AgileWorks R3

Web API 利用ガイド

R3.3 第 1 版(2026/07/15)

ATLED

目次／索引

1. はじめに 4
1.1. 本書の目的 4
1.2. 対象読者 4
1.3. 前提条件 4
2. Web API の概要 5
2.1. API の種類 5
2.2. 基本的な使い方／組み合わせ 5
2.2.1. データを取得する 5
2.2.2. 新しくデータを作成する 6
2.2.3. 既存のデータを更新／削除する 6
2.2.4. 添付ファイルをダウンロードする 6
2.2.5. SCIM プロトコルと利用して外部システム／外部サービスと連携する 6
2.3. 実行権限 6
3. 認証方式 7
3.1. Basic 認証 7
3.2. OAuth2 認証 7
3.2.1. OAuth2 認証でアクセストークンを発行する 7
3.2.2. 認可コードを取得する 9
3.2.3. 認可コードを使用してアクセストークンを発行する 10
3.2.4. リフレッシュトークンを使用してアクセストークンを再発行する 12
3.2.5. 管理サイトからアクセストークンを発行する 13
4. 共通仕様 14
4.1. リクエストヘッダ 14
4.2. エラー時の応答 15
4.3. その他 16
4.4. パフォーマンスに関する考慮事項 17
4.4.1. API 呼び出しの最適化 17
4.4.2. 大規模データ取得時の注意点 17
4.4.3. 運用における注意点 17
5. API の一覧とリファレンスガイド 18
5.1. リファレンスガイドの使い方 18
5.2. サービス実行 API の分類 21
5.3. 管理者権限不要のサービス実行 API 21
5.4. SCIMAPI 22
5.4.1. SCIM API とは 22
5.4.2. 主な機能 22
6. サンプルプログラム 23
6.1. 書類 API　サンプルプログラム 23
6.1.1. 書類データを取得する（Java） 23
6.1.2. 書類データを更新する（JavaScript） 25
6.1.3. 書類 PDF データをダウンロードする（Python） 26
6.1.4. 書類コメント/メモを取得する（Java） 27
6.1.5. 添付ファイルをダウンロードする（Python） 29
6.2. ワークフローAPI　サンプルプログラム 31
6.2.1. 書類を申請する（Java） 31
6.2.2. 書類を承認する（JavaScript） 33
6.3. 組織 API　サンプルプログラム 34
6.3.1. ユーザー情報を取得する（Java） 34
6.3.2. ユーザー情報を更新する（JavaScript） 36
6.3.3. 組織を作成する（Python） 37
6.3.4. 組織所属を作成する（Java） 38

◆ 改版履歴

版数	年月日	改版内容
第 1 版	2026 年 07 月 15 日	第 1 版作成

1. はじめに

1.1. 本書の目的

本書は、AgileWorks Web API を利用したアドオン開発の概念・アーキテクチャについて説明します。

本書では一部サンプルプログラムを提供していますが､用意するサンプルは API 利用方法の説明に主眼を置いていますので、業務ロジックにおいて共通関数化・クラス化すべき部分の最適化は行っておらず、必ずしもベストコーディングといえない方法を取っている箇所もあります。
あくまでサンプルという位置付けでとらえるように確認ください。

1.2. 対象読者

AgileWorks のアドオン開発を行う開発マネージャ、開発担当者

1.3. 前提条件

・プログラミング言語・手法に関する基本知識を理解していること
・AgileWorks 基本機能と仕様を理解していること
　特に「組織」「ユーザー」「セクションロール」「ユニバーサルロール」「フォーム」「回付ルール」「書類」「ワークフロー」といった AgileWorks 上の主要オブジェクトに関する概念と関連について理解していることを前提としています。

2. Web APIの概要

Web API とは、Web システムを外部から呼び出すための API です。 HTTP の技術をベースとした API のため、開発言語に依存することなく利用することができます。 リソースの操作は HTTP メソッドから指定し、結果は JSON 形式のフォーマットのデータとして受け取ることが可能です。

AgileWorks の Web API では、外部のシステムとの連携に使用可能な次のような機能を提供しています。

例：
- ユーザーや組織の一覧取得、作成、更新、削除
- 回付書類の情報一覧の取得
- ワークフローの各種操作（申請、承認、差戻しなど）
- 回付ルート上の処理ユーザー一覧の取得
- 回付履歴の取得
- 書類の添付ファイルや PDF データの取得

2.1. API の種類

AgileWorks の Web API は以下の 4 種類に分かれています。

- サービス実行 API
  - AgileWorks 上のデータ取得や追加、更新などの処理を実行します。
  - 何らかの処理を行う際に基本的にこのサービス実行 API を利用します。
- モデルオブジェクト取得 API
  - サービス実行 API へのリクエストボディに利用するモデルオブジェクトを取得します。
  - 新しくデータを追加、作成する際に利用します。
- ダウンロード API
  - 書類の添付ファイルをダウンロードします。
- SCIM API
  - SCIM プロトコルに準拠した組織、ユーザーを操作する API です。
  - SCIM プロトコルに対応したシステム／サービスとの連携を行う場合に利用できます。

上記の API を組み合わせて要件にあった連携処理を構築してください。

2.2. 基本的な使い方／組み合わせ

WebAPI を利用して各種操作を行う際の参考としてご利用ください。

2.2.1. データを取得する

1. 取得したい API に必要なモデルオブジェクトを取得（**モデルオブジェクト取得 API**）
2. 1 で取得したモデルオブジェクトを利用してリクエストボディを生成
3. 生成したリクエストボディでデータを取得（**サービス実行 API**）

一部のデータ取得系の API ではリクエストボディが不要、または書類 ID のみでデータを取得することができます。
※マスタ参照コンポーネント一覧取得 API（name=form.Form&method=listComponentMasterWindow）、
書類情報取得 API（name=doc.Doc&method=getDocHeader） など

2.2.2. 新しくデータを作成する

書類と書類以外で組み合わせが異なります。

■書類の場合

1. 新規書類データ作成 API に必要なモデルオブジェクトを取得
   （**モデルオブジェクト取得 API**）
2. 1 のレスポンスボディから「doc.PrepareDocRequestModel」を生成
3. 生成したリクエストボディで書類を新規作成
   （**サービス実行 API > 新規書類データ作成 API**）
4. 3 のレスポンスボディから「doc.DocModel」を取得
5. 4 で取得した「doc.DocModel」に入力したいフィールド値をセット（任意）
6. 用意した「doc.DocModel」を利用して書類を保存
   （**サービス実行 API > 新規書類データ保存 API**）
7. 保存した書類データに対して「書類下書き保存 API」または「書類作成/申請 API」を実行

■書類以外の場合

1. 作成したいデータに必要なモデルオブジェクトを取得
   （**モデルオブジェクト取得 API**）
2. 1 で取得したモデルオブジェクトを利用してリクエストボディを生成
3. 生成したリクエストボディでデータを作成
   （**サービス実行 API**）

2.2.3. 既存のデータを更新／削除する

操作したい対象データを取得し、更新の場合は

1. 対象データを特定
   「データを取得する」を参照
2. リクエストボディを作成
3. データの更新／削除を実行

2.2.4. 添付ファイルをダウンロードする

1. 対象のファイル ID を取得
   「書類添付情報一覧取得 API」の結果から取得します。
2. **ダウンロード API** を実行

2.2.5. SCIM プロトコルと利用して外部システム／外部サービスと連携する

SCIM プロトコル、及び連携する外部システム、外部サービスの仕様に合わせて SCIM API をご利用ください。

2.3. 実行権限

「全てのコンテンツ」と「全ての業務カテゴリ」の権限を持つ管理ロールを持っているユーザーで認証を行うと全ての Web API を実行することができます。

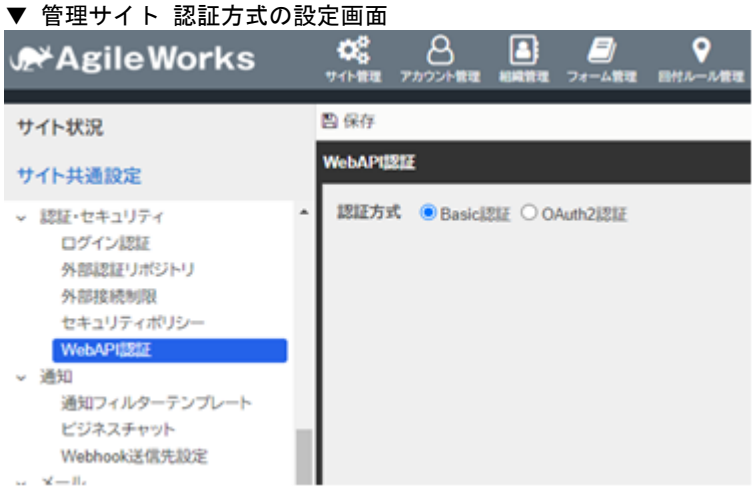
管理ロールを持っていない、または一部の管理権限しか持っていないユーザーは以下の Web API を実行することができます。

- モデルオブジェクト取得 API（R3.1.1 以降のみ）
- 各 API の管理者権限不要 API（NotAdminAPI）

3. 認証方式

管理サイト【サイト管理】→【サイト共通設定】→【サイト共通設定】→【認証・セキュリティ】→【WebAPI認証】から Web API の各種認証方式の設定が可能です。
デフォルトは OAuth2 認証となっており、選択していない認証方式は使用することができません。

▼ 管理サイト 認証方式の設定画面



3.1. Basic 認証

Basic 認証を利用する場合は、各ユーザーのログイン ID とパスワードを用いて認証を行います。
デフォルトは OAuth2 認証となっているため、認証方式「Basic 認証」を選択して、設定を保存してください。
※ライセンスに WebAPI オプションがない場合は Basic 認証を使用することはできません。

3.2. OAuth2 認証

OAuth2 認証を利用する場合は、アクセストークンを用いて Bearer 認証を行います。
使用するアクセストークンは下記のいずれかで発行できます。

- OAuth2 認証で発行する
- 管理サイトから発行する

※ライセンスに WebAPI オプションがない場合は管理サイトからのみ発行が可能となっています。
このアクセストークンは共通機能でのみ利用可能です。

3.2.1. OAuth2 認証でアクセストークンを発行する

OAuth2 認証で発行するために必要な手順は以下となります。

1. 管理サイトからクライアント ID を生成
2. OAuth2 認証リクエストの送信
3. ユーザー認証
4. 認可コードの取得
5. 認可コードでアクセストークンを発行

▼ Oauth2 認証　ベース URL

```
<プロトコル>://<サーバーアドレス>/<コンテキスト名>/oauth2
```

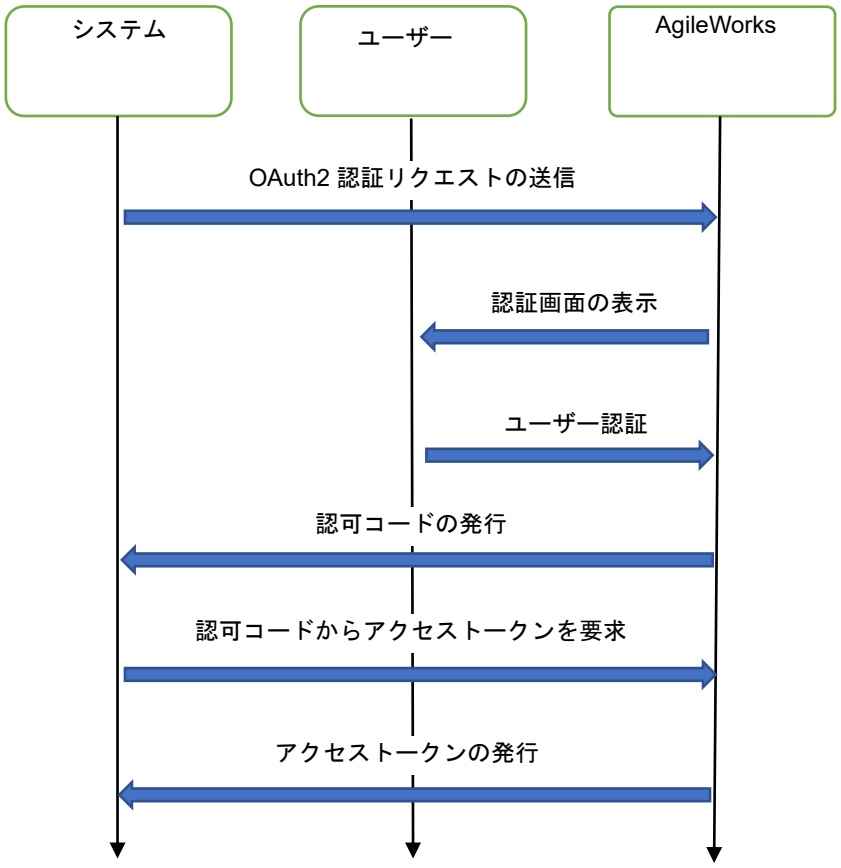
※プロトコル　: http, https
※サーバーアドレス　: AgileWorks が稼働しているサーバーのアドレス
※コンテキスト名　: AgileWorks Web アプリケーションのコンテキスト名（デフォルトは AgileWorks)

▼ 管理サイトでクライアント ID を作成する



クライアント ID は OAuth2 認証を使用して認可コードやアクセストークンを発行する際に必要となります。
OAuth2 認証を使用する前に必ず作成してください。
「リダイレクト URL のスキーマ」は、使用するリダイレクト URL に合わせるように設定してください。

▼ 認可コード取得 ～ アクセストークン発行までのフロー図



3.2.2. 認可コードを取得する

▼ HTTP メソッド

GET

▼ クエリパラメータ

パラメータ名	必須	説明
client_id	○	クライアント ID。 管理サイトで作成したクライアント ID を指定してください。
redirect_uri	○	リダイレクト URL。 任意の URL を URL エンコードして指定してください。
response_type	○	認可フロー。「code」を指定してください。
scope		スコープ群。
state		CSRF 対策用パラメータ。
code_challenge_method	○	code_challenge の暗号化方式（PKCE 対応）の指定。 S256 ：SHA-256 で暗号化 plain ：暗号化なし
code_challenge	○	暗号文字列（PKCE 対応）の指定。 ■code_challenge_method に「S256」を指定した場合 code_verifier を SHA-256 で暗号化し、 Base64URL 形式にエンコードした値を指定します。 ■code_challenge_method に「plain」を指定した場合 code_verifier の値を指定します。

⚠ 注意事項
「code_verifier」は半角英数字及び記号（ - . _ ~ ）からなる文字列です。
クライアント側で code_verifier を生成する必要があります。

「code_challenge」は SHA-256 で暗号化後、クエリパラメータとして利用できるように
Base64 形式の文字列から以下の文字の削除及び置換を行う必要があります。
・ パディング（文字詰めの「 = 」）の削除
・ 「 + 」を「 - 」に置換
・ 「 / 」を「 _ 」に置換

リクエスト URL 例

https://sample.atled.jp/AgileWorks/oauth2?client_id=aw_webapi_client_xxx-xxx-xxx
&redirect_uri=https%3A%2F%2Fredirect.atled.oauth2
&response_type=code&code_challenge_method=plain&code_challenge=oatuth2_ATLED

▼ Oauth2　認証画面



ユーザー認証に成功すると、「redirect_uri?code={認可コード}」へリダイレクトします。
この時のパラメータ：code が認可コードとなります。
（指定した場合は state もパラメータに含まれます。）

取得した認可コードの有効期限は発行から 10 分です。

3.2.3. 認可コードを使用してアクセストークンを発行する

▼ HTTP メソッド

POST

▼ クエリパラメータ

パラメータ名	必須	説明
client_id	○	クライアント ID。 認可コードリクエスト時と同様のクライアント ID を指定してください。
redirect_uri	○	リダイレクト URL。 認可コードリクエスト時と同様の URL を指定してください。
code	○	認可コード。 取得した認可コードを指定してください。
grant_type	○	OAuth 2.0 許可種別。 アクセストークン発行の際は「authorization_code」を指定してください。
code_verifier	○	PKCE の検証鍵。 認可コードリクエスト時に指定した code_verifier を指定してください。

リクエスト URL 例

https://sample.atled.jp/AgileWorks/oauth2?client_id=aw_webapi_client_xxx-xxx-xxx
&redirect_uri=https%3A%2F%2Fredirect.atled.oauth2&code=xxxxxxxxx&grant_type=authorization_code &code_verifier=oatuth2_ATLED

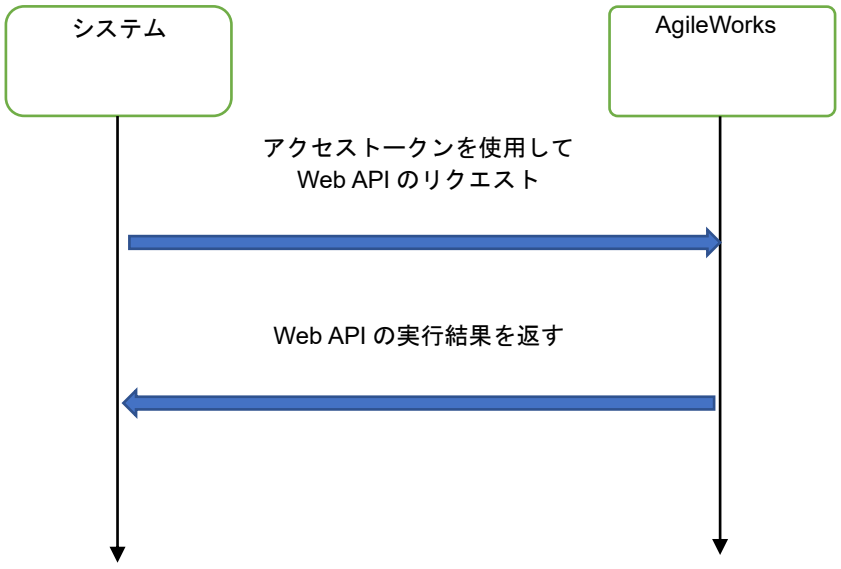
▼ レスポンスボディ例

```
{
    "access_token": "xxxxxxxxxx",
    "refresh_token": "xxxxxxxxxx",
    "token_type": "bearer",
    "expires_in": 3600
}
```

▼ レスポンスボディ　詳細

#	パラメータ	型	説明	備考
1	access_token	文字列	アクセストークン	
2	refresh_token	文字列	リフレッシュトークン	
3	token_type	文字列	アクセストークンの有効期限（秒）	3600 秒（1 時間）固定
4	expires_in	整数	トークン種別	bearer 固定

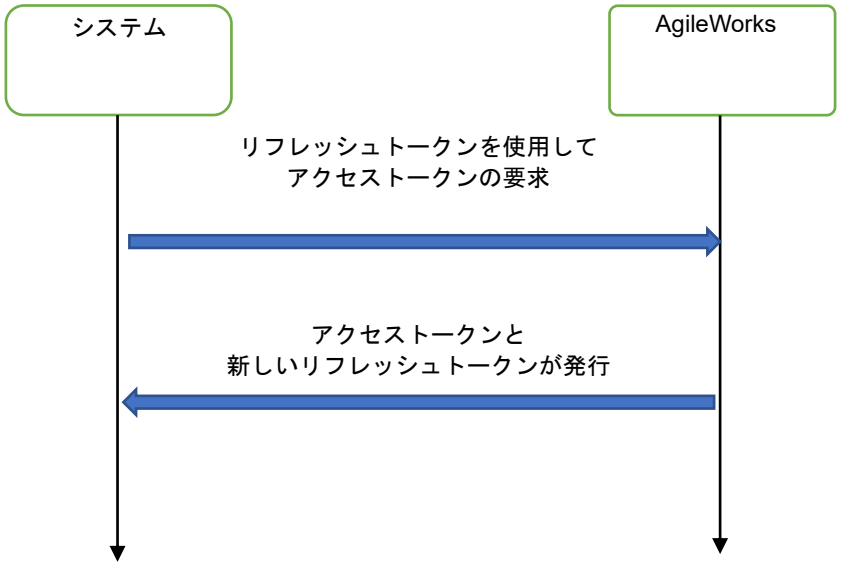
▼ 取得したアクセストークンを使用して Web API を実行する　フロー図



アクセストークンの有効期限は発行から 1 時間です。
アクセストークンの有効期限が切れた場合はリフレッシュトークンを使用してアクセストークンを再発行する必要があります。
一度使用した認可コードは再使用できないため、注意してください。

WebAPI 実行に関する詳細は「3.　共通仕様」、「4.　各 API について」を参照してください。

▼ リフレッシュトークンを使用してアクセストークンを再発行する フロー図



リフレッシュトークンは認可コードからアクセストークンを発行した際に一緒に発行されます。
リフレッシュトークンの有効期限は発行から 180 日です。
リフレッシュトークンの有効期限が切れた場合は、再度認可コードの取得から行ってください。

3.2.4. リフレッシュトークンを使用してアクセストークンを再発行する

▼ HTTP メソッド

POST

▼ クエリパラメータ

パラメータ名	必須	説明
client_id	○	クライアント ID。 管理サイトで作成したクライアント ID を指定してください。
redirect_uri	○	リダイレクト URL。 任意の URL を URL エンコードして指定してください。
refresh_token	○	リフレッシュトークン。 前回発行されたリフレッシュトークンを指定してください。
grant_type	○	OAuth 2.0 許可種別。 「refresh_token」を指定してください。

リクエスト URL 例

https://sample.atled.jp/AgileWorks/oauth2?client_id=aw_webapi_client_xxx-xxx-xxx&redirect_uri=https%3A%2F%2Fredirect.atled.oauth2&refresh_token=xxxxxxxxx&grant_type=refresh_token

▼ レスポンスボディ例

```
{
    "access_token": "xxxxxxxxxx",
    "refresh_token": "xxxxxxxxxx",
    "token_type": "bearer",
    "expires_in": 3600
}
```

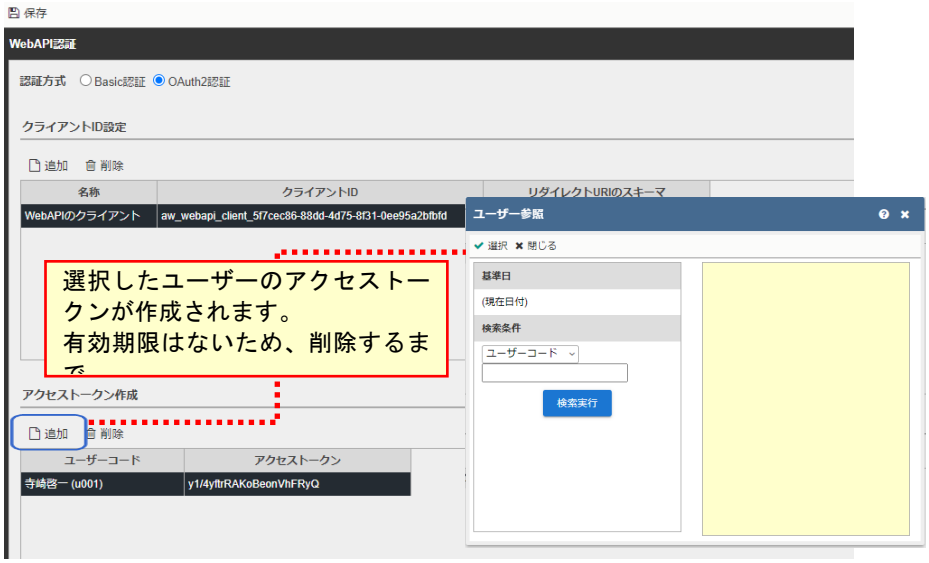
▼ レスポンスボディ　詳細

#	パラメータ	型	説明	備考
1	access_token	文字列	アクセストークン	
2	refresh_token	文字列	リフレッシュトークン	
3	token_type	文字列	アクセストークンの有効期限（秒）	3600 秒（1 時間）固定
4	expires_in	整数	トークン種別	bearer 固定

3.2.5. 管理サイトからアクセストークンを発行する

管理サイトで発行したアクセストークンを使用する場合は上記の流れは必要ありません。
ユーザー毎に作成されたアクセストークンを使用してください。
管理サイトで発行したアクセストークンに有効期限はないため、更新の必要はありません。
また、同じユーザーであっても作成し直した場合、アクセストークンは以前とは別で新規作成されます。

▼ 管理サイトで各ユーザーへのアクセストークンを作成する



4. 共通仕様

4.1. リクエストヘッダ

AgileWorks Web API を使用する際、http ヘッダに各情報を指定する必要があります。
下記の情報はリクエストヘッダの必須項目となります。

▼ 必須のリクエストヘッダ

ヘッダ名	内容
Authorization	使用する認証方法に応じて指定してください。 **■Basic 認証の場合** └「Authorization: <ログイン ID>:<パスワード>」 例：Authorization: admin:agileworks もしくは 「Authorization: Basic <ログイン ID>:<パスワード>」 ※「<ログイン ID>:<パスワード>」は BASE64 エンコードしたものを指定してください 例：Authorization: Basic YWRtaW46YWdpbGV3b3Jrcw== **■OAuth2 認証の場合** └「Authorization: Bearer <事前に取得したアクセストークン>」 例：Authorization: Bearer accesstoken
Content-Type	リクエストボディの形式に応じて指定してください。 **■JSON 形式** Content-Type: application/json; charset=UTF-8 もしくは Content-Type: text/json; charset=UTF-8

4.2. エラー時の応答

Web API は処理結果を示すため、次に示す標準的な HTTP コードを返します。

▼ HTTP ステータスコード例

ステータスコード	内容
200	OK リクエストが適切に処理されたことを示します。
400	Bad Request リクエストの構文が誤っている際のエラーです。
401	Unauthorized ユーザー認証に失敗した際のエラーです。 例：ログイン ID やパスワードが間違っている
403	Forbidden 操作を実行するために必要な権限がない際のエラーです。
404	Not Found 存在しないリソースに対してリクエストした際のエラーです。
500	Internal Server Error サーバー内部でエラーが発生した際のエラーです。
503	Service unavailable サーバーがリクエストを処理する準備ができていない際のエラーです。 例：サーバーがまだ起動中

API の実行に失敗した場合、下記の情報が含まれたレスポンスが返ります。
レスポンスの詳細は 4.2.　サービス実行 API の共通のレスポンスボディ例を参照してください。
この場合のステータスコードは 200 です。

▼ エラーレスポンス

```
{
   "resultStatus" : {
      "messageList" : {
         "entries" : [ {
            "code" : "結果コード",
            "text" : "結果説明"
         } ]
      },
      "status" : "結果ステータス",
      "code" : "結果コード",
      "text" : "結果説明"
   }
}
```

▼ エラーレスポンス例（ユーザー作成 API で重複したユーザーコードを指定して実行した場合）

```
{
   “resultStatus”: {
      “status”: “FAIL”,
      “messageList”: {
         “entries”: [{
            “code”: “SVCORG_W0001”,
            “text”: “指定されたユーザーコードは既に利用されています。”
         }]
      },
      “code”: “AWPCMN0000”,
      “text”: “サービス実行時にエラーが発生しました。”
   }
}
```

4.3. その他

▼ 日付書式
リクエストボディ、レスポンスボディにおける日付フォーマット (申請基準日／申請日時／最終適用終了日時など)は
下記のようになります。

JSON 形式
yyyy-MM-dd HH:mm:ss z
例；2022-04-22 00:00:00 JST

▼ リスト
複数のオブジェクトをリストとして保持する場合は以下のようになります。

JSON 形式
entries フィールドに格納された配列

```
"entryList" : {
   "entries" : [ {
      "field1" : "value1",
      "field2" : "value2"
    }, {
      "field1" : "value1",
      "field2" : "value2"
    } ]
}
```

▼ 外部接続制限
Web API は「外部接続制限」機能の対象となっているため、利用の際はこちらの設定も参照してください。
└ Aw01-主要機能ダイジェスト - 7.7. 外部接続制限
外部接続制限機能を設定することで Web API を使用することが可能な IP アドレスを制限することが可能です。

4.4. パフォーマンスに関する考慮事項

AgileWorks の Web API では実際にユーザーサイトや管理サイトで操作をした場合と同等の負荷が発生します。
また、リクエスト頻度の制限などは行わないため、短時間に大量のリクエストを送信すると、AgileWorks が稼働しているサーバーに過度な負荷がかかり、システム全体のパフォーマンスが低下する可能性があります。
安定した運用を維持するため、以下の点にご注意ください。

4.4.1. API 呼び出しの最適化

- 不要なリクエストの削減
  - データを取得する際は必要な情報のみをフィルタリングして取得することを推奨します。
  - 書類検索時に必要なフィールドのみに絞る　など
- キャッシュの活用
  - 頻繁に変わらないマスタデータなどは、毎回 API で取得するのではなく、
    クライアント側でキャッシュすることを検討してください。

4.4.2. 大規模データ取得時の注意点

- ページネーションの利用
  - 大量データの取得が想定される場合は、任意の条件で絞り込み分割して取得することを推奨します。
    これにより、クライアントとサーバー間の通信量を抑え、安定した応答を期待できます。

4.4.3. 運用における注意点

- 負荷テストの実施
  - 新しい連携システムを構築する際は、本番環境に導入する前に、想定される負荷に応じたパフォーマンステストを実施してください。
    これにより、実際の運用で問題が発生しないかを確認できます。
- サーバーリソースの監視
  - Web API を継続的に利用する際は、サーバーの CPU、メモリ、ディスク I/O などのリソース使用状況を定期的に監視し、負荷の急増が発生していないか検知できるような仕組みや検知後の対策シナリオなどを用意してください。

5. APIの一覧とリファレンスガイド

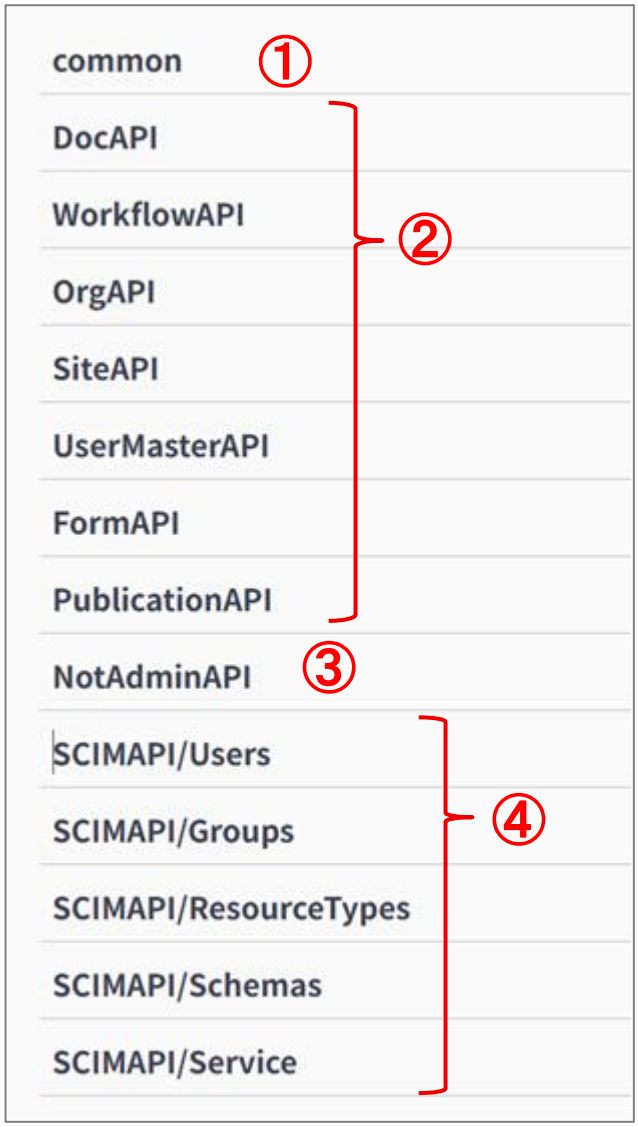
AgileWorks で利用可能な Web API の一覧はバージョン毎に以下 WEB ページよりご確認ください。

- R3.3.0 用
  - ➢ https://atled-workflow.github.io/AgileWorks-doc/api/R330/

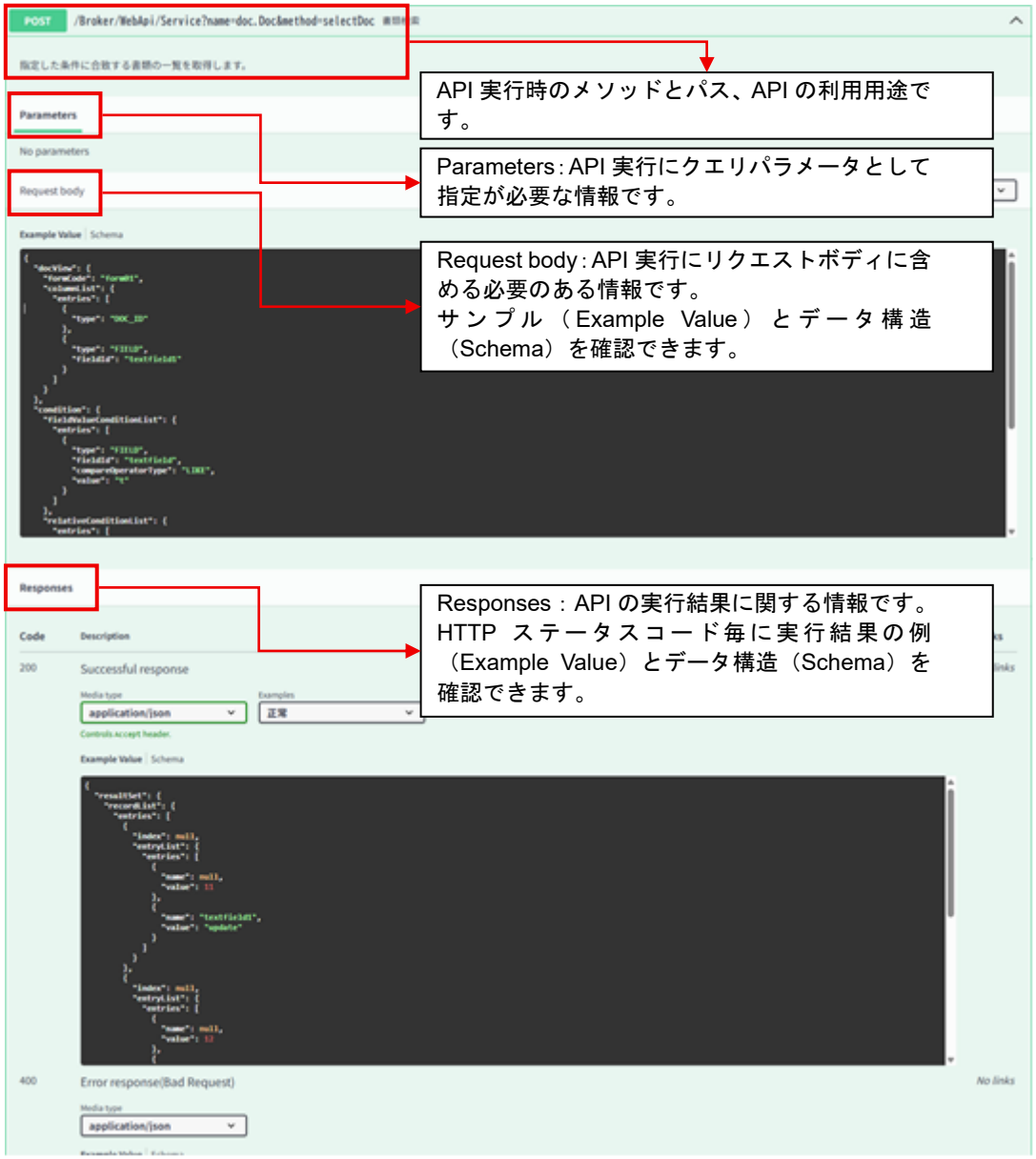
5.1. リファレンスガイドの使い方

AgileWorks の各 Web API の詳細について公開しています。

- ①「API の種類」にて説明したモデルオブジェクト取得、サービス実行、ダウンロードの詳細（common）
- ②サービス実行 API の詳細
- ③管理者権限不要のサービス実行 API の詳細（NotAdminAPI）
- ④SCIMAPI の詳細

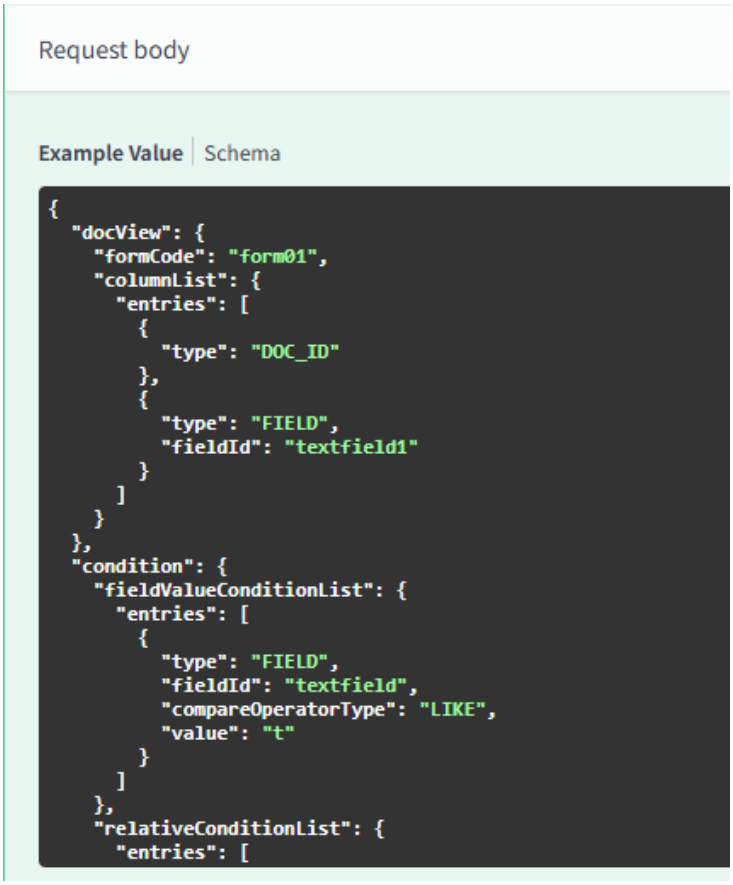


上記それぞれについて以下の情報を確認することができます。



Request body と Responses ではサンプル値とデータ構造を確認することができます。
実行時に必要なデータを確認する際の検証にはサンプル値（Sample Value）と、実際に連携する仕組みを構築する際にはデータ構造（Schema）を参照してください。

▼サンプル値



▼データ構造



5.2. サービス実行 API の分類

サービス実行 API では以下の種別に API が分類されています。
各分類の詳細はリファレンスガイドをご確認ください。

分類	用途
DocAPI	書類の作成や保存、情報取得
WorkflowAPI	書類の申請や承認などフローに関わる操作 回付状況などフローに関する情報の取得
OrgAPI	組織関連データの作成や保存、削除 組織関連データに関する情報の取得
SiteAPI	業務カテゴリに関する情報取得
UserMAsterAPI	外部マスタ(拡張、標準)の取込、及び情報の取得
FormAPI	フォーム、コンポーネントに関する情報の取得
PublicationAPI	公開フォルダ、公開フォームに関する情報の取得

5.3. 管理者権限不要のサービス実行 API

R3.1.1 では以下の API を実行することができます。

API	用途
新規書類データ作成	初期状態の書類データを作成する API です。 取得したレスポンスは「新規書類データ保存 API」で使用できます。
新規書類データ保存	新規に作成した書類データを保存します。 「新規書類データ作成 API」で取得した doc の値を雛形として、書類の各フィールドに値をセットし、本 API により書類データの保存を行います。 本 API で保存した書類データを画面から参照できるようにするには、さらに「書類作成 / 申請 API」を実行する必要があります。
書類検索	API 実行ユーザーが閲覧可能且つ 指定した条件に合致する書類の一覧を取得する API です。
書類表示	API 実行ユーザーが閲覧可能な書類を表示する HTML を取得する API です。
回付情報件数一覧取得	API 実行ユーザーが閲覧可能な回付情報(※)の件数の一覧を取得する API です。 ※ユーザーサイト > 処理待ち > 承認依頼　など
回付情報検索	API 実行ユーザーが閲覧可能な回付情報の一覧を検索する API です。 ※ユーザーサイト > 処理待ち > 承認依頼　など
書類作成/申請	API 実行ユーザーが、本人または代理の立場で書類作成/申請を行う API です。
所有している代理申請権限取得	API 実行ユーザーの所有している代理申請権限を取得する API です。
所有している代理承認権限取得	API 実行ユーザーの所有している代理承認権限を取得する API です。
公開フォルダ･公開フォーム取得	API 実行ユーザーの公開フォルダ・公開フォーム一覧を取得する API です。
ユーザー情報取得	API 実行ユーザー自身のユーザー情報を取得する API です。
バージョン情報取得	AgileWorks のバージョン情報を取得する API です。

5.4. SCIMAPI

5.4.1. SCIM API とは

SCIM（System for Cross-domain Identity Management）API は、アイデンティティ管理を簡素化および標準化するためのプロトコルと API 仕様です。主な目的は、異なるアプリケーションやシステム間でユーザーアカウントやアイデンティティ情報を効率的に管理できるようにすることを目的としています。

5.4.2. 主な機能

AgileWorks の SCIMAPI では次の機能を提供します。

#	機能名	概要
5	ユーザー管理	ユーザーアカウントを作成、更新、削除、検索するための機能
6	組織管理	組織を作成、更新、削除、検索するための機能
7	リソース種別	各スキーマで使用可能なリソースとその使用目的を参照する SCIM サーバーがサポートするリソースタイプ（User、Group）に関する情報を提供するためのエンドポイントです。
8	スキーマ定義	受け入れ可能なリソースを取得する SCIM サーバーがサポートするスキーマ（データモデル）を定義し、それに関連する属性やメタデータを提供するためのエンドポイントです。
9	サービス定義	サービスの詳細を取得する SCIM サーバがサポートするサービスプロバイダ構成情報に関する情報を提供するためのエンドポイントです。

各 API の詳細はリファレンスガイドをご確認ください。

6. サンプルプログラム

前章までの各 API を用いた基本的なプログラミングのサンプルを紹介します。
使用言語は Java、JavaScript、Python の 3 種類です。
本章で用意するサンプルはあくまでサンプルという位置付けで提供しているため、業務ロジックにおいて共通関数化・ク
ラス化すべき部分の最適化は行っておらず、必ずしもベストコーディングといえない方法を取っている箇所があります。
あくまでもサンプルであるという位置付けで参考程度に確認ください。

6.1. 書類 API　サンプルプログラム

記載されているサンプルプログラムは
・リクエスト、レスポンスボディは JSON 形式
・リクエストボディはプログラム内記述、もしくは json ファイルの読み込み
・レスポンスボディを json ファイルとして出力する
以上が共通事項となっています。
リクエスト、レスポンスボディの詳細に関しては各種リファレンスを確認してください。

6.1.1. 書類データを取得する（Java）

任意の書類データを取得する例です。
「書類情報取得 API」を使用します。

▼ 使用するリクエストボディ例

```
1
```

※取得したい書類の ID を指定してください。
　プログラム内で指定する場合、json ファイルの作成は不要です。
　今回のサンプルではプログラム内で指定しています。

▼ 実行例

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.net.HttpURLConnection;
import java.net.URL;

public class sample_1_1 {
	public static void main(String[] args){

		// URL 例
		final String baseUrl = "https://sample.atled.jp/AgileWorks/Broker/WebApi/Service";
		final String queryParameter = "?name=doc.Doc&method=getDoc";
		final String postUrl = baseUrl + queryParameter;

		// リクエストボディ 例
		final String requestBody = "17";

		// 出力先
		final String filePath = "c:/tmp/sample_1_1.json";

		try{
```

```
            final String result = execute(postUrl, requestBody);
            System.out.println(result);
            output(result, filePath);
        }catch (Exception e) {
            System.out.println("ERROR");
            return;
        }
    }

    public static String execute(final String postUrl, final String requestBody) throws IOException{

        // 接続オブジェクトを生成する
        final URL url = new URL(postUrl);
        final HttpURLConnection conn = (HttpURLConnection) url.openConnection();

        // 各種設定
        // HTTP のメソッドを POST に設定する
        conn.setRequestMethod("POST");
        // リクエストボディへの書き込みを許可する
        conn.setDoInput(true);
        // レスポンスボディの取得を許可する
        conn.setDoOutput(true);
        // リクエスト形式を Json に指定する
        conn.setRequestProperty("Content-Type", "application/json; charset=utf-8");
        // 認証方式を指定する例 (Basic 認証の場合)
        conn.setRequestProperty("Authorization", "admin:agileworks");
        // 認証方式を指定する例 (OAuth2 認証の場合)
        // conn.setRequestProperty("Authorization", "Bearer <アクセストークン>");
        // リクエストボディに json 文字列を書き込む
        final PrintStream ps = new PrintStream(conn.getOutputStream());
        ps.print(requestBody);
        ps.close();

        conn.connect();

        // HttpURLConnection から InputStream を取得し、読み出す
        final BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream(),
"UTF-8"));

        final StringBuilder result = new StringBuilder();
        String line;

        while ((line = br.readLine()) != null) {
            result.append(line);
        }

        return result.toString();
    }

    public static void output(final String result, final String file){
        try {
            FileWriter fw = new FileWriter(file, false);
            PrintWriter pw = new PrintWriter(new BufferedWriter(fw));
            pw.print(result);
            pw.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

6.1.2. 書類データを更新する（JavaScript）

任意の書類データを取得する例です。
「書類情報取得 API」と「書類データ更新 API」を使用します。

▼ 使用するリクエストボディ

「書類情報取得 API」で取得したレスポンスボディの doc 項目

※「書類情報取得 API」で取得したレスポンスボディを書き換えて使用します。
　今回のサンプルでは json ファイルを読み込んで使用しています。

▼ 実行例

```
// URL 例
const baseUrl = "https://sample.atled.jp/AgileWorks/Broker/WebApi/Service";
const queryParameter = "?name=doc.Doc&method=updateDoc";
const postUrl = baseUrl + queryParameter;

// ファイル出力先 例
const filePath = "c:/tmp/sample_1_2.json";

// リクエストボディ 例
const fs = require("fs");
let jsonData = JSON.parse(fs.readFileSync('C:/tmp/request_1_2.json', 'utf-8'));

// HTTP リクエスト
let XMLHttpRequest = require("xmlhttprequest").XMLHttpRequest;
let request = new XMLHttpRequest();
request.open('POST', postUrl);

// HTTP リクエストヘッダーに設定
// 認証情報例 (Basic 認証の場合)
request.setRequestHeader('Authorization', 'admin:agileworks');
// 認証情報例 (OAuth2 認証の場合)
// request.setRequestHeader('Authorization', 'Bearer <アクセストークン>');
// コンテンツタイプの指定
request.setRequestHeader('Content-Type', 'application/json; charset=UTF-8');

request.onload = function() {
	if (this.readyState==4) {
		let result = request.responseText;
		console.log(result);
		let objData = JSON.parse(result);
		let outputData = JSON.stringify(objData)
		// ファイル出力
		fs.writeFileSync(filePath, outputData);
	}
};
request.responseType = 'json';
request.send(JSON.stringify(jsonData));
```

6.1.3. 書類 PDF データをダウンロードする（Python）

任意の書類を PDF 形式で取得する例です。
「書類 PDF ファイル取得 API」を使用します。

▼ 使用するリクエストボディ例

```
{
   "instructionList" : {
      "entries" : [ {
          "docId" : "3",
          "isInsertTrail" : "true"
       } ]
   }
}
```

※取得したい書類の ID を docId に指定してください。
　プログラム内で指定する場合、json ファイルの作成は不要です。
　今回のサンプルではプログラム内で指定しています。

　isInsertTrail は証跡 PDF を挿入するかどうかの項目です。（省略可能）
　ファイル名はレスポンスヘッダから取得可能ですが「join.pdf」固定のため、重複には注意してください。

▼ 実行例

```
from wsgiref import headers
import requests
import json
import pprint

def main():
   baseUrl = "https://sample.atled.jp/AgileWorks/Broker/WebApi/Service"
   queryParameter = "?name=doc.Doc&method=getDocPdf"
   postUrl = baseUrl + queryParameter

   # リクエストヘッダの指定例（Basic 認証の場合）
   requestHeaders = {"Authorization": "admin:agileworks", "Content-Type": "application/json"}
   # リクエストヘッダの指定例（OAuth2 認証の場合）
   # requestHeaders = {"Authorization": "Bearer <アクセストークン>", "Content-Type": "application/json"}

   # リクエストボディの作成
   requestBody = {
      "instructionList" : {
         "entries" : [ {
            "docId" : "10"
         } ]
      }
   }

   # レスポンスの取得
   response = requests.post(postUrl, headers=requestHeaders, json=requestBody)

   # レスポンスヘッダからファイル名を抽出（join.pdf 固定）
   fileName = re.search(r'=(.*)', response.headers['Content-Disposition'])

   # 取得した PDF ファイルを保存
   filePath = "c:/tmp/" + fileName.group(1)
   # filePath = "c:/tmp/sample_1_3.pdf"
   with open(filePath, mode="wb") as saveFile:
      # PDF ファイル取得 API はレスポンスがバイナリなので content を指定
      saveFile.write(response.content)

if __name__ == '__main__':
      main()
```

6.1.4. 書類コメント/メモを取得する（Java）

任意の書類データのコメント一覧を取得する例です。
「書類コメント/メモ一覧取得 API」を使用します。

▼ 使用するリクエストボディ例

```
10
```

※コメントを取得したい書類の ID を指定してください。
　プログラム内で指定する場合、json ファイルの作成は不要です。
　今回のサンプルではプログラム内で指定しています。

▼ 実行例

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.net.HttpURLConnection;
import java.net.URL;

public class sample_1_4 {
	public static void main(String[] args){

		// URL 例
		final String baseUrl = "https://sample.atled.jp/AgileWorks/Broker/WebApi/Service";
		final String queryParameter = "?name=doc.Doc&method=listDocComment";
		final String postUrl = baseUrl + queryParameter;

		// リクエストボディ 例
		final String requestBody = "10";

		// 出力先
		final String filePath = "c:/tmp/sample_1_4.json";

		try{
			final String result = execute(postUrl, requestBody);
			System.out.println(result);
			output(result, filePath);
		}catch (Exception e) {
			System.out.println("ERROR");
			return;
		}
	}

	public static String execute(final String postUrl, final String requestBody) throws IOException{

		// 接続オブジェクトを生成する
		final URL url = new URL(postUrl);
		final HttpURLConnection conn = (HttpURLConnection) url.openConnection();

		// 各種設定
		// HTTP のメソッドを POST に設定する
		conn.setRequestMethod("POST");
		// リクエストボディへの書き込みを許可する
		conn.setDoInput(true);
		// レスポンスボディの取得を許可する
		conn.setDoOutput(true);
		// リクエスト形式を Json に指定する
		conn.setRequestProperty("Content-Type", "application/json; charset=utf-8");
		// 認証方式を指定する例 (Basic 認証の場合)
		conn.setRequestProperty("Authorization", "admin:agileworks");
		// 認証方式を指定する例 (OAuth2 認証の場合)
```

```
            // conn.setRequestProperty("Authorization", "Bearer <アクセストークン>");
            //  リクエストボディに json 文字列を書き込む
            final PrintStream ps = new PrintStream(conn.getOutputStream());
            ps.print(requestBody);
            ps.close();

            conn.connect();

            // HttpURLConnection から InputStream を取得し、読み出す
            final BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream(),
"UTF-8"));

            final StringBuilder result = new StringBuilder();
            String line;

            while ((line = br.readLine()) != null) {
                result.append(line);
            }

            return result.toString();
        }

        public static void output(final String result, final String file){
            try {
                FileWriter fw = new FileWriter(file, false);
                PrintWriter pw = new PrintWriter(new BufferedWriter(fw));
                pw.print(result);
                pw.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
}
```

6.1.5. 添付ファイルをダウンロードする（Python）

任意の書類に添付されているファイルを取得する例です。
「書類添付情報一覧取得 API」と「ダウンロード API」を使用します。

▼ 使用するリクエストボディ例

```
10
```

※取得したいファイルが添付されている書類の ID を指定してください。
　プログラム内で指定する場合、json ファイルの作成は不要です。
　今回のサンプルではプログラム内で指定しています。

▼ 実行例（添付ファイル情報を取得する）

```
from wsgiref import headers
import requests
import json
import pprint

def main():
    baseUrl = "https://sample.atled.jp/AgileWorks/Broker/WebApi/Service"
    queryParameter = "?name=doc.Doc&method=listDocAttachment"
    postUrl = baseUrl + queryParameter

    # リクエストヘッダの指定例（Basic 認証の場合）
    requestHeaders = {"Authorization": "admin:agileworks", "Content-Type": "application/json"}
    # リクエストヘッダの指定例（OAuth2 認証の場合）
    # requestHeaders = {"Authorization": "Bearer <アクセストークン>", "Content-Type": "application/json"}

    # リクエストボディの作成
    requestBody = 4

    response = requests.post(postUrl, headers=requestHeaders, json=requestBody)
    responseData = response.json()
    pprint.pprint(responseData)

    # レスポンスボディを保存
    filePath = "c:/tmp/sample_1_5.json"
    with open(filePath, mode="w") as saveFile:
        json.dump(responseData, saveFile, ensure_ascii=False, indent=4)

if __name__ == '__main__':
        main()
```

書類添付情報一覧取得 API の実行で得たレスポンスのうち、’storageFile’フィールドの’id’が添付ファイルのダウンロードに必要な情報となっています。
‘name’（ファイル名）を確認して必要な id を取得してください。
レスポンスヘッダからファイル名を抽出する際は、ファイル名に応じて URL デコードが必要な場合があります。

▼ 実行例 (添付ファイルをダウンロードする)

```
import re
from wsgiref import headers
import requests
import json
import pprint

def main():
    baseUrl = "https://sample.atled.jp/AgileWorks/Broker/WebApi/Download"
    queryParameter = "?s=<取得したファイル ID>"
    getUrl = baseUrl + queryParameter

    # リクエストヘッダの指定例 (Basic 認証の場合)
    requestHeaders = {"Authorization": "admin:agileworks", "Content-Type": "application/json"}
    # リクエストヘッダの指定例 (OAuth2 認証の場合)
    # requestHeaders = {"Authorization": "Bearer <アクセストークン>", "Content-Type": "application/json"}

    # レスポンスの取得
    response = requests.post(getUrl, headers=requestHeaders)

    # レスポンスヘッダからファイル名を抽出
    fileName = re.search(r"(?<=¥").*?(?=¥")", response.headers['Content-Disposition'])
    filePath = "c:/tmp/" + fileName.group()

    # レスポンスボディを保存
    with open(filePath, mode="wb") as saveFile:
        # ダウンロード API はレスポンスがバイナリなので content を指定
        saveFile.write(response.content)

if __name__ == '__main__':
        main()
```

6.2. ワークフローAPI　サンプルプログラム

6.2.1. 書類を申請する（Java）

任意の書類を申請する例です。
「新規書類データ作成 API 」「新規書類データ保存 API 」「書類作成/申請 API」を使用します。
一時保存や編集を行いたい場合は「新規書類データ保存 API」や「書類データ更新 API」を使用します。

▼ 使用するリクエストボディ例

```
{
   "docId" : 454,
   "applyUnitCode" : "AG011123",
   "applyUserCode" : "u001",
   "operateUserCode" : "u001",
   "formCode" : "mtg_record",
   "ruleCode" : "mtg_record_rule"
}
```

※申請したい書類の ID（事前に他 API で取得）を docId に指定してください。
　プログラム内で指定する場合、json ファイルの作成は不要です。
　今回のサンプルではプログラム内で指定しています。

▼ 実行例

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.net.HttpURLConnection;
import java.net.URL;

public class sample_2_1 {
	public static void main(String[] args){

		// URL 例
		final String baseUrl = "https://sample.atled.jp/AgileWorks/Broker/WebApi/Service";
		final String queryParameter = "?name=workflow.Workflow&method=start";
		final String postUrl = baseUrl + queryParameter;

		// リクエストボディ 例
		final String requestBody = "{" +
			" ¥"docId¥" : 7,"+
			" ¥"applyUnitCode¥" : ¥"AG000000¥"," +
			" ¥"applyUserCode¥" : ¥"u001¥"," +
			" ¥"operateUserCode¥" : ¥"u001¥"," +
			" ¥"formCode¥" : ¥"CheckForm¥"," +
			" ¥"ruleCode¥" : ¥"test¥"" +
		"}";

		// 出力先
		final String filePath = "c:/tmp/sample_2_1.json";

		try{
			final String result = execute(postUrl, requestBody);
			System.out.println(result);
			output(result, filePath);
		}catch (Exception e) {
			System.out.println("ERROR");
			return;
		}
	}

	public static String execute(final String postUrl, final String requestBody) throws IOException{
```

```
            // 接続オブジェクトを生成する
            final URL url = new URL(postUrl);
            final HttpURLConnection conn = (HttpURLConnection) url.openConnection();

            // 各種設定
            // HTTP のメソッドを POST に設定する
            conn.setRequestMethod("POST");
            // リクエストボディへの書き込みを許可する
            conn.setDoInput(true);
            // レスポンスボディの取得を許可する
            conn.setDoOutput(true);
            // リクエスト形式を Json に指定する
            conn.setRequestProperty("Content-Type", "application/json; charset=utf-8");
            // 認証方式を指定する例 (Basic 認証の場合)
            conn.setRequestProperty("Authorization", "admin:agileworks");
            // 認証方式を指定する例 (OAuth2 認証の場合)
            // conn.setRequestProperty("Authorization", "Bearer <アクセストークン>");
            // リクエストボディに json 文字列を書き込む
            final PrintStream ps = new PrintStream(conn.getOutputStream());
            ps.print(requestBody);
            ps.close();

            conn.connect();

            // HttpURLConnection から InputStream を取得し、読み出す
            final BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream(),
"UTF-8"));

            final StringBuilder result = new StringBuilder();
            String line;

            while ((line = br.readLine()) != null) {
                result.append(line);
            }

            return result.toString();
        }

        public static void output(final String result, final String file){
            try {
                FileWriter fw = new FileWriter(file, false);
                PrintWriter pw = new PrintWriter(new BufferedWriter(fw));
                pw.print(result);
                pw.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
}
```

6.2.2. 書類を承認する（JavaScript）

任意の書類を承認する例です。
「書類承認 API」を使用します。

▼ 使用するリクエストボディ例

```
{
   "docId" : 453,
   "operatorCode" : "u002",
   "ruleStepCode" : "STEP3",
   "comment" : "OK です"
}
```

※承認したい書類の ID を docId に指定してください。
　プログラム内で指定する場合、json ファイルの作成は不要です。
　今回のサンプルではプログラム内で指定しています。

▼ 実行例

```
// URL 例
const baseUrl = "https://sample.atled.jp/AgileWorks/Broker/WebApi/Service";
const queryParameter = "?name=workflow.Workflow&method=approve";
const postUrl = baseUrl + queryParameter;

// ファイル出力先 例
const filePath = "c:/tmp/sample_2_2.json";

// リクエストボディ 例
var jsonData = {
      'docId' : 7,
      'operatorCode' : 'u002',
      'ruleStepCode' : 'STEP2',
      'comment' : 'API で承認'
};

// HTTP リクエスト
let XMLHttpRequest = require("xmlhttprequest").XMLHttpRequest;
let request = new XMLHttpRequest();
request.open('POST', postUrl);

// HTTP リクエストヘッダーに設定
// 認証情報例 (Basic 認証の場合)
request.setRequestHeader('Authorization', 'admin:agileworks');
// 認証情報例 (OAuth2 認証の場合)
// request.setRequestHeader('Authorization', 'Bearer <アクセストークン>');
// コンテンツタイプの指定
request.setRequestHeader('Content-Type', 'application/json; charset=UTF-8');

request.onload = function() {
      if (this.readyState==4) {
            let result = request.responseText;
            console.log(result);
            let objData = JSON.parse(result);
            let outputData = JSON.stringify(objData)
            // ファイル出力
            fs.writeFileSync(filePath, outputData);
      }
};
request.responseType = 'json';
request.send(JSON.stringify(jsonData));
```

6.3. 組織 API　サンプルプログラム

6.3.1. ユーザー情報を取得する（Java）

特定のユーザー情報を取得する例です。
「ユーザー参照 API」を使用します。

▼ 使用するリクエストボディ例

```
{
   "condition" : {
      "code" : "u001",
      "criterionDate" : "2020-10-13 00:00:00 JST"
   }
}
```

※取得したいユーザーの各種条件を指定してください。
　プログラム内で指定する場合、json ファイルの作成は不要です。
　今回のサンプルではプログラム内で指定しています。

▼ 実行例

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.net.HttpURLConnection;
import java.net.URL;

public class sample_3_1 {
	public static void main(String[] args){

		// URL 例
		final String baseUrl = "https://sample.atled.jp/AgileWorks/Broker/WebApi/Service";
		final String queryParameter = "?name=org.Org&method=findUser";
		final String postUrl = baseUrl + queryParameter;

		// リクエストボディ 例
		final String requestBody = "{" +
			" ¥"condition¥" : {"+
				" ¥"code¥" : ¥"u001¥","+
				" ¥"criterionDate¥" : ¥"2022-07-01 00:00:00 JST¥"" +
			"}"+
		"}";

		// 出力先
		final String filePath = "c:/tmp/sample_3_1.json";

		try{
			final String result = execute(postUrl, requestBody);
			System.out.println(result);
			output(result, filePath);
		}catch (Exception e) {
			System.out.println("ERROR");
			return;
		}
	}

	public static String execute(final String postUrl, final String requestBody) throws IOException{

		// 接続オブジェクトを生成する
		final URL url = new URL(postUrl);
		final HttpURLConnection conn = (HttpURLConnection) url.openConnection();
```

```
            // 各種設定
            // HTTP のメソッドを POST に設定する
            conn.setRequestMethod("POST");
            // リクエストボディへの書き込みを許可する
            conn.setDoInput(true);
            // レスポンスボディの取得を許可する
            conn.setDoOutput(true);
            // リクエスト形式を Json に指定する
            conn.setRequestProperty("Content-Type", "application/json; charset=utf-8");
            // 認証方式を指定する例 (Basic 認証の場合)
            conn.setRequestProperty("Authorization", "admin:agileworks");
            // 認証方式を指定する例 (OAuth2 認証の場合)
            // conn.setRequestProperty("Authorization", "Bearer <アクセストークン>");
            // リクエストボディに json 文字列を書き込む
            final PrintStream ps = new PrintStream(conn.getOutputStream());
            ps.print(requestBody);
            ps.close();

            conn.connect();

            // HttpURLConnection から InputStream を取得し、読み出す
            final BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream(),
"UTF-8"));

            final StringBuilder result = new StringBuilder();
            String line;

            while ((line = br.readLine()) != null) {
                result.append(line);
            }

            return result.toString();
        }

        public static void output(final String result, final String file){
            try {
                FileWriter fw = new FileWriter(file, false);
                PrintWriter pw = new PrintWriter(new BufferedWriter(fw));
                pw.print(result);
                pw.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
}
```

6.3.2. ユーザー情報を更新する（JavaScript）

特定のユーザー情報を更新する例です。
「ユーザー作成 API」または「ユーザー参照 API」と「ユーザー更新 API」を使用します。

▼ 使用するリクエストボディ例

「ユーザー作成 API」または「ユーザー参照 API」で取得したレスポンスボディの user 項目

※更新したいユーザーの情報を事前に取得し、指定してください。
　プログラム内で指定する場合、json ファイルの作成は不要です。
　今回のサンプルでは json ファイルを読み込んで使用しています。

▼ 実行例

```
// URL 例
const baseUrl = "https://sample.atled.jp/AgileWorks/Broker/WebApi/Service";
const queryParameter = "?name=org.Org&method=updateUser";
const postUrl = baseUrl + queryParameter;

// ファイル出力先 例
const filePath = "c:/tmp/sample_3_2.json";

// リクエストボディ 例
const fs = require("fs");
let jsonData = JSON.parse(fs.readFileSync('C:/tmp/request_3_2.json', 'utf-8'));

// HTTP リクエスト
let XMLHttpRequest = require("xmlhttprequest").XMLHttpRequest;
let request = new XMLHttpRequest();
request.open('POST', postUrl);

// HTTP リクエストヘッダーに設定
// 認証情報例 (Basic 認証の場合)
request.setRequestHeader('Authorization', 'admin:agileworks');
// 認証情報例 (OAuth2 認証の場合)
// request.setRequestHeader('Authorization', 'Bearer <アクセストークン>');
// コンテンツタイプの指定
request.setRequestHeader('Content-Type', 'application/json; charset=UTF-8');

request.onload = function() {
    if (this.readyState==4) {
        let result = request.responseText;
        console.log(result);
        let objData = JSON.parse(result);
        let outputData = JSON.stringify(objData)
        // ファイル出力
        fs.writeFileSync(filePath, outputData);
    }
};
request.responseType = 'json';
request.send(JSON.stringify(jsonData));
```

6.3.3. 組織を作成する（Python）

組織を新規作成する例です。
「組織作成 API」を使用します。

▼ 使用するリクエストボディ例

```
{
      "code" : "testUnit007",
      "name" : "新組織",
      "validityDateFrom" : "2022-07-01 00:00:00 JST",
      "validityDateTo" : "2022-08-31 00:00:00 JST",
      "availableDateFrom" : "2022-07-01 00:00:00 JST",
      "availableDateTo" : "2022-08-31 00:00:00 JST"
}
```

※作成したい組織の各種情報を指定してください。
　プログラム内で指定する場合、json ファイルの作成は不要です。
　今回のサンプルではプログラム内で指定しています。

▼ 実行例

```
from wsgiref import headers
import requests
import json
import pprint

def main():
   baseUrl = "https://sample.atled.jp/AgileWorks/Broker/WebApi/Service"
   queryParameter = "?name=org.Org&method=addUnit"
   postUrl = baseUrl + queryParameter

   # リクエストヘッダの指定例（Basic 認証の場合）
   requestHeaders = {"Authorization": "admin:agileworks", "Content-Type": "application/json"}
   # リクエストヘッダの指定例（OAuth2 認証の場合）
   # requestHeaders = {"Authorization": "Bearer <アクセストークン>", "Content-Type": "application/json"}

   # リクエストボディの作成
   requestBody = {
      "code" : "testUnit007",
      "name" : "新組織",
      "validityDateFrom" : "2022-07-01 00:00:00 JST",
      "validityDateTo" : "2022-08-31 00:00:00 JST",
      "availableDateFrom" : "2022-07-01 00:00:00 JST",
      "availableDateTo" : "2022-08-31 00:00:00 JST"
   }

   # レスポンスの取得
   response = requests.post(postUrl, headers=requestHeaders, json=requestBody)
   responseData = response.json()
   pprint.pprint(responseData)

   # レスポンスボディを保存
   filePath = "c:/tmp/sample_3_5.json"
   with open(filePath, mode="w") as saveFile:
      json.dump(responseData, saveFile, ensure_ascii=False, indent=4)

if __name__ == '__main__':
      main()
```

6.3.4. 組織所属を作成する（Java）

組織の所属情報を新規作成する例です。
「組織所属作成 API」を使用します。

▼ 使用するリクエストボディ例

```
{
   "unitCode" : "AG000000",
   "userCode" : "u001",
   "availableDateFrom" : "2022-07-07 00:00:00 JST",
   "availableDateTo" : "2022-08-31 00:00:00 JST"
}
```

※組織所属情報を作成したいユーザーと組織のコードを指定してください。
　プログラム内で指定する場合、json ファイルの作成は不要です。
　今回のサンプルではプログラム内で指定しています。

▼ 実行例

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.net.HttpURLConnection;
import java.net.URL;

public class sample_3_4 {
	public static void main(String[] args){

		// URL 例
		final String baseUrl = "https://sample.atled.jp/AgileWorks/Broker/WebApi/Service";
		final String queryParameter = "?name=org.Org&method=addUnitAppointment";
		final String postUrl = baseUrl + queryParameter;

		// リクエストボディ 例
		final String requestBody = "{" +
			" ¥"unitCode¥" : ¥"testUnit007¥","+
			" ¥"userCode¥" : ¥"user005¥"," +
			" ¥"sectionRoleCode¥" : ¥"01¥","+
			" ¥"availableDateFrom¥" : ¥"2022-07-07 00:00:00 JST¥","+
			" ¥"availableDateTo¥" : ¥"2022-08-07 00:00:00 JST¥""+
		"}";

		// 出力先
		final String filePath = "c:/tmp/sample_3_4.json";

		try{
			final String result = execute(postUrl, requestBody);
			System.out.println(result);
			output(result, filePath);
		}catch (Exception e) {
			System.out.println("ERROR");
			return;
		}
	}

	public static String execute(final String postUrl, final String requestBody) throws IOException{

		// 接続オブジェクトを生成する
		final URL url = new URL(postUrl);
		final HttpURLConnection conn = (HttpURLConnection) url.openConnection();

		// 各種設定
		// HTTP のメソッドを POST に設定する
```

```
            conn.setRequestMethod("POST");
            // リクエストボディへの書き込みを許可する
            conn.setDoInput(true);
            // レスポンスボディの取得を許可する
            conn.setDoOutput(true);
            // リクエスト形式を Json に指定する
            conn.setRequestProperty("Content-Type", "application/json; charset=utf-8");
            // 認証方式を指定する例 (Basic 認証の場合)
            conn.setRequestProperty("Authorization", "admin:agileworks");
            // 認証方式を指定する例 (OAuth2 認証の場合)
            // conn.setRequestProperty("Authorization", "Bearer <アクセストークン>");
            // リクエストボディに json 文字列を書き込む
            final PrintStream ps = new PrintStream(conn.getOutputStream());
            ps.print(requestBody);
            ps.close();

            conn.connect();

            // HttpURLConnection から InputStream を取得し、読み出す
            final BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream(),
"UTF-8"));

            final StringBuilder result = new StringBuilder();
            String line;

            while ((line = br.readLine()) != null) {
                result.append(line);
            }

            return result.toString();
        }

        public static void output(final String result, final String file){
            try {
                FileWriter fw = new FileWriter(file, false);
                PrintWriter pw = new PrintWriter(new BufferedWriter(fw));
                pw.print(result);
                pw.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
}
```