

# Public APIs for Proofpoint Solutions

---

Revision E

March 2019

## Contents

<b>Proofpoint Public APIs.....</b>	<b>7</b>
Proofpoint Protection Server .....	7
Targeted Attack Protection .....	7
Proofpoint Threat Response .....	7
Emerging Threats .....	7
Proofpoint Enterprise Archiving.....	8
<b>Proofpoint Protection Server REST API Authentication .....</b>	<b>8</b>
Authentication.....	8
Certificate based authentication .....	9
Required Headers .....	9
Response Payload .....	9
Supported payloads.....	10
Request ID.....	11
Reflecting Headers.....	11
Reflecting header example.....	11
<b>Proofpoint End User Management API .....</b>	<b>13</b>
General API Features.....	13
Supported payloads.....	13
Required Headers .....	13
End User Resource .....	13
Resource Identifier .....	14
Methods.....	14
GET Method .....	14
Field Descriptions .....	16

Return codes .....	17
Examples .....	17
PUT Method.....	18
Request .....	18
PUT Method .....	19
Request .....	20
Response.....	21
Return Codes.....	21
Examples .....	22
POST Method.....	23
Request .....	23
Response.....	24
Return Codes.....	24
Examples .....	25
DELETE Method.....	25
Request .....	26
Response.....	26
Return Codes.....	26
Examples .....	26
Error handling .....	26
Error Message Body .....	26
Field Description .....	27
Category internalerrors .....	28
Category invalidarguments .....	28
Category invalidattributes .....	28
Category invalidemail .....	28
Category invalidgroups .....	29

Examples .....	29
Attributes .....	31
ISO Locales .....	33
Field Validation .....	34
Creation Source Identifiers .....	35
Curl Examples .....	35
<b>Proofpoint Quarantine Search REST API.....</b>	<b>37</b>
General API features .....	37
Payload .....	37
Quarantine resource .....	37
Resource Identifier .....	37
Available Methods: .....	37
GET Method .....	38
Request .....	39
Response.....	39
HTTP Body .....	40
Field Descriptions .....	41
Examples for dlpviolation JSON blobs .....	42
Example for messagestatus JSON blob.....	45
Return Codes:.....	46
Examples .....	46
PUT Method .....	47
POST Method.....	47
Request .....	48
Response.....	52
Examples .....	52
DELETE Method.....	53

Error Handling .....	53
Error Message Body .....	53
Possible Error Messages.....	53
<b>Exact Data Match REST API .....</b>	<b>55</b>
Authentication.....	55
Resource Identifier .....	55
Available Methods .....	55
Operations.....	55
Examples.....	57
List all available EDM files.....	57
Delete a file that does not exist .....	58
Delete a file that is in use .....	58
Delete multiple files .....	59
Upload a file.....	59
Upload a file that contains a duplicate column .....	60
<b>Administrator Management REST API .....</b>	<b>61</b>
General API features .....	61
Payload.....	61
Resource Identifier .....	61
GET Method .....	62
Response.....	62
Return Codes.....	62
HTTP Body .....	62
Field Description .....	63
Examples .....	63
PUT Method .....	65
Request .....	65

Response.....	66
Return Codes.....	66
Example.....	66
POST Method.....	68
Request .....	68
Response.....	68
Example.....	69
DELETE Method.....	69
Request .....	70
HTTP Body .....	70
Response.....	70
Example.....	70
Error Handling.....	70
Error Message Body .....	71
Examples .....	71
Field Validation .....	73
CURL Examples .....	73

## **Proofpoint Public APIs**

Proofpoint public APIs can be used to manage and gather information from the Proofpoint solutions.

### ***Proofpoint Protection Server***

Public documentation is included in this document for the REST APIs available for the Proofpoint Protection Server. The operations are accessed through port 10000. See the API general information in this document for access methods.

API general information: [Proofpoint Protection Server REST API Authentication](#)

Supported Proofpoint Protection Server APIs:

<b>API</b>	<b>PPS Version (and later)</b>
Quarantine Search	7.0.X
End User Management	7.5.X
Exact Data Match	8.10.X
Administrator Management	8.10.X

### ***Targeted Attack Protection***

TAP API documentation is available from this URL:

[https://help.proofpoint.com/Threat\\_Insight\\_Dashboard/API\\_Documentation](https://help.proofpoint.com/Threat_Insight_Dashboard/API_Documentation)

### ***Proofpoint Threat Response***

Threat Response API documentation is available from this URL:

[https://kb.proofpoint.com/kb\\_upload/file/PTR\\_API\\_Guide-3\\_0.pdf](https://kb.proofpoint.com/kb_upload/file/PTR_API_Guide-3_0.pdf)

### ***Emerging Threats***

Emerging Threats API documentation is available from this URL:

<http://apidocs.emergingthreats.net>

A datasheet for Emerging Threats is available from this URL:

<https://www.proofpoint.com/sites/default/files/proofpoint-et-intelligence-datasheet-a4-en.pdf>

## ***Proofpoint Enterprise Archiving***

The Proofpoint Enterprise Archiving legal hold API documentation is available from this URL:

<https://kb.proofpoint.com/index.php?View=afile&CategoryID=2303&EntryID=1096>

## **Proofpoint Protection Server REST API Authentication**

This section provides authentication information that applies to all of the Proofpoint Protection Server REST APIs available for the Proofpoint Protection Server.

### ***Authentication***

An administrator must have a Role that includes access to a specific REST API.

Proofpoint on Demand (PoD) administrators must file a support ticket to obtain a role with access to an API.

On premise administrators: edit the `filter.cfg` file and set the following key to true:

```
com.proofpoint.admin.apigui.enable=t
```

In the management interface, create a Role of Type *API* and select the APIs under **Managed Modules** for the Role so that you can give an administrator that Role.



The screenshot shows the 'Administrator > Roles > Add' page. At the top, there are tabs for 'System', 'Email Protection', and 'Information Protection'. Below the tabs, there are 'Add' and 'Cancel' buttons. The 'Role' section has fields for 'Type' (with radio buttons for 'UI' and 'API', where 'API' is selected), 'ID', and 'Description'. The 'Managed Modules' section is a table with a 'Module' column and checkboxes. The 'enduser' and 'Quarantine' modules are checked, while 'Spam Policy', 'administrator', 'policy', 'rules', and 'pss' are not.

Module	Selected
enduser	<input checked="" type="checkbox"/>
Spam Policy	<input type="checkbox"/>
administrator	<input type="checkbox"/>
policy	<input type="checkbox"/>
rules	<input type="checkbox"/>
pss	<input type="checkbox"/>
Quarantine	<input checked="" type="checkbox"/>

### ***Certificate based authentication***

To enable certificate based authentication for **System > Certificates > Settings > Client Certificates** for API access set the following key in `filter.cfg` to `true`:

```
com.proofpoint.httpe.admin.clientcert.enable=f
```

Certificate based authentication for API access is off by default.

Next, upload the actual certificate in `.pem` format on the **System > Certificates > Client Certificates** page in the management interface.

### ***Required Headers***

The client must specify the following headers:

**Host**

**Accept** (such as `application/json`)

**User-Agent**

**Content-Type** (when there is content)

### ***Response Payload***

The client can indicate the response payload by specifying the Accept header:

Accept: text/xml

In this case this will return an XML document

```
<?xml version="1.0" encoding="UTF-8"?><opt>
  <attributes>
    <Locale>enUS</Locale>
  </attributes>
  <created>2013-09-30 14:52:11</created>
  <email>foo@example.com</email>
  <uid>foo@example.com</uid>
</opt>
```

The Content-Type of the response reflects the actual payload protocol

Content-Type:text/xml; Charset=UTF-8

Alternatively a format parameter can be specified in the URL to overwrite the payload:

[https://server:10000/rest/v1/enduser/foo@example.com?\\_format=json](https://server:10000/rest/v1/enduser/foo@example.com?_format=json)

Result:

```
{"uid":"foo@example.com",,"created":"2013-09-30
14:52:11","email":"foo@example.com","attributes":{"Locale":"enUS"}}
```

Also the Content-Type of the response reflects the actual payload protocol

Content-Type:application/json; Charset=UTF-8

### **Supported payloads**

The following payloads are supported:

Accept	format
text/xml	xml
application/json	json

## Request ID

Each REST request will have a unique request ID assigned by the server. This request ID will be returned as a header (`x-pps-reqid`) and can be found in the Proofpoint logs.

```
x-pps-reqid:18901-1381378369-00057
```

This request ID is unique per PPS cluster.

Here is an example of a PPS *webservices* log entry:

```
[2013-10-09 21:12:49.077744 -0700] info ppsreqid=18901-1381378369-00057  
module=enduser method=get api=/v1/enduser/foo@example.com status=done t=0.01
```

## Reflecting Headers

PPS can be configured to reflect certain headers from the request back to the response. Customers can use this functionality to insert their own transaction ID in all requests and receive this transaction ID back in the response. The headers to reflect can be specified in `filter.cfg` per REST API.

The `filter.cfg` value needs to follow this example:

```
com.proofpoint.api.<restapi>.headersreflect=<header>[:<responseheader>][,<header>[:<responseheader>]]
```

Multiple headers can be specified and an option response header name can be specified if the response header is different than the request header.

### Reflecting header example

The value in the `filter.cfg` file:

```
com.proofpoint.api.enduser.headersreflect=TransactionID,x-ping:x-pong
```

When you make changes to the `filter.cfg` file the *admin* service and *apiservice* need to be restarted.

**Request headers:**

```
TransactionID: 9876543210  
x-ping: Foo
```

**Response headers:**

```
TransactionID: 9876543210  
x-pong: Foo
```

## Proofpoint End User Management API

The End User Management API allows for creating users, modifying users, deleting users, and managing entries on blocked lists and safe lists. It was implemented in PPS release 7.5.

### ***General API Features***

Authentication requires an API Role with an *enduser* Managed Module enabled and optional certificate based authentication. See the [Authentication](#) section for more information.

Once configured the End User Management REST APIs are available through the admin port (default 10000).

### ***Supported payloads***

The following payloads are supported:

Accept	format
text/xml	xml
application/json	json

### ***Required Headers***

The client must specify the following headers:

**Host**

**Accept** (such as application/json)

**User-Agent**

**Content-Type** (when there is content)

### ***End User Resource***

Describes a single "*enduser*" in the target PPS system. An *enduser* is identified by an email address or `uid`. The email address can be either the primary email address or any alias for that user.

**Resource Identifier**

/rest/v1/enduser/{email|uid}

Part	Required	Type	Example	Description
email	yes	string	foo@bar.com	Email address of the enduser. The user must exist in the Proofpoint Protection Server User Repository.
uid	yes	string	foo	uid of the end user.

**Methods****GET Method**

Gets the end user resource with the specified email address or UID.

**Request****HTTP Body**

None

**Response**

The GET response returns all user profile properties in either XML or JSON format (based on **Accept** header or **\_format** parameter)

The password is never returned in the response.

## HTTP Body

```
{
  alias          : ["bar\@example.com"],
  attributes     : {
    access_Digest          : 1,
    access_Encryption      : 1,
    access_Webapp          : 1,
    Attribute1             : "",
    Attribute2             : "",
    AuthSource             : 1,
    DigestContentType      : 5,
    DigestProfile          : 0,
    DigitalAssets          : 1,
    DigitalAssetsDocumentCreation : 0,
    EncryptionReadOnly     : 0,
    ExternalLocale         : "enUS",
    ExternalPasswordPolicy : 2,
    ExternalTemplate       : 1,
    ForwarderEnable        : 1,
    InboundSelfRemediation : 1,
    IncludeAuditFolder     : 0,
    Locale                 : "enUS",
    MessageAuditing        : 0,
    OptIn                  : 1,
    OutboundSelfRemediation : 1,
    PasswordOption         : 0,
    PasswordPolicy         : 0,
    RegulatoryCompliance   : 1,
    SafelistUseHeader      : 1,
    SecurityPolicy         : 0,
    SendDigest             : 1,
    SendEmptyDigest        : 1,
    SpamClassification     : "default",
    Template               : 0,
  },
  blocklist      : [],
  created        : "2013-10-14 20:15:12",
  creationsource : 1,
  email          : "foo\@example.com",
  firstname      : "John",
  groups         : ["Engineering", "Marketing"],
  lastmodified   : "2013-10-14 20:16:24",
  lastname       : "Doe",
  lastpasswordchange : "",
  lastsignon     : "",
}
```

```
safelist      : [],
suborg        : "Organization",
uid           : "foo\\@example.com",
type         : 0,
}
```

The attributes are returned in a calculated form based on the settings in the Organization and Sub-Org/Group membership of the user profile. In order to retrieve the raw attribute values the raw parameter can be specified (see the Examples).

### Field Descriptions

Field	Type	Example	Description	Read-Only
email	String	"email": "foo@bar.com"	Primary email address of the enduser.	
alias	Array of Strings	"alias": [ "f.foo@bar.com" ]	List of aliases for the enduser.	
firstname	String	"firstname" : "Jane"	First name of enduser.	
lastname	String	"lastname" : "Doe"	Last name of enduser.	
safelist	Array of Strings	["foo.com", "bar.com"]	List of all safe list entries for the enduser.	
blocklist	Array of Strings	["spammer.com"]	List of all block list entries for the enduser.	
uid	String	"uid" : "foo"	uid of the user.	
groups	Array of Strings	["engineering", "marketing"]	List of all the groups to which the user belongs.	
created	String	"2013-07-01 12:00:00"	Timestamp when profile was created (in UTC).	Y
lastmodified	String	"2013-07-01 12:00:00"	Timestamp when profile changes were made (in UTC).	Y
creationsource	Int	4	Source for the profile creation (see the table below).	Y
lastsignon	String	"2013-07-01 12:00:00"	Timestamp for the last login by the end user.	Y
lastpasswordchange	String	"2013-07-01 12:00:00"	Timestamp for the last password change.	Y



Field	Type	Example	Description	Read-Only
attributes	Hash		All attributes for this user profile. The value of these attributes is calculated based on the Organization, Sub-Org/Group membership and Profile settings. For a description of these attributes, see the table at the end of this document.	
suborg	String	"Organization"	Defines the sub-organization to which this user profile belongs.	
type	Int	0	0=User 1=Mailing List	

### Return codes

Code	Description
200	Success
404	End user does not exist

### Examples

#### GET user by UID

```
GET https://localhost:10000/rest/v1/enduser/foo
```

#### GET user by email address

The email address specified can be primary email address or any alias.

```
GET https://localhost:10000/rest/v1/enduser/foo@example.com
```

#### GET user by UID and return non-calculated attribute values

```
GET https://localhost:10000/rest/v1/enduser/foo?raw=1
```

### PUT Method

Modifies a user profile. A user profile can be identified by UID or email address (primary or any alias).

### Request

Field	Type	Example	Description
<b>uid</b>	String	userid123456789	UID of the user.
<b>safelist</b>	Array of Strings	"newsafelist@entry.com"	One or more safe list entries to be added. For safe and block list entries there is currently a maximum of 500 safe list entries and 500 block list entries per user. Entries are validated and need to be specified as either a email address or domain.
<b>blocklist</b>	Array of Strings	["x@spammer.com", "y@spammer.com"]	see safelist
<b>deletesafelist</b>	Array of Strings		One or more safe list entries to be deleted
<b>deleteblocklist</b>	Array of Strings		One or more block list entries to be deleted
<b>addemail</b>	Array of Strings	"joe_alias@example.com"	One or more email addresses to add (if in the same operation you delete the primary email address, the first email specified in addemail will be stored as primary)
<b>deleteemail</b>	Array of Strings	["joe.a@example.com", "joe_aa@example.com"]	One or more email addresses to be deleted from the profile. At least one email address needs to remain in the profile in order for this operation to succeed. If you delete the primary email address, the first alias (order undetermined for multiple aliases) will be promoted to the primary email address.
<b>addgroup</b>	Array of Strings	["marketing"]	One or more group names to be added to the profile. The group needs to exist in order for the group to be added to the profile.
<b>deletegroup</b>	Array of Strings	["engineering"]	One or more group names to be removed from the profile.
<b>email</b>	String	"bar@example.com"	Sets the primary email address
<b>firstname</b>	String	John	First Name
<b>lastname</b>	String	Doe	Last Name

Field	Type	Example	Description
<b>password</b>	String	"\$cre7Pa\$\$w0rd"	Specify a password for the end user. NOTE: The password specified is not validated against the password policy. NOTE: Only one way hashes are stored in the Proofpoint backend
<b>changepassword</b>	Boolean (0/1)	1	When set to 1, the user is required to change the password the next login.
<b>&lt;ATTRIBUTES&gt;</b>	Hash of one or more Attributes		One or more attributes specified in the table below can be specified. Attributes that are not specified will not be modified. Invalid attributes will be ignored.
<b>resetpassword</b>	Boolean (0/1)	1	Sets a temporary password for the user and sends out a welcome message with the new password and link to EUWEB
<b>welcome</b>	Boolean (0/1)	1	Sends welcome message with a link to EUWEB
<b>forcegroup</b>	Boolean (0/1)	1	Groups specified in the addgroup parameter will automatically be created if they do not exist. By default this feature is turned off.
<b>addsuborg</b>	String	"Engineering"	Add user to a specified sub-org. This call will fail if the sub-org does not exist (unless the forcesuborg flag has been specified). If the user is a member of another sub-org, the user will sever this relationship and will be moved under the specified sub-org.  NOTE: If you also assign the user to a group under a different sub-org, this call will fail and return an error
<b>deletesuborg</b>	Boolean (0/1)		If set to 1, user is moved from the suborg to the organization
<b>forcesuborg</b>	Boolean (0/1)		If set to 1, the sub-org will be created if the sub-org does not exist (only works in combination with the addsuborg parameter

## PUT Method

Modifies a user profile. A user profile can be identified by UID or email address (primary or any alias).

## Request

Field	Type	Example	Description
<b>uid</b>	String	userid123456789	UID of the user.
<b>safelist</b>	Array of Strings	"newsafelist@entry.com"	One or more safe list entries to be added. For safe and block list entries there is currently a maximum of 500 safe list entries and 500 block list entries per user. Entries are validated and need to be specified as either a email address or domain.
<b>blocklist</b>	Array of Strings	[[ "x@spammer.com", "y@spammer.com"]	see safelist
<b>deletesafelist</b>	Array of Strings		One or more safe list entries to be deleted
<b>deleteblocklist</b>	Array of Strings		One or more block list entries to be deleted
<b>addemail</b>	Array of Strings	"joe_alias@example.com"	One or more email addresses to add (if in the same operation you delete the primary email address, the first email specified in addemail will be stored as primary)
<b>deleteemail</b>	Array of Strings	["joe.a@example.com", "joe_aa@example.com"]	One or more email addresses to be deleted from the profile. At least one email address needs to remain in the profile in order for this operation to succeed. If you delete the primary email address, the first alias (order undetermined for multiple aliases) will be promoted to the primary email address.
<b>addgroup</b>	Array of Strings	["marketing"]	One or more group names to be added to the profile. The group needs to exist in order for the group to be added to the profile.
<b>deletegroup</b>	Array of Strings	["engineering"]	One or more group names to be removed from the profile.
<b>email</b>	String	"bar@example.com"	Sets the primary email address
<b>firstname</b>	String	John	First Name
<b>lastname</b>	String	Doe	Last Name
<b>password</b>	String	"\$cre7Pa\$\$w0rd"	Specify a password for the end user. NOTE: The password specified is not validated against the password policy. NOTE: Only one way hashes are stored in the Proofpoint backend
<b>changepassword</b>	Boolean (0/1)	1	When set to 1, the user is required to change the password the next login.
<b>&lt;ATTRIBUTES&gt;</b>	Hash of one or more Attributes		One or more attributes specified in the table below can be specified. Attributes that are not

Field	Type	Example	Description
			specified will not be modified. Invalid attributes will be ignored.
<b>resetpassword</b>	Boolean (0/1)	1	Sets a temporary password for the user and sends out a welcome message with the new password and link to EUWEB
<b>welcome</b>	Boolean (0/1)	1	Sends welcome message with a link to EUWEB
<b>forcegroup</b>	Boolean (0/1)	1	Groups specified in the addgroup parameter will automatically be created if they do not exist. By default this feature is turned off.
<b>addsuborg</b>	String	"Engineering"	Add user to a specified sub-org. This call will fail if the sub-org does not exist (unless the forcesuborg flag has been specified). If the user is a member of another sub-org, the user will sever this relationship and will be moved under the specified sub-org.  NOTE: If you also assign the user to a group under a different sub-org, this call will fail and return an error
<b>deletesuborg</b>	Boolean (0/1)		If set to 1, user is moved from the suborg to the organization
<b>forcesuborg</b>	Boolean (0/1)		If set to 1, the sub-org will be created if the sub-org does not exist (only works in combination with the addsuborg parameter

### Response

It will return the full user profile (see GET) when the update is successful.

### Return Codes

Code	Description
<b>200</b>	An existing end user has been modified according to the attached entity
<b>400</b>	Required fields are missing or have invalid data
<b>404</b>	End user does not exist
<b>413</b>	Request entity too large; the enduser data set exceeds the allowed system maximum size setting

## **Examples**

### **Set first name and last name given a UID**

```
PUT https://localhost:10000/rest/v1/enduser/foo
{ firstname : 'John', lastname : 'Doe' }
```

### **Change primary email address given a UID**

```
PUT https://localhost:10000/rest/v1/enduser/foo
{ email : 'new@example.com' }
```

### **Change primary email address given an email address**

```
PUT https://localhost:10000/rest/v1/enduser/foo@example.com
{ email : 'bar@example.com' }
```

### **Change primary email address given an email address (alternative method)**

```
PUT https://localhost:10000/rest/v1/enduser/foo@example.com
{ addemail : 'bar@example.com', deleteemail : 'foo@example.com' }
```

### **Change language to German (de) given an email address**

```
PUT https://localhost:10000/rest/v1/enduser/foo@example.com
{ attributes : { Locale : 'de' } }
```

### **Add a user to a group (given an email address)**

```
PUT https://localhost:10000/rest/v1/enduser/foo@example.com
{ addgroup : 'DisabledUsers' }
```

### **Remove a user from a group (given a UID)**

```
PUT https://localhost:10000/rest/v1/enduser/foo
{ deletegroup : 'DisabledUsers' }
```

### **Remove an email address from a user profile**

If you delete the primary email address, an alias will be promoted to primary email address. If there is no alias, the operation will fail as at least one email address needs to be specified per user profile.

```
PUT https://localhost:10000/rest/v1/enduser/foo
{ deleteemail : 'bar@example.com' }
```

## Change password

This forces user to change the password after next login.

```
PUT https://localhost:10000/rest/v1/enduser/foo
{ password : 'abc123', changepassword : 1 }
```

## POST Method

Create a new user. On the URI, you can either specify the primary email address or the UID. Both must be unique in the system and when you specify a UID, the primary email address needs to be specified as a parameter. All other parameters are optional.

### Request

Field	Type	Example	Description
<b>uid</b>	String	userid123456789	uid of the user. If no uid is specified during the create, the primary email address will be used as uid
<b>safelist</b>	Array of Strings	"newsafelist@entry.com"	One or more safe list entries to be added. For safe and block list entries there is currently a maximum of 500 safe list entries and 500 block list entries per user. Entries are validated and need to be specified as either a email address or domain.
<b>blocklist</b>	Array of Strings	[[ "x@spammer.com", "y@spammer.com"	see safelist
<b>addemail</b>	Array of Strings	"joe_alias@example.com"	One or more email addresses to add (if in the same operation you delete the primary email address, the first email specified in addemail will be stored as primary)
<b>addgroup</b>	Array of Strings	["marketing"]	One or more group names to be added to the profile. The group needs to exist in order for the group to be added to the profile.
<b>email</b>	String	"bar@example.com"	Sets the primary email address
<b>firstname</b>	String	John	First Name
<b>lastname</b>	String	Doe	Last Name
<b>password</b>	String	"\$cre7Pa\$\$w0rd"	Specify a password for the end user. NOTE: The password specified is not validated against the password policy.

Field	Type	Example	Description
			NOTE: Only one way hashes are stored in the Proofpoint backend
<b>changepassword</b>	Boolean (0/1)	1	When set to 1, the user is required to change the password the next login.
<b>&lt;ATTRIBUTES&gt;</b>	Hash of one or more Attributes		One or more attributes specified in the table below can be specified. Attributes that are not specified will not be modified. Invalid attributes will be ignored.
<b>resetpassword</b>	Boolean (0/1)	1	Sets a temporary password for the user and sends out a welcome message with the new password and link to EUWEB
<b>welcome</b>	Boolean (0/1)	1	Sends welcome message with a link to EUWEB
<b>forcegroup</b>	Boolean (0/1)	1	Groups specified in the addgroup parameter will automatically be created if they do not exist. By default this feature is turned off.
<b>type</b>	Int	1	0 = User (default)  1 = Mailing List
<b>addsuborg</b>	String	"Engineering"	Add user to a specified sub-org. This call will fail if the sub-org does not exist (unless the forcesuborg flag has been specified).  NOTE: If you also assign the user to a group under a different sub-org, this call will fail and return an error
<b>forcesuborg</b>	Boolean (0/1)		If set to 1, the sub-org will be created if the sub-org does not exist (only works in combination with the addsuborg parameter)

### **Response**

When the operation is successful the response will return the full user profile (see GET)

### **Return Codes**

Code	Description
<b>200</b>	User profile has been created
<b>400</b>	Required fields are missing or have invalid data (invalid email address or duplicate email)



## **Examples**

Create user using UID on the URI and specifying primary email address

```
POST https://localhost:10000/rest/v1/enduser/foo
{ email : 'foo@example.com' }
```

Create user using UID on the URI and specifying email address to add

```
POST https://localhost:10000/rest/v1/enduser/foo
{ addemail : 'foo@example.com' }
```

Create user using UID on the URI and adding multiple email addresses

```
POST https://localhost:10000/rest/v1/enduser/foo
{ addemail : [ 'foo@example.com', 'bar@example.com' ] }
```

In this case 'foo@example.com' will be the primary email address and bar@example.com an alias

Create user using UID on the URI and adding aliases as well as a primary email address

```
POST https://localhost:10000/rest/v1/enduser/foo
{ email : 'foo@example.com', addemail : 'bar@example.com' }
```

In this case 'foo@example.com' will be the primary email address and bar@example.com an alias

Create user using email on the URI and specifying UID

```
POST https://localhost:10000/rest/v1/enduser/foo@example.com
{ uid : 'foo' }
```

In this case 'foo@example.com' will be the primary email address and foo will be the UID

## **DELETE Method**

Delete a user profile. The user profile will be removed (UID, primary email address, aliases, attributes etc). User data like quarantine data and encryption keys will be preserved and will be available to the user once the user is re-created.

To delete a single email address from a user profile, please use the PUT method and specify the "deleteemail" parameter.

## Request

### HTTP Body

No options can be specified

## Response

Upon successful deletion of the user profile the response will be a HTTP 200 with the following body

```
{ status : 200 }
```

## Return Codes

Code	Description
200	User profile has been deleted
404	User profile does not exist

## Examples

Delete user by email address

```
DELETE https://localhost:10000/rest/v1/enduser/foo@example.com
```

Delete user by UID

```
DELETE https://localhost:10000/rest/v1/enduser/foo
```

## Error handling

The REST API will return detailed information regarding HTTP 4xx error returned.

### Error Message Body

```
{
  status : 400,
  message : "Invalid arguments",
  errors : {
    invalidarguments : [ { error : "User already exists", user :
"foo@example.com" } ]
  }
}
```

}  
}

### Field Description

Field	Required	Type	Example	Description
<b>status</b>	yes	Integer	404	HTTP status code
<b>message</b>	yes	String	Invalid arguments	Human readable error message
<b>errors</b>	no	Hash of Arrays	{ invalidarguments : [ { error : "User already exists", user : "foo@example.com" } ],  invalidemail : [ { error : "Invalid email address", email : "bad^%*^#@\$email.com" } ] }	A hash of error categories. If "errors" presents, at least one of the following category is listed in the hash. internalerrors invalidarguments invalidattributes invalidemail invalidgroups
<b>internalerrors</b>	no	Array of Hashes	[ { error : "Modify user failed", user : "foo@example.com" } ]	Error details grouped as internal errors. See Category internalerrors section
<b>invalidarguments</b>	no	Array of Hashes	[ { error : "User already exists", user : "foo@example.com" } ],  { error : "Invalid safelist entry", safelist : "safe&*^#@\$email.com" } ]	Error details grouped as invalid arguments. See Category invalidarguments section
<b>invalidattributes</b>	no	Array of Hashes	[ { error : "Invalid attribute name or value", ExternalTemplate : "s" } ],  { error : "Invalid attribute name or value", SpamClassification : "doh" },  { error : "Invalid attribute name or value", AuthSource : "3" } ]	Error details grouped as invalid attributes. See Category invalidattributes section
<b>invalidemail</b>	no	Array of Hashes	[ { error : "Email address already used", email : "foo1@example.com" } ],  { error : "Email address already used", addemail : "bar@example.com" },  { error : "Email address already used", addemail : "bar1@example.com" } ]	Error details grouped as invalid email. See Category invalidemail section

Field	Required	Type	Example	Description
<b>invalidgroups</b>	no	Array of Hashes	[ { error : "Not found", addgroup : "huh"},  { error : "Owned by other suborg", addgroup : "taipei"} ]	Error details grouped as invalid groups. See Category invalidgroups section

### *Category internalerrors*

Field	Required	Description	Possible Strings
<b>error</b>	yes	Human readable error message	Add/Modify user failed
<b>fieldname or user</b>	no	Specifying the fieldname/user causing the error	See Field in PUT/POST/GET/DELETE request tables

### *Category invalidarguments*

Field	Required	Description	Possible Strings
<b>error</b>	yes	Human readable error message	User not found Invalid password 'Attributes not defined as a hash Enduser must have email address Safelist limit exceeded Invalid safelist entry Blocklist limit exceeded Invalid blocklist entry
<b>&lt;field name&gt; or user</b>	no	Specifying the fieldname/user causing the error	See Field in PUT/POST/GET/DELETE request tables

### *Category invalidattributes*

Field	Required	Description	Possible Strings
<b>error</b>	yes	Human-readable error message	Invalid locale Invalid attribute name or value
<b>&lt;attribute name&gt;</b>	yes	Specifying the attribute name causing the error	See Attributes table

### *Category invalidemail*

Field	Required	Description	Possible Strings
<b>error</b>	yes	Human readable error message	Invalid email address Email address already used

			Add email failed
<b>addemail or email</b>	yes	Specifying the fieldname/user causing the error	

### **Category invalidgroups**

Field	Required	Description	Possible Strings
<b>error</b>	yes	Human readable error message	Group not found Group belongs to different suborg Add group failed Delete group failed
<b>addgroup or deletigroup</b>	yes	Specifying the group name causing the error	

### **Examples**

Get a nonexistent user

GET https://localhost:10000/rest/v1/enduser/invaliduser

results in a HTTP 404 error with the following response body

```
{
  status : 404,
  message : "Invalid arguments",
  errors : {
    invalidarguments : [{ error : "User not found", user :
"invaliduser" }]
  }
}
```

Create a user that already exists

POST https://localhost:10000/rest/v1/enduser/foo@example.com

```
{ uid : 'foo' }
```

results in a HTTP 400 error with the following response body

```
{
  status : 400,
  message : "Invalid arguments",
  errors : {
    invalidarguments : [ { error : "User already exists", user :
"foo@example.com" } ]
  }
}
```

```
    }
  }
}
```

### Create user using invalid email address

POST https://localhost:10000/rest/v1/enduser/bar

```
{ email : 'bad^%*#@$email.com' }
```

results in a HTTP 400 error with the following response body

```
{
  status : 400,
  message : "Invalid arguments",
  errors : {
    invalidemail : [ { error : "Invalid email address", email
: "bad^%*#@$email.com" } ]
  }
}
```

### Create user using invalid email and attributes

POST https://localhost:10000/rest/v1/enduser/u6@example.com  
 { uid : "u6", addemail : ["u6\_1@@(\$@\$(@\$)example.com)], attributes : {  
 OptIn : "1", ExternalTemplate : "s", SpamClassification : "doh",  
 AuthSource : "3" } }

results in a HTTP 400 error with the following response body

```
{
  status : 400,
  message : "Invalid arguments",
  errors : {
    invalidemail : [ { error : "Invalid email address", addemail :
"u6_1@@($@$(@$)example.com" } ],
    invalidattributes : [ { error : "Invalid attribute name or
value", ExternalTemplate : "s" },
      { error : "Invalid attribute name or
value", SpamClassification : "doh" },
      { error : "Invalid attribute name or
value", AuthSource : "3" } ]
  }
}
```

### Change user's email to one being used by other account

```
PUT https://localhost:10000/rest/v1/enduser/foo
{ email : 'fool@example.com' }
```

results in a HTTP 400 error with the following response body

```
{
  status : 400,
  message : "Invalid arguments",
  errors : {
    invalidemail : [ { error : "Email address already used", email :
"fool@example.com" } ]
  }
}
```

### Change user using invalid email, aliases and groups

```
PUT https://localhost:10000/rest/v1/enduser/foo
{ addemail : [ "bar@example.com", "bar1@example.com"], email :
"fool@example.com", addgroup : ["huh", "taipei" ] }
```

results in a HTTP 400 error with the following response body

```
{
  status : 400,
  message : "Invalid arguments",
  errors: {
    invalidemail : [ { error : "Email address already used", email :
"fool@example.com" },
{ error : "Email address already used", addemail
: "bar@example.com" },
{ error : "Email address already used", addemail
: "bar1@example.com" } ],
    invalidgroups : [ { error : "Not found", addgroup : "huh"},
{ error : "Owned by other suborg", addgroup :
"taipei"} ]
  }
}
```

## Attributes

Each user profile has a set of attributes that can be read or modified. The values are calculated based on the hierarchy defined (Organization/Sub-Org/Groups/Users) unless the raw parameter is specified during a GET.

The default raw value of all attributes is "-1" When this is defined, the value will be derived from the Organization or Sub-Org/Group membership. Set any value to -1 in order to use the derived value.

Attribute names are case-sensitive.

Attribute Name	Description	Example	Value Description
<b>Attribute1</b>	Generic attribute. Not Used		
<b>Attribute2</b>	Generic attribute. Not Used		
<b>AuthSource</b>	Authentication Source to be used	1	Internal ID of the authentication source
<b>DigestContentType</b>	Content type of how digest reports are generated	5	1 = HTML/Text 2 = Text Only 3 = HTML Only 4 = HTML/Text + HTTP Links 5 = HTML Only + HTTP Links 6 = Simple HTML/Text 7 = Simple HTML Only 8 = Simple HTML/Text + HTTP Links 9 = Simple HTML Only + HTTP Links
<b>DigestProfile</b>	Digest branding	0	Internal ID of branding profile
<b>DigitalAssets</b>	Enable DAS enforcements	1	0=off 1=on
<b>DigitalAssetsDocumentCreation</b>	Allow user to upload DAS documents	0	0=off 1=on
<b>EncryptionReadOnly</b>	Prevent user from Initiate/Reply/Forward in Encryption	0	0=off 1=on
<b>ExternalLocale</b>	Locale to be used for sending encryption message	enUS	ISO locale
<b>ExternalPasswordPolicy</b>	Password policy to be used for encryption recipients	2	Internal ID of password policy
<b>ExternalTemplate</b>	Branding profile for sending encryption message	1	Internal ID of branding profile
<b>ForwarderEnable</b>	Enable POP3 forwarder	1	0=off 1=on
<b>InboundSelfRemediation</b>	Enable Self Remediation	1	0=off 1=on
<b>IncludeAuditFolder</b>	Include Audit in digest report	0	0=off 1=on



Attribute Name	Description	Example	Value Description
<b>Locale</b>	User locale	enUS	ISO locale
<b>MessageAuditing</b>	Enable message auditing	0	0=off 1=on
<b>OptIn</b>	Enable spam filtering	1	0=off 1=on
<b>OutboundSelfRemediation</b>	Enable outbound self remediation	1	0=off 1=on
<b>PasswordOption</b>	Do not allow user to change password	0	0=off 1=on
<b>PasswordPolicy</b>	Password policy	0	Internal ID of password policy
<b>RegulatoryCompliance</b>	Enable Regulatory Compliance	1	0=off 1=on
<b>SafelistUseHeader</b>	Use header instead of envelop for the sender in digest reports	0	0=off 1=on
<b>SecurityPolicy</b>	Response Profile	0	Internal ID of response profile
<b>SendDigest</b>	Enable digest reports	1	0=off 1=on
<b>SendEmptyDigest</b>	Enable sending an empty digest report	0	0=off 1=on
<b>SpamClassification</b>	Spam classification profile to be used	default	ID of spam policy
<b>Template</b>	Branding template	0	Internal ID of branding profile
<b>access_Digest</b>	Access to Digest	0	0=off 1=on
<b>access_Encryption</b>	Access to Encryption	0	0=off 1=on
<b>access_Webapp</b>	Access to EUWEB	0	0=off 1=on

## ISO Locales

The following are the locales supported by PPS 8.x:

ID	Locale
sv	Swedish (Svenska)
it	Italian (Italiano)
es	Spanish (Español)
da	Danish (Dansk)
zh	Chinese(Simplified)

enus English (US)  
 pl Polish (Polski)  
 ru Russian  
 nl Dutch (Nederlands)  
 fi Finnish (Suomi)  
 ptbr Portuguese (Brazil)  
 zhtw Chinese(Traditional)  
 fr French (Francais)  
 de German (Deutsch)  
 ja Japanese

## Field Validation

Name	Min Size	Max Size	Format
uid	1	100	
firstname	0	100	
lastname	0	100	
email	6	255	Email validation as per RFC Spec
attributes.Locale			Needs to be a valid supported Locale (see table)
group	1	100	Required Field
suborg	1	100	Required Field
alias	6	255	Array of email addresses. Email address validation as per RFC Spec Minimum entries in the array is 0, maximum is 255
safelist	6	255	Array of email addresses. Email address validation as per RFC Spec Minimum entries in the array is 0, maximum is 255
blocklist	6	255	Array of email addresses. Email address validation as per RFC Spec Minimum entries in the array is 0, maximum is 255
deletesafelist	6	255	Array of email addresses. Email address validation as per RFC Spec Minimum entries in the array is 0, maximum is 255
deleteblocklist	6	255	Array of email addresses. Email address validation as per RFC Spec Minimum entries in the array is 0, maximum is 255

Name	Min Size	Max Size	Format
<b>addemail</b>	6	255	Array of email addresses. Email address validation as per RFC Spec Minimum entries in the array is 0, maximum is 255
<b>deleteemail</b>	6	255	Array of email addresses. Email address validation as per RFC Spec Minimum entries in the array is 0, maximum is 255
<b>addgroup</b>			Adds a user to an existing group. There is no minimum or maximum number of groups associated with a user.
<b>deletegroup</b>			Deletes a user from an existing group. There is no minimum or maximum number of groups associated with a user.
<b>password</b>			.

## Creation Source Identifiers

Each user profile has a creation source identifying the process that created this user profile.

This information is read-only

Source	ID
<b>CREATIONSOURCE_DEFAULT</b>	0
<b>CREATIONSOURCE_ADMIN</b>	1
<b>CREATIONSOURCE_IMPORT</b>	2
<b>CREATIONSOURCE_CLI</b>	3
<b>CREATIONSOURCE_API</b>	4
<b>CREATIONSOURCE_TRUSTEDMESSAGING</b>	5
<b>CREATIONSOURCE_EUWEB</b>	6
<b>CREATIONSOURCE_APPLIANCE_DOMAIN</b>	7
<b>CREATIONSOURCE_SYSTEM</b>	8
<b>CREATIONSOURCE_DIGEST</b>	9

## Curl Examples

An easy way to try out the API is to use `curl`. Below you will find several examples.

Note: when testing against a server without a cert, you will need to specify the `--insecure` flag. This should not be used against production servers.

Create new user

```
curl -u apiuser:userpw1! -X POST -H "Content-Type:application/json" -d
'{"email" : "foo@example.com" }'
https://testserver:10000/rest/v1/enduser/foo
```

#### Add user to a group

```
curl -u apiuser:userpw1! -X PUT -H "Content-Type:application/json" -d
'{"addgroup" : "honeypoint" }'
https://testserver:10000/rest/v1/enduser/foo
```

#### Delete user from a group

```
curl -u apiuser:userpw1! -X PUT -H "Content-Type:application/json" -d
'{"deletegroup" : "honeypoint" }'
https://testserver:10000/rest/v1/enduser/foo
```

#### Update Spam Policy for a user

```
curl -u apiuser:userpw1! -X PUT -H "Content-Type:application/json" -d
'{"attributes" : { "SpamClassification" : "testpolicy" } }'
https://testserver:10000/rest/v1/enduser/foo
```

## Proofpoint Quarantine Search REST API

Use the Quarantine Search REST API to search the Quarantine database. This API was implemented in PPS release 7.0.

### **General API features**

Authentication: basic authentication via PPS (requires API User with "end user API") and optional certificate based authentication. Refer to the [Authentication](#) section for more information.

Once configured the end user REST APIs are available through the admin port (default 10000).

### **Payload**

The following payloads are supported:

Accept
<b>application/json</b>

The client must specify the following headers:

Host

**Accept** (application/json)

User-Agent

**Content-Type** (when there is content)

### **Quarantine resource**

Represents the Proofpoint Protection Server Quarantine database.

### **Resource Identifier**

Resource identifier is the fixed path.

/rest/v1/quarantine

### **Available Methods:**

GET	Yes
<b>PUT</b>	No

DELETE	No
POST	Yes

## GET Method

Search for quarantine messages with the specified parameters.

Parameter	Required	Type	Example	Description	Note
queryid	no	string	queryid=search1	ID mainly for the REST client to keep track of the search results	If not specified, a unique ID will be returned in a successful response
from	from or rcpt or subject must be specified	string	from=foo@example.com from=foo* from=*example.com from=@bar.com	Envelope message sender equals, starts with, ends with or is in a domain such as "bar.com"	
rcpt	from or rcpt or subject must be specified	string	rcpt=bar@example.com rcpt=bar* rcpt=*example.com rcpt=@bar.com	Envelope message sender equals, starts with, ends with or is in a domain such as "bar.com"	
subject	from or rcpt or subject must be specified	string	subject=Team Meeting subject=Tea* subject=%E9%95%B7%E9%80%94%E9%9B%BB%E8%A9%B1 subject=*Free%20Shipping*	Message subject starts with, ends with or contains	mime-decoded and url-encoded utf8 string
startdate	no	UTC time string	startdate=2011-05-3 23:04:42	Start date of the date/time range	Defaults to the last 24 hours if neither startdate nor enddate is specified  If only the startdate is specified, all of the records inserted after the startdate will be searched
enddate	no	UTC time string	enddate=2011-05-4 23:04:42	End date of the date/time range	If only the enddate is specified, all of the records inserted before the enddate will be

Parameter	Required	Type	Example	Description	Note
					searched
<b>folder</b>	no	string	folder=Adult	Quarantine folder name	Defaults to "Quarantine" folder if no folder is specified
<b>guid</b>	no	string	guid=32c6be9dd09f572205c7c2f0e86cc861	Message Global Unique Identifier (generated by PPS) to retrieve raw data for a message	If it is specified and a messages is found, the response body contains the message's raw data instead of a JSON document
<b>dlpviolation</b>	optional	"t" or "details"	dlpviolation=t dlpviolation=details	Fetch the dlp violation data  "t" displays the number of Smart Identifiers or dictionaries found  "details" displays the actual Smart Identifiers or dictionaries found	
<b>messagestatus</b>	optional	"t"	messagestatus=t	Fetches message status and comments	

## Request

### HTTP Body

None

## Response

The response body of a successful request contains a JSON document. The result set returns are sorted by date/time in ascending order (newest first). Paged results (a series of result sets) are not supported. Data in the response is in UTF-8 format.

## HTTP Body

```
{
  "count" : 2,
  "records" : [
    {
      "processingserver" : "...",
      "date" : "2018-01-12 23:38:00",
      "subject" : "Up to $1000 Payday Loans - Instant Approvals",
      "messageid" : "YATQ2OUBWC3MFA2YUTDH.448380834@sam3.greatpromo.net",
      "folder" : "Quarantine",
      "size" : "6496",
      "rcpts" : [
        "foo@bar.com"
      ],
      "from" : "john@doe.com",
      "spamscore" : "100",
      "guid" : "vT_SjEF1Llfn9gML8YZzpVPUukjXQcSG",
      "host_ip" : "[10.24.40.4] [10.24.40.4]",
      "localguid" : "6:6:239"
    },
    {
      "processingserver" : "...",
      "date" : "2018-01-12 23:37:18",
      "subject" : "Up to $1000 Payday Loans - Instant Approvals",
      "messageid" : "NKC25LKOCDR72DBE06JF.225345479@email1.valuevalet.com",
      "folder" : "Quarantine",
      "size" : "6143",
      "rcpts" : [
        "ul@test.com"
      ]
    }
  ]
}
```



```

    ],
    "from" : "john@doe.com",
    "spamscore" : "100",
    "guid" : "edlp0qU9YXkWB5ngat91i9HUv6J-K-ep",
    "host_ip" : "[10.24.40.4] [10.24.40.4]",
    "localguid" : "6:6:4"
  }
],
"meta" : {
  "fqin" : "...",
  "queryid" : "...-1515801851",
  "query_params" : {
    "pretty" : "1",
    "subject" : "Up*",
    "from" : "john@doe.com"
  },
  "limit" : 1000,
  "duration" : "0.02"
}
}

```

### Field Descriptions

Field	Type	Description
meta	JSON blob	meta data
queryid	string	query ID specified in the request
duration	string	processing time
fqin	string	fully-qualified instance name of the server processing the request
limit	int	maximum number of records returned

Field	Type	Description
query_param	JSON blob	set of records
count	int	number of records found
records	JSON blob	set of records
from	string	sender address (envelope)
rcpts	JSON array	a list of envelope recipient addresses
subject	string	MIME-decoded message subject
date	UTC time string	timestamp for when the message was filtered by the server
spamscore	int	spam score
processingserver	string	fully-qualified instance name for the server processing the message
folder	string	Quarantine folder name
host_ip	string	the sending hostname and IP address
messageid	string	message ID
guid	string	the PPS global unique identifier used to retried the raw message
localguid	string	the PPS global unique identifier used to perform Quarantine actions
dlpviolation	JSON blob	list of locations (for example, body, subject) and/or attachment filenames (for example, job_description.doc) containing a list of Smart Identifiers and/or dictionary matches and exact_matches (Exact Data Match). See the dlpviolation JSON blob example below.
messagestatus	JSON blob	list of message status and comments. See the messagestatus JSON blob example below.

### Examples for dlpviolation JSON blobs

dlpviolation=t

This returns the *matches\_count* and *exact\_matches\_count*.

```
"dlpviolation" : {
```

```

"body" : {
  "ccard" : {
    "matches_count" : "1"
  },
  "ssn" : {
    "matches_count" : "2",
    "exact_matches_count" : "1"
  }
},
"subject" : {
  "ccard" : {
    "matches_count" : "1",
    "exact_matches_count" : "1"
  },
  "ssn" : {
    "matches_count" : "2",
    "exact_matches_count" : "1"
  }
},
"job_description.doc" : {
  "ssn" : {
    "matches_count" : "2"
  },
  "ICD-9" : {
    "matches_count" : "3"
  }
}
},

```

dlpviolation=details

Returns the matches for Smart Identifiers or Dictionary terms that triggered the rule.

```
"dlpviolation" : {
  "body" : {
    "ccard" : {
      "matches" : [
        "2627 5234 8190 3516"
      ]
    },
    "ssn" : {
      "matches" : [
        "123-45-6789",
        "111-11-1111"
      ],
      "exact_matches" : [
        "123-45-6789"
      ]
    }
  },
  "subject" : {
    "ccard" : {
      "matches" : [
        "6227-5234-8190-3511"
      ],
      "exact_matches" : [
        "6227-5234-8190-3511"
      ]
    },
    "ssn" : {
      "matches" : [
```

```

        "123-45-6789",
        "111-11-1111"
    ],
    "exact_matches" : [
        "123-45-6789"
    ]
}
},
"job_description.doc" : {
    "ssn" : {
        "matches" : [
            "575 08 1234",
            "561 31 1234"
        ]
    },
    "ICD-9" : {
        "matches" : [
            "breast engorgement in newborn",
            "cervix uteri",
            "female genital organs"
        ]
    }
}
},

```

**Example for messagestatus JSON blob**

```

"messagestatus" : [
    {
        "statusdesc" : "Escalated",
        "comment" : "this message is an EDM one",
        "status" : "2",
        "date" : "2017-11-14 20:51:59",
        "user" : "admin",
        "id" : "1"
    },

```

```
{
  "statusdesc" : "New",
  "comment" : "",
  "status" : "0",
  "date" : "2017-11-13 19:40:27",
  "user" : "",
  "id" : "0"
}
```

#### Return Codes:

Code	Description
200	Success
400	Invalid Argument
400	Invalid Time Range
404	Method Not Allowed
500	Database Unavailable
503	Maximum Requests Exceeded

#### Examples

GET messages that were inserted into the “Quarantine” folder before 2011-05-3 23:04:42 by *rcpt* ends with “bar.com” and *subject* contains “Free shipping.”

```
GET
https://localhost:10000/rest/v1/quarantine?rcpt=*bar.com&subject=*Free%20Shipping*&enddate=2011-05-3%2023%3A04%3A42
```

GET messages that were inserted in the “Attachment Defense” folder in the last 24 hours by *from* equals “foe@bar.com”.

```
GET
https://localhost:10000/rest/v1/quarantine?from=foo@bar.com&folder=Attachment%20Defense
```

GET raw message by GUID (Global Unique Identifier)

```
GET
https://localhost:10000/rest/v1/quarantine?guid=Tpkzn31onjJyU2n8nQPuk0nsjtf1NQJl
```

### ***PUT Method***

Inserts a message into the Quarantine. Not supported.

### ***POST Method***

To release, resubmit, forward, move, or delete messages.

## Request

Field	Description	Required or optional	Example
action	<p>Action to process the message or messages:</p> <p><b>release</b> – release the message to the email infrastructure without further scanning. The message remains in the folder and will be moved to the “deletedfolder” if specified.</p> <p><b>resubmit</b> – resubmits the message to the filtering modules. The message is removed from the folder and will not be moved to any folder, even if “deletedfolder” is specified.</p> <p><b>forward</b> – forwards the message to another recipient. The message remains in the folder and will be moved to the “deletedfolder” if specified.</p> <p><b>move</b> – moves the message to the specified “targetfolder”.</p> <p><b>delete</b> – deletes the message from the Quarantine. The message is removed from its folder and is moved to the “deletedfolder” if specified.</p>	required	“action”.”release”



Field	Description	Required or optional	Example
folder	Folder name: folder where the message is stored.	required	"folder": "HIPAA"
localguid	Message guid(s): comma-separated guides in folder_id:table_id:dbase_ id format (or in URL format if deployed in Cloud Quarantine mode).	required	"localguid": "2:2:8"  "localguid": "https://<cqshost>/v1/o bject/5ababd60-3b22-3803-82dd- 8d83498e5172/g5fsnj_sDLMk9hE CaJwccxqP6lQkr5k6"
deletedfolder	Deleted folder name:  Messages will be moved to this folder when <b>release</b> , <b>forward</b> , or <b>delete</b> are used.	optional  If specified, the folder for deleted messages must be for quarantined messages from the same type of module. For example, you cannot send deleted spam messages to a folder for deleted DLP Incidents, and vice versa.	"deletedfolder": "Deleted Incidents"
targetfolder	Target folder name:  The target folder name when <b>move</b> is used for moving messages between folders.	Required only if used with <b>move</b> .  If specified, the folder for moved messages must be for quarantined messages from the same type of module. For example, you cannot move spam messages to a folder for DLP Incidents, and vice versa.	"targetfolder": "PCI"
scan	Use "t" to rescan a message by the DLP and Attachment Defense filtering modules when <b>release</b> is used.	Optional and if specified, is only used by <b>release</b> .	"scan": "t"

Field	Description	Required or optional	Example
brandtemplate	<p>When Encryption is licensed, uses this Branding Template when an encrypted message is released. Used by <b>release</b>.</p> <p>If specified with "sender" the message will be encrypted and released using the sender's Branding Template.</p> <p>The Branding Templates are listed on the <b>System &gt; End User Services &gt; Branding Templates</b> page in the management interface (admin GUI).</p>	Optional and if specified, is only used by <b>release</b> .	"brandtemplate": "System_Default_Encryption_Branding"
securitypolicy	<p>The Secure Reader response profile to use when <b>release</b> is used for an encrypted message.</p> <p>If specified with "sender" the message will be encrypted and released using the sender's response profile.</p> <p>The Response Profiles are listed on the <b>Information Protection &gt; Encryption &gt; Response Profiles</b> page in the management interface (admin GUI).</p>	Optional and if specified, is only used by <b>release</b> .	"securitypolicy": "System_Response_Profile"
subject	<p>New subject. Overwrite the original Subject for the message with a new Subject. Used with <b>forward</b>.</p>	Optional and if specified, only used by <b>forward</b> .	"subject": "Subject: Testing 123."

Field	Description	Required or optional	Example
appendoldsubject	<p>Use "t" to append original Subject to the string specified in the "subject" field.</p> <p>Append the original Subject when the action is <b>forward</b>.</p>	Optional and if specified, only used when "subject" is specified.	"appendoldsubject": "t"
from	<p>The envelope <code>from</code> email address.</p> <p>Specify an envelope <code>from</code> email address when the action is <b>forward</b>.</p>	<p>Optional and if specified, only used by <b>forward</b>.</p> <p>If not specified, the default mailer <code>from</code> is used. See the <code>filter.cfg</code> key "com.proofpoint.filter.smt.p.default.from". If this key is not configured, the original envelope <code>from</code> is used.</p>	"from": "foo@bar.com"
headerfrom	<p>The header <code>from</code> email address.</p> <p>Specify the header <code>from</code> email address when the action is <b>forward</b>.</p>	<p>Optional and if specified, only used by <b>forward</b></p> <p>If neither "from" and "headerfrom" are specified, the default mailer <code>from</code> address is used.</p> <p>See <code>filter.cfg</code> key "com.proofpoint.filter.smt.p.default.from". If this key is not specified, the message header <code>from</code> is used.</p>	"headerfrom": "foo@bar.com"
to	<p>Recipient email address(es).</p> <p>Specify a comma-separated list of recipient email addresses when the action is <b>forward</b>.</p>	Optional and if specified, only used by <b>forward</b> .	"to": "john@doe.com"

Field	Description	Required or optional	Example
comment	<p>New message body.</p> <p>Specify a new message body when the action is <b>forward</b>.</p> <p>The original message is sent as an attachment.</p>	Optional and if specified, only used by <b>forward</b> .	"comment": "Hi Nancy, please see the attachment."

## Response

When the operation is successful, the response will be an HTTP 200 or 204 with the following body:

```
{ "status": "<status string>" }
```

## Examples

Release the message with *localguid* 14:14:7 from the PCI folder. When releasing, re-scan the message and encrypt the message with sender's Branding Template and Response Profile.

```
POST https://localhost:10000/rest/v1/quarantine { "deletedfolder":
"Deleted Incidents", "folder": "PCI", "localguid": "14:14:7", "action":
"release", "scan": "t", "brandtemplate": "sender", "securitypolicy":
"sender" }
```

Resubmit the message with *localguid* 14:14:4 from PCI folder to the filtering modules.

```
POST https://localhost:10000/rest/v1/quarantine { "folder": "PCI",
"localguid": "14:14:4", "action": "resubmit" }
```

Forward the message with *localguid* 14:14:3 from the PCI folder to another recipient. When building the new message, set the envelope from address from "john@doe.com" to "foo@bar.com", change the subject to "Redirecting..." and add the message body "Hello World". The original message will be attached to the forwarded message.

```
POST https://localhost:10000/rest/v1/quarantine { "deletedfolder":
"Deleted Incidents", "folder": "PCI", "localguid": "14:14:3", "action":
"forward", "from": "john@doe.com", "to": "foo@bar.com", "subject":
"Redirecting...", "comment": "Hello World" }
```

Move the message with *localguid 14:14:6* and the message with *localguid 14:14:9* from the PCI folder to the HIPAA folder.

```
POST https://localhost:10000/rest/v1/quarantine { "targetfolder" :
"HIPAA", "folder" : "PCI", "localguid" : "14:14:6,14:14:9", "action" :
"move" }
```

Delete the message with *localguid 14:14:2* from the PCI folder.

```
POST https://localhost:10000/rest/v1/quarantine { "folder" : "PCI",
"localguid" : "14:14:2", "action" : "delete" }
```

## ***DELETE Method***

To delete a message, use the POST method.

## ***Error Handling***

The REST API will return an error message

### ***Error Message Body***

```
<error message>
```

### ***Possible Error Messages***

Method Not Allowed

Maximum Requests Exceeded

Invalid Argument

Invalid Folder

No Message Found

Invalid Time Range

Invalid argument: to

Invalid argument: folder

Invalid argument: deletedfolder

Invalid argument: targetfolder

Invalid argument: folder and deletedfolder cannot be the same

Invalid argument: folder and targetfolder cannot be the same

Invalid argument: folder and deletedfolder type mismatches

Invalid argument: folder and targetfolder type mismatches

Cannot find message: <localguid>

Failed to delete message: <localguid>

Failed to resubmit message: <localguid>

Failed to release message: <localguid>

Failed to forward message: <localguid>

Failed to move message: <localguid>

## Exact Data Match REST API

The Exact Data Match (EDM) API allows administrators to connect to a database and upload data files to the Regulatory Compliance Exact Data Match file repository. This API was implemented in PPS release 8.10.

### Authentication

The administrator must have a Role that includes access to the Exact Data Match API. First create a *Role* (**System > Administrator > Roles**) of **Type API** that includes the *Exact Data Match* Managed Module. Then edit an existing administrator or add a new administrator with that *Role* assigned to him or her.

### Resource Identifier

/rest/v1/edm

### Available Methods

GET	Yes	Handles listing and deleting files
PUT	No	
DELETE	Yes	Deletes a single file by fileid
POST	Yes	Uploads new EDM files Deletes multiple files by their fileids

### Operations

Operation	HTTP Option	Path	Purpose	Parameters	Error Codes	Example
List	GET	/v1/edm/files	To list all EDM files that have been uploaded to a given system	None	200	curl -k -X GET "https://localhost:10010/rest/v1/edm/files -u <apiadmin>:<password>

Operation	HTTP Option	Path	Purpose	Parameters	Error Codes	Example
<b>Delete a single file</b>	DELETE	/v1/edm/<fileid>	To delete one EDM file by fileid from a given system	<fileid> in the path	Invalid fileid:400 Bad Request  fileid in use: 200 with Status: Skipped  File in use:200 with Status: Skipped  Correct fileid that is not in use  200 with status Deleted  Invalid fileid: 400	<pre>curl -k -X DELETE "https://localhost:10010/rest/v1/edm/1" -u &lt;apiadmin&gt;:&lt;password&gt;</pre>
<b>Delete multiple files</b>	POST	/v1/edm/files/batch	To delete multiple EDM files given their fileids	fileids:<1,2,3>	Code 200 with status for individual fileid  Code 400 if fileid parameter is missing	<pre>curl -k -X POST -u &lt;apiadmin&gt;:&lt;password&gt; -h "Content-Type:application/json" "https://localhost:10010/rest/v1/edm/files/batch" -data '{"fileids":"1,2,3,4"}'</pre>



Operation	HTTP Option	Path	Purpose	Parameters	Error Codes	Example
Upload	POST	/v1/edm?filename=<filename.csv>	<p>To upload an EDM CSV file using POST</p> <p>The parameter 'filename' takes the display name of the CSV file.</p>	Filename=<filename.csv>	<p>Code 200 for successful upload</p> <p>Code 400 for duplicate column name upload or invalid filename or invalid column name</p> <p>Code 500 if the database import failed and the file creation failed – these are corner cases and cannot be easily reproduced</p>	<pre>curl -k -X POST "https://localhost:10010/rest/v1/edm?filename=ssn_cc.csv" -u &lt;apiadmin&gt;:&lt;password&gt; --data-binary@ssn_cc.csv -H 'Content-type:text/plain;charset=utf-8'</pre> <pre>curl -k -X POST "https://localhost:10010/rest/v1/edm?filename=ssn_cc.csv" -u &lt;apiadmin&gt;:&lt;password&gt; --data-binary@ssn_cc.csv -H 'Content-type:text/csv;charset=utf-8'</pre> <p>-Allows uploading of UTF-8 characters in the CSV file. Otherwise, you can use text/csv or text/plain.</p>

## Examples

### List all available EDM files

```
curl -k "https://localhost:10010/rest/v1/edm/files" -u <apiadmin>:<password>
```

```
{
  "COUNT": 1,
  "FILES": [
    {
      "Columns": "monday,friday,wednesday",
      "File ID": "4",
      "File Size": "3.97K",
      "File Size in Bytes": "4062",
      "Filename": "edm_sample5.csv",
      "Last Updated": "Mon Jan 22 11:25:03 2018",
      "Number of Records": "28",
```

```

        "Used by Smart Identifier": "ABA Routing Number"
    }
]
}

```

### **Delete a file that does not exist**

```
curl -k -X DELETE "https://localhost:10010/rest/v1/edm/files/1" -u
<apiadmin>:<password>
```

```

{
  "COUNT": 1,
  "RESULTS": {
    "FAILURE": [
      {
        "FILEID": "1",
        "REASON": "Fileid 1 does not exist",
        "STATUS": "SKIPPED"
      }
    ],
    "SUCCESS": []
  }
}

```

### **Delete a file that is in use**

```
curl -k X DELETE "https://localhost:10010/rest/v1/edm/files/4" -u
<apiadmin>:<password>
```

```

{
  "COUNT": 1,
  "RESULTS": {
    "FAILURE": [
      {
        "FILEID": "4",
        "REASON": "Fileid 4 is in use.",
        "STATUS": "SKIPPED"
      }
    ],
    "SUCCESS": []
  }
}

```

### Delete multiple files

```
curl -k -X POST -u <apiadmin>:<password> -H "Content-Type:application/json"
"https://localhost:10010/rest/v1/edm/files/batch?pretty" --data
'{"fileids": "1,4,abc,11"}' | python -m json.tool
```

```
{
  "COUNT": 4,
  "RESULTS": {
    "FAILURE": [
      {
        "FILEID": "1",
        "REASON": "Fileid 1 does not exist",
        "STATUS": "SKIPPED"
      },
      {
        "FILEID": "4",
        "REASON": "Fileid 4 is in use.",
        "STATUS": "SKIPPED"
      },
      {
        "FILEID": "abc",
        "REASON": "Fileid is invalid",
        "STATUS": "SKIPPED"
      }
    ],
    "SUCCESS": [
      {
        "FILEID": "11",
        "STATUS": "DELETED"
      }
    ]
  }
}
```

### Upload a file

```
curl -k "https://localhost:10010/rest/v1/edm?filename=ssn_csv" -u
<apiadmin>:<password> --data-binary @ssn_csv.csv -H 'Content-
type:text/plain; charset=utf-8'
```

```
{"info": "Exact Data file imported  
successfully", "status": "upload_successful"}
```

***Upload a file that contains a duplicate column***

```
curl -k "https://localhost:10010/rest/v1/edm?filename=example" -u  
<apiadmin>:<password> --data-binary @sansample1.csv -H 'Content-  
type:text/plain; charset=utf-8'
```

Error code: 400 Bad Request

Upload failed, file contains duplicate column names that already exists  
in edm\_sample5.csv"

## Administrator Management REST API

Use the Administrator Management REST API to create, modify, and delete administrators, and to query administrator records with or without details. This API was implemented in PPS release 8.10.

### *General API features*

Authentication: basic authentication via PPS (requires API User with "end user API") and optional certificate based authentication. Refer to the [Authentication](#) section for more information.

Once configured the end user REST APIs are available through the admin port (default 10000).

### *Payload*

The following payloads are supported:

Accept	_format
text/xml	xml
application/json	json

The client must specify the following headers:

Host

**Accept** (application/json)

User-Agent

**Content-Type** (when there is content)

### *Resource Identifier*

Describes a single "administrator" in the target Proofpoint Protection Server system

/rest/vi/administrator

## GET Method

Gets the administrator list or specific administrator details.

Part	Required	Type	Example	Description
Id	no	string	foo	Administrator ID to get administrator profile.

### Response

The GET response returns all administrator records with details in either XML or JSON format (based on *Accept* header or *\_format* parameter). If the admin ID is specified in the GET request, specific administrator details are returned.

**Important:** The admin password is never returned in the response.

### Return Codes

Code	Description
200	Success
404	Not found
500	Server error

### HTTP Body

```
{
  "result": 200,
  "data": {
    "admininfo": {
      "last_changed": "1516175584",
      "id": "foo",
      "suborg": "suborg",
      "phone": "9090909090",
      "email": "foo@example.com",
      "comment": "Test Comment",
      "full_name": "John Doe",
```

```

        "role": "quarantine"
    }
}
}

```

### Field Description

Field	Type	Example	Description
full_name	String	"Steve John"	Full name of administrator
id	String	"foo"	Unique ID assigned to administrator
email	String	"foo@example.com"	Administrator email address
phone	Int	9896098960	Administrator Phone number
comment	String	"Added Comment"	Added comment in administrator profile.
role	String	"Quarantine"	Assigned role to administrator
suborg	String	"Organization"	Administrator Organization
last_changed	int	1516174025	Last updated timestamp

### Examples

#### Get admin details

```
GET https://localhost:10000/rest/v1/administrator/{id}
```

```

{
  "result": 200,
  "data": {
    "admininfo": {
      "last_changed": "1516175584",
      "id": "foo",
      "suborg": "suborg",
      "phone": "9090909090",

```

```

        "email": "foo@example.com",
        "comment": "Test Comment",
        "full_name": "John Doe",
        "role": "quarantine"
    }
}
}

```

### List all administrators

GET https://localhost:10000/rest/v1/administrator

```

{
  "result": 200,
  "data": {
    "admins": [
      {
        "last_changed": "1516174151",
        "id": "foo",
        "suborg": "suborg",
        "phone": "9896098960",
        "email": "foo@example.com",
        "comment": "Test comment",
        "full_name": "Foo",
        "role": "quarantine"
      },
      {
        "last_changed": "1516174151",
        "id": "bar",

```



```

        "suborg": "suborg",
        "phone": "9896198961",
        "email": "bar@example.com",
        "comment": "Lorem Ipsum",
        "full_name": "Bar",
        "role": "quarantine"
    }
]
}
}

```

## PUT Method

Modifies an administrator record. Supports the same options as creating a new administrator but requires the admin ID in the request (see the example).

### Request

Field	Type	Example	Description
full_name	String	"John Doe"	Full name of administrator
email	String	"john@example.com"	Administrator email address
phone	Int	9896098960	Administrator Phone number
comment	String	"Added comment to profile"	Added comment in administrator profile.
password	String	"\$cre7Pa\$\$w0rd"	Password to access. It must have 7 or more characters including at least 1 digit and 1 special character
role	String	"Quarantine"	Assigned role to administrator
suborg	String	"Organization"	Administrator Organization

Field	Type	Example	Description
password_hash	String	"\$apr1\$LS\$mk13unDCeJvyXo/naojx51"	Encrypted password. It will override password field if provided both.

**Response**

It will return the full administrator profile (see GET) when the update is successful.

**Return Codes**

Code	Description
200	Success
400	Bad request
403	Forbidden
404	Not found
500	Server error

**Example****Set the full name and address given an Administrator ID**

```
PUT https://localhost:10000/rest/v1/administrator/{id}
{ full_name : 'John Doe', email: 'john@example.com' }
```

```
{
  "data": {
    "admininfo": {
      "suborg": "suborg",
      "full_name": "John Doe",
      "phone": "9090909090",
      "last_changed": "1516175909",
      "email": "john@example.com",
      "comment": "comment",
      "id": "foo",
```

```
        "role": "quarantine"
      }
    },
    "message": "Administrator updated successfully.",
    "result": 200
  }
```

## POST Method

Create a new administrator.

### Request

Field	Type	Example	Description
id	String	"foo"	Administrator profile ID
full_name	String	"John Doe"	Full name of administrator
email	String	"john@example.com"	Administrator email address
phone	Int	9896098960	Administrator phone number
comment	String	"Added comment to profile"	Added comment in administrator profile.
password	String	"\$cre7Pa\$w0rd"	Password to access. It must have 7 or more characters including at least 1 digit and 1 special character
role	String	"Quarantine"	Assigned role to administrator
suborg	String	"Organization"	Administrator Organization
password_hash	String	"\$apr1\$LS\$mk13unDCeJvyXo/naojx51"	Encrypted password. It will override password field if provided both.

### Response

When the operation is successful the response will return the full administrator profile.

Code	Description
200	Success
400	Bad request
403	Forbidden
409	Conflict / Already exists
500	Server error

**Example**

```
POST https://localhost:10000/rest/v1/administrator
{ id: 'foo', password: '$ecre7Pa$$w0rd', full_name : 'John Doe',
email: 'john@example.com', role: 'quarantine', comment:"comment",
phone:"9090909090" }
```

```
{
  "data": {
    "admininfo": {
      "suborg": null,
      "full_name": "John Doe",
      "phone": "9090909090",
      "last_changed": "1516175909",
      "email": "john@example.com",
      "comment": "commment",
      "id": "foo",
      "role": "quarantine"
    }
  },
  "message": "Administrator created successfully.",
  "result": 200
}
```

**DELETE Method**

Deletes an administrator profile. Requires the administrator ID in the DELETE request (see the example).

**Request**

DELETE requires the administrator ID.

**HTTP Body**

No options can be specified.

**Response**

Upon successful deletion of the admin profile the response will be a HTTP 200 with the following body:

```
{"result":200,"message":"Administrator test deleted successfully."}
```

Code	Description
200	Success
400	Bad request
403	Forbidden
404	Not found
500	Server error

**Example**

**Delete an administrator by AdministratorID.**

```
DELETE https://localhost:10000/rest/v1/administrator/{id}
```

```
{"result":200,"message":"Administrator test deleted successfully."}
```

**Error Handling**

The REST API will return detailed information regarding HTTP 4XX errors.

### **Error Message Body**

```
{
  "errors": {
    "invalidarguments": [
      {
        "error": "Administrator ID must be specified"
      },
      {
        "error": "Password must be specified to create an
admin"
      },
      {
        "error": "Role must be specified to create an admin"
      }
    ]
  },
  "result": 400
}
```

### **Examples**

#### **GET request to a nonexistent administrator**

```
GET
https://localhost:10000/rest/v1/administrator/{nonexistent_id}
```

results in an HTTP 404 error with the following response body:

```
{
  "result": 404,
  "message": "Administrator does not exist"
}
```

**Incorrect PUT/DELETE request**

```
GET https://localhost:10000/rest/v1/administrator
```

results in an HTTP 404 error with the following response body:

```
{
  "result": 400,
  "message": "Administrator ID must be specified"
}
```

**POST request with an ID that already exists**

```
POST
https://localhost:10000/rest/v1/administrator/?id=foo&password=pa
ssword&role=role
```

results in an HTTP 404 error with the following response body:

```
{
  "status": 409,
  "message": "Admin foo already exists"
}
```

**PUT request with an invalid email address**

```
PUT
https://localhost:10000/rest/v1/administrator/foo?email=testemail
```

results in an HTTP 404 error with the following response body:

```
{
  "errors": {
    "invalidarguments": [
      {
        "error": "Invalid email address."
      }
    ]
  }
}
```



```

    }
  ]
},
"result": 400
}

```

### PUT request to update a restricted role

```

PUT
https://localhost:10000/rest/v1/administrator/foo?email=foo@example.com&role=root

```

results in an HTTP 403 error with the following response body:

```

{
  "status": 403,
  "message": "You are not allowed to add/update this profile.
Probably you are trying to add/modify with the restricted role."
}

```

### Field Validation

Name	Format
id	Required without spaces
password	Required
full_name	
email	Valid email address
phone	
comment	

### CURL Examples

Try the API using curl. This section contains examples. Note that if you are testing against a server without a cert, you will need to specify the --insecure flag. This flag should not be used with production servers.

**Create a new administrator**

```
curl -u apiuser:userpw1! -X POST -H "Content-Type:application/json" -d '{"id":"foo","password":"$cre7Pa$$w0rd","email":"foo@example.com"}' https://localhost:10000/rest/v1/administrator
```

**Update an administrator**

```
curl -u apiuser:userpw1! -X PUT -H "Content-Type:application/json" -d '{"email":"foo@example.com"}' https://localhost:10000/rest/v1/administrator/{id}
```

**Delete an administrator**

```
curl -u apiuser:userpw1! -X DELETE https://localhost:10000/rest/v1/administrator/{id}
```