Bloc New Server Description

Updated 6 Feb 2022

Purpose

Bloc Server has had a complete rewrite. There are several reasons behind this. The first is to move functionality from the mobile apps to the server. This has several benefits: that code is not duplicated on the different mobile platforms, updates to that code are instant and don't require users updating their app, the database is more secure (the previous architecture had several vulnerabilities – spoofing the identity of a Bloc app to access the database would not have been particularly difficult).

The initial purpose of Bloc Server was solely to use geo location code which is now redundant. Removing that and other redundant code has allowed a complete redesign of the server which now has a consistent interface, much better internal workflow and consistent error reporting.

Compatibility

The new Bloc Server is compatible with apps that call the old server apart from the new 'sender' parameter which is now required. This is sent in the request json at the same level as the existing 'action' and 'request' nodes which are unchanged for pre-existing actions.

So requests are now in the format:{"action":<action>", "request":{<request data>}, "sender":{ {<sender info>}} where sender info is:

"blocID":	bloc id of user	long
"platform":	platform (ie iOS or Android)	string
"version":	version of app	string

~~The two servers will run concurrently for a period. In order to switch to the new one simply change the http endpoint to http://blocservlet.uc2kbwes8t.eu-west-1.elasticbeanstalk.com/blocserver/.~~
The https endpoint should now be https://server.getonbloc.com/BlocServletCurrent this redirects to the actual server in use.

All existing requests should continue to work as before. The responses will be much the same as before other than have consistent error reporting.

At this time there is not an accessible sandbox version of the new server, development of the server has been carried out offline.

Most of the new server api is implemented in the DBManager class (on iOS). Effectively most of the calls to DBManager remain the same, their internal implementation now calls ServerManager as opposed to DynamoDBManager. The switch of implementation is on a compiler flag #if useServer. That flag is used in a few other places in the iOS source.

Data Formats

Bloc Date, Day, Month formats

Bloc Date is the number of seconds since 1 January 1970. Both iOS and java provide methods to convert native dates to this format. It is described as a double in much of this documentation but is in fact a number in string format in requests and responses.

Bloc Day is the number of elapsed days since 1 January 2018 where 1 January 2018 is 0. To compute the current bloc day, get the number of seconds since 1 January 1970 (ie the Bloc date), subtract 1514764800 (00:00 1 January 2018 in bloc date format) then divide by the number of seconds in a day:

```
{
    long temp = seconds - 1514764800;
    long day = temp / (24 * 60 * 60);
    NIDINFO("converted %ld secs to %d days via temp %ld", seconds, (int)day, temp);
    return((int)day);
}
```

Bloc Month is the number of elapsed months since January 2018 where January 2018 is 0.

Response data formats

Adverts contain the info in response to postGetAdvertList. They contain the following key/object pairs:

“advertID”:	id of Advert	long
“advertiserID”:	id of Advertiser	long
“eventID”:	id of Event	long
“placeID”:	id of Place (optional)	long
“text”:	Advert’s text	string (may be “”)
“imageName”:	name of image	string

“videoName”:	name of video	string
“logoName”:	name of logo	string
“latitude”:	latitude of Place	double
“longitude”:	longitude of Place	double
“adType”:	type of ad (B,S,G or E)	string
	B=Bronze, S=Silver, G=Gold, E=Event.	
“endDate”:	end date in bloc format	double
“startDate”:	start date in bloc format	double
“modDate”:	modified date in bloc format	double
“url”:	website url	string
“nearby”:	text to show in nearby notification	string

Chat as per the DynamoDB table

“send”:	sender’s fb ID	long *deprecated May2020*
“sendB”:	sender’s bloc ID	long *new May2020*
“date”:	date in bloc format	double
“rcvr”:	receiver’s fbID	long *deprecated May2020*
“rcvrB”:	receiver’s bloc ID	long *new May2020*
“text”:	chat text	string
“RecS”:	Receiver has seen flag	string
	‘Y’ if viewed. only relevant if receiver is current user	

CheckReview as per the DynamoDB table

“uniqueID”:	check review id	long
“blocID”:	user’s blocID	long *new May2020*
“date”:	date in bloc format	double
“advertID”:	associated Advert	long
“placeID”:	associated Place	long
“logo”:	name of logo	string

Earnchecks are used to decide whether a user has visited a Place on the date of their check-in there

“day”:	day of check in	int	
“eventID”:	id of Event	long	“latitude”:
latitude of Place	double		
“longitude”:	longitude of Place	double	
“radius”:	radius for Place	double	

Event as per the DynamoDB table

“uniqueID”:	Event ID	long
“placeID”:	Place ID	long
“lat”:	latitude of Place	double
“lng”:	longitude of Place	double
“date”:	date in bloc format	double
“type”:	type of Event (ie “Out”)	string
“people”:	attendees	(optional) array of longs *deprecated May2020*
“blocIDs”:	attendees	(optional) array of longs *new May 2020*
“advertType”:	type of ad (X,B,S,G or E) Optional X=None, B=Bronze, S=Silver, G=Gold, E=Event.	string

EventInfos are part of the response to query-people2. They contain the Event info repeated across different people that is visible on the People screen. They contain the following key/object pairs:

“eventID”:	id of Event	long
“placeID”:	id of Place	long
“name”:	unique name of Place	string
“distance”:	distance	double
“date”:	date in bloc format	double

Messages contain the info in response to get-bufferedMessages. They contain the following key/object pairs all after pmtik are optional depending on message type:

“blocID”:	BlocID	long
“date”:	bloc date of message	long
“pmtik”:	message type	int
“inviteKey”:	id of inviter	long
“rcvr”:	id of receiver	long
“superLikerKey”:	id of user	long
“badge”:	badge number	int
“accepterKey”:	id of user	long
“inviteUsedByKey”:	id of user	long
“titl”:	message title	string
“msg”:	message body	string

“SnNm”:	sender’s name	string
“date”:	date in bloc format	double
“news”:	newsflash id	long

Newsflash as per the DynamoDB table

“id”:	check review id	long	
“event”:	user’s fbID	long	
“date”:	date in bloc format	double	
“post”:	posted date in bloc format	double	
“prsn”:	associated Person fb id	long	*deprecated May2020*
“blocID”:	associated Person bloc id	long	*new May2020*
“desc”:	Newsflash text	string	
“type”:	type of newsflash	string	
	(One of “Welcome”, “Example”, “CheckIn”, “Out”, “New”, “Ad”, “App” , “ECP”, “ECC”)		
“titl”:	Event title (rarely used)	string	

Person as per the DynamoDB table

“blocID”:	id of Person	long	*new May20*
“fbID”:	id of Person	long	*deprecated May20*
“name”:	Person’s name	string	
“fstn”:	Person’s first name	string	
“sex”:	Person’s gender (M,F or O)	string	
~~“lastName”:~~	~~Person’s last name~~	~~string~~	
“over”:	over 18 flag (always “Y”)	string	
“last”:	date of last update in bloc format	double	
“bday”:	date in bloc format	double	
“push”:	QuickBlox id no longer used	int	*deprecated*
“nvts”:	list of people liked	array of longs	*deprecated May20*
“likesB”:	list of people liked	array of longs	*new May20*
“fnds”:	list of friends	array of longs	*deprecated May20*
“fndsB”:	list of friends	array of longs	*new May20*
“cImg”:	bit flag for custom images	int	
“flgs”:	Flags	long	
“liks”:	quantity of superlike tokens	int	
“sub”:	currently subscribed to Bloc Plus	string	

	("Y" or "N")		
"nxtT":	date of next tokens in bloc format	double	
"mths":	number of months left	int	
"expS":	date of expiry in bloc format	double	
"used":	number of tokens used	int	
"slee":	list of people superliked	array of longs	*deprecated May20*
"slac":	list of people accepted SLs from	array of longs	*deprecated May20*
"slrj":	list of people rejected SLs from	array of longs	*deprecated May20*
"slbk":	people accepted this users SLs	array of longs	*deprecated May20*
"slrcvd":	people who have sent user SLs	array of longs	*deprecated May20*
"sleeB":	list of people superliked	array of longs	*new May20*
"slacB":	list of people accepted SLs from	array of longs	*new May20*
"slrjB":	list of people rejected SLs from	array of longs	*new May20*
"slbkB":	people accepted this users SLs	array of longs	*new May20*
"slrcvdB":	people who have sent user SLs	array of longs	*new May20*
"abot":	About text	string	
"job":	Job	string	
"mplr":	Employer	string	
"scol":	School	string	
"loc":	Location	string	
"drnk":	Drink	string	
"smk":	Smoke	string	
"bmth":	Birthday Month	string	
"trik":	Trick	string	
"gnre":	Music Genre	string	
"trak":	Track	string	
"powr":	Power	string	
"bket":	Bucket list	string	
"anal1":	last Bloc Day user activated Bloc	int	
"anal2":	last Bloc Month user activated Bloc	int	
"bloc$":	bloc dollars in this account	int	*new Jan22*
"mobile":	mobile number	string	*new Jan22*
"email":	email	string	*new Dec21*

PersonInfos are the other part of the response to query-people2. They contain the Person info that is visible on the People screen. They contain the following key/object pairs:

"fbID":	facebook id of Person	long	*deprecated May 2020*
"blocID":	bloc id of Person	long	*new May 2020*

“eventID”:	id of Event	long
“first”:	Person’s first name	string
“cImg”:	bit flag for custom images	int
“bday”:	date in bloc format	double
“sex”:	Person’s gender (M,F or O)	string
“pop”:	popularity (number of superlikes)	int
“image”:	blurred image data	Base64EncodedString .png

Place as per the DynamoDB table

“id”:	id of Place	long
“name”:	name	string
“uniq”:	unique name	string
“type”:	list of types (bar, night_club etc)	array of string
“phone”:	phone number	string
“lat”:	latitude of Place	double
“lng”:	longitude of Place	double
“webs”:	website	string
“addr”:	address	string
“shpc”:	sort post code (being phased out)	string
“city”:	‘locality’	string
“cntry”:	country	string
“gogl”:	google id	string
“phone”:	phone number	string
“mtro”:	metropolis (being phased out)	int
“ecnt”:	number of events at this place	int
“advert”:	type of advert (X,B,S,G or E)	string
	Optional X=None, B=Bronze, S=Silver, G=Gold, E=Event.	

Specials are the data structure for the new specials section of the new explore screen. They are sent back in some responses from the server as listed below. They contain the following key/object pairs:

“specialID”:	id of Special	long
“eventID”:	id of Event	long
“placeID”:	id of Place	long
“text”:	text associated with Special	string (may be “”)
“location”:	contains:	
“latitude”:	latitude of Place	double
“longitude”:	longitude of Place	double
“imageName”:	name of the image	string

Venues are a combination of Bloc Events and Places. They are sent back in some responses from the server as listed below. They contain the following key/object pairs:

“eventID”:	id of Event	long
“placeID”:	id of Place	long
“eventTitle”:	title of Event	string (may be “”)
“placeName”:	unique name of Place	string
“location”:	contains:	
“latitude”:	latitude of Place	double
“longitude”:	longitude of Place	double
“count”:	number of Events at Place	int

Errors

If a request completes correctly the status code will be 200 (OK), if it fails the status code will be either 500 (Internal Server Error) or 400 (Bad Request). If there is an error there will also be an “error” key describing where in the server the error occurred and a “message” key describing the problem. There can be multiple errors reported, in which case the keys will have an int appended.

example response to an error

```
{ Status Code: 400, Headers {
   Connection =    (
      close
   );
   Date =    (
      "Thu, 28 Feb 2019 12:15:47 GMT"
   );
   "Transfer-Encoding" =    (
      Identity
   );
} }

JSON
{
   action = "get-earnChecks";
   error = "com.metafore.bloc.blocserver.EarnCheck : getEarnChecks : 166";
   error1 = "com.metafore.bloc.blocserver.EarnCheck : getEarnChecks : 171";
   error2 = "com.metafore.bloc.blocserver.EarnCheck : getEarnChecks : 176";
   message = "no day for getEarnChecks (need either day or startDay and endDay)";
   message2 = "no startDay for getEarnChecks (need either day or startDay and endDay)";
   message3 = "no endDay for getEarnChecks (need either day or startDay and endDay)";
}
```

<u>Image Handling Programming Overview</u>

Image handling needs to compromise between visual quality and time taken to download. The previous method of having 2 or 3 set sizes of image is no longer reasonable. Downloading larger than necessary images and then downscaling in the app is particularly inefficient. The new method consists of always uploading the highest resolution (ie largest) image possible to S3, the server then generates all the known sizes that could be required from the app. The app then downloads only the size of image required, keeping a note of the largest version of an image downloaded to date so that if a smaller image is required it can downscale the existing one as opposed to another download. If a particular size of image is not available on S3 then the app requests it from the server which generates it, saves it for future use, ensures that size is in the ImageSize table in the database and sends that image to the app.

Some requests to the server (ie query-people2) have an imageSize parameter giving the width of images that will be required so that the server can ensure they exist before being required.

A new server request '<u>new-image</u>' should be sent whenever a new image is uploaded - this will delete any scaled images based on the previous image and generate all those needed based on the new image (uploading of which must have been completed prior to sending this request).

All uploaded images should be square. Their size is defined by a single integer for the width which must be the same as the height.

User Incentive Programming Overview

All ⭐ adding/subtracting will be done by the server. All requests to the server that could modify the number of ⭐ will return the number of ⭐ the user has as a result of that interaction (and other interactions that the app may not have been aware of). The ios app has animations on the Store screen where the number of ⭐ is displayed, here that number is updated locally to make the user experience consistent. The number is then corrected if necessary when the transaction on the server is fully completed. There will be a screen accessible from the store screen that lists all transactions so any discrepancies can be understood.

All checkins have the potential to earn the user 3 ⭐ subject to a limit of one reward per day. When a user checks in, if they are at the venue on the day then they get rewarded immediately. If not, the server will remember the potential reward. Daily the app should ask the server for upcoming potential rewards, the server responds with a list of checkins for next two days (to solve the problem of missing a venue just after midnight). The app then has to monitor the user's location to see whether they enter a checked in venue. If they arrive at a venue then they are credited with 3 ⭐ and no more location monitoring is required that day.

For consistency with user incentives the following server requests need to be implemented simultaneously:

User Incentive Requests

deprecated May 2020

action: add-superlikeTokens

purpose: adds to the number of ⭐ for a user.
if reason is ‘Checkin’ the server checks that that user has not received any for that day yet. If they have the ‘amount’ added will be 0.

parameters:

“fbID”:	id of User	long
“amount”:	number of ⭐	int
“reason”:	reason for addition	string
“info”:	additional info (reason dependent)	long

‘reason’ must be one of the following:
‘Purchase’, when ⭐ are purchased in the store, info should be 0
‘Renewal’, when Bloc plus membership renews, info should be 0
‘Instagram’, when the instagram share is done, ,info should be 0
‘Invite’, when an invite is used, info should be the fbID of the new user of the invite
‘Checkin’, info should be events id
‘Provisional’, not currently used
‘First’, when the user first logs on to Bloc, , info should be 0

response: “result”: ‘liks’ : int of number of ⭐ this user currently has,
‘amount’ : number of ⭐ actually added (could be 0)
error if fails

————————————————

new May 2020

action: add-superlikeTokensB

added May 2020

purpose: adds to the number of ⭐ for a user.
if reason is ‘Checkin’ the server checks that that user has not received any for that day yet. If they have the ‘amount’ added will be 0.

parameters:

“blocID”:	id of User	long
“amount”:	number of ⭐	int
“reason”:	reason for addition	string
“info”:	additional info (reason dependent)	long

‘reason’ must be one of the following:
‘Purchase’, when ⭐ are purchased in the store, info should be 0
‘Renewal’, when Bloc plus membership renews, info should be 0
‘Instagram’, when the instagram share is done, ,info should be 0
‘Invite’, when an invite is used, info should be the fbID of the new user of the invite
‘Checkin’, info should be events id
‘Provisional’, not currently used
‘First’, when the user first logs on to Bloc, , info should be 0

response: “result”: ‘liks’ : int of number of ⭐ this user currently has,
‘amount’ : number of ⭐ actually added (could be 0)
error if fails

———————————————————————————————————————-

deprecated May 2020
<u>action</u>: send-superlike

<u>purpose</u>: sends a ⭐ from one user to another

<u>parameters</u>:

“from”:	id of user sending ⭐	long
“to”:	id of user receiving ⭐	int

response: “result”: ‘liks’ : int of number of ⭐ this user currently has, error if fails
———————————————————————————————————————-

new May 2020
<u>action</u>: send-superlikeB

<u>purpose</u>: sends a ⭐ from one user to another

<u>parameters</u>:

“fromB”:	id of user sending ⭐	long

“toB”:	id of user receiving ⭐	int

response: “result”: ‘liks’ : int of number of ⭐ this user currently has, error if fails

---

deprecated May 2020
action: submit_exchange

purpose: submits a user request to exchange ⭐ for money

parameters:

“fbID”:	id of user sending	long
“email”:	email address of user	string
“amount”:	number of ⭐ being exchanged	int

response: “result”: ‘liks’ : int of number of ⭐ this user currently has, error if fails

---

new May 2020
action: submit_exchangeB

purpose: submits a user request to exchange ⭐ for money

parameters:

“blocID”:	id of user sending	long
“email”:	email address of user	string
“amount”:	number of ⭐ being exchanged	int
“redeem_type”:	“cash” or “bloccoin”	string *new Dec21*

response: “result”: ‘liks’ : int of number of ⭐ this user currently has, error if fails

---

deprecated May 2020
action: get-earnChecks

purpose: get information on checkins
either for next two days from server (specify day as today)
or between 2 days (specify startDay and endDay)

parameters:

“fbID”:	id of user	long
“day”:	Bloc current day (optional)	int
“startDay”:	Bloc start day (optional)	int
“endDay”:	Bloc current day (optional)	int

response: “result”: list of Earnchecks

————————————————————————————————————-

new May 2020

action: get-earnChecksB

purpose: get information on checkins
either for next two days from server (specify day as today)
or between 2 days (specify startDay and endDay)

parameters:

“blocID”:	id of user	long
“day”:	Bloc current day (optional)	int
“startDay”:	Bloc start day (optional)	int
“endDay”:	Bloc current day (optional)	int

response: “result”: list of Earnchecks

————————————————————————————————————-

DynamoDB Access Requests

These requests replace the legacy direct access to the DynamoDB database that the Bloc apps have historically used. In early 2019 direct access to the db will be prohibited and all access will have to go via the server. The intention is to increase the security of the database.

deprecated May 2020
action: add-ignorePlace

purpose: add the place id to the list of places to ignore for check-in prompts for that user

parameters:

"fbID":	id of user	long
"placeID":	id of place	long

———————————————————————————————————-

new May 2020
action: add-ignorePlaceB

purpose: add the place id to the list of places to ignore for check-in prompts for that user

parameters:

"blocID":	id of user	long
"placeID":	id of place	long

response: success or error
———————————————————————————————————-

deprecated May 2020
action: add-toEvent

purpose: add the user to an event

parameters:

"fbID":	id of user	long
"eventID":	id of event	long

response: success or error

———————————————————————————————————-

new May 2020

action: add-toEventB

purpose: add the user to an event

parameters:

“blocID”:	id of user	long
“eventID”:	id of event	long

response: success or error

———————————————————————————————————-

deprecated May 2020

action: accept-superlike
This updates both the sender and receiver’s Person
and sends a notification to the original sender of the superlike letting them know the superlike has been accepted

purpose: accept a superlike that has been sent

parameters:

“fbID”:	id of user	long
“from”:	id of person that sent superlike	long

response: success or error

———————————————————————————————————-

new May 2020

action: accept-superlikeB
This updates both the sender and receiver’s Person
and sends a notification to the original sender of the superlike letting them know the superlike has been accepted

purpose: accept a superlike that has been sent

<u>parameters</u>:		
“blocID”:	id of user	long
“from”:	id of person that sent superlike	long

<u>response</u>: success or error

———————————————————————————-

<u>action</u>: check-placeGoogleID

<u>purpose</u>: find out whether a Place with specified google id exists in our db

<u>parameters</u>:

“gogl”:	google id	string

<u>response</u>: “result”: id of Bloc Place if found, error if fails

———————————————————————————-

<u>action</u>: check-placeUniqueName

<u>purpose</u>: find out whether a Place with ‘unique’ name exists in our db
NOTE: a match here does not mean this is the same place (check the location too)

<u>parameters</u>:

“uniq”:	unique name	string

<u>response</u>: “result”: id of Bloc Place if found, error if fails

———————————————————————————-

deprecated May 2020

<u>action</u>: dec-badge

<u>purpose</u>: decrement the badge number for the user

<u>parameters</u>:

“fbID”:	id of user	long

<u>response</u>: success or error

——————————————————————————————-

new May 2020

action: dec-badgeB

purpose: decrement the badge number for the user

parameters:

“blocID”:	id of user	long

response: success or error

——————————————————————————————-

deprecated May 2020

action: delete-match

purpose: delete the specified match

parameters:

“fbID”:	id of this user	long
“mids”:	id of user matched to remove	long

response: success or error

——————————————————————————————-

new May 2020

action: delete-matchB

purpose: delete the specified match

parameters:

“blocID”:	id of this user	long
“mids”:	id of user matched to remove	long

response: success or error

——————————————————————————————-

deprecated May 2020

action: delete-matchEvent

purpose: delete the specified MatchEvent

parameters:

“fbIDA”:	id of this user	long
“fbIDB”:	id of user matched to remove	long

response: success or error

———————————————————————————————————-

new May 2020

action: delete-matchEventB

purpose: delete the specified MatchEvent

parameters:

“blocIDA”:	id of this user	long
“blocIDB”:	id of user matched to remove	long

response: success or error

———————————————————————————————————-

deprecated May 2020

action: delete-person

purpose: delete the specified Person

parameters:

“fbID”:	id of Person	long

response: success or error

———————————————————————————————————-

new May 2020

action: delete-personB

purpose: delete the specified Person

<u>parameters</u>:

“blocID”:	id of Person	long

<u>response</u>: success or error

————————————————————————————————————-

<u>action</u>: delete-newsflash

<u>purpose</u>: delete the specified newsflash

<u>parameters</u>:

“id”:	id of Newsflash	long

<u>response</u>: success or error

————————————————————————————————————-

deprecated May 2020

<u>action</u>: delete-chats

<u>purpose</u>: delete all the chats to or from this user

<u>parameters</u>:

“fbID”:	id of user	long

<u>response</u>: success or error

————————————————————————————————————-

new May 2020

<u>action</u>: delete-chatsB

<u>purpose</u>: delete all the chats to or from this user

<u>parameters</u>:

“blocID”:	id of user	long

<u>response</u>: success or error

————————————————————————————————————-

deprecated May 2020

<u>action</u>: get-badge

<u>purpose</u>: get the badge number for specified user

<u>parameters</u>:

“fbID”:	id of user	long

<u>response</u>: “badge”: int
————————————————————————————————————-

new May 2020
<u>action</u>: get-badgeB

<u>purpose</u>: get the badge number for specified user

<u>parameters</u>:

“blocID”:	id of user	long

<u>response</u>: “badge”: int
————————————————————————————————————-

<u>action</u>: get-currentPlace

<u>purpose</u>: find whether the user is in a valid Place

<u>parameters</u>:

“lat”:	latitude of user’s location	double
“lng”:	longitude of user’s location	double

<u>response</u>: “result”: <u>Place</u> on success, error if fails
————————————————————————————————————-

deprecated May 2020
<u>action</u>: get-deletedPerson

<u>purpose</u>: check whether a Person has been deleted (optionally used when get-person returns an error)

<u>parameters</u>:

“fbID”:	id of user	long

response: “result”: “deleted” if the Person has been deleted
——————————————————————————————-

new May 2020
action: get-deletedPersonB

purpose: check whether a Person has been deleted (optionally used when get-person returns an error)

parameters:

“blocID”:	id of user	long

response: “result”: “deleted” if the Person has been deleted
——————————————————————————————-

action: get-event

purpose: read an Event

parameters:

“eventID”:	id of Event	long

response: “result”: Event or error
——————————————————————————————-

action: get-eventDatePlace

purpose: get an event at this Place on this date if it exists

parameters:

“date”:	date	double
“placeID”:	id of place	long

response: “result”: Event or error
——————————————————————————————-

deprecated May 2020
action: get-chatListFrom

purpose: get chats from specified user since specified date when a new user logs on since should be 0, further calls will have a since of the last time this request was made. This is a back up as Firebase Messaging does not guarantee delivery and chats can get lost if Firebase Messaging was the only method of receiving chats.

parameters:

“fbID”:	id of user	long
“since”:	Bloc Date	double

response: “chats”: list of Chats on success, error if fails

————————————————————————————————————-

new May 2020

action: get-chatListFromB

purpose: get chats from specified user since specified date
when a new user logs on since should be 0, further calls will have a since of the last time this request was made. This is a back up as Firebase Messaging does not guarantee delivery and chats can get lost if Firebase Messaging was the only method of receiving chats.

parameters:

“blocID”:	id of user	long
“since”:	Bloc Date	double

response: “chats”: list of Chats on success, error if fails

————————————————————————————————————-

deprecated May 2020

action: get-chatListTo

purpose: get chats to specified user since specified date
when a new user logs on since should be 0, further calls will have a since of the last time this request was made. This is a back up as Firebase Messaging does not guarantee delivery and chats can get lost if Firebase Messaging was the only method of receiving chats.

parameters:

“fbID”:	id of user	long
“since”:	Bloc Date	double

response: “chats”: list of Chats on success, error if fails
————————————————————————————————————-

new May 2020
action: get-chatListToB

purpose: get chats to specified user since specified date
when a new user logs on since should be 0, further calls will have a since of the last time this request was made. This is a back up as Firebase Messaging does not guarantee delivery and chats can get lost if Firebase Messaging was the only method of receiving chats.

parameters:

“blocID”:	id of user	long
“since”:	Bloc Date	double

response: “chats”: list of Chats on success, error if fails
————————————————————————————————————-

deprecated May 2020
action: get-checkReviews

purpose: get current check reviews for users

parameters:

“fbID”:	id of user	long

response: “result”: list of CheckReviews on success, error if fails

————————————————————————————————————-

new May 2020
action: get-checkReviewsB

purpose: get current check reviews for users

parameters:

“blocID”:	id of user	long

response:	"result": list of CheckReviews on success, error if fails

————————————————————————————————————-

deprecated May 2020

action:	get-image

purpose:	get the specified image
Please *only* use this to get images that are not currently on S3.
In almost all cases the image will be on S3, calling this will generate the image on S3 and return it to the caller, unless there is not a high res version of this image in which case it will return an error.

parameters:

"fbID":	id of user	long
"number":	number of image (where 0 means the main profile image)	int
"size":	width of (square) image	int

response:	"result": "image"

————————————————————————————————————-

new May 2020

action:	get-imageB

purpose:	get the specified image
Please *only* use this to get images that are not currently on S3.
In almost all cases the image will be on S3, calling this will generate the image on S3 and return it to the caller, unless there is not a high res version of this image in which case it will return an error.

parameters:

"blocID":	id of user	long
"number":	number of image (where 0 means the main profile image)	int
"size":	width of (square) image	int

response:	"result": "image"

action:	get-image

purpose: get the specified image
Please *only* use this to get images that are not currently on S3.
In almost all cases the image will be on S3, calling this will generate the image on S3 and return it to the caller, unless there is not a high res version of this image in which case it will return an error.

parameters:

“fbID”:	id of user	long
“number”:	number of image (where 0 means the main profile image)	int
“size”:	width of (square) image	int

response: “result”: “image”

————————————————————————————————-

deprecated May 2020

action: get-matchEvent

purpose: get event id for match

parameters:

“fbIDA”:	id of this user	long
“fbIDB”:	id of matched user	long

response: “result”: “eventID” event ID on success, error if fails

————————————————————————————————-

new May 2020

action: get-matchEventB

purpose: get event id for match

parameters:

“blocIDA”:	id of this user	long
“blocIDB”:	id of matched user	long

response: “result”: “eventID” event ID on success, error if fails

————————————————————————————————-

deprecated May 2020

action: get-matchList

purpose: get all matched users for a user

parameters:

“fbID”:	id of user	long

response: “result”: “mids” :array of longs on success, error if fails

—————————————————————————————————————-

new May 2020

action: get-matchListB

purpose: get all matched users for a user

parameters:

“blocID”:	id of user	long

response: “result”: “midsB” :array of longs of Bloc IDs on success, error if fails

—————————————————————————————————————-

action: get-newsFlash

purpose: read specified newsflash

parameters:

“id”:	id of newsflash	long

response: “result”: Newsflash on success, error if fails

—————————————————————————————————————-

deprecated May 2020

action: get-newsFlashes

purpose: et the newsflashes for a particular user between two dates

parameters:

“prsn”:	id of user	long
“start”:	start Bloc Date	double
“end”:	end Bloc Date	double

response:	“result”: list of Newsflashes on success, error if fails
———————————————————————————————————-

new May 2020

action:	get-newsFlashesB

purpose:	get the newsflashes for a particular user between two dates

parameters:

“blocID”:	id of user	long
“start”:	start Bloc Date	double
“end”:	end Bloc Date	double

response:	“result”: list of Newsflashes on success, error if fails
———————————————————————————————————-

modified May 2020

action:	get-person

purpose:	get specified Person

parameters:

“fbID”:	id of user	long *deprecated May 2020*
“facebookID”:	facebook id of user	long

response:	“result”: Person on success, error if fails
———————————————————————————————————-

new May 2020

action:	get-personB

purpose:	get specified Person

parameters:

“blocID”:	id of user	long

response:	“result”: Person on success, error if fails. Note as of May 2020 this structure has changed, all key/data pairs that referred to facebook IDs now have a ‘B’ at the end of the key and the data consists of Bloc IDs

——————————————————————————————————-

action: get-place

purpose: get the specified Place

parameters:

“id”:	id of Place	long

response: “result”: Place on success, error if fails

——————————————————————————————————-

action: get-unique

purpose: get the next id to assign to a new item for the specified table

parameters:

“table”:	name of table for new item	long

response: “result”:“unique”: uniqueID (long) on success, error if fails

——————————————————————————————————-

action: get-uniquePlaceID

purpose: get the next place id to assign to a new place

parameters:

none

response: “result”:“unique”: uniqueID (long) on success, error if fails

——————————————————————————————————-

deprecated May 2020

action: get-userLikesSentToday

purpose: get the number of ‘Likes’ this user has sent today (non Bloc Plus subscribers are limited to 3 Likes per day)

parameters:

“fbID”:   fbID   number

<u>response</u>:   “result”:“likesCount”: as int on success, error if fails
——————————————————————————————————————-

new May 2020
<u>action</u>:   get-userLikesSentTodayB

<u>purpose</u>:   get the number of ‘Likes’ this user has sent today (non Bloc Plus subscribers are limited to 3 Likes per day)

<u>parameters</u>:

“blocID”:	fbID	number

<u>response</u>:   “result”:“likesCount”: as int on success, error if fails
——————————————————————————————————————-

<u>action</u>:   get-version

<u>purpose</u>:   get the version (build) number of the latest app available for the specified OS

<u>parameters</u>:

“OS”:	os of device (“ios” only known operating system atm)	string

<u>response</u>:   “version”: version as int on success, error if fails
——————————————————————————————————————-

deprecated May 2020
<u>action</u>:   inc-badge

<u>purpose</u>:   increment the badge value for this user

<u>parameters</u>:

“fbID”:	id of user	long

<u>response</u>:   success or error
——————————————————————————————————————-

new May 2020

action: inc-badgeB

purpose: increment the badge value for this user

parameters:

"blocID":	id of user	long

response: success or error

———————————————————————————————————————-

deprecated May 2020

action: new-image

purpose: when a different image is uploaded to S3 by the app, use this to rebuild all the scaled images based on that image

parameters:

"fbID":	id of user	long
"number":	number of image	int
	(0 for profile, 1 for first extra image etc)	

response: success or error

———————————————————————————————————————-

new May 2020

action: new-imageB

purpose: when a different image is uploaded to S3 by the app, use this to rebuild all the scaled images based on that image

parameters:

"blocID":	id of user	long
"number":	number of image	int
	(0 for profile, 1 for first extra image etc)	

response: success or error

———————————————————————————————————————-

deprecated May 2020

action:	put-chat

purpose:	put a new chat

parameters:

“send”:	id of sender	long
“rcvr”:	id of receiver	long
“date”:	date	double
“text”:	text	string

response:	success or error

——————————————————————————————————————-

new May 2020

action:	put-chatB

purpose:	put a new chat

parameters:

“sendB”:	bloc id of sender	long
“rcvrB”:	bloc id of receiver	long
“date”:	date	double
“text”:	text	string

response:	success or error

——————————————————————————————————————-

deprecated May 2020

action:	put-checkReview

purpose:	add a CheckReview

parameters:

“fbID”:	id of user	long
“advertID”:	id of Advert	long
“placeID”:	id of Place	long
“date”:	date	double
“logo”:	name of logo file	string

response:	success or error

——————————————————————————————————————-

new May 2020

action: put-checkReviewB

purpose: add a CheckReview

parameters:

"blocID":	id of user	long
"advertID":	id of Advert	long
"placeID":	id of Place	long
"date":	date	double
"logo":	name of logo file	string

response: success or error

————————————————————————————————————-

deprecated May 2020

action: put-match

purpose: add a match

parameters:

"fbID":	id of this user	long
"mids":	user's fbID	long

response: success or error

————————————————————————————————————-

new May 2020

action: put-matchB

purpose: add a match

parameters:

"blocID":	bloc id of this user	long
"midsB":	bloc id of matched user	long

response: success or error

————————————————————————————————————-

deprecated May 2020

action: put-matchEvent

purpose: add MatchEvent

parameters:

“fbIDA”:	check review id	long
“fbIDB”:	user’s fbID	long
“eventID”:	event ID	long

response: success or error

———————————————————————————————————-

new May 2020

action: put-matchEventB

purpose: add MatchEvent

parameters:

“blocIDA”:	check review id	long
“blocIDB”:	user’s fbID	long
“eventID”:	event ID	long

response: success or error

———————————————————————————————————-

deprecated May 2020

action: put-newsflash

purpose: add newsflash

parameters:

“id”:	check review id	long
“event”:	event ID	long
“date”:	date in bloc format	double
“post”:	posted date in bloc format	double
“prsn”:	associated Person	long
“desc”:	Newsflash text	string
“type”:	type of newsflash	string
	(One of “Welcome”, “Example”, “CheckIn”, “Out”, “New”, “Ad”, “App” , “ECP”, “ECC” - Usually ‘Out’)	

“titl”:	Event title (rarely used)	string
“spec”:	Special text (rarely used)	string

<u>response</u>: success or error

———————————————————————————————————-

new May 2020

<u>action</u>: put-newsflashB

<u>purpose</u>: add newsflash

<u>parameters</u>:

“id”:	check review id	long
“event”:	event ID	long
“date”:	date in bloc format	double
“post”:	posted date in bloc format	double
“blocID”:	associated Person	long
“desc”:	Newsflash text	string
“type”:	type of newsflash (One of “Welcome”, “Example”, “CheckIn”, “Out”, “New”, “Ad”, “App” , “ECP”, “ECC” - Usually ‘Out’)	string
“titl”:	Event title (rarely used)	string
“spec”:	Special text (rarely used)	string

<u>response</u>: success or error

———————————————————————————————————-

deprecated May 2020

<u>action</u>: put-person

<u>purpose</u>: add the user to the database

<u>parameters</u>:

“fbID”:	id of Person	long
“name”:	Person’s name	string
“sex”:	Person’s gender (M,F or O)	string
“name”:	Person’s name	string
“over”:	over 18 flag (always “Y”)	string
“flgs”:	Flags	long
“liks”:	quantity of superlike tokens	int

“mths”:	number of months left	int
“used”:	number of tokens used	int
“anal1”:	last Bloc Day user activated Bloc	int
“anal2”:	last Bloc Month user activated Bloc	int

<u>optional parameters</u>:

“last”:	date of last update in bloc format	double
“bday”:	date in bloc format	double
“push”:	QuickBlox id no longer used	int
“nvts”:	list of people liked	array of longs
“fnds”:	list of friends	array of longs
“cImg”:	bit flag for custom images	int
“nxtT”:	date of next tokens in bloc format	double
“expS”:	date of expiry in bloc format	double
“fstn”:	Person’s first name	string
“slee”:	list of people superliked	array of longs
“slac”:	list of people accepted SLs from	array of longs
“slrj”:	list of people rejected SLs from	array of longs
“slbk”:	people accepted this users SLs	array of longs
“slrcvd”:	people who have sent user SLs	array of longs
“abot”:	About text	string
“job”:	Job	string
“mplr”:	Employer	string
“scol”:	School	string
“loc”:	Location	string
“drnk”:	Drink	string
“smk”:	Smoke	string
“bmth”:	Birthday Month	string
“trik”:	Trick	string
“gnre”:	Music Genre	string
“trak”:	Track	string
“powr”:	Power	string
“bket”:	Bucket list	string
“sub”:	currently subscribed to Bloc Plus (“Y” or “N”)	string
“fstu”:	date first used (“Y” or “N”)	long

<u>response</u>: success or error

————————————————————————————————————-

new May 2020

action: put-personB

purpose: add the user to the database

parameters:

"blocID":	bloc id of Person (*optional - if fb logon used*)	long
"fbID":	fb id of Person (*optional - if fb logon used*)	long
"name":	Person's name	string
"sex":	Person's gender (M,F or O)	string
"name":	Person's name	string
"over":	over 18 flag (always "Y")	string
"flgs":	Flags	long
"liks":	quantity of superlike tokens	int
"mths":	number of months left	int
"used":	number of tokens used	int
"anal1":	last Bloc Day user activated Bloc	int
"anal2":	last Bloc Month user activated Bloc int	

optional parameters:

"last":	date of last update in bloc format	double	
"bday":	date in bloc format	double	
"push":	QuickBlox id	int	*no longer used*
"nvts":	list of people liked	array of longs	*deprecated*
"likesB":	list of people liked	array of longs	*new*
"fnds":	list of friends	array of longs	*deprecated*
"fndsB":	list of friends	array of longs	*new*
"cImg":	bit flag for custom images	int	
"nxtT":	date of next tokens in bloc format	double	
"expS":	date of expiry in bloc format	double	
"fstn":	Person's first name	string	
"slee":	list of people superliked	array of longs	*deprecated*
"slac":	list of people accepted SLs from	array of longs	*deprecated*
"slrj":	list of people rejected SLs from	array of longs	*deprecated*
"slbk":	people accepted this users SLs	array of longs	*deprecated*
"slrcvd":	people who have sent user SLs	array of longs	*deprecated*
"sleeB":	list of people superliked	array of longs	*new*

“slacB”:	list of people accepted SLs from	array of longs	*new*
“slrjB”:	list of people rejected SLs from	array of longs	*new*
“slbkB”:	people accepted this users SLs	array of longs	*new*
“slrcvdB”:	people who have sent user SLs	array of longs	*new*
“abot”:	About text	string	
“job”:	Job	string	
“mplr”:	Employer	string	
“scol”:	School	string	
“loc”:	Location	string	
“drnk”:	Drink	string	
“smk”:	Smoke	string	
“bmth”:	Birthday Month	string	
“trik”:	Trick	string	
“gnre”:	Music Genre	string	
“trak”:	Track	string	
“powr”:	Power	string	
“bket”:	Bucket list	string	
“sub”:	currently subscribed to Bloc Plus (“Y” or “N”)	string	
“fstu”:	date first used (“Y” or “N”)	long	

response: success or error

———————————————————————————————————-

action: put-place

purpose: add a Place

parameters:

“id”:	id of Place	long
“name”:	name	string
“uniq”:	unique name	string
“type”:	list of types (bar, night_club etc)	array of string
“lat”:	latitude of Place	double
“lng”:	longitude of Place	double

optional parameters:

“phone”:	phone number	string
“webs”:	website	string

“addr”:	address	string
“shpc”:	sort post code (being phased out)	string
“city”:	‘locality’	string
“cntry”:	country	string
“gogl”:	google id	string
“phone”:	phone number	string
“mtro”:	metropolis (being phased out)	int
“ecnt”:	number of events at this place	int
“advert”:	type of advert (X,B,S,G or E)	string
	Optional X=None, B=Bronze, S=Silver, G=Gold, E=Event.	

response: success or error

——————————————————————————————————-

deprecated May 2020

action: put-review

purpose: add review

parameters:

“userID”:	id of user	long
“date”:	date in bloc format	double
“advertID”:	id of Advert	long
“stars”:	number of stars	int
“text”:	text of review	string

optional parameters:

“spend”:	amount spent	string (valid number with 2 decimal places)

response: success or error

——————————————————————————————————-

new May 2020

action: put-reviewB

purpose: add review

parameters:

“blocID”:	id of user	long

“date”:	date in bloc format	double
“advertID”:	id of Advert	long
“stars”:	number of stars	int
“text”:	text of review	string

optional parameters:

“spend”:	amount spent	string (valid number with 2 decimal places)

response: success or error

——————————————————————————————————————-

deprecated May 2020

action: put-push-token

purpose: add the user’s Firebase Cloud Messaging token

parameters:

“fbID”:	id of user	long
“fcmToken”:	the token	string

response: success or error

——————————————————————————————————————-

new May 2020

action: put-push-tokenB

purpose: add the user’s Firebase Cloud Messaging token

parameters:

“blocID”:	id of user	long
“fcmToken”:	the token	string

response: success or error

——————————————————————————————————————-

action: query-eventsToday

purpose: find events happening on the specified date.
returns at least one event if any happening on this date

returns a max of 50 events up to 100km distance (as of 13th Feb 2019)

parameters:

“lat”:	latitude of centre of search	double
“lng”:	longitude of centre of search	double
“date”:	date in bloc format	double

response: “result”: list of Events on success, error if fails

———————————————————————————————-

deprecated May 2020

action: query-firstInvite

purpose: is this the first time an invite has been used by this receiver (if so the sender will be rewarded)

parameters:

“sender”:	fbID of sender	long
“receiver”:	fbID of receiver	long

response: “result”: “first”=‘true’ if first time this receiver has used an invite

———————————————————————————————-

new May 2020

action: query-firstInviteB

purpose: is this the first time an invite has been used by this receiver (if so the sender will be rewarded). It is possible the invite that has caused this request was sent by an old version of the client in which case it will have a facebook id instead of a bloc id. One of either senderB or senderF must be present.

parameters:

“senderB”:	blocID of sender	long	*optional*
“senderF”:	facebook ID of sender	long	*optional*
“receiverB”:	blocID of receiver	long	

response: “result”: “first”=‘true’ if first time this receiver has used an invite

———————————————————————————————-

action: query-placeExists

purpose: returns the Bloc place matching the GMSPlace sent
creates it if it doesn’t already exist

parameters:

“gogl”:	google id	string
“name”:	google name	string
“type”:	types	array of string
“lat”:	latitude of centre of search	double
“lng”:	longitude of centre of search	double
“addr-comps”:	address components	dictionary/map
“locality”	locality	string
“country”	country	string
“city”	city	string
“phone”:	phone number	string
“webs”:	website	string
“addr”:	formatted address	string

response: “result”: Place on success, error if fails

——————————————————————————————————-

action: remove-checkReview

purpose: delete a CheckReview

parameters:

“checkID”:	id of CheckReview	long

response: success or error

——————————————————————————————————-

deprecated May 2020

action: reject-superlike

purpose: reject a superlike that has been sent
only updates the Person table for the recipient (fbID - user) of the superlike

parameters:

"fbID":	id of user	long
"from":	id of person that sent superlike	long

response: success or error

—————————————————————————————————————-

new May 2020

action: reject-superlikeB

purpose: reject a superlike that has been sent
only updates the Person table for the recipient (fbID - user) of the superlike

parameters:

"blocID":	id of user	long
"fromB":	id of person that sent superlike	long

response: success or error

—————————————————————————————————————-

deprecated May 2020

action: remove-fromEvent

purpose: remove the user from an event

parameters:

"eventID":	id of Event	long
"fbID":	id of user	long

response: success or error

—————————————————————————————————————-

new May 2020

action: remove-fromEventB

purpose: remove the user from an event

parameters:

"eventID":	id of Event	long

“blocID”:	id of user	long

response: success or error

———————————————————————————————————————-

deprecated May 2020

action: remove-like

purpose: ‘unsend’ like

parameters:

“fbID”:	id of sending user	long
“to”:	id of liked user	long

response: success or error

———————————————————————————————————————-

new May 2020

action: remove-likeB

purpose: ‘unsend’ like

parameters:

“blocID”:	id of sending user	long
“toB”:	bloc id of liked user	long

response: success or error

———————————————————————————————————————-

deprecated May 2020

action: send-like

purpose: send a ‘like’ to another user. Currently a thumbs up in the UI

parameters:

“fbID”:	id of user sending like	long
“to”:	id of user to send like to	long

response: success or error

———————————————————————————————————————-

new May 2020

action: send-likeB

purpose: send a ‘like’ to another user. Currently a thumbs up in the UI

parameters:

“blocID”:	bloc id of user sending like	long
“toB”:	bloc id of user to send like to	long

response: success or error

—————————————————————————————————————-

deprecated May 2020

action: send-message

purpose: send a Firebase Cloud Messaging message, parameters are dependent on the type of message being sent

parameters:

“pmtik”:	type of message	int
“inviteKey”:	id of inviter	long
“rcvr”:	id of receiver	long
“superLikerKey”:	id of user	long
“badge”:	badge number	int
“accepterKey”:	id of user	long
“inviteUsedByKey”:	id of user	long
“titl”:	message title	string
“msg”:	message body	string
“SnNm”:	sender’s name	string
“date”:	date in bloc format	double
“news”:	newsflash id	long

response: success or error

—————————————————————————————————————-

new May 2020

action: send-messageB

purpose:	send a Firebase Cloud Messaging message, parameters are dependent on the type of message being sent

parameters:

"pmtik":	type of message	int
"inviteB":	bloc id of inviter	long
"rcvrB":	bloc id of receiver	long
"sendB":	bloc id of sender	long
"superLikerB":	bloc id of user	long
"badge":	badge number	int
"accepterB":	bloc id of user	long
"inviteUsedByB":	bloc id of user	long
"titl":	message title	string
"msg":	message body	string
"SnNm":	sender's name	string
"date":	date in bloc format	double
"news":	newsflash id	long

response:	success or error

———————————————————————————————————-

deprecated May 2020

action:	set-chatSeen

purpose:	flag that this chat message has been viewed by the receiver

parameters:

"send":	id of sender	long
"rcvr":	id of receiver	long
"date":	date	double
"text":	text	string

response:	success or error

———————————————————————————————————-

new May 2020

action:	set-chatSeenB

purpose:	flag that this chat message has been viewed by the receiver

parameters:

“sendB”:	bloc id of sender	long
“rcvrB”:	bloc id of receiver	long
“date”:	date	double
“text”:	text	string

response: success or error

———————————————————————————————————————-

deprecated May 2020

action: update-personFriends

purpose: update the user’s friends

parameters:

“fbID”:	id of user	long
“fnds”:	list of friends	array of longs

response: success or error

———————————————————————————————————————-

new May 2020

action: update-personFriendsB

purpose: update the user’s friends

parameters:

“blocID”:	id of user	long
“fndsB”:	list of friends	array of longs

response: success or error

———————————————————————————————————————-

deprecated May 2020

action: update-personImageFlags

purpose: update the user’s image flags

parameters:

“fbID”:	id of user	long

“cImg”:	image flags	int

response: success or error

——————————————————————————————————-

new May 2020

action: update-personImageFlagsB

purpose: update the user’s image flags

parameters:

“blocID”:	id of user	long
“cImg”:	image flags	int

response: success or error

——————————————————————————————————-

deprecated May 2020

action: update-personStoreInfo

purpose: update the user’s store info (subscription on/off, months left, expiry date)

parameters:

“fbID”:	id of user	long
“sub”:	currently subscribed to Bloc Plus (“Y” or “N”)	string
“mths”:	number of months left	int
“expS”:	date of expiry in bloc format	double
“nxtT”:	date of next tokens in bloc format	double

response: success or error

——————————————————————————————————-

new May 2020

action: update-personStoreInfoB

purpose: update the user’s store info (subscription on/off, months left, expiry date)

parameters:

“blocID”:	id of user	long

“sub”:	currently subscribed to Bloc Plus (“Y” or “N”)	string
“mths”:	number of months left	int
“expS”:	date of expiry in bloc format	double
“nxtT”:	date of next tokens in bloc format	double

response: success or error

————————————————————————————————————-

deprecated May 2020

action: update-personUserFlags

purpose: update the user’s flags

parameters:

“fbID”:	id of user	long
“flgs”:	flags	long

response: success or error

————————————————————————————————————-

new May 2020

action: update-personUserFlagsB

purpose: update the user’s flags

parameters:

“blocID”:	id of user	long
“flgs”:	flags	long

response: success or error

————————————————————————————————————-

deprecated May 2020

action: update-personInfo

purpose: update one or more of the users info fields

parameters:

“fbID”:	id of user	long

optional parameters:

“abot”:	About text	string
“job”:	Job	string
“mplr”:	Employer	string
“scol”:	School	string
“loc”:	Location	string
“drnk”:	Drink	string
“smk”:	Smoke	string
“bmth”:	Birthday Month	string
“trik”:	Trick	string
“gnre”:	Music Genre	string
“trak”:	Track	string
“powr”:	Power	string
“bket”:	Bucket list	string

response: success or error

—————————————————————————————————————-

new May 2020

action: update-personInfoB

purpose: update the users info fields
any omitted info fields will be cleared down

parameters:

“blocID”:	id of user	long

optional parameters:

“abot”:	About text	string
“job”:	Job	string
“mplr”:	Employer	string
“scol”:	School	string
“loc”:	Location	string
“drnk”:	Drink	string
“smk”:	Smoke	string
“bmth”:	Birthday Month	string
“trik”:	Trick	string
“gnre”:	Music Genre	string
“trak”:	Track	string

“powr”:	Power	string
“bket”:	Bucket list	string

response: success or error

———————————————————————————————-

new Feb 2022

action: update-personEmailB

purpose: update the users email address

parameters:

“blocID”:	id of user	long
“email”:	email address	string

response: success or error

———————————————————————————————-

deprecated May 2020

action: zero-badge

purpose: set user’s badge to zero

parameters:

“fbID”:	id of user	long

response: success or error

———————————————————————————————-

new May 2020

action: zero-badgeB

purpose: set user’s badge to zero

parameters:

“blocID”:	id of user	long

response: success or error

Original Server Requests

These requests are those in the ‘original’ server - ie predated Autumn 2018. Apart from the need to add the sender info (documented here) they remain compatible.

action: query-events

purpose: find events happening on the specified date within a radius in metres of a point. The type is optional, Events for all types of Place are returned if type omitted. iOS specifies either “bar” or “night_club” for new explore screen

parameters:

“type”:	type of Place (optional)	string
“lat”:	latitude of centre of search	double
“lng”:	longitude of centre of search	double
“date”:	date in bloc format	double
“radiusInMeter”:	radius of search area	double

response: “result”: list of venues on success, error if fails

———————————————————————————————————-

action: query-people2

purpose: find the specified number of people going to events near the specified location between (inclusive) the specified dates. A list of people to not return can be specified (typically these people will have been returned by previous query-people2 requests).

parameters:

“lat”:	latitude of centre of search	double
“lng”:	longitude of centre of search	double
“startDate”:	start date in bloc format	double
“endDate”:	start date in bloc format	double
“number”:	number of people to return	double
“sentList”:	array of blocIDs to not return	array of longs
“imageSize”:	width of images needed	int
“filter”:	users filter preferences (M = males, F = females, A = All)	string

response: “result”: list of PersonInfos (in “people”:) & list of EventInfos (in “events”:) on success, error if fails

————————————————————————————————————-

action: query-events-rectangle

no longer supported

purpose: find events happening on the specified date within a rectangle. The type is optional, Events for all types of Place are returned if type omitted. iOS specifies either “bar” or “night_club” for new explore screen

parameters:

“type”:	type of Place (optional)	string
“minLat”:	minimum latitude of search	double
“maxLat”:	minimum latitude of search	double
“minLng”:	minimum longitude of search	double
“minLng”:	minimum longitude of search	double
“date”:	date in bloc format	double

response: “result”: list of venues on success, error if fails

————————————————————————————————————-

action: put-point

no longer supported

purpose: all Places referenced by Bloc must have valid geo location data – this is in a separate table. If a new Place is added to the database, or an old Place is used that does not have the ‘geo’ DynamoDB attribute set to Y, then this action should be requested and the ‘geo’ attribute set to ‘Y’;

parameters:

“blocID”:	Place ID	long
“lat”:	latitude of Place	double
“lng”:	longitude of Place	double

response: success or error

————————————————————————————————————-

action: query-special

purpose: find specials happening on and after the specified date (usually today) in the specified country. Country is currently usually “United Kingdom”

parameters:

“country”:	country (as per Place address from Google Places)	string
“date”:	date in bloc format	double

response: “result”: list of specials on success, error if fails

———————————————————————————————————————-

action: put-special

iOS management tool only

———————————————————————————————————————-

action: put-event

purpose: compute geo-location data and write Event info to DynamoDB.

parameters:

“placeID”:	Place ID	long
“uniqueID”:	Event ID	long
“lat”:	latitude of Place	double
“lng”:	longitude of Place	double
“date”:	date in bloc format	double
“fbID”:	fbID of user	long *deprecated May 2020*
“blocID”:	blocID of user	long *new May 2020*
“type”:	type of Event (“Out”)	string

response: success or error if fails

———————————————————————————————————————-

action: query-adverts

purpose: get a list of adverts to show.

parameters:

“lat”:	latitude of user	double
“lng”:	longitude of user	double
“date”:	current date in bloc format	double

response: “result”: list of adverts on success, error if fails
———————————————————————————————————-

deprecated May 2020
action: put-analytics

purpose: add analytics info.

parameters:

“userID”:	fbID of user	long
“type”:	analytics type	string

where type is one of: “checkin”, “match”, “likeSent”, “superLikeSent”, “sub1”, “sub6”, “sub12”, “buy5”, “buy25”, “buy60”, “buy100”, “earnInvite”, “earnFacebook”, “earnTwitter” or “earnInstagram”. Each analytics type indicates the user has just performed that action (The first 4 are self explanatory, the sub types are for subscriptions of the various lengths, the buy types are on buying that number of super likes and the earn types are when the user earns super likes by performing the appropriate social media action.)

response: error if fails
———————————————————————————————————-

new May 2020
action: put-analyticsB

purpose: add analytics info.

parameters:

“blocID”:	bloc ID of user	long
“type”:	analytics type	string

where type is one of: “checkin”, “match”, “likeSent”, “superLikeSent”, “sub1”, “sub6”, “sub12”, “buy5”, “buy25”, “buy60”, “buy100”, “earnInvite”, “earnFacebook”, “earnTwitter” or “earnInstagram”. Each analytics type indicates the user has just performed that action (The first 4 are self explanatory, the sub types are for subscriptions of the various lengths, the buy types are on buying that number of super likes and the earnXxxxxx types are when the user earns super likes by performing the appropriate social media action.)

response:	error if fails

——————————————————————————————————————-

action:	put-advertAnalytics	
purpose:	add adverts analytics info.	
parameters:		
“advertID”:	advert id	long
“impressions”:	number of times this advert has been drawn	int
“checkins”:	number of times the event associated with this advert has been checked into	int
“urltaps”:	number of times the url in this advert has been tapped	int
response:	error if fails	

——————————————————————————————————————-

new April 2021

action:	get-bufferedMessages	
purpose:	check whether there are any notifications that this user has missed	
parameters:		
“blocID”:	bloc ID of user	long
“since”:	Bloc Date	double
response:	error if fails, “messages’ (array of Messages) if successful	

——————————————————————————————————————-

new April 2021

action: get-newsFlashesAllB

purpose: get the newsflashes for a user and the newsflashes for all their friends (server loops on all friends for this user - quicker than looping in app)

parameters:

"blocID":	id of user	long
"start":	start Bloc Date	double
"end":	end Bloc Date	double

response: "result": list of Newsflashes on success, error if fails

——————————————————————————————————-

new April 2021

action: incorrectCoordinates

purpose: allows user to specify a place has wrong coordinates.
the server sends chris and josh an email with this info

parameters:

"blocID":	id of user making correction	long
"id":	place	long
"lat":	correct latitude of Place	double
"lng":	correct longitude of Place	double

response: error if fails

Phone Number Login and Signup API

Introduced August 2020 to allow registering for a Bloc account with phone number and to login to an existing ('phone number') Bloc account. There is no crossover between facebook and 'phone number' accounts. 'Phone number' accounts are identified by their email address. The phone number is only used for authorisation.

action: signup

purpose: create a new account

parameters:

"firstName" :	first name	string
"lastName" :	last name	string
"email" :	email	string
"password" :	password	string (optional)
"mobile" :	mobile	string (optional)
"dob" :	date of birth in bloc format	double
"gender" :	gender (M,F or O)	string
"appleID" :	apple id token	string (optional)

response: "result": Person on success, error if fails.

————————————————————————————————————-

action: login

purpose: login to an existing account
(note this does not login to facebook accounts)

parameters:

"email" :	email	string
"password" :	password	string

response: "result": Person on success, error if fails.

————————————————————————————————————-

action: reset-password

purpose: resets a forgotten password - sends a url in an email to the user

parameters:

"email" :	email	string

response: “result”: “OK” on success, error if fails.

loginApple

Apple Login

Introduced August 2020 to allow registering for a Bloc account with AppleID. Apple handle authentication and we just use the AppleID to identify our user. There is no crossover between facebook, ’phone number’ or ‘AppleID’ accounts. ‘AppleID’ accounts are identified by their appleID. This may be the same as their email address

action: loginApple

purpose: get a user’s account after they have logged in with AppleID

parameters:

"appleID" :	email	string

response: “result”: Person on success, error if fails.

Store Variables

Introduced November 2021 to support easy changing of amounts in store. Allows the numbers in Bloc Store to be modified without a new version of the app being required. Other variables are read from Apple’s Store

action: get-storeVariables *deprecated Nov 21*
action: get-storeVars

purpose: gets the Store Vars

parameters:
none

response: error if fails. “result’ with following keys if success

"type" :	single char defining order in display	string
"crypto" :	1 if exchanging for bloc$	string *optional*
"ukValue" :	description of what can be acquired	string
“number”:	amount of bloc stars required	string

Bloc Dollars

Introduced December 2021 to support the Bloc Dollars (Bloc$) cryptocurrency. Interfaces with Appycode’s Python Wallet Server

action: bloc$-changed

purpose: allows Python Wallet Server to tell us that a user’s bloc$ amount has changed

parameters:

"blocID" :	bloc ID of user	long
"bloc$" :	number of bloc$	string

response: “result”: “OK” on success, error if fails.

Index of Requests

get-image
get-imageB
get-matchEvent
get-matchEventB
get-matchList
get-matchListB
get-newsFlash
get-newsFlashes
get-newsFlashesAllB
get-newsFlashesB
get-person
get-personB
get-place
get-storeVars
get-uniquePlaceID
get-unique
get-userLikesSentToday
get-userLikesSentTodayB
get-version
inc-badge
inc-badgeB
incorrectCoordinates
login
loginApple
new-image
new-imageB
put-advertAnalytics
put-analytics
put-analyticsB
put-chat
put-chatB
put-checkReview
put-checkReviewB
put-event
put-match
put-matchB
put-matchEvent
put-matchEventB
put-newsflash
put-newsflashB

put-person
put-personB
put-place
put-push-token
put-push-tokenB
put-review
put-reviewB
put-special
query-adverts
query-events
query-eventsToday
query-firstInvite
query-firstInviteB
query-people2
query-placeExists
query-special
reject-superlike
reject-superlikeB
remove-checkReview
remove-fromEvent
remove-fromEventB
remove-like
remove-likeB
reset-password
send-like
send-likeB
send-message
send-messageB
send-superlike
send-superlikeB
set-chatSeen
set-chatSeenB
signup
submit_exchange
submit_exchangeB
update-personEmailB
update-personFriends
update-personFriendsB
update-personImageFlags
update-personImageFlagsB

update-personStoreInfo
update-personStoreInfoB
update-personUserFlags
update-personUserFlagsB
update-personInfo
update-personInfoB
zero-badge
zero-badgeB

Alterations

Error responses documented
Option for start / end day in get-earnChecks
Event in response has advertType not advert
result from get-MatchList and get-MatchEvent contain info inside sub dictionaries
userID missing from put-review
Newsflash event ID was incorrectly documented as ‘evID’ - it is ‘event’.
Documentation for query-eventsToday added.

21 April 2019
get-userLikesSentToday added
update-personStoreInfo added

15th May 2019
Typos in documentation corrected

18th May 2019
new-image added
get-currentPlace added
accept-superlike corrected

21st May 2019
delete-person documentation added

28th May 2019
add-ignorePlace added
imageSize parameter added to query-people2
Image Handling Programming Overview added
query-firstInvite ‘branchRef’ no longer used nor required

30th May 2019
delete-matchEvent typo modified

2nd June 2019
add-superlikeTokens modified to not add multiple sets of tokens if they already have received some for that day
added ‘nearby’ key to advert (info in response to query-adverts)

14th June 2019
reject-superlike added
get-image added

27th June 2019
query-placeExists documented

1st July 2019
query-placeExists add-comps documented
‘nxtT’ documentation added to update-personStoreInfo

18 Nov 2019
added spend to put-review

8 May 2020
Added Bloc ID based requests, parameters in requests and parameters in responses.
Old facebook id based requests are still supported but deprecated. Facebook IDs can be returned in responses for backward compatibility but can be ignored.
Including:
accept-superlikeB
add-ignorePlaceB
add-superlikeTokensB
add-toEventB
dec-badgeB
delete-chatsB
delete-matchB
delete-matchEventB
delete-personB
get-badgeB
get-chatListFromB
get-chatListToB
get-checkReviewsB
get-deletedPersonB
get-earnChecksB
get-event has blocIDs in returned event
get-eventDatePlace has blocIDs in returned event
get-imageB
get-matchEventB
get-matchListB has blocIDs in returned event (‘midsB’)
get-newsFlash has blocID in returned newsflash

get-newsFlashesB
get-personB
get-userLikesSentTodayB
inc-badgeB
new-imageB
put-analyticsB
put-chatB
put-checkReviewB
put-event blocID used if present
put-matchB
put-matchEventB
put-newsflashB
put-personB
put-push-tokenB
reject-superlikeB
query-firstInviteB
remove-likeB
send-likeB
send-messageB
send-superlikeB
set-chatSeenB
submit_exchangeB
update-personImageFlagsB
update-personStoreInfoB
update-personUserFlagsB
update-personInfoB
update-personEmailB
zero-badgeB

New request added get-unique

24 August 2020
login & signup added

26 September 2020
updated “sender” documentation
modified signup documentation

9 April 2021
query-people2 documentation updated (sentList now correctly described as list of blocIDs)
get-bufferedMessages added
get-newsFlashesAllB added

25 June 2021
send-MessageB updated to correctly show “sendB” parameter

14 Dec 2021
email added to get-PersonB response

17 Jan 2022
~~lastName added to get-PersonB response ?~~
mobile, blocDollars added to Person in get-PersonB response
correct endpoint https://server.getonbloc.com/BlocServletCurrent specified
bloc$-changed added
get-bufferedMessages documentation updated
get-newsFlashesAllB documentation updated
incorrectcoordinates documentation added
loginApple documentation added
reset-password added
submit_exchangeB updated - redeem_type added
get-storeVars documentation added

6 Feb 2022
update-personEmailB added