RADS

Integrating With External System

https://github.com/radservice/rad-community/fork

Last Revised on Oct 2021

Prerequisites

1. Understand RADS components in depth and able to make full use of its components.
2. Avid software developer with vast know-hows of web application technologies.

Content

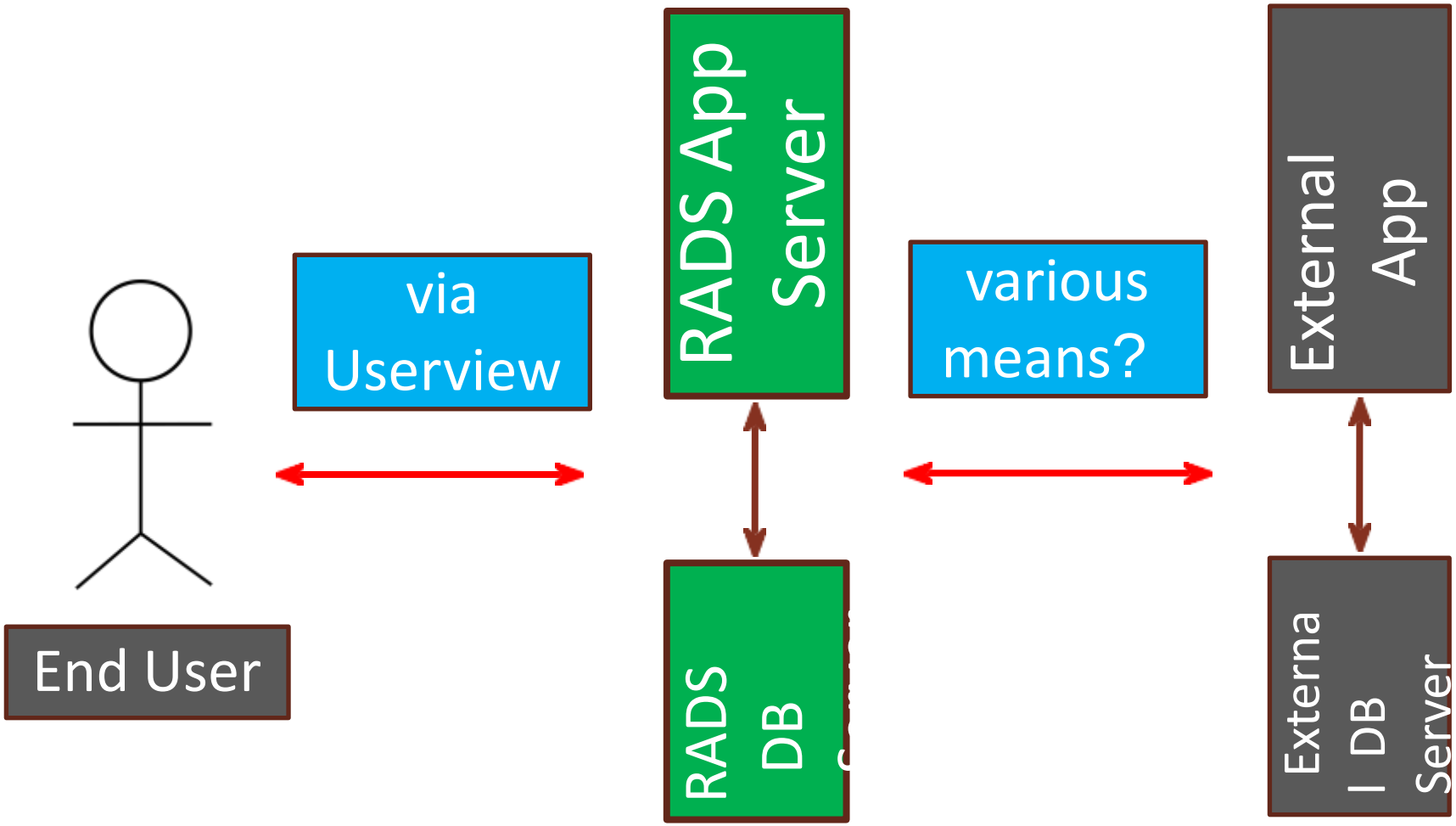
Chapter 1

Introduction

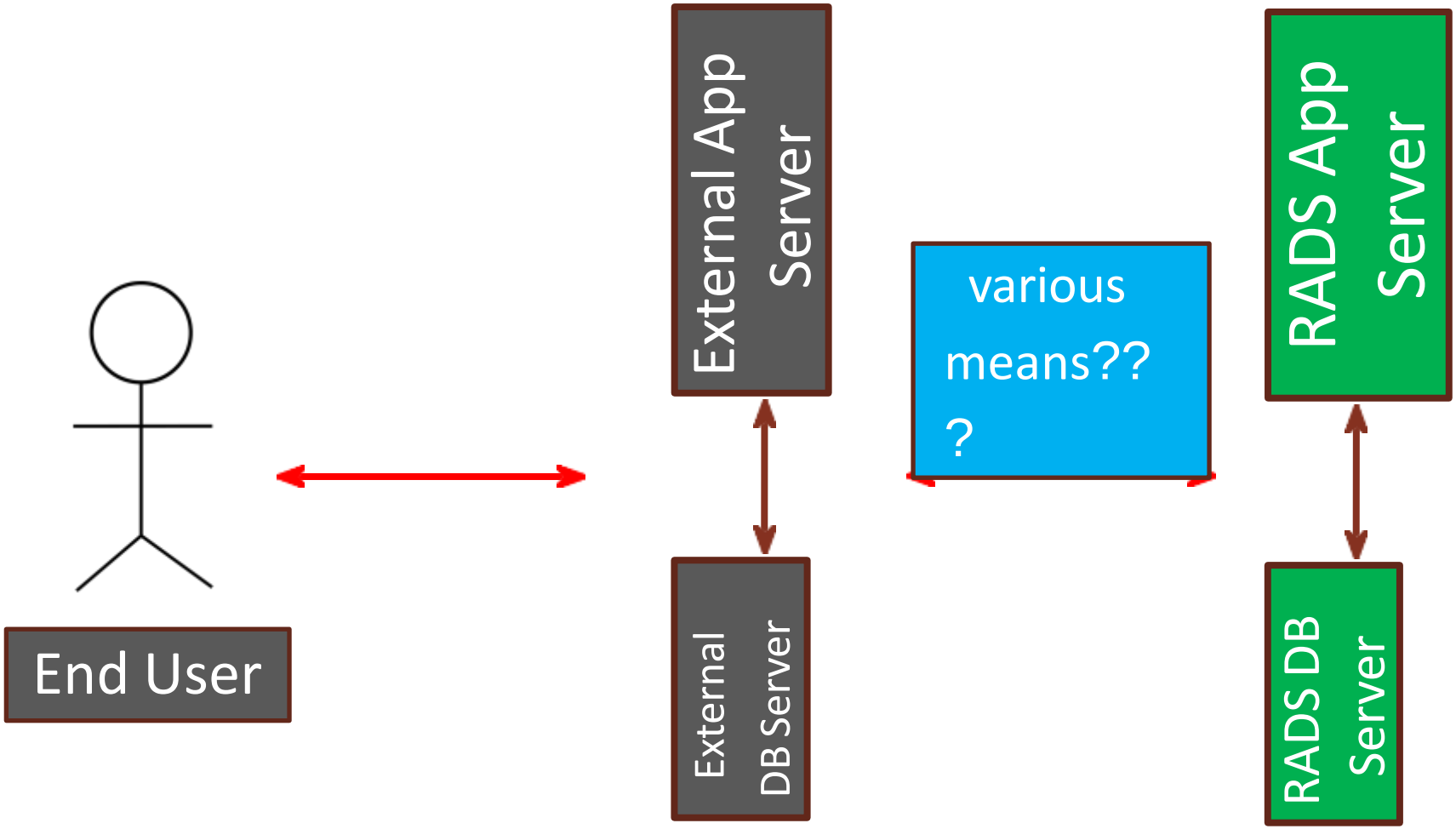
Introduction

- There are various ways to integrate RADS with external applications.
- Typically, there are 4 main components and the client user to be considered about.
  1. RADS Application Server
  2. RADS Database Server
  3. External Application Server
  4. External Application Database Server
  5. Client user

RADS as the Front-end

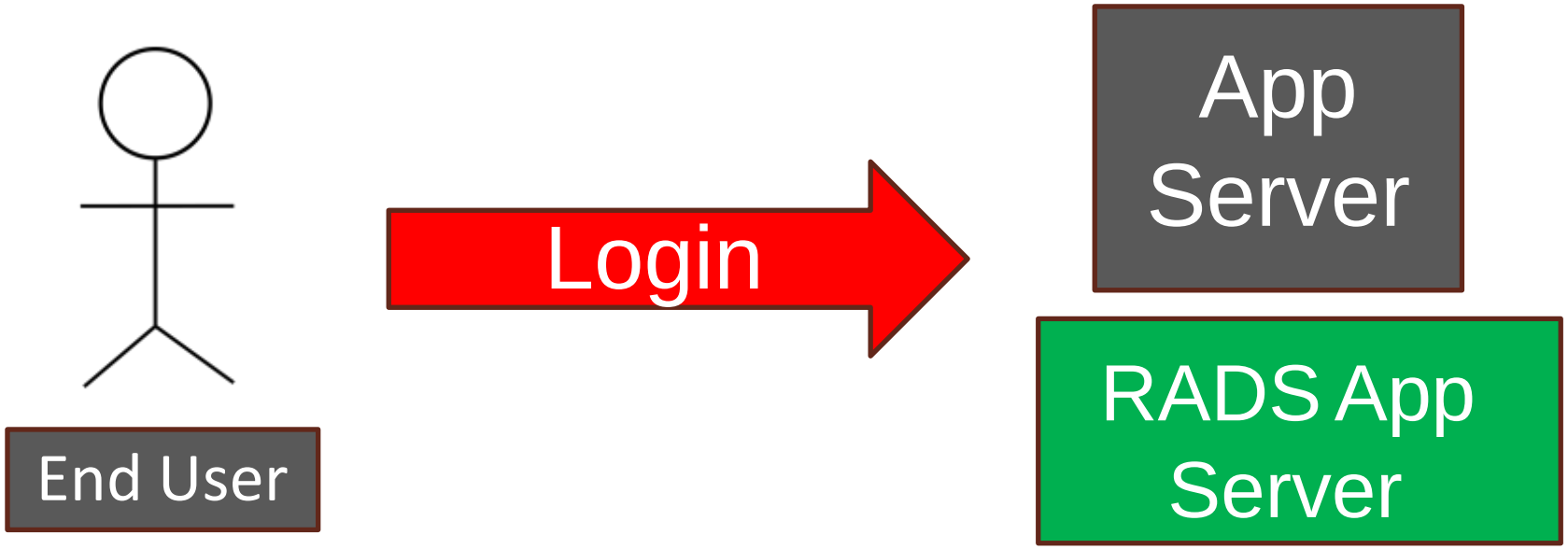


External App as the Front-end



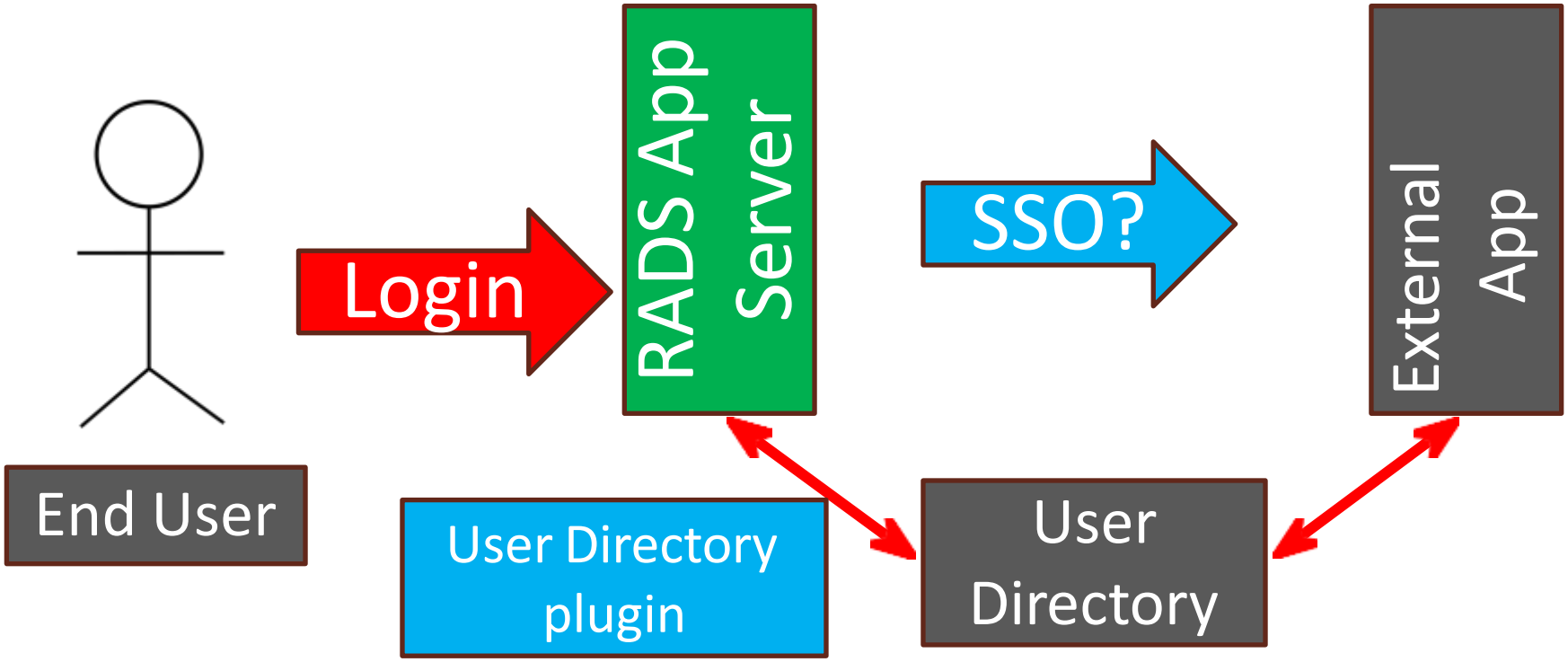
User Directory

- Is User session important?
- How do you want to handle the user session between the servers (RADS server and external application server)?



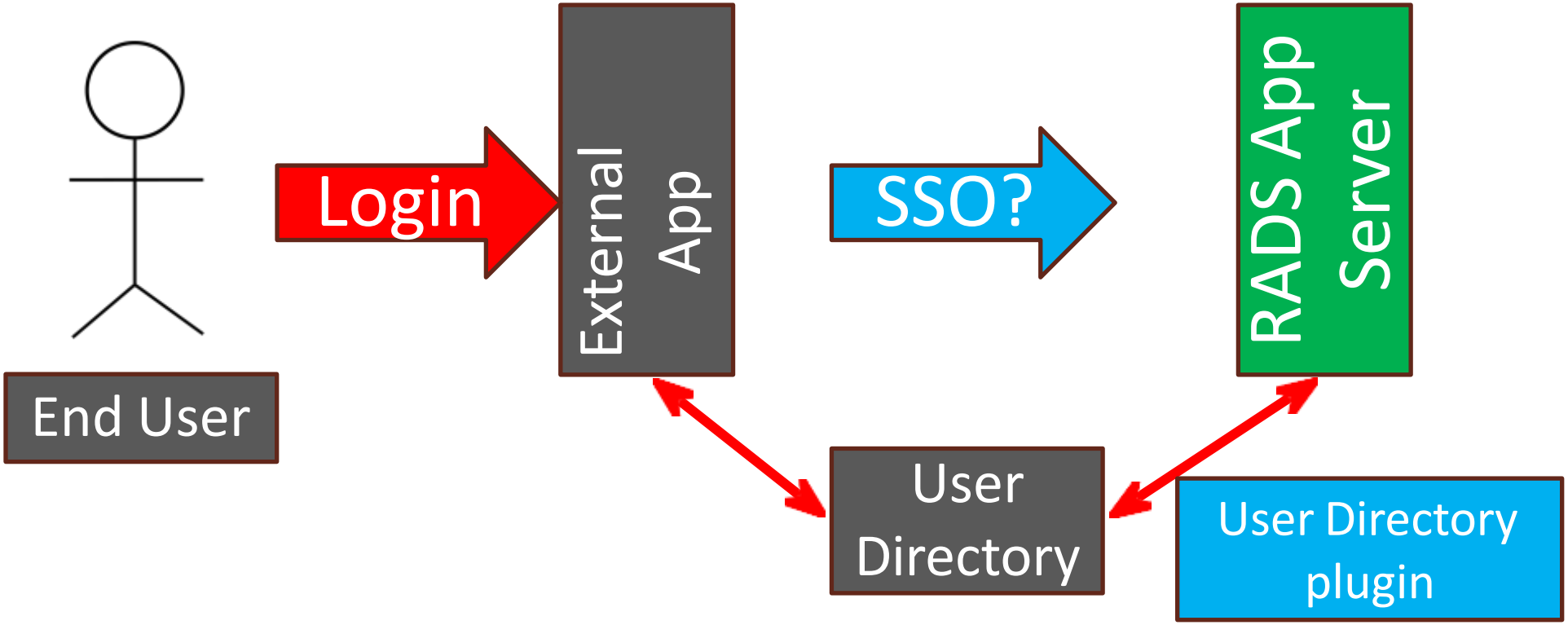
User Directory - RADS as Front-end

- Is the identity of the End User important to the External App Server?



User Directory - Ext App as Front-end

- Is the identity of the End User important to the RADS App Server?



Chapter Review

- Understand various ways of integration with RADS and external application server.

Chapter 2

JSON API

JSON API

- RADS provides a comprehensive list of APIs for Process related tasks.
- Full list of the APIs can be obtained from https://docs.rads.purwana.net/JSON+API

Sample List of APIs

- web/json/monitoring/running/process/list
- web/json/workflow/currentUsername
- web/json/workflow/assignment/completeWithVariable/(:activityId)
- web/json/workflow/assignment/complete/(:activityId)

Before We Start!

- Ensure you have configured the **API Domain Whitelist** in **General Settings** to allow JSON API requests.
- If a request is from a non-whitelisted domain, the response will be a HTTP 400 Bad Request

web/json/monitoring/running/process/list

- **URL:** /web/json/monitoring/running/process/list
- **Method:** HTTP GET/POST
- **Description:** Retrieve running process list
- **Parameters**
  - packageId - (Optional) package id
  - processId - (Optional) process definition id without version
  - processName - (Optional) process name
  - version - (Optional) process version
  - sort - (Optional) column name to be sorted
  - desc - (Optional) Boolean value to determine whether to sort by ascending or descending order (true equals to descending)
  - start - (Optional) where rows start from
  - rows - (Optional) number of rows per page

Sample Result


```
{
  "total":2,
  "desc":false,
  "sort":"name",
  "start":0,
  "data":
  [
    {"id":"3724_mdec_v1002_mdec_wp1",
      "serviceLevelMonitor":"<span
class=\"dot_red\"><\/span>",
      "name":"mdec_wp1","state":"open.running",
      "due":"Fri Mar 20 14:01:27 SGT 2009",
      "startedTime":"Fri Mar 20 13:51:27 SGT
2009","version":"2"},
    {"id":"3725_mdec_v1002_mdec_wp1",
      "serviceLevelMonitor":"<span
class=\"dot_red\"><\/span>",
      "name":"mdec_wp1",
      "state":"open.running",
      "due":"Fri Mar 20 14:03:16 SGT 2009",
      "startedTime":"Fri Mar 20 13:53:16 SGT 2009",
      "version":"2"}
  ]
}
```

web/json/workflow/currentUsername

- **URL**
  /web/json/workflow/currentUsername
- **Method**
  HTTP GET/POST
- **Description**
  Get current logged in user's username
- **Parameters**
  - callback - (Optional) a function (in JavaScript) to call back after invoking this method

Sample Result
{ "username":"admin" }

/web/json/workflow/assignment/completeWithVariable/(:activityId)

- **URL**
  /web/json/workflow/assignment/completeWithVariable/(:activityId)
- **Method** HTTP POST
- **Description**
  Set activity variable
  Variables can be passed as parameters with the **var_** prefix
- **Parameters**
  - callback - (Optional) a function (in JavaScript) to call back after invoking this method
  - activityId - activity id
  - var_(workflow variable id) - (Optional) set workflow variable value

/web/json/workflow/assignment/completeWithVariable/(:activityId)

Sample Result

{
"activityId": "1079_563_crm_process1_approve_proposal", "assignment": "admin",
"nextActivityId": "1093_563_crm_process1_send_proposal",
"processId": "563_crm_process1",
"status": "completed“
}

Chapter Review

- Understand and able to retrieve the list of available APIs.

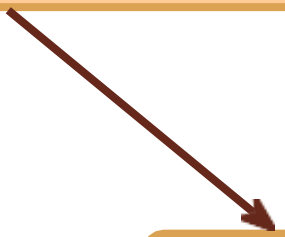
Chapter 3

JSON API Authentication

Using JSON API

- Can be called using AJAX call (front-end/web) or through server-backend to retrieve data from or to perform action to RADS System.
- Return response in JSON format.
- Example:

web/json/workflow/assignment/list/count



{"total":11}

Passing Parameters

{Host}/rad**/web/json/workflow/assignment/list/count**

This JSON API is accessed as **anonymous**

{Host}/rad**/web/json/workflow/assignment/list/count?j_username=admin &j_password=admin**

{Host}/rad**/web/json/workflow/assignment/list/count?j_username=admin &hash=*14ACD782DCFEB2BCDE2B271CCD559477***

By using **j_username** and **j_password** or **hash** parameter.
This JSON API is accessed as **admin**

Password Hashing

- Used in JSON API authentication and JavaScript Single Sign ON (SSO).
- Prevents a user's password from being directly exposed during authentication.
- **Formula**
  - md5(username + “::” + md5Base16(password))
  - E.g. Assuming that the username is “admin” and the password is “admin”, the resulting hash should be “**14ACD782DCFEB2BCDE2B271CCD559477**”.

- Online reference: https://docs.rads.purwana.net/Hashed+Password

Password Hashing Sample Java Code

```
public static String md5(String content)
    { try {
        MessageDigest m = MessageDigest.getInstance("MD5");
        byte[] data = content.getBytes();
        m.update(data, 0, data.length);
        BigInteger i = new BigInteger(1, m.digest());
        return String.format("%1$032X", i);
    } catch (Exception ex) {}
    return "";
}

public static String md5Base16(String content)
    { try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] bytes =
        md.digest(content.getBytes()); StringBuffer
        sb = new StringBuffer();
        for (int i = 0; i < bytes.length; i++) {
            byte b = bytes[i];
            String hex = Integer.toHexString((int) 0x00FF &
            b); if (hex.length() == 1) {
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString();
    } catch (Exception e) {}
    return "";
}
```

Master Login

- Master Login Username and Password is a set of credential that can be used to login to RADS system as different user.
- This is particularly catered to integration needs:-
  - Server-backend to RADS server where individual's credential is not required to perform calls on behalf.

- Reference:
https://docs.rads.purwana.net/JSON+API+Authentication

JSON API Authentication – Master Login

System Administration Settings

Master Login Username: master

Master Login Password: ••••••

master login hash:0b244463ce522dc21bc58b42d13abcf0

In System Setting > General Setting

{Host}/rad**/web/json/workflow/assignment/list/count?j_username={master-login-use rname}&j_password={master-login-password}&loginAs={username}**

{Host}/rad**/web/json/workflow/assignment/list/count?j_username={master-login-use rname}&hash={master-login-hash}&loginAs={username}**

By using **Master Login** feature, user's password is not exposed in JSON API authentication.

Basic Http Authentication

- A set of user credential encoded to login to RADS system.
- Credentials are passed in the request header.
- Username and password not exposed in clear text
- **Formula:**
  - "Basic" + base64(username + “:” + password)
  - E.g. Assuming that the username is “admin” and the password is “admin”, the result should be “**Basic YWRtaW46YWRtaW4=**”.
- Reference: https://docs.rads.purwana.net/JSON+API+Authentication#JSONAPIAuthentication-BasicHttpAuthentication

Caution

- Do **NOT** expose clear text password in the DOM
- Ensure that "API Domain Whitelist" & "API IP Whitelist" is configured according to principle of least privilege

Chapter Review

- Understand how to authenticate with RADS's JSON APIs.

Chapter 4

JavaScript API

Introduction

- Javascript API is a helper/utility with a set of methods to ease integration calls

```
<script type="text/javascript"
src="http://localhost:8080/rad/js/jquery/jquery-1.9.1.min.js"></s
cript>
<script type="text/javascript"
src="http://localhost:8080/rad/js/json/util.js" ></script>

<script type="text/javascript" >
[put your code here]
</script>
```

- util.js source code: \rad-consoleweb\src\main\webapp\js\json\util.js (GithHub link: https://github.com/radservice/rad-community/blob/master/rad-consoleweb/src/main/webapp/js/json/util.js)
- Reference: https://docs.rads.purwana.net/JavaScript+API

List of Available Methods

- ConnectionManager.post(url, callback, params)
- ConnectionManager.ajaxJsonp(url, callback, params)
- ConnectionManager.get(url, callback, params, xss)
- AssignmentManager.getCurrentUsername(baseUrl, callback)
- AssignmentManager.login(baseUrl, username, password, callback)
- AssignmentManager.loginWithHash(baseUrl, username, hash, callback)
- AssignmentManager.logout(baseUrl)
- AssignmentManager.completeAssignment(baseUrl, activityId, redirect)
- AssignmentManager.completeAssignmentWithVariable(baseUrl, activityId, variableData, redirect)
- UrlUtil.encodeUrlParam(url)
- getUrlParam(paramName)

And many more...

Sample usage - Start New Process Instance

- Starting a new process by calling the JSON API

Code:

```
url =
"http://localhost:8080/rad/web/json/workflow/process/start/purchaseRequest:latest:purchaseRequestProces
s ";

callback = {
     success: function(response){
          console.log(response);
     }
};
params = {var_status : 'test' };
ConnectionManager.post(url, callback, params);
```

Result:

```
[processId=6_purchaseRequest_purchaseRequestProcess,
processDefId=purchaseRequest#1#purchaseRequestProcess,
participantId=applicant, next user=[admin]]
```

- Sample coding can be obtained at 18.4.1.txt

Sample Usage - Get Current User

- Finding out the current logged in user by using the **`AssignmentManager.getCurrentUsername()`** method.

Code:

```
var callback = {
    success : function(response){
        console.log(response.username);
    }
}
AssignmentManager.getCurrentUsername('http://localhost:8080/rad', callback);
```

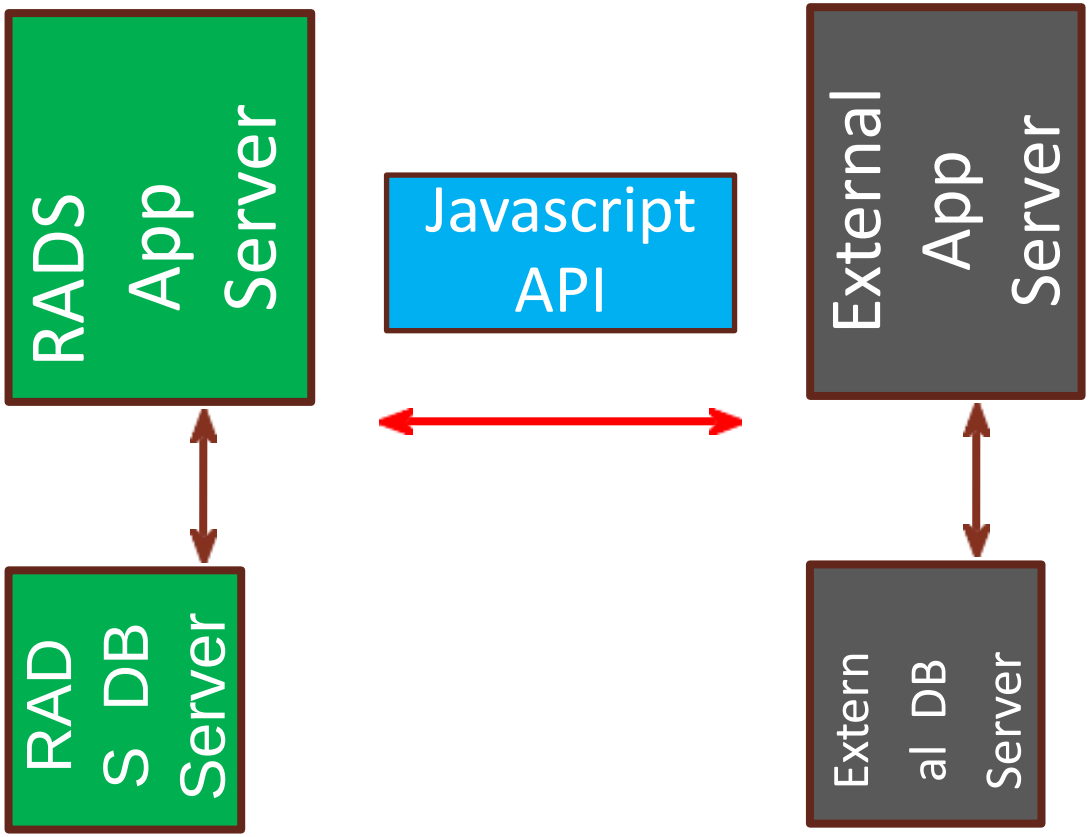
Result:

```
admin
```

- Sample coding can be obtained at 18.4.2.txt

Setup Suggestion

- Can you now imagine the following setup?



Chapter Review

- Understands the purpose and usages of the Javascript API.

Chapter 5

Single Sign On (SSO)

Introduction

- User logs in to external system and implicitly gains access to RADS without being prompted to login again.
- Can be achieved via Directory Manager plugins and programmatically via Web Service plugin

Reference:
https://docs.rads.purwana.net/Single+Sign+On+-+SSO

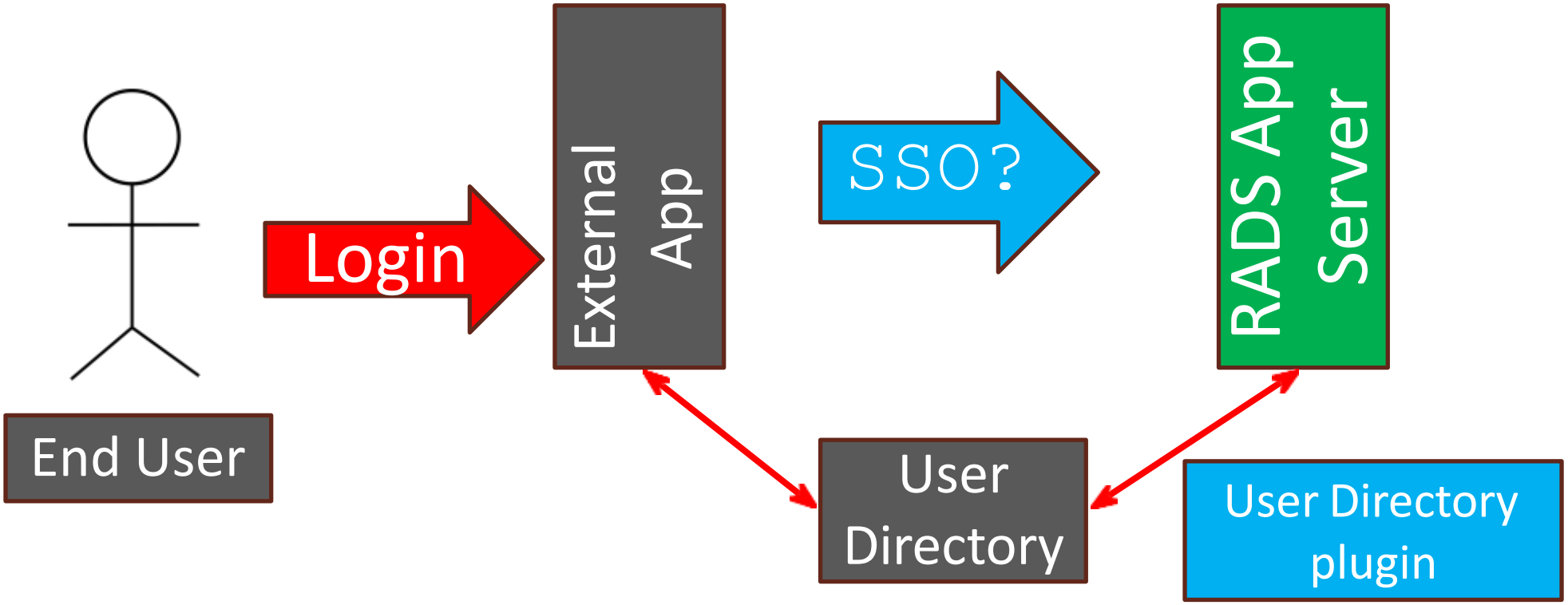
Various SSO Methods Available...

- G Suite
- Kerberos
- SAML (IDP initiated) (e.g.: Sharepoint, some ADs, etc.)
- OpenID Connect
- Programmatically via Web Service Plugin (Java)
- API Builder - SSO

See KB for detailed guides for all the above.

Setup Suggestion

- Can you now imagine the following setup?



Chapter Review

- Understand on how to use the SSO feature.

Chapter 6

Embedding Task Inbox into External System

Embedding Task Inbox into External System

- Display RADS **task inbox** in other web applications, such as portal (SharePoint, Liferay) and content management system (Joomla!, WordPress, Drupal, Alfresco).

Embed Code RSS

```
<link rel="stylesheet" type="text/css" href="http://localhost:8080/rad/css/portlet.css">
<script type="text/javascript" src="http://localhost:8080/rad/js/jquery/jquery-1.9.1.min.js"></script>
<script type="text/javascript" src="http://localhost:8080/rad/js/jquery/jquery-migrate-1.2.1.min.js"></script>
<script type="text/javascript" src="http://localhost:8080/rad/js/json/util.js"></script>
<div id="inbox1">
    <center><img src="http://localhost:8080/rad/images/v3/portlet_loading.gif" />
    </center>
</div>
```

Embedding Task Inbox into External System

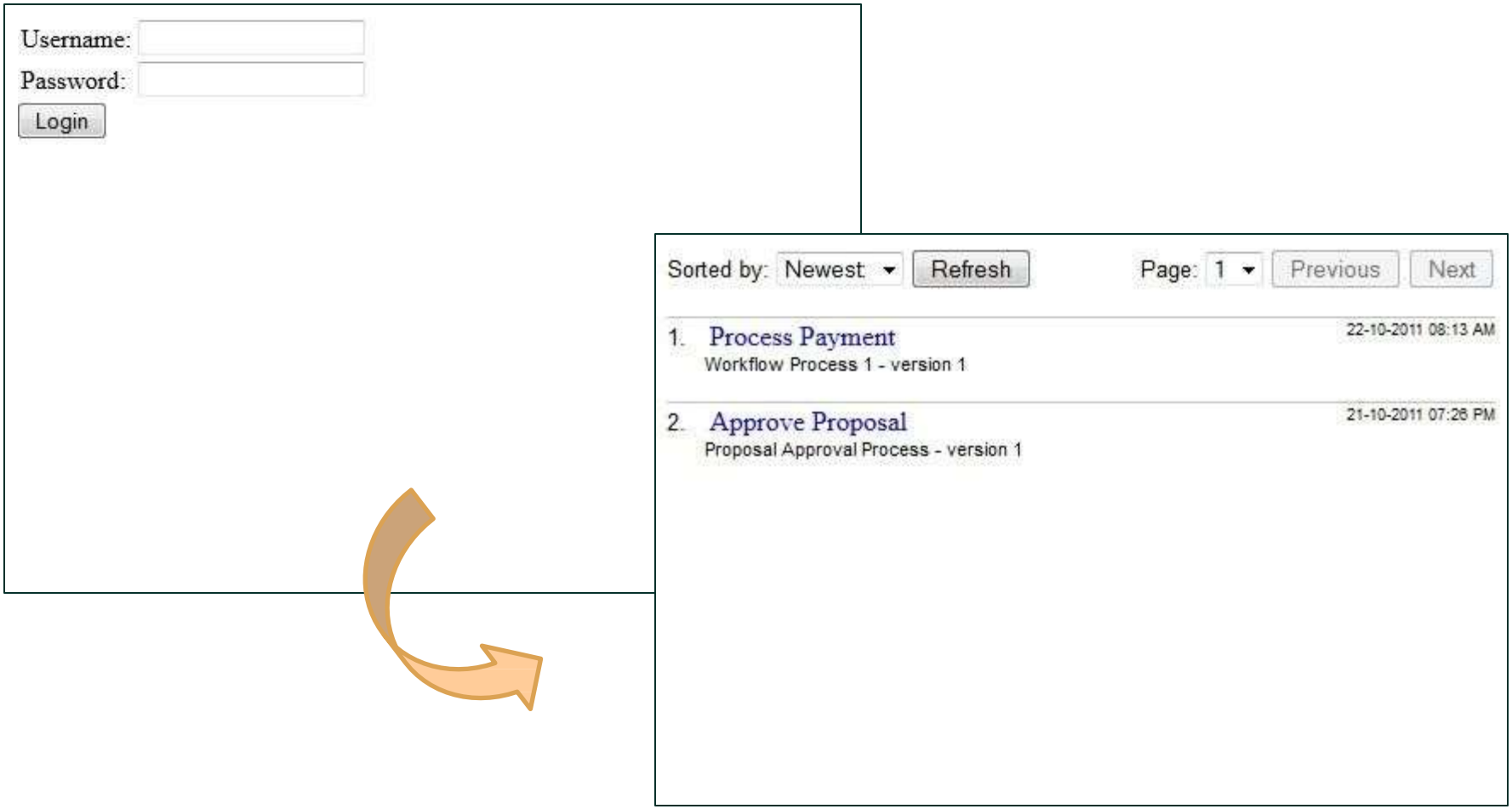
```
<link rel="stylesheet" type="text/css" href="http://localhost:8080/rad/css/portlet.css">
<script type="text/javascript"
src="http://localhost:8080/rad/js/jquery/jquery-1.9.1.min.js"></script>
<script type="text/javascript"
src="http://localhost:8080/rad/js/jquery/jquery-migrate-1.2.1.min.js"></script>
<script type="text/javascript" src="http://localhost:8080/rad/js/json/util.js"></script>
<div id="inbox1">
    <center><img src="http://localhost:8080/rad/images/v3/portlet_loading.gif" />
    </center>
</div>
<script type="text/javascript">
    $(document).ready(function() {

$.getScript('http://localhost:8080/rad/web/js/client/inbox.js?id=1&rows=5&divId=inbox1',
null);
    });
</script>
```

Unique id to identify the inbox instance

Number of rows to be displayed in this inbox instance

ID of the HTML DIV to display this inbox instance

Embedding Task Inbox into External System



Embedding Task Inbox into External System

Customize the look and feel of embedded task inbox by pointing to a new css file.

```
<link rel="stylesheet" type="text/css" href="http://localhost:8080/rad/css/portlet.css">
<script type="text/javascript"
src="http://localhost:8080/rad/js/jquery/jquery-1.9.1.min.js"></script>
<script type="text/javascript"
src="http://localhost:8080/rad/js/jquery/jquery-migrate-1.2.1.min.js"></script>
<script type="text/javascript" src="http://localhost:8080/rad/js/json/util.js"></script>
<div id="inbox1">
    <center><img src="http://localhost:8080/rad/images/v3/portlet_loading.gif" />
    </center>
</div>
<script type="text/javascript">
    $(document).ready(function() {

$.getScript('http://localhost:8080/rad/web/js/client/inbox.js?id=1&rows=5&divId=inbox1',
null);
    });
</script>
```

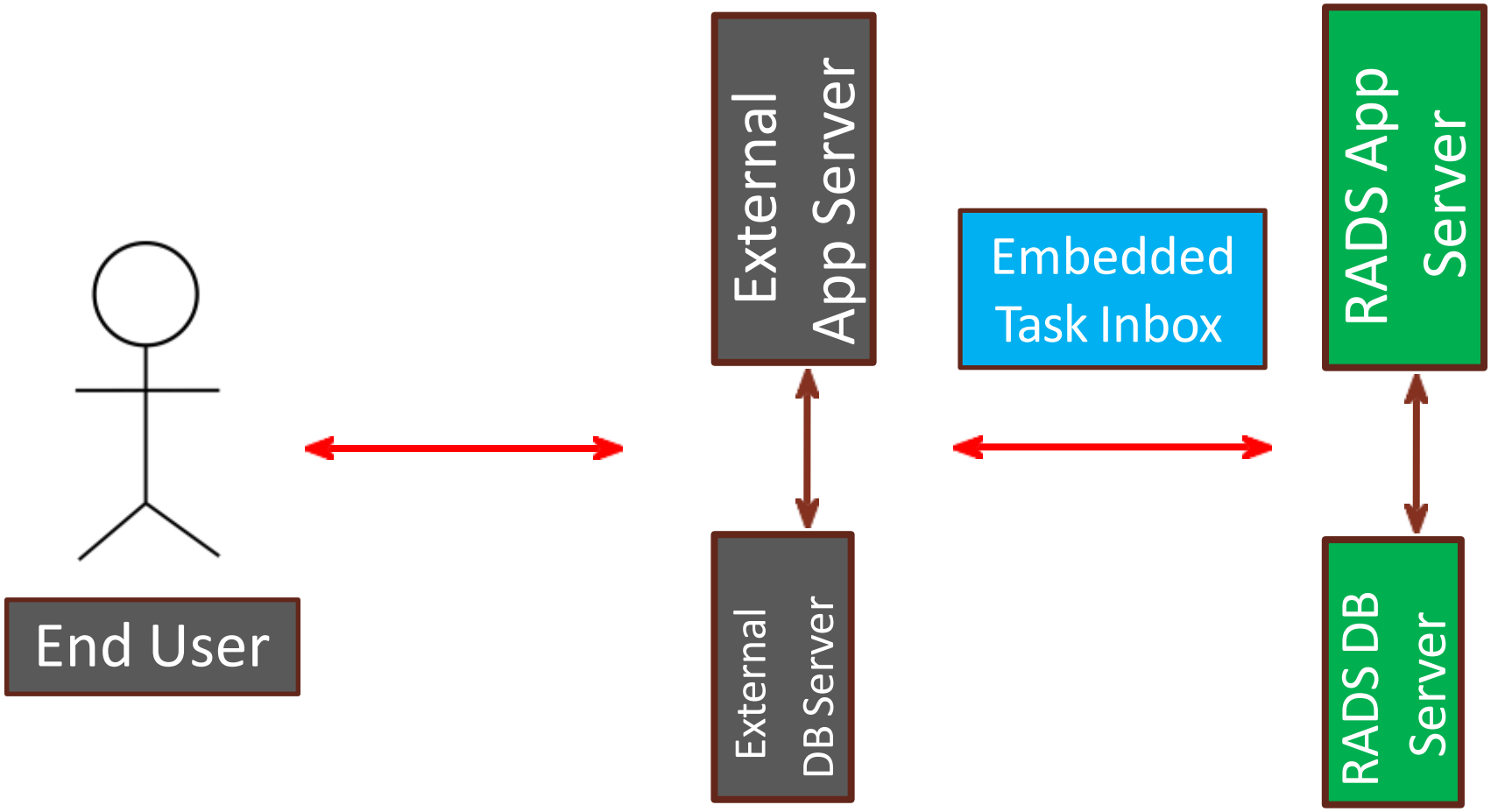
Embedding Task Inbox into External System with SSO

```
<link rel="stylesheet" type="text/css" href="http://localhost:8080/rad/css/portlet.css">
<script type="text/javascript"
src="http://localhost:8080/rad/js/jquery/jquery-1.9.1.min.js"></script>
<script type="text/javascript"
src="http://localhost:8080/rad/js/jquery/jquery-migrate-1.2.1.min.js"></script>
<script type="text/javascript" src="http://localhost:8080/rad/js/json/util.js"></script>
<div id="inbox1">
    <center><img src="http://localhost:8080/rad/images/v3/portlet_loading.gif" />
    </center>
</div>
<script type="text/javascript">
    $(document).ready(function(){
      var loginCallback = {
          success: function(){
$.getScript('http://localhost:8080/rad/web/js/client/inbox.js?id=1&rows=5&divId=inbox1',
null);
      }};
      AssignmentManager.login('http://localhost:8080/rad', 'admin', 'admin',
      loginCallback);
});
```

Load script after successfully logged in

Setup Suggestion

- Can you now imagine the following setup?



Chapter Review

- Understand on how to embed Task Inbox into external app.

Chapter 7

Embedding Userview Page in an iFrame

Embedding Userview Page in an IFrame

Embedding Userview Page in an IFrame

- The header, menu & footer of Userview can be removed by following:
  - By adding parameter "**embed=true**" in the **URL** OR
  - By modifying the **URL**

From

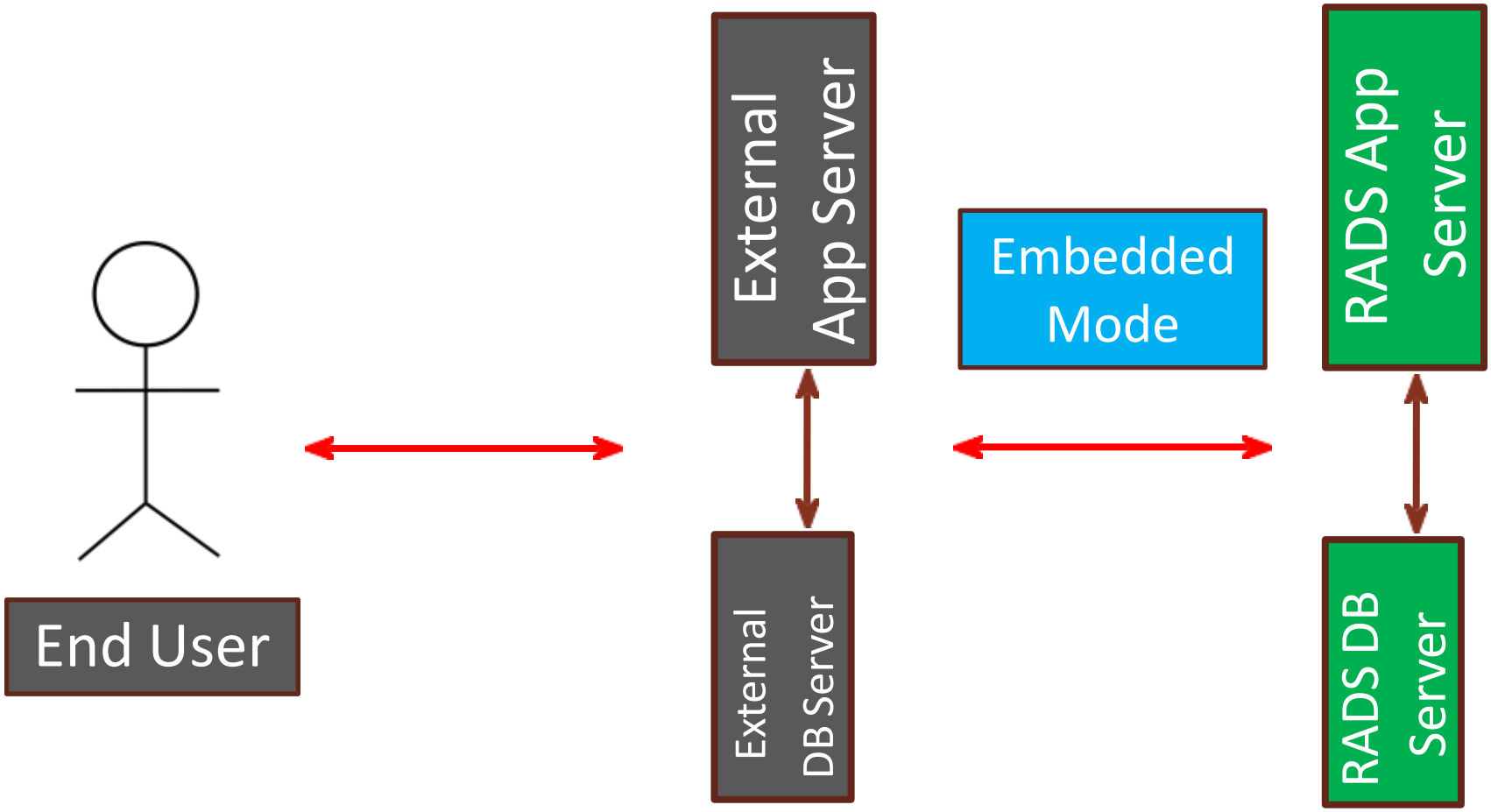
http://localhost:8081/rad/web/userview/crm/crm_userview_sales

To

http://localhost:8081/rad/web/**embed/**userview/crm/crm_userview_sales

Setup Suggestion

- Can you now imagine the following setup?



Exercise

- Choose a Userview Page.
- Add in the necessary parameter to turn it into the Embedded mode.
- Use iFrame to display it in your custom app.

Chapter Review

- Understands the purpose of using Embedded mode of Userview pages.

Chapter 8

Using the JSON Tool

Introduction

- The JSON Tool enables one to issue a JSON web service call, and to save the returned data into RADS's form data and/or into the process's workflow variable.

- Reference: https://docs.rads.purwana.net/JSON+Tool

How it works?

- The JSON Tool will call the JSON API endpoint, along with any required parameters.

Configure JSON Tool

Configure JSON Tool > Store To Form > Store To Workflow Variable

JSON URL *	https://api.binance.com/api/v3/ticker/price?symbol=BTCUSDT
Call Type	GET
Request Headers	NAME | VALUE
No Response Expected	☐
Debug Mode	☐

How it works?

- Data will be returned in JSON format.

```
INFO  28 Jul 2021 04:27:26 net.purwana.rads.apps.app.lib.JsonTool
      - GET :
https://api.binance.com/api/v3/ticker/price?symbol=BTCUSDT

INFO  28 Jul 2021 04:27:26 net.purwana.rads.apps.app.lib.JsonTool   -
https://api.binance.com/api/v3/ticker/price?symbol=BTCUSDT returned with status :
200

INFO  28 Jul 2021 04:27:26 net.purwana.rads.apps.app.lib.JsonTool   -
{"symbol":"BTCUSDT","price":"38069.47000000"}
```

```
{"symbol":"BTCUSDT","price":"38069.47000000"}
```

How it works?

- Data can be then stored into Form table or Workflow Variable.

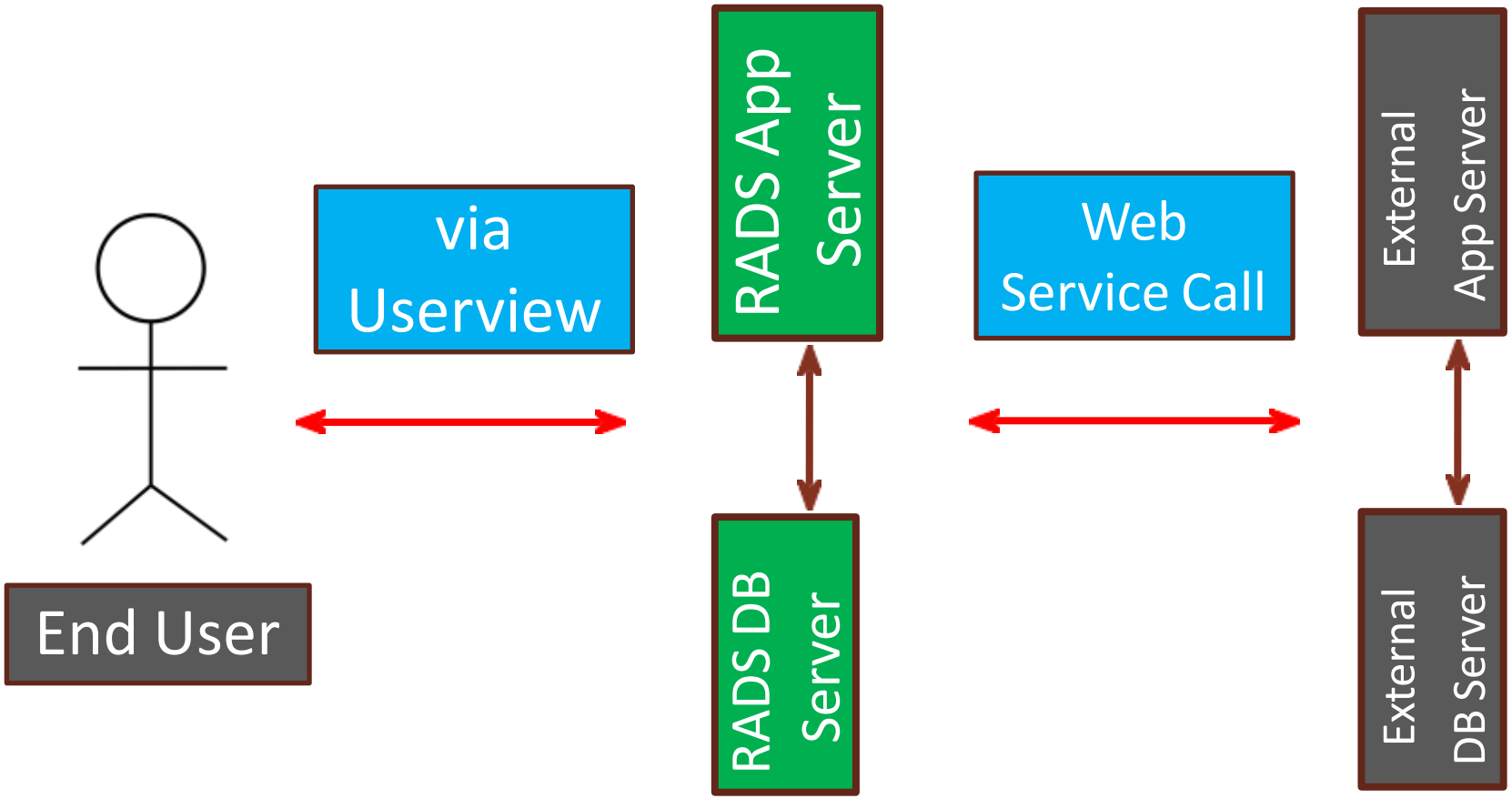
Exercise

- Try this JSON API endpoint: https://api.binance.com/api/v3/ticker/price?symbol=BTCUSDT
- Create a new RADS App with a **Activity -> Tool -> Activity** process.
- Map the Tool to a **JSON Tool**, configure accordingly.

Reference: https://binance-docs.github.io/apidocs/spot/en/#symbol-price-ticker

Setup Suggestion

- Can you now imagine the following setup?



Chapter Review

- Be able to make calls from RADS to other JSON web services by using the **JSON Tool**.

Chapter 9

Using the SOAP Tool

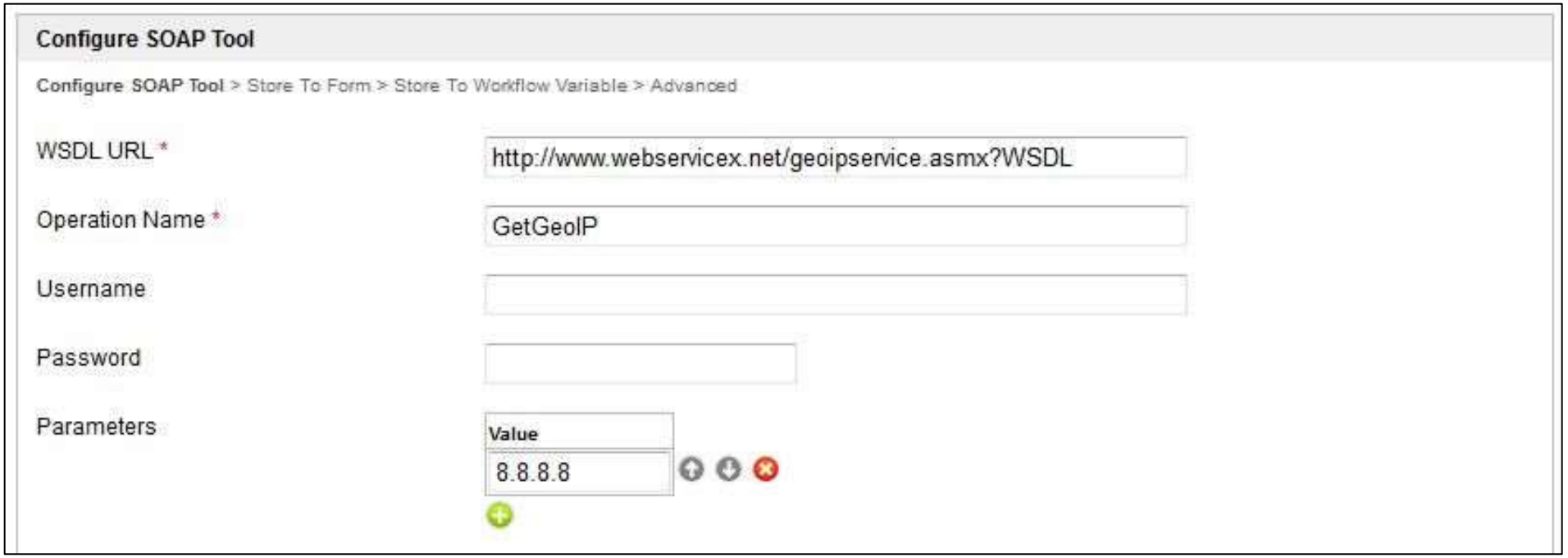
Introduction

- The SOAP Tool allows one to invoke web service for integration purpose to return useful information from external sources into the process instance.

- Reference: https://docs.rads.purwana.net/SOAP+Tool

How it works?

- The SOAP Tool will call the Web Service configured, passes the set of parameters set.



How it works?

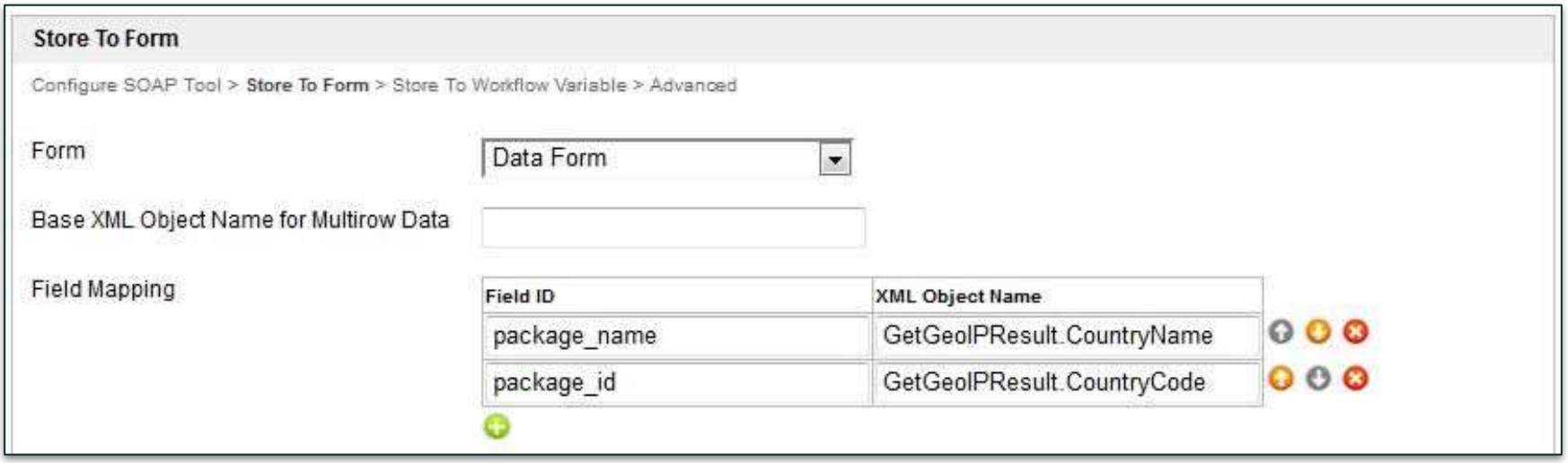
- Data will be returned in JSON format.

```
INFO 07 Jun 2013 10:54:37 SoapTool - <ns1:GetGeoIPResult
xmlns:ns1="http://www.webservicex.net/"><ns1:ReturnCode>1</ns1:ReturnCode><ns1:IP>8.
8.8.8</
ns1:IP><ns1:ReturnCodeDetails>Success</ns1:ReturnCodeDetails><ns1:CountryName>United
States</ns1:CountryName><ns1:CountryCode>USA</ns1:CountryCode></n s1:GetGeoIPResult>
INFO 07 Jun 2013 10:54:37 SoapTool - {"GetGeoIPResult":{"CountryName":"United
States","ReturnCodeDetails":"Success","ReturnCode":"1","IP":"8.8.8.8",
"CountryCode":"USA"}}
```


```
{
    "GetGeoIPResult": {
        "CountryName": "United States",
        "ReturnCodeDetails": "Success",
        "ReturnCode": "1",
        "IP": "8.8.8.8",
        "CountryCode": "USA"
    }
}
```

How it works?

- Data can be then stored into Form table or Workflow Variable.

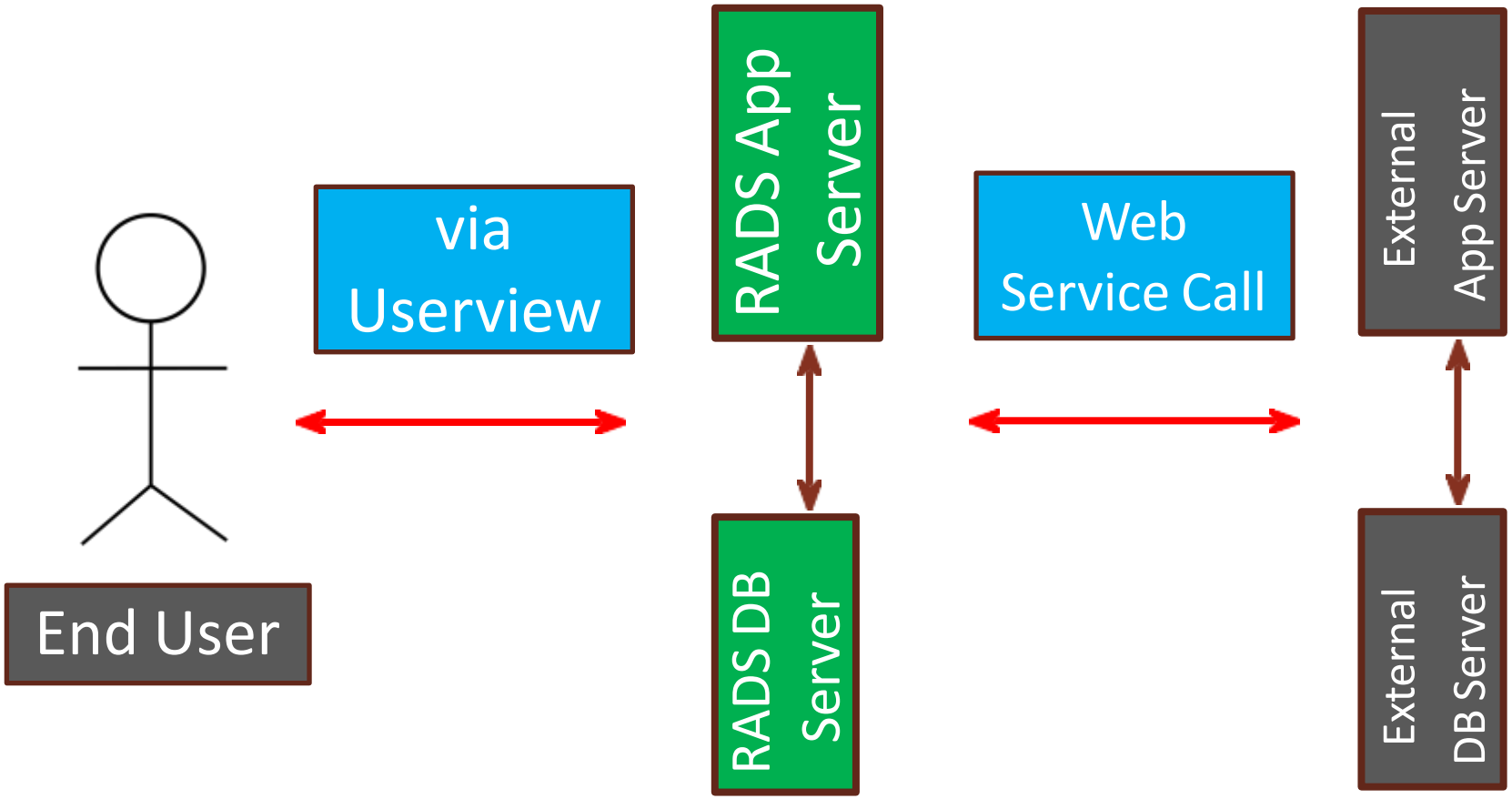


Exercise

- Go to https://documenter.getpostman.com/view/8854915/Szf26WHn#33a2b225-11a6-48d3-a695-fb0989cc4971
- Select a service, e.g.: "List of Countries by Name"
- Create a new RADS App with a **Activity -> Tool -> Activity** process.
- Map the Tool to a **SOAP Tool**, configure accordingly.

Setup Suggestion

- Can you now imagine the following setup?



Chapter Review

- Be able to make calls from RADS to other web services by using the **SOAP Tool**.

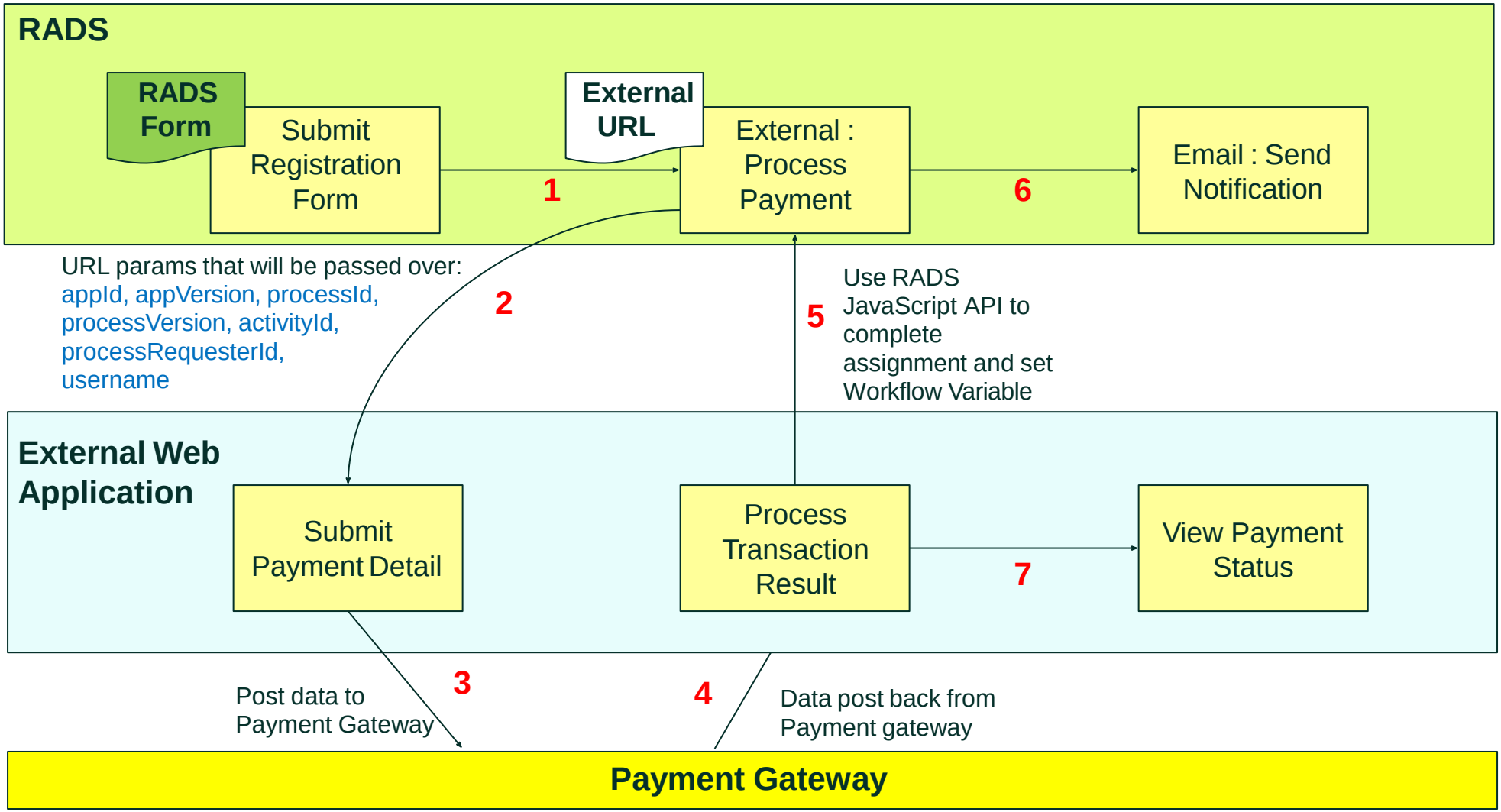
Chapter 10

Integrating with External Form

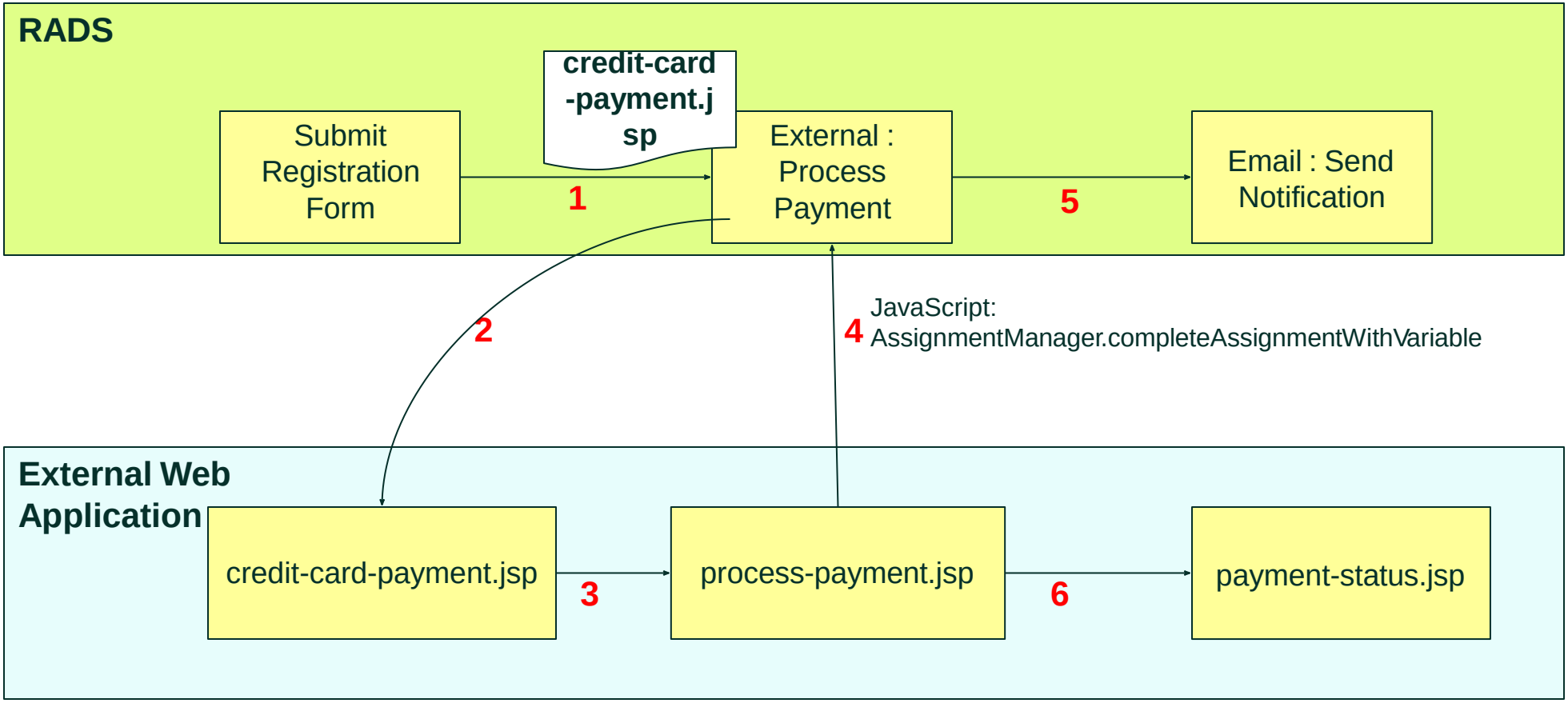
Integrating with External Form

- To perform (form) activities that RADS is not designed to do.
- To integrate with web data form built in external system, regardless of platform.
- To allow data to be submitted into external system within a process.

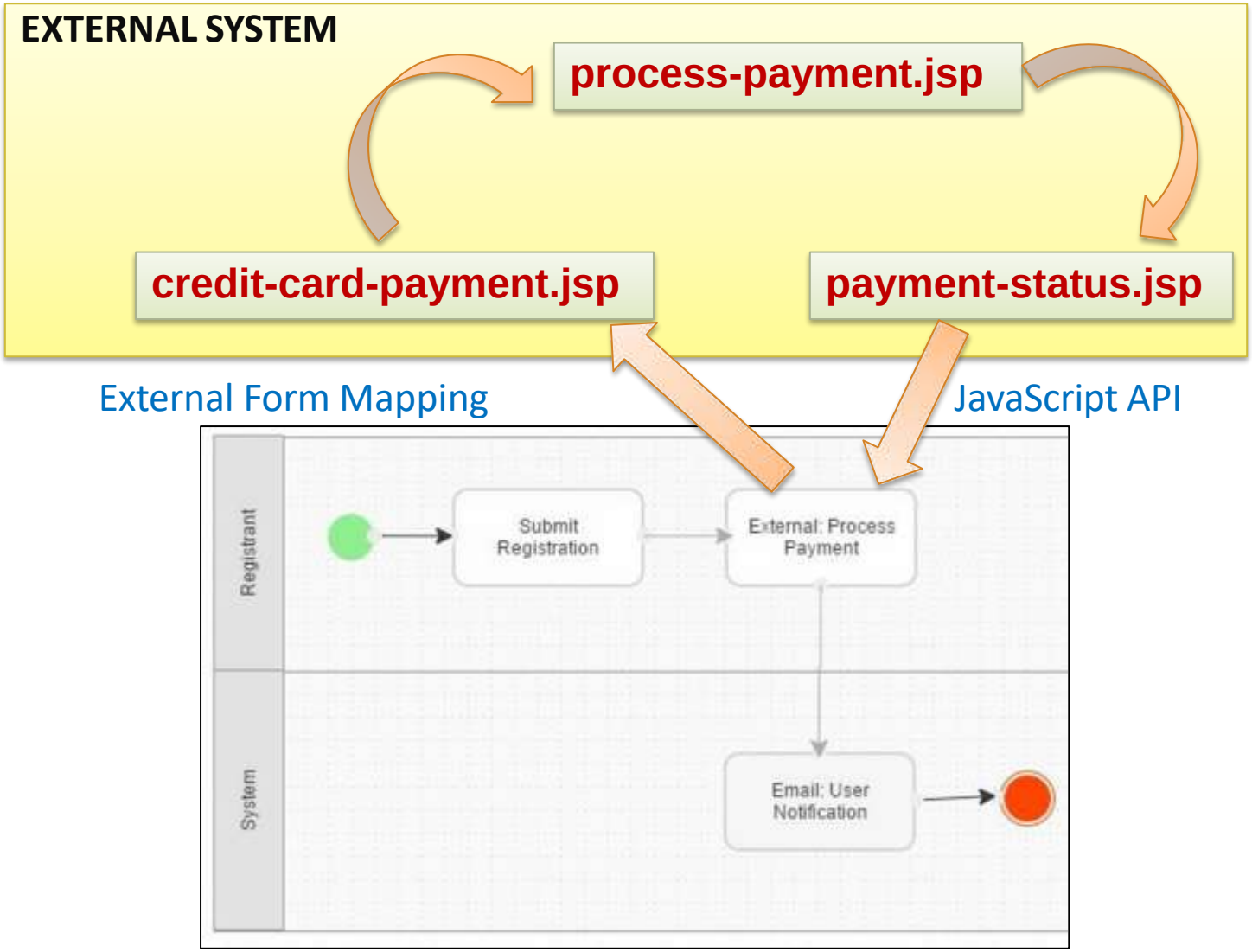
Scenario - Integrate with Payment Gateway



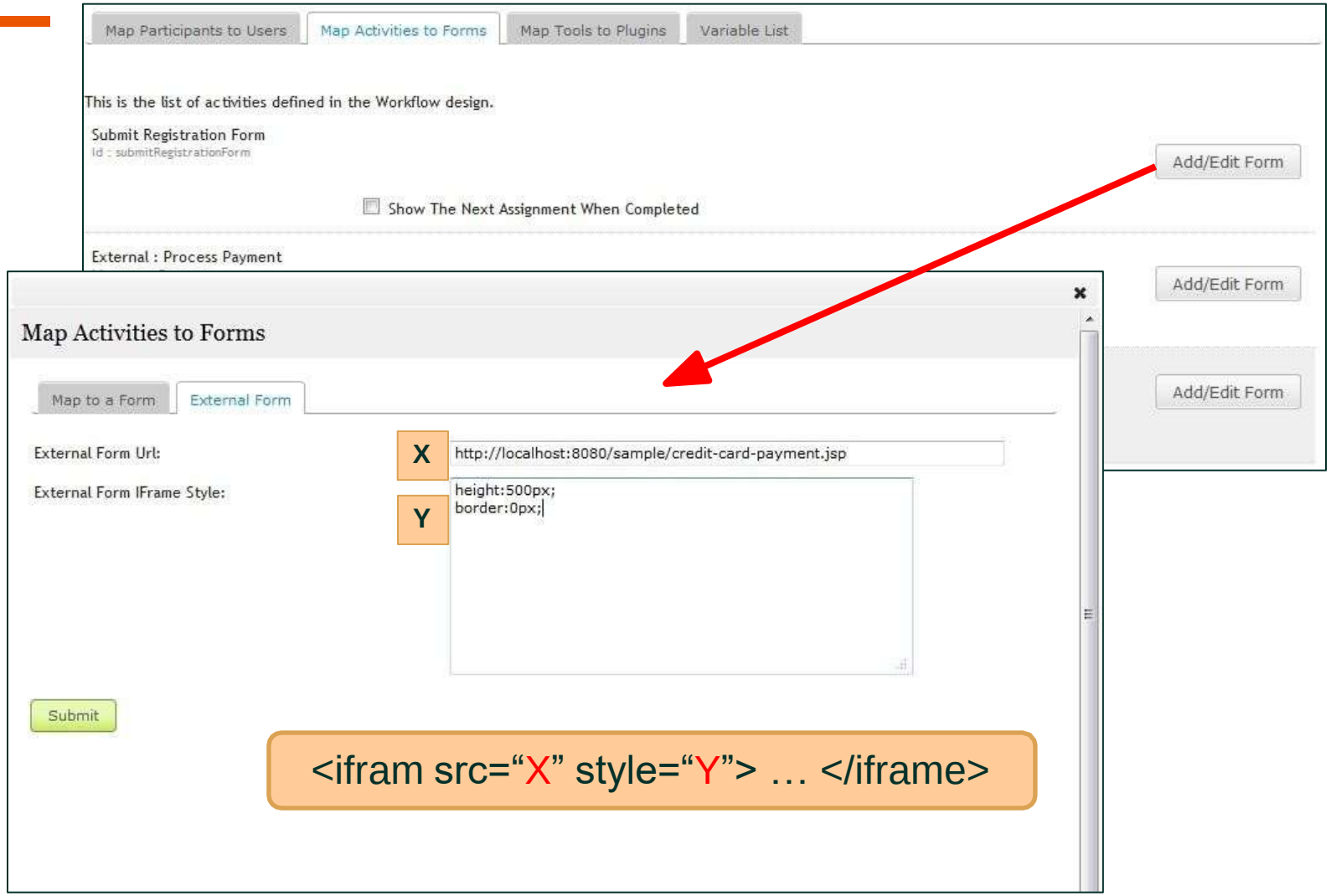
Example


RADS
credit-card-payment.jsp
Submit Registration Form
1
External : Process Payment
5
Email : Send Notification
2
4
JavaScript: AssignmentManager.completeAssignmentWithVariable
External Web Application
credit-card-payment.jsp
3
process-payment.jsp
6
payment-status.jsp

Example - Process



Map to an External Form



credit-card-payment.jsp

- Parameters that will be passed into external form from RADS:
  - appId
  - appVersion
  - processId
  - processVersion
  - activityId
  - processRequesterId
  - username
- **credit-card-payment.jsp** is the first external URL called, but it can call to any other web pages in the series, before getting back to RADS activity.

credit-card-payment.jsp

```
<h1>Credit Card Payment</h1>

<p>Parameters passed into this page:
<ul>
  <li>appId: <%=request.getParameter("appId")%></li>
  <li>appVersion: <%=request.getParameter("appVersion")%></li>
  <li>processId: <%=request.getParameter("processId")%></li>
  <li>processVersion: <%=request.getParameter("processVersion")%></li>
  <li>activityId: <%=request.getParameter("activityId")%></li>
  <li>processRequesterId :
<%=request.getParameter("processRequesterId")%></li>
  <li>username: <%=request.getParameter("username")%></li>
</ul>
</p>

<form method="POST"
action="process-payment.jsp?activityId=<%=request.getParameter("activityId")%>
&redirect=payment-status.jsp">
  <p>Assuming you have completed all payment details, now let's submit this
form to process-payment.jsp</p>
  <input type="submit" value="Submit" />
</form>
```

process-payment.jsp

- Process the payment details submitted from credit-card-payment.jsp, and make a Ajax JavaScript call to RADS, to
  - Complete the “External: Process Payment” assignment
  - Update workflow variable – “**transaction_no**”
- Redirect to payment-status.jsp to display summary of the transaction.
- Once the “External: Process Payment” is completed, subsequent workflow activities will be continued and performed in RADS.

process-payment.jsp

```
<script type="text/javascript"
src="http://localhost:8080/rad/js/jquery/jquery-1.9.1.min.js"></script>
<script type="text/javascript"
src="http://localhost:8080/rad/js/json/util.js"></script>

<script type="text/javascript">
  var callback = {
    success : function(response){

AssignmentManager.completeAssignmentWithVariable("http://localhost:8080/
jw",
        getUrlParam("activityId"),
        "var_transaction_no =TN00001",
        escape(getUrlParam("redirect")));
    }
  }
  AssignmentManager.login("http://localhost:8080/rad", "admin", "admin",
callback);
</script>
```

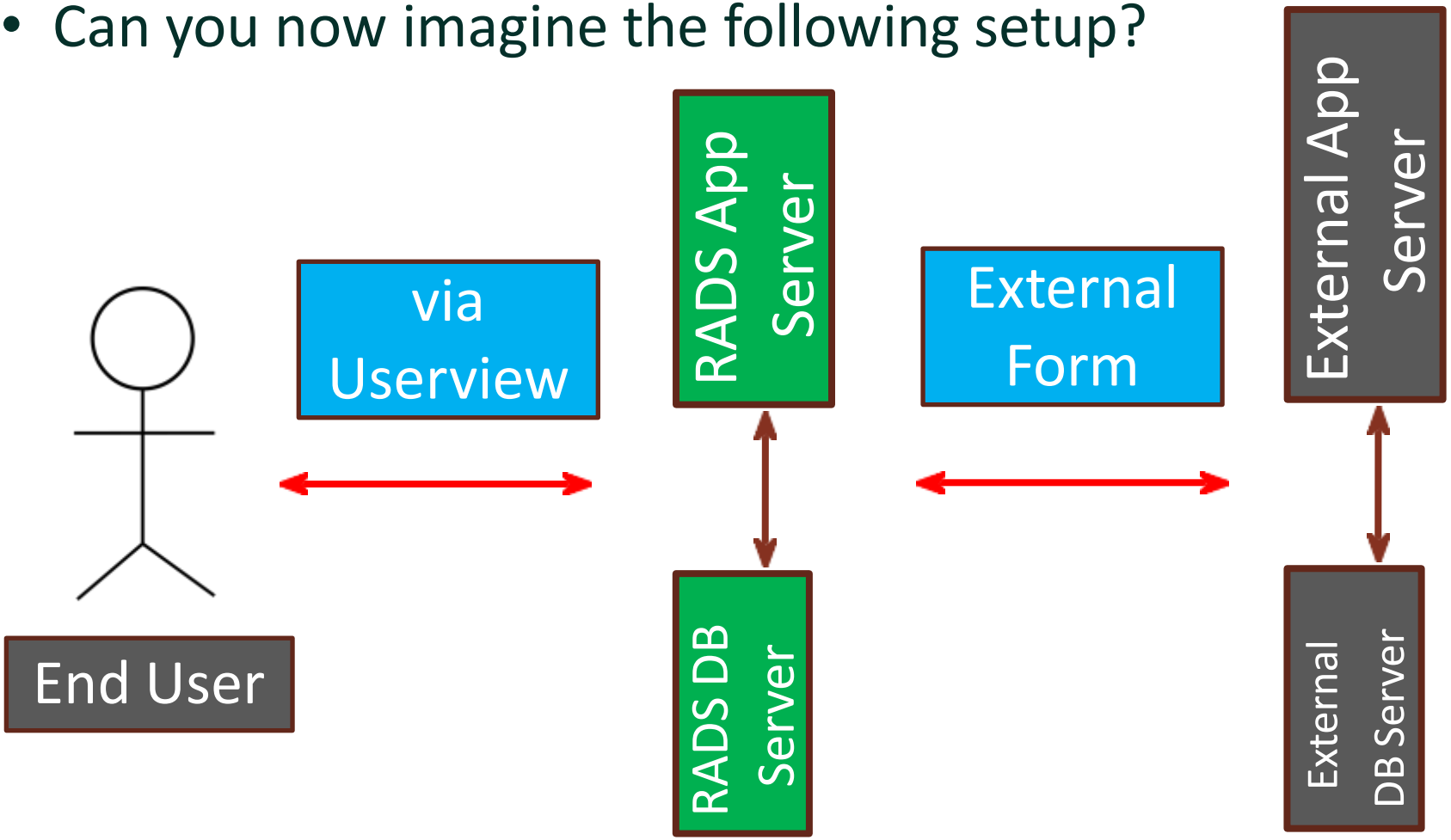
payment-status.jsp

- Display to user synchronously without interrupt subsequent workflow activities.

```
<h1>Payment Status</h1>

<p>Yeh~~~ payment transaction is completed.</p>
```

Setup Suggestion

- Can you now imagine the following setup?



Chapter Review

- Understand and being able to use External Form effectively.

Chapter 11

Using API Builder

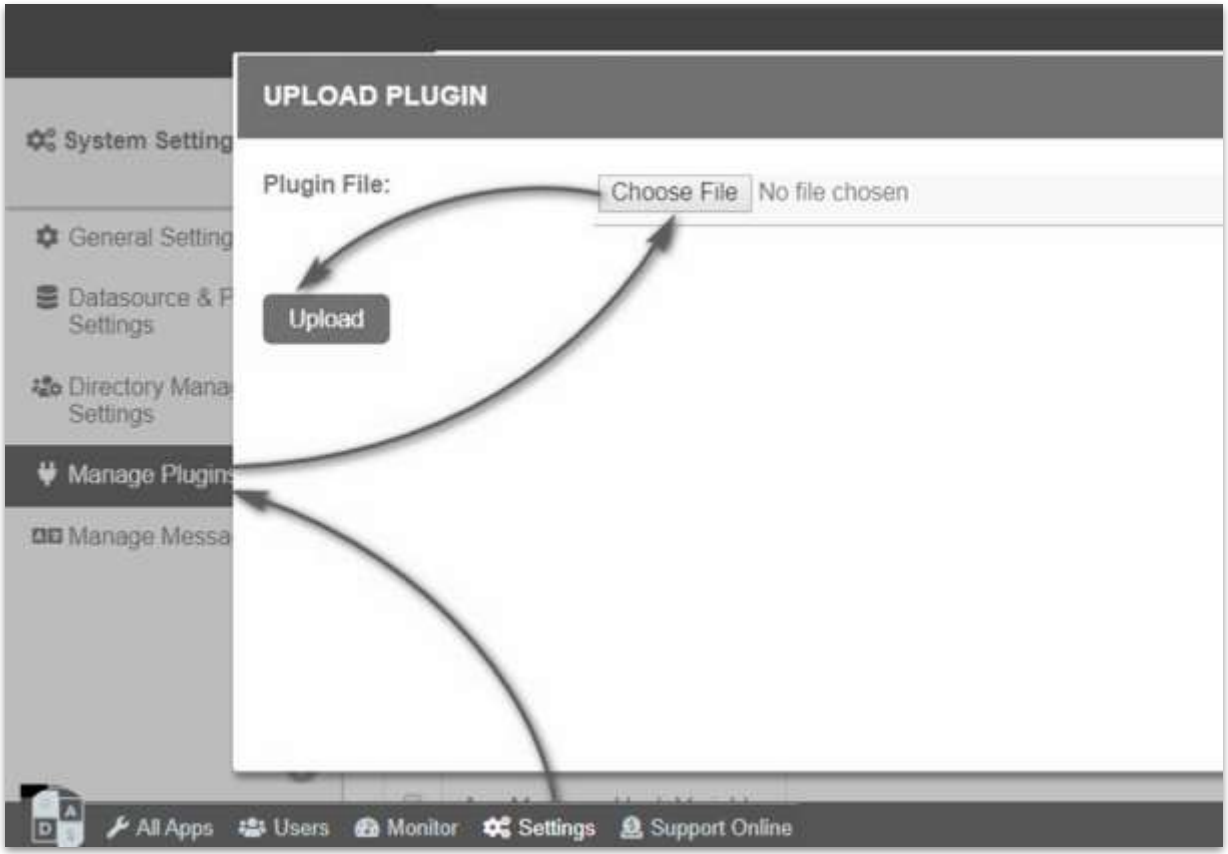
API Builder

- API Builder is a new type of plugin called Custom Builder.
- Allows one to create your own customized JSON APIs for RADS Apps.

Reference:
https://docs.rads.purwana.net/marketplace/API+Builder

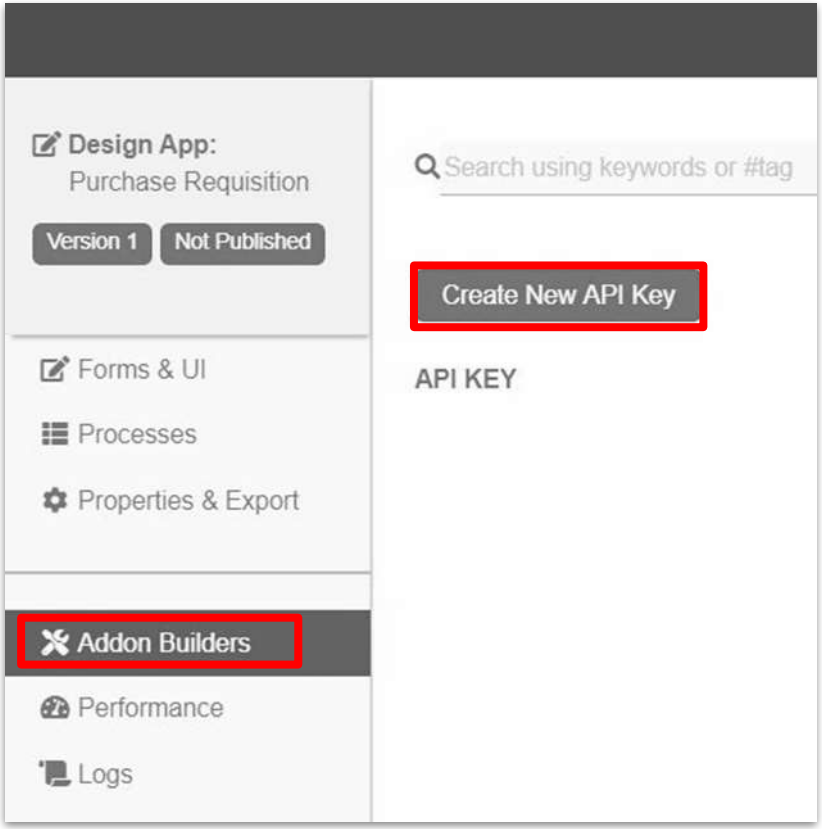
Installing API Builder

- Download the plugin from RADS Marketplace and import it into your RADS platform via Manage Plugin.



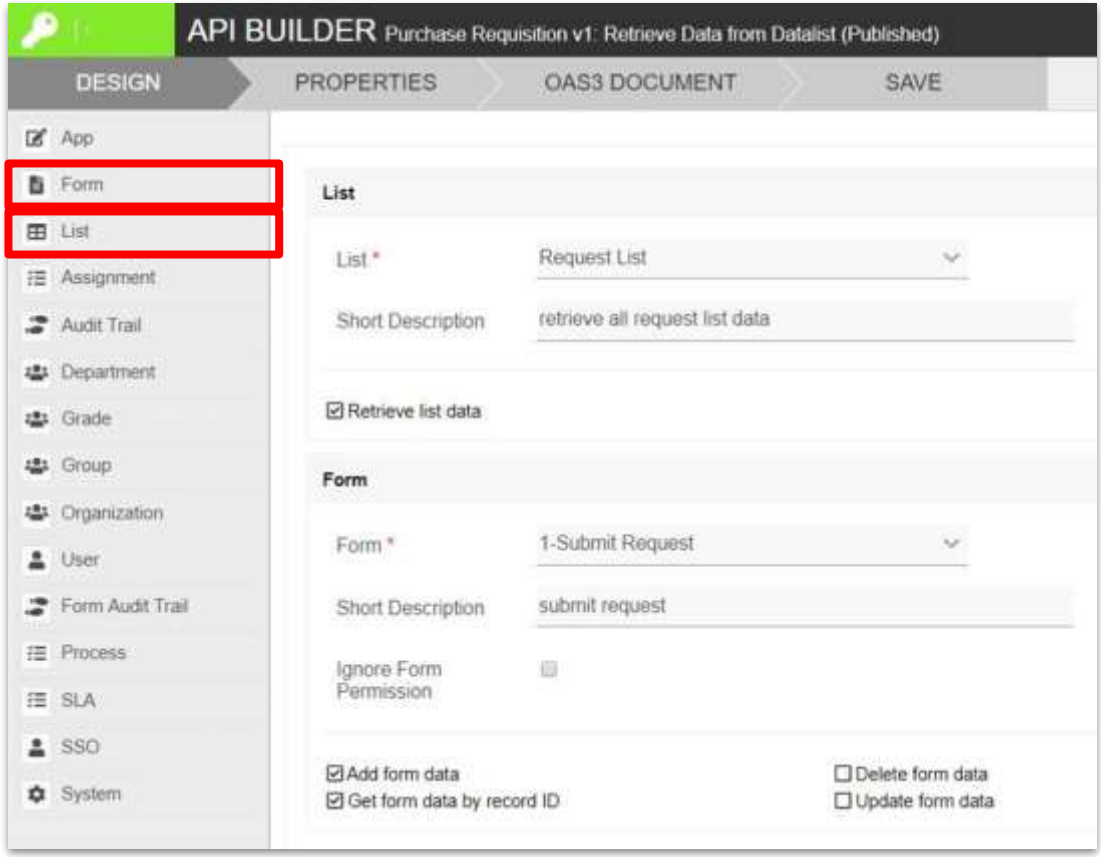
Exercise

- Continue to use the app from the previous chapters.
- Click on the new menu Addon Builders and click on Create new API Key.



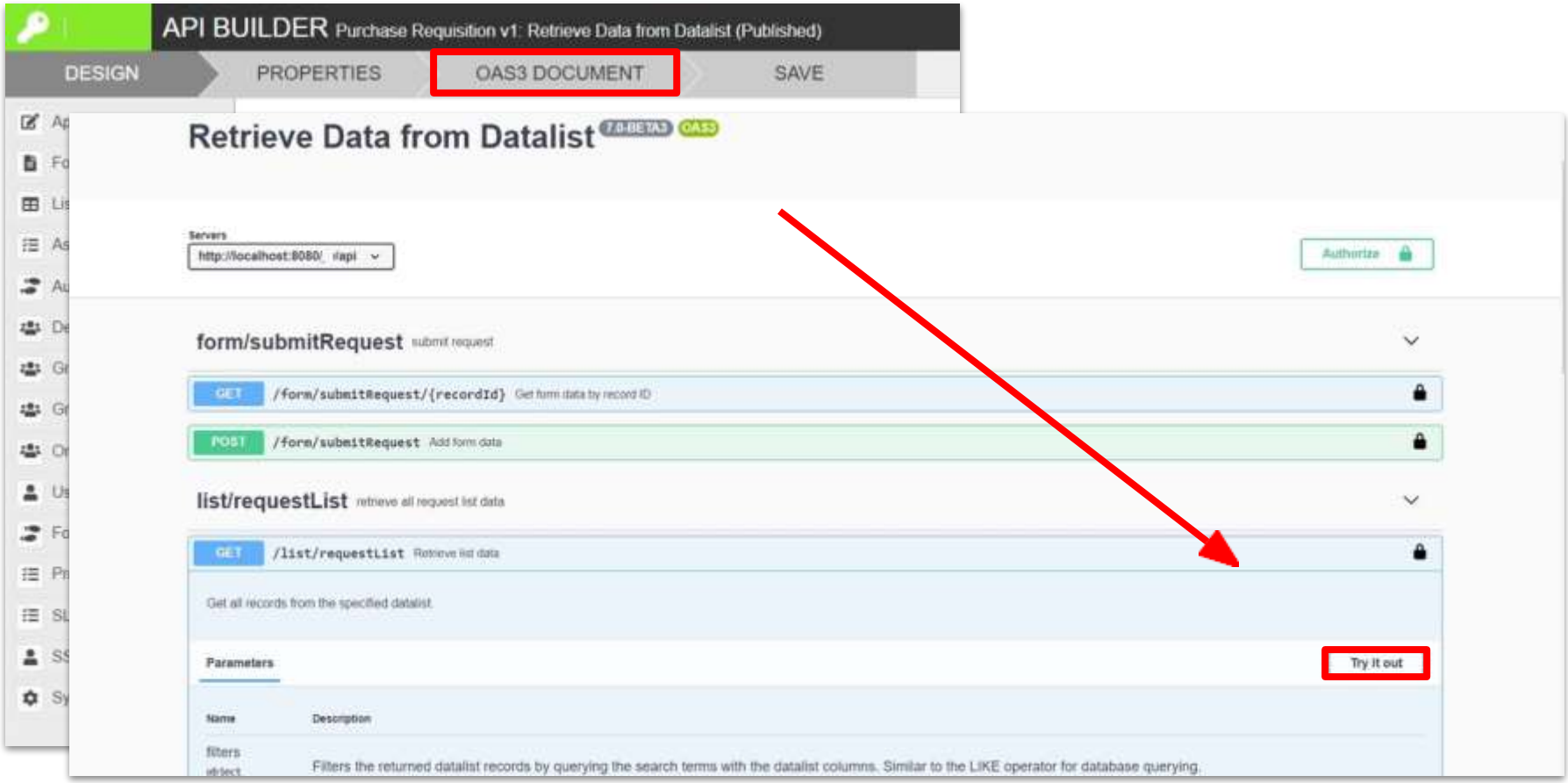
Exercise

- Try retrieving
  - all Request data from the list,
  - add form data to Submit Request and,
  - retrieve form data by record ID.



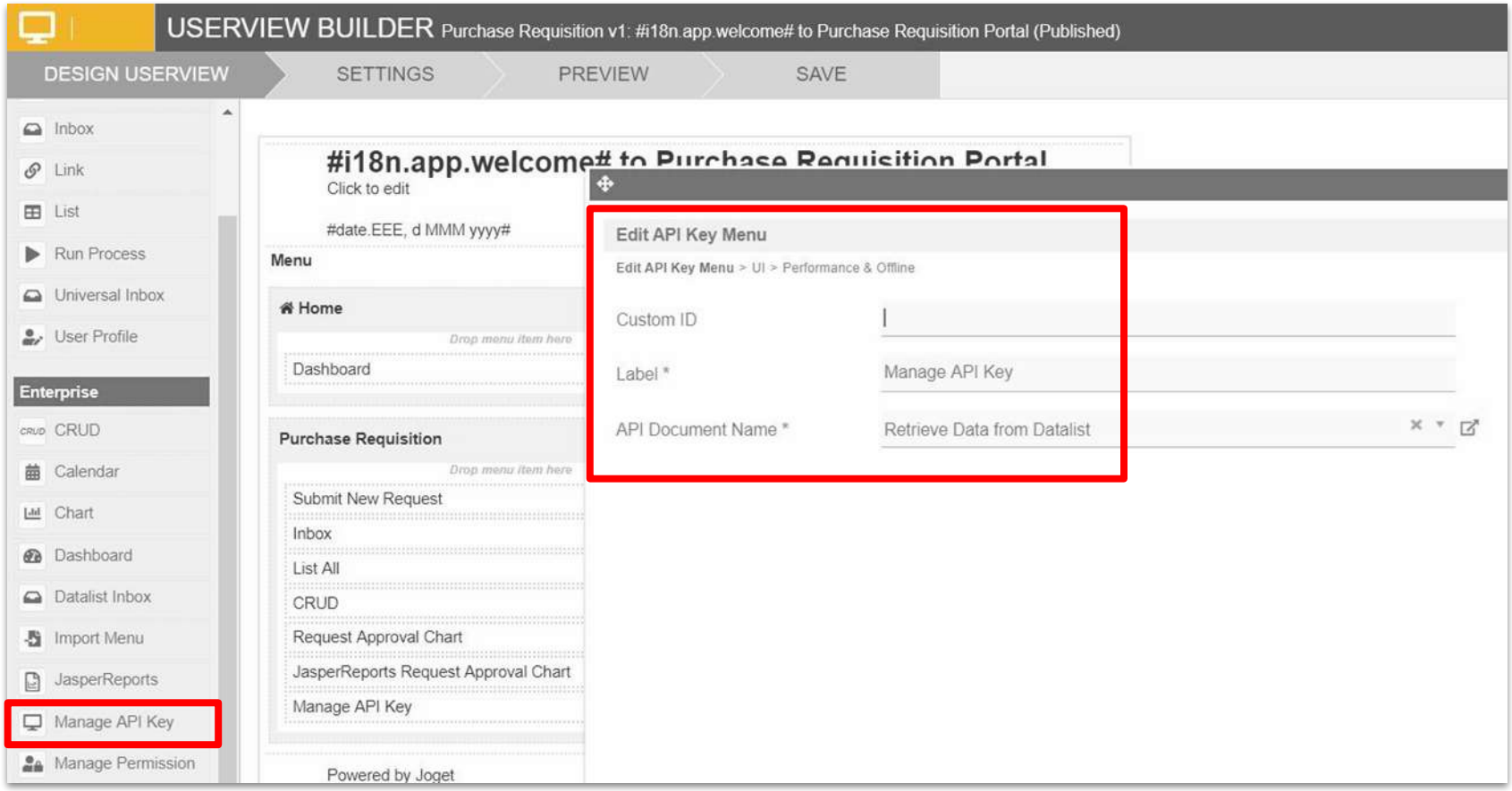
Generate OAS3 Document

- Click on OAS3 Document.
- Click on an API method and test it out.



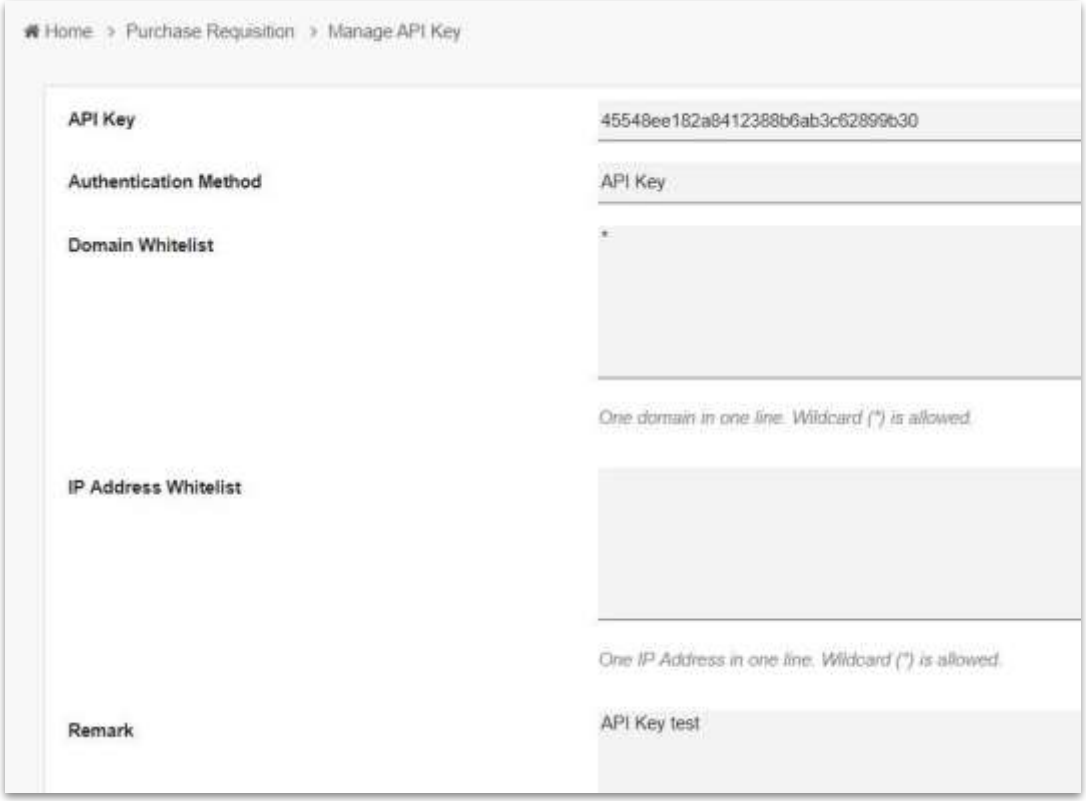
Manage API Key

- Userview element to control access to API methods created with API Builder.



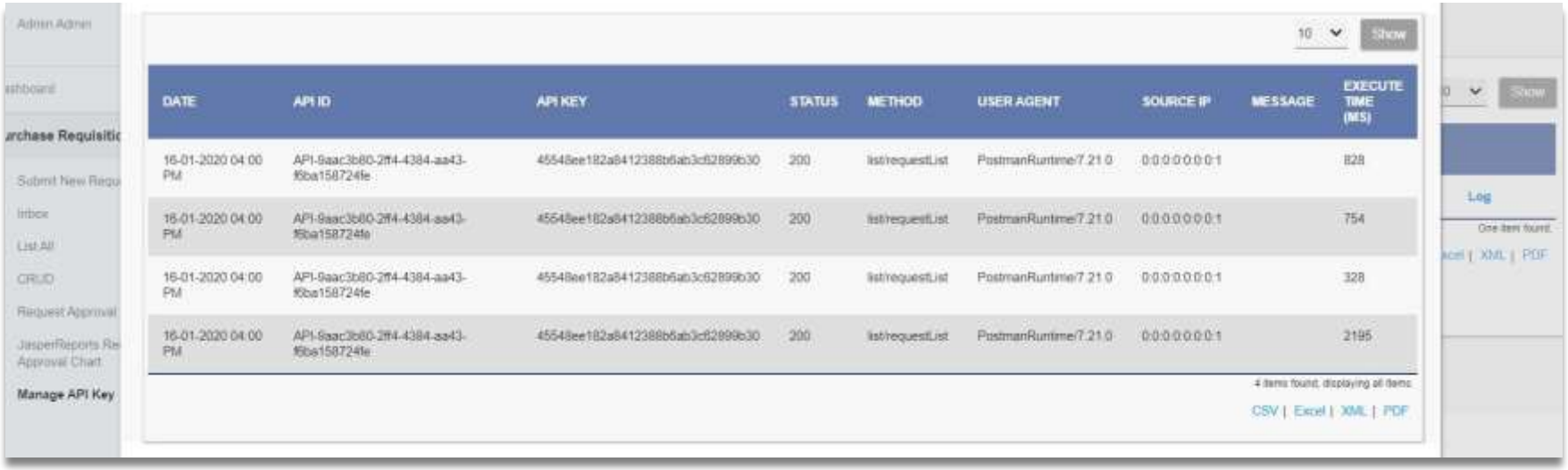
Exercise

- Create an API key.
- Try calling the API methods with and without the API key.



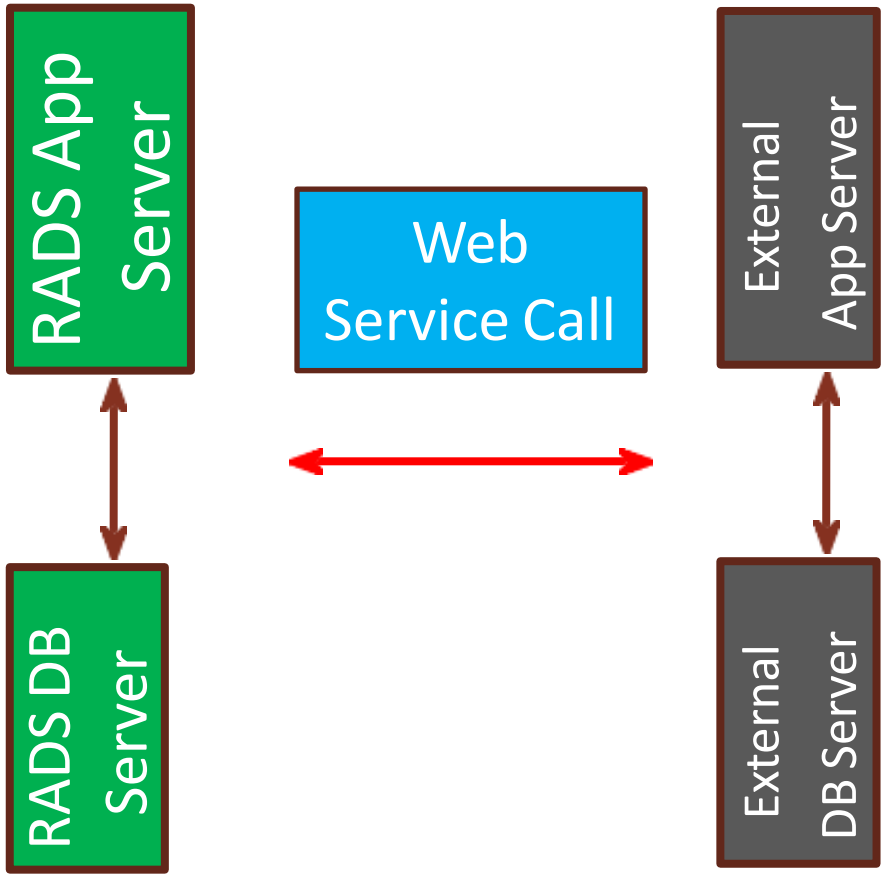
Manage API Key Log

- There’s also a log with details on each attempt.



Setup Suggestion

- Can you now imagine the following setup?



Chapter Review

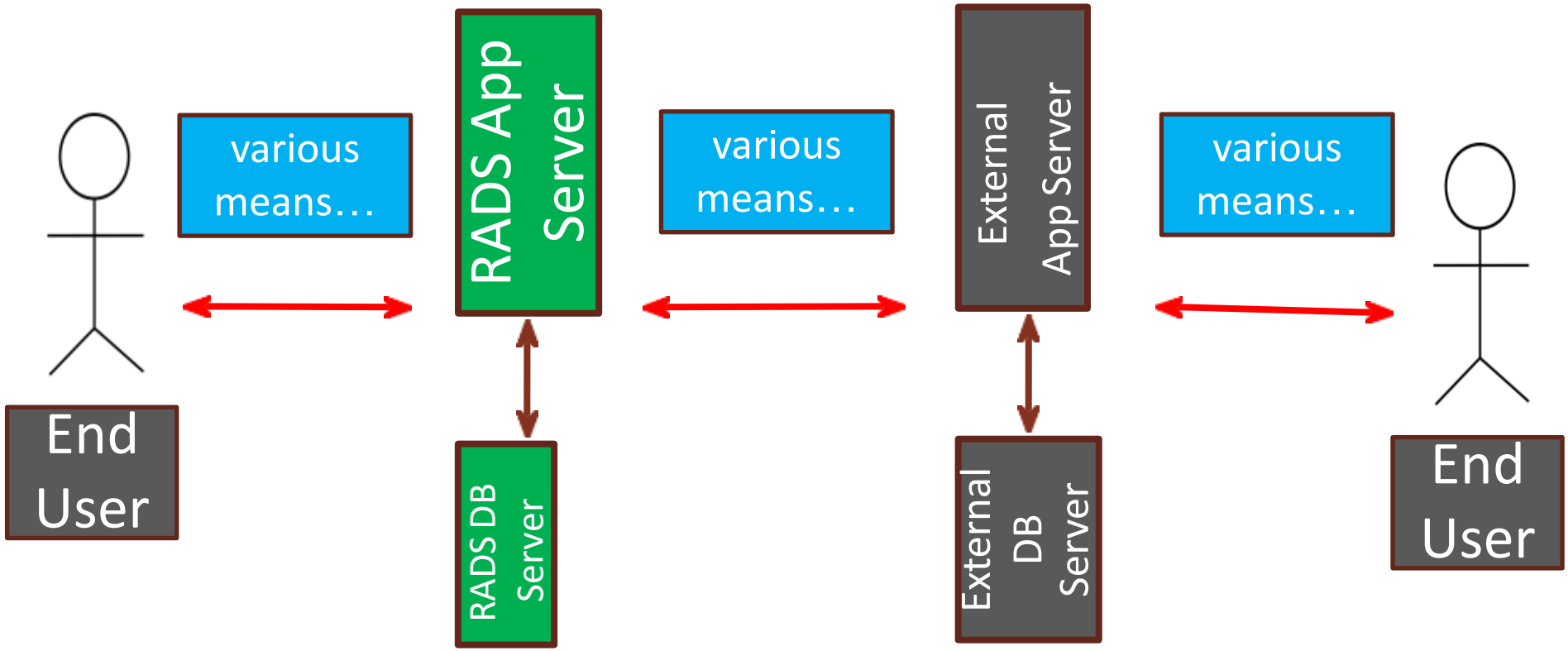
- Understand and being able to use API Builder effectively.

Module Review

1. Introduction
2. JSON API
3. JSON API Authentication
4. JavaScript API
5. Single Sign On (SSO)
6. Embedding Task Inbox into External System
7. Embedding Userview Page in an iFrame
8. JSON Tool
9. SOAP Tool
10. Integrating with External Form
11. Using API Builder

Module Review

- Can you now picture various setup scenarios to integrate RADS with your external app?



Stay Connected with RADS

- rads.purwana.net
- https://github.com/radservice/rad-community