PILAR – REST API (v2024.3)

3.10.2024

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: 2024.3.3

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)

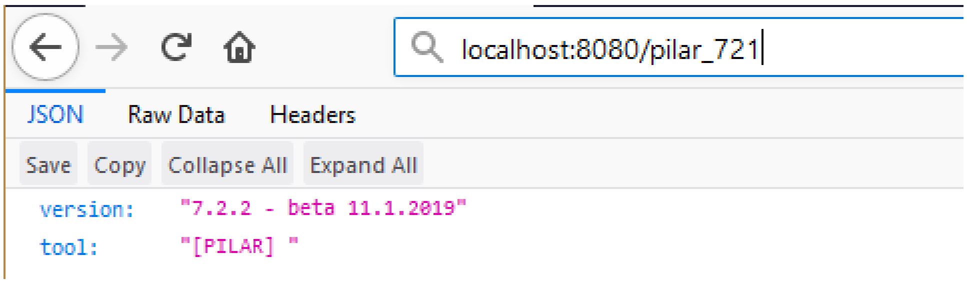
where risk analysis files are saved

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 Response

The response includes a status code according to standard HTTP status codes.

The most usual ones:

code	message	
200	OK	there may be partial errors. see below
400	BAD REQUEST	explained
401	UNAUTHORIZED	
403	FORBIDDEN	
404	NOT FOUND	
405	METHOD NOT ALLOWED	there may be partial errors. see below
500	INTERNAL SERVER ERROR	
501	NOT IMPLEMENTED	

5.1 No project,

```
status: 405
{       "status": 405,
        "error.array" : [
                {       "code" : 1420,
                        "message" : "no project",
                        "object" : null
                }
        ]
}
```

5.2 If the project is read-only

```
status: 401
{       "status": 401,
        "error.array": [
                {       "code": 1421        ,
                        "message": "read only project",
                        "object": project-id
                }
        ]
}
```

5.3 If the project is based on another project

```
status: 401
{	"status": 401,
	"error.array": [
		{	"code": 1422,
			"message": "based-on project",
			"object": project-id
		}
	]
}
```

5.4 Missing fields

When some field is missing in the query, and there is no default:

5.5 Unknown elements

If the command has succeeded, still some errors may happen that cannot be completed. For instance, you may post an order on several assets, but some assets are not found, then the command applies to the correct assets, and the incorrect ones are reported.

Example

```
status: 200
{
	"status": 200,
	"error.array": [
		{
			"code": 1401,
			"message": "unknown asset",
			"object": "HW"
		},
		{
			"code": 1405,
			"message": "unknown domain",
			"object": "unknown"
		}
	]
}
```

These are the available codes:

code	message	object
1401	unknown asset	asset id
1402	unknown class	class id
1403	unknown control	control id
1404	unknown dimension	dimension id
1405	unknown domain	domain id
1406	unknown evl	evl id

1407	unknown layer	layer id
1408	unknown maturity	maturity
1409	unknown phase	phase id
1410	unknown safeguard	safeguard id
1411	unknown source	source id
1412	unknown threat	threat id

5.6 Duplicated elements

If the command tries to create a duplicated element, it fails with an error report.

```
status: 403
{
      "status": 403,
      "error.array": [
            {
                  "code": 1441,
                  "message": "duplicated asset",
                  "object": "misión"
            }
      ]
}
```

These are the available codes

code	message	object
1441	duplicated asset	asset id
1442	duplicated domain	domain id
1443	duplicated layer	layer id
1444	duplicated phase	phase id
1445	duplicated source	source id

6 Assets

6.1 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.

6.2 data

6.2.1 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.

6.2.2 get

POST

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

Returns the data associated to an asset.

See example in project.data.get.

6.2.3 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.

DATA is an array of arrays of three strings each.

See example in project.data.set.

6.3 delete

POST

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

6.4 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)
"essential": boolean (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.

if the “essential” is present in the query, only essential assets are returned in the value is TRUE. Or only non-essential are returned if the value is FALSE.

6.5 modify

POST

```
{
"command": "assets.modify"
"codes": [ code, ... ]
"classes": [name, name … ] (optional)
"sources": [name, name … ] (optional)
"domain": name (optional)
"data": [ [key, name, 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.

6.5.1 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, name, 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.

6.6 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, name, 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
```

6.6.1 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, name, value] … ] (optional)
    "description": description (optional)
    "gdpr": see GDPR section (optional)
  }
}
}
```

6.7 predefined

POST

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

6.8 value

6.8.1 get

POST

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

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

```json
{
  "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
}
```


6.8.2 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.

6.9 Dependencies

6.9.1 List

```
{
"command": "assets.deps.list"
"codes": [ name, name … ] (optional)
"domains": [ name,… ] (optional)
"classes": [name, name … ] (optional)
"sources": [name, name … ] (optional)
"under": name (optional)
"essential": boolean (optional)
"depth": depth (optional; default: 1)
}
```

Assets are selected as in “assets.list”, but only dependencies are listed. The list of assets specifies the assets above. Assets below cannot be filtered.

6.9.2 Set

```
{
"command": "assets.deps.set"
"above": [ code, ... ]
"below": [ code, ... ]
"vector": vector (optional)
}
```

where codes are to identify assets, and the vector quantifies the dependency degree by security dimension:

6.9.2.1 example

6.9.3 Remove

```
{
"command": "assets.deps.remove"
"above": [ code, ... ]
"below": [ code, ... ]
}
```

6.9.3.1 Example

7 Classes

7.1 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.

8 Controls (EVL security measures)

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

8.1 Applicability

8.1.1 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.

8.1.2 set

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

8.2 Changes

```
{
"command": "controls.changes"
"evl”: name
"code": code
}
```

Reports the lists of controls and safeguards that may be modified when applicability or maturity are changed for the named control.

8.2.1 Sample answer

```
{ "status": 200,
  "controls": ["org","org.1","org.1.1", ...],
  "measures": ["IR.6","IR.6.1","IR.6.2", ...]
}
```

8.3 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.

8.3.1 Example query

```
{
  "command": "controls.children",
  "evl": "27002:2013",
  "codes": [ "5" ],
  "depth": 3
}
```

8.3.2 Example response

```
{
```

```
    "status": 200,
    "controls" : [
        {
            "code" : "5",
            "name" : "INFORMATION SECURITY POLICIES",
            "type" : "control",
            "level" : 1,
            "children" : [
                {
                    "code" : "5.1",
                    "name" : "MANAGEMENT DIRECTION ...",
                    "type" : "control",
                    "level" : 1,
                    "children" : [
                        {
                            "code" : "5.1.1",
                            "name" : "Policies for information security",
                            "type" : "control",
                            "level" : 1,
                            "children" : [
                                {
                                    "code" : "G.5.3",
                                    "name" : "Security policies",
                                    "type" : "measure",
                                    "level" : 2,
                                    "children" : [
                                        {
                                            "code" : "G.5.3.1",
                                            "name" : "Are ...",
                                            "level" : 1,
                                            "type" : "measure"
                                        },
                                        {
                                            "code" : "G.5.3.2",
                                            "name" : "It ...",
                                            "level" : 1,
                                            "type" : "measure"
                                        },
                                        {
                                            "code" : "G.5.3.3",
                                            "name" : "Resp...",
                                            "level" : 1,
                                            "type" : "measure"
                                        },
                                        {
                                            "code" : "G.5.3.4",
                                            "name" : "All ...",
                                            "level" : 1,
                                            "type" : "measure"
                                        },
                                        {
                                            "code" : "G.5.3.5",
                                            "name" : "Are ...",
```

```
                                        "level" : 2,
                                        "type" : "measure"
                                    },
                                    {
                                        "code" : "G.5.3.6",
                                        "name" : "Are ...",
                                        "level" : 1,
                                        "type" : "measure"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "code" : "5.1.2",
                        "name" : "Review ...",
                        "type" : "control",
                        "level" : 1,
                        "children" : [
                            {
                                "code" : "G.5.3.6",
                                "name" : "Are regularly reviewed",
                                "type" : "measure",
                                "level" : 1
                            }
                        ]
                    }
                ]
            }
        ]
    }
  ]
}
```

Type encodes the type of element. See control types, and safeguard types.

Level shows the level of the measure, a value between 0 (interesting) and 3 (critical).

8.4 Comments

8.4.1 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’.

8.4.1.1 Example

```
{
    "command": "controls.comments.get",
    "evl": "ens:2015",
    "codes": [ "org.1" ],
    "domain": "base"
}

{ "status": 200,
  "controls": [
    { "code": "org.1",
      "type": "control",
      "phase": "potencial",
      "comment": "this a test comment - no phase"
    },
    { "code": "org.1",
      "type": "control",
      "phase": "actual",
      "comment": "this a test comment - actual phase"
    }
  ]
}
```

8.4.2 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’.

8.4.2.1 Example

```
{
        "command": "controls.comments.set",
        "evl": "ens:2015",
        "codes": [ "org.1" ],
        "domain": "base",
        "phase": "actual",
        "comment": "this a test comment - actual phase"
}
```

8.5 Compensating controls

To manage compensating controls.

8.5.1 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

8.5.2 delete

To remove an already existing compensatory control.

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

8.5.3 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

8.5.4 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.

8.6 Doubts

You may associate a doubt mark a control. Doubt marks are read via the controls.one call.

```
{ "command": "controls.doubts.set",
  "evl": name
  "code": control code
  "doubt": boolean (optional, default: true)
}
```

Set "doubt" to true or false, to set or unset doubts on the control. Default is true.

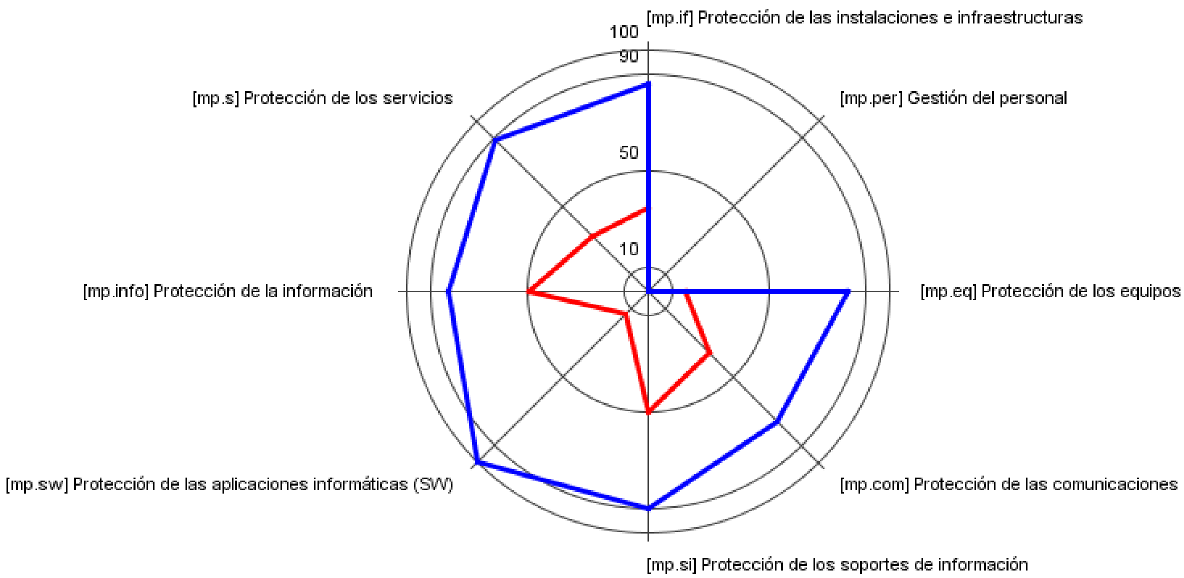
8.7 Graph

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:



8.8 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.

8.9 One

This query focuses on one (1) control and provides all the information related to it.

```
{
"command": "controls.one"
"evl”: name
"code": code
"domain": name (optional: default: base)
"mode": maturity-printing-mode
"phases": [ phase-code, ...]
"reference": phase-code        (optional)
"target": phase-code           (optional)
}
```

Returned fields

code	value	description
code	string	code of the control
name	string	name of the control
type	int	see section “control types”
level	int	0 – interesting – gray 1 – important – green 2 – very important – orange 3 – critical – red
father	string	if exists, the code of the father safeguard
children	string array	code and type of children controls / safeguards
rec	string	recommendation (may be null)
light	int	traffic light: see controls.one
sources	string array	if any, sources of information
doubts	int	doubts: see below
comments	int	comments: see below
app	int	applicability: see below
values	string array	maturity levels: see below

Applicability

Applicability is returned in the field “app” encoded as an integer value:

code	meaning
0	it does not apply
1	it does apply
2	some children apply, others do not

Mandatory

When a control is mandatory (required for compliance to the security profile), it is returned as a boolean TRUE. For example

```
"mandatory": true,
```

Doubts

When there are no doubts, the field is not returned in response. When there are doubts, it is reported as an integer in the field “doubts”:

code	meaning
1	there are doubts in this control
2	there are no doubts here, but there are doubts in children

Comments

If there is no comment, the field is not returned in response. When returned, here follows the encoding:

code	meaning
1	there are comments in this control
2	there are no comments here, but there are comments in children

Traffic light

If reference or target does not appear, no traffic light is returned. When returned, the traffic light is encoded as an integer value

```
"light": integer
```

code	meaning	color
0	ignored	black
1	not applies	light gray
2	none	white
3	far below	red
4	below	yellow
5	equals	green
6	above	blue

Maturity values

An array of values. Example:

```
"values": [
  { "phase": "phase_1",
    "value": "_-L3"
  },
  { "phase": "target",
```

```
        "value": "L2-L5"
      },
      { "phase": "PILAR",
        "value": "L2-L3"
      }
    ]
```

8.9.1 Example query

```
{
  "command": "controls.one",
  "evl": "ens:2015",
  "code": "org.1",
  "domain": "base",
  "phases": [ "actual", "current", "ENS" ],
  "reference": "actual",
  "target": "ENS",
  "mode": "M"
}
```

8.9.2 Example answer:

```
{
  "status": 200,
  "control": {
    "code": "org.1",
    "name": "Política de Seguridad",
    "type": "control",
    "level": 1,
    "father": "org",
    "children": [
      { "code": "org.1.1",
        "type": "question"
      },
      { "code": "org.1.2",
        "type": "question"
      },
      { "code": "org.1.3",
        "type": "question"
      },
      { "code": "org.1.4",
        "type": "question"
      },
      { "code": "org.1.r1",
        "type": "question"
      }
    ],
    "rec": "5",
    "light": 3,
    "app": 1,
    "mandatory": true,
    "values": [
      { "phase": "actual",
        "value": "_-L5"
```

```
        },
        { "phase": "current",
          "value": "_-L5"
        },
        { "phase": "ENS",
          "value": "L2-L3"
        }
      ]
    }
  }
```

8.10 Read

```
{
"command": "controls.read"
"evl”: name
"codes": [ code, ... ] (optional)
"patterns": [ pattern, ... ] (optional)
"path": [ step, ... ] (optional)
"domain": name
"phase": name
"mode": maturity-printing-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
9 | C	for some security profiles, like ENS, with formal compliance definition

8.10.1 Example query

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

8.10.2 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": ""
}
```

8.10.3 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": maturity-printing-mode (see controls.read)
}
```

8.11 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.

You select one (1) phase or, alternatively, you may select several phases with one API call

```
"phases": [ "current", "target" ]
```

8.12 Stats

Provides some statistics about a security profile.

- the number of explicit maturity values per top level groups of controls, per security domain, and aggregated

- maturity and compliance indexes per security domain

Example query

```
{
        "command": "controls.stats",
        "evl": "ens:2015"
}
```

Example response

```
{ "status": 200,
  "stats": [
    { "domain": "base",
      "control": "org",
      "count": 29
    },
    { "domain": null,
      "control": "org",
      "count": 29
    },
...    ...    ...

    { "domain": "base",
      "phase": "actual",
      "index.maturity": 0.10666666666666667,
      "index.compliance": 0.17777777777777778
    },
    { "domain": "base",
      "phase": "objetivo",
      "index.maturity": 0.5816666527589163,
      "index.compliance": 0.7777777777777778
    }
  ]
}
```

8.13 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.

8.14 Control types

There are several types of controls. This type is returned as a string with the label “type”

- control: 
- question: 
- see: 
- link: 
- compensatory: 

See also safeguard types.

8.15 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
"changes": boolean
"mode": maturity-printing-mode (see controls.read)
}
```

If ‘changes’ is true, the api returns the list of controls and security measures that change because of the command. The format is a string, using the mode specified, or ‘maturity’ by default.

8.15.1 Example query


```
{
  "command": "controls.write",
  "evl": "27002:2013",
  "path": [ "8", "8.1", "8.1.2" ],
  "domain": "base",
  "phase": "target",
  "value": "L2",
  "changes": true
}
```

8.15.2 Example response


```
{ "status": 200,
  "delta": [
    { "domain": "base",
      "delta": [
        { "control": "8",
          "values": [ "_-L1", "_-L2" ]
        },
        { "control": "8.1",
          "values": [ "L1", "L1-L2" ]
        },
        { "control": "8.1.2",
          "values": [ "L1", "L2" ]
        },
        { "measure": "AUX.1",
          "values": [ "L1", "L1-L2" ]
        },
        { "measure": "AUX.1.2",
          "values": [ "L1", "L2" ]
        },
        { "measure": "SW.1.1.4",
          "values": [ "L1", "L2" ]
        },
        { "measure": "COM.1.3",
          "values": [ "L1", "L2" ]
        },
        { "measure": "COM.1",
          "values": [ "L1", "L1-L2" ]
        },
        { "measure": "D.1.2",
          "values": [ "L1", "L2" ]
        },
        { "measure": "PS.3",
          "values": [ "L1", "L1-L2" ]
        },
        { "measure": "PS.3.3",
          "values": [ "L1", "L2" ]
        },
        { "measure": "S.2.2.2",
          "values": [ "L1", "L2" ]
        },
        { "measure": "MP.1.3.2",
          "values": [ "L1", "L2" ]
        },
        { "measure": "MP.1.3",
          "values": [ "L1", "L1-L2" ]
        },
        { "measure": "HW.1.1",
          "values": [ "L1", "L1-L2" ]
        },
        { "measure": "HW.1.1.3",
          "values": [ "L1", "L2" ]
```

```
          },
          { "measure": "L.2",
            "values": [ "L1", "L1-L2" ]
          },
          { "measure": "L.2.3",
            "values": [ "L1", "L2" ]
          },
          { "measure": "SW.1.1",
            "values": [ "L1", "L1-L2" ]
          }
        ]
      }
    ]
  }
```

8.16 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
```

8.17 Subsets

```
POST
    {
      "command": "controls.subset"
      "evl": name
      "code": code (optional)
    }
```

When there are subsets of a security profile, this call permits to consult which ones, and their contents.

EVL is the code of the security profile. E.g. “ens:2022”

CODE is the name of the subset. E.g. “round1”

If no CODE is provided, PILAR returns the list of available subsets.

The response is a list of subset names (if none is specified in the query) or the list of controls in the subset.

8.18 Clear

```
{
"command": "controls.clear"
"evl": name
"codes": [ code, ... ] (optional)
"patterns": [ pattern, ... ] (optional)
"path": [ step, ... ] (optional)
"domain": name
"phase": name
"what": ["source","comment","applicability","doubt","maturity"]
               (optional; default: clears all)
}
```

Clear (that is, remove) the mentioned elements of the selected control(s) in the selected domain(s) and phase(s).

8.18.1 Example query

```
{
  "command": "controls.clear",
  "evl": "27002:2013",
  "path": [ "8", "8.1", "8.1.2" ],
  "domain": "base",
  "phase": "target",
  "what": [ "maturity", "doubts" ]
}
```

9 CVE

9.1 Create / modify CVE

POST

CVSS version 2	CVSS version 3
{ "command": "cve.define", "id": code, "cvss3": cvss-vector }	{ "command": "cve.define", "id": code, "cvss2": cvss-vector }

9.2 Assign CVE to asset

POST

```
{
  "command": "cve.associate",
  "cve": cve identifier,
  "asset": asset code
}
```

9.3 Get CVE

POST	Response
{ "command": "cve.get", "id": cve identifier }	{ "cve": id "cvss2": cvss vector version 2 (if available) "cvss3": cvss vector version 3 (if available) }

9.4 Remove CVE

POST

```
{
  "command": "cve.remove",
  "cve": cve identifier
}
```

9.5 Remove CVE from asset

POST

```
{
  "command": "cve.dissociate",
  "cve": cve identifier,
  "asset": asset code
}
```

10 Domains, Security domains

10.1 delete

POST

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

10.2 list

POST

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

10.3 modify

POST

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

10.4 new

POST

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

11 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 the specification file referred from the configuration file (CAR)

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

12 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.

13 Impact

Impact values are usually part of the responses to “risk.down.list” and “risk.up.list”.

13.1 Accumulated impact summary

```
{
  "command": "impact.down.summary"
  "phases": [ name, ... ] (optional; default all phases)
  "phase": name (optional; default potential)
  "top": # (optional; default 20)
}
```

Reports the impact in each one the phases. The top parameter restricts listing to the top # rows of most relevant impact.

See “risk.down.summary”.

14 Layers

14.1 delete

POST

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

14.2 list

POST

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

14.3 modify

POST

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

14.4 new

POST

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

15 Phases

15.1 delete

POST

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

15.2 list

POST

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

15.3 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)
}
```

15.4 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.

15.5 show

POST

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

16 Project

16.1 add data

POST

```
{
  "command": "project.add"
  "data": [ [key, name, 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.
}
```

16.2 close

POST

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

16.3 data

16.3.1 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"
            },
```

16.3.2 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"
            },
```

16.3.3 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 1", "name 1", "val 1"],
       [ "key 2", "name 2", "val 2"]
  ]
}
```

16.4 download

POST

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

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

16.5 list

POST

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

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

16.6 modify

POST

```
{
  "command": "project.modify"
  "code": code (optional)
  "name": name (optional)
  "data": [ [key, name, 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.

16.7 new

POST

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

16.8 open

Opens an existing Project with read write access.

POST

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

16.9 options

POST

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

The following options may be set for the current project:

key	values
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

key	values
xdvt	0: OFF 1: ON
risk.evl.residual	0: likelihood 1: impact and likelihood
risk.residual	0: .. 4.2 3: 4.3 ..

16.10 marking

POST

```
{
  "command": "project.marking"
  "mark": value (integer)
}
```

Sets the security classification marking.

The values depend on the profile (car). Unless the user has personalized the security marks, the standard values are

Val	Text (English)
0	TOP SECRET
1	SECRET
2	CONFIDENTIAL
3	RESTRICTED
4	UNCLASSIFIED
5	<<blank>>

Example. To set classification mark to RESTRICTED,

```
{
  "command": "project.marking",
  "mark": 3
}
```

16.11 read

Opens an existing Project with read only access.

POST

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

16.12 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.

16.13 unlock

POST

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

16.14 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.

17 Report

17.1 Generation by template

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"
    }
    { "ask": "stage",
      "name": null,
      "elements": "base"
    },
    { "ask": "stages",
      "name": null,
      "elements": "base, acc"
    }
  ]
}
```

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 />
- <ask.stage />
- <ask.stages />

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": … }
```

If there is only one phase, or one domain, or one stage, you do not need to provide these hints. PILAR will choose the item.

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.

17.2 Available reports

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

17.2.1 Sample response

```
{ "status": 200,
  "reports": [
    { "template": "SOA_ens_2015_tpl.rtf",
      "name": "Declaración de Aplicabilidad: ..."},
```

17.3 Document download

Downloads a document from the reports repository. The document is not treated as a template but downloaded as-is.

```
{
  "command": "report",
  "template": "SomeFancyReport.pdf"
}
```

17.4 Inés

Generates an XML report to be loaded into INES, the annual reporting tool of the Spanish Government.

```
{
  "command": "report.ines",
  "domains": [ name, ... ]     (optional; default: all domains)
}
```

The output report is documented in CCN-STIC 804. It is an XML file, delivered in a ZIP file, encoded using Base-64.

18 Risk

18.1 Accumulated risk list

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

The range for impact and likelihood may be one of:

	Impact-range	Likelihood-range
	Risks where impact level is	Risks where ARO is
L	smaller than [3]	Below 1
M	[3], [4] or [5]	Between 1 and 10
H	[6] or higher	above 10

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}"
        },
```


18.2 Accumulated risk aggregated by asset

```
{
  "command": "risk.down.byasset"
  "assets": [ name, … ] (optional; default all)
  "threats": [ name, …] (optional; default all)
  "dimensions": [ name, …]
  "phase": name (optional; default potential)
}
```

Returns the accumulated risk, aggregated by security asset. Aggregation is done by security dimension.

18.3 Accumulated risk aggregated by domains

```
{
  "command": "risk.down.bydomain"
  "domains": [ name, … ] (optional; default all)
  "threats": [ name, …] (optional; default all)
  "dimensions": [ name, …]
  "phase": name (optional; default potential)
}
```

Returns the accumulated risk, aggregated by security domain.

18.4 Accumulated risk summary

```
{
  "command": "risk.down.summary"
  "phases": [ name, ... ] (optional; default all phases)
  "phase": name (optional; default potential)
  "top": # (optional; default 20)
}
```

Reports the risks in each one the phases. The top parameter restricts listing to the top% of risks sorted by risk in the referenced phase.

For instance, top: 10, trims response after 10 rows of highest risks.

18.4.1 Example response

```
{ "status": 200,
  "summary": [
    { "asset": "archive",
      "threat": "A.11",
      "dimension": "es.a",
      "risks": [ "{2,5}", "{0,98}", ...]
    },
```

18.5 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}"
		},
```

18.6 Graph

You may order a graph of the valuation of risks.

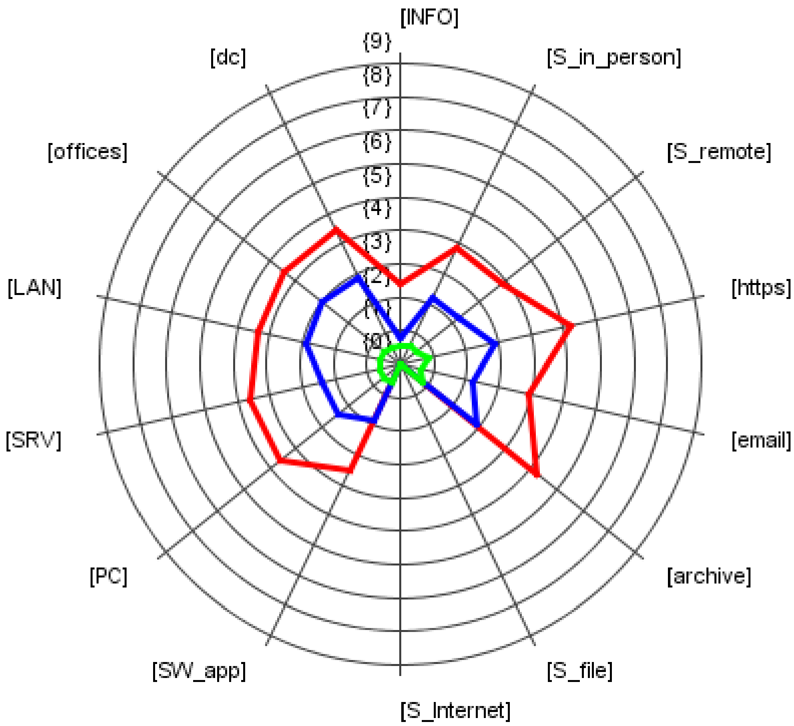
- "command": "risk.down.graph",
- "command": "risk.up.graph",

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:



18.7 Heat Map

A heat map is a special type of graph.

You may request a heatmap for the top accumulated risks

```
{
  "command": "risk.down.heatmap",
  "phases": [ name, ... ]      (optional; default all phases)
  "assets": [ name, ... ]      (optional; default all assets)
  "top": 20                    (optional; default: 20)
}
```

The API returns an image encoded as Base64, and a legend.

Example:

```
{
  "status": 200,
  "graph": "iVBORw0KGgoAAAANSUhEUgAAA3wA ...",
  "legend": [
    {
      "label": "a",
      "asset": "[pc] PC",
      "threat": "[A.11] Acceso no autorizado",
      "dimension": "I, C"
    },
    {
      "label": "b",
      "asset": "[server] Servidor",
      "threat": "[A.11] Acceso no autorizado",
      "dimension": "I, C, A"
    },
    {
      "label": "c",
      "asset": "[pc] PC",
      "threat": "[A.3] Manipulación de los registros de actividad",
      "dimension": "I"
    },
```

19 Safeguards (security measures)

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

19.1 Applicability

19.1.1 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.

19.1.2 set

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

19.2 Changes

```
{
"command": "safeguards.changes"
"code": code
}
```

Reports the lists of safeguards that may be modified when applicability or maturity are changed for the named safeguards.

19.2.1 Sample answer

```
{ "status": 200,
  "measures": [ "IA","IA.4.1","IA.4.2","IA.4.2.1", ...]
}
```

19.3 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.

19.4 Comments

19.4.1 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’.

19.4.2 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’.

19.5 Doubts

You may associate a doubt mark to a safeguard. Doubt marks are read via the safeguards.one call.

```
{  "command": "safeguards.doubts.set",
    "code": safeguard code
    "doubt": boolean (optional, default: true)
}
```

Set “doubt” to true or false, to set or unset doubts on the safeguard. Default is true.

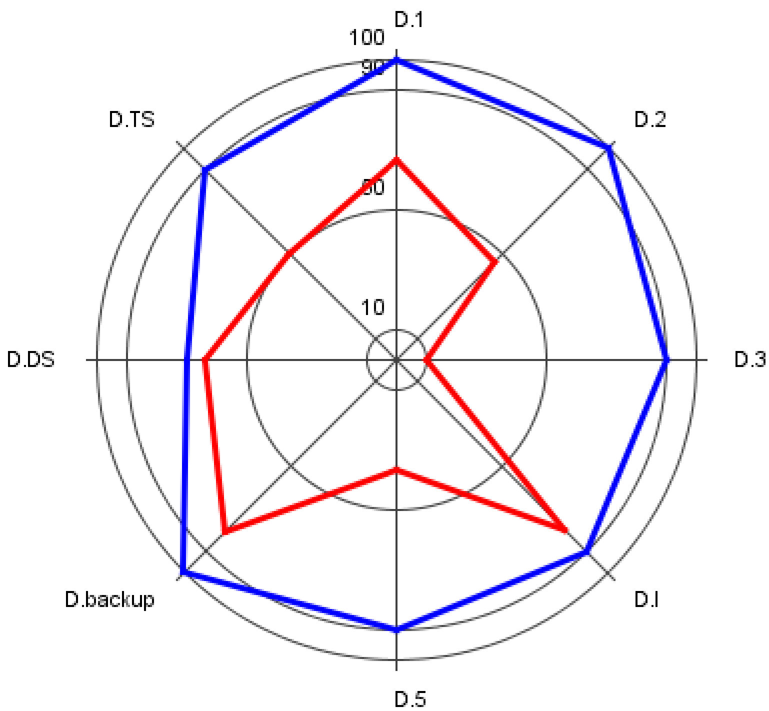
19.6 Graph

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:



19.7 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.

19.8 One

This query focuses on one (1) safeguard and provides all the information related to it.

```
{
"command": "safeguards.one"
"code": code
"domain": name (optional: default: base)
"phases": [ phase-code, ...]
"reference": phase-code       (optional)
"target": phase-code          (optional)
}
```

Returned fields

code	value	description
code	string	code of the safeguard
name	string	name of the safeguard
type	int	see section “safeguard types”
level	int	0 – interesting – gray umbrella 1 – important – green umbrella 2 – very important – orange umbrella 3 – critical – red umbrella
aspect	int	0 – management 1 – technical 2 – physical 3 – personnel
top	string	type of protection PR – prevention DR – deterrence EL – elimination IM – impact minimization CR – correction RC – recovery RK – risk DC – detection MN – monitoring AW – awareness AD – administrative STD – policy PROC – procedure CERT – certificate
father	string	if exists, the code of the father safeguard
children	string array	code and type of children safeguards
rec	string	recommendation (may be null)

light	int	traffic light: see controls.one
sources	string array	if any, sources of information
doubts	int	doubts: see controls.one
comments	int	comments: see controls.one
app	int	applicability: see controls.one
values	string array	maturity levels: see controls.one

19.8.1 Example query

```
{
  "command": "safeguards.one",
  "code": "D.2.1.1",
  "phases": [ "actual", "current", "ENS" ],
  "reference": "actual",
  "target": "ENS"
}
```

19.8.2 Example answer:

```
{
  "status": 200,
  "control": {
    "code": "D.2.1.1",
    "name": "...",
    "level": 1,
    "father": "D.2.1",
    "children": [
      "D.2.1.1.1",
      "D.2.1.1.2",
      ...
    ],
    "rec": null,
    "light": 2,
    "app": 1,
    "values": [
      { "phase": "actual",
        "value": ""
      },
      { "phase": "ENS",
        "value": "n.a."
      }
    ]
  }
}
```

19.9 Read

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

19.9.1 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, … ]
}
```

19.10 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.

19.11 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.

19.12 Safeguard types

There are several types of security measures:

- measure: 
- border: 
- cert: 
- book: <empty>

19.13 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
}
```

19.14 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
```

20 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
```

21 Threats

21.1 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.

21.2 Set standard values

```
{
"command": "threats.set"
"codes": [ code, ... ]
"classes": [ code, ... ]    (optional)
"likelihood": <double>
"deg": [ [ dimension, deg ] ]
}
```

You may specify several [dim,deg] pairs.

The order modifies the TSV used. If no classes are specified, it modifies the parameters of the classes that are already associated to the threat.

Example

```
{ "command": "threats.set",
  "codes": [ "X.11" ],
  "classes": [ "D" ],
  "likelihood": 10,
  "deg": [ ["en:A", 100] ]
}
```