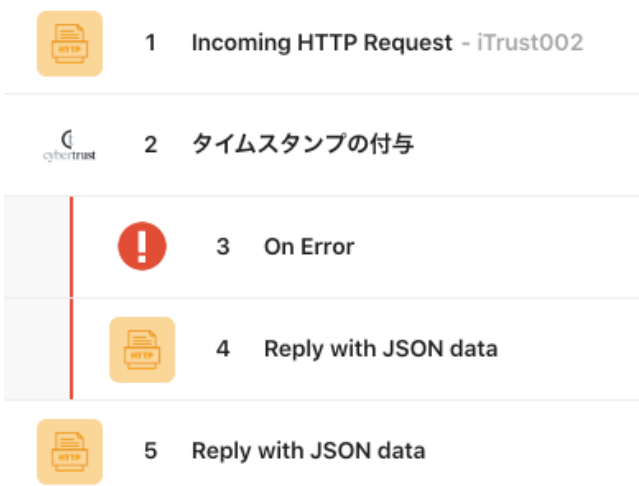


# Claris Connect iTrust タイムスタンプ フローの作成手順

1 Incoming HTTP Request (Webhookの作成)	1
2 タイムスタンプの付与	6
5 Reply with JSON data	13
3 On Error ( 4 Reply with JSON data)	16
エラー処理の設定	18
フローを有効化	19
API実行履歴を確認	20

下図がこれから作成するフローの全容となります。こちらのマニュアルでは、各ステップの設定方法を説明していきます。



## 1 Incoming HTTP Request (Webhookの作成)

Claris Connect にて左上の[ + ]ボタンを押し、[Create New] を選択します。

現在運用中 ...

My Projects Custom Connectors Templates

+

Create New

Create from Template

iTrust001

1 active 1 flow 6

Updated Nov 24, 2023 at 2:36 PM

kondo Document Test

1 flow 6

Updated Dec 5, 2023 at 11:06 AM

Nihongodame a12

1 flow 6

Updated Oct 18, 2023 at 4:34 PM

Untitled

1 flow 6

Updated Oct 18, 2023 at 4:26 PM

Untitled

2 flows 6

Updated Oct 18, 2023 at 4:50 PM

左上の [Webhooks] を押します。

← Untitled

Flows

Flows History App Connections

Approvals

Webhooks

Sharing

There are no flows in this project.

A flow is an automated workflow that connects two or more apps or utilities.

Create New Flow

画面真ん中の青いボタン [Create a webhook] ボタンを押します。

← Untitled

Flows

Webhooks

Approvals

Webhooks

Sharing

There are no webhooks in the project yet

Trigger flows using a webhook. [Learn More](#)

Create a webhook

名前は任意で設定し、[Create] ボタンを押します。

← Untitled

Flows

Approvals

Webhooks

Sharing

## Webhooks

Create new webhook

iTrust002

Create

Webhookの右端の下矢印アイコンを押して、プルダウンの中身を表示させます。

← Untitled

Flows

Approvals

Webhooks

Sharing

## Webhooks

Create new webhook

Enter the webhook name, which appears in the URL

Create

1 webhook



iTrust002

https://[redacted]/webhook/v1/itrust002/catch



## Project API Key

Copy the API key (which is project-wide) for authentication.

[Use authentication] と [Return sync response] をオンにしてください。

Webhooks

Enter the webhook name, which appears in the URL

Create

Sharing

1 webhook



iTrust002

https://[redacted]/webhook/v1/itrust002/catch



Use authentication

Require authentication for using the webhook.




Return sync response

Get a sync response when the webhook is used.

[Delete webhook](#)

下図の赤線の3つ、「Webhook の URL」「Project API Key」「AppID」をコピーし、FileMaker の [URLから挿入] ステップに付与する値として設定します。

1 webhook

 **iTrust002**  
https://[redacted]webhook/v1/itrust002/catch

☒ **Use authentication**  
Require authentication for using the webhook.

☒ **Return sync response**  
Get a sync response when the webhook is used.

[Delete webhook](#)

**Project API Key**  
Copy the API key (which is project-wide) for authentication.

[redacted]

Copy

Regenerate key

Use AppID:APIKey to authenticate. The AppID for the project [redacted]

Example cURL syntax: curl --user appid:apikey webhook\_url

```
6 # Webhook URL
7 変数を設定 [ $url ; 値: "https://[redacted].connect.claris.com/api/webhook/v1/itrust002/catch" ]
8 # Base64Encodeにある改行は不要なのでBase64EncodeRFCを使用する
9 変数を設定 [ $json ; 値: JSONSetElement ( "{}" ; "base64" ; Base64EncodeRFC ( 4648 ; iTrust::PDF ) ; 1 ) ]
10 # Claris Connect 側でのエラー処理を記述していない場合、最初のステップのWebhookが成功し、次のステップのiTrustへの送信でエラー
    て、最後の「Reply with JSON data」へ移行しないので、いつまで経っても切断されないので、最大時間の設定をお勧めします。
11 変数を設定 [ $最大時間 ; 値: 300 ]
12 変数を設定 [ $curlOption ; 値: "-X POST -S -m " & $最大時間 & " -H \"Content-type:application/json\" -d @$json" ]
13 # Webhook を「認証を使用」にした場合
14 変数を設定 [ $appid ; 値: [redacted] ]
15 変数を設定 [ $apikey ; 値: [redacted] ]
16 If [ not IsEmpty ( $appid ) and not IsEmpty ( $apikey ) ]
17     変数を設定 [ $curlOption ; 値: $curlOption & " --user " & $appid & ":" & $apikey ]
18 End If
```

左上の [Flows] を押します。

← Untitled

⌕ Flows

✔ Approvals

🔗 Webhooks


👥 Sharing

## Webhooks

Create new webhook

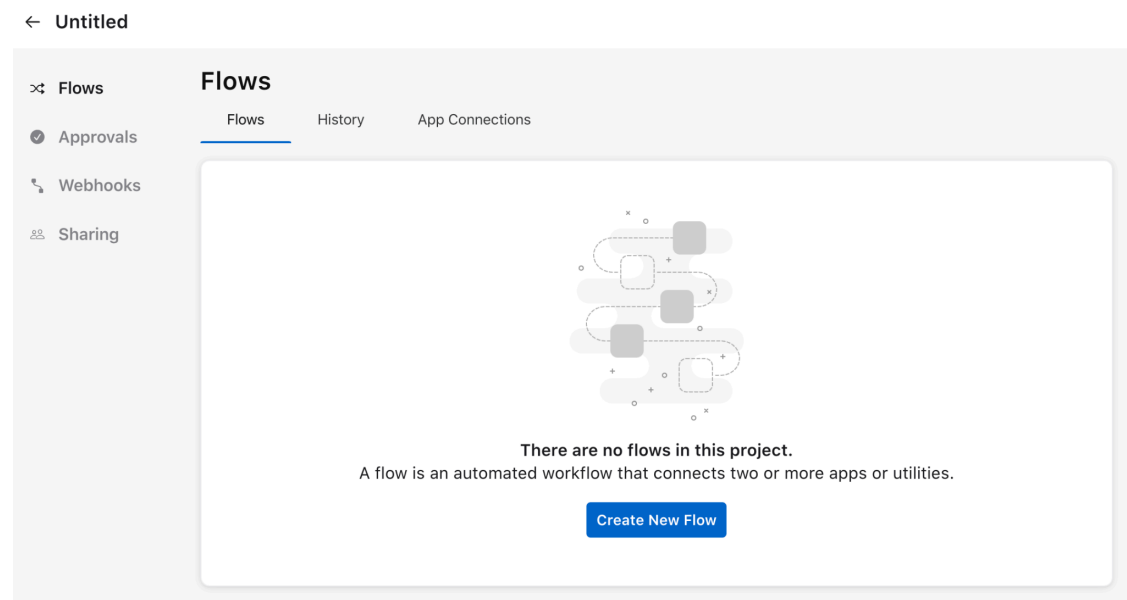
Create

1 webhook

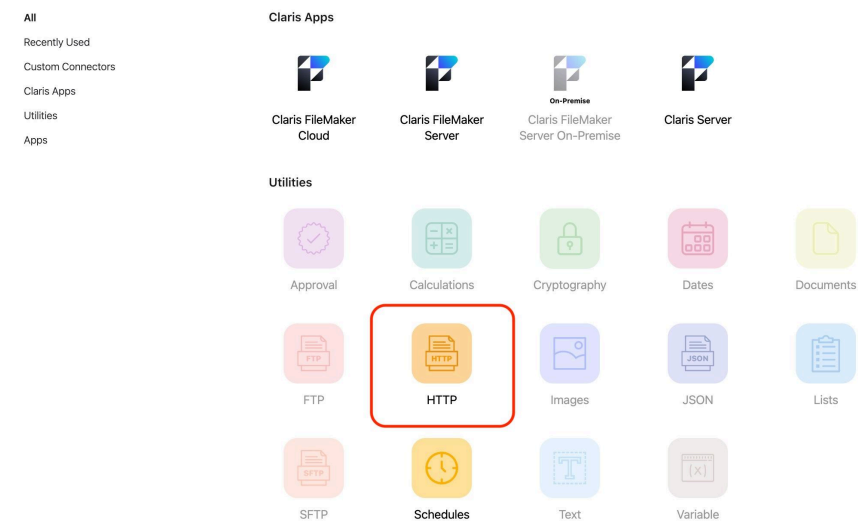
 **iTrust002**  
https://bccez3.apps.connect.claris.com/api/webhook/v1/itrust002/catch

**Project API Key**  
Copy the API key (which is project-wide) for authentication.

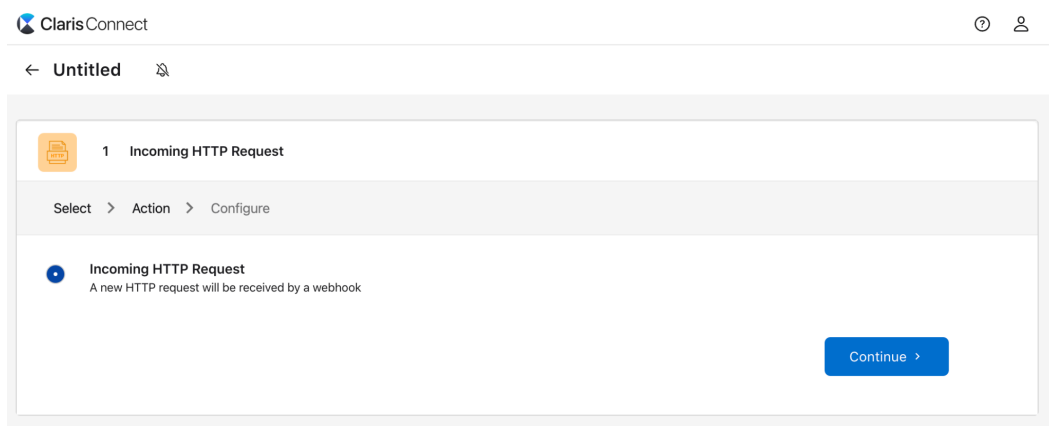
画面真ん中の [Create New Flow] を押します。



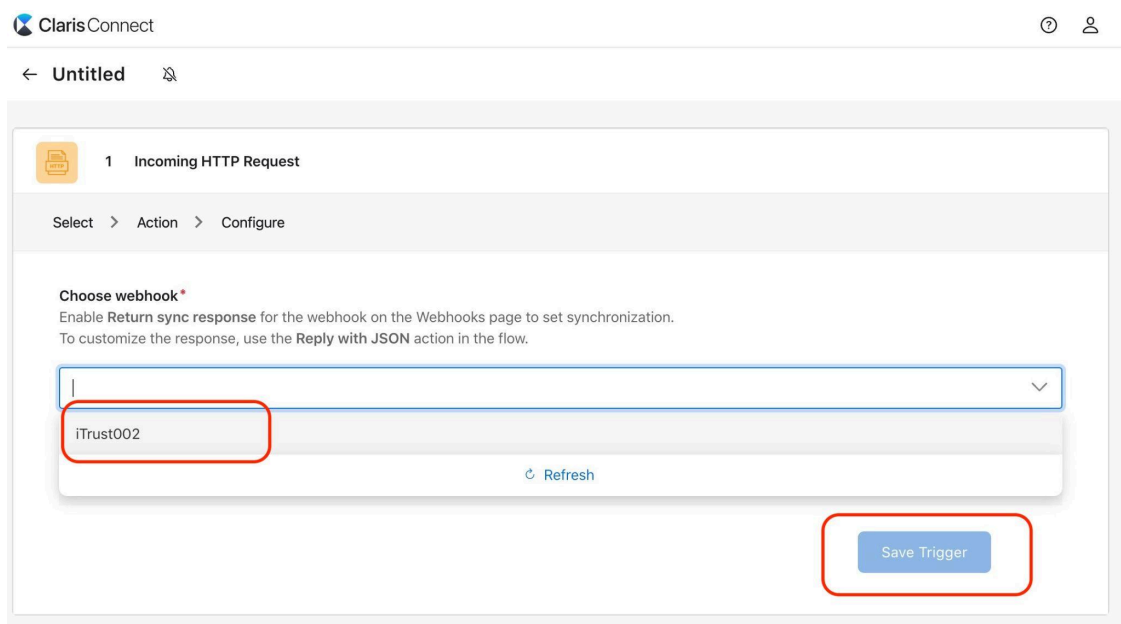
一覧から[HTTP]を選択します。



右下の [Continue] を押します。



プルダウンから先ほど作成した Webhook を選択します。この例では、[iTrust002] を選択します。ここではまだ [Save Trigger] がグレイアウトしています。

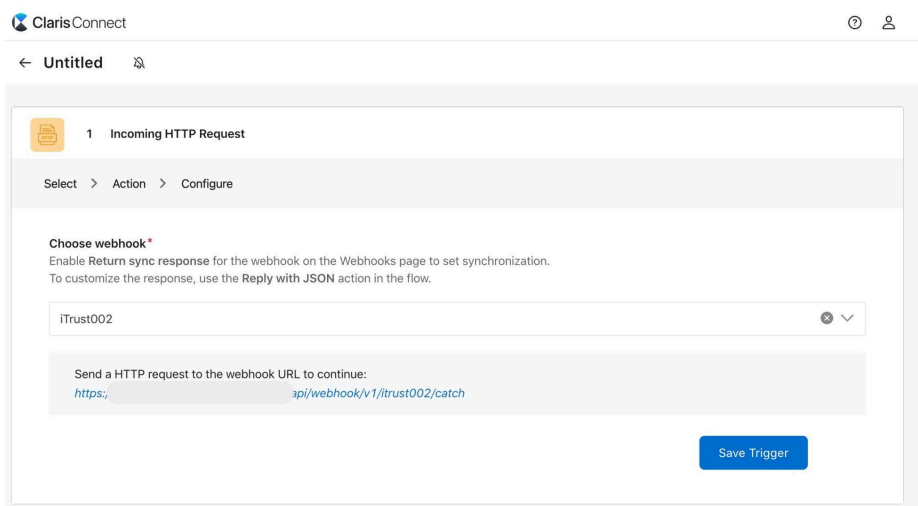


[SaveTrigger] ボタンを押せるようにするためには、Webhook の URL にリクエストを送る必要があります。Claris FileMaker のスクリプトにて、Base64エンコードした PDFデータをリクエストボディに入れて [URLから挿入] ステップを実行してください。

先程の「Webhook の URL」「Project API Key」「AppID」を Claris FileMaker スクリプトに 設定した状態で、リクエストボディに Base64エンコードした PDF データをセットします。[Base64EncodeRFC] 関数を使用して改行なしのエンコードテキストを生成する必要があります。

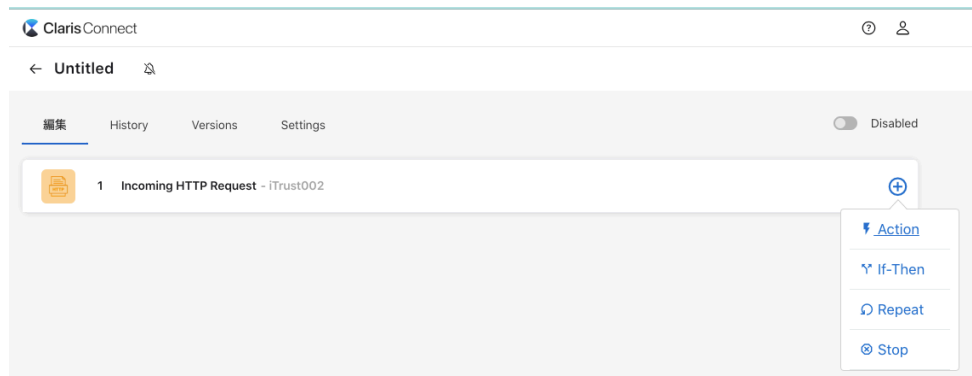
```
# Base64Encodeにある改行は不要なのでBase64EncodeRFCを使用する
変数を設定 [ $json ; 値: JSONSetElement ( "{}" ; "base64" ; Base64EncodeRFC ( 4648 ; iTrust::PDF ) ; 1 ) ]
```

Claris FileMaker から [URLから挿入] で Webhook に対してリクエストを送ると、[SaveTrigger] が押せるようになります。[SaveTrigger] を押します。

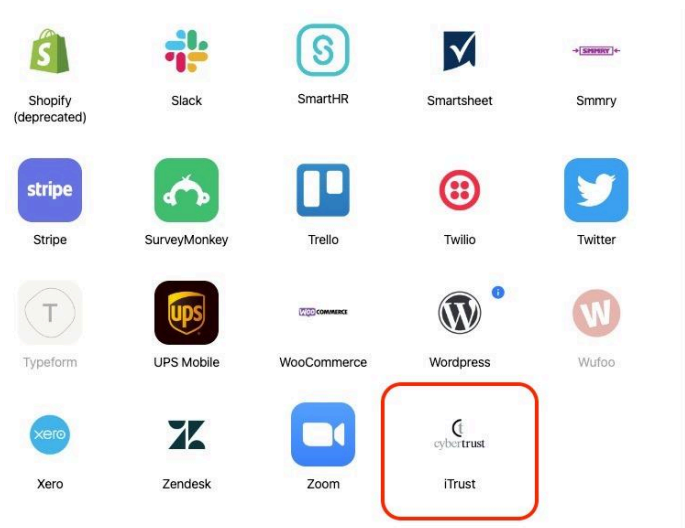


## 2 タイムスタンプの付与

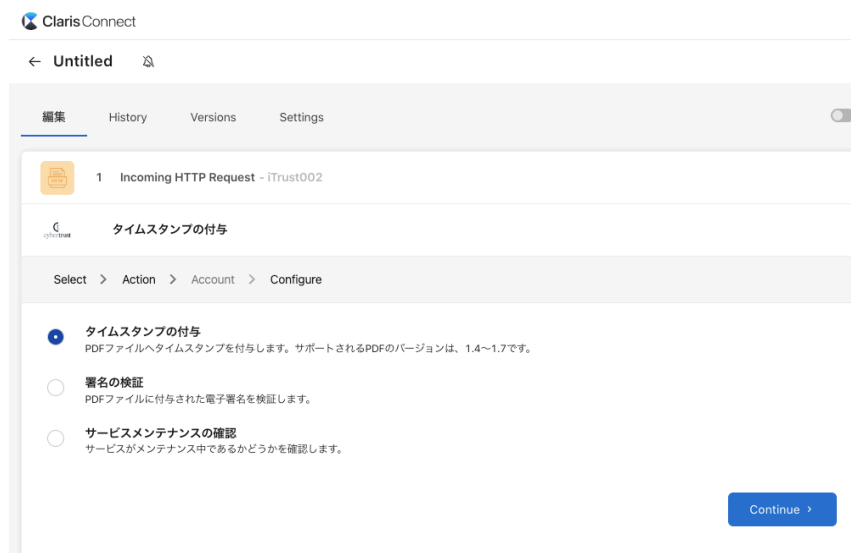
先程作成した「Incoming HTTP Request」の右端の [+] ボタンを押し、[Action] を押します。



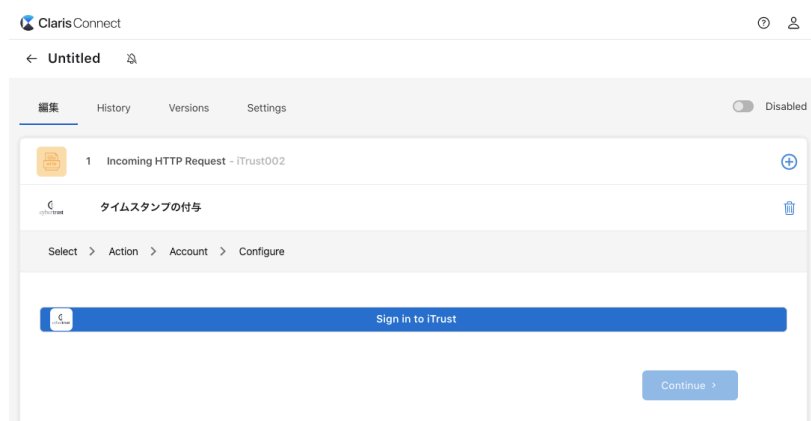
一覧から一番下にある [iTrust] を選択します。



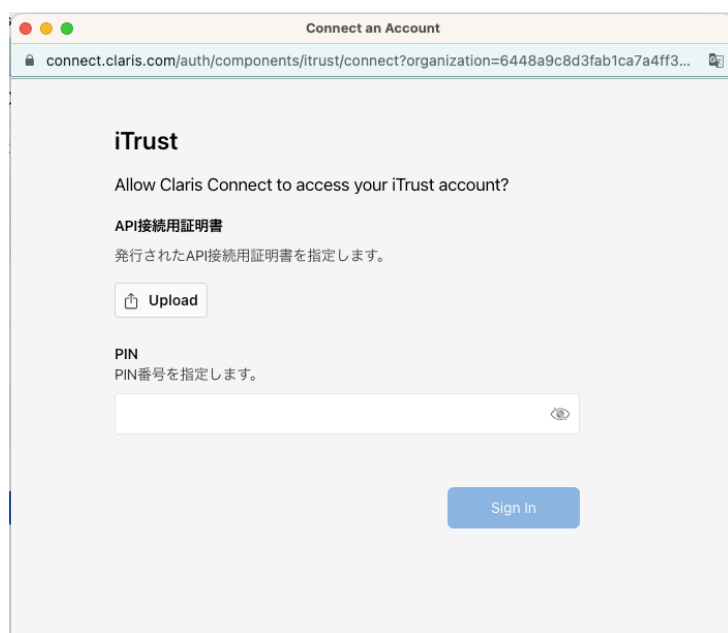
[タイムスタンプの付与] にチェックを入れていることを確認し、右下の[Continue]を押します。



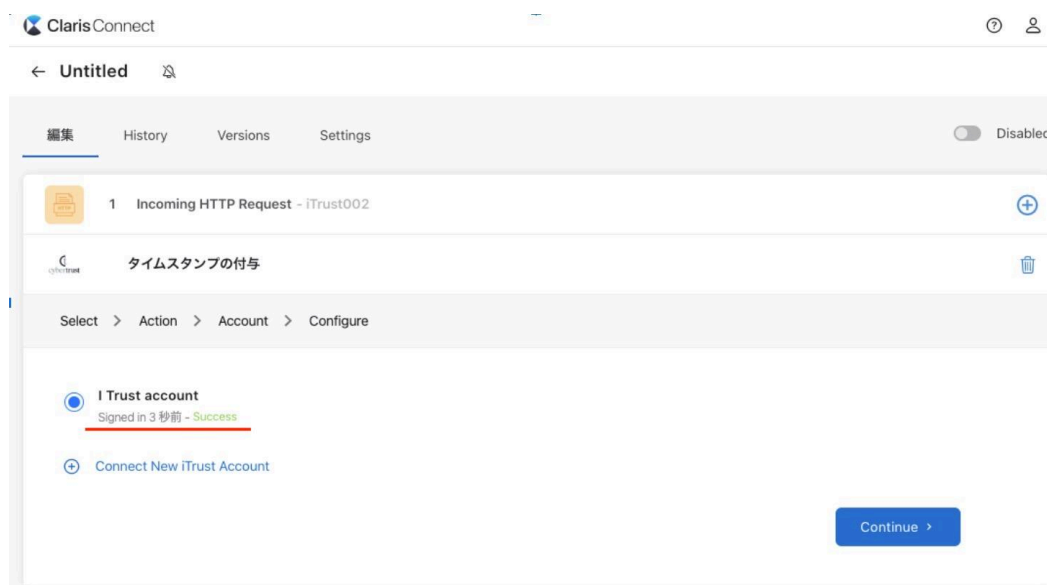
[Sign in to iTrust] を押します。



[Upload]ボタンから契約時に弊社からお渡ししている証明書をアップロードし、PIN番号を入力してください。その2つを設定できたら、右下の [Sign In] が押せる状態になります。



証明書のアップロードが成功したら、下図の赤線の箇所に "Success" と表示されます。右下の [Continue] を押します。



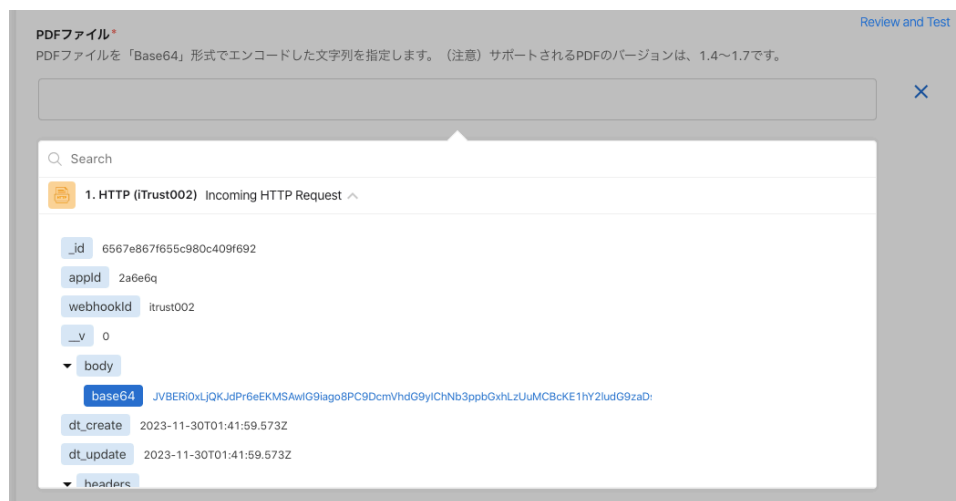
PDFファイル欄の右端にある [+]アイコンを押します。



下矢印を押してプルダウンを開いてください。



下図の "base64" をクリックします。



PDFファイルの欄に“base64”が下図のように設定されていることを確認します。



「タイムスタンプの用途」の右のプルダウンボタンを押し、[タイムスタンプを単一で使用]を選択します。



「電子署名およびタイムスタンプ情報の取得」は、取得したい場合は、[取得する]を選択してください。

#### タイムスタンプの用途\*

タイムスタンプの用途を指定します。

DocTimeStamp：標準的に利用するタイムスタンプの形式です。タイムスタンプが付与されているPDFファイルおよび未署名のPDFファイルに対して利用可能です。タイムスタンプが付与されているPDFファイルの有効期限の延長にはご利用いただけません。

ES\_A：長期署名規格（PADES）における有効期限延長にご利用可能です。長期署名形式（ES\_A）または既に付与されているタイムスタンプの有効期限の延長目的でタイムスタンプを付与する際に利用可能です。

タイムスタンプを単一で使用

#### 電子署名およびタイムスタンプ情報の取得

電子署名が付与されたPDFファイルに含まれる電子署名およびタイムスタンプの情報を取得するかどうかを指定します。

↑

🔍

取得しない

取得する

🔄 Clear current choice

Save

「スキャン文章に関連する情報の取得」を [取得する] にした場合、スキャナーを通したPDF であれば、その画像の解像度や、色がグレースケールかフルカラーかなどの情報をタイムスタンプの付与とともに取得できます。

#### 電子署名およびタイムスタンプ情報の取得

電子署名が付与されたPDFファイルに含まれる電子署名およびタイムスタンプの情報を取得するかどうかを指定します。

取得する

#### スキャン文書に関連する情報の取得

署名対象のPDFにファイルにおいて1ページ内の画像が1つでもある場合にスキャン文書とみなし、文書上の画像に関する解像度などの情報を取得するかどうかを指定します。

↑

🔍

取得しない

取得する

🔄 Clear current choice

最後に右下の [Save] を押して設定を保存します。

#### PDFファイル\*

PDFファイルを「Base64」形式でエンコードした文字列を指定します。（注意）サポートされるPDFのバージョンは、1.4～1.7です。

Step 1 body.base64

#### タイムスタンプの用途\*

タイムスタンプの用途を指定します。

DocTimeStamp：標準的に利用するタイムスタンプの形式です。タイムスタンプが付与されているPDFファイルおよび未署名のPDFファイルに対して利用可能です。タイムスタンプが付与されているPDFファイルの有効期限の延長にはご利用いただけません。

ES\_A：長期署名規格（PADES）における有効期限延長にご利用可能です。長期署名形式（ES\_A）または既に付与されているタイムスタンプの有効期限の延長目的でタイムスタンプを付与する際に利用可能です。

タイムスタンプを単一で使用

#### 電子署名およびタイムスタンプ情報の取得

電子署名が付与されたPDFファイルに含まれる電子署名およびタイムスタンプの情報を取得するかどうかを指定します。

取得する

#### スキャン文書に関連する情報の取得

署名対象のPDFにファイルにおいて1ページ内の画像が1つでもある場合にスキャン文書とみなし、文書上の画像に関する解像度などの情報を取得するかどうかを指定します。

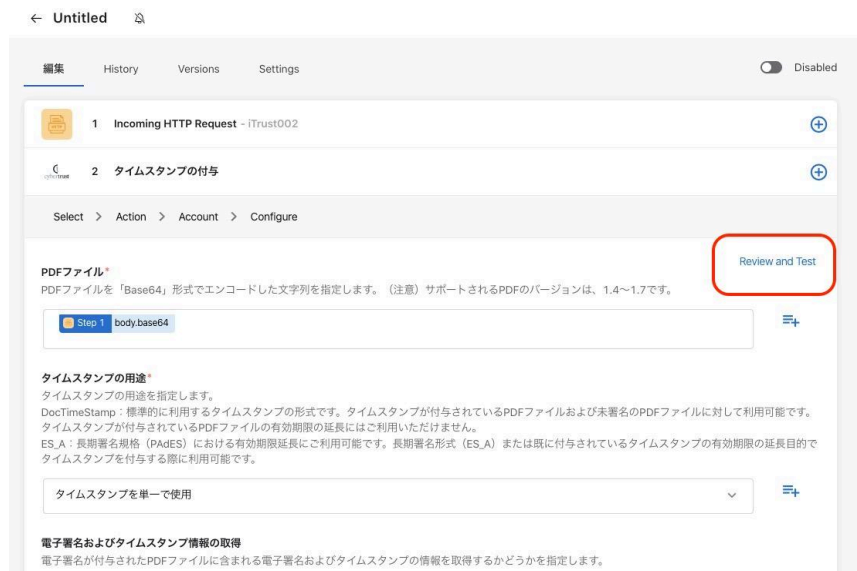
取得する

Save

上記で [Save] を押すとこのステップ自体の保存はできますが、実際にこのステップが成功するか確認する必要があります。先程作成した [タイムスタンプの付与] をクリックします。



[Review and Test] を押します。



[Review and Test] ボタンを押して画面を切り替えると、送ろうとしている実際のエンコードされた PDFデータなどが確認できます。

Select > Action > Account > Configure

[Edit Configuration](#)

## PDFファイル\*

PDFファイルを「Base64」形式でエンコードした文字列を指定します。（注意）サポートされるPDFのバージョンは、1.4～1.7です。

JVBERi0xLjQKJdPr6eEKMSAwIG9lago8PC9DcmVhdG9yIChNb3ppbGxhZUuMCBCKE1hY2ludG9zaDsgSW50ZWwgTWFjIE9TIFggMTBfMTVfN1wplEFwcGxlV2

## タイムスタンプの用途\*

タイムスタンプの用途を指定します。

DocTimeStamp: 標準的に利用するタイムスタンプの形式です。タイムスタンプが付与されているPDFファイルおよび未署名のPDFファイルに対して利用可能です。タイムスタンプが付与されているPDFファイルの有効期限の延長にはご利用いただけません。

ES\_A: 長期署名規格 (PaDES) における有効期限延長にご利用可能です。長期署名形式 (ES\_A) または既に付与されているタイムスタンプの有効期限の延長目的でタイムスタンプを付与する際に利用可能です。

DocTimeStamp

## 電子署名およびタイムスタンプ情報の取得

電子署名が付与されたPDFファイルに含まれる電子署名およびタイムスタンプの情報を取得するかどうかを指定します。

true

## スキャン文書に関連する情報の取得

署名対象のPDFにファイルにおいて1ページ内の画像が1つでもある場合にスキャン文書とみなし、文書上の画像に関する解像度などの情報を取得するかどうかを指定します。

true

画面を一番下までスクロールし、右下の [Test Action] を押します。

## タイムスタンプの用途

タイムスタンプの用途を指定します。

DocTimeStamp：標準的に利用するタイムスタンプの形式です。タイムスタンプが付与されているPDFファイルおよび未署名のPDFファイルに対して利用可能です。タイムスタンプが付与されているPDFファイルの有効期限の延長にご利用いただけませ。

ES\_A：長期署名規格 (PaDS) における有効期限延長にご利用可能です。長期署名形式 (ES\_A) または既に付与されているタイムスタンプの有効期限の延長目的でタイムスタンプを付与する際にご利用可能です。

DocTimeStamp

## 電子署名およびタイムスタンプ情報の取得

電子署名が付与されたPDFファイルに含まれる電子署名およびタイムスタンプの情報を取得するかどうかを指定します。

true

## スキャン文書に関連する情報の取得

署名対象のPDFにファイルにおいて1ページ内の画像が1つでもある場合にスキャン文書とみなし、文書上の画像に関する解像度などの情報を取得するかどうかを指定します。

true

[Test Action] を実行すると、数秒経過した後に、iTrust API が実際に返してくれるデータの中身を見ることができます。

“file” というキーの中に PDF データがエンコードされた状態で返ってきているのが確認できます。この [Test Action] を行わないと、次のステップで Save ができないため、必ず行う必要があります。

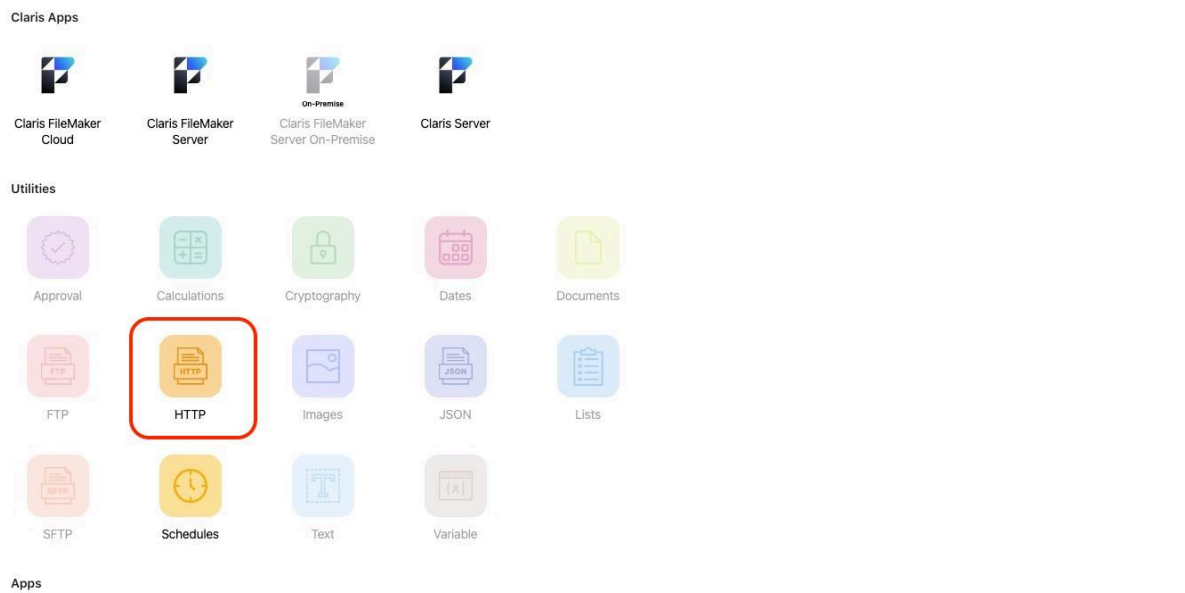
[illegible]

## 5 Reply with JSON data

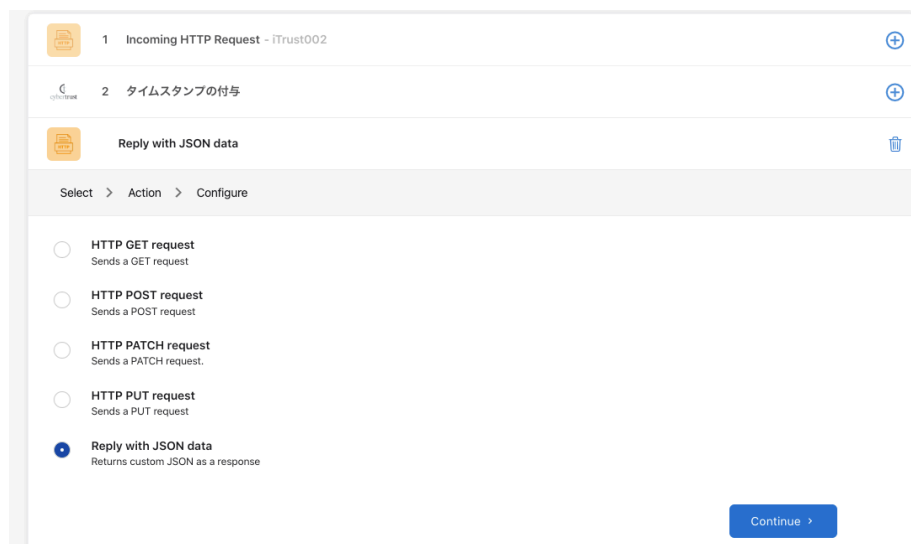
「タイムスタンプの付与」の右端の [+] ボタンを押し、[Action] を選択します。



一覧から[HTTP] を選択します。



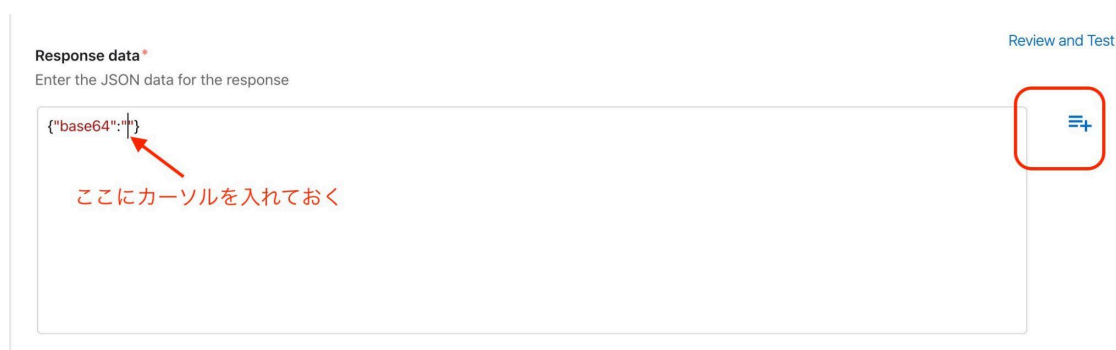
[Reply with JSON data] にチェックを入れ、[Continue] ボタンを押します。



レスポンスデータは JSON形式 で返したいので、{}(波括弧)でくくります。今回は、Clars FileMaker でレスポンスを受け取る時に、キーの名前を“base64”にしているので、こちらの設定でも “base64” と入力しています。



この "base64" というキーに対して PDFデータを入れたいので、下図のようにダブルクォーテーションの間にカーソルが入った状態で、右横の [+] ボタンを押します。



「2. iTrust タイムスタンプの付与」のプルダウンを開きます。

**Response data\***  
Enter the JSON data for the response

```
{ "base64": "" }
```

Search

- 1. HTTP (iTrust002) Incoming HTTP Request ▾
- 2. iTrust タイムスタンプの付与 ▾**

「2. iTrust タイムスタンプの付与」のプルダウンを開くと、iTrust API から受け取った値が表示されます。"file" をクリックします。この "file" が PDFデータになります。

Search

- 1. HTTP (iTrust002) Incoming HTTP Request ▾
- 2. iTrust タイムスタンプの付与 へ**

**file** JVBERi0xLjQKJdPr6eEKMSAwIG9iago8PC9DcmVhdG9yIChNb3ppbGxhLzUuMkCBcKE1hY2ludG9zaDsgSW...

signature-id 20231130112530C0RO0TGU00GJIM2R

signature-property

- time-stamp [List]
- 0
  - field-name Signature1
  - issuer-dn CN=SECOM TimeStamping CA3, O="SECOM Trust Systems CO.,LTD.", C=JP
  - subject-dn CN=AMANO-TSU-T2P3-1, OU=nShield TSS ESN:C5DD-8453-6DA7, OU=e-timing TSA, ..

下図のように、"base64" キーの値としてJSON形式で設定します。入力された値は、ダブルコーテーションで囲まれた中に入っていることをよく確認してください。

## Response data\*

Enter the JSON data for the response

```
{ "base64": "Step 2 file" }
```

PDFデータ以外にも返せるデータを選択することができます。例えば、"Signature-id" という値や、スキャン情報を返してくれる "scan-doc-property" など、取得したい値を設定することができます。

全ての キー と バリュー の両方に "ダブルコーテーション" を付けるのを忘れないようにしてください。

Select > Action > Configure

Response data\* Review and Test

Enter the JSON data for the response

```
{ "base64": "Step 2 file", "signatureId": "Step 2 signature-id", "scanDocProperty": "Step 2 scan-doc-property" }
```

成功した時のステータスコードを選択します。ここでは [200 - OK] を選択します。[Save] を押すとこのステップは保存されます。

Status code

Choose the status code to send in response

200 - OK

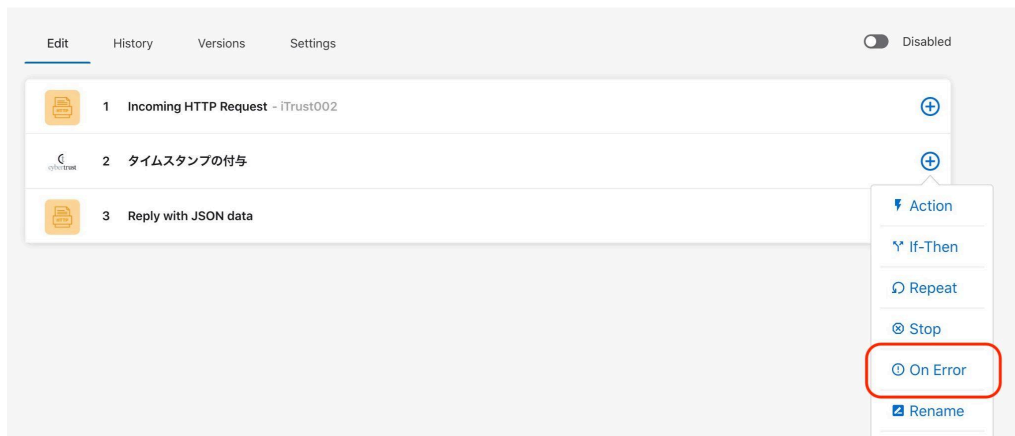
+

Save

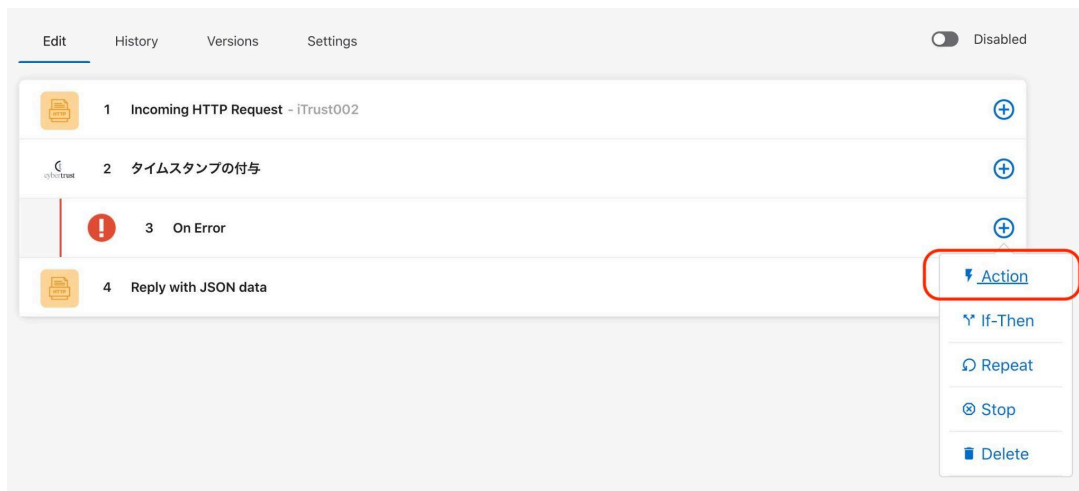
## 3 On Error ( 4 Reply with JSON data)

このステップでは、iTrust API からのレスポンスがエラーだった場合のステップを作成します。

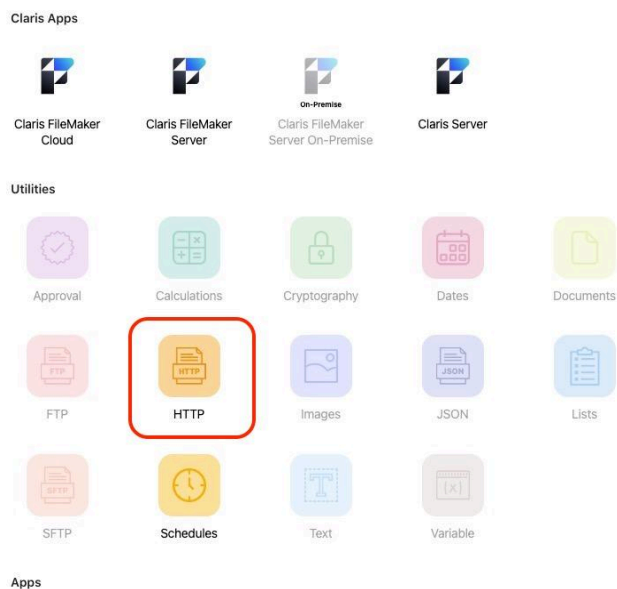
「タイムスタンプの付与」ステップ横の[+]ボタンを押し、[On Error] を選択します。



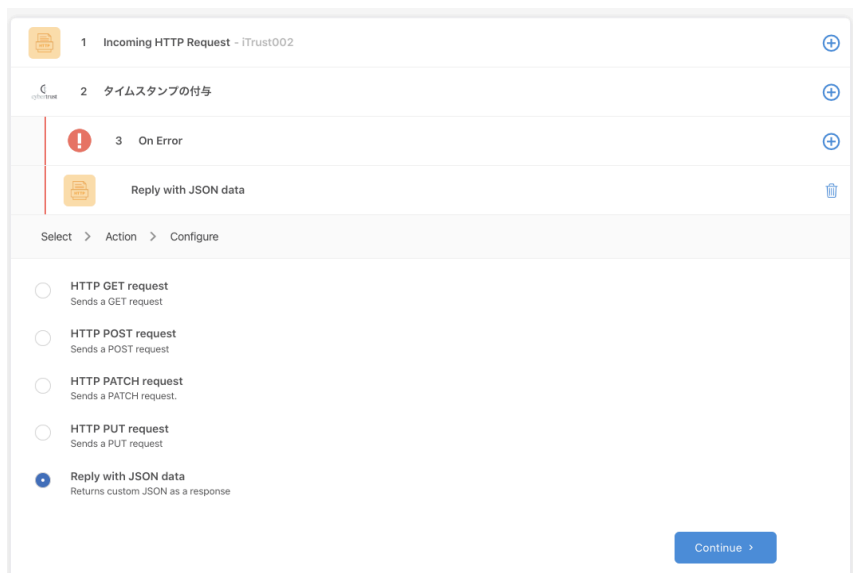
「On Error」ステップが新たに作成されるので、「On Error」ステップの右端の [+] ボタンを押し、[Action] を押します。



一覧から [HTTP]を選択します。



[Reply with JSON data] にチェックを入れた状態で、[Continue] を押します。



ここではエラー時に返す JSONデータを作成します。例えば、"base64" は空、"error" は "true" を設定しておきます。

## Response data \*

Enter the JSON data for the response

```
{ "base64": "", "error": true }
```

ステータスコードはレスポンスエラー400番台なので、ここでは「400 - Bad Request」にしておきます。[Save] を押して設定を保存します。

### Status code

Choose the status code to send in response

400 - Bad Request



Save

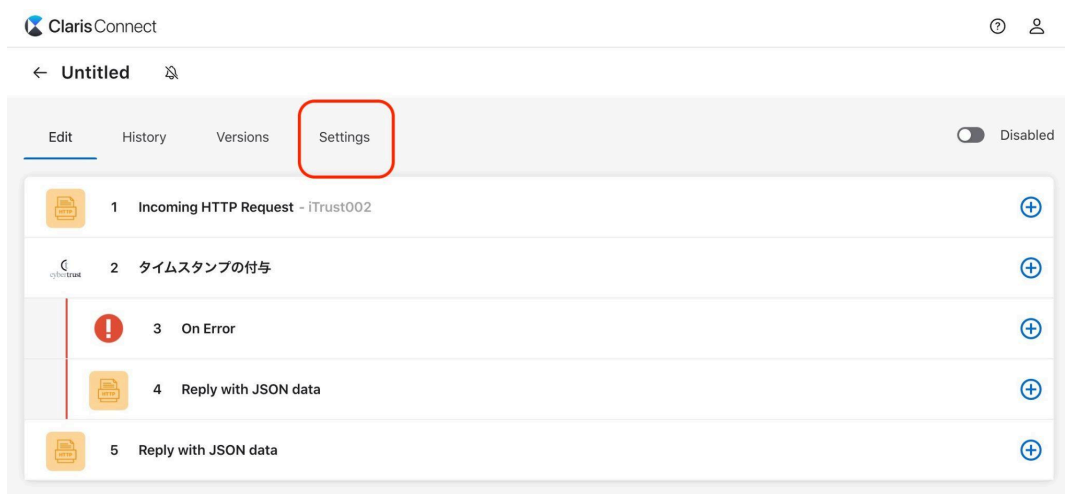
[URL から挿入] の cURLオプションに、-S を付けておくことをおすすめします。そうすることで、iTrust API の結果がエラーの場合、そのステータスコードを返すことができます（先程の「On Error」で設定したレスポンスの "error" が True か False かは関係ありません）。最終エラーが "0" ではない値を取得できるようになるため、エラーハンドリングに役立ちます。



## エラー処理の設定

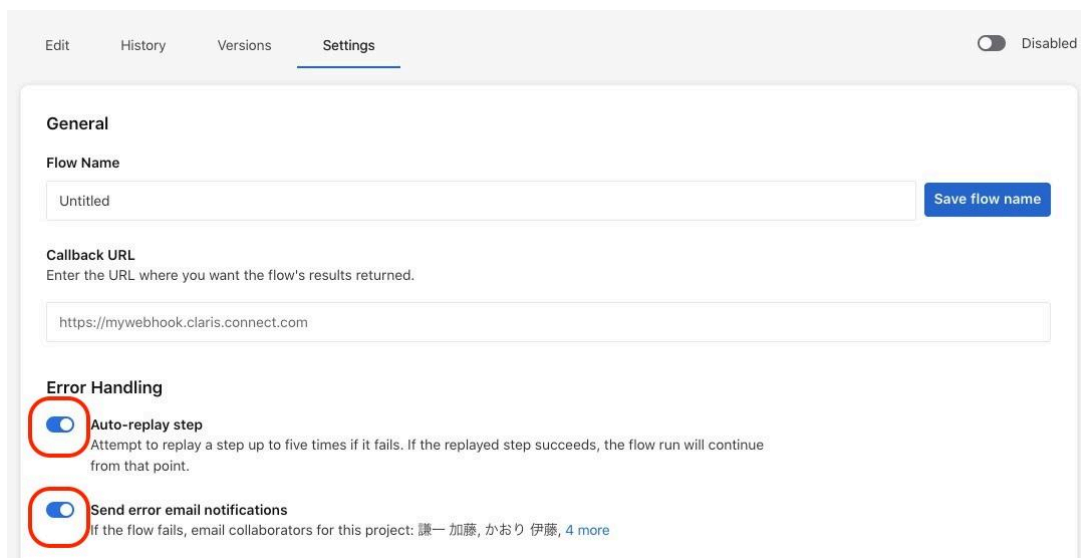
Claris Connect の設定で、API からのレスポンスがエラーだった場合に、最大5回まで再試行する設定と、エラーメール通知を送信する設定ができます。こちらの設定を行うことを推奨いたします。

[Setting] を押します。



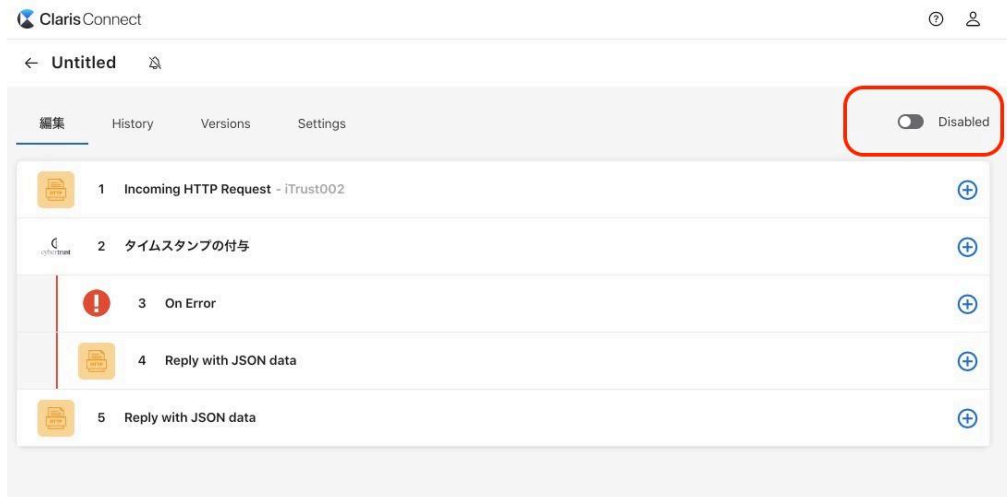
[Auto-replay step] と [Auto-replay step] のタブをオンにします。

※ 右上の [Disabled] がオンになっていると [Auto-replay step] をオンオフできなくなります。

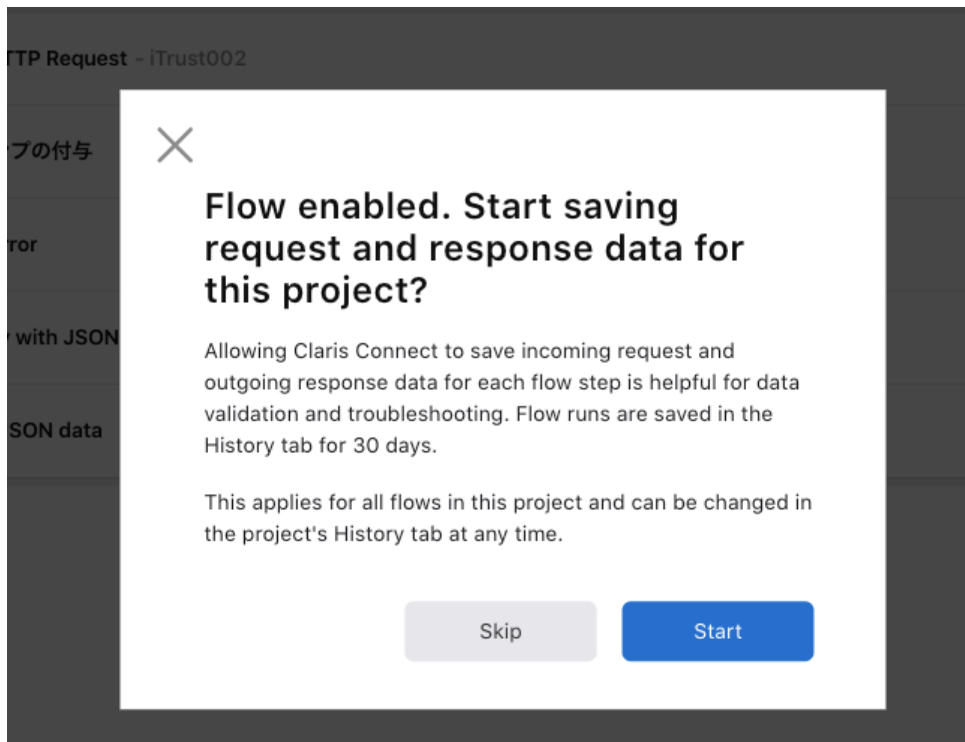


## フローを有効化

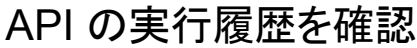
全ての設定を完了後、右上のボタン [Disabled] をオンにし、フローを有効化します。



有効化した後に表示されるダイアログは [start] を押します。



[Enabled] に変更されていることを確認します。



ClarisConnect

?

←

Untitled

編集

History

Versions

Settings

▶

今すぐ実行

Enabled

Status

All

Search

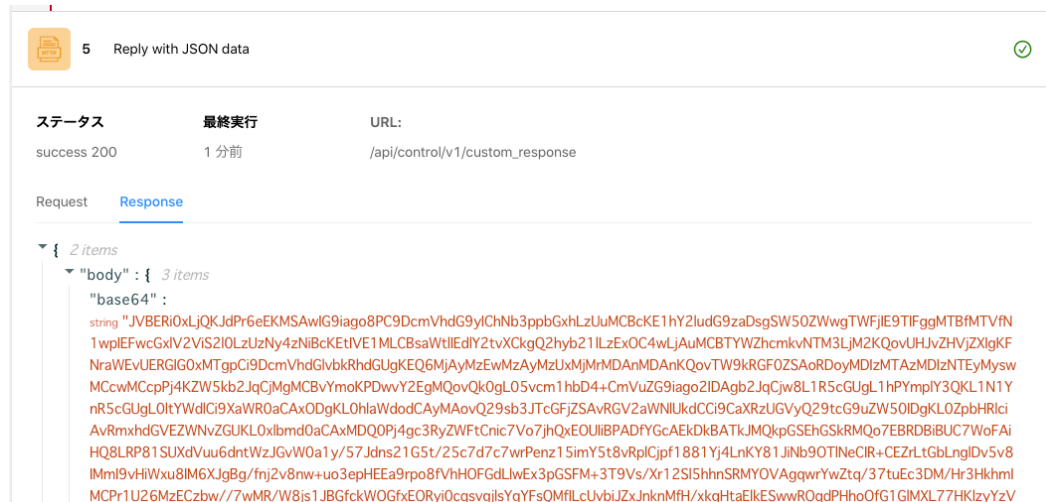
Status	Version	Name	Ran	Additional Information
<div>●</div> <div>Success</div>	8	Untitled	16 秒前	<div> <div></div> <div></div> </div>

<

1 of 1 pages

>

履歴からリクエストやレスポンスの中身を確認することができます。



設定手順の説明は以上となります。