      taxFree
    }
    transactions {
      key
      id
      kind
      status
      amount {
```

```
        currency
        value
      }
      createdAt
      updatedAt
      closedAt
      note
    }
  }
}
```

- 예시 코드:
  - example1 - 단일 주문내역 조회하기

`allOrders`

복수의 주문내역을 조회합니다. 한 번에 최대 1000개의 주문 내역을 조회할 수 있으며, pagination을 위한 cursor를 제공합니다. 자세한 사용법은 하단 예시를 참고하면 됩니다.

- 파라미터:
  - Pagination 관련 파라미터:
    - `after`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이후의 주문내역만을 불러옵니다.
    - `before`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이전의 주문내역만을 불러옵니다.
    - `first`: `Int` - 조건을 만족하는 모든 주문내역들 중 처음 몇 개의 주문내역을 불러올 지를 나타냅니다.
    - `last`: `Int` - 조건을 만족하는 모든 주문내역들 중 마지막 몇 개의 주문내역을 불러올 지를 나타냅니다.
    - 현재 `before`, `last`는 정상적으로 동작하지 않을 수 있으니 가급적 `after`, `first`를 위주로 사용하는 것을 권장합니다.
  - Search 관련 파라미터:
    - `dateFrom`: `Timestamp` - 주문 내역이 생성된 시각이 이 값 이후인 것들만을 불러옵니다. Unix timestamp이며, 생략하면 90일 전으로 설정됩니다.
    - `dateTo`: `Timestamp` - 주문 내역이 생성된 시각이 이 값 이전인 것들만을 불러옵니다. Unix timestamp이며, 생략하면 현재로 설정됩니다.
    - `note`: `String` - `note` 필드(원장주문코드)의 값에 입력한 값이 포함되는 주문 내역만을 검색합니다. 검색시 `dateFrom`과 `dateTo`를 사용하여 검색 기간을 90일 또는 그 이내로 지정해야 합니다.
    - `sellerNote`: `String` - `sellerNote` 필드(주문 관리 코드)의 값에 입력한 값이 포함되는 주문 내역만을 검색합니다. 검색시 `dateFrom`과 `dateTo`를 사용하여 검색 기간을 90일 또는 그 이내로 지정해야 합니다.
    - `status`: `OrderStatus` - 주어진 상태에 해당하는 주문내역만을 불러옵니다. `OrderStatus`의 가능한 값들은 아래를 참고해주세요.
    - `shippedAfter`: `Timestamp` - 송장번호가 입력된 시각이 이 값 이후인 것들만을 불러옵니다. Unix timestamp입니다.

        - `shippedBefore`: `Timestamp` - 송장번호가 입력된 시각이 이 값 이전인 것들만을 불러옵니다. Unix timestamp입니다.
        - `shippedBefore`는 `shippedAfter` 없이 단독으로 사용할 수 없습니다.
        - `shippedBefore` 또는 `shippedAfter`를 이용해 기간을 명시하는 경우, 기간은 반드시 90일 이내여야 합니다.

- 반환 값: `allOrders`는 pagination을 위한 정보와 주문내역 정보를 동시에 담고 있습니다.
    - `pageInfo`: pagination을 위한 정보입니다.
        - `hasNextPage`: 데이터가 뒤에 더 있는지를 나타냅니다.
        - `hasPreviousPage`: 데이터가 앞에 더 있는지를 나타냅니다.
        - `startCursor`: 현재 페이지의 첫 데이터에 대한 cursor 값입니다. 이 값을 `before` 파라미터에 넘겨주면 정확하게 이전 페이지의 데이터를 가져올 수 있습니다.
        - `endCursor`: 현재 페이지의 마지막 데이터에 대한 cursor 값입니다. 이 값을 `after` 파라미터에 넘겨주면 정확하게 다음 페이지의 데이터를 가져올 수 있습니다.
    - `edges`: 주문내역 데이터입니다.
        - `cursor`: 해당 주문내역의 pagination cursor입니다.
        - `node`: 주문내역에 대한 정보입니다. `order` 쿼리에서 노출하는 정보와 동일합니다.

- 예시 쿼리:

```
# 검색 없이 기본 조건만으로 가져오는 쿼리입니다.
query {
  allOrders {
    pageInfo {
      hasNextPage
      hasPreviousPage
      startCursor
      endCursor
    }
    edges {
      node {
        key
        id
        products {
          quantity
          price
          shippingType
          itemKey
          itemOptionInfo {
            optionAttributes {
              name
              value
            }
            price
          }
          trackingNumber
          shippingCompanyName
          shippedDate
          additionalAttributes {
            key
```

```
            value
          }
          taxFree
        }
        status
        shippingInfo {
          sender {
            name
            phoneNumber
            email
          }
          recipient {
            name
            phoneNumber
            destinationAddress {
              addr1
              addr2
              postalCode
            }
          }
          shippingFee
        }
        createdAt
        updatedAt
        note
        ordererNote
        sellerNote
        isBeingMediated
        adjustments {
          reason
          price
          taxFree
        }
        transactions {
          key
          id
          kind
          status
          amount {
            currency
            value
          }
          createdAt
          updatedAt
          closedAt
          note
        }
      }
    }
  }
}
```

- 예시 코드:

    - example1 - 기본 검색
    - example2 - 날짜 범위 검색
    - example3 - 날짜 범위 및 주문 상태 검색
    - example4 - 원장주문코드 검색
    - example5 - 주문관리 메모 검색
    - example6 - 페이지네이션
    - example7 - 송장번호 입력 날짜 범위 검색

`createOrder`

새 주문을 등록합니다.

- 파라미터:
    - `input`: `object(OrderInput)` - 입력할 데이터입니다.
    - `simulationResult`: `[object]` - 등록할 주문의 입력 데이터로 `simulateCreateOrder` 쿼리를 호출한 결과입니다. 이 필드에 값이 주어지는 경우, 시뮬레이션 결과와 실제 등록 예정인 주문의 내역이 다르면 주문에 실패합니다. 자세한 예시는 example2를 참고하면 되고, 입력 형식은 *데이터 형식 - `CreateOrderSimulationResult` & `CreateOrderSimulationResultInput`*을 참고하면 됩니다.
- 입력 데이터 (굵은 글씨는 필수 필드입니다):
    - `sender`: `SenderInput` - 보내는 사람의 정보입니다. `SenderInput` 타입에 대한 정보는 *데이터 형식 - `SenderInput`* 항목을 참고하면 됩니다. 입력하지 않으면 판매사 계정 기본값(오너클랜 웹사이트에서 직접 개별 주문시에 사용되는 값과 동일)이 사용됩니다.
    - **`recipient`**: `RecipientInput` - 받는 사람의 정보입니다. `RecipientInput` 타입에 대한 정보는 *데이터 형식 - `RecipientInput`* 항목을 참고하면 됩니다.
    - **`products`**: `[OrderProductInput]` - 주문할 상품들의 리스트입니다. `OrderProduct` 타입에 대한 정보는 *데이터 형식 - `OrderProductInput`* 항목을 참고하면 됩니다.
    - `note`: `String` - 주문 건에 대한 메모(원장주문코드)입니다.
    - `sellerNote`: `String` - 주문 건에 대한 판매자 메모(주문관리코드)입니다.
    - `ordererNote`: `String` - 주문 건에 대한 최종 구매자의 배송시 요청사항입니다.
    - `customsClearanceCode`: `CustomsClearanceCodeInput` - 해외배송상품 주문시 입력하는 데이터입니다. 해외배송 상품이 아닌 경우 이 필드를 아예 생략하는 것을 권장합니다.
        - `value`: `String` - 입력할 데이터입니다.
        - `type`: `CustomsClearanceCodeType` 타입에 대한 정보는 *데이터 형식 - `CustomsClearanceCodeType`* 항목을 참고하면 됩니다.
- 반환 데이터:
    - 새로 생성된 주문 정보들을 반환합니다. `createOrder` 단일 쿼리 요청에 대한 결과는 주문한 상품의 내용에 따라서 여러 개의 주문으로 나뉘어 생성될 수 있습니다.
        - 주문은 다음과 같은 경우에 별도의 주문으로 나뉘어 생성됩니다.
            - 공급사(`Item.metadata.vendorKey` 필드)가 다른 경우.
            - 배송비 부과 타입(`Item.shippingType` 필드)가 다른 경우.
            - 예를 들어, 공급사 A의 선불 상품 2개, B의 무료배송 상품 1개, 착불 상품 1개를 주문했다면 주문은 모두 3개가 생성됩니다.
- 예시 쿼리: 아래 2개의 코드 중 첫 번째는 실제 쿼리이고, 두 번째는 쿼리에서 쓰이는 `$input` variable을 정의한 것입니다. GraphQL Playground에서 테스트할 때는 쿼리를 입력하는 공간 아래에 *QUERY VARIABLES*에 입력하면 됩니다.(예시 데이터이므로 그대로 붙여넣어도 작동하지는 않습니다.)

```
mutation CreateOrder($input: OrderInput!) {
  createOrder(input: $input) {
    key
    id
    products {
      quantity
      price
      shippingType
      itemKey
      productName
      itemOptionInfo {
        optionAttributes {
          name
          value
        }
        price
      }
      trackingNumber
      shippingCompanyCode
      shippingCompanyName
      shippedDate
      additionalAttributes {
        key
        value
      }
      taxFree
    }
    status
    shippingInfo {
      sender {
        name
        phoneNumber
        email
      }
      recipient {
        name
        phoneNumber
        destinationAddress {
          addr1
          addr2
          postalCode
        }
      }
      shippingFee
    }
    createdAt
    updatedAt
    note
    ordererNote
    sellerNote
    isBeingMediated
    adjustments {
      reason
```

```
      price
      taxFree
    }
    transactions {
      key
      id
      kind
      status
      amount {
        currency
        value
      }
      createdAt
      updatedAt
      closedAt
      note
    }
  }
}
```

```
{
  "input": {
    "sender": {
      "name": "보내는이",
      "phoneNumber": "010-1234-5678",
      "email": "your_id@email.com"
    },
    "recipient": {
      "name": "받는이",
      "phoneNumber": "010-8765-4321",
      "destinationAddress": {
        "addr1": "서울 금천구 가산디지털1로 128",
        "addr2": "808호",
        "postalCode": "08507"
      }
    },
    "products": [
      {
        "quantity": 4,
        "itemKey": "W999999",
        "optionAttributes": [
          "블랙"
        ]
      },
      {
        "quantity": 3,
        "itemKey": "W999999",
        "optionAttributes": [
          "로즈골드"
        ]
      },
```

```
        {
          "quantity": 1,
          "itemKey": "W999998",
          "optionAttributes": [
            "골드"
          ]
        },
        {
          "quantity": 2,
          "itemKey": "W999998",
          "optionAttributes": [
            "실버"
          ]
        },
        {
          "quantity": 5,
          "itemKey": "W999997",
          "optionAttributes": [
            "화이트",
            "토끼당근"
          ]
        },
        {
          "quantity": 2,
          "itemKey": "W999996",
          "optionAttributes": [
            "그린티"
          ]
        },
        {
          "quantity": 1,
          "itemKey": "W999995",
          "optionAttributes": []
        }
      ],
      "note": "원장주문코드",
      "sellerNote": "주문관리코드",
      "ordererNote": "배송시 요청사항",
      "customsClearanceCode": {
        "type":"PersonalNumber",
        "value":"P012345678910"
      }
    }
  }
```

- 예시 코드:
  - example1 - 단일 주문 생성하기
  - example2 - 시뮬레이션 결과 검증이 들어간 단일 주문 생성하기
  - example3 - 해외배송 상품 주문 생성하기

simulateCreateOrder

주문을 실제로 등록하지 않고, 등록했을 때 예상되는 상품 금액과 배송비 정보를 제공합니다.

- 파라미터:
  - `input`: `object(OrderInput)` - 입력할 데이터입니다. `createOrder` 입력과 동일합니다.
- 입력 데이터: `createOrder` 입력과 동일합니다.
- 반환 데이터: 상품 금액 정보와 배송비 정보들을 반환합니다. 아래 3개 필드가 있는 object의 배열로 반환되며, 각 object는 1개의 주문에 대응됩니다.
  - `itemAmounts`: `[object]` - 상품 관련 금액입니다.
    - `amount`: `Int` - 주문 수량이 반영된, 상품의 최종 금액입니다.
    - `itemKey`: `String` - 상품 Key(오너클랜 상품코드)입니다.
  - `shippingAmount`: `Int` - 추가 배송비 등이 반영된, 최종 배송비입니다.
  - `extraShippingFeeExists`: `Boolean` - 추가 배송비 반영 여부입니다.
- 예시 쿼리: 아래 2개의 코드 중 첫 번째는 실제 쿼리이고, 두 번째는 쿼리에서 쓰이는 `$input` variable을 정의한 것입니다. GraphQL Playground에서 테스트할 때는 쿼리를 입력하는 공간 아래에 *QUERY VARIABLES*에 입력하면 됩니다.

```
mutation SimulateCreateOrder($input: OrderInput!) {
  simulateCreateOrder(input: $input) {
    itemAmounts {
      amount
      itemKey
    }
    shippingAmount
    extraShippingFeeExists
  }
}
```

```
{
  "input": {
    "sender": {
      "name": "보내는이",
      "phoneNumber": "010-1234-5678",
      "email": "your_id@email.com"
    },
    "recipient": {
      "name": "받는이",
      "phoneNumber": "010-8765-4321",
      "destinationAddress": {
        "addr1": "서울 금천구 가산디지털1로 128",
        "addr2": "808호",
        "postalCode": "08507"
      }
    },
    "products": [
      {
        "quantity": 4,
        "itemKey": "W999999",
        "optionAttributes": [
```

```
            "블랙"
          ]
        },
        {
          "quantity": 3,
          "itemKey": "W999999",
          "optionAttributes": [
            "로즈골드"
          ]
        },
        {
          "quantity": 1,
          "itemKey": "W999998",
          "optionAttributes": [
            "골드"
          ]
        },
        {
          "quantity": 2,
          "itemKey": "W999998",
          "optionAttributes": [
            "실버"
          ]
        },
        {
          "quantity": 5,
          "itemKey": "W999997",
          "optionAttributes": [
            "화이트",
            "토끼당근"
          ]
        },
        {
          "quantity": 2,
          "itemKey": "W999996",
          "optionAttributes": [
            "그린티"
          ]
        },
        {
          "quantity": 1,
          "itemKey": "W999995",
          "optionAttributes": []
        }
      ],
      "note": "원장주문코드",
      "sellerNote": "주문관리코드",
      "ordererNote": "배송시 요청사항",
      "customsClearanceCode": "통관고유번호"
    }
}
```

- 예시 코드:

    - example1 - 단일 주문 시뮬레이션

updateOrderNotes

기존 주문의 원청주문코드, 주문관리메모를 업데이트합니다.

- 파라미터:
  - `key`: `String` - 원청주문코드 또는 주문관리메모를 수정할 주문의 주문코드입니다.
  - `input`: `object(OrderUpdateNotesInput)` - 수정할 데이터입니다.
- 입력 데이터:
  - `note`: `String` - 원청주문코드입니다.
  - `sellerNotes`: `[SellerNoteInput]` - 주문관리메모입니다.
    - `sellerNote`: `String` - 설정할 주문관리메모입니다.
- 주의사항:
  - 주문관리메모는 하나의 주문에 주문 상품이 여러 개인 경우 주문 전체에 설정할 수도 있고 주문 상품별로 설정할 수도 있는데 다음 원칙을 따라야 합니다.
    - 주문 전체에 대해 설정하는 경우: 배열에 하나의 object만 있어야 하며, 주문 전체에 대해 하나의 주문관리메모만 노출됩니다.(Order.sellerNote)
    - 주문 상품별로 설정하는 경우: READ API(order)를 통해 얻은 주문 상품 순서대로 입력하고 싶은 주문관리메모 object를 배열로 지정하면 됩니다. 주문 전체에 대한 주문관리메모는 첫 번째 주문상품의 주문관리메모가 노출되며, 주문 상품 각각에 대해 주문관리메모가 노출됩니다.(Order.products.sellerNote)
- 반환 데이터:
  - 업데이트 된 주문 정보를 반환합니다. `order` 쿼리의 결과와 동일합니다.
- 예시 쿼리: 아래 2개의 코드 중 첫 번째는 실제 쿼리이고, 두 번째는 쿼리에서 쓰이는 `$input` variable을 정의한 것입니다. GraphQL Playground에서 테스트할 때는 쿼리를 입력하는 공간 아래에 *QUERY VARIABLES*에 입력하면 됩니다.(예시 데이터이므로 그대로 붙여넣어도 작동하지는 않습니다.)

```
mutation UpdateOrderNotes($input: OrderUpdateNotesInput!) {
  updateOrderNotes(key: "20200220000000000000A", input: $input) {
    key
    id
    products {
      quantity
      price
      shippingType
      itemKey
      productName
      itemOptionInfo {
        optionAttributes {
          name
          value
        }
        price
      }
      trackingNumber
      shippingCompanyCode
      shippingCompanyName
      shippedDate
```

```
      additionalAttributes {
        key
        value
      }
      taxFree
      sellerNote
    }
    status
    shippingInfo {
      sender {
        name
        phoneNumber
        email
      }
      recipient {
        name
        phoneNumber
        destinationAddress {
          addr1
          addr2
          postalCode
        }
      }
      shippingFee
    }
    createdAt
    updatedAt
    note
    ordererNote
    sellerNote
    isBeingMediated
    adjustments {
      reason
      price
      taxFree
    }
    transactions {
      key
      id
      kind
      status
      amount {
        currency
        value
      }
      createdAt
      updatedAt
      closedAt
      note
    }
  }
}
```

```
{
  "input": {
    "note": "원장주문코드 - 수정",
    "sellerNotes": [
      {
        "sellerNote": "주문관리메모1"
      },
      {
        "sellerNote": "주문관리메모2"
      },
      {
        "sellerNote": "주문관리메모3"
      }
    ]
  }
}
```

- 예시 코드:
  - example1 - 원장주문코드 수정
  - example2 - 주문관리메모 전체 수정
  - example3 - 주문관리메모 상품별 수정

`cancelOrder`

주문을 취소합니다. 단, 결제 완료(`paid`) 상태인 주문만 취소할 수 있습니다.

- 파라미터:
  - `key`: `String` - 취소할 주문의 주문코드입니다.
- 반환 데이터:
  - 취소된 주문의 정보를 반환합니다. `order` 쿼리의 결과와 동일합니다.
- 예시 쿼리: 아래 2개의 코드 중 첫 번째는 실제 쿼리이고, 두 번째는 쿼리에서 쓰이는 `$key` variable을 정의한 것입니다. GraphQL Playground에서 테스트할 때는 쿼리를 입력하는 공간 아래에 *QUERY VARIABLES*에 입력하면 됩니다.(예시 데이터이므로 그대로 붙여넣어도 작동하지는 않습니다.)

```
mutation CancelOrder($key: ID!) {
  cancelOrder(key: $key) {
    key
    id
    products {
      quantity
      price
      shippingType
      itemKey
      productName
      itemOptionInfo {
        optionAttributes {
          name
          value
        }
```

```
      price
    }
    trackingNumber
    shippingCompanyCode
    shippingCompanyName
    shippedDate
    additionalAttributes {
      key
      value
    }
    taxFree
    sellerNote
  }
  status
  shippingInfo {
    sender {
      name
      phoneNumber
      email
    }
    recipient {
      name
      phoneNumber
      destinationAddress {
        addr1
        addr2
        postalCode
      }
    }
    shippingFee
  }
  createdAt
  updatedAt
  note
  ordererNote
  sellerNote
  isBeingMediated
  adjustments {
    reason
    price
    taxFree
  }
  transactions {
    key
    id
    kind
    status
    amount {
      currency
      value
    }
    createdAt
    updatedAt
    closedAt
```

```
      note
    }
  }
}
```

```
{
  "key": "2020000000000000000A"
}
```

- 예시 코드:
  - example1 - 주문 취소

requestOrderCancellation

주문 취소를 요청합니다. 배송 준비중(`preparing`) 상태의 주문만 취소 요청이 가능합니다.

- 파라미터:
  - `key`: `String` - 주문 취소를 요청할 주문의 주문코드입니다.
  - `input`: `object(RequestOrderCAncellationInput)` - 주문 취소에 필요한 입력 데이터입니다.
- 입력 데이터:
  - `cancelReason`: `String` - 취소 요청 사유입니다.
- 반환 데이터:
  - 취소가 요청된 주문 정보를 반환합니다. `order` 쿼리의 결과와 동일합니다.
- 예시 쿼리: 아래 2개의 코드 중 첫 번째는 실제 쿼리이고, 두 번째는 쿼리에서 쓰이는 `$key`, `$input` variable을 정의한 것입니다. GraphQL Playground에서 테스트할 때는 쿼리를 입력하는 공간 아래에 *QUERY VARIABLES*에 입력하면 됩니다.(예시 쿼리이므로 그대로 붙여넣어도 작동하지는 않습니다.)

```
mutation RequestOrderCancellation($key: ID!, $input:
RequestOrderCancellationInput!) {
  requestOrderCancallation(key: $key, input: $input) {
    key
    id
    products {
      quantity
      price
      shippingType
      itemKey
      productName
      itemOptionInfo {
        optionAttributes {
          name
          value
        }
        price
      }
      trackingNumber
      shippingCompanyCode
      shippingCompanyName
```

```
        shippedDate
        additionalAttributes {
          key
          value
        }
        taxFree
        sellerNote
      }
      status
      shippingInfo {
        sender {
          name
          phoneNumber
          email
        }
        recipient {
          name
          phoneNumber
          destinationAddress {
            addr1
            addr2
            postalCode
          }
        }
        shippingFee
      }
      createdAt
      updatedAt
      note
      ordererNote
      sellerNote
      isBeingMediated
      adjustments {
        reason
        price
        taxFree
      }
      transactions {
        key
        id
        kind
        status
        amount {
          currency
          value
        }
        createdAt
        updatedAt
        closedAt
        note
      }
    }
  }
```

```
{
  "key": "2020000000000000000A",
  "input": {
    "cancelReason": "주문 취소 요청 사유."
  }
}
```

- 예시 코드:
  - example1 - 주문 취소 요청

sellerQnaArticle

단일 1:1 문의 게시판 글을 조회합니다.

- 파라미터:
  - **key**: `String` - 정보를 조회할 1:1 문의 게시판 글의 key입니다.
- 반환 데이터: 1:1 문의 게시판 글의 정보를 반환하며, 다음 내용을 액세스 할 수 있습니다.
  - `key`: `String` - 1:1 문의 게시판 글의 key입니다.
  - `id`: `String` - 1:1 문의 게시판 글의 API ID입니다.
  - `type`: `SellerQnaType` - 1:1 문의 게시판 글의 타입입니다. 가능한 값들은 *데이터 타입 - SellerQnaType*을 참고해주시면 됩니다.
  - `isSecret`: `Boolean` - 비밀글 여부입니다.
  - `title`: `String` - 글 제목입니다.
  - `content`: `String` - 글 내용입니다.
  - `files`: `[String]` - 첨부파일의 URL 리스트입니다.
  - `relatedItemKey`: `String` - 연관 상품이 있는 경우 상품코드입니다.
  - `relatedOrderKey`: `String` - 연관 주문이 있는 경우 주문코드입니다.
  - `createdAt`: `Timestamp` - 등록 시각입니다.
  - `recipientName`: `String` - 연관 주문이 있는 경우 주문의 주문자 이름입니다.
  - `comments`: `[String]` - 달린 댓글이 있는 경우 댓글 목록입니다. 댓글 작성 일시가 최근일수록 리스트의 앞에 위치합니다.
  - `subArticles`: `[SellerQnaArticle]` - 하위 글 목록입니다.
- 예시 쿼리: 아래 2개의 코드 중 첫 번째는 실제 쿼리이고, 두 번째는 쿼리에서 쓰이는 `$key` variable을 정의한 것입니다. GraphQL Playground에서 테스트할 때는 쿼리를 입력하는 공간 아래에 *QUERY VARIABLES*에 입력하면 됩니다.

```
# 하단에 주어진 key는 실제로 동작하는 key가 아닌, 예시로 주어진 key입니다.
query SellerQnaArticle($key: ID!) {
  sellerQnaArticle(key: $key) {
    key
    id
    type
    isSecret
    title
    content
    files
```

```
      relatedItemKey
      relatedOrderKey
      recipientName
      createdAt
      comments
      subArticles {
        key
        id
        type
        isSecret
        title
        content
        files
        relatedItemKey
        relatedOrderKey
        recipientName
        createdAt
        comments
      }
    }
  }
```

```
{
  "key": "1000000"
}
```

- 예시 코드:
  - example1 - 단일 1:1 문의 게시판 글 조회

`allSellerQnaArticles`

복수의 1:1 문의 게시판 글 목록을 조회합니다. 검색 조건을 설정한 경우 검색 조건에 부합하는 글만 조회합니다.

- 파라미터:
  - Pagination 관련 파라미터:
    - `after`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이후의 글만을 불러옵니다.
    - `before`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이전의 글만을 불러옵니다.
    - `first`: `Int` - 조건을 만족하는 모든 내역들 중 처음 몇 개의 글을 불러올 지를 나타냅니다.
    - `last`: `Int` - 조건을 만족하는 모든 내역들 중 마지막 몇 개의 글을 불러올 지를 나타냅니다.
  - Search 관련 파라미터:
    - `search`: `[SellerQnaSearch]`: 텍스트 검색을 지원합니다.
      - `value`: `String` - 검색할 값입니다.
      - `by`: `SellerQnaSearchBy`: 검색할 필드입니다. 가능한 필드는 `title`, `content`, `orderKey`, `itemKey`입니다.

        - `type`: `SellerQnaType`: 주어진 타입인 문의 글만을 불러옵니다. 가능한 값은 *데이터 타입 - SellerQnaType* 항목을 참고해주시면 됩니다.
        - `receiverName`: `String` - 주문자 이름을 검색합니다.
        - `dateFrom`: `Timestamp` - 글이 등록된 일자가 이 값 이후인 것들만을 불러옵니다. Unix timestamp이며, 생략하면 7일 전으로 설정됩니다. 최대 30일 간격으로 조회가 가능합니다.
        - `dateTo`: `Timestamp` - 글이 등록된 일자가 이 값 이전인 것들만을 불러옵니다. Unix timestamp이며, 생략하면 현재로 설정됩니다. 최대 30일 간격으로 조회가 가능합니다.
- 반환 값: `allSellerQnaArticles`는 pagination을 위한 정보와 내용에 대한 정보를 동시에 담고 있습니다.
    - `pageInfo`: pagination을 위한 정보입니다.
        - `hasNextPage`: 데이터가 뒤에 더 있는지를 나타냅니다.
        - `hasPreviousPage`: 데이터가 앞에 더 있는지를 나타냅니다.
        - `startCursor`: 현재 페이지의 첫 데이터에 대한 cursor 값입니다. 이 값을 `before` 파라미터에 넘겨주면 정확하게 이전 페이지의 데이터를 가져올 수 있습니다.
        - `endCursor`: 현재 페이지의 마지막 데이터에 대한 cursor 값입니다. 이 값을 `after` 파라미터에 넘겨주면 정확하게 다음 페이지의 데이터를 가져올 수 있습니다.
    - `edges`: 1:1 문의 글 데이터입니다.
        - `cursor`: 해당 글의 pagination cursor입니다.
        - `node`: 해당 글의 내용입니다. 타입은 `sellerQnaArticle` 쿼리의 반환 결과와 동일합니다.
- 예시 쿼리:

```
# 기본 검색. 7일전 내역까지 최대 100건 조회.
query AllSellerQnaArticles {
  allSellerQnaArticles() {
    pageInfo {
      hasNextPage
      hasPreviousPage
      startCursor
      endCursor
    }
    edges {
      cursor
      node {
        key
        id
        type
        isSecret
        title
        content
        files
        relatedItemKey
        relatedOrderKey
        recipientName
        createdAt
        comments
        subArticles {
          key
          id
          type
          isSecret
```

```
            title
            content
            files
            relatedItemKey
            relatedOrderKey
            recipientName
            createdAt
            comments
          }
        }
      }
    }
  }
```

- 예시 코드:
  - example1 - 기본 조건 조회
  - example2 - 14일간 내역 조회
  - example3 - 제목 검색
  - example4 - 수령인 이름 검색
  - example5 - 문의글 타입 검색

emergencyMessage

긴급메시지를 조회합니다.

- 파라미터:
  - **key**: String - 정보를 조회할 긴급메시지의 key입니다.
- 반환 데이터: 긴급메시지의 정보를 반환하며, 다음 내용을 액세스 할 수 있습니다.
  - key: String - 긴급메시지의 key입니다.
  - id: String - 긴급메시지의 API ID입니다.
  - createdAt: Timestamp - 긴급메시지 등록 시각입니다.
  - type: EmergencyMessageType - 긴급메시지 타입입니다. 가능한 값은 *데이터 타입 - EmergencyMessageType*을 참고해주시면 됩니다.
  - itemKey: String - 관련된 상품이 있는 경우 상품의 KEY입니다.
  - content: String - 긴급메시지의 내용입니다.
  - url: String - 관련 URL이 있는 경우 해당 URL입니다.
  - penalty: Int - 페널티 점수가 있는 경우 해당 점수입니다.
  - status: EmergencyMessageStatus - 긴급메시지 상태입니다. 가능한 값은 *데이터 타입 - EmergencyMessageStatus*를 참고해주시면 됩니다.
  - repliedAt: Timestamp - 답변 시각입니다. Unix timestamp로 주어집니다.
  - reply: String - 답변 내용입니다.
- 예시 쿼리: 아래 2개의 코드 중 첫 번째는 실제 쿼리이고, 두 번째는 쿼리에서 쓰이는 $key variable을 정의한 것입니다. GraphQL Playground에서 테스트할 때는 쿼리를 입력하는 공간 아래에 *QUERY VARIABLES*에 입력하면 됩니다.

```
# 하단에 주어진 key는 실제로 동작하는 key가 아닌, 예시로 주어진 key입니다.
query EmergencyMessage($key: ID!) {
  emergencyMessage(key: $key) {
```

```
      key
      id
      createdAt
      type
      itemKey
      content
      url
      penalty
      status
      repliedAt
      reply
    }
  }
```

```
{
  "key": "12345"
}
```

- 예시 코드:
  - example1 - 단일 긴급메시지 조회

`allEmergencyMessages`

복수의 긴급메시지 목록을 조회합니다. 검색 조건을 설정한 경우 검색 조건에 부합하는 긴급메시지만 조회합니다.

- 파라미터:
  - Pagination 관련 파라미터:
    - `after`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이후의 긴급메시지만을 불러옵니다.
    - `before`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이전의 긴급메시지만을 불러옵니다.
    - `first`: `Int` - 조건을 만족하는 모든 긴급메시지 중 처음 몇 개의 긴급메시지를 불러올 지를 나타냅니다.
    - `last`: `Int` - 조건을 만족하는 모든 긴급메시지 중 마지막 몇 개의 긴급메시지를 불러올 지를 나타냅니다.
  - Search 관련 파라미터:
    - `status`: `EmergencyMessageStatus` - 긴급메시지 상태입니다. 주어진 상태인 긴급메시지만을 불러오며, 명시되지 않은 경우 전체입니다.
- 반환 값: `allEmergencyMessages`는 pagination을 위한 정보와 내용에 대한 정보를 동시에 담고 있습니다.
  - `pageInfo`: pagination을 위한 정보입니다.
    - `hasNextPage`: 데이터가 뒤에 더 있는지를 나타냅니다.
    - `hasPreviousPage`: 데이터가 앞에 더 있는지를 나타냅니다.
    - `startCursor`: 현재 페이지의 첫 데이터에 대한 cursor 값입니다. 이 값을 `before` 파라미터에 넘겨주면 정확하게 이전 페이지의 데이터를 가져올 수 있습니다.

        - `endCursor`: 현재 페이지의 마지막 데이터에 대한 cursor 값입니다. 이 값을 `after` 파라미터에 넘겨주면 정확하게 다음 페이지의 데이터를 가져올 수 있습니다.
    - `edges`: 긴급메시지 데이터입니다.
        - `cursor`: 해당 긴급메시지의 pagination cursor입니다.
        - `node`: 해당 긴급메시지의 내용입니다. 타입은 `emergencyMessage` 쿼리의 반환 결과와 동일합니다.
- 예시 쿼리:

```
# 기본 검색. 7일전 내역까지 최대 100건 조회.
query AllEmergencyMessages {
  allEmergencyMessages() {
    pageInfo {
      hasNextPage
      hasPreviousPage
      startCursor
      endCursor
    }
    edges {
      cursor
      node {
        key
        id
        createdAt
        type
        itemKey
        content
        url
        penalty
        status
        repliedAt
        reply
      }
    }
  }
}
```

- 예시 코드:
    - example1 - 기본 조건 조회
    - example2 - 상태 조건 검색

notice

알림메모를 조회합니다.

- 파라미터:
    - **key**: `String` - 정보를 조회할 알림메모의 key입니다.
- 반환 데이터: 알림메모의 정보를 반환하며, 다음 내용을 액세스 할 수 있습니다.
    - `key`: `String` - 알림메모의 key입니다.
    - `id`: `String` - 알림메모의 API ID입니다.

    - `createdAt`: `Timestamp` - 알림메모의 등록 시각입니다.
    - `type`: `NoticeType` - 알림메모 타입입니다.
    - `content`: `String` - 알림메모 내용입니다.
    - `relatedItemKeys`: `[String]` - 관련된 상품이 있는 경우 상품코드 리스트입니다.
    - `relatedOrderKeys`: `[String]` - 관련된 주문이 있는 경우 주문코드 리스트입니다.
    - `checkedAt`: `Timestamp` - 확인한 알림메모의 경우 확인한 시각입니다.
- 예시 쿼리: 아래 2개의 코드 중 첫 번째는 실제 쿼리이고, 두 번째는 쿼리에서 쓰이는 `$key` variable을 정의한 것입니다. GraphQL Playground에서 테스트할 때는 쿼리를 입력하는 공간 아래에 *QUERY VARIABLES*에 입력하면 됩니다.

```
# 하단에 주어진 key는 실제로 동작하는 key가 아닌, 예시로 주어진 key입니다.
query Notice($key: ID!) {
  notice(key: $key) {
    key
    id
    createdAt
    type
    content
    authorType
    relatedItemKeys
    relatedOrderKeys
    checkedAt
  }
}
```

```
{
  "key": "54321"
}
```

- 예시 코드:
    - example1 - 단일 알림메모 조회

`allNotices`

복수의 알림메모를 조회합니다. 검색 조건을 설정한 경우 검색 조건에 부합하는 알림메모만 조회합니다.

- 파라미터:
    - Pagination 관련 파라미터:
        - `after`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이후의 알림메모만을 불러옵니다.
        - `before`: `String` - API 고유 cursor 값으로, 지정된 경우 이 cursor 값 이전의 알림메모만을 불러옵니다.
        - `first`: `Int` - 조건을 만족하는 모든 알림메모 중 처음 몇 개의 알림메모를 불러올 지를 나타냅니다.
        - `last`: `Int` - 조건을 만족하는 모든 알림메모 중 마지막 몇 개의 알림메모를 불러올 지를 나타냅니다.

    - Search 관련 파라미터:
        - `type`: `NoticeType` - 알림메모 상태입니다. 주어진 상태인 알림메모만을 불러오며, 명시되지 않은 경우 전체입니다.
        - `checked`: `Boolean` - 알림메모의 확인 여부입니다.
        - `dateFrom`: `Timestamp` - 알림메모가 등록된 일자가 이 값 이후인 것들만을 불러옵니다. Unix timestamp이며, 생략하면 7일 전으로 설정됩니다.
        - `dateTo`: `Timestamp` - 알림메모가 등록된 일자가 이 값 이전인 것들만을 불러옵니다. Unix timestamp이며, 생략하면 현재로 설정됩니다.
- 반환 값: `allNotices`는 pagination을 위한 정보와 내용에 대한 정보를 동시에 담고 있습니다.
    - `pageInfo`: pagination을 위한 정보입니다.
        - `hasNextPage`: 데이터가 뒤에 더 있는지를 나타냅니다.
        - `hasPreviousPage`: 데이터가 앞에 더 있는지를 나타냅니다.
        - `startCursor`: 현재 페이지의 첫 데이터에 대한 cursor 값입니다. 이 값을 `before` 파라미터에 넘겨주면 정확하게 이전 페이지의 데이터를 가져올 수 있습니다.
        - `endCursor`: 현재 페이지의 마지막 데이터에 대한 cursor 값입니다. 이 값을 `after` 파라미터에 넘겨주면 정확하게 다음 페이지의 데이터를 가져올 수 있습니다.
    - `edges`: 알림메모 데이터입니다.
        - `cursor`: 해당 알림메모의 pagination cursor입니다.
        - `node`: 해당 알림메모의 내용입니다. 타입은 `notice` 쿼리의 반환 결과와 동일합니다.
- 예시 쿼리:

```
# 기본 검색. 7일전 내역까지 최대 100건 조회.
query AllNotices {
  allNotices() {
    pageInfo {
      hasNextPage
      hasPreviousPage
      startCursor
      endCursor
    }
    edges {
      cursor
      node {
        key
        id
        createdAt
        type
        content
        authorType
        relatedItemKeys
        relatedOrderKeys
        checkedAt
      }
    }
  }
}
```

- 예시 코드:

    - example1 - 기본 조건 조회
    - example2 - 30일간 내역 조회
    - example3 - 30일간 미확인한 알림메모 검색
    - example4 - 알림메모 타입 검색

createSellerQnaArticle

판매사 1:1 문의 게시판에 글을 작성합니다.

- 파라미터:
    - **input**: object(SellerQnaArticleInput) - 입력할 데이터입니다.
- 입력 데이터 (굵은 글씨는 필수 필드입니다):
    - **type**: SellerQnaType - 문의글 타입입니다. 가능한 값은 *데이터 타입 - SellerQnaType*를 참고해주시면 됩니다.
    - **title**: String - 글 제목입니다.
    - **content**: String - 문의 내용입니다.
    - files: [Upload] - 첨부파일의 리스트입니다. 사용 방법은 아래 예제(example2 - 파일 업로드)를 참고해주시면 됩니다.
    - relatedItemKey: String - 연관 상품이 있는 경우 상품의 key입니다.
    - relatedOrderKey: String - 연관 주문이 있는 경우 주문의 key입니다.
- 반환 데이터:
    - 새로 생성된 문의글을 반환합니다.
- 예시 쿼리:

```
mutation CreateSellerQnaArticle($input: SellerQnaArticleInput!) {
  createSellerQnaArticle(input: $input) {
    id
    key
    type
    isSecret
    title
    content
    files
    relatedItemKey
    relatedOrderKey
    createdAt
    recipientName
    subArticles {
      id
      key
      type
      isSecret
      title
      content
      files
      relatedItemKey
      relatedOrderKey
      createdAt
      recipientName
      comments
```

```
    }
    parentArticles {
      id
      key
      type
      isSecret
      title
      content
      files
      relatedItemKey
      relatedOrderKey
      createdAt
      recipientName
      comments
    }
    comments
  }
}
```

```
{
  "input": {
      "type": "item",
      "title":"제목",
      "files": [null],
      "content":"문의 내용",
      "relatedItemKey": "W999999"
  }
}
```

- 예시 코드:
  - example1 - 판매사 1:1 문의 작성
  - example2 - 파일 업로드(nodeJS)

`requestRefundOrExchange`

주문한 상품을 반품/교환 신청합니다.

- 파라미터:
  - **`key`**: `String` - 반품/교환을 요청할 주문의 KEY 입니다.
  - **`input`**: `object(RefundExchangeOrderInput)` - 입력할 데이터입니다.
- 입력 데이터:
  - **`productStatus`**: `ProductStatusType` - 상품의 상태입니다. *데이터 타입 - ProductStatus*를 참고해주시면 됩니다.
  - **`reason`**: `refundOrExchangeReasonType` - 반품 사유입니다. *데이터 타입 - refundOrExchangeReasonType*를 참고해주시면 됩니다.

    - **detailedReason**: DetailedReasonInput - 반품 상세 사유입니다. *데이터 타입 - DetailedReasonInput*를 참고해주시면 됩니다.
    - **shippingfeePayment**: ShippingFeePaymentType - 반품 배송비 부과 타입입니다. *데이터 타입 - ShippingFeePaymentType*를 참고해주시면 됩니다.
    - **refundAddress**: RefundAddressInput - 반품 주소입니다. *데이터 타입 - RefundAddressInput*를 참고해주시면 됩니다.
    - refundTrackingInfo: RefundTrackingInfoInput - 반품 택배 정보입니다. *데이터 타입 - RefundTrackingInfoInput*를 참고해주시면 됩니다.
    - content: String - 공급사에 전달할 내용입니다.
    - files : [Upload] - 첨부파일의 리스트입니다.

- 반환 데이터:
    - 업데이트 된 주문 정보를 반환합니다. order 쿼리의 결과와 동일합니다.

- 예시 쿼리:

```
mutation RequestRefundOrExchange($key: ID!, $input: RefundExchangeOrderInput!) {
  requestRefundOrExchange(key: $key, input: $input) {
    key
    id
    products {
      quantity
      price
      shippingType
      itemKey
      productName
      itemOptionInfo {
        optionAttributes {
          name
          value
        }
        price
      }
      trackingNumber
      shippingCompanyCode
      shippingCompanyName
      shippedDate
      additionalAttributes {
        key
        value
      }
      taxFree
    }
    status
    shippingInfo {
      sender {
        name
        phoneNumber
        email
      }
```

```
    recipient {
      name
      phoneNumber
      destinationAddress {
        addr1
        addr2
        postalCode
      }
    }
    shippingFee
  }
  createdAt
  updatedAt
  note
  ordererNote
  sellerNote
  isBeingMediated
  adjustments {
    reason
    price
    taxFree
  }
  transactions {
    key
    id
    kind
    status
    amount {
      currency
      value
    }
    createdAt
    updatedAt
    closedAt
    note
  }
  refundDetails {
    status
    productStatus
    reason
    detailedReason {
      refundType
      detail
      explanation
    }
    shippingFeePayment
    returnCriteria
    refundAddress
    shippingCompany
    refundTrackingInfo {
      shippingCompanycode
      trackingNumber
    }
    content
```

```
    }
  }
}
```

```
{
  "key": "2020000000000000000A",
  "input": {
    "productStatus": "Unopened",
    "reason": "DamagedItem",
    "detailedReason": {
      "refundType": "Reship",
      "detail": "상세 내용.",
      "explanation": "기타 정보."
    },
    "shippingFeePayment": "CollectedOnDelivery",
    "refundAddress": {
      "refundAddressType": "ManuallyProvided",
      "address": "서울특별시"
    },
    "refundTrackingInfo": {
      "shippingCompanyCode": "1",
      "trackingNumber": "12345678910"
    },
    "content": "공급사에 전달할 내용.",
  }
}
```

- 예시 코드:
  - example1 - 판매사 반품/교환 요청 작성

데이터 타입

`ShippingType`

배송비 유형은 3가지로 구분됩니다. API에서는 아래 3가지 값들 중 하나를 반환하게 됩니다.

- `inAdvance`: 선불 전용 상품
- `uponArrival`: 착불 전용 상품
- `free`: 무료 배송 상품

`ItemStatus`

상품의 현재 상태를 나타내는 방식입니다.

- `soldout`: 일시 품절 상태입니다.
- `available`: 판매 가능 상태입니다.
- `unavailable`: 판매 불가인 상품입니다. 단, 재입고되거나 재판매 될 수 있습니다.
- `discontinued`: 단종된 상품이고, 재입고 되지 않는 상품입니다.

ItemSortCriteria

상품 정렬 기준입니다.

- dateDesc: 상품 수정일 기준 내림차순 정렬입니다.
- dateAsc: 상품 수정일 기준 오름차순 정렬입니다.
- default: key 기준 기본 정렬입니다.
- nameAsc: 상품명 기준 오름차순 정렬입니다.
- nameDesc: 상품명 기준 내림차순 정렬입니다.
- priceAsc: 가격 기준 오름차순 정렬입니다.
- priceDesc: 가격 기준 내림차순 정렬입니다.
- registerDateAsc: 상품 등록일 기준 오름차순 정렬입니다.
- registerDateDesc: 상품 등록일 기준 내림차순 정렬입니다.
- keyAsc: key 기준 오름차순 정렬입니다.
- keyDesc: key 기준 내림차순 정렬입니다.

ShippingLocationType

상품 배송지 타입입니다.

- domestic: 국내 배송 상품입니다.
- overseas: 해외 배송 상품입니다.

ImageSize

상품 이미지 사이즈입니다. auto, xlarge, large, medium, small의 3가지 사이즈를 제공합니다.

- auto - 원본 이미지 사이즈를 기반으로 최적화된 사이즈의 이미지를 제공합니다.
- xlarge - 가로 1000px, 세로 1000px의 이미지.
- large - 가로 640px, 세로 640px의 이미지.
- medium - 가로 300px, 세로 300px의 이미지.
- small - 가로 100px, 세로 100px의 이미지.

Currency

화폐 단위 타입입니다. ISO 4217에 따른 값을 지원하며, 현재 지원하는 값의 목록은 다음과 같습니다.

- KRW: 한국 원화 단위입니다.
- USD: 미국 달러화 단위입니다.
- CNY: 중국 위안화 단위입니다.

Language

언어 타입입니다. ISO 639-1과 ISO 3166-1 alpha-2를 조합한 형태의 값을 지원하며, 현재 지원하는 값의 목록은 다음과 같습니다.

- ko_KR: 대한민국 - 한국어
- en_US: 미국 - 영어
- zh_CN: 중국 - 중국어

PricePolicy

가격 정책 타입입니다.

- `free`: 가격 자율 제품입니다.
- `fixed`: 소비자 준수 가격이 있는 제품입니다. 반드시 `Item.fixedPrice` 필드를 확인해주시기 바랍니다.

ItemOption

상품 옵션 정보입니다. 오너클랜 API에서는 하나의 옵션 정보를 SKU(Stock Keeping Unit, 재고 관리 단위) 혹은 SKU의 모음으로 가정합니다. 따라서 오너클랜 웹사이트에서 옵션이 없는 것으로 조회되는 상품이라 하더라도, SKU가 있어야하므로 `Item.options` 필드에 값이 1개 제공되며, 옵션이 1개 있는 것으로 조회되는 상품도 `Item.options` 필드에 값이 1개 제공됩니다. 단, 옵션이 없는 것으로 조회되는 경우에는, `Item.options.optionAttributes`에 값이 없는 빈 배열이 반환되고, 옵션이 1개 있는 것으로 조회되는 상품은 `Item.options.optionAttributes`에 값이 있는, 원소 1개인 배열이 반환됩니다. 이를 통해 옵션이 없는 상품과 있는 상품을 구분할 수 있습니다.

- `optionAttributes`: `[ItemOptionAttribute]` - 옵션 속성에 대한 이름 - 값 배열입니다.
- `price`: `Float` - 옵션 가격입니다. `currency` 파라미터를 가질 수 있으며, 현재 지원하는 값은 `KRW`입니다.
- `quantity`: `Int`

ItemOptionAttribute

상품 옵션에 대한 속성 정보입니다.

- `name`: `String` - 옵션 속성 이름입니다.
- `value`: `String` - 옵션 속성 값입니다. 만약 값이 `null`이라면 해당 옵션 속성은 입력형 옵션 속성이라는 의미입니다.

Category

오너클랜 카테고리 정보입니다. 판매사 API에서는 하나의 카테고리에 대해 오너클랜 카테고리 코드와 이름만 조회가 가능합니다. 다른 항목은 조회할 경우 데이터를 반환하지 않습니다.

- `name`: `String` - 카테고리 이름입니다.
- `fullName`: `String` - 카테고리 전체 이름입니다.
- `key`: `String` - 카테고리 코드입니다.

Price

가격 정보를 위한 형식입니다. `KRW` 외에 `USD`, `CNY`를 지원합니다. `KRW` 이외의 화폐에 대해서는 현재 오너클랜 API에서 지원하지 않습니다.

- 파라미터
  - `currency`: `Currency` - 가격 정보의 화폐 단위를 지정할 수 있습니다. 기본값은 `KRW`와 같습니다.
- 반환값
  - `Int` | `Float`
- 예시 쿼리 1

```
# 실제로 테스트 할 수 있는 쿼리는 아닙니다.
query {
    test {
        price
    }
}
```

```
{
    "data": {
        "test": {
            "price": 10000
        }
    }
}
```

- 예시 쿼리 2

```
# 실제로 테스트 할 수 있는 쿼리는 아닙니다.
query {
    test {
        price_KRW: price(currency: KRW)
        price_USD: price(currency: USD)
        price_CNY: price(currency: CNY)
    }
}
```

```
{
    "data": {
        "test": {
            "price_KRW": 10000,
            "price_USD": 10.00,
            "price_CNY": 55.56
        }
    }
}
```

ReturnCriteria

상품의 반품시 반품접수 유형입니다.

- `vendor`: **공급사 반품접수** 유형으로, 공급사가 직접 택배사로 반품 택배를 접수합니다.
- `seller`: **원운송장 반품접수** 유형으로, 판매사가 원운송장에 기재되어 있는 주소로 반품 택배를 접수합니다.

ItemHistoryKind

상품 변경 이력의 유형입니다.

- `soldout`: **품절** 유형입니다.
- `discontinued`: **단종** 유형입니다.
- `optionSoldout`: **옵션 품절** 유형입니다.
- `optionDiscontinued`: **옵션 단종** 유형입니다.
- `restocked`: **재입고** 유형입니다.
- `priceIncreased`: **판매가 인상** 유형입니다.
- `priceDecreased`: **판매가 인하** 유형입니다.
- `categoryChanged`: **카테고리 변경** 유형입니다.
- `optionRestocked`: **옵션 재입고** 유형입니다.
- `optionPriceIncreased`: **옵션가 인상** 유형입니다.
- `optionPriceDecreased`: **옵션가 인하** 유형입니다.
- `shippingFeeIncreased`: **배송비 인상** 유형입니다.
- `shippingFeeDecreased`: **배송비 인하** 유형입니다.
- `returnShippingFeeIncreased`: **반품/교환비(편도) 인상** 유형입니다.
- `returnShippingFeeDecreased`: **반품/교환비(편도) 인하** 유형입니다.
- `shippingTypeChanged`: **배송방식 변경** 유형입니다.
- `optionChanged`: **옵션구성 변경** 유형입니다.
- `etc`: **기타** 유형입니다.
- `prohibited`: **유통금지** 유형입니다.
- `returnCriteriaChanged`: **반품접수유형 변경** 유형입니다.
- `stockChanged`: **재고 변경** 유형입니다.
- `optionStockChanged`: **옵션수량변경** 유형입니다.
- `fixedPriceIncreased`: **소비자가 인상** 유형입니다.
- `fixedPriceDecreased`: **소비자가 인하** 유형입니다.
- `modelChanged`: **모델명 변경** 유형입니다.
- `returnableChanged`: **반품가능여부 변경** 유형입니다.
- `contentChanged`: **상품 상세정보 변경** 유형입니다.
- `sellerOnlyContentChanged`: **상품 상단문구 변경** 유형입니다.
- `boxQuantityChanged`: **묶음배송수량 변경** 유형입니다.
- `adultOnlyChanged`: **미성년자 판매변경** 유형입니다.
- `noReturnReasonChanged`: **반품불가사유 변경** 유형입니다.
- `imageChanged`: **목록이미지 변경** 유형입니다.
- `nameChanged`: **상품명 변경** 유형입니다.
- `originChanged`: **원산지 변경** 유형입니다.
- `certificateInformationChanged`: **인증정보 고시 변경** 유형입니다.
- `pricePolicyChanged`: **판매유형 변경** 유형입니다.
- `productNotificationInformationChanged`: **상품정보제공 고시 변경** 유형입니다.
- `productionChanged`: **제조사 변경** 유형입니다.
- `searchKeywordsChanged`: **키워드 변경** 유형입니다.
- `taxFreeChanged`: **과세/면세 변경** 유형입니다.
- `additionalDocumentsChanged`: **추가서류 변경** 유형입니다.

OrderStatus

주문의 상태를 나타냅니다.

- `placed`: **미처리** 상태입니다.
- `paid`: **입금완료** 상태입니다.
- `preparing`: **발송 준비** 상태입니다.
- `cancelRequested`: **취소 요청** 상태입니다.
- `cancelled`: **주문 취소** 상태입니다.
- `shipped`: **발송 완료** 상태입니다.
- `exchangeRequested`: **교환 요청** 상태입니다.
- `exchanged`: **교환 완료** 상태입니다.
- `refundRequested`: **반품 요청** 상태입니다.
- `refundAccepted`: **반품 접수** 상태입니다.
- `refundShipped`: **반송 물품 발송 완료** 상태입니다.
- `refunded`: **반품 완료** 상태입니다.
- `refundClosed`: **반품 완료 후 PG 정산 완료** 상태입니다.

Address

주소 정보입니다. 기본주소와 상세주소, 우편번호 필드로 구성되어 있습니다.

- `addr1`: `String` - 기본주소입니다.
- `addr2`: `String` - 상세주소입니다.
- `postalCode`: `String` - 우편번호입니다.

Adjustment

주문 내역에 대해 조정이 발생한 경우, 조정 내역에 대한 정보입니다.

- `reason`: `String` - 조정이 발생한 사유입니다.
- `price`: `String` - 조정 대상 금액입니다.
- `taxFree`: `Boolean` - 조정 대상 금액에 대한 면세 여부입니다.

Transaction

거래 내역 정보입니다. 현재는 적립금 사용에 대해서만 지원하고 있습니다.

- `key`: `String` - 거래 내역 정보의 key값입니다.
- `id`: `String` - 거래 내역 정보의 API ID 값입니다.
- `kind`: `TransactionKind` - 거래 내역 타입입니다. 현재는 적립금 사용에 대해서만 지원하므로 `reserveTransfer`로 값이 고정되어 있습니다.
- `status`: `TransactionStatus` - 거래 내역의 상태입니다. 자세한 내용은 `TransactionStatus` 항목을 참고해주세요.
- `amount`: `Price` - 거래 내역에 해당하는 금액입니다. `currency`와 `value`로 구성되어 있으며, 현재는 적립금 사용에 대해서만 지원하므로 `currency`는 `RSV`로 고정되어 있습니다.
- `createdAt`: `Int` - 거래 내역이 생성된 시각입니다. Unix timestamp로 주어집니다.
- `updatedAt`: `Int` - 거래 내역의 상태가 최종적으로 변경된 시각입니다. Unix timestamp로 주어집니다.
- `closedAt`: `Int` - 거래 내역이 마감된 시각입니다. Unix timestamp로 주어집니다. 마감되지 않은 내역의 경우 `null`이 반환됩니다.
- `note`: `String` - 거래 내역에 대한 상세 정보입니다.

RefundedOrder

반품 요청 내역들입니다.

- `status`: `RefundProgressStatusType` - 진행 상태입니다.
- `productStatus`: `ProductStatusType` - 상품 상태입니다.
- `reason`: `RefundOrExchangeReasonType` - 반품 구분 타입입니다.
- `detailedReason`: `Object` - 반품 사유입니다. 하위 필드로 조회할 수 있는 내역은 아래와 같습니다.
    - `refundType`: `RefundType` - 반품 타입입니다.
    - `detail`: `String` - 상세 내용입니다.
    - `explanation`: `String` - 기타 정보입니다.
- `shippingFeePayment`: `String` - 반품 배송비입니다.
- `returnCriteria`: `ReturnCriteria` - 반품 접수 유형입니다.
- `refundAddress`: `Object` - 반품 주소 정보입니다. 하위 필드로 조회할 수 있는 내역은 아래와 같습니다.
    - `refundAddressType`: `RefundAddressType` - 반품 주소 타입입니다.
    - `address`: `String` - 반품 주소입니다.
- `shippingCompany`: `String` - 반품 택배 정보입니다.
- `refundTrackingInfo`: `Object` - 반품 택배 정보입니다. 하위 필드로 조회할 수 있는 내역은 아래와 같습니다.
    - `shippingCompanycode`: `String` - 택배사코드 입니다.
    - `trackingNumber`: `String` - 송장번호 입니다.
- `content`: `String`: 공급사에 전달할 메시지 내용입니다.

TransactionStatus

거래 내역의 상태입니다.

- `placed`: **처리 대기중** 상태입니다.
- `ongoing`: **처리중** 상태입니다.
- `closed`: **처리 완료** 상태입니다.

SenderInput

주문 생성시에 사용하는 보내는 이에 대한 정보입니다. 굵은 글씨로 표시된 필드는 필수 값입니다.

- **`name`**: `String` - 보내는 이의 이름입니다.
- **`phoneNumber`**: `String` - 보내는 이의 연락처입니다.
- `email`: `String` - 보내는 이의 이메일 주소입니다.

RecipientInput

주문 생성시에 사용하는 받는 이에 대한 정보입니다. 굵은 글씨로 표시된 필드는 필수 값입니다.

- **`name`**: `String` - 받는 이의 이름입니다.
- **`phoneNumber`**: `String` - 받는 이의 연락처입니다.
- **`destinationAddress`**: `Object` - 받는 이의 주소입니다.
    - **`addr1`**: `String` - 기본 주소입니다.
    - **`addr2`**: `String` - 상세 주소입니다.
    - **`postalCode`**: `String` - 우편번호입니다.
    - 참고사항: 현재 오너클랜에 데이터를 저장하는 방식의 한계상, `addr1` 필드와 `addr2`, `postalCode` 필드에 값을 나눠서 넣어도 생성된 주문을 조회하게 되면 `addr1` 필드에 값이 합쳐져서 조회되고,

`addr2`, `postalCode` 필드는 `null`로 조회됩니다.

OrderProductInput

주문 생성시에 사용하는 주문 제품 각각에 대한 정보입니다. 굵은 글씨로 표시된 필드는 필수 값입니다.

- **`quantity`**: `Int` - 주문 수량입니다.
- **`itemKey`**: `String` - 주문할 상품의 상품코드입니다.
- **`optionAttributes`**: `[String]` - 주문할 상품에 옵션이 있는 경우, 선택할 옵션의 값들입니다. 옵션이 여러 개인 경우 순서대로 입력해야 합니다.
- `ordererNote`: `String` - 각 주문 제품에 대한 배송 요청사항입니다. 있는 경우에만 입력하면 됩니다.

CreateOrderSimulationResult & CreateOrderSimulationResultInput

주문 생성 시뮬레이션 시에 반환되는 데이터 형식과, 주문 생성시에 생성 예정 내역과 동일한지 검증할 시뮬레이션 결과 정보입니다. 굵은 글씨로 표시된 필드는 필수 값입니다.

- **`itemAmounts`**: `[object]` - 주문할 상품에 대한 결과입니다.
  - **`itemKey`**: `String` - 상품의 오너클랜 상품코드입니다.
  - **`amount`**: `Int` - 상품 개수까지 반영한 최종 상품 금액입니다.
- **`shippingAmount`**: `Int` - 추가 배송비까지 고려된 최종 배송비입니다.
- **`extraShippingFeeExists`**: `Boolean` - 추가 배송비가 있는지의 여부입니다.

`simulateCreateOrder` 쿼리를 실행하면 입력 데이터에 대해 생성이 예상되는 주문 개수만큼 `CreateOrderSimulationResult` object를 갖는 배열이 반환되고, `createOrder`에 사용할 때도 역시 생성이 예상되는 주문 개수만큼 동일하게 입력해야 합니다. 자세한 예시 코드는 `simulateCreateOrder`와 `createOrder`의 예시 코드를 참조하면 됩니다.

SellerQnaType

판매사 1:1 문의 글 타입입니다.

- **`item`**: 상품 관련 문의글입니다.
- **`requestOrderCancellation`**: 주문 취소 관련 문의글입니다.
- **`returnOrRefund`**: 반품/교환 관련 문의글입니다.
- **`shippingScheduleOrStock`**: 출고 일정 또는 재고 문의글입니다.
- **`bulkStock`**: 대량 재고 문의글입니다.
- **`reportOrCertificate`**: 신고, 제재 또는 인증 관련 문의글입니다.
- **`tax`**: 회계, 세금계산서 또는 사업자정보 관련 문의글입니다.
- **`dafalza`**: 다팔자 관련 문의글입니다.
- **`system`**: 오너클랜 시스템 관련 문의글입니다.
- **`etc`**: 기타 문의글입니다.

EmergencyMessageType

긴급메시지 타입입니다.

- **`general`**: 일반 타입입니다.
- **`pricePolicyCompliance`**: 가격준수요청 타입입니다.
- **`saleStop`**: 판매중지요청 타입입니다.

EmergencyMessageStatus

긴급메시지 상태입니다.

- `noreply`: 답변하지 않은 상태입니다.
- `replied`: 답변 완료한 상태입니다.
- `checked`: 관리자가 확인한 상태입니다.

NoticeType

알림메모 타입입니다.

- `refusedCancellation`: 주문 취소 관련 알림메모입니다.
- `order`: 주문 관련 알림메모입니다.
- `item`: 상품 관련 알림메모입니다.
- `orderTracking`: 운송장 관련 알림메모입니다.
- `etc`: 기타 알림메모입니다.

CustomsClearanceCodeType

해외배송 상품의 세관 통관시 필요한 개인정보 입력 타입입니다.

- `PersonalNumber`: 개인통관 고유번호입니다. `P`로 시작하고 숫자 12자리가 이어지는 모두 13자리의 문자열이어야 합니다.
- `BirthDate`: 생년월일입니다. `YYYYMMDD` 형식으로 입력하면 됩니다.

ProductStatusType

반품/교환할 상품의 상태 타입입니다.

- `Unopened`: 미개봉 상태입니다.
- `UsedAfterOpening`: 개봉후 사용한 상태입니다.
- `UnusedAfterOpening`: 개봉후 미사용한 상태입니다.

RefundOrExchangeReasonType

반품 구분의 타입입니다.

- `Simple`: 단순변심입니다.
- `DamagedItem`: 상품 파손/불량입니다.
- `WrongItem`: 상품 오배송입니다.

RefundType

반품 타입입니다.

- `Reship`: 재발송.
- `Exchange`: 교환.
- `Refund`: 반품.

ShippingFeePaymentType

반품 배송비 타입입니다.

- `CollectedOnDelivery`: 착불.
- `InAdvance`: 선불.
- `Enclosed`: 동봉.

DetailedReasonInput

반품 사유 입력시에 사용하는 정보입니다.

- `refundType`: `RefundType` - 상품 상태입니다.
- `detail`: `String` - 상세 내용입니다.
- `explanation`: `String` - 기타 정보입니다.

RefundAddressInput

반품 주소 입력시에 사용하는 정보입니다.

- `refundAddressType`: `RefundAddressType` - 반품 주소의 타입입니다.
- `address`: `String` - 반품 주소입니다.

RefundAddressType

반품 주소 타입입니다.

- `RecipientAddress`: 상품 수령인 주소.
- `ManuallyProvided`: 직접입력.

RefundTrackingInfoInput

반품 송장번호 입력시에 사용하는 정보입니다.

- `shippingCompanyCode`: `String` - 택배사 번호입니다.
- `trackingNumber`: `String` - 송장 번호입니다.

RefundProgressStatusType

반품 진행 상태 타입입니다.

- `CancellationAvailable`: 취소 가능.
- `CancellationRequestable`: 취소 요청 가능.
- `RefundOrExchangeRequestable`: 환불/교환 가능.
- `CancellationRequestInProgress`: 취소 진행중.
- `RefundAvailabilityCheckInProgress`: 환불 가능 여부 확인중.
- `TrackingInformationAwaiting`: 송장번호 입력 대기중.
- `ProductReceiptAwaiting`: 상품 수령 대기중.
- `RefundCompleted`: 환불 완료.

오픈마켓 참고 정보(카테고리)

`Item` 타입의 정보(`item` 쿼리의 반환 데이터, `allItems` 쿼리의 반환 데이터)에서 `Item.metadata`에서는 옥션, 지마켓, 인터파크, 11번가, 스마트스토어(구 스토어팜), 쿠팡에 대해 해당 상품의 오픈마켓 등록 참고 정보(카테고리 코드)를 제공합니다. 하지만 해당 카테고리 정보는 오픈마켓 측의 정책에 의해 언제든지 변경될 수 있으며, 오너클랜 측에서 해당 변경 사항을 인지한 후 대다수의 상품에 반영이 되기까지 시간이 걸릴 수 있습니다. 따라서 해당 정보는 참고용으로만 사용하시고 정확한 정보가 아닐 수 있음을 염두에 두어야 합니다.

지원하는 오픈마켓 별 필드 정보는 다음과 같습니다.

- 옥션: `Item.metadata.auctionCategoryCode` - `String` / `Item.metadata.auctionCategoryFullName` - `String`
- 지마켓: `Item.metadata.gmarketCategoryCode` - `String` / `Item.metadata.gmarketCategoryFullName` - `String`
- 인터파크: `Item.metadata.interparkCategoryCode` - `String` / `Item.metadata.interparkCategoryFullName` - `String`
- 11번가: `Item.metadata.st11CategoryCode` - `String` / `Item.metadata.11stCategoryFullName` - `String`
- 스마트스토어(구 스토어팜): `Item.metadata.smartstoreCategoryCode` - `String` / `Item.metadata.smartstoreCategoryFullName` - `String`
- 쿠팡: `Item.metadata.coupangCategoryCode` - `String` / `Item.metadata.coupangCategoryFullName` - `String`
- 플레이오토: `Item.metadata.playautoCategoryCode` - `String` / `Item.metadata.playautoCategoryFullName` - `String`
- 이셀러스: `Item.metadata.esellersCategoryCode` - `String` / `Item.metadata.esellersCategoryFullName` - `String`

해당 마켓에 대한 카테고리 데이터가 없는 경우 `null`을 반환합니다.

인증정보(certificateInformation)

HACCP(해썹), 자율안전인증, 특허인증 등 상품에 해당하는 인증 정보들입니다.

현재 11종의 인증정보를 지원합니다. 각 인증 정보는 인증 정보 타입을 나타내는 `certificateType`와 각 인증 정보에 필요한 필드들로 구성됩니다. 자세한 내용은 다음과 같습니다.

- Haccp(해썹) 지정업소
  - `certificateType`: `1`
  - `companyName`: `String` - 지정업소의 이름입니다.
  - `certifiedDate`: `String` - 인증 일자입니다. 포맷: `YYYY-MM-DD`
- 친환경 농축산물
  - `certificateType`: `2`
  - `certificateName`: `String` - 인증명입니다.
  - `certificateCode`: `String` - 인증 번호입니다.
  - `expiryDate`: `String` - 인증 만료일자입니다. 포맷: `YYYY-MM-DD`
- 유기가공식품
  - `certificateType`: `3`
  - `certificateName`: `String` - 인증명입니다.
  - `certificateCode`: `String` - 인증번호입니다.
  - `expiryDate`: `String` - 인증 만료일자입니다. 포맷: `YYYY-MM-DD`
- 기타

    - certificateType: 4
    - certificateName: String - 인증명입니다.
    - certificateCode: String - 인증번호입니다.
    - certifiedDate: String - 인증 일자입니다. 포맷: YYYY-MM-DD
    - expiryDate: String - 인증 만료일자입니다. 포맷: YYYY-MM-DD
- 자율안전 / 안전인증
    - certificateType: 5
    - certificateName: String - 인증명입니다.
    - certificateCode: String - 인증번호입니다.
    - certifiedDate: String - 인증 일자입니다. 포맷: YYYY-MM-DD
- 적합성평가
    - certificateType: 6
    - certificateName: String - 인증명입니다.
    - certificateCode: String - 인증번호입니다.
    - certifiedDate: String - 인증 일자입니다. 포맷: YYYY-MM-DD
- 특허
    - certificateType: 7
    - certificateCode: String - 인증번호입니다.
- HACCP(해썹) 축산물
    - certificateType: 8
    - certificateName: String - 인증명입니다.
    - certifiedDate: String - 인증 일자입니다. 포맷: YYYY-MM-DD
- [어린이제품] 안전인증
    - certificateType: 9
    - certificateIssuer: String - 인증기관명입니다.
    - certificateCode: String - 인증번호입니다.
    - companyName: String - 인증받은 상호명입니다.
    - certifiedDate: String - 인증 일자입니다. 포맷: YYYY-MM-DD
- [어린이제품] 안전확인
    - certificateType: 10
    - certificateIssuer: String - 인증기관명입니다.
    - certificateCode: String - 인증번호입니다.
    - companyName: String - 인증받은 상호명입니다.
    - certifiedDate: String - 인증 일자입니다. 포맷: YYYY-MM-DD
- [어린이제품] 공급자적합성확인
    - certificateType: 11
    - certificateIssuer: String - 인증기관명입니다.
    - companyName: String - 인증받은 상호명입니다.
- [생활용품] 안전인증
    - certificateType: 12
    - certificateIssuer: String - 인증기관명입니다.
    - certificateCode: String - 인증번호입니다.
- [생활용품] 안전확인
    - certificateType: 13
    - certificateIssuer: String - 인증기관명입니다.
    - certificateCode: String - 인증번호입니다.

- [생활용품] 공급자적합성확인
  - certificateType: 14
  - certificateIssuer: String - 인증기관명입니다.
- [생활용품] 어린이보호포장
  - certificateType: 15
  - certificateIssuer: String - 인증기관명입니다.
  - certificateCode: String - 인증번호입니다.
- [전기용품] 안전인증
  - certificateType: 16
  - certificateIssuer: String - 인증기관명입니다.
  - certificateCode: String - 인증번호입니다.
- [전기용품] 자율안전확인
  - certificateType: 17
  - certificateIssuer: String - 인증기관명입니다.
  - certificateCode: String - 인증번호입니다.
- [전기용품] 공급자적합성확인
  - certificateType: 18
  - certificateIssuer: String - 인증기관명입니다.
- [방송통신기자재] 적합인증
  - certificateType: 19,
  - certificateIssuer: String - 인증기관명입니다.
  - certificateCode: String - 인증번호입니다.
  - companyName: String - 인증받은 상호명입니다.
  - modelName: String - 인증받은 모델명입니다.
  - certifiedDate: String - 인증 일자입니다. 포맷: YYYY-MM-DD
- [방송통신기자재] 적합등록
  - certificateType: 20,
  - certificateIssuer: String - 인증기관명입니다.
  - certificateCode: String - 인증번호입니다.
  - companyName: String - 인증받은 상호명입니다.
  - modelName: String - 인증받은 모델명입니다.
  - certifiedDate: String - 인증 일자입니다. 포맷: YYYY-MM-DD
- [방송통신기자재] 잠정인증
  - certificateType: 21,
  - certificateIssuer: String - 인증기관명입니다.
  - certificateCode: String - 인증번호입니다.
  - companyName: String - 인증받은 상호명입니다.
  - modelName: String - 인증받은 모델명입니다.
  - certifiedDate: String - 인증 일자입니다. 포맷: YYYY-MM-DD
- 의료기기 판매업 신고
  - certificateType: 81
  - certificateIssuer: String - 인증기관명입니다.
  - certificateCode: String - 인증번호입니다.
- 의료기기 품목 허가
  - certificateType: 82
  - certificateCode: String - 인증번호입니다.

- 사전광고 심의
  - `certificateType`: `83`
  - `certificateCode`: `String` - 인증번호입니다.
- 건강기능식품 판매업신고
  - `certificateType`: `84`
  - `certificateIssuer`: `String` - 인증기관명입니다.
  - `certificateCode`: `String` - 인증번호입니다.

상품속성

상품에 대한 속성값들입니다.

- `"ATTR_H_FOOD"` - 건강기능식품 여부입니다.
- `"ATTR_MEDICAL"` - 의료용품 여부입니다.
- `"ATTR_OVERSEA"` - 해외배송 상품 여부입니다.
- `"ATTR_OVERSEA_DIRECT"` - 해외 직배송 상품 여부입니다.
- `"PROHIBIT"` - 유통금지 상품 여부입니다. 이 값이 있는 경우 유통금지처리된 상품이므로 최대한 이른 시일 내에 상품을 내려주시면 됩니다.

카테고리 속성

카테고리에 대한 속성값들입니다.

- `"HFOOD"` - 건강기능식품 여부입니다.
- `"MEDI"` - 의료용품 여부입니다.

정보고시

해당 내용은 별도의 문서로 제공됩니다.

본 가이드 문서와 같이 전달된 `OwnerclanAPI_ProductNotificationInformation.pdf` 파일을 참고해주세요.

상품 운영시 참조사항

국내 배송상품 / 해외 배송상품 구분 방법

국내 배송상품과 해외 배송상품은 `Item.attributes` 필드의 `"ATTR_OVERSEA"`, `"ATTR_OVERSEA_DIRECT"` 값을 통해 구분할 수 있습니다.

- `"ATTR_OVERSEA"`, `"ATTR_OVERSEA_DIRECT"` 값이 모두 없는 경우: 국내 배송 상품입니다.
- `"ATTR_OVERSEA"` 값이 있는 경우: 해외배송 상품입니다.
- `"ATTR_OVERSEA_DELI"` 값이 있는 경우: 해외 직배송 상품입니다.

기본 배송비와 배송비가 다른 상품 구분 방법

대부분의 오너클랜 상품의 배송비는 2500원이나, 일부 상품의 경우 2500원이 아닐 수 있습니다. 따라서 `Item.shippingFee` 필드의 값을 꼭 확인해주셔야 합니다. 또한 반품배송비 역시 배송비와 다른 경우가 있을 수 있으니 `Item.metadata.returnShippingFee` 필드를 확인해주셔야 합니다.

옵션 가격이 있는 상품 구분 방법

오너클랜 웹사이트에서는 상품 가격과 옵션 가격에 차이가 있는 경우에만 옵션 선택시에 추가 금액을 표시하고 있습니다. 그러나 API에서는 각 옵션에 대해 차액이 아닌, 차액이 계산된 최종 금액을 표시하고 있습니다.

예를 들어, 상품 가격이 10000원이고, 옵션 A는 추가금액이 1000원, 옵션 B는 추가금액이 1500원, 옵션 C는 추가금액이 0원이라고 가정해보겠습니다. 오너클랜 웹사이트에서는 다음과 같이 표시됩니다.

옵션 A(+ 1,000원)

옵션 B(+ 1,500원)

옵션 C

API 응답은 다음과 같습니다.

```
{
    "item": {
        ...
        price: 10000,
        ...
        options: [
            {
                "optionAttributes": [
                    {
                        "name": "옵션 속성",
                        "value": "옵션 A"
                    }
                ],
                ...
                price: 11000
            },
            {
                "optionAttributes": [
                    {
                        "name": "옵션 속성",
                        "value": "옵션 B"
                    }
                ],
                ...
                price: 11500
            },
            {
                "optionAttributes": [
                    {
                        "name": "옵션 속성",
                        "value": "옵션 C"
                    }
                ],
                ...
                price: 10000
            }
        ]
```

```
    }
}
```

따라서 오너클랜 웹사이트에서 표기되는 것과 같이 각 옵션의 차액을 계산하기 위해서는, `Item.options[idx].price`의 값에서 `Item.price`의 값을 빼주면 됩니다.

별도로 허가가 있어야 하는 상품군 구분 방법

건강식품 상품군과 의료기기 상품군은 별도의 취급 허가가 있어야 판매가 가능합니다. `Item.attributes`를 참고하여 구분할 수 있으며, 건강식품은 `"ATTR_H_FOOD"` 속성이 있는 경우이고 의료기기는 `"ATTR_MEDICAL"` 속성이 있는 경우이니 참고하시면 됩니다.

배송방식의 차이

위에서 서술했듯이, 배송방식은 `all`, `inAdvance`, `uponArrival`, `free`의 4가지 타입이 있습니다.

- `all`(선불/착불) - 선불과 착불이 모두 선택 가능한 경우입니다.
- `inAdvance`(선불) - 선불 전용 타입입니다. 착불로는 결제가 불가능합니다.
- `uponArrival`(착불) - 착불 전용 타입입니다. 화물배송이거나 설치형 상품의 경우 대부분 이 타입입니다.
- `free`(무료배송) - 무료배송 타입입니다. 배송비는 0원입니다.

묶음배송 가능 수량

`Item.boxQuantity` 필드는 묶음배송 가능 수량을 의미합니다.

여러 종류의 상품을 주문하는 경우, 배송비는 각 상품의 배송비를 계산한 다음, 그 중의 최댓값이 적용됩니다. 각 상품의 배송비를 계산하는 방식은 다음과 같습니다.

- 묶음배송 가능 수량이 없다면 해당 상품은 몇 개를 구매하더라도 `Item.shippingFee`의 값을 적용받습니다.
- 묶음배송 가능 수량이 있다면, 구매 수량을 `a`, 묶음배송 가능 수량을 `b`라고 한다면 해당 상품의 배송비는 `a-1`을 `b`로 나눈 값에 `Item.shippingFee`의 값을 곱한 값으로 적용됩니다. 예를 들어, `Item.shippingFee`가 2500이고, 묶음배송 수량이 5개이고, 해당 상품을 1~5개를 구매하는 경우에는 2500원이 적용됩니다. 하지만 6~10개를 구매하는 경우에는 2500 * 2 = 5000원이 적용되는 방식입니다.

위의 과정을 거쳐 각 상품의 배송비를 계산한 뒤, 상품의 종류가 여러 개라면 그 중의 최댓값이 최종 배송비가 됩니다.

반품 불가 상품

일부 상품의 경우 반품이 불가능한 경우가 있습니다. `Item.returnable` 필드의 값이 `true`면 반품 가능 상품이며, `false`면 반품 불가 상품입니다. 반품 불가 상품의 경우, 공급사 측에서 반품 불가 사유를 기입한 경우 `Item.noReturnReason` 필드에서 확인할 수 있습니다.

상품 판매상태 업데이트 가이드라인

오너클랜 API에서는 상품 정보를 `item` 쿼리나 `allItems` 쿼리를 통해 조회할 때 상품 판매 상태에 상관없이 모든 상품의 정보를 제공합니다. 이는 기존에 정상적으로 판매중이던 제품이라도 품절이나 단종 상태로 변경될 수 있고, 이 경우 API를 통해 상품을 조회하는 측에서 이를 확인할 수 있게 하기 위함입니다. 단, `allItems` 쿼리에 한해서 상품 판매상태에 대한 필터 조건을 설정할 수 있도록 했습니다.(`status` 파라미터.) 즉,`item` 쿼리나

`allItems` 쿼리(`status` 파라미터를 별도로 설정하지 않은 경우)를 통해 얻는 상품 정보는 **판매중인 상품**일 수도 있지만, `Item.status` 필드의 값에 따라 **품절된 상품**, **단종된 상품**, **기타 다른 사유로 판매할 수 없는 상품**일 수도 있습니다. 따라서 상품 판매상태는 API를 통해 얻어가신 상품 정보들에 대해 지속적으로 업데이트되어야 하며, 다음과 같은 방법들로 진행할 수 있습니다.

1. 개별 상품 정보 조회를 통해 업데이트. - 이 방법은 API를 통해 조회해서 사용하고 있는 상품 각각의 API key(오너클랜 상품코드와 동일)를 알고 있을 때 사용 가능합니다. - 각각의 상품 정보를 직접 조회하므로 가장 안전하지만, 1초에 최대 1000건만을 요청할 수 있습니다. - `item` 쿼리의 `key` 파라미터를 상품의 API key로 설정해서 해당 상품 정보를 조회한 뒤, - `Item.status` 필드의 값에 따라 상품 판매상태를 적절히 업데이트 합니다.
    - `available`: 판매중인 상품입니다.
    - `soldout`: 품절인 상품으로, 판매중으로 취급하고 있다면 품절상태로 전환해야 합니다.
    - `discontinued`: 단종된 상품으로, 판매중이거나 품절상태로 취급하고 있다면 단종상태로 전환해야 합니다.
    - `unavailable`: 취급 불가능한 상품으로, 판매중, 품절, 단종상태라면 판매 불가상태로 전환해야 합니다.
2. `allItems` 쿼리를 통해 업데이트. - 이 방법은 API를 통해 조회해서 사용하고 있는 상품 각각의 API key(오너클랜 상품코드와 동일)를 모르거나 일정 시간마다 주기적으로 업데이트 할 때 사용 가능합니다. - 단, 주기적으로 하는 만큼 특정 시간대의 요청이 실패하거나 하는 경우에는 연동 정보가 부정확해질 수 있습니다. - 따라서 한 번에 요청하는 시간 범위는 업데이트 주기의 2배 이상으로 하고 적절한 오류 처리와 함께 사용할 것을 추천드립니다.
    - 예를 들어, 업데이트 주기가 1시간이라면 최소한 직전 2시간 변경분에 대해 쿼리하는 것을 추천드립니다.
    - 또한 만약 오전 10시에 오전 8시-10시의 변경 데이터에 대한 요청이 실패했다면, 이를 기록하여 다음 주기인 오전 11시에 오전 8시-10시의 변경 데이터까지 포함해 오전 8시-11시의 변경 데이터를 요청하는 방식으로 진행하는 것이 안정성을 높일 수 있습니다.

```
- `allItems` 쿼리의 `dateFrom`, `dateTo` 필드의 값을 업데이트 주기를 고려하여 적절히 설정합니다.
- 페이지네이션 기능을 이용해 해당 시간 범위에 변경된 모든 상품을 조회한 뒤
- 각 상품 정보의 `Item.status` 필드의 값에 따라 상품 판매상태를 적절히 업데이트 합니다.
  - `available`: 판매중인 상품입니다.
  - `soldout`: 품절인 상품으로, 판매중으로 취급하고 있다면 품절상태로 전환해야 합니다.
  - `discontinued`: 단종된 상품으로, 판매중이거나 품절상태로 취급하고 있다면 단종상태로 전환해야 합니다.
  - `unavailable`: 취급 불가능한 상품으로, 판매중, 품절, 단종상태라면 판매 불가상태로 전환해야 합니다.
```

3. `itemHistories` 쿼리를 통해 업데이트. - 이 방법은 기존에 오너클랜 품절/단종 고지 페이지의 데이터를 이용해 업데이트를 하고 있었거나, 상품 상태 변경 이력이 필요한 경우에 사용 가능합니다. - 또한 연동 시각을 기록하고 있는 경우, 해당 시각 이후의 변경이력 데이터를 이용해 최신 상태로 업데이트할 수 있습니다. - 특정 상품의 변경 이력을 조회하여 업데이트하려는 경우, `itemKey` 파라미터에 해당 상품의 오너클랜 상품코드를 입력하여 검색할 수 있으며, - 특정 기간의 변경 이력을 조회하여 업데이트하려는 경우, `dateFrom`, `dateTo` 파라미터에 각각 기간의 시작과 끝을 입력하여 해당 기간동안 기록된 모든 변경 내역을 조회할 수 있습니다. 조회한 뒤 각 상품코드별로 데이터를 분류하여 사용하면 됩니다. - 특정 타입의 변동 내역만을 조회하여 업데이트하려는 경우, `kind` 파라미터에 변경 이력 타입을 명시하면 됩니다.

단, 한 번에 여러 타입을 조회할 수 없으므로 필요한 경우 쿼리를 여러 번 요청하면 됩니다. - 기록된 연동 시각을 바탕으로 업데이트 하는 경우 시나리오는 다음과 같습니다.

- 업데이트하고자 하는 상품에 대한 연동 시각 이후의 변경 이력을 조회합니다.
- 예를 들어 어떤 판매중인 상품의 마지막 연동 시각이 2019년 1월 1일 오후 3시라고 가정하고, 2019년 1월 31일 오후 3시에 변경 이력을 조회했다고 가정합니다.
- 이때 변경 이력 중 2019년 1월 10일에 `soldout` 이력이 있고, 1월 21일에 `restocked` 이력이, 1월 28일에 `discontinued` 이력이 있다고 가정합니다.
- 이 경우 가장 마지막의 변경 내역은 `discontinued`이므로 이 상품은 현재 단종상태일 것으로 판단할 수 있고, 단종상태로 업데이트하면 됩니다.

오픈마켓 판매 가능 여부

오너클랜 API에서는 특정 상품의 오픈마켓 판매 가능 여부를 `Item.openmarketSellable` 필드를 이용해 제공하고 있습니다. 또한 `allItems` 쿼리의 경우 `openmarketSellable`이라는 조건 파라미터를 제공하여 오픈마켓 판매가 가능한 상품만, 혹은 오픈마켓 판매가 불가능한 상품만을 불러오는 기능을 제공합니다. 다만, 해당 파라미터를 사용하지 않으면 기본적으로는 오픈마켓 판매가 가능한 상품만을 불러오게 됩니다. 만약 오픈마켓 판매가 불가능한 상품(`Item.openmarketSellable` 필드의 값이 `false`인 경우)을 오픈마켓에 등록하는 경우 오픈마켓으로부터의 제재 등 불이익이 발생할 수 있습니다. 따라서 API로 조회한 상품 정보로 오픈마켓에 상품을 등록하는 경우, 반드시 `Item.openmarketSellable` 필드의 값이 `true`인지 확인한 다음에 `true`인 상품만 등록하는 것을 권장합니다.