GETTING STARTED WITH MOBILEDGEX SDK FOR YOUR IOS APPLICATION

Are you excited to connect to MobiledgeX Cloudlet Infrastructure and leverage the power that Mobile Edge Cloud offers? To do this, let’s integrate MobiledgeX SDK with your existing application, which exposes various services that MobiledgeX offers such as finding the nearest MobiledgeX Cloudlet Infrastructure for client-server communication or workload processing offload.

This Quickstart will teach you how to use the communications using the MobiledgeX MatchingEngine Library for our Communications API.

In this Quickstart, you will learn how to:

• Sign up for MobiledgeX and deploy your application to the cloudlets

• Set up your development environment to communicate with your cloudlets

• Request permission

• Your first API call by creating a request and register as a client

• Verify client location and find the nearest available cloudlet

• Start your first client-server communication

Sign up for MobiledgeX and deploy your application to the cloudlets

If you already have a MobiledgeX account and your application has been deployed, you are all set here. Go ahead and jump to next section.

If you already have a MobiledgeX account and your application has been deployed, you are all set here. Go ahead and jump to next section.

Before you can start using cloudlets, you will need to sign up for a MobiledgeX account.

Table 1 General Information

\*Mandatory to fill

Company Name\*

Developer Name\*

Application Name\*

Email\*

Address

Cloudlet Locations

Table 2 Client-Side Information

\*Mandatory to fill

Client-Side Operating System\*

IOS

Client-Side Programming Language\*

Swift

Client-Side IDE\*

Xcode

Table 3 Server-Side Deployment Information

\*Mandatory to fill

Application Type (Dockerized Container or VM)\*

Preferred Dockerized Registry\*

registry.mobiledgex.net:5000/*<developer name>/<application name>*

Minimum HW Requirement

GPU Requirement\*

Communication Protocol

Preferred Port Range (if applicable)

Config Map

i.e. Docker Manifest File

Assuming the server-side application is based on Docker, we will create you an account and send you a welcome email that includes instruction to use the MobiledgeX Docker registry to push your dockerized application using your Docker username. Please note that we create the Docker repository based on the Developer and Application Names provided in Table 1.

*docker login registry.mobiledgex.net:5000*

*Login using your docker username/pwd*

*docker push <developer name>/<application name>/<latest>*

*docker login registry.mobiledgex.net:5000*

*Login using your docker username/pwd*

*docker push <developer name>/<application name>/<latest>*

Once you have pushed your application, MobiledgeX will pull and deploy the application to the desired cloudlets. An email will be sent that contains detailed information about your deployed instances.

Table 4 Your Deployed Instances

Instance ID

5011

Cloudlet Name

Bonn

Operator Name

TDG

Instance State

AppStateReady

Access URI

<application>.bonn.TDG.mobiledgex.net

Now that your application instances have been deployed, you can start integrating MobiledgeX’s SDK to your IOS application!

To make things even easier, we will first set up your IOS Xcode environment by downloading MobiledgeX’s Demo App and SDK.

The demo app should build and run for you after you do

pod install

Optional: Launch NSLogger.app to see logged examples of all API commands when demo is run.

then just launch the project work space: MobiledgeXSDKDemo.IOS.xcworkspace and run

**MobiledgeX.com IOS SDK dependencies**

To use the SDK in your app.

Add the following to your project pod file or add Alamofire to your project directly

pod 'Alamofire'

pod 'NSLogger/Swift'

Then do the usual “pod install”

Add the following to your project

https://github.com/FutureKit/FutureKit

**Your first API call by creating a request and register as a client**

Now that we finally have dependencies out of the way, it is time to write some code by first creating a request and using that request to send an outbound REST API call to register as a client. Please type or paste in the code sample below. The *createRequest* API taking call to the SDK gathers specific information about the application, carrier information that the mobile device is running on. Using this information, we can use the *registerClient* API call to register the device to MobiledgeX’s cloudlets.

**RegisterClient as Curl**

curl -d '{"ver": 1, "AppName": “YourAppNameHere”, "DevName": "YourDevNameHere", "AppVers": "1.0"}' https://tdg2.dme.mobiledgex.net:38001/v1/registerclient -H "Content-Type: application/json" -v -X POST --cacert mex-ca.crt --key mex-client.key --cert mex-client.crt

**Reply json:**

["Status": RS\_SUCCESS,

"SessionCookie": eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1NDY5OTExODEsImlhdCI6MTU0NjkwNDc4MSwia2V5Ijp7InBlZXJpcCI6IjEyNy4wLjAuMSIsImRldm5hbWUiOiJNb2JpbGVkZ2VYIFNESyBEZW1vIiwiYXBwbmFtZSI6Ik1vYmlsZWRnZVggU0RLIERlbW8iLCJhcHB2ZXJzIjoiMS4wIiwia2lkIjo2fX0.Iq1YiaYC8H3UzPdkgqzOtx1XsdKW2T\_JkEM3\_7ssaOCWBWmdFnSmnasX2A4RfJfeSixbHEIn\_2aPY4JBWnGY1Q,

"TokenServerURI": http://mextest.tok.mobiledgex.net:9999/its?followURL=https://dme.mobiledgex.net/verifyLoc,

"Ver": 0]

Reply in swift

The swift json/Dictionary reply has 2 items of interest: sessioncookie and tokenserveruri

sessioncookie = reply["SessionCookie"] as! String

tokenserveruri = reply["TokenServerURI"] as! String

Above, you already dealt with registration of your application name or your developer name.

Step 1: Build request

let request = MexSDK.shared.createRegisterClientRequest( ver: “1”,

appName: ”Your application name”, // Your application name

devName: "Your developer name”, // Your developer name

appVers: ”1.0”

)

Step 2: Build url/api

let baseuri = MexUtil.shared.generateBaseUri(

MexUtil.shared.getCarrierName(), MexUtil.shared.dmePort)

let urlStr = baseUri + "/v1/registerclient"

Step 3: Post request

// postRequest() calls dealWithTrustPolicy() which handles certificates/trust before post

// you will need to add the certificates in .der form to your app.

future = MexSDK.shared.postRequest(urlStr, request)

future!.on(

success:

{

let reply = $0 as [String: Any] // Dictionary/json todo we need the reply documented somewhere

// save sessioncookie and tokenserveruri for later use

sessioncookie = reply["SessionCookie"] as! String

tokenserveruri = reply["TokenServerURI"] as! String

},

failure: { print("RegisterClient failed with error: \($0)") },

completion: { let \_ = $0 }

)

**Verify client location and find the nearest available cloudlet**

Once the device has been successfully registered,

the verifyLocation API can be used to verify the user’s current location and

the findCloudlet API can be used to find the nearest available cloudlet that has your server application running.

The findCloudlet API returns the URI (FQDN) of the application that can be used to form a client-server communication.

A detailed flowchart of these APIs can be found in our MEX-Developers-API document.

**FindCloudlet** CURL:

curl -d '{"ver": 1, "SessionCookie": "XYZ", "CarrierName": "TDG", "GpsLocation": {"lat": 37.0, "long": 137.2}}' https://tdg2.dme.mobiledgex.net:38001/v1/findcloudlet -H "Content-Type: application/json" -v -X POST --cacert mex-ca.crt --key mex-client.key --cert mex-client.crt

**Reply json**:

["FQDN": mobiledgexsdkdemomobiledgexsdkdemo10.mexdemo-centralus-cloudlet.azure.mobiledgex.net, "cloudlet\_location": {

altitude = 0;

course = 0;

"horizontal\_accuracy" = 0;

latitude = "41.878";

longitude = "-93.0977";

speed = 0;

timestamp = "<null>";

"vertical\_accuracy" = 0;

}, "status": FIND\_FOUND, "Ver": 0, "ports": <\_\_NSSingleObjectArrayI 0x600003b7c0d0>(

{

"FQDN\_prefix" = "mobiledgexsdkdemo-tcp.";

"internal\_port" = 7777;

proto = LProtoTCP;

"public\_path" = "";

"public\_port" = 7777;

}

)

]

**Reply in swift**

The swift json/Dictionary reply has 1 item of interest FQDN:

theFQDN = reply["FQDN"] as! String

**FindCloudlet** swift:

MobiledgeX.findNearestAvailableCloudlet()

let baseUri = MexUtil.shared.generateBaseUri(MexUtil.shared.getCarrierName(), MexUtil.shared.dmePort)

let urlStr = baseUri + "/v1/findcloudlet" // API

let loc = retrieveLocation()

let findCloudletRequest = createFindCloudletRequest( carrier, loc)

MexRegisterClient.shared.future = MexSDK.shared.postRequest(urlStr, findCloudletRequest)

MexRegisterClient.shared.future!.on(

success:

{

let reply = $0 as [String: Any] // where is reply documented? todo

let nearestCloudletFQDN = reply["FQDN"] as! String

},

failure: { print("findNearestAvailableCloudlet failed with error: \($0)") },

completion: { \_ = $0 // print("completed with result: \($0)")

}

)

**Verify location**

**VerifyLocation** CURL:

curl -d '{"ver": 1, "SessionCookie": "XYZ", "CarrierName": "TDG", "GpsLocation": {"lat": 37.0, "long": 137.2}, "VerifyLocToken": "abcdeff"}' https://tdg2.dme.mobiledgex.net:38001/v1/verifylocation -H "Content-Type: application/json" -v -X POST --cacert mex-ca.crt --key mex-client.key --cert mex-client.crt

**Reply**:

["tower\_status": TOWER\_UNKNOWN,

"GPS\_Location\_Accuracy\_KM": -1,

"ver": 0,

"gps\_location\_status": LOC\_ROAMING\_COUNTRY\_MATCH]

**Reply in swift**

**VerifyLocation swift**

let loc = retrieveLocation()

future = MexSDK.shared.postRequest(MexRegisterClient.shared.tokenserveruri, [String: Any](), "GetToken") // async

// NOTE special case: "GetToken" fails and its error result is parsed

future!.on(

success:

{

print("GetToken received value: \($0)")

let d = $0 as [String: Any]

let tokenReply = d["token"] as! String

var verifyLocationRequest = createVerifyLocationRequest(MexUtil.shared.carrierNameInUse, loc, "")

var tokenizedRequest = [String: Any]() // Dictionary/json

tokenizedRequest += verifyLocationRequest // Dictionary/json

tokenizedRequest["VerifyLocToken"] = tokenReply

let baseUri = MexUtil.shared.generateBaseUri(MexUtil.shared.carrierNameInUse, MexUtil.shared.dmePort)

let urlStr = baseUri + "/v1/verifylocation"

self.future = MexSDK.shared.postRequest(uri, tokenizedRequest)

self.future!.on(success: { print("Verifylocation received value: \($0)")

let d = $0 as [String: Any]

},

failure: { print("Verifylocation failed with error: \($0)")

},

completion: { let \_ = $0 //print("completed with result: \($0)")

})

},

failure: { print("GetToken failed with error: \($0)") },

completion: { \_ = $0 // print("completed with result: \($0)")

}

)

**~~~**

**GetInstalledApps**

**Getappinstlist** CURL:

curl -d '{"ver": 1, "SessionCookie": "XYZ", "CarrierName": "TDG", "GpsLocation": {"lat": 37.0, "long": 137.2}, "VerifyLocToken": "abcdeff"}' https://tdg2.dme.mobiledgex.net:38001/v1/getappinstlist -H "Content-Type: application/json" -v -X POST --cacert mex-ca.crt --key mex-client.key --cert mex-client.crt

**Reply json**:

["Status": AI\_SUCCESS, "Ver": 0, "Cloudlets": <\_\_NSArrayI 0x600002ce3ec0>(

{

Appinstances = (

{

AppName = "MobiledgeX SDK Demo";

AppVers = "1.0";

FQDN = "munich-mexdemo.tdg.mobiledgex.net";

ports = (

{

"FQDN\_prefix" = "mobiledgexsdkdemo-tcp.";

"internal\_port" = 7777;

proto = LProtoTCP;

"public\_path" = "";

"public\_port" = 7777;

}

);

}

);

CarrierName = TDG;

CloudletName = "munich-mexdemo";

Distance = "18387.77976411593";

GpsLocation = {

altitude = 0;

course = 0;

"horizontal\_accuracy" = 0;

latitude = "52.52";

longitude = "13.405";

speed = 0;

timestamp = {

nanos = 0;

seconds = 0;

};

"vertical\_accuracy" = 0;

};

},

{

Appinstances = (

{

AppName = "MobiledgeX SDK Demo";

AppVers = "1.0";

FQDN = "mobiledgexsdkdemomobiledgexsdkdemo10.mexdemo-centralus-cloudlet.azure.mobiledgex.net";

ports = (

{

"FQDN\_prefix" = "mobiledgexsdkdemo-tcp.";

"internal\_port" = 7777;

proto = LProtoTCP;

"public\_path" = "";

"public\_port" = 7777;

}

);

}

);

CarrierName = azure;

CloudletName = "mexdemo-centralus-cloudlet";

Distance = "11999.37221792214";

GpsLocation = {

altitude = 0;

course = 0;

"horizontal\_accuracy" = 0;

latitude = "41.878";

longitude = "-93.0977";

speed = 0;

timestamp = "<null>";

"vertical\_accuracy" = 0;

};

},

{

Appinstances = (

{

AppName = "MobiledgeX SDK Demo";

AppVers = "1.0";

FQDN = "mobiledgexsdkdemomobiledgexsdkdemo10.mexdemo-westindia-cloudlet.azure.mobiledgex.net";

ports = (

{

"FQDN\_prefix" = "mobiledgexsdkdemo-tcp.";

"internal\_port" = 7777;

proto = LProtoTCP;

"public\_path" = "";

"public\_port" = 7777;

}

);

}

);

CarrierName = azure;

CloudletName = "mexdemo-westindia-cloudlet";

Distance = "14828.68853606754";

GpsLocation = {

altitude = 0;

course = 0;

"horizontal\_accuracy" = 0;

latitude = "19.076";

longitude = "72.8777";

speed = 0;

timestamp = "<null>";

"vertical\_accuracy" = 0;

};

},

{

Appinstances = (

{

AppName = "MobiledgeX SDK Demo";

AppVers = "1.0";

FQDN = "mexdemo-krakow-cloudlet.tdg.mobiledgex.net";

ports = (

{

"FQDN\_prefix" = "mobiledgexsdkdemo-tcp.";

"internal\_port" = 7777;

proto = LProtoTCP;

"public\_path" = "";

"public\_port" = 7777;

}

);

}

);

CarrierName = TDG;

CloudletName = "mexdemo-krakow-cloudlet";

Distance = "18585.93574645938";

GpsLocation = {

altitude = 0;

course = 0;

"horizontal\_accuracy" = 0;

latitude = "50.0647";

longitude = "19.945";

speed = 0;

timestamp = "<null>";

"vertical\_accuracy" = 0;

};

},

{

Appinstances = (

{

AppName = "MobiledgeX SDK Demo";

AppVers = "1.0";

FQDN = "bonn-mexdemo.tdg.mobiledgex.net";

ports = (

{

"FQDN\_prefix" = "mobiledgexsdkdemo-tcp.";

"internal\_port" = 7777;

proto = LProtoTCP;

"public\_path" = "";

"public\_port" = 7777;

}

);

}

);

CarrierName = TDG;

CloudletName = "bonn-mexdemo";

Distance = "17914.08187216106";

GpsLocation = {

altitude = 0;

course = 0;

"horizontal\_accuracy" = 0;

latitude = "50.737";

longitude = "7.098";

speed = 0;

timestamp = {

nanos = 0;

seconds = 0;

};

"vertical\_accuracy" = 0;

};

},

{

Appinstances = (

{

AppName = "MobiledgeX SDK Demo";

AppVers = "1.0";

FQDN = "berlin-mexdemo.tdg.mobiledgex.net";

ports = (

{

"FQDN\_prefix" = "mobiledgexsdkdemo-tcp.";

"internal\_port" = 7777;

proto = LProtoTCP;

"public\_path" = "";

"public\_port" = 7777;

}

);

}

);

CarrierName = TDG;

CloudletName = "berlin-mexdemo";

Distance = "18387.77976411593";

GpsLocation = {

altitude = 0;

course = 0;

"horizontal\_accuracy" = 0;

latitude = "52.52";

longitude = "13.405";

speed = 0;

timestamp = {

nanos = 0;

seconds = 0;

};

"vertical\_accuracy" = 0;

};

}

)

]

~~~~~