IBM i コンテンツ　(10月版)

RPGⅢがメインのユーザー様における REST API利用方法について

日本アイ・ビー・エム株式会社
テクノロジー事業本部
Power Systems テクニカルセールス
澤田英寿

IBM i におけるREST APIを利用したアプリ/データ連携

IBM iに標準装備されたWebアプリケーション・サーバーを利用して基幹データをAPI公開！

実績のある基幹業務ロジックを使用することでセキュアなデータ・アクセス環境を実現

REST APIを利用することで
クライアント・アプリケーションは
自由に開発可能

API呼び出し

HTTP/HTTPS

IBM i に標準装備の
Webアプリケーション・サーバー

Webサービス・エンジン

統合Web アプリケーション・サーバー

- ILE RPG
- ILE COBOL
- JAVA
- OSS

既存プログラムによるセキュアなデータ・アクセス

IBM i

Db2 サーバー ジョブ

SQL

Db2 for i

クライアント・アプリケーションの場所も
オンプレミス/クラウドを選ばず配置可能

IBM iユーザーからのFAQ

基幹業務ロジックが、RPGⅢをメインで利用している場合は、REST APIを利用したアプリ・データ連携ができないのでしょうか。

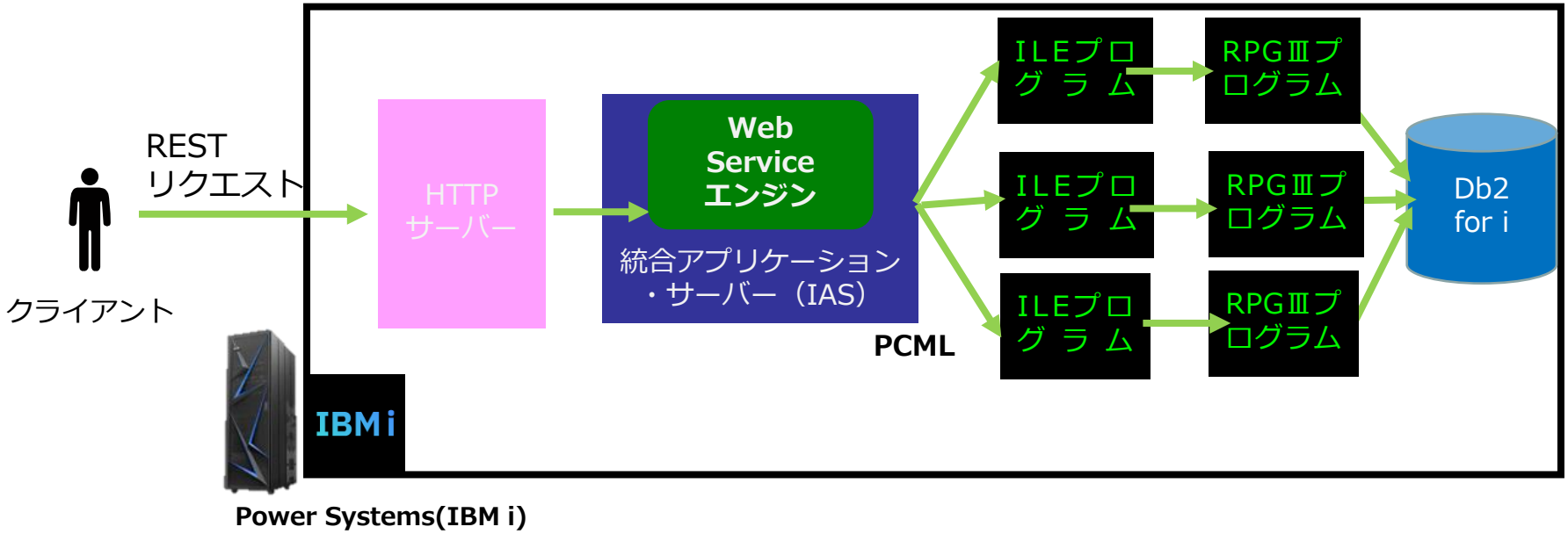
下記の２つの解決策があります

1. IBM i のプログラム (RPGⅢ) からILE RPGを経由して、REST APIサービス化する。
2. IBM i のプログラム (RPGⅢ) をILE RPGに変換して、REST API化する。

次ページ以降に、解説します。

1. IBM i のプログラム (RPGⅢ) からILE RPGを経由して、REST APIサービス化する

ILEプログラムで実装されたサービスをREST APIとして外部公開し、業務アプリのRPGⅢのPGMを呼び出す(全てのRPGをILE-RPGに変更する必要はない)
RPGⅢとILE RPGは普通に相互呼び出し可能です。



2. IBM i のプログラム (RPGⅢ) をILE RPGに変換して、REST API化する

業務アプリのRPGⅢのソースをILE-RPGに変換して、REST API化する

RPGソース変換(CVTRPGSRC)コマンドは，RPGⅢまたはRPG/400ソース・コードをILE RPGソース・コードに変換することができます。

例　　**RPGⅢ**ソースの**RPGⅣ**ソースへの変換（下記のようなコマンドで自動変換されます）
CVTRPGSRC FROMFILE(MYLIB/QRPGSRC) FROMMBR(XMPLE1) TOFILE(MYLIB/QRPGLESRC) TOMBR(*FROMMBR) EXPCPY(*NO) CVTRPT(*YES) LOGFILE(MYLIB/QRNCVTLG)

・このコマンドはファイルMYLIB/QRPGSRC中のRPGⅢソース・メンバーXMPLE1を，ファイルMYLIB/QRPGLESRC中の同じ名前のRPGⅣソース・メンバーに変換します。
・RPGⅢプログラムの/COPYステートメントは展開されません。これらは，RPGⅣプログラムで/COPYステートメントとして残ります。報告書が印刷されます。
　それぞれの変換の状況は，ファイルMYLIB/QRNCVTLGに配置されます。

このページ以降は、

「IBM iのREST API使用法の簡易ガイド」

になります。

IBM i のREST API使用法の簡易ガイド

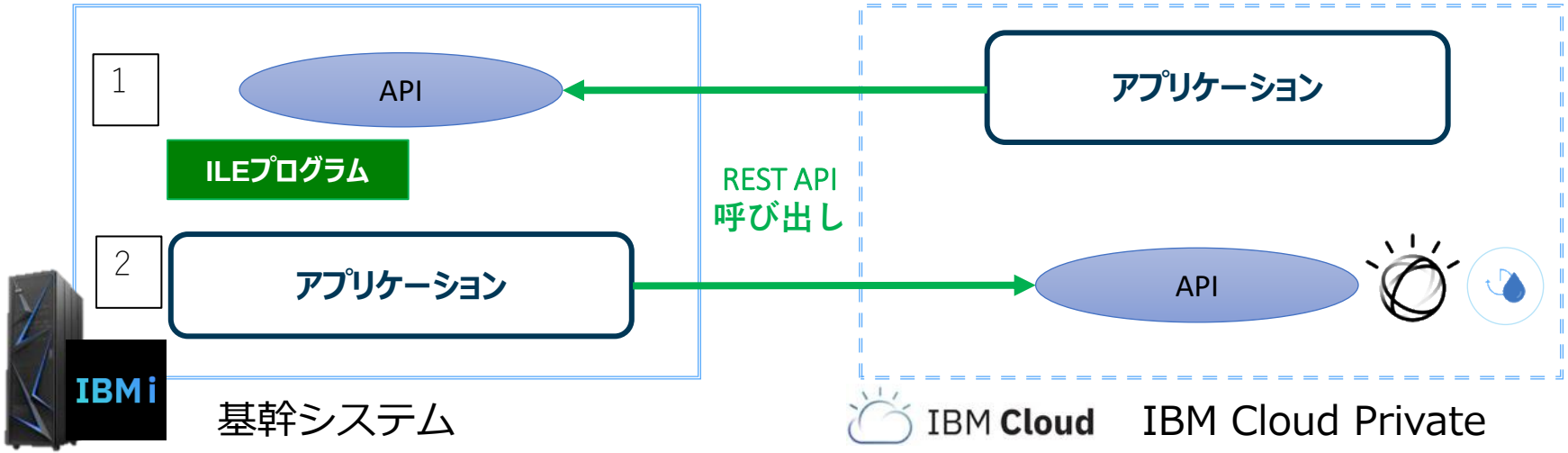
「IBM i でのREST API使用法の簡易ガイド」

目次

1. ILE RPGによるREST API活用

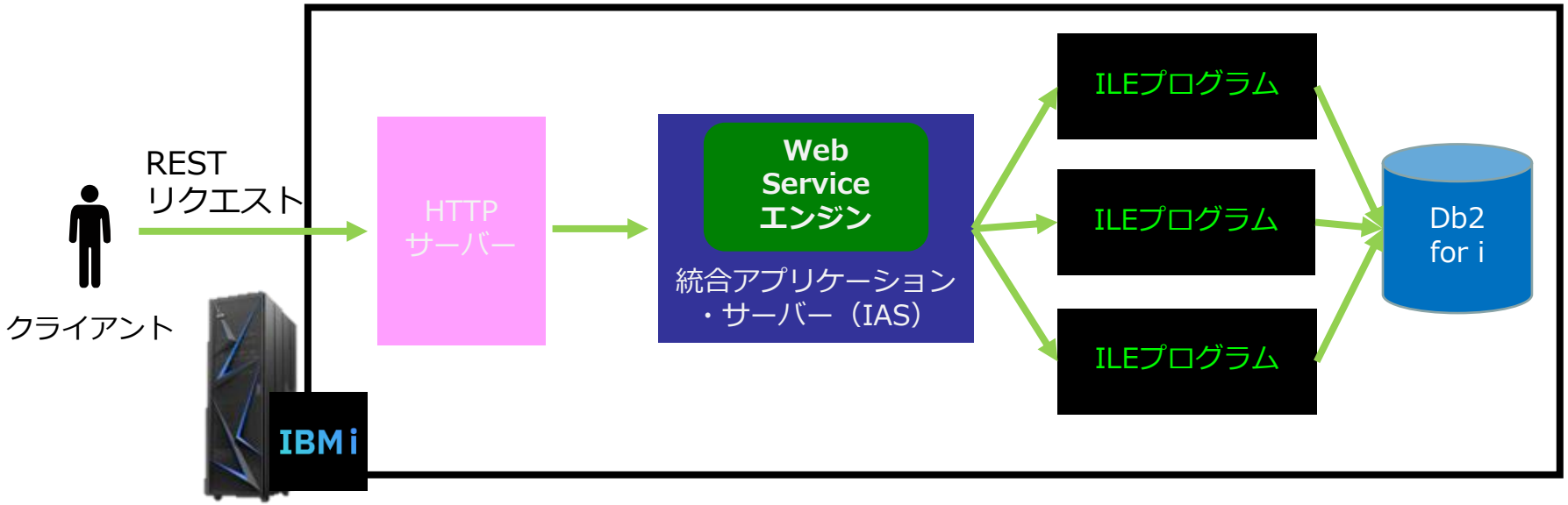
IBM i がAPIサービスと連携する2つのアプローチ

1. [外部 → IBM i] IBM i のILEプログラム(ILE RPG/ILE COBOL等)をREST APIサービス化し、外部のアプリケーションから呼び出す
2. [IBM i → 外部] IBM i のILEプログラム(ILE RPG/ILE COBOL等)から、外部のAPIを呼び出す



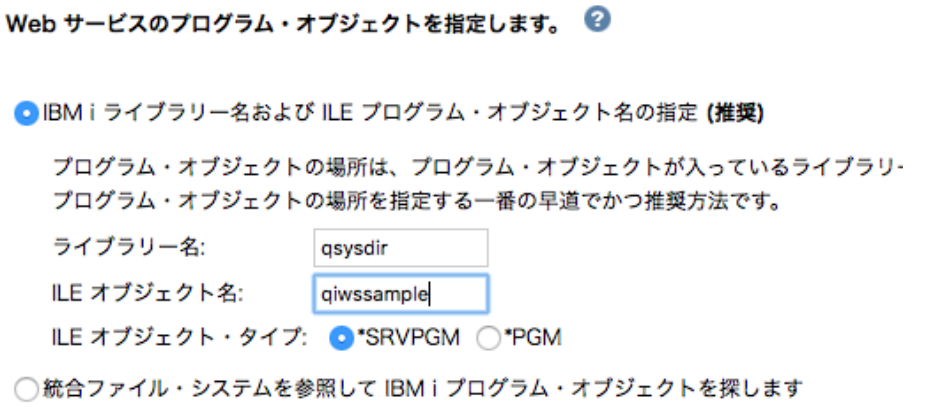
1. [外部 → IBM i]
IBM i のILEプログラム(ILE RPG/ILE COBOL等)をREST API サービス化し、外部のアプリケーションから呼び出す

IBM iに標準搭載されている統合アプリケーション・サーバー(IAS)で RESTをサポート
ILEプログラムで実装されたサービスをREST APIとして外部公開可能



IBM i統合アプリケーション・サーバーによる Web Serviceエンジンサポート

IBM i 7.2以降でサポート　（IBM i OS標準機能）
IBM i 7.2 HTTP グループPTF レベル6以降
WebサービスをデプロイするIASインスタンスはWeb Administration for i (http://<system>:2001/HTTPAdmin) からウィザードで作成可能
関連HTTPサーバー・インスタンスも自動作成



呼び出し先ILEプログラムの指定

APIアクセス・メソッドの定義

統合Webサービス・サーバーへのRESTサービスの作成ステップ

4点について検討、ウィザードで指定

・RESTサービスとして展開するILEプログラムの検討
・プログラムでの処理に対するHTTPプロトコルのメソッドの検討
・URIの検討（URI : Uniform Resource Identifier）
・入出力に関する検討
　入出力パラメーターとして受け入れる形式
　ILEプログラムの入力パラメーターとして渡す値の指定方法
　ILEプログラムの出力パラメーターとして応答コードやHTTPヘッダーを含ませるか

統合Webサービス・サーバーへのRESTサービスの作成ステップ

4点について検討、ウィザードで指定

・RESTサービスとして展開するILEプログラムの検討
・プログラムでの処理に対するHTTPプロトコルのメソッドの検討
・URIの検討
・入出力に関する検討
入出力パラメーターとして受け入れる形式
ILEプログラムの入力パラメーターとして渡す値の指定方法
ILEプログラムの出力パラメーターとして応答コードおよびHTTPヘッダーを含ませるか

- 対象プログラムでのCRUD操作の特定
  - **C**reate: データの作成
  - **R**ead: データの参照
  - **U**pdate: データの更新
  - **D**elete: データの削除
- リソースに対するCRUD操作とHTTPメソッドを対応付ける
  - **C**reate: **POST**
  - **R**ead: **GET**
  - **U**pdate: **PUT**
  - **D**elete: **DELETE**

統合Webサービス・サーバーへのRESTサービスの作成ステップ

4点について検討、ウィザードで指定

・RESTサービスとして展開するILEプログラムの検討
・プログラムでの処理に対するHTTPプロトコルのメソッドの検討
・URIの検討
・入出力に関する検討
入出力パラメーターとして受け入れる型
ILEプログラムの入力パラメーターとして渡す値の指定方法
ILEプログラムの出力パラメーターとして返すHTTPヘッダーを含ませるか

URI検討のポイント
- 操作対象のリソース (データ) を名詞で表現する
  - 処理内容は使用するHTTPメソッドで判別できる
- 容易に推測可能な語句を採用する

統合Webサービス・サーバーへのRESTサービスの作成ステップ

4点について検討、ウィザードで指定
- RESTサービスとして展開するILEプログラムの検討
- プログラムでの処理に対するHTTPプロシージャ・メソッドの検討
- URIの検討
- 入出力に関する検討
  入出力パラメーターとして受け入れる形式
  ILEプログラムの入力パラメーターとして渡す値の指定方法
  ILEプログラムの出力パラメーターとして応答コードやHTTPヘッダーを含めるか

- RESTサービスとしてデプロイする各プロシージャー単位で検討が必要
  - 入出力パラメーターとして受け入れる形式
    - JSON (application/json)
    - XML (application/xml)
  - 入力パラメーターの受け渡し方式
    - パス・セグメント (/cars/{color})
    - マトリックス・パラメーター (/cars;color=blue)
    - クエリー・パラメーター (/cars?color=blue)
  - 出力パラメーターとして応答コードやHTTPヘッダーを含めるか
    - アプリとしてHTTPヘッダーを返す場合、HTTPキャッシュに関するヘッダー情報を返すことが多い

統合Webサービス・サーバーへのRESTサービスの作成検討例

学生情報を処理するILEサービス・プログラム　STUDENTRSC
学生情報はテーブル STUDENTRSCライブラリーのSTUDENTDB に格納
提供プロシージャーは5つ

プロシージャー	処理内容
GETALL	学生情報を全件参照
GETBYID	指定した学生コードに該当の学生情報を参照
CREATE	新規の学生情報を登録
UPDATE	既存の学生情報を更新
REMOVE	指定した学生コードの学生情報を削除

統合Webサービス・サーバーへのRESTサービスの作成検討例

検討例
・プログラムでの処理に対するHTTPプロトコルのメソッドの検討
・URIの検討
・入出力に関する検討

URI			
/{context-root}/students			
プロシージャー	**HTTPメソッド**	**入出力**	**URI**
GETALL	GET	• 結果はJSON形式で出力 • HTTP応答コードとヘッダーも出力に含める	/students
GETBYID	GET	• 学生コードをパスセグメントで受け取る • 結果はJSON形式で出力 • HTTP応答コードとヘッダーも出力に含める	/students/{id}
CREATE	POST	• HTTP応答コードとヘッダーも出力に含める	/students
UPDATE	PUT	• HTTP応答コードを出力に含める	/students
REMOVE	DELETE	• 学生コードをパスセグメントで受け取る • HTTP応答コードを出力に含める	/students/{id}

RESTサービスとして作成するILE RPGプログラム

RESTサービスプログラムのソースは固定フォーマットでも、フリー・フォーマットでもOK

```
  //*************************************************************
  // getAll                                                     *
  //*************************************************************
P getAll          B                   EXPORT
D getAll          PI
D  students_...
D  LENGTH                       10i 0
D  students                           likeds(studentRec) dim(1000)
D                                     options(*varsize)
D  httpStatus                   10i 0
D  httpHeaders                 100a   dim(10)
 /FREE
  clear httpHeaders;
  clear students;
  students_LENGTH = 0;

  openStudentDB();

  setll *loval STUDENTDB;

  read(e) studentR;
  if (%ERROR);
    httpStatus = H_SERVERERROR;
    return;
  endif;

  dow (NOT %eof);
    students_LENGTH = students_LENGTH+1;
    students(students_LENGTH).studentID =  studentID;
    students(students_LENGTH).firstName =  firstName;
    students(students_LENGTH).lastName  =  lastName;
    students(students_LENGTH).gender    =  gender;

    read(e) studentR;
    if (%ERROR);
      httpStatus = H_SERVERERROR;
```

```
  //*************************************************************
  // getByID                                                    *
  //*************************************************************
P getByID         B                   EXPORT
D getByID         PI
D  studentID                     9a   const
D  student                            likeds(studentRec)
D  httpStatus                  10i 0
D  httpHeaders                100a   dim(10)
 /FREE
  clear httpHeaders;
  clear student;

  openStudentDB();

  chain(e) studentID STUDENTDB;
  if (%ERROR);
    httpStatus = H_SERVERERROR;
    return;
  elseif %FOUND;
    student.studentID = studentID;
    student.firstName = firstName;
    student.lastName  = lastName;
    student.gender    = gender;

    httpStatus = H_OK;
  else;
    httpStatus = H_NOTFOUND;
  endif;

  httpHeaders(1) = 'Cache-Control: no-cache, no-store';

  closeStudentDB();
 /END-FREE
P getByID         E
```

統合Webサービス・サーバーへのRESTサービスの作成

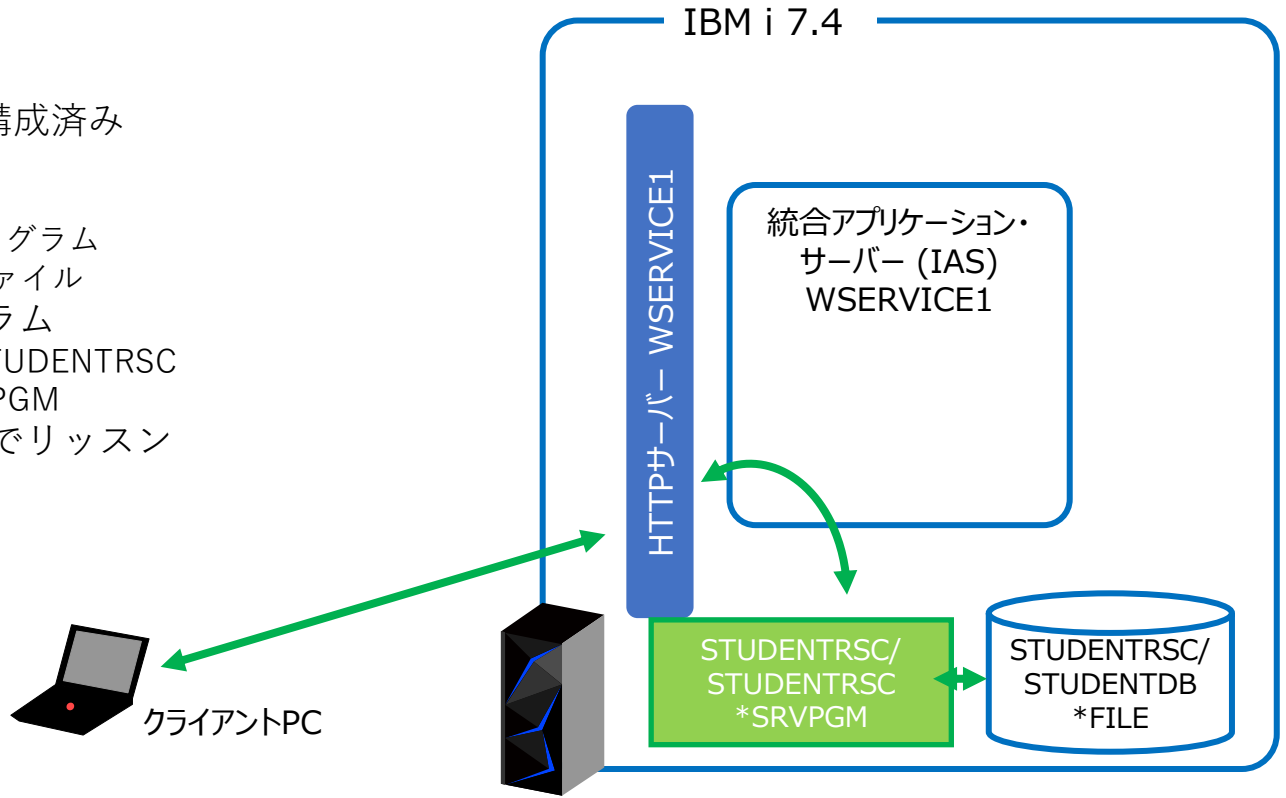
開発環境

IBM i 7.4

・全て単一のOS環境に構成済み
- HTTPサーバー
- IASインスタンス
- ILEサービス・プログラム
- データベース・ファイル

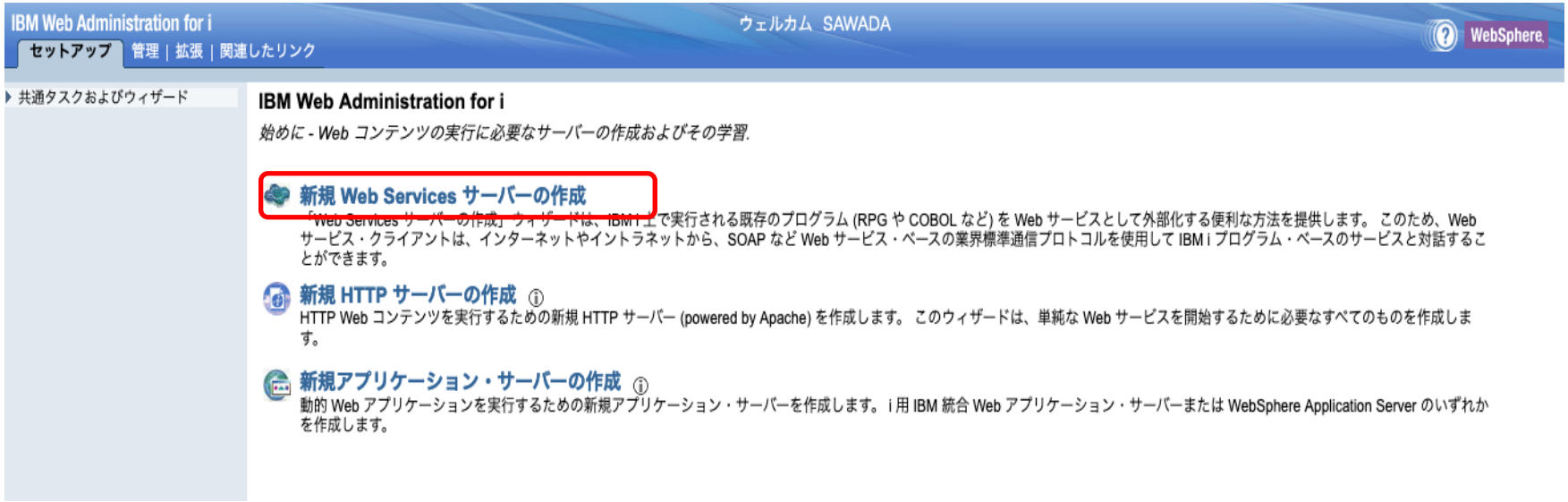
・ILEサービス・プログラム
- STUDENTRSC/STUDENTRSC
- 前掲検討例のSRVPGM

・HTTPサーバーは以下でリッスン
- HTTP: 10073



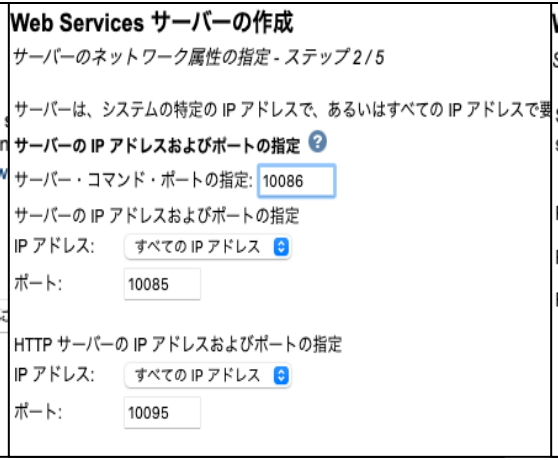
<参考>Webサービス用IASインスタンスの作成手順

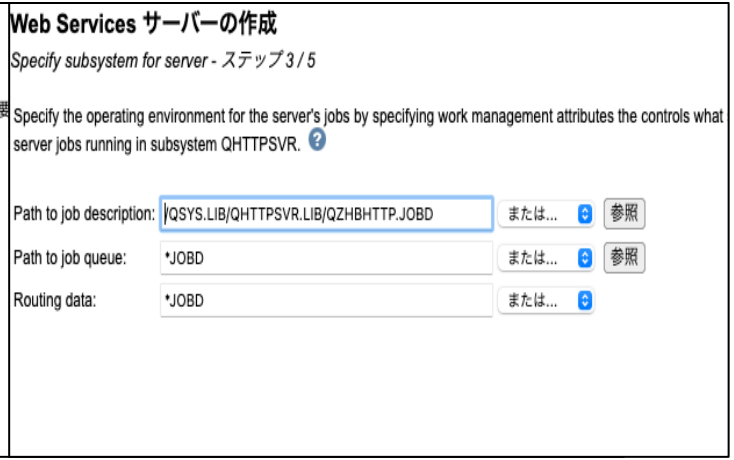
1. http://<system>:2001/HTTPAdmin にブラウザーでアクセス
2. 「セットアップ」タブの「新規　Web Serviceサーバーの作成」を選択



<参考>Webサービス用IASインスタンスの作成手順（続き）

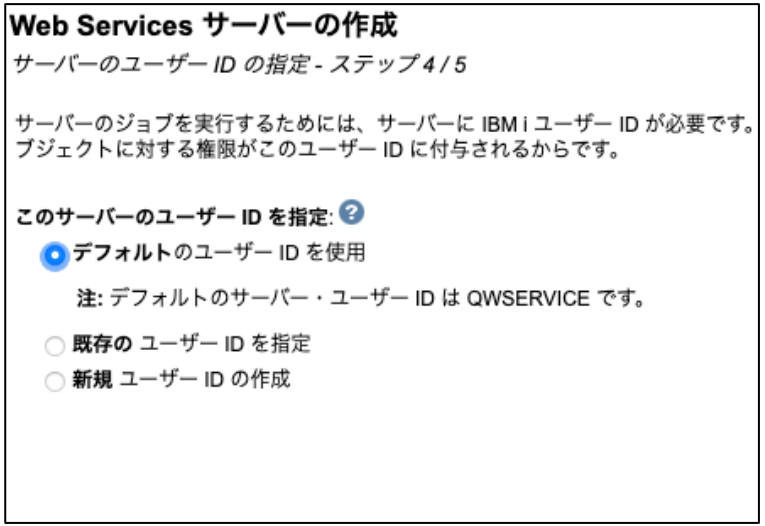
3. IASインスタンス名（およびテキスト）を入力、「Next」ボタンをクリック
4. IASインスタンスのポートを指定、「Next」ボタンをクリック
5. IASインスタンスのサブシステムを指定、「NEXT」ボタンをクリック

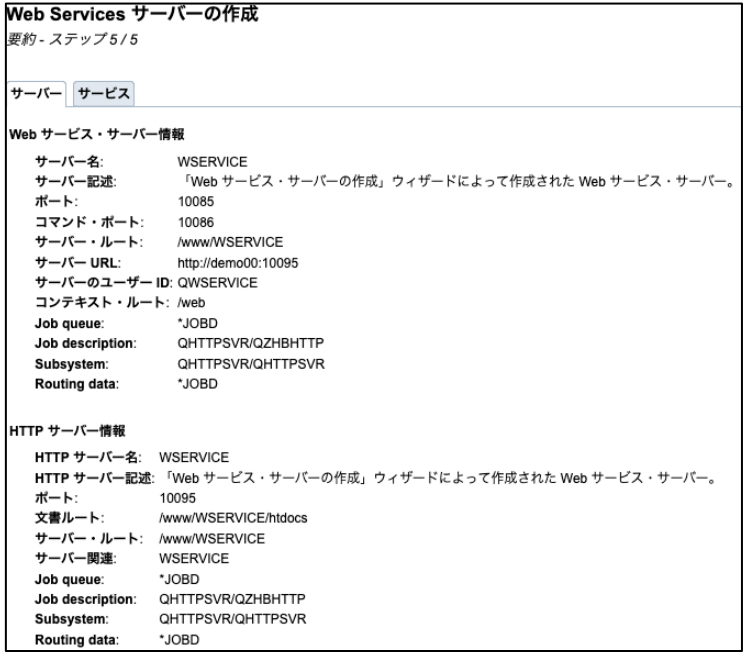




<参考>Webサービス用IASインスタンスの作成手順（続き）

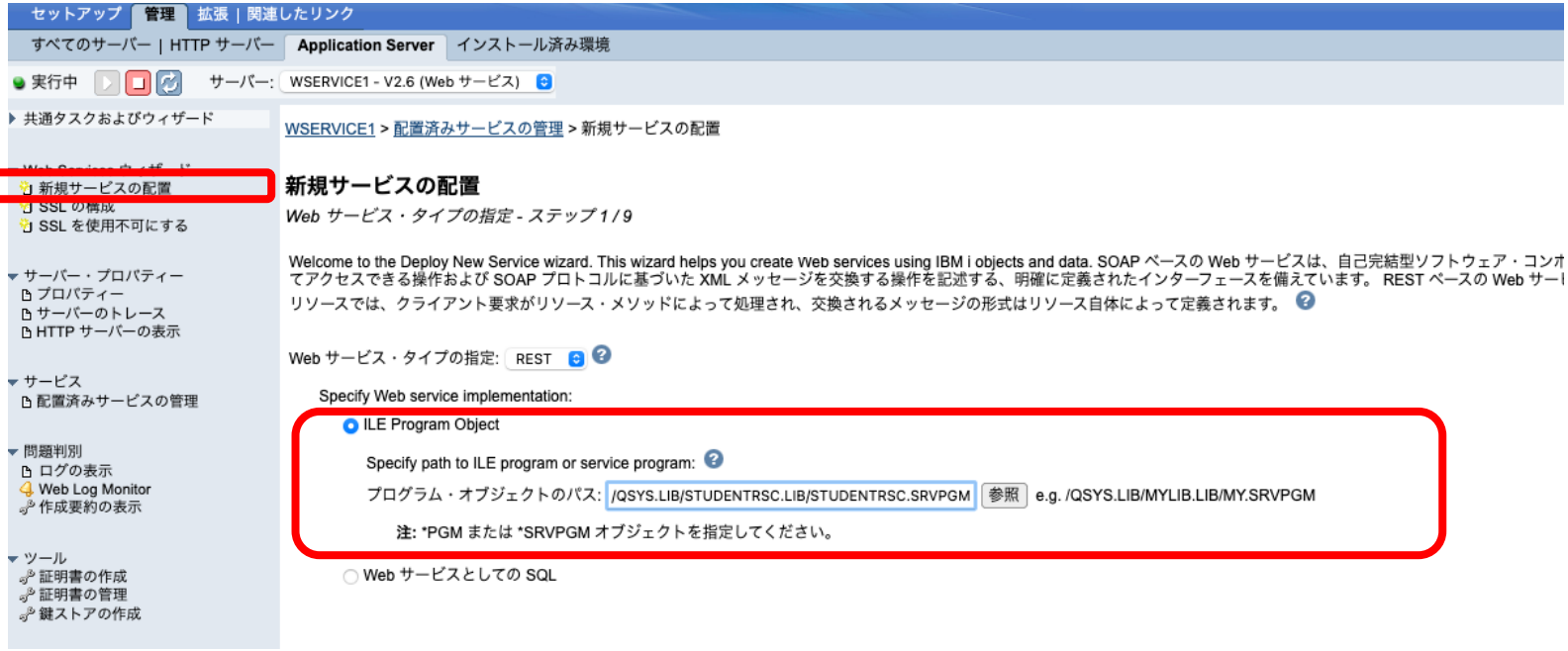
6. IASインスタンスのユーザーIDを指定、「NEXT」ボタンをクリック
7. サマリーを確認して「Finish」ボタンをクリック、作成が完了
   関連のHTTPサーバー・インスタンスも自動作成
   HTTPポートも確認可能





<参考>RESTサービスの作成手順

1. http://<system>:2001/HTTPAdmin にブラウザーでアクセス
2. 「管理」タブ→「Application Servers」タブ→サーバーを選択
3. 「Web Services ウィザード」→「新規サービスの配置」を選択
4. 「REST」を選択し、ILE PGM Objectに書きを入力し、「Next」ボタンをクリック

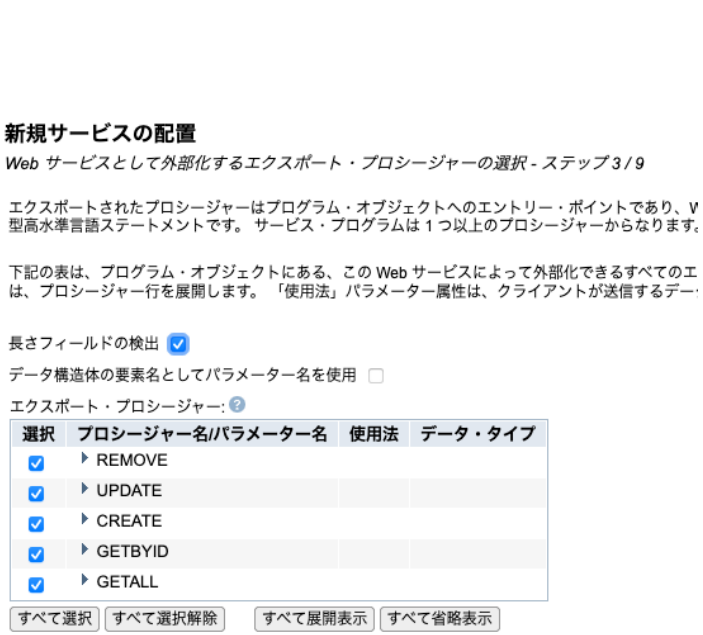


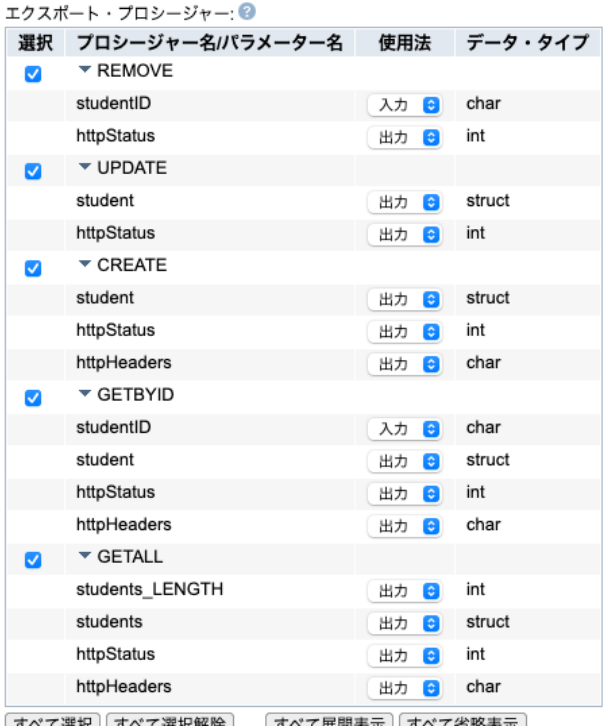
<参考>RESTサービスの作成手順（続き）

5. 「Resource name」にURI「students」を指定し、「Next」ボタンをクリック
   「URI path template」にはサービス判別のための正規表現を指定可能だが、今回は「/context-root/students」とするため指定不要

<参考>RESTサービスの作成手順（続き）

6. ILEオブジェクト内のプロシージャーの入出力パラメーターを調整・設定後、「Next」ボタンをクリック
   宣言時にCONSTを指定したパラメーターをinputとして自動認識

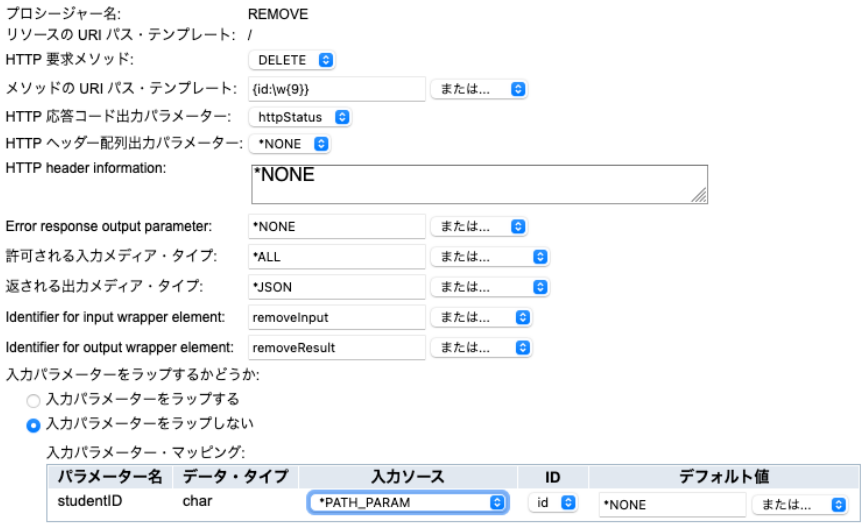




<参考>RESTサービスの作成手順（続き）

7. 各プロシージャーに対して以下を設定
・使用するHTTPメソッド
・URIパス・テンプレート（入力パラメーターをパス・セグメント等で渡す場合はここで指定）
・出力パラメーターにHTTPステータス, ヘッダーを含む場合、該当パラメーターの設定
・受け入れるメディア・タイプ（JSON, XML）
・入力パラメーターの受け渡し方法とILE側パラメーターのマッピング設定

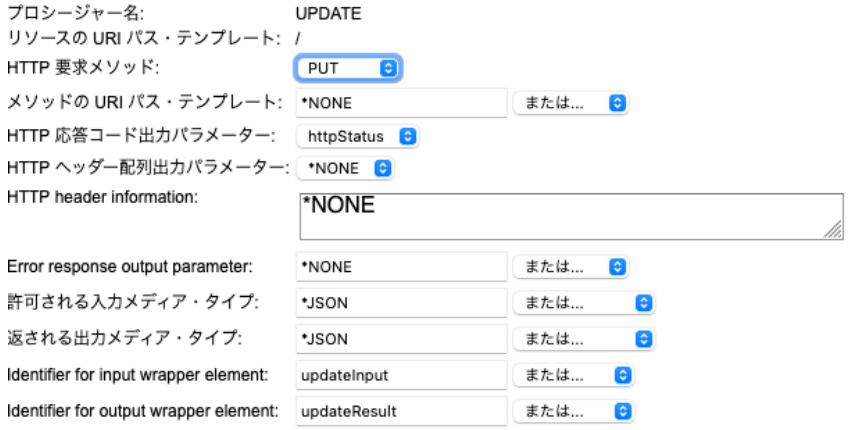
<REMOVEプロシージャーの例>



<参考>RESTサービスの作成手順（続き）

7. 各プロシージャーに対して以下を設定（続き）
・使用するHTTPメソッド
・URIパス・テンプレート（入力パラメーターをパス・セグメント等で渡す場合はここで指定）
・出力パラメーターにHTTPステータス, ヘッダーを含む場合、該当パラメーターの設定
・受け入れるメディア・タイプ（JSON, XML）
・入力パラメーターの受け渡し方法とILE側パラメーターのマッピング設定

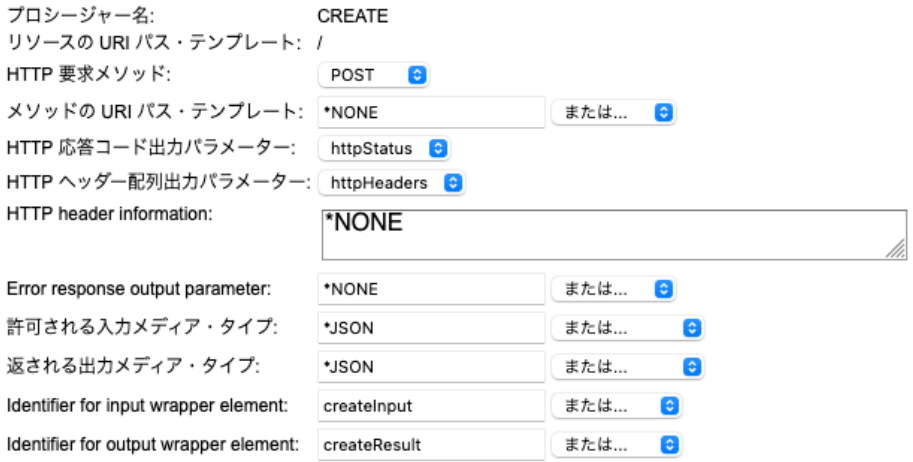
<UPDATEプロシージャーの例>



<参考>RESTサービスの作成手順（続き）

7. 各プロシージャーに対して以下を設定（続き）
・使用するHTTPメソッド
・URIパス・テンプレート（入力パラメーターをパス・セグメント等で渡す場合はここで指定）
・出力パラメーターにHTTPステータス, ヘッダーを含む場合、該当パラメーターの設定
・受け入れるメディア・タイプ（JSON, XML）
・入力パラメーターの受け渡し方法とILE側パラメーターのマッピング設定

<CREATE プロシージャーの例>



<参考>RESTサービスの作成手順（続き）

7.　各プロシージャーに対して以下を設定（続き）

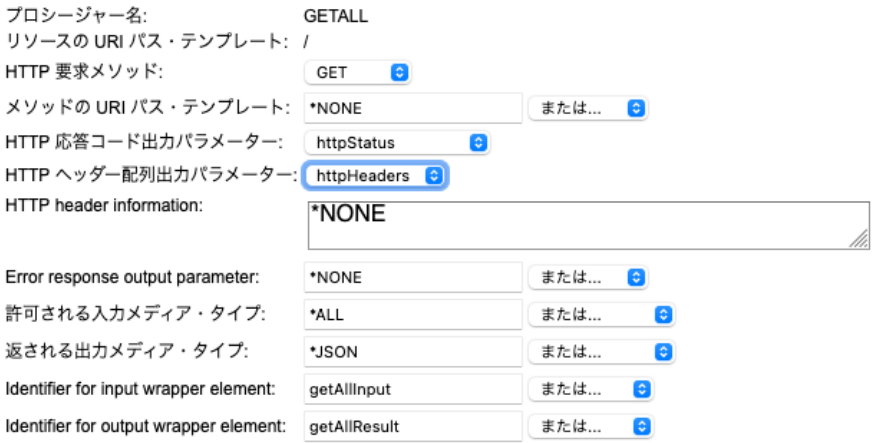
・使用するHTTPメソッド
・URIパス・テンプレート（入力パラメーターをパス・セグメント等で渡す場合はここで指定）
・出力パラメーターにHTTPステータス, ヘッダーを含む場合、該当パラメーターの設定
・受け入れるメディア・タイプ（JSON, XML）
・入力パラメーターの受け渡し方法とILE側パラメーターのマッピング設定

<GETBYIDプロシージャーの例>

<参考>RESTサービスの作成手順（続き）

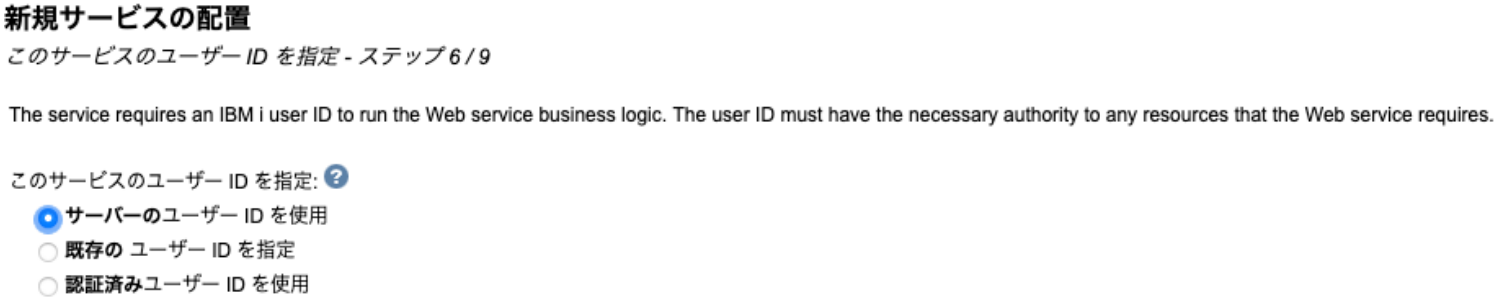
7．　各プロシージャーに対して以下を設定（続き）
・使用するHTTPメソッド
・URIパス・テンプレート（入力パラメーターをパス・セグメント等で渡す場合はここで指定）
・出力パラメーターにHTTPステータス, ヘッダーを含む場合、該当パラメーターの設定
・受け入れるメディア・タイプ（JSON, XML）
・入力パラメーターの受け渡し方法とILE側パラメーターのマッピング設定

<GETALLプロシージャーの例>



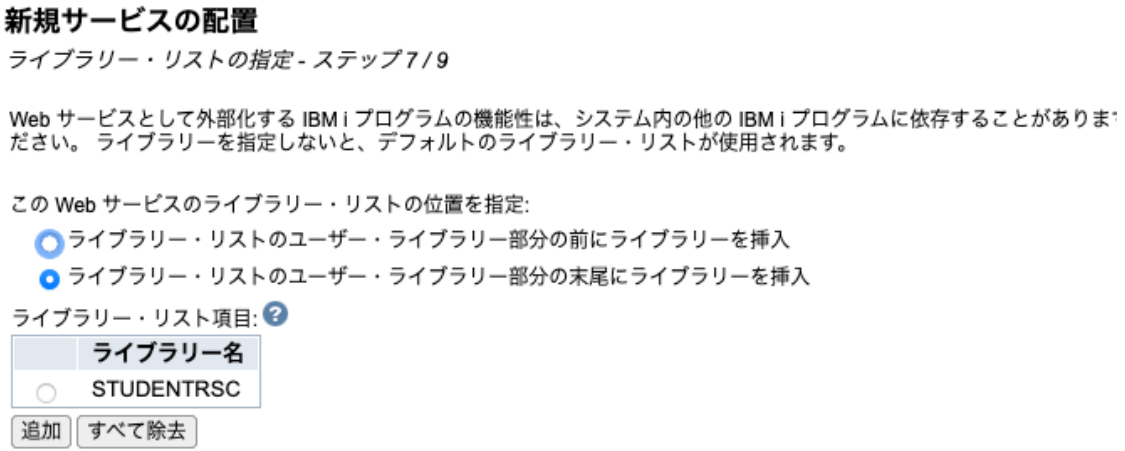
<参考>RESTサービスの作成手順（続き）

8. サービスが呼び出された際に利用されるユーザー・プロファイルを指定し、「Next」ボタンをクリック
   ホスト・サーバー・ジョブQZRCSRVSの現行ユーザーとなり、実行時のオブジェクト権限等の基点となる。
   今回はデモのためIASインスタンスと同一ユーザー・プロファイルとする



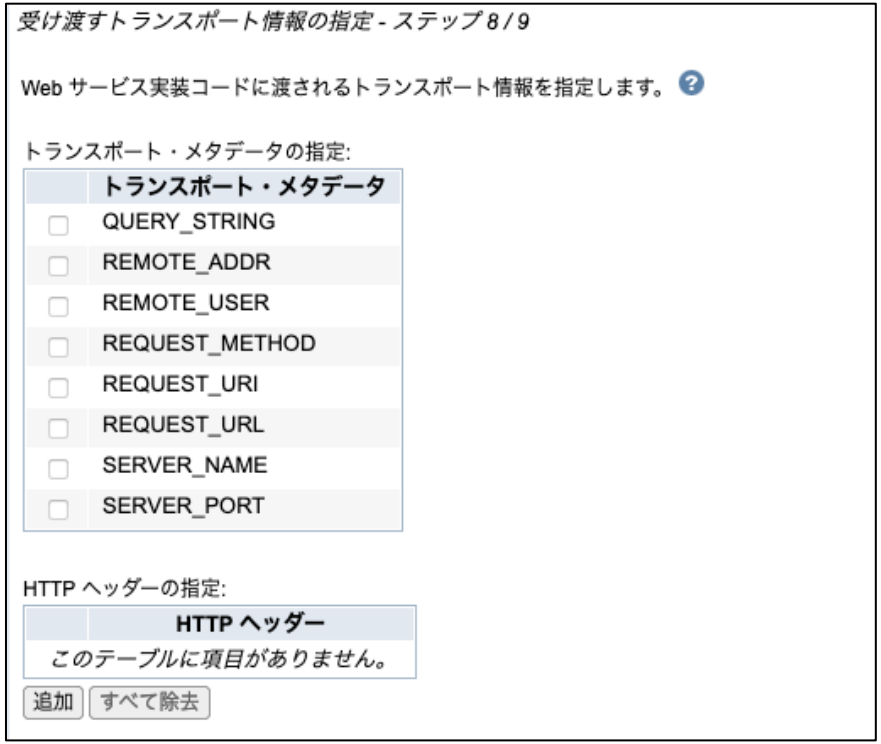
<参考>RESTサービスの作成手順（続き）

9. サービスが呼び出された際のライブラリー・リストを設定し、「Next」ボタンをクリック
   今回は特に必要ないため、追加指定はなし



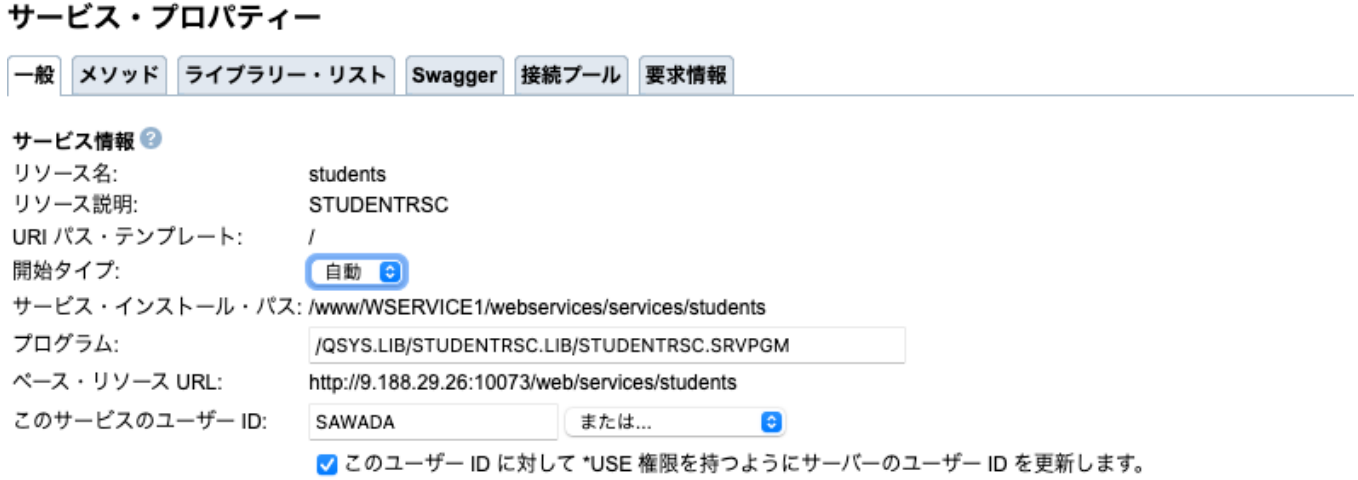
<参考>RESTサービスの作成手順（続き）

10. 受け渡すメタデータを選択し、「Next」ボタンをクリック
    今回はデフォルトのまま、特に指定しない



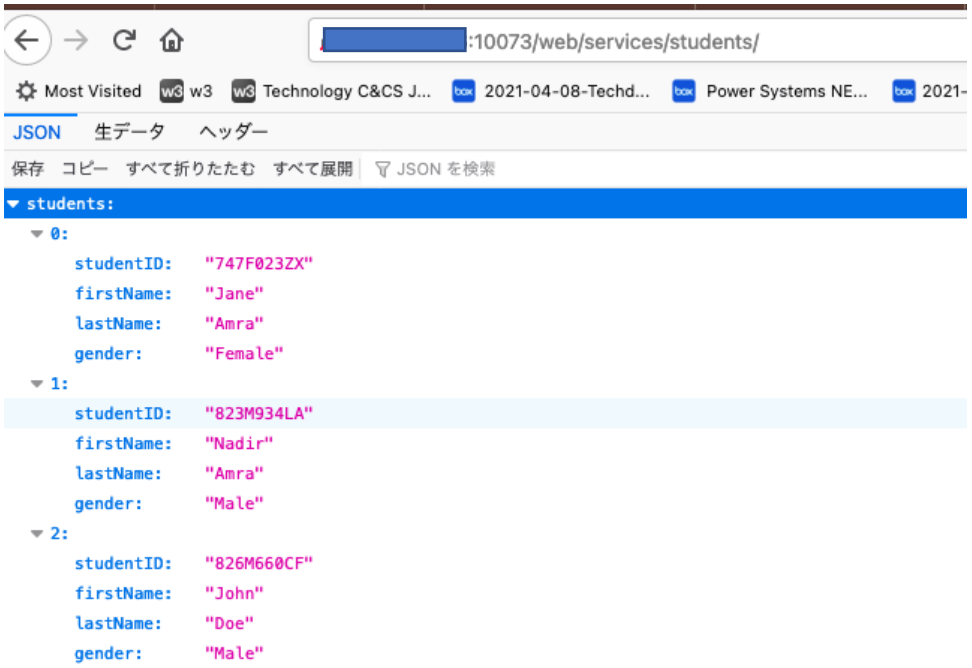
<参考>RESTサービスの作成手順（続き）

11. サマリーを確認し、「Finish」ボタンをクリック
デプロイされると、自動的に開始される
下記は作成後の、プロパティ



<参考>IAS上のRESTサービスの呼び出し

デフォルトではコンテキスト・ルートが/web/servicesとなる
例: http://<system>:10073/web/services/students
Webブラウザーを用いて確認



参考情報

IBM i アプリケーション開発
IBM Redbooks: Modernizing IBM i Applications from the Database up to the User Interface and Everything in Between
http://www.redbooks.ibm.com/abstracts/sg248185.html?Open
REST webサービス
IBM developerWorks: Building a REST service with integrated web services server for IBM i: Part 1
https://developer.ibm.com/tutorials/i-rest-web-services-server1/

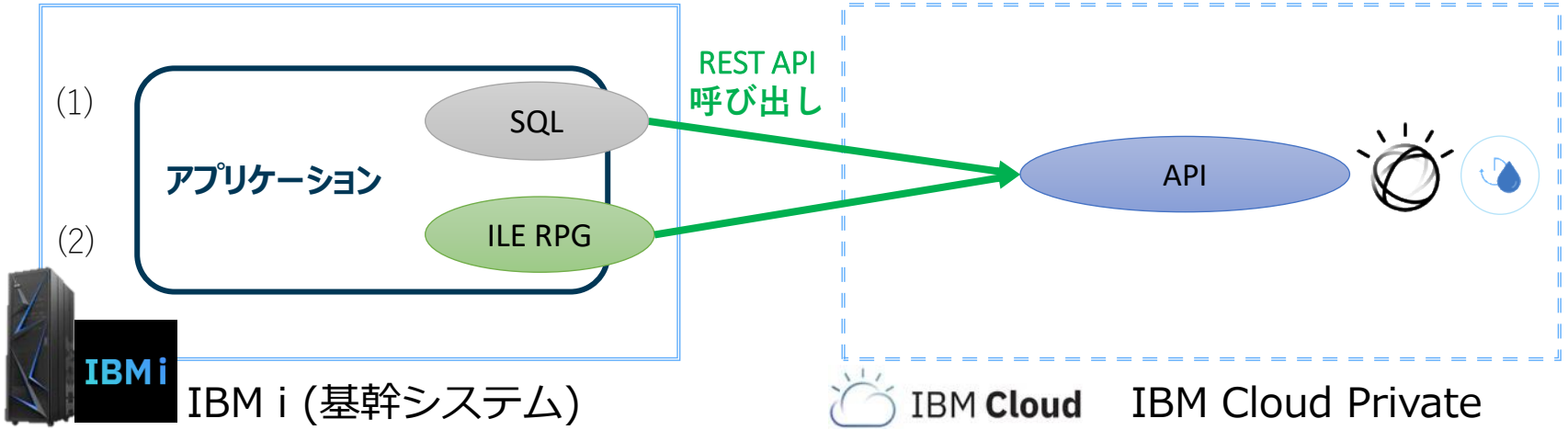
IBM developerWorks: Building a REST service with integrated web services server for IBM i: Part 2
華氏を摂氏に変換するRESTサービスのサンプル
https://developer.ibm.com/tutorials/i-rest-web-services-server2/

IBM developerWorks: Building a REST service with integrated web services server for IBM i: Part 3
学生情報をメンテナンスするRESTサービスのサンプル
https://developer.ibm.com/tutorials/i-rest-web-services-server3/

2. [IBM i → 外部] IBM i アプリケーション側から REST APIサービスを呼び出す

IBM i アプリケーション側からREST APIサービスを呼び出す2つの方法

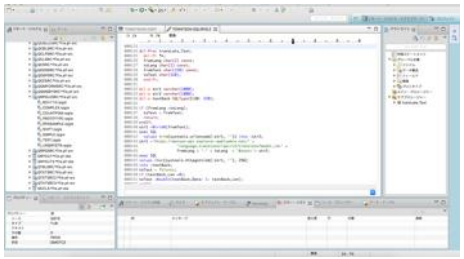
1. Db2 for i HTTPスカラー関数の活用
2. ILE RPGでのTransport APIの活用



IBM i 上のプログラムからクラウド・サービスをコール

Db2 for i のHTTPメソッド機能を活用したREST API呼び出し
「SYSTOOLS」スキーマでRESTのHTTPメソッドを呼び出すスカラー関数を提供

例）ILE RPGの組み込みSQLを使って、HTTPGETBLOB関数をREST APIをGETリクエストし、リソースを取得



開発ツール
Rational Developer for i

TOWATSON.SQLRPGLE
...
SYSTOOLS.HTTPGETBLOB
(:https://watson-api-explorer.mybluemix.net/language-translator/api/v2/translate?model_id=en-ja&text=This%20is%20a%20test. ,'')
CLOB
...

訳文の言語
原文の言語
原文
HTTP GETリクエスト
REST API Call
リソースを取得
訳文

Watson API
Language Translator

IBM Watson

Db2 for i
IBM i

IBM i
IBM Power

詳細はこちら：IBM i セミナー資料「ILE RPGからWatson APIを使用」
https://ibm.ent.box.com/s/e0zcytg2r6iw4v6op7ujpxzvryd7yht2

Db2 for i HTTPスカラー関数

IBM i のデータベース機能 (Db2 for i) でHTTPメソッドを呼び出す以下のスカラー関数をサポート

- httpGetClob
- httpGetBlob
- httpPostClob
- httpPostBlob
- httpPutClob
- httpPutBlob
- httpDeleteClob
- httpDeleteBlob

ライブラリーSYSTOOLSで提供しており、SQLで利用可能
アクセスしたいURLを入力として関数に引き渡し、CLOB、BLOBで値を受け取る

Db2 for i のHTTPメソッド機能の前提条件

- サポートOSバージョン
  IBM i 7.2以降
- 前提ライセンス・プログラム
  Java™ 1.6 以降 (5761-JV1 Option 11, 12, 14, 15)

【ご参考】エラー「 SYSTOOLSのオブジェクトが見つからない」場合の対処法

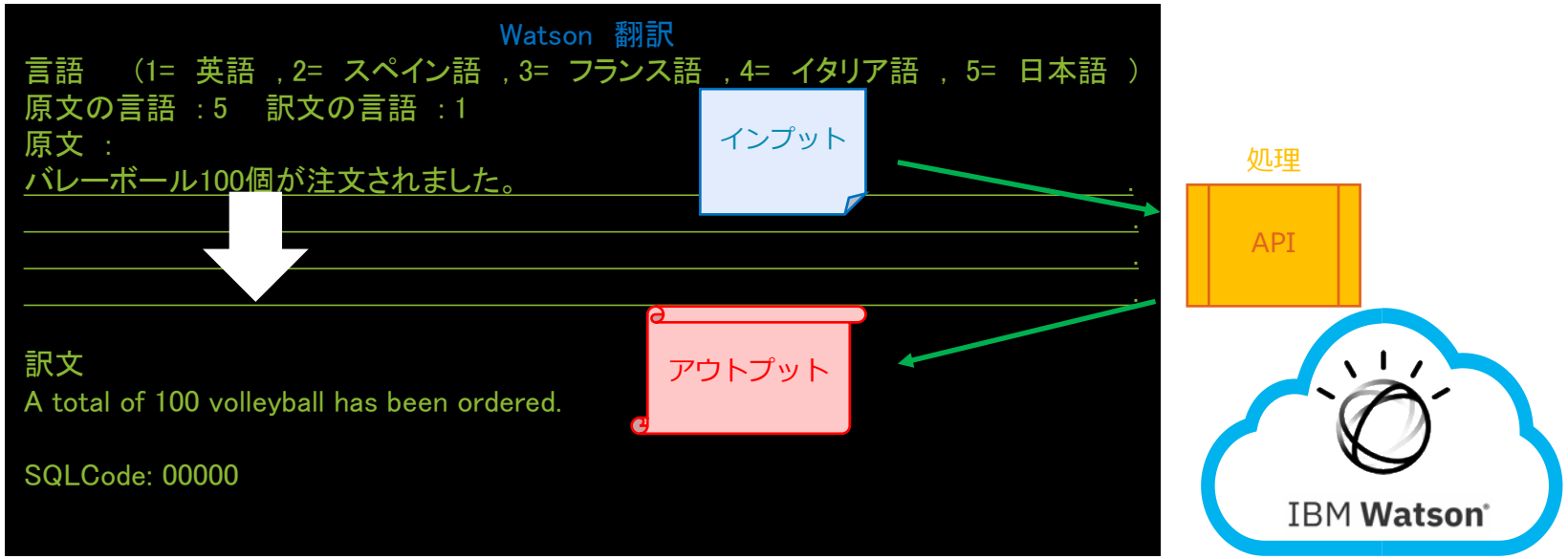
日本語環境の場合、下記前提PTFを適用

R720 SI64111
R730 SI64112
APAR SE66747 - OSP-DB UPDATES FOR SYSTOOLS NOT HAPPENING IN CCSID 290
https://www-304.ibm.com/support/docview.wss?uid=nas2SE66747

注：PTF適用後、SYSTOOLSのオブジェクトが追加されていなかった場合、"CALL QSYS/QSQSYSIBM"を実行

Watson Language Translatorを使ったILE RPGアプリケーションの例

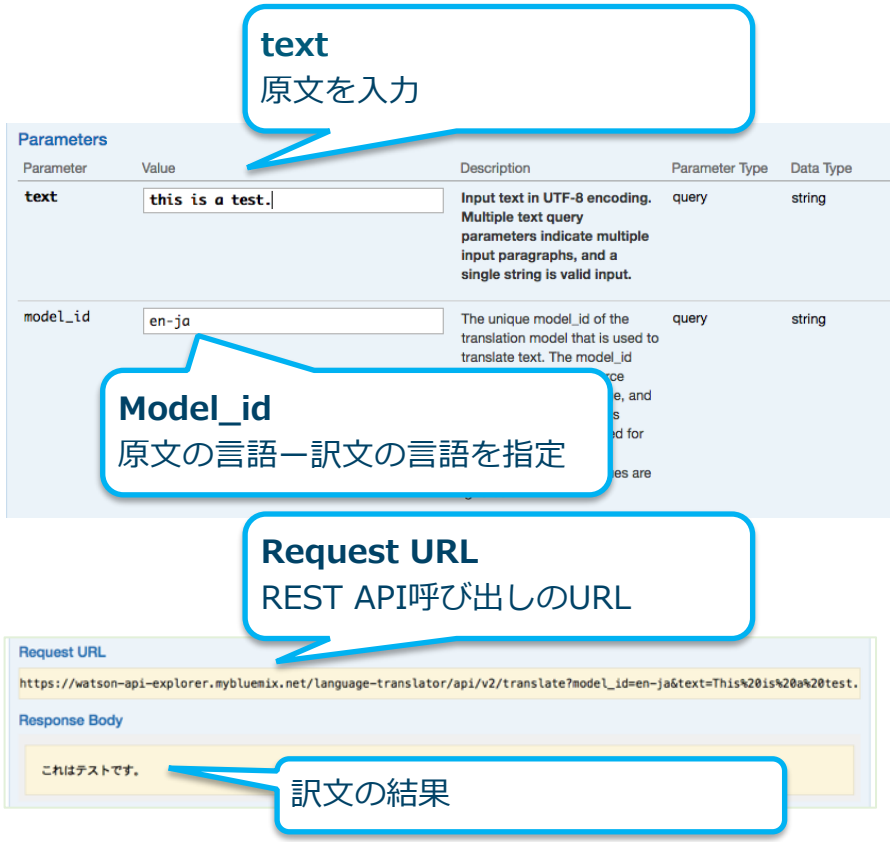
このILE RPGアプリケーションでは、Watson APIの「Language Translator」（翻訳機能）を使用
5250画面で入力した原文を「Language Translator」で翻訳し、5250画面に結果を出力



Language Translator API

Language Translator APIのURL構成
必須パラメーター
text　　: 例）This is a test.
model_id : 例) en-ja
Request URL
https://watson-api-explorer.mybluemix.net/language-translator/api/v2/translate?model_id=en-ja&text=This%20is%20a%20test.

上記URLの青太字部分をILE RPGアプリケーション内で3つのパラメータとして、REST呼び出しをします。



Watson API を使った RPG サンプル・アプリケーション - 表示装置ファイル

表示装置ファイル「TOWATSOND.DSPF」のサンプル・ソース

入力フィールド
FROMLANG：原文の言語
TOLANG：訳文の言語
FROMTEXT：原文
出力フィールド
TOTEXT：訳文
SQLCODEO ：エラー時のSQLコード
F3キー：プログラム終了

```
A                    INDARA
A                    DSPSIZ(24 80 *DS3)
A                    CF03(03)
A      R DATAR
A                  1 27'Watson翻訳'
A                    COLOR(BLU)
A                  3  1'言語 (1=英語, 2=スペイン語,'
A                  3 35'3=フランス語, 4=イタリア語,'
A                  3 67' 5=日本語)'
A                  4  1'原文の言語:'
A      FROMLANG      1  0B  4 15VALUES(1 2 3 4 5)
A                    EDTCDE(X)
A                  4 20'訳文の言語:'
A      TOLANG        1  0B  4 34VALUES(1 2 3 4 5)
A                    EDTCDE(X)
A                  6  1'原文:'
A      FROMTEXT    320O  B  8  1CHECK(LC)
A                 13  1'訳文'
A      TOTEXT      320A  O 14  1
A                 19  1'SQLCode:'
A      SQLCODEO      5  0O 19 10
A                 24  2'F3=Exit'
A                    COLOR(BLU)
```

Watson API を使った RPG サンプル・アプリケーション - メイン・プロシージャ -

ILERPG「TOWATSON.SQLRPGLE」のサンプル・ソース (1/3)

A) データ構造は言語コードの配列を定義（コードは、画面上の原文/訳文言語（1 =英語(en)、2 =スペイン語(es)など）に入力された番号に対応）

B) F3が押されるまで画面を表示しながらループ

C) transLate_Text()サブプロシージャを呼び出し、原文言語コード、訳文言語コード、原文、および、訳文のパラメータを渡す

```
**free
ctl-opt option(*srcStmt: *noDebugIO) dftactGrp(*no);
dcl-F toWatsonD workstn(*ext) usage(*input: *output) indDs(WSI);
dcl-Ds WSI qualified;
 F3Exit ind pos(3);
end-Ds;

A
dcl-Ds *n;
*n char(10) inz('enesfritja');
lang char(2) dim(5) pos(1);
end-Ds;
exec SQL
 set option naming = *SQL;
B
exfmt dataR;
dow not WSI.F3Exit;
C
transLate_Text(lang(fromLang) :
               lang(toLang) :
               fromText :
               toText);
SQLCodeO = SQLCODE;
exfmt dataR;
endDo;
*inLR = *on;
```

Watson API を使った RPG サンプル・アプリケーション - transLate_Text()サブプロシージャ -

ILERPG「TOWATSON.SQLRPGLE」のサンプル・ソース (2/3)

A) RPGはCLOBデータ型を認識しないため、textBackをSQLTypeのCLOB変数として定義
- この定義はコンパイル後2つのサブフィールド textBack-Len（データの長さ）とtextBack_Data（データ）を持つデータ構造になる
- ※「systools.httpgetclob」はデータタイプ「CLOB」で返す

B) 「systools.urlencode」を使用して、入力されたテキストをエンコード
- 問題の原因となる特殊文字（＆や<など）を同等のものに変換

```
dcl-Proc transLate_Text;
  dcl-Pi *n;
  fromLang char(2) const;
  toLang char(2) const;
  fromText char(320) const;
  toText char(320);
  end-Pi;
dcl-s str1 varchar(1000);
A dcl-s str2 varchar(1000);
dcl-s textBack SQLType(CLOB: 320);
if (fromLang =toLang);
  toText = fromText;
  return;
endIf;

B str1 =%trimR(fromText);
exec SQL
  values trim(systools.urlencode(:str1, '')) into :str2;
```

Watson API を使った RPG サンプル・アプリケーション - transLate_Text()サブプロシージャ 続き-

ILE RPG「TOWATSON.SQLRPGLE」のサンプル・ソース (3/3)

C) Watson APIをREST呼び出しし、翻訳を実行するURLを生成

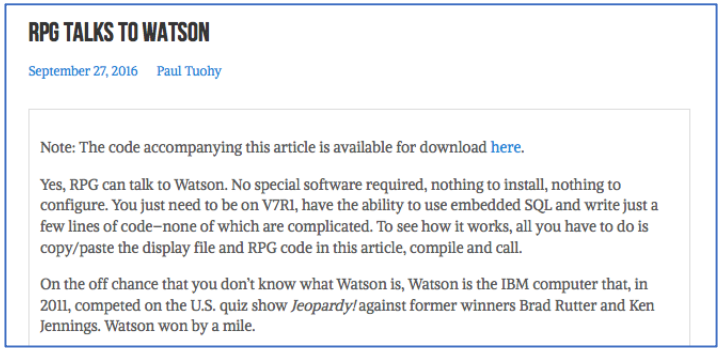
D) 「systools.httpgetclob」を使用して、Watson APIをREST呼び出し。

- 戻り値は、先に定義したtextBackに格納される

E) 結果が返された場合は、データの長さ(textBack_Len)分のデータ(textBack_Data)を戻り値に代入

C
```
str1 ='https://watson-api-explorer.mybluemix.net/' +
        'language-translator/api/v2/translate?model_id=' +
        fromLang + '-' + toLang  + '&text='+ str2;
exec SQL
```
D
```
values char(systools.httpgetclob(:str1, ''), 256)
into :textBack;
```
E
```
toText = *blanks;
if (textBack_Len >0);
toText =%subSt(textBack_Data: 1: textBack_Len);
endIf;
return;
end-Proc;
```

当プログラム例では、下記リンク先の内容、サンプル・アプリケーションを使用しています
Paul Tuohy 著 「RPG TALKS TO WATSON」Copyright © 2017 IT Jungle
https://www.itjungle.com/2016/09/27/fhg092716-story01/



RPG TALKS TO WATSON

September 27, 2016 Paul Tuohy

Note: The code accompanying this article is available for download here.

Yes, RPG can talk to Watson. No special software required, nothing to install, nothing to configure. You just need to be on V7R1, have the ability to use embedded SQL and write just a few lines of code—none of which are complicated. To see how it works, all you have to do is copy/paste the display file and RPG code in this article, compile and call.

On the off chance that you don't know what Watson is, Watson is the IBM computer that, in 2011, competed on the U.S. quiz show *Jeopardy!* against former winners Brad Rutter and Ken Jennings. Watson won by a mile.

REST利用APIのその他の活用例
– The Weather Company

気象データと基幹業務データを連携し、天候が影響を与えるビジネスに活用

StoreID	Name	City	Country	Lat	Lon
1	Store_1	Montpellier	France	43.614934	3.908162
2	Store_2	Rochester	USA	44.061604	-92.504532
3	Store_3	Osaka	Japan	34.4111	135.3112

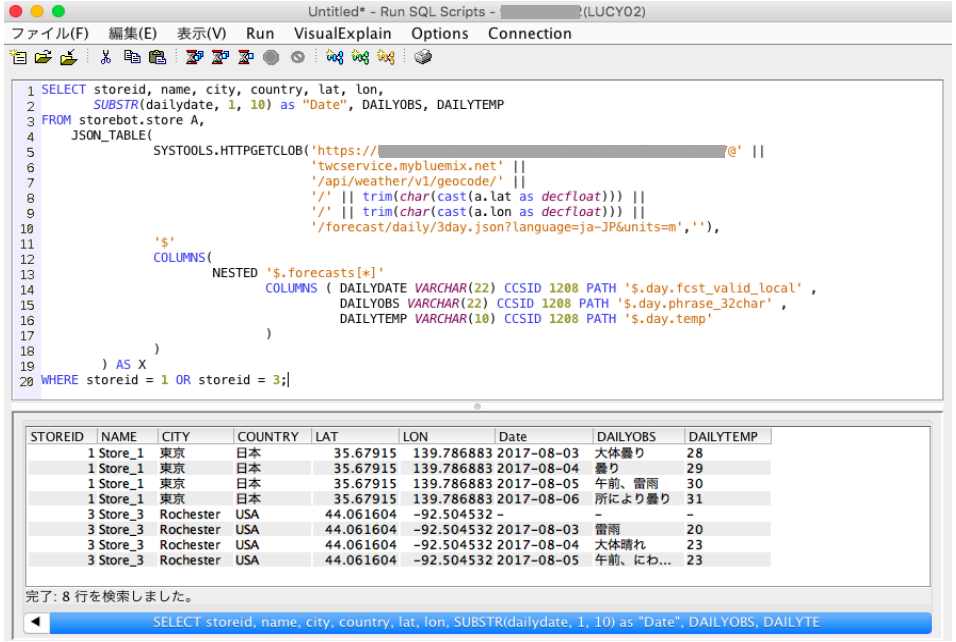
JSON_TABLE

StoreID	Name	City	Country	Lat	Lon	Observation	Feel like
1	Store_1	Montpellier	France	43.614934	3.908162	**快晴**	**11**
2	Store_2	Rochester	USA	44.061604	-92.504532	**所により曇り**	**17**
3	Store_3	Osaka	Japan	34.4111	135.3112	**小雨**	**21**

【サンプルSQL】Db2 for i テーブル上の店舗ごとの現在の日と今後3日間の天気予報の取得

既存テーブルと結合しデータの活用が可能

ILEプログラムで組み込みSQLを使用しREST APIをコール



詳細はこちら：IBM i 基幹業務データと気象データの連携_201708.pdf

https://ibm.ent.box.com/s/cv2i7opvvzr2bu2zb3n4t1ei0sidf19p

Weather Company Data APIsを使用したサンプルSQL 実行時の前提条件

- IBM i からインターネットへの接続経路があること
  - HTTPsのポートが使用可能

JSON_TABLEを使用する場合

IBM i のライセンス・プログラム、グループPTFレベルが以下を満たすこと

IBM i 7.2 Db2 PTF Group PTF SI99702 level 14以降

IBM i 7.3 Db2 PTF Group PTF SI99703 level 3 以降

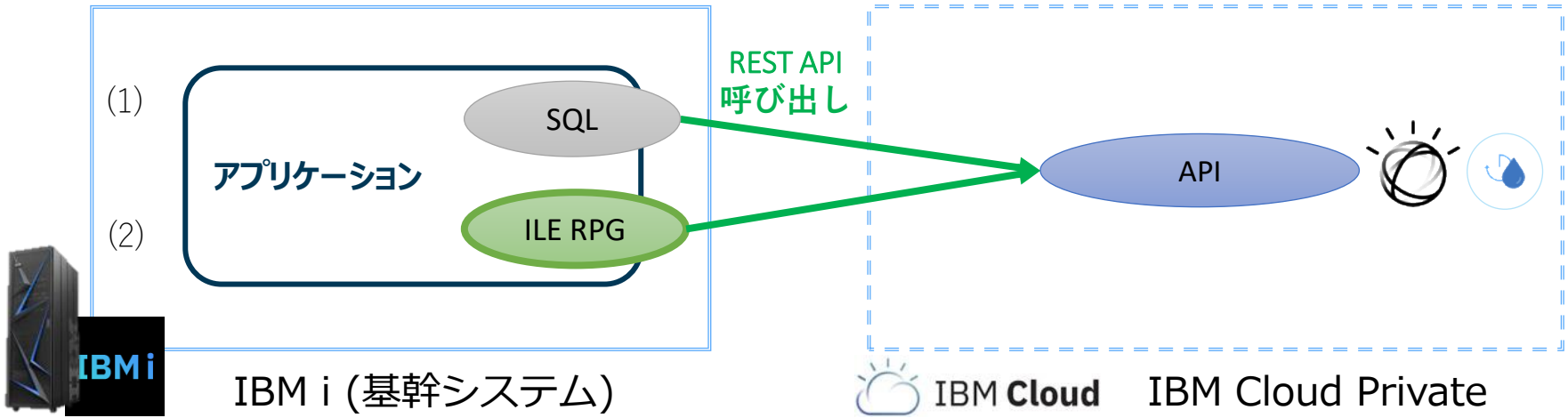
参考資料

https://developer.ibm.com/articles/i-json-table-trs/

2. [IBM i → 外部] IBM i アプリケーション側から REST APIサービスを呼び出す

IBM i アプリケーション側からREST APIサービスを呼び出す2つの方法

1. Db2 for i HTTPスカラー関数の活用
2. ILE RPGでのTransport APIの活用



ILE RPGでの統合Webサービス・クライアントAPIの活用

統合Webサービス・クライアント
Apache Axisを利用したSOAP webサービスをILEプログラムから呼び出すための機能として以前よりIBM i で提供しており、新たにREST webサービスを呼び出す機能を追加

Transport API
追加されたREST webサービスの呼び出し機能

Transport API	用途
axiscTransportCreate()	トランスポート・オブジェクトの生成
axiscTransportDestroy()	トランスポート・オブジェクトの破棄
axiscTransportReset()	トランスポート・オブジェクトを初期状態にリセット
axiscTransportSetProperty()	トランスポート属性への値のセット
axiscTransportGetProperty()	トランスポート属性値の取得
axiscTransportSend()	トランスポートを用いたデータの送信 (バッファー).
axiscTransportFlush()	バッファー・データのトランスポートを介した送出.
axiscTransportReceive()	トランスポートからの受信.
axiscTransportGetLastErrorCode()	直近で失敗したトランスポート処理のトランスポート・エラー・コードの取得
axiscTransportGetLastError()	直近で失敗したトランスポート処理のトランスポート・エラー・メッセージの取得

Transport APIの基本的な利用手順

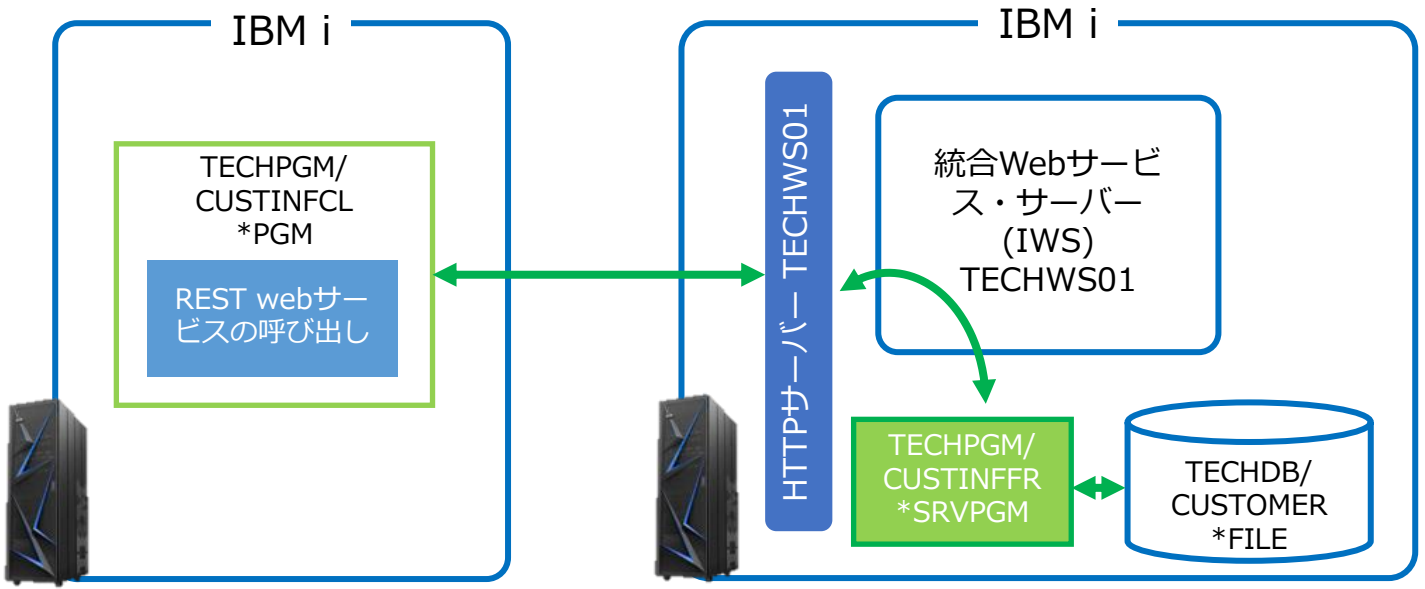
Transport APIを利用したILE RPGからのREST webサービスの呼び出しの基本的な手順

1. •axiscTransportCreate() を用いてトランスポート・オブジェクトを作成
2. •axiscTransportSetProperty() を用いてHTTPメソッドやHTTPヘッダーといった属性情報をトランスポート・オブジェクトに設定
3. •axiscTransportSend() を用いて送信するデータをバッファーに入れる（POSTやPUTで送信データがある場合）
4. •axiscTransportFlush() を用いてHTTPリクエストを送信
5. •axiscTransportReceive() を用いてHTTPレスポンスを受信
6. •axiscTransportDestroy() を用いてトランスポート・オブジェクトを削除

実体はライブラリーQSYSDIRにあるサービス・プログラムQAXIS10CC
上記APIのプロトタイプ定義が/QIBM/ProdData/OS/WebServices/V1/client/include/Axis.rpgleincとして提供されており、ILE RPGのソースに/COPYで取り込めば簡単に利用できる

【サンプル】Transport APIの利用手順

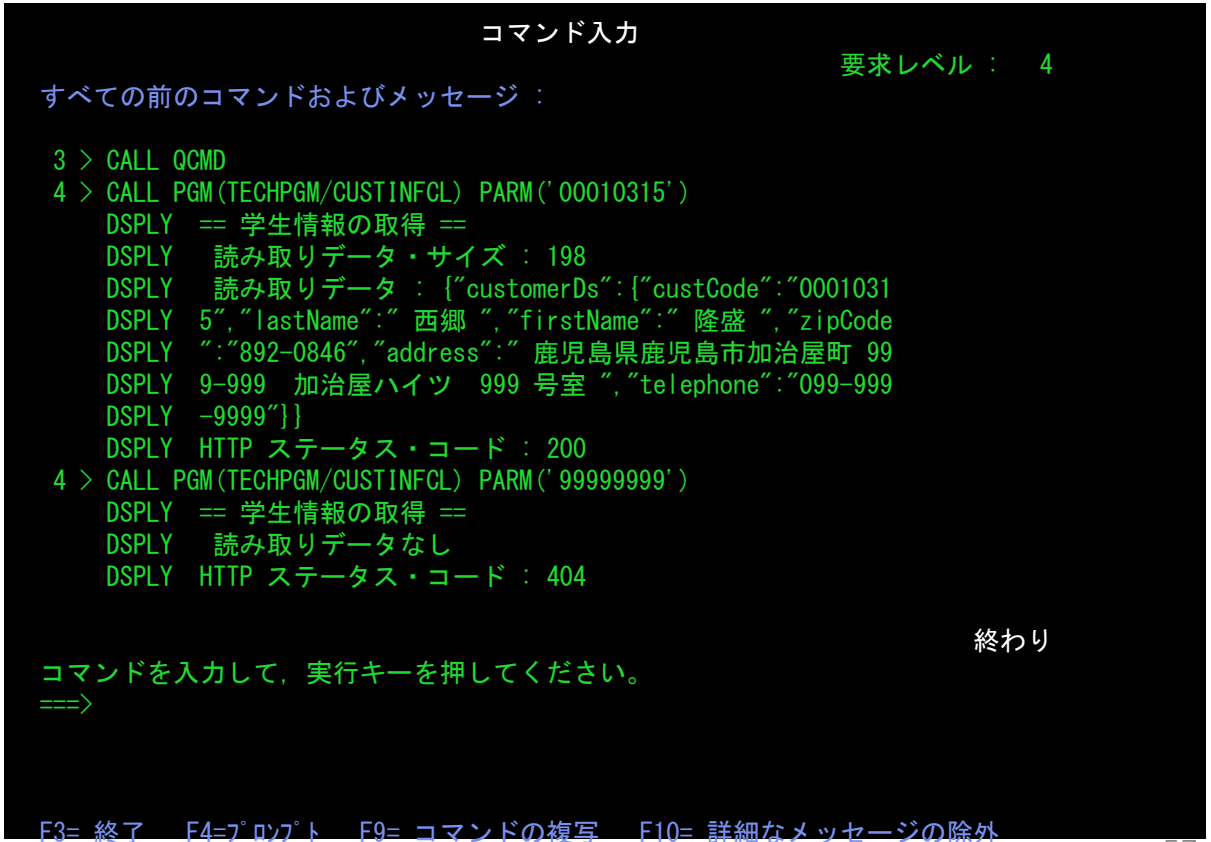
学生情報を処理するサービス・プログラムCUSTINFFRのREST webサービスを他のIBM i 上のILE RPGプログラムTECHPGM/CUSTINFCLから呼び出し



サンプル・プログラムの実行

TECHPGM/CUSTINFCLを実行時の5250画面での表示例

REST webサービス経由でJSON形式のデータを取得
データがない場合はHTTPステータス・コード404を取得



サービス・プログラムQSYSDIR/QAXIS10CCのバインド

Transport APIを使用するために、提供されているプロトタイプ定義を取り込む
下記2つのどちらかの方法を選択

1. CRTRPGMODコマンドでモジュールを作成した後に、CRTPGMコマンドでQAXIS10CCをバインド
2. バインド・ディレクトリーTECHPGM/CUSTINFCLを作成、QSYSDIR/QAXIS10CCを登録し、ソースの制御仕様書で作成したバインド・ディレクトリーを指定
   CRTBNDRPGコマンドで、QSYSDIR/QAXIS10CCをバインドした状態でプログラムを作成

以降、
2の方法を
ご紹介

```
                                コマンド入力                    KIWI
                                                  要求レベル :   4
前のコマンドおよびメッセージ :
   > CRTBNDDIR BNDDIR(TECHPGM/CUSTINFCL)
     バインド・ディレクトリー CUSTINFCL がライブラリー TECHPGM に作成された
     。
   > ADDBNDDIRE BNDDIR(TECHPGM/CUSTINFCL) OBJ((QSYSDIR/QAXIS10CC))
     ライブラリー TECHPGM のバインド・ディレクトリー CUSTINFCL に,  1 項目が
       追加され,  0 項目が追加されなかった。
```

＜ソース＞制御仕様書でバインド・ディレクトリーの指定

mainプロシージャーをメイン・プロシージャーとして定義
制御仕様書で作成したバインド・ディレクトリーを指定

```
**free
// 全体制御 ============================================================
  ctl-opt dftactgrp(*no);
  ctl-opt main(main);
  ctl-opt pgminfo(*pcml:*dclcase);
  ctl-opt bnddir('TECHPGM/CUSTINFCL');

  // システム提供のプロトタイプ定義をコピー
  /COPY /QIBM/ProdData/OS/WebServices/V1/client/include/Axis.rpgleinc

  // グローバル変数
  dcl-s rc int(10) inz(0);
  dcl-s tranHandle pointer;

  // ------------------------------------------------------------------
  // メイン・プロシージャー
  // 学生情報のREST webサービスを呼び出して取得
  // ------------------------------------------------------------------
```

＜ソース＞mainプロシージャー (1/3)

変数 uri にREST webサービスのURIを指定
サンプル・プログラムCUSTINFCLは学生コードを入力パラメーターとし、URIに入力された値をつなげる

QAXIS10CCはCPPLE (ILE C++) で作られているため、変数値は忘れずにヌル終端 (x'00'で終わる) にする.

指定したURIを入力パラメーターとし、API “axiscTransportCreate”でトランスポート・ハンドルを生成

```
dcl-proc main;
   dcl-pi *n;
     p_custCode char(8) const;
   end-pi;

   dcl-s uri char(200);
   dcl-s response char(32768);
   dcl-s propBuf char(100);
   dcl-s header pointer;
   dcl-s bytesRead int(10) inz(0);

   // トレース取得時はコメント解除
   // axiscAxisStartTrace('/tmp/axistransport.log': *NULL);

   // グローバル変数の初期化
   clear rc;
   clear tranHandle;

   // 学生情報取得のURIを指定
   uri = 'http://systemname:20080/web/services/customers/' + p_custCode + x'00';

   // HTTPトランスポート・ハンドルの生成
   tranHandle = axiscTransportCreate(uri:AXISC_PROTOCOL_HTTP11);
   if (tranHandle = *NULL);
     displayMessage('TransportCreate() failed');
     return;
   endif;
```

＜ソース＞mainプロシージャー (2/3)

属性値としてHTTPメソッドをAPI “axiscTransportSetProperty”を使用して指定
今回は指定した学生コードの情報照会のため、HTTPメソッドとしてはGETを指定

GETでの情報取得で、REST webサービス側にデータを送信しないため、今回はAPI “axiscTransportSend”は使用しない

属性の指定後 API “axiscTransportFlush”を用いてHTTPリクエストを送信
リクエストの送信後は API “axiscTransportReceive”を使用してレスポンスを受け取る
受け取るデータが存在する限りループして受け取る

```
// HTTPメソッドを定義し、トランスポートの属性に指定
propBuf = 'GET' + X'00';
axiscTransportSetProperty(tranHandle: AXISC_PROPERTY_HTTP_METHOD: %addr(propBuf));

displayMessage('==学生情報の取得==');

// HTTPリクエストを送信
rc = axiscTransportFlush(tranHandle);
if (rc = -1);
  checkError('TransportFlush()');
  return;
endif;

// HTTPレスポンスからデータを受け取り、画面に表示
rc = axiscTransportReceive(tranHandle:%addr(response):%size(response):0);
if (rc = 0);
  displayMessage('読み取りデータなし');
else;
  dow rc > 0 AND bytesRead < %size(response);
    bytesRead = bytesRead + rc;
    rc = axiscTransportReceive(tranHandle:
                               %addr(response)+bytesRead:
                               %size(response)-bytesRead:
                               0);
  enddo;
endif;
```

＜ソース＞mainプロシージャー (3/3)

今回の例では、受け取ったデータの画面表示はdisplayMessageプロシージャーを別途作成、利用している
データ表示が完了すると、API “axiscTransportDestroy”を用いて不要になったトランスポート・ハンドルを破棄し、処理を終了

```
    if (rc = -1);
      checkError ('TransportReceive()');
    elseif (bytesRead  > 0);
      displayMessage('読み取りデータ・サイズ: ' + %char(bytesRead));
      displayMessage('読み取りデータ: ' + response);
    endif;

    if (rc > -1);
      rc = axiscTransportGetProperty(tranHandle
                                     :AXISC_PROPERTY_HTTP_STATUS_CODE
                                     :%addr(header));
      if (rc = -1);
        checkError ('TransportGetProperty()');
      else;
        displayMessage('HTTPステータス・コード: ' + %str(header));
      endif;
    endif;

    // トランスポート・ハンドルの破棄
    axiscTransportDestroy(tranHandle);

    return;
  end-proc;
```

＜ソース＞メッセージ表示プロシージャー

displayMessageプロシージャーを作成し、
受け取ったデータを画面に表示

```
// ------------------------------------------------------
// メッセージ表示プロシージャー
// ------------------------------------------------------
dcl-proc displayMessage;
  dcl-pi *n;
    p_msg varchar(5000) const;
  end-pi;

  dcl-c lenPartMsg const(52);

  dcl-s i int(10) inz(0);
  dcl-s numOfLine int(10) inz(0);
  dcl-s pos int(10) inz(0);
  dcl-s len int(10) inz(0);
  dcl-s partMsg char(lenPartMsg) inz(*blank);

  numOfLine = %len(%trim(p_msg)) / lenPartMsg + 1;

  for i = 1 to numOfLine;
    pos = (lenPartMsg * (i - 1)) + 1;
    if (i = numOfLine);
      len = %len(p_msg) - lenPartMsg * (i - 1);
    else;
      len = lenPartMsg;
    endif;

    partMsg = %subst(p_msg:pos:len);
    dsply partMsg;
  endfor;
end-proc;
```

＜ソース＞エラー・チェック・プロシージャー

・Transport API使用時のエラーやHTTPステータス・コードのハンドリングを行うプロシージャーとしてcheckErrorプロシージャーを使用

・トランスポート・エラー・コード、またコードに関するメッセージを取得するAPIとしてそれぞれaxiscTransportGetLastErrorCode、axiscTransportGetLastErrorが用意されている

・HTTPステータス・コードの取得はaxiscTransportGetPropertyで属性から読み取る形で実装している

```
// ------------------------------------------------------------------
// エラー・チェック・プロシージャー
// ------------------------------------------------------------------
dcl-proc checkError;
  dcl-pi *n;
    p_msg varchar(5000) const;
  end-pi;

  dcl-s axisCode int(10);
  dcl-s statusCode pointer;
  dcl-s rc int(10);

  // Axisエラー・コードを取得
  axisCode = axiscTransportGetLastErrorCode(tranHandle);

  // 画面にエラー・コードおよびメッセージを表示
  displayMessage(p_msg + ' call failed: ' +
                 %char(axisCode) + ':' +
                 %str(axiscTransportGetLastError(tranHandle)));

  // HTTPでのエラーの場合はHTTPステータス・コードを取得、表示
  if (axisCode = EXC_TRANSPORT_HTTP_EXCEPTION);
    rc = axiscTransportGetProperty(tranHandle
                                   :AXISC_PROPERTY_HTTP_STATUS_CODE
                                   :%addr(statusCode));
    displayMessage('HTTPステータス・コード: ' + %str(statusCode));
  endif;
end-proc;
```

Transport APIの前提条件

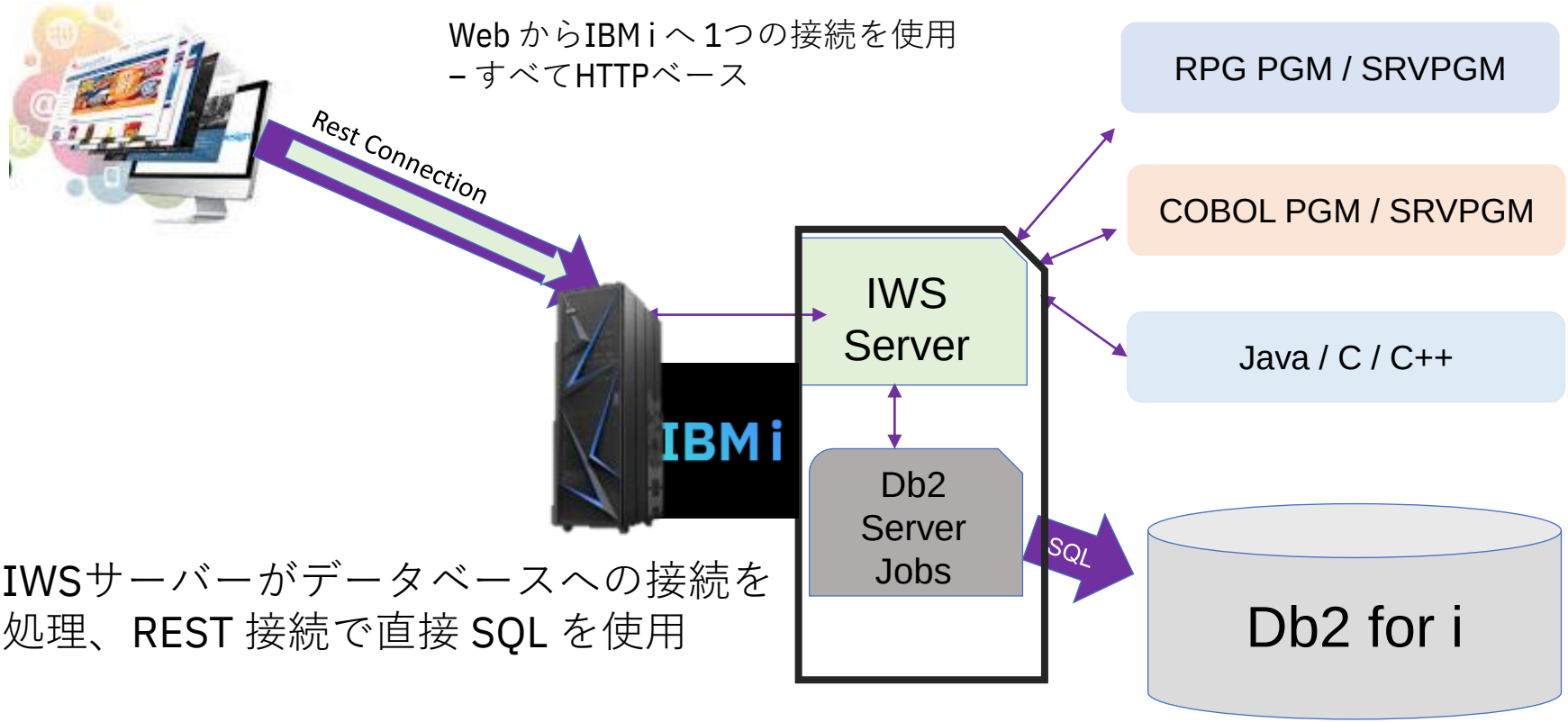
前提PTF
IBM i 7.2: SI76961, SI77246
IBM i 7.3: SI76960, SI77245

参考情報
<developerWorks: Send and receive user-defined SOAP and REST messages from RPG>
https://developer.ibm.com/articles/i-send-receive-user-defined-soap-rest-messages-trs/

2. [外部 → IBM i] DB照会をSQL記述で REST APIサービス化

（IBM i 7.4 又は、 IBM i 7.3 TR6以降の機能になります）

REST API を使用した新しい SQL アクセス

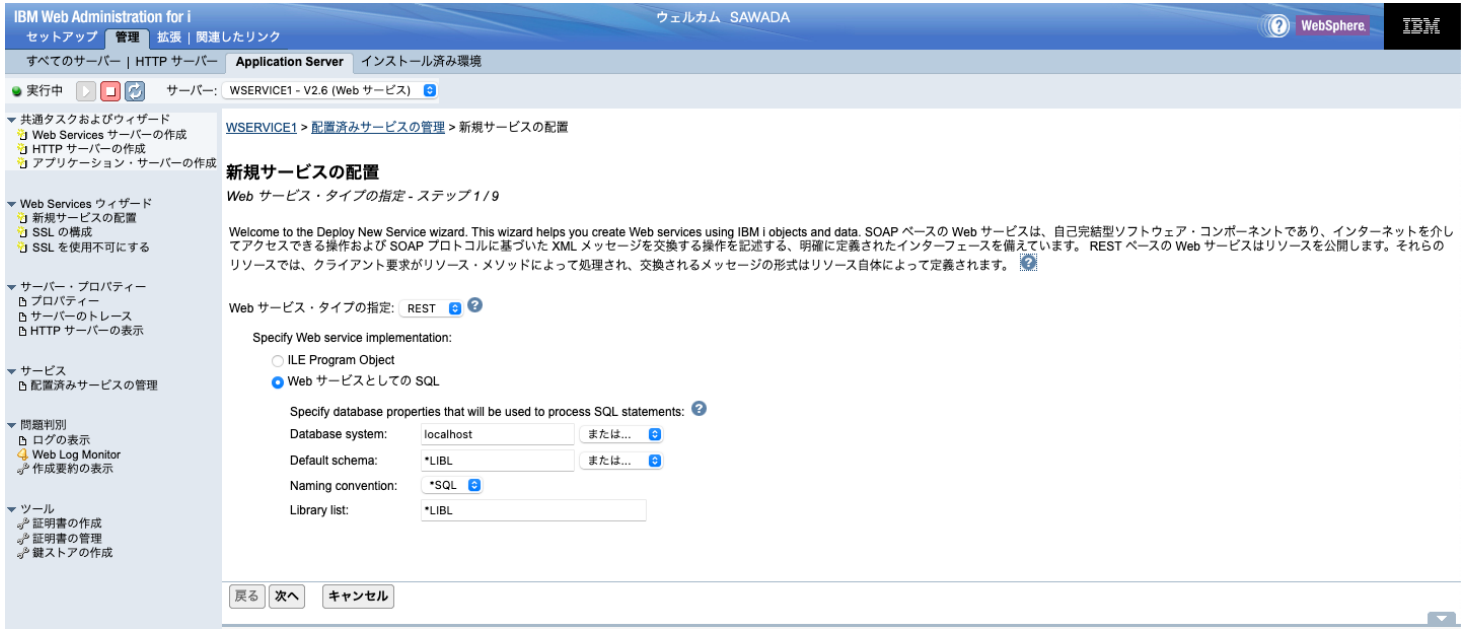


IWSサーバーがデータベースへの接続を処理、REST 接続で直接 SQL を使用

DB照会をSQL記述でRESTサービス化する手順 (1)

これは、Navigator for iのApplication Serverメニューにあ るWeb Serviceウィザードの「新規サービスの配置」から作成できる。

IBM i 7.4とIBM i 7.3 TR6ではウィザードステップのSpecify Web service implementationパラメータで「Webサービスとしての SQL」を選択することができる



DB照会をSQL記述でRESTサービス化する手順 (2)

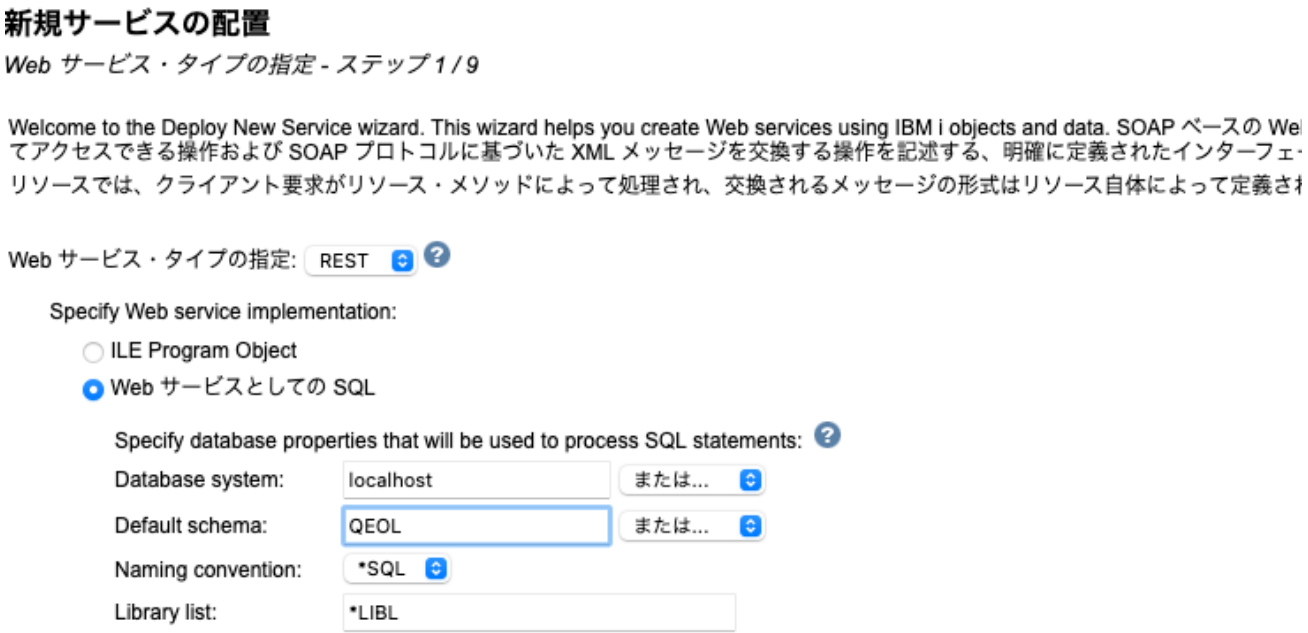
では下記に示すDb2 for iのサンプルデータ（QEOLライブラリーのTOKMSPテーブルを例に、外部からデータを参照するREST APIサービスの構成手順を追って紹介する。

メッセージ　グローバル変数および特殊レジスター　SELECT * FROM QEOL.TOKMSP

TKBANG	TKNAKN	TKNAKJ	TKADR1	TKADR2	TKTIKU	TKPOST	TKTELE	TKGURI	TKI
01010	ｱｲ ﾘﾖｶﾝ	阿井旅館	東京都渋谷区	桜ヶ丘２９	02	150	03-504-9293	698500	
01020	ｱｲ ｺｳｷﾞﾖｳ	阿井工業	東京都渋谷区	渋谷１－３	02	150	03-535-5951	452800	
01030	ｱｲｶﾜ ｺｳｷﾞﾖｳ	相川工業	東京都世田谷区	若林４－２４	06	154	03-964-6406	136200	
01040	ｱｲ ﾘﾖｺｳｼｬ	阿井旅行社	東京都品川区	東五反田１－１１	01	140	03-934-7946	3028300	1
01050	ｱｲ ｼｮｸﾋﾝK.K	阿井食品Ｋ．Ｋ	東京都荒川区	荒川５－４２	17	116	03-360-6701	541600	
01060	ｱｲ ｼﾞﾄﾞｳｼｬ	阿井自動車	東京都港区	芝公園１－２	14	105	03-860-2932	541100	
01070	ｱｲｶﾜ ｶﾒﾗ	相川カメラ	東京都新宿区	四谷１－１６	03	160	03-354-3018	367900	
01080	ｱｲｶﾜ ｺｳｸｳK.K	相川広告Ｋ．Ｋ	東京都渋谷区	広尾３－９	02	150	03-368-6366	318000	
01090	ｱｲｶﾜ ﾃﾞﾝｷK.K	相川電機Ｋ．Ｋ	東京都北区	滝の川７－１７	14	114	03-749-6271	877100	
01100	ｱｲｶﾜ ｶﾞｯｷﾃﾝ	相川楽器店	東京都港区	虎ノ門３－２１	22	105	03-922-1801	501300	
01110	ｱｲｶﾜ ｾｯｹｲｼﾞﾑｼｮ	相川設計事務所	東京都文京区	本郷１－２５	13	113	03-909-9124	446300	
01120	ｱｲｶﾜ ｼｮｳｼﾞ	相川商事	東京都葛飾区	新小岩２－２０	21	124	03-856-7896	315600	
01130	ｱｲ ｺｳｹﾞｲｼｬ	愛工芸社	東京都港区	南麻布１－１０	14	106	03-187-2308	635400	
01140	ｱｲｶﾜ ﾄｿｳﾃﾝ	相川塗装店	東京都練馬区	高野台２－１４	12	177	03-397-7993	159600	
01150	ｱｲｶﾜ ｳﾝｿｳK.K	相川運輸Ｋ．Ｋ	東京都杉並区	西荻３－２３	04	167	03-920-5933	84200	
01160	ｱｲｶﾜ ﾋﾞｮｳｲﾝ	相川病院	東京都目黒区	目黒４－２３	15	153	03-897-8588	146900	
01170	ｱｲｶﾜ ｶｲｹｲｼﾞﾑｼｮ	相川会計事務所	東京都中央区	東銀座７－１５	08	103	03-895-5141	464100	
01180	ｱｲｶﾜ ｼｮｳｶｲ	相川商会	東京都調布市	若葉２－２２	28	182	03-231-1579	674800	
01190	ｱｲｶﾜ ｹﾝｾﾂK.K	相川建設Ｋ．Ｋ	東京都江東区	嵯峨１－１１	23	135	03-751-8216	899000	
01200	ｱｲ ｲﾝｻﾂｺｳｷﾞﾖｳK.K	阿井印刷工業Ｋ．Ｋ	東京都千代田区	神田神保町１－１	09	101	03-384-3427	149100	
01210	ｱｲｼｰ ｴﾚｸﾄﾛﾆｸｽｺｳｷﾞﾖｳ	アイシーエレクトロ	東京都台東区	東上野３－２４	07	110	03-353-7744	1282500	
01220	ｱｲｶﾜ ﾃﾞﾊﾟｰﾄ	相川デパート	東京都中央区	入船２－５	08	104	03-731-1670	242600	
01230	ｱｲｶﾜ ｺｳｺｸｼｬ	相川広告社	東京都新宿区	西新宿４－３１	03	160	03-783-4171	487800	
01240	ｱｲｶﾜ ﾎｳｿｳK.K	相川包装Ｋ．Ｋ	東京都新宿区	大久保２－１８	03	160	03-893-2399	227000	
01250	ｱｲﾃﾞｱ ｻﾝｷﾞﾖｳK.K	アイデア産業Ｋ．Ｋ	東京都渋谷区	神宮前１－１１	02	150	03-318-2857	70200	
01260	ｱｲｶﾜ ｻﾝｷﾞﾖｳK.K	相川産業Ｋ．Ｋ	東京都北区	田端１－１５	22	114	03-249-6244	46400	
01270	ｱｲ ｼﾞﾄﾞｳｼｬﾌﾞﾋﾝｺｳｷﾞﾖｳ	阿井自動車部品工業	東京都目黒区	大橋２－７	15	153	03-886-4312	4220900	2
01280	ｱｲﾃﾞｱ ｷｮｳｶｲK.K	アイデア協会Ｋ．Ｋ	東京都杉並区	荻窪４－２１	04	167	03-471-8810	1799100	1
01290	ｱｲ ﾃﾞﾝｷｼｮｳｶｲ	愛電器商会	東京都杉並区	上高井戸１－５	04	168	03-190-4691	1554900	
01300	ｱｲ ﾃﾂｺｳｼｮ	阿井鉄工所	東京都豊島区	池袋３－３１	10	171	03-656-9641	479100	
02010	ｱｲｶﾜ ｾﾝﾀｰ	相川センター	東京都江東区	新木場１－４	23	136	03-512-9478	705200	

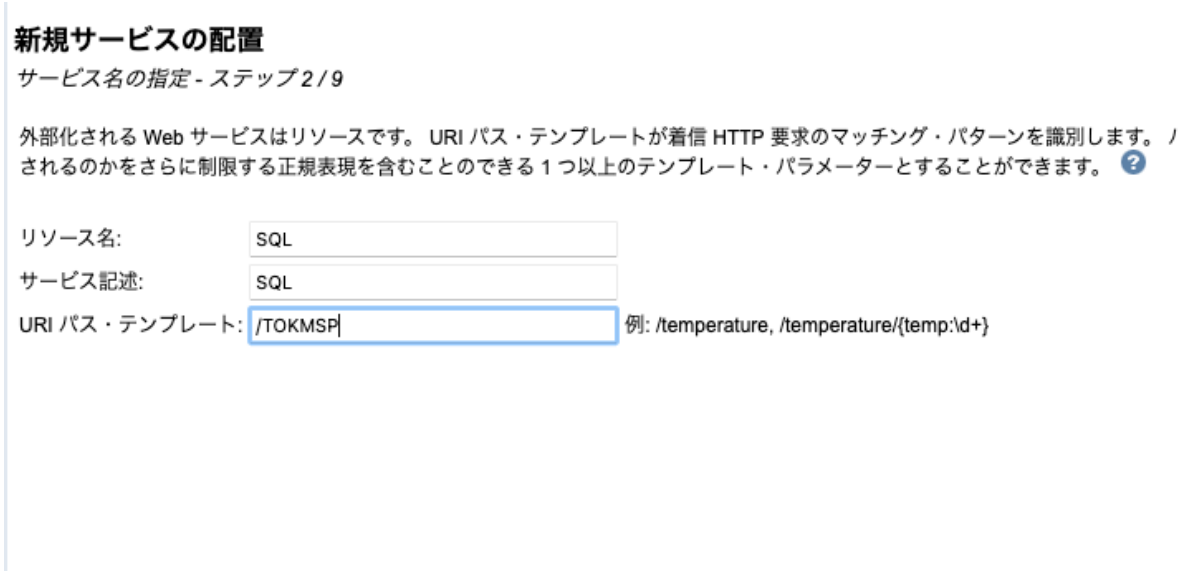
DB照会をSQL記述でRESTサービス化する手順 (3)

サービス化の対象とするテーブルのライブラリ(QEOL)を指定



DB照会をSQL記述でRESTサービス化する手順 (4)

リソース名とURIパス・テンプレートを指定する

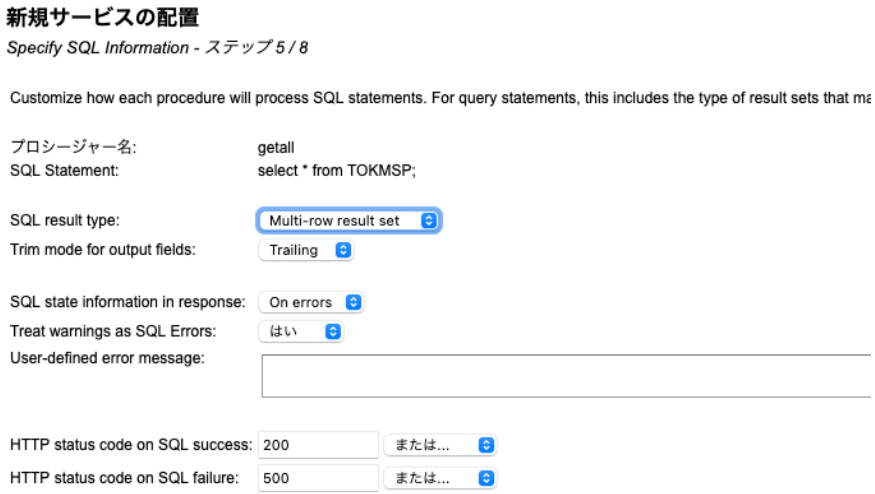


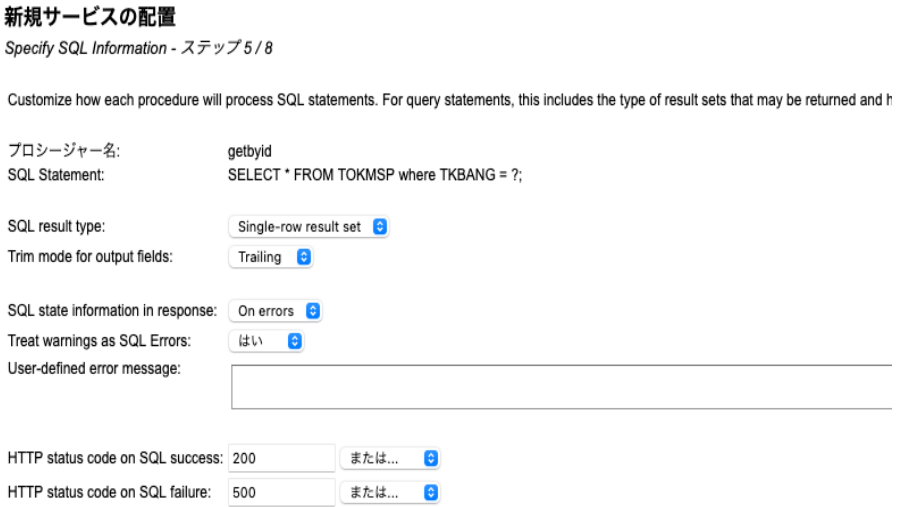
DB照会をSQL記述でRESTサービス化する手順 (5)

下記は2つのプロシージャを定義している例。getAllプロシージャは全レコードデータを返すメソッドとして定義している。getByIDプロシージャはTKBANGを指定して特定のレコードを返すメソッドとして定義している。

DB照会をSQL記述でRESTサービス化する手順 (6)

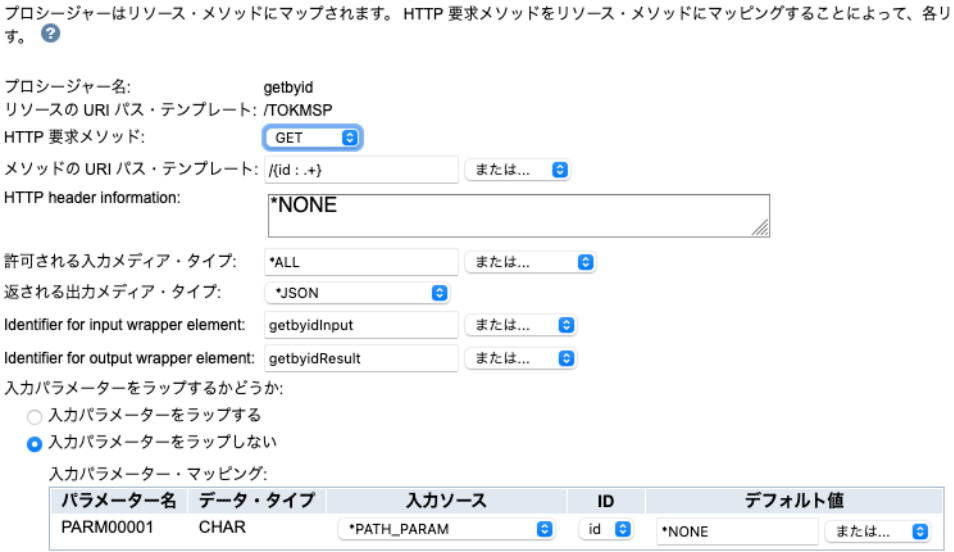
ウィザードのステップ5ではSQL処理をカスタマイズする。ここではgetByIDプロシージャのSQL result typeを「Single-row result set」に変更している。この記事の例では、getALLプロシージャを含め、それ以外のパラメータはデフォルトの設定。





DB照会をSQL記述でRESTサービス化する手順(7)

ウィザードのステップ6では各プロシージャのメソッド情報を指定する。getAllプロシージャはデフォルトのパラメータのままとし、getByIDプロシージャは下記のように、入力パラメータを設定してレコード番号を特定できるように定義する。



DB照会をSQL記述でRESTサービス化する手順(8)

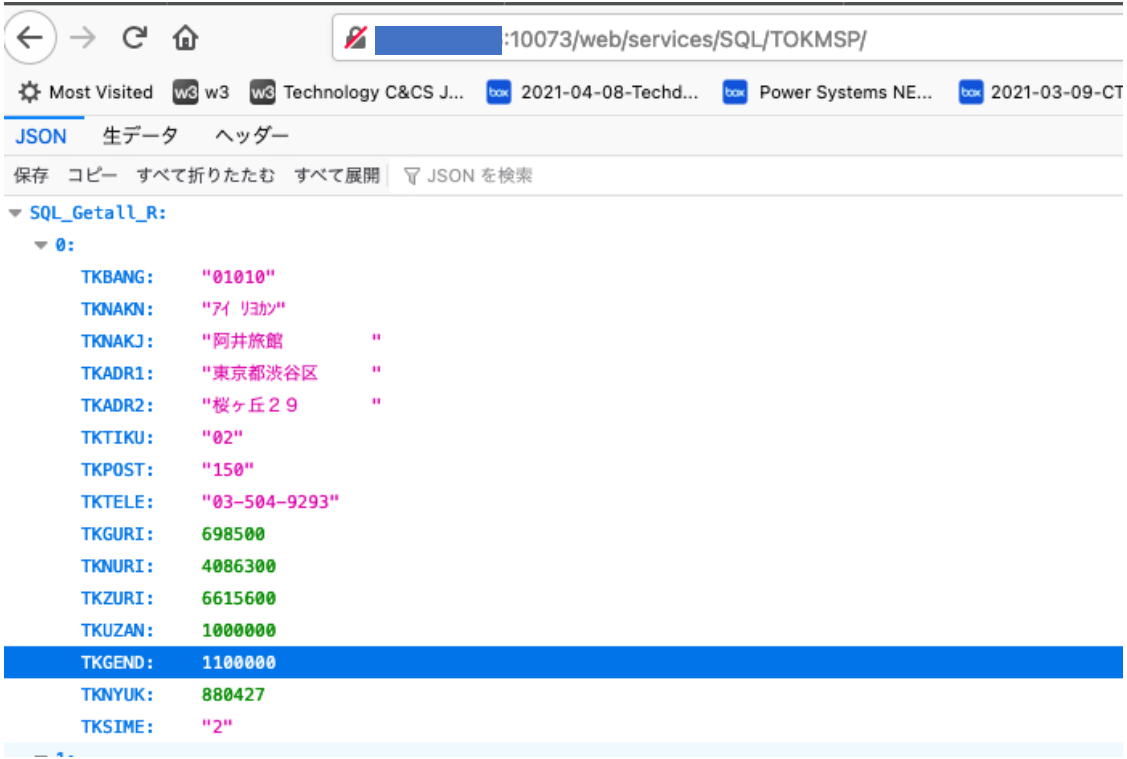
ウィザードのステップ7ではサービスの実行ユーザーIDを指定する

DB照会をSQL記述でRESTサービス化する手順 (9)

ウィザードのステップ8の要約画面で設定内容を確認して完了すると、サービスが起動する

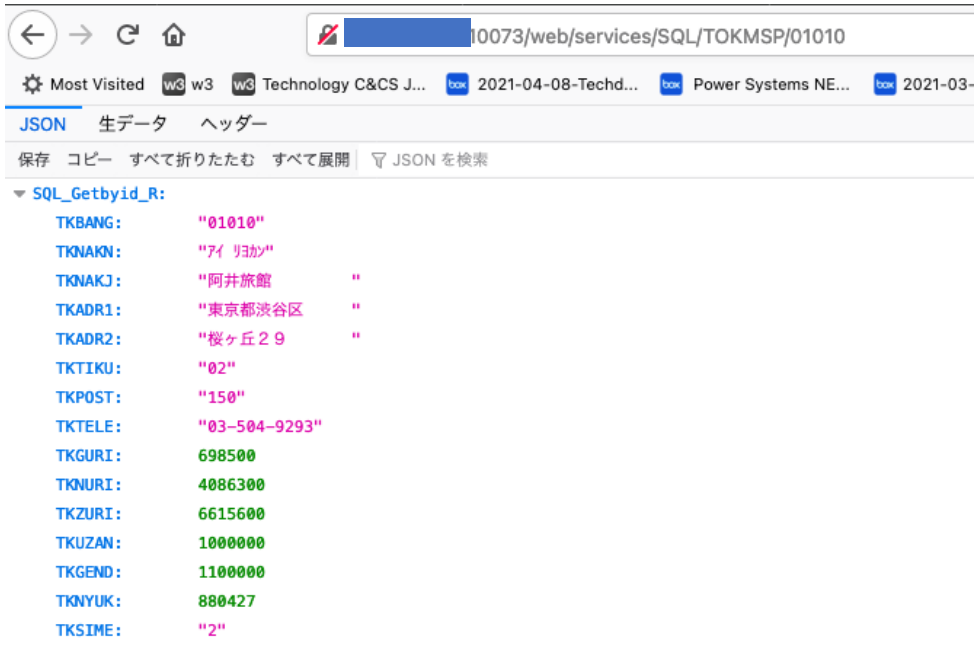
DB照会をSQL記述でRESTサービス化する手順 (10)

生成したREST APIサービスはWebブラウザでアクセスして稼働を確認できる。Db2 for iのデータをJSON形式で受け取ることができる。下記はGETALLメソッド.



DB照会をSQL記述でRESTサービス化する手順 (11)

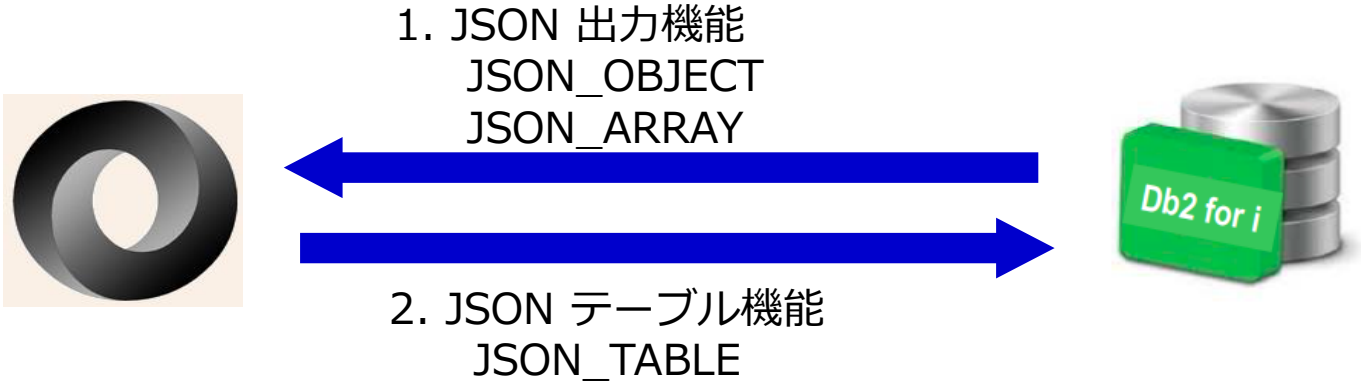
生成したREST APIサービスはWebブラウザでアクセスして稼働を確認できる。Db2 for iのデータをJSON形式で受け取ることができる。下記はGETBYIDメソッド。



3. JSONデータと Db2 for iデータの変換機能

Db2 for i と JSON オブジェクトの双方向データ交換

Webアプリケーションで広く使われる JSON オブジェクトへの対応を強化
- Db2 for i から JSON オブジェクトをSQLで生成する機能
- JSONオブジェクトからDb2 for iへのRDBデータへ挿入する機能



CUSTOMER

```
{"BANGO":"01010","CUSTNAME":"ｱｲ ﾘﾖｶﾝ
","CUSTNAMEJ":"阿井旅館　　"}
```

```
SELECT JSON_OBJECT (KEY 'BANGO' VALUE TKBANG,
                    KEY 'CUSTNAME' VALUE TRIM(TKNAKN),
                    KEY 'CUSTNAMEJ' VALUE TRIM(TKNAKJ)
                    ) AS CUSTOMER
                    FROM QEOL.TOKMSP LIMIT 1;
```

1. JSON形式への出力機能

JSON_OBJECT（JSONオブジェクトの生成機能）

```
SELECT JSON_OBJECT (KEY 'BANGO' VALUE TKBANG,
                    KEY 'CUSTNAME' VALUE TRIM(TKNAKN),
                    KEY 'CUSTNAMEJ' VALUE TRIM(TKNAKJ)
                    ) AS CUSTOMER
                    FROM QEOL.TOKMSP LIMIT 5;
```

（例）
QEOLライブラリー
のTOKMSPファイル

```
SELECT JSON_OBJECT (KEY 'BANGO' VALUE TKBANG, KEY 'CUSTNAME' VALUE TRIM(TK
CUSTOMER
{"BANGO":"01010","CUSTNAME":"ｱｲ ﾘﾖｶﾝ","CUSTNAMEJ":"阿井旅館          "}
{"BANGO":"01020","CUSTNAME":"ｱｲ ｺｳｷﾞﾖｳ","CUSTNAMEJ":"阿井工業          "}
{"BANGO":"01030","CUSTNAME":"ｱｲｶﾜ ｺｳｷﾞﾖｳ","CUSTNAMEJ":"相川工業          "}
{"BANGO":"01040","CUSTNAME":"ｱｲ ﾘﾖｺｳｼﾔ","CUSTNAMEJ":"阿井旅行社          "}
{"BANGO":"01050","CUSTNAME":"ｱｲ ｼﾖｸﾋﾞｳK.K","CUSTNAMEJ":"阿井食品Ｋ．Ｋ    "}
```

JSON_ARRAY（JSON配列の生成機能）

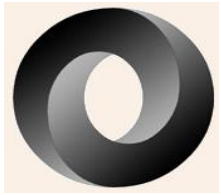
```
SELECT JSON_ARRAY (
                JSON_OBJECT(KEY 'BANGO' VALUE TKBANG),
                JSON_OBJECT(KEY 'CUSTNAME' VALUE TRIM(TKNAKN)),
                JSON_OBJECT(KEY 'CUSTNAMEJ' VALUE TRIM(TKNAKJ))
                ) AS CUSTOMER
               FROM QEOL.TOKMSP LIMIT 5;
```

（例）
QEOLライブラリー
のTOKMSPファイル

CUSTOMER
[{"BANGO":"01010"},{"CUSTNAME":"ｱｲ ﾘﾖｶﾝ"},{"CUSTNAMEJ":"阿井旅館 "}]
[{"BANGO":"01020"},{"CUSTNAME":"ｱｲ ｺｳｷﾞﾖｳ"},{"CUSTNAMEJ":"阿井工業 "}]
[{"BANGO":"01030"},{"CUSTNAME":"ｱｲｶﾜ ｺｳｷﾞﾖｳ"},{"CUSTNAMEJ":"相川工業 "}]
[{"BANGO":"01040"},{"CUSTNAME":"ｱｲ ﾘﾖｺｳｼｬ"},{"CUSTNAMEJ":"阿井旅行社 "}]
[{"BANGO":"01050"},{"CUSTNAME":"ｱｲ ｼｮｸﾋﾞｳK.K"},{"CUSTNAMEJ":"阿井食品Ｋ．Ｋ "}]

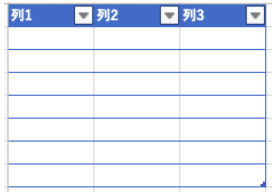
2. JSONテーブル機能

・JSONをRDB変換処理することができるSQL関数
　JSONデータをRDBの表形式に変換
　変換された表をSQLで処理可能
・SELECTステートメントのFROMで使用
　IFS上のJSONファイルを処理
　Webサービス等で直接取得して処理



JSON

2. JSON テーブル機能
JSON_TABLE



JSONテーブルの利用方法

基本構文

```
SELECT *FROM JSON_TABLE (
                        JSON形式データ,
                        'SQL JSON パス形式'
                        COLUMNS (カラム名　データタイプ　PATH カラム・パス
                                 )
                        ) AS X
```

・Db2 for i HTTP機能を使用
　SYSTOOLS.HTTPGETCLOBでJSON形式データを取得

（例）

```
SYSTOOLS.HTTPGETCLOB(
'http://express.heartrails.com/api/json?method=getStations&line=JR山手線',null)
```

（参考）JSONテーブルでの駅名取得 (1)

HeartRails Express API（駅名などをAPI でJSON形式で公開）
http://express.heartrails.com/api/json?method=getStations&line=JR山手線

（参考）JSONテーブルでの駅名取得（2）

JSON_TABLEを含むSQLを実行（ACSのSQLスクリプトで 下記を実行）で公開URLのJSONファイルを取得

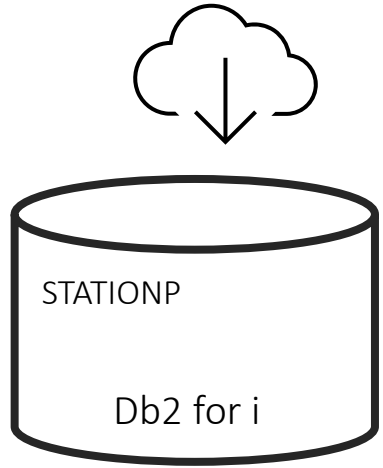
```
Select * from JSON_TABLE(
      SYSTOOLS.HTTPGETCLOB(
      ‘http://express.heartrails.com/api/json?method=getStations&line=JR山手線’,null),
      '$.response.station[*]'
         COLUMNS(   prev VARGRAPHIC(10) PATH '$.prev',
                  name VARGRAPHIC(10) PATH '$.name',
                  next VARGRAPHIC(10) PATH '$.next'
               )
               ) AS X;
```

PREV	NAME	NEXT
–	品川	大崎
品川	大崎	五反田
大崎	五反田	目黒
五反田	目黒	恵比寿
目黒	恵比寿	渋谷
恵比寿	渋谷	原宿
渋谷	原宿	代々木
原宿	代々木	新宿
代々木	新宿	新大久保
新宿	新大久保	高田馬場
新大久保	高田馬場	目白
高田馬場	目白	池袋
目白	池袋	大塚
池袋	大塚	巣鴨
大塚	巣鴨	駒込
巣鴨	駒込	田端
駒込	田端	西日暮里
田端	西日暮里	日暮里
西日暮里	日暮里	鶯谷
日暮里	鶯谷	上野
鶯谷	上野	御徒町
上野	御徒町	秋葉原
御徒町	秋葉原	神田
秋葉原	神田	東京
神田	東京	有楽町
東京	有楽町	新橋
有楽町	新橋	浜松町
新橋	浜松町	田町
浜松町	田町	高輪ゲートウェイ
田町	高輪ゲートウェイ	–

（参考）JSONテーブルでの駅名取得（3）

- API公開しているJSONデータを、IBM iのデータベースに保存）

```
**free
        ctl-opt dftactgrp(*no) main(main);
        dcl-proc main;
        exec sql insert into stationp
       select * from JSON_TABLE(
             SYSTOOLS.HTTPGETCLOB(
       'http://express.heartrails.com/api/json?method=getStations&line=JR山手線
',null),
             '$.response.station[*]'
                COLUMNS(   prev VARGRAPHIC(10) PATH '$.prev',
                        name VARGRAPHIC(10) PATH '$.name',
                        next VARGRAPHIC(10) PATH '$.next'
                    )
                    )  AS X;
       end-proc;
```



	前の駅名	駅名	次の駅名
000001	-	品川	大崎
000002	品川	大崎	五反田
000003	大崎	五反田	目黒
000004	五反田	目黒	恵比寿
000005	目黒	恵比寿	渋谷
000006	恵比寿	渋谷	原宿
000007	渋谷	原宿	代々木
000008	原宿	代々木	新宿
000009	代々木	新宿	新大久保
000010	新宿	新大久保	高田馬場
000011	新大久保	高田馬場	目白
000012	高田馬場	目白	池袋
000013	目白	池袋	大塚
000014	池袋	大塚	巣鴨
000015	大塚	巣鴨	駒込
000016	巣鴨	駒込	田端
000017	駒込	田端	西日暮里
000018	田端	西日暮里	日暮里
000019	西日暮里	日暮里	鶯谷

IBM i 統合アプリケーション・サーバーでのRESTサービス・サポートのメリット

- IBM i 同梱の標準機能なので**経済的**。
- RPG/COBOLメンバーもRESTサービス開発ができるので、**すぐ開始**。
- 既存のRPG/COBOL資産を活かしてDXができるので**生産的**。
- IBM i 内のアプリ・アーキテクチャーもマイクロサービス化してアジャイル対応できるのでDX対応インフラへの変換が**短期間**。
- Db2とJSON間のやりとりはSQLで一括変換できるのでプログラミングが**簡単**。