LINK Mobility MNP/HLR Lookup API

MNP/HLR Lookup API

Version 1.0.1; Last updated Sep 7, 2021

For help, see the following link https://linkmobility.com/support

Up-to-date version of this document is available at https://www.linkmobility.com/developer/

link mobility

Table of Contents

Changelog of this document

Date	Version	Author	Changes
2021-03-16	1.0	GJ	Initial version
2021-07-09	1.0.1	GJ	Amended error responses

Legal Information

Before you begin

This is the API used to receive MNP/HLR lookup information through Link Mobility's REST API or via a direct connection to the service. All users must first receive an account from LINK Mobility prior to connecting to this service. The LINK Mobility delivery team will manage the process.

Connection Points

There are four access points to the service via:

- API Gateway
- Direct TCP connection
- Direct UDP connection

Base URL for API Gateway connection

https://lookupauth.linkmobility.co.uk/oauth2/token - to obtain authentication token

https://lookup.linkmobility.co.uk/enumlookup/ - to lookup telephone number

Connection point for TCP direct connection

Address: dnsenum.linkmobility.co.uk

Port: 10054

Connection point for UDP direct connection

Address: dnsenum.linkmobility.co.uk

Port: 10055

Scope of this document

This document will describe the various ways to access the service, including Application Programming Interface (API) and direct connection methods to receive MNP/HLR lookup information through the Link Mobility platform.

The API Gateway is a REST API. A familiarity with REST APIs is assumed. Requests will be sent in JSON format with the response returned in JSON format. A basic familiarity with JSON is assumed.

Direct connections are TCP or UDP based. A basic familiarity with programming with socket connections is assumed. Requests and responses via these connections are also in a JSON format.

Capabilities of the platform

The platform is a high-capacity, high-availability MNP/HLR lookup service designed to let you request lookup information on a telephone number (MSISDN). Message contents detailed below.

Important connection notes

The API Gateway uses a REST API connection with a Oauth2.0 token serving as authentication. You must request and receive a Oauth2.0 token, using this in the header request of the lookup, to receive information on an MSISDN. Failure to do so will result in a 401 (forbidden response) from the service.

All other connection routes: TCP, UDP are available to whitelisted IP ranges with a username/password sent in request to ensure only valid customers can access this service. Failure to add in username/password in a direct connection request will result in a 'failed' response being returned.

API Gateway

Oauth2.0 token validation

```
Python example

client_id     = '<to be given to client via LINK Support>'
client_secret = '<to be given to client via LINK Support>'
token_url     = 'https://lookupauth.linkmobility.co.uk/oauth2/token'
scope         = 'linklookup/lookup'

client = BackendApplicationClient(client_id=client_id)
oauth  = OAuth2Session(client=client, scope=scope)
token  = oauth.fetch_token(token_url=token_url, client_id=client_id, client_secret=client_secret)
```

Response

```
'token' will include the Oauth2.0 token and allow you to request lookups via the request below
```

SEND / enumlookup

Request

```
api_call_headers={'Authorization': 'Bearer ' + token['access_token']}

tel_lookup_url='https://lookup.linkmobility.co.uk/enumlookup?tel=<telnum>&lookup=<lookup type>'

response = requests.get(url=tel_lookup_url, headers=api_call_headers, verify=True)
```

Parameter	Value	Type	Required	Description
tel	441234567890 (*example*)	string	Yes	MSISDN you wish to send the perform lookup on. Must be in fully qualified international dialling number, no + symbol.
lookup	mnp (*example*)	string	No	The type of lookup you wish to perform (mnp/hlr). If no lookup type is specified then the system will run a lookup dependant on your lookup profile. Normally this will be MNP, then if no information is found then a live HLR lookup will be performed.

Response

Successful response, authentication validated, request valid

Parameter	Sub-parameter	Value	Type	Description
statuscode		200	integer	
response	tel	441234567890 (*example*)	string	Confirmation of your requested telephone number lookup.

	dc	44 (example)	string	International dialling code of telephone number.
	mcc	234 (*example*)	string	MCC of the network (if mobile network)
	mnc	57 (*example*)	string	MNC of the network (if mobile network)
	cc	GB (*example*)	string	Country code acronym.
	ndc	1234 (*example*)	string	National dialling code.
	lookup	mnp (example)	string	Lookup type performed by LINK to get information on number (mnp/hlr)
	nt	mobile (example)	string	Number type (mobile/fixed)
	imsi	23457 (example)	string	Part of the IMSI.
	cic		string	Carrier Identification Code (if found)
	cn		string	Carrier short name (if found)
	npdi		string	NP Database Dip Indicator (indicates to any downstream systems a portability lookup has been performed)
	tadig		string	TADIG code
	roaming		string	Whether telephone number is roaming. 'yes', 'no' or 'na' (not available)
	rn		string	Routing Number (LRN for USA/Canada)
	rmcc		string	MCC of the roaming network
	rmnc		string	MNC of the roaming network
	np		string	Number Ported: 'yes', 'no' or 'na' (not available)
	ppcin		string	PPCIN code
	pres		string	Present: 'yes', 'no' or 'na' (not available)
	e		string	Error code if any, as an example may indicate invalid/absent subscriber in HLR lookup.

Client not allowed to use specific lookup type in JSON request

Parameter	Sub-parameter	Value	Type	Description
statuscode		200	integer	

response	tel	441234567890 (*example*)	string	Confirmation of your requested telephone number lookup.
	All other sub parameter values will be blank			
	e	F0	string	Error code 'F0' indicates client is not allowed to use this lookup type. E.g. if your account in MNP lookup only and you attempt an HLR lookup you will get this response in error parameter.

Client not allowed to lookup specific country in JSON request

Parameter	Sub-parameter	Value	Type	Description
statuscode		200	integer	
response	tel	441234567890 (*example*)	string	Confirmation of your requested telephone number lookup.
	All other sub parameter values will be blank			
	e	F1	string	Error code 'F1' indicates client is not allowed to lookup telephone numbers in this country. E.g. if your account only allows lookup to Germany and you request lookup to a UK number you will get this response in error parameter.

Invalid parameters in JSON request

Parameter	Sub-parameter	Value	Type	Description
statuscode		200	integer	
response	All other sub parameter values will be blank			
	e	F2	string	Error code 'F2' indicates client is not allowed to use this lookup type. E.g. if your account in MNP lookup only and you attempt an HLR lookup you will get this response in error parameter.

Unknown error

Parameter	Sub-parameter	Value	Type	Description
statuscode		200	integer	
response	All other sub parameter			

	values will be blank			
	e	F3	string	Error code 'F3' indicates an unknown error when looking up telephone number.

Invalid OAuth2.0 token sent in request

Parameter	Sub-parameter	Value	Type	Description
statuscode		401	integer	
response	message	Unauthorized	string	Request unauthorized, authentication invalid.

Direct connection

Parameters for direct connection to the lookup service differs slightly but follows the same methodology. It is important to note that your IP/IP range must be whitelisted before connection is allowed to the service. This will be done by LINK Support.

Once whitelisted a socket connection must be created (either TCP or UDP.)

After a successful socket connection the request must be in the following formatted string. Note: 'lookup' is an optional parameter:

```
{"clientid":"<LINK Support to supply to client>","password":"<LINK Support to supply to client>","tel":"<telnum>","lookup":"<lookup type>"}

e.g.

{"clientid":"XXX", "password":"YYY", "tel":"441234567890", "lookup":"hlr"}
```

Parameter	Value	Type	Required	Description
clientid	XXX	string	Yes	LINK Support to supply to client
password	YYY	string	Yes	LINK Support to supply to client
tel	441234567890 (*example*)	string	Yes	MSISDN you wish to send the perform lookup on. Must be in fully qualified international dialling number, no '+' symbol.
lookup	mnp (*example*)	string	No	The type of lookup you wish to perform (mnp/hlr). If no lookup type is specified then the system will run a lookup dependant on your lookup profile. Normally this will be MNP, then if no information is found then a live HLR lookup will be performed.

Response from the service will be in a JSON formatted string. The following is an example of the response you will receive from the service. Values will differ dependant on telephone number:

```
{"cc":"GBR","lookup":"hlr","np":"na","mnc":"30","roaming":"na","pres":"na","rmnc":"","tadig":"GBRME","e":"","nt":"mobile","cn":"EE-TM-23430","imsi":"23430","ndc":"7506","mcc":"234","rmcc":"","npdi":"","ppcin":"EE","tel":"441234567890","rn":"","cic":"44548","dc":"44"}
```

Parameters of the response will match parameters stated in API Gateway response.

Parameter	Sub-parameter	Value	Type	Description
response	tel	441234567890 (*example*)	string	Confirmation of your requested telephone number lookup.
	dc	44 (example)	string	International dialling code of telephone number.
	mcc	234	string	MCC of the network (if mobile network)

		(*example*)		
	mnc	57 (*example*)	string	MNC of the network (if mobile network)
	cc	GB (*example*)	string	Country code acronym.
	ndc	1234 (*example*)	string	National dialling code
	lookup	mnp (example)	string	Lookup type performed by LINK to get information on number (mnp/hlr.)
	nt	mobile (example)	string	Number type (mobile/fixed)
	imsi	23457 (example)	string	Part of the IMSI.
	cic		string	Carrier Identification Code (if found)
	cn		string	Carrier short name (if found)
	npdi		string	NP Database Dip Indicator (indicates to any downstream systems a portability lookup has been performed)
	tadig		string	TADIG code
	roaming		string	Whether telephone number is roaming. 'yes', 'no' or 'na' (not available)
	rn		string	Routing Number (LRN for USA/Canada)
	rmcc		string	MCC of the roaming network
	rmnc		string	MNC of the roaming network
	np		string	Number Ported: 'yes', 'no' or 'na' (not available)
	ppcin		string	PPCIN code
	pres		string	Present: 'yes', 'no' or 'na' (not available)
	e		string	Error code if any, as an example may indicate invalid/absent subscriber in HLR lookup.

Error codes

Error code may appear in ‘e’ response. These error codes are noted below.

Error code	Description
01	Unknown subscriber.
05	Unidentified subscriber.
09	Illegal subscriber.
1B	Absent subscriber.
F0	Client is not allowed to use this lookup type (e.g. only allowed MNP lookups, but request HLR).
F1	Client is not allowed to lookup telephone numbers in this country.
F2	Invalid parameters passed in request.
F3	Unknown error.

Examples

It is important to note these examples are for demonstration purposes only. When developing in a production environment all necessary error handling should be implemented to mitigate against error cases.

Python (using API Gateway)

Note: The following libraries must be installed via

- pip install requests
- pip install requests_oauthlib
- pip install oauthlib

```
# -*- coding: utf-8 -*-

# Python import libraries
import json
import time
import threading
import requests
import datetime
from oauthlib.oauth2 import BackendApplicationClient
from requests_oauthlib import OAuth2Session

# Setp client_id and secret for getting OAuth2
client_id = 'xxxx'      # client ID from LINK Support
client_secret = 'yyyy'  # client secret from LINK Support
token_url = 'https://lookupauth.linkmobility.co.uk/oauth2/token'
scope = 'linklookup/lookup'

# Fetch an access token
print("%s: Fetching token from: %s" % (str(datetime.datetime.now()), token_url))

client = None
oauth = None
token = None
api_call_headers = None

try:
    client = BackendApplicationClient(client_id=client_id)
    oauth = OAuth2Session(client=client, scope=scope)
    token = oauth.fetch_token(token_url=token_url,
                              client_id=client_id,
                              client_secret=client_secret)
    print("%s: Fetched token" % str(datetime.datetime.now()))
    api_call_headers = {'Authorization': 'Bearer ' + token['access_token']}

except Exception as err:
    print("Oauth2.0 token exception")
    exception_type = type(err).__name__
    print(exception_type)


def enum_lookup(api_hdr, tellist):
    try:
        lookup_url = 'https://lookup.linkmobility.co.uk/enumlookup?'
        print("%s: Started (%d #requests)" % (str(datetime.datetime.now()), len(tellist)))
        for telnum in tellist:
            tel_lookup_url = lookup_url + 'tel=' + telnum + '&lookup=mnp'
            print("%s: Request: %s" % (str(datetime.datetime.now()), tel_lookup_url))
            response = requests.get(url=tel_lookup_url, headers=api_call_headers, verify=True)
            json_response = response.json()
            if "tel" in json_response:
                print("Received tel" + json_response["tel"])
                print("%s: Response: %d - %s" % (str(datetime.datetime.now()), response.status_code,
response.json()))
            else:
                print("%s: Error: %d - %s" % (str(datetime.datetime.now()), response.status_code,
response.json()))
            time.sleep(0.005)
    except Exception as err:
        print("Lookup exception")
        exception_type = type(err).__name__
        print(exception_type)

    print("%s: Finished" % str(datetime.datetime.now()))
```

```
#Created the Threads
t1 = threading.Thread(target=enum_lookup, args=(api_call_headers, ['tel number']))

#Started the threads
t1.start()

#Joined the threads
t1.join()
```

Python (TCP direct)

```
import socket

# Setup variables
serverAddressPort = ("dnsenum.linkmobility.co.uk", 10054)
bufferSize = 512
received = None
data = "{\"clientid\":\"client_ID_from_LINK_Support\",\"password\":\"password_from_LINK_Support
\",\"tel\":\"tel_number\",\"lookup\":\"mnp\"}"

# Initialize a TCP client socket using SOCK_STREAM
tcp_client = socket.socket(family=socket.AF_INET, type=socket.SOCK_STREAM)

try:
    # Establish connection to TCP server and exchange data
    tcp_client.connect(serverAddressPort)
    tcp_client.sendall(data.encode())

    # Read data from the TCP server and close the connection
    received = tcp_client.recv(bufferSize)
finally:
    tcp_client.close()

print("Request: {}".format(data))
print("Response: {}".format(received.decode()))
```

Python (UDP direct)

```
import socket

# Setup variables
serverAddressPort = ("dnsenum.linkmobility.co.uk", 10055)
bufferSize = 512
received = None
data = "{\"clientid\":\"client_ID_from_LINK_Support\",\"password\":\"password_from_LINK_Support
\",\"tel\":\"tel_number\",\"lookup\":\"mnp\"}"

# Initialize a TCP client socket using SOCK_STREAM
udp_client = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
try:
    # Send to server using UDP socket
    udp_client.sendto(data.encode(),serverAddressPort)

    # Read data from the UDP server and close the connection
    received = udp_client.recvfrom(bufferSize)
finally:
    udp_client.close()

print("Request: {}".format(data))
print("Response: {}".format(received)[0])
```

C (TCP direct)

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>

#define MAX_BUFFER_SIZE 512

// socketCreate: Create a socket for server communication
short socketCreate(void) {
    short hSocket;
    printf("Create the socket\n");
    hSocket = socket(AF_INET, SOCK_STREAM, 0);
    return hSocket;
}

// socketConnect: Connect with server
int socketConnect(int hSocket) {
    int iRetval=-1;
    int ServerPort = 10054;    // Port
    struct sockaddr_in remote= {0};
    remote.sin_addr.s_addr = inet_addr("dnsenum.linkmobility.co.uk");    // Host
    remote.sin_family = AF_INET;
    remote.sin_port = htons(ServerPort);
    iRetval = connect(hSocket, (struct sockaddr *)&remote, sizeof(struct sockaddr_in));
    return iRetval;
}

// socketSend: Send the data to the server and set the timeout of 20 seconds
int socketSend(int hSocket,char* Rqst,short lenRqst) {
    int shortRetval = -1;
    struct timeval tv;
    tv.tv_sec = 20;    // 20 Secs Timeout
    tv.tv_usec = 0;

    if(setsockopt(hSocket, SOL_SOCKET, SO_SNDTIMEO, (char *)&tv,sizeof(tv)) < 0) {
        printf("Time Out\n");
        return -1;
    }

    shortRetval = send(hSocket, Rqst, lenRqst, 0);
    return shortRetval;
}

// socketReceive: Receive the data from the server
int socketReceive(int hSocket,char* Rsp,short RvcSize) {
    int shortRetval = -1;
    struct timeval tv;
    tv.tv_sec = 20;  // 20 Secs Timeout
    tv.tv_usec = 0;

    // Set socket options
    if(setsockopt(hSocket, SOL_SOCKET, SO_RCVTIMEO, (char *)&tv,sizeof(tv)) < 0) {
        printf("Time Out\n");
        return -1;
    }

    shortRetval = recv(hSocket, Rsp, RvcSize, 0);    // Receive socket data
    Rsp[shortRetval] = '\0';    // Ensure null terminate response

    return shortRetval;
}

int main() {
    // Declare variables
```

```
    int hSocket, read_size;
    struct sockaddr_in server;
    char request[MAX_BUFFER_SIZE] = {0};
    char reply[MAX_BUFFER_SIZE] = {0};

        // Create socket
    hSocket = socketCreate();
    if(hSocket == -1) {
        printf("Unable to create socket\n");
        return 1;
    }

    // Connect to server
    if (socketConnect(hSocket) < 0) {
        perror("Unable to connect to socket\n");
        return 1;
    }

    // Create request
    strcpy(request, "{\"clientid\":\"client_ID_from_LINK_Support\",\"password\":\"
password_from_LINK_Support \",\"tel\":\"tel_number\",\"lookup\":\"mnp\"}");
    printf("Request: %s\n", request);

    //Send data to the server
    socketSend(hSocket, request, strlen(request));

    //Received the data from the server
    read_size = socketReceive(hSocket, reply, MAX_BUFFER_SIZE);
    printf("Response: %s\n", reply);

    // Tidy up
    close(hSocket);
    shutdown(hSocket,0);
    shutdown(hSocket,1);
    shutdown(hSocket,2);
    return 0;
}
```

C (UDP direct)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define MAX_BUFFER_SIZE 512

int main() {
    // Declare variables
    int hSocket, read_size;
    struct sockaddr_in servaddr;
        char request[MAX_BUFFER_SIZE] = {0};
    char reply[MAX_BUFFER_SIZE] = {0};

    // Creating socket file descriptor
    if ( (hSocket = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));

    // Filling server information
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(10055);
    servaddr.sin_addr.s_addr = inet_addr("dnsenum.linkmobility.co.uk");

    // Create request
    strcpy(request, "{\"clientid\":\"client_ID_from_LINK_Support\",\"password\":\"
password_from_LINK_Support \",\"tel\":\"tel_number\",\"lookup\":\"mnp\"}");

    // Send request
    printf("Request: %s\n", request);
    sendto(hSocket, (const char *)request, strlen(request), 0, (const struct sockaddr *) &servaddr,
sizeof(servaddr));

    // Receive response
    read_size = recv(hSocket, (char *)reply, MAX_BUFFER_SIZE, 0);
    reply[read_size] = '\0';

    printf("Response : %s\n", reply);

    close(hSocket);
    return 0;
}
```

Java (TCP direct)

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

public void main() {
    String host = 'dnsenum.linkmobility.co.uk';
    int port = 10054;
    String resp = "";
    String totalresp = "";

    try (Socket socket = new Socket(host, port)) {
        OutputStream output = socket.getOutputStream();
        PrintWriter writer = new PrintWriter(output, true);

        String request = '{"clientid":"clientid from LINK Suport","password":"password from LINK
Support","tel":"telephone number","lookup":"lookup type (optional)"}';

        writer.println(request);

        BufferedReader bufreader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

        while (resp != null && resp.equals("bye") == false) {
            resp = bufreader.readLine();

            if (resp != null) {
                totalresp += resp;
            }
        }

        System.out.println('Data: ' + totalresp);

        bufreader.close();
        writer.close();
        output.close();
        socket.close();
    } catch (UnknownHostException ex) {
        Logger.getLogger(this.getName()).log(Level.SEVERE, null, ex);
    } catch (SocketException ex) {
        Logger.getLogger(this.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(this.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Java (UDP direct)

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

public void main() {
    try {
        String host = 'dnsenum.linkmobility.co.uk';
        int port = 10055;

        DatagramSocket clientSocket = new DatagramSocket();

        InetAddress address = InetAddress.getByName(host);

        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];

        String request = '{"clientid":"clientid from LINK Suport","password":"password from LINK
Support","tel":"telephone number","lookup":"lookup type (optional)"}';

        sendData = request.getBytes();

        DatagramPacket sendPacket = new DatagramPacket(request.getBytes(),
                                                       request.getBytes().length,
                                                       address,
                                                       port);

        clientSocket.send(sendPacket);

        DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

        clientSocket.receive(receivePacket);

        String resp = new String(receivePacket.getData());

        System.out.println('Received ' + resp.replace("\n", ""));

        clientSocket.close();
    } catch (UnknownHostException ex) {
        Logger.getLogger(this.getName()).log(Level.SEVERE, null, ex);
    } catch (SocketException ex) {
        Logger.getLogger(this.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(this.getName()).log(Level.SEVERE, null, ex);
    }
}
```