



インストール、設定、管理。

ユーザーマニュアル

バージョン1.0ベータ

2020年7月に

MiniBlocklyChainは、OpenQbit Inc.の登録商標であり、無償使用および商用ライセンスに基づいています。利用規約と条件：www.OpenQbit.com

内容

1. 序章です。	4
2. ブロックチェーンスキームにおけるパブリックネットワーク、プライベートネットワークとは？	5
3. Blocklyプログラミングとは？	6
4. タームツクスとは？	6
5. ミニBlocklyChainとは？	6
6. ミニBlocklyChainのプロセスアーキテクチャ	12
7. BlocklyChain機能図（ミニBlocklyChain）。	16
8. ミニブロックリコードとは？	17
9. ミニBlocklyChain通信ネットワークの設置	18
10. システムノード（携帯電話）での同期を分秒で行う。	42
11. Termux内のストレージ構成。	52
12. 「Tor」のネットワークインストールと「Syncthing」のインストール。	54
13. データベース「Redis」とSSH（Secure Shell）サーバのインストール。	55
14. 携帯電話（スマホ）でのSSHサーバの設定。	56
15. SSH(Secure Shell)サービスを利用した「Tor」のネットワーク設定。	64
16. 手動同期によるPeer to Peerシステム構成。	67
17. Ambientes Blockly (App Inventor, AppyBuilder y Thunkable)。	79
18. Proof of Quantum (PQu)とは何ですか？	81
19. ミニBlocklyChainにおけるブロックの定義と使い方	85
20. SQLiteデータベースのブロックの使用（MiniSQLite版）	114
21. セキュリティブロックの定義と使い方	119
22. Mini BlocklyChainでセキュリティパラメータを設定します。	131
23. 別館「KeyStoreとPublicKeysデータベースの作成」。	135
24. 別館「RESTful SQLiteのGET/POSTコマンド」。	149
25. 付録「JavaコードSQLite-Redisコネクタ」。	156
26. 別館「開発者のためのミニBlocklyChain」。	160
27. 別紙「BlocklyCodeスマートコントラクト	172
28. 付録「OpenQbit Quantum Computing」。	179

29. 付録 「SQLiteデータベースの拡張ブロック	183
30. 別館 「ミニBlocklyChainシステムの作成例	184
31. 別冊 「Ethereum & Bitcoin環境との統合について	212
32. ソフトウェアのライセンスと使用。	233

1. 序章です。

ブロックチェーンは一般的にビットコインや他の暗号通貨と関連付けられていますが、デジタルマネーだけでなく、ユーザー・企業にとって価値のあるあらゆる情報に利用できるため、これらは氷山の一角に過ぎません。1991年にスチュアート・ハーバーとW・スコット・ストルネットが暗号的に保護されたブロックの連鎖の最初の作品を記述したことによる起源を持つこの技術は、ビットコインの登場とともに普及する2008年まで注目されていませんでした。しかし、現在は他の商用アプリケーションでの利用が求められており、中期的には金融機関やIoT（モノのインターネット）など、いくつかの市場で成長すると予測されています。

ブロックチェーンとは、ブロックチェーンという言葉で知られるように、ネットワーク上の複数のノード（PC、スマートフォン、タブレットなどの電子機器）に分散された、合意された単一の記録のことです。暗号通貨の場合は、それぞれの取引が記録されている会計帳簿と考えることができます。

その動作は、その実装の内部的な詳細にまで踏み込むと複雑なものになりますが、基本的な考え方方はシンプルです。

各ブロックに格納されています。

1. 有効な記録または取引の数。

2. - そのブロックに関する情報。

3. 前のブロックと次のブロックとのリンクは、各ブロックのハッシュを介して、ブロックの指紋のようになる一意のコード。

したがって、各ブロックは前のブロックのハッシュからの情報を含んでいるため、各ブロックはチェーン内で特定の移動不可能な場所を持っています。チェーン全体がブロックチェーンを構成する各ネットワークノードに保存されるため、チェーンの正確なコピーがすべてのネットワーク参加者に保存されます。

新しいレコードが作成されると、まずネットワークノードでチェックと検証が行われ、その後、チェーンに連結された新しいブロックに追加されます。

今、自分自身の間で通信するデバイスのこのネットワークは、人の介入なしで相互作用する能力を持っていることを想像してみてください、ブロックチェーンの大きな利点は、それが自律的な意思決定を行うことができることです、ユーザーへのサービスの応答時間の利点は、24 時間利用可能な日、ビジネスのコストを最小限に抑え、すべての最初のこのために、すでにテストされたセキュリティのレベルを持っていると他の理由は、さまざまな公共および民間部門での使用でとても人気となっている。

2. ブロックチェーンスキームにおけるパブリックネットワーク、プライベートネットワークとは？

公共のネットワーク。- それはお互いに通信し、匿名性を維持するコンピュータやモバイルデバイスのネットワークですが、それは、任意の人や会社が対話し、接続するための許可を必要としないので、ネットワークのこのタイプでは、いつでも接続することができます彼らの取引で正式な方法でこのネットワークで対話する人が知られていない、例はビットコインのブロックチェーンですが、誰もが購入または販売するために入力することができます。通常、ネットワークのこのタイプは、デジタル金銭資産またはその同義語"cryptomonies"、例の購入と販売に向けられているか、または指示されています。DogCoin、Ethereum、LiteCoin、BitCoin、Wavesなど。

プライベートネットワーク。- これは、コンピュータやモバイル機器が相互に通信を行うネットワークですが、パブリックネットワークとは異なり、プライベートネットワークに接続してネットワークに参加するためには、事前にどこかの事業者（企業や個人）からの承認が必要です。通常、プライベート・ブロックチェーン・ネットワークは、企業や企業において、ブロックチェーンによって適用・監督された文書、プロセス、承認、および/またはビジネス上の意思決定の形で有形の価値を持つことができる様々な異なるタイプの情報を用いて取引やオペレーションを行うために使用されています。

3. Blocklyプログラミングとは？

Blocklyは、パズルのピースであるかのように組み合わせができるシンプルなコマンドのセットで構成された視覚的なプログラミング言語です。直感的でシンプルな方法でプログラミングを学びたい人や、すでにプログラミングの方法を知っていて、このタイプのプログラミングの可能性を見てみたい人にはとても便利なツールです。

Blocklyは、コンピュータ言語のいずれかの種類のバックグラウンドを必要としないプログラミングの形態です。

誰もが理解するのが難しいそれらのプログラミング言語をいじらずに携帯電話（スマートフォン）用のプログラムを作成することができます、ちょうど作成するためのシンプルで簡単かつ迅速な方法でグラフィカルな方法でブロックをまとめてください。

4. タームックスとは？

Termuxは、Android端末エミュレータであり、ルーティングや設定を必要とせずに直接動作するLinux環境のアプリケーションです。最低限の基本システムが自動的にインストールされています。

安定性やインストール・管理のしやすさからTermuxを使用しますが、Ubuntu Linux for Androidのインストール環境を使用しても問題ありません。

このLinux環境では、MiniBlocklyChainの通信プロセスの"コア"を手に入れることができます。

5. ミニBlocklyChainとは？

Mini BlocklyChainは、Android OS（オペレーティングシステム）を搭載した携帯電話向けに開発された完全機能型ブロックチェーン技術で、取引の送受信を行うノードとなります。私たちは、最

低限の知識があれば、プログラミングの方法を知らなくても、誰でも携帯電話用のプログラムを作成・開発し、パブリックまたはプライベートネットワークモードで独自のブロックチェーンを作成できるBlocklyプログラミングを通じて、「モジュラー」的に構造化された初のブロックチェーン技術を開発しました。あなたがそれを行うことができます独自のデジタル通貨を作成したい場合は、それを行うことができますまたは企業内でそれを使用するためのソリューション、可能性は、それぞれの実際のケースのニーズと、ユーザーが彼の要件に応じて採用または作成するビジネスロジックに基づいています。

Mini BlocklyChainは、取引システムを短時間で実装できる携帯電話向けの初のモジュール型ブロックチェーンです。

「モジュール」ブロックの定義と使用に入る前に、実装したい実際のケースに応じて、いつ、どのように、どこで適用するかを知るために、ミニBlocklyChainコンポーネントの基本的な概念を持っておく必要があります。

以下のコンセプトでは、どのようなユーザーをターゲットにしているかを強調しているので、プログラミングのスキルがない人にとっては、開発ユーザーのコンセプトが十分に理解できているかどうかは重要ではありません。

基本的な概念。

ノードです。- Mini BlocklyChainのパブリックネットワークまたはプライベートネットワークの一部であるAndroidオペレーティングシステムを搭載した各モバイルデバイス（電話、タブレットなど）は、このネットワークのノードとして名前が付けられます。

ハッシュだ- それは、データ、文字列、文書またはデジタル情報のいくつかの種類のセットに関連付けられているデジタル署名であり、このデジタル署名は、送信された情報の最初のコンテンツを変更することができないように、情報を送受信するのに役立ちます一意であり、再現不可能です

。

活動的に。- 人や企業にとって何らかの有形・無形の価値を持つデジタルデータや情報（文書、デジタルコイン、音楽、動画、画像、デジタル認証など）の重み付けが可能なあらゆるタイプのもの。

トランザクション。- ある種の資産を占有しているノード間での情報交換。

UXTOトランザクション - ノードから1つまたは複数のトランザクションをパックし、各ノードからのすべての未使用トランザクション（UXTO : Unspent Transaction Outputs）を新しいトランザクションの入力として使用することができるトランザクションのタイプです。これらのトランザクションは、各情報フローの要件に応じて規制可能な一定時間ごとに処理されるキューにグループ化されている。

トランザクション（入力）。- UXTOトランザクションをベースにしたトランザクションの一種で、UXTOトランザクションキューが入ると、預金や経費として分類されたトランザクションをトランザクションで処理し、各ユーザの最終結果の残高を持つことができます。

ソースアドレス。- これは、SQLite マスター キューで処理するトランザクションを生成または送信する人のアドレスです。64文字の英数字で、システムで作成・検証されます。

宛先アドレス。- 取引を受けて残高に加算したり、送られてきた資産を保管したりする人のアドレスです。64文字の英数字で、システムで作成・検証されます。

SQLiteマスター。- トランザクションを受け取り、業務の必要性に応じて特定の変数や一定時間ごとに処理されるキューを作成するAPI RESTデータベースです。

DataBase トランザクション。- Mini BlocklyChainのトランザクション情報を予約または保存しているSQLiteデータベースに反映されているトランザクションです。

A E S- Mini BlocklyChainのSQLiteデータベースに格納されているデータを暗号化するセキュリティ処理です。

バランスをとる。- Mini BlocklyChainで任意の取引処理を行った後、有形価値（暗号通貨）または無形価値（人や企業間のビジネスプロセス）のいずれかとして操作のバランスを取得します。

業務プロセス。- それは、エンドユーザーに結果を得るために何らかの情報の流れを伴うプロセスであり、エンドユーザーは、民間または公共部門の様々なセクターからの個人または企業である可能性があります。

公開鍵と秘密鍵。- 情報暗号化の一種で、お互いに関連付けられた2つの英数字キーが生成され、公開または非公開のネットワークを介して機密情報を送信するために使用されます。秘密鍵は情報を暗号化するために使用され、ネットワークを介して送信する際に改ざんされたり、一見して読み取れないようにするために、公開鍵は共有され、どのような人や企業でも見ることができるもので、これは送信された情報を確認するのに役立ち、また、有効な差出人だけが読むことができるようになるためのものです。

デジタル署名です。- これは秘密鍵で作成できる署名であり、この署名により、受信者は情報が変更されていないことを保証し、受信者にとって情報が有効であることを確認します。

デジタルアドレス（ミニBlocklyChainアドレス）。- ミニBlocklyChainの各ユーザーに固有の英数字で構成されたアドレスで、このアドレスは、ユーザー間のトランザクションを実行するために使用され、パブリックまたはプライベートネットワークかどうか。

マジックナンバー。- これは、各実行中のUTXOトランザクションのビジネスルールによって定義された乱数で、ノードがUTXOトランザクションを実行する候補となり、新しいブロックであるMini BlocklyChainを作成することを承認する役割を果たします。

新しいブロックの作成。- Mini BlocklyChainの新しいブロックは、UTXOトランザクションを処理するために勝利候補として出てきたノードが作成したハッシュを添付したものです。

PoW (Proof of Work) のコンセンサスプロトコル。- これは、すべてのノードを実行するテストであり、テストを正常に終了した最初のノードは、UTXOトランザクションを実行するために選択されたノードです。それは、結果（ハッシュ）を得るために数学的な計算で構成されているアルゴリズムであるミニBlocklyChainによって実行される各トランザクションで与えられたルールに応じて摩耗やコンピュータの情報処理リソースの需要に基づいて、ミニBlocklyChainの場合には、構造化されており、"マジックナンバー"を取得するために変更されているこれは、承認やコンセンサスを持つことがで

きるようにするための番号であるUTXOのトランザクションを実行することができます1つのノードの大
多数のことです。PoWは「魔法の数」を取得するためだけに使用するため、最大難易度は5となっ
ています。

ポーカコンセンサスプロトコル- これは、すべてのノードを実行し、PoWによって最初に構成されてい
るテストです"魔法の数"を取得し、後でそれはQRNG（量子乱数発生器）量子乱数発生器に
基づいて確率アルゴリズムを実行し、このプロセスは、すべてのノードの確率の平等を保証し、どの
ノードがUTXOトランザクションと新しいブロックの作成を実行する1つになります選択します。

メルクリの木。- これは、Mini BlocklyChain がトランザクションが有効であること、または外部の工
ンティティによって変更されていないことを保証するためのセキュリティ手法です。ハッシュメルクリツ
リーとは、2値または非2値のデータツリー構造で、シートではない各ノードには、その子ノードのラベ
ルまたは値の連結のハッシュがタグ付けされています。これらはハッシュリストとハッシュ文字列の
一般化です。

Redis DB。- 要求された新しいトランザクションを処理し、ノードに送信するために使用されるリアル
タイムトランザクションデータベース。

SQLite DB。- ネットワークの整合性を確保するために、Mini BlocklyChain のバランスと新しいブロッ
クが保存されているデータベース。

セントリー。- RedisとSQLite間のセキュリティとデータ整合性のコネクタ。このコネクタまたはプログラ
ムは、トランザクションのレビュー、処理、検証、配信、ノードへの翻訳を担当します。

OpenSSH。- Androidオペレーティングシステム内でタスクを実行するためのセキュリティコネクタ。

Termuxシェルターミナル。- Mini BlocklyChain のトランザクションを処理するためのサードパーティの
依存関係を見つけることができるプログラムで、このバージョン 1.0 では簡単にインストールするため
に使用されていますが、将来のバージョンでは現在開発中の BlocklyShell システムに置き換えら
れます。

財布- それは、あらゆるトランザクションにおいて2つの基本的なデータが保管されているデジタルリポジトリです。Mini BlocklyChainのアドレスや、UXTOの取引結果(Balance)を保管しているリポジトリです。

アプリケーション開発者またはプログラマー。

オブジェクト指向プログラミング- Mini BlocklyChainはJavaで作られています。

BlocklyCodeです。- Mini BlocklyChain環境で実行可能なインテリジェントコントラクトの用語で、BlocklyCodeはjavaプログラミングで作成されています。

Android用のOpenJDKです。- BlocklyCodeを作成して実行するAndroid用JDKとJREのスイートです。

QRNG（量子乱数発生器）。- これは量子乱数発生器で、Mini BlocklyChainは現在、これらを生成するための2つのAPIを持っています。

rsyncを使用しています。- それは、インクリメンタルデータの効率的な転送を提供するUnixとMicrosoft Windowsタイプのシステムのための無料のアプリケーションですが、圧縮されたデータと暗号化されたデータでも動作します。

フセンド- コマンドラインから簡単・安全にファイルを共有するためのアプリケーションです。

NTPです。- Network Time Protocol (ネットワークタイムプロトコル) は、ネットワーク上でパケットルーティングを介してコンピュータシステムのクロックを同期させるためのインターネットプロトコルです。

Termux（開発）。- オープンソースのアプリケーションやライブラリが豊富なシェルターミナル。

ブロック状のモジュール（データ）。- 開発者は、Mini BlocklyChain の機能を強化するためにモジュールを統合することができます。

ピア・ツー・ピア。- この用語は、ノード間の直接通信を指し、情報の更新が中央サーバに依存せず、各ノードが同じ情報を持つすべてのノード間で通信を行う中央サーバとして機能し、障害点を回避するのに役立つことを意味します。

シンクティング。- 「ピア・ツー・ピア」通信タイプを使用して、2つのデバイス間でデータやファイルを同期するためのツールです。

レッド・トー。- これは、インターネット上に低遅延の分散通信ネットワークを重ね合わせたもので、ユーザ間で交換されるメッセージのルーティングでは、ユーザの身元、すなわちIPアドレスを明らかにしない（ネットワーク全体の匿名性）。

モバイルルーティング。- お使いの携帯電話に外部ソフトウェアのインストールプロセスは、Linuxオペレーティングシステム（Android）のシステム管理者として入力するには、管理者ユーザーは"root"と呼ばれていますあなたの携帯電話を回転させることができるようになるには、任意のプロセスへのアクセス権を持っています。注意したいのは、携帯電話メーカー（スマートフォン）によっては、携帯電話に不具合があると保証がなくなると言っているところがあるということです。

6. ミニBlocklyChainのプロセスアーキテクチャ

ミニBlocklyChainは、そのアーキテクチャを形成する3つのプロセスによって形成されており、1つ目はビジネスプロセス、2つ目は通信プロセス、3つ目は補完的なモジュールやBlockyCodeの「インテリジェントな契約」を作成するための開発環境です。

ビジネスプロセスは、公共またはプライベートネットワークのいずれかでトランザクションを作成する一連のルーチンを形成するブロックであり、プロセスのこのタイプは、ビジネスの計画であり、どのように、いつ、何を、誰が、どこで、より多くの属性が順序付けられ、計画され、分散されるように、各トランザクションの主な機能が送信され、受信され、格納され、および/または拒否されたが達成されるように。第1工程は、次のように4つの種類のブロックに分けて構成されています。

1. データ入力ブロック
2. データ処理ブロック
3. セキュリティブロックだ
4. モジュラーデータブロック（開発者
5. 通信ボット。

データ入力ブロックは、最小4つの入力パラメータ（ソースアドレス、デスティネーションアドレス、アクティブ、データ）を持つトランザクションを受信するものであり、変数" data"に応じてより多くを持つことができ、これは、サードパーティによって開発されたブロックであるか、またはヌル値（NULL）を持つことができます。

データプロセスブロックは、各トランザクションの物流、計算、データの正規化、論理的判断、フローチェックを開発します。これらのタイプのブロックは、ビジネスプロセスにインテリジェンスを与えるために使用され、その名前が示すように、主にすべてのタイプの情報を処理し、異なるタイプのデータから変換することに基づいています。

セキュリティブロックは情報を検証するために使用され、トランザクションがその起源から目的地まで変更されていないことを保証するために、それらは常にブロックチェーン技術で使用される様々なセキュリティアルゴリズムで処理され、最も使用されるツールの中では、（ハッシュ署名、デジタル署名、AESデータ暗号化、デジタルアドレスの作成と検証、他の人の間で）です。

モジュラーデータブロックは、これらのサードパーティによって開発されたブロックは、ミニBlocklyChainは、公共または民間かどうか、各セクターのニーズに応じて新しいブロックによって豊かになるようにモジュラー方式で作成されたことを覚えています。モジュールの作成方法の詳細については、Developers Annex を参照してください。

我々は以前にコメントしたようにミニ BlocklyChian のアーキテクチャは、その 2 番目のコンポーネントの通信のプロセスですが、これらのプロセスは、TCP とソケットの通信を介して通信チャネルを担当しているものを送信し、トランザクションを受信の柔軟性を与るために通信のいずれかの時点での使用される別の手段によって、最も使用されています。移動式インターネット、Wifi は現在コミュニケーション Bluetooth の媒体を含むために働いています。

通信ブロックは、基本的には転送チャネルに実装されたセキュリティとデータの交換に基づいており、これは、送信または受信したトランザクションが通過するさまざまな段階でSSH (Secure Shell) プロトコルを介して通信に基づいています。

これらのプロセスは、TCPと"Peer to Peer"と呼ばれる接続を介してすべてのノードの情報を更新する機能を持っているので、通信部分は基本的なものであり、通信に介入するブロックは、中間サーバーの介入なしでノード間の情報交換に基づいているので、各ノードは、障害のポイントが最小またはほぼNULLであるノードのネットワークを作成することができる独立したものになります。同様に、各ノードが独立していることで、ビジネスのニーズに応じて個別に、あるいは全体として意思決定を行うことができます。

「Peer to Peer」アーキテクチャは、作成することを決定したパブリックネットワークまたはプライベートネットワークを形成する3つの部分から構成されており、どちらの場合も通信チャネルはノードからノードへ暗号化されています。

モバイル機器（スマートフォン）やwifi間の通信のための第一のコンポーネントは、ノードや機器が世界中どこにいてもおかしくないネットワークを提供することであり、MiniBlocklyChainが実装されている通信ネットワークは「Tor」ネットワークである。

Torネットワークとは？ - (<https://www.torproject.org>)

"**Tor** (スペイン語で **Te Onion** ルーターの頭文字を取ったもの) は、その主な目的は、ユーザー間で交換されるメッセージのルーティングは、ユーザーの身元、すなわち IP アドレス (ネットワークレベルでの匿名性) を明らかにしないインターネット上に重畳された低遅延分散通信ネットワークの開発であり、さらに、それを介して移動する情報の完全性と秘密性を維持するプロジェクトです。"

2番目のコンポーネントとそれほど重要ではないタスクは、MiniBlocklyChainのすべてのノードがいつでも同じデータを持っているか、またはノード間でこのタスクを実行するために同期化されたデータベースとファイルを持っていることを取得することです"Syncing"が実装されます。

シンクティングネットワークとは？ - (<https://syncing.net>)

Syncingは、Windows、Mac、Linux、Android、Solaris、Darwin、BSDで利用できるフリーのオープンソースのピアツーピアファイル同期アプリケーションです。ローカルネットワーク上のデバイス間、またはインターネット上のリモートデバイス間でファイルを同期することができます。

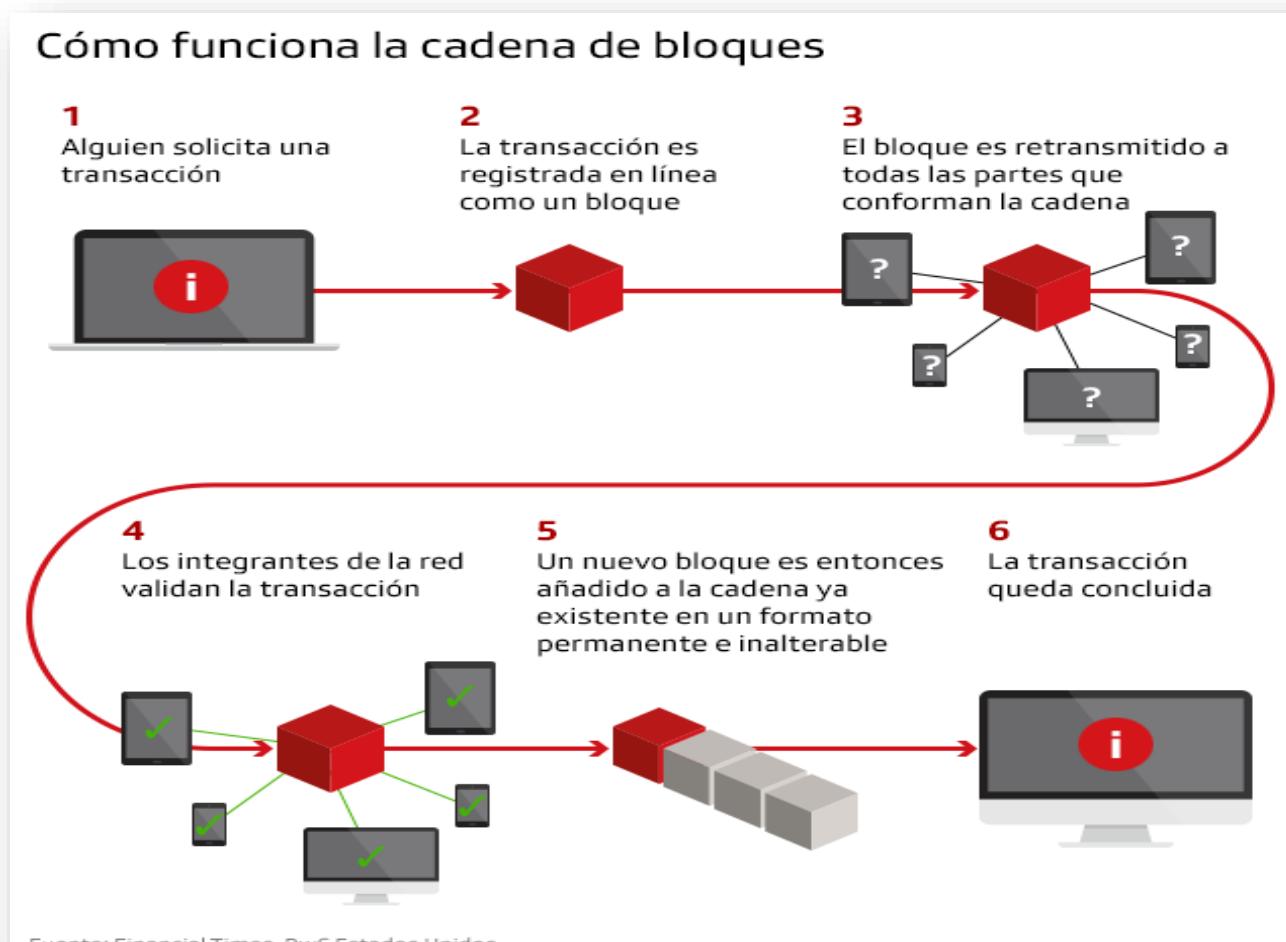
2つの前の通信コンポーネントに基づいて、我々は、ノード間の信頼ネットワークを開発するために開始することができますし、ユーザーや企業のあらゆるニーズを満たすために、ビジネスプロセスのバックボーンまたは"コア"となるデータ同期のレベルで。

通信ネットワークの3番目のコンポーネントはRedisデータベースで、受信した新しいトランザクションと送信した新しいトランザクションをリアルタイムで全ノードに通知します。

Redis DBのネットワークとは？ - (<https://redis.io>)

Redisはインメモリデータベースエンジンで、ハッシュテーブル(キー/値)での保存をベースにしていますが、オプションで耐久性のあるデータベースや永続的なデータベースとして使用することもできます。

7. BlocklyChain機能図（ミニBlocklyChain）。



8. ミニブロックリーコードとは？

Mini BlocklyCodeは、Java言語で作成されたプログラムで、ノードから独立したリポジトリに格納されています。これらのプログラムは、通常、インテリジェントコントラクトと呼ばれ、これらの条件が満たされたときに、承認や外部の人間の介入に依存することなく自動的に実行されるように論理的な条件で設計されています。

"スマートコントラクトとは、2つ以上の当事者（例えば、個人や組織）の間で登録された契約を促進し、確保し、執行し、実行するコンピュータプログラムのことです。このように、彼らは、交渉やそのような合意を定義する際に、いくつかの特定の条件が満たされた結果、特定の行動が起こるように、彼らを支援することになるだろう。

スマートコントラクトとは、どちらかの当事者やその代理人がコントロールしていないシステムに住み、他のコンピュータプログラムのif-then文のように動作する自動契約を実行するプログラムのことです。実物資産と相互作用する方法で行われているという違いがあります。人間の判断によらない事前にプログラムされた条件が発動されると、知的契約は対応する契約条項を実行する。

彼らは、従来の契約法よりも高いセキュリティを提供し、契約に関連する取引コストを削減することを目的としています。信頼（ビットコインなど）を必要としないシステムを介したデジタル価値の移転は、インテリジェントな契約を活用できる新たなアプリケーションへの扉を開きます。“

Mini BlocklyCodeは、Javaプログラミングの経験がある開発者を対象としています。Mini BlocklyChainでは、同一システムのセキュリティのために一部の種類のライブラリが制限されていますが、複数の当事者が作成または合意した契約価値を送受信するためのトランザクションを作成するためのライブラリを作成することができます。

すべてのMini BlocklyChainは、以下の前提条件を満たしています。

- ✓ 作成されたミニBlocklyChainは、システム上での実行ではなく、メッセージの実行のみに基づいていますが、これらのメッセージには、承認や物理的な契約の承認、物理的なアクションが含まれている場合があります。

- ✓ 作成されたMini BlocklyChainはすべて「監査人」通信ブロックを通過しなければならず、このブロックは悪意のあるコードや禁じられたコードを監視して、システム内の文章や不正な命令を見直す役割を担っています。
- ✓ すべてのMini BlocklyChainはソースノードのJVM上でのみ実行され、システム保護のためにプログラムはグローバルには実行されません。

詳細は、「開発者向け Mini BlocklyChain」の付録を参照してください。

9. ミニBlocklyChain通信ネットワークの設置

1. Mini SQLSyncネットワークのインストールと設定

このネットワークは、ノード間の「Peer to Peer」を介したメインネットワークと、Mini Sentinel RESTfulと呼ばれるバックアップネットワークによって形成されています。

我々は、RESTfulミニセンチネルのバックアップネットワークのインストールを開始しますが、このサービスは、ノードからのトランザクションを受信するのですが、このサービスは、後でRedisサービスにすべての暗号化されたトランザクションのキューを注入するコネクタを介して情報を変換するために決定された時間のためのトランザクションを格納することに基づいています。後でRedisデータベースサービスは、Mini BlocklyChainネットワークを統合するすべてのノードへの暗号化されたトランザクションキューのリリースをリアルタイムで実行します。

RESTfulタイプのサービスやデザインは、URLやインターネットアドレスを介してデータベースに情報を照会、修正、削除、挿入する簡単で簡単な方法です。大規模なスケーリングの必要性を持つ要件の使用のために、MySQL、Oracle、DB2または他の商用データベースとして別のタイプのデータベースを使用することができるようになります、単純に我々は、ミニBlocklyChainのコネクタを変更する必要がありますSQLite（無料版）のMini BlocklyChain DB他の（商用版）のコネクタのためのMini BlocklyChainのコネクタにSQLite（無料版）のコネクタを変更する必要があります。

重要な注意: RESTful Mini Sentinelの構成は、任意の種類のトランザクションをいつでも処理するわけではなく、ノードがトランザクション処理を適切に、そしてすべてのノードのために計画された同期化された時間で実行できるように、「情報を形づくる」サービスのみをフォーマットします。

RESTfulサービスのインストールはSQLiteデータベースをベースにしており、実用上はWindows環境でのインストールとなります。このモデルはLinuxタイプのOSでも簡単に複製することができます。

SQLite クエリや挿入のパフォーマンスやサポートを見直したい方は、これらのパフォーマンステストを参考にしてみてください。

<https://www.sami-lehtinen.net/blog/sqlite3-performance-testing>

<https://sqlite.org/src4/doc/trunk/www/lsmperf.wiki>

RESTful Mini Sentinelバックアップネットワークのインストール。今回はWindows10のパソコンにインストールして設定していくが、Ubuntu 18.09のLinux環境にも簡単にインストールできました。

要件。

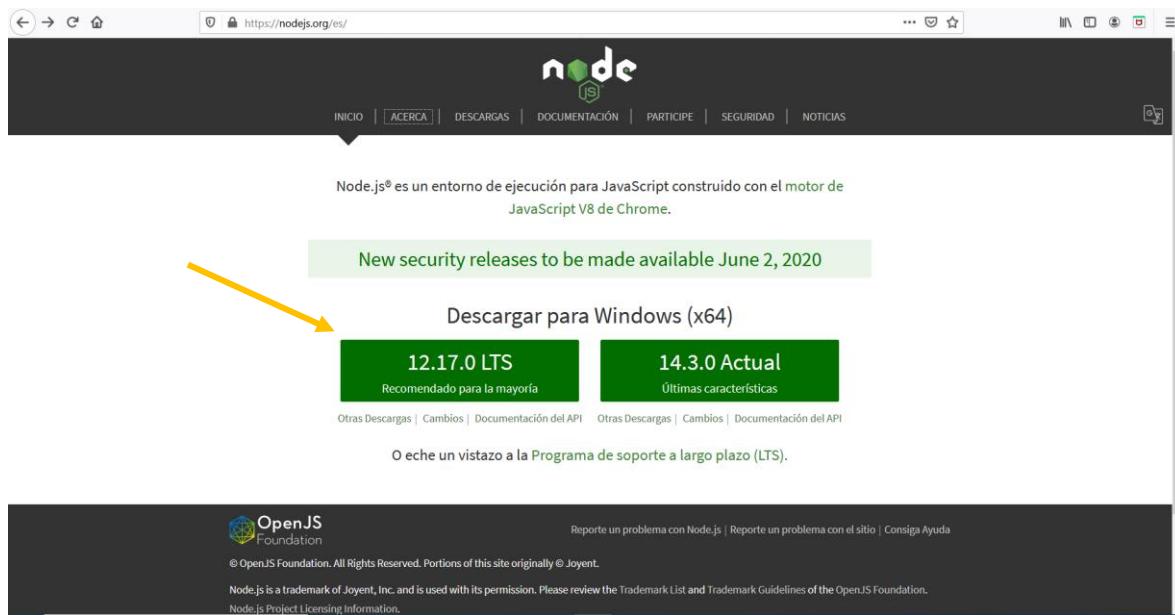
- ✓ RESTful モードの SQLite データベースサーバ (<https://github.com/olsonpm/sqlite-to-rest>)。
- ✓ SQLite-Redis Sentinel コネクタ
- ✓ Redis データベースサーバーをマスタースレーブモードで使用しています (<https://redis.io/topics/replication>)

RESTfulモードでのSQLiteデータベースサーバーのインストール

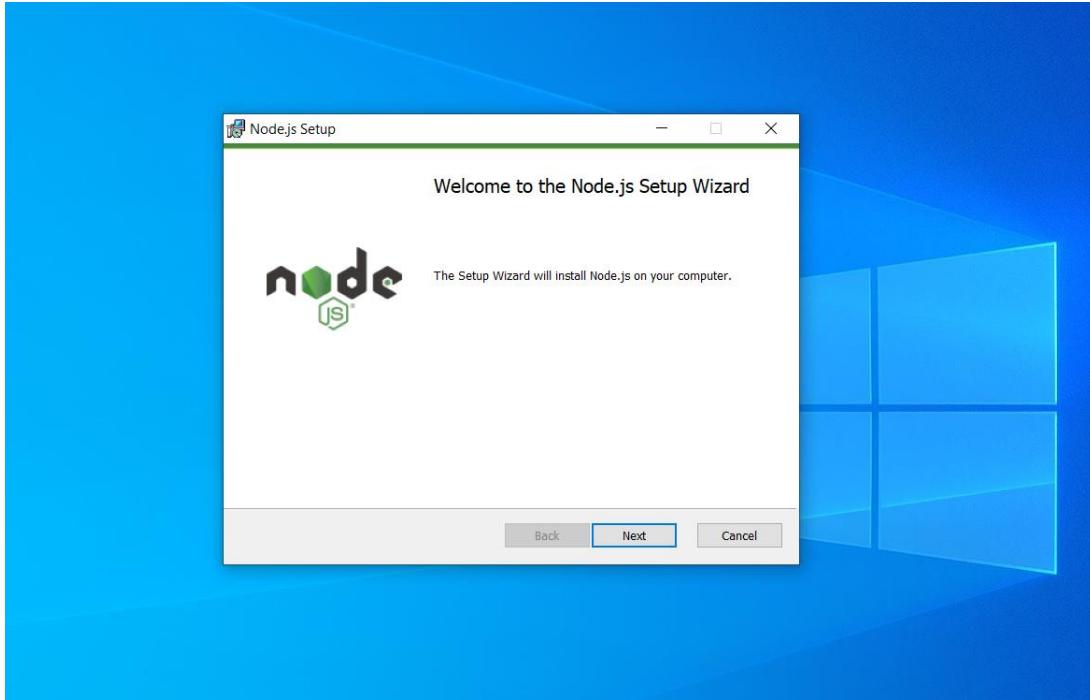
インストールには、以下のソフトウェアパッケージ、Nodejs、sqlite3、Git Bash for Windowsから始める必要があります。

Nodejsを入手するために、彼らの公式サイト <https://nodejs.org/es/> を参考にして、"Recommended for Most"バージョンを選択しました。

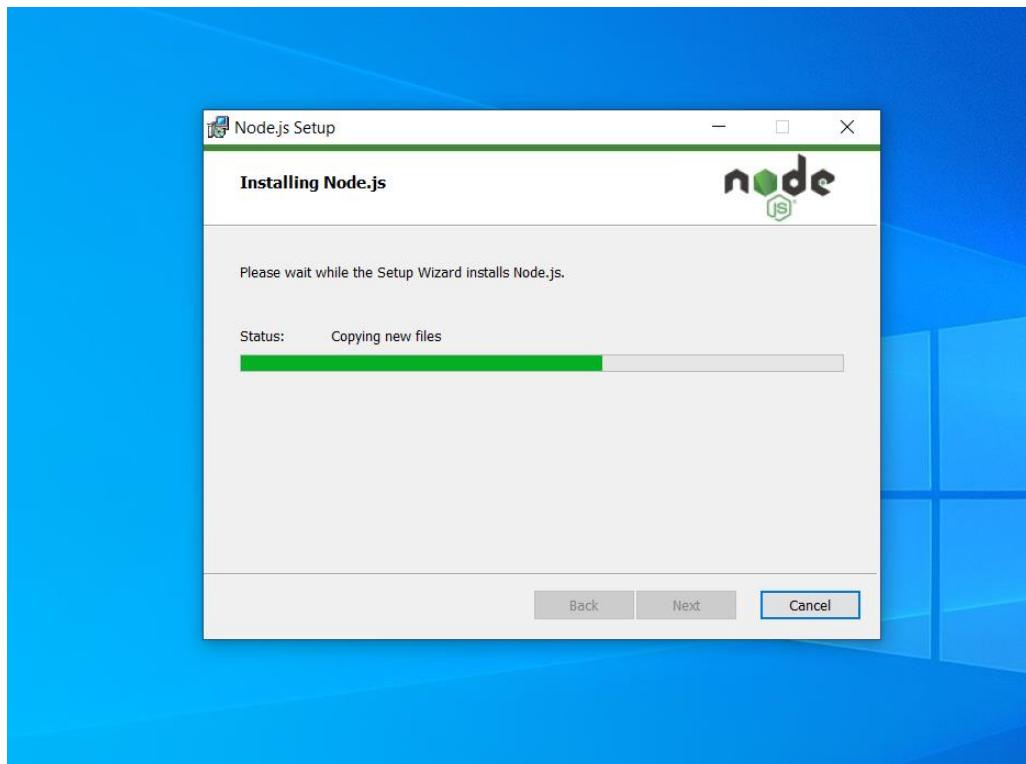
ミニBlocklyChain - DIY - 'Do It Yourself'

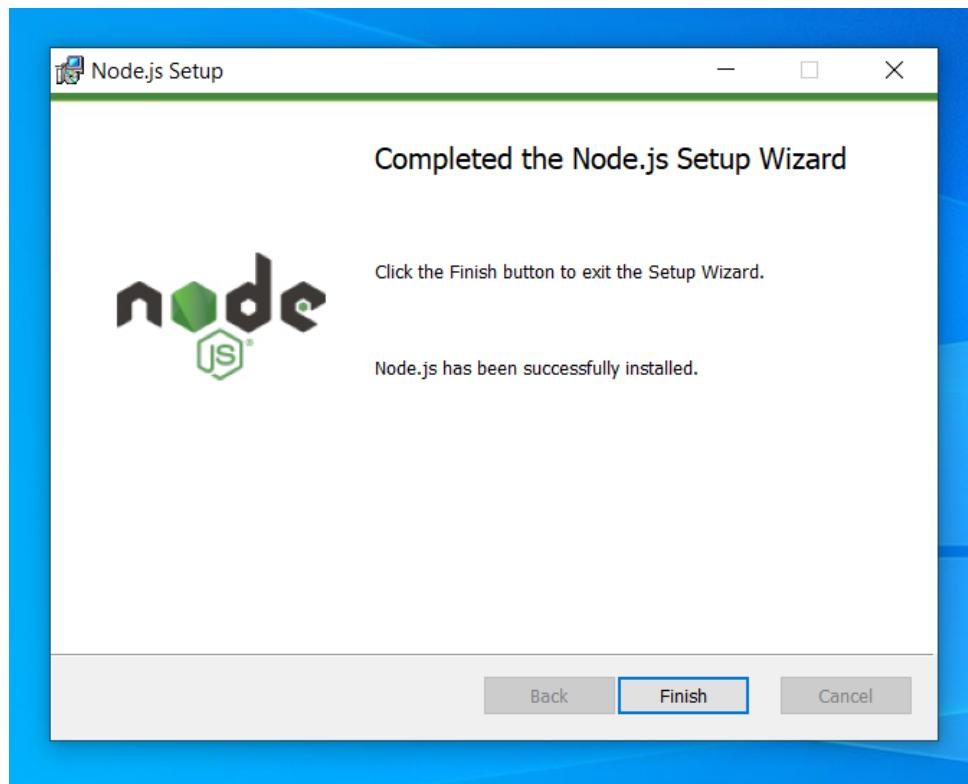


拡張子.msiのファイルをダウンロードした後、ダブルクリックしてインストールします。



私たちはデフォルトでインストールを実行し、あなたが尋ねる任意の追加オプションを選択することなく、単に「次へ」をクリックします。





Nodejsのインストールが終ったので、SQLiteデータベースのインストールを続けます。

私たちは彼らの公式サイト、<https://www.sqlite.org/index.html> に行き、あなたが "ダウンロード" する部分をクリックします。



What Is SQLite?

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured database engine in the world. SQLite is built into all mobile phones and most computers and is used by people around the world every day. [More Information...](#)

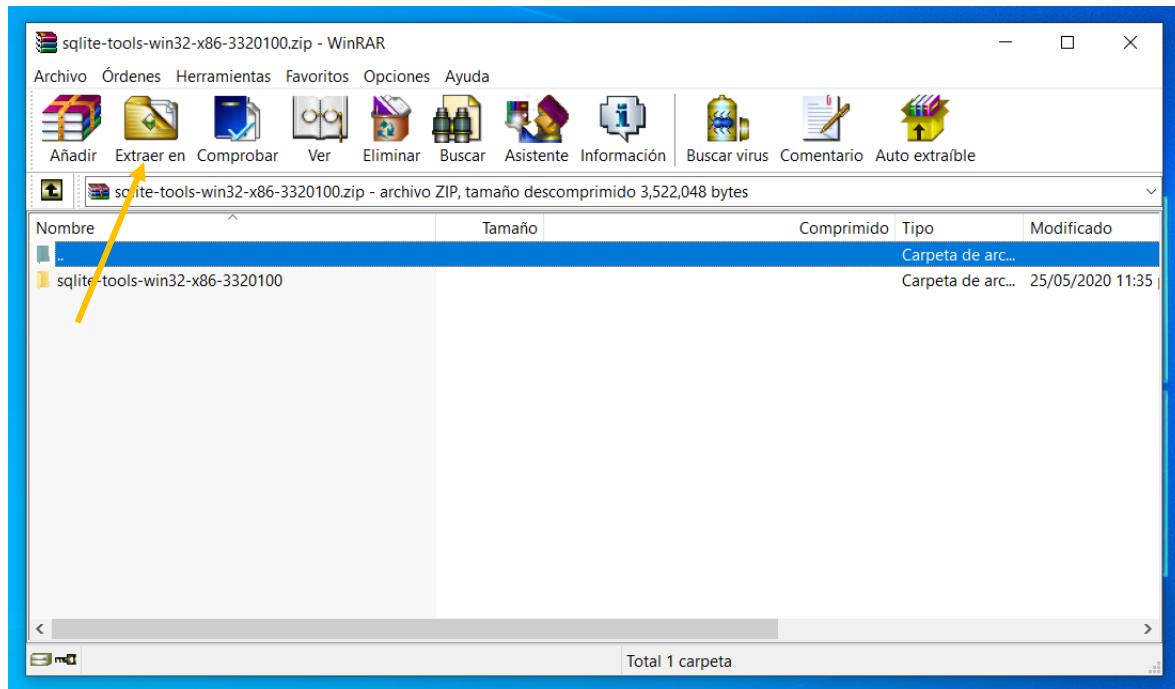
The SQLite [file format](#) is stable, cross-platform, and backwards compatible and the developers pledged to support it until 2050. SQLite database files are commonly used as containers to transfer rich content between systems. The file format for data [\[4\]](#). There are over 1 trillion (1e12) SQLite databases in active use [\[5\]](#).

SQLite [source code](#) is in the [public-domain](#) and is free to everyone to use for any purpose.

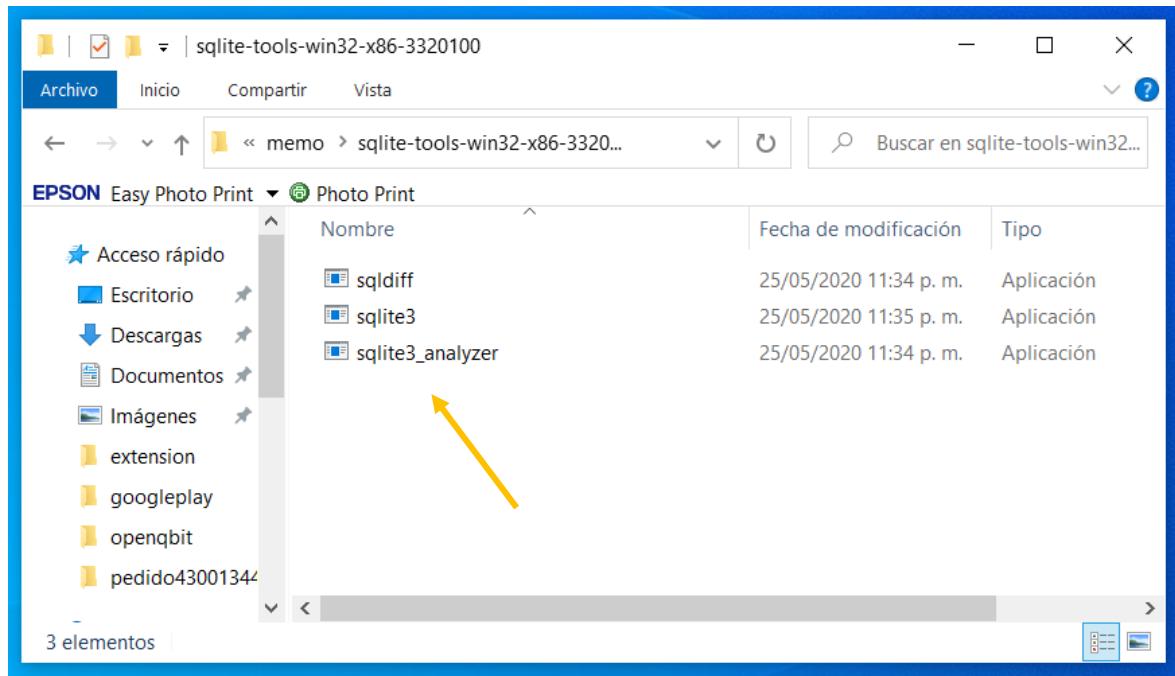
Latest Release

[Version 3.32.1 \(2020-05-25\)](#). [Download](#) [Prior Releases](#)

次に、"Precompiled Binaries for Windows"と書かれたオプションを選択し、ソフトウェアバージョン "sqlite-tools-win32-x86-3320100.zip"をクリックしてダウンロードが終了したら、簡単な方法でファイルを解凍します。

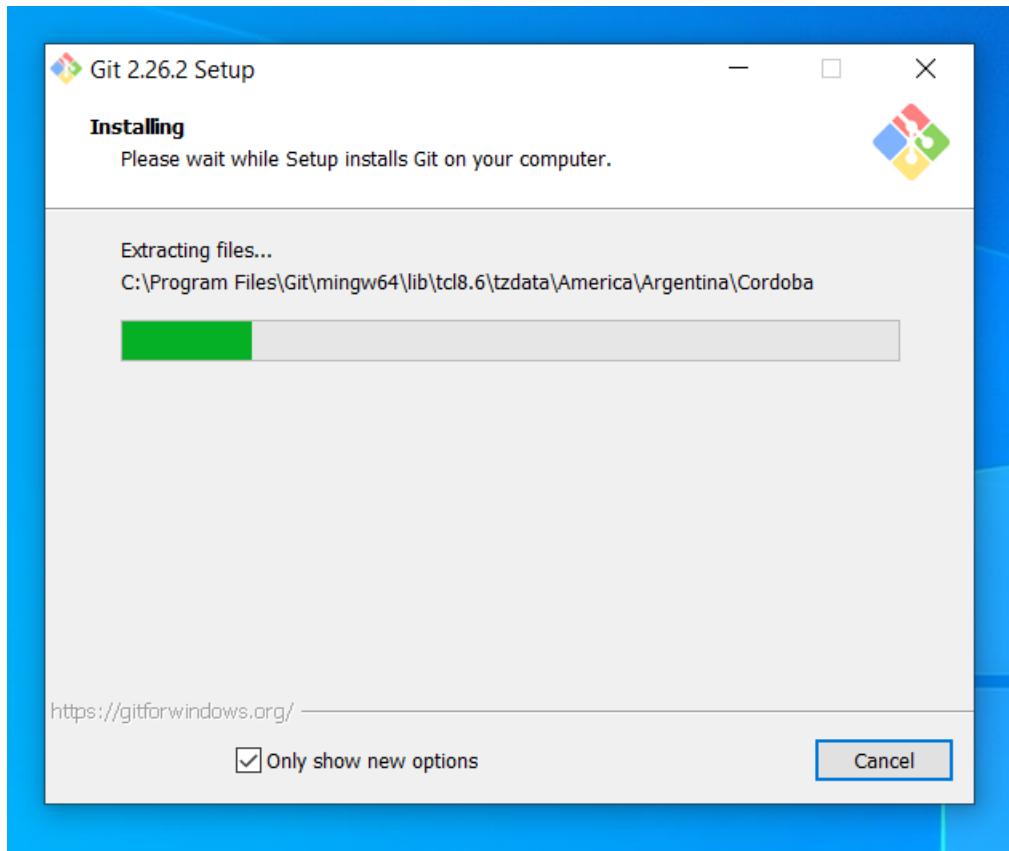


これを解凍した後、3つのファイルができます。これらは後に使用するSQLiteデータベースを作成するための環境です。



まずはLinuxライクなターミナルエミュレータであるGit Bashをインストールして、RESTfulバックアップサービスを適切に実行できるようにしていきます。

我々は彼らの公式サイトからそれをダウンロードし、<https://gitforwindows.org/>、[それが示すデフォルトのオプション](#)でインストールを続行します。



Windows 10マシンにnodejs、sqlite、Git Bashをインストールしたので、SQLiteデータベースを RESTfulモードでサポートする環境をインストールしていきます。

SQLiteにRESTfulの機能を付与するソフトをダウンロードした時点でこのためには、次のサイトに行かなければなりません。<https://github.com/olsonpm/sqlite-to-rest> この中で、緑色の右ボタン「クローンまたはダウンロード」をクリックして、オプション「ZIPをダウンロード」を選択すると、私たちのコンピュータにソフトウェアがダウンロードされます。

No description or website provided.

automatic-api

Branch: dev ▾ New pull request

Find file

Clone or download ▾

Clone with HTTPS ⓘ
Use Git or checkout with SVN using the web URL.
<https://github.com/olsonpm/sqlite-to-rest>

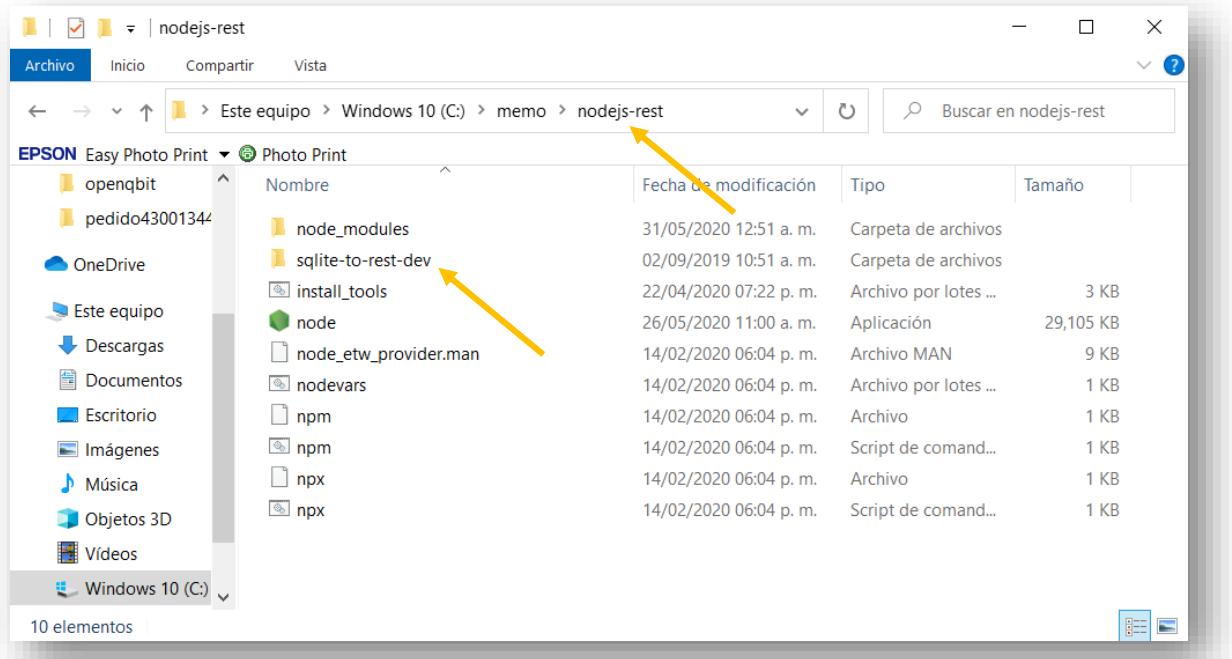
Open in Desktop Download ZIP

9 months ago 9 months ago 9 months ago

File	Description	Modified
olsonpm	add prettier and eslint	9 months ago
bin	add prettier and eslint	9 months ago
cli/commands	add prettier and eslint	9 months ago
docs	add prettier and eslint	9 months ago
lib	add prettier and eslint	9 months ago
tests	add prettier and eslint	9 months ago
.gitignore	add prettier and eslint	9 months ago

コンピュータにダウンロードした後、nodejsプログラムをインストールしたディレクトリやフォルダの中で解凍していき、"sqlite-to-rest-dev"という名前のディレクトリになります。

NOTE: sqlite-to-rest-devディレクトリがnodejsがインストールされたディレクトリ内にあることが重要です。



すべてをインストールした後、RESTfulバックアップ環境でノードのトランザクションを保存するために使用するSQLiteデータベースの設定に進みます。

テーブルのデザインとデータ構造。CMDコマンドリストでオンラインで実行するコマンド

```
sqlite3 op.sqlite3 "CREATE TABLE trans (id integer primary key, addro, addrd, value);"
```

```
sqlite3 op.sqlite3 "CREATE TABLE sign (id integer primary key, trans_id references trans (id), sign, hash);"
```

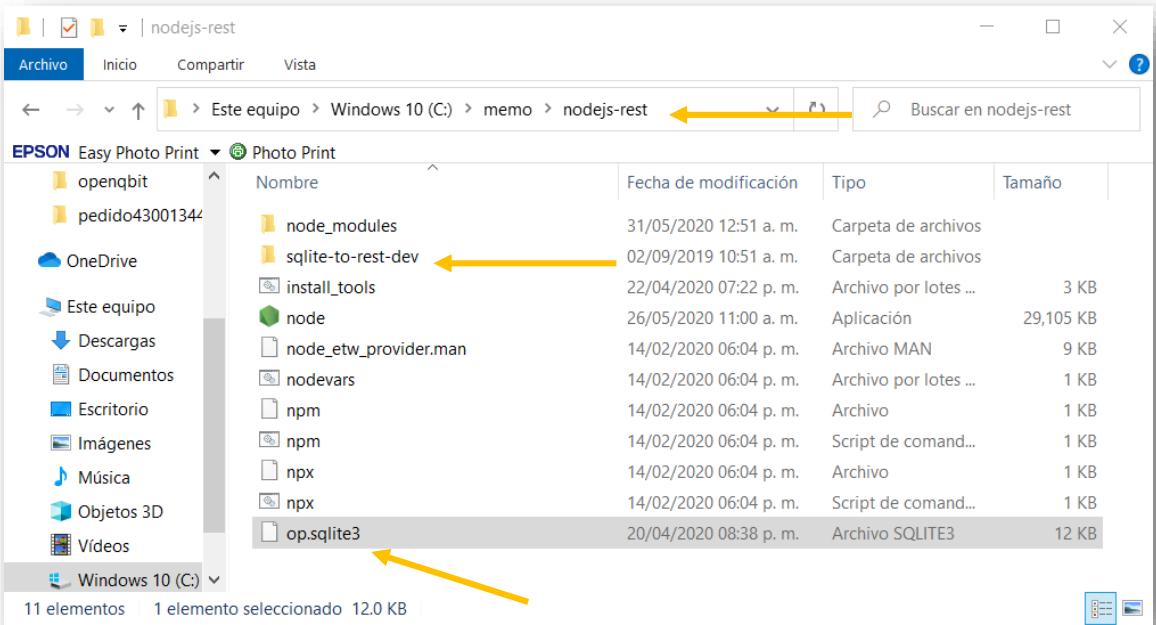
```
C:\memo\sqlite-tools-win32-x86-3320100>sqlite3 op.sqlite3 "CREATE TABLE trans(id integer primary key, addro, addrd, value);"
C:\memo\sqlite-tools-win32-x86-3320100>sqlite3 op.sqlite3 "CREATE TABLE sign(id integer primary key, trans_id references trans (id), sign, hash);"
C:\memo\sqlite-tools-win32-x86-3320100>dir
El volumen de la unidad C es Windows 10
El n m o de serie del volumen es: E019-5C05

Directorio de C:\memo\sqlite-tools-win32-x86-3320100

31/05/2020  02:29 a. m.    <DIR>        .
31/05/2020  02:29 a. m.    <DIR>        ..
31/05/2020  02:29 a. m.      12,288 op.sqlite3
25/05/2020  11:34 p. m.     516,608 sqldiff.exe
25/05/2020  11:35 p. m.     972,800 sqlite3.exe
25/05/2020  11:34 p. m.    2,032,640 sqlite3_analyzer.exe
                           4 archivos      3,534,336 bytes
                           2 dirs   137,272,078,336 bytes libres

C:\memo\sqlite-tools-win32-x86-3320100>
```

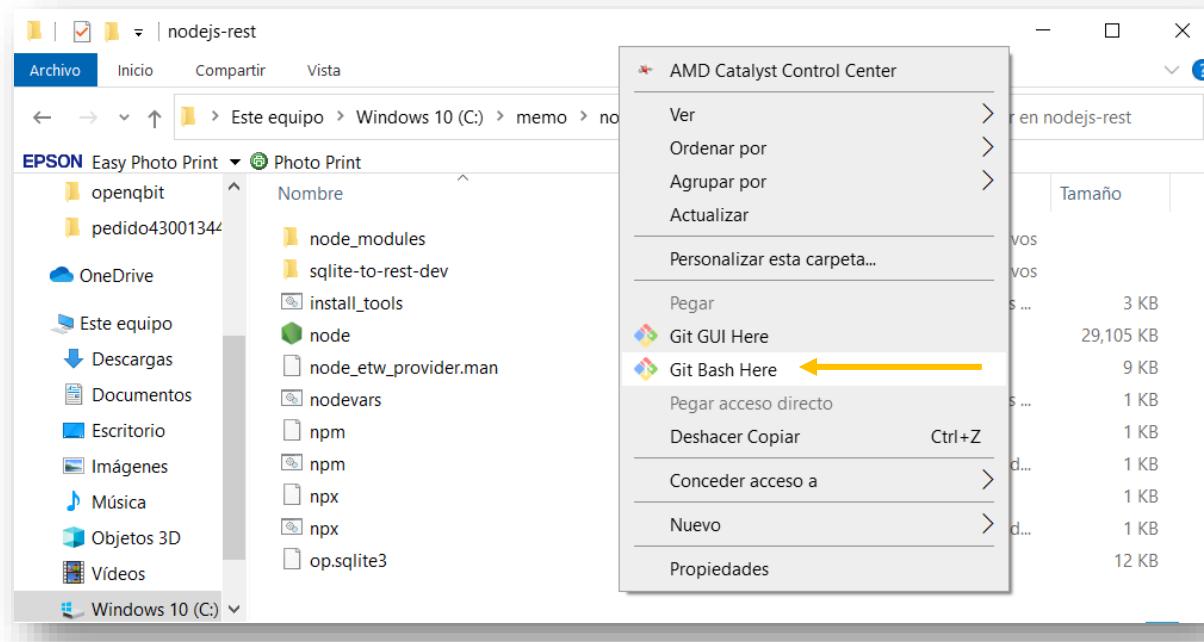
op.sqlite3データベースを作成した後、nodejsがインストールされたディレクトリにコピーを作成する必要があります。nodejsディレクトリでも"sqlite-to-rest-dev"のコピーでなければなりません。

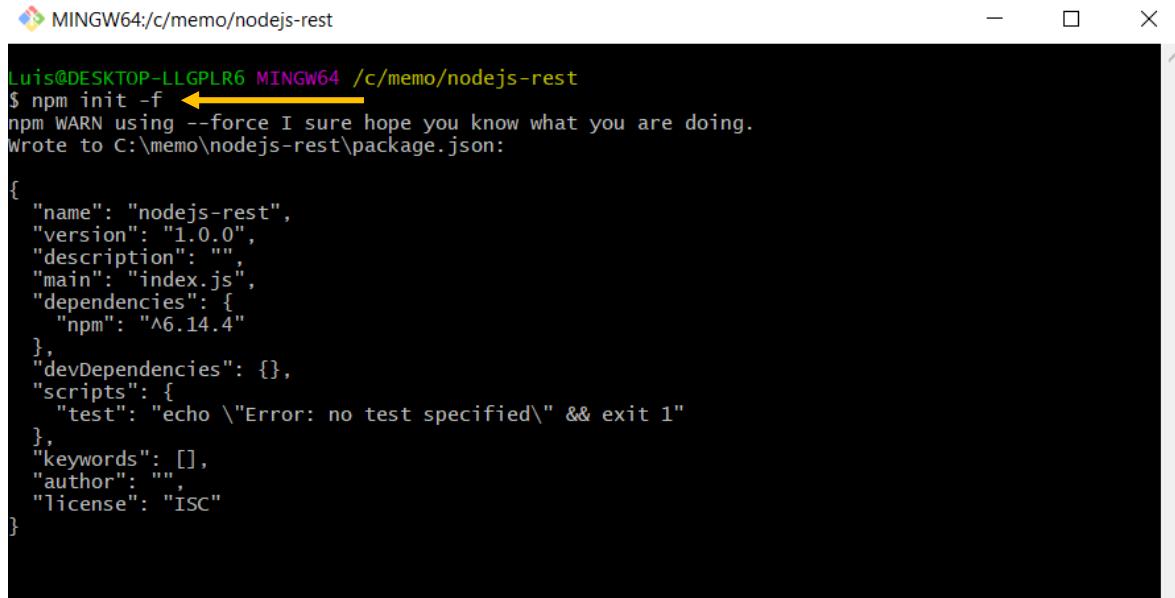


nodejsがインストールされているフォルダと、"sqlite-to-rest-dev"ソフトウェアがインストールされているフォルダに自分自身を配置します。ポインターのあるフォルダをポイントして右クリックしてメニューを表示させ、「Git Bash」を書いてある場所を選択してターミナルを開きます。

新しく開いたGit Bashターミナルで、以下のコマンドを実行します。

```
npm init -f
```

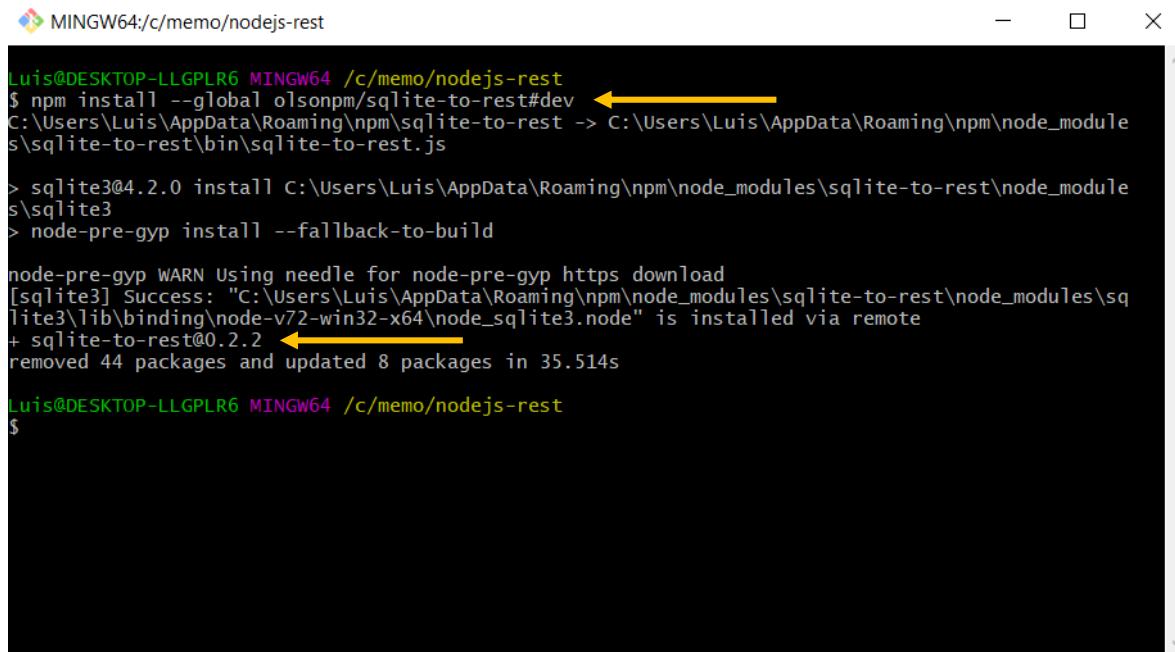




```
Luis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
$ npm init -f
npm WARN using --force I sure hope you know what you are doing.
Wrote to C:\memo\nodejs-rest\package.json:

{
  "name": "nodejs-rest",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "dependencies": {
    "npm": "^6.14.4"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

\$ npm install --global olsonpm/sqlite-to-rest#dev



```
Luis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
$ npm install --global olsonpm/sqlite-to-rest#dev
C:\Users\Luis\AppData\Roaming\npm\sqlite-to-rest -> C:\Users\Luis\AppData\Roaming\npm\node_modules\sqlite-to-rest\bin\sqlite-to-rest.js

> sqlite3@4.2.0 install C:\Users\Luis\AppData\Roaming\npm\node_modules\sqlite-to-rest\node_modules\sqlite3
> node-pre-gyp install --fallback-to-build

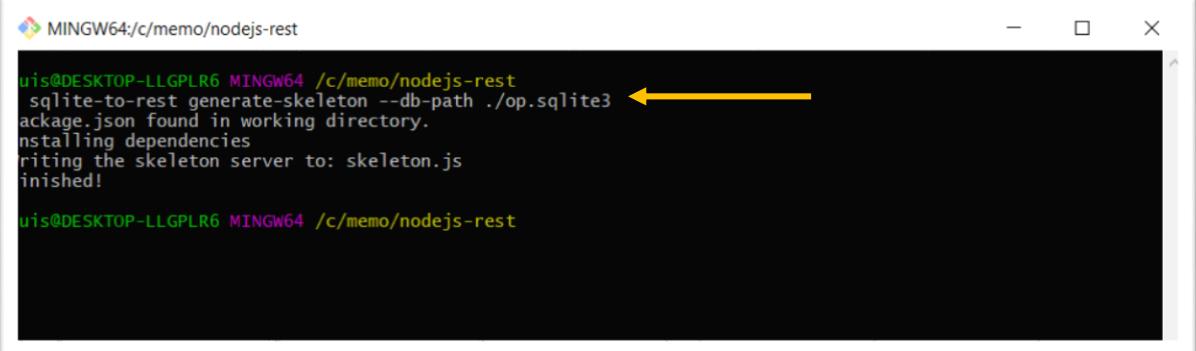
node-pre-gyp WARN Using needle for node-pre-gyp https download
[sqlite3] Success: "C:\Users\Luis\AppData\Roaming\npm\node_modules\sqlite-to-rest\node_modules\sqlite3\lib\binding\node-v72-win32-x64\node_sqlite3.node" is installed via remote
+ sqlite-to-rest@0.2.2
removed 44 packages and updated 8 packages in 35.514s

Luis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
$
```

コマンド実行後、「sqlite-to-rest」パッケージのインストールが表示されます。

以前に作成したop.sqlite3ベースでSQLite用のRESTful環境を生成しました。

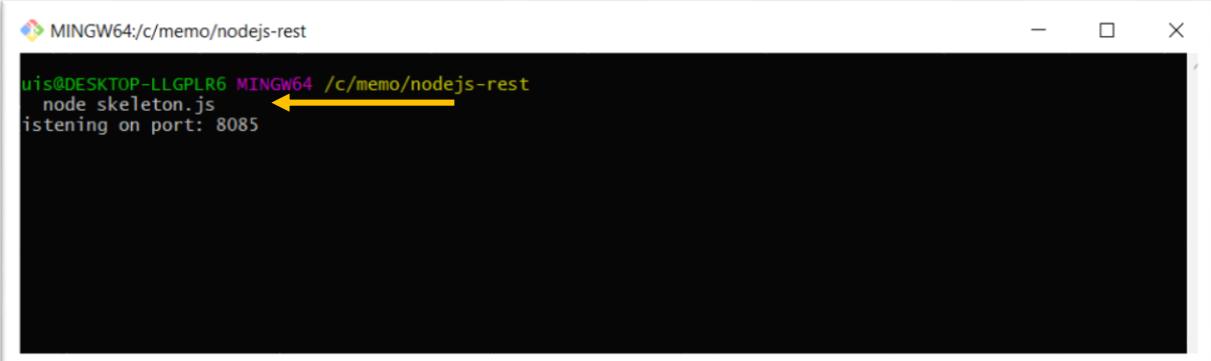
コマンドを実行します \$ sqlite-to-rest generate-skeleton --db-path ./op.sqlite3



```
uis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
sqlite-to-rest generate-skeleton --db-path ./op.sqlite3 ←
package.json found in working directory.
Installing dependencies
Writing the skeleton server to: skeleton.js
finished!

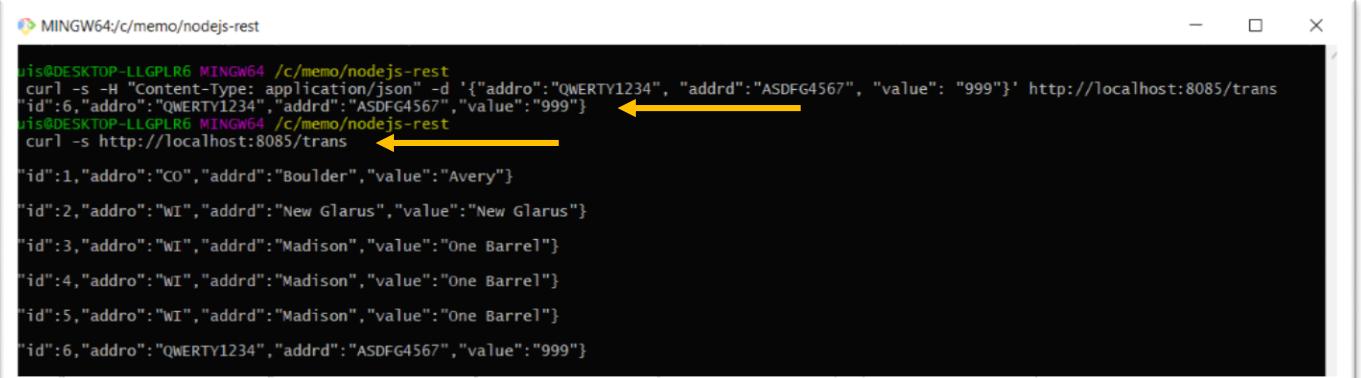
uis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
```

デフォルトポート8085でRESTful SQLiteサービスを起動しました。次のコマンドを実行します： \$ node skeleton.js



```
uis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
node skeleton.js ←
listening on port: 8085
```

RESTfulサービスにデータを追加して、データをクエリしてテストしました。



```
uis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
curl -s -H "Content-Type: application/json" -d '{"addr": "QWERTY1234", "addrd": "ASDFG4567", "value": "999"}' http://localhost:8085/transactions ←
"id": 6, "addr": "QWERTY1234", "addrd": "ASDFG4567", "value": "999"
uis@DESKTOP-LLGPLR6 MINGW64 /c/memo/nodejs-rest
curl -s http://localhost:8085/transactions ←
{"id": 1, "addr": "CO", "addrd": "Boulder", "value": "Avery"}
{"id": 2, "addr": "WI", "addrd": "New Glarus", "value": "New Glarus"}
{"id": 3, "addr": "WI", "addrd": "Madison", "value": "One Barrel"}
 {"id": 4, "addr": "WI", "addrd": "Madison", "value": "One Barrel"}
 {"id": 5, "addr": "WI", "addrd": "Madison", "value": "One Barrel"}
 {"id": 6, "addr": "QWERTY1234", "addrd": "ASDFG4567", "value": "999"}
```

SQLite RESTful サービスをテストするには、以下のコマンドを使用します。

データベースop.sqlite3内にあるテーブルtransにデータを挿入します。

```
$ curl -s -H "Content-Type: application/json" -d '{"addr": "QWERTY1234",  
"addrd": "ASDFG4567", "value": "999"}' http://localhost:8085/trans
```

trans テーブルのすべてのデータのクエリを作成します。

```
$ curl -s -H 'range: rows=0-2' http://localhost:8085/trans
```

すべての RESTful サービス（データの問い合わせ、挿入、更新、および/または削除）の詳細な使用方法については、付録「Restful SQLite GET/POST コマンド」を参照してください。

注意：以前のインストールでは、すべてのクエリは「localhost」というアドレスのURLで行われていましたが、サーバーがパブリックIP（インターネット）やプライベートIP（Wifi）で公開されている場合は問題なく動作します。SQLite RESTfulサービスとMini BlocklyChainを形成する通信ネットワークノード間の通信テストを行う際にテストを行います。

バックアップネットワークサービスのためのRedisデータベースのインストールは、Windows用のRedisソフトウェアをダウンロードするには、我々は、サイトに移動する必要があります
<https://github.com/microsoftarchive/redis/releases/tag/win-3.2.100> とZIPパッケージを選択します。

Pre-release

3.2.100

enricogior released this on 1 Jul 2016 · 1210 commits to 3.0 since this release

This is the first release of Redis on Windows 3.2.

This release is based on antirez/redis/3.2.1 plus some Windows specific fixes. It has passed all the standard tests but it hasn't been tested in a production environment.

Therefore, before considering using this release in production, make sure to test it thoroughly in your own test environment.

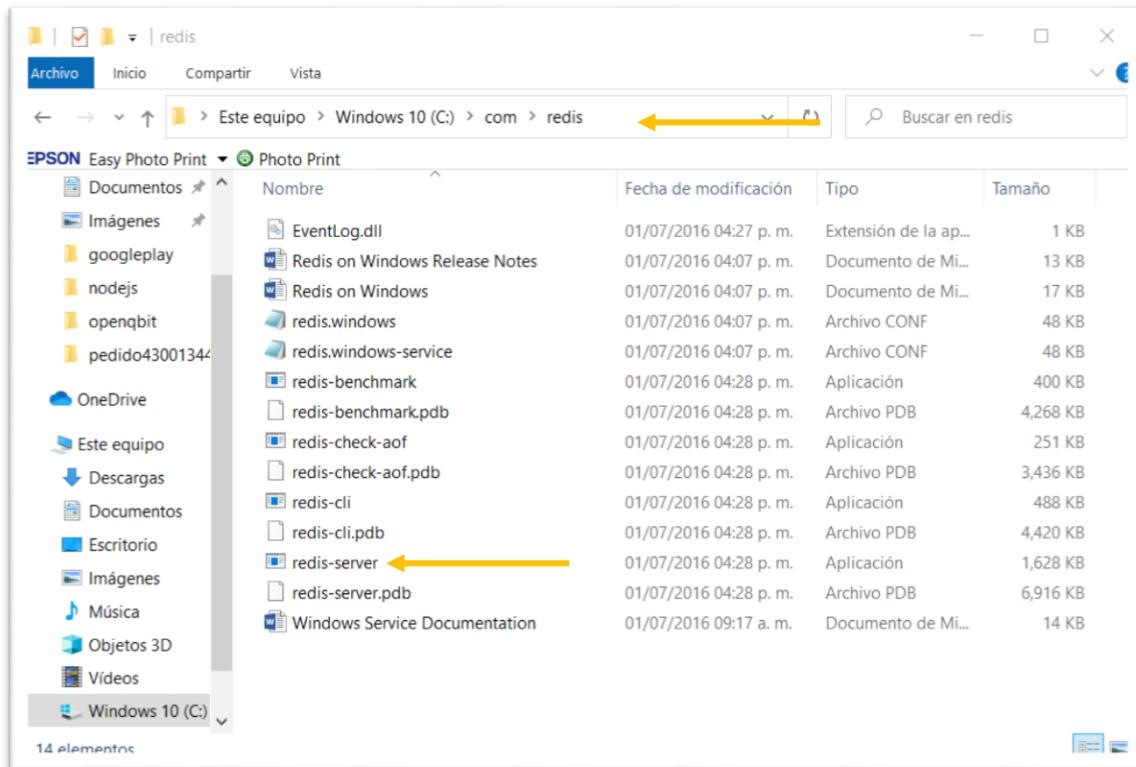
See the [release notes](#) for details.

Assets 4

Redis-x64-3.2.100.msi	5.8 MB
Redis-x64-3.2.100.zip	4.98 MB
Source code (zip)	
Source code (tar.gz)	

インストールを見つけるために、我々はWindows内で"redis"と呼ばれるディレクトリを作成し、ZIP拡張子を持つファイルをダウンロードし、我々は以前に作成されたディレクトリ内でそれを解凍し、我々はすでにインストールが完了したredisを持っています。

「redis-server」というコマンドをダブルクリックしてサーバーを起動し、インストールをテストします。



「redis-server」コマンドを実行すると、WindowsのCMDコマンドターミナルでデフォルトのポート6379

```

C:\com\redis>redis-server.exe
[10192] 31 May 13:50:45.340 # Warning: no config file specified, using the default config. In order to specify a config
file use C:\com\redis\redis-server.exe /path/to/redis.conf

Redis 3.2.100 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 10192

http://redis.io

[10192] 31 May 13:50:45.355 # Server started, Redis version 3.2.100
[10192] 31 May 13:50:45.355 * The server is now ready to accept connections on port 6379

```

でサーバーが動作しているのがわかります。

このテストでは、Windows 10用のRedis 3.2.100のバージョンを使用しています。Linuxオペレーティングシステムの場合、バージョン6.0.4（2020年5月リリース）を使用していますが、ミニBlocklyChainの構成では、Windowsバージョンは必要な機能を十分に備えています。

実行を停止するには、Ctrl + Cのキーの組み合わせを行います。私たちは、RedisとMini SQLSync通信ネットワークノード間の設定でWindows 10用のRedis 3.2.100データベースを構成するために進みます。

マスターは設定しているWindows 10用のRedisサーバーで、これがリアルタイムでデータを複製し、全てのノード（携帯電話）に同じ情報を同期させます。これらのノードには、**スレーブモード**でRedisサーバーがインストールされることになります。

この時点で、私たちがインストールして設定しているバックアップネットワークがどのように動作するかを確認し始めます。この構成は、リアルタイムですべてのノードと通信するために私たちを提供します、それは、任意のノードが"トランザクションキュー"情報を処理することができるのと同じ確率を持っているように、すべてのノードへの情報転送の平等性で、ノードによって処理される新しいトランザクションキューがあるときに、ネットワークのすべてのノードに通信するのに役立ちます。

SQLite-Redis Sentinelコネクタの設定を開始します。

このコネクタはJava言語で開発されたプログラムで、その名の通りSQLiteとRedis（マスター）データベースを接続します。

コネクタの機能は、SQLiteからRedis(マスター)に情報(トランザクション)を送信することであり、これにより「トランザクションキュー」をノード(スレーブ)に送信します。

SQLite-Redis SentinelコネクタのJavaコードの詳細については、Annex "SQLite-Redis Java Code Connector"を参照してください。

Windows 10用のRedis（マスター）データベースredis.confファイルの設定。

以下の変更や指示をファイルに追加し、変更を保存してRedisサーバーを起動します。

`tcp-keepalive`の設定を見つけて、コメントにあるように60秒に設定することから始めます。これにより、Redisがネットワークやサービスの問題を検出するのに役立ちます。

`ティーシーピーキーパライブ60`

`requirepass` ディレクティブを見つけて、強力なパスフレーズで設定します。Redisのトラフィックをセドパーティから保護する必要がありますが、これはRedisに認証を提供します。Redisは高速であり、境界線上のパスフレーズの試みを評価しないので、ブルートフォースの試みから保護するために、強力で複雑なパスフレーズを選択してください。

`requirepass type_your_network_master_password`

の例を示します。

`requirepass FPqwedsLMdf76ass7asddfd2g45vBN8ty99`

最後に、利用シーンに応じて調整したいオプション設定がいくつかあります。

Redisがいっぱいになると自動的に最も古い鍵と最も使用されていない鍵を入れたくない場合は、鍵の自動削除を無効にすることができます。

最大記憶領域ポリシーノエヴィクション

耐久性の保証を強化するために、アドオンでファイルの永続性を有効にすることができます。これにより、システム障害が発生した際に、より大きなファイルやややや遅いパフォーマンスを犠牲にしても、データの損失を最小限に抑えることができます。

付帯性

```
appendfilename "redis-staging-ao.aof"
```

変更を保存し、Windows 10用のRedisサービスを再起動するために進み、Ctrl + Cキーで停止し、Windows CMDコマンドラインを再度実行します。

```
C:\redis_redis_directory redis_server
```

この例では、(Slave)ノードが接続されていることがわかります。

```

:\\com\\redis> redis-server redis.conf
1:C 31 May 2020 23:44:56.633 # o000o000o000 Redis is starting o000o000o000
1:C 31 May 2020 23:44:56.634 # Redis version=5.0.7, bits=64, commit=00000000, modified=0,
id=51, just started
1:C 31 May 2020 23:44:56.634 # Configuration loaded
1:M 31 May 2020 23:44:56.635 * Increased maximum number of open files to 10032 (it was originally set to 1024).

Redis 3.2.100 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 51

http://redis.io

1:M 31 May 2020 23:44:56.648 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.51:May 31 May 2020 23:44:56.648 # Server initialized
1:M 31 May 2020 23:44:56.649 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
1:M 31 May 2020 23:44:56.669 * DB loaded from append only file: 0.020 seconds
1:M 31 May 2020 23:44:56.669 * Ready to accept connections
1:M 31 May 2020 23:49:57.013 * 10 changes in 300 seconds. Saving...
1:M 31 May 2020 23:49:57.031 * Background saving started by pid 82
2:C 31 May 2020 23:49:57.052 * DB saved on disk
1:M 31 May 2020 23:49:57.133 * Background saving terminated with success
1:M 31 May 2020 23:50:24.600 * Replica 192.168.1.68:6379 asks for synchronization
1:M 31 May 2020 23:50:24.602 * Full resync requested by replica 192.168.1.68:6379
1:M 31 May 2020 23:50:24.602 * Starting BGSAVE for SYNC with target: disk
1:M 31 May 2020 23:50:24.619 * Background saving started by pid 83
3:C 31 May 2020 23:50:24.642 * DB saved on disk
1:M 31 May 2020 23:50:24.670 * Background saving terminated with success
1:M 31 May 2020 23:50:24.689 * Synchronization with replica 192.168.1.68:6379 succeeded

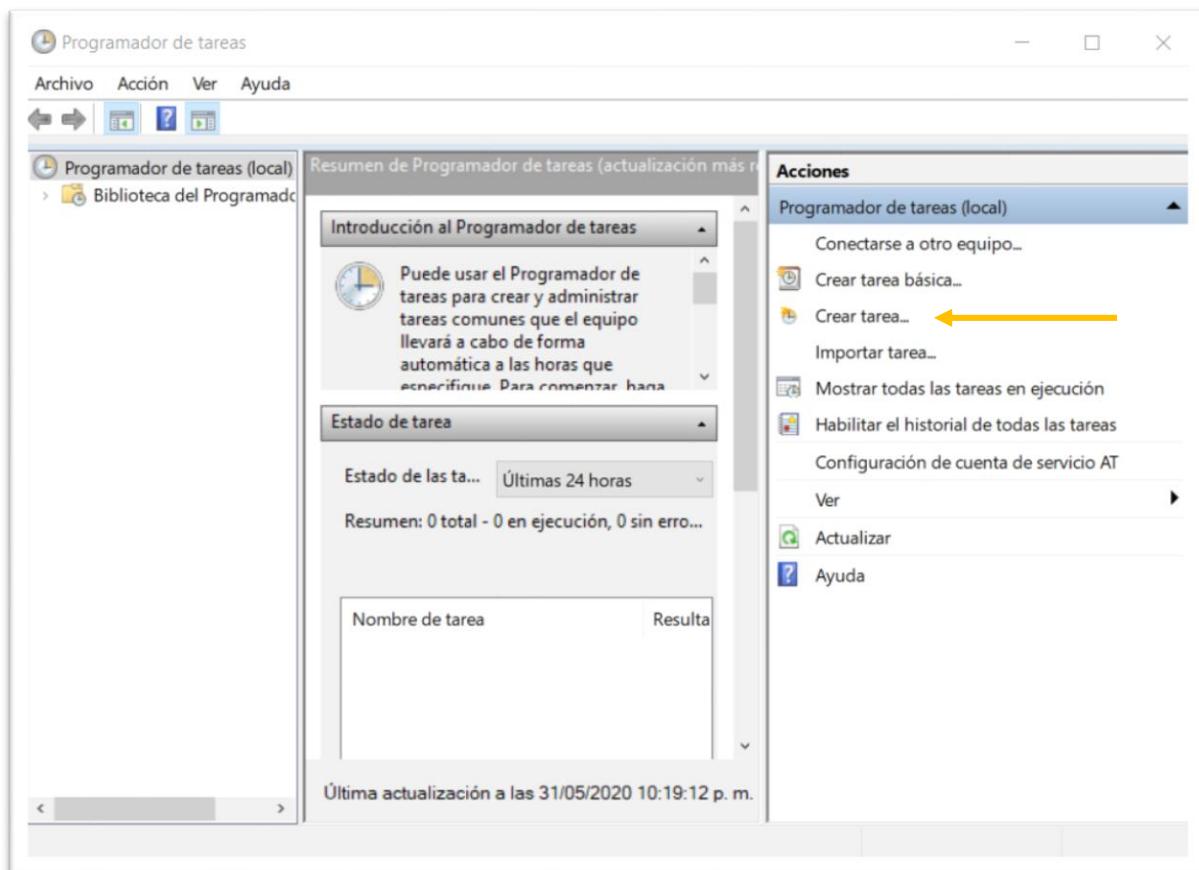
```

デフォルトのポート6379でIP 192.168.1.68のノードを同期したことがわかります。

今、私たちは、Window 10オペレーティングシステムでSQLite-Redis Sentinelコネクタを自動的に実行するタスクをスケジュールする必要があります。これをWindows10ツールで行うには、左下の「タスクスケジューラ」と入力して実行します。



ここでは、コネクタの実行を含めた「Create task」というタスクを新たに作成します。



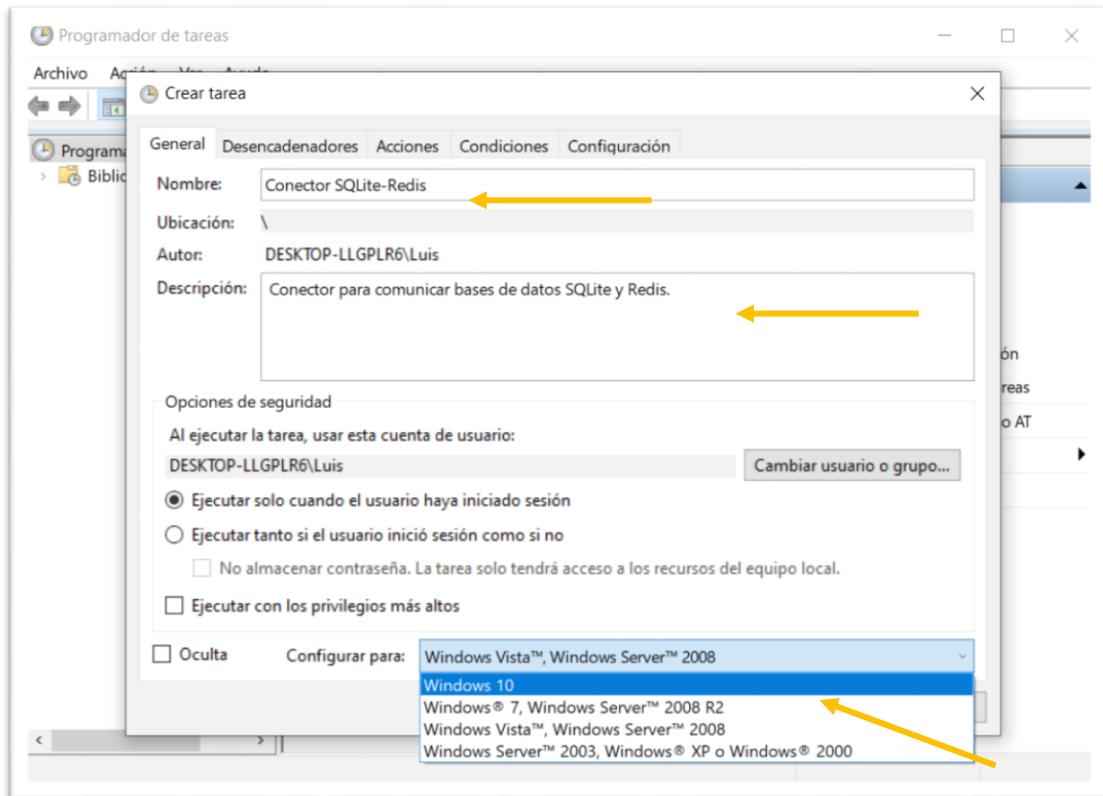
タスクの作成"をクリックすると、"一般"タブで以下のパラメータを与えなければならない追加ウイン

ドウが開きます。

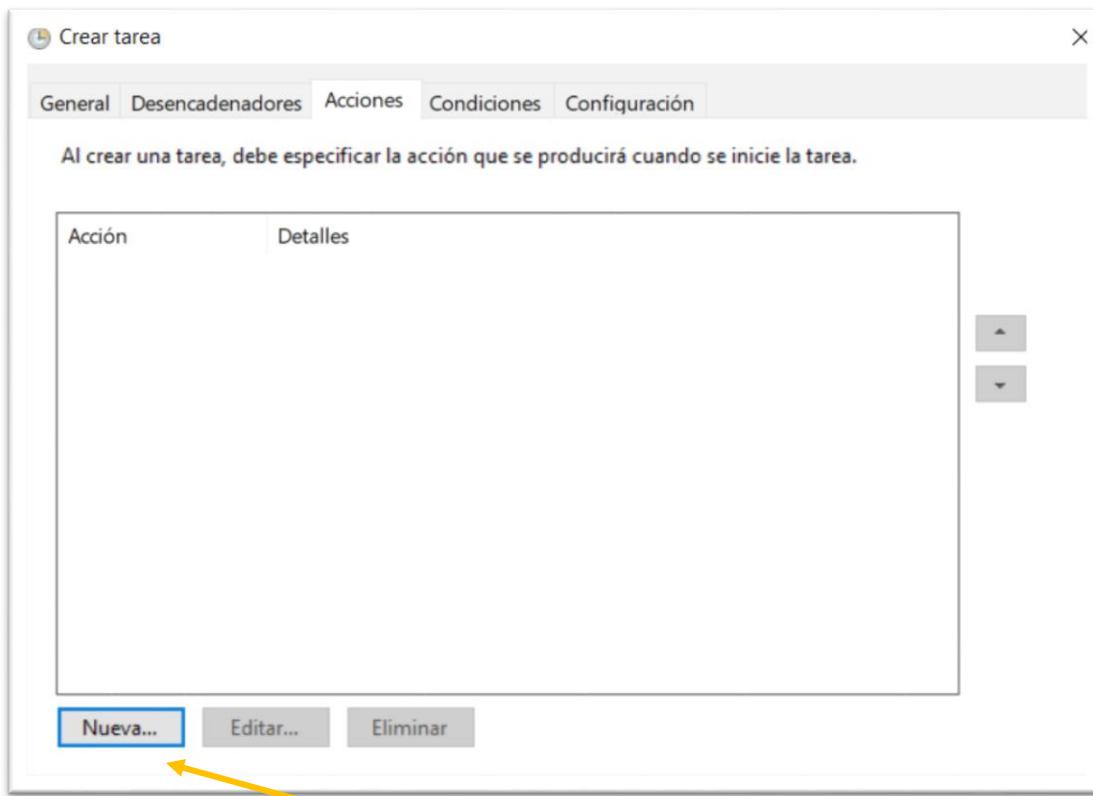
名前: タスク名

説明: オプション

設定対象: select_operating_system_windows_version



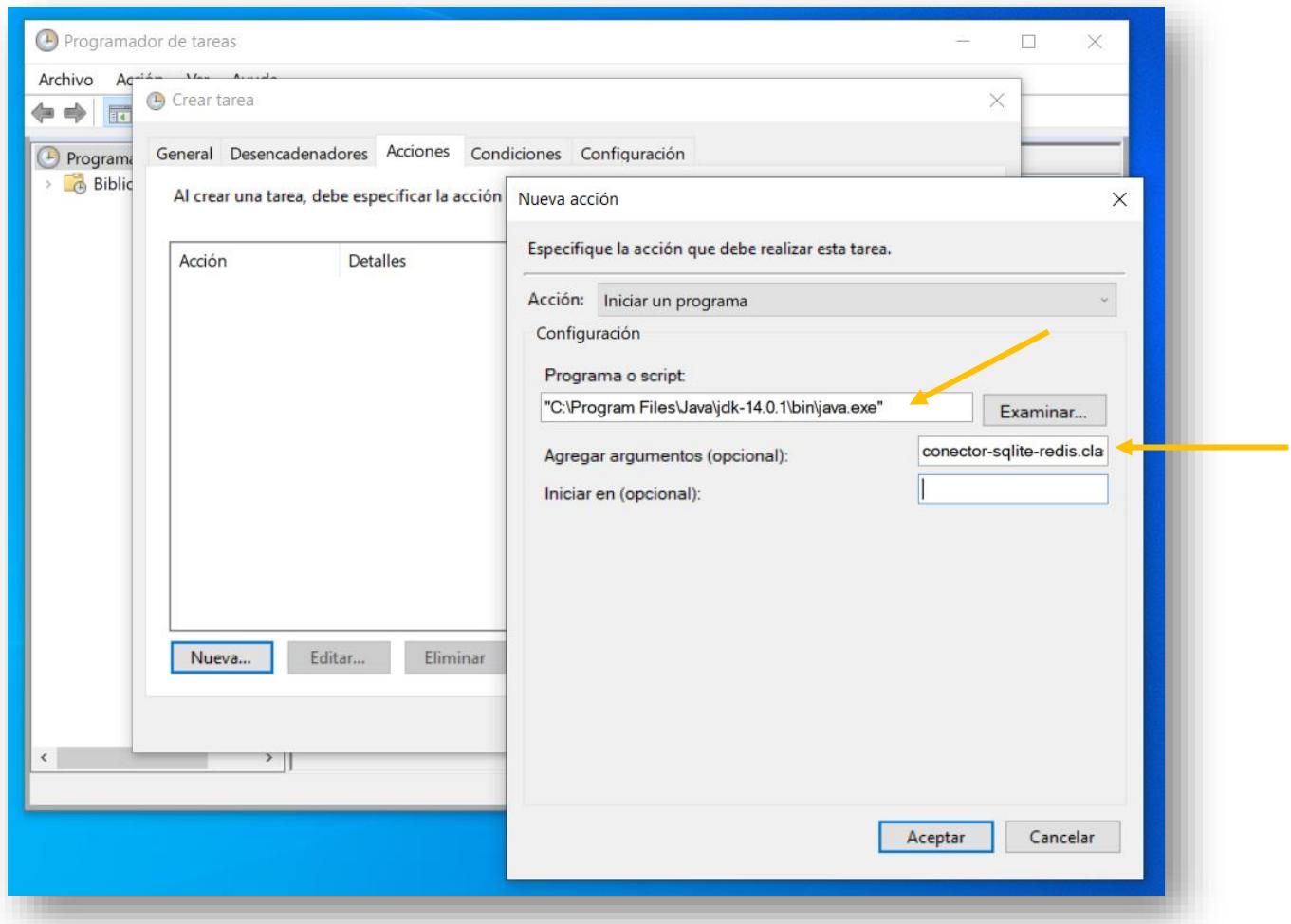
アクション」タブをクリックして「新規」ボタンをクリックして変更します。



以下のパラメータを与えます。

プログラムまたはスクリプト: connector_path

引数の追加: コネクタ名



上記のパラメータは、コネクタの位置によって異なる場合があります。主な考え方は、通常コマンドライン上で行われるような `connector-sqlite-redis-v1.class` プログラムの実行です。

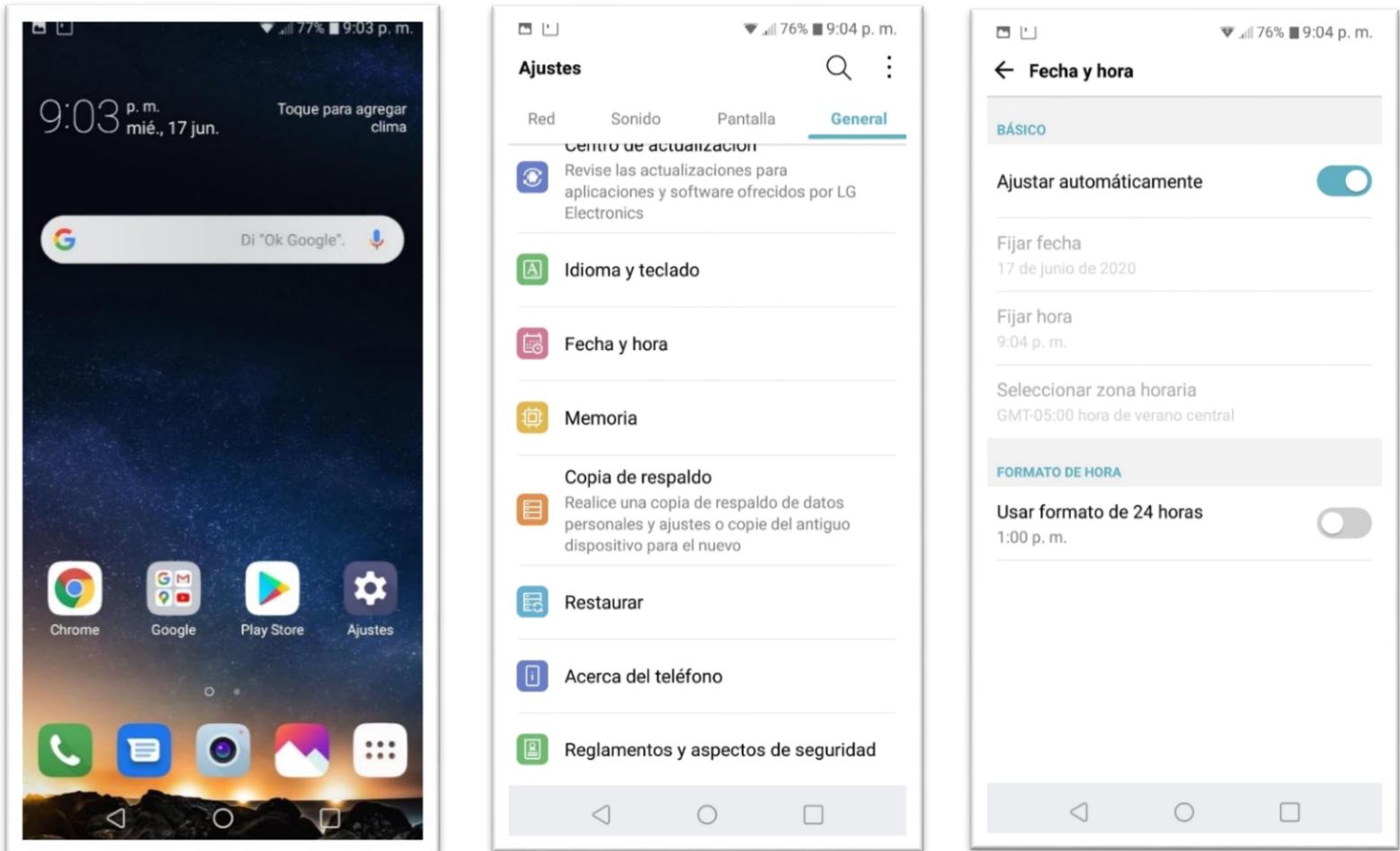
`C:\jdk_directory java connector-sqlite-redis-v1`

最後に、"Triggers"タブを選択しなければなりません。ここでは、タスクを実行させたい時間(日、時間、分)のパラメータを与えます。

10. システムノード（携帯電話）での同期を分秒で行う。

分・秒の部分を中心に全ノードを同期させることが非常に重要です。これは、すべてのノードで同時に実行されるように、ローカルシステムに確立されるタスクマネージャに依存するので、トランザクションキューの決定された時間に送信または公開が行われているときに、すべてのノードが同期される必要がありますので、これは、すべてのノードは、トランザクションキューを処理することができ、システムのブロックチェーンに追加するには、新しいブロックを生成することができるようになります。ノードを同期させるには、2つのオプションがあります。

ノードの同期の最初のオプションは、我々は簡単な方法でそれを行うことができるようになります。私たちのデバイスでは、Androidシステムを含む内部オプションを介してですが、我々は、**設定**の



一部に移動する必要があ

ります > 日付と時刻 > 調整を自動的に

前の設定では、世界のどの国もシステムのすべてのノードがすでに同期されているかに関係なく、分と秒で同期されていますが、それぞれの地理的な領域に基づいている可能性がありますが、何が変わるかは時間ですが、分と秒に応じて同期される必要があります。これは、すべてのノードのcronツールを使用して、スケジュールベースでタスクのプロセスを実行するのに十分です。

これは、"Peer to Peer"通信ネットワークを使用する場合に便利です。バックアップネットワークを使用する場合には、トランザクションキーの配布はクライアント-サーバモデルで行われ、サーバはcronツールを制御するものであるため、このプロセスは適用されません。

参考：<https://appinventor.mit.edu/explore/blogs/karen/2016/08.html>

2番目のオプションは、拡張機能 (ConnectorSSHClient) を使用して Curl コマンドを実行する外部 API を使用することです。

NTP(Network Time Protocol)の外部サービスを利用する場所です。

<http://worldtimeapi.org/>

さて、世界中にあるNTPサーバーから時間を取得する方法を見てみましょう。

拡張子(ConnectorSSHClient)を持つクエリの例。

```
curl "http://worldtimeapi.org/api/timezone/America/Mexico_City"
```

タームマックス端子との接続を行いました。



Curl コマンドを実行します。



Curl コマンドの結果は JSON 形式で、次のような結果になることを考慮しなければなりません。

```

abbreviation: "CDT"
client_ip: "200.77.16.151"
datetime: "2020-06-18T14:16:57.750466-05:00"
day_of_week: 4
day_of_year: 170
dst: true
dst_from: "2020-04-05T08:00:00+00:00"
dst_offset: 3600
dst_until: "2020-10-25T07:00:00+00:00"
raw_offset: -21600
timezone: "America/Mexico_City"
unixtime: 1592507817
utc_datetime: "2020-06-18T19:16:57.750466+00:00"
utc_offset: "-05:00"
week_number: 25

```

また、結果は以下のように、データをフォーマットせずにJSON形式にしたり、リニア形式のJSONにしたりすることもできます。

```
{
  "時刻": "CDT", "client_ip": "200.77.16.151", "datetime": "2020-06-18T14:19:07.216800-05:00", "day_of_week": 4, "day_of_year": 170, "dst": true, "dst_from": "2020-04-08:00", "dst_offset": 3600, "dst_until": "2020-10-25T07:00:00+00:00", "raw_offset": -21600, "timezone": "America/Mexico_City", "unixtime": 1592507947, "utc_datetime": "2020-06-18T19:19:07.216800+00:00", "utc_offset": "-05:00", "week_number": 25}
}
```

前述の2つの方法のいずれかで、「JSONTOOLS」のような既存のJSON拡張機能で情報をフィルタリングするか、テキスト処理でApp Inventorのフィルタを使用して、各システムの必要性に応じて時、日、秒のみを取得する必要があります。結果を処理した後、論理的な比較を行うことができ、その比較に基づいて、我々は後で我々は各ノードでその設定を参照してくださいする"cron"サービスですにプログラムされたタスクを実行することができます。

JSONTOOLS拡張機能のリファレンスです。

<https://thunkableblocks.blogspot.com/2017/07/jsontools-extension.html>

ここでは、ノードの時刻を同期させるための2つのオプションを確認しました。引き続き、各ノードに「cron」サービスを設定していきます。

Androidシステム（システムノード）でのCRONサービスによる自動タスクスケジューリング実行の設定

まず、自動スケジューラの仕組みを理解する必要があります。

cronサービスは通常、すべてのシステムにプリインストールされており、自動的に実行されるすべてのタスクがスケジューリングされているcrontabと呼ばれるファイルの中にはあります。

crontabファイルを\$ crontab -eで編集します。

m h dom mon dow ユーザコマンド

どこで。

- **m** はスクリプトが実行される分に対応し、値は 0 から 59 までです。
- **h** 正確な時間は、フォーマットは24時間であり、値は0から23に行く、0 12:00の真夜中である。
- **dom** は月の曜日を指定します。例えば、15日ごとに実行したい場合は 15 を指定します。
- **ダウ**は曜日を意味し、それは数字（0から7まで、0と7は日曜日です）または英語で一日の最初の3文字にすることができます：MON、TUE、WED、THU、FRI、SAT、SUN。
- **user** はコマンドを実行するユーザを定義します。root であっても、スクリプトを実行する権限を持ったユーザであれば別のユーザであっても構いません。
- **コマンド**とは、実行されるスクリプトのコマンドまたは絶対パスのことで、例: /home/user/scripts/update.sh、スクリプトを呼び出す場合は実行可能でなければなりません。

それを明確にするために、いくつかのcronタスクの例を説明します。

15 10 * * * ユーザ /home/user/scripts/update.sh

オープンキュービットドットコム

毎日午前10時15分にupdate.shスクリプトを実行します。

```
15 22 * * * * ユーザー /home/user/scripts/update.sh
```

毎日午後10時15分にupdate.shスクリプトを実行します。

```
00 10 * * 0 root apt-get - and update User root
```

毎週日曜日の午前10時に更新を実行します。

```
45 10 * * sun root apt-get - and update
```

ルートユーザーは毎週日曜日（日）の午前10時45分にアップデートを実行します。

エディタに変更を保存し、これでcronサービスの設定を完了しました。システムにcronがインストールされていない場合は、以下のコマンドで実行できます。

```
apt install cron
```

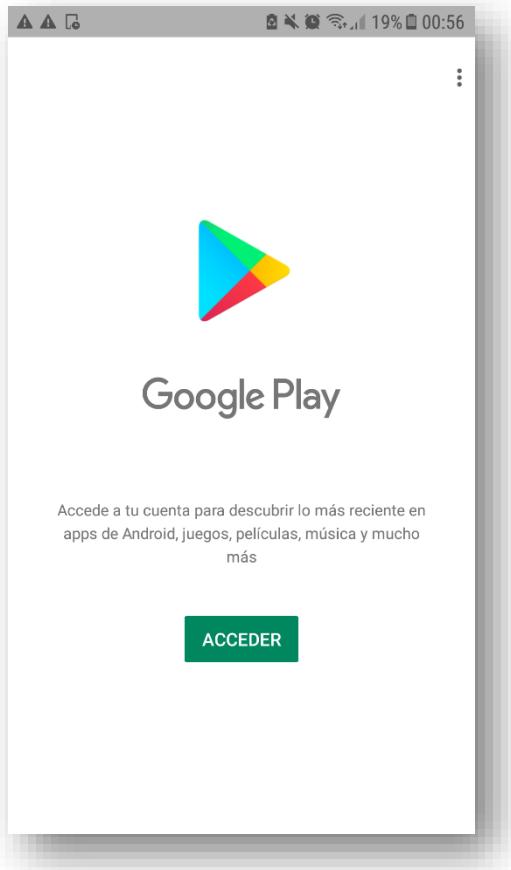
2. ネットワークノードのインストールと設定 - 携帯電話。

まずはMini BlocklyChainを使用するノードの通信ネットワークから見ていきましょう。

最初に我々は、すべてのAndroidシステムは、セキュリティとツールの柔軟性のためにLinuxに基づいているので、Linux環境を必要とし、我々は、通信ネットワークをインストールする環境を含む"Termux"端末を使用します。 TermuxはLinuxエミュレータで、ノード間の通信ネットワークを作成するために必要なパッケージをインストールします。

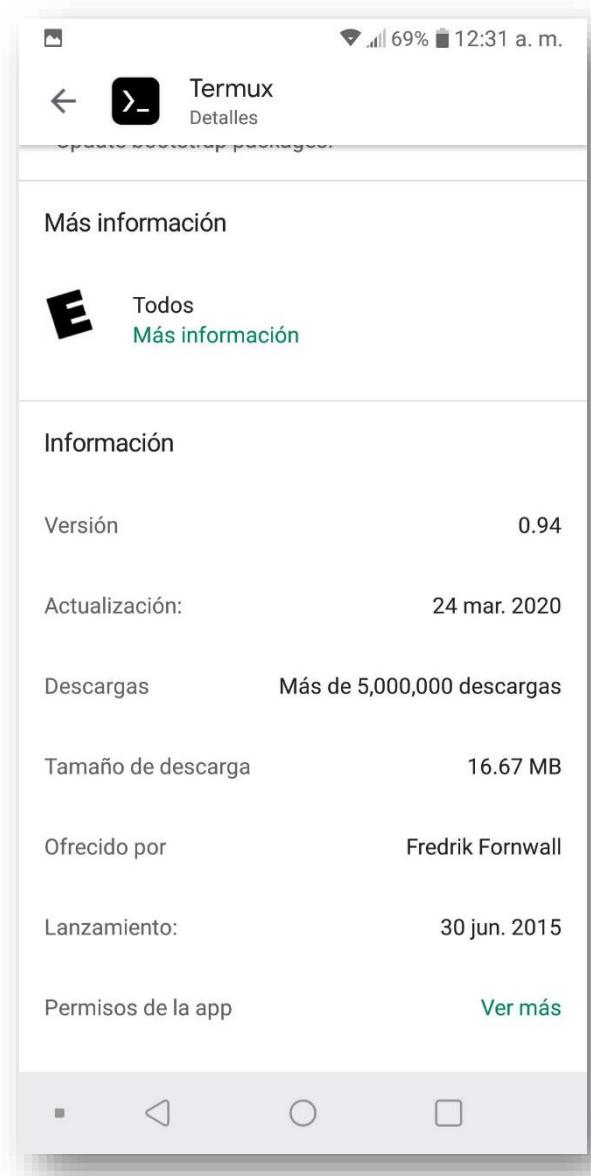
Termuxを使用する大きなメリットの一つは、携帯電話（スマートフォン）を「回転」させずにプログラムをインストールできることです。これにより、このインストールによってメーカー保証が失われることはありません。Termuxのインストール。

モバイルからGoogle Playのアイコンアプリ（play.google.com）にアクセスします。



アプリ「Termux」で検
一ル作業を開始します。

探し、選択してインスト



Termuxアプリケーションの起動。

起動後、Linuxオペレーティングシステムエミュレータのアップデートを実行するために、以下の2つのコマンドを実行する必要があります。

`apt update`

`aptアップグレード`

すべてのオプションを確認する Y(はい)

Termux

Home \$ apt update

\$

apt upgrade

```
Welcome to Termux!
Wiki: https://wiki.termux.com
Community forum: https://termux.com/community
Gitter chat: https://gitter.im/termux/termux
IRC channel: #termux on freenode

Working with packages:
* Search packages: pkg search <query>
* Install a package: pkg install <package>
* Upgrade packages: pkg upgrade

Subscribing to additional repositories:
* Root: pkg install root-repo
* Unstable: pkg install unstable-repo
* X11: pkg install x11-repo

Report issues at https://termux.com/issues
$
```

```
Welcome to Termux!
Wiki: https://wiki.termux.com
Community forum: https://termux.com/community
Gitter chat: https://gitter.im/termux/termux
IRC channel: #termux on freenode

Working with packages:
* Search packages: pkg search <query>
* Install a package: pkg install <package>
* Upgrade packages: pkg upgrade

Subscribing to additional repositories:
* Root: pkg install root-repo
* Unstable: pkg install unstable-repo
* X11: pkg install x11-repo

Report issues at https://termux.com/issues
$ apt update ←
```

```
Welcome to Termux!
Wiki: https://wiki.termux.com
Community forum: https://termux.com/community
Gitter chat: https://gitter.im/termux/termux
IRC channel: #termux on freenode

Working with packages:
* Search packages: pkg search <query>
* Install a package: pkg install <package>
* Upgrade packages: pkg upgrade

Subscribing to additional repositories:
* Root: pkg install root-repo
* Unstable: pkg install unstable-repo
* X11: pkg install x11-repo

Report issues at https://termux.com/issues
$ apt upgrade ←
```

11. Termux内のストレージ構成。

Termuxシステムをアップデートしてアップグレードした後、Termuxシステムで電話機の内部ストレージを見る方法を設定していきます。これにより、Termuxと電話機内の当社の情報をやり取りできるようになります。

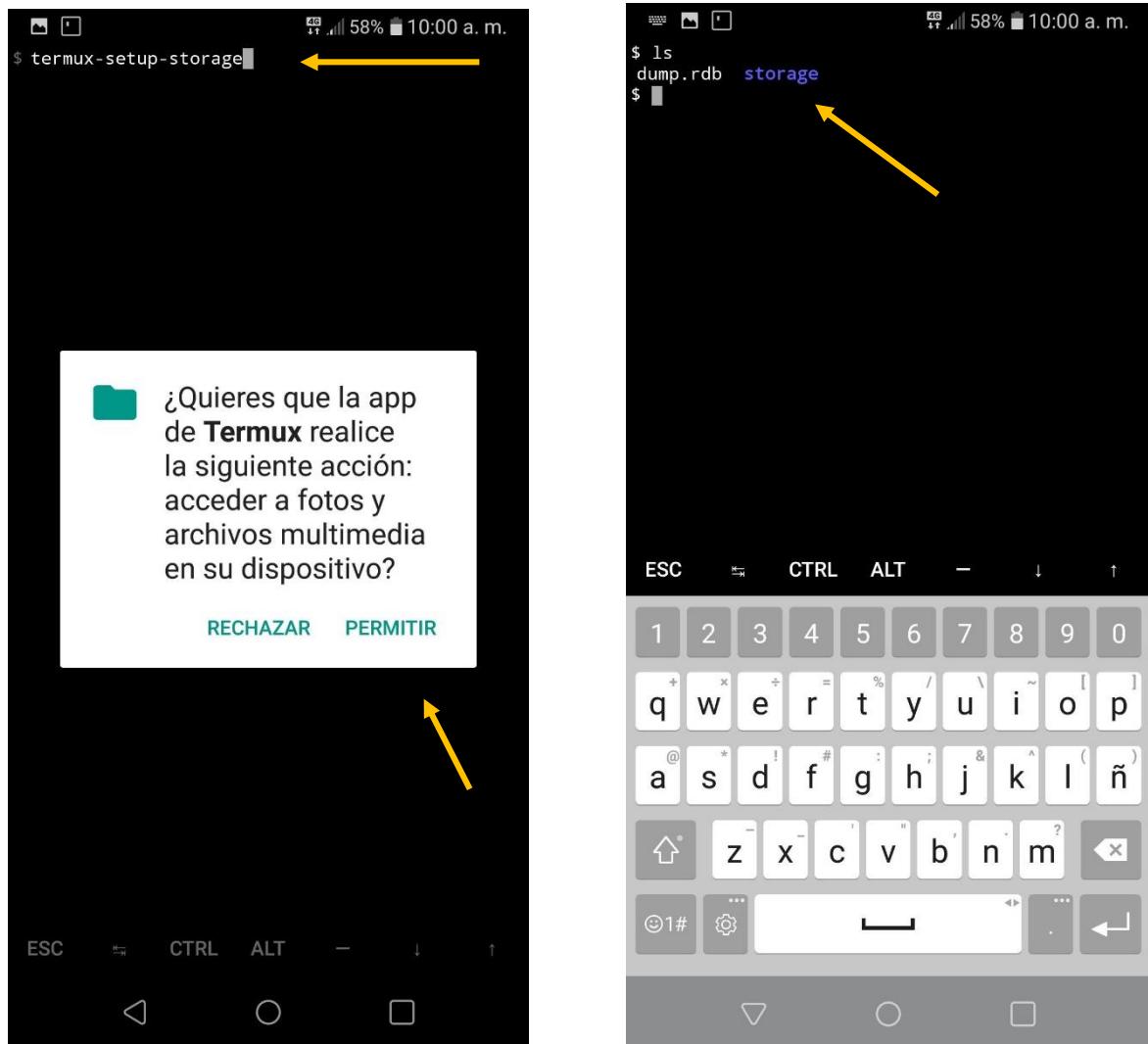
これは、Termuxターミナルで以下のコマンドを実行することで、簡単かつ迅速に行うことができます

◦

```
$ termux-setup-storage
```

先ほどのコマンドを実行すると、Termuxでの仮想ストレージ（ディレクトリ）の作成を確認するウインドウが表示されます。コマンドを与えて検証します。

```
ls
```



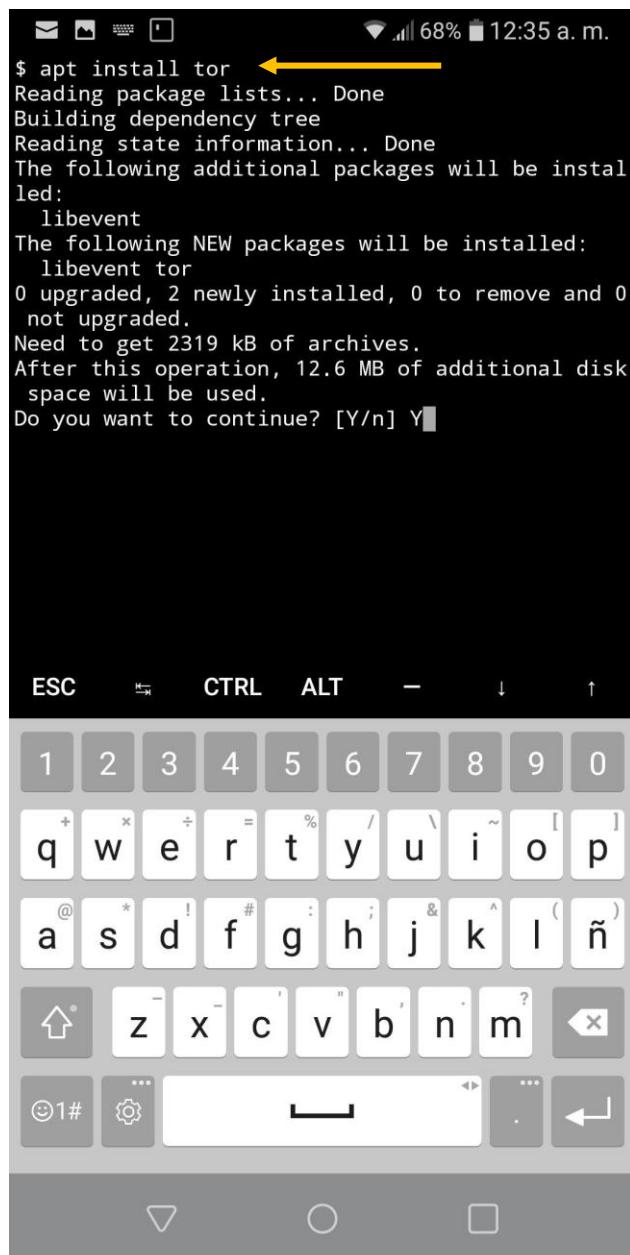
12.Torのネットワークインストールと「Syncthing」のインストール。

```
apt install tor
```

```
apt install syncthing
```

要求された場合は、両方のケースで大文字のYを入力してインストールを受け入れる...

```
apt install tor
```

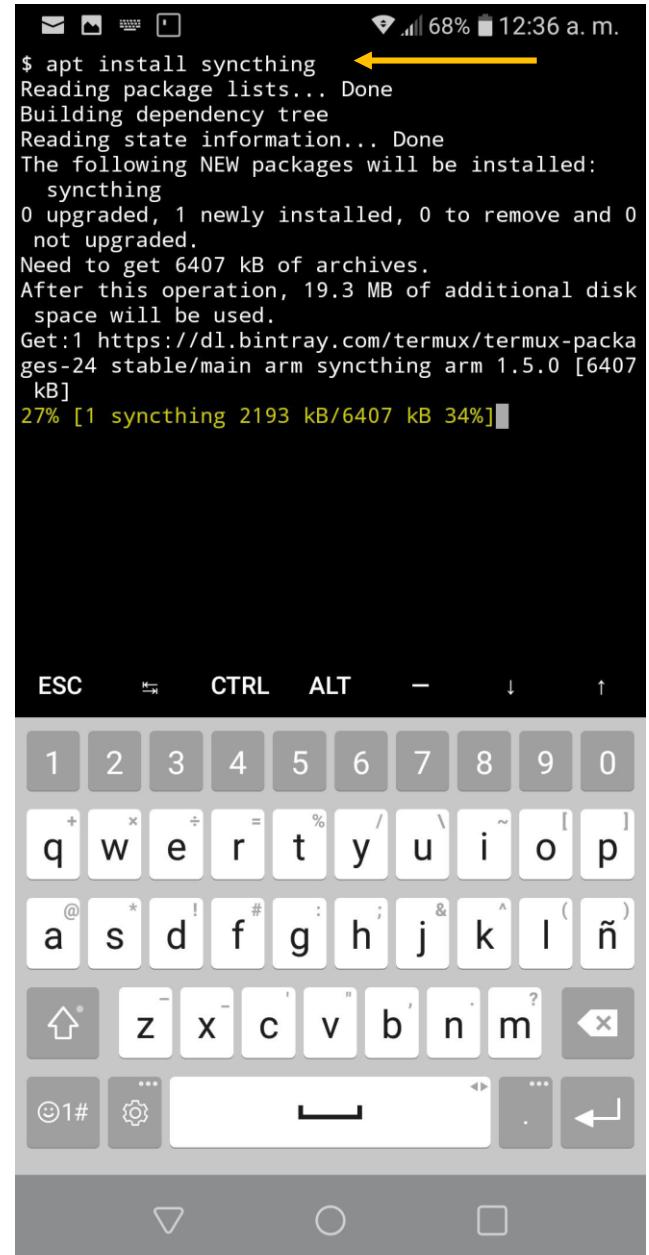


A screenshot of the Termux terminal application on a mobile device. The terminal window shows the command \$ apt install tor being typed. The screen also displays the system status bar at the top, showing signal strength, battery level (68%), and time (12:35 a.m.). Below the terminal window is a standard Android-style keyboard.

```
$ apt install tor
```

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
libevent
The following NEW packages will be installed:
libevent tor
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 2319 kB of archives.
After this operation, 12.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y

```
$ apt install syncthing
```



A screenshot of the Termux terminal application on a mobile device. The terminal window shows the command \$ apt install syncthing being typed. The screen also displays the system status bar at the top, showing signal strength, battery level (68%), and time (12:36 a.m.). Below the terminal window is a standard Android-style keyboard.

```
$ apt install syncthing
```

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
syncthing
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 6407 kB of archives.
After this operation, 19.3 MB of additional disk space will be used.
Get:1 https://dl.bintray.com/termux/termux-packages-24 stable/main arm syncthing arm 1.5.0 [6407 kB]
27% [1 syncthing 2193 kB/6407 kB 34%]

13. データベース「Redis」とSSH（Secure Shell）サーバのインストール。

```
apt install redis
```

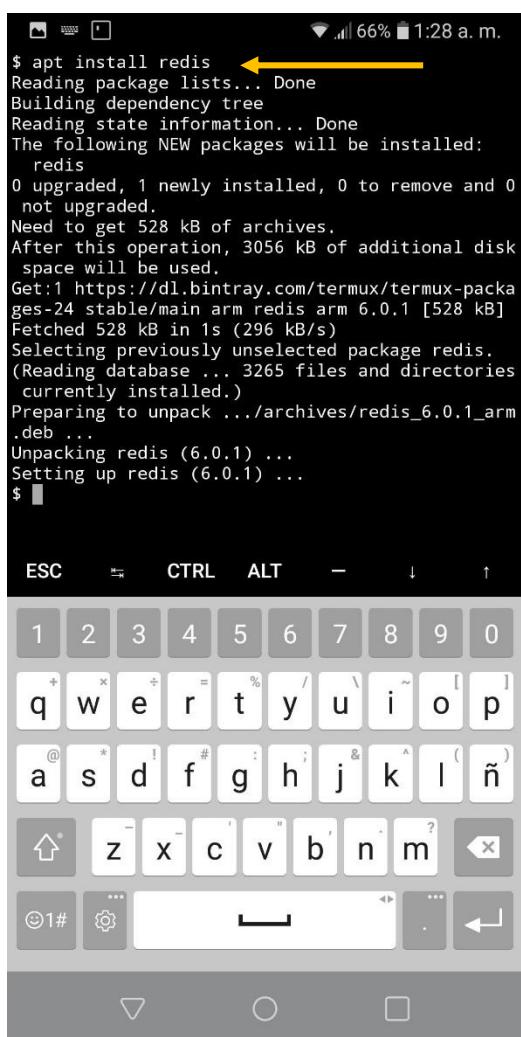
```
apt install openssh
```

```
apt install sshpass
```

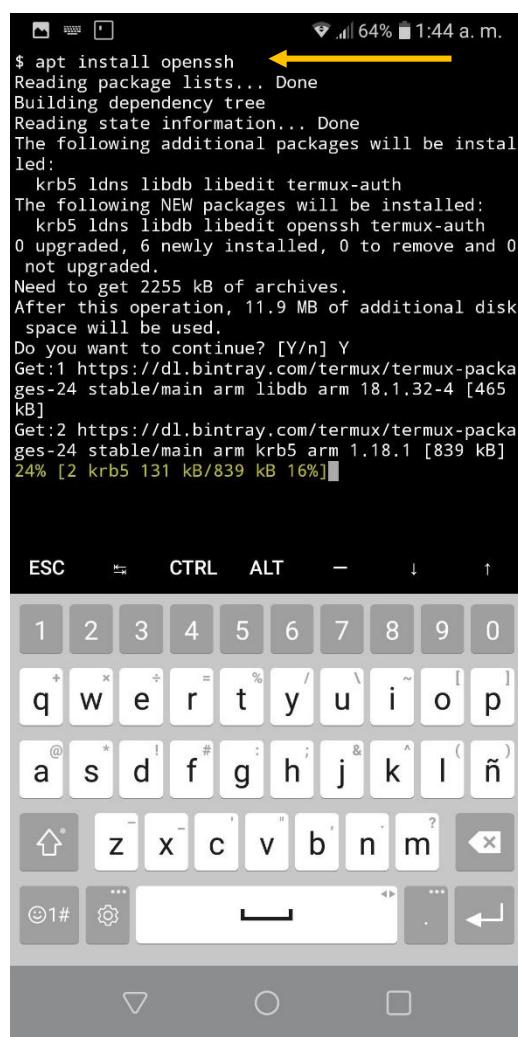
apt install redis

\$ apt install openssh

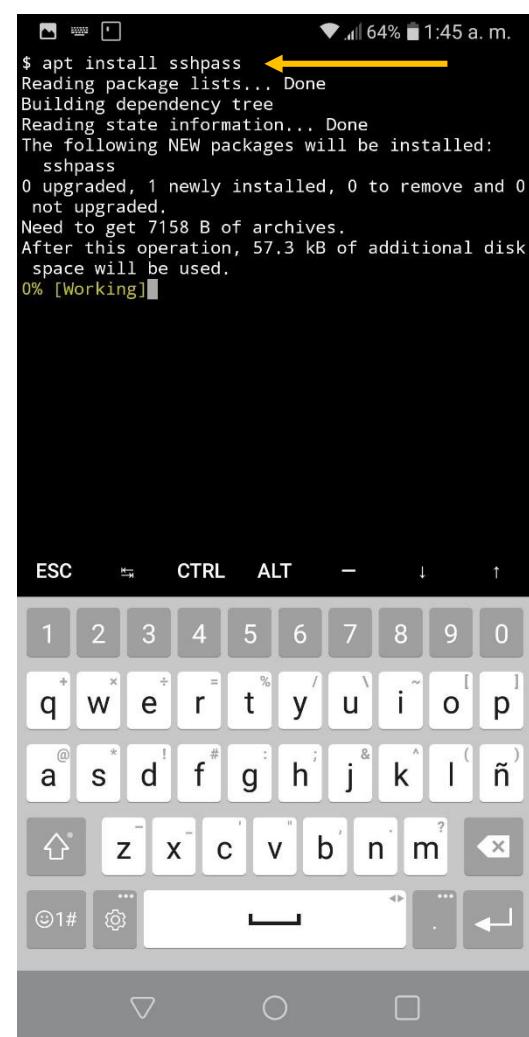
\$ apt install sshpass



```
$ apt install redis
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  redis
0 upgraded, 1 newly installed, 0 to remove and 0
not upgraded.
Need to get 528 kB of archives.
After this operation, 3056 kB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm redis arm 6.0.1 [528 kB]
Fetched 528 kB in 1s (296 kB/s)
Selecting previously unselected package redis.
(Reading database ... 3265 files and directories
currently installed.)
Preparing to unpack .../archives/redis_6.0.1_arm
.deb ...
Unpacking redis (6.0.1) ...
Setting up redis (6.0.1) ...
$
```



```
$ apt install openssh
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be instal
led:
  krb5_ldns libdb libedit termux-auth
The following NEW packages will be installed:
  krb5_ldns libdb libedit openssh termux-auth
0 upgraded, 6 newly installed, 0 to remove and 0
not upgraded.
Need to get 2255 kB of archives.
After this operation, 11.9 MB of additional disk
space will be used.
Do you want to continue? [Y/n] Y
Get:1 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm libdb arm 18.1.32-4 [465
kB]
Get:2 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm krb5 arm 1.18.1 [839 kB]
24% [2 krb5 131 kB/839 kB 16%]
```



```
$ apt install sshpass
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  sshpass
0 upgraded, 1 newly installed, 0 to remove and 0
not upgraded.
Need to get 7158 B of archives.
After this operation, 57.3 kB of additional disk
space will be used.
0% [Working]
```

我々は、通信ネットワークのインストールが完了した、我々はパッケージの設定を続行します：Tor
 、SyncthingとRedis DB。
 オープンキュービットドットコム

14. 携帯電話（スマホ）でのSSHサーバの設定。

携帯電話内のSSHサーバをPCから携帯電話に接続できるようにし、より高速で快適な作業ができるようにするとともに、Mini BlocklyChain内の通信ネットワークで使用するため、携帯電話内のSSHサーバのサービスが正常に動作するかどうかを確認する役割を果たします。

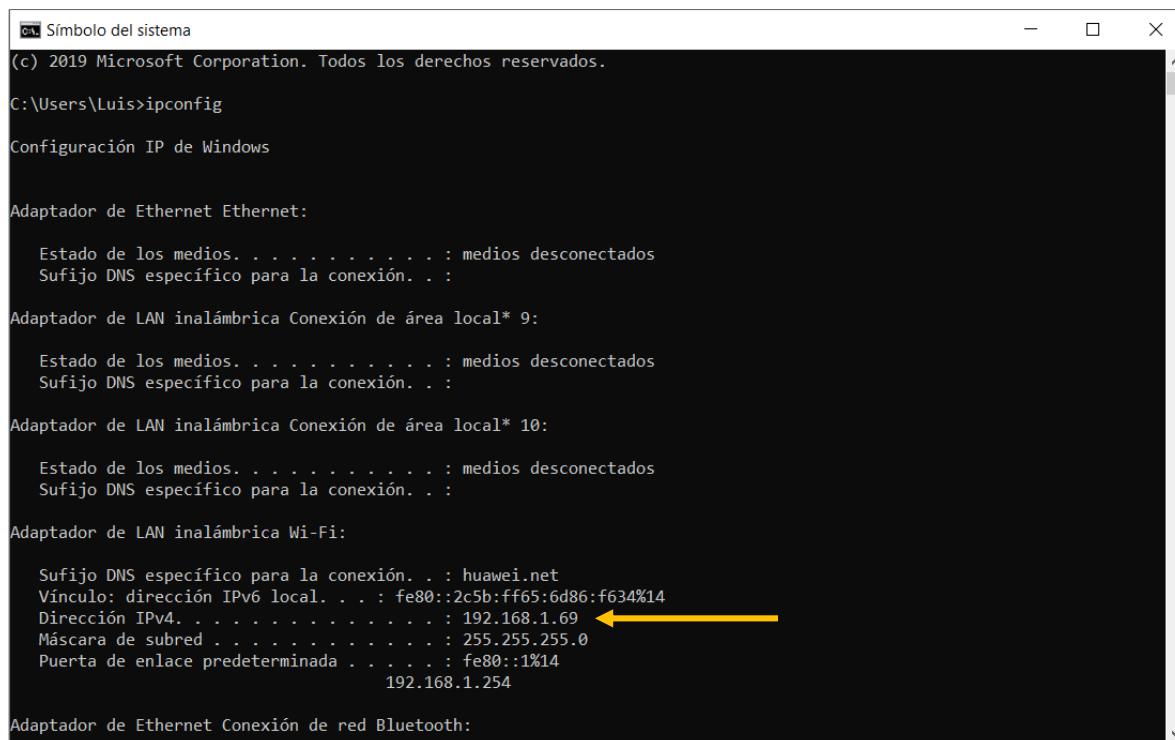
まずはモバイルとPCを同じWiFiネットワークに接続して、お互いに顔が見えるようにすることから始めます。IPまたはアドレスは、192.168.XXX.XXXに似ている必要があります。XXXの値は、各コンピュータでランダムに割り当てられる可変数字です。

この例は、LG Q6携帯電話とWindows 10 Homeを搭載したPCでテストしました。

PCがWiFiに接続しているIPやアドレスを確認するには、Windowsで端末を開く必要があります。

検索拡大鏡がある下のパネルでcmdと書いてEnterキーを押します。ターミナルが開きますので、その中にコマンドを書きます。

C:\User_Name> ipconfig



```

Símbolo del sistema
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Luis>ipconfig

Configuración IP de Windows

Adaptador de Ethernet Ethernet:
Estado de los medios. . . . . : medios desconectados
Sufijo DNS específico para la conexión. . . :

Adaptador de LAN inalámbrica Conexión de área local* 9:
Estado de los medios. . . . . : medios desconectados
Sufijo DNS específico para la conexión. . . :

Adaptador de LAN inalámbrica Conexión de área local* 10:
Estado de los medios. . . . . : medios desconectados
Sufijo DNS específico para la conexión. . . :

Adaptador de LAN inalámbrica Wi-Fi:
Sufijo DNS específico para la conexión. . . : huawei.net
Vínculo: dirección IPv6 local. . . : fe80::2c5b:ff65:6d86:f634%14
Dirección IPv4. . . . . : 192.168.1.69 ←
Máscara de subred . . . . . : 255.255.255.0
Puerta de enlace predeterminada . . . . : fe80::1%14
                                         192.168.1.254

Adaptador de Ethernet Conexión de red Bluetooth:

```

これでPCに割り当てられているIPが192.168.1.69であることがわかりますが、これはそれぞれのケースで異なる可能性が高いです。

注意：「IPv4アドレス」と書かれているアドレスは、ゲートウェイと混同しないように取る必要があります。

携帯電話の場合は、次のコマンドを入力して、携帯電話を持っているSSHサーバに接続するためのユーザ名を知る必要があります。

誰もが

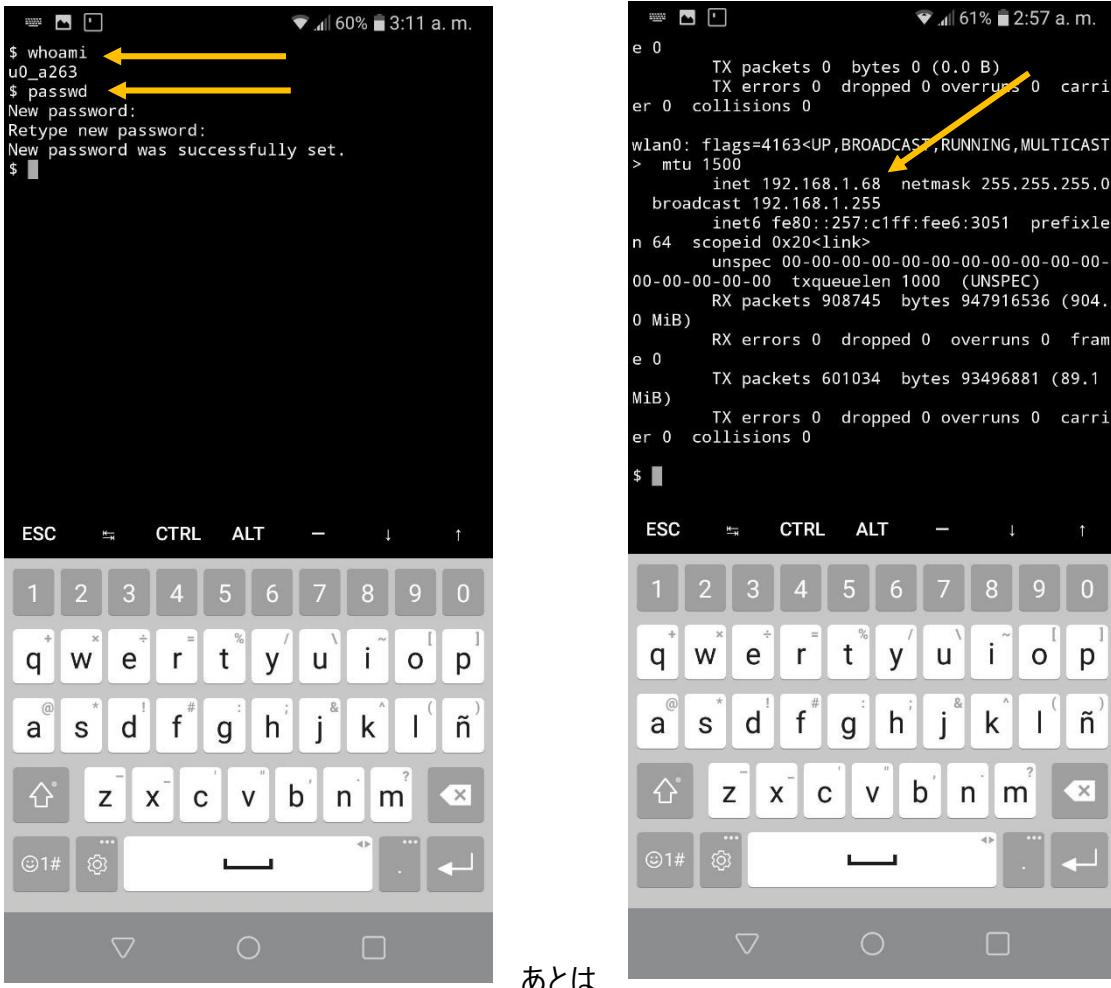
後日、このユーザーにパスワードを与えなければならないので、以下のコマンドを実行しなければなりません。

パスワードを指定する

それは私たちにパスワードを入力してEnterを押すように求められますが、再びそれは私たちがそれを確認し、Enterを押して、それが正常にされている場合は、"新しいパスワードが正常に設定されました"エラーをマーキングの場合には、パスワードが正しく入力されていない可能性があることを確認してください。再度手順を実行してください。

そして、TermuxでどのようなIPを持っているかを知るために、次のコマンドを入力します。

`ifconfig -a`



あとは

スマホのSSHサーバサービスを起動して、PCからのセッションを受信できるようにするだけです。

Termuxターミナルで以下のコマンドを実行してみましたが、何の結果も得られませんでした。

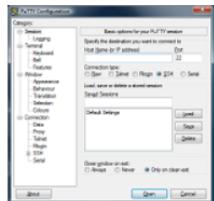
`sshd`



あとは、PCから電話のSSHサーバと通信するプログラムをPCにインストールする必要があります。

<https://www.putty.org>

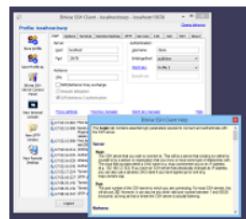
「ここからPuTTYをダウンロードできます」というリンクがある場所を選択してください。



Download PuTTY

PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the with source code and is developed and supported by a group of volunteers.

You can download PuTTY [here](#).



Bitvise SSH Client

Bitvise SSH Client is an SSH and SFTP client for Windows. It is developed and supported prof supports all features supported by PuTTY, as well as the following:

- graphical SFTP file transfer;
- single-click Remote Desktop tunneling;
- auto-reconnecting capability;
- dynamic port forwarding through an integrated proxy;
- an FTP-to-SFTP protocol bridge.

Bitvise SSH Client is **free to use**. You can [download it here](#).

32ビット版を選択してください、それはあなたのシステムが64ビットであるかどうかは問題ではありませんうまく動作します。

Download PuTTY: latest release

[Home](#) | [FAQ](#) | [Feedback](#) | [Licence](#) | [Updates](#) | [Mirror](#)
Download: **Stable** · [Snapshot](#) | [Docs](#) | [Changelog](#)

This page contains download links for the latest released version of PuTTY. Currently this is 0.73, released on 2019-09-29.

When new releases come out, this page will update to contain the latest, so this is a good page to bookmark or link to. Alternative Release versions of PuTTY are versions we think are reasonably likely to work well. However, they are often not the most up-to-date, so check the [development snapshots](#), to see if the problem has already been fixed in those versions.

Package files

You probably want one of these. They include versions of all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

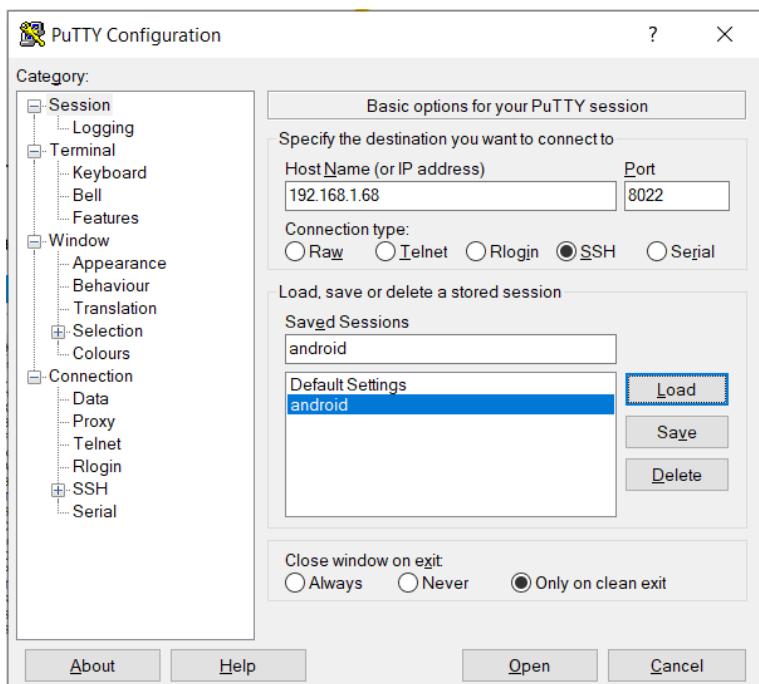
MSI ('Windows Installer')

32-bit:	putty-0.73-installer.msi	(or by FTP)	(signature)
64-bit:	putty-64bit-0.73-installer.msi	(or by FTP)	(signature)

Unix source archive

.tar.gz:	putty-0.73.tar.gz	(or by FTP)	(signature)
----------	-----------------------------------	------------------------------	-----------------------------

PCにダウンロードされたら、実行してデフォルトのオプションでインストールします。その後、PuTTYアプリケーションを起動します。



このセッションでは、携帯電話にインストールしたOpenSSHサーバーのデータを入力していきます。

携帯電話のIPを入力します。

ホスト名またはIPアドレス。

192.168.1.68 (IP例)

港だ

8022 (モバイルSSHサーバのデフォルトポート)。

保存されたセッション」でセッションに名前をつけて保存ボタンをクリックします。

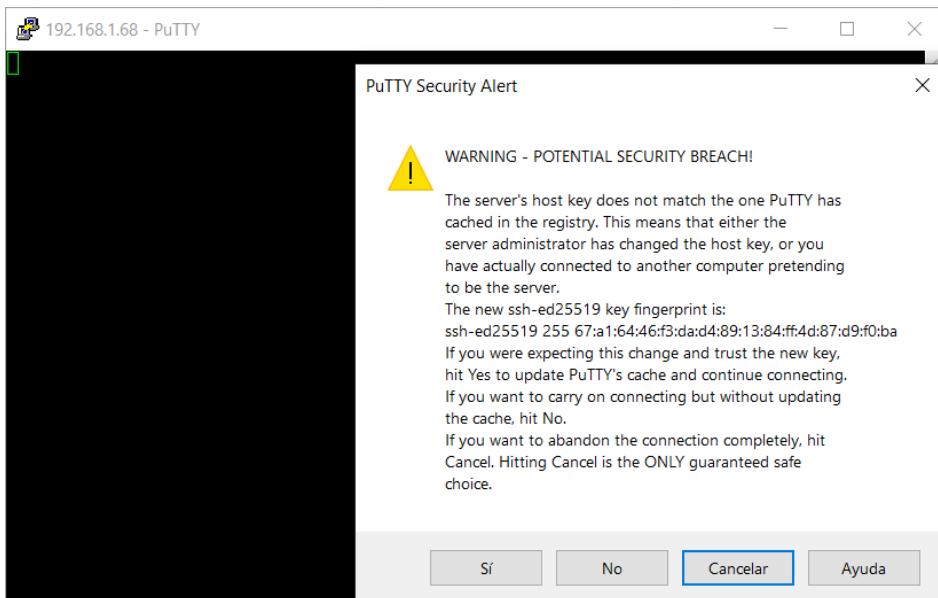
オープンキュービットドットコム

ページ 60 | 237

後で下の部分で我々は"開く"ボタンを与えるサーバーへの接続を開くために押す。

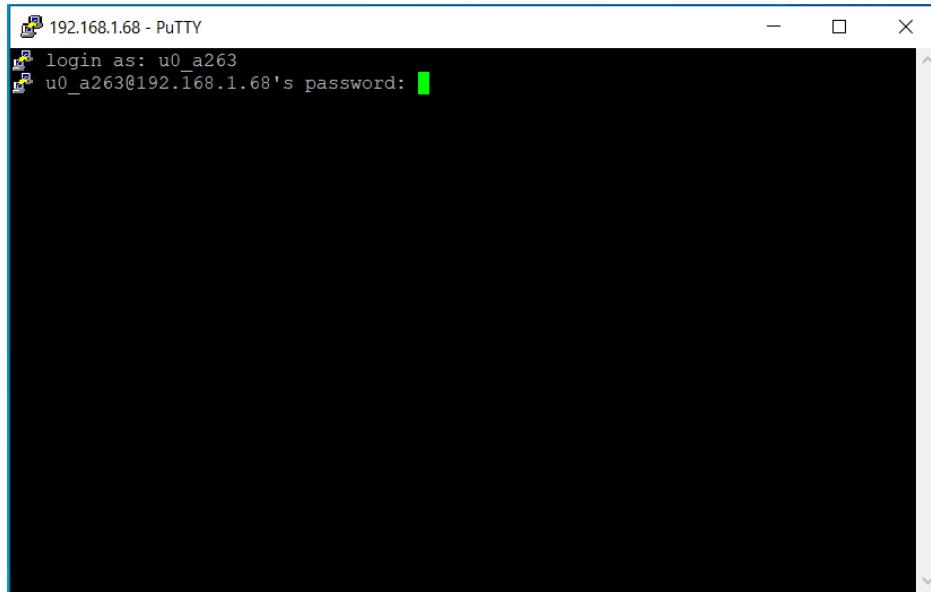
初めて接続したパソコンでは、「はい」ボタンをクリックして情報暗号化キーの確認を求められます

◦



後日、接続するユーザーを聞かれることになります。以前に取得した情報（ユーザーとパスワード）を利用させていただきます。

としてログインでは：私たちは私たちのユーザーを入力し、Enterを与える必要がありますが、その後、我々は再びEnterボタンを与えるパスワードを求められます。



データに間違いがなければ、PC（クライアント）から電話機（SSHサーバ）上で行われるSSH（Secure Shell）セッションになります。

A screenshot of a Termux terminal window titled "192.168.1.68 - PuTTY". The window displays a series of welcome messages and usage instructions. It starts with "Welcome to Termux!", followed by links to a wiki, community forum, Gitter chat, and IRC channel. Then, it provides instructions for working with packages, subscribing to additional repositories, and reporting issues. Finally, it ends with a prompt "\$" at the bottom.

重要な注意: 同じWiFiに接続されているPCのIP(アドレス)と携帯電話のIP(アドレス)は、おそらく私たちが切断して再接続するたびに変更されることを覚えているので、我々は各デバイスが持っているアドレスをダブルチェックする必要があります。これは、携帯電話のSSHサーバとPC(クライアント)を介してデバイス間の接続の成功を保証します。

今まで同じWiFiネットワークにしか接続できませんでしたが、スマホをPCと同じネットワーク外に移動させると、他のもっと複雑な通信機器を経由するのに別のネットワークが関係しているため、接続できなくなります。これは「Tor」のネットワークを設定すれば解決します。

この接続は、電話で導入したサーバーのサービスを確認するためだけのものであり、パソコンから電話へのリモートセッションにより快適な作業環境を実現するためのものであることを忘れないようにしましょう。

15. SSH(Secure Shell)サービスを利用した「Tor」のネットワーク設定。

PC からのリモートセッションでは、SSH サービスを有効にして "Tor" ネットワークの設定を開始します

。

「Tor」ネットワークを持つことの重要性は、同じ WiFi ネットワークにいることなく、インターネットを介して世界中のどこにいても通信できるという特性をデバイスに与えることです。

「Tor」ネットワークは、設定ファイル "torrc" にいくつかのパラメータがあるので、設定に柔軟性があります。このファイルはパス (`$/PREFIX/etc/tor/tor/torrc`) にあります。

エコー `$/PREFIX`

`/data/data/com.termux/files/usr`

そのため、編集する必要のあるファイルはパスになります。

`$/PREFIX/etc/tor/tor/torrc` は `/data/data/com.termux/files/usr/etc/tor/torrc` と同じです。

以下のコマンドを実行して、コマンドラインエディタ「vi」を使って設定ファイルの編集を進めます。

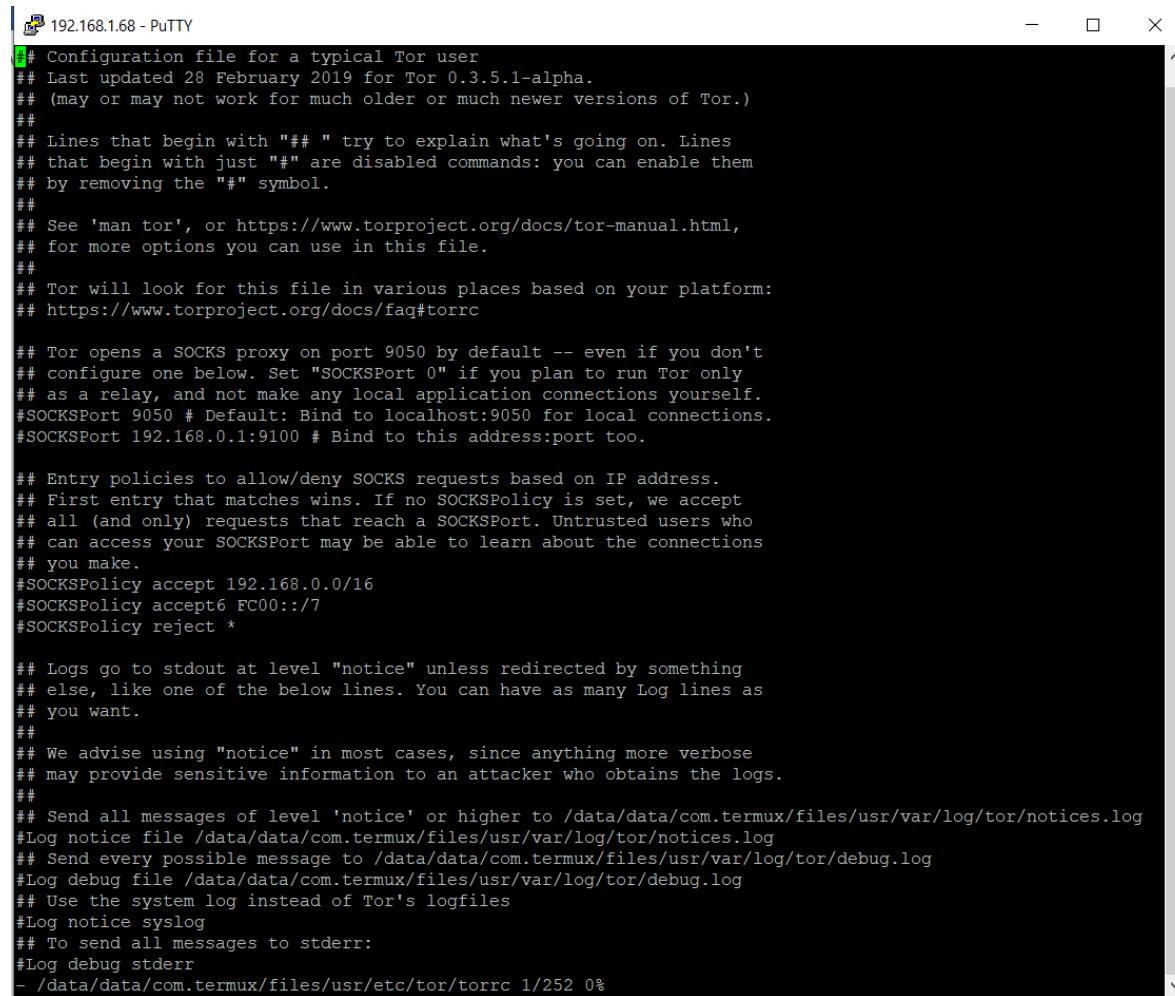
`vi /data/data/com.termux/files/usr/etc/tor/torrc`

例のごとく、そのファイルを編集してくれます。



```
192.168.1.68 - PuTTY
$ vi /data/data/com.termux/files/usr/etc/tor/torrc
```

torrcファイルを編集しました。



```
# Configuration file for a typical Tor user
## Last updated 28 February 2019 for Tor 0.3.5.1-alpha.
## (may or may not work for much older or much newer versions of Tor.)
##
## Lines that begin with "## " try to explain what's going on. Lines
## that begin with just "#" are disabled commands: you can enable them
## by removing the "#" symbol.
##
## See 'man tor', or https://www.torproject.org/docs/tor-manual.html,
## for more options you can use in this file.
##
## Tor will look for this file in various places based on your platform:
## https://www.torproject.org/docs/faq#torrc

## Tor opens a SOCKS proxy on port 9050 by default -- even if you don't
## configure one below. Set "SOCKSPort 0" if you plan to run Tor only
## as a relay, and not make any local application connections yourself.
#SOCKSPort 9050 # Default: Bind to localhost:9050 for local connections.
#SOCKSPort 192.168.0.1:9100 # Bind to this address:port too.

## Entry policies to allow/deny SOCKS requests based on IP address.
## First entry that matches wins. If no SOCKSPolicy is set, we accept
## all (and only) requests that reach a SOCKSPort. Untrusted users who
## can access your SOCKSPort may be able to learn about the connections
## you make.
#SOCKSPolicy accept 192.168.0.0/16
#SOCKSPolicy accept6 FC00::/7
#SOCKSPolicy reject *

## Logs go to stdout at level "notice" unless redirected by something
## else, like one of the below lines. You can have as many Log lines as
## you want.
##
## We advise using "notice" in most cases, since anything more verbose
## may provide sensitive information to an attacker who obtains the logs.
##
## Send all messages of level 'notice' or higher to /data/data/com.termux/files/usr/var/log/tor/notices.log
#Log notice file /data/data/com.termux/files/usr/var/log/tor/notices.log
## Send every possible message to /data/data/com.termux/files/usr/var/log/tor/debug.log
#Log debug file /data/data/com.termux/files/usr/var/log/tor/debug.log
## Use the system log instead of Tor's logfiles
#Log notice syslog
## To send all messages to stderr:
#Log debug stderr
- /data/data/com.termux/files/usr/etc/tor/torrc 1/252 0%
```

この「torrc」ファイルでは、以下のような変更を加えることで、ファイルが持っている行を追加したり
、使用したりする必要があります。

構文：SOCKSPort <アプリケーションポート番号

例：SOCKSPort 9050

SOCKSPort変数は、TCP-IPプロトコル上のこの通信ソケットがデフォルトでモバイルデバイス(電話)
を使用し、ポート9050がオープンまたは使用中であることを教えてくれます。

構文: HiddenServiceDir <アプリケーションの設定が保存されるディレクトリ>

例: HiddenServiceDir

/data/data/com.termux/files//data/data/data/com.termux/files/files/.

HiddenServiceDir変数は、Torネットワークを介して使用されるサービスの設定が保存されるディレクトリであることを示しています。このディレクトリには、サービスの設定とセキュリティファイルがあり、このディレクトリにはホスト名と呼ばれるファイルがあります。

私たちの場合はTorネットワーク上に実装されるSSHサービスを作成しているので、Torネットワークによって割り当てられたアドレスを見るることができます。このディレクトリは、Torネットワークサービスを起動すると自動的に作成されるので、作成する必要はありません。

Torネットワークから提供されるアドレスを確認するには、"more"コマンドを使用します。このコマンドを使用する前に、ホスト名ファイルの場合と同様に、"torrc"ファイルの設定を終えて、隠しディレクトリとその中のファイルが作成されるようにTorネットワークサービスを登録しておく必要があります

。

その他の /data/data/com.termux/files/

上記のコマンドは、拡張子が .onion に似ている英数字文字列のアドレスになります。

uqwthf6ojdmoybyzvudoq3x2weq7k7rjitblhba3pjbausk2nvjmx3wer.onion

最後に、HiddenServicePort変数を"torrc"設定ファイルに追加する必要があります。このパラメータは、サインアップするアプリケーションがTorネットワーク上でどのポートを使用するかをTorネットワークに通知します。

構文 : HiddenServicePort <出発地のPort> <ローカルまたはパブリックIP>: <出発地のPort>

O

HiddenServicePort <出発地のポート>

例。

HiddenServicePort 22 127.0.0.1:8022

O

HiddenServicePort 8022

以上で、一般サービス用のTorネットワークとSSH(Secure Shell)サービスの設定は完了しました。このサービスはSQLiteデータベースの更新通信に使用され、ミニBlocklyChainシステムの検証処理を保存します。

続きまして、Mini BlocklyChain通信網の構成を紹介します。

16. 手動同期によるPeer to Peerシステム構成。

1つは、ノードがプライベートネットワーク（Wifi）、パブリックネットワーク（インターネット）、またはこれら2つのハイブリッドにあるかどうかに関係なく、すべてのノードにいつでも同じ更新された（同期された）情報を提供すること、そしてこのタイプのアーキテクチャの2つ目のポイントは、ノード間の情報を転送、更新、参照するために仲介者（サーバー）に依存しないということです。これは、通信がノード間で直接行われるという事実に起因していますが、私たちの場合、Mini BlocklyChainでは、通信は仲介者を介さずにネットワークを形成する電話機間で直接行われます。

このタスクでは、「Peer to Peer」アーキテクチャに基づいて動作するオープンソースのSyncthingツールを使用します。

デバイス（携帯電話）用の Syncthing の設定は、手動でも自動でも見ることができます。手動形式は5ノードまでの同期に適用され、自動設定の数が多い場合に最適であり、選択した情報の同期を実行したいノードの登録に焦点を当てて処理を行います。

手始めに必要な要件は



1. Torネットワークのサービスを開始しました。
2. オプション - 推奨) QRコードを読み取ることができます。アプリケーションをインストールしている、我々は、Google Playからインストールするのは簡単で簡単ですApp inventor Androidアプリケーションをお勧めしますし、後でこのマニュアルでは、「ミニBlocklyChainのブロックの定義と使用」のセクションで使用します。
3. シンシングのサービスを開始したこと。

今後、Syncthingへのノード登録に役立つQRコードの読み取りに使用するApp Inventorアプリを紹介します。

以下の機種を用いて、2台のモバイル機器（電話機）間で実施例を行った。

モバイルデバイス#1：モデルLG Q6

モバイル端末2位：Samsungモデル

まずは以下のコマンドでTorのネットワークサービスと同期サービスを起動し、Termux内部で2つの端末を開き、それぞれのコマンドを別々に実行してみます。

\$ tor

Torネットワークプログラム起動コマンドを入力した後、100%実行されたことを確認することで、実行が成功したことを確認する必要があります。

Termux端末でTorプログラムを実行し、100%で実行されていることを確認します。

\$ tor

```
$ tor
May 23 23:00:30.932 [notice] Tor 0.4.3.5 running
on Linux with Libevent 2.1.11-stable, OpenSSL 1
.1.1g, Zlib 1.2.11, Liblzma 5.2.5, and Libzstd N
/A.
May 23 23:00:30.934 [notice] Tor can't help you
if you use it wrong! Learn how to be safe at htt
ps://www.torproject.org/download/download#warnin
g
May 23 23:00:30.939 [notice] Read configuration
file "/data/data/com.termux/files/usr/etc/tor/to
rrc".
May 23 23:00:30.970 [notice] I think we have 8 C
PUS, but only 6 of them are available. Telling T
or to only use 6. You can override this with the
NumCPUs option
May 23 23:00:30.973 [notice] Opening Socks liste
ner on 127.0.0.1:9050
May 23 23:00:30.974 [notice] Opened Socks liste
ner on 127.0.0.1:9050
May 23 23:00:30.000 [notice] Parsing GEOIP IPv4
file /data/data/com.termux/files/usr/share/tor/g
eoip.
May 23 23:00:32.000 [notice] Parsing GEOIP IPv6
file /data/data/com.termux/files/usr/share/tor/g
eoip6.
May 23 23:00:34.000 [notice] Bootstrapped 0% (st
arting): Starting
May 23 23:00:34.000 [notice] Starting with guard
context "default"
May 23 23:00:35.000 [notice] Bootstrapped 5% (co
nn): Connecting to a relay
May 23 23:00:36.000 [notice] Bootstrapped 10% (c
onn_done): Connected to a relay
May 23 23:00:36.000 [notice] Bootstrapped 14% (h
andshake): Handshaking with a relay
May 23 23:00:36.000 [notice] Bootstrapped 15% (h
andshake_done): Handshake with a relay done
May 23 23:00:36.000 [notice] Bootstrapped 20% (o
nehop_create): Establishing an encrypted directo
ry connection
May 23 23:00:36.000 [notice] Bootstrapped 25% (r
```

ESC ⌂ CTRL ALT - ↓ ↑

■ ◀ ○ □

```
ps://www.torproject.org/download/download#warnin
g
May 24 01:33:32.982 [notice] Read configuration
file "/data/data/com.termux/files/usr/etc/tor/to
rrc".
May 24 01:33:33.007 [notice] I think we have 8 C
PUS, but only 6 of them are available. Telling T
or to only use 6. You can override this with the
NumCPUs option
May 24 01:33:33.010 [notice] Opening Socks liste
ner on 127.0.0.1:9050
May 24 01:33:33.010 [notice] Opened Socks liste
ner on 127.0.0.1:9050
May 24 01:33:33.000 [notice] Parsing GEOIP IPv4
file /data/data/com.termux/files/usr/share/tor/g
eoip.
May 24 01:33:34.000 [notice] Parsing GEOIP IPv6
file /data/data/com.termux/files/usr/share/tor/g
eoip6.
May 24 01:33:35.000 [notice] Bootstrapped 0% (st
arting): Starting
May 24 01:33:37.000 [notice] Starting with guard
context "default"
May 24 01:33:38.000 [notice] Bootstrapped 5% (co
nn): Connecting to a relay
May 24 01:33:38.000 [notice] Bootstrapped 10% (c
onn_done): Connected to a relay
May 24 01:33:39.000 [notice] Bootstrapped 14% (h
andshake): Handshaking with a relay
May 24 01:33:39.000 [notice] Bootstrapped 15% (h
andshake_done): Handshake with a relay done
May 24 01:33:39.000 [notice] Bootstrapped 75% (e
nough_dirinfo): Loaded enough directory info to
build circuits
May 24 01:33:39.000 [notice] Bootstrapped 90% (a
p_handshake_done): Handshake finished with a rel
ay to build circuits
May 24 01:33:39.000 [notice] Bootstrapped 95% (c
ircuit_create): Establishing a Tor circuit
May 24 01:33:42.000 [notice] Bootstrapped 100% (d
one): Done
```

ESC ⌂ CTRL ALT - ↓ ↑

■ ◀ ○ □

そして、シンキングコマンドを実行します。

シンクティング

このコマンドを実行した後、それが自動的に開かない場合は、私たちの携帯電話のブラウザで管理ページを開きます。我々は、我々は通常、インターネット上でナビゲートするインストールされている任意のブラウザに移動することができますし、我々は、次のを置くことができます。

<http://127.0.0.1:8384>

それは私達がシステムのすべてのノード（電話）間で私達の情報を同期させるのを助けるツールの管理画面を開きます。



```
$ syncthing
[monitor] 04:02:07 INFO: Default folder created
and/or linked to new config
[start] 04:02:07 INFO: syncthing v1.5.0 "Fermium
Flea" (go1.14.2 android-arm) builder@6bdf862223
8a 2020-05-11 08:38:11 UTC
[start] 04:02:07 INFO: Generating ECDSA key and
certificate for syncthing...
[start] 04:02:08 INFO: Default folder created an
d/or linked to new config
[start] 04:02:08 INFO: Default config saved. Edi
t /data/data/com.termux/files/home/.config/synct
hing/config.xml to taste (with Syncthing stopped
) or use the GUI
[OWEPS] 04:02:08 INFO: My ID: OWEPSNA-6RZI6S7-SK
VOU65-V44BC5Z-CXZOJI4-C3JC7HM-IPY4V35-JQAKPAW
[OWEPS] 04:02:09 INFO: Single thread SHA256 perf
ormance is 12 MB/s using crypto/sha256 (12 MB/s
using minio/sha256-simd).
[OWEPS] 04:02:11 INFO: Hashing performance is 11
.17 MB/s
[OWEPS] 04:02:11 INFO: Migrating database to sch
ema version 1...
```



Otros dispositivos

- ⓘ Cambios recientes
- + Añadir un dispositivo

「シンクティング」の管理画面を開いているので、情報を同期させたいノードやノードの登録を進めていきます。この時点でQRコードを読み取るプログラムを占拠することになります。

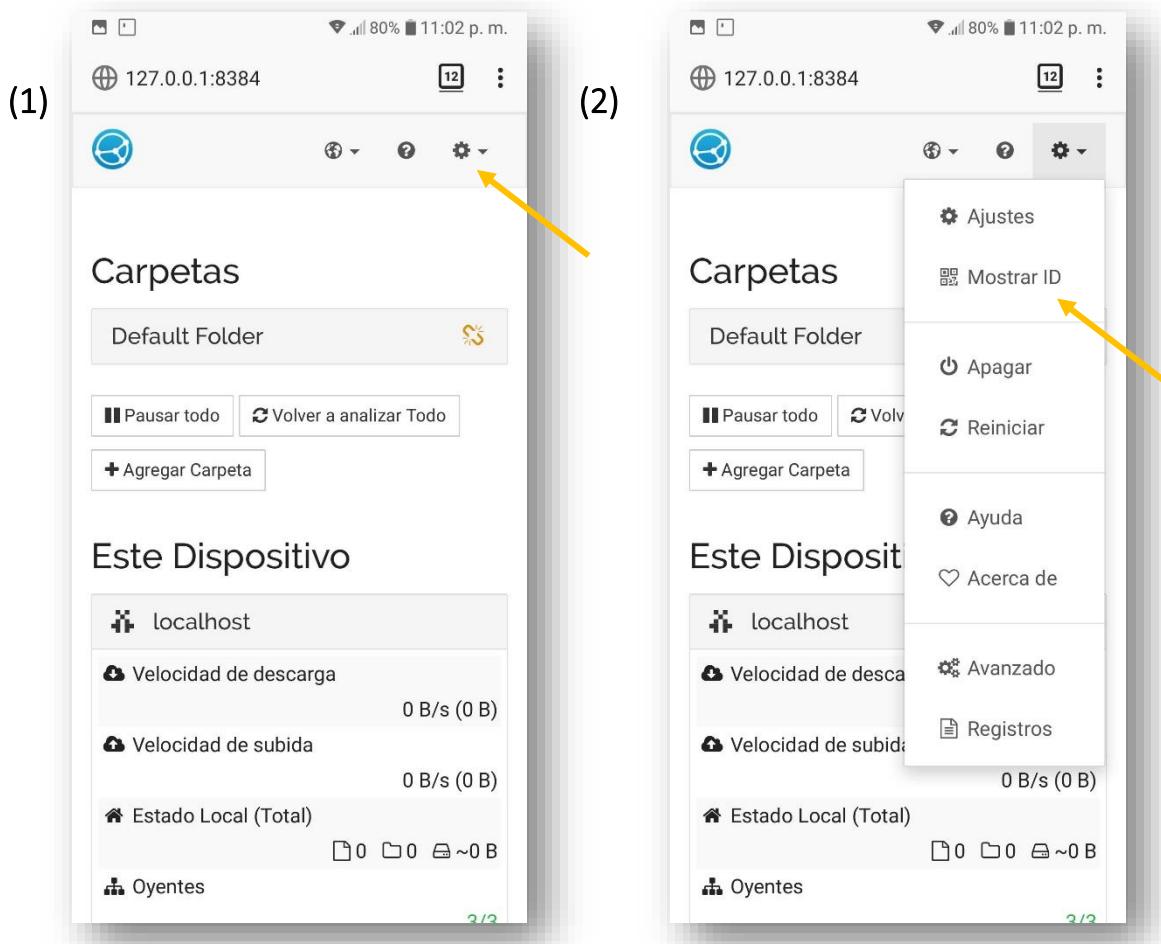
初めて起動したときの同期プログラムは、大文字の英数字8文字のグループで構成された電話機の一意の識別子を作成します。

うちの場合はLG Q6の電話IDをSamsungの電話に登録する必要があり、Samsungの電話IDはLG Q6に登録する必要があります。両方の電話に入っていないと正常に動作しません。

LG Q6のスマホでSamsungの携帯電話登録の手順を行います。

まず(1)同期機能付きSamsung携帯の管理画面(インターネットブラウザ)の右上にあるメニューハンバーガーをクリックします。

2番目の(2)ステップでは、メニューが表示されますが、この中でSamsungの"Show ID"をクリックします。

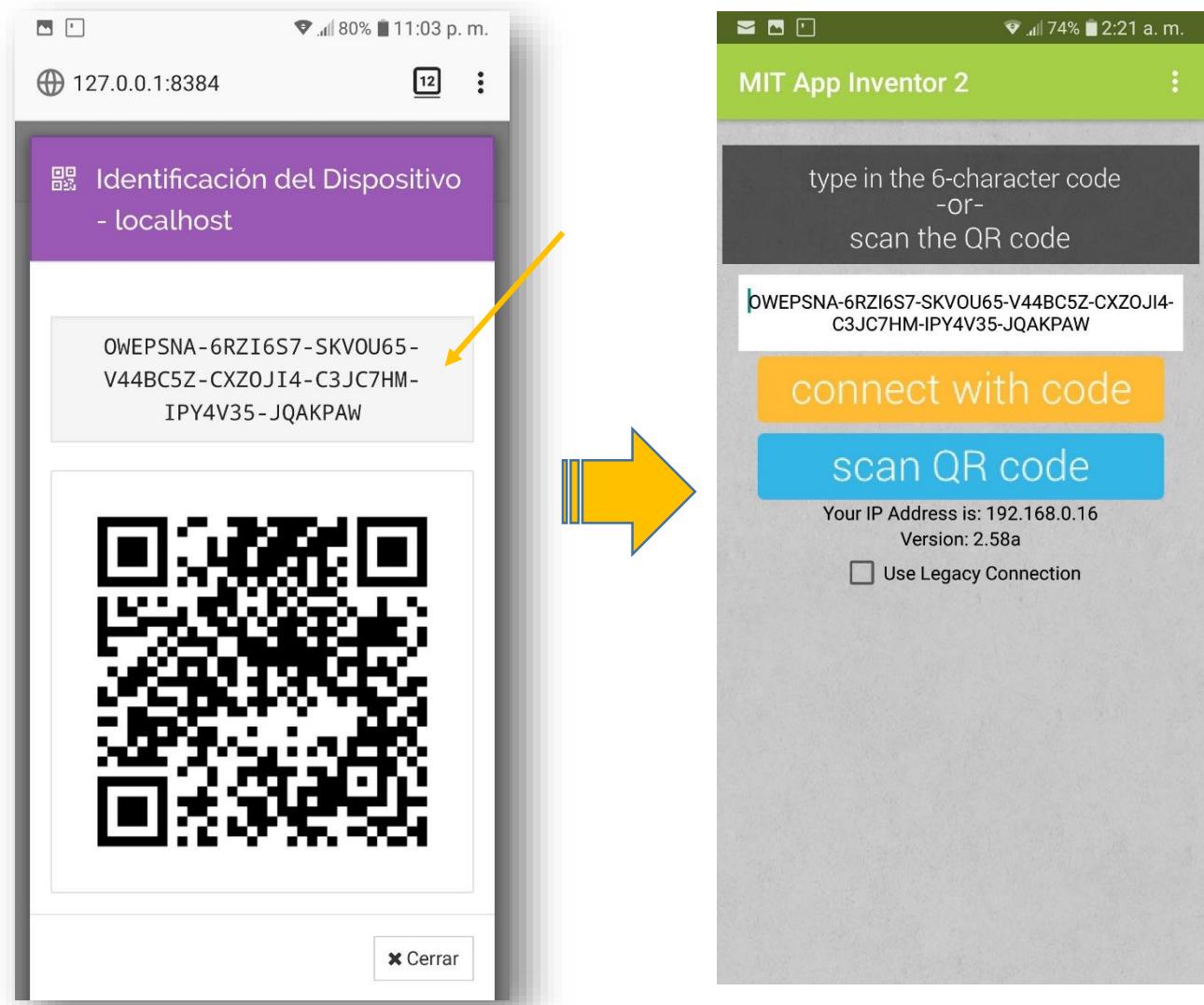


「IDを表示」をクリックすると、次の画面が表示され、Samsungの携帯電話のQRコードが表示されます。

OWEPSNA-6RZI6S7-SKVOU65-V44BC5Z-CXZOJI4-C3JC7HM-IPY4V35-JQAKPAW

そして、LG Q6の携帯電話では、このSamsungのQRコードをキャプチャするのに役立つプログラムは、App Inventorアプリ（オプション）から提案しているものですが、それを持っていない場合は、LG Q6で登録を開始するときに手動で導入することができますが、エラーを避けるためには、それを使用することをお勧めします。

LG Q6 mobileにインストールされているアプリプログラム発明者を使用して、同期したい遠隔地のSamsung携帯電話からQRコードをキャプチャします。



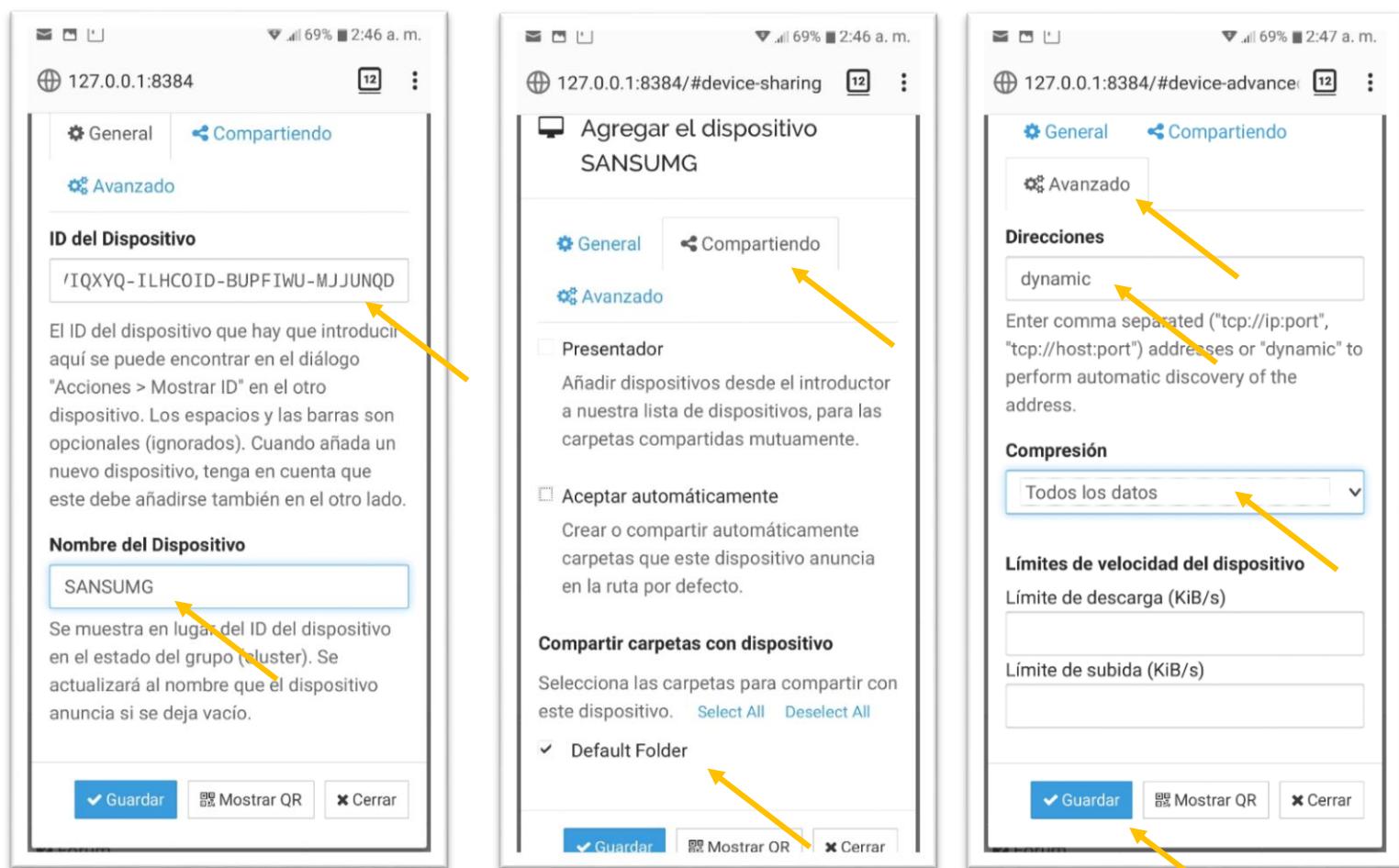
次に、キャプチャしたコードをクリックして、アプリ発明者プログラム内のQRコードをクリップボードにコピーし、「SELECT ALL」→「COPY」を選択します。

この情報をもとに、LG Q6にSamsungの登録を進めます。管理画面では「その他のデバイス」と書いてある下の方に移動し、「デバイスを追加」というボタンをクリックすると、SamsungのIDを導入して登録名を付けます。

次に、上部タブの「共有」をクリックして、「デフォルト」フォルダ内の下部オプションをマークします。

次に、上部の「詳細設定」タブをクリックして、「すべてのデータ」の下にあるデータ圧縮オプションを選択します。

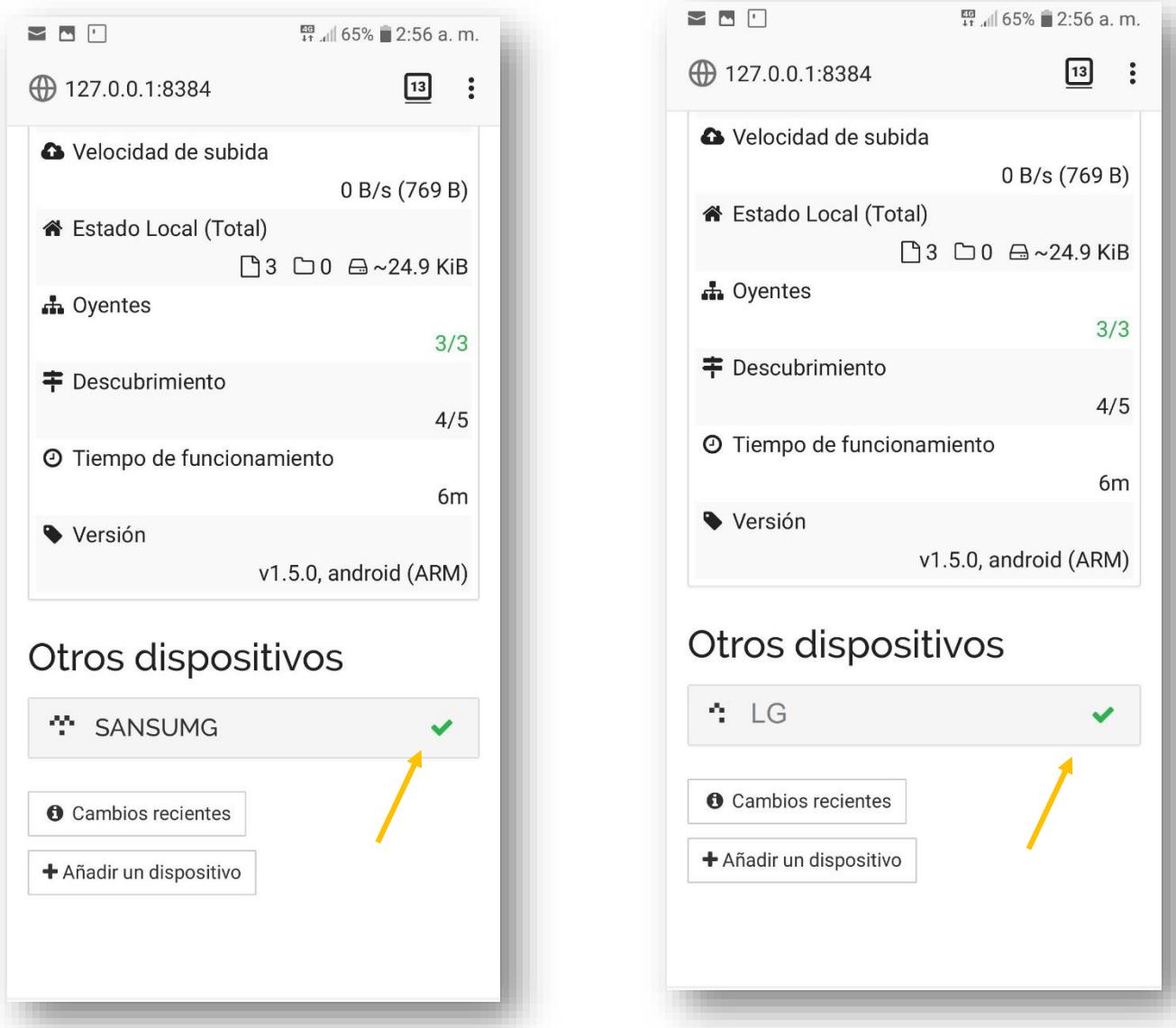
最後に、我々は下の保存ボタンをクリックして、我々はノードでの登録を完了し、この同じプロセスは、我々が同期したいリモートの電話や携帯電話で行う必要があります。



両方の携帯電話に保存して登録する場合は、デバイスが配置されている最大30秒の期間を待ち、デバイス間の接続が良好であることが緑色で確認されます。

デバイス#1 LG Q6

デバイス#2 Samsung



パス /data/data/com.termux/file/home/Sync にある Sync ディレクトリ内のすべての情報が同期され、変更があった場合はコピーされて同期されます。

Mini BlocklyChainで使用するために自動的にSyncthingを使用したPeer to Peerのシステム構成。

これまで手動で設定していましたが、最小限のノードを扱う場合には便利ですが、大量のノードを扱う場合には効率が悪いので、後でSyncthingManagerツールを使って自動化されたオンラインコマンドを使って自動で設定します。

(スレーブ)モードノードのRedisデータベース構成。

Android携帯電話用Redis(Slave)データベースの
`/data/data/com.termux/files/usr/etc/redis.conf`ファイルの設定。

ファイルに以下の変更や指示を追加し、変更を保存してRedisサーバーを起動します。

まず、`slaveof`行から#文字を見つけて削除します。このディレクティブは、Redisマスタサーバと安全に連絡を取るために使用する IPアドレスとポートをスペースで区切って指定します。デフォルトでは、Redisサーバはローカルインターフェースの6379をリッスンしますが、ネットワークセキュリティの各方法は、他の人のために何らかの方法でデフォルトを変更します。

使用する値は、ネットワークトラフィックを保護するために使用した方法によって異なります。

隔離されたネットワーク：隔離されたネットワークのIPアドレスとマスターサーバーのRedisポート(6379)を使用します(例：単純なIP_アドレス6379のスレーブ)。

stunnel or spiped: ローカルインターフェース (127.0.0.1) と設定されたポートを使用してトラフィックをトンネリングします。

PeerVPN: マスターサーバーのIP VPNアドレスと通常のRedisポートを使用します。

将军なら変えてくれるだろう。

`slaveof ip_contact_server master_contact_ port`

例: `slaveof 192.168.1.69 6379`

`masterauth` あなたのネットワークマスターpassword

例: `masterauth sdfssdfsdf 12WqE34Rfgthtdfd`

`requirepass your_network_slave_password`

例: `requirepass asdsjdsh34sds67sdFGbbnh`

以下のコマンドでTermuxターミナルからサーバーを保存して実行します。

```
$ redis-server redis.conf
```

実行後、Windows 10サーバー（マスター）とどのように同期しているかを見ることができます。

設定ファイル redis.conf \$ redis-server redis.conf

ノードの同期管理ツールのインストール。SyncthingManagerのインストールに進みます。

まず、SyncthingManager を正しく動作させるために必要なものをインストールします。

```
apt install Python
```

```
pip3 install -upgrade pip
```

```
npm install syncthingmanager
```

SyncthingManagerツールは、新しいノードと既存のノードを手動ではなく、自動で「ピアツーピア」を同期させるのに役立ちます。

```
$ apt install python ←
Reading package lists... Done
Building dependency tree
Reading state information... Done
python is already the newest version (3.8.3).
0 upgraded, 0 newly installed, 0 to remove and 0
not upgraded.
$ ←
```



```
$ pip3 install --upgrade pip ←
Requirement already up-to-date: pip in /data/dat
a/com.termux/files/usr/lib/python3.8/site-pac
kes (20.1.1)
$ ←
```



```
$ pip3 install syncthingmanager ←
Requirement already satisfied: syncthingmanager
in /data/datab/com.termux/files/usr/lib/python3.8/site-pac
kes (0.1.0)
Requirement already satisfied: syncthing in /dat
a/datab/com.termux/files/usr/lib/python3.8/site-pac
kes (from syncthingmanager) (2.3.1)
Requirement already satisfied: python-dateutil==2.6.1
in /data/datab/com.termux/files/usr/lib/python3.8/site-pac
kes (from syncthing->syncthingm
anager) (2.6.1)
Requirement already satisfied: requests==2.18.4
in /data/datab/com.termux/files/usr/lib/python3.8/site-pac
kes (from syncthing->syncthingmanager)
(2.18.4)
Requirement already satisfied: six>=1.5 in /data
/datab/com.termux/files/usr/lib/python3.8/site-pac
kes (from python-dateutil==2.6.1->syncthing->
syncthingmanager) (1.15.0)
Requirement already satisfied: chardet<3.1.0,>=3
.0.2 in /data/datab/com.termux/files/usr/lib/pyt
hon3.8/site-pac
kes (from requests==2.18.4->sync
thing->syncthingmanager) (3.0.4)
Requirement already satisfied: idna<2.7,>=2.5 in
/datab/com.termux/files/usr/lib/python3.8/site-pac
kes (from requests==2.18.4->syncthing->
syncthingmanager) (2.6)
Requirement already satisfied: certifi>=2017.4.1
7 in /data/datab/com.termux/files/usr/lib/python3
.8/site-pac
kes (from requests==2.18.4->syncthing->syncthingm
anager) (2020.4.5.1)
Requirement already satisfied: urllib3<1.23,>=1.
21.1 in /data/datab/com.termux/files/usr/lib/pyt
hon3.8/site-pac
kes (from requests==2.18.4->sync
thing->syncthingmanager) (1.22)
$ ←
```

17. Ambientes Blockly (App Inventor, AppyBuilder y Thunkable)。

App Inventorは、Google Labsが作成したAndroid OS用のアプリケーションを構築するためのソフトウェア開発環境です。ユーザーは、視覚的に、そして基本的なツールのセットから、一連のブロックをリンクしてアプリケーションを作成することができます。システムは無料で、Webから簡単にダウンロードすることができます。App Inventorで作成されたアプリケーションは、プログラミング言語の知識が不要なため、非常に簡単に作成することができます。

AppyBuilderやThunkableなど、Blocklyの技術を使用している現在の環境はすべて無料版があり、その使用方法は、インターネットを介してさまざまなサイトで使用することができますし、自宅にインストールすることもできます。

Mini BlocklyChain アーキテクチャを構成するブロックは、App inventor と AppyBuilder でテストされていますが、コードの最適化のため、他のプラットフォームでも動作するはずです。

オンライン版。

App Inventor。

<https://appinventor.mit.edu/>

AppyBuilderです。

<http://appybuilder.com/>

サンカブル。

<https://thunkable.com/>

お使いのパソコン（PC）にインストールするバージョン。

<https://sites.google.com/site/aprendeappinventor/instala-app-inventor>

Blocklyブロックの開発者のための環境。

<https://editor.appybuilder.com/login.php>

18. Proof of Quantum (PQu)とは何ですか？

ポーク。- "Proof of Quantum"はMini BlocklyChainのために開発されたコンセンサスアルゴリズムで、このテストは以下のように動作するTest of Work (PoW)の変形版です。

起動時のTest of Quantum (PoQu) は、「Test of Work」(PoW)と同じアルゴリズムで実行され、「ハッシュ」と呼ばれる数学パズルのような文字列を得るために、デバイス (PC、サーバー、タブレット、携帯電話) のプロセッサを働かせることを基本としています。

「ハッシュ」は、フレーズやテキストファイル、プログラム、画像、ビデオ、サウンド、またはデジタル情報の他の多様なタイプなどのデジタル情報のいくつかのタイプを導入する際に、結果としてデータのユニークで非再現性の高い方法でそれを表すデジタル署名を表す英数字の文字として私たちを与えるアルゴリズムまたは数学的なプロセスであることを覚えておいてください。ハッシュアルゴリズムは一方向ですが、これはあなたがその署名"ハッシュ"を取得するためにデータを入力すると、その逆のプロセスが実行できないことを意味し、署名"ハッシュ"を持って、我々はこのプロパティは、我々がインターネット上で送信する情報を処理するために私たちにセキュリティ上の利点を与える情報を取得したかを知ることができません。それはどのように動作しますか？ 非セキュアなチャネルを介して情報の任意の種類を送信し、そのそれぞれの"ソースハッシュ"とそれを伴う想像してみてください、情報を受信したときに受信者は、我々はそれを"宛先ハッシュ"と呼ばれる受信した情報の"ハッシュ"を取得することができますし、両方の"ハッシュ"が同じであれば、我々は情報が送信されたチャネルで変更されていないことを確認することができます"ソースハッシュ"とそれを確認してください、情報セキュリティプロセスのこのタイプは、現在使用されている単なる例です。

現在、アルゴリズムやハッシュ処理には、セキュリティのレベルが異なる種類があります。最も使用されているか、または知られているのは、次のとおりです。MD5、SHA256、SHA512。

SHA256の例。

以下のような連鎖や文章があります。"Mini BlocklyChain"はモジュール式です。

前の文字列にSHA256ハッシュを適用すると、次のハッシュが得られます。

f41af7e61c3b02fdd5e5c612302b62a2dd52fc38f9of97cb2afd827e8804db8

上記の英数字文字列は、上記の例の文を表す署名です。

もっと例を挙げると、インターネット上のサイトを利用することができます。

<https://emn178.github.io/online-tools/sha256.html>

「テストワーク」（PoW）アルゴリズムの場合、演算能力を利用してあらかじめ定義されたハッシュを取得することで動作します。

先ほどの「ミニBlocklyChainはモジュール式」のチェーンから取った「ハッシュ」を想像してみましょう。

f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9of97cb2af827e8804db8

その先頭にあるこの"ハッシュ"に我々は単にゼロ"0"を最初に置くことである難易度のパラメータを入れて、我々は難易度が4であると言えば、それは"0000" + "ハッシュ"を持っているだろうと言うことですこれに我々はそれを"シードハッシュ"と呼ぶことになります

600%0000 フェンチオン

今、我々は文字列である入力情報を知っていることを考慮に入れて："ミニBlocklyChainはモジュラーです"我々は文字列の最後にゼロから始まる番号を追加します "0"と我々はそれを呼び出すことになりますこれにそのハッシュを取り出します "ハッシュnonce"。

f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2af827e8804db80

俺たちはハッシュ・ノンスを手に入れた

7529f3ad273fc8a9eff12183f8d6f886821900750bb6b59c1504924dfd85a7c8

次に、新しい"ハッシュ・ノンス"と"ハッシュ・シード"の比較を実行し、それらが等しい場合には、最初に等しいものを見つけたノードが現在のトランザクションを処理する実行に勝利します。このプロセスは、デバイスの確率と計算強度に基づいており、「Proof of Work」テストではすべてのノードのコンセンサス・エクイティが得られます。

種ハッシュ」と「ハッシュ・ノンス」が一致しない場合、難易度を1つ上げて「ハッシュ・ノンス」を取り除き、上げている数を「ノンス」の数と呼び、一致するか同じになるまで「種ハッシュ」と比較します。

見ての通り、「ノンス」や「増加」という数は、平等の「ハッシュ」を得るのに役立つものです。

「仕事のテスト」(PoW) アルゴリズムに基づいて、量子のテスト (PoQu) アルゴリズムは、PoW が行うように数「nonce」を取得することに基づいており、1から5までの最小レベルの難易度を使用して、これは、合意を獲得するための候補となる権利を得るためにモバイルデバイスにのみ提供しています。

量子テスト (PoQu) は、携帯電話が最小のPoWを終了し、QRNGシステムで確率番号を取得するためのパスを獲得したときにアクティブになります。

QRNG（量子乱数発生器）は、量子乱数発生器であり、このシステムは、量子力学に基づいて真の乱数を生成することに基づいていますが、そのような番号を生成するために今日最も安全なシステムです。詳細は付録「OpenQbitを用いた量子計算」を参照してください。

ミニBlocklyChainは、最小のPoWとPoQuの両方のコンセッションタイプを実装することができます。

PoQuテストは"nonce"という数字を取得することに基づいています。この数字は"Magic Number"として知られており、"Peer to Peer"システムはこの数字が正しいかどうかを確認し、QRNGサーバーポールで乱数を取得します。この乱数は全ノードに登録され、 $((\text{Node Sum} / 2)) + 1$ を含むリストが作成され、このリストの中からコンセンサスの勝者候補(PoQu)となる確率が最も高いものが選ばれ、現在のトランザクションキューが実行されます。

PoQuアルゴリズムは、QRNGの乱数が本当に乱数であることを保証するために、NIST（国立標準技術研究所）のテストも使用しています。

<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf>

Mini BlocklyChainでは、PoW用のブロックとPoQu用のブロックを実装しました。

これらのブロックは、ハッシュの種類を使用しています： SHA256 無料で使用するためには、商用利用のための SHA512 と必要に応じてハッシュの他のタイプを持っています。

HASHの概念の詳細については、以下を参照してください。

https://es.wikipedia.org/wiki/Funcion_hash

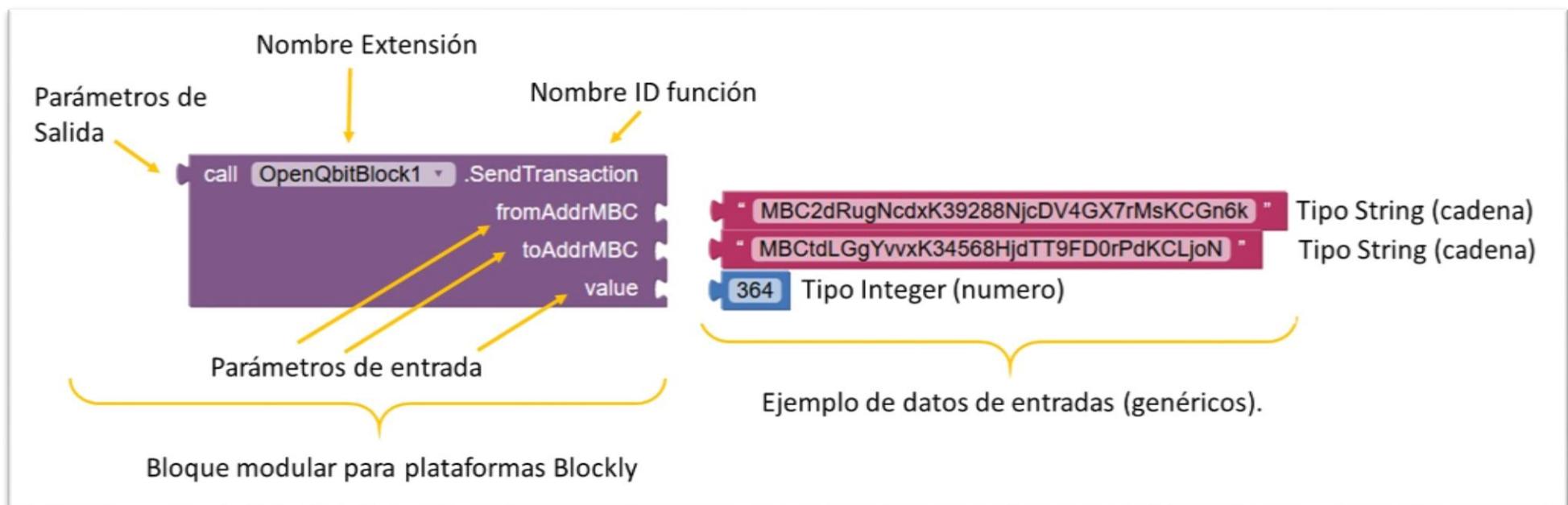
注意：携帯電話で使用されているPoW（Test of Work）は、サーバーやPCのように専用の数学処理を行わないため、最大難易度5までしか使用できません。私たちは、量子乱数発生器（QRNG）システムに入るためあなたのパスまたは許可を取得する機会を得るためにのみPoWアルゴリズムを使用して、量子乱数発生器（PoQu）アルゴリズムを実行するためにそれを使っています。

携帯電話では、ロックされて正常に反応しなくなることがあるので、最大難易度5は使用しないでください。

19. ミニBlocklyChainにおけるブロックの定義と使い方

まずは、すべてのブロックが持つであろうデータの分布、使用の構文、構成について説明します。

次の例では、入力データのタイプと同様に、モジュールブロックとその入出力パラメータを見ることができ、これらのデータのタイプはString（文字列）またはInteger（整数または10進数）です。どのように使用されているかを示し、正しく機能するように設定します。



各モジュールブロックはその説明を持ち、入力パラメータとして使用される他のブロックの必須または任意の依存関係がある場合には名前が付けられ、統合プロセスが発表されます。

内部的に"chain"と呼ばれる事前定義された配列に一時的な文字列リストを作成するためのブロック - (AddHash)。

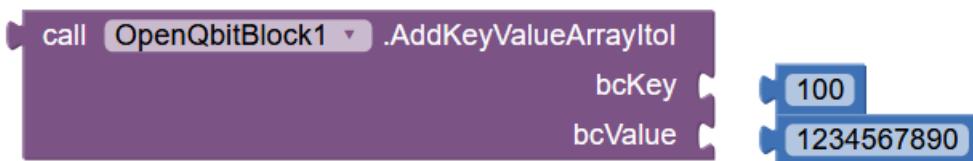


入力パラメータ：ブロック <文字列>

出力パラメータ：該当しません。

説明：ハッシュや文字列の一時的な配列を、内部的に「chain」と呼ばれる定義済みの配列に格納するためのブロック。

キー値配列(Integer-Integer)を作成するブロック - (AddKeyValueArrayItol)



入力パラメータ: bcKey <整数>, bcValue <整数>

出力パラメータ：入力時に入力された値を返します。

説明します。これは一時的なキー値の配置であり、内部名「Itol_UTXO」で定義されており、UTXOトランザクションを処理するのに便利である。

キー値配列を作成するブロック(Integer-String) - (AddKeyValueArrayItos)

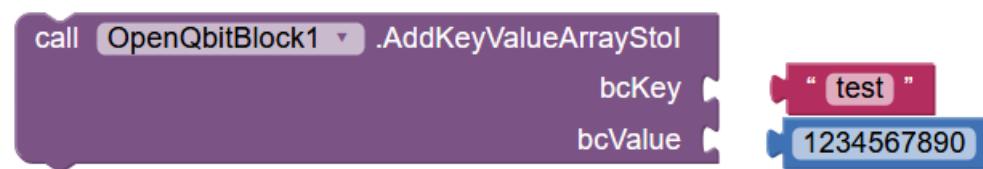


入力パラメータ: bcKey <整数>, bcValue <文字列>

出力パラメータ：入力時に入力された値を返します。

説明します。これは一時的なキー値の配置であり、内部名「`ItoS_UTXO`」で定義されており、UTXOトランザクションを処理するのに便利である。

キー値配列を作成するブロック(String-Integer) - (`AddKeyValueArrayItoi`)

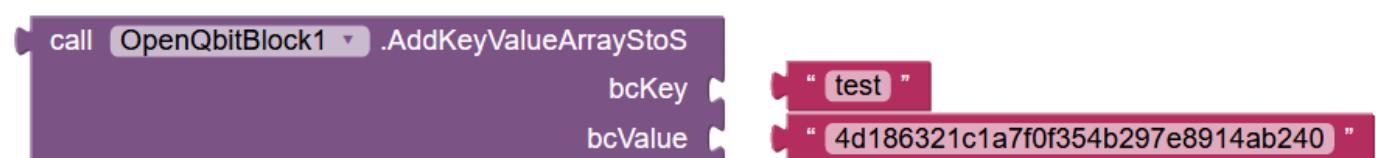


入力パラメータ: `bcKey` <文字列>, `bcValue` <整数>

出力パラメータ：入力時に入力された値を返します。

説明します。これは一時的なキー値の配置で、内部名 "Stol_UTXO" で定義されており、UTXO トランザクションを処理するのに便利です。

キー値配列を作成するブロック(String-String) - (`AddKeyValueArrayStoS`)



入力パラメータ: `bcKey` <String>, `bcValue` <String>.

出力パラメータ：入力時に入力された値を返します。

説明します。これは一時的なキー値の配置で、内部名 "StoS_UTXO" で定義されており、UTXO トランザクションを処理するのに便利です。

10進乱数量子数生成ブロック - (ApiGetQRNGdecimal)



入力パラメータ : qty <整数>

出力パラメータ: 入力されたランダムな量子十進数の量"qty"をJSON形式で0と1の範囲内で与えます。

例。

```
qty = 5; output: {"result": [0.5843012986202495, 0.7746497687824652, 0.05951126805960929, 0.1986079055812694, 0.03689783439899279]}
```

説明: 量子乱数発生器 (QRNG) API

10進乱数量子数生成ブロック - (ApiGetQRNGinteger)



入力パラメータ: qty <整数>, min <整数>, max <整数>

出力パラメータ: 入力されたランダムな量子整数の量"qty"を与えます。

例。

```
qty = 8, min = 1, max = 100; output: {"result": [3, 53, 11, 2, 66, 44, 9, 78]}
```

説明: 量子乱数発生器 (QRNG) API

文字列のハッシュブロック「SHA256(ApplySha256)となります。

call OpenQbitBlock1 .ApplySha256

input

" Mini BlocklyChain es modular "

入力パラメータ：入力 <文字列>

出力パラメータ：文字列の"SHA256"ハッシュを計算します。

例。

入力 = "ミニBlocklyChainはモジュール式"

出力 : f41af7e61c3b02fdd5e5c612302b62a2dd52fcb38f9de97cb2afd827e8804db8

説明：ハッシュ"SHA256"を削除する機能です。ShaまたはSHAを参照してください。セキュアハッシュアルゴリズム（Secure Hash Algorithm）は、米国国家安全保障局が設計したハッシュ関数のセット。SHA256は256ビットのアルゴリズムを使用しています。

定義済みの内部配列「チェーン」をクリーンアップするブロック（ClearBlockList）。

call OpenQbitBlock1 .ClearBlockList

入出力パラメータ：該当なし

説明：内部に一時的に配置された「鎖」を持つ要素をすべて削除します。

定義済みの内部配列(Integer-Integer)をクリーンアップするブロック - (ClearItol)。

call OpenQbitBlock1 .ClearItol

入出力パラメータ：該当なし

説明：内部仮配置「Itol_UTXO」を持つ要素を全て削除します。

定義済みの内部配列(Integer-String)をクリーンアップするブロック - (ClearItoS)

call OpenQbitBlock1 .ClearItoS

入出力パラメータ：該当なし

説明：内部仮配置「ItoS_UTXO」を持つ要素を全て削除します。

定義済みの内部配列(String-Integer)をクリーンアップするブロック - (ClearStol)。

 call OpenQbitBlock1 .ClearStol

入出力パラメータ：該当なし

説明：内部仮配置「Stol_UTXO」を持つ要素を全て削除します。

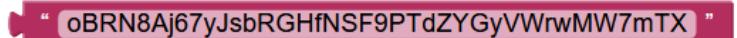
定義済みの内部配列(String-String)をクリーンアップするブロック - (ClearStoS)

 call OpenQbitBlock1 .ClearStoS

入出力パラメータ：該当なし

説明：内部仮配置「StoS_UTXO」を持つ要素を全て削除します。

文字列をBase10にデコードするブロック。(DecodeBase58)

 call OpenQbitBlock1 .DecodeBase58
input 

入力パラメータ：input<String>

出力パラメータ：ブロック (EncodeBase58) で使用された元の文字列を出力します。

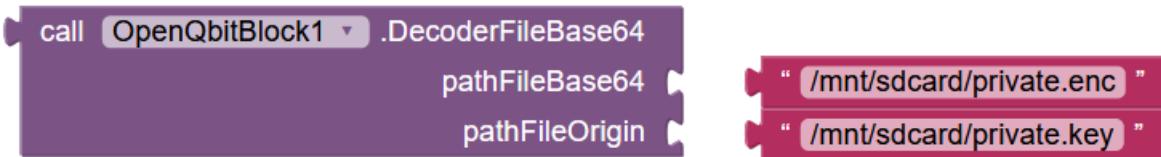
例。

入力 = oBRN8Aj67andJsbRGHfNSF9PTdZYGandVWrwMW7mTX

出力：「ミニBlocklyChainはモジュラーです。」

説明: Base58 文字列をブロック内で指定された元のテキストに変換します (EncodeBase58)

Base64アルゴリズム(DecoderFileBase64)でファイルをデコードするブロック。



入力パラメータ：`pathFileBase64 <String>`, `pathFileOrigin <String>`

出力パラメータ：ブロックに入力されたソースファイル（EncoderFileBase64）

説明: Base64 ファイルをブロック内に挿入された元のファイル (EncoderFileBase64) に変換します

◦

定義済みの内部配列「chain」が空かどうかを知るためのブロック。(EmptyBlockList)



入力パラメータ：該当しません。

出力パラメータ：空の場合は"True"を、データがある場合は"False"を返します。

説明： 定義済みの一時的な内部配列「chain」に要素があるかどうかを聞くブロック。

文字列をBase58にエンコードするブロック。(EncodeBase58)を使用しています。



入力パラメータ：`input<String>`

出力パラメータ：ブロック (EncodeBase58) で使用された元の文字列を出力します。

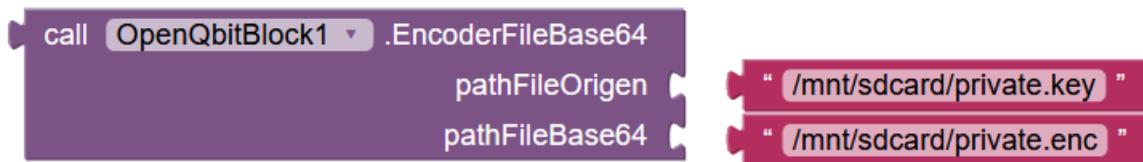
例。

入力 = 「ミニBlocklyChainはモジュラーです。」

Output: oBRN8Aj67yJsbRGHfNSF9PTdZYGyVWrwMW7mTX

Description: Base58で胸の鎖を鎖に変換します。Base58アルゴリズムは、大きな整数を英数字のテキストとして表現するために使用されるバイナリからテキストへのエンコード方式のグループで、ビットコインで使用するためにサトシ・ナカモトによって導入されました。

Base64 アルゴリズムでファイルをエンコードするためのブロック (EncoderFileBase64)。

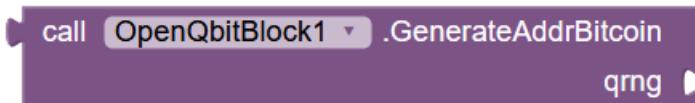


入力パラメータ: pathFileOrigin <String> , pathFileBase64 <String> 入力パラメータ:
pathFileOrigin <String> , pathFileBase64 <String> , pathFileBase64 <String> , pathFileBase64
<String> , pathFileBase64 <String> , pathFileBase64 <String> , pathFileBase64 <String> ,
pathFileBase64 <String> , pathFileBase64 <String> , pathFileBase64 <String> , pathFileBase64
<String> , pathFileBase64 <String> , pathFileBase64 <String> , pathFileBase64 <String> ,
pathFileBase64 <String> , pathFileBase64 <String> , pathFileBase64 <String> , pathFileBase64
<String> , pathFileBase64 <String> , pathFileBase64 <String> , pathFileBase64 <String> ,
pathFileBase64 <String> , pathFileBase64 <String> , pathFileBase64 <String>

出力パラメータ : Base64エンコードされたファイル

説明: 任意の形式のソースファイルをBase64ファイルに変換します。ファイル名は任意で、ユーザーが選択することができます。

ユーザーアドレスのブロック生成（GenerateAddrBitcoin）。



必須単位：ブロック（ApiGetQRNGinteger）。

依存関係（オプション）：OpenQbitFileEncryption 拡張モジュール、OpenQbitFileDecryption 拡張モジュール、OpenQbitSQLite 拡張モジュール。

入力パラメータ: qrng <必須依存性

出力パラメータ：34文字の英数字トランザクションアドレスとkeyStore使用アドレス

説明：ユーザーと秘密鍵生成器（取引を送信するためのデジタル署名）と公開鍵（取引を行うための公開アドレス）のための新しいビットコインの一般的な取引アドレスを作成するためのブロックです。このキージェネレーターは、基本的にはデジタルウォレットや通称「ウォレット」で使用するためのアドレスジェネレーターです。

このブロックは、Quantum Random Number Generator (QRNG) ブロックと組み合わせて使用し、2つの出力パラメータを KeyStore に挿入する必要があります。

KeyStoreは、秘密鍵を16進数形式で保存するデータベースです。

KeyStoreに格納されている16進数のアドレスの例

024C8E05018319ARD4BB04E184C307BFF115976A05F974C7D945B5151E490ERD

このベースは、ローカルユーザーのみが使用され、情報は暗号化され、このプロセスは、OpenQbitSQLite拡張子とOpenQbitAESEncryptionとOpenQbitAESDecryption情報暗号化拡張子を使用しています。

KeyStoreを作成するには、付録の「KeyStoreの作成」を参照してください。生成されたアドレスは同じビットコインアドレスアルゴリズムを使用し、最初のビットコインアドレス識別子は「1」となります。
。

ブロック (**G e n e r a t e A d d r B i t c o i n**) で生成されたアドレスは 34 文字の英数字で、以下のように 33 文字の英数字とビットコイン識別子の 1 とで構成されている。

12dRugNcdxK39288NjcDV4GX7rMsKCGn6k

ユーザーアドレス生成ブロック (**GenerateAddrEthereum**)。



強制依存：トークンの入手先：<https://accounts.blockcypher.com/signup>

依存関係 (オプション)：OpenQbitFileEncryption 拡張モジュール、OpenQbitFileDecryption 拡張モジュール、OpenQbitSQLite 拡張モジュール。

入力パラメータ: **token** < 必須依存 - String >

出力パラメータ：40英数字文字トランザクションアドレスは、私たちに共通のアドレスの42文字を与えるであろう初期のEthereumインジケーター「0x」が含まれていません。また、公開鍵と秘密鍵を返します。

例。

{

"**private**": "227ac59f480131272003c2d723a7795ebd3580acaab62b5c537989e2ce4e08ef"。

"**公開**":

"04e2d55ebccd32a7384e096df559cc36b856c64a16e5b402e10585dc3ea055672aafa84df8a859531570a650a8ab1e7a22949100efa1aa5f072c035551cac1ce".

"**address**": "14e150399b0399f787b4d6fe30d8b251375f0d66"}

説明：ユーザと秘密鍵生成器（トランザクションを送信するためのデジタル署名）と公開鍵（トランザクションを作成するための公開アドレス）のための新しいトランザクションアドレスを作成す

るためのブロック。このキージェネレーターは外部APIで、Wifiやモバイル接続が必要で、基本的にはデジタルウォレットや通称「ウォレット」で使用するためのアドレスジェネレーターです。

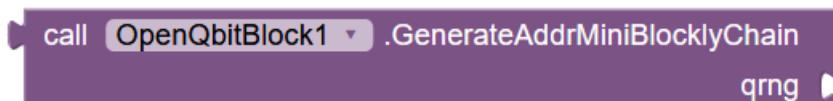
作成できるのは、秘密鍵を16進数形式で格納したデータベースであるaKeyStoreです。

KeyStoreに格納されている16進数のアドレスの例

0x14e150399b0399f787b4d6fe30d8b251375f0d66

秘密鍵はローカルユーザーのみが使用し、情報は暗号化されますが、このプロセスはOpenQbitSQLite拡張とOpenQbitAESEncryptionとOpenQbitAESDecryption情報暗号化拡張を使用しています。

ユーザアドレス生成ブロック（GenerateAddrMiniBlocklyChain）。



必須単位 : ブロック
（ApiGetQRNGinteger）。

依存関係（オプション）：OpenQbitFileEncription 拡張モジュール、OpenQbitFileDecryption 拡張モジュール、OpenQbitSQLite 拡張モジュール。

入力パラメータ: qrng <必須依存性

出力パラメータ：36文字の英数字のトランザクションアドレスとkeyStoreの使用アドレス。レビュー
ブロック方式（PairKeysMBC）。

説明：ユーザと秘密鍵生成器（トランザクションを送信するためのデジタル署名）と公開鍵（トランザクションを作成するための公開アドレス）の新しいトランザクションアドレスを作成するためのブロック。このキージェネレーターは、基本的にはデジタルウォレットや通称「ウォレット」で使用するためのアドレスジェネレーターです。

このブロックは、Quantum Random Number Generator (QRNG) ブロックと組み合わせて使用し、2つの出力パラメータを KeyStore に挿入する必要があります。

KeyStoreは、秘密鍵を16進数形式で保存するデータベースです。

KeyStoreに格納されている16進数のアドレスの例

024C8E05018319ARD4BB04E184C307BFF115976A05F974C7D945B5151E490ERD

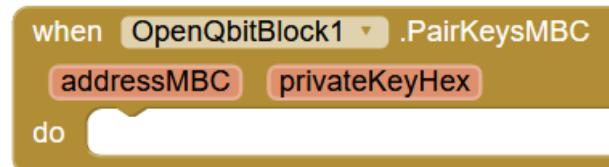
このベースは、ローカルユーザーのみが使用され、情報は暗号化され、このプロセスは、OpenQbitSQLite拡張子とOpenQbitAESEncryptionとOpenQbitAESDecryption情報暗号化拡張子を使用しています。

KeyStoreを作成するには、付録の「KeyStoreの作成」を参照してください。生成されたアドレスは同じビットコインアドレスアルゴリズムを使用しますが、唯一の違いは、「1」または「3」であるビットコインアドレスの識別子がMini BlocklyChainの識別子「MBC」に変更されることです。

ブロック (GenerateAddrMiniBlocklyChain) で生成されるアドレスは、3 6 文字の英数字であり、3 3 文字の英数字と、MiniBlocklyChainの識別子大文字「MBC」の 3 文字で構成され、以下のようになっている。

Mbc2dRugNcdxK39288NjcDV4GX7rMsKCGn6k

ブロック方式(PairKeysMBC)はブロック出力(GenerateAddrMiniBlocklyChain)です。



入力パラメータ：該当しません。

出力パラメータ: addressMBC <String>、privateKeyHex <String>。

説明します。公開ユーザアドレスをMini BlocklyChain形式で、秘密鍵を16進数形式で配信。

の例を示します。

addressMBC: MBC2dRugNcdxK39288NjcDV4GX7rMsKCGn6k

privateKeyHex:

024C8E05018319ARD4BB04E184C307BFF115976A05F974C7D945B5151E490ERD

ユーザー地址のブロック生成（GenerateKeyValuePair）。

```
call OpenQbitBlock1 .GenerateKeyValuePair
      qrng
```

必須単位：ブロック（ApiGetQRNGinteger）。

依存関係（オプション）：OpenQbitFileEncription 拡張モジュール、OpenQbitFileDecryption 拡張モジュール、OpenQbitSQLite 拡張モジュール。

入力パラメータ: qrng <必須依存性

出力パラメータ：ローカルユーザの公開鍵と主鍵を提供します。

説明: ECCアルゴリズム⁽¹⁾を使用し、16進数形式で公開鍵と主鍵を生成します。

⁽¹⁾ 楕円曲線暗号（ECC）は、楕円曲線の数学に基づいた非対称暗号や公開鍵暗号の一種です。

トランザクションに署名するためのブロック（GenerateSignature）。

```
call OpenQbitBlock1 .GenerateSignature
```

必要な依存関係：ブロック（GenerateKeyPairs）、ブロック（SenderLoadKeyPair）、ブロック（RecipientLoadKeyPair）。使用する前に、これらのブロックの前書きをしてください。

入力パラメータ：<必須チェックの依存関係>を入力します。

出力パラメータ：出力はありません。

説明: それは、受信者へのトランザクションで送信された資産に適用される送信者からのデジタル署名を生成し、この署名は、トランザクションがネットワークのいくつかのノードによって処理されているときに確認され、この署名を使用して、ミニBlocklyChainシステムは、資産がそれで変更されていないことを保証します送信され、それは送信者と受信者の関係に準拠しており、別のデジタルアドレスに流用または適用されていません。

ユーザーの資産残高の合計を取得するブロック（GetBalance）。

```
call OpenQbitBlock1 .GetBalance
    fromAddrMBC "MBC2dRugNcdxK39288NjcDV4GX7rMsKCGn6k"
```

必須依存関係：ブロック（ConnectorTransactionTail）。

入力パラメータ: fromTransTail <Array String>, <必須ボードの依存性

出力パラメータ：各トランザクションの入出庫資産費をチェックします。

説明: ユーザーが送信する資産を持っているか、または彼が受け取る資産が彼の残高に追加されているかどうかを承認するために残高をチェックします。

携帯電話からq22を取得するためのブロック。（GetDeviceID）

```
call OpenQbitBlock1 .GetDeviceID
```

入力パラメータ：該当なし

アウトバウンドパラメータ：携帯電話の内部識別子を配信します。

説明: 各デバイスに固有のIMEI携帯電話識別子を提供します。

文字列からRIPEMD-160ハッシュを削除するためのブロック。（リンペムド-160）



入力パラメータ: str <文字列>

出力パラメータ：文字列の"RIPEMD-160"ハッシュを計算します。

例。

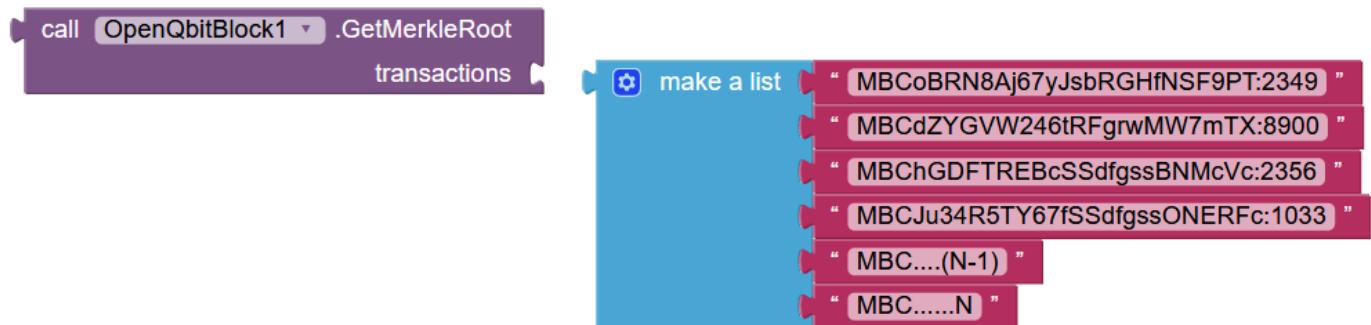
入力 = "ミニBlocklyChainはモジュール式"

出力 : ae29436e4b8ea8ed6143f3f92380dfa2f4f47336

Description: "RIPEMD-160"のハッシュを取得します。このハッシュは、ビットコインとミニBlocklyChainの有効なアドレスを生成する過程で使用されます。

RIPEMD-160 (RACE Integrity Primitives Evaluation Message Digestの頭文字をとったもの) は、160ビットのメッセージダイジェストアルゴリズム（および暗号ハッシュ関数）です。

メルクラーの木を計算するためのブロック。(GetMerkleRoot)



入力パラメータ: トランザクション <Array String>

出力パラメータ：ハッシュ型「SHA256」を配信します。

例。

入力 = トランザクションキュー、処理されるべきすべてのトランザクションの配置。

出力 : b4a44c42b6070825f763cd118d6ab49a8e80bb7cdc0225064f8e042b94196bd

説明: Merkleの木アルゴリズムを用いて"SHA256"ハッシュを取得する

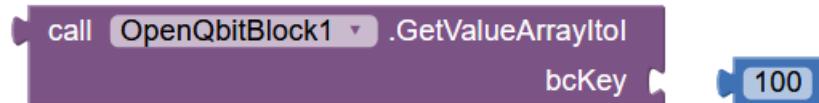
Merkle Hash Tree はバイナリまたは非バイナリのデータツリー構造で、リーフではない各ノードには、子ノードのラベルまたは値（リーフノードの場合）を連結したハッシュでタグ付けされています。これらはハッシュリストとハッシュ文字列の一般化です。

これにより、ツリールートノードのハッシュ値である1つのハッシュ値に、多数の別々のデータをリンクさせることができます。大規模データ構造の内容を安全かつ効率的に検証する方法を提供します。実用的なアプリケーションでは、ルートノードのハッシュは通常、その整合性を確保し、検証が完全に信頼できることを保証するために署名されています。リーフノードが与えられたハッシュツリーの一部であることを証明するには、ツリーのノード数の対数に比例するデータ量が必要です。

現在、メルクルツリーの主な用途は、ピアツーピアナットワーク内の他のピアから受信したデータブロックを、損傷や改ざんされずに安全に受信できるようにすることである。

これは、処理すべきトランザクションを持つキューが変更されていないことを検証し、Mini BlocklyChain ネットワークのすべてのノードに分散するために、その完全性を保証するために使用されます。すべてのノードは、適用される各トランザクションの整合性を保証するために、このアルゴリズムを実行する必要があります。

定義済み配列「Itol_UTXO」から値を取得するブロック（GetValueArrayItol）。

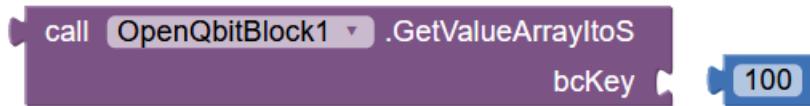


入力パラメータ： bcKey <整数>

出力パラメータ： 与えられた数値を入力として、ラベルに格納されている値<Integer>を返します。

説明します。これは、UTXO トランザクションを処理するのに便利な内部名「Itol_UTXO」で定義されている一時的なキー値配置へのリクエストである。

定義済み配列「ItoS_UTXO」から値を取得するブロック（GetValueArrayItoS）。



入力パラメータ： bcKey <整数>

出力パラメータ： ラベルに格納されている値 <String> を、入力として指定された数値で返します。

説明します。これは、UTXO トランザクションを処理するのに便利な内部名「ItoS_UTXO」で定義されている一時的なキー値配置へのリクエストである。

定義済み配列「StoI_UTXO」から値を取得するブロック（GetValueArrayStoI）。



入力パラメータ： bcKey <文字列>

出力パラメータ： 指定された名前を入力として、ラベルに格納されている値 <Integer> を返します。

説明します。これは、内部名"StoI_UTXO"で定義されている一時的なキー値配置へのリクエストであり、UTXO トランザクションを処理するのに便利である。

定義済み配列「StoS_UTXO」から値を取得するブロック（GetValueArrayStoS）。



入力パラメータ： bcKey <文字列>

出力パラメータ： 入力として指定された名前のラベルに格納されている値 <String> を返します。

説明します。これは、内部名"StoS_UTXO"で定義されている一時的なキー値配置へのリクエストであり、UTXO トランザクションを処理するのに便利である。

ブロックチェーンのブロックバリデータ。(IsChainValid)

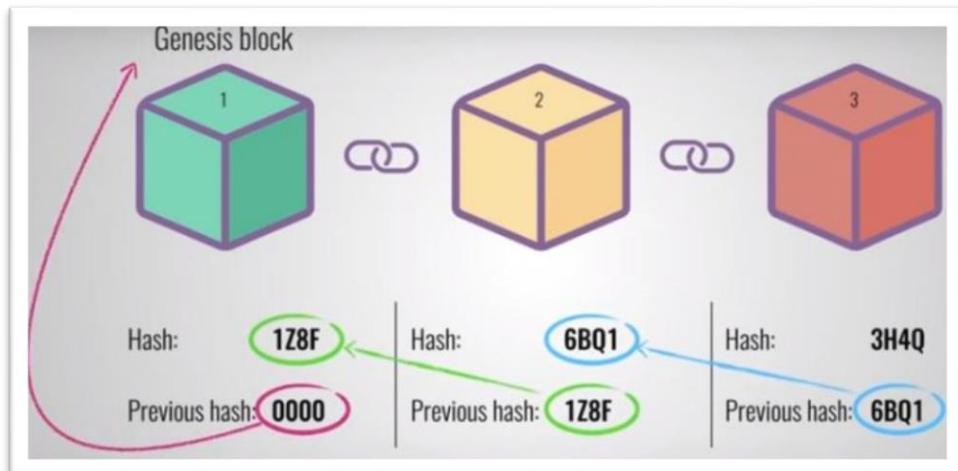
```
call OpenQbitBlock1 .IsChainValid
```

必要な依存関係：ブロック（GenerateKeyPairs）、ブロック（SenderLoadKeyPair）、ブロック（RecipientLoadKeyPair）、ブロック（GenerateSignature）。使用前にこれらのブロックの前書きをしてください。

入力パラメータ：該当しません。

出力パラメータ：ブロック文字列の検証が正しい場合は"True"を、検証に失敗した場合は"False"を出力します。

説明: それは以前にブロックチェーンシステムに挿入されているコンポーネントの検証を提供し、この検証は、全体のシステムの中で最も重要です。ブロックチェーンの検証がどのように行われているのか、あるいは一般的に知られている方法（BlockChain）。それはすべてのシステムの核となる部分です。



前の画像で見たように、ストレージシステムに追加された各ブロックは、ハッシュアルゴリズムによって前のブロックと関連しています。

例えば、新たに追加するブロックを作成する場合、新たな情報を表すハッシュは、新たな情報ブロックのハッシュ + 前のハッシュを取ることで得られる。このタイプのリンクを使用すると、ブロックチェーンのストレージの変更を最小限に抑えることができ、非常に高い効果的なデータセキュリティを実現します。

以下は、ブロックの文字列がSQLiteデータベースに格納されている方法の例です、我々は、前のハッシュが挿入された最後のブロック（処理されたトランザクションの最後のキー）の新しいハッシュにリンクされている方法を見てみましょう。"prevhash" および "newhash" カラムのそれぞれのハッシュは、その時点で処理されたトランザクションキーの情報を表す。

SQLite Expert Personal 5.3 (x64)

File View Database Object SQL Transaction Tools Help

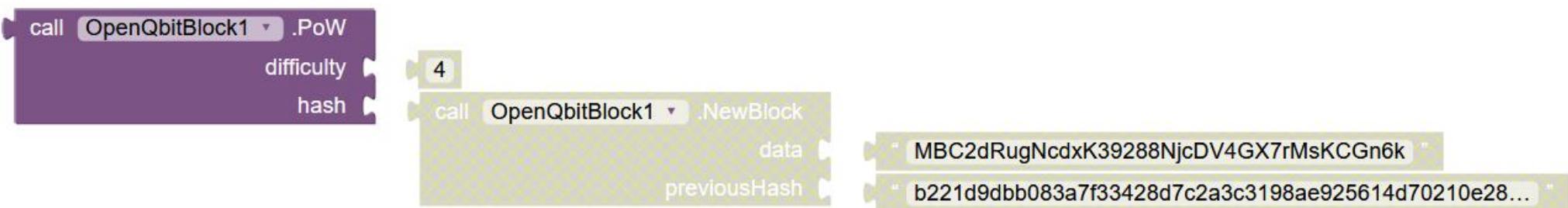
Database: miniblock Table: nblock File: C:\memo\thunkable\miniblock.db

miniblock

rowid	id	prevhash	newhash	opera	trans	balan
1	1	0767c864cef0334f27473902eb9868e7	bdc9065d20a4cd037bb1a7538486403e	2	0	0
2	2	bdc9065d20a4cd037bb1a7538486403e	6619f4809d73a267a4b9ac554bb4523a	1	5	5
3	3	6619f4809d73a267a4b9ac554bb4523a	4d186321c1a7f0f354b297e8914ab240	1	5	5

前のハッシュは新しいハッシュに関連しています

ブロックでPoW作業テストを行い、新たなブロックを採掘する。(Pow)



必須の依存関係: NewBlock.このブロックを使用する前に前置きをしてください。

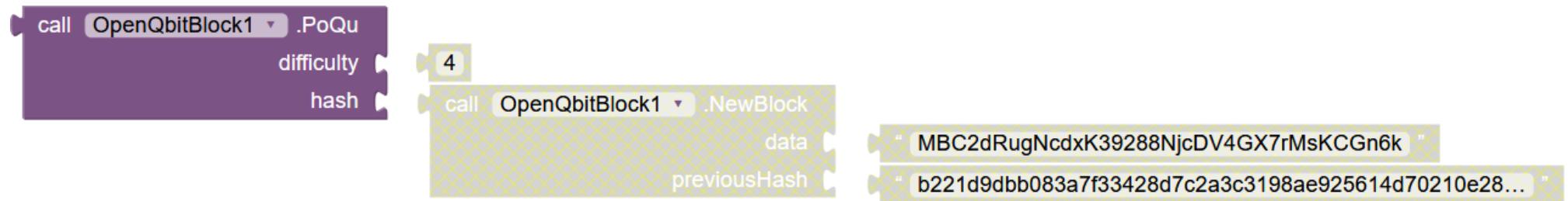
入力パラメータ : 難易度 <整数>、ハッシュ <強制依存度>

出力パラメータ: 入力パラメータの難易度で指定された#Number of "zero"で構成されたハッシュ"SHA256"を出力として与えます。

説明します。このブロックは、新しいブロックは、我々は以前にPoW（仕事の証明）のプロセスとして説明されているものである"マイニング"と呼ばれる活動を実行し、セクションを参照してください 量子 (PQu) の証明とは何ですか？

マイニングやPoWの実行は、ノードに数学的なパズルを解くタスクを与える概念です。Mini BlocklyChainの場合、最初にそれを解いた人は誰でも、処理を待っているトランザクションキューを処理することができることの勝者に選ばれるために、プロセスを継続する機会を得ます。

ブロックでPoW作業テストを行い、新たなブロックを採掘する。(PoQu)



必須の依存関係: **NewBlock**.このブロックを使用する前に前置きをしてください。

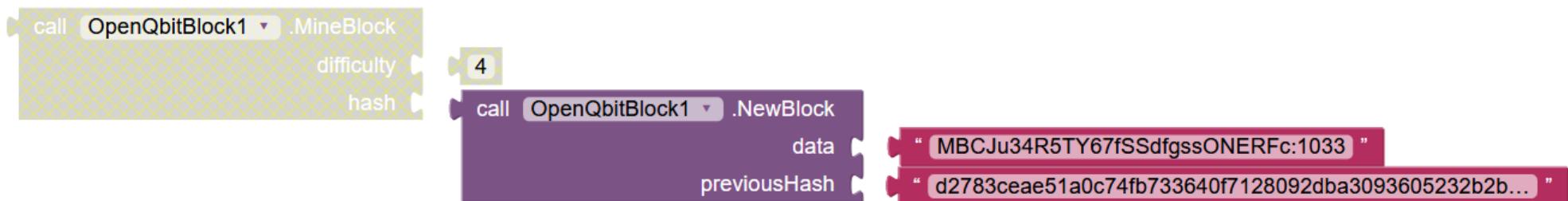
入力パラメータ：難易度 <整数>、ハッシュ <強制依存度>

出力パラメータ：数"マジックナンバー"の結果と、0と1の範囲内にあるランダム量子数16桁10進数型、例（0. 5843012986202495）と、入力パラメータの難易度で与えられた"ゼロ"の # 数で構成されたハッシュ"SHA256"の結果。

説明します。このブロックは、新しいブロックは、我々が以前にPoW（仕事の証明）のプロセスとして説明しているものですが、このプロセスは、最小1、最大5の範囲内にすることができます難易度のレベルを満たすために適した数"nonce"を計算した後に生成されたランダムな量子数の関数を呼び出します。セクションを参照してください 量子（PQu - プルーフ量子）の証明とは何ですか？

マイニングやPoWやPoQuを実行することは、数学的なパズルを解くタスクをノードに与える概念で、ブロックチェーンシステムの場合、最初にそれを解いたノードは、処理を待っているトランザクションキューを処理できる勝者に選ばれるために、処理を継続する機会を得ることができます。

新しいブロックを作成するためのブロック（NewBlock）。



入力パラメータ : data <String>, previousHash <String>

出力パラメータ: SHA256 (データ (入力パラメータ) + previousHash (入力パラメータ) + IMEI (内部パラメータ) + MerkleRoot (内部パラメータ) のように計算されたハッシュを提供します。)

説明します。このブロックは、処理対象の新規ブロックの作成を実行します。データの場合は入力パラメータの文字列で構成される「SHA256」ハッシュを出力として与え、前回のハッシュは既に処理済みでMini BlocklyChainブロック文字列システムに格納されている前回のトランザクション文字列のハッシュを基にしています。これらの 2 つは変数パラメータですが、固定され、情報とシステムの整合性を確保する 2 つの内部システム パラメータがありますが、これらの 2 つの内部パラメータは、携帯電話の一意の ID と処理されているトランザクション キューの merklet ツリーのハッシュです。

ノードの 総ローカル・トランザクションの ブロック (NumberTransactions)

```
call [OpenQbitBlock1 .NumberTransactions]
      calculateNumTransactions
```

必須依存関係：ブロック (AddKeyValueArray StoS)。このブロックを使用する前に前置きをしてください。

入力パラメータ：<必須依存>。

出力パラメータ：ノードへのローカルトランザクションの合計を返します。

説明します。ノードのローカル・アカウントにのみ適用されるトランザクションの合計を提供します。

バイナリファイルの受信者のアドレスを読み込むためのブロックです。(RecipientLoadKeyPair)

```
call [OpenQbitBlock1 .RecipientLoadKeyPair]
```

入力パラメータ：該当しません。

出力パラメータ：秘密鍵と公開鍵をBase64形式で返します。

説明: バイナリファイルから受信者のプライベートアドレスとパブリックアドレスを入力パラメータとして取得します。

内部仮配置"chain"の要素を削除するブロック(RemoveHash)。

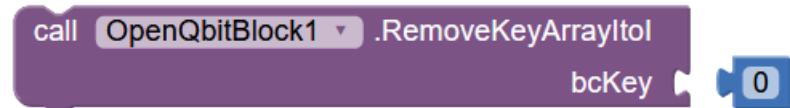
```
call [OpenQbitBlock1 .RemoveHash]
      block "test"
```

入力パラメータ：ブロック<文字列>。

出力パラメータ：該当しません。

説明: バイナリファイルから受信者のプライベートアドレスとパブリックアドレスを入力パラメータとして取得します。

内部仮配置の要素削除ブロック "ItoI_UTXO" (RemoveKeyArrayItoi)

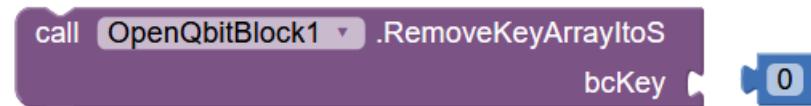


入力パラメータ: bcKey <整数>.

出力パラメータ: 該当なし、要素削除の型 <整数>

説明 : 内部仮配置「ItoI_UTXO」の数値ラベルに関連付けられた要素を削除する。

内部仮配置"ItoS_UTXO"の要素を削除するブロック(RemoveKeyArrayItoS)。

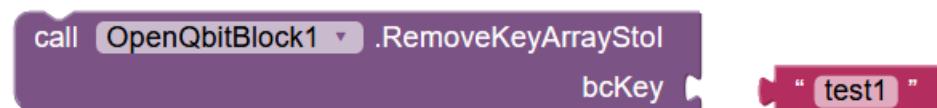


入力パラメータ: bcKey <整数>.

出力パラメータ: 該当なし、削除する要素の種類 <String>

説明 : 内部仮配置「ItoS_UTXO」の数値ラベルに関連付けられた要素を削除する。

内部仮配置の要素削除用ブロック "StoI_UTXO" (RemoveKeyArrayStoi)

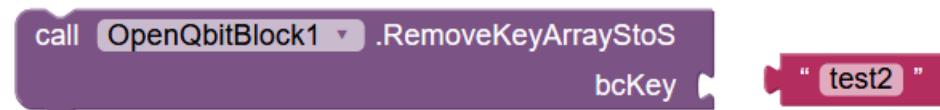


入力パラメータ: bcKey <文字列>.

出力パラメータ: 該当なし、要素削除の型 <整数>

説明：内部仮配置「StoI_UTXO」の数値ラベルに関連付けられた要素が削除される。

内部仮配置"StoS_UTXO"の要素削除用ブロック (RemoveKeyArrayStoS)

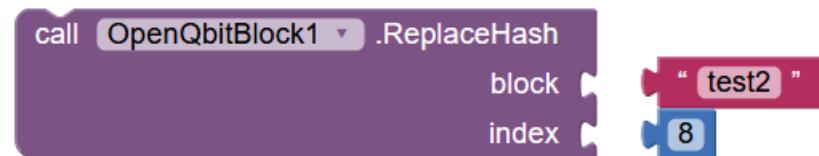


入力パラメータ: bcKey <文字列>.

出力パラメータ: 該当なし、削除する要素の種類 <String>

説明: 内部仮配置ラベル"StoS_UTXO"の要素を削除します。

内部仮配置「チーン」の値を置き換えるブロック (ReplaceHash) 。

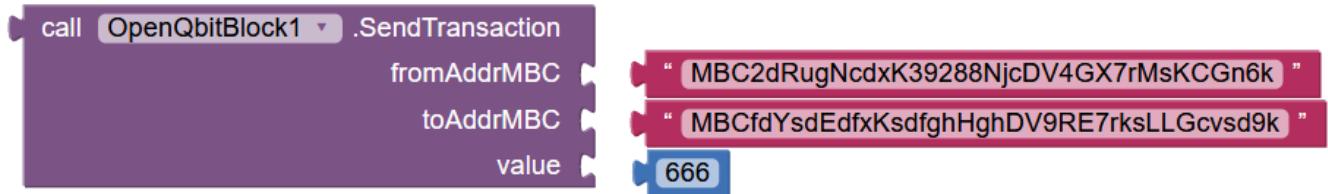


入力パラメータ : block <文字列>, index <整数>

出力パラメータ : 該当しません。

説明 : インデックス"index"に関連付けられた要素は、一時的な内部配置"chain"のラベル "block"の値に置き換えられます。

新規トランザクション (SendTrasaction) を開始 (送信) するためのブロック。



入力パラメータ : fromAddrMBC <文字列>, toAddrMBC <文字列>, value <整数>

出力パラメータ: トランザクションが正常に送信された場合は値"True"を返すか、それは彼が送信された成功しなかった場合は"False"を返します。

説明 : 送信者アドレスと受信者アドレスをバイナリ形式に変換し、処理対象のトランザクションキューにアタッチするブロック。

送信者のアドレスをバイナリファイルで読み込むブロック。(SenderLoadKeyPair)



入力パラメータ : 該当しません。

出力パラメータ : 秘密鍵と公開鍵をBase64形式で返します。

説明: バイナリファイルから送信者のプライベートアドレスとパブリックアドレスを入力パラメータとして取得します。

仮配置「鎖」の要素番号を取得するブロック(SizeBlockList)。



入力パラメータ : 該当しません。

出力パラメータ : 内部配列「chain」の要素番号を返します。

説明: 内部配列 chain の要素番号を取得する。

仮配置「I t o I _ U T X O」の要素番号を取得するブロック（S i z e I t o I

call OpenQbitBlock1 .SizeItol

入力パラメータ：該当しません。

出力パラメータ：内部配列「Itol_UTXO」の要素番号を返す。

説明：内部配列"Itol_UTXO"の要素番号を取得する。

仮配置「I t o S _ U T X O」の要素番号を取得するブロック（S i z e I t o S

call OpenQbitBlock1 .SizeIts

入力パラメータ：該当しません。

出力パラメータ：内部配列"ItoS_UTXO"の要素番号を返す。

説明：内部配列"ItoS_UTXO"の要素番号を取得する。

仮配置"StoI_UTXO"の要素番号を取得するブロック(SizeStoI)

call OpenQbitBlock1 .SizeStoI

入力パラメータ：該当しません。

出力パラメータ：内部配列"StoI_UTXO"の要素番号を返します。

説明：内部配列 "StoI_UTXO" の要素番号を取得する。

仮配置"StoS_UTXO"の要素番号を取得するブロック(SizeStoS)

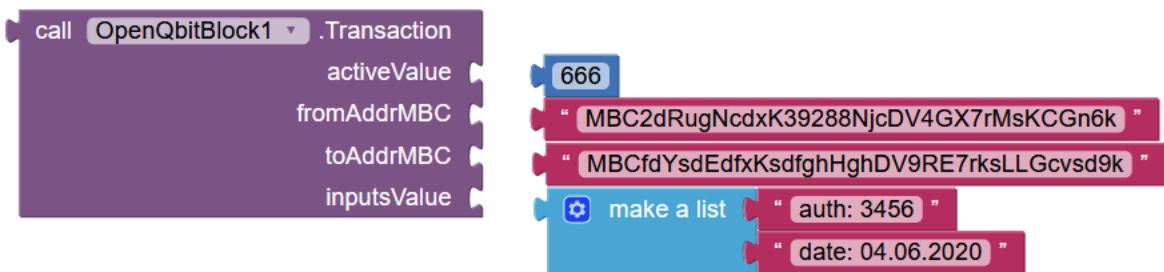
call OpenQbitBlock1 .SizeStoS

入力パラメータ：該当しません。

出力パラメータ：内部配列"StoS_UTXO"の要素番号を返します。

説明: 内部配列"StoS_UTXO"の要素番号を取得する。

値を追加したトランザクション（Transaction）の作成をブロックします。



入力パラメータ: activeValue <整数>, fromAddrMBC <文字列>, toAddrMBC <文字列>, inputsValue <配列文字列>.

出力パラメータ: バイナリ形式のアドレスと inputsValue を文字列 'String' に変換した値を与え、 ActiveValue 値のハッシュ "SHA256" を与えます。

説明: ノードが処理する新しいトランザクションを準備します。

ミニBlocklyChainアドレス検証ブロック(ValidateAddMiniBlocklyChain)



入力パラメータ：Mini BlocklyChain形式のユーザーID。

出力パラメータ: アドレスが正しい形式の場合は"True"、アドレスが無効な場合は"False"を返します。

説明： 入力された Mini BlocklyChain アドレスが正しいかどうかを検証するために、このブロックはアルゴリズムを適用して、アドレスが Mini BlocklyChain システムで使用されるアドレス作成メカニズムを使用して作成されたかどうかを検証します。

ビットコインのアドレス検証ブロック。(ValidateAddBitcoin)

```
call OpenQbitBlock1 .ValidateAddressBitcoin
      addr "12dRugNcdxK39288NjcDV4GX7rMsKCGn6k"
```

入力パラメータ: **addr** <文字列>, ビットコインでフォーマットされたユーザアドレス(初期識別子"1"のビットコインアドレスを受け付けます。識別子"3"のアドレスは適用されません)

出力パラメータ: アドレスが正しい形式の場合は"True"、アドレスが無効な場合は"False"を返します。

説明: 入力されたBitcoinアドレスが正しいかどうかを検証します。このブロックは、Bitcoinシステムで使用されるアドレス作成メカニズムを使用してアドレスが作成されたかどうかを検証するアルゴリズムを適用します。

現在のトランザクションの電子署名検証ブロック。(VerifySignature)。

```
call OpenQbitBlock1 .VerifySignature
```

入力パラメータ：該当しません。

出力パラメータ：チェックが有効な場合は"True"、無効な場合は"False"を返します。

説明: 作成したいトランザクションを送信する過程で、送信者が見たはずの電子署名の検証を取得します。この検証処理では、トランザクションが送信されたチャネルで送信されたソース値が変更されていないことを確認するとともに、トランザクションが適用されるべき受信者を検証する処理を行う。

20.SQLiteデータベースのブロックの使用（MiniSQLite版）

このセクションでは、ミニBlocklyChainシステムの機能性のために興味のある2つの主要な操作を実行するためにブロックを使用する方法を見ていきます。

注：SQLiteデータベース内のトランザクションは、Mini Blocklychainシステムで適用されるノードから送信されるトランザクションとは異なります。

データベース内のトランザクションはCRUD（Create、Read、Update、Delete）モデルに基づいており、SQLiteデータベース内の外部データまたは内部データのみで実行可能な処理です。ブロックチェーンシステムでは、主に作成のプロセスを使用しており、セキュリティのために、更新または削除のプロセスを破棄され、それがシステムの不可欠なセキュリティを与えるブロックのチェーンが格納されている詳細です。

一方、ノードによって送信されたトランザクションは、ミニBlocklyChainシステムのメンバー（ノード）間で資産のいくつかのタイプを送信するアクションを作ることを含むすべてのプロセスを参照してください。

我々は、定義とSQLiteデータベースMiniSQLiteバージョンのブロックの使用を開始しますこのバージョンは、8つのブロックでのみ統合されています。あなたは、SQLiteデータベース内のデータを操作するための完全なバージョンを持っていますが、実用的な目的のためには、MiniSQLiteバージョンのMini BlocklyChainシステムで十分です。

すべてのコンポーネントの使用法と説明を確認したい場合は、附属書「SQLiteデータベースの拡張ブロック」を参照してください。

MiniSQLiteのバージョンブロック。

SQLiteデータベースで何らかのトランザクションを開始するためのブロック (BeginTransaction)

call OpenQbitQSQLite1 .BeginTransaction

必須単位：ブロック(ImportDatabase)、ブロック(OpenDatabase)

入力パラメータ: <必須依存関係>の前に使用してください。

出力パラメータ: 該当しません。SQLite データベースでトランザクションを開始します。

説明: SQLite データベース内のプロセスを開始するブロックは、最初にデータベースのインポート ブロック (`ImportDatabase`) とデータベースのオープン ブロック (`OpenDatabase`) を実行している必要があります。

SQLiteでコミットするためのブロックです。(CommitTrasaction)

`call OpenQbitQSQLite1 .CommitTransaction`

必要な依存関係 : ブロック (`BeginTransaction`) 、ブロック (`ImportDatabase`) 、ブロック (`OpenDatabase`) 。

入力パラメータ: <必須依存関係>の前に使用してください。

出力パラメータ: 該当しません。SQLite データベース内のトランザクションのコミットを実行します。

説明: SQLite データベース上でコミット処理を開始するブロックでは、最初にデータベースのインポート ブロック (`ImportDatabase`) とデータベースのオープン ブロック (`OpenDatabase`) を実行する必要があります。

インポートまたはエクスポートされた SQLite データベースを閉じるためのブロック (`CloseDatabase`)

`call OpenQbitQSQLite1 .CloseDatabase`

必須単位 : ブロック(`ImportDatabase`)、ブロック(`OpenDatabase`)

入力パラメータ: <必須依存関係>の前に使用してください。

出力パラメータ: 該当しません。

説明: SQLite データベースを閉じるブロックでは、最初にデータベースのインポート ブロック (ImportDatabase) とデータベースのオープン ブロック (OpenDatabase) を実行している必要があります。

SQLite データベース (ExportDatabase) をエクスポートするブロック。



必須単位：ブロック(ImportDatabase)、ブロック(OpenDatabase)

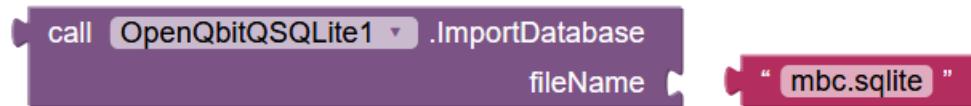
入力パラメータ: fileName <文字列> SQLite形式で既に作成されたデータベースを見つけることができるパスを入力します。

<強制的な依存関係>の前に使用します。

出力パラメータ：SQLite データベースにエクスポートします。

説明: SQLite データベースをエクスポートするブロックでは、最初にデータベースのインポート ブロック (ImportDatabase) とデータベース オープン ブロック (OpenDatabase) を実行している必要があります。

SQLiteデータベースをインポートするためのブロック。(インポートデータベース)

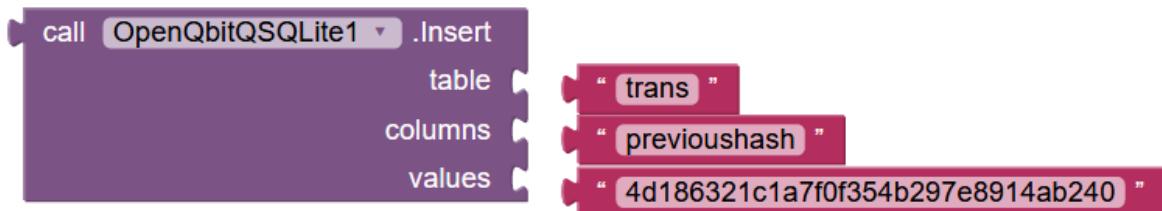


入力パラメータ: fileName <文字列> SQLite形式で既に作成されたデータベースを見つけることができるパスを入力します。

出力パラメータ：トランザクションを実行するためにSQLiteデータベースを起動します。

説明: SQLite データベース内のプロセスを開始するブロックは、最初にデータベースのインポート ブロック (**ImportDatabase**) とデータベースのオープン ブロック (**OpenDatabase**) を実行している必要があります。

SQLiteデータベースにデータを挿入するためのブロック。(挿入)



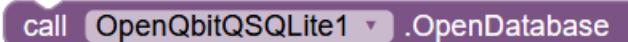
必須単位 : ブロック(**ImportDatabase**)、ブロック(**OpenDatabase**)

入力パラメータ: **table** < String > , **columns** < String > , **values** < String > , **Use before** < Mandatory dependence(s)> の前に使用します。

出力パラメータ: SQLite データベースにトランザクションを挿入します。

説明: SQLite データベースにデータを挿入するブロックは、最初にデータベースのインポート ブロック (**ImportDatabase**) とデータベースのオープン ブロック (**OpenDatabase**) を実行している必要があります。

SQLiteデータベース（**OpenDatabase**）を開くためのブロック



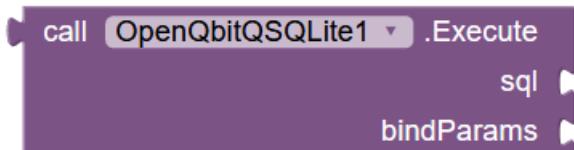
必須単位（複数可） : ブロック (**ImportDatabase**) 。

入力パラメータ: <必須依存関係>の前に使用してください。

出力パラメータ：該当しない、トランザクションを実行するために SQLite データベースを起動またはオープンします。

説明: SQLite データベースを起動するブロックは、前にある必要があります。

SQLiteでのデータクエリー（実行）のためのブロック。



必須単位：ブロック(ImportDatabase)、ブロック(OpenDatabase)

入力パラメータ: sql <文字列>, bindParams <文字列>, <必須依存関係>の前に使用する

出力パラメータ： SQLite データベースにトランザクションを作成するために SQL 文が実行されます。

説明：ブロックは、SQLiteデータベース内のSQL文を実行すると、最初にデータベース（ImportDatabase）とデータベース（OpenDatabase）を開くのブロックをインポートのブロックを実行している必要があります。

21. セキュリティブロックの定義と使い方

このセッションでは、Mini BlocklyChain ネットワークノードからのトランザクションを保存、検証、転送するためのセキュリティレベルを提供してくれるブロックの使用方法をレビューします。

セキュリティブロックは、以下の拡張子に基づいています。

- I. OpenQbitAESEncryption 拡張機能。
- II. OpenQbitAESDecryption 拡張機能。
- III. OpenQbitAEToString 拡張モジュール。
- IV. OpenQbitEncDecData 拡張機能です。
- V. OpenQbitFileHash 拡張モジュール。
- VI. OpenQbitRSA 拡張。
- VII. OpenQbitSSHClient 拡張機能 (必要)
- VIII. OpenQbitStringHash 拡張モジュール。

必須の OpenQbitSSHClient 拡張を除いて、上記の拡張は、パブリックまたはプライベートの Mini BlocklyChain ネットワークの作成に使用するためのオプションです。

ただし、それぞれのビジネスケースによっては、取引を安全に行うための体制を整えるためには、任意の拡張機能の利用を任意で適用する必要があります。

例えば、ハッシュを使用する場合には、次のようなアルゴリズムのオプションを使用することができます。文字列のMD5、SHA1、SHA128、SHA256、SHA512は、Mini BlocklyChainで作成された各システムの情報の流れに応じて、任意の種類のファイルに適用されるだけでなく、文字列のMD5、SHA1、SHA128、SHA256、SHA512にも適用されます。

OpenQbitAESEncryption 拡張機能。

AESセキュリティでファイルを暗号化するためにブロックします。(AESEncryption)。



入力パラメータ: pathFileIn <文字列> , pathFileOut <文字列> , pathFile <文字列> , pathFileVI <文字列> , passwd <文字列>。ファイル名は任意であり、各システム設計の裁量に委ねられています。

出力パラメータ：入力パラメータ pathFileIn で入力された AES 暗号化ファイル。

説明: 私たちに3つの出力ファイルを与えるブロックこれらの1つは、すでに256ビットのキーを持つAESアルゴリズムによって暗号化されたソースファイルですが、他の2つのファイルを復号化し、元のファイルを回復するために拡張子を制御するために使用されます。

OpenQbitAESDecryption 拡張機能。

AES ファイルを復号化するためのブロック。(AESDecryption)。



入力パラメータ: pathFileIn <文字列>, pathFileOut <文字列>, pathFile <文字列>, pathFileVI <文字列>, passwd <文字列>。ファイルの名前は、ブロック (AESEncryption) の結果として得られたものと同じです。

出力パラメータ：入力パラメータpathFileInにAESで復号化された元のファイルが入力され、この場合はファイルを復号化するためにpathFileInに暗号化されたファイルが入力され、pathFileOutには元のファイル（復号化されたファイル）が入力されます。

説明：pathFileOut パラメータ内のファイルを与えてくれるブロックで、256 ビットキーで AES アルゴリズムによって復号化された出力になります。

OpenQbitAEToString 拡張モジュール。

この拡張機能は、デバイスセッションごとに単一で使用するもので、すなわち、VI暗号化値が一時的に生成されるため、デバイスが再起動されていない場合（携帯電話）にのみ機能します。

注：デバイスが再起動されると、暗号化された文字列を復元することはできません。

一時的な文字列暗号化用ブロック(DecrypSecretText)

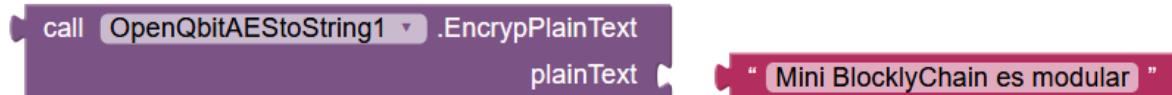


入力パラメータ：secretText <文字列>

出力パラメータ：256ビットキーのAESで復号化された元の文字列

説明：文字列を与えてくれるブロックは、そのブロック（EncryptPlainText）で紹介されていた入力パラメータです。

文字列をデコードするブロック (EncryptPlainText)



入力パラメータ: plainText <文字列>

出力パラメータ：256ビットのキーを使用してAESで暗号化された文字列。

説明: 256 ビットのデジタルキーを使用して AES で暗号化された英数字文字列を与えるブロック。

OpenQbitEncDecData拡張機能です。

汎用データベース用に特化した暗号化ブロック(暗号化データ)



入力パラメータ: plainText <文字列>

出力パラメータ：256ビットのキーを使用してAESで暗号化された文字列。

説明: 256 ビットのデジタルキーを使用して AES で暗号化された英数字文字列を与えるブロック。

汎用データベース用に特化した暗号化ブロック(DescriptData)



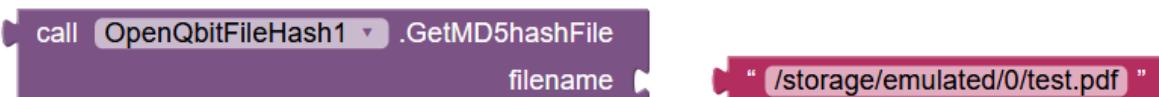
入力パラメータ: plainText <文字列>

出力パラメータ: 256ビットのキーを使用してAESで暗号化された文字列。

説明: 256 ビットのデジタルキーを使用して AES で暗号化された英数字文字列を与えるブロック。

OpenQbitFileHash 拡張モジュール。

ファイルからMD5を生成するブロック (GetMD5hashFile)

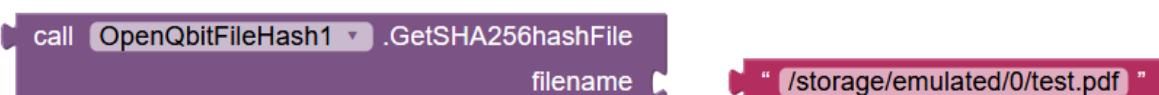


入力パラメータ: ファイル名 <文字列>

出力パラメータ: MD5 ハッシュファイルを出力します。

説明: 入力パラメータで与えられたファイルの MD5 ハッシュを作成するためのブロック。

ファイルから SHA256 を生成するブロック (GetSHA256hashFile)

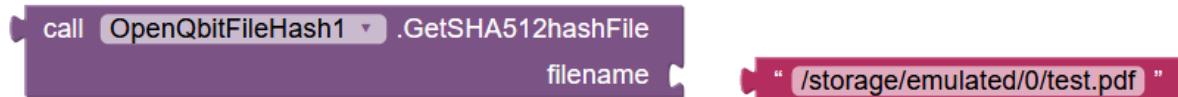


入力パラメータ: ファイル名 <文字列>

出力パラメータ: SHA256 アーカイブハッシュを提供します。

説明: 入力パラメータで与えられたファイルの SHA256 ハッシュを作成するブロック。

ファイルから SHA512 を生成するブロック (GetSHA512hashFile)

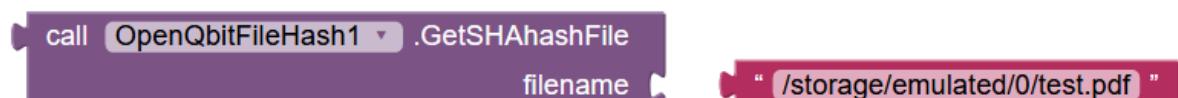


入力パラメータ : ファイル名 <文字列>

出力パラメータ: SHA256 アーカイブハッシュを提供します。

説明 : 入力パラメータで与えられたファイルの SHA256 ハッシュを作成するブロック。

ファイルから SHA1 を生成するブロック (GetSHA1hashFile)



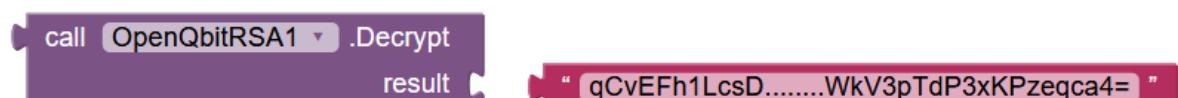
入力パラメータ : ファイル名 <文字列>

出力パラメータ : SHA1 アーカイブハッシュを提供します。

説明 : 入力パラメーターにダイの SHA1 ハッシュを作成するブロック。

OpenQbitRSA拡張。

RSAで文字列を復号化するためのブロック(Decrypt)



必要な依存関係: Block (Encrypt)、Block (OpenFromDiskPrivateKey)、Block (OpenFromDiskPublicKey)。

入力パラメータ: result <文字列>

出力パラメータ : RSAでデコードされた文字列。

説明: ブロックで使用されたサイズのキー(GenKeyPair)を使用して解読された英数字の文字列を与えてくれるブロック。

文字列をRSA(Encrypt)で暗号化するためのブロック



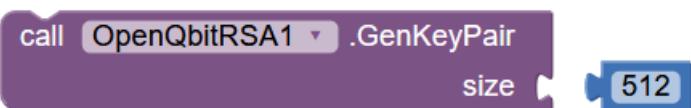
必要単位：ブロック(GenKeyPair)、ブロック(SaveFromDiskPrivateKey)、ブロック(SaveFromDiskPublicKey)

入力パラメータ：プレーン <文字列>

出力パラメータ：RSA暗号化された文字列

説明: ブロックで使用されたサイズのキー(GenKeyPair)を使用して解読された英数字の文字列を与えてくれるブロック。

RSAで文字列を復号化するためのブロック(Decrypt)

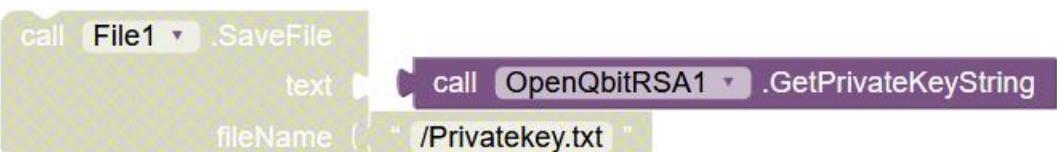


入力パラメータ: size <整数>

出力パラメータ：該当しません。

説明: 選択したサイズに基づいて秘密鍵と公開鍵を生成するブロック。

秘密鍵を取得するためのブロック (GetPrivateKeyString)



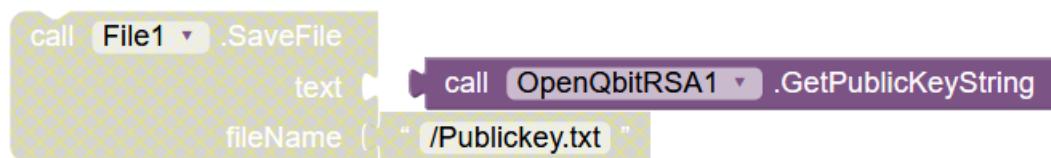
必須単位：ブロック（GenKeyValuePair）, ブロック（GenKeyValuePair）, ブロック（ファイル

入力パラメータ：該当しません。

出力パラメータ：RSA暗号化された文字列（秘密鍵）を持つファイル

説明：ブロック内で使用されたサイズ鍵(GenKeyValuePair)を使用して暗号化された秘密鍵を表す英数字の文字列を与えてくれるブロック。

秘密鍵を取得するブロック（GetPublicKeyString）



必須単位：ブロック（GenKeyValuePair）, ブロック（GenKeyValuePair）, ブロック（ファイル

入力パラメータ：該当しません。

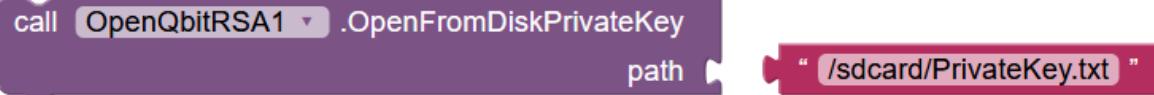
出力パラメータ：RSA暗号化された文字列（公開鍵）を持つファイル

説明：ブロック内で使用されている鍵サイズ（GenKeyValuePair）を使用して暗号化された公開鍵を表す英数字の文字列を与えてくれるブロック。

注：依存関係の前のブロック(GetPrivateKeyString)と(GetPublicKeyString)では、「ストレージ」パレットセッションのApp Inventor アプリケーションの汎用ブロック(File)を使用します。

このように、ブロック（ファイル）を用いて秘密鍵と公開鍵を格納する方法は、情報の操作性を向上させるのに役立ちます。

ファイルから秘密鍵を読み出すブロック (OpenFromDiskPrivateKey)。



必要な依存関係 : ブロック (SaveFromDiskPrivateKey) またはブロック (GetPrivateKeyString)。

入力パラメータ : path <文字列>

出力パラメータ : システムロード RSA秘密鍵暗号化文字列。

説明: 指定したファイルパスに格納されている秘密鍵の暗号化された英数字文字列を与えてくれるブロック。

ファイルから公開鍵を読み込むブロック (OpenFromDiskPublicKey)



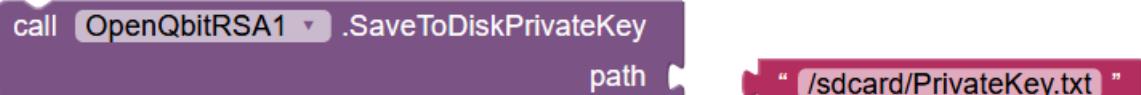
必須単位: ブロック (SaveFromDiskPublicKey) またはブロック (GetPublicKeyString)。

入力パラメータ : path <文字列>

出力パラメータ : システム RSA公開鍵暗号化文字列をロードします。

説明: 指定したファイルパスに格納されている公開鍵の暗号化された英数字文字列を与えてくれるブロック。

秘密鍵をファイルに保存するためのブロック (SaveToDiskPrivateKey)。



必須単位(s) : ブロック(GenKeyPair)。

入力パラメータ：プレーン <文字列>

出力パラメータ：RSA暗号化された文字列を含むファイル。秘密鍵

説明： ブロック内で使用されているサイズの秘密鍵(GenKeyPair)を使用して暗号化された英数字の文字列を持つファイルを提供してくれるブロック。

秘密鍵をファイルに保存するためのブロック (SaveToDiskPublicKey) 。



必須単位(s)：ブロック(GenKeyPair)。

入力パラメータ：プレーン <文字列>

出力パラメータ：RSA暗号化された文字列を含むファイル。公開鍵

説明： ブロックで使用されたサイズの公開鍵(GenKeyPair)を使用して暗号化された英数字の文字列を持つファイルを提供してくれるブロック。

OpenQbitSSHClient 拡張機能。

コネクタブロック SSH クライアント (ConnectorMiniBlocklyChain)

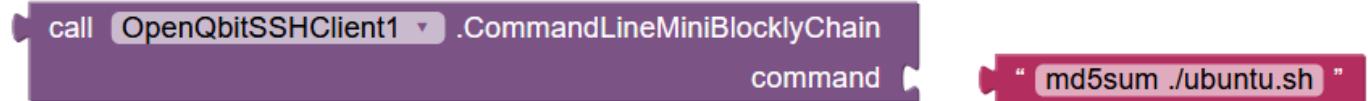


入力パラメータ: username <string>, password <string>, host <string>, port<integer>.

出力パラメータ: Termux 端末の ssh サーバとの接続に成功した場合は "Connect SSH" というメッセージを、失敗した場合は NULL というメッセージを出力します。

説明: Mini BlocklyChain と Termux 端末を SSH (Secure Shell) 通信プロトコルで接続するための通信ブロックです。

Termux Linuxターミナルでコマンドを実行するためのブロック (CommandLineMiniBlocklyChain)



。

入力パラメータ：コマンド <文字列>

出力パラメータ：実行されたコマンドやプログラムに応じて変数データを出力します。

説明：Termuxターミナルのコマンド実行ブロックは、ブロック (ConnectorMiniBlocklyChain) との接続を行うための前提条件は、すべての種類のコマンドをオンラインで実行することができますおよび/またはCLI (コマンドラインインターフェース) オンラインを持っているスクリプトやプログラムから特定の実行データを取得します。

DisconnectMiniBlocklyChainSSH.

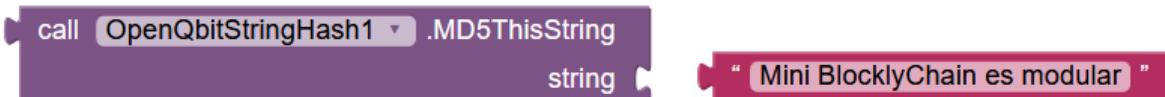


入出力パラメータ：該当なし（なし）

説明: SSH セッションを閉じるためのブロック。

OpenQbitStringHash 拡張モジュール。

MD5 文字列を生成するブロック (MD5ThisString)

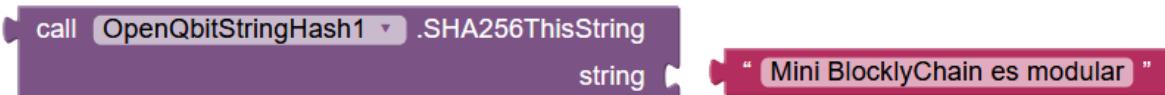


入力パラメータ：文字列

出力パラメータ: MD5 ハッシュを提供します。

説明: 入力パラメータで与えられた文字列の MD5 ハッシュを作成するためのブロック。

SHA256文字列（SHA256ThisString）を生成するブロック

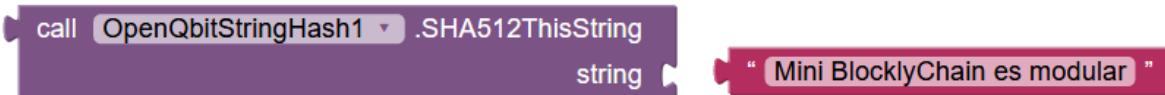


入力パラメータ：文字列

出力パラメータ: SHA256 ハッシュを提供します。

説明：入力パラメータで与えられた文字列の SHA256 ハッシュを作成するブロック。

SHA512文字列（SHA512ThisString）を生成するブロック

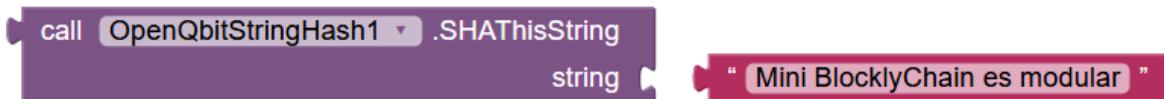


入力パラメータ：文字列

出力パラメータ: SHA512 ハッシュを提供します。

説明： 入力パラメータで与えられた文字列の SHA512 ハッシュを作成するブロック。

SHA1文字列（SHAThisString）を生成するブロック



入力パラメータ：文字列

出力パラメータ：SHA1 ハッシュを提供します。

説明：入力パラメタで与えられた文字列の SHA1 ハッシュを作成するブロック。

22. Mini BlocklyChainでセキュリティパラメータを設定します。

安全パラメータは、以下の構成要素で設計され、適用される任意のシステムの3つのコンポーネントに分かれています。

- a. Redis データベース（バックアップ通信網）
- b. ピアツーピア同期システム
- c. SQLiteデータベースを保護するためにセキュリティを統合する
- d. モジュール統合のための開発環境

a. Redis データベース（バックアップ通信網

危険なコマンドの名前を変更することで、Redisの追加の組み込み安全機能では、危険と考えられるコマンドの名前を変更したり、無効にしたりすることができます。

許可されていないユーザーによって実行されると、これらのコマンドを使用してデータの再構成、破壊、または削除を行うことができます。認証パスワードと同様に、コマンドの名前の変更や無効化は、`/etc/redis/redis.conf` ファイルの同じ SECURITY セクションで設定します。

危険と考えられるコマンドの一部: FLUSHDB、FLUSHALL、KEYS、PEXPIRE、DEL、CONFIG、SHUTDOWN、BGREWRITEAOF、BGSAVE、SAVE、SPOP、SRLEM、RENAME、DEBUG。これは完全なリストではありませんが、このリストにあるすべてのコマンドの名前を変更したり、無効にしたりすることは、Redisサーバのセキュリティを向上させるための良いスタートとなります。

特定のニーズやサイトのニーズに応じて、コマンドの名前を変更したり、コマンドを無効にしたりする必要があります。操作可能なコマンドは絶対に使わないとわかっているのであれば、それを無効にすることができます。逆に名前を変えた方がいいかもしれません。

Redisコマンドを有効または無効にするには、設定ファイルを再度開きます。

```
vi /etc/redis/redis.conf
```

警告：以下の手順では、コマンドを無効にしたり、名前を変更したりする方法を例示しています。自分に適用されるコマンドを無効にするか、名前を変更するかは、自分で選択するしかありません。コマンドの完全なリストを確認して、redis.io/commands でどのように悪用されるかを判断することができます。

コマンドを無効にするには、以下に示すように、コマンドの名前を変更して空の文字列 (引用符で囲まれた文字列) にするだけです。

```
/etc/redis/redis.conf
```

```
.
.
.
# コマンドを完全終了させることも可能です。
# 空の文字列です
#
```

```

リネームコマンド FLUSHDB ""
リネームコマンド FLUSHALL ""
rename-command DEBUG ""
...

```

コマンドの名前を変更するには、以下の例のように別の名前を付けます。名前を変更したコマンドは、他の人が推測するのは難しいですが、あなたが覚えやすいようにする必要があります。

/etc/redis/redis.conf

```

...
# rename-command CONFIG ""
名前変更コマンド SHUTDOWN SHUTDOWN_MENOT
名前変更コマンド CONFIG ASC12_CONFIG
...

```

変更を保存してファイルを閉じます。

コマンド名を変更した後、Redisを再起動して変更を適用します。

- sudo systemctl restart redis.service

新しいコマンドをテストするには、Redisのコマンドラインに入ります。

- レディスクリ

その後、認証を行います。

- auth your_redis_password

出力
宜い

先ほどの例のように、CONFIGコマンドの名前をASC12_CONFIGに変更したとします。 まず、元の CONFIGコマンドを使ってみてください。名前を変えたから、うまくいかないはずです。

- コングリゲーション

出力

(エラー) ERR 不明なコマンド 'config'

ただし、リネームされたコマンドは正常に呼び出すことができます。大文字小文字を区別しません

。

- `asc12_config get requirepass`

出力

1) "requirepass"
2) "your_redis_password"

いよいよ、再編成を締めくくることになります。

- 出る

すでにRedisのコマンドラインを使用していてRedisを再起動した場合は、再度認証が必要になることに注意してください。そうでない場合は、コマンドを入力するとこのエラーが表示されます。

出力

NOAUTH 認証が必要です。

b. ピアツーピア同期システム

ノード間の"ピアツーピア"システムの使用では、システムは、通信ネットワークに適用される次の3つの要素を持っています。

-プライベート：パソコン以外に情報が保存されていることはありません。危険化するような中央サーバは存在しない（合法的にも違法にも）。

-暗号化：すべての通信はTLS(Transport Layer Security)プロトコルで保護されています。

-認証済み：各ノードは強力な暗号化証明書で識別されます。あなたが明示的に許可したノードのみが、あなたの情報に接続することができます。

<https://docs.syncthing.net/users/security.html>

オンラインコマンドSyncthingManagerを使用します。

<https://github.com/classicsc/syncthingmanager>

c. SQLiteデータベースを保護するためにセキュリティを統合する

拡張機能（OpenQbitSQLite）またはその場合は拡張機能（ConnectorSSHClient）を使用してSQLite CLIを使用する場合、両方の拡張機能はセキュリティ拡張機能AES（OpenQbitEncDecData）と組み合わせることができます。

別紙「ミニBlocklyChainシステムの作成例」を参照してください。

d. モジュール統合のための開発環境

開発環境でのセキュリティを実装するには、2つのプロセスで行うことができます。

- OpenJDKライブラリの使用を制限する。
- 使用は許可されたシステムノードに制限されています。

23.別館「KeyStoreとPublicKeysデータベースの作成」。

KeyStoreは、認証証明書、公開鍵証明書、パスワード、またはユーザーの対応する秘密鍵（アドレス）などの一般的なセキュリティ鍵のいずれかであるセキュリティ証明書のリポジトリであり、トランザクションの作成や処理に使用される。

暗号化されたデータベースなので、所有者だけが利用できるようになっています。通常はローカルタイプですが、Mini BlocklyChain システムではセキュアなデータベースですが、全ノードで共有・分散されています。これは基本的に単純な理由によるもので、全ノードが全ユーザーのアドレス（パブリックアドレス）を知らなければならないからです。

KeyStoreを作成するには、以下の手順と要件に従う必要があります。

最初の要件は、保存されるストレージの種類を持っていることです。この場合、SQLiteデータベースを使用し、2つのKeyStoreを作成します。1つはユーザーの秘密鍵で、ローカルになり、情報を「暗号化」して保護され、もう1つは公開鍵を保存するデータベースで、このベースは、ネットワークのすべてのノードで共有されます。

第二の基本的な要件は、情報の暗号化のプロセスであり、これは、AESアルゴリズムを使用している(OpenQbitEncDecData)と呼ばれる汎用データベースで使用するための特殊な拡張機能を使用して行われます。

拡張機能(OpenQbitEncDecData)はブロックチェーンの単位要素の検証のために機能しますが、ブロックチェーンの完全性を確認する必要がある場合には効率的ではないので、コピーの暗号化も行うことになりますが、この場合には拡張機能を利用します。(OpenQbitAESEncryption) および(OpenQbitAESDecryption) は、参照データではなくファイル上で動作します。

注：KeyStoreデータベースが正常に機能するためには、Termux端末上でSSHサーバが動作している必要があります。ターミナルでコマンドを実行します。

sshd

AES アルゴリズムに基づく拡張機能は、あらゆるタイプのデータやファイルに使用することができ、このアルゴリズムは、量子コンピューティングに基づく攻撃に対する証明が証明されている唯一のものであることから選択されました。

Criptosistema	Categoría	Tamaño de clave	Parámetro de seguridad	Algoritmo cuántico estimado que rompa el criptosistema	Nº de qubits lógicos necesarios	Nº de qubits físicos necesarios	Tiempo necesario para romper el sistema	Estrategias de reemplazo cuántico-resilientes
AES-GCM	Cifrado simétrico	128	128	Algoritmo de Grover	2.953	$4,61 \times 10^6$	$2,61 \times 10^{12}$ años	
		192	192		4.449	$1,68 \times 10^7$		
		256	256		6.681	$3,36 \times 10^7$		
RSA	Cifrado asimétrico	1024	80	Algoritmo de Shor	2.290	$2,56 \times 10^6$	3,58 horas	Migrar a un algoritmo PQC seleccionado por el NIST
		2048	112		4.338	$6,2 \times 10^6$	28,63 horas	
		4096	128		8.434	$1,47 \times 10^7$	229 horas	
ECC Problema del logaritmo discreto	Cifrado asimétrico	256	128	Algoritmo de Shor	2.330	$3,21 \times 10^6$	10,5 horas	Migrar a un algoritmo PQC seleccionado por el NIST
		386	192		3.484	$5,01 \times 10^6$	37,57 horas	
		512	256		4.719	$7,81 \times 10^6$	95 horas	
SHA256	Minado de Bitcoin	N/A	72	Algoritmo de Grover	2.403	$2,23 \times 10^6$	$1,8 \times 10^4$ años	
PBKDF2 con 10.000 iteraciones	Hashing de contraseñas	N/A	66	Algoritmo de Grover	2.403	$2,23 \times 10^6$	$2,3 \times 10^7$ años	Abandonar la autenticación basada en contraseñas

上記は全科学アカデミー (National Academies of Science, Engineering and Medicine) を参考しています。

<https://www.nap.edu/catalog/25196/quantum-computing-progress-and-prospects>

説明

量子力学は、非常に小さな粒子（量子）の振る舞いを記述する物理学のサブフィールドであり、新しいコンピューティング・パラダイムの基礎を提供します。1980年代に量子システムの計算モデルを改善する方法として提唱された量子コンピューティングの分野は、近年、小規模デバイスの構築が進んだことで注目を集めています。しかし、実用的な量子コンピュータを大規模に実現するには、技術的に大きな進歩が必要です。

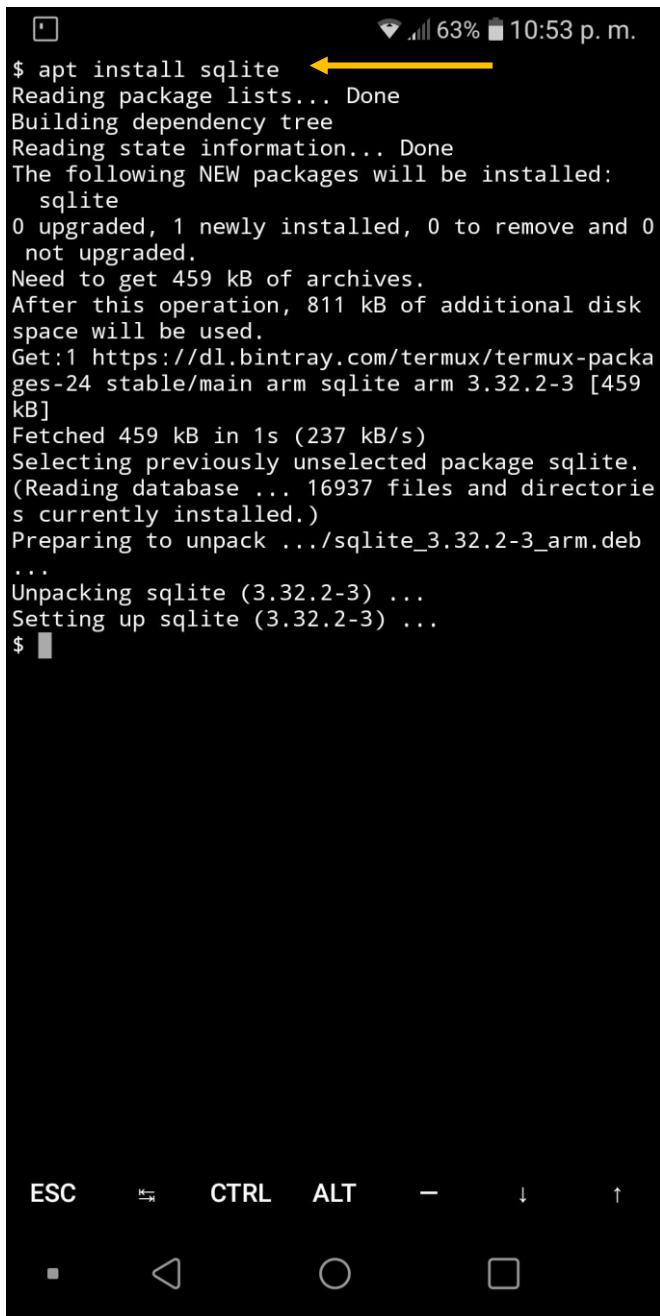
『量子コンピューティング：進歩と展望』は、この技術のユニークな特徴と限界を含む分野の紹介を行い、現実世界の問題に対処できる機能的な量子コンピュータを作成するための実現可能性と意味合いを評価しています。このレポートでは、ハードウェアとソフトウェアの要件、量子アルゴリズム、量子コンピューティングと量子デバイスの進歩の原動力、関連するユースケースに関するベンチマーク、必要な時間とリソース、成功の確率を評価する方法について考察しています。

TERMUX端末にSQLiteデータベースハンドラをインストールし、keystore.dbというデータベースを作成してsqlite3コマンドのCLI（コマンドライン）をテストします。

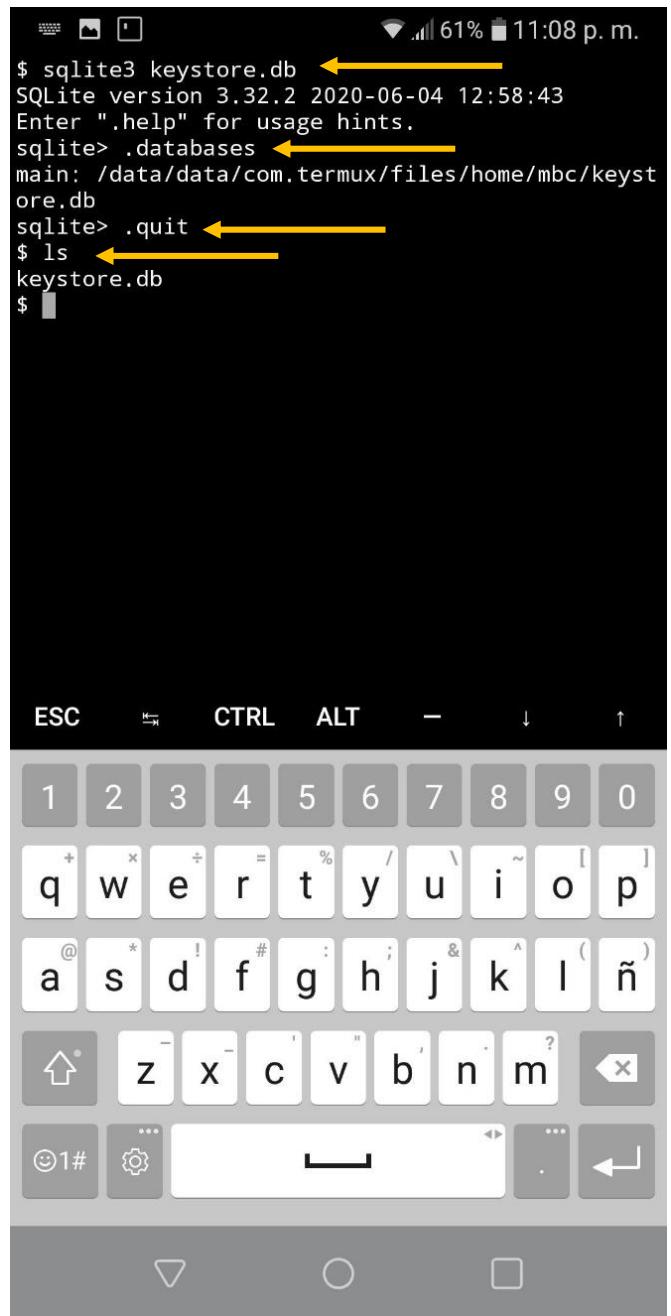
コマンドを使っています。

```
apt install sqlite
```

```
$ sqlite3 keystore.db
```



```
$ apt install sqlite
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
sqlite
0 upgraded, 1 newly installed, 0 to remove and 0
not upgraded.
Need to get 459 kB of archives.
After this operation, 811 kB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-packages-24 stable/main arm sqlite arm 3.32.2-3 [459
kB]
Fetched 459 kB in 1s (237 kB/s)
Selecting previously unselected package sqlite.
(Reading database ... 16937 files and directorie
s currently installed.)
Preparing to unpack .../sqlite_3.32.2-3_arm.deb
...
Unpacking sqlite (3.32.2-3) ...
Setting up sqlite (3.32.2-3) ...
$
```



```
$ sqlite3 keystore.db
SQLite version 3.32.2 2020-06-04 12:58:43
Enter ".help" for usage hints.
sqlite> .databases
main: /data/data/com.termux/files/home/mbc/keyst
ore.db
sqlite> .quit
$ ls
keystore.db
$
```

そして、sqlite>ハンドラ内で.databases文を実行して、作成しているデータベースのパスを確認し、データベースを終了して保存するために.quit文を与えます。

注意: sqlite>ハンドラ内の文やコマンドでは、まず構文の中にドット"."を入れなければなりません

。

最後に、(空の)データベースが既に作成されているかどうかを確認します。

ls

16進数、バイナリ、Mini BlocklyChain のユーザー参照アドレスの3つのフォーマットで主キーが保存されるテーブルの作成を進めます。

AES データの暗号化は、16 進数形式とバイナリ形式でのみ適用されます。ユーザのアドレス addrMBCとエイリアスの場合ではなく、それができるとトランザクションを受信するために、ネットワーク全体で共有する必要があります公開鍵だけでなく、このフィールドで検索を実行するためのエイリアスであるためです。

SQLiteのデータベース (keystore.db) に「privatekey」というテーブルを作成しました。

```
CREATE TABLE privatekey (
```

アイド整数主キー

別名 VARCHAR(50) NOT NULL

addrHexVARCHAR (65) NOT NULL

addrMBCVARCHAR (65) NOT NULL

addrBinBLOB NOT NULL

);

sqlite CLIで文章を実行してkeystore.dbデータベースを作成してみましょう。

以下のコマンドで再度sqlite3のコマンドラインを使ってみましょう。

\$ sqlite3

これはsqlite>データベースマネージャの中に送られます。この最初のものでは、すでに作成されたデータベースを開いて、マネージャの中で次の文章で作業できるようにします。

Sqlite> .open keystore.db

続いて、SQL文「CREATE TABLE」を入力して、privatekeyテーブルを作成します。

次に、同じテーブルを作成するための2つのオプションが示されていますが、1つは、それを統合する要素のすべてのSQL文が含まれている単一の行を介しています。第二の例は、セグメント化された方法で構造を統合する各要素の導入を通じたものである。

```
$ sqlite3
```

```
SQLiteバージョン3.32.2 2020-06-20 15:25:24
```

使用方法のヒントは「.help」を入力してください。

トランジエントなインメモリデータベースに接続されています。

.open FILENAME"を使用して、永続的なデータベースで再開します。

```
sqlite> .open keystore.db
```

```
sqlite> CREATE TABLE privatekey (id integer primary key AUTOINCREMENT NOT NULL, aka VARCHAR(50) NOT NULL, addrHex VARCHAR(65) NOT NULL, addrMBC VARCHAR(65) NOT NULL, addrBin BLOB NOT NULL);
```

```
sqlite> .quit
```

次に同じprivatekeyテーブルの作成を示しますが、SQL文をセグメント化して導入することで作成しています。

```
sqlite> CREATE TABLE privatekey ( ...> id 整数 AUTOINCREMENT ...> 別名 VARCHAR(50) NOT NULL。 ...> addrHex VARCHAR (65) NOT NULL。 ...> addrMBC VARCHAR (65) NOT NULL。 ...> addrBin BLOB NOT NULL ...> );
```

```
sqlite> .tables
```

秘匿キー

```
sqlite> .quit
```

上の2つの例では、同じ結果が得られます。後で我々は、privatekeyテーブルが作成されていることを確認するために.tables文を実行し、最後に我々はすでにSQLite keystore.dbデータベース内のprivatekeyテーブルを作成しているこのプロセスで.quit文を与えるでしょう。

この時点までに、keystore.dbデータベースの構造と、秘密鍵が暗号化されて保存されるprivatekeyテーブルはすでに存在しています。

これはTERMUX端末のあるノード（携帯電話）にも似たようなものが表示されているはずです。



The screenshot shows a Termux terminal window on an Android device. The terminal output is as follows:

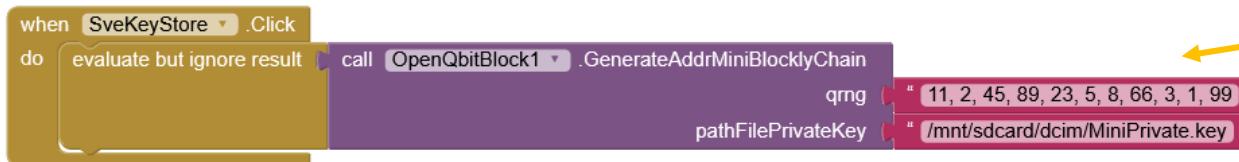
```
$ sqlite3
SQLite version 3.32.2 2020-06-04 12:58:43
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open keystore.db
sqlite> CREATE TABLE privatekey (
...>     id integer primary key AUTOINCREMENT,
...>     alias VARCHAR(50) NOT NULL,
...>     addrHex VARCHAR(65) NOT NULL,
...>     addrMBC VARCHAR(65) NOT NULL,
...>     addrBin BLOB NOT NULL
...> );
sqlite> .tables
privatekey
sqlite> .quit
$
```

A yellow bracket on the right side of the terminal output indicates the command `.open keystore.db`, which is highlighted with a yellow arrow. Another yellow bracket covers the command `.quit`, also highlighted with a yellow arrow. A callout box with a yellow border contains the text "新しい'privatekey'テーブルを作成する".

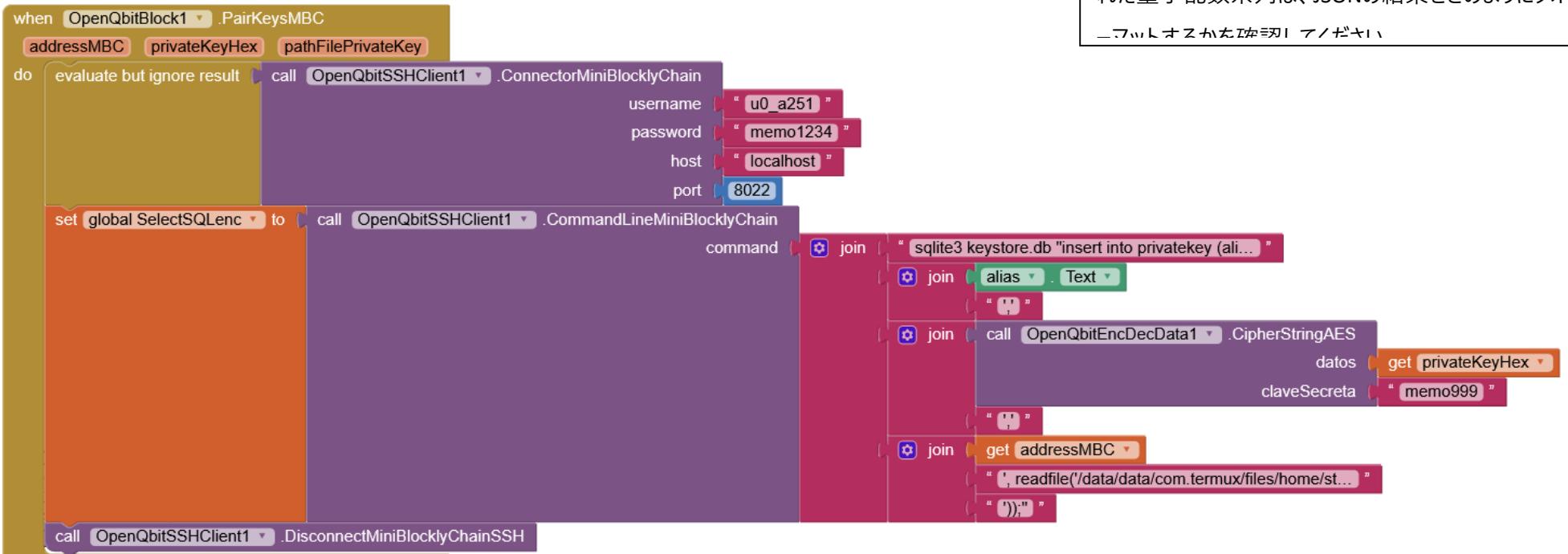
これからいくつかのセキュリティ拡張機能を使って「秘密鍵」のデータを挿入し、そのデータを参照していきます。

我々は、新しいユーザー・アドレス（GenerateAddrMiniBlocklyChain）を生成するためにブロックを使用して、このブロックは私たちに2つの形式（16進数とバイナリ）のプライベート・アドレスだけではなく、MBCの一般的なアドレス形式のパブリック・アドレスを与えます。我々はまた、これらの詳細を確認するために、拡張機能(OpenQbitSSHClient)と拡張機能(OpenQbitEncDecData)を使用しますセクション"セキュリティ・ブロックの定義と使用"を確認してください。Termuxターミナルでは、SSHサービスを実行しているはずです。

暗号化されたデータをkeystore.dbデータベースにINSERTします。



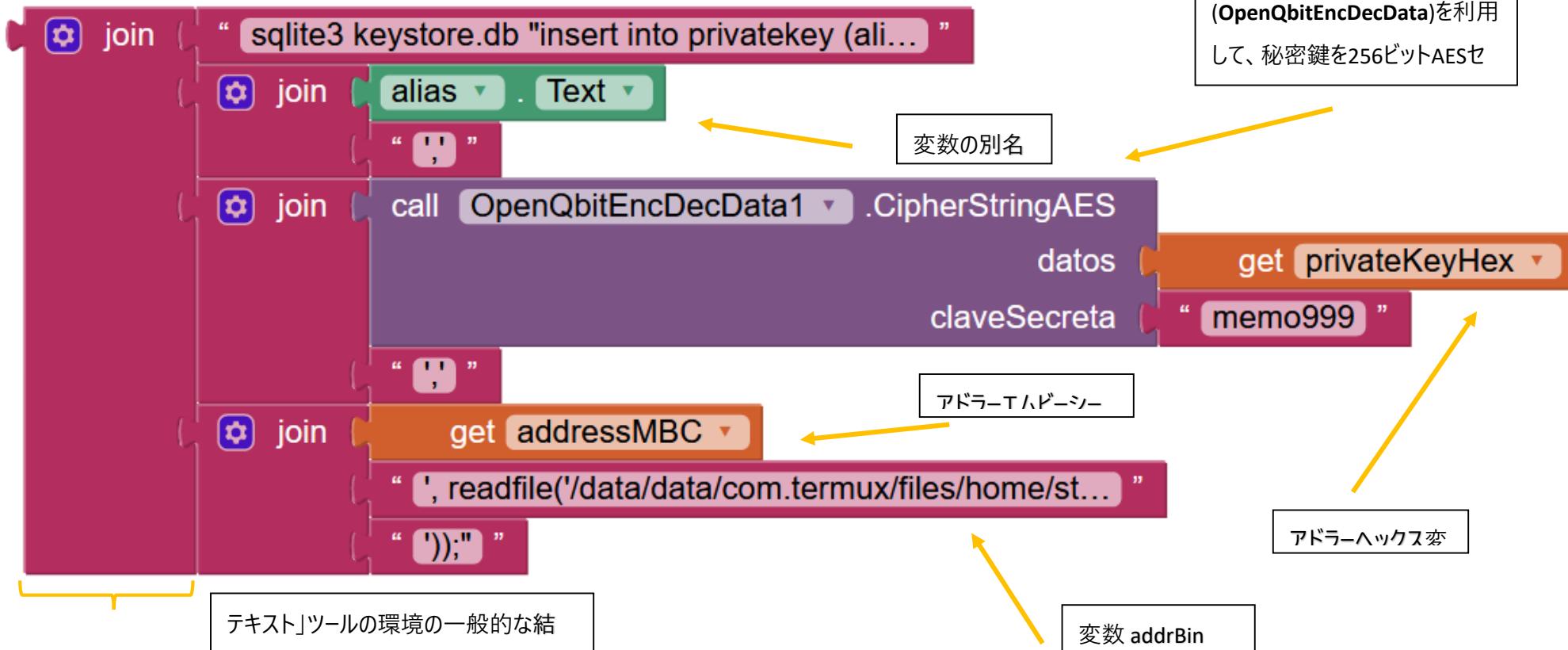
ブロックのエントリ(GenerateAddrMiniBlocklyChain)
はカンマ「,」で区切られた数字の系列のみを必要とするので、ブロック(ApiGetQRNGInteger)によって生成された量子乱数系列は、JSONの結果をどのようにフォーマットオプションで取り扱うかが示されています。



コマンドの構文は非常に重要なので、以下のコマンドをBlockly環境で正しい形式で導入する必要があります。

値の値は常に変数になります、例。

```
sqlite3 keystore.db "insert into privatekey (alias, addrHex, addrMBC, addrBin) values ('memo', 'QWERTY', 'MBC12345',  
readfile('/data/data/com.termux/files/home/storage/dcim/MiniPrivate.key'));"
```



ブロックを実行すると、keystore.dbデータベースのprivatekeyテーブルに以下のような結果が得られます。Termuxターミナルでsqliteハンドラを入力し、keystore.dbベースを開き、文を実行します。

\$ sqlite3

SQLiteバージョン3.32.2 2020-06-20 15:25:24

使用方法のヒントは「.help」を入力してください。

トランジエントなインメモリデータベースに接続されています。

.open FILENAME"を使用して、永続的なデータベースで再開します。

sqlite> .open keystore.db

sqlite> select * from privatekey.

```

$ sqlite3
SQLite version 3.32.2 2020-06-04 12:58:43
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open keystore.db
sqlite> select * from privatekey;
1|mexico|07JBBizTcC0Ce6a8Pwe5aTV41Ql1DKiUQZyPSfJuRbVUZnvIwQJcry4LilBMs5jHavQeMo1iN8rCO87V5Xka2YaKR4fswopeTQmI/Q+ipzI=|2eENpFt2HSuGtXNxoeiwfrp5d6e87Z7y5|0♦♦
sqlite>

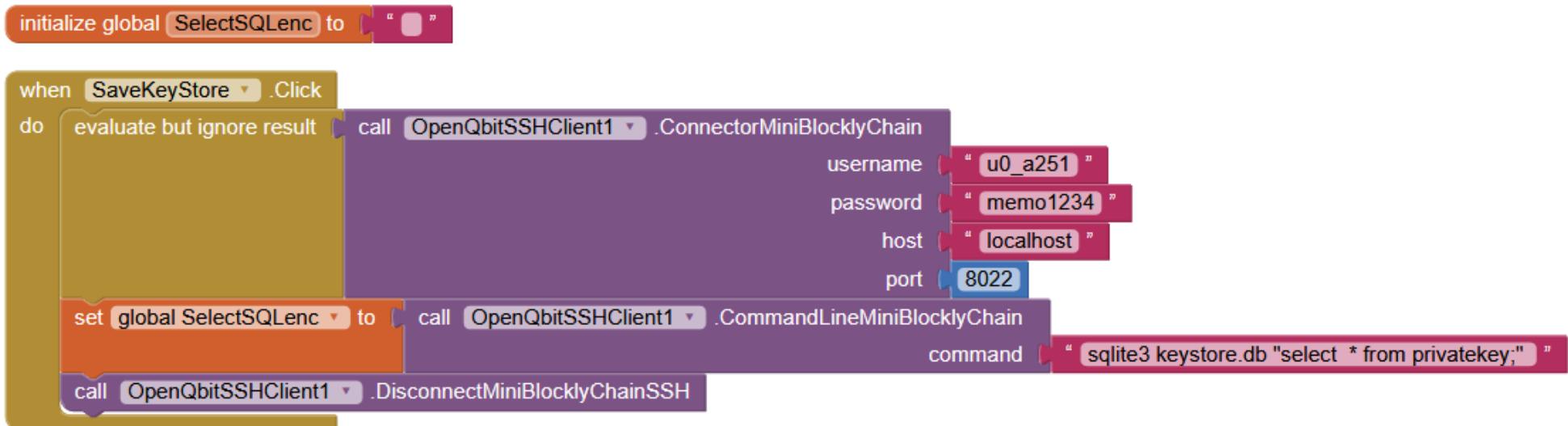
```

別名 : メキシコ

addrBin (バイナリの秘密鍵)

addrHex 暗号化

SQLite **keystore.db**データベースの**privatekey**テーブル内のすべてのデータをCONSULTします。



SQLite **keystore.db**データベースの**privatekey**テーブルのデータのエイリアスをCONSULTします。

`sqlite3 keystore.db "select alias from privatekey;"`

SQLite **keystore.db** データベースの **privatekey** テーブルで暗号化された **addrHex** カラムのすべてのデータをクエリします。

`sqlite3 keystore.db "select addrHex from privatekey;"`

SQLite **keystore.db**データベースの**privatekey**テーブルの**addrBin**秘密鍵を取得するためのCONSULT

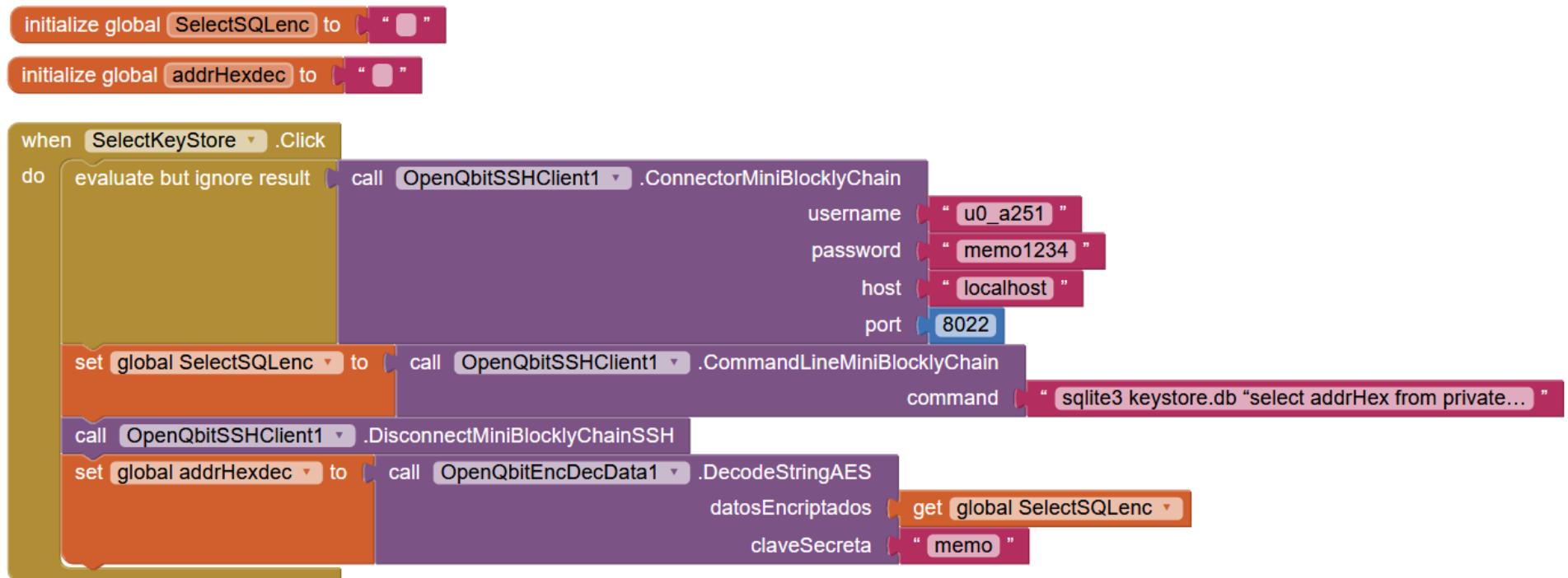
`sqlite3 writedb('PrivateKey.key', addrBin) FROM privatekey WHERE alias='mexico'; (2)`

(1) このタイプのクエリは、トランザクションの署名処理を実行するための秘密鍵を参照するための主なものになります。

SQLite keystore.dbデータベースのprivatekeyテーブルのaddrHexカラムのエイリアスによるデータ復号化クエリ

今回はブロック(DecodeStringAES)を使用して、"addrHex"カラムに格納されているデータをデコードします。

sqlite3 keystore.db "select addrHex from privatekey where='mexico';"



この相談は、私たちに16進数形式の秘密鍵を与え、これは、トランザクションの受信または送信の基本的な部分です。このkeystore.dbデータベースのバックアップを取っておくことが繰り返し推奨されています。

テーブルpublicaddrとデータベースのデザインpublickeys.db。

\$ sqlite3

SQLiteバージョン3.32.2 2020-06-20 15:25:24

使用方法のヒントは「.help」を入力してください。

トランジエントなインメモリデータベースに接続されています。

.open FILENAME"を使用して、永続的なデータベースで再開します。

sqlite> .open publickeys.db

sqlite> CREATE TABLE publicaddr (id integer primary key AUTOINCREMENT NOT NULL, pubMBC VARCHAR NOT NULL, pubHex VARCHAR NOT NULL, pubBin BLOB NOT NULL)。

sqlite> .quit

次に、公開テーブルの作成は、以下のSQL文をセグメント化して表示します。

```
sqlite> CREATE TABLE publishedaddr (
...> id 整数 AUTOINCREMENT
...> pubMBC VARCHAR NOT NULL。
...> pubHex VARCHAR NOT NULL。
...> pubBin BLOB NOT NULL
...> );
sqlite> .tables
公開アドレス
sqlite> .quit
```

```
$ sqlite3
SQLITE version 3.32.2 2020-06-04 12:58:43
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open publickeys.db
sqlite> CREATE TABLE publicaddr (
    ...> id integer primary key AUTOINCREMENT,
    ...> pubMBC VARCHAR NOT NULL,
    ...> pubHex VARCHAR NOT NULL,
    ...> pubBin BLOB NOT NULL
    ...> );
sqlite> .tables
publicaddr
sqlite> .quit
$
```

24.別館「RESTful SQLiteのGET/POSTコマンド」。

次に、バックアップネットワークのRestful SQLiteで使用できるさまざまなコマンドを示します。これらはgithub (<https://github.com/olsonpm/sqlite-to-rest>) の独自開発から相談を受けています。

CRUD操作（Create、Read、Update、Delete） RESTful。

以下に、RESTful API が疑似例の形で利用可能な操作のリストを示します。我々は、ベースの "op.sqlite" と、トランザクションのための2つの関連する "trans" テーブルと、2つの列 `id INTEGER PRIMARY KEY` を持つソースアドレス、デスティネーションアドレスと資産価値のための別の "sign" を想定しています。

制限事項に記載されているように、メソッド (DELETE と POST) は一度に1つの行にしか影響しないことに注意してください。

OBTAİN これにより、最大のバリエーションが可能になります。後ほど、利用可能なすべてのクエリ演算子を見てみましょう。

URLのすぐ下に見出しを指定することができます。

全行のtransRequests

/trans?id=1Where
id = 1

トランス
範囲 rows=0-2
最初の3行

/トランス
範囲 rows=-5最後の
5行

/トランス
範囲 行数0から

サーバが提供できる行数は、実際にはmaxRangeと総行数のどちらか少ない方になります。

/トランス
範囲: 行=1から

サーバが提供できる数だけ、1行目から開始します。

トランス

順序:

名前昇順

トランス

order: name

descOrdered by name descending

/トランス

順序: 名前 desc, id

投稿されたが、最初は名前の降順でソートされ、同点の場合はIDの昇順でソートされる。

/trans?id>1

ここで、id > 1

/trans?id>=2&id<5

ここで、id >= 2 と id < 5

/trans?name_NOTNULL

名前がヌルではない場合

/trans?name_ISNULLWhereの

名前がNULLの場合

/trans?id!=5&name_LIKE'Spotted%'.

ここで id != 5 で名前が "Spotted%" のような場合(引用符は無視)

/trans?id>=1&id<10&name_LIKE'Avery%'.

order: name desc, id range: rows=2-4

例のために寄稿しました。

"Avery%"のような名前を持つ1から9までの識別子を持つトランザクションを、名前の降順、識別子の昇順で取得し、結果の3行目から5行目までを取得します。またはSQLで。

DELETE すべての主キーを値に等しく設定したクエリ文字列が必要です。これには、最大1行削除が必要です。

/trans?id

=1 ID=1のトランスを削除します。

代わりにトランザクションが id と name で構成されるメインキーを持っていた場合。

/trans?id=1&name='Avery IPA'.

POST作成

クエリ文字列を渡してはいけません。クエリ文字列が渡された場合は、POST更新を想定しています。すべてのPOSTリクエストは、ヘッダのコンテンツタイプであるapplication / jsonを渡さなければなりません。

ボディには、キャンセルできないすべてのPRIMARY KEYカラムが含まれていなければならず、INTEGRATEではないことに注意してください。そうでなければ、どのフィールドが失われたかを示す400 応答が送信されます。null可能なカラムはnullになり、integer primary keyカラムはsqlite3の仕様に従って自動的に増加します。

URLのすぐ下にJSONデータを指定します。

トランス

{"id":1,"name":"Serendipity"} id = 1、 name = 'Serendipity'でトランスを作成します。

トランス

{"id":1} id = 1、 name = NULLのトランスを作成します。

/トランス

"名前": "セレンディピティ"

sqlite3仕様 INTEGER PRIMARY KEYで増加した以下の値にidを設定したトランザクションを作成します。

/トランス

{}

id を増やし、名前を NULL に設定したトランザクションを作成します。

POST更新

クエリ文字列を含む必要があります。クエリ文字列がない場合は、POSTの作成を前提とします。POSTの作成と同様に、ヘッダのコンテンツタイプはapplication / jsonが必要です。

クエリ文字列には、1つの行だけが更新されるように、すべての主キーを含める必要があります。誤った値が渡された場合は、問題のあるキーとともに 400 が返されます。

リクエストのボディは空ではないオブジェクトを含み、カラム名に対応する有効なキーを含まなければなりません。

URLのすぐ下にJSONデータを指定します。

/trans?id=1

{ "id": 2 }。

IDが1のトランザクションを2にして更新します。

/trans?id=1

{ "name": "MCBza45Rt56cvbgfdR2Swd788kj" }。

トランザクション名または値を「MCBza45Rt56cvbgfdR2Swd788kj」に設定し、IDが1のトランザクションを更新します。

取引デスクの代わりにidと名前で構成されたメインキーを持っていた場合。

/trans?id=1&name=MCBza45Rt56cvbgfdR2Swd788kj

{ "name": "MCB3ofFG5Hj678MNb09vLdfaaSx" }

idが1でアドレスがMCBza45Rt56cvbgfdR2Swd788kjのトランザクションを MCB3ofFG5Hj678MNb09vLdfaaSxに設定して更新します。

参考

isSqliteFile

ファイルの最初の16バイトをチェックして、'sqlite format 3'の後にヌルバイトが続くかどうかを確認してください。

isDirectory

fs.statsSync の結果に .isDirectory を付加した結果を返す。

イズファイル

fs.statsSync の結果に .isFile を続けて返す。

G E T コンサルティング事業者

照会条件には、接続記号、例えば id>5 & name = MCBza45Rt56cvbgfdR2Swd788kj を付けなければなりません。

二項演算子（後の値を必要とする）マン。

=
!=
>=
<=
>
<
ライク

LIKEが特別なのは、シンプルなオープニングとエンディングの引用符が必要だからです。そうでない場合は、分析が完了できなかった箇所と期待されていた内容を示す400エラーが発生します。例については、CRUD RESTful オペレーションを参照してください。

単一演算子（列の名前の後に続く必要があります

IS NULL

ノットヌル

ルータ構成オブジェクト

isLadenPlainObject

このオブジェクトの目的は、sqliteルータの一般的な設定を提供することです。以下のプロパティが許可されています。

prefix: isLadenString koa-router のプレフィックスビルダオプションに渡す文字列。例えば、スケルトンサーバーはプレフィックスを指定しないので、httpドメインのルートから直接ビールAPIをヒットさせることができます: // localhost: 8085 / trans. プレフィックスを' / api'に設定すると、http: // localhost: 8085 / api / trans. にリクエストを送信します。

allTablesAndViews: 表形式の設定オブジェクト

オープンキュービットドットコム

このオブジェクトで指定した設定は、すべてのテーブルとビューに適用されます。

tablesAndViews: isLadenPlainObject 過去のオブジェクトは、データベースのカラムまたはビューの名前と一致するキーを持っている必要があります。そうでない場合は、フレンドリーなエラーメッセージが表示されます。各テーブルとビューの値は、表形式の設定オブジェクトでなければなりません。

表形式設定オブジェクト

isLadenPlainObject ビューやテーブルに設定できる設定を表します。以下のプロパティを許可します。

maxRange: isPositiveNumber

デフォルトのアプリケーション: 1000

これは、サーバーがリクエストを許可する最大の範囲です。GETリクエストが範囲ヘッダなしで到着した場合、この仕様では、リソース全体が欲しいと仮定しています。GETの結果として得られた行数がmaxRangeよりも大きい場合、カスタムmax-rangeヘッダを使用して416のステータスが返されます。アプリケーションのデフォルト値は、作者が必要に応じてmaxRangeを設定することを期待して、意図的に保守的になっています。

「無限大」は有効な正の数であることに注意してください。

flags: isLadenArray

現在、受け入れられるインジケータは文字列「sendContentRangeInHEAD」のみです。設定されている場合、HEADリクエストは利用可能なコンテンツの範囲を content-rangeの形式で返します : * / <max-range>.設定可能な理由は、サーバーの負荷やテーブルの大きさによっては、最大範囲の計算がそれに見合う以上の作業になる可能性があるからです。

カスタムヘッダー

アプリケーション

order: このヘッダはGETのためだけに定義されており、sqlのORDER BYと同等と考えることができます。コンマで区切られたカラム名と、オプションでスペースと文字列 'asc' または 'desc' を含まなければ

ばなりません。誤った注文値が送信された場合は、400のレスポンスでどの注文値かが示されます。

回答

すべてが必ずしもカスタマイズされているわけではありませんが、すべての用途が仕様外であるため、明確にする必要があります。

G E T

max-range: 要求された行数が設定されたmaxRangeを超えた場合にこのヘッダが返されます。リクエストが範囲ヘッダを指定しなくても、そのリソースの結果として得られる行数は検証されることに注意してください。

内容の範囲: rfc7233の状態

ステータスコード206（部分的な内容）と416（不満足な範囲）のみが、Content-Rangeの意味を記述しています。

sqlite-to-restがステータスコード200で応答した場合、コンテンツ範囲ヘッダは、<row start> - <row end> / <row count>のフォーマット206で送信される。

リクエストが範囲ヘッダなしで送信され、その結果の行数がmaxRangeを超えた場合、内容の範囲を * / <行数>の416形式に設定した400が返されます。

このヘッダは HEAD レスポンスで返すことができることに注意してください。

accept-order: リクエストヘッダの順序に不正な構文や不正なカラム名が指定されている場合に返されます。詳しくは、下記のHEAD -> accept-orderをご覧ください。

25.付録「JavaコードSQLite-Redisコネクタ」。

SQLite-Redisの両データベースの通信間のリンクのコードを以下に示す。基本的にはご覧の通り、コネクタはSQLiteのフォーマットをRedisが受信するデータ（トランザクション）の一般的な文字列に変更します。

上記のプロセスは、接続されているすべてのノードに対して、現在のトランザクションキューを処理するための可能性のある候補であるトランザクションキューのトリガとして動作します。

Redisデータベースが持っているMaster-Slave構成でトランザクションキューを受信すると、リアルタイムで全ノードに均等に情報を配信します。

もう一つの基本的なポイントは、すべてのノードがそのクロックに同期されなければならないということですが、これは、我々はサーバーパールNTP（ネットワークタイムプロトコル）へのクエリの機能を使用して前に見たように行われます。

このコネクタは、すべてのトランザクションのMerkleルート・ツリーを計算することで、最初のレベルのデータ・セキュリティ・レビューも実行します。

SQLite-Redis コネクタのベースコードです。

```
インポートjava.sql.*.  
インポートjava.util.*.  
インポートjava.util.array.  
インポート java.util.list.  
java.util.arrayをインポートします。  
import java.security.*.  
import java.security.MessageDigest.  
import redis.client.jedis.Jedis.  
class RediSqlite{(レッドSQLite)  
    public String previousHash.  
    public String merkleRoot.  
    public static ArrayList<String> transactions = new ArrayList<String>().  
    public static ArrayList<String> treeLayer = new ArrayList<String>().  
    public static String getMerkleRoot() { {  
        int count = transactions.size().  
        int item = 1.  
        奇数の int = 0.  
        ArrayList<String> previousTreeLayer = transactions.
```

```

while(count > 1) {
    treeLayer = new ArrayList<String>().
    for(int i=1; i <= count ; i=i+2) {

        if (!esPar(count) && i == count) odd = 1.
            treeLayer.add(applySha256(previousTreeLayer.get(i-item)
+ previousTreeLayer.get(i-impar))。

    }
    アイテム = 1.
    奇数 = 0 である。
    count = treeLayer.size().
    previousTreeLayer = treeLayer.
    }
    String merkleRoot = (treeLayer.size() == 1) ? treeLayer.get(0) : "".
    merkleRoot を返します。
}

//メルクルツリーが偶数か奇数かの定義
static boolean enPar(int number){
    if (numero%2==0) return true; else return false.
}

public static String applySha256(String input){
    試してみる

    MessageDigest digest = MessageDigest.getInstance("SHA-256").
    //sha256を入力に適用します。
    byte[] hash = digest.digest(input.getBytes("UTF-8").
    StringBuffer hexString = new StringBuffer(); // これは16進数としてハッシュを含みます。
    for (int i = 0; i < hash.length; i++) {
        文字列 hex = Integer.toHexString(0xff & hash[i]).
        if(hex.length() == 1) hexString.append('0').
        16xString.append(hex).
    }
    hexString.toString() を返します。
}

```

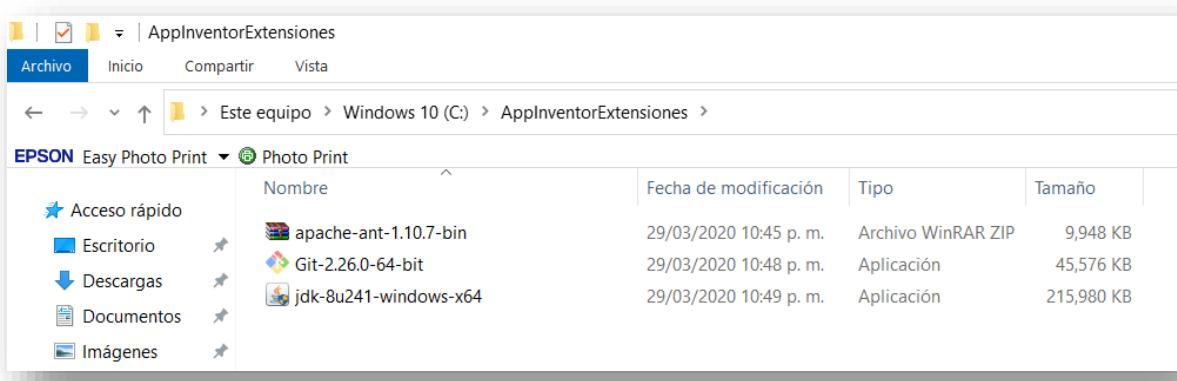
```
        catch(Exception e) {
            throw new RuntimeException(e);
        }
    }

public static void main(String args[]){
    試す
    Connection      con=DriverManager.getConnection("jdbc:sqlite:C:/memo/sqlite-tools-
win32-x86-3310100/trans.sqlite3");
    //ローカルホスト上のRedisサーバへの接続
    Jedis = new Jedis ("localhost");
    jedis.auth("memo1234")を使用しています。
    ステートメント stmt=con.createStatement();
    String sql = "SELECT * FROM brewery";
    ResultSet rs=stmt.executeQuery(sql);
    while(rs.next()) {
        transactions.add(rs.getString(2));
        jedis.set("LATAM:"+String.valueOf(rs.getInt(1)), "["+"\""+rs.getString(2)+"\""+","+"\""+rs.get-
String(3)+"+","+"+"\""+rs.getString(4)+"\"");
    }
    jedis.set("LATAM:merkleroot", getMerkleRoot());
    System.out.println("ArrayListの要素数：" +treeLayer.size());
    を .close() で使用します。
}catch(Exception      e){      System.out.println(e);}catch(Exception      e){
System.out.println(e);}catch(Exception  e){  System.out.println(e); }catch(Exception  e){
System.out.println(e); }catch(Exception  e){  System.out.println(e); }catch(Exception  e){
System.out.println(e); }catch(Exception  e){  System.out.println(e); }catch(Exception  e){
System.out.println(e); }catch(Exception  e){  System.out.println(e); }catch(Exception  e){
System.out.println(e);
}
```

26. 別館「開発者のためのミニBlocklyChain」。

この付属書では、Blockly環境用のJavaプログラミング言語に基づいて外部モジュールを生成する方法の設定、インストール、基本的な使い方をレビューし、ビジネスケースごとに特化したモジュールを作成し、ミニBlocklyChainシステムに機能性を挿入したり、モバイルアプリケーションに他の機能性を追加したりできるようになります。

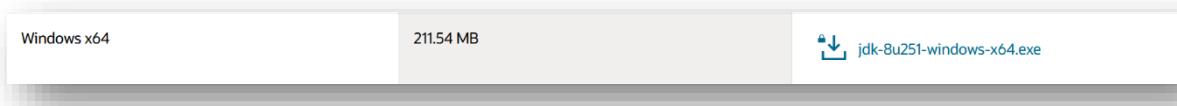
Windowsシステム内に「AppInventorExtensions」というディレクトリを作成し、以下の公開ソフトウェアパッケージをダウンロードします。



1. - JDK (Java Development Kit) の最新版をダウンロードします。

例：jdk-8u251-windows-x64.exe (211 MB)

<https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>



2.-JAVAを使ってアプリケーションを構築するApache

Antライブラリ、

<http://ant.apache.org/bindownload.cgi>、私の場合はAnt 1.10.8 (Binary Distributions) (apache-ant-1.10.8-bin.zip)をダウンロードしました。もっと進化したバージョンがあるかもしれません

◦

1.9.15 release - requires minimum of Java 5 at runtime

- 1.9.15 .zip archive: [apache-ant-1.9.15-bin.zip \[PGP\] \[SHA512\]](#)
- 1.9.15 .tar.gz archive: [apache-ant-1.9.15-bin.tar.gz \[PGP\] \[SHA512\]](#)
- 1.9.15 .tar.bz2 archive: [apache-ant-1.9.15-bin.tar.bz2 \[PGP\] \[SHA512\]](#)

1.10.8 release - requires minimum of Java 8 at runtime

- 1.10.8 .zip archive: [apache-ant-1.10.8-bin.zip \[PGP\] \[SHA512\]](#)
- 1.10.8 .tar.gz archive: [apache-ant-1.10.8-bin.tar.gz \[PGP\] \[SHA512\]](#)
- 1.10.8 .tar.bz2 archive: [apache-ant-1.10.8-bin.tar.bz2 \[PGP\] \[SHA512\]](#)
- 1.10.8 .tar.xz archive: [apache-ant-1.10.8-bin.tar.xz \[PGP\] \[SHA512\]](#)

Old Ant Releases

Older releases of Ant can be found [here](#). We highly recommend to not use those releases but upgrade to Ant's [latest](#) release.

3. - あなたのサイトからGit Bashをインストールしました <https://git-scm.com/download/win>

Downloading Git



You are downloading the latest (**2.27.0**) **64-bit** version of **Git for Windows**. This is the most recent [maintained build](#). It was released **9 days ago**, on 2020-06-01.

[Click here to download manually](#)

Other Git for Windows downloads

[Git for Windows Setup](#)

[32-bit Git for Windows Setup](#).

[64-bit Git for Windows Setup](#).

[Git for Windows Portable \("thumbdrive edition"\)](#)

[32-bit Git for Windows Portable](#).

[64-bit Git for Windows Portable](#).

The current source code release is version **2.27.0**. If you want the newer version, you can build it from [the source code](#).

4. - 「Apache Ant」の解凍が終わったら、例えば二重フォルダにします。

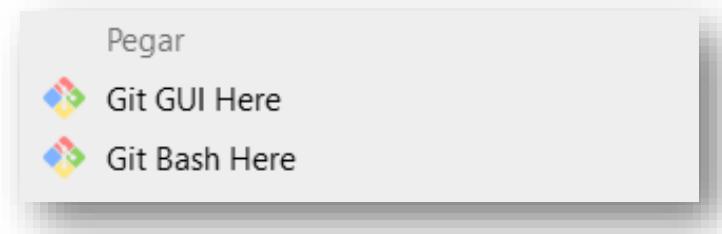
C:\AppInventorExtensions\apache-ant-1.10.8-bin\apache-ant-1.10.8-bin

- C:\Apache-ant-1.10.8-bin に変更します。

	Nombre	Fecha de modificación	Tipo	Tamaño
o	bin	01/09/2019 11:43 a. m.	Carpetas de archivos	
o	etc	01/09/2019 11:43 a. m.	Carpetas de archivos	
o	lib	01/09/2019 11:43 a. m.	Carpetas de archivos	
o	manual	01/09/2019 11:43 a. m.	Carpetas de archivos	
o	CONTRIBUTORS	01/09/2019 11:43 a. m.	Archivo	7 KB
o	contributors	01/09/2019 11:43 a. m.	Documento XML	33 KB
o	fetch	01/09/2019 11:43 a. m.	Documento XML	14 KB
o	get-m2	01/09/2019 11:43 a. m.	Documento XML	5 KB
o	INSTALL	01/09/2019 11:43 a. m.	Archivo	1 KB
o	KEYS	01/09/2019 11:43 a. m.	Archivo	94 KB
o	LICENSE	01/09/2019 11:43 a. m.	Archivo	15 KB
o	NOTICE	01/09/2019 11:43 a. m.	Archivo	1 KB
o	patch	01/09/2019 11:43 a. m.	Documento XML	2 KB
o	README	01/09/2019 11:43 a. m.	Archivo	5 KB
o	WHATSNEW	01/09/2019 11:43 a. m.	Archivo	250 KB

5. - Git Bashをインストールしました。インストール時のデフォルトのままにしておきました。

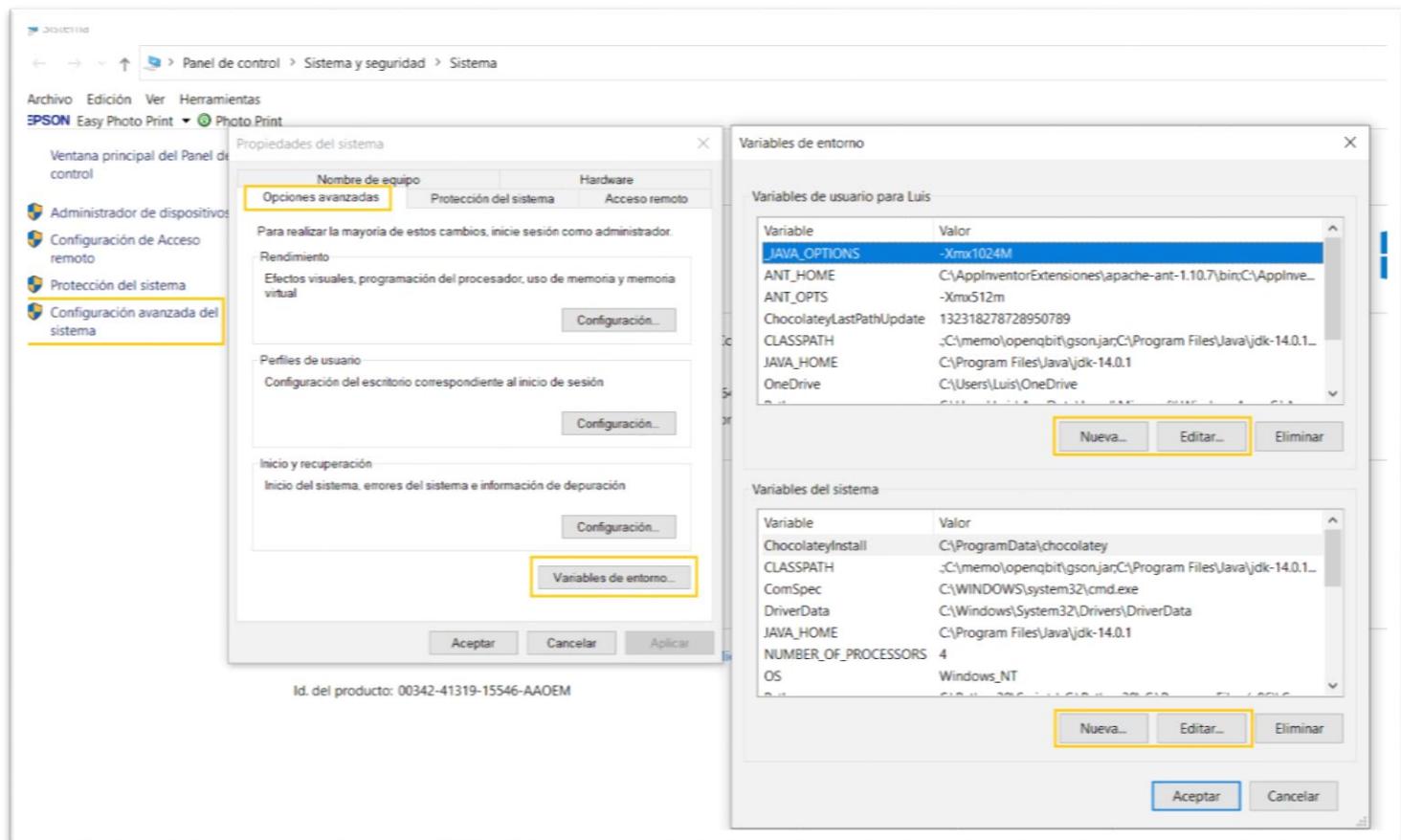
- 後日、ファイルブラウザの左ボタンをクリックすることで、Windowsのスタートボタンにリンクがあることがわかります。



6. - Java SE開発キットをインストールしました。Windowsのバージョンによっては、コンピュータを再起動する必要があるかもしれません。

Windows システム環境変数。環境変数を作成します。そのためには、私たちはそうします。

コントロールパネル→システム→システムの詳細設定→詳細オプション→環境変数。



新しい変数を置くか、既存の変数を編集するかに応じて、"新規..."または"編集..."ボタンを押します。

すでに確立されているアドレスに追加する場合は、セミコロンで区切っています。

Johnのユーザー変数セクションでは、これらの新しい...

JAVA_OPTIONSでは、値 -Xmx1024mを指定しています。

ANT_HOME we put of Value C: I'm sorryApplInventorExtensions\apache-ant-1.10.8-bin [つまり、apache-antを解凍したフォルダ]です。

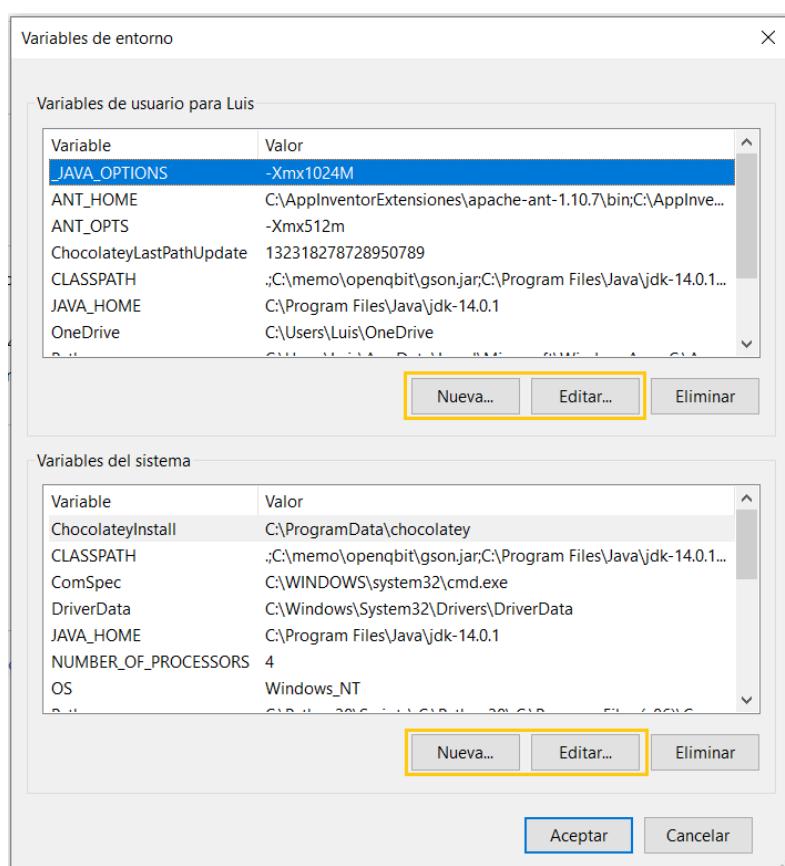
ANT_OPTSには値 -Xmx256Mを指定しました。

JAVA_HOMEはC:\Program Files\jdk1.8.0_131に設定されています。他の値がある場合は変更してください。jdkではなくjdrなので注意してください。

CLASSPATHは、値を入れています。

PATH に %ANT_HOME%bin;%JAVA_HOME%bin を追加しました。

注意: 変数はセミコロンで区切られています: variable-one; variable-n; variable-n+1



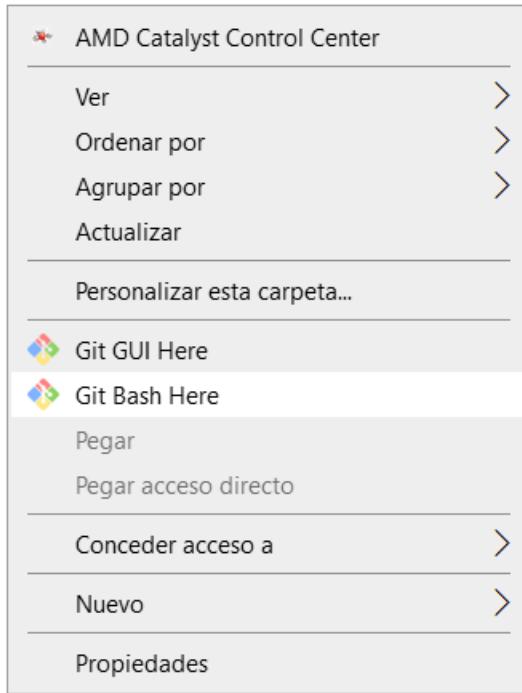
7. - コンピュ

-タでの App

Inventor クローンの作成

弊社のサーバー（PC）にApp Inventorの「クローン」コピーを作成し、インターネットから直接ダウンロードして、そのコピーを作成します。

そのためには、Git Bashアプリケーションを使って、それをクリックしてターミナルを開きます。



Git Bashターミナルでコマンドを実行します。

```
$ git clone https://github.com/mit-cml/appinventor-sources.git
```

A screenshot of a Git Bash terminal window. The window title is 'MINGW64 / C:/Users/Luis'. The command entered is '\$ git clone https://github.com/mit-cml/appinventor-sources.git'. The terminal output shows the cloning process: 'Cloning into 'appinventor-sources'...' followed by details about the objects being transferred, including the number of objects, the size of the received objects (553.30 MiB), the size of the pack file (494.00 KiB/s), and the completion of the cloning process. The command concludes with 'Checking out files: 100% (1758/1758), done.' The window has a standard Windows-style title bar with minimize, maximize, and close buttons.

リポジトリがあるサイト。

<https://github.com/mit-cml/appinventor-sources/>

App Inventor のソースが入った appinventor-sources というフォルダが作成されます。

Nombre	Fecha de modificación	Tipo	Tamaño
.github	30/03/2020 01:05 a. m.	Carpeta de archivos	
appinventor	03/05/2020 05:32 p. m.	Carpeta de archivos	
.gitmodules	30/03/2020 01:05 a. m.	Documento de tex...	1 KB
bootstrap	30/03/2020 01:05 a. m.	Shell Script	2 KB
LICENSE	30/03/2020 01:05 a. m.	Archivo	12 KB
README.md	30/03/2020 01:05 a. m.	Archivo MD	11 KB
sample-	30/03/2020 01:05 a. m.	Documento de tex...	1 KB
Vagrantfile	30/03/2020 01:05 a. m.	Archivo	1 KB

C:\Users - Apppinventor-sources\backup - Apppinventor-components -rc-openqbitに以下のディレクトリを作成しました。

その中に OpenQbitQRNGch.java というテストプログラムをコピーしました。

Nombre	Fecha de modificación	Tipo	Tamaño
OpenQbitQRNGch	11/06/2020 01:39 a. m.	Archivo JAVA	5 KB

エクステンションの作成。

RESPECT THE CAPS AND MINUSCULES、それは同じハローとハローではありません。

アクセントを入れてはいけない。最初の拡張機能は量子乱数発生器だ

テキストエディタのメモ帳++を使用して、OpenQbitQRNGch.javaというファイルを作成し、以下のコードで乱数量子数を生成します。

```
// Quantum Random Number Generator QRNG Switzerland APIをサポートするための機能です。

パッケージ com.openqbit.OpenQbitQRNGch.
import com.google.appinventor.components.annotations.DesignerComponent.
import com.google.appinventor.components.annotations.DesignerProperty.
import com.google.appinventor.components.annotations.PropertyCategory.
import com.google.appinventor.components.annotations.SimpleEvent.
import com.google.appinventor.components.annotations.SimpleFunction.
import com.google.appinventor.components.annotations.SimpleObject.
import com.google.appinventor.components.annotations.SimpleProperty.
import com.google.appinventor.components.common.ComponentCategory.
import com.google.appinventor.components.common.PropertyTypeConstants.
import com.google.appinventor.components.runtime.util.MediaUtil.
import com.google.appinventor.components.runtime.*.
import java.io.*.

@DesignerComponent(version = OpenQbitQRNGch.VERSION.
    説明 =
    "API 量活数ジェネレータ サービスとしての量活数 QRNGaaS、スイスの会社によって開発された Quantis の量活数生成のための Web
    API - " + "ギリヨンタ - OpenQbit.com "。
    category = ComponentCategory.EXTENSION
    nonVisible = true となります。
    iconName = "http://www.pinntar.com/logoQbit.png")
    シンプルオブジェクト(external = true)

public class OpenQbitQRNGch extends AndroidNonvisibleComponent implements
Component { .

    public static final int VERSION = 1.
    public static final String DEFAULT_TEXT_1 = "".
    private ComponentContainer container.
    private String text_1 = "".

    public OpenQbitQRNGch(ComponentContainer container) { .
        // メソッドを呼びます。
        // @DesignerProperty(editorType =
        // PropertyTypeConstants.PROPERTY_TYPE_STRING, defaultValue =
        // OpenQbitQRNGch.DEFAULT_TEXT_1 + " ")
        // @SimpleProperty(description = "API Get Quantum Random Number Generator,
        // 0~1の間の量活数を取得します。生対数値を入力してください")
        SimpleFunction(description = "API Get Quantum Random Number Generator,
        0~1の間の量活数を取得します。生対数値を入力してください")
        public String APIGetQRNGdecimal(String qty) throws Exception {
            文字列 url = "http://random.openqu.org/api/rand?size=" + qty.
```

```

String[] command = { "curl", url }.

ProcessBuilder process = new ProcessBuilder(command)。
オセス。
p = process.start()。
        BufferedReader reader = new BufferedReader(new
InputStreamReader(p.getInputStream())).
        StringBuilder builder = new StringBuilder().
        文字列 line = null。
        while ((line = reader.readLine()) != null) {
            builder.append(line)。

builder.append(System.getProperty("line.separator") )。
}
文字列 result = builder.toString()。
//System.out.print(result) です。
結果を出します。

}

@SimpleFunction(description =
"API量生成ジェネレータを取得し範囲の最小値から最大値までの間で数個のランダムな数値を生成するため、数値の可調整範囲を指定してください")
public String APIGetQRNGinteger(String qty, String min, String max)
throws Exception {
    文字列 URL = "http://random.openqu.org/api/randint?size=" + qty + "&min=" +
    min + "&max=" + max
    String[] command = { "curl", url }.

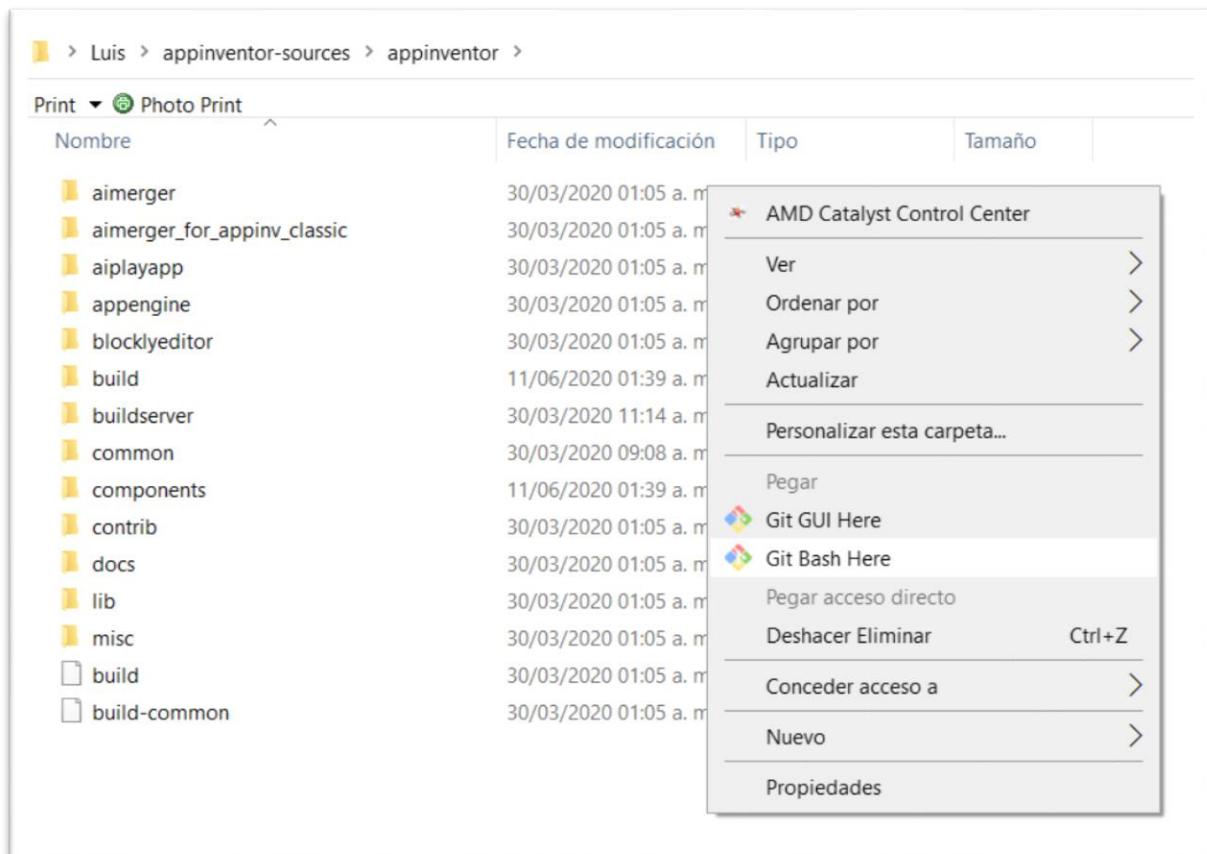
    ProcessBuilder process = new ProcessBuilder(command)。
    オセス。
    p = process.start()。
        BufferedReader reader = new BufferedReader(new
InputStreamReader(p.getInputStream())).
        StringBuilder builder = new StringBuilder().
        文字列 line = null。
        while ((line = reader.readLine()) != null) {
            builder.append(line)。

builder.append(System.getProperty("line.separator") )。
}
文字列 result = builder.toString()。
//System.out.print(result) です。
結果を出します。

}
}

```

Git Bash を実行する際には、以下のパスに位置を合わせて実行します: C: Luis-appinventor-sources-appinventor。このディレクトリで、マウスの左ボタンをクリックしてGit Bashターミナルを選択します。



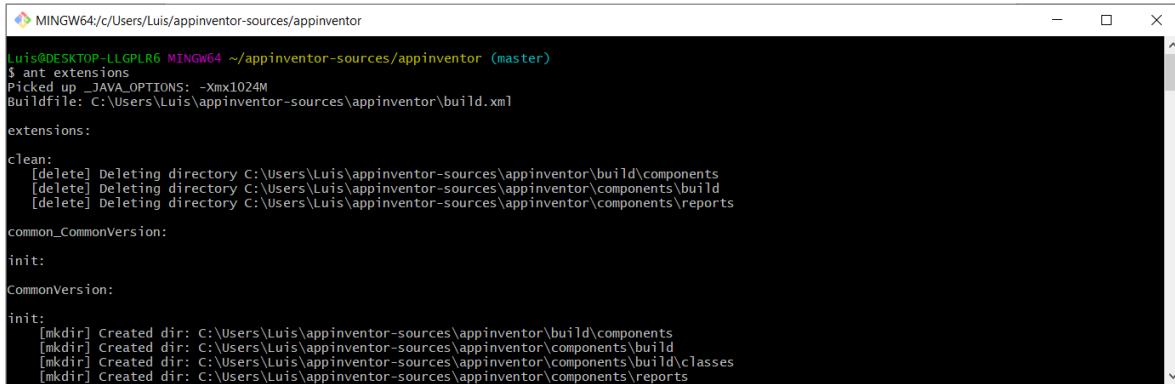
Git bash端末は以下のディレクトリに配置されます。

C:\Illustrator-Sources\appinventor-sources\appinventor (master)

```
MINGW64:/c/Users/Luis/appinventor-sources/appinventor
Luis@DESKTOP-LLGPLRG MINGW64 ~/appinventor-sources/appinventor (master)
$ |
```

Git Bashターミナルで、以下のコマンドを実行します。

アリの拡張子は、\$アリの拡張子



```
Luis@DESKTOP-LLGPLR6 MINGW64 ~/appinventor-sources/appinventor (master)
$ ant extensions
Picked up _JAVA_OPTIONS: -Xmx1024M
Buildfile: C:/Users/Luis/appinventor-sources/appinventor/build.xml

extensions:

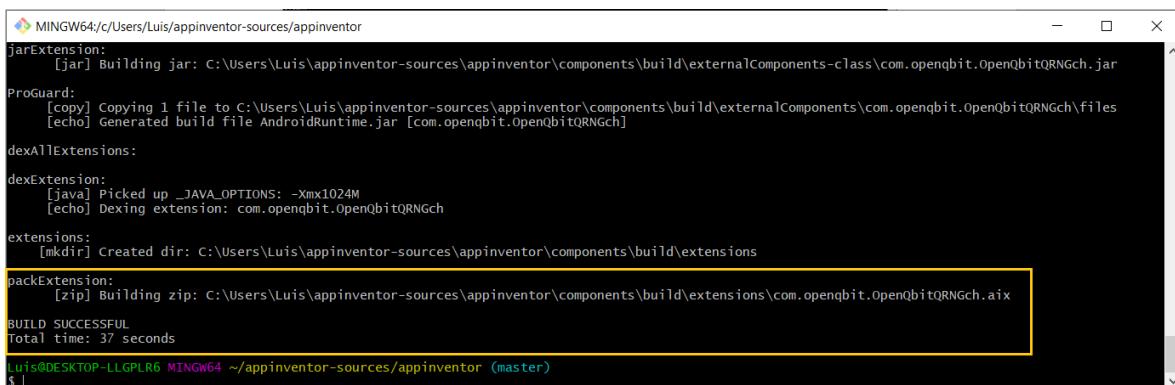
clean:
[delete] Deleting directory C:/Users/Luis/appinventor-sources/appinventor/build/components
[delete] Deleting directory C:/Users/Luis/appinventor-sources/appinventor/components/build
[delete] Deleting directory C:/Users/Luis/appinventor-sources/appinventor/components/reports

common_CommonVersion:
init:
CommonVersion:
init:
[mkdir] Created dir: C:/Users/Luis/appinventor-sources/appinventor/build/components
[mkdir] Created dir: C:/Users/Luis/appinventor-sources/appinventor/components/build
[mkdir] Created dir: C:/Users/Luis/appinventor-sources/appinventor/components/build/classes
[mkdir] Created dir: C:/Users/Luis/appinventor-sources/appinventor/components/reports
```

全てがうまくいったのならば、それを手に入れよう。BUILD SUCESSFULLY。

私たちの拡張子は、作成されています…

注意:拡張子フォルダの内容は、アリの拡張子を作成するたびに削除され、再作成されます。



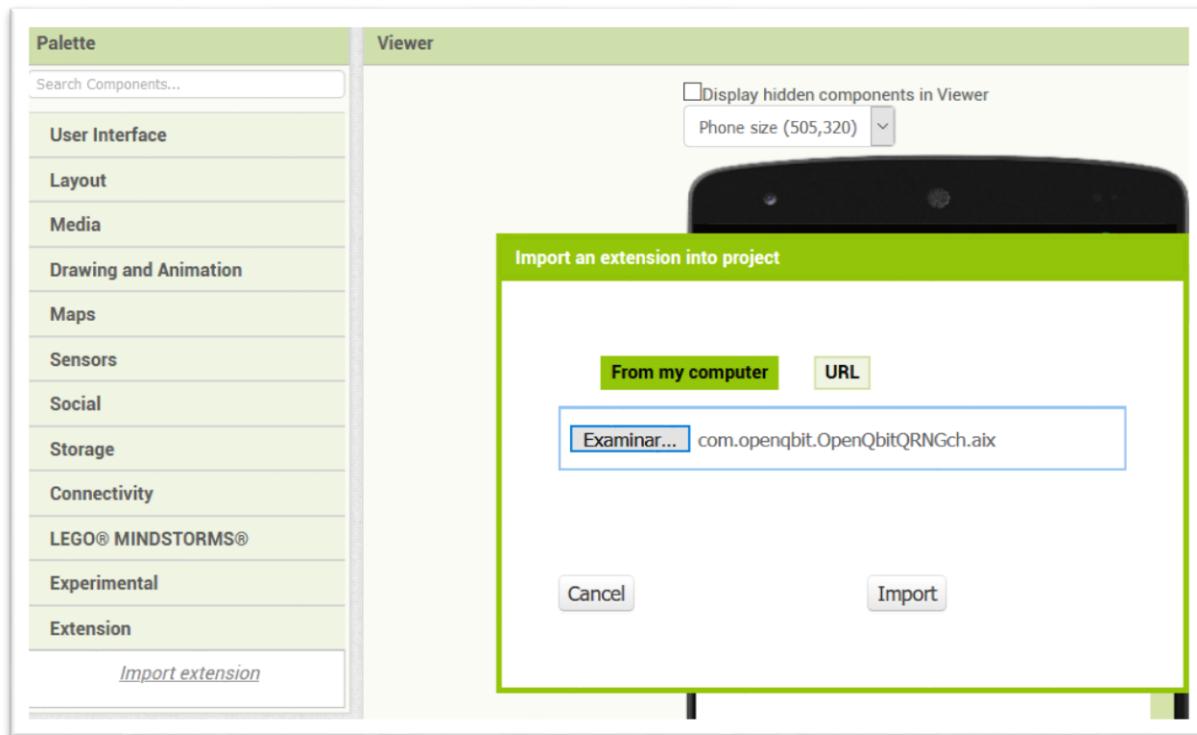
```
Luis@DESKTOP-LLGPLR6 MINGW64 ~/appinventor-sources/appinventor (master)
jarExtension:
[jar] Building jar: C:/Users/Luis/appinventor-sources/appinventor/components/build/externalComponents-class/com.openqbit.OpenQbitQRNGch.jar
ProGuard:
[copy] Copying 1 file to C:/Users/Luis/appinventor-sources/appinventor/components/build/externalComponents/com.openqbit.OpenQbitQRNGch/files
[echo] Generated build file AndroidRuntime.jar [com.openqbit.OpenQbitQRNGch]
dexAllExtensions:
dexExtension:
[java] Picked up _JAVA_OPTIONS: -Xmx1024M
[echo] Dexing extension: com.openqbit.OpenQbitQRNGch
extensions:
[mkdir] Created dir: C:/Users/Luis/appinventor-sources/appinventor/components/build/extensions
packExtension:
[zip] Building zip: C:/Users/Luis/appinventor-sources/appinventor/components/build/extensions/com.openqbit.OpenQbitQRNGch.aix
BUILD SUCCESSFUL
Total time: 37 seconds
Luis@DESKTOP-LLGPLR6 MINGW64 ~/appinventor-sources/appinventor (master)
```

そのフォルダに行こう

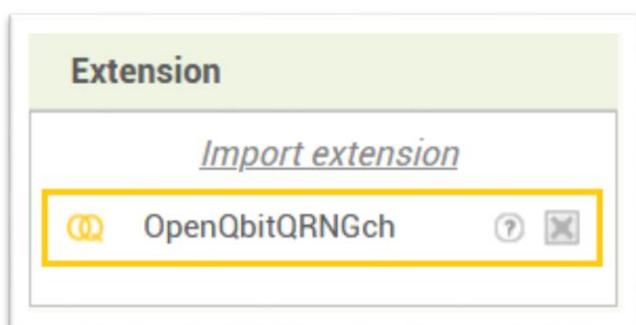
とファイルcom.openqbit.OpenQbitQRNGch.aixをコピーして、このファイルは私たちの拡張子です。

すでに App Inventor または別の Blockly システムにアカウントを持っている場合は、新しい拡張機能を追加してテストするために入力しました。

拡張機能のオプションがある下の方に移動し、「拡張機能のインポート」をクリックします。



「インポート」ボタンを押しました。これでブロック(APIGetQRNGdecimal)と(APIGetQRNGinteger)が使えるようになりました。



最初に作成した機能拡張はすでに持っています。

27. 別紙「BlocklyCodeスマートコントラクト

BlocklyCodesはjava言語で作られたプログラムです。一般的に「スマートコントラクト」と呼ばれるこの手のプログラムを作成するための拡張機能は、ユーザー（企業や人）間の契約を処理する方法です。

このブロック（BlocklyCode）は、すでに確立されたパラメータを持っているプログラムに実装されており、「スマートコントラクト」を支配する前提条件が満たされているときに、自動方法でミニBlocklyChainシステムによって実行される可能性がある契約の種類に応じて、そのこと。

示唆されたまたは入力パラメータ「入力」を考慮するためのパラメータは

有効期限

参照日

有効期限

参照時間

参照資産

固定資産合計

変動資産

契約メンバー

確認済みデータ（文字列）

確認済みデータ（ファイル名）

変数イベント

固定イベント

ドキュメントの検証

文字列の検証

署名の有効性

定義された間隔（整数）

小数点以下の間隔

最低限の設定

最大値を設定

ブロック（**BlocklyCode**）の使用を開始する前に、我々は最初に"スマートコントラクト"の実行を実行するシステムをインストールする必要があります、これはTermux OpenJDKとOpenJREのターミナルにインストールすることによって行われます。

OpenJDK（Open Java Development Kit）。- これは、Javaでプログラムを開発するためのツールであり、ライブラリ、コンパイラ、JVM（Java Virtual Machine）が含まれています。

OpenJRE（Open Java Runtime Environment）。- javaプログラムのみを実行するためのツールです。JVM（Java仮想マシン）が含まれています。

Mini BlocklyChainシステムを構成するノードのTermux端末にOpenJREとOpenJDKのインストールを進めます。

お部屋を設置しました

apt install - と wget



A screenshot of a Termux terminal window. The terminal shows the command \$ apt install wget being entered, with a yellow arrow pointing to the 'wget' part of the command. The output of the command follows, detailing the package list reading, dependency building, state information, and the download and unpacking of wget version 1.20.3-2. The terminal also includes a standard QWERTY keyboard layout at the bottom.

```
$ apt install wget
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  wget
0 upgraded, 1 newly installed, 0 to remove and 0
not upgraded.
Need to get 206 kB of archives.
After this operation, 430 kB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-packa
ges-24 stable/main arm wget arm 1.20.3-2 [206 kB]
Fetched 206 kB in 1s (128 kB/s)
Selecting previously unselected package wget.
(Reading database ... 16936 files and directorie
s currently installed.)
Preparing to unpack .../archives/wget_1.20.3-2_a
rm.deb ...
Unpacking wget (1.20.3-2) ...
Setting up wget (1.20.3-2) ...
$
```

OpenJREをダウンロードしました。

wget https://archive.org/download/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb

```
192.168.1.68 - PuTTY
$ wget https://archive.org/download/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
--2020-06-12 00:55:14-- https://archive.org/download/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
Resolving archive.org... 207.241.224.2
Connecting to archive.org|207.241.224.2|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://ia902807.us.archive.org/19/items/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb [following]
--2020-06-12 00:55:15-- https://ia902807.us.archive.org/19/items/openjdk-9-jre-headless_9.2017.8.20-1_x86_64/openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
Resolving ia902807.us.archive.org... 207.241.232.107
Connecting to ia902807.us.archive.org|207.241.232.107|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 231576072 (221M) [application/x-debian-package]
Saving to: 'openjdk-9-jre-headless_9.2017.8.20-1_arm.deb'

openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          8%[=====]   18.90M  1.06MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          8%[=====]   19.10M  1.10MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          8%[=====]   19.30M  1.09MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          8%[=====]   19.52M  1.08MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          8%[=====]   19.76M  1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          9%[=====]   19.98M  1.06MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          9%[=====]   20.21M  1.10MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          9%[=====]   20.44M  1.11MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          9%[=====]   20.63M  1.09MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          9%[=====]   20.87M  1.09MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          9%[=====]   21.07M  1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          9%[=====]   21.32M  1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          9%[=====]   21.57M  1.07MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          9%[=====]   21.80M  1.06MB/s
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb          100%[=====]  220.85M  714KB/s  in 3m 53s

2020-06-12 00:59:14 (971 KB/s) - 'openjdk-9-jre-headless_9.2017.8.20-1_arm.deb' saved [231576072/231576072]

$
```

OpenJDKをダウンロードしました。

\$ wget https://archive.org/download/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb

```
192.168.1.68 - PuTTY
$ wget https://archive.org/download/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
--2020-06-12 01:04:10-- https://archive.org/download/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
Resolving archive.org... 207.241.224.2
Connecting to archive.org|207.241.224.2|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://ia802807.us.archive.org/19/items/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb [following]
--2020-06-12 01:04:12-- https://ia802807.us.archive.org/19/items/openjdk-9-jdk-headless_9.2017.8.20-1_x86_64/openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
Resolving ia802807.us.archive.org... 207.241.232.117
Connecting to ia802807.us.archive.org|207.241.232.117|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 34976 (34K) [application/x-debian-package]
Saving to: 'openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb'

openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb          100%[=====]  34.16K  ---KB/s  in 0.01s

2020-06-12 01:04:13 (2.74 MB/s) - 'openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb' saved [34976/34976]

$
```

OpenJDK、OpenJREのTermux端末からのインストールを行います。

```
apt install - and ./openjdk-9-jre-headless_9.2017.8.20-1_arm.deb ./openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb ./openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
```



```
$ ls
openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb
openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
$ apt install -y ./openjdk-9-jdk-headless_9.2017.8.20-1_arm.deb ./openjdk-9-jre-headless_9.2017.8.20-1_arm.deb
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'openjdk-9-jdk-headless' instead
of './openjdk-9-jdk-headless_9.2017.8.20-1_arm.
deb'
Note, selecting 'openjdk-9-jre-headless' instead
of './openjdk-9-jre-headless_9.2017.8.20-1_arm.
deb'
The following additional packages will be instal
led:
  ca-certificates-java freetype libpng
The following NEW packages will be installed:
  ca-certificates-java freetype libpng
  openjdk-9-jdk-headless
  openjdk-9-jre-headless
0 upgraded, 5 newly installed, 0 to remove and 0
not upgraded.
Need to get 668 kB/232 MB of archives.
After this operation, 376 MB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-pac
ages-24 stable/main arm ca-certificates-java all
20200101 [110 kB]
Get:2 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm libpng arm 1.6.37-2 [190
kB]
Get:3 https://dl.bintray.com/termux/termux-pac
ges-24 stable/main arm freetype arm 2.10.2 [368
kB]
Get:4 /data/data/com.termux/files/home/openjdk/o
penjdk-9-jre-headless_9.2017.8.20-1_arm.deb open
jdk-9-jre-headless arm 9.2017.8.20-1 [232 MB]
Get:5 /data/data/com.termux/files/home/openjdk/o
penjdk-9-jdk-headless_9.2017.8.20-1_arm.deb open
jdk-9-jdk-headless arm 9.2017.8.20-1 [35.0 kB]
```



```
Get:5 /data/data/com.termux/files/home/openjdk/o
penjdk-9-jdk-headless_9.2017.8.20-1_arm.deb open
jdk-9-jdk-headless arm 9.2017.8.20-1 [35.0 kB]
Fetched 668 kB in 23s (28.0 kB/s)
Selecting previously unselected package ca-certifi
cates-java.
(Reading database ... 16939 files and directo
ries currently installed.)
Preparing to unpack .../ca-certificates-java_202
00101_all.deb ...
Unpacking ca-certificates-java (20200101) ...
Selecting previously unselected package libpng.
Preparing to unpack .../libpng_1.6.37-2_arm.deb
...
Unpacking libpng (1.6.37-2) ...
Selecting previously unselected package freetype
.
Preparing to unpack .../freetype_2.10.2_arm.deb
...
Unpacking freetype (2.10.2) ...
Selecting previously unselected package openjdk-
9-jre-headless.
Preparing to unpack .../openjdk-9-jre-headless_9
.2017.8.20-1_arm.deb ...
Unpacking openjdk-9-jre-headless (9.2017.8.20-1)
...
Selecting previously unselected package openjdk-
9-jdk-headless.
Preparing to unpack .../openjdk-9-jdk-headless_9
.2017.8.20-1_arm.deb ...
Unpacking openjdk-9-jdk-headless (9.2017.8.20-1)
...
Setting up libpng (1.6.37-2) ...
Setting up freetype (2.10.2) ...
Setting up ca-certificates-java (20200101) ...
Setting up openjdk-9-jre-headless (9.2017.8.20-1
) ...
Setting up openjdk-9-jdk-headless (9.2017.8.20-1
) ...
$ █
```

これでOpenJDKとOpenJRE環境のインストールは終了しました。これらのインストールには、BlocklyCodeを実行する環境となるJVM(Java Virtual Machine)が含まれています。

今、我々はオンラインでファイルを作成するエディタをインストールしますが、我々はviと呼ばれるエディタを使用して次のコマンドを実行します。

```
apt install vim
```

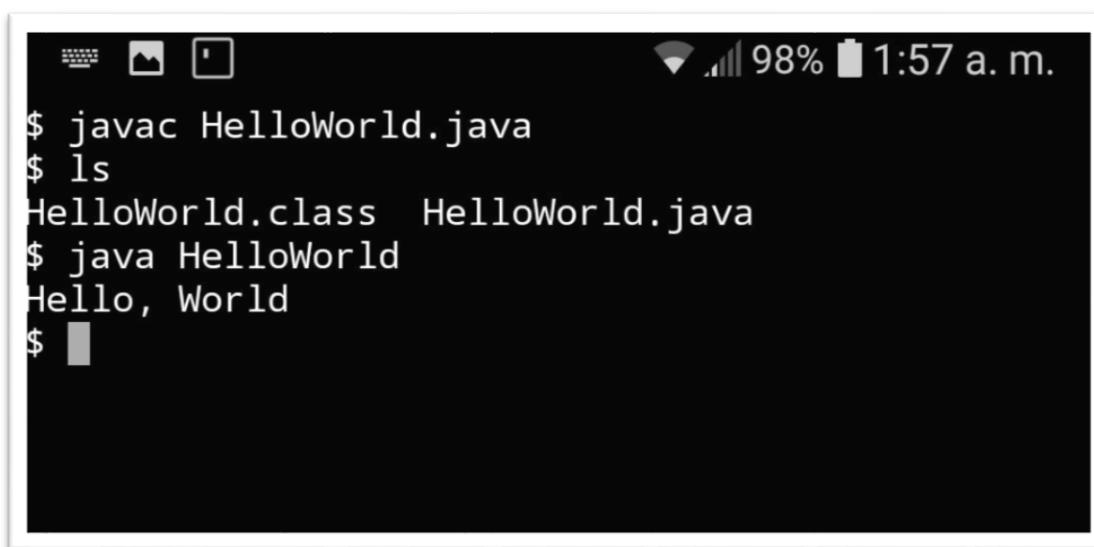
では、テストファイルをコンパイルして実行してみましょう。Termuxターミナルでviエディタで作成し、"Hello World"のjavaコードを書き、HelloWorld.javaファイルに保存します。

```
public class HelloWorld {  
    public static void main(String[] args) { . . .  
        // "Hello, World" をターミナルウインドウに出力します。  
        System.out.println("Hello, World");  
    }  
}
```

そして、Termuxターミナルで以下のコマンドを実行してコンパイルを行い、プログラムを実行します。

```
javac HelloWorld.java (実行後、バイトコードのHelloWorld.classファイルが作成されます)
```

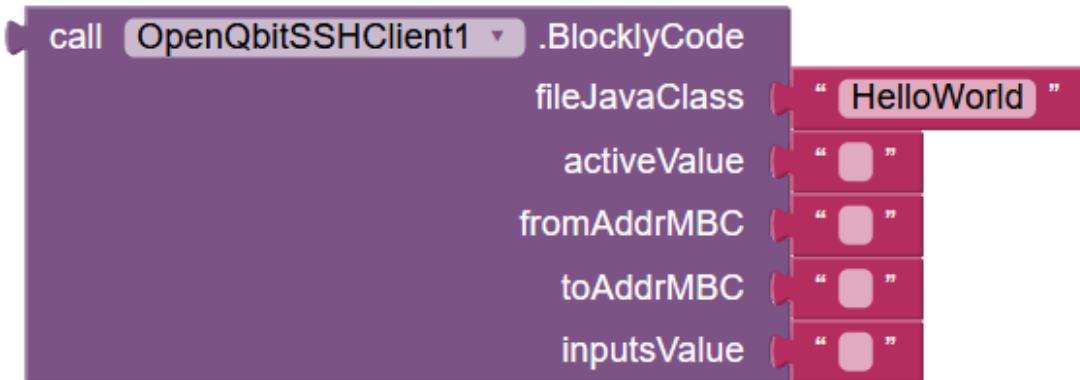
```
$ java HelloWorld (広告を実行して印刷した後、Hello World
```



The screenshot shows a Termux terminal window with a black background and white text. At the top, there are icons for signal strength, battery level (98%), and time (1:57 a.m.). The terminal prompt is '\$'. The user has run the following commands:

```
$ javac HelloWorld.java  
$ ls  
HelloWorld.class  HelloWorld.java  
$ java HelloWorld  
Hello, World  
$
```

ブロック(BlocklyCode)を使用して、このブロックは拡張子(ConnectorSSHClient)の中にあります。



前のブロックでは、HelloWorld.classファイルがどのように実行されるかを見ることができます、これはすでにコンパイルされ、ユーザベースのディレクトリに配置されています。

すでにjavaでコンパイルされたプログラムを実行可能なブロックで、開発環境ではバイトコードファイルのバイナリ形式で一般的に知られています。

このタイプのファイルは、Java仮想マシン（JVM）で実行する準備ができます。

バイトコードファイルを作成するには、2つの方法がありますまたは拡張子.CLASSで、ユーザーは快適さで彼のPCでそれを作成することができますまたは彼のようなも携帯電話で行うことができます、我々はすでに前にJDK（Java開発キット）をコンパイルする環境をインストールしていることを覚えて、キーボードの制限がカウントされますので、それはあまり効率的ではありませんが、また、Android環境を選択する場合には、ライブラリがインストールされているOpenJDKに限定されていることを考慮しなければならないでしょう。

入力パラメータは<inputsValue>で開放されており、その他の固定パラメータはactiveValue、fromaddrMBC、toAddrMBCです。

実行テストの場合、それらは内容がなくても空白のままにすることができ、バイトコードファイルだけがパラメータなしで実行されます。

前のブロックは、次のコマンドラインを実行しているようなものです。

\$ java HelloWorld

BlocklyCode用に開発されたプログラムは、それぞれの設計環境に応じて、"cron"エージェントを用いたルーチンで制御して実行し、Syncingmanagerと共有することができるようになります。

28. 付録「OpenQbit Quantum Computing」。

量子コンピューティングはどのように機能するのでしょうか？⁽²⁾

デジタルトランスフォーメーションは、これまでにないスピードで世界に変化をもたらしています。デジタル時代が終わろうとしていると思いますか？デジタルリテラシーは、社会的・経済的発展のギャップに対処するために、オープンな知識と技術を学ぶためのアクセス可能な機会が急務となっている分野としてすでに指摘されています。デジタル時代のキーコンセプトから学ぶことは、既存のモデルを驚異的なスピードとパワーで変革することができるもう一つの新しい技術の波、すなわち**量子技術の到来が間近に迫っているため、より一層重要になってきます。**

この記事では、従来のコンピューティングと量子コンピューティングの基本的な概念を比較し、他の関連分野への応用を探ります。

量子技術とは何か？

人類は歴史の中で、科学を通して自然の仕組みを理解し、技術を発展させてきました。1900年から1930年の間に、まだ十分に理解されていないいくつかの物理現象の研究は、新しい物理理論である**量子力学**を生み出しました。この理論は、分子や原子、電子などの自然界に生息するミクロの世界の働きを説明し、説明しています。この理論のおかげで、これらの現象を説明することができるようになっただけでなく、素粒子の現実が全く逆に不可解な、ほとんど魔法のような方法で動いていることや、ミクロの世界ではマクロの世界では起こらないような出来事が起こることを理解することができるようになりました。

これらの**量子特性**には、量子重ね合わせ、量子もつれ、量子テレポーテーションなどがあります。

- **量子重ね合わせ**は、粒子が同時に異なる状態になる方法を説明しています。
- **量子もつれ**は、2つの粒子がどのように相関しているかを説明しています。

- 量子テレポーテーションは、量子もつれを利用して、宇宙空間を移動することなく、ある場所から別の場所へ情報を送ることができます。

量子技術は、このような素粒子の量子的性質に基づいています。

この場合、今日では、量子力学でミクロの世界を理解することで、人々の生活を向上させる技術を発明したり、設計したりすることができるようになっています。量子現象を利用した技術は数多くあり、その中にはレーザーや磁気共鳴イメージング（MRI）のように半世紀以上も前からあるものもあります。しかし、現在、量子コンピューティング、量子情報、量子シミュレーション、量子光学、量子計測、量子時計、量子センサーなどの分野で技術革命が起きています。

量子コンピューティングとは？まず、古典的なコンピューティングを理解する必要があります。

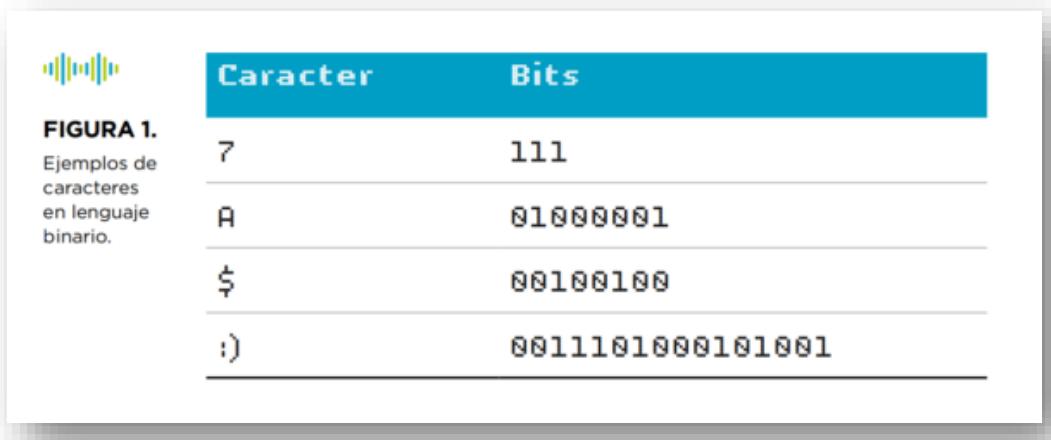


FIGURA 1. Ejemplos de caracteres en lenguaje binario.

Carácter	Bits
?	111
A	01000001
\$	00100100
:)	0011101000101001

量子コンピュータの仕組みを理解するためには、まず、私たちが日常的に使用しているコンピュータ（ここではデジタルコンピュータやクラシックコンピュータと呼ぶ）がどのように動くのかを説明するのが便利です。これらは、タブレットや携帯電話などの他の電子機器と同様に、メモリの基本単位としてビットを使用しています。これは、プログラムやアプリケーションがビット、つまり0と1のバイナリ言語で符号化されていることを意味します。私たちがこれらのデバイスのいずれかと対話するたびに、例えばキーボードのキーを押すことで、0と1の文字列が作成され、破壊され、コンピュータ内部で変更されます。

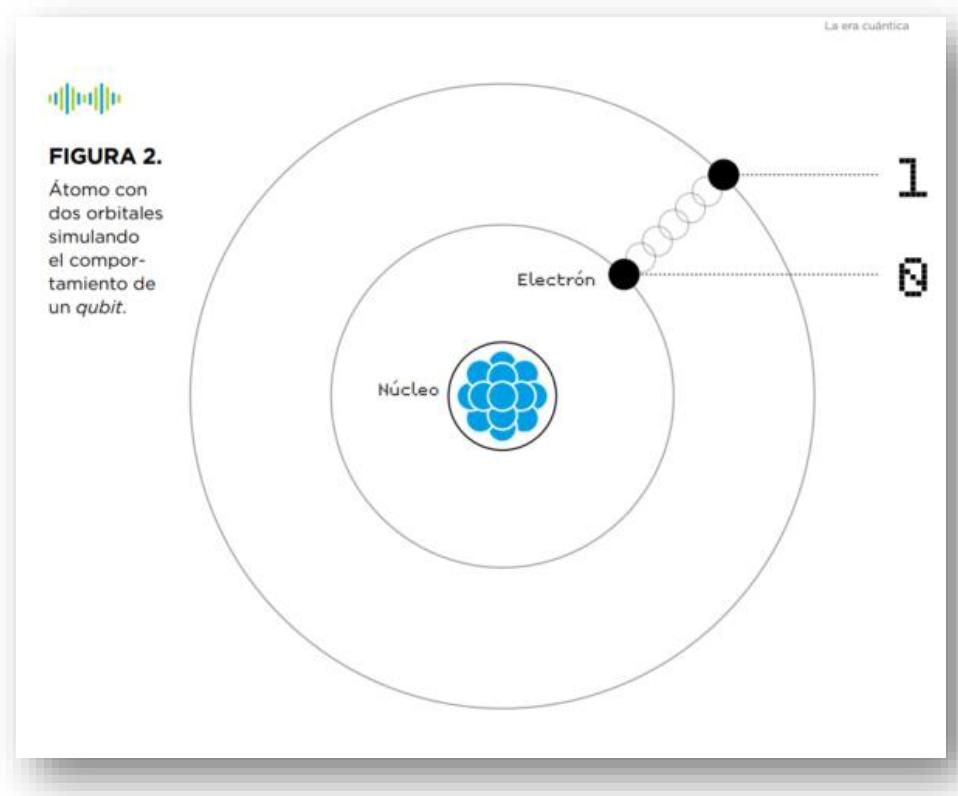
興味深いのは、この0と1が物理的にコンピュータの中にあるのは何なのかということです。ゼロ状態と1状態は、スイッチの役割を果たすトランジスタと呼ばれる微細な部品を介して循環する電

流の有無に対応しています。電流が流れていなければトランジスタは"オフ"でビット0に対応し、電流が流れているときは"オン"でビット1に対応します。

より簡単に言えば、ビット0とビット1が正孔に対応しているようなもので、空の正孔がビット0、電子が占有している正孔がビット1となります。このため、これらの装置はエレクトロニクスと呼ばれており、一例として、図1は、ある文字の二進法での書き込みを示している。今日のコンピュータの仕組みがわかったところで、量子の仕組みを理解してみましょう。

ビットからクビットへ

量子コンピューティングにおける情報の基本的な単位は、量子ビットまたは量子ピットです。キューピットは、定義上、2レベルの量子系であり、ビットと同様に、低レベルの励起またはエネルギーが0と定義された状態に対応する低レベル、または高レベルの励起または1と定義された状態に対応する高レベルの量子系することができます。しかし、ここに古典的なコンピューティングとの根本的な違いがありますが、クビットは0と1の間の無限の中間状態、例えば0と1の半分、または0の4分の3と1の4分の1の状態のような状態のいずれかにもなります。



量子アルゴリズム、指数関数的に強力で効率的なコンピューティング

量子コンピュータの目的は、量子システムとしての量子ビットのこれらの量子的性質を利用して、オーバーラップとインターリープを利用した量子アルゴリズムを実行することで、古典よりもはるかに大きな処理能力を提供することにあります。重要なのは、パラダイムの真の変化は、デジタルコンピュータや古典的なコンピュータが現在行っていることと同じことを行うことではなく、多くの記事で紹介されているように、量子アルゴリズムによって、ある操作を全く異なる方法で実行することができ、多くの場合、より効率的であることが判明します。

具体的な例を見てみましょう。私たちはボゴタにいて、そこに行くための100万の選択肢の中からリマに行くための最良のルートを知りたいと想像してみましょう ($N=1,000,000,000$)。コンピュータを使って最適なルートを見つけるためには、100万個のオプションをデジタル化する必要がありますが、これは古典的なコンピュータではビット言語に、量子コンピュータでは量子ビットに翻訳することを意味します。従来のコンピュータでは、目的のパスを見つけるまですべてのパスを1つずつ解析する必要がありました。しかし、量子コンピュータでは、量子並列性として知られるプロセスを利用して、一度にすべてのパスを考慮することができます。このことは、古典的なコンピュータが $N/2$ ステップまたは反復の順序、つまり50万回の試行を必要とするのに対し、量子コンピュータはレジストリに対する \sqrt{N} 回の操作、つまり1,000回の試行の後に最適なパスを見つけることを意味しています。

前のケースでは2次関数的な利点がありますが、他のケースでは指数関数的な利点もあり、 n 個のクビットで 2^n ビットに相当する計算能力が得られます。これを例示するために、約270量子ビットの量子コンピュータでは、宇宙の原子の数（約280個と推定される）よりも多くの基底状態を持つことができます。もう一つの例として、2000から2500量子ビットの量子コンピュータでは、現在使用されている暗号（いわゆる公開鍵暗号）を実質的にすべて破ることができると推定されています。

なぜ量子技術を知ることが重要なのか？

私たちは、ブロックチェーン、人工知能、ローン、モノのインターネット、バーチャルリアリティ、5G、3Dプリンター、ロボット、自律走行車などのさまざまな新興技術が、複数の分野やセクターでますます存在感を増しているデジタルトランスフォーメーションの瞬間にいます。これらの技術は、人間

の生活の質を向上させ、開発を加速させ、社会的インパクトを生み出すと呼ばれ、現代では並行して進歩しています。ブロックチェーンとIoT、ドローンと人工知能など、これらの技術を2つ以上組み合わせた製品を開発している企業を目にすることはほとんどありません。しかし、開発の初期段階では、開発者や技術的なプロファイルを持った人材が不足しているため、収束はまだ保留されているのが現状です。

その破壊的な可能性から、量子技術は、これらすべての新技術に収束するだけでなく、事実上すべての新技術に横断的な影響を及ぼすことが期待されています。量子コンピューティングは、認証、交換、データの安全な保存を脅かすことになり、サイバーセキュリティやブロックチェーンなど、暗号がより関連性の高い役割を持つ技術には大きな影響を与え、マイナーなネガティブな影響はあるが、5G、IoT、ドローンなどの技術にも考慮される。

量子コンピューティングを実践してみませんか？

C, C++, Java, Matlab, Maxima, Python, Octaveなどの様々なプログラミング言語を使用した数十種類の量子コンピュータシミュレータがすでにネット上で利用可能です。また、マイクロソフトが立ち上げたQ#のような新しい言語も。IBMやリゲッティなどのプラットフォームを介して、仮想量子マシンを探索したり、遊んだりすることができます。

Mini BlocklyChainは、OpenQbit.comの会社によって作成されたもので、さまざまなタイプの民間および公共部門のための量子コンピューティング技術の開発に焦点を当てています。

なぜMini BlocklyChainが他のブロックチェーンと違うのかというと、単純にシステムがモジュール化され、量子コンピューティングを含むように作られたからです。

- (2) <https://blogs.iadb.org/conocimiento-abierto/es/como-funciona-la-computacion-cuantica/>

29.付録「SQLiteデータベースの拡張ブロック

拡張機能OpenQbitSQLiteを構成する各ブロックの適切かつタイムリーな使用の詳細を確認するには、次のリンクで拡張機能の元の作者を確認してください。

OpenQbitSQLiteは、オリジナルデザインのクローンで、AESセキュリティレベルで使用するための小さな修正を加えています。

<https://github.com/frdfsnlght/aix-SQLite>

30. 別館「ミニBlocklyChainシステムの作成例

Mini BlocklyChainシステムの機能性やメリット、作りやすさなどを検証できるテストシステムを作ります。

我々は、設計と情報は、我々はすでにブロックのチェーンとして定義されているものが存在するストレージの開発を開始します。設計はSQLiteデータベースをベースにしており、各組織や設計に応じて、Microsoft SQL Server、MySQL、Maria DB、Oracle、DB2、PostgreSQLなどの他のデータベースに簡単に変更することができます。再びトランザクションとSQLiteデータベースへの同意のプロセスに起因するが、任意のレベルで、ビジネスや個人的なアプリケーションのために使用することができます。

この中でminibcと呼ばれるデータベース作成は、ブロック文字列を格納するテーブルを持つことになります。このデータベースは、ノードにインストールされる最終的なプログラムコードに埋め込まれますが、このストレージ場所は、"パッケージ化された資産"と呼ばれるApp InventorなどのBlockly環境の内部種です。

\$ sqlite3 minibc.db

SQLiteバージョン3.32.2 2020-06-20 15:25:24

使用方法のヒントは「.help」を入力してください。

トランジエントなインメモリデータベースに接続されています。

.open FILENAME"を使用して、永続的なデータベースで再開します。

sqlite> .quit

nblockテーブルのフィールドを設計します。

CREATE TABLE nblock (

id 整数主キー AUTOINCREMENT

```
, prevhashVARCHAR NOT NULL  
, newhashVARCHAR NOT NULL  
ntransINTEGER NOT NULL  
nonceINTEGER NOT NULL  
  
q r n g INTEGER NOT NULL  
);
```

nblockテーブルを作成しました。

```
$ sqlite3 minibc.db
```

```
SQLiteバージョン3.32.2 2020-06-20 15:25:24
```

使用方法のヒントは「.help」を入力してください。

トランジエントなインメモリデータベースに接続されています。

.open FILENAME"を使用して、永続的なデータベースで再開します。

```
sqlite> .open minibc.db
```

```
sqlite> CREATE TABLE nblock ( id integer primary key AUTOINCREMENT , prevhash VARCHAR  
NOT NULL , newhash VARCHAR NOT NULL , ntrans INTEGER NOT NULL ,nonce INTEGER  
NOT NULL ,qrng INTEGER NOT NULL ).
```

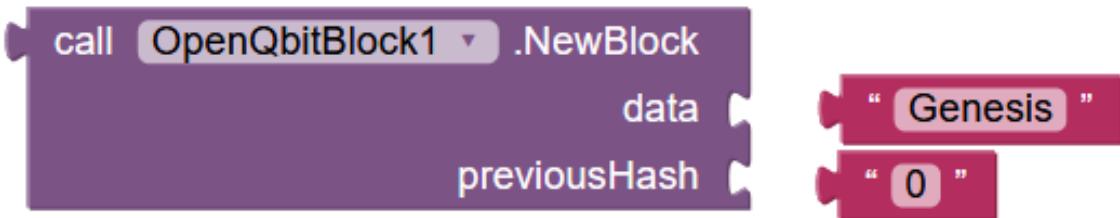
似たようなものが出でてきます。

```
$ sqlite3 minibc.db
SQLite version 3.32.2 2020-06-04 12:58:43
Enter ".help" for usage hints.
sqlite> CREATE TABLE nblock (
...>     id integer primary key AUTOINCREMENT
...>     , prevhash VARCHAR NOT NULL
...>     , newhash VARCHAR NOT NULL
...>     , ntrans INTEGER NOT NULL
...>     , nonce INTEGER NOT NULL
...>     , qrng INTEGER NOT NULL);
sqlite> .tables
nblock
sqlite> .quit
$
```

nblockテーブルを作成するためのSQL文です。

次に「Genesis」と呼ばれるシステムの初期ハッシュを作成しますが、これは我々が使用するブロック（NewBlock）のブロックチェーンの最初のブロックの新しいハッシュを作成することを基本としています。次に、"Genesis"ハッシュを元に"one"と呼ぶ第二のハッシュを作成します。これは、最初のブロックチェーンのハッシュ検証テストでは常に次のように準拠しなければならないため、第一のハッシュと整合性が取れていなければならないからです。

NewBlockです。Genesis"ハッシュを作成するために、入力パラメータを使用します。



入力パラメータ : data <String>, previousHash <String>

出力パラメータ : SHA256ハッシュを1つ。

この場合、初期値"0"に等しい"previousHash"を使用し、入力データの場合は"data"を"Genensis"という単語にします。

これにより、両方のデータの組み合わせの計算されたハッシュが得られます。

SHA256 (Genesis0) = eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfbb09ba54e802e642

上記の文字列はnblockテーブルに挿入されますが、この文字列は暗号化されていなければなりません。



我々は、minibc.dbデータベースにデータを挿入するためにOpenQbitSSHClient拡張機能を使用します、それはnblockテーブルのminibc.dbデータベースへのINSERT文になります。

```
sqlite3 keystore.db "insert into nblock (prevhash, newhash, ntrans, nonce, qrng) values ('0',
'eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfbb09ba54e802e642',
'1', '0', '0');
```

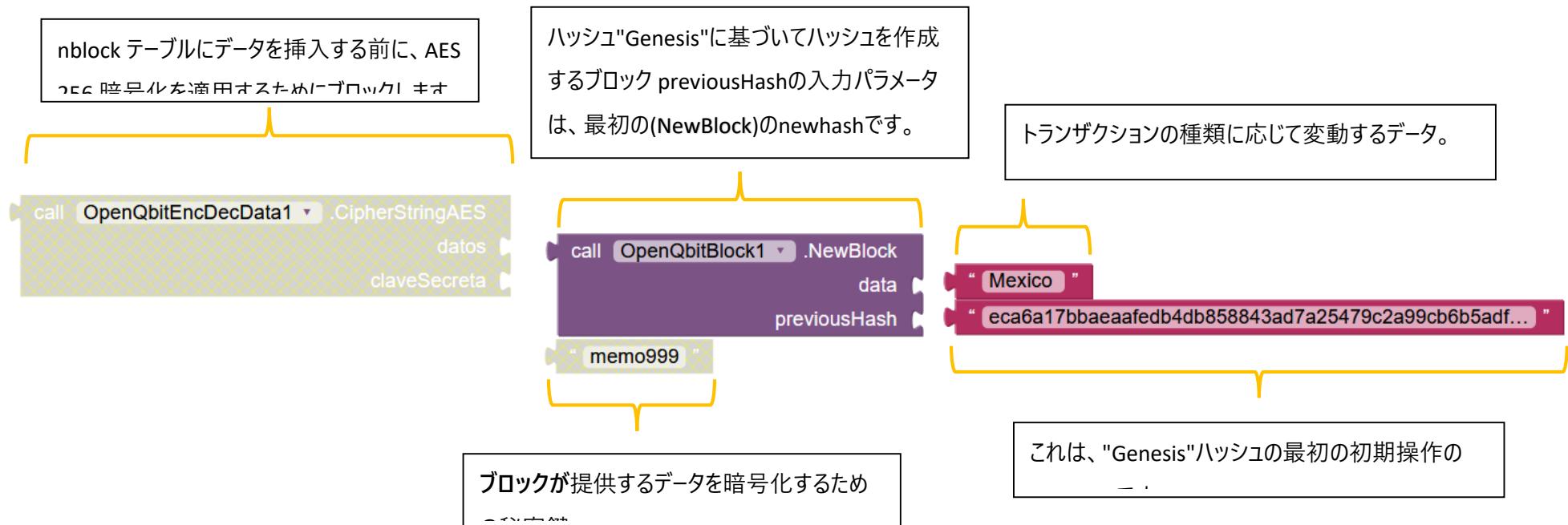
「Genesis」のハッシュを元にSQL文で「1」のハッシュを作成しました。

```
sqlite3 keystore.db "insert into nblock (prevhash, newhash, ntrans, nonce, qrng) values ('  
eca6a17baeaea, newhash, ntrans, nonce, qrng)db "insert into nblock (prevhash, newhash,  
ntrans, nonce, qrng) values ('  
eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfb09ba54e802e642'!  
f6a6b509376deebc8a5d70da9d0436943b76c3933f35c419ed48e78abab59a68','1','0','0')o "
```

ハッシュ"one"のnewhashは、次のように計算されます。

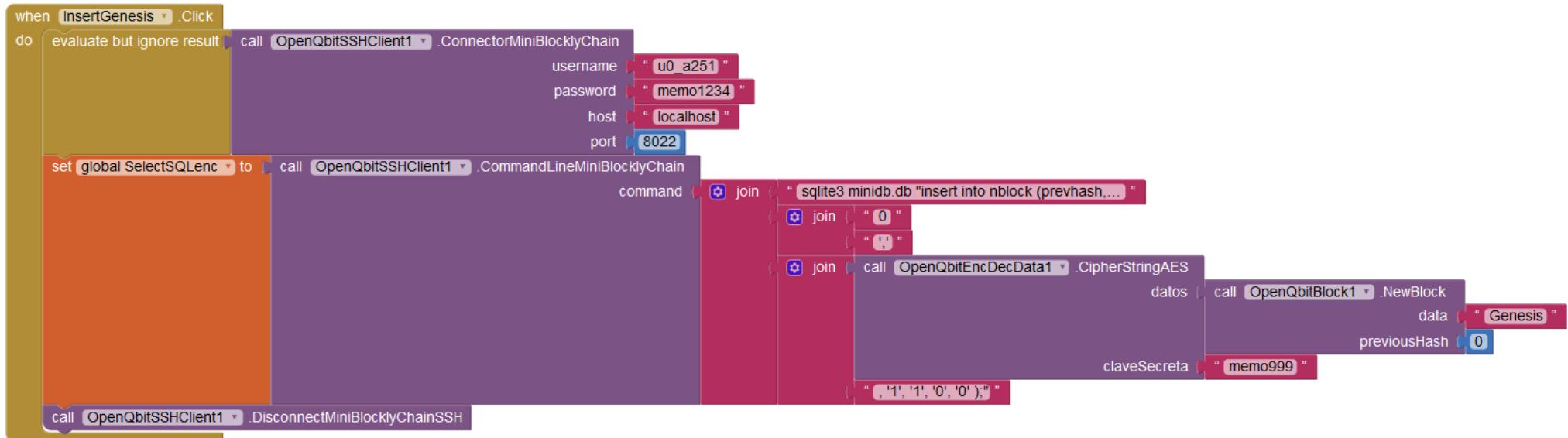
SHA256 (eca6a17bbaeaafedb4db858843ad7a25479c2a99cb6b5adfbb09ba54e802e642メキシコ) =
f6a6b509376deebc8a5d70da9d0436943b76c3933f35c419ed48e78ab59a68

最初の初期化ハッシュ"Genesis"に基づいた2番目のハッシュは、ブロックチェーンの初期検証はフィールドの平等性を遵守しなければならないため、最初に2つのINSERTSを行う必要があります： prevhash (最後のデータ) = newhash (最後のデータ)。



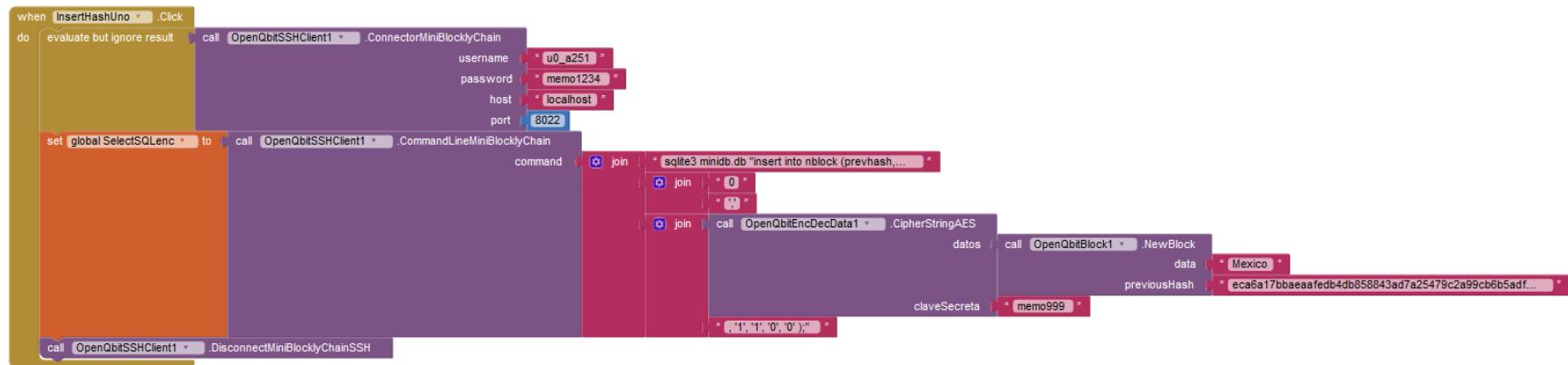
引き続き、上記のハッシュである"Genesis"ハッシュと"start"ハッシュのApp Inventor内での実行を作成していきます。

ここでは、"Genesis"ハッシュのINSERTが、App Inventorシステムのnblockテーブル内のブロックチェーンの初期構造にどのように統合されるかを示します。



"Genesis"ハッシュに基づいてnblockテーブルに1つ目のデータを挿入したので、"one"ハッシュを参照して2つ目のINSERTを行うように進めます。

これにより、App Inventor システムの nblock テーブル内のブロック チェーンの初期構造に "One" ハッシュの INSERT が統合されます。



重要な点は、minibc.dbデータベースとその内部構造(テーブル)は、"Peer to Peer"アーキテクチャに基づいてMini BlocklyChainネットワークを介してレプリケートされるということです。

これまでのところ、ブロックチェーンストレージの基本的かつ基本的な構造は完成しており、システム内で発生したり要求されたりした将来のトランザクションから新しいブロックを受け取る準備ができています。

すでに最初のハッシュデータ"Genesis"とハッシュ"one"をminibc.dbデータベースに挿入しましたが、次のSQLステートメントを使用してnblockテーブルのprevhashとnewhashフィールドにクエリを実行します。

この2つのフィールドを比較することで、ブロックチェーンが一貫しているかどうか、トランザクションの完全性とセキュリティを維持しているかどうかを知ることができますので、この点は基本的なことになります。これは、ブロックチェーンを見直すための出発点に過ぎません。将来的には、ブロックの使用方法を見直してメルクリツリーを計算し、ブロックチェーンの完全性とセキュリティを確保するためのもう一つの重要なツールとなるでしょう。

今のところは、以前に拡張機能 (OpenQbitSSHClient) を使用していたように、使用しなければならない構造またはSQL文のみをレビューします。後で、新しいブロックを追加するたびに、そのチェックのために等しいデータの単純比較を行うことができます。

prevhashの場合。

```
sqlite3 minibc.db "SELECT prevhash FROM nblock ORDER BY id DESC LIMIT 1;"
```

ニューハッシュのために。

```
sqlite3 minibc.db "SELECT newhash FROM nblock ORDER BY id DESC LIMIT 1;"
```

ここからは、新規取引および／または取引キューに挿入する取引の依頼をどのように行うかに焦点を当てていきます。

トランザクションをトランザクションキューに送信するための要件として、2つのステップがあります。

第一の要件は、当社の口座残高が、送付する金額をカバーできるだけの十分な「資産」を有していることです。資産」は数字や金額だけではなく、作成するそれぞれのシステムに応じて、どんな種類の「資産」でも作成できることを覚えておきましょう。

資産の例。

- ✓ 仮想通貨または暗号通貨」の新しい起源の本質的な量。
- ✓ デジタルデータを参照したデータ（全種類の文書

- ✓ 文字列またはすべてのタイプのファイルのためのハッシュ認証署名。
- ✓ デジタル情報またはその表現の取引または転送。

各ノードの「資産」の残高は、ブロック(GetBalance)を使って取得します。



第二の要件は、トランザクションを送信する際に最低限のパラメータを提供することであり、それは

- ✓ ソースアドレス。- は資産の発送元の住所です。
- ✓宛先アドレス。- は、送信された資産を受け取るユーザーのアドレスです。
- ✓ 活動的に。- トランザクションごとに送信するシステムごとに与えられる固有値を持つデータです。

上記アドレス（発信元-発信先）は、各ユーザーのアカウント作成時の公開鍵に対応しています。ユーザー アカウントを作成する際には、公開鍵と秘密鍵の2種類が作成されることを覚えておきましょう。

公開鍵は、システム全体で共有されるアドレスであり、システムの誰もが閲覧し、トランザクションの送受信に使用することができます。

秘密鍵は各ノードでローカルに使用されるアドレスで、その名前が示すように、送信されたトランザクションにセキュリティを与えるためのデジタル署名を作成するのに役立つプライベートな使用のためのものです。

前述のパラメータを使用するために、keystore.dbに格納されている"秘密鍵"アドレスがシステム内で共有されることなく各ノードでローカルに使用される2つのリポジトリと、publickeys.dbに格納されている"公開鍵"アドレスがシステムの全ノードで"Peer to Peer"プロトコルを介して共有されるもう1つのリポジトリに頼ることにします。

データベース"keystore.db"と"publickeys.db"は、別館「KeyStore & PublicKeysデータベースの作成」で作成済みです。

トランザクションキューのためのデータベースとシステムは、「RESTful Mini Sentinelネットワークのインストールと設定」のセクションすでに作成されています。op.sqlite3と呼ばれるトランザクションのためのデータベースを既に作成していますが、この中には2つのテーブルがあります。

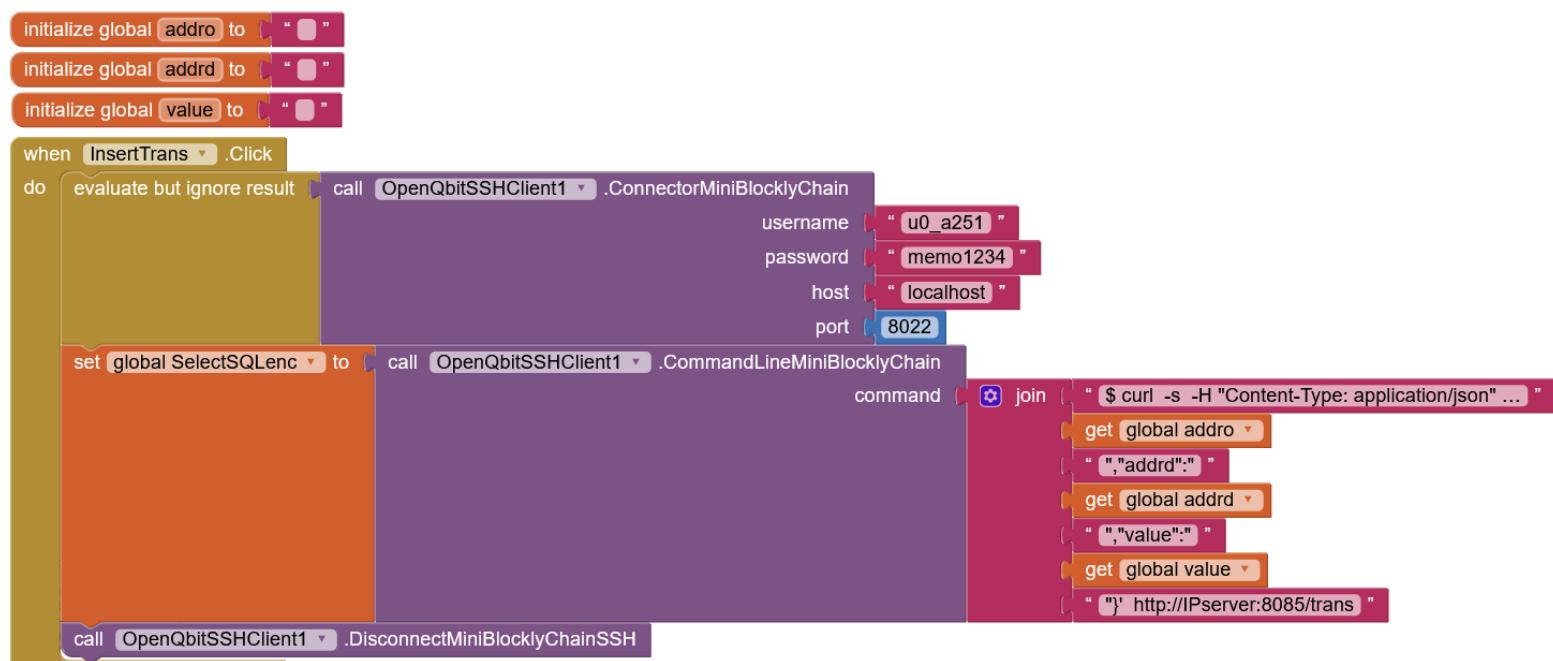
しかし、同じデータベースop.sqlite3を"Peer to Peer"モデルで変更して使用するためには、同期システムで共有モデルのデータベースを使用する必要があります。op.sqliteデータベースにRESTfulを適用したトランザクションキューへの操作の送信を開始します。

私たちは、"trans"テーブルに新しいトランザクションを作成しました。

```
$ curl -s -H "Content-Type: application/json" -d
'{"addro":"MBCNqkUDgQ6t48WhyoLTf5XxypgfzLUQ7Kj9",
"addrd":"MBCTDY1F2FvRjKCEabwTUa3EaE8BiM1Qqsh", "value"."999"}'
http://IPserver:8085/trans
```

```
# 出力
{
"id": 1,
"addro": "MBCNqkUDgQ6t48WhyoLTf5XxypgfzLUQ7Kj9",
"addrd": "MBCTDY1F2FvRjKCEabwTUa3EaE8BiM1Qqsh",
"値": "999"
}
```

App Inventorでの実装。



入力パラメータ: addro, addrd, valueは、アルゴリズムに入力できる変数であり、keystore.dbデータベースから抽出されます。

付録「KeyStoreデータベースの作成」を参照してください。

これで前回の取引のデジタル署名を挿入しました。表の中の"記号"

```
$ curl -s -H "Content-Type: application/json" -d '{curl -s -H "Content-Type: application/json" -d '  
"trans_id":1  
"sign": "59ahGVn8pZ4oduxtTWCCvRiqjWEEy1c62".  
"hash": "f6a6b509376deebc8a5d70da9d043694343b76c3933f35c419ed48e78abab59a68"}'  
http://IPserver:8085/sign
```

```
# 出力  
{  
"id": 1,  
"trans_id": 1,  
"sign": "59ahGVn8pZ4oduxtTWCCvRiqjWEEy1c62".  
"ノック":  
"f6a6b509376deebc8a5d70da9d043694343b76c3933f35c419ed48e78ab59a68"  
}
```

注: デジタル署名(sign)は、curl コマンドステートメントを送信する前に取得する必要があり、ブロック(GenerateSignature)で生成されます。

入力パラメータ : Trans_id, sign, hashはアルゴリズムに入力できる変数であり、ブロック(GenerateSignature)でトランザクションのデジタル署名が生成されます。

ここからは、取引が行われるたびに適用される電子署名を生成する手順を見ていきます。

トランザクションのための電子署名の生成ブロック(GenerateSignature)を使用するとき、結果としてファイルタイプ.sigがあります。このファイルは、テーブル"sign"のsignフィールドに保存されるのに使用します。

file.sigのようなバイナリデータを扱うには、それをbase64に変換して、"sign"テーブルのsign変数に格納しなければなりません。これを行うには、ブロック(EncoderFileBase64)を使用します。

各トランザクションの「符号」テーブルのハッシュフィールドの値がどのように計算されるか。

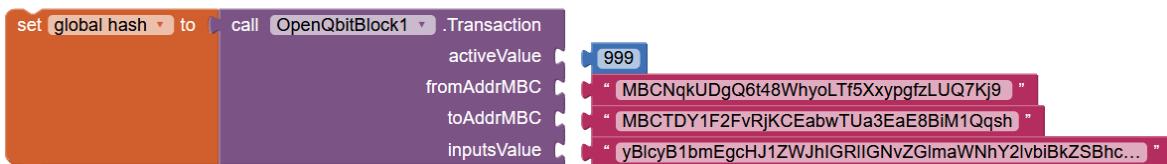
トランザクションハッシュ = SHA256(送信元アドレス + 送信先アドレス + 資産 + fileBase64.sig)

ハッシュ型SHA256の例。

SHA256(MBCNqkUDgQ6t48WhyoLTf5XxypgfzLUQ7Kj9 + MBCTDY1F2FvRjKCEabwTUa3EaE8BiM1Qqsh + 999+).

出力 : 9d45198faaef624f2e7d1897dd9b3cede6ecc7fbac516ed1756b350fe1d56b4

ハッシュの計算のためには、Block(**Transaction**)に傾く必要があります。



出力パラメータは、システム内の任意のノードから送信される各トランザクションに対して保存されるハッシュです。これは、ベースのop.sqliteの"sign"テーブルのsignフィールドに保存されます。

前の操作では、ユニークで再現性のないハッシュのみがキューに送信された各トランザクションを表し、この代表的なハッシュが送信された情報の全体であることを確認しました。

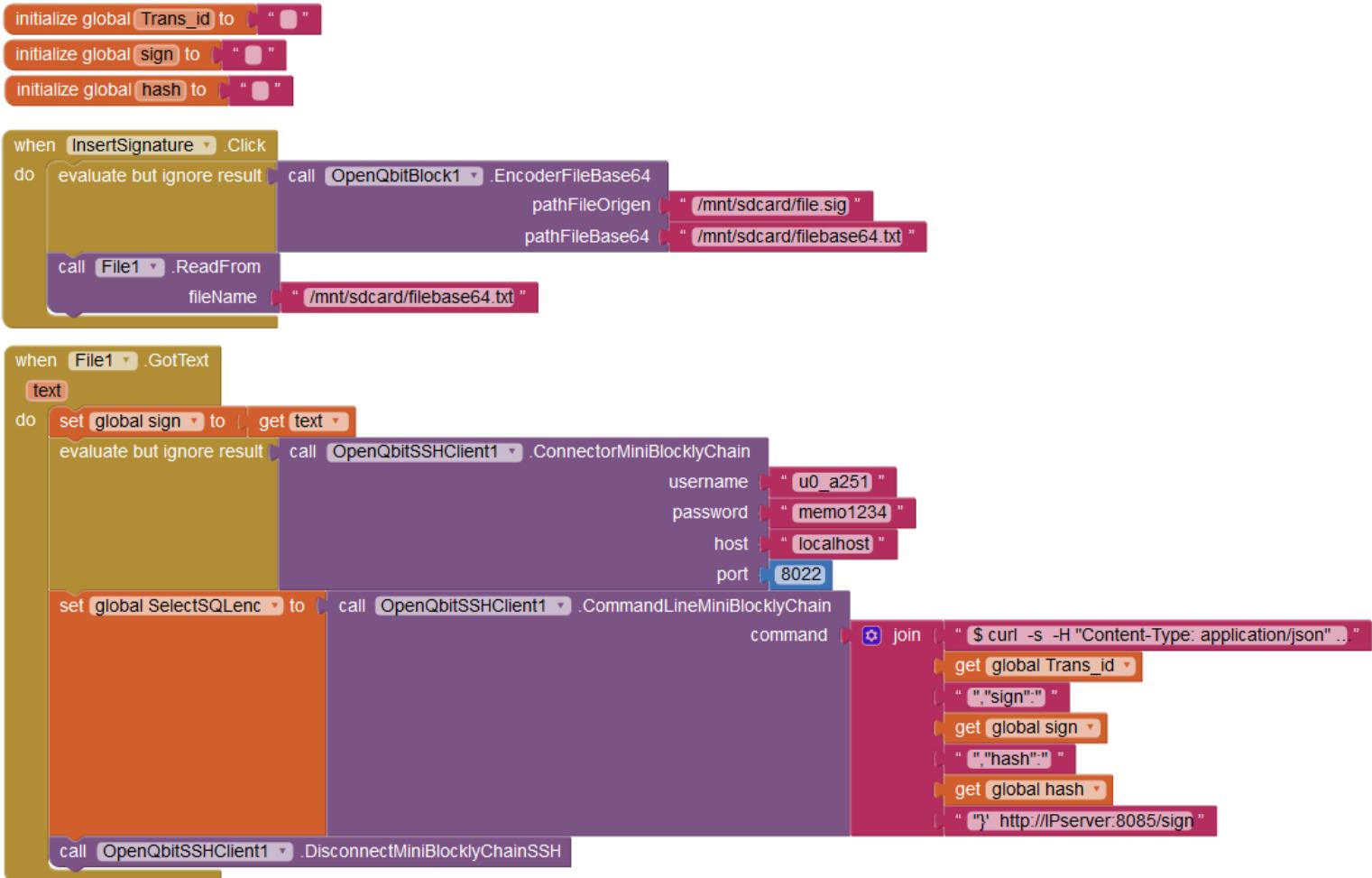
唯一欠けているのは、各トランザクションでどのノードが送信されるかを区別するための識別子であるTrans_id変数を含めることです。この例では、Trans_id変数に固定値"1"を設定します。なぜなら、それはネットワークで設定されている最初のノードだからです。この値は、それぞれの特定のデザインに応じて構文を変えることができるので、簡単のためにシンプルな識別子を選択しました。

すべての変数が定義されているので、App Inventorでブロック実行の構造とシーケンスを作成して、テーブル「sign」にINSERTを実行します。

注意: 各トランザクションの送信はop.sqlite3データベース内の2つのINSERTSで構成されていることを考慮に入れることが重要です。

op.sqlite3データベースの設計では、将来的には"sign"テーブルのみを暗号化することが可能であるという事実のため、2つの別々のテーブルを作成しました。

テーブル「sign」のINSERTのApp Inventor内にブロックとメソッドの実行構造を作成します。



これらはop.sqlite3データベース内にあり、これら2つのテーブルに挿入されたデータは、処理のためにすべてのノードに送信されるトランザクションキューを作成するために使用されます。

今回は、Java SQLite-Redisコネクタを使用します。このコネクタは、op.sqlite3データベースからRedisデータベースへのデータ変換を実行します。付録「Java SQLite-Redis コードコネクタ」を参照してください。

SQLite-Redis Connectorを実装した後、トランザクションキューはRedisシステムを介してシステムのすべてのノードに配信されます。

トランザクションキーを適切な方法で受信して処理できるようにするにはどうすればいいのか、というところから始めます。

トランザクション キューは、Redis サービスを介して配信されます。これは、App Inventor 環境でそれぞれの制御ブロックを持つリアルタイム データベースです。

App InventorのBlocklyシステム内でのRedis環境の構成。

注意点としては、本書執筆時点までのRedisサーバを制御するためのブロックは、App Inventorシステムでしか利用できないということです。App Inventor とは異なる Blockly システムを使用する場合は、拡張機能 (OpenQbitSSHClient) を介して Termux 端末にコマンドを送信することで、Redis CLI (コマンドライン) を使用して実行する必要があります。

Redis CLI (コマンドライン) での使用例。

Redisですべてのラベルやキーを取得するには

```
$ redis-cli KEYS '*'
```

キーから値を取得すること。

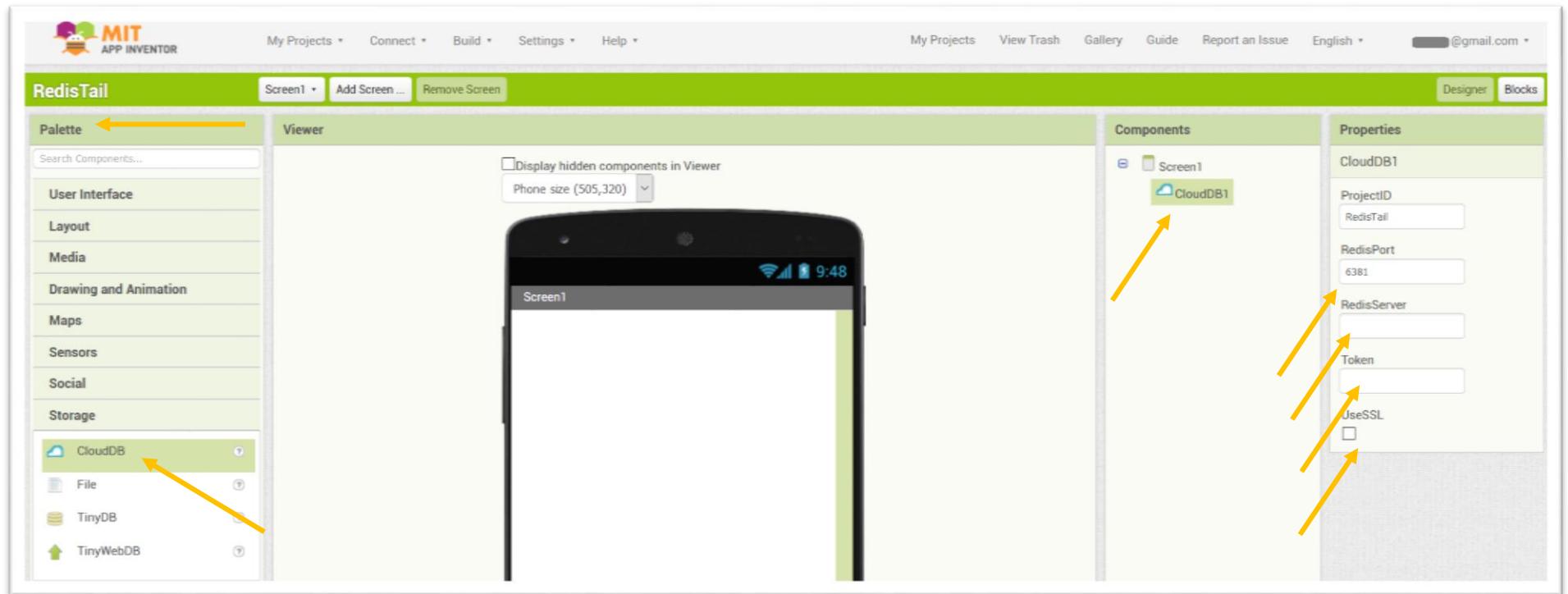
```
$ redis-cli GET <key>
```

今回はApp inventorを使用しているので、Redisで動作するためのブロックの使用方法をレビューします。

App Inventorの内部では、我々はRedisで使用するためにいくつかのブロックを持っている、我々は我々がする新しいプロジェクトを作成するために開始しました。マイプロジェクト > 新しいプロジェクトの開始



プロジェクトを作成した後、左上に移動し、"パレット"セクションで"ストレージ"をクリックし、オブジェクト"CloudDB"をドラッグして、Redisサーバーへの接続を設定するためのすべての機能を統合します。



設定は非常に簡単で、ノード(携帯電話)で実行されているRedisサーバ(Local)に接続できるように、次のパラメータを与えるだけです。

ProjectID.- これはRedisのトランザクションキューを識別するラベルを開始するIDです。

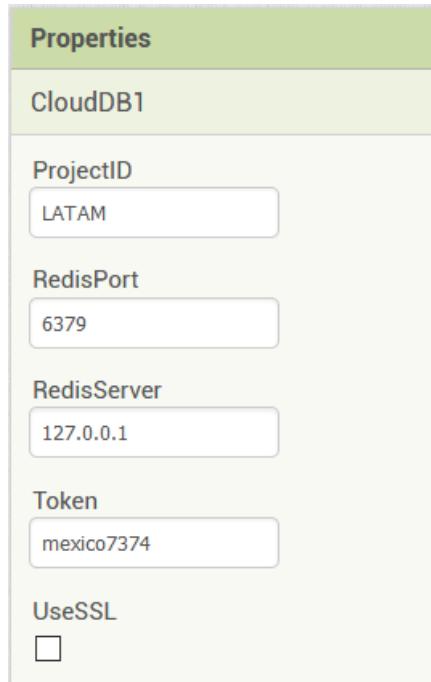
RedisPortを使用しています。- これは、デフォルトで接続するRedisサーバーのポートは6379です。

RedisServerです。- これは、私たちの場合、ノードのドメインまたはローカルIPであり、すべてのノードのそれは127.0.0.1になります。

トークン。- これは、接続を有効にしたRedisサーバーのパスワードです。

UseSSLを使用します。- このオプションはSSL証明書を使用する機会を与えてくれますが、私たちの場合は無効にしておきます。

我々のモデルとシステム設計では、すべてのネットワークノードで以下のパラメータを持つことになります。



Redisデータベース環境での制御とデータ処理を提供してくれるブロックを見直す時が来ました。

メソッドブロック(DataChanged)から始めます。



このメソッドは、設定したRedisサーバーに変更があるたびに2つの値を与えてくれます。

タグを使用しています。- この値は、トランザクションキューを識別するタグです。

の値を設定しています。- この値には、処理のために送信されたすべてのトランザクションのリストが含まれます。この値は、<String>型の文字列リストです。

の場合は、以下のように情報処理を開始するところです。

すべてのハッシュ値トランザクションのチェーンを提供してくれるので、このチェーンが有効であるかどうかを確認し、そのチェーンが発信元から変更されていないかどうかを確認することから始めます。大量の情報を検証するために、非常に有用なアルゴリズムを使ってこれを行います。

merkleの木のアルゴリズムを使います。このアルゴリズムを適用することで、私たちが受け取る情報（トランザクションキュー）の整合性のセキュリティを確保することができます。

以下の例では、Redisでトランザクションキューを作成し、Termux端末からRedis CLI(コマンドライン)を実行して"LATAM:HASH"キーを確認していますが、このキーを参照するにはSQLite-Redis Connectorを既に実行している必要があります。

```
C:memor>redis-cli
127.0.0.1:6379> get LATAM:HASH
"9d45198faaef624f2e7d1897dd9b3cde6eccca7fbac516ed1756b350fe1d56b4,"
f71c801a5fd25fc303ebc8c616204b4877ffb93006ec6a88bc30acf43ec250f5。
"60f8a3bcac1ea7d38e86efbc3e3e00480807d23f980391000766e804ed14ecb2,
8a6df1d38c22e0f9212052efa6136da3edf1fb1b2a3e25224ac3d689124b754]
127.0.0.1:6379>
```

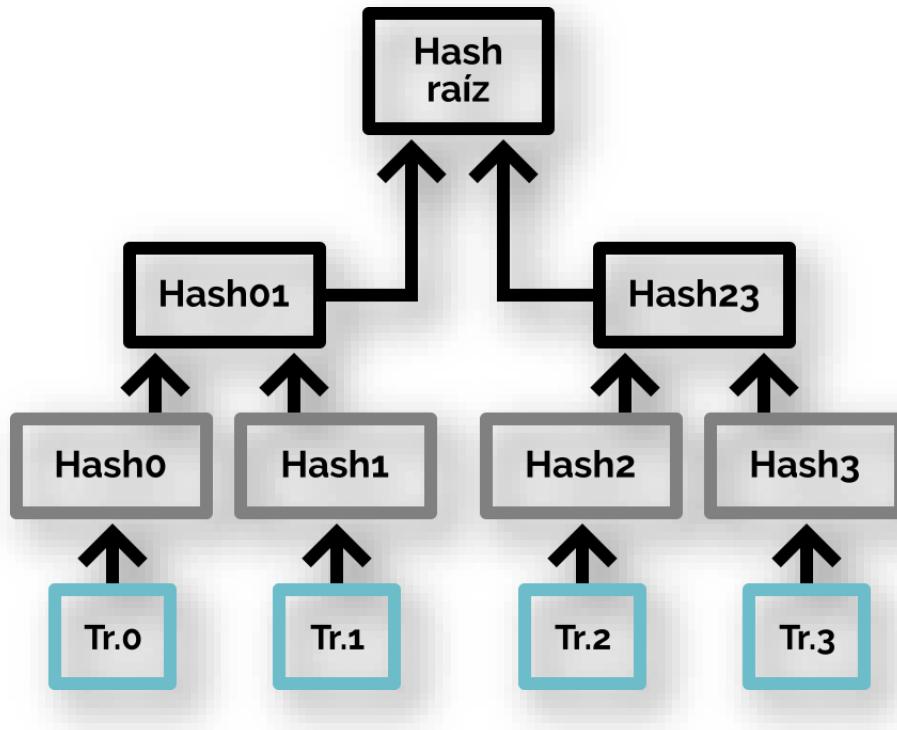
前のトランザクションキューは、それを構成する個々のトランザクションを表す4つのハッシュを見ているので、3つの要素しか持っていません。

あとはメルクリの木の仕組みを理解しましょう。

ハッシュメルクリツリーとは、2値または非2値のデータツリー構造で、シートではない各ノードには、その子ノードのラベルまたは値の連結のハッシュがタグ付けされています。これらはハッシュリストとハッシュ文字列の一般化です。

私たちの場合、4つの要素のメルクリ木を計算するために以下の計算を行います。これは、連結されたデータのペアを取り、それぞれのハッシュを取得する結果を計算することによって行われます。

このプロセスを説明した次の図を見てみましょう。



ハッシュベースのトランザクションキーは、ブロック (GetMerkleRoot) を介して引き出されます。



ハッシュルート:

51431822de7c94b90dc06d47b8f6275f315a4976c8479d30c32747fa90325432

その結果をローカルRedisシステムのLATAM:merklerootキーと比較し、データの整合性を確認するために両方のキーが一致しなければなりません。

メルクルツリーによって提供されるもう一つのセキュリティポイントは、特定のトランザクションがその起点からトランザクションキーに含まれていることを確認し、そのトランザクションが外部または内部の通信手段によって不正な方法で導入されていないことを確認することである。

鎖の和が奇数要素の場合は、最後の要素が重複してアルゴリズムの実行を開始するための配置を持っています。

もう一つの重要なポイントは、各トランザクションがその発信地-発信地に対応し、資産が発信地アドレスによって送信されたものであることを検証するための時間です。

ここにブロック（VerifySignature）があります。



```
call [OpenQbitBlock1 .VerifySignature]
```

前のブロックを実行する前に、まず「符号」テーブルからバイナリ形式のファイル（file.sig）をダウンロードする必要があります。

```
sqlite3 op.sqlite3 "select sign from sign where=id_addr;"
```

id_addr.これは、"trans"テーブルのソースアドレスのIDです。

前の検索リクエストでは、現在のトランザクションの電子署名のBase64形式のデータを提供してくれますが、これを元の形式（バイナリ）に変換するにはBlock(DecoderFileBase64)が必要です。

バイナリファイル(file.sig)を持っているので、発信者の公開鍵と受信者の公開鍵もバイナリ形式でシステムにロードしなければなりません。

- ✓ 公開鍵のホームアドレス
- ✓ 受信者の公開鍵アドレス
- ✓ アクティブに送信されました。
- ✓ 電子署名（file.sig）

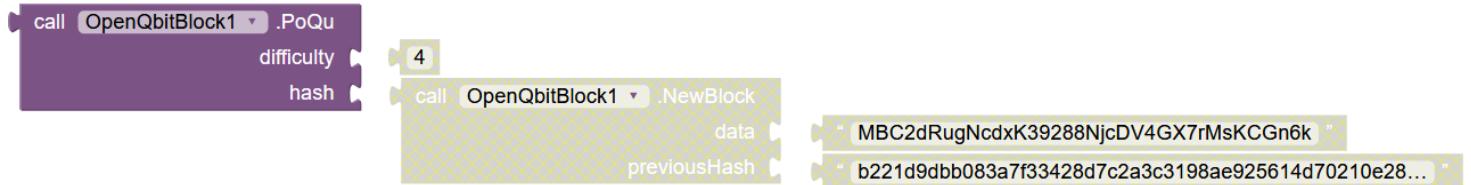
バイナリ形式の公開鍵は、共有データベース publickeys.db から適切な形式（バイナリ）でダウンロードできます。

この処理は、トランザクション・キー内の個々の操作に対して実行されなければならない。

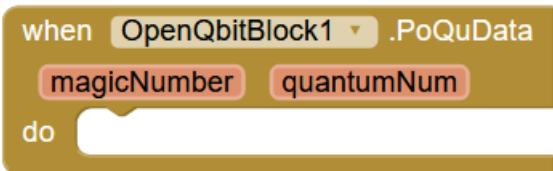
最後に、ブロックチェーンに次のブロックを追加して、トランザクションキューを処理するノードとして選ばれるように、システムがどのようにコンセンサスな方法でノードを選択するのかをレビューします。

このように、トランザクションキューを処理するための勝者ノードを選択する方法は、Mini BlocklyChainシステムのために開発されたアルゴリズムに基づいています。

PoQu「量子の証明」コンセンサスをどのように実装したかこのコンセンサス処理は、量子乱数の生成に基づいて、(PoQu)ブロックを用いて適用される。



まず、ブロック(PoQu)がメソッド(PoQuData)で配信される2つのパラメータを取得します。



`magicNumber`は数"nonce"は、5よりも大きくない難易度で内部PoWを作つて得られる整数であり、この番号は、ノードに最初の要件を与えることを担当している`magicNumber`とノードは、0から1の範囲内にある量子乱数の生成システムの数を取得するために要件を作ることができるようになります、この番号は、"投票"と呼ばれるテーブルに格納されます実行中のノードのためにランダムに確立された確率を与えるでしょう。

投票"テーブルは、各ノードによって計算されたquantumNumを格納します。このストレージは、各システム設計で確立された特定の時間まで、ノードの数で行われますが、それはエントリの $((N/2) + 1)$ を持つことをお勧めします。

重要なポイントは、この時点ですでにトランザクションキューの伝送で"ピアツーピア"ネットワークを使用する場合には、携帯電話の自動システムを介して各ノードのローカル時間の同期を確立している必要があるということです。

ノード内のcronエージェントの設定。システムノード（携帯電話）での時間同期（分・秒単位）」を参照してください。

この例では、バックアップ通信ネットワークを使用しているので、ノードの分と秒の同期は必要ありません。ノード間の他のすべてのプロセスは、「Peer to Peer」通信を介して行われます。

ここでは、この例で作成する"quorum.db"というデータベース内にある"vote"テーブルの構造、デザイン、作成を見ていきましょう。

§ sqlite3

SQLiteバージョン3.32.2 2020-06-20 15:25:24

使用方法のヒントは「.help」を入力してください。

トランジエントなインメモリデータベースに接続されています。

.open FILENAME"を使用して、永続的なデータベースで再開します。

sqlite> .open quorum.db

sqlite> CREATE TABLE vote (id integer primary key AUTOINCREMENT NOT NULL, node_imei VARCHAR NOT NULL, quantumNum INTEGER NOT NULL, nonce INTEGER NOT NULL)。

sqlite> .quit

同じ投票表の作成は以下のようになりますが、SQL文をセグメント化して導入することで作成しています。

§ sqlite3

SQLiteバージョン3.32.2 2020-06-20 15:25:24

使用方法のヒントは「.help」を入力してください。

トランジエントなインメモリデータベースに接続されています。

.open FILENAME"を使用して、永続的なデータベースで再開します。

```
sqlite> .open quorum.db
```

```
sqlite> CREATE TABLE vote (
```

```
...> id 整数 AUTOINCREMENT
...> node_ime_i VARCHAR NOT NULL.
...> quantumNu_m INTEGER NOT NULL.
...> nonc_e INTEGER NOT NULL
...> );
```

```
sqlite> .tables
```

投票

```
sqlite> .quit
```

ベースとなる"quorum.db"とテーブル投票の作成でも似たようなものが見られるでしょう。

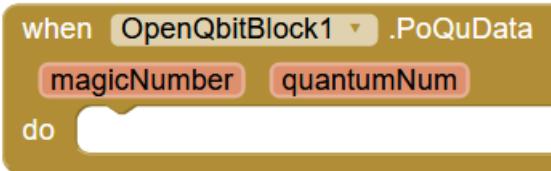


それでは、「投票」テーブルから値を取得する方法を見てみましょう。

node_imeiを使用しています。- この値は、ブロック(GetDeviceID)を使用して取得します。



quantumNum - この値は、(PoQu) ブロックを使用して得られる (PoQuData) メソッドの結果の 1 つです。



ノンス- この値は、ブロック(PoQu)を既に積分的に実行したことから得られ、メソッド(PoQuData)の magicNumberと同じです。

「投票」テーブルのINSERTSが完了したら、データベースのコピーを作成する必要があります。

一定時間が経過した後、各システムの設計に基づき、各ネットワークノード上で「cron」サービスが実行され、「投票」テーブルに次のSELECTの処理が行われるようになります。

```
SELECT node_imei FROM vote WHERE magicNumber= (SELECT max(magicNumber) FROM vote).
```

前のSELECTはより高い確率でIMEIの結果を返しましたが、今ではSELECTを実行する各ノードは結果のIMEIと自分のIMEIを比較し、一致したノードだけが最も高い確率でファイルを作成し、それはネットワーク"Peer to Peer"で複製され、勝利したノードのIMEIを含むIMEI.mbc形式のファイルになります。

勝利したノードは、トランザクションキューの処理を開始することができます。上記のすべてのブロックに基づいて

重要な2つのポイントは、それぞれのシステム制作によって、3つのプロセスを設計者ごとに見直し、カスタマイズしていく必要があるということです。

- 1.1. - 勝利ノードがトランザクションキューの処理を開始するとき、勝利ノードがオンラインであり、ネットワークとの通信が失われていないことを確認しなければならない方法またはプロセスを実装しなければならない。
2. - トランザクションキュー処理が開始されたとき、勝利したノードは、処理の開始を示す2つの制御フラグと、トランザクションキュー処理が終了したことを確認するための別のフラグを開始しなければならない。これらの2つのフラグは、ネットワーク上で全ノードに共有されなければならず、これは、接続の失敗や処理の失敗、または処理時間が長すぎることを見つけるのに役立つだろう。
- 3.- 通信障害等で当選ノードがトランザクションキューの処理ができていない場合には、即時確率範囲内の次のノードを選択する。

前述のポイントは、当選したノードがオンラインであることを確認するサービスで制御することができます、"cron"サービスを使用することができ、スクリプトは設計されたケースごとに開発する必要がありますが、シェルスクリプトの一般的な例を以下に示しますので、各ミニBlocklychainシステムのニーズに応じて変更することができます。

```
#!/bin/bash

dir="/data/data/com.termux/files/home/Sync/imei"です。

if [ !
而して

sqlite3 quorum.db "UPDATE vote SET magicNumber=0 WHERE magicNumber= (SELECT
max(magicNumber) FROM vote);"

然もなくば
```

```
MAX_NUM=$(sqlite3 quorum.db "SELECT max(magicNumber) FROM vote;")
IMEI_quorum=$(sqlite3 quorum.db "SELECT node_id FROM vote WHERE=MAX_NUM")
IMEI_local=$(cat device_imei) // ブロックを使う(GetDevice)
```

if [IMEI_quorum -eq IMEI_local] if [IMEI_quorum -eq IMEI_local

而して

touch \$MAX_NUM > IMEI.mbc

フイ

フイ

出る

31. 別冊「Ethereum & Bitcoin環境との統合について

今、私たちは、EthereumやBitcoinなどの暗号に特化した世界的に最もよく知られている2つのブロックチェーンシステムを統合する方法を見てみましょう。

まずは、Ethereum環境で可能な限りの取引を行ってくれるソフトウェアのインストールから始めてみましょう。

Ethereum（イーサリアム）とは？

Ethereumはオープンソースのプラットフォームであり、他のブロックチェーンとは異なり分散化されているため、Ethereumは多くのことを行うことができます。プログラム可能なので、開発者はこれを使って新しいタイプのアプリケーションを作成することができます。

これらの分散型アプリケーション（または「ダップス」）は、暗号モンタージュやブロックチェーン技術の恩恵を受けることができます。彼らは信頼性が高く、予測可能であり、一度Ethereumに「ロード」されると、常にスケジュール通りに実行されることを意味します。彼らはデジタル資産をコントロールして、新しいタイプの金融アプリケーションを作成することができます。彼らは分散化することができます、つまり、一人のエンティティや人が彼らを支配していないことを意味します。

現在、世界中の何千人の開発者がEthereum上でアプリケーションを作成し、新しいタイプのアプリケーションを発明しています。

- ETHまたは他の資産で安価で即時の支払いを可能にする暗号通貨のポートフォリオ
- デジタル資産を借りたり、貸したり、投資したりできる金融アプリケーション
- 分散型の市場で、デジタル資産を交換したり、現実世界の出来事についての「予測」を交換したりすることができます。
- ゲーム内に資産を持っていて、リアルマネーを獲得することもできるゲーム。
- それは、それが精巧に作られた、または作られた前提が満たされたときに実行される契約を持つプログラムであるインテリジェントな契約を持っています。

インテリジェントコントラクトは、DApps（分散型アプリケーション）との類似点を共有していますが、いくつかの重要な違いとは切り離されています。

スマートコントラクトと同様に、DAppは分散型ピアネットワークを介してプロバイダからのサービスにユーザーを接続するインターフェースです。しかし、スマートコントラクトは一定の参加者数を作成する必要がありますが、DAppsはユーザー数の制限がありません。さらに、スマートコントラクトのような金融アプリケーションに限定されるものではありません。

私たちのケースでは、Javaで開発され、シンプルで直感的な方法でEthereumブロックチェーンと対話することができます"Web3j"と呼ばれるライブラリを使用します。

以下のコマンドを実行して「Web3j」をインストールします。

```
curl -L get.web3j.io | sh
```

そして、ライブラリがこの変数を検索して正常に実行できるようにするために、実行ファイル"java"のあるパスを環境変数\$JAVA_HOMEに作成する必要があります。

JAVA_HOME= /data/data/com.termux/files/usr/bin

OpenQbit.com

変数が作成されたら、次のコマンドを実行して、ライブラリ"Web3j"がインストールされたディレクトリに行かなければなりませんが、重要なポイントは、"cd"のコマンドを実行した後、ディレクトリが隠されていることです。

```
cd .web3j
```

内部に入ったら、次のコマンドでライブラリが正常に動作しているかどうかをテストします。

```
./web3jバージョン
```

非常に似たような結果になります。

```
$ ls
source.sh  web3j  web3j-4.5.16
$ ./web3j version

Version: 4.5.16
Build timestamp: 2020-03-06 14:13:49.943 UTC
$
```

後日、Ethereumのブロックチェーン環境で使用するウォレットを以下の方法で作成します。

\$. ./web3jの財布を作成します。

先ほどのコマンドでethereumのアドレスがわかりました。

4598fe2fd6afe2508f58343c7d42f2ab492edf34

```
$ ./web3j wallet create
Please enter a wallet file password:
Please re-enter the password:
Please enter a destination directory location [/data/data/com.termux/files/home/.ethereum/testnet/keystore]:
Wallet file UTC--2020-06-27T06-12-23.819752000Z--4598fe2fd6afe2508f58343c7d42f2ab492edf34.json successfully created in: /data/data/com.termux/files/home/.ethereum/testnet/keystore
```

Bitcoin環境では、2つのオプションがあります。Bitcoinアカウントとその公開鍵と秘密鍵を生成するブロック()を使用します。これらの鍵を使用して、"Bitcoij" javaライブラリをインストールすることで、Bitcoin環境に統合することができます。

その結果、アドレスと暗号化されたデータを含むJSON形式のファイルが作成され、後に作成したアカウントの秘密鍵を生成するために使用されます。

このJSON型のファイルは、keystoreというデフォルトのディレクトリに作成されます。

ここからは、すでにWeb3jコマンドを使った操作や実行を行うことができます。

拡張機能(ConnectorSSHClient)を使用することで、これを行うことができます。

Web3j"ライブラリの異なるパラメータと操作を使用する方法を知るために、私たちは、その公式サイトのドキュメントを参照することによって私たちを助けることができます。

<https://docs.web3j.io/>

`npm install bitcoinj`



```
$ npm install bitcoinj
+ bitcoinj@0.0.0
added 1 package from 1 contributor and audited 2
09 packages in 20.528s
$
```

このライブラリを利用するためには、公式サイトに頼ることになります。

<https://bitcoinj.github.io/>

Bitcoinブロックチェーン環境に統合するための2つ目のオプションは、以下のようにTermux用のBitcoindパッケージをインストールすることです。

`apt install bitcoin`



```
$ pkg install bitcoin
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  bitcoin
0 upgraded, 1 newly installed, 0 to remove and 1
9 not upgraded.
Need to get 3601 kB of archives.
After this operation, 15.0 MB of additional disk
space will be used.
Get:1 https://dl.bintray.com/termux/termux-packa
ges-24 stable/main arm bitcoin arm 0.20.0 [3601
kB]
Fetched 3601 kB in 2s (1253 kB/s)
Selecting previously unselected package bitcoin.
(Reading database ... 19110 files and directorie
s currently installed.)
Preparing to unpack .../bitcoin_0.20.0_arm.deb .
..
Unpacking bitcoin (0.20.0) ...
Setting up bitcoin (0.20.0) ...
$
```

これで、Termux端末でbitcoindエージェントを実行すると、自動的にディレクトリにアドレスが作成されます。

/data/data/com.termux/files/hme/.bitcoin

今回のBitcoinの場合は、「Bitcoind」エージェントの以下のドキュメントに頼ることにします。

この場合は、それぞれのニーズに応じて拡張機能（ConnectorSSHClient）を使って「bitcoind」エージェントを実行することもできます。

bitcoindで使用するためのパラメータの説明。

名前

bitcoind - ビットコインコアデーモンの起動

SYNOPSIS

bitcoind [*options*] ビットコインコアデーモンの起動

説明

ビットコインコアデーモンの起動

オプション

-?

このヘルプメッセージを印刷して終了

-アラートノティファイ=<cmd

関連するアラートを受信した場合、または非常に長いフォークが表示された場合にコマンドを実行します (cmd の %s はメッセージに置き換えられます)。

-assumeevalid=<hex>。

このブロックが文字列に含まれている場合、彼と彼の祖先が有効であると仮定し、書き込み検証をスキップする可能性があります (0 for verify all, default:
000000000000000f1c54590ee18d15ec70e68c8cd4cfbadb1b4f11697eee, testnet:
000000000000037a8cd3e06cd5edbfe9dd1dbcc5dacab279376ef7fcf2b4c75)。

- ブロック通知=cmd

ベストブロックが変更されたときにコマンドを実行します(cmdの%sはブロックハッシュに置き換えられます)。

- ブロック再構築extratxn=<n>

コンパクトなブロック再構成のためにメモリに保持する余分なトランザクション (デフォルト: 100)

-blocksdir=<dir>の場合

ブロックディレクトリを指定する (デフォルト: <datadir>/blocks)

- ブロック内のみ

ネットワークパートナーからの取引を拒否する場合ポートフォリオやRPC取引は影響を受けません。(デフォルト: 0)

-conf=<アーカイブ>

設定ファイルを指定します。相対パスの先頭には、データディレクトリの位置が付けられます。(デフォルト: bitcoin.conf)

- デーモン

悪魔のようにバックグラウンドで動作し、コマンドを受け付けます。

-datadir=<dir>

データディレクトリを指定します。

-dbcache=<n>

データベースキャッシュの最大サイズ <n> MiB (4~16384、デフォルトは450)。さらに、未使用的 mempool メモリはこのキャッシュのために共有されます (-maxmempool を参照)。

-debuglogfile=<ファイル>

デバッグログファイルの場所を指定します。相対パスの先頭には、ネットワーク固有のデータディレクトリの位置が付けられます。(-nodebuglogfile を無効にする。デフォルト: debug.log)

-includeconf=<ファイル>

追加の設定ファイルを **-datadir** パスからの相対パスで指定します (コマンドラインからではなく、設定ファイルからのみ使用できます)

-ロードブロック=<ファイル>

開始時に外部ファイルblk000?

-maxmempool=<n>

トランザクションメモリプールを<n>メガバイト以下に保つ (デフォルト: 300)

-maxorphantx=<n>

メモリ内の切断可能なトランザクションの最大値<n>を保持する (デフォルト: 100)

-mempoolexpiry=<n>です。

トランザクションを<n>時間を超えてmempoolに保持しない (デフォルト: 336)

-par=<n>

ダッシュ内での検証スレッド数を設定します (-6~16、0 = auto、<0 = すべてのコアを解放、デフォルトは0)。

-パーシステムプール

シャットダウン時にmempoolを保存し、再起動時に読み込む場合(デフォルトは1)

-pid=<ファイル>

PID ファイルを指定します。相対パスの先頭には、ネットワーク固有のデータディレクトリの位置が付けられます。(デフォルト: bitcoind.pid)

-punza=<n>

古いブロックの剪定（除去）を可能にすることで、収納の必要性を軽減します。これにより、特定のブロックを削除するためにRPCのプルーニングチェーンを呼び出すことができ、MIBのターゲットサイズが提供されている場合には、古いブロックの自動プルーニングが可能になります。このモードは -

txindex と **-rescan** とは互換性がありません。警告: この設定を逆にするには、ブロックチェーン全体を再ダウンロードする必要があります (デフォルト: 0 = ブロックの剪定を無効化、1 = RPC による手動剪定を許可、>>550 = MIB で指定されたターゲットサイズ以下になるようにブロックファイルを自動的に剪定)

-再インデックス

ディスク上のblk*.datファイルの文字列状態とブロックインデックスを再構築します。

-再インデックスチェーンステート

現在インデックス化されているブロックからチェーンの状態を再構築します。剪定モードの場合や、ディスク上のブロックが破損している可能性がある場合は、代わりに完全な再インデックスを使用してください。

-溜め息

umask 077 の代わりに、デフォルトのシステムパーミッションで新しいファイルを作成 (ポートフォリオ機能が無効の場合のみ有効)

-トクスインデックス

getrawtransaction rpc コールで使用される完全なトランザクションインデックスを保持します (デフォルト: 0)

-バージョン

プリント & ゴーバージョン

接続オプション。

-アドノード=<ip>

接続先のノードを追加して、接続をオープンにしてみます(詳細は"addendum"のRPCコマンドヘルプを参照してください)。このオプションは、複数のノードを追加するために複数回指定することができます。

-banscore=<n>

行儀の悪い同僚を切断するためのしきい値(デフォルト: 100)

- (徳井) 今のうちに..

不正行為をした同僚が再接続しないようにするための秒数 (デフォルト: 86400)

-bind=<アドレス

与えられたアドレスに自分を縛り付け、常にそれに耳を傾ける。IPv6では[host]:port表記を使用します。

-connection=<ip

noconnect は自動接続を無効にします (このペアのルールは **-addnode** と同じです)。このオプションは、複数のノードに接続するために複数回指定することができます。

-見つける

自分の IP アドレスを検出する (デフォルト: リスニング時には 1、**-externalip** や **-proxy** ではなく **-proxy**)

-ディーエヌエス

addnode、**-seednode**、**-connect** の DNS 検索を許可する(デフォルト: 1)

-ディーエヌエスシード

アドレス数が少ない場合、DNSルックアップによるペアアドレスの問い合わせ (デフォルト: **-connection** が使用されていない限り1)

-イネーブルビップ61

BIP61による拒否メッセージの送信(デフォルトは1)

-EXTERNALIP=<IP>のようになります。

自分のパブリックアドレスを指定する

-きょうせいしゅ

同僚のアドレスを DNS 検索で常に照会する(デフォルト: 0)

-聞く

外部からの接続を受け入れる(デフォルト: **-proxy** または **-connection** がない場合は 1)

-聴いて

Tor隠しサービスを自動的に作成する(デフォルト: 1)

-maxconnections=<n>

同僚とのコネクションを最大で<n>維持する(デフォルト: 125)

-maxreceivebuffer=<n>のようになります。

接続あたりの最大受信バッファ、<n>*1000 バイト (デフォルト: 5000)

-maxsendbuffer=<n>

接続あたりの最大送信バッファ、<n>*1000 バイト (デフォルト: 1000)

-最大時間設定

ペアの平均時間補正の調整に許容される最大値。この分だけ、ローカルタイムのパースペクティブが前方の組と後方の組に影響を与えることができます。(デフォルト: 4200秒)

-maxuploadtarget=<n>

送信トラフィックを指定された目標値以下に保つようにします(24hの場合はMiB単位)、0 = 制限なし(デフォルトは0)

-onion=<<<ip:port>>のようになります。

別の SOCKS5 プロキシを使用して、Tor の隠れたサービスを介してピアに到達するには、**-noonion** を無効に設定します(デフォルトは **-proxy**)。

-オンリーネット=<ネット>

<net>ネットワーク(ipv4, ipv6, onion)を介してのみ発信接続を行います。着信接続は、このオプションの影響を受けません。このオプションは、複数のネットワークを許可するために複数回指定することができます。

-ペアフィルタ

開花フィルタによるフィルタリングとトランザクションのブロックをサポート(デフォルトは1)

-許可マルチシグ

リレーはP2SHマルチシグではありません(デフォルト: 1)

-ポート=<ポート

port' での接続を聞く (デフォルト: 8333, testnet: 18333, regtest: 18444)

-プロキシ=<ip:port

SOCKS5プロキシ経由で接続する場合は、-noproxyをoffに設定してください。

-プロキシランダム化

各プロキシ接続の認証情報をランダムにするこれにより、Tor フローの分離を有効にします (デフォルト : 1) 。

-seednode=<ip

ノードに接続してペアアドレスを取得し、切断します。このオプションは、複数のノードに接続するため複数回指定することができます。

-タイムアウト=<n

接続タイムアウトをミリ秒単位で指定します (最小 : 1、デフォルト : 5000)

-torcontrol=<ip>:<port>。

オニオンリスニングが有効な場合に使用する Tor 制御ポート (デフォルト: 127.0.0.1:9051)

-パスワード=<パス

Tor 制御ポートのパスワード (デフォルト: 空欄)

-アップニップ

UPnP を使用してリスニングポートを割り当てます (デフォルト: 0)

-ホワイトリスト=<アドレス

特定のアドレスへのリンクと、それに接続するホワイトリスト化されたパートナー。IPv6では [host]:port表記を使用します。

-ホワイトリスト=<IPアドレスまたはネットワーク

ホワイトリスト化されたペアは、与えられたIPアドレス（例：1.2.3.4）またはCIDRのアノテーションされたネットワーク（例：1.2.3.0/24）から接続されています。複数回指定することができます。ホワイトリスト化されたペアはDoSで禁止することはできません。

ポートフォリオのオプション。

-アドレスタイプ

使用するアドレスの種類 ("レガシー", "p2sh-segwit", "bech32", デフォルト: "p2sh-segwit")

-部分的なコストを避ける

各出力ごとに選択するのではなく、方向ごとに出力をグループ化し、すべてまたはなしを選択します。アドレスは一度しか使用されないのでプライバシーは強化されていますが(使われた後に誰かが送信しない限り)、追加された制限のために通貨の選択が最適でない可能性があるため、若干レートが高くなる可能性があります(デフォルト: 0)

-タイプ変更

どの為替レートを使用するか（「レガシー」、「p2sh-segwit」、「bech32」）。デフォルトは - **addressstype** と同じですが、-addressstype=p2sh-segwit はネイティブのセグウィットアドレスに送信する際にネイティブのセグウィット出力を使用します)

-財布から

ウォレットをロードせず、ウォレットからのRPCコールを無効にする

-ディスクカードキー=<amt

変更を料金表に追加して破棄する許容度を示す料金表のレート（BTC/kB単位）（デフォルト：0.0001）。注意：このレートでダストであれば出力は廃棄されるが、ダスト再送レートまでは常に廃棄され、それ以上の廃棄レートはより長いターゲットのレート推定によって制限される。

-fallbackfee=<amt>です。

料金見積もりのデータが不足している場合に使用する料金レート（BTC/kB単位）（デフォルト：0.0002

-キープール=<n

キープールのサイズを <n> に設定 (デフォルト: 1000)

-mintxfee=<amt

これより低いレート（BTC/kB単位）は、トランザクションの作成のためのゼロレートとみなされます（デフォルト：0.00001）。

-paytxfee=<amt>のようになります。

送信するトランザクションに追加するレート (BTC/kB単位) (デフォルト: 0.00)

-再スキャン

ブロックチェーンを再スキャンして、先頭のポートフォリオ・トランザクションの欠落を探す

-サルベージウォレット

ポート中の破損したウォレットの秘密鍵を取得しようとした場合

-情報交換の無効

トランザクションの送信時に未確認の変更を消費する（デフォルト：1

-txconfirmtarget=<n>。

支払手数料が設定されていない場合は、平均して n ブロック以内に確認を開始するための十分な手数料を含める（デフォルト：6）。

-ポートフォリオの維持管理

冒頭でポートフォリオを最新のフォーマットに更新する

-ウォレット=<パス>

ポートフォリオデータベースのパスを指定します。複数回指定して複数のポートフォリオを読み込むことができます。パスは絶対パスでない場合は<walletdir>との関係で解釈され、存在しない場合は作成されます(wallet.datファイルやログファイルを含むディレクトリなど)。後方互換性のために、<walletdir>にある既存のデータファイルの名前も受け入れます)。

-ワイルトフ^トロート^トキャスト

ポートフォリオ拡散取引を行う (デフォルトは1)。

-walletdir=<dir>

ポートフォリオを保存するディレクトリを指定する (デフォルト: 存在する場合は<datadir>/portfolios、そうでない場合は <datadir>)

-walletnotify=<cmd>

ポートフォリオトランザクション変更時のコマンド実行 (cmd の %s を TxID に置き換えています)。

-ワレトリーブ

アクティブ化されたRBF全体を含めるためのオプションを持つトランザクションを送信する (RPCのみ、デフォルト: 0)

-zapwallettxes=<モード>

ポートフォリオからすべてのトランザクションを削除し、ブロックチェーンの一部のみを最初に -rescan で取得する (1 = tx-metadata を保持、2 = tx-metadata を削除)

ZeroMQの通知オプション。

-zmqpubhashblock=<アドレス>です。

address」でパブリッシングブロックを有効にする

-zmqpubhashblockhwm=<n>

送信メッセージの公開ブロックを高い透かしで設定します（デフォルト：1000

-zmqpubhashtx=<アドレス>。

「アドレス」でのハッシュトランザクションの公開を有効にする

-ZMQPUBHASHTXHWM=<N>

高透かしハッシュトランザクションの出力メッセージを公開するように設定する(デフォルト: 1000)

-zmqpubrawblock=<アドレス>

address」で生のパブリッシングブロックを有効にします。

-zmqpubrawblockhwm=<n>

生のブロックの送信メッセージの高い透かしを置いて下さい（デフォルト: 1000

-zmqpubrawtx=<アドレス>です。

address」での生のトランザクションの公開を有効にする

-zmqpubrawtxhwm=<n>

発行総トランザクション出力メッセージの高透かしを設定する（デフォルト：1000

デバッグ/テストオプション。

-debug=<カテゴリ>

デバッグ情報を出力します (デフォルト: **-nodebug**, 'category' の指定はオプション)。<category> が指定されていない場合、または <category> = 1 の場合、すべてのデバッグ情報を出力します。<カテゴリ>には次のようなものがあります: net, tor, mempool, http, bench, zmq, db, rpc, estimatefee, addrman, selectcoins, reindex, cmpctblock, rand, prune, proxy, mempoolrej, libevent, coindb, qt, leveldb.

-debugexclude=<カテゴリ>

デバッグ情報をカテゴリから除外します。**debug=1**と組み合わせて使用することで、1つ以上の指定されたカテゴリを除くすべてのカテゴリのデバッグコードを生成することができます。

-ヘルプデバッグ

デバッグオプションを含むヘルプメッセージを表示して終了

-ロギップス

デバッグ出力に IP アドレスを含める (デフォルト: 0)

-ログタイムスタンプ

タイムスタンプ付きのデバッグ出力を準備する (デフォルト: 1)

-maxtxfee=<amt>

単一のポートフォリオ取引またはグロス取引で使用するための最大合計手数料 (BTC単位)。

-コンソールへの印刷

コンソールにトレース/デバッグ情報を送信します(デフォルト: -daemonがない場合は1)。ファイルへのロギングを無効にするには、**-nodebuglogfile**を設定します。

-shrinkdebugfile

クライアント起動時にdebug.logファイルを減らす (デフォルト: -debugがない場合は1)

-uacomment=<cmt>です。

ユーザーエージェント文字列にコメントを追加する

チーン選択オプション。

-テストネット

テストチーンを使用して...

ノードの再送オプション。

-バイトペシゴ

ブロードキャストおよびマイニングトランザクションにおけるシグオプあたりのバイト相当量（デフォルト：20

-データキャリア

中継および地雷データキャリアのトランザクション（デフォルト：1）

-データキャリア化

再送・抽出するデータキャリアのトランザクション内の最大データサイズ（デフォルト：83）

-メモリプールの再置換

メモリプール内のトランザクションの置換を有効にする（デフォルト：1）

-minrelaytxfee=<amt>

これより低い料金（BTC/kB単位）は、再送、抽出、トランザクションの作成のためのゼロ料金とみなされます（デフォルト：0.00001）。

-ホワイトリストフォーサーリレイ

トランザクションがすでに mempool にあるか、ローカルの再送ポリシーに違反している場合でも、ホワイトリストされたパートナーのトランザクションの再送を強制する（デフォルト：0）

-ホワイトリストリレー

トランザクションが送信されなくても、ホワイトリストに登録されたペアから受信した送信済みのトランザクションを受け入れる（デフォルト：1）。

作成オプションをブロックします。

-ブロックマックスウェイト=<n>

BIP141ブロックの最大重量を設定します（デフォルト：3996000）

-BLOCKMINTXFEE=<AMT>のようになります。

ブロック作成に含まれるトランザクションの最低手数料率（BTC/kB単位）を設定します。（デフォルト：0.00001）

RPC サーバーのオプション。

- レストラン

公開 REST リクエストを受け入れる (デフォルト: 0)

-rpcallowip=<ip>

指定したソースからのJSON-RPC接続を許可します。これらは、<ip>単一IP(例: 1.2.3.4)、ネットワーク/ネットワークマスク(例: 1.2.3.4/255.255.255.255.0)、ネットワーク/CIDR(例: 1.2.3.4/24)に対して有効です。このオプションは複数回指定できます。

-rpcauth=<userpw>

JSON-RPC接続のためのユーザー名とパスワード HMAC-SHA-256。userpw' フィールドは次のような形式になります: 'username': 'SALT': \$ 'HASH'.share/rpcauthには定型的なpythonスクリプトが含まれています。その後、クライアントは、`rpcuser=<USERNAME>/rpcpassword=<PASSWORD>` の引数ペアを使用して正常に接続します。このオプションは複数回指定できます。

-rpcbind=<アドレス>[:ポート]

特定のアドレスにリンクしてJSON-RPC接続をリッスンします。公共のインターネットなど信頼性の低いネットワークにRPCサーバを晒さないようにしましょう!このオプションは、**-rpcallowip**も渡されない限り無視されます。ポートはオプションで、**-rpcport**を上書きします。IPv6の場合は[host]:port表記を使用します。このオプションは複数回指定することができます (デフォルトは 127.0.0.0.1 と ::1、つまり localhost)。

-rpccookiefile=<loc>

認証クッキーの場所。相対パスの先頭には、ネットワーク固有のデータディレクトリの位置が付けられます。(デフォルト: データディレクトリ)

-rpcpassword=<pw>

JSON-RPC接続時のパスワード

-rpcport=<ポート>

`port`' で JSON-RPC 接続をリッスンします (デフォルト: 8332, testnet: 18332, regtest: 18443)

-rpcserialversion

生のトランザクションのシリアル化、または非言語モードで返されるブロックの16進数を設定し、
`segwit(0)`や`segwit(1)`ではなく(デフォルトは1)

-rpcthreads=<n

RPC 呼び出しを処理するスレッド数を設定します (デフォルト: 4)

-rpcuser=<user

JSON-RPC接続用のユーザー名

-ナーバ

コマンドラインコマンドとJSON-RPCを受け入れる

32. ソフトウェアのライセンスと使用。

Android

<https://source.android.com/setup/start/licenses>

Termux

<https://github.com/termux/termux-app/blob/master/LICENSE.md>

Node.js

<https://raw.githubusercontent.com/nodejs/node/master/LICENSE>

SQLite

<https://www.sqlite.org/copyright.html>

Git

<https://git-scm.com/about/free-and-open-source>

sqlite-to-rest

<https://github.com/olsonpm/sqlite-to-rest/blob/dev/license.txt>

Redis

<https://redis.io/topics/license>

WorldTime API

<http://worldtimeapi.org/pages/faqs#commercial-apps>

Tor Network

<https://github.com/torproject/tor/blob/master/LICENSE>

Syncthing Network

<https://forum.syncthing.net/t/syncthing-is-now-mplv2-licensed/2133>

OpenSSH

<https://www.openssh.com/features.html>

Putty SSH

<https://www.chiark.greenend.org.uk/~sgtatham/putty/licence.html>

MIT App Inventor 2 ユニバーサルとApp Inventor Blockly

<https://appinventor.mit.edu/about/termsofservice>

SQLite Expert Personal -フリーウェア

<http://www.sqliteexpert.com/download.html>

Apache Ant

<https://ant.apache.org/license.html>

WGET

<https://www.gnu.org/software/wget/>

OpenJDK

<https://openjdk.java.net/legal/>

外部拡張子。

JSONTOOLS

<https://thunkableblocks.blogspot.com/2017/07/jsontools-extension.html>

Mini BlocklyChainシステムのオープンソース版と商用版のライセンスについては、公式ウェブサイト
<http://www.openqbit.com> を参照してください。

Mini BlocklyChain、MiniBlockly、BlocklyCode、MiniBlockMiniChain、QBlocklyはOpenQbitで登録されている商標です。

ミニBlocklyChainが公開されています。

Mini BlocklyChainのすべてのコードとドキュメントは、作者によってパブリックドメインに捧げられています。すべてのコード作者と彼らが働く企業の代表者は、自分たちの貢献をパブリックドメインに捧げる宣誓供述書に署名しており、それらの宣誓供述書の原本はOpenQbit Mexicoのメインオフィスの安全な場所に保管されています。オリジナルの Mini BlocklyChain (OpenQbit) 拡張機能をソースコードまたはコンパイルされたバイナリとして、商用・非商用を問わず、どのような目的でも、どのような手段でも、誰でも自由に公開、使用、配布することができます。

前の段落は、Mini BlocklyChain 内のコードと文書の成果物、すなわち Mini BlocklyChain ライブラリの中で実際に大規模なアプリケーションと一緒にグループ化して出荷する部分に適用されます。コンパイルプロセスの一部として使用されるスクリプト(例えば、autoconfによって生成される「設定」スクリプト)は、他のオープンソースライセンスに含まれている可能性があります。しかし、これらのコンパイルスクリプトはいずれも最終的な Mini BlocklyChain デリバリブルライブラリには含まれていませんので、これらのスクリプトに関するライセンスは、Mini BlocklyChain ライブラリを複製して使用する権利を評価する際の要因となるべきではありません。

Mini BlocklyChainの成果物のコードはすべて一から書かれています。他のプロジェクトやオープンインターネットからのコードは一切使用していません。コードの各行を元の作者まで遡ることができます。それらの作者はすべてファイルにパブリックドメインの献辞を持っています。したがって、Mini

BlocklyChain のコードベースはクリーンであり、オープンソースプロジェクトではなく、他のオープンソースプロジェクトからライセンスされたコードで汚染されていません。

Mini BlocklyChainはオープンソースで、好きなだけコピーを作り、そのコピーで好きなことができる事を意味します。しかし、Mini BlocklyChainはオープンソースではありません。Mini BlocklyChain をパブリックドメインに保ち、コードがプロプライエタリなものやライセンスされたコンテンツに汚染されないようにするために、プロジェクトでは未知の人からのパッチを受け付けていません。Mini BlocklyChain のすべてのコードは、Mini BlocklyChain で使用するために特別に書かれたものなので、オリジナルです。インターネット上の未知のソースからコピーされたコードはありません。

Mini BlocklyChainはパブリックドメインであり、ライセンスは必要ありません。それでも、Mini BlocklyChainを使用する権利を法的に証明してほしいという組織もあります。このような状況になった場合には、以下のような状況が考えられます。

- あなたの会社は、著作権侵害のクレームに対して補償を求めていました。
- パブリックドメインを認めていない管轄区域でMini BlocklyChainを使用しています。
- あなたは、著作者の作品をパブリックドメインに置く権利を認めていない法域でMini BlocklyChainを使用しています。
- Mini BlocklyChainを使用して配布する法的権利を持っている証拠として、有形の法的文書を用意しておきたいものです。
- 法務部は免許証を買わなければならないと言っています。

上記の状況のいずれかに当てはまる場合は、Mini BlocklyChain開発者全員を雇用しているOpenQbit社が、Mini BlocklyChainタイトル保証を販売します。タイトルワランティとは、Mini BlocklyChainの主張された作者が眞の作者であり、作者がMini BlocklyChainをパブリックドメインに捧げる法的権利を持っていること、OpenQbitがライセンスの主張に対して精力的に防御することを示す法的文書です。Mini BlocklyChainのタイトル保証書の販売による収益はすべて、Mini BlocklyChainの継続的な改善とサポートのための資金として使用されます。

寄託されたコード

Mini BlocklyChainを完全にフリーでロイヤリティフリーに保つために、プロジェクトはパッチを受け付けていません。提案された変更をして、コンセプトの証明としてパッチを入れてくれると助かります。ただし、パッチを一から書き換えるも怒らないでください。このモダリティといいくつかの類似のサポートの購入なしでそれを使用する非商用またはオープンソースのライセンスの種類は、企業の規模に関係なく、個人または企業の使用は、次の法的な前提条件によって支配されます。

保証の免責事項。許諾者は、適用法によって要求されるか、または書面で合意されない限り、『作品』（および各投稿者はその投稿を提供する）を「現状のまま」提供し、明示または默示を問わず、いかなる種類の保証または条件（タイトル、非侵害、市販性、特定目的への適合性の保証または条件を含むがこれに限定されない）もなく、「現状のまま」で提供しています。あなたは、作品の正しい使用または再配布を決定し、本ライセンスの下での許可の行使に関連するリスクを引き受けるために単独で責任を負うものとします。

本ソフトウェアの使用により発生した金銭的・その他の損失は、当該当事者が負担するものとします。すべての法的紛争は、当事者がメキシコシティ、国メキシコの管轄下でのみ裁判所に提出します。

商業的なサポート、使用およびライセンスのためには、OpenQbitまたはその企業と利害関係のある当事者との間で契約または契約を締結する必要があります。

配信マーケティングの規約は予告なく変更される場合がありますので、サポート条項やライセンス条項の非商用・商用への変更は公式サイト www.openqbit.com でご確認ください。

任意の法的性質の任意の個人、ユーザー、民間または公共団体、または世界の任意の部分から、単にソフトウェアを使用する任意の人、ユーザー、民間または公共団体は、この文書に確立された条項と事前の通知なしに www.openqbit.co のポータルでいつでも変更することができ、非商業的または商業的な使用で OpenQbit の裁量で適用することができるものを条件なしで受け入れます。

Mini BlocklyChainについての質問や情報は、App Inventor のコミュニティか、さまざまな Blockly システムのコミュニティがそのまま利用できるようにしてください。AppBuilder、Trunkableなど、およ

び/または質問の需要のためのメールへ opensource@openqbit.com 3から5営業日からの回答を取ることができます。

商用利用で対応。
support@openqbit.com

業務用販売。
sales@openqbit.com

法的情報とライセンスに関する質問や懸念事項
legal@openqbit.com

