UIN and VIN ID[s] Generating Format Change

Key Areas:

- kernel-idgenerator-service
- kernel-idgenerator-vid
- kernel-core

GitHub

- Clone the https://github.com/ICTASL/udi-poc-commons repository.
- Checkout the branch 1.1.3-uin-customize and review the code.

UIN Generating Format:

MOSIP generates a pool of UINs before the registration process and stores them. The UIN generation policies can be defined or modified by country as per their requirement.

The UINs generated for the current implementation, follow the following policies:

- UIN should not contain any alphanumeric characters
- UIN should not contain any repeating numbers for 2 or more than 2 digits
- UIN should not contain any sequential number for 3 or more than 3 digits
- UIN should not be generated sequentially
- UIN should not have repeated block of numbers for 2 or more than 2 digits
- The last digit in the number should be reserved for a checksum
- The number should not contain '0' or '1' as the first digit.
- First 5 digits should be different from the last 5 digits (example - 4345643456)
- First 5 digits should be different to the last 5 digits reversed (example - 4345665434)
- UIN should not be a cyclic figure (example - 4567890123, 6543210987)
- UIN should be different from the repetition of the first two digits 5 times (example - 3434343434)
- UIN should not contain three even adjacent digits (example - 3948613752)
- UIN should not contain admin defined restricted number

  According to current modification the format will change as like **xxxx-xxxx** with separating hyphen. Further the document you can refer the below area and identify what are the modification related to UIN new format. It will deeply describe more information about code changes.

1. Properties values [application-local-property]

```
mosip.kernel.uin.length=8
mosip.kernel.uin.length.reverse-digits-limit=4
mosip.kernel.uin.length.digits-limit=4
```

2. Constants [UinGeneratorConstant]

```
 public static final Integer ID_GROUP_DIGIT_LIMIT = 4;
```

3. Regex [UinFilterUtil.java]

   Developing java RegEx pattern to match hyphen (-) separator.

```
String seperatorRegEx = "[0-9][0-9,-]*-[0-9,-]*[0-9]";
```

4. Validation [UinFilterUtils.java]

   Adding new validation **`checkGeneratePatternIsPresent()`** and call this method while the execute UIN generate process. Furthermore remove the hyphen and trimming whitespace in the generated ID, while the execute the **`validateLength()`** method.

5. Validation [ChecksumUtils.java]

   Remove the hyphen and trimming whitespace in the generated ID, while the execute the **`validateChecksum()`** method.

6. UIN format method [UinGeneratorImpl.java]

   The method **`generatedSingleIdFormat()`** call while the executing **`appendChecksum()`** method.

VIN Generating Format:

MOSIP will generate a pool of VIDs through a Batch Job. The number of VIDs generated will be configurable by the country. All the VIDs generated will be assigned a status “Available” which means that the VID is available for allocation to a UIN. Any request for VID allocation will pick up VIDs which have this status. The Batch Job to generate the pool will run every time the number of VIDs in the pool reduces to a configured number.

VID generation will happen according to the below logic:

1. VID generated should contain the number of digits as configured.
2. A generated VID should follow the below logic
   a. The number should not contain any alphanumeric characters
   b. The number should not contain any repeating numbers for 2 or more than 2 digits
   c. The number should not contain any sequential number for 3 or more than 3 digits
   d. The numbers should not be generated sequentially
   e. The number should not have repeated block of numbers for 2 or more than 2 digits
   f. The number should not contain the restricted numbers defined by the ADMIN
   g. The last digit in the number should be reserved for a checksum
   h. The number should not contain '0' or '1' as the first digit.

   According to current modification the format will change as like xxxx-xxxx-xxxx-xxxx with separating hyphen. Further the document you can refer the below area and identify what are the modification related to VIN new format. It will deeply describe more information about code changes.

1. Constants [VidPropertyConstant.java]

```
public static final Integer ID_GROUP_DIGIT_LIMIT = 4;
```

2. Regex [VidFilterUtils.java]

   Developing java RegEx pattern to match hyphen (-) separator.

```
String seperatorRegEx = "[0-9][0-9,-]*-[0-9,-]*[0-9]";
```

3. Validation [VidFilterUtils.java]

   Adding new validation **`checkGeneratePatternIsPresent()`** and call this method while the execute VIN generate process. Furthermore remove the hyphen and trimming whitespace in the generated ID, while the execute the **`validateLength()`** method.

4. Validation [ChecksumUtils.java]

   Remove the hyphen and trimming whitespace in the generated ID, while the execute the **`validateChecksum()`** method.

5. VIN format method [VidGeneratorImpl.java]

   The method **`generatedSingleIdFormat()`** call while the executing **`appendChecksum()`** method.

Docker build configurations:

1. Go to kernel-idgenerator-service project directory in your local machine
2. Build Docker file using this command
   docker build -t udipoc/registration-processor-uin-generator-stage:<add your tag to here> .
3. Verify the build image using below command
   docker images
4. Push your docker image to Docker hub
   docker push