目次

改訂情報

変更年月日	変更内容
2014-12-01	初版
2020-04-01	第2版 下記を追加・変更しました ■ 「*クライアントアプリケーションからOAuth認証機能を利用する方法* 」に「code_challenge」、「code_challenge_method」、「code_verifier」を追加しました。

はじめに

本書の目的

本書ではOAuth認証機能を用いたプログラミング方法や注意点等について説明します。

対象読者

次の開発者を対象としてます。

- OAuth認証を用いて intra-mart Accel Platform 上のリソースを提供したい開発者
- OAuth認証を用いて intra-mart Accel Platform 上のリソースを利用したクライアントアプリケーションを作成したい開発者

本書の構成

本書は上記の対象読者に応じて次の３つの構成を取っています。

- *intra-mart Accel Platform 上のリソースの提供方法（Java ）*

  Java でOAuth認証を用いた intra-mart Accel Platform 上のリソースを提供する方法について説明します。

- *intra-mart Accel Platform 上の機能の提供方法（ スクリプト開発モデル ）*

  スクリプト開発モデル でOAuth認証を用いた intra-mart Accel Platform 上のリソースを提供する方法について説明します。

- *クライアントアプリケーションからOAuth認証機能を利用する方法*

  OAuth認証を用いた intra-mart Accel Platform 上のリソースの利用方法について説明します。

intra-mart Accel Platform 上のリソースの提供方法（ Java ）

intra-mart Accel Platform 上のリソースをOAuth認証を通して提供する場合の作業手順は以下のとおりです。

1. 提供するリソースのスコープ（アクセス範囲）を設定する。
2. リソースを提供するURLを設定する。
3. リソースを提供するアプリケーションを設定する。
4. 提供するリソースを実装する。

ステップ1：提供するリソースのスコープ（アクセス範囲）を設定する。

空の＜oauth-client-scope.xml＞ファイルを作成して、以下を入力し保存します。

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-client-scopes-config
 xmlns="http://intra-mart.co.jp/system/oauth/provider/client/scope/config/oauth-client-scopes-config"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://intra-mart.co.jp/system/oauth/provider/client/scope/config/oauth-client-scopes-config oauth-client-scopes-config.xsd ">

 <scopes>
  <scope id="account">
   <default-subject>アカウント情報へのアクセス</default-subject>
   <localizations>
    <localize locale="ja">
     <subject>アカウント情報へのアクセス</subject>
     <text>ユーザアカウント情報へのアクセスを許可します。</text>
    </localize>
   </localizations>
  </scope>
 </scopes>

</oauth-client-scopes-config>
```

コラム

国際化情報（localize）は intra-mart Accel Platform で利用するロケール分、設定してください。

詳しい設定については 「 設定ファイルリファレンス 」 - 「 クライアントのアクセス範囲設定 」を参照してください。

作成した＜oauth-client-scope.xml＞ファイルを、「conf/oauth-client-scopes-config」直下に配置します。

スコープ（アクセス範囲）の設定内容はユーザがクライアントアプリケーションにアクセス許可を行う際に表示されます。

次に、リソースを提供するURLを設定しスコープを関連付けます。

ステップ2：リソースを提供するURLを設定する。

空の＜oauth-client-resource.xml＞ファイルを作成して、以下を入力し保存します。

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-client-resources-config
 xmlns="http://intra-mart.co.jp/system/oauth/provider/client/resource/config/oauth-client-resources-config"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://intra-mart.co.jp/system/oauth/provider/client/resource/config/oauth-client-resources-config oauth-client-resources-config.xsd ">

 <client-resources>
  <client-resource id="user-account" path="/oauth/user/account" type="java" target="jp.co.intra_mart.sample.UserAccount">
   <scope id="account" />
  </client-resource>
 </client-resources>

</oauth-client-resources-config>
```

コラム

設定したリソースに認可設定を行う場合は<client-resource>の子要素に<authz>タグを設定してください。

リソースに関連付けられたスコープがクライアントアプリケーションに許可されていない場合、リソースへのアクセスは拒否されます。

詳しい設定については 「 設定ファイルリファレンス 」 - 「 クライアントリソース設定 」を参照してください。

作成した＜oauth-client-resource.xml＞ファイルを、「conf/oauth-client-resources-config」直下に配置します。

次に、設定したリソースを参照可能なクライアントアプリケーションを設定します。

ステップ3：リソースを提供するアプリケーションを設定する。

空の＜oauth-client-detail.xml＞ファイルを作成して、以下を入力し保存します。

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-client-details-config
 xmlns="http://intra-mart.co.jp/system/oauth/provider/client/config/oauth-client-details-config"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://intra-mart.co.jp/system/oauth/provider/client/config/oauth-client-details-config oauth-client-details-config.xsd ">

 <client-details>
  <client-detail client-id="account-sample" client-secret="sample" authorized-grant-type="authorization_code">
   <default-name>サンプル・アプリケーション</default-name>
   <localizations>
    <localize locale="ja">
     <client-name>サンプル・アプリケーション</client-name>
     <description>アカウント情報を取得するサンプルアプリケーションです。</description>
    </localize>
   </localizations>
   <scopes>
    <scope id="account" />
   </scopes>
  </client-detail>
 </client-details>

</oauth-client-details-config>
```

コラム

国際化情報（localize）は intra-mart Accel Platform で利用するロケール分、設定してください。

クライアントのなりすましを防ぐため、 redirect-uri にクライアントアプリケーションのエンドポイントを設定することを推奨します。

詳しい設定については 「 設定ファイルリファレンス 」 - 「 クライアント詳細設定 」を参照してください。

作成した＜oauth-client-detail.xml＞ファイルを、「conf/oauth-client-details-config」直下に配置します。

ステップ4：提供するリソースを実装する。

<oauth-client-resource.xml>ファイルの<client-resource target>に設定した「jp.co.intra_mart.sample.UserAccount」クラスを作成します。

OAuth認証を通して提供するリソースの実装クラスを作成する場合は、”jp.co.intra_mart.foundation.oauth.provider.resource.ResourceExecutor”インタフェースを実装します。

```
public class UserAccount implements ResourceExecutor<AccountContext> {
   @Override
   @ResponseType("application/json")
   public AccountContext execute(final HttpServletRequest request, final HttpServletResponse
response) throws OAuthException {
      return Contexts.get(AccountContext.class);
   }
}
```

@ResponseType を指定することで、ContentTypeを指定することができます。
@ResponseTypeに「application/json」を指定するとオブジェクトをJSONに、「text/xml」を指定するとオブジェクトをXMLにシリアライズしてレスポンスが返却されます。
それ以外の値を設定した場合は、オブジェクトがそのまま返却されます。

> **注意**
>
> オブジェクトをXMLにシリアライズする際には、JAXBデータ バインディングが利用されます。
> シリアライズするオブジェクトにはJAXBアノテーションによるクラス定義を行ってください。

以上でリソースの提供が可能になります。
アクセストークンを取得した後に、「*https://localhost:8080/imart/oauth/user/account?access_token=<access_token>*」へリクエストを送信すると以下のようなレスポンスが返却されます。

```
{
  "applicationLicenses": [],
  "authenticated": true,
  "calendarId": "JPN_CAL",
  "dateTimeFormats": {
    "IM_DATETIME_FORMAT_DATE_STANDARD": "yyyy/MM/dd",
    "IM_DATETIME_FORMAT_DATE_SIMPLE": "MM/dd",
    "locale": "ja",
    "IM_DATETIME_FORMAT_DATE_INPUT": "yyyy/MM/dd",
    "IM_DATETIME_FORMAT_TIME_INPUT": "HH:mm",
    "format-set-id": "IM_DATETIME_FORMAT_SET_JA_BASE",
    "IM_DATETIME_FORMAT_TIME_STANDARD": "H:mm",
    "IM_DATETIME_FORMAT_TIME_TIMESTAMP": "H:mm:ss"
  },
  "encoding": "UTF-8",
  "firstDayOfWeek": 1,
  "homeUrl": "/home",
  "locale": "ja",
  "loginGroupId": "default",
  "loginTime": 1375753818835,
  "roleIds": [],
  "signature": "1xcbox8",
  "themeId": "im_theme_dropdown_blue",
  "timeZone": "Asia/Tokyo",
  "type": "jp.co.intra_mart.foundation.context.model.AccountContext",
  "userCd": "aoyagi",
  "userType": 1
}
```

intra-mart Accel Platform 上の機能の提供方法（ スクリプト開発モデル ）

intra-mart Accel Platform 上のリソースをOAuth認証を通して提供する場合の作業手順は以下のとおりです。

1. 提供するリソースのスコープ（アクセス範囲）を設定する。
2. リソースを提供するURLを設定する。
3. リソースを利用するアプリケーションを設定する。
4. 提供するリソースを実装する。

ステップ1：提供するリソースのスコープ（アクセス範囲）を設定する。

空の＜oauth-client-scope.xml＞ファイルを作成して、以下を入力し保存します。

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-client-scopes-config
 xmlns="http://intra-mart.co.jp/system/oauth/provider/client/scope/config/oauth-client-scopes-config"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://intra-mart.co.jp/system/oauth/provider/client/scope/config/oauth-client-scopes-config oauth-client-scopes-config.xsd ">

 <scopes>
  <scope id="account">
   <default-subject>アカウント情報へのアクセス</default-subject>
   <localizations>
    <localize locale="ja">
     <subject>アカウント情報へのアクセス</subject>
     <text>ユーザアカウント情報へのアクセスを許可します。</text>
    </localize>
   </localizations>
  </scope>
 </scopes>

</oauth-client-scopes-config>
```

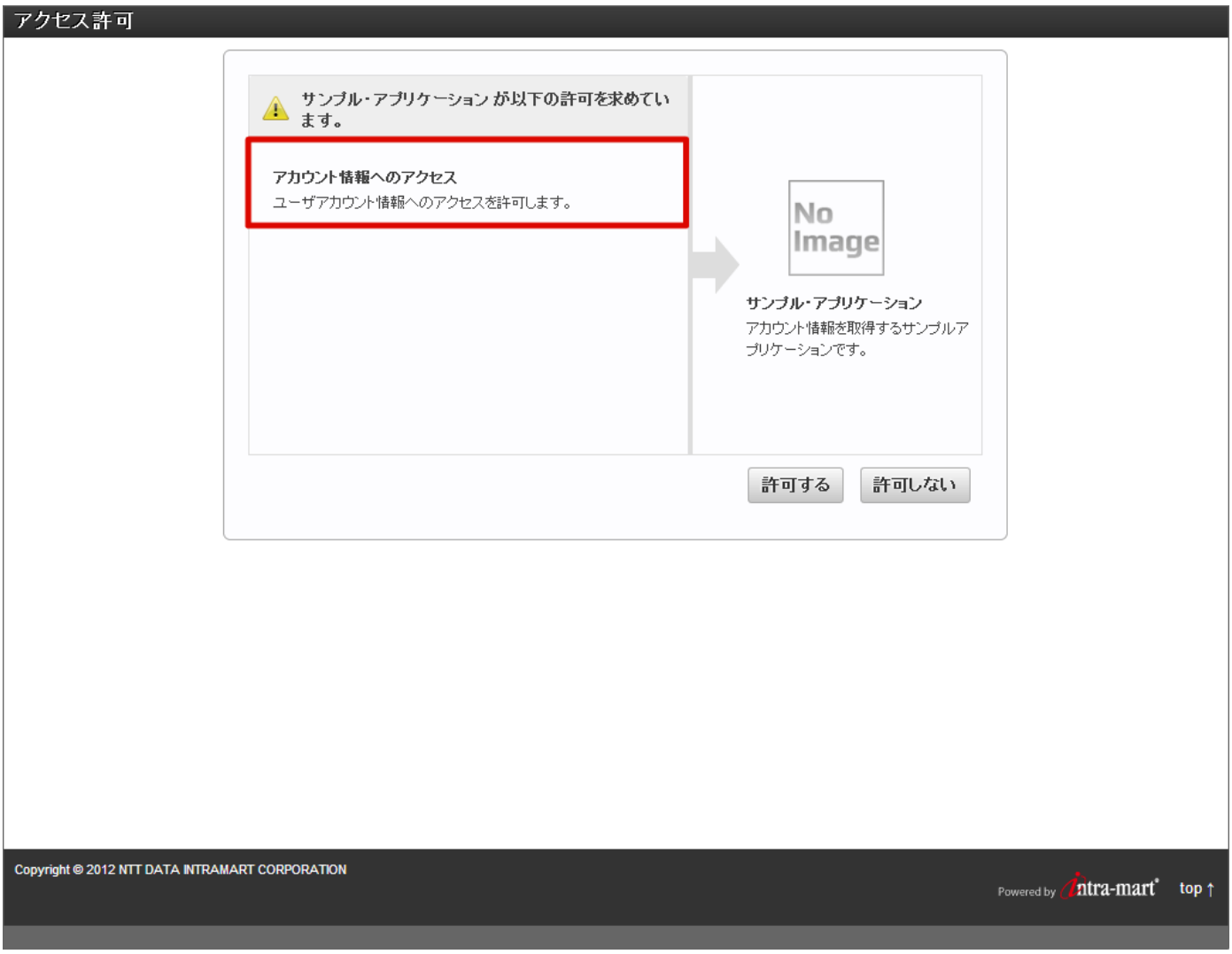
コラム

国際化情報（localize）は intra-mart Accel Platform で利用するロケール分、設定してください。

詳しい設定については 「 設定ファイルリファレンス 」 - 「 クライアントのアクセス範囲設定 」を参照してください。

作成した＜oauth-client-scope.xml＞ファイルを、「conf/oauth-client-scopes-config」直下に配置します。

スコープ（アクセス範囲）の設定内容はユーザがクライアントアプリケーションにアクセス許可を行う際に表示されます。



次に、リソースを提供するURLを設定しスコープを関連付けます。

ステップ2：リソースを提供するURLを設定する。

空の＜oauth-client-resource.xml＞ファイルを作成して、以下を入力し保存します。

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-client-resources-config
 xmlns="http://intra-mart.co.jp/system/oauth/provider/client/resource/config/oauth-client-resources-config"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://intra-mart.co.jp/system/oauth/provider/client/resource/config/oauth-client-resources-config oauth-client-resources-config.xsd ">

 <client-resources>
  <client-resource id="user-account" path="/oauth/user/account" type="jssp" target="sample/user_account">
   <scope id="account" />
  </client-resource>
 </client-resources>

</oauth-client-resources-config>
```

コラム

設定したリソースに認可設定を行う場合は<client-resource>の子要素に<authz>タグを設定してください。

リソースに関連付けられたスコープがクライアントアプリケーションに許可されていない場合、リソースへのアクセスは拒否されます。

詳しい設定については 「 設定ファイルリファレンス 」 - 「 クライアントリソース設定 」を参照してください。

作成した＜oauth-client-resource.xml＞ファイルを、「conf/oauth-client-resources-config」直下に配置します。

次に、設定したリソースを参照可能なクライアントアプリケーションを設定します。

ステップ3：リソースを利用するアプリケーションを設定する。

空の＜oauth-client-detail.xml＞ファイルを作成して、以下を入力し保存します。

```
<?xml version="1.0" encoding="UTF-8"?>
<oauth-client-details-config
 xmlns="http://intra-mart.co.jp/system/oauth/provider/client/config/oauth-client-details-config"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://intra-mart.co.jp/system/oauth/provider/client/config/oauth-client-details-config oauth-client-details-config.xsd ">

  <client-details>
    <client-detail client-id="account-sample" client-secret="sample" authorized-grant-type="authorization_code">
      <default-name>サンプル・アプリケーション</default-name>
      <localizations>
        <localize locale="ja">
          <client-name>サンプル・アプリケーション</client-name>
          <description>アカウント情報を取得するサンプルアプリケーションです。</description>
        </localize>
      </localizations>
      <scopes>
        <scope id="account" />
      </scopes>
    </client-detail>
  </client-details>

</oauth-client-details-config>
```

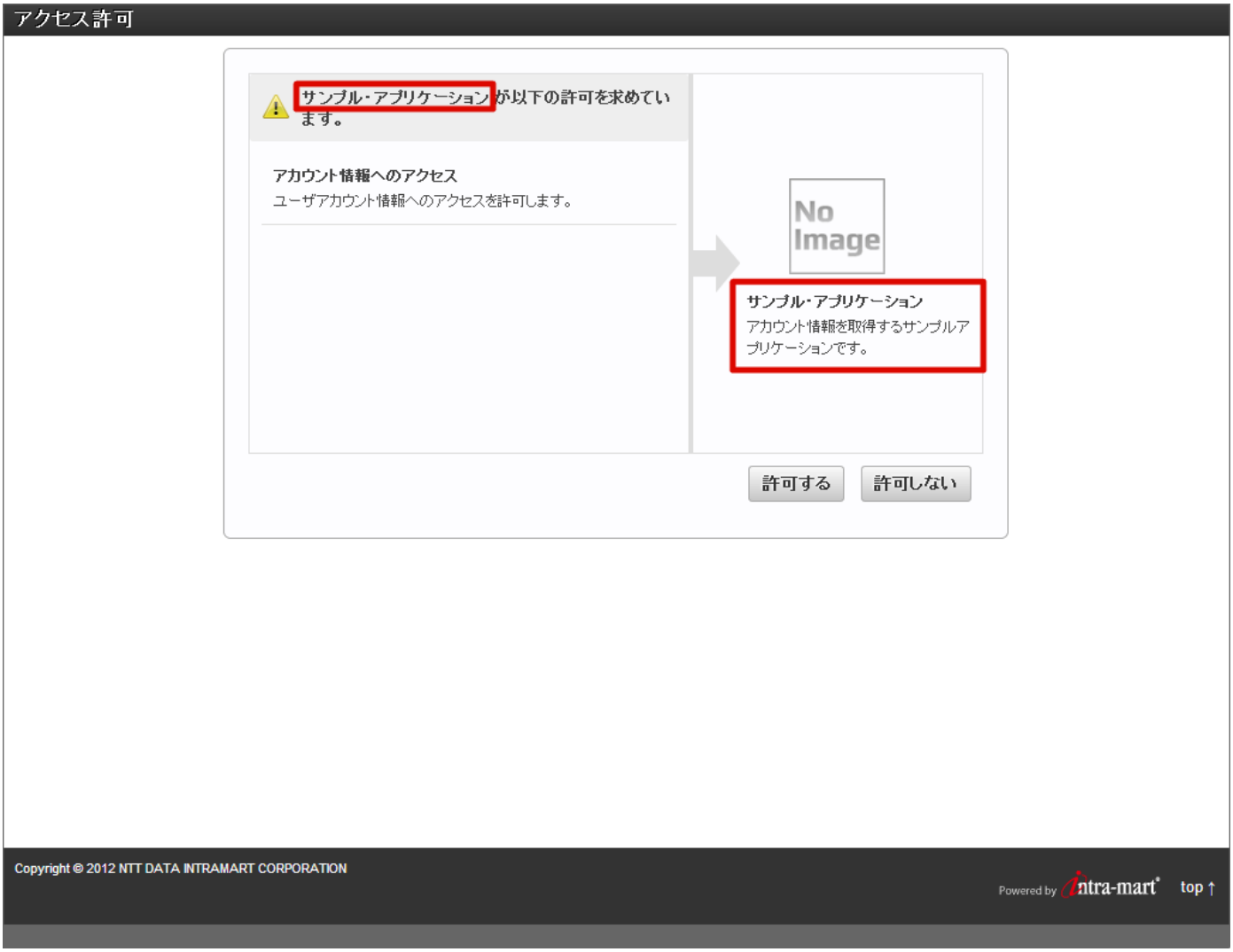
コラム

国際化情報（localize）は intra-mart Accel Platform で利用するロケール分、設定してください。

クライアントのなりすましを防ぐため、 redirect-uri にクライアントアプリケーションのエンドポイントを設定することを推奨します。

詳しい設定については 「 設定ファイルリファレンス 」 - 「 クライアント詳細設定 」を参照してください。

作成した＜oauth-client-detail.xml＞ファイルを、「conf/oauth-client-details-config」直下に配置します。



ステップ4：提供するリソースを実装する。

空の＜user_account.js＞ファイルを作成して、以下を入力し保存します。

```
function init(request) {
   var accountContext = Contexts.getAccountContext();
   var response = Web.getHTTPResponse();
   response.setContentType('application/json; charset=utf-8');
   response.sendMessageBodyString(ImJson.toJSONString(accountContext));
}
```

作成した＜user_account.js＞ファイルを、「jssp/src/sample」直下に配置します。

以上でリソースの提供が可能になります。
アクセストークンを取得した後に、「*https://localhost:8080/imart/oauth/user/account?access_token=<access_token>*」へリクエストを送信すると以下のようなレスポンスが返却されます。

```
{
  "applicationLicenses": [],
  "authenticated": true,
  "calendarId": "JPN_CAL",
  "dateTimeFormats": {
    "IM_DATETIME_FORMAT_DATE_STANDARD": "yyyy/MM/dd",
    "IM_DATETIME_FORMAT_DATE_SIMPLE": "MM/dd",
    "locale": "ja",
    "IM_DATETIME_FORMAT_DATE_INPUT": "yyyy/MM/dd",
    "IM_DATETIME_FORMAT_TIME_INPUT": "HH:mm",
    "format-set-id": "IM_DATETIME_FORMAT_SET_JA_BASE",
    "IM_DATETIME_FORMAT_TIME_STANDARD": "H:mm",
    "IM_DATETIME_FORMAT_TIME_TIMESTAMP": "H:mm:ss"
  },
  "encoding": "UTF-8",
  "firstDayOfWeek": 1,
  "homeUrl": "/home",
  "locale": "ja",
  "loginGroupId": "default",
  "loginTime": 1375753818835,
  "roleIds": [],
  "signature": "1xcbox8",
  "themeId": "im_theme_dropdown_blue",
  "timeZone": "Asia/Tokyo",
  "userCd": "aoyagi",
  "userType": 1
}
```

クライアントアプリケーションからOAuth認証機能を利用する方法

OAuth認証機能は開発しているアプリケーションのタイプによって使用する認証フローが異なります。
ここでは、アプリケーションのタイプ毎のOAuth認証機能の利用方法を説明します。

WebアプリケーションでOAuth認証機能を利用する方法

WebアプリケーションからOAuth認証機能を利用する場合は、 認可コードによる認可 フローを使用します。
その手順は以下の通りです。

1. フローはユーザがWebアプリケーションへリクエストを送信するところからはじまります。
2. Webアプリケーションは、ユーザのブラウザに intra-mart Accel Platform の認可エンドポイントへリダイレクトするレスポンスを返します。

```
https://localhost:8080/imart/oauth/authorize?response_type=code&client_id=
<client_id>&redirect_uri=<redirect_uri>
```

認可エンドポイントURIへのクエリーとして次のパラメータを付与します。

パラメータ名	必須項目	設定値	備考
response_type	○	認可コードを取得するためのリクエストであることを表す “code” を設定します。	
client_id	○	アプリケーションのクライアント識別子を指定します。	
redirect_uri	×	認可コードをリダイレクトするURIを指定します。	アプリケーションのリダイレクトURIに設定された値と一致しなければいけません。
scope	×	アプリケーションの要求するアクセス範囲を指定します。	アプリケーションリソースに設定されたアクセス範囲に一致しないければいけません。
state	×	リクエストとコールバックの間で状態を維持するために使用するランダムな値を指定します。	intra-mart Accel Platform から認可コードをアプリケーションへリダイレクトする際にこの値を付与します。
code_challenge	×	認可コードとアクセストークンを交換する際の検証に利用する値を指定します。	code_verifier　の値を code_challenge_method で指定した方法で計算して得られた値を指定します。

パラメータ名	必須項目	設定値	備考
code_challenge_method	×	code_verifier の値の計算方法を指定します。	`plain` または `S256` が指定可能です。

注意

stateパラメータにセッションに紐づく値を設定して、CSRF対策を行うことを強く推奨します。

注意

クライアントアプリケーションの設定で、コード交換用証明キー(PKCE)が要求されている場合、code_challenge、code_challenge_method は必須項目です。

コラム

code_challenge、code_challenge_method は 2020 Spring(Yorkshire) から追加されました。

3. 認可エンドポイントにアクセスすると、ユーザは認証を求められます。
（ブラウザ上で既に認証情報を保持している場合はユーザ認証はスキップされます。）

4. ユーザ認証を行うと、続いてクライアントアプリケーションの認可を求められます。

5. 認可されるとWebアプリケーションのリダイレクトURIへリダイレクトされます。

```
https://app.example.com/callback?code=5i7ruuubw9crhcd
```

Webアプリケーションはここで認可コードをURLパラメータから取得できます。
URLパラメータからは次のパラメータが取得できます。

パラメータ名	備考
code	発行された認可コードです。
state	認可エンドポイントへリダイレクトする際にstateパラメータを付与していた場合に取得可能です。

ユーザからアクセスを拒否された場合等、認証に失敗した場合は、URLパラメータにエラーコードが取得できます。

```
https://app.example.com/callback?error=access_denied
```

返却されるエラーコードについては、 *エラーコード一覧* を参照してください。

6. 認可コードを取得したら、 intra-mart Accel Platform のトークンエンドポイントへPOSTリクエストを送信します。

```
https://localhost:8080/imart/oauth/token?
grant_type=authorization_code&code=5i7ruuubw9crhcd&client_id=<client_id>&client_secret=
<client_secret>
```

トークンエンドポイントURIへのクエリーとして次のパラメータを付与します。

パラメータ名	必須項目	設定値	備考
grant_type	○	認可コードによる認可であることを表す “authorization_code” を設定します。	
code	○	1つ前のステップで取得した認可コードを指定します。	
client_id	○	アプリケーションのクライアント識別子を設定します。	
client_secret	○	アプリケーションのクライアントシークレットを設定します。	
redirect_uri	×	認可エンドポイントへリクエストを送信した際に指定したredirect_uriを設定します。	認可エンドポイントへリクエストを送信した際にredirect_uriを指定していた場合は必須です。
code_verifier	×	code_challenge の計算前の値を設定します。	認可エンドポイントへリクエストを送信した際にcode_challengeを指定していた場合は必須です。

コラム

code_verifier は 2020 Spring(Yorkshire) から追加されました。

intra-mart Accel Platform は以下のようにJSONでエンコードされたレスポンスを返してきます。

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
   "access_token": "718fedeab471477fa1c0deef626e0b0d",
   "token_type": "Bearer",
   "expires_in": 3600,
   "refresh_token": "5c7cc664da4a40e0a7103c3200509e01",
   "scope": "schedule"
}
```

パラメータ名	備考
access_token	発行されたアクセストークンです。
token_type	トークンの種別です。 intra-mart Accel Platform の標準実装では、ベアラートークンを返却します。
expires_in	アクセストークンの有効期間（秒）です。

パラメータ名	備考
refresh_token	アクセストークンの有効期限が切れた時に、新しいアクセストークンを取得するために利用されるリフレッシュトークンです。
scope	このアクセストークンで参照可能なリソースのアクセス範囲です。

認可コードの有効期限が過ぎていた場合等、クライアント認証に失敗した場合には、HTTP ステータスコード 400（Bad Request）を返却します。このレスポンスには、パラメータにエラーコードが含まれます。

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
   "error":"invalid_request"
}
```

返却されるエラーコードについては、 *エラーコード一覧* を参照してください。

7. 取得したアクセストークンを付与したリクエストを送信することで intra-mart Accel Platform 上のリソースへアクセスできます。

   アクセストークンをリクエストヘッダに付与します。

   ```
   GET /<resource_path> HTTP/1.1
   Host: localhost
   Authorization: Bearer 718fedeab471477fa1c0deef626e0b0d
   ```

   リソースへのアクセス時にアクセストークンを含んでいない場合やアクセストークンの認証に失敗した場合は、 WWW-Authenticate レスポンスヘッダを含んだレスポンスが返却されます。

   ```
   WWW-Authenticate: Bearer realm="OAuth Authorization",
                     error="invalid_token"
   ```

   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利用してアクセストークンの更新を行ってください。
   更新方法については 「 *アクセストークンの更新方法* 」を参照してください。

ネイティブアプリケーションでOAuth認証機能を利用する方法

ネイティブなモバイル、または、デスクトップアプリケーションのようなクライアントアプリケーションからOAuth認証機能を利用する場合は、 インプリシットグラント フローを使用します。
その手順は以下の通りです。

1. このフローでは、クライアントアプリケーションはユーザを認可エンドポイントへ向かわせます。

```
https://localhost:8080/imart/oauth/authorize?response_type=token&client_id=
<client_id>&redirect_uri=<redirect_uri>
```

認可エンドポイントURIへのクエリーとして次のパラメータを付与します。

パラメータ名	必須項目	設定値	備考
response_type	○	アクセストークンを取得するためのリクエストであることを表す “token” を設定します。	
client_id	○	アプリケーションのクライアント識別子を指定します。	
redirect_uri	×	アクセストークンをリダイレクトするURIを指定します。	アプリケーションのリダイレクトURIに設定された値と一致しなければいけません。
scope	×	アプリケーションの要求するアクセス範囲を指定します。	アプリケーションリソースに設定されたアクセス範囲に一致しないければいけません。
state	×	リクエストとコールバックの間で状態を維持するために使用するランダムな値を指定します。	intra-mart Accel Platform からアプリケーションへリダイレクトする際にこの値を付与します。

2. 認可エンドポイントにアクセスすると、ユーザは認証を求められます。
3. ユーザ認証を行うと、続いてクライアントアプリケーションの認可を求められます。
4. 認可されるとクライアントアプリケーションのリダイレクトURIにアクセストークン等のパラメータを含んだURLが返されます。
（パラメータはURLのハッシュ ’#’ サインの後に付与されます。）

```
myapp:callback#access_token=c0200e5c62334f7d93ec685478c40a11&token_type=Bearer&exp
```

リダイレクトURIに含まれるパラメータは以下のとおりです。

パラメータ名	備考
access_token	発行されたアクセストークンです。
token_type	トークンの種別です。 intra-mart Accel Platform の標準実装では、ベアラートークンを返却します。
expires_in	アクセストークンの有効期間（秒）です。
scope	このアクセストークンで参照可能なリソースのアクセス範囲です。

ユーザからアクセスを拒否された場合等、認証に失敗した場合は、パラメータにエラーコードが付与さ

れたURLが返却されます。

```
myapp:callback#error=access_denied
```

返却されるエラーコードについては、 *エラーコード一覧* を参照してください。

コラム

- フラグメントの中でこれらのパラメータが渡されるため、リダイレクト先へパラメータが渡されることはありません。

5. ネイティブアプリケーションは直接フラグメントに含まれる送られてきたパラメータをパースします。また、ブラウザベースのアプリケーションは、JavaScriptでリダイレクトURIのフラグメントとそのパラメータにアクセスして値を取得します。
6. リダイレクトURIに含まれるパラメータを取得したら、パラメータに含まれるアクセストークンを付与したリクエストを送信することで intra-mart Accel Platform 上のリソースへアクセスできます。

   アクセストークンをリクエストヘッダに付与します。

   ```
   GET /<resource_path> HTTP/1.1
   Host: localhost
   Authorization: Bearer 718fedeab471477fa1c0deef626e0b0d
   ```

   リソースへのアクセス時にアクセストークンを含んでいない場合やアクセストークンの認証に失敗した場合は、 WWW-Authenticate レスポンスヘッダを含んだレスポンスが返却されます。

   ```
   WWW-Authenticate: Bearer realm="OAuth Authorization",
                     error="invalid_token"
   ```

   アクセストークンの有効期限が切れた場合は、上記の手順で再度アクセストークンを取得する必要があります。

注意

インプリシットグラントフローを利用する場合は、Token置換攻撃の対策としてアクセストークンの確認を行うようにしてください。
確認方法については 「 *アクセストークンの確認方法* 」を参照してください。

アクセストークンの更新方法

取得したアクセストークンの有効期限が切れた場合、リフレッシュトークンを利用して新しいアクセストークンを取得できます。
その手順は以下の通りです。

1. クライアントアプリケーションは、 intra-mart Accel Platform のトークンエンドポイントへPOSTリクエストを送信します。

```
https://localhost:8080/imart/oauth/token?
grant_type=refresh_token&refresh_token=5c7cc664da4a40e0a7103c3200509e01&client_id=
<client_id>&client_secret=<client_secret>
```

トークンエンドポイントURIへのクエリーとして次のパラメータを付与します。

パラメータ名	必須項目	設定値	備考
grant_type	○	アクセストークンの更新のためのリクエストであることを表す“refresh_token” を設定します。	
refresh_token	○	アクセストークンを取得した時に取得したリフレッシュトークンを設定します。	
client_id	○	アプリケーションのクライアント識別子を設定します。	
client_secret	○	アプリケーションのクライアントシークレットを設定します。	
scope	×	アプリケーションの要求するアクセス範囲を指定します。	発行済みのアクセストークンのアクセス範囲内でなければいけません。

intra-mart Accel Platform は以下のようにJSONでエンコードされたレスポンスを返してきます。

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
   "access_token": "c0956bf8c90748fa85ef9356f54778dd",
   "token_type": "Bearer",
   "expires_in": 3600,
   "refresh_token": "309056c2c8da4f5b82ad5c4b7e272e54",
   "scope": "schedule"
}
```

パラメータ名	備考
access_token	発行されたアクセストークンです。
token_type	トークンの種別です。 intra-mart Accel Platform の標準実装では、ベアラートークンを返却します。
expires_in	アクセストークンの有効期間（秒）です。
refresh_token	アクセストークンの有効期限が切れた時に、新しいアクセストークンを取得するために利用されるリフレッシュトークンです。

パラメータ名	備考
scope	このアクセストークンで参照可能なリソースのアクセス範囲です。

リフレッシュトークンが不正な場合等、クライアント認証に失敗した場合には、HTTP ステータスコード 400（Bad Request）を返却します。このレスポンスには、パラメータにエラーコードが含まれます。

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
   "error":"invalid_request"
}
```

返却されるエラーコードについては、 *エラーコード一覧* を参照してください。

2. 取得したアクセストークンをパラメータに付与したリクエストを送信することで intra-mart Accel Platform 上のリソースへアクセスできます。

```
https://localhost:8080/imart/<resource_path>?
access_token=c0956bf8c90748fa85ef9356f54778dd
```

アクセストークンの有効期限が切れた場合は、上記の手順で再度アクセストークンの更新を行ってください。
この時に利用するリフレッシュトークンはアクセストークンを更新した時に取得したリフレッシュトークンを利用してください。
アクセストークンを更新する時に利用したリフレッシュトークンは利用できません。

アクセストークンの確認方法

インプリシットグラントフロー を利用している場合、認可サーバからのレスポンスに含まれるトークンを簡単に変更することが可能なため、Token置換攻撃が行われる可能性があります。
Token置換攻撃への対策としてクライアントアプリケーションは、受け取ったトークンが自分自身に発行されたものかどうか確認を行うべきです。

その手順は以下の通りです。

1. クライアントアプリケーションはアクセストークンをリクエストヘッダに付与し、 intra-mart Accel Platform のトークン確認エンドポイントへPOSTリクエストを送信します。

```
POST /imart/oauth/token/verify HTTP/1.1
Host: localhost
Authorization: Bearer c0200e5c62334f7d93ec685478c40a11
```

トークン確認エンドポイントURIへのクエリーとして次のパラメータを付与します。

パラメータ名	必須項目	設定値	備考
access_token	○	確認を行うアクセストークンを設定します。	

intra-mart Accel Platform は以下のようにJSONでエンコードされたレスポンスを返してきます。

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
   "audience": "sample_client",
   "user_cd": "aoyagi",
   "expires_in": 3510,
   "scope": "schedule"
}
```

パラメータ名	備考
audience	このアクセストークンの発行先クライアントIDです。
user_cd	ユーザコードです。
expires_in	アクセストークンの有効期間（秒）です。
scope	このアクセストークンで参照可能なリソースのアクセス範囲です。

取得したアクセストークンの内容に含まれる audience が自分自身のクライアントID と一致していることを確かめます。
audience の内容が自分自身のクラアントID と一致しない場合、そのアクセストークンは利用せず破棄してください。

エラーコード一覧

認証やリソース参照の失敗時に、intra-mart Accel Platform から返却されるエラーコードは以下の通りです。

- invalid_request

  リクエストに必要なパラメータが含まれていない場合や、パラメータの値が不正な場合に返却されます。

- invalid_client

  未知のクライアントである場合、クライアント認証情報が含まれていない場合、サポートされない認証方式が利用されている場合等、クライアントの認証に失敗した場合に返却されます。

- unauthorized_client

  現在の方法で認可コードを取得することを認可されていない、認証されたクライアントが当該のグラントタイプを利用する様に認可されていない場合に返却されます。

- access_denied

  ユーザにリクエストを拒否された場合に返却されます。

- unsupported_response_type

  指定されたレスポンスタイプがサポートされていない場合に返却されます。

- invalid_grant

  提供された認可グラントが不正、有効期限切れ、 失効している、認可リクエストで用いられたリダイレクト先URIとマッチしていない、他のクライアントに対して発行されたものである場合に返却されます。

- unsupported_grant_type

  指定されたグラントタイプがサポートされていない場合に返却されます。

- invalid_scope

  リクエストスコープが未知、または、その他の不当な形式である場合に返却されます。

- server_error

  リクエストの処理ができないような予期しない状況に遭遇した場合に返却されます。

- invalid_token

  アクセストークンが未知、または、その他の不当な形式である場合に返却されます。

- insufficient_authorization

  アクセス権限がない場合に返却されます。

- insufficient_scope

  アクセスに必要なスコープが認可されていない場合に返却されます。