DELIVERABLE

Project Acronym: F4W

Grant Agreement nbr: 000000

Project Title: FACTS4WORKERS

RESTdesc demo input/output

Revision:

Authors:

Joachim Van Herwegen (iMinds – Ugent – Multimedia Lab)

REVISION HISTORY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Revision | Date | Author | Organisation | Description |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

[1 Introduction 4](#_Toc418682620)

[2 API flow 4](#_Toc418682621)

[2.1 Worker 🡪 RESTdesc: Clean start 4](#_Toc418682622)

[2.2 RESTdesc 🡪 Worker: Ask the worker initial start data 4](#_Toc418682623)

[2.3 Worker 🡪 RESTdesc: Send initial worker response 4](#_Toc418682624)

[2.4 RESTdesc: Next step: Call API 1 5](#_Toc418682625)

[2.5 RESTdesc 🡪 API 1: Get calibration information 5](#_Toc418682626)

[2.6 API 1 🡪 RESTdesc: Send calibration information 6](#_Toc418682627)

[2.7 RESTdesc 🡪 Worker: Ask measurements 6](#_Toc418682628)

[2.8 Worker 🡪 RESTdesc: Send measurements 6](#_Toc418682629)

[2.9 RESTdesc: Next step: Call API 2 7](#_Toc418682630)

[2.10 RESTdesc 🡪 API 2: POST measurements 7](#_Toc418682631)

[2.11 API 2 🡪 RESTdesc: POST results 7](#_Toc418682632)

# Introduction

This document describes the complete input/output flow generated by the demo use case, giving developers a clearer picture of what is required to make use of the RESTdesc server.

# API flow

## Worker 🡪 RESTdesc: Clean start

{}

This is the very first call to the server so an empty POST call to demo/next is sufficient. In the future when there are multiple use cases it will be necessary to identify which goals needs to be completed. This could be solved by posting a JSON object containing a ‘goal’ field that contains the goal ID, or the contents of the goal file depending on where these get stored.

## RESTdesc 🡪 Worker: Ask the worker initial start data

{

"http:body": {

"message": "Please input the starting information. 'id' corresponds to the calibration ID."

},

"http:resp": {

"http:body": {

"id": "\_:b1\_sk4\_2"

}

},

"http:methodName": "GET",

"http:requestURI": "http://askTheWorker/start",

"output": "...",

"data": [...]

}

Since the first ‘API’ call is a call to the worker, RESTdesc immediately returns a response. “http:body” always corresponds to the input to the ‘API’, so in this case we simply included a message for the worker. “http:resp”:”http:body” corresponds to the expected response. The worker input should be sent in the same JSON format.

## Worker 🡪 RESTdesc: Send initial worker response

{

"json": { "id": 1 },

"eye": {

"http:body": {

"message": "Please input the starting information. 'id' corresponds to the calibration ID."

},

"http:resp": {

"http:body": {

"id": "\_:b1\_sk4\_2"

}

},

"http:methodName": "GET",

"http:requestURI": "http://askTheWorker/start"

"output": "...",

"data": [...],

}

}

A response to the RESTdesc server should always be a JSON object containing 2 fields. The first field, ‘json’, should contain the worker input in the correct JSON format. The ‘eye’ field should contain the RESTdesc message this is a response to. The server will use this to make sure all input values get assigned to the correct variables.

## RESTdesc: Next step: Call API 1

{

"http:resp": {

"http:body": {

"machine\_parameters": [

"\_:b2\_sk9\_2",

"\_:b2\_sk10\_2",

"\_:b2\_sk11\_2",

"\_:b2\_sk12\_2"

],

"tolerances": [

{ "min": "\_:b2\_sk5\_2", "max": "\_:b2\_sk6\_2" },

{ "min": "\_:b2\_sk7\_2", "max": "\_:b2\_sk8\_2" }

],

"part\_number": "\_:b2\_sk4\_2",

"operator": "\_:b2\_sk3\_2"

}

},

"http:methodName": "GET",

"data": [...],

"http:requestURI": "http://pacific-shore-4503.herokuapp.com/calibrations/1"

}

This is the response generated after receiving the previous input. Since the request URI is not a worker call, this response does not need to be sent anywhere. The response JSON here corresponds to the expected output format of the API. Since this is a GET request, there is no body. Currently our system automatically calls the next API if no user input is required. It is always possible to change this and just send a response again that indicates which API needs to be called next should this be preferred.

## RESTdesc 🡪 API 1: Get calibration information

This is a simple GET call to http://pacific-shore-4503.herokuapp.com/calibrations/1 so no body is required.

## API 1 🡪 RESTdesc: Send calibration information

{

"id": 2,

"operator": "Gianni",

"part\_number": "123",

"machine\_parameters": [ 1200.25, 0.0024, 13.7, 270 ],

"tolerances": [

{ "min": 134.3, "max": 134.35 },

{ "min": 0.37, "max": 0.4 }

],

"geometrical\_dimension": [ 134.33, 0.38 ],

"result": "ok",

"suggested\_parameters": [ 1200.25, 0.0024, 13.7, 270 ]

}

This is the full response from the API. It actually contains too much information since it already has a result value for example. This is not a problem since RESTdesc only takes the fields that correspond to the mapping in “http:resp”:”http:body” previously generated in section 2.4. All other values get ignored.

## RESTdesc 🡪 Worker: Ask measurements

{

"http:body": {

"machine\_parameters": [ "1200.25", "0.0024", "13.7", "270" ],

"part\_number": "123",

"message": "Please measure a new part with the following settings."

},

"http:resp": {

"http:body": {

"geometrical\_dimension": [ "\_:b3\_sk4\_2", "\_:b3\_sk5\_2" ]

}

},

"http:methodName": "GET",

"http:requestURI": "http://askTheWorker/doMeasurement",

"output": "...",

"data": [...]

}

The next API call is a worker API, so this information gets sent back to the client again. The “http:body” here contains all the information the worker needs to do his measurements: the machine parameters and the part number. Again it is indicated in which format the client should respond.

## Worker 🡪 RESTdesc: Send measurements

{

"json": {

"geometrical\_dimension": [ 134.34, 0.37 ]

},

"eye": {...}

}

This is similar to what is done in step 2.3.

## RESTdesc: Next step: Call API 2

{

"http:body": {

"geometrical\_dimension": [ 134.34, 0.37 ],

"tolerances": [

{ "min": 134.3, "max": 134.35 },

{ "min": 0.37, "max": 0.4 }

],

"machine\_parameters": [ 1200.25, 0.0024, 13.7, 270 ],

"part\_number": "123",

"operator": "Gianni"

},

"http:resp": {

"http:body": {

"suggested\_parameters": [

"\_:b4\_sk6\_1",

"\_:b4\_sk7\_1",

"\_:b4\_sk8\_1",

"\_:b4\_sk9\_1"

],

"result": "\_:b4\_sk5\_4",

"id": "\_:b4\_sk4\_1"

}

},

"http:methodName": "POST",

"http:requestURI": "http://pacific-shore-4503.herokuapp.com/calibrations",

"data": [...]

}

This is the EYE result after the second user input round. This is quite similar to the result of 2.4. Since this API requires a POST call, a body is included. This is the exact JSON the server will POST to the API URI.

## RESTdesc 🡪 API 2: POST measurements

{

"geometrical\_dimension": [ 134.34, 0.37 ],

"tolerances": [

{ "min": 134.3, "max": 134.35 },

{ "min": 0.37, "max": 0.4 }

],

"machine\_parameters": [ 1200.25, 0.0024, 13.7, 270 ],

"part\_number": "123",

"operator": "Gianni"

}

The data that gets sent here is the body part shown in 2.9. Since it is a normal API, the server does the call again.

## API 2 🡪 RESTdesc: POST results

{

"id": 3,

"operator": "Gianni",

"part\_number": "123",

"machine\_parameters": [ 1200.25, 0.0024, 13.7, 270 ],

"tolerances": [

{ "min": 134.3, "max": 134.35 },

{ "min": 0.37, "max": 0.4 }

],

"geometrical\_dimension": [ 134.34, 0.37 ],

"result": "ok",

"suggested\_parameters": [ 1200.25, 0.0024, 13.7, 270 ]

}

This is the JSON response of API 2. When we add this information to EYE, it will indicate our goal has been reached (result == “ok”). If the result is “recalibrate” instead of “ok”, the RESTdesc server will return a response similar to the one in 2.7.