AUTOCRYPT

PnC Secondary Actor Service API

Test API for MO, OEM, CPO and CPS

User manual and tutorial for the complete API
Version 1.0.0

AUTOCRYPT
v2gtest@autocrypt.io

Index

Version History

Version	Date	Comment
1.0.0	17 Feb 2020	Initial release
1.1.0	6 Mar 2020	text revision

Introduction

ISO/IEC 15118-2 describes the connection and communication between EVCC and SECC that it seems there is not much things to consider outside the scope of the standards as testing the interoperability between EVCC and SECC only. But the real V2G PnC eco-system is composed of other members so called the secondary actors such as MO, CPO, CPS and OEM. The secondary actors actually complete and define how the security of the charging environment can be achieved and how the payment and billing is performed. So, in the perspective of establishing the interoperability of the whole working PnC eco-system, tests including the secondary actors as described ISO 15118-2 and VDE manual (VDE-AR-E 2802-100-1:2017-10) is required. While the secondary actors are not ready yet to join the public test events such as Testing Symposium, the EVCC and SECC need some alternative way to test the interoperability with the backend environment. Here the secondary actor services including MO, CPO, CPS and OEM is presented for the requirement.

The V2G PnC secondary actor services are provided for the SECC and EVCC. The services include PKI related security and operational functions. The PKI functions works for issuing keys and certificates, verifying them according to the trust chains, and the operation functions take the role of delivering contract data to EV and authenticate the validity while charging. The detailed usages depending on EVCC and SECC are described including application tutorials.

Service Tutorial

2.1. Command-Line Interface

For each API functions, example cURL commands are provided to test if the service operates normally. cURL aka curl is a popular command line tool for transferring data using various network protocol. In this service, curl is used to send and receive PKI related requests and responses. Typical curl examples are presented in the API description table, but modifying the commands is easy to apply for various use-cases. Some of the tips are presented below. Before beginning, please install "curl" if not, for Mac and Ubuntu curl is included in the OS deployment by default and for Windows, it is very easy to install, and it is a free software.

First, let's begin with the simplest API which requests a certificate for a SECC. The API can be executed using the following curl command.

```
curl -X POST -d '{"seccId": "secc001"}' -H "Accept: application/json" -H "Content-Type: application/json" -w "\n\nResponse code : [%{http_code}]\n200 : Success\n405 : Error JSON request format\n\n" "http://v2gtest.autocrypt.io:15118/api/v2/cpo/seccCert"
```

Do not forget to change the SECC ID denoted in red to your own unique name, for example, adding your company name as prefix. If you do it the right way the expected response from the server is as follows.

```
{
    "status": "success",
    "message": "",
    "data": {
        "id": "du58FnQsSGG2U48q4nukXw==",
        "seccId": "secc001",
        "provCert": " MII...",
        "pfxCert": " Mizik...",
        "subjectDn": "CN=secc001, DC=CPO, OU=autocrypt, O=verisco GmbH, C=DE",
        "notBefore": "2019-11-21",
        "notAfter": "2020-02-22",
        "serial": "1000000000077",
```

```
        "version": "3",
        "sigAlgo": "SHA256WITHECDSA",
        "issuerDn": "DC=V2G, C=DE, O=verisco GmbH, CN=PKI-1_CRT_CPO_SUB2_VALID",
        "status": "VALID"
    }
}
Response code: [200]
200: Success
405: Error JSON request format
```

Looking at the response message, you will find certificate ID, corresponding SECC ID, the certificate for the SECC and other useful information such as subject DN which is used on revocation later. One more thing to keep your eyes on is the "O=verisco GmbH" in the subject DN and issuer DN section. In this secondary actor service, V2G root of 12th Testing Symposium is applied and all the sub CA's the constructed accordingly. The certificates this service manages are only leaf certificates of each entities of EVCC, SECC and charging contract. Issuing sub CA certificates will be provided soon but not available in the version. Did you notice that the request and response are transmitted in JSON format? If the request JSON format has a problem the response code 405 will be displayed.

2.2.Curl Command Detail

Looking back on the previous curl command,

```
curl -X POST -o outfile.json -d '{"seccId": "secc001"}' -H "Accept: application/json"
-H "Content-Type: application/json" -w "\n\nResponse code : [%{http_code}]\n200 :
Success\n405 : Error JSON request format\n\n"
"http://v2gtest.autocrypt.io:15118/api/v2/cpo/seccCert"
```

By adding the "**-o outfile.json**" option the result is saved as specified file path instead of being displayed in the console screen and discarded. No matter if the certificate information is already displayed and disappeared but the certificate is required later in the other situation. All the information can be downloaded using the other API provided.

If you are not familiar with REST API usage with HTTP(S), please take some time for googling or ask us some help to understand how it works and how to apply types of *method*. There are 'GET', 'PUT', 'POST' and 'DELETE' methods. Please be noted that always apply the right method as specified in the API document. In the previous example, 'POST' method is used and denoted in green in the code snippet. To apply other methods, the green word is the right place for the new method.

Now, take a look at the blue region of the command. In the "http://v2gtest.autocrypt.io" is the API base DNS, and a joke is the port number 15118. The now "http" is used to skip TLS certificate issue and make the testing easier, but "https" will soon be applied to reflect real PnC back-end environments. "api/v2/cpo/seccCert" is the API address for the specific function of managing SECC certificates. Please be sure to type in the right API name in the section.

And for the last part, the request information in the JSON can be found as '{"seccId": "*secc001*"}'. This means the certificate issue request is made with one input parameter of unique SECC ID. For the other APIs the the request JSON may contain several fields other than one. For those cases, the input JSON can be supplied as file path as follows.

```
curl -X POST -o outfile.json -d @file_path.json -H "Accept: application/json" -H
"Content-Type: application/json" -w "\n\nResponse code : [%{http_code}]\n200 :
Success\n405 : Error JSON request format\n\n"
"http://v2gtest.autocrypt.io:15118/api/v2/cpo/seccCert"
```

The remaining part is just the help message for response codes.

2.3. Contract Creation

It is needless to say the most complex and important procedure in the PnC world is generation of charging contract. One of the reasons which makes it complicated is that the generation request contains EV ID instead of EV certificate. MO(Mobility Operator, or EMSP) who is making a contract have to query the EV ID (aka OEM Provisioning ID, PCID) to OEM certificate pool to accept the EV certificate issued by OEM. But what makes it really complicated is the packaging process of the contract data required for the secure delivery to the EVCC through CPS, CCP, CPO and SECC. The process includes hashing, signing, key sharing and encoding operations that in turn

leaves the same amount of burden to the EVCC side. However, the test service provides simplest possible interface to test if the valid contract is made and the delivering channel works fine and finally the charging process operates normally with the delivered contract information.

Suppose an EV owner who wants to make a new contract goes to a MO and submit PCID. Then, MO requests authenticity and validity check to OEM for the OEM certificate corresponding the PCID. For the verified PCID, MO again requests the certificate. The purpose of pursuing the OEM provisioning certificate is that the public key from the certificate is used to encrypt the secret key destined to be used on decrypting contract private key.

To make things short, in this service, the vehicle is given with an arbitrary PCID. EVCC tester asks OEM to issue the vehicle certificate corresponding to the PCID. EV owner (or EVCC tester) submits the PCID to MO. MO checks PCID and gets the OEM certificate. MO creates contract data using the OEM certificate and passes all the required information to CPS. Ok, let's stop here to see the list of APIs used for to the aforementioned process. Please be noted that the process described is not 100% consistent with VDE manual. This is to make the test process easy and simple.

```
EVCC tester request to OEM to issue OEM certificate
curl -X POST -d '{"pcid": "pcid001"}' -H "Accept: application/json" -H "Content-Type:
application/json" "http://v2gtest.autocrypt.io:15118/api/v2/oem/provCert"

EVCC tester (or EV user) submit PCID to MO to generate new contract according to the
below
{
     MO asks OEM to check PCID
     curl -X GET "http://v2gtest.autocrypt.io:15118/api/v2/oem/verifyByPcid/pcid001"

     MO requests OEM the vehicle OEM certificate
     curl -X GET "http://v2gtest.autocrypt.io:15118/api/v2/oem/provCert/pcid001"

     Requests charging contract creation with OEM certificate and preferred contract ID
     which can be arbitrary.
     curl -X POST -d '{"dhCert": "MIICmjCCAkGgAw…", "emaid": "emaid001"}' -H "Accept:
     application/json"          -H          "Content-Type:          application/json"
     "http://v2gtest.autocrypt.io:15118/api/v2/mo/contract"
```

```
}
```

2.4. Issuing Certificates

The certificates can be categorized into two groups: simple certificates just related to devices and can be installed or assigned directly by the owner of the devices, and, certificates with complex relationships that issuing must be based on a series of authentication between a few actors other than the certificate issuer.

SECC certificates can be directly issued by CPO and installed. OEM certificates can also be issued and installed to EVCC in the same manner. This service provides issuing functions for SECC and EVCC. And renewing and revoking functions are also provided. In addition, those certificates can be downloaded for any purpose after the they are issued. The ID for SECC and EVCC are arbitrary in this version but they must be unique throughout the whole test service system. The current version of the service does not have the concept of account of the service user. So, please be noted that SECC ID and EVCC ID must be kept unique globally. Some ID used by some other service user can cause conflict. For example, "secc001" is used for the tutorial, but if you use that id for your SECC, a message will tell you that the name is already occupied. This is because there is no sign-in process on using the PnC test service that no account is required to use the service. If you need a separate account which allows private namespace, please contact the person in charge. For EVCC ID and contract ID the same rule is applied.

On the other hand, the process for charging contract is performed rather indirect way. As described in the section 2.3., the certificate for the charging contract is published as included in the contract data. The certificate is issued in the creation process of contract creation.

2.5.Downloading Certificates

The functions for downloading certificates are provided for SECC, EVCC respectively. Packaged contract data can also be downloaded in the form of V2G standard messages - CertificateInstallationRes, CertificateUpdateRes. There is no limit on the number of times allowed for downloading.

2.6.Renewing Certificates

The PnC test service also provides certificate renewal functions. The certificate of SECC and EVCC can be easily renewed using straightforward APIs with SECC ID or EVCC ID. Renewing contract certificate is embedded in the process of updating charging contract. The old certificate to be updated is required as input parameter and the revocation is also accompanied with the process.

2.7.Revoking Certificates

Testing if the introduced system successfully refuses to authenticate the revoked certificates is definitely the one of the major test cases, because it guarantees the stability of the business operation. The users can issue certificates and install them into devices, then confirm that they work fine with the system. And the consequences of revoking those working certificates can be checked. By the way, revocation takes different input parameters from issuing and updating. Instead of passing SECC ID or EVCC ID, the subject DN is required to keep from revocation by mistake.

2.8.Requesting V2G Contract Certificates to CPS

Two pairs of the most frequently tests V2G messages can be CertificateInstallationReq/Res and CertificateUpdateReq/Res. The V2G response messages generated in the backend are delivered to CPS and signed then stored. The CPO or CPO server can request the delivery of the contract related V2G response messages where those requests are usually in the form of V2G request. In the actual operation model, CPO should ask the contract to CPS and deliver it to SECC. But, in this test scale service, it is assumed that SECC also takes the role of CPO requesting CertificateInstallationRes and CertificateUpdateRes.

API Reference

3.1. CPO for SECC

3.1.1. Issue SECC Certificate by SECC ID

<table>
<tr><th colspan="3">Issue SECC Certificate by SECC ID</th></tr>
<tr><td>Method</td><td colspan="2">POST</td></tr>
<tr><td>URL</td><td colspan="2">/api/v2/cpo/seccCert</td></tr>
<tr><td rowspan="2">Request Form</td><td colspan="2">{
"seccId": "secc001"
}</td></tr>
<tr><td>seccId</td><td>SECC unique identifier</td></tr>
<tr><td>Command-line Example</td><td colspan="2">curl -X POST -d '{"seccId": "secc001"}' -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118/api/v2/cpo/seccCert"</td></tr>
<tr><td rowspan="12">Response on Success</td><td colspan="2">{
"status": "success",
"message": "",
"data": {
"id": "du58FnQsSGG2U48q4nukXw==",
"seccId": "secc001",
"provCert": " MII...",
"pfxCert": " Mizik…",
"subjectDn": "CN=secc001,DC=CPO,OU=autocrypt,O=verisco GmbH,C=DE",
"notBefore": "2019-11-21",
"notAfter": "2020-02-22",
"serial": "1000000000077",
"version": "3",
"sigAlgo": "SHA256WITHECDSA",
"issuerDn": "DC=V2G,C=DE,O=verisco GmbH,CN=PKI-1_CRT_CPO_SUB2_VALID",
"status": "VALID"
}
}</td></tr>
<tr><td>id</td><td>internal unique id</td></tr>
<tr><td>seccId</td><td>SECC unique identifier</td></tr>
<tr><td>provCert</td><td>SECC certificate encoded in base64</td></tr>
<tr><td>pfxCert</td><td>private key and certificate in PFX format encoded in base64</td></tr>
<tr><td>subjectDn</td><td>certificate SubjectDN</td></tr>
<tr><td>notBefore</td><td>certificate Not Before Date</td></tr>
<tr><td>notAfter</td><td>certificate Not After Date</td></tr>
<tr><td>serial</td><td>certificate Serial Number</td></tr>
<tr><td>version</td><td>certificate Version</td></tr>
<tr><td>sigAlgo</td><td>certificate Signature Algorithm</td></tr>
<tr><td>issuerDn</td><td>certificate IssuerDN</td></tr>
</table>

	status	certificate Status
Response on Failure	{ "status": "error", "message": "Exception Message", "data": "" }	

3.1.2. Issue SECC Certificate by CSR

Issue SECC certificate by CSR		
Method	**POST**	
URL	/api/v2/cpo/seccCertByCsr	
Request Form	{ "pkcs10Data": "MIHQMHgCA... ", "cn": “secc001” }	
	pkcs10Data	base64 encoded standard CSR for X.509 certificate
	cn	unique common name of SECC
Command-line Example	curl -X POST -d '{"pkcs10Data": "MIHQMHgCA...", "cn": "secc001"}' -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118/api/v2/cpo/seccCertByCsr" **or by specifying the file path containing JSON request form** curl -X POST -d @file_path.json -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118/api/v2/cpo/seccCertByCsr" where file_path.json contains { "pkcs10Data": "MIHQMHgCA...", "cn": “secc001” }	
Response on Success	{ "status": "success", "message": "", "data": { "subject": "secc001", "serial": "1000000000078", "certificateData": "Mll…" } }	
	subject	unique common name of SECC
	serial	certificate serial number
	certificateData	SECC certificate encoded in base64
Response on Failure	{ "status": "error", "message": "Exception Message", "data": "" }	

3.1.3. Download SECC Certificate by SECC ID

Download SECC Certificate by SECC ID		
Method	**GET**	
URL	/api/v2/cpo/seccCert/{seccId}	
Request Form	Not required	
Command-line Example	curl -X GET "http://v2gtest.autocrypt.io:15118/api/v2/cpo/seccCert/*seccId*"	
Response on Success	{ "status": "success", "message": "", "data": { "subject": "secc001", "serial": "1000000000078", "certificateData": "Mll…", } }	
	subject	unique common name of SECC
	serial	certificate serial number
	certificateData	SECC certificate encoded in base64
Response on Failure	{ "status": "error", "message": "Exception Message", "data": "" }	

3.1.4. Renew SECC Certificate by SECC ID

Renew SECC certificate by SECC ID		
Method	**PUT**	
URL	/api/v2/cpo/seccCert	
Request Form	{ "seccId": "*secc001*" }	
	seccId	SECC unique identifier
Command-line Example	curl -X PUT -d '{"seccId": "*secc001*"}' -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118//api/v2/cpo/seccCert"	

<table>
<tr><td rowspan="13">Response on Success</td><td colspan="2"><pre>{
    "status": "success",
    "message": "",
    "data": {
        "id": "Ws04DnUFSC6Pq1QTgNISag==",
        "seccId": "secc001",
        "provCert": " MII...",
        "pfxCert": " Mizik…",
        "subjectDn": "CN=secc001,DC=CPO,OU=autocrypt,O=verisco GmbH,C=DE",
        "notBefore": "2019-11-21",
        "notAfter": "2020-02-22",
        "serial": "1000000000078",
        "version": "3",
        "sigAlgo": "SHA256WITHECDSA",
        "issuerDn": "DC=V2G,C=DE,O=verisco GmbH,CN=PKI-1_CRT_CPO_SUB2_VALID",
        "status": "VALID"
    }
}</pre></td></tr>
<tr><td>id</td><td>internal unique id</td></tr>
<tr><td>seccId</td><td>SECC unique identifier</td></tr>
<tr><td>provCert</td><td>SECC certificate encoded in base64</td></tr>
<tr><td>pfxCert</td><td>private key and certificate in PFX format encoded in base64</td></tr>
<tr><td>subjectDn</td><td>certificate SubjectDN</td></tr>
<tr><td>notBefore</td><td>certificate Not Before Date</td></tr>
<tr><td>notAfter</td><td>certificate Not After Date</td></tr>
<tr><td>serial</td><td>certificate serial number</td></tr>
<tr><td>version</td><td>certificate version</td></tr>
<tr><td>sigAlgo</td><td>certificate signing algorithm</td></tr>
<tr><td>issuerDn</td><td>certificate IssuerDN</td></tr>
<tr><td>status</td><td>certificate status</td></tr>
<tr><td>Response on Failure</td><td colspan="2"><pre>{
    "status": "error",
    "message": Exception Message,
    "data": ""
}</pre></td></tr>
</table>

3.1.5. Revoke SECC Certificate by Subject DN

Revoke SECC certificate by subject DN	
Method	**DELETE**
URL	/api/v2/cpo/seccCert

Request Form	{
	"subjectDn": "CN=*secc123*,DC=CPO,OU=autocrypt,O=verisco GmbH,C=DE"
	}		

	subjectDn	Certificate subjectDN	
Command-line Example	curl -X DELETE -d '{"subjectDn":"CN=*secc123*,DC=CPO,OU=autocrypt,O=verisco GmbH,C=DE"}' -H "Accept: application/json" -H "Content-Type: application/json" -w "\n\nResponse code : [%{http_code}]\n200 : Success\n405 : Error JSON request format\n\n" "http://v2gtest.autocrypt.io:15118/api/v2/cpo/seccCert"		

Response on Success	{
	"status": "success",
	"message": "",
	"data": {
	"resultCode": "4000"|"4002"|"4003",
	"message": "revoke result message",
	"serial": "1000000000078"
	}
	}		

	resultCode	4000	Success
		4002	Already Revoked
		4003	Not existing certificate
	message	Revocation result message	
	serial	Certificate serial number	

Response on Failure	{
	"status": "error",
	"message": "Exception Message",
	"data": ""
	}		

3.1.6. Download SECC Certificate and Private Key in pfx format by serial number

Download SECC certificate and private key in pfx format by serial number		
Method	**POST**	
URL	/api/v2/cpo/downloadSeccCert	

Request Form	{
	"serial": "1000000000078",
	"password": "123123"
	}	

	serial	certificate serial number
	password	Password for PFX
Command-line Example	curl -X POST -d '{"serial": "1000000000078", "password": "123123"}' -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118//api/v2/cpo/downloadSeccCert"	

Response on Success	<pre>{
	  "status": "success",
	  "message": "",
	  "data": {
	  "pfxCert": "MIILiAIBAzCCC0E … 1h3n7sCAwGGoA=="
	  }
	}</pre>

	pfxCert: Base64 Encoded PFX data

Response on Failure	<pre>{
	  "status": "error",
	  "message": Exception Message,
	  "data": ""
	}</pre>

3.2. OEM

3.2.1. Issue OEM Provisioning Certificate with PCID

Issue OEM Provisioning Certificate with PCID(Provisioning Certificate ID)	
Method	**POST**
URL	/api/v2/oem/provCert

Request Form	<pre>{
	  "pcid": "pcid001"
	}</pre>

	pcid: EVCC unique ID
Command-line Example	curl -X POST -d '{"pcid": "*pcid001*"}' -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118//api/v2/oem/provCert"

Response on Success	<pre>{
	  "status": "success",
	  "message": "",
	  "data": {
	  "id": "6L/nc/faRaKLmYXXmkaseg==",
	  "pcid": " pcid001",
	  "provCert": “MII...”,
	  "pfxCert": “Mizik…”,
	  "subjectDn": "CN=pcid001,DC=OEM,OU=autocrypt,O=verisco GmbH,C=DE",
	  "notBefore": "2019-09-10",
	  "notAfter": "2019-12-11",
	  "serial": "1000000000079",
	  "version": “3”,
	  "sigAlgo":"SHA256WITHECDSA",
	  "issuerDn": "DC=OEM,C=DE,O=verisco GmbH,CN=PKI-1_CRT_OEM_SUB2_VALID",
	  "status": "VALID"
	  }
	}</pre>

	id: internal unique id
	pcid: EVCC unique id
	provCert: Base64 encoded certificate

	pfxCert	Base64 encoded PFX data
	subjectDn	certificate SubjectDN
	notBefore	certificate Not Before Date
	notAfter	certificate Not After Date
	serial	certificate serial number
	version	certificate version
	sigAlgo	certificate signing algorithm
	issuerDn	certificate issuer DN
	status	certificate status

Response on Failure	<pre>{
	"status": "error",
	"message": "Exception Message",
	"data": ""
	}</pre>	

3.2.2. Download OEM Provisioning Certificate by PCID

Download OEM Provisioning Certificate by PCID(Provisioning Certificate ID)		
Method	**GET**	
URL	/api/v2/oem/provCert/{pcid}	
Request Form	Not Required	
Command-line Example	curl -X GET "http://v2gtest.autocrypt.io:15118/api/v2/oem/provCert/pcid001"	

Response on Success	<pre>{
	"status": "success",
	"message": "",
	"data": {
	"id": "6L/nc/faRaKLmYXXmkaseg==",
	"pcid": " pcid001",
	"provCert": "MII... ",
	"pfxCert": "",
	"subjectDn": "CN=pcid001,DC=OEM,OU=autocrypt,O=verisco GmbH,C=DE",
	"notBefore": "2019-09-10",
	"notAfter": "2019-12-11",
	"serial": "1000000000079",
	"version": “3”,
	"sigAlgo":"SHA256WITHECDSA",
	"issuerDn": "DC=OEM,C=DE,O=verisco GmbH,CN=PKI-1_CRT_OEM_SUB2_VALID",
	"status": "VALID"
	}
	}</pre>	

	id	internal unique id
	pcid	EVCC unique id

	provCert	Base64 encoded certificate
	pfxCert	Base64 encoded PFX data
	subjectDn	certificate SubjectDN
	notBefore	certificate Not Before Date
	notAfter	certificate Not After Date
	serial	certificate serial number
	version	certificate version
	sigAlgo	certificate signing algorithm
	issuerDn	certificate issuer DN
	status	certificate status

Response on Failure	{
	"status": "error",
	"message": "Exception Message",
	"data": ""
	}	

3.2.3. Renew OEM Provisioning Certificate by PCID

Renew OEM Provisioning Certificate by PCID(Provisioning Certificate ID)		
Method	**PUT**	
URL	/api/v2/oem/provCert	

Request Form	{
	"pcid": "pcid001"
	}	

	pcid	Unique OEM provisioning certificate ID
Command-line Example	curl -X POST -d '{"pcid": "*pcid001*"}' -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118/api/v2/oem/provCert"	

Response on Success	{
	"status": "success",
	"message": "",
	"data": {
	"id": "5sNJwDGkTFuLwNz/mj2OKQ==",
	"pcid": "pcid001",
	"provCert": " MII...",
	"pfxCert": " Mizik…",
	"subjectDn": "CN=pcid001,DC=OEM,OU=autocrypt,O=verisco GmbH,C=DE",
	"notBefore": "2019-11-22",
	"notAfter": "2029-11-23",
	"serial": "1000000000080",
	"version": "3",
	"sigAlgo": "SHA256WITHECDSA",
	"issuerDn": "DC=OEM,C=DE,O=verisco GmbH,CN=PKI-1_CRT_OEM_SUB2_VALID",
	"status": "VALID"
	}
	}	

	id	internal unique id
	pcid	EVCC unique id
	provCert	Base64 encoded certificate
	pfxCert	Base64 encoded PFX data
	subjectDn	certificate SubjectDN
	notBefore	certificate Not Before Date
	notAfter	certificate Not After Date
	serial	certificate serial number
	version	certificate version
	sigAlgo	certificate signing algorithm
	issuerDn	certificate issuer DN
	status	certificate status
Response on Failure	{ "status": "error", "message": Exception Message, "data": "" }	

3.2.4. Revoke OEM Provisioning Certificate

Revoke OEM Provisioning Certificate by subject DN			
Method	**DELETE**		
URL	/api/v2/oem/provCert		
Request Form	{ "subjectDn": "CN=pcid003,DC=OEM,OU=autocrypt,O=verisco GmbH,C=DE" }		
	subjectDn	certificate subject DN	
Command-line Example	curl -X DELETE -d *'{"subjectDn": "CN=pcid003,DC=OEM,OU=autocrypt,O=verisco GmbH,C=DE"}'* -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118/api/v2/oem/provCert"		
Response on Success	{ "status": "success", "message": "", "data": { "resultCode": "4000"|"4002"|"4003", "message": "revoke result message", "serial": "1000000000080" }		
	resultCode	4000	success
		4002	already revoked
		4003	not existing certificate

	message	revocation message
	serial	certificate serial number

Response on Failure	<pre>{
	"status": "error",
	"message": "Exception Message",
	"data": ""
	}</pre>	

3.2.5. Verify OEM Provisioning Certificate

Verify OEM Provisioning Certificate by PCID		
Method	**GET**	
URL	/api/v2/oem/verifyByPcid/{pcid}	
Request Form	Not Required	
Command-line Example	curl -X GET "http://v2gtest.autocrypt.io:15118/api/v2/oem/verifyByPcid/*pcid001*"	

Response on Success	<pre>{
	"status": "success",
	"message": "",
	"data": {
	"resCode": "OK"
	}
	}</pre>	

	resCode	OK | FAILED_CertChainError | FAILED_CertificateExpired

Response on Failure	<pre>{
	"status": "error",
	"message": Exception Message,
	"data": ""
	}</pre>	

3.3.MO

3.3.1. Create Contract

Create Charging Contract using OEM certificate and emaid(contract ID)		
Method	**POST**	
URL	/api/v2/mo/contract	

Request Form	{
	"dhCert":"MIICmjCCAkGgAw…",
	"emaid": "emaid001"
	}	

	dhCert	OEM provisioning certificate
	emaid	unique ID for the contract
Command-line Example	curl -X POST -d '{"dhCert": "*MIICmjCCAkGgAw…*", "emaid": "emaid001"}' -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118//api/v2/mo/contract"	

Response on Success	{
	"status": "success",
	"message": "",
	"data": {
	"version": "penta-v2g-0.1",
	"certInstRes": "gJhwGYRUStDo6OB0X…W13DABWlkNAM2t6+ZA"
	}
	}	

	version	V2G message version
	certInstRes	CertificateInstallationRes message (EXI encoded then Base64 Encoded)

Response on Failure	{
	"status": "error",
	"message": Exception Message,
	"data": ""
	}	

3.3.2. Renew Contract

Renew(Update) Charging Contract using the Previous Contract Certificate		
Method	**PUT**	
URL	/api/v2/mo/contract	

Request Form	{
	"dhCert":"MIICmjCCAkGgAw…",
	"emaid": "emaid001"
	}	

	dhCert	expiring contract certificate used to encrypt the private key of the updated contract
	emaid	unique contract ID
Command-line Example	curl -X PUT -d '{"dhCert": "*MIICmjCCAkGgAw…*", "emaid": "emaid001"}' -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118//api/v2/mo/contract"	

Response on Success	<pre>{
	"status":"success",
	"message":"",
	"data":{
	"version":"penta-v2g-0.1",
	"certUpdtRes":"gJhwGYRUStDo6OB0X…W13DABWlkNAM2t6+ZA"
	}
	}</pre>	

	version	V2G message version
	certUpdtRes	CertificateUpdateRes message (EXI encoded then Base64 Encoded)

Response on Failure	<pre>{
	"status": "error",
	"message": Exception Message,
	"data": ""
	}</pre>	

3.3.3. Revoke Contract Data

Revoke Charging Contract by Certificate subject DN		
Method	**DELETE**	
URL	/api/v2/mo/contract	

Request Form	<pre>{
	"subjectDn": "CN=emaid002,DC=MO,OU=autocrypt,O=verisco GmbH,C=DE"
	}</pre>	

	subjectDN	certificate subject DN
Command-line Example	curl -X DELETE -d '{"subjectDn": "CN=emaid002,DC=MO,OU=autocrypt,O=verisco GmbH,C=DE"}' -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118//api/v2/mo/contract"	

Response on Success	<pre>{
	"status": "success",
	"message": "",
	"data": {
	"resultCode": "4000"|"4002"|"4003",
	"message": "revoke result message",
	"serial": "1000000000086"
	}</pre>	

	resultCode	4000 : success
		4002 : already revoked
		4003 : not existing contract
	message	revocation result message
	serial	certificate serial number

Response on Failure	<pre>{
	"status": "error",
	"message": Exception Message,
	"data": ""
	}</pre>	

3.3.4. Verify Contract Certificate for PaymentDetailReq/Res

Verify Contract Certificate for PaymentDetailReq/Res		
Method	**POST**	
URL	/api/v2/mo/paymentDetail	

Request Form	{
	"leaf": "MII…",
	"subca2": "MII…",
	"subca1": "Mll…",
	}	

	leaf	Contract certificate (base64 encoded)
	subca2	Mo SubCA2 certificate (base64 encoded)
	subca1	Mo SubCA1 certificate (base64 encoded)
Command-line Example	curl -X POST -d '{"leaf": "MII…", "subca2": "MII…", "subca1": "Mll…"}' -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118//api/v2/mo/paymentDetail"	

Response on Success	{
	"status": "success",
	"message": "",
	"data": {
	"resCode": "OK"
	}
	}	

	resCode	OK | FAILED_CertChainError | FAILED_CertificateExpired

Response on Failure	{
	"status": "error",
	"message": Exception Message,
	"data": ""
	}	

3.4.CCP

3.4.1. Download CertificateInstallationRes by CertificateInstallationReq

Download CertificateInstallationRes by CertificateInstallationReq		
Method	**POST**	
URL	/api/v2/ccp/certificateInstallation	

Request Form	{
	"v2gMessage": "gJhwGYRUStDo6O…QBWlkNAM2t6+ZE..."
	}	

	v2gMessage	CertificateInstallationReq message (base64 encoded)

Command-line Example	curl -X POST -d '{"v2gMessage": "*gJhwGYRUStDo6O…QBWlkNAM2t6+ZE...*"}' -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118//api/v2/ccp/certificateInstallation"
Response on Success	{ "status": "success", "message": "", "data": { "v2gMessage": "gJhwGYRUStDo6O…QBWlkNAM2t6+ZE..." } }
	v2gMessage: CertificateInstallationRes message (base64 encoded)
Response on Failure	{ "status": "error", "message": Exception Message, "data": "" }

3.4.2. Download CertificateUpdateRes by CertificateUpdateReq

Download CertificateUpdateRes by CertificateUpdateReq	
Method	**POST**
URL	/api/v2/ccp/certificateUpdate
Request Form	{ "v2gMessage":"gJhwGYRUStDo6O…QBWlkNAM2t6+ZE..." }
	v2gMessage: CertificateUpdateReq message (exi and base64 encoded)
Command-line Example	curl -X POST -d '{"v2gMessage": " gJhwGYRUStDo6O…QBWlkNAM2t6+ZE..."}' -H "Accept: application/json" -H "Content-Type: application/json" "http://v2gtest.autocrypt.io:15118/api/v2/ccp/certificateUpdate"
Response on Success	{ "status": "success", "message": "", "data": { "v2gMessage": "gJhwGYRUStDo6O…QBWlkNAM2t6+ZE..." } }
	v2gMessage: CertificateUpdateRes message (base64 encoded)
Response on Failure	{ "status": "error", "message": Exception Message, "data": "" }