

# CyberCAPTOR API Open Specification

DATE: September 2015

## Editors

1. Francois-Xavier Aguessy, Thales
  2. Stephan Neuhaus, ZHAW
  3. Roman Müntener, ZHAW
- 

## Abstract

This is the Open Specification of the REST API of **CyberCAPTOR**. It contains the API used between *CyberCAPTOR Server* and *CyberCAPTOR Client*, and the one of *CyberCAPTOR P2DS*.

**CyberCAPTOR** implements the FIWARE CyberSecurity Generic Enabler.

### Group (object)

- + gid (number) - The group's identifier, guaranteed to be unique for this group manager.
- + name (string) - A descriptive name of a group

### PeerInfo (object)

- + gid (number) - The group to which this peer belongs.
- + lastStatus (number) - The last known status of this peer (unknown = 0, started = 1, error = 2, stopped = 3)
- + peerName (string) - A descriptive name for the peer.
- + peerType (number) - The type of peer (input peer = 1, privacy peer = 2)
- + publicKey (string) - The peer's public key.
- + url (string) - The URL at which to contact the peer.

### DataSet (object)

- + data (string) - semicolon separated integers
- + peerName (string) - Name of the peer to add the data set for

### DataSets (object)

- + data (list) - List of semicolon separated integers
- + peerName (string) - Name of the peer

### GroupConfigurationInfo (object)

- + field (string) - Field value for the underlying MPC protocol; a good value is 9223372036854775783 (BigInteger as String)
- + maxElement (string) - Maximum value the underlying MPC protocol shall accept (BigInteger as String)
- + gid (int) - ID of the group
- + mpcProtocol (string) - MPC protocol to use.
- + numberOfItems (int) - Number of items within a single data sets.
- + numberOfTimeSlots (int) - Number of time slots (in other terms, how many data sets are expected). Negative values mean an infinite amount (pure streaming).
- + resultBufferSize (number) - Size of the buffer for final results

### GroupName (object)

- + name (string) - Name of the group

### Peer (object)

- + gid (number) - The group to which this peer belongs.
- + lastStatus (number) - The last known status of this peer (unknown = 0, started = 1, error = 2, stopped = 3)
- + peerName (string) - A descriptive name for the peer (informational only)
- .
- + peerType (number) - The type of peer (input peer = 1, privacy peer = 2)
- + publicKey (string) - The peer's public key.
- + url (string) - The URL at which to contact the peer.
- + registrationCode (string) - The registration code used to register this peer.
- + verified (boolean) - Whether this peer has been marked as verified or not
- .

### Registration (object)

- + gid (number) - ID of the group.
- + registrationCode (string) - The registration code.

### PeerConfigurationInfo (object)

- + name (string) - Name of the peer
- + registrationCode (string) - Registration code
- + publicKey (string) - public key (base64 encoded)
- + privateKey (string) - private key (base64 encoded)
- + groupMgmtURL (string) - URL of the group management service.
- + peerType (int) - Type of the peer (1 = input peer, 2 = privacy peer)
- + finalResultsURL (string) - URL to send final results to.

### PeerConfigurationInfoCollection (object)

- + peers - List of PeerConfiguration (peers)



# Table of Contents

CyberCAPTOR API Open Specification	1
Editors	1
Abstract	1
Group (object)	1
PeerInfo (object)	1
DataSet (object)	1
DataSets (object)	1
GroupConfigurationInfo (object)	1
GroupName (object)	2
Peer (object)	2
Registration (object)	2
PeerConfigurationInfo (object)	2
PeerConfigurationInfoCollection (object)	2
Table of Contents	4
Specifications	9
Introduction	9
Scored Attack Paths	9
Remediations	9
Privacy-Preserving Data Sharing	9
Terminology	10
Scored attack paths and remediations	10
Privacy-Preserving Data Sharing	10
Concepts	11
API Specification	13
CyberCAPTOR-Server REST API without initialization	13
Group Version [/rest/version]	13
Version - GET /rest/version	13
VersionDetailed - GET /rest/version/detailed	13
Group Configuration [/rest/json/configuration]	13
Get global remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/global	13
Set global remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/global	13
Get snort rule remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/snort-rule	14
Set snort rule remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/snort-rule	14
Get firewall rule remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/firewall-rule	14
Set firewall rule remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/firewall-rule	14
Get patch remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/patch	15
Set patch remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/patch	15
IDMEF	15
Add IDMEF alerts [/rest/json/idmef/add]	15
Add IDMEF alerts - POST /rest/json/idmef/add	15
CyberCAPTOR-Server REST API after initialization	15
Initialize [/rest/json/initialize]	16
Initialize from data on disk - GET /rest/json/initialize	16
Initialize from XML topology - POST /rest/json/initialize	16
Group Get the XML topology [/rest/json/topology]	16
Get XML topology - GET /rest/json/topology	16
Group Hosts [/rest/json/host/list]	17
Get the host list - GET /rest/json/host/list	17
Set the host list - POST /rest/json/host/list	17
Group Attack graphs [/rest/json/attack_graph]	18
Get the attack graph - GET /rest/json/attack_graph	18
Get the attack graph score - GET /rest/json/attack_graph/score	18
Get the topological attack graph - GET /rest/json/attack_graph/topological	18
Group Attack paths [/rest/json/attack_path]	18
Get the attack paths list - GET /rest/json/attack_path/list	18
Get the number of attack paths - GET /rest/json/attack_path/number	18
Get one attack path - GET /rest/json/attack_path/{id}	19
Get one attack path in topological form - GET /rest/json/attack_path/{id}/topological	19
Get the remediations to an attack path - GET /rest/json/attack_path/{id}/remediations	19
Simulate the remediation to an attack path - GET /rest/json/attack_path/{id}/remediation/{id_remediation}	19

Validate the remediation to an attack path - GET /rest/json/attack_path/{id}/remediation/{id_remediation}/validate	20
Get IDMEF alerts - GET /rest/json/idmef/alerts	20
<b>CyberCAPTOR-P2DS REST API</b>	<b>20</b>
<b>Peer Service [/peer]</b>	<b>20</b>
Add peer - POST /peer/{adminKey}	20
Delete peer - DELETE /peer/{peerName}/{adminKey}	21
List peers - GET /peers/{adminKey}	21
Add Input Data Set - POST /input/{registrationCode}	21
Add Input Data Sets - POST /inputs/{registrationCode}	22
Receive Message - POST /message/{recipient}/{sender}/{type}/{signature}	22
Start a peer - POST /start/{peerName}/{registrationCode}	22
Stop a peer - POST /stop/{peerName}/{registrationCode}	23
<b>Group Management Service [/group-mgmt]</b>	<b>23</b>
Verify Peer - POST /verify/{peerName}/{adminKey,verified}	23
Upload PublicKey - POST /publicKey/{peerName}/{registrationCode}	23
Get Configuration - GET /configuration/{peerName}/{registrationCode}	24
Get group information - GET /groupInfo/{peerName}/{registrationCode}	24
Get group - GET /group/{gid}/{adminKey}	25
Create group - POST /group/{adminKey}	25
Register a peer - POST /register/{registrationCode}/{url,name,type}	25
Generate registration code - POST /registration/{gid}/{adminKey}	26
Delete group - DELETE /group/{groupid}/{adminKey}	26
Delete peer - DELETE /peer/{peerName}/{adminKey}	26
Set configuration - POST /configuration/{adminKey}	27
Update peer status - POST /status/{peerName}/{registrationCode,status}	27
Get Peer - GET /peer/{peerName}/{adminKey}	28
Start peers - POST /start/{gid}/{adminKey}	28
Stop peers - POST /stop/{gid}/{adminKey}	28
Delete registration - DELETE /registration/{registrationCode}/{adminKey}	29
<b>Examples</b>	<b>29</b>
<b>CyberCAPTOR-Server REST API without initialization</b>	<b>29</b>
Group Version [/rest/version]	29
Version - GET /rest/version	29
VersionDetailed - GET /rest/version/detailed	29
Group Configuration [/rest/json/configuration]	30
Get global remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/global	30
Set global remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/global	30
Get snort rule remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/snort-rule	31
Set snort rule remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/snort-rule	31
Get firewall rule remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/firewall-rule	32
Set firewall rule remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/firewall-rule	32
Get patch remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/patch	33
Set patch remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/patch	33
<b>IDMEF</b>	<b>34</b>
Add IDMEF alerts [/rest/json/idmef/add]	34
Add IDMEF alerts - POST /rest/json/idmef/add	34
<b>CyberCAPTOR-Server REST API after initialization</b>	<b>35</b>
Initialize [/rest/json/initialize]	35
Initialize from data on disk - GET /rest/json/initialize	35
Initialize from XML topology - POST /rest/json/initialize	35
Group Get the XML topology [/rest/json/topology]	37
Get XML topology - GET /rest/json/topology	37
Group Hosts [/rest/json/host/list]	39
Get the host list - GET /rest/json/host/list	39
Set the host list - POST /rest/json/host/list	39
Group Attack graphs [/rest/json/attack_graph]	40
Get the attack graph - GET /rest/json/attack_graph	40
Get the attack graph score - GET /rest/json/attack_graph/score	40
Get the topological attack graph - GET /rest/json/attack_graph/topological	41
Group Attack paths [/rest/json/attack_path]	41
Get the attack paths list - GET /rest/json/attack_path/list	41
Get the number of attack paths - GET /rest/json/attack_path/number	41
Get one attack path - GET /rest/json/attack_path/{id}	42
Get one attack path in topological form - GET /rest/json/attack_path/{id}/topological	42
Get the remediations to an attack path - GET /rest/json/attack_path/{id}/remediations	42
Simulate the remediation to an attack path - GET /rest/json/attack_path/{id}/remediation/{id_remediation}	43
Validate the remediation to an attack path - GET /rest/json/attack_path/{id}/remediation/{id_remediation}/validate	43
Get IDMEF alerts - GET /rest/json/idmef/alerts	44
<b>CyberCAPTOR-P2DS REST API</b>	<b>44</b>
<b>Peer Service [/peer]</b>	<b>44</b>
Add peer - POST /peer/{adminKey}	44

Delete peer - DELETE /peer/{peerName}{?adminKey}	45
List peers - GET /peers{?adminKey}	46
Add Input Data Set - POST /input{?registrationCode}	47
Add Input Data Sets - POST /inputs{?registrationCode}	47
Receive Message - POST /message/{recipient}/{sender}/{type}{?signature}	48
Start a peer - POST /start/{peerName}{?registrationCode}	49
Stop a peer - POST /stop/{peerName}{?registrationCode}	49
Group Management Service [/group-mgmt]	50
Verify Peer - POST /verify/{peerName}{?adminKey,verified}	50
Upload PublicKey - POST /publicKey/{peerName}{?registrationCode}	51
Get Configuration - GET /configuration/{peerName}{?registrationCode}	52
Get group information - GET /groupInfo/{peerName}{?registrationCode}	52
Get group - GET /group/{gid}{?adminKey}	53
Create group - POST /group{?adminKey}	54
Register a peer - POST /register/{registrationCode}{?url,name,type}	55
Generate registration code - POST /registration/{gid}{?adminKey}	55
Delete group - DELETE /group/{groupId}{?adminKey}	56
Delete peer - DELETE /peer/{peerName}{?adminKey}	57
Set configuration - POST /configuration/{?adminKey}	57
Update peer status - POST /status/{peerName}{?registrationCode,status}	58
Get Peer - GET /peer/{peerName}{?adminKey}	59
Start peers - POST /start/{gid}{?adminKey}	60
Stop peers - POST /stop/{gid}{?adminKey}	60
Delete registration - DELETE /registration/{registrationCode}{?adminKey}	61

## References







# Specifications

This is the specification for Open API of the the FIWARE Cybersecurity Generic Enabler. First will be presented the API of the Cybersecurity GE that is used between the server (Cyber inputs generation, scored attack paths engine and remediation calculation) and the visualization client. Then will be presented the part of the Cybersecurity GE that is concerned with managing groups of peers for privacy-preserving data sharing, or P2DS. P2DS is achieved through secure multiparty computation, or SMCP.

## Introduction

### Scored Attack Paths

The attack graph engine is a Topological Vulnerability Analyser (TVA) able to generate, from the output of Cyber Data Extraction (the parts related to the network topology and vulnerabilities of the information system) the attack graph regrouping all possible attack paths. This engine also combines the attack graph with the Common Vulnerability Scoring System (CVSS), to provide a quantitative analysis of individual vulnerabilities.

The main attack paths of the attack graph provided by the Attack Graph Engine are processed by a scoring function that computes a score for each attack path based on its business impact and probability of occurrence. The score basically represents the risk level.

### Remediations

The remediation engine helps to mitigate the risks and to take efficient actions in accordance with the security policy by computing the different means to break the attack graph (so-called remediations) according to the AND/OR graph formalism and estimating a cost for each one. Once a security operator can actually select a specific attack path that he/she wants to prevent and get all the relevant alternatives of remediation.

### Privacy-Preserving Data Sharing

The main issue is the following: when organisations are asked to share data about security, they are naturally reluctant to do so, because revealing this data may lead to loss of trust or it may reveal details of the organisation's business that a competitor could use to its advantage. On the other hand, sharing data could be mutually beneficial. For example, when an organisation is the victim of a denial-of-service attack, it is useful to know whether other organisations are also a victim. This is where privacy-preserving data sharing comes in.

The technology for P2DS, called [SEPIA \(http://www.sepia.ee.ethz.ch/\)](http://www.sepia.ee.ethz.ch/) was developed as part of a PhD thesis at ETH Zurich.

In this API, there are several instances where a registration code is transported in the URL. This registration code is an authentication token; everyone with that authentication token can access the service. It is therefore a very, very good idea to use https on these calls, and probably https everywhere.

# Terminology

## Scored attack paths and remediations

The main terms used for the scored attack paths are:

- a **vulnerability scanner** automates the testing and discovery of services and known security weaknesses. For example, Nessus is a vulnerability scanner designed to automate the testing and discovery of known security issues.
- an **attack graph** is a directed graph containing all the attacks that are possible in an information system. It can be represented as a logical graph (AND/OR graph) describing the logical facts that are necessary to carry out an attack or as a topological graph describing which attacks can happen between hosts of the system. An attack graph is generally build thanks to the result of a vulnerability scanner.
- an **attack path** is an extract of an attack graph which focus an one/several/all ways to attack a specific target. Attack paths can be scored to be ranked and keep only the most important (likely or valuable) attack paths.
- a **remediation** is a way to prevent the execution of an attack path and protect its target. A remediation can be attached to an operational cost describing how it will cost effectively to an entreprise that wants to implement this remediation. Remediations can be, for example, the deployment of a firewall rule, of a Snort rule, or a patch.
- an **IDMEF** alert is a standardized alert that contains information about what has been detected, the sources of attack and targets of the attack. See <https://www.ietf.org/rfc/rfc4765.txt> for more information.

## Privacy-Preserving Data Sharing

Let's say we have three organisations, called Domain 1, Domain 2, and Domain 3 in the graphic, that want to know the total number of attacks seen in the last 24 hours, with a granularity of five minutes. In mathematical terms, what these organisations want is  $x_1 + x_2 + x_3$ , where  $x_1$ ,  $x_2$ , and  $x_3$  are vectors with  $24 \cdot 60 / 5 = 288$  elements, and they want to do this without revealing their own  $x_i$  to any of the other domains. Here is how the three domains could use P2DS for their needs.

First, each domain provides an input peer. This is a service that is run by each domain, which knows the original, private  $x_i$  from domain  $i$ .

Next, someone provides a number of privacy peers. These services can be run by anyone; they never have access to unencrypted data, so it doesn't matter who runs them. The only thing that matters is that the privacy peers are for the most part diligent, i.e., faithfully carry out their assigned task. The SEPIA protocol can tolerate a small number of malicious peers; only when more than this number of peers are malicious will the computation be deemed unsuccessful. The privacy peers execute a multi-round protocol in which they exchange encrypted information and perform computations on these encrypted values to get yet more encrypted values. No one learns the cleartext values of these encrypted vectors, not the privacy peers, not the domains.

But when the computation is finished, the end result becomes available in the

clear and each domain can learn the value of  $x_1 + x_2 + x_3$ . For example, Domain 1 learns the value of  $x_1 + x_2 + x_3$ , but knows nothing about  $x_2$  or  $x_3$ , except, trivially, their sum.

In our GE, the privacy peers are provided by the domains.

A final component of our contribution is the group manager. This is the service that knows which input peers and which privacy peers should cooperate in a computation, which keeps the peers' public key certificates, and which provides SEPIA configuration when it is time to start the computation. It does not have to be especially trusted (none of the data it has is particularly secret) but it must be authentic, in the sense that the data it keeps should be protected against unauthorised alteration.

There are a few caveats in using SEPIA:

- When there are only two input peers and they compute a sum, each peer can compute the contribution of the other peer through a simple subtraction: If I know  $x$  and  $x + y$ , I can compute  $y$  as  $(x + y) - x$ .
- In general, when there are  $n \geq 3$  input peers and  $n - 1$  of them collude to defraud the remaining one, they can simply exchange their own input vectors through a side channel.
- In general, SEPIA is secure in what is known as the honest but curious adversary model, in which adversaries try to learn the contents of messages but will not actively try to disrupt the protocol.

## Concepts

Please refer to the Introduction for the concepts used in this specification.



# API Specification

## CyberCAPTOR-Server REST API without initialization

This group of REST calls contains the API calls that **do not need** the `/initialize` call that loads the vulnerability and remediation database and generates the attack graph and the attack paths.

### Group Version [/rest/version]

Get REST API version information. Generally useful to test that the installation is working.

#### *Version - GET /rest/version*

Get the simple version of the API.

**Response 200** (text/plain)

[Go to example](#)

#### *VersionDetailed - GET /rest/version/detailed*

Get the API version in JSON.

**Response 200** (application/json)

[Go to example](#)

### Group Configuration [/rest/json/configuration]

This group contains the calls related to the configuration (remediation cost parameters...).

#### *Get global remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/global*

Get the global remediation cost parameters.

**Response 200** (application/json)

[Go to example](#)

#### *Set global remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/global*

Set the global remediation cost parameters.

**Request** (application/json)

**Response 200** (application/json)

[Go to example](#)

**Get snort rule remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/snort-rule**

Get the operational cost parameters for a snort rule.

**Response 200** (application/json)

[Go to example](#)

**Set snort rule remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/snort-rule**

Set the operational cost parameters for a snort rule.

**Request** (application/json)

**Response 200** (application/json)

[Go to example](#)

**Get firewall rule remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/firewall-rule**

Get the operational cost parameters for a firewall rule.

**Response 200** (application/json)

[Go to example](#)

**Set firewall rule remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/firewall-rule**

Set the operational cost parameters for a firewall rule.

**Request** (application/json)

**Response 200** (application/json)

[Go to example](#)

### *Get patch remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/patch*

Get the operational cost parameters for a patch.

**Response 200** (application/json)

[Go to example](#)

### *Set patch remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/patch*

Set the operational cost parameters for a patch.

**Request** (application/json)

**Response 200** (application/json)

[Go to example](#)

## IDMEF

REST API calls related to IDMEF alerts. See <https://www.ietf.org/rfc/rfc4765.txt> for more IDMEF alerts information.

### Add IDMEF alerts [/rest/json/idmef/add]

#### *Add IDMEF alerts - POST /rest/json/idmef/add*

From an XML IDMEF file containing alerts.

**Request** (application/xml)

**Response 200** (application/json)

[Go to example](#)

## CyberCAPTOR-Server REST API after initialization

This group contains the API calls **after** the `/initialize` call that loads the vulnerability and remediation database and generates the attack graph and the attack paths.

## Initialize [/rest/json/initialize]

Generates the attack graph and initializes the main objects needed by other API calls (database, attack graph, attack paths,...).

### **Initialize from data on disk - GET /rest/json/initialize**

From the data on disk (.csv inputs files and Nessus vulnerability scan)

**Response 200** (application/json)

[Go to example](#)

### **Initialize from XML topology - POST /rest/json/initialize**

From an XML topology file containing all information about network topology, firewalling, routing configuration, vulnerabilities...

**Request** (application/xml)

**Response 200** (application/json)

[Go to example](#)

## Group Get the XML topology [/rest/json/topology]

Get the XML topology (for example, this can be used to backup the topology, and to load it again with /initialize)

### **Get XML topology - GET /rest/json/topology**

Get the XML topology for backup

**Response 200** (application/xml)

[Go to example](#)



## Group Hosts [/rest/json/host/list]

This group contains the calls related to hosts, after initialization.

### *Get the host list - GET /rest/json/host/list*

Get the list of hosts with their security requirements.

**Response 200** (application/json)

[Go to example](#)

### *Set the host list - POST /rest/json/host/list*

Set the hosts and their security\_requirements.

**Request** (application/json)

**Response 200** (application/json)

[Go to example](#)

## Group Attack graphs [/rest/json/attack\_graph]

This group contains the calls related to the attack graph, after initialization.

### *Get the attack graph - GET /rest/json/attack\_graph*

Get the whole attack graph.

**Response 200** (application/json)

[Go to example](#)

### *Get the attack graph score - GET /rest/json/attack\_graph/score*

Get the attack graph score.

**Response 200** (application/json)

[Go to example](#)

### *Get the topological attack graph - GET /rest/json/attack\_graph/topological*

Get the attack graph in its topological form.

**Response 200** (application/json)

[Go to example](#)

## Group Attack paths [/rest/json/attack\_path]

This group contains the calls related to the attack paths, after initialization.

### *Get the attack paths list - GET /rest/json/attack\_path/list*

Get the list of attack paths.

**Response 200** (application/json)

[Go to example](#)

### *Get the number of attack paths - GET /rest/json/attack\_path/number*

Get the total number of attack paths.

**Response 200** (application/json)

[Go to example](#)

### **Get one attack path - GET /rest/json/attack\_path/{id}**

Get the attack path {id}.

#### **Parameters**

##### **id (Required, number )**

The number of attack path to get

**Response 200** (application/json)

[Go to example](#)

### **Get one attack path in topological form - GET /rest/json/attack\_path/{id}/topological**

Get the attack path {id} as a topological graph.

#### **Parameters**

##### **id (Required, number )**

The number of attack path to get in topological form

**Response 200** (application/json)

[Go to example](#)

### **Get the remediations to an attack path - GET /rest/json/attack\_path/{id}/remediations**

Get the remediations of the attack path {id}.

#### **Parameters**

##### **id (Required, number )**

The number of the attack path for which remediations will be calculated

**Response 200** (application/json)

[Go to example](#)

### **Simulate the remediation to an attack path - GET /rest/json/attack\_path/{id}/remediation/{id\_remediation}**

Simulate the remediation {id\_remediation} of the path {id}, and compute the new attack graph.

#### **Parameters**

##### **id (Required, number )**

The number of the attack path for which remediations will be calculated

##### **id\_remediation (Required, number )**

The number of the remediation to apply.

**Response 200** (application/json)

[Go to example](#)

**Validate the remediation to an attack path - GET**  
**/rest/json/attack\_path/{id}/remediation/{id\_remediation}/validate**

Validate that the remediation {id\_remediation} of the path {id} has been applied.

**Parameters**

**id (Required, number )**

The number of the attack path for which remediations will be calculated

**id\_remediation (Required, number )**

The number of the remediation to validate.

**Response 200** (application/json)

[Go to example](#)

**Get IDMEF alerts - GET /rest/json/idmef/alerts**

Get the IDMEF alerts that have been received by the server, and not yet sent to this client, and their potential dynamic remediations that could prevent the described attack.

**Response 200** (application/json)

[Go to example](#)

## CyberCAPTOR-P2DS REST API

### Peer Service [/peer]

**Add peer - POST /peer/{?adminKey}**

Add a peer to the service and register it at the group management service.

**Parameters**

**adminKey (Required, string )**

Admin key

**Request** (application/json)

**Response 200** (application/json)

**Response 403** (text/plain)

[Go to example](#)

### Delete peer - DELETE /peer/{peerName}/{?adminKey}

Delete a peer.

#### Parameters

**adminKey (Required, string )**

Admin key

**peerName (Required, string )**

Name of the peer.

**Response 200** (text/plain)

**Response 403** (text/plain)

**Response 404** (text/plain)

[Go to example](#)

### List peers - GET /peers{?adminKey}

List all peers.

#### Parameters

**adminKey (Required, string )**

Admin key

**Response 200** (application/json)

**Response 403** (text/plain)

[Go to example](#)

### Add Input Data Set - POST /input{?registrationCode}

This method can be used to add a single input data set. The target peer must be an input peer.

#### Parameters

**registrationCode (Required, string )**

Registration code

**Request** (application/json)

**Response 200** (text/plain)

**Response 400** (text/plain)

[Go to example](#)

### **Add Input Data Sets - POST /inputs/{?registrationCode}**

This method can be used to add multiple input data sets. The target peer must be an input peer.

#### **Parameters**

##### **registrationCode (Required, string )**

Registration code

**Request** (application/json)

**Response 200** (text/plain)

**Response 400** (text/plain)

[Go to example](#)

### **Receive Message - POST /message/{recipient}/{sender}/{type}{?signature}**

This method is called by the peer services automatically. This method will accept messages and verify their integrity.

#### **Parameters**

##### **recipient (Required, string )**

Name of the recipient

##### **sender (Required, string )**

Name of the sender

##### **type (Required, string )**

Type of the message. The types varies depending on the selected mpc protocol.

##### **signature (Required, string )**

Signature bytes base64-encoded.

**Request** (text/plain)

**Response 200** (text/plain)

**Response 404** (text/plain)

[Go to example](#)

### **Start a peer - POST /start/{peerName}{?registrationCode}**

Can be manually invoked or by the group management service. Starts the peer.

#### **Parameters**

##### **peerName (Required, string )**

Name of the peer

##### **registrationCode (Required, string )**

Registration code

**Response 200** (application/json)

**Response 400** (text/plain)

**Response 404** (text/plain)

[Go to example](#)

#### **Stop a peer - POST /stop/{peerName}/{?registrationCode}**

Can be manually invoked or by the group management service. Stops the peer.

##### **Parameters**

**peerName (Required, string )**

Name of the peer

**registrationCode (Required, string )**

Registration code

**Response 200** (text/plain)

**Response 400** (text/plain)

[Go to example](#)

## Group Management Service [/group-mgmt]

#### **Verify Peer - POST /verify/{peerName}/{?adminKey,verified}**

This method sets the verified flag of a peer.

##### **Parameters**

**peerName (Required, string )**

Name of the peer.

**adminKey (Required, string )**

Admin key

**verified (Required, boolean )**

Value of the verified flag (true or false).

**Response 200** (application/json)

**Response 403** (text/plain)

**Response 404** (text/plain)

[Go to example](#)

#### **Upload PublicKey - POST /publicKey/{peerName}/{?registrationCode}**

This method can be used to upload the public key of a peer.

### Parameters

**peerName (Required, string )**

Name of the Peer.

**registrationCode (Required, string )**

Registration code

**Request** (text/plain)

**Response 200** (application/json)

**Response 400** (text/plain)

**Response 404** (text/plain)

[Go to example](#)

### Get Configuration - GET /configuration/{peerName}{{?registrationCode}}

This method can be used to download the current group's configuration the peer is member of.

### Parameters

**peerName (Required, number )**

Name of the peer

**registrationCode (Required, string )**

Registration code

**Response 200** (application/json)

**Response 400** (text/plain)

**Response 404** (text/plain)

[Go to example](#)

### Get group information - GET /groupInfo/{peerName}{{?registrationCode}}

This method can be used to obtain information about the current group the peer is member of.

### Parameters

**peerName (Required, number )**

Name of the peer

**registrationCode (Required, string )**

Registration code

**Response 200** (application/json)

**Response 404** (text/plain)



**Response 400** (text/plain)

[Go to example](#)

#### **Get group - GET /group/{gid}/{?adminKey}**

This method can be used to obtain information about a specific group.

##### **Parameters**

**gid (Required, number )**

ID of the group

**adminKey (Required, string )**

Admin key

**Response 200** (application/json)

**Response 404** (text/plain)

[Go to example](#)

#### **Create group - POST /group{?adminKey}**

This method can be used to create a group.

##### **Parameters**

**adminKey (Required, string )**

Admin key

**Request** (application/json)

**Response 200** (application/json)

**Response 403** (text/plain)

[Go to example](#)

#### **Register a peer - POST /register/{registrationCode}/{?url,name,type}**

Used to register a peer (this method will be called by the peer services automatically).

##### **Parameters**

**registrationCode (Required, string )**

Registration code

**url (Required)**

input-peer/peer (required, string) - URL the peer can be reached at

**type (Required, number )**

Type of the peer (1 = input, 2 = privacy)

**name (Required, string )**

Name of the peer.

**Response 200** (application/json)

**Response 400** (text/plain)

[Go to example](#)

#### **Generate registration code - POST /registration/{gid}/{?adminKey}**

Generate a registration code.

##### **Parameters**

**gid (Required, number )**

ID of the group

**adminKey (Required, string )**

Admin key

**Response 200** (application/json)

**Response 404** (text/plain)

**Response 403** (text/plain)

[Go to example](#)

#### **Delete group - DELETE /group/{groupId}/{?adminKey}**

Delete a group.

##### **Parameters**

**groupId (Required, number )**

ID of the group

**adminKey (Required, string )**

Admin key

**Response 200** (text/plain)

**Response 404** (text/plain)

**Response 403** (text/plain)

[Go to example](#)

#### **Delete peer - DELETE /peer/{peerName}/{?adminKey}**

Delete a peer.

##### **Parameters**

**peerName (Required, number )**

Name of the peer.

**adminKey (Required, string )**

Admin key

**Response 200** (text/plain)

**Response 404** (text/plain)

**Response 403** (text/plain)

[Go to example](#)

### **Set configuration - POST /configuration/{?adminKey}**

Set the configuration for a group.

#### **Parameters**

**adminKey (Required, string )**

Admin key

**Request** (application/json)

**Response 200** (application/json)

**Response 404** (text/plain)

**Response 403** (text/plain)

[Go to example](#)

### **Update peer status - POST /status/{peerName}{?registrationCode,status}**

Update the status of a peer (the peer services will call this method automatically).

#### **Parameters**

**peerName (Required, string )**

Name of the peer.

**registrationCode (Required, string )**

Registration code

**status (Required, number )**

Status (1 = started, 2 = error, 3 = stopped, 0 = unknown)

**Response 200** (text/plain)

**Response 400** (text/plain)

**Response 404** (text/plain)

[Go to example](#)

### **Get Peer - GET /peer/{peerName}?adminKey**

Get a peer (including registration code).

#### **Parameters**

**peerName (Required, string )**

Name of the peer.

**adminKey (Required, string )**

Admin key

**Response 200** (application/json)

**Response 403** (text/plain)

**Response 404** (text/plain)

[Go to example](#)

### **Start peers - POST /start/{gid}?adminKey**

Starts all peers member of a group. This method will not start unverified peers.

#### **Parameters**

**gid (Required, number )**

ID of the group

**adminKey (Required, string )**

Admin key

**Response 200** (text/plain)

**Response 403** (text/plain)

[Go to example](#)

### **Stop peers - POST /stop/{gid}?adminKey**

Stops all peers member of a group.

#### **Parameters**

**gid (Required, number )**

ID of the group

**adminKey (Required, string )**

Admin key

**Response 200** (text/plain)

**Response 403** (text/plain)

[Go to example](#)

### Delete registration - DELETE /registration/{registrationCode}/{adminKey}

Delete a registration (code).

#### Parameters

**registrationCode (Required, string )**

Registration code

**adminKey (Required, string )**

Admin key

**Response 200** (text/plain)

**Response 403** (text/plain)

[Go to example](#)

## Examples

### CyberCAPTOR-Server REST API without initialization

#### Group Version [/rest/version]

##### Version - GET /rest/version

**Response 200** (text/plain)

Headers

**Content-Type:** text/plain

Body

4.4

[Go to specification](#)

##### VersionDetailed - GET /rest/version/detailed

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"version": "4.4"}
```

[Go to specification](#)

## Group Configuration [/rest/json/configuration]

*Get global remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/global*

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"global_parameters": {}}
```

[Go to specification](#)

*Set global remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/global*

**Request** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"global_parameters": {"expensesForIT": 15000}}
```

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{}
```

[Go to specification](#)

**Get snort rule remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/snort-rule**

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"operational_cost_parameters":{}}
```

[Go to specification](#)

**Set snort rule remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/snort-rule**

**Request** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"operational_cost_parameters":{"computationPowerCost":12,"skillRateMaintenance":1,"restartDuration":0.2,"usedStorage":1,"storageCost":5,"skillRateTests":0.7,"deploymentDuration":0.5,"businessApplication sTestsDuration":4,"maintenanceDuration":10,"remediationCost":10,"remediationUninstallDuration":0.5,"usedPower":1,"serviceUnavailabilityDeploymentDuration":0,"skillRateDeployment":2,"workCost":20,"restartCost":0}}
```

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

}

[Go to specification](#)

**Get firewall rule remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/firewall-rule**

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"operational_cost_parameters":{}}
```

[Go to specification](#)

**Set firewall rule remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/firewall-rule**

**Request** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"operational_cost_parameters":{"computationPowerCost":12,"skillRateMaintenance":1,"restartDuration":0.5,"usedStorage":0,"storageCost":10,"skillRateTests":1,"deploymentDuration":0.5,"businessApplicationsTestsDuration":6,"maintenanceDuration":0,"remediationCost":0,"remediationUninstallDuration":0.1,"usedPower":0.1,"serviceUnavailabilityDeploymentDuration":0,"skillRateDeployment":1.2,"workCost":20,"restartCost":10}}
```

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body



```
{}
```

[Go to specification](#)

**Get patch remediation cost parameters - GET /rest/json/configuration/remediation-cost-parameters/patch**

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"operational_cost_parameters":{}}
```

[Go to specification](#)

**Set patch remediation cost parameters - POST /rest/json/configuration/remediation-cost-parameters/patch**

**Request** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"operational_cost_parameters":{"computationPowerCost":5,"skillRateMaintenance":1,"restartDuration":0.5,"usedStorage":0,"storageCost":3,"skillRateTests":0.7,"deploymentDuration":3,"businessApplicationsTestsDuration":4,"maintenanceDuration":0.5,"remediationCost":5,"remediationUninstallDuration":1,"usedPower":0,"serviceUnavailabilityDeploymentDuration":0.5,"skillRateDeployment":1.5,"workCost":20,"restartCost":10}}
```

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

}

[Go to specification](#)

## IDMEF

Add IDMEF alerts [/rest/json/idmef/add]

*Add IDMEF alerts - POST /rest/json/idmef/add*

**Request** (application/xml)

Headers

Content-Type: application/xml

Body

```
<?xml version="1.0" encoding="UTF-8"?>
<idmef:IDMEF-Message xmlns:idmef="http://iana.org/idmef" version="1.0">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="bc-sensor01">
      <idmef:Node category="dns">
        <idmef:name>sensor.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc71f4f5.0xef449129">2000-03-09T10:01:25.93464Z</idmef:CreateTime>
    <idmef:Source ident="a1a2" spoofed="yes">
      <idmef:Node ident="a1a2-1">
        <idmef:Address ident="a1a2-2" category="ipv4-addr">
          <idmef:address>192.0.2.200</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Source>
    <idmef:Target ident="b3b4">
      <idmef:Node>
        <idmef:Address ident="b3b4-1" category="ipv4-addr">
          <idmef:address>192.0.2.50</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Target>
    <idmef:Target ident="c5c6">
      <idmef:Node ident="c5c6-1" category="nisplus">
        <idmef:name>lollipop</idmef:name>
      </idmef:Node>
    </idmef:Target>
    <idmef:Target ident="d7d8">
      <idmef:Node ident="d7d8-1">
```

```
<idmef:location>Cabinet B10</idmef:location>
<idmef:name>Cisco.router.b10</idmef:name>
</idmef:Node>
</idmef:Target>
<idmef:Classification text="Ping-of-death detected">
  <idmef:Reference origin="cve">
    <idmef:name>CVE-1999-128</idmef:name>
    <idmef:url>http://www.cve.mitre.org/cgi-bin/cvename.cgi?name
=CVE-1999-128</idmef:url>
  </idmef:Reference>
</idmef:Classification>
</idmef:Alert>
</idmef:IDMEF-Message>
```

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"success": "IDMEF alerts added successfully"}
```

[Go to specification](#)

## CyberCAPTOR-Server REST API after initialization

Initialize [/rest/json/initialize]

*Initialize from data on disk - GET /rest/json/initialize*

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"status": "Loaded"}
```

[Go to specification](#)

*Initialize from XML topology - POST /rest/json/initialize*

## Request (application/xml)

### Headers

Content-Type: application/xml

### Body

```
<topology>
  <machine>
    <name>linux-user-1</name>
    <security_requirement>7</security_requirement>
    <interfaces>
      <interface>
        <name>eth0</name>
        <ipaddress>192.168.1.111</ipaddress>
        <vlan>
          <name>user-lan</name>
          <label>user-lan</label>
        </vlan>
      </interface>
    </interfaces>
    <routes>
      <route>
        <destination>0.0.0.0</destination>
        <mask>0.0.0.0</mask>
        <gateway>192.168.1.111</gateway>
        <interface>eth0</interface>
      </route>
    </routes>
  </machine>
  <machine>
    <name>linux-user-2</name>
    <security_requirement>30</security_requirement>
    <interfaces>
      <interface>
        <name>eth0</name>
        <ipaddress>192.168.1.112</ipaddress>
        <vlan>
          <name>user-lan</name>
          <label>user-lan</label>
        </vlan>
      </interface>
    </interfaces>
    <services>
      <service>
        <name>mdns</name>
        <ipaddress>192.168.1.112</ipaddress>
        <protocol>udp</protocol>
        <port>5353</port>
      </service>
    </services>
    <vulnerabilities>
      <vulnerability>
        <type>remoteExploit</type>
        <cve>CVE-2007-2446</cve>
        <goal>privEscalation</goal>
      </vulnerability>
    </vulnerabilities>
  </machine>
</topology>
```

```
<cvss>10.0</cvss>
</vulnerability>
</vulnerabilities>
</service>
</services>
<routes>
<route>
<destination>0.0.0.0</destination>
<mask>0.0.0.0</mask>
<gateway>192.168.1.111</gateway>
<interface>eth0</interface>
</route>
</routes>
</machine>
</topology>
```

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"status": "Loaded"}
```

[Go to specification](#)

Group Get the XML topology [/rest/json/topology]

**Get XML topology - GET /rest/json/topology**

**Response 200** (application/xml)

Headers

**Content-Type:** application/xml  
**Content-Length:** 2174

Body

```
<topology>
  <machine>
    <name>linux-user-1</name>
    <cpe>cpe:/</cpe>
    <interfaces>
      <interface>
        <name>eth0</name>
```

```

    <vlan>
      <name>user-lan</name>
      <label>user-lan</label>
    </vlan>
    <ipaddress>192.168.1.111</ipaddress>
    <directly-connected>
      <ipaddress>192.168.1.112</ipaddress>
    </directly-connected>
  </interface>
</interfaces>
<services />
<routes>
  <route>
    <destination>0.0.0.0</destination>
    <mask>0.0.0.0</mask>
    <gateway>192.168.1.111</gateway>
    <interface>eth0</interface>
  </route>
</routes>
<input-firewall>
  <default-policy>ACCEPT</default-policy>
</input-firewall>
<output-firewall>
  <default-policy>ACCEPT</default-policy>
</output-firewall>
</machine>
<machine>
  <name>linux-user-2</name>
  <cpe>cpe:/</cpe>
  <interfaces>
    <interface>
      <name>eth0</name>
      <vlan>
        <name>user-lan</name>
        <label>user-lan</label>
      </vlan>
      <ipaddress>192.168.1.112</ipaddress>
      <directly-connected>
        <ipaddress>192.168.1.111</ipaddress>
      </directly-connected>
    </interface>
  </interfaces>
  <services>
    <service>
      <name>mdns</name>
      <ipaddress>192.168.1.112</ipaddress>
      <protocol>TCP</protocol>
      <port>5353</port>
      <CPE>cpe:/</CPE>
      <vulnerabilities>
        <vulnerability>
          <type>remoteExploit</type>
          <goal>privEscalation</goal>
          <cve>CVE-2007-2446</cve>
        </vulnerability>
      </vulnerabilities>
    </service>
  </services>

```

```
</service>
</services>
<routes>
  <route>
    <destination>0.0.0.0</destination>
    <mask>0.0.0.0</mask>
    <gateway>192.168.1.111</gateway>
    <interface>eth0</interface>
  </route>
</routes>
<input-firewall>
  <default-policy>ACCEPT</default-policy>
</input-firewall>
<output-firewall>
  <default-policy>ACCEPT</default-policy>
</output-firewall>
</machine>
</topology>
```

[Go to specification](#)

## Group Hosts [/rest/json/host/list]

### *Get the host list - GET /rest/json/host/list*

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"hosts": []}
```

[Go to specification](#)

### *Set the host list - POST /rest/json/host/list*

**Request** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"hosts": [{"security_requirements": [{"metric": "High", "name": "sec-req-xml"}], "name": "linux-user-1"}, {"security_requirements": [{"metric": "High", "name": "sec-req-xml"}], "name": "linux-user-2"}]}
```

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{}
```

[Go to specification](#)

Group Attack graphs [/rest/json/attack\_graph]

**Get the attack graph - GET /rest/json/attack\_graph**

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"attack_graph": {"arcs": {}, "vertices": {}}}
```

[Go to specification](#)

**Get the attack graph score - GET /rest/json/attack\_graph/score**

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"score": ""}
```



[Go to specification](#)

*Get the topological attack graph - GET /rest/json/attack\_graph/topological*

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"arcs": {}, "vertices": {}}
```

[Go to specification](#)

Group Attack paths [/rest/json/attack\_path]

*Get the attack paths list - GET /rest/json/attack\_path/list*

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"attack_paths": {}}
```

[Go to specification](#)

*Get the number of attack paths - GET /rest/json/attack\_path/number*

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"number": 2}
```

[Go to specification](#)

*Get one attack path - GET /rest/json/attack\_path/{id}*

**Parameters**

**id (Required, number )**

The number of attack path to get

**Response 200** (application/json)

Payload

**id (Not required, None)**

Headers

**Content-Type:** application/json

Body

```
{"attack_path": {}}
```

[Go to specification](#)

*Get one attack path in topological form - GET /rest/json/attack\_path/{id}/topological*

**Parameters**

**id (Required, number )**

The number of attack path to get in topological form

**Response 200** (application/json)

Payload

**id (Not required, None)**

Headers

**Content-Type:** application/json

Body

```
{"arcs": {}, "vertices": {}}
```

[Go to specification](#)

*Get the remediations to an attack path - GET /rest/json/attack\_path/{id}/remediations*

## Parameters

### id (Required, number )

The number of the attack path for which remediations will be calculated

**Response 200** (application/json)

Payload

### id (Not required, None)

Headers

Content-Type: application/json

Body

```
{"remediations":{}}
```

[Go to specification](#)

*Simulate the remediation to an attack path - GET*  
*/rest/json/attack\_path/{id}/remediation/{id\_remediation}*

## Parameters

### id (Required, number )

The number of the attack path for which remediations will be calculated

### id\_remediation (Required, number )

The number of the remediation to apply.

**Response 200** (application/json)

Payload

### id (Not required, None)

### id\_remediation (Not required, None)

Headers

Content-Type: application/json

Body

```
{"attack_graph":{"arcs":{},"vertices":{}}
```

[Go to specification](#)

*Validate the remediation to an attack path - GET*  
*/rest/json/attack\_path/{id}/remediation/{id\_remediation}/validate*

## Parameters

### id (Required, number )

The number of the attack path for which remediations will be calculated

### id\_remediation (Required, number )

The number of the remediation to validate.

### Response 200 (application/json)

#### Payload

### id (Not required, None)

### id\_remediation (Not required, None)

#### Headers

**Content-Type:** application/json

#### Body

```
{"success": "The remediation has been validated."}
```

[Go to specification](#)

### Get IDMEF alerts - GET /rest/json/idmef/alerts

### Response 200 (application/json)

#### Headers

**Content-Type:** application/json

#### Body

```
{"alerts": []}
```

[Go to specification](#)

## CyberCAPTOR-P2DS REST API

### Peer Service [/peer]

### Add peer - POST /peer{?adminKey}

## Parameters

### adminKey (Required, string )

Admin key

### **Request** (application/json)

#### Headers

**Content-Type:** application/json

#### Body

```
{ "finalResultsURL": "http://localhost:12001/p2ds-receiver/demo/receive", "peerType": 1, "name": "peerhans", "privateKey": "MFECAQAwEAYHkoZIZj0CAQYFK4EEACQEOjA4AgEBBDNyjBeP85atxkIfiYqW+0kUB2H3guXcQWXT/tXVktbn3MyUdRmNIL99G3rK1XoGSRAM6js=", "publicKey": "MH4wEAYHkoZIZj0CAQYFK4EEACQDagAEAjig6xXX4SuME5lRB2ADn7T7CgyH7LXbxy/oS5XhIElBPwz/40cwDac/VgGbDKa+HGBc/AGzwSlScoCDHc7WA1tSkRUkaW/LL9NbA6gIzJLMw+FV3RPor0vpJIofVcAaV6WI1r99v8Y=", "registrationCode": "TEST", "groupMgmtURL": "http://localhost:12001/p2ds-group-management/group-mgmt" }
```

### **Response 200** (application/json)

#### Headers

**Content-Type:** application/json

#### Body

```
{ "finalResultsURL": "http://localhost:12001/p2ds-receiver/demo/receive", "peerType": 1, "name": "peerhans", "privateKey": "MFECAQAwEAYHkoZIZj0CAQYFK4EEACQEOjA4AgEBBDNyjBeP85atxkIfiYqW+0kUB2H3guXcQWXT/tXVktbn3MyUdRmNIL99G3rK1XoGSRAM6js=", "publicKey": "MH4wEAYHkoZIZj0CAQYFK4EEACQDagAEAjig6xXX4SuME5lRB2ADn7T7CgyH7LXbxy/oS5XhIElBPwz/40cwDac/VgGbDKa+HGBc/AGzwSlScoCDHc7WA1tSkRUkaW/LL9NbA6gIzJLMw+FV3RPor0vpJIofVcAaV6WI1r99v8Y=", "registrationCode": "TEST", "groupMgmtURL": "http://localhost:12001/p2ds-group-management/group-mgmt" }
```

### **Response 403** (text/plain)

#### Headers

**Content-Type:** text/plain

[Go to specification](#)

### Delete peer - DELETE /peer/{peerName}/{adminKey}

#### Parameters

**adminKey** (Required, string )

Admin key

**peerName** (Required, string )

Name of the peer.

**Response 200** (text/plain)

Headers

**Content-Type:** text/plain

**Response 403** (text/plain)

Headers

**Content-Type:** text/plain

**Response 404** (text/plain)

Headers

**Content-Type:** text/plain

[Go to specification](#)

**List peers - GET /peers{?adminKey}**

**Parameters**

**adminKey (Required, string )**

Admin key

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"peers": ["name": "peerhans", "privateKey": "MFECAQAwEAYHKOZIZj0CAQYF  
K4EEACQE0jA4AgEBBDNyjBeP85atxkIfiYqW+0kUB2H3guXcQWXT/tXVktbn3MyUdRmN  
IL99G3rK1XoGSRAM6js=", "publicKey": "MH4wEAYHKOZIZj0CAQYFK4EEACQDagAE  
AJig6xXX4SuME5lRB2ADn7T7CgyH7LXbxy/oS5XhIElBPwz/40cwDac/VgGbDKa+HGBc  
/AGzwSlScoCDHc7WA1tSkRUkaW/LL9NbA6gIzJLMw+FV3RPor0vpJIofVcAaV6WI1r99  
v8Y=", "registrationCode": "TEST", "groupMgmtURL": "http://localhost:1  
2001/p2ds-group-management/group-mgmt" ] ] }
```

**Response 403** (text/plain)

Headers

Content-Type: `text/plain`

[Go to specification](#)

### Add Input Data Set - POST /input{?registrationCode}

#### Parameters

**registrationCode (Required, string )**

Registration code

**Request** (application/json)

Headers

Content-Type: `application/json`

Body

```
{"peerName": "peerhans", "data": "3;4"}
```

**Response 200** (text/plain)

Headers

Content-Type: `text/plain`

**Response 400** (text/plain)

Headers

Content-Type: `text/plain`

[Go to specification](#)

### Add Input Data Sets - POST /inputs{?registrationCode}

#### Parameters

**registrationCode (Required, string )**

Registration code

**Request** (application/json)

Headers

Content-Type: `application/json`

## Body

```
{"peerName": "peerhans", "data": ["3;4", "1;1"]}
```

## **Response 200** (text/plain)

### Headers

**Content-Type:** text/plain

## **Response 400** (text/plain)

### Headers

**Content-Type:** text/plain

[Go to specification](#)

## **Receive Message - POST /message/{recipient}/{sender}/{type}{?signature}**

### Parameters

#### **recipient (Required, string )**

Name of the recipient

#### **sender (Required, string )**

Name of the sender

#### **type (Required, string )**

Type of the message. The types varies depending on the selected mpc protocol.

#### **signature (Required, string )**

Signature bytes base64-encoded.

## **Request** (text/plain)

### Headers

**Content-Type:** text/plain

## Body

The data of the message as JSON.

## **Response 200** (text/plain)

### Headers

**Content-Type:** text/plain



## **Response 404** (text/plain)

### Headers

**Content-Type:** text/plain

[Go to specification](#)

## **Start a peer - POST /start/{peerName}/{?registrationCode}**

### Parameters

#### **peerName (Required, string )**

Name of the peer

#### **registrationCode (Required, string )**

Registration code

## **Response 200** (application/json)

### Headers

**Content-Type:** application/json

### Body

```
{ "gid": 1, "lastStatus": 0, "peerName": "peerhans", "peerType": 1, "publicKey": "MH4wEAYHKoZIzj0CAQYFK4EEACQDagAEAJig6xXX4SuME5lRB2ADn7T7CgyH7LXbxy/oS5XhIElBPwz/40cwDac/VgGbDKa+HGBc/AGzwSlScoCDHc7WA1tSkRUkaW/LL9NbA6gIzJLMw+FV3RPor0vpJIofVcAaV6WI1r99v8Y=", "url": "https://localhost:12001/p2ds-input-peer/peer" }
```

## **Response 400** (text/plain)

### Headers

**Content-Type:** text/plain

## **Response 404** (text/plain)

### Headers

**Content-Type:** text/plain

[Go to specification](#)

## **Stop a peer - POST /stop/{peerName}/{?registrationCode}**

### Parameters

**peerName (Required, string )**

Name of the peer

**registrationCode (Required, string )**

Registration code

**Response 200** (text/plain)

Headers

**Content-Type:** text/plain

**Response 400** (text/plain)

Headers

**Content-Type:** text/plain

[Go to specification](#)

## Group Management Service [/group-mgmt]

**Verify Peer - POST /verify/{peerName}{?adminKey,verified}**

**Parameters**

**peerName (Required, string )**

Name of the peer.

**adminKey (Required, string )**

Admin key

**verified (Required, boolean )**

Value of the verified flag (true or false).

**Response 200** (application/json)

Headers

**Content-Type:** application/json

**Response 403** (text/plain)

Headers

**Content-Type:** text/plain

**Response 404** (text/plain)

Headers

**Content-Type:** text/plain

[Go to specification](#)

### Upload PublicKey - POST /publicKey/{peerName}?registrationCode}

#### Parameters

##### **peerName (Required, string )**

Name of the Peer.

##### **registrationCode (Required, string )**

Registration code

#### **Request** (text/plain)

##### Headers

**Content-Type:** text/plain

##### Body

You need to upload the key as text/plain. The key needs to be transmitted as base64-encoded.

#### **Response 200** (application/json)

##### Headers

**Content-Type:** application/json

##### Body

```
{ "gid": 1, "lastStatus": 0, "peerName": "hanspeer", "peerType": 1, "publicKey": "MH4wEAYHKOZIZj0CAQYFK4EEACQDagAEAjig6xXX4SuME5lRB2ADn7T7CgyH7LXbxy/oS5XhIElBPwz/40cwDAc/VgGbDKa+HGBc/AGzwSlScoCDHc7WA1tSkRUkaW/LL9NbA6gIzJLMw+FV3RPor0vpJIofVcAaV6WI1r99v8Y=", "url": "https://localhost:12001/p2ds-input-peer/peer" }
```

#### **Response 400** (text/plain)

##### Headers

**Content-Type:** text/plain

#### **Response 404** (text/plain)

##### Headers

Content-Type: `text/plain`

[Go to specification](#)

**Get Configuration - GET /configuration/{peerName}/{registrationCode}**

**Parameters**

**peerName (Required, number )**

Name of the peer

**registrationCode (Required, string )**

Registration code

**Response 200** (application/json)

Headers

Content-Type: `application/json`

Body

```
{ "field": "1013", "gid": "1", "maxElement": "1000", "mpcProtocol": "additive", "numberOfItems": "2", "numberOfTimeSlots": "2" }
```

**Response 400** (text/plain)

Headers

Content-Type: `text/plain`

**Response 404** (text/plain)

Headers

Content-Type: `text/plain`

[Go to specification](#)

**Get group information - GET /groupInfo/{peerName}/{registrationCode}**

**Parameters**

**peerName (Required, number )**

Name of the peer

**registrationCode (Required, string )**

Registration code

**Response 200** (application/json)

## Headers

**Content-Type:** application/json

## Body

```
{
  "peers": [
    {
      "gid": 1,
      "lastStatus": 0,
      "peerName": "hanspeer",
      "peerType": 1,
      "publicKey": "MH4wEAYHKOZIZj0CAQYFK4EEACQDagAEAjig6xXX4SuME5lRB2ADn7T7CgyH7LXbxy/oS5XhIElBPwz/40cwDac/VgGbDKa+HGBc/AGzwSlScoCDHc7WA1tSkRUkaW/LL9NbA6gIzJLMw+FV3RPor0vpJIofVcAaV6WI1r99v8Y=",
      "url": "https://localhost:12001/p2ds-input-peer/peer"
    },
    {
      "gid": 1,
      "lastStatus": 0,
      "peerName": "peerhans",
      "peerType": 1,
      "publicKey": "MH4wEAYHKOZIZj0CAQYFK4EEACQDagAEAjig6xXX4SuME5lRB2ADn7T7CgyH7LXbxy/oS5XhIElBPwz/40cwDac/VgGbDKa+HGBc/AGzwSlScoCDHc7WA1tSkRUkaW/LL9NbA6gIzJLMw+FV3RPor0vpJIofVcAaV6WI1r99v8Y=",
      "url": "https://localhost:12001/p2ds-input-peer/peer"
    },
    {
      "gid": 1,
      "lastStatus": 0,
      "peerName": "ppeer",
      "peerType": 2,
      "publicKey": "MH4wEAYHKOZIZj0CAQYFK4EEACQDagAEAjig6xXX4SuME5lRB2ADn7T7CgyH7LXbxy/oS5XhIElBPwz/40cwDac/VgGbDKa+HGBc/AGzwSlScoCDHc7WA1tSkRUkaW/LL9NbA6gIzJLMw+FV3RPor0vpJIofVcAaV6WI1r99v8Y=",
      "url": "https://localhost:12001/p2ds-privacy-peer/peer"
    },
    {
      "gid": 1,
      "lastStatus": 0,
      "peerName": "ppeer2",
      "peerType": 2,
      "publicKey": "MH4wEAYHKOZIZj0CAQYFK4EEACQDagAEAjig6xXX4SuME5lRB2ADn7T7CgyH7LXbxy/oS5XhIElBPwz/40cwDac/VgGbDKa+HGBc/AGzwSlScoCDHc7WA1tSkRUkaW/LL9NbA6gIzJLMw+FV3RPor0vpJIofVcAaV6WI1r99v8Y=",
      "url": "https://localhost:12001/p2ds-privacy-peer/peer"
    },
    {
      "gid": 1,
      "lastStatus": 0,
      "peerName": "ppeer3",
      "peerType": 2,
      "publicKey": "MH4wEAYHKOZIZj0CAQYFK4EEACQDagAEAjig6xXX4SuME5lRB2ADn7T7CgyH7LXbxy/oS5XhIElBPwz/40cwDac/VgGbDKa+HGBc/AGzwSlScoCDHc7WA1tSkRUkaW/LL9NbA6gIzJLMw+FV3RPor0vpJIofVcAaV6WI1r99v8Y=",
      "url": "https://localhost:12001/p2ds-privacy-peer/peer"
    }
  ]
}
```

## Response 404 (text/plain)

## Headers

**Content-Type:** text/plain

## Response 400 (text/plain)

## Headers

**Content-Type:** text/plain

[Go to specification](#)

[Get group - GET /group/{gid}?adminKey](#)

## Parameters

**gid (Required, number )**

ID of the group

**adminKey (Required, string )**

Admin key

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"gid": "1", "name": "huhu"}
```

**Response 404** (text/plain)

Headers

**Content-Type:** text/plain[Go to specification](#)**Create group - POST /group{?adminKey}****Parameters****adminKey (Required, string )**

Admin key

**Request** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"name": "huhu"}
```

**Response 200** (application/json)

Headers

**Content-Type:** application/json**Response 403** (text/plain)

Headers

**Content-Type:** `text/plain`

[Go to specification](#)

### **Register a peer - POST /register/{registrationCode}/{?url,name,type}**

#### **Parameters**

##### **registrationCode (Required, string )**

Registration code

##### **url (Required)**

input-peer/peer (required, string) - URL the peer can be reached at

##### **type (Required, number )**

Type of the peer (1 = input, 2 = privacy)

##### **name (Required, string )**

Name of the peer.

#### **Response 200** (application/json)

##### Headers

**Content-Type:** `application/json`

##### Body

```
{ "gid": 1, "lastStatus": 0, "peerName": "peerhans", "peerType": 1, "publicKey": "MH4wEAYHKoZIzj0CAQYFK4EEACQDagAEAJig6xXX4SuME5lRB2ADn7T7CgyH7LXbxy/oS5XhIElBPwz/40cwDAc/VgGbDKa+HGBc/AGzwSlScoCDHc7WA1tSkRUkaW/1L9NbA6gIzJLMw+FV3RPor0vpJIofVcAaV6WI1r99v8Y=", "url": "https://localhost:12001/p2ds-input-peer/peer" }
```

#### **Response 400** (text/plain)

##### Headers

**Content-Type:** `text/plain`

[Go to specification](#)

### **Generate registration code - POST /registration/{gid}/{?adminKey}**

#### **Parameters**

##### **gid (Required, number )**

ID of the group

##### **adminKey (Required, string )**

Admin key

#### **Response 200** (application/json)

## Headers

**Content-Type:** `application/json`

## Body

```
{"gid": "1", "registrationCode": "TEST"}
```

## **Response 404** (text/plain)

### Headers

**Content-Type:** `text/plain`

## **Response 403** (text/plain)

### Headers

**Content-Type:** `text/plain`

[Go to specification](#)

## **Delete group - DELETE /group/{groupId}?adminKey**

### **Parameters**

#### **groupId (Required, number )**

ID of the group

#### **adminKey (Required, string )**

Admin key

## **Response 200** (text/plain)

### Headers

**Content-Type:** `text/plain`

## **Response 404** (text/plain)

### Headers

**Content-Type:** `text/plain`

## **Response 403** (text/plain)

### Headers



Content-Type: text/plain

[Go to specification](#)

### Delete peer - DELETE /peer/{peerName}/{adminKey}

#### Parameters

##### peerName (Required, number )

Name of the peer.

##### adminKey (Required, string )

Admin key

#### Response 200 (text/plain)

Headers

Content-Type: text/plain

#### Response 404 (text/plain)

Headers

Content-Type: text/plain

#### Response 403 (text/plain)

Headers

Content-Type: text/plain

[Go to specification](#)

### Set configuration - POST /configuration/{adminKey}

#### Parameters

##### adminKey (Required, string )

Admin key

#### Request (application/json)

Headers

Content-Type: application/json

Body

```
{"field":"1013","gid":"1","maxElement":"1000","mpcProtocol":"additive","numberOfItems":"2","numberOfTimeSlots":"2"}
```

**Response 200** (application/json)

Headers

**Content-Type:** application/json

Body

```
{"field":"1013","gid":"1","maxElement":"1000","mpcProtocol":"additive","numberOfItems":"2","numberOfTimeSlots":"2"}
```

**Response 404** (text/plain)

Headers

**Content-Type:** text/plain

**Response 403** (text/plain)

Headers

**Content-Type:** text/plain

[Go to specification](#)

**Update peer status - POST /status/{peerName}?registrationCode,status}**

### Parameters

**peerName (Required, string )**

Name of the peer.

**registrationCode (Required, string )**

Registration code

**status (Required, number )**

Status (1 = started, 2 = error, 3 = stopped, 0 = unknown)

**Response 200** (text/plain)

Headers

**Content-Type:** text/plain

**Response 400** (text/plain)

## Headers

**Content-Type:** text/plain

**Response 404** (text/plain)

## Headers

**Content-Type:** text/plain

[Go to specification](#)

## Get Peer - GET /peer/{peerName}?adminKey

### Parameters

**peerName (Required, string )**

Name of the peer.

**adminKey (Required, string )**

Admin key

**Response 200** (application/json)

## Headers

**Content-Type:** application/json

## Body

```
{ "gid": "1", "lastStatus": "1", "peerName": "hanspeer", "peerType": "1", "publicKey": "MH4wEAYHKOZIZj0CAQYFK4EEACQDagAEAjig6xXX4SuME5lRB2ADn7T7CgyH7LXbxy/oS5XhIElBPwz/40cwDac/VgGbDKa+HGBc/AGzwSlScoCDHc7WA1tSkRUkaW/LL9NbA6gIzJLMw+FV3Rpor0vpJIofVcAaV6WI1r99v8Y=", "registrationCode": "TEST", "url": "https://localhost:12001/p2ds-input-peer/peer" }
```

**Response 403** (text/plain)

## Headers

**Content-Type:** text/plain

**Response 404** (text/plain)

## Headers

**Content-Type:** text/plain

[Go to specification](#)

**Start peers - POST /start/{gid}?adminKey{**

**Parameters**

**gid (Required, number )**

ID of the group

**adminKey (Required, string )**

Admin key

**Response 200** (text/plain)

Headers

**Content-Type:** text/plain

**Response 403** (text/plain)

Headers

**Content-Type:** text/plain

[Go to specification](#)

**Stop peers - POST /stop/{gid}?adminKey{**

**Parameters**

**gid (Required, number )**

ID of the group

**adminKey (Required, string )**

Admin key

**Response 200** (text/plain)

Headers

**Content-Type:** text/plain

**Response 403** (text/plain)

Headers

**Content-Type:** text/plain

[Go to specification](#)

**Delete registration - DELETE /registration/{registrationCode}/{adminKey}**

### Parameters

**registrationCode (Required, string )**

Registration code

**adminKey (Required, string )**

Admin key

**Response 200** (text/plain)

Headers

**Content-Type:** text/plain

**Response 403** (text/plain)

Headers

**Content-Type:** text/plain

[Go to specification](#)



# References

- SEPIA (<http://www.sepia.ee.ethz.ch/>)