مرکز تنظیم مقررات نظام پایانه‌های فروشگاهی و سامانه مودیان

سند

«راهنمای اتصال به سامانه مودیان از طریق SDK جاوا»

اسفندماه ۱۴۰۲

مقدمه

در این سند راهنمای استفاده از SDK جاوا جهت سهولت در اتصال به سامانه مودیان شرح داده شده است. API سامانه مودیان از دو لایه تشکیل شده است، لایه انتقال و لایه مفهوم. لایه انتقال مستقل از اینکه چه نوع داده ای تبادل می‌شود، وظایف رمزنگاری و امضای بسته را بر عهده دارد. در لایه مفهوم، انواع بسته تعریف شده و بسته‌ها از طریق لایه انتقال به سامانه مودیان ارسال می‌شود.

متناظر با این لایه‌ها دو کتابخانه مستقل طراحی شده است. در صورتی که بسته جدیدی به سامانه اضافه شود، تنها کافی است که کتابخانه لایه محتوا بروزرسانی شود و یا می‌توان با استفاده از کتابخانه لایه انتقال بسته‌های جدید را ارسال کرد. کتابخانه مستقل دیگری با عنوان utils نیز ارائه شده است که در آن توابع کاربردی برای ارسال صورت حساب در آن گنجانده شده است.

آخرین تغییرات انجام شده در سند

ردیف	عنوان تغییرات	بخش مرتبط
۱	اضافه شدن فیلد tinc (شماره اقتصادی آژانس) به فیلدهای صورتحساب	قسمت ۱.۴
۲	تغییر وضعیت PENDING به IN_PROGRESS هنگام استعلام وضعیت صورتحساب	-

فهرست مطالب

۱-۱ شروع سریع

برای شروع سریع می‌توان کتابخانه‌های اصلی زیر را دانلود و به پروژه اضافه کرد و با استفاده از کد زیر صورت‌حساب را ارسال نمود. گرفتن اطلاعات سرور تنها یک بار برای دریافت کلید عمومی سازمان و گرفتن توکن دسترسی در صورت منقضی شدن آن تکرار می‌شود.

```
ApiConfig apiConfig = new ApiConfig().transferSignatory(
        SignatoryFactory.getInstance().createPKCS8Signatory(
                new File("C:/private.pem"), null));
TransferApi transferApi = new ObjectTransferApiImpl(apiConfig);
TaxApi taxApi = new DefaultTaxApiClient(transferApi, CLIENT_ID);

this.taxApi.getServerInformation();
taxApi.requestToken();

InvoiceDto invoiceDto = new InvoiceDto();
taxApi.sendMyInvoices(Collections.singletonList(invoiceDto));
```

۱-۲ لایه انتقال

وظیفه این لایه ارسال بسته به صورت همگام و یا ناهمگام سمت سامانه مودیان است. رمزنگاری و امضای بسته‌ها نیز توسط این لایه صورت می‌پذیرد.

۱-۲-۱ پیکربندی لایه انتقال

برای ایجاد کلاس TransferApi نیاز است که ابتدا آن را پیکربندی نمود. برای پیکربندی از کلاس ApiConfig استفاده می‌شود. مقادیر پیش فرض پیکربندی به صورت زیر است:

فیلد	نوع فیلد	مقادیر پیش فرض	توضیحات
httpRequestSender	HttpRequestSender	OkHttpRequestSender	کلاس پیاده‌ساز برای ارسال درخواست‌های http به صورت پیش‌فرض از OkHttp برای ارسال استفاده شده است.
normalizer	Normalizer	ObjectNormalizer	برای امضای یک شی، نیاز است که ابتدا آن را نرمال کنیم تا در مقصد به درستی امضاء بررسی شود. مقدار پیش فرض ObjectNormalizer است که یک شی را می‌تواند نرمال کند.

فیلد	نوع فیلد	مقادیر پیش فرض	توضیحات
encrypter	Encrypter	DefaultEncrypter	کلاس مربوط به چگونگی رمزنگاری بسته‌ها جهت ارسال به سامانه مودیان
transferSignatory	Signatory	Null	کلاس پیاده‌سازی مربوط به امضای یک رشته در لایه انتقال، با توجه به اینکه از چه مکانیزمی برای امضاء استفاده می‌شود، (توکن نرم‌افزاری، سخت‌افزاری و...) پیاده سازی‌های مختلفی برای این کلاس در نظر گرفته شده است. از طریق SignatoryFactory در SDK می‌توانید با داشتن فایل کلید خصوصی (به فرمت PKCS#8) یا به صورت Encode شده به فرمت Base64، یک شیء Signatory بسازید.
packetSignatory	Signatory	Null	کلاس پیاده‌سازی مربوط به امضای یک رشته در لایه محتوا، با توجه به اینکه از چه مکانیزمی برای امضاء استفاده می‌شود، (توکن نرم‌افزاری، سخت‌افزاری و...) پیاده‌سازی‌های مختلفی برای این کلاس در نظر گرفته شده است. از طریق SignatoryFactory در SDK می‌توانید با داشتن فایل کلید خصوصی (به فرمت PKCS#8) یا به صورت Encode شده به فرمت Base64، یک شیء Signatory بسازید.
apiProperties	ApiProperties	NormalApiProperties	در این فیلد باید یک شیء از یک پیاده سازی از اینترفیس ApiProperties را به کانفیگ بدهید. در حال حاضر دو پیاده سازی Normal و GSB از این کلاس وجود دارد که برای ارسال درخواست‌ها از طریق سرور GSB یا سرور خود سازمان می‌توان از آن‌ها استفاده نمود. هنگام ساختن یک شیء از این کلاس‌ها می‌توان Headerهای Http دلخواه یا نوع کلاینت (TSP یا SelfTSP) را مشخص نمود زیرا هر یک از این‌ها در آدرس APIهای فراخوانی شده برای هر درخواست تفاوت ایجاد می‌کنند. همچنین در

فیلد	نوع فیلد	مقادیر پیش فرض	توضیحات
			کلاس NormalApiProperties می‌توان ورژن API را هم مشخص نمود که آدرس‌های آن با آدرس پیشفرض متفاوت است (در صورت استفاده از ورژن ۱، این ورودی باید با "v1" پر شود.)
baseUrl	String	https://tp.tax.gov.ir/req/api	آدرس پایه اتصال به سامانه مودیان. این آدرس بر اساس اینکه به چه سروری (عملیاتی، آزمایشی) قرار است متصل شوند متفاوت خواهد بود. آدرس ارسال مستقیم مودیان به api/self-tsp ختم می‌شود و ارسال توسط شرکت‌های معتمد به api/tsp آدرس ختم می‌شود.
priorityLevel	PriorityLevel	NORMAL	الویت ارسال بسته‌های غیرهمگام. NORMALبرای کلاینت‌های عمومی با الویت عادی و HIGH برای کلاینت‌هایی که الویت بالا برای پردازش را دارند.

نمونه پیکربندی ساده API به صورت زیر است. برای پیکربندی، مشخص کردن کلاس امضا الزامی است. ورودی کلاس امضا کلید خصوصی است که کلید عمومی آن را در کارپوشه آپلود کرده‌اید.

```
ApiConfig apiConfig = new ApiConfig().transferSignatory(
        SignatoryFactory.getInstance().createPKCS8Signatory(
                new File("C:/private.pem"), null));
```

مقدار encrypter را می‌توان بعد از اینکه اطلاعات کلید عمومی سازمان را دریافت کردید تنظیم نمایید و یا همان ابتدا هنگام پیکربندی وارد کنید. نمونه پیکربندی کامل‌تر به صورت زیر است.

```
ApiConfig apiConfig = new ApiConfig()
        .baseUrl("https://tp.tax.gov.ir/req/api")
        .normalizer(new ObjectNormalizer())
        .encrypter(new DefaultEncrypter(ORG_PUBLIC_KEY, ORG_KEY_ID))
        .transferSignatory(
                SignatoryFactory.getInstance().createPKCS8Signatory(
                        new File("C:/private.pem"), null));
```

پیاده‌سازی ObjectTransferApiImpl نمونه پیاده‌سازی برای لایه انتقال می‌باشد. از کد زیر برای ایجاد کلاس لایه انتقال می‌توان استفاده نمود.

```
ApiConfig apiConfig = new ApiConfig()
        .baseUrl("https://tp.tax.gov.ir/req/api")
        .encrypter(new DefaultEncrypter(ORG_PUBLIC_KEY, ORG_KEY_ID))
        .transferSignatory(
                SignatoryFactory.getInstance().createPKCS8Signatory(
                        new File("C:/private.pem"), null
                ));
TransferApi transferApi = new ObjectTransferApiImpl(apiConfig);
```

۱– ۲– ۲ توابع ارسال بسته

دو تابع این لایه را ارائه می‌دهد. تابع ارسال همگام بسته و تابع ارسال ناهمگام بسته‌ها. تابع sendPacket یک بسته را به صورت همگام ارسال می‌کند و تابع sendPackets مجموعه‌ای از بسته‌ها را به صورت غیرهمگام ارسال می‌کند.

```
public interface TransferApi {

    AsyncResponseModel sendPackets(List<PacketDto> packets, Map<String,
String> headers, boolean encrypt, boolean sign);

    SyncResponseModel sendPacket(PacketDto packet, Map<String, String>
headers, boolean encrypt, boolean sign);
}
```

ورودی دوم این تابع هدرهای درخواست هستند. این هدرها اختیاری هستند و می توانید به عنوان ورودی ندهید. سه هدر در این لایه در نظر گرفته شده است. در صورتی که هر کدام از این هدرها در ورودی تعریف نشود، به صورت خودکار پر می‌شوند. هدر ها به شرح زیر است.

عنوان توکن	توضیحات
Authorization	توکن دسترسی به api
requestTraceId	شناسه تصادفی برای جلوگیری از تکرار درخواست
timestamp	زمان ارسال بسته به صورت timestamp به میلی ثانیه

قبل از ارسال هر بسته نیاز است که شناسه تصادفی تولید شده و از طریق هدر ارسال شود. استفاده از این شناسه مانع پردازش دوباره بسته‌های تکراری خواهد شد. قبل از ارسال بسته نیاز است که زمان تولید بسته را نیز به هدر ضمیمه نمود تا درخواست‌هایی که زمان ارسال آن‌ها بسیار گذشته مسدود شود.

در صورتی که بخواهید بسته به صورت رمزنگاری شده ارسال شود، فیلد encrypt را true قرار دهید.

در صورتی که بخواهید محتوای بسته امضاء شود مقدار فیلد sign را true قرار دهید.

۳-۲-۱ استفاده از توابع ارسال بسته

در این بخش چند نمونه کد برای ارسال بسته سمت سامانه مودیان شرح داده شده است. نمونه کد برای دریافت اطلاعات سرور به صورت زیر است.

```
    PacketDto packetDto = new PacketDto();
packetDto.setUid(UUID.randomUUID().toString());
packetDto.setPacketType(PacketType.PACKET_TYPE_GET_SERVER_INFORMATION);
packetDto.setFiscalId(شناسه حافظه);
packetDto.setData(null);
packetDto.setRetry(false);

SyncResponseModel response = transferApi.sendPacket(packetDto, null, false,
false);

Object responseData = response.getResult().getData();
ServerInformationModel si = mapper.convertValue(responseData,
ServerInformationModel.class);

KeyDto key = si.getPublicKeys().get(0);
DefaultEncrypter encrypter = new DefaultEncrypter(key.getKey(),
key.getId());
transferApi.getConfig().encrypter(encrypter);
```

با استفاده از اطلاعاتی که از سرور دریافت شده می‌توان، مقدار کلید عمومی سازمان را بروز نمود.

```
KeyDto key = si.getPublicKeys().get(0);
DefaultEncrypter encrypter = new DefaultEncrypter(key.getKey(),
key.getId());
transferApi.getConfig().encrypter(encrypter);
```

ارسال بسته برای دریافت توکن دسترسی به صورت زیر است.

```
GetTokenDto data = new GetTokenDto();
data.setUsername(clientId);

PacketDto packetDto = new PacketDto();
packetDto.setUid(UUID.randomUUID().toString());
packetDto.setPacketType(PacketType.PACKET_TYPE_GET_TOKEN);
packetDto.setFiscalId(شناسه حافظه);
```

```
packetDto.setData(data);
packetDto.setRetry(false);

SyncResponseModel response = transferApi.sendPacket(packetDto, null, false,
false);

Object responseData = response.getResult().getData();
TokenModel tokenModel= mapper.convertValue(responseData, TokenModel.class);
```

ایجاد بسته، برای ارسال صورت حساب به صورت زیر است.

```
AsyncResponseModel sendMyInvoices(List<InvoiceDto> invoices) {
    ArrayList<PacketDto> packets = new ArrayList<>();
    for (InvoiceDto invoiceDto : invoices) {
        PacketDto packetDto = new PacketDto();
        packetDto.setUid(UUID.randomUUID().toString());
        packetDto.setPacketType("INVOICE.V01");
        packetDto.setFiscalId(شناسه حافظه);
        packetDto.setData(invoiceDto);
        packetDto.setRetry(false);
        packets.add(packetDto);
    }

    HashMap<String, String> headers = new HashMap<>();
    headers.put(TransferConstants.AUTHORIZATION_HEADER, "Bearer " +
token.getToken());
    return transferApi.sendPackets(packets, headers, true, true);
}
```

۳-۱ لایه مفهوم

وظیفه این لایه ایجاد بسته‌هایی مطابق بسته‌هایی که سامانه مودیان پشتیبانی می‌کند، می‌باشد. بسته‌های ایجاد شده به کمک لایه انتقال، به سامانه مودیان ارسال می‌شود. در این بخش برای هر کدام از بسته‌ها نمونه کد ارائه شده است.

کلاس لایه مفهوم TaxApi است و پیاده سازی DefaultTaxApiClient به عنوان نمونه پیاده‌سازی این لایه در SDK قرار داده شده است. در سازنده این کلاس، شی مربوط به لایه انتقال و شناسه حافظه وارد می‌شود.

```
TaxApi taxApi = new DefaultTaxApiClient(transferApi, شناسه حافظه);
```

۱-۳-۱ دریافت اطلاعات سرور

در صورتی که کلید عمومی سازمان برای رمزنگاری در لایه انتقال داده نشده باشد، قبل از هر استفاده‌ای از taxApi نیاز است که یک بار تابع زیر صدا زده شود تا کلید عمومی سازمان در لایه انتقال بارگذاری شود.

```
ServerInformationModel serverInformation =
this.taxApi.getServerInformation();
```

۱-۳-۲ دریافت توکن دسترسی

برای گرفتن توکن دسترسی از تابع زیر استفاده می‌شود. با یک بار صدا زدن این api داخل کلاس DefaultTaxApiClientتوکن ذخیره می‌شود و در بقیه درخواست‌ها از آن استفاده می‌شود. طول عمر توکن در پاسخ بازگردانی می‌شود و از زمان دریافت آن تا مدت ذکر شده، توکن اعتبار دارد. در صورت منقضی شدن توکن و یا دریافت کد ۴۰۱ نیاز است که توکن دسترسی جدید دریافت شود.

تغییر توکن به صورت برنامه‌ریزی شده نیز فراهم شده است.

```
TokenModel token = this.taxApi.requestToken();
```

۱-۴ ارسال صورت حساب

برای ارسال صورت حساب نیاز است که فیلدهای زیر به صورت دقیق پر شوند.

کد	عنوان قلم اطلاعاتی	جایگاه	فیلد	نوع
۱	شماره منحصر به فرد مالیاتی	header	taxid	String
۲	تاریخ و زمان صدور صورتحساب(میلادی)	header	indatim	Long
۳	تاریخ و زمان ایجاد صورتحساب(میلادی)	header	Indati2m	Long
۴	نوع صورتحساب	header	inty	Integer
۵	سریال صورتحساب	header	inno	String
۶	شماره منحصر به فرد مالیاتی صورتحساب مرجع	header	irtaxid	String
۷	الگوی صورتحساب	header	inp	Integer

کد	عنوان قلم اطلاعاتی	جایگاه	فیلد	نوع
۸	موضوع صورتحساب	header	ins	Integer
۹	شماره اقتصادی فروشنده	header	tins	String
۱۰	نوع شخص خریدار	header	tob	Integer
۱۱	شماره/شناسه ملی/شناسه مشارکت مدنی/کد فراگیر خریدار	header	bid	String
۱۲	شماره اقتصادی خریدار	header	tinb	String
۱۳	کد شعبه فروشنده	header	sbc	String
۱۴	کد پستی خریدار	header	bpc	String
۱۵	کد شعبه خریدار	header	bbc	String
۱۶	نوع پرواز	header	ft	Integer
۱۷	شماره گذرنامه خریدار	header	bpn	String
۱۸	شماره پروانه گمرکی فروشنده	header	scln	String
۱۹	کد گمرک محل اظهار	header	scc	String
۲۰	شناسه یکتای ثبت قرارداد فروشنده	header	crn	String
۲۱	شماره اشتراک/ شناسه قبض بهره بردار	header	billid	String
۲۲	مجموع مبلغ قبل از کسر تخفیف	header	tprdis	BigDecimal
۲۳	مجموع تخفیفات	header	tdis	BigDecimal
۲۴	مجموع مبلغ پس از کسر تخفیف	header	tadis	BigDecimal
۲۵	مجموع مالیات بر ارزش افزوده	header	tvam	BigDecimal
۲۶	مجموع سایر مالیات، عوارض و وجوه قانونی	header	todam	BigDecimal
۲۷	مجموع صورتحساب	header	tbill	BigDecimal

کد	عنوان قلم اطلاعاتی	جایگاه	فیلد	نوع
۲۸	روش تسویه	header	setm	Integer
۲۹	مبلغ پرداختی نقدی	header	cap	BigDecimal
۳۰	مبلغ پرداختی نسیه	header	insp	BigDecimal
۳۱	مجموع سهم مالیات بر ارزش افزوده از پرداخت	header	tvop	BigDecimal
۳۲	مالیات موضوع ماده ۱۷	header	tax17	BigDecimal
۳۳	شناسه کالا/خدمت	body	sstid	String
۳۴	شرح کالا/خدمت	body	sstt	String
۳۵	واحد اندازه گیری	body	mu	String
۳۶	تعداد/مقدار	body	am	Double
۳۷	مبلغ واحد	body	fee	BigDecimal
۳۸	میزان ارز	body	cfee	BigDecimal
۳۹	نوع ارز	body	cut	String
۴۰	نرخ برابری ارز با ریال	body	exr	BigDecimal
۴۱	مبلغ قبل از تخفیف	body	prdis	BigDecimal
۴۲	مبلغ تخفیف	body	dis	BigDecimal
۴۳	مبلغ بعد از تخفیف	body	adis	BigDecimal
۴۴	نرخ مالیات بر ارزش‌افزوده	body	vra	BigDecimal
۴۵	مبلغ مالیات بر ارزش افزوده	body	vam	BigDecimal
۴۶	موضوع سایرمالیات و عوارض	body	odt	String
۴۷	نرخ سایرمالیات و عوارض	body	odr	BigDecimal

کد	عنوان قلم اطلاعاتی	جایگاه	فیلد	نوع
۴۸	مبلغ سایرمالیات و عوارض	body	odam	BigDecimal
۴۹	موضوع سایر وجوه قانونی	body	olt	String
۵۰	نرخ سایر وجوه قانونی	body	olr	BigDecimal
۵۱	مبلغ سایر وجوه قانونی	body	olam	BigDecimal
۵۲	اجرت ساخت	body	consfee	BigDecimal
۵۳	سود فروشنده	body	spro	BigDecimal
۵۴	حق‌العمل	body	bros	BigDecimal
۵۵	جمع کل اجرت، حق‌العمل و سود	body	tcpbs	BigDecimal
۵۶	سهم نقدی از پرداخت	body	cop	BigDecimal
۵۷	سهم ارزش افزوده از پرداخت	body	vop	BigDecimal
۵۸	شناسه یکتای ثبت قرارداد حق العملکاری	body	bsrn	String
۵۹	مبلغ کل کالا/خدمت	body	tsstam	BigDecimal
۶۰	شماره سوییچ پرداخت	payment	iinn	String
۶۱	شماره پذیرنده فروشگاهی	payment	acn	String
۶۲	شماره پایانه	payment	trmn	String
۶۳	شماره پیگیری	payment	trn	String
۶۴	شماره کارت پرداخت کننده صورتحساب	payment	pcn	String
۶۵	شماره/شناسه ملی/کد فراگیر اتباع غیر ایرانی پرداخت کننده صورتحساب	payment	pid	String
۶۶	تاریخ و زمان پرداخت صورتحساب	payment	pdt	Long
۶۷	شماره کوتاژ اظهارنامه گمرکی	header	cdcn	String

کد	عنوان قلم اطلاعاتی	جایگاه	فیلد	نوع
۶۸	تاریخ کوتاژ اظهارنامه گمرکی	header	cdcd	Integer
۶۹	مجموع وزن خالص	header	tonw	BigDecimal
۷۰	مجموع ارزش ریالی	header	torv	BigDecimal
۷۱	مجموع ارزش ارزی	header	tocv	BigDecimal
۷۲	وزن خالص	body	nw	BigDecimal
۷۳	ارزش ریالی کالا	body	ssrv	BigDecimal
۷۴	ارزش ارزی کالا	body	sscv	BigDecimal
۷۵	روش پرداخت	payment	pmt	Int
۷۶	مبلغ پرداختی	payment	pv	Long
۷۷	تفاوت نرخ خرید و فروش ارز/کارمزد فروش ارز	body	pspd	BigDecimal
۷۸	عیار	body	cui	BigDecimal
۷۹	شماره اقتصادی آژانس	header	tinc	String

شماره مالیاتی از سه بخش تشکیل می‌شود. بخش اول شناسه حافظه، بخش دوم تاریخ ایجاد صورت حساب و بخش سوم سریال صورت حساب است. برای سهولت تولید شماره مالیاتی می‌توان از کتابخانه Utils استفاده نمود. کد زیر نمونه تولید شماره مالیاتی را نمایش می‌دهد .

```
String taxId = TaxUtils.generateTaxId("A1116H", 10001, invoiceCreatedDate);
```

پارامتر اول شناسه حافظه، پارامتر دوم سریال صورت حساب که عددی حداکثر با طول ۱۲ رقم است و پارامتر آخر زمان صدور صورت حساب با فرمت Instant جاوا است.

نمونه کد ارسال صورت حساب به صورت زیر است.

```
    //Generate Random Serial number
Random random = new Random();
```

```
long randomSerialDecimal = random.nextInt(999999999);
Instant invoiceCreatedDate = Instant.now();
String taxId = TaxUtils.generateTaxId("A1116H", randomSerialDecimal,
invoiceCreatedDate);

InvoiceHeaderDto header = new InvoiceHeaderDto();
header.setIndatim(invoiceCreatedDate.toEpochMilli());
header.setIndati2m(invoiceCreatedDate.toEpochMilli());
header.setTaxid(taxId);
header.setInty(1);
header.setInp(1);
header.setInno(randomSerialDecimal);
header.setIns(1);
header.setTins("5555555555");
header.setTprdis(BigDecimal.valueOf(1000_000));
header.setTdis(BigDecimal.ZERO);
header.setTvam(BigDecimal.ZERO);
header.setTodam(BigDecimal.ZERO);
header.setTbill(BigDecimal.valueOf(1000_000));
header.setSetm(1);
header.setCap(BigDecimal.valueOf(1000_000));
header.setInsp(BigDecimal.valueOf(1000_000));
header.setTvop(BigDecimal.ZERO);
header.setTax17(BigDecimal.ZERO);

InvoiceBodyDto body = new InvoiceBodyDto();
body.setSstid("1111111111");
body.setSstt("پاستوریزه چرب کم شیر");
body.setMu(23);
body.setAm(2D);
body.setFee(BigDecimal.valueOf(500_000));
body.setPrdis(BigDecimal.valueOf(500_000));
body.setDis(BigDecimal.ZERO);
body.setAdis(BigDecimal.valueOf(500_000));
body.setVra(BigDecimal.ZERO);
body.setVam(BigDecimal.ZERO);
body.setTsstam(BigDecimal.valueOf(1000_000));


PaymentDto payment = new PaymentDto();
payment.setIinn("1131244211");
payment.setAcn("2131244212");
payment.setTrmn("3131244213");
payment.setTrn("4131244214");

InvoiceDto invoiceDto = new InvoiceDto();
invoiceDto.setBody(Collections.singletonList(body));
invoiceDto.setHeader(header);
invoiceDto.setPayments(Collections.singletonList(payment));

AsyncResponseModel responseModel =
taxApi.sendInvoices(Collections.singletonList(invoiceDto));
PacketResponse packetResponse = responseModel.getResult().get(0);
```

```
this.uid = packetResponse.getUid();
this.referenceNumber = packetResponse.getReferenceNumber();
```

۱–۵ استعلام نتیجه درخواست غیر همگام با استفاده از UID

خروجی ارسال بسته‌های غیرهمگام UID و referenceNumber است. فیلد uid سمت کلاینت و فیلد referenceNumber سمت سرور تولید می‌شود. با استفاده از این کدها می‌توان از وضعیت بسته ارسال شده با خبر شد. نمونه کد استعلام به صورت زیر است.

```
UidAndFiscalId uidAndFiscalId = new UidAndFiscalId();
uidAndFiscalId.setUid(this.uid);
uidAndFiscalId.setFiscalId(حافظه شناسه);
List<InquiryResultModel> inquiryResultModels =
this.taxApi.inquiryByUidAndFiscalId(Collections.singletonList(uidAndFiscalI
d));
```

۱–۶ استعلام نتیجه درخواست غیر همگام با استفاده از referenceNumber

با استفاده از این کدها می‌توان از وضعیت بسته ارسال شده با خبر شد. نمونه کد استعلام به صورت زیر است.

```
List<InquiryResultModel> inquiryResultModels =
this.taxApi.inquiryByReferenceId(Collections.singletonList(this.referenceNu
mber));
```

۱–۷ استعلام نتیجه درخواست غیر همگام بر اساس زمان

نتیجه درخواست‌ها از یک زمان بزرگتر را می‌توان با استفاده از این تابع با خبر شد. ورودی این تابع، تاریخ به صورت شمسی است.

```
List<InquiryResultModel> inquiryResultModels =
this.taxApi.inquiryByTime("14010101");
```

۱–۸ استعلام نتیجه درخواست غیر همگام بر اساس بازه زمان

نتیجه درخواست‌ها در یک بازه زمانی را می‌توان با استفاده از این تابع با خبر شد. ورودی این تابع، تاریخ به صورت شمسی است.

```
List<InquiryResultModel> inquiryResultModels =
this.taxApi.inquiryByTimeRange("14010101", "14020101");
```

۱–۹ گرفتن اطلاعات حافظه

ورودی این تابع شناسه حافظه بوده و خروجی آن اطلاعات حافظه مربوطه است.

```
FiscalInformationModel fiscalInformation =
taxApi.getFiscalInformation(CLIENT_ID);
```

۱–۱۰ استعلام کد اقتصادی

ورودی این تابع کد اقتصادی بوده و خروجی آن اطلاعات کد اقتصادی است.

```
EconomicCodeModel economicCodeInformation =
taxApi.getEconomicCodeInformation("5555555555");
```

۱–۱۱ گرفتن اطلاعات کالا و خدمات

ورودی این تابع فیلترها و صفحه بندی‌ها جهت گرفتن اطلاعات کالا و خدمات است. نمونه کد زیر برای بارگذاری اطلاعات ۱۰ کالا است.

```
SearchDto searchDto = new SearchDto();
searchDto.setPage(1);
searchDto.setSize(10);
SearchResultModel<ServiceStuffModel> serviceStuffList =
taxApi.getServiceStuffList(searchDto);
```

۱۲–۱ نمونه کد برای شرکت های معتمد

۱–۱۲–۱ ارسال با کلید شرکت معتمد

در این حالت بسته‌ها، چه در لایه انتقال و چه در لایه محتوا توسط کلید شرکت معتمد امضاء می‌شوند. تفاوت این روش ارسال با روش ارسال مستقیم توسط مودی، وجود شناسه حافظه‌های مختلف برای هر مودی و یک شناسه شرکت معتمد است. نمونه ارسال به صورت زیر خواهد بود.

```
InvoiceDtoWrapper invoiceDtoWrapper = new InvoiceDtoWrapper();
invoiceDtoWrapper.setInvoice(invoiceDto);
invoiceDtoWrapper.setFiscalId(MEMORY);

AsyncResponseModel responseModel =
taxApi.sendInvoices(Collections.singletonList(invoiceDtoWrapper));
```

۲–۱۲–۱ ارسال با کلید مودی

در این حالت محتوای بسته توسط کلید مودی امضاء و رمز می‌شود و توسط کلید شرکت معتمد ارسال می‌شود.

نمونه کد برای ایجاد بسته سمت مودی

```
ObjectNormalizer normalizer = new ObjectNormalizer();
String normalize = normalizer.normalize(invoiceDto, null);

Signatory signatory = SignatoryFactory.getInstance().createPKCS8Signatory(
        new File("C:/private.pem"), null);
String sign = signatory.sign(normalize);

PacketDto packetDto = new PacketDto();
packetDto.setUid(UUID.randomUUID().toString());
packetDto.setPacketType("INVOICE.V01");
packetDto.setFiscalId(MEMORY);
packetDto.setData(invoiceDto);
packetDto.setDataSignature(sign);
packetDto.setRetry(false);
```

بسته ایجاد شده با استفاده از api لایه انتقال ارسال می‌شود.

```
HashMap<String, String> headers = new HashMap<>();
if (token != null) {
    headers.put(TransferConstants.AUTHORIZATION_HEADER, "Bearer " + token);
}
AsyncResponseModel responseModel =
transferApi.sendPackets(Collections.singletonList(packetDto), headers,
true, false);
```