Coinstore API Documentation

The coinstore API exists to allow developers to interface into bitcoin type coins. It supports the developer by providing a consistent interface into multiple coins as well as extending the basic coin APIs to provide additional game and ecommerce functionality. The interface presented to the developer is based on the JSON standard.

Important:
Any developer wishing to use this API should contact oldandgrey via the Coin2 BB on bitcointalk.org to be allocated a game id/ application id.

JSON Interface

The JSON interface works by sending an array of data stored in JSON format to the API server using a POST transaction. The response is then received in JSON format from the API server and must be decoded before data is extracted and used. The commands to do this depend on the language/ environment you are using.

For example in PHP the command to convert an array to JSON format is :-

```
json_encode(array_to_be_encoded)
```

and the command to decode is :-

```
json_decode(array_to_be_decoded)
```

This document will use the examples in the test.php program (included as Appendix 1) to demonstrate the various API's currently available. Please note that these API's are not a definitive list as API's are being added as the need occurs.

We will first briefly look at authentication and then look at each API one at a time. Note that in any transaction there may be multiple faces to authentication/ communication with the API server.

Player related transactions – receiving payments, using in game tokens etc.
Game related transactions – making payouts

API Authentication

The API currently supports 3 types of authentication – ANON/ USER/ TOKEN

ANON authentication is used to provide access to those API functions that may be accessed by anyone whether or not they have an account in the Web Wallet or not. An example of this would be the recipient of an eCard. This is really “No Authentication”

USER authentication is where the developer supplies the userid and password of the account that they wish to work with. Typically this would be the userid and password of the game account when making payouts. This type of authentication is best suited to server to server authentication and should normally only be used in conjunction with TFA.

TOKEN authentication is where a pre-shared token is used to authenticate the user. This token is changed with every API call and has an expiry time. It is reset/ initiated when the user logs into the wallet to play a game or when any other type of authentication is used in the API.It provides a simple yet secure form of authentication.It is reset when any other form of authentication is used.

Care should be used regarding any kind of authentication especially for games that are downloaded onto users devices before running since these can often be disassembled or their connection “sniffed” to reveal userid’s and passwords.

In the following examples the array to be sent to the API server is $data. We will start by adding the values for authentication and then later add to these the required values for the API call we wish to make.

To use ANON Authentication

```
$data['AUTH'] = "ANON";
```

To use USER Authentication

```
$data['AUTH'] = "USER";
$data['username'] = $username;          // Your Web Wallet userid
$data['password'] = $password;          // Your Web Wallet password
```

To use TOKEN Authentication

```
$data['AUTH'] = "TOKEN";
$data['id_token'] = 'dc3cd3654ad79e2fa5af782a4e849b';
```


Most APIs return either a simple variable containing a value or an array similar to below:-

```
$response['response'] - Response from API
$response['isValid']   - true/ false
$response['error']     - Array of error messages
```

On receiving a response from the API the first thing is to check to see the result of "isValid".

Suggestion

I strongly suggest you download and install a "xampp" installation and drop the test program into the htdocs directory (typically C:\xampp\htdocs). This is a prepackaged installation of apache, mysql, php and perl. It is easy to install and will provide you with a full test webserver running on your PC. Using the test.php program provided in apendix 1 of this document you will be able to see the data you send to the API server and the data received back. This will help immensely your understanding of the API and how it works.

https://www.apachefriends.org/index.html

You can then call the test program with the URL :-

http://localhost?test.php=nn

Where nn is the test you wish to run. The program will show you the data sent to the API server and the data (response) received back. This is the easiest way to understand the API.

To enable you to play with authentication you can also use the extended URL

http://localhost/test.php?auth=USER&coin=C2&test=1

By substituting the values of USER, C2, 6 with the values you wish you can change the authentication method and see how it affects the transaction.

The currently supported APIs are as follows :-

```
jsonTest1                // Returns string "JSONText"
jsonTest2';              // Returns a fuller test message
getbalance';             // Get the Balance of my account
getaddressesbyaccount';  // Get all my wallet addresses for receiving coin2
getaccountaddress';      // Return current wallet bitcoin address
validateaddress';        // Validates a bitcoin address and returns details
getnewaddress';          // Creates a new receiving address for current wallet
getreceivedbyaccount';   //Gets the balance of your account
listtransactions';       // List recent transactions on your coin account
gettransaction';         // Get details of one specific transaction
sendtoaddress';          // Make a payment from your account to another address
coinpayout';             // A more flexible version of sendtoaddress
verifytoken";            // Check the in-game token is valid and not expired
usetoken";               // Mark the in-game token as used
getscores';              // Return details of the top 10 high scores for a game
storescore';             // Record
gettokens';              // Get all Users tokens for this game
getecarddetails';        // Get details for an ecard
get_webpay_token';       // Setup a webpay transaction for user payments
check_webpay_token';     // Check to see if user has made payment
```


API Transactions

jsonTest1

This is a simple API that will return a transaction containing three things :-

data	-	A simple string containing the text “jsonTest1”
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 1 in test.php

The following shows a sample session

Test 1

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => jsonTest1
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"json
Test1"}
```

Response :-

```
Array
(
    [data] => JSONText
    [token] => d0425e703dd4e31e41ffebca050284
    [lease] => 1800
)
```


jsonTest2

This is a simple API that will return a transaction containing three things :-

data	-	A simple string containing the text “jsonTest2”
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 2 in test.php

The following shows a sample session

Test 2

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => jsonTest2
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"json
Test2"}
```

Response :-

```
Array
(
    [data] => Array
        (
            [isValid] => true
            [response] => JSON Array Test
            [error] => Array
                (
                    [0] => Test Error Message
                )

        )

    [token] => 079fb1a81364e83c3113c52552fd40
    [lease] => 1800
)
```


getbalance

This is a simple API that returns the balance of the current account :-

data	-	Account balance
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 3 in test.php

The following shows a sample session

Test 3

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => getbalance
    [coin] => C2T
    [walletid] => mywalletid
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"getb
alance","coin":"C2T","walletid":"mywalletid"}
```

Response :-

```
Array
(
    [data] => 0
    [token] => 9d882c7608adefefec04e335e388c1
    [lease] => 1800
)
```


getaddressesbyaccount

This is a simple API that will return an array of the receiving addresses for the specified account :-

data	-	An array of Coin addresses
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 4 in test.php

The following shows a sample session

Test 4

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => getaddressesbyaccount
    [coin] => C2T
    [walletid] => mywalletid
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"geta
ddressesbyaccount","coin":"C2T","walletid":"mywalletid"}
```

Response :-

```
Array
(
    [data] => Array
        (
            [0] => CKP584yoo5EAxaZQcTaN2wv2ziAnSRJ859
            [1] => CUwSmkRvKBUmxQSzJ8N9T5SSy3BBeW86wS
        )

    [token] => 9ba455af406ea5085f535900bb8a9c
    [lease] => 1800
)
```


validateaddress

This is a simple API that checks if a given address is a valid address :-

data	-	An array containing a flag (isValid) which indicates if the address is valid
token	-	A token that may be used for token based authentication
Lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 5 in test.php

The following shows a sample session

Test 5

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => validateaddress
    [coin] => C2
    [parm1] => CGtAbZA3iaFSkRjgX1EMLWRcKt8DHeSLTC
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"valid
ateaddress","coin":"C2","parm1":"CGtAbZA3iaFSkRjgX1EMLWRcKt8DHeSLTC"}
```

Response :-

```
Array
(
    [data] => Array
        (
            [isValid] => true
            [address] => CGtAbZA3iaFSkRjgX1EMLWRcKt8DHeSLTC
            [ismine] => 1
            [isscript] =>
            [pubkey] =>
03050970650485ed7f63be40d715fd177c550954ec78ce9259c8f7870bb14b422c
            [iscompressed] => 1
        )

    [token] => 42e81700790769c00e5e0b693f89da
    [lease] => 1800
)
```


getnewaddress

This is a simple API that will create a new receiving address and return it :-

data	-	The new coin address
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 6 in test.php

The following shows a sample session

Test 6

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => getnewaddress
    [coin] => C2
    [walletid] => mywalletid
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"getn
ewaddress","coin":"C2","walletid":"mywalletid"}
```

Response :-

```
Array
(
    [data] => CdFg1Q2C4ajyqJjoa2v2qKjVsDkyWAebHC
    [token] => 7f9e20aa74d0ff74dbbc4e12434a66
    [lease] => 1800
)
```


getreceivedbyaccount

This is a simple API that will return the total value received by this account :-

data	-	Value of transactions received
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 8 in test.php

The following shows a sample session

Test 8

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => getreceivedbyaccount
    [coin] => C2
    [walletid] => mywalletid
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"getre
ceivedbyaccount","coin":"C2","walletid":"mywalletid"}
```

Response :-

```
Array
(
    [data] => 1080.05
    [token] => 011ab91ac43f5666397dbd8853ca91
    [lease] => 1800
)
```


listtransactions

This is a simple API that will list recent transactions in the wallet :-

data	-	An array where each element is an array of data representing a transaction
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 9 in test.php

The following shows a sample session

Test 9

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => listtransactions
    [coin] => C2
    [walletid] => mywalletid
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"listtr
ansactions","coin":"C2","walletid":"mywalletid"}
```

Response :-

```
Array
(
    [data] => Array
        (
            [0] => Array
                (
                    [account] => mywalletid
                    [address] => CNT5VSrpvr3eNo3FWjJGeXEMEAsWwgrxJL
                    [category] => send
                    [amount] => -1
                    [fee] => -0.01
                    [confirmations] => 316444
                    [blockhash] =>
fc9bbf9a84035083f2ecdcd7e737d584123c450792e15b228bd8ed91396c9445
                    [blockindex] => 2
                    [blocktime] => 1409304879
                    [txid] =>
3245d0e99375adb546c18ed96750ad8d4373d34a8899c23ac2389c18bb6e9c81
                    [time] => 1409304684
                    [timereceived] => 1409304684
                )

            [1] => Array
                (
                    [account] => mywalletid
                    [address] => CNT5VSrpvr3eNo3FWjJGeXEMEAsWwgrxJL
```



```
                        [category] => send
                        [amount] => -1
                        [fee] => -0.01
                        [confirmations] => 304286
                        [blockhash] =>
dc810bc69a63656baa2ac83af82ba8dea6803b5e8d460bc8f76798eab809c6c6
                        [blockindex] => 2
                        [blocktime] => 1410027122
                        [txid] =>
2039408bb61467b529da876673149cc2cc82f8bb155c3297f462fb178c7d9168
                        [time] => 1410027058
                        [timereceived] => 1410027058
                        [comment] => Test
                    )
            )

    [token] => 06553e99f6e8057a8a5fd808acf102
    [lease] => 1800
)
```


gettransaction

This is a simple API that will return details of a given transaction :-

data	-	An array containing details of the transaction
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 10 in test.php

The following shows a sample session

Test 10

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => gettransaction
    [coin] => C2
    [parm1] => 8a130a495fa6b3ef3545220c2742cb92e8f61a5f072154d31da85492fe2c9459
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"gettr
ansaction","coin":"C2","parm1":"8a130a495fa6b3ef3545220c2742cb92e8f61a5f072154d31da85492fe2
c9459"}
```

Response :-

```
Array
(
    [data] => Array
        (
            [isValid] => true
            [address] => 8a130a495fa6b3ef3545220c2742cb92e8f61a5f072154d31da85492fe2c9459
            [confirmations] => 321526
            [blockhash] =>
c5d3984f40a43a877797c94746b936057601d35dca03f5f3f8ebf2b1df9fb9e6
            [blockindex] => 2
            [blocktime] => 1409007438
            [timereceived] => 1409007335
            [amount] => 55
        )

    [token] => 332a42651e99c0257529c2201f787d
    [lease] => 1800
)
```


sendtoaddress

This is a simple API that will send a payment to a provided address :-

data	-	The transaction id of the successful payment
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 11 in test.php

The following shows a sample session

Test 11

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => sendtoaddress
    [coin] => C2
    [walletid] => mywalletid
    [parm1] => CGtAbZA3iaFSkRjgX1EMLWRcKt8DHeSLTC
    [parm2] => 1
    [parm3] => my comment
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"send
toaddress","coin":"C2","walletid":"mywalletid","parm1":"CGtAbZA3iaFSkRjgX1EMLWRcKt8DHeSL
TC","parm2":1,"parm3":"my comment"}
```

Response :-

```
Array
(
    [data] => d4f304553dce0b33126a27a2c50d48b1e9a7150d3c4bc47f0dd843e303ed2481
    [token] => 3205eb056dcc2e54e4f0c38501e588
    [lease] => 1800
)
```


coinpayout

This is a simple API that is an extension of the sendtoaddress API. As well as being able to send a payment to an address, you can also identify the recipient using an email address, a WebPay token, or a game token. On completion it returns :-

data	-	The transaction id of the successful payment
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 12 in test.php

The following shows a sample session

Test 12

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => coinpayout
    [coin] => C2
    [walletid] => mywalletid
    [addressType] => WEBPAY
    [parm1] => 981b45f4d019465a6a634255ddb83a85
    [parm2] => 1.05
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"coin
payout","coin":"C2","walletid":"mywalletid","addressType":"WEBPAY","parm1":"981b45f4d019465a
6a634255ddb83a85","parm2":1.05}
```

Response :-

```
Array
(
    [data] => 253791df2913145e3659f79aed2f8a4e3e77a6ef51d5c5750cd7ab2ff23db2d1
    [token] => 591e97a12330387c686a016f1e0079
    [lease] => 1800
)
```


verifytoken

This API takes a game token and returns an array containing a flag confirming if the token is valid (exists, has not expired and has not been completely used) :-

data	-	An array containing a flag confirming if the token is valid
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 13 in test.php

The following shows a sample session

Test 13

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => verifytoken
    [parm1] => 62YRF7DA5CNUMWWY
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"verif
ytoken","parm1":"62YRF7DA5CNUMWWY"}
```

Response :-

```
Array
(
    [data] => Array
        (
            [isValid] => true
            [response] => LIFE
        )

    [token] => f272713fe70c74c399ba122c6a50bc
    [lease] => 1800
)
```


usetoken

This API allows an application to notify the API server that the player wishes to “USE” the token. It returns :-

data	-	An array containing a flag confirming that the token has been accepted.
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 14 in test.php

The following shows a sample session

Test 14

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => usetoken
    [parm1] => 62YRF7DA5CNUMWWY
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"uset
oken","parm1":"62YRF7DA5CNUMWWY"}
```

Response :-

```
Array
(
    [data] => Array
        (
            [isValid] => true
            [response] => LIFE
        )

    [token] => 40099bb483b269d0aa4a917a9ecdd3
    [lease] => 1800
)
```


getscores

This is a simple API that will return a transaction containing three things :-

data	-	An array of the Top 10 scores for the game
token	-	A token that may be used for token based authentication
Lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 15 in test.php

The following shows a sample session

Test 15

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => getscores
    [parm1] => 62YRF7DA5CNUMWWY
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"gets
cores","parm1":"62YRF7DA5CNUMWWY"}
```

Response :-

```
Array
(
    [data] => Array
        (
            [isValid] => true
            [response] => Array
                (
                    [0] => 578 Oeloec
                    [1] => 560 Oeloec
                    [2] => 533 Oeloec
                    [3] => 477 Oeloec
                    [4] => 364 Oeloec
                    [5] => 345 nickquest
                    [6] => 337 Oeloec
                    [7] => 333 Oeloec
                    [8] => 309 nickquest
                    [9] => 307 wheelz1200
                )

        )

    [token] => f484e1b230630491d9c1036f8b28aa
    [lease] => 1800
)
```


getscores

Test 16

This shows that if the getscores API is called using a Construct 2 format array then the API will return an array in Construct 2 format :-

data	-	Construct 2 formatted array of top 10 scores
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 16 in test.php

The following shows a sample session

Data Sent :-

```
Array
(
    [data] => Array
        (
            [username] => myemail@mydomain.com
            [password] => mypassword
            [AUTH] => USER
            [request] => getscores
            [parm1] => 62YRF7DA5CNUMWWY
        )

    [c2dictionary] => true
)
```

Content :-

{"data":{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","reques
t":"getscores","parm1":"62YRF7DA5CNUMWWY"},"c2dictionary":"true"}

Response :-

```
Array
(
    [c2array] => 1
    [data] => Array
        (
            [0] => Array
                (
                    [0] => Array
                        (
                            [0] => 578 Oeloec
                        )

                )

            [1] => Array
                (
                    [0] => Array
                        (
                            [0] => 560 Oeloec
```



```
                )

        )

    [2] => Array
        (
            [0] => Array
                (
                    [0] => 533 Oeloec
                )

        )

    [3] => Array
        (
            [0] => Array
                (
                    [0] => 477 Oeloec
                )

        )

    [4] => Array
        (
            [0] => Array
                (
                    [0] => 364 Oeloec
                )

        )

    [5] => Array
        (
            [0] => Array
                (
                    [0] => 345 nickquest
                )

        )

    [6] => Array
        (
            [0] => Array
                (
                    [0] => 337 Oeloec
                )

        )

    [7] => Array
        (
            [0] => Array
                (
                    [0] => 333 Oeloec
                )
```



```
                    )

                [8] => Array
                    (
                        [0] => Array
                            (
                                [0] => 309 nickquest
                            )

                    )

                [9] => Array
                    (
                        [0] => Array
                            (
                                [0] => 307 wheelz1200
                            )

                    )

            )

        [size] => Array
            (
                [0] => 10
                [1] => 1
                [2] => 1
            )

        [token] => 8928f213197c5c45633c819158ef97
        [lease] => 1800
)
```


storescore

This API will store the players score on the API server. It uses a game token to identify the game that the score should be stored against. :-

data	-	Confirmation that the score has been stored
token	-	A token that may be used for token based authentication
Lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 17 in test.php

The following shows a sample session

Test 17

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => storescore
    [parm1] => 62YRF7DA5CNUMWWY
    [parm2] => 208
)
```

Content :-

{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"storescore","parm1":"62YRF7DA5CNUMWWY","parm2":"208"}

Response :-

```
Array
(
    [data] => Array
        (
            [isValid] => true
            [response] => Stored
        )

    [token] => ceb2f44d9032af92b847e9ab58c056
    [lease] => 1800
)
```


gettokens

This API returns an array of tokens that the player has purchased a game. The developer passes the Game Id and one of the players tokens. The API then uses these to identify the Player and all their tokens for this game. The API returns :-

data	-	An array of tokens purchased
token	-	A token that may be used for token based authentication
Lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 18 in test.php

The following shows a sample session

Test 18

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => gettokens
    [parm1] => X73U1FKS63NSTGD5
    [parm2] => 62YRF7DA5CNUMWWY
    [parm3] => ACTIVE
)
```

Content :-


```
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"getto
kens","parm1":"X73U1FKS63NSTGD5","parm2":"62YRF7DA5CNUMWWY","parm3":"ACTIVE"}
```


Response :-

```
Array
(
    [data] => Array
        (
            [data] => Array
                (
                    [0] => V7SR3Y58FVVP2ULC
                    [1] => 53XZWRPWJBJHMU7A
                    [2] => 62YRF7DA5CNUMWWY
                )

            [isValid] => true
            [response] => Got_Tokens
        )

    [token] => cc014de41259922d592016f324c9d9
    [lease] => 1800
)
```


getecarddetails

This API returns details of the the ecard purchased. This includes who it is from, who it is to, message and other details. These are then shown within the card when it is displayed. :-

data	-	An array of ecard information
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 19 in test.php

The following shows a sample session

Test 19

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => ANON
    [token] => 4bb9f85ba3af8c288326b8495e14c101
    [request] => getecarddetails
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"ANON","token":"4bb9f
85ba3af8c288326b8495e14c101","request":"getecarddetails"}
```

Response :-

```
Array
(
    [data] => Array
        (
            [from_name] => Mr Alan Smith
            [from_email] => from@hotmail.com
            [to_name] => Ester
            [to_email] => from@gmx.com
            [message] => Test eCard
            [alt_coin] => C2
            [isValid] => true
            [error] => Array
                (
                    [0] =>
                )

        )

    [token] =>
    [lease] =>
)
```


get_webpay_token

This API allows the game/ application to pass several lines of information regarding a purchase (an invoice) to the API server. These details are then stored on the API server and allocated a transaction number. This transaction number is returned to the calling program. The game/ application then opens a web page passing the transaction id as a parameter. This opens a secure Payment page within coinstore where after logging in the user is presented with the “Invoice details” and has an opportunity to either accept or reject the request to send payment. If accepted the payment is immediately transferred to the game/ applications account. The player then signifies completion of the payment by clicking a button or performing some action within the game/ application at which point the game/ application uses the API below (check_webpay_token) to confirm that the payment was made:-

data	-	The WebPay transaction id
token	-	A token that may be used for token based authentication
Lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 20 in test.php

The following shows a sample session

Test 20

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => get_webpay_token
    [merchant_code] => X73U1FKS63NSTGD5
    [category] => GAMES
    [currency] => C2
    [exchange_rate] => 1
    [total_price] => 75
    [total_lines] => 3
    [total_qty] => 9
    [product_id1] => A0001
    [product_desc1] => Line Description 1
    [unit_price1] => 10
    [qty1] => 3
    [product_id2] => B0001
    [product_desc2] => Line Description 2
    [unit_price2] => 5
    [qty2] => 5
    [product_id3] => C0001
    [product_desc3] => Line Description 3
    [unit_price3] => 20
    [qty3] => 1
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"get_
webpay_token","merchant_code":"X73U1FKS63NSTGD5","category":"GAMES","currency":"C2","ex
change_rate":"1","total_price":"75","total_lines":"3","total_qty":"9","product_id1":"A0001","product_d
esc1":"Line Description 1","unit_price1":"10","qty1":"3","product_id2":"B0001","product_desc2":"Line
```


Description 2","unit_price2":"5","qty2":"5","product_id3":"C0001","product_desc3":"Line Description 3","unit_price3":"20","qty3":"1"}

Response :-

Array
(
[data] => Array
(
[response] => b52cb418a649721d66d88e4561d47684
[isValid] => true
[error] => Array
(
[0] =>
)

)

[token] => 8df579c802cf5432d6e35e39ab404e
[lease] => 1800
)

check_webpay_token

This API is used to confirm that the player/ user has authorised payment to the game/ application :-

data	-	A simple string containing the text “jsonTest1”
token	-	A token that may be used for token based authentication
Lease	-	One of 3 values. “L” - Live, “A” - Accepted, “R” - Rejected

An example of this transaction is shown in Test 21 in test.php

The following shows a sample session

Test 21

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => check_webpay_token
    [merchant_code] => GH4DF5G3FF0553FH
    [webpay_token] => 981b45f4d019465a6a634255ddb83a85
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"chec
k_webpay_token","merchant_code":"GH4DF5G3FF0553FH","webpay_token":"981b45f4d019465a6a6
34255ddb83a85"}
```

Response :-

```
Array
(
    [data] => Array
        (
            [response] => L
            [isValid] => true
            [error] => Array
                (
                    [0] =>
                )

        )

    [token] => 0095cc2ec11aa74ba6ee293a586fdc
    [lease] => 1800
)
```


getcodevalue

This API is used to retrieve the current value of a token. This allows the game to store the value (cost) of an in-game item on the API server. This allows it to be more easily changed in response to exchange rates etc. :-

data	-	A simple string containing the text “jsonTest1”
token	-	A token that may be used for token based authentication
lease	-	The number of seconds the token is valid for

An example of this transaction is shown in Test 22 in test.php

The following shows a sample session

Test 22

Data Sent :-

```
Array
(
    [username] => myemail@mydomain.com
    [password] => mypassword
    [AUTH] => USER
    [request] => getcodevalue
    [GID] => X73U1FKS63NSTGD5
    [code] => POWERUP
)
Content :-
{"username":"myemail@mydomain.com","password":"mypassword","AUTH":"USER","request":"getc
odevalue","GID":"X73U1FKS63NSTGD5","code":"POWERUP"}
```

Response :-

```
Array
(
    [data] => Array
        (
            [data] => Array
                (
                    [value] => 800.00000000
                    [currency] => C2
                    [multi_code] => y
                    [uses] => 1
                    [expires] => 999999999
                )

            [isValid] => true
            [error] => Array
                (
                    [0] =>
                )

        )

    [token] => 8ed9f86ad167feed1a8249e11f3627
```



```
        [lease] => 1800
)
```


Test Program

```
<?php

// Address of my Test API
$url = "http://dev.coinstore.me/wallet/json";

$username = 'myemail@mydomain.com';      // Your Web Wallet userid
$password = 'mypassword';                                  // Your Web Wallet password
$walletid = 'mywalletid';                                                  // Your Wallet name

/*
 * ****************************
 * Check URL parameters to see if any
 * parameters sent to overide Auth vars
 * ****************************
 */

if (isset($_GET['auth'])) {
      $auth = $_GET['auth'];
} else {
      $auth = 'USER';
}

if (isset($_GET['test'])) {
      $test = $_GET['test'];
} else {
      $test = 1;
}

if (isset($_GET['coin'])) {
      $coin = $_GET['coin'];
} else {
      $coin = 'C2';
}

/*
 * ****************************
 * Set Authentication
 * (Sent as part of the request
 * not as a separate call)
 *
 * Use ONE of the following
 * ****************************
 */

switch ($auth) {
      case "USER":
            // Username
            $data['username'] = $username;                  // Your Web Wallet userid
            $data['password'] = $password;                  // Your Web Wallet password
            $data['AUTH'] = "USER";
            break;
```



```
        case "TOKEN":
            // TOKEN
            $data['id_token'] = 'dc3cd3654ad79e2fa5af782a4e849b';    // From web localstorage id_token
            $data['AUTH'] = "TOKEN";
            break;

        default:
            // Anonymous
            // $data['AUTH'] = "ANON";
            break;
}

/*
 * ************************************
 * Decide which test to run
 * ************************************
 */

switch ($test) {

        /*
         * Test API Transactions
         */
        case '1':
            $data['request'] = 'jsonTest1';                    // Returns string "JSONText"
            break;

        case '2':
            $data['request'] = 'jsonTest2';                    // The C2 address you want to send to
            break;

        /*
         * Wallet API Transactions
         */
        case '3':
            $data['request'] = 'getbalance';                   // Get the Balance of my C2 account
            $data['coin'] = $coin;
            $data['walletid'] = $walletid;
            break;

        case '4':
            $data['request'] = 'getaddressesbyaccount'; // Get all my wallet addresses for receiving coin2
            $data['coin'] = $coin;
            $data['walletid'] = $walletid;                      // The name you gave to your Web Wallet
            break;

        case '5':
            $data['request'] = 'getaccountaddress';
            $data['coin'] = $coin;
            $data['walletid'] = $walletid;                      // The name you gave to your Web Wallet
            break;

        case '6':
            $data['request'] = 'validateaddress';
            $data['coin'] = $coin;
            $data['parm1'] = 'CGtAbZA3iaFSkRjgX1EMLWRcKt8DHeSLTC';
```



```
            break;

        case '7':
            $data['request'] = 'getnewaddress';
            $data['coin'] = $coin;
            $data['walletid'] = $walletid;                  // The name you gave to your Web Wallet
            break;

        case '8':
            $data['request'] = 'getreceivedbyaccount';
            $data['coin'] = $coin;
            $data['walletid'] = $walletid;                  // The name you gave to your Web Wallet
            break;

        case '9':
            $data['request'] = 'listtransactions';
            $data['coin'] = $coin;
            $data['walletid'] = $walletid;                  // The name you gave to your Web Wallet
            break;

        case '10':
            $data['request'] = 'gettransaction';
            $data['coin'] = $coin;
            $data['parm1'] = '8a130a495fa6b3ef3545220c2742cb92e8f61a5f072154d31da85492fe2c9459';
// The name you gave to your Web Wallet
            break;

        case '11':
            $data['request'] = 'sendtoaddress';
            $data['coin'] = $coin;
            $data['walletid'] = $walletid;                  // The name you gave to your Web Wallet
            $data['parm1'] = 'CGtAbZA3iaFSkRjgX1EMLWRcKt8DHeSLTC';
            $data['parm2'] = 1;
            $data['parm3'] = 'my comment';
            break;

        case '12':
            $data['request'] = 'coinpayout';
            $data['coin'] = $coin;
            $data['walletid'] = $walletid;                          // The name you gave to your Web
Wallet
            $data['addressType'] = 'WEBPAY';                        // BITCOIN/ EMAIL/ TOKEN/
WEBPAY
            $data['parm1'] = '981b45f4d019465a6a634255ddb83a85'; // Webpay token/ Bitcoin addr/ Email
addr/ Token
            $data['parm2'] = 1.05;
            break;

        /*
          * Token API transactions
          */
        case '13':
            $data['request'] = "verifytoken";                       // Usec2token is same format and same
parameteres
            $data['parm1'] = "62YRF7DA5CNUMWWY";
            // $data['parm1'] = "XXXX-XXXX-XXXX-XXXX";
```



```
            break;

        case '14':
            $data['request'] = "usetoken";                          // Usec2token is same format and same
parameteres
            $data['parm1'] = "62YRF7DA5CNUMWWY";
            // $data['parm1'] = "XXXX-XXXX-XXXX-XXXX";
            break;

        /*
          * Top Ten Score Transactions
          */
        case '15':
            // Return as an array (Most Users)
            $data['request'] = 'getscores';
            $data['parm1'] = '62YRF7DA5CNUMWWY';                    // Users "SCORE" token (Allows
scores to be uploaded)
            break;

        case '16':
            // Return in Construct 2 compatible array
            $data['request'] = 'getscores';                         // Method for Construct 2 only
            $data['parm1'] = '62YRF7DA5CNUMWWY';                    // Users "SCORE" token (Allows
scores to be uploaded)
            // Save data to temporary variable
            $temp = $data;

            // Clear $data as construct 2 sends data as an array within an array
            $data = '';
            $data['data'] = $temp;                                  // Set to temporary array
            $data['c2dictionary'] = 'true';                         // Return result as Construct2
dictionary array
            break;

        case '17':
            $data['request'] = 'storescore';
            $data['parm1'] = '62YRF7DA5CNUMWWY';                    // Users "SCORE" token (Allows
scores to be uploaded)
            $data['parm2'] = '208';                                 // score
            break;

        case '18':
            $data['request'] = 'gettokens';                         // Get all Users tokens for this game
            $data['parm1'] = 'X73U1FKS63NSTGD5';                    // Game Code
            $data['parm2'] = '62YRF7DA5CNUMWWY';                    // Users "SCORE" token (Allows
scores to be uploaded)
            $data['parm3'] = 'ACTIVE';                              // Type of tokens to return
(ALL/USED/ACTIVE)
            break;

        /*
          * Get eCard
          */
        case '19':
            $data['AUTH'] = 'ANON';
            $data['token'] = '4bb9f85ba3af8c288326b8495e14c101';
```



```
            $data['request'] = 'getecarddetails';
            break;

        /*
         * ************************************
         * Tournament API calls
         * ************************************
         */

        case '20' :
            $data['request'] = 'get_webpay_token';
            $data['merchant_code'] = 'X73U1FKS63NSTGD5';          // Game ID
            $data['category'] = 'GAMES';                         // Category
            $data['currency'] = $coin;                           // Currency
            $data['exchange_rate'] = '1';                        //
            $data['total_price'] = '75';                         //
            $data['total_lines'] = '3';                          //
            $data['total_qty'] = '9';                            //

            $data['product_id1'] = 'A0001';                      // Game Product ID
            $data['product_desc1'] = 'Line Description 1';       //
            $data['unit_price1'] = '10';                         //
            $data['qty1'] = '3';                                 //

            $data['product_id2'] = 'B0001';                      // Game Product ID
            $data['product_desc2'] = 'Line Description 2';       //
            $data['unit_price2'] = '5';                          //
            $data['qty2'] = '5';                                 //

            $data['product_id3'] = 'C0001';                      / Game Product ID
            $data['product_desc3'] = 'Line Description 3';       //
            $data['unit_price3'] = '20';                         //
            $data['qty3'] = '1';                                 //
            break;

        case '21' :
            $data['request'] = 'check_webpay_token';
            $data['merchant_code'] = 'GH4DF5G3FF0553FH';         // Game ID
            $data['webpay_token'] = '981b45f4d019465a6a634255ddb83a85';                    //
Category
            break;

        case '22':
            $data['request'] = 'getcodevalue';
            $data['GID'] = 'X73U1FKS63NSTGD5';                  // Gid
            $data['code'] = 'POWERUP';                           // Code
            break;

        default:
            break;
}

/*
 * ************************************
 * Code to send Message
 * ************************************
```



```
 */

$content = json_encode($data); // Encode $jdata for sending to the C2 JSON URL
// Following code curtesy of stackoverflow on internet
$curl = curl_init($url);
curl_setopt($curl, CURLOPT_HEADER, false);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curl, CURLOPT_HTTPHEADER, array("Content-type: application/json"));
curl_setopt($curl, CURLOPT_POST, true);
curl_setopt($curl, CURLOPT_POSTFIELDS, $content);

$json_response = curl_exec($curl);

$status = curl_getinfo($curl, CURLINFO_HTTP_CODE);

if ($status != 200) {
      die("Error: call to URL $url failed with status $status, response $json_response, curl_error " .
curl_error($curl) . ", curl_errno " . curl_errno($curl));
}

curl_close($curl);

$response = json_decode($json_response, true);

echo "Data Sent :-<br><pre>";
print_r($data);
echo "</pre>";

echo "Content :-<br><pre>";
print_r($content);
echo "</pre>";

echo "Response :-<b><pre>";
print_r($response);
echo "</pre>";
?>
```

