Core Domain Services

Core Domain Services consists of the list of following services:

- Home Identifier Service
    - Open Location Code Service
- Makkal Identifier Service
    - Token Service for Makkal Id
- Makkal Movement Service
    - Golden Record at birth
    - Golden Record at death
- Query Service
    - Home Id
    - Makkal Id
    - Token for Makkal Id

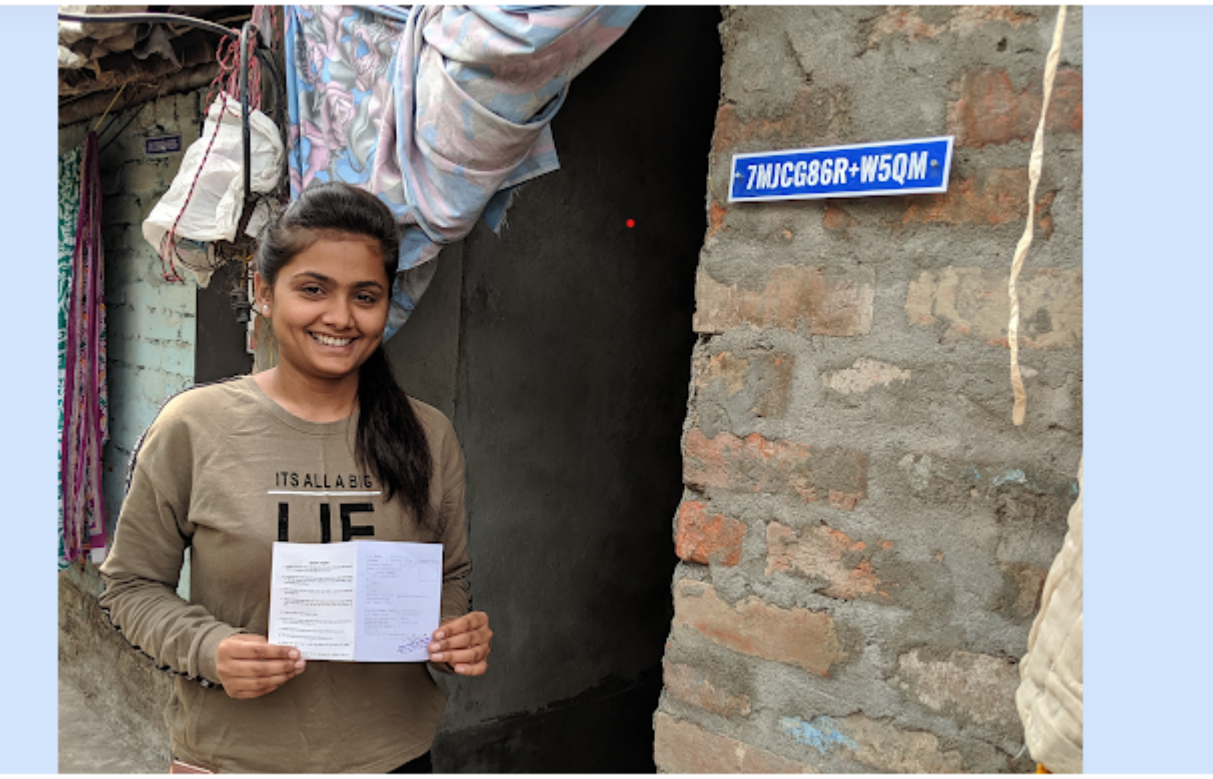
Event Sourcing

Here is the List of sources:

- Birth Certificate
    - Current Residents
    - Mother, Father, Siblings
- Death Certificate
    - Current Residents
    - Mother, Father
- Electricity Bill
    - Current Residents
    - All members
- Department of Civil Supplies – Public Distribution System
    - Current Resident
    - Family and its members
- Department of Social Welfare/Revenue – Old Age Pensions
    - Current Residents
    - Elderly People
- Department of Rural Development – MGNREGA, IAY/CM’s Green House Scheme
- Department of Health & Family Welfare – Comprehensive Health Insurance Scheme
    - Current Residents
    - All members
- Department of Labour & Employment – Schemes for Unorganized Labour Welfare Boards
    - Current Residents
    - Qualified Members
- Department of Adi-Dravidar and Tribal Welfare – Scholarship Schemes
    - Current Residents
    - Students
- Department of BC, MBC and Minorities Welfare – Scholarship Schemes
    - Current Residents
    - Students
- Department of Social Welfare – Welfare Schemes
- Department of Revenue (Land Administration), Department of Registration
    - Owner of the House Property
    - Property Tax
- Department of School Education - Common Database/Smart card / laptop schemes
    - Current Residents
    - Students
- Department of Commercial Taxes –Dealer Authentication using Aadhaar

    - Current Residents
    - Eligible Business Persons
- Department of Revenue Administration Disaster Management & Mitigation – Priceless Saree-Dhoti
    - Eligible Location
    - Eligible Family
    - Eligible Members
- Department of Social Welfare & Nutritious Meal Programme – Integrated Child Development Scheme
    - Current Residents
    - Eligible Children
- Department of Treasuries & Accounts – Pensioners database & Employees database
    - Current Residents
    - Eligible Elders
-

Home Identifier Service



96, Varadharajapuram, Poes Garden, Teynampet,
Chennai, Tamil Nadu 600018, India

27R3+H9 Chennai, Tamil Nadu, India

“From Property Registration, we get this address.”

“96, Varadharajapuram, Poes Garden, Teynampet.”

“Okay. Who owns the property?

“Senthil.”

“From the Electricity Department, who lives in that home?”

“Senthil.”

“Owner lives in his own home. It is a simple and straightforward use case.”

“If the names are different then it may be a rental home.”

“Could we have a model for Home?”

“From Domain Driven Design, Home is an Entity.”

“HomeEntity”

“What are the attributes for HomeEntity?”

“Address. Owner. Rental.”

“Could we have a real-life example?”

Address: 96, Varadharajapuram, Poes Garden, Teynampet.
Owner: Senthil
Renter: NULL

“Instead of the Owner Name, could we have the Makkal Id?”

“That would be a problem; Circular Dependency.”

“What is Circular Dependency?”

“What came first? Chicken or Egg?”

“We still don’t have Makkal Id; so let us not add here as an attribute.”

“The Property Registration Department is interested only in the owner.”

“The Electricity Department is interested only in the current resident.”

“Each department has its own boundary.”

“We couldn’t have all attributes in one Entity.”

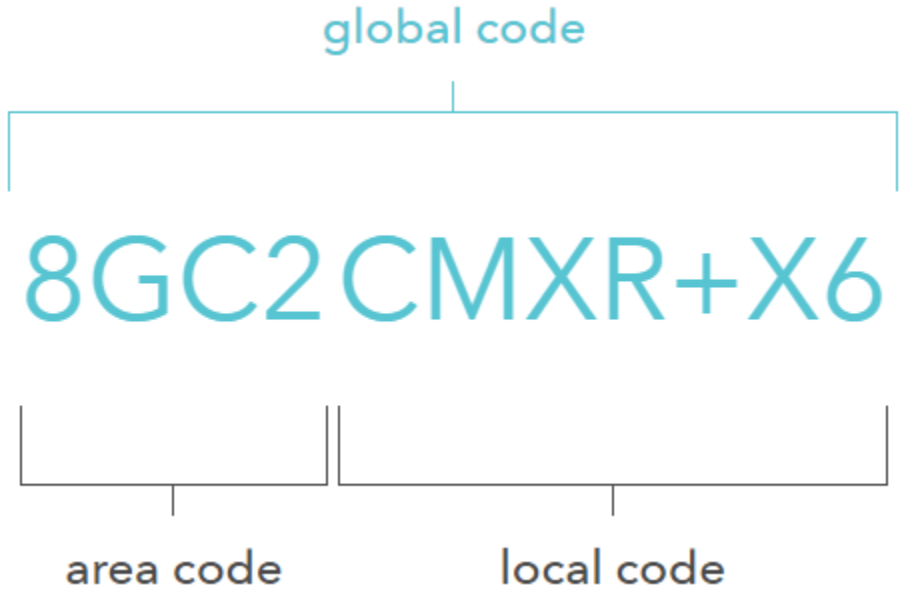
“What are the minimum attributes we need for HomeEntity.”

“Address.”

“96, Varadharajapuram, Poes Garden, Teynampet.”

“Could we generate an identifier for the above address?”

“We could use the Google Plus Codes for identifiers.”



HomeEntity:
Identifier: 27R3+H9
Physical Address: 96, Varadharajapuram, Poes Garden, Teynampet.

“Could we break down the physical address?”

HomeEntity:
Identifier: 27R3+H9
DoorNumber: 96
Street: Varadharajapuram
Area: Poes Garden
Division: Teynampet
City: Chennai
State: Tamil Nadu
Country: India

“Why Country?”

“If we get popular, then we may build the system for the entire planet.”

“The Property Registration system, how do they model?”

PropertyEntity:
HomeIdentifier: 27R3+H9
Owner: Senthil

“Could we have the Makkal Identifier for Senthil?”

“You could provide a place holder; later, update the record from Golden Record.”

PropertyEntity:
HomeIdentifier: 27R3+H9
Owner: Senthil; Makkal Identifier: NULL

“For the Electricity Department, could we have a similar model?”

PropertyEntity:
HomeIdentifier: 27R3+H9
Owner: Senthil; Makkal Identifier: NULL
Current Resident: Kumaran; Makkal Identifier: NULL

“Home Identifier Service depends on Open Location Code Service”

Open Location Code Service

https://github.com/google/open-location-code

Open Location Code is a technology that gives a way of encoding location into a form that is easier to use than latitude and longitude. The codes generated are called plus codes, as their distinguishing attribute is that they include a "+" character.

The technology is designed to produce codes that can be used as a replacement for street addresses, especially in places where buildings aren't numbered or streets aren't named.

Plus codes represent an area, not a point. As digits are added to a code, the area shrinks, so a long code is more precise than a short code.

Codes that are similar are located closer together than codes that are different.

A location can be converted into a code, and this (full) code can be converted back to a location completely offline, without any data tables to lookup or online services required.

Codes can be shortened for easier communication, in which case they can be used regionally or in combination with a reference location that all users of this short code need to be aware of. If the reference location is given in the form of a location name, use of a geocoding service might be necessary to recover the original location.

Algorithms to

- encode and decode full codes,
- shorten them relative to a reference location, and
- recover a location from a short code and a reference location given as latitude/longitude pair

are publicly available and can be used without restriction. Geocoding services are not a part of the Open Location Code technology.

Codes are made up of a sequence of digits chosen from a set of 20. The digits in the code alternate between latitude and longitude. The first four digits describe a one degree latitude by one degree longitude area, aligned on degrees. Adding two further digits to the code, reduces the area to 1/20th of a degree by 1/20th of a degree within the previous area. And so on - each pair of digits reduces the area to 1/400th of the previous area.

As an example, the Parliament Buildings in Nairobi, Kenya are located at 6GCRPR6C+24. 6GCR is the area from 2S 36E to 1S 37E. PR6C+24 is a 14 meter wide by 14 meter high area within 6GCR.

A "+" character is used after eight digits, to break the code up into two parts and to distinguish codes from postal codes.

There will be locations where a 10-digit code is not sufficiently precise, but refining it by a factor of 20 is i) unnecessarily precise and ii) requires extending the code by two digits. Instead, after 10 digits, the area is divided into a 4x5 grid and a single digit used to identify the grid square. A single grid refinement step reduces the area to approximately 3.5x2.8 meters.

Codes can be shortened relative to a location. This reduces the number of digits that must be remembered, by using a location to identify an approximate area, and then generating the nearest matching code. Shortening a code, if possible, will drop four or more digits from the start of the code. The degree to which a code can be shortened depends on the proximity of the reference location.

If the reference location is derived from a town or city name, it is dependent on the accuracy of the geocoding service. Although one service may place "Zurich" close to the Google office, another may move it by a hundred meters or more, and this could be enough to prevent the original code being recovered. Rather than a large city size feature to generate the reference location, it is better to use smaller, neighborhood features that will not have as much variation in their geocode results.

Guidelines for shortening codes are in the wiki.

Recovering shortened codes works by providing the short code and a reference location. This does not need to be the same as the location used to shorten the code, but it does need to be nearby. Shortened codes always include the "+" character so it is simple to compute the missing component.

- 8F+GG is missing six leading characters
- 6C8F+GG is missing four leading characters

The subdirectories contain sample implementations and tests for different languages. Each implementation provides the following functions:

- Test a code to see if it is a valid sequence
- Test a code to see if it is a valid full code (not all valid sequences are valid full codes)

- Encode a latitude and longitude to a standard accuracy (14 meter by 14 meter) code
- Encode a latitude and longitude to a code of any length
- Decode a code to its coordinates: low, high and center
- Shorten a full code relative to a location
- Extend a short code relative to a location

To support plus codes for searching, there are three different cases:

- global codes, such as "796RWF8Q+WF"
- local codes, such as "WF8Q+WF"
- local codes with a locality, such as "WF8Q+WF Praia, Cabo Verde"

https://github.com/google/open-location-code/wiki/Plus-codes-API

The API provides the following functions:

- Conversion of a latitude and longitude to a plus code (including the bounding box and the center);
- Conversion of a plus code to the bounding box and center.

Additionally, it can use the Google Geocoding API to:

- Include short codes and localities in the returned plus code (such as "WF8Q+WF Praia, Cape Verde" for "796RWF8Q+WF");
- Handle converting from a street address or business name to the plus code;
- Handle converting a short code and locality to the global code and coordinates.

The results are provided in JSON format. The API is loosely modeled on the Google Geocoding API.

A Plus codes API request takes the following form:

`https://plus.codes/api?parameters`

Note: URLs must be properly encoded (specifically, `+` characters must be encoded to `%2B`).

Required parameter:

- `address` — The address to encode. This can be any of the following (if the `ekey` parameter is also provided):
  - A latitude/longitude
  - A street address
  - A global code
  - A local code and locality
  - A local code and latitude/longitude (Deprecated in v2 of the API)

Recommended parameters:

- `key` — A Google API key. See API Keys. If this parameter is omitted, only latitude/longitude and global codes can be used in the `address` parameter, and locality information will not be returned. (This can also be specified using `ekey`.)
- `email` — Provide an email address that can be used to contact you.
- `language` — The language in which to return results. This won't affect the global code, but it will affect the names of any features generated, as well as the address used for the local code.
- If `language` is not supplied, results will be provided in English.

Example Requests (no Google API key)

The following two requests show how to convert a latitude and longitude to a code, or how to get the geometry of a global code:

- https://plus.codes/api?address=14.917313,-23.511313&email=YOUR_EMAIL_HERE
- https://plus.codes/api?address=796RWF8Q%2BWF&email=YOUR_EMAIL_HERE

Both of these requests will return the global code, it's geometry, and the center:


```
{
  "plus_code": {
    "global_code": "796RWF8Q+WF",
    "geometry": {
      "bounds": {
        "northeast": {
          "lat": 14.917375000000007,
          "lng": -23.511250000000018
        },
        "southwest": {
          "lat": 14.91725000000001,
          "lng": -23.511375000000015
        }
      },
      "location": {
        "lat": 14.917312500000008,
        "lng": -23.511312500000017
      }
    },
  },
  "status": "OK"
}
```


Example Requests (with Google API key)

Here are some example requests. You must include your Google API key in the request for these to work fully:

- https://plus.codes/api?address=14.917313,-23.511313&ekey=YOUR_ENCRYPTED_KEY&email=YOUR_EMAIL_HERE
- https://plus.codes/api?address=796RWF8Q%2BWF&ekey=YOUR_ENCRYPTED_KEY&email=YOUR_EMAIL_HERE
- https://plus.codes/api?address=WF8Q%2BWF%20Praia%20Cape%20Verde&ekey=YOUR_ENCRYPTED_KEY&email=YOUR_EMAIL_HERE

These requests would all return:


```
 "plus_code": {
    "global_code": "796RWF8Q+WF",
    "geometry": {
      "bounds": {
        "northeast": {
          "lat": 14.917375000000007,
          "lng": -23.511250000000018
        },
        "southwest": {
          "lat": 14.91725000000001,
          "lng": -23.511375000000015
        }
      },
      "location": {
        "lat": 14.917312500000008,
        "lng": -23.511312500000017
      }
    },
    "local_code": "WF8Q+WF",
    "locality": {
      "local_address": "Praia, Cape Verde"
    },
  },
  "status": "OK"
}
```


JSON Response Format

The JSON response contains two root elements:

- `"status"` contains metadata on the request. Other status values are documented here.
- `"plus_code"` contains the plus code information for the location specified in `address`.

There may be an additional `error_message` field within the response object. This may contain additional background information for the status code.

The `plus_code` structure has the following fields:

- `global_code` gives the global code for the latitude/longitude
- `bounds` provides the bounding box of the code, with the north east and south west coordinates
- `location` provides the center of the bounding box.

If a locality feature near enough and large enough to be used to shorten the code was found, the following fields will also be returned:

- `local_code` gives the local code relative to the locality
- `locality` provides the name of the locality using `local_address`.

If the `ekey` encrypted key is not provided the following fields will not be returned:

- `local_code`
- `locality`
- `local_address`

Locality Requests

The `address` parameter may match a large feature, such as:

```
https://plus.codes/api?address=Paris&ekey=YOUR_ENCRYPTED_KEY&email=YOUR_EMAIL_HERE
```

In this case, the returned code may not have full precision. This is because for large features, the returned code will be the **largest** code that fits entirely **within** the bounding box of the feature:


```
{
  "plus_code": {
    "global_code": "8FW4V900+",
    "geometry": {
      "bounds": {
        "northeast": {
          "lat": 48.900000000000006,
          "lng": 2.4000000000000057
        },
        "southwest": {
          "lat": 48.849999999999994,
          "lng": 2.3499999999999943
        }
      },
      "location": {
        "lat": 48.875,
        "lng": 2.375
      }
    }
  },
  "status": "OK"
}
```

Makkal Identifier Service

Family Members:

- Sivaji Ganesan
    - Makkal Identifier (expired)
    - Other Names
        - Ganesan, Shivaji, Nadigar Thilagam
    -
- Kamala
    - Makkal Identifier (expired)
- Ramkumar
    - Makkal Identifier
- Prabhu
    - Makkal Identifier

Home Identifier:

Plus Code: 26PV+H5W
Address: Actor Shivaji House, S Boag Rd, Parthasarathi Puram, T. Nagar, Chennai, Tamil Nadu 600017, India

Home Identifier	Makkal Identifier		
26PV+H5W	Sivaji Makkal Id		
26PV+H5W	Kamala Makkal Id		
26PV+H5W	Ramkumar Makkal Id		
26PV+H5W	Prabhu Makkal Id		

Makkal Movement Service

This service maintains the current residents at the home.
For example, Palanivel Thiagarajan

Makkal Identifier	Home Identifier	Start Date	End Date
PTR Makkal Id	W45X+97 Madurai, Tamil Nadu, India	2014	current
PTR Makkal Id	6PH57VP3+PQ, Singapore	2009	2014
PTR Makkal Id		2001	2008
PTR Makkal Id		1991	2000
PTR Makkal Id	MQQH+W7 Buffalo, NY, USA	1982	1990
PTR Makkal Id	RM3J+52 Trichy, Tamil Nadu, India	1976	1981
PTR Makkal Id	W45X+97 Madurai, Tamil Nadu, India	1966	1976

- Out of Home Town
    - Not eligible for ration
- Out of State
    - eligible for remote vote
- Out of Country
    - Not eligible for any benefits

Token Service for Makkal Identifier

Makkal Identifier	Token for Voter Registration	Token for Property Registration	Token for Vehicle Registration
Ramkumar Makkal Id	token-x1	token-y1	token-z1
Prabhu Makkal Id	token-x2	token-y2	token-z2

Madurai Corporation maintains birth, marriage and death certificates.

Madurai Corporation feeds data to Core Domain Service.

Madurai Corporation asks for the Makkal Identifier.

“Why do you need the Makkal Identifier?”

“We need to update our record for future reference.”

“But we don’t trust you and hand the Makkal Id over to you.”

“We will safely secure the Makkal Id, Sir.”

“You may secure them safely. But hackers may steal the Makkal Ids from you.”

“Then, give us some reference.”

“We will generate a token for Madurai Corporation and give those tokens to you.”

“Sounds Good. When we query Core Domain Service with a token, would you return the Golden Record?”

“Yes. We will return the Golden Record.”

Golden Birth Record

All corporations and local administrations feed the birth certificate details to Core Domain Service.

Local Administration	Transformer	Home Identifier	Makkal Identifier	Token Generator
Madurai Corporation Birth Certificates	csv => JSON	Create and update the home identifier	Create and update the Makkal Identifier	Generate the token
Chennai Corporation Birth Certificates	JSON => JSON			
Coimbatore Corporation Birth Certificates	XML => JSON			
Trichy Corporation Birth Certificates	Text => JSON			

Code Snippet

Data Feed:	Address: Actor Shivaji House, S Boag Rd, Parthasarathi Puram, T. Nagar, Chennai, Tamil Nadu 600017, India
Query Location Code Service	Request: Address Response: Plus Code (Open Location Code)
Home Identifier Service	Insert Address and Plus Code for Location
Makkal Identifier Service	Query for Makkal ID; If available, then update If not available, then create
Makkal Movement Service	Query with Home Identifier If available, then update If not available, then create
Token Service	Generate tokens for the set of records received and responds the source with token

```
URL url = new URL(
        "http://maps.googleapis.com/maps/api/geocode/json?address="
                + URIUtil.encodeQuery("Sayaji Hotel, Near balewadi stadium,
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("Accept", "application/json");

if (conn.getResponseCode() != 200) {
    throw new RuntimeException("Failed : HTTP error code : " + conn.getRespc
}
BufferedReader br = new BufferedReader(new InputStreamReader((conn.getInputS

String output = "", full = "";
while ((output = br.readLine()) != null) {
    System.out.println(output);
    full += output;
}

PincodeVerify gson = new Gson().fromJson(full, PincodeVerify.class);
response = new IsPincodeSupportedResponse(new PincodeVerifyConcrete(
        gson.getResults().get(0).getFormatted_address(),
        gson.getResults().get(0).getGeometry().getLocation().getLat(),
        gson.getResults().get(0).getGeometry().getLocation().getLng())) ;
try {
    String address = response.getAddress();
    Double latitude = response.getLatitude(), longitude = response.getLongit
    if (address == null || address.length() <= 0) {
        log.error("Address is null");
    }
} catch (NullPointerException e) {
    log.error("Address, latitude on longitude is null");
}
conn.disconnect();
```

double latitude = 63.7740574;
double longitude = 23.9011008;
OpenLocationCode olc = new OpenLocationCode(latitude, longitude, 10);
// the last parameter specifies the number of digits
String code = olc.getCode(); // this will be the full code

9GM5QWF2+JC

Insert Home Identifier

Generate Makkal Identifier:
// generate time based UUID

UUID uuid = Generators.timeBasedGenerator().generate();

983e7cc9-4e99-4c1a-9ad6-5cc8d4076b5d 983e7d05-09c9-48eb-961c-5eb550f054b4 983e7d20-a8bf-440c-954a-85d3246f80f3
Makkal Movement Service 983e7cc9-4e99-4c1a-9ad6-5cc8d4076b5d 9GM5QWF2+JC
Token for Makkal Id 983e7cc9-4e99-4c1a-9ad6-5cc8d4076b5d a8bf-440c-954a-85d3246f80f3