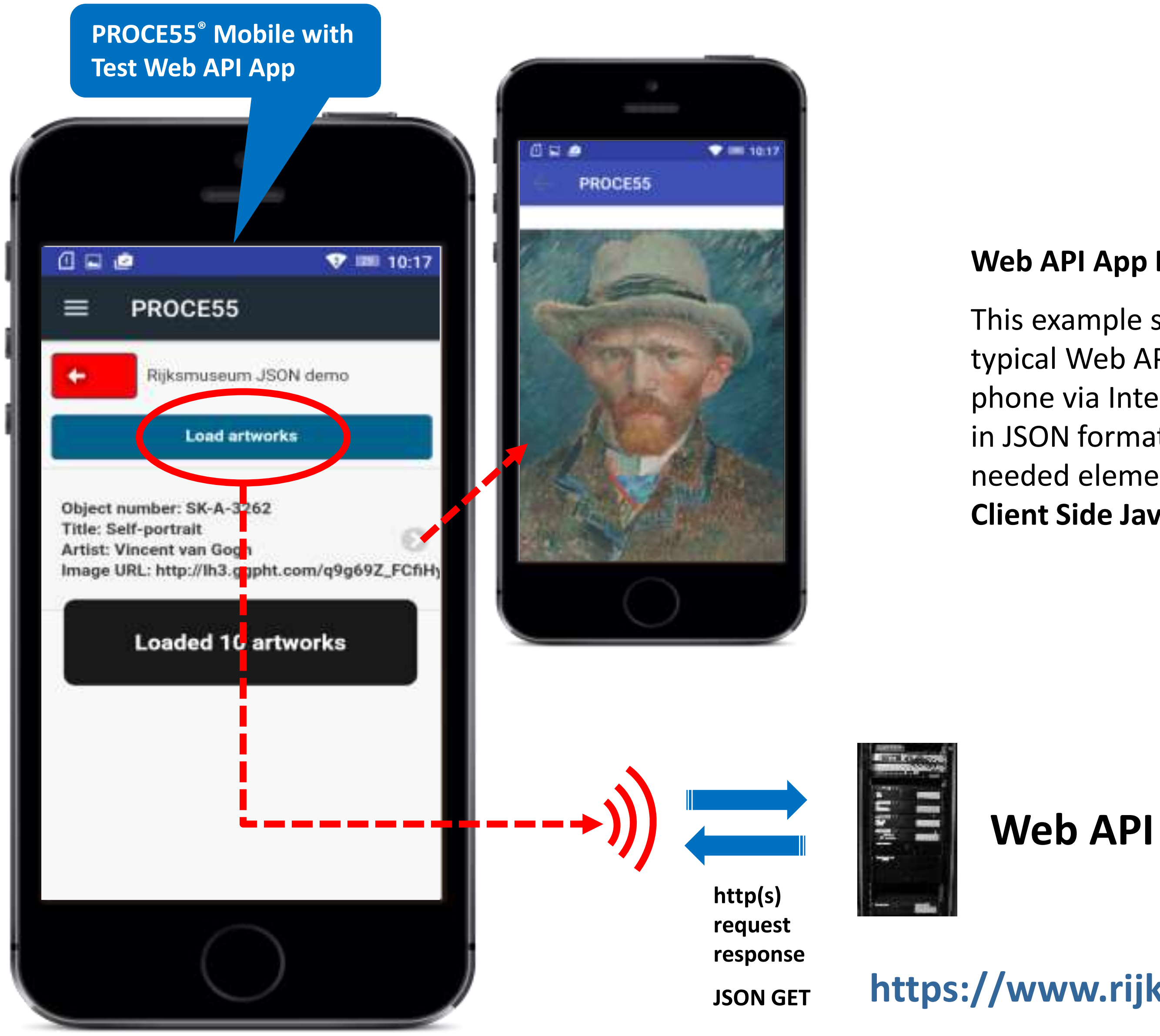
PROCE55® Mobile: Web API App



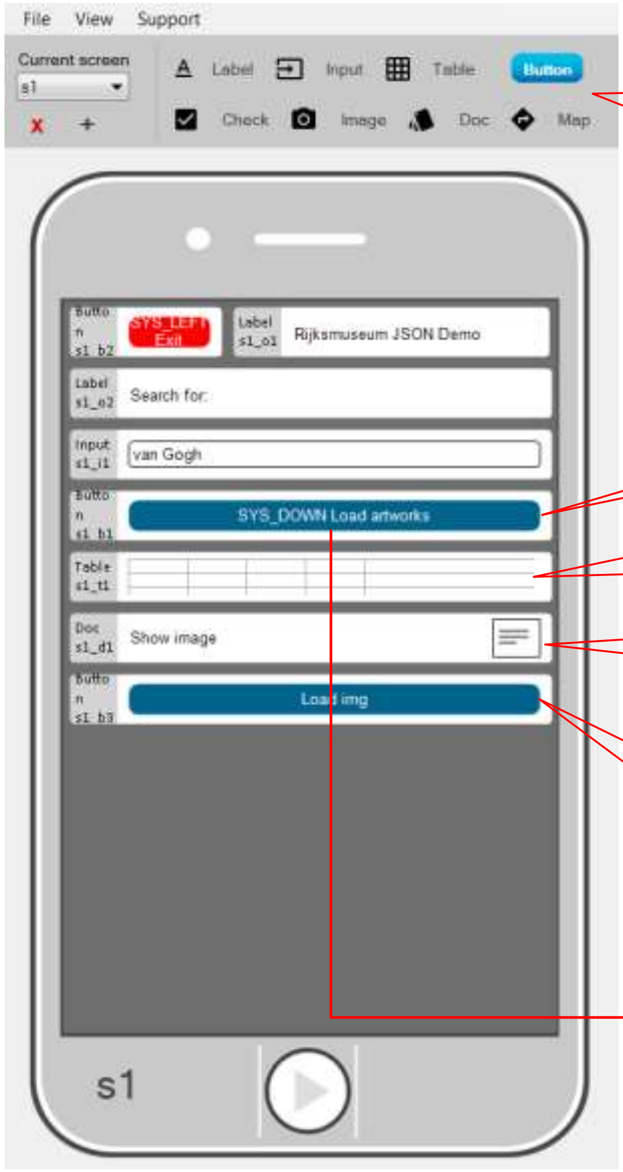
Object number: SK-A-3262
Title: Self-portrait
Artist: Vincent van Gogh
Image URL: http://lh3.ggpht.com/q9g69Z_FCfiHy

Web API App Example

This example shows how to access a typical Web API using your mobile phone via Internet. The returned data is in JSON format and for extraction of needed elements (text and picture) the **Client Side Java Script** function is used.

https://www.rijksmuseum.nl/api/...

PROCE55® Mobile: How to create Web API App – Step 1



pre-defined visual controls

Search Button

Table for results

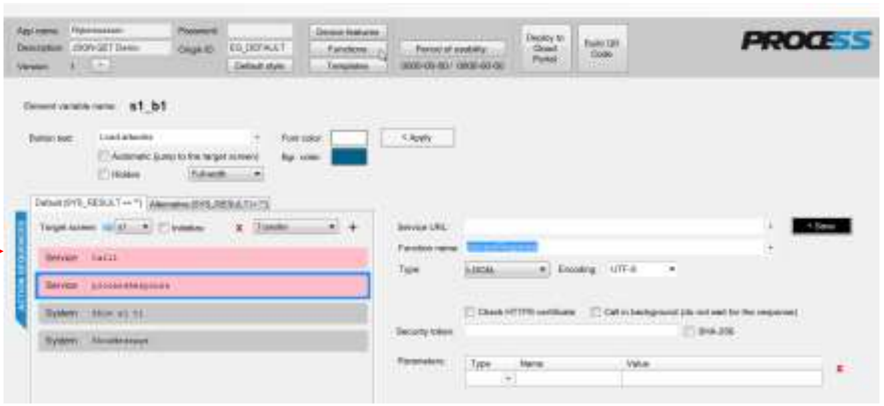
Image Viewer

Buttons for trigerring actions and navigation

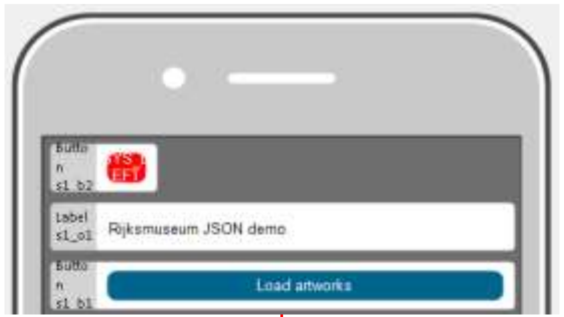
associate Button with Web Service call and subsequent processing using Java Script function

Design User Interface

The visual User Interface can be designed with the PROCE55® Modeler using the pre-defined **visual controls** which can be placed on the screens of the mobile App. **Buttons** are used to navigate among the screens. Each button can be associated with a **sequence of pre-defined actions**, the flow of actions is controlled by **conditions**. See our Guides for details.



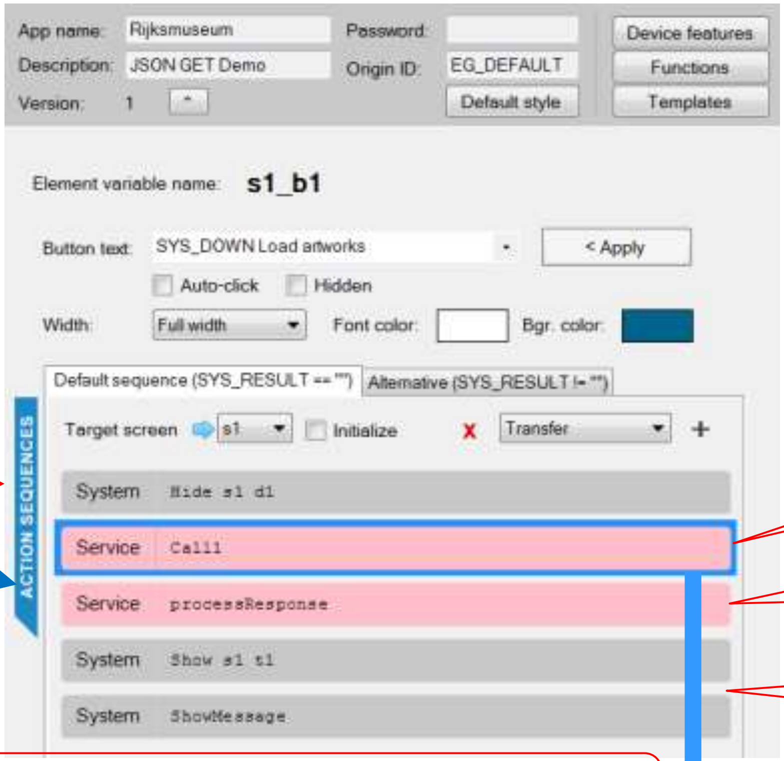
PROCE55® Mobile: How to create Web API App – Step 2



Label s1_l1: Rijksmuseum JSON demo

Button is associated with a sequence of actions

Insert Actions and select them to define their attributes. Now the Action Service is selected to define the Web Service interface



App name: Rijksmuseum
Description: JSON GET Demo
Version: 1
Password:
Origin ID: EG_DEFAULT

Element variable name: **s1_b1**

Button text: SYS_DOWN Load artworks
Auto-click
Hidden
Width: Full width
Font color:
Bgr. color:

Target screen: s1
Initialize
Transfer

```
System   Hide s1 d1
Service  Call1
Service  processResponse
System   Show s1 s1
System   ShowMessage
```

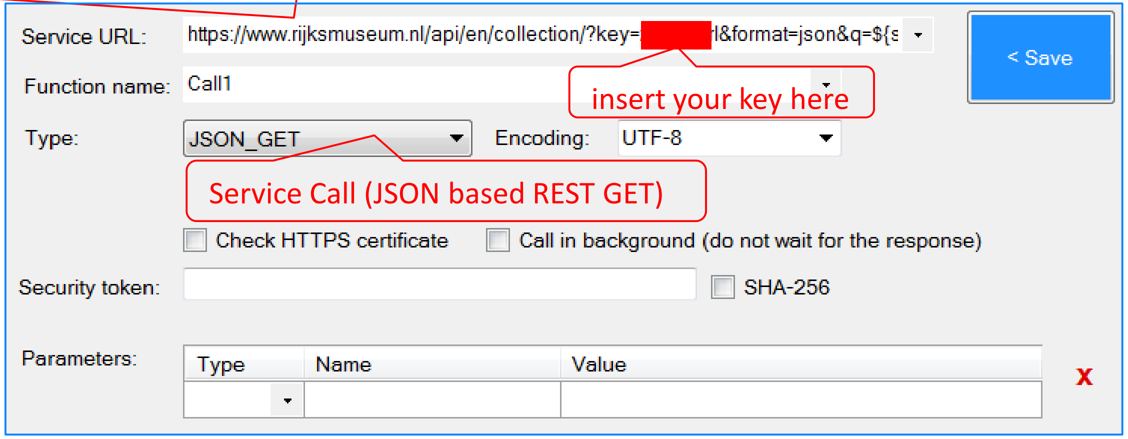
Service call

Function for processing of response[1]

Activate visual controls

Define the Application Logic (Client side) the application logic of the app can be defined as a sequence of predefined actions (building blocks) which encapsulate typical functionality including the conditions.

Insert Web API endpoint according to the documentation:
https://www.rijksmuseum.nl/api/en/collection/?key=<key>&format=json&q=<search query>



Service URL: https://www.rijksmuseum.nl/api/en/collection/?key=[redacted]l&format=json&q=${s
Function name: Call1
Type: JSON_GET
Encoding: UTF-8
Check HTTPS certificate
Call in background (do not wait for the response)
Security token:
SHA-256

Parameters:

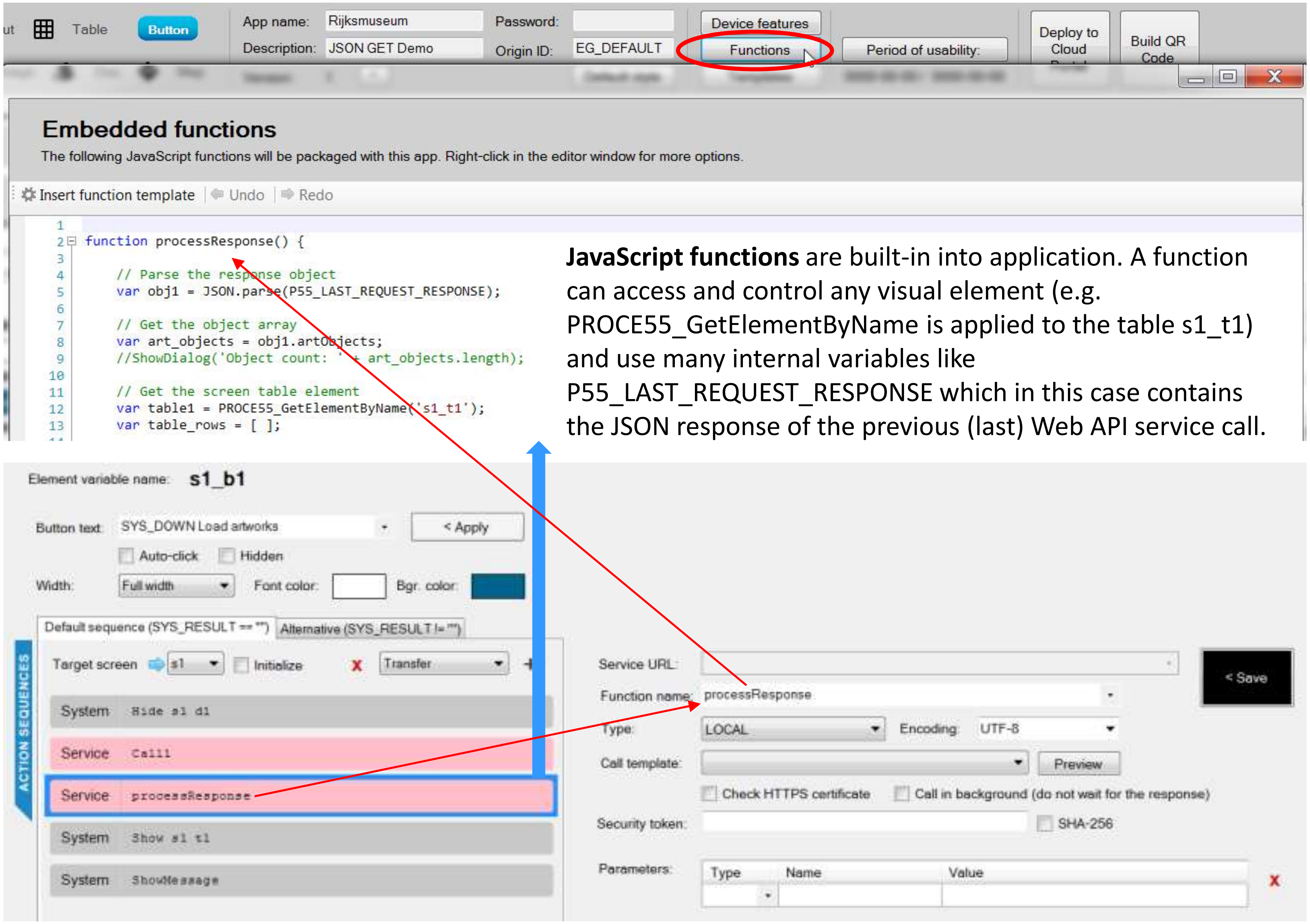
Type	Name	Value

insert your key here

Service Call (JSON based REST GET)

[1]In case of more complex logic you can use the **JavaScript functions** which can be built-in into application. In this example we are using the function to parse the more complex JSON response. Using functions you can access and control any visual element and many internal variables.

PROCE55® Mobile: How to create Web API App – Step 3



App name: Rijksmuseum
Description: JSON GET Demo
Password:
Origin ID: EG_DEFAULT

Embedded functions

The following JavaScript functions will be packaged with this app. Right-click in the editor window for more options.

```
function processResponse() {

    // Parse the response object
    var obj1 = JSON.parse(P55_LAST_REQUEST_RESPONSE);

    // Get the object array
    var art_objects = obj1.artObjects;
    //ShowDialog('Object count: ' + art_objects.length);

    // Get the screen table element
    var table1 = PROCE55_GetElementByName('s1_t1');
    var table_rows = [ ];
```

JavaScript functions are built-in into application. A function can access and control any visual element (e.g. PROCE55_GetElementByName is applied to the table s1_t1) and use many internal variables like P55_LAST_REQUEST_RESPONSE which in this case contains the JSON response of the previous (last) Web API service call.

Element variable name: **s1_b1**

Button text: SYS_DOWN Load artworks
Auto-click ☐ Hidden ☐
Width: Full width
Font color: Bgr. color:

< Apply
ACTION SEQUENCES
Default sequence (SYS_RESULT == "")
Alternative (SYS_RESULT != "")
< Save
Preview

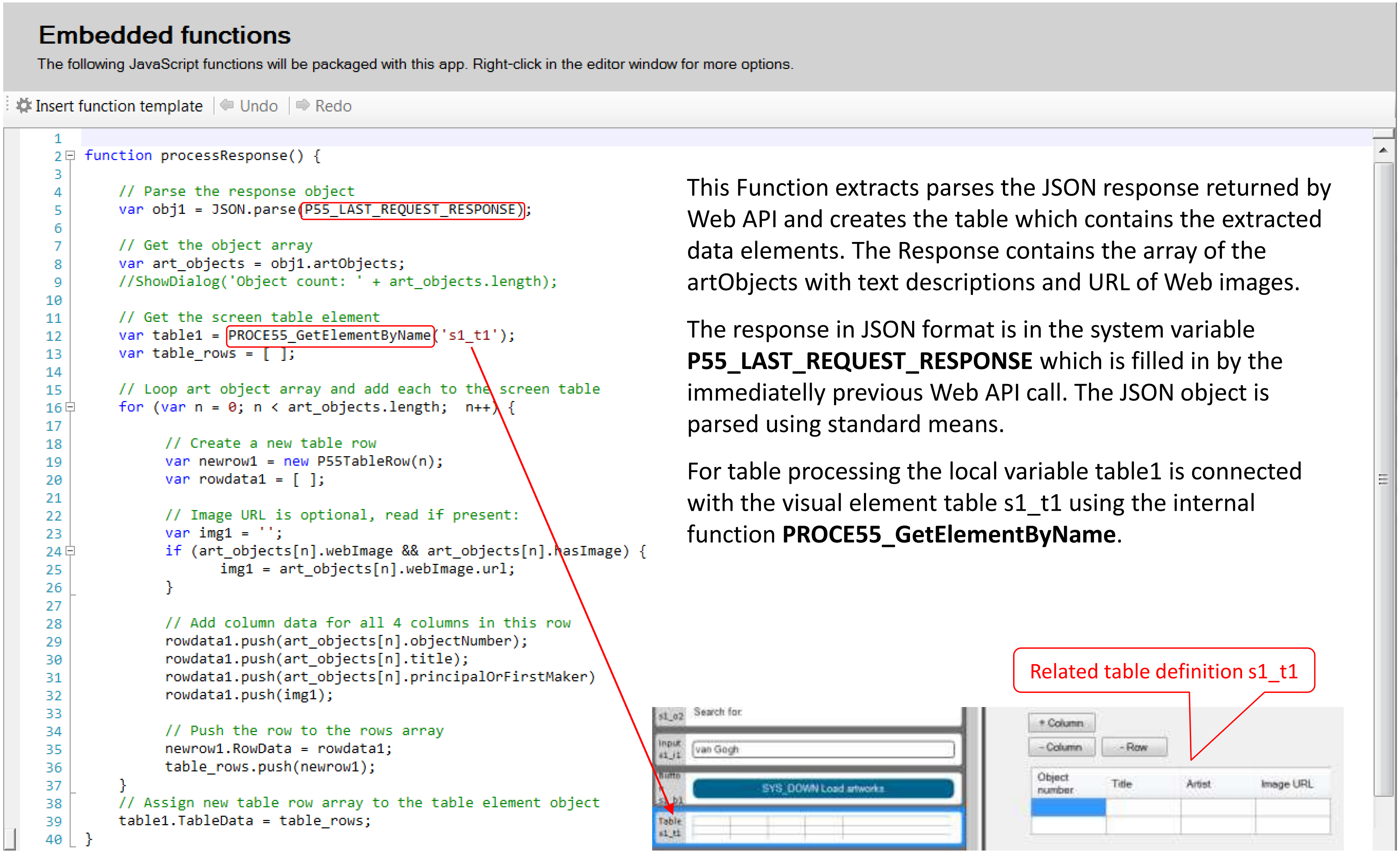
Target screen: s1 ☐ Initialize Transfer

System	Hide s1 d1
Service	Call1
Service	processResponse
System	Show s1 t1
System	ShowMessage

Service URL:
Function name: processResponse
Type: LOCAL Encoding: UTF-8
Call template:
☐ Check HTTPS certificate ☐ Call in background (do not wait for the response)
Security token: ☐ SHA-256
Parameters:

Type	Name	Value

PROCE55® Mobile: How to create Web API App – Function Code



Embedded functions

The following JavaScript functions will be packaged with this app. Right-click in the editor window for more options.

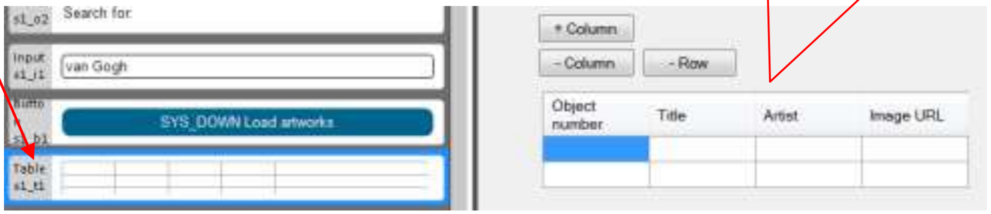
```
function processResponse() {

    // Parse the response object
    var obj1 = JSON.parse(P55_LAST_REQUEST_RESPONSE);

    // Get the object array
    var art_objects = obj1.artObjects;
    //ShowDialog('Object count: ' + art_objects.length);

    // Get the screen table element
    var table1 = PROCE55_GetElementByName('s1_t1');
    var table_rows = [ ];

    // Loop art object array and add each to the screen table
    for (var n = 0; n < art_objects.length;  n++) {

        // Create a new table row
        var newrow1 = new P55TableRow(n);
        var rowdata1 = [ ];

        // Image URL is optional, read if present:
        var img1 = '';
        if (art_objects[n].webImage && art_objects[n].hasImage) {
              img1 = art_objects[n].webImage.url;
        }

        // Add column data for all 4 columns in this row
        rowdata1.push(art_objects[n].objectNumber);
        rowdata1.push(art_objects[n].title);
        rowdata1.push(art_objects[n].principalOrFirstMaker)
        rowdata1.push(img1);

        // Push the row to the rows array
        newrow1.RowData = rowdata1;
        table_rows.push(newrow1);
    }
    // Assign new table row array to the table element object
    table1.TableData = table_rows;
}
```

This Function extracts parses the JSON response returned by Web API and creates the table which contains the extracted data elements. The Response contains the array of the artObjects with text descriptions and URL of Web images.

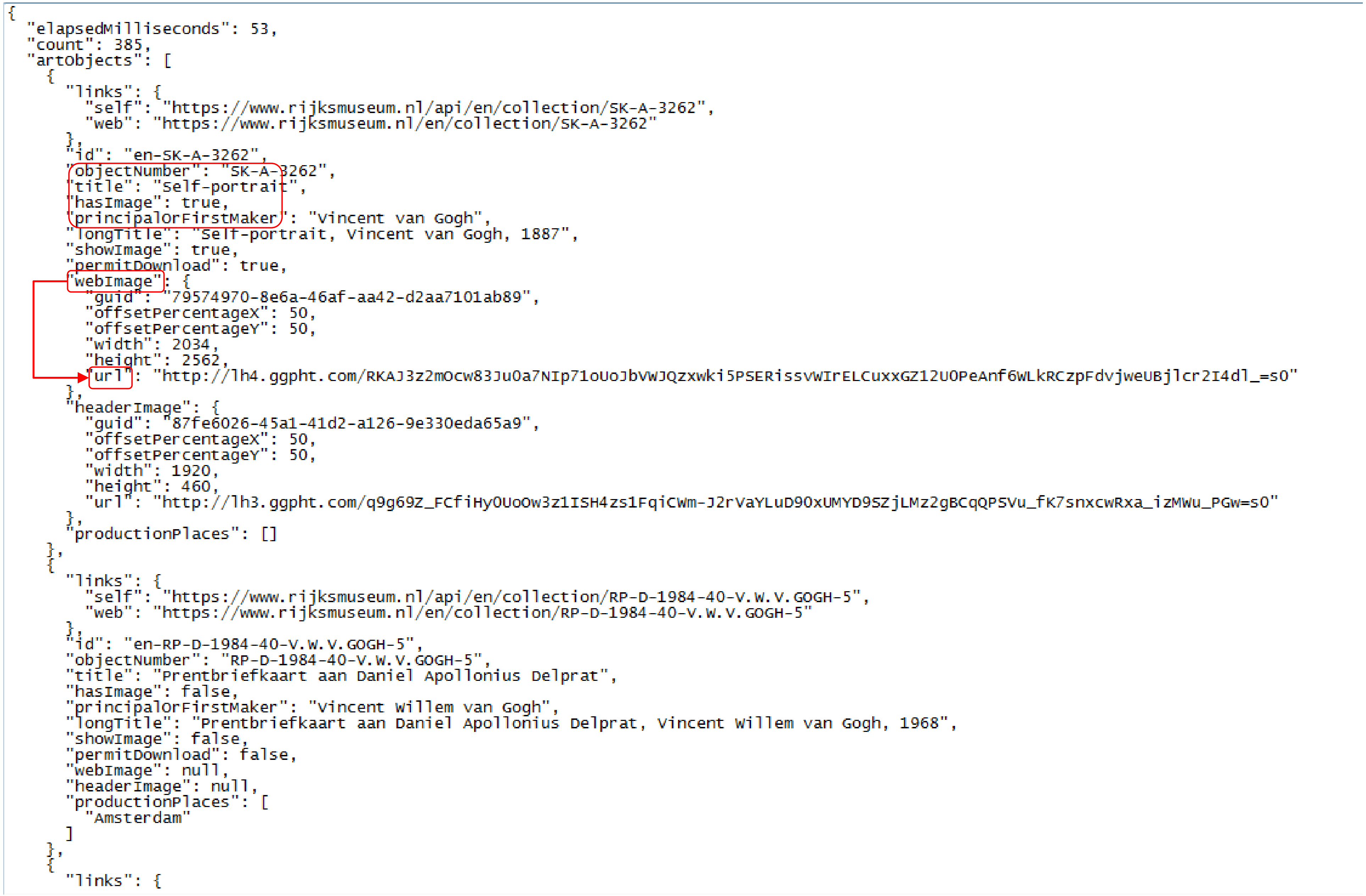
The response in JSON format is in the system variable **P55_LAST_REQUEST_RESPONSE** which is filled in by the immediatelly previous Web API call. The JSON object is parsed using standard means.

For table processing the local variable table1 is connected with the visual element table s1_t1 using the internal function **PROCE55_GetElementByName**.

Related table definition s1_t1



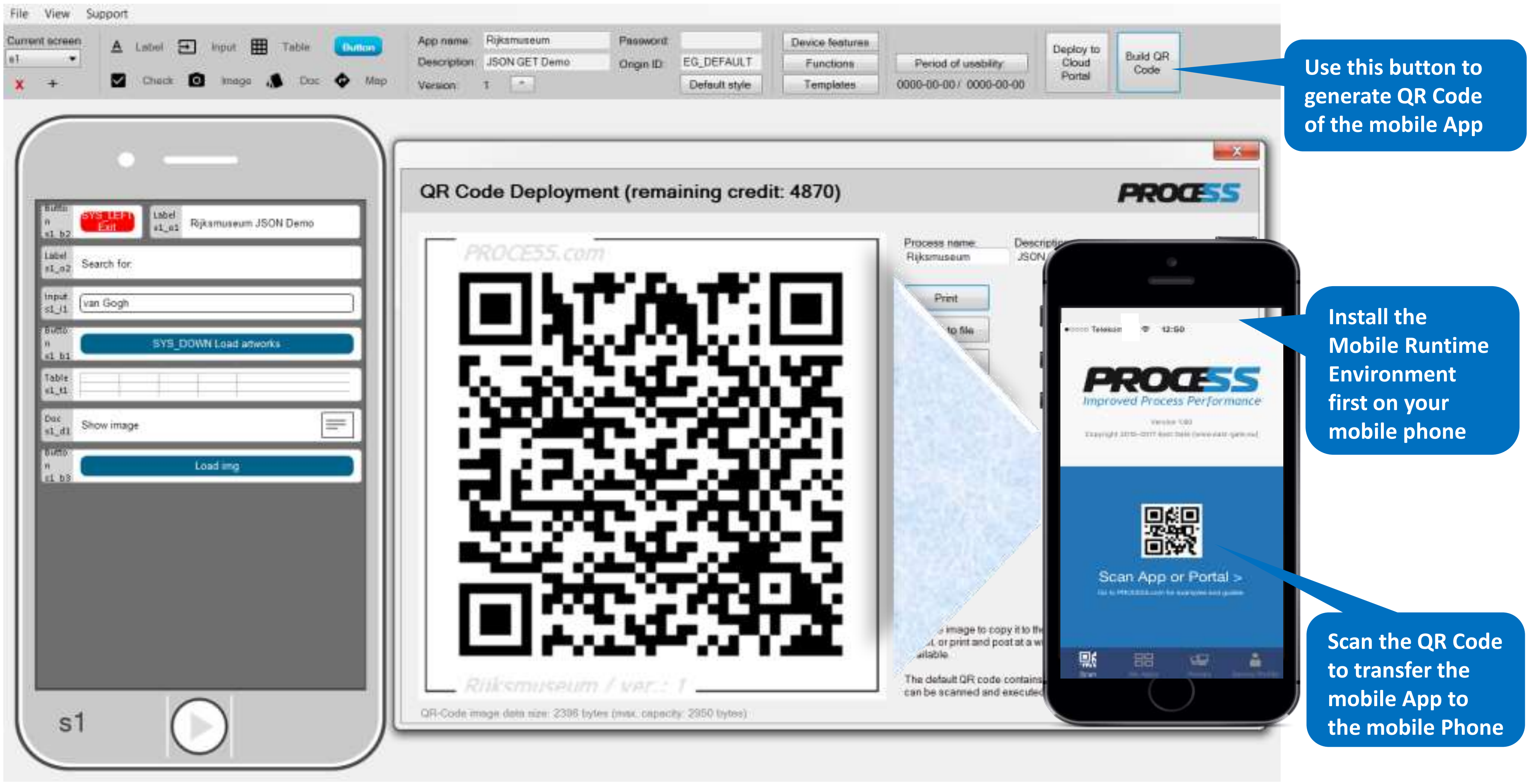
PROCE55® Mobile: How to create Web API App – JSON Fragment



```
{
  "elapsedMilliseconds": 53,
  "count": 385,
  "artObjects": [
    {
      "links": {
        "self": "https://www.rijksmuseum.nl/api/en/collection/SK-A-3262",
        "web": "https://www.rijksmuseum.nl/en/collection/SK-A-3262"
      },
      "id": "en-SK-A-3262",
      "objectNumber": "SK-A-3262",
      "title": "Self-portrait",
      "hasImage": true,
      "principalOrFirstMaker": "Vincent van Gogh",
      "longTitle": "Self-portrait, Vincent van Gogh, 1887",
      "showImage": true,
      "permitDownload": true,
      "webImage": {
        "guid": "79574970-8e6a-46af-aa42-d2aa7101ab89",
        "offsetPercentageX": 50,
        "offsetPercentageY": 50,
        "width": 2034,
        "height": 2562,
        "url": "http://lh4.ggpht.com/RKAJ3z2mOcw83Ju0a7NIp71oUoJbVWJQzxwki5PSERissVWIrELCuxxGZ12U0PeAnf6WLkRCzpFdvjweUBjlcr2I4dl_=s0"
      },
      "headerImage": {
        "guid": "87fe6026-45a1-41d2-a126-9e330eda65a9",
        "offsetPercentageX": 50,
        "offsetPercentageY": 50,
        "width": 1920,
        "height": 460,
        "url": "http://lh3.ggpht.com/q9g69Z_FCfiHyOUoOw3z1ISH4zs1FqiCWm-J2rVaYLuD90xUMYD9SZjLMz2gBCqQPSVu_fK7snxcwRxa_izMWu_PGw=s0"
      },
      "productionPlaces": []
    },
    {
      "links": {
        "self": "https://www.rijksmuseum.nl/api/en/collection/RP-D-1984-40-V.W.V.GOGH-5",
        "web": "https://www.rijksmuseum.nl/en/collection/RP-D-1984-40-V.W.V.GOGH-5"
      },
      "id": "en-RP-D-1984-40-V.W.V.GOGH-5",
      "objectNumber": "RP-D-1984-40-V.W.V.GOGH-5",
      "title": "Prentbriefkaart aan Daniel Apollonius Delprat",
      "hasImage": false,
      "principalOrFirstMaker": "Vincent Willem van Gogh",
      "longTitle": "Prentbriefkaart aan Daniel Apollonius Delprat, Vincent Willem van Gogh, 1968",
      "showImage": false,
      "permitDownload": false,
      "webImage": null,
      "headerImage": null,
      "productionPlaces": [
        "Amsterdam"
      ]
    },
    {
      "links": {
```

PROCE55® Mobile: How to create Web API App – Step 4

Deployment of the mobile App with PROCE55® Modeler is very simple compared with the procedures used in the App Stores. You can deploy the mobile App directly from the PROCE55® Modeler using QR Code or via Portals for complex deployment scenarios. **No compilation** is needed, you deploy to **iOS, Android or Windows Mobile Runtime Environment** directly without any transformation.



App name: Rijksmuseum
Description: JSON GET Demo
Version: 1
Password:
Origin ID: EG_DEFAULT
0000-00-00 / 0000-00-00

Label s1_o1: Rijksmuseum JSON Demo
Input s1_i1: van Gogh

QR Code Deployment (remaining credit: 4870)

Process name: Rijksmuseum
Description: JSON

PROCE55.com

Rijksmuseum / ver.: 1

QR-Code image data size: 2398 bytes (max. capacity: 2950 bytes)

The default QR code contains … can be scanned and executed …