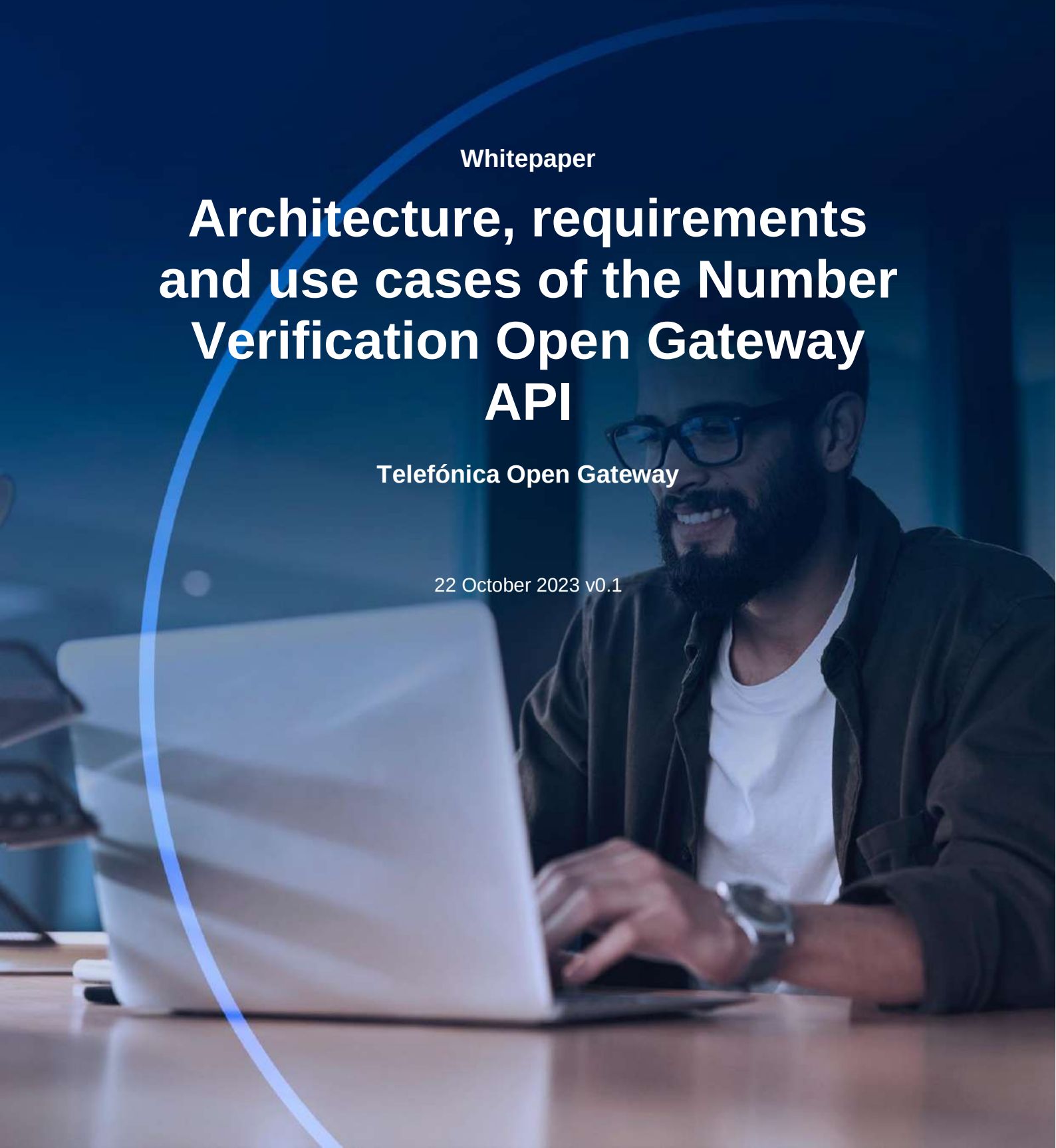
Telefónica

Whitepaper

Architecture, requirements and use cases of the Number Verification Open Gateway API

Telefónica Open Gateway

22 October 2023 v0.1



Índice

1. Introducción

1.1 Document Objective

The objective of this whitepaper is to provide a comprehensive understanding of the CAMARA Number Verification API, highlighting its capability to provide strong mobile-based user's authentication, enhanced security, and superior customer experience. It aims to cater to a technical audience, including developers, engineers, and integration specialists, who are seeking to use the API within their systems.

By delving into the intricacies of the CAMARA Number Verification API, this whitepaper aims to:

- Clearly articulate the purpose and functionality of the API.
- Provide technical insights and guidance on integrating and leveraging the API effectively.
- Showcase the advantages and benefits of adopting the CAMARA Number Verification API for aggregators and service providers.
- Highlight best practices, recommendations, and real-world case studies to illustrate successful API implementations.

Through this document, readers will gain a solid understanding of the API's architecture, components, integration requirements, security considerations, and the overall workflow involved in leveraging Number Verification capabilities. It aims to empower technical professionals with the knowledge and tools necessary to integrate the CAMARA Number Verification API seamlessly into their systems, facilitating the authentication of their users in a secure and agile way.

1.2 Background and Context of Digital User Authentication

1.2.1 Digital identity and authentication

The electronic or digital representation of an individual or organization is referred to as their "digital identity". It is a group of digitally saved identifying characteristics that collectively and specifically identify a person or legal organization and are exchanged during electronic transactions. These characteristics can be physically or electronically recorded, and they are digitally saved.

Opposite to a physical ID (e.g.: a passport), a digital ID can serve to authenticate an individual remotely over digital channels. It can track his/her activities and collect information, such as personal data, behavior, and interactions, but it can also be employed when identifying the individuals when accessing certain services. The parties (services, application providers, or other users) participating in the transaction need to

trust each other in the context of digital identities and electronic transactions. Authenticity and dependability as the basis for trust.

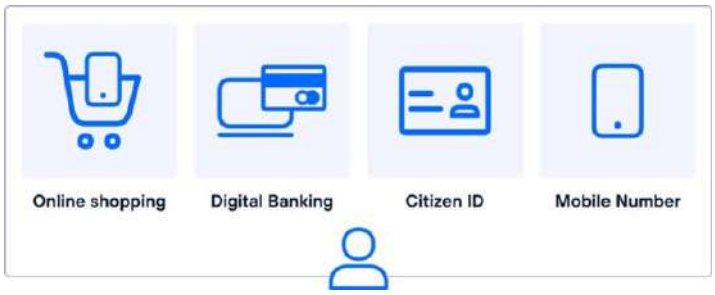


Figure 1. Four examples of common activities that require digital IDs:

Online shopping / Customer ID

Secure online transactions require proper customer identification. While a customer ID provides increased security for the user, it also helps providers manage data, such as tracking customers' transactions, preferences, or demographic information. Additionally, by detecting patterns, they can improve customer service, personalize marketing campaigns, and even prevent fraud.

E-banking ID

Security is even more relevant in these use cases. The e-banking ID for a user to access digital bank services usually consists of a username/password, contract number, and other digital factors. Customers can view their bank account information and perform transactions, such as paying bills or trading securities, once logged in.

Citizen ID

Authorities provide citizens with secure access to their online tools and tasks 24/7, enhancing public service and customer interaction. Citizens can, for example, order official documents online, avoiding time-consuming office visits. Furthermore, every taxpayer finds a personal identification number on the tax return they must complete, allowing them to submit the documents electronically.

Mobile Phone Number ID

In mobile communication, the phone number (or MSISDN) is employed to identify each user’s line. It allows the user to communicate with other parties using a familiar and convenient format that internationally identifies its phone connection when calling or accessing other mobile services. Nowadays, mobile devices are the primary way in which people access online services and even make and receive payments.

1.2.2 Multi-Factor Authentication

To use any kind of digital services, users must authenticate themselves on the provider’s website/app with their credentials. An authentication factor is a category of evidence that a person must present to prove they are who they say they are, like a

user and password in an email account, a national or citizen ID for legal services or the phone number in a phone call. The three (main) authentication factors are:

Knowledge Factor → **something you know**, e.g., username and password.

Possession Factor → **something you physically have**, e.g., mobile phone.

Inherence Factor → **something you are**, e.g., fingerprint.

Individual authentication factors on their own may present security vulnerabilities, sometimes due to user behavior patterns and habits and other times, because of the limitations of technology. Today, many organizations use multiple authentication factors to control access to secure data systems and applications. Passwords and pin numbers must be memorized by users when using a knowledge-based authentication factor. As a result, users may create unnecessarily simple passwords and update them very rarely, making them easy to guess or steal.

Figure 2. Authentication methods according to their nature

Biometric and possession-based authentication factors may be the strongest means of securing a network or application against unauthorized access. Combining these methods into a multi-factor authentication process decreases the likelihood that a hacker could gain unauthorized access to the secured network. For instance, a banking application may employ a multi-factor authentication joining the security of a PIN code (knowledge), validation of smartphone by the mobile number (possession) and face recognition (inherence), for the login of customers and the validation of transactions. For enhanced security, the joint use of 2 factors of authentication or 2FA for short, knowledge and possession is a highly recommended option. The combination of the

possession factor (a mobile phone) and biometric signature (e.g fingerprint) is also becoming very popular.

1.2.3 Seamless and transparent authentication using network connection.

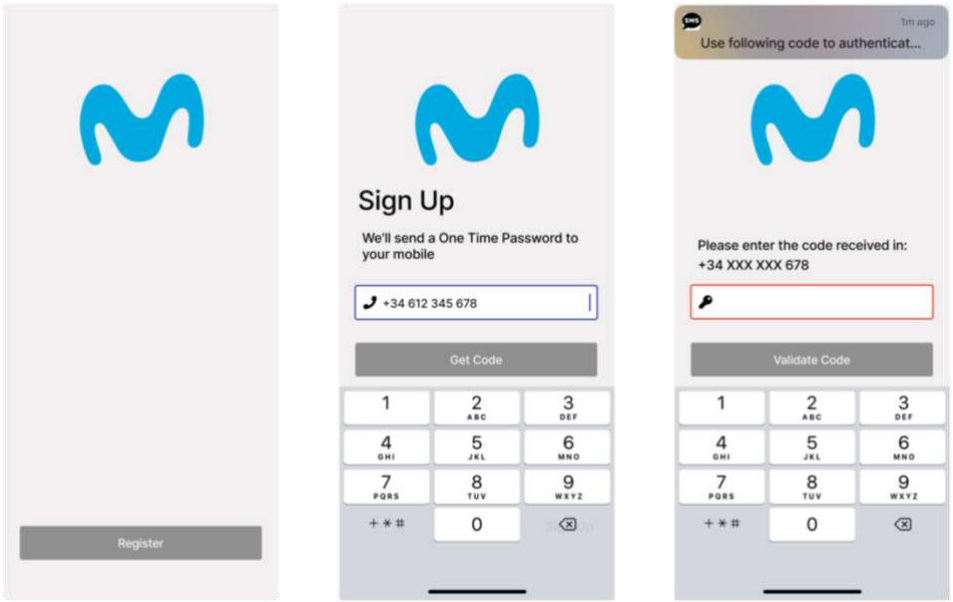
Smartphones are extensively used by users to access digital services. The cell phone is the most used device for managing a wide range of products and services, from food delivery to gym memberships to vehicle insurance and even bank accounts. As a result, utilizing an authentication system that incorporates the smartphone itself (possession factor) is considered as the most convenient.

Nowadays, digital customers are used to employ multi-factor authentication mechanisms that involve using their smartphone as a proof of possession. Code generators are an option that allows to validate that the customer is using the same device in each of the interaction, but they do not validate the possession and usage of the phone number. SMS One-time Passwords (OTPs) are the most used possession authentication factor, because of their simplicity and re-usage of the widespread existing SMS delivery systems.

SMS OTPs are based on the premise that a user owning a phone will be able to read a received SMS sent to their phone number, and will be able to manually introduce it in a designated application field. When registering in a new application (or getting logged or validating a transaction) users are required to use SMS OTP for:

- Validating that the phone number they registered with is the one they own (and they are currently using) à *Number Verification process.*
- Validating that they are in possession of the same phone number as they registered à *Possession factor.*

For that purpose, users introduce their phone number during the registration process, and wait for the receival of the security code as part of an SMS content.



After receiving the SMS, users can read the code, temporally memorize it, and manually introduce the code in the application, so they can validate the process.

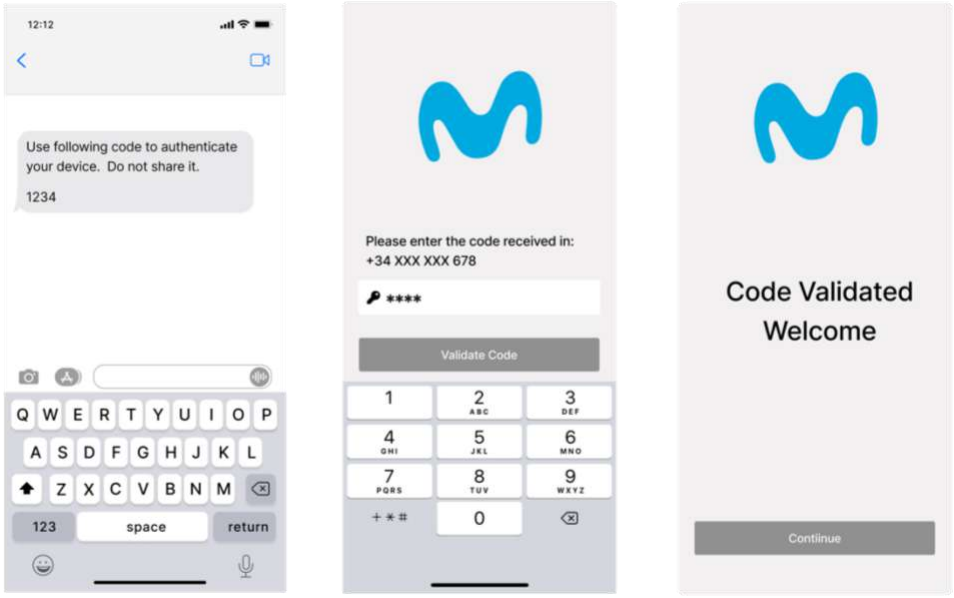


Figure 3. User authentication flow with SMS OTP method.

Multiple steps are required by a user to register or validate a digital transaction that requires mobile possession factor authentication by SMS OTP, what leads in several issues related to user experience, or even security as the text message with the authorization code is not encrypted. This method is also vulnerable to scams like SIM Swapping. In an era where cyberthreats and the risk of data breaches are on the rise, strengthening security measures is critical. Many people believe that the more steps individuals take during identity verification, the safer both consumers and organizations become in today's digital environment. This notion, however, does not imply that authentication must be difficult or time-consuming for customers. In fact, providing unneeded friction in the early phases of the user journey can lead to a poor customer experience, potentially leading to customers abandoning sign-ups or purchases entirely.

Apart from the non-transparent user experience, SMS was not architected initially to act as a possession factor, and the National Institute of Standards and Technology (NIST) states that “due to the risk that SMS may be intercepted or redirected,” businesses should consider alternative authentication methods.

Because of this, new solutions are now available that leverage carrier network-based technology to provide strong authentication. When consumers turn on their phones, the device and SIM are automatically connected to the carrier and authenticated. They don't have to do anything as a user. Today, carrier network-based user authentication is the most secure and frictionless approach, which makes use of mobile network capabilities to enable strong authentication with higher security and better user experiences compared to other methods.

1.3 How does the Number Verification API help to facilitate authentication?

The Number Verification API leverages the internal telco mechanisms to transparently authenticate users based on their connection of their devices to the network.

In comparison with current available solutions for user authentication, the validation of number based on network mechanisms are easier to use by customers and provide more security, since no manual interaction is required, and no plain-text codes are employed.

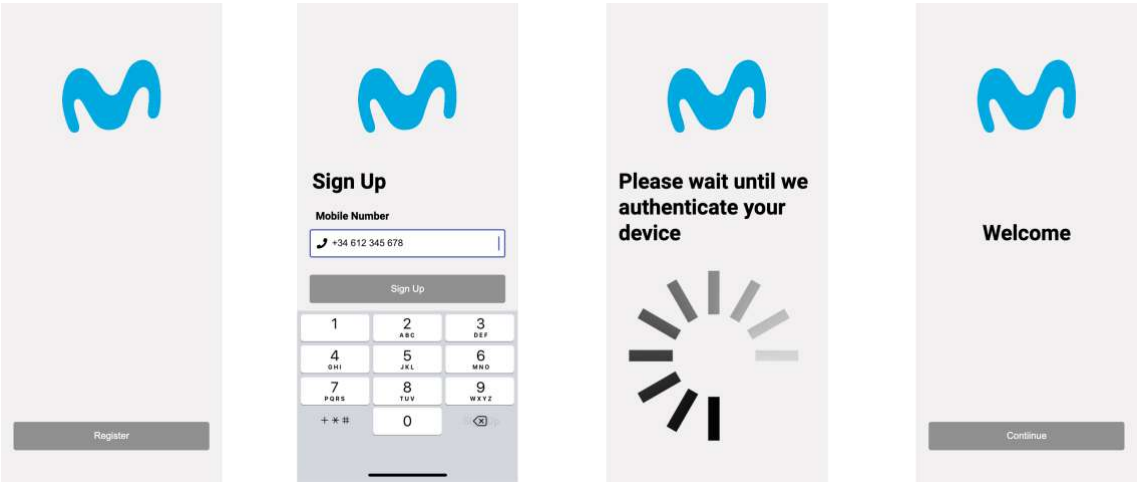


Figure 4. Authentication flow experience with mobile network-based number verification.

The Number Verification service verifies that the provided mobile phone number (MSISDN) is the one paired with the device from which the data communication is happening. Thus, it verifies that the user is interacting with a digital service from a device with the same mobile phone number as it is declared.

Operators employ different techniques or mechanisms to identify the ownership and identity of the user's connection, so the subscription and, therefore, the phone number can be securely and univocally identified. It's clear that the mechanism will be valid in those scenarios where the user's device is connected to the mobile network, so SDKs or applications may employ mechanisms to validate and ensure the connection so that the network identification can be performed. Wi-Fi (and tethering) or VPNs may not allow the Number Verification service to properly authenticate the user's connection, since the service employs the mobile network connection to validate the user with enhanced security. Application SDK will need to ensure that the device is using a mobile connection.

Main features of the Number Verification API are:

- **Superior user experience, frictionless to the user:**

Once the user submits the phone number to the app, the verification is entirely invisible. They do not have to go to an authenticator app, there are no PINs to mistype, no URLs to click on, and no actions to review and approve.

Automatic fill-in tools are getting introduced in the applications for the received codes, but that requires allowing applications to read all the received SMSs of the user.

- **Faster authentication mechanism:**

Number Verification allows the authentication to be performed automatically, instead of waiting for an SMS to be received or opening other authentication applications manually.

- **Improved transaction conversion:**

Facilitating the user experience, transactions and registration conversions rates are improved, reducing the sign-up abandonment.

- **No phishing attacks for account takeovers:**

The user cannot be socially engineered as there is no information to be phished nor manual process to be performed by the user. More secure than SMS OTP, which can be vulnerable to hijacking and man-in-the-middle attacks like SIM Swapping.

- **Unified integration via APIs:**

Easy integration with existing applications due to the use of standard authentication mechanism (e.g., OAuth2 Authorization Code grant*) and simple APIs.

*https://www.rfc-editor.org/rfc/rfc6749#section-4.1

- **Open to advanced protection:**

Ready to create stronger services with additional Open Gateway APIS:

    - **SIM SWAP**: Ensures that the user line has not been compromised.
    - **Device Location Verification**: Adds additional risk scoring information by validating users' transactions based on their location, e.g., validating if they are in the ATM where a transaction is being executed.

- **Improved privacy:**

This method does not leave any traces in the phone (like SMS that remain archived).

1.4 Number Verification API in CAMARA

The GSMA Open Gateway initiative, led by the GSMA (Global System for Mobile Communications Association), aims to drive collaboration and interoperability among telcos, aggregators, and service providers in the mobile ecosystem. It provides a platform for industry stakeholders to develop and deploy innovative mobile services, including digital identity solutions.

By participating in the GSMA Open Gateway initiative, telcos and aggregators can leverage the collective expertise and resources of the mobile industry to accelerate the adoption of digital services in different business scopes.

Figure 1: Logo for the CAMARA Project within Linux Foundation.

Network authentication mechanisms are available in the industry for years, but difficult integrations and market fragmentation have prevented them to fully succeed. Now, the Number Verification API itself is standardized in the CAMARA Telco Global API Alliance, facilitated by the GSMA. The CAMARA standardization of this API brings together telcos and service providers from around the world to establish best practices, share knowledge, and promote industry-wide cooperation. As a result, in the scope of the Open Gateway initiative, this API can be integrated by any kind of company in the digital services industry around the world in an easy, fast, and seamless way.

2. Overview of the Number Verification CAMARA API

2.1 Definition of the Number Verification CAMARA API

The CAMARA API validate the user identity by confirming the ownership of the phone number which they are registering, by matching it with the number that operator identifies from the user's device connection.validate the user identity by confirming the ownership of the phone number which they are registering, by matching it with the number that operator identifies from the user's device connection.

The API specifies the following two operations:

- `POST verify`: answers the question 'does the introduced phone number match the one that the user is currently using?'. This operation just needs the phone

number to be checked (parameter ‘phoneNumber’) and is the preferred option for user authentication.

- `GET device-phone-number`: answers the question ‘which phone number match the is user is currently using?’ This operation doesn’t require any input and returns as answer the phone number identified as the one the user is currently using.

The Number Verification CAMARA API enables developers to directly integrate the authentication mechanism into their application, with a seamless experience to their customers. Also, this API can be combined with other Open Gateway APIs related to the anti-fraud scope that may complement the experience and increase the security.

The inclusion of channel partners and service aggregators in the integration chain facilitates the integration of telco functionalities with other mechanisms like security algorithms or additional security measures or external data sources, for instance including back-up authentication mechanisms in case the network authentication is not available or enhancing the service with additional Open Gateway APIs such us Device Location Verification or SIM SWAP.

2.2 Advantages and benefits of using Number Verification API

The CAMARA Number Verification API offers numerous advantages and benefits for the industry interested in enforcing identity protection. Here are some key arguments highlighting the advantages of using the CAMARA API:

1. **Secure authentication and identity validation:** Based on network mechanisms instead of less secure Over the Top mechanisms, and simplest than hardware tokens. Works on any internet enabled mobile device connected to carrier mobile data network, even when roaming (and even in Wi-Fi, if a temporal network transition is allowed).

2. **Improved user experience:** No need to copy and paste or remember one-time passwords sent via SMS, improving the experience and security of the validation processes.

3. **Anti-fraud suite:** Number Verification is just one of the Open Gateway APIs related to the protection of the customers identity in the scope of mobile digital services. Other APIs can be used to enforce this protection in different circumstances and use cases. For example, the APIs SIM SWAP, Device Location Verification, and Know Your Customer - Match, among others.

4. **Usability:** The CAMARA APIs are designed to be developer-friendly and easy to set up and use. It simplifies the integration process for telcos and any kind of

clients, allowing them to offer Number Verification as an option to authenticate the users.

5. **Footprint:** The CAMARA standardization of Number Verification guarantees a common accesso to the functionality across Telco operators and countries.

6. **Security:** The CAMARA guidelines guarantees a common privacy and security framework that tackles the needs of the service providers while preserving the rights of the customers and.

2.3 Use Cases

In this section, we will explore two targeted use cases for the Number Verification CAMARA API, highlighting its relevance and benefits in the following domains: Media, Entertainment & XR, Financial Services & Insurances and E-Commerce & Retail.

2.3.1 Secured transactions

Secure applications require the validation of the identity of the users when managing different activities, like registration, password recovery or confirming transactions.

SMS OTP is widely used to prove that the user is in possession of the mobile device associated with the mobile number that identifies the user. Instead, the application can request a seamless authentication of the mobile device via the Number Verification API, proving ownership of the phone number and providing the application with confirmation that the user owns the device and phone number, needed to validate a banking transaction (e.g., money transference) or initiate a password recovery process.

Password recovery often requires complex processes, which include remembering certain personal data or accessing other apps, like email, which can be hard to complete for certain people like seniors. By employing the network as the identity to start the password recovery, developers ensure that users can easily access again to their services.

Number Verification allows to secure the transactions while improving the user experience and ensuring the transaction conversion rates. Other APIs can further enhance security, such as SIM SWAP.

Figure 2: Number Verification API secures the banking transactions

Developer needs:

- Secure validation of the user identity, concretely the ownership of the registered phone number.
- Build sophisticated algorithms to evaluate fraud risks in different circumstances.
- Higher conversion rate in transactions and more secure validation of critical processes

2.3.2 Fast login experience

Online services such as social media applications require multi-factor authentication mechanisms for users to log in.

The login process in certain mobile applications requires a manual process where the users have to prove that they are in possession of the device that they have registered with the application (Level of Authentication/Assurance 2 or LOA2). SMS-OTP is the most used method, allowing the application to validate that the user is using the registered mobile phone when logging in (possession factor). Number Verification makes it possible to simplify and speed up the login procedures without manual interactions from the user (e.g., copy-paste of code).

Using Number Verification API combined with additional Open Gateway services can improve the level of security of the applications, integrating the complete anti-fraud suite of APIs from CAMARA.

Figure 3: Number Verification API helps users to log in to application.

Developer needs:

- Frictionless login process and optimized and transparent user experience without manual processes.
- Optimized and transparent user experience without manual processes with higher onboarding conversion rate.
- Secure application access with additional authentication factors.

3. Architecture and Components

3.1 High-Level Architecture

The following figures describe the high-level architecture of the three scenarios where Number Verification CAMARA API can be used.

The first figure represents the scenario where a Service Provider uses the Number Verification API through a Channel Partner e.g., from the marketplace of a Hyperscaler (e.g., Microsoft Azure, AWS, Vonage, Google Cloud). The Service Provider or developer uses a SDK provided by the channel partner to implement the integration with the API. In this way, the Service Provider can directly use the Number Verification API. In fact, they could also use any other available APIs (whether they are related to the anti-fraud suite or not).

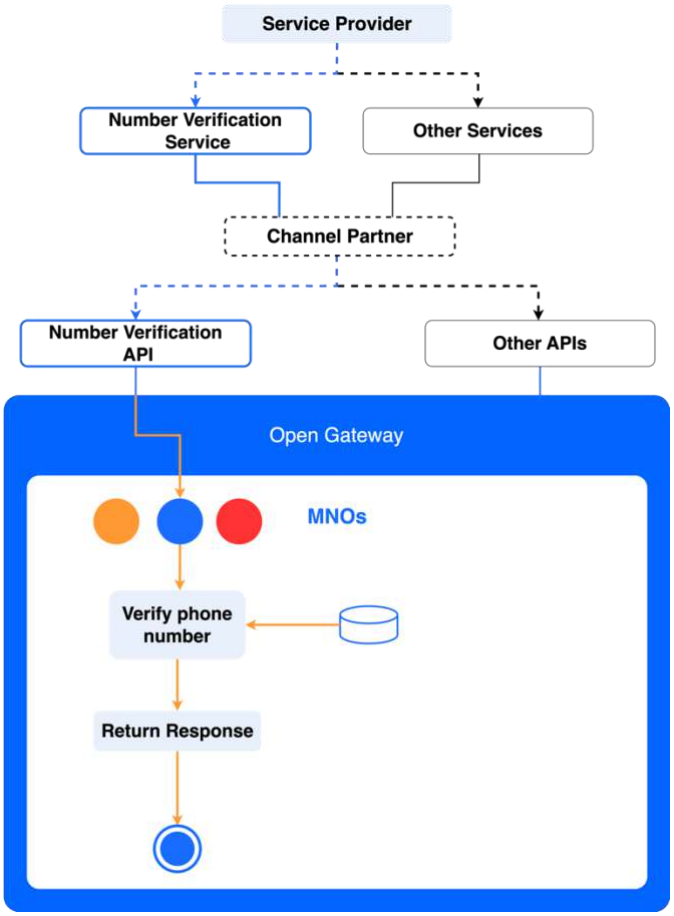


Figure 4: Number Verification API used from a Channel Partner

The second figure represents the scenario where a Service Provider uses the Number Verification API from the service provided by an Aggregator. The Aggregator could build a sophisticated identity service by combining multiple APIs from Open Gateway, other external APIs, implementing their own in-house products, etc.

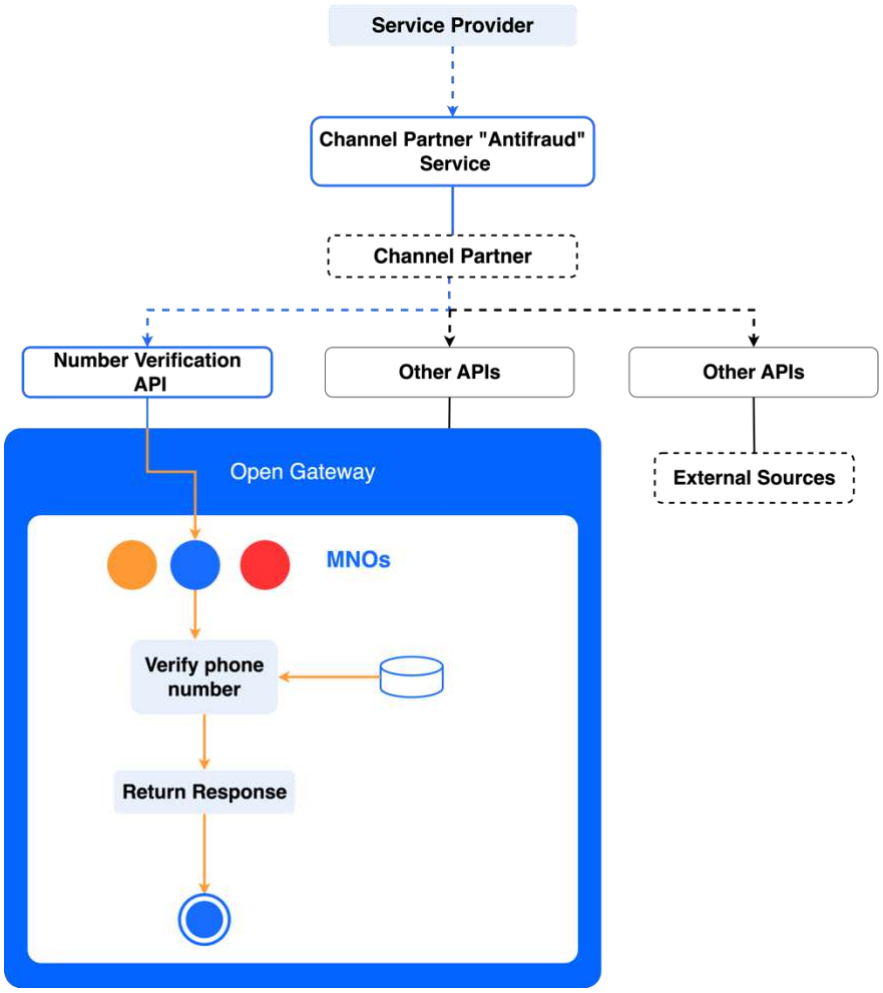


Figure 5: Number Verification API used from the service of a Channel Partner

The third figure represents the scenario where a Service Provider uses the Number Verification API directly from Open Gateway. It can be either integrating with operators or using a kind of gateway to them.

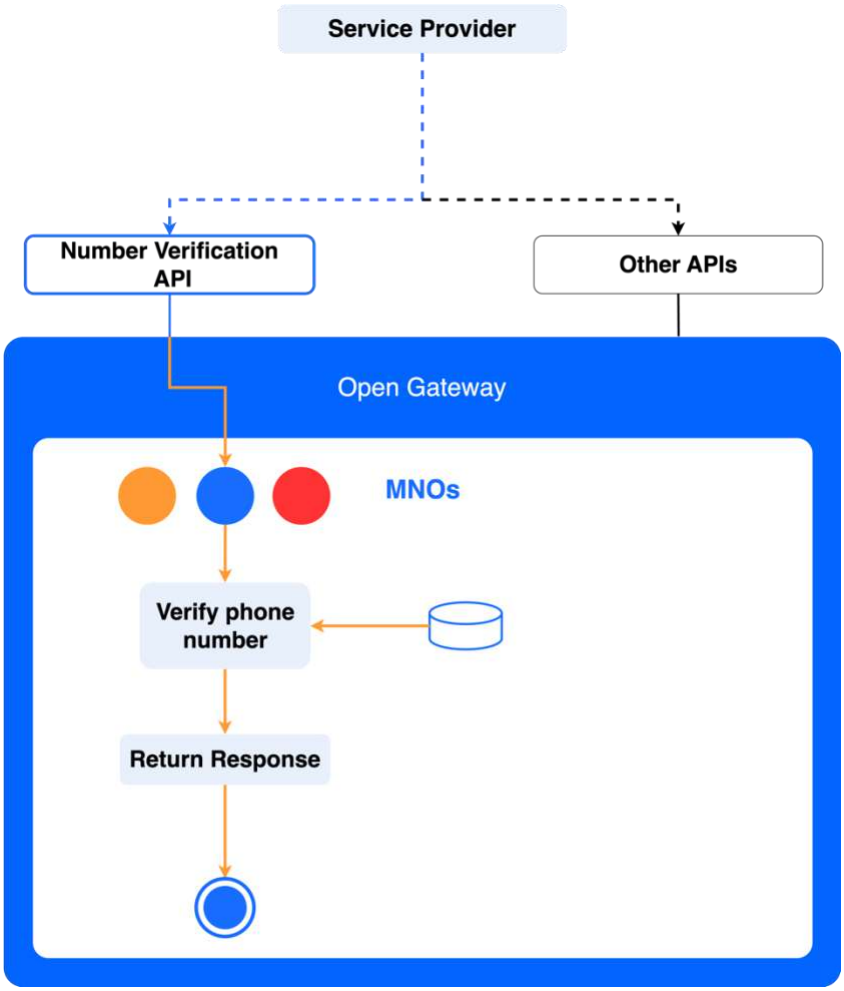


Figure 6: Number Verification API used directly integrating with Telco operators.

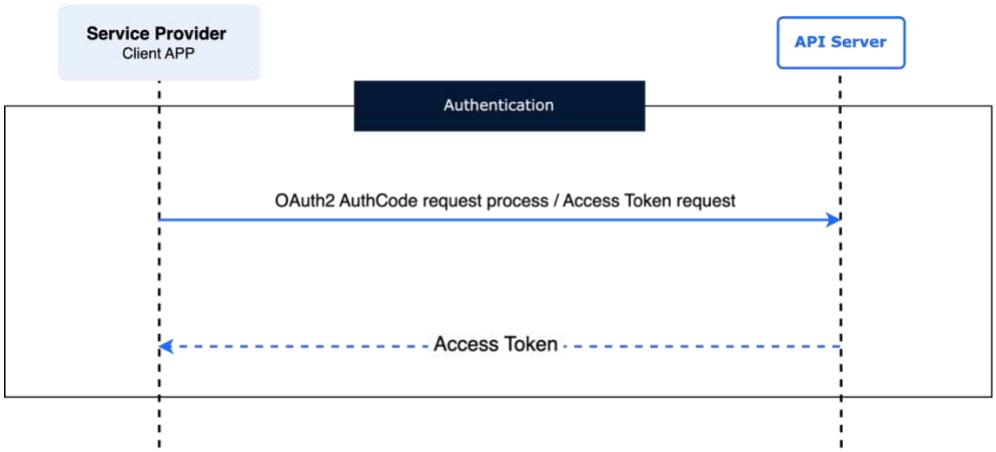
- **Service provider:** also known as developers, company whose application takes advantage of the features of Number Verification API to authenticate their customers. When using the Number Verification API from a Hyperscaler, the service provider shall implement the request to the API by using the SDK provided by the Hyperscaler in the marketplace. When using the Number Verification API from an Aggregator, the digital service provider must implement the request to the API or SDK exposed by the Aggregator, that could give added value to the service provider.

- **Hyperscaler:** they are usually cloud players that provide marketplaces where APIs and other services can be contracted by clients that are used to consume cloud products. In this case, they reach agreements with MNOs in the Open Gateway context to publish their APIs and then they set prices and conditions to the consumption of their products by the digital service providers.

- **Aggregator/Integrator:** the role is similar to the hyperscalers' one, but they can aggregate other services to provide digital service providers with more sophisticated products and personalized experiences.
- **Open Gateway Operator platform:** the hyperscalers and aggregators set up integrations with the MNOs in the Open Gateway context.

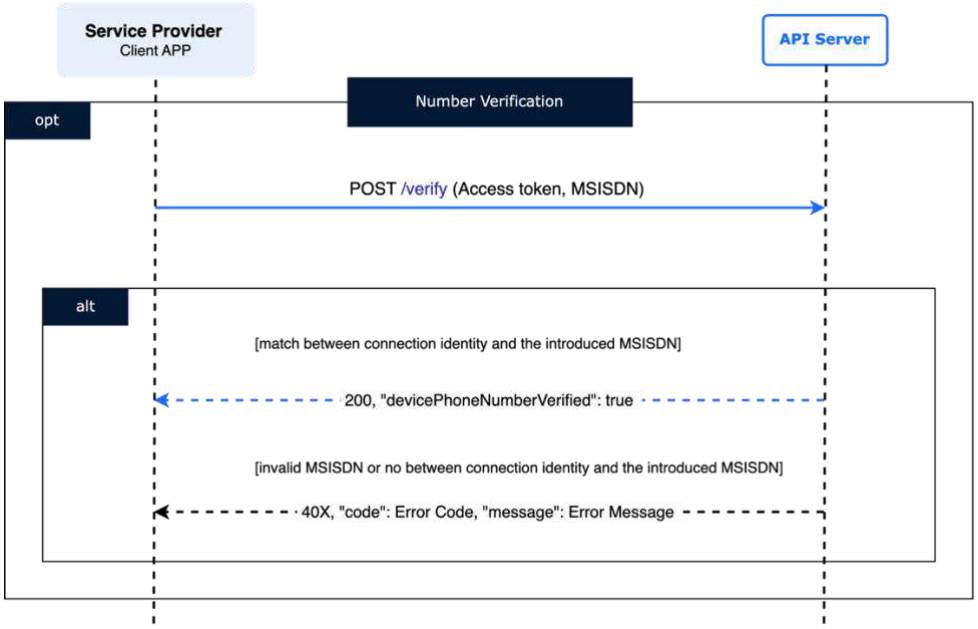
3.2 How to use the API: workflow and implementation

The most relevant interaction of this API is the user's connection authentication. With this process (detailed in Subscriber authentication), the Operator is capable of authenticating the users (network subscribers) based on their connection and provide a valid access token which identifies them.



Number Verification API allows two different operations, although the main usage is focused on the verification endpoint.

The operation '**verify"** confirms whether the introduced MSISDN is the same as the one identified in the authentication process by the access token, returning true in case of match, and false in case any trouble or miss-authentication is detected.



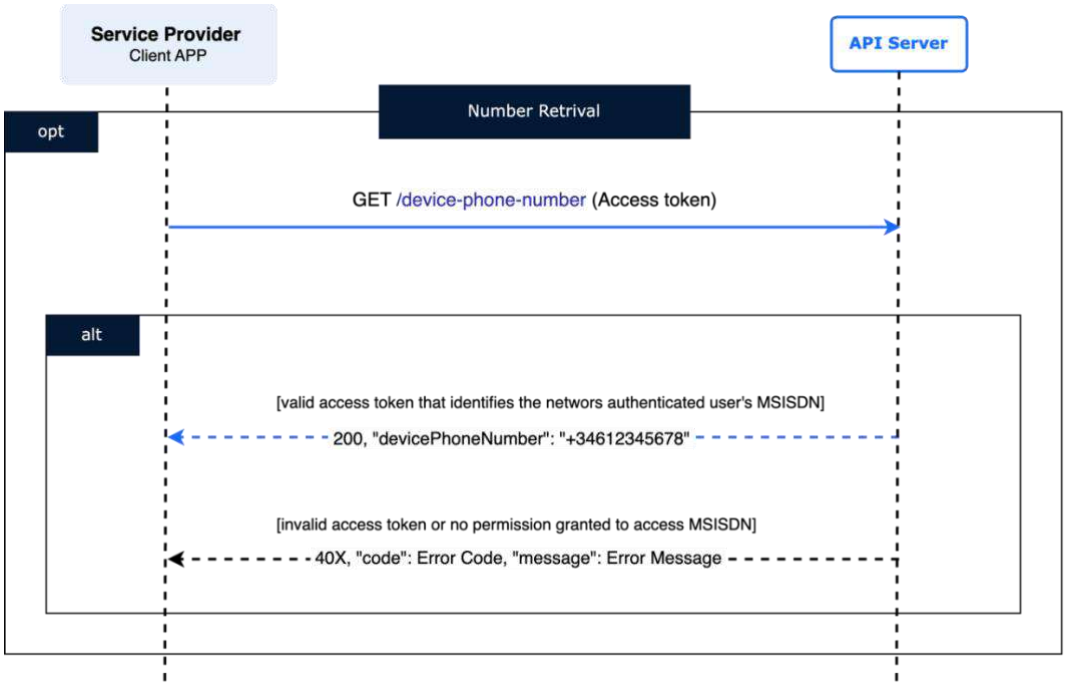
Request example: (POST)

```
{
      “phoneNumber”: “34612345678”
}
```

Response example:

```
{
      "devicePhoneNumberVerified": true
}
```

The operation (get) ‘**devicePhoneNumber’** directly returns the phone number of the user, as identified in the access token.



Request example: (GET)

Response example:

```
{
      "devicePhoneNumber": "+34612345678"
}
```

Apart from the explicit definition of the API REST requests, when the API is used from the marketplace of a Hyperscaler, an SDK will be provided to make the corresponding requests. The objective of such SDK is to make it easy to integrate the API call into the software of the developer. The following is **just an indicative example of how the Number Verification 'verify' operation could work using Python**:

```
from Example.OpenGatewaySDK import NumberVerification

# Users Authentication and access token request
NumberVerification.Authorize()

# One-line Open Gateway query (returns true or false)
verified = NumberVerification.verify(phoneNumber)
```

4. Technical Requirements and Considerations

This chapter provides a comprehensive overview of the integration flows needed for seamless integration with the Open Gateway Operator platform (API Gateway).

4.1 Privacy Management and Access Tokens

Several Open Gateway API enable developers or applications the access and usage of personal information or data. "**Using" personal data refers to any manipulation (in any way) of the any information which are related to an identified or identifiable natural person**, even just reading it. The usage of personal data, therefore, requires to be legitimized.

For controlling the access to personal data, consuming an **Open Gateway API requires the usage of an access token**. This mechanism validates the secure access of the consuming application to the data exposed by the API and identifies the exact data that is being accessed.

The access token allows to **track the data consumption**, ensure the proper p**rivacy management** (e.g., validating if the customer has rights to access) and gives the possibility to directly **provide subscribers control over the access to their data,** enabling consent capture/revoking mechanisms.

For that reason, the developer/application shall first request a valid access token that identifies:

- Data (what) that is going to be accessed.
- Purpose (for what), reason why the data is going to be accessed.
- 3-legged identification of the parties interacting:
    - Data consumer: Application
    - Data provider: Operator
    - Data owner/origin: Network Subscriber

Legitimizing the access to personal data can imply multiple procedures, that shall be validated by the operator during the access token retrieval. In case that the lawful basis to consume the API is explicit **Consent**, the capture procedure of such consent may be launched during the token creation, ensuring that the subscriber/user is identified and can clearly confirm or reject the access to the data. Legitimate interest could also apply to this API, but this decision is to be taken by the telco according to its legal/privacy assessment.

Number Verification API is usually employed for antifraud purposes, so legitimate interest is quite often applied as the lawful basis to consume the API. A legitimate interest lawful basis does not require explicit consent, but the subscriber is informed about the access and can revoke it in any moment. It is the duty of each telco to implement the channels and procedures to manage lawful management by its subscribers.

4.2 Channel partners

As previously introduced, Open Gateway enables the integration of Channel Partners (aka aggregators) that facilitate the integration of APIs and improves the experience of developers and the functionalities of the products. Channel Partners can include service aggregators, hyperscalers or any other partner which will collaborate in the commercialization of the Open Gateway services.

Channel partners oversee aggregating:

- **Aggregate developers and applications:** Channel partners facilitate the reach to developers and applications and creates homogeneous and unique integration towards operator. *Multiple applications accessing one operator.*

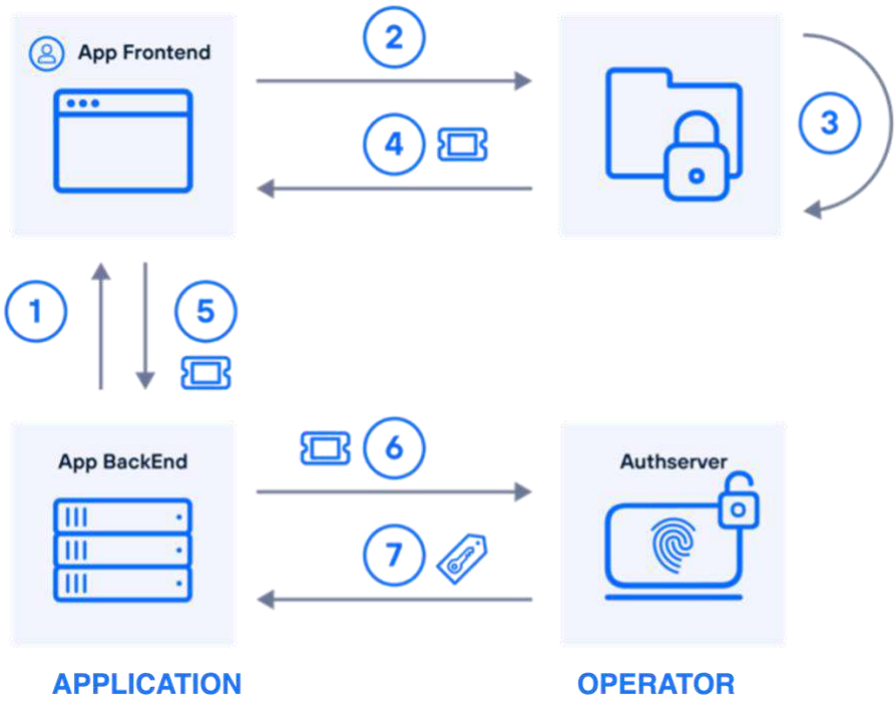
- **Aggregate subscribers and operators:** Channel partners facilitates the integration of applications to multiple operators, ensuring that a regional coverage can be easily achieved for applications. Developer only needs to integrate once to reach multiple operators, without knowing which operator manages each of the subscribers whose data is being accessed. *One application accessing multiple operators.*

Channel partners may also provide a developer friendly interface for the access of APIs, for instance by an integrated SDK that enables the access to Open Gateway (both consume APIs but also access control or access token management). Channel partner may also enable the usage of multiple APIs in one single step with SDK.

4.3 Subscriber authentication

In the case of Number Verification API, the access token not only provides information of the consumption chain to manage privacy and security, but also allows to properly identify the subscriber whose data is being accessed. A standard front-end access token retrieval flow (OAuth2 Authorization Code grant) is employed to ensure the proper identification of the user’s connection/device, by identifying the connection in the operator’s network.

In a simplified approach, the frontend access token generation is depicted in the following diagram.



1. Application backends request the application frontend to initiate the AuthCode OAuth2 flow.
2. Device browser redirects the request to the authorization endpoint of the operator.
3. The request is authenticated in the operator's auth server based on the device's connection. Operators may use the IP origin address or employ other network mechanisms to authenticate the connection, such as Header Enrichment. A lawful base is checked, and consent is captured in the device's browser, if necessary.
4. The authserver provides a valid AuthCode to the application as response.
5. Application frontend redirects the code to the application backend.
6. The application backend requests an access token to the authserver, providing the code as input.
7. The authserver provides finally the API access token, so application can directly consume the Number Verification API (or the specific required API).

The detailed sequence diagram can be found as part of the API standard definition in CAMARA, where the standard OAuth2 Authorization Code grant is detailed for the usage of Number Verification API.

In an Open Gateway flow where a channel partner can be involved between the application and the operator(s), the complete flow requires additional interactions with the channel partner (detailed in Annex - Detailed Flows). Channel partner, by providing

an API access SDK, will simplify and hide the complete flow, so developer will not need to consider the interactions between aggregator and operator.

5. API Documentation

The Number Verification API is standardized in CAMARA, and the corresponding documentation is in this repository at GitHub:

https://github.com/camaraproject/NumberVerification

The version of the specification explained in this document is 0.3.1, which can be found in this link:

https://github.com/camaraproject/NumberVerification/blob/main/code/API_definitions/number_verification.yaml

The operations defined in the specification have been explained in this document in the section How to use the API: workflow. The CAMARA specification itself includes more documentation about the objective and use of the API.

6. Conclusions

Anti-fraud tools include ensuring the identity of the user who is accessing the services, where Number Verification CAMARA API can improve security and user experience. Reducing the risk of fraudulent access to applications like bank accounts, while improving the conversion rate of applications with a better and transparent interaction flow for the customers. Different use cases are considered from the registration process when users identity is validated, to securing the login or access to applications.

Number Verification API can be used also in collaboration with other Open Gateway services for reaching a deeper protection and user's validation, like SIM SWAP (prevent SIM related frauds) or Device Location Verification (location authentication factor for ensuring data identity).

The integration of the API and the different architectures have been identified and detailed, providing a deep introduction on the usage and integration of Number Verification CAMARA API.

7. Other relevant information

You can join now the Telefónica Open Gateway Developer Hub to test our API, develop use cases with the power of the network and improve user experiences.
Join Developer Hub

If you are interested in the potential of Telefónica Open Gateway and you are willing to collaborate with us, you can access our exclusive Partner Program:
Join Partner Program

For further questions about the initiative, don't hesitate to contact our experts:
Contact our experts

8. References and Additional Resources

8.1 Additional information about Telefónica Open Gateway Initiative

Learn more about the Number Verification API and other Open Gateway APIs and services in Telefónica in our website: https://opengateway.telefonica.com/

8.2 Additional information of the Number Verification CAMARA API

The Number Verification CAMARA API official documentation is collected in the following GitHub Repository:
https://github.com/camaraproject/NumberVerification

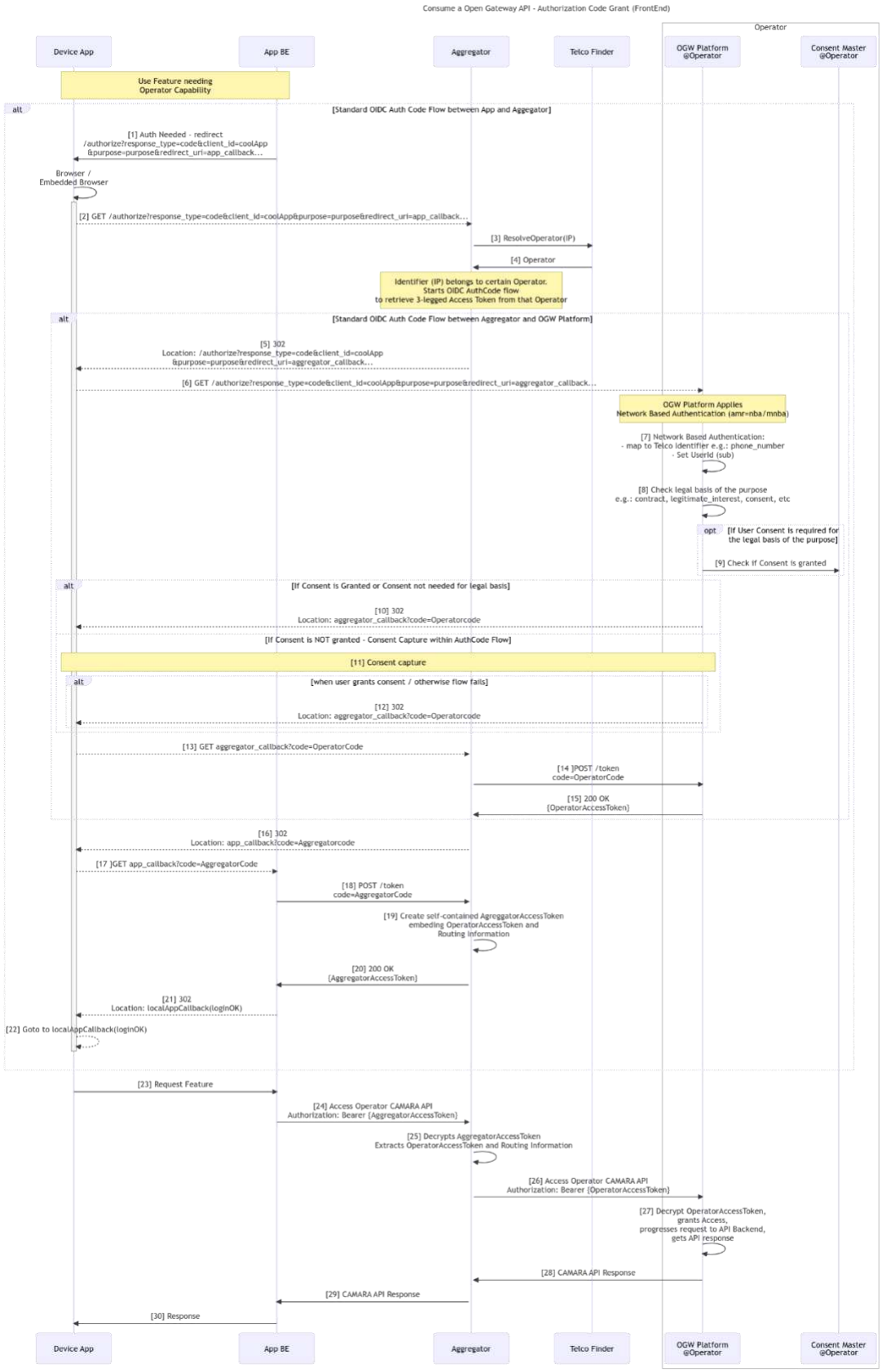
8.3 Glossary of Terms

TERM	DEFINITION
Aggregator	Aggregator or 'Channel Partners' aggregate Operator's CAMARA standardized APIs to build Open Gateway-based services and implement Operator end-point routing based on final user identification on the network.
API Gateway	An intermediary platform that allows communication between different systems and APIs, providing a centralized and standardized approach for accessing and utilizing APIs. The Open Gateway operator platform is the API GW platform in the operator that exposes standardized APIs so third-party services can consume them in a secure and consistent way. Operator platform APIs are based on REST/HTTP. OAuth 2.0 and OpenID Connect are standard security mechanisms to control access to the APIs. APIs are reachable from the Internet and all traffic is encrypted with TLS.
AuthCode	Authentication method to validate the user's identity during the authentication process.
CAMARA	CAMARA is an open-source project within Linux Foundation to define, develop and test the APIs. CAMARA works in close collaboration with the GSMA Operator Platform Group to align API requirements and publish API definitions and APIs. Harmonization of APIs is achieved through fast and agile created working code with developer-friendly documentation. API definitions and reference implementations are free to use (Apache2.0 license). The tool to manage the work and outcomes of the APIs standardization at CAMARA is GitHub: https://github.com/camaraproject
Consent	The explicit permission given by the user for the processing of their personal data, as required by privacy regulations such as GDPR (General Data Protection Regulation).

CSP	Communication Service Provider, refereed to a telecommunications operator that is providing a mobile/fixed/satellite connection to a set of network subscribers. In case of mobile network, they are usually known as MNO or Mobile Network Operators.
IDP	Identity Provider, a service that authenticates and verifies the identity of users.
MSISDN	Mobile Station International Subscriber Directory Number, aka phone number, is the unique ID that identifies a device or user in a phone call or in a data session. It's the public ID that CSPs globally use to identify their subscriber's lines.
Network Subscriber	User of the operator's network. In Open Gateway, subscriber is the network user whose data of configuration is being accessed or modified by the API.
Open Gateway	An industry initiative led by GSMA (Global System for Mobile Communications Association) that transforms telecom networks into future-ready platforms, enabling seamless integration and access to telco capabilities through standardized APIs. Sometimes referred as OGW.
Open Code Repository	A platform or repository where developers can access and collaborate on open-source code and projects, such as GitHub.
OAuth 2.0 / OpenID Connect	Standards and protocols for user authentication and authorization, allowing secure access to APIs and services.
Privacy-by-Default	A principle that ensures privacy protection is integrated into systems and processes by default, requiring explicit user consent for the processing of personal data.
SIM Swap	This refers to the act of asking a telco operator for a new SIM card arguing that the original one has been lost, damaged or stolen. When the new SIM card begins to be in force, the old one becomes invalid, and it is said that a SIM swap has happened. This can be a legitimate action. But

	this API helps to detect whether it happens due to an illegitimate action or not.
SDK	Software Development Kit, a set of tools, libraries, and documentation that enables developers to build applications for a specific platform or system.
User Identifier	A unique identifier associated with a user, such as an IP address or MSISDN, used for authentication, routing, and identification purposes.

9. Annex - Detailed Flows



Flow description:

First, the application backend instructs the application frontend in the device to initiate the OIDC authorization code flow with the aggregator (Steps 1-22).

As per the standard authorization code flow, the device application is redirected to the aggregator's authorization endpoint (Steps 1-2), providing a redirect_uri (app_callback) pointing to the application backend (where the auth code will eventually be sent), as well as the purpose for accessing the data.

The aggregator receives the request from the device application and collects the IP from which the device application is accessing. The aggregator will first discover the operator for the connection in the device where the application is running by querying the telco finder (Steps 3-4).

Once the telco operator is known, the aggregator itself initiates the OIDC authorization code flow with the operator (Steps 5-15).

As per the standard authorization code flow, the device application is redirected to the authorize endpoint of the OGW platform (Steps 5-6), providing a redirect_uri (aggregator_callback) pointing to the aggregator where the auth code will be sent, as well as the purpose for accessing the data originally sent by the device application to the aggregator.

The OGW platform receives the request from the device application and does the following:

- Use network-based authentication mechanism to obtain the user identifier, i.e.: MSISDN. Set the OAuth sub to the unique user ID in the operator (Step 7).
- Check if user consent is required, which depends on the lawful basis associated with the purpose ("legitimate interest", "contract", "consent", etc.). If necessary, it will check in the operator's consent master whether user consent has already been given for this identifier and for the requested purpose(s) (Steps 8-9).

Then, two alternatives may occur:

Scenario 1: User consent is not required, or consent is already given (Step 10). The OGW platform will continue the authorization code flow by redirecting to the aggregator's redirect_uri (aggregator_callback) and including the authorization code (OperatorCode).

Scenario 2: Consent is required and not yet provided by user (Steps 11-12)

- The operator performs the consent capture. Since the authorization code grant involves the frontend, the consent can be captured directly from the user.
- Once the user has given consent, the authorization code flow continues by redirecting to the aggregator's redirect_uri (aggregator_callback) and including the authorization code (OperatorCode).

Once the aggregator receives the redirect with the authorization code (OperatorCode - Step 13), it will retrieve the access token from the OGW operator platform (OperatorAccessToken) (Steps 14-15).

Then the aggregator will continue the authorization code flow by redirecting to the application's redirect_uri (app_callback) and including the authorization code (AggregatorCode - Steps 16-17).

The application will request an access token to aggregator (Step 18). Aggregator will create a new access token, AggregatorAccessToken (*), by creating a JWT extended with additional claims that will carry (Step 19):

- The access token created by the operator (OperatorAccessToken).
- Routing information to know where to route later API calls using the AggregatorAccessToken.

(*)NOTE: As mentioned above, there are other ways to implement the same concept. For example, the aggregator could store the OperatorAccessToken (as well as the necessary routing information) in a database and use a reference token to access it.

The AggregatorAccessToken created is encrypted so that no relevant information is exposed.

The AggregatorAccessToken is made available to the application (Step 20). The application will complete the OIDC flow by interacting backend with frontend (Steps 21-23).

Now the application has a valid access token that can be used to invoke the CAMARA API offered by the aggregator (Step 24).

The aggregator will decrypt the access token, check its validity, find the routing information inside, and extract the OperatorAccessToken (Step 25). Then it will forward the CAMARA API request unchanged to the operator, including the OperatorAccessToken in the request (Step 26). Finally, the operator will validate OperatorAccessToken, grant access to the API based on the scopes bound to the access token, progress request to the corresponding API backend and retrieve the API response (Step 27).

Finally, the operator will provide API response to the aggregator (Step 28) which will be then sent to the application (Steps 29-30).

opengateway.telefonica.com