

# *SmartRead*

## SmartRead API v3 概要

2024/08/16

株式会社 Cogent Labs



COGENT  
LABS

SHAPING THE FUTURE  
WITH ARTIFICIAL INTELLIGENCE

## 本資料について

本資料は、2024年7月22日時点の「SmartRead API v3」の概要紹介資料です。  
API ドキュメントの補足資料としてご参照ください。

以下の留意点をあらかじめご確認ください。

- ・ 「API ドキュメント」と内容について差分がある場合、「API ドキュメント」の内容を正として使用してください。
- ・ SmartRead On-Premises 固有の管理系APIについては紹介していません。  
マニュアル「SmartRead On-Premises API ご利用ガイド」を参照ください。
- ・ **「SmartRead非定型文書強化バージョン」用API（v4）は紹介していません。**

# 目次

- 01 SmartRead API とは
- 02 事前準備
- 03 SmartRead API v2 使用時の留意点
- 04 定型文書の読み取り処理の実行例（Shell+Curl）
- 05 読み取り処理の自動化
- 参考 用語の解説

# 01 SmartRead API とは

# SmartRead API の目的



## 連携

APIを利用して業務アプリケーションへの機能組み込みや、コネクタによるRPA連携などにより、文書データ化後の処理を自動化できます。



APIを通して  
業務アプリケーションと連携

SmartReadではWeb APIを提供しています。APIを使うことで、お客様が使用されているアプリケーションからシームレスにSmartReadを呼び出して業務に組み込むことができます。



RPA製品と連携して  
業務をさらに効率化

RPA製品からSmartReadを呼び出せるコネクタ部品を無償で提供。この部品によりRPAのシナリオから文書のデータ化処理を自動化できます。

無償コネクタ提供RPA: [UiPath](#)、[BizRobo!](#)、[WinActor](#)

SmartRead Webページより



SDKは提供していません。

# SmartRead API のバージョン

- APIの種類
  - REST API
- バージョン
  - Version 4
    - 「SmartRead非定型文書強化バージョン」で使用可能
  - Version 3
    - 「SmartRead」で使用可能
    - サポート終了予定無し
  - Version 2
    - 「Tegaki」、「SmartRead」で使用可能
    - サポート終了予定無し

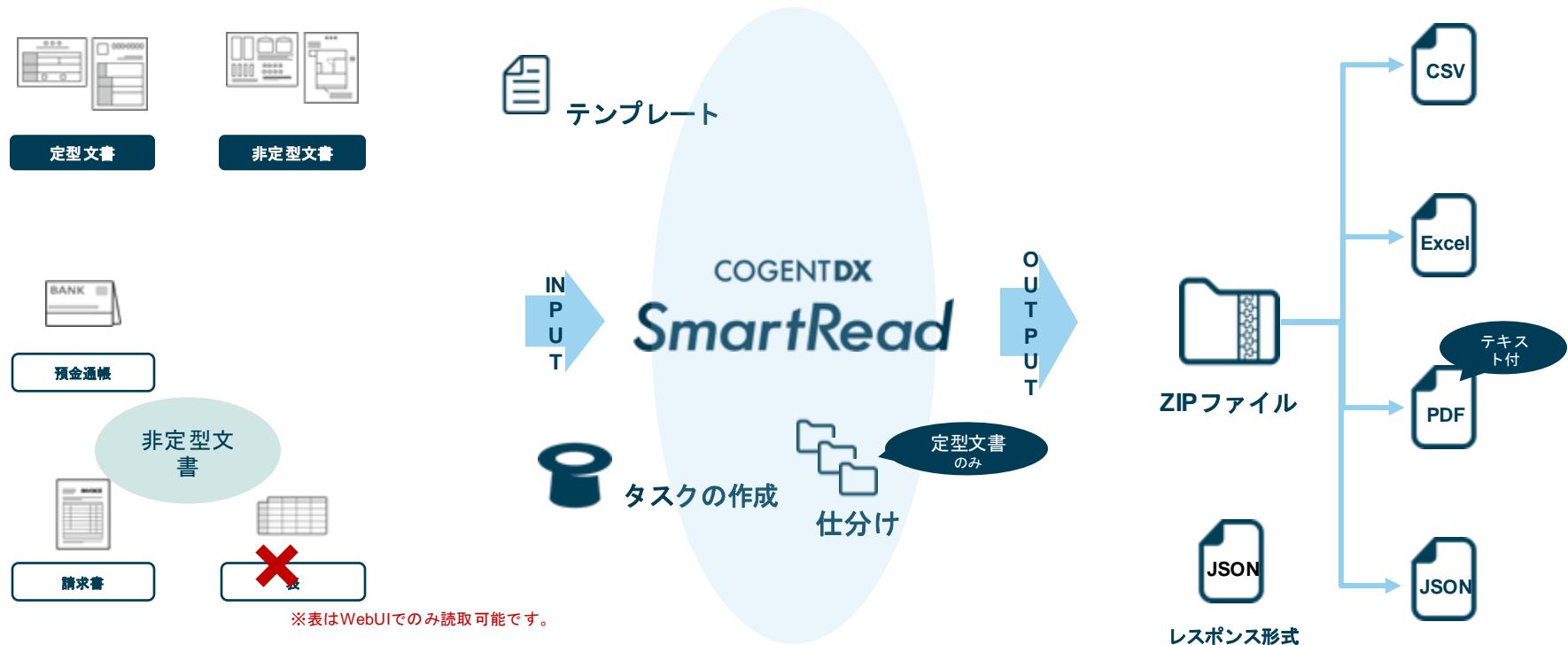


「REST API」が呼び出せる開発言語であれば、  
プログラム開発は可能です。

# 製品と使用できるAPIバージョン

製品	v2	v3	v4
SmartRead非定型文書強化バージョン			<input type="radio"/>
SmartRead	<input type="radio"/>	<input type="radio"/>	
Tegaki	<input type="radio"/>		

# SmartRead API 全体イメージ



※表はWebUIでのみ読み取可能です。



項目（フィールド）

※項目を読み取る（フィールドAPIの利用）場合は、API v2 を使用してください。

# SmartRead API 処理概要

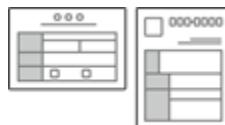
- 基本動作

- 「タスク」と呼ばれる業務の単位を作成し、その「タスク」に対し、「**1リクエスト=1ファイル**」で「読み取対象文書」をSmartReadにPOSTします。
  - 定型文書を読み取る場合、事前にテンプレートの作成が必要になり、「タスク」にそのテンプレート（テンプレートID）の指定が必要になります。
- その後、出力処理を実行し、結果をダウンロードします。

- フィールドAPI

- 「読み取対象文書」そのものをPOSTするのではなく、項目画像（文書上に書かれた読み取項目だけを画像抽出したもの）をPOSTすることも可能です。
  - その際、使用するAPIのことを「**フィールドAPI**」と呼びます。
  - 「**フィールドAPI**」は、「**SmartRead API v3**」で提供されておらず、**「SmartRead API v2」でのみ動作します。**
- その後、読み取結果をResponseのJSON形式でGETします。

文書ファイルそのもの



文書上に書かれた読み取項目だけを  
画像抽出したもの

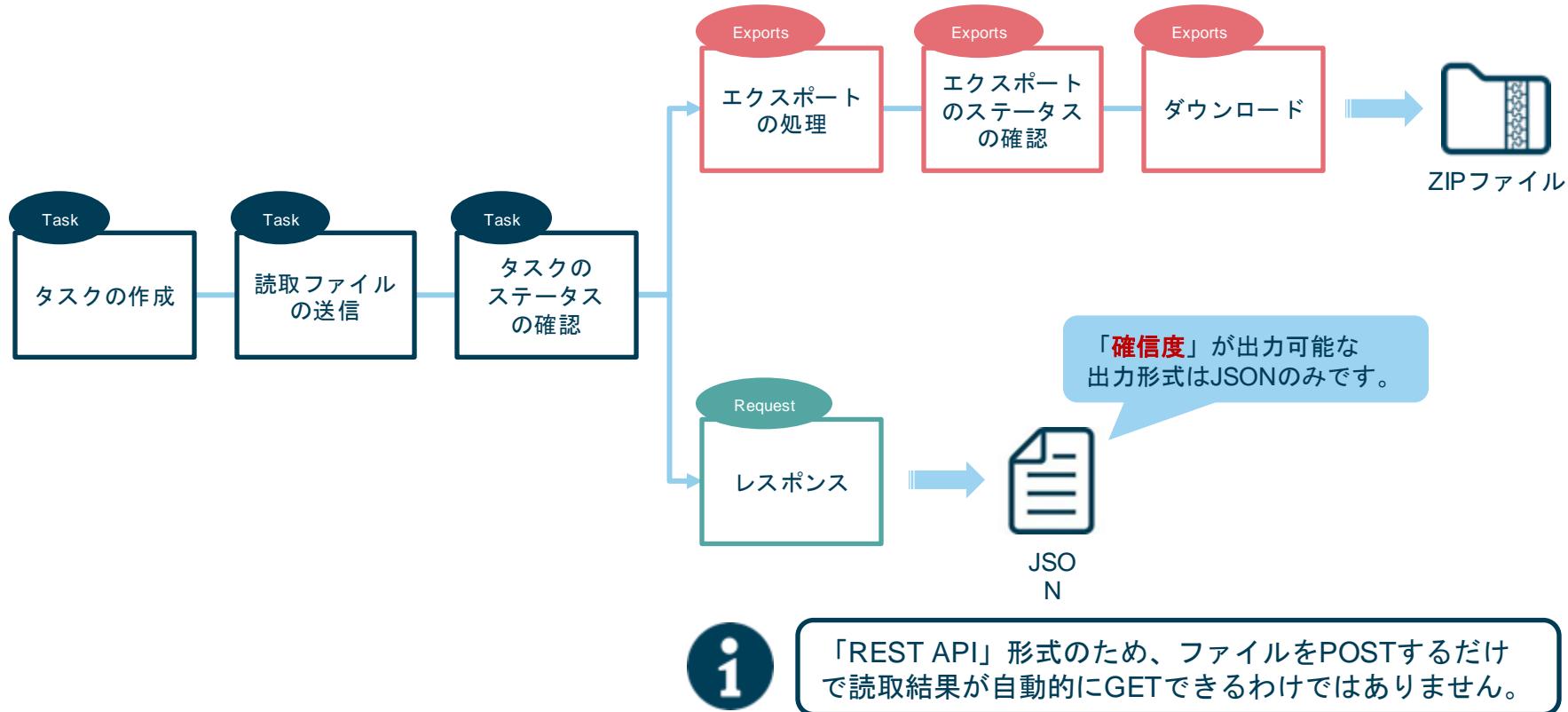


# 読み取対象文書と使用する SmartRead API のバージョン



文書種類	内容	テンプレート	API v2	仕分け (API v3)	API v3
定型文書	1ファイル (1ページ)	必要	○	-	○
	マルチページ ファイル	必要	×	○	○
非定型文書	預金通帳	不要	×	-	○
	請求書	不要	×	-	○
全文読み取り	1ファイル (1ページ)	不要	×	-	○
	マルチページ ファイル	不要	×	-	○
その他	表	不要	×	-	×
	項目画像	不要	○	-	×

# 読み取り処理の基本的な SmartRead API v3 処理の流れ



# SmartRead API v3 で提供される主な機能 1/2

API	機能	内容
Task	タスクを作成する	空のタスクを作成します。読み取り処理には最初に必ず作成する必要があります。
	タスクの一覧を取得する	所属するグループのタスクの一覧を取得します。
	タスクの詳細を取得する	タスクの詳細と、処理中のリクエストおよびファイルのサマリーを取得します。 Export API を実行する前に必ず実行する必要があります。
	タスクを更新する	タスクの名前、詳細、ラベルを更新します。 既存のタスク情報を更新したい場合に使用します。
	タスクを削除する	指定したタスクIDのタスクを削除します。
	タスクにファイルを送信する	複数ファイル一括送信はできません。1ファイルずつ送信する必要があります。 ファイルを送信するとリクエストIDが発行されます。
	タスクに含まれるリクエストをリストする	指定したタスクIDに含まれるリクエスト（ファイル）をリストします。
	出力の設定を表示	タスクで設定された出力形式を表示します。
	出力の設定を更新	タスクで設定された出力形式を更新します。 既存のタスク情報を更新したい場合に使用します。

# SmartRead API v3 で提供される主な機能 2/2

API	機能	内容
Export	タスクから出力を作成する	処理結果を出力します。 この時点で、処理結果はSmartRead上に存在しています。
	出力の詳細を取得する	Exportのステータスを取得します。
	出力結果をダウンロードする	出力結果をZIPファイル形式でダウンロードします。 ダウンロード後、ZIPファイルの解凍が必要です。
Request	読み取ステータスもしくは読み取結果を取得	指定したリクエストIDのリクエストの詳細を取得します。
	リクエスト結果を取得する	処理結果をJSON形式で取得します。
Template	テンプレートをSmartReadに登録する	テンプレートを登録します。 登録はWebUIでも可能です。
	テンプレートデータを更新する	テンプレートのデータを更新します。
	テンプレートの詳細入手する	テンプレートのメタデータとフィールド情報を取得します。
	テンプレートを削除する	指定したテンプレートIDのテンプレートを削除します。

## 02 事前準備

## 事前準備

---

1. APIキーの入手
2. テンプレートの作成（定型文書を読み取る場合のみ）
3. テンプレートIDの入手（定型文書を読み取る場合のみ）
4. APIドキュメントの確認

# 1. APIキーの入手

1. WebUIの「マイページ」ページで確認します。（左メニューの名前をクリック）
2. APIキーをクリックすると、クリップボードにコピーされます。



The screenshot shows the 'My Page' section of the SmartRead WebUI. It displays the following user information:

項目	値
名前	Masayuki Hoki
メールアドレス	mhoki-demo@ cogent.co.jp
パスワード	パスワード有効期限 2022年3月13日 00:00:00
権限	管理者
所属グループ	Masayuki Hoki
組織	Cogent Demo
ユーザーID	3bae8d5-7750-4e33-8f5c-862e3f393ec7
APIキー	 abff9999-xxxx-xxxx-xxxx-xxxxxxxxxxxx
作成日	2021年12月15日 13:54:32
作成者	Marie Kohata
最終更新日	2021年12月15日 14:26:13



WebUIを利用しないAPIのみ使用可能な  
「APIユーザー」も作成可能です。

## 2. テンプレートの作成 (WebUIで作成)

1. WebUI上で作成します。
2. テンプレートは作成後、SmartRead上で保存されます。  
(JSONファイルとしてエクスポートすることも可能です。)



### 3. テンプレートIDの入手

1. WebUI上でテンプレートを作成します。
2. 該当のテンプレート詳細ページを開きます。
3. Webページ内でテンプレートIDを確認することができません。
4. WebブラウザのURL欄の文字列がテンプレートIDになります。  
こちらのIDをプログラム内で指定してください。

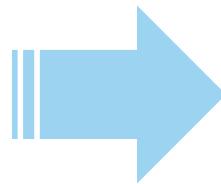
テンプレートID  


<https://app.smartread.jp/template/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX>



The screenshot shows the 'Template List' page of the SmartRead web interface. On the left sidebar, 'Template List' is highlighted. The main area displays a table with several template entries. A red dashed box highlights the first entry, which is labeled '1. クリック' (Click). Another red dashed box highlights the entire row of the first entry, labeled '2. ダブルクリック' (Double Click).

名前	作成日
テンプレート1	2021-12-10 17:00:01
テンプレート2	2021-12-10 17:00:02
テンプレート3	2021-12-10 17:00:03
データカード	2021-12-10 17:00:04
テンプレート	2021-12-10 17:00:05




The screenshot shows the 'Template Detail' page of the SmartRead web interface. The URL in the address bar matches the one shown above. The page displays various details about the selected template, including its file size (1487px wide, 2301px high, 21 KB), field types (Single Line, Multi Line, Block Style, Check Box), and creation date (2021-12-10 17:19:36). A red box highlights the 'Template ID' field, which contains the value 'テンプレート'.

ファイル名	テンプレート
ファイルサイズ	幅 1487 px 高さ 2301 px サイズ 21 KB
フィールド	シングルライン 4 マルチライン 0 ブロックスタイル 4 チェックボックス 9

※タスクIDも同様に、「タスク結果詳細」ページのURL欄でタスクIDを確認します。

## 4. API ドキュメント (<https://docs.smartread.jp/>)

※SmartRead API v4 用APIドキュメント (<https://docs-gsr.smartread.jp/>)



API ドキュメント ヘッダーメニュー

ページ	内容
概要	<ul style="list-style-type: none"><li>SmartRead用語の解説</li><li>基本的なAPIの使い方</li></ul>
APIメソッド	<ul style="list-style-type: none"><li>API機能/仕様/設定方法</li><li>レスポンス内容/形式</li><li>サンプルコード</li></ul>
エラーコード	<ul style="list-style-type: none"><li>HTTPステータスコード</li><li>内部エラーコード</li></ul>
出力フォーマット	<ul style="list-style-type: none"><li>読み取対象文書の種類と出力形式</li><li>JSON出力スキーマ</li></ul>

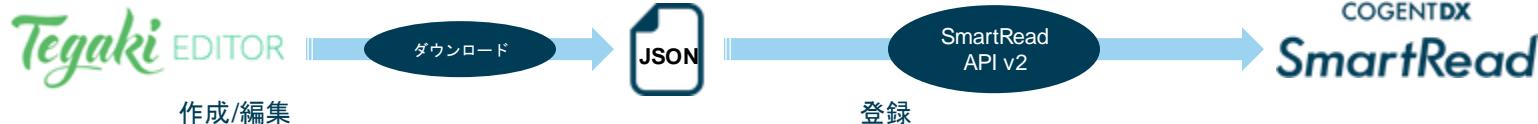
## 03 SmartRead API v2 使用時の留意点

SmartRead WebUI は、SmartRead API v3 用に作られています。  
このため、WebUI上で SmartRead API v2 に関する以下の操作、ダウンロード  
を行うことができません。

- テンプレート
- 読取結果結果

# SmartRead API v2 用テンプレートの作成/登録/編集

1



2

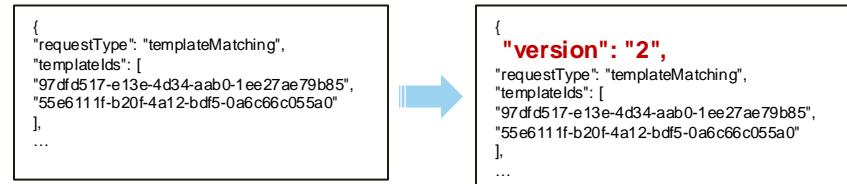


## 1. Tegaki Editor を使用する場合

- そのまま SmartRead API v2 でも利用可能です。
- SmartReadへの登録は **SmartRead API v2 を使用**してください。
  - SmartRead WebUI での登録はできません。

## 1. SmartRead WebUI を使用する場合

- そのままでは SmartRead API v2 で利用できません。
  - JSONファイルを編集（1行追加）してください。
- SmartReadへの登録は **SmartRead API v2 を使用**してください。
  - SmartRead WebUI での登録はできません。



# Tegaki API v2 からの SmartRead API v2 への移行



Tegaki API v2 は、SmartRead API v2 と同じ仕様です。  
このため、既存のTegaki用プログラムはほぼそのままご利用可能ですが、  
移行時には以下の留意点をご確認ください。

変更項目	SmartReadで動作させるために必要となる作業
エンドポイント	<p>https://api.tegaki.ai/hwr/v2/...</p> <p>↓</p> <p>https://api.<b>smartread.jp</b>/hwr/v2/...</p>
APIキー	<p>SmartRead用APIキーの設定が必要</p> <ol style="list-style-type: none"><li>1. SmartRead WebUI 「マイページ」を表示</li><li>2. ページ上のAPIキーをクリップボードにコピー</li><li>3. 既存プログラム内APIキー→コピーしたAPIキー</li></ol>
テンプレートID	<p>SmartRead用テンプレートIDの設定が必要</p> <ol style="list-style-type: none"><li>1. TegakiからテンプレートJSONファイルをエクスポート</li><li>2. SmartRead API v2 を使いJSONファイルを登録 (<u>WebUIでの登録は不可</u>)</li><li>3. 既存プログラム内テンプレート→ID戻り値であるテンプレートID</li></ol>

## 04 定型文書の読み取り処理の実行例（Shell+Curl）

※JSONファイルを読み込む形式での実行例です。  
※斜体文字列は環境によって入力値が変わります。

# エクスポートを使用した読み取り処理フロー

Step

1	タスクを作成 (POST) → タスクIDを取得
2	タスクIDに対し、ファイルをアップロード (POST) → リクエストIDを取得
3	タスク・ステータスの確認 (GET)
4	タスクIDに対し、エクスポートを実行 (POST) → エクスポートIDを取得
5	エクスポート・ステータスの確認 (GET)
6	タスクID, エクスポートIDに対し、結果を取得 (GET) → 結果をZIPファイルで取得

INPUTの定義

リクエスト内容の定義  
(例：「預金通帳」を読みたい)

OUTPUTの定義

出力形式の定義  
(例：「Excel」で出力したい)

※出力形式はタスク作成時でも定義可能です。



ステータス確認処理は必ず実行してください。  
(Step3, 5)

# INPUTの定義

**Task**

タスクを作成する

SmartReadが処理する一連のファイルを接続するコンテナとして、次のタスクを作成します。レスポンスに含まれる `taskid` を指定して、別のエンドポイントから後にファイルを追加できます。`requestType` により、以下のタスクを実行できます。

- templateMatching - 定型文書・仕分帳面から読み取りまでを自動実行
- templateMatchingOnly - 定型文書・仕分帳面のみを実行
- templateMatchingWithVerification - 定型文書・仕分帳面までを実行（SmartReadの画面で仕分帳面、読み取りが実行されます）
- form - 定型文書・読み取りのみを実行（仕分帳面は行いません）
- freeform - 非定型文書・読み取りのみを実行（仕分帳面は行いません）

**REQUEST BODY SCHEMA: application/json**

```

{
  "exportSettings": {
    "any": "EsportsBody"
  },
  "name": {
    "string": "タスク名",
    "maxLength": 30,
    "minLength": 1
  },
  "allowInheritData": {
    "boolean": "データの利用を承認",
    "default": false
  },
  "taskPatternId": {
    "string": "タスク定式ID",
    "description": "APIエンドポイントで指定しているtaskPatternIdと同じものを指定してください"
  },
  "description": {
    "string": "タスクの説明",
    "maxLength": 800,
    "minLength": 1
  },
  "label": {
    "object": {
      "label": "タスクラベル",
      "description": "リクエストにラベルを付けるために、キーと値の組み合わせを文字列にしてマップです。リクエストあたりの最大ラベル数は20で、キーと値の最大長はそれぞれ255文字です"
    }
  },
  "languages": {
    "array": {
      "language": "言語"
    }
  },
  "requestType": {
    "enum": [
      "form",
      "freeform"
    ],
    "label": "タスク定式の種類"
  }
}

```



タスクの種類	読取対象文書・処理内容
form	<b>定型文書</b> 「仕分処理無（テンプレート1種類）」
bankBook	<b>預金通帳</b>
invoice	<b>請求書</b>
templateMatching	<b>定型文書</b> 「仕分処理から読み取りまで」を自動実行
templateMatchingOnly	<b>定型文書</b> 「仕分処理のみ」を実行
templateMatchingWithVerification	<b>定型文書</b> 「仕分処理まで」を実行
freeform	<b>非定型文書</b>

# OUTPUTの定義

## Exports

タスクから出力を作成する。

タスクに含まれるすべての処理結果を出力します。現在はテンプレートベースのタスクはCSVとExcelフォーマットをサポートしており、フリーフォームのタスクはテキスト付PDFフォーマットをサポートしています。

REQUEST BODY SCHEMA: application/json

値が指定されていない場合、デフォルトは次のようにになります:

リクエスト タイプ	デフォルトのエクスポート タイプ	デフォルトの集計
freeform	searchablePdf	oneFilePerRequest
form	excel	oneFilePerTemplate
templateMatching	excel	oneFilePerTemplate
templateMatchingWithVerification	excel	oneFilePerTemplate
bankBook	csv	oneFilePerRequest
invoice	csv	oneFilePerRequest

```

type          string   出力タイプ
              csv
aggregation  string
              Enum: "oneFilePerTemplate", "oneFile",
              "oneFilePerRequest"
              結果の出力形式
  
```

出力タイプ	出力結果
csv	CSV
excel	Excel
searchablePdf	テキスト付PDF (Searchable PDF)
json	JSON

# Step1：タスクを作成（POST）

- 空タスクの作成

## サンプルプログラム

```
#!/bin/bash

curl -X POST ¥
-H "Authorization:apikey XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" ¥
-H "Content-Type:application/json" ¥
--data-binary "@/Work/task.json" ¥
https://api.smartread.jp/v3/task
```

## サンプルJSONファイル (task.json)

```
{
  "requestType": "templateMatching",
  "templateIds": [
    "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
  ],
  "name": "Sample",
  "description": "Test TemplateMatching",
  "labels": {
    "classification": "資料掲載用のサンプル読み取りです。"
  },
  "languages": [
    "ja"
  ]
}
```



## レスポンス結果

```
{"taskId": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"}
```

## Step2：タスクIDに対し、ファイルをアップロード (POST)

- Step1で作成したタスク（タスクID）へ読み取対象文書の送信

サンプルプログラム

```
#!/bin/bash

curl -X POST ¥
-H "Authorization:apikey XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" ¥
-F "image=@/Work/payroll.png" ¥
https://api.smartread.jp/v3/task/:taskId/request
```



Step1で作成されたタスクID



レスポンス結果

```
{" requestId": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" }
```

## Step3：タスク・ステータスの確認（GET）

- 正常完了の確認（「OCR\_COMPLETED」）

### サンプルプログラム

```
#!/bin/bash

curl -X GET \
-H "Authorization:apikey XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" \
-H "Content-Type:application/json" \
--data-binary "@/Work/export.json" \
https://api.smartread.jp/v3/task/:taskId
```

↑  
Step1で作成されたタスクID



### 結果

```
{"userId": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX", "name": "Payroll", "labels": {"classification": "Test"}, "languages": ["ja"], "description": "Sample test", "allowUseOfData": true, "requestType": "payrollReport", "modified": "2022-01-13T07:52:05.845Z", "created": "2022-01-13T07:52:05.844Z", "orgId": "XXXXXXXX-XXXX-XXXX-XXXX-XXXX-XXXXXX", "groupId": "XXXXXXXX-XXXX-XXXX-XXXX-XXXX-XXXXXX", "taskId": "XXXXXXXX-XXXX-XXX X-XXXX-XXXXXX", "state": "OCR_RUNNING", "requestStateSummary": {"OCR_RUNNING": 0, "OCR_COMPLETED": 1, "OCR_FAILED": 0, "OCR_VERIFICATION_COMPLETED": 0, "SORTING_RUNNING": 0, "SORTING_COMPLETED": 0, "SORTING_FAILED": 0}, "numOfRequests": 1, "numOfPages": 1, "formStateSummary": {"OCR_RUNNING": 0, "OCR_COMPLETED": 1, "OCR_FAILED": 0, "OCR_VERIFICATION_COMPLETED": 0, "SORTING_RUNNING": 0, "SORTING_COMPLETED": 0, "SORTING_FAILED": 0, "SORTING_DROPPED": 0}}
```

## Step5：エクスポート・ステータスの確認（GET）

- 正常完了の確認（「COMPLETED」）

### サンプルプログラム

```
#!/bin/bash

curl -X GET ¥
-H "Authorization:apikey XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" ¥
-H "Content-Type:application/json" ¥
--data-binary "@/Work/export.json" ¥
https://api.smartread.jp/v3/task/:taskId/export/:exportId
```

Step1で作成されたタスクID



Step4で作成されたエクスポートID



### レスポンス結果

```
{"type":"excel","aggregation":"oneFile",
"exportId":"XXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
"state":"COMPLETED",
"taskId":"XXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"}
```

## Step6：読み取結果ファイルの取得

- ZIPファイルの取得

サンプルプログラム

```
#!/bin/bash

curl -X GET \
-H "Authorization:apikey XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" \
-o "@/Work/v3_Task/result.zip" \
https://api.smartread.jp/v3/task/:taskId/export/:exportId/download
```

↑  
Step1で作成されたタスクID

↑  
Step4で作成されたエクスポートID



レスポンス結果

```
result.zip
```

## Step4：タスクIDに対し、エクスポートを実行（POST）

- エクスポート処理の実行（CSV出力の場合）

サンプルプログラム

```
#!/bin/bash

curl -X POST \
-H "Authorization:apikey XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" \
-H "Content-Type:application/json" \
--data-binary "@/Work/export.json" \
https://api.smartread.jp/v3/task/:task_id/export
```

↑  
Step1で作成されたタスクID

サンプルJSONファイル（export.json）

```
{
  "type": "CSV",
  "aggregation": "oneFile"
}
```



結果

```
{"exportId":"XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"}
```

## 05 読取処理の自動化

# SmartReadと他社製品との連携

APIを使うことで、他システム連携/自動化を実現することができます。

プログラム開発の敷居を下げるため、内部的なAPIの動きを意識せずとも、  
容易にシステム連携できる連携ソリューションをご用意しています。

- ・ 他社製品標準機能への組み込み
- ・ 連携コネクター
- ・ サンプルプログラム



RPA



BizteX Connect

iPaaS



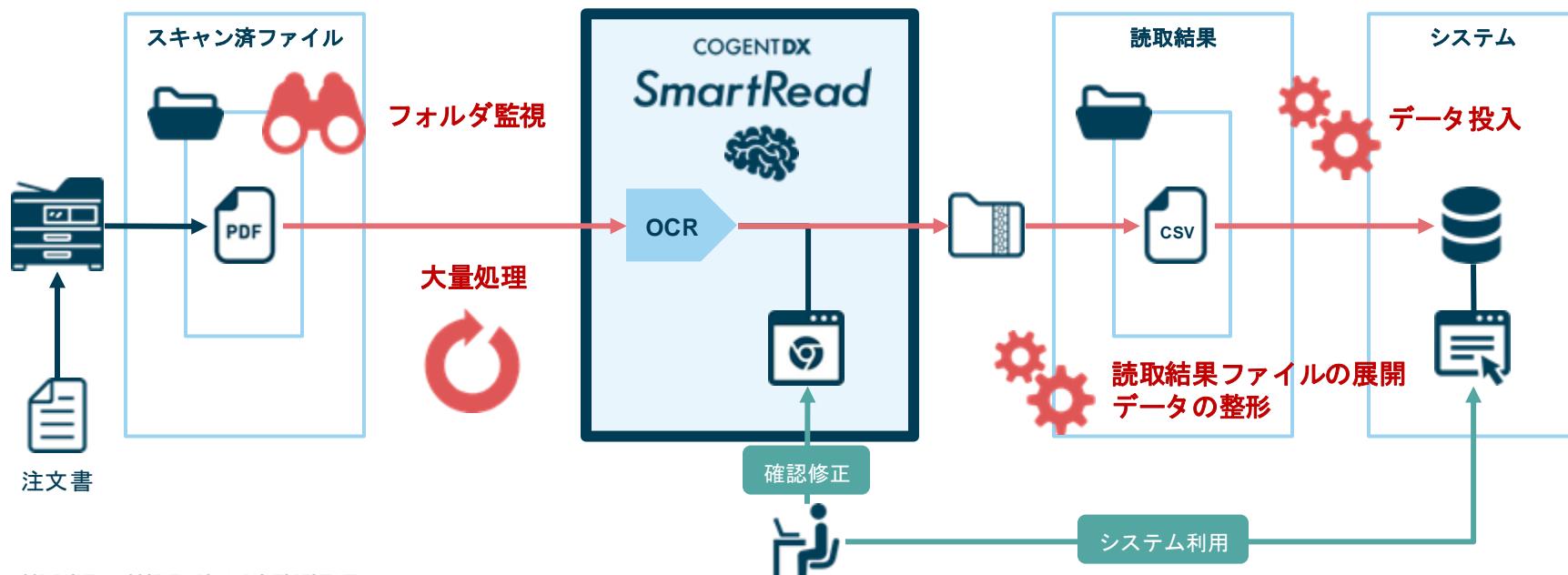
# 他ソリューションとのAPI連携による読み取り処理の自動化イメージ



ロボットがつなげる  
(処理フローの自動化)

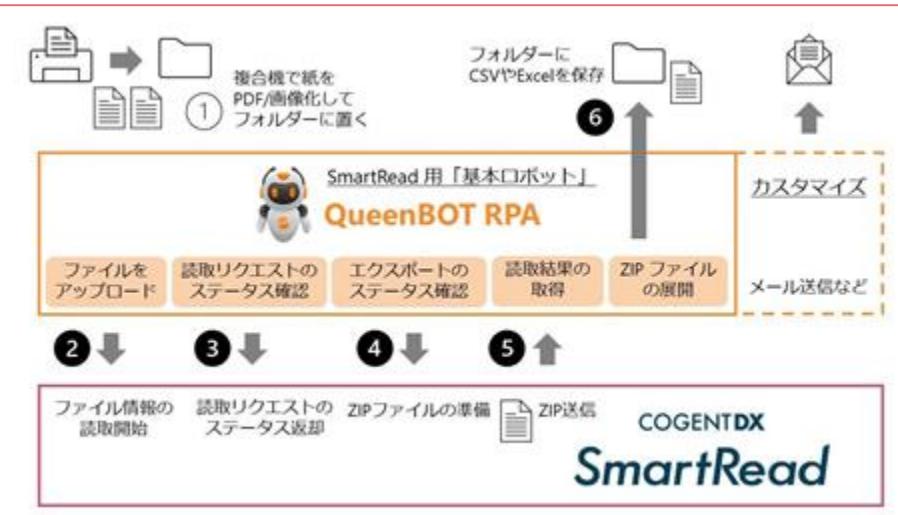


赤い線：ロボットの処理  
緑の線：ユーザーの作業



## RPAとの連携

読み取り処理の基本ステップをパラメータ設定のみですぐに業務フローに組み込めるRPA用ロボットパート「RPAコネクター」を無償提供しています。



QueenBOT RPA の場合



- BizRobo!
- Blue Prism
- QueenBOT RPA
- UiPath
- WinActor



「Power Automate」用ロボットフローについては、サンプルプログラム、もしくはパートナー様ソリューションとしてご提供いたします。

# RPAコネクターの中身

```

SmartReadSample
├サンプルロボット
| └サンプルロボット.ums7
├ライブラリ
| └エクスポート.ums7
| └エクスポート確認.ums7
| └ダウンロード.ums7
| └タスク作成.ums7
| └リクエスト.ums7
| └リクエスト確認.ums7
| └削除.ums7
└input
  ├sample
  | └export_csv.json
  | └export_excel.json
  | └export_pdf.json
  | └task_freeform.json
  | └task_verification.json
  | └task_withoutverification.json
  | └export.json
  | └task.json
└download
└upload
└response
    ... サンプルロボット格納フォルダ
    ... サンプルロボット本体
    ... コネクター格納フォルダ
    ... エクスポート API 用
    ... エクスポート確認 API 用
    ... ダウンロード API 用
    ... タスク作成 API 用
    ... リクエスト API 用
    ... リクエスト確認 API 用
    ... 削除 API 用
    ... API 用 JSON 格納フォルダ
    ... サンプル JSON 格納フォルダ
    ... CSV エクスポート用 Sample (定型文書用)
    ... Excel エクスポート用 Sample (定型文書用)
    ... PDF エクスポート用 Sample (主に非定型用)
    ... 非定型文書のタスク用 Sample
    ... 定型文書の仕分け・読み取り確認有りタスク用 Sample
    ... 定型文書の仕分け・読み取り確認無しタスク用 Sample
    ... エクスポート用 JSON
    ... タスク作成用 JSON
    ... ダウンロードファイル格納フォルダ
    ... アップロードファイル格納フォルダ
    ... レスポンス JSON ファイル格納フォルダ
  
```

- マニュアル
- 各種ロボットと関連フォルダ
- 基本動作サンプルプログラム

※WinActorコネクターの場合



SmartRead API v3 を使用しています。

# Power Automate for Desktop サンプルプログラム

---

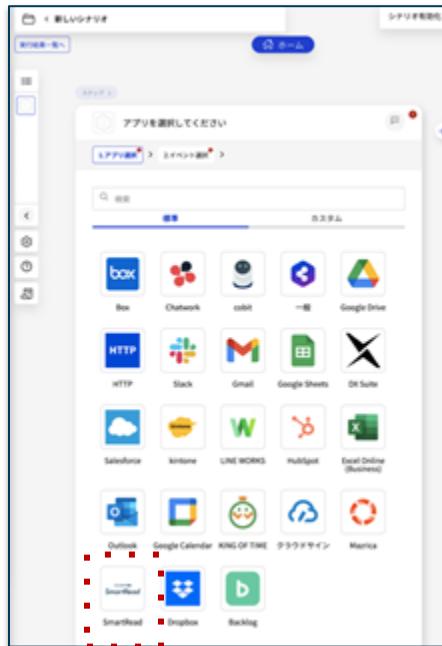


- 定型文書を読み取り、読み取り結果をExcel形式で出力
- 定型文書を読み取り、読み取り結果をkintoneにインポート
- PDFを読み取り、テキスト付PDF形式で出力

# Biztex Connect (iPaaS) との連携

標準でSmartReadが利用可能です。

メール、クラウドサービスとの連携をWeb画面上で容易に作成できます。



 Biztex Connect

# 参考 用語の解説

# 押さえておくべき主な用語

---

- APIキー
- テンプレート
- タスク
- 仕分け

# APIキー



SmartRead API を実行する際に必要となる認証キーになります。  
APIキーは、SmartRead WebUI 上で確認します。

```
headers = {"Authorization": "apikey XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"}  
response = requests.get(url, headers=headers)
```

## テンプレート（定型文書読取時のみ必要）

# テンプレート



雛形 フォーム フォーマット のようなもので

記入箇所が定義された設定ファイルのこと

※テンプレートにより、どの位置（箇所）にどういう内容が記載されているか定義が可能

# タスク

# タスク



- ・どのテンプレートを使う？
- ・結果はExcel出力する？
- ...



## 読み取り処理の受付単位（＝業務）

※複数テンプレートを設定した場合、SmartReadが自動的に仕分処理



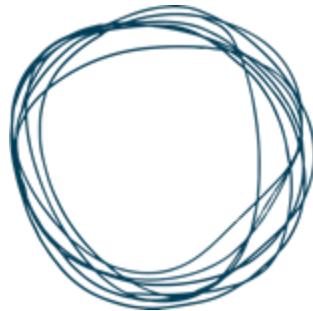
SmartRead API v3 を使用する場合には、必ず「タスク」の作成が必要になります。

## 仕分け（仕分処理）



読み取った定型文書が  
どのテンプレートで作られた文書か？  
を仕分けするSmartRead上の処理機能のこと

✖ 読取対象文書ファイルを種類毎に  
Windows OS 上でフォルダ分類する



COGENT  
LABS

#### 免責事項

- ・ 本文書はご参考資料としてご提供するものです。本文書に関して弊社は質問へのご回答などサポートのご提供はいたしません。
- ・ 本文書は作成時において弊社が知り得た情報をもとに作成したものです。弊社は、本文書の正確性、網羅性、最新性などのいかなる事項についても保証いたしません。
- ・ 弊社は、本文書に関して、請求の原因の如何を問わず損害賠償責任その他の責任を負いません。