Guida Tecnica all’integrazione dei servizi in App IO

Data documento 25/07/202
Versione documento v1.2
Versione API 1.1.0

Indice

Indice **2**

Nomenclatura **4**

Introduzione **4**

1. Primi passi nel Developer Portal **5**

2 Abilitazioni **6**
- 2.1 Abilitazione invio messaggi a codici fiscali test 7
- 2.2 Abilitazione invio massivo 7
- 2.3 Abilitazione invio avvisi di pagamento pagoPA 7
- 2.4 Abilitazione subscription feed 7
- 2.5 Abilitazione gestione dei servizi 7

3. Messaggi **8**
- 3.1 Procedura invio messaggi 8
  - 3.1.1 Controllo pre-invio 8
  - 3.1.2 Invio messaggio 9
  - 3.1.3 Controllo post-invio 10
- 3.2 API 11
  - 3.2.1 Get a User Profile using POST 11
  - 3.2.2 Get a User Profile 12
  - 3.2.3 Submit a Message passing the user fiscal_code in the request body 12
  - 3.2.4 Submit a Message passing the user fiscal_code as path parameter 14
  - 3.2.5 Get Message 15
  - 3.2.6 Get Subscriptions Feed 17

4. Servizi **18**
- 4.1 Attributi 19
- 4.2 API 24
  - 4.2.1 Create Service 24
  - 4.2.2 Get User Services 25
  - 4.2.3 Get Service 25
  - 4.2.4 Update Service 26
  - 4.2.5 Regenerate Service Key 27
  - 4.2.6 Upload service logo 27
  - 4.2.7 Upload organization logo 28
- 4.3 Test visualizzazione in App IO dei servizi 29

5. API possibili errori **32**

FAQ **33**
- 1. Posso cancellare un servizio? 33

2. Posso visualizzare un servizio di test ancora non visibile in App IO? 33
3. Posso inviare i messaggi al mio codice fiscale? 34
4. Perchè ci sono due api-key per servizio? 34
5. E’ possibile pagare avvisi pagoPA in App IO? 34
6. Cosa succede se l’ente creditore modifica l’importo dell’avviso pagoPA? 34

Nomenclatura

Utente: soggetto iscritto al developer portal https://developer.io.italia.it/
Servizio: prestazione erogata da un ente
Cittadino: soggetto utilizzatore dell'App IO

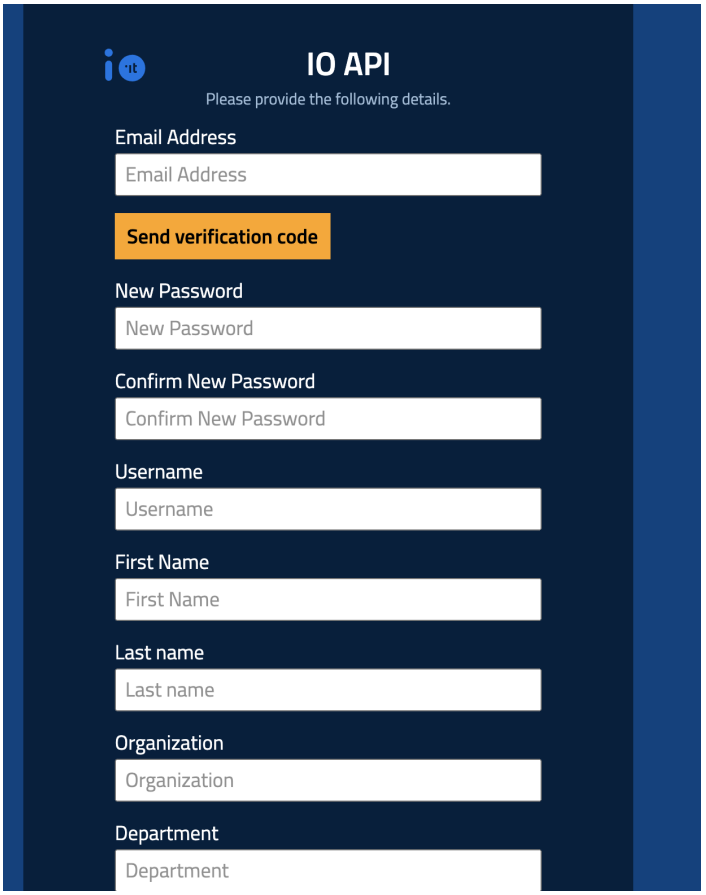
Introduzione

Il presente documento è una guida all'utilizzo dell'API di IO e l'integrazione dei servizi pubblici:

1. la definizione delle api è disponibile all'indirizzo https://developer.io.italia.it/openapi.html
2. per utilizzare le api è necessario iscriversi al developer portal disponibile all'indirizzo https://developer.io.italia.it

1. Primi passi nel Developer Portal

Il primo passo per utilizzare le api di IO è l'**iscrizione al developer portal** (https://developer.io.italia.it).



Per completare l'iscrizione sarà richiesto di validare:

1. un indirizzo email
2. un numero di cellulare
3. dati anagrafici e di riferimento dell'Ente

Al termine del processo di registrazione entrare nel developer portal e **creare il primo servizio** (o sottoscrizione):

1. entrare nella sezione "Profilo (sottoscrizioni)" nella colonna a sinistra
2. se necessario modificare i dati del servizio precompilati con i dati inseriti al momento della registrazione al developer portal
3. creare un servizio cliccando sul pulsante "Aggiungi sottoscrizione"
4. visualizzare e salvare le chiavi segrete (api-key) associate al servizio

Al termine di questi passi preliminari l'utente sarà in grado di utilizzare esclusivamente le seguenti api di IO utilizzando il cittadino di test con codice fiscale **AAAAAA00A00A000A**:

1. Submit a Message passing the user fiscal_code in the request body
2. Submit a Message passing the user fiscal_code as path parameter
3. Get Message

4. Get a User Profile using POST
5. Get a User Profile

L'invio dei messaggi in questa prima fase preliminare si tradurrà nell'invio di una mail all'indirizzo email inserito in fase di registrazione al portale stesso.
A questo punto è possibile **inviare il primo messaggio** al cittadino di test AAAAAA00A00A000A utilizzando le API di IO. Di seguito è riportato un esempio a cui deve essere sostituito il valore **__YOUR_API_KEY__** con una delle api-key del servizio generato in precedenza.

Di seguito viene riportato un esempio di chiamata all'API tramite il comando curl, eseguibile da terminale, in alternativa è possibile utilizzare software di gestione API (es. POSTMAN) avendo accortezza di compilare i dati dell'header e del Content-Type, in particolare la proprietà di API KEY identificato da 'Ocp-Apim-Subscription-Key':

```
### REQUEST
curl --location --request POST
'https://api.io.pagopa.it/api/v1/messages/AAAAAA00A00A000A' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__' \
--header 'Content-Type: application/json' \
--data-raw '{
    "content": {
        "subject": "My first IO app message with min 10 character",
        "markdown": "This is my first message to the IO app. Use
body markdown format with min 80 character"
    }
}'

### RESPONSE
{
  "id": "01EM6X4JB9VSZTQ8H16KMQFCEJ"
}
```

2 Abilitazioni

Gli utenti iscritti al developer portal hanno la possibilità di interagire esclusivamente con il cittadino di test con codice fiscale **AAAAAA00A00A000A**. Per poter inviare messaggi a cittadini reali e lo sblocco di altre api è necessario richiedere un'abilitazione esplicita.

2.1 Abilitazione invio messaggi a codici fiscali test

Per poter essere abilitati all'invio di messaggi verso dei codici fiscali reali e appartenenti agli utenti stessi, occorre contattare gli amministratori del Developer Portal all'indirizzo email onboarding@io.italia.it specificando:

1. email dell'utente iscritto al developer portal
2. id servizio (o sottoscrizione)
3. codici fiscali da abilitare alla ricezione di push notification e messaggi all'interno dell'App IO

Il processo di abilitazione prevede l'invio di un messaggio in App IO ai codici fiscali indicati contenente un codice di verifica univoco. Per completare l'abilitazione, l'utente dovrà rimandare il codice di verifica ricevuto come indicato nelle istruzioni inviate in App IO.

2.2 Abilitazione invio massivo

Per poter essere abilitati all'invio verso qualsiasi codice fiscale reale è necessario essere abilitati all'invio massivo. Questa abilitazione sarà concessa solo a seguito degli adempimenti contrattuali e il superamento della fase di test.

2.3 Abilitazione invio avvisi di pagamento pagoPA

Per poter essere abilitati all'invio di avvisi di pagamento pagoPA in fase di test occorre contattare gli amministratori del Developer Portal all'indirizzo email onboarding@io.italia.it specificando:

1. email dell'utente iscritto al developer portal
2. id servizio (o sottoscrizione)

Per la fase di test saranno abilitati pagamenti per importo massimo di 0,10€.

2.4 Abilitazione subscription feed

Per poter essere abilitati al subscription feed occorre contattare gli amministratori del Developer Portal all'indirizzo email onboarding@io.italia.it specificando:

1. email dell'utente iscritto al developer portal

Questa abilitazione è limitata agli utenti che afferiscono a servizi sviluppati per enti centrali (es ministeri, enti di livello nazionale, ecc).

2.5 Abilitazione gestione dei servizi

Le api dei servizi permettono agli utenti la completa gestione autonoma dei servizi pubblicabili in App IO. Per l'utilizzo di queste api è necessario effettuare una richiesta formale di abilitazione inviando una mail a onboarding@io.italia.it indicando:

1. il nome utente utilizzato per l'iscrizione al developer portal
2. l'ente o gli enti per cui l'utente opera

L'abilitazione sarà concessa se l'utente è in possesso di queste condizioni:

1. l'utente deve essersi iscritto al developer portal e aver creato almeno un servizio
2. l'utente compare come delegato nel contratto di adesioni sottoscritto da uno o più enti

3. Messaggi

L'invio di messaggi è la prima tipologia di comunicazione messa a disposizione dalle API di IO. I messaggi sono comunicazioni personali dirette ad uno specifico cittadino identificato tramite il suo codice fiscale. Non è possibile inviare messaggi ad un gruppo di cittadini, ogni messaggio deve essere personale e diretto ad ogni cittadino.

3.1 Procedura invio messaggi

Il processo di invio dei messaggi prevede di eseguire tre step:

1. controllo pre-invio
2. invio messaggio
3. controllo post-invio

Nel seguito saranno definiti i controlli da effettuare per ogni step con un esempio finale, mentre per la documentazione completa delle api è necessario far riferimento alla documentazione specifica.

Importante: Tutte le api di IO possono restituire delle risposte di errore quindi è necessario implementare lato client dei meccanismi per la corretta di questo tipo di risposte. Ad esempio tutte le api possono restituire lo status code 429 che rappresenta un segnale di sovraccarico dell'infrastruttura di IO: in questo caso è necessario implementare un meccanismo di retry e diminuire il rate delle richieste inserendo delle pause.

3.1.1 Controllo pre-invio

Prima di inviare un messaggio ad un cittadino è necessario effettuare i seguenti controlli:

1. il cittadino è iscritto ad IO
2. il cittadino non ha disattivato le comunicazioni del servizio (di default i cittadini sono iscritti a tutti i servizi, ma è possibile che un cittadino abbia disattivato le comunicazioni da un servizio)

Il controllo pre-invio può essere effettuato in due modalità:

1. utilizzare l'api Get a User Profile (o in alternativa Get a User Profile using POST) per ogni codice fiscale a cui si vuole inviare un messaggio e controllare i valori della risposta. La risposta è ritenuta positiva se entrambe le condizioni sono verificate:
    a. lo status code della risposta è **200**
    b. nel body di risposta il campo **sender_allowed=true**

2. utilizzando l'api Get Subscriptions Feed che permette di effettuare il download degli hash dei codici fiscali dei cittadini iscritti/disiscritti ad un servizio in un determinato giorno. Lo scopo del subscription feed è mettere a disposizione degli enti centrali (es. ministeri, enti di livello nazionale, ecc) uno strumento per minimizzare le chiamate verso l'infrastruttura di IO attraverso un filtro costruito dall'ente stesso. Le informazioni del subscription feed sono aggiornate al giorno precedente. Per la costruzione del subscription feed è necessario interrogare il servizio per ogni giorno a partire dalla data del 24/03/2020 fino alla data attuale. Questa modalità è riservata agli enti centrali o di livello nazionale.

Esempio d'uso con la versione POST di Get a User Profile using POST:

```
### REQUEST
curl --location --request POST
'https://api.io.pagopa.it/api/v1/profiles' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__' \
--header 'Content-Type: application/json' \
--data-raw '{
    "fiscal_code": "AAAAAA00A00A000A"
}'
```

risposta corretta con status 200:

```
### RESPONSE
{
    "sender_allowed": true
}
```

3.1.2 Invio messaggio

I messaggi sono comunicazioni personali dirette ad uno specifico cittadino identificato tramite il suo codice fiscale. Non è possibile inviare messaggi ad un gruppo di cittadini, ogni messaggio deve essere personale e diretto ad ogni cittadino. Per inviare un messaggio è necessario utilizzare l'api Submit a Message passing the user fiscal_code as path parameter (o in alternativa Submit a Message passing the user fiscal_code in the request body). Il dettaglio di queste api è descritto nel paragrafo dedicato alle api.
Importante: L'invio del messaggio avviene in modalità asincrona per cui lo status code 201 indica che il messaggio è stato preso in carico dall'infrastruttura di IO per la successiva elaborazione. L'elaborazione può fallire per cui è necessario

effettuare un controllo post-invio. Per poter effettuare il controllo post-invio è sono richiesti l'ID del messaggio e il codice fiscale del cittadino a cui si riferisce il messaggio.

Esempio di invio messaggio utilizzando Submit a Message passing the user fiscal_code as path parameter

```
### REQUEST
curl --location --request POST
'https://api.io.pagopa.it/api/v1/messages/AAAAAA00A00A000A' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__' \
--header 'Content-Type: application/json' \
--data-raw '{
    "content": {
        "subject": "My first IO app message with min 10 character",
        "markdown": "This is my first message to the IO app. Use
body markdown format with min 80 character"
    }
}'

### RESPONSE
{
  "id": "01EM6X4JB9VSZTQ8H16KMQFCEJ"
}
```

3.1.3 Controllo post-invio

Il controllo dell'invio del messaggio avviene tramite l'utilizzo dell'ID del messaggio ottenuto al passo precedente contestualmente al codice fiscale del cittadino. Per effettuare la verifica viene usata l'api Get Message.
Qui è importante verificare lo status del messaggio, che dovrà trovarsi nello status di "Processed" verificabile dalla risposta ottenuta dalla API.

Esempio d'uso con la versione GET di Get Message utilizzando CURL:

```
###  REQUEST
curl --location --request GET
'https://api.io.pagopa.it/api/v1/messages/AAAAAA00A00A000A/01EM6X4JB
9VSZTQ8H16KMQFCEJ' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__'
```

```
### RESPONSE
{
   "message": {
       "content": {
           "subject": "My first IO app message with min 10
character",
           "markdown": "This is my first message to the IO app. Use
body markdown format with min 80 character"
       },
       "created_at": "2021-02-18T08:17:01.775Z",
       "fiscal_code": "AAAAAA00A00A000A",
       "id": "01EM6X4JB9VSZTQ8H16KMQFCEJ",
       "sender_service_id": "01EYNQ0864HKYR1Q9PXPJ18W7G"
   },
   "notification": {
       "email": "SENT",
       "webhook": "SENT"
   },
   "status": "PROCESSED"
}
```

3.2 API

3.2.1 Get a User Profile using POST

https://developer.io.italia.it/openapi.html#operation/getProfileByPOST
Api che permette di controllare che il cittadino identificato tramite codice fiscale sia iscritto ad IO e che il servizio possa inviare comunicazioni al cittadino stesso.
Il codice fiscale del cittadino andrà inserito nel body della post request.
La risposta è ritenuta positiva se entrambe le condizioni sono verificate:

1) lo status code della risposta è **200**
2) nel body di risposta il campo **sender_allowed=true**

Api equivalente a Get a User Profile.

```
### REQUEST
```

```
curl --location --request POST
'https://api.io.pagopa.it/api/v1/profiles' \
--header 'Content-Type: application/json' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__' \
--data-raw '{
   "fiscal_code": "AAAAAA00A00A000A"
}'

### RESPONSE
{
    "sender_allowed": true
}
```

3.2.2 Get a User Profile

https://developer.io.italia.it/openapi.html#operation/getProfile

Attenzione: la seguente API sarà a breve deprecata, pertanto se ne sconsiglia l'uso.

Api che permette di controllare che il cittadino identificato tramite codice fiscale sia iscritto ad IO e che il servizio possa inviare comunicazioni al cittadino stesso.

La risposta è ritenuta positiva se entrambe le condizioni sono verificate:

3) lo status code della risposta è **200**
4) nel body di risposta il campo **sender_allowed=true**

```
### REQUEST
curl --location --request GET
'https://api.io.pagopa.it/api/v1/profiles/AAAAAA00A00A000A' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__'

### RESPONSE
{
    "sender_allowed": true
}
```

3.2.3 Submit a Message passing the user fiscal_code in the request body

https://developer.io.italia.it/openapi.html#operation/submitMessageforUserWithFiscalCodeInBody

Api per l'invio di messaggi verso un cittadino identificato tramite codice fiscale. Prima di inviare un messaggio è importante verificare che il cittadino sia iscritto ad IO e che il servizio possa inviare comunicazioni al cittadino stesso.

time_to_live [deprecated]
Descrizione: Tempo espresso in secondi che specifica il tempo di retry di delivery del messaggio
Obbligatorio: no
Default: 3600
Tipo: intero
Esempio: 3600

content
dati relativi al content:

subject
Descrizione: Titolo del messaggio la cui lunghezza deve essere compresa tra 10 e 120 caratteri
Obbligatorio: si
Tipo: stringa
Esempio: "My first IO app message with min 10 character"

markdown
Descrizione: Testo del messaggio in formato markdown la cui lunghezza deve essere compresa tra 80 e 10000 caratteri. Esempio di editor online per fare dei test https://stackedit.io/app#
Obbligatorio: si
Tipo: stringa
Esempio: "This is my first message to the IO app. Use body markdown format with min 80 character"

due_date
Descrizione: Proprietà che permette di associare al messaggio un promemoria. Il formato data deve essere ISO-8601 e timezone UTC.
Obbligatorio: no
Tipo: stringa
Esempio: "2018-10-13T00:00:00.000Z"

payment_data
dati relativi al payment_data:

amount
Descrizione: Importo in eurocents dell'avviso di pagamento emesso su piattaforma pagoPA. Per l'invio degli avvisi di pagamento è necessario richiedere specifica l'abilitazione.

Obbligatorio: si per pagamenti pagoPA
Tipo: intero
Esempio: 100

notice_number
Descrizione: Codice avviso di un avviso di pagamento emesso su piattaforma pagoPA. Per l'invio degli avvisi di pagamento è necessario richiedere specifica l'abilitazione. E' importante che il codice fiscale del servizio mittente corrisponda al codice fiscale dell'ente creditore che emette l'avviso pagoPA.
Obbligatorio: si per pagamenti pagoPA
Tipo: stringa
Esempio: "301011100007347557"

invalid_after_due_date
Descrizione: In app visualizza il pagamento come scaduto se la data attuale è successiva a due_date.
Obbligatorio: no
Default: false
Tipo: booleano
Esempio: false

prescription_data [beta-reserved]

```
### REQUEST
curl --location --request POST
'https://api.io.pagopa.it/api/v1/messages' \
--header 'Content-Type: application/json' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__' \
--data-raw '{
    "content": {
    "subject": "Welcome new user !",
    "markdown": "# This is a markdown header\n\nto show how easily
markdown can be converted to **HTML**\n\nRemember: this has to be a
long text."
    },
    "fiscal_code": "AAAAAA00A00A000A"
}'

### RESPONSE
{
  "id": "01EM6X4JB9VSZTQ8H16KMQFCEJ"
}
```

3.2.4 Submit a Message passing the user fiscal_code as path parameter

https://developer.io.italia.it/openapi.html#operation/submitMessageforUser
Attenzione: la seguente API sarà a breve deprecata, pertanto se ne sconsiglia l'uso.

Api equivalente a Submit a Message passing the user fiscal_code in the request body.

```
### REQUEST
curl --location --request POST
'https://api.io.pagopa.it/api/v1/messages/AAAAAA00A00A000A' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__' \
--header 'Content-Type: application/json' \
--data-raw '{
    "content": {
        "subject": "My first IO app message with min 10 character",
        "markdown": "This is my first message to the IO app. Use
body markdown format with min 80 character"
    }
}'

### RESPONSE
{
  "id": "01EM6X4JB9VSZTQ8H16KMQFCEJ"
}
```

3.2.5 Get Message

https://developer.io.italia.it/openapi.html#operation/getMessage
Api che controlla lo stato di invio del messaggio recuperando il contenuto. E' necessario interrogare l'API con il codice fiscale del cittadino oggetto del messaggio e l'identificativo del messaggio.

message
dati relativi al message

created_at
Descrizione: data di creazione del messaggio formato

fiscal_code
Descrizione: codice fiscale del cittadino a cui è stato inviato il messaggio

id

Descrizione: identificativo del messaggio ottenuto con la Submit a message

sender_service_id
Descrizione: identificato del servizio associato al messaggio

content
dati relativi al content sono gli stessi inviati tramite il submit del messaggio

notification
dati relativi a notification

email
Descrizione: stato invio del messaggio
Tipo: stringa
- SENT: email inviata correttamente
- THROTTLED: errore temporaneo per sovraccarico, il messaggio potrà essere recapitato entro il TTL e per un massimo di 7 giorni
- EXPIRED: raggiunto il massimo TTL del messaggio
- FAILED: errore permanente della notifica

webhook
Descrizione: stato invio notifica push
- SENT: notifica inviata
- THROTTLED: errore temporaneo per sovraccarico, il messaggio potrà essere recapitato entro il TTL e per un massimo di 7 giorni
- EXPIRED: raggiunto il massimo TTL del messaggio
- FAILED: errore permanente della notifica
Tipo: stringa
Esempio: “SENT”

status
Descrizione: stato dell’invio del messaggio.
- ACCEPTED: il messaggio è stato inserito in coda per il salvataggio
- THROTTLED: errore temporaneo per sovraccarico, il messaggio potrà essere recapitato entro il TTL e per un massimo di 7 giorni
- FAILED: errore permanente nel salvataggio del messaggio
- PROCESSED: il messaggio è stato inviato
- REJECTED: messaggio scartato perchè il destinatario non esiste o ha bloccato le comunicazioni dal servizio
Tipo: stringa
Esempio: “PROCESSED”

```
###  REQUEST
```

```
curl --location --request GET
'https://api.io.pagopa.it/api/v1/messages/AAAAAA00A00A000A/01EM6X4JB
9VSZTQ8H16KMQFCEJ' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__'

### RESPONSE
{
   "message": {
       "content": {
           "subject": "My first IO app message with min 10
character",
           "markdown": "This is my first message to the IO app. Use
body markdown format with min 80 character"
       },
       "created_at": "2021-02-18T08:17:01.775Z",
       "fiscal_code": "AAAAAA00A00A000A",
       "id": "01EM6X4JB9VSZTQ8H16KMQFCEJ",
       "sender_service_id": "01EYNQ0864HKYR1Q9PXPJ18W7G"
   },
   "notification": {
       "email": "SENT",
       "webhook": "SENT"
   },
   "status": "PROCESSED"
}
```

3.2.6 Get Subscriptions Feed

https://developer.io.italia.it/openapi.html#operation/getSubscriptionsFeedForDate
Api che permette di ottenere i codici fiscali in modalità hashed che si sono sottoscritti o disiscritti per una specifica data (UTC - YYYY-MM-DD). In questo modo è possibile scaricare gli utenti iscritti ad un servizio (identificato dalla API KEY), facendo una request per ogni giorno a partire dal 24/03/2020 e salvando i dati nella propria infrastruttura.
Per una risposta corretta ottengo:

1. lo status code della risposta è **200**
2. nel body di risposta il campo **subscriptions** e **unsubscriptions**

Lo scopo del subscription feed è mettere a disposizione degli enti centrali uno strumento per minimizzare le chiamate verso l'infrastruttura di IO attraverso un filtro applicato dall'ente stesso.
In nessun caso sarà possibile richiedere i codici fiscali in chiaro.
Prima che un servizio venga messo visibile in app l'Api restituirà l'elenco completo degli hash dei codici fiscali dei cittadini iscritti/disiscritti ad IO in un determinato giorno.

```
### REQUEST
curl --location --request GET
'https://api.io.pagopa.it/api/v1/subscriptions-feed/2020-02-23' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__'

### RESPONSE
{
    "dateUTC": "2020-02-23",
    "subscriptions": [
"e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855"
],
    "unsubscriptions": [ ]
}
```

4. Servizi

I servizi sono le prestazioni erogate dall'ente verso i cittadini. Per l'inserimento dei servizi è possibile utilizzare due modalità:

1) creazione del servizio e richiesta di pubblicazione tramite il developer portal
2) gestione autonoma dei servizi tramite api

In questo capitolo sarà trattata esclusivamente la gestione dei servizi tramite api che necessita la richiesta un'abilitazione specifica al Team di IO.
Importante: la proprietà **is_visible** deve sempre essere impostata a **false** per i servizi di test.

4.1 Dati obbligatori

Per pubblicare un **servizio in produzione** è **obbligatorio** inserire le seguenti informazioni:

1) service_name
2) organization_name
3) organization_fiscal_code

4) authorized_cidrs
5) is_visible
6) max_allowed_payment_amount (solo per servizi che prevedono l'invio di avvisi pagoPA)
7) service_metadata
    a) scope
    b) privacy_url
    c) description
    d) almeno uno o più canali di contatto diretto a cui il cittadino può chiedere assistenza (phone, mail, pec, support_url)

Inoltre è **obbligatorio** caricare il logo del servizio e dell'organizzazione con dimensioni 300x300 in formato png sfondo bianco o trasparente.
Dopo aver inserito le informazioni obbligatorie sarà possibile mettere il servizio in produzione cambiando opportunamente la proprietà **is_visible**.

4.2 Attributi

service_name
Descrizione: Nome del servizio visualizzato in App IO.
Obbligatorio: si
Esempio: "Servizio avviso Carta di Identità"

department_name
Descrizione: Nome dell'Unità Operativa dell'ente che eroga il servizio
Obbligatorio: si
Tipo: stringa
Esempio: "Anagrafe"

organization_name
Descrizione: Nome ente che eroga il servizio
Obbligatorio: si
Tipo: stringa
Esempio: "Comune di Milano"

organization_fiscal_code
Descrizione: Codice fiscale dell'ente che eroga il servizio
Obbligatorio: si
Tipo: stringa
Esempio: "00907501001"

authorized_cidrs
Descrizione: Lista ip/subnet autorizzate per richiamare il servizio. Di default non sono applicate restrizioni, è obbligatorio inserire le restrizioni ip prima del passaggio in produzione.

Obbligatorio: si, prima del passaggio in produzione
Tipo: lista di stringhe
Esempio: “1.2.3.4/32”

version
Descrizione: Numero di versione del servizio. Il versionamento è automaticamente gestito dall’infrastruttura di IO e non è necessario inviare questo campo.
Obbligatorio: no
Tipo: intero
Esempio: 1

require_secure_channels
Descrizione: Flag che permette di anonimizzare le notifiche inviate agli utenti (email e push). Il valore di predefinito è false. Scenari:

1. require_secure_channels=false
    a. titolo del messaggio mostrato nella notifica push in App IO
    b. titolo del messaggio e corpo inviato nella notifica email
2. require_secure_channels=true
    a. titolo del messaggio mostrato nella notifica push in App IO generico: “Hai un nuovo messaggio”
    b. titolo del messaggio e corpo inviato nella notifica email generico: “Hai un nuovo messaggio”

Obbligatorio: no
Tipo: booleano
Esempio: false

is_visible
Descrizione: Flag che modifica la visibilità del servizio in App IO. E’ importante impostare il valore a false per i servizi in fase di test.
Obbligatorio: si
Tipo: booleano
Esempio: false

service_id
Descrizione: Identificativo del servizio. E’ il riferimento da usare come parametro nel path della richiesta per invocare i servizi come Update Service e Upload Service Logo. Questa proprietà non è modificabile ed è considerata in sola lettura.
Tipo: stringa
Esempio: “01EXPKTF05Z2AWEN0PN99QZE79”

id
Descrizione: Identificativo versionato del servizio: è la composizione di "service_id"+"version" con zero fill. Questa proprietà non va inviata non va inviata nelle request e sarà aggiornata in automatico ad ogni modifica del servizio. Questa proprietà non è modificabile ed è considerata in sola lettura.

Tipo: stringa
Esempio: “01EXPKTF05Z2AWEN0PN99QZE79-0000000000000001”

authorized_recipients
Descrizione: Lista dei codici fiscali a cui il servizio può inviare messaggi. Alla creazione di un servizio è possibile inviare messaggi soltanto al cittadino di test con codice fiscale AAAAAA00A00A000A. Se l’utente è abilitato all’invio a qualsiasi codice fiscale questa restrizione non si applica.
Obbligatorio: no
Tipo: lista di stringhe
Esempio: “AAAAAA00A00A000A”

max_allowed_payment_amount
Descrizione: Importo massimo autorizzato nei messaggi associati ad un avviso di pagamento pagoPA. L’importo è inteso in eurocents, cioè come parte intera più due cifre decimali. Ad esempio per impostare una soglia di pagamento di 1500,00€ inserire il valore 150000.
Obbligatorio: si se il servizio prevede l’invio di avvisi di pagamento pagoPA
Tipo: intero
Esempio:”1000000”

primary_key
Descrizione: Chiave primaria del servizio da inviare nell’header Ocp-Apim-Subscription-Key. Tramite l’api XXXX sarà generata una nuova chiave primaria. Chiave primaria e secondaria sono equivalenti e sono duplicate per poter effettuare il cambio delle chiavi senza interruzione del servizio.
Tipo: stringa
Esempio:”__YOUR_API_KEY__”

secondary_key
Descrizione: Chiave secondaria del servizio da inviare in alternativa alla chiave primaria nell’header Ocp-Apim-Subscription-Key. Tramite l’api XXXX sarà generata una nuova chiave secondaria. Chiave primaria e secondaria sono equivalenti e sono duplicate per poter effettuare il cambio delle chiavi senza interruzione del servizio.
Tipo: stringa
Esempio:”__YOUR_API_KEY__”

service_metadata
dati relativi al service_metadata:

description
Descrizione: Descrizione del servizio in formato markdown. Esempio di editor online per fare dei test https://stackedit.io/app#
Obbligatorio: si se il flag is_visible è true

Tipo: stringa
Esempio: “testo di prova [IO Italia](https://io.italia.it/)”

web_url
Descrizione: Indirizzo del sito web del servizio
Obbligatorio: no
Tipo: booleano
Esempio: “https://io.italia.it”

app_ios
Descrizione: Se il servizio prevede l'utilizzo di una specifica APP, indicare il link dell'APP sullo store ios.
Obbligatorio: no
Tipo: stringa
Esempio: “https://apps.apple.com/it/app/io/id1501681835”

app_android
Descrizione: Se il servizio prevede l'utilizzo di una specifica APP, indicare il link dell'APP sullo store android.
Obbligatorio: no
Tipo: stringa
Esempio: “https://play.google.com/store/apps/details?id=it.pagopa.io.app”

tos_url
Descrizione: Indirizzo web dei termini e condizioni d’uso del servizio.
Obbligatorio: no
Tipo: stringa
Esempio: “https://io.italia.it/app-content/tos_privacy.html”

privacy_url
Descrizione: Indirizzo web per visualizzare per l’informativa sulla privacy del servizio.
Obbligatorio: si
Tipo: stringa
Esempio: “https://io.italia.it/privacy-policy/”

address
Descrizione: Indirizzo dell’ente/unità operativa che eroga il servizio
Obbligatorio: no
Tipo: stringa
Esempio: “Via Roma, 56”

phone
Descrizione: Numero di telefono dell’ente/unità operativa che eroga il servizio

Obbligatorio: almeno uno o più canali di contatto diretto a cui il cittadino può chiedere assistenza (phone, mail, pec, support_url)
Tipo: stringa
Esempio: “0825889988”

email
Descrizione: Indirizzo email dell’ente/unità operativa che eroga il servizio
Obbligatorio: almeno uno o più canali di contatto diretto a cui il cittadino può chiedere assistenza (phone, mail, pec, support_url)
Tipo: stringa
Esempio: “info@mail.it”

pec
Descrizione: Indirizzo email pec dell’ente/unità operativa che eroga il servizio
Obbligatorio: almeno uno o più canali di contatto diretto a cui il cittadino può chiedere assistenza (phone, mail, pec, support_url)
Tipo: stringa
Esempio: “info@mail.pec.it”

cta
Descrizione: Proprietà che permette la creazione di bottoni personalizzati all’interno della scheda servizio. Questa funzionalità è in beta, non valorizzare questo attributo.
Obbligatorio: no
Tipo: stringa
Esempio: “”

token_name
Descrizione: Nome del token utilizzato per il processo di Single Sign On (SSO) tra App IO e un servizio esposto dall’ente. Questa funzionalità è in beta, non valorizzare questo attributo.
Obbligatorio: no
Tipo: stringa
Esempio: “”

support_url
Descrizione: Indirizzo web per le richieste di assistenza dell’ente/unità operativa che eroga il servizio
Obbligatorio: almeno uno o più canali di contatto diretto a cui il cittadino può chiedere assistenza (phone, mail, pec, support_url)
Tipo: stringa
Esempio: “https://io.italia.it/support”

scope

Descrizione: Proprietà che inserisce il servizio nella lista dei servizi locali o nazionali. Possibili valori:

1. LOCAL per i servizi a carattere locale (es. comuni)
2. NATIONAL per i servizi di interesse nazionale (es ministeri, agenzie/enti nazionali, regioni)

Obbligatorio: si
Tipo: stringa
Esempio: “LOCAL”\\\\\\\\

4.3 Controllo di qualità

Qualora il servizio non abbia tutti i dati indicati come obbligatori al paragrafo 4.1 questo sarà presente nella lista dell’App IO ma con l’indicazione “in arrivo”. I servizi in questo stato non permettono l’invio dei messaggi ai cittadini.

4.4 API

4.4.1 OpenAPI

Web

https://developer.io.italia.it/openapi.html

File

E’ reperibile al seguente indirizzo di github il file openAPI 2.0:
https://github.com/pagopa/io-functions-services/blob/master/openapi/index.yaml

Sono presenti alcune estensioni custom che non saranno bene interpretate da tool aderenti allo standard openAPI 2.0 (fka swagger https://swagger.io/specification/v2/):

- `x-extensible-enum`: dove utilizzata sostituisce quello che nello standard è espresso con `enum` è in corso la sua sostituzione con `enum` standard.
- `x-import`: per alcune definition specifica dove si trova la definizione del tipo nelle librerie di PagoPA (Typescript), non ha un corrispettivo nello standard OpenAPI. Non è immediatamente evidente quali siano i requisiti di formato di un campo specifico e si deve recuperare manualmente la definizione puntata dalla reference.

- `x-one-of:` usato in combinazione con `allOf` sta ad indicare che `allOf` non rappresenta una intersezione dei tipi elencati ma piuttosto si comporta come `oneOf` introdotto con la versione 3 dello standard OpenAPI. È utilizzato nel tipo `ServicePayload`

- `x-example:` aggiunge un esempio dove lo standard non permette di utilizzare il tag `example`

4.4.2 Create Service

https://developer.io.italia.it/openapi.html#operation/createService
Api che permette ad un utente la creazione di un servizio.
Per la sola creazione del servizio è possibile usare l'api-key di un altro servizio già creato nel developer portal. L'utilizzo delle api come Update Service, Upload Service Logo necessitano delle api-key specifiche del servizio.
Suggerimento: creare un servizio principale non visibile in app e usare l'api-key di questo servizio per creare gli altri servizi.
Importante: **is_visible** deve sempre essere impostato a **false** per i servizi di test.

Esempio con informazioni minime per la creazione di un servizio di test:

```
### REQUEST
curl --location --request POST
'https://api.io.pagopa.it/api/v1/services' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__' \
--header 'Content-Type: application/json' \
--data-raw '{
    "service_name": "Test",
    "department_name": "IT",
    "organization_name": "Test",
    "organization_fiscal_code": "00000000000",
    "authorized_cidrs": [],
    "is_visible": false,
    "service_metadata": {
        "scope": "LOCAL"
    }
}'

### RESPONSE
{
  "authorized_cidrs": [],
  "authorized_recipients": ["AAAAAA00A00A000A"],
  "department_name": "IT",
  "id": "01EYATQMGR0SQT93001Z3ACZVR-0000000000000000",
  "is_visible": false,
```

```
  "max_allowed_payment_amount": 0,
  "organization_fiscal_code": "00000000000",
  "organization_name": "Test",
  "require_secure_channels": false,
  "service_id": "01EYATQMGR0SQT93001Z3ACZVR",
  "service_metadata": { "scope": "LOCAL" },
  "service_name": "Test",
  "version": 0,
  "primary_key": "__YOUR_PRIMARY_API_KEY__",
  "secondary_key": "__YOUR_SECONDARY_API_KEY__"
}
```

4.4.3 Get User Services

https://developer.io.italia.it/openapi.html#operation/getUserServices
Api che restituisce la lista dei servizi di un utente (service_id).

Esempio:

```
### REQUEST
curl --location --request GET
'https://api.io.pagopa.it/api/v1/services' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__'

### RESPONSE
{ "items":
  [
   "01EWZJJ5MBJ34HKZGB2Z0733D2",
   "01EYATQMGR0SQT93001Z3ACZVR"
  ]
}
```

4.4.4 Get Service

https://developer.io.italia.it/openapi.html#operation/getService
Restituisce tutte le informazioni relative ad un servizio. E' obbligatorio utilizzare l'api-key del servizio stesso e inserire il **service_id** come path parameter.

Esempio:

```
### REQUEST
curl --location --request GET
'https://api.io.pagopa.it/api/v1/services/SERVICE_ID' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__'
```

```
### RESPONSE
{
  "authorized_cidrs": [],
  "authorized_recipients": ["AAAAAA00A00A000A"],
  "department_name": "IT",
  "id": "SERVICE_ID-0000000000000000",
  "is_visible": false,
  "max_allowed_payment_amount": 0,
  "organization_fiscal_code": "00000000000",
  "organization_name": "PagoPA",
  "require_secure_channels": false,
  "service_id": "SERVICE_ID",
  "service_name": "Test",
  "version": 0,
  "primary_key": "__YOUR_PRIMARY_API_KEY__",
  "secondary_key": "__YOUR_SECONDARY_API_KEY__"
}
```

4.4.5 Update Service

https://developer.io.italia.it/openapi.html#operation/updateService
Aggiorna le informazioni relative ad un servizio. E' obbligatorio utilizzare l'api-key del servizio stesso e inserire il **service_id** come path parameter.

Esempio:

```
### REQUEST
curl --location --request PUT
'https://api.io.pagopa.it/api/v1/services/SERVICE_ID' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__' \
--header 'Content-Type: application/json' \
--data-raw '{
    "service_name": "Test",
    "department_name": "IT",
    "organization_name": "Test",
    "organization_fiscal_code": "00000000001",
    "authorized_cidrs": [],
    "is_visible": false,
    "service_metadata": {
        "scope": "LOCAL"
    }
}'

### RESPONSE
```

```
{
  "authorized_cidrs": [],
  "authorized_recipients": ["AAAAAA00A00A000A"],
  "department_name": "IT",
  "id": "SERVICE_ID-0000000000000001",
  "is_visible": false,
  "max_allowed_payment_amount": 0,
  "organization_fiscal_code": "00000000001",
  "organization_name": "Test",
  "require_secure_channels": false,
  "service_id": "SERVICE_ID",
  "service_metadata": { "scope": "LOCAL" },
  "service_name": "Test",
  "version": 1,
  "primary_key": "__YOUR_PRIMARY_API_KEY__",
  "secondary_key": "__YOUR_SECONDARY_API_KEY__"
}
```

4.4.6 Regenerate Service Key

https://developer.io.italia.it/openapi.html#operation/regenerateServiceKey
Api che aggiorna le api-ley di un servizio. E' obbligatorio utilizzare l'api-key del servizio stesso, inserire il **service_id** come path parameter e specificare il tipo di chiave da rigenerare (PRIMARY_KEY o SECONDARY_KEY).

Esempio:

```
### REQUEST
curl --location --request PUT
'https://api.io.pagopa.it/api/v1/services/SERVICE_ID/keys' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__' \
--header 'Content-Type: application/json' \
--data-raw '{
    "key_type": "SECONDARY_KEY"
}'

### RESPONSE
{
 "primary_key":"__YOUR_PRIMARY_API_KEY__",
 "secondary_key":"__YOUR_SECONDARY_API_KEY__"
}
```

4.4.7 Upload service logo

https://developer.io.italia.it/openapi.html#operation/uploadServiceLogo

Api che permette di caricare il logo di un servizio. E' obbligatorio utilizzare l'api-key del servizio stesso, inserire il **service_id** come path parameter e inserire nel body del messaggio il logo in formato base64.
Importante: Le dimensioni del logo dovranno essere necessariamente 300x300 pixel immagine in formato png con sfondo bianco o trasparente.
E' possibile che eseguendo un upload del logo immediatamente dopo la creazione del servizio, l'api Upload service logo restituisca un errore 401. In caso di errore ripetere l'Upload service logo o attendere qualche secondo dopo la creazione del servizio.
Per controllare che il logo del servizio sia caricato correttamente interrogare https://assets.cdn.io.italia.it/logos/services/**SERVICE_ID**.png in cui **SERVICE_ID** è l'id servizio in lowercase.

```
### REQUEST
curl --location --request PUT
'https://api.io.pagopa.it/api/v1/services/SERVICE_ID/logo' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__' \
--header 'Content-Type: application/json' \
--data-raw '{
    "logo": "<<Base64ImageString>>"
}'

### RESPONSE
{}
```

4.4.8 Upload organization logo

https://developer.io.italia.it/openapi.html#operation/uploadOrganizationLogo
Api che permette di caricare il logo di una organizzazione. E' obbligatorio utilizzare l'api-key del servizio stesso, inserire il **organization_fiscal_code** come path parameter e inserire nel body del messaggio il logo in formato base64.
Importante: Le dimensioni del logo dovranno essere necessariamente 300x300 pixel immagine in formato png con sfondo bianco o trasparente.
Per controllare che il logo dell'ente sia caricato correttamente interrogare https://assets.cdn.io.italia.it/logos/organizations/**ORGANIZATION_FISCAL_CODE**.png in cui **ORGANIZATION_FISCAL_CODE** è l'organization_fiscal_code privato degli eventuali zeri iniziali del codice fiscale dell'ente.

```
### REQUEST
curl --location --request PUT
'https://api.io.pagopa.it/api/v1/organizations/ORGANIZATION_FISCAL_CODE/logo' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__' \
--header 'Content-Type: application/json' \
--data-raw '{
```

```
    "logo": "<<Base64ImageString>>"
}'

### RESPONSE
{}
```

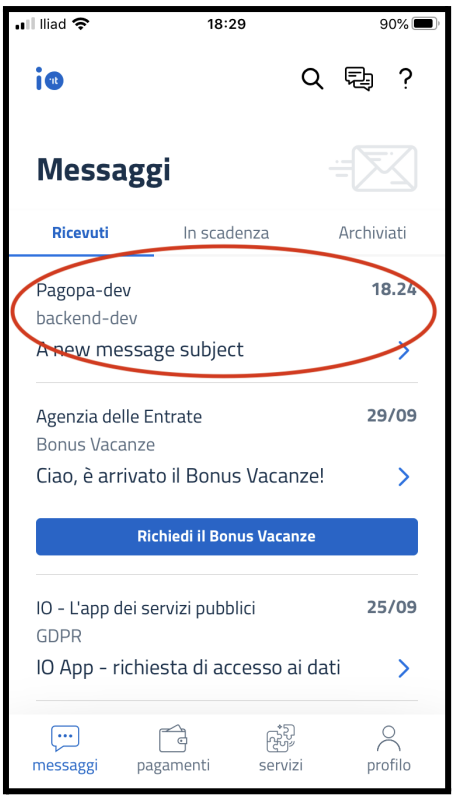
4.5 Test visualizzazione in App IO dei servizi

E' accedere alla scheda di un servizio di test ancora non visibile in App IO (**is_visible**=false) seguendo le istruzioni descritte in questo paragrafo.
Importante: E' necessario che l'utente sia abilitato a poter inviare messaggi al proprio codice fiscale.
Procedura di visualizzazione:

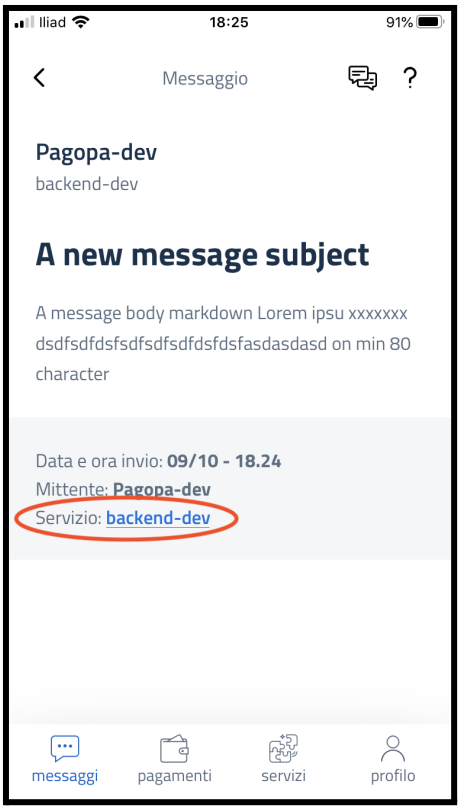
1. creare in servizio di test (dal developer portal o tramite api)
2. tramite il servizio creato, inviare un messaggio al proprio codice fiscale

```
curl --location --request POST
'https://api.io.pagopa.it/api/v1/messages/FISCAL_CODE' \
--header 'Ocp-Apim-Subscription-Key: __YOUR_API_KEY__' \
--header 'Content-Type: application/json' \
--data-raw '{
    content": {
        "subject": "A new message subject",
        "markdown": "A message body markdown Lorem ipsu xxxxxxx
dsdfsdfdsfsdfsdfsdfdsfdsfasdasdasd on min 80 character"
    }
}'
```

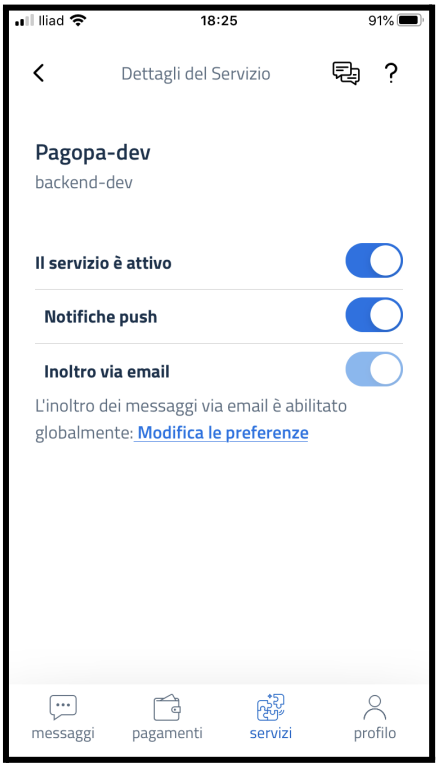
3. Attendere l'arrivo del messaggio in App, se necessario forzare l'aggiornamento dei messaggi effettuando un pull-to-refresh



4. Entrare nel messaggio per visualizzarne il dettaglio



5. Cliccare sul link presente in fondo al messaggio riportante il nome del servizio che ha inviato il messaggio, questo permetterà di visualizzare la scheda del servizio creato



5. API possibili errori

In questa sezione troviamo alcuni dei possibili errori a fronte di un invio di una richiesta alle API.

- Errore 429:

Tutte le api possono restituire lo status code 429 che rappresenta un segnale di sovraccarico dell'infrastruttura di IO: in questo caso è necessario implementare un meccanismo di retry e diminuire il rate delle richieste inserendo delle pause.

- Errore 400:

```
{
```

```
   "detail": "value [undefined] at [root.0] is not a valid
[Exact<NewMessage>]\nvalue [undefined] at [root.1] is not a valid [{
time_to_live: (integer >= 3600 and < 604800 | 604800) }]",
   "status": 400,
   "title": "Invalid (Exact<NewMessage> & { time_to_live: (integer >= 3600 and <
604800 | 604800) })"
}
```


In questo caso l'errore è dovuto ad una errata trasmissione del body, come ad esempio un body non inviato nel formato corretto (JSON)

- Errore 401:


```
{
   "statusCode": 401,
   "message": "Access denied due to invalid subscription key. Make sure to
provide a valid key for an active subscription."
}
```


Accertarsi di aver inserito nell'header il valore corretto della chiave Ocp-Apim-Subscription-Key e di utilizzare una delle due chiavi presenti nella sezione Profilo (sottoscrizioni) che trovate nel Portale di gestione del Servizio.


```
{
   "statusCode": 401,
   "message": "Access denied due to missing subscription key. Make sure to
include subscription key when making requests to an API."
}
```


In questo caso è mancante il valore della chiave Ocp-Apim-Subscription-Key nell'header della richiesta. Utilizzare una delle due chiavi presenti nella sezione Profilo (sottoscrizioni) che trovate nel Portale di gestione del Servizio.

- Errore 403:


```
{
   "detail": "You are not allowed to issue requests for the recipient.",
   "status": 403,
   "title": "Recipient forbidden"
}
```

Accertarsi di aver inserito un codice fiscale valido o presente nel test

```
{
   "detail": "You do not have enough permission to complete the operation you
requested",
   "status": 403,
   "title": "You are not allowed here"
}
```

Accertarsi di aver inserito un IP valido nella LISTA IP di origine autorizzati del servizio.

- Errore 404:

```
{
   "statusCode": 404,
   "message": "Resource not found"
}
```

Accertarsi di aver scritto correttamente il path della richiesta, es: `https://api.io.pagopa.it/api/v1/profiles`

FAQ

1. Posso cancellare un servizio?

No, al momento non è possibile cancellare un servizio. Rinominare il servizio aggiungendo nel **service_name** il prefisso **DELETED_**.

2. Posso visualizzare un servizio di test ancora non visibile in App IO?

Si, attraverso la procedura descritta in Test visualizzazione in App IO del servizio.

3. Posso inviare i messaggi al mio codice fiscale?

Si, se sei uno sviluppatore puoi richiedere di essere abilitato all'invio di messaggi al proprio codice fiscale attraverso la procedura descritta in Abilitazione invio messaggi a codici fiscali test.

4. Perchè ci sono due api-key per servizio?

Chiave primaria e secondaria sono equivalenti e sono duplicate per poter effettuare il cambio delle chiavi senza interruzione del servizio.

5. E' possibile pagare avvisi pagoPA in App IO?

Si, l'App IO si interfaccia direttamente al sistema di pagamento pagoPA. Nella sezione Portafoglio saranno visualizzate solo le operazioni di pagamento effettuate in App IO.

6. Cosa succede se l'ente creditore modifica l'importo dell'avviso pagoPA?

L'App IO è collegata al sistema di pagamento pagoPA e prima di avviare il flusso di pagamento in app verifica ed attualizza l'importo dell'avviso di pagamento.