ファイル受け取りシステム 技術仕様書

目次

[１ 概要 4](#_Toc147245204)

[２ 動作環境 5](#_Toc147245205)

[３ 使用方法 6](#_Toc147245206)

[３-１ ファイルを受けとる方法 6](#_Toc147245207)

[３-２ ファイルを送信する方法 8](#_Toc147245208)

[４ 詳細な仕様 9](#_Toc147245209)

[５ ソースコードの閲覧方法 10](#_Toc147245210)

[６ 構成 12](#_Toc147245211)

[６-１ コンピューターの構成 12](#_Toc147245212)

[６-２ フレームワークやライブラリ等の構成 13](#_Toc147245213)

[６-２-１ 送信プログラムのフレームワークやライブラリ 13](#_Toc147245214)

[６-２-２ 中継プログラムのフレームワークやライブラリ 13](#_Toc147245215)

[６-２-３ 受信プログラムのフレームワークやライブラリ 13](#_Toc147245216)

[６-３ 本番環境におけるプログラムの配置場所 14](#_Toc147245217)

[６-３-１ 送信プログラムの配置場所 14](#_Toc147245218)

[６-３-２ 中継プログラムの配置場所 15](#_Toc147245219)

[６-３-３ 受信プログラムの配置場所 16](#_Toc147245220)

[６-４ ファイル構成 17](#_Toc147245221)

[６-４-１ 送信プログラムのファイル構成 17](#_Toc147245222)

[６-４-２ 中継プログラムのファイル構成 18](#_Toc147245223)

[６-４-３ 受信プログラムのファイル構成 19](#_Toc147245224)

[６-５ 関数一覧 20](#_Toc147245225)

[６-５-１ 送信プログラムの関数一覧 20](#_Toc147245226)

[６-５-２ 中継プログラムの関数一覧 21](#_Toc147245227)

[６-５-３ 受信プログラムの関数一覧 22](#_Toc147245228)

[６-６ API一覧 23](#_Toc147245229)

[６-６-１ 認証済みURLの取得 23](#_Toc147245230)

[６-６-２ ファイルのアップロード 24](#_Toc147245231)

[６-６-３ ファイルの最終更新日時の取得 25](#_Toc147245232)

[６-６-４ ファイル本体のダウンロード 26](#_Toc147245233)

[７ 動作の流れ 27](#_Toc147245234)

[７-１ 大まかな動作の流れ 27](#_Toc147245235)

[７-２ 詳細な動作の流れ 28](#_Toc147245236)

[７-２-１ アップロードの流れ 28](#_Toc147245237)

[７-２-２ 最終更新日時を保存する流れ 28](#_Toc147245238)

[７-２-３ 受信プログラムの起動順序 29](#_Toc147245239)

[７-２-４ 最終更新日時を監視する流れ 29](#_Toc147245240)

[８ デバッグの方法 30](#_Toc147245241)

[９ 更新方法 31](#_Toc147245242)

[９-１ 送信プログラムの更新方法 31](#_Toc147245243)

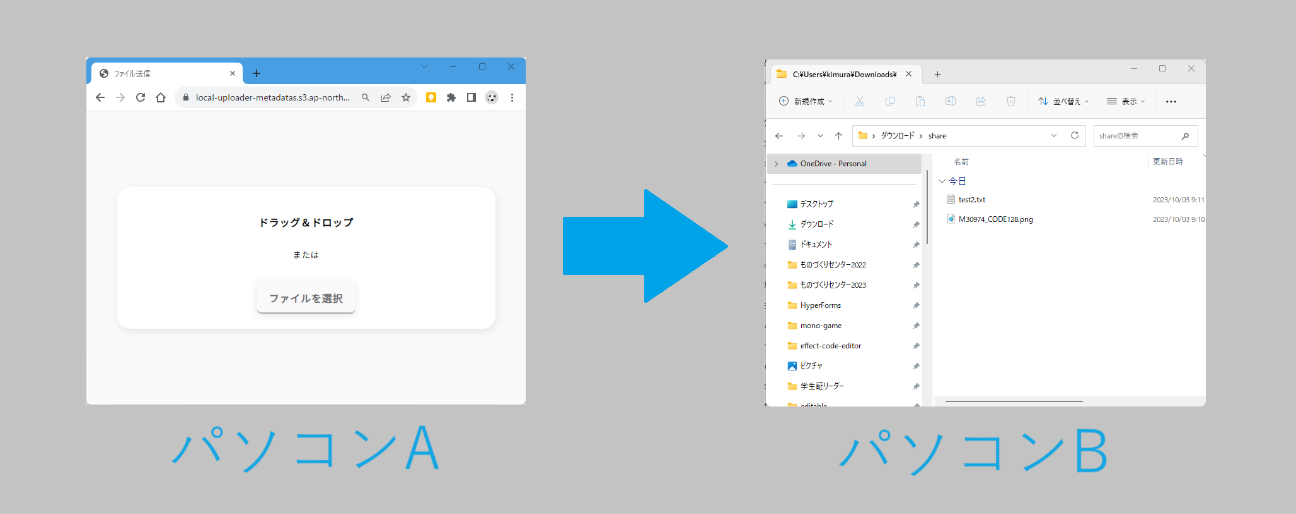
[９-２ 中継プログラムの更新方法 32](#_Toc147245244)

[９-３ 受信プログラムの更新方法 33](#_Toc147245245)

# 概要

本システムは、webページから特定のコンピューターへファイルを送信するシステムです。

想定している使用用途は、ものづくりセンター内のプラズマ加工機や３Dプリンター等の機械設備の運用です。不特定多数が機械設備を利用するにあたって、各々が設計したファイルを施設内のwindowsパソコンへ転送する必要があります。このシステムを活用すると、ホームページからファイルを転送することができます。



# 動作環境

本システムを利用するためには、以下の要件を満たすコンピューターが必要です。

**受信側に必要な動作環境**

* Windows 64bitのみ動作確認済。
* インターネットに接続されている必要があります。

**送信側に必要な動作環境**

* ブラウザが動くコンピューターなら、何でもOKです。

# 使用方法

## ファイルを受けとる方法

1. 以下のURLまたはQRコードから、受信用アプリケーションをダウンロードしてください。

<https://local-uploader-metadatas.s3.ap-northeast-1.amazonaws.com/download.html>

QR コード

自動的に生成された説明

以下の画像のようにGoogle Chromeにブロックされた場合は、右側の小さい矢印をクリックして、ダウンロードを続行してください。

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション

自動的に生成された説明

1. windowsのコンピューターで、受信用アプリケーションを起動してください。「PCが保護されました」と表示された場合は、「詳細情報」をクリックした後、「続行」をクリックしてください。

グラフィカル ユーザー インターフェイス, アプリケーション

自動的に生成された説明　 グラフィカル ユーザー インターフェイス, アプリケーション

自動的に生成された説明

1. 受信用アプリケーションが起動します。

**グラフィカル ユーザー インターフェイス, テキスト, アプリケーション

自動的に生成された説明**

1. ファイルを受け取りたいフォルダを選択してください。

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, メール

自動的に生成された説明

1. 送信用webページのURLを、ファイルを送信して欲しい人（学生等）に公開してください。

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション

自動的に生成された説明

1. ファイルを受け取る準備ができました。以下の条件を満たしている間、継続的にファイルを受信して、保存先フォルダの中身を更新します。

**条件**

* + 受信用アプリケーションが起動している必要があります。
  + 受信側のコンピューターがインターネットに接続されている必要があります。
  + 受け取り用のフォルダが存在している必要があります。

## ファイルを送信する方法

受信側の担当者からURLを教えて貰い、アクセスして、ファイルをアップロードしてください。

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, PowerPoint

自動的に生成された説明

# 詳細な仕様

1. 受信側のファイルの更新頻度は、15秒につき1回です。アップロードした後に15秒以内に反映されない場合は、何らかの不具合が発生しています。
2. 受信側のパソコンの電源が切れている場合もアップロードは可能です。受信側のパソコンの電源を入れた際に、電源が落ちていた間に受信したファイルを、まとめてダウンロードします。
3. １台のコンピューターの中で、受信用アプリケーションを２つ以上同時に動かすことはできません。
4. 送信用のwebページのURLは、受信側のデバイスごとに固有です。
   * 受信用アプリケーションを再起動した場合 → URLは変わりません。
   * 受信用パソコンの中身の回路基板を交換した場合 → URLが変わる可能性があります。
   * 受信するフォルダを変えた場合 → URLは変わりません。

# ソースコードの閲覧方法

送信プログラムと受信プログラムのソースコードは、以下のGitHubリポジトリに保管しています。

<https://github.com/bibakichi/share_file_to_local>

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, メール

自動的に生成された説明

ただし、GitHub上のコードを書き換えても本番環境に反映されません。CI/CD（※）等の設定は行っていないためです。更新の際は、各所に散らばっている本番環境のプログラムを、手動で入れ換える必要があります。

※CI/CDとは、GitHub上のコードを自動で本番環境に反映させる仕組みのこと。

中継プログラムのソースコードは、以下のAWS Lambda関数に保存されています。

* Lambda関数「local\_uploader\_sign\_url」

<https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/functions/local_uploader_sign_url?fullscreen=true&tab=code>

グラフィカル ユーザー インターフェイス, テキスト, メール

自動的に生成された説明

* Lambda関数「local\_uploader\_handle\_change」

[https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/functions/local\_uploader\_handle\_change?fullscreen=true&tab=code](https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1%23/functions/local_uploader_handle_change?fullscreen=true&tab=code)

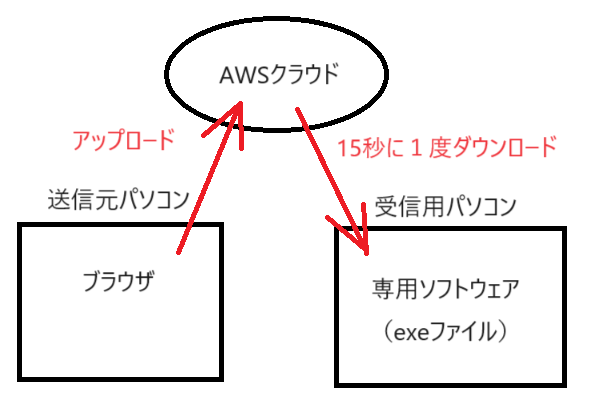
グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, メール

自動的に生成された説明

# 構成

## コンピューターの構成

本システムの構成を説明するうえで、２台のコンピューターと１つのクラウドが登場します。それぞれ送信元パソコンと受信用パソコン、そして中継用のAWSクラウドです。



## フレームワークやライブラリ等の構成

### 送信プログラムのフレームワークやライブラリ

送信プログラムはHTML, CSS, JavaScriptで書かれており、ブラウザ上で動作します。つまり、送信側のプログラムは単なるwebページだと思ってください。jQueryなどのライブラリやフレームワークは使っていません。

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, PowerPoint

自動的に生成された説明

### 中継プログラムのフレームワークやライブラリ

AWS S3とLambdaで構成しています。Lambdaの言語はnode.jsです。npmパッケージはデフォルトのものしか使用しておらず、追加のインストールは不要です。データベースやAWS API Gateway、AWS CloudFrontは使っておりません。

### 受信プログラムのフレームワークやライブラリ

Node.jsとelectronを使っています。また、パッケージ管理にはnpmを使用しています。ビルドするためにはいくつかのパッケージが必要です。詳しくは ”package.json” を確認してください。

**グラフィカル ユーザー インターフェイス, テキスト, アプリケーション

自動的に生成された説明**

## 本番環境におけるプログラムの配置場所

### 送信プログラムの配置場所

送信元パソコンで動作するwebページは、AWS S3に静的ファイルとして配置しています。配置場所は、バケット「local-uploader-metadatas」の直下です。専用のバケットを用意するのが勿体なかったため、メタデータ用のバケットに入れています。ファイルは ”index.html” の１つのみです。（CSSとJavaScriptをインラインで記述しているため）

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, PowerPoint

自動的に生成された説明

AWS S3の管理画面には、以下のURLからアクセスできます。

<https://s3.console.aws.amazon.com/s3/buckets/local-uploader-metadatas?region=ap-northeast-1>

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション

自動的に生成された説明

### 中継プログラムの配置場所

中継プログラムは、AWSのクラウド上に展開しています。

* Lambda関数「local\_uploader\_sign\_url」

<https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/functions/local_uploader_sign_url?fullscreen=true&tab=code>

* Lambda関数「local\_uploader\_handle\_change」

[https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/functions/local\_uploader\_handle\_change?fullscreen=true&tab=code](https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1%23/functions/local_uploader_handle_change?fullscreen=true&tab=code)

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, メール

自動的に生成された説明

* S3バケット「local-uploader-files」

<https://s3.console.aws.amazon.com/s3/buckets/local-uploader-files?region=ap-northeast-1&tab=objects>

* S3バケット「local-uploader-metadatas」

<https://s3.console.aws.amazon.com/s3/buckets/local-uploader-metadatas?region=ap-northeast-1>

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション

自動的に生成された説明

### 受信プログラムの配置場所

受信プログラムは、実行可能形式（exeファイル）で事務室内のパソコンにインストールしています。

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, メール

自動的に生成された説明

また、インターネットからexeファイルをダウンロードできるようにもしています。AWS S3のバケット「local-uploader-metadatas」の直下に、ファイル名「share\_file\_client.exe」で公開しています。専用のバケットを用意するのが勿体なかったため、メタデータ用のバケットに入れています。

コンピューターのスクリーンショット

自動的に生成された説明

## ファイル構成

### 送信プログラムのファイル構成

ファイルは ”index.html” の１つのみです。（CSSとJavaScriptをインラインで記述しているため）

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, PowerPoint

自動的に生成された説明

### 中継プログラムのファイル構成

* Lambda関数「local\_uploader\_sign\_url」
  + ***index.mjs*** … このLambda関数は、ユーザーがファイルをアップロードする直前に起動します。送信側が直接AWS S3にアップロードできるように、認証済みURLを発行します。
* Lambda関数「local\_uploader\_handle\_change」
  + ***index.mjs*** … このLambda関数は、S3にファイルがアップロードされたときに起動します。各ファイルの最終更新日時をJSONデータにまとめて、受信プログラムが閲覧できるように公開します。
* S3バケット「local-uploader-files」

→受け取ったファイルを保存するためのバケットです。

* S3バケット「local-uploader-metadatas」

→各ファイルの最終更新日時を保存するためのバケットです。以下のようなJSONデータが保存されています。

テキスト

自動的に生成された説明

### 受信プログラムのファイル構成

* ***forge.config.js*** … 実行可能ファイルに変換するための設定を記述しています。詳細は「Electron Forge」のホームページを参照してください。<https://www.electronforge.io/>
* ***package.json*** … npmパッケージの管理の為の情報を記述しています。
* ***package-lock.json*** … npmパッケージの管理の為の情報を記述しています。
* フォルダ「src」
  + ***main.js*** … 真っ先に呼び出される処理です。この処理が２つのスレッドを起動させます。２つのスレッドは同時並列に実行されます。
  + ***thread\_main.js*** … メインスレッド。GUIの処理を担当する。
  + ***thread\_sub.js*** … サブスレッド。ファイルの受信処理を担当する。バックグラウンド処理として働き、15秒に１度サーバーを監視する。
  + ***index.html*** … 受信側のGUI 。electronが表示します。
  + ***style.css*** … 受信側のGUI 。electronが表示します。
  + ***preload.js*** … 受信側のGUI 。electronが表示します。
  + ***get\_target\_folder.js*** … どこに保存すればいいかを取得する処理。
  + ***get\_device\_id.js*** … デバイスごとに固有のIDを取得する処理。

## 関数一覧

### 送信プログラムの関数一覧

* ファイル「index.html」
  + ***関数「window.onload」*** … ページが読み込まれた直後に実行する関数。入力フォームやドラッグ＆ドロップのエリアの初期設定を行う。
  + ***関数「uploadMultipleFiles」*** … 複数のファイルを一斉送信する関数。
  + ***関数「uploadFile」*** … ファイル１つを送信する関数。
  + ***関数「showLoader」*** … ローダー（アップロード中のグルグル）を表示する関数。
  + ***関数「deleteLoader」*** … ローダー（アップロード中のグルグル）を消す関数。

### 中継プログラムの関数一覧

* Lambda関数「local\_uploader\_sign\_url」

このLambda関数は、ユーザーがファイルをアップロードする直前に起動します。送信側が直接AWS S3にアップロードできるように、認証済みURLを発行します。

* + ファイル「index.mjs」
    - ***メイン関数*** … このLambda関数は、メイン関数のみで動作しています。
* Lambda関数「local\_uploader\_handle\_change」

このLambda関数は、S3にファイルがアップロードされたときに起動します。各ファイルの最終更新日時をJSONデータにまとめて、受信プログラムが閲覧できるように公開します。

* + ファイル「index.mjs」
    - ***メイン関数*** … アップロードされた全てのファイルについて処理を行います。for文で繰り返して、以下の関数「\_oneRecord」を呼び出します。
    - ***関数「\_oneRecord」***… アップロードされたファイルを１個ずつ処理します。
    - ***関数「\_streamToString」***… テキストファイルのデータを、文字列に変換します。

### 受信プログラムの関数一覧

* フォルダ「src」
  + ファイル「main.js」
    - ***メイン処理*** … ２つのスレッドを起動させる。２つのスレッドは同時並列に実行される。
  + ファイル「thread\_main.js」
    - ***メイン関数*** … メインスレッド。GUIの処理を担当します。
    - ***関数「createWindow」*** … GUIの操作画面を開く関数。
  + ファイル「thread\_sub.js」
    - ***メイン関数*** … サブスレッド。ファイルの受信処理を担当する。バックグラウンド処理として働き、15秒に１度サーバーを監視する。
    - ***関数「\_update」*** … サーバーを監視して、ファイルが追加されていたらダウンロードする。
    - ***関数「\_getOldMetaData」*** … ローカルコンピューターに保存されている、古いメタデータを取得する。
    - ***関数「\_getNewMetaData」*** … サーバーで管理されている、最新のメタデータを取得する。
    - ***関数「\_saveFile」*** … 任意のURLからバイナリファイルを取得して、ファイルとして保存する関数。
  + ファイル「preload.js」
    - ***イベント「DOMContentLoaded」*** … ページが読み込まれた直後に実行されるイベント。
    - ***イベント「selected-directory」*** … メインプロセスから送られたデータを受信したときに、実行されるイベント。
    - ***イベント「render-my-data」*** … メインプロセスから送られたデータを受信したときに、実行されるイベント。
  + ファイル「get\_target\_folder.js」
    - ***メイン関数*** … どこに保存すればいいかを取得する関数。
  + ファイル「get\_device\_id.js」

***メイン関数*** … デバイスごとに固有のIDを取得する関数。

## API一覧

### 認証済みURLの取得

* URL：<https://te7ambvdcpwzj7eqfo3w2x3k3m0tntan.lambda-url.ap-northeast-1.on.aws/>
* メソッド：POST
* 認証：不要
* クエリパラメータ：不要
* Content-Type：application/json
* リクエストボディ：以下のパラメータを持つJSONを送ってください。
  + deviceId：送信先のデバイスID。
  + fileName：今から送信したいファイルの名前。
  + fileType：今から送信したいファイルのMIMEタイプ。
* レスポンス：認証済みURL。プレーンテキスト形式。

### ファイルのアップロード

* URL：上記のAPIで取得した認証済みURLに対して、リクエストを送ってください。
* メソッド：PUT**（POSTではないので注意！）**
* クエリパラメータ：不要
* リクエストボディ：ファイル本体

### ファイルの最終更新日時の取得

* URL：https://local-uploader-metadatas.s3.ap-northeast-1.amazonaws.com/**<受信側のパソコンのデバイスID>**.json

例

<https://local-uploader-metadatas.s3.ap-northeast-1.amazonaws.com/444fa66f164ab73eb84a0185c3838c9c.json>

* メソッド：GET
* 認証：不要
* クエリパラメータ：不要
* レスポンス：以下のようなJSON。

テキスト

自動的に生成された説明

### ファイル本体のダウンロード

* URL：https://local-uploader-files.s3.ap-northeast-1.amazonaws.com/**<受信側のパソコンのデバイスID>** / **<ダウンロードしたいファイルの名前>**

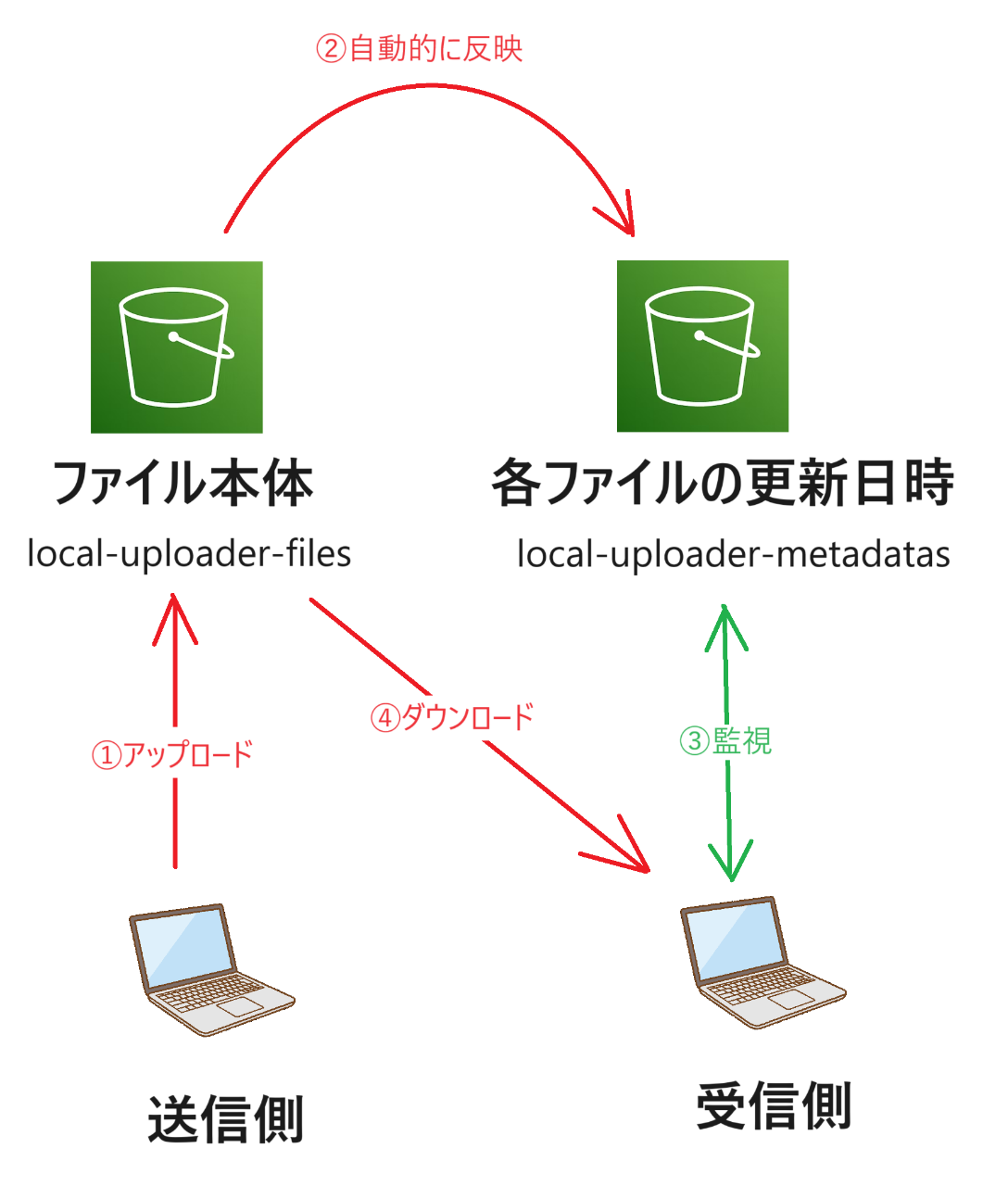
例

<https://local-uploader-files.s3.ap-northeast-1.amazonaws.com/444fa66f164ab73eb84a0185c3838c9c/sample.jpg>

* メソッド：GET
* 認証：不要
* クエリパラメータ：不要
* レスポンス：ファイル本体。

# 動作の流れ

## 大まかな動作の流れ



1. ファイルをS3バケット「local-uploader-files」にアップロードする。
2. サーバーがS3バケット「local-uploader-files」の変化を自動的に検知して、プログラムを起動する。プログラムは各ファイルの最終更新日時をJSONファイルにまとめて、S３バケット「local-uploader-metadatas」に保存する。
3. 受信側のパソコンは、15秒に１度、S３バケット「local-uploader-metadatas」を監視する。
4. 監視の結果、もし最終更新日時に変化があったら、ファイル本体を自動的にダウンロードする。

## 詳細な動作の流れ

### アップロードの流れ

S3へのアップロード処理を軽くするために、ダイレクトアップロード方式を採用した。流れは以下の通り。

1. 送信側プログラムがLambda関数「local\_uploader\_sign\_url」にリクエストを送る。
2. Lambda関数「local\_uploader\_sign\_url」が認証済みURLを発行して、送信側プログラムに返す。
3. 送信側プログラムが認証済みURLを使って、ファイルをダイレクトアップロードする。

### 最終更新日時を保存する流れ

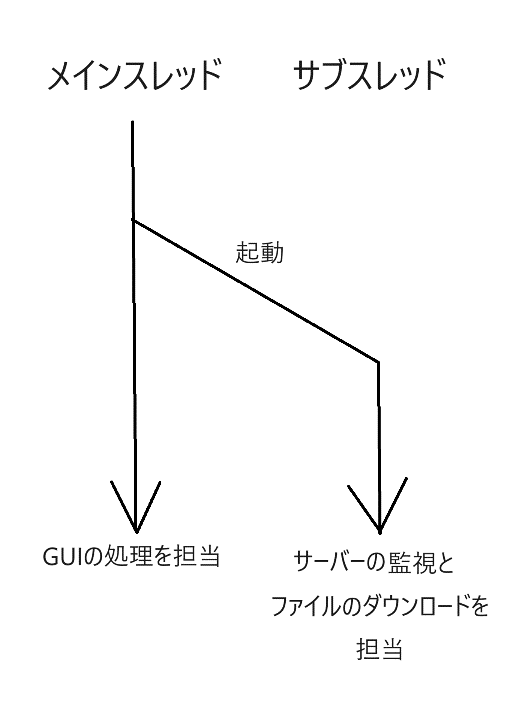
1. S3バケット「local-uploader-files」にファイルがアップロードされる。
2. Lambda関数のトリガーが発動し、Lambda関数「local\_uploader\_handle\_change」が実行される。
3. 各ファイルの最終更新日時を調べ、S3バケット「local-uploader-metadatas」に保存する。以下の例のようなJSONファイル形式で保存する。JSONのキーはファイル名、値はタイムスタンプ。

テキスト

自動的に生成された説明

### 受信プログラムの起動順序

受信プログラムでは、２つのスレッドを同時並列に実行します。



### 最終更新日時を監視する流れ

15分に一度、以下の処理を行う。（pooling）

1. 受信プログラムが、サーバーから「最新の最終更新日時」を取得する。
2. 受信プログラムが、ローカルコンピューターに保存されている「古い最終更新日時」を取得する。
3. 「最新の最終更新日時」と「古い最終更新日時」を比較する。

もし最終更新日時が変化していなかったら、「何もアップロードされなかった」とみなして、何もしない。

もし最終更新日時が変化していたら、「何かがアップロードされた」とみなして、ファイル本体をダウンロードする。

1. 「手順①で取得した最終更新日時」をローカルコンピューターに保存する。これが次回のループでの「古い最終更新日時」になる。ちょうどバケツリレーのようなイメージ。

# デバッグの方法

エラー情報などのログは、以下のテキストファイルに保存されます。

C:\Users\ <ユーザー名> \AppData\Roaming\share\_file\_client\logs

# 更新方法

GitHubからコードを自動送信するような、CI/CD等の設定は行っておりません。更新の際は、コードを手動で書き換える必要があります。

### 送信プログラムの更新方法

送信元パソコンで動作するwebページは、AWS S3に静的ファイルとして配置しています。配置場所は、バケット「local-uploader-metadatas」の直下です。専用のバケットを用意するのが勿体なかったため、メタデータ用のバケットに入れています。ファイルは ”index.html” の１つのみです。（CSSとJavaScriptをインラインで記述しているため）

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, PowerPoint

自動的に生成された説明

以下のURLからAWS S3の管理画面にアクセスすると、プログラムを更新できます。

<https://s3.console.aws.amazon.com/s3/buckets/local-uploader-metadatas?region=ap-northeast-1>

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション

自動的に生成された説明

### 中継プログラムの更新方法

Lambdaコンソールから直接書き換えてください。以下のURLから更新できます。

* Lambda関数「local\_uploader\_sign\_url」

<https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/functions/local_uploader_sign_url?fullscreen=true&tab=code>

* Lambda関数「local\_uploader\_handle\_change」

[https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/functions/local\_uploader\_handle\_change?fullscreen=true&tab=code](https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1%23/functions/local_uploader_handle_change?fullscreen=true&tab=code)

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, メール

自動的に生成された説明

### 受信プログラムの更新方法

受信プログラムはelectronで作っています。以下のコマンドを参考にしてください。

依存しているパッケージの一括インストール：npm install

テスト実行：npm start

実行可能ファイルの生成：npm run make

コンピューターのスクリーンショット

自動的に生成された説明

実行可能形式のプログラム（exeファイル）は、事務室内のパソコンにインストールしています。

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, メール

自動的に生成された説明

また、インターネットからexeファイルをダウンロードできるようにもしています。AWS S3のバケット「local-uploader-metadatas」の直下に、ファイル名「share\_file\_client.exe」で公開しています。専用のバケットを用意するのが勿体なかったため、メタデータ用のバケットに入れています。

コンピューターのスクリーンショット

自動的に生成された説明

プログラムを更新する際は、これらのexeファイルを更新してください。