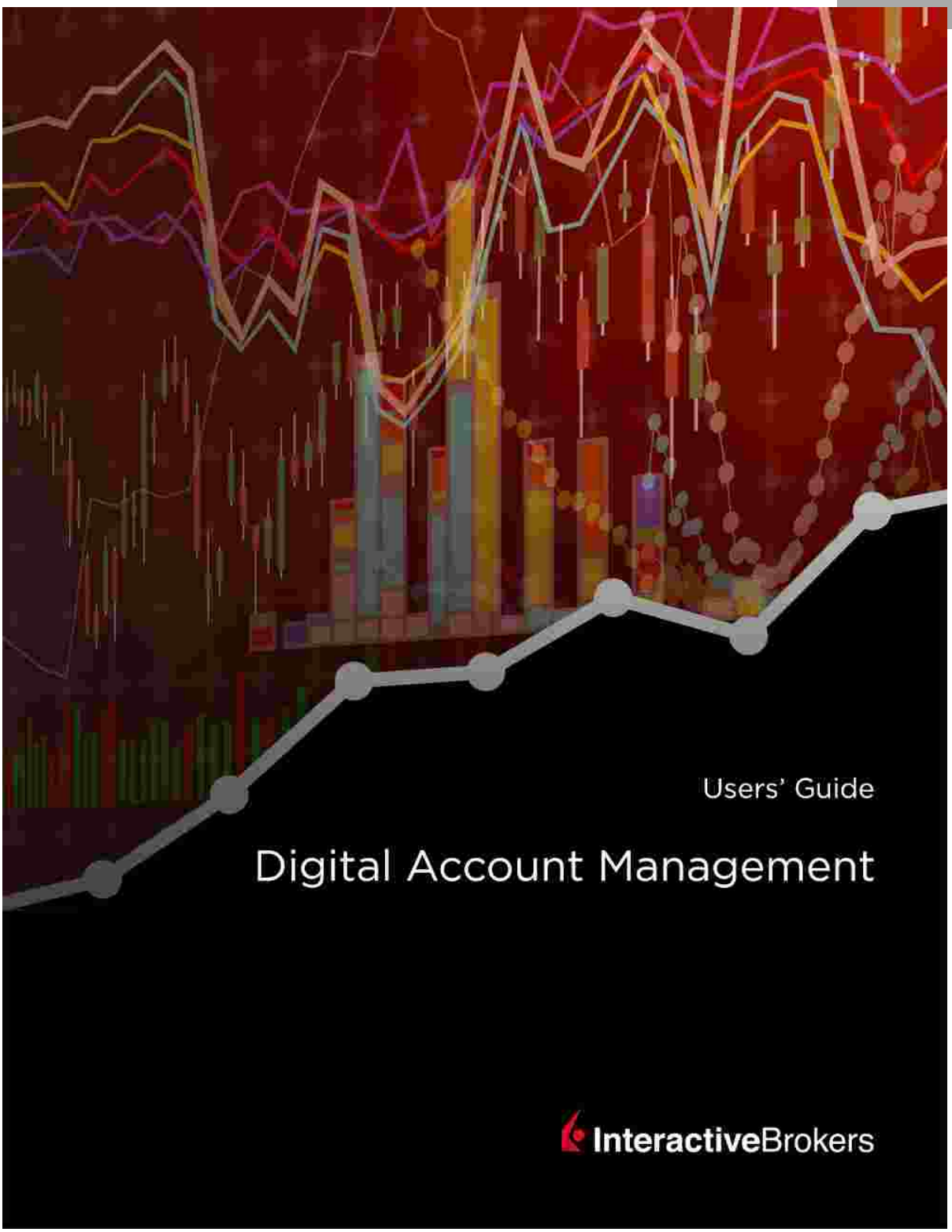
Users' Guide
Digital Account Management
InteractiveBrokers

Digital Account Management Users' Guide

December 2018 - Version 3.0

Any symbols displayed within these pages are for illustrative purposes only, and are not intended to portray any recommendation.

Contents

Contents **3**

Overview **4**

Web Service Implementation Details **5**

1. /create 5

2. /update 6

3. /validateXML 7

4. /getStatus 9

5. /getXMLSchema 10

6. /submitDocuments 11

7. /getAccountStatus 12

8. /getPendingTasks 13

Generate PGP Keypair **16**

Request CSID **18**

Encrypt and Encode File **19**

Decode and Decrypt File **20**

Overview

Digital AM Web Service provides a mechanism by which clients can create new accounts and update existing accounts with Interactive Brokers in real time. It alleviates the need to FTP XML files to Interactive Brokers for account creation and account management.

Digital AM Web Service will be available on the Web at the following URLs:

Production Instance

https://[domain]/ws/eca, where [domain] is IB domain serving different regions.

The domains are: www.interactivebrokers.com, www.interactivebrokers.com.hk and www.ibkr.com.cn.

Quality Instance

https://qa.interactivebrokers.com/ws/eca

Web Service Implementation Details

The discussion below provides an overview of service calls, including request parameters that need to be passed to the service and response from the call.

1. /create

Service to create new accounts.

HTTP Request Type POST

Request Parameters in JSON Format:

```
{
 "CSID":"<client-service-id>",
 "payload":"<encrypted and signed archive file, encoded in base64>"
}
```

Service accepts the following parameters

- CSID Unique identifier issued by Interactive Brokers.
- payload Encrypted and signed archive file, encoded in base64. The archive must have an XML document with information about new accounts. The archive can also have forms and agreements in PDF format. Please note that forms and agreements are not mandatory.

Service Response in JSON Format:

```
{
 "timestamp": "<timestamp>",
 "isProcessed":"<true|false>",
```

"batchId":"<batchId>",

"fileData":" {

"name":"<response-file-name>",

"data":"<encrypted response XML, encoded in base64>"

}

}

Service will return isProcessed flag to indicate whether the request was processed in real time.

When request is processed in real time, fileData section will have account credentials in an encrypted XML file, encoded in base64. In case request is not processed in real time, batchId can be used to query the status at a later date.

Note:

- Please review the section, Generate PGP Key Pair, for details on how to generate PGP Key Pair.
- Please review the section, Request CSID, for details on how to Request CSID from Interactive Brokers.
- Please review the section, Encrypt, Sign, and Encode Binary Files, for details on how to Encrypt, Sign, and Encode Binary Files.
- Please review the section, Decode and Decrypt Binary Files, for details on how to Decode, and Decrypt Binary Files.

2. /update

Service to update existing accounts

HTTP Request Type POST

Request Parameters in JSON Format:

```
{
 "CSID":"<client-service-id>",
 "payload":"<encrypted and signed archive file, encoded in base64>"
}
```

Service accepts the following parameters

- CSID Unique identifier issued by Interactive Brokers.
- payload Encrypted and signed archive file, encoded in base64.

Service Response in JSON Format:

```
{
 "timestamp": "<timestamp>",
 "fileData":" {
  "name":"<response-file-name>",
  "data":"<encrypted response XML, encoded in base64>"
 }
}
```

Service will return an encrypted XML file, encoded in base64.

The XML file will contain results of the update request.

3. /validateXML

Service to validate XML documents

HTTP Request Type POST

Request Parameters in JSON Format:

```
{
 "CSID":"<client-service-id>",
 "payload":"<encrypted and signed XML document, encoded in base64>"
}
```

Service accepts the following parameters

- CSID Unique identifier issued by Interactive Brokers.
- payload Encrypted and signed XML document, encoded in base64.

Service Response in JSON Format:

```
{
 "timestamp": "<timestamp>",
 "isValid":"<true|false>",
 "errors": [
  {"error":"<error-message>"},
  {"error":"<error-message>"}
 ]
}
```

Service will return isValid flag to indicate whether XML document is valid or not.

In case XML document is not valid, errors section will provide details about the errors.

4. /getStatus

Service to get status of a request

HTTP Request Type GET

Request Parameters Name/Value Pairs

CSID=<client-service-id>&batchId=<batch-id>

Service accepts the following parameters

- CSID Unique identifier issued by Interactive Brokers.
- batchId Batch Id issued by IB for a given request.

Service Response in JSON Format:

```
{
 "timestamp": "<timestamp>",
 "dateSubmitted","<date-submitted>",
 "isProcessed":"<true|false>",
 "fileData":" {
  "name":"<response-file-name>",
  "data":"<encrypted response XML, encoded in base64>"
 }
}
```

Service will return isProcessed flag to indicate whether the request has been processed or not.

In case request has been processed, fileData section will provide details in an encrypted XML file, encoded in base64.

5. /getXMLSchema

Service to get XML schema for creating or updating accounts

HTTP Request Type GET

Request Parameters Name/Value Pairs

CSID=<client-service-id>&type=<type>

Service accepts the following parameters

- CSID Unique identifier issued by Interactive Brokers.
- type Can be either
- create (to get XML Schema to creating new accounts) or,
- update (to get XML Schema to update existing accounts)

Service Response in JSON Format:

```
{
 "timestamp": "<timestamp>",
 "fileData":" {
  "name":"<response-file-name>",
  "data":"<encrypted XML Schema, encoded in base64>"
 }
}
```

6. /submitDocuments

Service provides a mechanism to submit standard forms and agreements that you typically submit with every new application. Instead of submitting standard forms and agreements with every application, you can submit these documents at the start of every business day. We store these documents on our servers, and will use them for new application requests.

Note: The service will not process Tax Form Documents. They should be submitted for every application.

HTTP Request Type POST

Request Parameters in JSON Format:

```
{
 "CSID":"<client-service-id>",
 "payload":"<encrypted and signed archive file, encoded in base64>"
}
```

Service accepts the following parameters

- CSID Unique identifier issued by Interactive Brokers.
- payload Encrypted and signed archive file, containing forms and agreements

Service Response in JSON Format:

```
{
 "timestamp": "<timestamp>",
```

"isProcessed":"<true|false>",

"fileData":" {

"name":"<response-file-name>",

"data":"<encrypted response XML, encoded in base64>"

}

}

7. /getAccountStatus

Service to get status of one of more accounts.

HTTP Request Type POST

Request Parameters in JSON Format:

{

"CSID":"<client-service-id>",

"accountIds":[“<ID1>","<ID2>",”<ID3>”,...]

}

Service accepts the following parameters

- CSID Unique identifier issued by Interactive Brokers.
- accountIds List of one of more Account Ids.

Service Response in JSON Format:

{

"timestamp": "<timestamp>",

"status" : [{

"accountId" : "<ID1>",

"isError" : false,

"description" : "<Status Description>",

"status" : "<Status Code>"

}, {

"accountId" : "ID2",

"isError" : true,

"error" : "<Error Description>."

}]

}

8. /getPendingTasks

Service used to get pending tasks.

HTTP Request type POST.

Request Parameters in JSON Format:

{

"CSID":"<client-service-id>",

"accountIds":[“<ID1>","<ID2>",”<ID3>”,...]

}

Service accepts the following parameters

- CSID Unique identifier issued by Interactive Brokers.
- accountIds List of one of more Account Ids.

Service Response in JSON format:


```
{
"result" : [
{
"pendingTasks" : [
{
"isRequiredForTrading" : "true",
"isOnlineTask" : "true",
"formNumber" : "2109",
"formName" : "Legal Acknowledgement",
"action" : "to complete",
"isRequiredForApproval" : "true",
"taskNumber" : "0"
}
],
"isError" : false,
"isPendingTaskPresent" : true,
"acctId" : "U2562992",
"error" : ""
}
],
```

"timestamp" : "2018-11-27 06:24:36"

}

Generate PGP Keypair

Follow the steps below to generate PGP Keypair.

Note: Text highlighted in bold is user input.

[user@host ~]$ gpg --gen-key

gpg (GnuPG) 2.0.14; Copyright (C) 2009 Free Software Foundation, Inc.

Select what kind of key you want:

(1) RSA and RSA (default)

---

(4) RSA (sign only)

Your selection? 1

RSA keys may be between 1024 and 4096 bits long.

What keysize do you want? (2048) 4096

Requested keysize is 4096 bits

Specify how long the key should be valid.

0 = key does not expire

<n> = key expires in n days

<n>y = key expires in n years

Key is valid for? (0) 0

Key does not expire at all

Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: Bank One

Email address: bankone@bankonemail.com

Comment: DAM Key

You selected this USER-ID:

"Bank One (DAM Key) <bankone@bankonemail.com>"

---

pub 4096R/C1024DEA 2015-09-10

Key fingerprint = C9E1 A964 8FFE A463 2C79 E700 F933 8E46 C102 4DEA

uid Bank One (DAM Key) <bankone@bankonemail.com>

sub 4096R/CA555322 2015-09-10

Exporting the public key:

[user@host ~]$ gpg --armor --export bankone@bankonemail.com > bankone.asc

Convert public key from ASCII to BINARY format:

[user@host ~]$ gpg --dearmor < bankone.asc > bankone.bin

Encode BINARY version of public key in base64:

[user@host ~]$ cat bankone.bin | base64 | tr -d '\n' > bankone.bin.encoded

Encoded Public Key (bankone.bin.encoded) needs to be exchanged with Interactive Brokers.

Request CSID

Send an email to Interactive Brokers at salesengineering@interactivebrokers.com. Please include Master Account Id and encoded version of your Public Key with the email. Interactive Brokers will email your IB Public Key and CSID (Client Service Id).

Note: IB Public Key (IBPublicKey.encoded) sent by Interactive Brokers will be encoded in base64.

Once you receive IB Public Key, follow these steps below:

Decode IB Public Key

[user@host ~]$ cat IBPublicKey.encoded | base64 -d > IBPublicKey.asc

Import IB Public Key in your key ring:

[user@host ~]$ gpg --import -vv IBPublicKey.asc

Sign IB Public Key using the following command.

[user@host ~]$ gpg –sign-key ibpublickey@bankonemail.com

Encrypt and Encode File

Encrypt and Sign ZIP file:

```
[user@host ~]$  gpg -vv --default-key bankone@bankonemail.com
                --yes --output BankOneData.file.gpg
                --batch --passphrase <Passphrase set when generating GPG Key Pair>
                --encrypt --sign -r ibpublickey@bankonemail.com BankOneData.file.zip
```

Encode File in base64:

```
[user@host ~]$ cat BankOneData.file.gpg | base64  > BankOneData.file.gpg.encoded
```

Decode and Decrypt File

Decode File encoded in base64:

[user@host ~]$ cat BankOneData.file.gpg.encoded | base64 -d > BankOneData.file.gpg

Decrypt file:

[user@host ~]$ gpg --decrypt -o BankOneData.file BankOneData.file.gpg