Introduction

The Secure Messaging Platform Application Programming Interface (API) allows third-party developers to request data and initiate processes with a specific customer Portal. This data can then be used or manipulated outside of the Secure Messaging Platform, and in some cases, can be returned to the Secure Messaging platform.

Client Services API Framework Sample

This sample client application framework (SDK) is a Visual Studio 2010 .Net 4 class library that implements the most common commands available in the API.

- Sample .NET project: http://help.secure-messaging.com/docs/Secure_Messaging_API_SDK_1012.zip

API Services

To execute an operation using the API, you need to submit an HTTP POST request to the URL that corresponds to the Secure Messaging platform you are accessing containing the operation that you wish to perform as a JSON object in the body of the request. The output of the command is returned to you as a JSON object in the body of the HTTP response.

In order to get started, you will need to obtain the following information from your account manager (the information provided in this document is only a sample and does not work):

- penGUID = 42e28744-08a8-4ce9-ab99-e923198923b4
- penName = ABC Medical Secure Email
- servicesUrl = https://secure-messaging.com/abcmedical/services/api.aspx
- homeUrl = https://secure-messaging.com/abcmedical/web/

where https://secure-messaging.com corresponds to the URL of the Secure Messaging platform server URL hosting the customer platform and the 'code' correspond to the Secure Messaging platform code used to access the Secure Message Center (Webmail).

Authentication and Security

The API is only available through a secure HTTPS channel. Accessing the API via a regular HTTP channel will result in a 500 HTTP error code returned by the server.

Each API request that you send need to contain a session token, which will be use to authorize access to the operation specified in the API request. Session tokens are only available to active users and administrative users of the Secure Messaging platform, and those tokens only authorize operations within the same Secure Messaging platform.

By default, session tokens expire after 60 minutes. As such, you will need to submit a request to the above URL at least once every hour.

NOTE: It is recommended that you keep the token in memory rather than writing it to a file.

The API also includes an automatic throttling mechanism to avoid abuse. This mechanism will ban specific users, computers or IP addresses that query the system too frequently or in an improper manner. Note that there are other criteria available to administrators to allow / ban other users or IP addresses.

JSON Request / Response format

Accessing the API must be done sending HTTP POST requests to the corresponding Secure Messaging platform server with the body of the request using a JSON object. The response will also be represented as a JSON object.

The API’s request/response JSON objects follow the specification described below. For details about the syntax supported by the JSON values refer to Appendix A: JSON Data Types.
Request structure

An API Request corresponds to an unnamed JSON object with two mandatory objects within: header and data.

The example below shows the body of the HTTP request used by a third party application to authenticate with a Secure Messaging platform and retrieve a session token.

```
01 POST /services/api.aspx HTTP/1.1
02 
03 {
04   "header": {
05     "command": "login",
06     "timestamp": new Date(1217290505123),
07     "clientName": "OL12.0.0.6316",
08     "clientVersion": "1.6.2229"
09   },
10  "data": {
11    "emailAddress": "user.one@corp-national.com",
12    "password": "p455w0rd"
13  }
14 }
```

**NOTE:** The text in this example has been formatted for readability purposes only.

Header section

The Request Header is a mandatory JSON object named header with the following JSON values as parameters:

<table>
<thead>
<tr>
<th>token</th>
<th>String / Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The session validation token returned once you've logged into the system.</td>
</tr>
<tr>
<td></td>
<td>Authentication commands do not require this value, but all messaging-related commands need it.</td>
</tr>
<tr>
<td></td>
<td>The token is valid only for the length of the session. Third party applications must keep the session alive (using cookies) in order to be able to use the API Services.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>command</th>
<th>String / Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>One of the possible commands supported as listed below in the API Command section.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>timestamp</th>
<th>DateTime / Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The date and time of the request expressed in GMT/UTC format.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>clientName</th>
<th>String / Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies the name of the third party application requesting commands. This is a mandatory field but its value is not restricted to any specific convention.</td>
</tr>
<tr>
<td></td>
<td>You are free to use this value for your own purposes and follow your own notation. However, the API can respond to a specific list of approved applications in a customized way. Some applications, such as the Secure MS Outlook Toolbar have available extended parameters and some specific commands available to it.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>clientVersion</th>
<th>String / Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicates the third party application version number. This value helps the application synchronizing with the appropriate version of the API.</td>
</tr>
</tbody>
</table>

Optional parameters do not need to be included in the request. For example, the login command shown in the example above does not require a token parameter as this value will be returned by the corresponding response.
Data section

The Request Data is a mandatory JSON object named data with a list of JSON values as input parameters. Each individual command specifies if the corresponding request parameter is mandatory or optional and the data type of the parameter.

Response structure

An API Response corresponds to an unnamed JSON Object with two objects within: header and a data.

The example below shows the response that is sent when a third party application authenticates with a Secure Messaging platform and receives a session token back.

```
01 200 OK
02
03 {  
04   "header": {  
05     "command": "login",  
06     "timestamp": new Date(1217290505123),  
07     "errorCode": 0,  
08     "errorDesc": "",  
09     "serverVersion": "1.6.2229"
10   }  
11   "data": {  
12     "token": "Y642AV2B-73F3-479B-81FE-FA48H9835BF2",
13   }
14 }
```

Header section

The Response Header is a JSON object named header with the following JSON values as parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>String / Mandatory</td>
<td>Command is one of the possible commands listed below in the API Commands section. It corresponds to the command sent in the request.</td>
</tr>
<tr>
<td>timestamp</td>
<td>DateTime / Mandatory</td>
<td>The date and time of the response expressed in GMT /UTC format.</td>
</tr>
<tr>
<td>errorCode</td>
<td>Number / Mandatory</td>
<td>Indicates the error code as specified by the Secure Messaging platform server. If the request was processed successfully the system will return 0 (zero). If there was a problem processing the request command, the response will include the appropriate error code in this output parameter. Each command describes the error codes available.</td>
</tr>
<tr>
<td>errorDesc</td>
<td>String / Mandatory</td>
<td>Error description corresponding to the error code included in the previous parameter. If there are no errors, this parameter returns an empty string. The error message returned by the server includes only generic information about the issue found. Developers are encouraged to add to this information in the context of where the problem occurred. In some occasions it is a good idea to replace the message altogether by an appropriate contextual message.</td>
</tr>
<tr>
<td>serverVersion</td>
<td>String / Mandatory</td>
<td></td>
</tr>
</tbody>
</table>

PAGE 4 of 65
Indicates the API Services’ version number.

As with the request specification, optional parameters do not need to be included in the response. Some extensions to the Secure Messaging platform Server may not include optional parameters or some specific clients require extra parameters that other applications do not.

**Data section**

The Response Data section is an optional JSON object named data with a list of JSON values as output parameters.

Some commands do not return any data. In these cases the data section may be omitted.

Each command specifies if the response parameters are mandatory or optional and the data type of each parameter.

Typically, when the system finds an error for a given request, the response contains only a header section with the appropriate error code and description with an empty data section. However, some error codes returned by the server may include extra information, in which case the data section is also included in the response (e.g., the banned word when this rule is broken by a message that contains banned words).

**HTTP Headers**

All API requests must include the following HTTP headers:

**Content Type**

The POST request must specify the value application/x-www-form-urlencoded for the Content-Type header.

**Cookies**

For security reasons, interacting with the API requires that the client stores the HTTP Cookie sent by the Secure Messaging platform server the first time you initiate an HTTP connection.

Also, preserving the HTTP session using this ASP.NET Cookie helps to increase the security and performance of the system.

Subsequent calls to the same Secure Messaging platform using this token require you to pass the cookie in the HTTP headers in order to preserve the session state in the server. Failure to do this will result in new server sessions created for each HTTP connection to the server rendering the token session useless.

**email2 Command X-Header**

As an extra control mechanism the API requires you to include the command code as an extended HTTP header named X-email2Cmd. This is the same command code included in the body of the request in the JSON header.

Consider the following examples that shows the HTTP headers exchange for the login and getMember commands (Notice that in this example the focus is on the HTTP headers, not the actual command):

First Request:

```
01 POST /services/api.aspx HTTP/1.1
02 User-Agent: email2Client
03 X-email2Cmd: login
04 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
05 Host: vmqa1:9999
06 Content-Length: 234
07 Expect: 100-continue
08 Connection: Close
09
10 {"data":{"emailAddress":"user.four@corp-national.com","password":"1324"},
   "header":{"command":"login","timestamp":new Date(1230743250349),"clientName":"apiNetTest","clientVersion":"0.1.0.0"}}
```
In this case, in line number 3 we specify as an X-Header attribute the command code also contained in the body of the request starting in line number 10. Line 4 shows how to specify the content-type header.

Notice that the host vmqa1:9999 (line 5) is only an example.

Response:

```
01 HTTP/1.1 200 OK
02 Server: Microsoft-IIS/5.1
03 Date: Wed, 31 Dec 2008 17:07:30 GMT
04 X-Powered-By: ASP.NET
05 Connection: close
06 X-AspNet-Version: 2.0.50727
07 Set-Cookie: ASP.NET_SessionId=mg4idv55a0e2kmuqjtw5prm; path=/; HttpOnly
08 Cache-Control: private
09 Content-Type: text/html
10 Content-Length: 166
11 "data":{"token":"1e13a3d-ebc3-44ed-9392-cb5dca16a704"},
   "header":{"command":"login","timestamp":new
       Date(1230743251148),"errCode":0,"errDesc":"","serverVersion":"1.6.2229"}
```

Notice line number 7 (in bold) where the server returns the HTTP Cookie with the ASP.NET session. You need to preserve this value along with the session token returned in line 12.

Subsequent calls need to preserve this value as shown in the following request to obtain the member's profile:

```
01 POST /services/api.aspx HTTP/1.1
02 User-Agent: DesktopAgent
03 Cookie: ASP.NET_SessionId=mg4idv55a0e2kmuqjtw5prm
04 X-email2Cmd: getMember
05 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
06 Host: vmqa1:9999
07 Content-Length: 225
08 Expect: 100-continue
09 Connection: Close
10 "data":{"emailAddress":"user.four@corp national.com"},"header":{"command":"getMember",
       "timestamp":new Date(1230743251241),"token":"1e13a3d-ebc3-44ed-9392-cb5dca16a704","clientName":"apiNetTest","clientVersion":"0.1.0.0"}}
```

In this second request, besides sending the command in the header as shown in line number 4, the ASP.NET Cookie is also passed back to the server in the appropriate format in line number 3.

**Error handling**

When integrating a third party application with the API you must look for successful commands at two levels.

First, you need to check the actual HTTP error code returned by the communication channel. In general, a successful response contains error code 200 (OK). The API does not make use of any other HTTP error code to report an error. If you receive an error code other than 200, you must follow a standard action as for any web application.

For more information about standard HTTP error codes you can review the following link: [www.w3.org/Protocols/HTTP/HTRESP.html](http://www.w3.org/Protocols/HTTP/HTRESP.html).

If the HTTP command succeeded (200, OK), the success of the command is included in the header JSON object of the Response.

Each command includes a section with the list of error codes supported by that command.
API Commands

The current version of the API supports the commands described in this section. The commands are grouped in three major sections:

- Authentication services
- Member provisioning services
- Secure messaging services

Each section includes a summary with a high-level description of the commands included in it and then subsections with the details for each command, including the request and response parameters supported by the command, followed by a list of error codes, and examples.

The description provided for each command focus on their data section. The header section of the JSON request/response follows the guidelines described in the JSON Request / Response section.

The examples provided for each command focus on the JSON body of the request/response, leaving out the HTTP headers used in the actual communication. The format of the actual HTTP request/response must follow the guidelines explained in the HTTP Headers section.

Authentication services

Before you can use any of the commands provided by the API, you need to obtain a valid session token with the corresponding Secure Messaging platform server.

The following commands are available in this section:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>Logs into a Secure Messaging platform (specified by the corresponding URL) using the supplied credentials. If successful, the Login command will return a session token which can be used to invoke other commands from the server during the length of the session. You must provide an active user’s email address to authenticate with the Secure Messaging platform.</td>
</tr>
<tr>
<td>Register Member*</td>
<td>Initiates the full registration process for a new user. The new user will receive a basic email message notification to confirm ownership of the email address. The email notification contains a keyword that must be used to complete the registration.</td>
</tr>
<tr>
<td>Confirm Registration*</td>
<td>Confirm registration for new user. Must be completed after registration and before login.</td>
</tr>
<tr>
<td>Forgot Password</td>
<td>Initiate the process to recover access to a Secure Account. It sends the corresponding user an email notification with a link to change their password. The user must be registered.</td>
</tr>
</tbody>
</table>

Login

The purpose of the login command is to authenticate the user and obtain a session token to be used in subsequent commands to consume the services provided by the API.

To obtain a session token you need a valid user’s email address or the userid of an authorized Secure Messaging platform administrator. Depending on the credentials used, the session token will give you access to different commands in the API.

The syntax described below corresponds to the authentication mechanism available for a user. Please refer to the Examples section below for the full syntax of the HTTP Request and Response.
Request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailAddress</td>
<td>String</td>
<td>Mandatory</td>
</tr>
<tr>
<td>password</td>
<td>String</td>
<td>Mandatory</td>
</tr>
<tr>
<td>requestAuthKey</td>
<td>Boolean</td>
<td>Optional</td>
</tr>
</tbody>
</table>

emailAddress

Valid user Email Address. The email address uniquely identifies the user.

Depending on the privileges assigned to the user, the session token obtained will give access to different commands and to specific information returned by these commands.

In general, a user can retrieve and change their own information and obtain limited information about other users. Some administrative users can also provision new users or update their profiles.

password

Password to login to the services.

requestAuthKey

Use the requestAuthKey parameter with value true to obtain an authorization key in the response which you can cache in your local application and use in lieu of emailAddress and password with the authenticate command.

Response parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

This is the global unique identifier for the session initiated with this Secure Messaging platform.

The session token is valid throughout the length of the session initiated with the server and must be used with all subsequent commands. Sessions expire after a given amount of time, at which time you need to obtain a new session token. Session tokens can be used only from the remote location from where it was obtained.

authKey

The authKey is an authorization key that can be used with the authenticate command to obtain a new session without having to pass an emailAddress and password.

Error codes

1090001 Unexpected / Unknown login error.

If you receive this error code please contact support immediately.

1090002 Unknown Secure Messaging platform.

The session token was assigned to a different Secure Messaging platform than the specified URL.

1090003 Wrong request format.

This is a valid JSON object, but the request has an invalid format. You may receive this message when you do not include mandatory fields or assigned an invalid value to one of the fields (e.g., wrong data type)

1090004 User is not registered.

1090005 User is in pending registration status.

1090006 Attempting to login / register on a ‘closed’ Secure Messaging platform. You must be invited to this Secure Messaging platform by an existing Professional user.

1090009 Email Address or Password incorrect.

If you receive this error code please contact support immediately.

1090010 System configuration error obtaining session token.

If you receive this error code please contact support immediately.
Examples

The following example attempts to obtain a session token with the test vmqa1:9999 PEN. In this case the full HTTP headers are included to show how the X-email2Cmd X-Header field is used.

Notice that the body of the request has been formatted for readability purposes only:

```
01 POST /services/api.aspx HTTP/1.1
02 User-Agent: email2Client
03 X-email2Cmd: login
04 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
05 Host: vmqa1:9999
06 Content-Length: 188
07 Expect: 100-continue
08 Connection: Close
09 {
10   "header": {
11     "command":"login",
12     "timestamp":new Date(1230922366313),
13     "clientName":"apiNetTest",
14     "clientVersion":"0.1.0.0"
15   },
16   "data": {
17     "emailAddress":"user.two@corp-national.com",
18     "password":"1234"
19   }
20 }
```

The line 3 in the previous example shows how you need to pass the X-email2Cmd X-Header. Line 4 specifies the Content Type required for the POST command. Also notice that in this case the line 2 shows how you can use your own User-Agent header.

The following response shows a successful login action and a session token is included in the response. The HTTP headers are also included to show the ASP.NET session returned by the server:

```
01 HTTP/1.1 200 OK
02 Server: Microsoft-IIS/5.1
03 Date: Fri, 02 Jan 2009 18:52:51 GMT
04 X-Powered-By: ASP.NET
05 Connection: close
06 X-AspNet-Version: 2.0.50727
07 Set-Cookie: ASP.NET_SessionId=as0rpwf0llqsyovowc55orrl; path=/; HttpOnly
08 Cache-Control: private
09 Content-Type: text/html
10 Content-Length: 166
11 {
12   "header": {
13     "command":"login",
14     "timestamp":new Date(1230922371548),
15     "errCode":0,
16     "errDesc":"
17     "serverVersion":"1.6.2229"
18   },
19   "data": {
20     "token":"1f4b5f33-d857-44e1-af65-d51baee2d325"
21   }
22 }
```
The following response shows the case when there is a problem with the user or Password used to authenticate this session (notice how no HTTP headers are included from this point on):

```json
{
  "header": {
    "command": "login",
    "timestamp": new Date(1230922371548),
    "errCode": 8863,
    "errDesc": "Email address or Password incorrect.",
    "serverVersion": "1.6.2229"
  }
}
```

The data section is not included in the response sent by the server. When a JSON object is null, you can omit this field in the JSON string. However, including it with an explicit null value is also a valid alternative. For example, the following response is equivalent to the previous example:

```json
{
  "header": {
    "command": "login",
    "timestamp": new Date(1230922371548),
    "errCode": 8863,
    "errDesc": "Email address or Password incorrect."
  },
  "data": null
}
```
Authenticate

Command code: authenticate

The purpose of the `authenticate` command is to authenticate the user and obtain a session token to be used in subsequent commands to consume the services provided by the Secure Messaging platform.

This command requires an authorization key that was previously obtained using the `login` command.

The purpose of this command is to allow third-party applications to seed the authentication passing the user credentials (email address and password) only once in a controlled way and obtain this authorization key than can then be stored locally with the application.

Request parameters

<table>
<thead>
<tr>
<th>authKey</th>
<th>String / Mandatory</th>
</tr>
</thead>
</table>
| This is the authorization key that you previously obtained with the `login` command.

Response parameters

<table>
<thead>
<tr>
<th>token</th>
<th>String</th>
</tr>
</thead>
</table>
| This is the global unique identifier for the session initiated with this Secure Messaging platform.

The session token is valid throughout the length of the session initiated with the server and must be used with all subsequent commands. Sessions expire after a given amount of time, at which time you need to obtain a new session token. Session tokens can be used only from the remote location from where it was obtained.

Error codes

1090001 Unexpected / Unknown login error.
   If you receive this error code please contact support immediately.

1090002 Unknown Secure Messaging platform.
   The session token was assigned to a different Secure Messaging platform than the specified URL.

1090003 Wrong request format.
   This is a valid JSON object, but the request has an invalid format. You may receive this message when you do not include mandatory fields or assigned an invalid value to one of the fields (e.g., wrong data type)
Examples

In the following example we assume the login command had been called previously and authorization key
("519ac58c138e4d44568a989a4b33f47d") was obtained and cached:

```plaintext
POST /services/api.aspx HTTP/1.1
User-Agent: email2Client
X-email2Cmd: authenticate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Host: vmqal:9999
Content-Length: 188
Expect: 100-continue
Connection: Close

{  
  "header": {  
    "command":"authenticate",  
    "timestamp":new Date(1230922366313),  
    "clientName":"apiNetTest",  
    "clientVersion":"0.1.0.0"  
  },  
  "data": {  
    "authKey":"519ac58c138e4d44568a989a4b33f47d"
  }
}
```

At the time this command was executed the authorization key was still valid and the following response was obtained from the server with a valid session token:

```plaintext
{  
  "header": {  
    "command":"authenticate",  
    "timestamp":new Date(1230922371548),  
    "errCode":0,  
    "errDesc":null,  
    "serverVersion":"1.6.2229"
  },  
  "data": {  
    "token":"fdd6adfd-b0a5-58fd-a91e-188b2be00e12"
  }
}
```
Forgot Password

The purpose of the `forgotPassword` command is to initiate the process of recovering access to a Secure Account on behalf of the user.

This command sends the corresponding user an email notification with a link to change their Password. The user must be registered with the Secure Messaging platform in order to work. Once the user clicks on the link, he or she is redirected to the Secure Messaging platform (Webmail) where they can choose a new Password.

This command does not require a session token (it can be executed by any user.)

**Request parameters**

<table>
<thead>
<tr>
<th>emailAddress</th>
<th>String / Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid user Email Address. The email address uniquely identifies the user in the Secure Messaging platform. The user must be registered with the Secure messaging platform in order for this command to complete the operation.</td>
<td></td>
</tr>
</tbody>
</table>

**Response parameters**

The `forgotPassword` command does not return a data section. You should check the header of the response to verify the action was completed successfully. See the Examples section for more details.

**Error codes**

1091001 Unexpected / Unknown forgotPassword error.
   If you receive this error code please contact support immediately.

1091002 Unknown Secure Messaging platform.
   The URL of the specified Secure Messaging platform didn’t match any authorized platform.

1091003 Wrong request format.
   This is a valid JSON object, but the request has an invalid format. You may receive this message when you do not include mandatory fields or assigned an invalid value to one of the fields (e.g., wrong data type)

1091004 The Email Address provided does not exist in the Secure Messaging platform.

1091005 The specified user is not registered with the Secure Messaging platform.

1091006 System configuration error when creating the forgotPassword email notification.
   If you receive this error code please contact support immediately.

1091007 System configuration error sending email notification.
   If you receive this error code please contact support immediately.

**Examples**

Because this command does not require authentication, there is no need to send the session token or the Asp.Net session id as with the login command.

The following example shows how to initiate the forget password workflow for a test vmqa1:9999 PEN. The request must be sent to the Secure Messaging platform API URL and must contain the X-email2Cmd X-Header parameter as shown below. Notice that the body of the request has been formatted for readability purposes only:
The line 3 in the previous example shows how you need to pass the X-email2Cmd X-Header. Line 4 specifies the Content Type required for the POST command.

If the user's Email Address exists and is registered, the server will reply with the following response:

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
Date: Wed, 14 Jan 2009 03:45:44 GMT
X-Powered-By: ASP.NET
Connection: close
X-AspNet-Version: 2.0.50727
Set-Cookie: ASP.NET_SessionId=yqdj2xfecyvk41bdo4shwcbj; path=/; HttpOnly
Cache-Control: private
Content-Type: text/html
Content-Length: 127

{
  "header": {
    "command":"forgotPassword",
    "timestamp":new Date(1231904744392),
    "errCode":0,
    "errDesc":"
    "serverVersion":"1.6.2261"
  }
}
```
### User / Member provisioning services

The following commands are available to provision new users in the corresponding Secure Messaging platform or to retrieve and / or update the user profile and their privileges.

Security: You need a session token to use these services. The information retrieved by these services depends on the privileges assigned to the user logged in.

To obtain a session token you first need to call the Login command. These following commands are available in this section:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Get Member</strong></td>
<td>Get the user’s profile. This command may be used by any active user of the Secure Messaging platform. If it does not have administrative privileges on the platform, the Get Member command can be used only to retrieve information about the same user.</td>
</tr>
<tr>
<td><strong>Add Member</strong></td>
<td>Add a user to the Secure Messaging platform. This command requires an administrative level user to use it. Depending on the requirements of the third party application, the new user can automatically be pre-registered by the administrator.</td>
</tr>
<tr>
<td><strong>Update Member</strong></td>
<td>Update the user’s information. This command requires an administrative level user to use it.</td>
</tr>
<tr>
<td><strong>Get Members List</strong></td>
<td>Retrieves the list of users with their basic status information. This command requires an administrative level user to use it.</td>
</tr>
</tbody>
</table>

#### Get Member

Command code: `getMember`

Use the `getMember` command to retrieve the information associated to a specified user identified by its Email Address.

The information returned by this command depends on the security level assigned to the user provided to obtain the session token.

Please refer to the Examples section below for the full syntax of the HTTP Request and Response.

**Request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>emailAddress</code></td>
<td>String / Mandatory</td>
<td>The Email Address that identifies the user you are requesting information for.</td>
</tr>
</tbody>
</table>

**Response parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>emailAddress</code></td>
<td>String</td>
<td>User’s Email Address. The Email Address uniquely identifies the user in the Secure Messaging platform.</td>
</tr>
<tr>
<td><code>status</code></td>
<td>String</td>
<td>Values: active, disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The current status of the user in this Secure Messaging platform. Inactive users cannot login to the Secure Messaging platform or use any of the services provided by the API.</td>
</tr>
<tr>
<td><strong>created</strong></td>
<td>Datetime (UTC)</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>Date and time the user was created in UTC / GMT format.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>package</strong></th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the User Group the user is currently assigned to.</td>
<td></td>
</tr>
<tr>
<td>The information provided for the User Group includes the identifying guid and flag to indicate if the assignment is permanent or temporary and for how long.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>.guid</strong></th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the global unique identifier for the User Group.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>.isTrial</strong></th>
<th>Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies if the User Group assignment is in trial mode or not.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>.expiry</strong></th>
<th>Datetime (UTC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the date and time (UTC) this User Group assignment will expire. This value is included only if this is a temporary assignment (isTrial is true).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>registration</strong></th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object specifying the registration status of the user.</td>
<td></td>
</tr>
<tr>
<td>If the user has not registered, this attribute is null (or not included in the response).</td>
<td></td>
</tr>
<tr>
<td>If it's included in the response, it contains two properties: status and timestamp. The status attribute has one of the following possible values and the timestamp attribute indicates when that status was changed:</td>
<td></td>
</tr>
<tr>
<td>- started: The user has started the registration process.</td>
<td></td>
</tr>
<tr>
<td>- pending: The user has submitted all the registration information, but is waiting to confirm their email address.</td>
<td></td>
</tr>
<tr>
<td>- completed: The user completed successfully the registration process.</td>
<td></td>
</tr>
<tr>
<td>- pendingExpired: The user submitted the registration information, but has not confirmed the email address. It is possible the user has not received the corresponding confirmation email message.</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**
```
"registration": {
   "status": "completed",
   "timestamp": new Date(1217290505123)
}
```

<table>
<thead>
<tr>
<th><strong>memberProfile</strong></th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object specifying the attributes that define the user’s profile, including <strong>firstName</strong> and <strong>lastName</strong>. Other attributes supported by the API are available to fully describe the new user in the system.</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**
```
"memberProfile" : {
   "firstName": "Joe",
   "lastName": "Doe"
}
```

<table>
<thead>
<tr>
<th><strong>.firstName</strong></th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>First name of the user</td>
<td></td>
</tr>
</tbody>
</table>
### extendedProfile

This list provides a flexible mechanism to extend the user's profile with attributes that may not correspond to specific Secure Messaging platform concepts.

You can use this list for your own purposes, like for storing identifying keys that will link this user to an external system.

Each item in the list consists of four values: **group**, **key**, **type**, and **value**. When you create a new extended value you must specify an identifying group and name. This combination must be unique and the system has some reserved groups that cannot be used by external systems via this API.

The value attribute depends on the type of field specified. You have three alternatives:

- **numeric**: Numeric (integer) value.
- **shortText**: Text value of 2,500 characters or less. Use this option when you want to store short descriptions or list items.
- **longText**: Free-form text field with a maximum length of 10,000 characters.

The content of this field is not encrypted in the database. You have the option to encrypt the content of this attribute using your own mechanism and just passing a BASE64 string value in this field with type longText. You also have the option of passing just one group-name combination with a value containing an XML/JSON representation of this extended profile attribute.

**Example:**

```
"extendedProfile" : 
[
   {"group":"payment", "key":"userID", "type":"shortText", "value":"1234579"},
   {"group":"payment", "key":"amount", "type":"numeric", "value":120},
   {"group":"payment", "key":"freq", "type":"shortText", "value":"monthly"}
]
```

### metricsUsage

List of metrics and usage tracked for this user.

Each metric usage has four values: **name**, **interval**, **limit**, and **usage**. The name field identifies the metric; interval specifies the frequency the metric is enforced and can be any of the following values: Daily (D), Monthly (M), and NoInterval (I); the limit defines how much of this metric is allowed to this user to use; and usage indicates how much is currently used.

The list of metrics, including their frequency and limit, is defined at the User Group the user belongs to. For many of the metrics available the administrator can change the frequency of the metric for a given User Group to accommodate specific needs. Also, the limit can be specified for individual users overriding the User Group definition.

The system keeps track of a series of metrics and more are constantly added to the list. Currently, the supported metrics are:

- **friendInvites**: How many invitations available (no frequency –absolute value).
- **maxMessages**: How many emails the user can send per interval of time, e.g., daily.
- **packagePromotions**: How many available User Group promotions or sponsorships.

**Example:**

```
"metricsUsage":
[
   {"name": "friendInvites", "limit": 10, "interval": "M", "usage": 5},
   {"name": "maxMessages", "limit": 50, "interval": "D", "usage": 37},
   {"name": "packagePromotions", "limit": 2, "interval": "I", "usage": 0}
]
```
Error codes

1105001  Unexpected / Unknown error getting the user’s profile.
If you receive this error code please contact support immediately.

1105002  Unknown Secure messaging platform.
The session token was assigned to a different Secure Messaging platform than the specified URL.

1105003  Wrong Request format.
This is a valid JSON object, but the request has an invalid format. You may receive this message when you do not include mandatory fields or assigned an invalid value to one of the fields (e.g., wrong data type)

1105004  System configuration error retrieving the user’s profile.
If you receive this error code please contact support immediately.

1105005  System configuration error: unknown User Group.
The User Group (package) assigned to the user does not exist. If you receive this error code please contact support immediately.

1105006  System configuration error retrieving user metric.
The metrics cannot be found for the specified user. If you receive this error code please contact support immediately.

1105007  Security error.
You do not have enough privileges to retrieve this user’s profile.
You are always allowed to retrieve the profile for the same user used to acquire the session token. If you want to retrieve the profile for a different user, you need to have administrative privileges.
Examples

The following request attempts to retrieve the user's information:

```json
01 {  02   "header": {  03     "command": "getMember",  04     "timestamp": new Date(1230922371877),  05     "clientName": "apiNetTest",  06     "clientVersion": "0.1.0.0"  07   },  08   "data": {  09     "emailAddress": "user.two@corp-national.com" 10   }
11 }
```

The response below shows the User Two user as an active user of the Secure Messaging platform:

```json
01 {  02   "header": {  03     "command": "getMember",  04     "timestamp": new Date(1230922372582),  05     "clientName": "apiNetTest",  06     "clientVersion": "0.1.0.0"  07   },  08   "data": {  09     "emailAddress": "user.two@corp-national.com", 10     "status": "active", 11     "created": new Date(1230665903250), 12     "package": { 13       "isTrial": true, 14       "expiry": new Date(1231188786100), 15       "guid": "C136EC82-D86F-4F6B-2B1A30035C28" 16     }, 17     "registration": { 18       "status": "completed", 19       "timestamp": new Date(1230666175490) 20     }, 21     "memberProfile": { 22       "firstName": "User", 23       "lastName": "Two" 24     }, 25     "metricsUsage": [ 26       {"usage":1, "key":"friendInvites", "limit":5, "interval":"D"}, 27       {"usage":23, "key":"maxMessages", "limit":50, "interval":"D"}, 28       {"usage":0, "key":"packagePromotions", "limit":10, "interval":"I"} 29     ] 30   }
31 }
```

In this example, the user is registered and has been temporarily assigned to a User Group. The command also includes the expiration date for this assignment. The user has 10 User Group (package) promotions / sponsorships available, can invite 4 more users, and can send another 27 secure messages for that day.
Add Member

Use the addMember command to provision a new user in the corresponding Secure Messaging platform.

Creating a new user requires appropriate privileges by the user used to obtain a session token.

The new user can be created pre-registered, in which case you provide the appropriate password. Pre-registered users still need to confirm they own the Email Address and will receive a notification message for that purpose.

Request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailAddress</td>
<td>String / Mandatory</td>
<td>User’s Email Address. The email address uniquely identifies the user.</td>
</tr>
<tr>
<td>status</td>
<td>String / Mandatory</td>
<td>Values: active, disable</td>
</tr>
<tr>
<td>preRegister</td>
<td>Boolean / Mandatory</td>
<td>Specify if the user will be pre-registered or not.</td>
</tr>
<tr>
<td>password</td>
<td>String / Optional</td>
<td>Password to login to the Secure Messaging platform.</td>
</tr>
<tr>
<td>skipRegConfirmation</td>
<td>Boolean / Optional</td>
<td>Skip the registration confirmation step. When this option is used, the user will be fully provisioned with their registration already confirmed.</td>
</tr>
</tbody>
</table>
When you create a new user you have the option to specify a specific User Group (package). If you don't specify a User Group the system will assign the default value configured in the Web Admin Console.

You do not create new User Groups with this command. The specified User Group must already exist before you can use it.

To assign a User Group you need to indicate three values: guid, isTrial, and expiry date as explained below. You identify the User Group by its guid and indicate if this is a permanent or temporary assignment and for how long.

Example:
```
"package":
{
  "guid": "KSJDLASKDJASD",
  "isTrial": false
}
```

### guid
- **Type:** String / Mandatory
- This is the global unique identifier for the User Group. You can obtain the packageGuid from the Web Admin Console.
- You may specify a specific User Group GUID to be assigned to the new user. If you don’t specify a User Group (package), the default User Group configured in the Web Admin Console for new users will be used instead.
- If you preRegister the user, this value is overridden by the corresponding value configured in the Web Admin Console.

### isTrial
- **Type:** Boolean / Mandatory
- Specifies if the User Group (package) assignment is in trial mode or not.
- When a User Group is assigned in trial mode, you must also specify the expiry date. When the expiration time is completed the system will automatically reassign the user to the User Group specified in the Web Admin Console.

### expiry
- **Type:** Datetime (UTC) / Optional
- Specifies the date and time (UTC) this User Group assignment will expire.
- This value is not considered if isTrial is set to false.

### memberProfile
- **Type:** Object / Mandatory
- Object specifying the attributes that define the user’s profile, including firstName and lastName. Other attributes supported are available to fully describe the new user in the system.

### firstName
- **Type:** String / Mandatory
- First name of the user.

### lastName
- **Type:** String / Mandatory
- Last name of the user.

### extendedProfile
- **Type:** List(Objec) / Optional
- This list provides a flexible mechanism to extend the user’s profile with attributes that may not correspond to specific Secure Messaging platform concepts.
- You can use this list for your own purposes, like for storing identifying keys that will link this user to an external system.
- Each item in the list consists of four values: group, key, type, and value. When you create a new extended value you must specify an identifying group and name. This combination must be unique and the system
has some reserved groups that cannot be used by external systems via this API. The value attribute depends on the type of field specified. You have three alternatives:

- **numeric**: Numeric (integer) value.
- **shortText**: Text value of 2,500 characters or less. Use this option when you want to store short descriptions or list items.
- **longText**: Free-form text field with a maximum length of 10,000 characters.

The content of this field is not encrypted in the database. You have the option to encrypt the content of this attribute using your own mechanism and just passing a BASE64 string value in this field with type longText. You also have the option of passing just one group-name combination with a value containing an XML/JSON representation of this extended profile attribute.

Example:

```json
"extendedProfile" : [  
  {"group":"payment", "key":"userID", "type":"shortText", "value":"1234579"},  
  {"group":"payment", "key":"amount", "type":"numeric", "value":120},  
  {"group":"payment", "key":"freq", "type":"shortText", "value":"monthly"}  
]
```

**Response parameters**

<table>
<thead>
<tr>
<th><strong>status</strong></th>
<th>String / Values: active, disable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the user is provisioned correctly, the system will return the current status of the user.</td>
</tr>
</tbody>
</table>

**Error codes**

- **1107001** Unexpected / Unknown error creating the user's profile.
  If you receive this error code please contact support immediately.

- **1107002** Unknown Secure Messaging platform.
  The session token was assigned to a different Secure Messaging platform than the specified URL.

- **1107003** Wrong Request format.
  This is a valid JSON object, but the request has an invalid format. You may receive this message when you do not include mandatory fields or assigned an invalid value to one of the fields (e.g., wrong data type)

- **1107004** User already exists.
  The specified Email Address already exists. Use the **updateMember** command to update the member’s information.

- **1107005** System configuration error creating the user record.
  If you receive this error code please contact support immediately.

- **1107006** System configuration error registering the user.
  You specified the user to be registered, but the server cannot complete the operation.
  If you receive this error code please contact support immediately.

- **1107008** System configuration error sending notification.
  The new user was created and registered successfully, but the welcome message failed to be delivered.
  Consider sending a new welcome message to this user.

- **1107009** Security error.
  You do not have enough privileges to add a new user.
Examples

This following request shows how you can create a new user. In this case we are attempting to add a new user without pre-registration and with the default User Group configured in the Web Admin Console:

```json
01 {
02   "header":{
03     "command":"addMember",
04     "timestamp":new Date(1230945974831),
05     "token":"3110be81-786e-4c61-bcea-18e935d5447c",
06     "clientId":"apiNetTest",
07     "clientVersion":"0.1.0.0"
08   },
09   "data":{
10     "emailAddress":"user.two@corp-national.com",
11     "status":"active",
12     "preRegister":false,
13     "memberProfile":{
14       "firstName":"QA",
15       "lastName":"Two"
16     }
17   }
18 }
```

This sample response shows the case when the member already exists:

```json
01 {
02   "header":{
03     "command":"addMember",
04     "timestamp":new Date(1230945974909),
05     "errCode":1107004,
06     "errDesc":"Member already exists!",
07     "serverVersion":"1.6.2229"
08   }
09 }
```

If it works, the user will receive this response:

```json
01 {
02   "header":{
03     "command":"addMember",
04     "timestamp":new Date(1230945974909),
05     "errCode":0,
06     "errDesc":"
07     "serverVersion":"1.6.2229"
08   },
09   "data":{
10     "status":"active"
11   }
12 }
```

In this case the user will receive the standard invitation message specified in the Web Admin Console.

The following example attempts to create a pre-registered user (where the password is provided):
In this case the user will receive a confirmation message. The user will have to accept the link to activate the Secure Account.
Update Member

Use the `updateMember` command to update a user’s profile. This command requires administrative privileges.

The user must already be registered before using this command. You must provide the user’s Email Address to identify the record you want to update. Most attributes are optional. You may specify only the specific attributes you want to change or all the user profile values available if needed.

Request parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>emailAddress</td>
<td>String</td>
<td>Mandatory</td>
<td>Email Address that identifies the user in the Secure Messaging platform. You cannot change the email address for a user. The email address specified in this field identifies the user you want to update its information.</td>
</tr>
<tr>
<td>status</td>
<td>String</td>
<td>Optional</td>
<td>Values: active, disable You may want to use the <code>disabled</code> value to temporarily disable the user.</td>
</tr>
</tbody>
</table>
| package | Object | Optional | Assign temporarily or permanently a new User Group to this user. Notice that you don’t create new User Groups with this command. The specified User Group must exist before you can use it. To assign a User Group you need to indicate three values: `guid`, `isTrial`, and `expiry` date as explained below. You identify the User Group (package) by its `guid` and indicate if this is a permanent or temporary assignment and for how long. **Example:**
```
"package":
{
  "guid": "KSJDLASKDJASD",
  "isTrial": false
}
```
| .guid | String | Mandatory | This is the global unique identifier for the User Group. You can obtain the `packageGuid` from the Web Admin Console. You may specify a specific Package GUID to be assigned to the new user. If you don’t specify a User Group (package), the default User Group configured in the Web Admin Console for new users will be used instead. If you `preRegister` the user, this value is overridden by the corresponding value configured in the Web Admin Console. |
| .isTrial | Boolean | Mandatory | Specifies if the User Group assignment is in trial mode or not. When a User Group is assigned in trial mode, you must also specify the `expiry` date. When the expiration time is completed the system will automatically reassign the user to the User Group specified in the Web Admin Console. |
| .expiry | Datetime (UTC) | Optional | Specifies the date and time (UTC) this User Group assignment will expire. |
This value is not considered if `isTrial` is set to false.

<table>
<thead>
<tr>
<th>memberProfile</th>
<th>Object / Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object specifying the attributes that define the user’s profile, including <code>firstName</code> and <code>lastName</code>. Other attributes supported are available to fully describe the new User.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.firstName</th>
<th>String / Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>First name of the User</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.lastName</th>
<th>String / Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last name of the User</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>metrics</th>
<th>List(Object) / Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update the metric limits for this User.</td>
<td></td>
</tr>
</tbody>
</table>

Each metric has three values: `name`, `interval`, and `limit`. The name field identifies the metric and is mandatory; interval specifies the frequency the metric is enforced and is an optional value that can be any of the following: Daily (D), Monthly (M), and NoInterval (I); and the limit defines how much of this metric is allowed to this user to use and is also an optional parameter.

The list of metrics, including their frequency and limit, is defined at the User Group the user belongs to. For many of the metrics available the administrator can change the frequency of the metric for a given user to accommodate specific needs. Also, the limit can be specified for individual Users overriding the User Group definition.

The system keeps track of a series of metrics and more are constantly added to the list. Currently, the supported metrics are:

- **friendInvites**: How many invitations available (no frequency – absolute value).
- **maxMessages**: How many emails the user can send per interval of time, e.g., daily.
- **packagePromotions**: How many available User Group promotions or sponsorships.

**Example**:
```
"metrics":
[
  {"name": "friendInvites", "limit": 10, "interval": "M", "usage": 5},
  {"name": "maxMessages", "limit": 50, "interval": "D", "usage": 37}
]
```

<table>
<thead>
<tr>
<th>extendedProfile</th>
<th>List(Object) / Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>This list provides a flexible mechanism to extend the User’s profile with attributes that may not correspond to specific Secure Messaging platform concepts.</td>
<td></td>
</tr>
</tbody>
</table>

You can use this list for your own purposes, like for storing identifying keys that will link this user to an external system.

Each item in the list consists of four values: `group`, `key`, `type`, and `value`. When you create a new extended value you must specify an identifying group and name. This combination must be unique and the system has some reserved groups that cannot be used by external systems via this API.

The value attribute depends on the type of field specified. You have three alternatives:

- `numeric`: Numeric (integer) value.
- `shortText`: Text value of 2,500 characters or less. Use this option when you want to store short descriptions or list items.
- `longText`: Free-form text field with a maximum length of 10,000 characters.

The content of this field is not encrypted in the database. You have the option to encrypt the content of this attribute using your own mechanism and just passing a BASE64 string value in this field with type `longText`. You also have the option of passing just one group-name combination with a value containing an XML/JSON representation of this extended profile attribute.

**Example**:
"extendedProfile":
[
  {"group":"payment", "key":"userID", "type":"shortText", "value":"1234579"},
  {"group":"payment", "key":"amount", "type":"numeric", "value":120},
  {"group":"payment", "key":"freq", "type":"shortText", "value":"monthly"}
]

Response parameters

<table>
<thead>
<tr>
<th>status</th>
<th>String / Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Values: active, disable</td>
</tr>
</tbody>
</table>

If the user is provisioned correctly, the system will return the current status of the User.

Error codes

1106001 Unhandled / Unknown error updating the User's profile.
If you receive this error code please contact support immediately.

1106002 Unknown Secure Messaging platform.
The session token was assigned to a different Secure Messaging platform than the specified URL.

1106003 Wrong Request format.
This is a valid JSON object, but the request has an invalid format. You may receive this message when you do not include mandatory fields or assigned an invalid value to one of the fields (e.g., wrong data type)

1106004 Security error.
You do not have enough privileges to update this user’s profile.

1106005 System configuration error retrieving the user’s profile.
If you receive this error code please contact support immediately.

1106006 System configuration error retrieving the User Group (package).
If you receive this error code please contact support immediately.

1106007 System configuration error updating the User Group (package).
If you receive this error code please contact support immediately.

1106008 Metric does not exist.
The specified User’s metric does not exist.

Examples

The following request shows how a user can be updated to change the last name and assign a new metric to the user’s profile (in this case, to change the amount of package promotions available to the User). This example also shows how to use the extended profile to save billing information. This 'billing' information has no special meaning in the system, however, it can be used by third party applications for their own purposes.

In this example, the credit card information (number) is sent encrypted (the Secure Messaging platform will not know how to decrypt this information):
If the action is allowed, the system will send this response:

```
01 {  
02   "header":{  
03     "command": "updateMember",  
04     "timestamp": new Date(1230945974909),  
05     "errCode": "0",  
06     "errDesc": "",  
07     "serverVersion": "1.6.2229"  
08   },  
09   "data":{  
10     "status": "active"  
11   }  
12 }
```
Secure messaging services

All these commands require a session token and the information exchanged must belong to the authenticated user.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Mailbox*</td>
<td>Get the full list of secure messages in a user’s inbox. Requires a session token.</td>
</tr>
<tr>
<td>Get Message Info*</td>
<td>Get all secure message information related to a selected secure message. Requires a session token.</td>
</tr>
<tr>
<td>Pre Create Message</td>
<td>Creates a record in the database for a secure message and provides a global identifier (GUID) for the new secure message to associate content to it. Must happen before a Save Message or Send Message commands. Requires a session token.</td>
</tr>
<tr>
<td>Save Message</td>
<td>Save secure message’s content to the provided message identifier. Requires a valid and active message GUID. Requires a session token.</td>
</tr>
<tr>
<td>Send Message</td>
<td>Completes the transaction for a secure message checking all business rules and privileges of the participants in the conversation and sends the notification to the intended parties if it applies. Requires a valid and active message GUID. Requires a session token.</td>
</tr>
<tr>
<td>Add Attachments</td>
<td></td>
</tr>
<tr>
<td>Send Bulk Message*</td>
<td>Used to send bulk secure messages to multiple recipients in one operation. The bulk-mailer command supports a flexible template language, which can be used to send personalized secure messages to multiple users in one action. Optionally, this command may include a pre-parsed content for each user, allowing further customization of the secure messages. Requires a session token.</td>
</tr>
</tbody>
</table>

Pre Create Message

Command code: preCreateMessage

Use the preCreateMessage command to create a new secure message.

If successful, the server will return the message global unique identifier (GUID) for the new secure message along with the list of default message options associated to this message according to the privileges and attributes of the sender’s profile.

The information returned by this command depends on the security level assigned to the user provided to obtain the session token.

Please refer to the Examples section below for the full syntax of the HTTP Request and Response.

Request parameters

<table>
<thead>
<tr>
<th>action</th>
<th>String / Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values: newMessage, reply, replyAll, forward</td>
<td></td>
</tr>
</tbody>
</table>

This attribute indicates the type of secure message to create.

The newMessage value creates a new message starting a new conversation, whereas the reply, replyAll, and forward values indicate that this new secure message is part of an existing conversation, in which case you also need to provide the parentMsgGuid parameter.

Creating a newMessage requires that the User Group assigned to the sender has the appropriate privileges for this action. An appropriate error code is provided if the sender does not have the appropriate privileges.

Creating a newMessage requires that the User Group assigned to the sender has the appropriate privileges for this action. An appropriate error code is provided if the sender does not have the appropriate privileges.

Creating a newMessage requires that the User Group assigned to the sender has the appropriate privileges for this action. An appropriate error code is provided if the sender does not have the appropriate privileges.

Creating a newMessage requires that the User Group assigned to the sender has the appropriate privileges for this action. An appropriate error code is provided if the sender does not have the appropriate privileges.

Creating a newMessage requires that the User Group assigned to the sender has the appropriate privileges for this action. An appropriate error code is provided if the sender does not have the appropriate privileges.

Creating a newMessage requires that the User Group assigned to the sender has the appropriate privileges for this action. An appropriate error code is provided if the sender does not have the appropriate privileges.

Creating a newMessage requires that the User Group assigned to the sender has the appropriate privileges for this action. An appropriate error code is provided if the sender does not have the appropriate privileges.

Creating a newMessage requires that the User Group assigned to the sender has the appropriate privileges for this action. An appropriate error code is provided if the sender does not have the appropriate privileges.

Creating a newMessage requires that the User Group assigned to the sender has the appropriate privileges for this action. An appropriate error code is provided if the sender does not have the appropriate privileges.
### parentMsgGuid

<table>
<thead>
<tr>
<th>String / Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the Global Unique Identifier of the message you are trying to link the message with. This parameter is mandatory for the <strong>action</strong> parameter values <strong>reply</strong>, <strong>replyAll</strong>, and <strong>forward</strong>.</td>
</tr>
</tbody>
</table>

### Response parameters

<table>
<thead>
<tr>
<th>msgGuid</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the Global Unique Identifier (GUID) for the new message. When you start creating a new secure message by calling preCreate command you will receive this <strong>msgGuid</strong> identifier. You should store this value for the length of the transaction as it is used by all the remaining messaging commands that complete the information for the new message.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>parentTo</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <strong>parentTo</strong> parameter is a comma-separated list of email addresses and it is intended to be used in the TO field of the message’s recipients list. This value is normally empty, but in some occasions and depending on the permissions of the user or their package attributes, the server will enforce that the user sends this message to the specified list of email addresses. In this case, an attempt to send the secure message to a different list of email addresses may result in an error when invoking the <strong>saveMessage</strong> command. An example of this is the scenario when the User Group forces users to communicate exclusively with a pre-determined list of users. At this point this <strong>parentTo</strong> parameter is provided only as a reference for client applications to alert the end user in the corresponding user interface.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>parentCc</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <strong>parentCc</strong> parameter is a comma-separated list of email addresses and it is intended to be used in the TO field of the message’s recipients list. The same rules described for <strong>parentTo</strong> parameter apply to the <strong>parentCc</strong> parameter.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>penDefOptions</th>
<th>List of Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEPRECATED</strong></td>
<td>This field is being deprecated. Please refrain from using this field as a future release of the server will not include this field anymore.</td>
</tr>
<tr>
<td>This is the list of default message options supported by the Secure Messaging Platform. These parameters are provided so that client applications can inform end users in the corresponding user interface about these message options and to be used as default values in the <strong>sendMessage</strong> command if the user doesn’t specify values for these options. These default values can be set by the administrator for the entire Secure Messaging Platform. Each item in the list of default message options has a value and blocked attributes:</td>
<td></td>
</tr>
<tr>
<td>o <strong>value</strong>. The value attribute is a dynamic type parameter that indicates the default value to be used for the corresponding message option. For example, a boolean value for <strong>allowTracking</strong> indicates that the option can be enabled or not (true/false).</td>
<td></td>
</tr>
<tr>
<td>o <strong>blocked</strong>. The blocked attribute is a hint to the client application to configure the user interface accordingly to allow or disallow the end user changing the value of the specific message option. For example, the following message option specification for <strong>allowTracking</strong> indicates that when the <strong>saveMessage</strong> command is invoked the message will be created with the do-not-reply attribute:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;allowTracking&quot;:</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>&quot;value&quot;: false,</td>
<td></td>
</tr>
<tr>
<td>&quot;blocked&quot;: true</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>The blocked attribute is only a hint for the user interface. Client applications may still decide to ignore it and allow the end user to change the value even when it is indicated as blocked. Doing this may result on an</td>
<td></td>
</tr>
</tbody>
</table>
appropriate error when invoking the `sendMessage` command.

Below it’s the list of options currently supported:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.allowReply</td>
<td>Specifies if this secure message accepts reply actions.</td>
</tr>
<tr>
<td>.allowForward</td>
<td>Specifies if this secure message accepts forward action.</td>
</tr>
<tr>
<td>.allowTracking</td>
<td>Specifies if this secure message will have tracking available for its recipients.</td>
</tr>
<tr>
<td>.shareTracking</td>
<td>Specifies if the sender of this secure message will be able to share tracking information with recipients.</td>
</tr>
</tbody>
</table>

Error codes

1205001 Unexpected / Unknown error creating the secure message.
If you receive this error code please contact support immediately.

1205002 Unknown Secure Messaging platform / Unknown Session.
The session token was assigned to a different Secure Messaging platform than the specified URL.

1205003 Wrong Request format.
This is a valid JSON object, but the request has an invalid format. You may receive this message when you do not include mandatory fields or assigned an invalid value to one of the fields (e.g., wrong data type)

1205010 Invalid secure message.
An error occurred while validating the secure message options or user credentials over this message.
You may not have enough privileges to complete this action or one of the specific rules governing User Groups may be preventing this action.

1205036 Message set For Your Eyes Only with Secure Account Password.
The parent message you are replying/forwarding has been marked with For Your Eyes Only with Secure Account Password and the current session has not unblocked access to the original message yet.

1205037 Message set For Your Eyes Only with a Unique Password.
The parent message you are replying/forwarding has been marked with For Your Eyes Only with a Unique Password and the current session has not unblocked access to the original message yet.

1205907 This user does not have access to this action.
The User Group assigned to the user does not allow sending new secure messages or reply/forward action for an existing message.

1205909 Cannot forward.
The message you are trying to forward does not allow this action.
You may receive this error when replying to a message and adding new recipients to the conversation. In this case the reply action is considered a forward action.

1205912 Message terminated.
The parent message you are replying/forwarding has been terminated and access to it cannot be granted to continue the conversation.

1205915 Message recalled.
The parent message you are replying/forwarding has been recalled and access to it cannot be granted to continue the conversation.

Examples
The following request creates a secure message for the user currently logged in (determined by the session token):

```json
01 { 
02   "header": { 
03     "command": "preCreateMessage", 
04     "token": "9c885fa6-9b57-4c1f-89909", 
05     "timestamp": new Date(1230922371877), 
06     "clientName": "apiNetTest", 
07     "clientVersion": "0.1.0.0"
08   },
09   "data": { 
10     "action": "newMessage"
11   }
12 }
```

The response below allows the action and includes the message GUID along with the list of default message options supported:

```json
01 { 
02   "header": { 
03     "command": "preCreateMessage", 
04     "timestamp": new Date(1297725440435), 
05     "errCode": 0, 
06     "errDesc": "", 
07     "serverVersion": "2.0.4058"
08   },
09   "data": { 
10     "msgGuid": "23C9A222-1BE2-4670-AF1E-15A38027059D", 
11     "parentTo": "", 
12     "parentCc": "", 
13     "penDefOptions": { 
14       "allowReply": { 
15         "value": true, 
16         "blocked": false
17       }, 
18       "allowForward": { 
19         "value": true, 
20         "blocked": false
21       }, 
22       "allowTracking": { 
23         "value": true, 
24         "blocked": false
25       }, 
26       "shareTracking": { 
27         "value": true, 
28         "blocked": false
29       }, 
30       "freezeDate": { 
31         "value": new Date(-62135596800000), 
32         "blocked": false
33       }
34     }
35   }
36 }
```

In this example the message is created successfully and its GUID is returned. The message does not have an enforced list of recipients and none of the message options are blocked, meaning that the client application may decide to offer the end user the possibility of setting these values when completing the secure message.
Save Message

Command code: `saveMessage`

Use the `saveMessage` command to update the content and metadata of a secure message before it is sent (completed).

Some of the message information that you can change with this command includes: list of recipients, subject line, message body, message options, unique password, and more. All content / configuration parameters are optional, so you may decide to change them individually or all at once depending on the specific nature of your client application.

When the call is successful, the server will return the current status of the message, which should be Draft status in most cases, and the current list of message options assigned to this message. The server also returns other indicators to be used in optional actions like adding new file attachments to the message and a flag to confirm which side will be in charge of notifying the recipients of the new secure message.

Please refer to the Examples section below for the full syntax of the HTTP Request and Response.

**Request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>msgGuid</strong></td>
<td>String / Mandatory</td>
<td>This is the Global Unique Identifier (GUID) for the message being updated. When you start creating a new secure message by calling <code>preCreateMessage</code> command you will receive this msgGuid identifier. You should store this value for the length of the transaction as it is used by all the remaining messaging commands that complete the information for the new message.</td>
</tr>
<tr>
<td><strong>recipients</strong></td>
<td>Object / Optional</td>
<td>This object represents the list of recipients in the secure message. It may be comprised of any of the TO:, CC:, and BCC: attributes (all optional). Each one of these corresponds to a comma-separated list of email addresses.</td>
</tr>
<tr>
<td>.to</td>
<td>String / Optional</td>
<td>This is a comma-separated list of email addresses to be added to the TO: field in the secure message.</td>
</tr>
<tr>
<td>.cc</td>
<td>String / Optional</td>
<td>This is a comma-separated list of email addresses to be added to the CC: field in the secure message.</td>
</tr>
<tr>
<td>.bcc</td>
<td>String / Optional</td>
<td>This is a comma-separated list of email addresses to be added to the BCC: field in the secure message.</td>
</tr>
<tr>
<td><strong>subject</strong></td>
<td>String / Optional</td>
<td>This is the subject line of the secure message.</td>
</tr>
<tr>
<td><strong>body</strong></td>
<td>String / Optional</td>
<td>This is the content of the secure message. The secure message’s content may be in HTML or TEXT format according to the format parameter. The content of the <code>body</code> parameter must be stored encoded in BASE64 format.</td>
</tr>
<tr>
<td><strong>format</strong></td>
<td>String / Optional</td>
<td>This is the format of the <code>body</code> of the secure message. Values: HTML, TEXT</td>
</tr>
</tbody>
</table>
### msgOptions

This is the list of message options currently attached to this secure message.

Each item in the list has a dynamic type value. For example, a boolean value for `allowTracking` indicates that the option can be enabled or not (true/false).

Below is the current list of supported message options. Some of these options may not be enabled and as such setting a value for them may not have an impact on the corresponding message.

<table>
<thead>
<tr>
<th>Option</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowReply</td>
<td>Boolean</td>
</tr>
<tr>
<td>allowForward</td>
<td>Boolean</td>
</tr>
<tr>
<td>allowTracking</td>
<td>Boolean</td>
</tr>
<tr>
<td>shareTracking</td>
<td>Boolean</td>
</tr>
<tr>
<td>forYourEyesOnlyType</td>
<td>String</td>
</tr>
<tr>
<td>fyeoType</td>
<td>String</td>
</tr>
</tbody>
</table>

#### .allowReply
 Specifies if this secure message accepts reply actions.

#### .allowForward
 Specifies if this secure message accepts forward action.

#### .allowTracking
 Specifies if this secure message will have tracking available for its recipients.

#### .shareTracking
 Specifies if the sender of this secure message will be able to share tracking information with recipients.

#### .forYourEyesOnlyType
 The `forYourEyesOnlyType` field indicates if the message was marked with the “For Your Eyes Only” feature. The FYEO feature provides an extra layer of security by allowing setting a password for each individual message that needs to be entered at the time of accessing the message.

It can be one of the following values:

- **disabled**. The current message does not use For Your Eyes Only.
- **uniquePass**. The message has a unique password assigned to it that needs to be provided at the time the message is retrieved.
- **penPass**. In this case, the message does not have a unique password assigned to it, but the retriever needs to provide their own Service credentials in order to access the content of the message. This is an intermediate solution in which a message password doesn’t need to be exchanged between the sender and recipients, but it asks for confirmation of each recipients’ credentials before accessing the message.

#### .fyeoType
 Values: disabled, penPass, uniquePass

The `fyeoType` field indicates if the message was marked with the “For Your Eyes Only” feature. The FYEO feature provides an extra layer of security by allowing setting a password for each individual message that needs to be entered at the time of accessing the message.

It can be one of the following values:

- **disabled**. The current message does not use For Your Eyes Only.
- **uniquePass**. The message has a unique password assigned to it that needs to be provided at the time the message is retrieved.
- **penPass**. In this case, the message does not have a unique password assigned to it, but the retriever needs to provide their own Service credentials in order to access the content of the message. This is an intermediate solution in which a message password doesn’t need to be exchanged between the sender and recipients, but it asks for confirmation of each recipients’ credentials before accessing the message.

#### craCode
 The `craCode` parameter corresponds to the Challenge Response Authentication (CRA) feature that can be assigned to this message.

When a user invites a new user and the CRA option is enabled in the Secure Messaging platform, they must assign a code or password that the new user must enter when registering. This can be set to the new user’s customer ID or PIN number assigned by your organization.

#### validateDraft
 When true (default is false) this parameter indicates to the Server that it should enforce the validation rules to the draft message.

The validation rules are normally applied only at the `sendMessage` command (they are always applied at this point). This flag is available so that client applications can provide early feedback to end users about the validity of the secure message without having to send it yet.

### Response parameters
<table>
<thead>
<tr>
<th>date</th>
<th>Datetime (UTC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date and time the message was created in UTC / GMT format.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>attachChunkSize</th>
<th>Numeric (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>This value indicates the maximum file size (in Bytes) to be accepted by the Server when uploading attachments. If an attachment goes over this limit, the client application must split the file in chunks of this size.</td>
<td></td>
</tr>
<tr>
<td>Uploading attachments in chunks of a pre-defined size allows client applications to better manage their resources. They could choose to implement a library where multiple chunks are uploaded in parallel or sequentially depending on their unique requirements.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>msgOptions</th>
<th>List of values / Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the list of message options currently attached to this secure message.</td>
<td></td>
</tr>
<tr>
<td>Each item in the list has a dynamic type value. For example, a boolean value for allowTracking indicates that the option can be enabled or not (true/false).</td>
<td></td>
</tr>
<tr>
<td>Below is the current list of supported message options. Some of these options may not be enabled and as such setting a value for them may not have an impact on the corresponding message.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.allowReply</th>
<th>Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies if this secure message accepts reply actions.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.allowForward</th>
<th>Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies if this secure message accepts forward action.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.allowTracking</th>
<th>Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies if this secure message will have tracking available for its recipients.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.shareTracking</th>
<th>Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies if the sender of this secure message will be able to share tracking information with recipients.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.forYourEyesOnlyType</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEPRECATED</strong> This field is being deprecated. Please refrain from using this field as a future release of the server will not include this field anymore. Please use fyeoType instead.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.fyeoType</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values: disabled, penPass, uniquePass</td>
<td></td>
</tr>
<tr>
<td>The fyeoType field indicates if the message was marked with the “For Your Eyes Only” feature. The FYEO feature provides an extra layer of security by allowing setting a password for each individual message that needs to be entered at the time of accessing the message.</td>
<td></td>
</tr>
<tr>
<td>It can be one of the following values:</td>
<td></td>
</tr>
<tr>
<td>o <strong>disabled</strong>. The current message does not use For Your Eyes Only.</td>
<td></td>
</tr>
<tr>
<td>o <strong>uniquePass</strong>. The message has a unique password assigned to it that needs to be provided at the time the message is retrieved.</td>
<td></td>
</tr>
<tr>
<td>o <strong>penPass</strong>. In this case, the message does not have a unique password assigned to it, but the retriever needs to provide their own Service credentials in order to access the content of the message. This is an intermediate solution in which a message password doesn’t need to be exchanged between the sender and recipients, but it asks for confirmation of each recipients’ credentials before accessing the message.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>e2K</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the uncompressed version of the Secure Message Signature (e2K) included in all email notifications.</td>
<td></td>
</tr>
<tr>
<td>A compressed version of this key must be used by the client application if it needs to send the email notifications. When the server delivers the notifications, the client application will receive the compressed version of the e2K along with the entire content of the notification according to the templates in the Web Admin Console.</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>.host</td>
<td>String</td>
</tr>
<tr>
<td>.penGuid</td>
<td>String</td>
</tr>
<tr>
<td>.mailGUID</td>
<td>String</td>
</tr>
<tr>
<td>.parentMailGuid</td>
<td>String</td>
</tr>
<tr>
<td>useEmail2SmtpServer</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

**Error codes**

1204001 Unexpected / Unknown error creating the secure message.
   If you receive this error code please contact support immediately.

1204002 Unknown Secure Messaging platform / Unknown Session.
   The session token was assigned to a different Secure Messaging platform than the specified URL or the session has expired.

1204003 Wrong Request format.
   This is a valid JSON object, but the request has an invalid format. You may receive this message when you do not include mandatory fields or assigned an invalid value to one of the fields (e.g., wrong data type)

1204016 The message cannot be sent because the following user(s) have blocked you - <list of users>
   You may get this error code only when using the validateDraft parameter.

1204017 This message cannot be sent because it contains the following blacklisted keywords - <list of keywords>
   You may get this error code only when using the validateDraft parameter.

1204026 You don't have privileges to invite new users to the Secure Messaging platform.
   You are inviting new users with this secure message and your User Group does not allow you to do that or you do not have enough invites available.
   You may get this error code only when using the validateDraft parameter.

1204044 CRA code required to invite new user.
   You may get this error code only when using the validateDraft parameter.

1204801 You cannot save draft on an invitation messages.
   Invitation messages do not allow saving drafts.

1204906 The following recipient(s) are not allowed to receive secure messages - <list of users>
   You may get this error code only when using the validateDraft parameter.

1204907 This user does not have access to this action.
   The User Group assigned to the user does not allow sending new secure messages or reply / forward action for an existing secure message.
   You may get this error code only when using the validateDraft parameter.

1204909 Cannot forward.
The message you are trying to forward does not allow this action.

You may receive this error when replying to a message and adding new recipients to the conversation. In this case the reply action is considered a forward action.

You may get this error code only when using the validateDraft parameter.

1204912  Message terminated.

The parent message you are replying / forwarding has been terminated and access to it cannot be granted to continue the conversation.

You may get this error code only when using the validateDraft parameter.

1204915  Message recalled.

The parent message you are replying / forwarding has been recalled and access to it cannot be granted to continue the conversation.

You may get this error code only when using the validateDraft parameter.

Examples

The following request attempts to update a secure message passing user.two@corp-national.com in the TO: field of the recipients' list. The subject line and the body of the message are also included in the request. The request also shows an example of how to assign non-default values to the message options. The validateDraft parameter is not set to avoid validations (this is an operation intended for saving a draft of the message):

```javascript
{   "header": {       "command": "saveMessage",       "token": "9c885fa6-9b57-4clf-89909",       "timestamp": new Date(1230922389877),       "clientName": "apiNetTest",       "clientVersion": "0.1.0.0"   },
   "data": {       "msgGuid": "23C9A222-1BE2-4670-AF1E-15A38027059D",       "recipients": {           "to": [               "user.two@corp-national.com"           ]       },
       "body": "&lt;html&gt;&lt;body&gt;This is the HTML content of the message&lt;/body&gt;&lt;/html&gt;",       "subject": "Testing simple secure message",       "format": "HTML",       "msgOptions": {           "allowReply": true,
           "allowForward": true,
           "allowTracking": true,
           "shareTracking": false,
        },
       "validateDraft": false   }
```

The response below allows the action and includes the current status of the secure message along with the list of message options currently assigned to this message. The e2k signature components list is also included in the response:
01 {  
02  "header": {  
03      "command": "saveMessage",  
04      "timestamp": new Date(1297725440435),  
05      "errCode": 0,  
06      "errDesc": "",  
07      "serverVersion": "2.0.4058"  
08  },  
09  "data":{  
10      "status": "DR",  
11      "date": new Date(1297725439467),  
12      "e2K": {  
13          "host": "https://secure-messaging.com/code/services/services.aspx",  
14          "penGuid": "M909D15-XXXX-4C52-B00F-90009",  
15          "mailGuid": "ABCCCCC-XXXX-8IG3-B5C4-660E4A0B5014",  
16          "parentMailGuid": 0  
17      },  
18      "attachChunkSize": 4096,  
19      "msgOptions": {  
20          "allowReply": true,  
21          "allowForward": true,  
22          "allowTracking": true,  
23          "shareTracking": false,  
24          "freezeDate": new Date(-62135596800000)  
25      }  
26  }  
27 }
Add Attachments

Use the `addAttachments` command to assign a new file attachment to the secure message.

This command is not intended to upload the actual files. Instead, client applications are expected to use this command to declare their intention of uploading a file at a future time. The record for the file attachment is created in the system associated to the given secure message and it is marked with status uploading.

If there are no errors, the Server will return the list of file attachments parts the client application will need to upload at a future time. This list includes the specific URLs for where each attachment part needs to be uploaded.

Depending on the Web Admin Console settings each file attachment may need to be divided into smaller parts and potentially uploaded to different File Servers. This architecture allows the system to handle large amounts of file transfers and offers greater flexibility when scaling the system. For client applications working with the API, it offers the possibility of uploading multiple file attachment parts in parallel improving the efficiency and reliability of these applications.

Request parameters

<table>
<thead>
<tr>
<th><code>msgGuid</code></th>
<th><code>String / Mandatory</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the Global Unique Identifier (GUID) of the secure message you are adding file attachments to. When you start creating a new secure message by calling <code>preCreateMessage</code> command you will receive this <code>msgGuid</code> identifier. You should store this value for the length of the transaction as it is used by all the remaining messaging commands that complete the information for the new message.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>attachments</code></th>
<th><code>List(Object) / Mandatory</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use this parameter to describe the file attachments to be associated with the secure message. At least one attachment must be included in the list. For each attachment, the minimum information to be included is the file name and size in bytes. Other optional and extensible parameters may be included with the request. The information for each attachment must be described passing an instance of the <code>info</code> attribute. The structure of this <code>info</code> parameter is described below:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>.fileName</code></th>
<th><code>String / Mandatory</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the actual file name in the file system of the file attachment file you intend to upload. This <code>fileName</code> doesn't need to be unique in the system. Each attachment file has a Global Unique Identifier (GUID) used to distinguish each file, so you can upload multiple copies of the same file without overriding previous copies.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>.size</code></th>
<th><code>Numeric / Mandatory</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the size of the file in bytes.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>.extensions</code></th>
<th><code>Object / Optional</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>The <code>extensions</code> parameter is a free-form object stored securely ‘as-is’. It is intended to be used by client applications for their own specific purpose. It allows storing specific LOB (line of business) information to a file attachment that is relevant only to the client application making use of it. For example, it may be used to add an external or third party system identifier for this file in the system to establish a relationship with the external system. The content of this object may even be encrypted with a proprietary mechanism known only to the client application provider to further increase the confidentiality of sensitive information. The system will securely store this information along with all the meta-data associated to the secure message. Given the inherent flexibility of this mechanism your application must be ready to deal with a dynamic list of attributes in this extensions <code>parameter</code>. This list of attributes may even be different for each attachment.</td>
<td></td>
</tr>
</tbody>
</table>
Response parameters

<table>
<thead>
<tr>
<th>chunkSize</th>
<th>Numeric (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>This value indicates the maximum file size (in Bytes) to be accepted by the server when uploading file attachments. If an attachment goes over this limit, the client application must split the file in chunks of this <code>chunkSize</code> size. Uploading attachments in chunks of a pre-defined size allows client applications to better manage their resources. They could choose to implement a library where multiple chunks are uploaded in parallel or sequentially depending on their unique requirements. This value will have a value of zero '0' when no changes have been made to the <code>chunkSize</code> already reported in the <code>saveMessage</code> command.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>attachmentParts</th>
<th>List(List(Object))</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the list of file attachment parts that need to be uploaded to the system. The first level of the list corresponds to the attachment itself (as whole). Each one of these is further divided into their specific parts according to the <code>chunkSize</code> parameter. For each part the sequence number is returned in the attribute <code>sequenceNum</code> along with the <code>uri</code> where the file should be uploaded. If you receive more than one entry in the list of parts for one specific attachment your client application must split the file according to the <code>chunkSize</code> parameter and upload them with the corresponding sequence number to the appropriate location.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.sequenceNum</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the order of this part or chunk in the whole file. This is a 0-based index of the attachment parts.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.uri</th>
<th>Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the full URL of where the file needs to be uploaded using an HTTP POST action. The parameters in the URL grant sufficient permissions to the client application to upload the file directly to the specified File Server. Please notice that this URL may be different for each part.</td>
<td></td>
</tr>
</tbody>
</table>

Error codes

1207001 Unexpected / Unknown error creating the secure message.
If you receive this error code please contact support immediately.

1207002 Unknown Secure Messaging platform / Unknown Session.
The session token was assigned to a different Secure Messaging platform than the specified URL.

1207003 Wrong Request format.
This is a valid JSON object, but the request has an invalid format. You may receive this message when you do not include mandatory fields or assigned an invalid value to one of the fields (e.g., wrong data type)

1207005 The specified msgGUID is not valid for the Secure Messaging platform.

1207000006 General error associating the provided file attachment(9s) to the specified secure message.
If you receive this error, please contact support to provide more information.
Examples

The following request adds a new file attachment to an existing secure message for the user currently logged in (determined by the session token):

```json
01 { 02   "header": { 03     "command": "addAttachments", 04     "token": "9c885fa6-9b57-4c1f-89909", 05     "timestamp": new Date(1230922371877), 06     "clientName": "apiNetTest", 07     "clientVersion": "0.1.0.0" 08   }, 09   "data": { 10     "msgGuid": "23C9A222-1BE2-4670-AF1E-15A38027059D", 11     "attachments": [ 12       { 13         "info": { 14           "fileName": "longSampleFile.exe", 15           "size": 622736, 16           "extensions": { 17             "str-displayName": "longSampleFile.exe" 18           }, 19           "type": "normal" 20         } 21       } 22     ] 23   } 24 }
```

The response below allows the action including the list of file attachment parts the client application will need to upload:

```json
01 { 02   "header": { 03     "command": "addAttachments", 04     "timestamp": new Date(1297725518560), 05     "errCode": 0, 06     "errDesc": ", 07     "serverVersion": "2.0.4058" 08   }, 09   "data": { 10     "chunkSize": 0, 11     "attachmentParts": [ 12       "d9d6713e-8e61-4efa-bc86-1c039d4edf2e": [ 13         { 14           "sequenceNum": 0, 15           "url": "https://secure-messaging.com/code/services/attachments.aspx?pt=06B45CBF-A931-4689-8B1B-8D2AAABA86AFE" 16         } 17       ] 18     ] 19   } 20 }
```

In this example the secure message is created successfully and its GUID is returned. The message does not have an enforced list of recipients and none of the message options are blocked, meaning that the client application may decide to offer the end user the possibility of setting these values when completing the secure message.
Send Message

Use the `sendMessage` command to complete a secure message and initiate the notification workflow for the intended recipients.

After this command is executed no further modifications can be submitted to the secure message. You can, however, change its current status to further block access to it if required; but you cannot change identifying metadata for this message anymore.

The system executes a last check for the validity of the operation. For example, while the message was updated with one or more calls to commands like `saveMessage` or `addAttachments`, the client application may have registered a specific list of recipients with this message. However, none of these recipients have actually been added to the system yet as it is in Draft state. Only at the time the message is sent are all the validations executed. In particular, the system checks for new users in the list of recipients and alerts the client application with the appropriate error codes. In this way, the system executes multiple types of validations on the operation to make sure that the rules defined in the Web Admin Console and the privileges assigned to the User Group of the user sending the message (identified implicitly by the session token used to initiate the command) lie within the expected parameters.

If there are no errors, the server will return a confirmation for the corresponding notification message that needs to be sent to the corresponding recipients. The system provides a SaaS SMTP Service and by default will deliver the email notifications to the recipients so that client applications do not need to worry about this action. However, some clients may prefer to use their own SMTP notification implementation or even alternative mechanism, in which case the system will return with the appropriate information to notify these users. In particular, the pre-populated content of the notification email in TEXT and HTML format is included in the response if it was configured in the Web Admin Console. The client application can then directly notify these users using a private mechanism.

**Request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msgGuid</td>
<td>String / Mandatory</td>
<td>This is the Global Unique Identifier (GUID) of the secure message you are sending. When you start creating a new secure message by calling <code>preCreateMessage</code> command you will receive this <code>msgGuid</code> identifier. You should store this value for the length of the transaction as it is used by all the remaining messaging commands that complete the information for the new message.</td>
</tr>
<tr>
<td>messagePassword</td>
<td>String / Optional</td>
<td>Use the <code>messagePassword</code> parameter in conjunction with the <code>uniquePass</code> F.Y.E.O. message option. This password corresponds to the unique secret that needs to be known by the recipients when accessing the message.</td>
</tr>
<tr>
<td>craCode</td>
<td>String / Optional</td>
<td>The <code>craCode</code> parameter corresponds to the Challenge Response Authentication (CRA) feature that can be assigned to this message. When a user invites a new user and the CRA option is enabled in the Secure Messaging platform, they must assign a code or password that the new user must enter when registering. This can be set to the new user’s customer ID or PIN number assigned by your organization.</td>
</tr>
<tr>
<td>createNewMembers</td>
<td>Boolean / Mandatory</td>
<td>Use this Boolean parameter to specify if the system should automatically invite new users that are included in the list of recipients of this message, but do not exist in the system yet. If you specify a false value for this parameter the server will return an error if the message includes new users in the list of recipients. This is an alternative and more direct mechanism to provision new users. If you are using this API you may prefer to use the <code>addMember</code> command to have greater control over the privileges and quotas the new user is being assigned to. For example, when you provision a new user using the <code>addMember</code> command you can specify the User Group this user should be assigned to, which governs the privileges assigned to the new user. Using this method you can also by-pass other restrictions that may be set at the Secure Messaging platform level like domains that are not allowed explicitly. You can also either assign a password for the new user or pre-register them in the system.</td>
</tr>
</tbody>
</table>
When you allow auto-provisioning new users with this parameter, all the default provisioning rules are used to determine if the operation is valid. For example, does the sender user have enough privileges to invite new users? Have all the invitations allowed to a user being exhausted for the current day? Is the domain allowed? There are also rules that control the automatic assignment of the User Group to the new user according to the Web Admin Console settings.

Allowing the creation of new users in this automatic way is provided so that client applications may duplicate the functionality provided in the native Secure Webmail application that uses this same API to allow users to invite new users just by sending them a secure message.

The recommended workflow for a client application using this command is to always pass a false value in this parameter and check for the specific error code that notifies that new members have been detected. Then, according to the specific requirements of the LOB implemented, the client application may call the same command again passing the true value in this parameter to allow the operation.

Response parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>useEmail2SmtpServer</td>
<td>Boolean</td>
<td>This value indicates if the server sent the corresponding email notifications to the intended recipients. If you receive a true value in this parameter you do not need to do anything else. However, if you receive a false value in this parameter your client application must take care of sending the appropriate email notification (or any other notification mechanism your LOB application may support). Currently the configuration to specify if the server should use the provided SaaS solution for SMTP notifications is set at the Web Admin Console level and cannot be overridden for individual messages. Whether or not your client application must send the notifications to the intended recipients, the server will return the content of such notification in TEXT and/or HTML format according to the Web Admin Console settings. The content of these notifications were created using the notification templates configured in the Web Admin Console and were parsed with the individual information appropriate for this message.</td>
</tr>
<tr>
<td>imBodyHTML</td>
<td>String</td>
<td>This is the HTML content of the email notification message sent to the recipients of this secure message. Depending on the value of useEmail2SmtpServer, the client application must use this content and notify the corresponding recipients. If the useEmail2SmtpServer is true, the client application does not need to send another notification message, although it may be required depending on the specific requirements of the LOB. This HTML message includes the compressed format for the secure message key.</td>
</tr>
<tr>
<td>imBody</td>
<td>String</td>
<td>This is the TEXT version of the imBodyHTML output parameter.</td>
</tr>
</tbody>
</table>

Error codes

1206001 Unexpected / Unknown error creating the secure message.
If you receive this error code please contact support immediately.

1206002 Unknown Secure Messaging platform / Unknown Session.
The session token was assigned to a different Secure Messaging platform than the specified URL or the session has expired.

1206003 Wrong Request format.
This is a valid JSON object, but the request has an invalid format. You may receive this message when you do not include mandatory fields or assigned an invalid value to one of the fields (e.g., wrong data type)
The message cannot be sent because the following user(s) have blocked you - <list of users>

This message cannot be sent because it contains the following banned keywords - <list of keywords>

You don't have privileges to invite new user.
You are inviting new user with this message and your User Group does not allow you to do that or you do not have enough invites available.

CRA code required to invite new user.

You cannot save draft on an invitation messages.
Invitation messages do not allow saving drafts.

The following recipient(s) are not allowed to receive messages from this Secure Messaging platform - <list of users>

This user does not have access to this action.
The User Group assigned to the user does not allow sending new secure messages or reply / forward action for an existing message.

Cannot forward.
The message you are trying to forward does not allow this action.
You may receive this error when replying to a message and adding new recipients to the conversation. In this case the reply action is considered a forward action.

Message terminated.
The parent message you are replying / forwarding has been terminated and access to it cannot be granted to continue the conversation.

Message recalled.
The parent message you are replying / forwarding has been recalled and access to it cannot be granted to continue the conversation.
Examples

The following confirms and sends a previously created secure message for the user currently logged in (determined by the session token). We allow auto-provisioning new users:

```json
01 {  
02   "header": {  
03     "command": "sendMessage",  
04     "timestamp": new Date(1297725429092),  
05     "token": "9c885fa6-9b57-4c1f-89909",  
06     "clientName": "email2.TestApp",  
07     "clientVersion": "1.0"  
08   },  
09   "data": {  
10     "msgGuid": "23C9A222-1BE2-4670-AF1E-15A38027059D",  
11     "createNewMembers": true  
12   }
}
```

The response below allows the action including the content for the email notifications that were sent out to the recipients as well as the final indication that the server delivered the notifications:

```json
01 {  
02   "header": {  
03     "command": "sendMessage",  
04     "timestamp": new Date(1297725441357),  
05     "errCode": 0,  
06     "errDesc": "",  
07     "serverVersion": "2.0.4058"  
08   },  
09   "data": {  
10     "useEmail2SmtpServer": true,  
11     "imBody": "*** email2 Private Email Network ***\r\n\nQA TesterOne has sent you a confidential message using email2 Private Email Network. Follow the secure link below to retrieve your confidential message from QA TesterOne:\r\n\nhttps://secure-messaging.com/code/web/?e=user.two@corp-national.com&mg=23C9A222-1BE2-4670-AF1E-15A38027059D\r\n\r\nDo not reply to this notification message. If you do not know or trust User One (user.one@corp-national.com) or to unsubscribe please visit http://www.secure-messaging.com/unsubscribe\r\n\r\nBEGIN E-MAIL2 SIGNATURE\r\n\n\nBEGIN E-MAIL2 SIGNATURE\r\n\nEND E-MAIL2 SIGNATURE\r
```

---
User One has sent you a confidential message using Corp National Secure Mail.

Follow the secure link below to retrieve your confidential message from User One.

https://secure.email2.com/open/web/?e=qa-two@email2.com&mg=23C9A222-1BE2-4670-15A38027059D

Do not reply to this notification message. If you do not know or trust QA TesterOne (qa-one@email2.com) or to unsubscribe please visit http://www.email2.com/abuse.

---

(Sent using SaaS SMTP Service)
Get Message List

Command code: `getMessageList`

Use the `getMessageList` command to retrieve a list of secure messages sent or received by the user based on filters and sorting criteria.

The information for each message returned by this command is limited to only the minimum identifying information. The idea is to obtain a quick list of all the messages available and use other commands (e.g., `getMessageDetails`) to obtain more specific information about that message.

Third party application integrators are encouraged to cache this list locally and query the server to obtain new messages on regular basis. The list of messages returned by this command is limited to a page size controlled by the caller. You should iterate through the message pages until you have obtained the complete list of messages.

**Request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageNum</td>
<td>Numeric / Mandatory</td>
<td>Use this value to choose which page to return based on the filters and sorting applied. The amount of message per page is configured via the <code>pageSize</code> parameter. The <code>pageNum</code> value must be an integer value between 1 and the maximum number of pages according to the <code>pageSize</code> parameter.</td>
</tr>
</tbody>
</table>
| responseType  | String / Mandatory | This value dictates the response returned to the client. We allow these options to both return cleaner and more succinct responses as well as returning this information more quickly and efficiently as fewer resources are needed. Currently, these are the supported values for this:  
  o `idsOnly`. Using the `idsOnly` option will return only the message guid for all messages returned in the response.  
  o `basic`. When using the `basic` option most metadata values are returned in the response excepting the full list of recipients and the body of the message. This option includes the date and time of when the message was sent and the subject of the message. Use this option when you want to create a list of messages the user can access. |
| pageSize      | Numeric / Mandatory | The `pageSize` parameter dictates the size of the pages of the messages returned in the response. Please notice that this value cannot be greater than 200. |
| filter        | Object / Optional  | Use this parameter to describe the filter applied to the messages returned by this command. This parameter is used to filter the returned messages based on which group or folder these correspond to. Currently, the following values are supported:  
  o `inbox`. Use the `inbox` option to return only messages that have been sent to you.  
  o `sent`. Use the `sent` option to return only messages that you have sent.  
  o `draft`. Use the `draft` option to return messages that are still being edited.  
  o `archive`. Use the `archive` option to return messages that have been archived by the user.  
  o `trash`. Use the `trash` option to return messages that have been put in the trash by the user. |
Secure Messaging

Platform – Client Services API Admin User Guide

This parameter is used to filter the returned messages based on specific content in the message subject and the email address of the sender or recipient.

Note: Messages are stored encrypted at rest on the server. As a result searches cannot be performed on the content of the messages.

Example:

// Sent messages with subject containing the word “test”
"filter":
{
   "group": "sent", "searchCriteria": "test"
}

sort

Use this parameter to describe the sorting applied to the messages returned by this command. If this option is not specified a default sort criteria is applied.

Example:

// Sort messages by date in descending order
"sort":
{
   "type": "date", "direction": "desc"
}

Response parameters

The response is a list of messages. How much information is included for each message in the list depends on the value passed to the responseType parameter in the request.

Below is included the description of all the fields that may come back in the response for each message.

List of messages matching the criteria in the request. Each message will contain the identifier (guid) and depending on the responseType it will also include the remaining fields.

The guid is a global unique identifier for the message. Available with all values of responseType.

The recipients field is a list of all the recipients in the message. This field is available only with the basic value for responseType, and when the filter field is equals to sent in the input parameters.

Each recipient contains the following attributes:

- email. This is the email address of the recipient.
- field. This is the field in which the recipient appears in the message header. It can be one of the...
following values:

- **to.** The TO field of the message.
- **cc.** The Cc field of the message.
- **bcc.** Bcc field of the message.
- **firstName.** The first name of the recipient.
- **lastName.** The recipient’s last name.

Please notice that the name of the recipient is obtained from the live user’s profile. That means that if you call this service a second time you may get a different value for the recipient’s name if the corresponding user profile is changed.

<table>
<thead>
<tr>
<th>.sender</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <strong>sender</strong> field is the original sender of the message. This field is returned with all values of the <strong>responseType</strong> parameter.</td>
<td></td>
</tr>
<tr>
<td>Each sender contains the following attributes:</td>
<td></td>
</tr>
<tr>
<td>- <strong>email.</strong> This is the email address of the sender.</td>
<td></td>
</tr>
<tr>
<td>- <strong>firstName.</strong> The first name of the sender.</td>
<td></td>
</tr>
<tr>
<td>- <strong>lastName.</strong> The sender’s last name.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.subject</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <strong>subject</strong> field corresponds to the message’s subject line. This field is available only with the <strong>basic</strong> value for <strong>responseType</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.method</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Values: received, sent</strong></td>
<td></td>
</tr>
<tr>
<td>The <strong>method</strong> field describes the access type you have to this message. If you are the sender of the message you will have <strong>sent</strong> access, whereas if you have received the message as one of the recipients or another delegated method you will have the <strong>received</strong> access.</td>
<td></td>
</tr>
<tr>
<td>The access <strong>method</strong> also controls the level of information returned for each message.</td>
<td></td>
</tr>
<tr>
<td>This field is returned with all values of the <strong>responseType</strong> parameter.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.date</th>
<th>Datetime (UTC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <strong>date</strong> field corresponds to the date and time the message was sent.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.status</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Values: pending, active, recalled, uploadingAttachments, draft</strong></td>
<td></td>
</tr>
<tr>
<td>The <strong>status</strong> field indicates the current status of the message. This field is returned with all values of the <strong>responseType</strong> parameter and it can be one of the following values:</td>
<td></td>
</tr>
<tr>
<td>- <strong>pending.</strong> The message has not been delivered yet. Typical scenarios include a message has just been created or it is waiting for attachments to be uploaded prior to send the notification, but no other content information has been provided.</td>
<td></td>
</tr>
<tr>
<td>- <strong>active.</strong> The message was delivered and it’s currently available to the sender and all authorized recipients.</td>
<td></td>
</tr>
<tr>
<td>- <strong>recalled.</strong> The sender of the message has recalled this message and is no longer available to the recipients of this message (it is still available to the sender)</td>
<td></td>
</tr>
<tr>
<td>- <strong>uploadingAttachments.</strong> The client application has started uploading attachments, but has not completed the operation yet.</td>
<td></td>
</tr>
<tr>
<td>- <strong>draft.</strong> The message has not been sent yet and it contains draft information submitted earlier.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>.hasAttachments</th>
<th>Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>This Boolean value is based on the message containing attachments or not. This field is available only with the <strong>basic</strong> value for <strong>responseType</strong>.</td>
<td></td>
</tr>
</tbody>
</table>
.hasVideo

This Boolean value is based on the message containing a video message or not. This field is available only with the basic value for responseType.

.isRetrieved

This Boolean value indicates if the user corresponding to the session issuing this command has retrieved the message or not. This field is available only with the basic value for responseType.

Use this field to identify unread messages. Please notice that if a message has been read by the user of the current session, it doesn’t mean that other recipients have also retrieved it. For more details about the retrieval status please see the tracking commands.

.fyeoType

Values: disabled, penPass, uniquePass

The fyeoType field indicates if the message was marked with the “For Your Eyes Only” feature. The FYEO feature provides an extra layer of security by allowing setting a password for each individual message that needs to be entered at the time of accessing the message.

This field is available only with the basic value for responseType and it can be one of the following values:

- **disabled**. The current message does not use For Your Eyes Only.
- **uniquePass**. The message has a unique password assigned to it that needs to be provided at the time the message is retrieved.
- **penPass**. In this case, the message does not have a unique password assigned to it, but the retriever needs to provide their own Service credentials in order to access the content of the message. This is an intermediate solution in which a message password doesn’t need to be exchanged between the sender and recipients, but it asks for confirmation of each recipients’ credentials before accessing the message.

Error codes

1110001  Unexpected / Unknown error retrieving the message list.
If you receive this error code please contact support immediately.

1110002  Unknown Secure Messaging platform / Unknown Session.
The session token was assigned to a different Secure Messaging platform than the specified URL or the session has expired.

1110003  Wrong Request format.
This is a valid JSON object, but the request has an invalid format. You may receive this message when you do not include mandatory fields or assigned an invalid value to one of the fields (e.g., wrong data type)

1110004  General error retrieving the message list data.
If you receive this error, please contact support to provide more information.

1110005  General error retrieving specific data for a specific message.
If you receive this error, please contact support to provide more information.

1110006  General error retrieving specific state data for a message.
If you receive this error, please contact support to provide more information.

1110007  General error retrieving specific method data for a message.
If you receive this error, please contact support to provide more information.
Examples

The following example obtains the 3 most recent messages in the Inbox, including the sender's name:

```json
{  
  "header": {  
    "command": "getMessageList",  
    "timestamp": new Date(1297725429092),  
    "token": "9c885fa6-9b57-4c1f-89909",  
    "clientName": "email2.TestApp",  
    "clientVersion": "1.0"  
  },  
  "data": {  
    "pageNum": 1,  
    "responseType": "basic",  
    "pageSize": 4,  
    "filter": {  
      "group": "inbox"  
    },  
    "sort": {  
      "type": "date",  
      "direction": "desc"  
    }  
  }
}
```

The response includes some basic information for each message, but more importantly, it contains the unique identifier (GUID) for each message in the list. The client application should cache this list and query the server for details about each message or for tracking information:

```json
{  
  "header": {  
    "command": "getMessagesList",  
    "timestamp": new Date(1297725441357),  
    "errCode": 0,  
    "errDesc": "",  
    "serverVersion": "2.0.4058"  
  },  
  "data": {  
    "messages": [  
      {  
        "guid": "A5C919C0-6619-4A19-9223-922D2ECAA8C9",  
        "subject": "Testing secure message with attachments",  
        "sender": {  
          "email": "user.one@corp-national.com",  
          "firstName": "User",  
          "lastName": "One"  
        },  
        "date": new Date(1349480087027),  
        "hasAttachments": true,  
        "hasVideo": false,  
        "isRetrieved": true,  
        "status": "active",  
        "method": "received",  
        "fyeoType": "disabled"  
      },  
      {  
        "guid": "7F077618-72D5-4EA8-9763-0A14E04E0A4",  
        "subject": "Testing simple secure message",  
        "sender": {  
          "email": "user.one@corp-national.com",  
          "firstName": "User",  
          "lastName": "One"  
        },  
        "date": new Date(1349480062603),  
        "hasAttachments": false,  
      }
    ]
  }
}
"hasVideo":false,
"isRetrieved":false,
"status":"active",
"method":"received",
"fyeoType":"disabled"
},

{
"guid": "932497D2-89E4-4178-883F-8E9810C4046F",
"subject": "simple 2",
"sender": {
  "email": "user.two@corp-national.com",
  "firstName": "User",
  "lastName": "Two"
},
"date": new Date(1349395023620),
"hasAttachments":false,
"hasVideo":false,
"isRetrieved":true,
"status":"active",
"method":"received",
"fyeoType":"disabled"}
Get Message Details

Use the `getMessageDetails` command to retrieve the details of a message.

The content available in the response depends on the permissions assigned to the user for this particular message. The user retrieving the message must specify the global unique identifier of the message, the detail level of the information returned, plus an optional unique password if required, and will obtain all the metadata available for that message, plus the content if requested and available to the user.

**Request parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>guid</td>
<td>String / Mandatory</td>
<td>Global unique identifier of the message. You receive this <code>guid</code> for each message when you get the list of messages for the current session.</td>
</tr>
<tr>
<td>messagePassword</td>
<td>String</td>
<td>Use the <code>messagePassword</code> parameter when you know the message has been marked to use For Your Eyes Only (F.Y.E.O.) feature. If the message requires F.Y.E.O. you must provide the unique password for the message if the <code>fyeoType</code> attribute of that message is <code>uniquePass</code>, otherwise if the <code>fyeoType</code> is <code>penPass</code> the user’s password for the service must be provided in this parameter.</td>
</tr>
</tbody>
</table>
| responseType    | String / Mandatory| This value dictates the response returned to the client. The following values are supported:  
  - `full`. All information available for this message is returned to the caller according to the privileges granted to the current session’s user over the message identified by the `guid` parameter.  
  - `metadata`. This option returns all the information available associated to this message except for the actual content of the message (the `body` response field). Use this option when you have already retrieved the message in a previous session and want to refresh the related information or for cases where you want to obtain as much information available for a particular message without marking it as retrieved in tracking. The current user doesn’t need full access to the message to access its metadata. |
| includeExtensions | Boolean | Default: false  
This Boolean value indicates if message extensions should be included in the response for the specified message. Each message can be assigned an arbitrary list of key-value pairs. The system uses some of these extensions to add extended information for each message that is relevant to the internal of the system. For example, the system uses this feature to store the draft list of recipients of the message as part of the native client applications. Your application may or may not assign proprietary sensitive information to this field and as such each third-party application provider should evaluate making this information available as part of the metadata of the message or under full access only. For example, third-party integrators working with Secure Forms should use `includeExtensions = true` when retrieving the full content of the message to retrieve the data for the secure form post. The system uses the `json-contactTag` message extension to store encrypted the content of the secure form post. |

**Response parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>guid</td>
<td>String</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>status</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>creationDate</td>
<td>Datetime (UTC)</td>
</tr>
<tr>
<td>from</td>
<td>Object</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>subject</td>
<td>String</td>
</tr>
<tr>
<td>body</td>
<td>String</td>
</tr>
<tr>
<td>isRetrieved</td>
<td>Boolean</td>
</tr>
<tr>
<td>parentGuid</td>
<td>String</td>
</tr>
<tr>
<td>composeAction</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>String</td>
</tr>
</tbody>
</table>
The **format** specifies the format of the message and it controls how to interpret the **body** field of the message. It can have two values:

- **html**. The content of the message is in HTML format and should be handled accordingly.
- **text**. The content of the message is in TEXT format and should be handled accordingly.

### extensions

The **extensions** field contains a variable list of key-value pairs used by the system and third-party integrators to extend the functionality supported by each message. The key and value members must be of String value.

Note for C# developers: This key-value pairs list match the C# type Dictionary<string,string>

Each message may contain a different number of extensions depending on the context in which the message was created. The system makes use of the following keywords:

- **draft-to, draft-cc, draft-bcc**. Used by the Webmail and Mobile applications. It contains the list of TO, CC, and BCC recipients for a message while in compose mode.
- **notificationEmailList**. List of notification email addresses used by the system when sending email notifications for a secure message.
- **json-contactTag**. This is a serialized JSON string which contains the information associated to a secure form post. If a message contains this key in the **extensions** list the message corresponds to a secure form submission. The deserialized JSON value has the following structure:
  - **email**. The email address of the user submitting the secure form. This is an optional feature supported by secure forms that allow the native client applications like the Webmail client and the Outlook Toolbar to use this email address when the recipient of the secure form post replies to the message, so that the response is directed automatically to the intended person that submitted the secure form and not the system's email address from where the notification was delivered.
  - **items**. This is another key-value pair list with all the INPUT fields included in the POST action of the secure form post. The content of this list depends completely on the design used by the client for their secure form.

### options

This is the list of message options currently attached to this secure message. Each item in the list has a dynamic type value. For example, a boolean value for **allowTracking** indicates that the option can be enabled or not (true/false).

Below is the current list of supported message options. Some of these options may not be enabled and as such setting a value for them may not have an impact on the corresponding message.

<table>
<thead>
<tr>
<th>Option</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.allowReply</td>
<td>Boolean</td>
<td>Specifies if this secure message accepts reply actions.</td>
</tr>
<tr>
<td>.allowForward</td>
<td>Boolean</td>
<td>Specifies if this secure message accepts forward action.</td>
</tr>
<tr>
<td>.allowTracking</td>
<td>Boolean</td>
<td>Specifies if this secure message will have tracking available for its recipients.</td>
</tr>
<tr>
<td>.shareTracking</td>
<td>Boolean</td>
<td>Specifies if the sender of this secure message will be able to share tracking information with recipients.</td>
</tr>
<tr>
<td>.forYourEyesOnlyType</td>
<td>String</td>
<td><strong>DEPRECATED</strong> This field is being deprecated. Please refrain from using this field as a future release of the server will not include this field anymore. Please use <strong>fyeoType</strong> instead.</td>
</tr>
<tr>
<td>.fyeoType</td>
<td>String</td>
<td>Values: disabled, penPass, uniquePass</td>
</tr>
</tbody>
</table>
The **fyeoType** field indicates if the message was marked with the “For Your Eyes Only” feature. The FYEO feature provides an extra layer of security by allowing setting a password for each individual message that needs to be entered at the time of accessing the message.

It can be one of the following values:

- **disabled**. The current message does not use For Your Eyes Only.
- **uniquePass**. The message has a unique password assigned to it that needs to be provided at the time the message is retrieved.
- **penPass**. In this case, the message does not have a unique password assigned to it, but the retriever needs to provide their own Service credentials in order to access the content of the message. This is an intermediate solution in which a message password doesn’t need to be exchanged between the sender and recipients, but it asks for confirmation of each recipients’ credentials before accessing the message.

### attachments

This is the list of file attachments associated to this secure message.

Each item in the list represents an instance of an attachment object. Below is the description of each attachment field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.guid</td>
<td>String</td>
<td>The guid field is a global unique identifier for this attachment. It is used to identify this attachment in next operations, like retrieving the attachment.</td>
</tr>
<tr>
<td>.status</td>
<td>String</td>
<td>The status field specifies the current status of this attachment. These are the values currently supported:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>o <strong>wait</strong>. The attachment specification was created in the system, but no parts of the file have been uploaded yet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>o <strong>upload</strong>. The file attachment is currently being uploaded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>o <strong>active</strong>. All attachment parts have been uploaded successfully. The attachment is now available to be downloaded by approved users.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>o <strong>cancel</strong>. The attachment was declared as part of the message, but the upload action was cancelled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>o <strong>deleted</strong>. The attachment was deleted by its owner.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>o <strong>recalled</strong>. The attachment was recalled by the owner. This means that the attachment is still available to the owner, but recipients of the message where this attachment was included no longer have access to it.</td>
</tr>
<tr>
<td>.fileName</td>
<td>String</td>
<td>The file name of the attachment</td>
</tr>
<tr>
<td>.fileSize</td>
<td>Numeric</td>
<td>The file size of the attachment in bytes.</td>
</tr>
<tr>
<td>.partCount</td>
<td>Numeric</td>
<td>The number of parts the file attachments has been split on. By default the system splits each file attachments in chunks of 4MB.</td>
</tr>
<tr>
<td>.uploadedBy</td>
<td>String</td>
<td>This is the original owner of this attachment.</td>
</tr>
<tr>
<td>.uploadDate</td>
<td>Datetime (UTC)</td>
<td>This is the date and time when this attachment was uploaded.</td>
</tr>
</tbody>
</table>
| .extendedProperties | Object / Optional | The extendedProperties field is a free-form object stored securely ‘as-is’. It is intended to be used by client applications for their own specific purpose. It allows storing specific LOB (line of business) information to a file attachment that is relevant only to the client application making use of it. For example, it may be used to
add an external or third party system identifier for this file in the system to establish a relationship with the external system. The content of this object may even be encrypted with a proprietary mechanism known only to the client application provider to further increase the confidentiality of sensitive information. The system will securely store this information along with all the meta-data associated to the secure message. Given the inherent flexibility of this mechanism your application must be ready to deal with a dynamic list of attributes in this field. This list of attributes may even be different for each attachment.

Error codes

**112001** Unexpected / Unknown error retrieving the message list.
If you receive this error code please contact support immediately.

**112002** Unknown Secure Messaging platform / Unknown Session.
The session token was assigned to a different Secure Messaging platform than the specified URL or the session has expired.

**112003** Wrong Request format.
This is a valid JSON object, but the request has an invalid format. You may receive this message when you do not include mandatory fields or assigned an invalid value to one of the fields (e.g., wrong data type)

**112004** General error validating the message.
One or more of the validation rules to access a message were not met. Typical scenarios include user credentials do not allow the action over the specified message, the message doesn’t exist.
For more details, please contact Support to provide more information.

**1120915** Validation access check failed: Message recalled.
The message has been marked as recalled and the current user is not the owner of the message.

**112036** Validation access check failed: Message is F.Y.E.O. and session credentials are invalid.

**112037** Validation access check failed: Message is F.Y.E.O. and unique password was not provided.

Examples

The following example retrieves the metadata information for a message for which the current user is a valid recipient:

```json
01 {  
02   "header": {  
03     "command": "getMessageDetails",  
04     "timestamp": new Date(1297725429092),  
05     "token": "9c885fa6-9b57-4c1f-89909",  
06     "clientName": "email2.TestApp",  
07     "clientVersion": "1.0"  
08   },  
09   "data":{  
10     "guid":"2E68C540-692C-4C8F-92D3-1068FE1821F1",  
11     "responseType":"metadata"  
12   }  
13 }
```
The response indicates the message is valid and all the access validation rules were satisfied. In this case the message contains two attachments and has not been retrieved yet by the caller. The message also indicates that it was a new message (not a reply or forward action) and has only one recipient. The response also includes the message options allowed for this message.

```js
01 {
02   "header": {
03       "command": "getMessageDetails",
04       "timestamp": new Date(1349900307145),
05       "errCode": 0,
06       "errDesc": ":",
07       "serverVersion": "2.0.4058"
08   },
09   "data": {
10       "guid": "2E68C440-692C-4C8F-92D3-1068FE1821F1",
11       "status": "active",
12       "creationDate": new Date(1349470166857),
13       "from": {
14           "email": "user.one@corp-national.com",
15           "firstName": "User",
16           "lastName": "One"
17       },
18       "subject": "Testing secure message with attachments",
19       "isRetrieved": false,
20       "hiddenRecipients": false,
21       "composeAction": "newMessage",
22       "format": "HTML",
23       "recipients": [
24           {
25               "field": "to",
26               "email": "user.two@corp-national.com",
27               "firstName": "User",
28               "lastName": "Two"
29           }
30       ],
31       "options": {
32           "allowReply": true,
33           "allowForward": true,
34           "allowTracking": true,
35           "shareTracking": true,
36           "fyeoType": "disabled"
37       },
38       "attachments": [
39           {
40               "guid": "6943bd7b-def8-43ee-88fd-8323cf09564f",
41               "status": "active",
42               "fileName": "Copy of ap_trans.xlsx",
43               "fileSize": 30044,
44               "partCount": 1,
45               "uploadedBy": "user.one@corp-national.com",
46               "uploadDate": new Date(1349470166170),
47               "extendedProperties": {}  
48           },
49           {
50               "guid": "e2004156-8971-4369-845d-d7ed5e6a1212",
51               "status": "active",
52               "fileName": "Copy of ap_trans3.xlsx",
53               "fileSize": 23397,
54               "partCount": 1,
55               "uploadedBy": "user.one@corp-national.com",
56               "uploadDate": new Date(1349470166623),
57               "extendedProperties":{}
58           }
59       ]
60   }
61 }
62 ```
Delete Message

Use the **deleteMessage** command to remove a message from the list of active messages.

Deleting a message only hides its content from active lists. For auditability purposes messages cannot be deleted from the system. They get moved to a deleted state where the message is no longer available by any of the participants of that message.

In order to issue this command, the current session's user profile must have enough privileges to invoke this feature.

### Request parameters

**guids**

List(String) / Mandatory

Global unique identifier of the messages that you need to delete.

You receive this **guid** for each message when you get the list of messages for the current session.

### Response parameters

This command does not return any data section, only the content of the header section of the response.

### Error codes

**1150001** Unexpected / Unknown error deleting a message.

If you receive this error code please contact support immediately. An unexpected/unhandled error was raised after executing this command.

**1150002** Wrong Request format.

The format of the request is not supported by the server or the request data is missing.

**1150003** Wrong Request format. Missing list of messages to delete.

This is a valid JSON request, but there are no message GUIDs specified with the operation.

**1150004** There was a problem deleting secure message. Please try again.

There was a problem with the delete operation. One or more of the specified message GUIDs do not exist or you do not have privileges to delete the specified messages.

### Examples

The following example attempts to delete one message given the message’s GUID:

```json
01  {
02    "header": {
03      "command": "deleteMessage",
04      "timestamp": new Date(1297725429092),
05      "token": "9c885fa6-9b57-4c1f-89909",
06      "clientName": "email2.TestApp",
07      "clientVersion": "1.0"
08    },
09    "data":{
10      "guids": ["944C446F-D2C4-4378-8FF6-6C457162031D"
11    ]
12  }
13  }
14 }
```
The response includes an error code and error description as a result of the operation. In this case, the response indicates the operation succeeded:

```json
{
  "header": {
    "command": "deleteMessage",
    "timestamp": new Date(1349900307145),
    "errCode": 0,
    "errDesc": ",
    "serverVersion": "2.0.4058"
  },
  "data": {}
}
```
Appendix: email2 JSON Data Types

This is the list of data types supported by the parameters included in the request / response objects.

The API's request / response JSON objects follow the specification described below. For a general description of the JSON specification, see http://www.json.org/.

String

A string is a collection of zero or more Unicode characters, wrapped in double quotes, using backslash escapes:

A character is represented as a single character string. A string is very much like a C or Java string.

Example:

"firstName" : "John",
"description" : "This is a "description" using the escape sequence"

Number

A number is very much like a C or Java number.
**Example:**

"age" : 4,
"salary" : 120.52

**Boolean**

Boolean values are specified by the words true and false.

**Example:**

"useDefault" : true

**DateTime**

In the API, date time values correspond to a standard JavaScript Date constructor. The parameter for the constructor is an integer that represents the number of milliseconds since 01/01/70 00:00:00.

Unless specified otherwise, all commands in the API store dates in GMT format. You must do the appropriate conversion from your local time zone when setting/retrieving date time parameters.

A sample implementation of how to calculate the milliseconds value in C# will be like this:

```csharp
long val = (dateTime.Ticks - (new DateTime(1970, 1, 1)).Ticks) / (long)10000;
```

The example below corresponds to January 6th 2009, 5:17:45 AM GMT, which is the same than January 5th 2009, 9:17:45 PM PST (GMT -8).

**Example:**

"created" : new Date(1231219065979)

**List / Array**

A list or array is an ordered collection of JSON values. An array begins with [ (left bracket) and ends with ] (right bracket). Values are separated by the , (comma) character.
A value can be a string in double quotes, or a number, or true or false or null, or an object or an array. These structures can be nested.

**Example:**

```
"contacts" : ["support@email2.com", "notifications@email2.com"],
"recipients" : null
```

**Object**

A JSON Object is an unordered set of name/value pairs.

```
An object begins with { (left brace) and ends with } (right brace). Each name is a string followed by : (colon) and then the value. The name/value pairs are separated by , (comma) character.

Whitespace can be inserted between any pair of tokens.

Because the order is not important, members of the JSON object can be placed anywhere within the opening and closing braces. The following two examples are equivalent representations of the same request command:

**Example:**

```json
{"data":{"emailAddress":"qa-four@email2.com"},
"header":{"command":"getMember","timestamp":new Date(1230743251241),"token":"1ea13a3d-ebc3-44ed-9392-cb5dca16a704","clientName":"apiNetTest","clientVersion":"0.1.0.0"}}
```
Example:
{
    "header": {
        "command": "getMember",
        "timestamp": new Date(1230743251241),
        "token": "1ea13a3d-ebc3-44ed-9392-cb5dca16a704",
        "clientName": "apiNetTest",
        "clientVersion": "0.1.0.0"
    },
    "data": {
        "emailAddress": "qa-four@email2.com"
    }
}

An Request / Response correspond to an unnamed JSON Object.