PILAR – REST API (v7.4)

José a.mañas

5.12.2019

1 Introduction

This document describes deployment and usage of PILAR as a service in a Tomcat.

PILAR is a tool for risk analysis and management.

Tomcat is a java servlet container developed by Apache Software Foundation.

REST (Representational State Transfer) is a software architectural style used to connect clients to web services.

Versions for this document:

PILAR: 7.4

Tomcat: 9.0

2 Installation

First, download and install a running version of PILAR, including its license.

Then, download the war file to deploy the server proxy to access PILAR via HTTP requests using the REST API.

https://www.pilar-tools.com/download/vxx/pilar_xxx.war

This file is to be deployed in your Tomcat installation.

2.1 Configuration

Please edit a few configuration parameters.

pilar.tool

pilar, or basic, or micro, according to your installation

pilar.home

where pilar is installed

pilar.car

configuration of PILAR

pilar.license

your license

pilar.log

for pilar to report

users.registry

where authorized users are saved; that is, authentication tokens

users.home (for saving in filesystem; for database storage, see below)

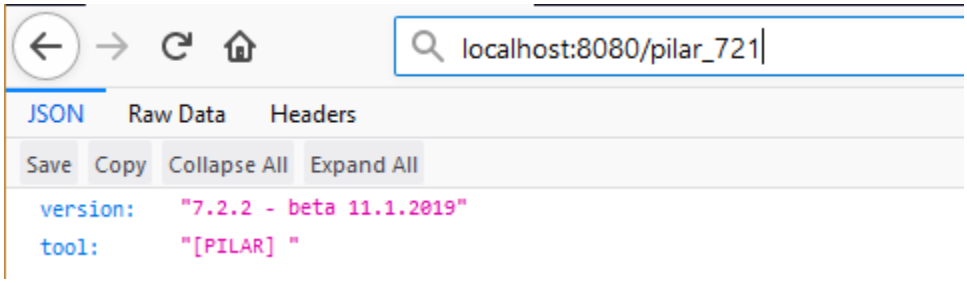
where risk analysis files are saved

Example

```
    <servlet>
        <servlet-name>pilar</servlet-name>
        <servlet-class>
          org.glassfish.jersey.servlet.ServletContainer
        </servlet-class>
        <init-param>
            <param-name>jersey.config.server.provider.packages</param-name>
            <param-value>pilar</param-value>
        </init-param>
        <init-param>
            <param-name>pilar.tool</param-name>
            <param-value>
              pilar
            </param-value>
        </init-param>
        <init-param>
            <param-name>pilar.home</param-name>
            <param-value>
              C:\Program Files (x86)\PILAR_7.2
            </param-value>
        </init-param>
        <init-param>
            <param-name>pilar.car</param-name>
            <param-value>
              CIS_en.car
            </param-value>
        </init-param>
        <init-param>
            <param-name>pilar.license</param-name>
            <param-value>
              C:\Users\USERNAME\AppData\Local\PILAR\LICENSE.lic
            </param-value>
        </init-param>
        <init-param>
            <param-name>pilar.log</param-name>
            <param-value>
              pilar.log
            </param-value>
        </init-param>
        <init-param>
            <param-name>users.registry</param-name>
            <param-value>
              C:\Users\USERNAME\AppData\Local\users-registry
            </param-value>
        </init-param>
        <init-param>
            <param-name>users.home</param-name>
            <param-value>
              C:\Users\USERNAME\AppData\Local\users
            </param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
```

2.2 initial test

Using a browser:



Command line

```
$ curl -i localhost:8080/pilar

HTTP/1.1 200
Content-Type: application/json
Content-Length: 54
Date: Sat, 12 Jan 2019 08:36:56 GMT

{"version":"7.2.2 - beta 11.1.2019","tool":"[PILAR] "}
```

2.3 Users

Users are expected to get an authorization before using the system. For two reasons:

- only authorized users are accepted
- PILAR needs to separate users' projects apart

This authorization is associated to a session, and the SessionId is used as authorization token.

As authentication database, currently, PILAR uses a registry file. This file is specified in parameter "users.registry". One line per user with the following format

```
$ cat registry
user1:digest(password1)
user2:digest(password2)
user3:digest(password3)
user4:diesgt(password4)
```

where digest(x) is the result of the tomcat realm digester.

For instance, if user "john" uses password "secret". First, calculate digest("secret"):

```
$ bin/digest.bat secret
secret:2577406ac00d6b76ebd9c276b110c9b06ce96496747f0b17af7c0bef86ece91c$1$d36f8de
5164a00f879a86e35438b1b382be56a6901cc48088d7ac8885dacf3010b277c369f0097a496bfea06
762672b8dde72b90c2da83791af85ceebb822a26
```

then input it into registry:

```
john:2577406ac00d6b76ebd9c276b110c9b06ce96496747f0b17af7c0bef86ece91c$1$d36f8de516
4a00f879a86e35438b1b382be56a6901cc48088d7ac8885dacf3010b277c369f0097a496bfea06762
672b8dde72b90c2da83791af85ceebb822a26
```

You may use "//" to comment lines out.

2.4 Usage

Example

GET login user password

First request must be a login using GET method. If successful, a session is setup. Following references should meet that session. Session maintenance depends on the applet server configuration. Typically, by means of cookies.

```
$ curl -i -X GET
'http://localhost:8080/pilar?command=login&user=john&password=secret'

HTTP/1.1 200
Set-Cookie: JSESSIONID=7BB13C3A73B84E13EF6ED6E6F5B9C67B; Path=/pilar; HttpOnly
Content-Length: 0
Date: Mon, 04 Feb 2019 07:41:15 GMT
```

If it fails, an error is returned, and there is no session id:

```
$ curl -i -X GET
'http://localhost:8080/pilar?command=login&user=john&password=secret2'

HTTP/1.1 401
Content-Type: text/html;charset=utf-8
Content-Language: en
Content-Length: 1068
Date: Mon, 04 Feb 2019 07:43:44 GMT
```

In order to use a session, the session id must be used in every query.

Here follows a command sequence as example:

```
$ curl -i -X GET
'http://localhost:8080/pilar?command=login&user=john&password=secret' -c
cookie.txt

HTTP/1.1 200
Set-Cookie: JSESSIONID=4581338BBF3B87983724CE372C701839; Path=/pilar; HttpOnly
Content-Length: 0
Date: Mon, 04 Feb 2019 09:05:43 GMT
```

```
$ cat cookie.txt
# Netscape HTTP Cookie File
# https://curl.haxx.se/docs/http-cookies.html
# This file was generated by libcurl! Edit at your own risk.

#HttpOnly_localhost     FALSE   /pilar   FALSE   0       JSESSIONID
4581338BBF3B87983724CE372C701839
```

```
$ curl -i -X POST http://localhost:8080/pilar --header "Content-Type:
application/json" -d "@version.json" -b cookie.txt

HTTP/1.1 200
Content-Type: application/json
Content-Length: 64
Date: Mon, 04 Feb 2019 09:07:23 GMT

{"version":"7.2.2 - 24.1.2019","tool":"[PILAR] ","status":"200"}
```

post SESSION version.json

Just to test the session is working.

```
$ cat version.json
{ "command": "version" }
```

A session may look like this

post SESSION project.open.json

```
$ cat project.open.json
{
  "command": "project.open",
  "project": "example_en.mgr",
  "opt.password": "password",
}
```

post SESSION assets.list.json

```
$ cat assets.list.json
{
  "command": "assets.list",
}
```

post SESSION project.save.json

```
$ cat project.save.json
{
  "command": "project.save",
  "project": "test",
  "opt.password": "password",
}
```

3 Using filesystem to store project files

For storage, there shall be a directory per user, with her projects

```
$ ls -R users/
users/:
john

users/john:
example_en.mgr  test.mgr  test_1.mgr
```

This directory is specified in parameter “users.home”.

4 Using a database to store users’ projects

Alternatively to saving projects in a shared directory, users.home, they may be saved in a shared database.

Instead of users.home, you must define in the configuration the following elements

users.db.url

data base url

users.db.jar

absolute path to .jar connector

users.db.class

driver class in path

users.db.user

db user

users.db.password (optional)

db password, if any

users.db.prefix

boolean

If prefix is TRUE, users’ projects will be saved using the userID as prefix, and the project name.

Otherwise, projects will be saved using userID:name es project name

Please note that not every database system supports the prefix concept. It is accepted, for instance, in Oracle data bases.

An example may help to clarify

users.db.url	jdbc.mysql://localhost/pilar
users.db.jar	/java/mysql-connector-java-5.1.8-bin.jar
users.db.class	com.mysql.jdbc.Driver
users.db.user	pilar
users.db.password	password
users.db.prefix	false

5 Assets

5.1 associate

Associates one or more essential assets with a security domain.

POST

```
{
  "command": "assets.associate"
  "codes": [ name, name … ]
  "domain": name
  "associate": true | false (optional; default: true)
}
```

Use

```
  "associate": false
```

to dissociate an asset from a security domain.

5.2 data

5.2.1 default

POST

```
{
  "command": "assets.data.default",
  "code": asset code
}
```

Returns the standard data associated to an asset. Entries are read from INFO configuration file.

See example in project.data.default.

5.2.2 get

POST

```
{
  "command": "project.data.get",
  "code": asset code
}
```

Returns the data associated to an asset.

See example in project.data.get.

5.2.3 set

POST

```
{
  "command": "project.data.set",
  "code": asset code
  "clear": boolean,     (optional)
  "data": [ key, name, value ]
}
```

Sets data into the asset. If 'clear' is set to true, previous data are removed.

See example in project.data.set.

5.3 delete

POST

```
{
"command": "assets.delete"
"codes": [ code, ... ]
}
```

5.4 list

POST

```
{
"command": "assets.list"
"codes": [ name, name … ] (optional)
"domains": [ name,… ] (optional)
"classes": [name, name … ] (optional)
"sources": [name, name … ] (optional)
"under": name (optional)
"report": [ name, … ] (optional)
}
```

Returns the assets that match the conditions: code, domain, class, source, or father.

"report" is a collection of words to control the components of the asset to return. The following ones are defined:

code, name, domain, description, classes, sources, data, dependencies

If the asset in directly in a layer, the "layer" attribute is returned.

If the asset is in a group, the "father" attribute is returned.

5.5 modify

POST

```
{
"command": "assets.modify"
"codes": [ code, ... ]
"classes": [name, name … ] (optional)
"sources": [name, name … ] (optional)
"domain": name (optional)
"data": { key: value, … } (optional)
"description": description (optional)
"gdpr": see GDPR section (optional)
"clear": Boolean (optional, default: true)
}
```

Classes, if provided, enrich or replace old ones, according to CLEAR.

Sources, if provided, enrich or replace old ones, according to CLEAR.

Data, if provided, enrich or replace old ones, according to CLEAR.

GDPR, if provided, enrich or replace old ones, according to CLEAR.

5.6 modify one

POST

```
{
"command": "assets.modify1"
"code": code
"new.code": code (optional)
"new.name": name (optional)
"classes": [name, name … ] (optional)
"sources": [name, name … ] (optional)
"domain": name (optional)
"data": { key: value, … } (optional)
"description": description (optional)
"gdpr": see GDPR section (optional)
"clear": Boolean (optional, default: true)
}
```

Classes, if provided, enrich or replace old ones, according to CLEAR.

Sources, if provided, enrich or replace old ones, according to CLEAR.

Data, if provided, enrich or replace old ones, according to CLEAR.

GDPR, if provided, enrich or replace old ones, according to CLEAR.

5.7 new

POST

```
{
"command": "assets.new"
"code": code
"name": name (optional)
"classes": [name, name … ] (optional)
"sources": [name, name … ] (optional)
"domain": name (optional, default: base)
"data": { key: value, … } (optional)
"description": description (optional)
"gdpr": see GDPR section (optional)
}
```

You may either locate the asset in a layer or under another asset. In each case, the asset becomes the last one in the layer or in the asset group, respectively.

To locate the asset as the last one in a layer, specify

```
"layer": layer code
```

To locate the asset as the last one under another asset, specify

```
"under": asset code
```

To locate the asset before, and at the same level as another asset, specify

```
"before": asset code
```

5.8 new (multi-asset)

In order to create several assets within the same request, the creating info may be repeated as an array:

```
{
"command": "assets.new"
"assets": [
  { "code": code
    "name": name (optional)
    "classes": [name, name … ] (optional)
    "sources": [name, name … ] (optional)
    "domain": name (optional)
    "data": { key: value, … } (optional)
    "description": description (optional)
    "gdpr": see GDPR section (optional)
  }
}
}
```

5.9 predefined

POST

```
{
"command": "assets.predefined"
}
```

5.10 value: get

POST

```
{
"command": "assets.value.get"
"code": code
}
```

It returns the own value, and the accumulated values, dimension by dimension. For example:

```
{
  "asset": {
    "code": "new-asset",
    "values": [
      {
        "level": "3",
        "why": [
          "3.lro"
        ],
        "comment": "an explanation",
        "dimension": "es.d"
      }
    ],
    "accumulated": [
      {
        "level": "4",
        "dimension": "es.d"
      },
      {
        "level": "0",
        "dimension": "es.i"
      },
      {
        "level": "7",
        "dimension": "es.c"
      },
      {
        "level": "0",
        "dimension": "es.a"
      },
      {
        "level": "0",
        "dimension": "es.t"
      },
      {
        "level": "0",
        "dimension": "es.v"
      },
      {
        "level": "0",
        "dimension": "es.dp"
      }
    ]
  },
  "status": "200"
}
```


5.11 value: set

POST

```
{
"command": "assets.value.set"
"code": code
"values": [ ... ]
}
```

where each value is a structure on itself, quite similar to the structure of get value reports:

```
{ "dimension": dimension-code,
  "level": level,
  "why": [ criteria ],
  "comment": a comment
}
```

where criteria and comment are optional.

6 Classes

6.1 Children

```
{
"command": "classes.children"
"codes": [ code, ... ] (optional)
"depth": depth (optional; default: 1)
}
```

The API returns the children of the asset classes, recursively, down to depth. Default depth is 1: direct children.

7 Controls (EVL security measures)

For referencing controls using PATTERNS and PATH, see Report Templates documentation.

7.1 Applicability: get

```
{
"command": "controls.applicability.get"
"evl”: name
"codes": [ code, ... ] (optional)
"patterns": [ pattern, ... ] (optional)
"path": [ step, ... ] (optional)
"domain": name
"name": true | false (optional)
}
```

If parameter ‘name’ is present and its value is TRUE, then the response includes the name of the control.

7.2 Applicability: set

```
{
"command": "controls.applicability.set"
"evl”: name
"codes": [ code, ... ] (optional)
"patterns": [ pattern, ... ] (optional)
"path": [ step, ... ] (optional)
"domain": name
"app": true | false
}
```

7.3 Children

```
{
"command": "controls.children"
"evl”: name
"codes": [ code, ... ] (optional)
"patterns": [ pattern, ... ] (optional)
"path": [ step, ... ] (optional)
"depth": depth (optional; default: 1)
}
```

The API returns the children of the controls, recursively, down to depth. Default depth is 1: direct children.

7.4 Comments: get

```
{
"command": " controls.comments.get"
"evl”: name
"codes": [ code, ... ] (optional)
"patterns": [ pattern, ... ] (optional)
"path": [ step, ... ] (optional)
"domain": name
"phase": name (optional)
"phases": [ name, name ... ] (optional)
}
```

The server will collect controls from codes, patterns and path.

If no phase is explicit, the server will use the pseudo phase ‘potential’.

7.5 Comments: set

```
{
"command": " controls.comments.set"
"evl”: name
"codes": [ code, ... ] (optional)
"patterns": [ pattern, ... ] (optional)
"path": [ step, ... ] (optional)
"domain": name
"phase": name (optional)
"comment": comment
}
```

The server will collect controls from codes, patterns and path.

If no phase is explicit, the server will use the pseudo phase ‘potential’.

7.6 Compensating controls

To manage compensating controls.

7.6.1 add

To add a compensatory control.

```
{
"command": " controls.compensating.add"
"evl": name
"old": name       // the control to be compensated
"data": { key : value }
}
```

Example

```
{
  "command": "controls.compensating.add",
  "evl": "ens:2015",
  "old": "op.pl.1",
  "data": {
    "compensating.measures.id": "[op.pl.1 alt] alternative to …",
    "compensating.measures.more": "more and more ..."
  }
}
```

It is important to note that keys under data are meaningful:

- compensating.measures.id has to include the unique code of the new control, between brackets
- the other keys may match those in configuration directory specification for compensating controls; for instance: bib_en/compensating_en.json

7.6.2 delete

To remove an already existing compensatory control.

```
{
"command": " controls.compensating.del"
"evl": name
"control": name         // the control to be deleted
}
```

7.6.3 modify

To modify an already existing compensatory control.

```
{
"command": " controls.compensating.mod"
"evl": name
"control": name         // the control to be modified
"data": { key : value }
}
```

Old data are replaced by new data key pairs. It includes the special entry "compensating.measures.id", that modifies code and name of the compensating measure.

It is important to note that keys under data are meaningful:

- "compensating.measures.id" has to include the unique code of the new control, between brackets
- the other keys may match those in configuration directory specification for compensating controls; for instance: bib_en/compensating_en.json

7.6.4 read

To retrieve the associated data of an already existing compensatory control.

```
{
"command": " controls.compensating.read"
"evl": name
"control": name         // the control to be modified
}
```

Returns an array with the data associated to the compensating control.

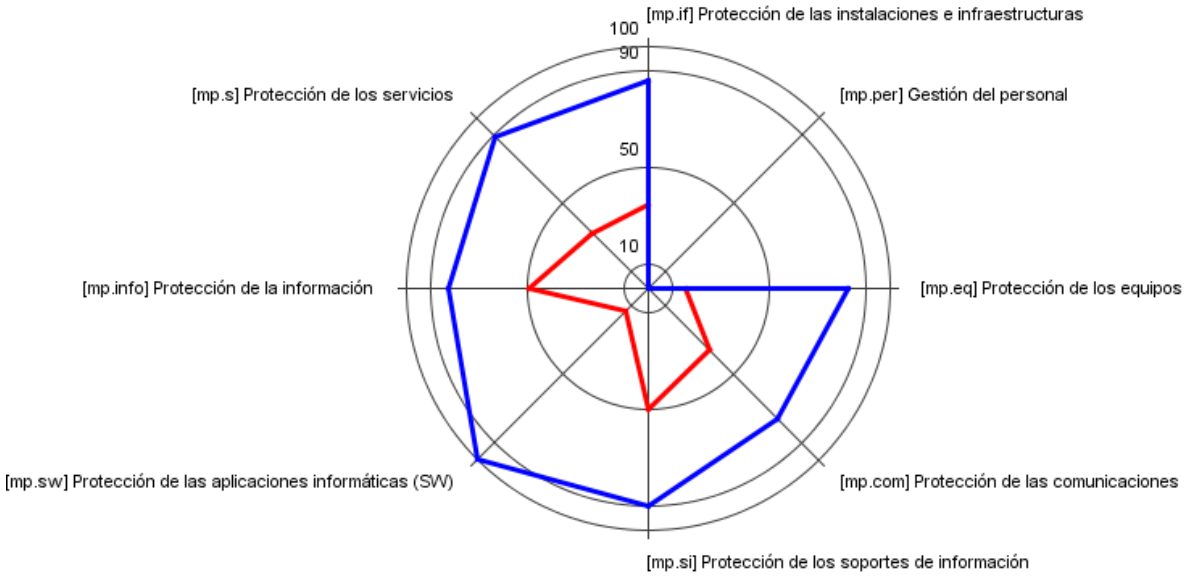
7.7 Graph

You may order a graph of the valuation of controls. See Graph section for further information.

Example query:

```
{
  "command": "controls.graph",
  "graph": "radar.lines",
  "evl": "ens:2015",
  "patterns": [ "mp.*" ],
  "domain": "base",
  "phases": [ "current", "target" ],
  "control-text": "[code] name",
  "phase-text": "[code] name"
}
```

The outcome is like this:



7.8 List

```
{
"command": "controls.list"
"evl”: name
"codes": [ code, ... ] (optional)
"patterns": [ pattern, ... ] (optional)
"path": [ step, ... ] (optional)
"domain": name (optional)
"applies": true | false (optional)
"mandatory": true | false (optional)
}
```

For each control, report on code, name, recommendation, and information sources.

If no domain es explicit, PILAR refers to base domain.

If applies is present, PILAR reduces the list to the controls that apply or to the controls that do not apply.

If mandatory is present, PILAR reduces the list to the controls that are mandatory or to the controls that are not.

7.9 Read

```
{
"command": "controls.read"
"evl”: name
"codes": [ code, ... ] (optional)
"patterns": [ pattern, ... ] (optional)
"path": [ step, ... ] (optional)
"domain": name
"phase": name
"mode": mode
}
```

Mode establishes the format of the response. See documentation on report templates. Default is “maturity”.

mode value	**prints**
0 | M | maturity	control maturity [range]
1 | M_M	control and safeguard maturity [range]
2 | _M	safeguard maturity [range]
3 | MA	control approximate maturity
4 | MA_MA	control and safeguard approximate maturity
5 | _MA	safeguard approximate maturity
6 | P | percent	safeguard percent (in ENS; compliance percent)
7 | P_P	control percent (safeguard percent)
8 | P_	control percent

Example query

```
{
  "command": "controls.read",
  "evl": "27002:2013",
  "codes": [ "6.1", "7.1.1", "8"],
  "path": [ "8", "8.*" ],
  "domain": "base",
  "phase": "current",
}
```

Response:

```
{
  "controls": [
    {
      "code": "7.1.1",
      "value": ""
    },
    {
      "code": "8",
      "value": "_-L5"
    },
    {
      "code": "6.1",
      "value": ""
    },
    {
      "code": "8.1",
      "value": "L5"
    },
    {
      "code": "8.2",
      "value": "n.a."
    },
    {
      "code": "8.3",
      "value": ""
    }
  ],
  "status": "200"
}
```


Let’s change mode:


```
  "mode": "P"
```


Fragment of response:

```
{
  "code": "8",
  "value": "50%"
},
{
  "code": "8.1",
  "value": "100%"
},
{
  "code": "8.2",
  "value": ""
},
{
  "code": "8.3",
  "value": ""
}
```

7.10 Read (multi-phase)

In order to return with a single request several values, you may specify an array of phases:

```
{
"command": "controls.read"
"evl”: name
"codes": [ code, ... ] (optional)
"patterns": [ pattern, ... ] (optional)
"path": [ step, ... ] (optional)
"domain": name
"phases": [ name, … ]
"mode": mode
}
```

7.11 Select

```
{
"command": "controls.select"
"evl”: name
"selected": code (optional)
"path": [ step, ... ] (optional)
"domain": name
"phase": name
}
```

You can select only one control, identified either by code, or by path.

7.12 Suggest

Given some risks, PILAR proposes controls to address it.

You may select those risks that match security domain, asset, threat and security dimension

```
{
"command": "controls.suggest"
"evl": security profile
"domains": [ code, ... ] (optional)
"assets": [ code, ... ] (optional)
"threats": [ code, ... ] (optional)
"dimensions": [ code, ... ] (optional)
"phase": name
"top": number (optional, default: 10)
}
```

Or you may specify one or more risks

```
{
"command": "controls.suggest"
"evl": security profile
"risks": [ <risk>, ... ]
"phase": name
"top": number (optional, default: 10)
}
```

where each risk is defined as

```
{ "asset": code, "threat": code, "dimension": code }
```

See “safeguards.suggest” for an example.

7.13 Write

```
{
"command": "controls.write"
"evl”: name
"codes": [ code, ... ] (optional)
"patterns": [ pattern, ... ] (optional)
"path": [ step, ... ] (optional)
"domain": name
"phase": name
"value": na L0 L1 L2 L3 L4 null
}
```

7.14 Compensated (removed)

When a control is compensated, you may set a maturity value that will not propagate to children.

In order to set a value that is compensated, add a flag to the controls.write query:

```
"compensated": true
```

When values are read from PILAR, the flag is returned in the controls.read response:

```
"compensated": true
```

8 CVE

8.1 Assign CVE to asset

8.2 Create / modify CVE

8.3 Get CVE

8.4 Remove CVE from asset

9 Domains, Security domains

9.1 delete

POST

```
{
  "command": "domains.delete"
  "codes": [ code, ... ]
}
```

9.2 list

POST

```
{
  "command": "domains.list"
}
```

9.3 modify

POST

```
{
  "command": "domains.modify"
  "code": code,
  "new.code": code, (optional)
  "new.name": name (optional)
}
```

9.4 new

POST

```
{
  "command": "domains.new"
  "code": code
  "name": name (optional)
  "under": code (optional)
  "sources": [name, name … ] (optional)
  "description": description (optional)
}
```

10 GDPR: Personal data

Personal data administrative information may be attached to projects and to individual assets.

Example:

```
  "gdpr": [
    { "key" : "gdpr.hr.opt3", "value" : false },
    { "key" : "gdpr.hr.opt4", "value" : true },
    { "key" : "gdpr.data.txt.2", "value" : "new responsible"
},
    { "key" : "gdpr.data.txt.3", "value" : "new data 3" }
  ]
```

Keys are taken from configuration file (CAR)

```
CIS_en.car:
    gdpr.data= gdpr_en.txt
```

11 Graphs

There are a number of graphs you may ask PILAR to generate. Currently

- safeguards.graph
- controls.graph
- risk.down.graph
- risk.up.graph

All of them share a number of calling arguments, and return.

Call arguments

graph

- radar.lines
- radar.areas
- radar.sectors
- lines.horizontal
- lines.vertical

- bars.horizontal
- bars.vertical

asset-text

control-text

phase-text

safeguard-text

Describes the format for including assets, etc in the graph. The default value is

- [code] name

PILAR responds with a graph, in PNG format, encoded as a base64 string.

And a legend describing the series presented.

Example:

```
{
      "status": "200",
      "graph": "iVBORw0KGgoAAAANSUhEUgAAAaUAAAFICAYAA…
      "legend": [
            {
                  "color": "#ff0000",
                  "name": "[potential] "
            },
            {
                  "color": "#0000ff",
                  "name": "[current] current frame"
            },
            {
                  "color": "#00ff00",
                  "name": "[target] target frame"
            }
      ]
}
```

Color is presented as a sequence on 3 2-hex numbers for red, green, and blue components. E.g. “#ff0000” for RED.

12 Layers

12.1 delete

POST

```
{
  "command": "layers.delete"
  "codes": [ code, ... ]
}
```

12.2 list

POST

```
{
  "command": "layers.list"
}
```

12.3 modify

POST

```
{
  "command": "layers.modify"
  "code": code,
  "new.code": code (optional)
  "new.name": name (optional)
  "sources": [name, name … ] (optional)
  "description": description (optional)
}
```

12.4 new

POST

```
{
  "command": "layers.new"
  "code": code
  "name": name (optional)
  "sources": [name, name … ] (optional)
  "description": description (optional)
}
```

13 Phases

13.1 delete

POST

```
{
  "command": "phases.delete"
  "codes": [ code, ... ]
}
```

13.2 list

POST

```
{
  "command": "phases.list"
}
```

13.3 modify

POST

```
{
  "command": "phases.modify"
  "code": code,
  "new.code": code (optional)
  "new.name": name (optional)
  "sources": [name, name … ] (optional)
  "description": description (optional)
  "date": day.month.year (optional)
}
```

13.4 new

POST

```
{
  "command": "phases.new"
  "code": code
  "name": name (optional)
  "at": position (optional)
  "sources": [name, name … ] (optional)
  "description": description (optional)
  "date": day.month.year (optional)
}
```

Position is the position of the new phase in the phase list. Counting from 0. If no position is provided, the phase is added at the end of user phases.

13.5 show

POST

```
{
  "command": "phases.show"
  "codes": [ code, ... ]
}
```

14 Project

14.1 add data

POST

```
{
  "command": "project.add"
  "data": { key: value, … } (optional)
  "description": description (optional)
  "gdpr": see GDPR section (optional)
}
```

If data is provided, previous values are retained, unless there is a new value for some key, that replaces the old one.

Description, if provided, replaces previous one.

GDPR data, if provided, replace previous ones.

This command is equivalent to

```
{
  "command": "project.modify",
  "clear": false,
  etc.
}
```

14.2 close

POST

```
{
  "command": "project.close"
}
```

14.3 data

14.3.1 default

POST

```
{
  "command": "project.data.default",
}
```

Returns the standard data associated to a project. Entries are read from INFO configuration file.

Example

```
{
      "status" : "200",
      "data" : [
            {
                  "key" : "org",
                  "name" : "Organización"
            },
            {
                  "key" : "desc",
                  "name" : "Descripción"
            },
```

14.3.2 get

POST

```
{
  "command": "project.data.get",
}
```

Returns the data associated to a project.

Example:

```
{
      "status" : "200",
      "data" : [
            {
                  "key" : "desc",
                  "name" : "Descripción",
                  "value" : "Pequeña oficina del ciudadano"
            },
            {
                  "key" : "propietario",
                  "name" : "propietario",
                  "value" : "Juan García Iturriaga"
            },
```

14.3.3 set

POST

```
{
  "command": "project.data.set",
  "clear": boolean,     (optional)
  "data": [ key, name, value ]
}
```

Sets data into the project. If ‘clear’ is set to true, previous data are removed.

Example:

```
{
  "command": "project.data.set",
  "clear": true,
  "data": [
    { "key": "key 1",
      "name": "name 1",
      "value": "val 1",
    },
    { "key": "key 2",
      "name": "name 2",
      "value": "val 2",
    }
  ]
}
```

14.4 download

POST

```
{
  "command": "project.download",
}
```

The current project is downloaded as a byte array encoded into base64. Mgr format, without password.

14.5 list

POST

```
{
  "command": "project.list"
}
```

Lists project information: code, name, data, and gdpr information.

14.6 modify

POST

```
{
  "command": "project.modify"
  "code": code (optional)
  "name": name (optional)
  "data": { key: value, … } (optional)
  "description": description (optional)
  "gdpr": see GDPR section (optional)
  "clear": Boolean (optional, default: true)
}
```

If data is provided, all previous values are removed.

Description, if provided, replaces previous one.

GDPR data, if provided, replace previous ones.

CLEAR, if true, implies that data and gdpr sets are cleared before loading new values.

14.7 new

POST

```
{
  "command": "project.new"
  "code": code
  "name": name (optional)
  "data": { key: value, … } (optional)
  "description": description (optional)
}
```

14.8 open

Opens an existing Project with read write access.

POST

```
{
  "command": "project.open"
  "project": filename | URL
  "password": password (optional)
}
```

14.9 options

POST

```
{
  "command": "project.options"
  "options": { key : value, ... }
}
```

The following options may be set for the current project:

key	values
accountability	0: manual 1: max(I, C)
authenticity	0: manual 1: max(I, C)
blank_measures	-2: n.a. 0: L0
threats	0: manual 1: mix 2: automatic
valuation	0: dependencies 1: domains 2: mix
format.frequency	0: potential 1: likelihood 2: ARO 3: level 4: ease
format.degradation	0: level 1: percent
format.maturity	0: maturity 1: status
evl.push	0: no 1: one by one 2: all
phases	0: linked 1: independent
phases_domains	0: first domains 1: first phases
xor	0: highest 1: selected
safeguards.threats	0: suggest 1: select
xdvt	0: OFF 1: ON
risk.evl.residual	0: likelihood 1: impact and likelihood
risk.residual	0: .. 4.2 3: 4.3 ..

14.10 read

Opens an existing Project with read only access.

POST

```
{
  "command": "project.read",
  "project": filename | URL,
  "password": password (optional)
}
```

14.11 save

POST

```
{
  "command": "project.save",
  "project": name (optional)
  "password": password (optional)
}
```

The Project name is translated into a filename if stored on file system, or into a URL using the configuration URL for a database.

Project name and optional password may be skipped if the project was open; the open values will be used for saving.

14.12 unlock

POST

```
{
  "command": "project.unlock",
  "project": name
}
```

14.13 upload

POST

```
{
  "command": "project.upload",
  "mgr": base64(file.mgr)
}
```

Takes a mgr file, encoded into base64, and loads it as current project on server side. The file shall have no password.

15 Report

Generates a report using templates in PILAR installation. See documentation on Report Templates.

Here is an example, explained below:

```
{
  "command": "report",
  "template": "Riesgo_es_tpl.rtf",

  "define": [
    { "ask": "phase",
      "name": null,
      "elements": "current"
    },
    { "ask": "phases",
      "name": null,
      "elements": "current, target"
    },
    { "ask": "domain",
      "name": null,
      "elements": "base"
    },
    { "ask": "domains",
      "name": null,
      "elements": "base, bps"
    }
  ]
}
```

The template attribute refers to a file in the library loaded from the context (car) when PILAR is instantiated in the server.

The define section provides data for the ask commands that may appear in the template. With respect to the example, it provides information for

- <ask.phase />
- <ask.phases />
- <ask.domain />
- <ask.domains />

If a name is used in the template, you have to copy it into the json spec

```
<ask.phases name=(last-period) />
{ "ask": "phases", "name": "last-period", "elements": … }
```

Other definitions may be specified in the request

```
  "define": [
    { "name": ...
      "value": ...
    },
```

PILAR returns a status and, if successful, the report in rtf format, zipped, and encoded in base64

```
{
  "status": "200",
  "info.array": [
    { "message": "Start template: Riesgo_es_tpl.rtf" }
  ],
  "report": "UEsDBBQACAgIAMiZ204AAAAAAAAAAAAAAAAKAAA..."
}
```

Decode “report” into a file.zip.

In the zip file there is a single member, “report.rtf”, that is the requested report.

16 Risk

16.1 Accumulated risk list

{
"command": "risk.down.list"
"assets": [name, …] (optional)
"threats": [name, …] (optional)
"dimensions": [name, …] (optional)
"phase": name (optional)
}
	{
	"command": "risk.down.list"
	"domains": [name, …] (optional)
	"threats": [name, …] (optional)
	"dimensions": [name, …] (optional)
	"phase": name (optional)
	}

It reports either the risks associated to one or more assets, or to one or more security domains. Optionally filtered by threat or dimension.

If phase is null, or there is no explicit phase, the potential risk is reported.

Risks are reported like this

```
{
	"status" : "200",
	"risks" : [
		{
			"asset" : "equipo",
			"threat" : "N.*",
			"dimension" : "es.d",
			"impact" : "[M-]",
			"likelihood" : "0,1",
			"risk" : "{1,8}"
		},
		{
			"asset" : "equipo",
			"threat" : "E.25",
			"dimension" : "es.d",
			"impact" : "[M-]",
			"likelihood" : "1",
			"risk" : "{2,7}"
		},
```

16.2 Deflected risk list (indirect risk)

{
"command": "risk.up.list"
"assets": [name, …] (optional)
"threats": [name, …] (optional)
"dimensions": [name, …] (optional)
"phase": name (optional)
}
	{
	"command": "risk.up.list"
	"domains": [name, …] (optional)
	"threats": [name, …] (optional)
	"dimensions": [name, …] (optional)
	"phase": name (optional)
	}

It reports either the risks associated to one or more assets, or to one or more security domains. Optionally filtered by threat or dimension.

If phase is null, or there is no explicit phase, the potential risk is reported.

Risks are reported like this

```
{
	"status" : "200",
	"risks" : [
		{
			"above" : "misión",
			"below" : "equipo",
			"threat" : "N.*",
			"dimension" : "es.d",
			"impact" : "[M-]",
			"likelihood" : "0,1",
			"risk" : "{1,8}"
		},
		{
			"above" : "misión",
			"below" : "equipo",
			"threat" : "E.25",
			"dimension" : "es.d",
			"impact" : "[M-]",
			"likelihood" : "1",
			"risk" : "{2,7}"
		},
```


16.3 Graph

You may order a graph of the valuation of risks.

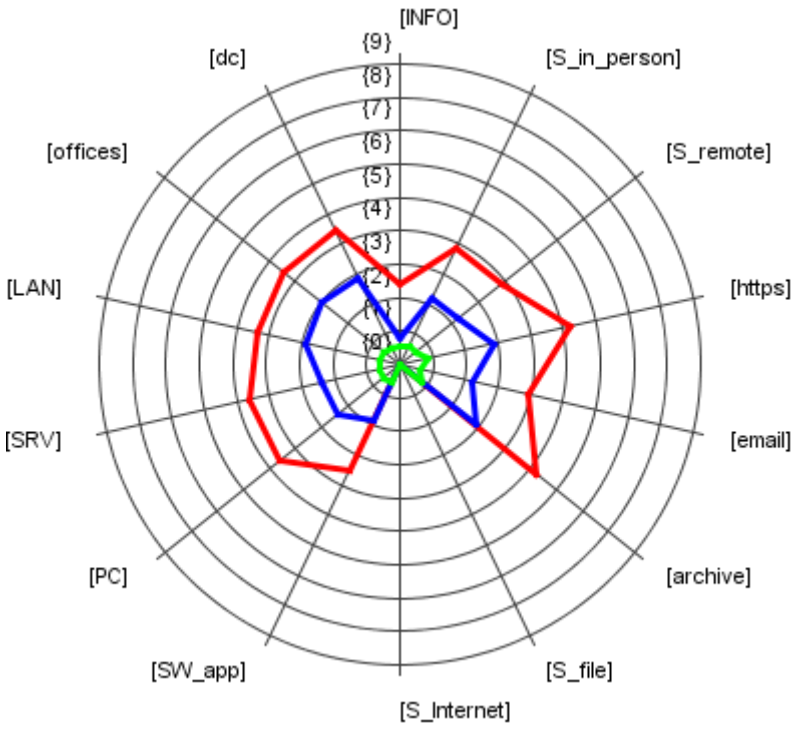
- "command": "risk.down.graph",
- "command": "risk.up.graph",

See Graph section for further information.

Example query:

```
{
  "command": "risk.down.graph",
  "graph": "radar.lines",
  "domains": [ "base" ],
  "phases": [ null, "current", "target" ],
  "asset-text": "[code] ",
  "phase-text": "[code] name"
}
```

The outcome is like this:



17 Safeguards (security measures)

For referencing safeguards using PATTERNS and PATH, see Report Templates documentation.

17.1 Applicability: get

```
{
  "command": "safeguards.applicability.get"
  "codes": [ code, ... ] (optional)
  "patterns": [ pattern, ... ] (optional)
  "path": [ step, ... ] (optional)
  "domain": name
  "name": true | false (optional)
}
```

If parameter ‘name’ is present and its value is TRUE, then the response includes the name of the safeguard.

17.2 Applicability: set

```
{
  "command": "safeguards.applicability.set"
  "codes": [ code, ... ] (optional)
  "patterns": [ pattern, ... ] (optional)
  "path": [ step, ... ] (optional)
  "domain": name
  "app": true | false
}
```

17.3 Children

```
{
  "command": " safeguards.children"
  "codes": [ code, ... ] (optional)
  "patterns": [ pattern, ... ] (optional)
  "path": [ step, ... ] (optional)
  "depth": depth (optional; default: 1)
}
```

The API returns the children of the safeguard, recursively, down to depth. Default depth is 1: direct children.

17.4 Comments: get

```
{
  "command": "safeguards.comments.get"
  "codes": [ code, ... ] (optional)
  "patterns": [ pattern, ... ] (optional)
  "path": [ step, ... ] (optional)
  "domain": name
  "phase": name (optional)
  "phases": [ name, name ... ] (optional)
}
```

The server will collect safeguards from codes, patterns and path.

If no phase is explicit, the server will use the pseudo phase ‘potential’.

17.5 Comments: set

```
{
  "command": "safeguards.comments.set"
  "codes": [ code, ... ] (optional)
  "patterns": [ pattern, ... ] (optional)
  "path": [ step, ... ] (optional)
  "domain": name
  "phase": name (optional)
  "comment": comment
}
```

The server will collect safeguards from codes, patterns and path.

If no phase is explicit, the server will use the pseudo phase ‘potential’.

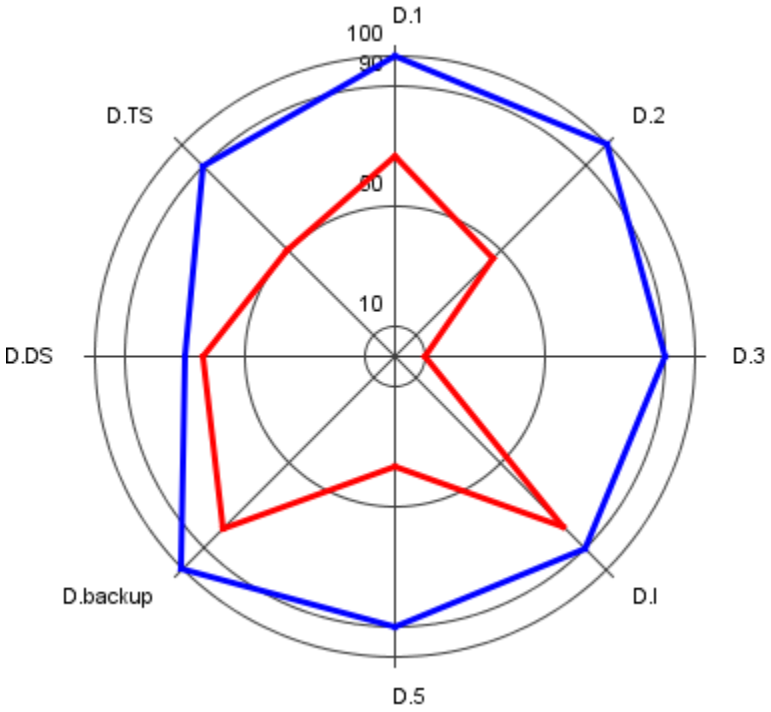
17.6 Graph

You may order a graph of the valuation of safeguards. See Graph section for further information.

Example query:

```
{
  "command": "safeguards.graph",
  "graph": "radar.lines",
  "path": [ "D", "+" ],
  "domain": "base",
  "phases": [ "current", "target" ],
  "safeguard-text": "code",
  "phase-text": "[code] name"
}
```

The outcome is like this:



17.7 List

```
{
  "command": "safeguards.list"
  "codes": [ code, ... ] (optional)
  "patterns": [ pattern, ... ] (optional)
  "path": [ step, ... ] (optional)
  "domain": name (optional)
  "applies": true | false (optional)
}
```

For each safeguard, report on code, name, recommendation, and information sources.

If no domain es explicit, PILAR refers to base domain.

If applies is present, PILAR reduces the list to the safeguards that apply or to the controls that do not apply.

17.8 Read

```
{
  "command": "safeguards.read"
  "codes": [ code, ... ] (optional)
  "patterns": [ pattern, ... ] (optional)
  "path": [ step, ... ] (optional)
  "domain": name
  "phase": name
}
```

17.9 Read (multi-phase)

In order to return with a single request several values, you may specify an array of phases:

```
{
  "command": " safeguards.read"
  "codes": [ code, ... ] (optional)
  "patterns": [ pattern, ... ] (optional)
  "path": [ step, ... ] (optional)
  "domain": name
  "phases": [ name, … ]
}
```

17.10 Select

```
{
  "command": "safeguards.select"
  "selected": code (optional)
  "path": [ step, ... ] (optional)
  "domain": name
  "phase": name
}
```

You can select only one safeguard, identified either by code, or by path.

17.11 Suggest

Given some risks, PILAR proposes safeguards to address it.

You may select those risks that match security domain, asset, threat and security dimension

```
{
  "command": "safeguards.suggest"
  "domains": [ code, ... ] (optional)
  "assets": [ code, ... ] (optional)
  "threats": [ code, ... ] (optional)
  "dimensions": [ code, ... ] (optional)
  "phase": name
  "top": number (optional, default: 10)
}
```

Or you may specify one or more risks

```
{
  "command": "safeguards.suggest"
  "risks": [ <risk>, ... ]
  "phase": name
  "top": number (optional, default: 10)
}
```

where each risk is defined as

```
{ "asset": code, "threat": code, "dimension": code }
```

Sample query

```
{
  "command": "safeguards.suggest",
  "phase": "current",
  "risks": [
    { "asset": "LAN", "threat": "A.24", "dimension": "D" }
  ]
}
```

Response

```
{
      "status" : "200",
      "suggest" : [
            {
                  "domain" : "base",
                  "safeguards" : [
                        {
                              "weight" : 62,
                              "code" : "HW.SC"
                        },
                        {
                              "weight" : 61,
                              "code" : "COM.CM.a"
                        },
                        {
                              "weight" : 61,
                              "code" : "COM.CM.d"
                        },
                        {
                              "weight" : 61,
                              "code" : "COM.CM.c"
                        },
```

The weight is a number from 0 (least important) to 100 (most important.

17.12 Write

```
{
  "command": "safeguards.write"
  "codes": [ code, ... ] (optional)
  "patterns": [ pattern, ... ] (optional)
  "path": [ step, ... ] (optional)
  "domain": name
  "phase": name
  "value": na L0 L1 L2 L3 L4 null
}
```

17.13 Compensated

When a safeguard is compensated, you may set a maturity value that will not propagate to children.

In order to set a value that is compensated, add a flag to the safeguards.write query:

```
"compensated": true
```

When values are read from PILAR, the flag is returned in the safeguards.read response:

```
"compensated": true
```

18 Session

Sessions are established via “login” and terminated via “logout”.

One session holds one PILAR project.

Login and logout are GET requests

login UUU XXX

```
?command=login&user=UUU&password=XXX
```

logout

```
?command=logout
```

19 Threats

19.1 Children

```
{
  "command": "threats.children"
  "codes": [ code, ... ] (optional)
  "depth": depth (optional; default: 1)
}
```

The API returns the children of the threats, recursively, down to depth. Default depth is 1: direct children.