

FUJITSU Hybrid IT Service FJcloud-O

API Management

ユーザーズガイド

～ユースケース編～

第 1.2 版

富士通株式会社

FUJITSU Hybrid IT Service FJcloud-O  
API Management ユーザーズガイド ～ユースケース編～ 第 1.2 版

発行日 2020 年 8 月

All Right Reserved, Copyright© 富士通株式会社

●本書の無断複製・転載を禁じます。



# はじめに

本書は、FUJITSU Hybrid IT Service FJcloud-O API Management（以下、本サービス）が提供する API Proxy の 作成方法を目的別に記載したユースケース集（以下、本マニュアル）です。

※本サービスの簡単な使い方については、「FUJITSU Hybrid IT Service FJcloud-O API Management ユーザーズガイド ~本編~」をご参照ください。

本サービスには、以下の4つの Policy 群が用意されています。

API Proxy に Policy を適用することで、プログラムコードを記述することなく API の振る舞いを制御することができます。

- TRAFFIC MANAGEMENT（トラフィック管理）  
API のトラフィックに関する処理（流量制限、キャッシュの設定等）をおこなう Policy です。
- SECURITY（セキュリティ）  
API のセキュリティに関する制御（認証、脆弱性対策等）をおこなう Policy です。
- MEDIATION（データ加工・変換）  
API のデータ加工（形式変換、メッセージ修正等）をおこなう Policy です。
- EXTENTION（拡張）  
スクリプトの実行やメッセージ内のデータ収集、ログ記録をおこなう Policy です。

本書では、SECURITY、MEDIATION、EXTENTION の一部の Policy を使用して以下の機能を実現する API Proxy の作成方法について説明します。

※（）内は、主に使用する Policy 群を表しています。

- 認証（SECURITY）
- API マッシュアップ（MEDIATION、EXTENTION）
- 変換（MEDIATION）

## お願い

- 本書の内容について予告無く変更をおこなうことがあります。
- 本書で使用しているバックエンドサービス（API）の仕様について、予告無く変更されることがあります。また、API を使用することで生じたいかなる損害も一切の責任を負いません。自己の責任の上で使用してください。
- 本書で使用している画面イメージ、実行例等は、最新環境のものとは異なることがありますのでご了承ください。



本チュートリアルでサンプルとして使用しております「livedoor 天気予報 API」は、2020年7月31日でサービスが終了しました。

設定例につきましては、参考手順としてご覧いただけますよう、お願い申し上げます。

# 目次

1. 認証	8
1.1. API キー認証	8
1.1.1. ユースケース概要	8
1.1.2. 手順	9
1.2. OAuth 2.0 認証	29
1.2.1. API 認証 (OAuth2.0) 概要	29
1.2.2. ユースケース概要	30
1.2.3. 手順	33
2. API マッシュアップ	76
2.1. API マッシュアップ	76
2.1.1. ユースケース概要	76
2.1.2. 手順	78
2.1.2.1. バックエンドサービスの実行結果の結合	78
2.1.2.2. バックエンドサービスの実行結果をクエリパラメーターとして付与したバックエンドサービスの呼び出し	94
2.2. Javascript でのマッシュアップ処理	104
2.2.1. ユースケース概要	104
2.2.2. 手順	106
3. 変換	126
3.1. HTTP メソッド変換	126
3.1.1. ユースケース概要	126
3.1.2. 手順	127
3.2. リクエスト Body 変換	144
3.2.1. ユースケース概要	144
3.2.2.1. XML → JSON 変換	146
3.2.2.2. JSON → JSON 変換 (キー変更)	161
3.3. レスポンス Body 変換	170
3.3.1. ユースケース概要	170
3.3.2. 手順	171
3.3.2.1. JSON → XML 変換	171
3.3.2.2. JSON → HTML 変換	182
A. 付録	190
A1. SECURITY	190
A1.1. Verify API Key XML 仕様	190
A1.2. OAuth v2.0 (GenerateAuthorizationCode) XML 仕様	190
A1.3. OAuth v2.0 (GenerateAccessToken) XML 仕様	191
A1.4. OAuth v2.0 (RefreshAccessToken) XML 仕様	193

<b>A2. MEDIATION</b> .....	195
A2.1. JSON to XML XML 仕様.....	195
A2.2. XML to JSON XML 仕様.....	196
A2.3. Assign Message XML 仕様.....	197
A2.4. Extract Variables XML 仕様.....	200
<b>A3. EXTENSION</b> .....	202
A3.1. Service Callout XML 仕様.....	202

# 1. 認証

本サービスでは、認証機能を持たない既存のバックエンドサービス（API）に対して、プログラムコードを変更することなく認証機能を付加することができます。

ここでは、SECURITY の Policy を使用した「API キー認証」「OAuth 2.0 認証」について説明します。

## 1.1. API キー認証

API キー認証では、バックエンドサービス（API）の呼び出しの際に、API キーを使用した認証を行います。

API キー認証では、API を呼び出す際の URL や HTTP リクエストのヘッダーなどに設定された API キーを検証することで、リクエストの認証を行います。

API キーが設定されていないリクエストは、API を呼び出す際に認証エラーとなり、API にアクセスできないため、不正なアクセスから保護することができます。

### 1.1.1. ユースケース概要

ここでは、バックエンドサービス（API）を呼び出す際の URL（クエリパラメーター）に指定された API キーを検証し、リクエストを認証する API Proxy を実装します。

クエリパラメーターに正しい API キーが設定されている場合のみ、API へのアクセスを許可し、その他の場合は、認証エラーのレスポンスを返却します。

バックエンドサービスとして、[天気情報を取得する API](#) を使用します。

クエリパラメーターに都市コードを指定すると、指定された都市の天気情報を取得します。

レスポンスデータ（天気情報）は JSON 形式で返却されます。

API Proxy の実装フローは以下の通りです。

- 1) API Proxy の作成
  - 1-1) API Proxy の作成
  - 1-2) Conditional Flow (GET) の作成
- 2) Product の作成
- 3) アプリ開発者 (Developer) の登録
- 4) アプリ (Developer App) の登録
- 5) API キー (Consumer Key) の確認
- 6) Verify API Key ポリシーの作成 (API キーの認証)
- 7) 動作確認

使用する Policy は以下の通りです。

- Verify API Key ポリシー

## 1.1.2. 手順

### 1) API Proxy の作成

API Proxy を作成します。

#### 1-1) API Proxy の作成

画面上部の「APIs」メニューをクリックし、API Proxies 画面に遷移します。

The screenshot shows the API Management dashboard. At the top, there is a navigation bar with 'API Management' and 'Dashboard' tabs. The 'APIs' tab is highlighted with a red box, and a red arrow points to it with the text 'ここをクリック!'. Below the navigation bar, there are filters for 'Hour', 'Day', 'Week', 'Month', and 'Custom', and a date range selector set to 'From Fri Mar 10 2017, 10:00 am To Fri Mar 17 2017, 10:00 am'. The main content area is titled 'Proxy Traffic' and shows 'No traffic in the selected date range.' Below this, there are two sections: 'Developer Engagement' and 'Developer Apps', both showing 'No traffic in the selected date range.'

API Proxies 画面で、「+ API Proxy」ボタンをクリックします。

The screenshot shows the API Proxies management page. At the top, there is a navigation bar with 'API Management' and 'Dashboard' tabs. The 'APIs' tab is highlighted with a red box, and a red arrow points to it with the text 'ここをクリック!'. Below the navigation bar, there are tabs for 'List' and 'Analytics'. The main content area is titled 'API Proxies' and shows a table of API Proxies. The table has columns for 'API Proxy', 'Environments', 'Traffic', 'Message Trend by Hour', 'Avg Time', 'Error Rate', 'Modified', and 'Actions'. The table contains three rows of data. The '+ API Proxy' button is highlighted with a red box and a red arrow pointing to it with the text 'ここをクリック!'.

API Proxy	Environments	Traffic	Message Trend by Hour	Avg Time	Error Rate	Modified	Actions
handson-api20160217020	test	1	↓	8029.00 ms	100.00 %	3 hours ago	<a href="#">Delete</a> <a href="#">Roles (1)</a>
Hello-World-Nodejs-2	test, prod	3	↓	37.00 ms	0.00 %	19 hours ago	<a href="#">Delete</a> <a href="#">Roles (1)</a>
Hello-World-Nodejs	test, prod	0				a day ago	<a href="#">Delete</a> <a href="#">Roles</a>


➤ Build a Proxy (TYPE)

「Reverse proxy (most common)」を選択し、「Next」ボタンをクリックします。

### Build a Proxy

- Reverse proxy (most common)  
Route inbound requests to backend services.  
 Use OpenAPI Optionally associate the proxy with an OpenAPI (Swagger) document
- Node.js App  
Create a new app in JavaScript and optionally add policies.
- SOAP service  
Create a RESTful or pass-through proxy for a SOAP service.
- No Target  
Create a simple API proxy that does not route to any backend target.
- Proxy bundle  
Import an existing proxy from a zip archive.

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04



➤ Build a Proxy (DETAILS)

API Proxy の情報を入力し、「Next」ボタンをクリックします。以下は入力例です。

- Proxy Name : apikey-auth (任意の名前)
- Proxy Base Path : /apikey-auth (任意のパス (自動入力))
- Existing API : http://weather.livedoor.com (任意の API)

### Build a Proxy

TYPE > **DETAILS** > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Specify the proxy details.

Proxy Name \*   
Valid characters are letters, numbers, dash (-), and underscore (\_).

Proxy Base Path \*   
A path component that uniquely identifies this API proxy. The public-facing URL of this API proxy is comprised of your organization name, an environment where this API proxy is deployed, and this Proxy Base Path. Example URL http://test-sandbox-test.apigee.net/apikey-auth

Existing API \*   
Defines the target URL invoked on behalf of this API proxy. Any URL that is accessible over the open Internet can be used. Example: https://api.usergrid.com

Description

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

**ここをクリック!**

➤ Build a Proxy (SECURITY)

Authentication : 「Pass through (none)」 を選択し、「Next」 ボタンをクリックします。

### Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Secure access for users and clients.

Authorization  Pass through (none)  
 API Key  
 OAuth 2.0

Browser  Add CORS headers

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Next



➤ Build a Proxy (VIRTUAL HOSTS)

設定を変えずに「Next」ボタンをクリックします。

Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Select the virtual hosts this proxy will bind to when it is deployed. You must select at least one virtual host. [Learn more...](#)

<input checked="" type="checkbox"/> Name	Environment	Host Aliases
<input checked="" type="checkbox"/> default	prod	http://:10080
	test	http://:10080
<input checked="" type="checkbox"/> secure	prod	https://:10443
	test	https://:10443

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Next

➤ Build a Proxy (BUILD)

設定を変えずに「Build and Deploy」ボタンをクリックします。

Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

You are ready to build and deploy your API proxy.

Deploy Environments  prod  test

Proxy Name apikey-auth

Proxy Type Reverse proxy

Virtual Hosts default, secure

Security None

Browser Do not allow direct requests from a browser via CORS

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Build and Deploy

➤ Build a Proxy (SUMMARY)

API Proxy の作成が完了したら、API Proxy のリンクをクリックします。

Build a Proxy



- ✓ Generated proxy
- ✓ Uploaded proxy
- ✓ Deployed to test

**ここをクリック！** → View `apikey-auth` proxy in the editor .

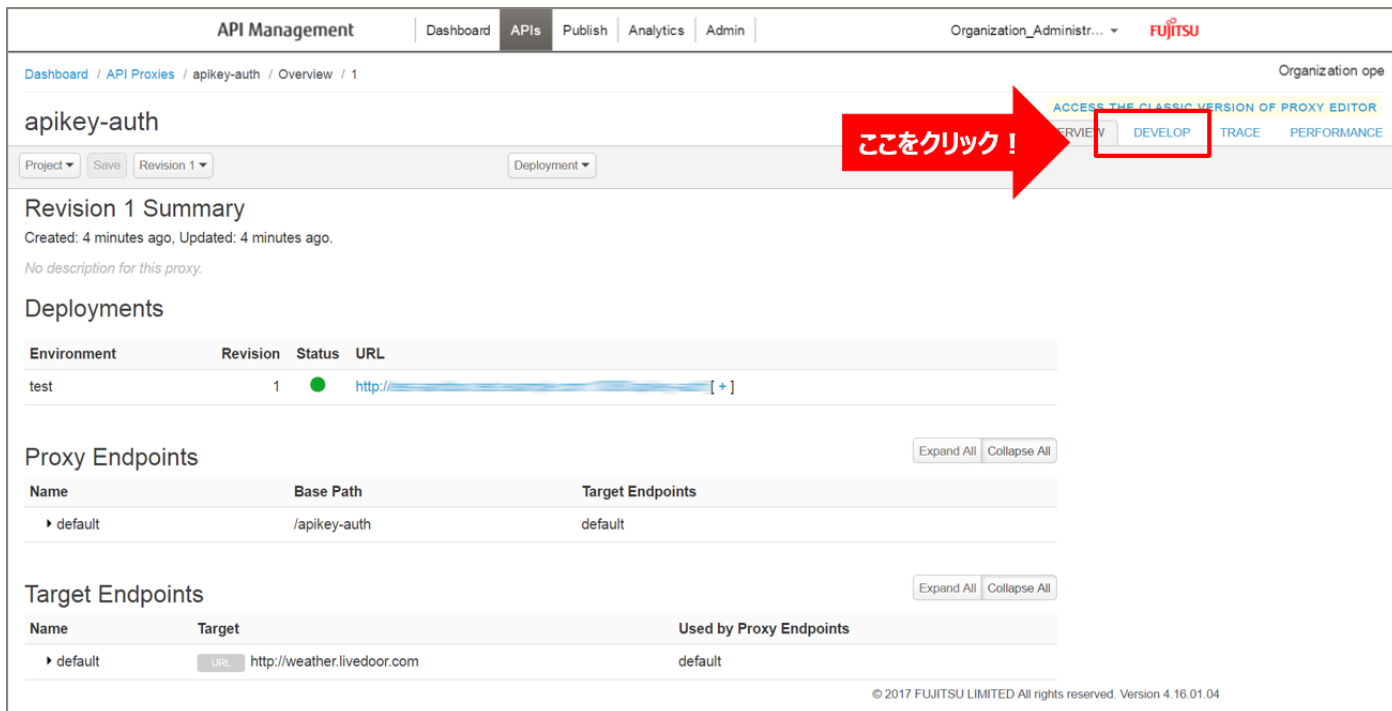
© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

## 1-2) Conditional Flow (GET) の作成

バックエンドサービスに対するリソースパスと処理 (HTTP Method) の定義を行います。

クライアントからのリクエストがここで定義したパターンと一致する場合に、API キーの認証処理(「6) Verify API Key ポリシーの作成」で設定) を実行します。

「DEVELOP」タブをクリックし、API proxy editor を開きます。



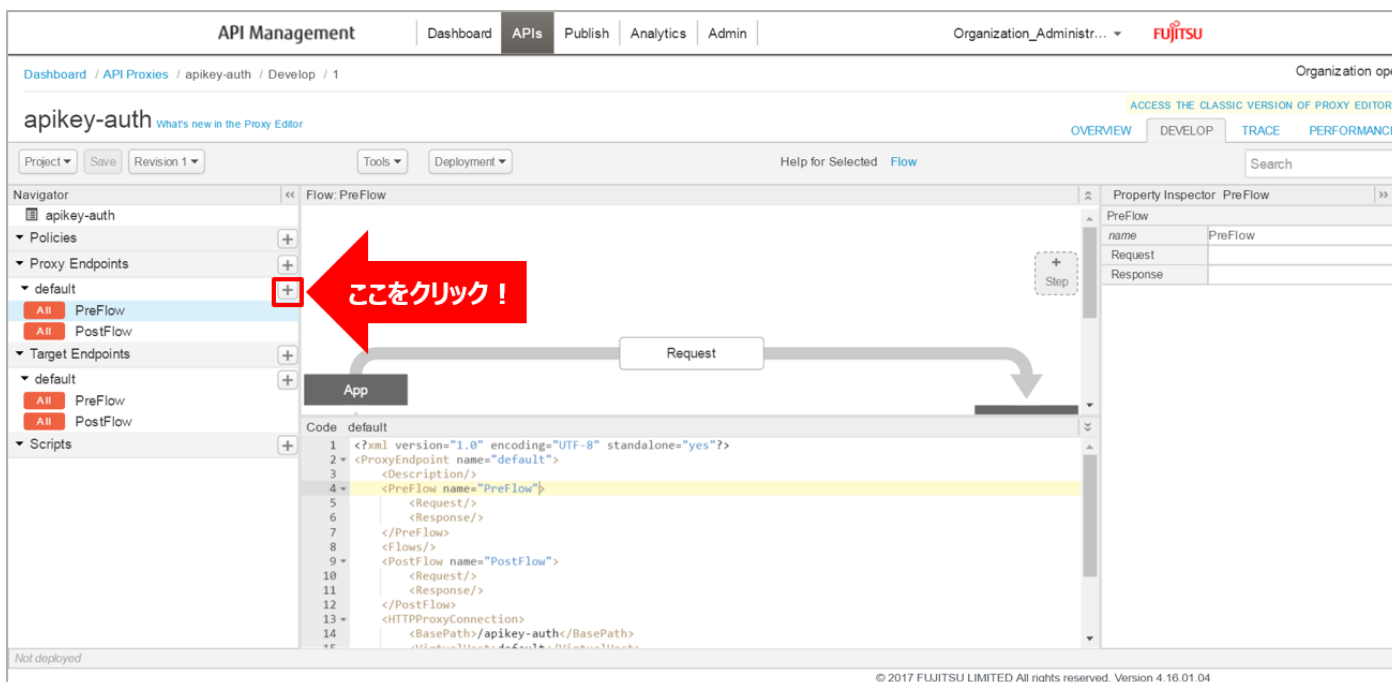
The screenshot shows the 'API Management' console for the 'apikey-auth' proxy. The 'DEVELOP' tab is highlighted in the top navigation bar, with a red arrow pointing to it and the text 'ここをクリック!' (Click here!). The main content area shows the 'Revision 1 Summary', 'Deployments' table, 'Proxy Endpoints' table, and 'Target Endpoints' table. The 'Proxy Endpoints' table has a '+' button next to the 'default' endpoint.

Environment	Revision	Status	URL
test	1	●	http://[+]

Name	Base Path	Target Endpoints
▶ default	/apikey-auth	default

Name	Target	Used by Proxy Endpoints
▶ default	URL http://weather.livedoor.com	default

default の「+」ボタンをクリックし、New Conditional Flow を開きます。

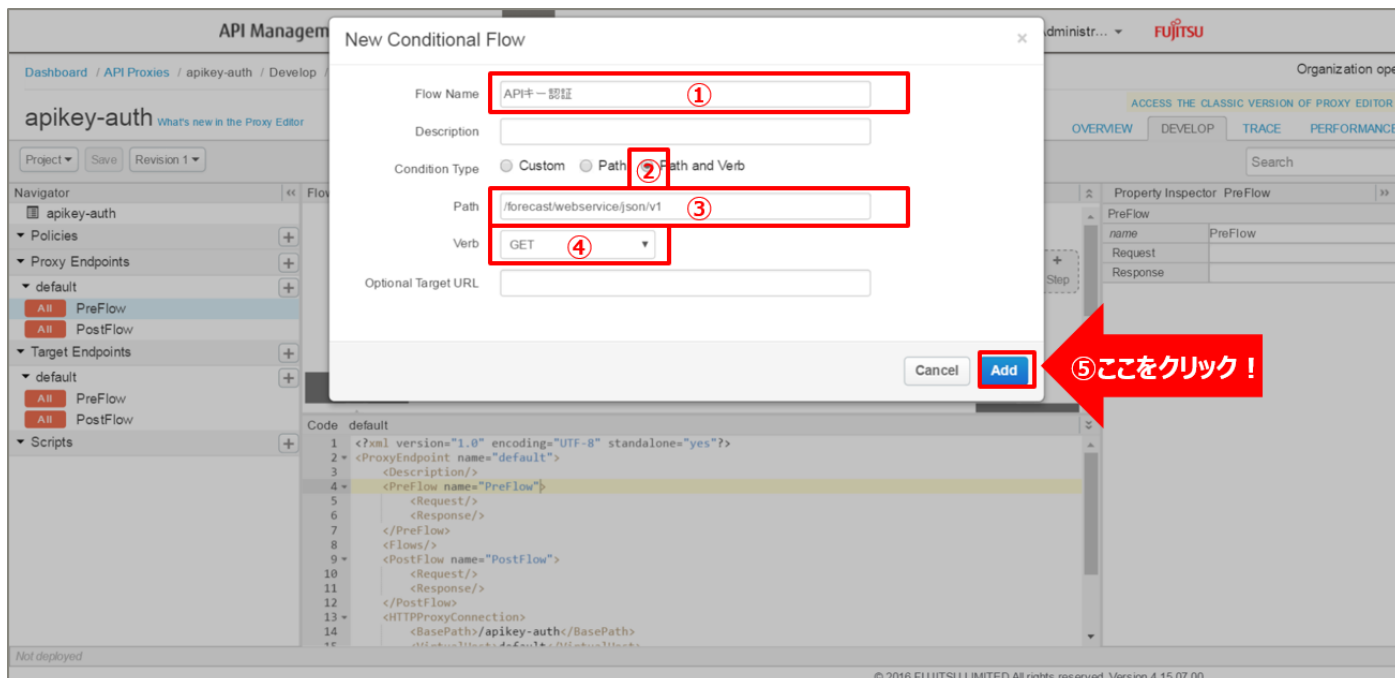


The screenshot shows the 'API Management' console for the 'apikey-auth' proxy in the 'DEVELOP' view. The left sidebar shows the 'default' proxy endpoint with a '+' button highlighted by a red arrow and the text 'ここをクリック!' (Click here!). The main content area shows the 'Flow: PreFlow' editor with a 'Request' step and a 'Code' editor containing XML configuration for the PreFlow.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ProxyEndpoint name="default">
3 <Description/>
4 <PreFlow name="PreFlow">
5 <Request/>
6 <Response/>
7 </PreFlow>
8 </Flows/>
9 <PostFlow name="PostFlow">
10 <Request/>
11 <Response/>
12 </PostFlow>
13 <HTTPProxyConnection>
14 <BasePath>/apikey-auth</BasePath>
```

Flow の情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Flow Name : API キー認証 (任意の名前)
- Condition Type : 「Path and Verb」を選択
- Path : /forecast/websevice/json/v1 (任意のパス)
- Verb : 「GET」を選択



## 2) Product の作成

Product を作成します。

Product とは、作成した API Proxy をグループ化したものです。

API キー認証等のセキュリティ設定は、作成した Product 単位で管理することができます。

画面上部の「Publish」メニューから「Products」をクリックして Products 画面に遷移し、「+ Product」ボタンをクリックします。

The image shows two screenshots of the API Management console. The top screenshot shows the 'Publish' menu with 'Products' highlighted, indicated by a red arrow and the text '①ここをクリック！'. The bottom screenshot shows the 'Products' page with a '+ Product' button highlighted, indicated by a red arrow and the text '②ここをクリック！'. The table below lists the products in the system.

Product	Keys	Created	Modified	Action
<a href="#">weather-oauth2-product</a>	1	Apr 7, 2016 7:26:59 PM	Apr 20, 2016 2:14:37 PM	History   Delete   Roles
<a href="#">ssl-yamaguchi-product</a>	1	Apr 11, 2016 4:34:04 PM	Apr 11, 2016 4:34:04 PM	History   Delete   Roles
<a href="#">handson-user001-product</a>	1	Feb 4, 2016 9:18:35 AM	Feb 4, 2016 9:18:35 AM	History   Delete   Roles
<a href="#">ssl_test</a>	1	Oct 7, 2015 10:20:50 AM	Nov 16, 2015 2:19:16 PM	History   Delete   Roles
<a href="#">Zabbix監視用プロダクト</a>	1	Oct 22, 2015 7:29:38 PM	Oct 22, 2015 7:29:38 PM	History   Delete   Roles
<a href="#">Zabbixプロダクト</a>	1	Oct 22, 2015 7:09:33 PM	Oct 22, 2015 7:26:12 PM	History   Delete   Roles
<a href="#">ssl_test2</a>	1	Oct 21, 2015 10:56:31 AM	Oct 21, 2015 10:56:31 AM	History   Delete   Roles
<a href="#">foo-bar-baz</a>	1	Oct 20, 2015 3:24:21 PM	Oct 20, 2015 3:24:21 PM	History   Delete   Roles
<a href="#">Health-Check</a>	1	Sep 17, 2015 4:34:35 PM	Sep 17, 2015 4:34:35 PM	History   Delete   Roles
<a href="#">Test Product</a>	1	Aug 7, 2015 10:17:04 AM	Aug 7, 2015 10:17:04 AM	History   Delete   Roles
<a href="#">rehel7test</a>	5	Jul 10, 2015 3:50:51 PM	Jul 10, 2015 3:50:51 PM	History   Delete   Roles
<a href="#">key_Manual</a>	5	Jul 8, 2015 11:04:45 AM	Jul 8, 2015 1:37:20 PM	History   Delete   Roles
<a href="#">sanolest-product</a>	4	Jul 6, 2015 5:10:15 PM	Jul 7, 2015 3:15:26 PM	History   Delete   Roles
<a href="#">ZabbixGroup</a>	4	Jul 1, 2015 2:35:55 PM	Jul 6, 2015 4:59:05 PM	History   Delete   Roles

Product の情報を入力し、「Save」ボタンをクリックします。以下は入力例です。

◇ Product Details

- Name : apikey-auth-product (任意の名前)
- Display Name : apikey-auth-product (任意の名前 (自動入力))
- Environment : 「test」, 「prod」 を選択
- Access : 「Public」 を選択
- Key Approval Type : 「Automatic」 を選択

◇ Resources

- API Proxies : apikey-auth (「1) API Proxy の作成」で作成した API Proxy を選択)

The screenshot shows the 'New Product' configuration page in the API Management console. The 'Product Details' section is highlighted with a red box and contains the following information:

- Name: apikey-auth-product
- Display Name: apikey-auth-product
- Description: (empty)
- Environment:  test  test2  prod
- Access:  Public (Visible only to any registered developer during app registration)
- Key Approval Type:  Automatic
- Quota: (empty) requests every (empty) Select...
- Allowed OAuth Scopes: (empty)

The 'Resources' section shows a table with columns for API Proxy, Revision, and Resource Path. The 'API Proxies' column has a dropdown menu with 'apikey-auth' selected, marked with a red circle and the number 3. A red arrow points to the '+ API Proxy' button with the text '②ここをクリック！'. At the bottom right, a red arrow points to the 'Save' button with the text '④ここをクリック！'. A text box in the Resources section states: '「apikey-auth」を選択 ※②,③の操作を繰り返すことで複数のAPIProxyを登録できます'. The footer of the page reads '© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04'.

### 3) アプリ開発者 (Developer) の登録

Developer を登録します。

登録した Developer は、次の手順でアプリ (Developer App) に割り当てます。

画面上部の「Publish」メニューから「Developers」をクリックして Developers 画面に遷移し、「+ Developer」ボタンをクリックします。

The image shows two screenshots of the API Management console. The top screenshot shows the 'Publish' menu with 'Developers' highlighted, and a red arrow pointing to it with the text '①ここをクリック！'. The bottom screenshot shows the 'Developers' page with a red arrow pointing to the '+ Developer' button and the text '②ここをクリック！'. A table of developers is visible in the bottom screenshot.

Developer	Email	Username	App	Keys	Actions
dev_admin dev_admin	dev_admin@jp.fujitsu.com	dev_admin	4	Pending	<input type="button" value="Delete"/>
ops user	XXXXXXXX@jp.fujitsu.com	opsuser	2	2	<input type="button" value="Delete"/>
ssl test	test@example.com	ssl_test	2	2	<input type="button" value="Delete"/>
ssl test2	test2@example.com	ssl_test2	1	1	<input type="button" value="Delete"/>
test 1	test1@test.test	test1	6	Pending	<input type="button" value="Delete"/>
test sano	s.akhiko@jp.fujitsu.com	sano	1	1	<input type="button" value="Delete"/>
test test	test@example.co.jp	test_user	1	1	<input type="button" value="Delete"/>
test test	test@test.test	test	2	2	<input type="button" value="Delete"/>
yuuki taki	test@test.test	taki2	2	2	<input type="button" value="Delete"/>

Developer の情報を入力し、「Save」ボタンをクリックします。

API Management | Dashboard | APIs | Publish | Analytics | Admin | Organization\_Administr... | FUJITSU

Dashboard / Developers / New Developer | Organization open Environment prod

### New Developer

**Developer Details**

First Name 太郎

Last Name 富士通

Email fj-taro@example.com

Username fj-taro

• 適当なFist Name,Last Name,Email,Username  
を入力します。

**Custom Attributes**

Name	Value	Actions
		+ Custom Attribute

ここをクリック

Save

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04



#### 4) アプリ (Developer App) の登録

Developer を割り当てる Developer App を作成します。

Developer App に Developer を割り当てることで、固有の APIKey が生成されます。リクエストの際にこの API Key を利用して API Product にアクセスさせることで、Developer の特定とセキュリティの確保を行います。

1 つの Developer App に対しては、1 人の Developer と 1 つ以上の Product を割り当てる必要があります。

画面上部の「Publish」メニューから「Developer Apps」をクリックして Developer Apps 画面に遷移し、「+ Developer App」ボタンをクリックします。

The image shows two screenshots of the API Management console. The top screenshot shows the 'Publish' menu with 'Developer Apps' highlighted and a red arrow pointing to it with the text '①ここをクリック！'. The bottom screenshot shows the 'Developer Apps' page with a red arrow pointing to the '+ Developer App' button and the text '②ここをクリック！'. A table of existing Developer Apps is also visible in the bottom screenshot.

App	Developer	App Family	Company	Key	Metrics for Last 24 Hours		Registered	Actions
					Traffic	Error Rate (%)		
ssl-yamaguchi-app	ssl test	default		1			Apr 11, 2016 4:36:51 PM	<a href="#">x Delete</a>
weather-oauth2-app	dev_admin dev_admin	default		1			Apr 7, 2016 7:28:14 PM	<a href="#">x Delete</a>
handson-user001-app	dev_admin dev_admin	default		1			Feb 4, 2016 9:26:00 AM	<a href="#">x Delete</a>
ZabbixApp	ops user	default		1			Oct 22, 2015 7:32:37 PM	<a href="#">x Delete</a>
Zabbix	ops user	default		1			Oct 22, 2015 7:25:21 PM	<a href="#">x Delete</a>
ssl_test_app2	ssl test2	default		1			Oct 21, 2015 10:57:13 AM	<a href="#">x Delete</a>
AAAAA	dev_admin dev_admin	default		Pending			Oct 20, 2015 3:25:26 PM	<a href="#">x Delete</a>
ssl_test_app	ssl test	default		1			Oct 7, 2015 11:09:03 AM	<a href="#">x Delete</a>
Health-Check	test test	default		1			Sep 17, 2015 4:36:28 PM	<a href="#">x Delete</a>
Test App	test test	default		1			Aug 7, 2015 10:17:37 AM	<a href="#">x Delete</a>
test app3	dev_admin dev_admin	default		1			Jul 28, 2015 2:39:13 PM	<a href="#">x Delete</a>
test app2	test 1	default		Pending Revoked			Jul 28, 2015 2:38:43 PM	<a href="#">x Delete</a>
test app1	test 1	default		1			Jul 28, 2015 2:38:14 PM	<a href="#">x Delete</a>

Developer App の情報を入力し、「Save」ボタンをクリックします。

API Management | Dashboard | APIs | Publish | Analytics | Admin | Organization\_Administr... | FUJITSU

Dashboard / Developer Apps / New Developer App | Organization ope

### New Developer App

**Developer App Details**

Name: apikey-auth-app  
Display Name: apikey-auth-app ①  
Developer: 太郎 富士通 (tj-taro@example...)  
Callback URL:   
A callback URL is only required for 3-legged OAuth

Notes:

**Products**

Product	Status	Consumer Key	Consumer Secret	Actions
apikey-auth-product ②				✓ -

At least one product is required.

**Custom Attributes**

Name	Value	Actions
------	-------	---------

+ Custom Attribute

Cancel Save ④

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

- Name : 「apikey-auth-app」(任意の名前)
- Display Name : 自動入力されます。
- Developer: 登録したDeveloperを選択

②ここをクリック!

④ここをクリック!

## 5) API キー (Consumer Key) の確認

Developer Apps 画面のアプリ一覧に、登録したアプリが追加されます。  
追加されたアプリ名を選択し、登録情報を確認します。

API Management | Dashboard | APIs | Publish | Analytics | Admin | Organization\_Administr... | FUJITSU

Dashboard / Developer Apps | Organization ope | Environment: prod

### Developer Apps

List | Analytics

Search: All | All | Pending | Revoked | Approved | 1—22 of 22 | + Developer App

App	Developer	Company	Key	Metrics for Last 24 Hours		Registered	Actions
				Traffic	Error Rate (%)		
apikey-auth-app	太郎 富士通	default	1			May 24, 2016 4:59:05 PM	✕ Delete
ssl-yamaguchi-app	ssl test	default	1			Apr 11, 2016 4:36:51 PM	✕ Delete
weather-oauth2-app	dev_admin dev_admin	default	1			Apr 7, 2016 7:28:14 PM	✕ Delete
handson-user001-app	dev_admin dev_admin	default	1			Feb 4, 2016 9:26:00 AM	✕ Delete
ZabbixApp	ops user	default	1			Oct 22, 2015 7:32:37 PM	✕ Delete
Zabbix	ops user	default	1			Oct 22, 2015 7:25:21 PM	✕ Delete
ssl_test_app2	ssl test2	default	1			Oct 21, 2015 10:57:13 AM	✕ Delete
AAAAA	dev_admin dev_admin	default	Pending			Oct 20, 2015 3:25:26 PM	✕ Delete
ssl_test_app	ssl test	default	1			Oct 7, 2015 11:09:03 AM	✕ Delete
Health-Check	test test	default	1			Sep 17, 2015 4:36:28 PM	✕ Delete
Test App	test test	default	1			Aug 7, 2015 10:17:37 AM	✕ Delete
test app3	dev_admin dev_admin	default	1			Jul 28, 2015 2:39:13 PM	✕ Delete
test app2	test 1	default	Pending Revoked			Jul 28, 2015 2:38:43 PM	✕ Delete

ここをクリック!

Consumer Key 欄の「Show」ボタンをクリックすると、API キーが表示されます。

The screenshot shows the 'API Management' interface for a developer app named 'apikey-auth-app'. The 'Products' table has a 'Consumer Key' column with a 'Show' button. A red arrow points to this button with the text 'ここをクリック!'.

Product	Status	Consumer Key	Consumer Secret
apikey-auth-product	Approved		

Below the screenshot, a large grey arrow points to the second screenshot, which shows the 'Consumer Key' column with the key value 'pX8PKIevGJZkSWQIRFH8JvLdTGnm4TAd' displayed. The 'Show' button is now 'Hide'.

Product	Status	Consumer Key	Consumer Secret
apikey-auth-product	Approved	pX8PKIevGJZkSWQIRFH8JvLdTGnm4TAd	

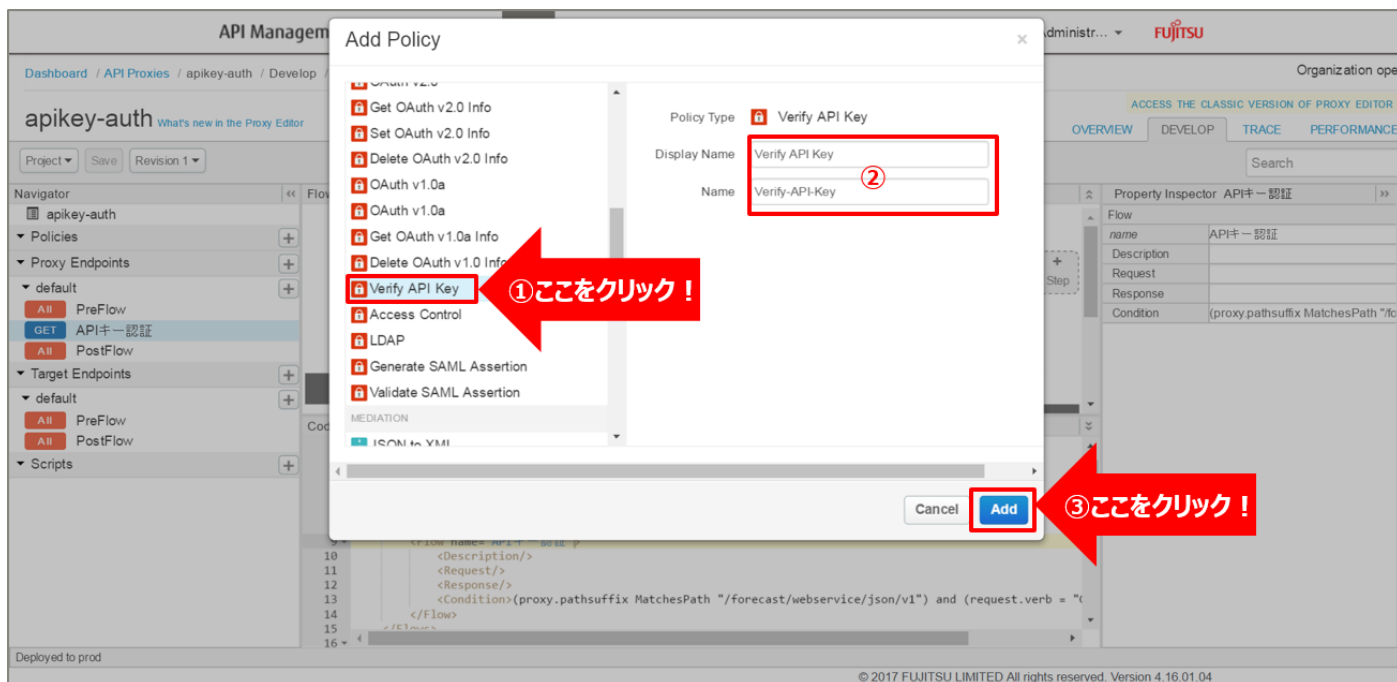
## 6) Verify API Key ポリシーの作成

API キーの検証・認証処理を行う Verify API Key ポリシーを追加します。

事前に登録されたアプリからのリクエストのみ、API へのアクセスを許可するようにします。

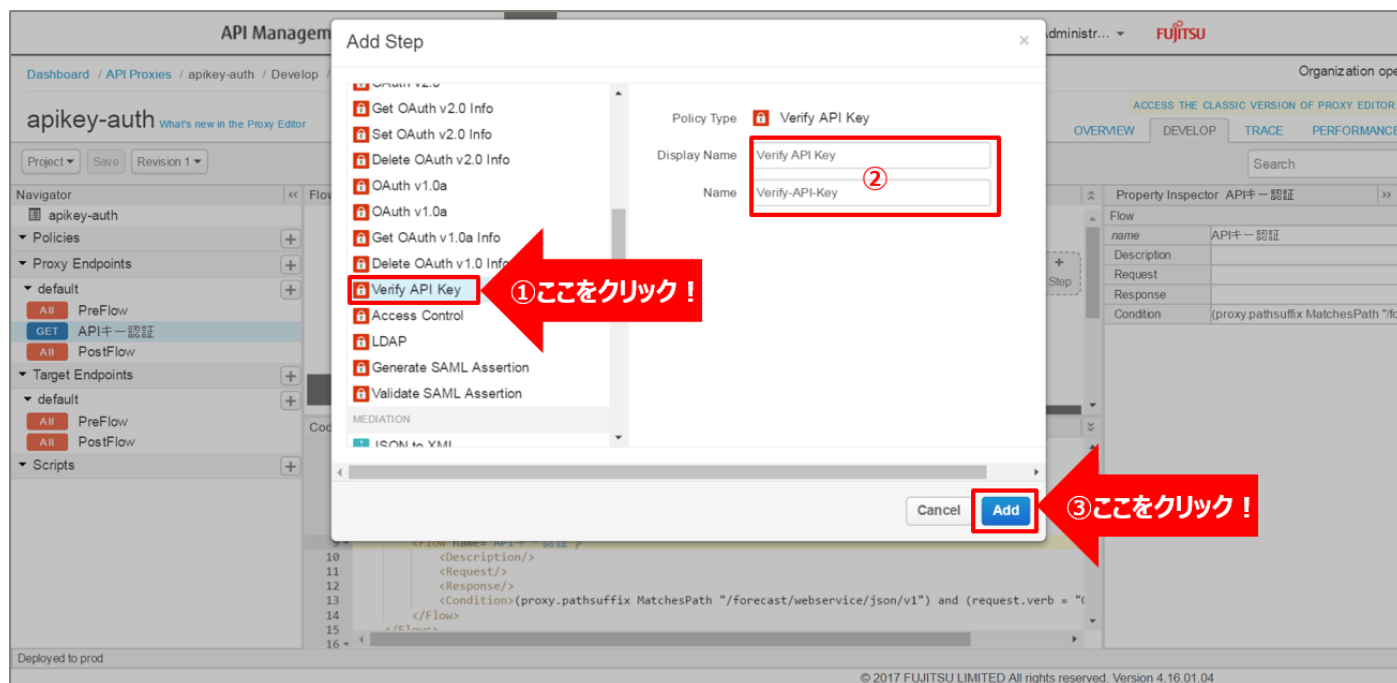
API Proxies 画面に遷移し、「1) API Proxy の作成」で作成した API Proxy を選択した後、「DEVELOP」タブをクリックして API proxy editor を開きます。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Verify API Key」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : Verify API Key (任意の名前)
- Name : Verify-API-Key (任意の名前)



追加した Verify API Key ポリシーの編集画面を表示されます。  
必要に応じてポリシーの定義を編集します。

The screenshot shows the 'Verify API Key' policy configuration in the API Management console. The XML code is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VerifyAPIKey async="false" continueOnError="false" enabled="true" name="Verify-API-Key">
  <DisplayName>Verify API Key</DisplayName>
  <Properties/>
  <APIKey ref="request.queryparam.apikey"/>
</VerifyAPIKey>
```

The line `<APIKey ref="request.queryparam.apikey"/>` is highlighted in red, and a callout box points to it with the text "APIキーの取得先を指定".

#### 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VerifyAPIKey async="false" continueOnError="false" enabled="true" name="Verify-API-Key">
  <DisplayName>Verify API Key</DisplayName>
  <Properties/>
  <APIKey ref="request.queryparam.apikey"/>
</VerifyAPIKey>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A1.1. Verify API Key XML 仕様](#)」をご参照ください。

## 7) ポリシーの配置

以下の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。

The screenshot displays the API Management interface for configuring a policy. The left sidebar shows a tree view with 'Policies' expanded, containing 'Verify API Key'. The main area shows a flow diagram with 'App' and 'Server' components. A 'Verify A' policy icon is being dragged into the flow. The top right shows the 'Property Inspector' for the selected policy.

③ここをクリック！

②ここにドラッグ！

①ここをクリック！

Property Inspector APIキー認証	
Flow	
name	APIキー認証
Description	
Request	
Step	
Name	Verify-API-Key
Response	
Condition	(proxy pathsuffix MatchesPath */fc

## 8) 動作確認

作成した API Proxy の動作確認を行います。

### 【API キーを指定した場合】

以下の例のように、URL へアクセスすると JSON 形式のデータが返却されます。

http://{FQDN}:10080/apikey-

auth/forecast/webservice/json/v1?city={CityCode}&apikey={ConsumerKey}

※http プロトコルでの動作確認例です（ポート番号は 10080 です）。

※{FQDN}: API Proxy の作成時に生成された URL のホスト名 (FQDN) を指定します。

※{CityCode} は、任意の都市コードに置き換えてください。(例: 400040 など)

※{ConsumerKey} は、「[5\) API キー \(Consumer Key\) の確認](#)」で確認した Consumer Key に置き換えてください。

```
{
  - pinpointLocations: [
    - [
      link: "http://weather.livedoor.com/area/forecast/4020200",
      name: "大牟田市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4020300",
      name: "久留米市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4020700",
      name: "柳川市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4021000",
      name: "八女市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4021100",
      name: "筑後市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4021200",
      name: "大川市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4021600",
      name: "小郡市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4022500",
      name: "うきは市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4022800",
      name: "朝倉市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4022900",
      name: "みやま市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4044700",
      name: "...
```

### 【API キーを指定しない場合】

以下の例のように、API キーを指定しない場合は認証エラーとなりデータを取得できないため、不正なアクセスから保護することができます。

http://{FQDN}:10080/apikey-auth/forecast/webservice/json/v1?city={CityCode}

※http プロトコルでの動作確認例です（ポート番号は 10080 です）。

※{FQDN}: API Proxy の作成時に生成された URL のホスト名（FQDN）を指定します。

※{CityCode} は、任意の都市コードに置き換えてください。（例：400040 など）

```
[
  - fault: [
    - faultstring: "Failed to resolve API Key variable request.queryparam.apikey",
    - detail: [
      - errorcode: "steps.oauth.v2.FailedToResolveAPIKey"
    ]
  ]
]
```



## 1.2. OAuth 2.0 認証

### 1.2.1. API 認証 (OAuth2.0) 概要

OAuth 2.0 は、ネットワーク上のリソースに対して安全なアクセスを実現させるためのプロトコル仕様です。このプロトコル仕様は、[RFC 6749, “The OAuth 2.0 Authorization Framework”](#) として定義されています。

本サービスでは、ポリシーを使用することで、OAuth 2.0 を使用した API の認可を実装することができます。

本サービスは、OAuth Service Provider としての機能を有しており、トークンの払い出しや有効期限の管理など、運用面を意識することなく OAuth 2.0 の仕組みを利用することが可能です。

また、OAuth 2.0 が定義する以下の 4 つのアクセス権限付与フロータイプ (Grant 種別) 全てに対応しています。

種別	利用シーン
Authorization Code Grant	(保護情報への) 長期にわたるアクセスが必要な、サーバサイド Web アプリ (Web サーバ)
Implicit Grant	一時的なアクセスが必要な、Web ブラウザ上で動作するクライアントサイド Web アプリ
Resource Owner Password Grant	リソースオーナーから信頼されクレデンシャルを保持する、Web ブラウザ以外で動作するクライアント
Client Credentials Grant	自分自身がリソースオーナーである、または別の手段でリソースオーナーから事前にアクセス認可されている、Web ブラウザ以外で動作するクライアント

本ユースケースでは、4 つの Grant 種別のうち、Authorization Code Grant を利用した API の認可について説明します。

### 1.2.2. ユースケース概要

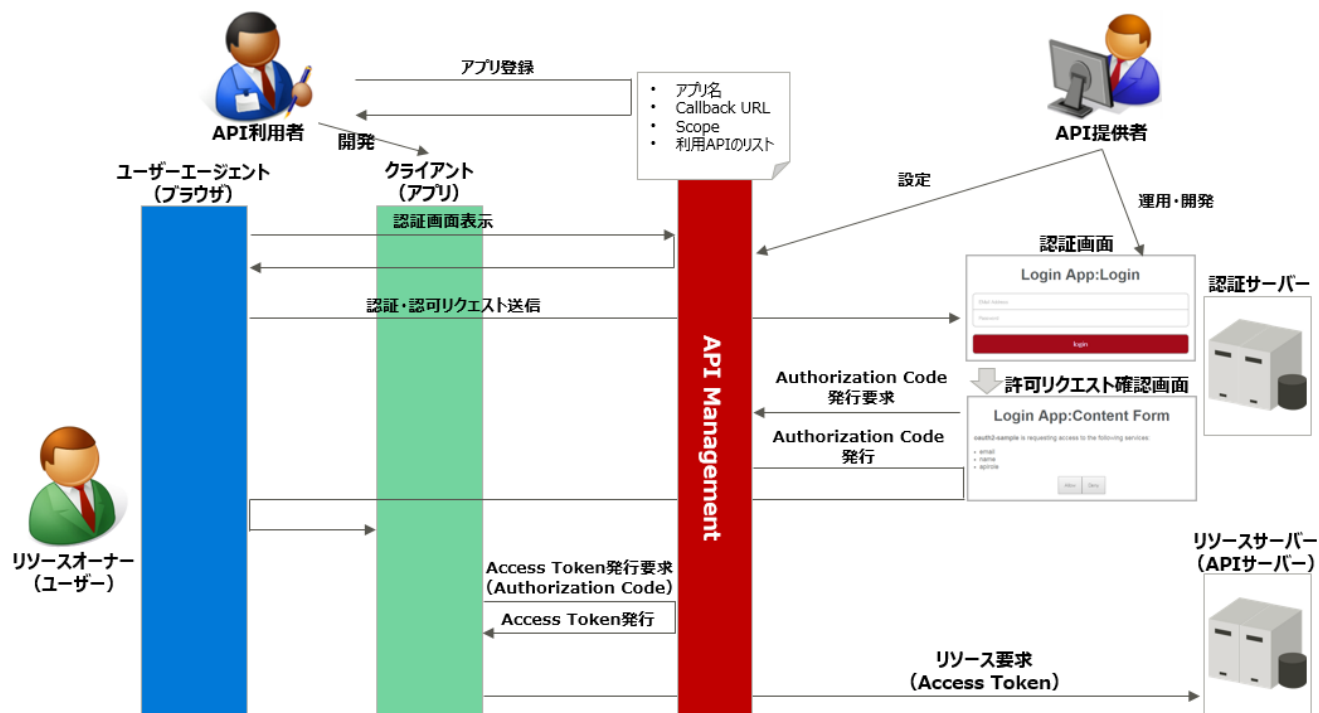
このユースケースでは、本サービスを経由した OAuth 2.0 Authorization Code Grant の仕組み（認証・認可フロー）について説明します。

主なロールは以下の通りです。

ロール	概要
API 提供者	API の開発者です。
API 利用者	クライアント（アプリ）の開発者です。
リソースオーナー	ユーザーデータ（リソース）の所有者であり、クライアント（アプリ）にアクセス認可を与える存在です。 クライアント（アプリ）を使用するエンドユーザーです。
クライアント	リソースアクセスの認可を受けるサードパーティアプリです。
認証サーバ	認証処理を行うサーバです。
リソースサーバ	リソースを保持するサーバです。
ユーザーエージェント	リソースオーナーの Web ブラウザです。
FUJITSU Hybrid IT Service FJcloud-O API Management	サービスプロバイダーとして機能します。認可処理を行い、Authorization Code（認可コード）や Access Token（アクセストークン）を発行します。

本サービスを経由した OAuth 2.0 Authorization Code Grant フローは以下の通りです。

※正常系の概略フローです。



フロー番号	対象者	フロー概要
①	API 提供者	認証サーバ (ログインアプリ)、リソースサーバを用意します。
②	API 提供者	本サービス上に OAuth 2.0 ポリシーを適用した API Proxy を作成します。(※1.2.3. 手順 2)~3) に該当)
③	API 提供者	API 利用者からの申請 (アプリ名, Callback URL, Scope, 利用 API ...etc.) により、本サービスにアプリ情報を登録します。 また、登録の際に払い出された client_id, client_secret を、API 利用者に通知します。
	API 利用者	通知された client_id, client_secret を組み込んだアプリ (クライアントアプリ) を作成します。
④	リソースオーナー	ユーザーエージェントからクライアントを経由して本サービスにアクセスします。この際、リクエスト情報に client_id, state を付与します。 state は、クライアント側でユニークキーを生成し、セッションに格納しておきます。
	FUJITSU Hybrid IT Service FJcloud-O API Management	client_id が登録済みか確認します。
⑤	FUJITSU Hybrid IT Service FJcloud-O API Management	登録済みであれば、リダイレクトにより認証サーバの認証画面をユーザーエージェントにロードします。
⑥	リソースオーナー	認証画面に認証情報を入力し、認証サーバに送信します。 ※クライアントや本サービスを經由せず、直接認証サーバへアクセスして認証を受けます。
	認証サーバ	受け取った認証情報をもとに認証します。 認証後、許可リクエスト確認画面をユーザーエージェントにロードし、リソースオーナーにクライアントによるリソースへのアクセス要求の許可を問い合わせます。
	リソースオーナー	問い合わせに対し、アクセス要求を許可します。
⑦	認証サーバ	アクセス要求が許可された場合、本サービスに Authorization Code の発行を要求します。
⑧	FUJITSU Hybrid IT Service FJcloud-O API Management	Authorization Code の発行後、リダイレクト URI を認証サーバに送信します。 ※Authorization Code、state はリダイレクト URI 内に含まれます。
	認証サーバ	受け取ったリダイレクト URI を用いて、ユーザーエージェントを指定された場所 (クライアント) にリダイレクトします。
⑨	クライアント	セッション上のユニークキーと state パラメーターの値が同じか確認します。同じであれば、リクエストに client_id, client_secret, Authorization Code を付与し、本サービスに Access Token を要求します。

		※ユーザーエージェントを経由せず、直接本サービスと通信します。
⑩	FUJITSU Hybrid IT Service FJcloud-O API Management	client_id, client_secret でクライアントを認証し、Authorization Code を検証します。問題がなければ、クライアントに Access Token を発行します。
⑪	クライアント	Access Token を使ってリソースサーバにリソースへのアクセスを要求します。

### 1.2.3. 手順

#### 1) 事前準備

API 提供者は、認証サーバとリソースサーバを用意します。

また、認証サーバから本サービスにアクセスできるように接続設定を行っておきます。

#### 2) Access Token 発行 API Proxy の作成

API 提供者が実施します。

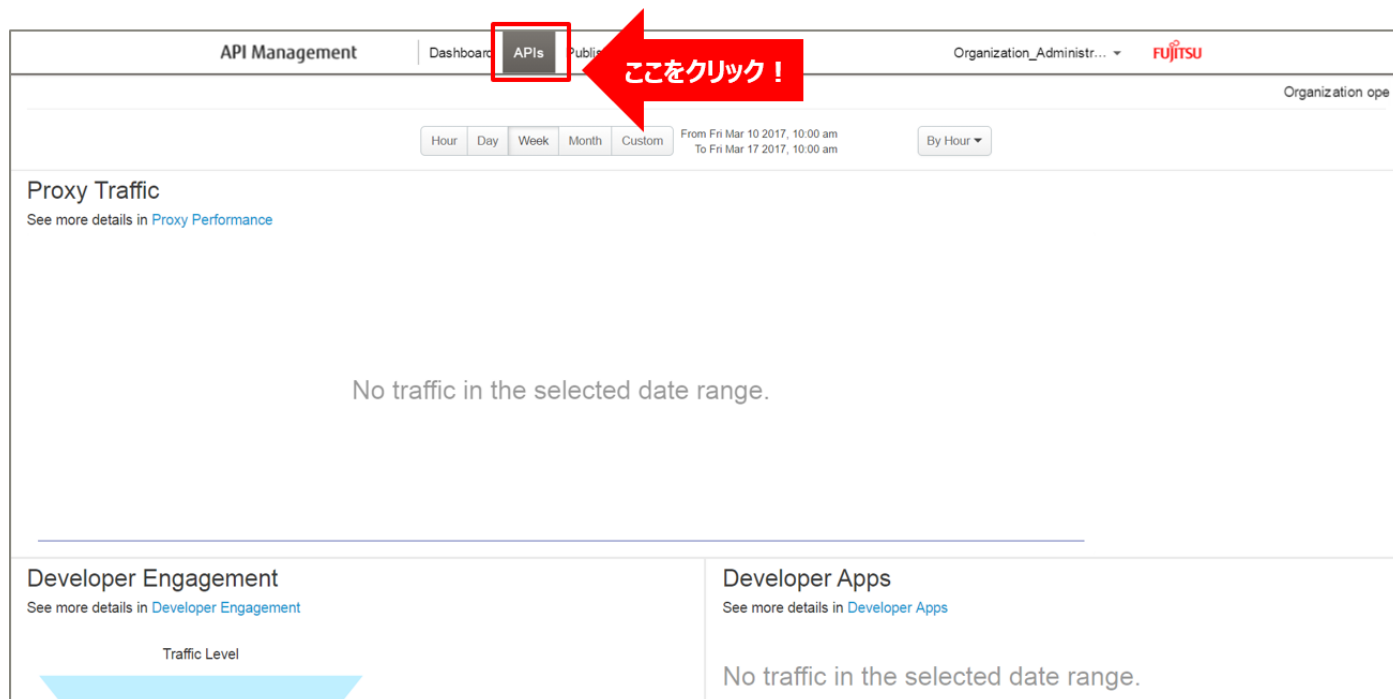
Access Token を発行するために必要な API を API Proxy として作成します。

本サービスを経由した OAuth 2.0 Authorization Code Grant フローの ④ から ⑩ の処理を実装します。

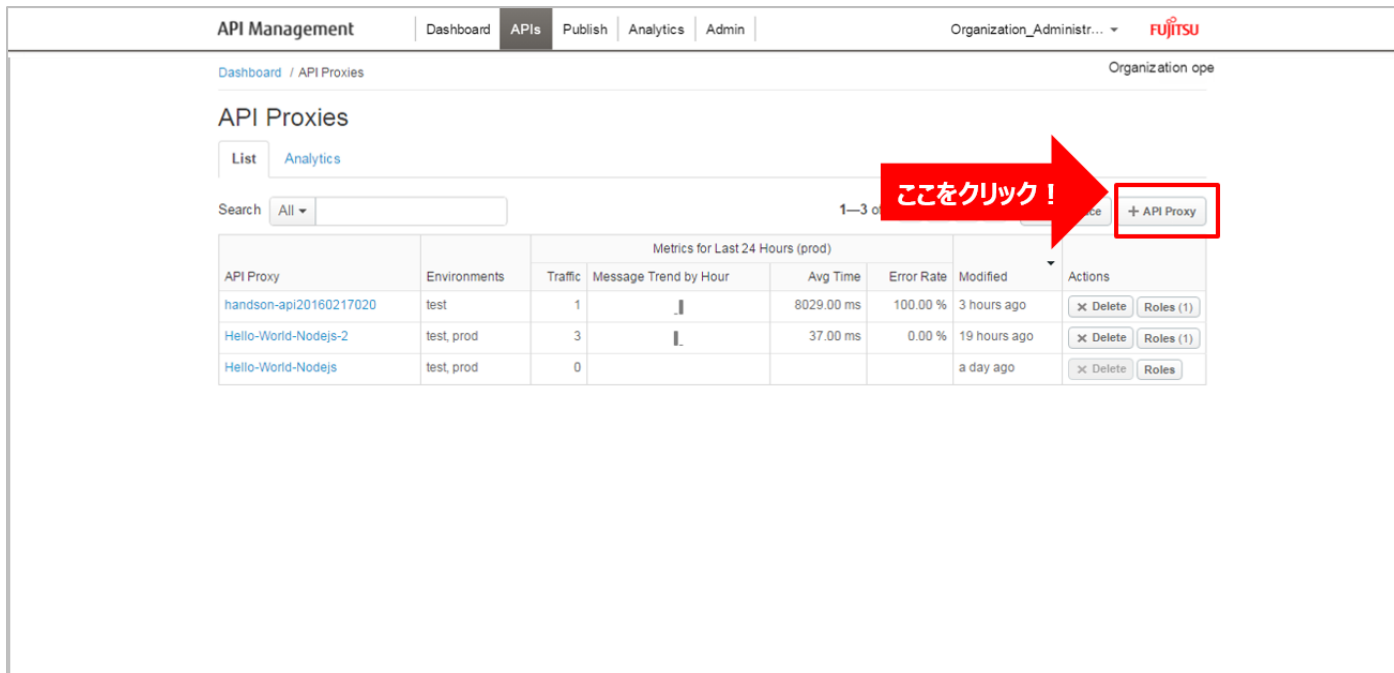
#### 2-1) API Proxy の作成

API Proxy を作成します。

画面上部の「APIs」メニューをクリックし、API Proxies 画面に遷移します。

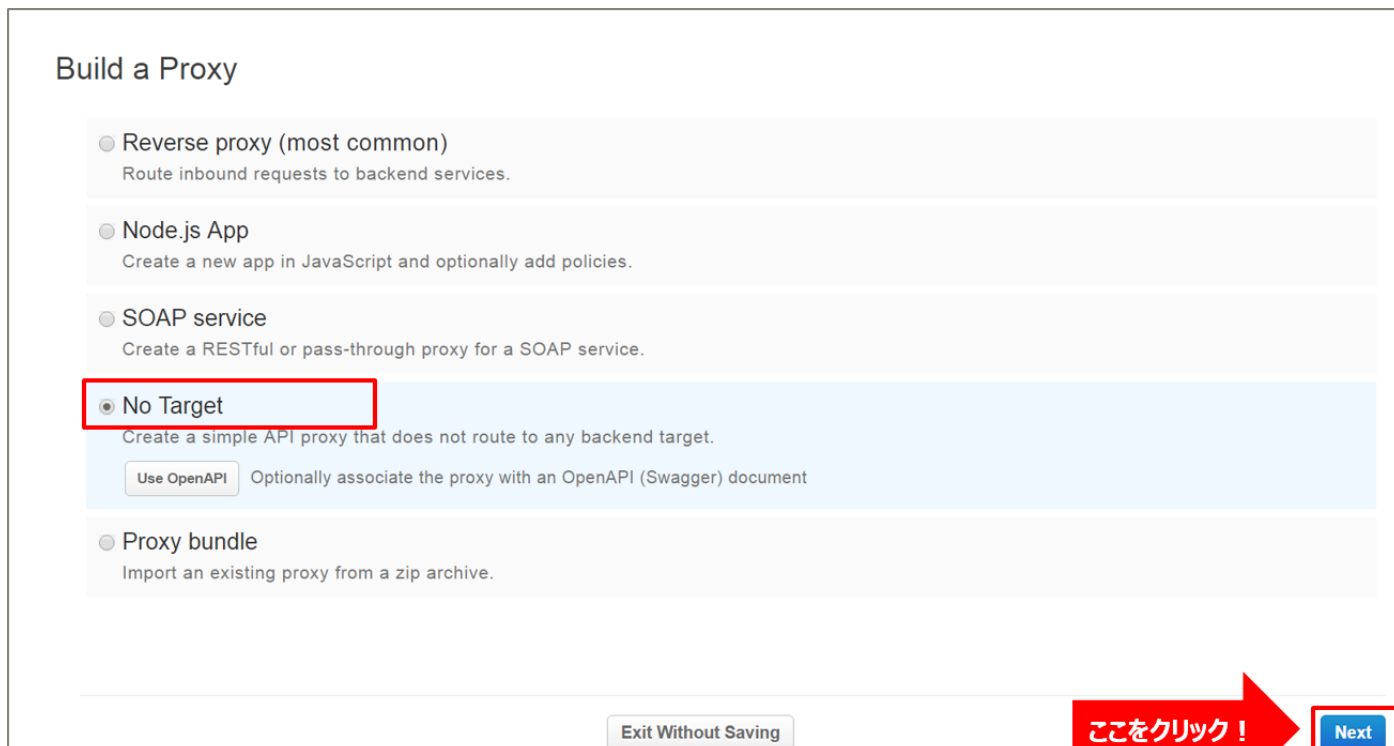


API Proxies 画面で、「+ API Proxy」ボタンをクリックします。



➤ Build a Proxy (TYPE)

「No Target」を選択し、「Next」ボタンをクリックします。



➤ Build a Proxy (DETAILS)

API Proxy の情報を入力し、「Next」ボタンをクリックします。以下は入力例です。

- Proxy Name : oauth (任意の名前)
- Proxy Base Path : /oauth (任意のパス (自動入力))

### Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Specify the proxy details.

Proxy Name\*   
Valid characters are letters, numbers, dash (-), and underscore (\_).

Proxy Base Path\*   
A path component that uniquely identifies this API proxy. The public-facing URL of this API proxy is comprised of your organization name, an environment where this API proxy is deployed, and this Proxy Base Path. Example URL <http://test-sandbox-test.apigee.net/oauth>

Description

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

[Previous](#) [Exit Without Saving](#) [Next](#)

➤ Build a Proxy (SECURITY)

Authentication : 「Pass through (none)」 を選択し、「Next」 ボタンをクリックします。

### Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Secure access for users and clients.

Authorization

- Pass through (none)
- API Key
- OAuth 2.0

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving Next



➤ Build a Proxy (VIRTUAL HOSTS)

設定を変えずに「Next」ボタンをクリックします。

Build a Proxy

TYPE > DETAILS > SECURITY > **VIRTUAL HOSTS** > BUILD > SUMMARY

Select the virtual hosts this proxy will bind to when it is deployed. You must select at least one virtual host. [Learn more...](#)

<input checked="" type="checkbox"/> Name	Environment	Host Aliases
<input checked="" type="checkbox"/> default	prod	http://[redacted]:10080
	test	http://[redacted]:10080
<input checked="" type="checkbox"/> secure	prod	https://[redacted]:10443
	test	https://[redacted]:10443

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous      Exit Without Saving      **ここをクリック!**      **Next**

➤ Build a Proxy (BUILD)

設定を変えずに「Build and Deploy」ボタンをクリックします。

Build a Proxy

TYPE DETAILS SECURITY VIRTUAL HOSTS BUILD SUMMARY

You are ready to build and deploy your API proxy.

Deploy Environments  prod  test

Proxy Name oauth

Proxy Type No Target

Virtual Hosts secure, default

Security None

Browser Do not allow direct requests from a browser via CORS

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Build and Deploy

➤ Build a Proxy (SUMMARY)

API Proxy の作成が完了したら、API Proxy のリンクをクリックします。

Build a Proxy

TYPE DETAILS SECURITY VIRTUAL HOSTS BUILD SUMMARY

✓ Generated proxy

✓ Uploaded proxy

✓ Deployed to test

**ここをクリック!** View **oauth** proxy in the editor .

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

## 2-2) authorize API の作成 (④, ⑤)

フローの ④, ⑤ の処理を実装します。

### (1) Conditional Flow (GET) の作成 (authorize エンドポイントの作成)

バックエンドサービスに対するリソースパスと処理 (HTTP Method) の定義を行います。

クライアントからのリクエストがここで定義したパターンと一致する場合に、以降の手順で設定する処理を実行します。

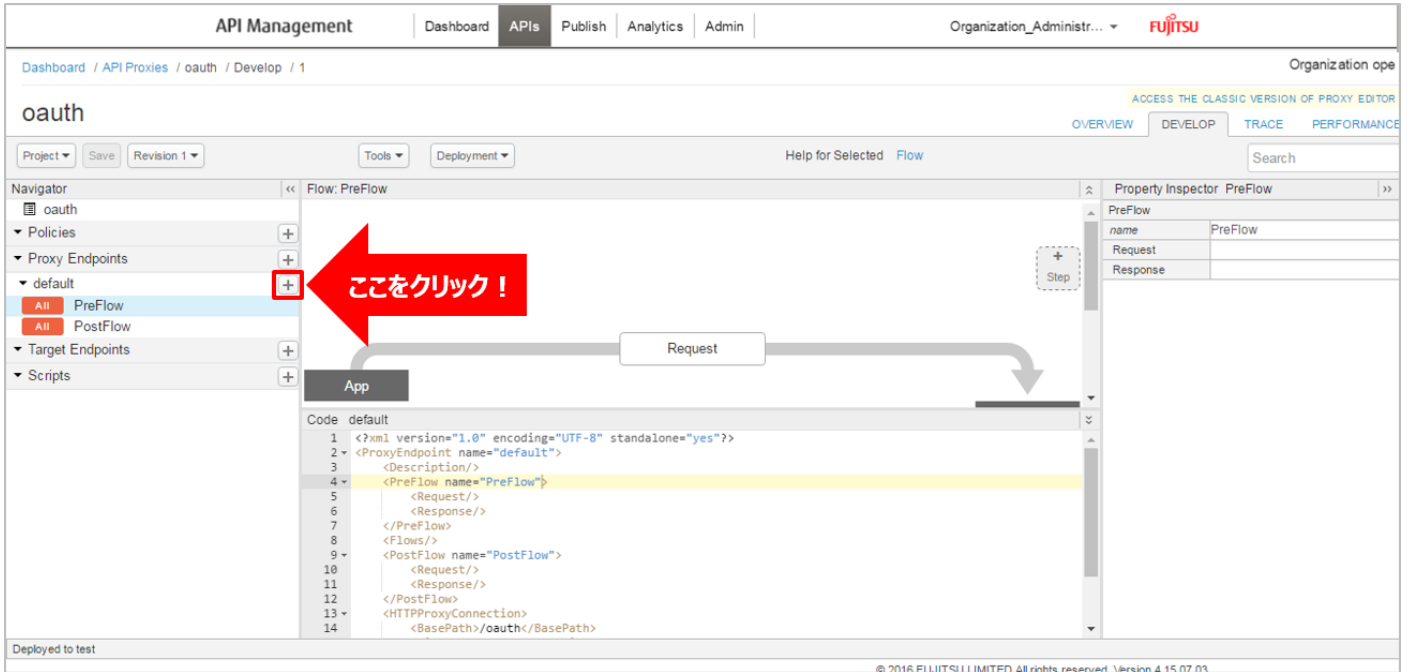
「Develop」タブをクリックし、API proxy editor を開きます。

The screenshot shows the API Management console for the 'oauth' proxy. A red arrow points to the 'DEVELOP' tab, which is highlighted with a red box. The page displays the following information:

- Revision 1 Summary:** Created: 5 minutes ago, Updated: 5 minutes ago. No description for this proxy.
- Deployments:** A table with columns: Environment, Revision, Status, URL. One deployment is shown for 'test' environment, revision 1, with a green status and URL 'http://...'. A '+' icon is next to the URL.
- Proxy Endpoints:** A table with columns: Name, Base Path, Target Endpoints. One endpoint is shown for 'default' with base path '/oauth' and target endpoints 'none'. 'Expand All' and 'Collapse All' buttons are present.
- Target Endpoints:** No Target Endpoints for this proxy.

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

default の「+」ボタンをクリックし、New Conditional Flow を開きます。



Flow の情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Flow Name : authorize (任意の名前)
- Condition Type : 「Path and Verb」を選択
- Path : /authorize (任意のパス)
- Verb : 「GET」を選択

New Conditional Flow

Flow Name: authorize (1)

Description:

Condition Type:  Custom  Path and Verb (2)

Path: /authorize| (3)

Verb: GET (4)

Optional Target URL:

(5) ここをクリック! Add

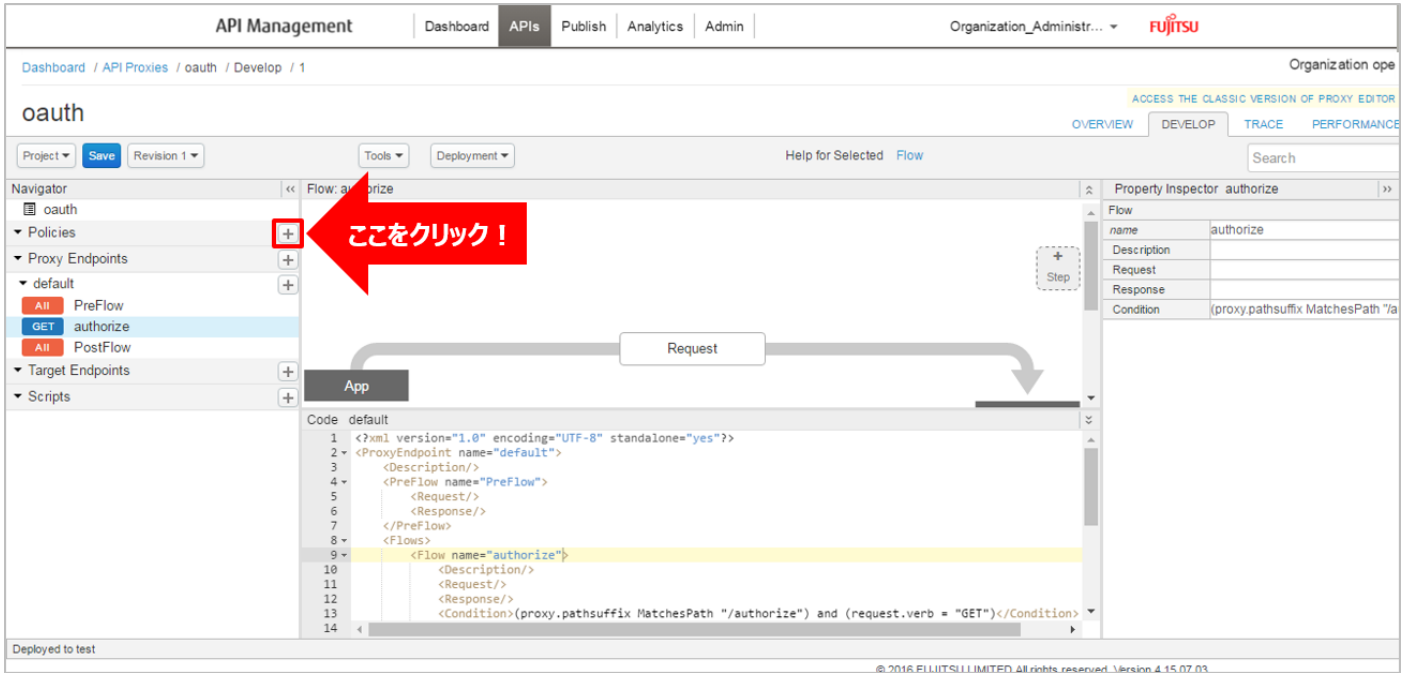
(2) client\_id チェック処理の実装 (API キー認証) (4)

client\_id のチェック処理を実装します。

client\_id の検証・認証処理を行う Verify API Key ポリシーを追加します。

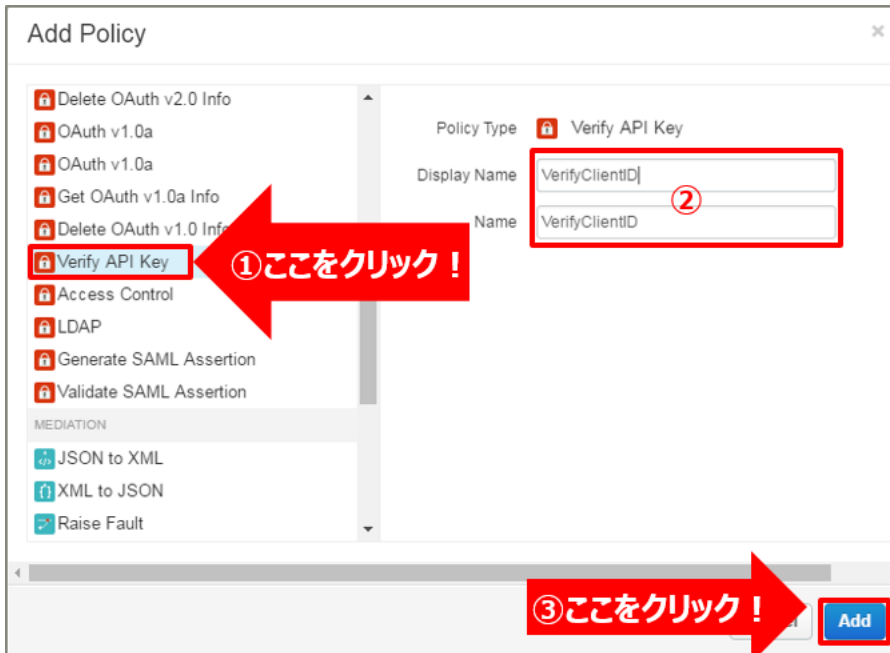
事前に登録されたアプリからのリクエストのみ、API へのアクセスを許可するようにします。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Verify API Key」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : VerifyClientID (任意の名前)
- Name : VerifyClientID (任意の名前)



追加した Verify API Key ポリシーの編集画面を表示されます。  
必要に応じてポリシーの定義を編集します。

API Management

Dashboard / API Proxies / oauth / Develop / 1

oauth

Project Save Revision 1 Tools

Navigator

- oauth
- Policies
  - VerifyClientID
- Proxy Endpoints
  - default
    - All PreFlow
    - GET authorize
    - All PostFlow
  - Target Endpoints
  - Scripts

Policy: VerifyClientID

Display Name VerifyClientID

Name VerifyClientID

Attached to Flows Not used in any flows

Code VerifyClientID

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <VerifyAPIKey async="false" continueOnError="false" enabled="true" name="VerifyClientID">
3   <DisplayName>VerifyClientID</DisplayName>
4   <Properties/>
5   <APIKey ref="request.queryparam.client_id"/>
6 </VerifyAPIKey>
```

client\_idの取得先を指定

①ここをクリック!

ポリシー編集画面

②編集

Deployed to test

© 2016 FUJITSU LIMITED. All rights reserved. Version 4.15.07.03

#### 【定義例】

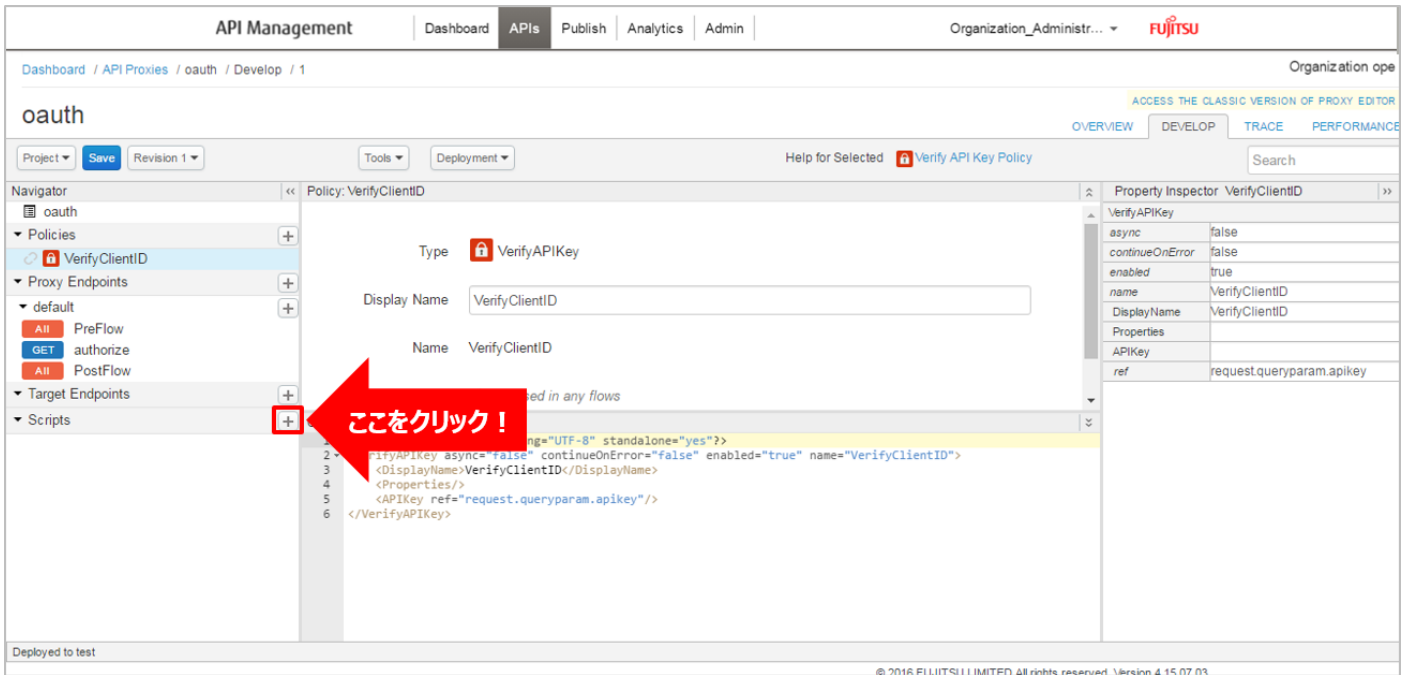
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<VerifyAPIKey async="false" continueOnError="false" enabled="true" name="VerifyClientID">
  <DisplayName>VerifyClientID</DisplayName>
  <Properties/>
  <APIKey ref="request.queryparam.client_id"/>
</VerifyAPIKey>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A1.1. Verify API Key XML仕様](#)」をご参照ください。

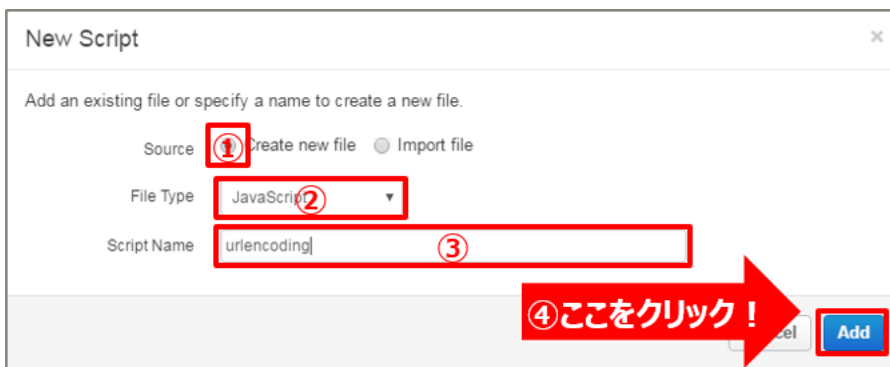
(3) クエリパラメーターエンコーディング処理の実装 (JavaScript) (4) スクリプトファイルを作成します。

Scripts の「+」ボタンをクリックし、New Script を開きます。



スクリプトの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Source : 「Create new file」を選択
- File Type : 「JavaScript」を選択
- Script Name : urlencoding.js (任意の名前)



Scripts に作成されたスクリプトを選択し、スクリプト編集画面を表示します。  
クエリパラメーターから情報を抽出してエンコーディングします。

【定義例】

```
var client_id = context.getVariable("request.queryparam.client_id");
var state = context.getVariable("request.queryparam.state");
var scope = context.getVariable("request.queryparam.scope");
var redirect_uri = context.getVariable("request.queryparam.redirect_uri");

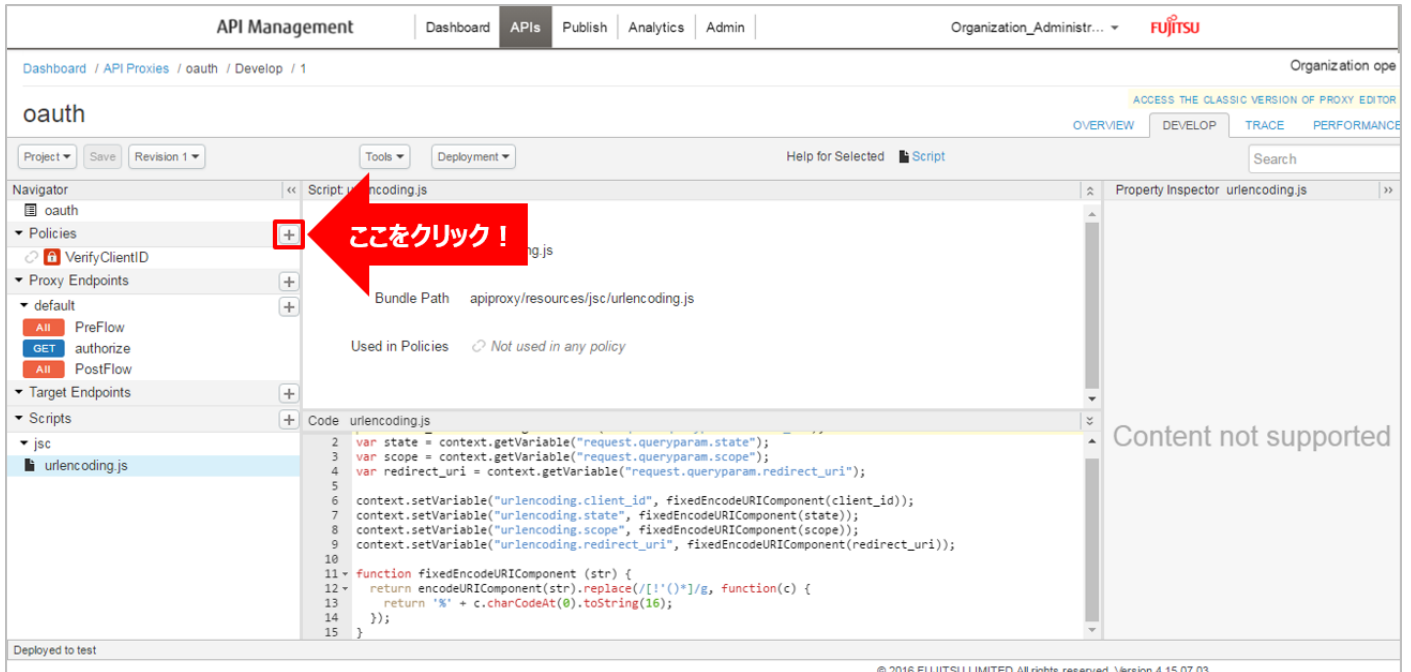
context.setVariable("urlencoding.client_id", fixedEncodeURIComponent(client_id));
context.setVariable("urlencoding.state", fixedEncodeURIComponent(state));
context.setVariable("urlencoding.scope", fixedEncodeURIComponent(scope));
context.setVariable("urlencoding.redirect_uri", fixedEncodeURIComponent(redirect_uri));

function fixedEncodeURIComponent (str) {
  return encodeURIComponent(str).replace(/[\!'()*]/g, function(c) {
    return '%' + c.charCodeAt(0).toString(16);
  });
}
```



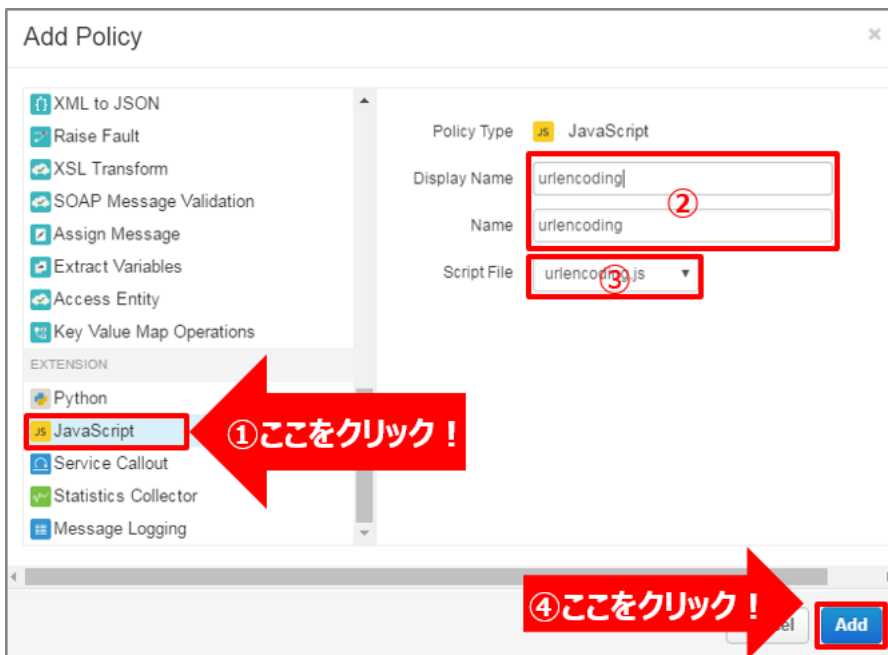
JavaScript を実行するための JavaScript ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「JavaScript」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。  
以下は入力例です。

- Display Name : urlencoding (任意の名前)
- Name : urlencoding (任意の名前)
- Script File : 「urlencoding.js」を選択 (前述の手順で作成したスクリプトを選択)

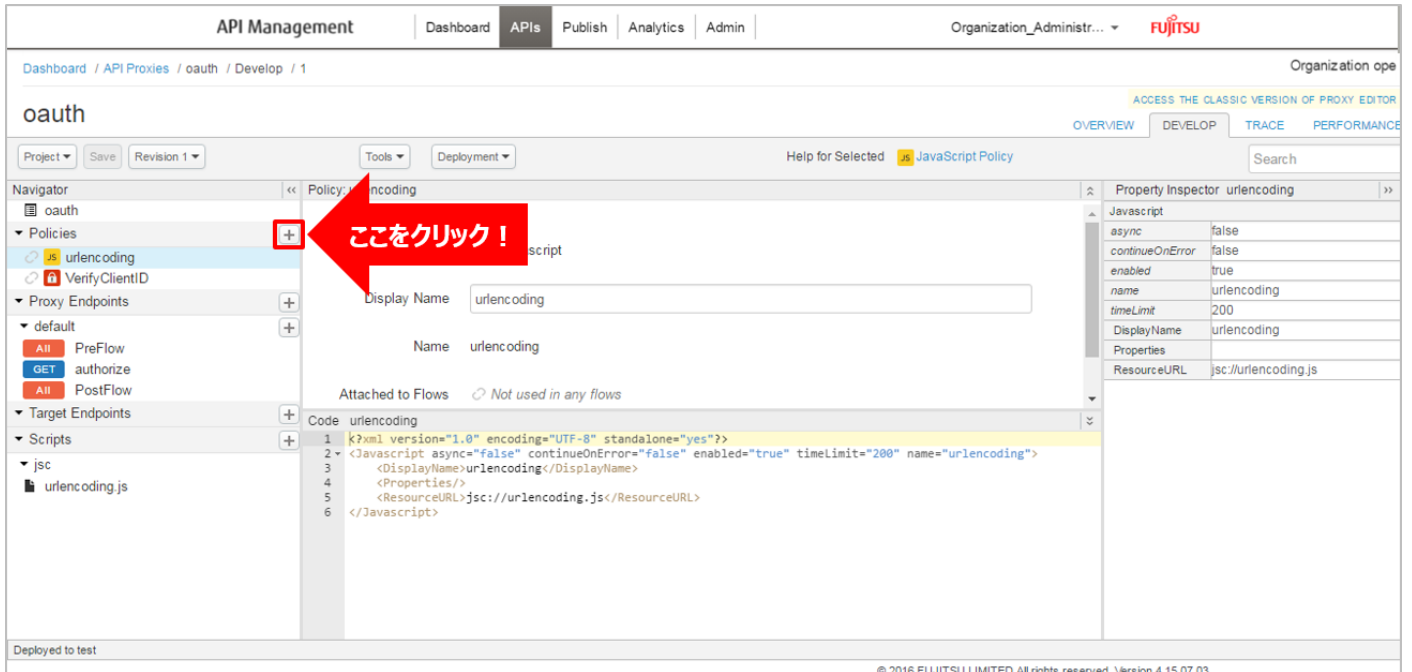


#### (4) ログインアプリへのリダイレクト処理の実装 (5)

ログインアプリへのリダイレクト処理を実装します。

HTTP メッセージを作成する Assign Message ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Assign Message」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : RedirectLoginAPP (任意の名前)
- Name : RedirectLoginAPP (任意の名前)



追加した Assign Message ポリシーを選択し、ポリシー編集画面を表示します。  
 必要に応じてポリシーの定義を編集します。  
 リダイレクト先として、ログインアプリの URI を指定します。

**API Management**

Dashboard / API Proxies / oauth / Develop / 1

oauth

Project Save Revision 1 Tools

Navigator

- oauth
  - urlencoding
  - Verify ClientID
  - Proxy Endpoints
    - default
      - PreFlow
        - authorize
      - PostFlow
    - Target Endpoints
    - Scripts
      - urlencoding.js

Policy: RedirectLoginAPP

Display Name

Attached to Flow

Code RedirectLoginAPP

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <AssignMessage async="false" continueOnError="false" enabled="true" name="RedirectLoginAPP">
3   <DisplayName>RedirectLoginAPP</DisplayName>
4   <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
5   <AssignTo createNew="true" type="response">response</AssignTo>
6   <Remove>
7     <Payload>true</Payload>
8   </Remove>
9   <Set>
10    <StatusCode>302</StatusCode>
11    <ReasonPhrase>Found</ReasonPhrase>
12    <Headers>
13      <Header name="Location">
14        <![CDATA[http://example.com:8080/oauth2-sample/loginApp-
15        login.jsf?client_id={urlencoding.client_id}&state={urlencoding.state}&scope={urlencoding.scope}&redirect_uri={urlencoding.redirect_uri}]]>
16      </Header>
17    </Headers>
18  </Set>
19  <!-- Set this flow variable to indicate the response has been set -->
20  <AssignVariable>
21    <Name>flowResponse.ready</Name>
22    <Value>true</Value>
23  </AssignVariable>
24 </AssignMessage>
  
```

後述の認証画面のURL

ここをクリック!

編集

ポリシー編集画面

Deployed to test

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.03

## 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true"
name="RedirectLoginAPP">
  <DisplayName>RedirectLoginAPP</DisplayName>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="true" type="response">response</AssignTo>
  <Remove>
    <Payload>true</Payload>
  </Remove>
  <Set>
    <StatusCode>302</StatusCode>
    <ReasonPhrase>Found</ReasonPhrase>
    <Headers>
      <Header name="Location">
        ![CDATA[http://example.com:8080/oauth2-sample/loginApp-
login.jsf?client_id={urlencoding.client_id}&state={urlencoding.state}&scope={urlencoding.scope
}&redirect_uri={urlencoding.redirect_uri}]]>
      </Header>
    </Headers>
  </Set>
  <!-- Set this flow variable to indicate the response has been set -->
  <AssignVariable>
    <Name>flowResponse.ready</Name>
    <Value>true</Value>
  </AssignVariable>
</AssignMessage>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.3. Assign Message XML 仕様](#)」をご参照ください。

(5) authorize API へのポリシー適用 (④, ⑤)

以下の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。

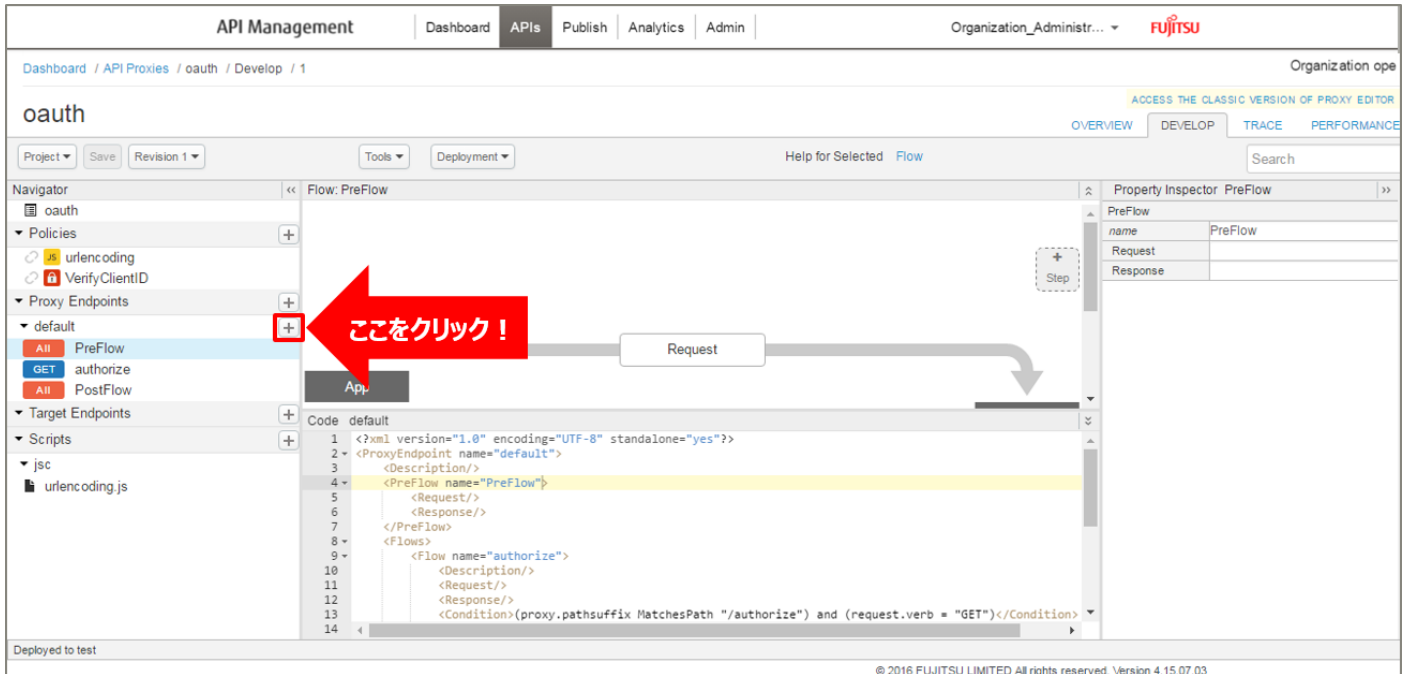
The screenshot displays the API Management console for the 'oauth' project. The 'authorize' API is selected, and its configuration is shown in the central flow diagram. The flow consists of a 'Request' step and a 'Response' step. The 'Request' step is currently empty, while the 'Response' step has a 'RedirectLoginAPP' policy attached. The 'Property Inspector' on the right shows the details of the 'authorize' flow, including the 'Request' and 'Response' steps. The 'Request' step is currently empty, and the 'Response' step has a 'RedirectLoginAPP' policy attached. The 'Save' button is highlighted with a red box and arrow labeled ④. The 'authorize' endpoint is highlighted with a red box and arrow labeled ①. The 'VerifyClientID' and 'urlencoding' policies are highlighted with a red box and arrow labeled ②. The 'RedirectLoginAPP' policy is highlighted with a red box and arrow labeled ③.

## 2-3) authorizationcode API の作成 (⑧)

フローの ⑧ の処理を実装します。

### (1) Conditional Flow (POST) の作成 (authorizationcode エンドポイントの作成)

default の「+」ボタンをクリックし、New Conditional Flow を開きます。



Flow の情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Flow Name : authorizationcode (任意の名前)
- Condition Type : 「Path and Verb」を選択
- Path : /authorizationcode (任意のパス)
- Verb : 「POST」を選択

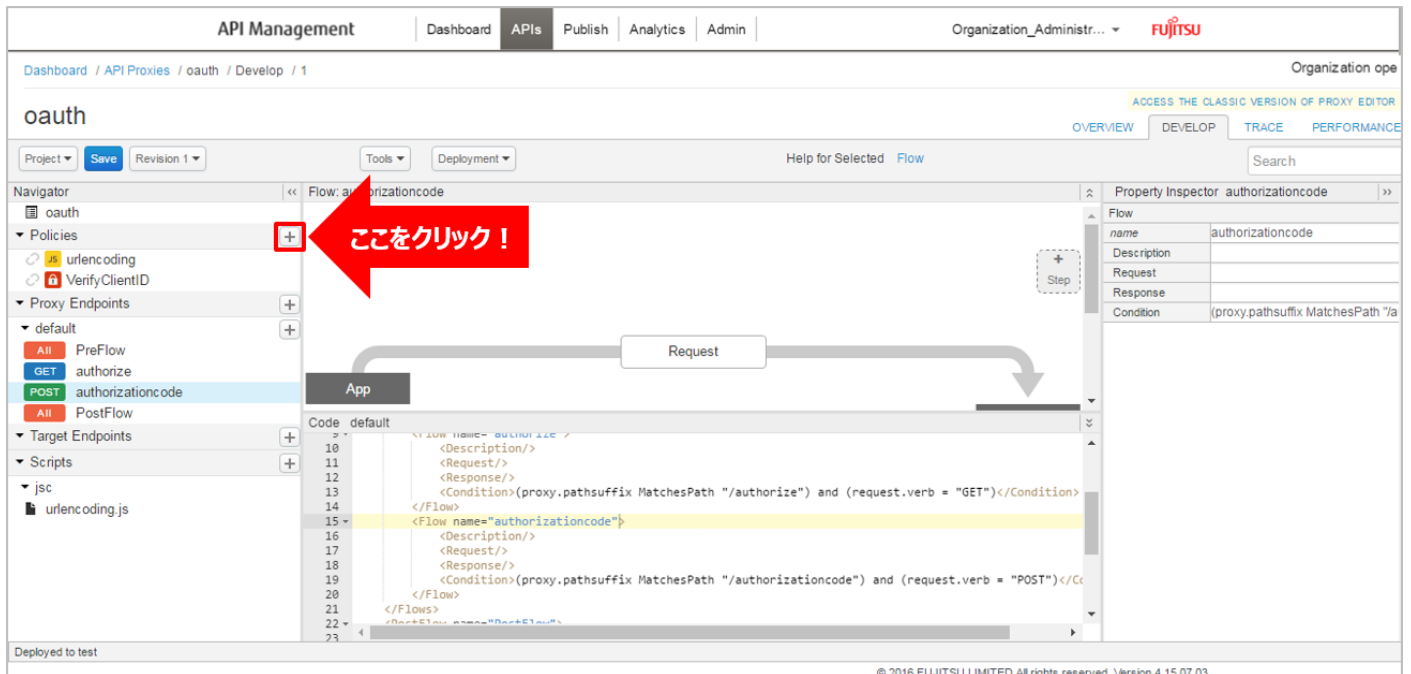
The 'New Conditional Flow' dialog box is shown. The 'Flow Name' field contains 'authorizationcode' (1). The 'Condition Type' is set to 'Path and Verb' (2). The 'Path' field contains '/authorizationcode' (3). The 'Verb' dropdown is set to 'POST' (4). A red arrow points to the 'Add' button (5) with the text 'ここをクリック!' (Click here!).

## (2) Authorization Code 発行処理の実装 (8)

Authorization Code 発行処理を実装します。

OAuth v2.0 ポリシー (GenerateAuthorizationCode) を追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「OAuth v2.0」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。

以下は入力例です。

- Display Name : GenerateAuthorizationCode (任意の名前)
- Name : GenerateAuthorizationCode (任意の名前)



追加した OAuth v2.0 ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

The screenshot shows the API Management interface. On the left, the 'GenerateAuthorizationCode' policy is selected under the 'oauth' group. A red arrow labeled '①ここをクリック！' points to this policy. The main area displays the XML definition of the policy, with the 'Attributes' section highlighted in yellow. A red arrow labeled '②編集' points to the '任意情報格納設定' (Custom Information Storage Settings) section on the right, which shows a table of attributes: 'apirole' and 'id', both with 'display' set to 'false'.

任意情報格納設定とは、ユーザーを識別するための ID などの任意情報を Access Token に紐付ける設定です。以下の例の場合、クエリパラメータの apirole, id の値を Access Token に紐付けます。

Access Token に紐付けられた任意情報は API Proxy 上で変数 (accesstoken.id, accesstoken.apirole) として利用可能です。

【定義例】

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OAuthV2 async="false" continueOnError="false" enabled="true"
name="GenerateAuthorizationCode">
  <DisplayName>GenerateAuthorizationCode</DisplayName>
  <Operation>GenerateAuthorizationCode</Operation>
  <GenerateResponse enabled="true"/>
  <ExpiresIn>60000</ExpiresIn>
  <ClientId>request.queryparam.client_id</ClientId>
  <RedirectUri>request.queryparam.redirect_uri</RedirectUri>
  <ResponseType>request.queryparam.response_type</ResponseType>
  <State>request.queryparam.state</State>
  <Attributes>
    <Attribute name="apirole" ref="request.queryparam.apirole" display="false"/>
    <Attribute name="id" ref="request.queryparam.id" display="false"/>
  </Attributes>
</OAuthV2>

```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。



※定義内容の詳細は、「[A1.2. OAuth v2.0 \(GenerateAuthorizationCode\) XML仕様](#)」をご参照ください。

### (3) authorizationcode API へのポリシー適用 (8)

以下の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。

The screenshot shows the API Management console for the 'oauth' API. The main area displays a flow diagram with 'Request' and 'Response' steps. A 'Server' icon is also present. The 'Policies' list on the left includes 'GenerateAuthorizationCode', 'urlencoding', and 'VerifyClientID'. The 'Proxy Endpoints' list includes 'PreFlow', 'authorize', 'authorizationcode', and 'PostFlow'. The 'Scripts' list includes 'jsc' and 'urlencoding.js'. The 'Save' button is highlighted in the top left. The 'GenerateAuthorizationCode' policy is highlighted in the Policies list. The 'POST authorizationcode' endpoint is highlighted in the Proxy Endpoints list. The 'GenerateAuthorizationCode' policy icon is highlighted in the flow diagram. The Property Inspector on the right shows the configuration for the 'authorizationcode' flow, including the name, description, request, and response steps.

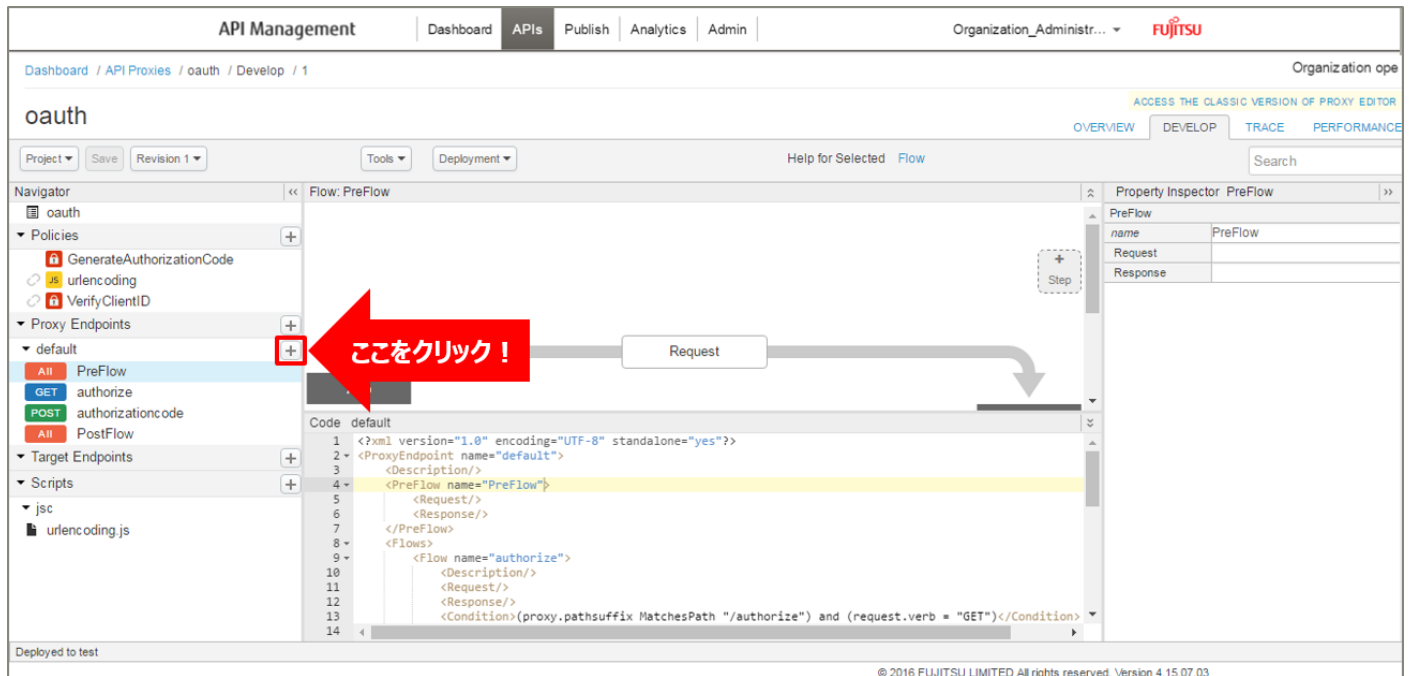
Property Inspector authorizationcode	
Flow	
name	authorizationcode
Description	
Request	
Response	
Step	
Name	GenerateAuthorizationCode
Condition	(proxy.pathsuffix.MatchesPath "a

## 2-4) accesstoken API の作成 (⑩)

フローの ⑩ の処理を実装します。

### (1) Conditional Flow (POST) の作成 (accesstoken エンドポイントの作成)

default の「+」ボタンをクリックし、New Conditional Flow を開きます。



Flow の情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Flow Name : accesstoken (任意の名前)
- Condition Type : 「Path and Verb」を選択
- Path : /accesstoken (任意のパス)
- Verb : 「POST」を選択

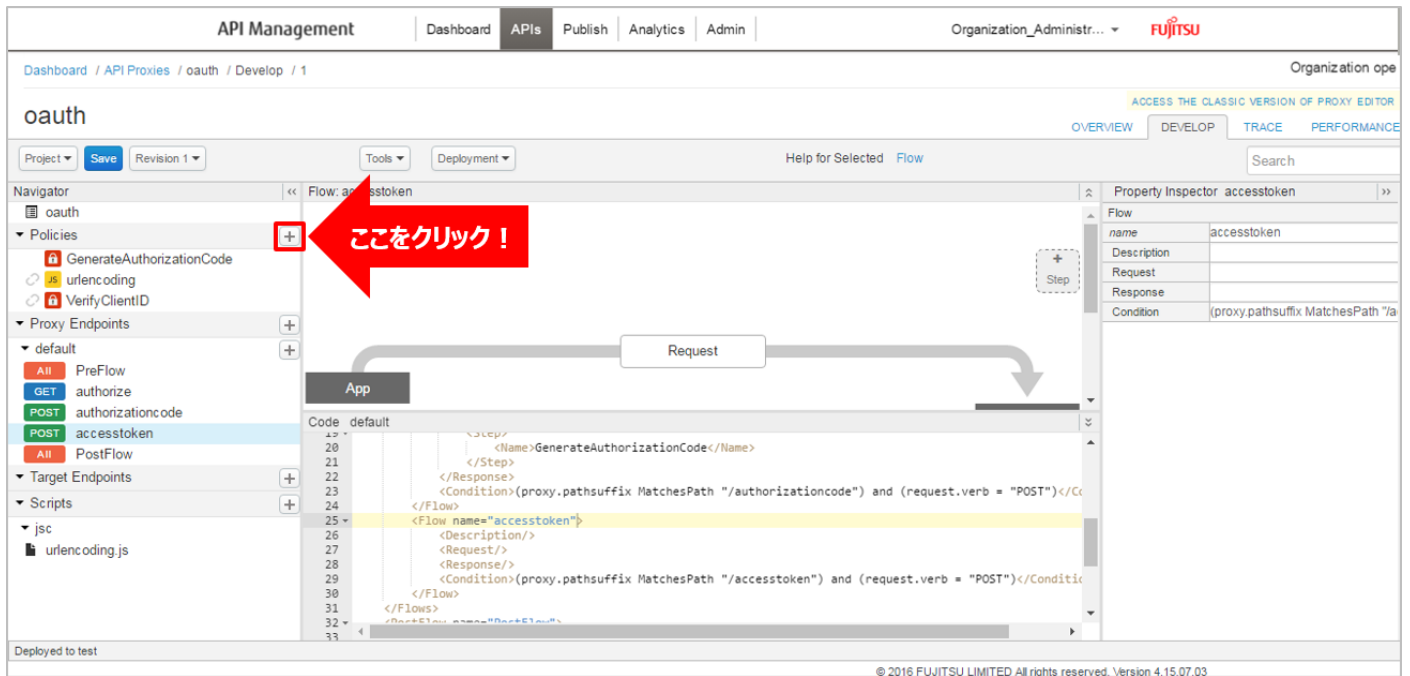
The 'New Conditional Flow' dialog box is shown. The 'Flow Name' field is set to 'accesstoken' (1). The 'Condition Type' is set to 'Path and Verb' (2). The 'Path' field is set to '/accesstoken' (3). The 'Verb' dropdown is set to 'POST' (4). A red arrow points to the 'Add' button with the text 'ここをクリック!'.

## (2) Access Token 発行処理の実装 (⑩)

Access Token 発行処理を実装します。

OAuth v2.0 ポリシー (GenerateAccessToken) を追加します。

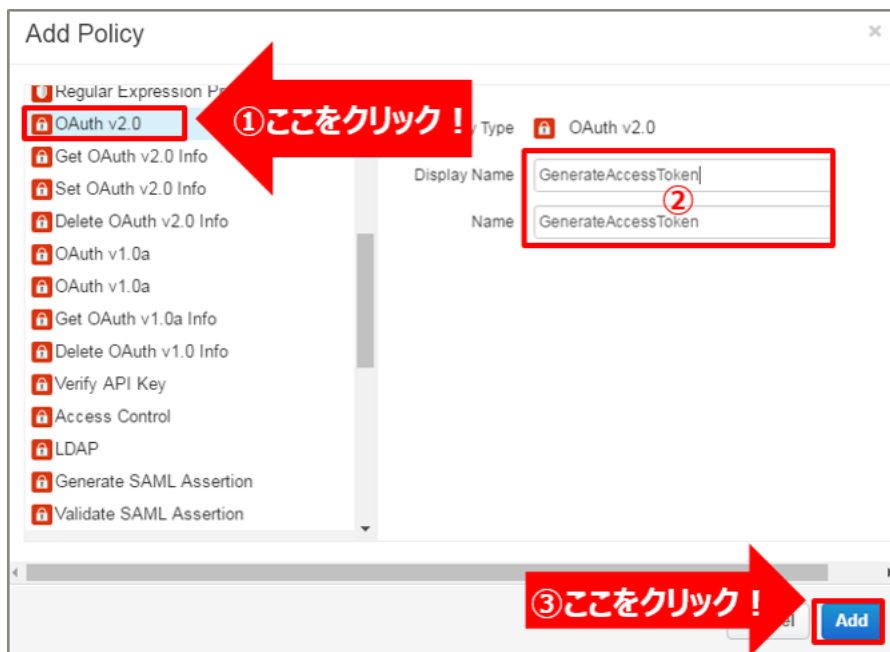
Policies の「+」ボタンをクリックし、Add Policy を開きます。



「OAuth v2.0」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。

以下は入力例です。

- Display Name : GenerateAccessToken (任意の名前)
- Name : GenerateAccessToken (任意の名前)



追加した OAuth v2.0 ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

The screenshot shows the API Management interface. On the left, the 'Policies' list includes 'GenerateAccessToken', which is highlighted with a red box and a red arrow labeled '①ここをクリック！'. The main editor displays the XML configuration for the 'GenerateAccessToken' policy. A red box highlights the XML code, and a red arrow labeled '②編集' points to the 'Edit' button in the bottom right corner. The XML code is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OAuthV2 async="false" continueOnError="false" enabled="true" name="GenerateAccessToken">
  <DisplayName>GenerateAccessToken</DisplayName>
  <Operation>GenerateAccessToken</Operation>
  <SupportedGrantTypes>
    <GrantType>authorization_code</GrantType>
  </SupportedGrantTypes>
  <GenerateResponse enabled="true"/>
  <ExpiresIn>60000</ExpiresIn>
  <GrantType>request.queryparam.grant_type</GrantType>
  <Code>request.queryparam.code</Code>
  <RedirectUri>request.queryparam.redirect_uri</RedirectUri>
  <Scope>request.queryparam.scope</Scope>
  <State>request.queryparam.state</State>
</OAuthV2>
```

#### 【定義例】

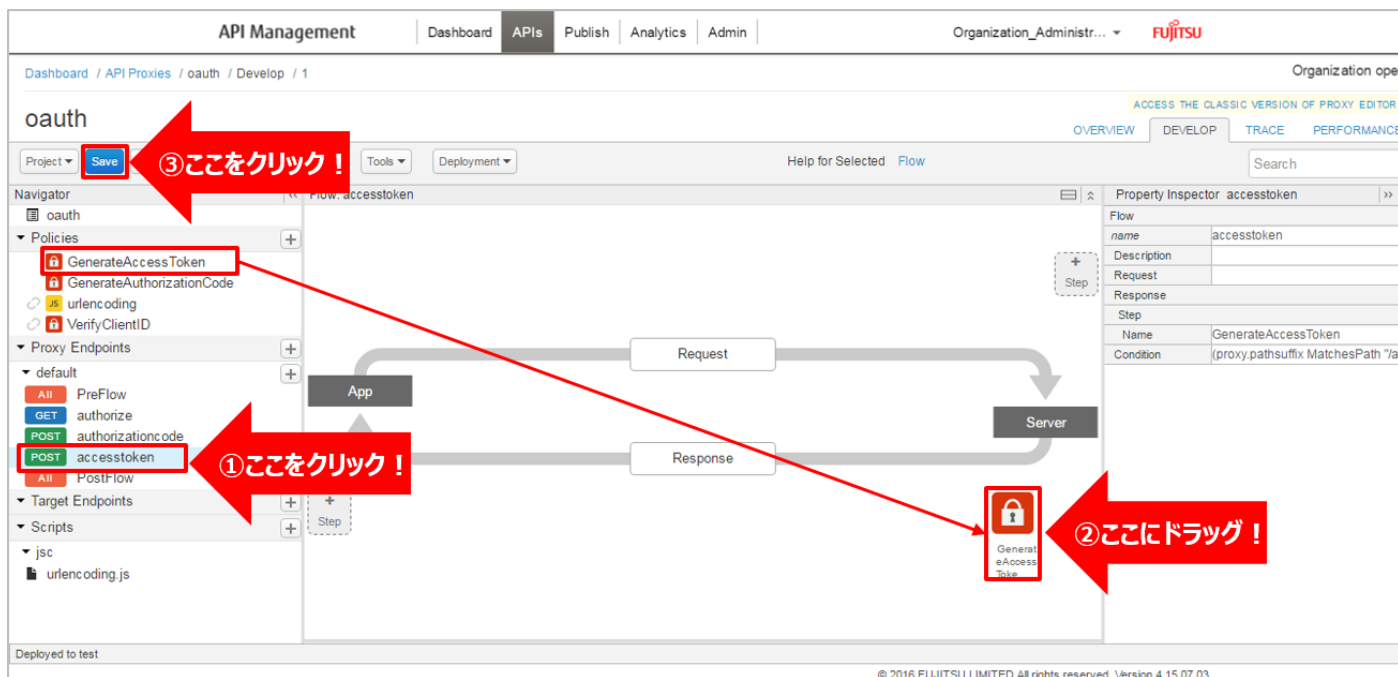
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OAuthV2 async="false" continueOnError="false" enabled="true"
name="GenerateAccessToken">
  <DisplayName>GenerateAccessToken</DisplayName>
  <Operation>GenerateAccessToken</Operation>
  <SupportedGrantTypes>
    <GrantType>authorization_code</GrantType>
  </SupportedGrantTypes>
  <GenerateResponse enabled="true"/>
  <ExpiresIn>60000</ExpiresIn>
  <GrantType>request.queryparam.grant_type</GrantType>
  <Code>request.queryparam.code</Code>
  <RedirectUri>request.queryparam.redirect_uri</RedirectUri>
  <Scope>request.queryparam.scope</Scope>
  <State>request.queryparam.state</State>
</OAuthV2>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A1.3. OAuth v2.0 \(GenerateAccessToken\) XML仕様](#)」をご参照ください。

### (3) accesstoken API へのポリシー適用 (10)

以下の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。

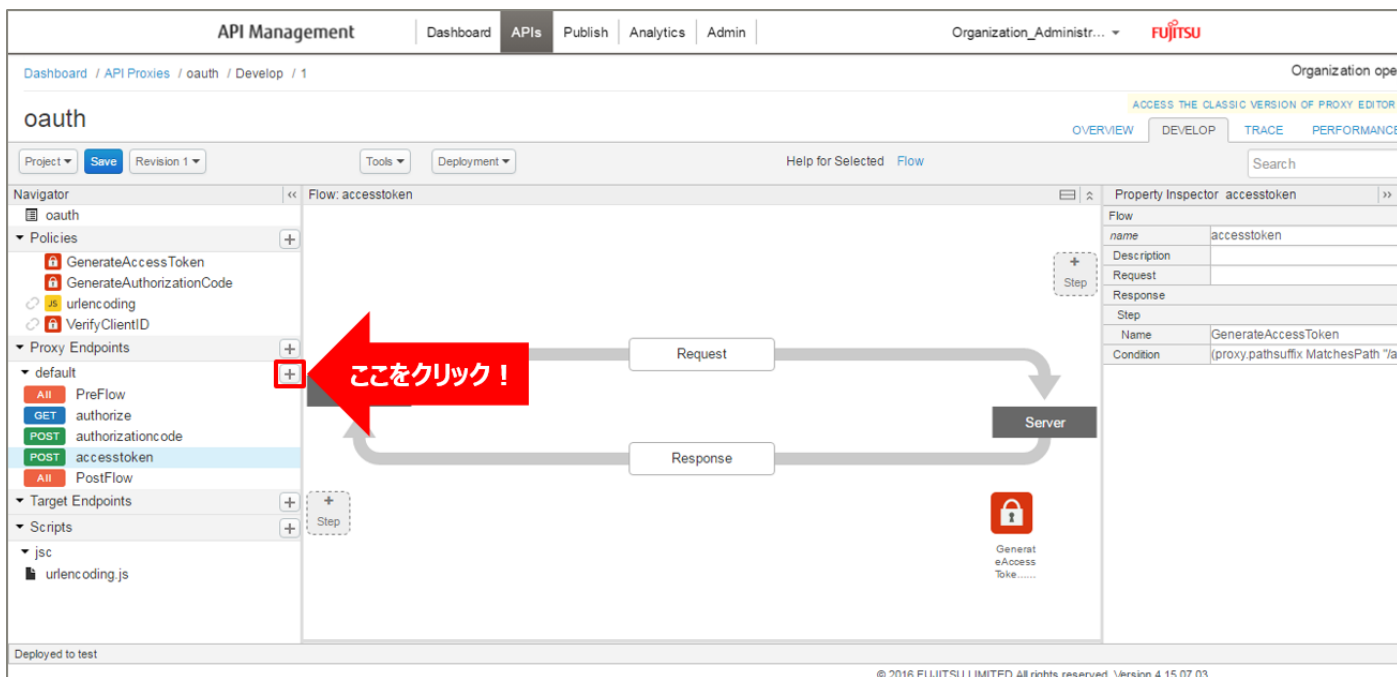


### 2-5) refreshtoken API の作成

Access Token の有効期限が切れた場合の再発行処理を実装します。

#### (1) Conditional Flow (POST) の作成 (refreshtoken エンドポイントの作成)

default の「+」ボタンをクリックし、New Conditional Flow を開きます。



Flow の情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Flow Name : refreshtoken (任意の名前)
- Condition Type : 「Path and Verb」を選択
- Path : /refreshtoken (任意のパス)
- Verb : 「POST」を選択

New Conditional Flow

Flow Name: refreshtoken ①

Description:

Condition Type:  Custom  Path and Verb ②

Path: /refreshtoken ③

Verb: POST ④

Optional Target URL:

⑤ここをクリック!

Add

## (2) Refresh Token 発行処理の実装

Refresh Token 発行処理を実装します。

OAuth v2.0 ポリシー (RefreshAccessToken) を追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。

API Management | Dashboard | APIs | Publish | Analytics | Admin | Organization\_Administr... | FUJITSU

Dashboard / API Proxies / oauth / Develop / 1

oauth

Project | Save | Revision 1 | Tools | Deployment | Help for Selected Flow | Search

Navigator: oauth

- oauth
- ▼ Policies
- GenerateAccessToken
- GenerateAuthorizationCode
- urlencoding
- VerifyClientID
- ▼ Proxy Endpoints
- default
- All PreFlow
- GET authorize
- POST authorizationcode
- POST accesstoken
- POST refreshtoken
- All PostFlow
- ▼ Target Endpoints
- ▼ Scripts
- jsc
- urlencoding.js

Flow: refreshtoken

Property Inspector: refreshtoken

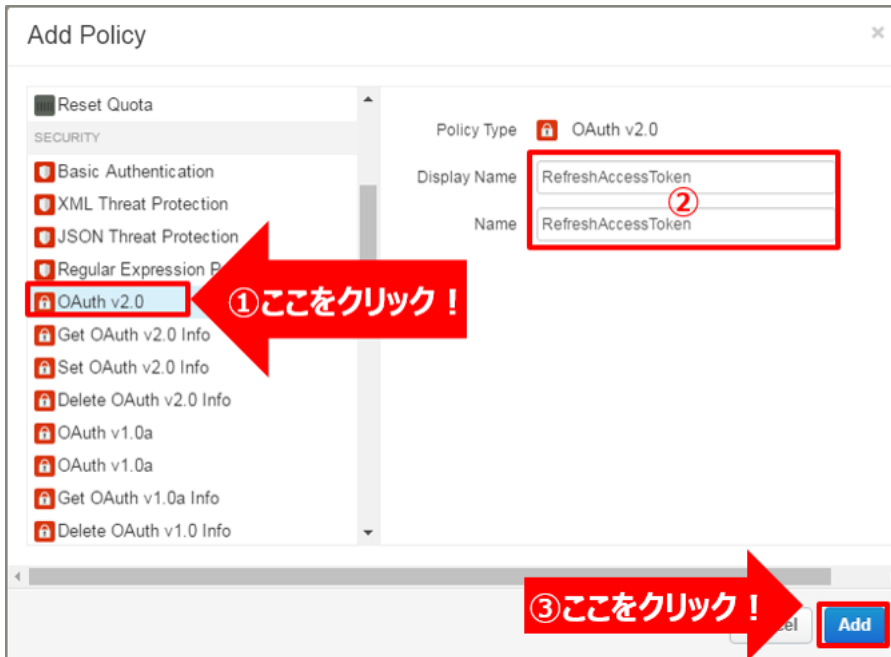
Flow	
name	refreshtoken
Description	
Request	
Response	
Condition	(proxy.pathsuffix MatchesPath "/re

Deployed to test

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.03

「OAuth v2.0」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。  
 以下は入力例です。

- Display Name : RefreshAccessToken (任意の名前)
- Name : RefreshAccessToken (任意の名前)



追加した OAuth v2.0 ポリシーを選択し、ポリシー編集画面を表示します。  
 必要に応じてポリシーの定義を編集します。



## 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OAuthV2 async="false" continueOnError="false" enabled="true" name="RefreshAccessToken">
  <DisplayName>RefreshAccessToken</DisplayName>
  <ExternalAuthorization>false</ExternalAuthorization>
  <Operation>RefreshAccessToken</Operation>
  <GenerateResponse enabled="true"/>
  <ExpiresIn>120000</ExpiresIn>
  <GrantType>request.queryparam.grant_type</GrantType>
  <RefreshToken>request.queryparam.refresh_token</RefreshToken>
  <Attributes>
    <Attribute name="apirole" ref="request.queryparam.apirole" display="false"/>
  </Attributes>
</OAuthV2>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A1.4. OAuth v2.0 \(RefreshAccessToken\) XML 仕様](#)」をご参照ください。

### (3) refreshtoken API へのポリシー適用

以下の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。

The screenshot shows the API Management console interface. The main area displays a flow diagram with 'App' and 'Server' components. A 'RefreshAccessToken' policy icon is being dragged from the left-hand 'Policies' list to the flow. Three red arrows with Japanese text provide instructions: ① 'ここをクリック!' (Click here!) points to the 'refreshtoken' endpoint in the 'Proxy Endpoints' list; ② 'ここにドラッグ!' (Drag here!) points to the 'RefreshAccessToken' policy icon; ③ 'ここをクリック!' (Click here!) points to the 'Save' button in the top left.

Property Inspector refresh_accesstoken	
Flow	
name	refresh_accesstoken
Description	
Request	
Response	
Step	
Name	RefreshAccessToken
Condition	[proxy.pathsuffix MatchesPath "re



### 3) API Proxy への Access Token 認証適用 (⑪)

API 提供者が実施します。

本サービスを経由した OAuth 2.0 Authorization Code Grant フローの ⑪ の処理を実装します。これから作成していく API Proxy に、「2) Access Token 発行 API Proxy の作成」で作成した認証 API を組み込みます。

#### 3-1) API Proxy の作成

##### (1) API Proxy の作成

「2) Access Token 発行 API Proxy の作成」の手順を参考に、API Proxy を作成します。次の画面では、以下のように設定します。

##### ➤ Build a Proxy (TYPE)

「Reverse proxy (most common)」を選択します。

##### ➤ Build a Proxy (DETAILS)

作成する API Proxy の情報を入力します。

- Proxy Name : 任意の名前
- Proxy Base Path : 任意のパス (自動入力)
- Existing API : アクセス先 API の URL

##### ➤ Build a Proxy (SECURITY)

Authentication : 「OAuth 2.0」、「Publish API Product」を選択します。

##### (2) Conditional Flow の作成

「2) Access Token 発行 API Proxy の作成」の手順を参考に、Conditional Flow を作成します。New Conditional Flow では以下のように設定します。

- Flow Name : 任意の名前
- Condition Type : 「Path and Verb」を選択
- Path : 任意のパス
- Verb : 任意のメソッド

### (3) リソースサーバへのリクエスト設定

リソースサーバへのリクエスト情報を設定します。

以下の例では、Assign Message ポリシーを使用して、Access Token に関連付けられている apirole, id をリクエストヘッダーに設定しています。

#### 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true" name="Set-Header-
OAuth-v20">
  <DisplayName>Set Header OAuth v20</DisplayName>
  <Properties/>
  <Add>
    <Headers>
      <Header name="apirole">{accesstoken.apirole}</Header>
      <Header name="id">{accesstoken.id}</Header>
    </Headers>
  </Add>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="request"/>
</AssignMessage>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.3. Assign Message XML 仕様](#)」をご参照ください。

#### (4) リソースサーバからのレスポンス編集

リソースサーバからのレスポンス情報を編集します。

以下の例では、Assign Message ポリシーを使用して、リソースサーバからのレスポンス情報をクライアントに返却する前に、apirole, id をリクエストヘッダーから削除しています。

これにより、クライアントに情報が送信されず隠蔽することができます。

##### 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true" name="Remove-Header-
OAuth-v20">
  <DisplayName>Remove Header OAuth v20</DisplayName>
  <Properties/>
  <Remove>
    <Headers>
      <Header name="apirole"/>
      <Header name="id"/>
    </Headers>
  </Remove>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="request"/>
</AssignMessage>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.3. Assign Message XML 仕様](#)」をご参照ください。

## (5) ポリシーの配置

以下の例の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。

The screenshot displays the API Management console for a project named 'weather-oauth2'. The 'Policies' section is active, showing a flow diagram with 'App' and 'Server' components. Two policy configuration windows are overlaid on the interface:

- Set Header OAuth v20:** A configuration window showing XML code for adding headers. The code includes an `<Add>` block with `<Headers>` containing `<Header name="apirole">{accesstoken.apirole}</Header>` and `<Header name="id">{accesstoken.id}</Header>`.
- Remove Header OAuth v20:** A configuration window showing XML code for removing headers. The code includes a `<Remove>` block with `<Headers>` containing `<Header name="apirole"/>` and `<Header name="id"/>`.

The right-hand side of the console shows a 'Step' configuration table for the 'Set-Header-OAuth-v20' policy:

Description	Request
Step	
Name	Set-Header-OAuth-v20
Response	
Step	
Name	remove-header-authorization
Condition	(proxy.pathsuffix MatchesPath "/fc

以上で API 提供者による API Proxy の定義が完了し、API 利用者からの利用申請を受け付けることができますようになります。

## 4) API Proxy へのアプリ登録申請 (③)

API 利用者は、API 提供者に以下の情報を提示して、本サービスへのアプリ登録申請を行います。アプリ登録後、API 提供者から登録したアプリの `client_id`, `client_secret` が提供されます。

- アプリ名
- Callback URL
- Scope
- 利用 API
- 苗字
- 名前
- メールアドレス
- 本サービス上のユーザー名（半角英数字）

## 5) API Proxy へのアプリ登録 (③)

「4) API Proxy へのアプリ登録申請」の申請内容に基づき、API 提供者は本サービスにアプリの登録を行います。

### 5-1) Product の作成

Product を作成します。

Product とは、作成した API Proxy をグループ化したものです。

API キー認証等のセキュリティ設定は、作成した Product 単位で管理することができます。

画面上部の「Publish」メニューから「Products」をクリックして Products 画面に遷移します。

The screenshot shows the API Management interface. The top navigation bar includes 'Dashboard', 'APIs', 'Publish', 'Analytics', and 'Admin'. The 'Publish' menu is open, showing 'Products', 'Developers', and 'Developer Apps'. A red arrow points to 'Products' with the text 'ここをクリック!'. Below the menu, the 'API Proxies' section is visible, showing a table of proxies with columns for API Proxy, Environments, Traffic, Message Trend by Hour, Avg Time, Error Rate, Modified, and Actions.

「+ Product」ボタンをクリックします。

The screenshot shows the 'Products' page in the API Management interface. The top navigation bar includes 'Dashboard', 'APIs', 'Publish', 'Analytics', and 'Admin'. The 'Publish' menu is open, showing 'Products', 'Developers', and 'Developer Apps'. A red arrow points to the '+ Product' button with the text 'ここをクリック!'. Below the button, the 'Products' section is visible, showing a table of products with columns for Product, Keys, Created, Modified, and Actions.

Product の情報を入力し、「Save」ボタンをクリックします。

◇ Product Details

- Name : 任意の名前
- Display Name : 任意の名前 (自動入力)
- Environment : 「test」, 「prod」 を選択
- Access : 「Public」 を選択
- Key Approval Type : 「Automatic」 を選択
- Allowed OAuth Scopes : 申請内容を入力

◇ Resources

- API Proxies : 「2) Access Token 発行 API Proxy の作成」, 「3) API Proxy への Access Token 認証適用」で作成した 2つの API Proxy を選択

The screenshot shows the 'API Management' console with the following sections and annotations:

- Product Details:**
  - Name:** weather-oauth2-product (Red box)
  - Display Name:** weather-oauth2-product
  - Environment:** test, prod (Red box)
  - Access:** Public (Red box)
  - Key Approval Type:** Automatic (Red box)
  - Allowed OAuth Scopes:** name, email, apirole (Red box)
- Resources:**
  - API Proxies:** A table with columns 'API Proxy' and 'Actions'. The 'API Proxy' column contains 'oauth2' and 'weather-oauth2'. The 'Actions' column has 'Delete' buttons. A red box highlights the '+ API Proxy' button, with a red arrow pointing to it and the text 'ここをクリック!' (Click here!).
- Custom:**
  - Attributes:** A table with columns 'Name' and 'Value'. A red box highlights the 'Save' button, with a red arrow pointing to it and the text 'ここをクリック!' (Click here!).

Annotations and instructions:

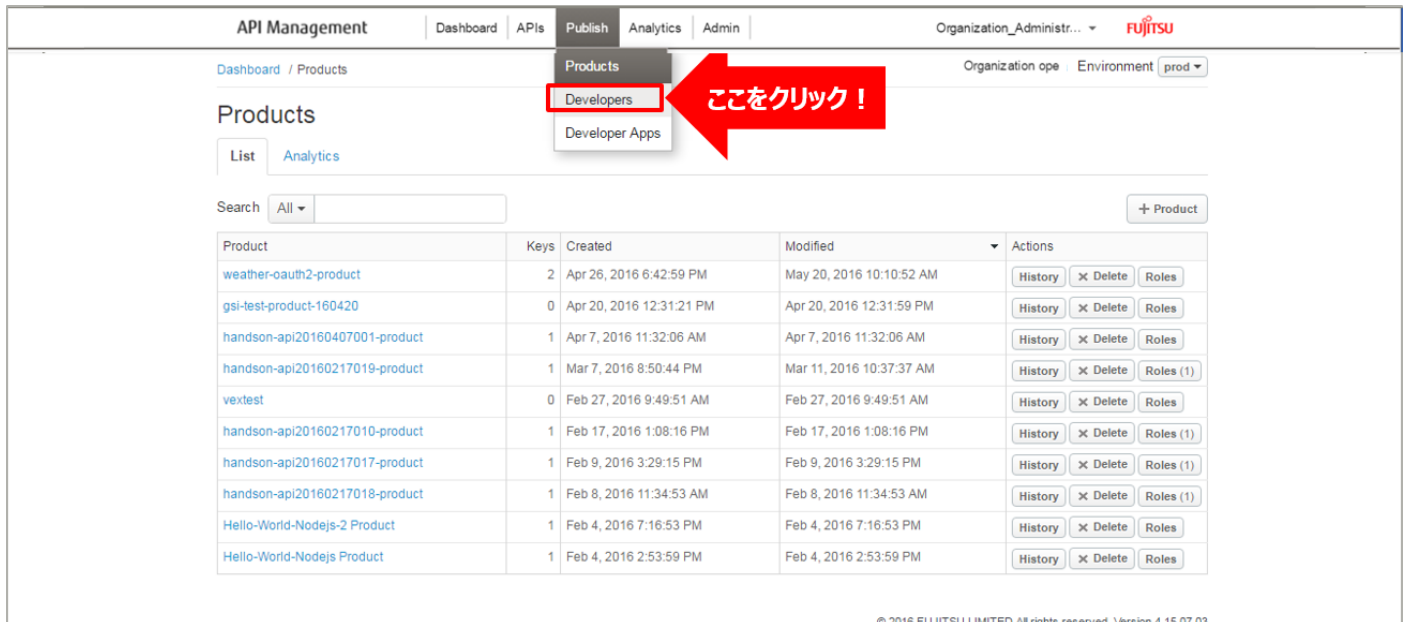
- Nameに「weather-oauth2-product」など申請された内容を入力します。
- Display Nameは自動入力されます。
- 以下の通り設定を行います。
  - Environment : test, prod
  - Access : Public
  - Key Approval Type : Automatic
- Scopeの定義を行います。(Optional) 複数の権限が必要な場合は「,」区切りで記述します。
- 申請された内容を入力します。
- Access Token発行API Proxy及びAccess Token認証を適用するAPI Proxyを選択します。
- API Proxyの追加
- 設定完了後Saveします。

## 5-2) アプリ開発者 (Developer) の登録

Developer を登録します。

登録した Developer は、次の手順でアプリ (Developer App) に割り当てます。

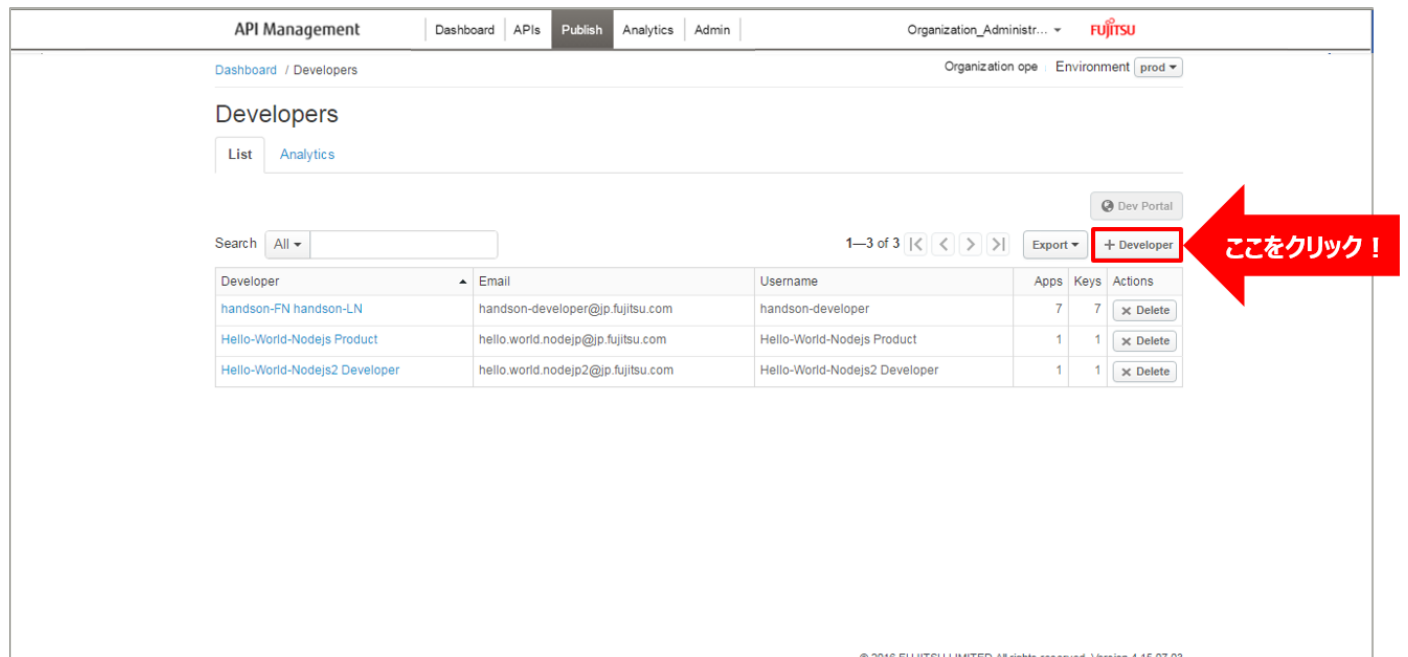
画面上部の「Publish」メニューから「Developers」をクリックして Developers 画面に遷移します。



The screenshot shows the API Management console interface. At the top, there is a navigation bar with tabs for 'Dashboard', 'APIs', 'Publish', 'Analytics', and 'Admin'. The 'Publish' tab is selected. Below the navigation bar, there is a breadcrumb trail 'Dashboard / Products'. A dropdown menu is open under 'Publish', showing options for 'Products', 'Developers', and 'Developer Apps'. The 'Developers' option is highlighted with a red box, and a red arrow points to it with the text 'ここをクリック!'. Below the dropdown, there is a search bar and a '+ Product' button. The main content area displays a table of products with columns for Product, Keys, Created, Modified, and Actions. The table contains several rows of product information.

Product	Keys	Created	Modified	Actions
<a href="#">weather-oauth2-product</a>	2	Apr 26, 2016 6:42:59 PM	May 20, 2016 10:10:52 AM	<a href="#">History</a> <a href="#">Delete</a> <a href="#">Roles</a>
<a href="#">gsi-test-product-16D420</a>	0	Apr 20, 2016 12:31:21 PM	Apr 20, 2016 12:31:59 PM	<a href="#">History</a> <a href="#">Delete</a> <a href="#">Roles</a>
<a href="#">handson-api20160407001-product</a>	1	Apr 7, 2016 11:32:06 AM	Apr 7, 2016 11:32:06 AM	<a href="#">History</a> <a href="#">Delete</a> <a href="#">Roles</a>
<a href="#">handson-api20160217019-product</a>	1	Mar 7, 2016 8:50:44 PM	Mar 11, 2016 10:37:37 AM	<a href="#">History</a> <a href="#">Delete</a> <a href="#">Roles (1)</a>
<a href="#">vextest</a>	0	Feb 27, 2016 9:49:51 AM	Feb 27, 2016 9:49:51 AM	<a href="#">History</a> <a href="#">Delete</a> <a href="#">Roles</a>
<a href="#">handson-api20160217010-product</a>	1	Feb 17, 2016 1:08:16 PM	Feb 17, 2016 1:08:16 PM	<a href="#">History</a> <a href="#">Delete</a> <a href="#">Roles (1)</a>
<a href="#">handson-api20160217017-product</a>	1	Feb 9, 2016 3:29:15 PM	Feb 9, 2016 3:29:15 PM	<a href="#">History</a> <a href="#">Delete</a> <a href="#">Roles (1)</a>
<a href="#">handson-api20160217018-product</a>	1	Feb 8, 2016 11:34:53 AM	Feb 8, 2016 11:34:53 AM	<a href="#">History</a> <a href="#">Delete</a> <a href="#">Roles (1)</a>
<a href="#">Hello-World-Nodejs-2 Product</a>	1	Feb 4, 2016 7:16:53 PM	Feb 4, 2016 7:16:53 PM	<a href="#">History</a> <a href="#">Delete</a> <a href="#">Roles</a>
<a href="#">Hello-World-Nodejs Product</a>	1	Feb 4, 2016 2:53:59 PM	Feb 4, 2016 2:53:59 PM	<a href="#">History</a> <a href="#">Delete</a> <a href="#">Roles</a>

「+ Developer」 ボタンをクリックします。



The screenshot shows the API Management console interface for the 'Developers' page. At the top, there is a navigation bar with tabs for 'Dashboard', 'APIs', 'Publish', 'Analytics', and 'Admin'. The 'Publish' tab is selected. Below the navigation bar, there is a breadcrumb trail 'Dashboard / Developers'. A dropdown menu is open under 'Publish', showing options for 'Products', 'Developers', and 'Developer Apps'. The 'Developers' option is highlighted with a red box, and a red arrow points to it with the text 'ここをクリック!'. Below the dropdown, there is a search bar and a '+ Developer' button. The main content area displays a table of developers with columns for Developer, Email, Username, Apps, Keys, and Actions. The table contains three rows of developer information.

Developer	Email	Username	Apps	Keys	Actions
<a href="#">handson-FN handson-LN</a>	handson-developer@jp.fujitsu.com	handson-developer	7	7	<a href="#">Delete</a>
<a href="#">Hello-World-Nodejs Product</a>	hello.world.nodejp@jp.fujitsu.com	Hello-World-Nodejs Product	1	1	<a href="#">Delete</a>
<a href="#">Hello-World-Nodejs2 Developer</a>	hello.world.nodejp2@jp.fujitsu.com	Hello-World-Nodejs2 Developer	1	1	<a href="#">Delete</a>

Developer の情報を入力し、「Save」ボタンをクリックします。

API Management | Dashboard | APIs | Publish | Analytics | Admin | Organization\_Administr... | FUJITSU

Dashboard / Developers / New Developer | Organization ope | Environment | prod

### New Developer

Developer Details

First Name 太郎

Last Name 富士通

Email fj-taro@example.com

Username fj-taro

Custom Attributes

Name	Value	Actions
------	-------	---------

+ Custom Attribute

Cancel Save

ここをクリック!

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.03

### 5-3) アプリ (Developer App) の登録

Developer を割り当てる Developer App を作成します。

Developer App に Developer を割り当てることで、固有の client\_id, client\_secret が生成されます。

1つの Developer App に対しては、1人の Developer と1つ以上の Product を割り当てる必要があります。

画面上部の「Publish」メニューから「Developer Apps」をクリックして Developer Apps 画面に遷移します。

API Management | Dashboard | APIs | Publish | Analytics | Admin | Organization\_Administr... | FUJITSU

Dashboard / Developers | Organization ope | Environment | prod

### Developers

List Analytics

Products

Developers

Developer Apps

ここをクリック!

Search All

1-3 of 3 |< < > >| Export + Developer

Developer	Email	Username	Apps	Keys	Actions
handson-FN handson-LN	handson-developer@example.com	handson-developer	7	7	✕ Delete
Hello-World-Nodejs Product	hello.world.nodejp@jp.fujitsu.com	Hello-World-Nodejs Product	1	1	✕ Delete
Hello-World-Nodejs2 Developer	hello.world.nodejp2@jp.fujitsu.com	Hello-World-Nodejs2 Developer	1	1	✕ Delete

Dev Portal

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.03



「+ Developer App」 ボタンをクリックします。

API Management | Dashboard | APIs | Publish | Analytics | Admin | Organization\_Administr... | FUJITSU

Dashboard / Developer Apps | Organization open | Environment prod

### Developer Apps

List | Analytics

Search All | All | Pending | Revoked | Approved | 1-9 of 9 | + Developer App

App	Developer	App Family	Company	Key	Metrics for Last 24 Hours		Registered	Actions
					Traffic	Error Rate (%)		
sample-app	handson-FN handson-LN	default		1			May 16, 2016 1:41:40 PM	✕ Delete
weather-oauth2-app	handson-FN handson-LN	default		1			Apr 26, 2016 6:44:02 PM	✕ Delete
handson-api20160407001-app	handson-FN handson-LN	default		1			Apr 7, 2016 11:34:09 AM	✕ Delete
handson-api20160217019-app	handson-FN handson-LN	default		1			Mar 7, 2016 8:51:28 PM	✕ Delete
handson-api20160217010-app	handson-FN handson-LN	default		1			Feb 17, 2016 1:09:15 PM	✕ Delete
handson-api20160217017-app	handson-FN handson-LN	default		1			Feb 9, 2016 3:32:47 PM	✕ Delete
handson-api20160217018-app	handson-FN handson-LN	default		1			Feb 8, 2016 11:36:21 AM	✕ Delete
Hello-World-Nodejs2 App	Hello-World-Nodejs2 Developer	default		1			Feb 4, 2016 7:18:33 PM	✕ Delete
Hello-World-Nodejs App	Hello-World-Nodejs Product	default		1			Feb 4, 2016 3:05:43 PM	✕ Delete

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.03

Developer App の情報を入力し、「Save」 ボタンをクリックします。

#### ◇ Developer App Details

- Name : 申請内容の名前を入力
- Display Name : 申請内容の名前を入力 (自動入力)
- Developer : 「5-2) アプリ開発者 (Developer) の登録」で登録した Developer を選択
- Callback URL : 申請内容の Callback URL を入力

#### ◇ Products

「+ Product」 ボタンをクリックし、「5-1) API グルーピング (Product) の作成」で作成した API Product を選択

API Management | Dashboard | APIs | Publish | Analytics | Admin | Organization\_Administr... | FUJITSU

### Developer App Details

Name: weather-oauth2-app  
Display Name: weather-oauth2-app  
Developer: 太郎 富士通 (sj-taro@exampl...  
Callback URL: http://127.0.0.1:8080/oauth2-sample/clientApp-  
Notes:   
Products:   
Custom Attributes:   
Cancel Save

• Nameに「weather-oauth2-app」など申請された内容を入力します。  
• Display Nameは自動入力されます。  
• Developerに先ほど登録を行ったDeveloperを選択します。  
• 先ほど登録したAPI Productを選択します。  
• API Productの追加  
• 設定完了後Saveします。

ここをクリック!  
ここをクリック!

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.03

#### 5-4) client\_id, client\_secret の確認

Developer Apps 画面のアプリ一覧に、登録したアプリが追加されます。

追加されたアプリ名を選択し、client\_id, client\_secret を確認します。

- client\_id : Consumer Key 欄の「Show」をボタンクリックすると、client\_id が表示されます。
- client\_secret : Consumer secret 欄の「Show」ボタンをクリックすると、client\_secret が表示されます。

API 提供者は、表示された client\_id, client\_secret を API 利用者に通知します。

The screenshot shows the 'weather-oauth2-app' page in the API Management console. Under the 'Developer App Details' section, there are fields for Name, Display Name, Registered date, Developer, and Callback URL. Below this is a 'Products' table with the following data:

Product	Status	Consumer Key	Consumer Secret
<a href="#">weather-oauth2-product</a>	Approved	[Redacted]	[Redacted]

Red arrows point to the 'Show' buttons in the 'Consumer Key' and 'Consumer Secret' columns, with the text 'ここをクリック!' (Click here!). Callout boxes explain that clicking 'Show' displays the client\_id and client\_secret respectively.

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.03

## 6) ログインアプリの作成 (①)

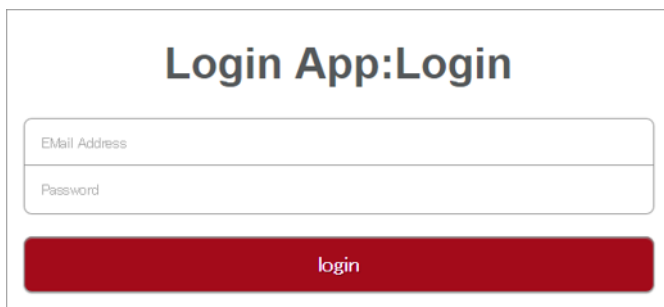
API 提供者が実施します。

### 6-1) 認証画面の作成

以下の仕様に基づいて、フローの ⑤, ⑥ で使用する認証画面を作成します。

- ユーザー名、パスワードの入力画面を表示
- ユーザー名およびパスワードのチェック機能を実装
- 認証成功時、許可リクエスト確認画面 (次項) へ遷移

#### 【画面サンプル】



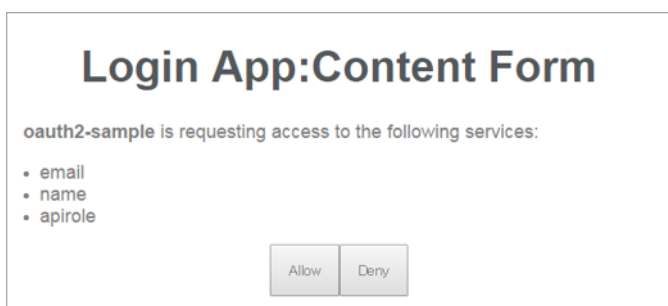
The screenshot shows a login form with the title "Login App:Login". It contains two input fields: "Email Address" and "Password". Below the fields is a prominent red button labeled "login".

### 6-2) 許可リクエスト確認画面の作成

以下の仕様に基づいて、フローの ⑥ で使用する許可リクエスト確認画面を作成します。

- リソースへのアクセス要求の許可/拒否を問い合わせる画面を表示
- アクセス要求が許可された場合、本サービス上の API を実行して Authorization Code の発行を要求 (⑦)
- レスポンスの Location ヘッダーの値へリダイレクトを実行 (⑧)

#### 【画面サンプル】



The screenshot shows an authorization screen with the title "Login App:Content Form". It contains the text "oauth2-sample is requesting access to the following services:" followed by a bulleted list: "email", "name", and "apirole". At the bottom, there are two buttons: "Allow" and "Deny".

Authorization Code の発行要求は、「2-3) authorizationcode API の作成」で作成した API を呼び出すことで行います。API の仕様は以下の通りです。

- HTTP メソッド：POST
- リクエスト情報
  - リソースパス： /oauth2/authorizationcode
  - 必須クエリパラメーター： client\_id, redirect\_uri, response\_type
  - 任意クエリパラメーター： state, redirect\_uri, scope, 任意情報

- レスポンス情報
  - ヘッダー：Location（ログインアプリで Location ヘッダーの値にリダイレクトを実行）
  - クエリパラメーター：Authorization code が付与される

【呼び出し例】

```
http://example.com:10080/oauth2/authorizationcode?client_id=QX8RDSpSZAhtND6CDBa32ZIrjC3GFkAI&state=mjy50qHgkl&redirect_uri=http%3A%2F%2Fexample.com%3A8080%2Foauth2-sample%2FclientApp-getaccesstoken.jsf&response_type=code&scope=name%20apirole&apirole=admin&id=00001
```

## 7) クライアントアプリの作成 (③)

API 利用者が実施します。

以下の仕様に基づいて、クライアントアプリを作成します。

### 7-1) client\_id チェック処理 (authorize API) の呼び出し (④)

「2-2) authorize API の作成」で作成した API を呼び出す処理を実装します。API の仕様は以下の通りです。

- HTTP メソッド：GET
- リクエスト情報
- リソースパス： /oauth2/authorize
  - 必須クエリパラメーター： client\_id
  - 任意クエリパラメーター： state, redirect\_uri, scope
    - ※ユニークキーを生成し、セッション等に保存しておく。
    - 生成したユニークキーは state パラメーターの値として付与を行う。
- レスポンス情報
  - ヘッダー：Location（本サービスで Location ヘッダーの値にリダイレクトを実行）

【呼び出し例】

```
http://expamle.com:10080/oauth2/authorize?client_id=QX8RDSpSZAhtND6CDBa32ZIrjC3GFkAI&state=mjy50qHgkl&scope=name%20apirole&redirect_uri=http%3A%2F%2Fexample.com%3A8080%2Foauth2-sample%2FclientApp-getaccesstoken.jsf
```

## 7-2) Access Token 発行処理 (accesstoken API) の呼び出し (㊟)

「2-4) accesstoken API の作成」で作成した API を呼び出す処理を実装します。

accesstoken API を呼び出す前に、state パラメーターのチェックを実行するようにします。API の仕様は以下の通りです。

- HTTP メソッド：POST
- リクエスト情報
  - リソースパス： /oauth2/accesstoken
  - 必須クエリパラメーター： code, grant\_type,
  - 任意クエリパラメーター： redirect\_uri
  - ヘッダー： Authorization (client\_id, client\_secret を使用した Basic 認証)
- レスポンス情報
  - Body：  
【例】

```
{
  "scope": "name apirole",
  "refresh_token_issued_at": "1462933753440",
  "expires_in": "59",
  "developer.email": "developer@jp.fujitsu.com",
  "refresh_token": "98y8CkI2qgQHax0aM3D04PrSDc4ROUVr",
  "client_id": "QX8RDSpSZAhtND6CDBa32ZIrjC3GFkAI",
  "access_token": "oWDHoLD3832ET7FmQnVjae7EGjh3",
  "refresh_token_expires_in": "0",
  "refresh_count": "0",
  "id": "00001",
  "apirole": "admin"
}
```

### 【呼び出し例】

```
http://example.com:10080/oauth2/accesstoken?code=VxZ7DWpM&grant_type=authorization_code&redirect_uri=http%3A%2F%2Fexample.com%3A8080%2Foauth2-sample%2FclientApp-getaccesstoken.jsf
```

### 7-3) Access Token 再発行処理 (refresh\_token API) の呼び出し

「2-5) refresh\_token API の作成」で作成した API を呼び出す処理を実装します。API の仕様は以下の通りです。

- HTTP メソッド : POST
- リクエスト情報
  - リソースパス : /oauth2/refresh\_accesstoken
  - 必須クエリパラメーター : grant\_type, refresh\_token
  - ヘッダー : Authorization (client\_id, client\_secret を使用した Basic 認証)
- レスポンス情報
  - Body :

```
【例】  
{  
  "scope": "name apirole",  
  "refresh_token_issued_at": "1462933753440",  
  "expires_in": "59",  
  "developer.email": "developer@jp.fujitsu.com",  
  "refresh_token": "98y8CkI2qgQHax0aM3D04PrSDc4ROUVr",  
  "client_id": "QX8RDSpSZAhtND6CDBa32ZIrjC3GFkAI",  
  "access_token": "oWDHoLD3832ET7FmQnVjae7EGjh3",  
  "refresh_token_expires_in": "0",  
  "refresh_count": "0",  
  "id": "00001",  
  "apirole": "admin"  
}
```

#### 【呼び出し例】

```
http://example.com:10080/oauth2/refresh_token?grant_type=refresh_token&refresh_token=J3ehTFknuHvBf3zfhAESAdBdcra5KUTF
```

#### 7-4) リソース要求処理

「3) API Proxy への Access Token 認証適用」で作成した API を呼び出す処理を実装します。API の仕様は以下の通りです。

※以下の「任意」と記載された項目は、3) で作成した API Proxy の内容を反映します。

- HTTP メソッド：任意
- リクエスト情報
  - リソースパス：任意
  - ヘッダー：Authorization (Bearer \$access\_token)
- レスポンス情報
  - Body：有効期限切れ

```
{
  "fault": {
    "detail": {
      "errorcode": "keymanagement.service.access_token_expired"
    },
    "faultstring": "Access Token expired"
  }
}
```

## 2. API マッシュアップ

本サービスでは、1 つの API Proxy で 複数のバックエンドサービスを呼び出すことができます。

呼び出し先として 1 つのバックエンドサービスを複数呼び出したり、異なるバックエンドサービスを呼び出したり、目的に応じて呼び出し先を変えることができます。

また、リクエストやレスポンスから必要な情報だけを抽出して一時保存することができ、一時保存した情報を他のバックエンドサービスへのリクエストやクライアントへのレスポンスに使用することができます。

このように、複数の処理を束ねた新たな API を作成することができます。

### 2.1. API マッシュアップ

#### 2.1.1. ユースケース概要

ここでは、API マッシュアップの 2 つのユースケースを説明します。

【ユースケース 1：バックエンドサービスの実行結果の結合】

1 つのバックエンドサービス(API)を 2 回呼び出し、その結果をまとめてクライアントに返却する API Proxy を実装します。

バックエンドサービスとして、[天気情報を取得する API](#) を使用します。

API にアクセスする際の URL に都市コードを指定すると、指定された都市の天気情報を取得します。

API を 2 回実行し、実行結果をまとめてクライアントへ返却します。

レスポンスデータ（天気情報）は JSON 形式で返却されます。

API Proxy の実装フローは以下の通りです。

- 1) API Proxy の作成
  - 1-1) API Proxy の作成
  - 1-2) Conditional Flow (GET) の作成
- 2) Service Callout ポリシーの作成（天気情報の取得）
- 3) Extract Variables ポリシーの作成 1（天気情報の変数化 1）
- 4) Extract Variables ポリシーの作成 2（天気情報の変数化 2）
- 5) Assign Message ポリシーの作成（HTTP レスポンスの作成）
- 6) ポリシーの配置
- 7) 動作確認

使用する Policy は以下の通りです。

- Extract Variables ポリシー
- Service Callout ポリシー



【ユースケース 2：バックエンドサービスの実行結果をクエリパラメーターとして付与したバックエンドサービスの呼び出し】

ユースケース 1 で作成した API Proxy を利用します。

取得した天気情報から都道府県名の情報を抽出し、都道府県名から市区町村リストを取得する別のバックエンドサービスを呼び出す際に利用するよう変更します。

取得した情報（天気情報・市区町村リスト）はまとめてクライアントに返却します。

API Proxy の実装フローは以下の通りです。

- 1) Extract Variables ポリシーの修正（都道府県名の変数化）
- 2) Service Callout ポリシーの作成（市区町村リストの取得）
- 3) Extract Variables ポリシーの作成（市区町村リストの変数化）
- 4) Assign Message ポリシーの修正（HTTP レスポンスの修正）
- 5) ポリシーの配置
- 6) 動作確認

使用する Policy は以下の通りです。

- Assign Message ポリシー
- Extract Variables ポリシー
- Service Callout ポリシー

## 2.1.2. 手順

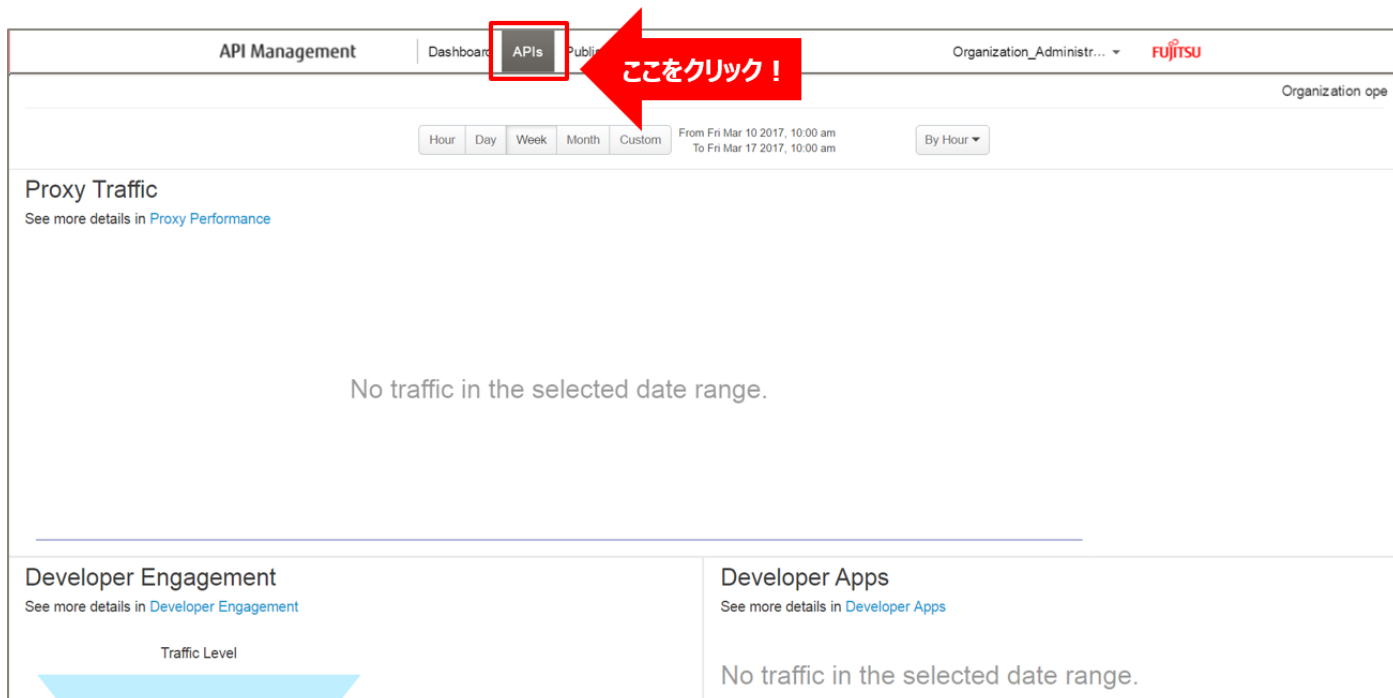
### 2.1.2.1. バックエンドサービスの実行結果の結合

#### 1) API Proxy の作成

API Proxy を作成します。

#### 1-1) API Proxy の作成

画面上部の「APIs」メニューをクリックし、API Proxies 画面に遷移します。



API Proxies 画面で、「+ API Proxy」ボタンをクリックします。

The screenshot shows the 'API Proxies' page in the API Management console. The page has a navigation bar with 'Dashboard', 'APIs', 'Publish', 'Analytics', and 'Admin'. The 'APIs' tab is selected. The page title is 'API Proxies' and there is a search bar. A table lists existing API proxies with columns for API Proxy, Environments, Traffic, Message Trend by Hour, Avg Time, Error Rate, Modified, and Actions. A red arrow points to the '+ API Proxy' button in the top right corner.

API Proxy	Environments	Traffic	Message Trend by Hour	Avg Time	Error Rate	Modified	Actions
handson-api20160217020	test	1	↓	8029.00 ms	100.00 %	3 hours ago	× Delete Roles (1)
Hello-World-Nodejs-2	test, prod	3	↓	37.00 ms	0.00 %	19 hours ago	× Delete Roles (1)
Hello-World-Nodejs	test, prod	0				a day ago	× Delete Roles

➤ Build a Proxy (TYPE)

「Reverse proxy (most common)」を選択し、「Next」ボタンをクリックします。

The screenshot shows the 'Build a Proxy' form. The 'Reverse proxy (most common)' option is selected and highlighted with a red box. Below it are other options: 'Node.js App', 'SOAP service', 'No Target', and 'Proxy bundle'. At the bottom right, there is an 'Exit Without Saving' button and a 'Next' button. A red arrow points to the 'Next' button.

➤ Build a Proxy (DETAILS)

API Proxy の情報を入力し、「Next」ボタンをクリックします。以下は入力例です。

- Proxy Name : mashup-sample (任意の名前)
- Proxy Base Path : /mashup-sample (任意のパス (自動入力))
- Existing API : http://weather.livedoor.com (任意の API)

### Build a Proxy

TYPE > **DETAILS** > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Specify the proxy details.

Proxy Name \*   
Valid characters are letters, numbers, dash (-), and underscore (\_).

Proxy Base Path \*   
A path component that uniquely identifies this API proxy. The public-facing URL of this API proxy is comprised of your organization name, an environment where this API proxy is deployed, and this Proxy Base Path. Example URL http://test-sandbox-test.apigee.net/mashup-sample

Existing API \*   
Defines the target URL invoked on behalf of this API proxy. Any URL that is accessible over the open Internet can be used. Example: https://api.usergrid.com

Description

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

**ここをクリック!**

➤ Build a Proxy (SECURITY)

Authentication : 「Pass through (none)」 を選択し、「Next」 ボタンをクリックします。

### Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Secure access for users and clients.

Authorization  Pass through (none)  
 API Key  
 OAuth 2.0

Browser  Add CORS headers

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

[Previous](#) [Exit Without Saving](#) **ここをクリック!** [Next](#)

➤ Build a Proxy (VIRTUAL HOSTS)

設定を変えずに「Next」ボタンをクリックします。

Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Select the virtual hosts this proxy will bind to when it is deployed. You must select at least one virtual host. [Learn more...](#)

<input checked="" type="checkbox"/> Name	Environment	Host Aliases
<input checked="" type="checkbox"/> default	prod	http://:10080
	test	http://:10080
<input checked="" type="checkbox"/> secure	prod	https://:10443
	test	https://:10443

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Next

➤ Build a Proxy (BUILD)

設定を変えずに「Build and Deploy」ボタンをクリックします。

Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

You are ready to build and deploy your API proxy.

Deploy Environments  prod  test

Proxy Name apikey-auth

Proxy Type Reverse proxy

Virtual Hosts default, secure

Security None

Browser Do not allow direct requests from a browser via CORS

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Build and Deploy

➤ Build a Proxy (SUMMARY)

API Proxy の作成が完了したら、API Proxy のリンクをクリックします。

Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

- ✓ Generated proxy
- ✓ Uploaded proxy
- ✓ Deployed to test

ここをクリック！ view **mashup-sample** proxy in the editor .

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

## 1-2) Conditional Flow (GET) の作成

バックエンドサービスに対するリソースパスと処理 (HTTP Method) の定義を行います。

クライアントからのリクエストがここで定義したパターンと一致する場合に、以降の手順で設定する処理を実行します。

「DEVELOP」タブをクリックし、API proxy editor を開きます。

The screenshot shows the API Management console for the 'mashup-sample' proxy. The 'DEVELOP' tab is highlighted with a red box and a red arrow pointing to it with the text 'ここをクリック!' (Click here!). The console displays the 'Revision 1 Summary', 'Deployments' table, 'Proxy Endpoints' table, and 'Target Endpoints' table.

Environment	Revision	Status	URL
test	1	<span style="color: green;">●</span>	http://... [+]

Name	Base Path	Target Endpoints
default	/mashup-sample	default

Name	Target	Used by Proxy Endpoints
default	URL http://weather.livedoor.com	default

default の「+」ボタンをクリックし、New Conditional Flow を開きます。

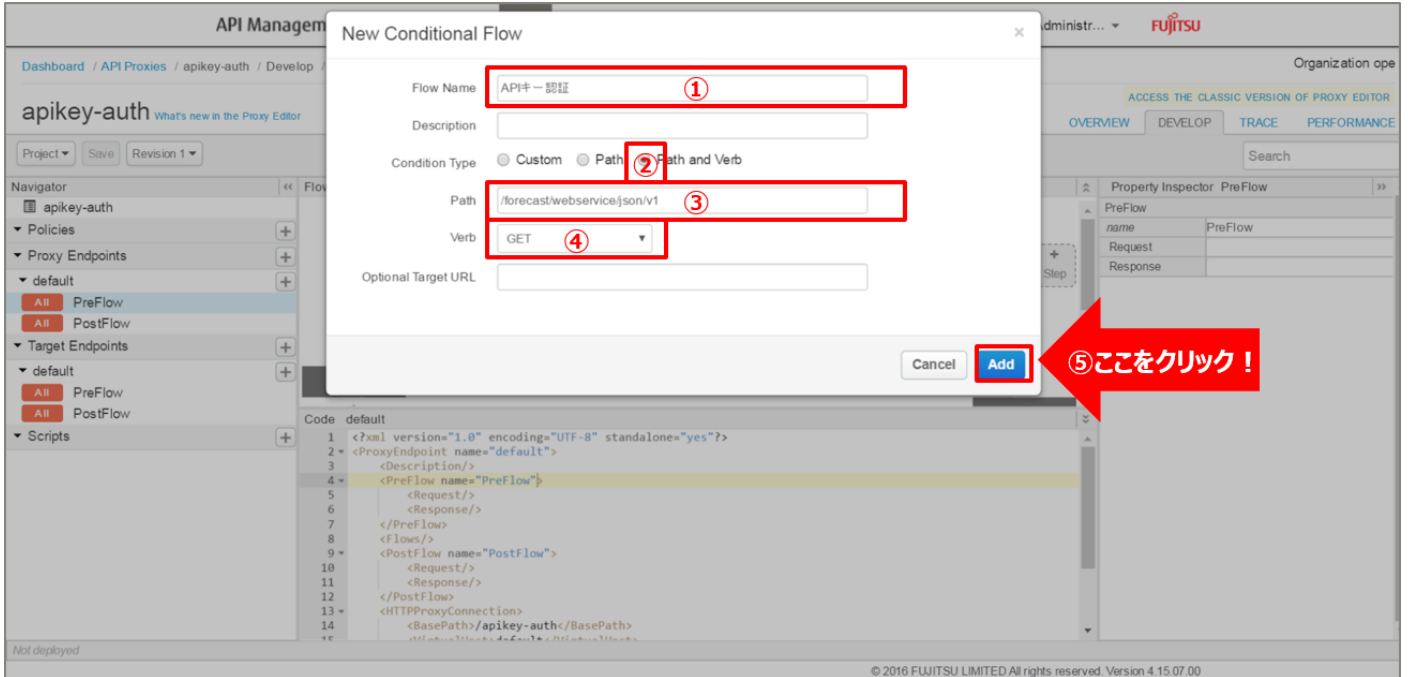
The screenshot shows the API Management console for the 'apikey-auth' proxy. The 'default' proxy endpoint is selected in the 'Proxy Endpoints' list, and a red box highlights the '+' button next to it with a red arrow pointing to it and the text 'ここをクリック!' (Click here!). The 'New Conditional Flow' dialog box is open, showing a 'Request' step in a flow diagram and a code editor with XML configuration.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ProxyEndpoint name="default">
3   <Description/>
4   <PreFlow name="PreFlow">
5     <Request/>
6     <Response/>
7   </PreFlow>
8   <Flows/>
9   <PostFlow name="PostFlow">
10    <Request/>
11    <Response/>
12  </PostFlow>
13  <HTTPProxyConnection>
14    <BasePath>/apikey-auth</BasePath>
```



Flow の情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

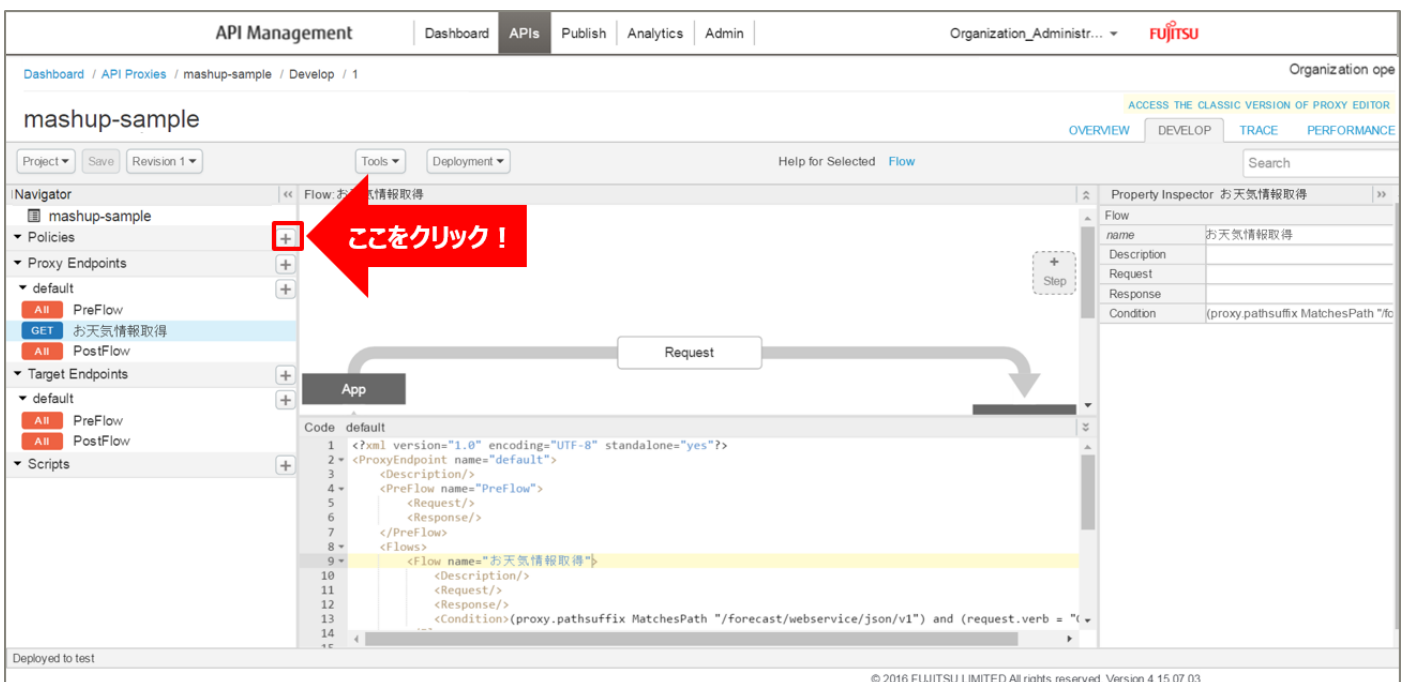
- Flow Name : お天気情報取得 (任意の名前)
- Condition Type : 「Path and Verb」を選択
- Path : /forecast/webservice/json/v1 (任意のパス)
- Verb : 「GET」を選択



## 2) Service Callout ポリシーの作成 (天気情報の取得)

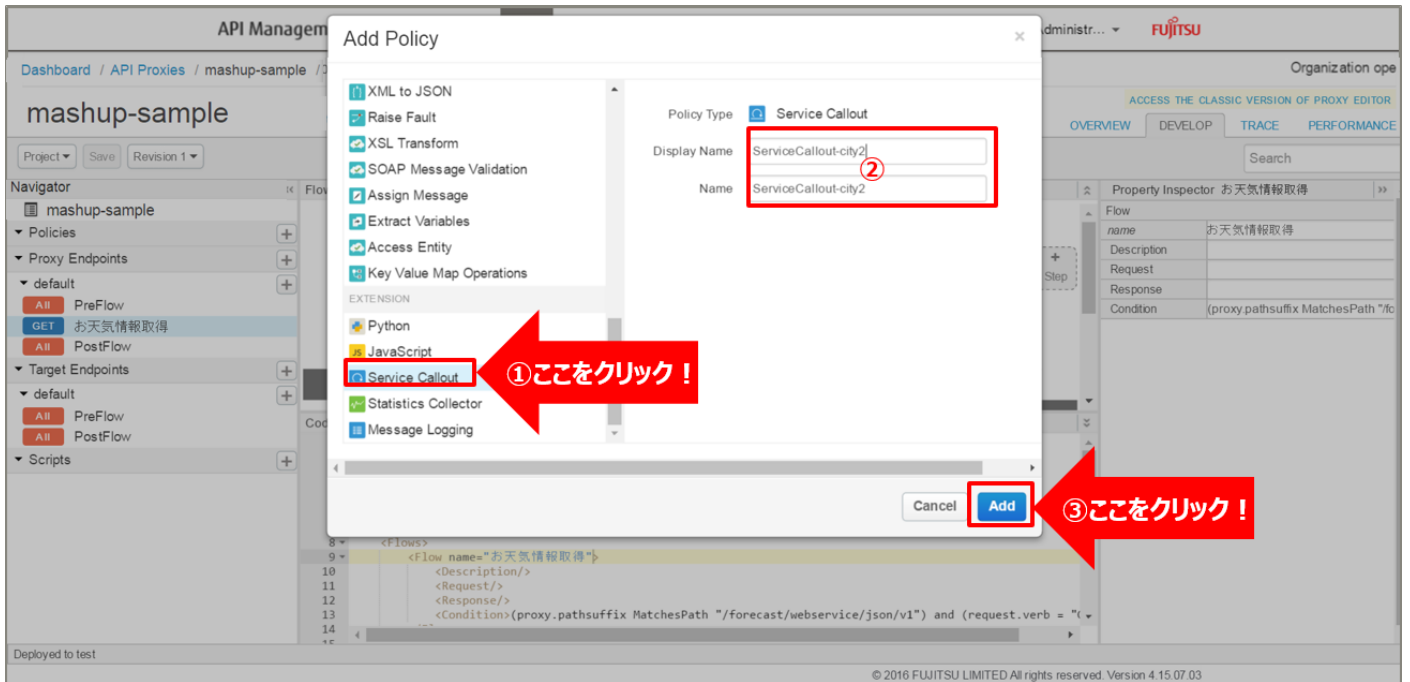
指定された都市の天気情報を取得する API を呼び出す Service Callout ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Service Callout」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。  
 以下は入力例です。

- Display Name : ServiceCallout-city2 (任意の名前)
- Name : ServiceCallout-city2 (任意の名前)



追加した Service Callout ポリシーを選択し、ポリシー編集画面を表示します。  
 必要に応じてポリシーの定義を編集します。



【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceCallout async="false" continueOnError="false" enabled="true" name="ServiceCallout-
city2">
  <DisplayName>ServiceCallout-city2</DisplayName>
  <Properties/>
  <Request clearPayload="true" variable="ServiceCallout-city2">
    <IgnoreUnresolvedVariables>>false</IgnoreUnresolvedVariables>
    <Set>
      <QueryParams>
        <QueryParam name="city">{request.queryparam.city2}</QueryParam>
      </QueryParams>
      <Verb>GET</Verb>
    </Set>
  </Request>
  <Response>calloutResponse-city2</Response>
  <HTTPTargetConnection>
    <Properties/>
    <URL>http://weather.livedoor.com/forecast/webservice/json/v1</URL>
  </HTTPTargetConnection>
</ServiceCallout>
```

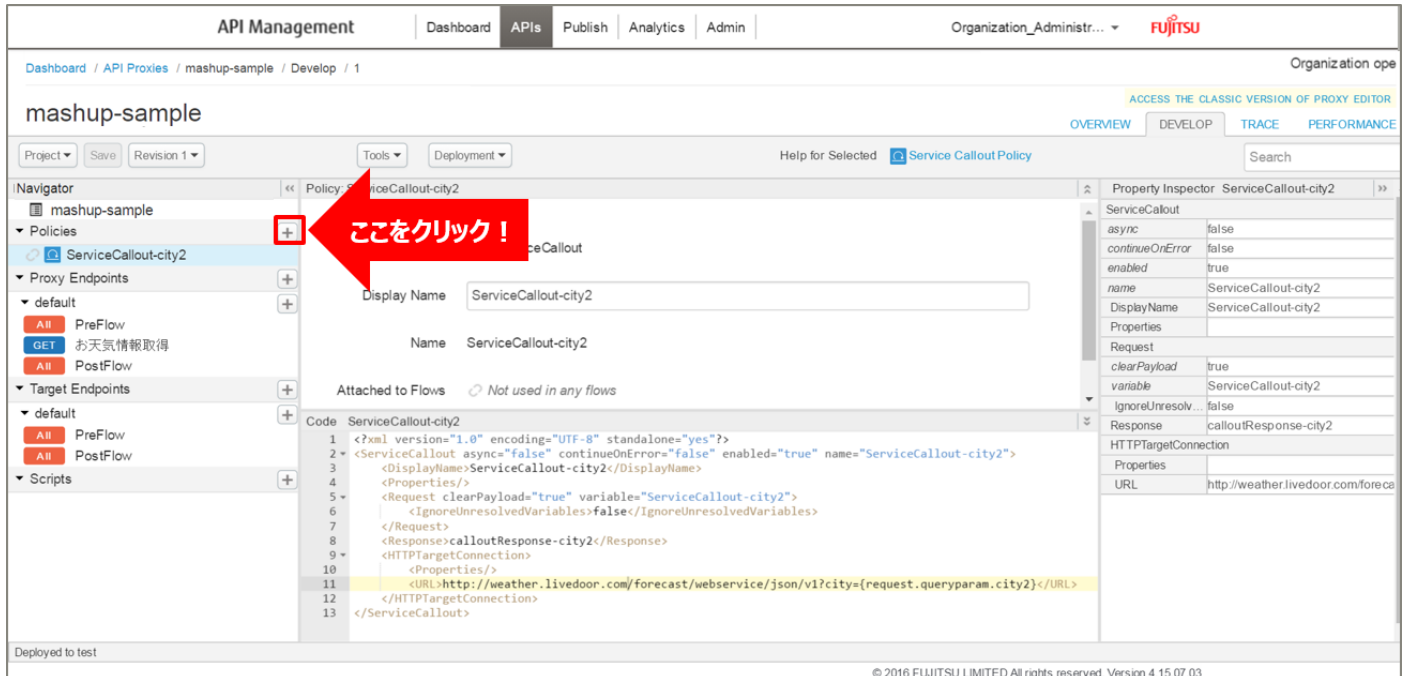
※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A3.1. Service Callout XML 仕様](#)」をご参照ください。

### 3) Extract Variables ポリシーの作成 1 (天気情報の変数化 1)

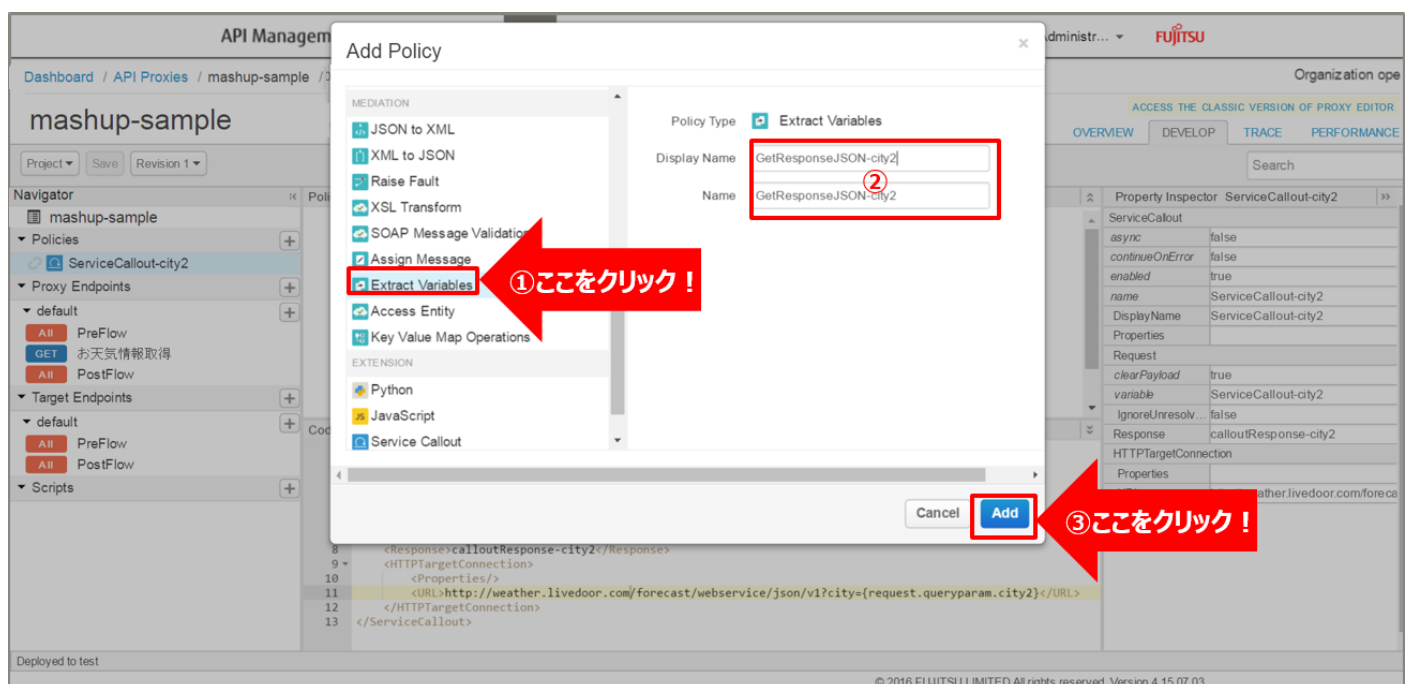
Service Callout ポリシーで取得した天気情報を変数として格納(変数化)する Extract Variables ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Extract Variables」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : GetResponseJSON-city2 (任意の名前)
- Name : GetResponseJSON-city2 (任意の名前)



追加した Extract Variables ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

#### 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true"
name="GetResponseJSON-city2">
  <DisplayName>GetResponseJSON-city2</DisplayName>
  <JSONPayload>
    <Variable name="object">
      <JSONPath>$/JSONPath</JSONPath>
    </Variable>
  </JSONPayload>
  <Source>calloutResponse-city2</Source>
  <VariablePrefix>calloutresponse-city2</VariablePrefix>
</ExtractVariables>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.4. Extract Variables XML 仕様](#)」をご参照ください。

#### 4) Extract Variables ポリシーの作成 2 (天気情報の変数化 2)

3) と同様の手順で、エンドポイントのレスポンスを変数として格納 (変数化) する Extract Variables ポリシーを作成します。

以下は Add Policy で入力するポリシーの情報の入力例です。

- Display Name : GetResponseJSON-city (任意の名前)
- Name : GetResponseJSON-city (任意の名前)

追加した Extract Variables ポリシーを選択し、ポリシー編集画面を表示します。

必要に応じてポリシーの定義を編集します。

**GetResponseJSON-city.xml**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true" name="GetResponseJSON-city">
  <DisplayName>GetResponseJSON-city</DisplayName>
  <JSONPayload>
    <Variable name="object">
      <JSONPath>${}</JSONPath>
    </Variable>
  </JSONPayload>
  <Source>response</Source>
  <VariablePrefix>response-city</VariablePrefix>
</ExtractVariables>
```

<JSONPayload>  
<Variable name="object">  
 <JSONPath>\${}</JSONPath>  
</Variable>  
</JSONPayload>

JSON全体を変数へ格納。

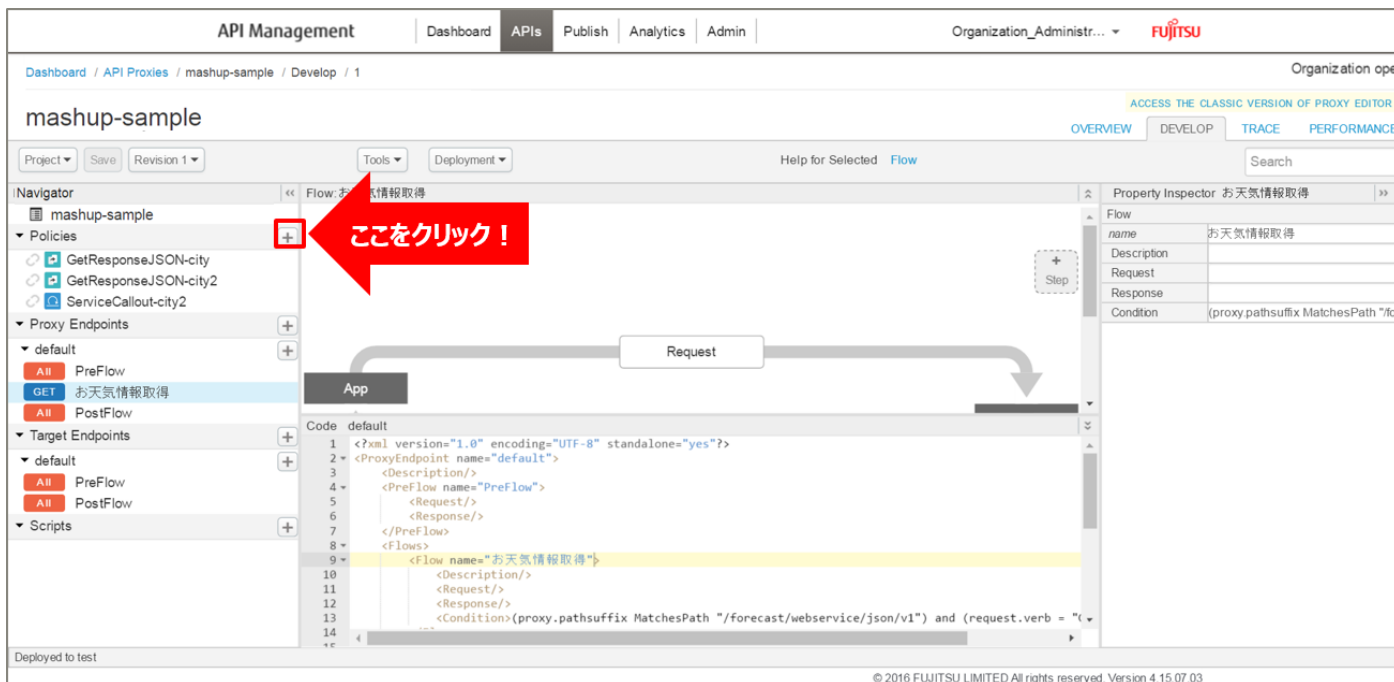
#### 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true"
name="GetResponseJSON-city">
  <DisplayName>GetResponseJSON-city</DisplayName>
  <JSONPayload>
    <Variable name="object">
      <JSONPath>${}</JSONPath>
    </Variable>
  </JSONPayload>
  <Source>response</Source>
  <VariablePrefix>response-city</VariablePrefix>
</ExtractVariables>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

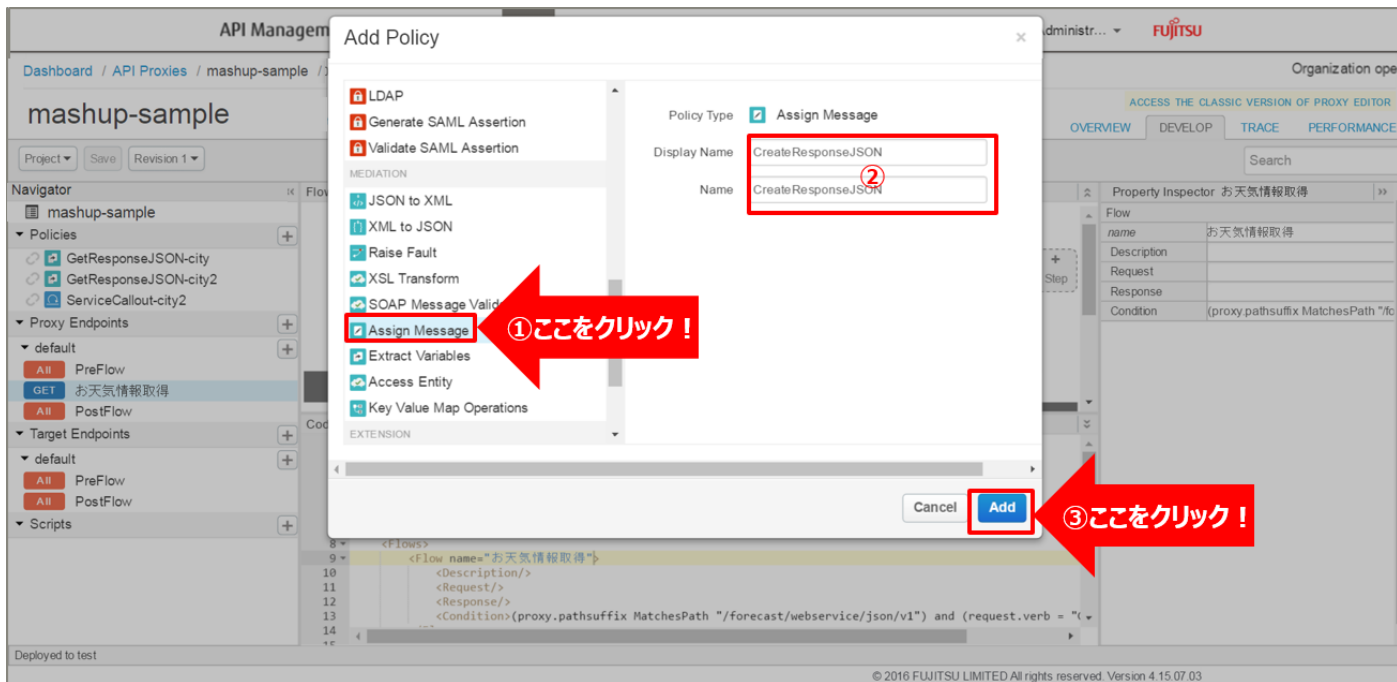
※定義内容の詳細は、「[A2.4. Extract Variables XML 仕様](#)」をご参照ください。

5) Assign Message ポリシーの作成 (HTTP レスポンスの作成)  
Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Assign Message」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。  
以下は入力例です。

- Display Name : CreateResponseJSON (任意の名前)
- Name : CreateResponseJSON (任意の名前)





追加した Assign Message ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

**ここをクリック!**

**2つのJSONをJSONArrayとする**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true" name="CreateResponseJSON">
  <DisplayName>CreateResponseJSON</DisplayName>
  <Copy source="response">
    <StatusCode>true</StatusCode>
  </Copy>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="response"/>
  <Set>
    <Payload contentType="application/json;charset=utf-8" variablePrefix="@ " variableSuffix="#">
      [@response-city.object#,@calloutresponse-city2.object#]
    </Payload>
  </Set>
</AssignMessage>
```

#### 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true"
name="CreateResponseJSON">
  <DisplayName>CreateResponseJSON</DisplayName>
  <Copy source="response">
    <StatusCode>true</StatusCode>
  </Copy>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="response"/>
  <Set>
    <Payload contentType="application/json;charset=utf-8" variablePrefix="@ "
variableSuffix="#">
      [@response-city.object#,@calloutresponse-city2.object#]
    </Payload>
  </Set>
</AssignMessage>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.3. Assign Message XML 仕様](#)」をご参照ください。



## 6) ポリシーの配置

以下の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。

The screenshot shows the API Management console for a project named 'mashup-sample'. The main area displays a flow diagram with a 'Request' step and a 'Response' step. The flow is connected to a 'Server' component. The Navigator on the left shows the following structure:

- mashup-sample
  - Policies
    - CreateResponseJSON (3)
    - GetResponseJSON-city (3)
    - GetResponseJSON-city2 (2)
    - ServiceCallout-city2 (2)
  - Proxy Endpoints
    - default
      - PreFlow
      - GET お天気情報取得 (selected)
      - PostFlow
  - Target Endpoints
    - default
      - PreFlow
      - PostFlow
  - Scripts

The flow diagram shows the following steps:

- Request
  - ServiceCallout-city2
  - GetResponseJSON-city2
- Response
  - CreateResponseJSON
  - GetResponseJSON-city

The Property Inspector on the right shows the configuration for the selected step:

Property Inspector お天気情報取得	
Flow	
name	お天気情報取得
Description	
Request	
Step	
Name	ServiceCallout-city2
Step	
Name	GetResponseJSON-city2
Response	
Step	
Name	GetResponseJSON-city
Step	
Name	CreateResponseJSON
Condition	(proxy.pathsuffix MatchesPath "fo

Red annotations in the image include:

- ④ここをクリック! (Click here!) pointing to the 'Save' button.
- ②ここにドラッグ! (Drag here!) pointing to the 'ServiceCallout-city2' and 'GetResponseJSON-city2' policies.
- ③ここにドラッグ! (Drag here!) pointing to the 'CreateResponseJSON' and 'GetResponseJSON-city' policies.
- ①ここをクリック! (Click here!) pointing to the 'GET お天気情報取得' endpoint in the Navigator.

## 7) 動作確認

作成した API Proxy の動作確認を行います。

以下の例のように、URL へアクセスすると JSON 形式のデータが返却されます。

http://{FQDN}:10080/mashup-

sample/forecast/webservice/json/v1?city={CityCode}&city2={CityCode2}

※http プロトコルでの動作確認例です（ポート番号は 10080 です）。

※{FQDN}: API Proxy の作成時に生成された URL のホスト名（FQDN）を指定します。

※{CityCode}, {CityCode2} は、任意の都市コードに置き換えてください。（例：400040, 410020 など）

```
[ - [
  - pinpointLocations: [
    - [
      link: "http://weather.livedoor.com/area/forecast/4020200",
      name: "大牟田市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4020300",
      name: "久留米市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4020700",
      name: "柳川市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4021000",
      name: "八女市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4021100",
      name: "筑後市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4021200",
      name: "大川市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4021600",
      name: "小郡市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4022500",
      name: "うきは市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4022800",
      name: "朝倉市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4022900",
      name: "みやま市"
    ],
    - [
      link: "http://weather.livedoor.com/area/forecast/4044700"
    ]
  ]
]
```

### 2.1.2.2. バックエンドサービスの実行結果をクエリパラメーターとして付与したバックエンドサービスの呼び出し

「2.1.2.1 バックエンドサービスの実行結果の結合」で作成した API Proxy を利用します。

「APIs」メニューをクリックし、API Proxies の一覧から、編集する API Proxy を選択します。

# 1) Extract Variables ポリシーの修正（都道府県名の変数化）

「2.1.2.1.3) Extract Variables ポリシーの作成 1（天気情報の変数化 1）」で作成した Extract Variables ポリシーを選択し、ポリシー編集画面を表示します。

取得したレスポンスから都道府県名の情報を抽出して変数化する処理を追加します。



## 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true"
name="GetResponseJSON-city2">
  <DisplayName>GetResponseJSON-city2</DisplayName>
  <JSONPayload>
    <Variable name="object">
      <JSONPath>$</JSONPath>
    </Variable>
    <Variable name="prefecture">
      <JSONPath>$.location.prefecture</JSONPath>
    </Variable>
  </JSONPayload>
  <Source>calloutResponse-city2</Source>
  <VariablePrefix>calloutresponse-city2</VariablePrefix>
</ExtractVariables>
```

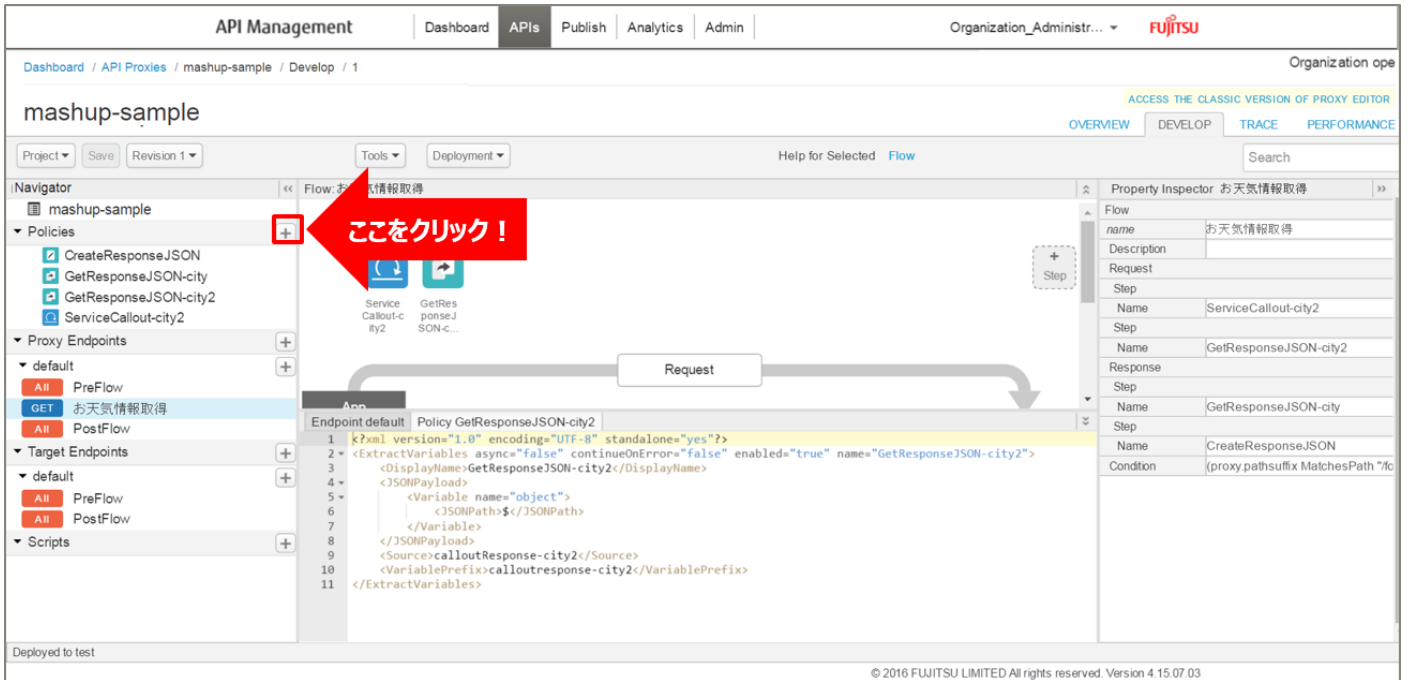
※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.4. Extract Variables XML 仕様](#)」をご参照ください。

## 2) Service Callout ポリシーの作成（市区町村リストの取得）

指定された都道府県の市区町村リスト取得する API を呼び出す Service Callout ポリシーを追加します。

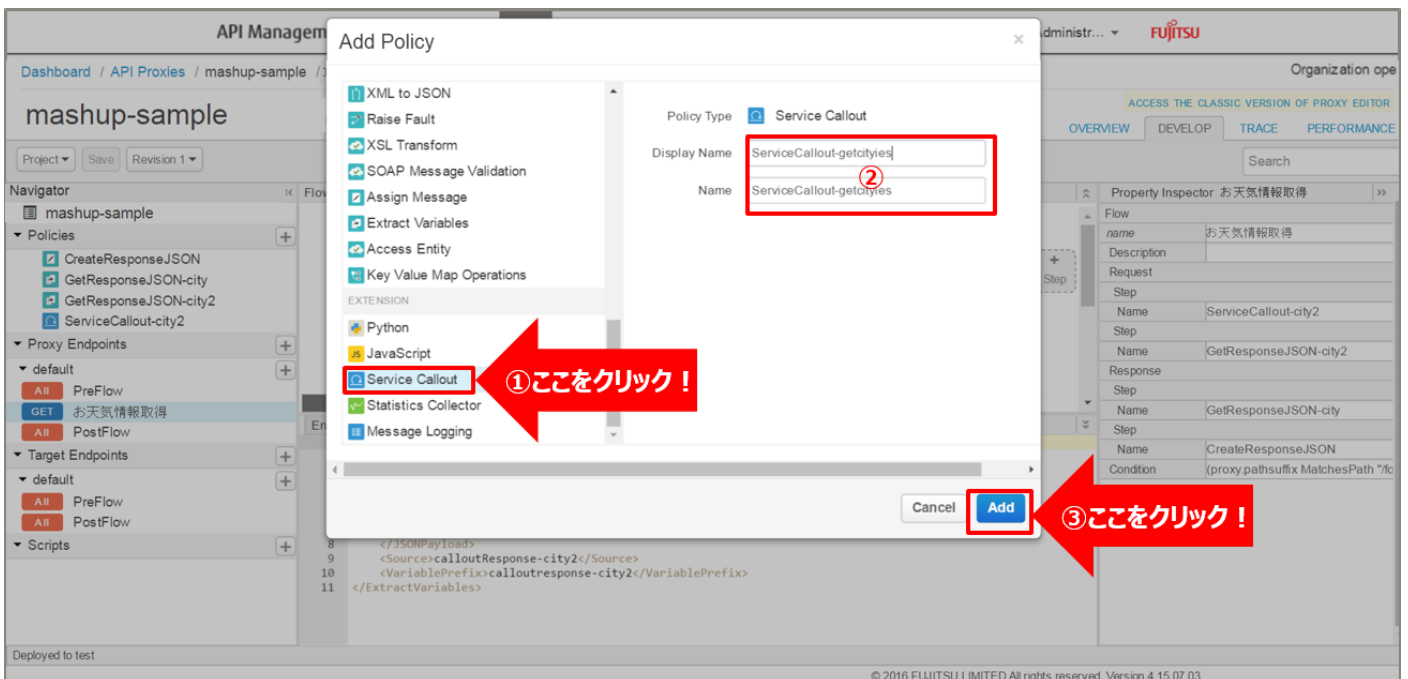
Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Service Callout」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。

以下は入力例です。

- Display Name : ServiceCallout-getcities（任意の名前）
- Name : ServiceCallout-getcities（任意の名前）



追加した Service Callout ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

**ServiceCallout-getcities.xml**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceCallout async="false" continueOnError="false" enabled="true" name="ServiceCallout-getcities">
  <DisplayName>ServiceCallout-getcities</DisplayName>
  <Properties/>
  <Request clearPayload="true" variable="ServiceCallout-getcities">
    <IgnoreUnresolvedVariables>false</IgnoreUnresolvedVariables>
  </Request>
  <Set>
    <QueryParams>
      <QueryParam name="method">getCities</QueryParam>
      <QueryParam name="prefecture">{calloutresponse-city2.prefecture}</QueryParam>
    </QueryParams>
    <Verb>GET</Verb>
  </Set>
</Request>
<Response>calloutResponse-getcities</Response>
<HTTPTargetConnection>
  <Properties/>
  <URL>http://geoapi.heartrails.com/api/json</URL>
</HTTPTargetConnection>
</ServiceCallout>
```

レスポンス結果をクエリパラメータへ指定

ここをクリック!

Response	calloutResponse
HTTPTargetConnection	
Properties	
URL	http://example.com

Code

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ServiceCallout async="false" continueOnError="false" enabled="true" name="ServiceCallout-getcities">
3   <DisplayName>ServiceCallout-getcities</DisplayName>
4   <Properties/>
5   <Request clearPayload="true" variable="myRequest">
6     <IgnoreUnresolvedVariables>false</IgnoreUnresolvedVariables>
7   </Request>
8   <Response>calloutResponse</Response>
9   <HTTPTargetConnection>
10    <Properties/>
11    <URL>http://example.com</URL>
12  </HTTPTargetConnection>
13 </ServiceCallout>
```

Deployed to test

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.03

【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceCallout async="false" continueOnError="false" enabled="true" name="ServiceCallout-
getcities">
  <DisplayName>ServiceCallout-getcities</DisplayName>
  <Properties/>
  <Request clearPayload="true" variable="ServiceCallout-getcities">
    <IgnoreUnresolvedVariables>>false</IgnoreUnresolvedVariables>
    <Set>
      <QueryParams>
        <QueryParam name="method">getCities</QueryParam>
        <QueryParam name="prefecture">{calloutresponse-
city2.prefecture}</QueryParam>
      </QueryParams>
      <Verb>GET</Verb>
    </Set>
  </Request>
  <Response>calloutResponse-getcities</Response>
  <HTTPTargetConnection>
    <Properties/>
    <URL>http://geoapi.heartrails.com/api/json</URL>
  </HTTPTargetConnection>
</ServiceCallout>
```

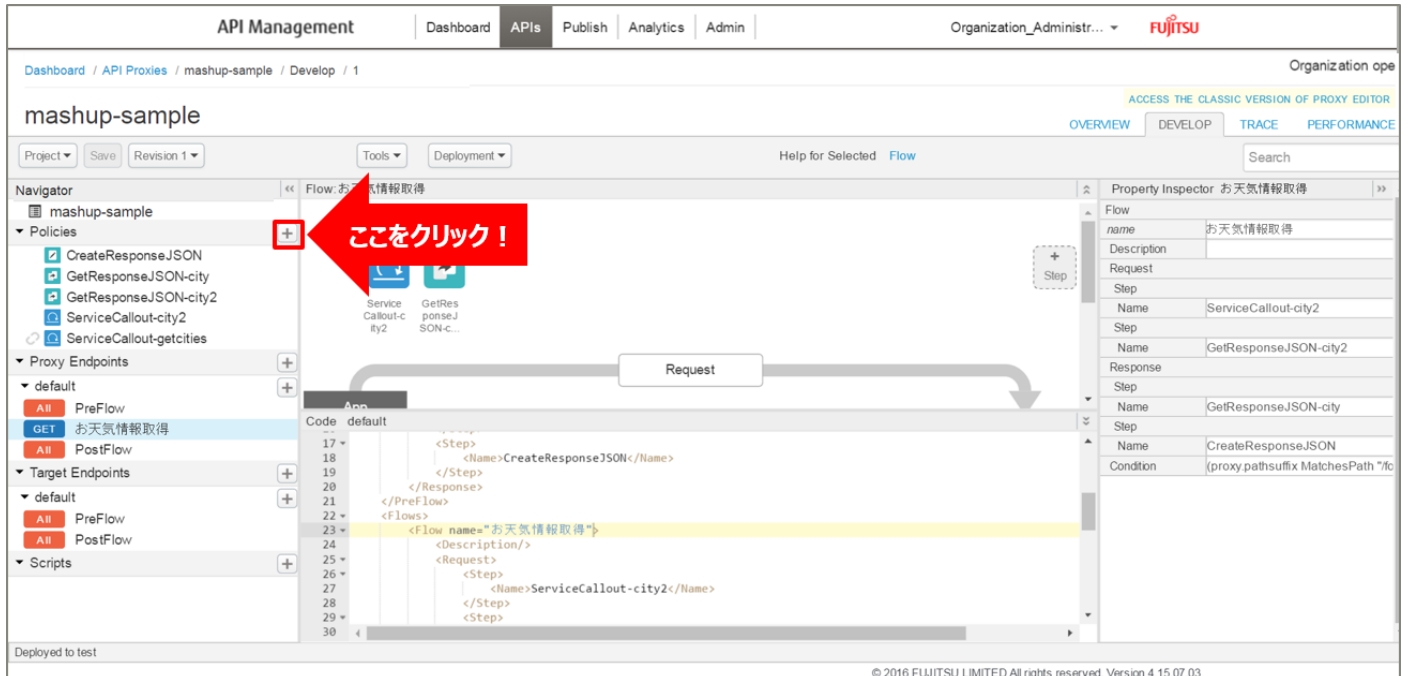
※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A3.1. Service Callout XML 仕様](#)」をご参照ください。

### 3) Extract Variables ポリシーの作成（市区町村リストの変数化）

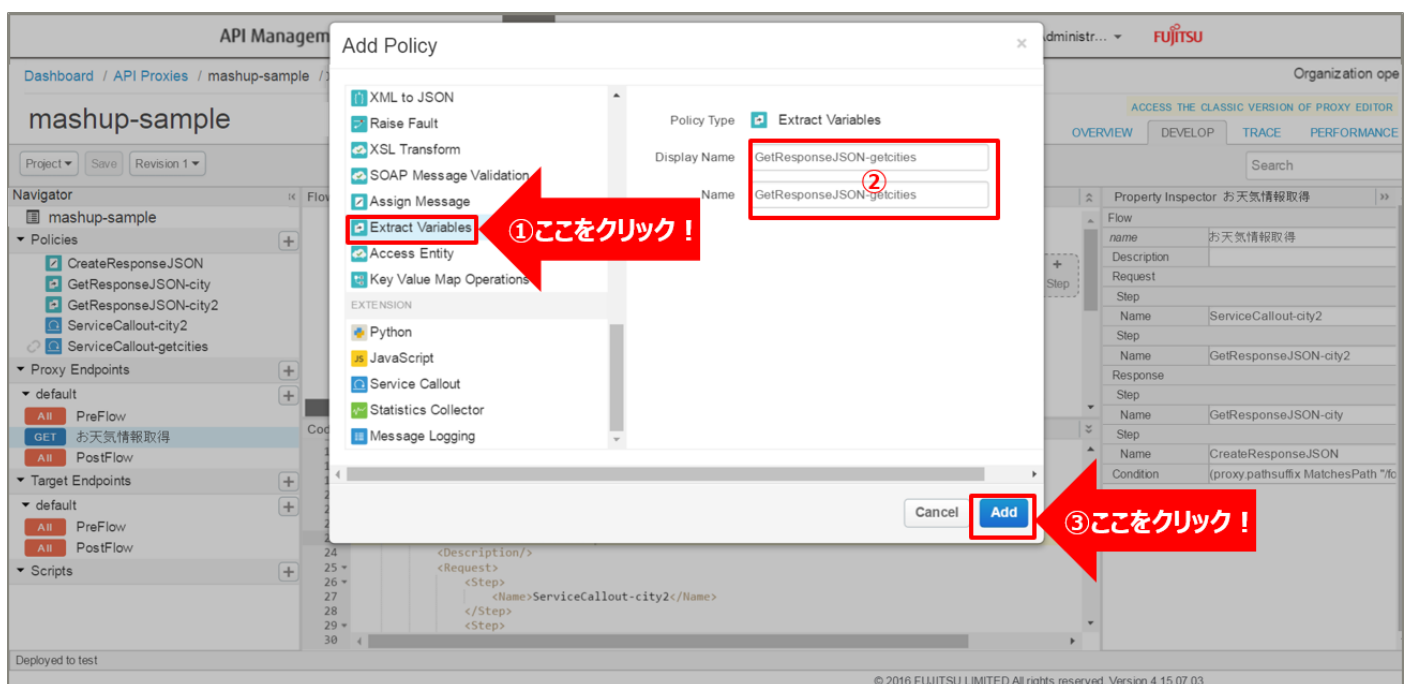
Service Callout ポリシーで取得した市区町村リストを変数として格納（変数化）する Extract Variables ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Extract Variables」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : GetResponseJSON-getcities（任意の名前）
- Name : GetResponseJSON-getcities（任意の名前）



追加した Extract Variables ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

The screenshot shows the API Management interface for a project named 'mashup-sample'. The left sidebar shows a tree view of policies, with 'GetResponseJSON-getcities' selected. A red arrow points to this policy with the text 'ここをクリック!'. The main area displays the XML definition for the 'GetResponseJSON-getcities.xml' policy. A red box highlights the XML code, and a grey box on the right contains the text 'JSON全体を変数へ格納。'. The XML code is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true" name="GetResponseJSON-getcities">
  <DisplayName>GetResponseJSON-getcities</DisplayName>
  <JSONPayload>
    <Variable name="object">
      <JSONPath>${}/JSONPath</JSONPath>
    </Variable>
  </JSONPayload>
  <Source>calloutResponse-getcities</Source>
  <VariablePrefix>calloutresponse-getcities</VariablePrefix>
</ExtractVariables>
```

#### 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true"
name="GetResponseJSON-getcities">
  <DisplayName>GetResponseJSON-getcities</DisplayName>
  <JSONPayload>
    <Variable name="object">
      <JSONPath>${}/JSONPath</JSONPath>
    </Variable>
  </JSONPayload>
  <Source>calloutResponse-getcities</Source>
  <VariablePrefix>calloutresponse-getcities</VariablePrefix>
</ExtractVariables>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.4. Extract Variables XML 仕様](#)」をご参照ください。



#### 4) Assign Message ポリシーの修正 (HTTP レスポンスの修正)

「Assign Message ポリシーの作成 (HTTP レスポンスの作成)」で作成した Assign Message ポリシーを選択し、ポリシー編集画面を表示します。

HTTP メッセージに市区町村リストの情報を追加します。

**ここをクリック!**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true" name="CreateResponseJSON">
  <DisplayName>CreateResponseJSON</DisplayName>
  <Copy source="response">
    <StatusCode>true</StatusCode>
  </Copy>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="response"/>
  <Set>
    <Payload contentType="application/json;charset=utf-8" variablePrefix="@ " variableSuffix="#">
      [ @response-city.object#, @calloutresponse-city2.object#, @calloutresponse-getcities.object# ]
    </Payload>
  </Set>
</AssignMessage>
```

**3つのJSONをJSONArrayとする**

AssignTo	
createNew	false
transport	http
type	response
Set	
Payload	[ @response-city.object#, @
contentType	application/json;charset=utf-8
variablePrefix	@
variableSuffix	#

Code: CreateResponseJSON

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <AssignMessage async="false" continueOnError="false" enabled="true" name="CreateResponseJSON">
3   <DisplayName>CreateResponseJSON</DisplayName>
4   <Copy source="response">
5     <StatusCode>true</StatusCode>
6   </Copy>
7   <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
8   <AssignTo createNew="false" transport="http" type="response"/>
9   <Set>
10    <Payload contentType="application/json;charset=utf-8" variablePrefix="@ " variableSuffix="#">
11      [ @response-city.object#, @calloutresponse-city2.object# ]
12    </Payload>
13  </Set>
14 </AssignMessage>
```

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.03

## 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true"
name="CreateResponseJSON">
  <DisplayName>CreateResponseJSON</DisplayName>
  <Copy source="response">
    <StatusCode>true</StatusCode>
  </Copy>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="response"/>
  <Set>
    <Payload contentType="application/json;charset=utf-8" variablePrefix="@@"
variableSuffix="#">
      [@response-city.object#, @calloutresponse-city2.object#, @calloutresponse-
getcities.object#]
    </Payload>
  </Set>
</AssignMessage>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.3. Assign Message XML 仕様](#)」をご参照ください。

## 5) ポリシーの配置

以下の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。

The screenshot shows the API Management console for a flow named 'お天気情報取得'. The left sidebar contains a 'Policies' list with several policies, including 'CreateResponseJSON', 'GetResponseJSON-city', 'GetResponseJSON-city2', 'GetResponseJSON-getcities', 'ServiceCallout-city2', and 'ServiceCallout-getcities'. The main area shows a flow diagram with 'Request' and 'Response' steps. The 'Response' step is highlighted with a red box and the text '①ここをクリック!'. The 'Policies' list is also highlighted with a red box and the text '③ここをクリック!'. A red arrow points from the 'Policies' list to the flow diagram with the text '②ここにドラッグ!'. The top bar has a 'Save' button highlighted with a red box and the text '③ここをクリック!'.

## 6) 動作確認

作成した API Proxy の動作確認を行います。

以下の例のように、URL へアクセスすると JSON 形式のデータが返却されます。

http://{FQDN}:10080/mashup-

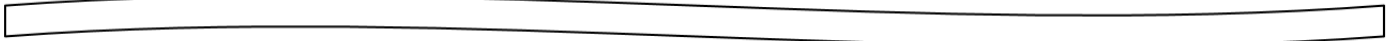
sample/forecast/webservice/json/v1?city={CityCode}&city2={CityCode2}

※http プロトコルでの動作確認例です（ポート番号は 10080 です）。

※{FQDN}: API Proxy の作成時に生成された URL のホスト名（FQDN）を指定します。

※{CityCode}, {CityCode2} は、任意の都市コードに置き換えてください。（例：400040, 410020 など）

```
[
  - [
    - pinpointLocations: [
      - [
        link: "http://weather.livedoor.com/area/forecast/4020200",
        name: "大牟田市"
      ],
      - [
        link: "http://weather.livedoor.com/area/forecast/4020300",
        name: "久留米市"
      ],
      - [
        link: "http://weather.livedoor.com/area/forecast/4020700",
        name: "柳川市"
      ],
      - [
        link: "http://weather.livedoor.com/area/forecast/4021000",
        name: "八女市"
      ],
    ],
  ],
],
```



```
- [
  - response: [
    - location: [
      - [
        city: "佐賀市",
        city_kana: "さがし"
      ],
      - [
        city: "唐津市",
        city_kana: "からつし"
      ],
      - [
        city: "鳥栖市",
        city_kana: "とすし"
      ],
      - [
        city: "多久市",
        city_kana: "たくし"
      ],
    ],
  ],
],
```

追加したAPIのレスポンス

## 2.2. Javascript でのマッシュアップ処理

### 2.2.1. ユースケース概要

JavaScript ポリシーを使用することで、バックエンドサービス（API）へのリクエストや、クライアントへのレスポンスに対して、より複雑な処理をすることができるようになります。

ここでは、異なる 2 つの API の実行結果を、JavaScript 処理にて整形（結合）してからクライアントに返却する API Proxy を実装します。

バックエンドサービスとして、資産管理情報を返却する API と資産稼働状況を返却する API を使用します。2 つの API から返却されたレスポンス（資産管理情報・資産稼働状況）に対して、JavaScript にて共通の ID を持つデータ同士を結合させる処理を行います。

結合されたレスポンスデータは JSON 形式で返却されます。

使用する API は、それぞれ以下のようなレスポンスを返却する想定とします。

#### 資産管理情報

```
{
  "items": [
    {
      "device_id": "00:00:00:00:00:00",
      "item_id": "000",
      "item_name": "現在位置",
      "item_category": "00",
      "time_stamp": "2016-01-01T00:00:00Z",
      "item_img": "http://x/im/pcgazou.png",
      "item_info": "補足情報",
      "item_status": "00"
    },
    {
      "device_id": "32:00:3C:01:B6:05",
      "item_id": "002",
      "item_name": "FMV123 (日本)",
      "item_category": "01",
      "time_stamp": "2015-11-05T05:23:44Z",
      "item_img": "http://x/im/pcgazou.png",
      "item_info": "補足情報",
      "item_status": "01"
    }
  ]
}
```

#### 資産稼働状況

```
{
  "geoworkingstateinfo": [
    {
      "device_id": "00:00:00:00:00:00",
      "device_name": "SN034e8390q4e",
      "latitude": "35.673942",
      "longitude": "139.758820",
      "device_status": "00",
      "time_stamp": "2016-01-01T00:00:00Z"
    },
    {
      "device_id": "32:00:3C:01:B6:05",
      "device_name": "SN034e83",
      "latitude": "35.673942",
      "longitude": "139.758820",
      "device_status": "00",
      "time_stamp": "2016-01-01T00:00:00Z"
    }
  ]
}
```

マッシュアップ後は以下のようなレスポンスを返却する想定とします。

### マッシュアップデータ

```
{
  "work": [
    {
      "device_id": "00:00:00:00:00:00",
      "item_id": "000",
      "item_name": "現在位置",
      "item_category": "00",
      "item_img": "http://x/im/pcgazou.png",
      "item_info": "補足情報",
      "item_status": "00",
      "device_name": "SN034e8390q4e",
      "latitude": "35.673942",
      "longitude": "139.758820",
      "device_status": "00",
      "time_stamp": "2016-01-01T00:00:00Z"
    },
    {
      "device_id": "32:00:3C:01:B6:05",
      "item_id": "002",
      "item_name": "FMV123 (日本) ",
      "item_category": "01",
      "item_img": "http://x/im/pcgazou.png",
      "item_info": "補足情報",
      "item_status": "01",
      "device_name": "SN034e83",
      "latitude": "35.673942",
      "longitude": "139.758820",
      "device_status": "00",
      "time_stamp": "2016-01-01T00:00:00Z"
    }
  ]
}
```

API Proxy の実装フローは以下の通りです。

- 1) API Proxy の作成
  - 1-1) API Proxy の作成
  - 1-2) Conditional Flow (GET) の作成
- 2) Service Callout ポリシーの作成 1 (資産情報の取得)
- 3) Service Callout ポリシーの作成 2 (資産稼働状況の取得)
- 4) Extract Variables ポリシーの作成 1 (資産情報の変数化)
- 5) Extract Variables ポリシーの作成 2 (資産稼働状況の変数化)
- 6) JavaScript ポリシーの作成 (API 実行結果の結合)
- 7) Assign Message ポリシーの作成 (HTTP レスポンスの作成)
- 8) ポリシーの配置
- 9) 動作確認

使用する Policy は以下の通りです。

- Assign Message ポリシー
- Extract Variables ポリシー
- JavaScript ポリシー
- Service Callout ポリシー

## 2.2.2. 手順

### 1) API Proxy の作成

API Proxy を作成します。

#### 1-1) API Proxy の作成

画面上部の「APIs」メニューをクリックし、API Proxies 画面に遷移します。

The screenshot shows the API Management dashboard. The top navigation bar includes 'API Management', 'Dashboard', 'APIs', 'Publish', and 'Admin'. The 'APIs' menu is highlighted with a red box, and a red arrow points to it with the text 'ここをクリック!'. Below the navigation bar, there are filters for 'Hour', 'Day', 'Week', 'Month', and 'Custom', and a date range selector set to 'From Fri Mar 10 2017, 10:00 am To Fri Mar 17 2017, 10:00 am'. The main content area is titled 'Proxy Traffic' and displays 'No traffic in the selected date range.' Below this, there are two sections: 'Developer Engagement' and 'Developer Apps', both showing 'No traffic in the selected date range.'

API Proxies 画面で、「+ API Proxy」ボタンをクリックします。

The screenshot shows the API Proxies management page. The top navigation bar includes 'API Management', 'Dashboard', 'APIs', 'Publish', 'Analytics', and 'Admin'. The 'APIs' menu is highlighted with a red box, and a red arrow points to it with the text 'ここをクリック!'. Below the navigation bar, there are filters for 'List' and 'Analytics', and a search bar. The main content area is titled 'API Proxies' and displays a table of API Proxies. The table has columns for 'API Proxy', 'Environments', 'Traffic', 'Message Trend by Hour', 'Avg Time', 'Error Rate', 'Modified', and 'Actions'. The table contains three rows of data. The '+ API Proxy' button is highlighted with a red box, and a red arrow points to it with the text 'ここをクリック!'.

API Proxy	Environments	Metrics for Last 24 Hours (prod)				Modified	Actions
		Traffic	Message Trend by Hour	Avg Time	Error Rate		
<a href="#">handson-api20160217020</a>	test	1		8029.00 ms	100.00 %	3 hours ago	<a href="#">x Delete</a> <a href="#">Roles (1)</a>
<a href="#">Hello-World-Nodejs-2</a>	test, prod	3		37.00 ms	0.00 %	19 hours ago	<a href="#">x Delete</a> <a href="#">Roles (1)</a>
<a href="#">Hello-World-Nodejs</a>	test, prod	0				a day ago	<a href="#">x Delete</a> <a href="#">Roles</a>

➤ Build a Proxy (TYPE)

「No Target」を選択し、「Next」ボタンをクリックします。

### Build a Proxy

- Reverse proxy (most common)  
Route inbound requests to backend services.
- Node.js App  
Create a new app in JavaScript and optionally add policies.
- SOAP service  
Create a RESTful or pass-through proxy for a SOAP service.
- No Target  
Create a simple API proxy that does not route to any backend target.  
 Use OpenAPI    Optionally associate the proxy with an OpenAPI (Swagger) document
- Proxy bundle  
Import an existing proxy from a zip archive.

---

**ここをクリック!**

➤ Build a Proxy (DETAILS)

API Proxy の情報を入力し、「Next」ボタンをクリックします。以下は入力例です。

- Proxy Name : api-mashup (任意の名前)
- Proxy Base Path : /api-mashup (任意のパス (自動入力))

### Build a Proxy

TYPE > **DETAILS** > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Specify the proxy details.

Proxy Name\*   
Valid characters are letters, numbers, dash (-), and underscore (\_).

Proxy Base Path\*   
A path component that uniquely identifies this API proxy. The public-facing URL of this API proxy is comprised of your organization name, an environment where this API proxy is deployed, and this Proxy Base Path. Example URL http://test-sandbox-test.apigee.net/api-mashup

Description

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

[Previous](#) [Exit Without Saving](#) **ここをクリック!** [Next](#)



➤ Build a Proxy (SECURITY)

Authentication : 「Pass through (none)」 を選択し、「Next」 ボタンをクリックします。

### Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Secure access for users and clients.

Authorization

- Pass through (none)
- API Key
- OAuth 2.0

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Next

➤ Build a Proxy (VIRTUAL HOSTS)

設定を変えずに「Next」ボタンをクリックします。

Build a Proxy

TYPE DETAILS SECURITY VIRTUAL HOSTS BUILD SUMMARY

Select the virtual hosts this proxy will bind to when it is deployed. You must select at least one virtual host. [Learn more...](#)

<input checked="" type="checkbox"/> Name	Environment	Host Aliases
<input checked="" type="checkbox"/> default	prod	http://:10080
	test	http://:10080
<input checked="" type="checkbox"/> secure	prod	https://:10443
	test	https://:10443

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Next

➤ Build a Proxy (BUILD)

設定を変えずに「Build and Deploy」ボタンをクリックします。

Build a Proxy

TYPE DETAILS SECURITY VIRTUAL HOSTS BUILD SUMMARY

You are ready to build and deploy your API proxy.

Deploy Environments  prod  test

Proxy Name api-mashup

Proxy Type No Target

Virtual Hosts default, secure

Security None

Browser Do not allow direct requests from a browser via CORS

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Build and Deploy

➤ Build a Proxy (SUMMARY)

API Proxy の作成が完了したら、API Proxy のリンクをクリックします。

### Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

- ✓ Generated proxy
- ✓ Uploaded proxy
- ✓ Deployed to test

**ここをクリック!** [View api-mashup proxy](#) in the editor .

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

## 1-2) Conditional Flow (GET) の作成

バックエンドサービスに対するリソースパスと処理 (HTTP Method) の定義を行います。

クライアントからのリクエストがここで定義したパターンと一致する場合に、以降の手順で設定する処理を実行します。

「Develop」タブをクリックし、API proxy editor を開きます。

The screenshot shows the API Management console for the 'api-mashup' proxy. The 'Develop' tab is selected, and a red arrow points to the 'DEVELOP' button in the top right corner. The page displays the 'Revision 1 Summary', 'Deployments' table, 'Proxy Endpoints' table, and 'Target Endpoints' section.

Environment	Revision	Status	URL
test	1	●	http://... [ + ]

Name	Base Path	Target Endpoints
default	/api-mashup	none

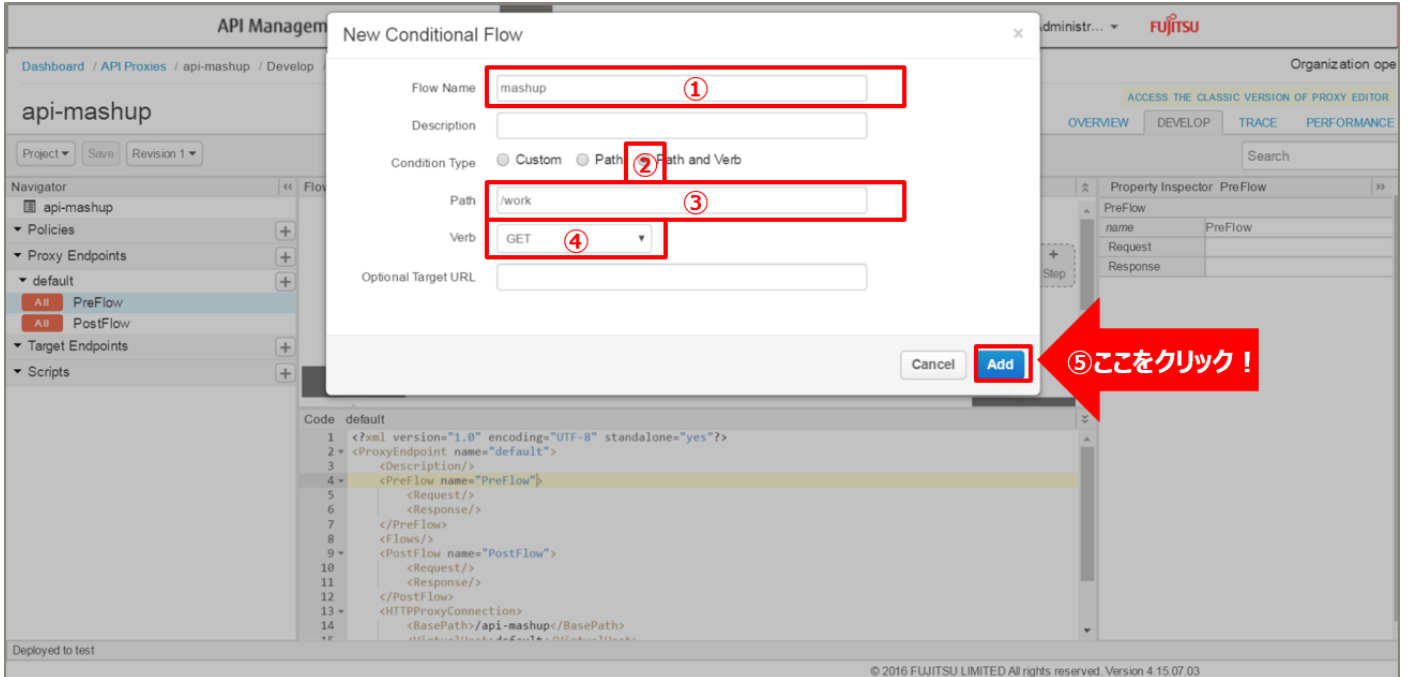
default の「+」ボタンをクリックし、New Conditional Flow を開きます。

The screenshot shows the 'New Conditional Flow' dialog box open for the 'default' proxy endpoint. A red arrow points to the '+' button next to the 'default' endpoint in the 'Proxy Endpoints' list. The dialog box shows a flow diagram with a 'Request' step and a 'Response' step. The 'Code' section shows the XML configuration for the 'PreFlow'.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ProxyEndpoint name="default">
3 <Description/>
4 <PreFlow name="PreFlow">
5 <Request/>
6 <Response/>
7 </PreFlow>
8 </Flows/>
9 <PostFlow name="PostFlow">
10 <Request/>
11 <Response/>
12 </PostFlow>
13 <HTTPProxyConnection>
14 <BasePath>/api-mashup</BasePath>
```

Flow の情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

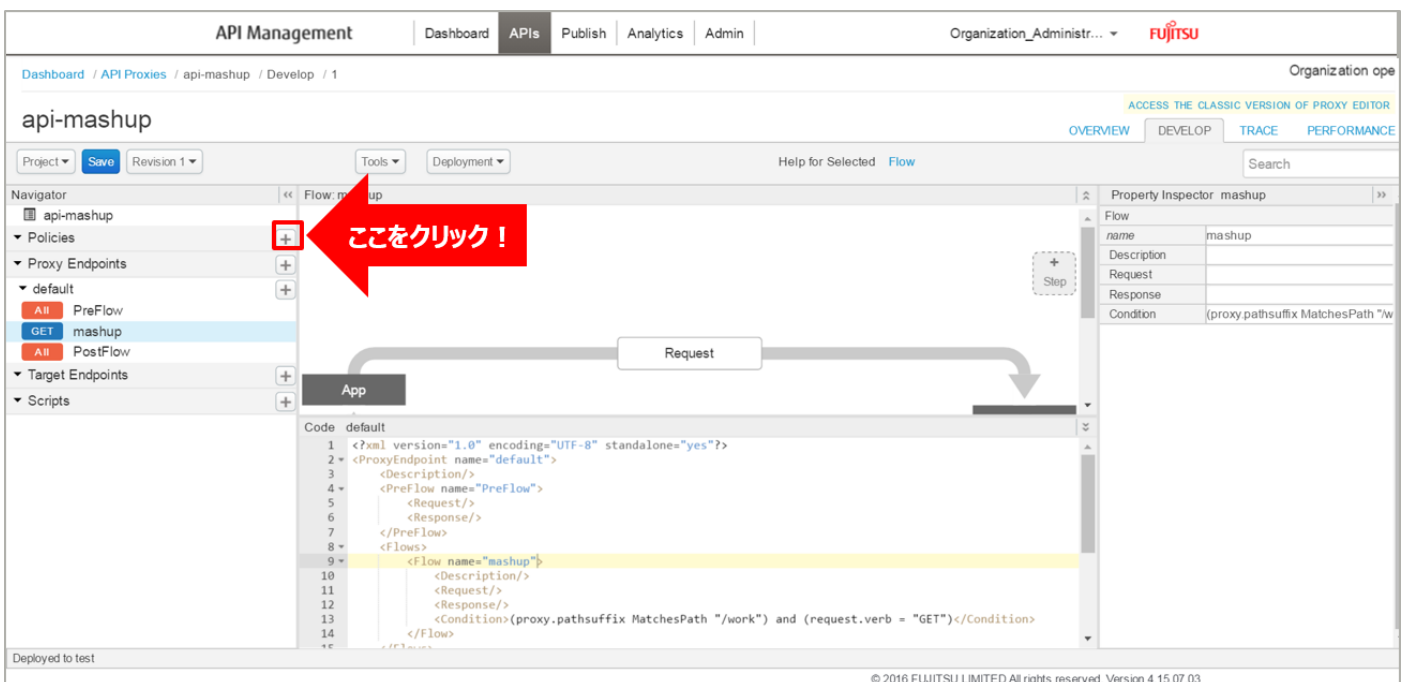
- Flow Name : mashup (任意の名前)
- Condition Type : 「Path and Verb」を選択
- Path : /work (任意のパス)
- Verb : 「GET」を選択



## 2) Service Callout ポリシーの作成 1 (資産情報の取得)

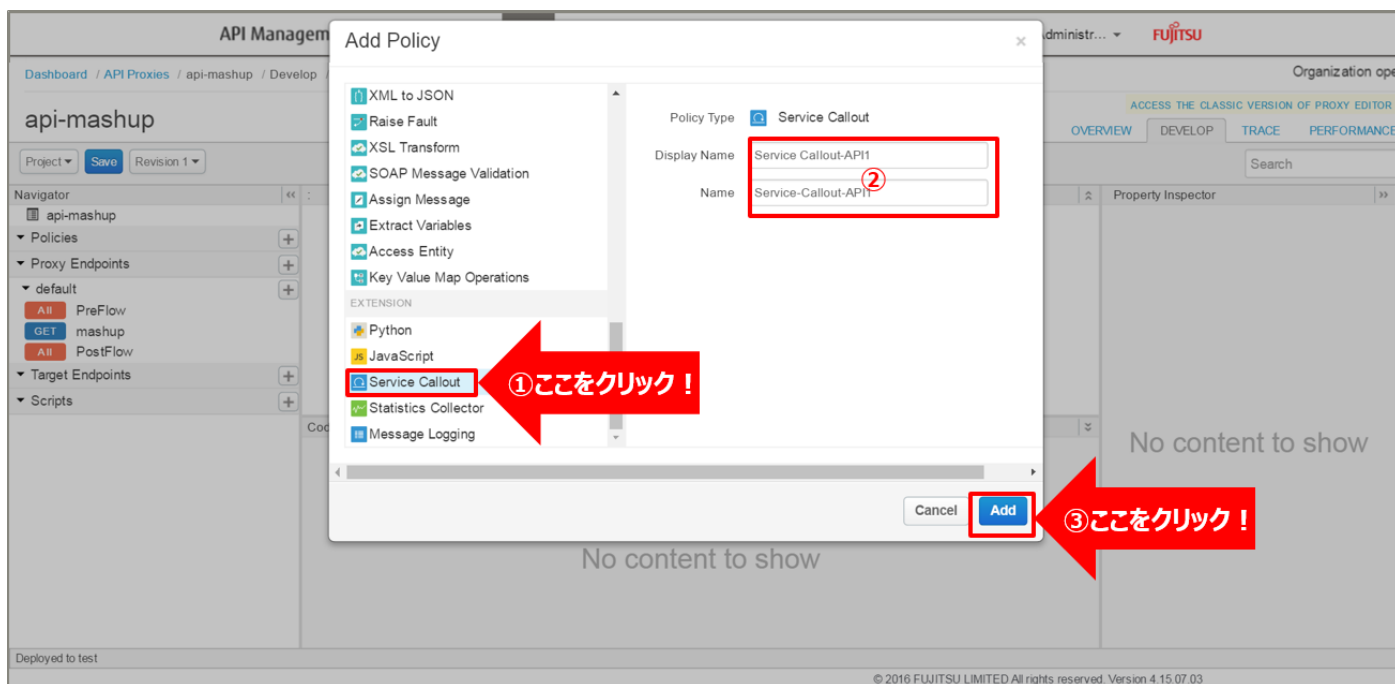
資産管理情報を取得する API を呼び出す Service Callout ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。

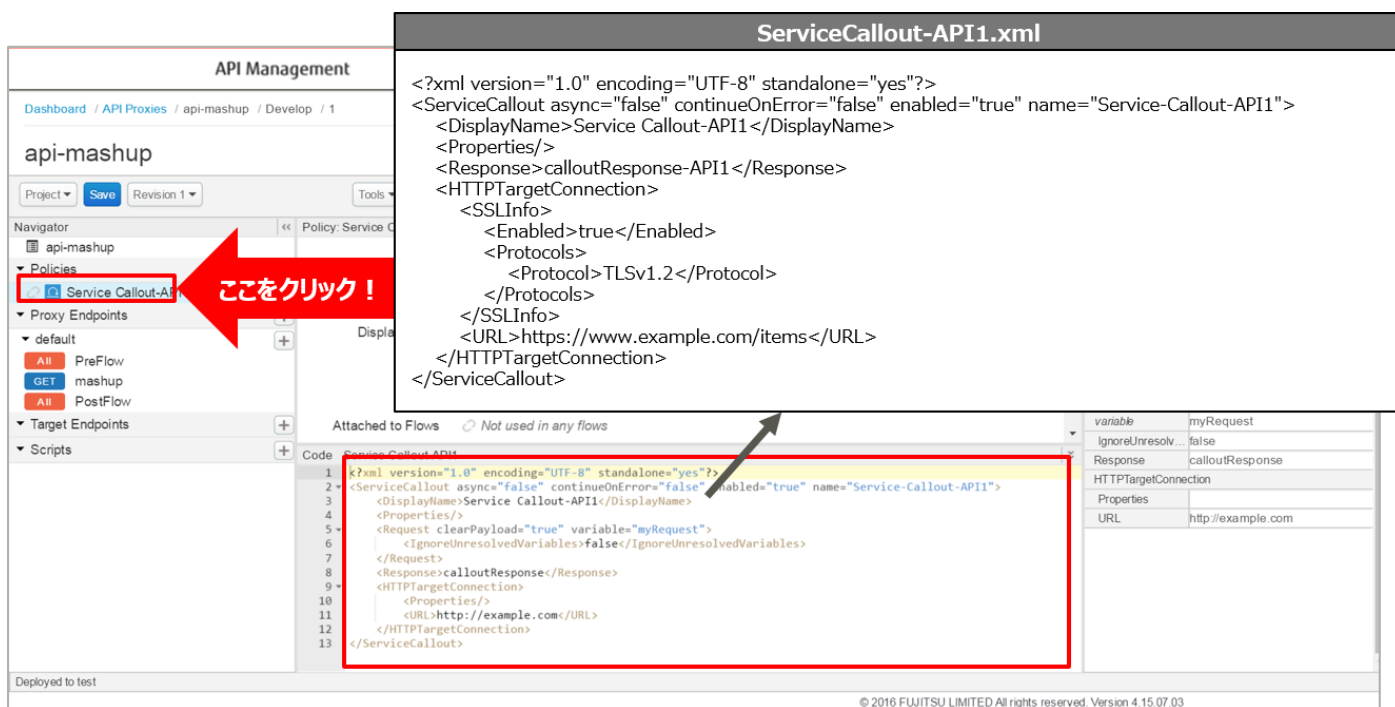


「Service Callout」をクリック後、以下の通り入力し、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : Service Callout-API1 (任意の名前)
- Name : Service Callout-API1 (任意の名前)



追加した Service Callout ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。



【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceCallout async="false" continueOnError="false" enabled="true" name="Service-Callout-API1">
  <DisplayName>Service Callout-API1</DisplayName>
  <Properties/>
  <Response>calloutResponse-API1</Response>
  <HTTPTargetConnection>
    <SSLInfo>
      <Enabled>true</Enabled>
      <Protocols>
        <Protocol>TLSv1.2</Protocol>
      </Protocols>
    </SSLInfo>
    <URL>https://www.example.com/items</URL>
  </HTTPTargetConnection>
</ServiceCallout>
```

※定義内容の詳細は、「[A3.1. Service Callout XML 仕様](#)」をご参照ください。

3) Service Callout ポリシーの作成 2 (資産稼働状況の取得)

2) と同様の手順で、資産稼働状況を取得する API を呼び出す Service Callout ポリシーを追加します。

以下は Add Policy で入力するポリシーの情報の入力例です。

- Display Name : Service Callout-API2 (任意の名前)
- Name : Service Callout-API2 (任意の名前)

追加した Service Callout ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

```
ServiceCallout-API2.xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceCallout async="false" continueOnError="false" enabled="true" name="Service-Callout-API2">
  <DisplayName>Service Callout-API2</DisplayName>
  <Properties/>
  <Response>calloutResponse-API2</Response>
  <HTTPTargetConnection>
    <SSLInfo>
      <Enabled>true</Enabled>
      <Protocols>
        <Protocol>TLSv1.2</Protocol>
      </Protocols>
    </SSLInfo>
    <URL>https://www.example.com/geoworkingstateinfo</URL>
  </HTTPTargetConnection>
</ServiceCallout>
```

【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceCallout async="false" continueOnError="false" enabled="true" name="Service-Callout-API2">
  <DisplayName>Service Callout-API2</DisplayName>
  <Properties/>
  <Response>calloutResponse-API2</Response>
  <HTTPTargetConnection>
    <SSLInfo>
      <Enabled>true</Enabled>
      <Protocols>
        <Protocol>TLSv1.2</Protocol>
      </Protocols>
    </SSLInfo>
    <URL>https://www.example.com/geoworkingstateinfo</URL>
  </HTTPTargetConnection>
</ServiceCallout>
```

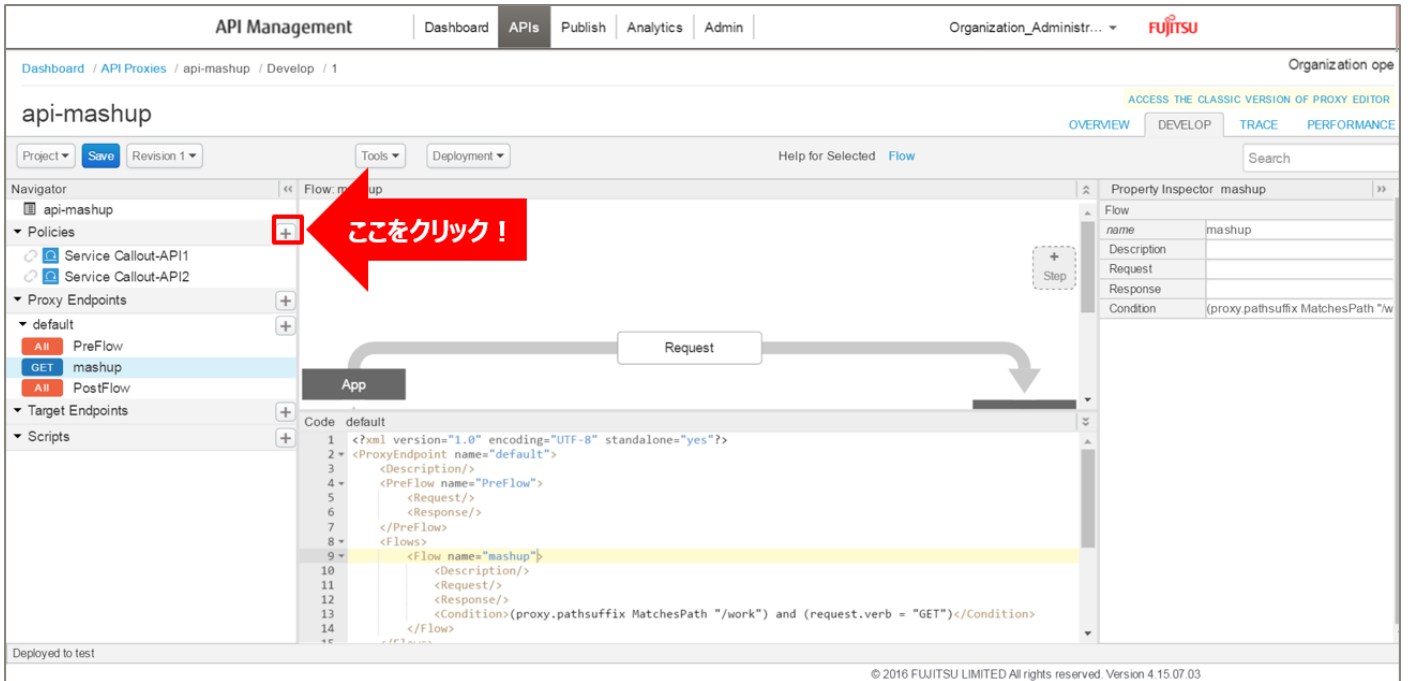
※定義内容の詳細は、「[A3.1. Service Callout XML 仕様](#)」をご参照ください。

4) Extract Variables ポリシーの作成 1（資産情報の変数化）

Service Callout ポリシーで取得した資産情報を変数として格納（変数化）する Extract Variables ポリシーを追加します。

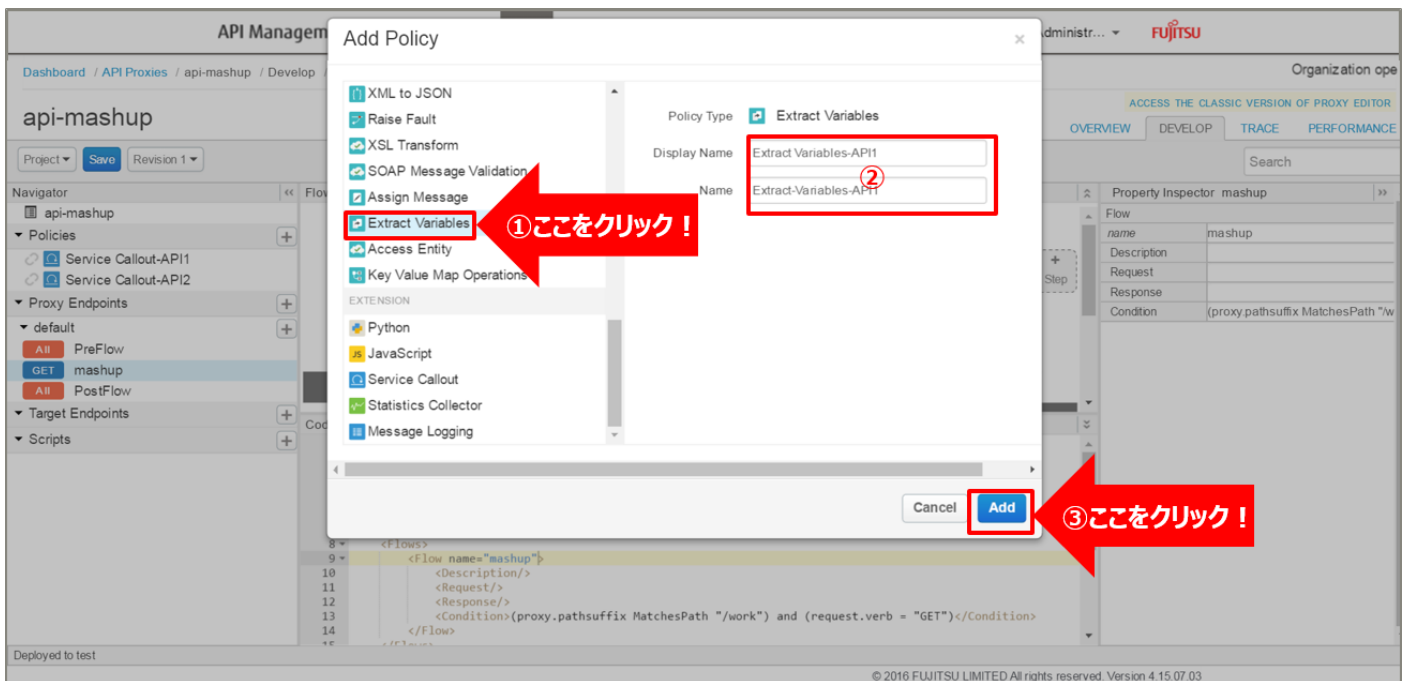


Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Extract Variables」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。  
以下は入力例です。

- Display Name : Extract Variables-API1 (任意の名前)
- Name : Extract Variables-API1 (任意の名前)



追加した Extract Variables ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

**API Management**

Dashboard / API Proxies / api-mashup / Develop / 1

api-mashup

Project Save Revision 1 Tools

Navigator

- api-mashup
  - API Proxies
    - Service Callout-API1
    - Service Callout-API2
  - Proxy Endpoints
    - default
      - PreFlow
      - GET mashup
      - PostFlow
  - Target Endpoints
  - Scripts

Policy: Extract Variab

Display Na

Attached to Flows Not used in any flows

Code

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ExtractVariables async="false" continueOnError="false" enabled="true" name="Extract-Variables-API1">
3   <DisplayName>Extract Variables-API1</DisplayName>
4   <Properties/>
5   <URIPath name="name"/>
6   <QueryParam name="name"/>
7   <Header name="name"/>
8   <FormParam name="name"/>
9   <Variable name="name"/>
10  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
11  <JSONPayload>
12    <Variable name="name">
13      <JSONPath>{example}</JSONPath>
14    </Variable>
15  </JSONPayload>
16  <Source clearPayload="false">calloutResponse-API1</Source>
17  <VariablePrefix>API1Response</VariablePrefix>
18 </ExtractVariables>

```

QueryParam

name	name
Header	name
name	name
FormParam	name
name	name
Variable	name
name	name
IgnoreUnresolv...	true
JSONPayload	
Variable	
name	name
JSONPath	{example}
Source	calloutResponse-API1

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.03

#### 【定義例】

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true" name="Extract-Variables-API1">
  <DisplayName>Extract Variables-API1</DisplayName>
  <Properties/>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <JSONPayload>
    <Variable name="json">
      <JSONPath>$.items</JSONPath>
    </Variable>
  </JSONPayload>
  <Source clearPayload="false">calloutResponse-API1</Source>
  <VariablePrefix>API1Response</VariablePrefix>
</ExtractVariables>

```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.4. Extract Variables XML 仕様](#)」をご参照ください。

## 5) Extract Variables ポリシーの作成 2 (資産稼働状況の変数化)

4) と同様の手順で、Service Callout ポリシーで取得した資産稼働状況を変数として格納 (変数化) する Extract Variables ポリシーを追加します。

以下は Add Policy で入力するポリシーの情報の入力例です。

- Display Name : Extract Variables-API2 (任意の名前)
- Name : Extract Variables-API2 (任意の名前)

追加した Extract Variables ポリシーを選択し、ポリシー編集画面を表示します。

必要に応じてポリシーの定義を編集します。

**ExtractVariables-API2.xml**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true" name="Extract-Variables-API2">
  <DisplayName>Extract Variables-API2</DisplayName>
  <Properties/>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <JSONPayload>
    <Variable name="json">
      <JSONPath>$.geoworkingstateinfo</JSONPath>
    </Variable>
  </JSONPayload>
  <Source clearPayload="false">calloutResponse-API2</Source>
  <VariablePrefix>API2Response</VariablePrefix>
</ExtractVariables>
```

<JSONPayload>  
<Variable name="json">  
<JSONPath>\$.geoworkingstateinfo</JSONPath>  
</Variable>  
</JSONPayload>

JSON全体を変数へ格納。

### 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true" name="Extract-Variables-API2">
  <DisplayName>Extract Variables-API2</DisplayName>
  <Properties/>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <JSONPayload>
    <Variable name="json">
      <JSONPath>$.geoworkingstateinfo</JSONPath>
    </Variable>
  </JSONPayload>
  <Source clearPayload="false">calloutResponse-API2</Source>
  <VariablePrefix>API2Response</VariablePrefix>
</ExtractVariables>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.4. Extract Variables XML 仕様](#)」をご参照ください。

## 6) JavaScript ポリシーの作成 (API 実行結果の結合)

JavaScript を実行するための JavaScript ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。

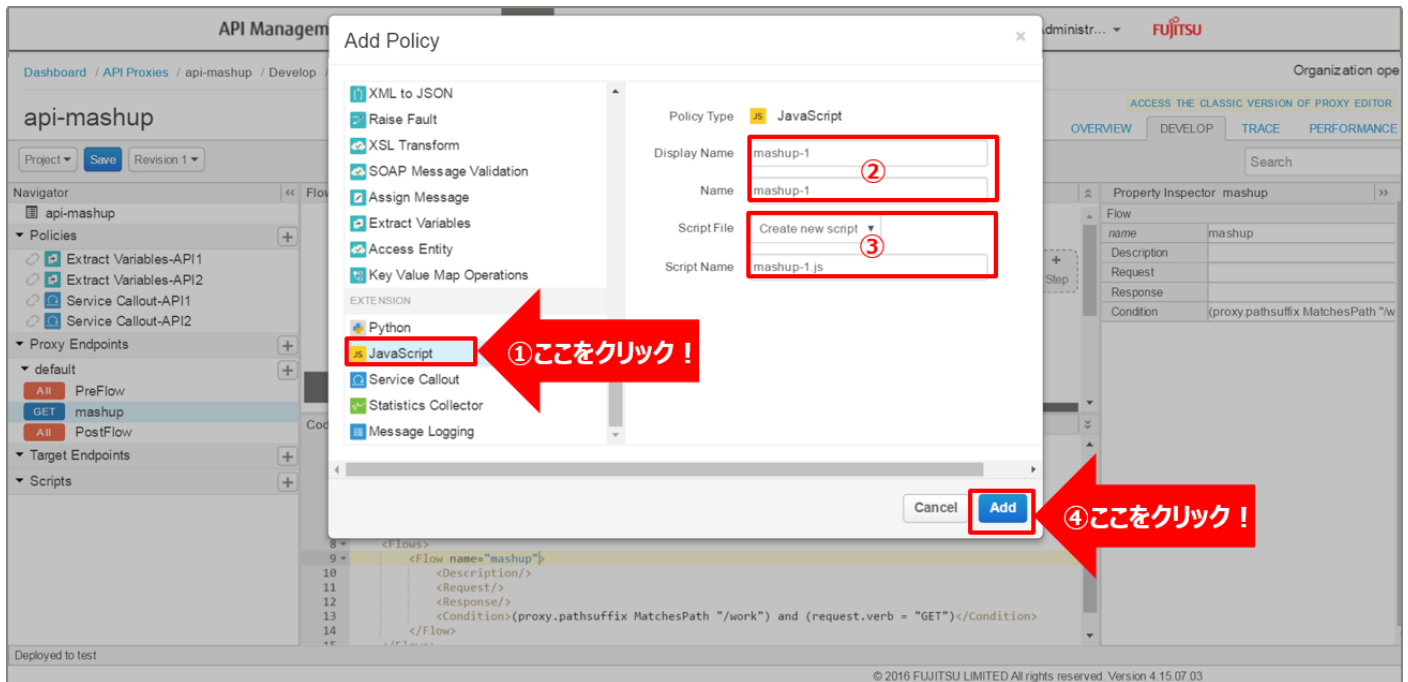
The screenshot shows the API Management console interface. The top navigation bar includes 'API Management', 'Dashboard', 'APIs', 'Publish', 'Analytics', and 'Admin'. The breadcrumb trail is 'Dashboard / API Proxies / api-mashup / Develop / 1'. The main content area is titled 'api-mashup' and has tabs for 'OVERVIEW', 'DEVELOP', 'TRACE', and 'PERFORMANCE'. The 'DEVELOP' tab is active, showing a flow diagram with 'App' and 'Request' components. A red arrow points to a '+' button in the 'Policies' section of the left-hand 'Navigator' pane. The 'Code' pane shows XML configuration for a 'default' proxy endpoint, including a 'PreFlow' and a 'Flow' named 'mashup'. The 'Property Inspector' on the right shows details for the 'mashup' flow.

name	mashup
Description	
Request	
Response	
Condition	(proxy.pathsuffix MatchesPath "/work")

```
Code default
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ProxyEndpoint name="default">
3   <Description/>
4   <PreFlow name="PreFlow">
5     <Request/>
6     <Response/>
7   </PreFlow>
8   <Flows>
9     <Flow name="mashup">
10      <Description/>
11      <Request/>
12      <Response/>
13      <Condition>(proxy.pathsuffix MatchesPath "/work") and (request.verb = "GET")</Condition>
14    </Flow>
15  </Flows>
16 </ProxyEndpoint>
```

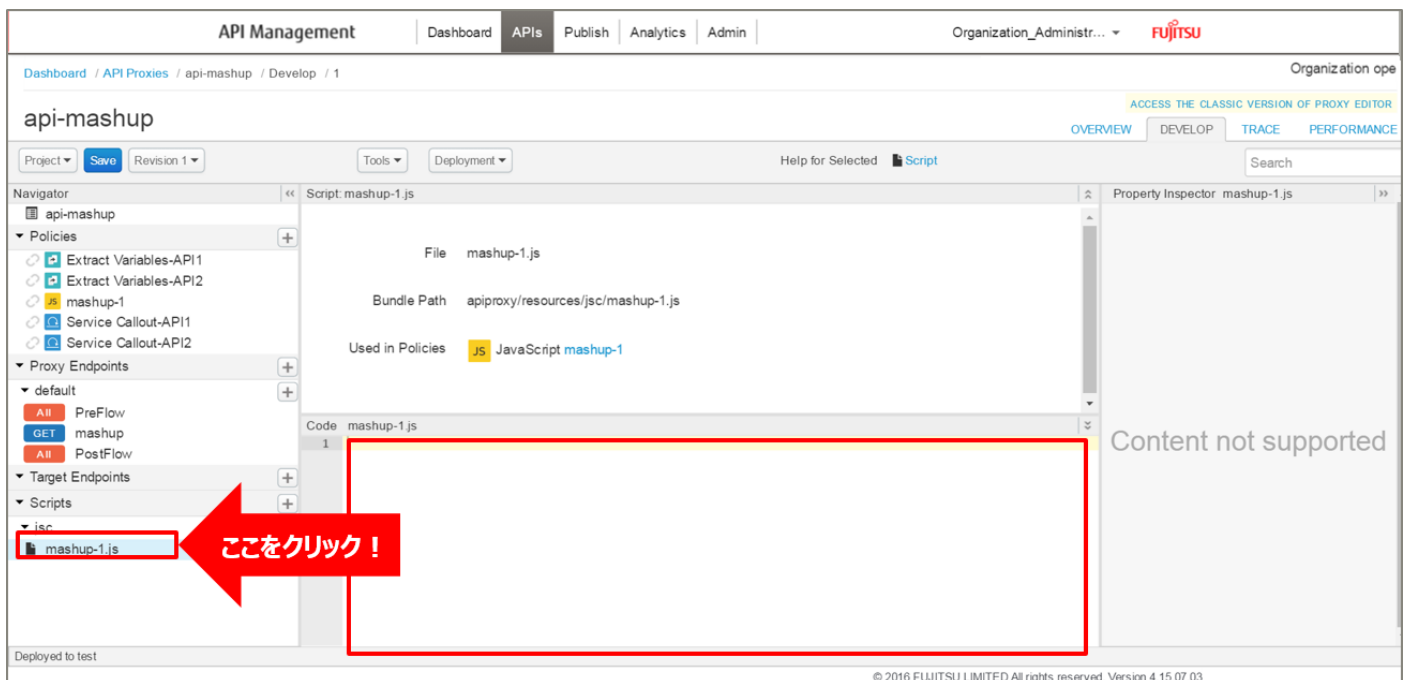
「JavaScript」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。  
以下は入力例です。

- Display Name : mashup-1 (任意の名前)
- Name : mashup-1 (任意の名前)
- Script File : 「Create new script」を選択
- Script Name : mashup-1.js (任意の名前)



Scripts に作成されたスクリプトを選択し、スクリプト編集画面を表示します。

取得した資産管理情報および資産稼働状況から、共通 ID(device\_id)を持つデータを抽出して結合させるスクリプトを定義します。



【定義例】

```
//API1 のレスポンスを変数に格納
var api1_response = context.getVariable("API1Response.json");
var api1_jsondata = JSON.parse(api1_response);
//API2 のレスポンスを変数に格納
var api2_response = context.getVariable("API2Response.json");
var api2_jsondata = JSON.parse(api2_response);

var workstate = [];

//API2 レスポンスをループ処理
for(var i = 0; i < api2_jsondata.length; i++){
  //API1 レスポンスをループ処理
  for(var j = 0; j < api1_jsondata.length; j++){
    var mashupJSON = {};
    if(api2_jsondata[i].device_id == api1_jsondata[j].device_id){
      //API1 と API2 の device_id が一致した場合、必要な情報をそれぞれレスポンスから取得
      mashupJSON.device_id = api2_jsondata[i].device_id;
      mashupJSON.item_id = api1_jsondata[j].item_id;
      mashupJSON.item_name = api1_jsondata[j].item_name;
      mashupJSON.item_category = api1_jsondata[j].item_category;
      mashupJSON.item_img = api1_jsondata[j].item_img;
      mashupJSON.item_info = api1_jsondata[j].item_info;
      mashupJSON.item_status = api1_jsondata[j].item_status;
      mashupJSON.device_name = api2_jsondata[i].device_name;
      mashupJSON.latitude = api2_jsondata[i].latitude;
      mashupJSON.longitude = api2_jsondata[i].longitude;
      mashupJSON.device_status = api2_jsondata[i].device_status;
      mashupJSON.time_stamp = api2_jsondata[i].time_stamp;
      //マッシュアップした JSON データを配列に入れる
      workstate.push(mashupJSON);
    }
  }
}

//マッシュアップした配列を JSON 配列形式にする
var mashupResponse = {};
mashupResponse.workstate = workstate;
var JSONresponse = JSON.stringify(mashupResponse);
context.setVariable("mashupResponse", JSONresponse);
```

## 7) Assign Message ポリシーの作成 (HTTP レスポンスの作成)

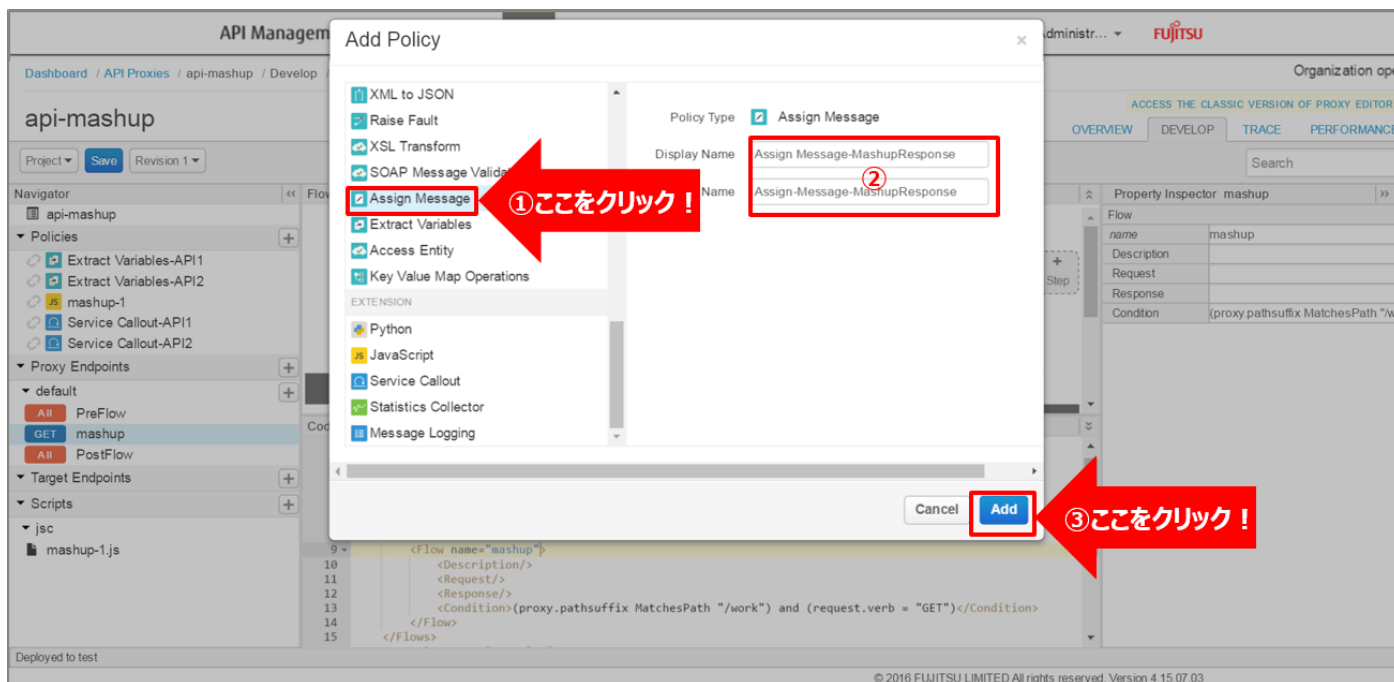
HTTP メッセージを作成する Assign Message ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Assign Message」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。  
以下は入力例です。

- Display Name : Assign Message-MashupResponse (任意の名前)
- Name : Assign Message-MashupResponse (任意の名前)



追加した Assign Message ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true" name="Assign-Message-MashupResponse">
  <DisplayName>Assign Message-MashupResponse</DisplayName>
  <Properties/>
  <Set>
    <Payload contentType="application/json; charset=utf-8" variablePrefix="@" variableSuffix="#">
      @mashupResponse#
    </Payload>
  </Set>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="request"/>
</AssignMessage>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.3. Assign Message XML 仕様](#)」をご参照ください。



## 8) ポリシーの配置

以下の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。

API Management | Dashboard | APIs | Publish | Analytics | Admin | Organization\_Administr... | FUJITSU

Dashboard / API Proxies / api-mashup / Develop / 1

Organization open

ACCESS THE CLASSIC VERSION OF PROXY EDITOR

OVERVIEW | DEVELOP | TRACE | PERFORMANCE

Project Save ④ここをクリック! Tools Deployment Help for Selected Flow Search

Navigator

- api-mashup
  - Assign Message-MashupResponse ③
  - Extract Variables-API1 ②
  - Extract Variables-API2 ②
  - mashup-1 ③
  - Service Callout-API1 ②
  - Service Callout-API2 ②
- Proxy Endpoints
  - default
    - PreFlow
    - GET mashup ①ここをクリック!
    - PostFlow
  - Target Endpoints
- Scripts
  - jsc
    - mashup-1.js

Flow mashup

- Request
  - Service Callout-API1
  - Extract Variable s-API1
  - Service Callout-API2
  - Extract Variable s-API2
- Response
  - Assign Message-MashupResponse
  - mashup-1

Property Inspector mashup

Flow	Property Inspector mashup
name	mashup
Description	
Request	
Step	
Name	Service-Callout-API1
Step	
Name	Extract-Variables-API1
Step	
Name	Service-Callout-API2
Step	
Name	Extract-Variables-API2
Response	
Step	
Name	mashup-1
Step	
Name	Assign-Message-MashupRespor
Condition	(proxy.pathsuffix MatchesPath "w

Deployed to test

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.03

## 9) 動作確認

作成した API Proxy の動作確認を行います。

以下の例のように、URL へアクセスすると JSON 形式のデータが返却されます。

http://{FQDN}:10080/api-mashup/work

※http プロトコルでの動作確認例です (ポート番号は 10080 です)。

※{FQDN}: API Proxy の作成時に生成された URL のホスト名 (FQDN) を指定します。

```
{
  - workstate: [
    - [
      device_id: "00:00:00:01:00:00",
      item_id: "001",
      item_name: "FMV100",
      item_category: "01",
      item_img: "01.png",
      item_info: "(J) FMV100",
      item_status: "01",
      device_name: "(J) SN034e8390q40",
      latitude: "35.876261",
      longitude: "139.763921",
      device_status: "01",
      time_stamp: "2016-01-01T00:00:00Z"
    ],
    - [
      device_id: "00:00:00:01:00:01",
      item_id: "002",
      item_name: "FMV101",
      item_category: "01",
      item_img: "01.png",
      item_info: "(J) FMV101",
      item_status: "01",
      device_name: "(J) SN034e8390q41",
      latitude: "35.87756",
      longitude: "139.76285",
      device_status: "01",
      time_stamp: "2016-01-01T00:00:01Z"
    ],
    - [
      device_id: "00:00:00:01:00:02",
      item_id: "003",
      item_name: "FMV102",
      item_category: "01",
      item_img: "06.png",
      item_info: "(J) FMV102",
      item_status: "06",
      device_name: "(J) SN034e8390q42",
      latitude: "35.872012",
      longitude: "139.759015",
      device_status: "06",
      time_stamp: "2016-01-01T00:00:02Z"
    ]
  ]
}
```

### 3. 変換

本サービスは、インターネット経由でアクセスできる HTTP ベースのサービスであれば、どのような API でもバックエンドサービスとして指定することができます。

ポリシーを使用することで、クライアントからのリクエスト情報をバックエンドサービスが要求する形式に変換したり、バックエンドサービスからのレスポンス情報をクライアントが要求する形式に変換したりすることができるためです。

#### 3.1. HTTP メソッド変換

##### 3.1.1. ユースケース概要

バックエンドサービスに対する処理 (HTTP Method) を変換する API Proxy を実装します。

バックエンドサービスとして、ユーザー情報を削除 (DELETE) する API を使用します。

クエリパラメーターにユーザー ID を指定すると、指定されたユーザー情報を削除します。ポリシーを使用することで、削除したユーザー情報をレスポンスとして取得 (GET) できるようにします。

レスポンスデータ (ユーザー情報) は JSON 形式で返却されます。

API Proxy の実装フローは以下の通りです。

- 1) API Proxy の作成
  - 1-1) API Proxy の作成
  - 1-2) Conditional Flow (DELETE) の作成
- 2) Extract Variables ポリシーの作成 (ユーザー ID の抽出)
- 3) Service Callout ポリシーの作成 (ユーザー情報の削除・取得)
- 4) Extract Variables ポリシーの作成 (ユーザー情報の変数化)
- 5) Assign Message ポリシーの作成 (HTTP レスポンスの作成)
- 6) ポリシーの配置
- 7) 動作確認

使用する Policy は以下の通りです。

- Assign Message ポリシー
- Extract Variables ポリシー
- Service Callout ポリシー

### 3.1.2. 手順

#### 1) API Proxy の作成

API Proxy を作成します。

##### 1-1) API Proxy の作成

画面上部の「APIs」メニューをクリックし、API Proxies 画面に遷移します。

The screenshot shows the API Management dashboard. The top navigation bar includes 'API Management', 'Dashboard', 'APIs', 'Publish', and 'Admin'. The 'APIs' menu is highlighted with a red box, and a red arrow points to it with the text 'ここをクリック!'. Below the navigation bar, there are filters for 'Hour', 'Day', 'Week', 'Month', and 'Custom', and a date range selector set to 'From Fri Mar 10 2017, 10:00 am To Fri Mar 17 2017, 10:00 am'. The main content area is titled 'Proxy Traffic' and displays 'No traffic in the selected date range.' Below this, there are two sections: 'Developer Engagement' and 'Developer Apps', both showing 'No traffic in the selected date range.'

API Proxies 画面で、「+ API Proxy」ボタンをクリックします。

The screenshot shows the 'API Proxies' list page. The top navigation bar includes 'API Management', 'Dashboard', 'APIs', 'Publish', 'Analytics', and 'Admin'. The 'APIs' menu is highlighted. Below the navigation bar, there are tabs for 'List' and 'Analytics'. A search bar is present. The main content area displays a table of API Proxies. The table has columns for 'API Proxy', 'Environments', 'Traffic', 'Message Trend by Hour', 'Avg Time', 'Error Rate', 'Modified', and 'Actions'. The table contains three rows of data. A red box highlights the '+ API Proxy' button in the top right corner, and a red arrow points to it with the text 'ここをクリック!'.

API Proxy	Environments	Metrics for Last 24 Hours (prod)				Modified	Actions
		Traffic	Message Trend by Hour	Avg Time	Error Rate		
<a href="#">handson-api20160217020</a>	test	1		8029.00 ms	100.00 %	3 hours ago	<a href="#">x Delete</a> <a href="#">Roles (1)</a>
<a href="#">Hello-World-Nodejs-2</a>	test, prod	3		37.00 ms	0.00 %	19 hours ago	<a href="#">x Delete</a> <a href="#">Roles (1)</a>
<a href="#">Hello-World-Nodejs</a>	test, prod	0				a day ago	<a href="#">x Delete</a> <a href="#">Roles</a>

➤ Build a Proxy (TYPE)

「No Target」を選択し、「Next」ボタンをクリックします。

### Build a Proxy

- Reverse proxy (most common)  
Route inbound requests to backend services.
- Node.js App  
Create a new app in JavaScript and optionally add policies.
- SOAP service  
Create a RESTful or pass-through proxy for a SOAP service.
- No Target  
Create a simple API proxy that does not route to any backend target.  
 Use OpenAPI    Optionally associate the proxy with an OpenAPI (Swagger) document
- Proxy bundle  
Import an existing proxy from a zip archive.

ここをクリック！

➤ Build a Proxy (DETAILS)

API Proxy の情報を入力し、「Next」ボタンをクリックします。以下は入力例です。

- Proxy Name : http-method-conversion (任意の名前)
- Proxy Base Path : /http-method-conversion (任意のパス (自動入力))

### Build a Proxy

TYPE → **DETAILS** → SECURITY → VIRTUAL HOSTS → BUILD → SUMMARY

Specify the proxy details.

Proxy Name \*   
Valid characters are letters, numbers, dash (-), and underscore (\_).

Proxy Base Path \*   
A path component that uniquely identifies this API proxy. The public-facing URL of this API proxy is comprised of your organization name, an environment where this API proxy is deployed, and this Proxy Base Path. Example URL http://test-sandbox-test.apigee.net/http-method-conversion

Description

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

[Previous](#) [Exit Without Saving](#) **ここをクリック!** [Next](#)

➤ Build a Proxy (SECURITY)

Authentication : 「Pass through (none)」 を選択し、「Next」 ボタンをクリックします。

### Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Secure access for users and clients.

Authorization

- Pass through (none)
- API Key
- OAuth 2.0

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Next

➤ Build a Proxy (VIRTUAL HOSTS)

設定を変えずに「Next」ボタンをクリックします。

Build a Proxy

TYPE DETAILS SECURITY VIRTUAL HOSTS BUILD SUMMARY

Select the virtual hosts this proxy will bind to when it is deployed. You must select at least one virtual host. [Learn more...](#)

<input checked="" type="checkbox"/> Name	Environment	Host Aliases
<input checked="" type="checkbox"/> default	prod	http://:10080
	test	http://:10080
<input checked="" type="checkbox"/> secure	prod	https://:10443
	test	https://:10443

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Next

➤ Build a Proxy (BUILD)

設定を変えずに「Build and Deploy」ボタンをクリックします。

Build a Proxy

TYPE DETAILS SECURITY VIRTUAL HOSTS BUILD SUMMARY

You are ready to build and deploy your API proxy.

Deploy Environments  prod  test

Proxy Name http-method-conversion

Proxy Type No Target

Virtual Hosts default, secure

Security None

Browser Do not allow direct requests from a browser via CORS

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Build and Deploy

➤ Build a Proxy (SUMMARY)

API Proxy の作成が完了したら、API Proxy のリンクをクリックします。

Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

- ✓ Generated proxy
- ✓ Uploaded proxy
- ✓ Deployed to test

ここをクリック！ [http-method-conversion](#) proxy in the editor .

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

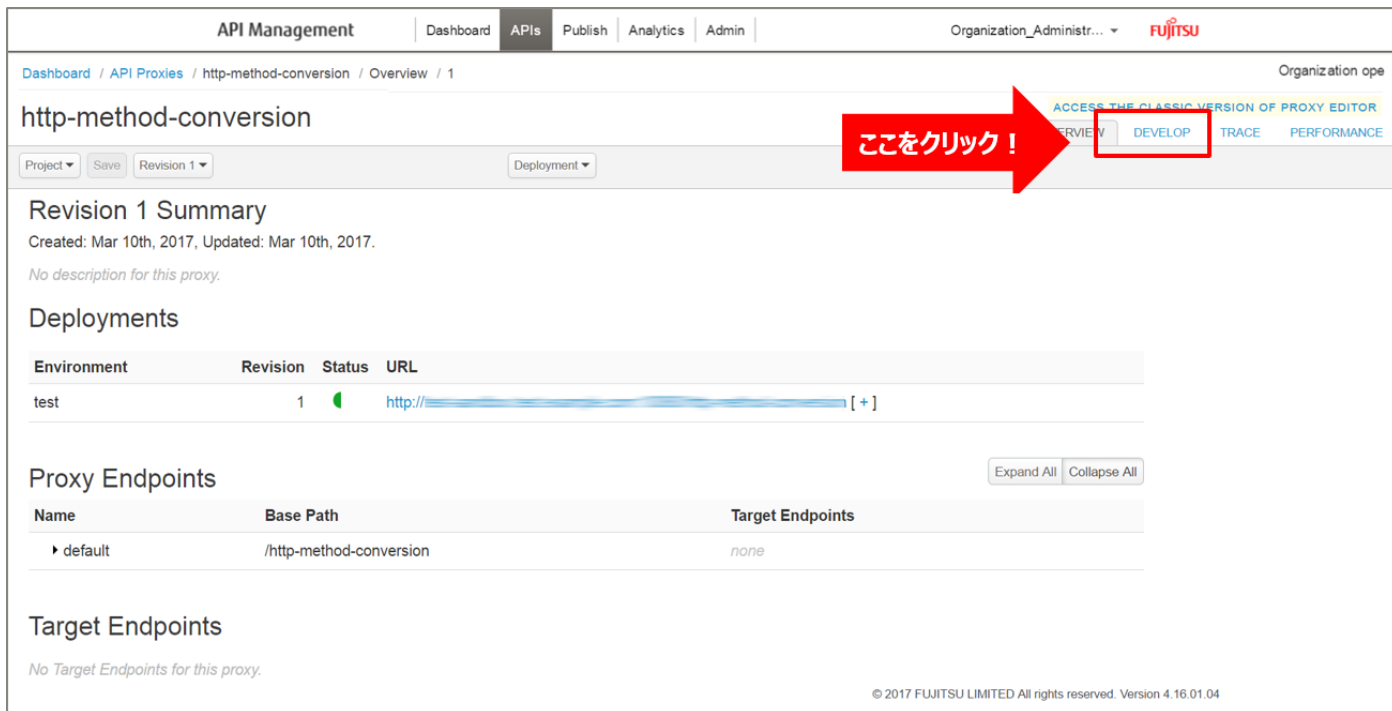


## 1-2) Conditional Flow (DELETE) の作成

バックエンドサービスに対するリソースパスと処理 (HTTP Method) の定義を行います。

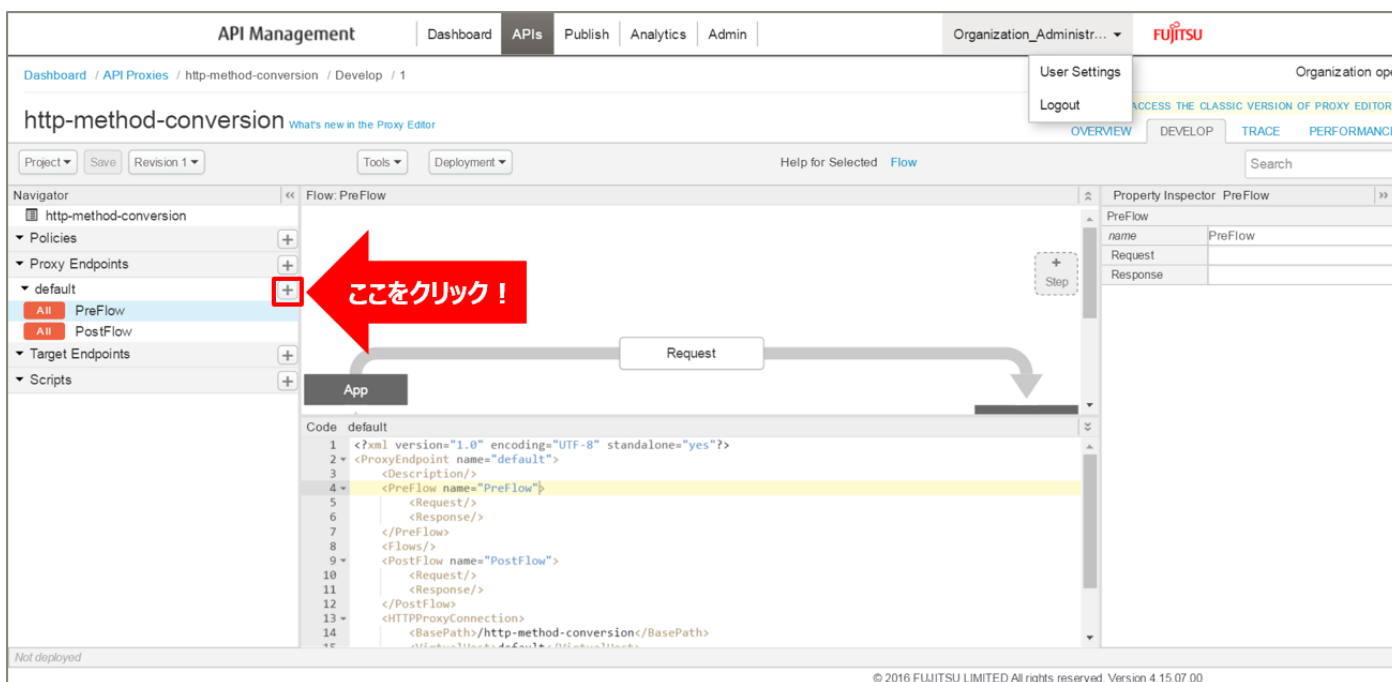
クライアントからのリクエストがここで定義したパターンと一致する場合に、以降の手順で設定する処理を実行します。

「Develop」タブをクリックし、API proxy editor を開きます。



The screenshot shows the API Management console for the 'http-method-conversion' proxy. The top navigation bar includes 'Dashboard', 'APIs', 'Publish', 'Analytics', and 'Admin'. The 'DEVELOP' tab is highlighted with a red box and a red arrow pointing to it with the text 'ここをクリック!' (Click here!). Below the navigation bar, there is a 'Revision 1 Summary' section, a 'Deployments' table with one entry for 'test', and a 'Proxy Endpoints' table with one entry for 'default'. The 'Target Endpoints' section is empty.

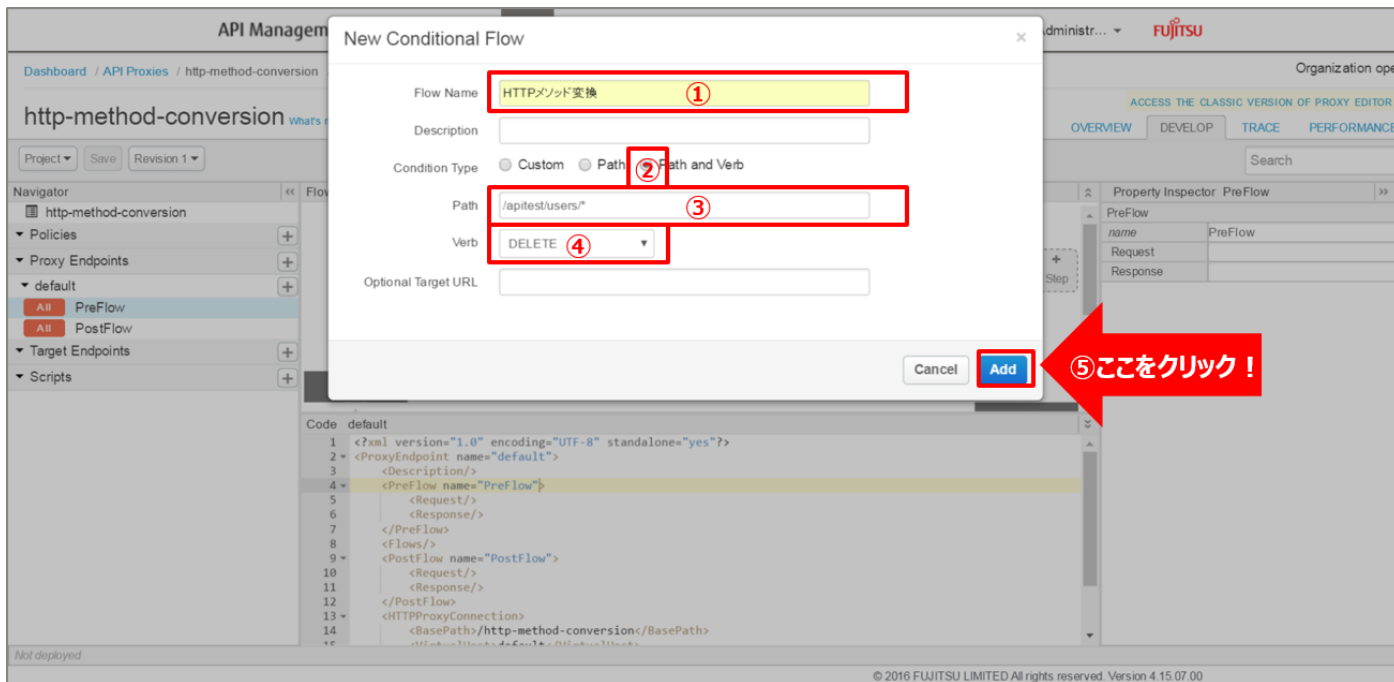
default の「+」ボタンをクリックし、New Conditional Flow を開きます。



The screenshot shows the API Management console for the 'http-method-conversion' proxy in the 'Develop' editor. The left sidebar shows a tree view with 'default' selected, and a red arrow points to the '+' button next to it with the text 'ここをクリック!' (Click here!). The main area shows a flow diagram with a 'Request' step and an 'App' step. The 'Code' section shows XML configuration for the proxy endpoint, including a 'PreFlow' section.

Flow の情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

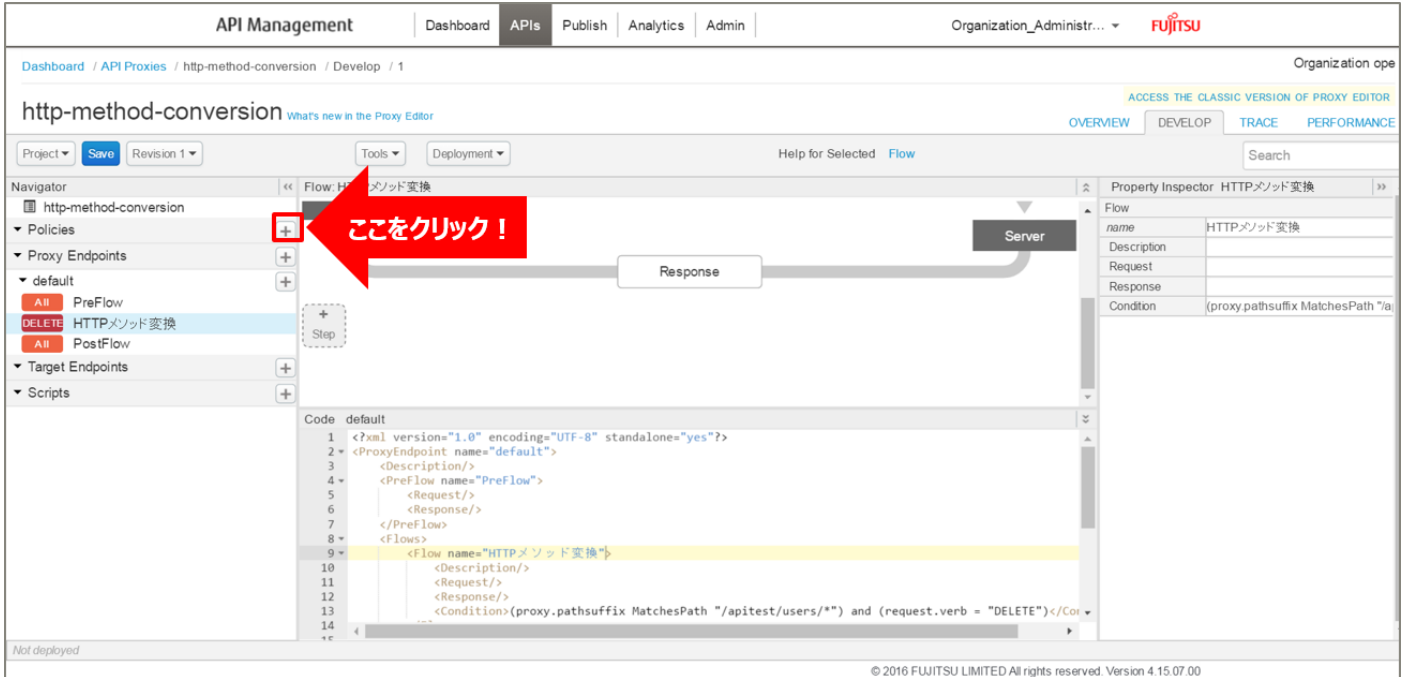
- Flow Name : HTTP メソッド変換 (任意の名前)
- Condition Type : 「Path and Verb」を選択
- Path : /apitest/users/\* (任意のパス)  
※/apitest/users/\* の様に動的に変わる値には\*を指定します
- Verb : 「DELETE」を選択



## 2) Extract Variables ポリシーの作成 (ユーザー ID の抽出)

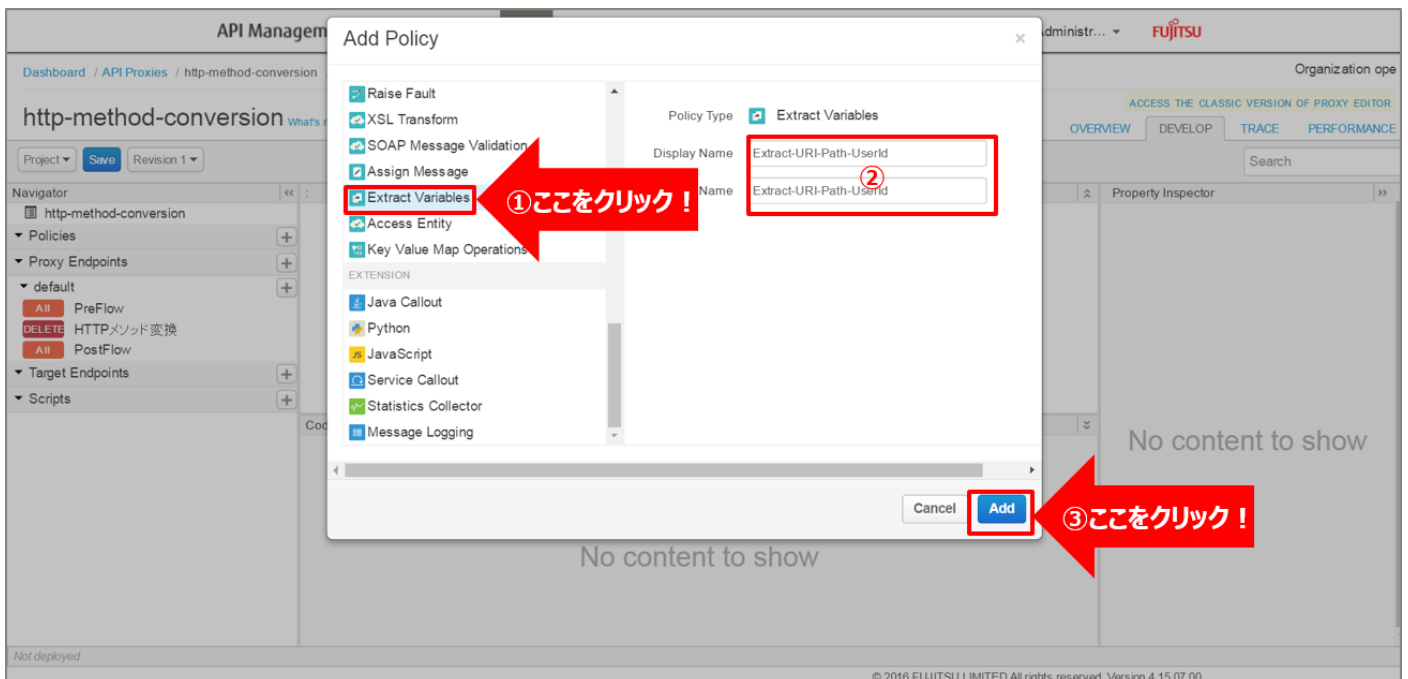
指定された URI のパスから、パターン (ユーザー ID) に一致する値を抽出して変数に格納する Extract Variables ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Extract Variables」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : Extract-URI-Path-UserId (任意の名前)
- Name : Extract-URI-Path-UserId (任意の名前)



追加した Extract Variables ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

**ExtractVariables-URIPath.xml**

URIからパラメータを取得

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true" name="Extract-URI-Path-UserId">
  <DisplayName>Extract-URI-Path-UserId</DisplayName>
  <Properties/>
  <URIPath name="name">
    <Pattern ignoreCase="false">/apitest/users/{userId}</Pattern>
  </URIPath>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <Source clearPayload="false">request</Source>
  <VariablePrefix>urirequest</VariablePrefix>
</ExtractVariables>
```

ここをクリック!

#### 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true" name="Extract-URI-Path-UserId">
  <DisplayName>Extract-URI-Path-UserId</DisplayName>
  <Properties/>
  <URIPath name="name">
    <Pattern ignoreCase="false">/apitest/users/{userId}</Pattern>
  </URIPath>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <Source clearPayload="false">request</Source>
  <VariablePrefix>urirequest</VariablePrefix>
</ExtractVariables>
```

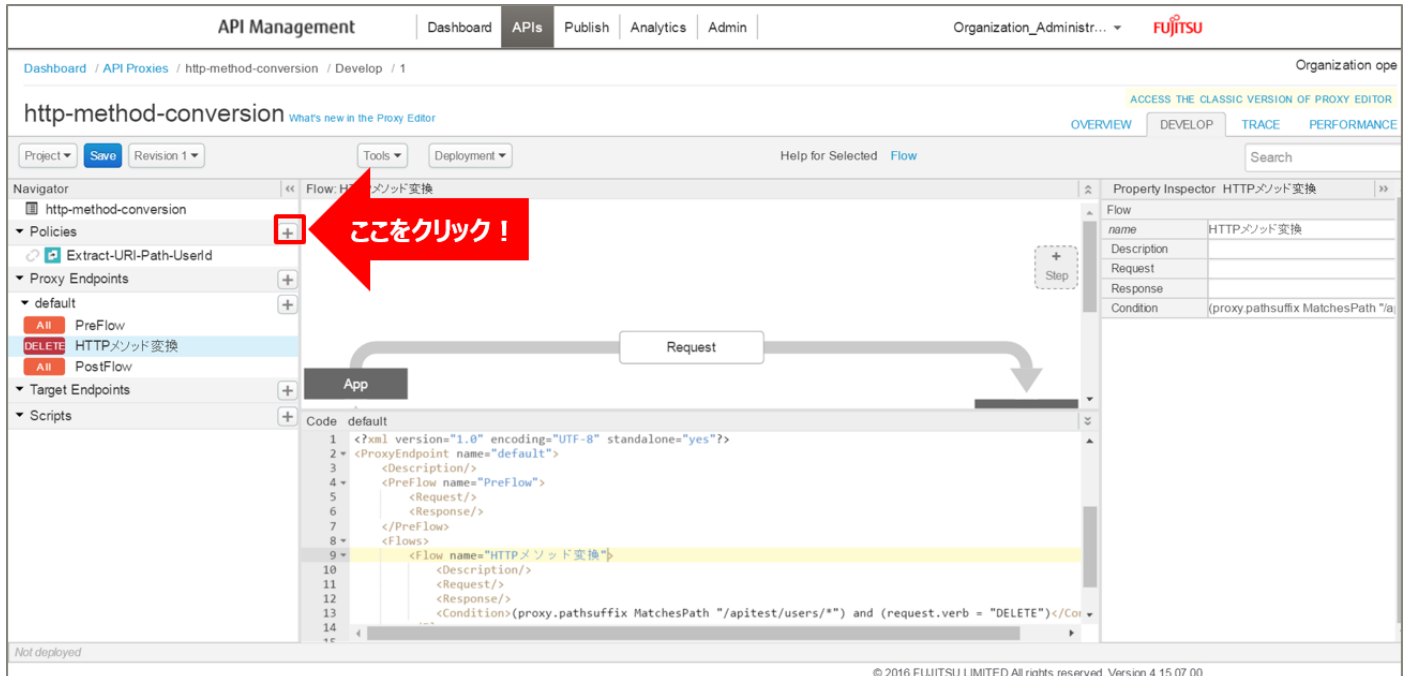
※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.4. Extract Variables XML 仕様](#)」をご参照ください。

### 3) Service Callout ポリシーの作成（ユーザー情報の削除・取得）

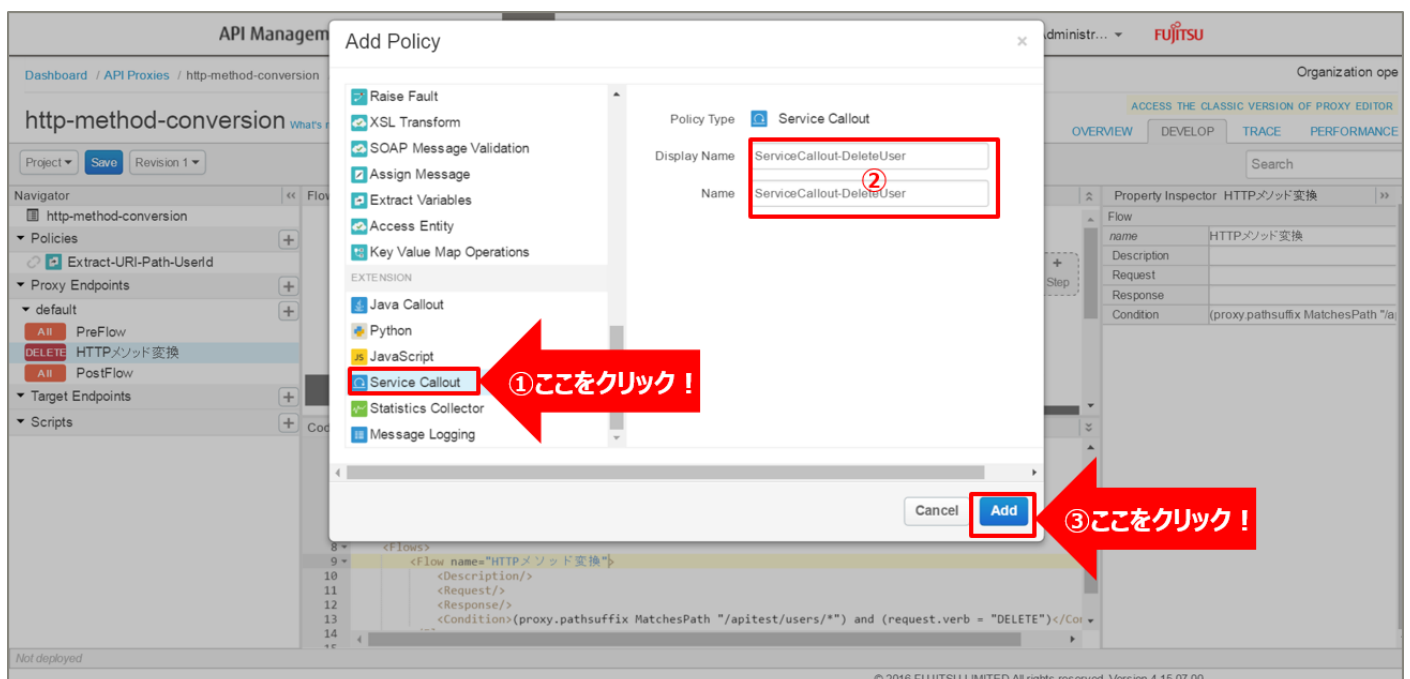
ユーザー情報を削除する API を呼び出し、削除したユーザー情報を取得する Service Callout ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Service Callout」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : ServiceCallout-DeleteUser (任意の名前)
- Name : ServiceCallout-DeleteUser (任意の名前)



追加した Service Callout ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceCallout async="false" continueOnError="true" enabled="true" name="ServiceCallout-DeleteUser">
  <DisplayName>ServiceCallout-DeleteUser</DisplayName>
  <Properties/>
  <Request clearPayload="true" variable="myRequest">
    <Set>
      <QueryParams>
        <QueryParam name="userId">{urirequest.userId}</QueryParam>
      </QueryParams>
      <Verb>GET</Verb>
    </Set>
  </Request>
  <Response>calloutResponse</Response>
  <HTTPTargetConnection>
    <URL>https://www.exemple.com/sensing_webapi/api/apitest/query/deleteuser</URL>
  </HTTPTargetConnection>
</ServiceCallout>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A3.1. Service Callout XML 仕様](#)」をご参照ください。

#### 4) Extract Variables ポリシーの作成（ユーザー情報の変数化）

2) と同様の手順で、Service Callout ポリシーで取得した通り一情報を変数として格納（変数化）する Extract Variables ポリシーを追加します。

以下は Add Policy で入力するポリシーの情報の入力例です。

- Display Name : Extract-ServiceCallout（任意の名前）
- Name : Extract-ServiceCallout（任意の名前）

追加した Extract Variables ポリシーを選択し、ポリシー編集画面を表示します。

必要に応じてポリシーの定義を編集します。

**ExtractVariables-ServiceCallout.xml**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true" name="Extract-ServiceCallOut">
  <DisplayName>Extract-ServiceCallOut</DisplayName>
  <Properties/>
  <JSONPayload>
    <Variable name="object">
      <JSONPath>$/JSONPath</JSONPath>
    </Variable>
  </JSONPayload>
  <Source clearPayload="false">calloutResponse</Source>
  <VariablePrefix>callout</VariablePrefix>
</ExtractVariables>
```

<JSONPayload>  
 <Variable name="object">  
 <JSONPath>\$/JSONPath</JSONPath>  
 </Variable>  
</JSONPayload>

**JSON全体を変数へ格納。**

#### 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true" name="Extract-ServiceCallOut">
  <DisplayName>Extract-ServiceCallOut</DisplayName>
  <Properties/>
  <JSONPayload>
    <Variable name="object">
      <JSONPath>$/JSONPath</JSONPath>
    </Variable>
  </JSONPayload>
  <Source clearPayload="false">calloutResponse</Source>
  <VariablePrefix>callout</VariablePrefix>
</ExtractVariables>
```

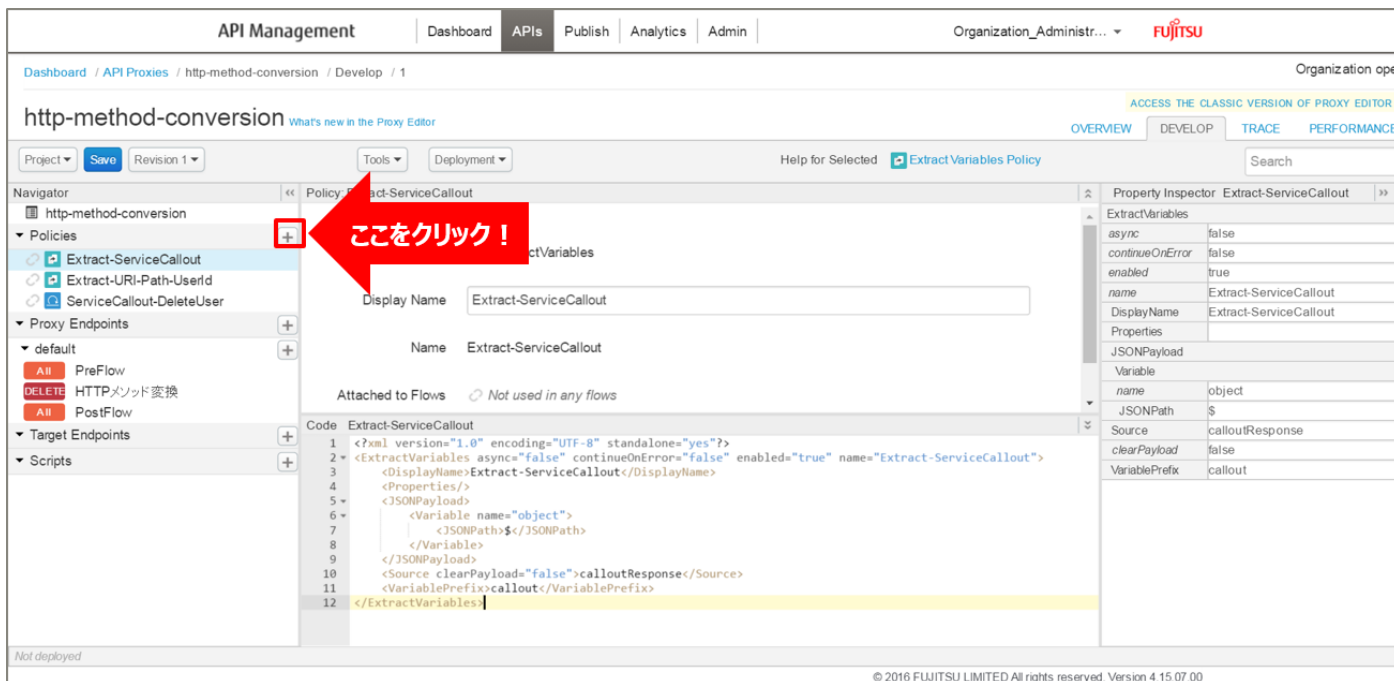
※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.4. Extract Variables XML 仕様](#)」をご参照ください。

## 5) Assign Message ポリシーの作成 (HTTP レスポンスの作成)

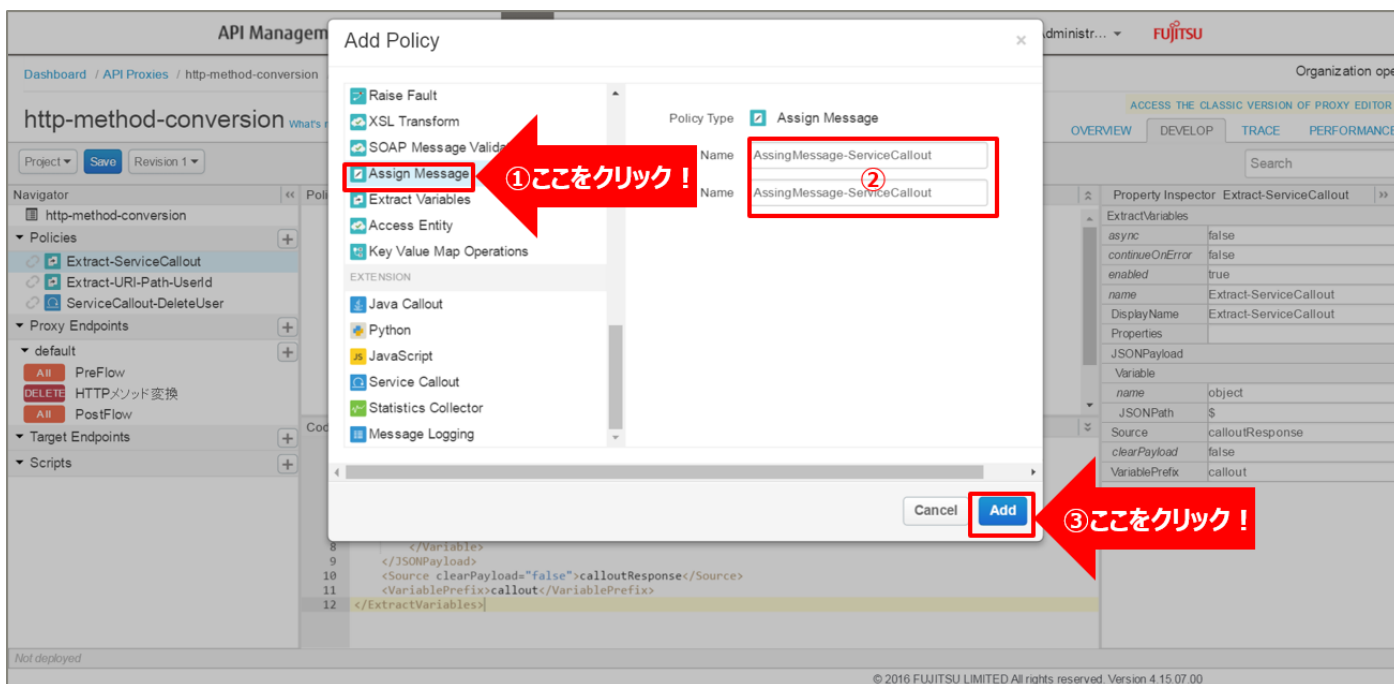
HTTP メッセージを作成する Assign Message ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Assign Message」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。  
以下は入力例です。

- Display Name : AssingMessage-ServiceCallout (任意の名前)
- Name : AssingMessage-ServiceCallout (任意の名前)





追加した Assign Message ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true" name="AssignMessage-ServiceCallout">
  <DisplayName>AssignMessage-ServiceCallout</DisplayName>
  <Properties/>
  <AssignTo createNew="false" transport="http" type="response"/>
  <Copy source="calloutResponse">
    <StatusCode>true</StatusCode>
  </Copy>
  <Set>
    <Payload contentType="application/json;charset=utf-8" variablePrefix="@ " variableSuffix="#">
      @callout.object#
    </Payload>
  </Set>
</AssignMessage>

```

#### 【定義例】

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true" name="AssignMessage-ServiceCallout">
  <DisplayName>AssignMessage-ServiceCallout</DisplayName>
  <Properties/>
  <AssignTo createNew="false" transport="http" type="response"/>
  <Copy source="calloutResponse">
    <StatusCode>true</StatusCode>
  </Copy>
  <Set>
    <Payload contentType="application/json;charset=utf-8" variablePrefix="@ "
    variableSuffix="#">
      @callout.object#
    </Payload>
  </Set>
</AssignMessage>

```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.3. Assign Message XML 仕様](#)」をご参照ください。

## 6) ポリシーの配置

以下の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。

The screenshot shows the API Management console interface. The top navigation bar includes 'Dashboard', 'APIs', 'Publish', 'Analytics', and 'Admin'. The current page is 'http-method-conversion / Develop / 1'. The main area displays the 'Flow: HTTPメソッド変換' configuration. The 'Policies' list on the left includes 'AssignMessage-ServiceCallout', 'Extract-ServiceCallout', 'Extract-URI-Path-UserId', and 'ServiceCallout-DeleteUser'. The 'PreFlow' step is selected, and the 'HTTPメソッド変換' policy is being configured. The flow diagram shows a 'Request' step leading to a 'Server' step, and a 'Response' step leading back to the 'Request' step. The 'Assign Message-ServiceCallout' policy is being dragged into the 'Response' step. The 'Save' button is highlighted in the top left corner.

④ここをクリック！

②ここにドラッグ！

- Extract-ServiceCallout
- ServiceCallout-DeleteUser
- Extract-URI-Path-UserId

①ここをクリック！

③ここにドラッグ！

● Assign Message-ServiceCallout

Not deployed

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.00

## 7) 動作確認

作成した API Proxy の動作確認を行います。

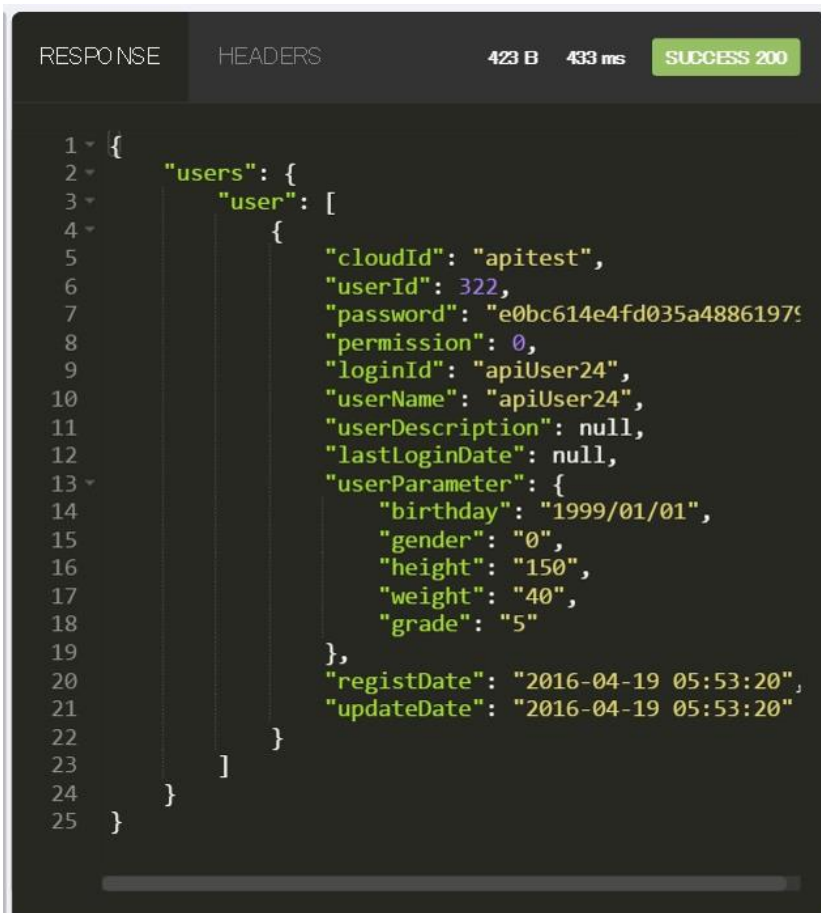
以下の例のように、URL へアクセスすると指定したユーザーが削除され、削除したユーザー情報が JSON 形式で返却されます。

http://{FQDN}:10080/http-method-conversion/apitest/users/{UserID}

※http プロトコルでの動作確認例です（ポート番号は 10080 です）。

※{FQDN}: API Proxy の作成時に生成された URL のホスト名（FQDN）を指定します。

※{UserID} は、任意のユーザー ID に置き換えてください。



```
RESPONSE HEADERS 423 B 433 ms SUCCESS 200
1 {
2   "users": {
3     "user": [
4       {
5         "cloudId": "apitest",
6         "userId": 322,
7         "password": "e0bc614e4fd035a4886197",
8         "permission": 0,
9         "loginId": "apiUser24",
10        "userName": "apiUser24",
11        "userDescription": null,
12        "lastLoginDate": null,
13        "userParameter": {
14          "birthday": "1999/01/01",
15          "gender": "0",
16          "height": "150",
17          "weight": "40",
18          "grade": "5"
19        },
20        "registDate": "2016-04-19 05:53:20",
21        "updateDate": "2016-04-19 05:53:20"
22      }
23    ]
24  }
25 }
```

## 3.2. リクエスト Body 変換

### 3.2.1. ユースケース概要

バックエンドサービスを呼び出す前に、リクエストの変換を行う API Proxy を実装します。  
ここでは、2つのユースケースを説明します。

#### 【ユースケース 1：XML → JSON 変換】

バックエンドサービスとして、ユーザー情報を登録する API を使用します。  
XML 形式のユーザー情報を、JSON 形式に変換してから API を呼び出すようにします。  
レスポンスとして、登録したユーザー情報が JSON 形式で返却されます。

リクエストパラメータは、以下のような形式を想定しています。

#### 変換前のリクエストパラメータ

```
<?xml version="1.0" encoding="UTF-8"?>
<USER>
  <cloudId>apitest</cloudId>
  <password>6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b</password>
  <permission>0</permission>
  <loginId>apiUser27</loginId>
  <userName>apiUser27</userName>
  <userDescription>test</userDescription>
  <lastLoginDate/>
  <birthday>1999/01/01</birthday>
  <gender>0</gender>
  <height>150</height>
  <weight>40</weight>
  <grade>5</grade>
</USER>
```

#### 変換後のリクエストパラメータ

```
{
  "cloudId": "apitest",
  "password": "6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b",
  "permission": 0,
  "loginId": "apiUser27",
  "userName": "apiUser27",
  "userDescription": "test",
  "lastLoginDate": {},
  "birthday": "1999/01/01",
  "gender": 0,
  "height": 150,
  "weight": 40,
  "grade": 5
}
```

API Proxy の実装フローは以下の通りです。

- 1) API Proxy の作成
  - 1-1) API Proxy の作成
  - 1-2) Conditional Flow (POST) の作成
- 2) XML to JSON ポリシーの作成 (リクエスト情報の変換)
- 3) Extract Variables ポリシーの作成 (ユーザー情報の変数化)
- 4) Assign Message ポリシーの作成 (HTTP リクエストの作成)
- 5) ポリシーの配置

## 6) 動作確認

使用する Policy は以下の通りです。

- XML to JSON ポリシー
- Assign Message ポリシー
- Extract Variables ポリシー

【ユースケース 2：JSON → JSON 変換（キー変更）】

ユースケース 1 で作成した API Proxy を利用します。

JSON 形式のリクエストパラメーターの一部を変更（パラメーターキー変換）してから、バックエンドサービスを呼び出すよう変更します。

レスポンスとして、登録したユーザー情報が JSON 形式で返却されます。

リクエストパラメーターは、以下のような形式を想定しています。

変換前のリクエストパラメータ

```
{
  "cloudId": "apitest",
  "password": "6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b",
  "permission": 0,
  "loginId": "apiUser28",
  "userName": "apiUser28",
  "userDescription": "test",
  "lastLoginDate": {},
  "birthday": "1999/01/01",
  "gender": 0,
  "height": 150,
  "weight": 40,
  "grade": 5
}
```

変換後のリクエストパラメータ

```
{
  "site": "apitest",
  "password": "6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b",
  "permission": 0,
  "loginId": "apiUser28",
  "userName": "apiUser28",
  "userDescription": "test",
  "lastLoginDate": {},
  "birthday": "1999/01/01",
  "gender": 0,
  "height": 150,
  "weight": 40,
  "grade": 5
}
```

API Proxy の実装フローは以下の通りです。

- 1) Extract Variables ポリシーの作成（ユーザー情報の変数化）
- 2) Assign Message ポリシーの作成（HTTP リクエストの作成）
- 3) ポリシーの解除
- 4) ポリシーの配置
- 5) 動作確認

使用する Policy は以下の通りです。

- Assign Message ポリシー
- Extract Variables ポリシー

### 3.2.2. 手順

#### 3.2.2.1. XML → JSON 変換

##### 1) API Proxy の作成

API Proxy を作成します。

##### 1-1) API Proxy の作成

画面上部の「APIs」メニューをクリックし、API Proxies 画面に遷移します。

The screenshot shows the API Management console interface. At the top, there is a navigation bar with 'API Management' on the left and 'Organization Administr...' and 'FUJITSU' on the right. In the center of the navigation bar, the 'APIs' menu item is highlighted with a red box, and a red arrow points to it with the text 'ここをクリック!' (Click here!). Below the navigation bar, there is a 'Proxy Traffic' section with a date range selector set to 'From Fri Mar 10 2017, 10:00 am To Fri Mar 17 2017, 10:00 am' and a 'By Hour' dropdown. The main content area displays 'No traffic in the selected date range.' Below this, there are two sections: 'Developer Engagement' and 'Developer Apps', both showing 'No traffic in the selected date range.' The 'Developer Engagement' section has a 'Traffic Level' indicator with a blue bar.

API Proxies 画面で、「+ API Proxy」ボタンをクリックします。

The screenshot shows the 'API Proxies' page in the API Management console. At the top, there are navigation tabs: 'Dashboard', 'APIs', 'Publish', 'Analytics', and 'Admin'. The 'APIs' tab is selected. The page title is 'API Proxies' and the organization is 'Organization Administr... FUJITSU'. Below the title, there are tabs for 'List' and 'Analytics'. A search bar is present with a dropdown menu set to 'All'. A table displays the list of API proxies with columns for 'API Proxy', 'Environments', 'Traffic', 'Message Trend by Hour', 'Avg Time', 'Error Rate', 'Modified', and 'Actions'. The table contains three entries: 'handson-api20160217020', 'Hello-World-Nodejs-2', and 'Hello-World-Nodejs'. A red arrow points to the '+ API Proxy' button in the top right corner of the table area.

API Proxy	Environments	Metrics for Last 24 Hours (prod)				Modified	Actions
		Traffic	Message Trend by Hour	Avg Time	Error Rate		
handson-api20160217020	test	1	↓	8029.00 ms	100.00 %	3 hours ago	<a href="#">x Delete</a> <a href="#">Roles (1)</a>
Hello-World-Nodejs-2	test, prod	3	↓	37.00 ms	0.00 %	19 hours ago	<a href="#">x Delete</a> <a href="#">Roles (1)</a>
Hello-World-Nodejs	test, prod	0				a day ago	<a href="#">x Delete</a> <a href="#">Roles</a>

➤ Build a Proxy (TYPE)

「Reverse proxy (most common)」を選択し、「Next」ボタンをクリックします。

The screenshot shows the 'Build a Proxy' form. The 'Reverse proxy (most common)' option is selected and highlighted with a red box. Below it, there is a checkbox for 'Use OpenAPI' with the text 'Optionally associate the proxy with an OpenAPI (Swagger) document'. There are four other options: 'Node.js App', 'SOAP service', 'No Target', and 'Proxy bundle'. At the bottom of the form, there is an 'Exit Without Saving' button and a 'Next' button. A red arrow points to the 'Next' button.

Build a Proxy

- Reverse proxy (most common)  
Route inbound requests to backend services.
- Node.js App  
Create a new app in JavaScript and optionally add policies.
- SOAP service  
Create a RESTful or pass-through proxy for a SOAP service.
- No Target  
Create a simple API proxy that does not route to any backend target.
- Proxy bundle  
Import an existing proxy from a zip archive.

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

➤ Build a Proxy (DETAILS)

API Proxy の情報を入力し、「Next」ボタンをクリックします。以下は入力例です。

- Proxy Name : requestbody-conversion (任意の名前)
- Proxy Base Path : /requestbody-conversion (任意のパス (自動入力))
- Existing API : https://www.example.com (任意の API)

### Build a Proxy

TYPE > **DETAILS** > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Specify the proxy details.

Proxy Name \*   
Valid characters are letters, numbers, dash (-), and underscore (\_).

Proxy Base Path \*   
A path component that uniquely identifies this API proxy. The public-facing URL of this API proxy is comprised of your organization name, an environment where this API proxy is deployed, and this Proxy Base Path. Example URL http://test-sandbox-test.apigee.net/requestbody-conversion

Existing API \*   
Defines the target URL invoked on behalf of this API proxy. Any URL that is accessible over the open Internet can be used. Example: https://api.usergrid.com

Description

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

**ここをクリック!**



➤ Build a Proxy (SECURITY)

Authentication : 「Pass through (none)」 を選択し、「Next」 ボタンをクリックします。

### Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Secure access for users and clients.

Authorization  Pass through (none)  
 API Key  
 OAuth 2.0

Browser  Add CORS headers

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

[Previous](#) [Exit Without Saving](#) **ここをクリック!** [Next](#)

➤ Build a Proxy (VIRTUAL HOSTS)

設定を変えずに「Next」ボタンをクリックします。

Build a Proxy

TYPE DETAILS SECURITY VIRTUAL HOSTS BUILD SUMMARY

Select the virtual hosts this proxy will bind to when it is deployed. You must select at least one virtual host. [Learn more...](#)

<input checked="" type="checkbox"/>	Name	Environment	Host Aliases
<input checked="" type="checkbox"/>	default	prod	http://:10080
		test	http://:10080
<input checked="" type="checkbox"/>	secure	prod	https://:10443
		test	https://:10443

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Next

➤ Build a Proxy (BUILD)

設定を変えずに「Build and Deploy」ボタンをクリックします。

Build a Proxy

TYPE DETAILS SECURITY VIRTUAL HOSTS BUILD SUMMARY

You are ready to build and deploy your API proxy.

Deploy Environments  prod  test

Proxy Name requestbody-conversion

Proxy Type Reverse proxy

Virtual Hosts default, secure

Security None

Browser Do not allow direct requests from a browser via CORS

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Build and Deploy

➤ Build a Proxy (SUMMARY)

API Proxy の作成が完了したら、API Proxy のリンクをクリックします。

Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

- ✓ Generated proxy
- ✓ Uploaded proxy
- ✓ Deployed to test

ここをクリック！ [requestbody-conversion](#) proxy in the editor .

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

## 1-2) Conditional Flow (POST) の作成

バックエンドサービスに対するリソースパスと処理 (HTTP Method) の定義を行います。

クライアントからのリクエストがここで定義したパターンと一致する場合に、以降の手順で設定する処理を実行します。

「Develop」タブをクリックし、API proxy editor を開きます。

The screenshot shows the 'API Management' console for the 'requestbody-conversion' proxy. The 'DEVELOP' tab is highlighted with a red box and a red arrow pointing to it with the text 'ここをクリック!' (Click here!). The interface includes a navigation menu, a revision summary, a deployments table, proxy endpoints, and target endpoints.

Environment	Revision	Status	URL
test	1	●	http://[redacted]

Name	Base Path	Target Endpoints
default	/requestbody-conversion	default

Name	Target	Used by Proxy Endpoints
default	URL: https://www.example.com	default

default の「+」ボタンをクリックし、New Conditional Flow を開きます。

The screenshot shows the 'API Management' console for the 'requestbody-conversion' proxy in the 'Develop' view. A red arrow points to the '+' button next to the 'default' proxy endpoint in the left-hand 'Navigator' pane. The main area shows a flow diagram with a 'Request' step and an 'App' component. The XML code for the proxy is visible at the bottom.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ProxyEndpoint name="default">
3   <Description/>
4   <PreFlow name="PreFlow">
5     <Request/>
6   </PreFlow>
7 </ProxyEndpoint>
8 <Flows/>
9 <PostFlow name="PostFlow">
10  <Request/>
11  <Response/>
12 </PostFlow>
13 <HTTPProxyConnection>
14   <BasePath>/requestbody-conversion</BasePath>
15 </HTTPProxyConnection>
```

Flow の情報を入力し、「Add」 ボタンをクリックします。以下は入力例です。

- Flow Name : リクエスト Body 変換 (任意の名前)
- Condition Type : 「Path and Verb」 を選択
- Path : /sensing\_webapi/api/apitest/users (任意のパス)
- Verb : 「POST」 を選択

The screenshot displays the 'New Conditional Flow' dialog box in the API Management console. The dialog is used to create a new conditional flow. The fields are filled with the following information:

- Flow Name: リクエストBody変換 (1)
- Description: (empty)
- Condition Type: Path and Verb (2)
- Path: /sensing\_webapi/api/apitest/users (3)
- Verb: POST (4)
- Optional Target URL: (empty)

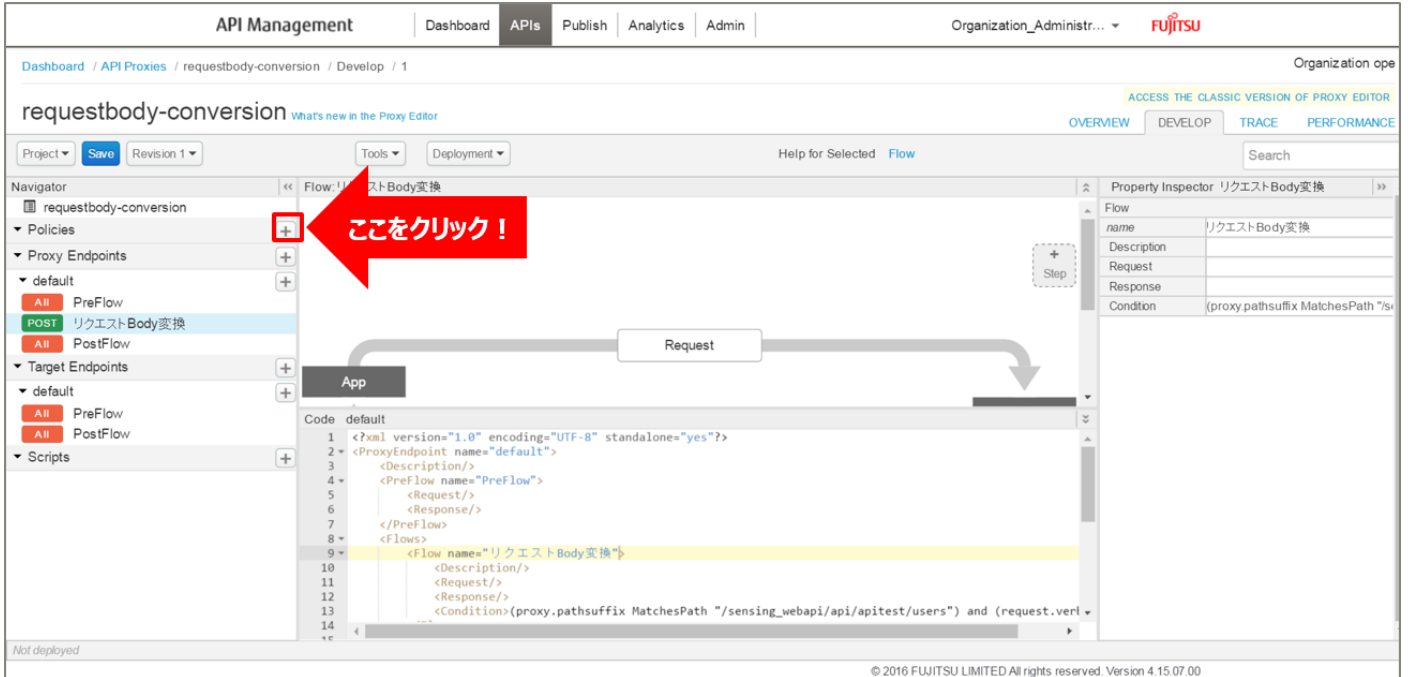
The 'Add' button is highlighted with a red arrow and the text '⑤ここをクリック!' (5 Click here!).

```
Code default
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ProxyEndpoint name="default">
3   <Description/>
4   <PreFlow name="PreFlow">
5     <Request/>
6     <Response/>
7   </PreFlow>
8   <Flows/>
9   <PostFlow name="PostFlow">
10    <Request/>
11    <Response/>
12  </PostFlow>
13 </HTTPProxyConnection>
14 <BasePath>/requestbody-conversion</BasePath>
15
```

## 2) XML to JSON ポリシーの作成（リクエスト情報の変換）

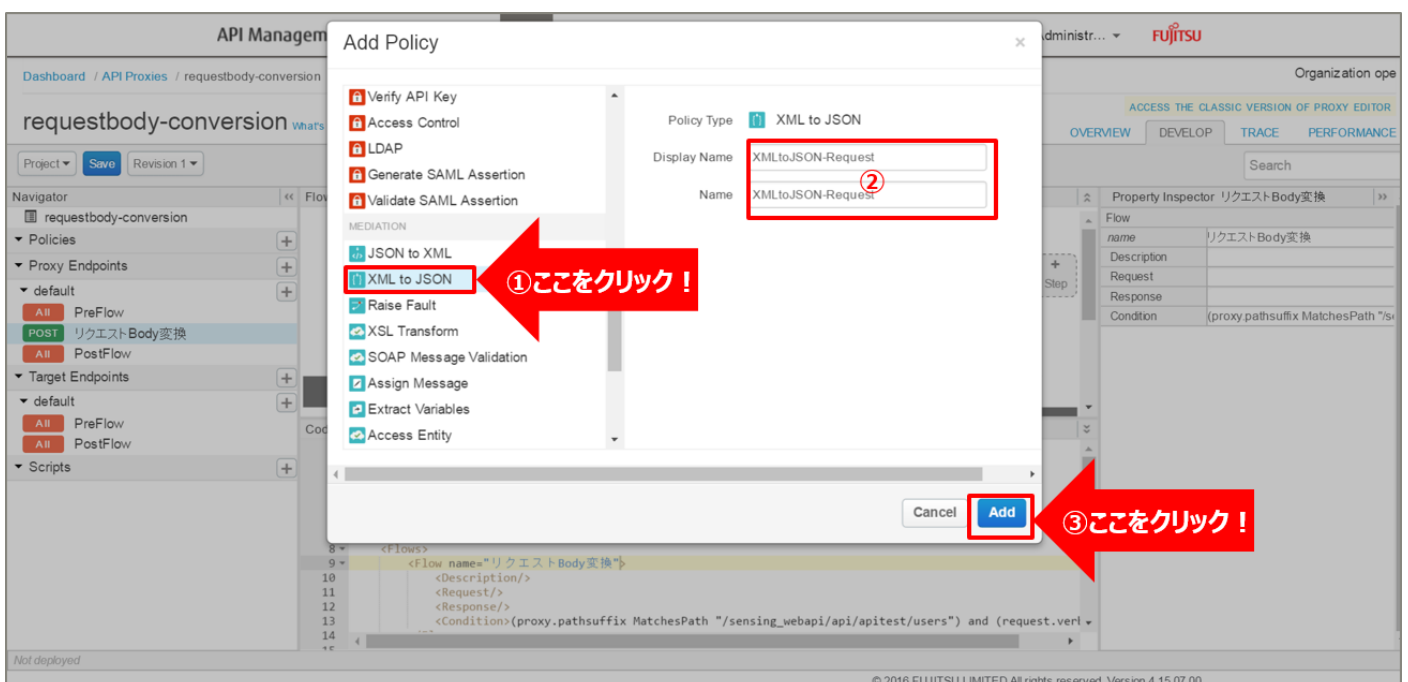
リクエスト情報やレスポンス情報を、XML 形式から JSON 形式に変換する XML to JSON ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「XML to JSON」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : XMLtoJSON-Request (任意の名前)
- Name : XMLtoJSON-Request (任意の名前)



追加した XML to JSON ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

The screenshot shows the API Management interface for editing a policy named 'XMLtoJSON-Request'. The left sidebar shows a tree view with 'Policies' expanded, and 'XMLtoJSON-Request' selected. A red arrow points to this selection with the text 'ここをクリック!'. The main editor area shows the XML configuration for the policy, which is highlighted with a red box. The XML code is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<XMLToJSON async="false" continueOnError="false" enabled="true" name="XMLtoJSON-Request">
  <DisplayName>XMLtoJSON-Request</DisplayName>
  <Properties/>
  <OutputVariable>request</OutputVariable>
  <Source>request</Source>
  <Options>
    <RecognizeNull>true</RecognizeNull>
  </Options>
</XMLToJSON>
```

#### 【定義例】

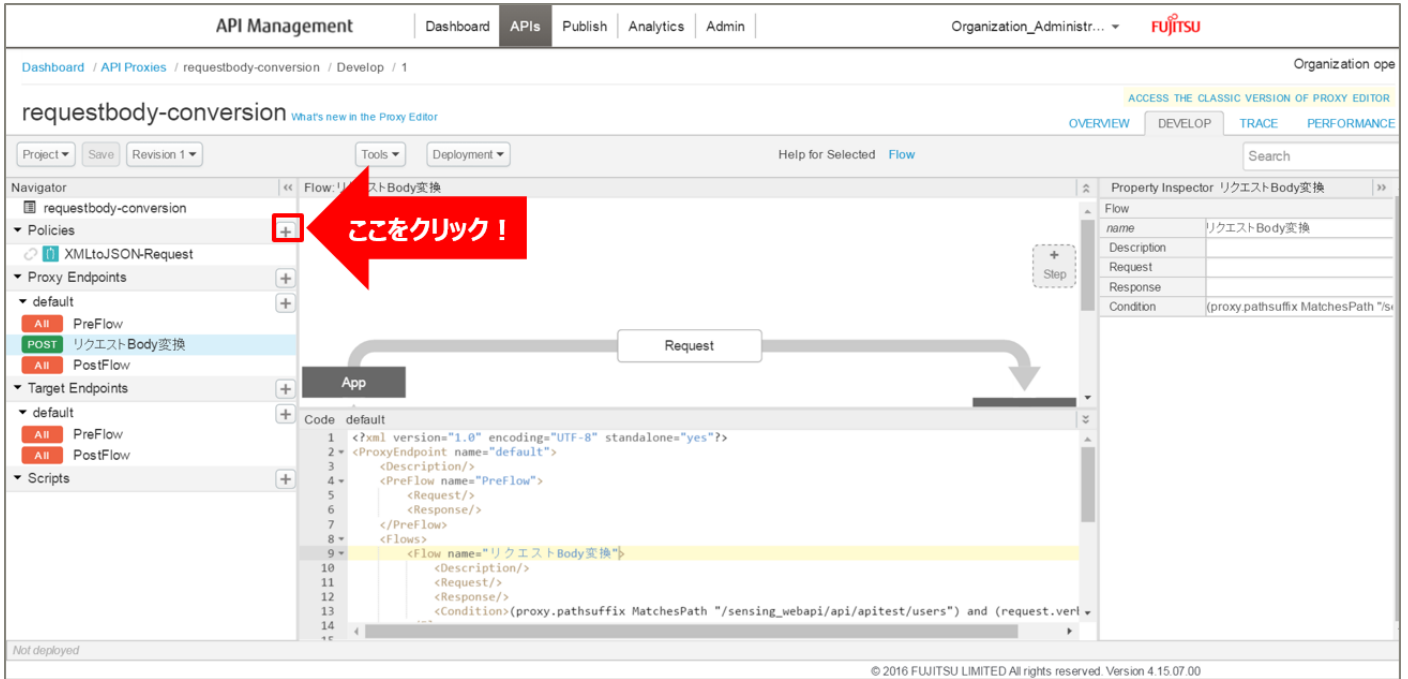
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<XMLToJSON async="false" continueOnError="false" enabled="true" name="XMLtoJSON-Request">
  <DisplayName>XMLtoJSON-Request</DisplayName>
  <Properties/>
  <OutputVariable>request</OutputVariable>
  <Source>request</Source>
  <Options>
    <RecognizeNull>true</RecognizeNull>
  </Options>
</XMLToJSON>
```

※定義内容の詳細は、「[A2.2. XML to JSON XML 仕様](#)」をご参照ください。

### 3) Extract Variables ポリシーの作成（ユーザー情報の変数化）

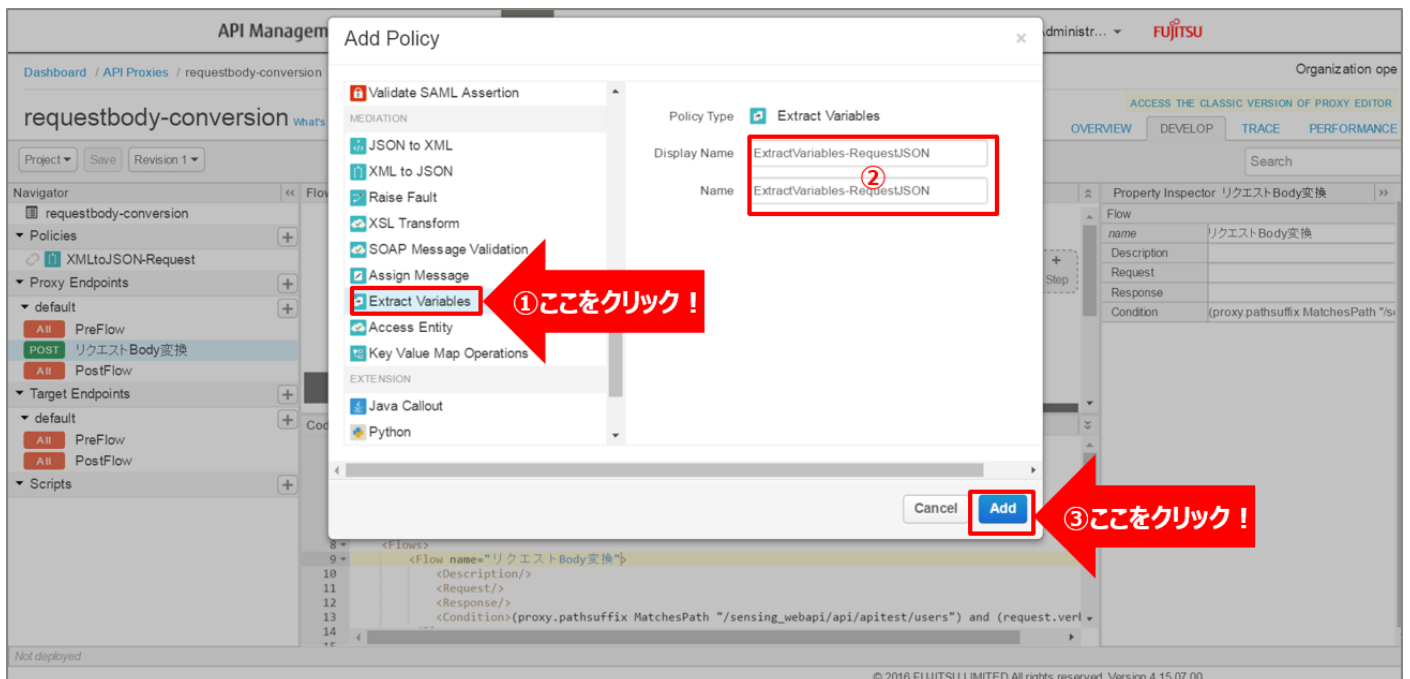
JSON 形式に変換したリクエスト情報からユーザー情報を抽出して変数として格納（変数化）する Extract Variables ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Extract Variables」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : ExtractVariables-RequestJSON (任意の名前)
- Name : ExtractVariables-RequestJSON (任意の名前)





追加した Extract Variables ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

**ExtractVariables-RequestJSON.xml**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true" name="ExtractVariables-RequestJSON">
  <DisplayName>ExtractVariables-RequestJSON</DisplayName>
  <Properties/>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <JSONPayload>
    <Variable name="USER" type="string">
      <JSONPath>$.USER</JSONPath>
    </Variable>
  </JSONPayload>
  <Source clearPayload="false">request</Source>
  <VariablePrefix>jsonrequest</VariablePrefix>
</ExtractVariables>
```

**USER配下をまとめて変数に格納**

ここをクリック!

#### 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true"
name="ExtractVariables-RequestJSON">
  <DisplayName> ExtractVariables-RequestJSON</DisplayName>
  <Properties/>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <JSONPayload>
    <Variable name="USER" type="string">
      <JSONPath>$.USER</JSONPath>
    </Variable>
  </JSONPayload>
  <Source clearPayload="false">request</Source>
  <VariablePrefix>jsonrequest</VariablePrefix>
</ExtractVariables>
```

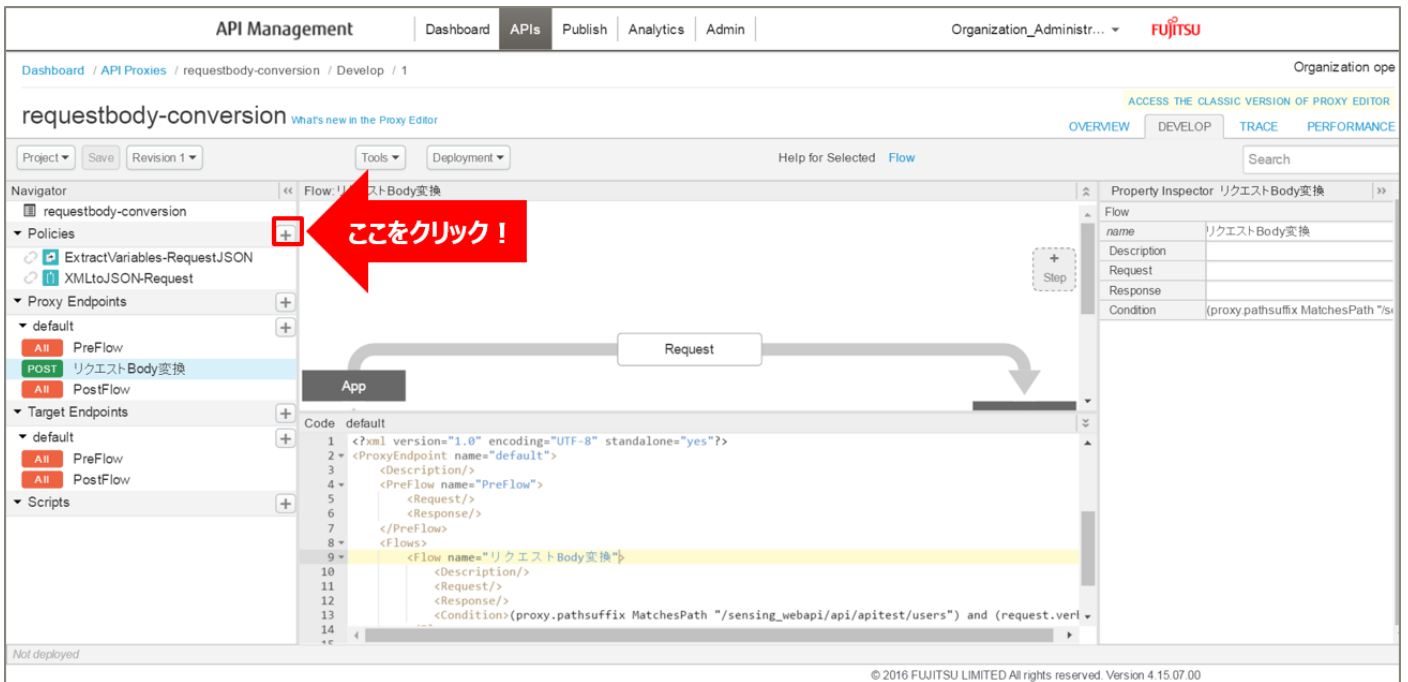
※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.4. Extract Variables XML 仕様](#)」をご参照ください。

#### 4) Assign Message ポリシーの作成 (HTTP リクエストの作成)

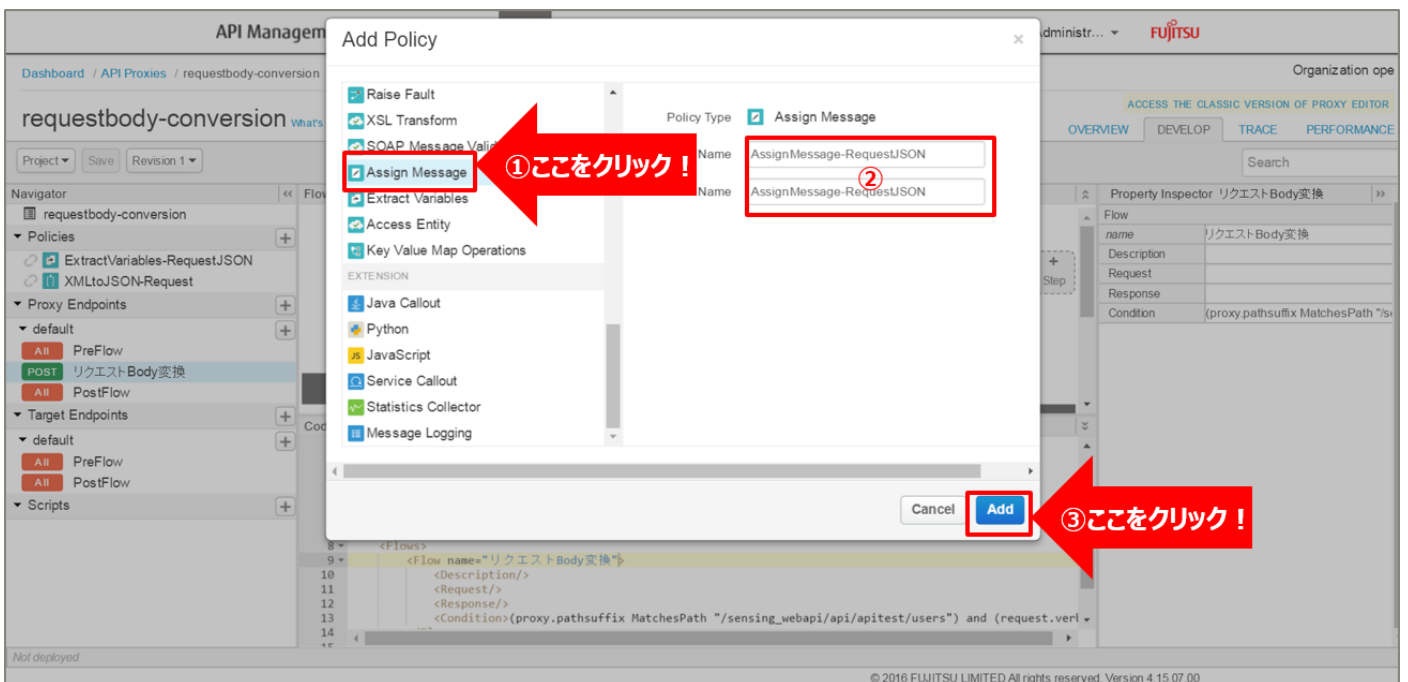
HTTP メッセージを作成する Assign Message ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Assign Message」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : AssignMessage-RequestJSON (任意の名前)
- Name : AssignMessage-RequestJSON (任意の名前)



追加した Assign Message ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

**AssignMessage-RequestJSON.xml**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true" name="AssignMessage-RequestJSON">
  <DisplayName>AssignMessage-RequestJSON</DisplayName>
  <Properties/>
  <Set>
    <Payload contentType="application/json" variablePrefix="@ " variableSuffix="#">
      @jsonrequest.USER#
    </Payload>
    <Verb>POST</Verb>
  </Set>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="request"/>
</AssignMessage>
```

登録するJSONデータをセット

ここをクリック!

Attached to Flows Not used in any flows

Code AssignMessage-RequestJSON

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <AssignMessage async="false" continueOnError="false" enabled="true" name="AssignMessage-RequestJSON">
3   <DisplayName>AssignMessage-RequestJSON</DisplayName>
4   <Properties/>
5   <Copy source="request">
6     <Headers/>
7     <QueryParams/>
8     <FormParams/>
9     <Payload/>
10    <Verb/>
11    <StatusCode/>
12    <ReasonPhrase/>
13    <Path/>
14  </Copy>
```

Copy

source	request
Headers	
QueryParams	
FormParams	
Payload	
Verb	
StatusCode	
ReasonPhrase	
Path	
Remove	
Headers	
Header	
name	h1
QueryParams	

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.00

#### 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true" name="AssignMessage-RequestJSON">
  <DisplayName>AssignMessage-RequestJSON</DisplayName>
  <Properties/>
  <Set>
    <Payload contentType="application/json" variablePrefix="@ " variableSuffix="#">
      @jsonrequest.USER#
    </Payload>
    <Verb>POST</Verb>
  </Set>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="request"/>
</AssignMessage>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.3. Assign Message XML 仕様](#)」をご参照ください。

## 5) ポリシーの配置

以下の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。

The screenshot displays the API Management console for a proxy named 'requestbody-conversion'. The flow 'リクエストBody変換' is shown with a sequence of steps: Request, XMLtoJSON-Request, ExtractVariables-RequestJSON, AssignMessage-RequestJSON, and Server. The 'POST リクエストBody変換' policy is highlighted in the Proxy Endpoints list. The 'Save' button is located in the top left corner of the editor.

③ここをクリック！

②ここにドラッグ！

①ここをクリック！

Not deployed

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.00

## 6) 動作確認

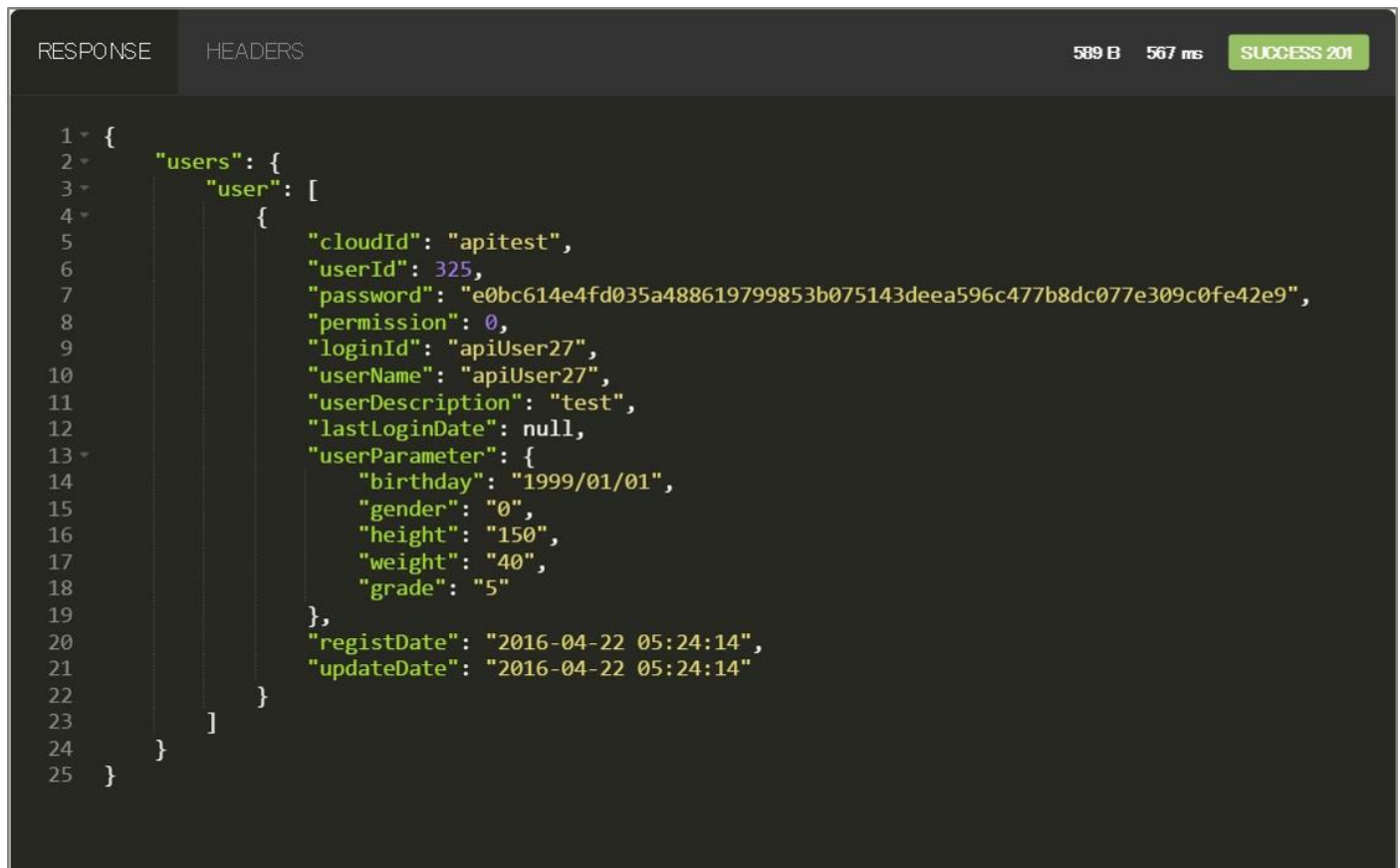
作成した API Proxy の動作確認を行います。

以下の例のように、URL へアクセスするとユーザー情報が登録され、登録したユーザー情報が JSON 形式で返却されます。

http://{FQDN}:10080/requestbody-conversion/sensing\_webapi/api/apitest/users

※http プロトコルでの動作確認例です（ポート番号は 10080 です）。

※{FQDN}: API Proxy の作成時に生成された URL のホスト名（FQDN）を指定します。



```
RESPONSE HEADERS 589 B 567 ms SUCCESS 201
1 {
2   "users": {
3     "user": [
4       {
5         "cloudId": "apitest",
6         "userId": 325,
7         "password": "e0bc614e4fd035a488619799853b075143deea596c477b8dc077e309c0fe42e9",
8         "permission": 0,
9         "loginId": "apiUser27",
10        "userName": "apiUser27",
11        "userDescription": "test",
12        "lastLoginDate": null,
13        "userParameter": {
14          "birthday": "1999/01/01",
15          "gender": "0",
16          "height": "150",
17          "weight": "40",
18          "grade": "5"
19        },
20        "registDate": "2016-04-22 05:24:14",
21        "updateDate": "2016-04-22 05:24:14"
22      }
23    ]
24  }
25 }
```

### 3.2.2.2. JSON → JSON 変換（キー変更）

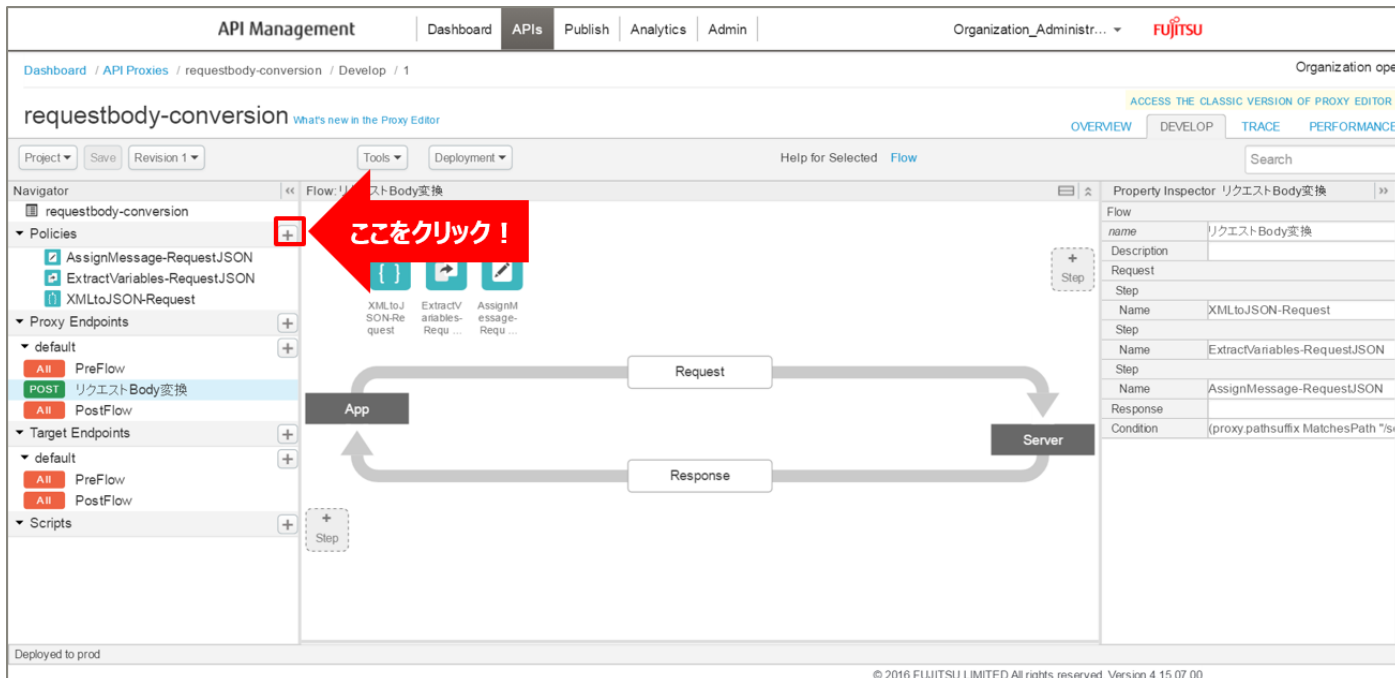
「3.2.2.1 XML → JSON 変換」で作成した API Proxy を利用します。

「APIs」メニューをクリックし、API Proxies の一覧から、編集する API Proxy を選択します。

# 1) Extract Variables ポリシーの作成（ユーザー情報の変数化）

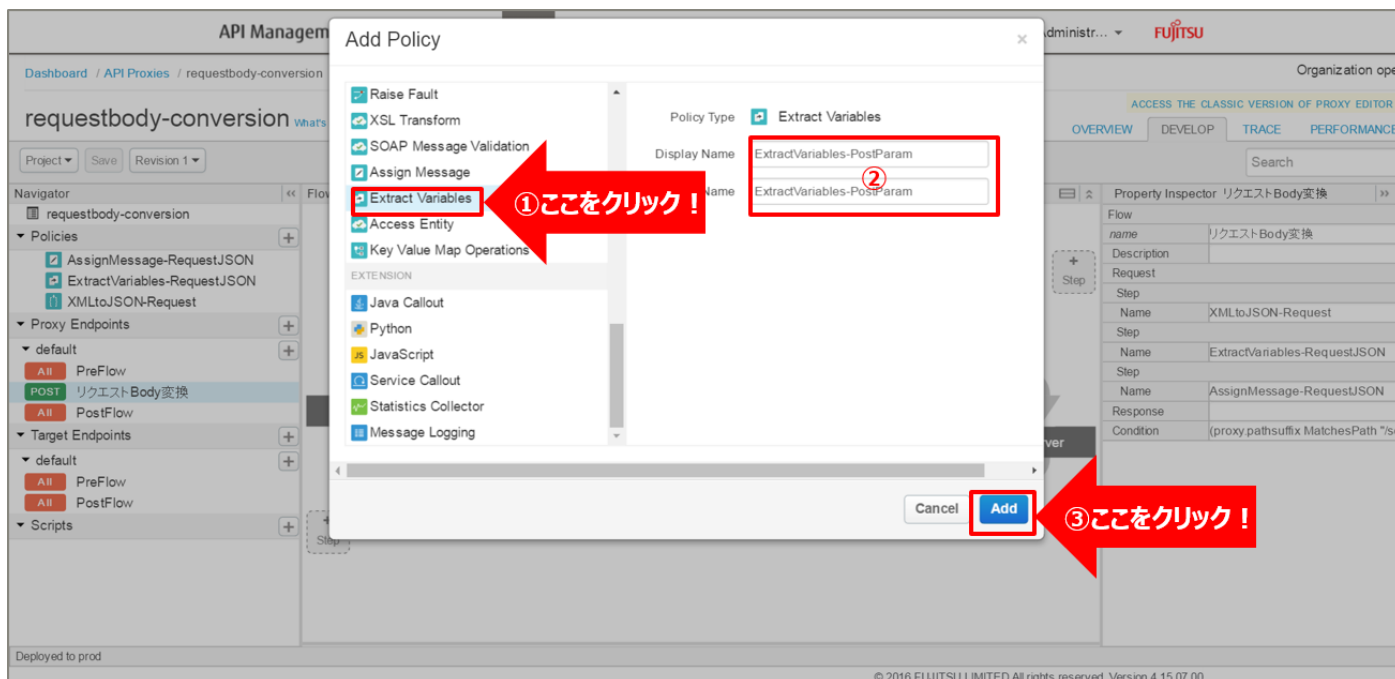
パラメーターキーを変更したリクエスト情報を変数として格納（変数化）する Extract Variables ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Extract Variables」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : ExtractVariables-PostParam（任意の名前）
- Name : ExtractVariables-PostParam（任意の名前）



追加した Extract Variables ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

**API Management**

Dashboard / API Proxies / requestbody-conversion / Develop

requestbody-conversion

Project Save Revision 1 Tools

Navigator

- requestbody-conversion
- Policy: ExtractVa
- Policies
  - AssignMessage-Req
  - ExtractVariables-PostParam**
  - ExtractVariables-Req
  - XMLToJson-Request
- Proxy Endpoints
  - default
- Target Endpoints
  - default
    - PreFlow
    - Post (リクエストBody変換)
    - PostFlow
- Scripts

Deployed to prod

**ExtractVariables-PostParam.xml**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true" name="ExtractVariables-PostParam">
  <DisplayName>ExtractVariables-PostParam</DisplayName>
  <Properties/>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <JSONPayload>
    <Variable name="site" type="string">
      <JSONPath>$.site</JSONPath>
    </Variable>
    <Variable name="password" type="string">
      <JSONPath>$.password</JSONPath>
    </Variable>
    <Variable name="permission" type="integer">
      <JSONPath>$.permission</JSONPath>
    </Variable>
  </JSONPayload>
</ExtractVariables>
```

JSONデータを変数に格納

ここをクリック！

以下省略、詳細はポリシーテンプレートを参照〜

Code ExtractVariables-PostParam

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ExtractVariables async="false" continueOnError="false" enabled="true" name="ExtractVariables-PostParam">
3   <DisplayName>ExtractVariables-PostParam</DisplayName>
4   <Properties/>
5   <URIPath name="name"/>
6   <QueryParam name="name"/>
7   <Header name="name"/>
8   <FormParam name="name"/>
9   <Variable name="name"/>
10  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
11  <JSONPayload>
12    <Variable name="name">
13      <JSONPath>{example}</JSONPath>
14    </Variable>
15  </JSONPayload>
16 </ExtractVariables>
```

Header	name
name	name
FormParam	name
name	name
Variable	name
name	name
IgnoreUnresolve...	true
JSONPayload	
Variable	name
name	name
JSONPath	{example}
Case	name

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.00

【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true"
name="ExtractVariables-PostParam">
  <DisplayName>ExtractVariables-PostParam</DisplayName>
  <Properties/>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <JSONPayload>
    <Variable name="site" type="string">
      <JSONPath>$.site</JSONPath>
    </Variable>
    <Variable name="password" type="string">
      <JSONPath>$.password</JSONPath>
    </Variable>
    <Variable name="permission" type="integer">
      <JSONPath>$.permission</JSONPath>
    </Variable>

    ~ 略 ~

    <Variable name="grade" type="integer">
      <JSONPath>$.grade</JSONPath>
    </Variable>
  </JSONPayload>
  <Source clearPayload="false">request</Source>
  <VariablePrefix>customrequest</VariablePrefix>
</ExtractVariables>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

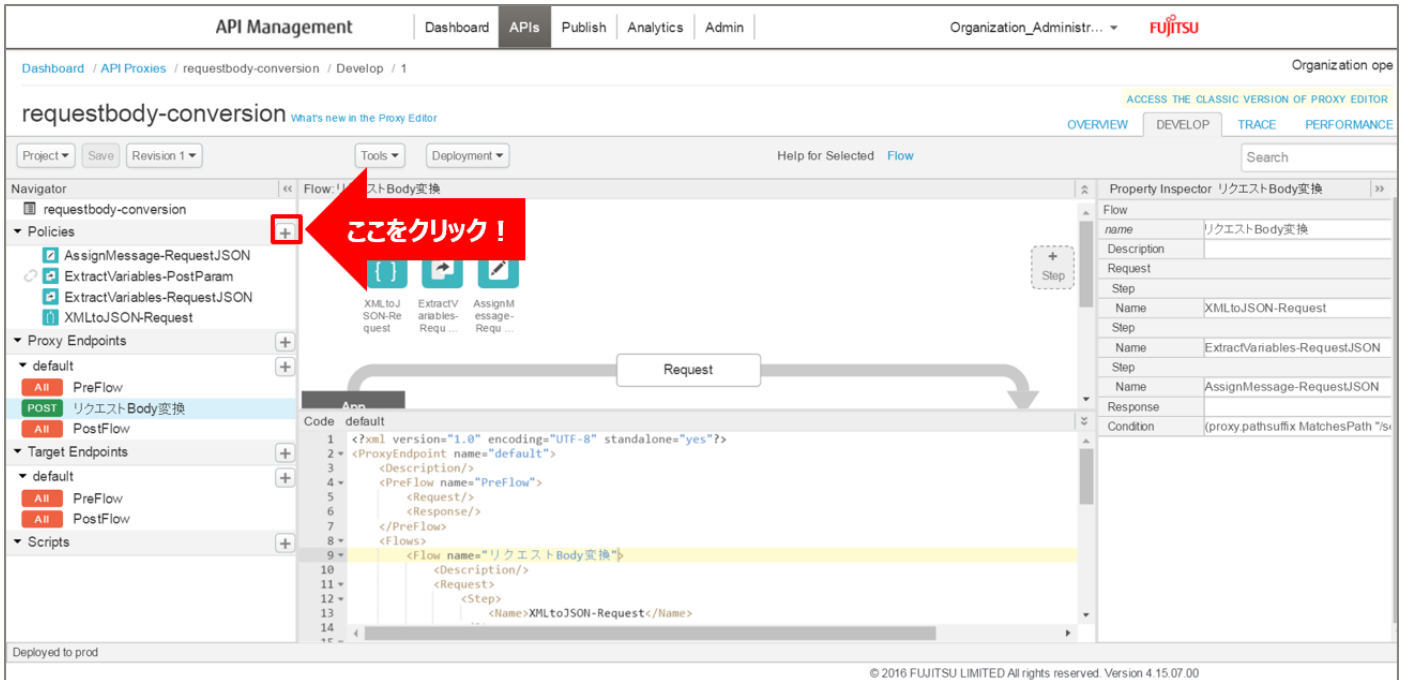
※定義内容の詳細は、「[A2.4. Extract Variables XML 仕様](#)」をご参照ください。



## 2) Assign Message ポリシーの作成 (HTTP リクエストの作成)

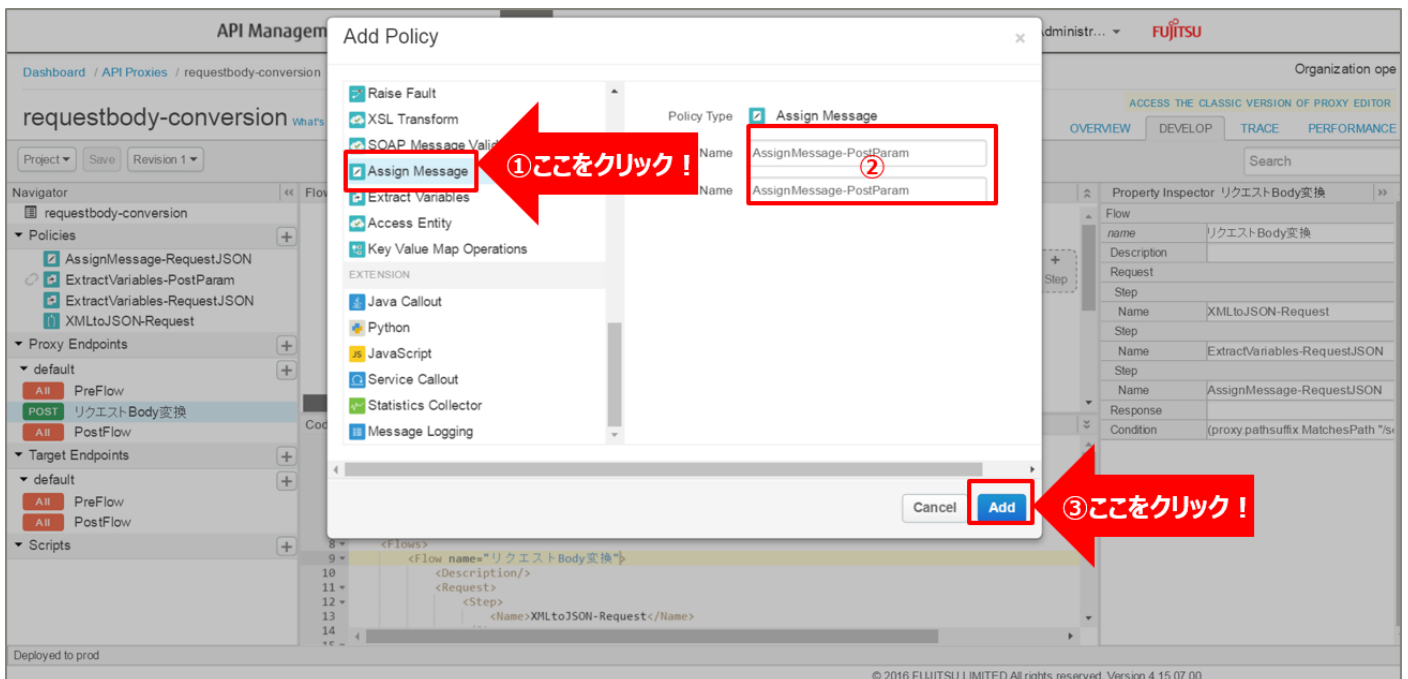
HTTP メッセージを作成する Assign Message ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Assign Message」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : AssignMessage-PostParam (任意の名前)
- Name : AssignMessage-PostParam (任意の名前)



追加した Assign Message ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

**API Management**

Dashboard / API Proxies / requestbody-conversion / Develop /

requestbody-conversion

Project Save Revision 1 Tools

requestbody-conversion Policy: AssignMe

AssignMessage-PostParam

AssignMessage-Req

ExtractVariables-PostPa

ExtractVariables-RequestJSON

XMLtoJSON-Request

Proxy Endpoints

default

All PreFlow

POST リクエスト Body変換

All PostFlow

Target Endpoints

default

All PreFlow

All PostFlow

Scripts

Attached t

Code AssignM

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>

2 <AssignMessage async="false" continueOnError="false" enabled="true" name="AssignMessage-PostParam">

3 <DisplayName>AssignMessage-PostParam</DisplayName>

4 <Set>

5 <Payload contentType="application/json" variablePrefix="@ " variableSuffix="#">

6 {

7 "cloudId": "@customrequest.site#",

8 "password": "@customrequest.password#",

9 "permission": "@customrequest.permission#",

10 "loginId": "@customrequest.loginId#",

11 "userName": "@customrequest.userName#",

12 "userDescription": "@customrequest.userDescription#",

13 "birthday": "@customrequest.birthday#",

14 "gender": "@customrequest.gender#",

15 "height": "@customrequest.height#",

16 "weight": "@customrequest.weight#",

17 "grade": "@customrequest.grade#"

18 }</Payload>

19 <Verb>POST</Verb>

20 </Set>

21 <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>

22 <AssignTo createNew="false" transport="http" type="request"/>

23 </AssignMessage>

24 <QueryParams/>

25 <FormParams/>

26 <Payload/>

27 <Verb/>

28 <StatusCode/>

29 <ReasonPhrase/>

30 <Path/>

31 </Copy>

Path

Remove

Headers

Header

name h1

QueryParams

QueryParam

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.00

【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true" name="AssignMessage-
PostParam">
  <DisplayName>AssignMessage-PostParam</DisplayName>
  <Set>
    <Payload contentType="application/json" variablePrefix="@ " variableSuffix="#">
      {
        "cloudId":"@customrequest.site#",
        "password":"@customrequest.password#",
        "permission":@customrequest.permission#,
        "loginId":"@customrequest.loginId#",
        "userName":"@customrequest.userName#",
        "userDescription":"@customrequest.userDescription#",
        "birthday":"@customrequest.birthday#",
        "gender":@customrequest.gender#,
        "height":@customrequest.height#,
        "weight":@customrequest.weight#,
        "grade":"@customrequest.grade#"
      }
    </Payload>
    <Verb>POST</Verb>
  </Set>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="request"/>
</AssignMessage>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.3. Assign Message XML 仕様](#)」をご参照ください。

### 3) ポリシーの解除

「3.2.2.1. 5) ポリシーの配置」で配置したポリシーを解除します。

配置済みのポリシーにカーソルを合わせると、「x」が表示されます。

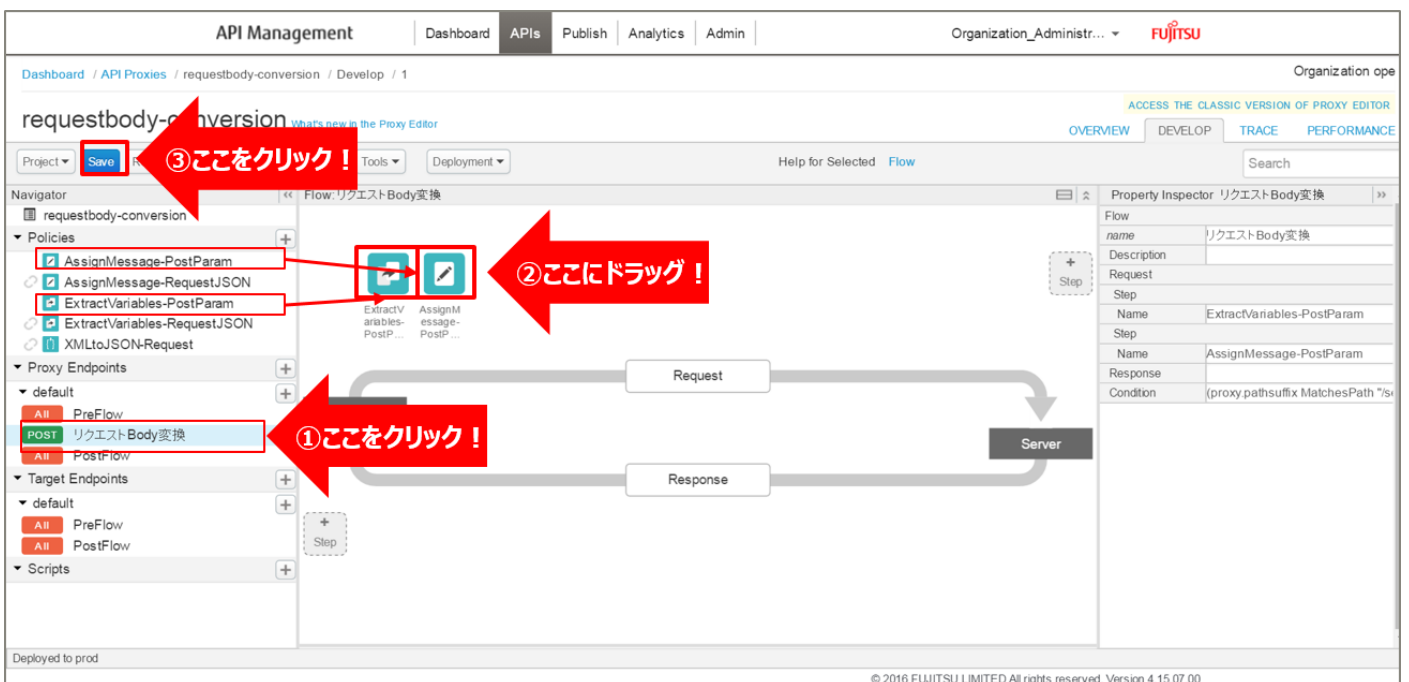
以下のポリシーにカーソルを合わせ、「x」をクリックしてポリシーを解除します。

- XMLtoJSON-Request
- ExtractVariables-RequestJSON
- AssignMessage-RequestJSON



### 4) ポリシーの配置

以下の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。



## 5) 動作確認

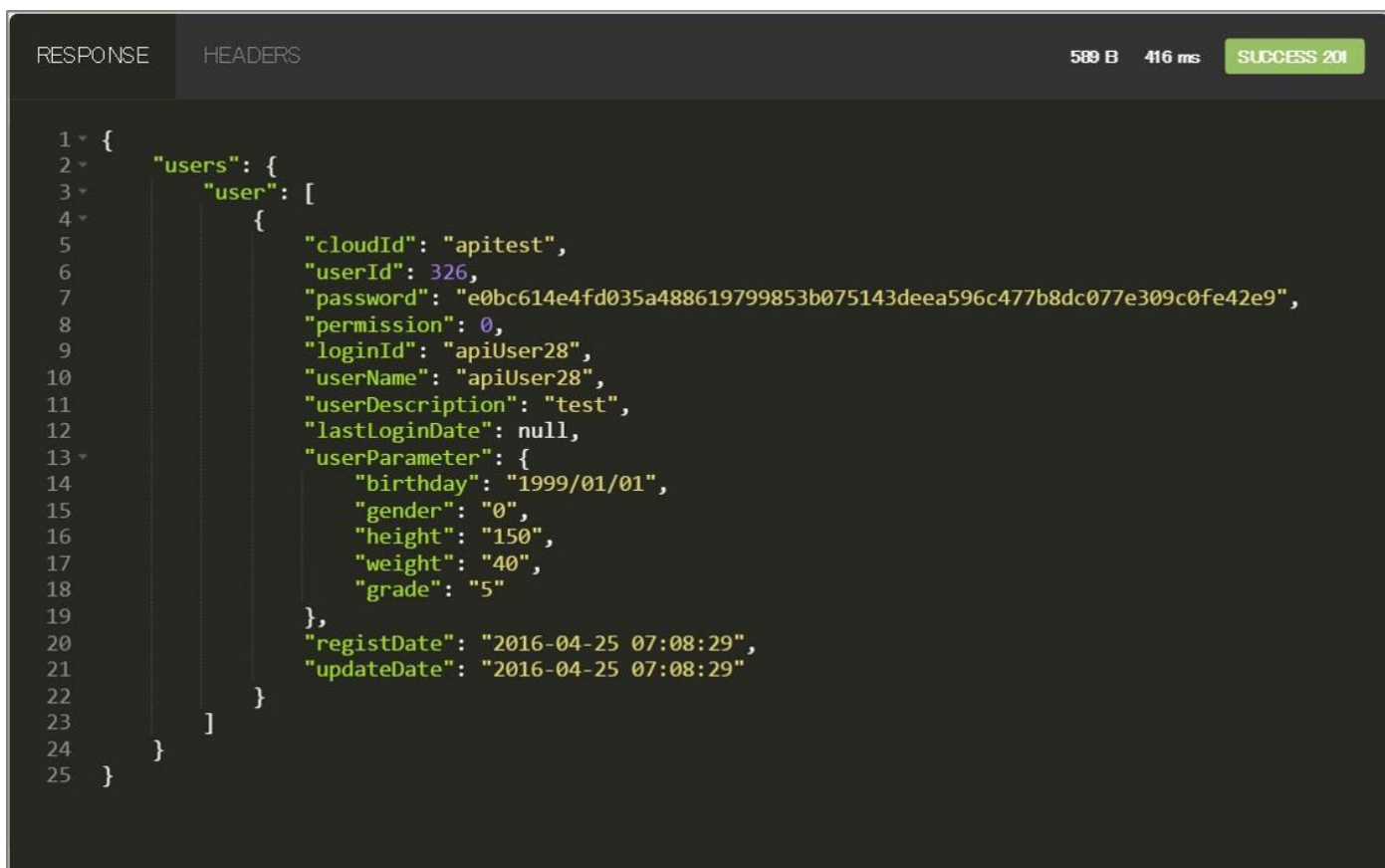
作成した API Proxy の動作確認を行います。

以下の例のように、URL へアクセスするとユーザー情報が登録され、登録したユーザー情報が JSON 形式で返却されます。

http://{FQDN}:10080/requestbody-conversion/sensing\_webapi/api/apitest/users

※http プロトコルでの動作確認例です（ポート番号は 10080 です）。

※{FQDN}: API Proxy の作成時に生成された URL のホスト名（FQDN）を指定します。



```
RESPONSE HEADERS 589 B 416 ms SUCCESS 201
1 {
2   "users": {
3     "user": [
4       {
5         "cloudId": "apitest",
6         "userId": 326,
7         "password": "e0bc614e4fd035a488619799853b075143deea596c477b8dc077e309c0fe42e9",
8         "permission": 0,
9         "loginId": "apiUser28",
10        "userName": "apiUser28",
11        "userDescription": "test",
12        "lastLoginDate": null,
13        "userParameter": {
14          "birthday": "1999/01/01",
15          "gender": "0",
16          "height": "150",
17          "weight": "40",
18          "grade": "5"
19        },
20        "registDate": "2016-04-25 07:08:29",
21        "updateDate": "2016-04-25 07:08:29"
22      }
23    ]
24  }
25 }
```

### 3.3. レスポンス Body 変換

#### 3.3.1. ユースケース概要

クライアントにレスポンスを返却する前に、レスポンスの形式変換を行う API Proxy を実装します。  
ここでは、2 つのユースケースを説明します。

##### 【ユースケース 1：JSON → XML 変換】

バックエンドサービスとして、[天気情報を取得する API](#) を使用します。

API にアクセスする際の URL に都市コードを指定すると、指定された都市の天気情報を取得します。

API から返却された JSON 形式のレスポンスデータ（天気情報）を、XML 形式に変換してクライアントに返却するようにします。

API Proxy の実装フローは以下の通りです。

- 1) API Proxy の作成
  - 1-1) API Proxy の作成
  - 1-2) Conditional Flow (GET) の作成
- 2) JSON to XML ポリシーの作成（レスポンス情報の変換）
- 3) ポリシーの配置
- 4) 動作確認

使用する Policy は以下の通りです。

- JSON to XML ポリシー

##### 【ユースケース 2：JSON → HTML 変換】

ユースケース 1 で作成した API Proxy を利用します。

API から返却された JSON 形式のレスポンスデータ（天気情報）を、HTML 形式に変換してクライアントに返却するようにします。

API Proxy の実装フローは以下の通りです。

- 1) Extract Variables ポリシーの作成（天気情報の変数化）
- 2) Assign Message ポリシーの作成（HTTP レスポンスの作成）
- 3) ポリシーの解除
- 4) ポリシーの配置
- 5) 動作確認

使用する Policy は以下の通りです。

- Assign Message ポリシー
- Extract Variables ポリシー

### 3.3.2. 手順

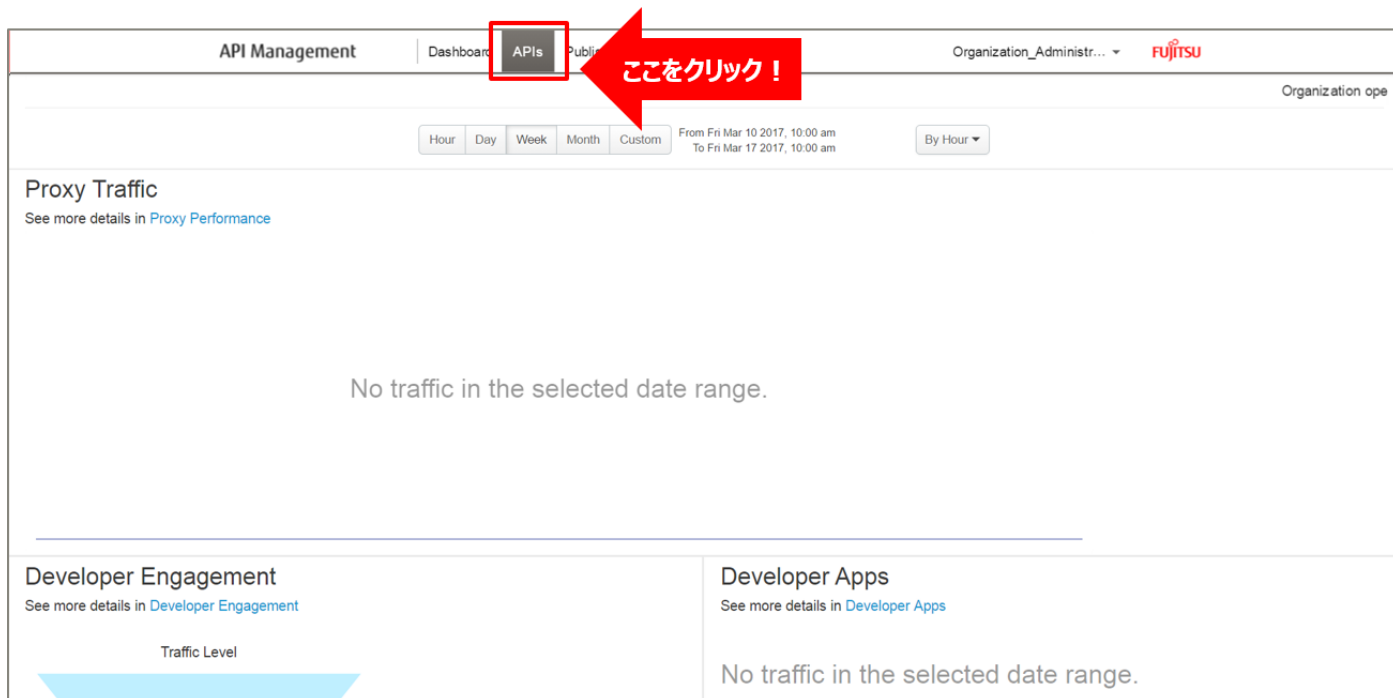
#### 3.3.2.1. JSON → XML 変換

##### 1) API Proxy の作成

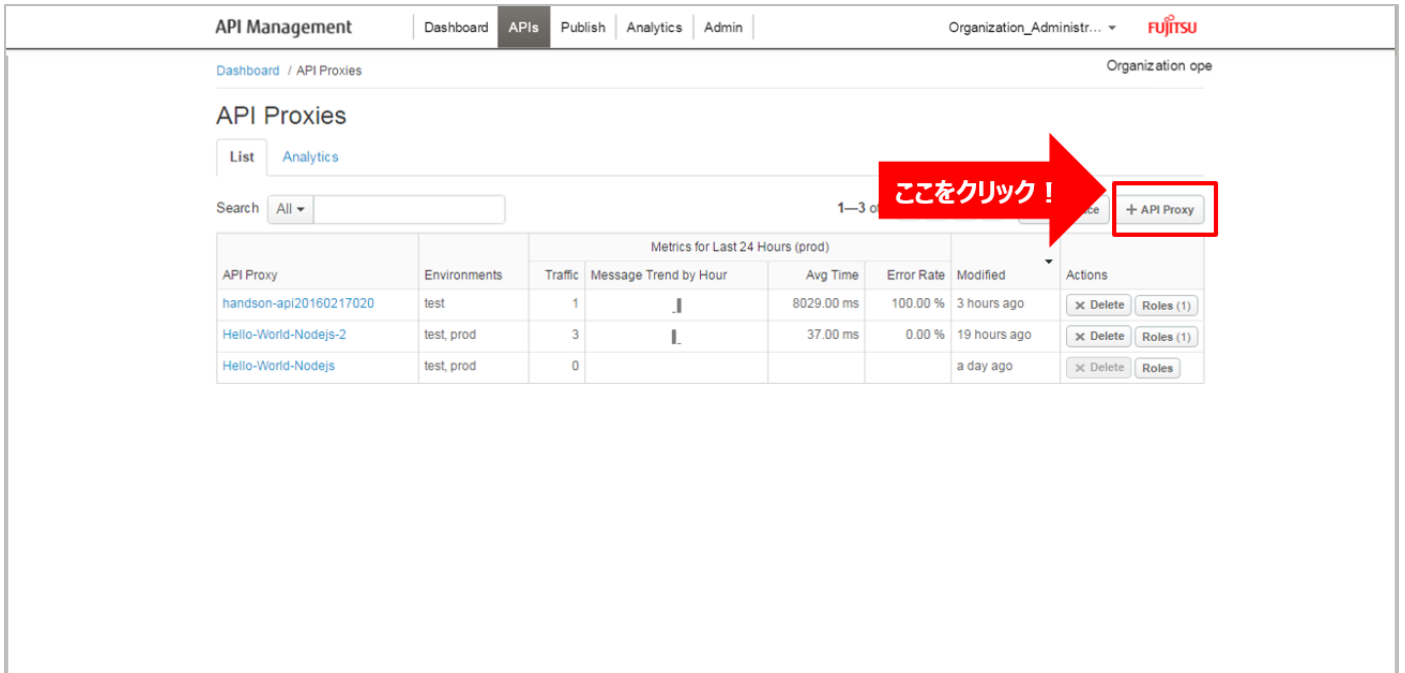
API Proxy を作成します。

##### 1-1) API Proxy の作成

画面上部の「APIs」メニューをクリックし、API Proxies 画面に遷移します。



API Proxies 画面で、「+ API Proxy」ボタンをクリックします。

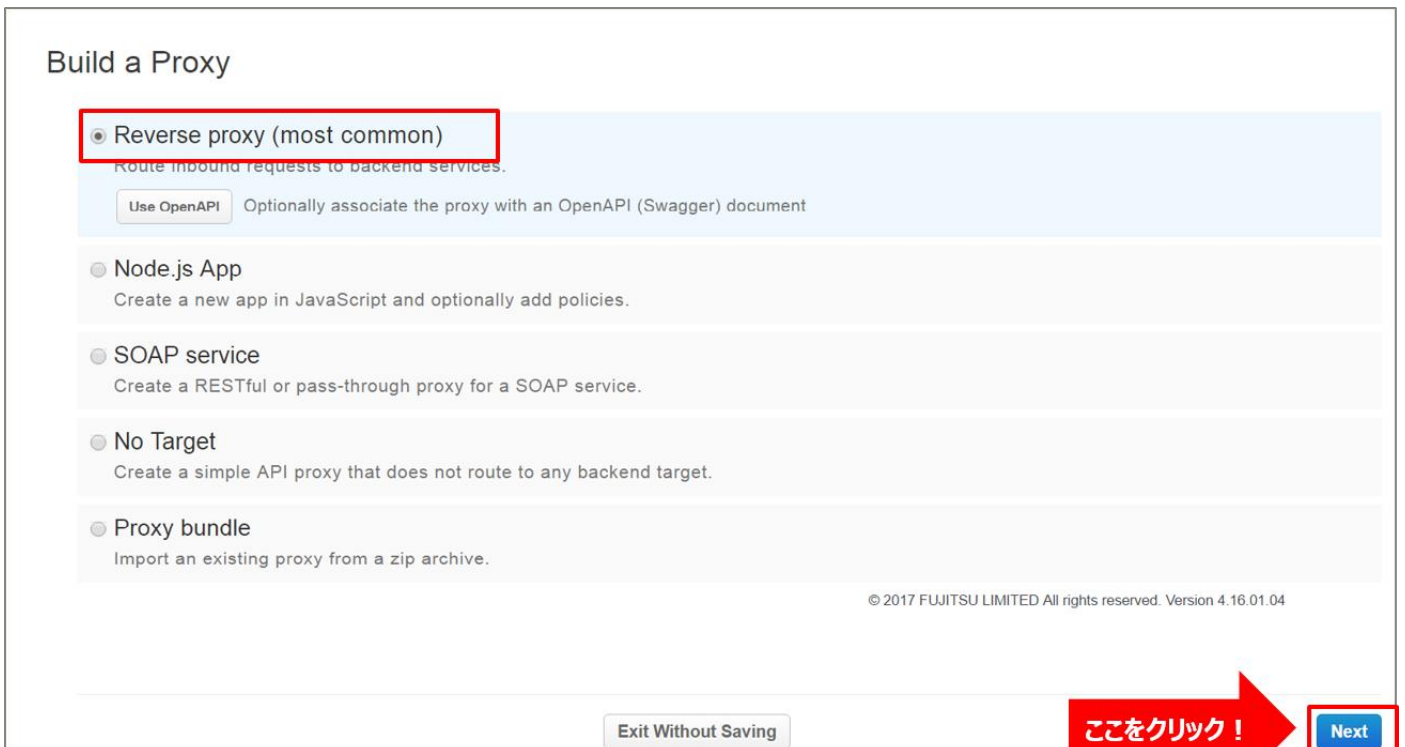


The screenshot shows the 'API Proxies' page in the API Management console. The page has a navigation bar with 'APIs' selected. Below the navigation bar, there are tabs for 'List' and 'Analytics'. A search bar is present with 'All' selected. A table displays the list of API proxies with columns for API Proxy, Environments, Metrics for Last 24 Hours (prod), and Actions. A red arrow points to the '+ API Proxy' button in the top right corner.

API Proxy	Environments	Metrics for Last 24 Hours (prod)				Modified	Actions
		Traffic	Message Trend by Hour	Avg Time	Error Rate		
handson-api20160217020	test	1	↓	8029.00 ms	100.00 %	3 hours ago	<a href="#">x Delete</a> <a href="#">Roles (1)</a>
Hello-World-Nodejs-2	test, prod	3	↓	37.00 ms	0.00 %	19 hours ago	<a href="#">x Delete</a> <a href="#">Roles (1)</a>
Hello-World-Nodejs	test, prod	0				a day ago	<a href="#">x Delete</a> <a href="#">Roles</a>

➤ Build a Proxy (TYPE)

「Reverse proxy (most common)」を選択し、「Next」ボタンをクリックします。



The screenshot shows the 'Build a Proxy' form. The 'Reverse proxy (most common)' option is selected and highlighted with a red box. Below this option, there is a checkbox for 'Use OpenAPI' and a description. Other options include 'Node.js App', 'SOAP service', 'No Target', and 'Proxy bundle'. At the bottom right, there is a 'Next' button highlighted with a red box and a red arrow pointing to it.

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04



➤ Build a Proxy (DETAILS)

API Proxy の情報を入力し、「Next」ボタンをクリックします。以下は入力例です。

- Proxy Name : responsebody-conversion (任意の名前)
- Proxy Base Path : /responsebody-conversion (任意のパス (自動入力))
- Existing API : http://weather.livedoor.com (任意の API)

### Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Specify the proxy details.

Proxy Name \*   
Valid characters are letters, numbers, dash (-), and underscore (\_).

Proxy Base Path \*   
A path component that uniquely identifies this API proxy. The public-facing URL of this API proxy is comprised of your organization name, an environment where this API proxy is deployed, and this Proxy Base Path. Example URL http://test-sandbox-test.apigee.net/responsebody-conversion

Existing API \*   
Defines the target URL invoked on behalf of this API proxy. Any URL that is accessible over the open Internet can be used. Example: https://api.usergrid.com

Description

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

**ここをクリック!**

➤ Build a Proxy (SECURITY)

Authentication : 「Pass through (none)」 を選択し、「Next」 ボタンをクリックします。

### Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

Secure access for users and clients.

Authorization  Pass through (none)  
 API Key  
 OAuth 2.0

Browser  Add CORS headers

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Next

➤ Build a Proxy (VIRTUAL HOSTS)

設定を変えずに「Next」ボタンをクリックします。

Build a Proxy

TYPE DETAILS SECURITY VIRTUAL HOSTS BUILD SUMMARY

Select the virtual hosts this proxy will bind to when it is deployed. You must select at least one virtual host. [Learn more...](#)

<input checked="" type="checkbox"/> Name	Environment	Host Aliases
<input checked="" type="checkbox"/> default	prod	http://:10080
	test	http://:10080
<input checked="" type="checkbox"/> secure	prod	https://:10443
	test	https://:10443

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Next

➤ Build a Proxy (BUILD)

設定を変えずに「Build and Deploy」ボタンをクリックします。

Build a Proxy

TYPE DETAILS SECURITY VIRTUAL HOSTS BUILD SUMMARY

You are ready to build and deploy your API proxy.

Deploy Environments  prod  test

Proxy Name responsebody-conversion

Proxy Type Reverse proxy

Virtual Hosts default, secure

Security None

Browser Do not allow direct requests from a browser via CORS

© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

Previous Exit Without Saving **ここをクリック!** Build and Deploy

➤ Build a Proxy (SUMMARY)

API Proxy の作成が完了したら、API Proxy のリンクをクリックします。

Build a Proxy

TYPE > DETAILS > SECURITY > VIRTUAL HOSTS > BUILD > SUMMARY

- ✓ Generated proxy
- ✓ Uploaded proxy
- ✓ Deployed to test

ここをクリック！ [View responsebody-conversion proxy in the editor .](#)

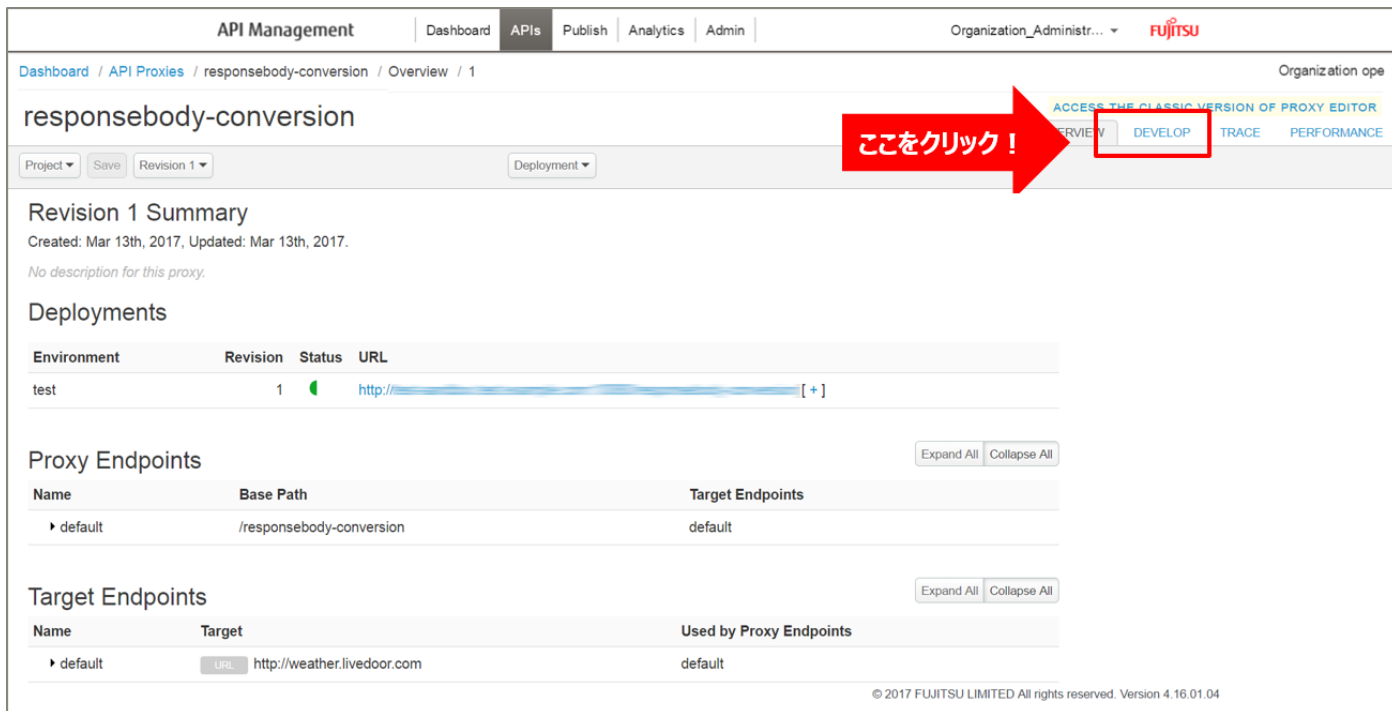
© 2017 FUJITSU LIMITED All rights reserved. Version 4.16.01.04

## 1-2) Conditional Flow (GET) の作成

バックエンドサービスに対するリソースパスと処理 (HTTP Method) の定義を行います。

クライアントからのリクエストがここで定義したパターンと一致する場合に、以降の手順で設定する処理を実行します。

「Develop」タブをクリックし、API proxy editor を開きます。



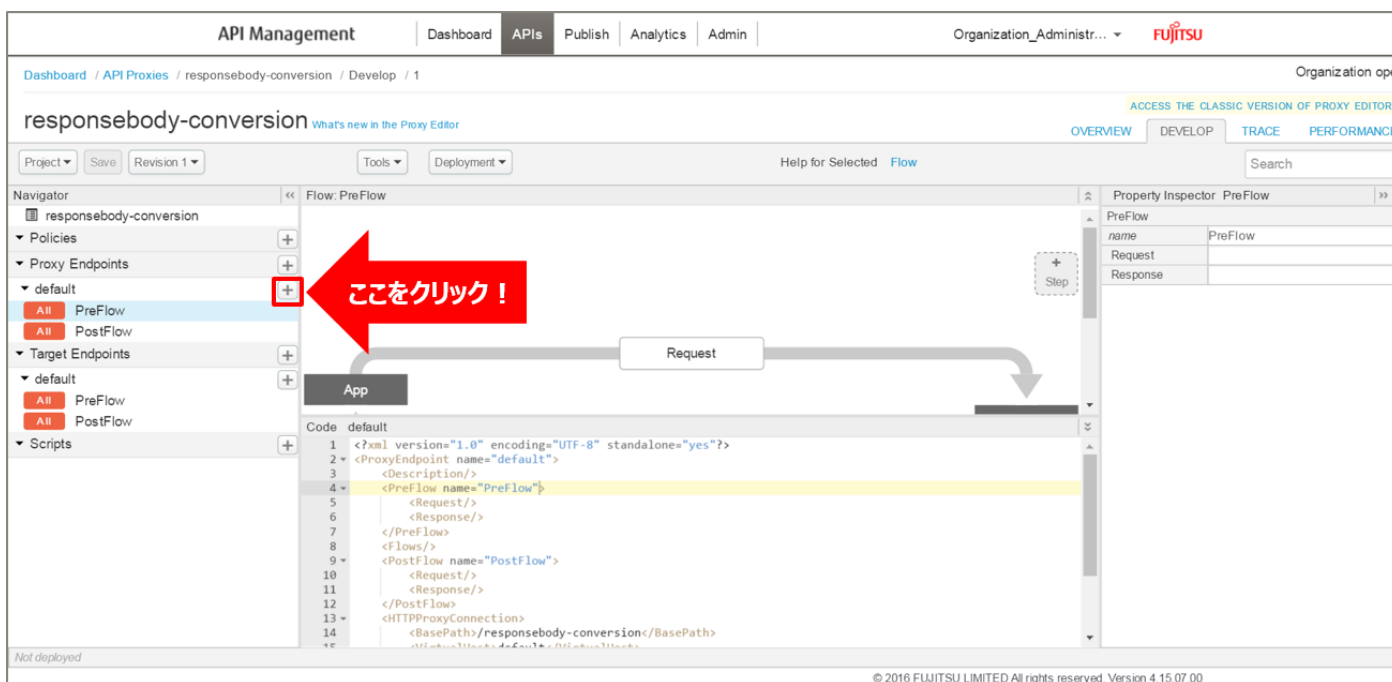
The screenshot shows the API Management console for the 'responsebody-conversion' proxy. The 'Develop' tab is selected, and a red arrow points to the 'DEVELOP' button in the top navigation bar. The page displays the 'Revision 1 Summary', 'Deployments' table, 'Proxy Endpoints' table, and 'Target Endpoints' table.

Environment	Revision	Status	URL
test	1	<span style="color: green;">●</span>	http://... [+]

Name	Base Path	Target Endpoints
default	/responsebody-conversion	default

Name	Target	Used by Proxy Endpoints
default	URL http://weather.livedoor.com	default

default の「+」ボタンをクリックし、New Conditional Flow を開きます。

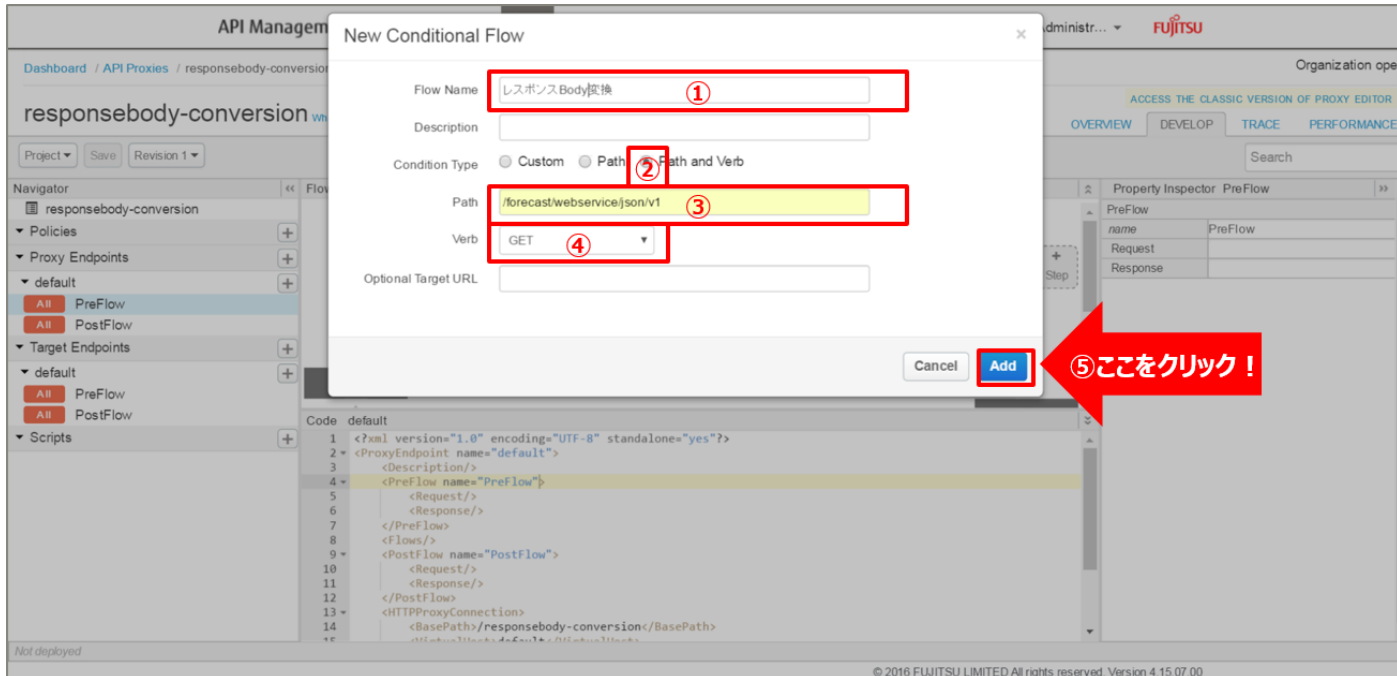


The screenshot shows the API Management console for the 'responsebody-conversion' proxy in the 'Develop' tab. The 'PreFlow' section is expanded, and a red arrow points to the '+' button next to it. The 'Code' section shows the XML configuration for the PreFlow.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ProxyEndpoint name="default">
3   <Description/>
4   <PreFlow name="PreFlow">
5     <Request/>
6   </PreFlow/>
7 </ProxyEndpoint/>
8 <Flows/>
9 <PostFlow name="PostFlow">
10  <Request/>
11  <Response/>
12 </PostFlow/>
13 <HTTPProxyConnection>
14   <BasePath>/responsebody-conversion</BasePath>
15 </HTTPProxyConnection/>
16 </ProxyEndpoint/>
```

Flow の情報を入力し、「Add」 ボタンをクリックします。以下は入力例です。

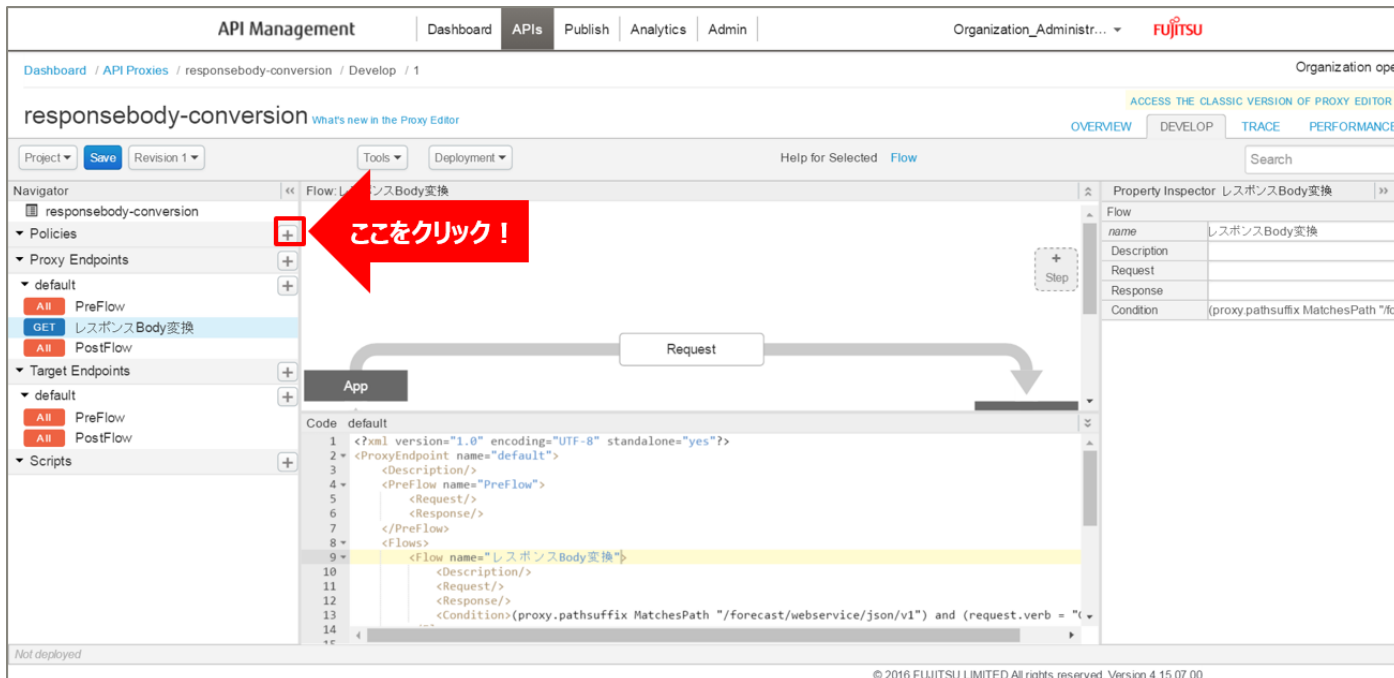
- Flow Name : レスポンス Body 変換 (任意の名前)
- Condition Type : 「Path and Verb」 を選択
- Path : /forecast/websevice/json/v1 (任意のパス)
- Verb : 「GET」 を選択



## 2) JSON to XML ポリシーの作成（レスポンス情報の変換）

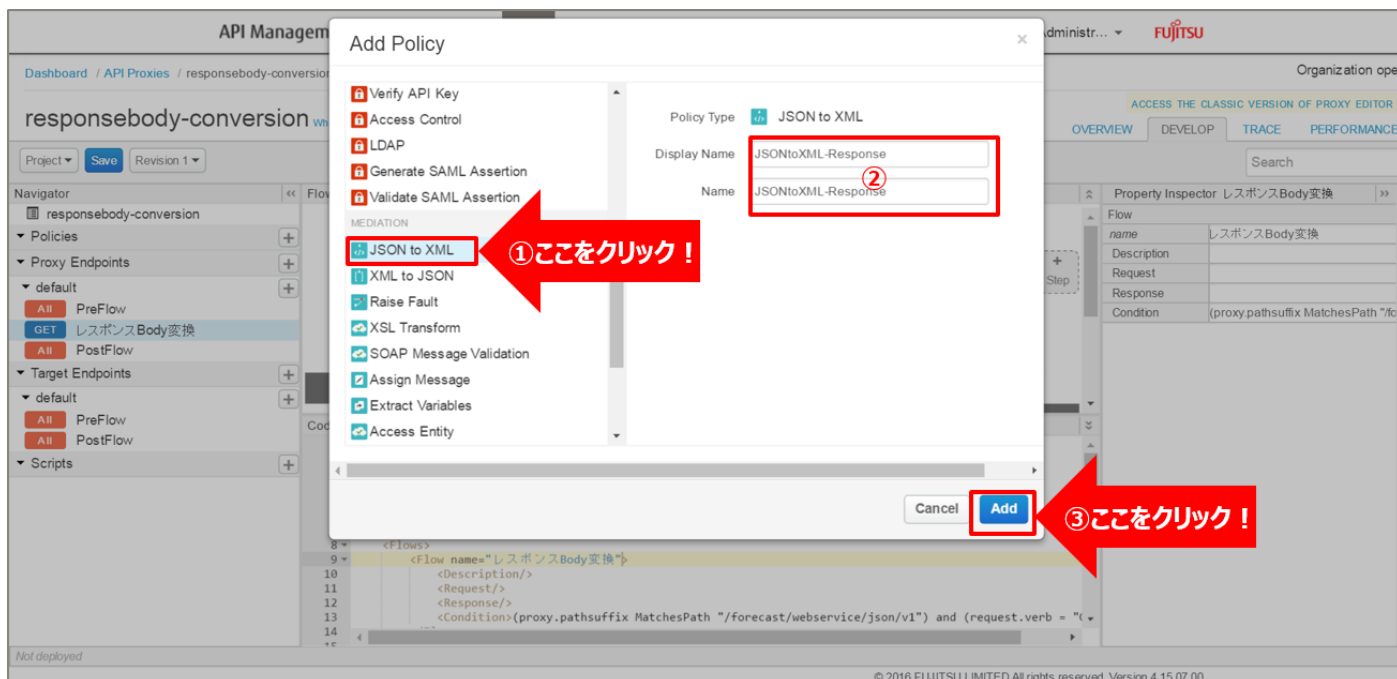
リクエスト情報やレスポンス情報を、JSON 形式から XML 形式に変換する JSON to XML ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「JSON to XML」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : JSONtoXML-Response (任意の名前)
- Name : JSONtoXML-Response (任意の名前)



追加した XML to JSON ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

The screenshot shows the API Management interface for editing a policy named 'JSONtoXML-Response.xml'. The left sidebar shows a tree view with 'Policies' expanded, and 'JSONtoXML-Response' selected. A red arrow points to this policy with the text 'ここをクリック!' (Click here!). The main editor shows the XML code for the policy, with a red box highlighting the definition. A black box above the code shows the simplified XML definition:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<JSONToXML async="false" continueOnError="false" enabled="true" name="JSONtoXML-Response">
  <DisplayName>JSONtoXML-Response</DisplayName>
  <Properties/>
  <OutputVariable>response</OutputVariable>
  <Source>response</Source>
</JSONToXML>
```

#### 【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<JSONToXML async="false" continueOnError="false" enabled="true" name="JSONtoXML-Response">
  <DisplayName>JSONtoXML-Response</DisplayName>
  <Properties/>
  <OutputVariable>response</OutputVariable>
  <Source>response</Source>
</JSONToXML>
```

※定義内容の詳細は、「[A2.1. JSON to XML XML 仕様](#)」をご参照ください。



### 3) ポリシーの配置

以下の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。

The screenshot displays the API Management interface for configuring a policy. The main area shows a flow diagram with 'Request' and 'Response' steps. A 'Server' icon is connected to the 'Response' step. The 'JSONtoXML-Response' policy is being dragged from the 'Policies' list on the left to the 'Response' step. The 'Save' button is highlighted in the top left corner.

③ここをクリック！

①ここをクリック！

②ここにドラッグ！

Property Inspector レスポンスBody変換	
Flow	
name	レスポンスBody変換
Description	
Request	
Response	
Step	
Name	JSONtoXML-Response
Condition	(proxy pathsuffix MatchesPath "fc

Not deployed

© 2016 FUJITSU LIMITED All rights reserved. Version 4.15.07.00

#### 4) 動作確認

作成した API Proxy の動作確認を行います。

以下の例のように、URL へアクセスすると JSON 形式のデータが返却されます。

`http://{FQDN}:10080/responsebody-conversion/forecast/webservice/json/v1?city={CityCode}`

※http プロトコルでの動作確認例です（ポート番号は 10080 です）。

※{FQDN}: API Proxy の作成時に生成された URL のホスト名（FQDN）を指定します。

※{CityCode} は、任意の都市コードに置き換えてください。（例：400040 など）

```
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<Root>
  <pinpointLocations>
    <link>http://weather.livedoor.com/area/forecast/4020200</link>
    <name>大牟田市</name>
  </pinpointLocations>
  <pinpointLocations>
    <link>http://weather.livedoor.com/area/forecast/4020300</link>
    <name>久留米市</name>
  </pinpointLocations>
  <pinpointLocations>
    <link>http://weather.livedoor.com/area/forecast/4020700</link>
    <name>柳川市</name>
  </pinpointLocations>
  <pinpointLocations>
    <link>http://weather.livedoor.com/area/forecast/4021000</link>
    <name>八女市</name>
  </pinpointLocations>
  <pinpointLocations>
    <link>http://weather.livedoor.com/area/forecast/4021100</link>
    <name>筑後市</name>
  </pinpointLocations>
  <pinpointLocations>
    <link>http://weather.livedoor.com/area/forecast/4021200</link>
    <name>大川市</name>
  </pinpointLocations>
  <pinpointLocations>
    <link>http://weather.livedoor.com/area/forecast/4021600</link>
    <name>小都市</name>
  </pinpointLocations>
  <pinpointLocations>
    <link>http://weather.livedoor.com/area/forecast/4022500</link>
    <name>うきは市</name>
  </pinpointLocations>
  <pinpointLocations>
    <link>http://weather.livedoor.com/area/forecast/4022800</link>
    <name>朝倉市</name>
  </pinpointLocations>
  <pinpointLocations>
    <link>http://weather.livedoor.com/area/forecast/4022900</link>
    <name>みやま市</name>
  </pinpointLocations>
  <pinpointLocations>
    <link>http://weather.livedoor.com/area/forecast/4044700</link>
    <name>筑前町</name>
  </pinpointLocations>
  <pinpointLocations>
    <link>http://weather.livedoor.com/area/forecast/4044800</link>
    <name>筑前町</name>
  </pinpointLocations>
```

#### 3.3.2.2. JSON → HTML 変換

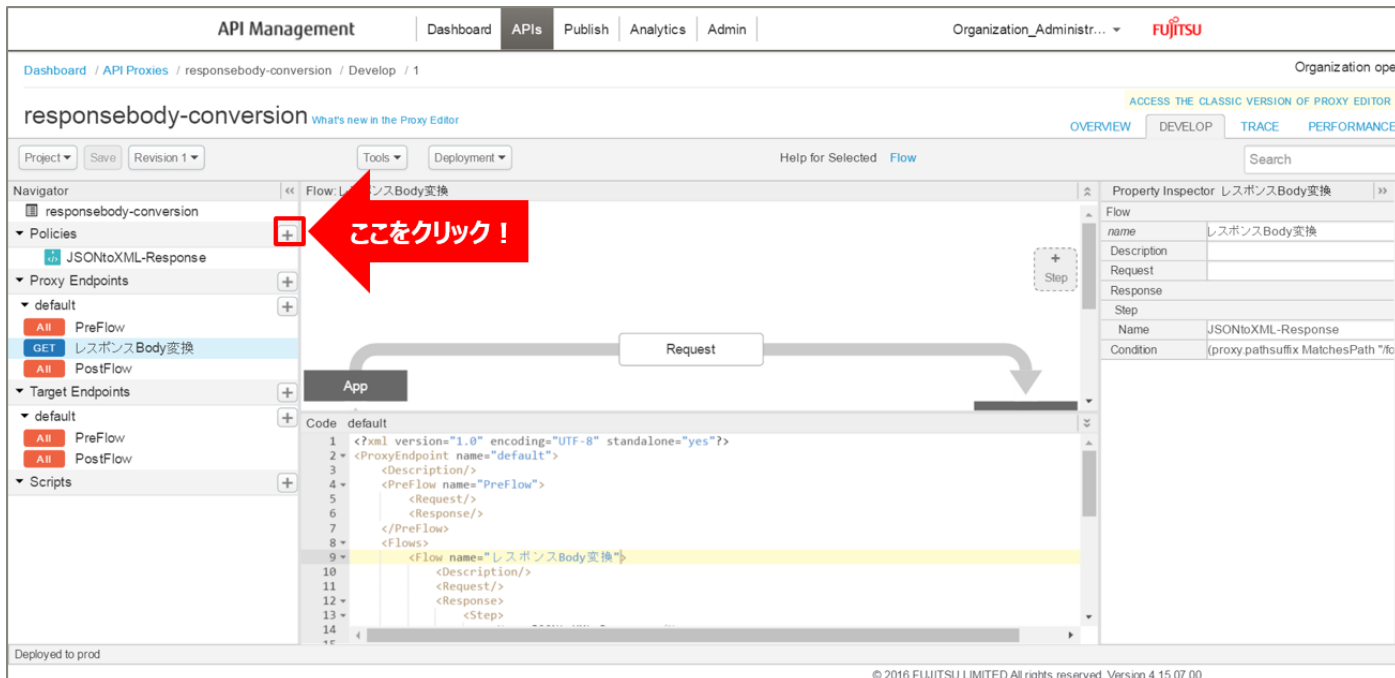
「3.3.2.1 JSON → XML 変換」で作成した API Proxy を利用します。

「APIs」メニューをクリックし、API Proxies の一覧から、編集する API Proxy を選択します。

# 1) Extract Variables ポリシーの作成（天気情報の変数化）

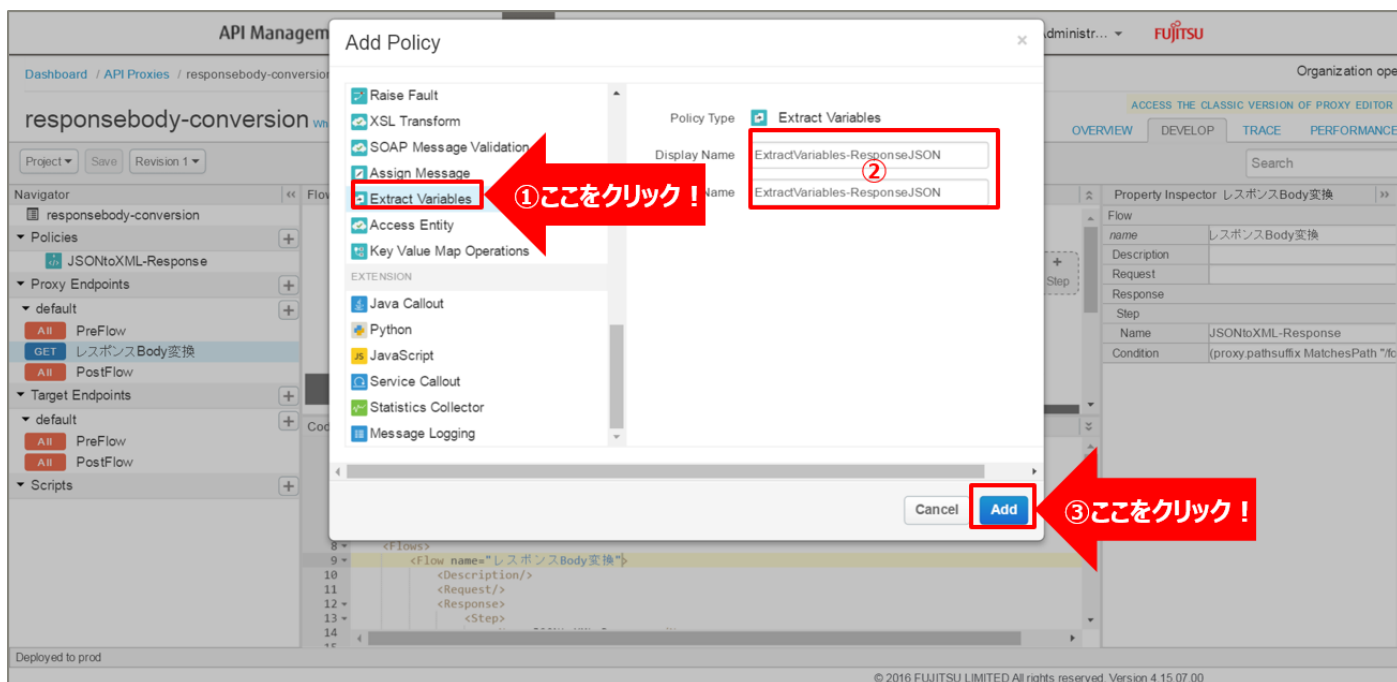
パラメーターキーを変更したリクエスト情報を変数として格納（変数化）する Extract Variables ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。



「Extract Variables」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : ExtractVariables-ResponseJSON（任意の名前）
- Name : ExtractVariables-ResponseJSON（任意の名前）



追加した Extract Variables ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。

【定義例】

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false" enabled="true"
name="ExtractVariables-ResponseJSON">
  <DisplayName>ExtractVariables-ResponseJSON</DisplayName>
  <JSONPayload>
    <Variable name="todayDateLabel">
      <JSONPath>$.forecasts[0].dateLabel</JSONPath>
    </Variable>
    <Variable name="todayTelop">
      <JSONPath>$.forecasts[0].telop</JSONPath>
    </Variable>
    <Variable name="todayDate">
      <JSONPath>$.forecasts[0].date</JSONPath>
    </Variable>
  </JSONPayload>
  <Source>response</Source>
  <VariablePrefix>responsejson</VariablePrefix>
</ExtractVariables>

```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.4. Extract Variables XML 仕様](#)」をご参照ください。

## 2) Assign Message ポリシーの作成 (HTTP レスポンスの作成)

HTTP メッセージを作成する Assign Message ポリシーを追加します。

Policies の「+」ボタンをクリックし、Add Policy を開きます。

The screenshot shows the API Management console interface. The 'Policies' section is expanded, and a red arrow points to a '+' button next to the 'Policies' header. A red callout box with the text 'ここをクリック!' (Click here!) is positioned over the '+' button. The main area displays a flow diagram with a 'Request' step and an 'App' step. The 'Code' section shows XML configuration for the flow, with the following content:

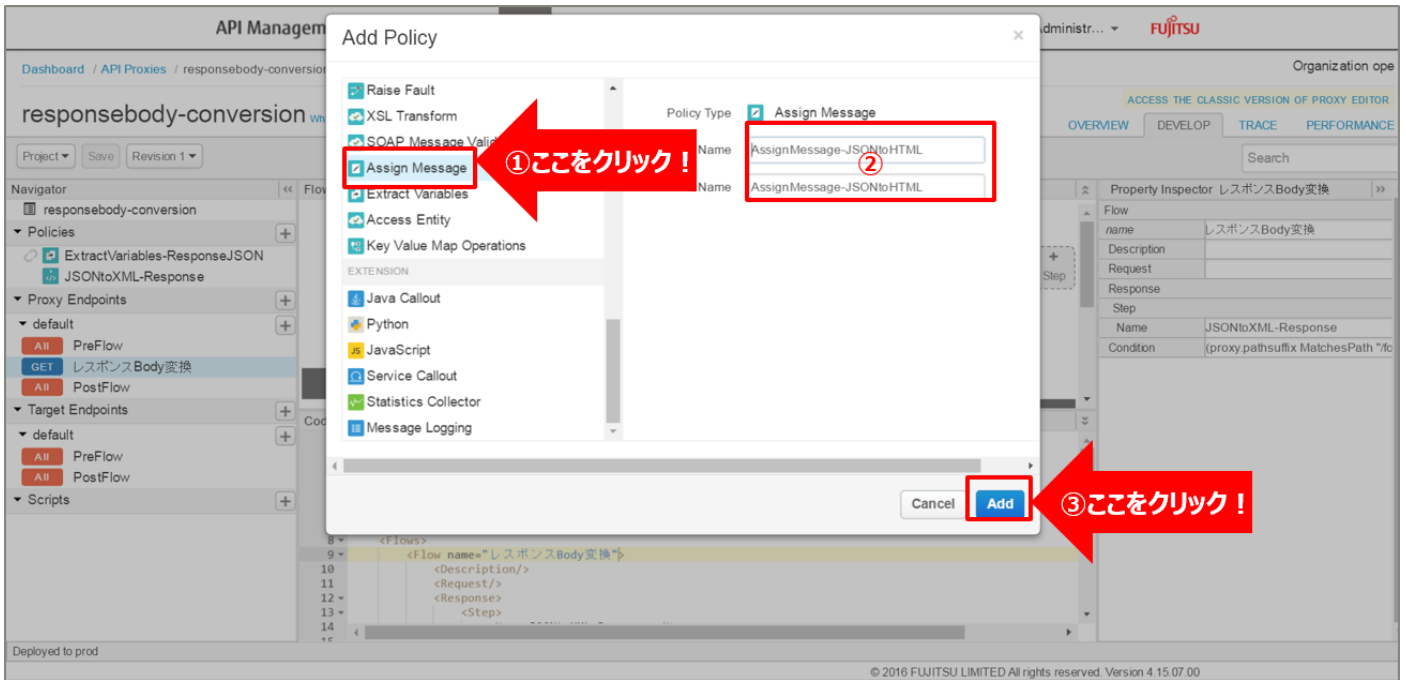
```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ProxyEndpoint name="default">
3   <Description/>
4   <Preflow name="Preflow">
5     <Request/>
6     <Response/>
7   </Preflow/>
8   <Flows>
9     <Flow name="レスポンスBody変換">
10      <Description/>
11      <Request/>
12      <Response/>
13      <Step/>
14    </Flow/>
15  </Flows/>
16 </ProxyEndpoint/>
```

The 'Property Inspector' on the right shows the details for the selected flow:

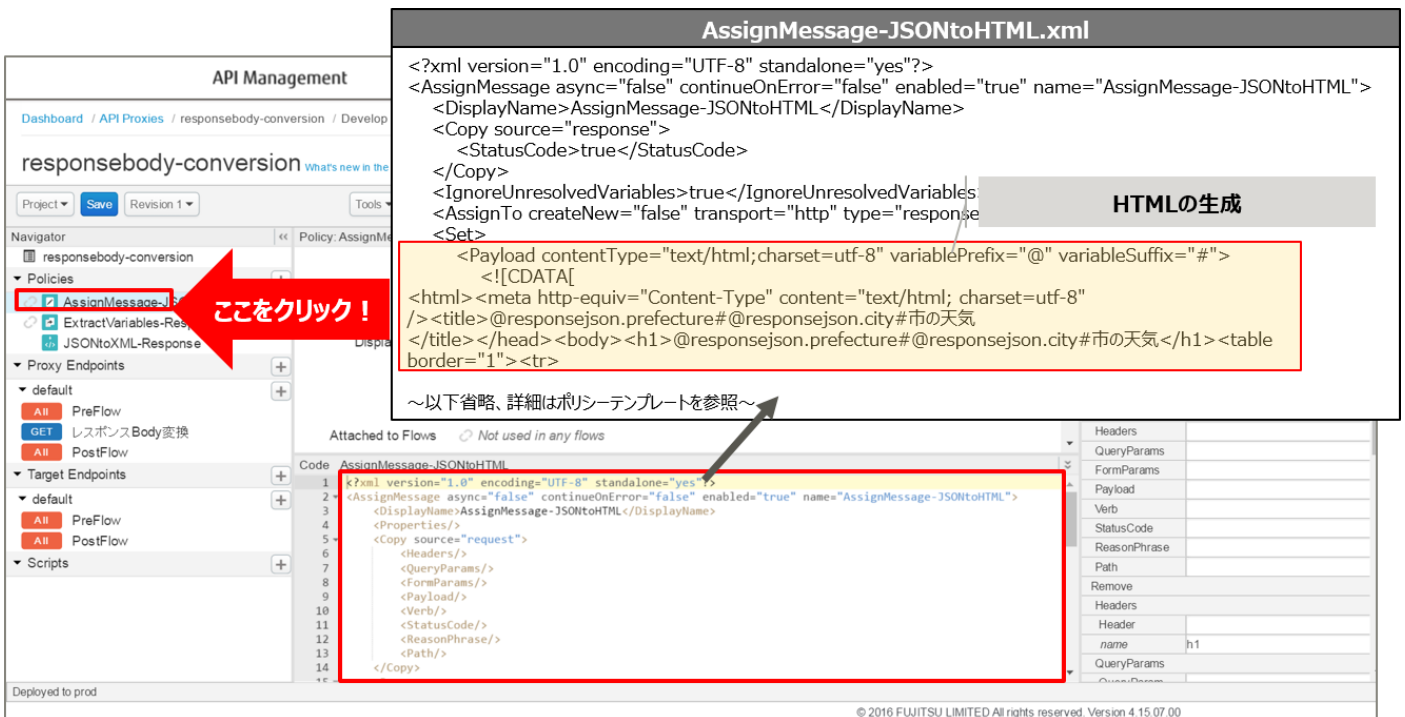
Property Inspector レスポンスBody変換	
Flow	
name	レスポンスBody変換
Description	
Request	
Response	
Step	
Name	JSONtoXML-Response
Condition	(proxy pathsuffix MatchesPath "fo

「Assign Message」をクリック後、ポリシーの情報を入力し、「Add」ボタンをクリックします。以下は入力例です。

- Display Name : AssignMessage-JSONtoHTML (任意の名前)
- Name : AssignMessage-JSONtoHTML (任意の名前)



追加した Assign Message ポリシーを選択し、ポリシー編集画面を表示します。  
必要に応じてポリシーの定義を編集します。



【定義例】

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true" name="AssignMessage-
JSONtoHTML">
  <DisplayName>AssignMessage-JSONtoHTML</DisplayName>
  <Copy source="response">
    <StatusCode>true</StatusCode>
  </Copy>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="response"/>
  <Set>
    <Payload contentType="text/html;charset=utf-8" variablePrefix="@@"
variableSuffix="#">
      <![CDATA[
<html><meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/><title>@responsejson.prefecture#@responsejson.city#市の天気
</title></head><body><h1>@responsejson.prefecture#@responsejson.city#市の天気
</h1><table border="1"><tr>
~ 略 ~
</tr></table><br /><p>@responsejson.description#</p></body></html>
]]>
    </Payload>
  </Set>
</AssignMessage>
```

※変更箇所や定義のポイントとなる箇所を赤字で示しています。

※定義内容の詳細は、「[A2.3. Assign Message XML 仕様](#)」をご参照ください。

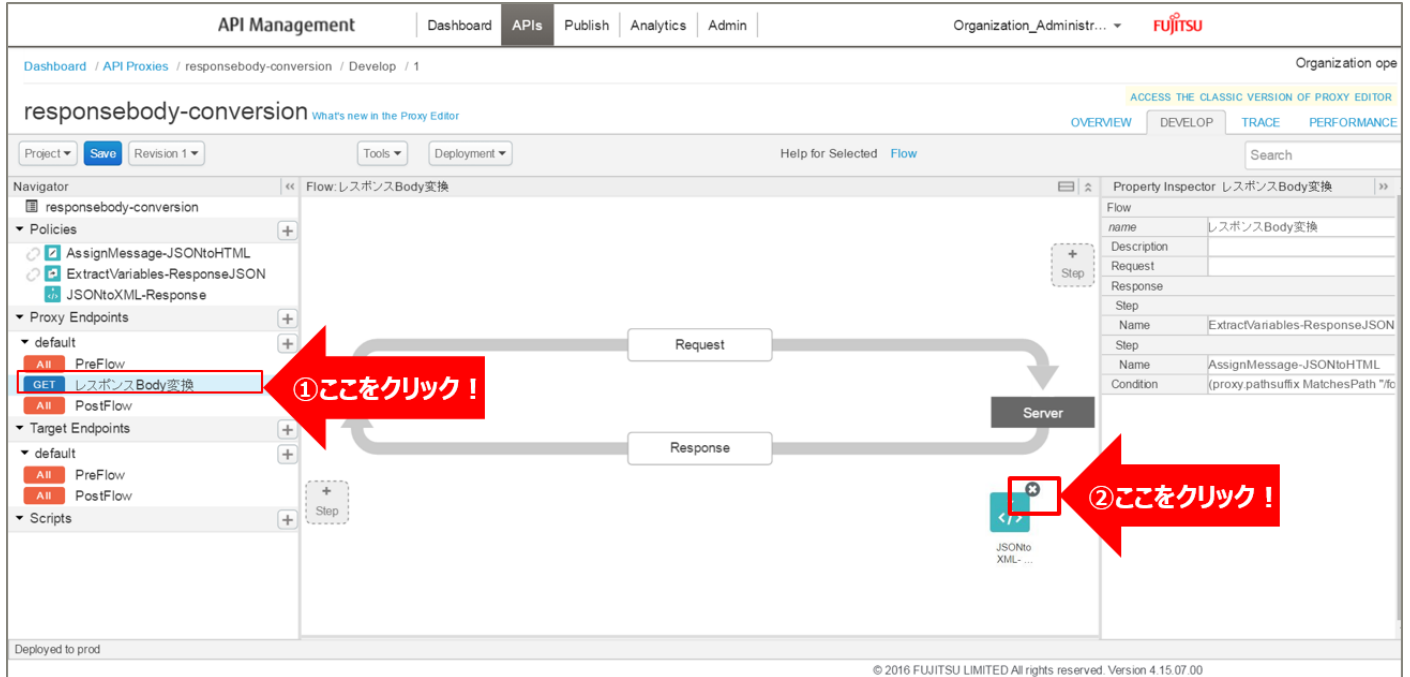
### 3) ポリシーの解除

「3.3.2.1. 3) ポリシーの配置」で配置したポリシーを解除します。

配置済みのポリシーにカーソルを合わせると、「x」が表示されます。

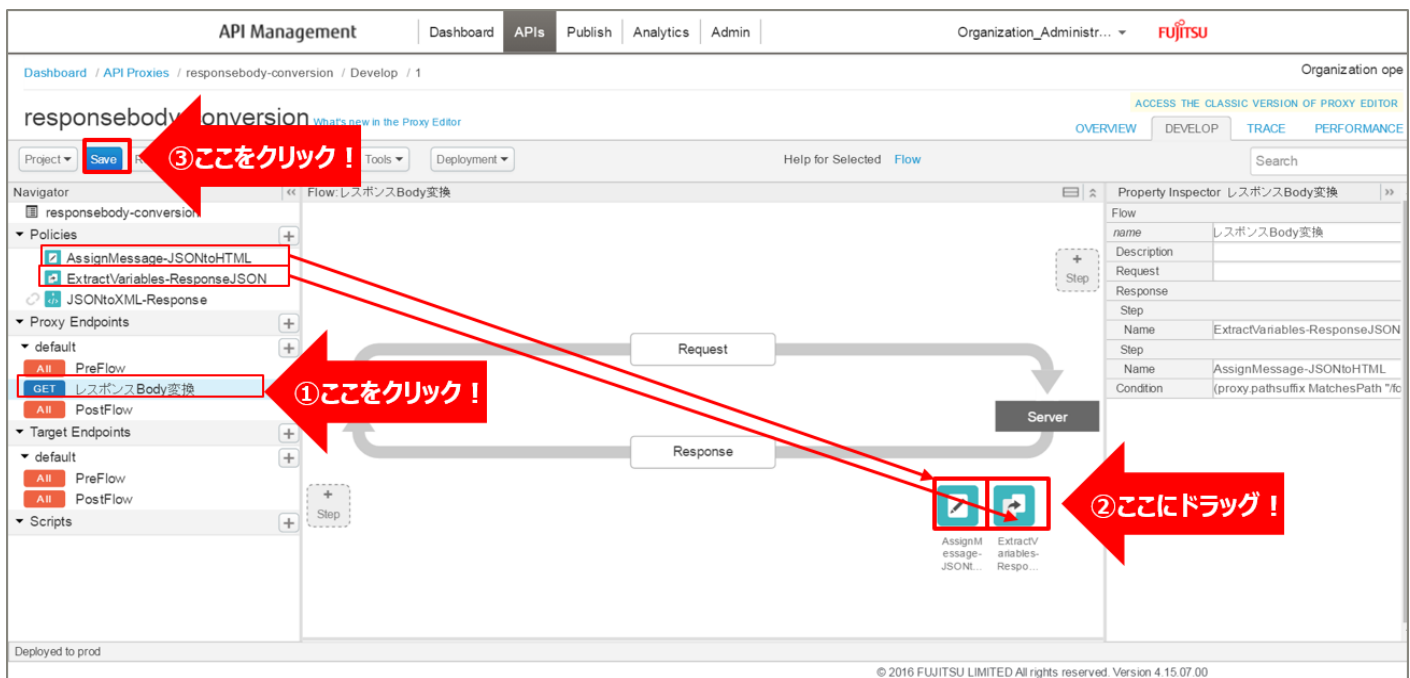
以下のポリシーにカーソルを合わせ、「x」をクリックしてポリシーを解除します。

- JSONtoXML-Response



### 4) ポリシーの配置

以下の通り、作成したポリシーを配置し、「Save」ボタンをクリックします。





## 5) 動作確認

作成した API Proxy の動作確認を行います。

以下の例のように、URL へアクセスすると JSON 形式のデータが返却されます。

`http://{FQDN}:10080/responsebody-conversion/forecast/websevice/json/v1?city={CityCode}`

※http プロトコルでの動作確認例です（ポート番号は 10080 です）。

※{FQDN}: API Proxy の作成時に生成された URL のホスト名 (FQDN) を指定します。

※{CityCode} は、任意の都市コードに置き換えてください。(例: 400040 など)

### 福岡県久留米市の天気

今日 (2016-04-21)	明日 (2016-04-22)
	



九州北部地方は、低気圧や前線の影響により雨となり、非常に激しく降っている所があります。21日の九州北部地方は、低気圧や前線の影響により、雨で雷を伴い非常に激しく降る所があるでしょう。22日の九州北部地方は、気圧の谷の影響により曇りですが、次第に高気圧に覆われて晴れとなるでしょう。波の高さは、九州北部地方の沿岸の海域では、21日は4メートル、22日は2メートルでしょう。豊後水道では、21日は3メートル、22日は1.5メートルでしょう。福岡県の内海では、21日は瀬戸内側では1.5メートル、有明海では1メートル、22日は0.5メートルでしょう。〈天気変化等の留意点〉筑後地方では、21日は、低気圧や前線の影響により雨となり、屋前から屋過ぎには雷を伴い1時間50ミリの非常に激しい雨の降る所があるでしょう。土砂災害に警戒してください。また、陸上でも強風が予想されるため飛来物などに留意してください。

## A. 付録

本書で使用している Policy の XML 仕様について説明します。

### A1. SECURITY

以下の Policy の XML 仕様について説明します。

 Verify API Key	アクセスを許可する API Key を指定することができます。
 OAuth v2.0	OAuth2.0 のエンドポイントに対する設定（アクセストークンの生成やチェック等）ができます。

#### A1.1. Verify API Key XML 仕様

<VerifyAPIKey> attributes

- 1) name 属性                      ポリシー名。<DisplayName> element と同じにしておきます。
- 2) continueOnError 属性        ポリシー失敗時にエラーを返すよう false を設定します。
- 3) enabled 属性                    ポリシー適用設定。true を設定します。
- 4) async 属性                      廃止された項目。false 固定です。

<DisplayName> element

<VerifyAPIKey> attributes の name 属性と同じにしておきます。  
Policy Editor 内で表示されるのはこの値です。

<APIKey> element

- 1) ref 属性                         API キー情報の取得先を指定します。

#### A1.2. OAuth v2.0 (GenerateAuthorizationCode) XML 仕様

<OAuthV2> attributes

- 1) name 属性                      ポリシー名。<DisplayName> element と同じにしておきます。
- 2) continueOnError 属性        ポリシー失敗時にエラーを返すよう false を設定します。
- 3) enabled 属性                    ポリシー適用設定。true を設定します。
- 4) async 属性                      廃止された項目。false 固定です。

<DisplayName> element

<OAuthV2> attributes の name 属性と同じにしておきます。  
Policy Editor 内で表示されるのはこの値です。

<Operation> element

ポリシーの Operation を設定します。「GenerateAuthorizationCode」と設定します。

<GenerateResponse> element

- 1) enabled 属性                    true を設定すると、レスポンスを生成して返します。

false を指定するとレスポンスは返されません。

#### <ExpiresIn> element

アクセストークン、リフレッシュトークン、認証コードの有効期限をミリ秒単位で設定します。

有効期限はシステム生成値+ExpiresIn の値となります。

ExpiresIn が-1 の場合、有効期限は無限となります。

ExpiresIn が未設定の場合、システムに設定されたデフォルト値となります。

有効期限はフロー変数として指定することもでき、その場合はデフォルト値よりもフロー変数の値が優先されます。

#### <ClientId> element

クライアント ID を指定します。クエリパラメーターや http ヘッダーを指定できます。

#### <RedirectUri> element

リダイレクト URI を指定します。

#### <ResponseType> element

response\_type を指定します。

#### <State> element

ステータス情報を指定します。クエリパラメーター、http ヘッダーを指定できます。

#### <Attributes><Attribute> element

AccessToken や AuthorizationCode にカスタム属性を追加します。

ユーザーを識別するための ID など任意情報を Access Token に紐付ける設定です。

- 1) name 属性                      attribute の名前です。
- 2) ref 属性                        attribute の値です。
- 3) display 属性                    true に設定すると、attribute はレスポンスに表示されません。  
デフォルトは false です。

#### 【記述例】

```
<Attributes>
  <Attribute name="apirole" ref="request.queryparam.apirole" display="false"/>
  <Attribute name="id" ref="request.queryparam.id" display="false"/>
</Attributes>
```

#### 情報取得系 element の指定方法

指定種別	記述方法 (XXXXXXX は、element 毎に書き換える)
http ヘッダー	request.header.XXXXXXX
クエリパラメーター	request.queryparam.XXXXXXX
フォームパラメーター	request.formparam.XXXXXXX

### A1.3. OAuth v2.0 (GenerateAccessToken) XML 仕様

#### <OAuthV2> attributes

- 1) name 属性                      ポリシー名。〈DisplayName〉 element と同じにしておきます。
- 2) continueOnError 属性      ポリシー失敗時にエラーを返すよう false を設定します。
- 3) enabled 属性                  ポリシー適用設定。true を設定します。
- 4) async 属性                    廃止された項目。false 固定です。

〈DisplayName〉 element

〈OAuthV2〉 attributes の name 属性と同じにしておきます。  
Policy Editor 内で表示されるのはこの値です。

〈Operation〉 element

ポリシーの Operation を設定します。「GenerateAccessToken」と設定します。

〈SupportedGrantTypes〉 element

サポートする Grant 種別を指定します (authorization\_code, client\_credentials, implicit, password)。今回は、「authorization\_code」を設定します。

〈GenerateResponse〉 element

- 1) enabled 属性                  true を設定すると、レスポンスを生成して返します。  
false を指定するとレスポンスは返されません。

〈ExpiresIn〉 element

アクセストークン、リフレッシュトークン、認証コードの有効期限をミリ秒単位で設定します。  
有効期限はシステム生成値+ExpiresIn の値となります。

ExpiresIn が-1 の場合、有効期限は無限となります。

ExpiresIn が未設定の場合、システムに設定されたデフォルト値となります。

有効期限はフロー変数として指定することもでき、その場合はデフォルト値よりもフロー変数の値が優先されます。

〈GrantType〉 element

Grant 種別の取得先を指定します。

http ヘッダー、クエリパラメーター、フォームパラメーター (デフォルト) を指定できます。

〈Code〉 element

AuthorizationCode を指定します。

http ヘッダー、クエリパラメーター、フォームパラメーター (デフォルト) を指定できます。

〈RedirectUri〉 element

リダイレクト URI を指定します。

〈Scope〉 element

スコープの取得先を指定します。

〈State〉 element

ステータス情報を指定します。http ヘッダー、クエリパラメーターを指定できます。

情報取得系 element の指定方法

指定種別	記述方法 (XXXXXX は、element 毎に書き換える)
http ヘッダー	request.header.XXXXXX
クエリパラメーター	request.queryparam.XXXXXX

## A1.4. OAuth v2.0 (RefreshAccessToken) XML 仕様

## 〈OAuthV2〉 attributes

- 1) name 属性                      ポリシー名。〈DisplayName〉 element と同じにしておきます。
- 2) continueOnError 属性        ポリシー失敗時にエラーを返すよう false を設定します。
- 3) enabled 属性                    ポリシー適用設定。true を設定します。
- 4) async 属性                      廃止された項目。false 固定です。

## 〈DisplayName〉 element

〈OAuthV2〉 attributes の name 属性と同じにしておきます。  
Policy Editor 内で表示されるのはこの値です。

## 〈ExternalAuthorization〉 element

外部認証を利用する場合、true を設定します。(デフォルト値は false)

## 〈Operation〉 element

ポリシーの Operation を設定します。「RefreshAccessToken」と設定します。

## 〈GenerateResponse〉 element

- 1) enabled 属性                    true を設定すると、レスポンスを生成して返します。  
false を指定するとレスポンスは返されません。

## 〈ExpiresIn〉 element

アクセストークン、リフレッシュトークン、認証コードの有効期限をミリ秒単位で設定します。  
有効期限はシステム生成値+ExpiresIn の値となります。

ExpiresIn が-1 の場合、有効期限は無限となります。

ExpiresIn が未設定の場合、システムに設定されたデフォルト値となります。

有効期限はフロー変数として指定することもでき、その場合はデフォルト値よりもフロー変数の値が優先されます。

## 〈GrantType〉 element

Grant 種別の取得先を指定します。

http ヘッダー、クエリパラメーター、フォームパラメーター (デフォルト) を指定できます。

## 〈ResponseType〉 element

response\_type を指定します。

## 〈Attributes〉×〈Attribute〉 element

AccessToken や AuthorizationCode にカスタム属性を追加します。

ユーザーを識別するための ID など任意情報を Access Token に紐付ける設定です。

- 1) name 属性                      attribute の名前です。
- 2) ref 属性                         attribute の値です。
- 3) display 属性                    true に設定すると、attribute はレスポンスに表示されません。  
デフォルトは false です。

【記述例】





```
<Attributes>  
  <Attribute name="apirole" ref="request.queryparam.apirole" display="false"/>  
  <Attribute name="id" ref="request.queryparam.id" display="false"/>  
</Attributes>
```

情報取得系 element の指定方法

指定種別	記述方法 (XXXXXX は、element 毎に書き換える)
http ヘッダー	request.header.XXXXXX
クエリパラメーター	request.queryparam.XXXXXX
フォームパラメーター	request.formparam.XXXXXX

## A2. MEDIATION

以下の Policy の XML 仕様について説明します。

 JSON to XML	JSON 形式を XML 形式に変換します。
 XML to JSON	XML 形式を JSON 形式に変換します。
 Assign Message	HTTP Request または Response メッセージの作成・修正が可能です。
 Extract Variables	リクエストまたはレスポンスが指定した条件（URI Path や Query Param 等）と一致した場合、変数を指定して追加することができます。

### A2.1. JSON to XML XML 仕様

<JSONToXML> attributes

- 1) name 属性                      ポリシー名。<DisplayName> element と同じにしておきます。
- 2) continueOnError 属性        ポリシー失敗時にエラーを返すよう false を設定します。
- 3) enabled 属性                    ポリシー適用設定。true を設定します。
- 4) async 属性                      廃止された項目。false 固定です。

<DisplayName> element

<JSONToXML> attributes の name 属性と同じにしておきます。

Policy Editor 内で表示されるのはこの値です。

<Source> element

XML に変換したい JSON が含まれている変数、request、response を指定します。

<OutputVariable> element

変換した XML を格納する変数、request、response を指定します。

<Options> element

JSON→XML 変換のオプション設定です。

<Options>element を使用する場合は、<Format>element を使用できません。

主な Options の詳細設定は以下の通りです。

- 1) <Options>/<RecognizeNumber> element  
true の場合は数字を数字として扱い、false の場合は文字列として扱います。
- 2) <Options>/<RecognizeBoolean> element  
true の場合は「true/false」を boolean 型として扱い、false の場合は文字列として扱います。
- 3) <Options>/<RecognizeNull> element  
true の場合は空値を null として扱い、false の場合は{}（空）として扱います。
- 4) <Options>/<NullValue> element  
Null 値を変換するときの文字列を指定する。デフォルトは「NULL」となっています。

<Format> element

4 つの変換フォーマット（xml.com, yahoo, google, badgerFish）が用意されています。

<Format>element を使用する場合は<Options>element を使用できません。

1) xml.com

```
<RecognizeNull>true</RecognizeNull>  
<TextNodeName>#text</TextNodeName>  
<AttributePrefix>@</AttributePrefix>
```

2) yahoo

```
<RecognizeNumber>true</RecognizeNumber>  
<TextNodeName>content</TextNodeName>
```

3) google

```
<TextNodeName>$t</TextNodeName>  
<NamespaceSeparator>$</NamespaceSeparator>  
<TextAlwaysAsProperty>true</TextAlwaysAsProperty>
```

4) badgerFish

```
<TextNodeName>$</TextNodeName>  
<TextAlwaysAsProperty>true</TextAlwaysAsProperty>  
<AttributePrefix>@</AttributePrefix>  
<NamespaceSeparator>:</NamespaceSeparator>  
<NamespaceBlockName>@xmlns</NamespaceBlockName>  
<DefaultNamespaceNodeName>$</DefaultNamespaceNodeName>
```

## A2.2. XML to JSON XML 仕様

<XMLtoJson> attributes

- 1) name 属性                      ポリシー名。<DisplayName> element と同じにしておきます。
- 2) continueOnError 属性        ポリシー失敗時にエラーを返すよう false を設定します。
- 3) enabled 属性                    ポリシー適用設定。true を設定します。
- 4) async 属性                      廃止された項目。false 固定です。

<DisplayName> element

<XMLtoJSON> attributes の name 属性と同じにしておきます。

Policy Editor 内で表示されるのはこの値です。

<Source> element

JSON に変換したい XML が含まれている変数、request、response を指定します。

<OutputVariable> element

変換した JSON を格納する変数、request、response を指定します。

<Options> element

XML→JSON 変換のオプション設定です。

<Options>element を使用する場合は、<Format>element を使用できません。

主な Options の詳細設定は以下の通りです。

1) <Options>/<RecognizeNumber> element

true の場合は数字を数字として扱い、false の場合は文字列として扱います。



2) <Options>/<RecognizeBoolean> element

true の場合は「true/false」を boolean 型として扱い、false の場合は文字列として扱います。

3) <Options>/<RecognizeNull> element

true の場合は空値を null として扱い、false の場合は{}（空）として扱います。

4) <Options>/<NullValue> element

Null 値を変換するときの文字列を指定する。デフォルトは「NULL」となっています。

<Format> element

4 つの変換フォーマット（xml.com, yahoo, google, badgerFish）が用意されています。

<Format>element を使用する場合は<Options>element を使用できません。

1) xml.com

```
<RecognizeNull>true</RecognizeNull>
```

```
<TextNodeName>#text</TextNodeName>
```

```
<AttributePrefix>@</AttributePrefix>
```

2) yahoo

```
<RecognizeNumber>true</RecognizeNumber>
```

```
<TextNodeName>content</TextNodeName>
```

3) google

```
<TextNodeName>$t</TextNodeName>
```

```
<NamespaceSeparator>$</NamespaceSeparator>
```

```
<TextAlwaysAsProperty>true</TextAlwaysAsProperty>
```

4) badgerFish

```
<TextNodeName>$</TextNodeName>
```

```
<TextAlwaysAsProperty>true</TextAlwaysAsProperty>
```

```
<AttributePrefix>@</AttributePrefix>
```

```
<NamespaceSeparator>.</NamespaceSeparator>
```

```
<NamespaceBlockName>@xmlns</NamespaceBlockName>
```

```
<DefaultNamespaceNodeName>$</DefaultNamespaceNodeName>
```

### A2.3. Assign Message XML 仕様

<AssignMessage> attributes

1) name 属性                   ポリシー名。<DisplayName> element と同じにしておきます。

2) continueOnError 属性       ポリシー失敗時にエラーを返すよう false を設定します。

3) enabled 属性                ポリシー適用設定。true を設定します。

4) async 属性                 廃止された項目。false 固定です。

<DisplayName> element

<AssignMessage> attributes の name 属性と同じにしておきます。

Policy Editor 内で表示されるのはこの値です。

<AssignTo> element

Assign Message ポリシーの操作対象（リクエスト・レスポンス）の変数を定義します。



したオブジェクトへコピーします。source、AssignTo とともにリクエストである場合のみ有効です。

<Copy>/<Path> element

true の場合、source 属性で指定されたオブジェクト内のパスを<AssignTo>element で指定したオブジェクトへコピーします。source、AssignTo とともにリクエストである時場合のみ有効です。

<Copy>/<StatusCode> element

true の場合、source 属性で指定されたオブジェクトの Status-Code を<AssignTo>element で指定したオブジェクトへコピーします。source、AssignTo とともにレスポンスである時場合のみ有効です。

<Copy>/<ReasonPhrase> element

true の場合、source 属性で指定されたオブジェクトの Reason-Phrase を<AssignTo>element で指定したオブジェクトへコピーします。source、AssignTo とともにレスポンスである時場合のみ有効です。

<Remove> element

<AssignTo>element で指定したオブジェクトから値を削除します。

<Remove>/<Headers> element

<AssignTo>element で指定されたオブジェクトの HTTP ヘッダーを削除します。

<Remove>/<QueryParams> element

<AssignTo>element で指定されたオブジェクトのクエリパラメーターを削除します。

<Remove>/<FormParams> element

<AssignTo>element で指定されたオブジェクトのフォームパラメーターを削除します。

<Remove>/<Payload> element

<AssignTo>element で指定されたオブジェクトのペイロードを削除します。

<Add> element

<AssignTo>element で指定したオブジェクトへ値を追加します。

<Add>/<Headers> element

<AssignTo>element で指定されたオブジェクトに HTTP ヘッダーを追加します。

<Add>/<QueryParams> element

<AssignTo>element で指定されたオブジェクトにクエリパラメーターを追加します。

<Add>/<FormParams> element

<AssignTo>element で指定されたオブジェクトにフォームパラメーターを追加します。

<Set> element

<AssignTo>element で指定したオブジェクトへ値を設定します。

<Set>/<Headers> element

<AssignTo>element で指定されたオブジェクトの HTTP ヘッダーを設定します。

<Set>/<QueryParams> element

<AssignTo>element で指定されたオブジェクトのクエリパラメーターを設定します。

<Set>/<FormParams> element

<AssignTo>element で指定されたオブジェクトのフォームパラメーターを設定します。

<Set>/<Payload> element

<AssignTo>element で指定されたオブジェクトのペイロードを設定します。

<Set>/<Version> element

<AssignTo>element で指定されたオブジェクトの HTTP バージョンを設定します。

<Set>/<Verb> element

<AssignTo>element で指定されたオブジェクトの HTTP メソッドを設定します。  
AssignTo がリクエストである場合のみ有効です。

<Set>/<Path> element

<AssignTo>element で指定されたオブジェクト内のパスを設定します。

<Set>/<StatusCode> element

<AssignTo>element で指定されたオブジェクトの Status-Code を設定します。  
AssignTo がレスポンスである場合のみ有効です。

<Set>/<StatusCode> element

<AssignTo>element で指定されたオブジェクトのフォームパラメーターを設定します。  
AssignTo がレスポンスである場合のみ有効です。

## A2.4. Extract Variables XML 仕様

<ExtractVariables> attributes

- 1) name 属性                    ポリシー名。<DisplayName> element と同じにしておきます。
- 2) continueOnError 属性       ポリシー失敗時にエラーを返すよう false を設定します。
- 3) enabled 属性                ポリシー適用設定。true を設定します。
- 4) async 属性                 廃止された項目。false 固定です。

<DisplayName> element

<ExtractVariables> attributes の name 属性と同じにしておきます。Policy Editor 内で表示されるのはこの値です。

<Source> element

抽出の対象となる変数を指定します。

- 1) clearPayload 属性        true を指定すると、抽出処理後に指定されたペイロードをクリアします。

<VariablePrefix> element

抽出した値を格納する変数名の接頭語を指定します。

<IgnoreUnresolvedVariables> element

true を指定すると、解決できない参照変数は null として処理されます。false を指定すると、解決できない参照変数はエラーとして処理されます。

<URIPath> element

リクエストメッセージの指定された URI パスからパターンに一致する値を抽出します。  
レスポンスメッセージの場合には動作しません。

- 1) ignoreCase 属性        true を指定すると、パターンにマッチした場合に無視します。

<QueryParam> element

リクエストメッセージの指定されたクエリパラメーターからパターンに一致する値を抽出します。  
レスポンスメッセージの場合には動作しません。

- 1) name 属性                クエリパラメーターの名前を指定します。同じ名前のクエリパラメーターが複数ある場合はインデックスを利用します。

<Header> element

指定されたリクエストまたはレスポンスの http ヘッダーから値を抽出します。

- 1) name 属性                      http ヘッダーの名前を指定します。同じ名前の http ヘッダーが複数ある場合はインデックスを利用します。

#### <FormParam> element

指定されたリクエストまたはレスポンスのフォームパラメーターから値を抽出します。

- 1) name 属性                      フォームパラメーターの名前を指定します。

#### <Variable> element

指定された変数名から値を抽出します。

- 1) name 属性                      変数の名前を指定します

#### <XMLPayload> element

XML 形式のメッセージから値を抽出します。

- 1) stopPayloadProcessing 属性    true を指定すると、1 つだけ値を抽出します。デフォルトは false です。

#### <XMLPayload>/<Namespaces> element

Xpath の namespaces を指定します。

- 1) prefix 属性                      namespaces の接頭語を指定します。

#### <XMLPayload>/<Variable> element

抽出された値が格納される変数名を指定します。

- 1) name 属性                      変数の名前を指定します。
- 2) type 属性                      変数の値のデータ型を指定します。

#### <XMLPayload>/<Variable>/<XPath> element

データ抽出元となる XML 形式メッセージのパスを指定します。

## A3. EXTENSION

以下の Policy の XML 仕様について説明します。

 Service Callout	外部サービス呼び出します。
---	---------------

### A3.1. Service Callout XML 仕様

<ServiceCallout> attributes

- 1) name 属性                      ポリシー名。<DisplayName> element と同じにしておきます。
- 2) continueOnError 属性        ポリシー失敗時にエラーを返すよう false を設定します。
- 3) enabled 属性                    ポリシー適用設定。true を設定します。
- 4) async 属性                      廃止された項目。false 固定です。

<DisplayName> element

<ServiceCallout> attributes の name 属性と同じにしておきます。  
Policy Editor 内で表示されるのはこの値です。

<Request> element

- 1) variable 属性                    request メッセージを含む値の名前を設定します。
- 2) clearPayload 属性                true を設定すると、request メッセージで使用されるメモリを解放するために、リクエストが http ターゲットに送信された後、request メッセージを含む変数をクリアします。ここでは false を設定します。

<Request>/<IgnoreUnresolvedVariables> element

true を設定すると、リクエストでのどんな未解決変数エラーも無視します。  
ここでは false を設定します。

<Request>/<Remove>, <Copy>, <Add>, <Set>

上記の定義仕様は、Assign Message Policy と同じです。

<Response> element

外部サービスからのレスポンスを含む変数名を指定します。

<Timeout> element

外部サービスからの応答の待ち時間を指定します（ミリ秒単位）。

<HTTPTargetConnection> element

呼び出すサービス（バックエンドサービス）の接続情報を指定します。

<HTTPTargetConnection>/<URL> element

呼び出すサービス（バックエンドサービス）の URL を指定します。

<HTTPTargetConnection>/<SSLInfo> element

呼び出すサービス（バックエンドサービス）への TLS/SSL の設定をします。

<HTTPTargetConnection>/<Properties> element

HTTP トランスポートのプロパティの設定をします。

<HTTPTargetConnection>/<LoadBalancer> element

ロードバランサーの設定をします。

<LocalTargetConnection> element

ローカルプロキシを呼び出す設定をします。指定方法には以下の2つがあります。

<LocalTargetConnection>/<APIProxy> element

<LocalTargetConnection>/<ProxyEndpoint> element

呼び出すローカルプロキシのAPIプロキシ名と対応するプロキシエンドポイントを指定します。

<LocalTargetConnection>/<Path> element

呼び出すローカルプロキシをパスで指定します。