Link.it

PDND Interoperabilità - API CORE: confronto v2/v3

Sintesi

Il 18 marzo 2026 PagoPA ha pubblicato la versione 3 delle API Core di PDND Interoperabilità, deprecando contestualmente la v2. La v3 introduce un livello di sicurezza superiore grazie all'adozione della specifica DPoP (RFC 9449), amplia le operazioni disponibili in modalità machine-to-machine e consolida la struttura della specifica OpenAPI.

Aspetto		v2 (deprecated)	v3 (corrente)
Autenticazione		Bearer token standard	DPoP (RFC 9449) — token sender-constrained
URL base		api.interop.pagopa.it/v2	api.interop.pagopa.it/v3
Endpoint totali		193	205 (+12 nuovi)
Endpoint rimossi		—	Nessuno (piena retrocompatibilità funzionale)
Gestione chiavi M2M		Solo da front office	Disponibile via API
Gestione utenti client M2M		Solo da front office	Disponibile via API
Nuovo tag users		Assente	Presente (GET /users, GET /users/{id})
Header di risposta		Rate limit only	+ Digest + Agid-JWT-Signature (INTEGRITY)
Paginazione	⚙	Parametri inline per endpoint	Parametri centralizzati in components
Schemi ID tipizzati	⚙	Assenti	Presenti (AgreementId, ClientId, ecc.)

⚙ Dettaglio tecnico interno alla specifica OpenAPI — nessun impatto comportamentale per i client già integrati.

Stato versioni: la v3 è la versione corrente e attiva in produzione su https://api.interop.pagopa.it/v3. La v2 rimane accessibile su https://api.interop.pagopa.it/v2 ma è ufficialmente deprecata. Sono ancora attesi un secondo rilascio (creazione client e portachiavi erogatore) e un terzo rilascio (analisi del rischio e template e-service).

1. Autenticazione — da Bearer a DPoP

La modifica più rilevante introdotta dalla v3 riguarda il meccanismo di autenticazione. La v2 adottava il classico schema Bearer (RFC 6750): il client negoziava un access token presso il server di autorizzazione e lo allegava a ogni richiesta nell'header Authorization. La v3 adotta invece DPoP (Demonstrating Proof of Possession, RFC 9449), uno schema che vincola il token alla chiave crittografica del client che lo ha ottenuto.

1.1. Meccanismo DPoP

Con DPoP il client deve compiere i seguenti passi per ogni richiesta:

- Generare (o riutilizzare) una coppia di chiavi asimmetriche DPoP (tipicamente EC o RSA).
- Costruire una DPoP Proof JWT firmata con la propria chiave privata, contenente: metodo HTTP (htm), URI canonico della richiesta (htu), timestamp (iat), jti univoco per prevenire il replay, e — quando richiesto dal server — il nonce fornito dal server.
- Inviare la richiesta con due elementi nell'intestazione HTTP: Authorization: DPoP <access_token> e DPoP: <proof_jwt>.
- Ripetere la generazione della DPoP Proof per ogni chiamata (il proof è strettamente legato a URI e metodo).

Schema	v2	v3
Tipo	HTTP Bearer	HTTP DPoP + API Key (header)
Header Authorization	Authorization: Bearer <token>	Authorization: DPoP <token>
Header aggiuntivo	—	DPoP: <proof_jwt>
Binding al client	No — token riusabile da chiunque	Sì — legato alla chiave privata del client
Protezione replay	Nessuna	Built-in (nonce, htm, htu, iat nel proof)
RFC di riferimento	RFC 6750	RFC 9449

1.2. Perché DPoP è più sicuro

Con Bearer, un token rubato in transito (o estratto da un log) può essere riutilizzato da un attaccante su qualsiasi endpoint finché non scade. Con DPoP il token è sender-constrained: anche possedendo il token, un attaccante non può usarlo senza la chiave privata corrispondente. Il campo cnf (confirmation) nel payload del token contiene il thumbprint

JWK della chiave pubblica DPoP, e il server verifica che la proof allegata alla richiesta sia firmata con la chiave corrispondente.

1.3. Impatto sulla richiesta di voucher

L'access token DPoP-bound viene ottenuto dallo stesso endpoint di negoziazione del voucher PDND (auth.interop.pagopa.it/token.oauth2). Il client deve presentare la DPoP Proof anche durante la fase di token request, in modo che il server di autorizzazione possa legare il token alla chiave pubblica DPoP del client. Il campo cnf.jkt nel token risultante conterrà il thumbprint della chiave.

2. URL base

Il percorso di base dell'API è cambiato in coerenza con il versioning della piattaforma:

Versione	URL base
v2	https://api.interop.pagopa.it/v2
v3	https://api.interop.pagopa.it/v3

ℹ️ Il percorso /v2 e /v3 è parte integrante dell'URL: tutti i client che fruiscono delle API devono aggiornare il connettore. Non è previsto alcun redirect automatico dalla v2 alla v3.

3. Nuovi endpoint

La v3 porta il totale degli endpoint da 193 a 205, aggiungendo 12 operazioni distribuite su tre aree funzionali. Nessun endpoint esistente è stato rimosso: i 193 endpoint della v2 sono presenti e funzionalmente compatibili in v3.

Metodo	Path	operationId	Tag
POST	/clients/{clientId}/keys	createClientKey	clients
DELETE	/clients/{clientId}/keys/{keyId}	deleteClientKeyById	clients
GET	/clients/{clientId}/users	getClientUsers	clients
POST	/clients/{clientId}/users	addClientUser	clients
DELETE	/clients/{clientId}/users/{userId}	removeClientUser	clients
POST	/producerKeychains/{id}/keys	createProducerKeychainKey	producerKeychains
DELETE	/producerKeychains/{id}/keys/{keyId}	deleteProducerKeychainKeyById	producerKeychains

Metodo	Path	operationId	Tag
GET	/producerKeychains/{id}/users	getProducerKeychainUsers	producerKeychains
POST	/producerKeychains/{id}/users	addProducerKeychainUser	producerKeychains
DELETE	/producerKeychains/{id}/users/{userId}	removeProducerKeychainUser	producerKeychains
GET	/users	getUsers	users
GET	/users/{userId}	getUser	users

3.1. Gestione chiavi dei client (tag: clients)

Nelle API v2 le operazioni di aggiunta e rimozione delle chiavi pubbliche sui client erano accessibili esclusivamente tramite il front office della piattaforma. La v3 espone quattro endpoint che rendono queste operazioni automatizzabili in M2M:

- POST /clients/{clientId}/keys — carica una nuova chiave pubblica su un client, fornendo nome, PEM base64 della chiave, algoritmo e uso (sig/enc) tramite il payload KeySeed.
- DELETE /clients/{clientId}/keys/{keyId} — rimuove una chiave pubblica dal client identificata dal kid.
- GET /clients/{clientId}/users — elenca gli utenti (operatori di sicurezza) associati al client.
- POST /clients/{clientId}/users — associa un utente al client tramite il payload LinkUser.
- DELETE /clients/{clientId}/users/{userId} — rimuove l'associazione tra un utente e il client.

3.2. Gestione chiavi dei portachiavi erogatore (tag: producerKeychains)

Simmetrica rispetto alla gestione dei client, la v3 introduce gli stessi cinque endpoint per i portachiavi degli erogatori (ProducerKeychain), abilitando la gestione programmatica delle chiavi pubbliche lato erogatore:

- POST /producerKeychains/{keychainId}/keys — carica una chiave pubblica nel portachiavi erogatore.
- DELETE /producerKeychains/{keychainId}/keys/{keyId} — rimuove una chiave dal portachiavi.
- GET /producerKeychains/{keychainId}/users — elenca gli utenti associati al portachiavi.
- POST /producerKeychains/{keychainId}/users — associa un utente al portachiavi.
- DELETE /producerKeychains/{keychainId}/users/{userId} — dissocia un utente dal

portachiavi.

3.3. Nuovo tag users

Viene introdotto un tag users completamente nuovo, con due endpoint di sola lettura che permettono di interrogare il registro degli utenti della piattaforma:

- GET /users — restituisce la lista degli utenti, con possibilità di filtraggio.
- GET /users/{userId} — restituisce il dettaglio di un singolo utente (userId, nome, cognome, ruoli).

ℹ️ Per le operazioni in scrittura (POST, DELETE) è richiesta la nomina di un amministratore dell'ente come responsabile amministrativo del client API Interoperabilità. Le operazioni di sola lettura (GET) non richiedono adempimenti aggiuntivi.

4. Header di risposta — pattern INTEGRITY_REST_02

In v3 tutte le risposte HTTP — incluse quelle di errore (400, 401, 403, 404, 409, 429) — includono due nuovi header che implementano il pattern di integrità del messaggio INTEGRITY_REST_02 definito dalle Linee Guida AgID ModI:

Header	Disponibile in	Descrizione
X-Rate-Limit-Limit	v2 e v3	Numero massimo di richieste consentite nell'intervallo
X-Rate-Limit-Remaining	v2 e v3	Richieste rimanenti nell'intervallo corrente
X-Rate-Limit-Interval	v2 e v3	Durata dell'intervallo in millisecondi
Digest	**solo v3**	Hash SHA-256 del body di risposta (pattern: SHA-256=<base64>=)
Agid-JWT-Signature	**solo v3**	JWT firmato con la chiave M2M del gateway; contiene digest, content-type e opzionalmente content-encoding

4.1. Header Digest

L'header Digest contiene l'hash SHA-256 del corpo della risposta, codificato in Base64 secondo RFC 3230, nel formato:

```
Digest: SHA-256=<43 caratteri base64>=
```

Il pattern usato nella specifica è ^SHA-256=[a-zA-Z0-9+/]{43}=$. Questo consente al client di verificare che il corpo ricevuto non sia stato alterato in transito.

4.2. Header AgID-JWT-Signature

L'header Agid-JWT-Signature è un JWT firmato con la chiave privata M2M del gateway PDND. Il payload del JWT include un campo signed_headers contenente il valore del Digest, il Content-Type e — opzionalmente — il Content-Encoding della risposta. La struttura è quella di un JWT compatto (header.payload.signature) con pattern:

```
Agid-JWT-Signature: <base64url>.<base64url>.<base64url>
```

La presenza di questo header consente al client di verificare l'autenticità della risposta oltre alla sua integrità, dal momento che la firma è prodotta dal gateway con la propria chiave privata.

5. Paginazione – refactoring dei parametri

La v3 introduce una ristrutturazione formale dei parametri di paginazione, senza modificarne il comportamento. In v2 i parametri offset e limit erano dichiarati inline all'interno di ogni singolo endpoint (con la conseguente duplicazione della definizione per tutti i 40 endpoint paginati). In v3 sono stati centralizzati nella sezione components/parameters e referenziati tramite $ref.

Parametro	Valore	Note
offset	integer, format: int32, minimum: 0	Identico in v2 e v3
limit	integer, format: int32, min: 1, max: 50	Identico in v2 e v3; in v3 è required tramite componente condiviso

Oltre a offset e limit, la v3 centralizza anche altri parametri riutilizzati su più endpoint: EserviceIdsParam, TemplateIdsParam, DelegatorIdsParam, DelegateIdsParam, LastEventId, EventsLimit, EventsDelegationId. Questo migliora la leggibilità e la manutenibilità della specifica, ma non ha alcun impatto comportamentale per i client.

> ℹ I parametri di paginazione restano funzionalmente invariati. I client già integrati con la v2 non devono modificare la logica di paginazione delle chiamate, solo il base URL.

6. Nuovi schemi del modello dati

La v3 introduce 21 nuovi schemi nella sezione components/schemas. Nessuno schema esistente è stato rimosso. I modelli dati degli oggetti di dominio principali (Agreement, EService, Purpose, Tenant, Client, ecc.) sono invariati rispetto alla v2.

Schema	Categoria	Descrizione
KeySeed	Input	Payload per il caricamento di una chiave pubblica (nome, PEM base64, algoritmo, uso)
KeyUse	Enum	Enum che distingue le chiavi destinate a firma (sig) da quelle a cifratura (enc)
User	Output	Rappresentazione di un utente: userId, nome, cognome, ruoli
Users	Output	Lista di oggetti User
LinkUser	Input	Payload per associare un utente a un client o portachiavi erogatore
AgreementId	Tipo ID	String UUID — identificatore tipizzato di un Agreement
AttributeId	Tipo ID	String UUID — identificatore tipizzato di un Attribute
ClientId	Tipo ID	String UUID — identificatore tipizzato di un Client
DelegationId	Tipo ID	String UUID — identificatore tipizzato di una Delegation
DescriptorId	Tipo ID	String UUID — identificatore tipizzato di un Descriptor
EServiceId	Tipo ID	String UUID — identificatore tipizzato di un EService
EServiceTemplateId	Tipo ID	String UUID — identificatore tipizzato di un EServiceTemplate
Kid	Tipo ID	String — identificatore tipizzato di una chiave pubblica
ProducerKeychainId	Tipo ID	String UUID — identificatore tipizzato di un ProducerKeychain
PurposeId	Tipo ID	String UUID — identificatore tipizzato di una Purpose

Schema	Categoria	Descrizione
PurposeTemplateId	Tipo ID	String UUID — identificatore tipizzato di un PurposeTemplate
TenantId	Tipo ID	String UUID — identificatore tipizzato di un Tenant
DelegatorId	Tipo ID	String UUID — identificatore tipizzato di un Delegator
DelegateId	Tipo ID	String UUID — identificatore tipizzato di un Delegate
TemplateId	Tipo ID	String UUID — identificatore tipizzato di un Template
VoidObject	Output	Oggetto vuoto restituito dalle operazioni che non producono payload (es. DELETE)

6.1. Schermi funzionali (KeySeed, KeyUse, User, Users, LinkUser)

Questi schemi supportano direttamente i nuovi endpoint descritti nella sezione 3:

- KeySeed — corpo della richiesta per POST /clients/{id}/keys e POST /producerKeychains/{id}/keys. Richiede i campi: key (stringa PEM in base64), use (enum KeyUse: sig o enc), alg (algoritmo, es. RS256, ES256), name (stringa 5–60 caratteri).
- KeyUse — enumerazione con due valori: sig (chiave di firma) ed enc (chiave di cifratura/accordo di chiave). Determina il campo use della JWK registrata.
- User — oggetto restituito da GET /users e GET /users/{userId}: contiene userId (UUID), name, familyName e un array di ruoli.
- Users — array di oggetti User, restituito da GET /users.
- LinkUser — corpo della richiesta per POST /clients/{id}/users e POST /producerKeychains/{id}/users.

6.2. Identificatori tipizzati

La v3 introduce un tipo dedicato per ogni identificatore di oggetto di dominio (es. AgreementId, ClientId, EServiceId, TenantId, Kid, ecc.). Tutti sono string/uuid (o string nel caso di Kid), già utilizzati come tipo dei parametri di path e delle proprietà nei modelli esistenti. Il refactoring migliora la leggibilità della specifica e consente ai generatori di codice di produrre tipi fortemente tipizzati anziché semplici string.

6.3. VoidObject

Le operazioni che non restituiscono un payload significativo (tipicamente le DELETE e alcune POST) rispondono ora con uno schema esplicito VoidObject — un oggetto vuoto con additionalProperties: false — anziché con un body assente o non documentato. Questo rende la specifica più precisa e compatibile con i generatori di client che richiedono uno schema di risposta definito per ogni operazione.