CHANGELOG

mercoledì 8 novembre 2017

Versione 1.1

- Versione iniziale

venerdì 10 novembre 2017 11:34:30

Versione 1.2

- Modificata codifica della password in invio
- Aggiunta tabella errori di ritorno delle funzioni

martedì 14 novembre 2017 16:58:28

Versione 1.3

- Aggiunto comando REST GET (*getusertotalfiles*)
- Aggiunto comando REST GET (*downloadfile*)
- Aggiornata tabella errori numerici

lunedì 09 Gennaio 2018

Versione 1.4

- Aggiunto comando di ricerca (*getusersearchfile*)
- Modifiche ai comandi downloadfile e uploadfile

Venerdì 12 Gennaio 2018

Versione 1.5

- Aggiunto comando che restituisce il numero totale di file
- Aggiornata tabella errori numerici per i nuovi comandi aggiunti

Mercoledì 31 Gennaio 2018

Versione 1.6

- Modificato comando (*getuserlicense*)

Versione 1.7

Lunedì 8 Aprile 2018

- Modificati errori di ritorno comando (*getuserlicense*)

Versione 1.8

Mercoledì 4 Luglio 2018

- Aggiunto comando (*getregisternewuser*)
- Aggiornata tabella errori numerici per i nuovi comandi aggiunti

Versione 1.9

14/09/18

- Modificati errori di ritorno comando (*postuploadsinglefile*)
- Aggiunto comando (*getfactomdownload*)
- Correzioni errori di digitazione

Versione 1.10

- Aggiunta modifiche nuovo comando (*postuploadsinglefilev2*)
- Aggiunta modifiche nuovo comando (*getsigneruser*)
- Aggiunta modifiche nuovo comando (*getusersearchfilesdate*)
- Aggiunto Tipo Utente 4 = Enterprise

11/10/18

Versione 1.11

- Modificata chiamata (*getsigneruser*) in (*getdocumentinfo*)

16/10/18

- Aggiornate informazioni su (*getdocumentinfo*)
- Modificati errori di ritorno comando (*getdocumentinfo*)

30/10/18

- Aggiornamento chiamata (*getuserfilesfromdates*)
- Aggiornamento codici errore di ritorno

31/10/18

- Aggiornamento chiamata (*getuserfilesfromdates*)
- Aggiornamento codici errore di ritorno
- Aggiunta chiamata (*getusertotalfilesfromdates*)

Version 1.12

28/11/18

- Aggiornamento chiamata (*getuserfilesfromdates*)
- Aggiornamento codici errore di ritorno
- Aggiunta chiamata (*getusertotalfilesfromdates*)

Version 1.12.1

12/12/18

- Correzione ai parametri della chiamata (*getusertotalfilesfromdates*)

Version 1.12.2

- Aggiunta specifiche per invio dai firmatario

09/01/19

Versione 1.13

- Modifica e correzione per invio dei firmatari

30/01/19

Versione 1.14

- Modifica e correzione dei dati inviati alla chiamata per retrocompatibilità (*getdocumentinfo*)

07/02/19

Versione 1.15

- Aggiunta chiamata caricamento V3 con aggiunta campo SHA256PADES:
    - FileNameSHA256PADES = SHA256 del file PADES firmato e successivamente codificato

08/03/21

Versione 1.16

- Aggiunta chiamata GET che ritorna la versione del Server Lambda Service (*version*)
- Invio dei soli metadati del file caricato sul Lambda Service (*postuploadfilev4*)
- Invio tramite POST di un caricamento UPLOAD (*postuploadfilev6*)

Lista chiamate RESTful verso Server Lambda - Versione 1.16

Richiedi versione delle API attuali sul Lambda Service

https://services.gt50.org/lambdaclient/version

Risultato:

```
{
    "message":"LAMBDA SERVICE API v2.0.0",
    "http_code":201,
    "error":false
}
```

TIPO CHIAMATA HTTP/S: *GET*

ESEMPIO:

Richiesta: https://services.gt50.org/lambdaclient/version

--

Richiedi il token di utente Premium con OTP:

https://services.gt50.org/lambdaclient/getregisteruser/:userid/:password/:otp

- :userid = Nome utente registrato sul portale https://services.gt50.org/
- :password = Password (Codificata in HEX, vedi NOTE)
- :otp = Numero OTP

Risultato:

```
{
    "error": false,
    "http_code": 201,
    "TOKEN": "Cbmyey75ZA,1517232768,f5829aa092cc47bb372ffcbc06bd97b7df76bb26"
}
```

TIPO CHIAMATA HTTP/S: *GET*

ESEMPIO:

Richiesta: https://services.gt50.org/lambdaclient/getregisteruser/enrico/2122C2A32425262F28293D3F5E23/550301

--

Richiedi il token di utente DEMO senza OTP:

https://services.gt50.org/lambdaclient/getregisteruserpassword/:userid/:password

- :userid = Nome utente registrato sul portale https://services.gt50.org/
- :password = Password (Codificata in HEX, vedi NOTE)

TIPO CHIAMATA HTTP/S: *GET*

Risultato:

```
{
    "error": false,
    "http_code": 201,
    "TOKEN": "ix4ZlmVCV9,1517830029,a5dca3388f89c6f234dac9c5ce2ae89ef2200ad9"
}
```

ESEMPIO: Richiesta: https://services.gt50.org/lambdaclient/getregisteruserpassword/enrico3/2122C2A32425262F28293D3F5E23

--

Chiedi informazioni dell'utente registrato:

https://services.gt50.org/lambdaclient/getuserlicense/:userid/:token

- :userid = Nome utente registrato sul portale https://services.gt50.org/
- :token = Token valido rilasciato dalle precedenti chiamate

Risultato:

```
{
    "error": false,
    "http_code": 201,
    "STATUS": "VALID: 1000:4:500:2.03 MB:0"
}
```

TIPO CHIAMATA HTTP/S: *GET*

Richiesta: https://services.gt50.org/lambdaclient/getuserlicense/enrico/MBah3CrHgu,1517825614,77d2429e05cb05f3bb34cee7909a0d6cac68aa35

Il campo STATUS:

Ritorna: VALID se licenza valida, NOT VALID se licenza scaduta. I campi sono separati da ":"

1. Riporta il numero di caricamenti possibili acquistati
2. Riporta il numero di caricamenti effettuati
3. Spazio acquisito in MB per i documenti da caricare
4. Spazio utilizzato in MB dai documeti caricati
5. Uso dell'OTP: 0 = No OTP, 1 = Yes OTP
6. Tipo di licenza: 1 = Demo 1 mese, 2 = Demo 2 mesi, 3 = Utente pagante 12 mesi, 4 = Utente Enterprise
7. Data scadenza licenza: Data ed ora in formato stringa (AAAAMMGGHHMMSS) di 14 caratteri numerici

--

Lista files per utente:

https://services.gt50.org/lambdaclient/getuserfiles/:userid/:token/:star/:stop

- :userid = Nome utente registrato sul portale https://services.gt50.org/
- :token = Token valido rilasciato dalle precedenti chiamate
- :start = Numero di lista da cui partire
- :stop = Numero di lista finale

Risultato:


```
{
    "error": false,
    "http_code": 201,
    "total_files": "5",
    "arrayFile": [
        {
            "IDFile": "1",
            "FileName": "001File.pdf",
            "SHA256":
"32B5FD70C0F38E3E85457291228C19F07DAAE1C776901DF7886C2A37CBE0699C",
            "DateTime": "2017-10-31 14:39:28"
        },
        {
            "IDFile": "2",
            "FileName": "004File.PDF",
            "SHA256":
"33020CC3071C97C1543C296C78A2573ACD2980E171300B4AB17B09B1089E9EC9",
            "DateTime": "2017-10-31 15:06:52"
        },
        {
            "IDFile": "3",
            "FileName": "002File.PDF",
```

```
            "SHA256":
"426D717DC162383AB2AF99B50FB684DB46561218B28FF463C9D070480F2C7761",
            "DateTime": "2017-11-03 12:51:49"
        },
        {
            "IDFile": "4",
            "FileName": "005File.PDF",
            "SHA256":
"E0A79A1B692929AD1F613F5370B7AFCF2CF0DC1F5FFD64B2DD9DB762925A3959",
            "DateTime": "2017-11-03 12:56:55"
        },
        {
            "IDFile": "5",
            "FileName": "FileText.txt",
            "SHA256":
"148EE834A6ADE2D8414A2B83EAFAB49044811EBAC1AC02ECDC33783CFD9E5F7E",
            "DateTime": "2017-11-09 09:42:35"
        }
    ],
    "RESPONSE": "OK"
}
```


TIPO CHIAMATA HTTP/S: *GET*

Richiesta: https://services.gt50.org/lambdaclient/getuserfiles/enrico/ix4ZlmVCV9,1517830029,a5dca3388f89c6f234dac9c5ce2ae89ef2200ad9/0/5

--

Lista files per utente in un intervallo di tempo:

https://services.gt50.org/lambdaclient/getuserfilesfromdates/:userid/:token/:search/:start_date/:stop_date/:star/:stop

- :userid = Nome utente registrato sul portale https://services.gt50.org/
- :token = Token valido rilasciato dalle precedenti chiamate
- :search = stringa di ricerca codificata in base64 (La ricerca avviene per il campo FileName)
- :start_date = stringa di ricerca iniziale in formato AAAAMMGG
- :stop_date = stringa di ricerca finale in formato AAAAMMGG
- :start = Numero di lista da cui partire
- :stop = Numero di lista finale

TIPO CHIAMATA HTTP/S: *GET*

Risultato:


```
{
    "error": false,
    "http_code": 201,
    "total_files": "6",
    "arrayFile": [
```

```
        {
            "IDFile": "953",
            "FileName": "file.txt",
            "SHA256":
"4E26E71D7451E19D3A37ADDC9A318C97370E35727272E819F4347BA00C4A8FF8",
            "DateTime": "2018-10-11 15:50:26",
            "DateTimeUpload": "20181011155026"
        },
        {
            "IDFile": "952",
            "FileName": "file.txt",
            "SHA256":
"4E26E71D7451E19D3A37ADDC9A318C97370E35727272E819F4347BA00C4A8FF8",
            "DateTime": "2018-10-11 15:43:12",
            "DateTimeUpload": "20181011154312"
        },
        {
            "IDFile": "951",
            "FileName": "file.txt",
            "SHA256":
"4E26E71D7451E19D3A37ADDC9A318C97370E35727272E819F4347BA00C4A8FF8",
            "DateTime": "2018-10-11 15:29:10",
            "DateTimeUpload": "20181011152910"
        },
        {
            "IDFile": "950",
            "FileName": "file.txt",
            "SHA256":
"4E26E71D7451E19D3A37ADDC9A318C97370E35727272E819F4347BA00C4A8FF8",
            "DateTime": "2018-10-11 15:22:45",
            "DateTimeUpload": null
        },
        {
            "IDFile": "949",
            "FileName": "file.txt",
            "SHA256":
"4E26E71D7451E19D3A37ADDC9A318C97370E35727272E819F4347BA00C4A8FF8",
            "DateTime": "2018-10-11 10:56:47",
            "DateTimeUpload": "20181011105647"
        },
        {
            "IDFile": "948",
            "FileName": "file.txt",
            "SHA256":
"4E26E71D7451E19D3A37ADDC9A318C97370E35727272E819F4347BA00C4A8FF8",
            "DateTime": "2018-10-08 12:58:59",
            "DateTimeUpload": "20181008125859"
        }
    ],
```

```
    "RESPONSE": "OK"
}
```

--

Totale files per utente in un intervallo di tempo:

https://services.gt50.org/lambdaclient/getusertotalfilesfromdates/:userid/:search/:token/:start_date/:stop_date

- :userid = Nome utente registrato sul portale https://services.gt50.org/
- :search = stringa di ricerca codificata in base64 (La ricerca avviene per il campo FileName)
- :token = Token valido rilasciato dalle precedenti chiamate
- :start_date = stringa di ricerca iniziale in formato AAAAMMGG
- :stop_date = stringa di ricerca finale in formato AAAAMMGG

TIPO CHIAMATA HTTP/S: *GET*

--

Cerca files per utente data una stringa:

https://services.gt50.org/lambdaclient/getusersearchfiles/:userid/:token/:search_string

- :userid = Nome utente registrato sul portale https://services.gt50.org/
- :token = Token valido rilasciato dalle precedenti chiamate
- :search_string = stringa di ricerca codificata in base64 (La ricerca avviene per il campo FileName)

Risultato:

```
{
    "error": false,
    "http_code": 201,
    "total_files": "5",
    "arrayFile": [
        {
            "IDFile": "1",
            "FileName": "001File.pdf",
            "SHA256":
"32B5FD70C0F38E3E85457291228C19F07DAAE1C776901DF7886C2A37CBE0699C",
            "DateTime": "2017-10-31 14:39:28"
        },
        {
            "IDFile": "2",
            "FileName": "001File.PDF",
            "SHA256":
"33020CC3071C97C1543C296C78A2573ACD2980E171300B4AB17B09B1089E9EC9",
            "DateTime": "2017-10-31 15:06:52"
        },
        {
            "IDFile": "3",
            "FileName": "001File.PDF",
```

```
            "SHA256":
"426D717DC162383AB2AF99B50FB684DB46561218B28FF463C9D070480F2C7761",
            "DateTime": "2017-11-03 12:51:49"
        },
        {
            "IDFile": "4",
            "FileName": "001File.PDF",
            "SHA256":
"E0A79A1B692929AD1F613F5370B7AFCF2CF0DC1F5FFD64B2DD9DB762925A3959",
            "DateTime": "2017-11-03 12:56:55"
        },
        {
            "IDFile": "5",
            "FileName": "001File.PDF",
            "SHA256":
"148EE834A6ADE2D8414A2B83EAFAB49044811EBAC1AC02ECDC33783CFD9E5F7E",
            "DateTime": "2017-11-09 09:42:35"
        }
    ],
    "RESPONSE": "OK"
}
```


TIPO CHIAMATA HTTP/S: *GET*

Richiesta: https://services.gt50.org/lambdaclient/getuserfiles/enrico/ix4ZlmVCV9,1517830029,a5dca3388f89c6f234dac9c5ce2ae89ef2200ad9/aWwgZmlsZSBkaSByaWNlcmNhIMOoIHF1ZXN0byAn

--

Richiesta numero di files per singolo utente:

https://services.gt50.org/lambdaclient/getusertotalfiles/:user/:token

- :userid = Nome utente registrato sul portale https://services.gt50.org/
- :token = Token valido rilasciato dalle precedenti chiamate

Risultato:


```
{
    "error": false,
    "http_code": 201,
    "TOTAL_FILES": 54
}
```


TIPO CHIAMATA HTTP/S: *GET*

Richiesta: https://services.gt50.org/lambdaclient/getusertotalfiles/enrico/ix4ZlmVCV9,1517830029,a5dca3388f89c6f234dac9c5ce2ae89ef2200ad9

--

Richiesta numero di files per singolo utente per singola ricerca:

https://services.gt50.org/lambdaclient/getusersearchtotalfiles/:user/:token/:search

- :userid = Nome utente registrato sul portale https://services.gt50.org/
- :token = Token valido rilasciato dalle precedenti chiamate
- :search_string = stringa di ricerca codificata in base64 (La ricerca avviene per il campo FileName)

Risultato:

```
{
    "error": false,
    "http_code": 201,
    "TOTAL_FILES": 5
}
```

TIPO CHIAMATA HTTP/S: *GET*

Richiesta: https://services.gt50.org/lambdaclient/ggetusersearchtotalfiles/enrico/ix4ZlmVCV9,1517830029,a5dca3388f89c6f234dac9c5ce2ae89ef2200ad9/RmlsZQ===

--

Caricamento file su Lambda Service V3:

https://services.gt50.org/lambdaclient/postuploadsinglefilev3

Esempio di caricamento:

```
{
  "TokenID": "YJYlwB64a6,1506499124,1e53b9fd7595065c4d01acd7e4d0cbca00137a11",
  "UserID": "MARCO",
  "FileName": "FileText.txt",
  "FileBase64": "UXVlc3RvIMOoIHVuIGZpZSBkaSB0ZXN0IHBlciBpbCBiYXNlNjQ=",
  "FileNameSHA256": "148ee834a6ade2d8414a2b83eafab49044811ebac1ac02ecdc33783cfd9e5f7e",
  "FileNameSHA256PADES":
"5d79a5149368e7dd368e69e87227fac6c5e6abfad6d52b20ffd531ee36107805",
  "FileNameDateTime": "20171222234545",
  "SignerUser": "VEVTVCBGSVJNQSBTSUdORVI="
}
```

- TokenID = Token valido rilasciato dalle precedenti chiamate
- UserID = Nome utente registrato sul portale https://services.gt50.org/
- FileName = Nome file originario del documento caricato
- FileBase64 = Codifica in BASE64 del file da caricare
- FileNameSHA256 = SHA256 del file originario in formato PDF poi caricato non firmato
- FileNameSHA256PADES = SHA256 del file PADES firmato e successivamente codificato
- FileNameDateTime = Data ed ora in formato stringa (AAAAMMGGHHMMSS) di 14 caratteri numerici
- SignerUser = Codifica in BASE64 della stringa JSON contenente i dati del firmatario cifrati con algoritmo AES256 e con la chiave che è l'hash SHA256 dela chiave casuale presente all'interno del QRCode.

- Base64(aes256(Kdata, JSON(dati)))
- dove Kdata = SHA256(K-qrcode)
  - I dati da inserire devono essere a loro volta in formato JSON.
  - Qualora il relativo contenuto manchi inserire un campo vuoto: ""

```
{
  "Subject": "<Common Name>",
  "Organization": "<Organization>",
  "OrganizationUnit": "<Organization Unit>",
  "Country": "<Country>",
  "SerialNumber": "<Serial Number in HEX Text>",
  "SignDate":"yyyy/mm/dd hh:nn:ss",
  "Valid From": "yyyy/mm/dd hh:nn:ss",
  "Valid To" : "yyyy/mm/dd hh:nn:ss",
  "Issuer" : "<Issuer> ",
  "IssuerCommonName" : "<Issue Common Name>",
  "IssuerCountry" : "<Issuer Country>"
}
```

Esempio reale:

```
{
  "Subject":"Angelucci Marco",
  "Organization":"NON PRESENTE",
  "OrganizationUnit":"",
  "Country":"IT",
  "SerialNumber":"1C6BAED576F2E0FE5765A207E8C690D22",
  "SignDate":"16/11/2018 10:04:43",
  "Valid From":"19/05/2017 23:59:59",
  "Valid To":"18/05/2020 23:59:59",
  "Issuer":"ArubaPEC S.p.A.",
  "IssuerCommonName":"ArubaPEC S.p.A. NG CA 3",
  "IssuerCountry":"IT"
}
```

Risultato:

```
{
    "error": false,
    "http_code": 201,
    "UPLOAD": "OK"
}
```

TIPO CHIAMATA HTTP/S: *POST*

--

Caricamento file su Lambda Service V2:

https://services.gt50.org/lambdaclient/postuploadsinglefilev2

Esempio di caricamento:

```
{
  "TokenID": "YJYlwB64a6,1506499124,1e53b9fd7595065c4d01acd7e4d0cbca00137a11",
  "UserID": "MARCO",
  "FileName": "FileText.txt",
  "FileBase64": "UXVlc3RvIMOoIHVuIGZpZSBkaSB0ZXN0IHBlciBpbCBiYXNlNjQ=",
  "FileNameSHA256": "148ee834a6ade2d8414a2b83eafab49044811ebac1ac02ecdc33783cfd9e5f7e",
  "FileNameSHA256PADES":
"5d79a5149368e7dd368e69e87227fac6c5e6abfad6d52b20ffd531ee36107805",
  "FileNameDateTime": "20171222234545",
  "SignerUser": "VEVTVCBGSVJNQSBTSUdORVI="
}
```

- TokenID = Token valido rilasciato dalle precedenti chiamate
- UserID = Nome utente registrato sul portale https://services.gt50.org/
- FileName = Nome file originario del documento caricato
- FileBase64 = Codifica in BASE64 del file da caricare
- FileNameSHA256 = SHA256 del file originario in formato PDF poi caricato non firmato
- FileNameDateTime = Data ed ora in formato stringa (AAAAMMGGHHMMSS) di 14 caratteri numerici
- SignerUser = Codifica in BASE64 della stringa JSON contenente i dati del firmatario cifrati con algoritmo AES256 e con la chiave che è l'hash SHA256 dela chiave casuale presente all'interno del QRCode.
    - Base64(aes256(Kdata, JSON(dati)))
    - dove Kdata = SHA256(K-qrcode)
        - I dati da inserire devono essere a loro volta in formato JSON.
        - Qualora il relativo contenuto manchi inserire un campo vuoto: ""

```
{
  "Subject": "<Common Name>",
  "Organization": "<Organization>",
  "OrganizationUnit": "<Organization Unit>",
  "Country": "<Country>",
  "SerialNumber": "<Serial Number in HEX Text>",
  "SignDate":"yyyy/mm/dd hh:nn:ss",
  "Valid From": "yyyy/mm/dd hh:nn:ss",
  "Valid To" : "yyyy/mm/dd hh:nn:ss",
  "Issuer" : "<Issuer> ",
  "IssuerCommonName" : "<Issue Common Name>",
  "IssuerCountry" : "<Issuer Country>"
}
```

Esempio reale:

```
{
  "Subject":"Angelucci Marco",
  "Organization":"NON PRESENTE",
  "OrganizationUnit":"",
  "Country":"IT",
  "SerialNumber":"1C6BAED576F2E0FE5765A207E8C690D22",
  "SignDate":"16/11/2018 10:04:43",
  "Valid From":"19/05/2017 23:59:59",
  "Valid To":"18/05/2020 23:59:59",
  "Issuer":"ArubaPEC S.p.A.",
  "IssuerCommonName":"ArubaPEC S.p.A. NG CA 3",
  "IssuerCountry":"IT"
}
```


Risultato:


```
{
    "error": false,
    "http_code": 201,
    "UPLOAD": "OK"
}
```


TIPO CHIAMATA HTTP/S: *POST*

-- **Caricamento file su Lambda Service:**

https://services.gt50.org/lambdaclient/postuploadsinglefile

Esempio di caricamento:


```
{
  "TokenID": "YJYlwB64a6,1506499124,1e53b9fd7595065c4d01acd7e4d0cbca00137a11",
  "UserID": "MARCO",
  "FileName": "FileText.txt",
  "FileBase64": "UXVlc3RvIMOoIHVuIGZpZSBkaSB0ZXN0IHBlciBpbCBiYXNlNjQ=",
  "FileNameSHA256": "148ee834a6ade2d8414a2b83eafab49044811ebac1ac02ecdc33783cfd9e5f7e",
  "FileNameDateTime": "20171222234545"
}
```


- TokenID = Token valido rilasciato dalle precedenti chiamate
- UserID = Nome utente registrato sul portale https://services.gt50.org/
- FileName = Nome file originario del documento caricato
- FileBase64 = Codifica in BASE64 del file da caricare
- FileNameSHA256 = SHA256 del file originario in formato PDF poi caricato non firmato
- FileNameDateTime = Data ed ora in formato stringa (AAAAMMGGHHMMSS) di 14 caratteri numerici

Risultato:

```
{
    "error": false,
    "http_code": 201,
    "UPLOAD": "OK"
}
```

TIPO CHIAMATA HTTP/S: *POST*

--

Caricamento solo dei metadati associati ad un file Lambda su Lambda Store

https://services.gt50.org/lambdaclient/postuploadfilev4/:user/:token/:hashfilesha256/:filename/:filenamedatatime/

- :user = Nome utente registrato sul portale https://services.gt50.org/
- :token = Token valido rilasciato dalle precedenti apposite chiamate
- :hashfilesha256 = SHA256 identificativo del file
- :filename = Nome file del file inviato fino ad un massimo di 256 Byte
- :filenamedatatime = Data ed ora in formato stringa (AAAAMMGGHHMMSS) così come riportata nel QRCode CodApp: 0306 (Si usa lo stesso codice applicazione ma con un diverso numero di campi)

TIPO CHIAMATA HTTP/S: *POST*

Richiesta:

https://services.gt50.org/lambdaclient/postuploadfilev4/MARCODEMO/trKboj6c79,1603727677,4819e5cabd749aa949d9a45105709e1a943d8ed9/700febb601c5ff4af6529a21538b6bb600b473c290a1ca41db62475bfa182722/FileText.txt/20210615140030

Risultato:

```
{
    "error": false,
    "http_code": 201,
    "UPLOAD": "OK"
}
```

ATTENZIONE: Questa chiamata inserisce nello store solo i metadati solitamente associati ad un documento Lambda. Il vero e proprio file binario sarà conservato in formato blob binario codificato su servizio S3 con un'apposita configurazione da concordare con GT50.

Codici di errore uguali a quelli di una chiamate *postuploadsinglefilev3*

--

Caricamento file su Lambda Service tramite singola chiamata POST Form URL Encoded

https://services.gt50.org/lambdaclient/postuploadfilev6

Codifica Header:

Content-Type = application/x-www-form-urlencoded

Dati da inviare nell'array e relativi campi:

- **userid** = Nome utente registrato sul portale https://services.gt50.org/
- **tokenid** = Token valido rilasciato dalle precedenti apposite chiamate
- **hashfilesha256** = SHA256 identificativo del file
- **filename** = Nome file del file inviato fino ad un massimo di 256 Byte
- **filenamedate** = Data ed ora in formato stringa (AAAAMMGGHHMMSS) così come riportata nel QRCode CodApp: 0306 (Si usa lo stesso codice applicazione ma con un diverso numero di campi)
- **filebase64** = Codifica in BASE64 del file da caricare

TIPO CHIAMATA HTTP/S: *POST*

Risultato:

```
{
    "error": false,
    "http_code": 201,
    "UPLOAD": "OK"
}
```

Codici di errore uguali a quelli di una chiamate *postuploadsinglefilev3*

--

Scaricare un file da Lambda Service

https://services.gt50.org/lambdaclient/downloadfile/:hashfilesha256/:filenamedatatime

- :hashfilesha256 = SHA256 identificativo del file
- :filenamedatatime = Data ed ora in formato stringa (AAAAMMGGHHMMSS) così come riportata nel QRCode CodApp: 0306 (Si usa lo stesso codice applicazione ma con un diverso numero di campi)

TIPO CHIAMATA HTTP/S: *GET*

Richiesta:

https://services.gt50.org/lambdaclient/downloadfile/D37B9395C2BAF168F977CE9FF9EC007D7270FC84CBF1549324BFC8DFC34333A9/20170475112233

--

Scaricare informazioni da Lambda Service sul firmatario del documento PDF caricato

https://services.gt50.org/lambdaclient/getdocumentinfo/:hashfilesha256(/:filenamedatatime)

- :hashfilesha256 = SHA256 identificativo del file
- :filenamedatatime = Data ed ora in formato stringa (AAAAMMGGHHMMSS) così come riportata nel QRCode CodApp: 0306 (Si usa lo stesso codice applicazione ma con un diverso numero di campi). Il campo è opzionale, non indicandolo si ottiene la ricerca di informazioni per documenti V1 che non avevano il campo :filenamedatatime

TIPO CHIAMATA HTTP/S: *GET*

Richiesta:

https://services.gt50.org/lambdaclient/getsigneruser/D37B9395C2BAF168F977CE9FF9EC007D7270FC84CBF1549324BFC8DFC34333A9/20170475112233

(Questa chiamata è pubblica e NON necessita di alcun token od utente)

Risultato:

Viene restituito lo stato del documento, tutte le informazioni sulla firma del documento e se disponibili anche tutte le informazioni relative a FACTOM.


```
{
    "error": false,
    "http_code": 201,
    "STATUS": 0,
    "MESSAGE": "Il documento è stato sospeso per motivi logistici",
    "SIGNERINFO": "VEVTVCBGSVJNQSBTSUdORVI=",
    "FACTOM": "a:3:
{s:7:\"chainid\";s:64:\"76d9a11cb21ca4a24385b255b1f70bb147d5ed2a4d7a50123b558639533d042
2\";s:7:\"content\";s:214:\"7b224861736846696c65223a22344532364537314437343531453139443
341333741444443394133313843393733373045333537323732373245383139463433343742413030433441
38464638222c224461746554696d65223a22323031383039313231363333233227d\";s:6:\"extids\";a
:2:
{i:0;s:8:\"47543530\";i:1;s:512:\"8935c091b75a50d2f0ef2a8fca9d2c05705617381e765fe27eadc
d3195f2accf187beb39f4e61400db6a5de22db2289d0702ca4d6093f69d3566a351f9659aafea9ef9c0cd3c
bab04047967ad287bb53998ec49958b9cbc63164e30794be3b15f8de8ed3d7156914f3642213910d7a98993
da5115d792904bbf55c766dd9f53cda03696d1716c24c183594ddfd3df03575cfda3e6860bbc755f38aa18a
d2c9bd72b0a5d3f89d375f1d76d815167be2824bb8de9ee449b88f8a7dba885d9c7a2253c97d0ff25db12f3
4e04bb286d50abb3d7dcf64e74deb24093cf65fb784c21435f8cdf93a41e8d2b03e6d31e432b6385da90ffc
86c5a88a34aeb5aa8c95ac5b\";}}
}
```


Questa chiamata deve essere fatta dal client precedentemente a: (*downloadfile*)

Tutti che non hanno dati ritorneranno stringhe vuote: "".

- "STATUS": 0 = Presente, 1 = Cancellato, 2 = Sospeso.
- "SIGNERINFO": Base64 dei dati del firmatario cifrato con HASH256 della chiave casuale presente nel QRCode (Da mostrare a video in ogni caso).
- "MESSAGE": Messaggio da mostrare a video solo nel caso della STATUS 1 o 2, altrimenti sara una stringa vuota "".
- "FACTOM": Se "" dati non presenti.

--

Scaricare informazioni da Lambda Service della BlockChain FACTOM

https://services.gt50.org/lambdaclient/getfactomdownload/:hashfilesha256/:filenamedatatime

- :hashfilesha256 = SHA256 identificativo del file
- :filenamedatatime = Data ed ora in formato stringa (AAAAMMGGHHMMSS) così come riportata nel QRCode CodApp: 0307 (Si usa lo stesso codice applicazione ma con un diverso numero di campi)

TIPO CHIAMATA HTTP/S: *GET*

Richiesta:

https://services.gt50.org/lambdaclient/getfactomdownload/D37B9395C2BAF168F977CE9FF9EC007D7270FC84CBF1549324BFC8DFC34333A9/20170475112233

(Questa chiamata è pubblica e NON necessita di alcun token od utente)

oppure in test:

http://services-clone.gt50.org/lambdaclient/getfactomdownload/4E26E71D7451E19D3A37ADDC9A318C97370E35727272E819F4347BA00C4A8FF8/20180912163323

(Questa chiamata è pubblica e NON necessita di alcun token od utente)

Risultato:


```
{
    "error": false,
    "http_code": 201,
    "RESULT": "a:3:
{s:7:\"chainid\";s:64:\"76d9a11cb21ca4a24385b255b1f70bb147d5ed2a4d7a50123b558639533d042
2\";s:7:\"content\";s:214:\"7b224861736846696c65223a22344532364537314437343531453139443
34133374144444339413331384339373337304533353732373237324538313946343334374241303043344138464638222c224461746554696d65223a22323031383039313231363333233227d\";s:6:\"extids\";a
:2:
{i:0;s:8:\"47543530\";i:1;s:512:\"8935c091b75a50d2f0ef2a8fca9d2c05705617381e765fe27eadcd3195f2accf187beb39f4e61400db6a5de22db2289d0702ca4d6093f69d3566a351f9659aafea9ef9c0cd3cbab04047967ad287bb53998ec49958b9cbc63164e30794be3b15f8de8ed3d7156914f3642213910d7a98993da5115d792904bbf55c766dd9f53cda03696d1716c24c183594ddfd3df03575cfda3e6860bbc755f38aa18ad2c9bd72b0a5d3f89d375f1d76d815167be2824bb8de9ee449b88f8a7dba885d9c7a2253c97d0ff25db12f34e04bb286d50abb3d7dcf64e74deb24093cf65fb784c21435f8cdf93a41e8d2b03e6d31e432b6385da90ffc86c5a88a34aeb5aa8c95ac5b\";}}"
}
```


--

Creazione Nuovo Utente su Lambda Service:
http://services.gt50.org/lambdaclient/getregisternewuser/:username/:password/:email:/:hashcheck

- :username = Nome Utente scelto e da verificare
- :password = Password (Codificata in HEX, vedi NOTE)
- :email = EMail scelta a cui verrà inviato il messaggio di verifica
- :hashckeck = Hash codificato SHA256 della DATA al momento dell'invio (YYYYMMDD) + be5bb14c-c273-433b-b876-062651480b58

TIPO CHIAMATA HTTP/S: *GET*

Risultato:

```
{
    "error": false,
    "http_code": 201,
    "STATUS": "CREATED"
}
```

Esempio di richiesta:

http://services.gt50.org/lambdaclient/getregisternewuser/enrico11/7465737431323334/hudibun@trimsj.com/E0ACE2265CD3E8C8D35BA5ECEDE3F205E919E1E053DF47F84E7AEF567C01D470

--

NOTE:

Esempio codifica HEX in PHP:

```
<?php
function String2Hex($string){
    $hex='';
    for ($i=0; $i < strlen($string); $i++){
        $hex .= dechex(ord($string[$i]));
    }
    return $hex;
}


function Hex2String($hex){
    $string='';
    for ($i=0; $i < strlen($hex)-1; $i+=2){
        $string .= chr(hexdec($hex[$i].$hex[$i+1]));
    }
    return $string;
}
```

--

Lista errori di ritorno singole chiamate:

Funzione	Codice Errore	Spiegazione
getregisteruser	9	Error: Password inserted NOT valid
getregisteruser	10	Error: OTP inserted NOT valid.
getregisteruser	11	Error: UserID inserted NOT valid.
getregisteruser	12	Error: Invalid data entered.

getuserlicense	20	Error: UserID inserted NOT valid.
getuserlicense	21	STATUS: LICENSE IS VALID BUT NOT TOKEN.
getuserlicense	22	Error: Invalid data entered.
getuserlicense	23	STATUS: LICENSE IS NOT VALID AND TOKEN NOT VALID.
getuserfiles	30	Error: UserID inserted NOT valid.
getuserfiles	31	STATUS: NOT VALID
getuserfiles	32	Error: Invalid data entered.
postuploadsinglefile	40	Error: File NOT uploaded.
postuploadsinglefile	41	Error: No data inserted in DB.
postuploadsinglefile	42	Error: Invalid UserID entered or UserID NOT enabled.
postuploadsinglefile	43	STATUS: NOT VALID
postuploadsinglefile	44	Error: Invalid data entered.
postuploadsinglefile	45	Error: Wrong JSON format
postuploadsinglefile	46	Error: FACTOM ERROR, file not uploaded in BlockChain.
downloadfile	55	Error: File not present
downloadfile	56	Error: Wrong file format request
getusertotalfiles	60	Error: UserID inserted NOT valid.
getusertotalfiles	61	STATUS: NOT VALID
getusertotalfiles	62	Error: Invalid data entered.
getusersearchfiles	70	Error: UserID inserted NOT valid.
getusersearchfiles	71	STATUS: NOT VALID
getusersearchfiles	72	Error: Invalid data entered.

getusersearchtotalfiles	80	Error: UserID inserted NOT valid.
getusersearchtotalfiles	81	STATUS: NOT VALID
getusersearchtotalfiles	82	Error: Invalid data entered.
getregisternewuser	92	Error: Invalid data entered.
getregisternewuser	93	Error: Username/email address is already in use by others.
getregisternewuser	94	Error: Username/email NOT inserted.
getregisternewuser	95	Error: in the EMAIL send function.
getregisternewuser	96	Error: check HASH, it is incorrect.
getfactomdownload	100	Error: FACTOM return an error.
getfactomdownload	101	Error: hashfile and/or filedatetime NOT in DB.
getfactomdownload	102	Error: Invalid data entered.
getdocumentinfo	110	Error: FACTOM return an error.
getdocumentinfo	111	Error: hashfile and/or filedatetime NOT in DB.
getdocumentinfo	112	Error: Invalid data entered.
getuserfilesfromdates	120	Error: UserID inserted NOT valid.
getuserfilesfromdates	121	STATUS: NOT VALID
getuserfilesfromdates	122	Error: Invalid data entered.
getusertotalfilesfromdates	130	Error: UserID inserted NOT valid.
getusertotalfilesfromdates	131	STATUS: NOT VALID
Getusertotalfilesfromdatess	132	Error: Invalid data entered.