スマートコントラクト実証実験報告書

不動産売買におけるスマートコントラクトの有用性の検証

2018年 8月

スマートコントラクト実証実験

本書は、ルーデン・ホールディングス株式会社で不動産物件の売買における、仮想通貨及びスマートコントラクトの利用・使用に関して、その実用性や有益性を検証するための実験について記述したものである。

1. スマートコントラクト実証実験の目的

本実験の目的は、不動産売買における、仮想通貨およびスマートコントラクトの実用性を検証するである。

- 不動産の売買において、スマートコントラクトの仕組みを有効に利用することができるのか。
- 不動産売買において仮想通貨決済は技術的に利用することができかつ有用なものかどうか。
- これらの仕組みを利用することで改善することができる業務は何であり、具体的にどのように改善できるものか。

2. 結論

不動産売買において、仮想通貨およびスマートコントラクトを利用することは、以下の点で有用であると結論付けられる。

[不動産物件の買主側のメリット]

- 物件引渡までに要する時間を大幅に短縮できる。
- 売買契約条件などの交渉や締結までの時間を、現在業務に比べて短縮できる。
- 物件購入代金の支払い後の、不当な引渡し拒否等のトラブルを軽減できる。

[不動産物件の売主側のメリット]

- 物件購入代金の入金までに要する時間を大幅に短縮できる。
- 売買契約条件などの交渉や締結までの時間を、現在業務に比べて短縮できる。
- 購入申し込み後や、売買契約締結後の未払い等による時間的な逸失トラブルや、売買の中断といった不測の事態を軽減、回避できる。

特に、スマートコントラクトや仮想通貨決済を活用することは、単純にシステム化による業務効率化だけではなく、手戻りや契約の反故といった不測の事態を防ぐことにもつながり、その導入効果はかなり高いものと見込まれる。

3. 実験概要

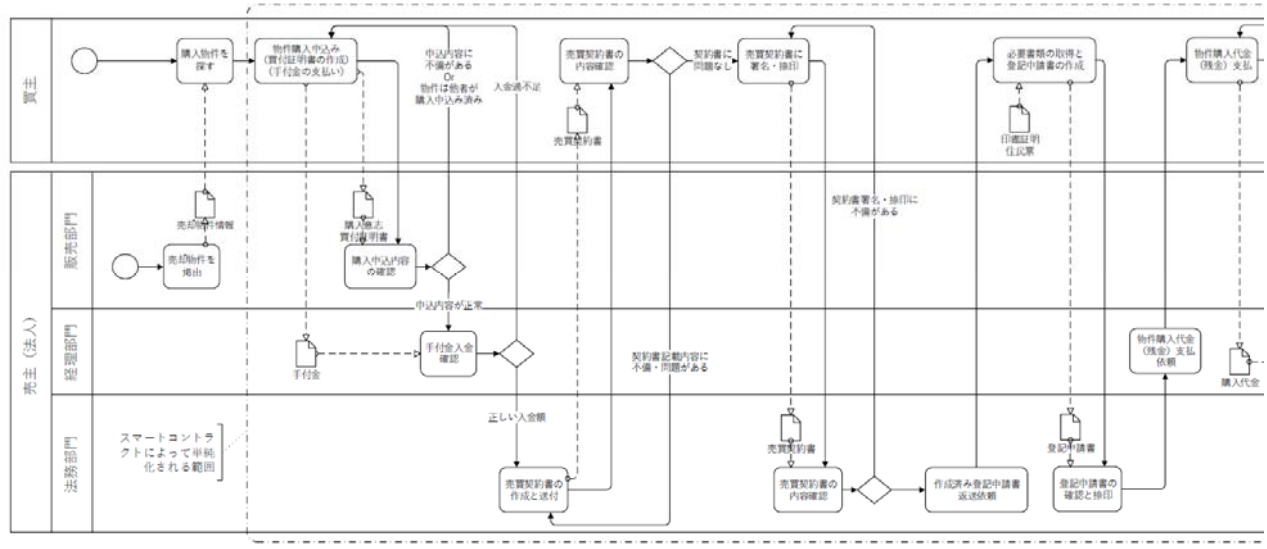
この章では、実験システムを構築するにあたって事前に検討された内容、特に業務フローを中心に解説し、スマートコントラクトを用いることが机上で有用であるかどうかを検証した一連の考察を含めた実験の実施概要を説明するものである。

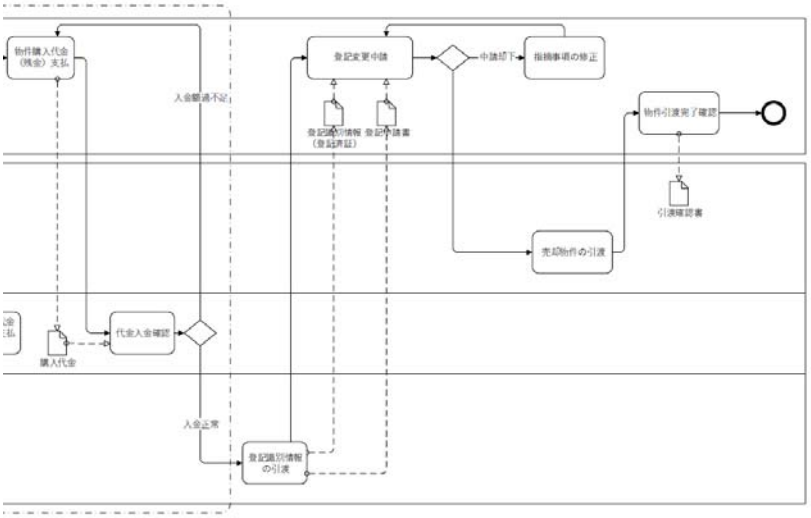
3.1. 想定する既存業務フロー

本実験にあたり、不動産売買における従前の業務フローの前提を、以下のように想定した。

1. 不動産物件を購入する買主は個人。
2. 売却物件を保有するのは法人であり、特に不動産業法人かどうかは定めていない。
3. 売却される物件は速やかに売却できる状態のものであり、抵当権が設定されているなどの複雑な条件を含まないものとする。
4. 物件売買に際し、仲介する不動産業者や、登記申請にあたって司法書士・弁護士を利用することは想定していない。（すべて自力で取得を行うものとする。）

これらの条件を踏まえ、流れ図に記したものが以下のものである。





このフロー図を正常な場合のみの手順で表すと、

1. 売主(販売部門)が売却物件を掲出、買主が購入物件を探す。

2. 買主が当該物件の購入申込みを行う。

3. 売主(販売部門)が購入申込内容を確認する。

4. 売主(経理部門)が購入申込みと同時に支払われた手付金の入金を確認する。

5. 売主(法務部門)が売買契約書を作成し、売主側の捺印をした状態で買主に送付する。

6. 買主側は送られた売買契約書を確認し、問題が無ければ契約書に署名・捺印して返送する。

7. 売主は買主が作成する登記申請書を送付するように要求する。

8. 買主は法務局に提出する登記申請書を作成して売主に送付する。合わせて、自分の印鑑証明書などを取得し、準備する。

9. 売主(法務部門)が買主から送られてきた登記申請書の内容を確認し、問題が無ければ署名・捺印し、物件の購入代金(手付金を除く金額)の支払いを要求する。

10. 買主は購入代金全額（手付金額を差し引いたもの）を送金する。

11. 売主(経理部門)が代金の支払いを確認する。

12. 売主から売却物件の登記識別情報(登記済証)と買主から送付されている登記申請書を合わせて買主に送付する。

13. 買主は法務局にて登記申請書を提出して、登記申請を行う。

14. 正しく登記申請が行われたのち、物件の引き渡しを行う。問題が無ければ買主は売主に対して引渡確認書を提示して、物件の売買が無事終了となる。

基本的には、買主と売主との間でそのフローが往復し続けるものであるが、特に書面や入金確認など、その内容を相互に逐次確認せねばならず、場合によっては手戻りが多く発生する恐れがある。

この従前の業務フローでは、購入申込みから物件購入代金の支払まで数日以上の時間を要する恐れがあり、なおかつ書類に不備があるために何度も返送しなおす、その代金の支払いが成されない、など最悪は契約の不成立に至るようなケースも想定される。

これらを踏まえ、スマートコントラクトおよび仮想通貨決済を活用した業務フローでは、できる限り買主と売主の業務が往復しないようかつ手戻り条件が少なくなるように想定されるべきである。

なお、この前提としている買主・売主が直接に交信する手順というのは、現在の不動産物件の売買においては非常に稀なケースであると考えられるが、基本的にはこれらの双方の業務の一部を仲介者などが補助、代行するものと考えれば、基本的な業務の流れは大きくは異ならない、として考えられている。

3.2. スマートコントラクトを活用した業務フロー

「3.1 想定する既存業務フロー」で示されたフローでは、購入申込みから物件購入代金の支払まで数日以上の時間を要する恐れがあり、なおかつ書類に不備があるために何度も返送しなおす、その代金の支払いが成されない、など最悪は契約の不成立に至るようなケースも想定されるなど、非常に不安定な業務フローであることが理解できた。

これらを踏まえ、スマートコントラクトおよび仮想通貨決済を活用した業務フローでは、できる限り買主と売主の業務が往復しないようかつ手戻り条件が少なくなるように、以下の条件を前提に想定した。

1. 不動産物件を購入する買主は個人であり、事前に必要な個人情報をルーデン社に提示し、確認されているものとする。

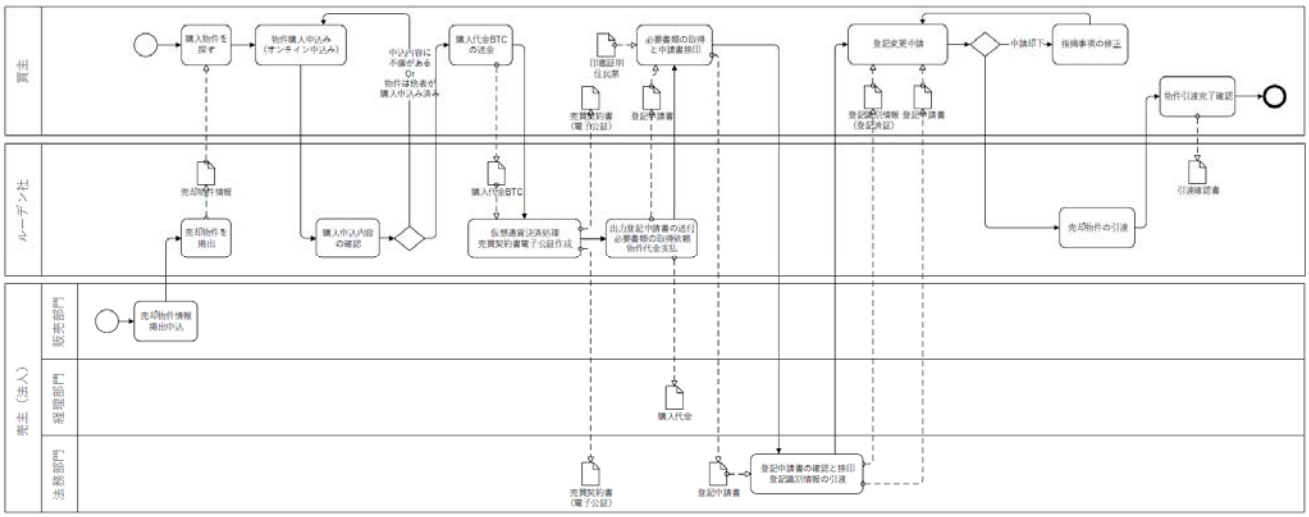
2. 売却物件を保有するのは法人であり、特に不動産業法人かどうかは定めていない。

3. 売却される物件は速やかに売却できる状態のものであり、抵当権が設定されているなどの複雑な条件を含まないものとする。

4. スマートコントラクトおよび仮想通貨決済の機能提供はルーデン・ホールディングスまたはその関連会社が行うものとする。

5. 物件売買に際し、仲介する不動産業者や、登記申請にあたって司法書士・弁護士を利用することは想定していない。（すべて自力で取得を行うものとする。）

これらの条件を踏まえ、流れ図に記したものが以下のものである。



このフロー図を正常な場合のみの手順で表すと、

1. 売主(販売部門)が売却物件を掲出、買主が購入物件を探す。

2. 買主が当該物件の購入申込みを行う。

3. ルーデン社(システム)が購入申込内容を確認する。

4. 買主はルーデン社の仮想通貨口座に仮想通貨(ビットコイン)を送金する。

5. ルーデン社(システム)は仮想通貨の送金完了を確認すると直ちに売買契約書のスマートコントラクトおよび仮想通貨の日本円への売却を実行する。なお、公証された売買契約書は速やかに買主および売主に共有される。また連続して、物件の登記申請

書を出力し、それを必要書類の取得依頼とともに買主に送付する。同時に物件売却代金を売主に引き渡す。

6. 買主は、ルーデン社から送られてきた登記申請書を基に、法務局に提出する登記申請書を作成して売主に送付する。合わせて、自分の印鑑証明書などを取得し、準備する。

7. 売主(法務部門)が買主から送られてきた登記申請書の内容を確認し、問題が無ければ署名・捺印し、買主に返送する。

8. 買主は法務局にて登記申請書を提出して、登記申請を行う。

9. 正しく登記申請が行われたのち、物件の引き渡しを行う。問題が無ければ買主は売主に対して引渡確認書を提示して、物件の売買が無事終了となる。

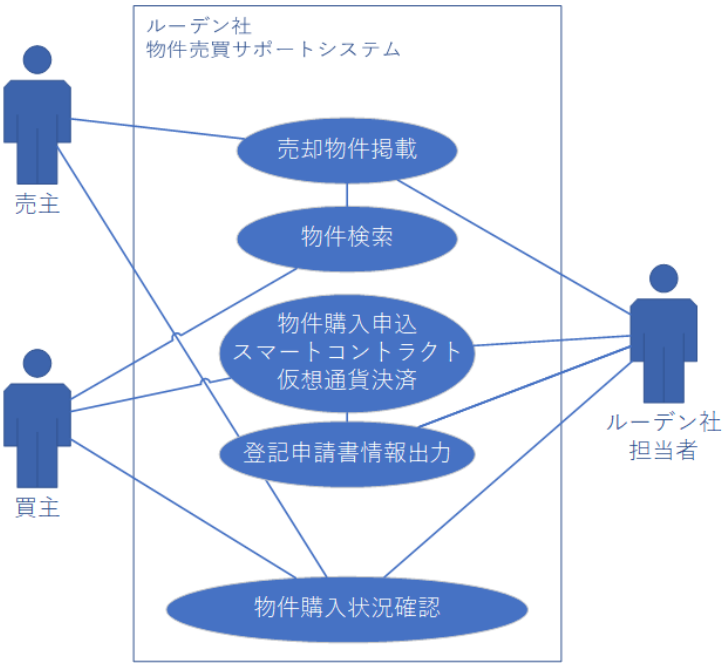
上記のスマートコントラクトを活用した手順と、「3.1 想定する既存業務フロー」で示された手順を比べると、その手順がおよそ 2/3 に削減されていることがわかる。特に「3.1 想定する既存業務フロー」での手順 4.から 11.に相当する手順が大幅に削減・改善されていることがわかる。加えて重要な点として、買主と売主との間でその作業が往復し続けないので、特に書面や入金確認など、その内容を相互に逐次確認する作業がほとんど削減されるため、契約の不成立に至るようなケースを避けることが可能になる、ということである。

この想定業務フローの整理によって判明したこととして、スマートコントラクトや仮想通貨決済を活用することは、単純にシステム化による業務効率化だけではなく、手戻りや契約の反

故といった不測の事態を防ぐことにもつながり、その導入効果はかなり高いものと見込まれる。

3.3. 実験における動作環境と条件

本実験は、以下のようなユースケースのシステムを想定しており、スマートコントラクト及び仮想通貨決済に必要な機能のみ実装し、その有用性を検証するものとする。



特に実験で検証を行うべき、スマートコントラクト及び仮想通貨決済を中心として実装し、検証を実施することとする。

なお本実験は、以下のような環境で行うものとする。

種類	使用ソフトウェア	備考
OS	Ubuntu 18.04	Docker コンテナとして環境を構築
アプリケーション動作環境	node.js 8.x python 3.x	node.js アプリが REST API サーバ(httpd)として作動
使用言語	JavaScript (TypeScript)	
使用ミドルウェア	RabbitMQ 3.6.x	
DBMS	なし	本試験では、データをファイルとして永続化
その他	Electron 3.1.3	ビットコインウォレットアプリ

システム構成にあたって、検証に必要な機能を構成する最低限のものを備えるとして、本来業務システムに備えるべきデータベースシステムや冗長化構成などの障害耐性やパフォーマンス向上、セキュリティ担保などを目的とした構成やソフトウェアは省略している。

売買契約書の公証を行うに当たっては、nem の試験用環境(TestNet)を使用するものとする。この試験用環境は、現実の価値を持つ XEM(nem における仮想通貨)ではなく、試験環境だけで有効な XEM(現実では無価値な XEM)を取得して利用する、ネットワーク上のブロックチェーンに接続されているコンピュータ(ノード)の台数が異なる、試験用のアドレスを用いる、等の条件が異なるだけで、処理や機能は本番用と全く同一の環境である。

また、物件売買に使用するビットコイン(BTC)の決済には、bitFlyer 社の Web サービスおよび API 接続機能を利用するものとする。bitFlyer 社の Web サービスや API 接続には試験

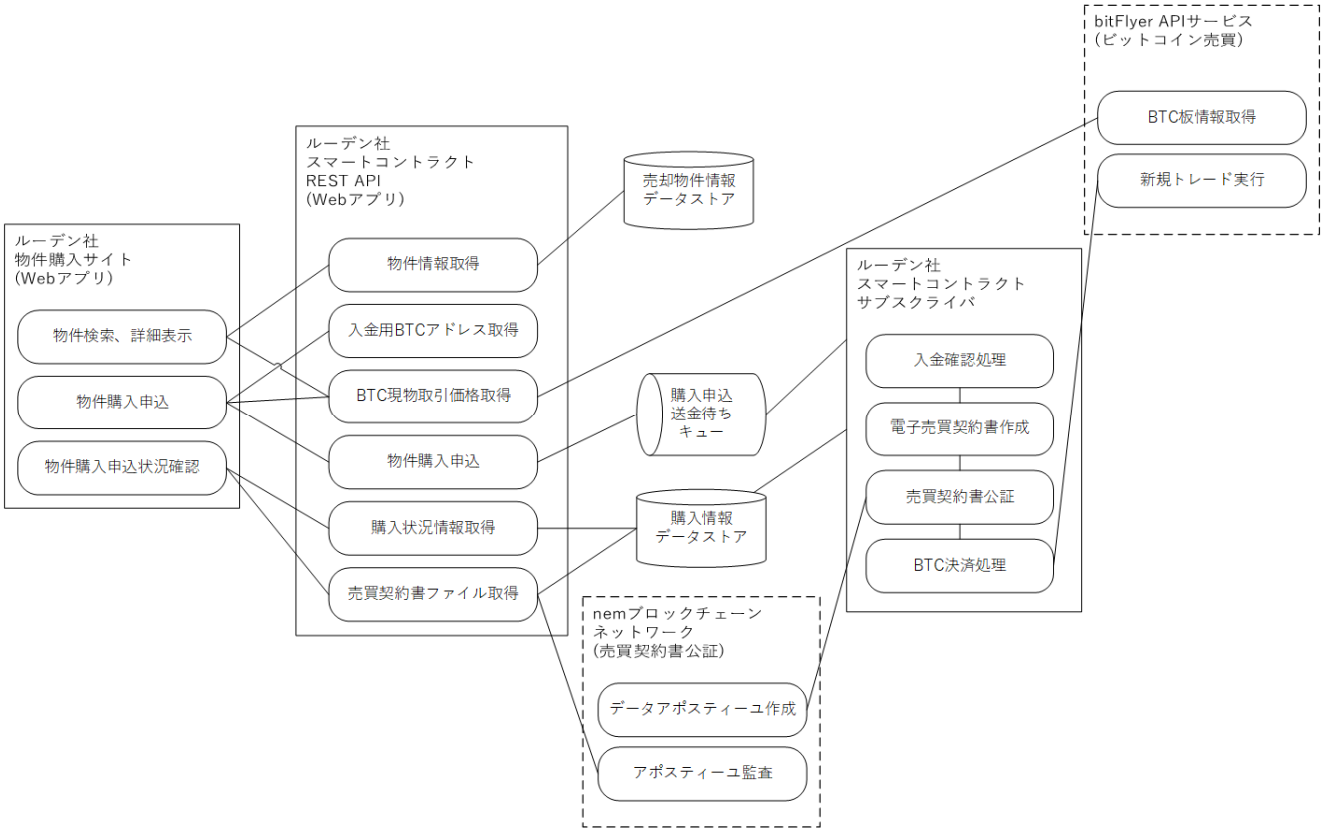
や評価用のものは存在しないため、本物のビットコインを用いて試験を行う。ただし、高額なビットコインでの取引試験はできない(想定している不動産物件売買に必要なビットコイン相当量を保有していない)ために、bitFlyer 社が取り扱うことができる最小単位である1mBTC(0.001BTC)を基準に、API 接続やビットコインの現物売りの試験を行うものとする。

4. 実験内容詳細

本章では、実験用システムを構築し、実際に稼働させてみた際の画面操作イメージを基に、システム内部で行っている処理を、より詳細に説明する。

4.1. 実験システムのアプリケーション機能構造

本実験を行うためのアプリケーションソフトウェアは、以下の図のような機能及び機能間の相互関連によって動作する。



本システムは主に以下のようなソフトウェアから構成されている。

1. 物件購入サイト

売主が売却希望している物件情報を掲載し、購入希望者(買主)はパソコンやスマートフォンのブラウザを通してサイトを閲覧する。希望条件に合った物件が見つかった場合、本サイト上で購入手続きを行う。

なお、本サイトを利用するにあたって購入者は、事前に必要な個人情報を本システム運営者に提示し、承認されているものとする。また、掲載されている物件は、速やかに売却できる状態のものであり、抵当権が設定されているなどの複雑な条件を含まないものとしている。

2. REST API サーバ

物件購入サイトからの処理要求を受け付け、各種データストアの情報を応答したり、購入希望情報を受け付ける等の処理を行う。このシステムは UI を提供しない。

また、仮想通貨(ビットコイン)の現在価格の取得や、公証された売買契約書の監査及び取得などは、外部のシステムと連携して処理を行う。これらの処理を隠蔽し提供する機能も有する。

3. スマートコントラクトサブスクライバ

REST API サーバの一部として動作し、メッセージキューサーバと協調して動作する、バックグラウンド処理用のサービス。

仮想通貨の送金(ビットコイン送金)は、(買主が)送金処理要求を送出してから(ルーデン社)受取口座の着金までに相当の時間を要する。その為、物件購入サイトで受け付けた購入申込と送金の完了が近しいタイミングで処理されるとは限らないために、物件購入サイトーREST API サーバを通して受け付けた購入申込と物件購入代金となる仮想通貨の着金を待ち合わせるために、メッセージキューを使用した待ち合わせの処理を行う。入金が確認された購入申込は、スマートコントラクトの処理が行われる。具体的には、自動的に売買契約書を作成し、それをブロックチェーンによって公証する。連続して仮想通貨を売却する処理も同時に行う。

4.2. 実験システムの実行画面

4.2.1. 購入物件を探す（買主）

まず買主が、購入物件を探すところから実験は始まる。なお、実験では事前に売主より売却物件情報を得ており、それが実験対象となる物件購入サイトに掲載(公開)されているものとしている。

掲載物件の売却価格(BTC 価格)は、リアルタイムに bitFlyer の BTC 価格が適用される。物件情報には売却希望価格は円で設定されており、物件の円価格＋手数料等相当の価格が、bitFlyer から取得できる BTC のマーケット価格によって算出され、表示されている。

買主はこの物件 BTC 価格以上のビットコインを保有している場合に、当該物件の購入申込を行うことが可能となる。

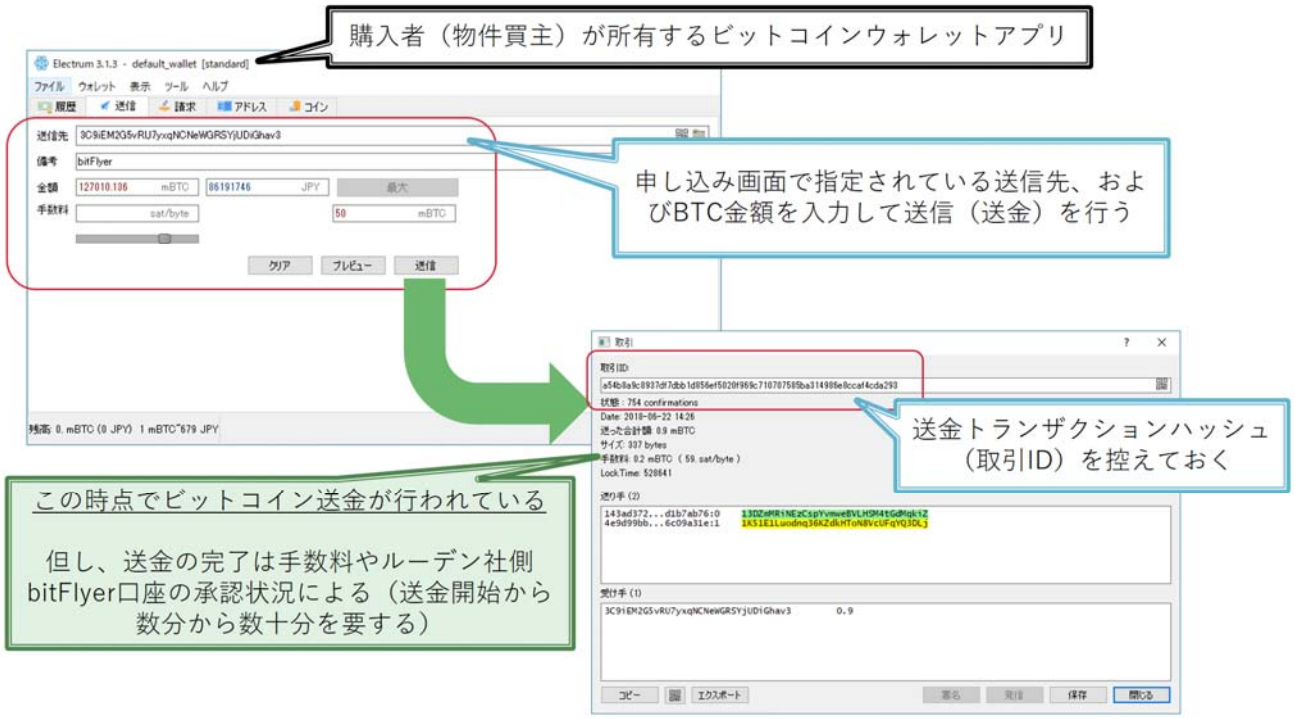
4.2.2. 物件購入申込み（買主）

物件の購入を希望する場合、買主は物件掲載ページ上の「ビットコインで購入」ボタンを押す。これによって、一旦この物件は購入申込を仮に受け付けている状態になる。(実験システムではこの仮申込状態による排他制御/他者が同時に同じ物件を購入するのを防ぐ、といった機能は割愛している。)

上図のように画面には送金先の口座(bitFlyer 上のルーデン社のウォレットアドレス)が表示されている。買主はこのアドレス宛に、自分が保有するビットコインのうち、物件購入に必要な量の BTC を送金する必要がある。

4.2.3. 購入代金（ビットコイン）の送金（買主）

買主は、前項のような画面に表示されている ルーデン社の bitFlyer ウォレットアドレス宛に、物件購入価格相当の BTC を送金する。



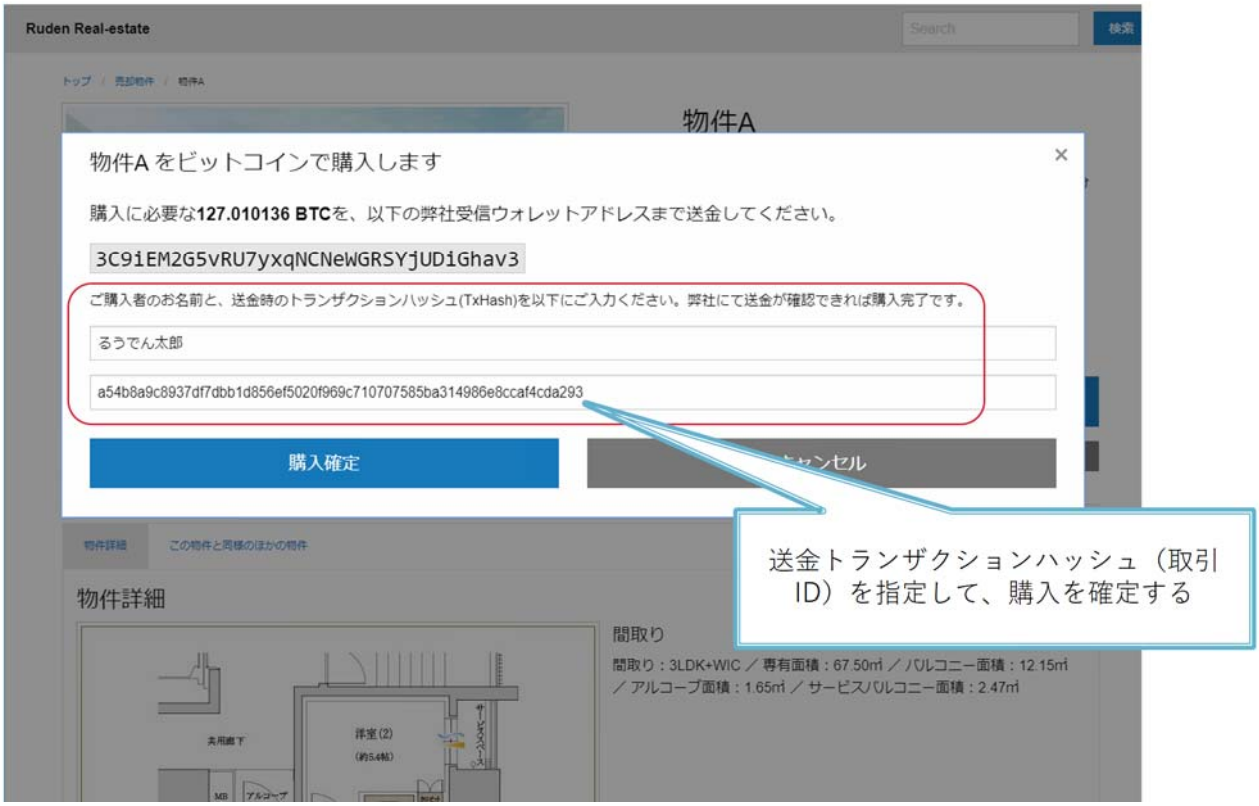
ビットコインの送金にあたっては、買主が所有するビットコインウォレットアプリから送金を行う。上図では「Electrum」というウォレットアプリを使用している画面を示している。

なお、送金にあたっての手数料等は買主負担で、送金手数料によって送金完了までの処理時間が変化する。これはビットコインのプルーフオブワーク(PoW)の速度によるものである。

実験においては、現物の 1mBTC(1 マイクロ BTC/1BTC の千分の一)を送金することで検証を行っている。

4.2.4. 購入申込み確定（買主）

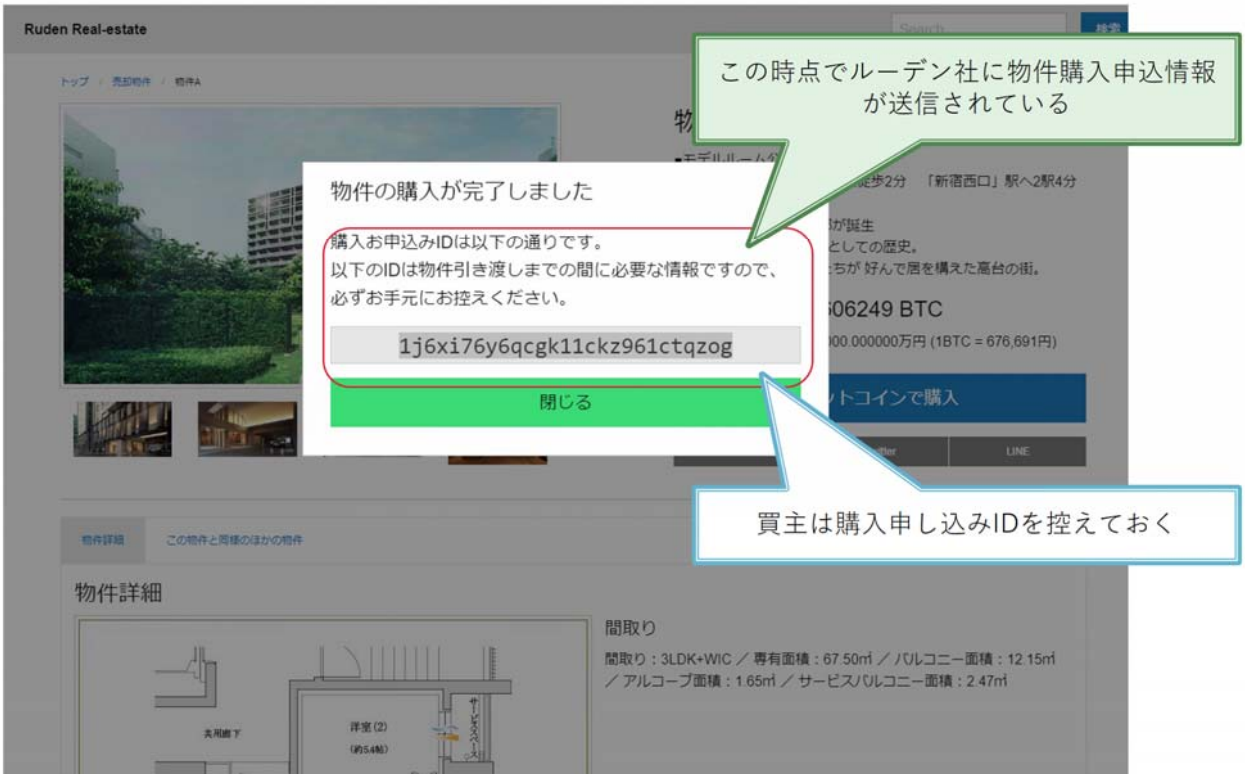
買主は、自分のビットコインウォレットより送金した際に得られる、送金トランザクションハッシュ(取引 ID または TxID)を、物件購入サイト上にて入力する。



この入力によって、システムは仮申込状態の購入希望を BTC 入金待ち状態に変更し、ルーデン社の bitFlyer 上のビットコインウォレットアドレスへの着金確認処理に移行する。

4.2.1. 購入申込み完了（買主）

前項で買主が送金トランザクションハッシュを入力することで、買主としての購入申し込みは完了となる。この時システムは、ルーデン社のビットコインウォレットへの着金確認処理を行っている。

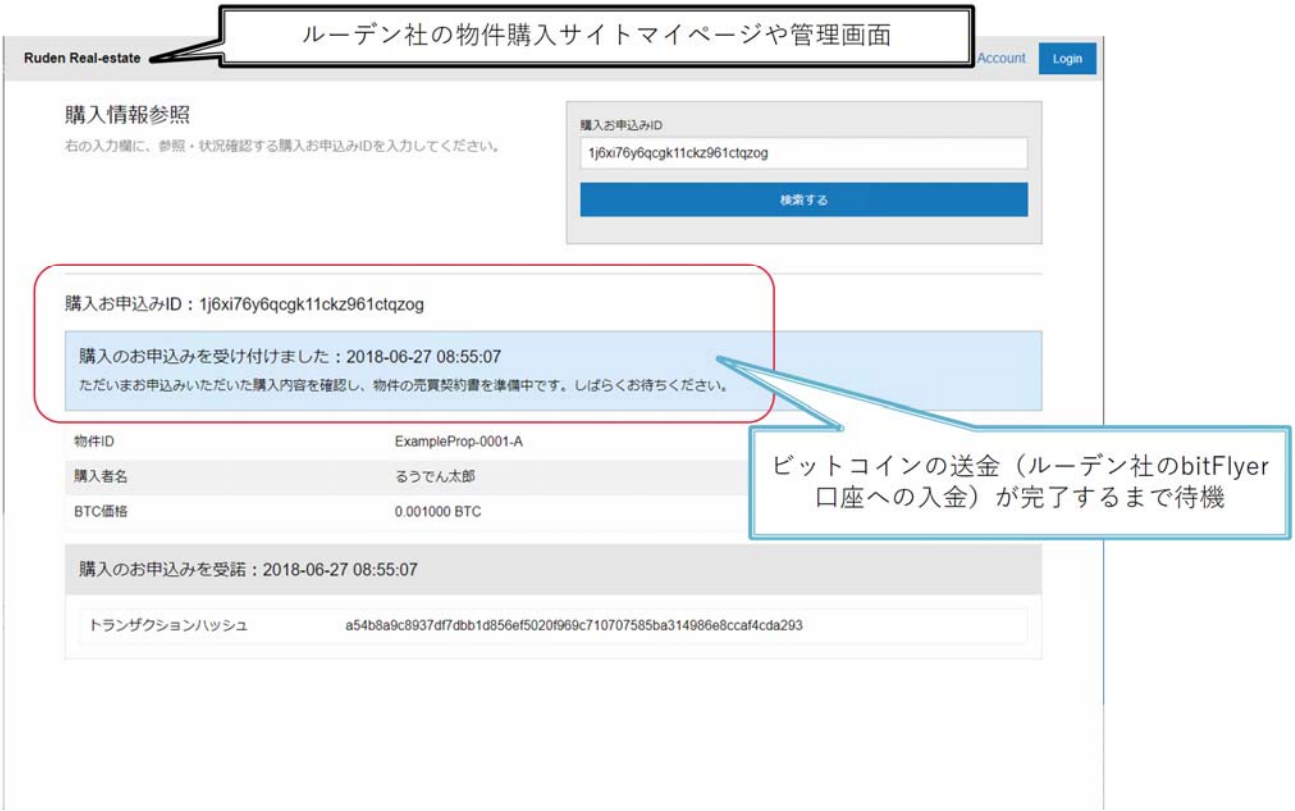


買主は上図のように画面に表示されている購入申込 ID(注文番号)を控えておく。この購入申込 ID はシステム上で一意に生成され、買主、売主、ルーデン社ともに共通の、この購入処理を表す識別子となる。

4.2.1. 購入状況を確認する（買主・売主）

物件購入サイトには、売却物件の検索及び購入のほか、マイページ(買主向け)や管理画面(売主およびルーデン社)が提供されると想定している。

このマイページや管理画面における購入情報参照機能では申し込みを行った購入情報の現在状況の確認や、スマートコントラクト処理が完了した後の売買契約書の取得(ダウンロード)を行うための機能を提供する。

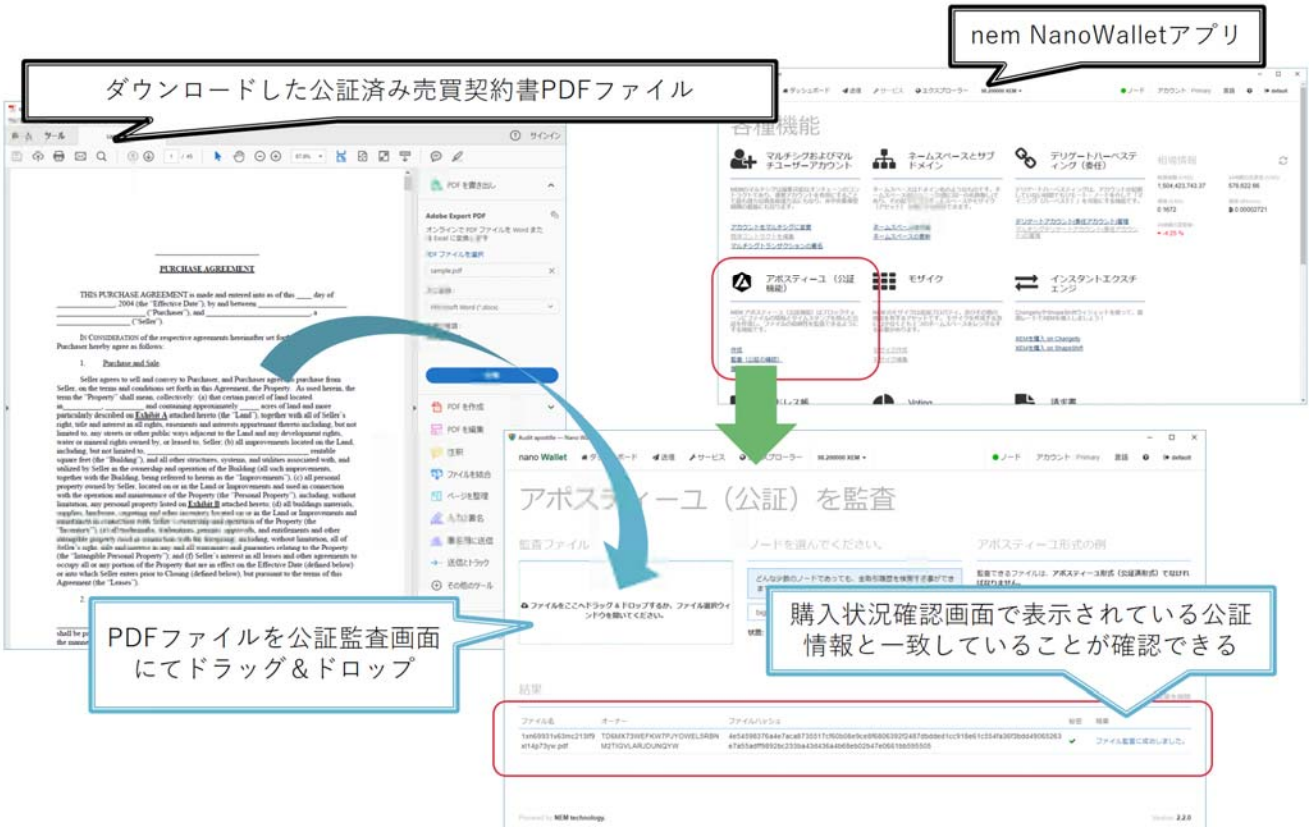


上図は購入申し込み直後のため、まだ bitFlyer 上のルーデン社ビットコインウォレットに着金が完了していない状態を示している。

着金が確認されると、システムは直ちにスマートコントラクトの処理や画僧通貨決済を実施する。着金確認からは遅くとも数秒で処理が完了し、購入情報参照機能の画面上では以下のような表示に切り替わる。

というファイル名であることが必須である。

このファイル名に則ったファイルをダウンロードして、nem の公式アプリである「nano Wallet」によって、その文書の公証および改ざんされていない状態を確認することができる。「nano Wallet」を用いた公証の監査手順は、下図の通りである。

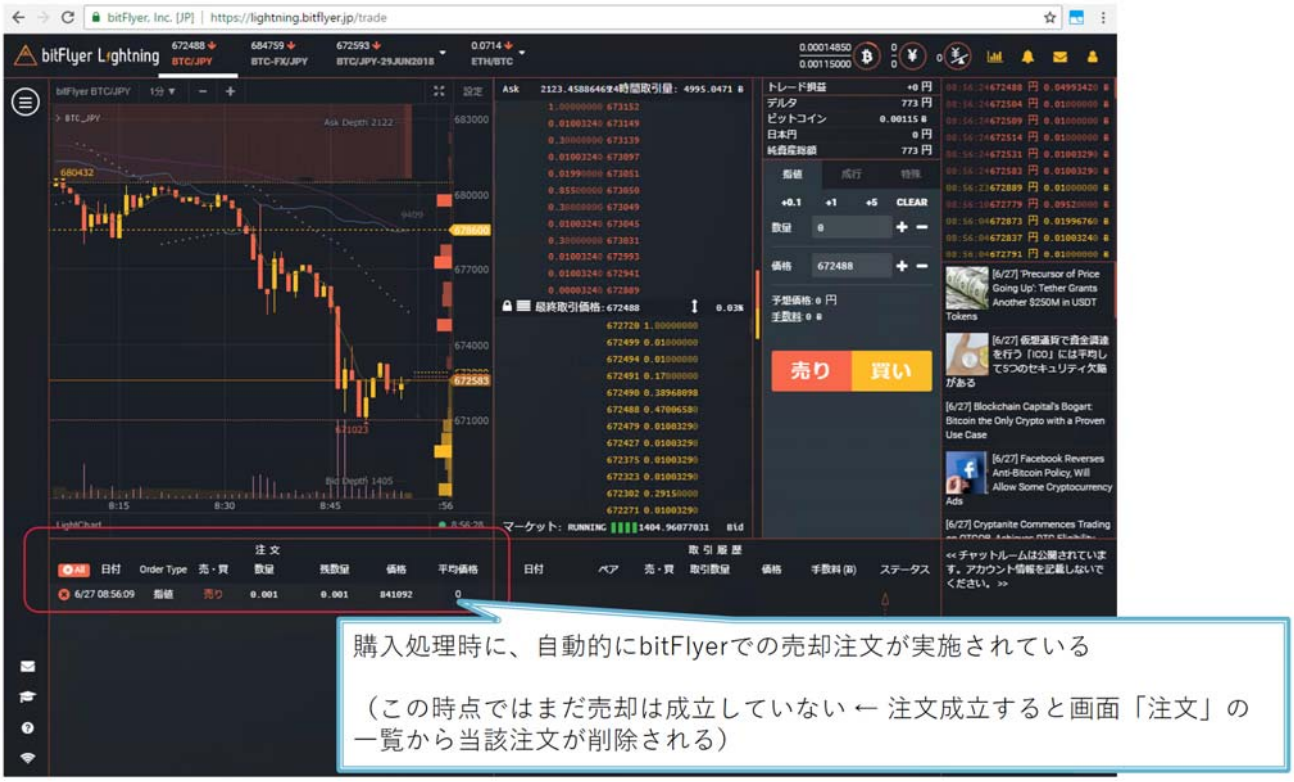


仮に、売買契約書は更新された場合でも、適切に公証を更新することも可能であるが、その際は買主、売主と共有している、古い契約書ファイルや公証情報も更新する必要がある。

4.2.1. 購入代金（ビットコイン）の現物売却状況を確認する（買主・売主）

スマートコントラクトにおいて、物件の購入代金に相当する仮想通貨(ビットコイン)は、bitFlyer の現物取引機能によって、直ちに売りに出される。

本来、この機能はシステムによって自動的に実施され、日本円に決済されるものであるが、このビットコインが実際に売却されている状況は、bitFlyer が提供する「bitFlyer Lightning」という取引ツールにて確認することができる。下図は、その「bitFlyer Lightning」にて、買主より得たビットコインの現物が売りに出されていることを確認している画面である。



なお、実験においては、現物の 1mBTC(1 マイクロ BTC/1BTC の千分の一)を送金することで検証を行っているため、上図画面でも表示されているのは 0.001BTC(1mBTC)となっている。実際には、物件の円価格価格＋手数料等相当の価格を BTC 換算した量が表示されている。

このビットコインの売却が完了すると、システムはスマートコントラクトに纏わる一連の自動処理が完了する。実験では、これらの処理がすべて自動的に完了することが確認できた。

4.3. nem におけるアポスティーユ処理

本実験のスマートコントラクトに関する処理は、nem のアポスティーユ(公証)の機能を利用することによって実現している。nem が提供する各機能が公式にライブラリとして公開されており、そのライブラリを利用することで nem が有する機能を自由に使用することが可能になっている。しかしながら、ライブラリには搭載されていてもドキュメントやコードサンプルが公式にはそろっていない機能も多数存在しており、本システムが利用する「アポスティーユ(公証)」機能も、公式の文書にはその使い方が記されていないものである。

本章は、nem アポスティーユに関して実験に際して調査した内容を記したものである。

```
import * as fs from 'fs';
import * as path from 'path';

import * as _ from 'lodash';
import * as moment from 'moment';
import * as nem from 'nem-sdk';

const NEM_ADDRESS = _.get(process.env, 'NEM_ADDRESS', 'privateKey');
const NEM_PRIVATEKEY = _.get(process.env, 'NEM_PRIVATEKEY', 'privateKey');
const ORDERS_PATH = path.join(_.get(process.env, 'DATA_PATH') || '../data', '/orders');

export const apostilleDocument = (orderId: string, fileContent: string): Promise<any> => {
  const common = nem.default.model.objects.create('common')('', NEM_PRIVATEKEY);
```

```
  const filestr = nem.default.crypto.js.enc.Base64.parse(fileContent);
  const filename = `${orderId}.pdf`;
  const apostille = nem.default.model.apostille.create(common,
    filename, filestr, orderId, // Tag
    nem.default.model.apostille.hashing['SHA256'], false, '', true,
    nem.default.model.network.data.testnet.id // 本番環境は mainnet.id
  );
  const date = moment(apostille.transaction.timestamp).utc().format('YYYY-MM-DD');
  const endpoint = nem.default.model.objects.create('endpoint')(
    nem.default.model.nodes.defaultTestnet,  // 本番環境は nodes.defaultMainnet
    nem.default.model.nodes.defaultPort
  );

  return nem.default.model.transactions.send(common, apostille.transaction, endpoint)
    .then((result: any) => {
      if (result.type >= 2) return Promise.reject(result);
      const apostilleFilename = `${orderId} -- Apostille TX ${result.transactionHash.data} --
Date ${date}.pdf`;
      fs.writeFileSync(path.join(ORDERS_PATH, `/${orderId}/${apostilleFilename}`), new
Buffer(fileContent, 'base64'));
      return Promise.resolve({
        address: NEM_ADDRESS,
        transaction: apostille.transaction,
        transactionHash: result.transactionHash.data,
        originalFile: filename,
        apostilleFile: apostilleFilename
      });
    });
}
```

上のコードは、本実験システムにおける nem アポスティーユ機能を利用している JavaScript(TypeScript)のコードサンプルである。なお、このコードは実験の前提条件の通りテスト環境への接続を行っているが、コード内コメントの箇所にて、MainNet への接続に変えることで、本番環境への接続を行うことが可能である。

nem のアポスティーユ機能を利用するに際して注意しなければならない点は、nem の公式ライブラリが提供する crypt 機能が提供する Base64 エンコードを利用して、公証するファイル(実験では売買契約書の PDF ファイル)の全データを得なければならない、ということである。

```
const filestr = nem.default.crypto.js.enc.Base64.parse(‘公証するファイルの中身’);
const filename = `公証するファイルの名前`;
const apostille = nem.default.model.apostille.create(common,
  filename, filestr, `公証時に保持するタグ`,
  nem.default.model.apostille.hashing['SHA256'], false, '', true,
  nem.default.model.network.data.testnet.id
);
```

これによって得られるファイルコンテンツを、nem ライブラリの appostille.create に引き渡すことで、公証に必要な情報が得られる。

この処理の後は、他の nem サンプルにも記載されている通り、nem のブロックチェーンネットワークに送信すれば、公証処理の完了である。

```
const endpoint = nem.default.model.objects.create('endpoint')(
  nem.default.model.nodes.defaultTestnet
  nem.default.model.nodes.defaultPort
);

return nem.default.model.transactions.send(common, apostille.transaction, endpoint)
  .then((result: any) => {

    // 送信後の処理

  })
;
```