

# ロボットの実行パターン

## 1. スケジューラを通して定期的に行う方法

バッチ

リアルタイム

- 付属のソフトウェア（Management Console）を使用

## 2. Kappletを通してユーザ個別に行う方法

バッチ

リアルタイム

- 付属のソフトウェア（Kapplet）を使用

## 3. 他のシステムをI/Fとしてユーザードリブンで行う方法

バッチ

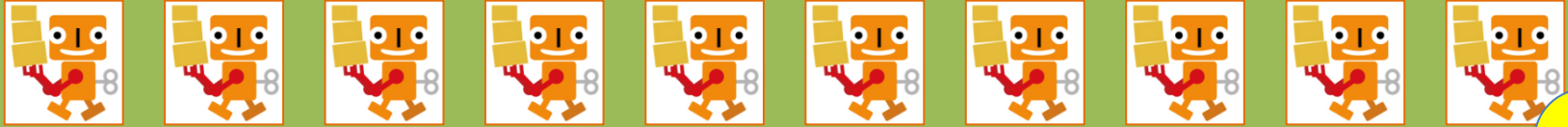
リアルタイム

- I/Fとなる画面部分をJavaもしくはC#で作成する必要あり
- Ver8.3以降はRESTコール（Web API）で行うことも可

# 1. スケジューラからロボットを実行する仕組み（全体像）

サーバサイド

BizRobo!サーバ



RoboServer

3

Web/Appサーバ

2

Management Console

Schedules

1

1. クライアントがブラウザからスケジュールを設定
2. 設定されたスケジュールに従って、Management ConsoleがRoboServerに命令（内部ではAPI経由で呼び出し）
3. RoboServerがAPIで指定されたロボットを実行

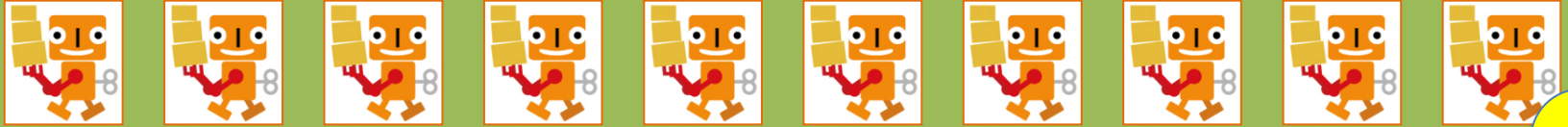
クライアントサイド



## 2. Kappletからロボットを実行する仕組み（全体像）

サーバサイド

BizRobo!サーバ



RoboServer

3

Web/Appサーバ

2

Management Console

Kapplets (KappZone)

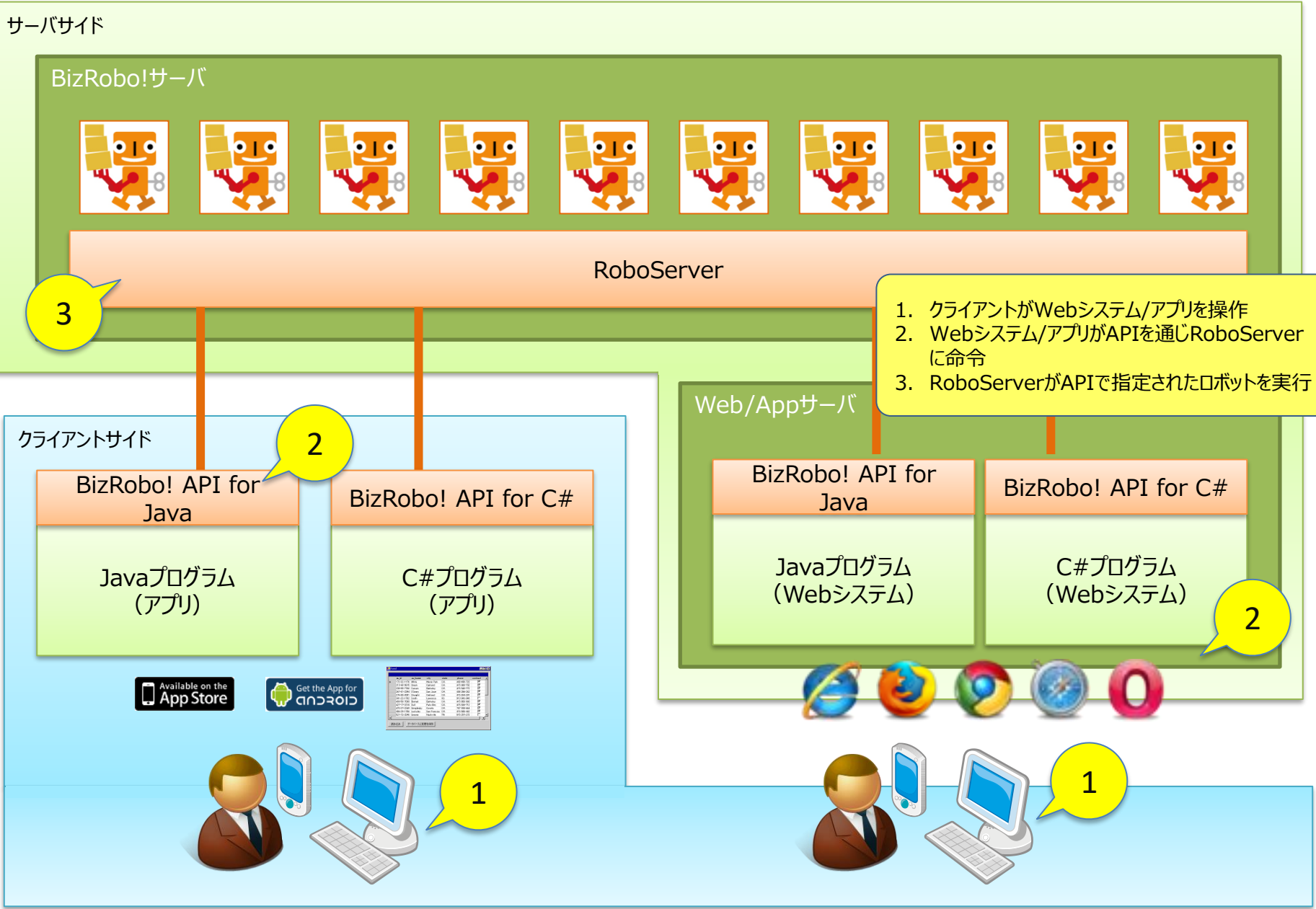
1

1. クライアントがブラウザからKappletを設定
2. Kappletにスケジュールが設定された場合には、設定されたスケジュールに従って、直接実行ボタンが押された場合には即時にRoboServerに命令（内部ではAPI経由で呼び出し）
3. RoboServerがAPIで指定されたロボットを実行

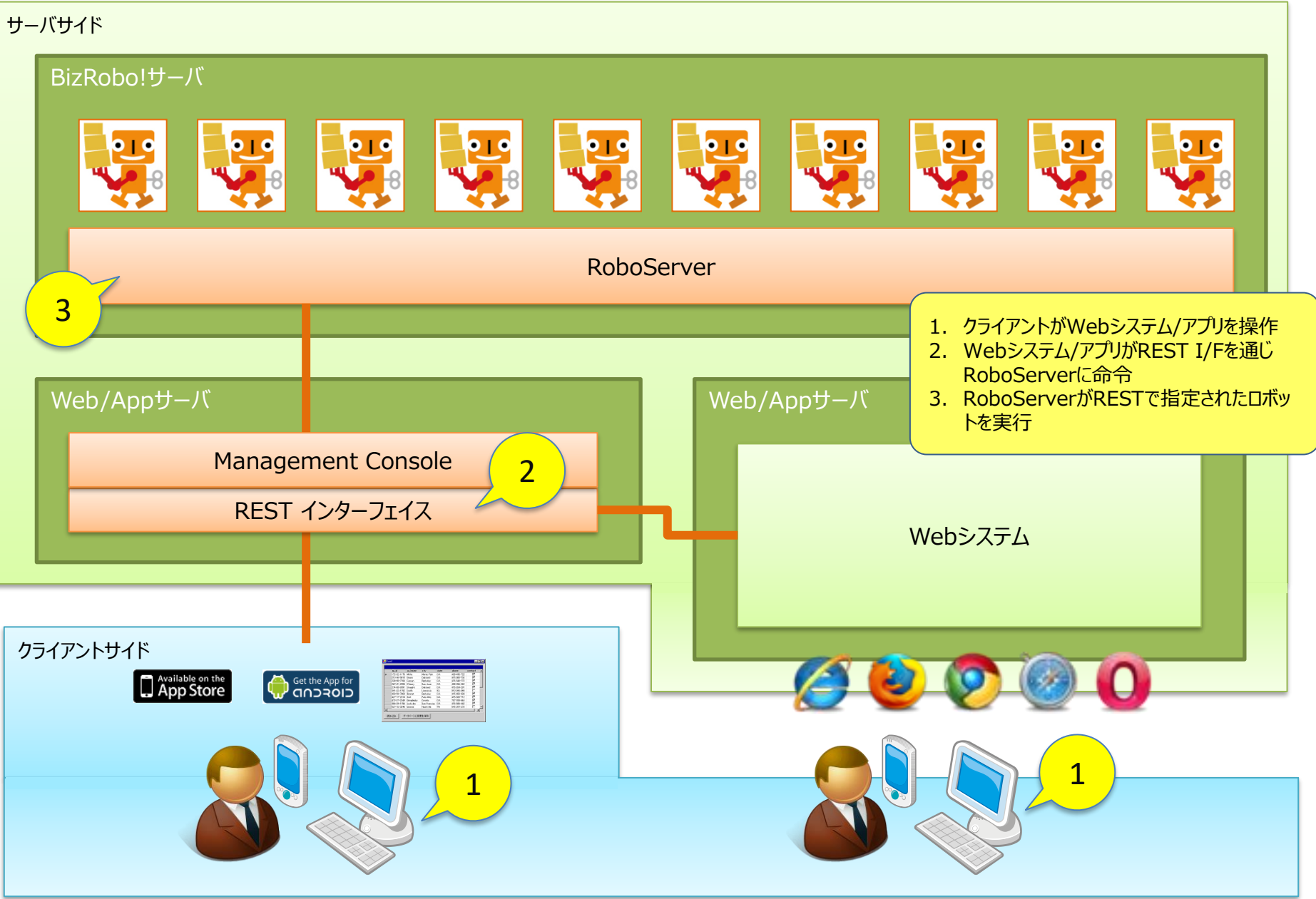
クライアントサイド



### 3. 他のシステムからロボットを実行する仕組み（全体像：API編）



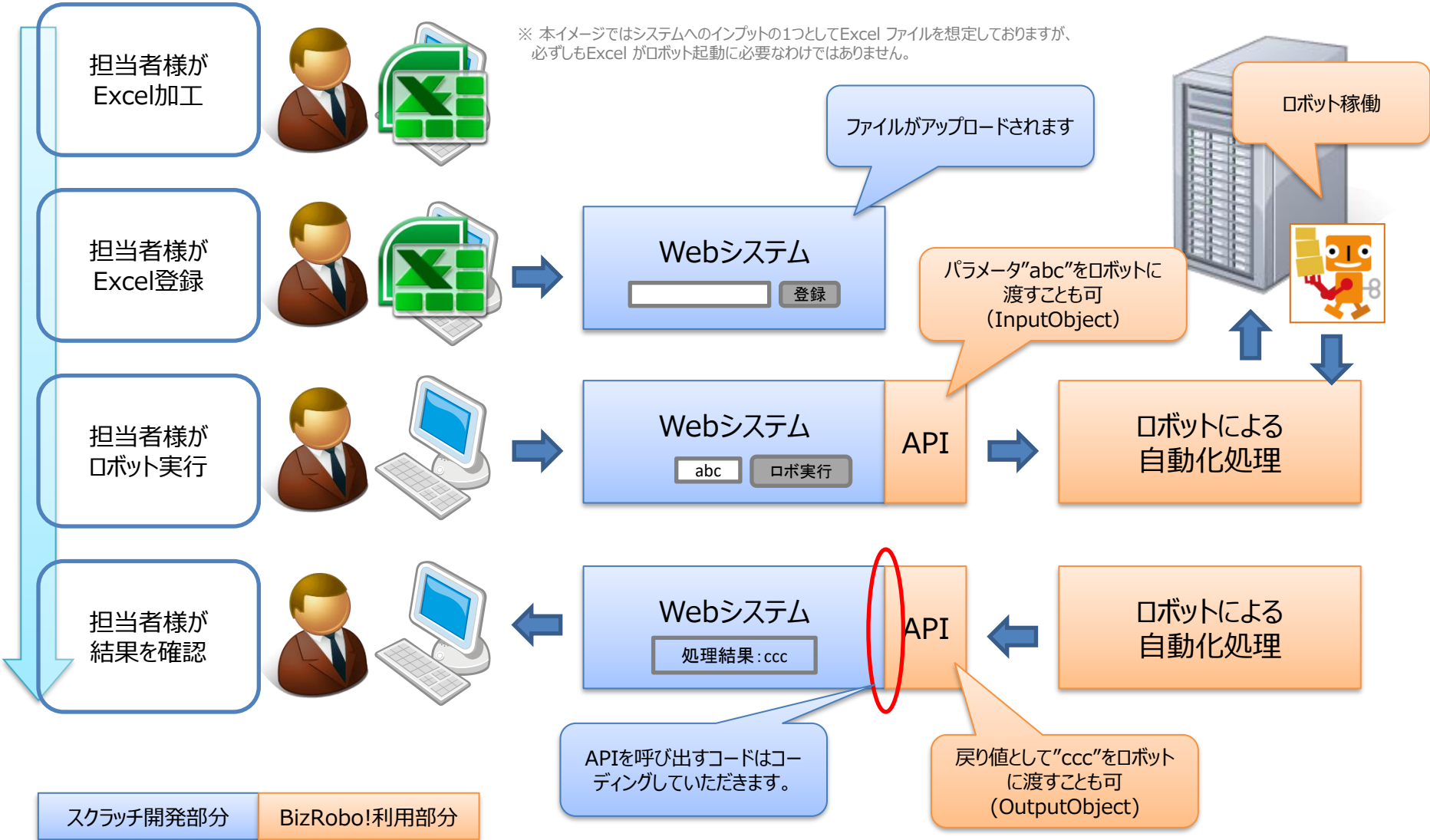
### 3. 他のシステムからロボットを実行する仕組み（全体像：REST編）



例

# API経由で他のシステムからロボットを実行する場合の具体的なロボット実行イメージ (Webシステム想定)

※ 本イメージではシステムへのインプットの1つとしてExcel ファイルを想定しておりますが、必ずしもExcel がロボット起動に必要なわけではありません。



# APIの使い方

APIにはJava用、C#用の2種類があり、どちらもAPIとして提供されているコードをプログラム中に組み込むことで、ロボットとの通信が可能になります。

使用方法については、ご利用頂いているバージョンのDeveloper's Guide(開発者ガイド)をご確認ください。

また、Management Consoleの以下の画面よりサンプルコードを取得することができます。

```
Sample Code to Execute 156_air2_his.robot

Java | C#

package com.example;

import com.kapowtech.robosuite.api.java.rql.*;
import com.kapowtech.robosuite.api.java.rql.construct.*;
import com.kapowtech.robosuite.api.java.rql.engine.*;
import com.kapowtech.robosuite.api.java.rql.engine.dist.*;
import com.kapowtech.robosuite.api.java.rql.engine.remote.*;
import com.kapowtech.robosuite.api.java.rql.engine.remote.socket.*;
import java.util.*;

public class Example {



    public static void main(String[] args) throws RQLException {

        // Create the builder for the execute request
        ExecuteRequestBuilder builder = new ExecuteRequestBuilder("Library:/xxx_呼び出すロボットの名前.robot");

        // Create RqlObjectBuilder to construct the Input input required to call this robot
        RQLObjectBuilder inputObject1= builder.createInputObject("Input");
        inputObject1.setAttribute("Config", (Binary) null);
        inputObject1.setAttribute("deptcd", (String) "TYO");
        inputObject1.setAttribute("arrivcd", (String) null);

        // Credentials used by RoboServer when accessing the repository
```

Time: 18:06 | Local Time: 18:06 | Logged in as: bitrobo

Data	Logs	Admin		
Snippets	Resources	OAuth		
Items per page: 40	Refresh Every: minute			
Last Modified	Run Now	API	REST	Download
012-11-30 13:30:29				
012-11-30 10:04:15				

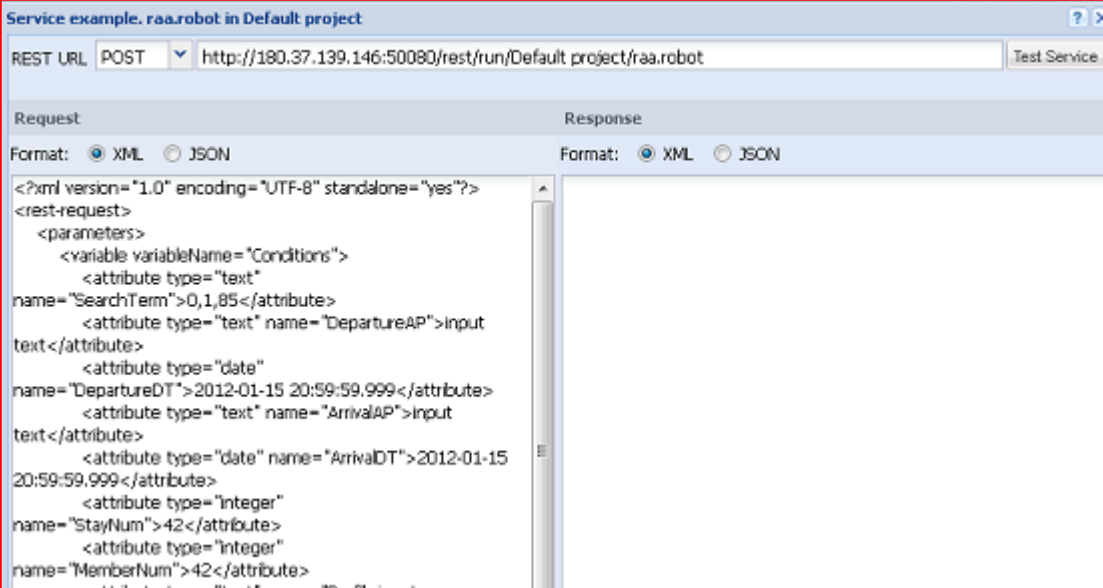
# RESTの使い方

REST インターフェイスを利用してWebサービスとしてロボットを実行することが可能です。

使用方法については、ご利用頂いているバージョンのUser's Guide(ユーザガイド)もしくはHelp(ヘルプ)サイトにて、REST 項目の内容をご確認ください。

Management Consoleの以下の画面よりサンプルコードを取得することができます。

ロボットに渡すパラメータ、および戻り値の形式をXMLまたはJSONで指定することができます。



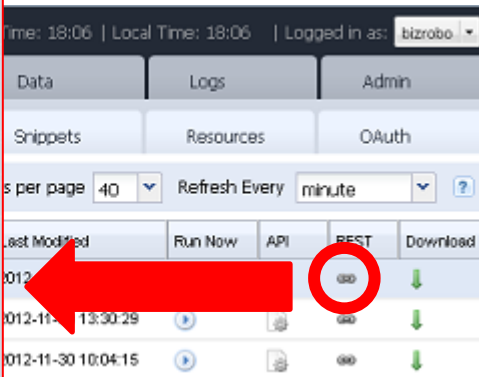
Service example. raa.robot in Default project

REST URL POST http://180.37.139.146:50080/rest/run/Default project/raa.robot Test Service

Request Response

Format: XML JSON Format: XML JSON

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rest-request>
  <parameters>
    <variable variableName="Conditions">
      <attribute type="text"
name="SearchTerm">0,1,85</attribute>
      <attribute type="text" name="DepartureAP">input
text</attribute>
      <attribute type="date"
name="DepartureDT">2012-01-15 20:59:59.999</attribute>
      <attribute type="text" name="ArrivalAP">input
text</attribute>
      <attribute type="date" name="ArrivalDT">2012-01-15
20:59:59.999</attribute>
      <attribute type="integer"
name="StayNum">42</attribute>
      <attribute type="integer"
name="MemberNum">42</attribute>
    
```



Time: 18:06 | Local Time: 18:06 | Logged in as: bitrobo

Data Logs Admin

Snippets Resources OAuth

Items per page 40 Refresh Every minute

Last Modified	Run Now	API	REST	Download
012-11-30 13:30:29			REST	Download
012-11-30 10:04:15			REST	Download